UNIVERSITI TUNKU ABDUL RAHMAN (UTAR) VENUE/CLASSROOM BOOKING WEB PORTAL

WONG JIA KEEN

UNIVERSITI TUNKU ABDUL RAHMAN

Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal

WONG JIA KEEN

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

October 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	:	JayKay
Name	:	Wong Jia Keen
ID No.	:	22UEB00528
Date	:	30 September 2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal"** was prepared by **WONG JIA KEEN** 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	:	Beh Hooi Ching, Michelle
Date	:	30 September 2024

ACKNOWLEDGEMENTS

Progressing a project is a remarkable journey. I am grateful that I had encountered people that had inspired me to work harder and overcame many hardships and having a number of remarkable moments.

First and foremost, I would like to express my heartfelt gratitude to my project supervisor, Ms. Beh Hooi Ching for her effort, guidance and fruitful advice that had led me step by step to complete this project. Next, I would like to thank Mr. Chiah Kah Hin and my colleagues in Ideal Vision Integration Sdn Bhd. whom I had collaborated with during my internship. They had provided me insights and lessons on how to build a system from scratch, in terms of requirements gathering and technical aspects.

On the other hand, I would like to extend my sincere gratitude to those UTAR parties who had been willing to assist me to progress on this project. Without their participation on the requirements gathering and testing phases, this project would not be completed smoothly. Additionally, I would like to pay tribute to my family and my friends for supporting me from the start of this project. With their moral support and encouragement, they keep me motivated, pushing me to the limits to deliver the best of this project.

ABSTRACT

It is common that a university hosts numerous venues or classrooms to cater the academic needs, but it could be rare for a university to implement an efficient management system that handles the venue usage requests. This happened in Universiti Tunku Abdul Rahman, which the affected parties range from the management, lecturers to the students. Grabbing this opportunity, an upgraded system which aims to solve this context is introduced and enables the end-users to book venues/classrooms according to their usage and to view the venues/classrooms details if they are unfamiliar with. When it comes to booking request updates, it will notify the involved parties via email and SMS, keeping them alert in any circumstances. The end-users could view their booking request in PDF report format so that it could be used as a clarification tool. The system is also equipped with an instant messaging function, so that the booking requesters could clarify with the management side if they had any doubts on the usage. Other than catering the venue usage requests, it assists the management party to allocate class sessions with the venue/classroom. Accessing authority and data encrypted measures are featured in this system to keep the data as confidential as possible. Adopting the phased development methodology, this project had undergone the necessary phases from the requirements elicitation, development, testing and closure in the scheduled time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
DDL	Data Manipulation Language
DFD	Data Flow Diagram
DGS	Department of General Services
DSA	Department of Student Affairs
HOD	Head of Department
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
KIS	Keep it Simple
PHP	Hypertext Preprocessor
RAD	Rapid Application Development
SDLC	Software Development Life Cycle
SQL	Structured Query Language
SUS	System Usability Scale
URL	Uniform Resource Locator
UTAR	Universiti Tunku Abdul Rahman
WBS	Work Breakdown Structure
XHR	XMLHttpRequest
XP	Extreme Programming

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Located in Sungai Long, the UTAR campus is formed by the KA Block and the KB Block buildings (UTAR, n.d.). Aside from managerial offices, both buildings consist of lecture theatres, tutorial rooms and lab rooms to accommodate the student count (UTAR, 2024). On rare occasions, these venues mentioned are open to the public for running their events (UTAR, n.d.). Other than using the venues for classes and the outsiders' usage, the venues are open for student events as well, like holding regular practice or meetings based on their co-curricular activities. As the university runs a hectic schedule and these venues are limited, currently there is a dedicated yet computed system for the administrative staff to manage the venue/classroom booking requests (Foo, 2024).

However, it was observed that the system is only accessible by UTAR administrative staff and lecturers. Though the system had existed since a long time ago, there have been no upgrades so far and the functionalities supported are considered outdated, in terms of both administrative functions and booking functions (Foo, 2024). Therefore, this project is determined to refine the system's requirements along with the working logic, and to upgrade the entire system technically. Retaining the platform as web-based, the upgraded system aims to let the UTAR community book the venues with ease. Booking a venue should not be a challenging task and the necessary information in the system should be revealed so that fewer misunderstandings could arise and ensure the transparency of the booking mechanism.

Mentioning this project is development-based, it is expected that a transformed Venue/Classroom Booking Web Portal could be delivered as planned at the end of the project.

1.2 Importance of the Project

This subsection describes the essentialness of coming out with an idea of the development of the UTAR Venue/Classroom Booking Web Portal.

The current venue booking system is only available to UTAR lecturers, while the administrative functions are handled by the staff from the Department of General Services (DGS) mainly and partly from the Department of Student Affairs (DSA) (Foo, 2024). As time goes by, the functionalities are considered limited nowadays and could not cater to users' needs anymore, which in turn received lots of criticism from the system users (Foo, 2024). According to the interview with the DGS staff, the time slot arrangement for the venue bookings on the current mechanism is not properly coordinated between departments, which increases their workload to often clarify with the booking requesters (Foo, 2024). As UTAR students are one of the booking requesters, it is targeted that UTAR students could be recognized as part of the end-user in this project.

Therefore, a thorough case study is needed to clarify the actual booking procedures which aid the development logic, obtain the users' points-of-view to examine the unsatisfactory parts of the existing system, and analyze their expectations on an upgraded system. The results collected are crucial as they provide a clear direction for the software development, like what features the involved parties expect and other non-functional details. In short, this project's outcome is considered as an upgrade to the existing UTAR venue booking system and to be released for real usage.

1.3 Problem Statements

This subsection describes the issues faced by the UTAR community which urges the development of the UTAR Venue/Classroom Booking Web Portal.

1.3.1 Current booking process lacks transparency

Should a comprehensive venue booking system show all vital information rather than hiding it for no reason. According to the interviews conducted with one of the students, there is no detailed information that states why the booking requests are being rejected (Ng, 2024). Furthermore, there is no communication between the administrator and the booking requester, unless the booking requester does so (Soo, 2024). With this to be said, the rejected booking requester is completely unaware of the actual reasons why the booking request was disapproved. This may cause the booking requester to get confused and lose confidence in using the venue booking system if such an issue persists. This loophole could be rectified by amending the existing system's logic workflows and presenting it through form inputs.

Referring to Appendix A, it is suggested that the rejection process requires the administrator to input the reason why the booking request is being rejected. This process is feasible, by guiding the administrator to fill the rejection reason once the administrator chooses to reject the pending booking request. As a result, the booking requester could read the rejected reason other than being updated on the status only. In case the booking requester is not satisfied with the approval results, he/she may choose to contact the administrator by establishing a chat session with the administrator. These solutions solve the problem of information transparency and communication problems, since all relevant information is disclosed.

1.3.2 Limited yet outdated functionalities offered

In terms of functionalities, a proper venue booking system for a university should be comprehensive, not only serving the booking purpose but also for the communication aspect. Nevertheless, the existing system features lack comprehensiveness and could not fulfil the current end-users' requirements anymore. According to the interview with the DGS staff (Foo, 2024), this causes most of the information to be neglected unintentionally and has to be observed manually, which directly contributes to the happening of booking requester conflicts due to human error that overlooks pieces of information. Throughout many years of release, there have been no major upgrades to the venue booking mechanism (Foo, 2024). This situation is considered a failure of the existing system as the project team failed to address these possible situations at the project initiation phase and could not cater for further enhancement. Additionally, the project author holds opinions that the current venue booking system could be upgraded with the modern technological trends, like the booking requester to chat with the DGS staff or HOD if needed and to get updated with each venue's availability which these features are lacking in the current system. Such features could diversify the functionality of the venue booking system and level up the end-users' satisfaction rate as it requires less manual work.

According to the requirements gathering result, most of the interviewees wanted a graphical layout function that shows the mappings for all venues/classrooms in UTAR, along with the availability status. They eagerly wanted this function as they had to blindly state their preferred venue/classroom without knowing the status and the availability date for the current system. Such a problem is the most common issue faced by all interviewees; therefore this function will be prioritized.

1.3.3 Poor usage experience upon using the current venue booking system

A system which has a large group of users should be designed with state-of-the-art elements and the navigating logics should be logically organized so that the end-users could experience fewer errors in decision-making and most importantly increase productivity with lesser cognitive load. Nonetheless, the existing venue booking system's layout is inelegant in terms of design, as the website still uses old-fashioned elements like non-guiding forms and non-informative navigation links. Moreover, dark green tone serves as the main colour scheme for the current system, which could make the users find it boring to use if they had to use it for a longer duration. This could be proven by the rating of some lecturers who find the design boring, according to the survey (2024). With the result collected, it is determined that the system must be redesigned and ensure it fulfils the Human Interaction Computer Design fundamentals.

The redesigned venue booking system interface strives to achieve a set of reputable design principles (Ben, 1986). The first principle states that "Strive for consistency", this could be achieved as there will be a standardized style of design to be applied across the entire system, including the font type chosen, form design, labelling and diagram sizes (Ben, 1986). Another achievable principle states that “Offer informative feedback”, where it emphasizes that the end-user must be clear about the actions performed and the current stage they are at all times (Ben, 1986). Stating that “Design dialogue to yield closure”, so that the end-user would not need to guess the next action and are aware of the information as well (Ben, 1986). This could be achieved by designing interactive pop-ups and ensuring the messages convey the system status and actions.

Another notable rule states that “Support internal locus of control” where users could feel the sense of controlling the system by their own, therefore the system redesigned should provide options for configuring the method of displaying the information (Ben, 1986). Using the current UTAR Venue/Classroom Booking System as an example, the administrators could choose how many rows of information to be shown in the dynamic Data Table. Last but not least, the last principle declares “Reduce short-term memory load”. This principle reminds the design elements and the information loaded to be as simplistic as possible, to get the users to be more attentive to the tasks (Ben, 1986). Short yet precise hints could be used in the layouts for this case.

1.3.4 Limited group of users currently

Stating that the boundary of the system is limited within the controlled environment, then all parties in the environment should have the right to access the knowledge of the system. However, according to the interview and survey conducted, the current UTAR venue booking mechanism is only open to the DSA, DGS administrative staff and lecturers. The students possess little or no knowledge of the workflows and could cause miscommunication if there are any mishaps. Venue booking details should not be something confidential that the details need to be hidden from the students just because of their role. In fact, some students who are representatives of the co-curricular societies need to reserve a venue through email, which leads to incomplete information being conveyed at times and the response duration could be unstable. Stating that the student’s needs must be entertained, but the administration still needs to play their role in reviewing the booking reasons. It is planned that there will be an input box that the administration needs to fill if rejected, to ensure the fairness of the reviewing process.

1.4 Aim and Objectives

This subsection states the goals to be achieved by the project. With the problems clearly identified in the subsection above, the system to be developed is determined to bring good to the UTAR community by reducing venue/classroom booking inconveniences.

1. To conduct a requirements and satisfaction study on the current venue/classroom booking system.
2. To identify the areas for improvement on the current booking system and compile solutions to solve the problem statements within project completion.
3. To develop an independent yet upgraded venue/classroom booking system for UTAR context.
4. To achieve a basic satisfaction score of 80% through conducting System Usability Testing.

1.5 Proposed Solution

This subsection suggests the solution that is appropriate to solve the project's problem.

To overcome the drawbacks, a web-based system dedicated to UTAR venue/classroom booking will be developed. Stating the platform running is on the web, end-users could access it through laptops, desktop computers or any mobile devices, provided they are connected to the Internet. Differentiating the end-users into groups like administrators, lecturers and students, these groups of users could access the system to manage the venue booking matters. Different groups of end-users will be authorized with certain functionalities so that the management roles and boundaries are clearly defined.

In terms of technical structure, the web system will be built using PHP which utilizes HTML elements as the layout and CSS with Bootstrap 2 to define the styling of the elements displayed. With the involvement of Bootstrap, the system could be usable on mobile devices as well. Mentioning it runs on the web, the system will be hosted on XAMPP server with the database PhpMyAdmin supported on it. The database requires the knowledge of MYSQL and it is sufficient for storing, retrieving and amending the booking information by the end-users.

It is planned that this system would perform emailing when it comes to changing of booking information which could trigger an update to the parties involved, therefore external libraries like PHPMailer would be used for this case. As some students mentioned that they would not check their email inbox frequently, therefore it was advised to make use of the Instant Messaging application to perform the notifying action. At times end-users may not get connected to the Internet, therefore SMS is the most recommended approach to get them updated on the latest venue booking changes. This could be achieved by using the SMS API library by Bulk360. Another mentionable library is the Data Table, which is used to display the information fetched from the database in a managed yet user-defined way. The following figure illustrates the relationships between the components mentioned above:

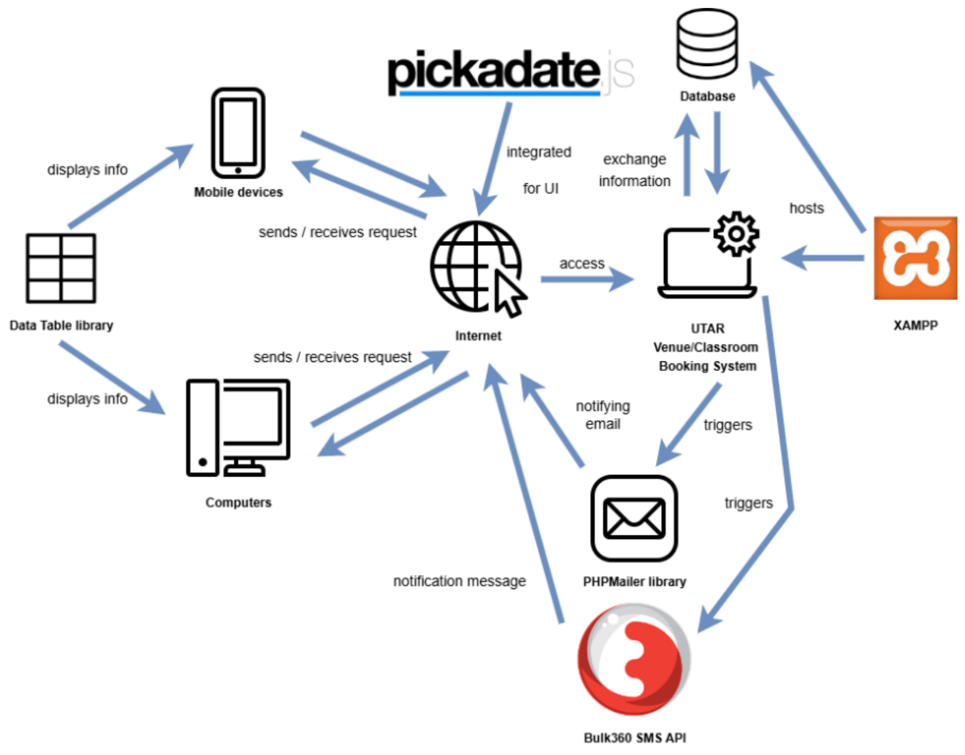


Figure 1.1: Overview of the Proposed Solution

1.6 Project Approach

A software development project does not start simply without any preliminary research, as it determines the project scope and further decides the tasks to be done. Therefore, this subsection will describe the research and development approach to be used.

1.6.1 Research Approach

The research is a hybrid of qualitative and quantitative methods. Quantitative research aims to study predictable yet regular viewpoints in the form of rating scales (reference) (Sukamolson, 2007). This research is suitable for measuring the satisfaction level of the end-users on the various aspects of the current UTAR venue/classroom booking system, including user interactiveness and satisfaction with the functionalities supported. However, the results collected are limited in terms of comprehensiveness as there is no descriptive information supported with their satisfaction level selected (Sukamolson, 2007).

Therefore, quantitative research methods would be utilised too. In contrast, the qualitative research method defines that the human thought and behavioural data could be personal and contextual (Sukamolson, 2007). Meaning to say, qualitative research results could be more descriptive and help the project author in understanding the underlying phenomena better.

Stating that mixed research methods are used in this project study, qualitative research will be conducted in an online questionnaire survey, while quantitative research will be conducted in interviews. Based on the responses and interview content, the end-users' expectations will be presented in analytical representations like graphs to show the data scientifically. The opinions collected would be formalized and considered in the development stage, presented as the Use Case Descriptions.

1.6.2 Development Approach

Phased Development Approach is chosen as the software development approach in this project. Categorized under Rapid Application Development which aims to develop the application faster than the traditional Waterfall Model and

Parallel Model, this approach segments the system's functionalities to be delivered into versions to integrate (Dennis, Barbara and David, 2009). In traditional models, it is hard to revert to previous cycles to conduct code changes, but Phased Development Approach solves this problem by enabling another code change to be conducted after a version release. The following figure illustrates how this approach works.

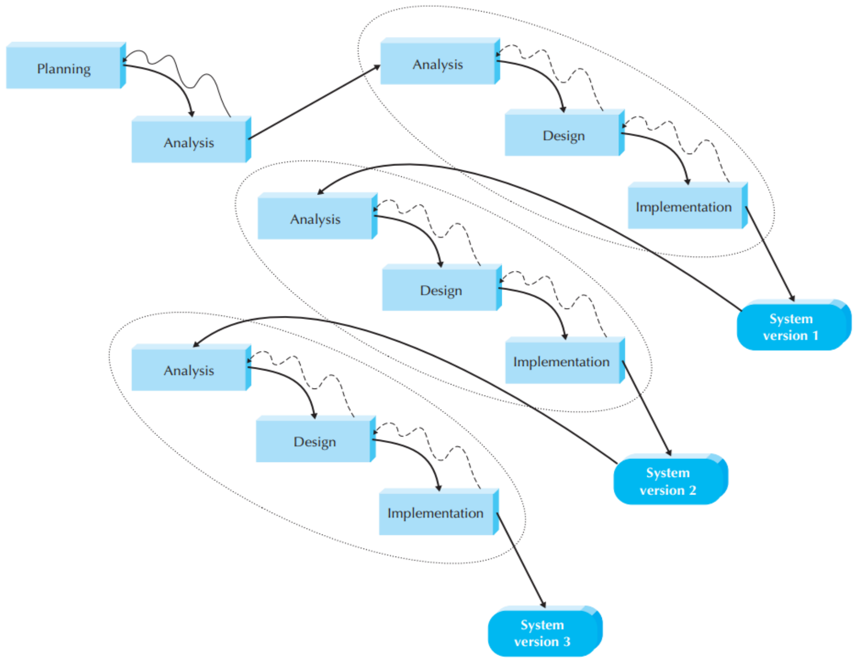


Figure 1.2: Overview of the Phased Development Approach (Dennis, Barbara and David, 2009)

As the UTAR venue/classroom booking system comprises many features in which flexible revert is most needed for fixing unexpected system errors, Phased Development is the optimum approach to fit this project.

1.7 Scope and Limitation of the Project

This subsection describes the boundary of this project, to ensure the details gathered are controlled and meaningful to the initiation phase of the UTAR Venue/Classroom Booking Web System development. However, there are still restrictions which will be discussed in this subsection as well.

1.7.1 Target Users

This part describes the potential end-users of the UTAR Venue/Classroom Booking System.

1.7.1.1 DGS Venue Allocation staff and HOD

This group of users are responsible for daily handling the booking requests. In detail, they are the parties to set the policy on booking the venues and coordinate the venue availability. For example, they need to inspect incoming booking requests for approval or rejection, and to communicate with the booking requester. If necessary, they would need to prepare the facilities that are requested.

1.7.1.2 UTAR lecturers

Stating that UTAR is a university, the venues/classrooms are used often for running classes or tutorial sessions. Due to special occasions where replacement or additional classes are needed, the lecturers need to go through a series of procedures with the administrative staff to book the venue/classroom. According to Appendix B, the lecturers need to seek approval from the Head of Department (HOD) first before submitting the request to the DGS staff. Therefore, UTAR lecturers are recognized in this case and split into two categories of users: HOD and normal-level teaching staff.

1.7.1.3 UTAR students

UTAR students often request the venues after class hours, due to their usage for running co-curricular activities or preparations for presentations or studies. While the demand from this group of users is large, they are a part of the end-users for this system to be developed. However due to the probability of misuse, this group of users must be supervised by enforcing a stricter validation upon their request, which they need to seek approval from their lecturer first before submitting the request to the DGS administrative staff to confirm the booking. Administrative staff will need to confirm the presence of the lecturer's consent on their request to proceed.

1.7.2 System Scope

The UTAR venue/classroom booking web portal is a web-based system which is open to UTAR community, utilizing web development and database technologies to build it. Being hosted using a web server solution, it can be accessed using desktop computers or mobile devices, with the device connected to the Internet. In terms of development, the system will be built using Visual Studio Code along with draw.io to sketch the requirements drawing. Axure RP 9 will be used for prototyping as well.

1.7.3 Project Modules

The system is composed of a total of four modules. Starting from the login module which aims to authenticate the registered users, the users will be directed to the dashboard module after being authenticated. Different modules indicate different functions which will be discussed.

1.7.3.1 Login/Logout Module

The system users (regardless of administrative and non-administrative) will need to input their UTAR email address and password to verify the identity of processing the requests. Descriptive messages will be displayed if the login fails, telling the user what should be noticed. At the same time, a logout module will be included to clear the using session of the current user, securing the users' information. A resetting password submodule is included as well, so that the end-users have the authority to handle their passwords by their own, in case of forgetting.

1.7.3.2 Dashboard Module

This is the main page for system users after login successfully. There will be a sidebar which provides a list of drawer tabs for users to perform identical functions. The main layout will be built with data tables for users to have an overview of booking relevant data.

1.7.3.2.1 Administrative Users

Administrative users could have an overview of their booking records and pending booking requests in the main layout. The list of drawer tabs enables the users to either redirect to the booking module, venue/classroom management module, events management module, profile module and messages module.

