CLASSROOM FINDER SYSTEM WITH STUDENT AVAILABILITY, SPACE AND TIME CONSTRAINT

LOH YONG BIN

UNIVERSITI TUNKU ABDUL RAHMAN

CLASSROOM FINDER SYSTEM WITH STUDENT AVAILABILITY, SPACE AND TIME CONSTRAINT

LOH YONG BIN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	yongbin
Name	:	LOH YONG BIN
ID No.	:	2200831
Date	:	17 May 2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"CLASSROOM FINDER SYSTEM WITH STUDENT AVAILABILITY, SPACE AND TIME CONSTRAINT"** was prepared by **LOH YONG BIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Associate Prof. Dr. Norazah binti Yusof
Date	:	17 May 2024

Signature	:	
Co-Supervisor	:	
Date	:	

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my supervisor, Dr. Norazah binti Yusof for her invaluable guidance, support and encouragement throughout every stage of this project. Her expertise, patience and dedication have been pivotal in shaping my understanding, refining my methodologies and navigating through challenges. Her encouragement and belief in my abilities have inspired me to strive for excellence and persevere in the face of obstacles. I am truly grateful for the invaluable lessons learned under their tutelage.

Next, I would also like to extend heartfelt appreciation to my family members for their endless love, encouragement and understanding. Their unwavering support and sacrifices have been the cornerstone of my academic journey. Their belief in my capabilities, even during moments of self-doubt, has provided me with the strength and motivation to pursue my aspirations wholeheartedly. Their constant encouragement and unwavering faith in my abilities have been a constant source of inspiration.

Lastly, I would like to extend my deepest gratitude to my friends who have been my pillars of strength and sources of joy throughout this journey. Their encouragement, camaraderie and moral support have been invaluable in sustaining my enthusiasm and resilience during challenging times. Their belief in my potential and unwavering encouragement have been instrumental in overcoming obstacles and achieving milestones. Their friendship has enriched my life in countless ways, and I am truly blessed to have them by my side.

ABSTRACT

In every educational institution, the process of conducting student assessments is very important to evaluate students' academic progress, understanding and performance. A crucial aspect of this process is the allocation of suitable rooms for the academic activities. However, the traditional approach of manually allocating rooms is labour-intensive and prone to errors, particularly when considering constraints such as student availability, space requirements and scheduling conflicts. Therefore, an automate Classroom Finder System is developed to automate the room finding and allocation for student assessments. The primary focus of the system is on facilitating efficient and conflict-free room assignments while considering various constraints and requirements by exploring the suitable Artificial Intelligence approaches and scheduling algorithms. Therefore, the project scopes are to identify the specific constraints and requirements to be considered in the Classroom Finder System and also explore the suitable Artificial Intelligence approaches and room allocation and scheduling algorithms. The project is also aimed to implement real-time management, communication and updates of the assessment and room details to provide instant updates and notifications. Scrum methodology was selected for the development methodology of this project while the tools of HTML, CSS, JavaScript, Bootstrap, Jinja2 Template Engine, Flask, SQLite and VS Code are used in the development of this project. In the end of project, the unit testing, user acceptance testing, and usability testing has also been conducted to test the system. In conclusion, all the objectives are achieved, and limitations are analysed while recommendations for future enhancements are discussed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
AJAX	Asynchronous JavaScript And XML
CRUD	Create, Read, Update, Delete
CSP	Constraint Satisfaction Problem
CSS	Cascading Style Sheets
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
FYP	Final Year Project
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
RTM	Requirements Traceability Matrix
SDLC	Software Development Life Cycle
SQL	Structured Query Language
SUS	System Usability Scale
UAT	User Acceptance Test
UI	User Interface
UML	Unified Modelling Language
WBS	Work Breakdown Structure
SUS	System Usability Scale
UAT	User Acceptance Test

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

In every educational institution, the process of conducting student assessments such as quizzes and tests is very important as it serves as a fundamental tool to evaluate students' academic progress, understanding and performance (Fuentealba, 2011). A crucial aspect of this process is the allocation of suitable rooms where students can undertake their academic activities and assessment. However, the traditional approach of manually allocating rooms for these assessment poses significant challenges and limitation. It is often resulting in a time-consuming and labour-intensive process that demands meticulous coordination and effort from lecturers and administrators.

One of the primary challenges encountered in the traditional approach is the difficulty in finding suitable rooms that meet the specific constraints and requirements. Student availability, space requirements and time limitations are all critical factors that must be taken into consideration when allocating suitable rooms for assessments (Norman, et al., 1996) (Deris, et al., 1997) (Carter, et al., 1994). It can be a challenging task to ensure that assessments are scheduled at times while all students can participate without conflicting with other academic activities. As the student population continues to expand, the complexities of room allocation process also have escalated. Moreover, finding rooms that are adequately sized to accommodate the number of students taking the assessment is vital for promoting a conducive and comfortable testing environment. There is also another factor which is needed to be considered when allocating suitable rooms which is equipment constraint. The process of finding and allocating rooms which have adequate equipment is also important to improve the student assessment process.

Furthermore, the growing diversity of course offerings and academic programs introduces further complexity into the room finding and allocation process (Murray, et al., 2007). Different courses may have varying requirements such as the need for specialized equipment or technology which must be considered when allocating rooms (Carter, et al., 1994). If there is a failure to meet these specific needs, it can hinder the effectiveness of assessments and compromise the overall educational experience. After room finding and room allocation process, there is a need of manual communication to discuss the assessment details such as the space requirement and time limitation. After that, there is also a need to discuss the student availability by asking every student manually. Once there is the assigned room, date and time for the assessment, the lecturers need to inform the students for their assessment details manually. If there is any changes or updates to the student assessment details, there is a need for administrators to inform the lecturers while the lecturers need to inform the students. This can lead to inconvenience and time-consuming especially when the students keep increasing.

In order to tackle these challenges, this project is initiated to analyse the issues and challenges faced in room finding and allocation for student assessments and propose solutions regarding to the issues and limitations found. This project will also be discussing the background of the problem, problem statements, project objectives, proposed solution, proposed approach as well as the project scope.

1.2 Background of the problem

Traditionally, the process of finding and allocating the rooms for academic activities is very time-consuming and labour-intensive. The process involves human intervention and decision-making without the assistance of automated systems or any room allocation and scheduling algorithms.

During the traditional room allocation process, the lecturers and administrators need to plan and conduct the student assessment such as quizzes, tests or other academic activities for their respective courses to assess the students' performance (Fuentealba, 2011). They have often found themselves facing difficulties in finding and allocating appropriate room that align with specific constraints. They need to consider the constraints such as student availability, space requirement, time limitation and any special requirements such as the need for specific equipment or technology and these constraints have increased the complexity to the room finding and allocation process (Murray, et al., 2007) (Carter, et al., 1994). It is the most challenging part in traditional process to consider the student availability because it needs to check every student's timetable so that every student can conduct the assessments in the available time. As the number of students keep increasing, it also led to a more complex and time-consuming room allocation process.

After that, the administrators responsible for room finding and allocation checks the availability of various room for the chosen date and time. This might involve consulting a physical schedule or a calendar to see which rooms are free during the intended time slot. After that, the administrators also need to evaluate the room capacity manually to ensure that it can accommodate the expected number of students participating in the assessment as overcrowding the room may negatively impact the students' experience and performance. If there is not any room which can completely accommodate all the students, then the administrators would need to manually calculate the capacity and combine the room to fit all the students to participate in the assessment. This process will lead to inefficient resources utilization.

Once the suitable room is identified, the administrators reserve the room for the chosen date and time slot. This could involve making notes on a physical schedule or filling out reservation forms. There are also high possibilities of scheduling conflicts where multiple academic activities require the same room at the same time. If there are the scheduling conflicts, the administrators must manually find alternative arrangements which is allocating the room for the same academic activity again and it will cause repeating works for the room finding and allocation task.

Furthermore, the lecturers need to manually communicate the assessment details such as the space requirement and time limitation to the respective administrators and discussing the student availability with every student. Lecturers also need to inform their students for the assigned room, date and time for their assessments manually. If there is any changes or updates to the assessment schedule, it also needs to update the lecturers and students manually. This can lead to inconvenience for lectures, administrators and students. It is essential to note that the manual room allocation process can become very complex and time-consuming as the number of academic activities, courses, and students keep increasing. Additionally, manual processes are more susceptible to errors such as double bookings which can lead to scheduling conflicts, disruptions and inconveniences for students and lecturers and inefficient resource utilization.

Thus, Classroom Finder System which an innovative and technology-driven solution can revolutionize the room finding and allocation process by harnessing the power of artificial intelligence approaches and scheduling algorithms. While the Classroom Finder System can automate the room finding and allocation for academic activities, optimize the resource utilization, minimize the scheduling conflicts, empowering lecturers to focus on delivering quality education and students to engage in a conducive learning environment.

1.3 Problem Statement

There are three main problems that faced by most of the educational institutions during the room finding and allocation for student assessments. The three main problem statements are inefficient and complex manual room allotment and management, multiple constraints and scheduling conflicts and manual communication and updates.

1.3.1 Inefficient and Complex Manual Room Allotment and Management

The current process of room finding and allocation for academic activities in educational institutions lacks automation and relies heavily on manual methods while making the process become a time-consuming and labour-intensive task and resulting in inefficiencies and complexities (Rane, et al., 2021). Lecturers and administrators spend considerable time finding and identifying suitable rooms for student assessments which involves planning assessment, checking student availability, checking room availability, assessing room capacity, resolving conflicts and manual communication for updates.

The manual process becomes even more inefficient and complex when considering various constraints such as the student availability, space requirements, time limitations and special requirements. With the increasing number of students, the process of manually checking their timetables and finding suitable times slot for assessment becomes increasingly difficult. The process of checking room availability across numerous schedules can be time-consuming and led to delays in finalizing room allotments. The manual room management process also becomes complex when the number of rooms in institutions increases. This also making it harder to maintain the room availability through physical schedules or calendars. Furthermore, the process of manually assessing room capacity also becomes challenging due to the large number of rooms in institutions and combining rooms to accommodate all students becomes a complex process. It is leading to underutilization of resources. When reserving the rooms manually, there may also result in conflict if multiple reservations are made for the same room during or after the room availability checking.

Since the manual room allotment and management process is plagued by inefficiencies and complexities, there is indeed an increased difficulties during the process. The time-consuming nature of the manual room allotment process also leaves little room for lecturers and administrators to focus on their core responsibilities. Hence, with the classroom finder system, lecturers and administrators can effortlessly plan the assessments. The system can also streamline the room allocation process to check the room availability efficiently and assess the room capacities accurately to ensure optimal allocation. By eliminating the complexities of manual room allotment and management, the Classroom Finder System empowers educational institutions to optimize their operations and enhance the productivity.

1.3.2 Multiple Constraints and Scheduling Conflicts

The manual room finding and allocation process for student assessments in educational institutions faces challenges due to multiple constraints and the potential for scheduling conflicts. Lecturers and administrators must carefully plan the assessment and allocate the room while considering various constraints including the student availability, space requirements, time limitations and other special requirements such as equipment constraint (Carter, et al., 1994).

For the student availability, students often have overlapping schedules and enrolled in multiple courses. Therefore, it is difficult to find suitable time slots for assessments where all required students can attend. However, it is extremely challenging and practically impossible to find suitable time slots for assessments according to student availability through the manual process due to complexity of comparing individual student timetables. This constraint may also lead to decreased student participation and potentially unfair evaluations. For the space requirement, different assessments may require varying space capacities based on classroom size or specific assessment needs. Therefore, the space requirement must be considered during the process since allocating rooms without considering the requirements may result in overcrowded or underutilized spaces. For the time limitations, it is important to schedule the assessment efficiently within the available time slots. It is because it can significantly avoid overlaps and ensure all the assessments can be completed on time. Manual scheduling may always lead to time conflicts and require to rescheduling and disrupting the academic calendar. For the special requirements such as equipment requirements, some assessments may require the special equipment requirements such as computer or other equipment. Therefore, it is important to find the room which consists of the specific equipment for the assessments.

However, balancing and handling these constraints manually is complex, time-consuming and prone to errors especially when the number of course, students and rooms increases. The administrators or lecturers are difficult and impossible to find the most suitable room while considering the constraints manually. This manual allocation based on the multiple constraints often result in inefficient and suboptimal room allocation, underutilized resource and scheduling conflicts in the end.

During the manual assessment scheduling process, it may also result in scheduling conflicts when there are multiple assessments scheduled simultaneously or overlapping in the same room (Rane, et al., 2021). Additionally, there is also the conflicts when assessment time slot is clashing with the students' timetable. These conflicts can be attributed to the absence of efficient room allocation and scheduling algorithms which will lead to suboptimal resource utilization with some rooms being underutilized while others are overbooked. While resolving these conflicts manually requires the administrators to find alternative arrangements which can be time-consuming. As a result, these conflicts can negatively impact the assessment process and cause inconvenience to both lecturers and students. These will ultimately affect the overall efficiency of the scheduling process.

1.3.3 Manual Communication and Updates

The existing manual room allocation process in educational institutions relies on communication between administrators, lecturers and students and it may lead to inefficiencies and delays. Manual communication and updates are prone to errors and resulting in miscommunication, confusion and scheduling conflicts. In the current process, administrators and lecturers must manually communicate the assessment details with each other to allocate and reserve rooms for the assessments. This involves exchanging information through emails, forms or physical schedules which can be time-consuming and increases the chances of miscommunication. However, miscommunication can result in incorrect room finding and assignments and lead to confusion for students and lecturers.

Furthermore, manual updates can lead to delays in conveying important information such as changes in room allocations or assessment schedules. As a result, lecturers and students may not receive timely notifications about room or assessment schedules changes or updates. This can cause inconvenience and confusion among the lecturers and students. The lack of real-time updates can also lead to scheduling conflicts when multiple lecturers attempt to reserve the same room simultaneously. Without instant updates, lecturers may unknowingly book the same room for different academic activities and lead to disruptions to the academic activities and conflicts (Rane, et al., 2021).

The manual communication process places an additional administrative burden on administrators and lecturers as they need to spend time and effort on managing room allocation updates and coordinating with one another. In a manual system, lecturers or administrators may not have real-time visibility into room availability or changes in the schedule. This lack of visibility can make it challenging to make informed decisions and optimize room allocation efficiently. In dynamic educational environment, room availability may change frequently due to unexpected events or adjustments in the academic schedule. Without real-time updates in the room allocation process, lecturers may encounter difficulties in securing suitable rooms for assessments at the desired time slots. When the lecturers want to change the assessment timeslot and the room, it is also difficult to make changes and inform the students about the changes.

To overcome these challenges, the proposed Classroom Finder System aims to provide real-time updates and online communication. The system can instantly update room availability, assess conflicts and notify users when there are changes in room allocations. The system can also allow lecturers and administrators to communicate online to discuss anything about room allocation. This streamlined communication process will improve efficiency, reduce scheduling conflicts and enhance the overall room allocation experience for administrators, lecturers and students.

1.4 Project Objectives

The objectives to be achieved in this project are as follow:

1. To identify the specific constraints and requirements to be considered in the Classroom Finder System.
2. To explore the suitable Artificial Intelligence approaches and room allocation and scheduling algorithms.
3. To implement real-time management, communication and updates of the assessment and room details to provide instant updates and notifications
4. To develop the automate Classroom Finder System that employs the identified Artificial Intelligent approaches for assigning suitable rooms based on the identified constraints.

1.5 Project Solution

There are a few problems and inefficiencies associated with the current manual room allocation process which are inefficient and complex manual room allotment and management, multiple constraints and scheduling conflicts and manual communication and updates. Before the system had been developed, the thoroughly identification and understanding of the specific constraints and requirements involved had been made. This includes considerations such as student availability, space requirements, time limitations and any special equipment or technology needs. The comprehensive analysis of these constraints served as the foundation in developing the effective solution within the system.

By addressing the challenges outlined in the problem statements, an intelligent and automated Classroom Finder System was developed to revolutionize traditional room allocation processes for student assessments (Yepuri et al., 2018). This system aimed to effectively solve the scheduling problem by allocating suitable rooms for various academic activities and it designed for administrator, lecturers and students. The Classroom Finder System was able to manage the lecturer and student information which includes create, read, update and delete operations. Additionally, it facilitated efficient management of rooms, courses, course timetables and course enrolments. These features were crucial components of the room allocation process as they enabled the system to construct student timetables based on course schedules and enrolment data which can address the constraints related to student availability. Furthermore, the system incorporated room information such as capacity and equipment availability to effectively handle equipment and space constraints during the allocation process. By encompassing these functionalities, the Classroom Finder System streamlined room allocation procedures and significantly enhanced the overall efficiency of academic activities within educational institutions. The system also conducted real-time data integration as it integrated seamlessly with student timetables, room capacity and academic schedules to ensure the accurate and up-to-date allocation decisions.

Besides, the Classroom Finder System leveraged the cutting-edge technologies such as Artificial Intelligence and advanced scheduling algorithms to revolutionize the way rooms are allocated for student assessments while optimizing resource utilization and enhancing the overall academic experience. The analysis of Artificial Intelligence approaches and algorithms such as Constraint Satisfaction Problem (CSP), Genetic Algorithm (GA) and Automated Planning also had been made. After careful consideration, suitable Artificial Intelligence approach which was Constraint Satisfaction Problem was selected and implemented in the system. Besides, the appropriate room allocation and scheduling algorithms which was Backtracking algorithm was also selected and implemented in the system to solve the CSPs. The algorithm was systematically searching through the possible solutions and backtracking when a dead end is reached. It is a general-purpose algorithm often employed in conjunction with CSPs to efficiently explore the solution space and find valid assignments that satisfy the constraints. Therefore, the Classroom Finder System was served as an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessments. By incorporating AI approaches and scheduling algorithms, the system automated room allocation while considering the constraints such as student availability, space requirements, time limitations and equipment requirements. The AI approaches and scheduling algorithm was intelligently analysing these constraints to suggest optimal room assignments for assessments. It also able to reduce the manual effort required in the traditional allocation process significantly.

The system also provided a user-friendly interface for lecturers, administrators and students to interact with the system. The lectures and administrators were able to view the slot and room availability for the assessment after inputting the assessment details, preferences and constraints through an intuitive interface. It resulted in conflict-free room assignments with optimal resource utilization. After the room and assessment had been allocated and created, there was a real-time update and notification sent to the lecturers and students who involved in the assessment through email. Not only assessment creation, any changes included updating and deleting the assessment will also provide real-time notifications to the users. This can enhance the notification and reduce the uncertainty between the users. The lecturers and students can also access the system to view the assessment schedules which in the list or in the form of calendar (Rane, et al., 2021).

In conclusion, the Classroom Finder System aimed to optimize the room allocation process which led to increased efficiency, reduced scheduling conflicts and improved resource utilization. Lecturers and administrators were empowered to allocate rooms effortlessly while students will benefit from a streamlined assessment experience. The Classroom Finder System offered a comprehensive and innovative solution to the challenges posed by manual room allotment and management. By harnessing the power of AI algorithms, real-time updates and user-friendly design, this system elevated the efficiency, accuracy and overall effectiveness of room allocation in educational institutions.

1.6 Project Approach

The software development process is the structured approach to develop software for a system or project while sometimes also known as the Software Development Life Cycle (SDLC).

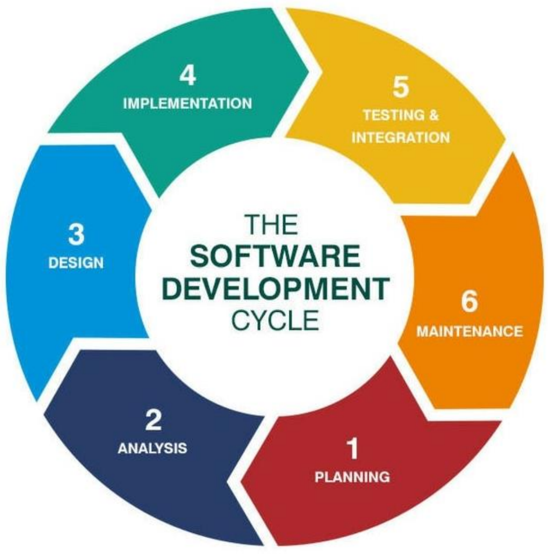


Figure 1.1: Software Development Life Cycle (SDLC)

SDLC is a sequence of steps which take place during the development of a piece of software (Luenendonk, 2020). It consists of several stages which are planning, analysis, design, implementation, testing and integration and maintenance (Hamad, 2020). There are several software development approaches are available which are waterfall approach, spiral approach, agile approach, incremental approach, rapid approach and so on. All approaches or methodologies have different strengths and weaknesses, and they play a crucial role in contributing to a successful project. Therefore, it is important to select a suitable software development approach. In this case, agile approach was selected as the project development approach or methodology.

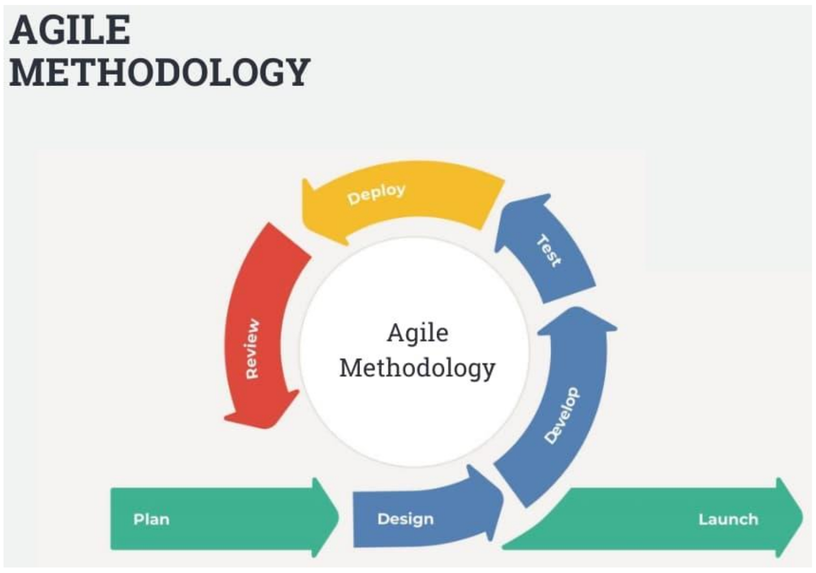


Figure 1.2: Agile Methodology

Given the project's dynamic nature and evolving requirements, Agile was chosen as the development methodology. It is a project management framework which breaks project down into several dynamic phases known as sprints (Laoyan, 2022). It is also a modern software development methodology that emphasized iterative and collaborative approaches to project management and delivery. It is built on the principles of flexibility, customer involvement and continuous improvement. Agile methodologies aim to address the challenges of rapidly changing requirements and evolving user needs and ensure continuous improvement throughout the development lifecycle. In the context of the Classroom Finder System project, Agile provided an effective framework to manage the complexity of room allocation and scheduling.

The Agile framework is an umbrella for several different variations. There are a few types of the Agile methodologies which are Scrum, Kanban, Extreme Programming and so on. For the development of the Classroom Finder System, Scrum methodology was adopted as the project management methodology as it emphasized on short, time-boxed iterations known as sprints which can facilitate iterative and incremental development, frequent feedback and close collaboration among team members which can ensure the transparency, accountability and stakeholder satisfaction. This can manage the complexity of the project effectively and deliver a valuable solution.

The development process was organized into three iterative sprints with each focused on delivering specific features or functionalities. The three iterative sprints are student, lecturer, room, course, course timetable, course enrolment and assessment management, AI-powered room allocation and real-time update, notification and communication.

Sprint 1: Student, Lecturer, Room, Course, Course Timetable, Course Enrolment and Assessment Management

The first sprint focused on building the foundation model and groundwork for the Classroom Finder System. This included the process of defining the core data structures that used to manage the student, lecturer, room, course, course timetable, course enrolment and assessment. Once the data structures were defined, the database tables were created to ensure the system can create, read, update and delete the database record as needed. Additionally, a user-friendly web-based interface was developed to facilitate seamless interaction and data input by lecturers and administrators.

Sprint 2: AI-Powered Room Allocation

In the second sprint, the focus shifted to implementing AI-powered room allocation functionality. By leveraging the Constraint Satisfaction Problem (CSP) approach, the system was designed to intelligently allocate rooms based on various constraints such as student availability, space requirements, time limitation and equipment requirements. In this sprint, it also involved the design and implementation of the core logic for the AI-powered room allocation.

Sprint 3: Real-Time Update, Notification and Communication

During the final sprint, it focused on the implementation of the features of real-time update, notification and communication. This involved implementing mechanisms for instant notifications of room allocations, assessment scheduling changes and other relevant updates to the users. Not only this, but the system also enabled real-time communication between lecturers and administrators which can foster a greater collaboration and transparency to enhance the overall user experience and satisfaction.

Throughout the development process, the Agile principles of continuous improvement and adaptation were embraced. Regular review meetings and retrospectives were conducted to assess progress, identify challenges and make necessary adjustments to ensure the successful delivery of the Classroom Finder System. By adopting an Agile approach and leveraging the Scrum methodology, it aims to deliver a robust, user-friendly and efficient Classroom Finder System that meets the needs of lecturers, administrators and students alike. Through iterative development cycles and close collaboration, the system was able to evolve to address emerging requirements and deliver maximum value to its users.

1.7 Project Scope

The project aimed to develop a web-based application called Classroom Finder System, which designed to automate and optimize the process of finding and allocating rooms for student assessments within educational institutions. The primary focus of the system is on facilitating efficient and conflict-free room assignments while considering various constraints and requirements by exploring the suitable Artificial Intelligence approaches and scheduling algorithms. Therefore, the project scopes were to identify the specific constraints and requirements to be considered in the Classroom Finder System and also explore the suitable Artificial Intelligence approaches and room allocation and scheduling algorithms. The project was also aimed to implement real-time management, communication and updates of the assessment and room details to provide instant updates and notifications. In this section, the project scope was divided into two parts which are system scope and user scope.

1.7.1 System Scope

For the system scope, the project modules are defined to provide a structured overview of the different components or subsystems that constitute the entire system. The project modules are login and logout module, lecturer and student module, room module, course module, assessment module, room allocation module, real-time update, notification module, communication module and calendar module.

Login and Logout Module

In login and logout module, there are three users included in this module which are admin, lecturer and student. The users in this module can login into the system by authenticating themselves using their email and password credentials. For example, when there is a user login into the system, the system authenticates the user by using their email and password to determine which type of user is the user. Upon login, the system verifies the user’s credentials and determines their role. Based on the user's role, the system grants appropriate access permissions and redirects them to the homepage. The system also employs robust security measures such as hashing to protect user credentials during the authentication process. The users in this module can also logout so their account will be logout and cannot be accessed anymore. Therefore, their privacy can be protected well.

Lecturer and Student Module

In lecturer and student module, there is one user include in the module which is administrator. The system allows administrator to view the lecturer and student list by clicking the specific type of user navigation link in the navigation bar. In the specific type of user list page, the administrator can add the specific user by clicking the add button and entering the information of users to register. The system also allows the administrator to update and delete the user through the buttons in the action column of the user list.

Room module

In room module, there is one user include in the module which is administrator. The system allows the administrator to view the list for all the rooms in the Classroom Finder System. Administrator can also add new room when there is new room available in the educational institution. The administrator can add new book by entering all the information of the books including the id, type, capacity, require room equipment and so on. The system also allows the administrator to update and delete the specific room.

Course Module

In course module, there is one user include in the module which is administrator. The system allows the administrator to view the list for all the courses in the Classroom Finder System. The system also allows administrator to add new course when there is new course in the educational institution. The administrator can add new course by entering all the information of the course including the id, info while also allowing the administrator to update and delete the specific room. However, in course module, the system also allows to add, update and view the course timetable and course enrolment of the specific course. The administrator can manage the timetable of the course and enrolment of the course by the users.

Assessment Module

In assessment module, there are three users include in the module which are administrator, lecturer and student. For the administrator and lecturer, the system allows the administrator and lecturer to view the list for the assessments in the Classroom Finder System. The system also allows administrator and lecturer to add new assessment when there is new assessment in the educational institution. The administrator and lecturer can add new assessment by entering all the information including the id, info, course, date, start time, end time and room while also allowing to update and delete the specific assessment. But if the assessment is over, the system is not allowed to change the assessment details in the Classroom Finder System. For the student, the system only allows the student to view the assessments which the student enrolled.

Room Allocation/Find Room Module

In room allocation module, there are two users include in the module which are administrator and lecturer. This module leverages AI-powered approach which is Constraint Satisfaction Problem and augmented by the Backtracking algorithm to intelligently allocate rooms for assessments. When the users are adding and updating the assessment, the system allows users to choose the room based on the result of the Constraint Satisfaction Problem algorithm to intelligently allocate rooms for assessments based on various constraints like student availability, space requirements, time limitations and equipment requirements.

Real-Time Update, Notification Module

In real-time update and notification module, all users include in the module which are administrator, lecturer and student. The module allows real-time update and notification after any changes of the assessment details. The system will update the details and notify the assessment details to the enrolled user through the email. Therefore, the user can be notified seamlessly to avoid the any delays for the announcement. However, there are two users which can notify the users who enrolled in the assessment, which are administrator and lecturer.

The administrator and lecturer can notify the assessment users with some messages and attachment of the assessment details.

Communication Module

In communication module, there are two users include in this module which are administrator and lecturer. The system allows administrator and lecturer to communicate through the system especially when lecturer needs to communicate with administrator regarding the issues of the room allocation or assessment creation process or any other issues.

Calendar Module

In calendar module, the users included are lecturer and students. The system allows lecturer and student to view the assessment schedules which enrolled or handled by the users. The lecturer and student are able to view the assessment schedules in the view of calendar which provides a better visualization of the schedules.

1.7.2 User Scope

In this system, the user of Classroom Finder System within the educational institution are administrators, lecturers and students.

<u>Administrator</u>

Administrator is the user that had full access to all the system functionalities. Administrator had the administrative privileges to manage and maintain the users to use the system which are the lecturers and students. Not only the user management, but administrator can also manage the room which consists of the room capacity and equipment to be utilized in the consideration of space and equipment constraints. Besides, administrator can also manage the course, course timetable and course enrolment in the educational institution. This can manage the timetable of the courses and the enrolment of the user for the courses. When the student enrolled the course, the student timetable is formed to be considered in the student availability constraint which directly impacts the room finding and allocation process. Not only these, but administrator can also manage the assessment which includes the management of room allocation process by finding the slot and room which available for the assessment. The administrator not only can manage the assessment, but administrator can also view and notify the students of the assessments about the assessment details through emailing notifications. In the end, if there is any problem or issue within the Classroom Finder System, administrators can also enter the chat room of lecturers and real-time chat with the lecturers.

<u>Lecturer</u>

Lecturer is also one of the users within the system. In the Classroom Finder System, the lecturer can manage the assessment same as the privilege of administration which the lecturer can create, read, update and delete the assessment. The lecturer also can manage the room allocation process when creating and updating the assessment as the lecturers can find the suitable slot and room for the assessment. Beside than the assessment management, the lecturer can view the assessment schedule in the calendar to have a better visualization. In the end, the lecturer can also real-time communicate and chat with the administrator when the lecturer is facing any problem with the room allocation process or the Classroom Finder System.

Student

In the system, the student can also view the assessment details which they involved with the filtering of all records or past or active assessment records only. Next, the student can also view the assessment schedule in the form of calendar which help the student to view the assessment more easily.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The process of solving a real-world timetabling problem manually often requires a significant amount of computation time. A lot of research has been invested to provide automated support for human timetables. The solution approaches to timetabling problem can be broadly classified into two categories which are Operations Research (OR) which range from the use of mathematical programming to heuristics and Artificial Intelligence (AI) which include tabu search, genetic algorithms and constraint satisfaction problem approach. In this chapter, the study will be conducted to provide a detailed review on similar research works of the problem domain which include problem domain and the approaches and methods used to solve the timetabling problem. In this chapter, section 2 discusses the problem domain, section 3 presents application of OR method in solving the timetabling problem and section 4 presents the Artificial Intelligence approaches.

2.2 Problem Domain

Timetabling is a critical process of involving the allocation of events or activities to various resources such as time slots, spaces (rooms) and personnel (student, lecturer) while ensuring that all constraints are met (Deris, et al., 1997). The objective is to achieve an optimal allocation that maximizes resource utilization or minimizes constraint violations (Rane, et al., 2021). This process is integral to multiple domains which include education, healthcare, transport and sports. In this project, we are focusing on the education domain. Educational timetabling plays a significant role in ensuring the effective scheduling of student assessments. Petrovic and Burke (2004) note that educational timetabling is a major administrative activity in most universities. It encompasses two main categories which are course and exam timetabling. These timetabling problems entail the allocation of courses or exams to classrooms and timeslots which all subject to predetermined constraints (Petrovic & Burke, 2004).

Traditionally, timetabling has been approached manually which relies on heuristics and iterative resource assignments until a feasible and acceptable schedule is achieved. However, this approach is prone to human errors and inefficiencies which lead to conflicts such as overlapping resources and double bookings (Rane, et al., 2021). Not only that, due to the rapid increase in student enrollment, courses and available resources, the complexity of timetabling has escalated. This timetabling is then resulting in non-polynomial-hard or NP-hard combinatorial optimization problems which has no specific solutions or difficult to find an optimal solution because it is impossible to enumerate all nodes in such a large search space (Rane, et al., 2021; Deris, et al., 1997). The nature of this problem also renders exhaustive examination of all combinations impractical and time-consuming (El-Sakka, 2015).

The timetabling problem is characterized by a distinction between hard and soft constraints to generate conflict-free and optimized timetable. Hard constraints are non-negotiable and must be satisfied for a feasible solution. On the other hand, soft constraints are desirable but not essential while their satisfaction contributes to optimized solutions. Within the context of educational timetabling, some common hard constraints include the prohibition of individuals being allocated to multiple places simultaneously and the requirement that the total resources used in each time period do not exceed available resources. In order to achieve an optimized solution, it involves not only satisfying hard constraints but also optimizing the satisfaction of soft constraints. However, due to their volume and complexity, it is often challenging to satisfy all soft constraints. The evaluation of the quality of a feasible timetable is also depends on how effectively these soft constraints are fulfilled (Petrovic & Burke, 2004). While addressing these constraints manually, it can lead to inefficiencies, suboptimal resource utilization and wastage of valuable time for faculties and members involved (Rane, et al., 2021).

In educational institutions, timetabling extends to schedule various assessments which are quizzes and tests. These assessments can be categorized into synchronous and asynchronous assessments (Mallari, et al., 2023). Synchronous assessments involve conducting assessments simultaneously for all students while asynchronous assessments provide flexibility in terms of timing and location. Therefore, this timetabling for quiz or test is similar to exam timetabling as they fall under same category of synchronous assessment. In the context of this project, the focus is on developing a Classroom Finder System which pertains to the educational timetabling problem. The goal is to assign quizzes or tests to available resources while considering constraints related to timeslots, room space, and personnel. This project aims to provide an efficient and automated solution to the challenges inherent in the timetabling process by leveraging appropriate techniques and approaches.

2.3 Operations Research

Operations Research (OR) is a multidisciplinary field that uses mathematical models, algorithms and analytical methods to aid in decision-making and problem-solving. It has found extensive applications in various domains which include optimization, scheduling, resource allocation and logistics. In the context of educational institutions, Operation Research plays a crucial role in tackling complex timetabling problems.

In the early days of educational timetabling research, the graph colouring problem (GCP) was a common approach employed to model and solve timetabling problems. Additionally, researchers explored the use of integer linear programming (ILP) techniques for representing and solving timetabling problems (Zhang, 2005).

The GCP is a well-researched problem in graph theory. It involves assigning colours to the vertices of a graph such that adjacent vertices have distinct colours. Interestingly, the GCP can be closely related to the timetabling problem. In its simplest form, timetabling can be modelled as a GCP where nodes represent tasks, colours represent timeslots and edges indicate constraints between tasks (Zhang, 2005).

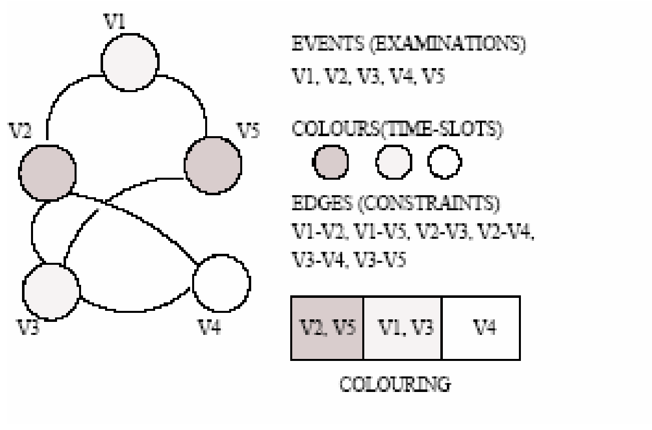


Figure 2.1: Demonstration of equivalence between the GCP

The figure 2.1 demonstrates the equivalence between the GCP and timetabling problem. The vertices in the graph represent lectures, examinations, quiz and test while the different colors represent different timeslots. The constraints are represented as edges which indicate the tasks cannot share the same timeslot. For example, V1-V2 means V1 and V2 cannot be held on the same time. But V2 and V5 have the same colour, so they can be held on the same time. The process of solving the GCP effectively addresses mathematical programming characteristics in timetabling problems (Zhang, 2005).

While the GCP approach has been employed to achieve feasible solutions for timetabling problems, its applicability has its bounds. For instance, the GCP technique works well for small-scale problems but it faces challenges in scaling up for larger timetabling instances. The real-world timetabling problems are often complex and require handling large amounts of data. Therefore, it is making the scalability of GCP-based solutions a concern (Zhang, 2005).

2.4 Artificial Intelligence

Artificial Intelligence (AI) has emerged as a pivotal approach in solving intricate and challenging problems across various domains. In educational timetabling, AI techniques have been harnessed to devise innovative solutions that optimize resource allocation, adhere to constraints and enhance the efficiency of scheduling processes.

In order to address the complexities of educational timetabling, researchers have turned to AI techniques which include metaheuristic methods and Constraint Satisfaction Problem (CSP) approach. There are various meta-heuristic methods such as simulated annealing, tabu search and genetic algorithms have been employed to tackle various educational timetabling problems along with the formalized CSP approach.

2.4.1 Genetic Algorithms

Genetic algorithms were inspired by principles of evolutionary biology which offer a powerful approach to solve optimization problems. Genetic algorithms explore solution spaces and converge towards optimal or near-optimal solutions by simulating inheritance, mutation and natural selection. These algorithms are particularly well-suited for problems with extensive search spaces. Genetic algorithms maintain many individuals of solutions in the form of a population. Individuals (parents) are chosen from the population and are then mated to form a new individual (child). The child is further mutated to introduce diversity into the population. Genetic algorithms are maintaining populations of solutions which are evolved through selection, crossover and mutation operations (Zhang, 2005).

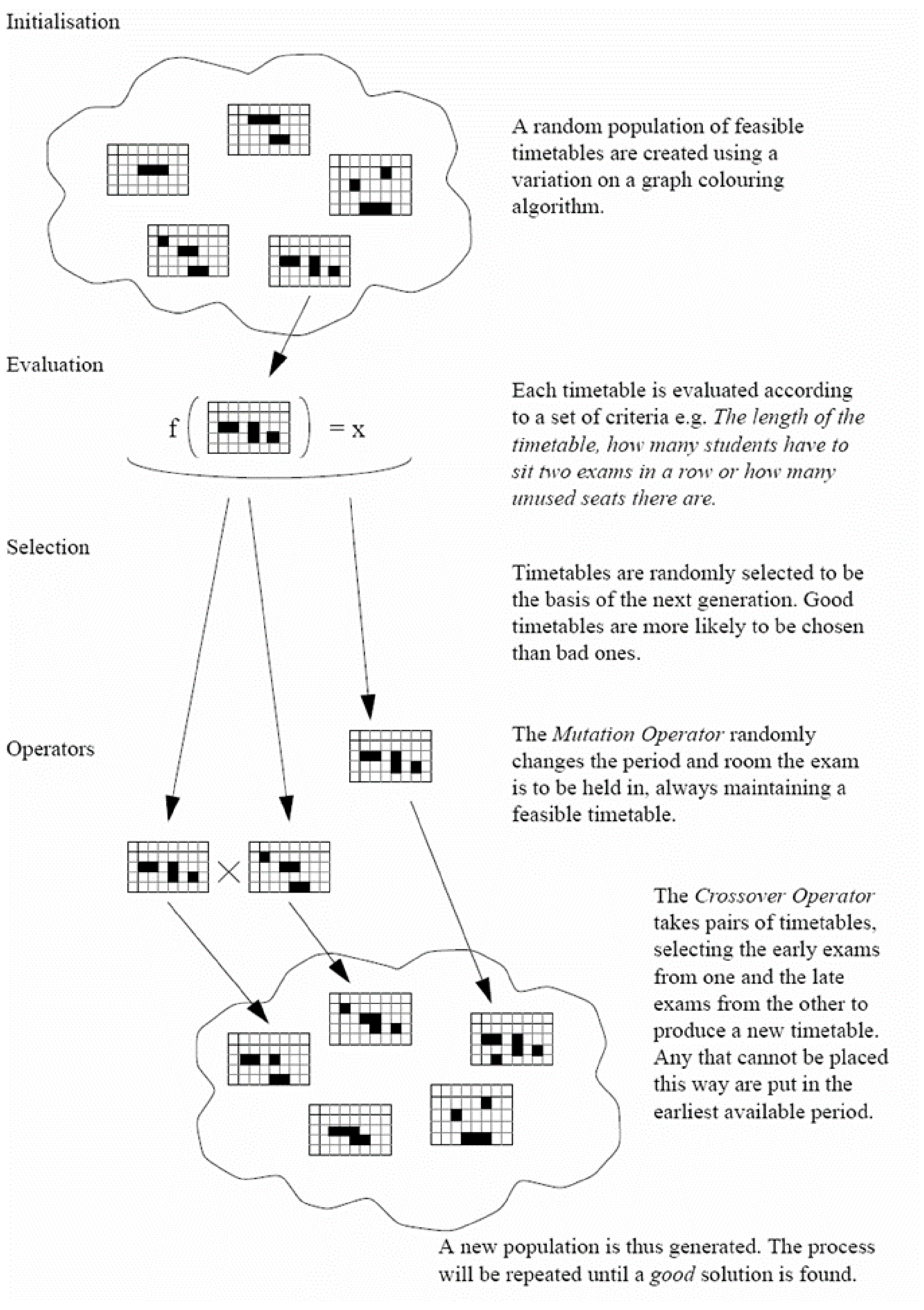

Initialisation
A random population of feasible timetables are created using a variation on a graph colouring algorithm.
Evaluation
f ( ) = x
Each timetable is evaluated according to a set of criteria e.g. *The length of the timetable, how many students have to sit two exams in a row or how many unused seats there are.*
Selection
Timetables are randomly selected to be the basis of the next generation. Good timetables are more likely to be chosen than bad ones.
Operators
The *Mutation Operator* randomly changes the period and room the exam is to be held in, always maintaining a feasible timetable.
The *Crossover Operator* takes pairs of timetables, selecting the early exams from one and the late exams from the other to produce a new timetable. Any that cannot be placed this way are put in the earliest available period.
A new population is thus generated. The process will be repeated until a *good* solution is found.

Figure 2.2: Genetic Algorithm

According to Burke, Elliman and Weare (1994), genetic algorithm initiates by creating a random collection or population of timetables. These timetables are then assessed using specific criteria. Based on this assessment, certain population members which are timetables are chosen as parents for the next generation of timetables. By giving preference to the superior timetables in the selection process, the weaker ones are discarded. This approach simultaneously guides the search towards the most favorable regions within the search space. This method is illustrated as in figure 2.2.

According to Abramson and Abela (1991), genetic algorithms are used to determine which individuals survive in the populatioin by measuring te fitness of the individual. The greater the fitness or effectiveness, the higher the chances of an individual's survival. The adaptation of genetic algorithms to timetabling involves evaluating solutions based on their fitness which reflecting how well they meet constraints and objectives.

2.4.2 Tabu Search

Tabu search is another AI-based metaheuristic method employed for solving global optimization problems. It relies on multi-level memory management and response exploration to improve solutions iteratively. It can be applied to solve the timetabling problem (Zhang, 2005). Hertz (1992) and Costa (1994) presented some specific applications to timetabling problem.

Tabu search operates as a general search algorithm for identifying the global minimum of a function *f* within a feasible set *x*. For each solution *s* in *x*, a set of feasible solutions is established which called neighborhood *N*(*s*). These solutions result from applying a simple modification *m* to *s*. The process initiates from an initial feasible solution and aims to approach a global optimal solution step by step. Notably at each stage, the algorithm replaces the current solution with a candidate solution even if conflicts arise between the candidate and the initial solution (Costa, 1994).

While applying tabu search to solve the timetabling problem, the process involves the following steps (Hertz, 1992). As we discussed earlier, the timetabling problems can be formulated as graph coloring where lecture correspond to graph vertices, time slots correspond to colors and an edge is linking two vertices if the corresponsing lectures cannot be given at the same time. Tabu search is tailored to accommodate graph coloring issues. A feasible solution is defined as a schedule adhering to a subset C of constraints and a neighboring solution is derived by modifying the schedule of a single lecture. When lectures are restricted to a single period, this modification is equivalent to change the color of a vertex. In this application, we consider two lectures which are A and B that share common students. If lecture A cannot be scheduled during period p due to a constraint, then any solution where A is placed at time p is prohibited. However, if this constraint is not present and lecture B is scheduled at time p, scheduling another lecture A at the same time is permissible even if it violates the constraint of avoiding overlapping lectures for shared students. The rationale behind this allowance is that lecture A might optimally be placed at time p and the constraint violation can be mitigated by relocating lecture B. Despite its utility, it is worth noting that tabu search can face limitations in larger problem spaces (Hertz, 1992).

2.4.3 Automated Planning

Automated planning for timetabling problems involves a systematic approach to scheduling classes, exams or events within educational institutions. The process begins with the representation of the problem where various elements are defined. The variables are introduced to represent entities like classes, instructors, rooms and time slots. Each variable has a domain specifying the feasible values it can take such as available classrooms or time periods. The constraints are defined to capture the intricate rules and limitations of the scheduling task which encompass factors like room capacity, instructor availability and student preferences.

The problem representation is then translated into a formal planning domain which set the stage for algorithmic solutions. Automated planning algorithms which include popular techniques such as STRIPS or Fast Downward are employed to search for an optimal schedule. These algorithms aim to find a sequence of scheduling actions or assignments that satisfy all constraints and objectives and lead to a feasible timetable eventually.

Upon generating a schedule, a validation step ensures it adheres to all predefined constraints. If discrepancies are detected, adjustments are made and the search process may be reinitiated to refine the schedule further. Besides, the optimization criteria such as minimizing room changes or travel time between classes can be incorporated to enhance the quality of timetable.

The final schedule is then integrated into the scheduling system which allows for seamless implementation and dissemination. The automated planning empowers educational institutions to solve complex timetabling challenges efficiently which is resulting in optimized schedules that enhance resource utilization and overall operational effectiveness.

2.4.4 Constraint Satisfaction Problem (CSP)

Constraint Satisfaction Problem is an AI technique that can be applied to solve problem by identifying constraints and finding solutions that satisfy those constraints. CSP has a wide range of applications which include scheduling, resource allocation and automated reasoning. It has found applications in the domain of university timetabling (Brittan & Farley, 1971; Jaffar & Maher, 1994).

A distinctive aspect of CSP is its capability to categorize constraints into both hard and soft constraints. In the context of scheduling, hard constraints are conditions that must be met without exceptions while soft constraints may be breached but should still be satisfied as much as possible. CSP possesses the characteristic of being NP-complete. It has been put into practice in domains like planning and scheduling which encompass problems such as the job-shop problem, car sequencing problem, vehicle routing problem and rostering problem (Zhang, 2005).

Timetabling problems is a type of assignment problems with large number of complex constraints, thus usually can be easily modelled as CSPs (Brailsford, et al., 1999). CSP is exceptionally well-suited for timetabling issues as it facilitates a more straightforward expression of the constraints. While the performance of CSP can be sensitive to minor shifts in problem formulation, the different formulation options still remain within the boundaries of logical equivalence (Geske, 1998).

2.4.4.1 Definition

Constraint satisfaction problem (CSP) deals with assignment of values from its domains to each variable such that no constraint is violated. CSP has three components which are variables, domains and constraints. For example, in the context of quiz or test timetabling, the variables represent the tasks or activities that need to be scheduled. Each variable corresponds to a particular task such as a quiz or test. The domain of a variable represents the possible values it can take. For example, the domain of each variable could represent the available time slots or rooms. The constraints define the relationships and restrictions among variables which it can represent conditions such as "Quiz A and Quiz B should not be scheduled in the same room at the same time" or "Lecturer X cannot teach two classes simultaneously." They limit the possible values that a variable can have. In general, CSP consists of a finite set of variable X with respective domains D which list the possible values for each variable D and set of constraints C.

The CSPs are an important class of combinatorial optimization problems. A solution of a CSP is a consistent assignment of all variables to values in such a way that all the constraints are satisfied. There are two approaches to solve CSP which are consistency technique and search algorithms (Zhang, 2005).

2.4.4.2 Consistency

Consistency technique is a technique which aim to reduce the search space by iteratively eliminating the values from domains that cannot satisfy the constraints. It is also a technique based on removing inconsistent values from the domains of variable until the solution is found. However, there are questions on how to identify the inconsistent and redundant values. Over the years, there is a number of consistency concepts have been developed to help in identifying the values. The consistency concepts are defined in such a way that if the presence of a value in a domain, then it can be deduced to be redundant (Dechter, 2003). The one of the popular consistency techniques is arc-consistency where for each binary constraints, values in the domains of each individual variables are removed if they cannot participate in a valid assignment so there exists a consistent combination of values. After that, the path consistency and k-consistency are extended from this idea to check consistency among multiple variables.

2.4.4.3 Search algorithms

A variety of algorithms designed for resolving Constraint Satisfaction Problems (CSPs) which explore potential assignments of values to variables systematically. These algorithms guarantee the discovery of a solution if one exists or demonstrate insolvability of the problem. However, they might demand considerable time for these outcomes. This section introduces three systematic search algorithms which are a basic backtracking algorithm, look-ahead and look-back scheme. While the straightforward backtracking method is generally inefficient and seldom used in practice, it is presented here for comparison against the more advanced look-ahead and look-back approaches (Zhang, 2005).

2.4.4.3.1 Backtracking

Backtracking represents the most prevalent approach for systematic search. In this algorithm, the current variable is given a value from its domain. This assignment is subsequently validated against the existing partial solution. If any constraints between the current variable and previous ones are violated, the assignment is abandoned and an alternative value for the current variable is chosen. If all feasible values for the current variable have been exhausted, the algorithm backtracks to the preceding variable and assigns it a new value (Zhang, 2005).

There is an inherent shortcoming of this algorithm which is thrashing. This term signifies repeated failures due to identical causes. However, an intelligent form of backtracking can be employed to circumvent thrashing. This intelligent method entails performing backtracking directly to the variable that initially caused the failure. There is another drawback relates to redundant work where clashing variable values are not remembered. A backtracking-based technique exists that overcomes both of these limitations which is known as dependency-directed backtracking. This method is traditionally used in truth maintenance systems. Additionally, a shortcoming of the basic approach is its delayed conflict detection as it is unable to predict conflicts before they materialize. This issue can be mitigated by applying consistency techniques to forward check for potential conflicts (Kumar, 1992).

2.4.4.3.2 Look ahead cheme or Forward Checking

The look ahead scheme come into play when the algorithm is about to allocate a value to the subsequent variable (Dechter, 2003). There are two methods are encompassed within the look ahead schemes which are forward checking and the maintenance of arc consistency through look-ahead. Forward checking represents the simplest illustration of the look ahead approach (Haralick & Elliott, 1980). It identifies inconsistencies earlier than the basic backtracking technique. It enabled the early elimination of branches in the search tree destined for failure for forward checking to reduce both the search tree's size and the overall computational effort. However, it is important to acknowledge that forward checking entails more computations when each assignment is appended to the ongoing partial solution.

The forward checking algorithm check the constraints among the current variable, prior variables and upcoming variables. Upon assigning a value to the present variable, any value in the domain of an upcoming variable that clashes with this assignment gets eliminated from the domain. The significance of this lies in promptly recognizing when the domain of an upcoming variable becomes empty, it is signalling the inconsistency of the ongoing partial solution. Similar to earlier methods, this prompts the trial of another value for the current variables or initiates backtracking to the preceding variable. The states of the domains of upcoming variables are reset to their conditions before the assignment that triggered the failure. In contrast, basic backtracking would only have detected this failure when considering the upcoming variable and subsequently realizing that none of its potential values were compatible with the current partial solution (Barták, 1998).

For the full look-ahead strategy termed Maintaining Arc Consistency (MAC) scheme, it does not same as forward checking which evaluates constraints solely between the current variable and subsequent variables. MAC possesses an added capability which it identifies conflicts among future variables as well. Consequently, the advantage of MAC lies in its capacity to detect clashes between upcoming variables and enable the premature elimination of search tree branches that would otherwise result in failure. This advantage surpasses the capabilities of forward checking in pruning problematic branches from the search tree.

2.4.4.3.3 Look back scheme

The look-back scheme is employed when the algorithm reaches an impasse or dead-end and prepares to execute the backtracking phase (Dechter, 2003). All look-back methods share a common downside of delayed conflict detection. While they address inconsistencies when they arise, they lack the capacity to prevent these inconsistencies from emerging. Every look-back algorithm examines the reasons behind failures and facilitate a return to the responsible decisions to reevaluate and deduce extraneous compound values. Consequently, learning from failure experiences contributes to more efficient future searches (Tsang, 2014).

Within the look-back framework, there are techniques which are back-jumping and back-marking. Analogous to backtracking, back-jumping operates on one variable at a time. Given a variable, back-jumping seeks a valid value for it and ensures the compatibility with recorded labels up to that point. If no suitable value can be assigned to the current variable, the algorithm initiates backtracking. However, during this backtracking process, back-jumping conducts a thorough analysis to pinpoint the "culprit decisions" responsible for the failure. In situations where every value within the domain of current variable conflicts with committed labels, back-jumping backtracks to the most recent culprit decision. This approach contrasts with basic backtracking which usually backtracks to the immediate preceding variable (Tsang, 2014).

The other technique which is back-marking is to minimize the necessity for assessing compatibility by storing incompatible labels that have already been confirmed. It circumvents the repetition of compatibility checks that have already proven successful as outlined by Tsang (2014). While to optimize the process, back-marking retains information about the highest backtracking level for each variable. This strategic information empowers back-marking to avoid redundant compatibility assessments, thus efficiently bypassing checks that are already known to either succeed or fail (Zhang, 2005).

2.4.4.4 Variable and Value Ordering

According to Kumar (1992), there are several ways to enhance the performance of backtracking algorithms. These improvements can include conducting constraint propagation at search nodes of the tree, ensuring logical consistency throughout the process, employing intelligent backtracking strategies or selecting the order of variables and the sequence in which values are assigned carefully. In this section, we focus on the significance of effective variable ordering in enhancing search algorithm performance.

The variable ordering can be categorized as either static or dynamic ordering. In static ordering, the sequence of variables is predetermined before the search begins and remains constant throughout. For dynamic ordering, it adapts based on the current state of the search although it may not be suitable for all tree search algorithms (Haralick & Elliott, 1980). For instance, simple backtracking lacks additional information during the search to justify changing the initial ordering.

The sequence in which variables are considered for assignment has a profound impact on the time required to solve a Constraint Satisfaction Problem (CSP). The order in which values are evaluated for each variable also plays a pivotal role. There are common principles guide the selection of both variable and value ordering. For variable ordering, the "first-fail" principle is prevalent which emphasize tackling variables where failure is most likely to occur early in the search. For value ordering, it adheres to the "first-succeed" principle which focus on values that are likely to lead to successful solutions. Decisions regarding ordering influence the efficiency of search strategies significantly. The ordering decisions also affect the factors such as backtracking frequency, search space pruning in look-ahead algorithms and computational efficiency when compatibility checks are resource-intensive (Tsang, 2014).

2.4.4.4.1 Variable Ordering

The most widely used variable ordering principle is based on the "first-fail" concept which is prioritizing variables where failure is more probable. If failure is inevitable, so the process of detecting failure early in the process is advantageous. On the other hand, if the current partial solution can potentially expand into a complete solution, all remaining variables must be instantiated. In such cases, the process of selecting the variable with the smallest domain size is beneficial because it is likely to be the most difficult to find a value for and likely to encounter a dead end sooner. This choice is made with the reason that if the current partial solution does not lead to a complete solution, the current branch will eventually reach a dead end.

2.4.4.4.2 Value Ordering

On the other hand, the focus shifts to value selection when we are striving to achieve a complete solution based on past assignments. The "Succeed First" principle emphasizes choosing values that are likely to lead to success while avoiding those that could result in dead-ends. It is essential to pick values that have a higher likelihood of success to minimize the need for backtracking. If values are chosen in a way that leads to inevitable failure, this would result in a dead-end. Therefore, we are necessitating backtracking to a previous variable. In such cases, each value for the current variable eventually needs to be considered and the overall search space remains unchanged.

2.5 Software Development Methodology

A Software Development Methodology is a comprehensive and structured approach that governs the entire lifecycle of software development projects. It serves as a guiding framework for planning, designing, implementing, testing, deploying and maintaining software systems. The primary purpose of a methodology is to provide a systematic and organized path to produce high-quality software efficiently and effectively. It does this by defining the processes, procedures, best practices and roles involved in software development and ensuring that all stakeholders are aligned and understand their responsibilities.

A well-defined methodology typically includes a process framework outlining the stages of development which is from initial concept to post-deployment maintenance. It also assigns specific roles and responsibilities to team members and establishes clear lines of accountability. Additionally, methodologies specify the creation of artifacts and deliverables such as requirement documents, design diagrams and test plans to document key aspects of the project. Not only that, communication and collaboration are paramount which can foster regular interactions among team members, stakeholders and users to ensure that project goals and objectives are met.

The quality assurance and testing procedures are integral parts of any methodology with guidelines for identifying and rectifying defects and ensuring that the software aligns with user requirements. Furthermore, change management mechanisms are often included to address evolving project needs and adapt to new information. Overall, a Software Development Methodology serves as a roadmap for successful software development and provides a structured and adaptable approach to navigate the complexities of modern software projects. Therefore, this section will explore different software development methodologies which are Waterfall, Prototyping, Iterative and Incremental, Spiral, Rapid application development and Agile which includes Scrum, Kanban and Extreme Programming.

2.5.1 Waterfall Methodology

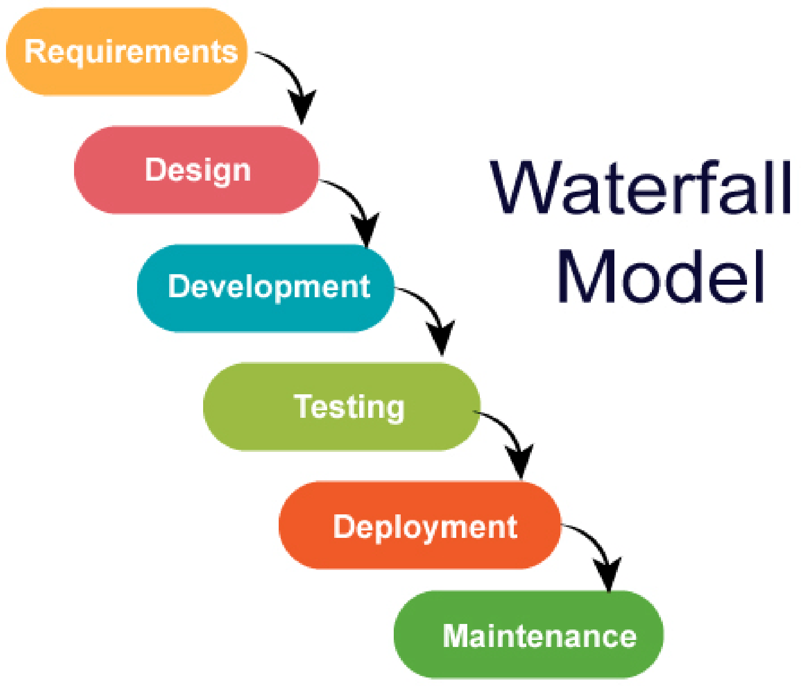


Figure 2.3: Waterfall Methodology

The Waterfall methodology is a linear, sequential approach to software development, where each phase must be completed before moving to the next. It starts with comprehensive requirements gathering where project stakeholders such as clients and users document their needs and expectations. Once the requirements are crystallized, the project moves on to the system design phase where architects and designers create comprehensive blueprints for the structure, functionality and user interfaces of the software. After that, there is development phase which the developers coding based on the design specifications. After coding, the rigorous testing is aiming to detect and rectify any defects or issues. Once the software passes the testing phase, it is deployed into the production environment and users gain access to the final product. Finally, the maintenance phase ensures ongoing support, updates and bug fixes. The Waterfall methodology's structured, step-by-step approach emphasizes documentation and is best suited for projects with well-defined and stable requirements where changes are expected to be minimal. However, it may lack flexibility when addressing evolving needs or adapting to unexpected challenges during development.

2.5.2 Prototyping

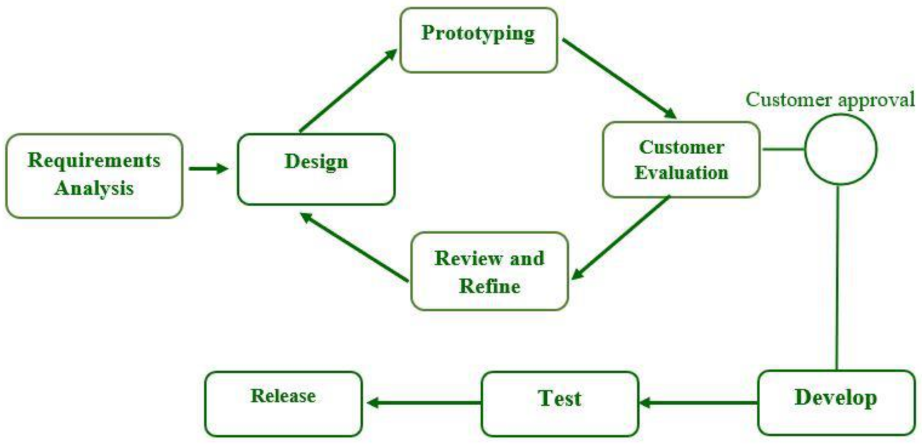


Figure 2.4: Prototyping Methodology

The prototyping methodology is an iterative and interactive approach to software development that places a strong emphasis on creating visual and functional prototypes of a system early in the project. The process starts with a preliminary understanding of user requirements which is then used to build a working model of the software's core features and interfaces. Unlike the traditional methodologies which prioritize comprehensive planning and documentation upfront, prototyping encourages quick and tangible representations of the software's functionality. The users and stakeholders actively participate in the prototype evaluation and providing feedback or insights that help refine the specifications and enhance the software design. This iterative cycle continues with the successive prototypes evolving to address user feedback and changing requirements. Once the prototype aligns closely with user expectations and requirements, the development team transitions to the full-scale implementation phase. The prototyping methodology is particularly valuable when requirements are not well-defined, when user needs may change or when the project's success depends on early user involvement and feedback. It accelerates the development process by identifying and addressing issues early and resulting in a more user-centred and effective final product.

2.5.3 Iterative and Incremental

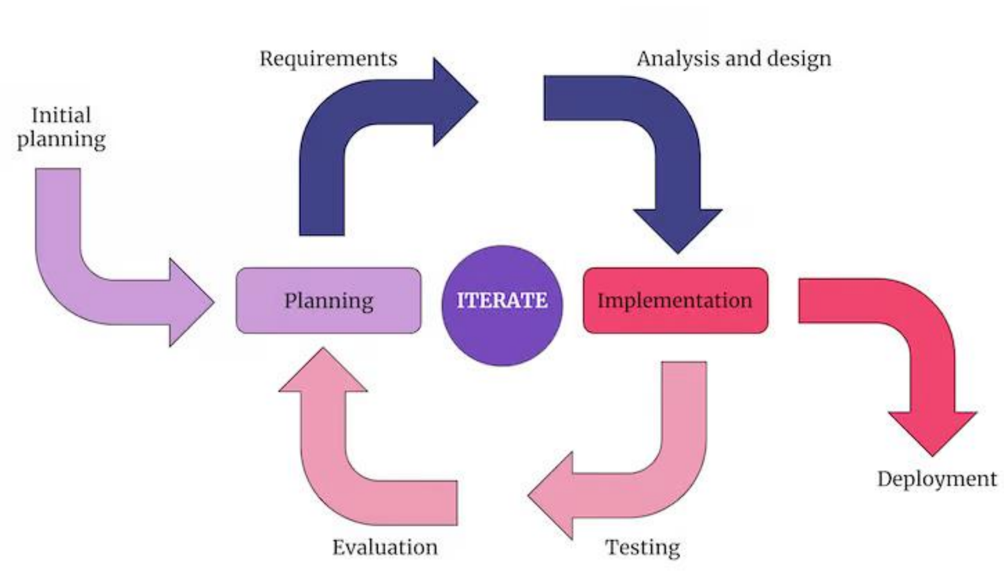


Figure 2.5: Iterative and Incremental Methodology

The Iterative and Incremental development is a dynamic approach to software development that divides a project into small, manageable portions or iterations which each building upon the work of the previous one. In this methodology, the software is continuously improved and refined through a repetitive process of planning, requirements analysis and designing, implementing, testing and evaluation. The iterations typically have fixed time frames often range from 2 to 4 weeks which a subset of the software features or functionality is developed and delivered. Unlike traditional methodologies that require a comprehensive and upfront specification, the iterative and incremental development begins with an initial set of requirements and as the project progresses through successive iterations, additional features or improvements are incorporated based on user feedback and evolving requirements. This approach fosters flexibility, adaptability and the ability to respond to changing project needs and priorities. It also encourages close collaboration with stakeholders which allow them to see the progress and provide the feedback early and frequently. Therefore, it can lead to a more robust and user-centric final product.

2.5.4 Spiral Model

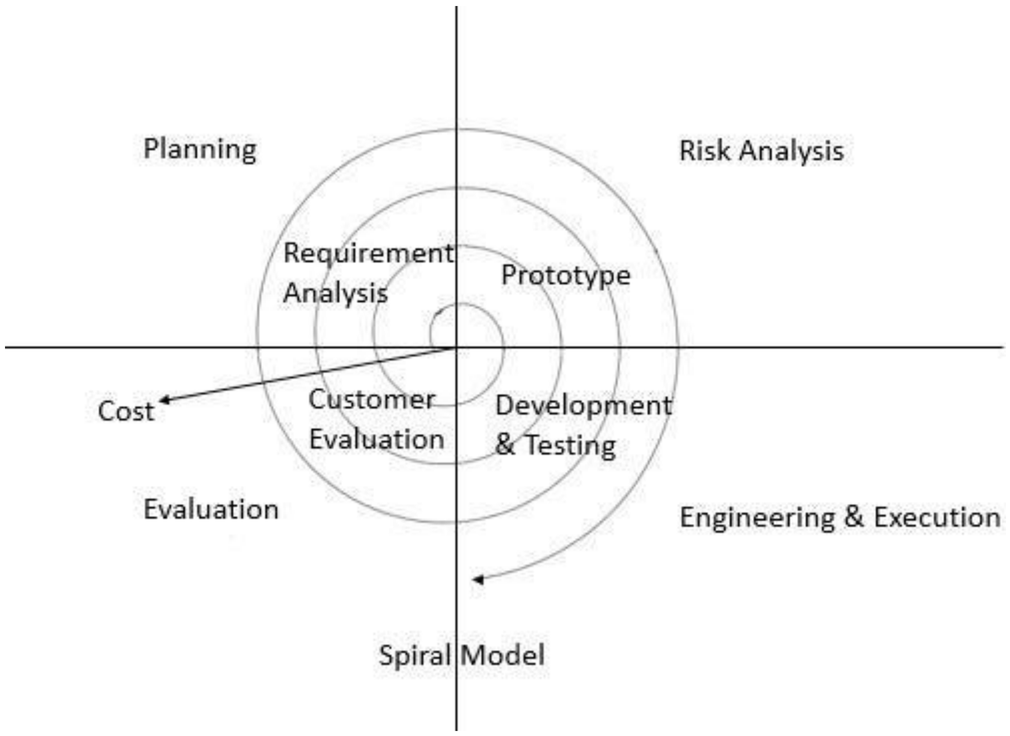


Figure 2.6: Spiral Methodology

The Spiral methodology is a risk-driven approach to software development that combines the iterative nature of Agile methodologies with elements of the traditional Waterfall model. It revolves around a series of iterative cycles or spirals where each comprised of four primary phases which are planning, risk analysis, engineering and evaluation. The process begins with the planning phase where the project objectives, requirements and constraints are defined. The next phase focuses on risk analysis where potential challenges, uncertainties and risks are identified, assessed and prioritized. After that, the engineering phase involves the actual development of the software. It is guided by the insights gained during risk analysis. The evaluation phase marks the conclusion of each spiral with a review and decision point to determine whether to proceed to the next iteration. This iterative and risk-driven approach allows for better management of evolving requirements, uncertainties and potential issues. It is particularly well-suited for complex projects where risks and uncertainties are high as it emphasizes continuous assessment and adaptation throughout the development process.

2.5.5 Rapid Application Development (RAD)

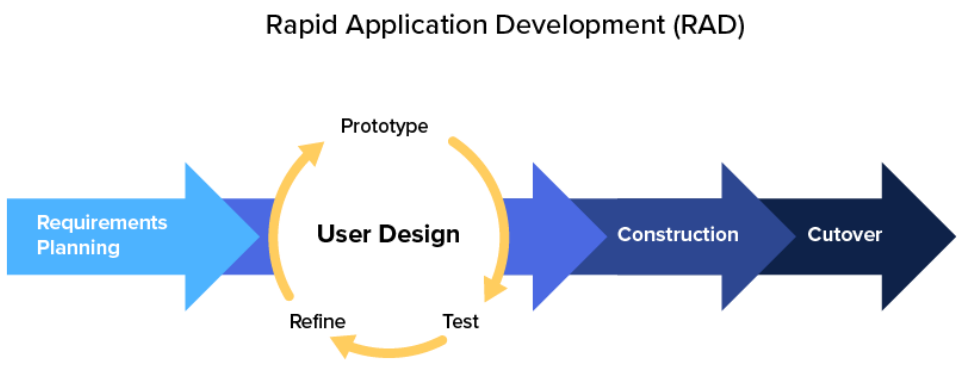


Figure 2.7: Rapid Application Development

Rapid Application Development (RAD) is an agile software development approach that prioritizes speed and flexibility in the development process. Unlike traditional methodologies that follow a linear sequence, RAD emphasizes rapid prototyping, quick iterations and active user involvement. RAD projects typically begin with a workshop or brainstorming session to gather user requirements which are then used to create a visual prototype of the software. This prototype serves as a working model that allows users to interact with and provide feedback on the system functionality and design. Based on this feedback, the development team iteratively refines and enhances the prototype by adding new features and addressing user concerns in short development cycles. The RAD methodology encourages continuous user feedback and collaboration to ensure that the software aligns closely with user needs. This approach is particularly beneficial for projects with rapidly changing requirements, tight deadlines or situations where early delivery of a functional product is essential. RAD also accelerates the development process, delivers software incrementally and reduces the risk of misunderstandings between developers and users.

2.5.5 Agile Methodology

Agile is an iterative and collaborative approach to software development that prioritizes flexibility, customer collaboration and delivering incremental value. The agile methodologies are including Rapid Application Development, Scrum and Kanban which break projects into small and manageable iterations. It typically lasts 2 to 4 weeks during which cross-functional teams work to create functional increments of the software. Agile projects start with a high-level vision and a prioritized backlog of features or user stories while the teams plan their work for each iteration accordingly. The daily stand-up meetings also foster communication and coordination while the progress is transparently tracked on visual boards or digital tools. Agile also encourages frequent collaboration with stakeholders and end-users to enable continuous feedback and adaptation. This iterative approach allows for the incorporation of changing requirements and making it well-suited for projects where the scope is not fully understood initially or where evolving user needs drive the development decisions. Agile also promotes delivering value early and often to ensure that the software evolves in response to user feedback and remains aligned with business objectives.

2.5.5.1 Scrum

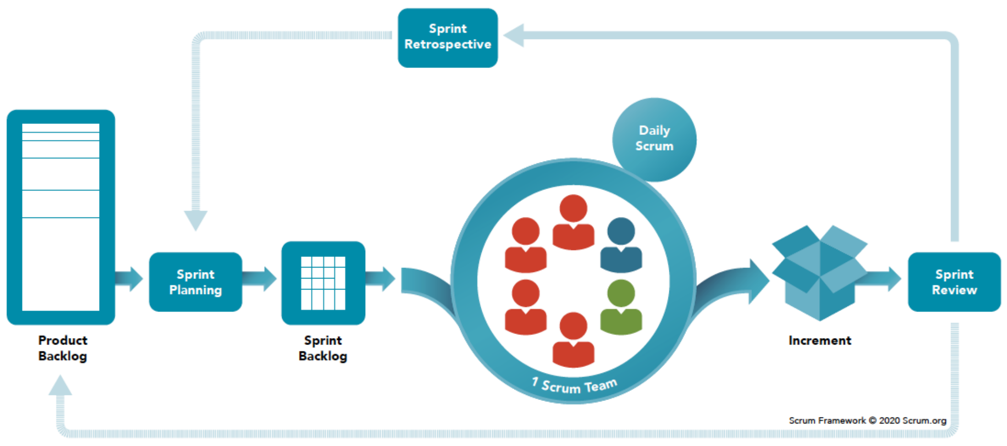


Figure 2.8: Scrum Methodology

Scrum is an agile framework that offers a structured and highly collaborative approach to software development. It divides projects into time-bound iterations known as sprints which take no more than 4 weeks and end with a working version of the application. Scrum teams are cross-functional with roles Scrum Master, Product Owner and Development Team members. The process begins with software requirement formulation and a prioritized backlog of work items known as user stories or features which are tackled during each sprint. All the stories are making up the product backlog. The daily scrum or daily stand-up meetings facilitate communication and coordination among team members and assess the progress while sprint planning sessions determine which backlog items will be addressed in the upcoming sprint. Scrum also emphasizes transparency and visual management with tasks tracked on a Kanban board or similar tool. At the end of each sprint, a sprint review allows stakeholders to provide feedback on the delivered increment while a sprint retrospective enables the team to reflect on its performance and make improvements. The iterative and incremental nature of Scrum allows for flexibility, adaptation to changing requirements and the delivery of valuable product increments with each sprint. It also fosters close collaboration between development teams and stakeholders to achieve project goals efficiently.

2.5.5.2 Kanban

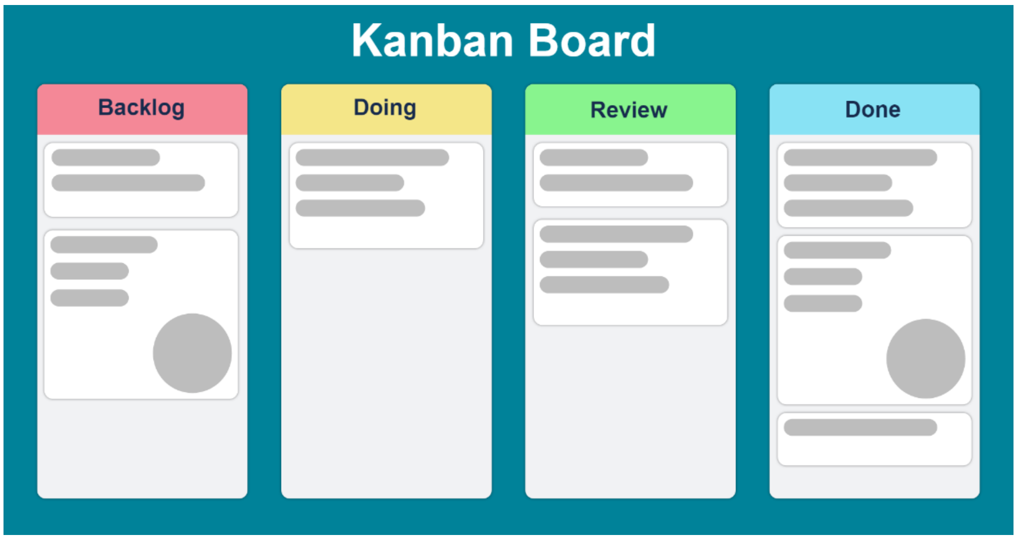


Figure 2.9: Kanban Methodology

Kanban is an agile methodology that offers a visual and flow-based approach to software development and project management. It centers around the use of a Kanban board which is a visual representation of the workflow where tasks or work items are represented as cards moving through various stages. Kanban is unlike with Scrum, it does not prescribe fixed time-bound iterations but the work is pulled through the system as capacity allows. The teams set limits on the number of tasks allowed in each stage which help to balance work and prevent overloading team members. The methodology also emphasizes continuous improvement and optimizing workflow efficiency. The new tasks are added to the board as they arise and priority is often determined by business needs or customer requirements. Kanban fosters flexibility and responsiveness and enables teams to adapt to changing priorities and demands. It is particularly useful for teams that have a continuous flow of work where tasks vary in size and complexity and where the focus is on managing and improving the process over time.

2.5.6 Comparison of Software Development Methodologies

Table 2.1: Comparison of Software Development Methodologies

Methodology	Characteristics	Strengths	Weaknesses
Waterfall	• Linear-Sequential Process • Comprehensive Documentation • Minimal Customer Involvement	• Good Clarity and Documentation • Predictable Timeline and Budget • Quality Assurance	• Inflexible to Changing Requirements • Limited Customer Feedback • Long Delivery Time • High Risk of Failure • No Deliverable Until Late
Prototyping	• Iterative Process • User-Centred • Quick Development of Functional Prototypes	• User Engagement • Risk Mitigation • Flexible to Changing and Unclear Requirements	• Time-Consuming • Resource Intensive • Not Ideal for Projects with Highly Regulated Requirements
Iterative and Incremental	• Phased or Incremental Approach • Continuous Feedback • Partial Functionality	• Early Deliveries • Flexible to Changing and Unclear Requirements • Quality Assurance • Risk Reduction through Early Testing and Feedback	• Complex Management of Multiple Iterations and Coordinating Changes • Resource Intensive When Develop and Test Increment • Not Ideal for Small Project
Spiral	• Risk-Driven • Iterative and Incremental	• Early Deliveries • Adaptability to Requirements Changes • Risk Mitigation through Risk Analysis	• Complex Management of Multiple Spiral and Risk Assessment • Resource Intensive • Costly Changes when Late

Rapid Application Development	• Quick Prototyping • Iterative and Incremental • User Involvement • Minimal Planning	• Quick Development • Reusable code • High User Satisfaction • Reduced Risk with Early Prototype	• Dependency on User Availability and Involvement • Limited Scope as It Focuses Rapid Delivery
Scrum	• Three core roles: Product Owner, Scrum Master and Development Team • Time-boxed iterations known as Sprints • Daily Scrum / Meeting to assess progress • Product Backlog with list of all features, enhancements and bug fixed • Sprint Planning involves selecting items from Product Backlog to work on during the sprint • Sprint Review is to showcase the work completed to stakeholder and gather feedback • Retrospectives is to reflect team processes and area improvement	• Adaptability to Requirements Changes • Transparency to project progress and priorities • Stakeholder Engagement in Regular Sprint Review • Quality Focus for Each Increment • Improved Collaboration among Team Members • Faster Delivery	• Learning Curve to adapt to a new way of working • Dependency on Team Discipline • Dependency on Stakeholder Availability • Limited Predictability of Feature Completion

Based on the comparison of the software development methodologies, scrum methodology was chosen as the proposed development methodology of the Classroom Finder System project. This choice is well-founded compared to other methodologies due to its alignment with the characteristics and specific needs of the project. The flexibility and adaptability approach of Scrum make it an excellent fit for this project where the requirements may keep evolving and changing especially in the educational environment where schedules and room availability can change frequently. Scrum allows for flexibility and adapts to changing requirements. Scrum also encourages incremental development which allows to deliver the working increments of the system after each sprint. This aligns well with this project as it can progressively add features and functionalities to ensure the essential components are available earlier in the development process.

The strengths of Scrum such as its focus on quality, transparency, early deliveries and risk management align with the critical aspects of the Classroom Finder System. By providing regular opportunities for stakeholder feedback and emphasizing incremental progress, Scrum ensures that the system remains responsive to changing requirements while maintaining high standards of quality and accuracy. In contrast, other methodologies such as Waterfall may not accommodate the evolving nature of the project requirements and the need for continuous stakeholder engagement. The Prototyping and Rapid Application Development (RAD) may provide rapid development and user satisfaction but may lack the level of transparency and structured documentation required for a project of this nature. The Iterative and Incremental methodologies also offer advantages but may not be as well-suited as Scrum due to the level of complexity involved in coordinating room allocation, student schedules and constraints. The Spiral methodology which focused on risk management may introduce more complexity than necessary for the Classroom Finder System and the resource-intensive nature of this approach could be less practical for the project scale.

In conclusion, the adaptability, transparency, focus on stakeholder engagement and emphasis on quality of Scrum make it the most suitable methodology for the Classroom Finder System project. It provides the project team with the tools and processes needed to efficiently develop and deliver a high-quality solution while accommodating evolving requirements and mitigating risks effectively.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this chapter, it delved into the critical aspect of selecting and justifying the software development methodology that guided the implementation of the Classroom Finder System. The choice of the appropriate methodology was important to the project success as it governed how we plan, execute and manage the development process. The process of selecting the right software development methodology for the project involved a comprehensive evaluation of various methodologies and their compatibility with our project's objectives and constraints. Additionally, a work breakdown structure and Gantt chart were provided as references and tracking tools for the entire development process. These aids helped in organizing tasks and allocating resources efficiently. Moreover, this chapter specified the development tools used in this project which ensured that the chosen tools align with the selected methodology and project requirements.

After careful consideration and evaluation in chapter 2 which was the literature review of the various methodologies, we had opted for the Agile methodology. It was because our project was of moderate size and complexity, it made the iterative and incremental approach of Agile suitable for adapting to the evolving requirements since educational institutions timetabling often experienced changes in requirements. Agile also emphasized continuous collaboration with stakeholders which was crucial in an educational context where lecturers, administrators and students played the important roles. Agile was a versatile and adaptive methodology that aligns well with the development of Classroom Finder System. It promoted incremental development, frequent stakeholder feedback and adaptability which were essential for the project success.

In order to apply Agile methodology, the Scrum framework within the Agile methodology was adopted. The structured approach of the Scrum methodology helped us to manage the development process more effectively.

3.2 Software Development Methodology

In this project, the Scrum methodology was implemented to address the dynamic and adaptable nature of the Classroom Finder System development. It was divided into five distinct phases which were initiation, planning and estimates, implementation, review and retrospective and release phase.

3.2.1 Initiation

The initiation phase was one of the most critical phases of the Scrum methodology as it was the foundation of the project vision, objective and key deliverable. During the initiation phase of the project, a preliminary study of the critical project elements was conducted. This involved a thorough investigation into the background of the project problem and the underlying causes. This investigation provided the necessary context for formulating the precise problem statements. After the problem statements were established, the project objectives were outlined carefully to offer a clear sense of purpose and alignment for all subsequent project activities. The objectives were crafted to direct the project towards its final outcomes more effectively.

In this phase, the project solution and project approach to address the problem were developed. In order to achieve this, an in-depth exploratory analysis was conducted. This analysis seemed to identify and compare the various Artificial Intelligence approaches and software development methodologies to propose a solution and approach for this project. This preliminary provided an initial direction for how the project will proceed. Furthermore, the scope of the project was also defined in the initiation phase. The project scopes included the project boundaries which were system scope and the user scope to provide information of the features and functionalities that were included in this project deliverable. This ensured a clear understanding of the project limits. There was also one of the important processes which was forming the product backlog from the scrum epics. The product backlog was the list of features which referred to as product epics. The backlog then served as the building blocks for the project development. Besides, the initial phase involved identifying all the individuals and defining their roles and responsibilities.

In summary, the initial phase was a crucial starting point for the project which helped to define the problem statement, project objective, scope, project solution and project approach. It produced a well-defined project vision, a comprehensive list of stakeholders and an initial product backlog. The information and outputs from the initiation phase served as the foundation for the project. They guided the project direction, informed the subsequent phases and set the stage for the project progression.

3.2.2 Sprint

There were three sprints in this project which were user, room, course, course timetable, course enrolment and assessment management, AI-powered room allocation and real-time update, notification and communication. Each of the sprints involved three scrum phases which were planning and estimates, implementation, review and retrospective phases.

3.2.2.1 Planning and Estimates

During this phase, the project high-level objectives were broken down into smaller and actionable tasks. The product epics were divided into user stories to make them more manageable for development. The user stories were the small pieces of work that described the specific functions or features. However, the user stories constituted the product backlog which was a list of necessary capabilities the project needs.

The one of the critical principles of Scrum was its iterative development framework. The work was organized into the scrum sprints which last for 2 to 4 weeks periods had end with shippable increments of a product. During this planning and estimates phase, the sprint planning was conducted to select the user stories from the product backlog to work on during a specific sprint. The user stories that selected for the specific sprint was also known as sprint backlog. In three of the sprints, the sprint backlog is comprising the user stories which related to user, room, course, timetable and enrolment of course and assessment management, AI-powered room allocation and real-time update, notification and communication. There was also another key process which is estimation. The estimation was conducted in this phase to assess each of the user stories for the effort required to complete it. In three of the sprints, the estimation was conducted for the user stories which related to the user, room, course, timetable and enrolment of course and assessment management, AI-powered room allocation and real-time update, notification and communication. This estimation process helped the teams to plan the sprint effectively.

At the end of the planning and estimates phases, there was a sprint backlog which comprised user stories to be worked on in the upcoming sprint as well as the estimations for each user story.

3.2.2.2 Implementation

In this implementation phase, the tasks and activities which identified in the sprint backlog were executed to accomplish the product goals and complete the project deliverables. There were 3 sprints in this project which were user, room, course, timetable and enrolment of course and assessment management, AI-powered room allocation and real-time update, notification and communication. In each of the sprints, there were key processes conducted which were daily stand-up meetings or daily scrum and burndown chart monitoring.

In the first sprint, the implementation of the user, room, course, timetable and enrolment of course and assessment management was conducted. The implementation was involving the development of the user interface for administrators to manage the user, room, course, timetable and enrolment of course and assessment. There was also development of the user interfaces and functionalities for student and lecturer to interact with the interface. Not only that, the database table creation of the user, room, course, timetable and enrolment of the course and assessment was conducted in first sprint to accomplish the project objectives. During the second sprint, AI-powered room allocation was implemented. The core logic for AI-powered room allocation while considering the constraints such as student availability, space requirements and time limitation were implemented into this project. The room allocation was implemented by using constraint satisfaction problems (CSP) AI technique. During the last sprint, there was an implementation of real-time update, notification and communication. After each of the room allocation or assessment timetabling process, there was a real-time update mechanism for room availability and allocations. In this sprint, there was also an implementation of the real-time communication among the administrator and lecturer or real-time notifications for room assignment, assessment timetabling details after any changes in assessment timetabling and room assignment.

In three of the sprints, there were daily scrum conducted every day to check on the task status, discuss the progress and address any obstacles. There were also conducting the burndown chart monitoring. The burndown charts can be used to track the progress compared to the planned work. It provided a visual representation of work completed and remaining. Lastly, the incremental progress was achieved during each sprint, and they were contributing to the overall project goals.

3.2.2.3 Review and Retrospective

The review and retrospective phase were an important part of the agile scrum process. It was focusing on the reflection and improvement after the completion of each sprint. It was emphasized on reviewing the product progress, reflecting on the progress and making necessary adjustments.

Once each of the sprint was completed, the sprint review and sprint retrospective were conducted. The sprint review allowed stakeholders and team members to evaluate the progress of the project while the sprint retrospective provided a chance to reflect on their experience, share their thoughts and decide how to optimize the process. The sprint review was also focusing on the product while the sprint retrospective was focusing on the process. During the sprint review, the progress of the previous sprint was evaluated against the product backlog. Based on the evaluation results, the completed user stories in the previous sprint backlog or product backlog were decided as completed or new ones need to be added. If there was a need to add new user stories, the process of updating or refining the product backlog was known as product backlog grooming. This process focusses to ensure that the product backlog was aligning with the project goals and suitable for planning the next sprint. During the sprint retrospective, the team reflected on the process of the sprint by sharing what goes well, what did not and the way to enhance the process. This process focussed to ensure the continuous improvement.

After the sprint review and retrospective was conducted followed by each of the sprints, the process went back to the planning and estimates phase.

The cycle continued and new sprint planning began. However, once the final deliverables were completed, the product retrospective was conducted to evaluate how the project can improve in the future.

3.2.2.4 Release

The release phase was the final phase of the Scrum methodology. During the release phase, the final project deliverables and final year report were prepared whereby the findings of the entire project were documented. This process ensured that all project deliverables met the required quality standards and were ready to release. In this release phase, it also focused on the stakeholder communication. The final products and report were shared with stakeholders while their feedback was gathered to ensure the satisfaction of the project. After that, the final deliverables and report were submitted, and project retrospective can be conducted to assess the improvement in the future projects. Once all of these processes were done, it marked the end of the project.

3.3 Development Tools

There were several development tools which were required in the development of the Classroom Finder System. The development tools were Visual Studio Code, Flask, Jinja2 Templating Engine, Bootstrap, SQLite and SQL Alchemy.

3.3.1 Visual Studio Code

Visual Studio Code is a powerful and highly versatile code editor developed by Microsoft. It is favoured by developers for its lightweight nature and rich set of features. In the context of the Classroom Finder System project, Visual Studio Code serves as the primary integrated development environment (IDE) for writing, managing and debugging the code for the application. It offers essential features such as syntax highlighting, code auto-completion and Git integration which are making it an excellent choice for Python development. The one of the key advantages of Visual Studio Code is its extensibility. It provides a vast library of extensions that can enhance the development experience and add support for multiple programming languages. This extensibility makes it a valuable tool for tackling the backend development of the Classroom Finder System including implementing Constraint Satisfaction Problem (CSP) algorithms and AI techniques. Visual Studio Code also enables efficient code testing and debugging which is essential for building a robust and functional backend.

3.3.2 Flask

Flask is a lightweight and open-source Python web framework that provides essential tools and features for developing the web applications and REST API in Python easily. Flask is known for its simplicity and ease of learning which easy to learn than Django framework. Therefore, it makes Flask web framework a suitable choice and Flask is selected as the foundation for building the backend of the Classroom Finder System. In the context of this project, Flask is used to create backend routes, handle HTTP requests and responses and interact with the database. It also plays a crucial role in defining the business logic of the Classroom Finder System. The lightweight nature of Flask is making it ideal for developing the web applications with a focus on specific functionality.

3.3.3 Jinja2 Template Engine

Jinja2 is a templating engine for Python that integrates seamlessly with Flask. It plays a critical role in creating dynamic web pages based on data passed from the backend. Jinja2 allows the generation of HTML pages dynamically by embedding placeholders (templates) within the HTML code. These placeholders are replaced with actual data when the page is rendered in the browser. This is particularly useful when there is a need to display data from Flask backend in the web application. Jinja2 also supports template inheritance which means the creation of a base HTML template with common elements like headers, footers and navigation menus. Then, this base template can be extended to create specific pages by adding content blocks. This promotes code reusability and consistency in the web application design. With Jinja2, the data from the Flask backend can pass to the HTML templates easily. This enables the display of the dynamic content such as user-specific information or real-time updates in the web pages.

3.3.4 Bootstrap

Bootstrap is a popular front-end framework that enhances the user interface (UI) and user experience (UX) of the web application. It is used to complement Flask by providing a framework for building an intuitive and visually appealing user interface for the web application. The combination of Flask and Bootstrap ensures that the frontend of Classroom Finder System is user-friendly and aesthetically pleasing.

3.3.5 SQLite

SQLite is a lightweight and self-contained SQL database engine that does not require a separate server process. It is widely used in embedded systems and applications due to its simplicity, reliability, and efficiency. In the Classroom Finder System, SQLite serves as the data storage and management solution. It plays a pivotal role in storing information about user, classrooms, course, course schedules, course enrolment, assessment details and other relevant data. The use of SQLite allows for efficient and structured data management which enables the application to create, read, update and delete records as needed. This relational database management system ensures that the Classroom Finder System can retrieve and store data efficiently to support essential functionalities like room allocation, course scheduling and enrolment, and assessment management. The robust capability of SQLite makes it a suitable choice for handling the complex data structures and relationships required by this project.

3.3.6 SQLAlchemy

SQLAlchemy is a powerful Python library that simplifies database interactions in Python applications especially when working with relational databases like SQLite. It provides a high-level Object-Relational Mapping (ORM) framework that bridges the gap between Python objects and database tables. SQLAlchemy allows defining Python classes that correspond to database tables which known as models. Each model class defines the structure of a database table which includes the columns and their data types. In the context of the Classroom Finder System, models for tables related to user, classrooms, course, course schedules and enrolment, and assessments are defined using SQLAlchemy. SQLAlchemy ORM also enables defining the structure of the database schema using Python code which is allowing tables, columns, indexes and relationships to be created and modified using Python classes and their attributes. Additionally, SQLAlchemy provides a powerful query-building API that allows constructing complex SQL queries using Python syntax as well as defining relationships between models corresponding to relationships between database tables.

3.4 Work Plan

3.4.1 Work Breakdown Structure (WBS)

0.0 Classroom Finder System

1.0 Planning Phase

- 1.1 Analyse the project title
- 1.2 Study of Background of Problem
- 1.3 Identify Problem Statement
- 1.4 Identify Project Objectives
- 1.5 Identify Project Solution
- 1.6 Identify Project Approach
- 1.7 Identify Project Scope

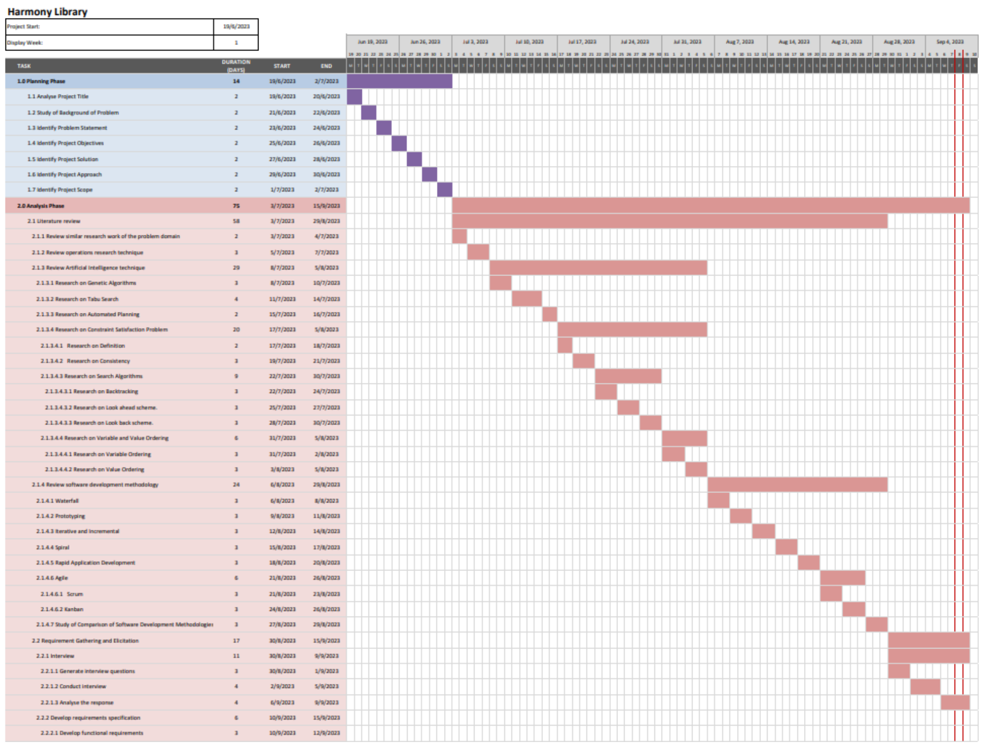
2.0 Analysis Phase

- 2.1 Literature review
    - 2.1.1 Review similar research work of the problem domain
    - 2.1.2 Review operations research technique
    - 2.1.3 Review Artificial Intelligence technique
        - 2.1.3.1 Research on Genetic Algorithms
        - 2.1.3.2 Research on Tabu Search
        - 2.1.3.3 Research on Automated Planning
        - 2.1.3.4 Research on Constraint Satisfaction Problem
            - 2.1.3.4.1 Research on Definition
            - 2.1.3.4.2 Research on Consistency
            - 2.1.3.4.3 Research on Search Algorithms
                - 2.1.3.4.3.1 Research on Backtracking
                - 2.1.3.4.3.2 Research on Look ahead scheme.
                - 2.1.3.4.3.3 Research on Look back scheme.
            - 2.1.3.4.4 Research on Variable and Value Ordering

2.1.3.4.4.1 Research on Variable Ordering
2.1.3.4.4.2 Research on Value Ordering
2.1.4 Review software development methodology
2.1.4.1 Waterfall
2.1.4.2 Prototyping
2.1.4.3 Iterative and Incremental
2.1.4.4 Spiral
2.1.4.5 Rapid Application Development
2.1.4.6 Agile
2.1.4.6.1 Scrum
2.1.4.6.2 Kanban
2.1.4.7 Study of Comparison of Software Development Methodologies
2.2 Requirement Gathering and Elicitation
2.2.1 Interview
2.2.1.1 Generate interview questions
2.2.1.2 Conduct interview
2.2.1.3 Analyse the response
2.2.2 Develop requirements specification
2.2.2.1 Develop functional requirements
2.2.2.2 Develop non- functional requirements
3.0 Design
3.1 Use Case Modelling
3.1.1 Develop use case diagram
3.1.2 Develop use case description
3.2 Develop design prototypes
4.0 Implementation
4.1 Initiation
4.1.1 Develop product backlog
4.2 Sprint 1
4.2.1 Planning and Estimates 1

4.5.3 Release Project Deliverables

3.4.2 Gantt Chart

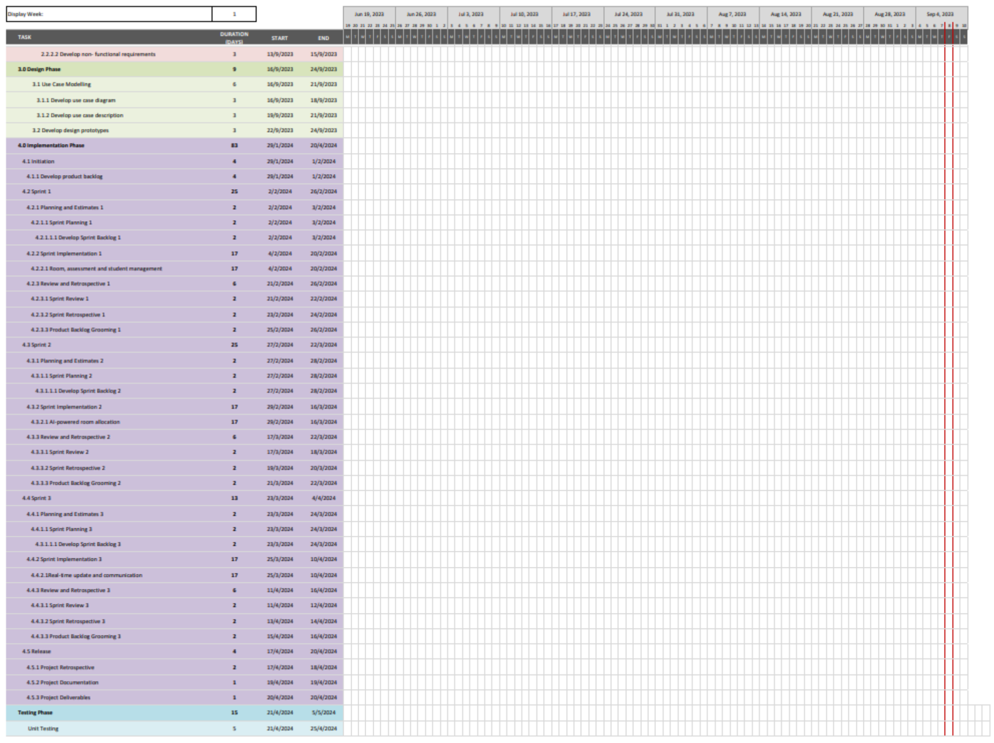


Harmony Library

Project Start:	19/6/2023
Display Week:	1

TASK	DURATION (DAYS)	START	END
1.0 Planning Phase	**14**	19/6/2023	2/7/2023
1.1 Analyse Project Title	2	19/6/2023	20/6/2023
1.2 Study of Background of Problem	2	21/6/2023	22/6/2023
1.3 Identify Problem Statement	2	23/6/2023	24/6/2023
1.4 Identify Project Objectives	2	25/6/2023	26/6/2023
1.5 Identify Project Solution	2	27/6/2023	28/6/2023
1.6 Identify Project Approach	2	29/6/2023	30/6/2023
1.7 Identify Project Scope	2	1/7/2023	2/7/2023
2.0 Analysis Phase	**75**	3/7/2023	15/9/2023
2.1 Literature review	58	3/7/2023	29/8/2023
2.1.1 Review similar research work of the problem domain	2	3/7/2023	4/7/2023
2.1.2 Review operations research technique	3	5/7/2023	7/7/2023
2.1.3 Review Artificial Intelligence technique	29	8/7/2023	5/8/2023
2.1.3.1 Research on Genetic Algorithms	3	8/7/2023	10/7/2023
2.1.3.2 Research on Tabu Search	4	11/7/2023	14/7/2023
2.1.3.3 Research on Automated Planning	2	15/7/2023	16/7/2023
2.1.3.4 Research on Constraint Satisfaction Problem	20	17/7/2023	5/8/2023
2.1.3.4.1 Research on Definition	2	17/7/2023	18/7/2023
2.1.3.4.2 Research on Consistency	3	19/7/2023	21/7/2023
2.1.3.4.3 Research on Search Algorithms	9	22/7/2023	30/7/2023
2.1.3.4.3.1 Research on Backtracking	3	22/7/2023	24/7/2023
2.1.3.4.3.2 Research on Look ahead scheme.	3	25/7/2023	27/7/2023
2.1.3.4.3.3 Research on Look back scheme.	3	28/7/2023	30/7/2023
2.1.3.4.4 Research on Variable and Value Ordering	6	31/7/2023	5/8/2023
2.1.3.4.4.1 Research on Variable Ordering	3	31/7/2023	2/8/2023
2.1.3.4.4.2 Research on Value Ordering	3	3/8/2023	5/8/2023
2.1.4 Review software development methodology	24	6/8/2023	29/8/2023
2.1.4.1 Waterfall	3	6/8/2023	8/8/2023
2.1.4.2 Prototyping	3	9/8/2023	11/8/2023
2.1.4.3 Iterative and Incremental	3	12/8/2023	14/8/2023
2.1.4.4 Spiral	3	15/8/2023	17/8/2023
2.1.4.5 Rapid Application Development	3	18/8/2023	20/8/2023
2.1.4.6 Agile	6	21/8/2023	26/8/2023
2.1.4.6.1 Scrum	3	21/8/2023	23/8/2023
2.1.4.6.2 Kanban	3	24/8/2023	26/8/2023
2.1.4.7 Study of Comparison of Software Development Methodologie	3	27/8/2023	29/8/2023
2.2 Requirement Gathering and Elicitation	17	30/8/2023	15/9/2023
2.2.1 Interview	11	30/8/2023	9/9/2023
2.2.1.1 Generate interview questions	3	30/8/2023	1/9/2023
2.2.1.2 Conduct interview	4	2/9/2023	5/9/2023
2.2.1.3 Analyse the response	4	6/9/2023	9/9/2023
2.2.2 Develop requirements specification	6	10/9/2023	15/9/2023
2.2.2.1 Develop functional requirements	3	10/9/2023	12/9/2023

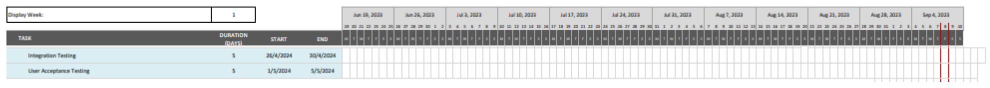
Figure 3.1: Gantt Chart (Part A)



Display Week:	1

TASK	DURATION (DAYS)	START	END
2.2.2.2 Develop non- functional requirements	3	13/9/2023	15/9/2023
3.0 Design Phase	**9**	16/9/2023	24/9/2023
3.1 Use Case Modelling	6	16/9/2023	21/9/2023
3.1.1 Develop use case diagram	3	16/9/2023	18/9/2023
3.1.2 Develop use case description	3	19/9/2023	21/9/2023
3.2 Develop design prototypes	3	22/9/2023	24/9/2023
4.0 Implementation Phase	**83**	29/1/2024	20/4/2024
4.1 Initiation	4	29/1/2024	1/2/2024
4.1.1 Develop product backlog	4	29/1/2024	1/2/2024
4.2 Sprint 1	25	2/2/2024	26/2/2024
4.2.1 Planning and Estimates 1	2	2/2/2024	3/2/2024
4.2.1.1 Sprint Planning 1	2	2/2/2024	3/2/2024
4.2.1.1.1 Develop Sprint Backlog 1	2	2/2/2024	3/2/2024
4.2.2 Sprint Implementation 1	17	4/2/2024	20/2/2024
4.2.2.1 Room, assessment and student management	17	4/2/2024	20/2/2024
4.2.3 Review and Retrospective 1	6	21/2/2024	26/2/2024
4.2.3.1 Sprint Review 1	2	21/2/2024	22/2/2024
4.2.3.2 Sprint Retrospective 1	2	23/2/2024	24/2/2024
4.2.3.3 Product Backlog Grooming 1	2	25/2/2024	26/2/2024
4.3 Sprint 2	25	27/2/2024	22/3/2024
4.3.1 Planning and Estimates 2	2	27/2/2024	28/2/2024
4.3.1.1 Sprint Planning 2	2	27/2/2024	28/2/2024
4.3.1.1.1 Develop Sprint Backlog 2	2	27/2/2024	28/2/2024
4.3.2 Sprint Implementation 2	17	29/2/2024	16/3/2024
4.3.2.1 AI-powered room allocation	17	29/2/2024	16/3/2024
4.3.3 Review and Retrospective 2	6	17/3/2024	22/3/2024
4.3.3.1 Sprint Review 2	2	17/3/2024	18/3/2024
4.3.3.2 Sprint Retrospective 2	2	19/3/2024	20/3/2024
4.3.3.3 Product Backlog Grooming 2	2	21/3/2024	22/3/2024
4.4 Sprint 3	13	23/3/2024	4/4/2024
4.4.1 Planning and Estimates 3	2	23/3/2024	24/3/2024
4.4.1.1 Sprint Planning 3	2	23/3/2024	24/3/2024
4.3.1.1.1 Develop Sprint Backlog 3	2	23/3/2024	24/3/2024
4.4.2 Sprint Implementation 3	17	25/3/2024	10/4/2024
4.4.2.1Real-time update and communication	17	25/3/2024	10/4/2024
4.4.3 Review and Retrospective 3	6	11/4/2024	16/4/2024
4.4.3.1 Sprint Review 3	2	11/4/2024	12/4/2024
4.4.3.2 Sprint Retrospective 3	2	13/4/2024	14/4/2024
4.4.3.3 Product Backlog Grooming 3	2	15/4/2024	16/4/2024
4.5 Release	4	17/4/2024	20/4/2024
4.5.1 Project Retrospective	2	17/4/2024	18/4/2024
4.5.2 Project Documentation	1	19/4/2024	19/4/2024
4.5.3 Project Deliverables	1	20/4/2024	20/4/2024
Testing Phase	**15**	21/4/2024	5/5/2024
Unit Testing	5	21/4/2024	25/4/2024

Figure 3.2: Gantt Chart (Part B)



Display Week:	1

TASK	DURATION (DAYS)	START	END
Integration Testing	5	26/4/2024	30/4/2024
User Acceptance Testing	5	1/5/2024	5/5/2024

Figure 3.3: Gantt Chart (Part C)

CHAPTER 4

PROJECT SPECIFICATION

4.1 Introduction

This chapter was primarily focusing on the project specifications. It involved outlining the system requirements in terms of both functional and non-functional requirements. Based on the functional requirement, the requirement modelling was conducted to produce use case diagram and use case description. The requirement modelling was conducted to give a visualization and understanding on how users interact with the system. It defined the various scenarios or use cases in which the system used. Lastly, the prototype was developed to demonstrate the functionality and design of the system. It can serve as a visual representation that stakeholders can interact with and understand the system better.