1.7.3.2.2 Non-Administrative Users

Non-administrative users could have an overview of their booking records. The dashboard also provides a list of tabs where non-administrative users could access identical functions, inclusive of the booking tab, profile tab and the messages tab.

1.7.3.3 Booking Module

This module comprises venue booking-related submodules, which are accessible by all levels of users.

1.7.3.3.1 Booking Input Submodule

This submodule requires the user to choose their desired venue/classroom within UTAR and pick the event date. Users need to fill in the booking reasons too so that it fulfils the UTAR regulations for approval. Upon selection of venues, the venue’s related details will be displayed for reference, including some preview pictures, description, capacity count, facilities available and previous booking records. The information chosen here will be directed to the administrative users for further processing.

1.7.3.3.2 Request Managing Submodule

Accessible by administrative users only, this submodule aims to let them have an overview of all booking requests, regardless of pending or completed requests. For pending requests, administrative users could choose to reject or approve the request. Any actions could trigger the system to notify the booking requester through email and SMS. They could also start a conversation with any booking requesters by selecting any booking requests submitted.

1.7.3.3.3 Venue Viewing Submodule

Accessible to all levels of users, this submodule enables the users to have a view of the floor plans for both KA and KB blocks, to familiarise themselves with the campus layout.

1.7.3.3.4 Booking Details Viewing Submodule

While the data table displayed in the dashboard is just a simplified yet information-limited view, this submodule displays all its booking requests. Users can choose to generate a PDF version of that booking request by selecting the PDF function in each row of booking requests.

1.7.3.3.5 Booking Details Editing Submodule

Accessed through the booking viewing module, this submodule lets the users make amendments to booking requests. This could only work on those booking requests that have not been marked completed. It is mentionable that editing is not allowed for those booking requests that are ongoing, to prevent booking conflicts from happening.

1.7.3.4 Messages Module

Starting from the administrative users' intention to start the conversation, this module enables them to communicate with the booking requester in the form of a chatbot. Non-administrative users could view the messages archive in the Messages tab located in the sidebar. Do note that this module does not serve as a notification platform.

1.7.3.5 Profile Module

This module enables all levels of users to view their profile and edit their information if needed. They could only edit the email address and the contact number, to ensure the information could be conveyed.

1.7.3.6 Venue/Classroom Management Module

Managed by administrative-level users, this module aims to let the users update the information or status of the venues/classrooms manually if any availability conflicts arise.

1.7.3.7 Calendar Events Management Module

The administrative users could add or remove any pre-planned calendar events (such as examinations, public holidays or events) on the calendar in this module.

Technically, PickaDate.js library will be used so that the users can interact with the calendar more interactively. Stating that some dates are not available for venue/classroom booking, this module could prevent further manual communication if the booking requester chooses a date when the venue is not available.

1.7.3.8 Trimester Management Module

Accessed by DGS staff only, this module aims to let them manage the trimester information, that could relate to arranging classes daily which further relates to venue/classroom booking. The managerial functions include adding, viewing, editing and deleting the trimester information.

1.7.3.9 Class Management Module

Accessed by DGS staff only, venue/classroom could be used for daily class events. Stating that the venues may need to be blocked for booking during class hours, this module could enable the DGS staff to add classes for the entire trimester, which automatically books the venue/classroom for the timing chosen. The managerial functions include adding, viewing, editing and deleting the added class information.

1.7.4 Limitation of Project

The investigation is limited as the parties mentioned above have a direct relationship with UTAR. There were occasions when outsiders rented UTAR venues for their events, but these happenings are rare and it could be time-consuming to gather information from them. Besides that, there is no specific audience to be interviewed when it comes to gathering insights from outsiders. To solve this issue, the booking procedure is meant to be designed and implemented based on the UTAR community's opinions, since they use this system more than the public.

In terms of modules that are not covered, the registration module is not included in the project scope. As every individual in the UTAR community has a unique email address, therefore they could just log in to the system without any additional procedures. Another module is the payment function, since the

UTAR community could book the venue/classroom with zero rental fees. The payment is only applicable to outsiders, who are not counted as part of the end-users of the system.

Based on the interviews conducted with the students, the departmental staff (DGS and DSA) will inquire about the facilities they need when running an activity, like the presence of a microphone Public Address (PA) system if the activity is organized in the Multi-Purpose Hall. The system to be developed will not be featuring this input as this could be another field of managing facilities. From the opinion of DSA, this equipment/facilities preparation deserves to be featured in another individual system (Foo, 2024).

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Technological advancements have brought benefits to mankind as they automate human tasks which were being done manually all the while and the results are promising (Anil Kumar, Bawge and Kumar, 2021). Originating from the industrial sector to control the machinery operations, the trend spreads to the managerial sector and daily life since most of the tasks are repetitive and serve a general purpose. Ever since the spark of technological trends, people have started to investigate the replaceability of computing on human tasks and have been working hard to produce such technologies. For example, Microsoft Office software was released in 1990 to increase the productivity of common office work, and the features are still evolving today (Microsoft, 2023). Following the boom of Internet technology in the 90s, many tasks are transforming into web-oriented applications today, like the booking of airline tickets or any company management portal system (Amanda, et al., 2010). This ensures the information could be conveyed directly to the end-users and everyone could perform the tasks as long as they have an Internet connection, which indirectly boosts working efficiency.

Mentioning that the Web-oriented applications offer airline reservation service, these types of products are also relevant to pre-booking, like venue, accommodation and events etc. Limiting the scope to universities, there are a few academia that have this system developed for their venue bookings, like the Universiti Putra Malaysia, University Malaya, Universiti Teknologi MARA. Therefore this chapter will compare the software development approaches and analyze each university’s venue booking system specialties based on a round of use. The experience gained in this stage also helps the project author to clarify the necessary features in the system to be developed and to review the right development approach to comply. Besides that, types of hosting platforms and types of databases will be studied as well.

2.2 Development Methodologies

Software development methodologies are a formalized process that the project team needs to adhere so that the deliverables could be delivered on time, fulfilling the quality standards and facilitating communication among themselves (Dennis, Barbara and David, 2009). It is undeniable that the methodology starts from the planning phase then evolves into analysis, design and lastly the implementation phase (Dennis, Barbara and David, 2009). However as the business requirements get complex and vary according to the customers' needs nowadays, various development methodologies are introduced. The working practices could be different for the four phases depending on the methodology adopted. Generally, three types of methodologies exist, which are Agile Development, Structured Design ones and Rapid Application Development. Therefore, this subsection aims to distinguish the uniqueness of each methodology and to decide which to be applied for the UTAR Venue/Classroom Booking Web Portal.

2.2.1 Structured Design

Before this design methodology was introduced, the software development was being carried out in a chaotic environment and the activities were rated ad hoc (Dennis, Barbara and David, 2009). Seeing that the entire development cycle is a mess, Structured Design was introduced in the industry to restructure the SDLC (Software Development Life Cycle) phases to advance from phase to phase. This section aims to introduce Waterfall Development and Parallel Development as the activities for both approaches are being arranged in an orderly manner.

2.2.1.1 Waterfall Development

This is a classic software development approach that structures the development activities in a linear and subsequent method, similar to the actual waterfall which passes from one path to the following path (Dennis, Barbara and David, 2009). The succeeding tasks only start when the outputs in the preceding tasks are accepted in terms of meeting the quality. To describe in terms of SDLC, the planning phase kicks off with gaining the business requirements and project initiation (Roger, 2010). After collecting business requirements and determining the project objectives and setup, the project moves to the analysis stage to conduct requirements modelling. Then, the project moves to the design stage where the project author needs to build the software product technically. Ensuring every piece of functionalities are included in the system developed, then the system is only taken to the implementation phase for further testing. The testing results determine if the system is eligible as an end product or not. This could be represented by the following figure.

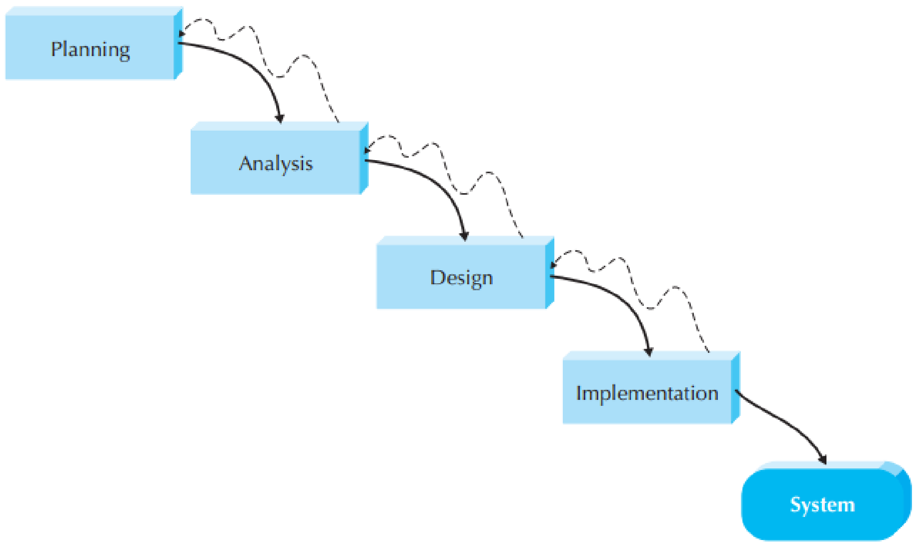


Figure 2.1: Life Cycle for Waterfall Model (Dennis, Barbara and David, 2009)

Seeing that this development approach is being organized systematically, however, the tasks are heavily dependent on each other. This causes the project team to spend more time waiting if one of the tasks requires more time to be completed, which leads to project inefficiency (Roger, 2010). Moreover, stating that the waterfall flows ascendingly, it is not flexible for the approach to revert to previous phases to conduct changes (Dennis, Barbara and David, 2009). The modern software development pace often changes from time to time radically, which is not fit for today’s software project (Roger, 2010).

However, the project author must fulfill the following characteristics if he/she wants to follow this development approach, which the customers are clear on the business requirements and they need to be clearly defined at the analysis stage, so that changes in scope could be prevented. Another point is that the customer must be patient to examine the end product (Roger, 2010). It is only until the end of the phase then the customer can test out the product. However, changes could be deadly if the customer is unsatisfied with the deliverable. Therefore, the Waterfall Model is highly avoided when it comes to selecting the right development approach.

2.2.1.2 Parallel Development

In contrast to following the sequential phases like what is being portrayed in the Waterfall Model, the Parallel Development model is being proposed to speed up the delivery of each project activity (Dennis, Barbara and David, 2009). The activities in the design phase are split into subprojects to run simultaneously, after conducting the planning phase and analysis which identifies the project requirements (Dennis, Barbara and David, 2009). The splitting enables the project team to focus on designing the functionalities to be delivered and avoiding project slack.

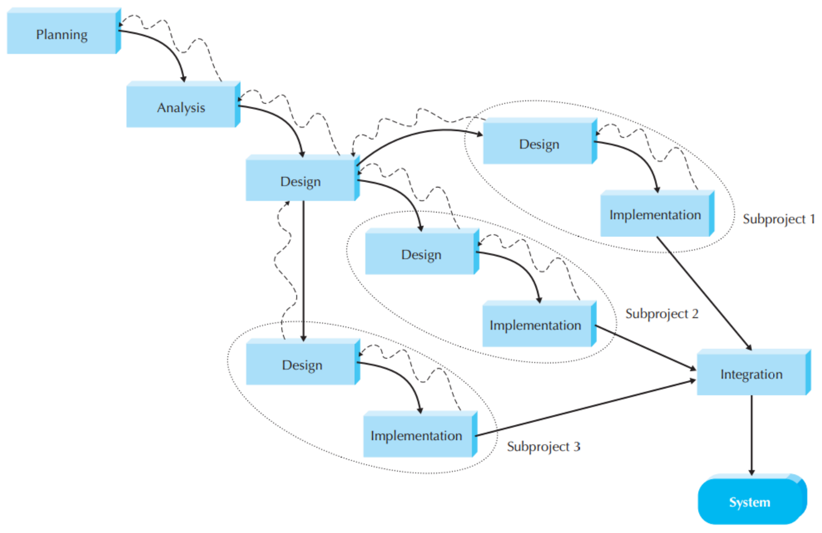


Figure 2.2: Software Development Life Cycle for Parallel Development (Dennis, Barbara and David, 2009)

Referring to Figure 2.2, the subprojects which had done designing will be integrated to form a complete final software end product. It was discussed that this approach could greatly reduce the project schedule to deliver the final product, however some subprojects may be correlated with each other (Dennis, Barbara and David, 2009). Last-minute changes or decisions could affect another dependent subproject, causing possible schedule delays and mostly more integration work to be done (Dennis, Barbara and David, 2009).

To comment, Parallel Development is ideal for software projects that have a complex scope, and the project structure is reliable. Project teams having a rush schedule could consider this approach but not project teams that have unclear schedules (Dennis, Barbara and David, 2009). This approach urges the project team to have clearly defined user requirements and experienced in the technology used for development (Dennis, Barbara and David, 2009).

2.2.2 Rapid Application Development

Structured design methodologies possess disadvantages that may cause inefficiencies in software development like delaying project deliverables and consuming more project cost. A modern development branch, which is the Rapid Application Development (RAD) is therefore presented to counter these problems, through shifting the SDLC phases to get the software project delivering something working as fast as it could (Dennis, Barbara and David, 2009). Compared with the Structured Design methodologies, the RAD makes use of strategic digital tools to facilitate the SDLC activities, raising the delivery speed and the overall quality of the software project (Dennis, Barbara and David, 2009).

This subsection aims to introduce two methodologies under this development branch, which could aid the project author in selecting the right methodology to develop.

2.2.2.1 Phased

Though the structured development approach only releases the software product to the customer after the entire development, this will cause an enormous amount of rework and resources to be spent to cater the customers’ changes. To prevent the situation described above from happening, regular revision could be introduced where the entire software project is divided into repetitive cycles so that different issues could be realized in each stage and efforts could be distributed to tackle the challenges faced.

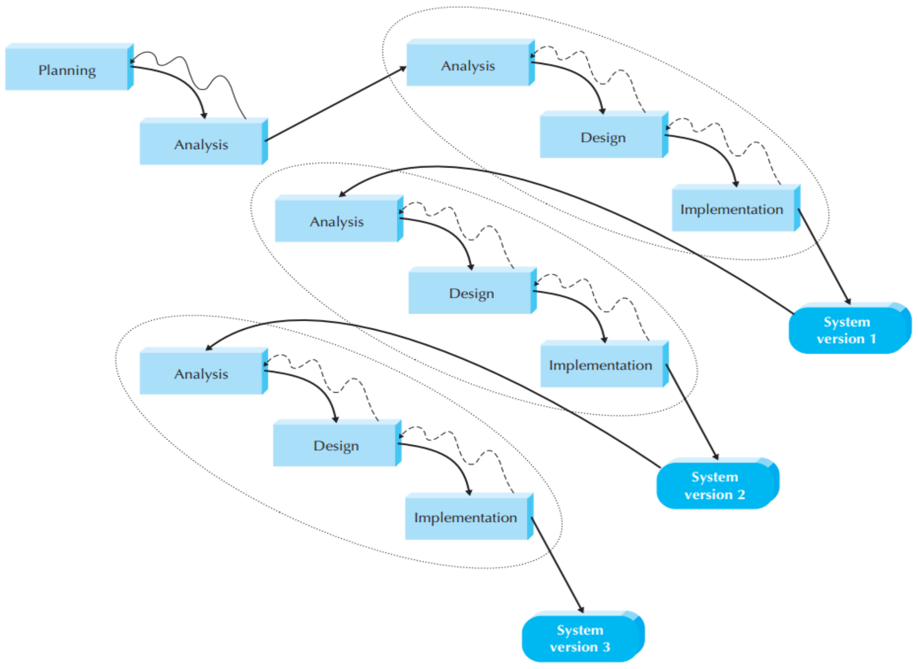


Figure 2.3: Software Development Life Cycle for Phased Development Approach (Dennis, Barbara and David, 2009)

Based on the figure above, it was observed that the software product had been segregated into several versions for development. Key requirements will be built for the first version release, followed by the second release and so on to include more lower-prioritized requirements (Dennis, Barbara and David, 2009). With this approach implemented, the project author could know the issues reported by the users, which corrective actions could be taken then included in the succeeding version release. Regardless of the existence of problems reported, the project author still needs to undergo the usual SDLC in preparing for the next version release. Further analysis will need to be conducted on requirements prepared from the project initiation stage, so that the project author gets a deeper understanding on the scope (Dennis, Barbara and David, 2009). The entire process repeats until the project development is marked as complete or the system developed is obsolete (Dennis, Barbara and David, 2009).

Portraying the advantages of getting the software product fast to the customers, this could increase the business value of the project author little by little. Moreover, the project author is flexible in adding more ideas when it comes to revising the previous phases. Through arranging the importance order of the functionalities to be included, it enables the project author to have more time to brainstorm extra requirements. Therefore, project teams who are working under phased development must have the ability to recognize which requirements should be ranked further, since some of the functions are working concurrently and need to be delivered in the first few versions to gain valuable feedback (Dennis, Barbara and David, 2009). Phased methodology is considered an all-rounder, as it gives ample time for the project team to identify and learn if more efforts are needed to be spent on clarifying the user requirements and learning new development technologies (Dennis, Barbara and David, 2009). Software projects which have tight schedules and high complexity levels could also implement this methodology (Dennis, Barbara and David, 2009).

2.2.2.2 Prototyping

Customers could be unclear about the details of the requirements due to a lack of knowledge of the scope. As such, the customers could only give a general description of the functionalities that they desired (Roger, 2010). This could result in the project author being hesitant about what should be featured in the system to match the expectations. Seeing such a situation surface, this methodology is introduced. In this methodology, a prototype or simply a sample model will be built throughout the entire SDLC, functioning as a communication medium between the project author and the customers so that the former could get a better understanding of what the customers want (Roger, 2010).

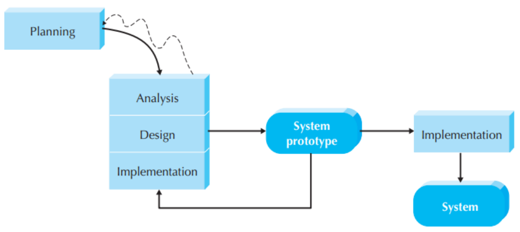


Figure 2.4: Software Development Life Cycle for Prototyping methodology (Dennis, Barbara and David, 2009)

Figure 2.4 depicts the SDLC phases flows of the Prototyping methodology. Like any other methodologies, identification of requirements will be carried out in the planning stage through communication. Then the project author will iterate the analysis, design and implementation phase to set up a prototype that provides a simple layout and visible features (Dennis, Barbara and David, 2009). The aim of releasing the prototype model to the customers periodically is to gain their feedback, whether the prototype matches their expectations or any improvement areas (Roger, 2010). The entire process mentioned will be looping until the prototype delivers all functions (Dennis, Barbara and David, 2009). The prototype will be refined to become the actual product that is appropriate to use.

Software projects which adopt this methodology will be executed in a fast-paced fashion as a cycle of these activities is to be carried out as fast as possible, to get the deliverables formalized within the deadline (Dennis, Barbara and David, 2009). However, there is a deadly drawback in this methodology, that is the prototype which had been evolving for a duration may get its initial features affected, if there are drastic amendments suggested in the middle (Dennis, Barbara and David, 2009). Therefore, these fundamental aspects need to be discovered through thorough study and discussion so that reworking could be reduced to the lowest cost.

Speaking of prototyping, it depends on the project team's considerations whether to modify the prototype continuously to the end product, or to discard then design a formal system based on the feedback gathered (Dennis, Barbara and David, 2009). Generally, this methodology is suitable for software projects in which the user requirements are unclear, and the schedule is short but clear (Dennis, Barbara and David, 2009). Though the processes need to proceed quickly, it is not suitable for complex software projects, and the project team has mere knowledge of the development technologies (Dennis, Barbara and David, 2009).

2.2.3 Agile Development

With new business factors often arising suddenly, end-users’ business requirements also switch rapidly to respond to the changes and stay competitive. Responding to changes could be costly as efforts need to be made to revise and rework, which could be not effective if applying this practice in methodologies stated before. Therefore, this methodology introduced is to provide a practical mechanism for the project author to adapt to the changes appropriately, mostly by reducing the modelling work and documentation to be done, in contrast with the Structured Design and Rapid Application Development (Dennis, Barbara and David, 2009). Stating that unpredictability occurs from time to time, the changes must be presented incrementally in quality, with the same pace as with the changes stated.

Agile Development stresses such values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan (Roger, 2010).

Extreme Programming (XP) is one of the common techniques used in Agile software development. To relate to the core values mentioned above, XP practices fruitful communication with the customers by encouraging informal collaborations between the project team and the end-users (Dennis, Barbara and David, 2009). This could urge the bond to be established between both parties and feedback could be provided continuously to improve the software product. With the informal form of communication, XP however does not put much attention on documentation as another form of communication medium (Dennis, Barbara and David, 2009). XP primarily focuses on the working software at the end, more than tracking over details recorded in the documentation.

Through encouraging the customers to voice their concerns, XP appreciates these viewpoints which could be received from three angles, that are the software product, the end-users and the project team (Roger, 2010). The test results after executing planned testing strategies could be a form of feedback to the project author. While the opinions conveyed by the end-users are recorded in the form of use cases, so that the system analysts could identify the inputs, outputs, processes to be performed and related use case descriptions. As effective communication occurs bidirectionally, the project author will respond as well by updating the end-users regarding the software changes along with the project managerial impact.

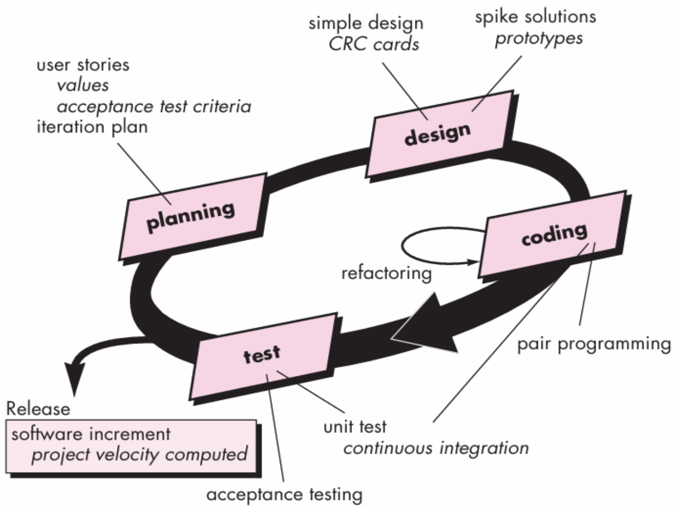


Figure 2.5: Software Development Life Cycle for Extreme Programming (Roger, 2010)

In terms of SDLC activities which is illustrated by Figure 2.5 above, XP initiates the planning phase by collecting the stakeholders' requirements in the form of stories and being assigned a priority value and development duration (Dennis, Barbara and David, 2009). The stories gathered will be grouped by the project team and the stakeholders to decide which to be included in the next release. Be aware that XP is a type of Agile development, therefore these stories could be rewritten, eliminated or split in the future (Roger, 2010). Moving to the design phase, XP complies with the KIS (Keep It Simple) rule where a simple design is used as a guideline when it comes to the implementation phase where each story written needs to be presented (Roger, 2010). Though mentioning simple, excessive functionalities are not advisable to include in the design. Upon designing, refactoring which is known as a software practice to modify code structures often takes place in XP, to refine the code design.

The next phase will be coding. In XP, the project author does not write source codes but writes unit tests for the software product (Roger, 2010). The tests planned could aid the project author in getting a full understanding of what functionalities to be included to pass the tests. Pair programming which two people work on a single story is practiced in XP, with the concept of combination of ideas that could ensure the source codes are written to fulfill the standards other than just performing the desired functions only (Roger, 2010). The source code that passed the unit tests was then compiled with works from other pairs. Stated that unit tests were written to examine the functionalities to be delivered by the source codes, however there is a testing phase where the software project gradually moves on. This round of testing could be treated as regression testing, to test if the changes conducted before do not affect the properly functioning codes (Roger, 2010). Acceptance testing also takes place and is normally conducted by customers, to review the software product features and performance (Roger, 2010).

Agile development methodologies require a strong sense of discipline and teamwork so that the changes could be conveyed in time and the results are presentable to the stakeholders (Dennis, Barbara and David, 2009). Furthermore, it is suitable for small-scaled yet experienced project teams, as they have the confidence to deal with the overhead caused by the sudden changes (Dennis, Barbara and David, 2009). Regarding the nature of the software project, this methodology is not advised to be adopted in critical software projects, since this methodology practices less documentation which results in challenging maintenance (Dennis, Barbara and David, 2009). Generally, Agile development methodologies are best used with software projects which have ambiguous user requirements, experienced project team, a non-critical nature and a short time yet clearly defined schedule.

2.2.4 Summary

There are several factors to consider when it comes to choosing the right software development approach, and some of them have been described in each approach. This could be summarized in the following table.

Table 2.1: Comparison Table for each Software Development Approach (Dennis, Barbara and David, 2009)

Aspects / Type	**Waterfall**	**Parallel**	**Phased**	**Prototyping**	**Extreme Programming**

With Unfamiliar Technology	Poor	Poor	Good	Poor	Poor
Reliable Project	Good	Good	Good	Poor	Good
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good
Complex Project	Good	Good	Good	Poor	Poor
With Short time Schedule	Poor	Good	Excellent	Excellent	Excellent

The functional requirements are collected through qualitative and quantitative studies, it is considered clear as it comprises what is being encountered by the end-users and their descriptive expectations of the system to be developed. The project is determined to be constructed using web development technologies, therefore the project author is confident to build it using PHP as the mainstream since the project author has experience in using this language. The system is simple to understand but it could be complex when it comes to validation and authentication of the information flow as some functions could be restricted to certain levels of users.

Seeing that the phased development approach enables the project author to focus on each feature to be released in an iteration, the project author decides to adopt this as the backbone for supporting the entire project development. Furthermore, phased development could motivate the project author to be disciplined in building the system as the project timeline could be limited.

2.3 Review of Databases used for System to be Developed

Data flows are essential and it urges the importance of the database model to be used. Other than storing the pre-defined data like the venues' names and capacity, the database should support flexible information amendment and retrieval. As the modern database models are classified into relational and non-relational types, therefore a study is required to determine the suitable data model to be implemented for this project.

2.3.1 Relational Model

As its name suggests, the relational model classifies the database into relations, where each relation represents a table with rows and columns filled with values (Elmasri and Navathe, 2016). Each row signifies a unique object. In terms of schema architecture, this model is structured until the attributes within a relation are consistent with the data format and the information stored (Elmasri and Navathe, 2016). Every relation is interlinked to ensure the relationships between the information are established and support various ways to view the data. This type of model is heavily constrained to show the consistency of the overall database model and the validness of the attributes stored (Elmasri and Navathe, 2016). For example, the key constraint specifies that a table should have a non-repetitive yet unique key (primary key) to identify the uniqueness of an attribute and the attribute constraint states whether null values should be omitted or not. This characteristic enhances the stability of the Relational Model by structuring the relations through normalization (Elmasri and Navathe, 2016).

Technically, the relational model uses Structured Query Language (SQL) to perform data transactions, being adopted widely in MySQL and Oracle databases (Garba and Abubakar, 2020). There are a few relational model databases available in the market, such as the open-source Microsoft Access, PostgreSQL or integrated ones. As the platform selected to host the system is XAMPP, it comes with the default database model which is the PHPMyAdmin.

2.3.2 Non-Relational Model

Considering the advancement of Internet technologies which attracted lots of users and resulted in a huge amount of data to be stored, a more flexible data storage system is suggested in this case (Elmasri and Navathe, 2016). This is because the traditional relational models are too restrictive, causing more efforts to be made to scale the database to suit one's needs (Elmasri and Navathe, 2016). Another aspect is that non-relational models do not require a schema to be built (Elmasri and Navathe, 2016). Normally known as NoSQL, users could define their schema and it turns out to be less restrictive like what the relational model does (Elmasri and Navathe, 2016). Therefore, this database model supports multiple types of data storing options, which the four main types could be the document-based NoSQL (stores in JSON format, which could be accessed by identical document ID), NoSQL key-values stores (stores data by assigning a key to it), graph-based NoSQL (stores in graphs which displays connecting nodes between data) and column-based NoSQL (partitioning table into column families to store information) (Elmasri and Navathe, 2016). For example, Amazon which uses DynamoDB implements the key-value data stores while Facebook which utilizes Cassandra mixes the concept of key-value stores and column-based systems (Garba and Abubakar, 2020). Due to its flexible nature, more development projects have adopted this data model.

2.3.3 Summary

Mentioning that the scope of the system to be developed is controlled in the UTAR community and the information to be handled is simple yet precise, the relational database model is best suited for this case as most of the data are interlinked to present the relationships between the data. As each relation (table) is built specifically for a purpose, there is a demand for normalized data so that data anomalies such as insertion anomalies (insertion fails due to the presence of null values), update anomalies (update fails on another attribute due to having same value on different relations) and deletion anomalies (one record fails to be deleted due to having interrelated value with another relation) can be prevented, as well as maintaining the data consistency. This matches the properties of relational models where it emphasizes consistency in maintaining database state. A database should stay stable before the execution of data manipulation, meaning that the modification of data should take place only after the execution of a transaction.

Furthermore, the scope and demanding size are stable which do not need the use of non-relational database models. The requirements have been defined in the initiation phase and would not have to be redrafted frequently since the project scope is controlled to venue/classroom booking in the UTAR environment. While the non-relational model is suitable for projects that have relatively more data that changes rapidly and unstructured ones (Elmasri and Navathe, 2016).

The learning curve for the Relational Model is relatively smaller compared to the Non-Relational Model to the project author. Using the Relational database model is expected to reduce the learning effort as it uses DDL (Data Manipulation Language) to perform data manipulation. Technically, there is no standardized syntax upon using the Non-relational data model, depending on the database platform used (Garba and Abubakar, 2020). Mentioning the data in the Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal are normalized yet interlinked, the model chosen should be capable of supporting complex queries so that effective retrieval or manipulation could take effect. The relational model which utilizes SQL could perform dynamic functions than the non-relational model in case of this, such as to use of multiple join conditions or nested queries.

The deciding factors of choosing the right database model are being compared and summarized in the following table:

Table 2.2: Comparison Table for database Relational Model and Non-Relational Model (Garba and Abubakar, 2020)

Aspects / **Type Model**	**Relational Model**	**Non-relational Model**
Architecture Structure	Schema	● Document-based (JSON) ● Key-values stores

		● Graph-based ● Column-based
Flexibility	Low	High
Degree of Data Normalization	High	Low
Language	SQL performing Data Definition Language	No standardized language, depending on which database solution provider chosen
Query capability	Supports complex joining, nested queries	Supports single data retrieval only
Data Format	● Structured ● Smaller amount of data compared with non-relational model ● Data changes are not radical	● Could be mix of structured and unstructured data ● Larger amount of data compared with relational model ● Fast data changes
Scale of project	Project's scope is mostly fixed and minor adjustments on requirements	Project's scope is wide and requirement changes are radical

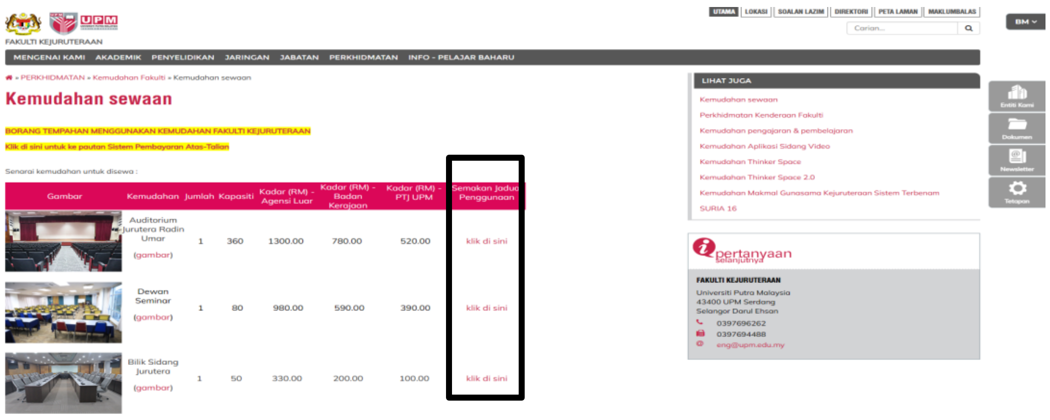
2.4 Review of Existing Similar Systems

To further explore what a comprehensive venue/classroom booking system should be equipped with, a few systems that were specifically dedicated to performing this purpose in Malaysian universities have been selected for comparison. The booking scope is controlled at requesting a lab or classroom, since these facilities are commonly used by the lecturers and the students. This section aims to describe the findings based on the research on these systems, mostly focusing on the functionalities and usage experience.

2.4.1 Universiti Putra Malaysia (UPM)

Developed through web hosting, the venue/classroom booking system for Universiti Putra Malaysia is diversified, it is available based on different faculties. Since the Faculty of Engineering holds the most information on classroom booking, therefore this faculty's system is referenced. To explain the design structure, the website contents are written in Malay language, but giving the users an option to change the language to English. However, only some of the elements have been changed, which could be hostile to use for users who are not familiar with the Malay language. Nevertheless, the website's layout is simple yet informative to use, which enables the users to navigate between pages effortlessly, without being affected by the language factor. In terms of accessibility, the website could be easily accessed through searching in the search engines. Kindly refer to Figure 2.6 to look at the design structure described.

In terms of checking the availability of venues on a specific date, UPM did a decent job on this aspect. As this is the functionality that most users want, UPM offers this feature in the form of schedule checking. Being listed together with other essential information like the payment rate and the capacity count, the users could click a directive link that redirects them to a Google Calendar widget which enables them to check if the venue chosen is occupied for a chosen date or not. The widget is purely for checking the events only, supported with the event descriptions and event time which is clear enough for the users to know the status of the venue. However this checking availability feature is only available to those functional halls, not for the labs or classrooms. This feature should be open to all facilities, not to those that may cater outsiders' requirements. Kindly refer to the bordered parts in Figure 2.6 to know how the directive links work and Figure 2.7 to look at the Google Widget described.



UTAMA | LOKASI | SOALAN LAZIM | DIREKTORI | PETA LAMAN | MAKLUMBALAS

FAKULTI KEJURUTERAAN

MENGENAI KAMI AKADEMIK PENYELIDIKAN JARINGAN JABATAN PERKHIDMATAN INFO - PELAJAR BAHARU

» PERKHIDMATAN » Kemudahan Fakulti » Kemudahan sewaan

Kemudahan sewaan

BORANG TEMPAHAN MENGGUNAKAN KEMUDAHAN FAKULTI KEJURUTERAAN

Klik di sini untuk ke pautan Sistem Pembayaran Atas-Talian

Senarai kemudahan untuk disewa :

Gambar	Kemudahan	Jumlah	Kapasiti	Kadar (RM) - Agensi Luar	Kadar (RM) - Badan Kerajaan	Kadar (RM) - PTJ UPM	Semakan Jadual Penggunaan
	Auditorium Jurutera Radin Umar (gambar)	1	360	1300.00	780.00	520.00	klik di sini
	Dewan Seminar (gambar)	1	80	980.00	590.00	390.00	klik di sini
	Bilik Sidang Jurutera (gambar)	1	50	330.00	200.00	100.00	klik di sini

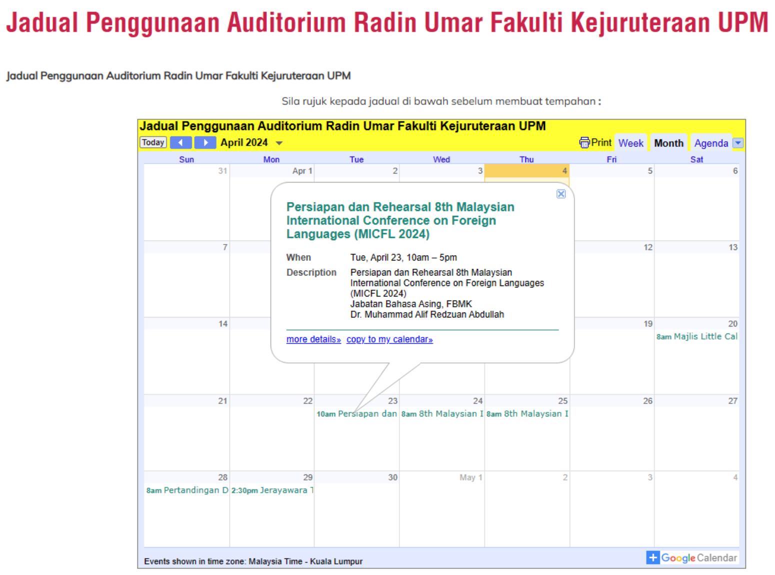
LIHAT JUGA

Kemudahan sewaan
Perkhidmatan Kenderaan Fakulti
Kemudahan pengajaran & pembelajaran
Kemudahan Aplikasi Sidang Video
Kemudahan Thinker Space
Kemudahan Thinker Space 2.0
Kemudahan Makmal Gunasama Kejuruteraan Sistem Terbenam
SURIA 16

pertanyaan selanjutnya

FAKULTI KEJURUTERAAN
Universiti Putra Malaysia
43400 UPM Serdang
Selangor Darul Ehsan
0397696262
0397694488
eng@upm.edu.my

Figure 2.6: Design Structure of the UPM Faculty of Engineering’s venue booking website



Jadual Penggunaan Auditorium Radin Umar Fakulti Kejuruteraan UPM

Jadual Penggunaan Auditorium Radin Umar Fakulti Kejuruteraan UPM

Sila rujuk kepada jadual di bawah sebelum membuat tempahan :

Jadual Penggunaan Auditorium Radin Umar Fakulti Kejuruteraan UPM

Today April 2024 Print Week Month Agenda

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	Apr 1	2	3	4	5	6
7					12	13
14					19	20 8am Majlis Little Cal
21	22	23 10am Persiapan dan	24 8am 8th Malaysian I	25 8am 8th Malaysian I	26	27
28 8am Pertandingan D	29 2:30pm Jerayawara T	30	May 1	2	3	4

Persiapan dan Rehearsal 8th Malaysian International Conference on Foreign Languages (MICFL 2024)

When Tue, April 23, 10am – 5pm

Description Persiapan dan Rehearsal 8th Malaysian International Conference on Foreign Languages (MICFL 2024)
Jabatan Bahasa Asing, FBMK
Dr. Muhammad Alif Redzuan Abdullah

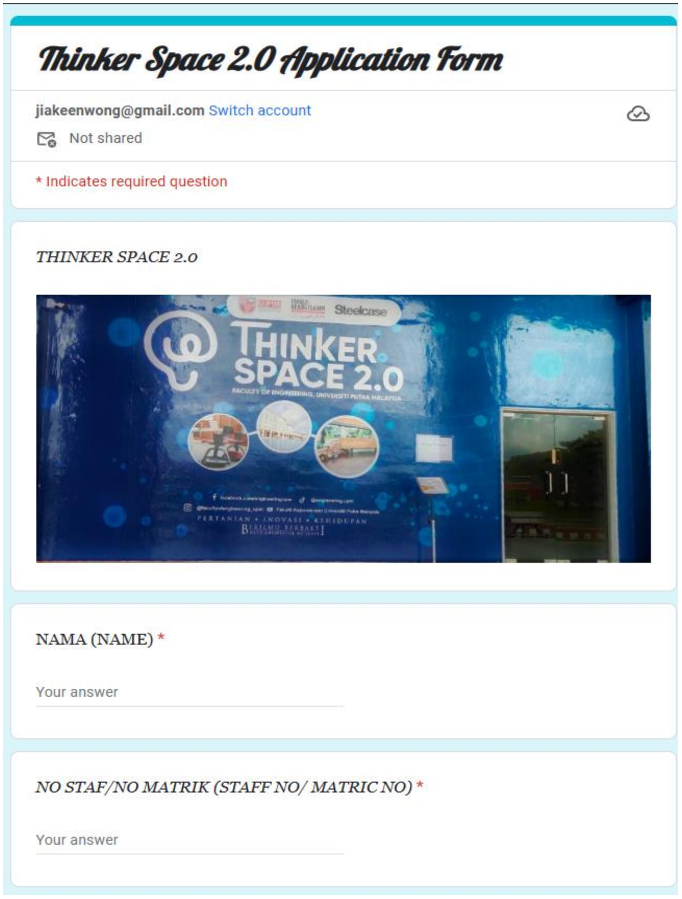
more details» copy to my calendar»

Events shown in time zone: Malaysia Time - Kuala Lumpur

Figure 2.7: Google Calendar widget

UPM also provides clear information on the contact details for each venue/classroom. The person in-charge’s names, email addresses and contact numbers are marked at the bottom section of each venue/classroom website. Other than that, the faculty’s contact number, email address and mailing address are also listed for the users to refer.

It was noticed that the booking procedure is inconsistent for every venue/classroom. While most of the venues/classrooms are required to book using a Google Form or Zoho form, some could even be accessed through a generated QR code which other venues/classrooms may not have. This happened due to handling by multiple persons in-charge. Separated with the payment function, the format of the booking forms are standard, which requires common booking details, like the booking reason and capacity count. It could be better if the UPM could standardize all venues/classrooms under a form. Do refer to Figure 2.8 and Figure 2.9 to know how the forms look like.



Thinker Space 2.0 Application Form

jiakeenwong@gmail.com Switch account

Not shared

* Indicates required question

THINKER SPACE 2.0

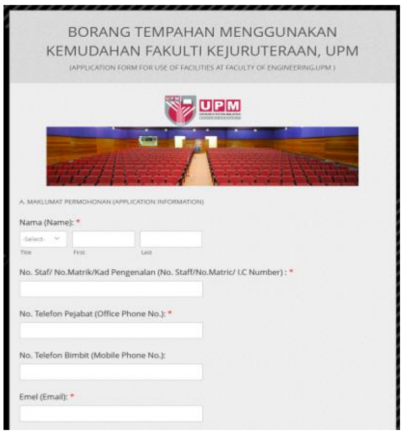
NAMA (NAME) *

Your answer

NO STAF/NO MATRIK (STAFF NO/ MATRIC NO) *

Your answer

Figure 2.8: Snapshot of one of the venue booking forms, hosted through Google Form



BORANG TEMPAHAN MENGGUNAKAN KEMUDAHAN FAKULTI KEJURUTERAAN, UPM

(APPLICATION FORM FOR USE OF FACILITIES AT FACULTY OF ENGINEERING,UPM)

A. MAKLUMAT PERMOHONAN (APPLICATION INFORMATION)

Nama (Name): *

-Select- Title First Last

No. Staf/ No.Matrik/Kad Pengenalan (No. Staff/No.Matric/ I.C Number) : *

No. Telefon Pejabat (Office Phone No.): *

No. Telefon Bimbit (Mobile Phone No.):

Emel (Email): *

Figure 2.9: Snapshot of one of the venue booking forms, hosted through Zoho Form

2.4.2 Universiti Teknologi MARA (UiTM)

The venue/classroom booking system for UiTM is hosted on the web. It enables public users, UiTM alumni, UiTM lecturers and students to use it. Like any other system, the new user needs to register. This website could be searched on the search engines and the main page shows the title which is not misleading. Refer to Figure 2.10 to see the homepage of the venue booking system of UiTM.