4.2 Functional and Non-Functional Requirements Specification

This section depicted the system requirements specification of Classroom Finder System which can be classified into two categories which were functional requirements and non-functional requirements.

4.2.1 Functional Requirements

For the functional requirements, the requirements were divided by three roles which were administrator, lecturer and student.

Functional requirements for administrator:

1. The system should allow administrator to login and logout the account.
2. The system should allow administrator to manage (create, read, update, delete) the student.
3. The system should allow administrator to manage (create, read, update, delete) the lecturer.

4. The system should allow administrator to manage (create, read, update, delete) the room.

5. The system should allow administrator to manage (create, read, update, delete) the course.

6. The system should allow administrator to manage (create, read, update, delete) the course timetable.

7. The system should allow administrator to manage (create, read, update, delete) the course enrolment.

8. The system should allow administrator to manage (create, read, update, delete) the assessment.

9. The system should allow administrator to advance flexible find the slot and room for the assessment creation.

10. The system should allow administrator to find the room for the assessment creation.

11. The system should allow administrator to view the student who enrols in the assessment.

12. The system should allow administrator to notify the student who enrols in the assessment.

13. The system should allow administrator online chat or communication with lecturer.

Functional requirements for lecturer:

1. The system should allow lecturer to login and logout the account.

2. The system should allow lecturer to manage (create, read, update, delete) the assessment.

3. The system should allow lecturer to advance flexible find the slot and room for the assessment creation.

4. The system should allow lecturer to find the room for the assessment creation.

5. The system should allow lecturer to view the student who enrols in the assessment.

6. The system should allow lecturer to notify the student who enrols in the assessment.

7. The system should allow lecturer to view the calendar.

8. The system should allow lecturer online chat or communication with administrator.

Functional requirements for student:

1. The system should allow student to register the account.
2. The system should allow student to login and logout the account.
3. The system should allow student to view the assessment.
4. The system should allow student to view the calendar.

4.2.2 Non-Functional Requirements

The non-functional requirements were known as quality attributes or supplementary requirements which specified the operation of a system rather than its specific behaviours. Unlike functional requirements that describe what the system should do, non-functional requirements define how the system should perform. These requirements were crucial for ensuring the overall quality, usability, reliability, security and performance of the system. The non-functional requirements of the system are outlined as below:

1. Performance
    a. The system shall handle a large number of users, rooms, courses, course timetable, course enrolment and assessment efficiently.
    b. The system shall respond to user interaction within 5 seconds.
    c. The system shall display a loading indicator during data retrieval or processing operations to inform users about ongoing tasks.
2. Usability
    a. The system shall feature an intuitive and user-friendly interface that enables users to interact with it effortlessly which promotes ease of navigation and task completion.
    b. The system shall prompt users for confirmation before executing potentially destructive actions such as deleting records or modifying critical data to prevent accidental data loss or unintended changes.
3. Reliability
    a. The system shall maintain high availability and reliability to ensure a maximum downtime of no more than 2 hours per year to minimize disruptions to users.
4. Security
    a. The system shall implement user authentication mechanisms to prevent unauthorized access and require users to provide valid credentials before accessing sensitive information or performing actions.
    b. The system shall employ encryption or hashing techniques to safeguard user credentials and ensure that passwords are securely stored and transmitted.

4.3 System Use Case

4.3.1 Use Case Diagram

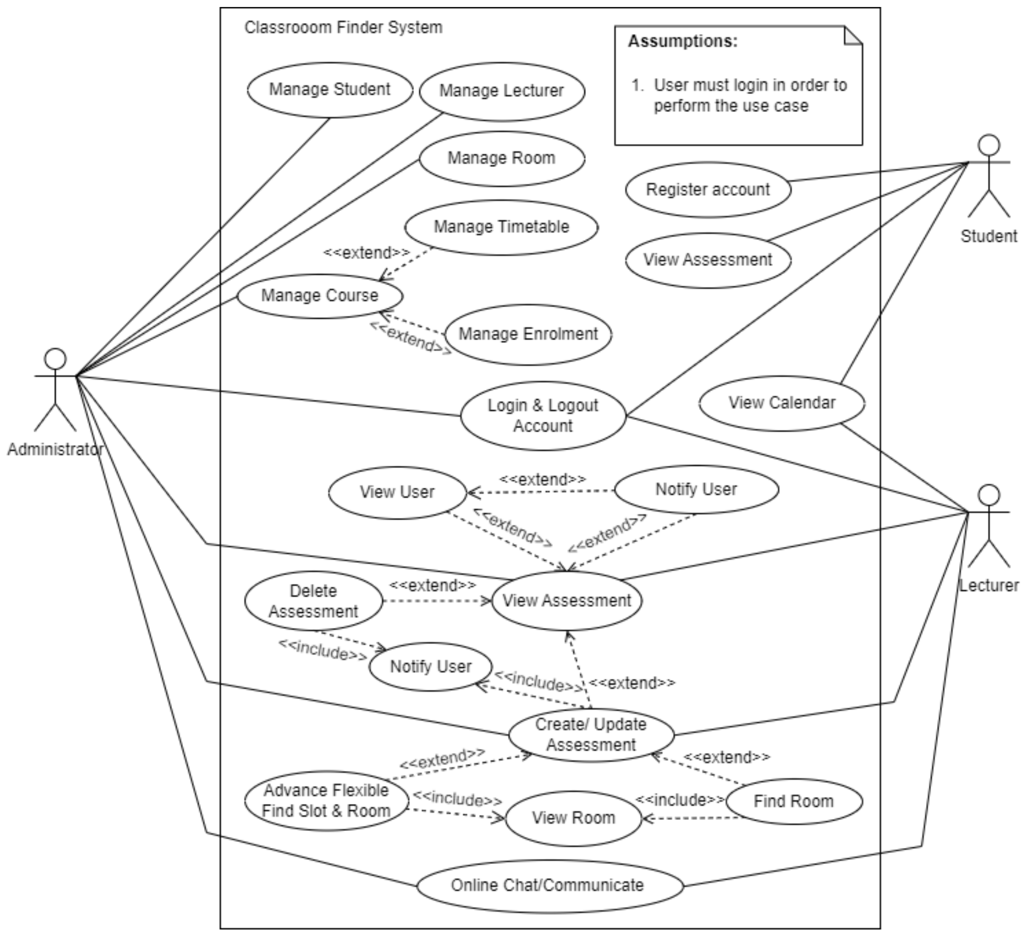


Figure 4.1: Use Case Diagram

4.3.2 Use Case Description

Table 4.1: Use Case for Register Account

Use Case Name: **Register Account**	ID: **UC001**	Importance Level: **High**
Primary Actor**: Student**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Student – want to register an account.**		
Brief Description**: This use case describes how the system allows student to register an account.**		
Trigger**: Students want to register an account.**		
Relationships: Association **: Student** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. Student selects “Sign Up” in the menu bar. 2. The system prompts students to enter the registration details to register an account. (S1) 3. The system verifies and authenticates the user details. (3E1) 4. The student is registered successfully. 5. Use case ends.		
Sub-flows: **S1:** **2.1: Student enters the ID.** **2.2: Student enters the Username.** **2.3: Student enters the Email.** **2.4: Student enters the Password.** **2.5: Student enters the Confirm Password.**		
Alternate/Exceptional Flows: 3E1 – Invalid user details to register an account or user existed.		

Table 4.2: Use Case for Login Account

Use Case Name**: Login Account**	ID**: UC002**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer, Student**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator, Lecturer, Student – want to login their account to access the system.**		
Brief Description**: This use case describes how the system allows administrator, lecturer and student to login their account before accessing the system.**		
Trigger**: Administrator, lecturer and student want to login their account before accessing the system.**		
Relationships: Association **: Administrator, Lecturer, Student** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 6. Administrator, lecturer and student select "Login" in the menu bar. 7. The system prompts administrator, lecturer and student to enter the login details to login the system. (S1) 8. The system verifies and authenticates the login details. (3E1) 9. The administrator, lecturer and student are logged into the system successfully. 10. Use case ends.		
Sub-flows: **S1:** **2.1: Administrator, lecturer and student enter the User Email.** **2.2: Administrator, lecturer and student enter the User Password.**		
Alternate/Exceptional Flows: 3E1 – Invalid login details to login the system.		

Table 4.3: Use Case for Logout Account

Use Case Name**: Logout Account**	ID**: UC003**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer, Student**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Administrator, Lecturer, Student – want to logout their account.**		
Brief Description**: This use case describes how the system allows administrator, lecturer and student to logout their account.**		
Trigger**: Administrator, lecturer and student want to logout their account.**		
Relationships: Association **: Administrator, Lecturer, Student** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. Administrator, lecturer and student select "Logout" in the menu bar. 2. The administrator, lecturer and student are logged out the system successfully. 3. Use case ends.		
Sub-flows:		
Alternate/Exceptional Flows:		

Table 4.4: Use Case for Manage Student

Use Case Name**: Manage Student**	ID**: UC004**	Importance Level**: High**
Primary Actor**: Administrator**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Administrator – wants to view, add, update and delete the student.**		
Brief Description**: This use case describes how the system allows administrator to view, add, update and delete the student.**		
Trigger**: Administrator wants to manage the student.**		
Relationships: Association **: Administrator** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator selects “Student” in the menu bar to view the student. 2. The system shows all the students’ details. (S1) 3. The administrator selects “Add New Student” in the student list page to create the student. 3.1 System prompts administrator to enter the student details for student creation. (S2) 3.2 The system verifies the student details to create the student. (3E1) 3.3 Student is created. 4. The administrator selects “Update” in the student list page to update the particular student. 4.1 System prompts administrator to enter the student details to update the student. (S3) 4.2 The system verifies the student details to update the student. (4E1) 4.3 Student is updated. 5. The administrator selects “Delete” in the student list page to delete the particular student. 5.1 System prompts administrator to cancel or confirm to delete the student.		

5.2 The system checks and verifies the student details to delete the student. (5E1) 5.3 Student is deleted. 6. Use case ends.
Sub-flows: **S1:** **2.1: The system shows the student's ID.** **2.2: The system shows the student's username.** **2.3: The system shows the student's email.** **S2:** **3.1.1: Administrator enters the student's ID.** **3.1.2: Administrator enters the student's username.** **3.1.3: Administrator enters the student's email.** **S3:** **4.1.1: Administrator enters the student's username.** **4.1.2: Administrator enters the student's email.**
Alternate/Exceptional Flows: 3E1 – Invalid student details to create student or student existed. 4E1 – Invalid student details to update student or student existed. 5E1 – Unable to delete student due to involving in course enrolment.

Table 4.5: Use Case for Manage Lecturer

Use Case Name**: Manage Lecturer**	ID**: UC005**	Importance Level**: High**
Primary Actor**: Administrator**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator – wants to view, add, update and delete the lecturer.**		
Brief Description**: This use case describes how the system allows administrator to view, add, update and delete the lecturer.**		
Trigger**: Administrator wants to manage the lecturer.**		
Relationships: Association **: Administrator** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator selects "Lecturer" in the menu bar to view the lecturer. 2. The system shows all the lecturers' details. (S1) 3. The administrator selects "Add New Lecturer" in the lecturer list page to create the lecturer. 3.1 System prompts administrator to enter the lecturer details for lecturer creation. (S2) 3.2 The system verifies the lecturer's details to create the lecturer. (3E1) 3.3 Lecturer is created. 4. The administrator selects "Update" in the lecturer list page to update the particular lecturer. 4.1 System prompts administrator to enter the lecturer details to update the lecturer. (S3) 4.2 The system verifies the lecturer's details to update the lecturer. (4E1) 4.3 Lecturer is updated. 5. The administrator selects "Delete" in the lecturer list page to delete the particular lecturer.		

5.1 System prompts administrator to cancel or confirm to delete the lecturer. 5.2 The system checks and verifies the lecturer details to delete the lecturer. (5E1) 5.3 Lecturer is deleted. 6. Use case ends.
Sub-flows: **S1:** **2.1: The system shows the lecturer's ID.** **2.2: The system shows the lecturer's username.** **2.3: The system shows the lecturer's email.** **S2:** **3.1.1: Administrator enters the lecturer's ID.** **3.1.2: Administrator enters the lecturer's username.** **3.1.3: Administrator enters the lecturer's email.** **S3:** **4.1.1: Administrator enters the lecturer's username.** **4.1.2: Administrator enters the lecturer's email.**
Alternate/Exceptional Flows: 3E1 – Invalid lecturer details to create lecturer or lecturer existed. 4E1 – Invalid lecturer details to update lecturer or lecturer existed. 5E1 – Unable to delete lecturer due to involving in course enrolment.

Table 4.6: Use Case for Manage Room

Use Case Name**: Manage Room**	ID**: UC006**	Importance Level**: High**
Primary Actor**: Administrator**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Administrator – wants to view, add, update and delete the room.**		
Brief Description**: This use case describes how the system allows administrator to view, add, update and delete the room.**		
Trigger**: Administrator wants to manage the room.**		
Relationships: Association **: Administrator** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator selects “Room” in the menu bar to view the room. 2. The system shows all the room details. (S1) 3. The administrator selects “Add New Room” in the room list page to create the room. 3.1 System prompts administrator to enter the room details for room creation. (S2) 3.2 The system verifies the room details to create the room. (3E1) 3.3 Room is created. 4. The administrator selects “Update” in the room list page to update the particular room. 4.1 System prompts administrator to enter the room details to update the room. (S3) 4.2 The system verifies the room details to update the room. (4E1) 4.3 Room is updated. 5. The administrator selects “Delete” in the room list page to delete the particular room. 5.1 System prompts administrator to cancel or confirm to delete the room.		

5.2 The system checks and verifies the room details to delete the room. (5E1) 5.3 Room is deleted. 6. Use case ends.
Sub-flows: **S1:** **2.1: The system shows the room’s ID.** **2.2: The system shows the room’s Type.** **2.3: The system shows the room’s Capacity.** **2.4: The system shows the room’s Equipment Availability.** **S2:** **3.1.1: Administrator enters the room’s ID.** **3.1.2: Administrator enters the room’s Type.** **3.1.3: Administrator enters the room’s Capacity.** **3.1.4: Administrator ticks the room’s Equipment Availability.** **S3:** **4.1.1: Administrator enters the room’s Type.** **4.1.2: Administrator enters the room’s Capacity.** **4.1.3: Administrator ticks the room’s Equipment Availability.**
Alternate/Exceptional Flows: 3E1 – Invalid room details to create room or room existed. 4E1 – Invalid room details to update room or room existed. 5E1 – Unable to delete room due to using in assessment.

Table 4.7: Use Case for Manage Course

Use Case Name**: Manage Course**	ID**: UC007**	Importance Level**: High**
Primary Actor**: Administrator**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator – wants to view, add, update and delete the course.**		
Brief Description**: This use case describes how the system allows administrator to view, add, update and delete the course.**		
Trigger**: Administrator wants to manage the course.**		
Relationships: Association **: Administrator** Include **: N/A** Extend **: Manage Timetable, Manage Enrolment** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator selects “Course” in the menu bar to view the course. 2. The system shows all the course details. (S1) 3. The administrator selects “Add New Course” in the course list page to create the course. 3.1 System prompts administrator to enter the course details for course creation. (S2) 3.2 The system verifies the course details to create the course. (3E1) 3.3 Course is created. 4. The administrator selects “Update” in the course list page to update the particular course. 4.1 System prompts administrator to enter the course details to update the course. (S3) 4.2 The system verifies the course details to update the course. (4E1) 4.3 Course is updated. 5. The administrator selects “Delete” in the course list page to delete the particular course. 5.1 System prompts administrator to cancel or confirm to delete the course.		

5.2 The system checks and verifies the course details to delete the course. (5E1) 5.3 Course is deleted. 6. Use case ends
Sub-flows: **S1:** **2.1: The system shows the course's ID.** **2.2: The system shows the course's Info.** **S2:** **3.1.1: Academic staff enters the course's ID.** **3.1.2: Academic staff enters the course's Info.** **S3:** **4.1.1: Academic staff enters the course's Info.**
Alternate/Exceptional Flows: 3E1 – Invalid course details to create course or course existed. 4E1 – Invalid course details to update course or course existed. 5E1 – Unable to delete course due to using in course timetable, course enrolment or assessment.

Table 4.8: Use Case for Manage Course Timetable

Use Case Name**: Manage Course Timetable**	ID**: UC008**	Importance Level**: High**
Primary Actor**: Administrator**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Administrator – wants to view, add, update and delete the course timetable.**		
Brief Description**: This use case describes how the system allows administrator to view, add, update and delete the course timetable.**		
Trigger**: Administrator wants to manage the course timetable.**		
Relationships: Association **: Administrator** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator selects "View Timetable" in the course list page to view the particular course timetable. 2. The system shows particular course timetable details. (S1) 3. The administrator selects "Add Timetable" in the course timetable list page to create the course timetable. 3.1 System prompts administrator to enter the course timetable details for course timetable creation. (S2) 3.2 The system verifies the course timetable details to create the course timetable. (3E1) 3.3 Course Timetable is created. 4. The administrator selects "Update" in the course timetable list page to update the particular course timetable. 4.1 System prompts administrator to enter the course timetable details to update the course timetable. (S3) 4.2 The system verifies the course timetable details to update the course timetable. (4E1) 4.3 Course Timetable is updated.		

5. The administrator selects "Delete" in the course timetable list page to delete the particular course timetable. 5.1 The system prompts administrators to cancel or confirm to delete the course timetable. 5.2 The system checks and verifies the course timetable details to delete the course timetable. (5E1) 5.3 Course Timetable is deleted. 6. Use case ends.
Sub-flows: **S1:** **2.1: The system shows the Course Timetable ID.** **2.2: The system shows the Weekday.** **2.3: The system shows the Session.** **2.4: The system shows the Session No.** **2.5: The system shows the Start Time.** **2.6: The system shows the End Time.** **S2:** **3.1.1: Administrator enters the Session.** **3.1.2: Administrator enters the Session No.** **3.1.3: Administrator enters the Weekday.** **3.1.4: Administrator enters the Start Time.** **3.1.5: Administrator enters the End Time.** **S3:** **4.1.1: Administrator enters the Session.** **4.1.2: Administrator enters the Session No.** **4.1.3: Administrator enters the Weekday.** **4.1.4: Administrator enters the Start Time.** **4.1.5: Administrator enters the End Time.**
Alternate/Exceptional Flows: 3E1 – Invalid course timetable details to create course timetable or course timetable existed. 4E1 – Invalid course timetable details to update course timetable or course timetable existed.

5E1 – Unable to delete course timetable due to using in course enrolment.

Table 4.9: Use Case for Manage Course Enrolment

Use Case Name: **Manage Course Enrolment**	ID: **UC009**	Importance Level: **High**
Primary Actor**: Administrator**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Administrator – wants to view, add, update and delete the course enrolment.**		
Brief Description**: This use case describes how the system allows administrator to view, add, update and delete the course enrolment.**		
Trigger**: Administrator wants to manage the course enrolment.**		
Relationships: Association **: Administrator** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator selects "View Enrolment" in the course list page to view the particular course enrolment. 2. The system shows particular course enrolment details. (S1) 3. The administrator selects "Add Enrolment" in the course enrolment list page to create the course enrolment. 3.1 System prompts administrator to enter the course enrolment details for course enrolment creation. (S2) 3.2 The system verifies the course enrolment details to create the course enrolment. (3E1) 3.3 Course Enrolment is created. 4. The administrator selects "Update" in the course enrolment list page to update the particular course enrolment. 4.1 System prompts administrator to enter the course enrolment details to update the course enrolment. (S3) 4.2 The system verifies the course enrolment details to update the course enrolment. (4E1)		

4.3 Course Enrolment is updated. 5. The administrator selects "Delete" in the course enrolment list page to delete the particular course enrolment. 5.1 System prompts administrator to cancel or confirm to delete the course enrolment. 5.2 Course Enrolment is deleted. 6. Use case ends.
Sub-flows: **S1:** **2.1: The system shows the Course Enrolment ID.** **2.2: The system shows the User ID.** **2.3: The system shows the Session.** **2.4: The system shows the Session No.** **S2:** **3.1.1: Administrator selects the User Type.** **3.1.2: Administrator selects the User.** **3.1.3: Administrator selects the Session.** **3.1.4: Administrator selects the Session No.** **S3:** **4.1.1: Administrator selects the User Type.** **4.1.2: Administrator selects the User.** **4.1.3: Administrator selects the Session.** **4.1.4: Administrator selects the Session No.**
Alternate/Exceptional Flows: 3E1 – Invalid course enrolment details to create course enrolment or course enrolment existed. 4E1 – Invalid course enrolment details to update course enrolment or course enrolment existed.

Table 4.10: Use Case for View Assessment

Use Case Name**: View Assessment**	ID**: UC010**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer, Student**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator, Lecturer and Student– want to view the assessment.**		
Brief Description**: This use case describes how the system allows administrator, lecturer and student to view the assessment.**		
Trigger**: Administrator, lecturer and student want to view the assessment.**		
Relationships: Association **: Administrator, Lecturer, Student** Include **: N/A** Extend **: Create or Update Assessment, Delete Assessment, View Assessment User, Notify Assessment User** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator, lecturer and student select "Assessment" in the menu bar to view the assessment. 2. The system shows all the assessment details. (S1) 3. Use case ends.		
Sub-flows: **S1:** **2.1: The system shows the Assessment ID.** **2.2: The system shows the Assessment Date.** **2.3: The system shows the Assessment Course.** **2.4: The system shows the Assessment Info.** **2.5: The system shows the Assessment Start Time.** **2.6: The system shows the Assessment End Time.** **2.7: The system shows the Assessment Room.**		
Alternate/Exceptional Flows:		

Table 4.11: Use Case for Create and Update Assessment

Use Case Name: **Create and Update Assessment**	ID: **UC011**	Importance Level: **High**
Primary Actor: **Administrator, Lecturer**		Use Case Type: **Detail, Essential**
Stakeholders and Interests: **Administrator, Lecturer – want to create or update the assessment.**		
Brief Description: **This use case describes how the system allows administrator and lecturer to create or update the assessment.**		
Trigger: **Administrator and lecturer want to create or update the assessment.**		
Relationships: Association : **Administrator, Lecturer** Include : **Notify Assessment User** Extend : **Find Room, Advance Flexible Find Room** Generalization : **N/A**		
Normal Flow of Events: 1. The administrator and lecturer select "Assessment" in the menu bar to view the assessment. 2. The system shows all the assessment details. (S1) 3. The administrator and lecturer select "Add Assessment" in the assessment list page to create the assessment. 3.1 System prompts administrator and lecturer to enter the assessment details for assessment creation. (S2) 3.2 The system verifies the assessment details to create the assessment. (3E1) 3.3 Assessment is created. 4. The administrator and lecturer select "Update Assessment" in the assessment list page to update the particular assessment. 4.1 System prompts administrator and lecturer to enter the assessment details to update the assessment. (S3) 4.2 The system verifies the assessment details to update the assessment. (4E1)		

4.3 Assessment is updated.

5. Use case ends.

Sub-flows:

S1:

2.1: The system shows the Assessment ID.

2.2: The system shows the Assessment Date.

2.3: The system shows the Assessment Course.

2.4: The system shows the Assessment Info.

2.5: The system shows the Assessment Start Time.

2.6: The system shows the Assessment End Time.

2.7: The system shows the Assessment Room.

S2:

3.1.1: Administrator and lecturer enter the Course of Assessment.

3.1.2: Administrator and lecturer enter the Assessment Info.

3.1.3: Administrator and lecturer enter the Date.

3.1.4: Administrator and lecturer select the Equipment needed.

3.1.5: Administrator and lecturer enter the Start Time.

3.1.6: Administrator and lecturer enter the End Time.

3.1.7: Administrator and lecturer click the "Find Room" button or "Advance Flexible Find Room & Slot" button to find the room for the assessment.

3.1.8: Administrator and lecturer choose the room for the assessment.

S3:

4.1.1: Administrator and lecturer enter the Assessment Info.

4.1.2: Administrator and lecturer enter the Date.

4.1.3: Administrator and lecturer select the Equipment needed.

4.1.4: Administrator and lecturer enter the Start Time.

4.1.5: Administrator and lecturer enter the End Time.

4.1.6: Administrator and lecturer click the "Find Room" button or "Advance Flexible Find Room & Slot" button to find the room for the assessment.

4.1.7: Administrator and lecturer choose the room for the assessment.

Alternate/Exceptional Flows: 3E1 – Invalid assessment details to create assessment. 4E1 – Invalid assessment details to update assessment.

Table 4.12: Use Case for Delete Assessment

Use Case Name: **Delete Assessment**	ID**: UC012**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Administrator, Lecturer – want to delete the assessment.**		
Brief Description**: This use case describes how the system allows administrator and lecturer to delete the assessment.**		
Trigger**: Administrator and lecturer want to delete the assessment.**		
Relationships: Association **: Administrator, Lecturer** Include **: Notify Assessment User** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator and lecturer select "Assessment" in the menu bar to view the assessment. 2. The system shows all the assessment details. (S1) 3. The administrator and lecturer select "Delete" in the assessment list page to delete the particular assessment. 4. System prompts administrator to cancel or confirm to delete the assessment. 5. Assessment is deleted. 6. Use case ends.		
Sub-flows: **S1:** **2.1: The system shows the Assessment ID.** **2.2: The system shows the Assessment Date.** **2.3: The system shows the Assessment Course.** **2.4: The system shows the Assessment Info.** **2.5: The system shows the Assessment Start Time.** **2.6: The system shows the Assessment End Time.**		

2.7: The system shows the Assessment Room.
Alternate/Exceptional Flows:

Table 4.13: Use Case for View Assessment User

Use Case Name**: View Assessment User**	ID**: UC013**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator, Lecturer – want to view the assessment user.**		
Brief Description**: This use case describes how the system allows administrator and lecturer to view the assessment user.**		
Trigger**: Administrator and lecturer want to view the assessment user.**		
Relationships: Association **: Administrator, Lecturer** Include **: N/A** Extend **: Notify Assessment User** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator and lecturer select "Assessment" in the menu bar to view the assessment. 2. The system shows all the assessment details. (S1) 3. The administrator and lecturer select "View Student" on the assessment list page to view the users enroll in particular assessment. 4. The system shows all the users enrolled in the assessment. (S2) 5. Use case ends.		
Sub-flows: **S1:** **2.1: The system shows the Assessment ID.** **2.2: The system shows the Assessment Date.** **2.3: The system shows the Assessment Course.** **2.4: The system shows the Assessment Info.** **2.5: The system shows the Assessment Start Time.** **2.6: The system shows the Assessment End Time.** **2.7: The system shows the Assessment Room.** **S2:**		

4.1: The system shows the User ID. **4.2: The system shows the Username.** **4.3: The system shows the Email.**
Alternate/Exceptional Flows:

Table 4.14: Use Case for Notify Assessment User

Use Case Name**: Notify Assessment User**	ID**: UC014**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator, Lecturer – want to notify the assessment user.**		
Brief Description**: This use case describes how the system allows administrator and lecturer to notify the assessment user.**		
Trigger**: Administrator and lecturer want to notify the assessment user.**		
Relationships: Association **: Administrator, Lecturer** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator and lecturer select "Assessment" in the menu bar to view the assessment. 2. The system shows all the assessment details. (S1) 3. The administrator and lecturer select "Notify User" on the assessment list page to notify the student enroll in particular assessment. 4. System prompts administrator and lecturer to enter the notification details to notify the user. (S2) 5. The system notifies all the students enrolled in the assessment through email. 6. Use case ends.		
Sub-flows: **S1:** **2.1: The system shows the Assessment ID.** **2.2: The system shows the Assessment Date.** **2.3: The system shows the Assessment Course.** **2.4: The system shows the Assessment Info.** **2.5: The system shows the Assessment Start Time.**		

2.6: The system shows the Assessment End Time. **2.7: The system shows the Assessment Room.** **S2:** **4.1: Administrator and lecturer enter the message** **4.2: Administrator and lecturer tick to attach assessment details or not**
Alternate/Exceptional Flows:

Table 4.15: Use Case for Find Room

Use Case Name**: Find Room**	ID**: UC015**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator, Lecturer – want to find the room for assessment.**		
Brief Description**: This use case describes how the system allows administrator and lecturer to find the room for assessment.**		
Trigger**: Administrator and lecturer want to find the room for assessment.**		
Relationships: Association **: Administrator, Lecturer** Include **: View Room** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator and lecturer select "Find Room" button in the form of assessment creation or update page to find the room for the assessment. 2. The system shows all the rooms available for the assessment. (S1) 3. The administrator and lecturer select the desired room for the assessment. 4. Use case ends.		
Sub-flows: **S1:** **2.1: The system shows the Room ID.** **2.2: The system shows the Room Type.** **2.3: The system shows the Room Equipment Availability.**		
Alternate/Exceptional Flows:		

Table 4.16: Use Case for Advance Flexible Find Slot and Room

Use Case Name**: Advance Flexible Find Slot and Room**	ID**: UC016**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Administrator, Lecturer – want to advance flexible finds the slot and room for assessment.**		
Brief Description**: This use case describes how the system allows administrator and lecturer to advance flexible finds the slot and room for assessment.**		
Trigger**: Administrator and lecturer want to advance flexible finds the slot and room for assessment.**		
Relationships: Association **: Administrator, Lecturer** Include **: View Room** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator and lecturer select “Advance Flexible Find Slot & Room” button in the form of assessment creation or update page to find the room for the assessment. 2. The System prompts administrator and lecturer to enter the advance searching criteria for searching the slot and room. (S1) 3. The system verifies the searching criteria to search the slot and room. (3E1) 4. The system shows all the slots and rooms available for the assessment. (S2) 5. The administrator and lecturer select the desired room for the assessment. 6. Use case ends.		
Sub-flows: **S1:**		

2.1: Administrator and lecturer enter the Time From. **2.2: Administrator and lecturer enter the Time To.** **2.3: Administrator and lecturer enter the Duration Hour.** **2.4: Administrator and lecturer enter the Duration Minute.** **S2:** **4.1: The system shows the list of slots.** **4.2: The system shows the Room ID for each slot.** **4.3: The system shows the Room Type for each slot.** **4.4: The system shows the Room Capacity for each slot.** **4.4: The system shows the Room Equipment Availability for each slot.**
Alternate/Exceptional Flows: 3E1 – Invalid searching criteria to search the slot and room.

Table 4.17: Use Case for Online Chat/Communication

Use Case Name**: Online Chat/Communication**	ID**: UC017**	Importance Level**: High**
Primary Actor**: Administrator, Lecturer**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Administrator, Lecturer – want to online communicate with each other.**		
Brief Description**: This use case describes how the system allows administrator and lecturer to online communicate with each other.**		
Trigger**: Administrator and lecturer want to online communicate with each other.**		
Relationships: Association **: Administrator, Lecturer** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The administrator selects "Chat" in the menu bar to view the chat room. 1.1 The system shows all the chat room details. (S1) 1.2 The administrator selects "Enter Chat Room" in the chat room list page to enter the particular chat room to online communicate with each other. 2. The lecturer selects "Chat" in the menu bar to create lecturer's own chat room. 3. Use case ends.		
Sub-flows: **S1:** **1.1.1: The system shows the list of chat rooms with the Chat Room ID.**		
Alternate/Exceptional Flows:		

Table 4.18: Use Case for View Calendar

Use Case Name**: View Calendar**	ID**:** **UC018**	Importance Level**: High**
Primary Actor**: Student, Lecturer**		Use Case Type**: Detail, Essential**
Stakeholders and Interests**:** **Student and Lecturer– want to view the assessment schedule in the form of calendar.**		
Brief Description**: This use case describes how the system allows student and lecturer to view the assessment schedule in the form of calendar.**		
Trigger**: Student and lecturer want to view the assessment schedule in the form of calendar.**		
Relationships: Association **: Student, Lecturer** Include **: N/A** Extend **: N/A** Generalization **: N/A**		
Normal Flow of Events: 1. The student and lecturer select "Calendar" in the menu bar to view the assessment schedule in the form of calendar. 2. The system shows all the assessment details in the form of calendar. (S1) 3. Use case ends.		
Sub-flows: **S1:** **2.1: The system shows the Assessment Date.** **2.2: The system shows the Assessment Start Time.** **2.3: The system shows the Assessment End Time.** **2.4: The system shows the Assessment Info.** **2.5: The system shows the Assessment Room.**		
Alternate/Exceptional Flows:		

4.4 Prototype Development

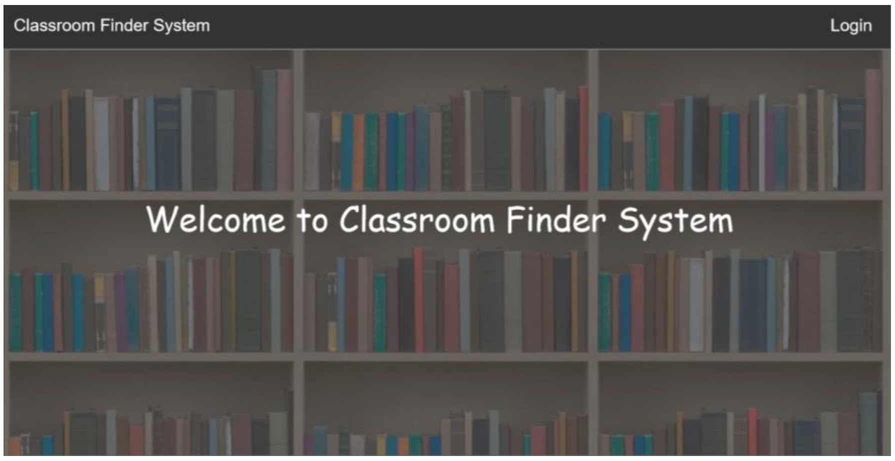


Figure 4.2: Prototype - Home Page

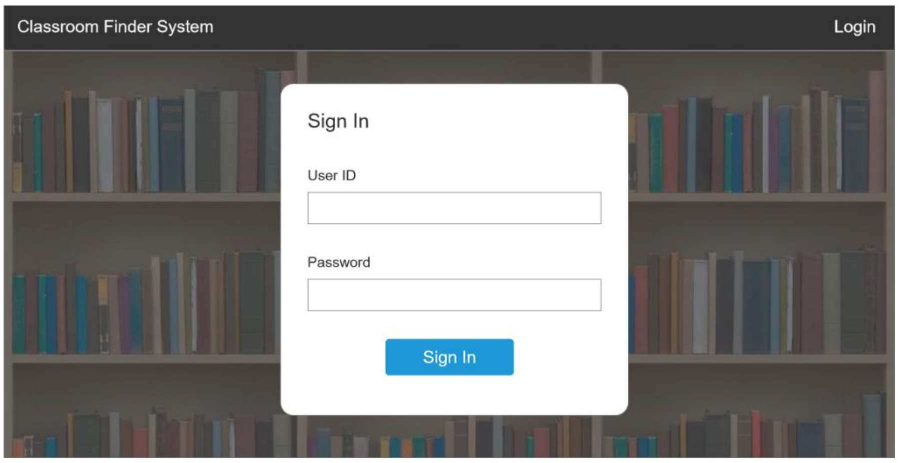


Figure 4.3: Prototype - Login Page

Student

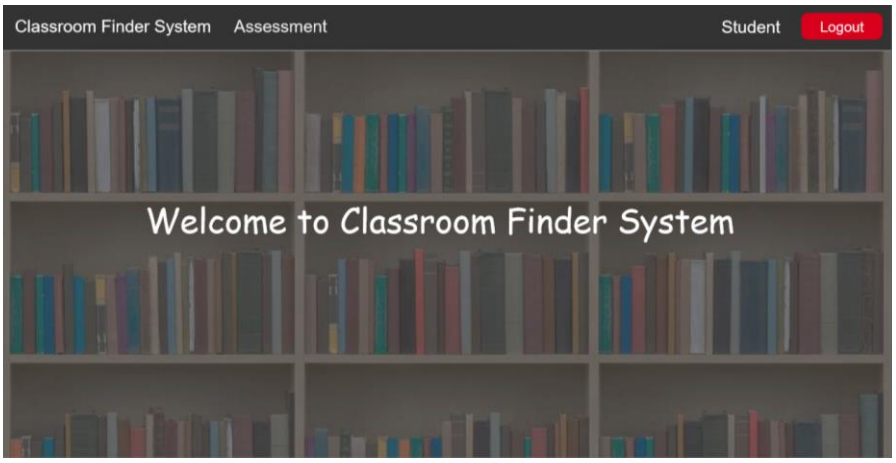
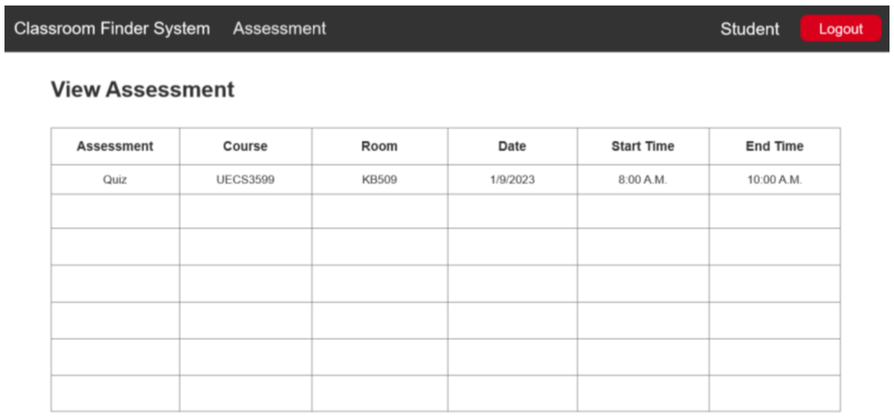


Figure 4.4: Prototype - Student Home Page



View Assessment

Assessment	Course	Room	Date	Start Time	End Time
Quiz	UECS3599	KB509	1/9/2023	8:00 A.M.	10:00 A.M.

Figure 4.5: Prototype - Student View Assessment

Lecturer

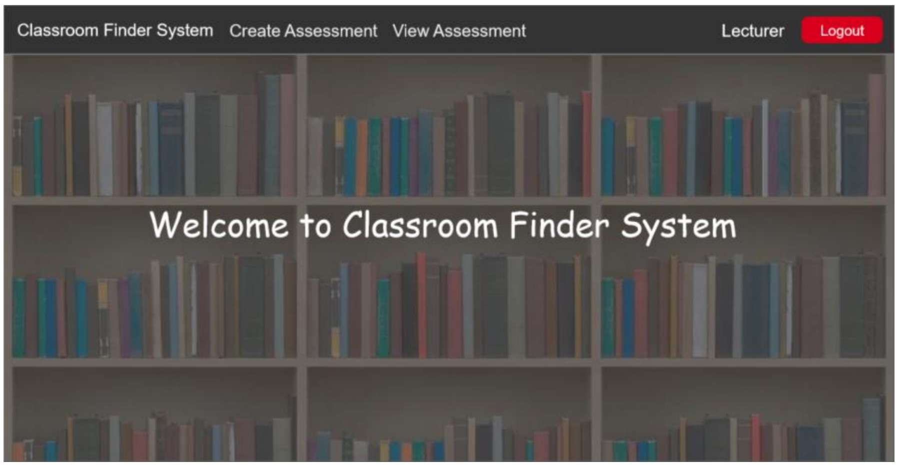
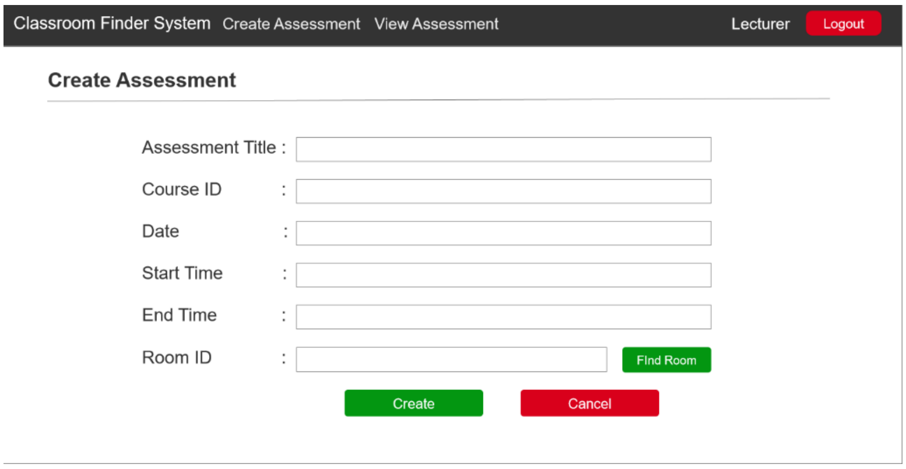


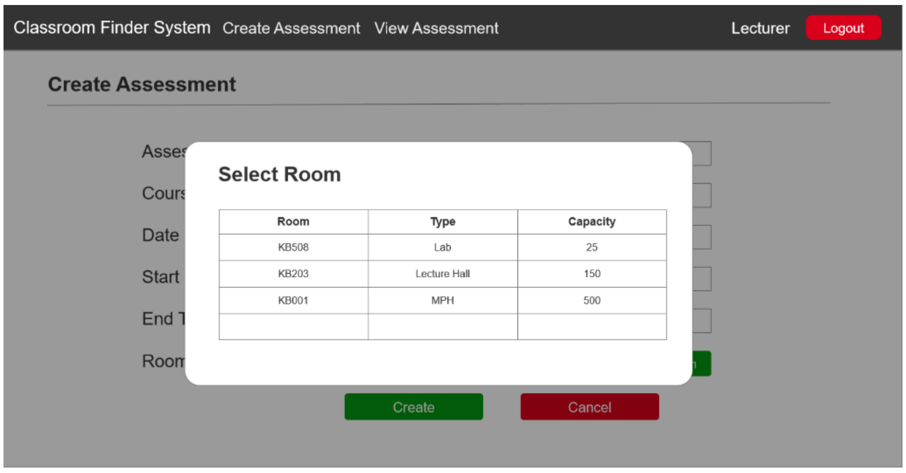
Figure 4.6: Prototype - Lecturer Home Page



Create Assessment

Assessment Title :	
Course ID :	
Date :	
Start Time :	
End Time :	
Room ID :	

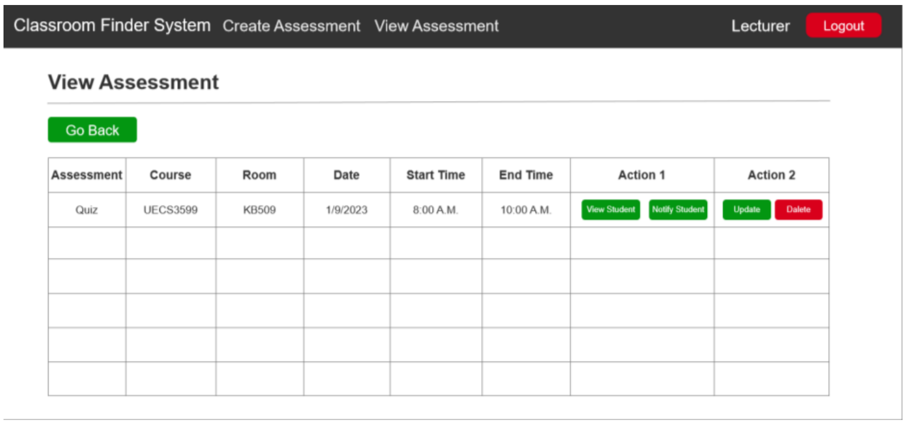
Figure 4.7: Prototype - Lecturer Create Assessment



Select Room

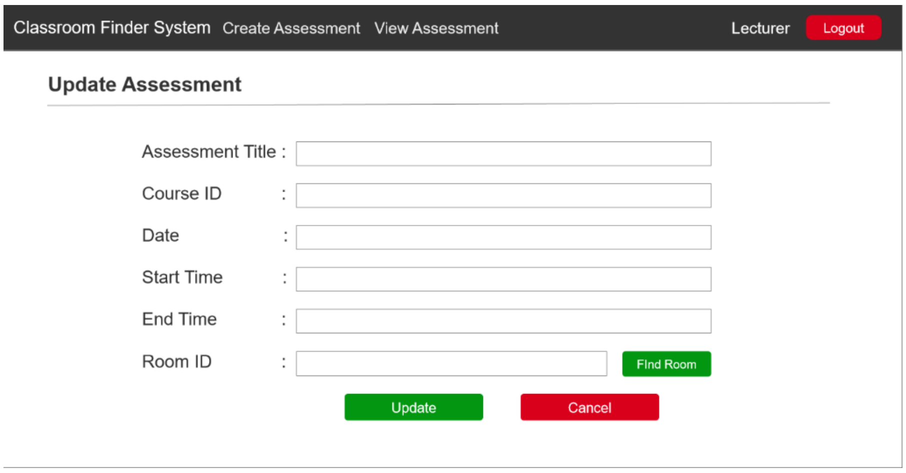
Room	Type	Capacity
KB508	Lab	25
KB203	Lecture Hall	150
KB001	MPH	500

Figure 4.8: Prototype - Lecturer View Room



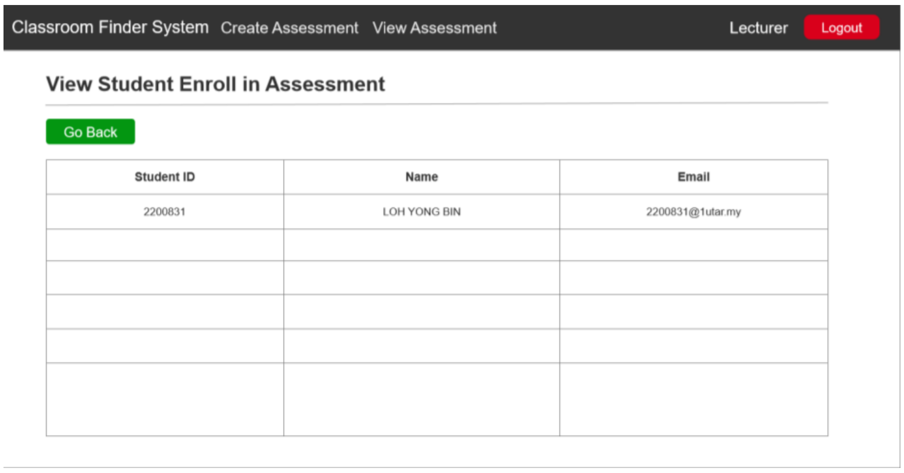
Assessment	Course	Room	Date	Start Time	End Time	Action 1	Action 2
Quiz	UECS3599	KB509	1/9/2023	8:00 A.M.	10:00 A.M.	View Student Notify Student	Update Delete

Figure 4.9: Prototype - Lecturer View Assessment



Assessment Title :	
Course ID :	
Date :	
Start Time :	
End Time :	
Room ID :	

Figure 4.10: Prototype - Lecturer Update Assessment



Student ID	Name	Email
2200831	LOH YONG BIN	2200831@1utar.my

Figure 4.11: Prototype - Lecturer View Student

<u>Administrator</u>

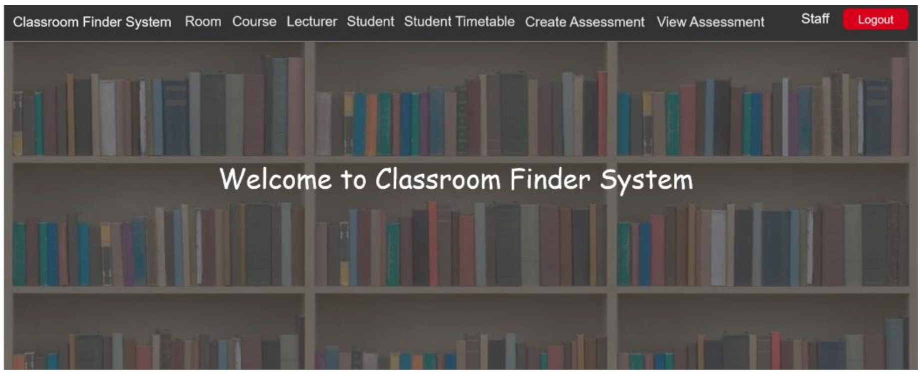
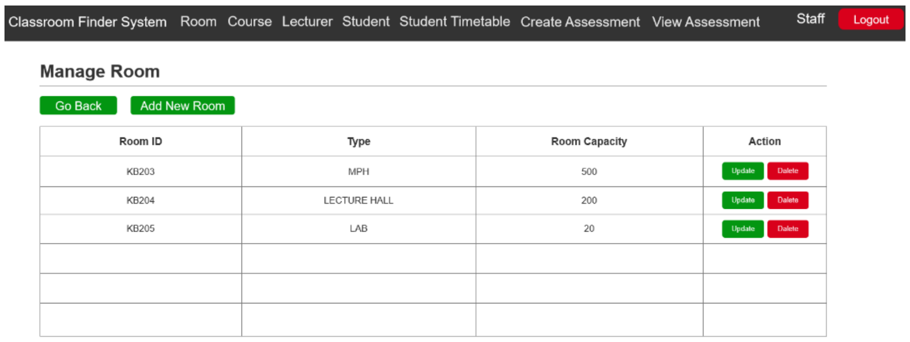


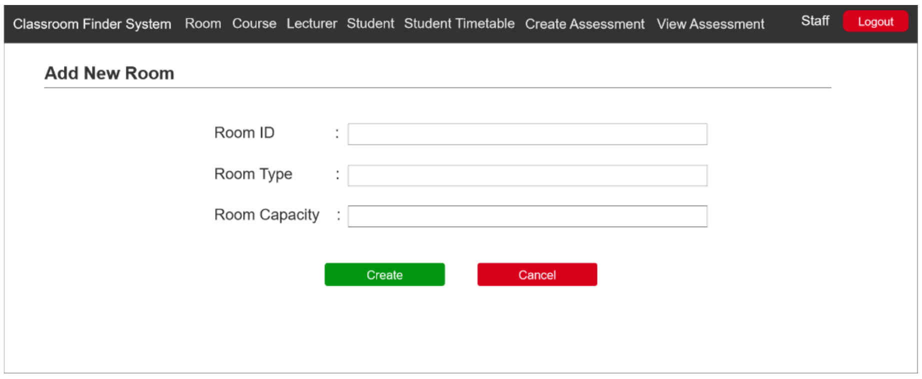
Figure 4.12: Prototype - Admin Home Page



Manage Room

Room ID	Type	Room Capacity	Action
KB203	MPH	500	Update Delete
KB204	LECTURE HALL	200	Update Delete
KB205	LAB	20	Update Delete

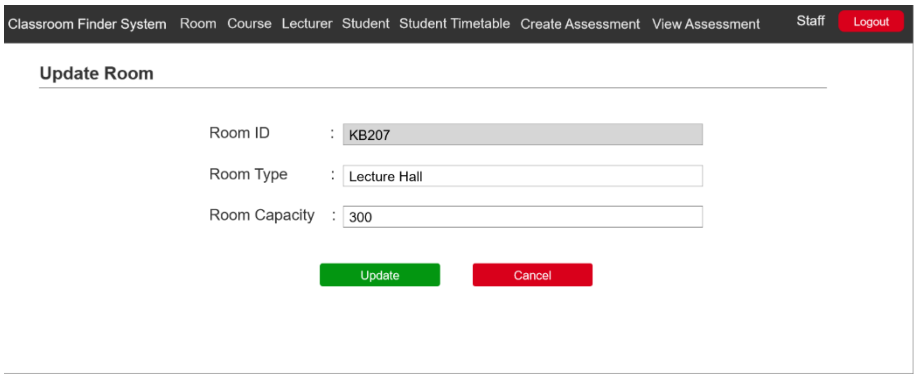
Figure 4.13: Prototype - Admin Manage Room



Add New Room

Room ID	:
Room Type	:
Room Capacity	:

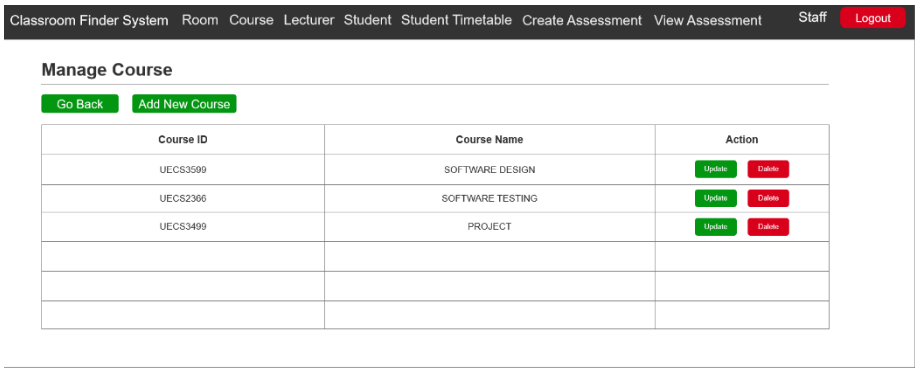
Figure 4.14: Prototype - Admin Add Room



Update Room

Room ID	:	KB207
Room Type	:	Lecture Hall
Room Capacity	:	300

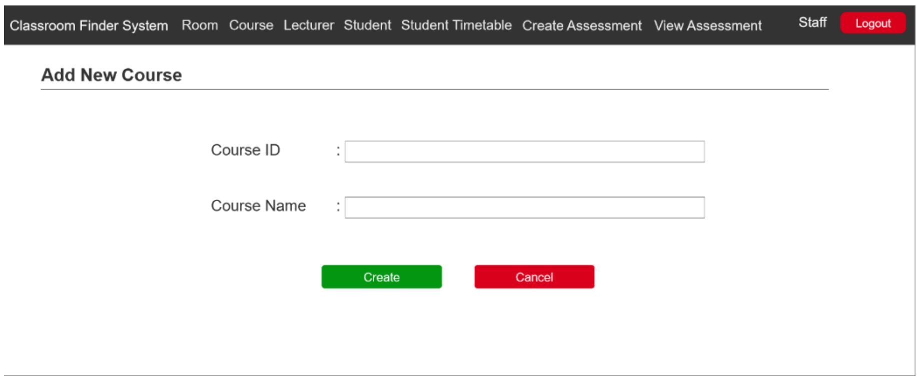
Figure 4.15: Prototype - Admin Update Room



Manage Course

Course ID	Course Name	Action
UECS3599	SOFTWARE DESIGN	Update Delete
UECS2366	SOFTWARE TESTING	Update Delete
UECS3499	PROJECT	Update Delete

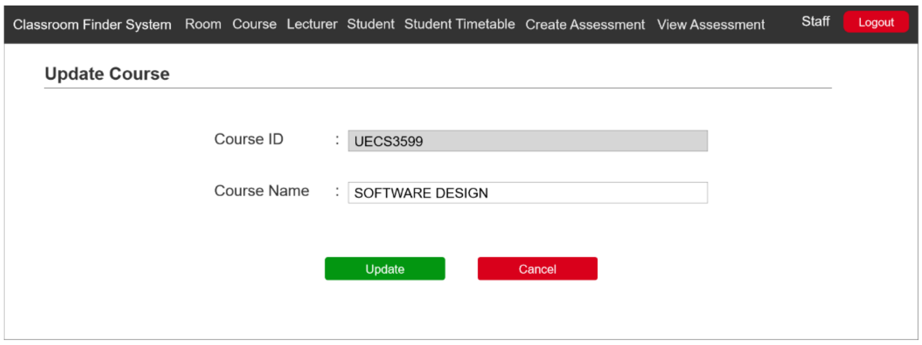
Figure 4.16: Prototype - Admin Manage Course



Add New Course

Course ID	:	
Course Name	:	

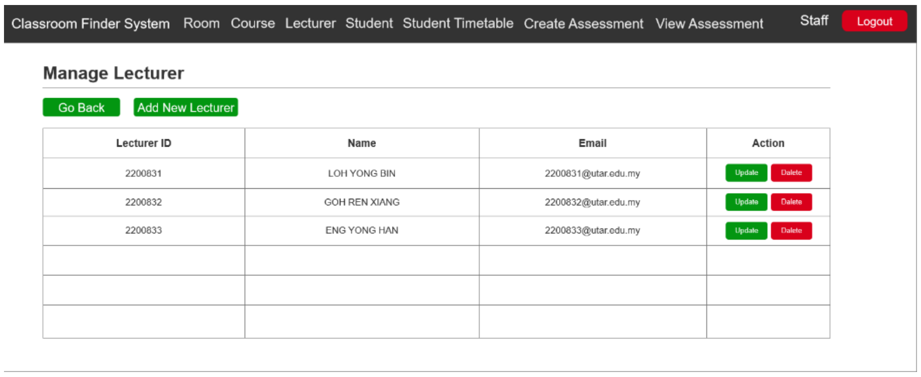
Figure 4.17: Prototype - Admin Add Course



Update Course

Course ID	:	UECS3599
Course Name	:	SOFTWARE DESIGN

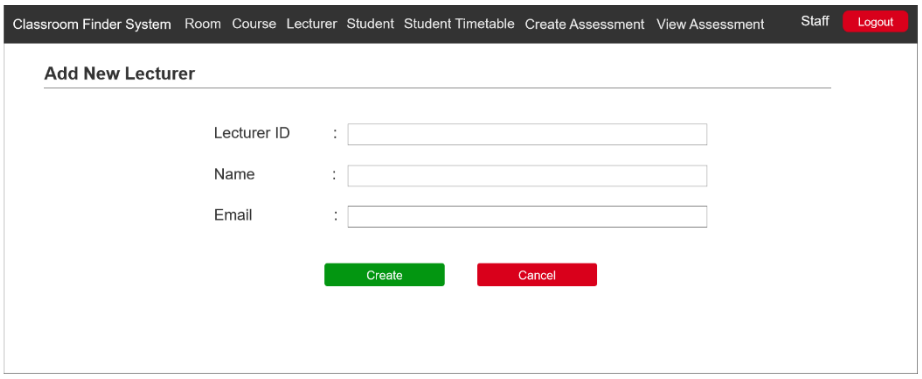
Figure 4.18: Prototype - Admin Update Course



Manage Lecturer

Lecturer ID	Name	Email	Action
2200831	LOH YONG BIN	2200831@utar.edu.my	Update Delete
2200832	GOH REN XIANG	2200832@utar.edu.my	Update Delete
2200833	ENG YONG HAN	2200833@utar.edu.my	Update Delete

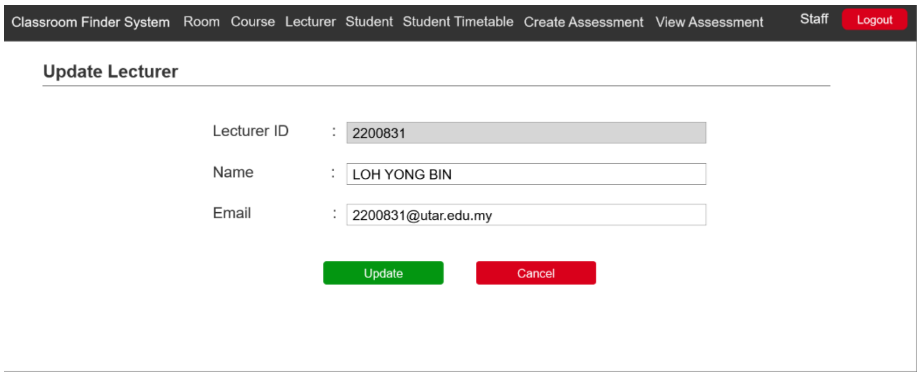
Figure 4.19: Prototype - Admin Manage Lecturer



Add New Lecturer

Lecturer ID	:	
Name	:	
Email	:	

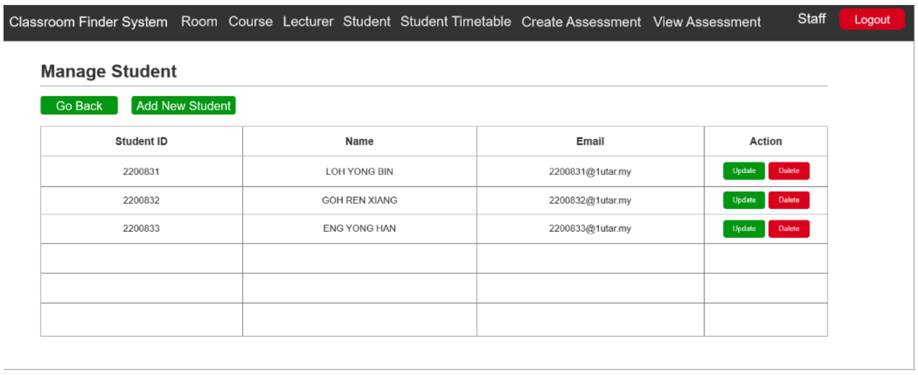
Figure 4.20: Prototype - Admin Add Lecturer



Update Lecturer

Lecturer ID	:	2200831
Name	:	LOH YONG BIN
Email	:	2200831@utar.edu.my

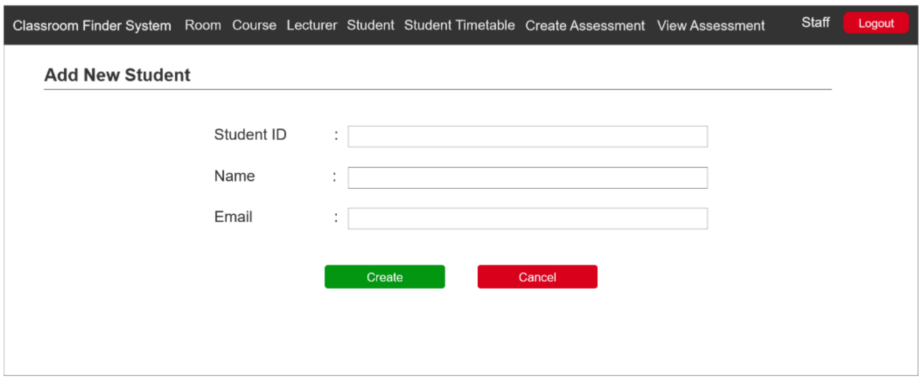
Figure 4.21: Prototype - Admin Update Lecturer



Manage Student

Student ID	Name	Email	Action
2200831	LOH YONG BIN	2200831@1utar.my	Update Delete
2200832	GOH REN XIANG	2200832@1utar.my	Update Delete
2200833	ENG YONG HAN	2200833@1utar.my	Update Delete

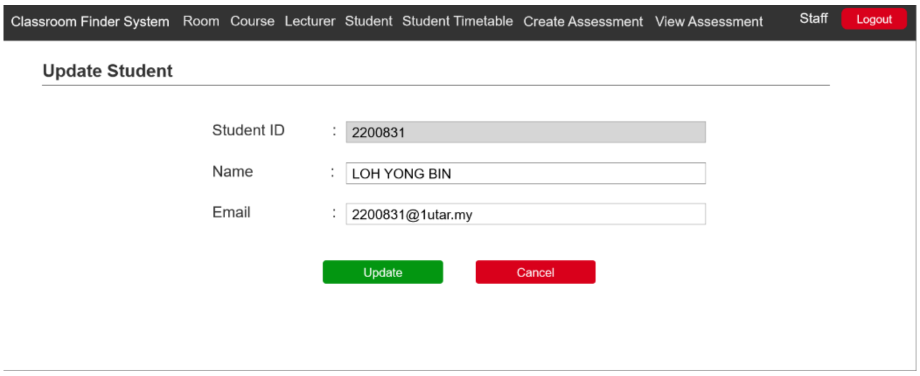
Figure 4.22: Prototype - Admin Manage Student



Add New Student

Student ID	:	
Name	:	
Email	:	

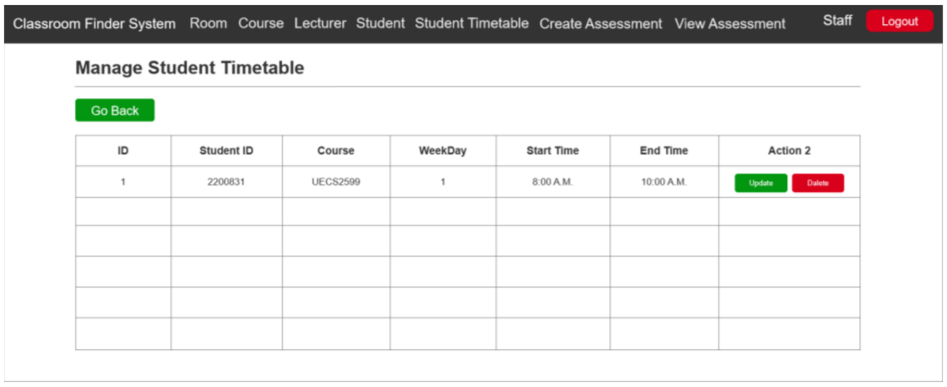
Figure 4.23: Prototype - Admin Add Student



Update Student

Student ID	:	2200831
Name	:	LOH YONG BIN
Email	:	2200831@1utar.my

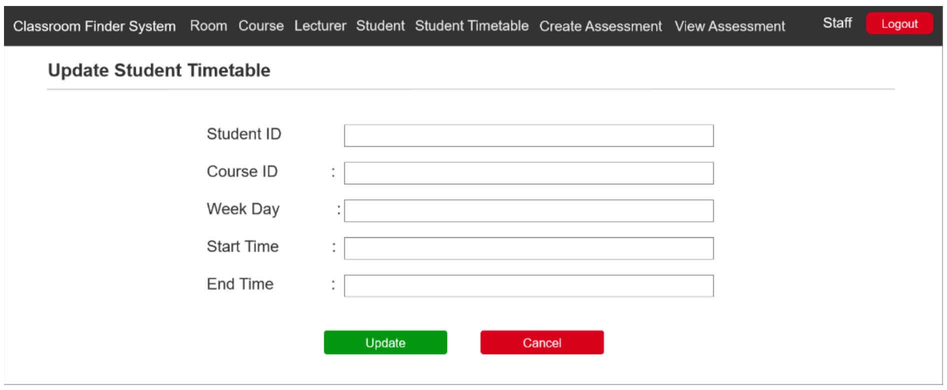
Figure 4.24: Prototype - Admin Update Student



Manage Student Timetable

ID	Student ID	Course	WeekDay	Start Time	End Time	Action 2
1	2200831	UECS2599	1	8:00 A.M.	10:00 A.M.	Update Delete

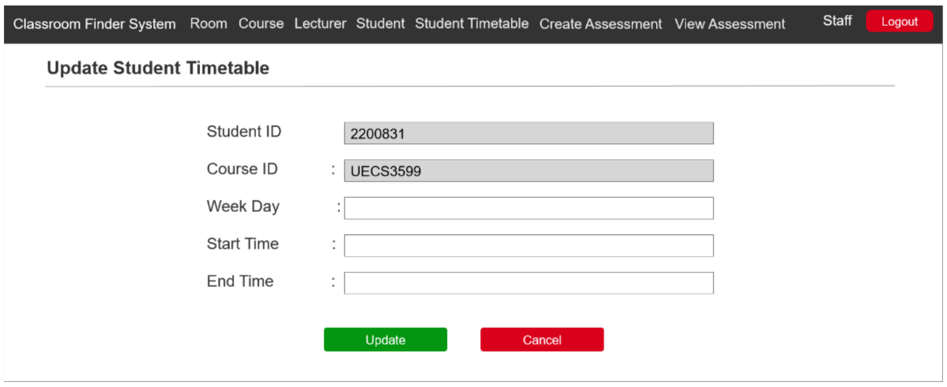
Figure 4.25: Prototype - Admin Manage Student Timetable



Update Student Timetable

Student ID	
Course ID :	
Week Day :	
Start Time :	
End Time :	

Figure 4.26: Prototype - Admin Add Student Timetable



Update Student Timetable

Student ID	2200831
Course ID :	UECS3599
Week Day :	
Start Time :	
End Time :	

Figure 4.27: Prototype - Admin Update Student Timetable

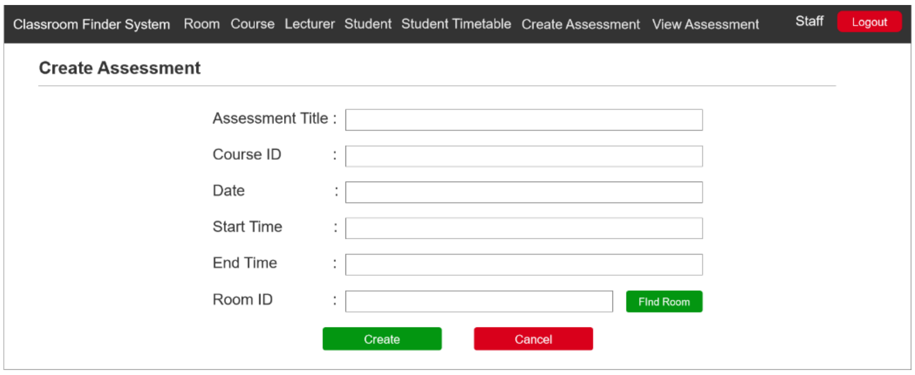
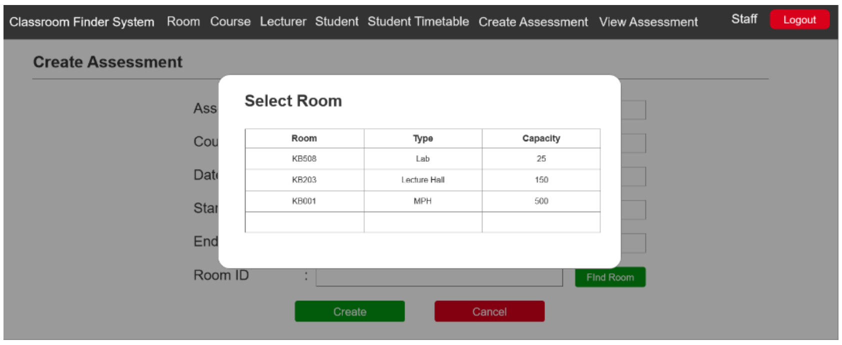
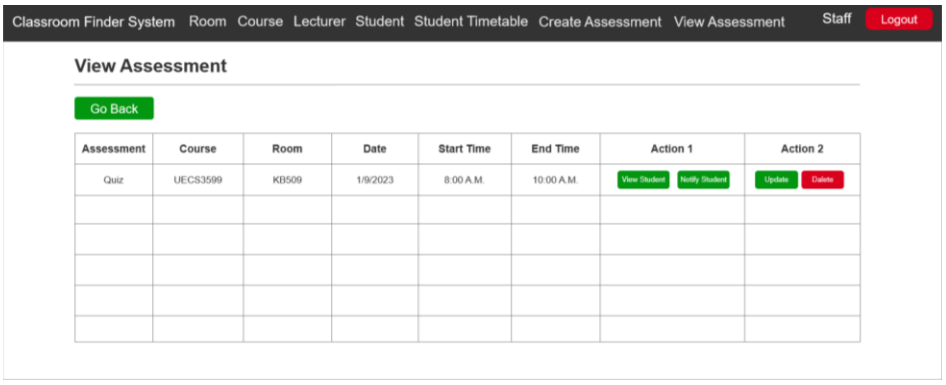


Figure 4.28: Prototype - Admin Create Assessment



Room	Type	Capacity
KB508	Lab	25
KB203	Lecture Hall	150
KB001	MPH	500

Figure 4.29: Prototype - Admin Find Room



Assessment	Course	Room	Date	Start Time	End Time	Action 1	Action 2
Quiz	UECS3599	KB509	1/9/2023	8:00 A.M.	10:00 A.M.	View Student Notify Student	Update Delete

Figure 4.30: Prototype - Admin View Assessment

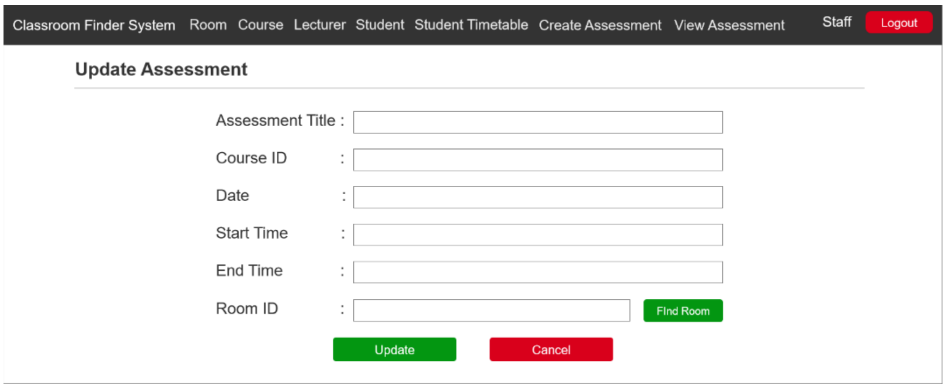
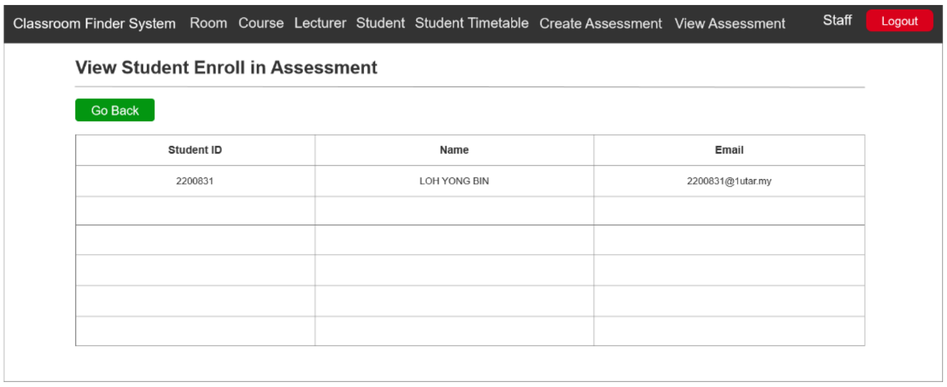


Figure 4.31: Prototype - Admin Update Assessment



View Student Enroll in Assessment

Student ID	Name	Email
2200831	LOH YONG BIN	2200831@1utar.my

Figure 4.32: Prototype - Admin View Student Enrol in Assessment

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

This chapter delves into the critical aspect of system design for the Classroom Finder System which is emphasizing the necessity for a robust design to ensure scalability, efficiency and reliability. It outlines the system architectural design and translates the system requirements into comprehensive design modelling diagrams which include data flow diagrams (DFDs), entity-relationship diagrams (ERDs), class diagrams, activity diagrams and use case diagrams. Not only these, but this chapter also presents user interface design which are aligned with use cases and user workflows to ensure an intuitive and efficient user experience.

5.2 System Architecture Design

In this project, a widely adopted three-tier architecture was implemented. This architecture divides the system into three distinct layers which are the presentation tier, the application tier and the data tier. At the top is the presentation tier which is responsible for managing user interactions and interface communication. It collects user inputs, processes them and forwards the operations to the application tier. The application tier acts as the core engine of the system to handle business logic and orchestrate operations received from the presentation tier. It then communicates with the data tier to retrieve or update information as needed. The data tier which situated at the bottom is in charge of storing and managing data, executing queries and responding to requests from the application tier. This three-tier approach offers several advantages that include enhanced flexibility, scalability and security. By separating concerns into distinct layers, the architecture enables easier maintenance, updates and modifications without disrupting other parts of the system. Additionally, it facilitates robust security management by allowing different tiers to implement authorization mechanisms tailored to specific user roles and permissions. By adopting this architecture, the project aims to establish a solid foundation for the efficient development and seamless operation of the Classroom Finder System.

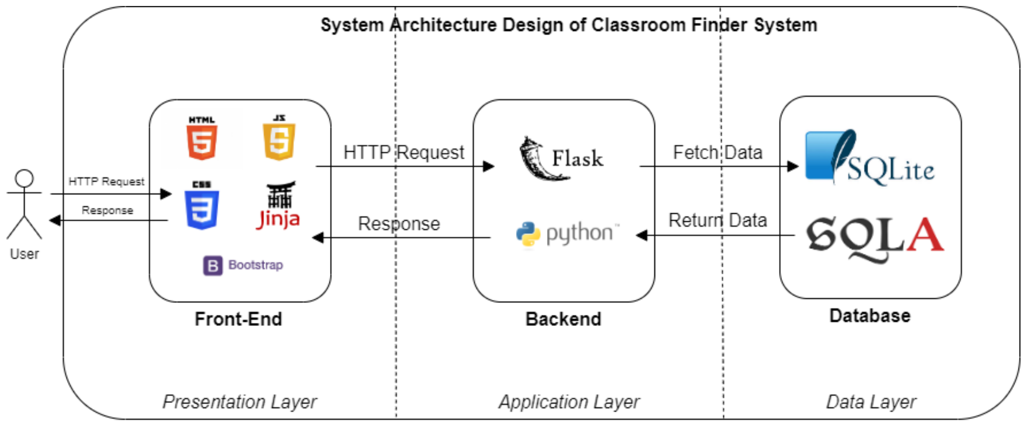


Figure 5.1: System Architecture Design

The system architecture of the Classroom Finder System is carefully crafted to ensure efficiency, scalability and maintainability. Based on figure 5.1, it follows a three-tier layer approach which consists of the presentation layer, application layer and data layer while each layer is fulfilling specific roles and responsibilities.

At the top of the architecture stack is the presentation layer which serves as the interface between users and the application. This layer is responsible for rendering the user interface and handling user interactions. It includes a combination of front-end technologies such as HTML, CSS, JavaScript, and the Jinja template engine which is integrated with the Flask framework. Additionally, the Bootstrap framework is utilized to enhance the visual aesthetics and responsiveness of the user interface. When users access the system through their web browsers, their HTTP requests are received by the presentation layer. This layer processes the requests, validates the user inputs and prepares the necessary data to be sent to the application layer for further processing.

Situated between the presentation and data layers, the application layer houses the core logic and functionality of the Classroom Finder System. The application layer contains the backend components which responsible to handle incoming requests from the presentation layer, execute business logic and interact with the data layer to retrieve or manipulate data as needed. Flask, a lightweight and versatile Python web framework also is utilized to develop the backend of the system. Flask provides a robust foundation for building web applications, offering features such as routing, request handling and template rendering. Within the application layer, Python scripts and modules implement the various functionalities of the system which include user authentication, room allocation algorithms and user, room, course, schedule, enrolment and assessment management. The application layer also orchestrates the flow of data and logic to ensure that the user requests are processed accurately and efficiently.

At the bottom of the architecture stack lies the data layer which is responsible for managing the storage and retrieval of data used by the system. This layer comprises a relational database management system (RDBMS) implemented using SQLite, a lightweight and self-contained database engine. Moreover, SQLAlchemy, a powerful ORM library for Python is utilized to interact with the SQLite database and perform database operations such as querying, inserting, updating and deleting data. The data layer stores information related to users, classrooms, courses, schedules, enrolment and assessments to ensure that data is organized and accessible for efficient system operation. The application layer also communicates with the data layer to fetch or modify data based on user requests to achieve the goals of maintaining data integrity and consistency throughout the system.

In conclusion, the three-tier architecture design of the Classroom Finder System offers several benefits. It promotes separation of concerns to allow for modular development and easier maintenance of the system components. The use of popular frameworks and technologies such as Flask, SQLite and SQLAlchemy ensure compatibility, reliability and extensibility. Additionally, the architecture facilitates scalability and enables the system to accommodate growing user demands and evolving requirements over time. By adopting this architectural approach, the Classroom Finder System is well-equipped to deliver a seamless and intuitive user experience while maintaining robustness and performance.

5.3 System Design Modelling

System design modelling plays a crucial role in the development of the Classroom Finder System as it is providing a comprehensive blueprint for the system architecture and functionality. This section outlines the various modelling techniques employed to translate system requirements into concrete design specifications. By leveraging a combination of data flow diagrams (DFDs), entity-relationship diagrams (ERDs), class diagrams, activity diagrams and use case diagrams, the system design modelling section aims to encapsulate the system's structure, behaviour and user interaction in a systematic and intuitive manner.

5.3.1 Entity-Relationship Diagram (ERD)

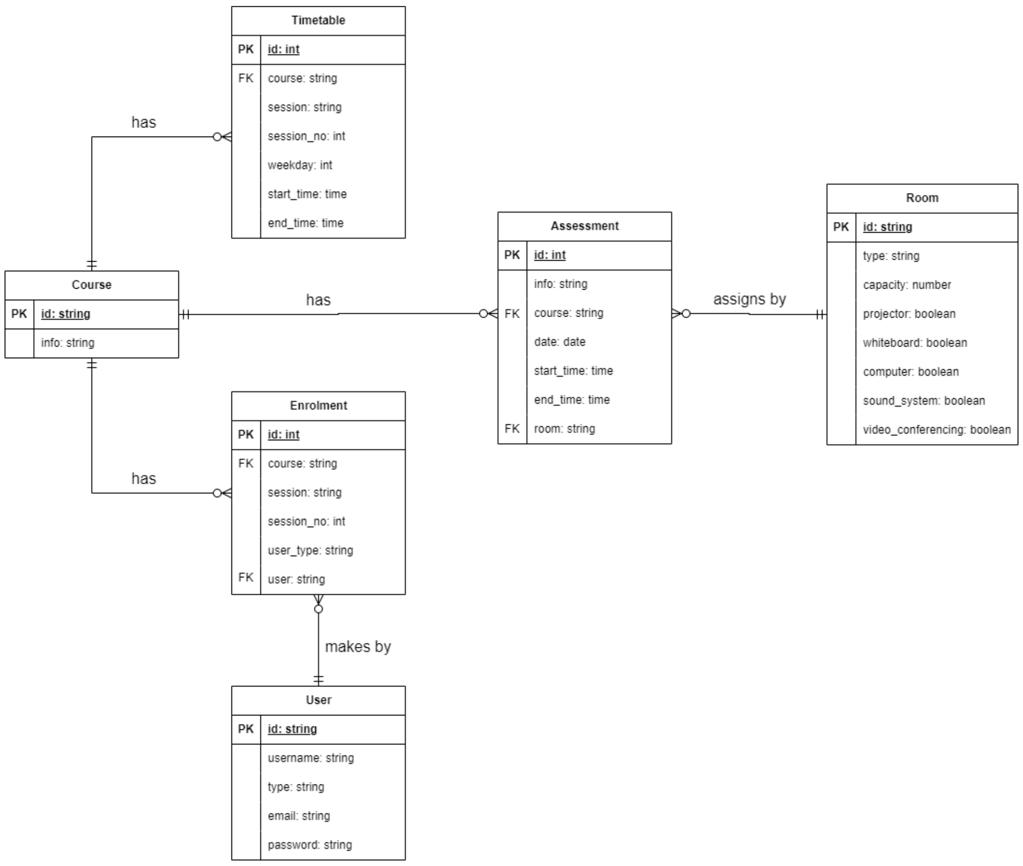


Figure 5.2: Entity-Relationship Diagram

Entity-Relationship Diagram (ERD) depicts the relationships between different entities or objects in the Classroom Finder System which emphasizes the structure of the system's data model. ERD consists of entities (representing real-world objects), attributes (properties of entities) and relationships (connections between entities). By visualizing the data model through ERD, stakeholders can understand the system's data structure and identify the entities, attributes and relationships essential for system functionality.

5.3.2 Class Diagram

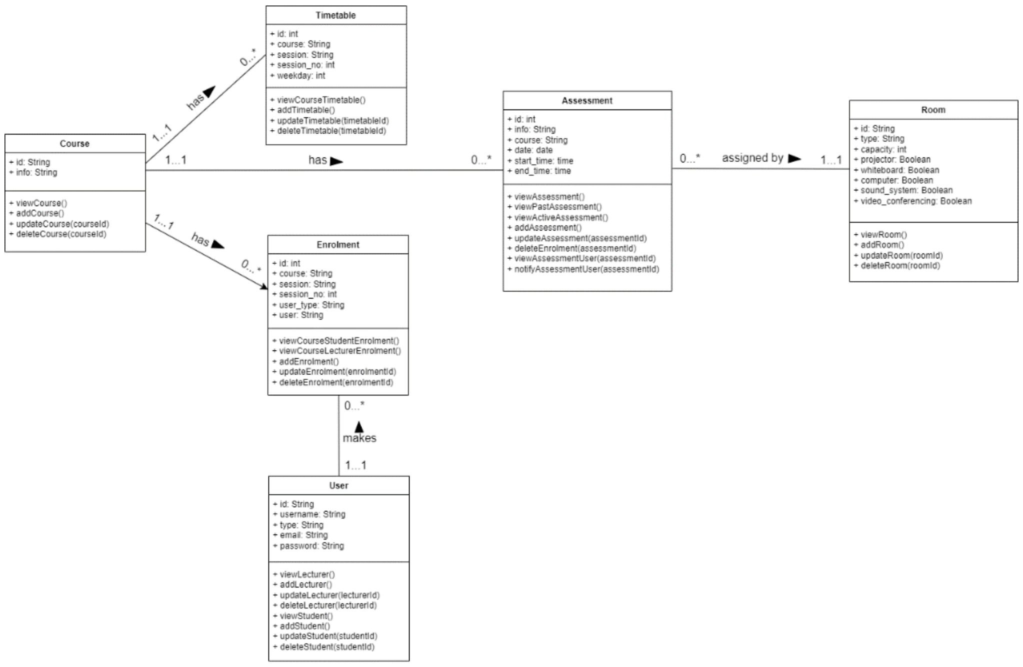


Figure 5.3: Class Diagram

Class Diagram provides a static view of the system's object-oriented design which showcases the classes, attributes, methods and relationships within the system. Class Diagram is instrumental in defining the system's object-oriented architecture, delineating the components and interactions that constitute the system's functionality. By mapping out the class structure and relationships, Class Diagrams facilitate the implementation of object-oriented programming principles and support the development of robust and maintainable software components.

5.3.3 Data Flow Diagrams (DFDs)

Data Flow Diagrams (DFDs) serve as a visual representation of the flow of data within the Classroom Finder System which illustrates how information moves between processes, data stores and external entities. The DFDs provide insights into the system's data processing logic to highlight the data transformations and interactions at different levels of abstraction. The key levels of DFDs include the context diagram, level 1 DFD and level 2 DFD while each offer progressively detailed views of the system's data flow and processing.

5.3.3.1 Context Diagram

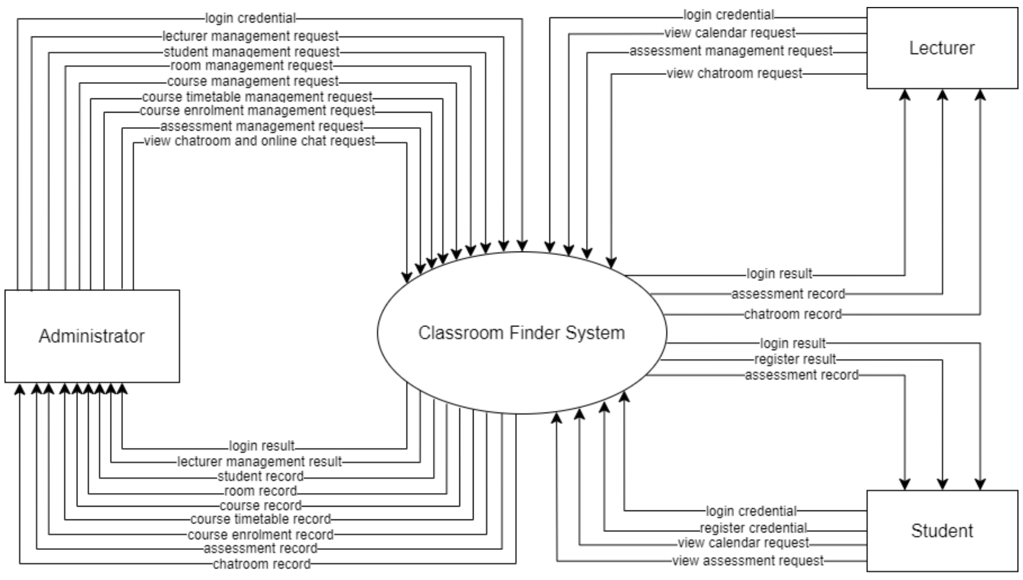


Figure 5.4: Context Diagram

5.3.3.2 Level 1 Data Flow Diagram

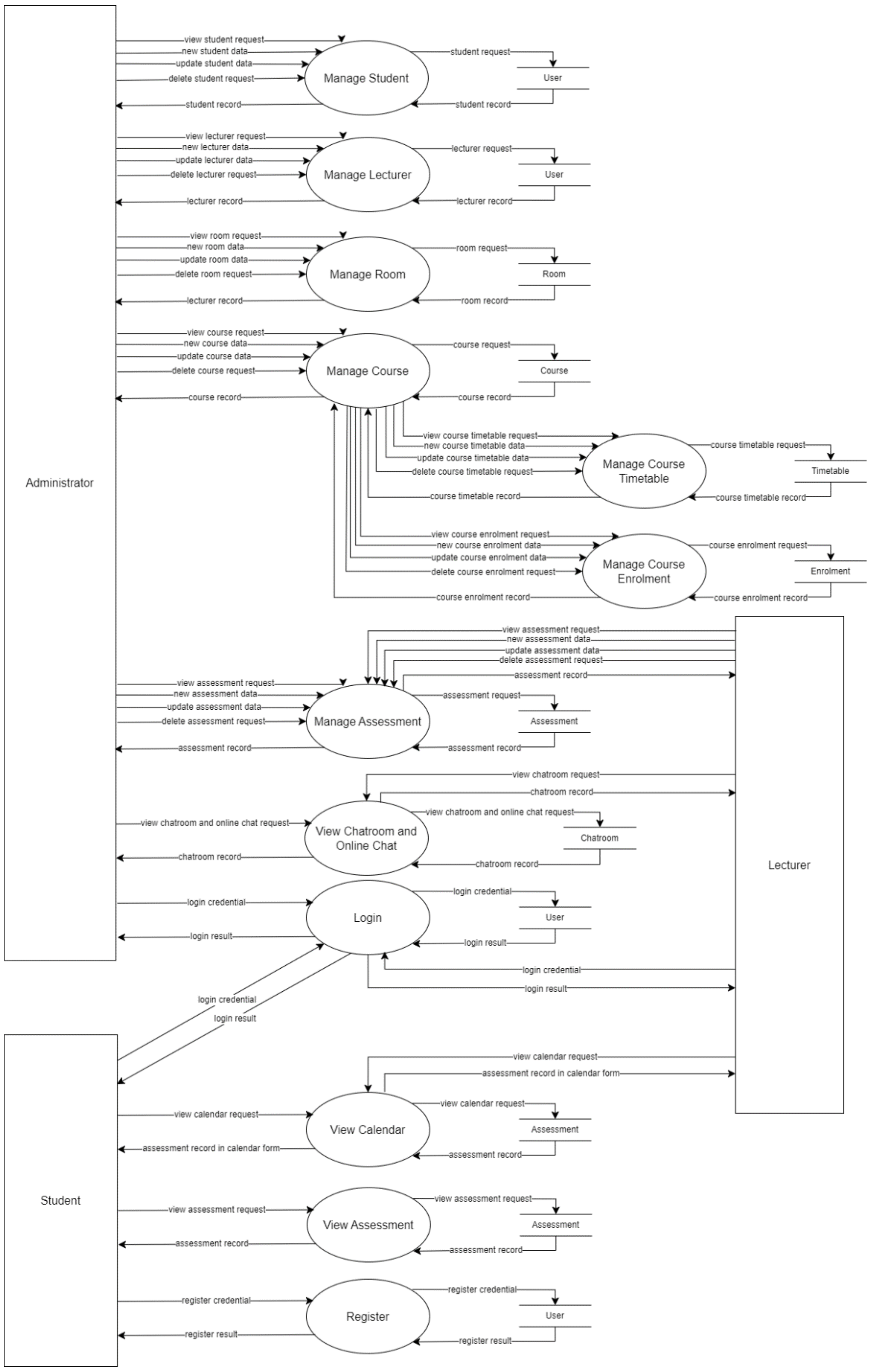


Figure 5.5: Level 1 Data Flow Diagram

5.3.3.3 Level 2 Data Flow Diagram

5.3.3.3.1Manage Lecturer

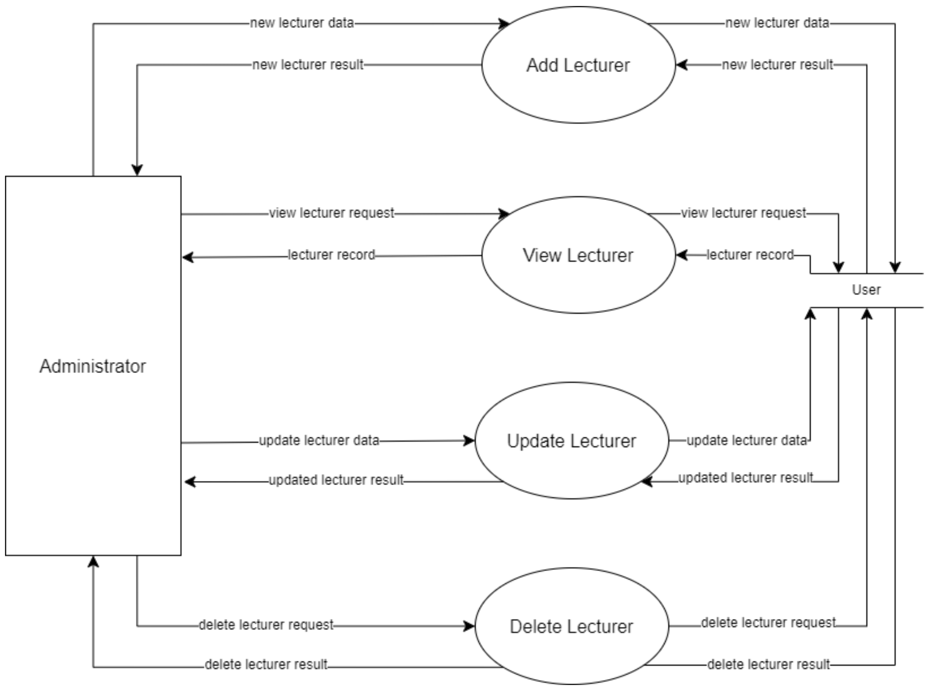


Figure 5.6: Level 2 DFD (Manage Lecturer)

5.3.3.3.2Manage Student

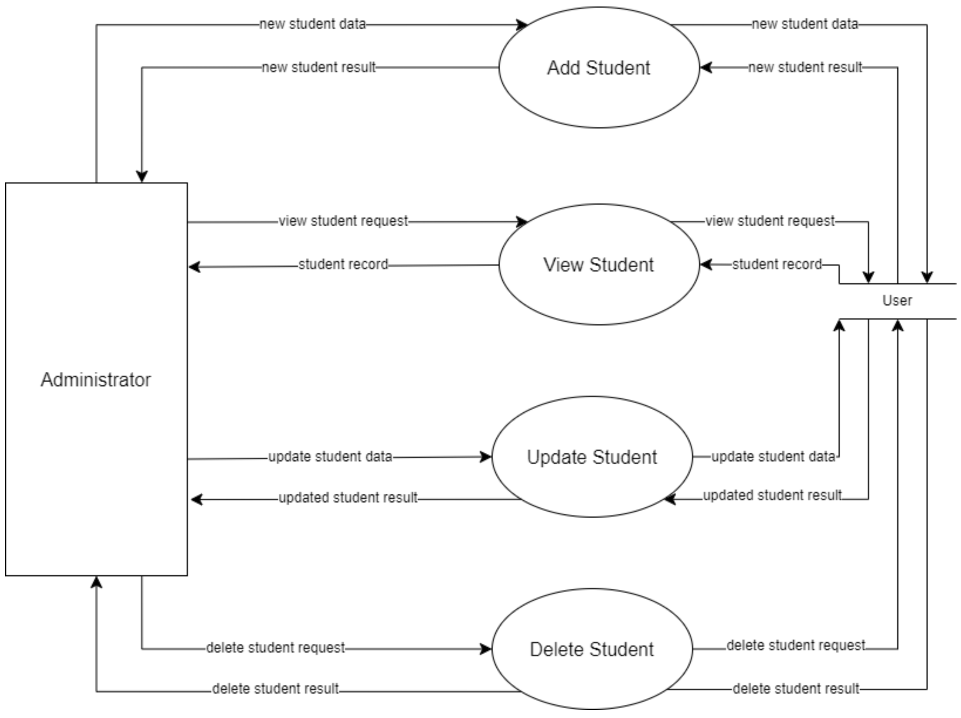


Figure 5.7: Level 2 DFD (Manage Student)

5.3.3.3.3Manage Room

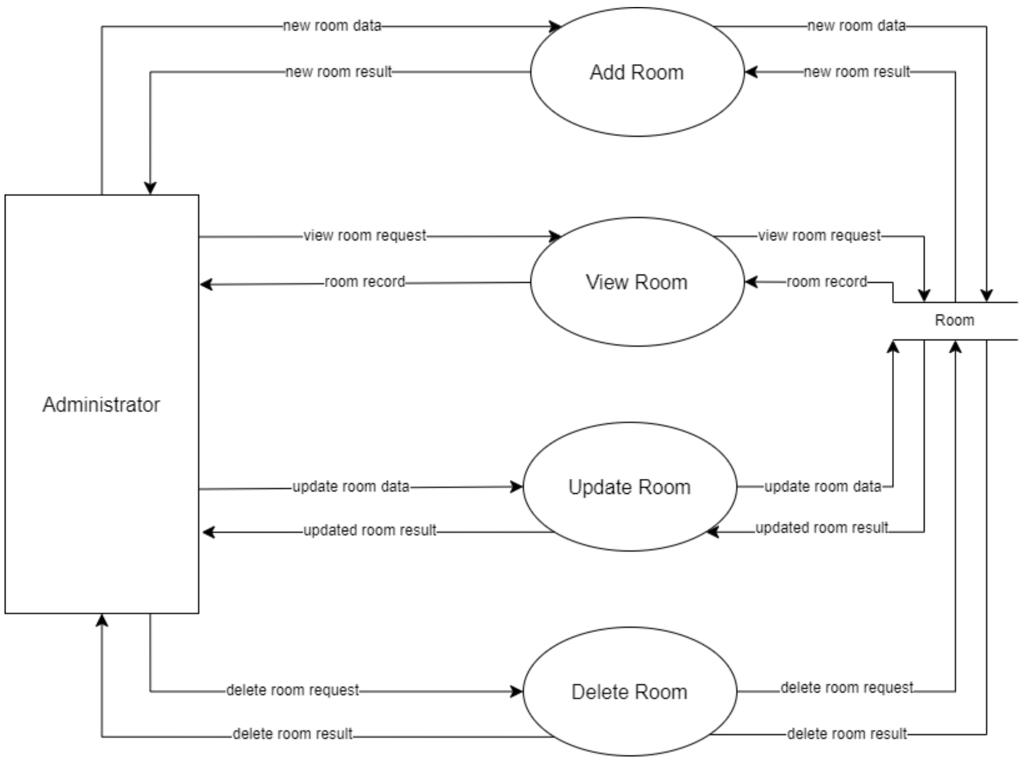


Figure 5.8: Level 2 DFD (Manage Room)

5.3.3.3.4Manage Assessment

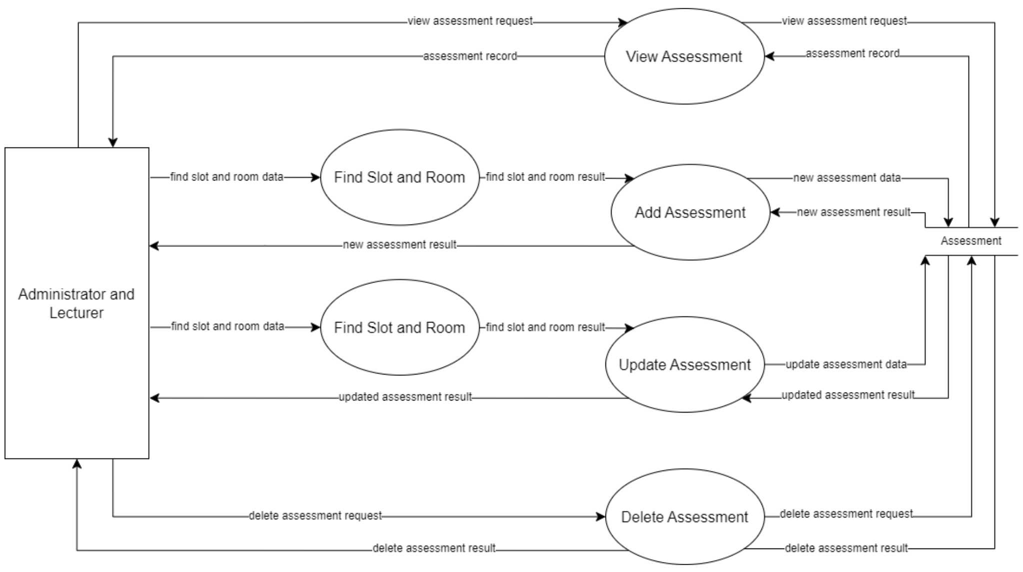


Figure 5.9: Level 2 DFD (Manage Assessment)

5.3.3.3.4Manage Course, Course Timetable, Course Enrolment

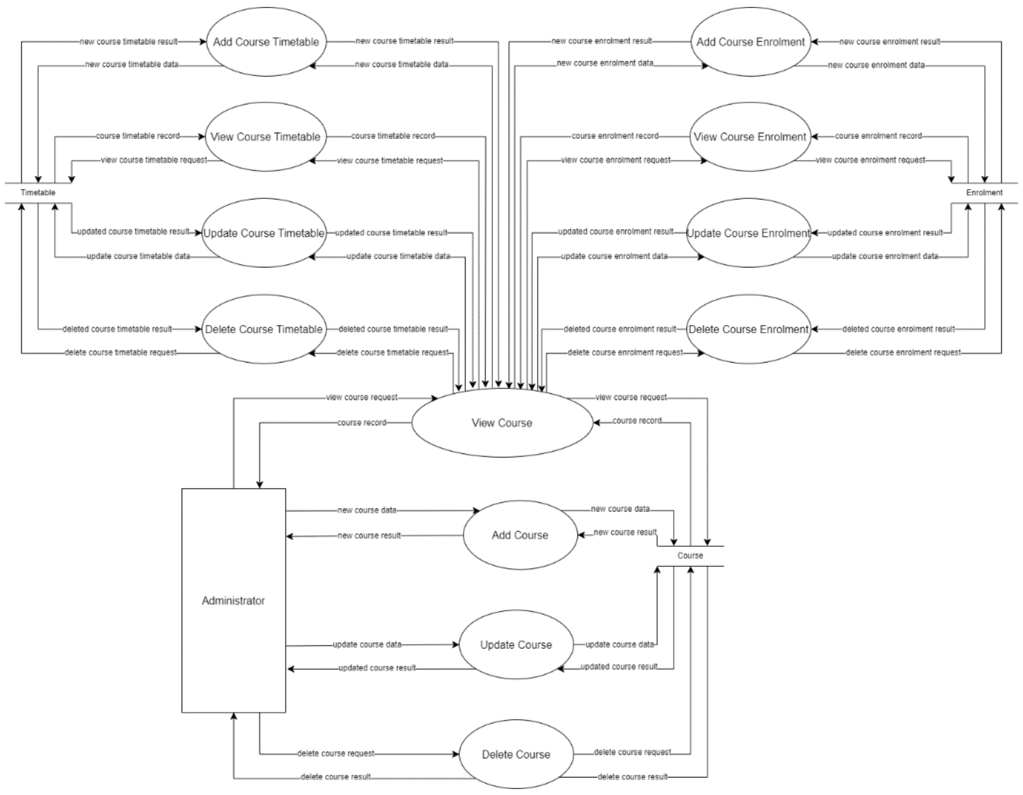

Add Course Timetable
new course timetable result
new course timetable data
new course timetable result
new course timetable data
View Course Timetable
course timetable record
view course timetable request
course timetable record
view course timetable request
Timetable
Update Course Timetable
updated course timetable result
update course timetable data
updated course timetable result
update course timetable data
Delete Course Timetable
deleted course timetable result
delete course timetable request
deleted course timetable result
delete course timetable request
Add Course Enrolment
new course enrolment result
new course enrolment data
new course enrolment result
new course enrolment data
View Course Enrolment
course enrolment record
view course enrolment request
course enrolment record
view course enrolment request
Enrolment
Update Course Enrolment
updated course enrolment result
update course enrolment data
updated course enrolment result
update course enrolment data
Delete Course Enrolment
deleted course enrolment result
delete course enrolment request
deleted course enrolment result
delete course enrolment request
View Course
view course request
course record
view course request
course record
Administrator
Add Course
new course data
new course result
new course data
new course result
Course
Update Course
update course data
updated course result
update course data
updated course result
Delete Course
delete course request
delete course result
delete course request
delete course result

Figure 5.10: Level 2 DFD (Manage Course, Timetable, Enrolment)

5.3.4 Use Case Diagram and Use Case Description

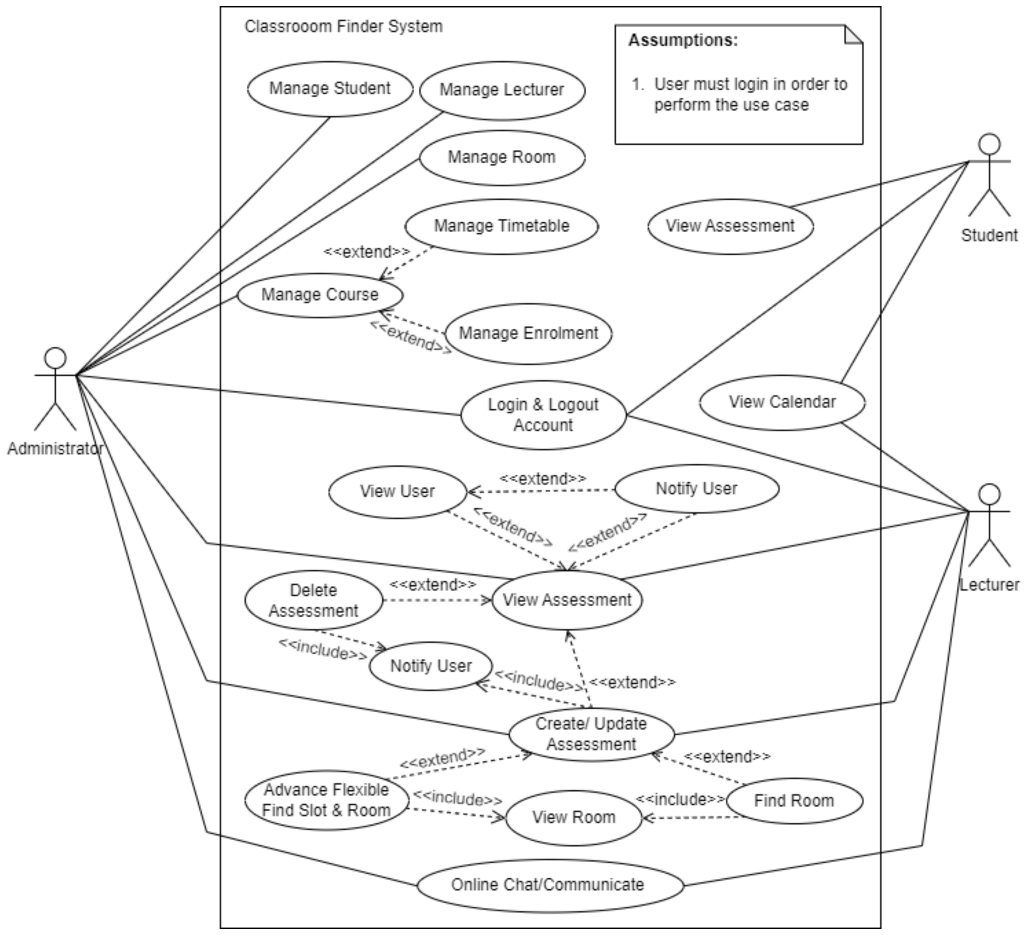


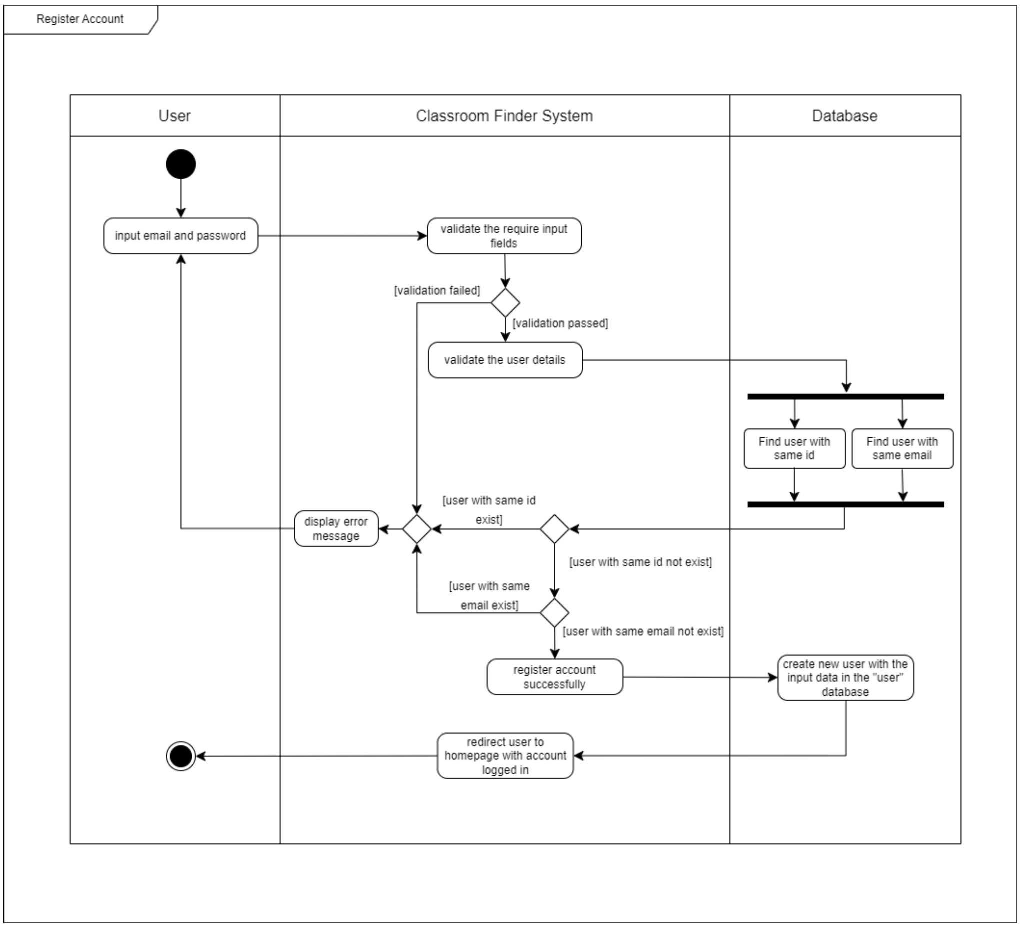
Figure 5.11: Use Case Diagram

In Chapter 4, the Use Case Diagram is introduced as a means to offer a broad perspective on the system's functionalities and user interactions. It delineates the various use cases and actors involved in the system to provide stakeholders with an overarching view of system behaviour. Additionally, the chapter also presents Use Case Descriptions which furnish detailed narratives for each use case. These descriptions outline the sequential steps involved which are including preconditions, post-conditions and alternative flows. By synergizing the Use Case Diagrams and Use Case Descriptions, stakeholders gain a comprehensive understanding of the system's behavioural requirements and user interactions, thus facilitating effective communication and validation of system requirements.