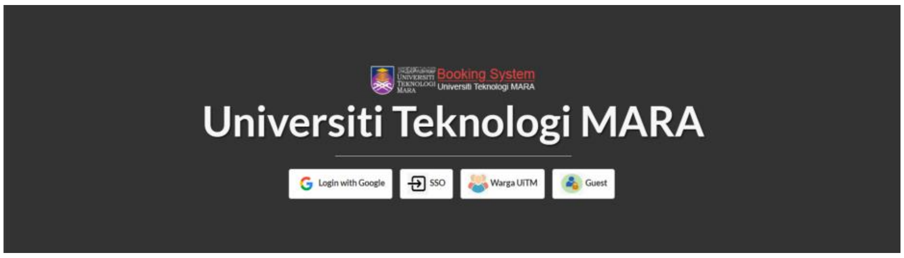


Figure 2.10: Snapshot of the homepage of UiTM venue booking system

Users are directed to a dashboard page after login. There are other services offered like to book equipment or to reserve a software, but booking venue will be prioritized in this study since it relates with the project scope. The dashboard will show the number of venue, equipment and software reservations submitted by the user itself. Along with the numbers will be a calendar, which shows the status of the venue booking requests made by the user, whether they are approved or in the progress of reviewing. It is noteworthy to mention that the UiTM website's elements are built in the English language, which could be usable for all nations of users. Refer to Figure 2.11 to see the dashboard page.

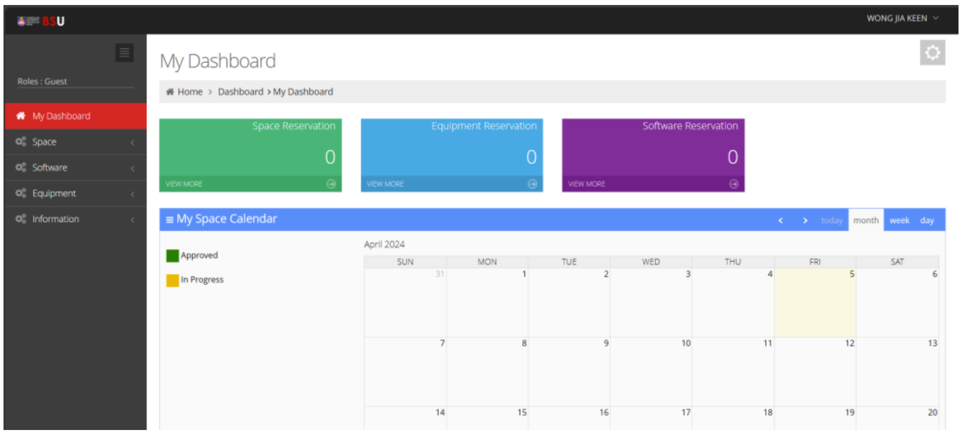
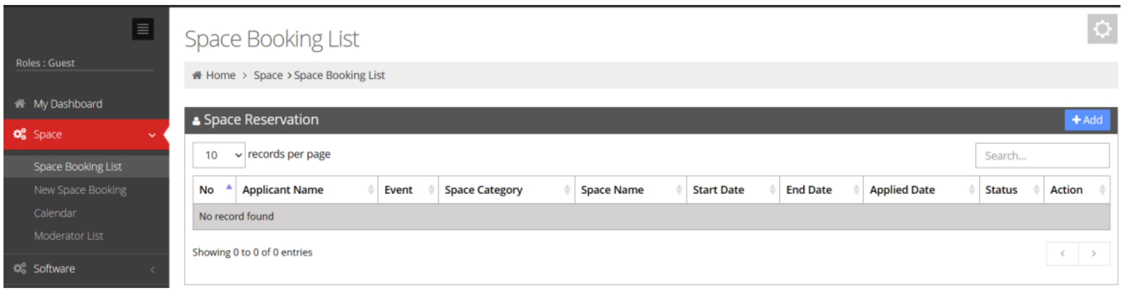

BSU
WONG JIA KEEN
Roles : Guest
My Dashboard
Space
Software
Equipment
Information
My Dashboard
Home > Dashboard > My Dashboard
Space Reservation 0 VIEW MORE
Equipment Reservation 0 VIEW MORE
Software Reservation 0 VIEW MORE
My Space Calendar
today month week day
Approved
In Progress
April 2024
SUN MON TUE WED THU FRI SAT

Figure 2.11: Dashboard of the UiTM venue booking page

From the sidebar venue booking submenu, there are four options for the user to choose. The first one will be the reservation viewing list, which is responsible to fetch all venue booking records made by the user. Representing in the table form, the user is provided with the option to customize how much data to be shown in the table. Refer to Figure 2.12 to see the request list.



Space Booking List

Home › Space › Space Booking List

Space Reservation + Add

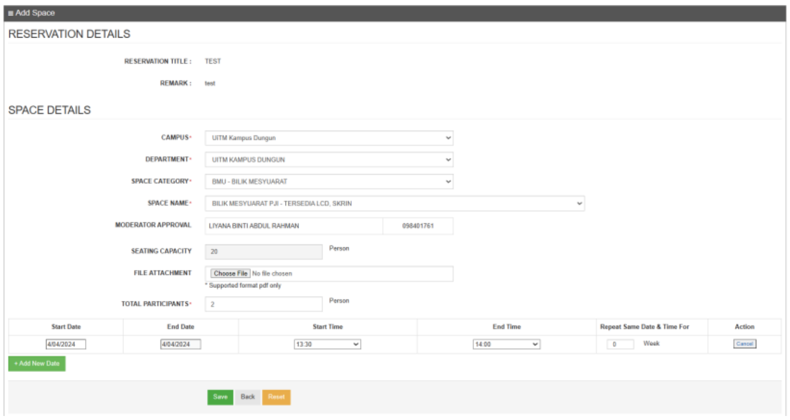
10 records per page Search...

No	Applicant Name	Event	Space Category	Space Name	Start Date	End Date	Applied Date	Status	Action
No record found									

Showing 0 to 0 of 0 entries

Figure 2.12: Request Viewing page

Upon adding new booking, it was noticed that the user needs to agree on a Malay-language written agreement regarding the terms and conditions to adhere when using the venue booked. Next, the user will need to follow a series of steps by filling the booking objective and any remarks, followed by the process of choosing the right campus, right faculty where the venue belongs to, the venue desired, the capacity count and any file attachments that support the booking. After choosing the venue, both the moderator and the person in-charge’s name and contact details will be shown, giving the users a helpful guide when it comes to facing usage problems. It is mentionable that the system allows the venue to be booked repetitively for the same date and time chosen by the user, which allows consecutive venue bookings to be completed in one shot. Refer to Figure 2.13 to get a look at the booking form.



RESERVATION DETAILS

RESERVATION TITLE :	TEST
REMARK :	test

SPACE DETAILS

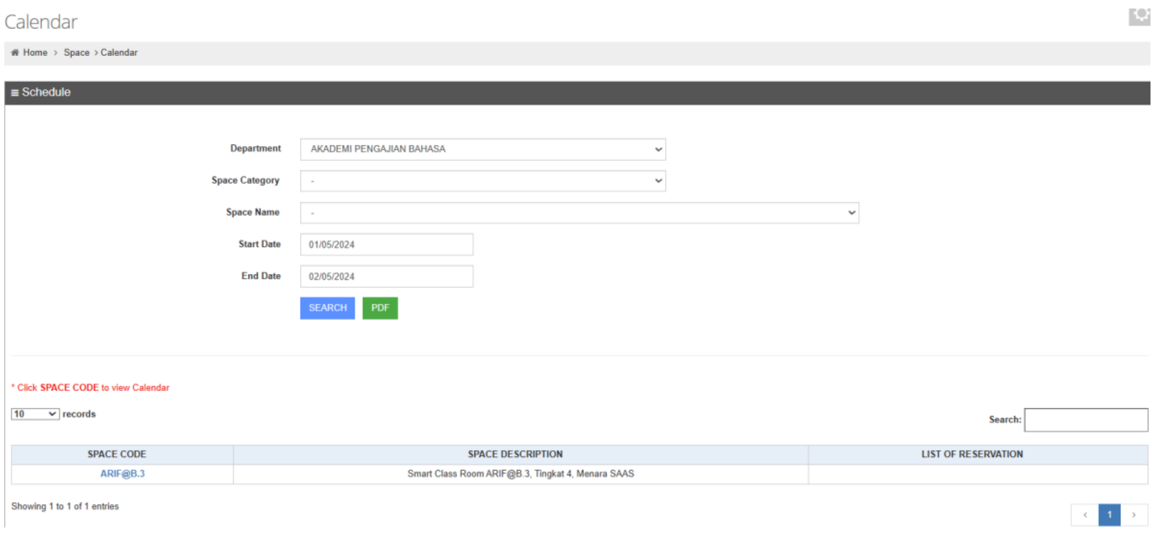
CAMPUS*	UITM Kampus Dungun	
DEPARTMENT*	UITM KAMPUS DUNGUN	
SPACE CATEGORY*	BMU - BILIK MESYUARAT	
SPACE NAME*	BILIK MESYUARAT PJI - TERSEDIA LCD, SKRIN	
MODERATOR APPROVAL	LIYANA BINTI ABDUL RAHMAN	098401761
SEATING CAPACITY	20	Person
FILE ATTACHMENT	Choose File No file chosen * Supported format pdf only	
TOTAL PARTICIPANTS*	2	Person

Start Date	End Date	Start Time	End Time	Repeat Same Date & Time For	Action
4/04/2024	4/04/2024	13:30	14:00	0 Week	Cancel

Figure 2.13: Process of submitting a venue booking request

However, insertion anomaly happens here, as if the passed dates are allowed to book. This issue should be noticeable as it could cause a conflict in booking logics. After confirming booking, there are also no indicative messages that inform the user about the status, which could be misleading for the booking requester at times.

The user can check the availability status of a venue by choosing the "Calendar" option in the sidebar. The user needs to choose the right venue and the date to check, so that any events booked will be fetched from the database and displayed in the underneath table. Do refer to Figure 2.14 to see how the checking works.


Calendar

Home › Space › Calendar

≡ Schedule

Department	AKADEMI PENGAJIAN BAHASA
Space Category	-
Space Name	-
Start Date	01/05/2024
End Date	02/05/2024

* Click SPACE CODE to view Calendar

10 records

Search:

SPACE CODE	SPACE DESCRIPTION	LIST OF RESERVATION
ARIF@B.3	Smart Class Room ARIF@B.3, Tingkat 4, Menara SAAS	

Showing 1 to 1 of 1 entries

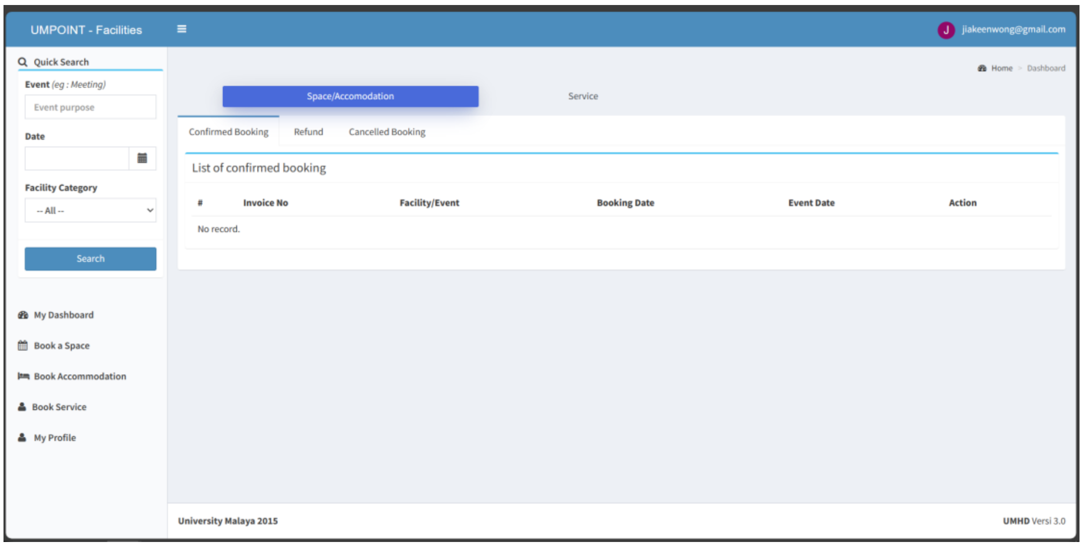
Figure 2.14: Interface of the Calendar function, to check the availability status of a venue

While the last option in the sidebar is the moderator lists, unfortunately the page is down due to technical reasons. To conclude, the UiTM venue booking system lacks many descriptive features that could aid the user to make decisions when it comes to venue booking, which is considered a function to include in the development project.

2.4.3 Universiti Malaya (UM)

The venue/classroom booking system for Universiti Malaya (UM) is hosted on the web as well, which enables the parties from the university (students and staff) and public users to use the system. Also requires signup for authentication and validation, the website is easy to trace as it is searchable on the Internet like the two systems mentioned above.

Users are directed to a dashboard page after login. There are other services offered like booking accommodation or reserving a service, but booking venue will be prioritized in this study since it relates with the project scope. The page is built with a layout which consists of tabs that enable the users to view confirmed, refunded and cancelled bookings. Other than that, there is a sidebar consisting of multiple directive links and a quick searching feature, enabling the users to find their desired venue to book. The overall website is designed in simple-fashioned, and the information displayed is believed to be on par with the user's request. The website’s contents are built in the English language. Refer to Figure 2.15 to understand the layout of the website.

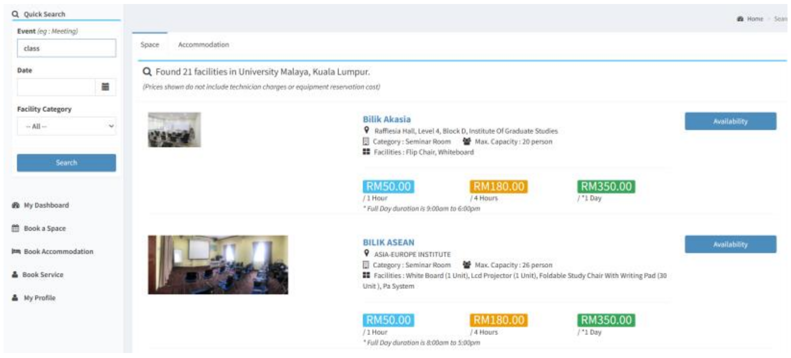


List of confirmed booking

#	Invoice No	Facility/Event	Booking Date	Event Date	Action
No record.					

Figure 2.15: Snapshot of the Universiti Malaya’s venue booking website

The users could type their preferences like event purpose, event date and facility category to choose their desired venue. From the searching results generated, each option is provided with the venue name, exact location position, which category it belongs to, the capacity count, the facilities ready and the payment rate. Do refer to Figure 2.16 to know the representation of the searching results.



Found 21 facilities in University Malaya, Kuala Lumpur.

(Prices shown do not include technician charges or equipment reservation cost)

Bilik Akasia

Rafflesia Hall, Level 4, Block D, Institute Of Graduate Studies

Category : Seminar Room Max. Capacity : 20 person

Facilities : Flip Chair, Whiteboard

RM50.00 / 1 Hour RM180.00 / 4 Hours RM350.00 / *1 Day

** Full Day duration is 9:00am to 6:00pm*

BILIK ASEAN

ASIA-EUROPE INSTITUTE

Category : Seminar Room Max. Capacity : 26 person

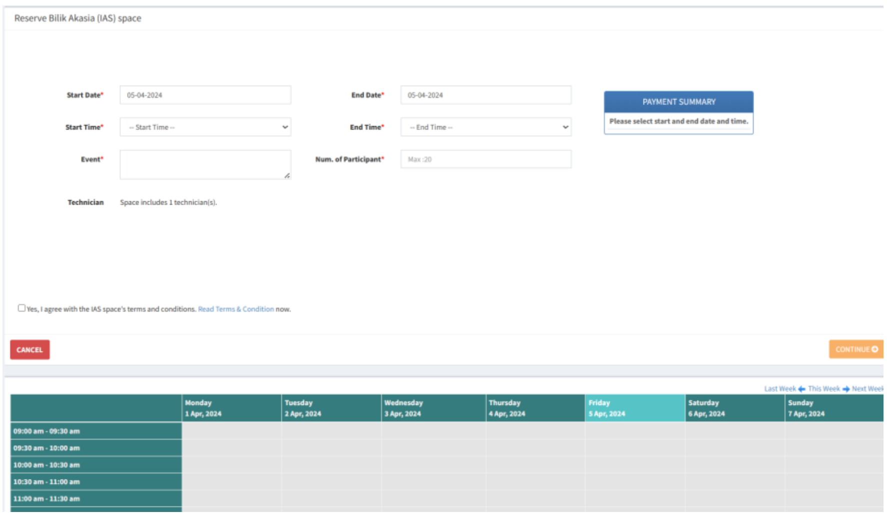
Facilities : White Board (1 Unit), Lcd Projector (1 Unit), Foldable Study Chair With Writing Pad (30 Unit), Pa System

RM50.00 / 1 Hour RM180.00 / 4 Hours RM350.00 / *1 Day

** Full Day duration is 8:00am to 5:00pm*

Figure 2.16: Searching results based on the user preferences input in the left sidebar

It also supports the feature of checking the availability of the venue/classroom, by displaying the time slot for a day in the calendar form. Do refer to Figure 2.17 to know the check availability interface.



Reserve Bilik Akasia (IAS) space

Start Date*	05-04-2024	End Date*	05-04-2024
Start Time*	-- Start Time --	End Time*	-- End Time --
Event*		Num. of Participant*	Max :20
Technician	Space includes 1 technician(s).		

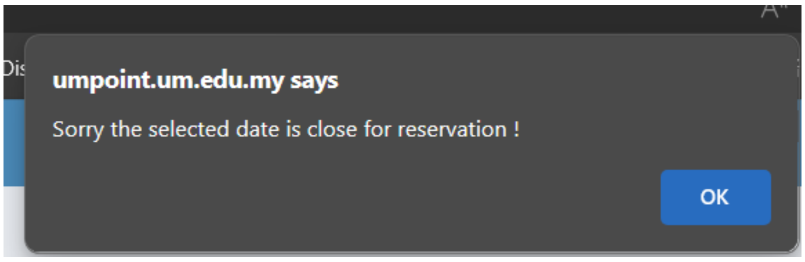
Please select start and end date and time.

☐ Yes, I agree with the IAS space's terms and conditions. Read Terms & Condition now.

	Monday 1 Apr, 2024	Tuesday 2 Apr, 2024	Wednesday 3 Apr, 2024	Thursday 4 Apr, 2024	Friday 5 Apr, 2024	Saturday 6 Apr, 2024	Sunday 7 Apr, 2024
09:00 am - 09:30 am							
09:30 am - 10:00 am							
10:00 am - 10:30 am							
10:30 am - 11:00 am							
11:00 am - 11:30 am							

Figure 2.17: Interface for checking availability of a venue

Though there are no signs on the calendar shown in the bottom part of Figure 2.17 that specific dates are unavailable for reservation, the system may hint the user through a pop-up message after clicking the orange continue button. Do refer to Figure 2.18 to get a look at the system hint.

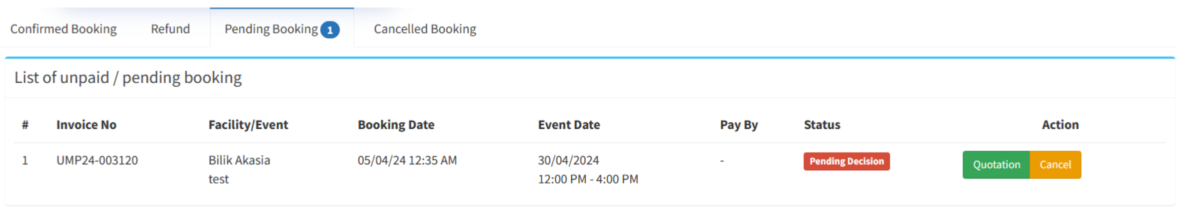


umpoint.um.edu.my says

Sorry the selected date is close for reservation !

Figure 2.18: Popup message of invalid reservation

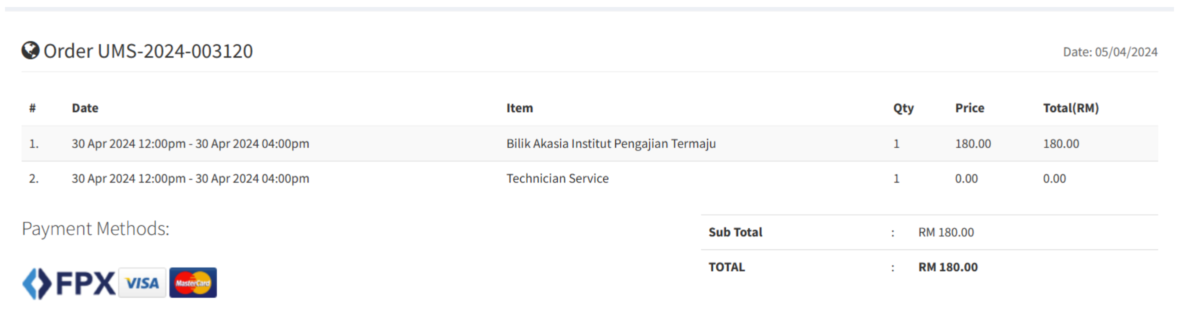
The booking procedures on the UM website are consistent and presented in a clear and logical manner: the user needs to undergo the payment process to verify the venue booked after selecting and confirming the desired venue to book. Upon confirming payment, the user can view the quotation of the booking request made. Refer to Figures 2.19 and 2.20 to get a better understanding of the quotation viewing after confirming the venue.