5.3.5 Activity Diagram

Activity Diagrams capture the dynamic behaviour of the Classroom Finder System which help to illustrate the sequence of activities or processes involved in completing a particular task or use case. Activity Diagrams consist of nodes which represent as activities or actions and edges which depict the transitions or flows between activities. By visualizing the workflow and decision points within the system, Activity Diagrams provide insights into the system's process logic and help identify potential bottlenecks or inefficiencies in task execution.

5.3.5.1 Register account



Register Account
User
Classroom Finder System
Database
input email and password
validate the require input fields
[validation failed]
[validation passed]
validate the user details
Find user with same id
Find user with same email
[user with same id exist]
display error message
[user with same id not exist]
[user with same email exist]
[user with same email not exist]
register account successfully
create new user with the input data in the "user" database
redirect user to homepage with account logged in

Figure 5.12: Activity Diagram for Register Account

5.3.5.2 Login account

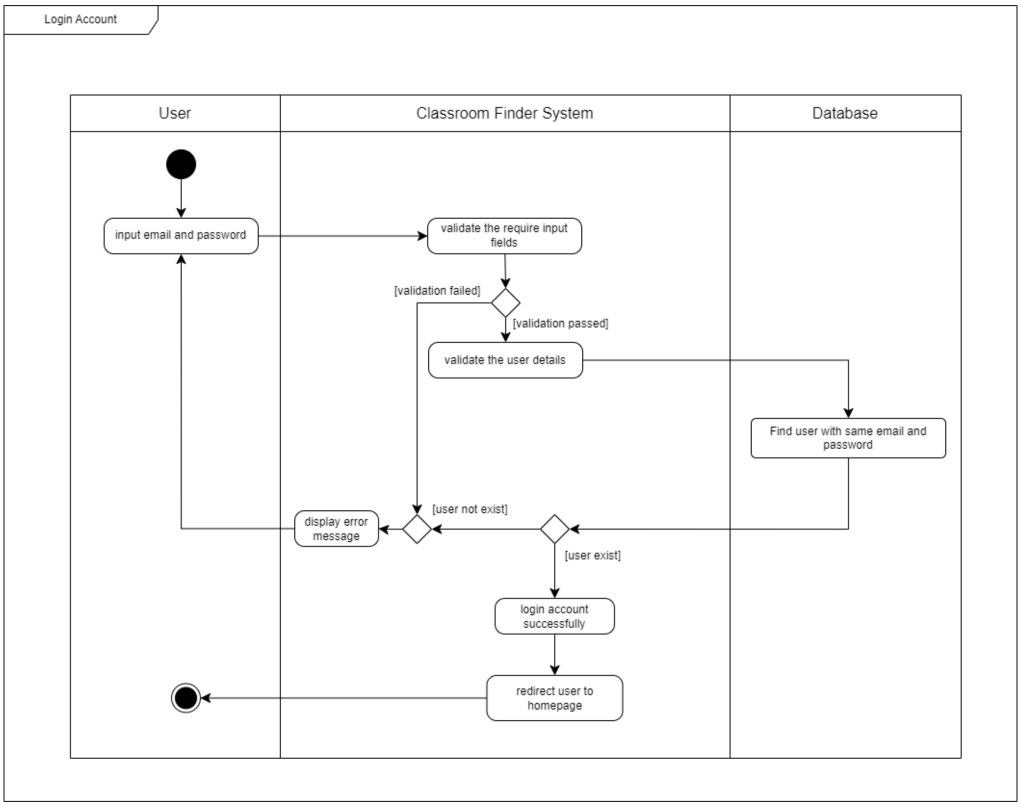


Figure 5.13: Activity Diagram for Login Account

5.3.5.3 Add Lecturer

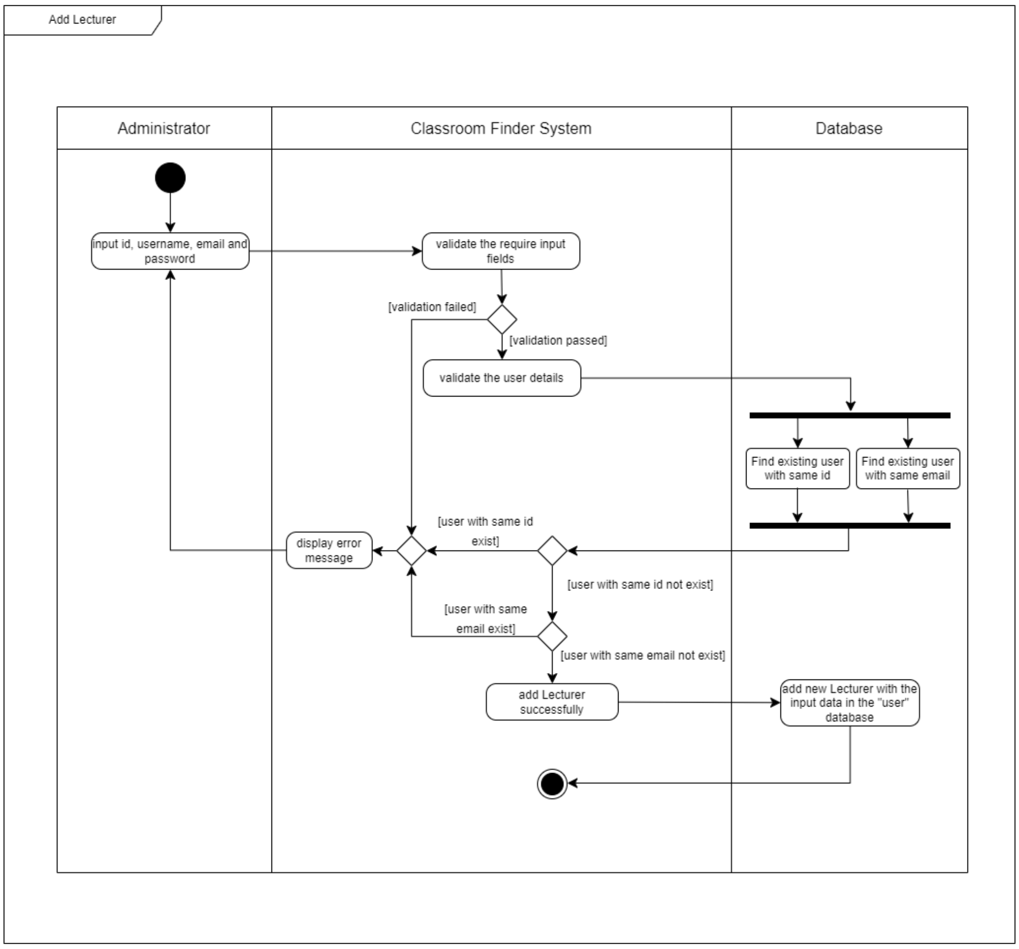

Add Lecturer
Administrator
Classroom Finder System
Database
input id, username, email and password
validate the require input fields
[validation failed]
[validation passed]
validate the user details
Find existing user with same id
Find existing user with same email
[user with same id exist]
display error message
[user with same id not exist]
[user with same email exist]
[user with same email not exist]
add Lecturer successfully
add new Lecturer with the input data in the "user" database

Figure 5.14: Activity Diagram for Add Lecturer

5.3.5.4 View Lecturer

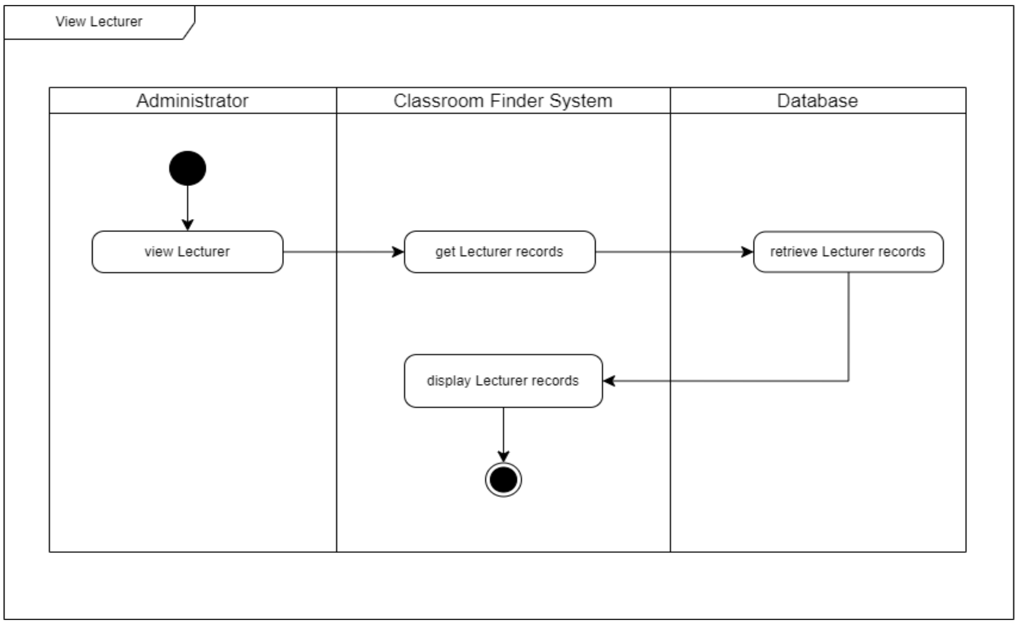


Figure 5.15: Activity Diagram for View Lecturer

5.3.5.5 Update Lecturer

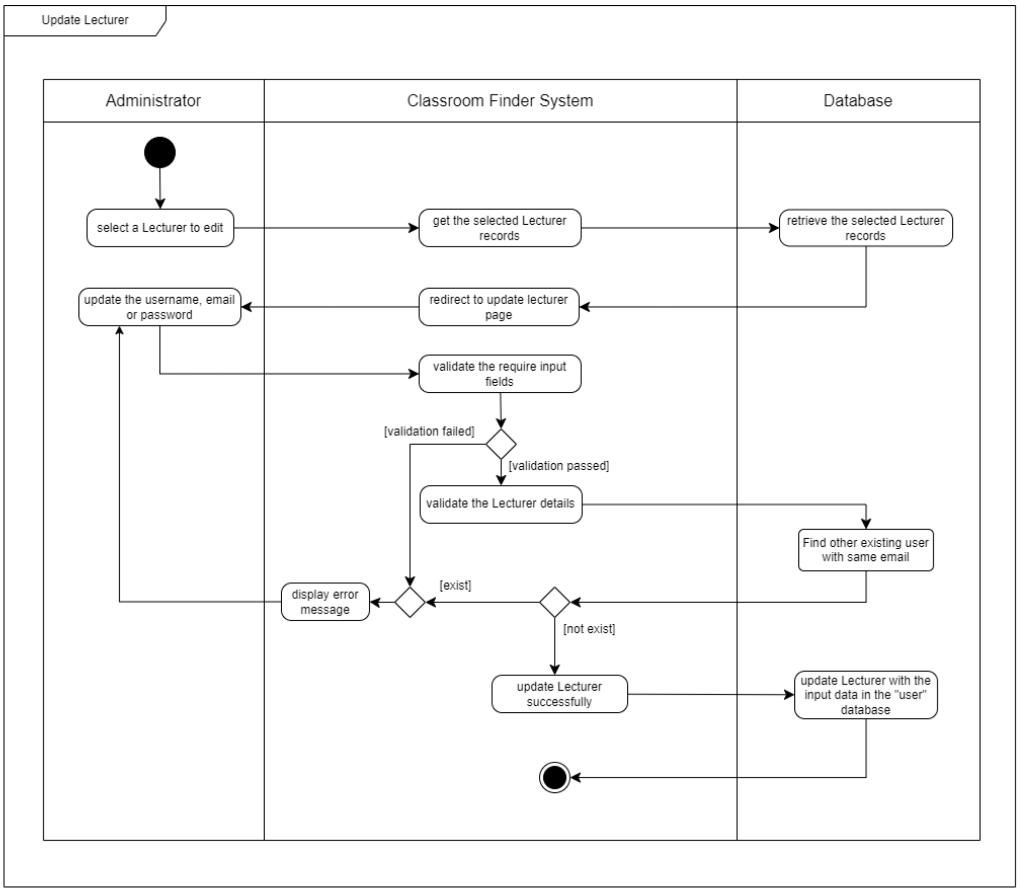

Update Lecturer
Administrator
Classroom Finder System
Database
select a Lecturer to edit
get the selected Lecturer records
retrieve the selected Lecturer records
update the username, email or password
redirect to update lecturer page
validate the require input fields
[validation failed]
[validation passed]
validate the Lecturer details
Find other existing user with same email
[exist]
display error message
[not exist]
update Lecturer successfully
update Lecturer with the input data in the "user" database

Figure 5.16: Activity Diagram for Update Lecturer

5.3.5.6 Delete Lecturer

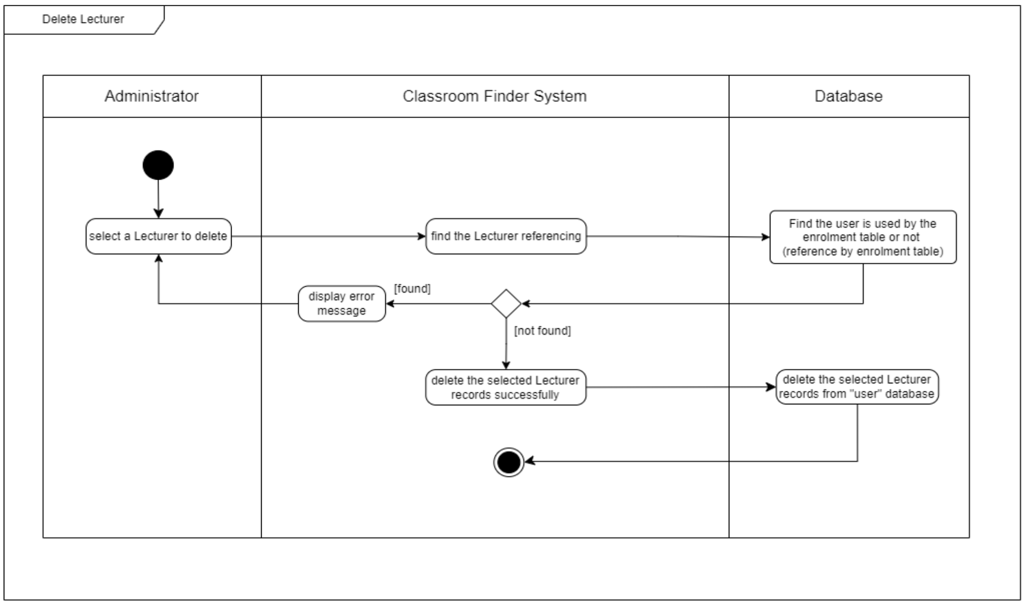

Delete Lecturer
Administrator
Classroom Finder System
Database
select a Lecturer to delete
find the Lecturer referencing
Find the user is used by the enrolment table or not (reference by enrolment table)
display error message
[found]
[not found]
delete the selected Lecturer records successfully
delete the selected Lecturer records from "user" database

Figure 5.17: Activity Diagram for Delete Lecturer

5.3.5.7 Add Student

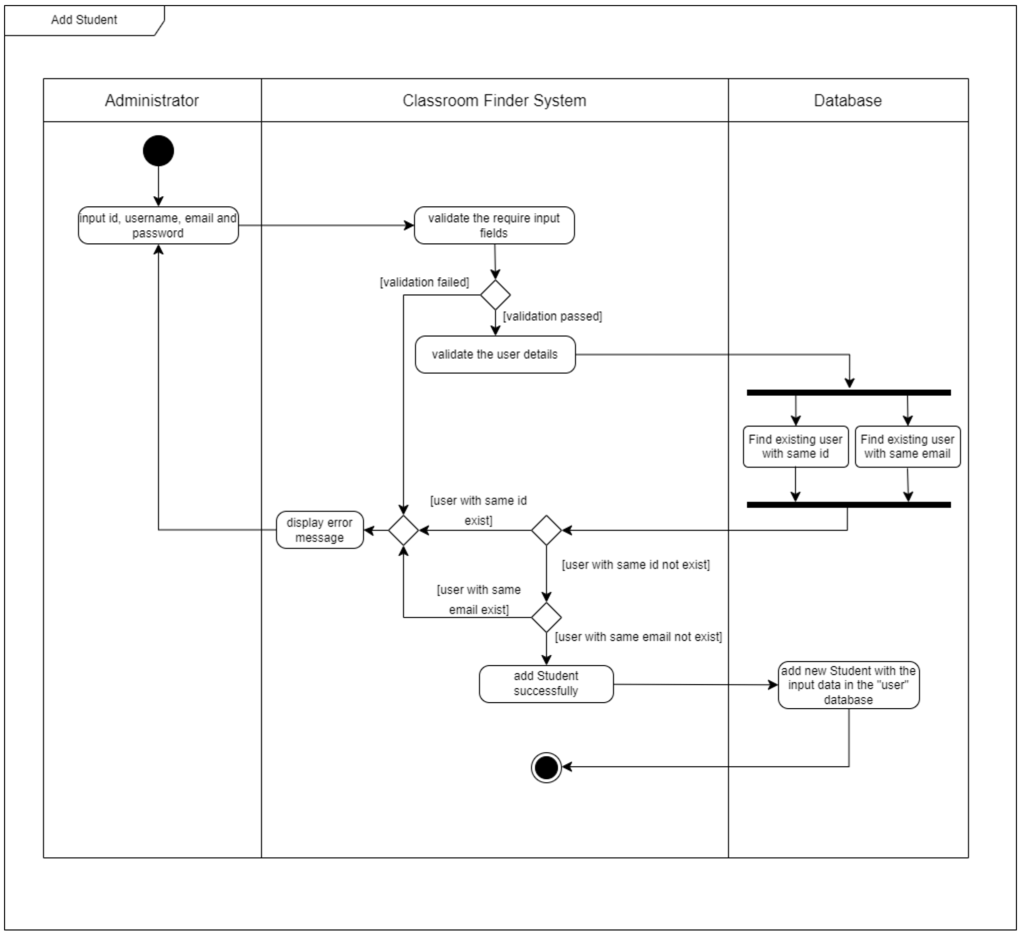

Add Student
Administrator
Classroom Finder System
Database
input id, username, email and password
validate the require input fields
[validation failed]
[validation passed]
validate the user details
Find existing user with same id
Find existing user with same email
[user with same id exist]
display error message
[user with same id not exist]
[user with same email exist]
[user with same email not exist]
add Student successfully
add new Student with the input data in the "user" database

Figure 5.18: Activity Diagram for Add Student

5.3.5.8 View Student

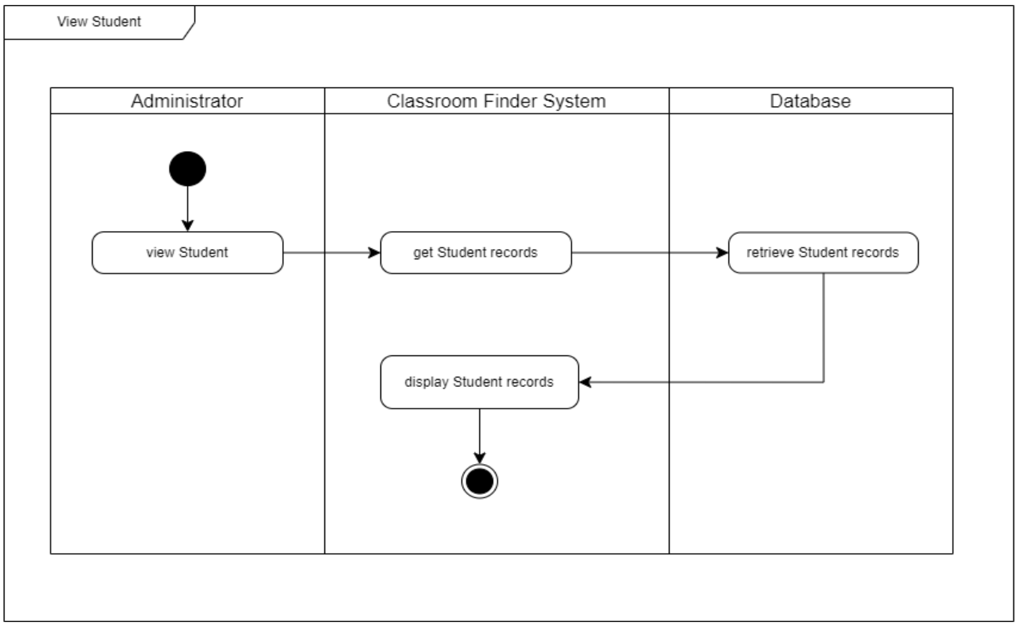


Figure 5.19: Activity Diagram for View Student

5.3.5.9 Update Student

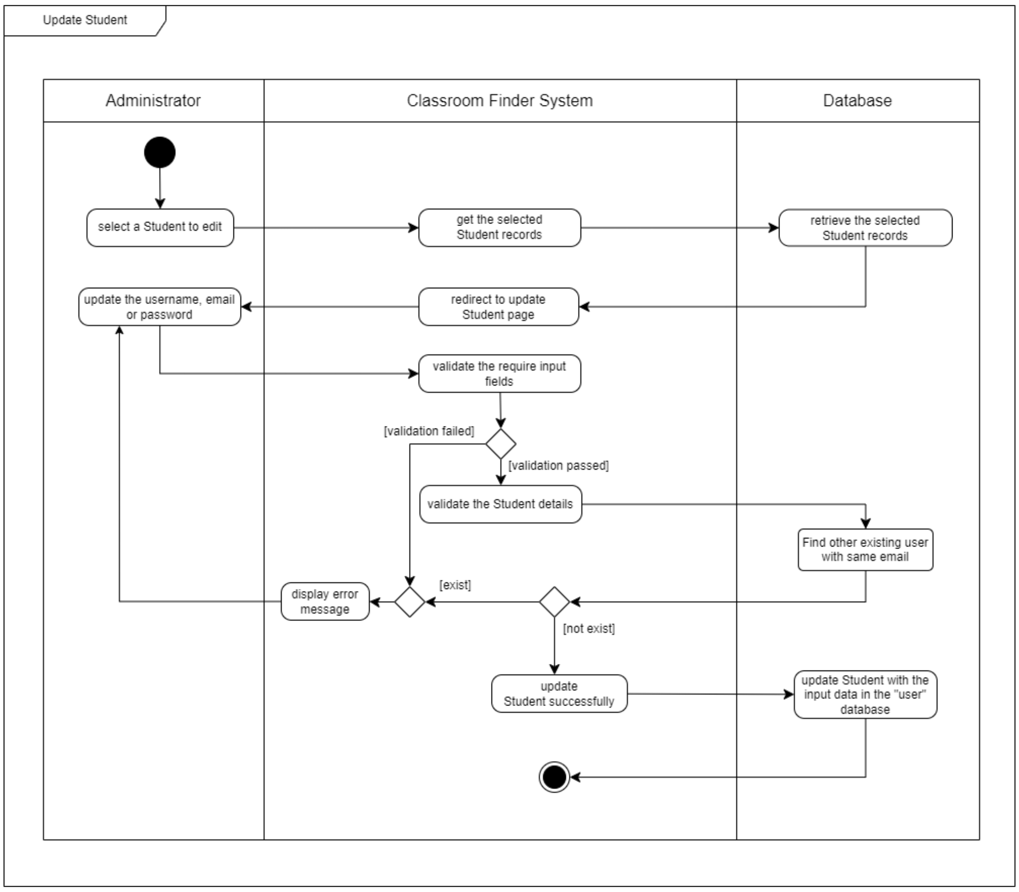

Update Student
Administrator
Classroom Finder System
Database
select a Student to edit
get the selected Student records
retrieve the selected Student records
update the username, email or password
redirect to update Student page
validate the require input fields
[validation failed]
[validation passed]
validate the Student details
Find other existing user with same email
display error message
[exist]
[not exist]
update Student successfully
update Student with the input data in the "user" database

Figure 5.20: Activity Diagram for Update Student

5.3.5.10 Delete Student

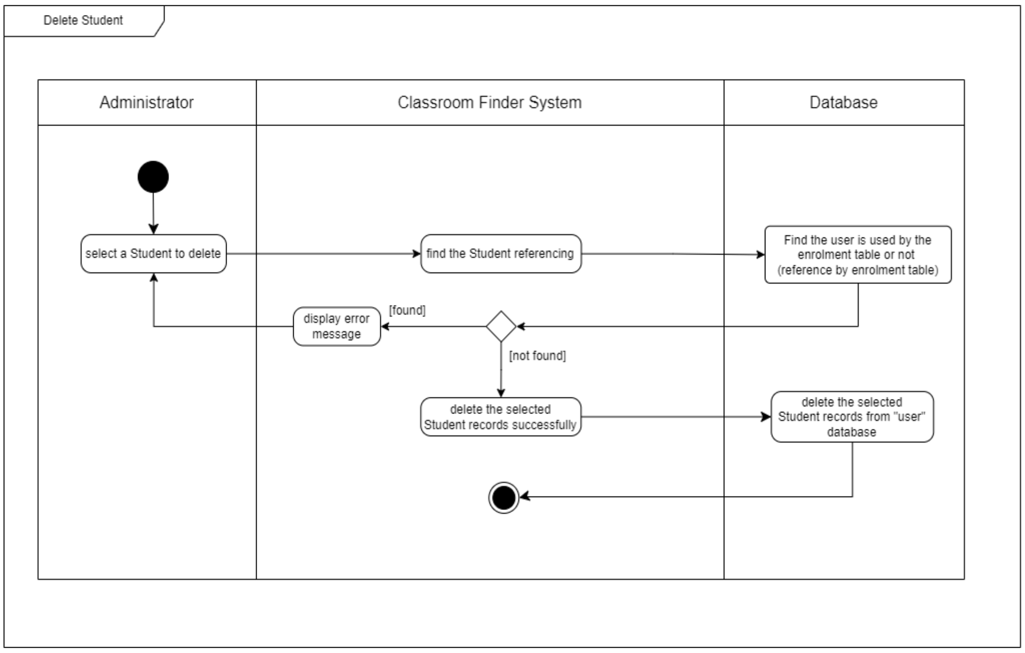

Delete Student
Administrator
Classroom Finder System
Database
select a Student to delete
find the Student referencing
Find the user is used by the enrolment table or not (reference by enrolment table)
display error message
[found]
[not found]
delete the selected Student records successfully
delete the selected Student records from "user" database

Figure 5.21: Activity Diagram for Delete Student

5.3.5.11 Add Room

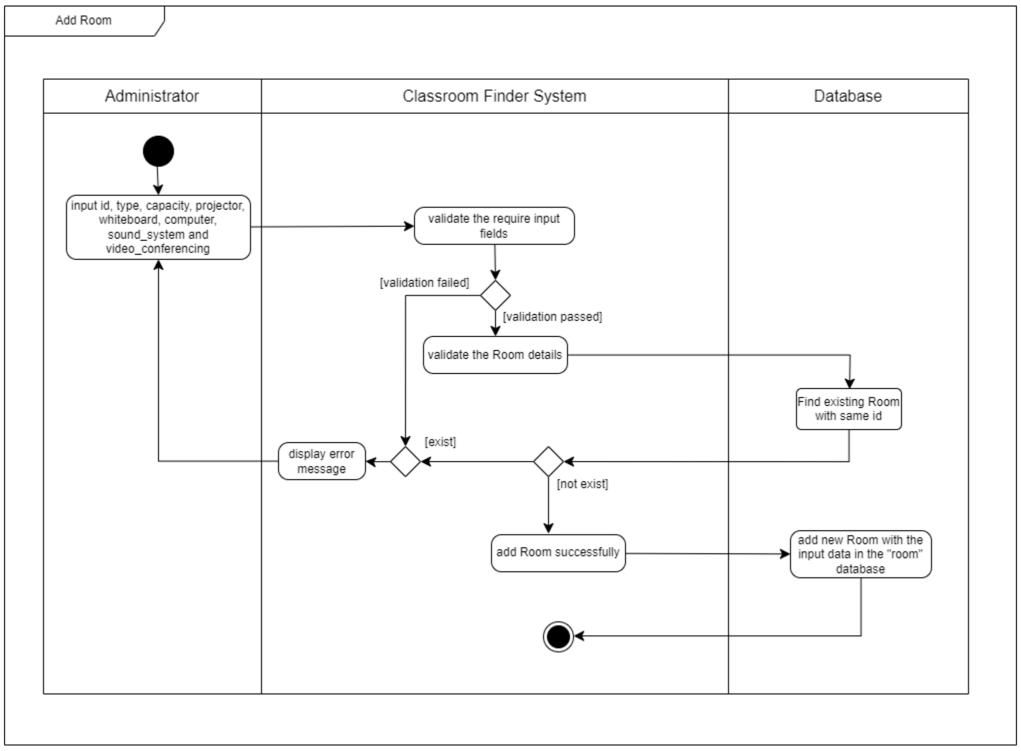

Add Room
Administrator
Classroom Finder System
Database
input id, type, capacity, projector, whiteboard, computer, sound_system and video_conferencing
validate the require input fields
[validation failed]
[validation passed]
validate the Room details
Find existing Room with same id
[exist]
display error message
[not exist]
add Room successfully
add new Room with the input data in the "room" database

Figure 5.22: Activity Diagram for Add Room

5.3.5.12 View Room

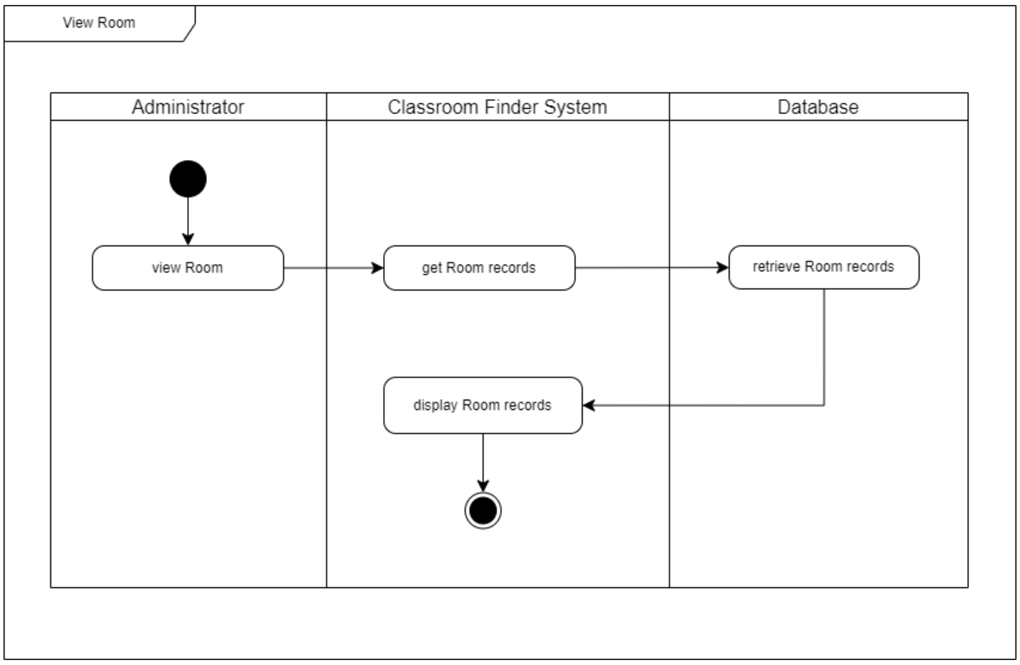


Figure 5.23: Activity Diagram for View Room

5.3.5.13 Update Room

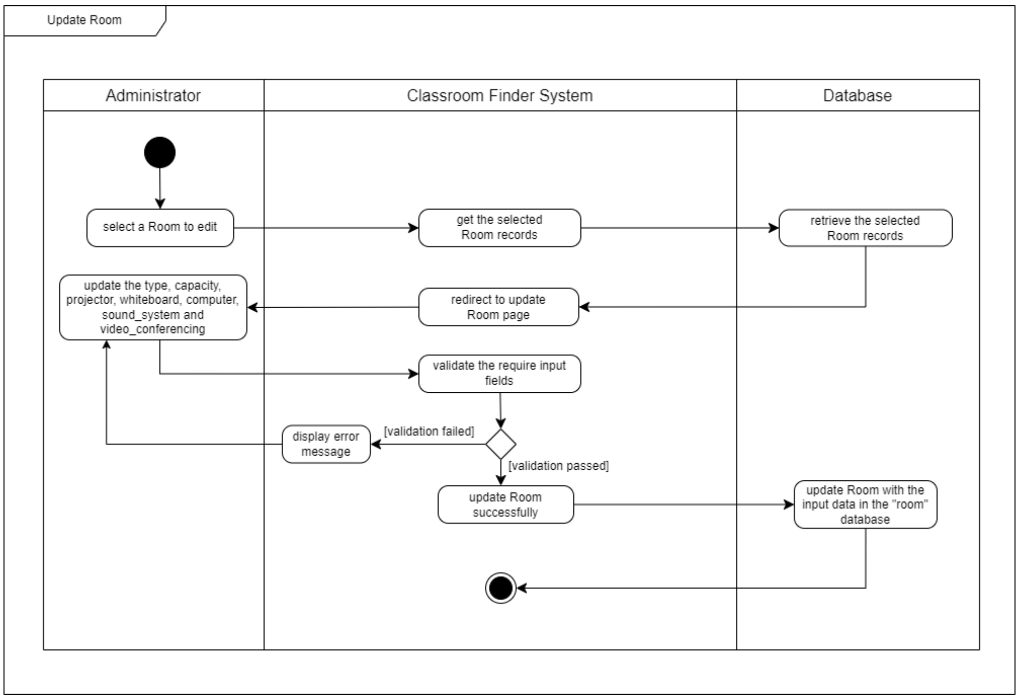

Update Room
Administrator
Classroom Finder System
Database
select a Room to edit
get the selected Room records
retrieve the selected Room records
redirect to update Room page
update the type, capacity, projector, whiteboard, computer, sound_system and video_conferencing
validate the require input fields
[validation failed]
display error message
[validation passed]
update Room successfully
update Room with the input data in the "room" database

Figure 5.24: Activity Diagram for Update Room

5.3.5.14 Delete Room

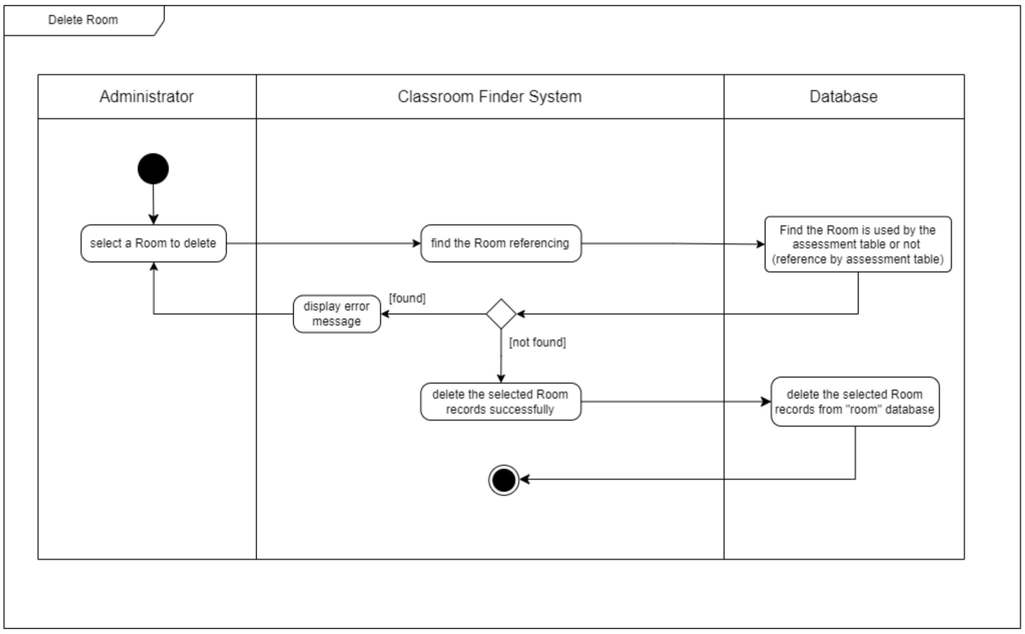

Delete Room
Administrator
Classroom Finder System
Database
select a Room to delete
find the Room referencing
Find the Room is used by the assessment table or not (reference by assessment table)
[found]
display error message
[not found]
delete the selected Room records successfully
delete the selected Room records from "room" database

Figure 5.25: Activity Diagram for Delete Room

5.3.5.15 Add Course

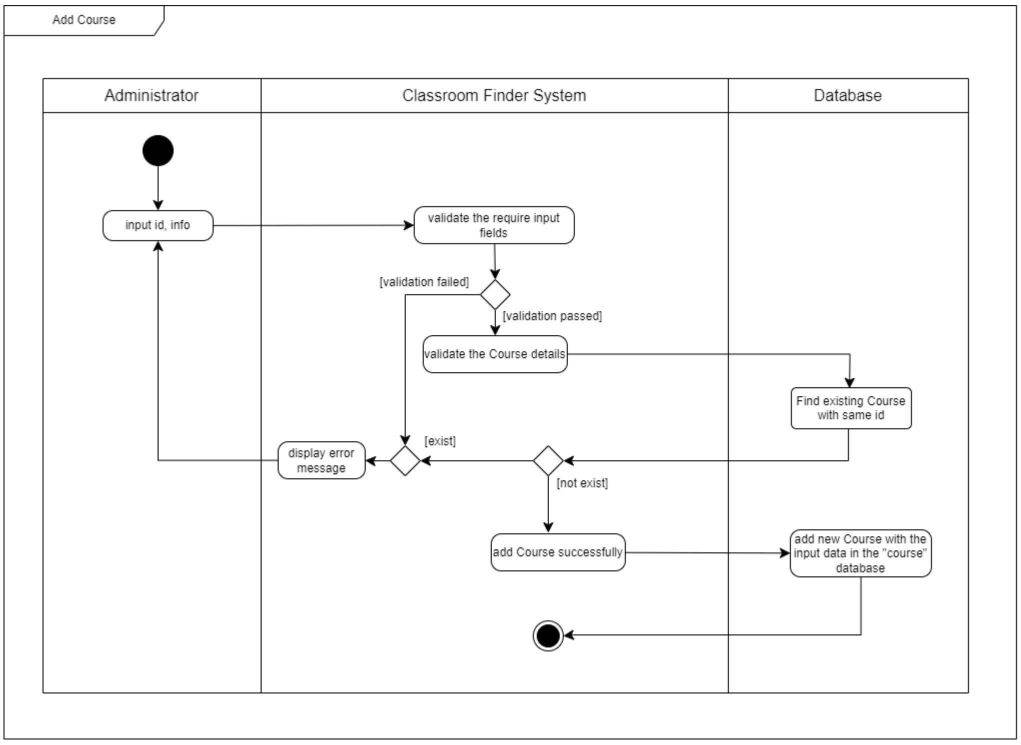


Figure 5.26: Activity Diagram for Add Course

5.3.5.16 View Course

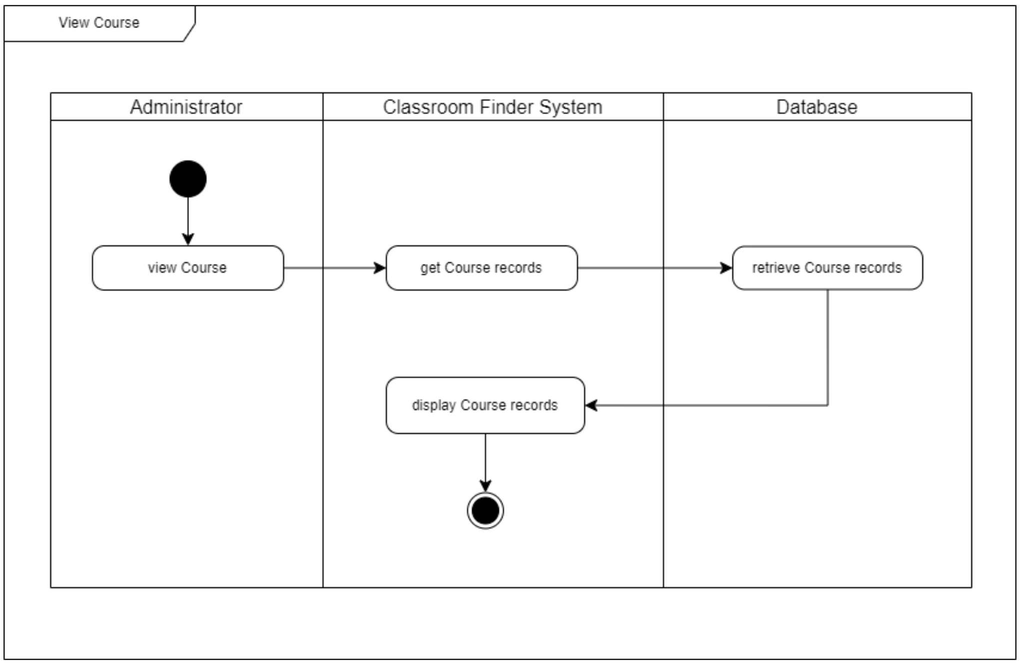


Figure 5.27: Activity Diagram for View Course

5.3.5.17 Update Course

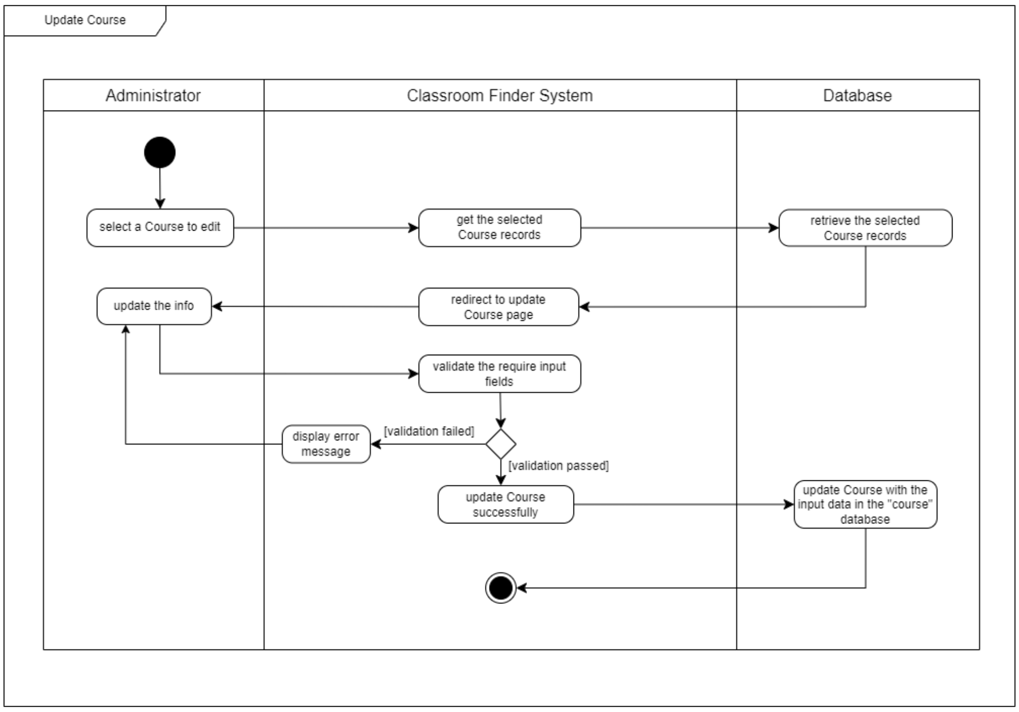

Update Course
Administrator
Classroom Finder System
Database
select a Course to edit
get the selected Course records
retrieve the selected Course records
update the info
redirect to update Course page
validate the require input fields
display error message
[validation failed]
[validation passed]
update Course successfully
update Course with the input data in the "course" database

Figure 5.28: Activity Diagram for Update Course

5.3.5.18 Delete Course

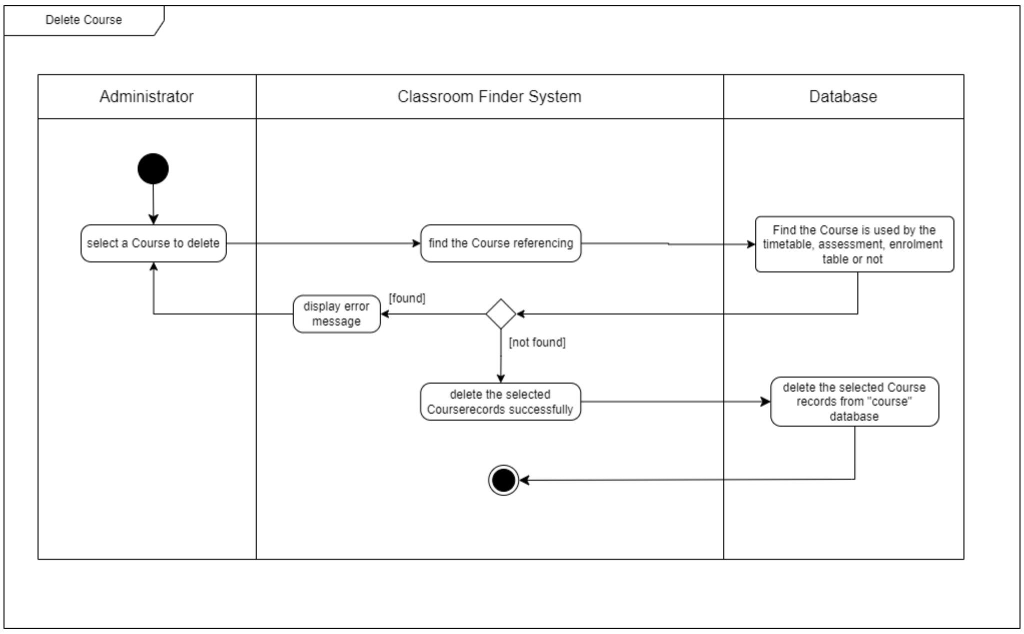

Delete Course
Administrator
Classroom Finder System
Database
select a Course to delete
find the Course referencing
Find the Course is used by the timetable, assessment, enrolment table or not
display error message
[found]
[not found]
delete the selected Courserecords successfully
delete the selected Course records from "course" database

Figure 5.29: Activity Diagram for Delete Course

5.3.5.19 Add Course Timetable

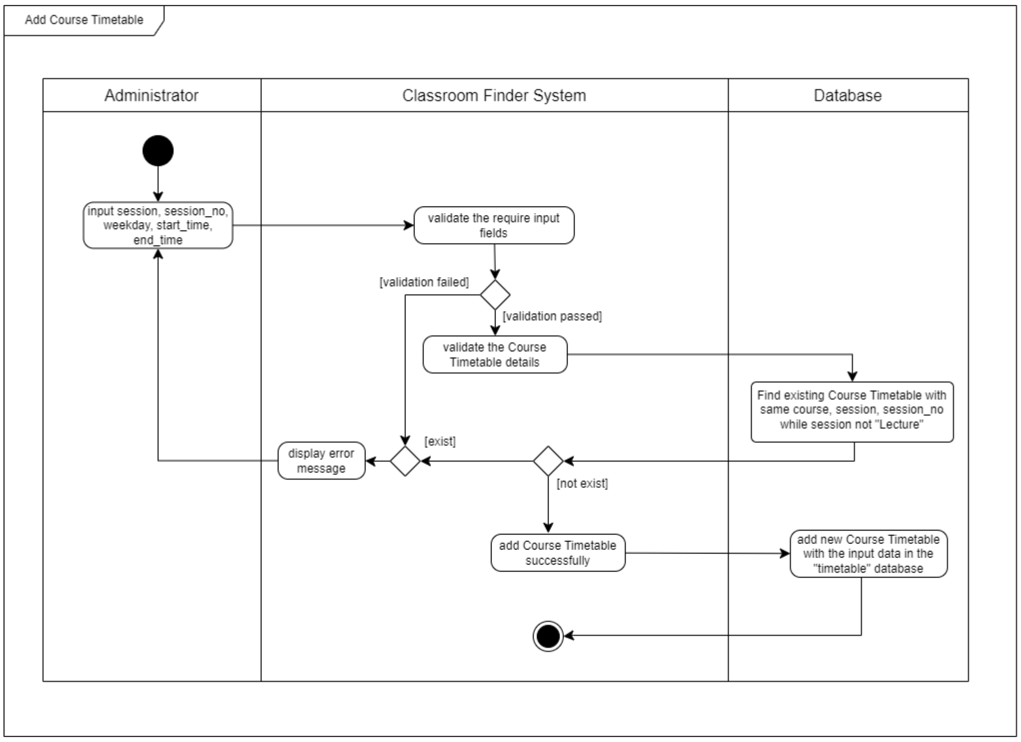

Add Course Timetable
Administrator
Classroom Finder System
Database
input session, session_no, weekday, start_time, end_time
validate the require input fields
[validation failed]
[validation passed]
validate the Course Timetable details
Find existing Course Timetable with same course, session, session_no while session not "Lecture"
[exist]
display error message
[not exist]
add Course Timetable successfully
add new Course Timetable with the input data in the "timetable" database

Figure 5.30: Activity Diagram for Add Course Timetable

5.3.5.20 View Course Timetable

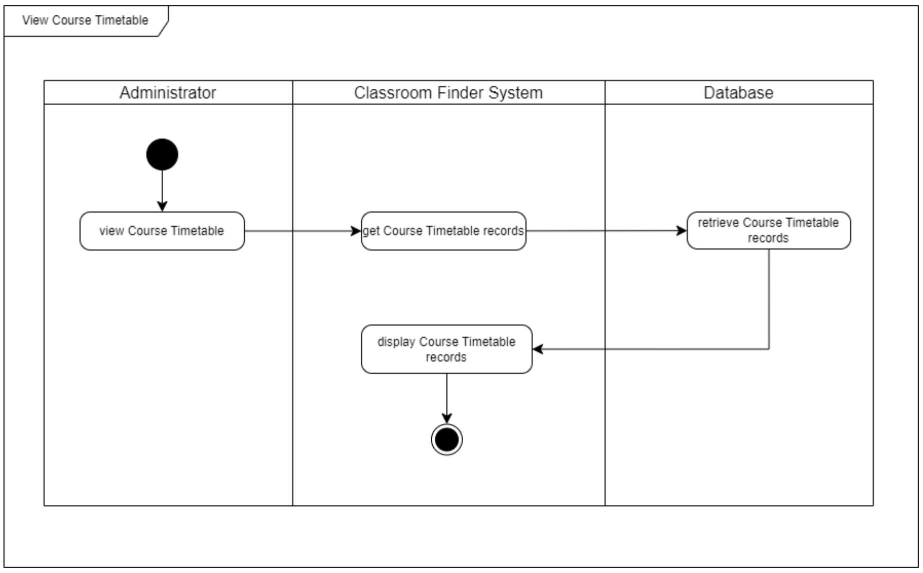

View Course Timetable
Administrator
Classroom Finder System
Database
view Course Timetable
get Course Timetable records
retrieve Course Timetable records
display Course Timetable records

Figure 5.31: Activity Diagram for View Course Timetable

5.3.5.21 Update Course Timetable

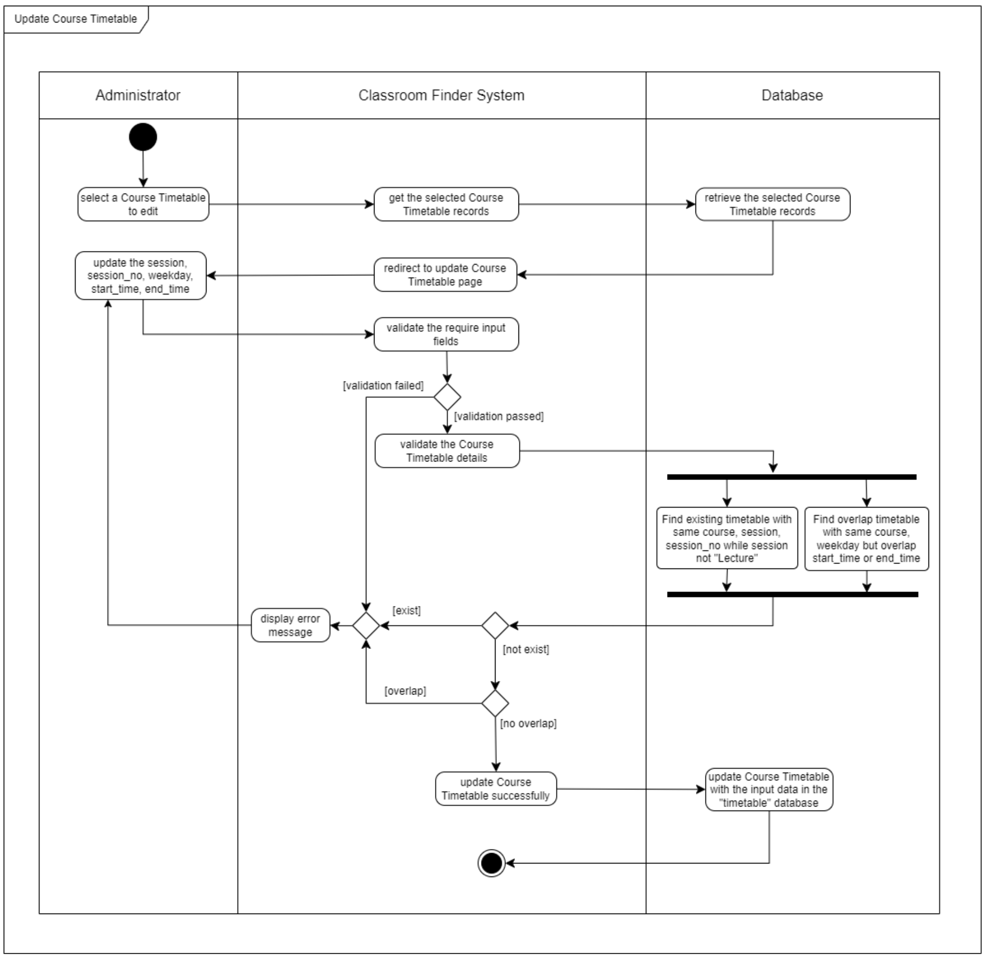

Update Course Timetable
Administrator
Classroom Finder System
Database
select a Course Timetable to edit
get the selected Course Timetable records
retrieve the selected Course Timetable records
update the session, session_no, weekday, start_time, end_time
redirect to update Course Timetable page
validate the require input fields
[validation failed]
[validation passed]
validate the Course Timetable details
Find existing timetable with same course, session, session_no while session not "Lecture"
Find overlap timetable with same course, weekday but overlap start_time or end_time
display error message
[exist]
[not exist]
[overlap]
[no overlap]
update Course Timetable successfully
update Course Timetable with the input data in the "timetable" database

Figure 5.32: Activity Diagram for Update Course Timetable

5.3.5.22 Delete Course Timetable

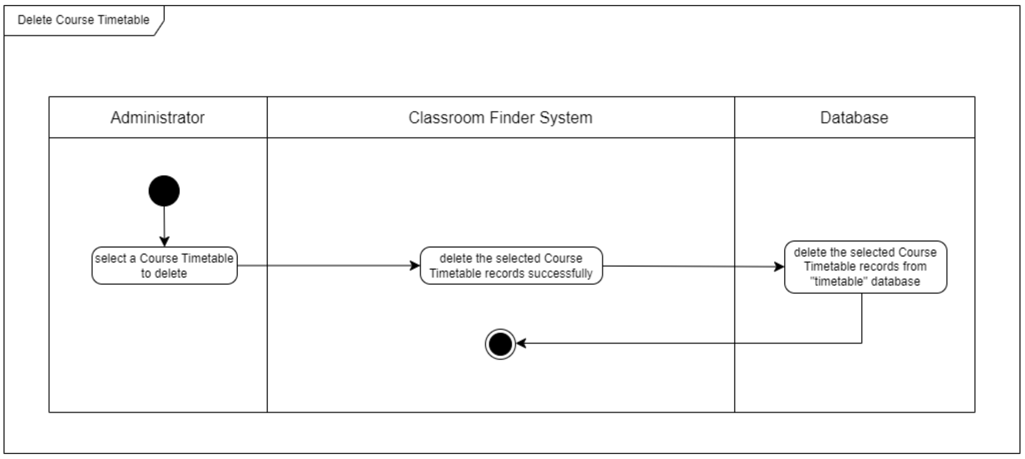

Delete Course Timetable
Administrator
Classroom Finder System
Database
select a Course Timetable to delete
delete the selected Course Timetable records successfully
delete the selected Course Timetable records from "timetable" database

Figure 5.33: Activity Diagram for Delete Course Timetable

5.3.5.23 Add Course Enrolment

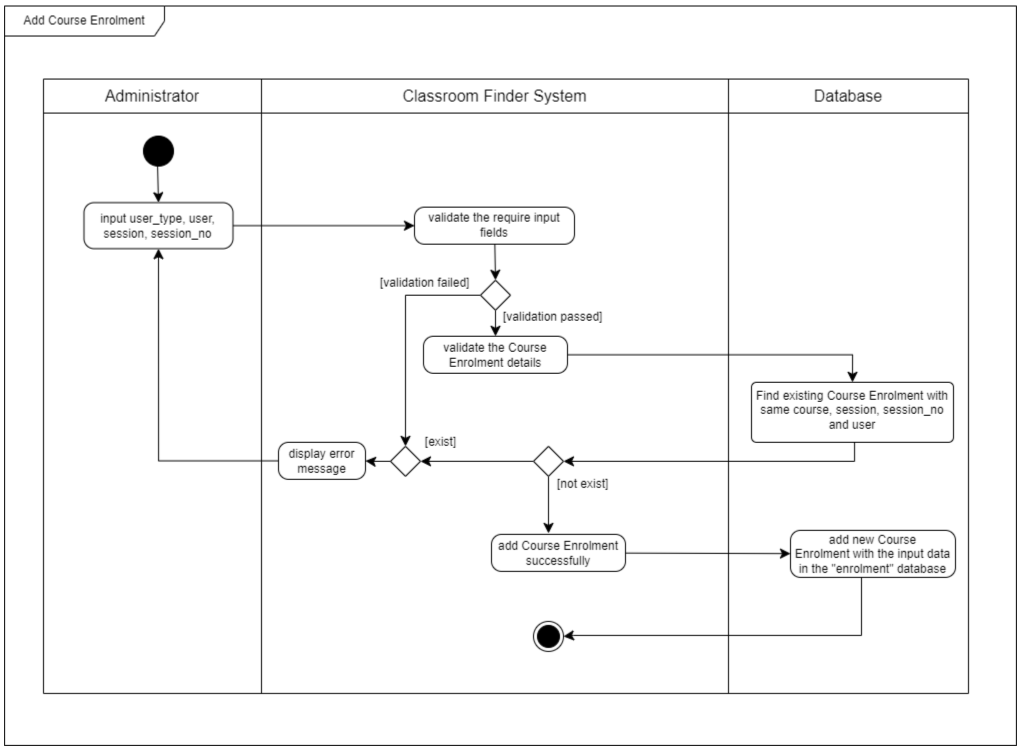

Add Course Enrolment
Administrator
Classroom Finder System
Database
input user_type, user, session, session_no
validate the require input fields
[validation failed]
[validation passed]
validate the Course Enrolment details
Find existing Course Enrolment with same course, session, session_no and user
[exist]
display error message
[not exist]
add Course Enrolment successfully
add new Course Enrolment with the input data in the "enrolment" database

Figure 5.34: Activity Diagram for Add Course Enrolment

5.3.5.24 View Course Enrolment

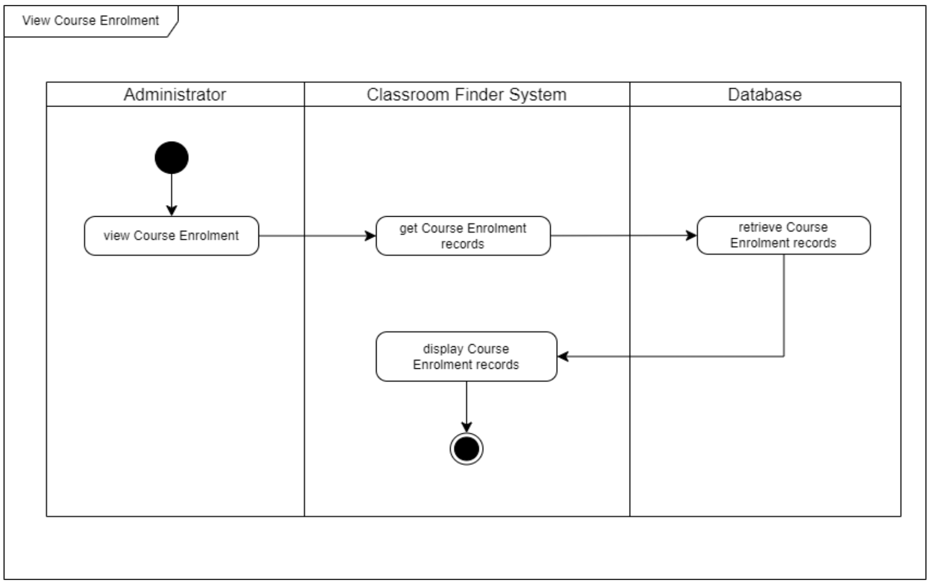

View Course Enrolment
Administrator
Classroom Finder System
Database
view Course Enrolment
get Course Enrolment records
retrieve Course Enrolment records
display Course Enrolment records

Figure 5.35: Activity Diagram for View Course Enrolment

5.3.5.25 Update Course Enrolment

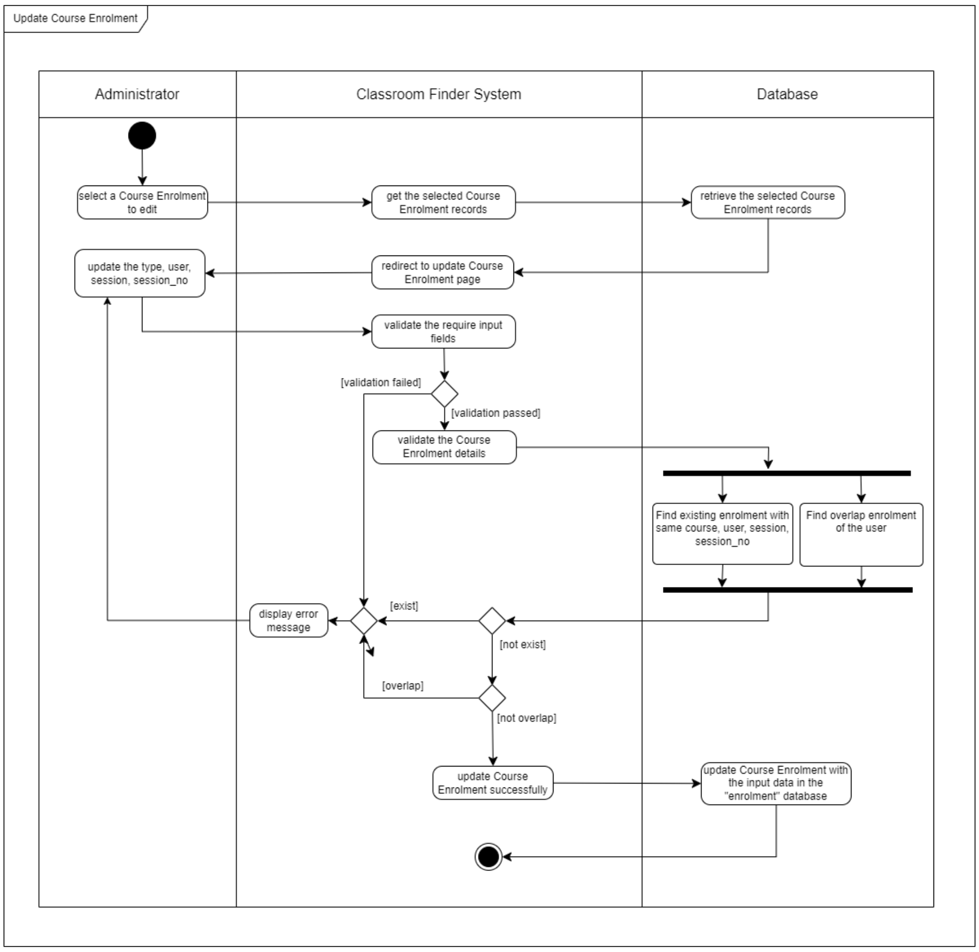

Update Course Enrolment
Administrator
Classroom Finder System
Database
select a Course Enrolment to edit
get the selected Course Enrolment records
retrieve the selected Course Enrolment records
update the type, user, session, session_no
redirect to update Course Enrolment page
validate the require input fields
[validation failed]
[validation passed]
validate the Course Enrolment details
Find existing enrolment with same course, user, session, session_no
Find overlap enrolment of the user
display error message
[exist]
[not exist]
[overlap]
[not overlap]
update Course Enrolment successfully
update Course Enrolment with the input data in the "enrolment" database

Figure 5.36: Activity Diagram for Update Course Enrolment

5.3.5.26 Delete Course Enrolment

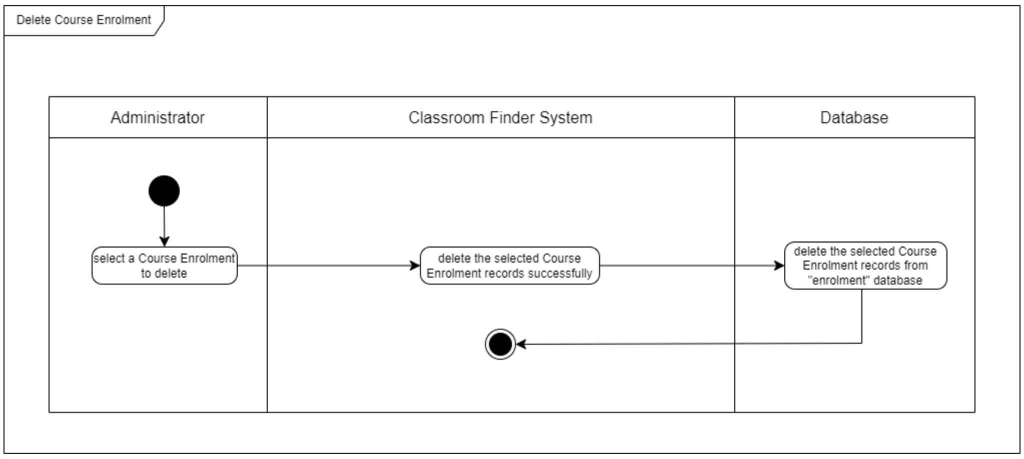

Delete Course Enrolment
Administrator
Classroom Finder System
Database
select a Course Enrolment to delete
delete the selected Course Enrolment records successfully
delete the selected Course Enrolment records from "enrolment" database

Figure 5.37: Activity Diagram for Delete Course Enrolment

5.3.5.27 Add Assessment

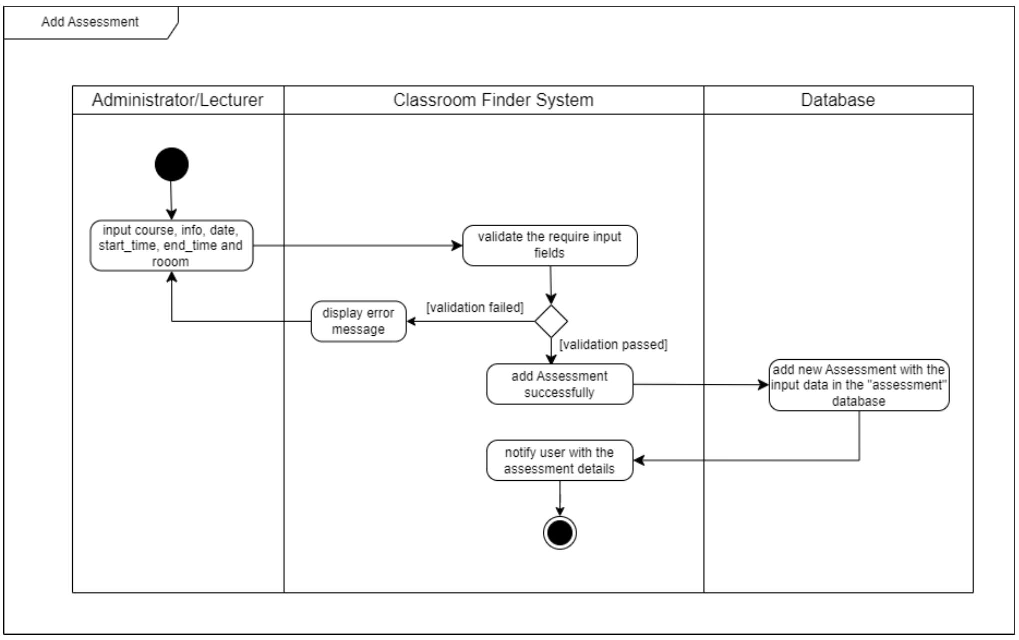

Add Assessment
Administrator/Lecturer
Classroom Finder System
Database
input course, info, date, start_time, end_time and rooom
validate the require input fields
[validation failed]
display error message
[validation passed]
add Assessment successfully
add new Assessment with the input data in the "assessment" database
notify user with the assessment details

Figure 5.38: Activity Diagram for Add Assessment

5.3.5.28 View Assessment

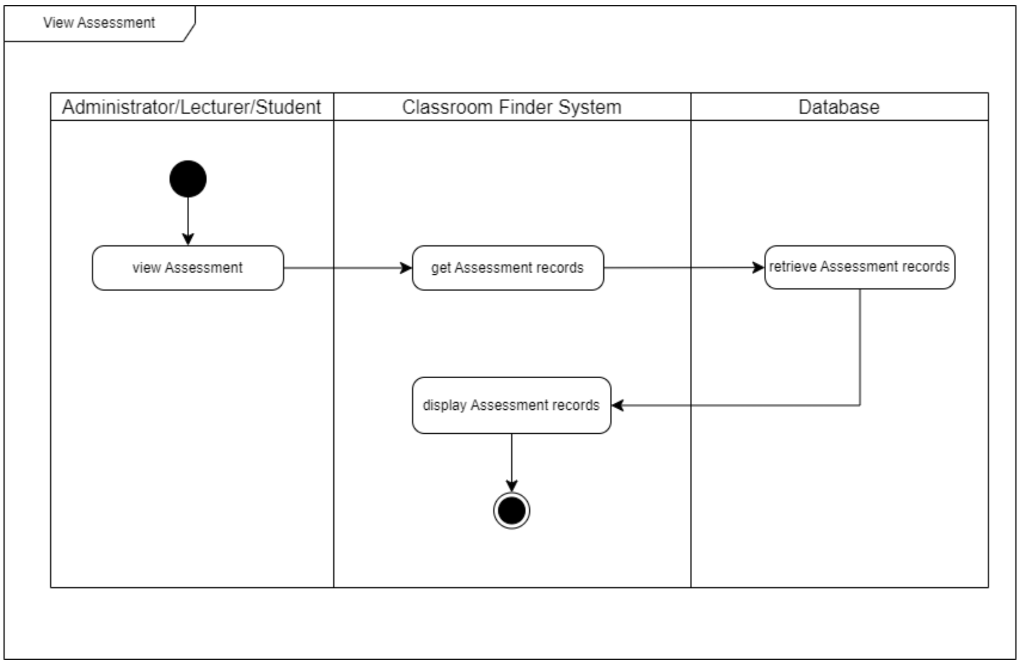

View Assessment
Administrator/Lecturer/Student
Classroom Finder System
Database
view Assessment
get Assessment records
retrieve Assessment records
display Assessment records

Figure 5.39: Activity Diagram for View Assessment

5.3.5.29 Update Assessment

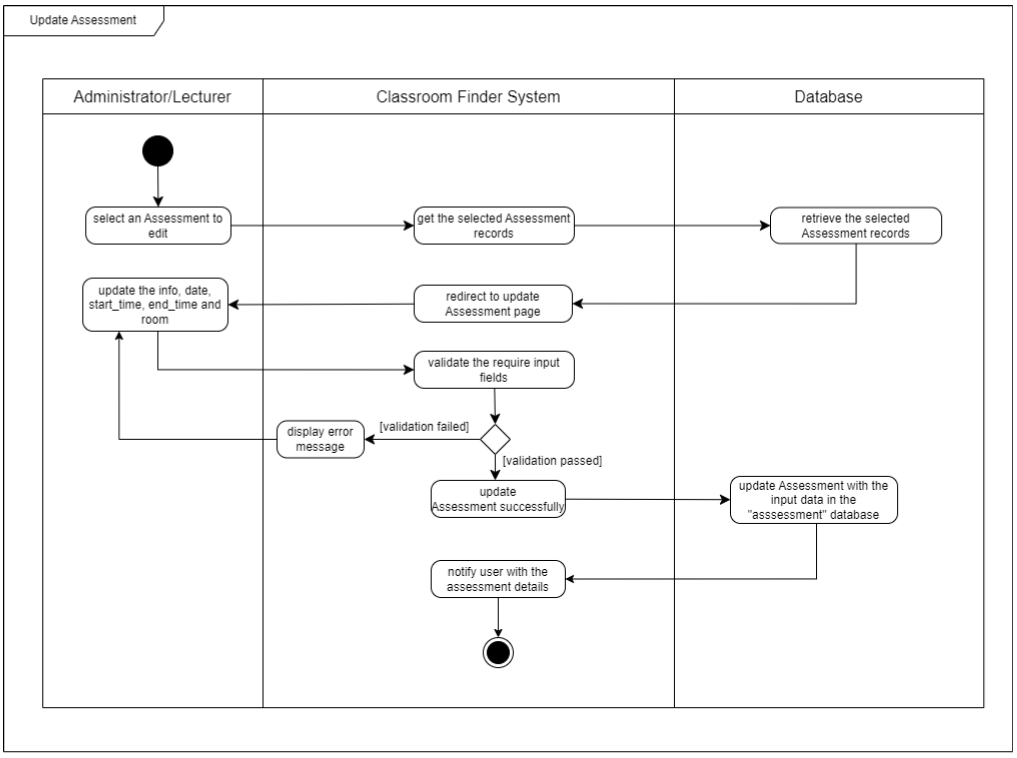

Update Assessment
Administrator/Lecturer
Classroom Finder System
Database
select an Assessment to edit
get the selected Assessment records
retrieve the selected Assessment records
update the info, date, start_time, end_time and room
redirect to update Assessment page
validate the require input fields
display error message
[validation failed]
[validation passed]
update Assessment successfully
update Assessment with the input data in the "asssessment" database
notify user with the assessment details

Figure 5.40: Activity Diagram for Update Assessment

5.3.5.30 Delete Assessment

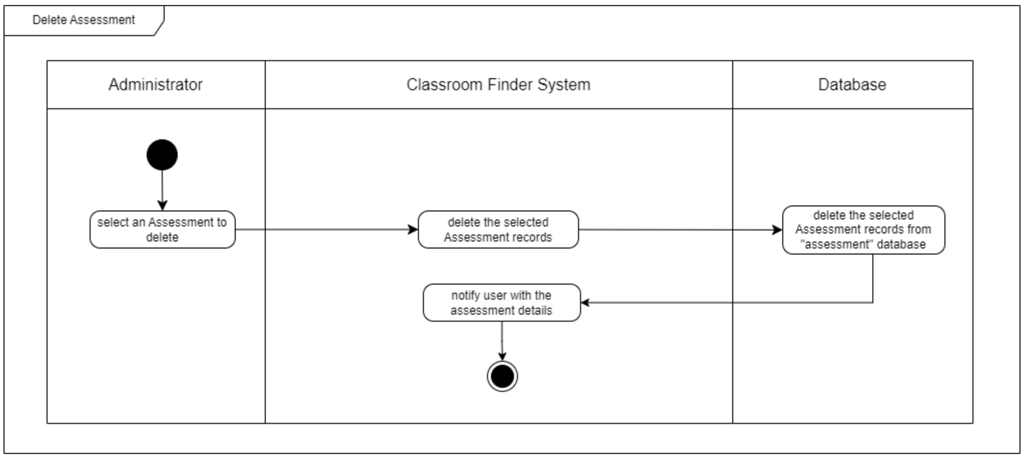

Delete Assessment
Administrator
Classroom Finder System
Database
select an Assessment to delete
delete the selected Assessment records
delete the selected Assessment records from "assessment" database
notify user with the assessment details

Figure 5.41: Activity Diagram for Delete Assessment

5.3.5.31 Online Chat/Communication (Admin)

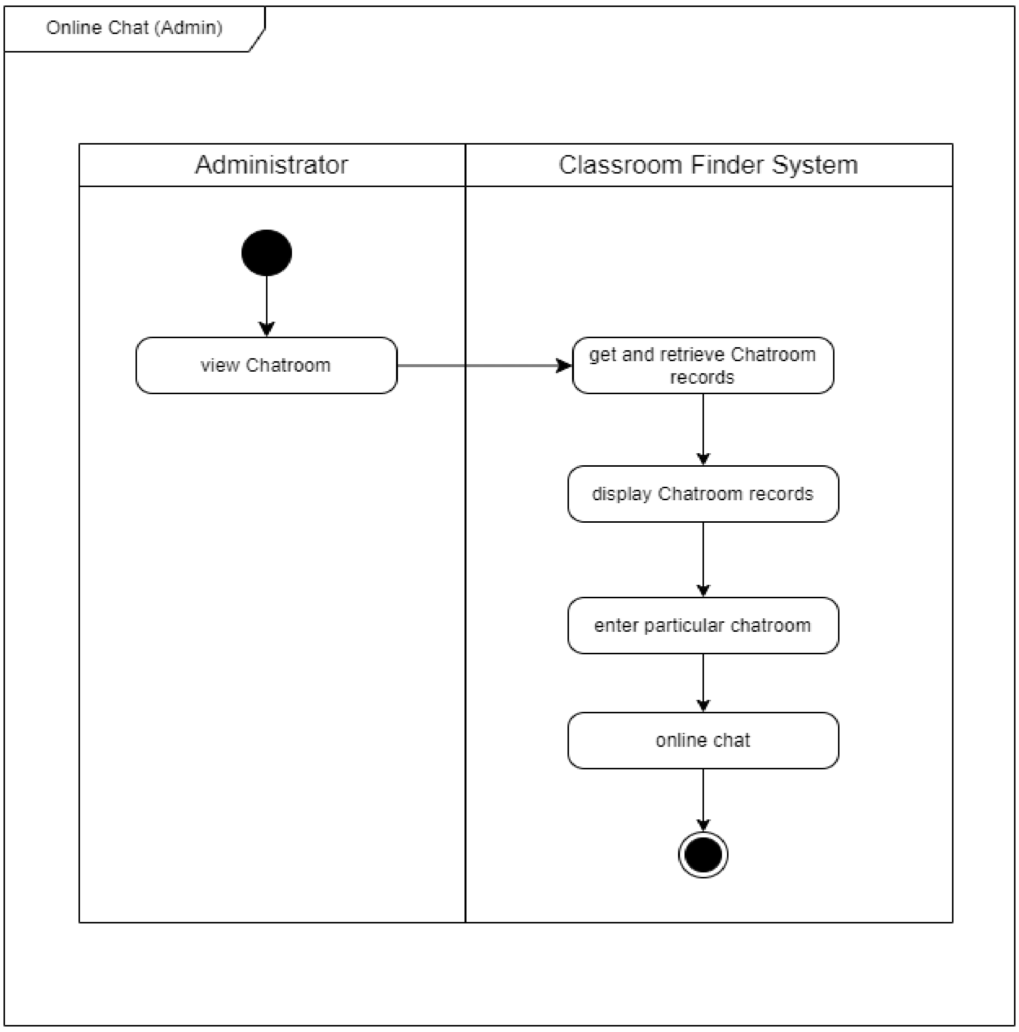


Figure 5.42: Activity Diagram for Online Chat/Communication (Admin)

5.3.5.32 Online Chat/Communication (Lecturer)

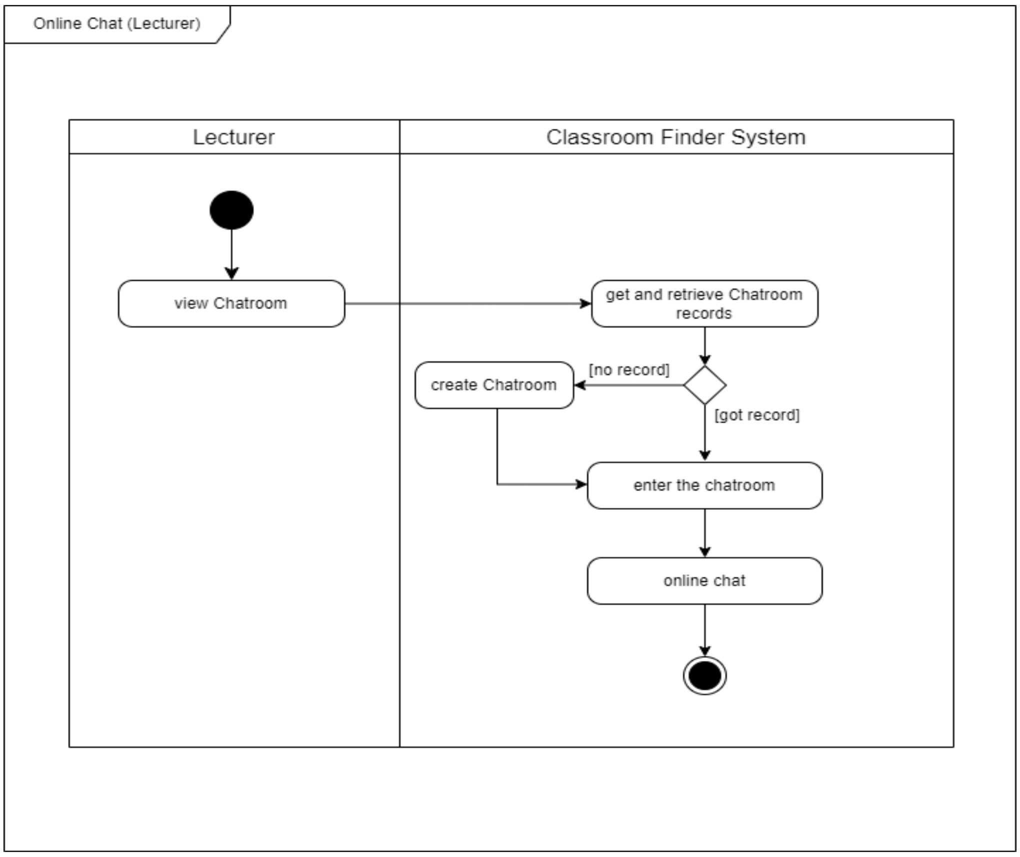


Figure 5.43: Activity Diagram for Online Cha/Communication (Lecturer)

5.3.5.33 View Calendar

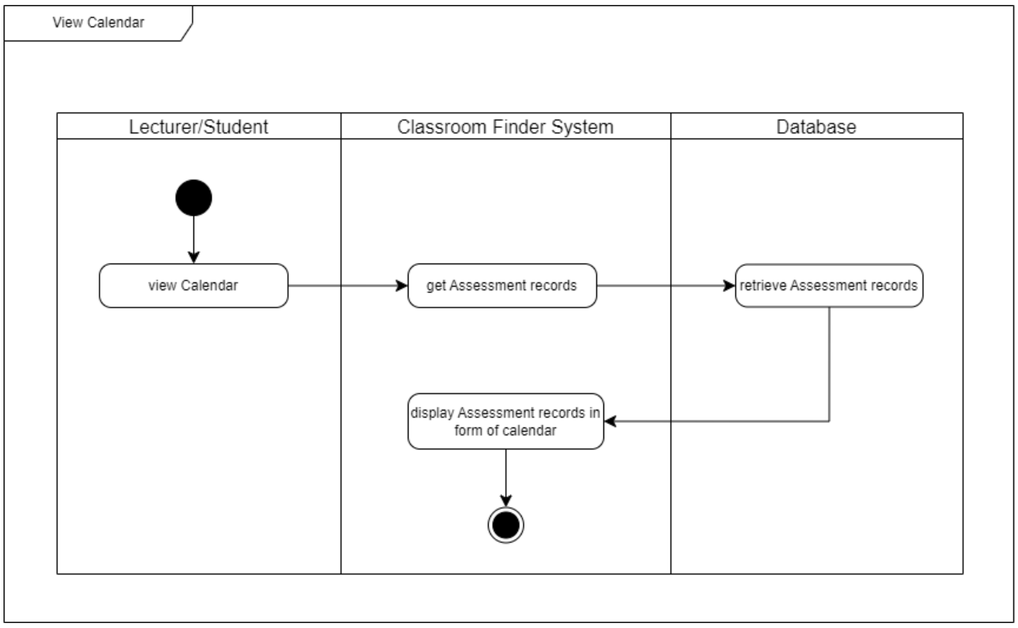
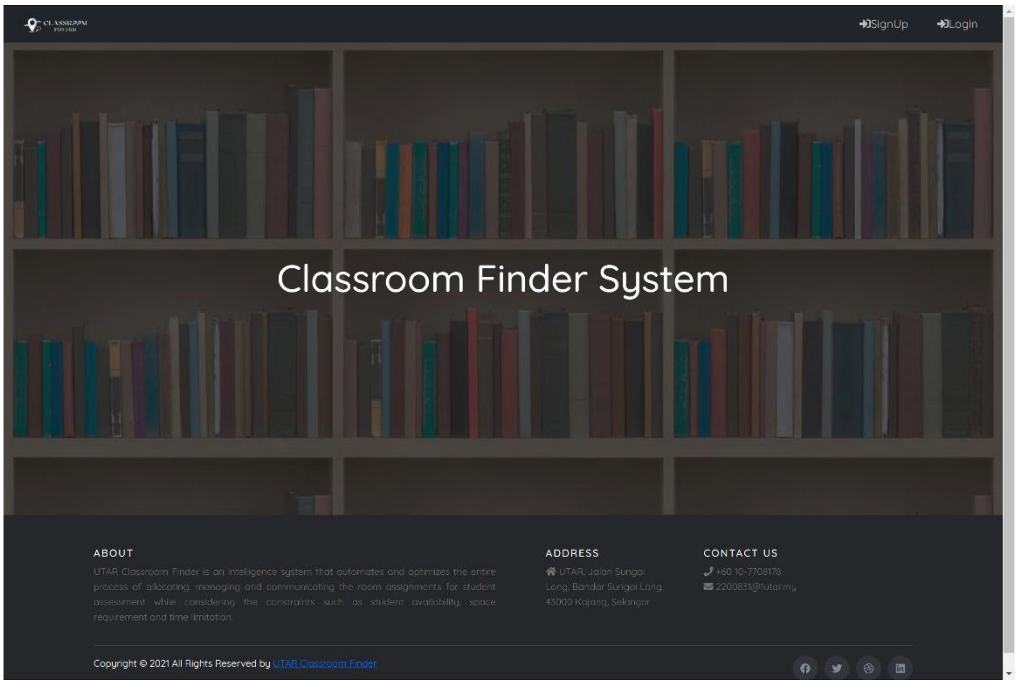


Figure 5.44: Activity Diagram for View Calendar

5.4 User Interface Designs

User Interface Designs offer a graphical representation of the system's user interfaces which showcases the layout, navigation and interaction elements of the Classroom Finder System. By designing intuitive and user-friendly interfaces, stakeholders can enhance user experience and ensure seamless interaction with the system. In this section, the user interface designs are presented.