List of unpaid / pending booking

#	Invoice No	Facility/Event	Booking Date	Event Date	Pay By	Status	Action
1	UMP24-003120	Bilik Akasia test	05/04/24 12:35 AM	30/04/2024 12:00 PM - 4:00 PM	-	Pending Decision	Quotation Cancel

Figure 2.19: Dashboard interface after confirming venue to book



Order UMS-2024-003120 Date: 05/04/2024

#	Date	Item	Qty	Price	Total(RM)
1.	30 Apr 2024 12:00pm - 30 Apr 2024 04:00pm	Bilik Akasia Institut Pengajian Termaju	1	180.00	180.00
2.	30 Apr 2024 12:00pm - 30 Apr 2024 04:00pm	Technician Service	1	0.00	0.00

Payment Methods:

Sub Total	:	RM 180.00
TOTAL	:	RM 180.00

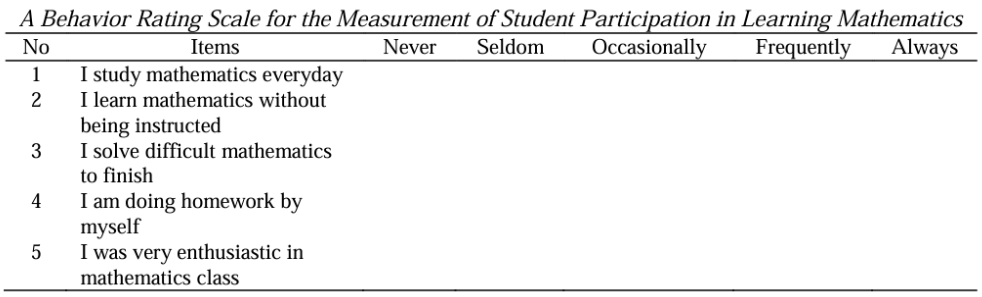
Figure 2.20: Interface for viewing quotation

To comment on the UM venue booking system, the website is well constructed and it is indeed a role model to refer to when it comes to the design phase of the system development.

2.5 System Usability Scale (SUS)

Aside from testing the system developed so that it functions well, a more practical approach to measure the product's success is to conduct a System Usability Test among the end-users. It pushes the project author to work harder to increase the non-functional requirements of the product other than focusing on the functional parts. This subsection describes how the System Usability Scale could aid in raising the competency of a software product.

Introduced by John Brooke (1995), this mechanism computes the satisfaction score from the end-users based on their usage experience on the product. Constructed based on the Likert scale which could be represented by Figure 2.21, it is a series of questions that all are required to be answered by the respondents (Brooke, 1995). The responses assert the respondent's perspective in giving consent or objection to the statements, managed in an ascending level from low to high.

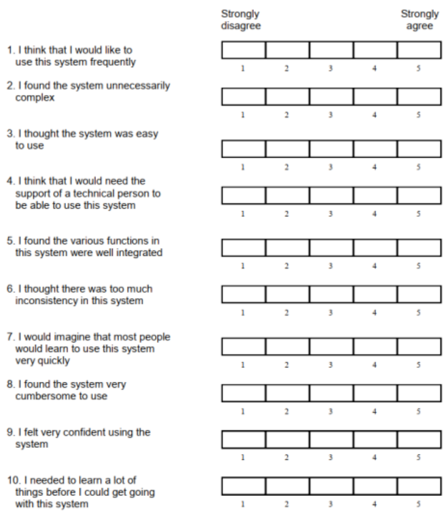


A Behavior Rating Scale for the Measurement of Student Participation in Learning Mathematics

No	Items	Never	Seldom	Occasionally	Frequently	Always
1	I study mathematics everyday					
2	I learn mathematics without being instructed					
3	I solve difficult mathematics to finish					
4	I am doing homework by myself					
5	I was very enthusiastic in mathematics class					

Figure 2.21: Example of Likert scale survey (Mumu et al., 2022)

To prepare the statements, the project author needs to select from 50 possible aspects, which were tested by 20 individuals who had experience using two software systems that both possess contrasting difficulty levels (Brooke, 1995). The aspects that displayed the most extreme responses will be included in the statements for conducting the SUS. According to Brooke (1995), these selected aspects show close intercorrelations, from 0.7 to 0.9. Another reason is that these aspects which had strong consent and objection could prevent the respondents from simply responding to the statements, which means they need to think wisely about giving their opinions.



	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 2.22: Example of System Usability Scale (SUS) (Brooke, 1995)

It was mentioned by Brooke (1995) that the SUS could only be carried out before any opinion sharing, and the respondents are urged to note down based on their instinct. To describe the computations, the odd-numbered statements need to minus the score position by 1 (n-1) while the even-numbered statements will need to do 5 minus the score position (5-n) (Brooke, 1995). After computing each statement's scores, the scores will need to be summed up, then undergo multiplication with 2.5 to get the System Usability Score.

The System Usability Score could be classified into grading categories to provide a persuasive guide to the public (Brooke, 1995). Based on the study conducted by Sauro and Lewis, they read the scores in the curved grading scale, classifying the gradings by determining the position of the SUS score (Lewis, 2018). For example, the median score which is equivalent to the SUS score range is rated C, graded average. Whereas scores which had entered the highest and lowest 15 percentile scores will be categorized into A and F boundaries correspondingly (Lewis, 2018). More specific grading will be awarded if the 15% of the SUS scores of a certain range are divided in detail, such as A+, A and A- (Lewis, 2018). The figure below provides a detailed guide on the grading of the SUS scores.

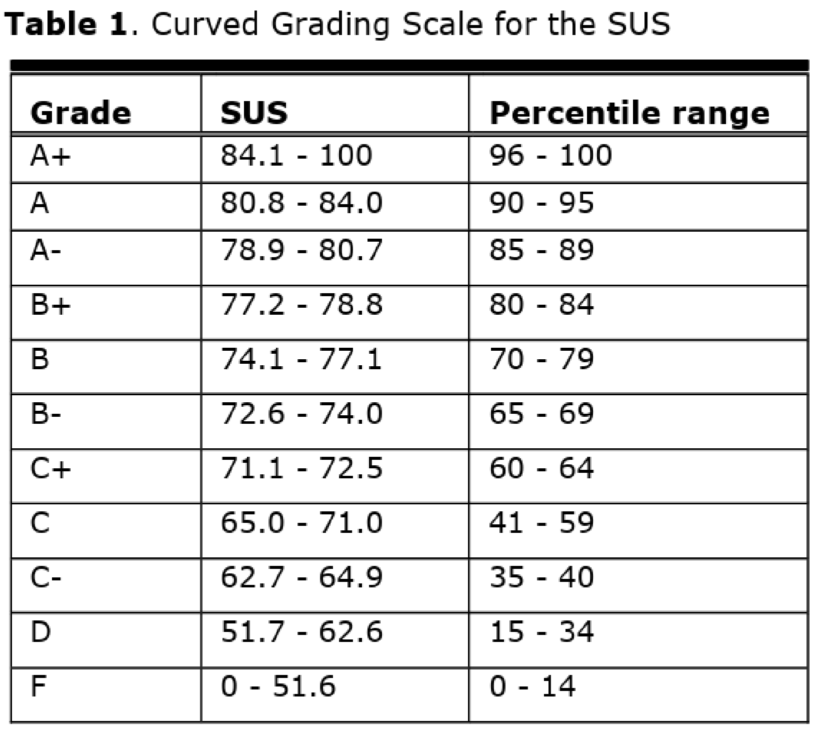


Table 1. Curved Grading Scale for the SUS

Grade	SUS	Percentile range
A+	84.1 - 100	96 - 100
A	80.8 - 84.0	90 - 95
A-	78.9 - 80.7	85 - 89
B+	77.2 - 78.8	80 - 84
B	74.1 - 77.1	70 - 79
B-	72.6 - 74.0	65 - 69
C+	71.1 - 72.5	60 - 64
C	65.0 - 71.0	41 - 59
C-	62.7 - 64.9	35 - 40
D	51.7 - 62.6	15 - 34
F	0 - 51.6	0 - 14

Figure 2.23: Grade for System Usability Scale (Lewis, 2018)

Assessing the public’s response on the product, the project author is able to identify the areas of improvement and to propose precautionary actions to tackle. Being a scientific yet measurable method, it ensures a moderate result to be obtained and helps the project author to study the customer behaviour and characteristics, provided if the respondent selected are widely dispersed. Meaning to say, the project author should be attentive in preparing the SUS statements and observe the respondents’ background, so that the results obtained are not biased.

2.6 Summary

This chapter has provided insight into the characteristics of software development approaches, databases planned for development, deployment platforms to be used and venue booking systems currently available in the market. A proper software development approach could help the project author streamline the project activities so that the deliverables could be delivered on time with the least overhead.

Regarding the database and deployment platforms selected for the Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal, this study aids the project author in deciding which tool to use directly when the project moves into the development stage. Appropriate tools chosen not only could eliminate the development burden but could speed up the development pace while elevating the efficiency of the system developed.

A total of three systems which perform the venue booking function are chosen for comparison, to identify the features or any details that should be a role model for the system to be developed. This comparison helped the project author to plan for the comprehensiveness of the system to be developed so that the functions integrated counter the problem statements. Good elements should be modified according to user context to cope with their requirements, whereas mediocre elements should be avoided during the development phase.

A thorough study on the System Usability Scale (SUS) gave the project author a concept to measure the user acceptance level. A questionnaire is needed so that a fruitful user experience could be gained after releasing the beta version for them to use.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

A software development project could be ensured with a smooth working flow, with pertinent software development methodology chosen and the software development details planned precisely. This chapter aims to give a detailed explanation on the phases of the chosen methodology and the development tools chosen by the project author. In terms of tasks identification and arrangement, this chapter also lists the activities in Work Breakdown Structure (WBS) and the timeframe for each activity in a Gantt chart.

3.2 Software Development Methodology

Phased methodology led the entire development project of the Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal. The project started from Planning, followed by Analyzing and Designing, Development phase, Testing phase, then finalized by the Closing phase. Since numerous revisions were carried out to give a clearer understanding of the project scope, iterations of the analysis, design and testing phases had took place with each iteration producing an improved version which was used on the succeeding iteration. Key features were prioritized so that the main problem statement had been solved in the early version. The project officially progressed to the Testing phase once the development iterations had been completed, which numerous types of tests needs to be conducted to determine the outcome’s success. After being tested, the project had been evaluated to point out the improvement areas, which was considered as moving to the Closing phase. The following figure summaries how the phased development methodology worked in this project.

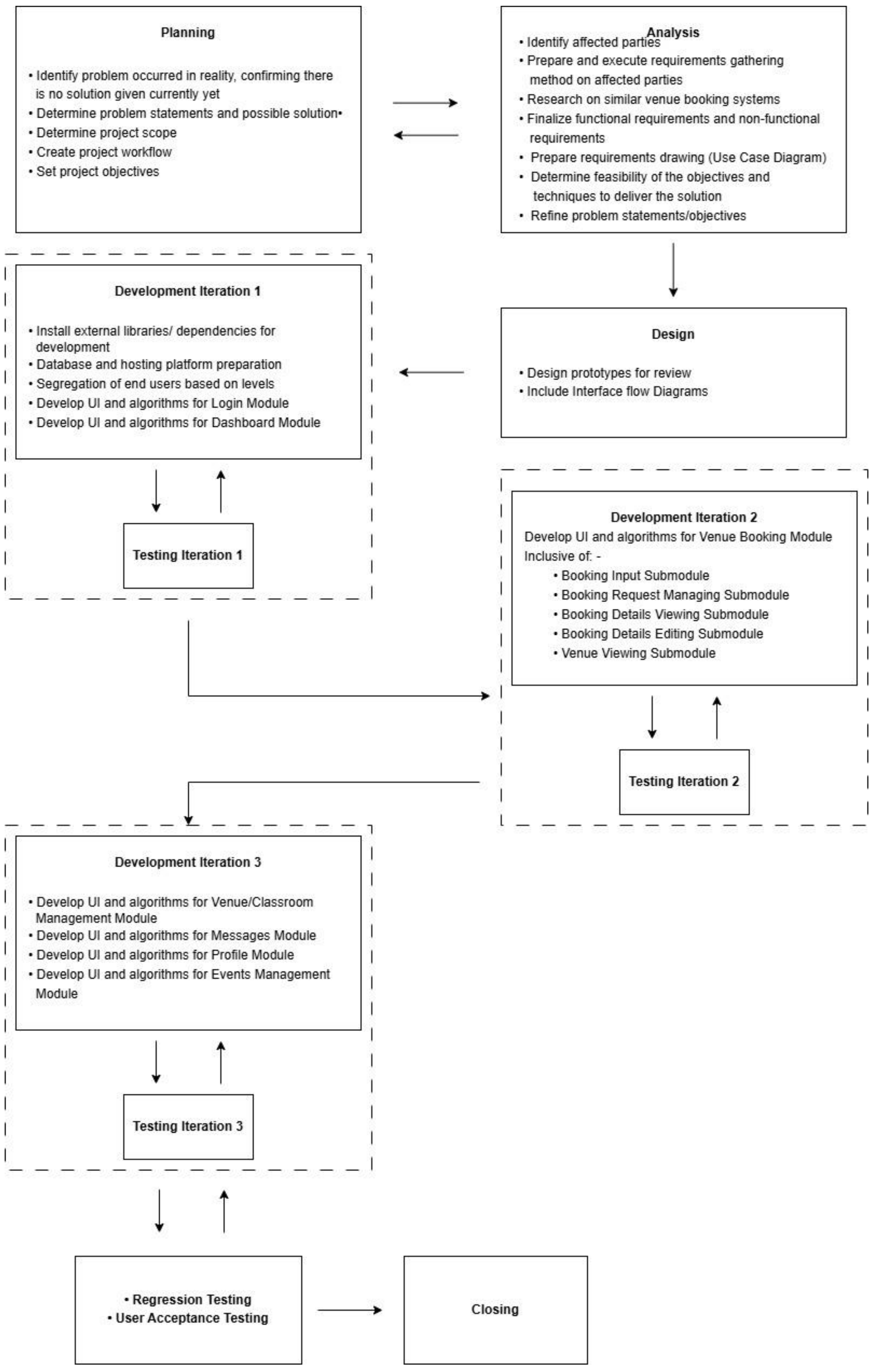

Planning
• Identify problem occurred in reality, confirming there is no solution given currently yet
• Determine problem statements and possible solution•
• Determine project scope
• Create project workflow
• Set project objectives
Analysis
• Identify affected parties
• Prepare and execute requirements gathering method on affected parties
• Research on similar venue booking systems
• Finalize functional requirements and non-functional requirements
• Prepare requirements drawing (Use Case Diagram)
• Determine feasibility of the objectives and techniques to deliver the solution
• Refine problem statements/objectives
Design
• Design prototypes for review
• Include Interface flow Diagrams
Development Iteration 1
• Install external libraries/ dependencies for development
• Database and hosting platform preparation
• Segregation of end users based on levels
• Develop UI and algorithms for Login Module
• Develop UI and algorithms for Dashboard Module
Testing Iteration 1
Development Iteration 2
Develop UI and algorithms for Venue Booking Module
Inclusive of: -
• Booking Input Submodule
• Booking Request Managing Submodule
• Booking Details Viewing Submodule
• Booking Details Editing Submodule
• Venue Viewing Submodule
Testing Iteration 2
Development Iteration 3
• Develop UI and algorithms for Venue/Classroom Management Module
• Develop UI and algorithms for Messages Module
• Develop UI and algorithms for Profile Module
• Develop UI and algorithms for Events Management Module
Testing Iteration 3
• Regression Testing
• User Acceptance Testing
Closing

Figure 3.1: Phased Development Methodology Workflow for the Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal

3.2.1 Planning

Initiating the system development project, this phase marked a kickstart by having the project objectives, problem statements determined, since these aspects had motivated the formulation of the development project. Project scopes were clarified and project activities were scheduled in this phase as well.

3.2.1.1 Project Proposal

The project proposal served as the key document in starting a project, whose aim is to provide insight to the stakeholders on the theme of the project. Firstly, the project author was aware of emerged obstacles that are feasible to solve within the knowledge. Informal questioning techniques or observation were used to gain a basic understanding of the problem boundaries. After confirming the hypothesis had truly caused inconveniences to certain parties, the project author set project objectives that meet the SMART criteria. The objectives had shaped the project scope to prevent the project study overloaded with unnecessary information. The right software development methodology and requirements elicitation techniques were confirmed in this stage so that the project author could logically outline the activities and conduct a professional study on the problems respectively.

3.2.1.2 Requirements Gathering and Elicitation

The software development project started by having the requirements gathered from the affected parties. Since the parties affected range from students, lecturers, HOD and DGS staff, the requirements were solicited in a different manner, which will be discussed in detail in the next subchapter. A comparison study on similar systems was conducted at the same time, to observe which aspects could be done better and which was noteworthy to be used as a reference. At the end of this phase, the responses and observations collected were finalized as Functional and Non-functional requirements, as an insight for building the system.

3.2.1.2.1 Survey

This technique was specifically designed for lecturers since they are using the venue/classroom mechanism routinely for additional classes or internal meetings. The survey consisted of 8 questions that was being hosted by Google Documents and being publicized to all lecturers from UTAR Sungai Long campus through emailing. To keep the lecturers engaged in answering the survey, the questions were designed in a hybrid of subjective and objective questions. This tactic was to inspire the lecturers to think more based on their experiences of using the existing booking system. The survey was aimed to get a minimum count of 30 responses due to time constraint. However, overtimed responses were welcomed as well, as a source of insight to confirm the challenges faced while using the existing system.

3.2.1.2.2 Interview

A few students are co-curricular society executives which frequently performed the venue/classroom booking process, these students were invited to gather requirements. To get different perspectives, two societies were chosen from different co-curricular society types (Course-based, General interest, Performing arts & creative and Self-defense) so that every party could voice their feedback. The interview questions were fixed in 6 questions in total, but the order was conducted flexibly, depending on their response given. Asking in a fixed template could limit the responses received.

The administrative staff that handles the venue booking matters was from the Department of General Services (DGS). The interview consisted of 7 fixed questions and was conducted flexibly too. The interviewing questions were mostly the same as the ones in the survey, which was conducted among the lecturers, due to both groups of users being the official end-users for the current system. As surveys could gain limited information and the person in-charge is not in a large population, so the interview method was the best for gaining administrative opinions on the venue/classroom booking matter.

The interview was an unstructured one to encourage the respondents to give actual responses. Based on the responses and interview content, the end-users' expectations were formalized and were deeply considered in the development of the system.

3.2.1.3 Project Scheduling

Project Scheduling specifies the activities planned, recorded as Work Breakdown Structure (WBS). The WBS categorized the activities to be done in each phase and each iteration. The modules to be included in the system had their development sequence listed.

As WBS only specified the activities, the Gantt chart was used to list the timeframe for each phase. Technically, the project author had input the estimated start date and end date for all activities, along with work duration computed by the difference of the dates mentioned. This served as an overview for the stakeholders to know the scale and the workload of the project. Other than that, it motivated the project author to follow the timeline in adherence to deliver the outcomes on time.

3.2.2 Analysis

After scheduling, the software development phase had entered the analysis stage. Since the requirements elicitation techniques were selected, then the project author had utilized the techniques on the targeted parties respectively, like to interview the administrative staff from DGS and a few students while carrying out a one-week online survey with the UTAR lecturers. Concurrently, the project author had searched for similar systems, mostly those systems that were also being used in the university context to match the project scope. A trial had been carried out on these systems. Through the trial experience on these systems and observations, this had provided the project author opportunity for construction and thoughts that possible additional features to be included. Possibly, the project author had revised the feasibility of the objectives set and the requirements gathered, confirming that both opinions were not conflicting. Reversal to the preceding phase for refinement occurred at times.

At the end of this stage, the project author had analyzed the results and possessed a full understanding of what should be included in the system to be developed. Practically, the project author had drafted a Use Case Diagram which defined the relationship between the system's features and the end-users. Use Case Descriptions were listed to describe the functional flows of the modules. At the same time, a prototype which consisted of the possible layouts was designed with Axure RP9, so that it was presented to the stakeholders for a preview.

3.2.2.1 Review of Similar Systems

The project author had selected 3 Malaysian universities' venue/classroom booking systems to identify their strengths and weaknesses, as an exemplar for the construction of this project. The features supported by these three systems were delineated and concluded in Chapter 2. The following lists the properties that were considered in this project, in no particular order:

1. Visitor Management
    - Authentication of user login
    - User Profile viewing
    - User Profile editing
    - Different functionalities for users based on user status
2. Booking Management
    - Submit booking request
    - View booking requests
    - Approve/Reject booking requests (administrative staff only)
    - Review past booking details
    - Notification through email and SMS after any updates on the booking
3. Messages
    - View all messages
    - Chat with selected person
4. Venue/Classroom Management
    - View all venues/classrooms

- Edit status and information for all venues/classrooms (administrative staff only)

5. Calendar Event Management
    - View all events on calendar
    - Arrange events on calendar (administrative staff only)

3.2.3 Design, Development & Testing

The project evolved to the design phase gradually after finalising the requirements and the objectives. As there were many modules performed by this system, therefore the development work was decomposed into four iterations which each delivered a vital functionality. To explain briefly, the first iteration had got the system ready for development through environment setup and the login and dashboard modules designed. The second iteration mainly focused on the venue booking module, where all submodules used for booking venues were constructed and tested individually. In the third iteration, user profile module, messages module, venue/classroom and events management module were constructed, which was a mix of normal users and administrative users' functionalities. The last iteration was the testing, to examine if all modules mentioned could work together as a complete venue booking system.