5.4.1 Home Page



ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Copyright © 2021 All Rights Reserved by UTAR Classroom Finder

Figure 5.45: Home Page

5.4.2 Register Page

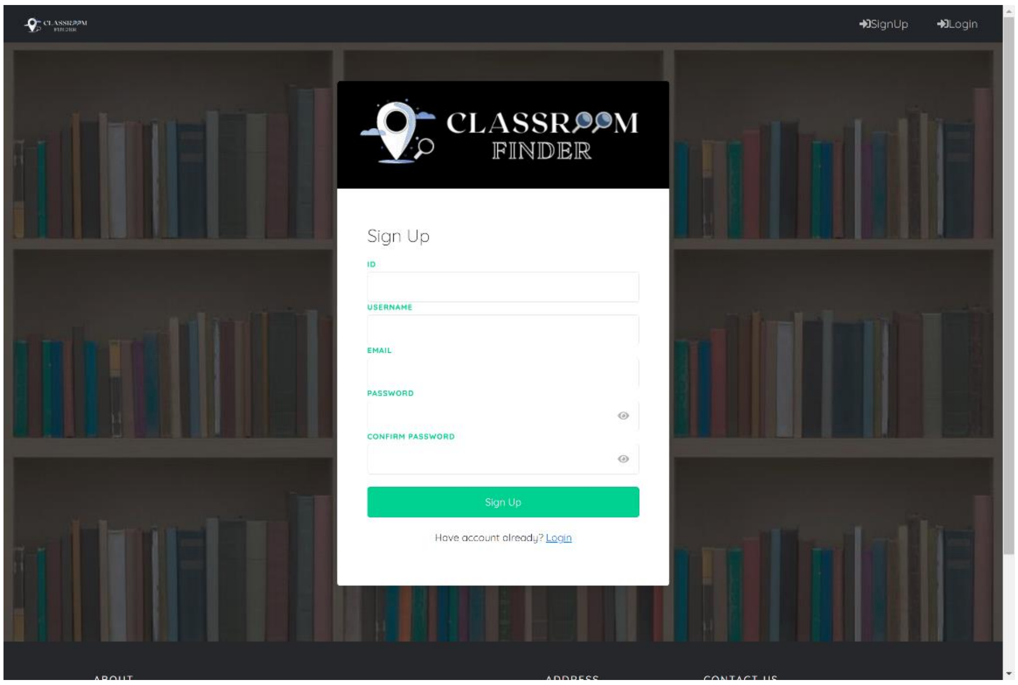
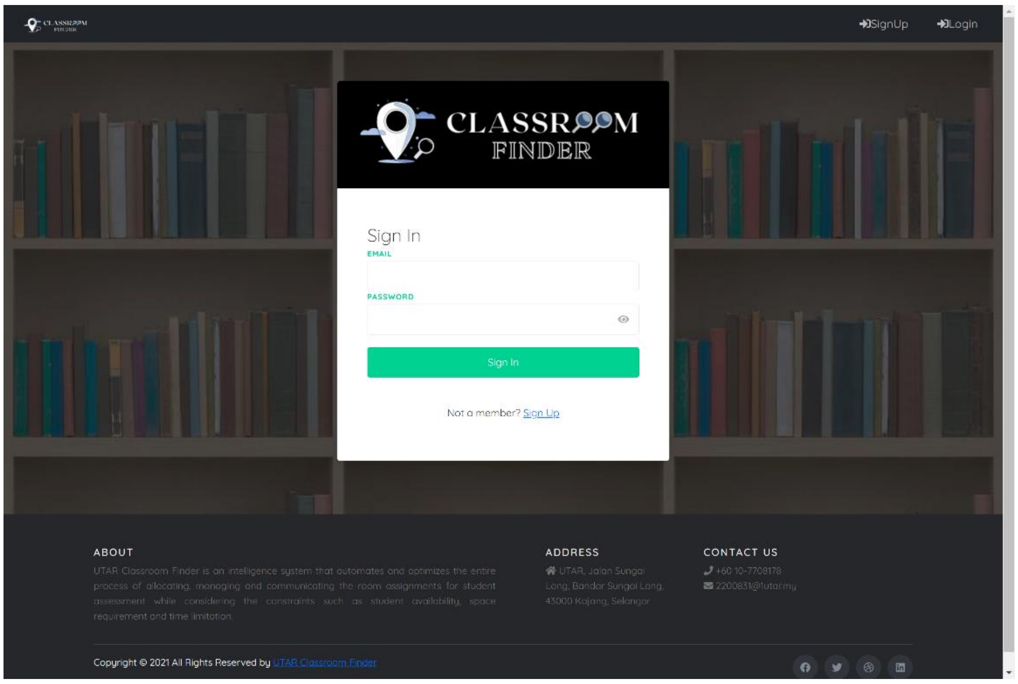


Figure 5.46: Register Page

5.4.3 Login Page



UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

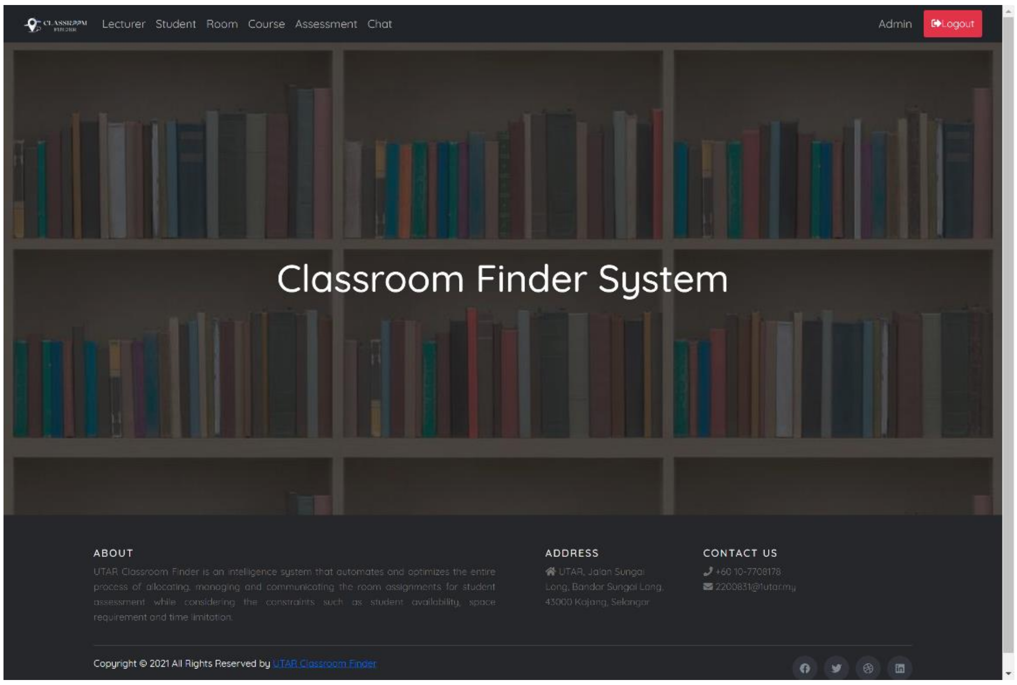
+60 10-7708178

2200831@1utar.my

Copyright © 2021 All Rights Reserved by UTAR Classroom Finder

Figure 5.47: Login Page

5.4.4 Home Page (Administrator)



ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

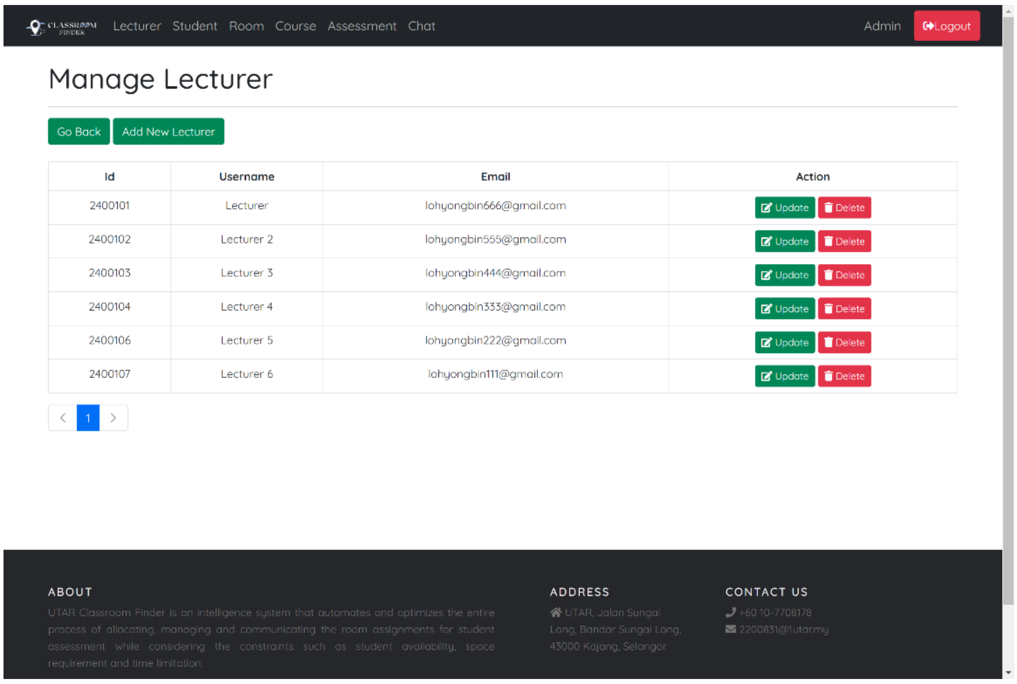
CONTACT US

+60 10-7708178
2200831@1utar.my

Copyright © 2021 All Rights Reserved by UTAR Classroom Finder

Figure 5.48: Home Page (Administrator)

5.4.5 View Lecturer Page (Administrator)



Manage Lecturer

Id	Username	Email	Action
2400101	Lecturer	lohyongbin666@gmail.com	Update Delete
2400102	Lecturer 2	lohyongbin555@gmail.com	Update Delete
2400103	Lecturer 3	lohyongbin444@gmail.com	Update Delete
2400104	Lecturer 4	lohyongbin333@gmail.com	Update Delete
2400106	Lecturer 5	lohyongbin222@gmail.com	Update Delete
2400107	Lecturer 6	lohyongbin111@gmail.com	Update Delete

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

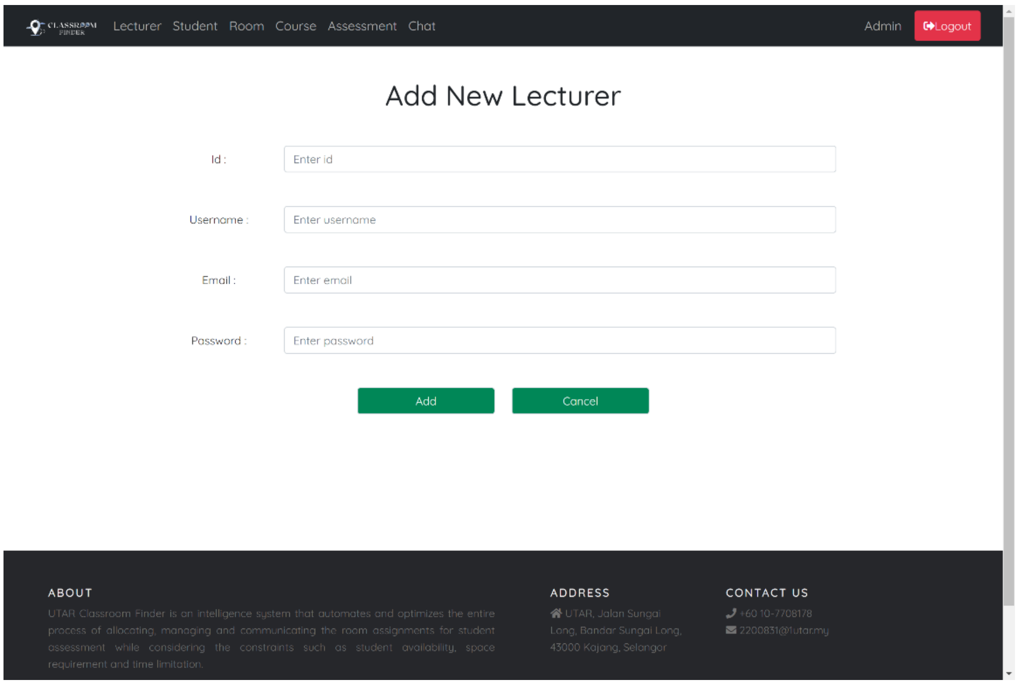
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.49: View Lecturer Page (Administrator)

5.4.6 Add Lecturer Page (Administrator)



Add New Lecturer

Id :	Enter id
Username :	Enter username
Email :	Enter email
Password :	Enter password

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

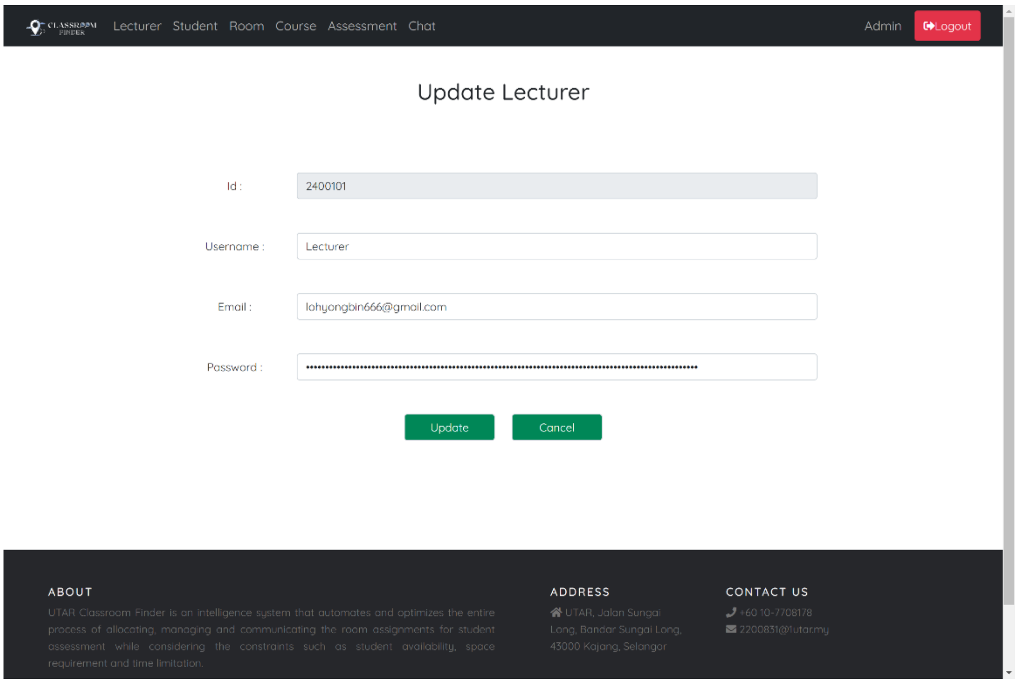
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.50: Add Lecturer Page (Administrator)

5.4.7 Update Lecturer Page (Administrator)



Update Lecturer

Id :	2400101
Username :	Lecturer
Email :	lohyongbin666@gmail.com
Password :	••

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

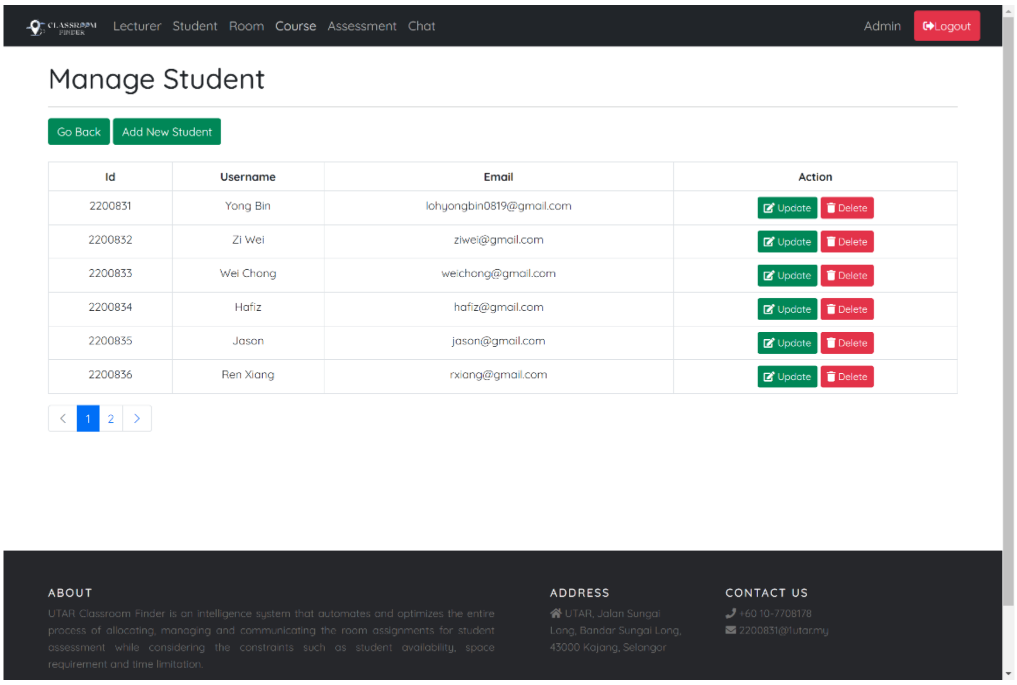
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.51: Update Lecturer Page (Administrator)

5.4.8 View Student Page (Administrator)



Manage Student

Id	Username	Email	Action
2200831	Yong Bin	lohyongbin0819@gmail.com	Update Delete
2200832	Zi Wei	ziwei@gmail.com	Update Delete
2200833	Wei Chong	weichong@gmail.com	Update Delete
2200834	Hafiz	hafiz@gmail.com	Update Delete
2200835	Jason	jason@gmail.com	Update Delete
2200836	Ren Xiang	rxiang@gmail.com	Update Delete

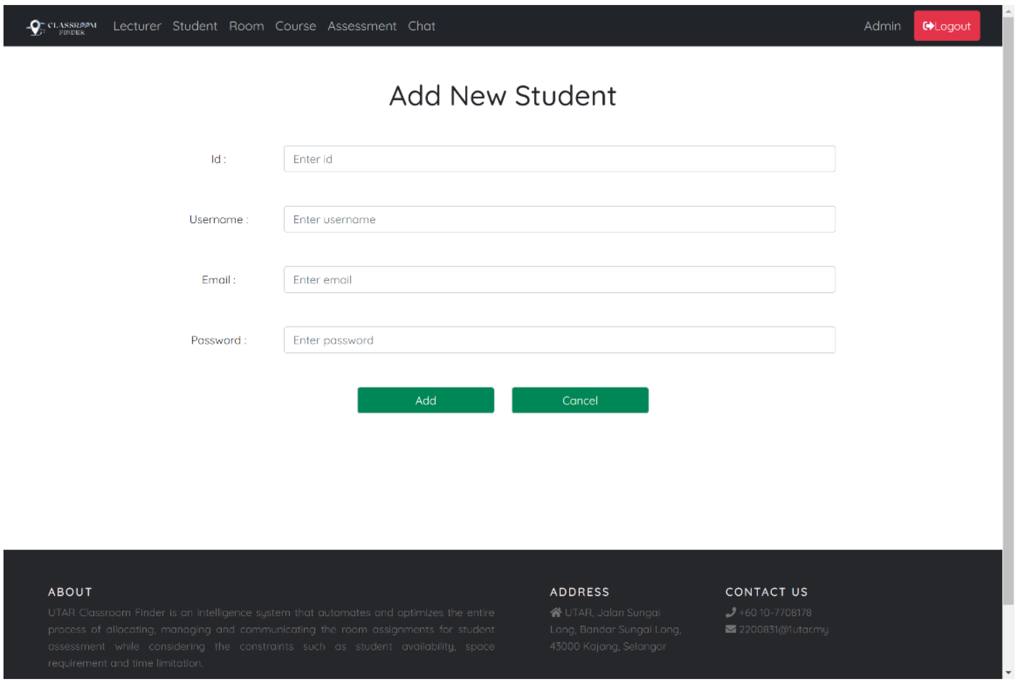
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.52: View Student Page (Administrator)

5.4.9 Add Student Page (Administrator)



Add New Student

Id :	Enter id
Username :	Enter username
Email :	Enter email
Password :	Enter password

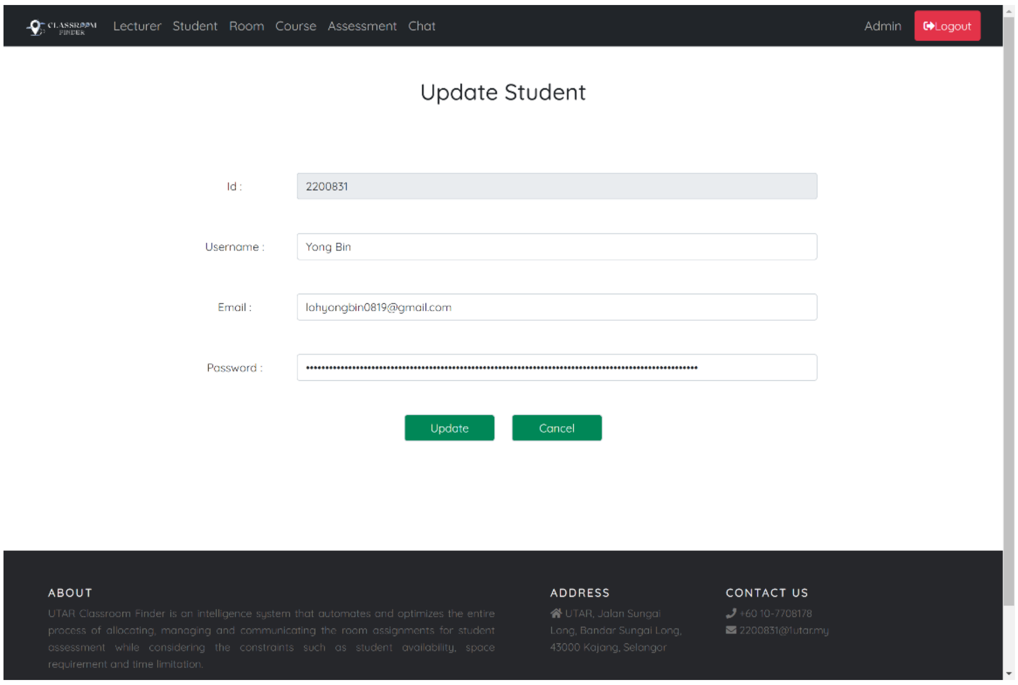
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.53: Add Student Page (Administrator)

5.4.10 Update Student Page (Administrator)



Update Student

Id : 2200831

Username : Yong Bin

Email : lohyongbin0819@gmail.com

Password : ••

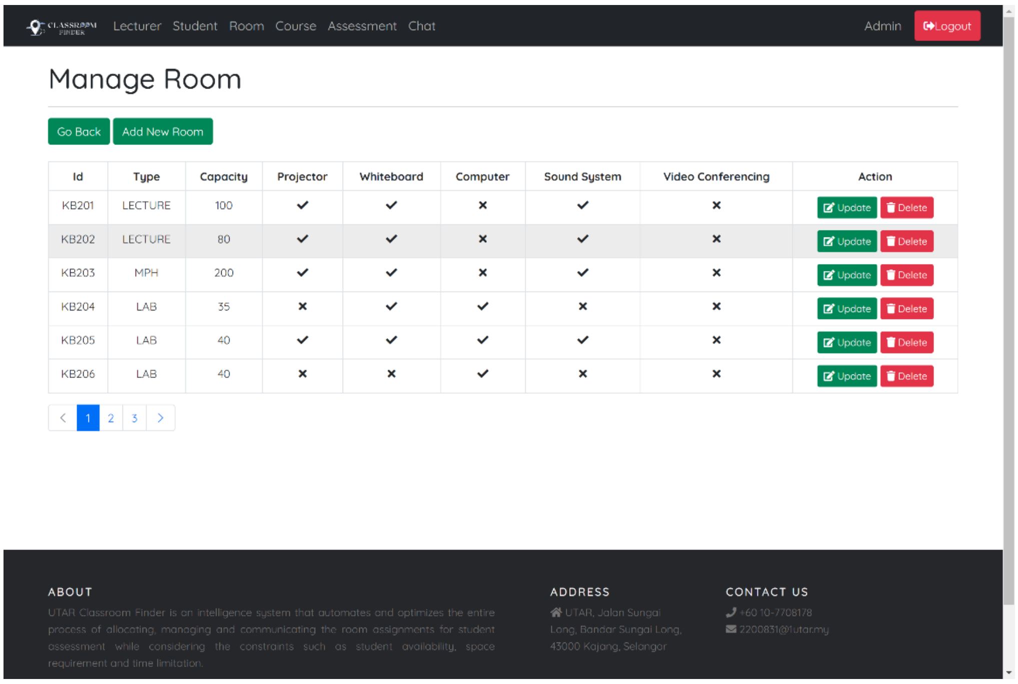
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.54: Update Student Page (Administrator)

5.4.11 View Room Page (Administrator)



Manage Room

Id	Type	Capacity	Projector	Whiteboard	Computer	Sound System	Video Conferencing	Action
KB201	LECTURE	100	✓	✓	✗	✓	✗	Update Delete
KB202	LECTURE	80	✓	✓	✗	✓	✗	Update Delete
KB203	MPH	200	✓	✓	✗	✓	✗	Update Delete
KB204	LAB	35	✗	✓	✓	✗	✗	Update Delete
KB205	LAB	40	✓	✓	✓	✓	✗	Update Delete
KB206	LAB	40	✗	✗	✓	✗	✗	Update Delete

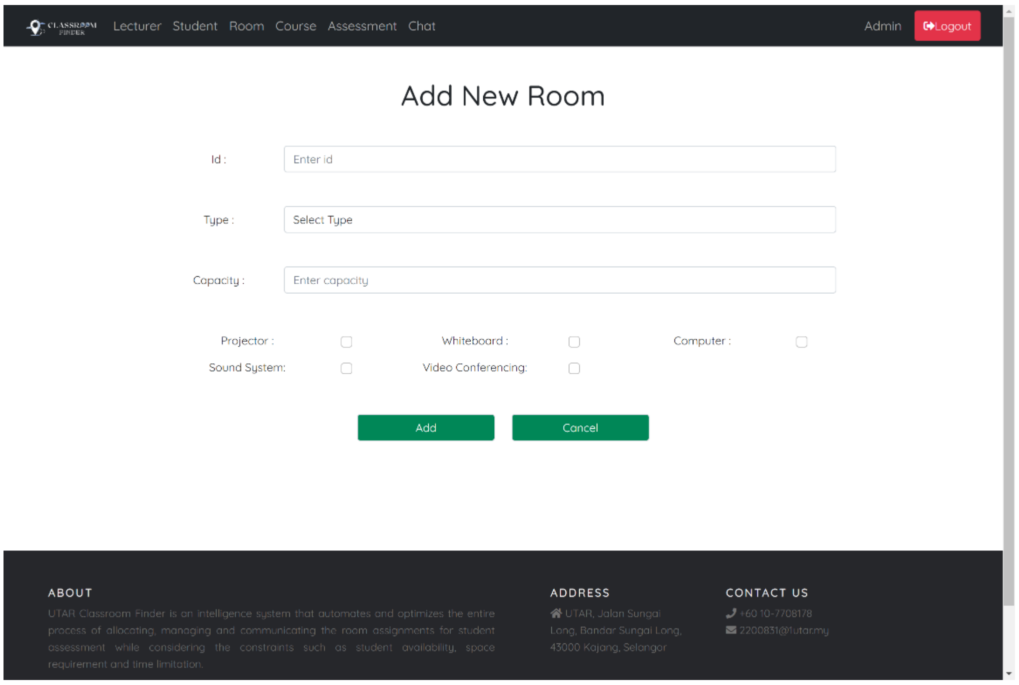
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.55: View Room Page (Administrator)

5.4.12 Add Room Page (Administrator)



Add New Room

Id :	Enter id
Type :	Select Type
Capacity :	Enter capacity

Projector : ☐ Whiteboard : ☐ Computer : ☐
Sound System: ☐ Video Conferencing: ☐

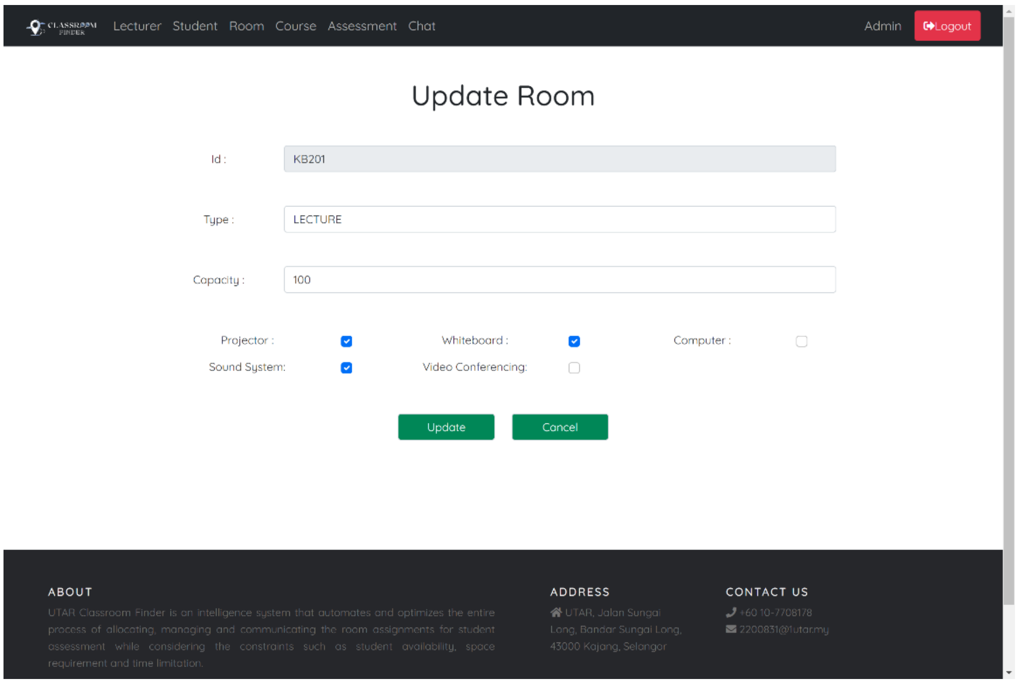
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.56: Add Room Page (Administrator)

5.4.13 Update Room Page (Administrator)



Update Room

Id :	KB201
Type :	LECTURE
Capacity :	100

Projector : ☑ Whiteboard : ☑ Computer : ☐
Sound System: ☑ Video Conferencing: ☐

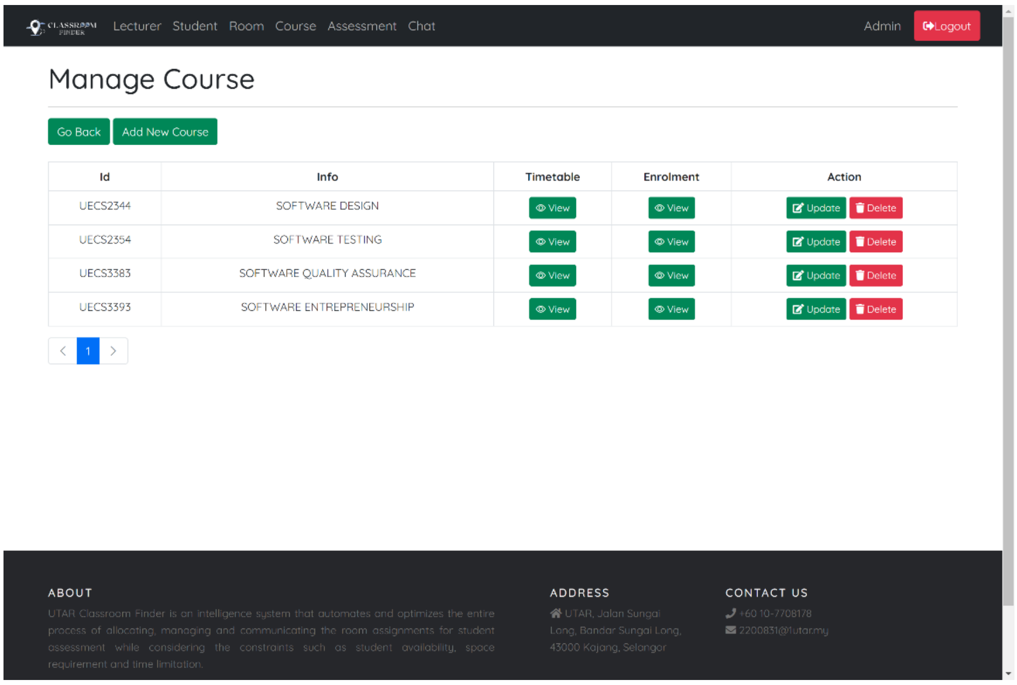
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.57: Update Room Page (Administrator)

5.4.14 View Course Page (Administrator)



Manage Course

Id	Info	Timetable	Enrolment	Action
UECS2344	SOFTWARE DESIGN	View	View	Update Delete
UECS2354	SOFTWARE TESTING	View	View	Update Delete
UECS3383	SOFTWARE QUALITY ASSURANCE	View	View	Update Delete
UECS3393	SOFTWARE ENTREPRENEURSHIP	View	View	Update Delete

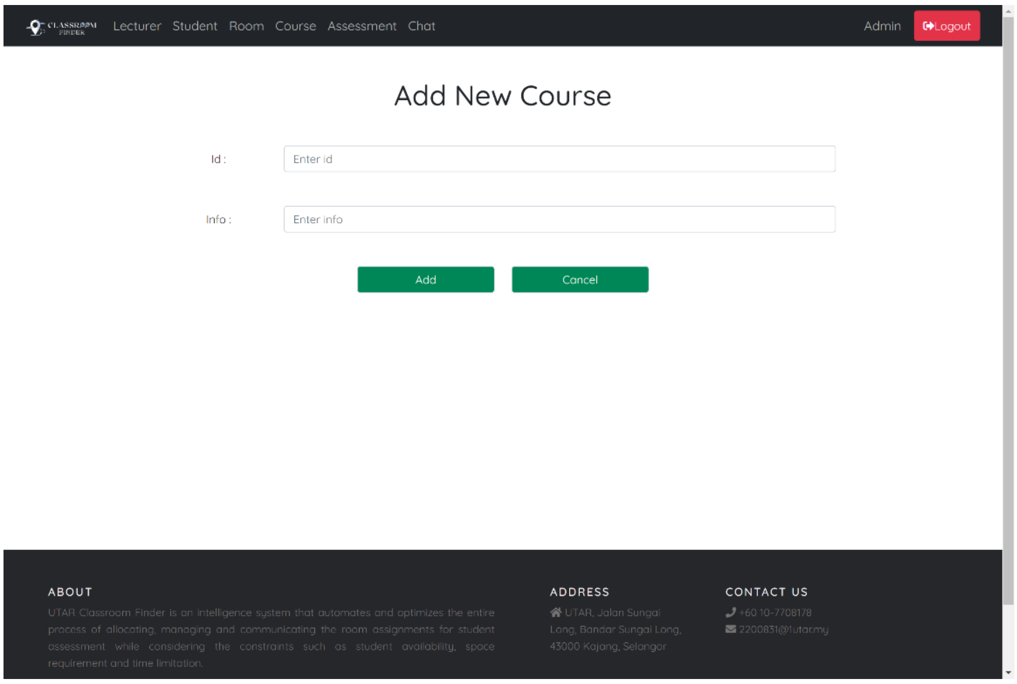
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.58: View Course Page (Administrator)

5.4.15 Add Course Page (Administrator)



Add New Course

Id : Enter id

Info : Enter info

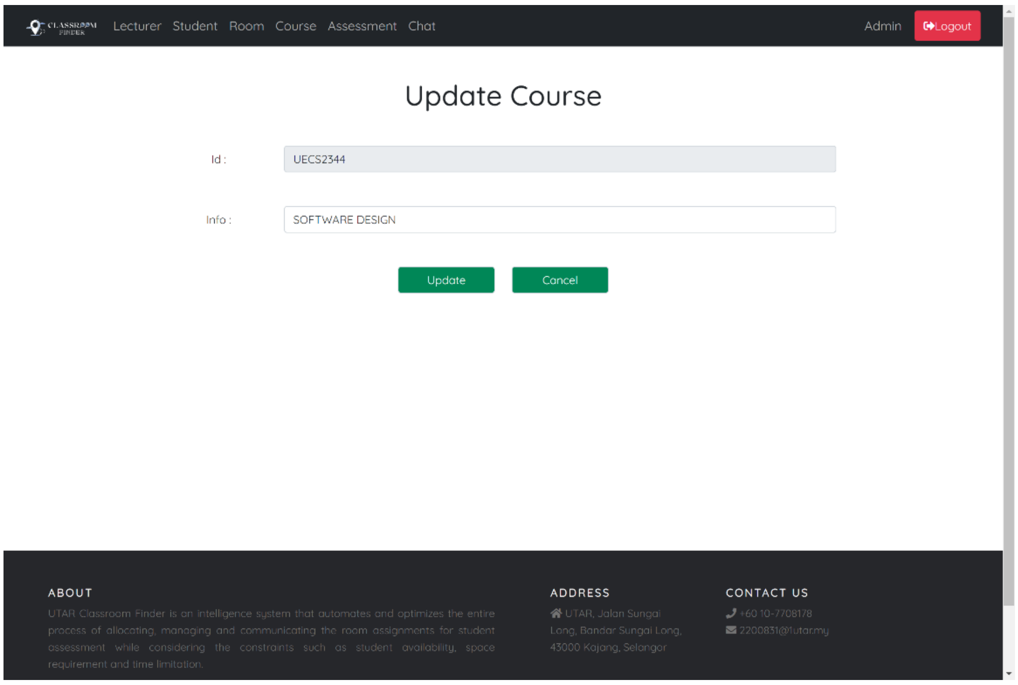
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.59: Add Course Page (Administrator)

5.4.16 Update Course Page (Administrator)



Update Course

Id : UECS2344

Info : SOFTWARE DESIGN

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

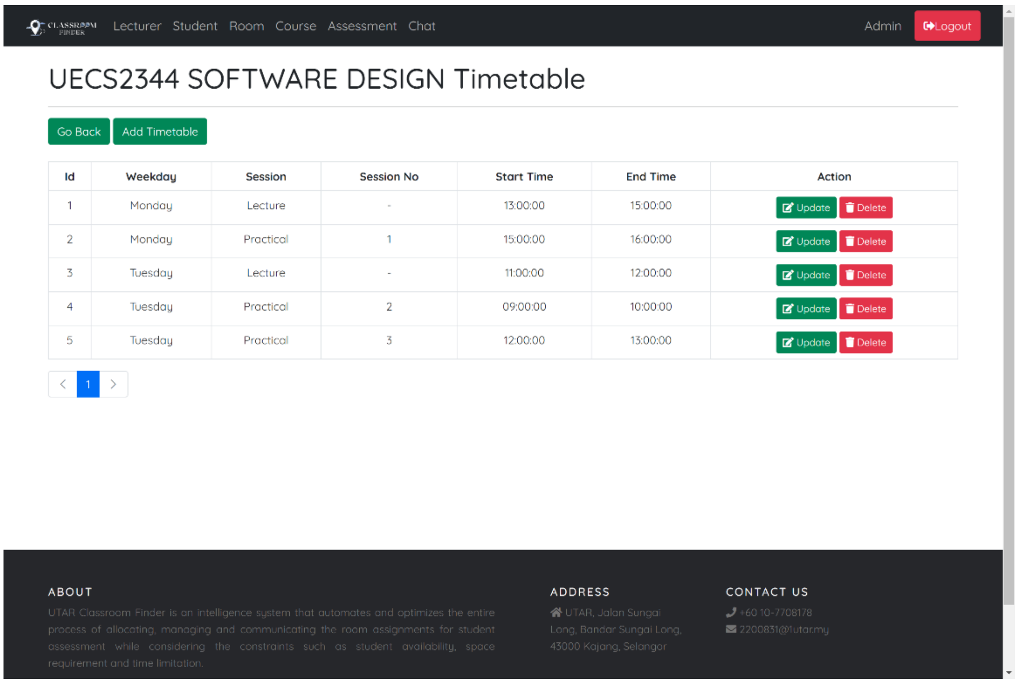
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.60: Update Course Page (Administrator)

5.4.17 View Course Timetable Page (Administrator)



UECS2344 SOFTWARE DESIGN Timetable

Id	Weekday	Session	Session No	Start Time	End Time	Action
1	Monday	Lecture	-	13:00:00	15:00:00	Update Delete
2	Monday	Practical	1	15:00:00	16:00:00	Update Delete
3	Tuesday	Lecture	-	11:00:00	12:00:00	Update Delete
4	Tuesday	Practical	2	09:00:00	10:00:00	Update Delete
5	Tuesday	Practical	3	12:00:00	13:00:00	Update Delete

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

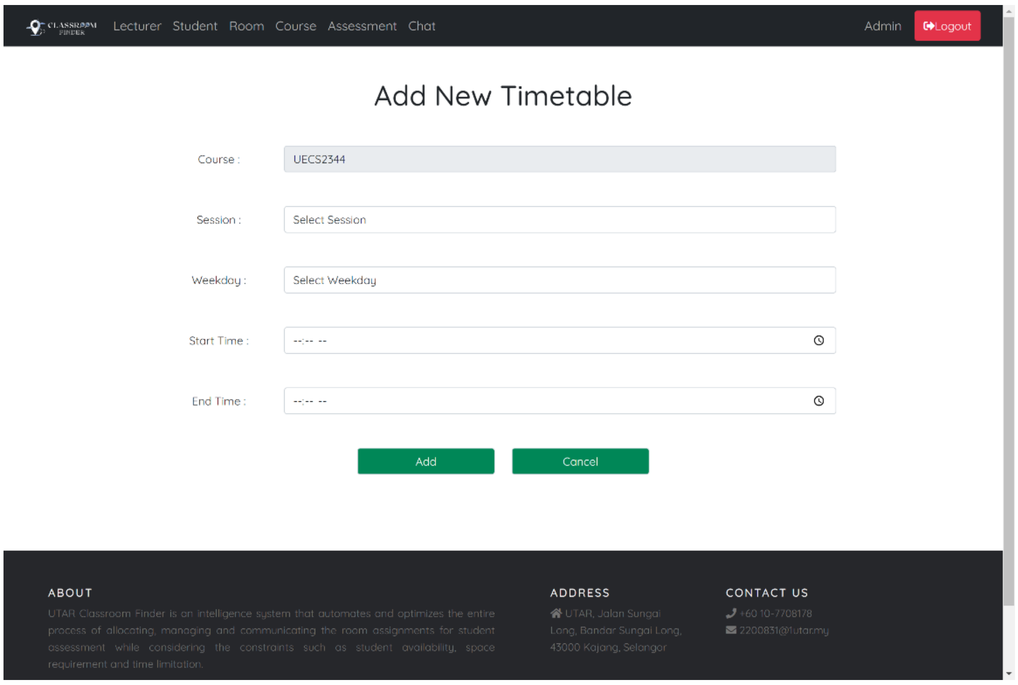
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.61: View Course Timetable Page (Administrator)

5.4.18 Add Course Timetable Page (Administrator)



Add New Timetable

Course : UECS2344

Session : Select Session

Weekday : Select Weekday

Start Time : --:-- --

End Time : --:-- --

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

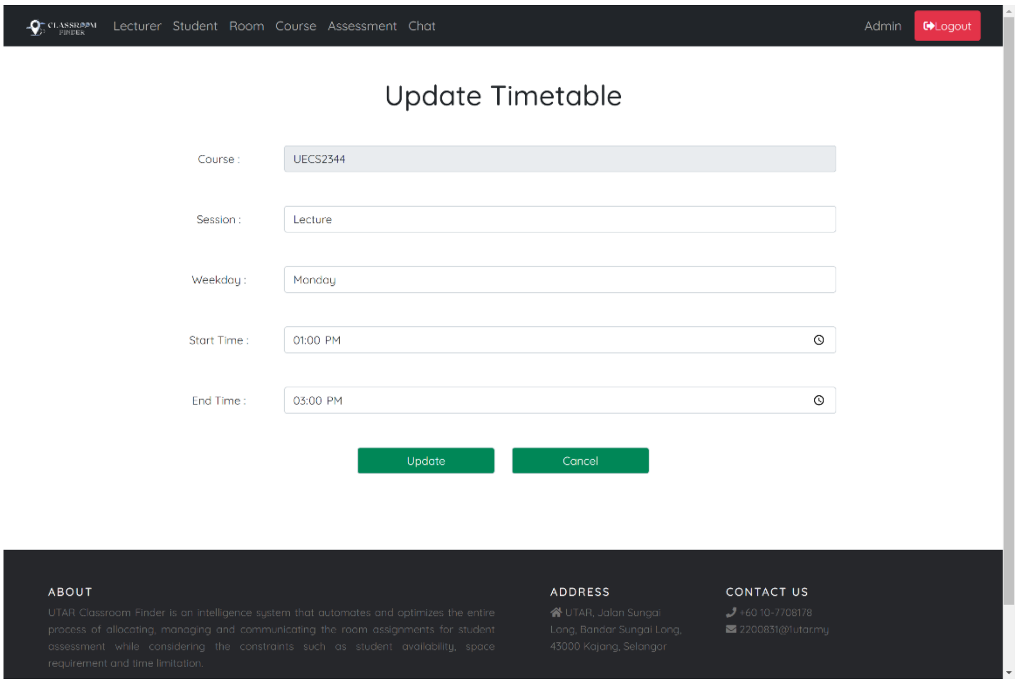
CONTACT US

+60 10-7708178

2200831@1utar.my

Figure 5.62: Add Course Timetable Page (Administrator)

5.4.19 Update Course Timetable Page (Administrator)



Update Timetable

Course : UECS2344

Session : Lecture

Weekday : Monday

Start Time : 01:00 PM

End Time : 03:00 PM

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

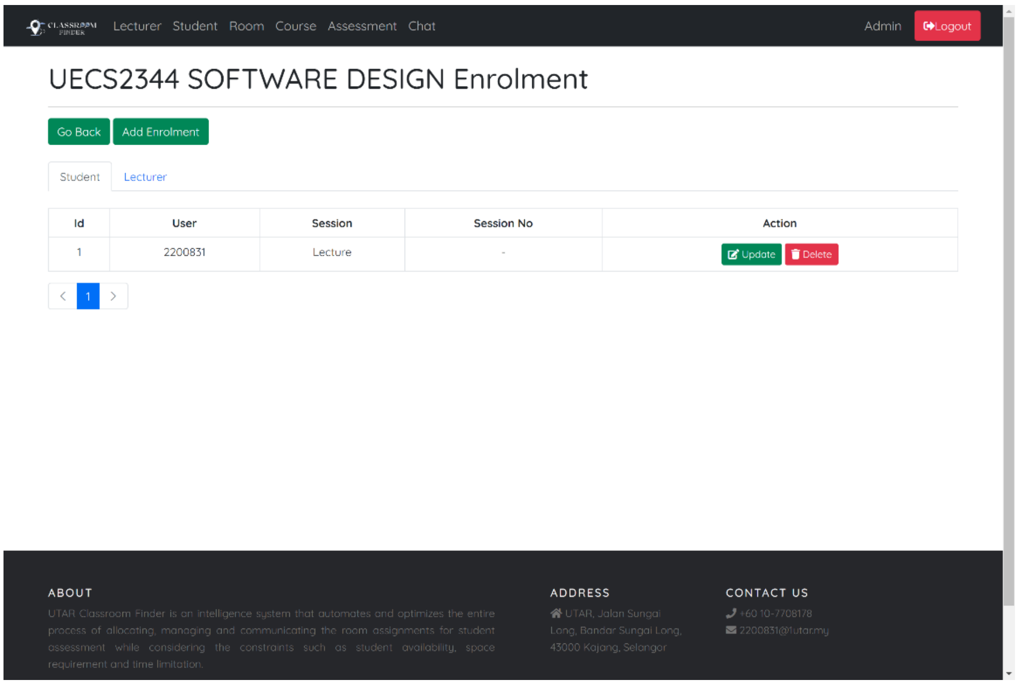
CONTACT US

+60 10-7708178

2200831@1utar.my

Figure 5.63: Update Course Timetable Page (Administrator)

5.4.20 View Course Student Enrolment Page (Administrator)



UECS2344 SOFTWARE DESIGN Enrolment

Id	User	Session	Session No	Action
1	2200831	Lecture	-	Update Delete

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

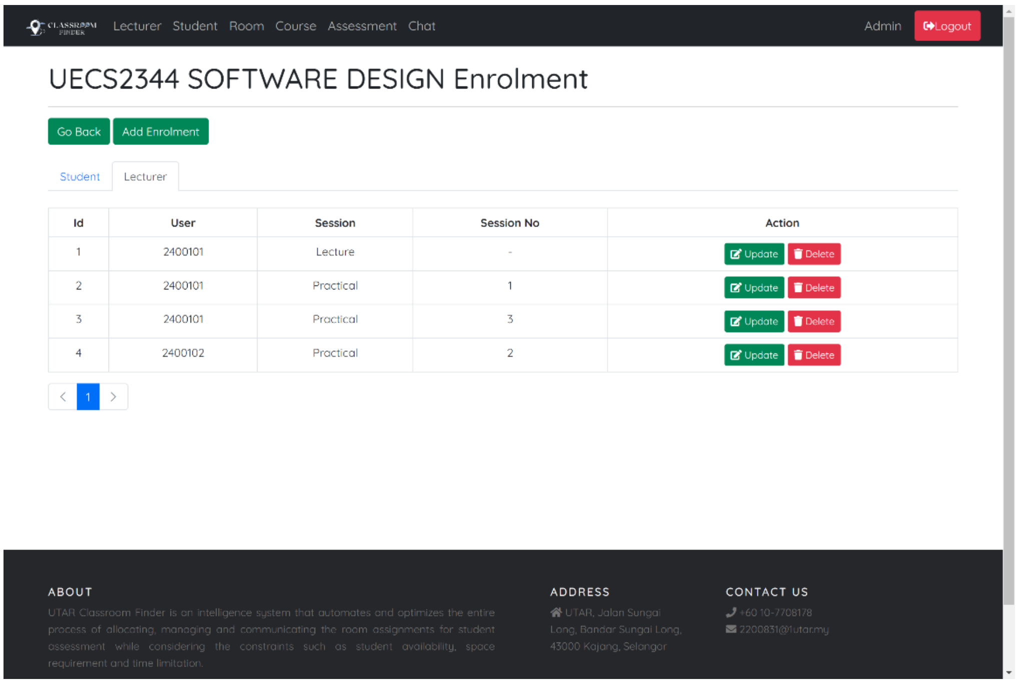
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.64: View Course Student Enrolment Page (Administrator)

5.4.21 View Course Lecturer Enrolment Page (Administrator)



UECS2344 SOFTWARE DESIGN Enrolment

Id	User	Session	Session No	Action
1	2400101	Lecture	-	Update Delete
2	2400101	Practical	1	Update Delete
3	2400101	Practical	3	Update Delete
4	2400102	Practical	2	Update Delete

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

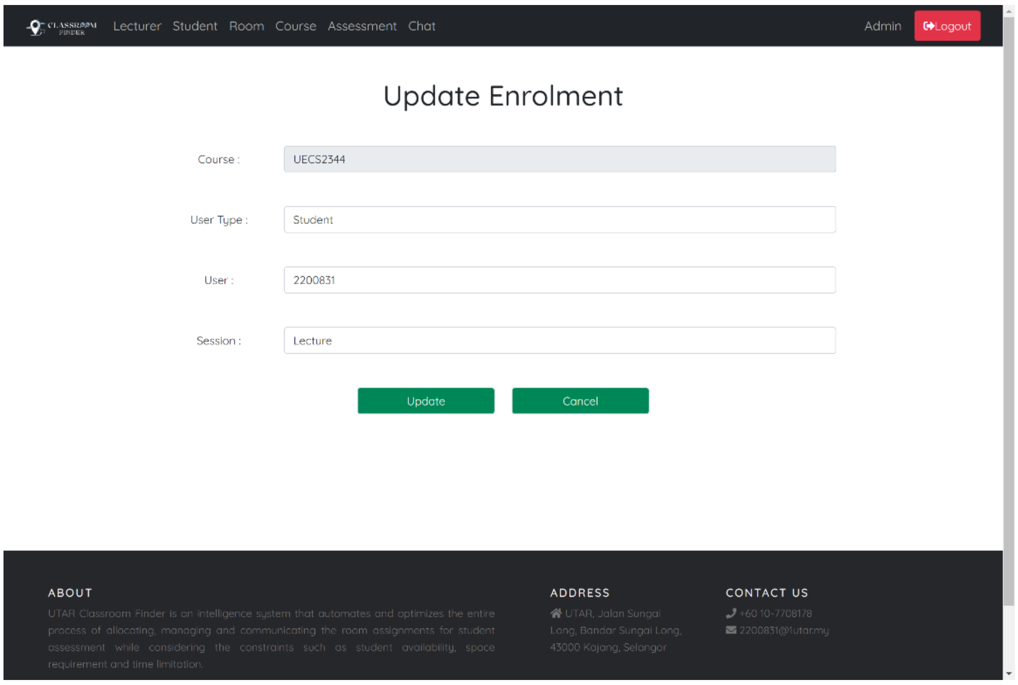
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.65: View Course Lecturer Enrolment Page (Administrator)

5.4.22 Update Course Enrolment Page (Administrator)



Update Enrolment

Course :	UECS2344
User Type :	Student
User :	2200831
Session :	Lecture

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

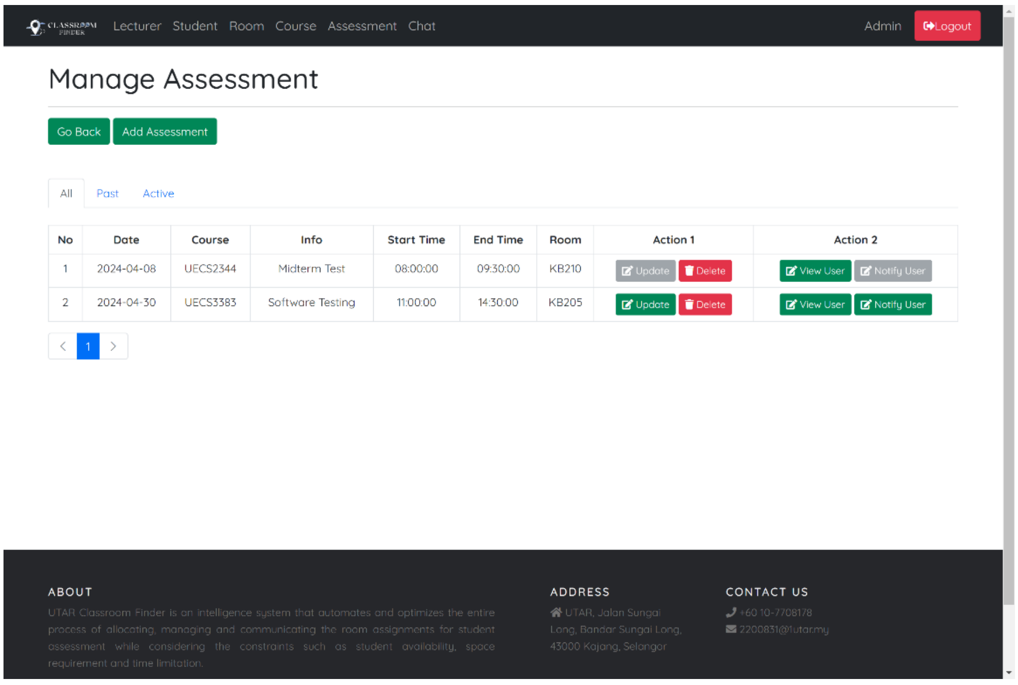
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.66: Update Course Enrolment Page (Administrator)

5.4.23 View All Assessment Page (Administrator)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-30	UECS3383	Software Testing	11:00:00	14:30:00	KB205	Update Delete	View User Notify User

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

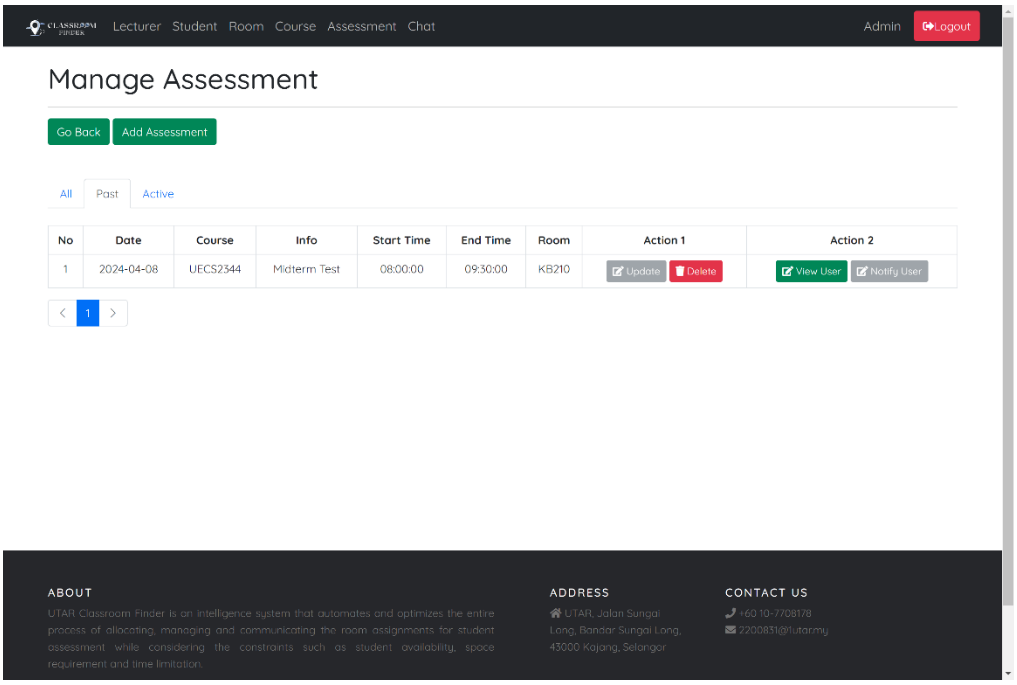
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.67: View All Assessment Page (Administrator)

5.4.24 View Past Assessment Page (Administrator)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

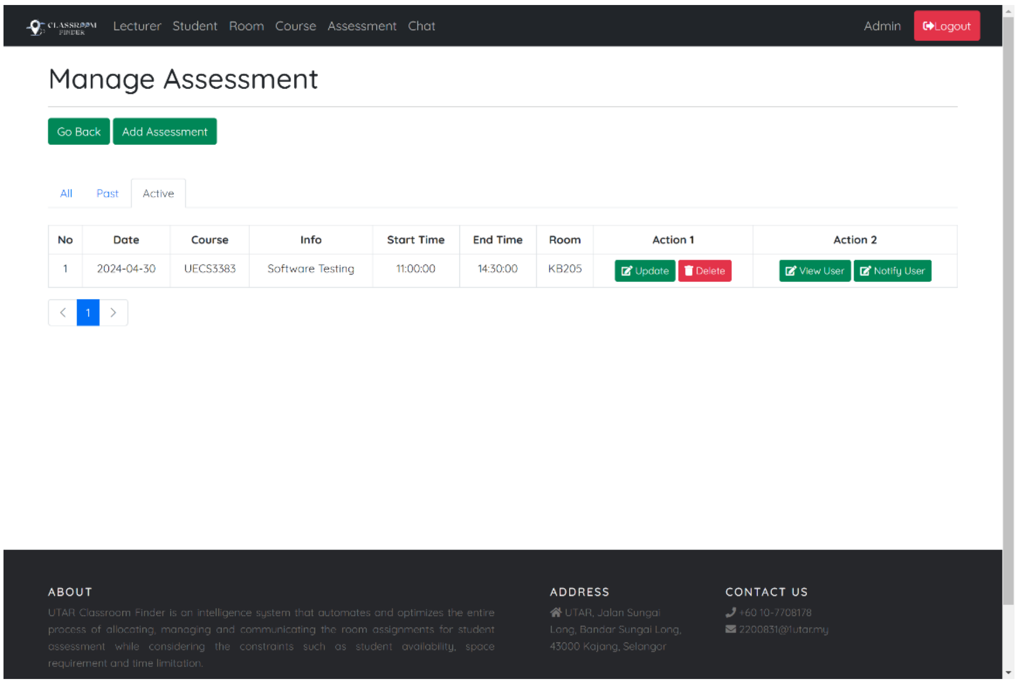
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.68: View Past Assessment Page (Administrator)

5.4.25 View Active Assessment Page (Administrator)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-30	UECS3383	Software Testing	11:00:00	14:30:00	KB205	Update Delete	View User Notify User

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

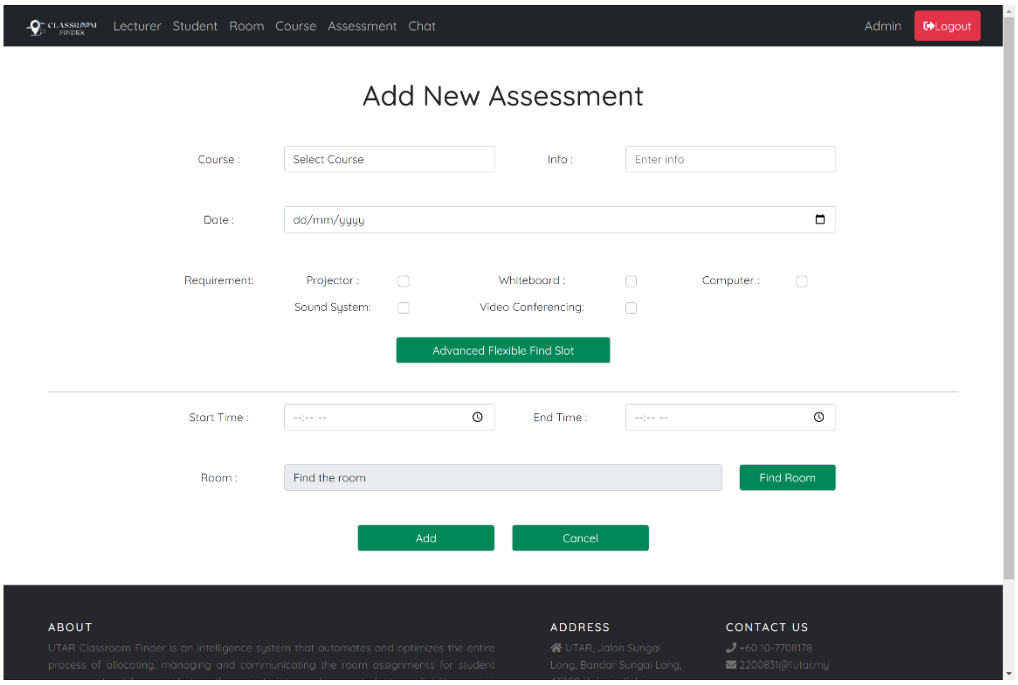
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.69: View Active Assessment Page (Administrator)

5.4.26 Add Assessment Page (Administrator)



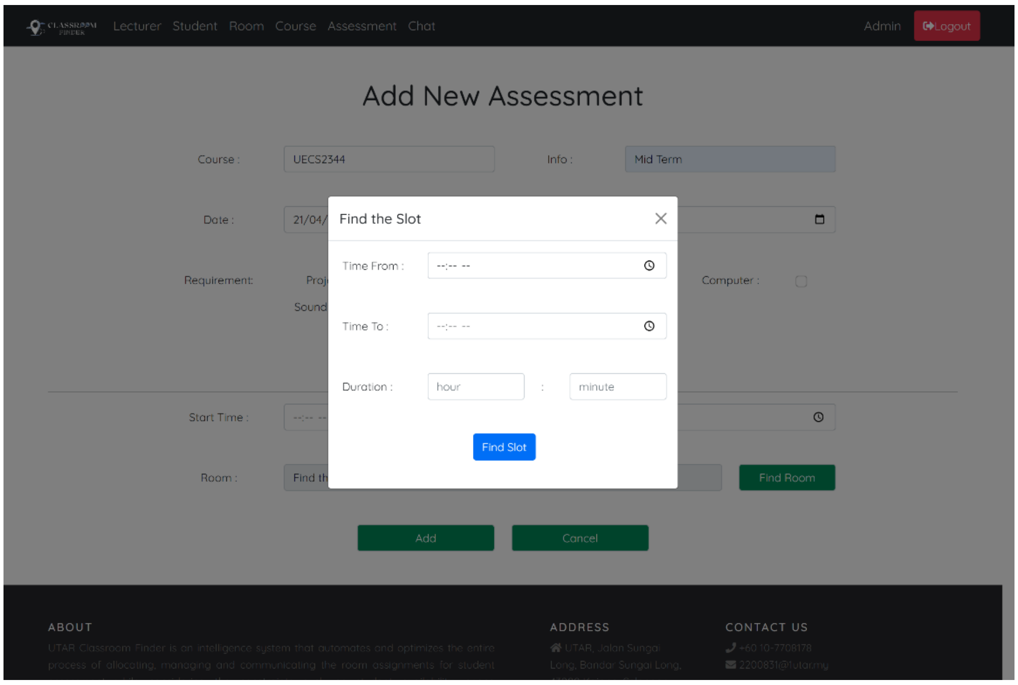
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student

UTAR, Jalan Sungai Long, Bandar Sungai Long,

+60 10-7708178

2200831@1utar.my

Figure 5.70: Add Assessment Page (Administrator)



Course : UECS2344

Info : Mid Term

Date : 21/04/

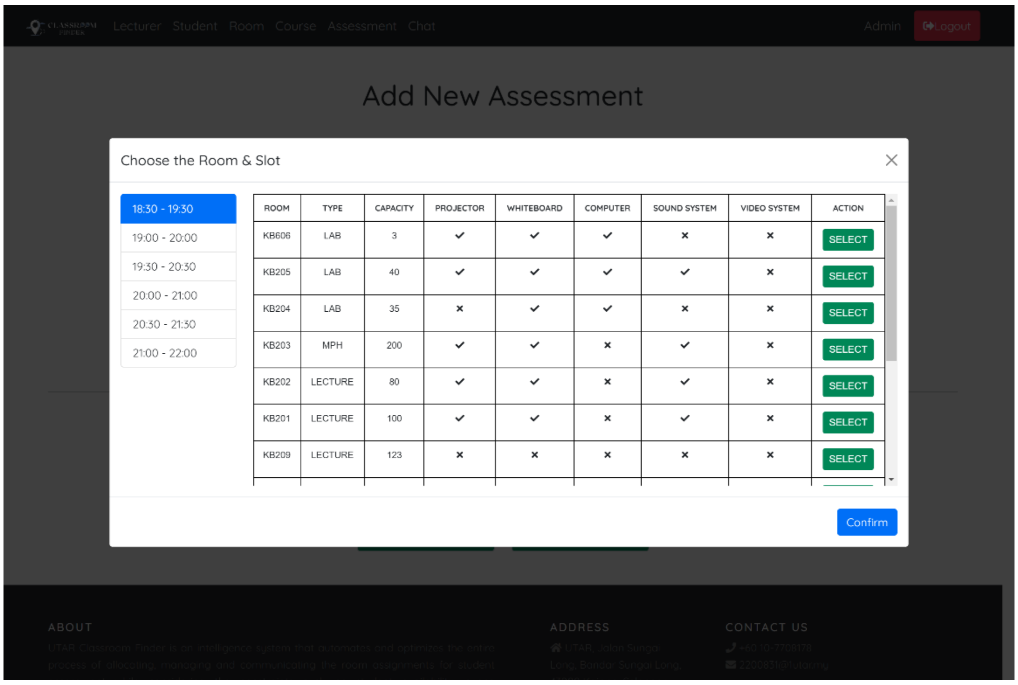
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student

UTAR, Jalan Sungai Long, Bandar Sungai Long,

+60 10-7708178

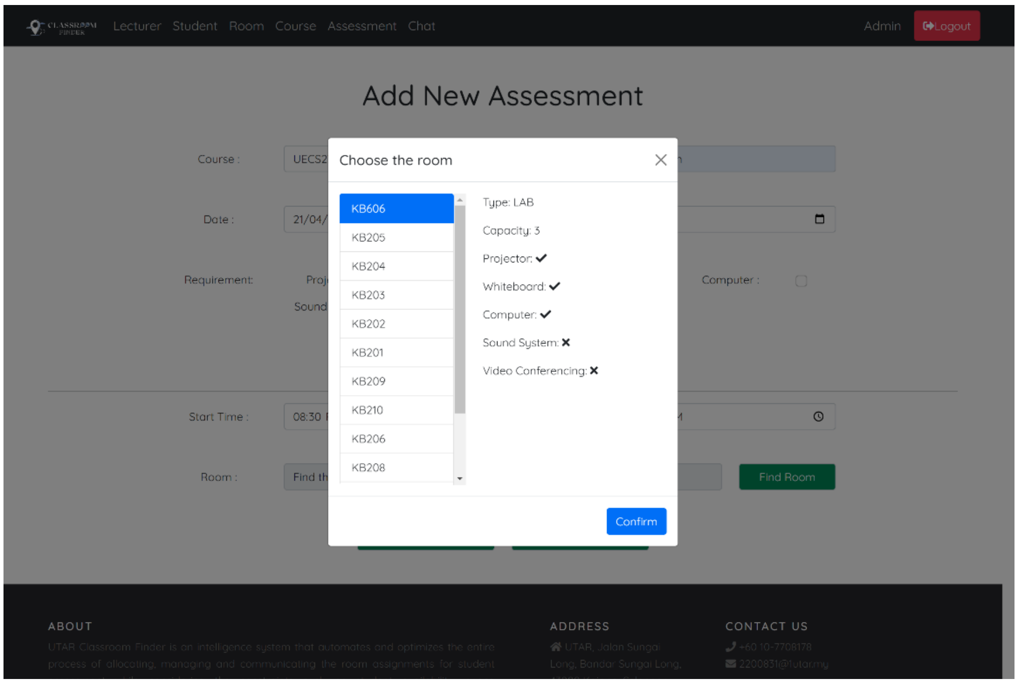
2200831@1utar.my

Figure 5.71: Advance Find Slot and Room (Administrator)



ROOM	TYPE	CAPACITY	PROJECTOR	WHITEBOARD	COMPUTER	SOUND SYSTEM	VIDEO SYSTEM	ACTION
KB606	LAB	3	✓	✓	✓	✗	✗	SELECT
KB205	LAB	40	✓	✓	✓	✓	✗	SELECT
KB204	LAB	35	✗	✓	✓	✗	✗	SELECT
KB203	MPH	200	✓	✓	✗	✓	✗	SELECT
KB202	LECTURE	80	✓	✓	✗	✓	✗	SELECT
KB201	LECTURE	100	✓	✓	✗	✓	✗	SELECT
KB209	LECTURE	123	✗	✗	✗	✗	✗	SELECT

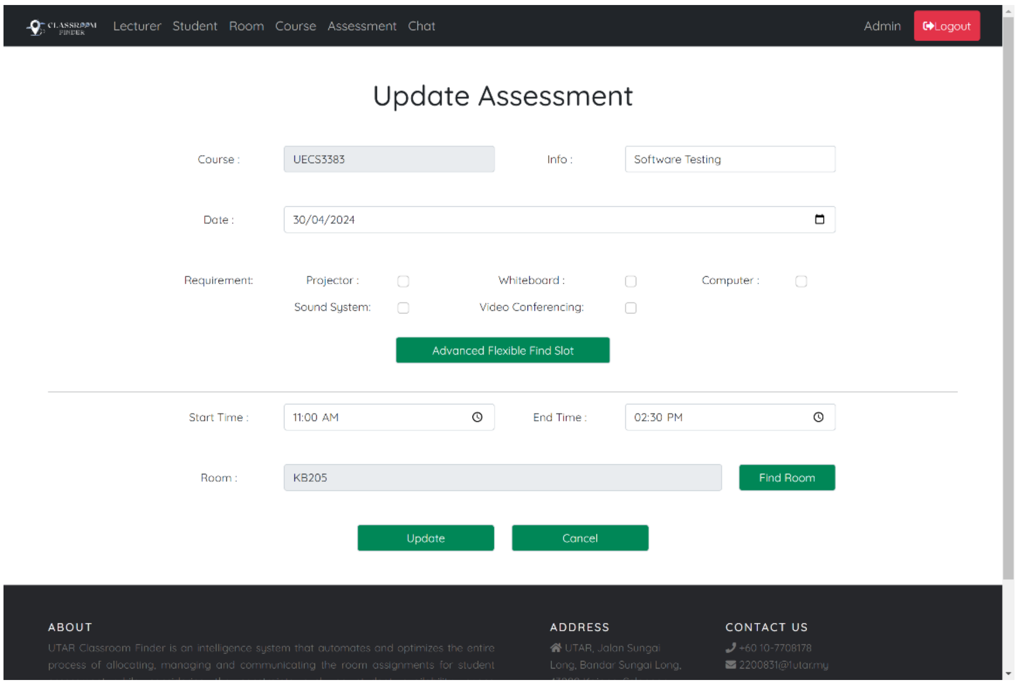
Figure 5.72: Advance Find Slot and Room 2 (Administrator)



Type: LAB
Capacity: 3
Projector: ✓
Whiteboard: ✓
Computer: ✓
Sound System: ✗
Video Conferencing: ✗

Figure 5.73: Find Room (Administrator)

5.4.27 Update Assessment Page (Administrator)


Lecturer Student Room Course Assessment Chat
Admin Logout

Update Assessment

Course :	UECS3383	Info :	Software Testing
Date :	30/04/2024		

Requirement: Projector : ☐ Whiteboard : ☐ Computer : ☐ Sound System: ☐ Video Conferencing: ☐

Advanced Flexible Find Slot

Start Time :	11:00 AM	End Time :	02:30 PM
Room :	KB205		

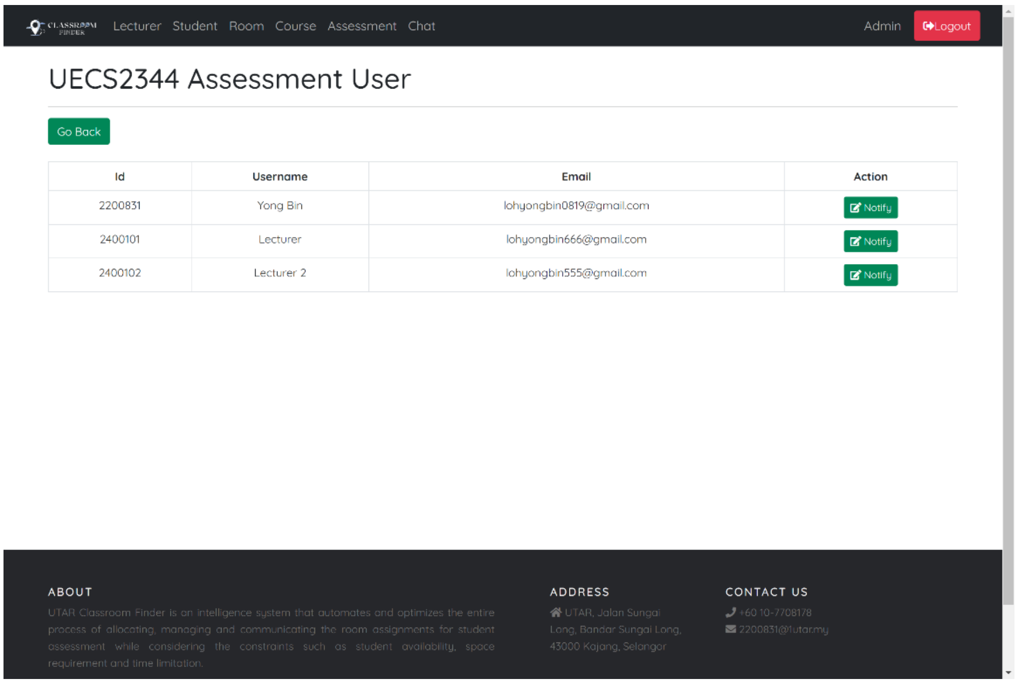
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long,

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.74: Update Assessment Page (Administrator)

5.4.28 View Assessment User Page (Administrator)



UECS2344 Assessment User

Id	Username	Email	Action
2200831	Yong Bin	lohyongbin0819@gmail.com	Notify
2400101	Lecturer	lohyongbin666@gmail.com	Notify
2400102	Lecturer 2	lohyongbin555@gmail.com	Notify

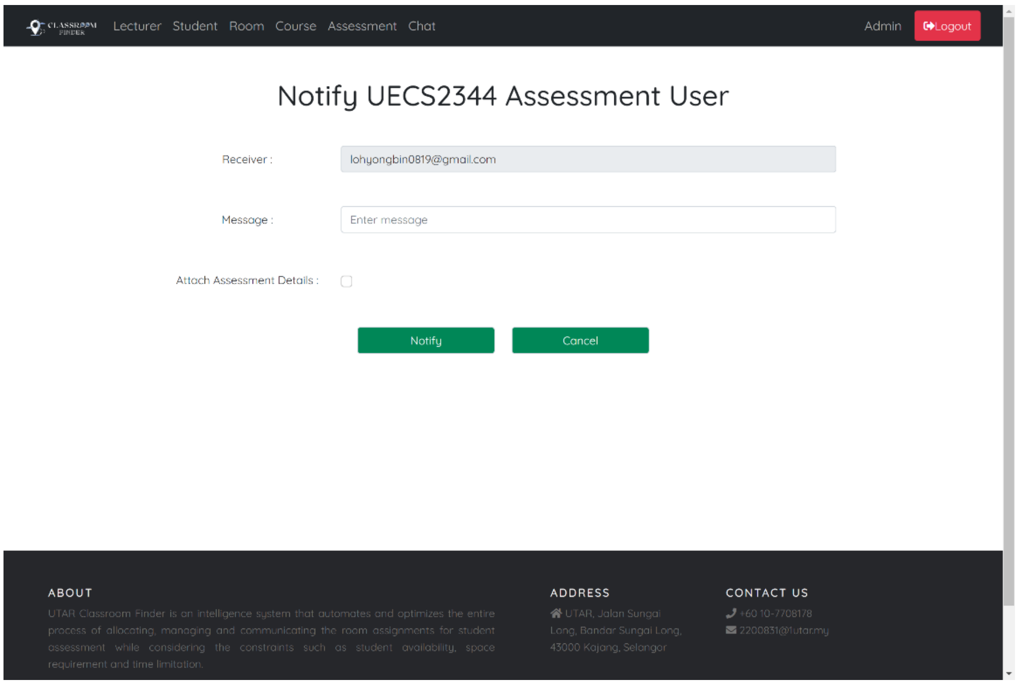
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.75: View Assessment User Page (Administrator)

5.4.29 Notify User Page (Administrator)



Notify UECS2344 Assessment User

Receiver :	lohyongbin0819@gmail.com
Message :	Enter message
Attach Assessment Details :	☐

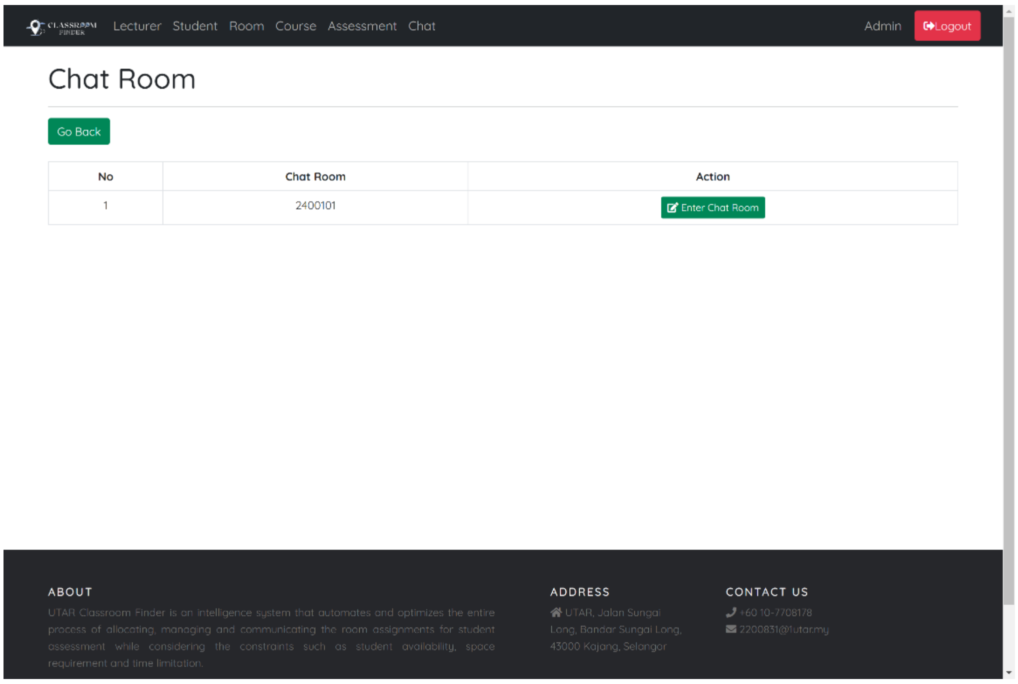
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.76: Notify User Page (Administrator)

5.4.30 View Chatroom Page (Administrator)



Chat Room

No	Chat Room	Action
1	2400101	Enter Chat Room

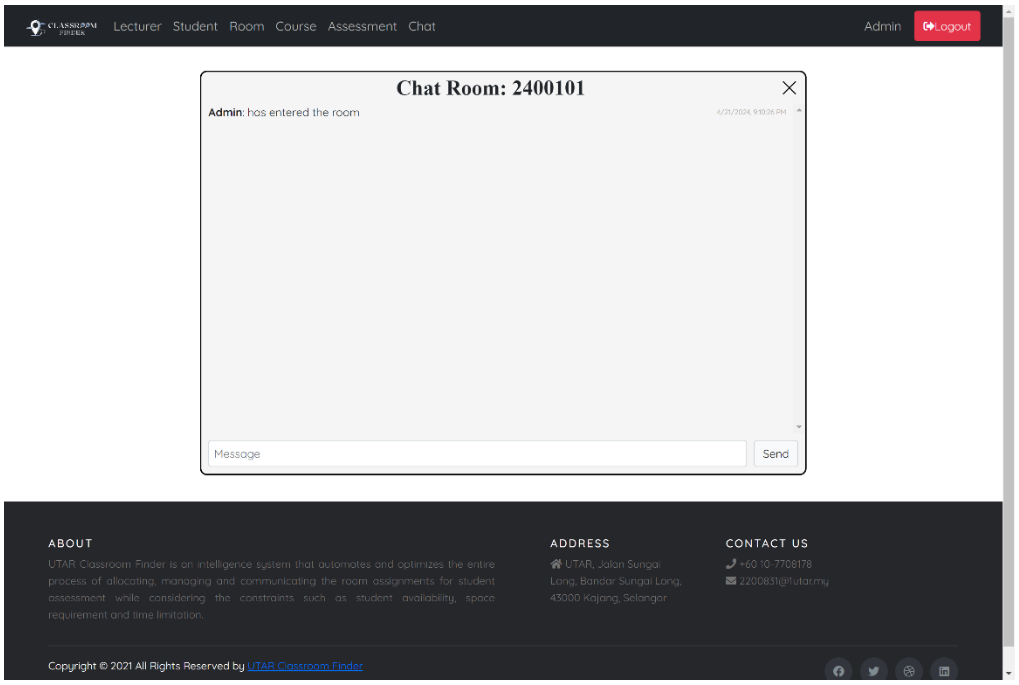
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.77: View Chatroom Page (Administrator)

5.4.31 Online Chat Page (Administrator)



Admin: has entered the room 4/21/2024, 9:10:26 PM

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

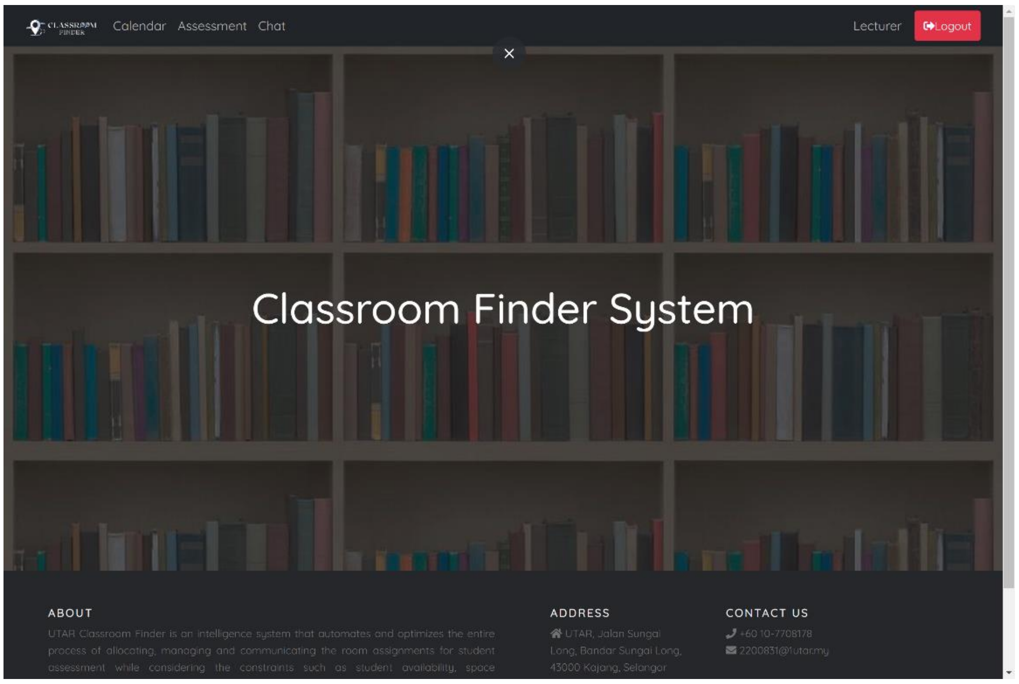
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

+60 10-7708178
2200831@1utar.my

Copyright © 2021 All Rights Reserved by UTAR Classroom Finder

Figure 5.78: Online Chat Page (Administrator)

5.4.32 Home Page (Lecturer)



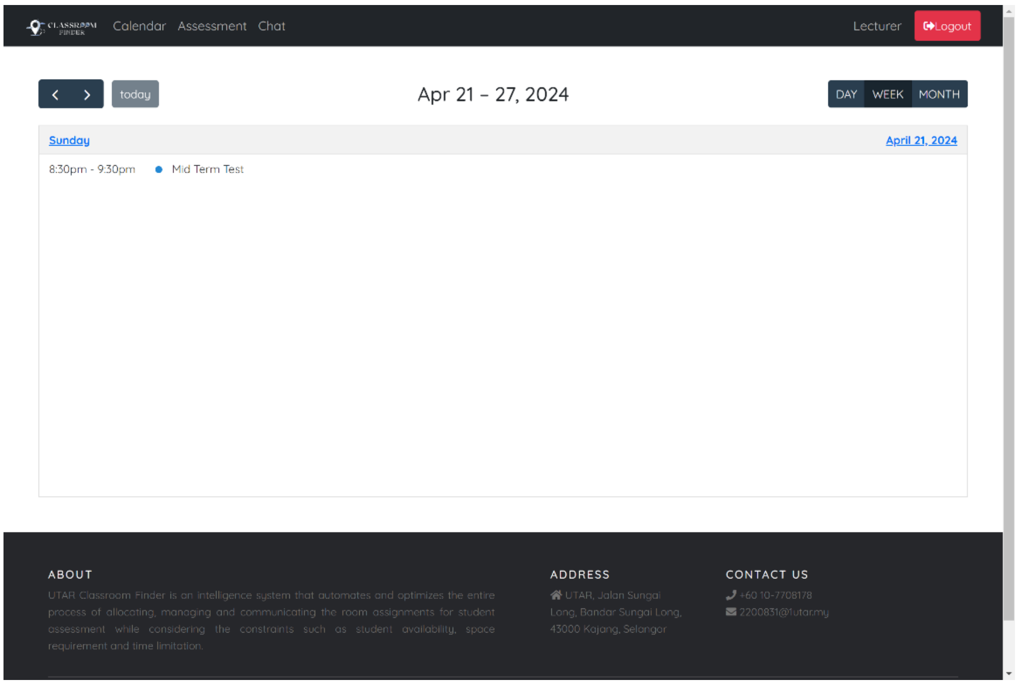
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

+60 10-7708178
2200831@1utar.my

Figure 5.79: Home Page (Lecturer)

5.4.33 View Calendar Page (Lecturer)



Apr 21 – 27, 2024

Sunday April 21, 2024

8:30pm - 9:30pm ● Mid Term Test

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

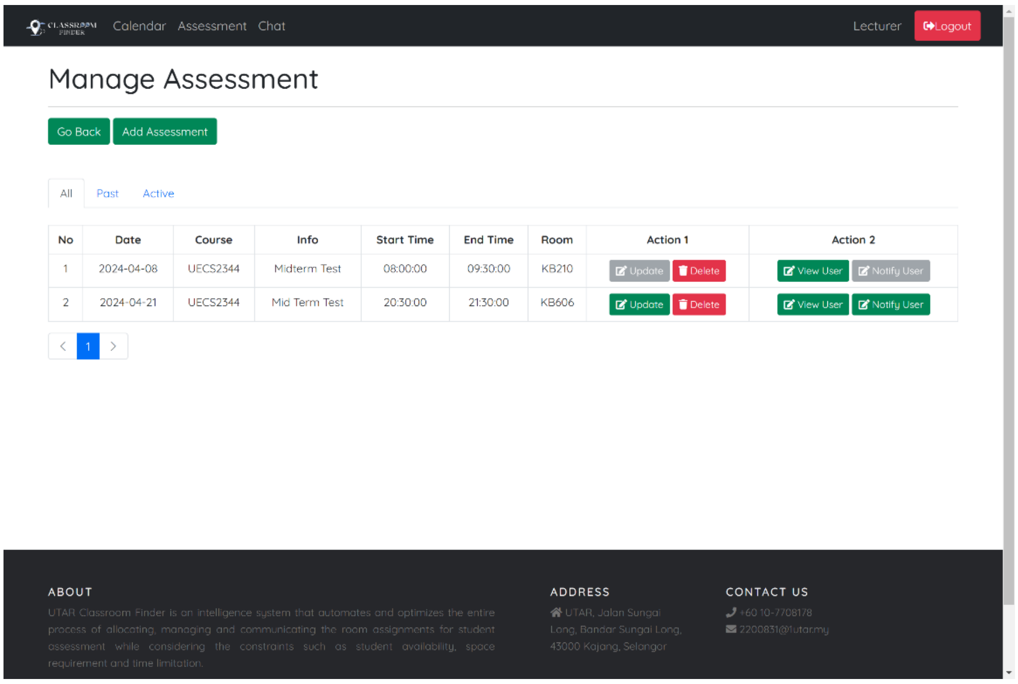
CONTACT US

+60 10-7708178

2200831@1utar.my

Figure 5.80: View Calendar Page (Lecturer)

5.4.34 View Assessment Page (Lecturer)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

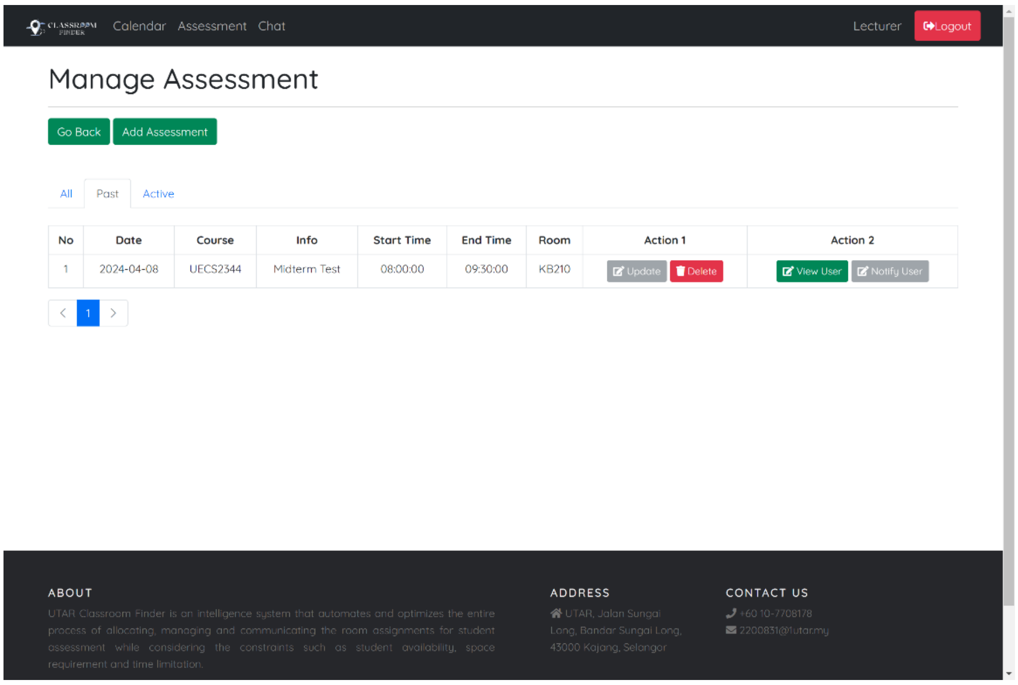
CONTACT US

+60 10-7708178

2200831@1utar.my

Figure 5.81: View Assessment Page (Lecturer)

5.4.35 View Past Assessment Page (Lecturer)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

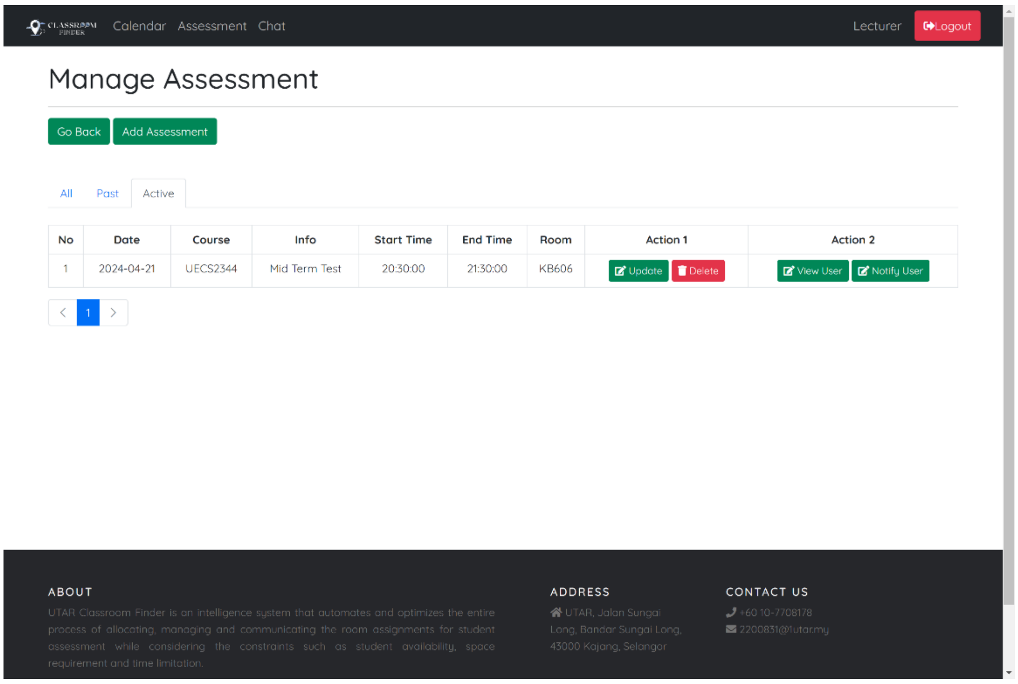
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.82: View Past Assessment Page (Lecturer)

5.4.36 View Active Assessment Page (Lecturer)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

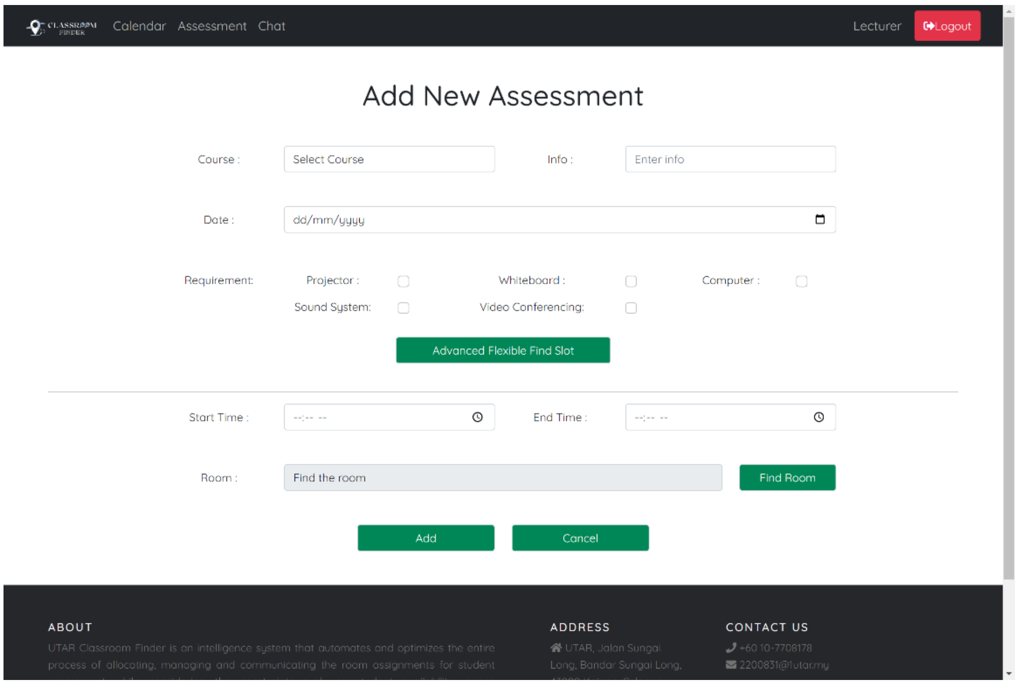
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.83: View Active Assessment Page (Lecturer)

5.4.37 Add Assessment Page (Lecturer)



Add New Assessment

Course :	Select Course	Info :	Enter info
Date :	dd/mm/yyyy		

Requirement: Projector : ☐ Whiteboard : ☐ Computer : ☐ Sound System: ☐ Video Conferencing: ☐

Start Time :	--:-- --	End Time :	--:-- --
Room :	Find the room		

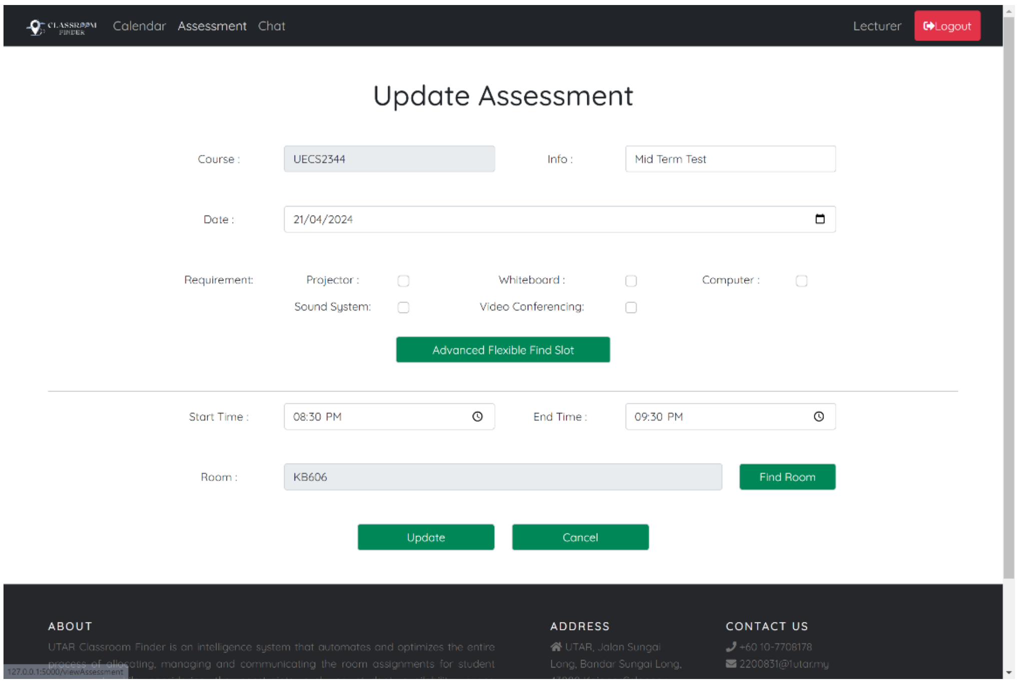
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long,

CONTACT US
+60 10-7708178
2200831@1utar.my

Figure 5.84: Add Assessment Page (Lecturer)

5.4.38 Update Assessment Page (Lecturer)



Update Assessment

Course :	UECS2344	Info :	Mid Term Test
Date :	21/04/2024		

Requirement: Projector : ☐ Whiteboard : ☐ Computer : ☐ Sound System: ☐ Video Conferencing: ☐

Start Time :	08:30 PM	End Time :	09:30 PM
Room :	KB606		

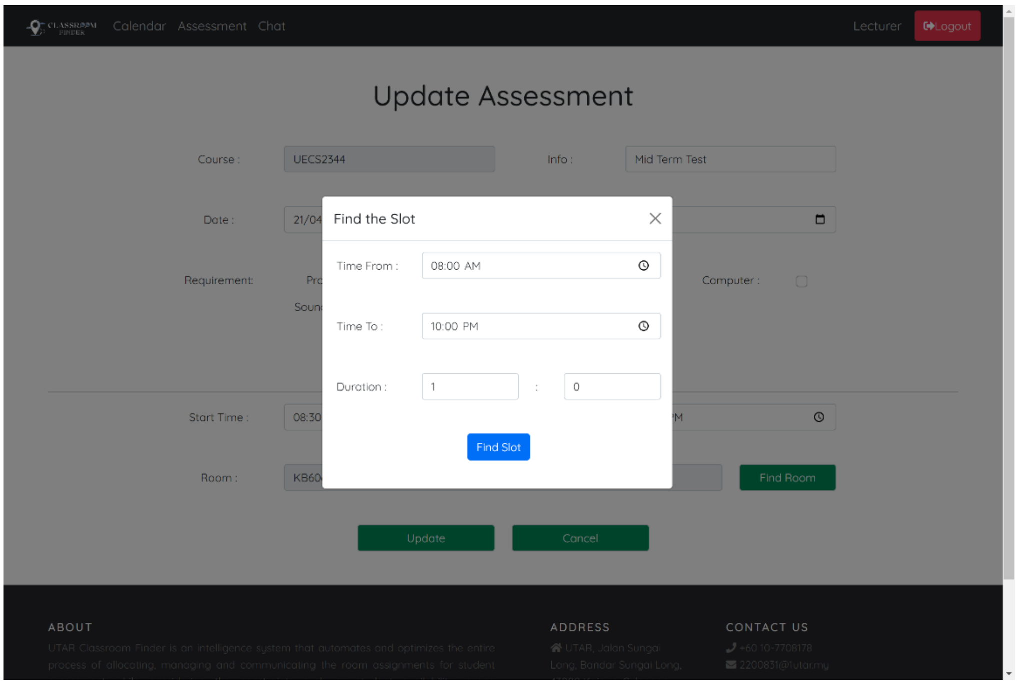
ABOUT
UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student

ADDRESS
UTAR, Jalan Sungai Long, Bandar Sungai Long,

CONTACT US
+60 10-7708178
2200831@1utar.my

127.0.0.1:5000/viewAssessment

Figure 5.85: Update Assessment Page (Lecturer)



Course :	UECS2344	Info :	Mid Term Test

Find the Slot

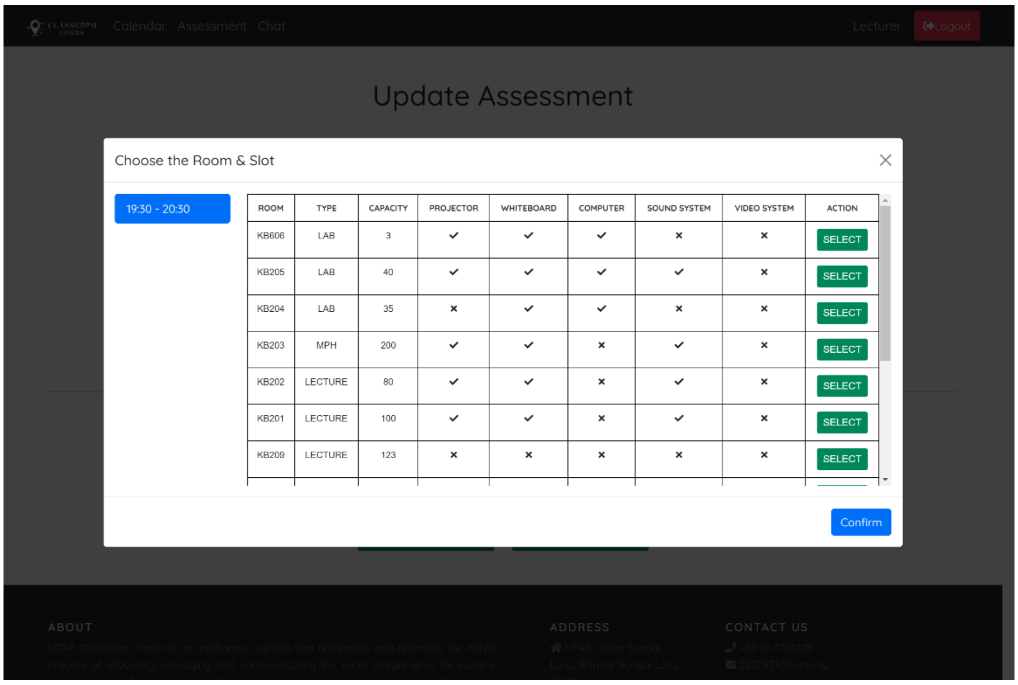
Time From :	08:00 AM
Time To :	10:00 PM
Duration :	1 : 0

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student

UTAR, Jalan Sungai Long, Bandar Sungai Long,

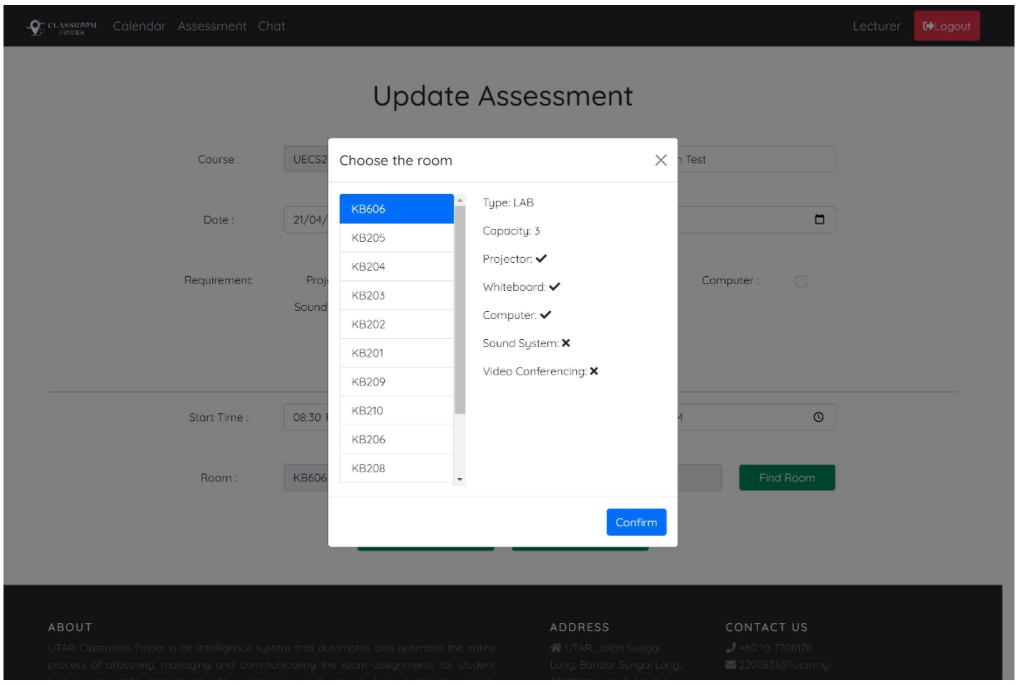
+6010-7708178

Figure 5.86: Advance Find Slot and Room



ROOM	TYPE	CAPACITY	PROJECTOR	WHITEBOARD	COMPUTER	SOUND SYSTEM	VIDEO SYSTEM	ACTION
KB606	LAB	3	✓	✓	✓	×	×	SELECT
KB205	LAB	40	✓	✓	✓	✓	×	SELECT
KB204	LAB	35	×	✓	✓	×	×	SELECT
KB203	MPH	200	✓	✓	×	✓	×	SELECT
KB202	LECTURE	80	✓	✓	×	✓	×	SELECT
KB201	LECTURE	100	✓	✓	×	✓	×	SELECT
KB209	LECTURE	123	×	×	×	×	×	SELECT

Figure 5.87: Advance Find Slot and Room 2


Update Assessment

Choose the room

KB606
KB205
KB204
KB203
KB202
KB201
KB209
KB210
KB206
KB208

Type: LAB

Capacity: 3

Projector: ✓

Whiteboard: ✓

Computer: ✓

Sound System: ✗

Video Conferencing: ✗

Confirm

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long,

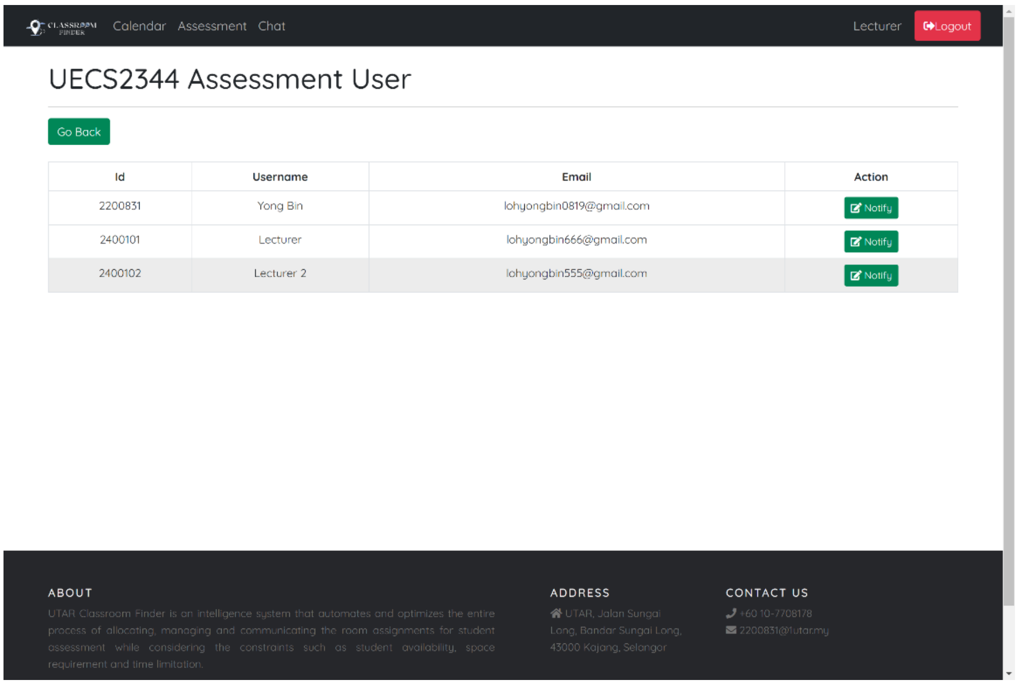
CONTACT US

+60 10-7708178

2200831@1utar.my

Figure 5.88: Find Room

5.4.39 View Assessment User Page (Lecturer)


UECS2344 Assessment User

Id	Username	Email	Action
2200831	Yong Bin	lohyongbin0819@gmail.com	Notify
2400101	Lecturer	lohyongbin666@gmail.com	Notify
2400102	Lecturer 2	lohyongbin555@gmail.com	Notify

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

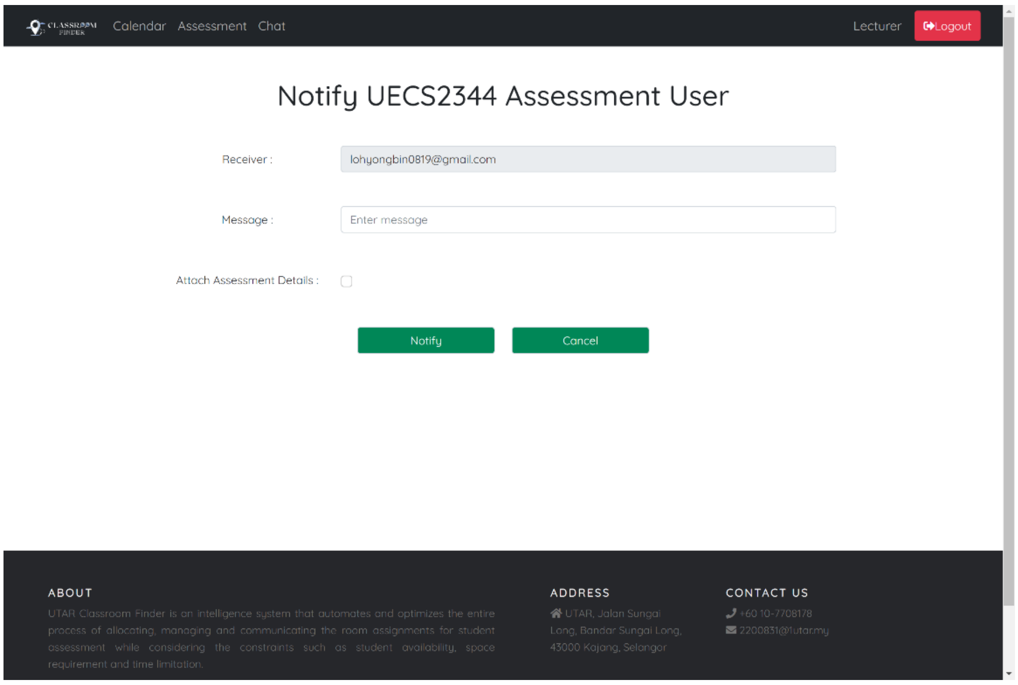
CONTACT US

+60 10-7708178

2200831@1utar.my

Figure 5.89: View Assessment User Page (Lecturer)

5.4.40 Notify Assessment User Page (Lecturer)



Notify UECS2344 Assessment User

Receiver : lohyongbin0819@gmail.com

Message : Enter message

Attach Assessment Details : ☐

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

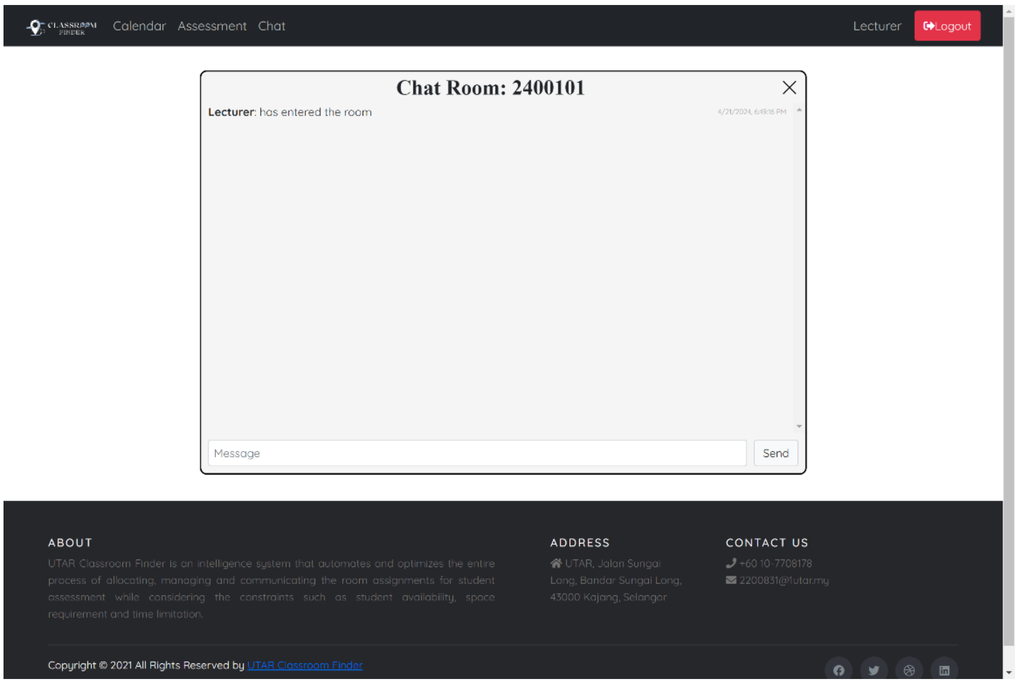
CONTACT US

+60 10-7708178

2200831@1utar.my

Figure 5.90: Notify Assessment User Page (Lecturer)

5.4.41 Online Chat Page (Lecturer)



Chat Room: 2400101

Lecturer: has entered the room 4/21/2024, 6:49:16 PM

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

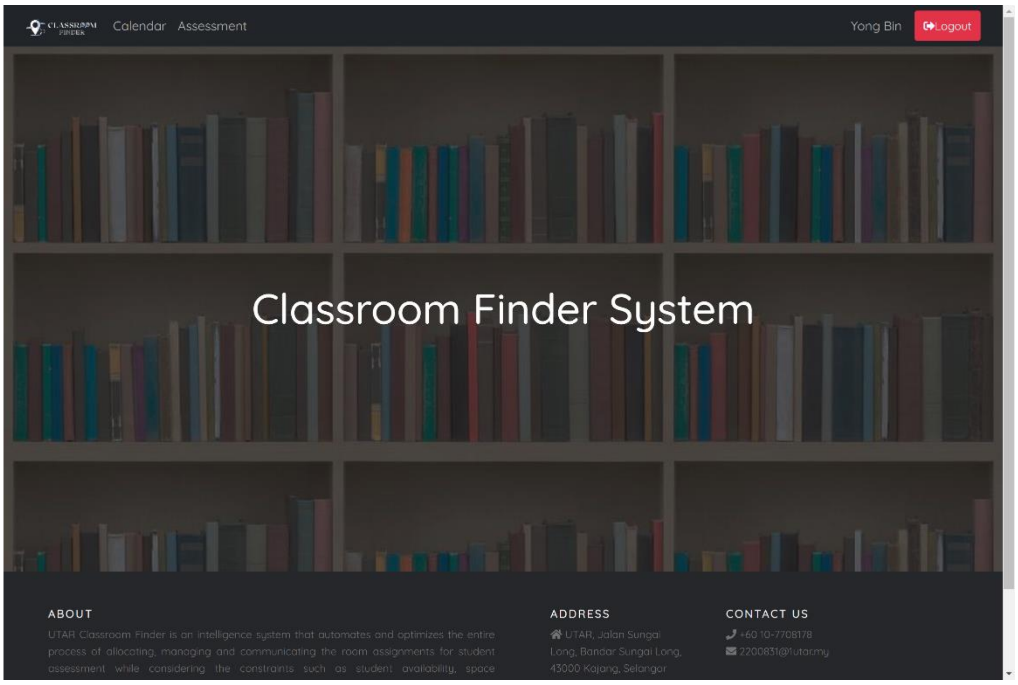
+60 10-7708178

2200831@1utar.my

Copyright © 2021 All Rights Reserved by UTAR Classroom Finder

Figure 5.91: Online Chat Page (Lecturer)

5.4.42 Home Page (Student)



ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space

ADDRESS

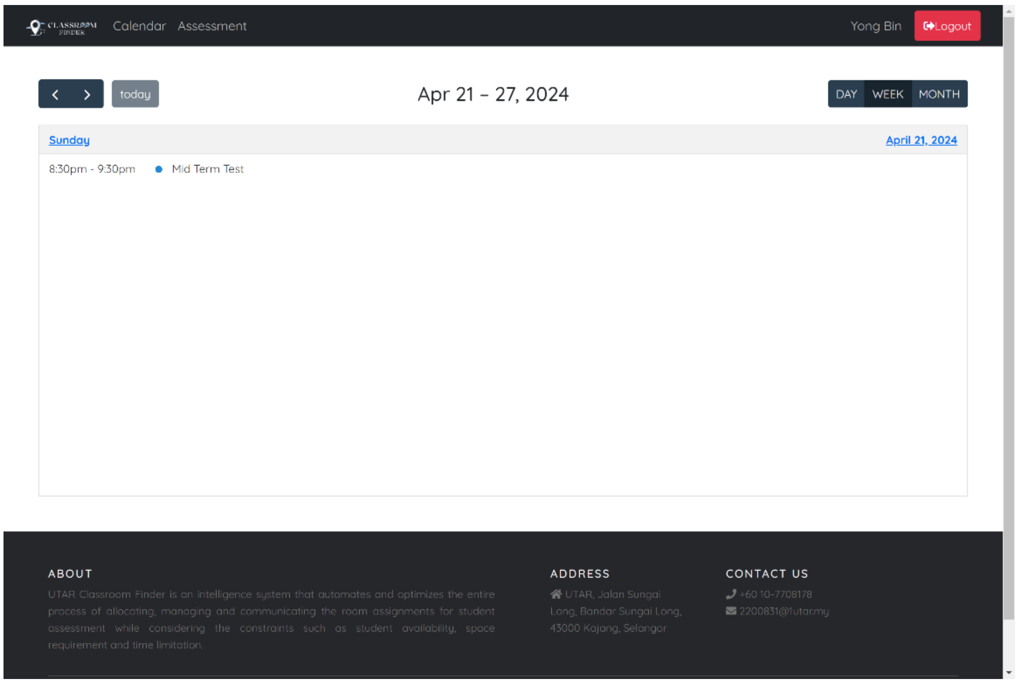
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.92: Home Page (Student)

5.4.43 View Calendar Page (Student)



8:30pm - 9:30pm ● Mid Term Test

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

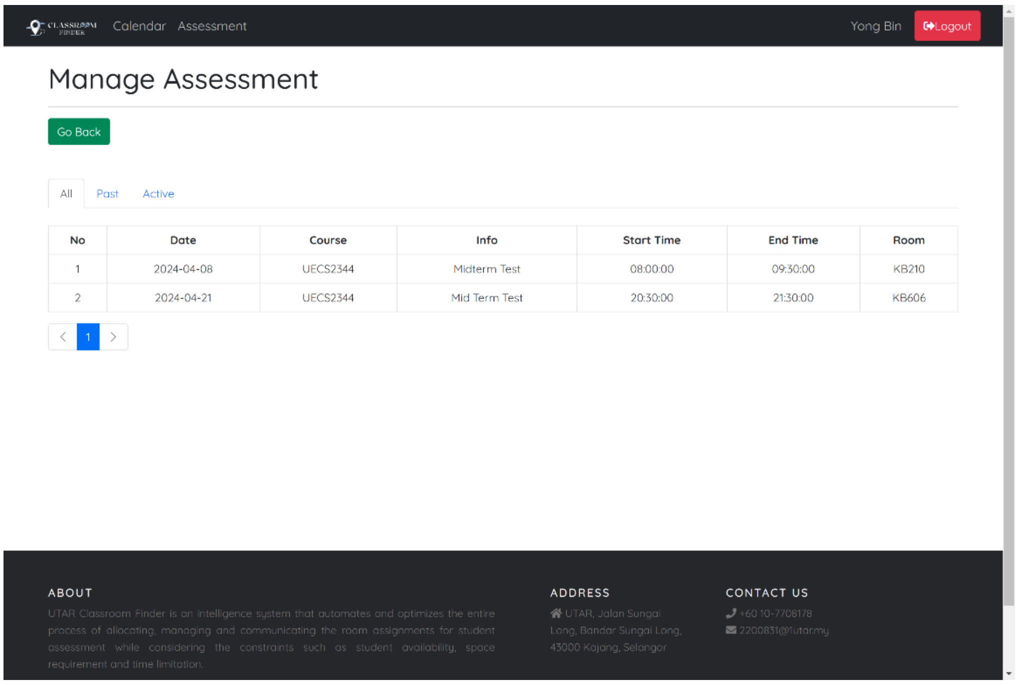
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.93: View Calendar Page (Student)

5.4.44 View Assessment Page (Student)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

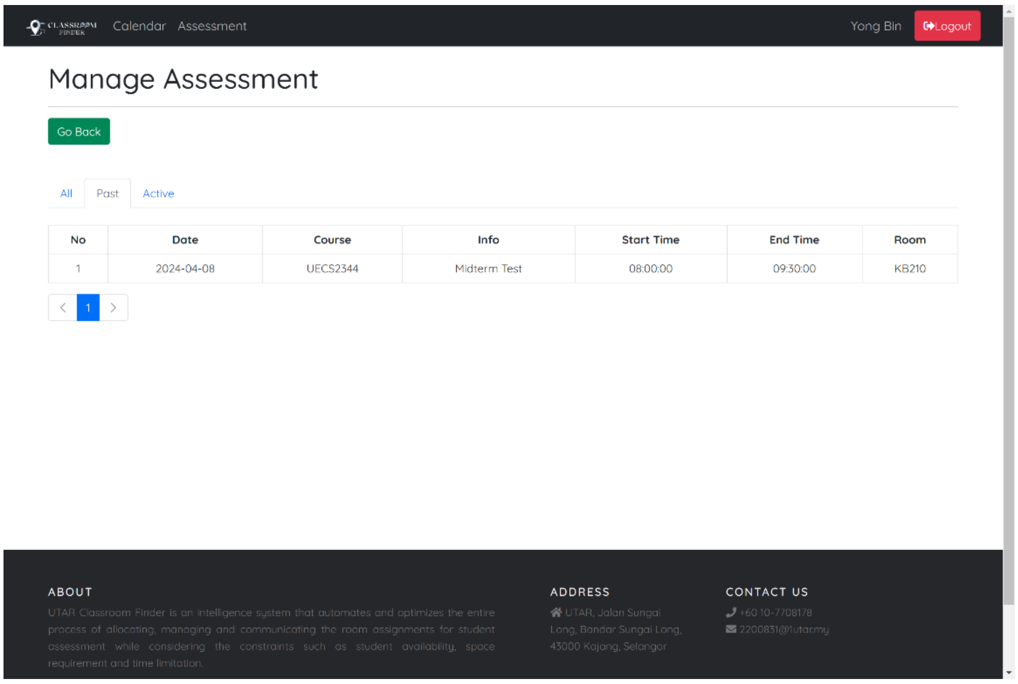
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.94: View Assessment Page (Student)

5.4.45 View Past Assessment Page (Student)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

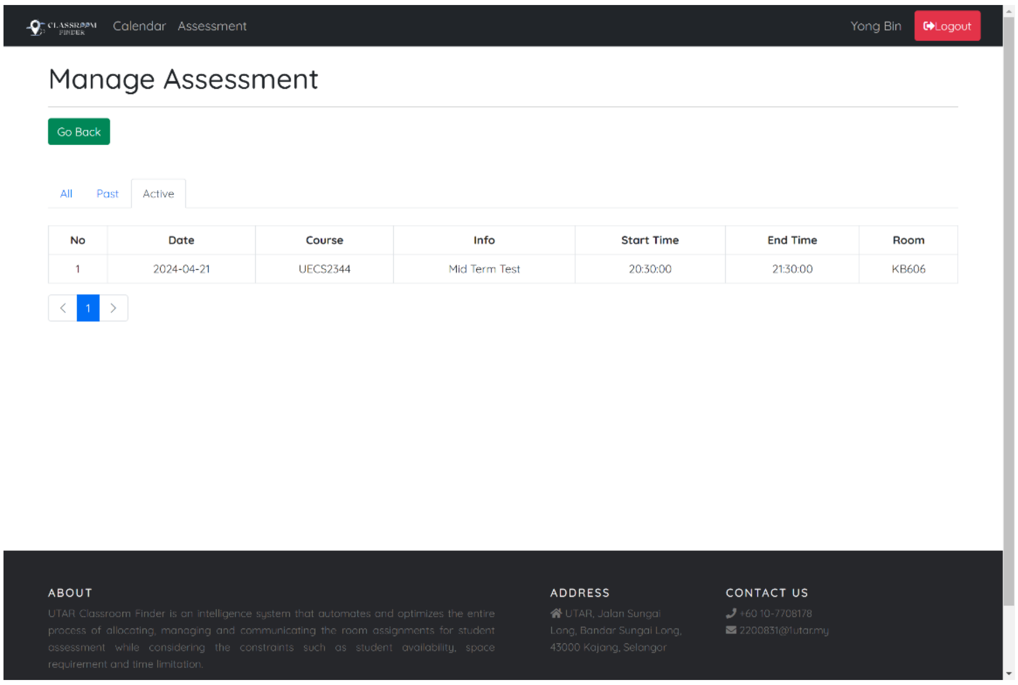
UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.95: View Past Assessment Page (Student)

5.4.46 View Active Assessment Page (Student)



Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room
1	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606

ABOUT

UTAR Classroom Finder is an intelligence system that automates and optimizes the entire process of allocating, managing and communicating the room assignments for student assessment while considering the constraints such as student availability, space requirement and time limitation.

ADDRESS

UTAR, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor

CONTACT US

+60 10-7708178
2200831@1utar.my

Figure 5.96: View Active Assessment Page (Student)

CHAPTER 6

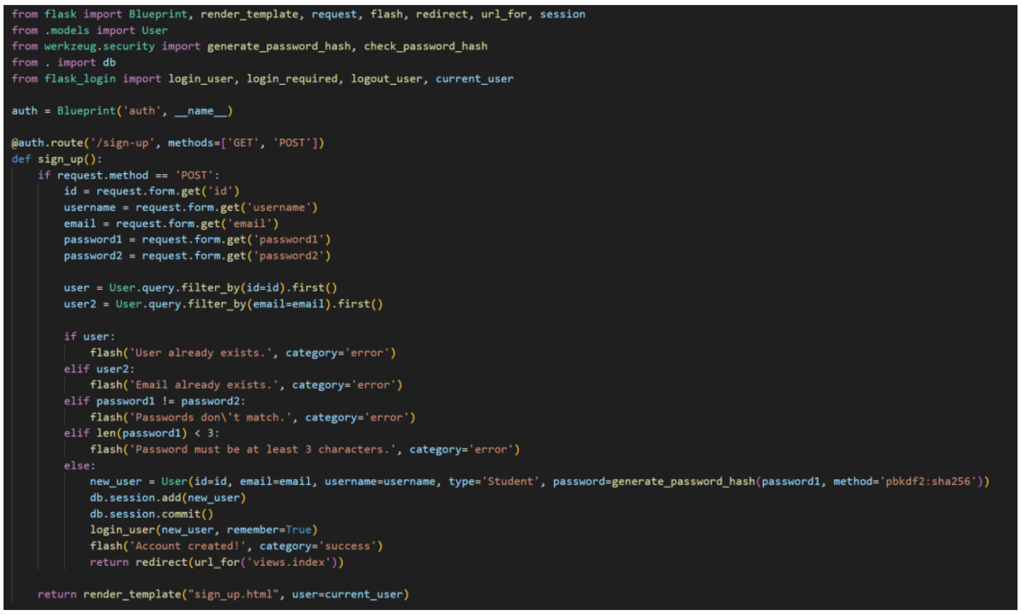
SYSTEM IMPLEMENTATION

6.1 Introduction

The implementation phase of the Classroom Finder System represents a significant milestone in the development journey where theoretical concepts and design specifications are translated into tangible software components. In this chapter, it embarks on a comprehensive exploration of the system implementation which structured around the modular architecture comprising various functional modules. Each module represents a distinct aspect of the system's functionality and serves a specific purpose in facilitating seamless interaction and management within the application. From user authentication and management to course and assessment scheduling, each module encapsulates a unique set of features designed to enhance the overall usability, efficiency and reliability of the Classroom Finder System.

The modular approach adopted in the implementation process allows for the systematic development and integration of individual components which can help to promote the flexibility, scalability and maintainability. By breaking down the system into discrete modules, focus on addressing the specific requirements and functionalities can ensure the clarity, modularity and ease of management throughout the development lifecycle. Throughout this chapter, it delves into each module implementation details which highlights the technologies, tools and methodologies employed to realize the system functionalities effectively. In this project, the key modules that constitute the Classroom Finder System are registration module, login and logout module, lecturer management module, student management module, room management module, course management module, course timetable management module, course enrolment management module, assessment management module, calendar module and online chat or communication module.

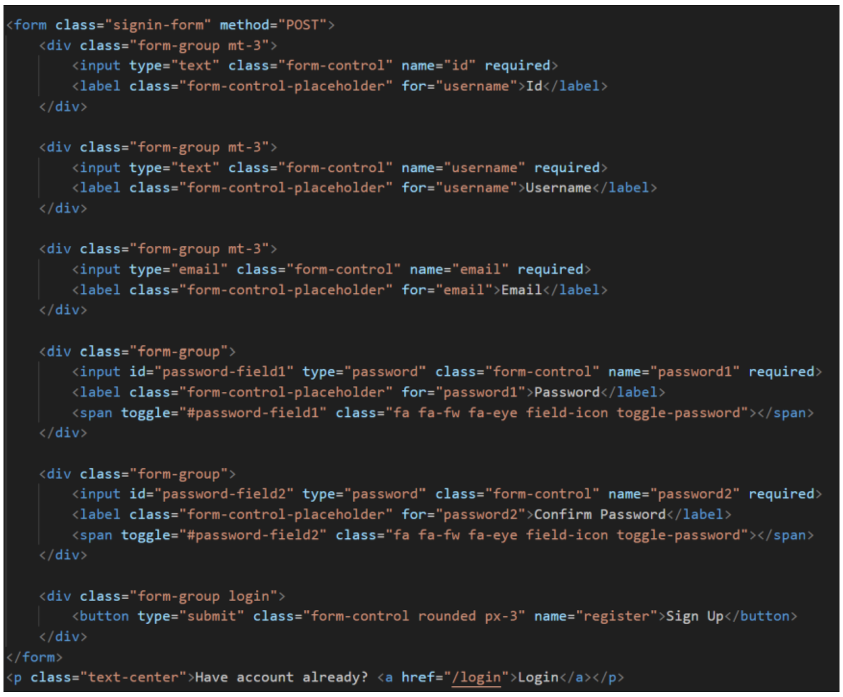
6.2 Registration Module



```python
from flask import Blueprint, render_template, request, flash, redirect, url_for, session
from .models import User
from werkzeug.security import generate_password_hash, check_password_hash
from . import db
from flask_login import login_user, login_required, logout_user, current_user

auth = Blueprint('auth', __name__)

@auth.route('/sign-up', methods=['GET', 'POST'])
def sign_up():
    if request.method == 'POST':
        id = request.form.get('id')
        username = request.form.get('username')
        email = request.form.get('email')
        password1 = request.form.get('password1')
        password2 = request.form.get('password2')

        user = User.query.filter_by(id=id).first()
        user2 = User.query.filter_by(email=email).first()

        if user:
            flash('User already exists.', category='error')
        elif user2:
            flash('Email already exists.', category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 3:
            flash('Password must be at least 3 characters.', category='error')
        else:
            new_user = User(id=id, email=email, username=username, type='Student', password=generate_password_hash(password1, method='pbkdf2:sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Account created!', category='success')
            return redirect(url_for('views.index'))

    return render_template("sign_up.html", user=current_user)
```

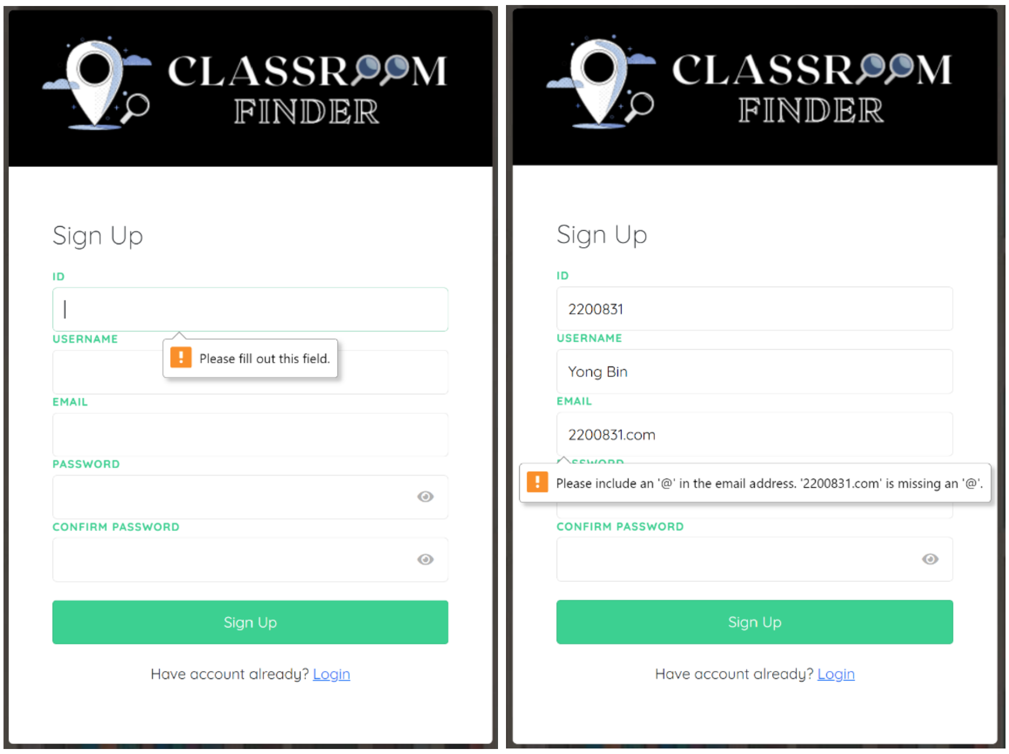
Figure 6.1: Implementation Code for Registration Module

The registration module plays a crucial role in the Classroom Finder System by facilitating the creation of new student accounts. Based on figure 6.1, it begins by encapsulating its functionalities within a Flask Blueprint named 'auth' which promotes the modularity and organization within the application's codebase. Within this module, the primary routes are defined to handle user registration which is '/sign-up'.



```html
<form class="signin-form" method="POST">
    <div class="form-group mt-3">
        <input type="text" class="form-control" name="id" required>
        <label class="form-control-placeholder" for="username">Id</label>
    </div>

    <div class="form-group mt-3">
        <input type="text" class="form-control" name="username" required>
        <label class="form-control-placeholder" for="username">Username</label>
    </div>

    <div class="form-group mt-3">
        <input type="email" class="form-control" name="email" required>
        <label class="form-control-placeholder" for="email">Email</label>
    </div>

    <div class="form-group">
        <input id="password-field1" type="password" class="form-control" name="password1" required>
        <label class="form-control-placeholder" for="password1">Password</label>
        <span toggle="#password-field1" class="fa fa-fw fa-eye field-icon toggle-password"></span>
    </div>

    <div class="form-group">
        <input id="password-field2" type="password" class="form-control" name="password2" required>
        <label class="form-control-placeholder" for="password2">Confirm Password</label>
        <span toggle="#password-field2" class="fa fa-fw fa-eye field-icon toggle-password"></span>
    </div>

    <div class="form-group login">
        <button type="submit" class="form-control rounded px-3" name="register">Sign Up</button>
    </div>
</form>
<p class="text-center">Have account already? <a href="/login">Login</a></p>
```

Figure 6.2: Registration Form with Validation



Please fill out this field.

ID: 2200831

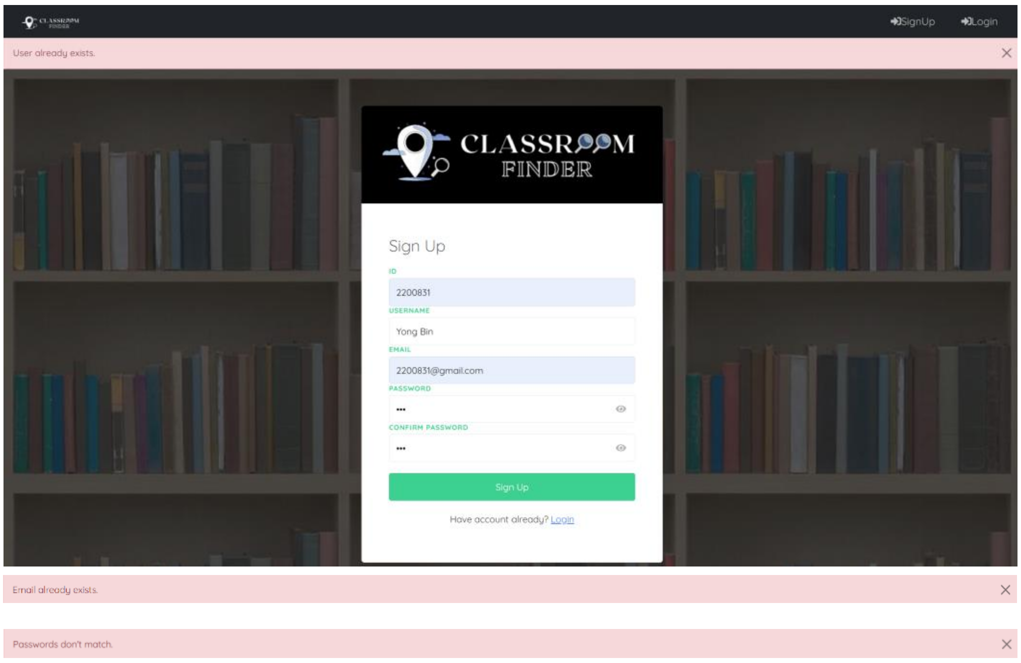
USERNAME: Yong Bin

EMAIL: 2200831.com

Please include an '@' in the email address. '2200831.com' is missing an '@'.

Figure 6.3: Empty & Wrong Input Format Value

The '/sign-up' route serves as the entry point for user registration which allows the students to submit their registration details via a HTML form. Before submitting their registration details of id, username, email, password and confirm password, all the input values are required to be filled in to perform the registration process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.


SignUp Login

User already exists.

Sign Up

ID
2200831

USERNAME
Yong Bin

EMAIL
2200831@gmail.com

PASSWORD
•••

CONFIRM PASSWORD
•••

Sign Up

Have account already? Login

Email already exists.

Passwords don't match.

Figure 6.4: Registration Failed with Error Messages

Upon receiving a POST request containing the user's registration data, the route's function processes this data using the Flask request object's 'request.form' attribute. The function then proceeds to validate the submitted data to ensure its integrity and compliance with predefined rules. The key validation checks are verifying the uniqueness of user IDs and email addresses, ensuring password match and enforcing minimum password length requirements. Throughout the registration process, the system provides real-time feedback to users via Flash messages which categorize messages as either 'success' or 'error' based on the outcome of the registration attempt. These messages inform users of the success or failure of their registration attempt and provide actionable insights into any issues encountered.

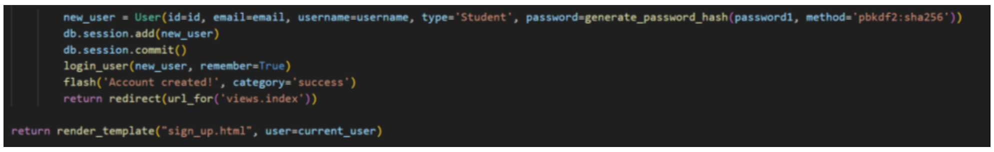
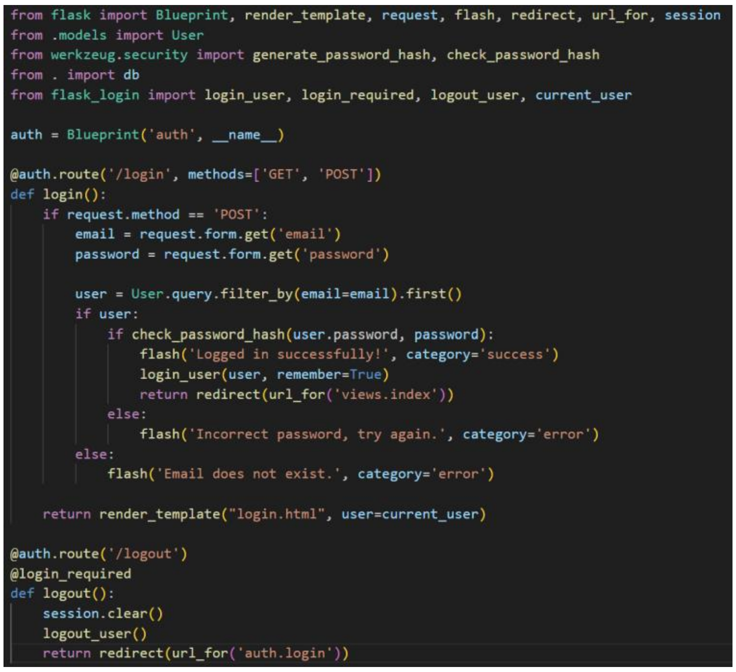

```
        new_user = User(id=id, email=email, username=username, type='Student', password=generate_password_hash(password1, method='pbkdf2:sha256'))
        db.session.add(new_user)
        db.session.commit()
        login_user(new_user, remember=True)
        flash('Account created!', category='success')
        return redirect(url_for('views.index'))

return render_template("sign_up.html", user=current_user)
```

Figure 6.5: Implementation Code for Password Hashing Technique

Security is also paramount during the user registration while the module employs robust security practices to protect user data. The passwords are securely hashed using the Werkzeug library's 'generate_password_hash' function before being stored in the database. This is to ensure that the sensitive information remains protected. During login, passwords are securely compared against their hashed counterparts using the 'check_password_hash' function to authenticate users securely. Upon successful registration, the newly created user is automatically logged in using Flask-Login's 'login_user' function, which associates the user's session with their account. This seamless login process enhances user experience and streamlines access to protected resources within the system.

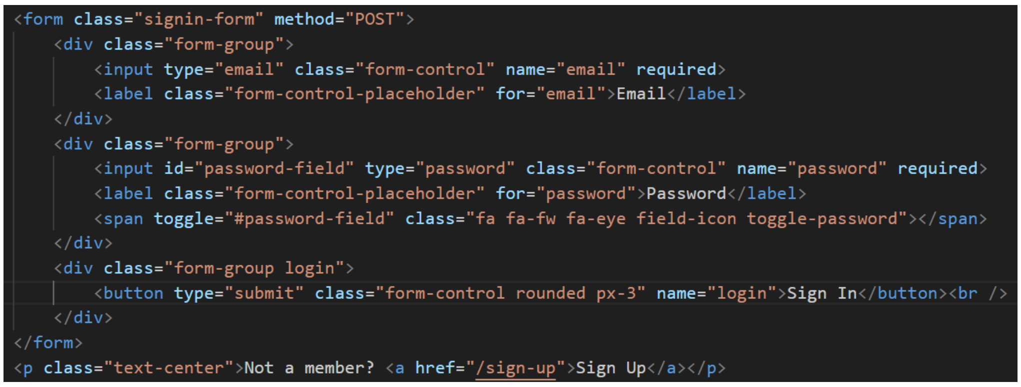
6.2 Login and Logout Module



```python
from flask import Blueprint, render_template, request, flash, redirect, url_for, session
from .models import User
from werkzeug.security import generate_password_hash, check_password_hash
from . import db
from flask_login import login_user, login_required, logout_user, current_user

auth = Blueprint('auth', __name__)

@auth.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        user = User.query.filter_by(email=email).first()
        if user:
            if check_password_hash(user.password, password):
                flash('Logged in successfully!', category='success')
                login_user(user, remember=True)
                return redirect(url_for('views.index'))
            else:
                flash('Incorrect password, try again.', category='error')
        else:
            flash('Email does not exist.', category='error')

    return render_template("login.html", user=current_user)

@auth.route('/logout')
@login_required
def logout():
    session.clear()
    logout_user()
    return redirect(url_for('auth.login'))
```

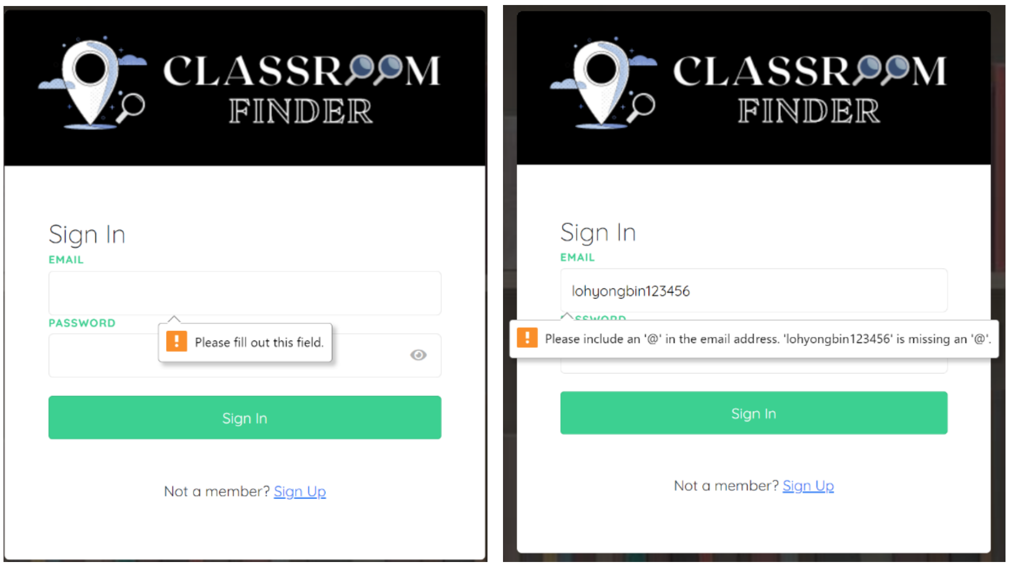
Figure 6.6: Implementation Code for Login and Logout Module

The login and logout modules are integral components of the authentication system within the Classroom Finder System which responsible for managing user authentication sessions and ensuring secure access to the application. These modules are implemented using Flask Blueprints which allow for modular and organized structuring of the application's routes and functionalities.



```html
<form class="signin-form" method="POST">
    <div class="form-group">
        <input type="email" class="form-control" name="email" required>
        <label class="form-control-placeholder" for="email">Email</label>
    </div>
    <div class="form-group">
        <input id="password-field" type="password" class="form-control" name="password" required>
        <label class="form-control-placeholder" for="password">Password</label>
        <span toggle="#password-field" class="fa fa-fw fa-eye field-icon toggle-password"></span>
    </div>
    <div class="form-group login">
        <button type="submit" class="form-control rounded px-3" name="login">Sign In</button><br />
    </div>
</form>
<p class="text-center">Not a member? <a href="/sign-up">Sign Up</a></p>
```

Figure 6.7: Login Form with Validation



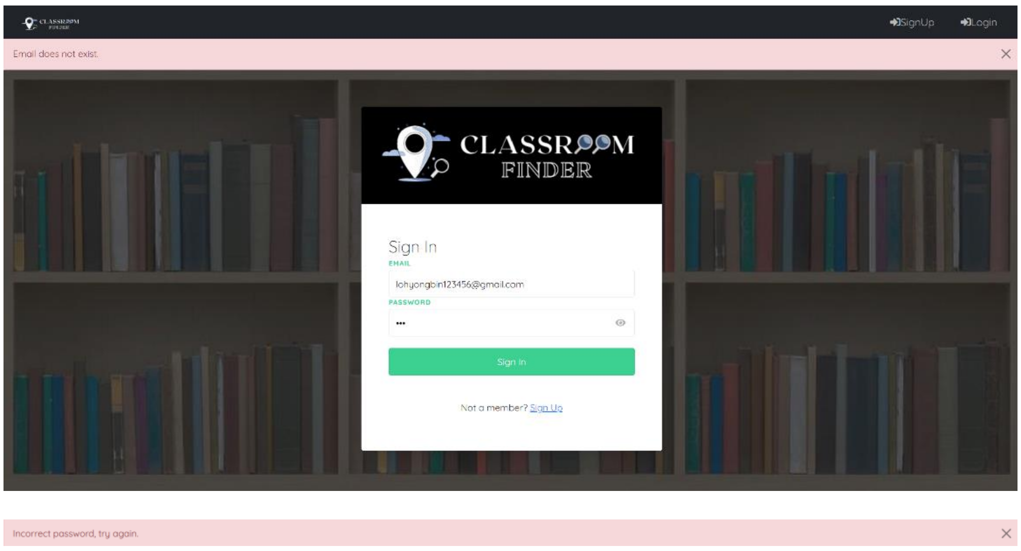
Please fill out this field.

lohyongbin123456

Please include an '@' in the email address. 'lohyongbin123456' is missing an '@'.

Figure 6.8: Empty & Wrong Input Format Value

The login module which accessible via the '/login' route, provides users with the means to authenticate themselves and access the system. It also allows the user to submit their login details via a HTML form. Before submitting their login details of email and password, all the input values are required to be filled in to perform the registration process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



Email does not exist.

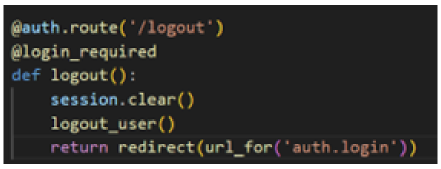
lohyongbin123456@gmail.com

•••

Incorrect password, try again.

Figure 6.9: Login Failed with Error Messages

Upon receiving a POST request containing user credentials which include the email and password from the login form, the login route's function retrieves the corresponding user record from the database using Flask-SQLAlchemy's query capabilities. If a user with the provided email exists, the function proceeds to verify the submitted password against the hashed password stored in the user's record using the Werkzeug library's 'check_password_hash' function. If the password matches, it is indicating successful authentication, then the user will be logged in using Flask-Login's 'login_user' function. A Flash message is then displayed to inform the user of their successful login and they are redirected to the system's main page. If the provided credentials are invalid or the user does not exist, the appropriate Flash messages are generated to notify the user of the encountered error which can be shown as figure 6.9 above.



```
@auth.route('/logout')
@login_required
def logout():
    session.clear()
    logout_user()
    return redirect(url_for('auth.login'))
```

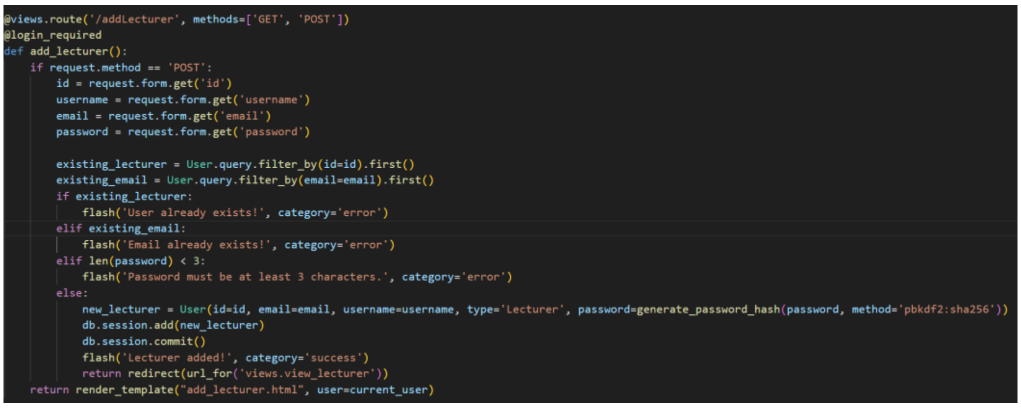
Figure 6.10: Implementation Code for Logout Module

Conversely, the logout module which accessible via the '/logout' route, facilitates the termination of a user's session and the subsequent logout from the system. This module is protected by the 'login_required' decorator provided by Flask-Login to ensure that only authenticated users can access it. Upon accessing the logout route, the user's session is cleared and the users are logged out using Flask-Login's 'logout_user' function. Subsequently, the user is redirected to the login page to reauthenticate if necessary. This process effectively terminates the user's session and ensures the security of their account.

6.3 Lecture Management Module

The lecturer management module is an essential component of the Classroom Finder System which responsible for facilitating the creation, viewing, modification, and deletion of lecturer accounts within the application. This module comprises several routes and functions implemented using the Flask web framework and SQLAlchemy ORM.

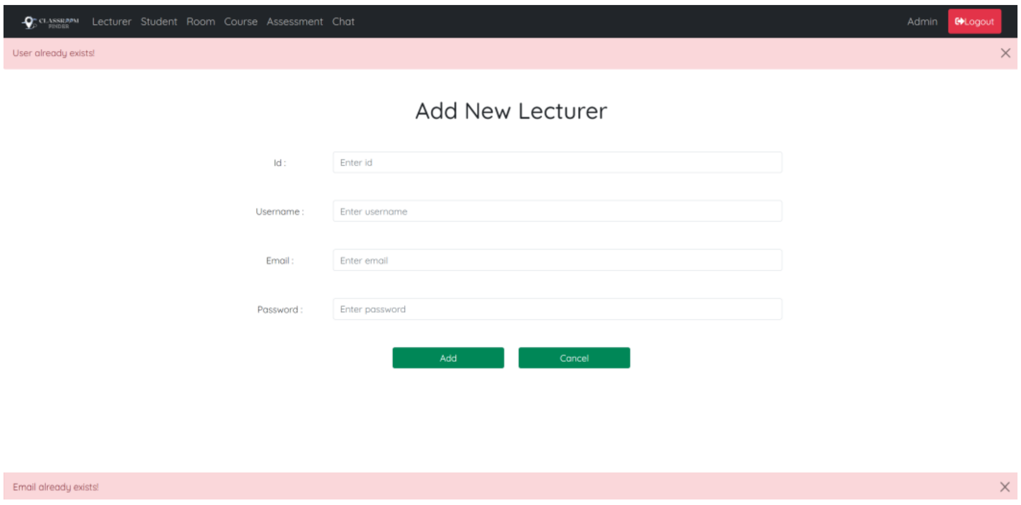
6.3.1 Add Lecturer



```python
@views.route('/addLecturer', methods=['GET', 'POST'])
@login_required
def add_lecturer():
    if request.method == 'POST':
        id = request.form.get('id')
        username = request.form.get('username')
        email = request.form.get('email')
        password = request.form.get('password')

        existing_lecturer = User.query.filter_by(id=id).first()
        existing_email = User.query.filter_by(email=email).first()
        if existing_lecturer:
            flash('User already exists!', category='error')
        elif existing_email:
            flash('Email already exists!', category='error')
        elif len(password) < 3:
            flash('Password must be at least 3 characters.', category='error')
        else:
            new_lecturer = User(id=id, email=email, username=username, type='Lecturer', password=generate_password_hash(password, method='pbkdf2:sha256'))
            db.session.add(new_lecturer)
            db.session.commit()
            flash('Lecturer added!', category='success')
            return redirect(url_for('views.view_lecturer'))
    return render_template("add_lecturer.html", user=current_user)
```

Figure 6.11: Implementation Code for Adding Lecturer

The 'add_lecturer' function which available at the '/addLecturer' route, facilitates the addition of new lecturer accounts to the system. It allows the administrator to submit the lecturer details via a HTML form. Before submitting the lecturer details of id, username, email and password, all the input values are required to be filled in to perform the lecturer creation process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



User already exists!

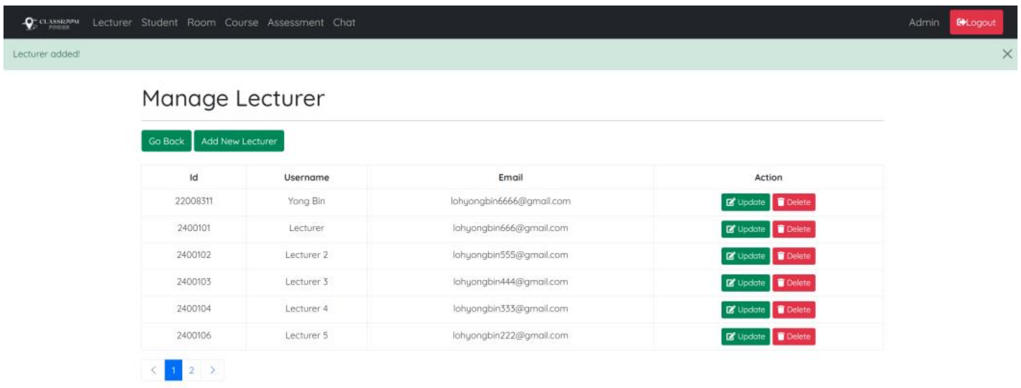
Add New Lecturer

Id :	Enter id
Username :	Enter username
Email :	Enter email
Password :	Enter password

Email already exists!

Figure 6.12: Creation Failed with Error Messages

Upon receiving a POST request containing the necessary information which are id, username, email, and password from the add lecturer form, the function validates the provided data and checks for any existing user or email conflicts.



Lecturer added!

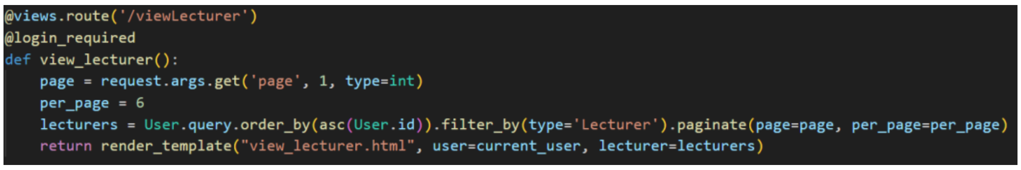
Manage Lecturer

Id	Username	Email	Action
22008311	Yong Bin	lohyongbin6666@gmail.com	Update Delete
2400101	Lecturer	lohyongbin666@gmail.com	Update Delete
2400102	Lecturer 2	lohyongbin555@gmail.com	Update Delete
2400103	Lecturer 3	lohyongbin444@gmail.com	Update Delete
2400104	Lecturer 4	lohyongbin333@gmail.com	Update Delete
2400106	Lecturer 5	lohyongbin222@gmail.com	Update Delete

Figure 6.13: Creation Successfully with Success Messages

If validation passes, a new lecturer record is created in the database and the user is redirected to the view lecturer page with a success message.

6.3.2 View Lecturer

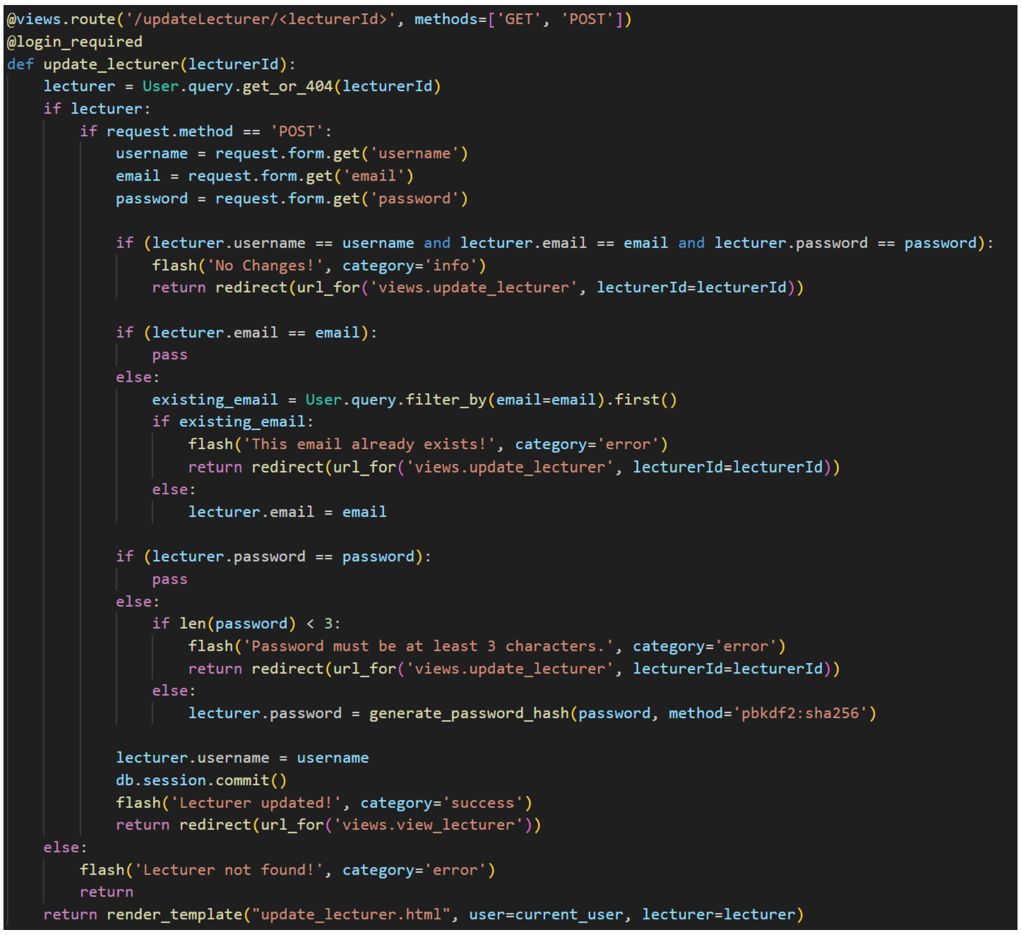


```
@views.route('/viewLecturer')
@login_required
def view_lecturer():
    page = request.args.get('page', 1, type=int)
    per_page = 6
    lecturers = User.query.order_by(asc(User.id)).filter_by(type='Lecturer').paginate(page=page, per_page=per_page)
    return render_template("view_lecturer.html", user=current_user, lecturer=lecturers)
```

Figure 6.14: Implementation Code for Viewing Lecturer

The 'view_lecturer' function which accessible via the '/viewLecturer' route, enables administrator with appropriate privileges to view a paginated list of existing lecturer accounts. The function retrieves the lecturer records from the database, orders them by their IDs and paginates the results to enhance usability. The administrator can navigate through the paginated list to access details about each lecturer such as their username and email address.

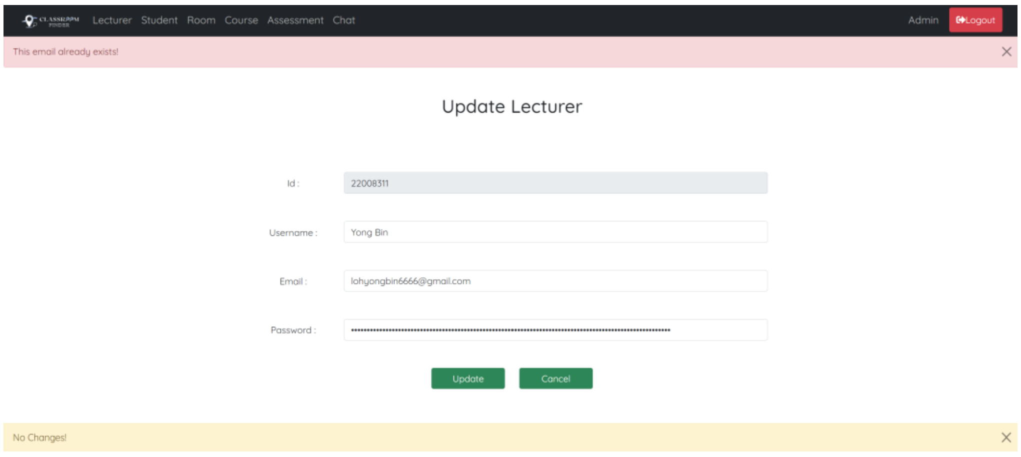
6.3.3 Update Lecturer



```python
@views.route('/updateLecturer/<lecturerId>', methods=['GET', 'POST'])
@login_required
def update_lecturer(lecturerId):
    lecturer = User.query.get_or_404(lecturerId)
    if lecturer:
        if request.method == 'POST':
            username = request.form.get('username')
            email = request.form.get('email')
            password = request.form.get('password')

            if (lecturer.username == username and lecturer.email == email and lecturer.password == password):
                flash('No Changes!', category='info')
                return redirect(url_for('views.update_lecturer', lecturerId=lecturerId))

            if (lecturer.email == email):
                pass
            else:
                existing_email = User.query.filter_by(email=email).first()
                if existing_email:
                    flash('This email already exists!', category='error')
                    return redirect(url_for('views.update_lecturer', lecturerId=lecturerId))
                else:
                    lecturer.email = email

            if (lecturer.password == password):
                pass
            else:
                if len(password) < 3:
                    flash('Password must be at least 3 characters.', category='error')
                    return redirect(url_for('views.update_lecturer', lecturerId=lecturerId))
                else:
                    lecturer.password = generate_password_hash(password, method='pbkdf2:sha256')

            lecturer.username = username
            db.session.commit()
            flash('Lecturer updated!', category='success')
            return redirect(url_for('views.view_lecturer'))
    else:
        flash('Lecturer not found!', category='error')
        return
    return render_template("update_lecturer.html", user=current_user, lecturer=lecturer)
```

Figure 6.15: Implementation Code for Updating Lecturer

The 'update_lecturer' function which accessible via the '/updateLecturer/<lecturerId>' route, allows administrators to update existing lecturer account details. The administrators can access the update lecturer form for a specific lecturer by providing their Id in the route. The administrator can update the lecturer by submitting the lecturer details via the HTML form. Before submitting the lecturer details of username, email and password, all the input values are required to be filled in to perform the lecturer updating process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



This email already exists!

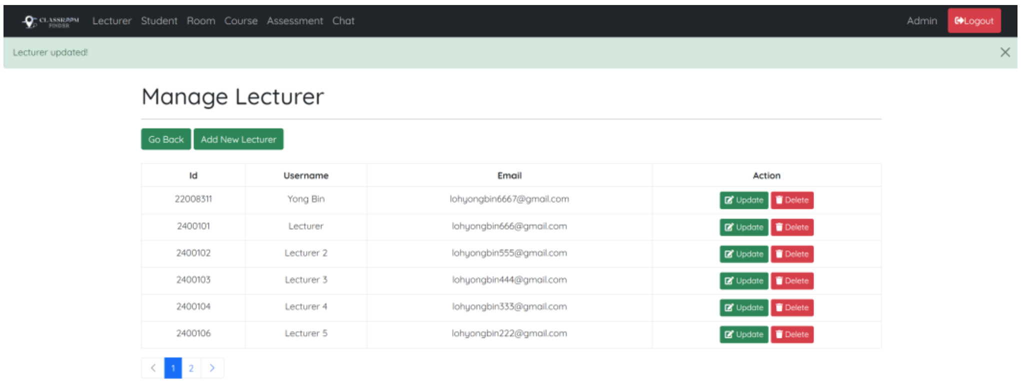
Update Lecturer

Id :	22008311
Username :	Yong Bin
Email :	lohyongbin6666@gmail.com
Password :	••

No Changes!

Figure 6.16: Update Failed with Error Messages

Upon receiving a POST request containing the necessary information which are username, email, and password from the update lecturer form, the function retrieves the lecturer record associated with the provided Id and validates the submitted changes. It will validate and check for the email conflicts. However, if there is not any changes, the administrator will be notify no changes for the updated data.



Lecturer updated!

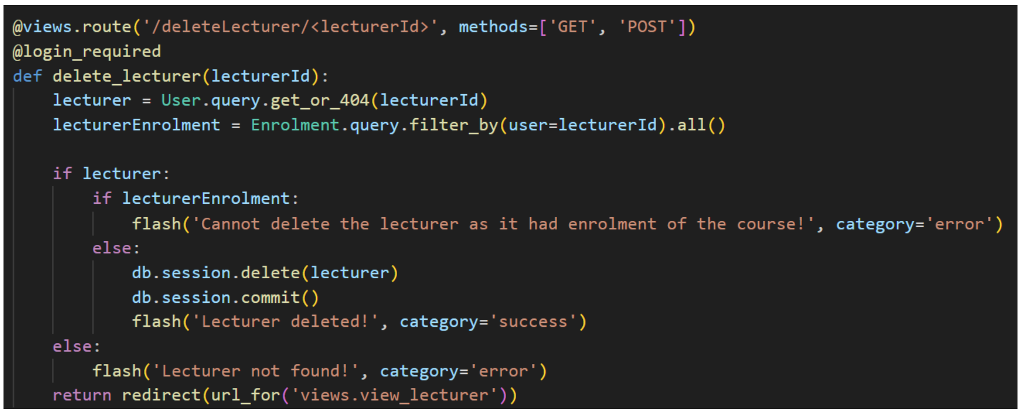
Manage Lecturer

Id	Username	Email	Action
22008311	Yong Bin	lohyongbin6667@gmail.com	Update Delete
2400101	Lecturer	lohyongbin666@gmail.com	Update Delete
2400102	Lecturer 2	lohyongbin555@gmail.com	Update Delete
2400103	Lecturer 3	lohyongbin444@gmail.com	Update Delete
2400104	Lecturer 4	lohyongbin333@gmail.com	Update Delete
2400106	Lecturer 5	lohyongbin222@gmail.com	Update Delete

Figure 6.17: Update Successfully with Success Messages

If the submitted data is valid and differs from the existing record, the lecturer details are updated in the database and the administrator is redirected to the view leccturer page with a success message.

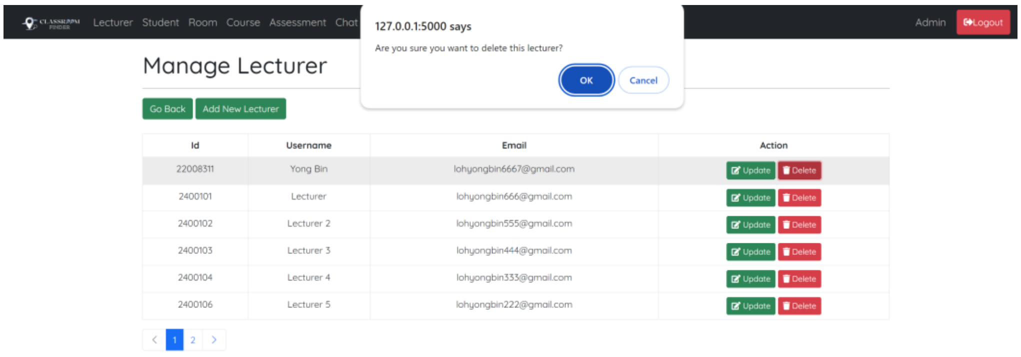
6.3.4 Delete Lecturer



```python
@views.route('/deleteLecturer/<lecturerId>', methods=['GET', 'POST'])
@login_required
def delete_lecturer(lecturerId):
    lecturer = User.query.get_or_404(lecturerId)
    lecturerEnrolment = Enrolment.query.filter_by(user=lecturerId).all()

    if lecturer:
        if lecturerEnrolment:
            flash('Cannot delete the lecturer as it had enrolment of the course!', category='error')
        else:
            db.session.delete(lecturer)
            db.session.commit()
            flash('Lecturer deleted!', category='success')
    else:
        flash('Lecturer not found!', category='error')
    return redirect(url_for('views.view_lecturer'))
```

Figure 6.18: Implementation Code for Deleting Lecturer

Finally, the 'delete_lecturer' function which available at the '/deleteLecturer/<lecturerId>' route, enables administrators to remove lecturer accounts from the system.



127.0.0.1:5000 says

Are you sure you want to delete this lecturer?

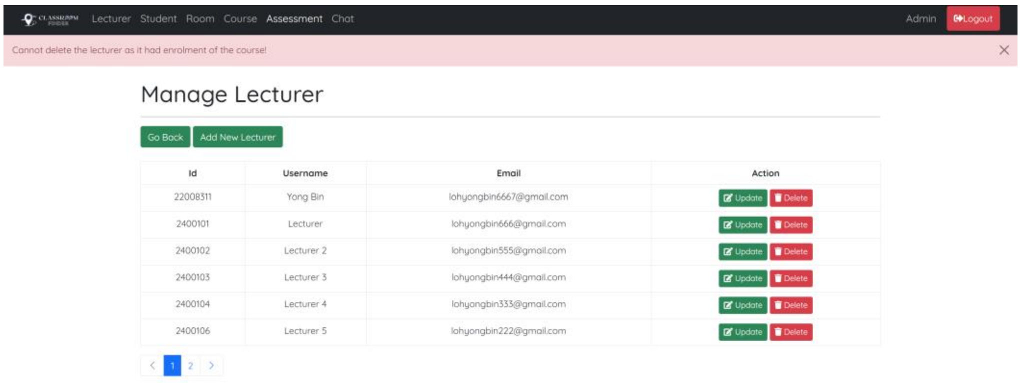
Manage Lecturer

Id	Username	Email	Action
22008311	Yong Bin	lohyongbin6667@gmail.com	Update Delete
2400101	Lecturer	lohyongbin666@gmail.com	Update Delete
2400102	Lecturer 2	lohyongbin555@gmail.com	Update Delete
2400103	Lecturer 3	lohyongbin444@gmail.com	Update Delete
2400104	Lecturer 4	lohyongbin333@gmail.com	Update Delete
2400106	Lecturer 5	lohyongbin222@gmail.com	Update Delete

< 1 2 >

Figure 6.19: Confirmation Prompt before Deleting Lecturer

Before accessing the delete lecturer route, the system will prompt the administrators with a confirmation dialog before proceeding with the deletion. This confirmation prompt serves as a safeguard, allowing users to verify their action and prevent accidental deletion of lecturer records. Once confirmed, the function checks for any associated enrolment records. If no enrolment records exist, the lecturer is deleted from the database and a success message is displayed.



Cannot delete the lecturer as it had enrolment of the course!

Manage Lecturer

Id	Username	Email	Action
22008311	Yong Bin	lohyongbin6667@gmail.com	Update Delete
2400101	Lecturer	lohyongbin666@gmail.com	Update Delete
2400102	Lecturer 2	lohyongbin555@gmail.com	Update Delete
2400103	Lecturer 3	lohyongbin444@gmail.com	Update Delete
2400104	Lecturer 4	lohyongbin333@gmail.com	Update Delete
2400106	Lecturer 5	lohyongbin222@gmail.com	Update Delete

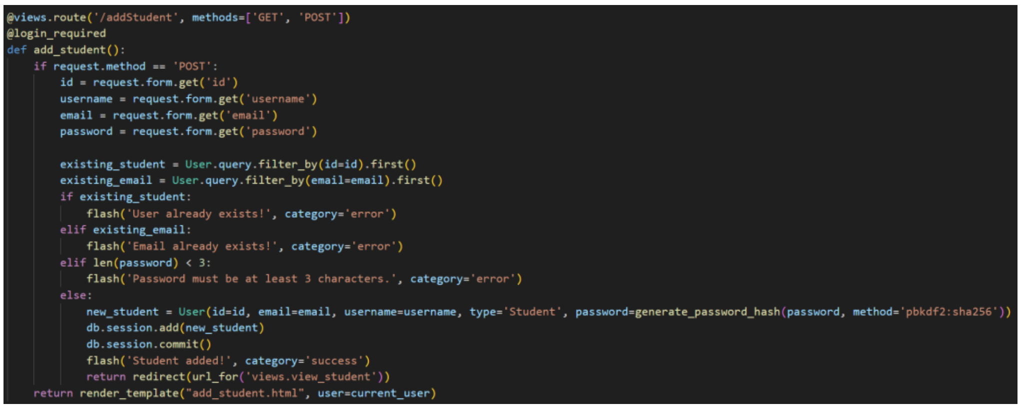
Figure 6.20: Delete Failed with Error Messages

Otherwise, an error message is shown to indicate that the lecturer cannot be deleted due to existing enrolment records.

6.4 Student Management Module

The student management module is an essential component of the Classroom Finder System which responsible for facilitating the creation, viewing, modification, and deletion of student accounts within the application. This module comprises several routes and functions implemented using the Flask web framework and SQLAlchemy ORM.

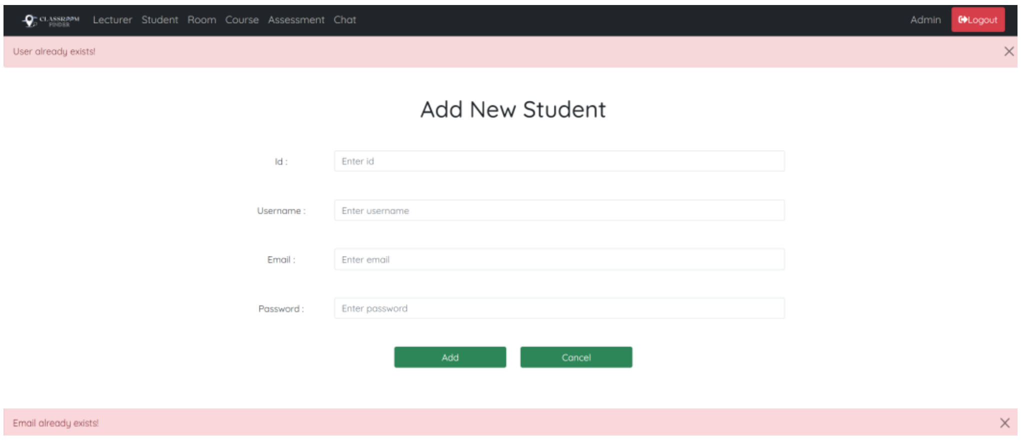
6.4.1 Add Student



```python
@views.route('/addStudent', methods=['GET', 'POST'])
@login_required
def add_student():
    if request.method == 'POST':
        id = request.form.get('id')
        username = request.form.get('username')
        email = request.form.get('email')
        password = request.form.get('password')

        existing_student = User.query.filter_by(id=id).first()
        existing_email = User.query.filter_by(email=email).first()
        if existing_student:
            flash('User already exists!', category='error')
        elif existing_email:
            flash('Email already exists!', category='error')
        elif len(password) < 3:
            flash('Password must be at least 3 characters.', category='error')
        else:
            new_student = User(id=id, email=email, username=username, type='Student', password=generate_password_hash(password, method='pbkdf2:sha256'))
            db.session.add(new_student)
            db.session.commit()
            flash('Student added!', category='success')
            return redirect(url_for('views.view_student'))
    return render_template("add_student.html", user=current_user)
```

Figure 6.21: Implementation Code for Adding Student

The 'add_student' function which available at the '/addStudent' route, facilitates the addition of new student accounts to the system. It allows the administrator to submit the student details via a HTML form. Before submitting the student details of id, username, email and password, all the input values are required to be filled in to perform the student creation process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



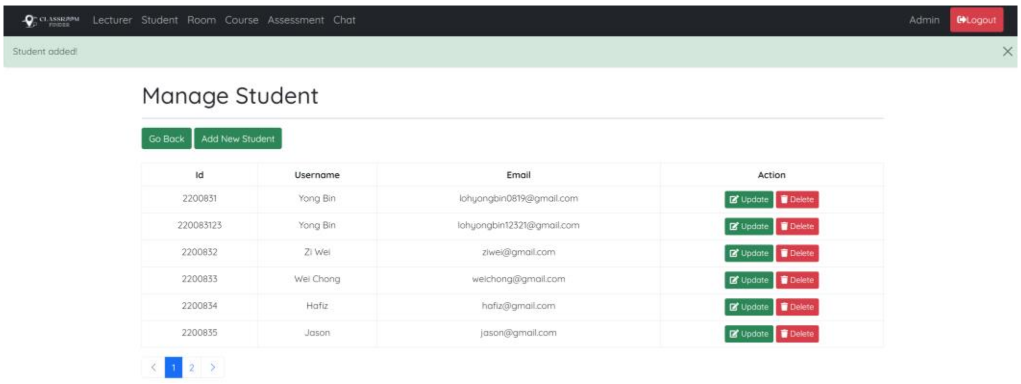
User already exists!

Add New Student

Email already exists!

Figure 6.22: Creation Failed with Error Messages

Upon receiving a POST request containing the necessary information which are id, username, email, and password from the add student form, the function validates the provided data and checks for any existing user or email conflicts.



Student added!

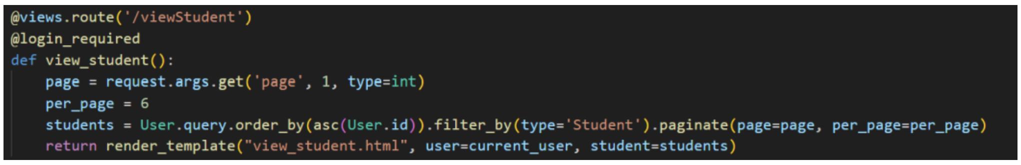
Manage Student

Id	Username	Email	Action
2200831	Yong Bin	lohyongbin0819@gmail.com	Update Delete
220083123	Yong Bin	lohyongbin12321@gmail.com	Update Delete
2200832	Zi Wei	ziwei@gmail.com	Update Delete
2200833	Wei Chong	weichong@gmail.com	Update Delete
2200834	Hafiz	hafiz@gmail.com	Update Delete
2200835	Jason	jason@gmail.com	Update Delete

Figure 6.23: Creation Successfully with Success Messages

If validation passes, a new student record is created in the database and the user is redirected to the view student page with a success message.

6.4.2 View Student

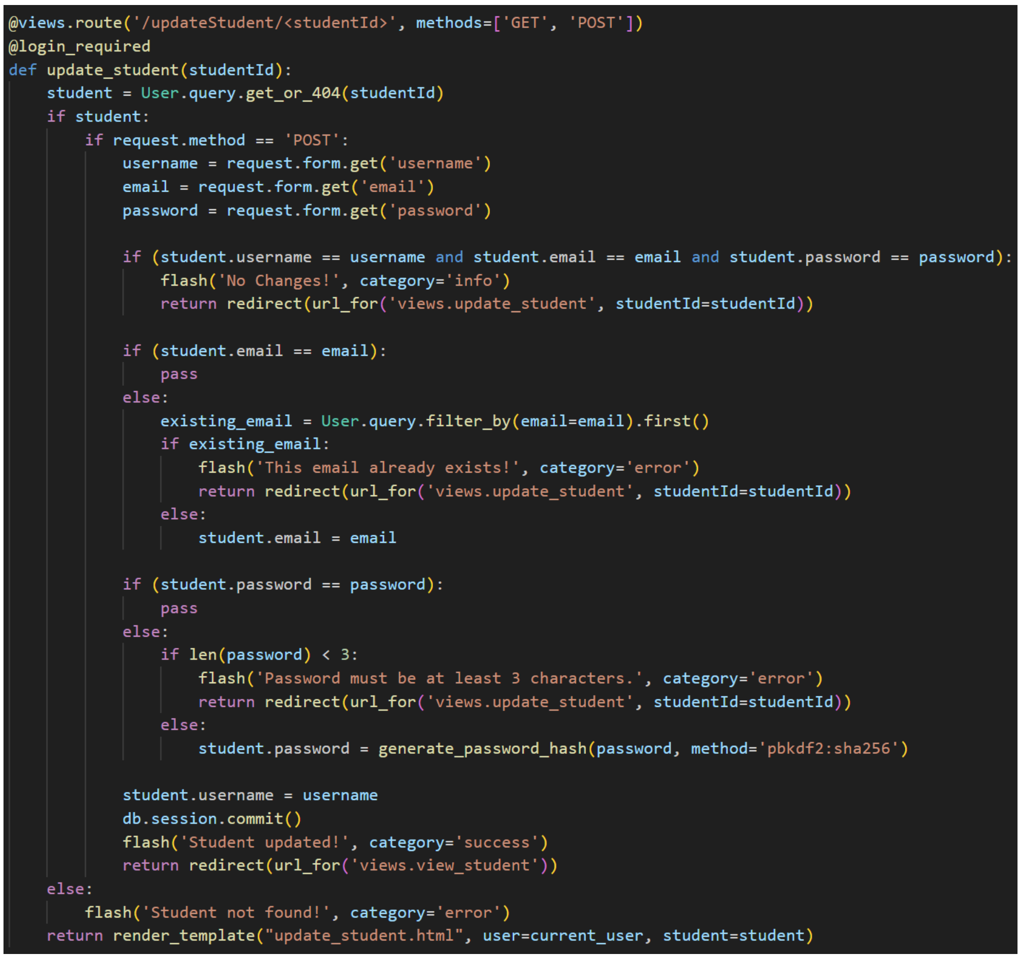


```
@views.route('/viewStudent')
@login_required
def view_student():
    page = request.args.get('page', 1, type=int)
    per_page = 6
    students = User.query.order_by(asc(User.id)).filter_by(type='Student').paginate(page=page, per_page=per_page)
    return render_template("view_student.html", user=current_user, student=students)
```

Figure 6.24: Implementation Code for Viewing Student

The 'view_student' function which accessible via the '/view Student' route, enables administrator with appropriate privileges to view a paginated list of existing student accounts. The function retrieves the student records from the database, orders them by their IDs and paginates the results to enhance usability. The administrator can navigate through the paginated list to access details about each student such as their username and email address.

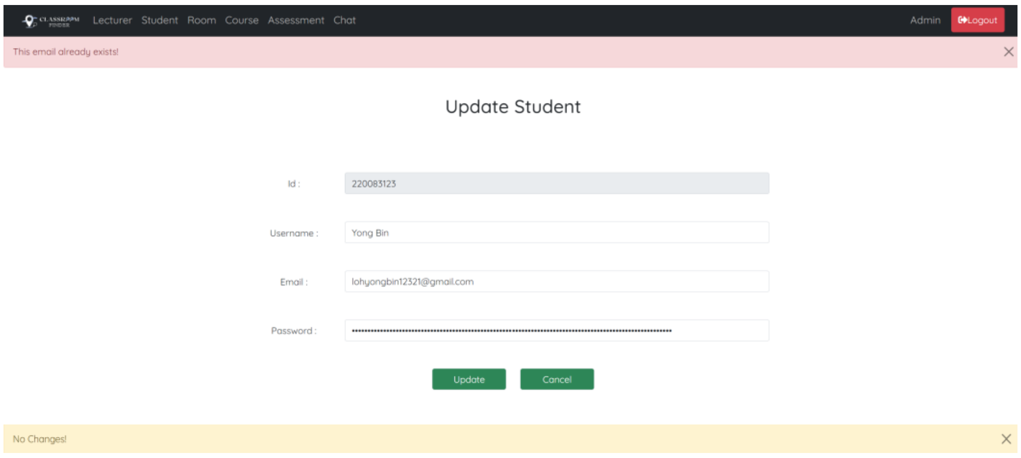
6.4.3 Update Student



```python
@views.route('/updateStudent/<studentId>', methods=['GET', 'POST'])
@login_required
def update_student(studentId):
    student = User.query.get_or_404(studentId)
    if student:
        if request.method == 'POST':
            username = request.form.get('username')
            email = request.form.get('email')
            password = request.form.get('password')

            if (student.username == username and student.email == email and student.password == password):
                flash('No Changes!', category='info')
                return redirect(url_for('views.update_student', studentId=studentId))

            if (student.email == email):
                pass
            else:
                existing_email = User.query.filter_by(email=email).first()
                if existing_email:
                    flash('This email already exists!', category='error')
                    return redirect(url_for('views.update_student', studentId=studentId))
                else:
                    student.email = email

            if (student.password == password):
                pass
            else:
                if len(password) < 3:
                    flash('Password must be at least 3 characters.', category='error')
                    return redirect(url_for('views.update_student', studentId=studentId))
                else:
                    student.password = generate_password_hash(password, method='pbkdf2:sha256')

            student.username = username
            db.session.commit()
            flash('Student updated!', category='success')
            return redirect(url_for('views.view_student'))
    else:
        flash('Student not found!', category='error')
    return render_template("update_student.html", user=current_user, student=student)
```

Figure 6.25: Implementation Code for Updating Student

The 'update_student' function which accessible via the '/updateStudent/<studentId>' route, allows administrators to update existing student account details. The administrators can access the update student form for a specific student by providing their Id in the route. The administrator can update the student by submitting the student details via the HTML form. Before submitting the student details of username, email and password, all the input values are required to be filled in to perform the student updating process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



This email already exists!

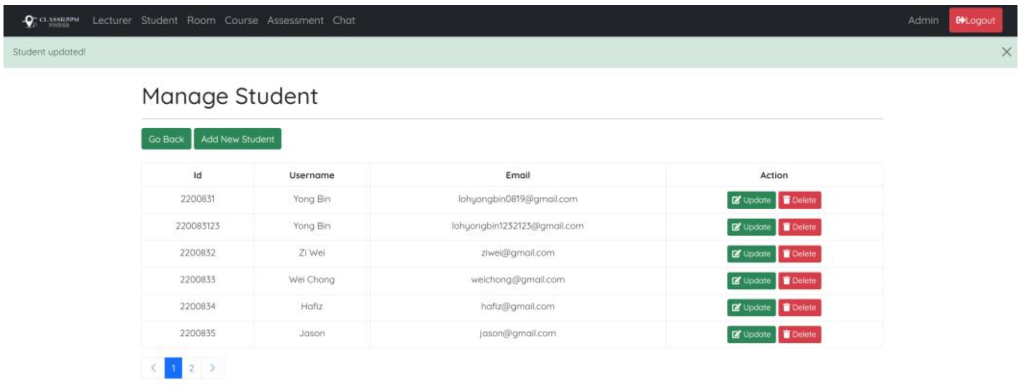
Update Student

Id :	220083123
Username :	Yong Bin
Email :	lohyongbin12321@gmail.com
Password :	••

No Changes!

Figure 6.26: Update Failed with Error Messages

Upon receiving a POST request containing the necessary information which are username, email, and password from the update student form, the function retrieves the student record associated with the provided Id and validates the submitted changes. It will validate and check for the email conflicts. However, if there are not any changes, the administrator will be notifying no changes for the updated data.



Student updated!

Manage Student

Id	Username	Email	Action
2200831	Yong Bin	lohyongbin0819@gmail.com	Update Delete
220083123	Yong Bin	lohyongbin1232123@gmail.com	Update Delete
2200832	Zi Wei	ziwei@gmail.com	Update Delete
2200833	Wei Chong	weichong@gmail.com	Update Delete
2200834	Hafiz	hafiz@gmail.com	Update Delete
2200835	Jason	jason@gmail.com	Update Delete

Figure 6.27: Update Successfully with Success Messages

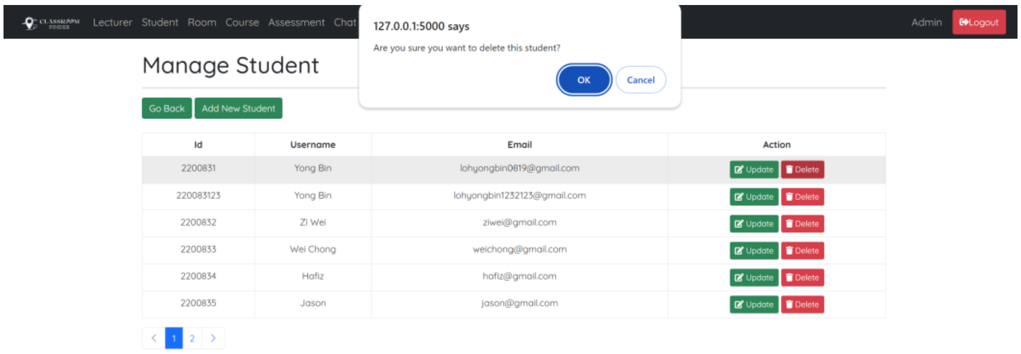
If the submitted data is valid and differs from the existing record, the student details are updated in the database and the administrator is redirected to the view student page with a success message.

6.4.4 Delete Student

```
@views.route('/deleteStudent/<studentId>', methods=['GET', 'POST'])
@login_required
def delete_student(studentId):
    student = User.query.get_or_404(studentId)
    studentEnrolment = Enrolment.query.filter_by(user=studentId).all()

    if student:
        if studentEnrolment:
            flash('Cannot delete the student as it had enrolment of the course!', category='error')
        else:
            db.session.delete(student)
            db.session.commit()
            flash('Student deleted!', category='success')
    else:
        flash('Student not found!', category='error')
    return redirect(url_for('views.view_student'))
```

Figure 6.28: Implementation Code for Deleting Student

Finally, the 'delete_student' function which available at the '/deleteStudent/< studentId>' route, enables administrators to remove student accounts from the system.



127.0.0.1:5000 says

Are you sure you want to delete this student?

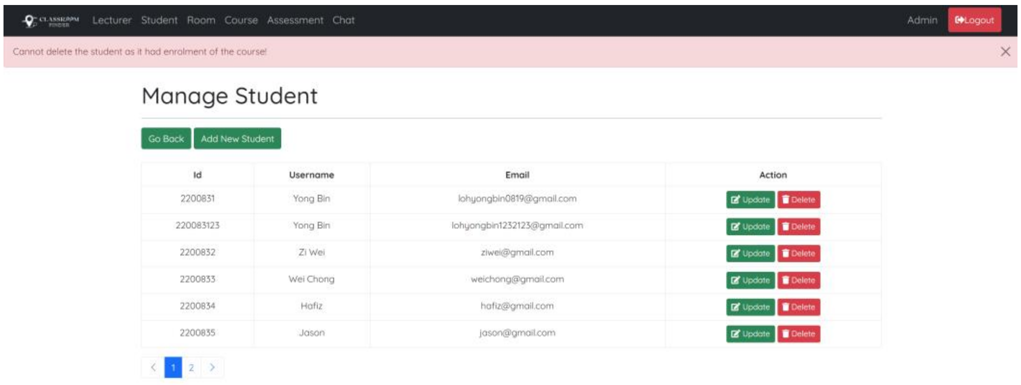
Manage Student

Id	Username	Email	Action
2200831	Yong Bin	lohyongbin0819@gmail.com	Update Delete
220083123	Yong Bin	lohyongbin1232123@gmail.com	Update Delete
2200832	Zi Wei	ziwei@gmail.com	Update Delete
2200833	Wei Chong	weichong@gmail.com	Update Delete
2200834	Hafiz	hafiz@gmail.com	Update Delete
2200835	Jason	jason@gmail.com	Update Delete

< 1 2 >

Figure 6.29: Confirmation Prompt before Deleting Student

Before accessing the delete student route, the system will prompt the administrators with a confirmation dialog before proceeding with the deletion. This confirmation prompt serves as a safeguard, allowing users to verify their action and prevent accidental deletion of student records. Once confirmed, the function checks for any associated enrolment records. If no enrolment records exist, the student is deleted from the database and a success message is displayed.



Cannot delete the student as it had enrolment of the course!

Manage Student

Id	Username	Email	Action
2200831	Yong Bin	lohyongbin0819@gmail.com	Update Delete
220083123	Yong Bin	lohyongbin1232123@gmail.com	Update Delete
2200832	Zi Wei	ziwei@gmail.com	Update Delete
2200833	Wei Chong	weichong@gmail.com	Update Delete
2200834	Hafiz	hafiz@gmail.com	Update Delete
2200835	Jason	jason@gmail.com	Update Delete

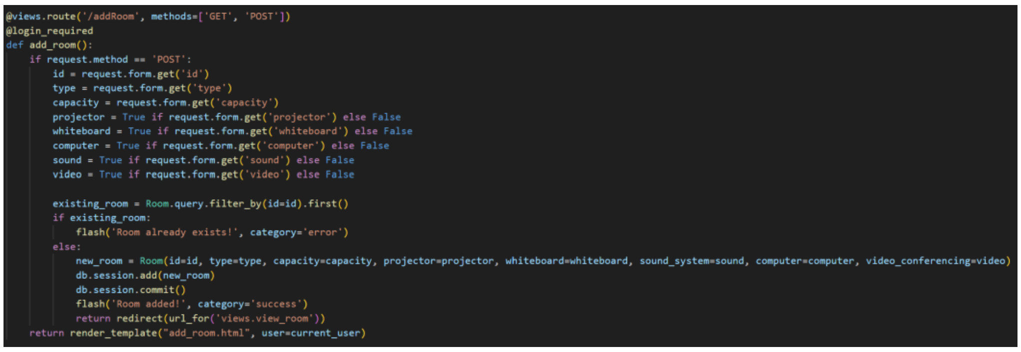
Figure 6.30: Delete Failed with Error Messages

Otherwise, an error message is shown to indicate that the student cannot be deleted due to existing enrolment records.

6.5 Room Management Module

The room management module is an integral component of the Classroom Finder System which responsible for facilitating the efficient administration and oversight of available rooms within the educational institution. This module comprises a suite of functionalities designed to streamline the management of room resources, encompassing operations such as viewing existing rooms, adding new rooms, updating room details, and deleting rooms from the system.

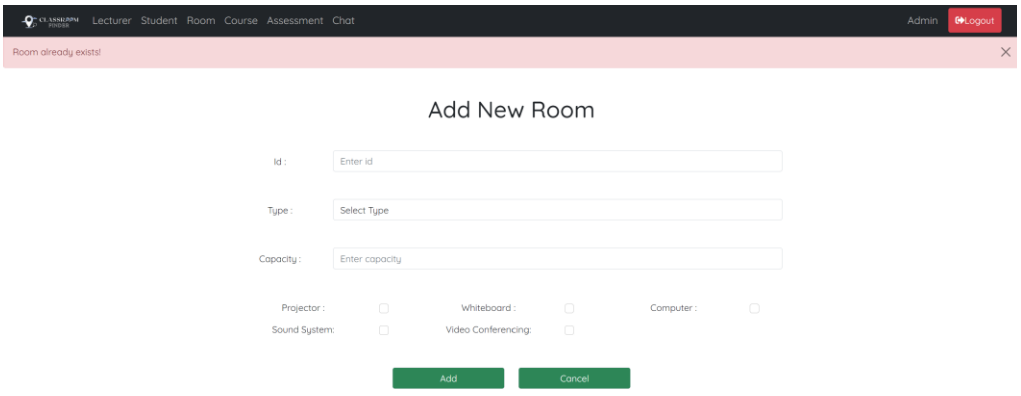
6.5.1 Add Room



```python
@views.route('/addRoom', methods=['GET', 'POST'])
@login_required
def add_room():
    if request.method == 'POST':
        id = request.form.get('id')
        type = request.form.get('type')
        capacity = request.form.get('capacity')
        projector = True if request.form.get('projector') else False
        whiteboard = True if request.form.get('whiteboard') else False
        computer = True if request.form.get('computer') else False
        sound = True if request.form.get('sound') else False
        video = True if request.form.get('video') else False

        existing_room = Room.query.filter_by(id=id).first()
        if existing_room:
            flash('Room already exists!', category='error')
        else:
            new_room = Room(id=id, type=type, capacity=capacity, projector=projector, whiteboard=whiteboard, sound_system=sound, computer=computer, video_conferencing=video)
            db.session.add(new_room)
            db.session.commit()
            flash('Room added!', category='success')
            return redirect(url_for('views.view_room'))
    return render_template("add_room.html", user=current_user)
```

Figure 6.31: Implementation Code for Adding Room

The 'add_room' function enables authorized users to augment the system's repository of rooms by adding new entries for additional spaces within the institution. It allows the administrator to submit the room details via a HTML form. Before submitting the room details of id, type, capacity and the equipment availability, all the input values are required to be filled in to perform the room creation process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



Room already exists!

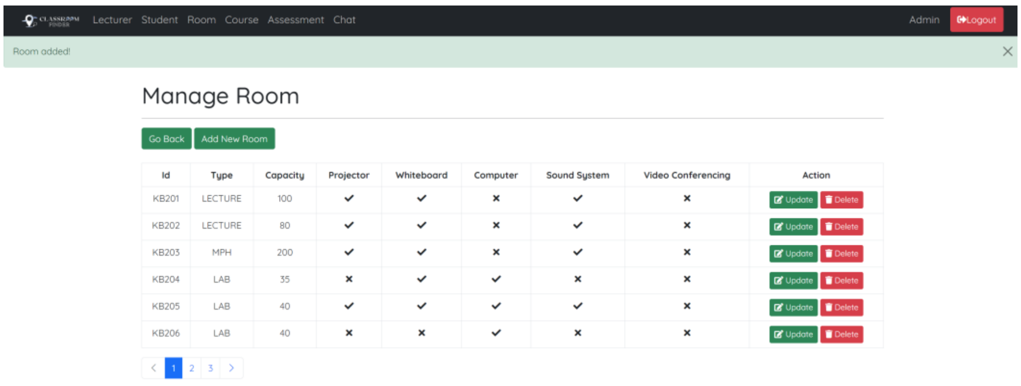
Add New Room

Id :	Enter id
Type :	Select Type
Capacity :	Enter capacity

Projector :	☐	Whiteboard :	☐	Computer :	☐
Sound System:	☐	Video Conferencing:	☐		

Figure 6.32: Creation Failed with Error Messages

Upon submission of room details via the user interface, the system validates the input and verifies the uniqueness of room identifiers to prevent duplication. If there is duplication of the room, the system will notify the administrator with the error message.



Room added!

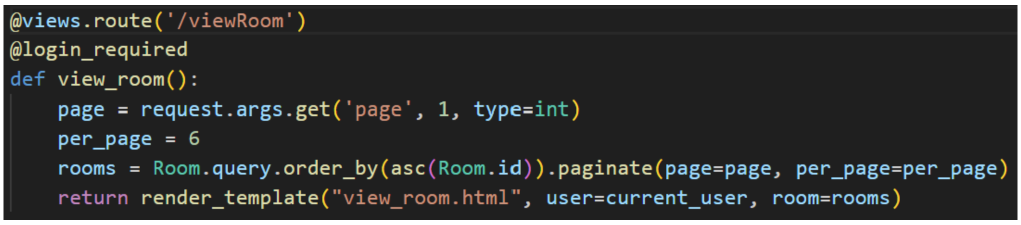
Manage Room

Id	Type	Capacity	Projector	Whiteboard	Computer	Sound System	Video Conferencing	Action
KB201	LECTURE	100	✓	✓	✗	✓	✗	Update Delete
KB202	LECTURE	80	✓	✓	✗	✓	✗	Update Delete
KB203	MPH	200	✓	✓	✗	✓	✗	Update Delete
KB204	LAB	35	✗	✓	✓	✗	✗	Update Delete
KB205	LAB	40	✓	✓	✓	✓	✗	Update Delete
KB206	LAB	40	✗	✗	✓	✗	✗	Update Delete

Figure 6.33: Creation Successfully with Success Messages

Upon successful validation, the new room entry is persisted to the database and administrators are promptly notified of the successful addition.

6.5.2 View Room

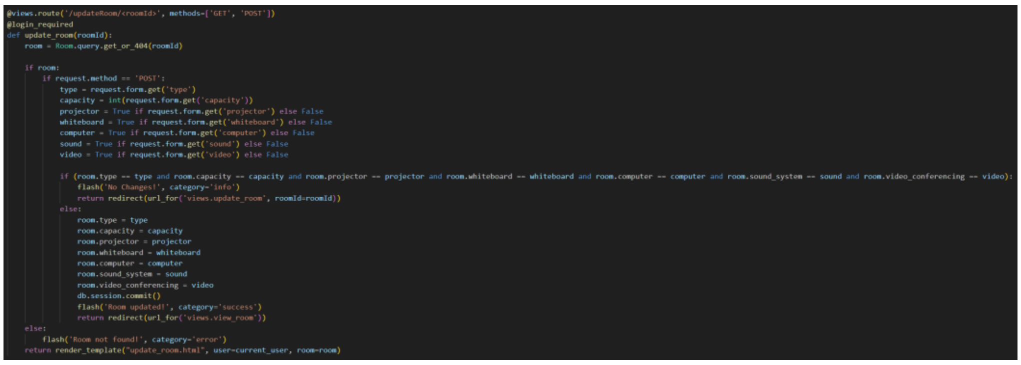


```
@views.route('/viewRoom')
@login_required
def view_room():
    page = request.args.get('page', 1, type=int)
    per_page = 6
    rooms = Room.query.order_by(asc(Room.id)).paginate(page=page, per_page=per_page)
    return render_template("view_room.html", user=current_user, room=rooms)
```

Figure 6.34: Implementation Code for Viewing Room

The 'view_room' function serves as the entry point for the administrators to access a comprehensive listing of all available rooms which presented in a paginated format to enhance usability and navigation. The administrators can navigate through the paginated list to view room details and gain insights into room capacities and amenities.

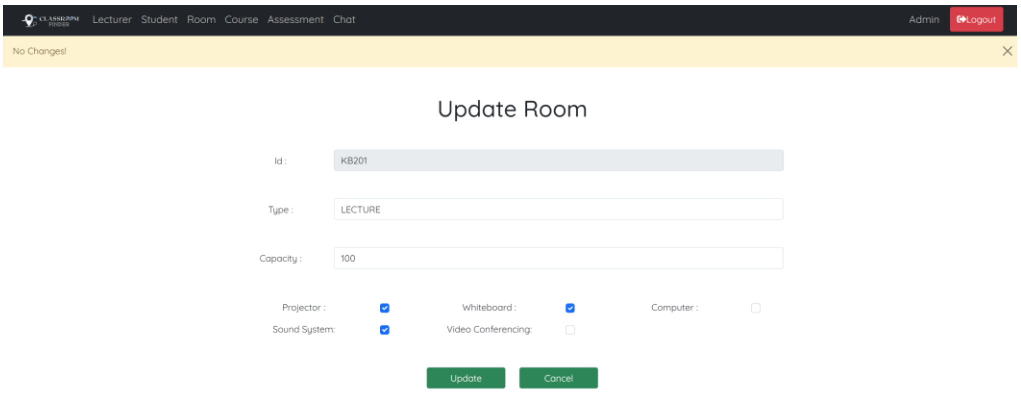
6.5.3 Update Room



```python
@views.route('/updateRoom/<roomId>', methods=['GET', 'POST'])
@login_required
def update_room(roomId):
    room = Room.query.get_or_404(roomId)

    if room:
        if request.method == 'POST':
            type = request.form.get('type')
            capacity = int(request.form.get('capacity'))
            projector = True if request.form.get('projector') else False
            whiteboard = True if request.form.get('whiteboard') else False
            computer = True if request.form.get('computer') else False
            sound = True if request.form.get('sound') else False
            video = True if request.form.get('video') else False

            if (room.type == type and room.capacity == capacity and room.projector == projector and room.whiteboard == whiteboard and room.computer == computer and room.sound_system == sound and room.video_conferencing == video):
                flash('No Changes!', category='info')
                return redirect(url_for('views.update_room', roomId=roomId))
            else:
                room.type = type
                room.capacity = capacity
                room.projector = projector
                room.whiteboard = whiteboard
                room.computer = computer
                room.sound_system = sound
                room.video_conferencing = video
                db.session.commit()
                flash('Room updated!', category='success')
                return redirect(url_for('views.view_room'))
    else:
        flash('Room not found!', category='error')
    return render_template("update_room.html", user=current_user, room=room)
```

Figure 6.35: Implementation Code for Updating Room

The 'update_room' function which accessible via the '/updateRoom/< roomId>' route, allows administrators to update existing room details. The administrators can access the update room form for a specific room by providing their Id in the route. The administrator can update the room by submitting the room details via the HTML form. Before submitting the room details of type, capacity and equipment availability, all the input values are required to be filled in to perform the room updating process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



No Changes!

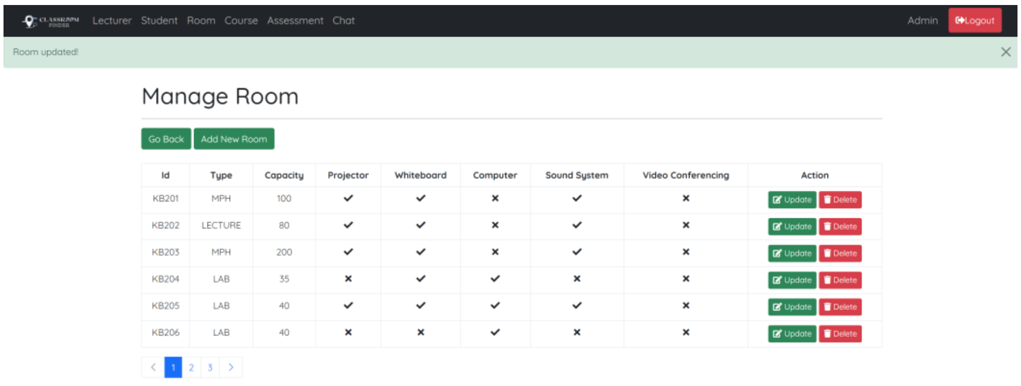
Update Room

Id :	KB201
Type :	LECTURE
Capacity :	100

Projector :	✓	Whiteboard :	✓	Computer :	☐
Sound System:	✓	Video Conferencing:	☐		

Figure 6.36: Update Failed with Error Messages

Upon receiving a POST request containing the necessary information which are type, capacity and equipment availability from the update room form, the function retrieves the room record associated with the provided Id and validates the submitted changes. If there are not any changes, the administrator will be notifying no changes for the updated data.



Room updated!

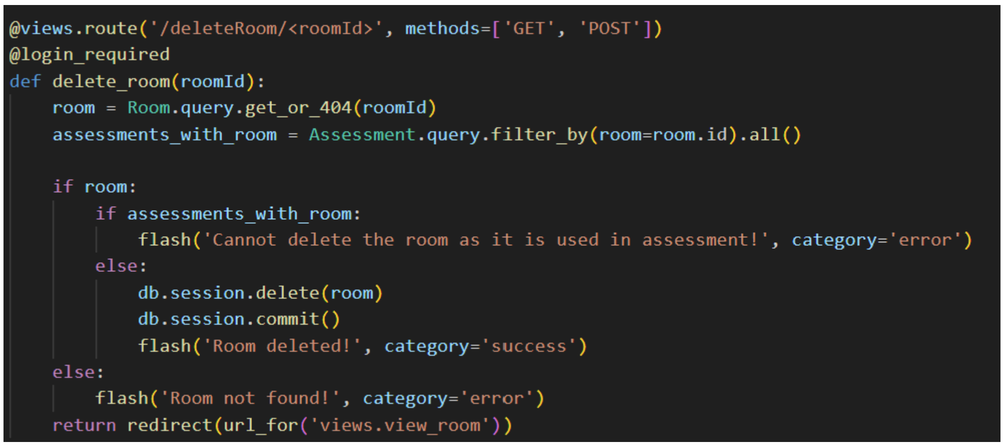
Manage Room

Id	Type	Capacity	Projector	Whiteboard	Computer	Sound System	Video Conferencing	Action
KB201	MPH	100	✓	✓	✗	✓	✗	Update Delete
KB202	LECTURE	80	✓	✓	✗	✓	✗	Update Delete
KB203	MPH	200	✓	✓	✗	✓	✗	Update Delete
KB204	LAB	35	✗	✓	✓	✗	✗	Update Delete
KB205	LAB	40	✓	✓	✓	✓	✗	Update Delete
KB206	LAB	40	✗	✗	✓	✗	✗	Update Delete

Figure 6.37: Update Successfully with Success Messages

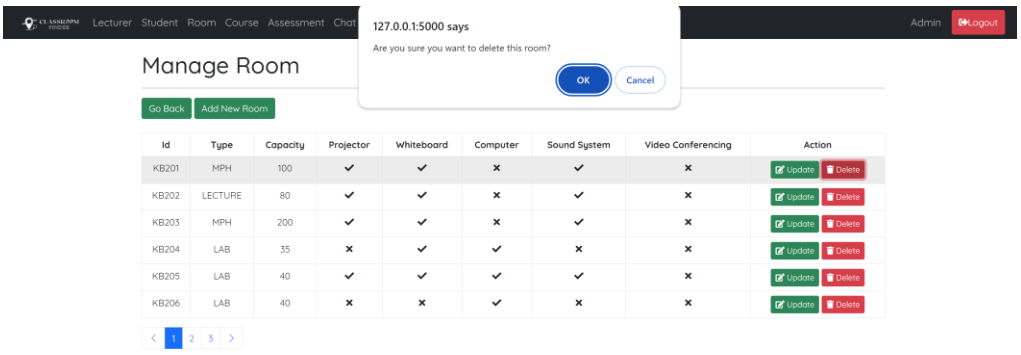
If the submitted data is valid and differs from the existing record, the room details are updated in the database and the administrator is redirected to the view room page with a success message.

6.5.4 Delete Room



```python
@views.route('/deleteRoom/<roomId>', methods=['GET', 'POST'])
@login_required
def delete_room(roomId):
    room = Room.query.get_or_404(roomId)
    assessments_with_room = Assessment.query.filter_by(room=room.id).all()

    if room:
        if assessments_with_room:
            flash('Cannot delete the room as it is used in assessment!', category='error')
        else:
            db.session.delete(room)
            db.session.commit()
            flash('Room deleted!', category='success')
    else:
        flash('Room not found!', category='error')
    return redirect(url_for('views.view_room'))
```

Figure 6.38: Implementation Code for Deleting Room



127.0.0.1:5000 says

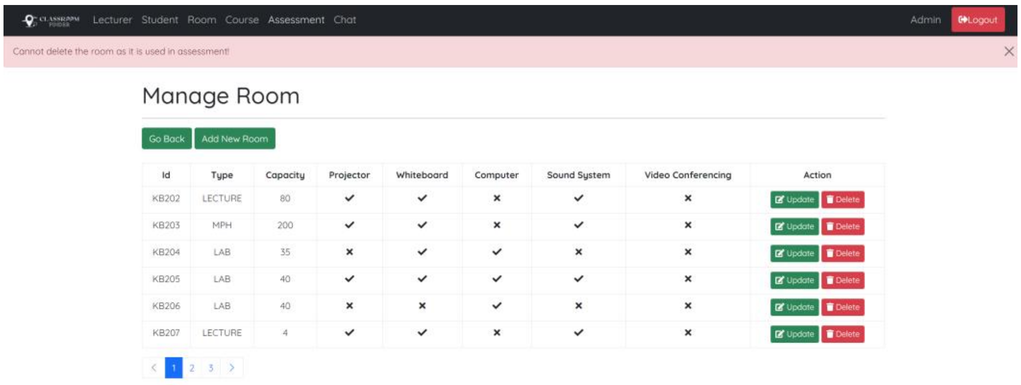
Are you sure you want to delete this room?

Manage Room

Id	Type	Capacity	Projector	Whiteboard	Computer	Sound System	Video Conferencing	Action
KB201	MPH	100	✓	✓	✗	✓	✗	Update Delete
KB202	LECTURE	80	✓	✓	✗	✓	✗	Update Delete
KB203	MPH	200	✓	✓	✗	✓	✗	Update Delete
KB204	LAB	35	✗	✓	✓	✗	✗	Update Delete
KB205	LAB	40	✓	✓	✓	✓	✗	Update Delete
KB206	LAB	40	✗	✗	✓	✗	✗	Update Delete

Figure 6.39: Confirmation Prompt before Deleting Room

To maintain data integrity and ensure accurate representation of available resources, the 'delete_room' function incorporates a confirmation mechanism to verify deletion requests.



Cannot delete the room as it is used in assessment!

Manage Room

Id	Type	Capacity	Projector	Whiteboard	Computer	Sound System	Video Conferencing	Action
KB202	LECTURE	80	✓	✓	✗	✓	✗	Update Delete
KB203	MPH	200	✓	✓	✗	✓	✗	Update Delete
KB204	LAB	35	✗	✓	✓	✗	✗	Update Delete
KB205	LAB	40	✓	✓	✓	✓	✗	Update Delete
KB206	LAB	40	✗	✗	✓	✗	✗	Update Delete
KB207	LECTURE	4	✓	✓	✗	✓	✗	Update Delete

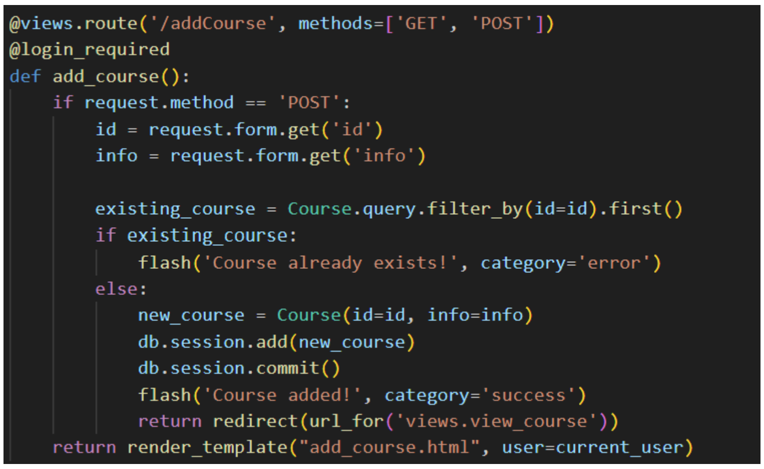
Figure 6.40: Delete Failed with Error Messages

Upon initiating the deletion process, the system checks for any dependencies or associations with assessments to ensure that rooms utilized in ongoing assessments are not inadvertently deleted. If no such dependencies exist, the room entry is removed from the system with users promptly notified of the successful deletion.

6.6 Course Management Module

The course management module plays a pivotal role in facilitating the effective administration and oversight of academic courses offered within the educational institution. This module encompasses a comprehensive suite of functionalities designed to streamline the management of course-related information which includes viewing existing courses, adding new courses, updating course details and deleting courses from the system.