3.2.3.1 First Phase

Marked as the start of the development, the computing environment had to be prepared for the development work. Necessary software tools like Visual Studio 2022 and XAMPP were installed. Port configurations for database connection and hosting were setup as there were porting conflicts arose in the internal computing system. Useful extensions were included as well, to speed up the development progress.

To gain confidence, the project author started by developing those smaller modules first. The login module and the user authentication function were focused, as both served as the base for using any web-based system. Different information was set on a unified dashboard page since different levels of users had different information to view. Validation errors were included so that unauthorized users were aware of them.

Testing on the login and dashboard functions was carried out once they had been constructed. Amendment of source codes or referring to the Use Case Descriptions often happened since the project author needs to ensure they match the functionality flows.

3.2.3.2 Second Phase

After the user access and dashboard modules had been constructed, the development evolved to the main function which is the venue booking modules. This module includes of booking input submodule (for the end-users to submit a booking request), booking request managing submodule (to reject/approve/reject), booking details viewing submodule (either view in PDF or in the webpage column form), booking detail editing submodule (for administrative users to add booking notes) and venue viewing submodule (to display mapping for all venues/classrooms in both KA and KB buildings). External libraries like FPDF, Bulk360 SMS API and PHPMailer were utilized here as some submodules were featured with notification functions. Information displaying must be accurate at this stage and the procedures must obey the venue booking procedural flowcharts provided by the UTAR communities. Like the previous phase, all submodules were tested before moving to the next development iteration.

3.2.3.3 Third Phase

After the development of venue booking modules and minor parts of the venue/classroom viewing, the software development continued to the venue/classroom management module (handled by administrative staff, to update the availability status of each venue manually), messages module (to transmit messages bidirectionally and to retrieve all chatting records), profile module (to view personal details and editing) and the events management module (for the administrative users to edit the planned events in the calendar and the non-administrative users to view). Functional testing took place for these submodules mentioned, before moving to the final iteration for user acceptance testing.

3.2.3.4 Fourth Phase

After all modules had been constructed and passed all individual functional tests, the modules were tested comprehensively to examine the full functioning of the Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal. The project author had set testing environments based on mocking real-world scenarios so that the system could handle unexpected events. Bugs or hidden defects were found and the project author had solved it. Integration points between modules were examined as these parts are prone to have errors hidden in it. Lastly, regression testing was carried out so that the source code changes did not affect other well-functioned modules.

3.2.4 Closing

In this phase, end-users had tried on the developed system, which provided feedback that aids in conducting User Acceptance Test (UAT). Based on the feedback, the project author had concluded the improvement areas and prepared for the report writing. The report includes the system architecture, the working flows of the modules, the snapshots of the final system and the testing details. Finally, the project author had created the presentation slides to present the system developed and its development process.

3.3 Tools Used

If a workman wishes to do a good job, he must first sharpen his tools. Development tools are essential for software development projects, as they produce the software product. In this section, development tools used in this Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal will be discussed.

3.3.1 Visual Studio Code

Supporting the development phase, this software served as the source code IDE (Integrated Development Environment), enabling the project author to organize files and external libraries, along with configuring the environment needed to write source codes for the project. This code editor supports various extensions to be used for development, which is a useful tool that accelerates the development process. Copilot and Auto Close Tag were the few tools that the project author favoured due to their ability to automate many manual tasks, like to give code suggestions and self-declare the ending tag for programming languages which use tags, respectively. This software constantly updates so that it could provide more improved features to the developers.

3.3.2 XAMPP Server

This software is responsible for hosting websites in their own device environment, as preliminary hosting is required to examine the functionality of every module written in a web development project. Another point is that the server was integrated with the PhpMyAdmin database, which was used as an active communication tool between the application developed and the database, directly.

3.3.3 Axure RP 9

This software played the main role in developing the prototype. As the prototype was reviewed by the stakeholders to give feedback, this software enabled the project author to simulate a high-fidelity prototype of the web application, in which the display elements and functions performed are mostly the same. Implementing in a drag-and-drop style, this tool however cannot be a final product development tool due to its limited functionality and did not support the interchange of information with the external parties like what actual web application does.

3.3.4 ProjectLibre

The project author used this software to plan for tasks for the entire project. The field included project planning, analysis, design and closing. Specifying these main fields, the activities need to be further broken down to elicit the tasks to be performed to form the project’s component gradually. The breakdowns were achieved through indentation in this software. Besides that, the project author had assigned the dates for each task, controlling the project timeline.

3.4 Work Breakdown Structure (WBS)

1.0 Planning

- 1.1 Study problem background
- 1.2 Define Problem Statements
- 1.3 Define Project Objectives
- 1.4 Define Project Solutions
    - 1.4.1 Propose Initial Solutions
    - 1.4.2 Analyse Feasibility of the Solutions
- 1.5 Define Project Approach
    - 1.5.1 Determine Research Approach
    - 1.5.2 Determine System Development Approach
- 1.6 Define Project Scope
    - 1.6.1 Identify System Scope
    - 1.6.2 Identify Targeted End-users
    - 1.6.3 Identify Project Modules
- 1.7 Project Scheduling
    - 1.7.1 Establish Work Breakdown Structure
        - 1.7.1.1 Define Main Tasks
        - 1.7.1.2 Breakdown of Main Tasks into Smaller Tasks
    - 1.7.2 Establish Gantt Chart
        - 1.7.2.1 Identify Task Dependency
        - 1.7.2.2 Estimate Task Duration
        - 1.7.2.3 Draft Gantt Chart
        - 1.7.2.4 Review Gantt Chart
        - 1.7.2.5 Finalize Gantt Chart

2.0 Analysis and Design

- 2.1 Requirements Gathering
    - 2.1.1 Conduct Interview
        - 2.1.1.1 Identify Interview Respondents
        - 2.1.1.2 Design Interview Questions
        - 2.1.1.3 Review Interview Questions
        - 2.1.1.4 Hold Interview
    - 2.1.2 Conduct Survey
        - 2.1.2.1 Identify Survey Respondents

- 2.1.2.2 Design Survey Questions
- 2.1.2.3 Review Survey Questions
- 2.1.2.4 Distribute Survey
- 2.1.3 Literature Review
  - 2.1.3.1 Identify Research Areas
  - 2.1.3.2 Review on Development Aspects
    - 2.1.3.2.1 Review of Development Approaches
    - 2.1.3.2.2 Review of Deployment Platforms
    - 2.1.3.2.3 Review of Development Languages
    - 2.1.3.2.4 Review of Database Models
    - 2.1.3.2.5 Review of Deployment Platforms
  - 2.1.3.3 Review of Similar Systems
    - 2.1.3.3.1 Review of UPM
    - 2.1.3.3.2 Review of UiTM
    - 2.1.3.3.3 Review of UM
- 2.2 Analyse Interview Data
- 2.3 Analyse Survey Responses
- 2.4 Requirement Elicitation
  - 2.4.1 Filter Most Recommended Features
  - 2.4.2 Define Functional and Non-functional Requirements
  - 2.4.3 Review Functional and Non-functional Requirements
  - 2.4.4 Refine Functional and Non-functional Requirements
- 2.5 Design Use Case Diagram
- 2.6 Generate Use Case Descriptions
- 2.7 Design Entity Relation Diagram
- 2.8 Design Prototypes
- 2.9 Design Interface Flow Diagrams
- 3.0 Check for feedbacks concluded from Project I
- 4.0 Amend report based on previous submission
- 5.0 First Iteration for design, develop and testing
  - 5.1 Setup Deployment Platform
    - 5.1.1 Download XAMPP Web Server
    - 5.1.2 Install
    - 5.1.3 Test run on browser

5.2 Setup database
- 5.2.1 Configure database route
- 5.2.2 Testing
  - 5.2.2.1 Create dummy data tables
  - 5.2.2.2 Test run on browser

5.3 Setup External Libraries
- 5.3.1 Download Bootstrap library
- 5.3.2 Declare Bootstrap library in project repository
- 5.3.3 Install PHPMailer
- 5.3.4 Environment configuration for PHPMailer in project repository
- 5.3.5 Install Bulk360 package
- 5.3.6 Declare Bulk360 configurations in project repository
- 5.3.7 Install FPDF package
- 5.3.8 Set environment configuration of FPDF in project repository

5.4 Create Login Module
- 5.4.1 Design Login Page User Interface
- 5.4.2 Developing authentication logics for login process
  - 5.4.2.1 Valid Login
  - 5.4.2.2 Invalid Login
- 5.4.3 Test Login Module
  - 5.4.3.1 Test for Valid Login
  - 5.4.3.2 Test for Invalid Login
- 5.4.4 Design Forget Password Module Interface
  - 5.4.4.1 Interface of Forget Password
  - 5.4.4.2 Interface of Reset Password
- 5.4.5 Developing logics for resetting password
  - 5.4.5.1 Forget password
  - 5.4.5.2 Reset password
- 5.4.6 Test Forget Password module
  - 5.4.6.1 Forget password
  - 5.4.6.2 Reset password

5.5 Create Dashboard Module

5.5.1 Dashboard Module for Administrative users

5.5.1.1 Design Dashboard User Interface

5.5.1.1.1 Sidebar design

5.5.1.1.2 Main Layout design

5.5.1.2 Define data to be displayed

5.5.2 Dashboard Module for Non-administrative users

5.5.2.1 Design Dashboard User Interface

5.5.2.1.1 Sidebar design

5.5.2.1.2 Main Layout design

5.5.2.2 Define data to be displayed

5.5.3 Testing

5.5.3.1 Test for Administrative users Dashboard

5.5.3.2 Test for Non-administrative users Dashboard

6.0 Second iteration for design, develop and testing

6.1 Create Venue Booking Module

6.1.1 Booking Input Submodule

6.1.1.1 Design User Interface

6.1.1.2 Develop working logics for getting user input

6.1.2 Testing for Venue Input Module

6.1.2.1 Test for valid user inputs

6.1.2.2 Test for invalid user inputs

6.1.3 Request Managing Submodule

6.1.3.1 Design User Interface

6.1.3.2 Develop working logics for approving pending booking requests

6.1.3.3 Develop working logics for rejecting pending booking requests

6.1.3.4 Develop working logics for starting conversation

6.1.4 Testing for Request Managing Submodule

6.1.4.1 Testing working logics for approving pending booking requests

6.1.4.2 Testing working logics for rejecting pending booking requests

6.1.4.3 Testing working logics for starting conversation

- 6.1.5 Booking Details Viewing Submodule
    - 6.1.5.1 Design User Interface
    - 6.1.5.2 Developing working logics for showing booking details
        - 6.1.5.2.1 Develop viewing in PDF function
        - 6.1.5.2.2 Develop view all details page
            - 6.1.5.2.2.1 Design User Interface
            - 6.1.5.2.2.2 Define data to be fetched
- 6.1.6 Testing for Booking Details Viewing Submodule
    - 6.1.6.1 Test working logics for showing booking details
        - 6.1.6.1.1 Test view in PDF function
        - 6.1.6.1.2 Test All Details page
- 6.1.7 Booking Details Editing Submodule
    - 6.1.7.1 Design User Interface
    - 6.1.7.2 Developing working logics for editing booking details
- 6.1.8 Testing for Booking Details Editing Submodule
    - 6.1.8.1 Test data correctness fetched in page
    - 6.1.8.2 Test data editing functionality
- 6.1.9 Venue Viewing Submodule
    - 6.1.9.1 Design User Interface
    - 6.1.9.2 Develop working logics for venue viewing
        - 6.1.9.2.1 Correctness of venue's availability status fetched from database
        - 6.1.9.2.2 Retrieval of all venue's name and facilities details
- 6.1.10 Testing Venue Viewing Submodule
    - 6.1.10.1 Testing for venue's availability status retrieval
    - 6.1.10.2 Testing for venue's name and facilities details retrieval

7.0 Third iteration for design,develop and testing

- 7.1 Venue/Classroom Management Module
    - 7.1.1 Design User Interface
    - 7.1.2 Develop working logics for venues retrieval

- 7.1.3 Develop working logics for updating venue availability status
  - 7.1.3.1 Update to available
  - 7.1.3.2 Update to not available
- 7.2 Testing for Venue/Classroom Management Module
  - 7.2.1 Testing for updating venue availability status
    - 7.2.1.1 Update to available
    - 7.2.1.2 Update to not available
- 7.3 Messages Module
  - 7.3.1 Design User Interface for chatting
  - 7.3.2 Develop working logics for messages transmission
- 7.4 Testing for Messages Module
  - 7.4.1 Testing for correctness of messages retrieved
  - 7.4.2 Testing for transmitting messages
    - 7.4.2.1 Transmit from Administrator users side
    - 7.4.2.2 Transmit from Non-administrator users side
- 7.5 Profile Module
  - 7.5.1 Design Profile page User Interface
  - 7.5.2 Develop working logics for Profile Module
    - 7.5.2.1 Profile information retrieval
    - 7.5.2.2 Profile information editing
- 7.6 Testing for Profile Module
  - 7.6.1 Testing for correctness of profile information retrieved
  - 7.6.2 Testing for profile information editing function
- 7.7 Events Management Module
  - 7.7.1 Design User Interface
  - 7.7.2 Develop working logics for Events Management Module
    - 7.7.2.1 Add events
    - 7.7.2.2 Remove events
- 7.8 Testing for Events Management Module
  - 7.8.1 Testing for correctness of events information shown on calendar
  - 7.8.2 Testing for add events function

7.8.3 Testing for remove events function

8.0 Fourth iteration for design, develop and testing

8.1 Trimester Management Module

8.1.1 Add New Trimester

8.1.2 View Existing/Added Trimester

8.1.3 Edit Existing/Added Trimester Information

8.1.4 Delete Trimester

8.2 Testing for Trimester Management Module

8.2.1 Testing for add trimester operations

8.2.2 Testing for retrieval of added trimester

8.2.3 Testing for editing trimester information

8.2.4 Testing for deletion of selected trimester

8.3 Class Management Module

8.3.1 Add Class

8.3.2 View Class

8.3.3 Edit Class Information

8.3.4 Delete Class

8.4 Testing for Class Management Module

8.4.1 Testing for add class operations

8.4.3 Testing for retrieval of added class

8.4.3 Testing for editing class information

8.4.4 Testing for deletion of selected class information

8.5 Unit testing

8.6 Regression testing

8.7 User Acceptance Test

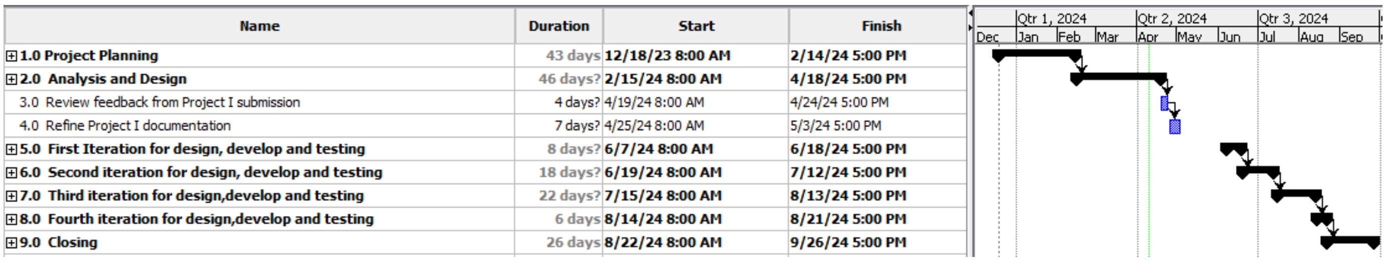
9.0 Closing

9.1 Create project documentation

9.2 Finalize project documentation

3.5 Gantt Chart

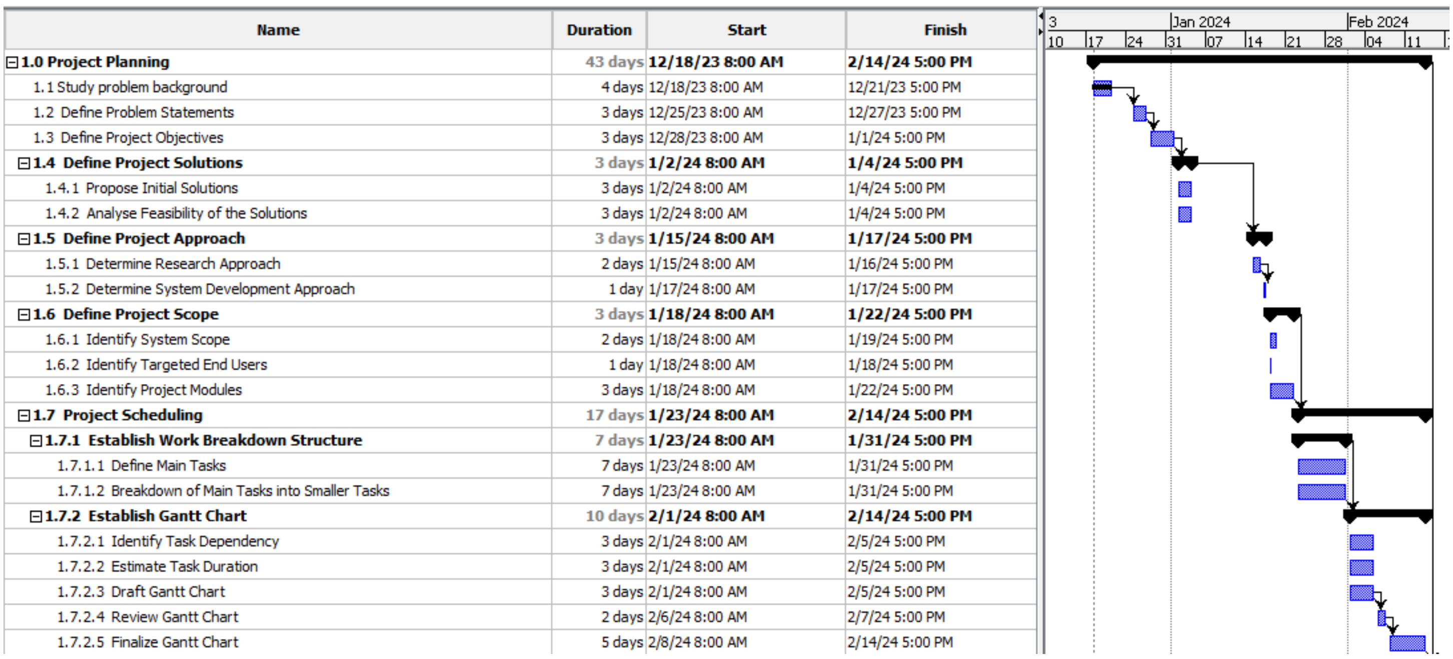
3.5.1 Overview of the Project Timeline



Name	Duration	Start	Finish
⊞ **1.0 Project Planning**	43 days	**12/18/23 8:00 AM**	**2/14/24 5:00 PM**
⊞ **2.0 Analysis and Design**	46 days?	**2/15/24 8:00 AM**	**4/18/24 5:00 PM**
3.0 Review feedback from Project I submission	4 days?	4/19/24 8:00 AM	4/24/24 5:00 PM
4.0 Refine Project I documentation	7 days?	4/25/24 8:00 AM	5/3/24 5:00 PM
⊞ **5.0 First Iteration for design, develop and testing**	8 days?	**6/7/24 8:00 AM**	**6/18/24 5:00 PM**
⊞ **6.0 Second iteration for design, develop and testing**	18 days?	**6/19/24 8:00 AM**	**7/12/24 5:00 PM**
⊞ **7.0 Third iteration for design,develop and testing**	22 days?	**7/15/24 8:00 AM**	**8/13/24 5:00 PM**
⊞ **8.0 Fourth iteration for design,develop and testing**	6 days	**8/14/24 8:00 AM**	**8/21/24 5:00 PM**
⊞ **9.0 Closing**	26 days	**8/22/24 8:00 AM**	**9/26/24 5:00 PM**

Figure 3.2: Gantt Chart overview for the Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal

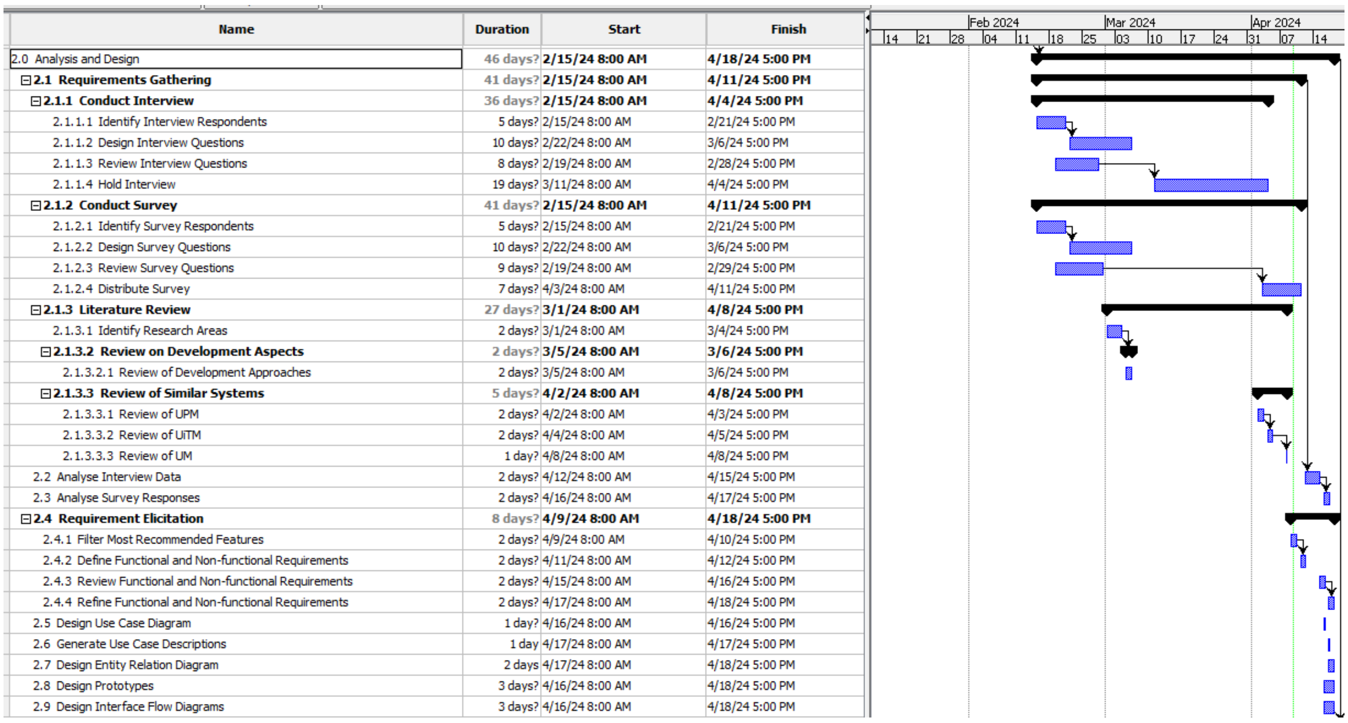
3.5.2 Planning Phase Timeline



Name	Duration	Start	Finish
⊟ **1.0 Project Planning**	43 days	**12/18/23 8:00 AM**	**2/14/24 5:00 PM**
1.1 Study problem background	4 days	12/18/23 8:00 AM	12/21/23 5:00 PM
1.2 Define Problem Statements	3 days	12/25/23 8:00 AM	12/27/23 5:00 PM
1.3 Define Project Objectives	3 days	12/28/23 8:00 AM	1/1/24 5:00 PM
⊟ **1.4 Define Project Solutions**	3 days	**1/2/24 8:00 AM**	**1/4/24 5:00 PM**
1.4.1 Propose Initial Solutions	3 days	1/2/24 8:00 AM	1/4/24 5:00 PM
1.4.2 Analyse Feasibility of the Solutions	3 days	1/2/24 8:00 AM	1/4/24 5:00 PM
⊟ **1.5 Define Project Approach**	3 days	**1/15/24 8:00 AM**	**1/17/24 5:00 PM**
1.5.1 Determine Research Approach	2 days	1/15/24 8:00 AM	1/16/24 5:00 PM
1.5.2 Determine System Development Approach	1 day	1/17/24 8:00 AM	1/17/24 5:00 PM
⊟ **1.6 Define Project Scope**	3 days	**1/18/24 8:00 AM**	**1/22/24 5:00 PM**
1.6.1 Identify System Scope	2 days	1/18/24 8:00 AM	1/19/24 5:00 PM
1.6.2 Identify Targeted End Users	1 day	1/18/24 8:00 AM	1/18/24 5:00 PM
1.6.3 Identify Project Modules	3 days	1/18/24 8:00 AM	1/22/24 5:00 PM
⊟ **1.7 Project Scheduling**	17 days	**1/23/24 8:00 AM**	**2/14/24 5:00 PM**
⊟ **1.7.1 Establish Work Breakdown Structure**	7 days	**1/23/24 8:00 AM**	**1/31/24 5:00 PM**
1.7.1.1 Define Main Tasks	7 days	1/23/24 8:00 AM	1/31/24 5:00 PM
1.7.1.2 Breakdown of Main Tasks into Smaller Tasks	7 days	1/23/24 8:00 AM	1/31/24 5:00 PM
⊟ **1.7.2 Establish Gantt Chart**	10 days	**2/1/24 8:00 AM**	**2/14/24 5:00 PM**
1.7.2.1 Identify Task Dependency	3 days	2/1/24 8:00 AM	2/5/24 5:00 PM
1.7.2.2 Estimate Task Duration	3 days	2/1/24 8:00 AM	2/5/24 5:00 PM
1.7.2.3 Draft Gantt Chart	3 days	2/1/24 8:00 AM	2/5/24 5:00 PM
1.7.2.4 Review Gantt Chart	2 days	2/6/24 8:00 AM	2/7/24 5:00 PM
1.7.2.5 Finalize Gantt Chart	5 days	2/8/24 8:00 AM	2/14/24 5:00 PM

Figure 3.3: Planning phase Gantt Chart overview for the project

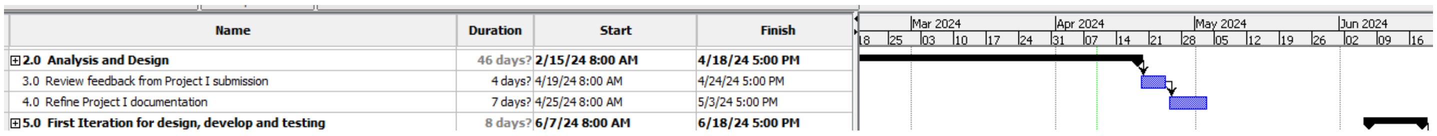
3.5.3 Analysis and Design Timeline



Name	Duration	Start	Finish
2.0 Analysis and Design	46 days?	2/15/24 8:00 AM	4/18/24 5:00 PM
⊟2.1 Requirements Gathering	41 days?	2/15/24 8:00 AM	4/11/24 5:00 PM
⊟2.1.1 Conduct Interview	36 days?	2/15/24 8:00 AM	4/4/24 5:00 PM
2.1.1.1 Identify Interview Respondents	5 days?	2/15/24 8:00 AM	2/21/24 5:00 PM
2.1.1.2 Design Interview Questions	10 days?	2/22/24 8:00 AM	3/6/24 5:00 PM
2.1.1.3 Review Interview Questions	8 days?	2/19/24 8:00 AM	2/28/24 5:00 PM
2.1.1.4 Hold Interview	19 days?	3/11/24 8:00 AM	4/4/24 5:00 PM
⊟2.1.2 Conduct Survey	41 days?	2/15/24 8:00 AM	4/11/24 5:00 PM
2.1.2.1 Identify Survey Respondents	5 days?	2/15/24 8:00 AM	2/21/24 5:00 PM
2.1.2.2 Design Survey Questions	10 days?	2/22/24 8:00 AM	3/6/24 5:00 PM
2.1.2.3 Review Survey Questions	9 days?	2/19/24 8:00 AM	2/29/24 5:00 PM
2.1.2.4 Distribute Survey	7 days?	4/3/24 8:00 AM	4/11/24 5:00 PM
⊟2.1.3 Literature Review	27 days?	3/1/24 8:00 AM	4/8/24 5:00 PM
2.1.3.1 Identify Research Areas	2 days?	3/1/24 8:00 AM	3/4/24 5:00 PM
⊟2.1.3.2 Review on Development Aspects	2 days?	3/5/24 8:00 AM	3/6/24 5:00 PM
2.1.3.2.1 Review of Development Approaches	2 days?	3/5/24 8:00 AM	3/6/24 5:00 PM
⊟2.1.3.3 Review of Similar Systems	5 days?	4/2/24 8:00 AM	4/8/24 5:00 PM
2.1.3.3.1 Review of UPM	2 days?	4/2/24 8:00 AM	4/3/24 5:00 PM
2.1.3.3.2 Review of UiTM	2 days?	4/4/24 8:00 AM	4/5/24 5:00 PM
2.1.3.3.3 Review of UM	1 day?	4/8/24 8:00 AM	4/8/24 5:00 PM
2.2 Analyse Interview Data	2 days?	4/12/24 8:00 AM	4/15/24 5:00 PM
2.3 Analyse Survey Responses	2 days?	4/16/24 8:00 AM	4/17/24 5:00 PM
⊟2.4 Requirement Elicitation	8 days?	4/9/24 8:00 AM	4/18/24 5:00 PM
2.4.1 Filter Most Recommended Features	2 days?	4/9/24 8:00 AM	4/10/24 5:00 PM
2.4.2 Define Functional and Non-functional Requirements	2 days?	4/11/24 8:00 AM	4/12/24 5:00 PM
2.4.3 Review Functional and Non-functional Requirements	2 days?	4/15/24 8:00 AM	4/16/24 5:00 PM
2.4.4 Refine Functional and Non-functional Requirements	2 days?	4/17/24 8:00 AM	4/18/24 5:00 PM
2.5 Design Use Case Diagram	1 day?	4/16/24 8:00 AM	4/16/24 5:00 PM
2.6 Generate Use Case Descriptions	1 day	4/17/24 8:00 AM	4/17/24 5:00 PM
2.7 Design Entity Relation Diagram	2 days	4/17/24 8:00 AM	4/18/24 5:00 PM
2.8 Design Prototypes	3 days?	4/16/24 8:00 AM	4/18/24 5:00 PM
2.9 Design Interface Flow Diagrams	3 days?	4/16/24 8:00 AM	4/18/24 5:00 PM

Figure 3.4: Analysis and Design phase Gantt Chart overview

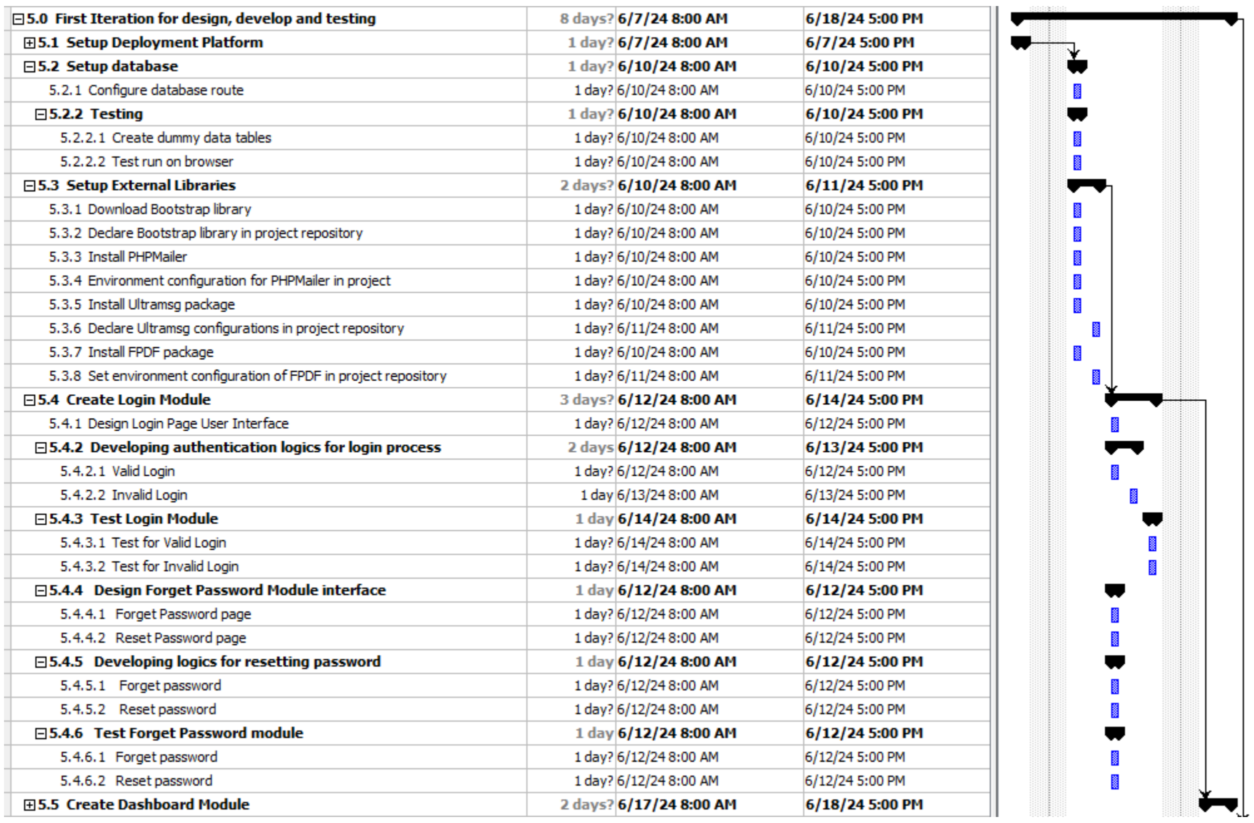
3.5.4 Transition from Project I to Project II



Name	Duration	Start	Finish
⊞2.0 Analysis and Design	46 days?	2/15/24 8:00 AM	4/18/24 5:00 PM
3.0 Review feedback from Project I submission	4 days?	4/19/24 8:00 AM	4/24/24 5:00 PM
4.0 Refine Project I documentation	7 days?	4/25/24 8:00 AM	5/3/24 5:00 PM
⊞5.0 First Iteration for design, develop and testing	8 days?	6/7/24 8:00 AM	6/18/24 5:00 PM

Figure 3.5: Transition from Project I to Project II after Project I submission

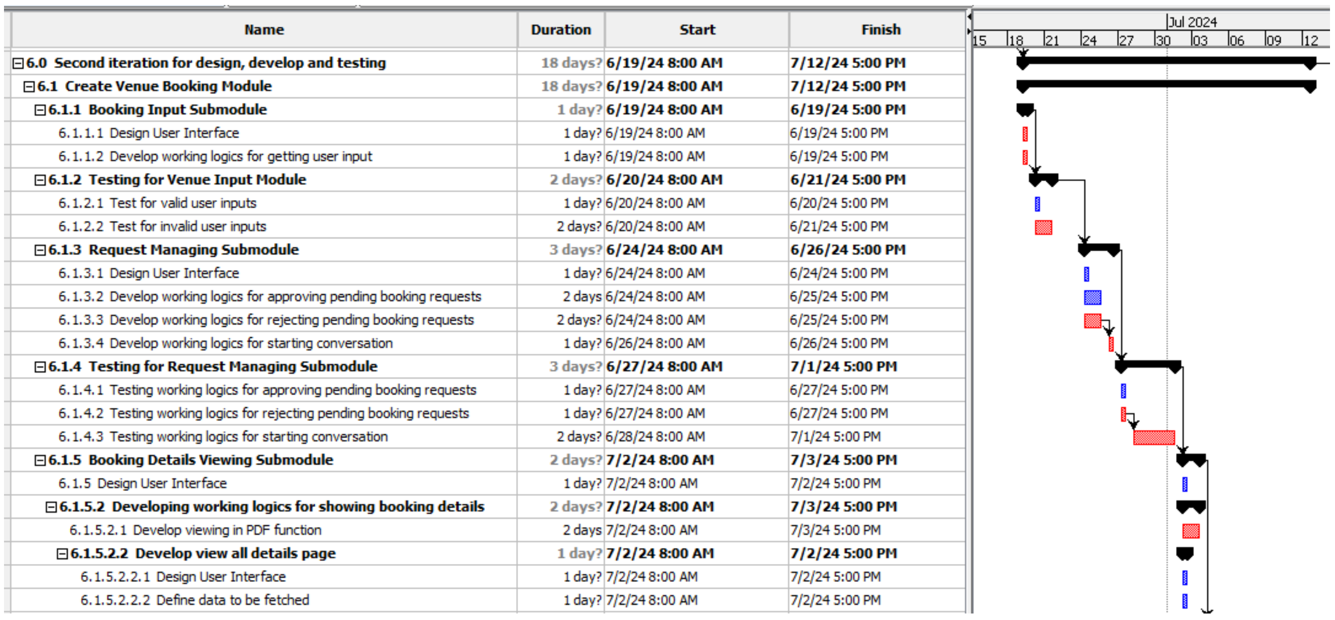
3.5.5 First iteration for design, develop and testing



5.0 First Iteration for design, develop and testing	8 days?	**6/7/24 8:00 AM**	**6/18/24 5:00 PM**
5.1 Setup Deployment Platform	1 day?	**6/7/24 8:00 AM**	**6/7/24 5:00 PM**
5.2 Setup database	1 day?	**6/10/24 8:00 AM**	**6/10/24 5:00 PM**
5.2.1 Configure database route	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.2.2 Testing	1 day?	**6/10/24 8:00 AM**	**6/10/24 5:00 PM**
5.2.2.1 Create dummy data tables	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.2.2.2 Test run on browser	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.3 Setup External Libraries	2 days?	**6/10/24 8:00 AM**	**6/11/24 5:00 PM**
5.3.1 Download Bootstrap library	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.3.2 Declare Bootstrap library in project repository	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.3.3 Install PHPMailer	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.3.4 Environment configuration for PHPMailer in project	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.3.5 Install Ultramsg package	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.3.6 Declare Ultramsg configurations in project repository	1 day?	6/11/24 8:00 AM	6/11/24 5:00 PM
5.3.7 Install FPDF package	1 day?	6/10/24 8:00 AM	6/10/24 5:00 PM
5.3.8 Set environment configuration of FPDF in project repository	1 day?	6/11/24 8:00 AM	6/11/24 5:00 PM
5.4 Create Login Module	3 days?	**6/12/24 8:00 AM**	**6/14/24 5:00 PM**
5.4.1 Design Login Page User Interface	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.4.2 Developing authentication logics for login process	2 days	**6/12/24 8:00 AM**	**6/13/24 5:00 PM**
5.4.2.1 Valid Login	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.4.2.2 Invalid Login	1 day	6/13/24 8:00 AM	6/13/24 5:00 PM
5.4.3 Test Login Module	1 day	**6/14/24 8:00 AM**	**6/14/24 5:00 PM**
5.4.3.1 Test for Valid Login	1 day?	6/14/24 8:00 AM	6/14/24 5:00 PM
5.4.3.2 Test for Invalid Login	1 day?	6/14/24 8:00 AM	6/14/24 5:00 PM
5.4.4 Design Forget Password Module interface	1 day	**6/12/24 8:00 AM**	**6/12/24 5:00 PM**
5.4.4.1 Forget Password page	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.4.4.2 Reset Password page	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.4.5 Developing logics for resetting password	1 day	**6/12/24 8:00 AM**	**6/12/24 5:00 PM**
5.4.5.1 Forget password	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.4.5.2 Reset password	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.4.6 Test Forget Password module	1 day	**6/12/24 8:00 AM**	**6/12/24 5:00 PM**
5.4.6.1 Forget password	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.4.6.2 Reset password	1 day?	6/12/24 8:00 AM	6/12/24 5:00 PM
5.5 Create Dashboard Module	2 days?	**6/17/24 8:00 AM**	**6/18/24 5:00 PM**

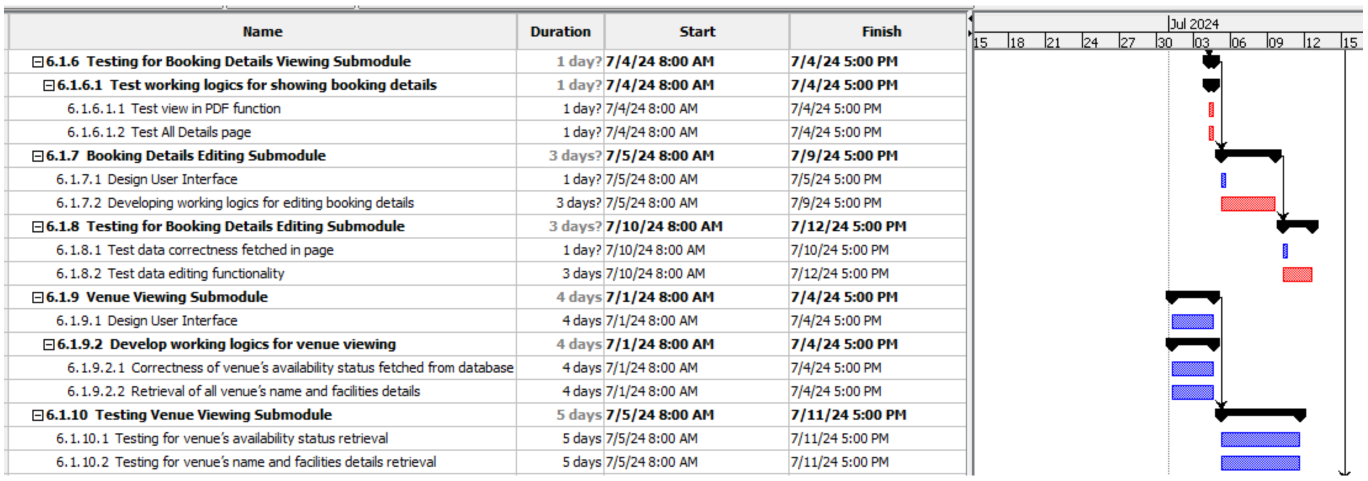
Figure 3.6: First iteration for design, develop and testing timeline