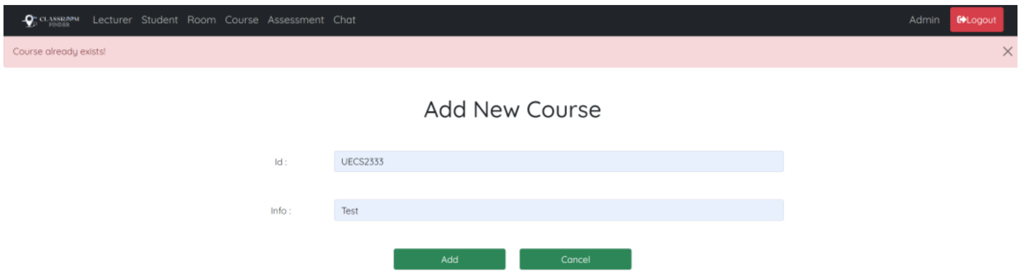
6.6.1 Add Course



```
@views.route('/addCourse', methods=['GET', 'POST'])
@login_required
def add_course():
    if request.method == 'POST':
        id = request.form.get('id')
        info = request.form.get('info')

        existing_course = Course.query.filter_by(id=id).first()
        if existing_course:
            flash('Course already exists!', category='error')
        else:
            new_course = Course(id=id, info=info)
            db.session.add(new_course)
            db.session.commit()
            flash('Course added!', category='success')
            return redirect(url_for('views.view_course'))
    return render_template("add_course.html", user=current_user)
```

Figure 6.41: Implementation Code for Adding Course

The authorized user which is the administrator is empowered to augment the system's repository of courses by utilizing the 'add_course' function to create new entries for additional academic offerings. It allows the administrator to submit the course details via a HTML form. Before submitting the course details of id and info, all the input values are required to be filled in to perform the room creation process. If there is any input field which has empty value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



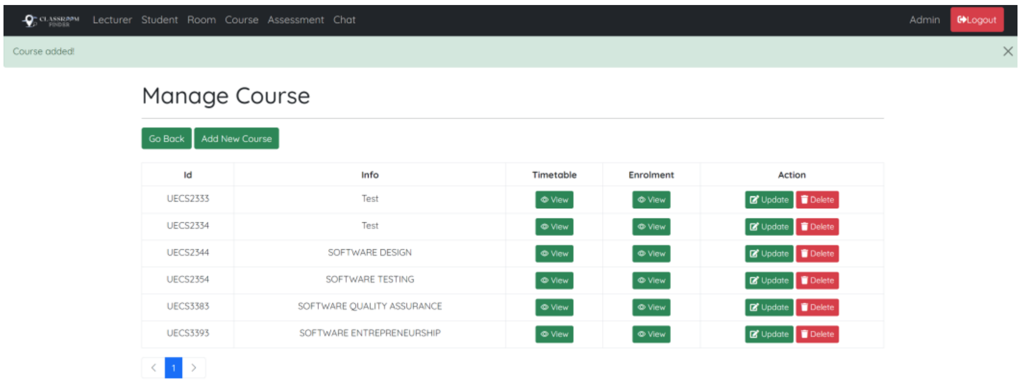
Course already exists!

Add New Course

Id :	UECS2333
Info :	Test

Figure 6.42: Creation Failed with Error Messages

Upon submission of course details via the user interface, the system validates the input and verifies the uniqueness of course identifiers to prevent duplication. If there is duplication of the course, the system will notify the administrator with the error message.



Course added!

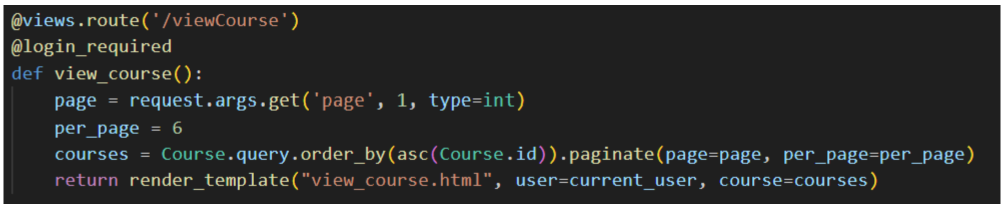
Manage Course

Id	Info	Timetable	Enrolment	Action
UECS2333	Test	View	View	Update Delete
UECS2334	Test	View	View	Update Delete
UECS2344	SOFTWARE DESIGN	View	View	Update Delete
UECS2354	SOFTWARE TESTING	View	View	Update Delete
UECS3383	SOFTWARE QUALITY ASSURANCE	View	View	Update Delete
UECS3393	SOFTWARE ENTREPRENEURSHIP	View	View	Update Delete

Figure 6.43: Creation Successfully with Success Messages

Upon successful validation, the new course entry is persisted to the database and administrators are promptly notified of the successful addition.

6.6.2 View Course

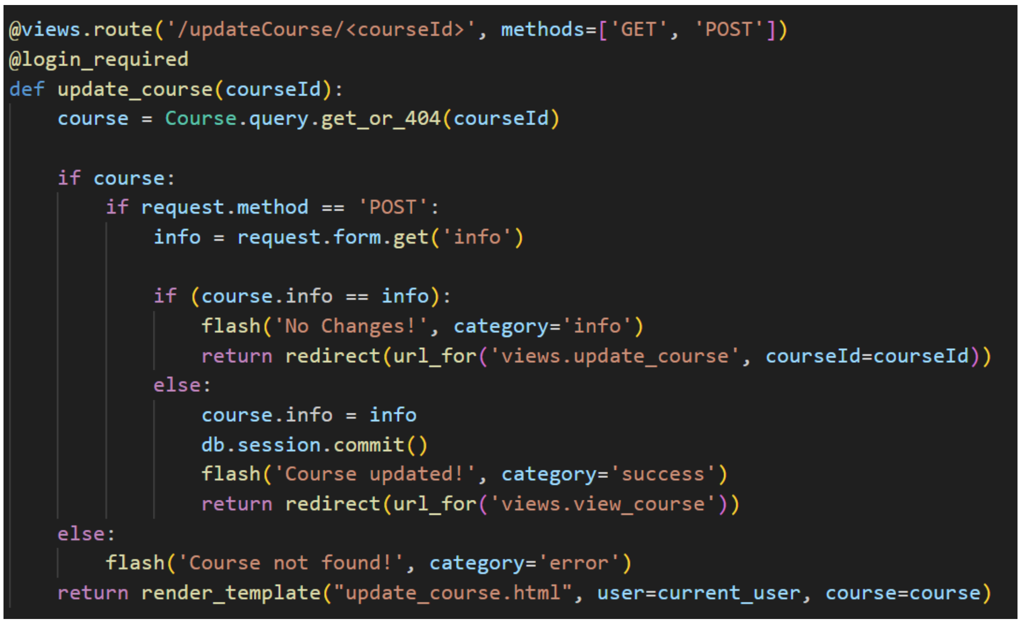


```python
@views.route('/viewCourse')
@login_required
def view_course():
    page = request.args.get('page', 1, type=int)
    per_page = 6
    courses = Course.query.order_by(asc(Course.id)).paginate(page=page, per_page=per_page)
    return render_template("view_course.html", user=current_user, course=courses)
```

Figure 6.44: Implementation Code for Viewing Course

The 'view_course' function serves as the entry point for administrator to access a consolidated listing of all available courses which presented in a paginated format to enhance readability and navigation. The administrator can navigate through the paginated list to review course details such as course identifiers and descriptions to gain valuable insights into the array of academic offerings available within the institution.

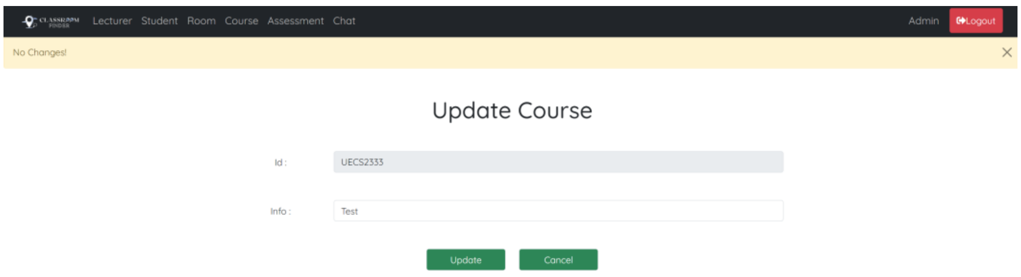
6.6.3 Update Course



```python
@views.route('/updateCourse/<courseId>', methods=['GET', 'POST'])
@login_required
def update_course(courseId):
    course = Course.query.get_or_404(courseId)

    if course:
        if request.method == 'POST':
            info = request.form.get('info')

            if (course.info == info):
                flash('No Changes!', category='info')
                return redirect(url_for('views.update_course', courseId=courseId))
            else:
                course.info = info
                db.session.commit()
                flash('Course updated!', category='success')
                return redirect(url_for('views.view_course'))
    else:
        flash('Course not found!', category='error')
    return render_template("update_course.html", user=current_user, course=course)
```

Figure 6.45: Implementation Code for Updating Course

The 'update_course' function which accessible via the '/updateCourse/< courseId>' route, allows administrators to update existing course details. The administrators can access the update course form for a specific course by providing their Id in the route. The administrator can update the course by submitting the course details via the HTML form. Before submitting the course details of info, all the input values are required to be filled in to perform the room updating process. If there is any input field which has empty value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



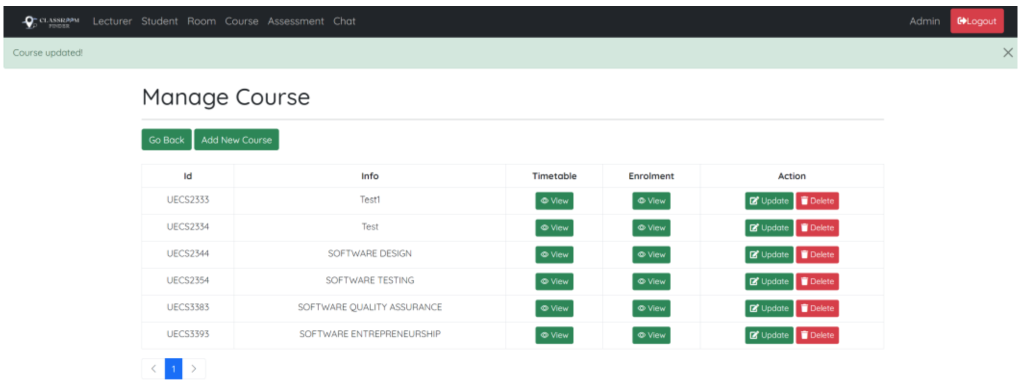
No Changes!

Update Course

Id :	UECS2333
Info :	Test

Figure 6.46: Update Failed with Error Messages

Upon receiving a POST request containing the necessary information which are info from the update room form, the function retrieves the course record associated with the provided Id and validates the submitted changes. If there are not any changes, the administrator will be notifying no changes for the updated data.



Course updated!

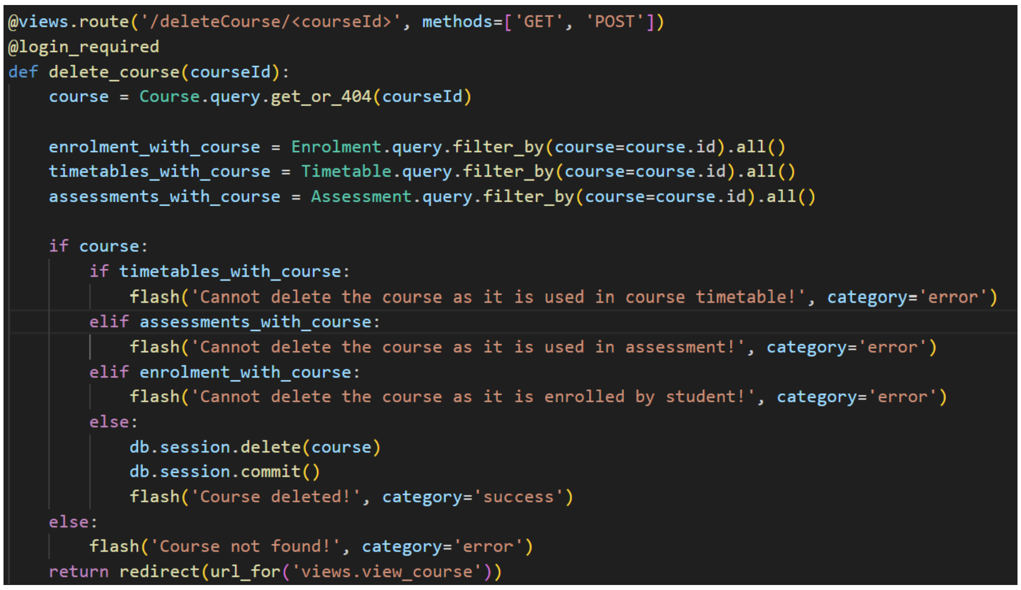
Manage Course

Id	Info	Timetable	Enrolment	Action
UECS2333	Test1	View	View	Update Delete
UECS2334	Test	View	View	Update Delete
UECS2344	SOFTWARE DESIGN	View	View	Update Delete
UECS2354	SOFTWARE TESTING	View	View	Update Delete
UECS3383	SOFTWARE QUALITY ASSURANCE	View	View	Update Delete
UECS3393	SOFTWARE ENTREPRENEURSHIP	View	View	Update Delete

Figure 6.47: Update Successfully with Success Messages

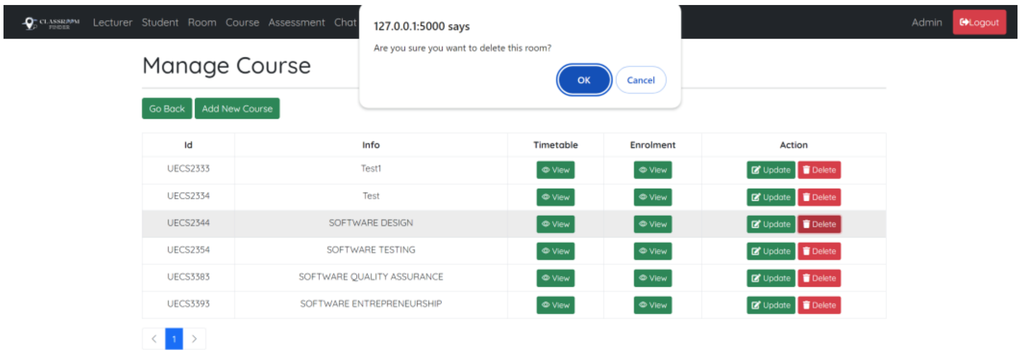
If the submitted data is valid and differs from the existing record, the course details are updated in the database and the administrator is redirected to the view course page with a success message.

6.6.4 Delete Course



```python
@views.route('/deleteCourse/<courseId>', methods=['GET', 'POST'])
@login_required
def delete_course(courseId):
    course = Course.query.get_or_404(courseId)

    enrolment_with_course = Enrolment.query.filter_by(course=course.id).all()
    timetables_with_course = Timetable.query.filter_by(course=course.id).all()
    assessments_with_course = Assessment.query.filter_by(course=course.id).all()

    if course:
        if timetables_with_course:
            flash('Cannot delete the course as it is used in course timetable!', category='error')
        elif assessments_with_course:
            flash('Cannot delete the course as it is used in assessment!', category='error')
        elif enrolment_with_course:
            flash('Cannot delete the course as it is enrolled by student!', category='error')
        else:
            db.session.delete(course)
            db.session.commit()
            flash('Course deleted!', category='success')
    else:
        flash('Course not found!', category='error')
    return redirect(url_for('views.view_course'))
```

Figure 6.48: Implementation Code for Deleting Course



127.0.0.1:5000 says

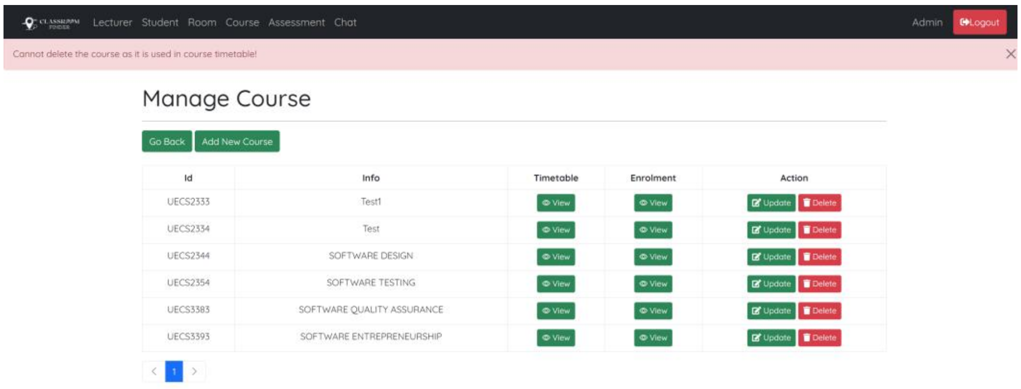
Are you sure you want to delete this room?

Manage Course

Id	Info	Timetable	Enrolment	Action
UECS2333	Test1	View	View	Update Delete
UECS2334	Test	View	View	Update Delete
UECS2344	SOFTWARE DESIGN	View	View	Update Delete
UECS2354	SOFTWARE TESTING	View	View	Update Delete
UECS3383	SOFTWARE QUALITY ASSURANCE	View	View	Update Delete
UECS3393	SOFTWARE ENTREPRENEURSHIP	View	View	Update Delete

Figure 6.49: Confirmation Prompt before Deleting Course

To maintain data integrity and ensure accurate representation of academic offerings, the 'delete_course' function incorporates a confirmation mechanism to verify deletion requests.



Cannot delete the course as it is used in course timetable!

Manage Course

Id	Info	Timetable	Enrolment	Action
UECS2333	Test1	View	View	Update Delete
UECS2334	Test	View	View	Update Delete
UECS2344	SOFTWARE DESIGN	View	View	Update Delete
UECS2354	SOFTWARE TESTING	View	View	Update Delete
UECS3383	SOFTWARE QUALITY ASSURANCE	View	View	Update Delete
UECS3393	SOFTWARE ENTREPRENEURSHIP	View	View	Update Delete

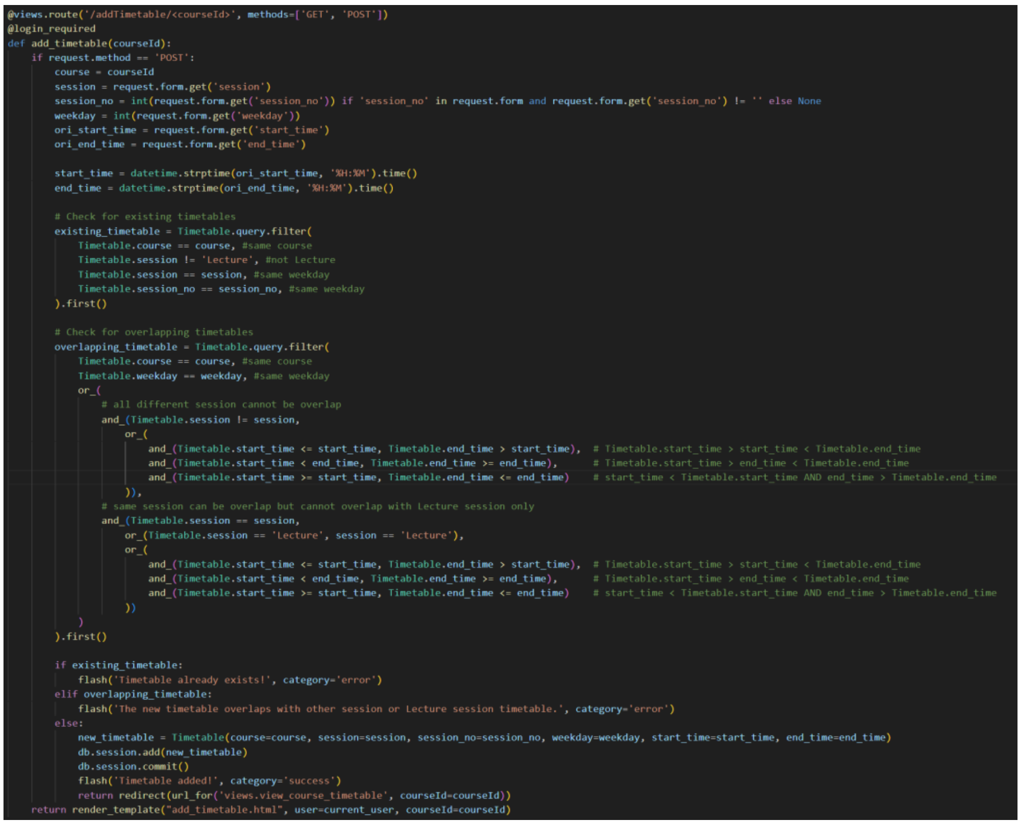
Figure 6.50: Delete Failed with Error Messages

The system checks for any dependencies or associations with enrolments, timetables or assessments to ensure that courses utilized in ongoing academic activities are not inadvertently deleted. If no such dependencies exist, the course entry is removed from the system with users promptly notified of the successful deletion.

6.7 Course Timetable Management Module

The timetable management module serves as a critical component within the Classroom Finder System which facilitates the efficient scheduling and organization of course timetables to optimize academic planning and resource utilization. This module encompasses a suite of functionalities designed to enable administrators and academic staff to view, add, update and delete course timetables seamlessly.

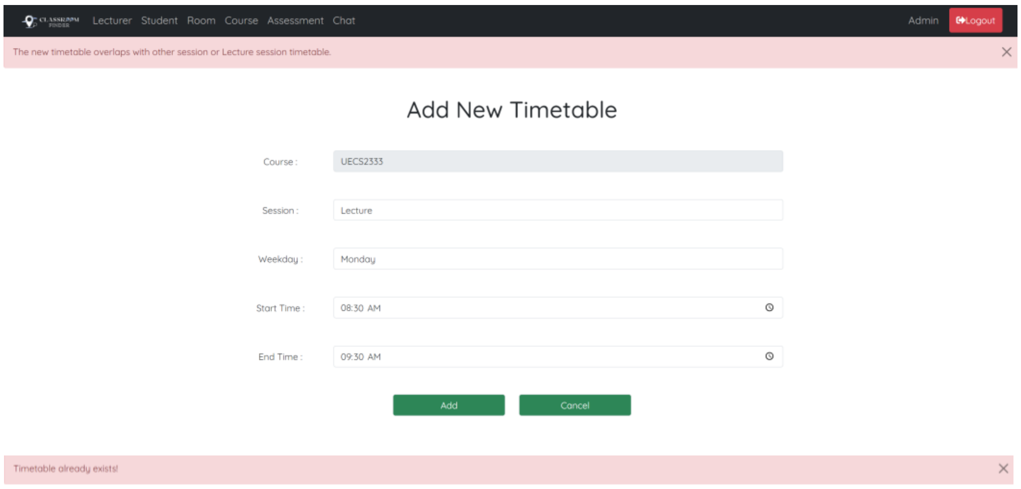
6.7.1 Add Course Timetable



```python
@views.route('/addTimetable/<courseId>', methods=['GET', 'POST'])
@login_required
def add_timetable(courseId):
    if request.method == 'POST':
        course = courseId
        session = request.form.get('session')
        session_no = int(request.form.get('session_no')) if 'session_no' in request.form and request.form.get('session_no') != '' else None
        weekday = int(request.form.get('weekday'))
        ori_start_time = request.form.get('start_time')
        ori_end_time = request.form.get('end_time')

        start_time = datetime.strptime(ori_start_time, '%H:%M').time()
        end_time = datetime.strptime(ori_end_time, '%H:%M').time()

        # Check for existing timetables
        existing_timetable = Timetable.query.filter(
            Timetable.course == course, #same course
            Timetable.session != 'Lecture', #not Lecture
            Timetable.session == session, #same weekday
            Timetable.session_no == session_no, #same weekday
        ).first()

        # Check for overlapping timetables
        overlapping_timetable = Timetable.query.filter(
            Timetable.course == course, #same course
            Timetable.weekday == weekday, #same weekday
            or_(
                # all different session cannot be overlap
                and_(Timetable.session != session,
                    or_(
                        and_(Timetable.start_time <= start_time, Timetable.end_time > start_time),  # Timetable.start_time > start_time < Timetable.end_time
                        and_(Timetable.start_time < end_time, Timetable.end_time >= end_time),      # Timetable.start_time > end_time < Timetable.end_time
                        and_(Timetable.start_time >= start_time, Timetable.end_time <= end_time)    # start_time < Timetable.start_time AND end_time > Timetable.end_time
                    )),
                # same session can be overlap but cannot overlap with Lecture session only
                and_(Timetable.session == session,
                    or_(Timetable.session == 'Lecture', session == 'Lecture'),
                    or_(
                        and_(Timetable.start_time <= start_time, Timetable.end_time > start_time),  # Timetable.start_time > start_time < Timetable.end_time
                        and_(Timetable.start_time < end_time, Timetable.end_time >= end_time),      # Timetable.start_time > end_time < Timetable.end_time
                        and_(Timetable.start_time >= start_time, Timetable.end_time <= end_time)    # start_time < Timetable.start_time AND end_time > Timetable.end_time
                    ))
            )
        ).first()

        if existing_timetable:
            flash('Timetable already exists!', category='error')
        elif overlapping_timetable:
            flash('The new timetable overlaps with other session or Lecture session timetable.', category='error')
        else:
            new_timetable = Timetable(course=course, session=session, session_no=session_no, weekday=weekday, start_time=start_time, end_time=end_time)
            db.session.add(new_timetable)
            db.session.commit()
            flash('Timetable added!', category='success')
            return redirect(url_for('views.view_course_timetable', courseId=courseId))
    return render_template("add_timetable.html", user=current_user, courseId=courseId)
```

Figure 6.51: Implementation Code for Adding Timetable

For administrators tasked with creating new course timetables, the 'add_timetable' function offers an intuitive interface for specifying timetable session details, including session types, session numbers, weekdays, and session times. It allows the administrator to submit the timetable details via a HTML form. Before submitting the timetable details of session, session_no, weekday, start time and end time, all the input values are required to be filled in to perform the timetable creation process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



The new timetable overlaps with other session or Lecture session timetable.

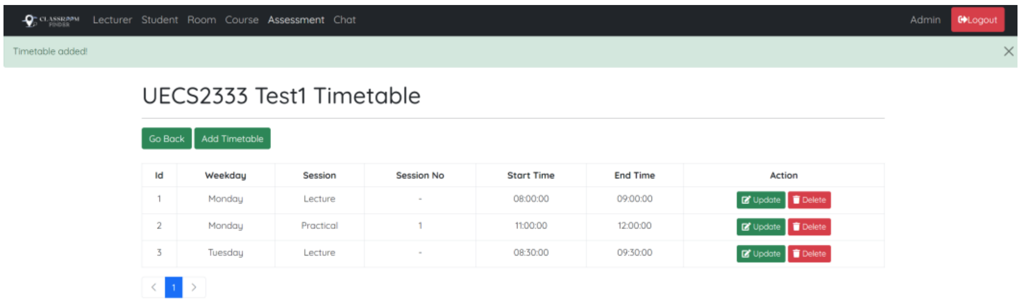
Add New Timetable

Course :	UECS2333
Session :	Lecture
Weekday :	Monday
Start Time :	08:30 AM
End Time :	09:30 AM

Timetable already exists!

Figure 6.52: Creation Failed with Error Messages

Upon submission of timetable details via the user interface, to prevent scheduling conflicts and ensure data integrity, the system performs rigorous validation checks to verify the uniqueness of timetable entries and detect overlapping schedules with existing sessions or lectures.



Timetable added!

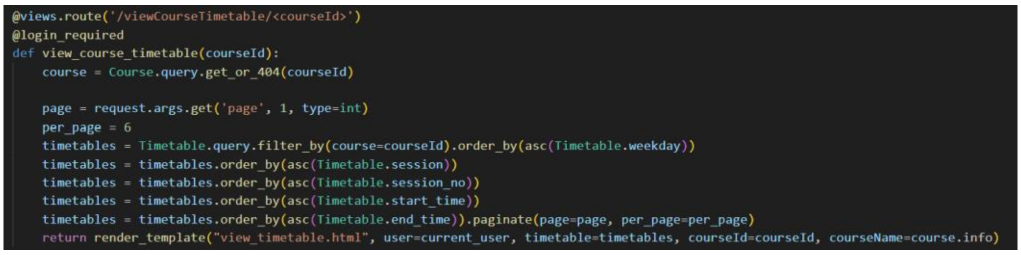
UECS2333 Test1 Timetable

Id	Weekday	Session	Session No	Start Time	End Time	Action
1	Monday	Lecture	-	08:00:00	09:00:00	Update Delete
2	Monday	Practical	1	11:00:00	12:00:00	Update Delete
3	Tuesday	Lecture	-	08:30:00	09:30:00	Update Delete

Figure 6.53: Creation Successfully with Success Messages

Upon successful validation, the new timetable entry is persisted to the database and administrators are promptly notified of the successful addition.

6.7.2 View Course Timetable

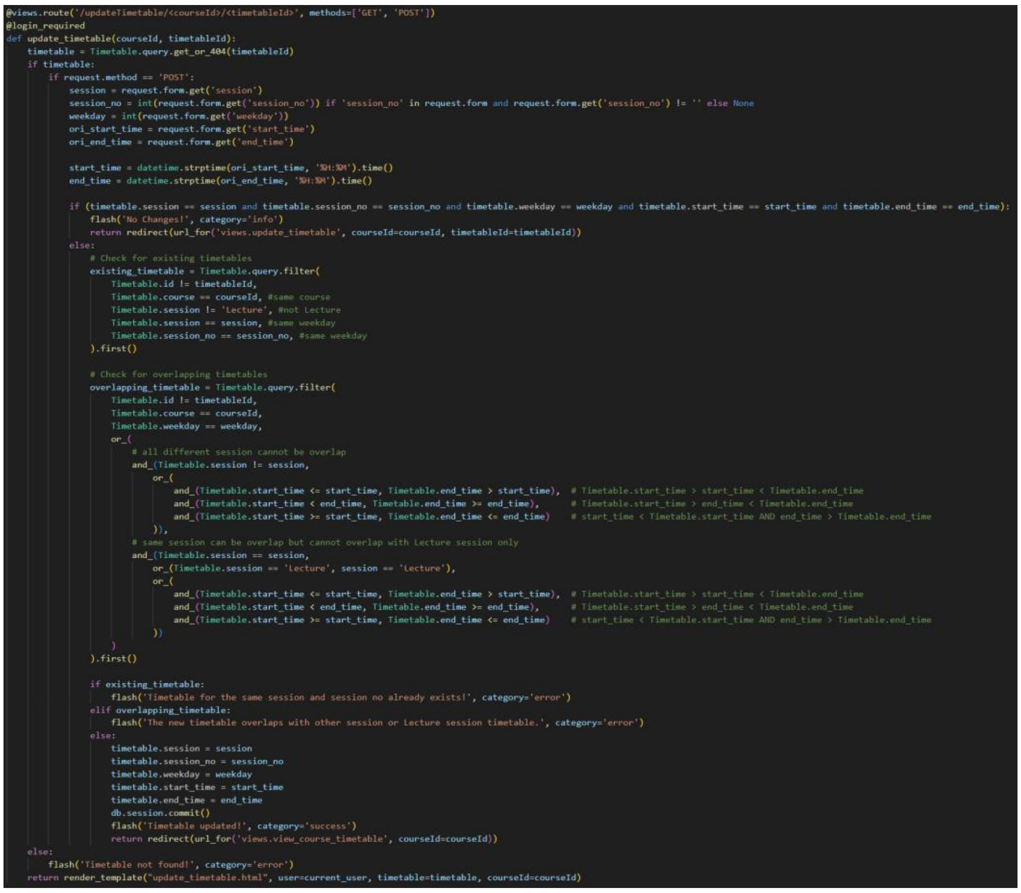


```python
@views.route('/viewCourseTimetable/<courseId>')
@login_required
def view_course_timetable(courseId):
    course = Course.query.get_or_404(courseId)

    page = request.args.get('page', 1, type=int)
    per_page = 6
    timetables = Timetable.query.filter_by(course=courseId).order_by(asc(Timetable.weekday))
    timetables = timetables.order_by(asc(Timetable.session))
    timetables = timetables.order_by(asc(Timetable.session_no))
    timetables = timetables.order_by(asc(Timetable.start_time))
    timetables = timetables.order_by(asc(Timetable.end_time)).paginate(page=page, per_page=per_page)
    return render_template("view_timetable.html", user=current_user, timetable=timetables, courseId=courseId, courseName=course.info)
```

Figure 6.54: Implementation Code for Viewing Course Timetable

The 'view_course_timetable' function enables users to access and review course timetables associated with a specific academic course which provides a comprehensive overview of scheduled sessions, including weekdays, session start time, session end time, session and session number. By leveraging pagination techniques, the system enhances user experience by presenting course timetables in a structured and navigable format and facilitating easy access to pertinent information.

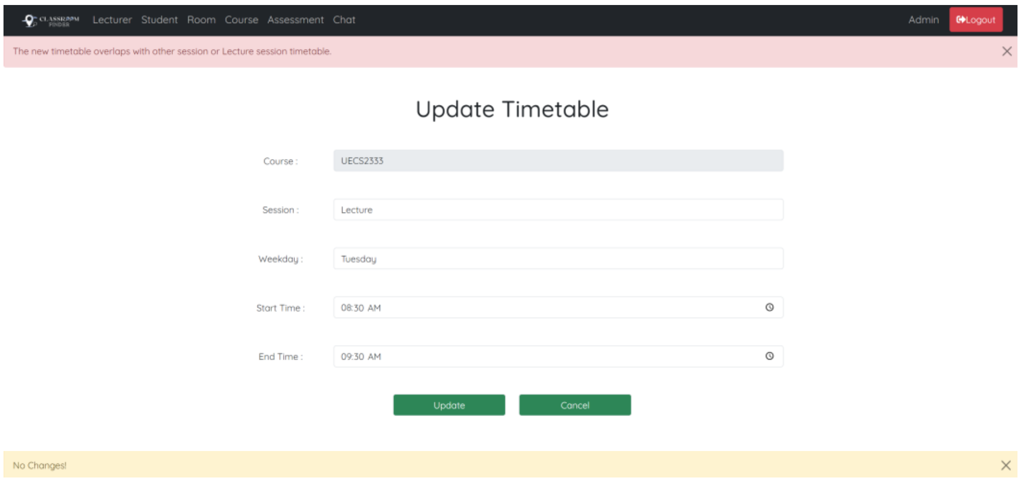
6.7.3 Update Course Timetable



```python
@views.route('/updateTimetable/<courseId>/<timetableId>', methods=['GET', 'POST'])
@login_required
def update_timetable(courseId, timetableId):
    timetable = Timetable.query.get_or_404(timetableId)
    if timetable:
        if request.method == 'POST':
            session = request.form.get('session')
            session_no = int(request.form.get('session_no')) if 'session_no' in request.form and request.form.get('session_no') != '' else None
            weekday = int(request.form.get('weekday'))
            ori_start_time = request.form.get('start_time')
            ori_end_time = request.form.get('end_time')

            start_time = datetime.strptime(ori_start_time, '%H:%M').time()
            end_time = datetime.strptime(ori_end_time, '%H:%M').time()

            if (timetable.session == session and timetable.session_no == session_no and timetable.weekday == weekday and timetable.start_time == start_time and timetable.end_time == end_time):
                flash('No Changes!', category='info')
                return redirect(url_for('views.update_timetable', courseId=courseId, timetableId=timetableId))
            else:
                # Check for existing timetables
                existing_timetable = Timetable.query.filter(
                    Timetable.id != timetableId,
                    Timetable.course == courseId, #same course
                    Timetable.session != 'Lecture', #not Lecture
                    Timetable.session == session, #same weekday
                    Timetable.session_no == session_no, #same weekday
                ).first()

                # Check for overlapping timetables
                overlapping_timetable = Timetable.query.filter(
                    Timetable.id != timetableId,
                    Timetable.course == courseId,
                    Timetable.weekday == weekday,
                    or_(
                        # all different session cannot be overlap
                        and_(Timetable.session != session,
                            or_(
                                and_(Timetable.start_time <= start_time, Timetable.end_time > start_time),  # Timetable.start_time > start_time < Timetable.end_time
                                and_(Timetable.start_time < end_time, Timetable.end_time >= end_time),      # Timetable.start_time > end_time < Timetable.end_time
                                and_(Timetable.start_time >= start_time, Timetable.end_time <= end_time)    # start_time < Timetable.start_time AND end_time > Timetable.end_time
                            )),
                        # same session can be overlap but cannot overlap with Lecture session only
                        and_(Timetable.session == session,
                            or_(Timetable.session == 'Lecture', session == 'Lecture'),
                            or_(
                                and_(Timetable.start_time <= start_time, Timetable.end_time > start_time),  # Timetable.start_time > start_time < Timetable.end_time
                                and_(Timetable.start_time < end_time, Timetable.end_time >= end_time),      # Timetable.start_time > end_time < Timetable.end_time
                                and_(Timetable.start_time >= start_time, Timetable.end_time <= end_time)    # start_time < Timetable.start_time AND end_time > Timetable.end_time
                            ))
                    )
                ).first()

                if existing_timetable:
                    flash('Timetable for the same session and session no already exists!', category='error')
                elif overlapping_timetable:
                    flash('The new timetable overlaps with other session or Lecture session timetable.', category='error')
                else:
                    timetable.session = session
                    timetable.session_no = session_no
                    timetable.weekday = weekday
                    timetable.start_time = start_time
                    timetable.end_time = end_time
                    db.session.commit()
                    flash('Timetable updated!', category='success')
                    return redirect(url_for('views.view_course_timetable', courseId=courseId))
    else:
        flash('Timetable not found!', category='error')
    return render_template("update_timetable.html", user=current_user, timetable=timetable, courseId=courseId)
```

Figure 6.55: Implementation Code for Updating Course Timetable

The 'update_timetable' function which accessible via the '/updateTimetable/< courseId>/< timetableId>' route, allows administrator to update existing course timetable details for the particular course. The administrator can access the update course timetable form of a specific course by providing the courseId and timetableId in the route. The administrator can update the course timetable by submitting the course timetable details via the HTML form. Before submitting the course timetable details of session, session_no, weekday, start time and end time, all the input values are required to be filled in to perform the timetable updating process while each input value also needs to meet its respective input format. If there is any input field which has empty value or wrong input format value, the system will display the error message to notify the administrator when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



The new timetable overlaps with other session or Lecture session timetable.

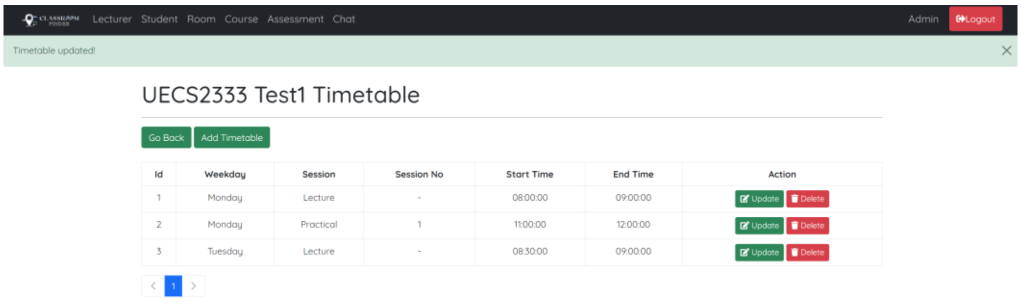
Update Timetable

Course :	UECS2333
Session :	Lecture
Weekday :	Tuesday
Start Time :	08:30 AM
End Time :	09:30 AM

No Changes!

Figure 6.56: Update Failed with Error Messages

Upon receiving a POST request containing the necessary information which are session, session_no, weekday, start time and end time from the update timetable form, the function retrieves the course timetable record associated with the provided Id and validates the submitted changes. The system employs sophisticated algorithms to detect and prevent scheduling conflicts, thereby safeguarding against inadvertent overlaps and ensuring the accuracy of updated timetable entries. While if there are not any changes, the administrator will be notifying no changes for the updated data.



Timetable updated!

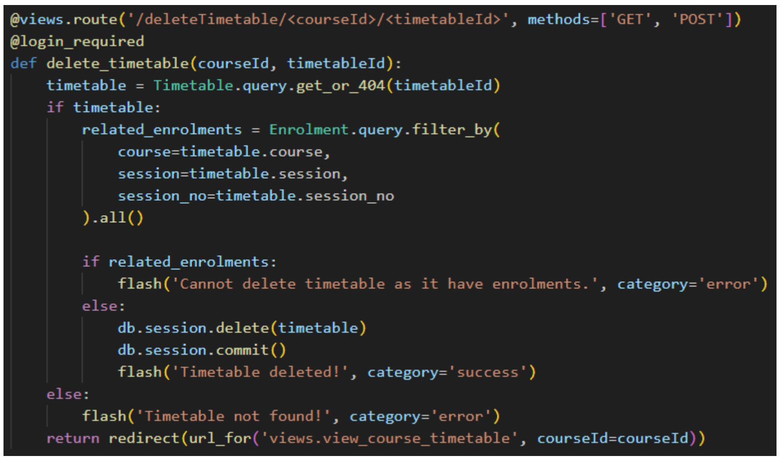
UECS2333 Test1 Timetable

Id	Weekday	Session	Session No	Start Time	End Time	Action
1	Monday	Lecture	-	08:00:00	09:00:00	Update Delete
2	Monday	Practical	1	11:00:00	12:00:00	Update Delete
3	Tuesday	Lecture	-	08:30:00	09:00:00	Update Delete

Figure 6.57: Update Successfully with Success Messages

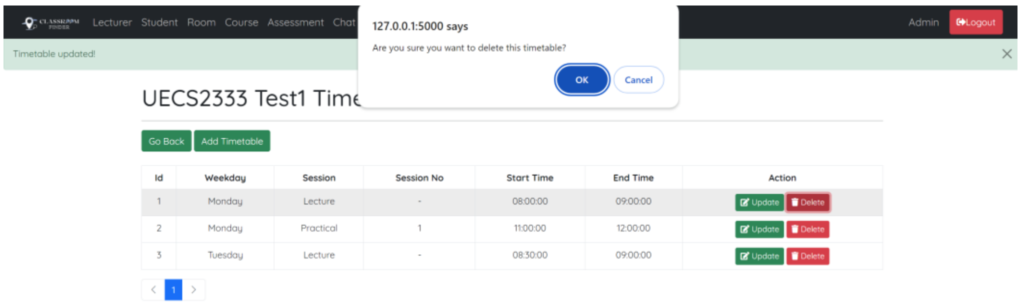
If the submitted data is valid and differs from the existing record, the course timetable details are updated in the database and the administrator is redirected to the view course timetable page with a success message.

6.7.4 Delete Course Timetable



```
@views.route('/deleteTimetable/<courseId>/<timetableId>', methods=['GET', 'POST'])
@login_required
def delete_timetable(courseId, timetableId):
    timetable = Timetable.query.get_or_404(timetableId)
    if timetable:
        related_enrolments = Enrolment.query.filter_by(
            course=timetable.course,
            session=timetable.session,
            session_no=timetable.session_no
        ).all()

        if related_enrolments:
            flash('Cannot delete timetable as it have enrolments.', category='error')
        else:
            db.session.delete(timetable)
            db.session.commit()
            flash('Timetable deleted!', category='success')
    else:
        flash('Timetable not found!', category='error')
    return redirect(url_for('views.view_course_timetable', courseId=courseId))
```

Figure 6.58: Implementation Code for Deleting Course Timetable



127.0.0.1:5000 says

Are you sure you want to delete this timetable?

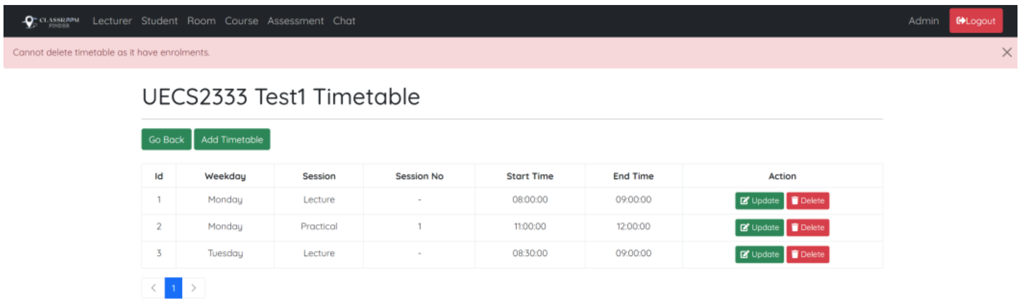
Timetable updated!

UECS2333 Test1 Time

Id	Weekday	Session	Session No	Start Time	End Time	Action
1	Monday	Lecture	-	08:00:00	09:00:00	Update Delete
2	Monday	Practical	1	11:00:00	12:00:00	Update Delete
3	Tuesday	Lecture	-	08:30:00	09:00:00	Update Delete

Figure 6.59: Confirmation Prompt before Deleting Course Timetable

To maintain data integrity and ensure accurate representation of timetable of the academic offerings, the 'delete_timetable' function incorporates a confirmation mechanism to verify deletion requests.



Cannot delete timetable as it have enrolments.

UECS2333 Test1 Timetable

Id	Weekday	Session	Session No	Start Time	End Time	Action
1	Monday	Lecture	-	08:00:00	09:00:00	Update Delete
2	Monday	Practical	1	11:00:00	12:00:00	Update Delete
3	Tuesday	Lecture	-	08:30:00	09:00:00	Update Delete

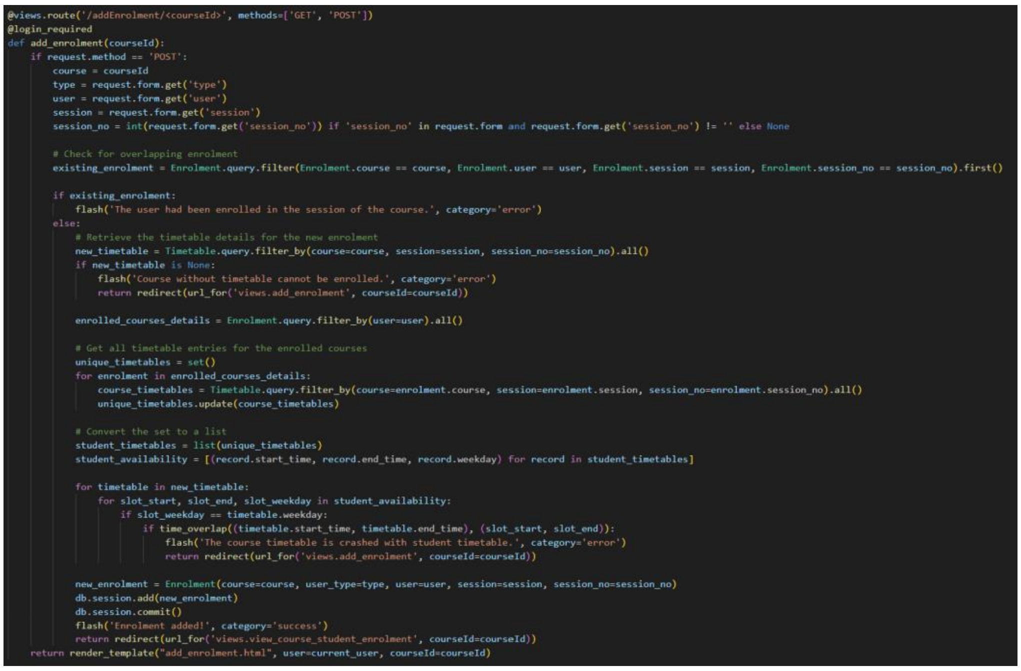
Figure 6.60: Delete Failed with Error Messages

To maintain data consistency and optimize resource allocation, the 'delete_timetable' function also incorporates robust validation mechanisms to prevent the deletion of course timetables associated with active enrolments or sessions. By enforcing these safeguards, the system mitigates the risk of data loss or disruption to ongoing academic activities to foster a reliable and resilient scheduling environment.

6.8 Course Enrolment Management Module

The course enrolment management module serves as a fundamental component of the Classroom Finder System which facilitates the seamless registration and organization of students and lecturers into specific academic courses. This module encompasses a suite of functionalities designed to enable administrators to efficiently manage enrolment processes, monitor student and lecturer participation and ensure the smooth operation of academic activities.

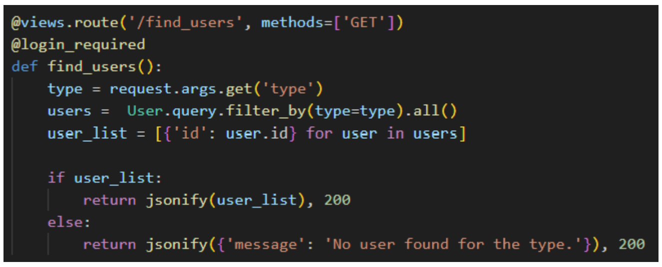
6.8.1 Add Course Enrolment



```python
@views.route('/addEnrolment/<courseId>', methods=['GET', 'POST'])
@login_required
def add_enrolment(courseId):
    if request.method == 'POST':
        course = courseId
        type = request.form.get('type')
        user = request.form.get('user')
        session = request.form.get('session')
        session_no = int(request.form.get('session_no')) if 'session_no' in request.form and request.form.get('session_no') != '' else None

        # Check for overlapping enrolment
        existing_enrolment = Enrolment.query.filter(Enrolment.course == course, Enrolment.user == user, Enrolment.session == session, Enrolment.session_no == session_no).first()

        if existing_enrolment:
            flash('The user had been enrolled in the session of the course.', category='error')
        else:
            # Retrieve the timetable details for the new enrolment
            new_timetable = Timetable.query.filter_by(course=course, session=session, session_no=session_no).all()
            if new_timetable is None:
                flash('Course without timetable cannot be enrolled.', category='error')
                return redirect(url_for('views.add_enrolment', courseId=courseId))

            enrolled_courses_details = Enrolment.query.filter_by(user=user).all()

            # Get all timetable entries for the enrolled courses
            unique_timetables = set()
            for enrolment in enrolled_courses_details:
                course_timetables = Timetable.query.filter_by(course=enrolment.course, session=enrolment.session, session_no=enrolment.session_no).all()
                unique_timetables.update(course_timetables)

            # Convert the set to a list
            student_timetables = list(unique_timetables)
            student_availability = [(record.start_time, record.end_time, record.weekday) for record in student_timetables]

            for timetable in new_timetable:
                for slot_start, slot_end, slot_weekday in student_availability:
                    if slot_weekday == timetable.weekday:
                        if time_overlap((timetable.start_time, timetable.end_time), (slot_start, slot_end)):
                            flash('The course timetable is crashed with student timetable.', category='error')
                            return redirect(url_for('views.add_enrolment', courseId=courseId))

            new_enrolment = Enrolment(course=course, user_type=type, user=user, session=session, session_no=session_no)
            db.session.add(new_enrolment)
            db.session.commit()
            flash('Enrolment added!', category='success')
            return redirect(url_for('views.view_course_student_enrolment', courseId=courseId))
    return render_template("add_enrolment.html", user=current_user, courseId=courseId)
```

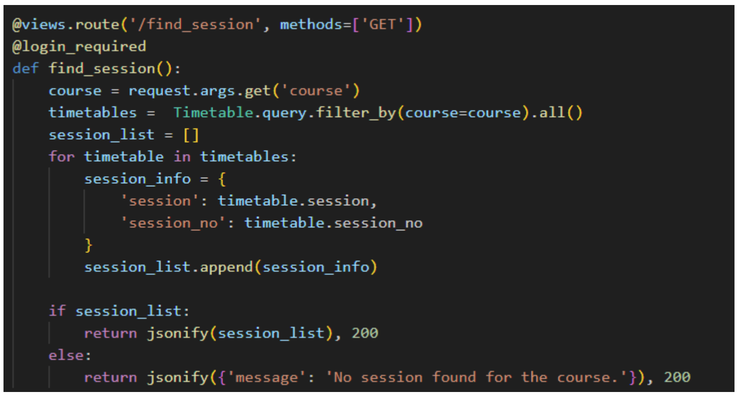
Figure 6.61: Implementation Code for Adding Course Enrolment

For administrators tasked with adding new enrolments, the 'add_enrolment' function offers an intuitive interface for specifying enrolment details, including user type (student or lecturer), user Id, session and session number. It allows the administrator to submit the enrolment details via a HTML form.



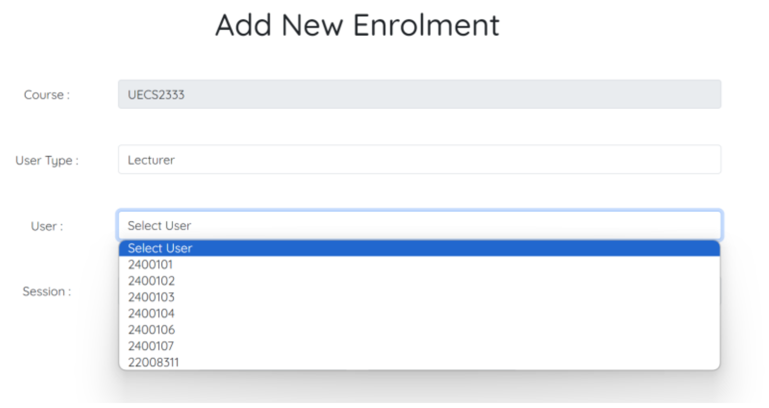
```python
@views.route('/find_users', methods=['GET'])
@login_required
def find_users():
    type = request.args.get('type')
    users =  User.query.filter_by(type=type).all()
    user_list = [{'id': user.id} for user in users]

    if user_list:
        return jsonify(user_list), 200
    else:
        return jsonify({'message': 'No user found for the type.'}), 200
```

Figure 6.62: Implementation Code for Finding User based on Type



```python
@views.route('/find_session', methods=['GET'])
@login_required
def find_session():
    course = request.args.get('course')
    timetables =  Timetable.query.filter_by(course=course).all()
    session_list = []
    for timetable in timetables:
        session_info = {
            'session': timetable.session,
            'session_no': timetable.session_no
        }
        session_list.append(session_info)

    if session_list:
        return jsonify(session_list), 200
    else:
        return jsonify({'message': 'No session found for the course.'}), 200
```

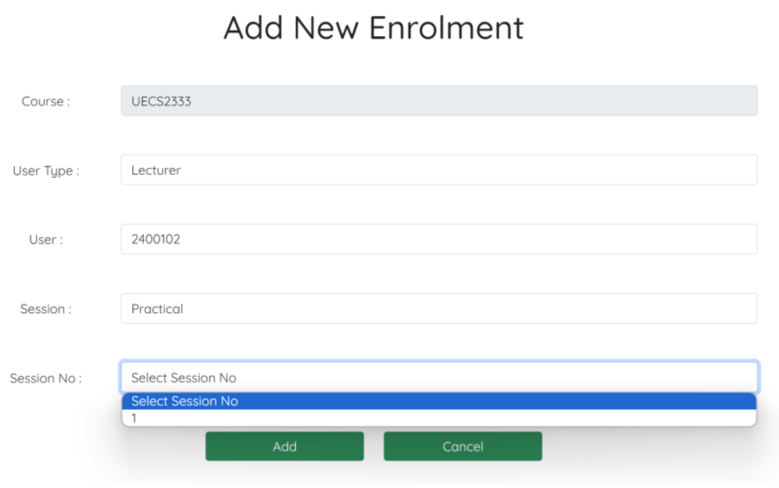
Figure 6.63: Implementation Code for Finding Session No based on Session



Add New Enrolment

Course :	UECS2333
User Type :	Lecturer
User :	Select User
Session :	

Figure 6.64: Get the user based on user type

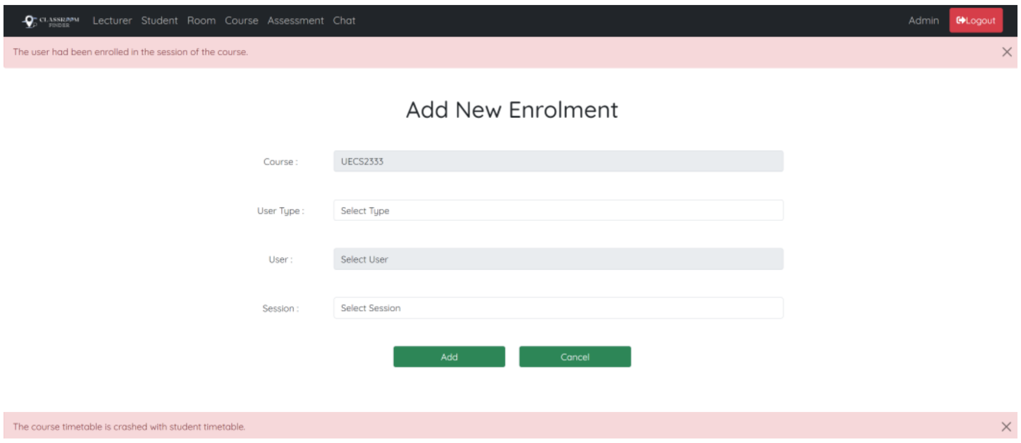


Add New Enrolment

Course :	UECS2333
User Type :	Lecturer
User :	2400102
Session :	Practical
Session No :	Select Session No

Figure 6.65: Get the session no based on session

In the HTML form, the 'find_users' and 'find_session' functions enhance the user experience by providing dynamic search capabilities which allow administrators to quickly locate users or session no based on specific criteria. By offering real-time access to relevant information, these functions empower users with the tools necessary to streamline enrolment processes and enhance overall operational efficiency within the institution. Before submitting the enrolment details, all the input values are required to be filled in to perform the enrolment creation process. If there is any input field which has empty value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



The user had been enrolled in the session of the course.

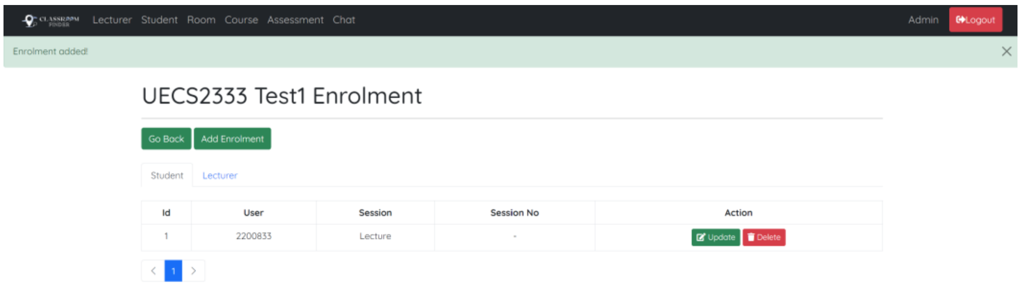
Add New Enrolment

Course :	UECS2333
User Type :	Select Type
User :	Select User
Session :	Select Session

The course timetable is crashed with student timetable.

Figure 6.66: Creation Failed with Error Messages

Upon submission of enrolment details via the user interface, to ensure data integrity and prevent scheduling conflicts, the system performs rigorous validation checks to verify the uniqueness of enrolment entries and detect overlapping schedules with existing enrolments or course timetables.



Enrolment added!

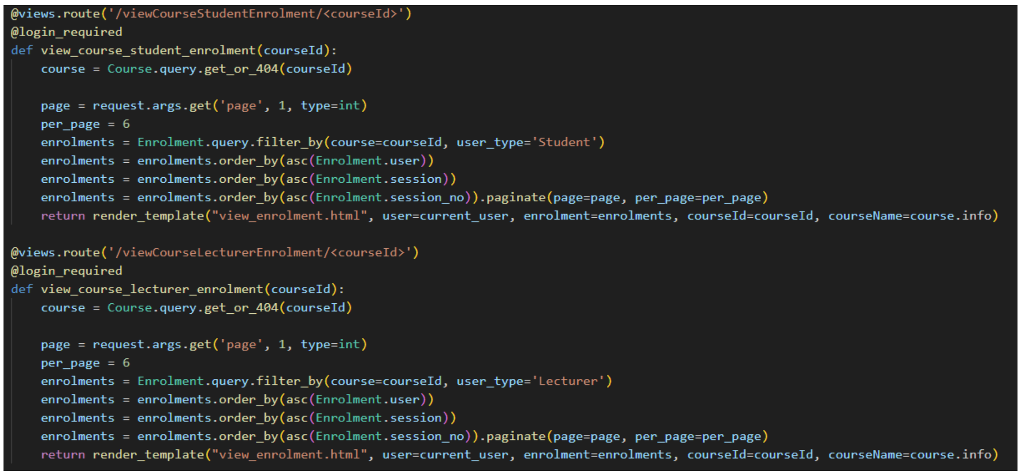
UECS2333 Test1 Enrolment

Id	User	Session	Session No	Action
1	2200833	Lecture	-	Update Delete

Figure 6.67: Creation Successfully with Success Messages

Upon successful validation, the new enrolment entry is persisted to the database and administrators are promptly notified of the successful addition.

6.8.2 View Course Enrolment

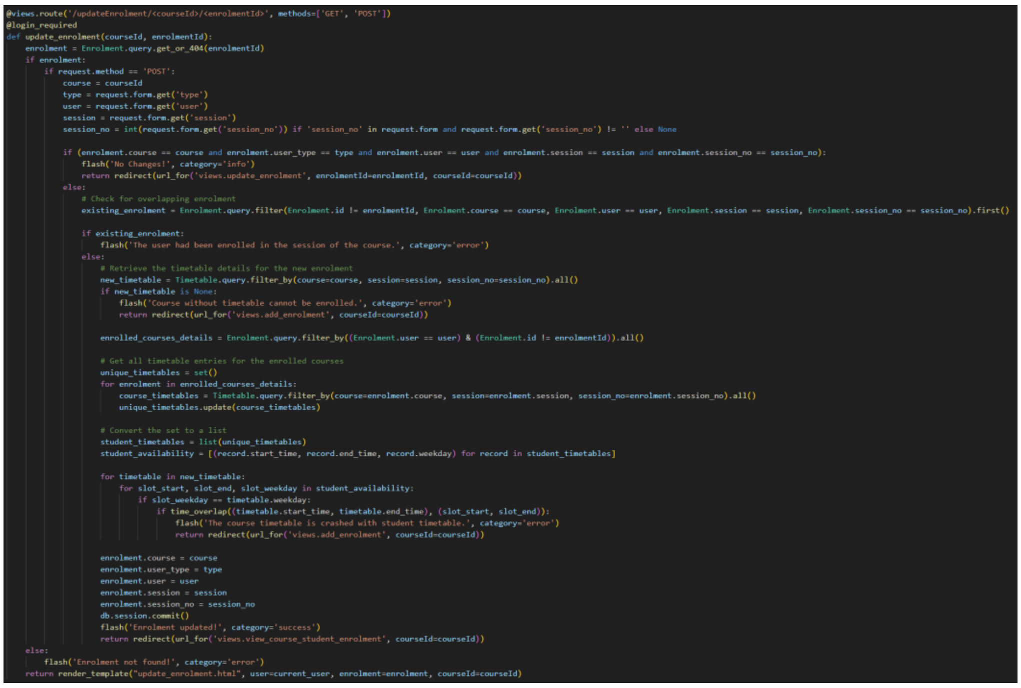


```python
@views.route('/viewCourseStudentEnrolment/<courseId>')
@login_required
def view_course_student_enrolment(courseId):
    course = Course.query.get_or_404(courseId)

    page = request.args.get('page', 1, type=int)
    per_page = 6
    enrolments = Enrolment.query.filter_by(course=courseId, user_type='Student')
    enrolments = enrolments.order_by(asc(Enrolment.user))
    enrolments = enrolments.order_by(asc(Enrolment.session))
    enrolments = enrolments.order_by(asc(Enrolment.session_no)).paginate(page=page, per_page=per_page)
    return render_template("view_enrolment.html", user=current_user, enrolment=enrolments, courseId=courseId, courseName=course.info)

@views.route('/viewCourseLecturerEnrolment/<courseId>')
@login_required
def view_course_lecturer_enrolment(courseId):
    course = Course.query.get_or_404(courseId)

    page = request.args.get('page', 1, type=int)
    per_page = 6
    enrolments = Enrolment.query.filter_by(course=courseId, user_type='Lecturer')
    enrolments = enrolments.order_by(asc(Enrolment.user))
    enrolments = enrolments.order_by(asc(Enrolment.session))
    enrolments = enrolments.order_by(asc(Enrolment.session_no)).paginate(page=page, per_page=per_page)
    return render_template("view_enrolment.html", user=current_user, enrolment=enrolments, courseId=courseId, courseName=course.info)
```

Figure 6.68: Implementation Code for Viewing Course Enrolment

The 'view_course_student_enrolment' and 'view_course_lecturer_enrolment' functions provide users with comprehensive views of course enrolments for both students and lecturers respectively. By leveraging pagination techniques, these functions present enrolment data in a structured format which allow administrator to access essential information such as user Id and session details. By offering a user-friendly interface, these functions enhance accessibility and streamline the monitoring of enrolment activities.

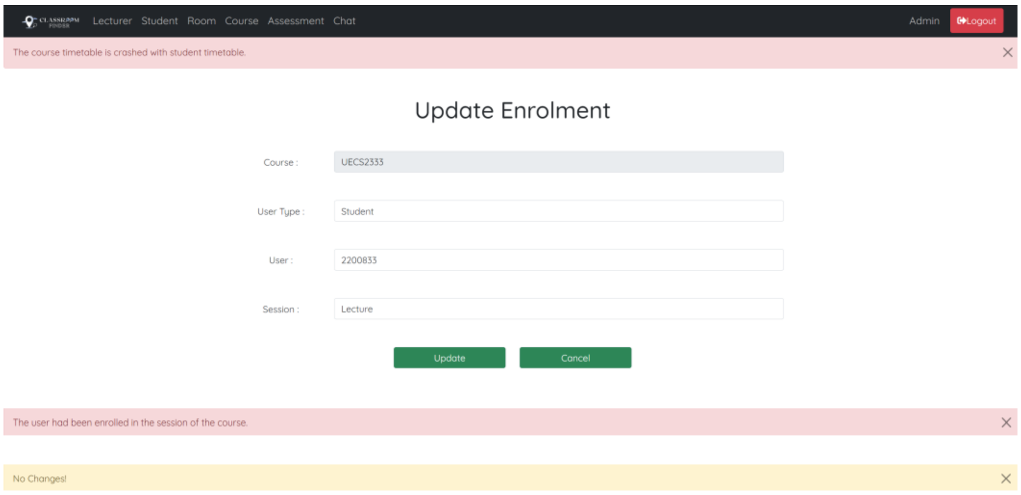
6.8.3 Update Course Enrolment



```python
@views.route('/updateEnrolment/<courseId>/<enrolmentId>', methods=['GET', 'POST'])
@login_required
def update_enrolment(courseId, enrolmentId):
    enrolment = Enrolment.query.get_or_404(enrolmentId)
    if enrolment:
        if request.method == 'POST':
            course = courseId
            type = request.form.get('type')
            user = request.form.get('user')
            session = request.form.get('session')
            session_no = int(request.form.get('session_no')) if 'session_no' in request.form and request.form.get('session_no') != '' else None

            if (enrolment.course == course and enrolment.user_type == type and enrolment.user == user and enrolment.session == session and enrolment.session_no == session_no):
                flash('No Changes!', category='info')
                return redirect(url_for('views.update_enrolment', enrolmentId=enrolmentId, courseId=courseId))
            else:
                # Check for overlapping enrolment
                existing_enrolment = Enrolment.query.filter(Enrolment.id != enrolmentId, Enrolment.course == course, Enrolment.user == user, Enrolment.session == session, Enrolment.session_no == session_no).first()

                if existing_enrolment:
                    flash('The user had been enrolled in the session of the course.', category='error')
                else:
                    # Retrieve the timetable details for the new enrolment
                    new_timetable = Timetable.query.filter_by(course=course, session=session, session_no=session_no).all()
                    if new_timetable is None:
                        flash('Course without timetable cannot be enrolled.', category='error')
                        return redirect(url_for('views.add_enrolment', courseId=courseId))

                    enrolled_courses_details = Enrolment.query.filter_by((Enrolment.user == user) & (Enrolment.id != enrolmentId)).all()

                    # Get all timetable entries for the enrolled courses
                    unique_timetables = set()
                    for enrolment in enrolled_courses_details:
                        course_timetables = Timetable.query.filter_by(course=enrolment.course, session=enrolment.session, session_no=enrolment.session_no).all()
                        unique_timetables.update(course_timetables)

                    # Convert the set to a list
                    student_timetables = list(unique_timetables)
                    student_availability = [(record.start_time, record.end_time, record.weekday) for record in student_timetables]

                    for timetable in new_timetable:
                        for slot_start, slot_end, slot_weekday in student_availability:
                            if slot_weekday == timetable.weekday:
                                if time_overlap((timetable.start_time, timetable.end_time), (slot_start, slot_end)):
                                    flash('The course timetable is crashed with student timetable.', category='error')
                                    return redirect(url_for('views.add_enrolment', courseId=courseId))

                    enrolment.course = course
                    enrolment.user_type = type
                    enrolment.user = user
                    enrolment.session = session
                    enrolment.session_no = session_no
                    db.session.commit()
                    flash('Enrolment updated!', category='success')
                    return redirect(url_for('views.view_course_student_enrolment', courseId=courseId))
    else:
        flash('Enrolment not found!', category='error')
    return render_template("update_enrolment.html", user=current_user, enrolment=enrolment, courseId=courseId)
```

Figure 6.69: Implementation Code for Updating Course Timetable

The 'update_enrolment' function which accessible via the '/updateEnrolment/< courseId>/< enrolmentId>' route, allows administrator to update existing course enrolment details for the particular course. The administrator can access the update course enrolment form of a specific course by providing the courseId and enrolmentId in the route. The administrator can update the course enrolment by submitting the course enrolment details via the HTML form. In the HTML form, the 'find_users' and 'find_session' functions also enhance the user experience by providing dynamic search capabilities which allow administrators to quickly locate users or session no based on specific criteria. Before submitting the course enrolment details, all the input values are required to be filled in to perform the enrolment updating process. If there is any input field which has empty value, the system will display the error message to notify the administrator when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



The course timetable is crashed with student timetable.

Update Enrolment

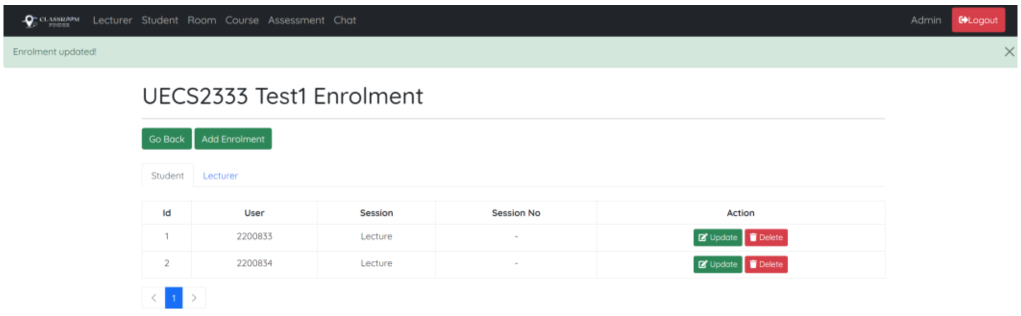
Course :	UECS2333
User Type :	Student
User :	2200833
Session :	Lecture

The user had been enrolled in the session of the course.

No Changes!

Figure 6.70: Update Failed with Error Messages

Upon receiving a POST request containing the necessary information which are user and session details from the update enrolment form, the function retrieves the course enrolment record associated with the provided Id and validates the submitted changes. The system employs sophisticated algorithms to detect and prevent conflicts such as duplicate enrolments or overlapping schedules, thereby safeguarding against inadvertent errors and ensuring the accuracy of updated enrolment entries.



Enrolment updated!

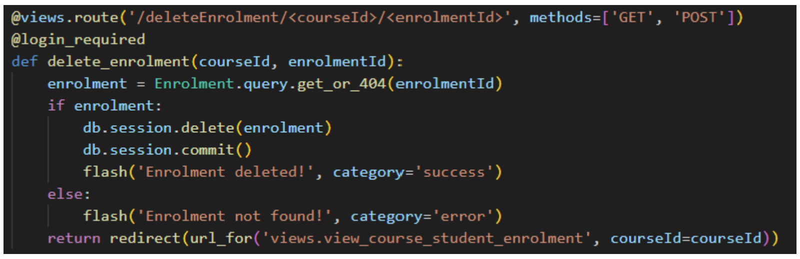
UECS2333 Test1 Enrolment

Id	User	Session	Session No	Action
1	2200833	Lecture	-	Update Delete
2	2200834	Lecture	-	Update Delete

Figure 6.71: Update Successfully with Success Messages

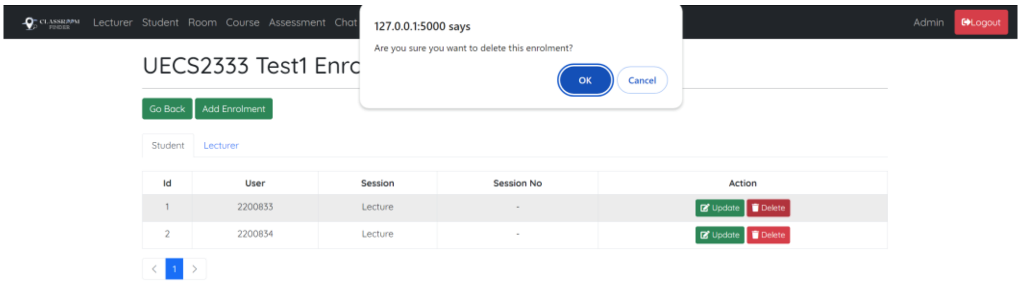
If the submitted data is valid and differs from the existing record, the course enrolment details are updated in the database and the administrator is redirected to the view course enrolment page with a success message.

6.8.4 Delete Course Enrolment



```
@views.route('/deleteEnrolment/<courseId>/<enrolmentId>', methods=['GET', 'POST'])
@login_required
def delete_enrolment(courseId, enrolmentId):
    enrolment = Enrolment.query.get_or_404(enrolmentId)
    if enrolment:
        db.session.delete(enrolment)
        db.session.commit()
        flash('Enrolment deleted!', category='success')
    else:
        flash('Enrolment not found!', category='error')
    return redirect(url_for('views.view_course_student_enrolment', courseId=courseId))
```

Figure 6.72: Implementation Code for Deleting Course Enrolment



127.0.0.1:5000 says

Are you sure you want to delete this enrolment?

OK Cancel

UECS2333 Test1 Enro

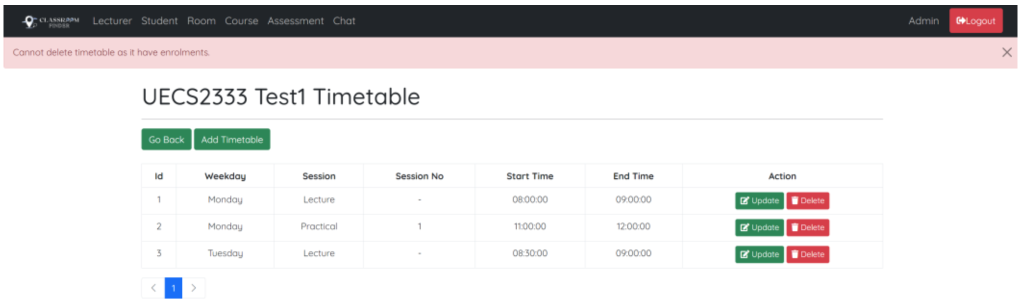
Go Back Add Enrolment

Student Lecturer

Id	User	Session	Session No	Action
1	2200833	Lecture	-	Update Delete
2	2200834	Lecture	-	Update Delete

Figure 6.73: Confirmation Prompt before Deleting Course Enrolment

To maintain data consistency and optimize resource allocation, the 'delete_enrolment' function incorporates robust validation mechanisms to prevent the deletion of enrolments associated with active course sessions. By enforcing these safeguards, the system mitigates the risk of data loss or disruption to ongoing academic activities to foster a reliable and resilient enrolment environment.



Cannot delete timetable as it have enrolments.

UECS2333 Test1 Timetable

Id	Weekday	Session	Session No	Start Time	End Time	Action
1	Monday	Lecture	-	08:00:00	09:00:00	Update Delete
2	Monday	Practical	1	11:00:00	12:00:00	Update Delete
3	Tuesday	Lecture	-	08:30:00	09:00:00	Update Delete

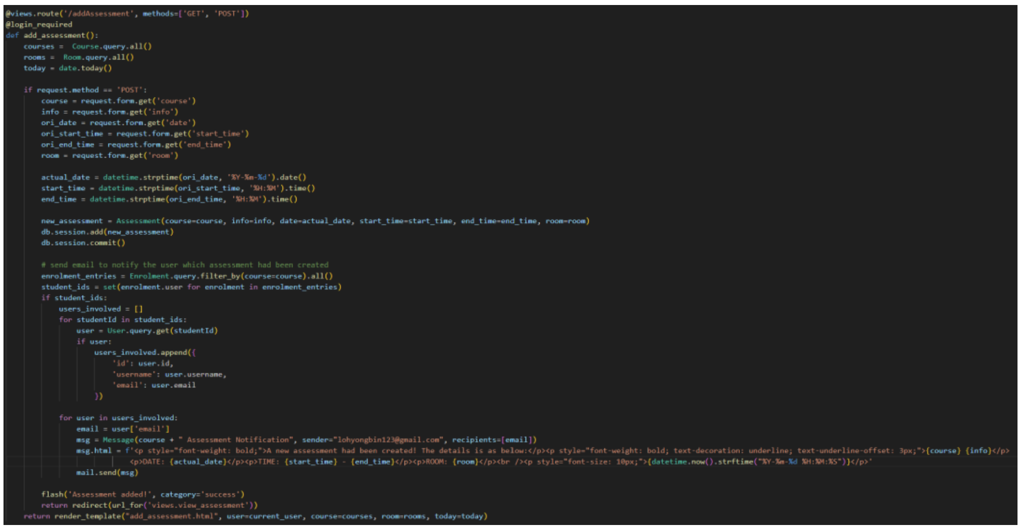
Figure 6.74: Delete Failed with Error Messages

To maintain data consistency and optimize resource allocation, the 'delete_timetable' function also incorporates robust validation mechanisms to prevent the deletion of course timetables associated with active enrolments or sessions. By enforcing these safeguards, the system mitigates the risk of data loss or disruption to ongoing academic activities to foster a reliable and resilient scheduling environment.

6.9 Assessment Management, Room Allocation/ Find Room and Notification Modules

The assessment management module is a pivotal component of the Classroom Finder System which designed to facilitate the creation or scheduling, view, update, delete the assessments within academic courses. During the process of creating and updating the assessment, the process is requiring the find room module to find the room for creating and updating the assessment. Therefore, the room allocation or find room module is included in this section. During the medication of the assessment, the system will also notify the users about the assessment details. Therefore, the notification module is also included in this section. These modules encompass a suite of functionalities tailored to streamline assessment processes, ensure timely communication with stakeholders and enhance overall administrative efficiency.

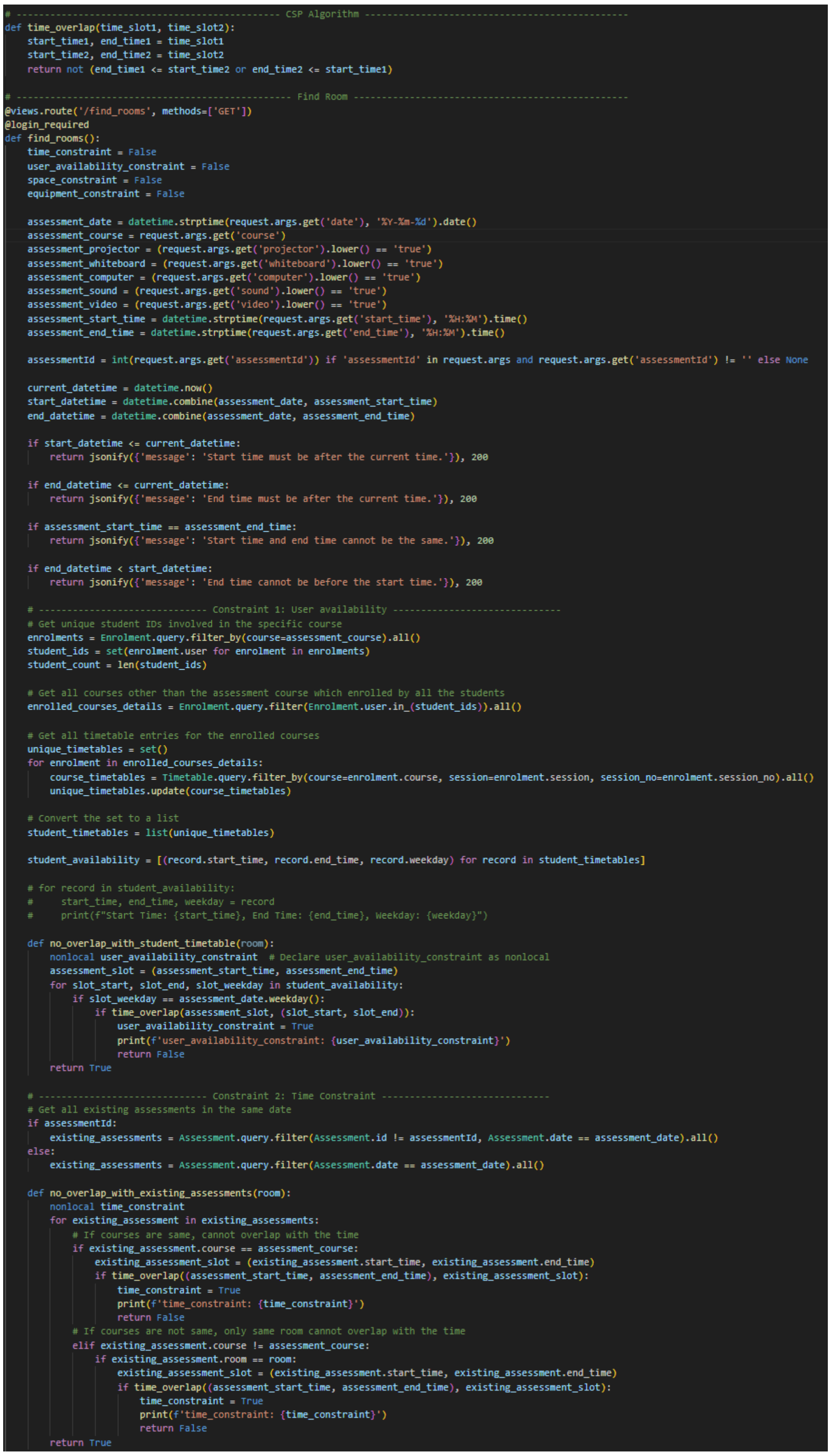
6.9.1 Add Assessment



```python
@views.route('/addAssessment', methods=['GET', 'POST'])
@login_required
def add_assessment():
    courses =  Course.query.all()
    rooms =  Room.query.all()
    today = date.today()

    if request.method == 'POST':
        course = request.form.get('course')
        info = request.form.get('info')
        ori_date = request.form.get('date')
        ori_start_time = request.form.get('start_time')
        ori_end_time = request.form.get('end_time')
        room = request.form.get('room')

        actual_date = datetime.strptime(ori_date, '%Y-%m-%d').date()
        start_time = datetime.strptime(ori_start_time, '%H:%M').time()
        end_time = datetime.strptime(ori_end_time, '%H:%M').time()

        new_assessment = Assessment(course=course, info=info, date=actual_date, start_time=start_time, end_time=end_time, room=room)
        db.session.add(new_assessment)
        db.session.commit()

        # send email to notify the user which assessment had been created
        enrolment_entries = Enrolment.query.filter_by(course=course).all()
        student_ids = set(enrolment.user for enrolment in enrolment_entries)
        if student_ids:
            users_involved = []
            for studentId in student_ids:
                user = User.query.get(studentId)
                if user:
                    users_involved.append({
                        'id': user.id,
                        'username': user.username,
                        'email': user.email
                    })

            for user in users_involved:
                email = user['email']
                msg = Message(course + " Assessment Notification", sender="lohyongbin123@gmail.com", recipients=[email])
                msg.html = f'<p style="font-weight: bold;">A new assessment had been created! The details is as below:</p><p style="font-weight: bold; text-decoration: underline; text-underline-offset: 3px;">{course} {info}</p>
                        <p>DATE: {actual_date}</p><p>TIME: {start_time} - {end_time}</p><p>ROOM: {room}</p><br /><p style="font-size: 10px;">{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}</p>'
                mail.send(msg)

        flash('Assessment added!', category='success')
        return redirect(url_for('views.view_assessment'))
    return render_template("add_assessment.html", user=current_user, course=courses, room=rooms, today=today)
```

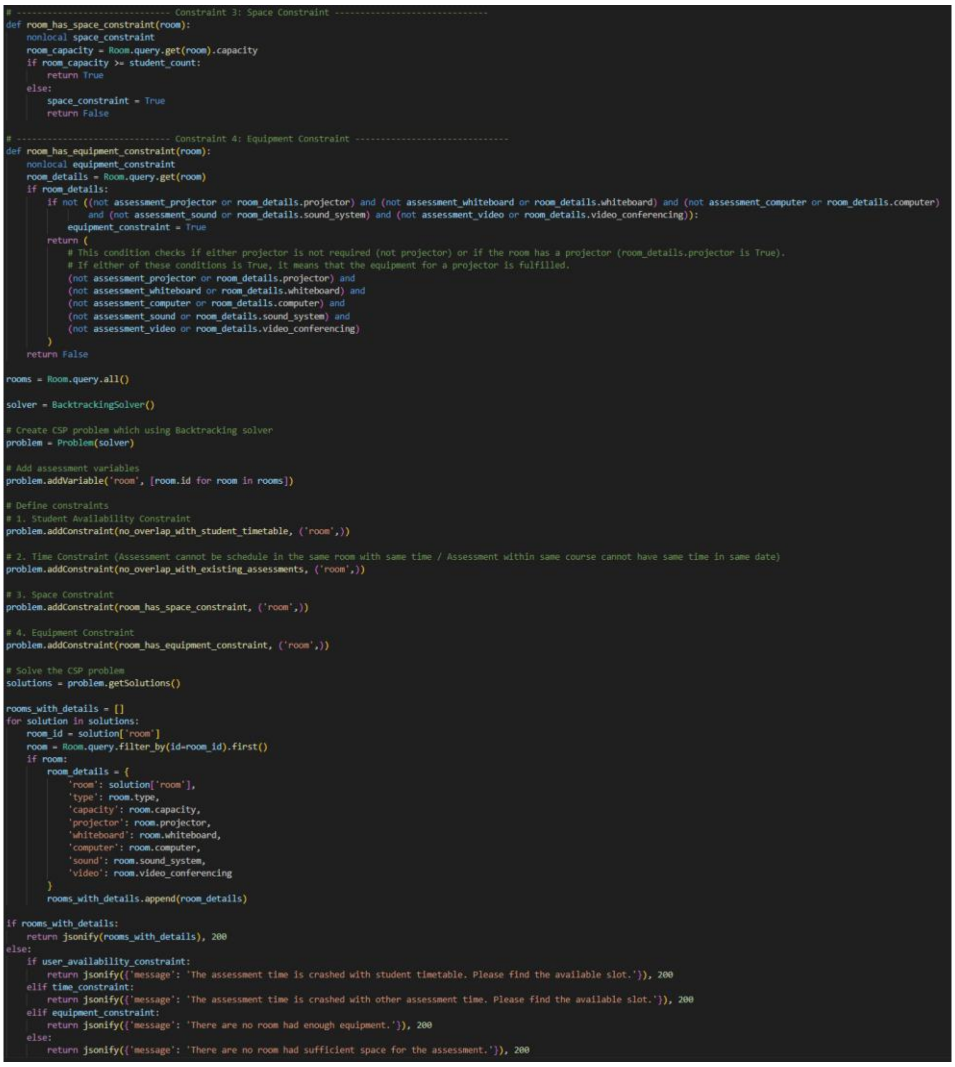
Figure 6.75: Implementation Code for Adding Assessment

The 'add_assessment' function empowers administrators to create new assessments with ease, providing intuitive form-based inputs for specifying assessment details such as course, assessment info, date, time, and room venue. It allows the administrator to submit the assessment details via a HTML form.



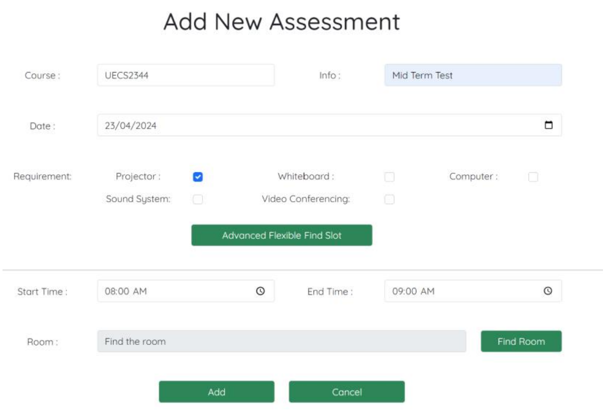
```python
# ---------------------------------------------- CSP Algorithm ----------------------------------------------
def time_overlap(time_slot1, time_slot2):
    start_time1, end_time1 = time_slot1
    start_time2, end_time2 = time_slot2
    return not (end_time1 <= start_time2 or end_time2 <= start_time1)

# ------------------------------------------------ Find Room ------------------------------------------------
@views.route('/find_rooms', methods=['GET'])
@login_required
def find_rooms():
    time_constraint = False
    user_availability_constraint = False
    space_constraint = False
    equipment_constraint = False

    assessment_date = datetime.strptime(request.args.get('date'), '%Y-%m-%d').date()
    assessment_course = request.args.get('course')
    assessment_projector = (request.args.get('projector').lower() == 'true')
    assessment_whiteboard = (request.args.get('whiteboard').lower() == 'true')
    assessment_computer = (request.args.get('computer').lower() == 'true')
    assessment_sound = (request.args.get('sound').lower() == 'true')
    assessment_video = (request.args.get('video').lower() == 'true')
    assessment_start_time = datetime.strptime(request.args.get('start_time'), '%H:%M').time()
    assessment_end_time = datetime.strptime(request.args.get('end_time'), '%H:%M').time()

    assessmentId = int(request.args.get('assessmentId')) if 'assessmentId' in request.args and request.args.get('assessmentId') != '' else None

    current_datetime = datetime.now()
    start_datetime = datetime.combine(assessment_date, assessment_start_time)
    end_datetime = datetime.combine(assessment_date, assessment_end_time)

    if start_datetime <= current_datetime:
        return jsonify({'message': 'Start time must be after the current time.'}), 200

    if end_datetime <= current_datetime:
        return jsonify({'message': 'End time must be after the current time.'}), 200

    if assessment_start_time == assessment_end_time:
        return jsonify({'message': 'Start time and end time cannot be the same.'}), 200

    if end_datetime < start_datetime:
        return jsonify({'message': 'End time cannot be before the start time.'}), 200

    # ------------------------------ Constraint 1: User availability ------------------------------
    # Get unique student IDs involved in the specific course
    enrolments = Enrolment.query.filter_by(course=assessment_course).all()
    student_ids = set(enrolment.user for enrolment in enrolments)
    student_count = len(student_ids)

    # Get all courses other than the assessment course which enrolled by all the students
    enrolled_courses_details = Enrolment.query.filter(Enrolment.user.in_(student_ids)).all()

    # Get all timetable entries for the enrolled courses
    unique_timetables = set()
    for enrolment in enrolled_courses_details:
        course_timetables = Timetable.query.filter_by(course=enrolment.course, session=enrolment.session, session_no=enrolment.session_no).all()
        unique_timetables.update(course_timetables)

    # Convert the set to a list
    student_timetables = list(unique_timetables)

    student_availability = [(record.start_time, record.end_time, record.weekday) for record in student_timetables]

    # for record in student_availability:
    #     start_time, end_time, weekday = record
    #     print(f"Start Time: {start_time}, End Time: {end_time}, Weekday: {weekday}")

    def no_overlap_with_student_timetable(room):
        nonlocal user_availability_constraint  # Declare user_availability_constraint as nonlocal
        assessment_slot = (assessment_start_time, assessment_end_time)
        for slot_start, slot_end, slot_weekday in student_availability:
            if slot_weekday == assessment_date.weekday():
                if time_overlap(assessment_slot, (slot_start, slot_end)):
                    user_availability_constraint = True
                    print(f'user_availability_constraint: {user_availability_constraint}')
                    return False
        return True

    # ------------------------------ Constraint 2: Time Constraint ------------------------------
    # Get all existing assessments in the same date
    if assessmentId:
        existing_assessments = Assessment.query.filter(Assessment.id != assessmentId, Assessment.date == assessment_date).all()
    else:
        existing_assessments = Assessment.query.filter(Assessment.date == assessment_date).all()

    def no_overlap_with_existing_assessments(room):
        nonlocal time_constraint
        for existing_assessment in existing_assessments:
            # If courses are same, cannot overlap with the time
            if existing_assessment.course == assessment_course:
                existing_assessment_slot = (existing_assessment.start_time, existing_assessment.end_time)
                if time_overlap((assessment_start_time, assessment_end_time), existing_assessment_slot):
                    time_constraint = True
                    print(f'time_constraint: {time_constraint}')
                    return False
            # If courses are not same, only same room cannot overlap with the time
            elif existing_assessment.course != assessment_course:
                if existing_assessment.room == room:
                    existing_assessment_slot = (existing_assessment.start_time, existing_assessment.end_time)
                    if time_overlap((assessment_start_time, assessment_end_time), existing_assessment_slot):
                        time_constraint = True
                        print(f'time_constraint: {time_constraint}')
                        return False
        return True
```

Figure 6.76: Implementation Code for Finding Room by Using Constraint Satisfaction Problem AI Technique and Backtracking Algorithm (Part A)



```python
# ------------------------------ Constraint 3: Space Constraint ------------------------------
def room_has_space_constraint(room):
    nonlocal space_constraint
    room_capacity = Room.query.get(room).capacity
    if room_capacity >= student_count:
        return True
    else:
        space_constraint = True
        return False

# ------------------------------ Constraint 4: Equipment Constraint ------------------------------
def room_has_equipment_constraint(room):
    nonlocal equipment_constraint
    room_details = Room.query.get(room)
    if room_details:
        if not ((not assessment_projector or room_details.projector) and (not assessment_whiteboard or room_details.whiteboard) and (not assessment_computer or room_details.computer)
                and (not assessment_sound or room_details.sound_system) and (not assessment_video or room_details.video_conferencing)):
            equipment_constraint = True
        return (
            # This condition checks if either projector is not required (not projector) or if the room has a projector (room_details.projector is True).
            # If either of these conditions is True, it means that the equipment for a projector is fulfilled.
            (not assessment_projector or room_details.projector) and
            (not assessment_whiteboard or room_details.whiteboard) and
            (not assessment_computer or room_details.computer) and
            (not assessment_sound or room_details.sound_system) and
            (not assessment_video or room_details.video_conferencing)
        )
    return False

rooms = Room.query.all()

solver = BacktrackingSolver()

# Create CSP problem which using Backtracking solver
problem = Problem(solver)

# Add assessment variables
problem.addVariable('room', [room.id for room in rooms])

# Define constraints
# 1. Student Availability Constraint
problem.addConstraint(no_overlap_with_student_timetable, ('room',))

# 2. Time Constraint (Assessment cannot be schedule in the same room with same time / Assessment within same course cannot have same time in same date)
problem.addConstraint(no_overlap_with_existing_assessments, ('room',))

# 3. Space Constraint
problem.addConstraint(room_has_space_constraint, ('room',))

# 4. Equipment Constraint
problem.addConstraint(room_has_equipment_constraint, ('room',))

# Solve the CSP problem
solutions = problem.getSolutions()

rooms_with_details = []
for solution in solutions:
    room_id = solution['room']
    room = Room.query.filter_by(id=room_id).first()
    if room:
        room_details = {
            'room': solution['room'],
            'type': room.type,
            'capacity': room.capacity,
            'projector': room.projector,
            'whiteboard': room.whiteboard,
            'computer': room.computer,
            'sound': room.sound_system,
            'video': room.video_conferencing
        }
        rooms_with_details.append(room_details)

if rooms_with_details:
    return jsonify(rooms_with_details), 200
else:
    if user_availability_constraint:
        return jsonify({'message': 'The assessment time is crashed with student timetable. Please find the available slot.'}), 200
    elif time_constraint:
        return jsonify({'message': 'The assessment time is crashed with other assessment time. Please find the available slot.'}), 200
    elif equipment_constraint:
        return jsonify({'message': 'There are no room had enough equipment.'}), 200
    else:
        return jsonify({'message': 'There are no room had sufficient space for the assessment.'}), 200
```

Figure 6.77: Implementation Code for Finding Room by Using Constraint Satisfaction Problem AI Technique and Backtracking Algorithm (Part B)



Add New Assessment

Course : UECS2344 Info : Mid Term Test

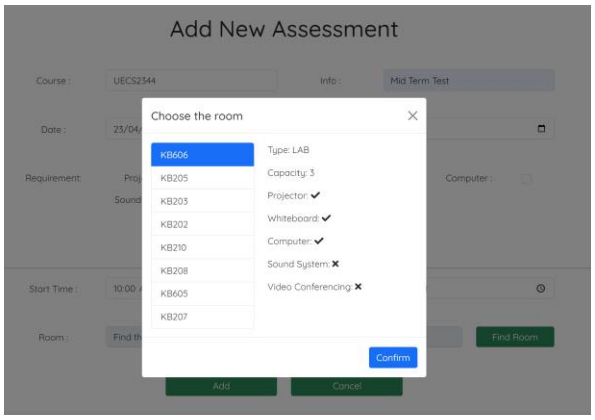
Date : 23/04/2024

Requirement: Projector : [✓] Whiteboard : [ ] Computer : [ ]
Sound System: [ ] Video Conferencing: [ ]

Start Time : 08:00 AM End Time : 09:00 AM

Room : Find the room

Figure 6.78: Find Room


Add New Assessment

Course : UECS2344
Info : Mid Term Test
Date : 23/04/
Requirement: Proj Sound Computer :
Start Time : 10:00
Room : Find th

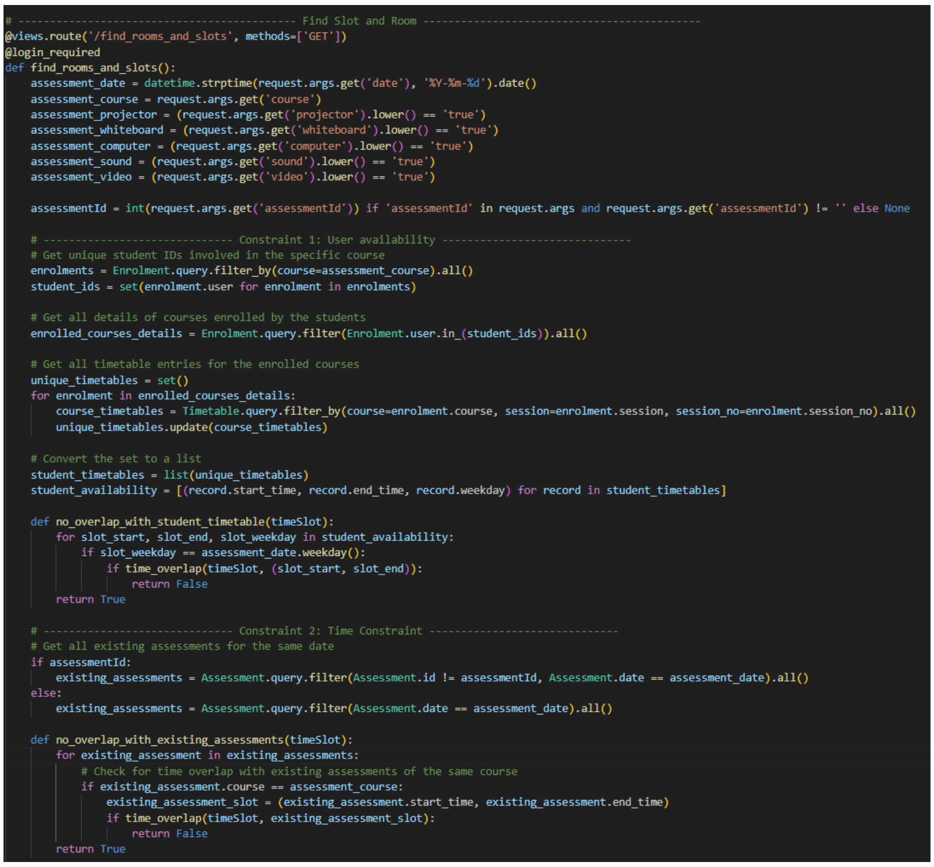
Choose the room

KB606
KB205
KB203
KB202
KB210
KB208
KB605
KB207

Type: LAB
Capacity: 3
Projector: ✓
Whiteboard: ✓
Computer: ✓
Sound System: ✗
Video Conferencing: ✗

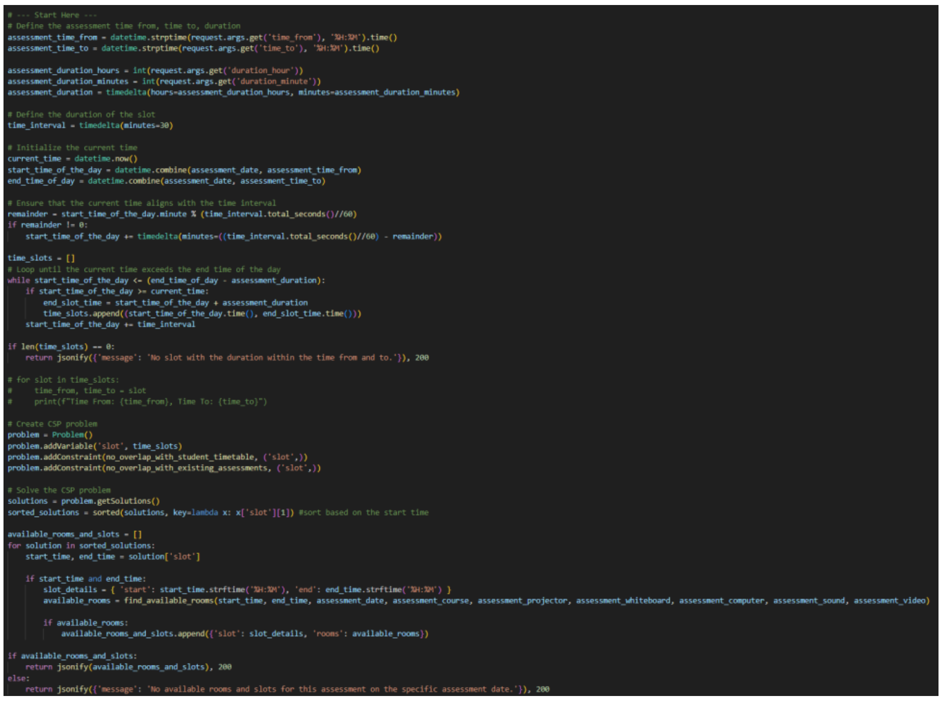
Figure 6.79: Find Room Result

In the HTML form, the system revolves around managing assessments which include finding suitable rooms for conducting the assessments. The find_rooms function is one of the key components of the assessment management system. When the user clicks the “Find Room” button, the find_rooms function is triggered to find available rooms for scheduling an assessment. It takes into account several constraints which include user availability, time conflicts with existing assessments, space limitations and equipment requirements such as projectors, whiteboards, computers, sound systems and video conferencing facilities. It first gathers information about the assessment such as date, course, required equipment and start and end times. After that, user availability is checked against existing timetables to ensure that the assessment does not conflict with students' schedules. Next, time constraints ensure that the assessment does not overlap with existing assessments on the same date and that rooms are available during the specified time slot. Space constraints also verify that the selected rooms can accommodate the number of students enrolled in the course. Lastly, equipment constraints ensure that the chosen rooms have the necessary equipment for the assessment. The function is using Constraint Satisfaction Problem (CSP) AI techniques which utilize the backtracking algorithm to find suitable solutions that satisfy all constraints. Once solutions are found, the function returns a list of available rooms that meet the criteria, or it provides appropriate error messages if constraints cannot be satisfied. The sample result can be referred by figure 6.78: find room result.



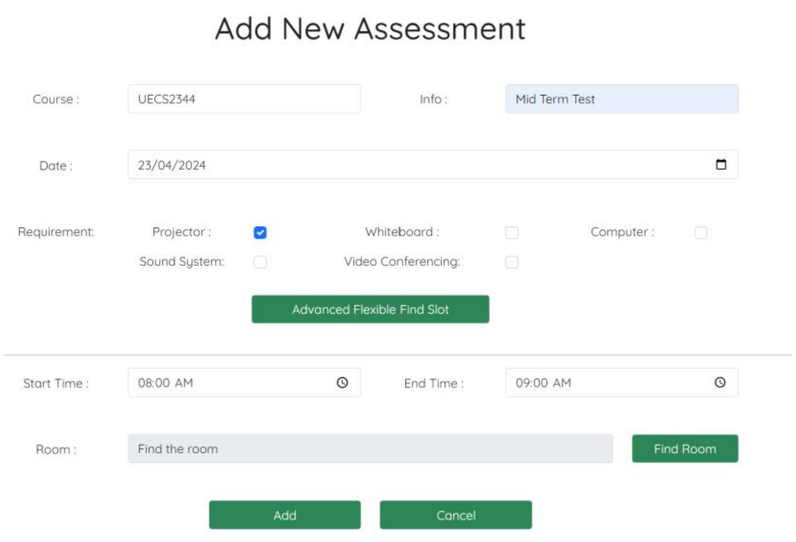
```python
# ------------------------------------------- Find Slot and Room -------------------------------------------
@views.route('/find_rooms_and_slots', methods=['GET'])
@login_required
def find_rooms_and_slots():
    assessment_date = datetime.strptime(request.args.get('date'), '%Y-%m-%d').date()
    assessment_course = request.args.get('course')
    assessment_projector = (request.args.get('projector').lower() == 'true')
    assessment_whiteboard = (request.args.get('whiteboard').lower() == 'true')
    assessment_computer = (request.args.get('computer').lower() == 'true')
    assessment_sound = (request.args.get('sound').lower() == 'true')
    assessment_video = (request.args.get('video').lower() == 'true')

    assessmentId = int(request.args.get('assessmentId')) if 'assessmentId' in request.args and request.args.get('assessmentId') != '' else None

    # ---------------------------- Constraint 1: User availability ----------------------------
    # Get unique student IDs involved in the specific course
    enrolments = Enrolment.query.filter_by(course=assessment_course).all()
    student_ids = set(enrolment.user for enrolment in enrolments)

    # Get all details of courses enrolled by the students
    enrolled_courses_details = Enrolment.query.filter(Enrolment.user.in_(student_ids)).all()

    # Get all timetable entries for the enrolled courses
    unique_timetables = set()
    for enrolment in enrolled_courses_details:
        course_timetables = Timetable.query.filter_by(course=enrolment.course, session=enrolment.session, session_no=enrolment.session_no).all()
        unique_timetables.update(course_timetables)

    # Convert the set to a list
    student_timetables = list(unique_timetables)
    student_availability = [(record.start_time, record.end_time, record.weekday) for record in student_timetables]

    def no_overlap_with_student_timetable(timeSlot):
        for slot_start, slot_end, slot_weekday in student_availability:
            if slot_weekday == assessment_date.weekday():
                if time_overlap(timeSlot, (slot_start, slot_end)):
                    return False
        return True

    # ---------------------------- Constraint 2: Time Constraint ----------------------------
    # Get all existing assessments for the same date
    if assessmentId:
        existing_assessments = Assessment.query.filter(Assessment.id != assessmentId, Assessment.date == assessment_date).all()
    else:
        existing_assessments = Assessment.query.filter(Assessment.date == assessment_date).all()

    def no_overlap_with_existing_assessments(timeSlot):
        for existing_assessment in existing_assessments:
            # Check for time overlap with existing assessments of the same course
            if existing_assessment.course == assessment_course:
                existing_assessment_slot = (existing_assessment.start_time, existing_assessment.end_time)
                if time_overlap(timeSlot, existing_assessment_slot):
                    return False
        return True
```

Figure 6.80: Implementation Code for Advance Finding Slot and Room by Using Constraint Satisfaction Problem Algorithm (Part A)



```python
# --- Start Here ---
# Define the assessment time from, time to, duration
assessment_time_from = datetime.strptime(request.args.get('time_from'), '%H:%M').time()
assessment_time_to = datetime.strptime(request.args.get('time_to'), '%H:%M').time()

assessment_duration_hours = int(request.args.get('duration_hour'))
assessment_duration_minutes = int(request.args.get('duration_minute'))
assessment_duration = timedelta(hours=assessment_duration_hours, minutes=assessment_duration_minutes)

# Define the duration of the slot
time_interval = timedelta(minutes=30)

# Initialize the current time
current_time = datetime.now()
start_time_of_the_day = datetime.combine(assessment_date, assessment_time_from)
end_time_of_day = datetime.combine(assessment_date, assessment_time_to)

# Ensure that the current time aligns with the time interval
remainder = start_time_of_the_day.minute % (time_interval.total_seconds()//60)
if remainder != 0:
    start_time_of_the_day += timedelta(minutes=((time_interval.total_seconds()//60) - remainder))

time_slots = []
# Loop until the current time exceeds the end time of the day
while start_time_of_the_day <= (end_time_of_day - assessment_duration):
    if start_time_of_the_day >= current_time:
        end_slot_time = start_time_of_the_day + assessment_duration
        time_slots.append((start_time_of_the_day.time(), end_slot_time.time()))
    start_time_of_the_day += time_interval

if len(time_slots) == 0:
    return jsonify({'message': 'No slot with the duration within the time from and to.'}), 200

# for slot in time_slots:
#     time_from, time_to = slot
#     print(f"Time From: {time_from}, Time To: {time_to}")

# Create CSP problem
problem = Problem()
problem.addVariable('slot', time_slots)
problem.addConstraint(no_overlap_with_student_timetable, ('slot',))
problem.addConstraint(no_overlap_with_existing_assessments, ('slot',))

# Solve the CSP problem
solutions = problem.getSolutions()
sorted_solutions = sorted(solutions, key=lambda x: x['slot'][1]) #sort based on the start time

available_rooms_and_slots = []
for solution in sorted_solutions:
    start_time, end_time = solution['slot']

    if start_time and end_time:
        slot_details = { 'start': start_time.strftime('%H:%M'), 'end': end_time.strftime('%H:%M') }
        available_rooms = find_available_rooms(start_time, end_time, assessment_date, assessment_course, assessment_projector, assessment_whiteboard, assessment_computer, assessment_sound, assessment_video)

        if available_rooms:
            available_rooms_and_slots.append({'slot': slot_details, 'rooms': available_rooms})

if available_rooms_and_slots:
    return jsonify(available_rooms_and_slots), 200
else:
    return jsonify({'message': 'No available rooms and slots for this assessment on the specific assessment date.'}), 200
```

Figure 6.81: Implementation Code for Advance Finding Slot and Room by Using Constraint Satisfaction Problem Algorithm (Part B)



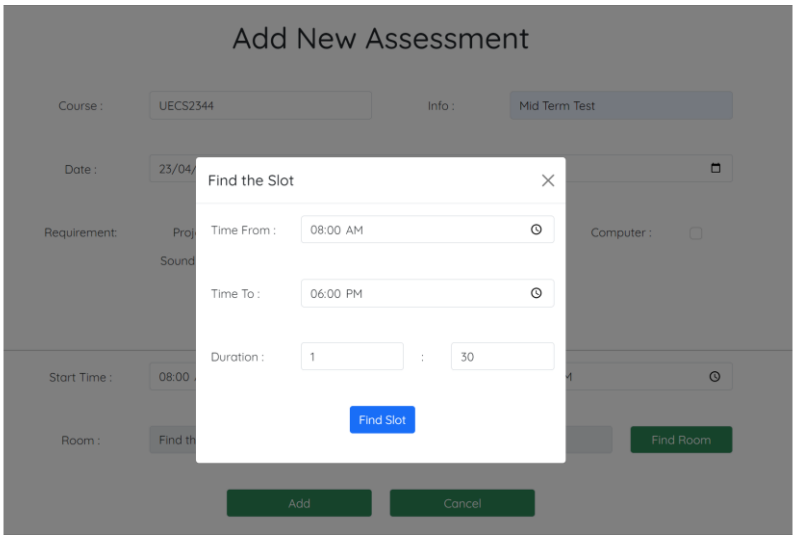
Add New Assessment

Course :	UECS2344	Info :	Mid Term Test
Date :	23/04/2024		

Requirement: Projector : ✓ Whiteboard : ☐ Computer : ☐ Sound System: ☐ Video Conferencing: ☐

Start Time :	08:00 AM	End Time :	09:00 AM
Room :	Find the room		

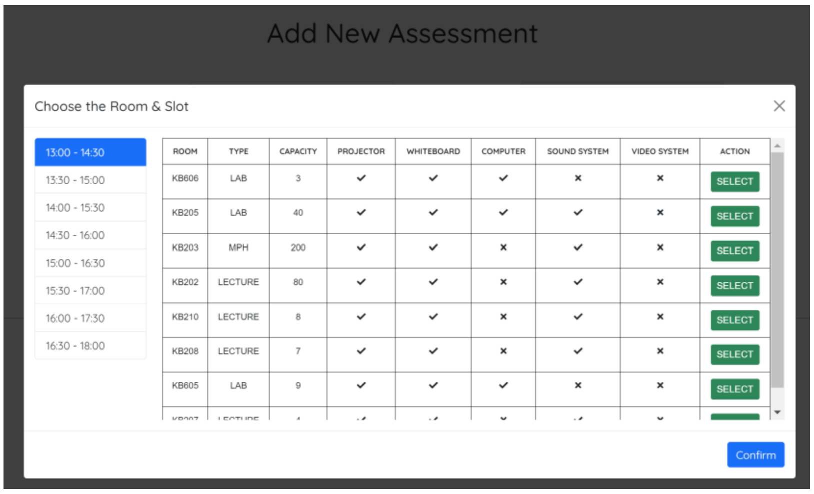
Figure 6.82: Advance Flexible Find Slot and Room (Part A)



Find the Slot

Time From :	08:00 AM		
Time To :	06:00 PM		
Duration :	1	:	30

Figure 6.83: Advance Flexible Find Slot and Room (Part B)



Add New Assessment

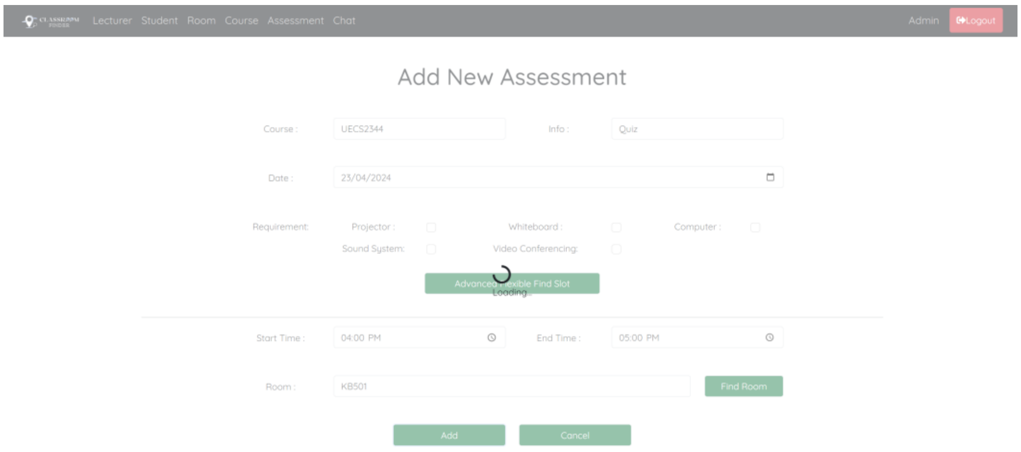
Choose the Room & Slot

13:00 - 14:30
13:30 - 15:00
14:00 - 15:30
14:30 - 16:00
15:00 - 16:30
15:30 - 17:00
16:00 - 17:30
16:30 - 18:00

ROOM	TYPE	CAPACITY	PROJECTOR	WHITEBOARD	COMPUTER	SOUND SYSTEM	VIDEO SYSTEM	ACTION
KB606	LAB	3	✓	✓	✓	×	×	SELECT
KB205	LAB	40	✓	✓	✓	✓	×	SELECT
KB203	MPH	200	✓	✓	×	✓	×	SELECT
KB202	LECTURE	80	✓	✓	×	✓	×	SELECT
KB210	LECTURE	8	✓	✓	×	✓	×	SELECT
KB208	LECTURE	7	✓	✓	×	✓	×	SELECT
KB605	LAB	9	✓	✓	✓	×	×	SELECT

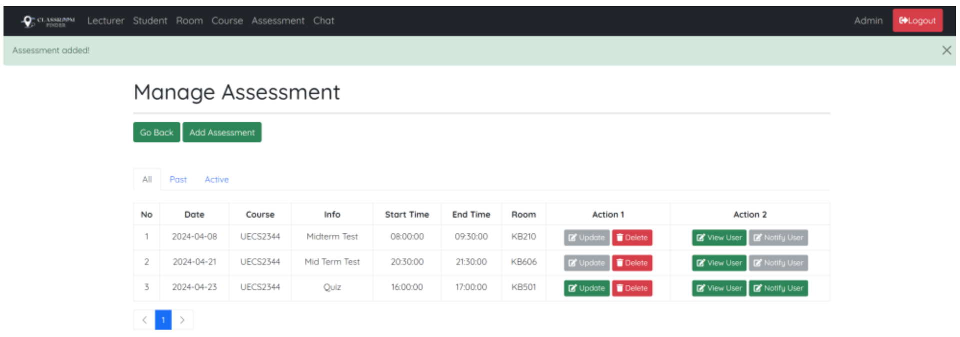
Figure 6.84: Advance Flexible Find Slot and Room Result

In the HTML form, the system revolves around managing assessments which also include advance finding suitable slots and rooms for conducting the assessments. The find_rooms_and_slots function is similar to find_rooms but it focuses on finding available time slots in addition to suitable rooms. It considers the same constraints as the find_rooms function but also allows the user to specify a desired assessment duration. It will calculate available time slots within the specified time range which help to ensure that they do not conflict with student timetables or existing assessments. Like find_rooms function, it utilizes CSP techniques to find feasible solutions and sort them based on start time for better readability. After that, it returns a list of available time slots along with the corresponding rooms that meet all constraints. If no suitable rooms or time slots are found, the function will provide appropriate error messages to the user. Before submitting the assessment details to the backend, all the input values are required to be filled in to perform the assessment creation process. If there is any input field which has empty value, the system will display the error message to notify the user when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.



Course : UECS2344
Info : Quiz
Date : 23/04/2024
Requirement: Projector : ☐ Whiteboard : ☐ Computer : ☐
Sound System: ☐ Video Conferencing: ☐
Start Time : 04:00 PM
End Time : 05:00 PM
Room : KB501

Figure 6.85: Display Loading Indicator when Creating the Assessment



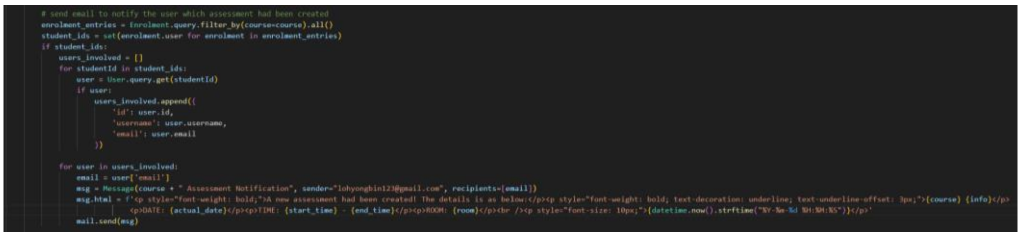
Assessment added!

Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User
3	2024-04-23	UECS2344	Quiz	16:00:00	17:00:00	KB501	Update Delete	View User Notify User

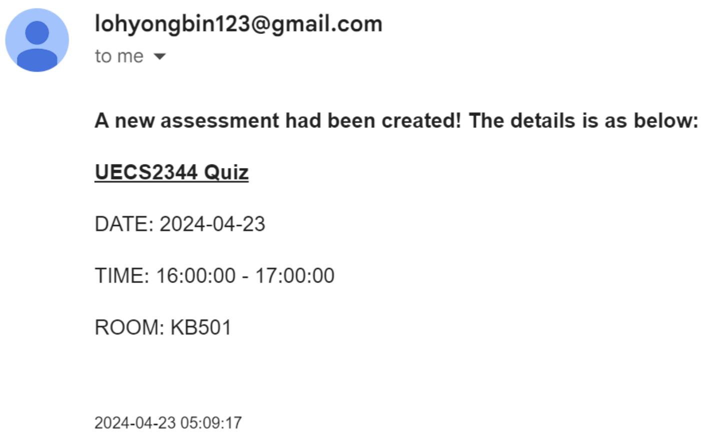
Figure 6.86: Creation Successfully with Success Messages

Upon successful submission, the new assessment entry is persisted to the database and administrators are promptly notified of the successful addition. While the creation process of the assessment takes a few seconds, the loading indicator is shown to indicate that it is creating the assessment at that time.



```
# send email to notify the user which assessment had been created
enrolment_entries = Enrolment.query.filter_by(course=course).all()
student_ids = set(enrolment.user for enrolment in enrolment_entries)
if student_ids:
    users_involved = []
    for studentId in student_ids:
        user = User.query.get(studentId)
        if user:
            users_involved.append({
                'id': user.id,
                'username': user.username,
                'email': user.email
            })

    for user in users_involved:
        email = user['email']
        msg = Message(course + " Assessment Notification", sender="lohyongbin123@gmail.com", recipients=[email])
        msg.html = f'<p style="font-weight: bold;">A new assessment had been created! The details is as below:</p><p style="font-weight: bold; text-decoration: underline; text-underline-offset: 3px;">{course} {info}</p>
                <p>DATE: {actual_date}</p><p>TIME: {start_time} - {end_time}</p><p>ROOM: {room}</p><br /><p style="font-size: 10px;">{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}</p>'
        mail.send(msg)
```

Figure 6.87: Implementation Code for Finding User Enrolled and Notify Them



lohyongbin123@gmail.com
to me

A new assessment had been created! The details is as below:

UECS2344 Quiz

DATE: 2024-04-23

TIME: 16:00:00 - 17:00:00

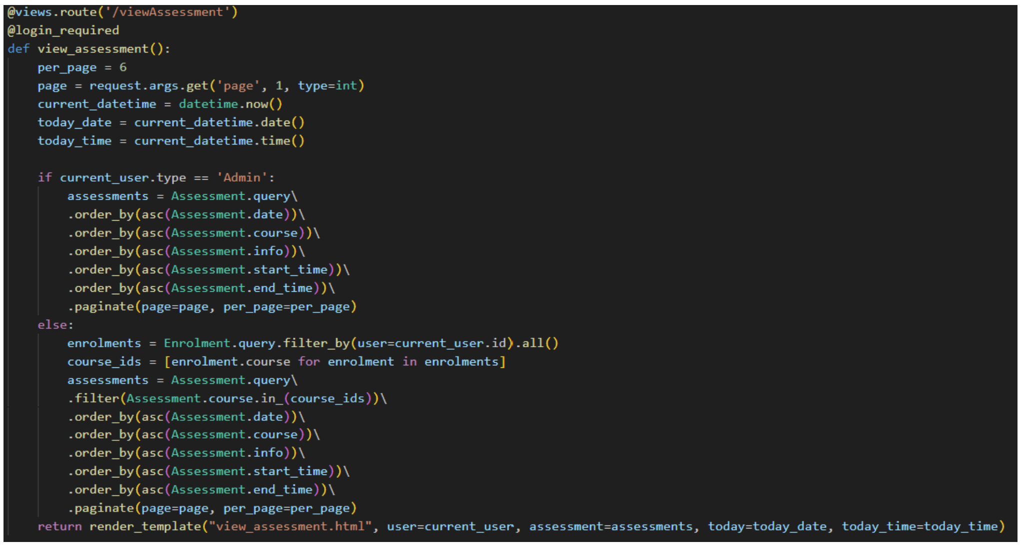
ROOM: KB501

2024-04-23 05:09:17

Figure 6.88: Real-Time Notification of the Assessment

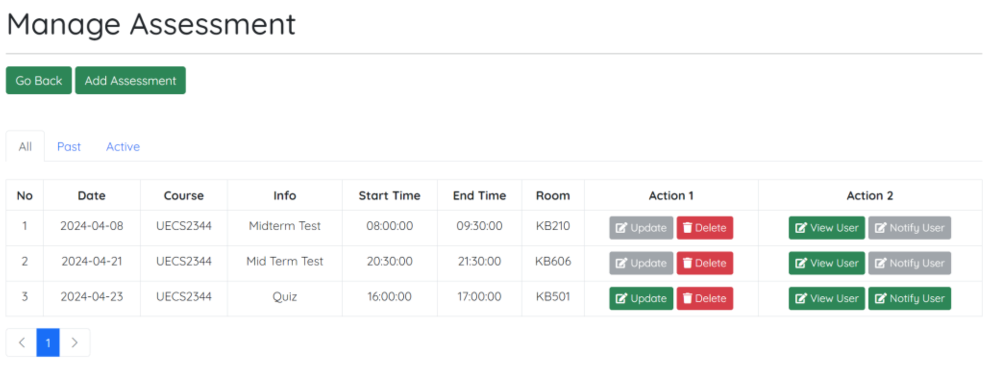
After the creation of assessment, the system will also automatically notify the users which enrolled in the assessment via email to ensure real-time notification, timely communication and foster a collaborative assessment environment.

6.9.2 View Assessment



```
@views.route('/viewAssessment')
@login_required
def view_assessment():
    per_page = 6
    page = request.args.get('page', 1, type=int)
    current_datetime = datetime.now()
    today_date = current_datetime.date()
    today_time = current_datetime.time()

    if current_user.type == 'Admin':
        assessments = Assessment.query\
        .order_by(asc(Assessment.date))\
        .order_by(asc(Assessment.course))\
        .order_by(asc(Assessment.info))\
        .order_by(asc(Assessment.start_time))\
        .order_by(asc(Assessment.end_time))\
        .paginate(page=page, per_page=per_page)
    else:
        enrolments = Enrolment.query.filter_by(user=current_user.id).all()
        course_ids = [enrolment.course for enrolment in enrolments]
        assessments = Assessment.query\
        .filter(Assessment.course.in_(course_ids))\
        .order_by(asc(Assessment.date))\
        .order_by(asc(Assessment.course))\
        .order_by(asc(Assessment.info))\
        .order_by(asc(Assessment.start_time))\
        .order_by(asc(Assessment.end_time))\
        .paginate(page=page, per_page=per_page)
    return render_template("view_assessment.html", user=current_user, assessment=assessments, today=today_date, today_time=today_time)
```

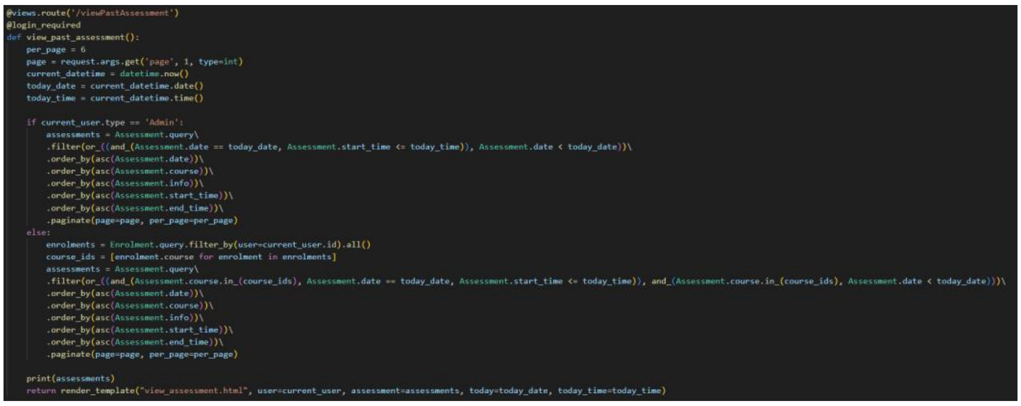
Figure 6.89: Implementation Code for Viewing All Assessment



No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User
3	2024-04-23	UECS2344	Quiz	16:00:00	17:00:00	KB501	Update Delete	View User Notify User

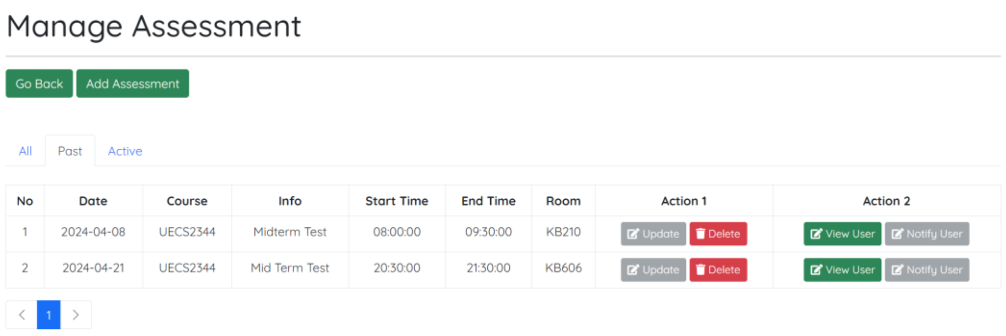
Figure 6.90: View All Assessment

The 'view_assessment' function serves as a central hub for accessing all assessment details which present a paginated view of assessments sorted by date, course and other pertinent attributes. By leveraging dynamic filtering based on user roles, this function offers administrators comprehensive insights into all assessments which foster proactive planning and resource allocation while lecturers and student can only view the assessments that they enrolled.



```python
@views.route('/viewPastAssessment')
@login_required
def view_past_assessment():
    per_page = 6
    page = request.args.get('page', 1, type=int)
    current_datetime = datetime.now()
    today_date = current_datetime.date()
    today_time = current_datetime.time()

    if current_user.type == 'Admin':
        assessments = Assessment.query\
        .filter(or_((and_(Assessment.date == today_date, Assessment.start_time <= today_time)), Assessment.date < today_date))\
        .order_by(asc(Assessment.date))\
        .order_by(asc(Assessment.course))\
        .order_by(asc(Assessment.info))\
        .order_by(asc(Assessment.start_time))\
        .order_by(asc(Assessment.end_time))\
        .paginate(page=page, per_page=per_page)
    else:
        enrolments = Enrolment.query.filter_by(user=current_user.id).all()
        course_ids = [enrolment.course for enrolment in enrolments]
        assessments = Assessment.query\
        .filter(or_((and_(Assessment.course.in_(course_ids), Assessment.date == today_date, Assessment.start_time <= today_time)), and_(Assessment.course.in_(course_ids), Assessment.date < today_date)))\
        .order_by(asc(Assessment.date))\
        .order_by(asc(Assessment.course))\
        .order_by(asc(Assessment.info))\
        .order_by(asc(Assessment.start_time))\
        .order_by(asc(Assessment.end_time))\
        .paginate(page=page, per_page=per_page)

    print(assessments)
    return render_template("view_assessment.html", user=current_user, assessment=assessments, today=today_date, today_time=today_time)
```

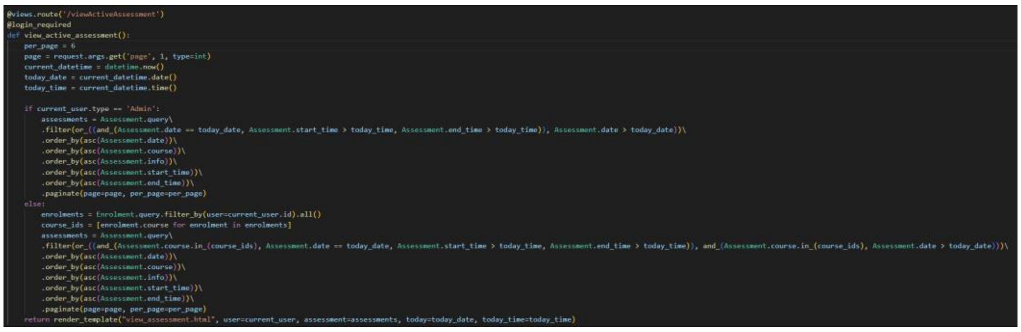
Figure 6.91: Implementation Code for Viewing Past Assessment



No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User

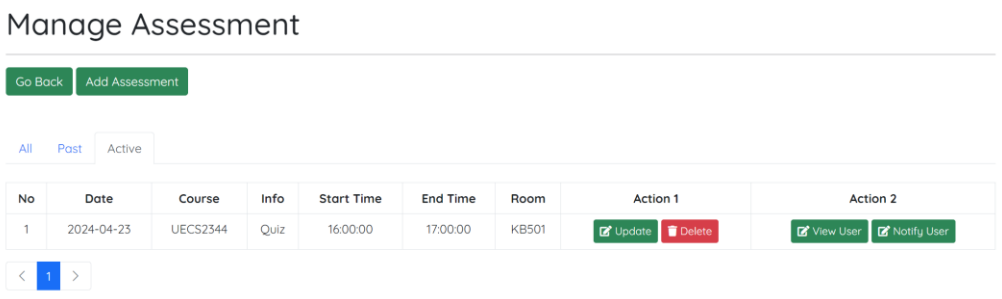
Figure 6.92: View Past Assessment

The 'view_past_assessment' function provides a streamlined interface for accessing historical assessment data. By leveraging date-based and user role filtering, this function enables particular users to review specific past assessments which administrator will be able to view all the past assessments while lecturers and students will only be able to view their own assessments.



```
@views.route('/viewActiveAssessment')
@login_required
def view_active_assessment():
    per_page = 6
    page = request.args.get('page', 1, type=int)
    current_datetime = datetime.now()
    today_date = current_datetime.date()
    today_time = current_datetime.time()

    if current_user.type == 'Admin':
        assessments = Assessment.query\
        .filter(or_((and_(Assessment.date == today_date, Assessment.start_time > today_time, Assessment.end_time > today_time)), Assessment.date > today_date))\
        .order_by(asc(Assessment.date))\
        .order_by(asc(Assessment.course))\
        .order_by(asc(Assessment.info))\
        .order_by(asc(Assessment.start_time))\
        .order_by(asc(Assessment.end_time))\
        .paginate(page=page, per_page=per_page)
    else:
        enrolments = Enrolment.query.filter_by(user=current_user.id).all()
        course_ids = [enrolment.course for enrolment in enrolments]
        assessments = Assessment.query\
        .filter(or_((and_(Assessment.course.in_(course_ids), Assessment.date == today_date, Assessment.start_time > today_time, Assessment.end_time > today_time)), and_(Assessment.course.in_(course_ids), Assessment.date > today_date)))\
        .order_by(asc(Assessment.date))\
        .order_by(asc(Assessment.course))\
        .order_by(asc(Assessment.info))\
        .order_by(asc(Assessment.start_time))\
        .order_by(asc(Assessment.end_time))\
        .paginate(page=page, per_page=per_page)
    return render_template("view_assessment.html", user=current_user, assessment=assessments, today=today_date, today_time=today_time)
```

Figure 6.93: Implementation Code for Viewing Active Assessment



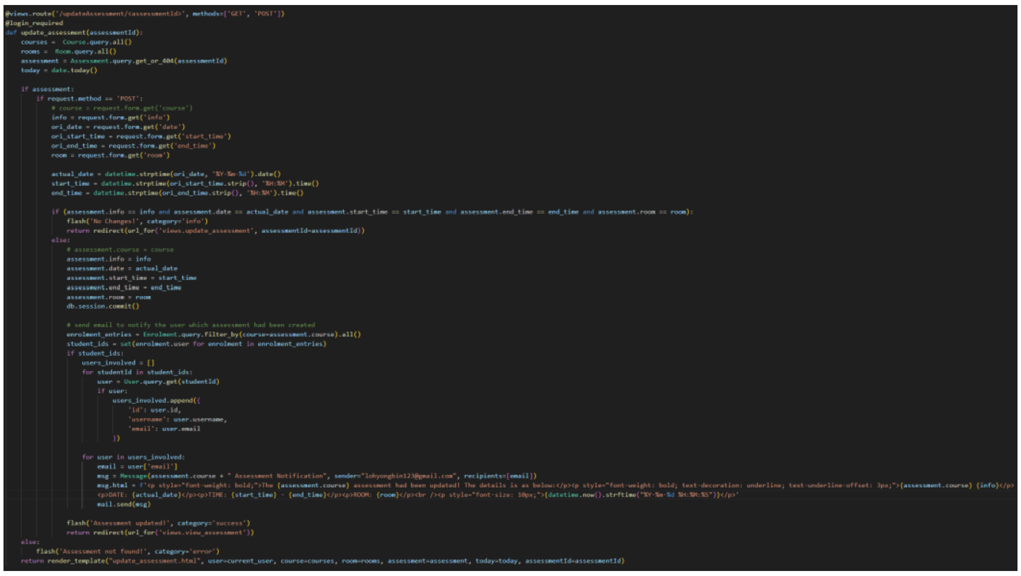
Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-23	UECS2344	Quiz	16:00:00	17:00:00	KB501	Update Delete	View User Notify User

Figure 6.94: View Active Assessments

The 'view_active_assessment' function offers real-time visibility into active assessments which allow users to monitor ongoing assessment activities and respond promptly to any emerging issues. By leveraging date and time-based filtering, this function ensures that users can focus on assessments occurring within the current timeframe to enhance operational agility and responsiveness.

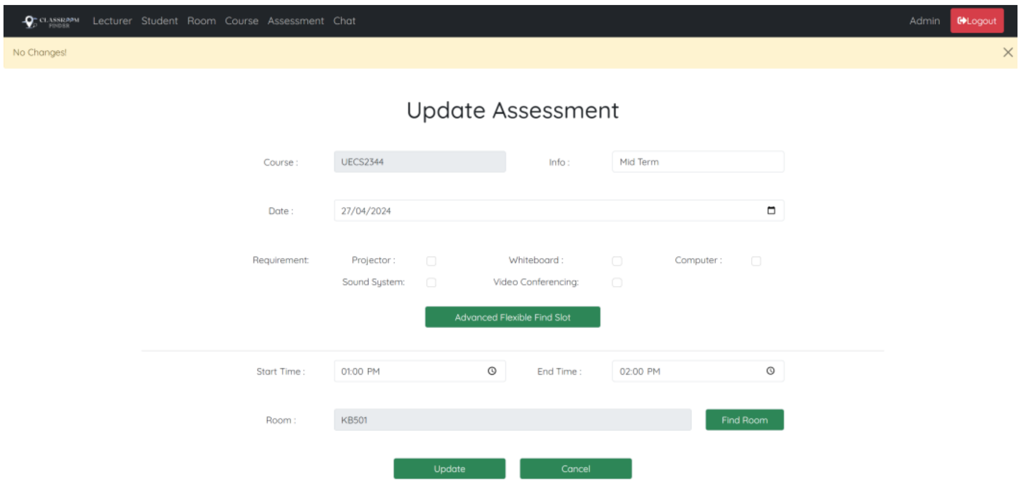
6.9.3 Update Assessment



```python
@views.route('/updateAssessment/<assessmentId>', methods=['GET', 'POST'])
@login_required
def update_assessment(assessmentId):
    courses =  Course.query.all()
    rooms =  Room.query.all()
    assessment = Assessment.query.get_or_404(assessmentId)
    today = date.today()

    if assessment:
        if request.method == 'POST':
            # course = request.form.get('course')
            info = request.form.get('info')
            ori_date = request.form.get('date')
            ori_start_time = request.form.get('start_time')
            ori_end_time = request.form.get('end_time')
            room = request.form.get('room')

            actual_date = datetime.strptime(ori_date, '%Y-%m-%d').date()
            start_time = datetime.strptime(ori_start_time.strip(), '%H:%M').time()
            end_time = datetime.strptime(ori_end_time.strip(), '%H:%M').time()

            if (assessment.info == info and assessment.date == actual_date and assessment.start_time == start_time and assessment.end_time == end_time and assessment.room == room):
                flash('No Changes!', category='info')
                return redirect(url_for('views.update_assessment', assessmentId=assessmentId))
            else:
                # assessment.course = course
                assessment.info = info
                assessment.date = actual_date
                assessment.start_time = start_time
                assessment.end_time = end_time
                assessment.room = room
                db.session.commit()

                # send email to notify the user which assessment had been created
                enrolment_entries = Enrolment.query.filter_by(course=assessment.course).all()
                student_ids = set(enrolment.user for enrolment in enrolment_entries)
                if student_ids:
                    users_involved = []
                    for studentId in student_ids:
                        user = User.query.get(studentId)
                        if user:
                            users_involved.append({
                                'id': user.id,
                                'username': user.username,
                                'email': user.email
                            })

                    for user in users_involved:
                        email = user['email']
                        msg = Message(assessment.course + " Assessment Notification", sender="lohyongbin123@gmail.com", recipients=[email])
                        msg.html = f'<p style="font-weight: bold;">The {assessment.course} assessment had been updated! The details is as below:</p><p style="font-weight: bold; text-decoration: underline; text-underline-offset: 3px;">{assessment.course} {info}</p>
                        <p>DATE: {actual_date}</p><p>TIME: {start_time} - {end_time}</p><p>ROOM: {room}</p><br /><p style="font-size: 10px;">{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}</p>'
                        mail.send(msg)

                flash('Assessment updated!', category='success')
                return redirect(url_for('views.view_assessment'))
    else:
        flash('Assessment not found!', category='error')
    return render_template("update_assessment.html", user=current_user, course=courses, room=rooms, assessment=assessment, today=today, assessmentId=assessmentId)
```

Figure 6.95: Implementation Code for Updating Assessment

The 'update_assessment' function which accessible via the '/updateAssessment/<assessmentId>' route, allows administrator to update existing assessment details. The administrator can access the update assessment form by providing the assessmentId in the route. The administrator can update the assessment by submitting the assessment details via the HTML form. In the HTML form, the find_rooms and find_rooms_and_slots functions also enhance the room allocation process by identifying available rooms and time slots that meet various constraints such as user availability, existing assessments, space, and equipment requirements. Before submitting the assessment details, all the input values are required to be filled in to perform the assessment updating process. If there is any input field which has empty value, the system will display the error message to notify the administrator when submitting the form. Once it passed the form validation, the form will send the POST request to the backend to process the data.


Lecturer Student Room Course Assessment Chat
Admin
Logout
Advanced Flexible Find Slot
Find Room
Update
Cancel

No Changes!

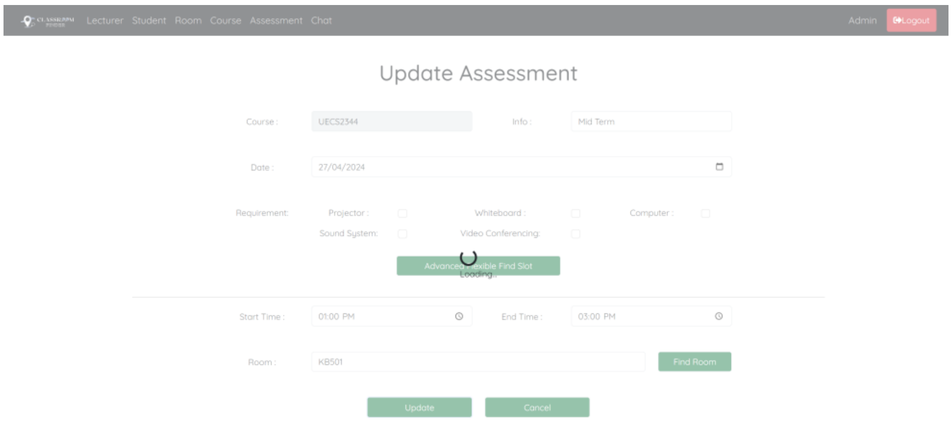
Update Assessment

Course :	UECS2344	Info :	Mid Term
Date :	27/04/2024		
Start Time :	01:00 PM	End Time :	02:00 PM
Room :	KB501		

Requirement: Projector: Whiteboard: Computer: Sound System: Video Conferencing:

Figure 6.96: Update Failed with Error Messages

Upon receiving a POST request containing the necessary information which are assessment details from the update assessment form, the function retrieves the assessment record associated with the provided Id and validates the submitted changes.

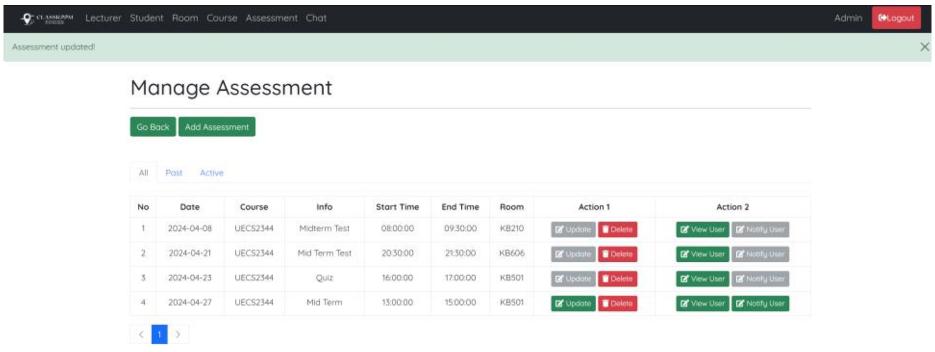

Lecturer Student Room Course Assessment Chat
Admin
Logout
Advanced Flexible Find Slot
Loading...
Find Room
Update
Cancel

Update Assessment

Course :	UECS2344	Info :	Mid Term
Date :	27/04/2024		
Start Time :	01:00 PM	End Time :	03:00 PM
Room :	KB501		

Requirement: Projector: Whiteboard: Computer: Sound System: Video Conferencing:

Figure 6.97: Display Loading Indicator when Updating the Assessment



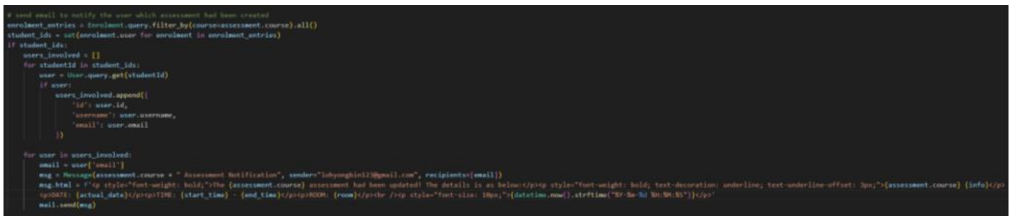
Assessment updated!

Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User
3	2024-04-23	UECS2344	Quiz	16:00:00	17:00:00	KB501	Update Delete	View User Notify User
4	2024-04-27	UECS2344	Mid Term	13:00:00	15:00:00	KB501	Update Delete	View User Notify User

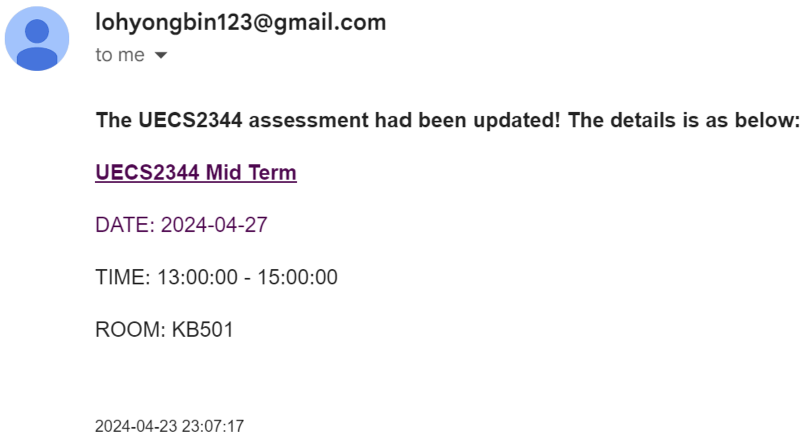
Figure 6.98: Update Successfully with Success Messages

If the submitted data is valid and differs from the existing record, the assessment details are updated in the database and the administrator is redirected to the view assessment page with a success message. While the updating process of the assessment takes a few seconds, the loading indicator is shown to indicate that it is updating the assessment at that time.



```
# send email to notify the user which assessment had been created
enrolment_entries = Enrolment.query.filter_by(course=assessment.course).all()
student_ids = set(enrolment.user for enrolment in enrolment_entries)
if student_ids:
    users_involved = []
    for studentId in student_ids:
        user = User.query.get(studentId)
        if user:
            users_involved.append({
                'id': user.id,
                'username': user.username,
                'email': user.email
            })

    for user in users_involved:
        email = user['email']
        msg = Message(assessment.course + " Assessment Notification", sender="lohyongbin123@gmail.com", recipients=[email])
        msg.html = f'<p style="font-weight: bold;">The {assessment.course} assessment had been updated! The details is as below:</p><p style="font-weight: bold; text-decoration: underline; text-underline-offset: 3px;">{assessment.course} {info}</p>
        <p>DATE: {actual_date}</p><p>TIME: {start_time} - {end_time}</p><p>ROOM: {room}</p><br /><p style="font-size: 10px;">{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}</p>'
        mail.send(msg)
```

Figure 6.99: Implementation Code for Finding User Enrolled and Notify Them



lohyongbin123@gmail.com
to me

The UECS2344 assessment had been updated! The details is as below:

UECS2344 Mid Term

DATE: 2024-04-27

TIME: 13:00:00 - 15:00:00

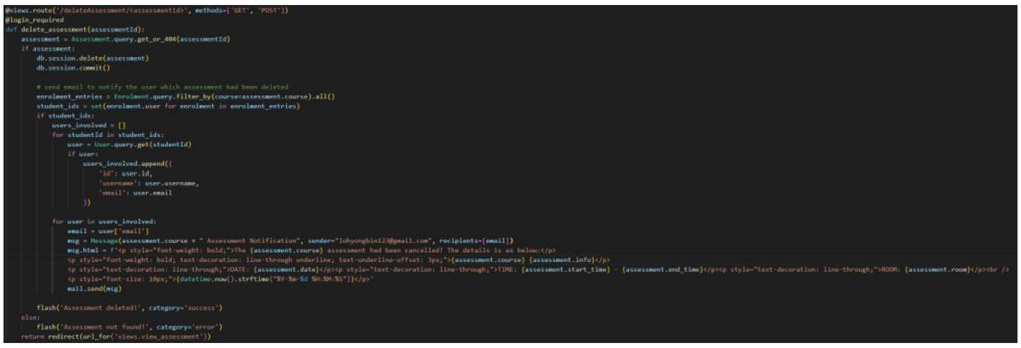
ROOM: KB501

2024-04-23 23:07:17

Figure 6.100: Real-Time Notification of the Assessment

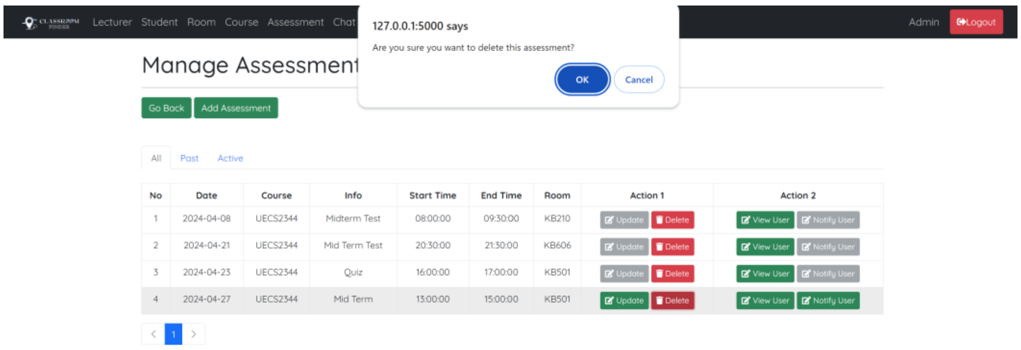
After the updates of assessment, the system will also automatically notify the users which enrolled in the assessment via email to ensure real-time notification of the assessment and the room details.

6.9.4 Delete Assessment



```
@views.route('/deleteAssessment/<assessmentId>', methods=['GET', 'POST'])
@login_required
def delete_assessment(assessmentId):
    assessment = Assessment.query.get_or_404(assessmentId)
    if assessment:
        db.session.delete(assessment)
        db.session.commit()

        # send email to notify the user which assessment had been deleted
        enrolment_entries = Enrolment.query.filter_by(course=assessment.course).all()
        student_ids = set(enrolment.user for enrolment in enrolment_entries)
        if student_ids:
            users_involved = []
            for studentId in student_ids:
                user = User.query.get(studentId)
                if user:
                    users_involved.append({
                        'id': user.id,
                        'username': user.username,
                        'email': user.email
                    })

            for user in users_involved:
                email = user['email']
                msg = Message(assessment.course + " Assessment Notification", sender="lohyongbin123@gmail.com", recipients=[email])
                msg.html = f'<p style="font-weight: bold;">The {assessment.course} assessment had been cancelled! The details is as below:</p>
                <p style="font-weight: bold; text-decoration: line-through underline; text-underline-offset: 3px;">{assessment.course} {assessment.info}</p>
                <p style="text-decoration: line-through;">DATE: {assessment.date}</p><p style="text-decoration: line-through;">TIME: {assessment.start_time} - {assessment.end_time}</p><p style="text-decoration: line-through;">ROOM: {assessment.room}</p><br />
                <p style="font-size: 10px;">{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}</p>'
                mail.send(msg)

        flash('Assessment deleted!', category='success')
    else:
        flash('Assessment not found!', category='error')
    return redirect(url_for('views.view_assessment'))
```

Figure 6.101: Implementation Code for Deleting Assessment



127.0.0.1:5000 says

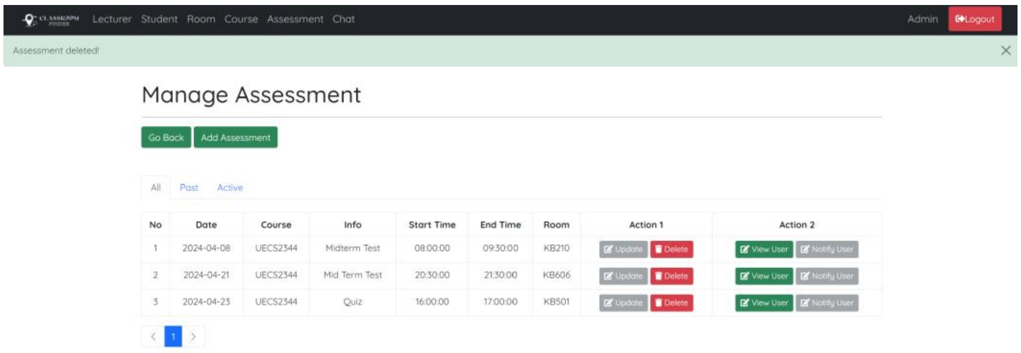
Are you sure you want to delete this assessment?

Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User
3	2024-04-23	UECS2344	Quiz	16:00:00	17:00:00	KB501	Update Delete	View User Notify User
4	2024-04-27	UECS2344	Mid Term	13:00:00	15:00:00	KB501	Update Delete	View User Notify User

Figure 6.102: Confirmation Prompt before Deleting Assessment

In scenarios where assessments need to be removed or cancelled, the 'delete_assessment' function provides administrators with a straightforward mechanism for deleting the assessment records while having confirmation before the deletion.

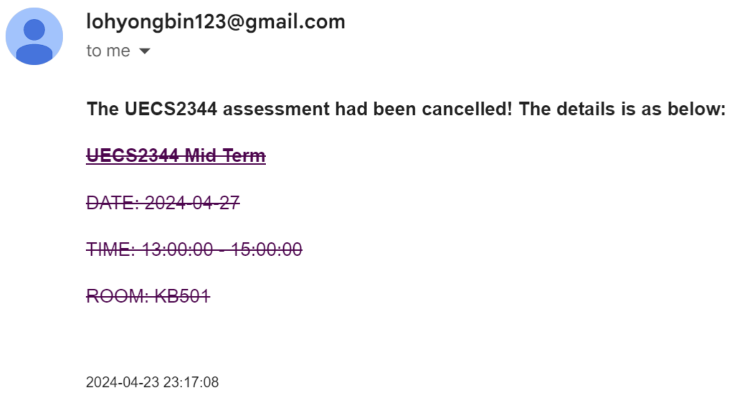


Assessment deleted!

Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User
3	2024-04-23	UECS2344	Quiz	16:00:00	17:00:00	KB501	Update Delete	View User Notify User

Figure 6.103: Delete Successfully with Success Messages


lohyongbin123@gmail.com
to me

The UECS2344 assessment had been cancelled! The details is as below:

~~**UECS2344 Mid Term**~~

~~DATE: 2024-04-27~~

~~TIME: 13:00:00 - 15:00:00~~

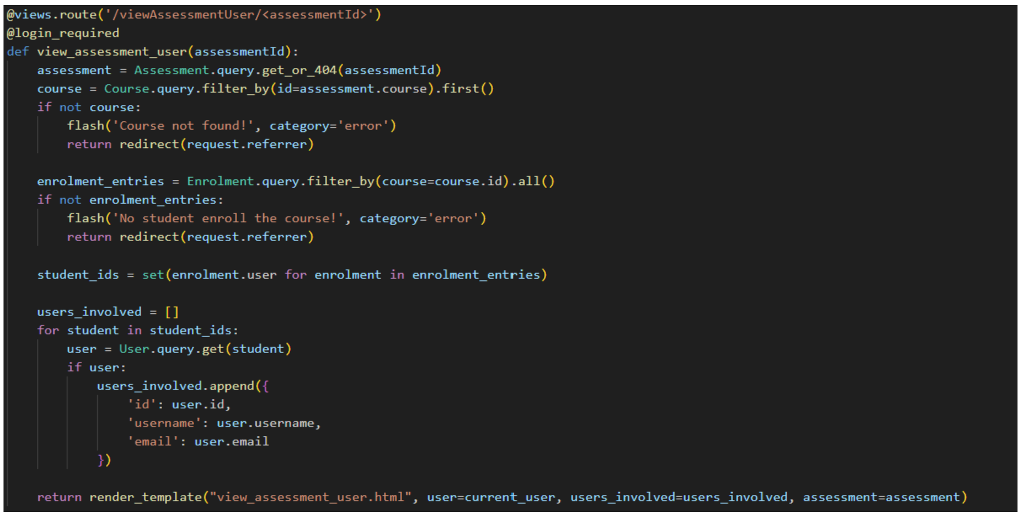
~~ROOM: KB501~~

2024-04-23 23:17:08

Figure 6.104: Real-Time Notification of the Assessment after Deletion

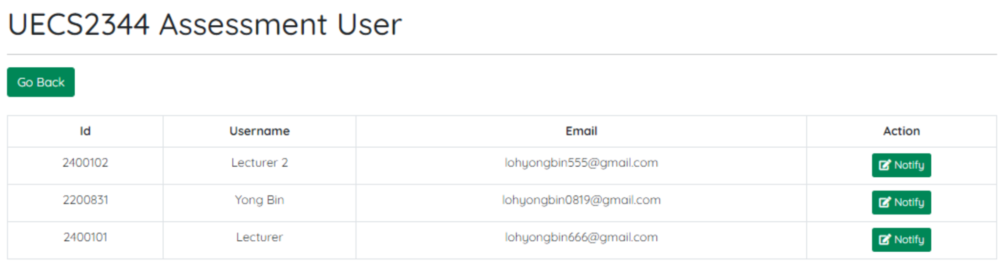
Upon deletion, the system triggers email notifications to affected users. This can help to ensure that the transparency and minimizing disruption to instructional activities.

6.9.5 View Assessment User



```
@views.route('/viewAssessmentUser/<assessmentId>')
@login_required
def view_assessment_user(assessmentId):
    assessment = Assessment.query.get_or_404(assessmentId)
    course = Course.query.filter_by(id=assessment.course).first()
    if not course:
        flash('Course not found!', category='error')
        return redirect(request.referrer)

    enrolment_entries = Enrolment.query.filter_by(course=course.id).all()
    if not enrolment_entries:
        flash('No student enroll the course!', category='error')
        return redirect(request.referrer)

    student_ids = set(enrolment.user for enrolment in enrolment_entries)

    users_involved = []
    for student in student_ids:
        user = User.query.get(student)
        if user:
            users_involved.append({
                'id': user.id,
                'username': user.username,
                'email': user.email
            })

    return render_template("view_assessment_user.html", user=current_user, users_involved=users_involved, assessment=assessment)
```

Figure 6.105: Implementation Code for Viewing Users Who Enrol in the Assessment



UECS2344 Assessment User

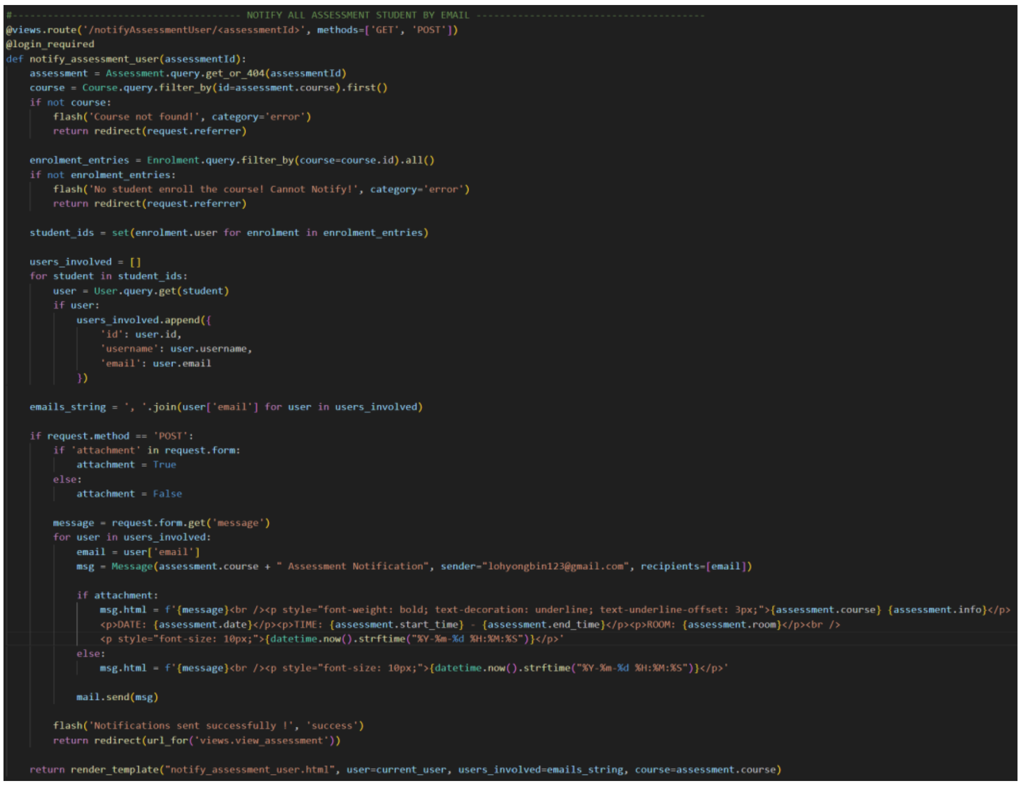
Go Back

Id	Username	Email	Action
2400102	Lecturer 2	lohyongbin555@gmail.com	Notify
2200831	Yong Bin	lohyongbin0819@gmail.com	Notify
2400101	Lecturer	lohyongbin666@gmail.com	Notify

Figure 6.106: View Users Who Enrol in the particular Assessment

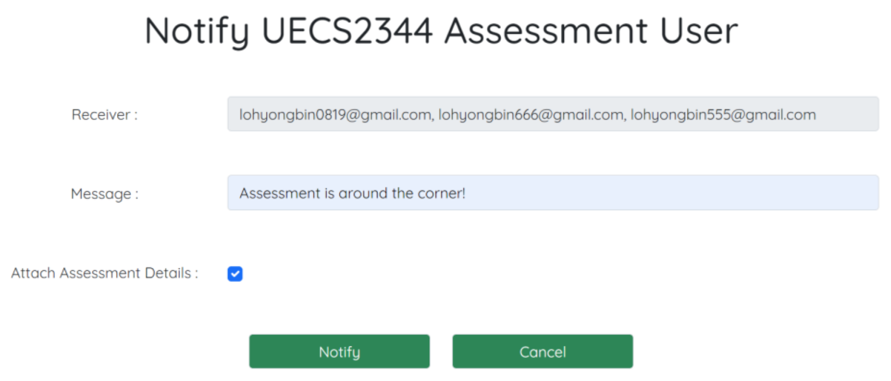
The view_assessment_user function plays a crucial role in our application by allowing authenticated users to review student enrolment details for a specific assessment. This Flask route fetches the assessment's information from the database based on the provided assessment ID which is to ensure the robust error handling in case of non-existent assessments. It then retrieves associated course details and corresponding enrolment entries to present a concise list of unique students involved in the assessment. By fetching user details for each student ID and rendering them in a user-friendly template, the function provides valuable insights into student participation which enhance transparency and facilitating informed decision-making within our application's ecosystem.

6.9.6 Notify Assessment User



```python
#---------------------------------------- NOTIFY ALL ASSESSMENT STUDENT BY EMAIL ----------------------------------------
@views.route('/notifyAssessmentUser/<assessmentId>', methods=['GET', 'POST'])
@login_required
def notify_assessment_user(assessmentId):
    assessment = Assessment.query.get_or_404(assessmentId)
    course = Course.query.filter_by(id=assessment.course).first()
    if not course:
        flash('Course not found!', category='error')
        return redirect(request.referrer)

    enrolment_entries = Enrolment.query.filter_by(course=course.id).all()
    if not enrolment_entries:
        flash('No student enroll the course! Cannot Notify!', category='error')
        return redirect(request.referrer)

    student_ids = set(enrolment.user for enrolment in enrolment_entries)

    users_involved = []
    for student in student_ids:
        user = User.query.get(student)
        if user:
            users_involved.append({
                'id': user.id,
                'username': user.username,
                'email': user.email
            })

    emails_string = ', '.join(user['email'] for user in users_involved)

    if request.method == 'POST':
        if 'attachment' in request.form:
            attachment = True
        else:
            attachment = False

        message = request.form.get('message')
        for user in users_involved:
            email = user['email']
            msg = Message(assessment.course + " Assessment Notification", sender="lohyongbin123@gmail.com", recipients=[email])

            if attachment:
                msg.html = f'{message}<br /><p style="font-weight: bold; text-decoration: underline; text-underline-offset: 3px;">{assessment.course} {assessment.info}</p>
                <p>DATE: {assessment.date}</p><p>TIME: {assessment.start_time} - {assessment.end_time}</p><p>ROOM: {assessment.room}</p><br />
                <p style="font-size: 10px;">{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}</p>'
            else:
                msg.html = f'{message}<br /><p style="font-size: 10px;">{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}</p>'

            mail.send(msg)

        flash('Notifications sent successfully !', 'success')
        return redirect(url_for('views.view_assessment'))

    return render_template("notify_assessment_user.html", user=current_user, users_involved=emails_string, course=assessment.course)
```

Figure 6.107: Implementation Code for Notifying Users Who Enrol in the Assessment



Notify UECS2344 Assessment User

Receiver : lohyongbin0819@gmail.com, lohyongbin666@gmail.com, lohyongbin555@gmail.com

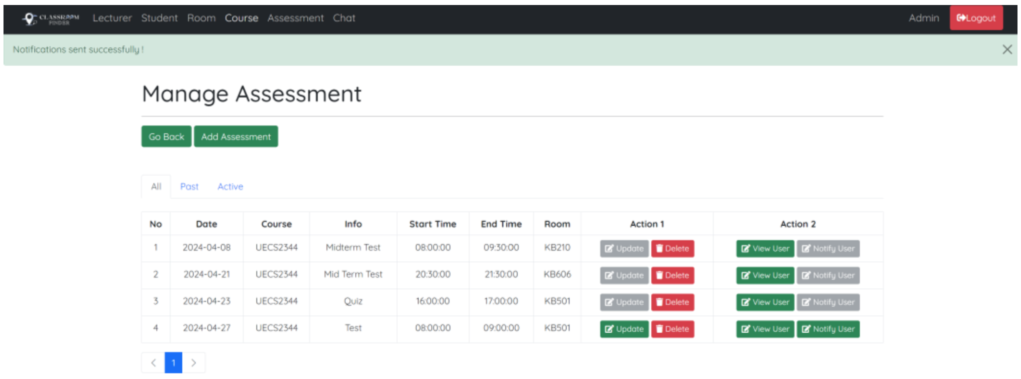
Message : Assessment is around the corner!

Attach Assessment Details : ✓

Figure 6.108: Notify Users Who Enrol in the particular Assessment

The notify_assessment_user function is a Flask route designed to notify all users enrolled in a specific assessment via email. It begins by retrieving the assessment details and associated course information from the database. If the course or enrolment entries are not found, appropriate error messages are flashed to the user interface. Then, it collects the email addresses of all enrolled students and prepares the email message.

The route supports both GET and POST methods. In the POST method, it handles the form submission which is including the message content and an optional attachment. If the attachment is required, the message will be including the assessment details as the attachment in the email message.

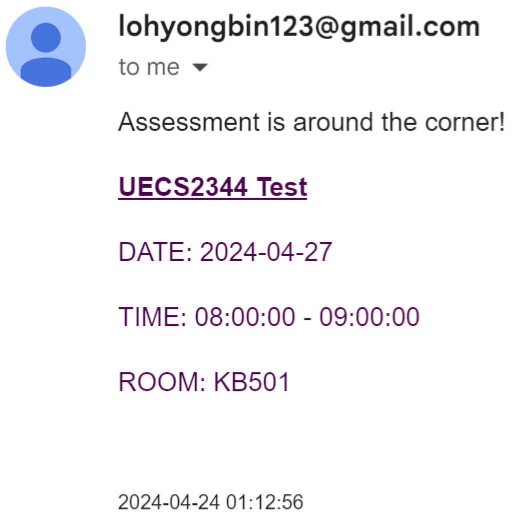


Notifications sent successfully !

Manage Assessment

No	Date	Course	Info	Start Time	End Time	Room	Action 1	Action 2
1	2024-04-08	UECS2344	Midterm Test	08:00:00	09:30:00	KB210	Update Delete	View User Notify User
2	2024-04-21	UECS2344	Mid Term Test	20:30:00	21:30:00	KB606	Update Delete	View User Notify User
3	2024-04-23	UECS2344	Quiz	16:00:00	17:00:00	KB501	Update Delete	View User Notify User
4	2024-04-27	UECS2344	Test	08:00:00	09:00:00	KB501	Update Delete	View User Notify User

Figure 6.109: Notify Successfully with Success Messages



lohyongbin123@gmail.com

to me

Assessment is around the corner!

UECS2344 Test

DATE: 2024-04-27

TIME: 08:00:00 - 09:00:00

ROOM: KB501

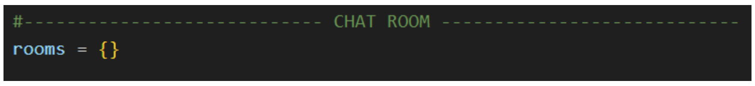
2024-04-24 01:12:56

Figure 6.110: Notification of the Assessment

For each enrolled student, it constructs an email message with the provided content and sends it using Flask-Mail. Success messages are flashed upon completion of notification.

6.10 Communication Module

The communication module in this project serves to enable real-time chat functionality among users through Flask-SocketIO which to ensure seamless interaction within the application. There are two types of users involved in this module which are administrator and lecturers. For the administrator, the administrator can view all the chatroom which the lecturers have created, enter the chatroom and chat with the lecturers. For the lecturers, the lecturers are directly viewing their own chatroom by creating the room once the lecturers view the chat.

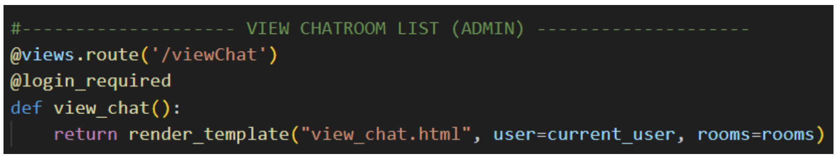


```
#--------------------------- CHAT ROOM ---------------------------
rooms = {}
```

Figure 6.111: Implementation Code to Manage Room

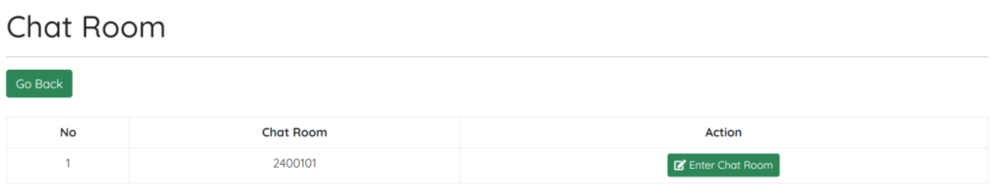
It begins with the management of chat rooms, where a dictionary called rooms keeps track of active rooms, each uniquely identified by a roomId. This setup allows for efficient organization and access to ongoing conversations.

6.10.1 View Chatroom (Administrator)



```
#------------------- VIEW CHATROOM LIST (ADMIN) -------------------
@views.route('/viewChat')
@login_required
def view_chat():
    return render_template("view_chat.html", user=current_user, rooms=rooms)
```

Figure 6.112: Implementation Code to View Chat Room (Administrator)



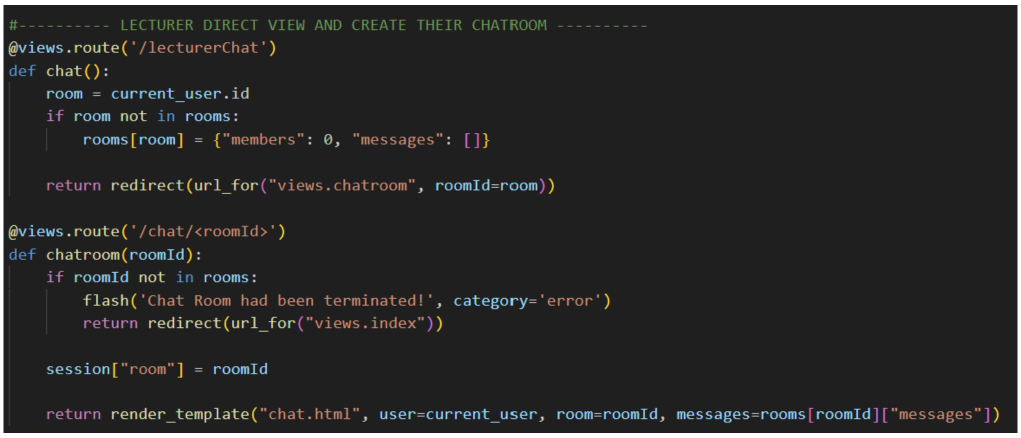
Chat Room

No	Chat Room	Action
1	2400101	Enter Chat Room

Figure 6.113: View Chatroom List

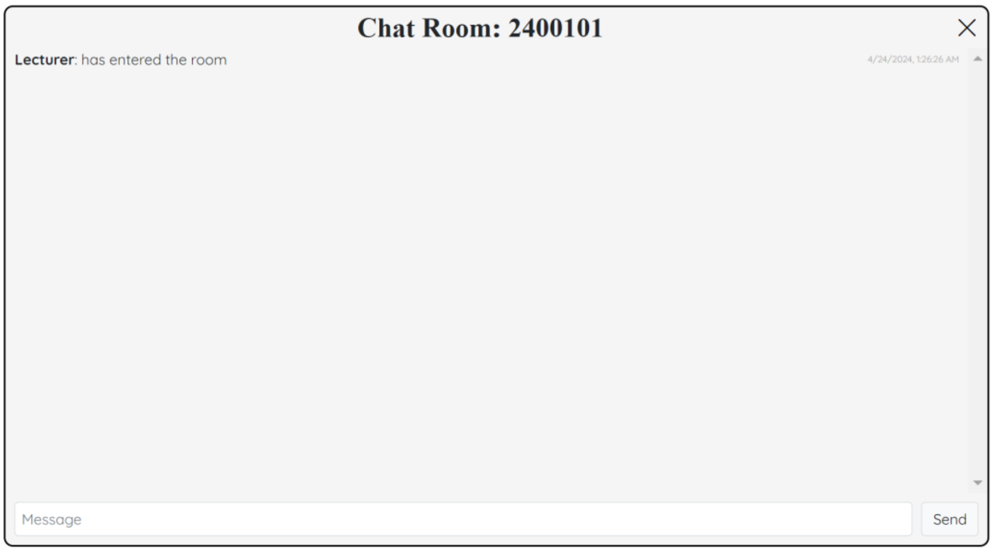
The view_chat route presents an interface for users to view available chat rooms which aims to enhance the user engagement and collaboration.

6.10.2 View Chatroom (Lecturer)



```python
#---------- LECTURER DIRECT VIEW AND CREATE THEIR CHATROOM ----------
@views.route('/lecturerChat')
def chat():
    room = current_user.id
    if room not in rooms:
        rooms[room] = {"members": 0, "messages": []}

    return redirect(url_for("views.chatroom", roomId=room))

@views.route('/chat/<roomId>')
def chatroom(roomId):
    if roomId not in rooms:
        flash('Chat Room had been terminated!', category='error')
        return redirect(url_for("views.index"))

    session["room"] = roomId

    return render_template("chat.html", user=current_user, room=roomId, messages=rooms[roomId]["messages"])
```

Figure 6.114: Implementation Code to View Chat Room (Lecturer)



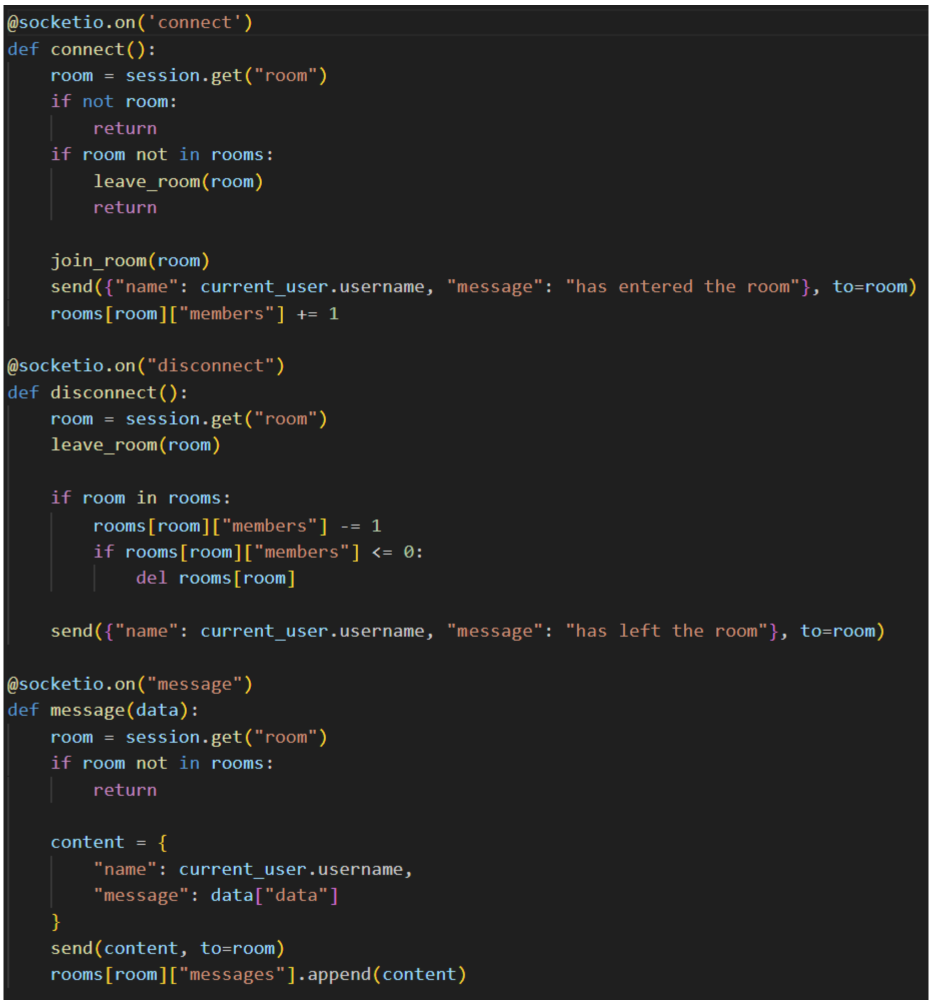
Chat Room: 2400101

Lecturer: has entered the room 4/24/2024, 1:26:26 AM

Figure 6.115: View the lecturer own Chat Room

For lecturers, the chat route facilitates direct access or creation of chat rooms which help to ensure a seamless experience tailored to their needs. Upon entering a chat room (chatroom route), users are provided with a dedicated space to view existing messages and send new ones which to promote the effective communication between administrator and lecturer.

6.10.3 Socket IO Event Handling



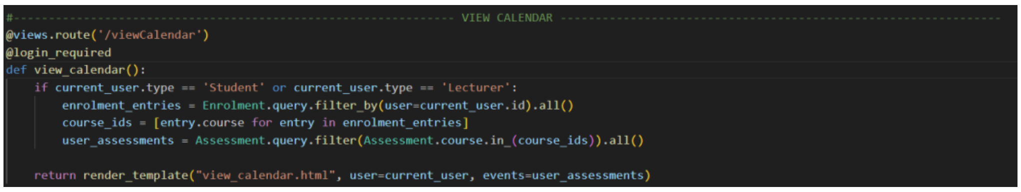
```python
@socketio.on('connect')
def connect():
    room = session.get("room")
    if not room:
        return
    if room not in rooms:
        leave_room(room)
        return

    join_room(room)
    send({"name": current_user.username, "message": "has entered the room"}, to=room)
    rooms[room]["members"] += 1

@socketio.on("disconnect")
def disconnect():
    room = session.get("room")
    leave_room(room)

    if room in rooms:
        rooms[room]["members"] -= 1
        if rooms[room]["members"] <= 0:
            del rooms[room]

    send({"name": current_user.username, "message": "has left the room"}, to=room)

@socketio.on("message")
def message(data):
    room = session.get("room")
    if room not in rooms:
        return

    content = {
        "name": current_user.username,
        "message": data["data"]
    }
    send(content, to=room)
    rooms[room]["messages"].append(content)
```

Figure 6.116: Implementation Code to Handle Socket IO Event

SocketIO events are utilized to handle connection and disconnection of users (connect and disconnect events) to ensure that the users are dynamically added or removed from rooms based on their online status. Additionally, the message event enables the real-time broadcasting of messages to all members within a chat room which can help to foster the instant communication and collaboration. Overall, this communication module enhances the user interaction within the application and facilitate the efficient communication and collaboration in real-time.

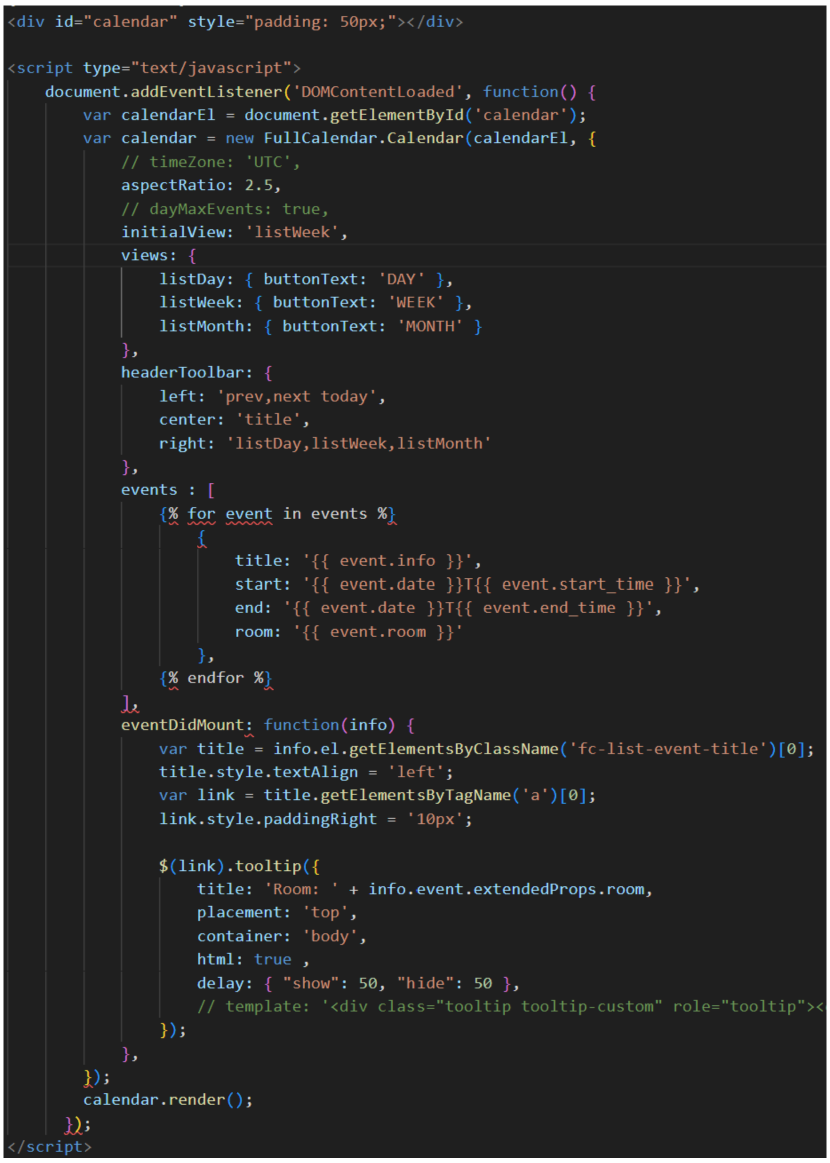
6.11 Calendar Module

The calendar module of the project facilitates users whether they are students or lecturers, to effectively manage their schedules through an intuitive and interactive calendar interface.



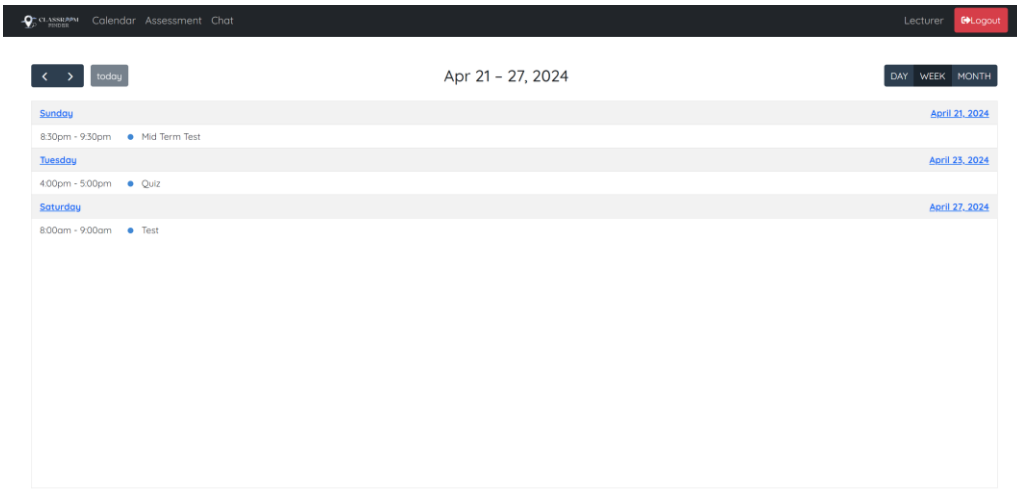
```
#---------------------------------------------------------------- VIEW CALENDAR ------------------------------------------------------------------
@views.route('/viewCalendar')
@login_required
def view_calendar():
    if current_user.type == 'Student' or current_user.type == 'Lecturer':
        enrolment_entries = Enrolment.query.filter_by(user=current_user.id).all()
        course_ids = [entry.course for entry in enrolment_entries]
        user_assessments = Assessment.query.filter(Assessment.course.in_(course_ids)).all()

    return render_template("view_calendar.html", user=current_user, events=user_assessments)
```

Figure 6.117: Implementation Code to View Calendar\



```
<div id="calendar" style="padding: 50px;"></div>

<script type="text/javascript">
    document.addEventListener('DOMContentLoaded', function() {
        var calendarEl = document.getElementById('calendar');
        var calendar = new FullCalendar.Calendar(calendarEl, {
            // timeZone: 'UTC',
            aspectRatio: 2.5,
            // dayMaxEvents: true,
            initialView: 'listWeek',
            views: {
                listDay: { buttonText: 'DAY' },
                listWeek: { buttonText: 'WEEK' },
                listMonth: { buttonText: 'MONTH' }
            },
            headerToolbar: {
                left: 'prev,next today',
                center: 'title',
                right: 'listDay,listWeek,listMonth'
            },
            events : [
                {% for event in events %}
                    {
                        title: '{{ event.info }}',
                        start: '{{ event.date }}T{{ event.start_time }}',
                        end: '{{ event.date }}T{{ event.end_time }}',
                        room: '{{ event.room }}'
                    },
                {% endfor %}
            ],
            eventDidMount: function(info) {
                var title = info.el.getElementsByClassName('fc-list-event-title')[0];
                title.style.textAlign = 'left';
                var link = title.getElementsByTagName('a')[0];
                link.style.paddingRight = '10px';

                $(link).tooltip({
                    title: 'Room: ' + info.event.extendedProps.room,
                    placement: 'top',
                    container: 'body',
                    html: true ,
                    delay: { "show": 50, "hide": 50 },
                    // template: '<div class="tooltip tooltip-custom" role="tooltip"><
                });
            },
        });
        calendar.render();
    });
</script>
```

Figure 6.118: HTML Code to Display Calendar

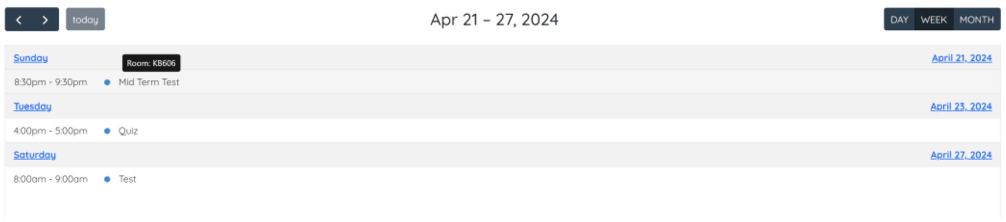


Apr 21 – 27, 2024

Sunday April 21, 2024
8:30pm - 9:30pm ● Mid Term Test
Tuesday April 23, 2024
4:00pm - 5:00pm ● Quiz
Saturday April 27, 2024
8:00am - 9:00am ● Test

Figure 6.119: View Calendar

Upon accessing the calendar page via the view_calendar route, the users are presented with a personalized view tailored to their role, displaying relevant events and assessments associated with their enrolled courses. Powered by the FullCalendar library, the calendar interface offers flexibility by allowing users to switch between different views such as day, week and month, thereby accommodating various time scales based on their preferences. The navigation controls embedded within the header toolbar enable seamless navigation between dates and facilitate changes in the view mode. The events retrieved from the database are dynamically populated in the calendar which provide the users with essential details such as event titles, start and end dates and room information.



Apr 21 – 27, 2024

Sunday Room: KB606 April 21, 2024
8:30pm - 9:30pm ● Mid Term Test
Tuesday April 23, 2024
4:00pm - 5:00pm ● Quiz
Saturday April 27, 2024
8:00am - 9:00am ● Test

Figure 6.120: Tooltip Feature

Moreover, each event title is equipped with a tooltip feature, offering additional context by displaying details like the room where the event takes place when hovered over. This comprehensive feature set empowers users to stay organized, informed and efficient in managing their schedules within the educational context.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

In the realm of software development, the process of ensuring the reliability, functionality and usability of a system or application is paramount to its success. System testing plays a crucial role in this process by rigorously evaluating the developed solution against predefined requirements and specifications. It serves as the final phase of the software development lifecycle where the focus shifts from individual components to the integrated system as a whole. This phase encompasses a series of tests designed to validate the system's behaviour, performance and compliance with user expectations.

The purpose of this chapter is to provide an in-depth exploration of system testing for the developed solution, encompassing evaluation, test planning and execution. With a comprehensive understanding of the system's functionalities and requirements, the testing process aims to identify defects, inconsistencies and areas for improvement, ultimately ensuring the delivery of a robust and reliable product. Therefore, in this chapter, it delves into the development of a detailed test plan which covers all aspects of the system's functionality and user interactions. Additionally, it explores the execution of various testing methodologies which includes end-user testing and supplementary testing types beyond specification testing to validate the system's performance under different scenarios. Through systematic testing and meticulous documentation of results, this chapter endeavours to demonstrate the thoroughness and effectiveness of the testing process in verifying the solution's quality and meeting stakeholder expectations.

7.2 Unit Testing

The unit testing is a fundamental practice in software development aimed at verifying the correctness and functionality of individual units or components of a software system. These units are typically the smallest testable parts of an application such as functions, methods or classes. The primary objectives of unit testing are to ensure that each unit performs as expected, isolating and testing it in isolation from the rest of the system. Unit testing is a crucial practice in software development that promotes code quality, early defect detection and maintainability which ultimately contributing to the overall reliability and success of the software product.

7.2.1 Unit Test Cases and Results

Table 7.1: Unit Test Case for Sign Up

Test Case ID	TC-001		**Module**		Sign Up	
Test Title	Sign Up					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	-					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Sign Up account with valid id, username, email and password	1. Enter id, username, email, password and confirm password 2. Click "Sign Up" button	Id: 2200831 Username: YB Email: lohyongbin0819@gmail.com Password: 123 Confirm Password: 123	Account is signed up successfully. The success message of "Account created!" is flashed.	User account should be created into the system.	Account is signed up successfully. The success message of "Account created!" is flashed.	Pass
Sign Up account with empty id, username, email and password	1. Enter id, username, email, password and confirm password	Id: null Username: null Email: null Password: null Confirm Password: null	The error message of "Please fill out this field" is shown below the	-	The error message of "Please fill out this field" is shown below the	Pass

	2. Click “Sign Up” button		input field respectively.		input field respectively.	
Sign Up account with valid id, username, password but wrong input format of email	1. Enter id, username, email, password and confirm password 2. Click “Sign Up” button	Id: 2200831 Username: YB Email: lohyongbin Password: 123 Confirm Password: 123	The error message of “Please include an ‘@’ in the email address. ‘lohyongbin’ is missing an ‘@’” is shown below the input field with wrong format respectively.	-	The error message of “Please include an ‘@’ in the email address. ‘lohyongbin’ is missing an ‘@’” is shown below the input field with wrong format respectively.	Pass
Sign Up account with valid id, username, email but invalid length of password which less than 3 characters	1. Enter id, username, email, password and confirm password 2. Click “Sign Up” button	Id: 2200831 Username: YB Email: lohyongbin0819@gmail.com Password: 12 Confirm Password: 12	The error message of “Password must be at least 3 characters.” is flashed.	-	The error message of “Password must be at least 3 characters.” is flashed.	Pass

Sign Up account with valid id, username, email but password not match with the confirm password	1. Enter id, username, email, password and confirm password 2. Click "Sign Up" button	Id: 2200831 Username: YB Email: lohyongbin0819@gmail.com Password: 123 Confirm Password: 1234	The error message of "Password don't match." is flashed.	-	The error message of "Password don't match." is flashed.	Pass
Sign Up account with valid id, username, email and password but id already existed	1. Enter id, username, email, password and confirm password 2. Click "Sign Up" button	Id: 2200831 Username: YB Email: lohyongbin0819@gmail.com Password: 123 Confirm Password: 123	The error message of "User already exists." is flashed.	-	The error message of "User already exists." is flashed.	Pass
Sign Up account with valid id, username, email and password but email already existed	1. Enter id, username, email, password and confirm password 2. Click "Sign Up" button	Id: 2200831 Username: YB Email: lohyongbin0819@gmail.com Password: 123 Confirm Password: 123	The error message of "Email already exists." is flashed.	-	The error message of "Email already exists." is flashed.	Pass

Table 7.2: Unit Test Case for Login

Test Case ID	TC-002		**Module**		Login and Logout	
Test Title	Login					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Users have a registered account					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Login account with valid registered email and password	1. Enter email and password 2. Click “Sign In” button	Email: lohyongbin0819@gmail.com Password: 123	Account is signed in successfully. The success message of “Logged in successfully!” is flashed.	The user should be redirected to Home Page based on their user roles.	Account is signed in successfully. The success message of “Logged in successfully!” is flashed.	Pass
Login account with empty email and password	1. Enter email and password 2. Click “Sign In” button	Email: null Password: null	The error message of “Please fill out this field” is shown below the input field respectively.	-	The error message of “Please fill out this field” is shown below the input field respectively.	Pass

Login account with email and password but wrong input format of email	1. Enter email and password 2. Click “Sign In” button	Email: lohyongbin0819gmail.com Password: 123	The error message of “Please include an ‘@’ in the email address. ‘lohyongbin0819gmail.com’ is missing an ‘@’” is shown below the input field with wrong format respectively.	-	The error message of “Please include an ‘@’ in the email address. ‘lohyongbin0819gmail.com’ is missing an ‘@’” is shown below the input field with wrong format respectively.	Pass
Login account with valid but not registered email and correct password	1. Enter email and password 2. Click “Sign In” button	Email: lohyongbin111@gmail.com Password: 123	The error message of “Email does not exist.” is flashed.	-	The error message of “Email does not exist.” is flashed.	Pass
Login account with valid registered email and incorrect password	1. Enter email and password 2. Click “Sign In” button	Email: lohyongbin0819@gmail.com Password: 1234	The error message of “Incorrect password, try again.” is flashed.	-	The error message of “Incorrect password, try again.” is flashed.	Pass

Table 7.3: Unit Test Case for Logout

Test Case ID	TC-003		**Module**		Login and Logout	
Test Title	Logout					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Users have logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Logout the account	1. Click "Logout" button in the navigation menu	-	Account is signed out successfully.	The user should be redirected to Home Page without any account logged in.	Account is signed out successfully. The user is redirect to Home Page without any account logged in.	Pass

Table 7.4: Unit Test Case for View Lecturer

Test Case ID	TC-004		**Module**		Lecturer Management Module	
Test Title	View Lecturer					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View the lecturer list	1. Click the "Lecturer" button in the navigation menu	-	All the lecturer records are displayed.	-	All the lecturer records are displayed.	Pass

Table 7.5: Unit Test Case for Add Lecturer

Test Case ID	TC-005		**Module**		Lecturer Management Module	
Test Title	Add Lecturer					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in to the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Add New Lecturer with valid id, username, email and password	1. Enter valid id, username, email and password 2. Click “Add” button	Id: 2200832 Username: Lecturer Email: 2200832@gmail.com Password: 123	The lecturer is added successfully. A success message of “Lecturer added!” is flashed.	Admin should be redirected to View Lecturer Page.	The lecturer is added successfully. A success message of “Lecturer added!” is flashed.	Pass
Add New Lecturer with empty id, username, email and password	1. Enter id, username, email and password 2. Click “Add” button	Id: null Username: null Email: null Password: null	The error message of “Please fill out this field” is shown below the input field respectively.	No lecturer should be added	The error message of “Please fill out this field” is shown below the input field respectively.	Pass

Add New Lecturer with valid id, username, password but wrong input format of email	1. Enter id, username, email and password 2. Click “Add” button	Id: 2200832 Username: Lecturer Email: 2200832gmail.com Password: 123	The error message of “Please include an ‘@’ in the email address. ‘2200832gmail.com’ is missing an ‘@’” is shown below the input field with wrong format respectively.	No lecturer should be added	The error message of “Please include an ‘@’ in the email address. ‘2200832gmail.com’ is missing an ‘@’” is shown below the input field with wrong format respectively.	Pass
Add New Lecturer with valid id, username, email but invalid length of password which less than 3 characters	1. Enter id, username, email and password 2. Click “Add” button	Id: 2200832 Username: Lecturer Email: 2200832@gmail.com Password: 12	The error message of “Password must be at least 3 characters.” is flashed.	No lecturer should be added	The error message of “Password must be at least 3 characters.” is flashed.	Pass
Add New Lecturer with valid id, username, email	1. Enter id, username, email and password 2. Click “Add” button	Id: 2200832 Username: Lecturer Email: 2200832@gmail.com Password: 123	The error message of “User already exists.” is flashed.	No lecturer should be added	The error message of “User already exists.” is flashed.	Pass

and password but id already existed						
Add New Lecturer with valid id, username, email and password but email already existed	1. Enter id, username, email and password 2. Click “Add” button	Id: 2200832 Username: Lecturer Email: 2200832@gmail.com Password: 123	The error message of “Email already exists.” is flashed.	No lecturer should be added	The error message of “Email already exists.” is flashed.	Pass

Table 7.6: Unit Test Case for Update Lecturer

Test Case ID	TC-006		**Module**	Lecturer Management Module		
Test Title	Update Lecturer					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Update the Lecturer with valid username, email and password	1. Enter valid username, email and password 2. Click "Update" button	Username: Lecturer Email: 2200833@gmail.com Password: 123	The lecturer is updated successfully. A success message of "Lecturer updated!" is flashed.	The lecturer's information should be successfully updated in the system.	The lecturer is updated successfully. A success message of "Lecturer updated!" is flashed.	Pass
Update the Lecturer with empty username, email and password	1. Enter username, email and password 2. Click "Update" button	Username: null Email: null Password: null	The error message of "Please fill out this field" is shown below the input field respectively.	The lecturer's information should remain unchanged.	The error message of "Please fill out this field" is shown below the input field respectively.	Pass

Update the Lecturer with valid username, password but wrong input format of email	1. Enter username, email and password 2. Click "Update" button	Username: Lecturer Email: 2200833gmail.com Password: 123	The error message of "Please include an '@' in the email address. '2200833gmail.com' is missing an '@'" is shown below the input field with wrong format respectively.	The lecturer's information should remain unchanged.	The error message of "Please include an '@' in the email address. '2200833gmail.com' is missing an '@'" is shown below the input field with wrong format respectively.	Pass
Update the Lecturer with valid username, email but invalid length of password which less than 3 characters	1. Enter username, email and password 2. Click "Update" button	Username: Lecturer Email: 2200833@gmail.com Password: 12	The error message of "Password must be at least 3 characters." is flashed.	The lecturer's information should remain unchanged.	The error message of "Password must be at least 3 characters." is flashed.	Pass
Update the Lecturer with valid username, email and password but	1. Enter username, email and password 2. Click "Update" button	Username: Lecturer Email: 2200833@gmail.com Password: 123	The error message of "This email already exists!" is flashed.	The lecturer's information should remain unchanged.	The error message of "This email already exists!" is flashed.	Pass

email already existed						
Update the Lecturer with same username, email and password	1. Remain the same username, email and password 2. Click "Update" button	Username: Lecturer Email: 2200833@gmail.com Password: 123	The error message of "No Changes!" is flashed.	The lecturer's information should remain unchanged.	The error message of "No Changes!" is flashed.	Pass

Table 7.7: Unit Test Case for Delete Lecturer

Test Case ID	TC-007		**Module**		Lecturer Management Module	
Test Title	Delete Lecturer					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Delete the specific lecturer which do not have any enrolment	1. Click the “Delete” button of the specific lecturer which do not have any enrolment	-	The lecturer is deleted successfully. A success message of “Lecturer deleted!” is flashed.	The lecturer should be successfully removed from the system.	The lecturer is deleted successfully. A success message of “Lecturer deleted!” is flashed.	Pass
Delete the specific lecturer which have one or more enrolment	1. Click the “Delete” button of the specific lecturer which have one or more enrolment	-	The error message of “Cannot delete the lecturer as it had enrolment of the course!” is flashed.	The lecturer should remain undeleted due to existing enrolments.	The error message of “Cannot delete the lecturer as it had enrolment of the course!” is flashed.	Pass

Table 7.8: Unit Test Case for View Student

Test Case ID	TC-008		**Module**	Student Management Module		
Test Title	View Student					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View the student list	1. Click the "Student" button in the navigation menu	-	All the student records are displayed.	-	All the student records are displayed.	Pass

Table 7.9: Unit Test Case for Add Student

Test Case ID	TC-009		**Module**		Lecturer Management Module	
Test Title	Add Student					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in to the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Add New Student with valid id, username, email and password	1. Enter valid id, username, email and password 2. Click "Add" button	Id: 2200833 Username: Student Email: 2200833@gmail.com Password: 123	The student is added successfully. A success message of "Student added!" is flashed.	Admin should be redirected to View Student Page.	The student is added successfully. A success message of "Student added!" is flashed.	Pass
Add New Student with empty id, username, email and password	1. Enter id, username, email and password 2. Click "Add" button	Id: null Username: null Email: null Password: null	The error message of "Please fill out this field" is shown below the input field respectively.	No student should be added	The error message of "Please fill out this field" is shown below the input field respectively.	Pass
Add New Student with valid id, username,	1. Enter id, username, email and password	Id: 2200833 Username: Student	The error message of "Please include an '@' in the email address.	No student should be added	The error message of "Please include an '@' in the email address.	Pass

password but wrong input format of email	2. Click "Add" button	Email: 2200833gmail.com Password: 123	'2200833gmail.com' is missing an '@'" is shown below the input field with wrong format respectively.		'2200833gmail.com' is missing an '@'" is shown below the input field with wrong format respectively.	
Add New Student with valid id, username, email but invalid length of password which less than 3 characters	1. Enter id, username, email and password 2. Click "Add" button	Id: 2200833 Username: Student Email: 2200833@gmail.com Password: 12	The error message of "Password must be at least 3 characters." is flashed.	No student should be added	The error message of "Password must be at least 3 characters." is flashed.	Pass
Add New Student with valid id, username, email and password but id already existed	1. Enter id, username, email and password 2. Click "Add" button	Id: 2200833 Username: Student Email: 2200833@gmail.com Password: 123	The error message of "User already exists." is flashed.	No student should be added	The error message of "User already exists." is flashed.	Pass
Add New Student with valid id, username, email and password but	1. Enter id, username, email and password	Id: 2200833 Username: Student Email: 2200833@gmail.com	The error message of "Email already exists." is flashed.	No student should be added	The error message of "Email already exists." is flashed.	Pass

email already existed	2. Click “Add” button	Password: 123				

Table 7.10: Unit Test Case for Update Student

Test Case ID	TC-010		**Module**	Student Management Module		
Test Title	Update Student					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Update the Student with valid username, email and password	1. Enter valid username, email and password 2. Click "Update" button	Username: Student Email: 2200834@gmail.com Password: 123	The student is updated successfully. A success message of "Student updated!" is flashed.	The student's information should be successfully updated in the system.	The student is updated successfully. A success message of "Student updated!" is flashed.	Pass
Update the Student with empty username, email and password	1. Enter username, email and password 2. Click "Update" button	Username: null Email: null Password: null	The error message of "Please fill out this field" is shown below the input field respectively.	The student's information should remain unchanged.	The error message of "Please fill out this field" is shown below the input field respectively.	Pass

Update the Student with valid username, password but wrong input format of email	1. Enter username, email and password 2. Click "Update" button	Username: Student Email: 2200834gmail.com Password: 123	The error message of "Please include an '@' in the email address. '2200834gmail.com' is missing an '@'" is shown below the input field with wrong format respectively.	The student's information should remain unchanged.	The error message of "Please include an '@' in the email address. '2200834gmail.com' is missing an '@'" is shown below the input field with wrong format respectively.	Pass
Update the Student with valid username, email but invalid length of password which less than 3 characters	1. Enter username, email and password 2. Click "Update" button	Username: Student Email: 2200834@gmail.com Password: 12	The error message of "Password must be at least 3 characters." is flashed.	The student's information should remain unchanged.	The error message of "Password must be at least 3 characters." is flashed.	Pass
Update the Student with valid username, email and password but	1. Enter username, email and password 2. Click "Update" button	Username: Student Email: 2200834@gmail.com Password: 123	The error message of "This email already exists!" is flashed.	The student's information should remain unchanged.	The error message of "This email already exists!" is flashed.	Pass

email already existed						
Update the Student with same username, email and password	1. Remain the same username, email and password 2. Click "Update" button	Username: Student Email: 2200834@gmail.com Password: 123	The error message of "No Changes!" is flashed.	The student's information should remain unchanged.	The error message of "No Changes!" is flashed.	Pass

Table 7.11: Unit Test Case for Delete Student

Test Case ID	TC-011		**Module**		Student Management Module	
Test Title	Delete Student					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Delete the specific student which do not have any enrolment	1. Click the "Delete" button of the specific student which do not have any enrolment	-	The student is deleted successfully. A success message of "Student deleted!" is flashed.	The student should be successfully removed from the system.	The student is deleted successfully. A success message of "Student deleted!" is flashed.	Pass
Delete the specific student which have one or more enrolment	1. Click the "Delete" button of the specific student which have one or more enrolment	-	The error message of "Cannot delete the student as it had enrolment of the course!" is flashed.	The student should remain undeleted due to existing enrolments.	The error message of "Cannot delete the student as it had enrolment of the course!" is flashed.	Pass

Table 7.12: Unit Test Case for View Room

Test Case ID	TC-012		**Module**		Room Management Module	
Test Title	View Room					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View the room list	1. Click the "Room" button in the navigation menu	-	All the room records are displayed.	-	All the room records are displayed.	Pass

Table 7.13: Unit Test Case for Add Room

Test Case ID	TC-013		**Module**	Room Management Module		
Test Title	Add Room					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin has logged in to the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Add New Room with valid id, type, capacity, projector, whiteboard, computer, sound_system and video_conferencing	1. Enter valid id, type, capacity, projector, whiteboard, computer, sound_system and video_conferencing 2. Click "Add" button	Id: KB203 Type: LAB Capacity: 25 Projector: True Whiteboard: True Computer: True Sound_system: True Video_conferencing: False	The room is added successfully. A success message of "Room added!" is flashed.	Admin should be redirected to View Room Page.	The room is added successfully. A success message of "Room added!" is flashed.	Pass
Add New Room with empty id, type, capacity, projector, whiteboard,	1. Enter id, type, capacity, projector, whiteboard, computer,	Id: null Type: null Capacity: null Projector: null	The error message of "Please fill out this field" and "Please select an	No room should be added	The error message of "Please fill out this field" and "Please select an item in the list" are	Pass

computer, sound_system and video_conferencing	sound_system and video_conferencing 2. Click “Add” button	Whiteboard: null Computer: null Sound_system: null Video_conferencing: null	item in the list” are shown below the input field respectively.		shown below the input field respectively.	
Add New Room with valid id, type, capacity, projector, whiteboard, computer, sound_system and video_conferencing but id already existed	1. Enter id, type, capacity, projector, whiteboard, computer, sound_system and video_conferencing 2. Click “Add” button	Id: KB203 Type: LAB Capacity: 25 Projector: True Whiteboard: True Computer: True Sound_system: True Video_conferencing: False	The error message of “Room already exists.” is flashed.	No room should be added	The error message of “Room already exists.” is flashed.	Pass

Table 7.14: Unit Test Case for Update Room

Test Case ID	TC-014		**Module**		Room Management Module	
Test Title	Update Room					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Update the Room with valid type, capacity, projector, whiteboard, computer, sound_system and video_conferencing	1. Enter valid type, capacity, projector, whiteboard, computer, sound_system and video_conferencing 2. Click “Update” button	Type: LAB Capacity: 30 Projector: False Whiteboard: True Computer: True Sound_system: True Video_conferencing: False	The room is updated successfully. A success message of “Room updated!” is flashed.	The room’s information should be successfully updated in the system.	The room is updated successfully. A success message of “Room updated!” is flashed.	Pass
Update the Room with empty type, capacity, projector,	1. Enter type, capacity, projector, whiteboard,	Type: null Capacity: null null	The error message of “Please fill out this field” and “Please select	The room’s information should	The error message of “Please fill out this field” and “Please select an	Pass

whiteboard, computer, sound_system and video_conferencing	computer, sound_system and video_conferencing 2. Click “Update” button	Projector: null Whiteboard: null Computer: null Sound_system: null Video_conferencing: null	an item in the list” are shown below the input field respectively.	remain unchanged.	item in the list” are shown below the input field respectively.	
Update the Room with same type, capacity, projector, whiteboard, computer, sound_system and video_conferencing	1. Remain the same type, capacity, projector, whiteboard, computer, sound_system and video_conferencing 2. Click “Update” button	Type: LAB Capacity: 30 Projector: False Whiteboard: True Computer: True Sound_system: True Video_conferencing: False	The error message of “No Changes!” is flashed.	The room’s information should remain unchanged.	The error message of “No Changes!” is flashed.	Pass

Table 7.15: Unit Test Case for Delete Room

Test Case ID	TC-015		**Module**		Room Management Module	
Test Title	Delete Room					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Delete the specific room which do not be assigned to any assessment	1. Click the "Delete" button of the specific room which do not be assigned to any assessment	-	The room is deleted successfully. A success message of "Room deleted!" is flashed.	The room should be successfully removed from the system.	The room is deleted successfully. A success message of "Room deleted!" is flashed.	Pass
Delete the specific room which have assigned to one or more assessment	1. Click the "Delete" button of the specific student which have assigned to one or more assessment	-	The error message of "Cannot delete the room as it is used in assessment!" is flashed.	The room should remain undeleted.	The error message of "Cannot delete the room as it is used in assessment!" is flashed.	Pass

Table 7.16: Unit Test Case for View Course

Test Case ID	TC-016		**Module**		Course Management Module	
Test Title	View Course					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View the course list	1. Click the "Course" button in the navigation menu	-	All the course records are displayed.	-	All the course records are displayed.	Pass

Table 7.17: Unit Test Case for Add Course

Test Case ID	TC-017		**Module**	Course Management Module		
Test Title	Add Course					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin has logged in to the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Add New Course with valid id and info	1. Enter valid id and info 2. Click “Add” button	Id: UECS3599 Info: Project	The course is added successfully. A success message of “Course added!” is flashed.	Admin should be redirected to View Course Page.	The course is added successfully. A success message of “Course added!” is flashed.	Pass
Add New Course with empty id and info	1. Enter valid id and info 2. Click “Add” button	Id: null Info: null	The error message of “Please fill out this field” is shown below the input field respectively.	No course should be added	The error message of “Please fill out this field” is shown below the input field respectively.	Pass
Add New Room with valid id and info but id already existed	1. Enter valid id and info 2. Click “Add” button	Id: UECS3599 Info: Project	The error message of “Course already exists.” is flashed.	No course should be added	The error message of “Course already exists.” is flashed.	Pass

Table 7.18: Unit Test Case for Update Course

Test Case ID	TC-018		**Module**		Course Management Module	
Test Title	Update Course					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Update the Course with valid info	1. Enter valid info 2. Click “Update” button	Info: Project 2	The course is updated successfully. A success message of “Course updated!” is flashed.	The course’s information should be successfully updated in the system.	The course is updated successfully. A success message of “Course updated!” is flashed.	Pass
Update the Course with empty info	1. Enter empty info 2. Click “Update” button	Info: null	The error message of “Please fill out this field” is shown below the input field.	The course’s information should remain unchanged.	The error message of “Please fill out this field” is shown below the input field.	Pass
Update the Course with same info	1. Remain the same info 2. Click “Update” button	Info: Project 2	The error message of “No Changes!” is flashed.	The course’s information should remain unchanged.	The error message of “No Changes!” is flashed.	Pass

Table 7.19: Unit Test Case for Delete Course

Test Case ID	TC-019		**Module**		Course Management Module	
Test Title	Delete Course					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Delete the specific course which do not have any course timetable, enrolment and assessment	1. Click the "Delete" button of the specific course which do not have any course timetable, enrolment and assessment	-	The course is deleted successfully. A success message of "Course deleted!" is flashed.	The course should be successfully removed from the system.	The course is deleted successfully. A success message of "Course deleted!" is flashed.	Pass
Delete the specific course which have one or more course timetable	1. Click the "Delete" button of the specific course which have one or more course timetable	-	The error message of "Cannot delete the course as it is used in course timetable!" is flashed.	The course should remain undeleted due to having course timetable.	The error message of "Cannot delete the course as it is used in course timetable!" is flashed.	Pass

Delete the specific course which have one or more course enrolment	1. Click the “Delete” button of the specific course which have one or more course enrolment	-	The error message of “Cannot delete the course as it is enrolled by user!” is flashed.	The course should remain undeleted.	The error message of “Cannot delete the course as it is enrolled by user!” is flashed.	Pass
Delete the specific course which have one or more assessment of the specific course	1. Click the “Delete” button of the specific course which have one or more assessment of the specific course	-	The error message of “Cannot delete the course as it is used in assessment!” is flashed.	The course should remain undeleted.	The error message of “Cannot delete the course as it is used in assessment!” is flashed.	Pass

Table 7.20: Unit Test Case for View Course Timetable

Test Case ID	TC-020		**Module**		Timetable Management Module	
Test Title	View Course Timetable					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account. System has the course records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View the course timetable of the specific course	1. Click the “Course” button in the navigation menu to view the course 2. Click the “View” button under Timetable column of the specific course	-	All the course timetable records of the specific course are displayed.	-	All the course timetable records of the specific course are displayed.	Pass

Table 7.21: Unit Test Case for Add Course Timetable

Test Case ID	TC-021		**Module**		Timetable Management Module	
Test Title	Add Course Timetable					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in to the account. System has the course records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Add New Course Timetable with valid session, weekday, start time and end time while Session is ‘Lecture’	1. Enter valid session, weekday, start time and end time 2. Click “Add” button	Session: Lecture Weekday: Monday Start Time: 8:00AM End Time: 10:00AM	The course timetable is added successfully. A success message of “Timetable added!” is flashed.	Admin should be redirected to View Course Timetable Page of the specific course.	The course timetable is added successfully. A success message of “Timetable added!” is flashed.	Pass
Add New Course Timetable with valid	1. Enter valid session, session	Session: Practical	The course timetable is added successfully. A	Admin should be redirected	The course timetable is added successfully. A	Pass

session, session no, weekday, start time and end time while Session is not 'Lecture'	no, weekday, start time and end time 2. Click "Add" button	Session No: 1 Weekday: Tuesday Start Time: 8:00AM End Time: 10:00AM	success message of "Timetable added!" is flashed.	to View Course Timetable Page of the specific course.	success message of "Timetable added!" is flashed.	
Add New Course Timetable with empty session, weekday, start time and end time	1. Enter empty session, weekday, start time and end time 2. Click "Add" button	Session: null Weekday: null Start Time: null End Time: null	The error message of "Please fill out this field" and "Please select an item in the list" is shown below the input field respectively.	No course timetable should be added	The error message of "Please fill out this field" and "Please select an item in the list" is shown below the input field respectively.	Pass
Add New Course Timetable with valid session, session no, weekday, start time and end time while Session is 'Practical'	1. Enter valid session, session no, weekday, start time and end time 2. Click "Add" button	Session: Practical Session No: 1 Weekday: Tuesday	The error message of "Timetable already exists." is flashed.	No course timetable should be added	The error message of "Timetable already exists." is flashed.	Pass

or ‘Tutorial’ but overlap with the existing timetable record with same session, session no		Start Time: 8:00AM End Time: 10:00AM				
Add New Course Timetable with valid session, session no, weekday, start time and end time but the time is overlap with other timetable record of different session. (User must attend all different sessions so cannot be overlap)	1. Enter valid session, session no, weekday, start time and end time 2. Click “Add” button	Session: Tutorial Session No: 2 Weekday: Tuesday Start Time: 8:00AM End Time: 10:00AM	The error message of “The new timetable overlaps with other session or Lecture session timetable.” is flashed.	No course timetable should be added	The error message of “The new timetable overlaps with other session or Lecture session timetable.” is flashed.	Pass
Add New Course Timetable with valid session, session no,	1. Enter valid session, weekday,	Session: Lecture	The error message of “The new timetable overlaps with other	No course timetable	The error message of “The new timetable overlaps with other	Pass

weekday, start time and end time but the time is overlap with other timetable record of same session when session is 'Lecture'. (User must attend all 'Lecture' sessions so time of all 'Lecture' sessions cannot be overlap)	start time and end time 2. Click "Add" button	Weekday: Monday Start Time: 9:00AM End Time: 11:00AM	session or Lecture session timetable." is flashed.	should be added	session or Lecture session timetable." is flashed.	

Table 7.22: Unit Test Case for Update Course Timetable

Test Case ID	TC-022		**Module**	Timetable Management Module		
Test Title	Update Course Timetable					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin has logged in to the account. System has the course records. System has the course timetable records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Update the specific Course Timetable with valid session, weekday, start time and end time while Session is ‘Lecture’	1. Enter valid session, weekday, start time and end time 2. Click “Update” button	Session: Lecture Weekday: Monday Start Time: 8:00AM End Time: 10:00AM	The course timetable is updated successfully. A success message of “Timetable updated!” is flashed.	Admin should be redirected to View Course Timetable Page of the specific course.	The course timetable is updated successfully. A success message of “Timetable updated!” is flashed.	Pass

Update the specific Course Timetable with valid session, session no, weekday, start time and end time while Session is ‘Practical’ or ‘Tutorial’	1. Enter valid session, session no, weekday, start time and end time 2. Click “Update” button	Session: Practical Session No: 1 Weekday: Tuesday Start Time: 8:00AM End Time: 10:00AM	The course timetable is updated successfully. A success message of “Timetable updated!” is flashed.	Admin should be redirected to View Course Timetable Page of the specific course.	The course timetable is updated successfully. A success message of “Timetable updated!” is flashed.	Pass
Update the specific Course Timetable with empty session, weekday, start time and end time	1. Enter empty session, weekday, start time and end time 2. Click “Update” button	Session: null Weekday: null Start Time: null End Time: null	The error message of “Please fill out this field” and “Please select an item in the list” is shown below the input field respectively.	Course timetable should not be updated	The error message of “Please fill out this field” and “Please select an item in the list” is shown below the input field respectively.	Pass
Update the specific Course Timetable with valid session, session no,	1. Enter valid session, session no, weekday, start time and end time	Session: Practical Session No: 1	The error message of “Timetable for the same session and session no	Course timetable should not be updated	The error message of “Timetable for the same session and session no	Pass

weekday, start time and end time while Session is ‘Practical’ or ‘Tutorial’ but overlap with the existing timetable record with same session, session no	2. Click “Update” button	Weekday: Tuesday Start Time: 9:00AM End Time: 11:00AM	already exists!” is flashed.		already exists!” is flashed.	
Update the specific Course Timetable with valid session, session no, weekday, start time and end time but the time is overlap with other timetable record of different session. (User must attend all different sessions so cannot be overlap)	1. Enter valid session, session no, weekday, start time and end time 2. Click “Update” button	Session: Tutorial Session No: 2 Weekday: Tuesday Start Time: 8:00AM End Time: 10:00AM	The error message of “The updated timetable overlaps with other session or Lecture session timetable.” is flashed.	Course timetable should not be updated	The error message of “The updated timetable overlaps with other session or Lecture session timetable.” is flashed.	Pass

Update the specific Course Timetable with valid session, session no, weekday, start time and end time but the time is overlap with other timetable record of same session when session is 'Lecture'. (User must attend all 'Lecture' sessions so time of all 'Lecture' sessions cannot be overlap)	1. Enter valid session, weekday, start time and end time 2. Click "Update" button	Session: Lecture Weekday: Monday Start Time: 9:00AM End Time: 11:00AM	The error message of "The updated timetable overlaps with other session or Lecture session timetable." is flashed.	Course timetable should not be updated	The error message of "The updated timetable overlaps with other session or Lecture session timetable." is flashed.	Pass
Update the specific Course Timetable with same session, session no,	1. Remain the same session, session no, weekday, start time and end time	Session: Practical Session No: 1 Weekday: Tuesday	The error message of "No Changes!" is flashed.	The course timetable's information should remain unchanged.	The error message of "No Changes!" is flashed.	Pass

weekday, start time and end time	2. Click “Update” button	Start Time: 8:00AM End Time: 10:00AM				

Table 7.23: Unit Test Case for Delete Course Timetable

Test Case ID	TC-023		**Module**		Timetable Management Module	
Test Title	Delete Course Timetable					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account. System has the course records. System has the course timetable records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Delete the specific course timetable which do not have any course enrolment of same course, session, session no with this specific course timetable	1. Click the “Delete” button of the specific course timetable which do not have any course enrolment of same course, session, session no with this specific course timetable	-	The course timetable is deleted successfully. A success message of “Timetable deleted!” is flashed.	The course timetable should be successfully removed from the system.	The course timetable is deleted successfully. A success message of “Timetable deleted!” is flashed.	Pass

Delete the specific course timetable which have one or more course enrolment of same course, session, session no with this specific course timetable	1. Click the “Delete” button of the specific course timetable which have one or more course enrolment of same course, session, session no with this specific course timetable	-	The error message of “Cannot delete timetable as it has enrolments.” is flashed.	The course timetable should remain undeleted due to having course enrolment with this course timetable.	The error message of “Cannot delete timetable as it has enrolments.” is flashed.	Pass

Table 7.24: Unit Test Case for View Course Enrolment

Test Case ID	TC-024		**Module**		Enrolment Management Module	
Test Title	View Course Enrolment					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account. System has the course records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View the course enrolment of the specific course for student	1. Click the “Course” button in the navigation menu to view the course 2. Click the “View” button under Enrolment column of the specific course	-	The course enrolment records of the specific course for student are displayed.	-	The course enrolment records of the specific course for student are displayed.	Pass
View the course enrolment of the specific course for lecturer	1. Click the “Course” button in the navigation menu to view the course 2. Click the “View” button under Enrolment column of the specific course 3. Click the “Lecturer” tab in the Course Enrolment list	-	The course enrolment records of the specific course for lecturer are displayed.	-	The course enrolment records of the specific course for lecturer are displayed.	Pass

Table 7.25: Unit Test Case for Add Course Enrolment

Test Case ID	TC-025		**Module**		Enrolment Management Module	
Test Title	Add Course Enrolment					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in to the account. System has the course records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Add New Course Enrolment with valid user type, user and session while Session is 'Lecture'	1. Enter valid user type, user and session 2. Click "Add" button	User Type: Student User: 2200831 Session: Lecture	The course enrolment is added successfully. A success message of "Enrolment added!" is flashed.	Admin should be redirected to View Course Enrolment Page of the specific course.	The course enrolment is added successfully. A success message of "Enrolment added!" is flashed.	Pass
Add New Course Enrolment with valid user type, user and session while	1. Enter valid user type, user, session and session no 2. Click "Add" button	User Type: Student User: 2200831 Session: Practical	The course enrolment is added successfully. A success message of "Enrolment added!" is flashed.	Admin should be redirected to View Course Enrolment Page	The course enrolment is added successfully. A success message of "Enrolment added!" is flashed.	Pass

Session is 'Practical' or 'Tutorial'		Session no: 1		of the specific course.		
Add New Course Enrolment with empty user type, user and session	1. Enter empty user type, user and session 2. Click "Add" button	User Type: null User: null Session: null	The error message of "Please select an item in the list" is shown below the input field respectively.	No course enrolment should be added	The error message of "Please select an item in the list" is shown below the input field respectively.	Pass
Add New Course Enrolment with valid user type, user, session and session no but enrolment is overlap with existing enrolment with the same course, user, session and session no	1. Enter valid user type, user and session, session no 2. Click "Add" button	User Type: Student User: 2200831 Session: Practical Session no: 1	The error message of "The user had been enrolled in the session of the course." is flashed.	No course enrolment should be added	The error message of "The user had been enrolled in the session of the course." is flashed.	Pass
Add New Course Enrolment with valid user type, user,	1. Enter valid user type, user and	User Type: Student User: 2200831	The error message of "Course without	No course enrolment should be added	The error message of "Course without	Pass

session and session no but the timetable for the specific course, session and session no does not exist	session, session no 2. Click “Add” button	Session: Practical Session no: 1	timetable cannot be enrolled.” is flashed.		timetable cannot be enrolled.” is flashed.	
Add New Course Enrolment with valid user type, user, session and session no but the enrolment is clashed with other enrolment (clashed with user timetable)	1. Enter valid user type, user and session, session no 2. Click “Add” button	User Type: Student User: 2200831 Session: Practical Session no: 2	The error message of “The course timetable is clashed with student timetable.” is flashed.	No course enrolment should be added	The error message of “The course timetable is clashed with student timetable.” is flashed.	Pass

Table 7.26: Unit Test Case for Update Course Enrolment

Test Case ID	TC-026		**Module**		Enrolment Management Module	
Test Title	Update Course Enrolment					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in to the account. System has the course records. System has the course enrolment records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Update the Course Enrolment with valid user type, user and session while Session is 'Lecture'	1. Enter valid user type, user and session 2. Click "Update" button	User Type: Student User: 2200831 Session: Lecture	The course enrolment is updated successfully. A success message of "Enrolment updated!" is flashed.	Admin should be redirected to View Course Enrolment Page of the specific course.	The course enrolment is updated successfully. A success message of "Enrolment updated!" is flashed.	Pass
Update the Course Enrolment with valid user type, user and session while	1. Enter valid user type, user, session and session no	User Type: Student User: 2200831	The course enrolment is updated successfully. A success message of	Admin should be redirected to View Course Enrolment Page	The course enrolment is updated successfully. A success message of	Pass

Session is ‘Practical’ or ‘Tutorial’	2. Click “Update” button	Session: Practical Session no: 1	“Enrolment updated!” is flashed.	of the specific course.	“Enrolment updated!” is flashed.	
Update the Course Enrolment with empty user type, user and session	1. Enter empty user type, user and session 2. Click “Update” button	User Type: null User: null Session: null	The error message of “Please select an item in the list” is shown below the input field respectively.	Course enrolment should not be updated	The error message of “Please select an item in the list” is shown below the input field respectively.	Pass
Update the Course Enrolment with valid user type, user, session and session no but enrolment is overlap with existing enrolment with the same course, user, session and session no	1. Enter valid user type, user and session, session no 2. Click “Update” button	User Type: Student User: 2200831 Session: Practical Session no: 1	The error message of “The user had been enrolled in the session of the course.” is flashed.	Course enrolment should not be updated	The error message of “The user had been enrolled in the session of the course.” is flashed.	Pass

Update the Course Enrolment with valid user type, user, session and session no but the timetable for the specific course, session and session no does not exist	1. Enter valid user type, user and session, session no 2. Click "Update" button	User Type: Student User: 2200831 Session: Practical Session no: 1	The error message of "Course without timetable cannot be enrolled." is flashed.	Course enrolment should not be updated	The error message of "Course without timetable cannot be enrolled." is flashed.	Pass
Update the Course Enrolment with valid user type, user, session and session no but the enrolment is clashed with other enrolment (clashed with user timetable)	1. Enter valid user type, user and session, session no 2. Click "Update" button	User Type: Student User: 2200831 Session: Practical Session no: 2	The error message of "The course timetable is clashed with student timetable." is flashed.	Course enrolment should not be updated	The error message of "The course timetable is clashed with student timetable." is flashed.	Pass
Update the specific Course Enrolment with same user type,	1. Remain the same user type, user, session and session no	User Type: Student User: 2200831	The error message of "No Changes!" is flashed.	The course enrolment's information	The error message of "No Changes!" is flashed.	Pass

user, session and session no	2. Click “Update” button	Session: Practical Session no: 1		should remain unchanged.		

Table 7.27: Unit Test Case for Delete Course Enrolment

Test Case ID	TC-027		**Module**		Enrolment Management Module	
Test Title	Delete Course Enrolment					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin has logged in the account. System has the course records. System has the course enrolment records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Delete the specific course enrolment	1. Click the “Delete” button of the specific course enrolment	-	The course enrolment is deleted successfully. A success message of “Enrolment deleted!” is flashed.	The course enrolment should be successfully removed from the system.	The course enrolment is deleted successfully. A success message of “Enrolment deleted!” is flashed.	Pass

Table 7.28: Unit Test Case for View Assessment

Test Case ID	TC-028		**Module**	Assessment Management Module		
Test Title	View Assessment					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin/ Lecturer/Student has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View all the assessment for Administrator	1. Click the "Assessment" button in the navigation menu to view the assessment	-	All the assessment records are displayed.	-	All the course assessment records are displayed.	Pass
View all the past assessment for Administrator	1. Click the "Assessment" button in the navigation menu to view the assessment 2. Click the "Past" tab in the assessment list		All the past assessment records are displayed.	-	All the past assessment records are displayed.	Pass
View all the active assessment for Administrator	1. Click the "Assessment" button in the navigation menu to view the assessment 2. Click the "Active" tab in the assessment list		All the active assessment records are displayed.	-	All the active assessment records are displayed.	Pass
View all the assessment which	1. Click the "Assessment" button in the navigation menu to view the assessment	-	All the assessment records which the	-	All the assessment records which the	Pass

Lecturer enrolled in for Lecturer			lecturer enrolled in are displayed.		lecturer enrolled in are displayed.	
View all the past assessment which Lecturer enrolled in for Lecturer	1. Click the "Assessment" button in the navigation menu to view the assessment 2. Click the "Past" tab in the assessment list	-	All the past assessment records which the lecturer enrolled in are displayed.	-	All the past assessment records which the lecturer enrolled in are displayed.	Pass
View all the active assessment which Lecturer enrolled in for Lecturer	1. Click the "Assessment" button in the navigation menu to view the assessment 2. Click the "Active" tab in the assessment list	-	All the active assessment records which the lecturer enrolled in are displayed.	-	All the active assessment records which the lecturer enrolled in are displayed.	Pass
View all the assessment which Student enrolled in for Student	1. Click the "Assessment" button in the navigation menu to view the assessment	-	All the assessment records which the student enrolled in are displayed.	-	All the assessment records which the student enrolled in are displayed.	Pass
View all the past assessment which Student enrolled in for Student	1. Click the "Assessment" button in the navigation menu to view the assessment 2. Click the "Past" tab in the assessment list	-	All the past assessment records which the student enrolled in are displayed.	-	All the past assessment records which the student enrolled in are displayed.	Pass

View all the active assessment which Student enrolled in for Student	1. Click the “Assessment” button in the navigation menu to view the assessment 2. Click the “Active” tab in the assessment list	-	All the active assessment records which the student enrolled in are displayed.	-	All the active assessment records which the student enrolled in are displayed.	Pass

Table 7.29: Unit Test Case for Add Assessment

Test Case ID	TC-029		**Module**		Assessment Management Module	
Test Title	Add Assessment					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin/Lecturer has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Add New Assessment with valid course, info, date, equipment requirement, start time, end time and room	1. Enter valid course, info, date, equipment requirement, start time, end time and room 2. Click "Add" button	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 8:00AM End Time: 10:00AM Room: KB203	The assessment is added successfully. A success message of "Assessment added!" is flashed.	Admin should be redirected to View Assessment Page of the specific course.	The assessment is added successfully. A success message of "Assessment added!" is flashed.	Pass

Add New Assessment with empty course, info, date, equipment requirement, start time, end time and room	1. Enter empty course, info, date, equipment requirement, start time, end time and room 2. Click “Add” button	Course: null Info: null Date: null Projector: null Whiteboard: null Computer: null Whiteboard: null Sound System: null Video Conferencing: null Start Time: null End Time: null Room: null	The error message of “Please select an item in the list” and “Please fill out this field” are shown below the input field respectively.	No assessment should be added	The error message of “Please select an item in the list” and “Please fill out this field” are shown below the input field respectively.	Pass
Add New Assessment with valid course, info, date, equipment requirement, start time, end time and room but end time is before the start time	1. Enter valid course, info, date, equipment requirement, start time, end time and room 2. Click “Add” button	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False	The error message of “End Time cannot before or same as Start Time.” is flashed.	No assessment should be added	The error message of “End Time cannot before or same as Start Time.” is flashed.	Pass

		Start Time: 10:00AM End Time: 8:00AM Room: KB203				

Table 7.30: Unit Test Case for Update Assessment

Test Case ID	TC-030		**Module**		Assessment Management Module	
Test Title	Update Assessment					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin/Lecturer has logged in the account. System has the assessment records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Update the Assessment with valid info, date, equipment requirement, start time, end time and room	1. Enter valid info, date, equipment requirement, start time, end time and room 2. Click "Update" button	Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 8:00AM End Time: 10:00AM Room: KB203	The assessment is updated successfully. A success message of "Assessment updated!" is flashed.	Admin should be redirected to View Assessment Page of the specific course.	The assessment is updated successfully. A success message of "Assessment updated!" is flashed.	Pass

Update the Assessment with empty info, date, equipment requirement, start time, end time and room	1. Enter empty course, info, date, equipment requirement, start time, end time and room 2. Click "Update" button	Info: null Date: null Projector: null Whiteboard: null Computer: null Whiteboard: null Sound System: null Video Conferencing: null Start Time: null End Time: null Room: null	The error message of "Please select an item in the list" and "Please fill out this field" are shown below the input field respectively.	The assessment should not be updated	The error message of "Please select an item in the list" and "Please fill out this field" are shown below the input field respectively.	Pass
Update the Assessment with valid info, date, equipment requirement, start time, end time and room but end time is before the start time	1. Enter valid course, info, date, equipment requirement, start time, end time and room 2. Click "Update" button	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 10:00AM	The error message of "End Time cannot before or same as Start Time." is flashed.	The assessment should not be updated	The error message of "End Time cannot before or same as Start Time." is flashed.	Pass

		End Time: 8:00AM Room: KB203				
Update the specific Assessment with same info, date, equipment requirement, start time, end time and room	1. Remain the same info, date, equipment requirement, start time, end time and room 2. Click “Update” button	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 8:00AM End Time: 10:00AM Room: KB203	The error message of “No Changes!” is flashed.	The assessment information should remain unchanged.	The error message of “No Changes!” is flashed.	Pass

Table 7.31: Unit Test Case for Delete Assessment

Test Case ID	TC-031		**Module**		Assessment Management Module	
Test Title	Delete Assessment					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin/Lecturer has logged in the account. System has the assessment records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Delete the specific assessment	1. Click the "Delete" button of the specific assessment	-	The assessment is deleted successfully. A success message of "Assessment deleted!" is flashed.	The assessment should be successfully removed from the system.	The assessment is deleted successfully. A success message of "Assessment deleted!" is flashed.	Pass
Delete the specific past assessment	1. Click the "Delete" button of the specific past assessment	-	The error message of "Cannot delete past assessment!" is flashed.	The assessment should not be deleted	The error message of "Cannot delete past assessment!" is flashed.	Pass

Table 7.32: Unit Test Case for View Assessment User

Test Case ID	TC-032		**Module**	Assessment Management Module		
Test Title	View Assessment User					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin/ Lecturer has logged in the account. System has the assessment records.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View all the users enrolled in the assessment	1. Click the "View User" button of the specific assessment	-	All the records of users which enrolled in the assessment are displayed.	-	All the records of users which enrolled in the assessment are displayed.	Pass

Table 7.33: Unit Test Case for Find Room

Test Case ID	TC-033		**Module**		Find Slot and Room Module	
Test Title	Find the available Room during Creating or Updating Assessment Process					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin/ Lecturer has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Find the Available Room with valid course, info, date, equipment requirement, start time, end time	1. Enter valid course, info, date, equipment requirement, start time, end time 2. Click the "Find Room" button to find the available room for the specific duration of time	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 8:00AM End Time: 10:00AM	Pop Up a Modal which display all the available room with room details	-	Pop Up a Modal which display all the available room with room details	Pass
Find the Available Room	1. Enter empty course, info, date, equipment	Course: null Info: null	The alert message of	The modal which	The alert message of	Pass

with empty course, info, date, equipment requirement, start time, end time	requirement, start time, end time 2. Click the "Find Room" button to find the available room for the specific duration of time	Date: null Projector: null Whiteboard: null Computer: null Whiteboard: null Sound System: null Video Conferencing: null Start Time: null End Time: null	"Please fill in all the fields before find the room." is prompted.	display the available room with room details should not be popped up.	"Please fill in all the fields before find the room." is prompted.	
Find the Available Room with valid course, info, date, equipment requirement, start time, end time but start time is before current time	1. Enter valid course, info, date, equipment requirement, start time, end time 2. Click the "Find Room" button to find the available room for the specific duration of time	Current Date: 24/04/2024 Current Time: 2:00PM Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False	The alert message of "Start time must be after the current time." is prompted.	The modal which displays the available room with room details should not be popped up.	The alert message of "Start time must be after the current time." is prompted.	Pass

		Start Time: 1:00PM End Time: 3:00PM				
Find the Available Room with valid course, info, date, equipment requirement, start time, end time but end time is before start time	1. Enter valid course, info, date, equipment requirement, start time, end time 2. Click the “Find Room” button to find the available room for the specific duration of time	Current Date: 24/04/2024 Current Time: 2:00PM Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 3:00PM End Time: 1:00PM	The alert message of “End time cannot be before the start time.” is prompted.	The modal which displays the available room with room details should not be popped up.	The alert message of “End time cannot be before the start time.” is prompted.	Pass
Find the Available Room with valid course, info, date, equipment	1. Enter valid course, info, date, equipment requirement, start time, end time	Current Date: 24/04/2024 Current Time: 2:00PM Course: UECS2344 Info: MidTerm Test	The alert message of “Start time and end time cannot be the	The modal which displays the available	The alert message of “Start time and end time cannot	Pass

requirement, start time, end time but start time is same as end time	2. Click the "Find Room" button to find the available room for the specific duration of time	Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 3:00PM End Time: 3:00PM	same." is prompted.	room with room details should not be popped up.	be the same." is prompted.	
Find the Available Room with valid course, info, date, equipment requirement, start time, end time but Assessment Time is clashed with the enrolled user timetable	1. Enter valid course, info, date, equipment requirement, start time, end time 2. Click the "Find Room" button to find the available room for the specific duration of time	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 3:00PM End Time: 3:00PM	The alert message of "The assessment time is clashed with student timetable. Please find the available slot." is prompted.	The modal which displays the available room with room details should not be popped up.	The alert message of "The assessment time is clashed with student timetable. Please find the available slot." is prompted.	Pass

Find the Available Room with valid course, info, date, equipment requirement, start time, end time but Assessment Time is clashed with the other assessment time of same course assessment	1. Enter valid course, info, date, equipment requirement, start time, end time 2. Click the “Find Room” button to find the available room for the specific duration of time	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 3:00PM End Time: 3:00PM	The alert message of “The assessment time is clashed with other assessment time. Please find the available slot.” is prompted.	The modal which displays the available room with room details should not be popped up.	The alert message of “The assessment time is clashed with other assessment time. Please find the available slot.” is prompted.	Pass
Find the Available Room with valid course, info, date, equipment requirement, start time, end time	1. Enter valid course, info, date, equipment requirement, start time, end time 2. Click the “Find Room” button to find the available	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True	The alert message of “There are no room had enough equipment.” is prompted.	The modal which displays the available room with room	The alert message of “There are no room had enough equipment.” is prompted.	Pass

but there is no room have the require equipment	room for the specific duration of time	Sound System: True Video Conferencing: False Start Time: 3:00PM End Time: 3:00PM		details should not be popped up.		
Find the Available Room with valid course, info, date, equipment requirement, start time, end time but there is no room have the enough space to fit all the enrolled user	1. Enter valid course, info, date, equipment requirement, start time, end time 2. Click the “Find Room” button to find the available room for the specific duration of time	Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False Start Time: 3:00PM End Time: 3:00PM	The alert message of “There are no room had sufficient space for the assessment.” is prompted.	The modal which displays the available room with room details should not be popped up.	The alert message of “There are no room had sufficient space for the assessment.” is prompted.	Pass

Table 7.34: Unit Test Case for Find Slot and Room

Test Case ID	TC-034		**Module**		Find Slot and Room Module	
Test Title	Find the available Slot and Room during Creating or Updating Assessment Process					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin/ Lecturer has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Find the Available Slot and Room with valid course, info, date, equipment requirement and valid Time From, Time To, Duration Hour and Duration Minute	1. Enter valid course, info, date, equipment requirement 2. Click the "Advanced Flexible Find Slot and Room" button to pop up a modal to flexible find the available slot for finding available the room 3. Enter valid Time From, Time To, Duration Hour and Duration Minute 4. Click the "Find Slot and Room" button to find the	1. Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False 2. Time From: 08:00AM Time To: 6:00PM Duration Hour: 1	Pop Up a Modal which display all the available room with room details	-	Pop Up a Modal which display all the available room with room details	Pass

	available slots and rooms for the assessment	Duration Minute: 30				
Find the Available Slot and Room with empty course, info, date, equipment requirement	1. Enter empty course, info, date, equipment requirement 2. Click the "Advanced Flexible Find Slot and Room" button to pop up a modal to flexible find the available slot for finding available the room	1. Course: null Info: null Date: null Projector: null Whiteboard: null Computer: null Whiteboard: null Sound System: null Video Conferencing: null	The alert message of "Please fill in the date and course fields before find the slot." is prompted.	The modal which flexible find the available slot should not be popped up.	The alert message of "Please fill in the date and course fields before find the slot." is prompted.	Pass
Find the Available Slot and Room with valid course, info, date, equipment requirement and empty Time From, Time To, Duration Hour	1. Enter valid course, info, date, equipment requirement 2. Click the "Advanced Flexible Find Slot and Room" button to pop up a modal to flexible find the available slot for finding available the room	1. Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False	The alert message of "Please fill in all the fields before find the slot." is prompted.	The modal which displays the available slot and room should not be popped up.	The alert message of "Please fill in all the fields before find the slot." is prompted.	Pass

and Duration Minute	3. Enter empty Time From, Time To, Duration Hour and Duration Minute 4. Click the “Find Slot and Room” button to find the available slots and rooms for the assessment	2. Time From: null Time To: null Duration Hour: null Duration Minute: null				
Find the Available Slot and Room with valid course, info, date, equipment requirement and valid Time From, Time To, Duration Hour and Duration Minute but Time From is before 8:00AM which is the earliest	1. Enter valid course, info, date, equipment requirement 2. Click the “Advanced Flexible Find Slot and Room” button to pop up a modal to flexible find the available slot for finding available the room 3. Enter valid Time From, Time To, Duration Hour and Duration Minute 4. Click the “Find Slot and Room” button to find the	1. Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False 2. Time From: 05:00AM Time To: 6:00PM Duration Hour: 1 Duration Minute: 30	The alert message of “Invalid time from. Please enter a time between 08:00 and 22:00.” is prompted.	The modal which displays the available slot and room should not be popped up.	The alert message of “Invalid time from. Please enter a time between 08:00 and 22:00.” is prompted.	Pass

operating hour for educational institution	available slots and rooms for the assessment					
Find the Available Slot and Room with valid course, info, date, equipment requirement and valid Time From, Time To, Duration Hour and Duration Minute but Time To is before Time From	1. Enter valid course, info, date, equipment requirement 2. Click the “Advanced Flexible Find Slot and Room” button to pop up a modal to flexible find the available slot for finding available the room 3. Enter valid Time From, Time To, Duration Hour and Duration Minute 4. Click the “Find Slot and Room” button to find the available slots and rooms for the assessment	1. Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False 2. Time From: 09:00AM Time To: 08:00AM Duration Hour: 1 Duration Minute: 30	The alert message of “Invalid Time From and To. Time From cannot after Time To.” is prompted.	The modal which displays the available slot and room should not be popped up.	The alert message of “Invalid Time From and To. Time From cannot after Time To.” is prompted.	Pass
Find the Available Slot	1. Enter valid course, info, date, equipment requirement	1. Course: UECS2344 Info: MidTerm Test	The alert message of	The modal which	The alert message of	Pass

and Room with valid course, info, date, equipment requirement and valid Time From, Time To, Duration Hour and Duration Minute but Time To is before Time From	2. Click the "Advanced Flexible Find Slot and Room" button to pop up a modal to flexible find the available slot for finding available the room 3. Enter valid Time From, Time To, Duration Hour and Duration Minute 4. Click the "Find Slot and Room" button to find the available slots and rooms for the assessment	Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False 2. Time From: 09:00AM Time To: 08:00AM Duration Hour: 1 Duration Minute: 30	"Invalid Time From and To. Time From cannot after Time To." is prompted.	displays the available slot and room should not be popped up.	"Invalid Time From and To. Time From cannot after Time To." is prompted.	
Find the Available Slot and Room with valid course, info, date, equipment requirement and valid Time From, Time To but	1. Enter valid course, info, date, equipment requirement 2. Click the "Advanced Flexible Find Slot and Room" button to pop up a modal to flexible find the available slot for finding available the room	1. Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True	The alert message of "Invalid duration. Please enter hours between 0 and 23 while minutes among	The modal which displays the available slot and room should not	The alert message of "Invalid duration. Please enter hours between 0 and 23 while minutes among	Pass

invalid Duration Hour or Duration Minute	3. Enter valid Time From, Time To but invalid Duration Hour or Duration Minute 4. Click the "Find Slot and Room" button to find the available slots and rooms for the assessment	Video Conferencing: False 2. Time From: 09:00AM Time To: 12:00PM Duration Hour: 1 Duration Minute: -7	0, 15, 30 and 45 only." is prompted.	be popped up.	0, 15, 30 and 45 only." is prompted.	
Find the Available Slot and Room with valid course, info, date, equipment requirement and valid Time From, Time To, Duration Hour and Duration Minute but no searching slot generated due to	1. Enter valid course, info, date, equipment requirement 2. Click the "Advanced Flexible Find Slot and Room" button to pop up a modal to flexible find the available slot for finding available the room 3. Enter valid Time From, Time To, Duration Hour and Duration Minute 4. Click the "Find Slot and Room" button to find the	1. Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False 2. Time From: 09:00AM Time To: 10:00AM Duration Hour: 1	The alert message of "No slot with the duration within the time from and to." is prompted.	The modal which displays the available slot and room should not be popped up.	The alert message of "No slot with the duration within the time from and to." is prompted.	Pass

Duration of the assessment does not fit in the searching slot of Time From and To	available slots and rooms for the assessment	Duration Minute: 30				
Find the Available Slot and Room with valid course, info, date, equipment requirement and valid Time From, Time To, Duration Hour and Duration Minute but do not fulfil the constraints of student availability, space	1. Enter valid course, info, date, equipment requirement 2. Click the “Advanced Flexible Find Slot and Room” button to pop up a modal to flexible find the available slot for finding available the room 3. Enter valid Time From, Time To, Duration Hour and Duration Minute 4. Click the “Find Slot and Room” button to find the available slots and rooms for the assessment	1. Course: UECS2344 Info: MidTerm Test Date: 24/04/2024 Projector: True Whiteboard: True Computer: True Whiteboard: True Sound System: True Video Conferencing: False 2. Time From: 09:00AM Time To: 6:00PM Duration Hour: 1 Duration Minute: 30	The alert message of “No available rooms and slots for this assessment on the specific assessment date.” is prompted.	The modal which displays the available slot and room should not be popped up.	The alert message of “No available rooms and slots for this assessment on the specific assessment date.” is prompted.	Pass

constraint, time constraint or equipment constraint						

Table 7.35: Unit Test Case for Notify User

Test Case ID	TC-035		**Module**		Notification Module	
Test Title	Notify the user					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Admin/Lecturer has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Notify all the users about the assessment details with valid notification message input	1. Click the “Assessment” button in the navigation menu 2. Click the “Notify User” button of the specific assessment record to notify the users who enrolled in that specific assessment 3. Enter message and choose whether to attach assessment details in the notification	Message: The assessment is around the corner Attach assessment details: True	An email with the assessment notification is sent to all the users who enrolled the assessment. The success message of “Notifications sent successfully!” is flashed.	Admin or Lecturer should be redirected to View Assessment Page	An email with the assessment notification is sent to all the users who enrolled the assessment. The success message of “Notifications sent successfully!” is flashed.	Pass

Notify all the users about the assessment details with empty notification message input	1. Click the “Assessment” button in the navigation menu 2. Click the “Notify User” button of the specific assessment record to notify the users who enrolled in that specific assessment 3. Enter empty message and choose whether to attach assessment details in the notification	Message: null Attach assessment details: null	The error message of “Please fill out this field” is shown below the input field.	The notification should not be able to send and notify the users.	The error message of “Please fill out this field” is shown below the input field.	Pass
Notify all the users about the assessment details with valid notification message input but no user enrolled in the assessment	1. Click the “Assessment” button in the navigation menu 2. Click the “Notify User” button of the specific assessment record to notify the users who enrolled in that specific assessment	-	The error message of “No student enrols the course! Cannot Notify!” is flashed	-	The error message of “No student enrols the course! Cannot Notify!” is flashed	Pass

Table 7.36: Unit Test Case for View Chatroom

Test Case ID	TC-036		**Module**	Chat Module		
Test Title	View the chatroom					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View all the available chatroom	1. Click the “Chat” button in the navigation menu	-	All the chatrooms are displayed.	-	All the chatrooms are displayed.	Pass

Table 7.37: Unit Test Case for Real-Time/Online Chat

Test Case ID	TC-037		**Module**	Chat Module		
Test Title	Real-Time/Online Chat					
Plan Date	19 April 2024		**Planned By**	Loh Yong Bin		
Execution Date	22 April 2024		**Executed By**	Loh Yong Bin		
Pre-condition(s)	Admin/Lecturer has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
Real-time and Online Chat for Administrator	1. Click the "Enter Chat Room" button to enter the specific lecturer's chatroom to chat with the specific lecturer	-	Enter the chatroom successfully with the chatroom messages.	-	Enter the chatroom successfully with the chatroom messages.	Pass
Real-time and Online Chat for Lecturer	1. Click the "Chat" button in the navigation menu	-	Enter the chatroom successfully with the chatroom messages.	-	Enter the chatroom successfully with the chatroom messages.	Pass

Table 7.38: Unit Test Case for View Calendar

Test Case ID	TC-038		**Module**		Calendar Module	
Test Title	View the calendar					
Plan Date	19 April 2024		**Planned By**		Loh Yong Bin	
Execution Date	22 April 2024		**Executed By**		Loh Yong Bin	
Pre-condition(s)	Lecturer/Student has logged in the account.					
Test Case Description	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Pass/Fail**
View the assessment schedule in calendar view	1. Click the “Calendar” button in the navigation menu	-	All the assessment schedules are displayed in the view of calendar.	-	All the assessment schedules are displayed in the view of calendar.	Pass

7.3 Browser Compatibility Testing

In today's digital landscape, where users access web content through a variety of browsers and devices, browser compatibility testing emerges as a critical component of software testing particularly for web applications and websites. This type of testing ensures that the functionality, layout and performance of a web application remain consistent across different browsers and versions regardless of the underlying rendering engine and interpretation of web standards.

Browser compatibility testing plays a pivotal role in the testing strategy which serves several essential purposes. First and foremost, it aims to enhance the overall user experience by guaranteeing that our web application operates seamlessly across a diverse range of browsers which include popular options like Google Chrome, Mozilla Firefox and Microsoft Edge. By accommodating users who utilize different browsers, Classroom Finder System aims to maximize the reach and accessibility of the application and potentially expand the user base and market presence.

In the testing approach, the comprehensive Browser Compatibility Testing was executed to validate the functionality and performance of the web application across various browsers. Through diligent testing and attention to detail, this project aims to deliver a robust and reliable web application that meets the diverse needs and expectations of the user base.

Table 7.39: Browser Compatibility Testing Result

Browser / **Test Cases**	**Chrome**	**Firefox**	**Edge**
<u>Contents</u>			
Images, fonts display properly	Pass	Pass	Pass
Layout consistency (button, modal, etc.)	Pass	Pass	Pass
Layout responsiveness	Pass	Pass	Pass

Functionalities (Main Features)			
Login and Logout	Pass	Pass	Pass
Manage Lecturer	Pass	Pass	Pass
Manage Student	Pass	Pass	Pass
Manage Room	Pass	Pass	Pass
Manage Course	Pass	Pass	Pass
Manage Course Timetable	Pass	Pass	Pass
Manage Course Enrolment	Pass	Pass	Pass
Manage Assessment	Pass	Pass	Pass
View Assessment User	Pass	Pass	Pass
Find Room	Pass	Pass	Pass
Advance Flexible Find Slot and Room	Pass	Pass	Pass
Notify User through Email	Pass	Pass	Pass
Real-Time Communication/Chat	Pass	Pass	Pass
View Calendar	Pass	Pass	Pass

7.4 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) is a critical phase in the software testing process which focuses on validating whether a system meets the requirements and expectations of its end users. Unlike other types of testing which primarily assess technical aspects of the software, UAT evaluates the software's usability, functionality and overall user experience from the perspective of the intended audience. It involves real end users or stakeholders interacting with the system in a simulated or real-world environment to verify that it meets their needs and is fit for purpose before deployment.

In this project, total of 8 users have actively participated to perform the user acceptance testing. The users are divided into three groups: 2 users are actively participated in the Administrative UAT, 3 users are actively participated in the Lecturer UAT, and 2 users are actively participated in the Student UAT. The users are to test all the modules of Classroom Finder System. Below are the templates which used for the user acceptance test for Administrator, Lecturer and Student while the results of the UAT are displayed in the appendix. In this user acceptance testing, the test cases are having “Pass” status when the user pass it while having “Fail” when the user fail the test case.

7.4.1 User Acceptance Testing (UAT) for Student

Table 7.40: UAT Test Cases for Student

Test Module	Test Case ID	Test Description	Status	Comment
Sign Up	SUAD-001	1. Able to sign up a new account		
Login and Logout	SUAD-002	1. Able to login the registered account		
	SUAD-003	2. Able to logout the account		
Assessment	SUAD-004	1. Able to view all the assessment enrolled		
	SUAD-005	2. Able to view all the past assessment enrolled		
	SUAD-006	3. Able to view all the active assessment enrolled		
Calendar	SUAD-007	1. Able to view the assessment schedule in the view of calendar		

7.4.2 User Acceptance Testing (UAT) for Lecturer

Table 7.41: UAT Test Cases for Lecturer

Test Module	Test Case ID	Test Description	Status	Comment
Login and Logout	LUAD-001	1. Able to login the lecturer account		
	LUAD-002	2. Able to logout the lecturer account		
Assessment	LUAD-003	1. Able to view all the assessment enrolled		
	LUAD-004	2. Able to view all the past assessment enrolled		
	LUAD-005	3. Able to view all the active assessment enrolled		
	LUAD-006	4. Able to add new assessment		
	LUAD-007	5. Able to update the assessment		
	LUAD-008	6. Able to delete the assessment		
	LUAD-009	7. Able to view all the assessment user		
Find Slot and Room	LUAD-010	1. Able to find the available room		
	LUAD-011	2. Able to find the available slot and room		
Notification	LUAD-012	1. Able to notify all the user of the assessment		

Calendar	LUAD-013	1. Able to view the assessment schedule in the view of calendar		
Chat	LUAD-014	1. Able to real-time and online chat with Administrator		

7.4.3 User Acceptance Testing (UAT) for Administrator

Table 7.42: UAT Test Cases for Administrator

Test Module	**Test Case ID**	**Test Description**	**Status**	**Comment**
Login and Logout	AUAD-001	1. Able to login the lecturer account		
	AUAD-002	2. Able to logout the lecturer account		
Lecturer	AUAD-003	1. Able to view the lecturer list		
	AUAD-004	2. Able to add new lecturer		
	AUAD-005	3. Able to update the lecturer		
	AUAD-006	4. Able to delete the lecturer		
Student	AUAD-007	1. Able to view the student list		
	AUAD-008	2. Able to add new student		
	AUAD-009	3. Able to update the student		
	AUAD-010	4. Able to delete the student		
Room	AUAD-011	1. Able to view the room list		
	AUAD-012	2. Able to add new room		
	AUAD-013	3. Able to update the room		
	AUAD-014	4. Able to delete the room		

Course	AUAD-015	1. Able to view the course list		
	AUAD-016	2. Able to add new course		
	AUAD-017	3. Able to update the course		
	AUAD-018	4. Able to delete the course		
Timetable	AUAD-019	1. Able to view the course timetable list		
	AUAD-020	2. Able to add new course timetable when session is ‘Lecture’		
	AUAD-021	3. Able to add new course timetable when session is not ‘Lecture’		
	AUAD-022	3. Able to update the course timetable when session is ‘Lecture’		
	AUAD-023	4. Able to update the course timetable when session is not ‘Lecture’		
	AUAD-024	5. Able to delete the course timetable		
Enrolment	AUAD-025	1. Able to view the course enrolment list for student		
	AUAD-026	2. Able to view the course enrolment list for lecturer		
	AUAD-027	3. Able to add new course enrolment		

		when session is 'Lecture'		
	AUAD-028	4. Able to add new course enrolment when session is not 'Lecture'		
	AUAD-029	5. Able to update the course enrolment when session is 'Lecture'		
	AUAD-030	6. Able to update the course enrolment when session is not 'Lecture'		
	AUAD-031	7. Able to delete the course enrolment		
Assessment	AUAD-032	1. Able to view all the assessment enrolled		
	AUAD-033	2. Able to view all the past assessment enrolled		
	AUAD-034	3. Able to view all the active assessment enrolled		
	AUAD-035	4. Able to add new assessment		
	AUAD-036	5. Able to update the assessment		
	AUAD-037	6. Able to delete the assessment		
	AUAD-038	7. Able to view all the assessment user		

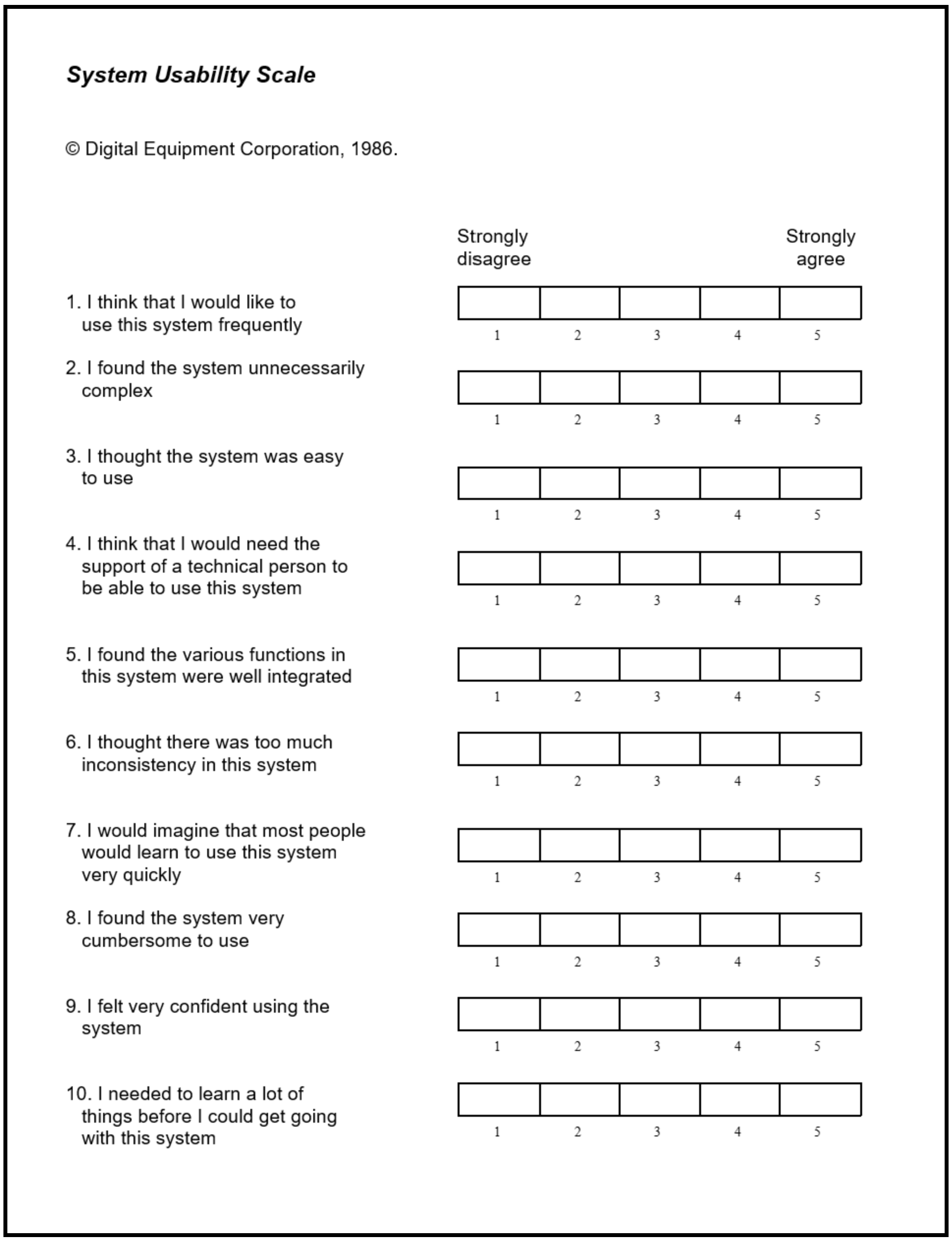
Find Slot and Room	AUAD-039	1. Able to find the available room		
	AUAD-040	1. Able to find the available slot and room		
Notification	AUAD-041	1. Able to notify all the user of the assessment		
Chat	AUAD-042	1. Able to view all the chatroom		
	AUAD-043	2. Able to real-time and online chat with lecturer		

7.5 System Usability Testing

System Usability Testing (SUT) is a crucial phase in the software development lifecycle aimed at evaluating the user-friendliness and effectiveness of the developed system. In this phase, the focus shifts to gathering feedback from actual end-users to assess their experience interacting with the software interface and functionalities. The primary objective of SUT is to identify the usability issues, gauge user satisfaction and validate the extent to which the system meets user needs and expectations.

In this project, the System Usability Scale (SUS), a widely recognized and standardized questionnaire developed by John Brooke in 1986 is employed. The SUS questionnaire consists of a series of statements designed to measure the user's perception of system usability, efficiency and overall user experience. The testers are asked to rate their level of agreement with each statement on a Likert scale which ranges from "Strongly Disagree" to "Strongly Agree." A total of 8 testers participated in the System Usability Testing phase which represents the diverse user demographics and roles. The SUS questionnaire is distributed electronically to testers via WhatsApp which allows for convenient and efficient data collection while the testers are instructed to complete the questionnaire after interacting with the system for a specified duration.

In conclusion, the System Usability Testing phase provides the valuable insights into the user experience and usability of the developed system. By incorporating the user feedback and addressing the usability issues, this project aims to enhance the overall usability and user satisfaction of the system. The findings from the SUT phase will inform future iterations of the software development process to ensure that the system meets the needs and expectations of its intended users.



System Usability Scale

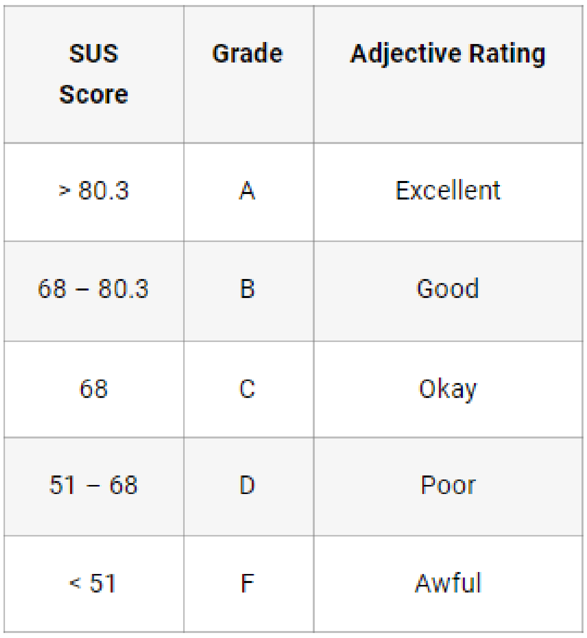
© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure 7.1: SUS Questionnaire

Once the SUT is conducted, it generates a quantifiable measure called the SUS score which ranges from 0 to 100 and offers information about overall customer satisfaction and usability. A higher score indicates improved usability, whereas a lower score indicates possible usability issues.

In order to calculate the SUS score, the process entails converting user responses from the SUS questionnaire into numerical values and then conducting computations based on these values. Initially, each response option in the SUS survey corresponds to a specific point value which facilitates a numerical representation of users' subjective feedback. For instance, strongly disagree equates to 1 point, disagree to 2 points, neutral to 3 points, agree to 4 points and strongly agree to 5 points. Subsequently, the SUS survey's 10 statements are segregated into odd-numbered and even-numbered questions for calculation purposes. The odd-numbered questions' points are subtracted by 1 and total points are subtracted by 5 to yield the 'x' value while the even-numbered questions' points subtracted by 5 and total points subtracted by 25 provide the 'y' value. The final SUS score is computed using the formula: SUS Score = (x + y) * 2.5. This computation integrates both 'x' and 'y' values to generate the SUS score which serves as an indicator of the system's perceived usability.



SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Figure 7.2: SUS Grading Table

According to the SUS Grading depicted in Figure 7.2, the SUS Score is categorized into different grades to assess the usability of the system. A score above 80.3 is classified as Grade A, indicating excellence in usability. Scores ranging from 68 to 80.3 are designated as Grade B, representing good usability. Grade C encompasses scores of 68, signifying an acceptable level of usability. Scores falling between 51 and 68 are categorized as Grade D, reflecting poor usability. Lastly, scores lower than 51 are labelled as Grade F, indicating an extremely poor level of usability. This grading system offers a comprehensive framework for evaluating the usability of the system based on standardized thresholds.

Table 7.43: SUS Questionnaire Result

Tester	**Questions**										**Total**	**Score**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**		
1	5	1	5	1	5	1	5	1	5	1	40	100
2	5	1	5	1	5	1	5	1	5	1	40	100
3	5	1	5	1	5	1	5	1	5	1	40	100
4	4	1	4	1	5	1	4	1	4	1	36	90
5	4	1	3	1	5	2	5	1	5	1	36	90
6	4	1	4	1	5	1	4	1	4	1	36	90
7	5	1	4	1	5	1	5	1	5	1	39	90
8	4	2	4	1	5	1	4	1	5	1	36	90
Average SUS Score												93.75

The presented table provides a condensed overview of the SUS questionnaire results obtained from individual users while the detailed findings are available in the Appendix. The average SUS score for the system is reported as 93.75 which is indicating a highly positive perception of the system's usability among users. This exceptional score underscores users' favourable impressions of the system's ease of use and effectiveness. Therefore, with a SUS score of 93.75, the system's usability is deemed excellent according to the SUS grading criteria outlined in the SUS grading table in figure 7.2.

7.6 Requirement Traceability Matrix

In this section, a comprehensive overview of the requirement traceability matrix is provided to establish the connections between various components such as use cases, unit test cases, user acceptance tests and the functional requirements outlined in the preceding chapter. This matrix serves as a crucial tool to ensure that all functional requirements are adequately addressed and validated through the corresponding use cases and test cases, thus facilitating thorough testing and validation of the system's functionality and adherence to user requirements.

7.6.1 Use Case Table

Table below shows all the use cases within Classroom Finder System with use case ID and use case Name.

Table 7.44: Use Case Table

Use Case ID	**Use Case Name**
UC001	Register Account
UC002	Login Account
UC003	Logout Account
UC004	Manage Student
UC005	Manage Lecturer
UC006	Manage Room
UC007	Manage Course
UC008	Manage Course Timetable
UC009	Manage Course Enrolment
UC010	View Assessment
UC011	Create or Update Assessment
UC012	Delete Assessment
UC013	View Assessment User
UC014	Notify Assessment User
UC015	Find Room
UC016	Advance Flexible Find Slot and Room
UC017	Online Chat/Communication
UC018	View Calendar

7.6.2 Functional Requirement Table

Table below shows all the functional requirements within Classroom Finder System with functional requirements ID and functional requirements Statement.

Table 7.45: Functional Requirement Table

Functional Requirement ID	Functional Requirement Statement
FR001	The system should allow administrator to login and logout the account.
FR002	The system should allow administrator to manage (create, read, update, delete) the student.
FR003	The system should allow administrator to manage (create, read, update, delete) the lecturer.
FR004	The system should allow administrator to manage (create, read, update, delete) the room.
FR005	The system should allow administrator to manage (create, read, update, delete) the course.
FR006	The system should allow administrator to manage (create, read, update, delete) the course timetable.
FR007	The system should allow administrator to manage (create, read, update, delete) the course enrolment.
FR008	The system should allow administrator to manage (create, read, update, delete) the assessment.
FR009	The system should allow administrator to advance flexible find the slot and room for the assessment creation.
FR010	The system should allow administrator to find the room for the assessment creation.
FR011	The system should allow administrator to view the student who enrols in the assessment.
FR012	The system should allow administrator to notify the student who enrols in the assessment.

FR013	The system should allow administrator online chat or communication with lecturer.
FR014	The system should allow lecturer to login and logout the account.
FR015	The system should allow lecturer to manage (create, read, update, delete) the assessment.
FR016	The system should allow lecturer to advance flexible find the slot and room for the assessment creation.
FR017	The system should allow lecturer to find the room for the assessment creation.
FR018	The system should allow lecturer to view the student who enrols in the assessment.
FR019	The system should allow lecturer to notify the student who enrols in the assessment.
FR020	The system should allow lecturer to view the calendar.
FR021	The system should allow lecturer online chat or communication with administrator.
FR022	The system should allow student to register the account.
FR023	The system should allow student to login and logout the account.
FR024	The system should allow student to view the assessment.
FR025	The system should allow student to view the calendar.

7.6.3 Traceability Matrix

The following table presents the traceability matrix which is offering a clear depiction of the interconnections among unit test cases, user acceptance test cases, functional requirements and use cases documented in the preceding chapter. This matrix serves as a visual aid which illustrate the relationships and dependencies between different elements of the system's development process to facilitate a comprehensive understanding of how each component contributes to the fulfilment of overall project objectives.

Table 7.46: Traceability Matrix

Use Case ID	**Functional Requirement ID**	**Unit Test Case ID**	**UAT Test Case ID**
UC001	FR022	TC-001	SUAD-001
UC002	FR001, FR014, FR023	TC-002	SUAD-002, LUAD-001, AUAD-001
UC003	FR001, FR014, FR023	TC-003	SUAD-003, LUAD-002, AUAD-002
UC004	FR002	TC-008, TC-009, TC-010, TC-011	AUAD-007, AUAD-008, AUAD-009, AUAD-010
UC005	FR003	TC-004, TC-005, TC-006, TC-007	AUAD-003, AUAD-004, AUAD-005, AUAD-006
UC006	FR004	TC-012, TC-013, TC-014, TC-015	AUAD-011, AUAD-012, AUAD-013, AUAD-014
UC007	FR005	TC-016, TC-017, TC-018, TC-019	AUAD-015, AUAD-016,

			AUAD-017, AUAD-018
UC008	FR006	TC-020, TC-021, TC-022, TC-023	AUAD-019, AUAD-020, AUAD-021, AUAD-022, AUAD-023, AUAD-024
UC009	FR007	TC-024, TC-025, TC-026, TC-027	AUAD-025, AUAD-026, AUAD-027, AUAD-028, AUAD-029, AUAD-030, AUAD-031
UC010	FR008, FR015, FR024	TC-028	SUAD-004, SUAD-005, SUAD-006, LUAD-003, LUAD-004, LUAD-005, AUAD-032, AUAD-033, AUAD-034
UC011	FR008, FR015	TC-029, TC-030	LUAD-006, LUAD-007, AUAD-035, AUAD-036
UC012	FR008, FR015	TC-031	LUAD-008, AUAD-037
UC013	FR011, FR018	TC-032	LUAD-009, AUAD-038
UC014	FR012, FR019	TC-035	LUAD-012, AUAD-041
UC015	FR010, FR017	TC-033	LUAD-010, AUAD-039

UC016	FR009, FR016	TC-034	LUAD-011, AUAD-040
UC017	FR013, FR021	TC-036, TC-037	LUAD-014, AUAD-042, AUAD-043
UC018	FR020, FR025	TC-038	SUAD-007, LUAD-013

CHAPTER 8

CONCLUSION AND RECOMMENDATIONS

8.1 Introduction

In the realm of educational institutions, the effective allocation of rooms for student assessments stands as a cornerstone for fostering an environment conducive to learning and academic growth. Traditionally, this process has been fraught with challenges which often characterized by manual efforts, time-consuming tasks and a high potential for errors. However, recognizing the need for innovation and efficiency, this project embarked on the development of a comprehensive solution which is the Classroom Finder System.

Through meticulous planning, analysis and implementation, this project has culminated in the creation of a robust web-based application designed to streamline and optimize the room finding and allocation process for student assessments. Building upon a foundation of understanding the complexities and limitations of manual methods, this project sought to harness the power of technology to revolutionize room allocation in educational institutions.

This introduction sets the stage for a detailed exploration of the conclusions drawn from the development of the Classroom Finder System. By examining the objectives achieved, the challenges addressed, and the solutions implemented, this project aims to provide valuable insights into the potential benefits of adopting an automated approach to room allocation. Through a comprehensive analysis, this report aims to elucidate the impact of the Classroom Finder System and pave the way for future developments in optimizing educational resource management.

8.2 Objective Achievement

The objectives outlined at the inception of this project have been diligently pursued and achieved which marked the significant strides towards enhancing the efficiency and effectiveness of room allocation processes in educational institutions. Firstly, the objective to identify specific constraints and requirements pertinent to the Classroom Finder System has been successfully met. Through thorough analysis of the system, research on the problem statement and proposed solution in Chapter 1 and the literature review on existing problem, solution and system, the key constraints such as student availability, space requirements, time limitations and equipment requirements were identified and incorporated into the system's design. This comprehensive understanding laid the groundwork for developing a solution tailored to address the unique challenges faced in room allocation.

Secondly, the exploration of suitable Artificial Intelligence approaches and room allocation algorithms has yielded fruitful outcomes. By leveraging AI techniques such as the Constraint Satisfaction Problem (CSP) and the Backtracking algorithm, the system intelligently navigates the complexities of room allocation while considering multiple constraints simultaneously. The implementation of these algorithms empowers the system to generate optimal room assignments that align with various requirements while minimizing conflicts and maximizing resource utilization.

Thirdly, the objective to implement real-time management, communication, updates and notification has been effectively realized. Through seamless integration of real-time data updates and notification mechanisms, the system ensures that users are promptly informed of any changes or updates related to assessment schedules or room allocations through email. This real-time functionality enhances transparency, reduces the likelihood of scheduling conflicts and fosters improved collaboration among administrators, lecturers, and students.

Finally, the development of the automate Classroom Finder System represents the culmination of these objectives into a tangible and impactful solution. The system's user-friendly interface which coupled with its robust backend algorithms empowers the educational institutions to streamline the room allocation process, minimize administrative burden and optimize resource utilization. By automating previously manual tasks and leveraging advanced AI technologies, the Classroom Finder System sets a new standard for efficiency and effectiveness in educational resource management.

In conclusion, the achievement of these objectives underscores the transformative potential of the Classroom Finder System in revolutionizing room allocation processes. Through a holistic approach encompassing system design, algorithmic implementation and real-time functionality, this project has laid the groundwork for enhancing the academic experience and fostering a conducive learning environment for students and educators alike.

8.3 Limitation and Recommendation

While the Classroom Finder System represents a significant advancement in addressing the challenges of room allocation within educational institutions, several limitations and areas for improvement have been identified throughout the development process.

Firstly, the limitation lies in the system's inadequate support for importing and exporting various types of educational data beyond assessment records such as course details, course timetables, and course enrolment information. Presently, the system may lack efficient mechanisms to import comprehensive datasets encompassing diverse educational aspects, hindering its ability to seamlessly integrate with external educational management systems or share data with other stakeholders. This limitation restricts users' flexibility in migrating comprehensive educational data into or out of the Classroom Finder System. To overcome this limitation, the system should enhance its importing and exporting capabilities to encompass a broader range of educational data which includes course details, timetables, and enrolment records. Implementing standardized data exchange formats which similar to CSV or XML (eXtensible Markup Language) can facilitate the seamless transfer of diverse educational data between the Classroom Finder System and external sources. Additionally, developing user-friendly interfaces that allow administrators to map data fields between systems during import/export processes can enhance the system's adaptability to varying data structures. Furthermore, integrating with industry-standard protocols like OAuth (Open Authorization) for secure data sharing and API (Application Programming Interface) endpoints for programmatic access to educational data can enhance interoperability with external systems.

Next, the limitation could be the lack of comprehensive reporting and analytics features within the system. While the Classroom Finder System effectively manages room allocations, assessments and user interactions, it may lack robust reporting tools to analyse data trends, assess system performance and derive actionable insights for educational administrators. This limitation restricts the system's ability to provide stakeholders with valuable insights into room utilization, assessment scheduling efficiency and user engagement metrics and hinder the informed decision-making and strategic planning processes within educational institutions. To address this limitation, the system should integrate advanced reporting and analytics features that empower administrators, lecturers, and students to visualize and analyse educational data effectively. The implementation of customizable dashboards with interactive data visualizations such as charts, graphs and heatmaps can enable stakeholders to monitor key performance indicators and trends in room utilization, assessment schedules and user engagement metrics in real-time. Additionally, incorporating data mining and machine learning algorithms can facilitate predictive analytics which allows the system to forecast future trends, identify potential issues and recommend optimization strategies proactively. Furthermore, providing export functionalities for generated reports in standard formats like PDF or Excel can enable stakeholders to share insights easily with other decision-makers and stakeholders.

Last but not least, the limitation of system's reliance on accurate data inputs poses a significant challenge. The inaccuracies or inconsistencies in student schedules, room capacities or assessment requirements can lead to suboptimal room allocation decisions and operational inefficiencies. This limitation could potentially undermine the overall effectiveness of the Classroom Finder System and impact its ability to streamline the room allocation process and facilitate conflict-free assessments. While to address this limitation, the implementation of robust data validation mechanisms is essential. By developing validation protocols that ensure the integrity of data inputs which include real-time verification of student schedules and room availability, the system can minimize the risk of inaccuracies. The regular data update procedures should also be established to maintain the accuracy of information stored in the system. Additionally, integrating data quality control measures such as automated checks and error detection algorithms can further enhance the reliability of data inputs and mitigate the impact of this limitation on room allocation decisions.

REFERENCES

Abramson, D. & Abela, J., 1991. A PARALLEL GENETIC ALGORITHM FOR SOLVING THE SCHOOL TIMETABLING PROBLEM. *Division of Information Technology,* pp. 1-11.

Brailsford, S. C., Potts, C. N. & Smith, B. M., 1999. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research,* 119(3), pp. 557-581.

Brittan, J. N. G. & Farley, F. J. M., 1971. College timetable construction by computer. *The Computer Journal,* 14(4), pp. 361-365.

Burke, E., Elliman, D. & Weare, R., 1994. A Genetic Algorithm Based University Timetabling System. *In Proceedings of the 2nd east-west international conference on computer technologies in education,* Volume 1, pp. 35-40.

Carter, M. W., Laporte, G. & Chinneck, J. W., 1994. A General Examination Scheduling System. *Interfaces,* 24(3), pp. 109-120.

Costa, D., 1994. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research,* 76(1), pp. 98-110.

Dechter, R., 2003. *Constraint Processing.* s.l.:Morgan Kaufmann.

Deris, S., Ohta, H., Omatu, S. & Samat, . P., 1997. University timetabling by constraint-based reasoning:. *Journal of the Operational Research Society,* pp. 1178-1190.

Despa, M. L., 2014. Comparative study on software development methodologies. *Database Systems Journal,* 5(3).

El-Sakka, T., 2015. University Course Timetable using Constraint Satisfaction and Optimization. *International Journal of Computing Academic Research,* 4(3), pp. 83-95.

Fuentealba, C., 2011. The Role of Assessment in the Student Learning Process. *Journal of veterinary medical education,* 38(2), pp. 157-162.

Geske, U., 1998. *Use of Declarative Programming for Solving Industrial Optimization Problems.* [Online] Available at: https://www.ercim.eu/publication/Ercim_News/enw32/geske.html
[Accessed 28 8 2023].

H., 2020. *6 critical phases of the Systems Development Life Cycle.* [Online] Available at: https://unitanzania.com/the-systems-development-life-cycle-sdlc/ [Accessed 13 8 2023].

Haralick, R. M. & Elliott, G. L., 1980. Increasing tree search efficiency for constraint. *Artificial Intelligence,* Volume 14, pp. 263-313.

Hertz, A., 1992. Finding a feasible course schedule using Tabu search. *Discrete Applied Mathematics,* 35(3), pp. 255-270.

Jaffar, J. & Maher, M. J., 1994. Constraint logic programming: a survey. *The Journal of Logic Programming,* Volume 19, pp. 503-581.

Kumar, V., 1992. Algorithms for Constraint-Satisfaction Problems: A Survey. *AI magazine,* 13(1), pp. 32-44.

Laoyan, S., 2022. *What is Agile methodology? (A beginner's guide).* [Online] Available at: https://asana.com/resources/agile-methodology [Accessed 13 8 2023].

Li, R., Mallaria, C. B. & San Juan, J. L., n.d. The University Coursework Timetabling Problem: An Optimization Approach to Synchronizing Course Calendars.

Luenendonk, M., 2020. *7 Basic Software Development Life Cycle (SDLC) Methodologies: Which One is Best?.* [Online] Available at: https://www.cleverism.com/software-development-life-cycle-sdlc-methodologies/
[Accessed 13 8 2023].

Mallari, C. B. C., San Juan, J. L. G. & Li, R. C., 2023. The university coursework timetabling problem: An optimization approach to synchronizing course calendars. *Computers & Industrial Engineering.*

Murray, K., Muller, T. & Rudova, H., 2007. Modeling and Solution of a Complex University. *Practice and Theory of Automated Timetabling VI,* p. 189–209.

Norman, M. G., Paechter, B., Cumming, A. & Luchian, H., 1996. Extensions to a Memetic Timetabling System. *Practice and Theory of Automated Timetabling,* p. 251–265.

Petrovic, S. & Burke, E., 2004. University Timetabling.

Rane, M. V. et al., 2021. Automated Timetabling System for University Course. *International Conference on Emerging Smart Computing and Informatics (ESCI).*

Tsang, E., 2014. *Foundations of Constraint Satisfaction: The Classic Text.* s.l.:BoD–Books on Demand.

Yepuri, V. K., Pamu, G. C., Kodali, N. & V, P. L., 2018. Examination Management Automation System. *International Research Journal of Engineering and Technology (IRJET),* 5(4).

Zhang, L., 2005. Solving the timetabling problem using constraint satisfaction programming.

Zhang, L. & Lau, S. K., 2005. Constructing university timetable using constraint satisfaction programming approach. *International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA 2005), International Conference on Intelligent Agents, Web Technologies and Internet Commerce,* Volume 2, pp. 55-60.

APPENDICES

Appendix A: User Acceptance Testing Results

Student 1:

Name: Goh Ren Xiang

Test Execution Date: 15/04/2024

Test Module	Test Case ID	Test Description	Status	Comment
Sign Up	SUAD-001	1. Able to sign up a new account	Pass	
Login and Logout	SUAD-002	1. Able to login the registered account	Pass	
	SUAD-003	2. Able to logout the account	Pass	
Assessment	SUAD-004	1. Able to view all the assessment enrolled	Pass	
	SUAD-005	2. Able to view all the past assessment enrolled	Pass	
	SUAD-006	3. Able to view all the active assessment enrolled	Pass	
Calendar	SUAD-007	1. Able to view the assessment schedule in the view of calendar	Pass	Can display in 30 days calendar view

Student 2:

Name: Eng Yong Han

Test Execution Date: 15/04/2024

Test Module	Test Case ID	Test Description	Status	Comment
Sign Up	SUAD-001	1. Able to sign up a new account	Pass	Can implement verification of email address
Login and Logout	SUAD-002	1. Able to login the registered account	Pass	
	SUAD-003	2. Able to logout the account	Pass	
Assessment	SUAD-004	1. Able to view all the assessment enrolled	Pass	
	SUAD-005	2. Able to view all the past assessment enrolled	Pass	
	SUAD-006	3. Able to view all the active assessment enrolled	Pass	
Calendar	SUAD-007	1. Able to view the assessment schedule in the view of calendar	Pass	

Student 3:

Name: Ong Han Lun

Test Execution Date: 15/04/2024

Test Module	Test Case ID	Test Description	Status	Comment
Sign Up	SUAD-001	1. Able to sign up a new account	Pass	
Login and Logout	SUAD-002	1. Able to login the registered account	Pass	
	SUAD-003	2. Able to logout the account	Pass	
Assessment	SUAD-004	1. Able to view all the assessment enrolled	Pass	
	SUAD-005	2. Able to view all the past assessment enrolled	Pass	
	SUAD-006	3. Able to view all the active assessment enrolled	Pass	
Calendar	SUAD-007	1. Able to view the assessment schedule in the view of calendar	Pass	

<u>Lecturer 1:</u>

Name: Loh Chia Hui

Test Execution Date: 18/04/2024

Test Module	Test Case ID	Test Description	Status	Comment
Login and Logout	LUAD-001	1. Able to login the lecturer account	Pass	
	LUAD-002	2. Able to logout the lecturer account	Pass	
Assessment	LUAD-003	1. Able to view all the assessment enrolled	Pass	
	LUAD-004	2. Able to view all the past assessment enrolled	Pass	
	LUAD-005	3. Able to view all the active assessment enrolled	Pass	
	LUAD-006	4. Able to add new assessment	Pass	
	LUAD-007	5. Able to update the assessment	Pass	
	LUAD-008	6. Able to delete the assessment	Pass	
	LUAD-009	7. Able to view all the assessment user	Pass	
Find Slot and Room	LUAD-010	1. Able to find the available room	Pass	
	LUAD-011	2. Able to find the available slot and room	Pass	
Notification	LUAD-012	1. Able to notify all the user of the assessment	Pass	

Calendar	LUAD-013	1. Able to view the assessment schedule in the view of calendar	Pass	
Chat	LUAD-014	1. Able to real-time and online chat with Administrator	Pass	

Lecturer 2:

Name: Grace Tok

Test Execution Date: 18/04/2024

Test Module	Test Case ID	Test Description	Status	Comment
Login and Logout	LUAD-001	1. Able to login the lecturer account	Pass	
	LUAD-002	2. Able to logout the lecturer account	Pass	
Assessment	LUAD-003	1. Able to view all the assessment enrolled	Pass	
	LUAD-004	2. Able to view all the past assessment enrolled	Pass	
	LUAD-005	3. Able to view all the active assessment enrolled	Pass	
	LUAD-006	4. Able to add new assessment	Pass	
	LUAD-007	5. Able to update the assessment	Pass	
	LUAD-008	6. Able to delete the assessment	Pass	
	LUAD-009	7. Able to view all the assessment user	Pass	
Find Slot and Room	LUAD-010	1. Able to find the available room	Pass	The time input field is not user-friendly and effective
	LUAD-011	2. Able to find the available slot and room	Pass	The time input field is not user-

				friendly and effective
Notification	LUAD-012	1. Able to notify all the user of the assessment	Pass	
Calendar	LUAD-013	1. Able to view the assessment schedule in the view of calendar	Pass	
Chat	LUAD-014	1. Able to real-time and online chat with Administrator	Pass	

<u>Lecturer 3:</u>

Name: Kelvin Tan

Test Execution Date: 18/04/2024

Test Module	**Test Case ID**	**Test Description**	**Status**	**Comment**
Login and Logout	LUAD-001	1. Able to login the lecturer account	Pass	
	LUAD-002	2. Able to logout the lecturer account	Pass	
Assessment	LUAD-003	1. Able to view all the assessment enrolled	Pass	
	LUAD-004	2. Able to view all the past assessment enrolled	Pass	
	LUAD-005	3. Able to view all the active assessment enrolled	Pass	
	LUAD-006	4. Able to add new assessment	Pass	
	LUAD-007	5. Able to update the assessment	Pass	
	LUAD-008	6. Able to delete the assessment	Pass	
	LUAD-009	7. Able to view all the assessment user	Pass	
Find Slot and Room	LUAD-010	1. Able to find the available room	Pass	
	LUAD-011	2. Able to find the available slot and room	Pass	
Notification	LUAD-012	1. Able to notify all the user of the assessment	Pass	Can provide more flexible message

				customization and designing for emailing
Calendar	LUAD-013	1. Able to view the assessment schedule in the view of calendar	Pass	
Chat	LUAD-014	1. Able to real-time and online chat with Administrator	Pass	

<u>Administrator 1:</u>

Name: Poh Kim Lee

Test Execution Date: 19/04/2024

Test Module	Test Case ID	Test Description	Status	Comment
Login and Logout	AUAD-001	1. Able to login the lecturer account	Pass	
	AUAD-002	2. Able to logout the lecturer account	Pass	
Lecturer	AUAD-003	1. Able to view the lecturer list	Pass	
	AUAD-004	2. Able to add new lecturer	Pass	
	AUAD-005	3. Able to update the lecturer	Pass	
	AUAD-006	4. Able to delete the lecturer	Pass	
Student	AUAD-007	1. Able to view the student list	Pass	
	AUAD-008	2. Able to add new student	Pass	
	AUAD-009	3. Able to update the student	Pass	
	AUAD-010	4. Able to delete the student	Pass	
Room	AUAD-011	1. Able to view the room list	Pass	
	AUAD-012	2. Able to add new room	Pass	
	AUAD-013	3. Able to update the room	Pass	
	AUAD-014	4. Able to delete the room	Pass	

Course	AUAD-015	1. Able to view the course list	Pass	
	AUAD-016	2. Able to add new course	Pass	
	AUAD-017	3. Able to update the course	Pass	
	AUAD-018	4. Able to delete the course	Pass	
Timetable	AUAD-019	1. Able to view the course timetable list	Pass	
	AUAD-020	2. Able to add new course timetable when session is ‘Lecture’	Pass	
	AUAD-021	3. Able to add new course timetable when session is not ‘Lecture’	Pass	
	AUAD-022	3. Able to update the course timetable when session is ‘Lecture’	Pass	
	AUAD-023	4. Able to update the course timetable when session is not ‘Lecture’	Pass	
	AUAD-024	5. Able to delete the course timetable	Pass	
Enrolment	AUAD-025	1. Able to view the course enrolment list for student	Pass	
	AUAD-026	2. Able to view the course enrolment list for lecturer	Pass	
	AUAD-027	3. Able to add new course enrolment	Pass	

		when session is 'Lecture'		
	AUAD-028	4. Able to add new course enrolment when session is not 'Lecture'	Pass	
	AUAD-029	5. Able to update the course enrolment when session is 'Lecture'	Pass	
	AUAD-030	6. Able to update the course enrolment when session is not 'Lecture'	Pass	
	AUAD-031	7. Able to delete the course enrolment	Pass	
Assessment	AUAD-032	1. Able to view all the assessment enrolled	Pass	
	AUAD-033	2. Able to view all the past assessment enrolled	Pass	
	AUAD-034	3. Able to view all the active assessment enrolled	Pass	
	AUAD-035	4. Able to add new assessment	Pass	
	AUAD-036	5. Able to update the assessment	Pass	
	AUAD-037	6. Able to delete the assessment	Pass	
	AUAD-038	7. Able to view all the assessment user	Pass	

Find Slot and Room	AUAD-039	1. Able to find the available room	Pass	
	AUAD-040	1. Able to find the available slot and room	Pass	
Notification	AUAD-041	1. Able to notify all the user of the assessment	Pass	
Chat	AUAD-042	1. Able to view all the chatroom	Pass	
	AUAD-043	2. Able to real-time and online chat with lecturer	Pass	

<u>Administrator 2:</u>

Name: Tan Siew Yan

Test Execution Date: 19/04/2024

Test Module	Test Case ID	Test Description	Status	Comment
Login and Logout	AUAD-001	1. Able to login the lecturer account	Pass	
	AUAD-002	2. Able to logout the lecturer account	Pass	
Lecturer	AUAD-003	1. Able to view the lecturer list	Pass	
	AUAD-004	2. Able to add new lecturer	Pass	
	AUAD-005	3. Able to update the lecturer	Pass	
	AUAD-006	4. Able to delete the lecturer	Pass	
Student	AUAD-007	1. Able to view the student list	Pass	
	AUAD-008	2. Able to add new student	Pass	
	AUAD-009	3. Able to update the student	Pass	
	AUAD-010	4. Able to delete the student	Pass	
Room	AUAD-011	1. Able to view the room list	Pass	
	AUAD-012	2. Able to add new room	Pass	
	AUAD-013	3. Able to update the room	Pass	
	AUAD-014	4. Able to delete the room	Pass	

Course	AUAD-015	1. Able to view the course list	Pass	
	AUAD-016	2. Able to add new course	Pass	
	AUAD-017	3. Able to update the course	Pass	
	AUAD-018	4. Able to delete the course	Pass	
Timetable	AUAD-019	1. Able to view the course timetable list	Pass	
	AUAD-020	2. Able to add new course timetable when session is ‘Lecture’	Pass	
	AUAD-021	3. Able to add new course timetable when session is not ‘Lecture’	Pass	
	AUAD-022	3. Able to update the course timetable when session is ‘Lecture’	Pass	
	AUAD-023	4. Able to update the course timetable when session is not ‘Lecture’	Pass	
	AUAD-024	5. Able to delete the course timetable	Pass	
Enrolment	AUAD-025	1. Able to view the course enrolment list for student	Pass	
	AUAD-026	2. Able to view the course enrolment list for lecturer	Pass	
	AUAD-027	3. Able to add new course enrolment	Pass	

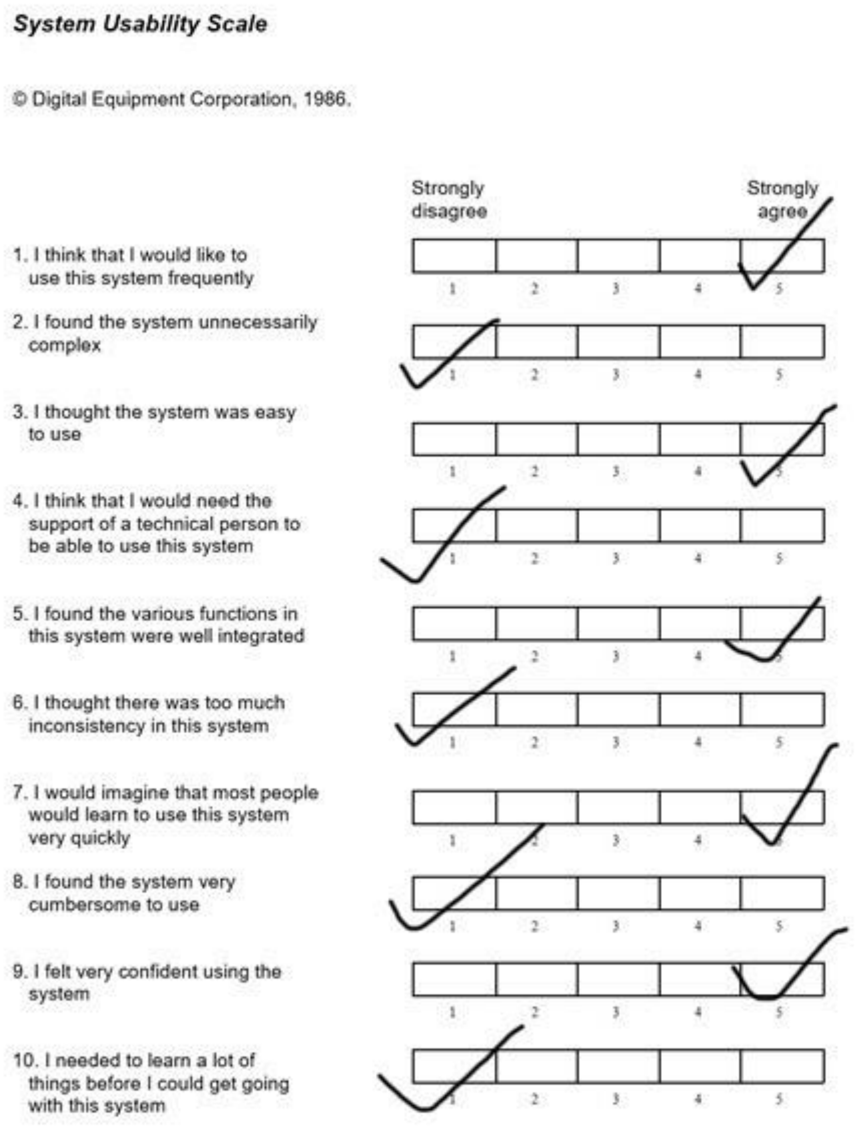
		when session is 'Lecture'		
	AUAD-028	4. Able to add new course enrolment when session is not 'Lecture'	Pass	
	AUAD-029	5. Able to update the course enrolment when session is 'Lecture'	Pass	
	AUAD-030	6. Able to update the course enrolment when session is not 'Lecture'	Pass	
	AUAD-031	7. Able to delete the course enrolment	Pass	
Assessment	AUAD-032	1. Able to view all the assessment enrolled	Pass	
	AUAD-033	2. Able to view all the past assessment enrolled	Pass	
	AUAD-034	3. Able to view all the active assessment enrolled	Pass	
	AUAD-035	4. Able to add new assessment	Pass	
	AUAD-036	5. Able to update the assessment	Pass	
	AUAD-037	6. Able to delete the assessment	Pass	
	AUAD-038	7. Able to view all the assessment user	Pass	

Find Slot and Room	AUAD-039	1. Able to find the available room	Pass	
	AUAD-040	1. Able to find the available slot and room	Pass	
Notification	AUAD-041	1. Able to notify all the user of the assessment	Pass	
Chat	AUAD-042	1. Able to view all the chatroom	Pass	
	AUAD-043	2. Able to real-time and online chat with lecturer	Pass	

Appendix B: SUS Questionnaires

Student 1:

Name: Goh Ren Xiang



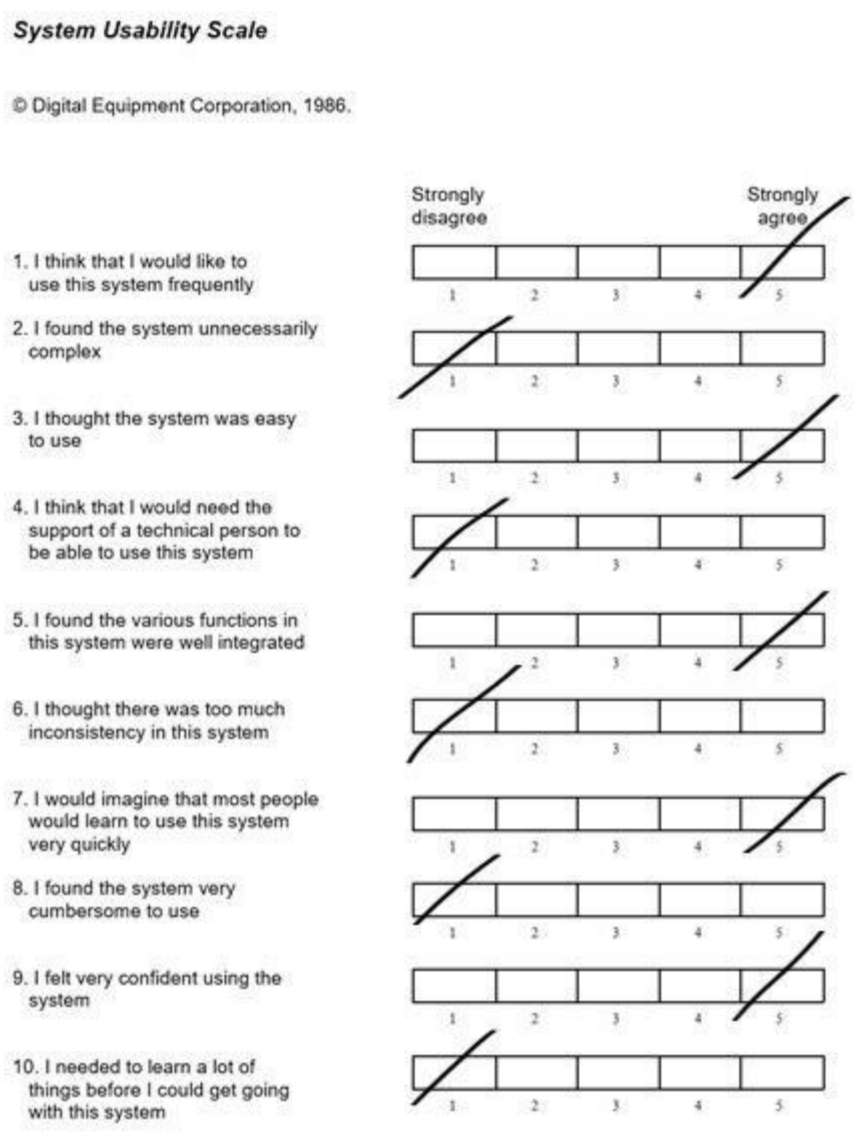
System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree 1	2	3	4	Strongly agree 5
1. I think that I would like to use this system frequently					✓
2. I found the system unnecessarily complex	✓				
3. I thought the system was easy to use					✓
4. I think that I would need the support of a technical person to be able to use this system	✓				
5. I found the various functions in this system were well integrated					✓
6. I thought there was too much inconsistency in this system	✓				
7. I would imagine that most people would learn to use this system very quickly					✓
8. I found the system very cumbersome to use	✓				
9. I felt very confident using the system					✓
10. I needed to learn a lot of things before I could get going with this system	✓				

<u>Student 2:</u>

Name: Eng Yong Han



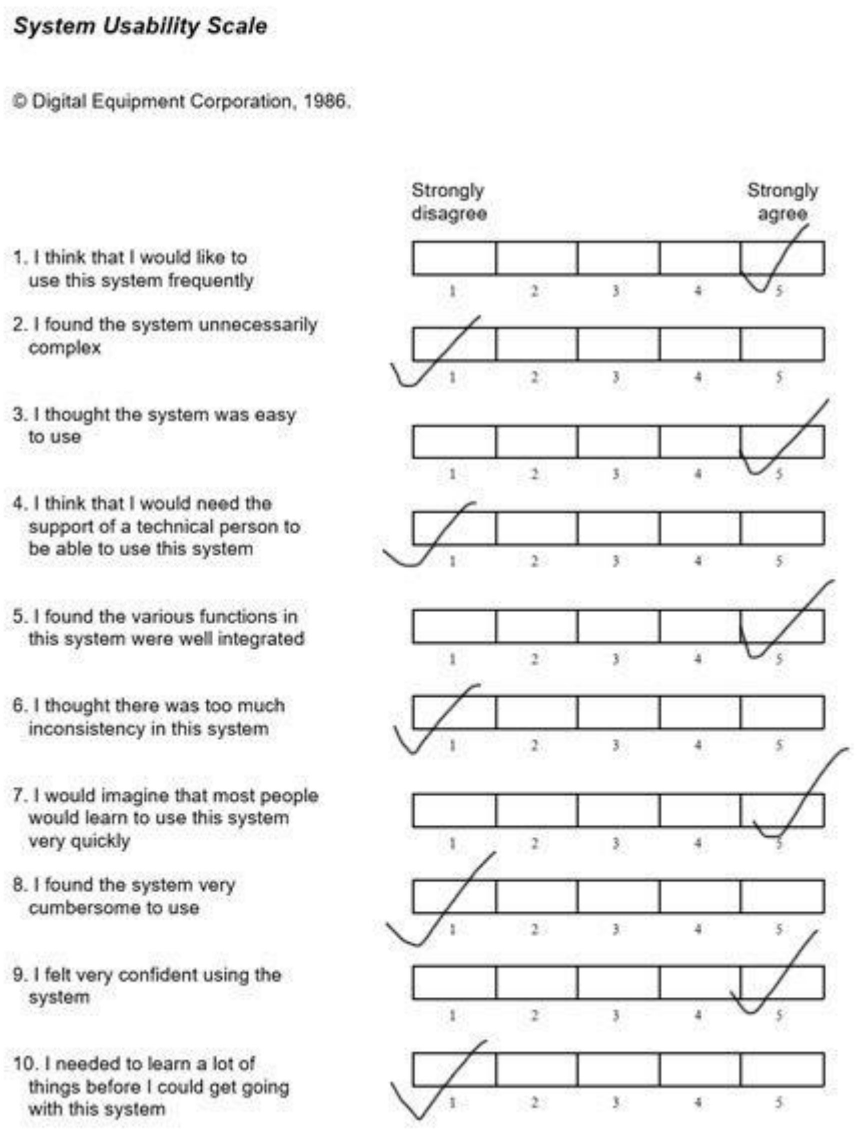
System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree 1	2	3	4	Strongly agree 5
1. I think that I would like to use this system frequently					✓
2. I found the system unnecessarily complex	✓				
3. I thought the system was easy to use					✓
4. I think that I would need the support of a technical person to be able to use this system	✓				
5. I found the various functions in this system were well integrated					✓
6. I thought there was too much inconsistency in this system	✓				
7. I would imagine that most people would learn to use this system very quickly					✓
8. I found the system very cumbersome to use	✓				
9. I felt very confident using the system					✓
10. I needed to learn a lot of things before I could get going with this system	✓				

Student 3:

Name: Ong Han Lun



System Usability Scale

	Strongly disagree				Strongly agree
	1	2	3	4	5
1. I think that I would like to use this system frequently					✓
2. I found the system unnecessarily complex	✓				
3. I thought the system was easy to use					✓
4. I think that I would need the support of a technical person to be able to use this system	✓				
5. I found the various functions in this system were well integrated					✓
6. I thought there was too much inconsistency in this system	✓				
7. I would imagine that most people would learn to use this system very quickly					✓
8. I found the system very cumbersome to use	✓				
9. I felt very confident using the system					✓
10. I needed to learn a lot of things before I could get going with this system	✓				

<u>Lecturer 1:</u>

Name: Loh Chia Hui

System Usability Scale

	Strongly disagree				Strongly agree
	1	2	3	4	5
1. I think that I would like to use this system frequently				✓	
2. I found the system unnecessarily complex	✓				
3. I thought the system was easy to use				✓	
4. I think that I would need the support of a technical person to be able to use this system	✓				
5. I found the various functions in this system were well integrated					✓
6. I thought there was too much inconsistency in this system	✓				
7. I would imagine that most people would learn to use this system very quickly				✓	
8. I found the system very cumbersome to use	✓				
9. I felt very confident using the system				✓	
10. I needed to learn a lot of things before I could get going with this system	✓				

Lecturer 2:

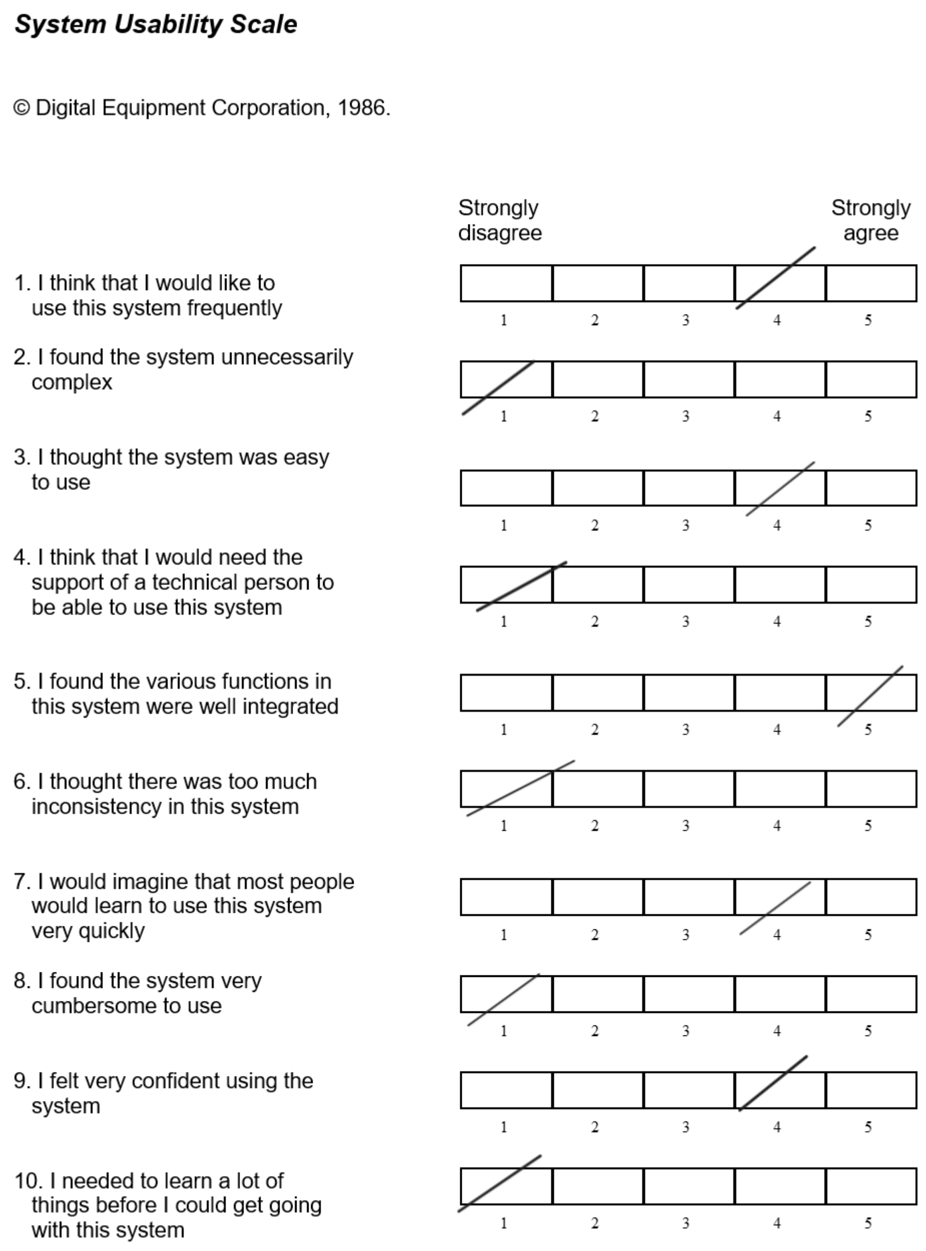
Name: Grace Tok

System Usability Scale

	Strongly disagree				Strongly agree
	1	2	3	4	5
1. I think that I would like to use this system frequently				✓	
2. I found the system unnecessarily complex	✓				
3. I thought the system was easy to use			✓		
4. I think that I would need the support of a technical person to be able to use this system	✓				
5. I found the various functions in this system were well integrated					✓
6. I thought there was too much inconsistency in this system		✓			
7. I would imagine that most people would learn to use this system very quickly					✓
8. I found the system very cumbersome to use	✓				
9. I felt very confident using the system					✓
10. I needed to learn a lot of things before I could get going with this system	✓				

<u>Lecturer 3:</u>

Name: Kelvin Tan



System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	[✓ 4]	5
2. I found the system unnecessarily complex	[✓ 1]	2	3	4	5
3. I thought the system was easy to use	1	2	3	[✓ 4]	5
4. I think that I would need the support of a technical person to be able to use this system	[✓ 1]	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	[✓ 5]
6. I thought there was too much inconsistency in this system	[✓ 1]	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	[✓ 4]	5
8. I found the system very cumbersome to use	[✓ 1]	2	3	4	5
9. I felt very confident using the system	1	2	3	[✓ 4]	5
10. I needed to learn a lot of things before I could get going with this system	[✓ 1]	2	3	4	5

<u>Administrator 1:</u>

Name: Poh Kim Lee

System Usability Scale

	Strongly disagree				Strongly agree
	1	2	3	4	5
1. I think that I would like to use this system frequently					✓
2. I found the system unnecessarily complex	✓				
3. I thought the system was easy to use				✓	
4. I think that I would need the support of a technical person to be able to use this system	✓				
5. I found the various functions in this system were well integrated					✓
6. I thought there was too much inconsistency in this system	✓				
7. I would imagine that most people would learn to use this system very quickly					✓
8. I found the system very cumbersome to use	✓				
9. I felt very confident using the system					✓
10. I needed to learn a lot of things before I could get going with this system	✓				

<u>Administrator 2:</u>

Name: Tan Siew Yan

System Usability Scale

	Strongly disagree				Strongly agree
	1	2	3	4	5
1. I think that I would like to use this system frequently				✓	
2. I found the system unnecessarily complex		✓			
3. I thought the system was easy to use				✓	
4. I think that I would need the support of a technical person to be able to use this system	✓				
5. I found the various functions in this system were well integrated					✓
6. I thought there was too much inconsistency in this system	✓				
7. I would imagine that most people would learn to use this system very quickly				✓	
8. I found the system very cumbersome to use	✓				
9. I felt very confident using the system					✓
10. I needed to learn a lot of things before I could get going with this system	✓				