3.5.6 Second iteration for design, develop and testing



Name	Duration	Start	Finish
6.0 Second iteration for design, develop and testing	18 days?	**6/19/24 8:00 AM**	**7/12/24 5:00 PM**
6.1 Create Venue Booking Module	18 days?	**6/19/24 8:00 AM**	**7/12/24 5:00 PM**
6.1.1 Booking Input Submodule	1 day?	**6/19/24 8:00 AM**	**6/19/24 5:00 PM**
6.1.1.1 Design User Interface	1 day?	6/19/24 8:00 AM	6/19/24 5:00 PM
6.1.1.2 Develop working logics for getting user input	1 day?	6/19/24 8:00 AM	6/19/24 5:00 PM
6.1.2 Testing for Venue Input Module	2 days?	**6/20/24 8:00 AM**	**6/21/24 5:00 PM**
6.1.2.1 Test for valid user inputs	1 day?	6/20/24 8:00 AM	6/20/24 5:00 PM
6.1.2.2 Test for invalid user inputs	2 days?	6/20/24 8:00 AM	6/21/24 5:00 PM
6.1.3 Request Managing Submodule	3 days?	**6/24/24 8:00 AM**	**6/26/24 5:00 PM**
6.1.3.1 Design User Interface	1 day?	6/24/24 8:00 AM	6/24/24 5:00 PM
6.1.3.2 Develop working logics for approving pending booking requests	2 days	6/24/24 8:00 AM	6/25/24 5:00 PM
6.1.3.3 Develop working logics for rejecting pending booking requests	2 days?	6/24/24 8:00 AM	6/25/24 5:00 PM
6.1.3.4 Develop working logics for starting conversation	1 day?	6/26/24 8:00 AM	6/26/24 5:00 PM
6.1.4 Testing for Request Managing Submodule	3 days?	**6/27/24 8:00 AM**	**7/1/24 5:00 PM**
6.1.4.1 Testing working logics for approving pending booking requests	1 day?	6/27/24 8:00 AM	6/27/24 5:00 PM
6.1.4.2 Testing working logics for rejecting pending booking requests	1 day?	6/27/24 8:00 AM	6/27/24 5:00 PM
6.1.4.3 Testing working logics for starting conversation	2 days?	6/28/24 8:00 AM	7/1/24 5:00 PM
6.1.5 Booking Details Viewing Submodule	2 days?	**7/2/24 8:00 AM**	**7/3/24 5:00 PM**
6.1.5 Design User Interface	1 day?	7/2/24 8:00 AM	7/2/24 5:00 PM
6.1.5.2 Developing working logics for showing booking details	2 days?	**7/2/24 8:00 AM**	**7/3/24 5:00 PM**
6.1.5.2.1 Develop viewing in PDF function	2 days	7/2/24 8:00 AM	7/3/24 5:00 PM
6.1.5.2.2 Develop view all details page	1 day?	**7/2/24 8:00 AM**	**7/2/24 5:00 PM**
6.1.5.2.2.1 Design User Interface	1 day?	7/2/24 8:00 AM	7/2/24 5:00 PM
6.1.5.2.2.2 Define data to be fetched	1 day?	7/2/24 8:00 AM	7/2/24 5:00 PM

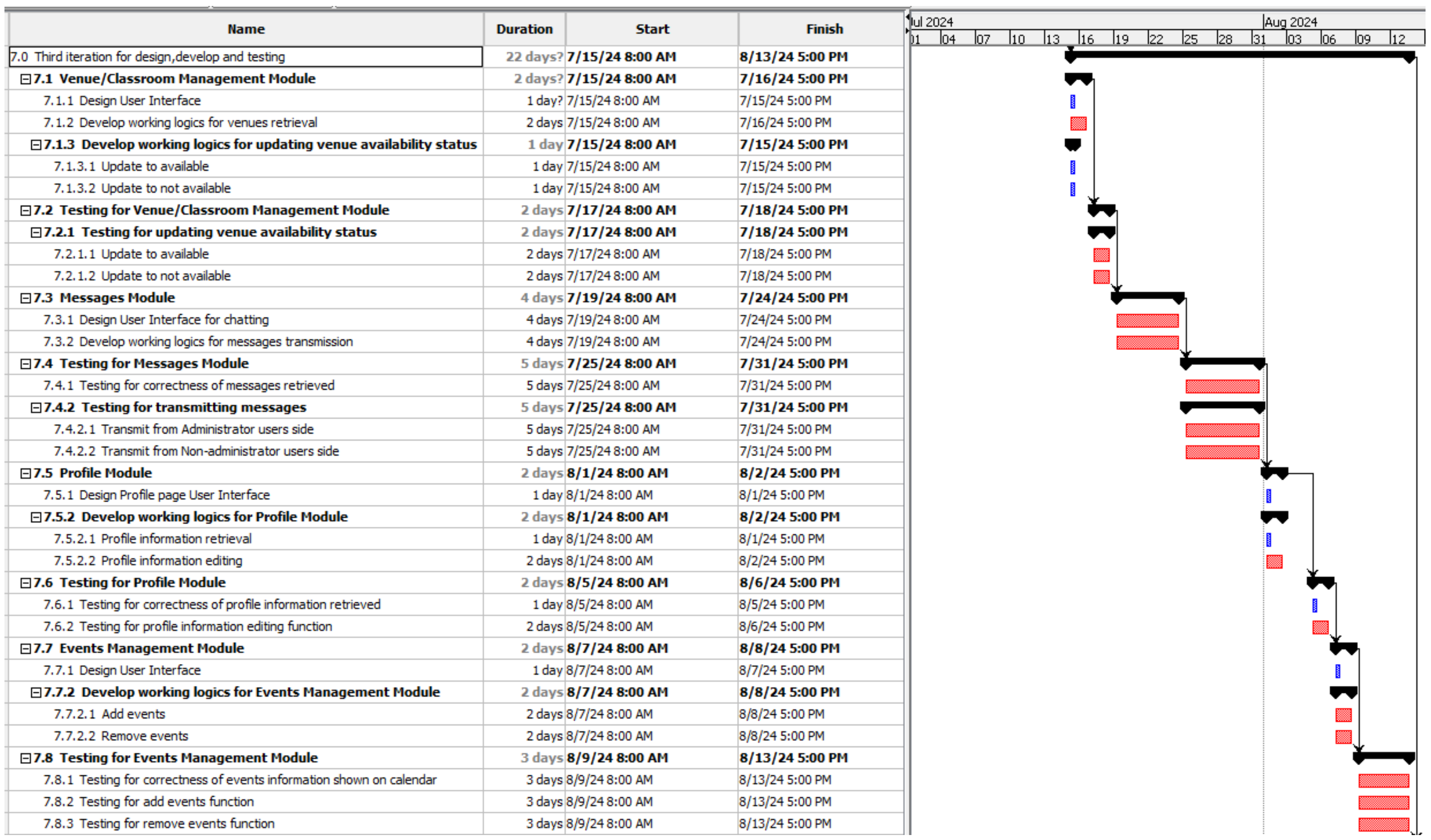
Figure 3.7: Part 1 Second iteration for design, develop and testing timeline



Name	Duration	Start	Finish
⊟6.1.6 Testing for Booking Details Viewing Submodule	**1 day?**	**7/4/24 8:00 AM**	**7/4/24 5:00 PM**
⊟6.1.6.1 Test working logics for showing booking details	**1 day?**	**7/4/24 8:00 AM**	**7/4/24 5:00 PM**
6.1.6.1.1 Test view in PDF function	1 day?	7/4/24 8:00 AM	7/4/24 5:00 PM
6.1.6.1.2 Test All Details page	1 day?	7/4/24 8:00 AM	7/4/24 5:00 PM
⊟6.1.7 Booking Details Editing Submodule	**3 days?**	**7/5/24 8:00 AM**	**7/9/24 5:00 PM**
6.1.7.1 Design User Interface	1 day?	7/5/24 8:00 AM	7/5/24 5:00 PM
6.1.7.2 Developing working logics for editing booking details	3 days?	7/5/24 8:00 AM	7/9/24 5:00 PM
⊟6.1.8 Testing for Booking Details Editing Submodule	**3 days?**	**7/10/24 8:00 AM**	**7/12/24 5:00 PM**
6.1.8.1 Test data correctness fetched in page	1 day?	7/10/24 8:00 AM	7/10/24 5:00 PM
6.1.8.2 Test data editing functionality	3 days	7/10/24 8:00 AM	7/12/24 5:00 PM
⊟6.1.9 Venue Viewing Submodule	**4 days**	**7/1/24 8:00 AM**	**7/4/24 5:00 PM**
6.1.9.1 Design User Interface	4 days	7/1/24 8:00 AM	7/4/24 5:00 PM
⊟6.1.9.2 Develop working logics for venue viewing	**4 days**	**7/1/24 8:00 AM**	**7/4/24 5:00 PM**
6.1.9.2.1 Correctness of venue's availability status fetched from database	4 days	7/1/24 8:00 AM	7/4/24 5:00 PM
6.1.9.2.2 Retrieval of all venue's name and facilities details	4 days	7/1/24 8:00 AM	7/4/24 5:00 PM
⊟6.1.10 Testing Venue Viewing Submodule	**5 days**	**7/5/24 8:00 AM**	**7/11/24 5:00 PM**
6.1.10.1 Testing for venue's availability status retrieval	5 days	7/5/24 8:00 AM	7/11/24 5:00 PM
6.1.10.2 Testing for venue's name and facilities details retrieval	5 days	7/5/24 8:00 AM	7/11/24 5:00 PM

Figure 3.8: Part 2 for Second iteration for design, develop and testing timeline

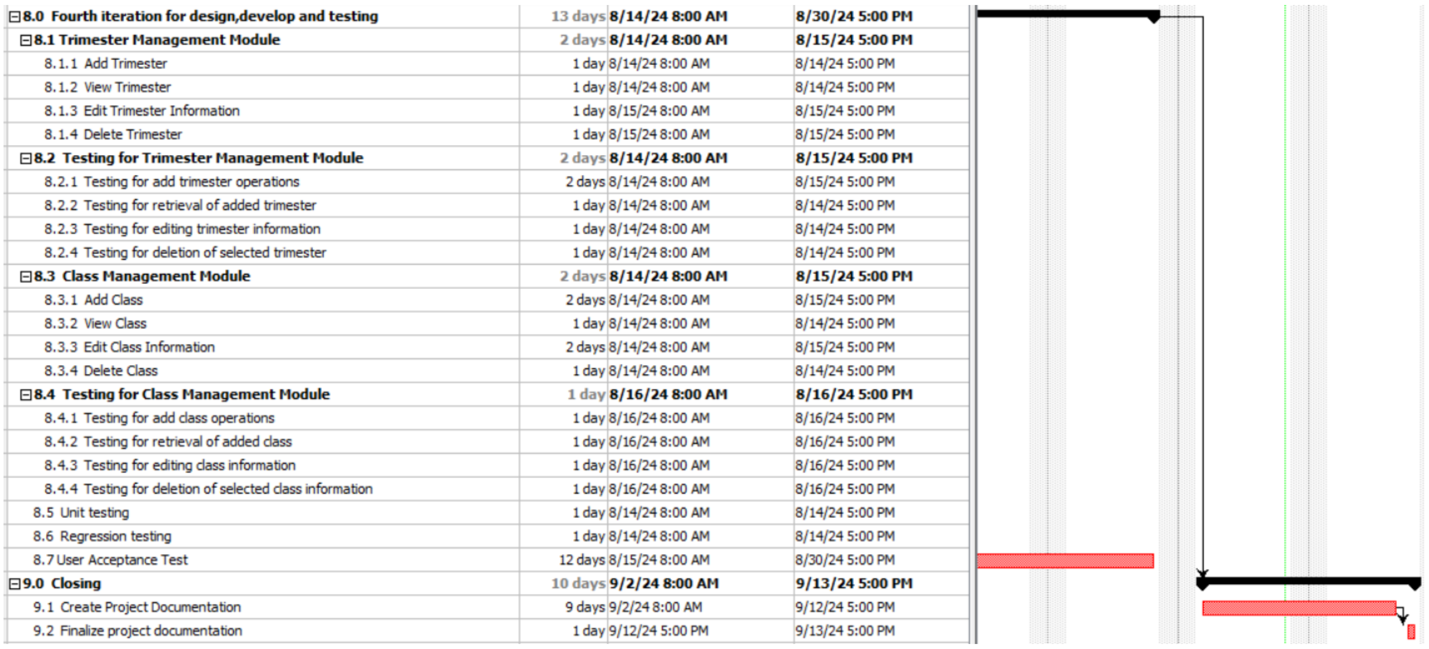
3.5.7 Third iteration for design, develop and testing



Name	Duration	Start	Finish
7.0 Third iteration for design,develop and testing	22 days?	**7/15/24 8:00 AM**	**8/13/24 5:00 PM**
⊟7.1 Venue/Classroom Management Module	**2 days?**	**7/15/24 8:00 AM**	**7/16/24 5:00 PM**
7.1.1 Design User Interface	1 day?	7/15/24 8:00 AM	7/15/24 5:00 PM
7.1.2 Develop working logics for venues retrieval	2 days	7/15/24 8:00 AM	7/16/24 5:00 PM
⊟7.1.3 Develop working logics for updating venue availability status	**1 day**	**7/15/24 8:00 AM**	**7/15/24 5:00 PM**
7.1.3.1 Update to available	1 day	7/15/24 8:00 AM	7/15/24 5:00 PM
7.1.3.2 Update to not available	1 day	7/15/24 8:00 AM	7/15/24 5:00 PM
⊟7.2 Testing for Venue/Classroom Management Module	**2 days**	**7/17/24 8:00 AM**	**7/18/24 5:00 PM**
⊟7.2.1 Testing for updating venue availability status	**2 days**	**7/17/24 8:00 AM**	**7/18/24 5:00 PM**
7.2.1.1 Update to available	2 days	7/17/24 8:00 AM	7/18/24 5:00 PM
7.2.1.2 Update to not available	2 days	7/17/24 8:00 AM	7/18/24 5:00 PM
⊟7.3 Messages Module	**4 days**	**7/19/24 8:00 AM**	**7/24/24 5:00 PM**
7.3.1 Design User Interface for chatting	4 days	7/19/24 8:00 AM	7/24/24 5:00 PM
7.3.2 Develop working logics for messages transmission	4 days	7/19/24 8:00 AM	7/24/24 5:00 PM
⊟7.4 Testing for Messages Module	**5 days**	**7/25/24 8:00 AM**	**7/31/24 5:00 PM**
7.4.1 Testing for correctness of messages retrieved	5 days	7/25/24 8:00 AM	7/31/24 5:00 PM
⊟7.4.2 Testing for transmitting messages	**5 days**	**7/25/24 8:00 AM**	**7/31/24 5:00 PM**
7.4.2.1 Transmit from Administrator users side	5 days	7/25/24 8:00 AM	7/31/24 5:00 PM
7.4.2.2 Transmit from Non-administrator users side	5 days	7/25/24 8:00 AM	7/31/24 5:00 PM
⊟7.5 Profile Module	**2 days**	**8/1/24 8:00 AM**	**8/2/24 5:00 PM**
7.5.1 Design Profile page User Interface	1 day	8/1/24 8:00 AM	8/1/24 5:00 PM
⊟7.5.2 Develop working logics for Profile Module	**2 days**	**8/1/24 8:00 AM**	**8/2/24 5:00 PM**
7.5.2.1 Profile information retrieval	1 day	8/1/24 8:00 AM	8/1/24 5:00 PM
7.5.2.2 Profile information editing	2 days	8/1/24 8:00 AM	8/2/24 5:00 PM
⊟7.6 Testing for Profile Module	**2 days**	**8/5/24 8:00 AM**	**8/6/24 5:00 PM**
7.6.1 Testing for correctness of profile information retrieved	1 day	8/5/24 8:00 AM	8/5/24 5:00 PM
7.6.2 Testing for profile information editing function	2 days	8/5/24 8:00 AM	8/6/24 5:00 PM
⊟7.7 Events Management Module	**2 days**	**8/7/24 8:00 AM**	**8/8/24 5:00 PM**
7.7.1 Design User Interface	1 day	8/7/24 8:00 AM	8/7/24 5:00 PM
⊟7.7.2 Develop working logics for Events Management Module	**2 days**	**8/7/24 8:00 AM**	**8/8/24 5:00 PM**
7.7.2.1 Add events	2 days	8/7/24 8:00 AM	8/8/24 5:00 PM
7.7.2.2 Remove events	2 days	8/7/24 8:00 AM	8/8/24 5:00 PM
⊟7.8 Testing for Events Management Module	**3 days**	**8/9/24 8:00 AM**	**8/13/24 5:00 PM**
7.8.1 Testing for correctness of events information shown on calendar	3 days	8/9/24 8:00 AM	8/13/24 5:00 PM
7.8.2 Testing for add events function	3 days	8/9/24 8:00 AM	8/13/24 5:00 PM
7.8.3 Testing for remove events function	3 days	8/9/24 8:00 AM	8/13/24 5:00 PM

Figure 3.9: Third iteration for design, develop and testing timeline

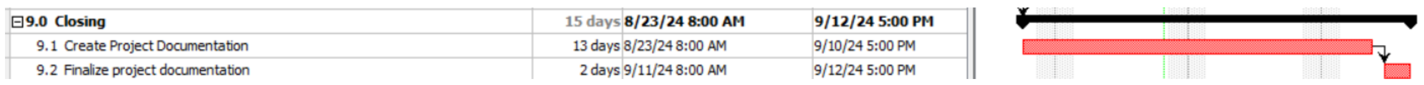
3.5.8 Fourth iteration for design, develop and testing



⊟8.0 Fourth iteration for design,develop and testing	13 days	8/14/24 8:00 AM	8/30/24 5:00 PM
⊟8.1 Trimester Management Module	2 days	8/14/24 8:00 AM	8/15/24 5:00 PM
8.1.1 Add Trimester	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
8.1.2 View Trimester	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
8.1.3 Edit Trimester Information	1 day	8/15/24 8:00 AM	8/15/24 5:00 PM
8.1.4 Delete Trimester	1 day	8/15/24 8:00 AM	8/15/24 5:00 PM
⊟8.2 Testing for Trimester Management Module	2 days	8/14/24 8:00 AM	8/15/24 5:00 PM
8.2.1 Testing for add trimester operations	2 days	8/14/24 8:00 AM	8/15/24 5:00 PM
8.2.2 Testing for retrieval of added trimester	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
8.2.3 Testing for editing trimester information	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
8.2.4 Testing for deletion of selected trimester	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
⊟8.3 Class Management Module	2 days	8/14/24 8:00 AM	8/15/24 5:00 PM
8.3.1 Add Class	2 days	8/14/24 8:00 AM	8/15/24 5:00 PM
8.3.2 View Class	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
8.3.3 Edit Class Information	2 days	8/14/24 8:00 AM	8/15/24 5:00 PM
8.3.4 Delete Class	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
⊟8.4 Testing for Class Management Module	1 day	8/16/24 8:00 AM	8/16/24 5:00 PM
8.4.1 Testing for add class operations	1 day	8/16/24 8:00 AM	8/16/24 5:00 PM
8.4.2 Testing for retrieval of added class	1 day	8/16/24 8:00 AM	8/16/24 5:00 PM
8.4.3 Testing for editing class information	1 day	8/16/24 8:00 AM	8/16/24 5:00 PM
8.4.4 Testing for deletion of selected class information	1 day	8/16/24 8:00 AM	8/16/24 5:00 PM
8.5 Unit testing	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
8.6 Regression testing	1 day	8/14/24 8:00 AM	8/14/24 5:00 PM
8.7 User Acceptance Test	12 days	8/15/24 8:00 AM	8/30/24 5:00 PM
⊟9.0 Closing	10 days	9/2/24 8:00 AM	9/13/24 5:00 PM
9.1 Create Project Documentation	9 days	9/2/24 8:00 AM	9/12/24 5:00 PM
9.2 Finalize project documentation	1 day	9/12/24 5:00 PM	9/13/24 5:00 PM

Figure 3.10: Fourth iteration for design, develop and testing timeline

3.5.9 Closing phase



⊟9.0 Closing	15 days	8/23/24 8:00 AM	9/12/24 5:00 PM
9.1 Create Project Documentation	13 days	8/23/24 8:00 AM	9/10/24 5:00 PM
9.2 Finalize project documentation	2 days	9/11/24 8:00 AM	9/12/24 5:00 PM

Figure 3.11: Closing phase Gantt Chart overview

3.6 Summary

With a planned phases of activities to follow within a fixed timeframe, it was expected that the development project of the Universiti Tunku Abdul Rahman (UTAR) Venue/Classroom Booking Web Portal could meet the schedule described in the Gantt chart, achieving the functionalities mentioned in the subchapter 3.2.1.2.3 as well. In the process of preparing for development, interviews, survey and comparison must be carried out foremost to clarify the dos and don’ts of such a system. Since the development tools had been finalized, the project should consistently produce outputs and gain constructive feedback at each iteration phase, since this project adopts the phased software development methodology which its motto is to respond to changes. Luckily, the outcomes delivered were meeting the expectations.

CHAPTER 4

PROJECT SPECIFICATION

4.1 Introduction

Based on the opinions and needs gained from the requirements elicitation activity, the requirements are finalized and will be described in this chapter. A detailed explanation of each module's flow will be included, along with the snapshots of the prototype developed. The logical flow between the modules will be illustrated as well through figures, giving a clearer view of the relationships between the modules in the UTAR Venue/Classroom Booking Web Portal.

4.2 Fact Finding

Qualitative and quantitative methods were utilized when conducting requirements elicitation from the UTAR community, depending on the participants' status. As a descriptive response is required, interviewing would be the best method. The interviews started in March 2024 and took around a month to complete, involving a total of 9 respondents.

Whereas for quantitative methods, an online survey was applied to get opinions. After passing the ethical clearance, the online survey managed to obtain 15 responses. The survey questions were split into three sections, which could be referred to Appendix D.

The following table describes the utilization of both research methods in this development project.

Table 4.1: Comparison of research methods

Aspect / **Methods**	**Quantitative**	**Qualitative**
Approach used	Online survey	Physical interview
Participant's status	Lecturers	Students and DGS staff

Participant count	14	8

4.2.1 Interview with DGS

The interview involved the Head of DGS, Mr. Foo Cher Siang and the assistant Miss Yee Sau Keng. The following describes the responses concluded from this interview.

To get a full understanding of the booking procedures, Mr Foo was asked to give a detailed description of this matter. According to Mr Foo (2024), the booking requesters are divided into two categories, that are the UTAR lecturers and students. Lecturers need to submit their venue/classroom booking request to their respective Faculty Head of Department (HOD) for review. Once the request is agreed upon by the HOD, then the request will be transferred to the DGS for a second round of review and venue/classroom arrangement. If the requested venue/classroom is available, then DGS will approve then notify the requesting lecturer through email. Else if the venue is occupied, then DGS will suggest another venue. Students' requests will be handled by the DSA (Department of Students Affairs), followed by DGS to book the venue/classrooms eventually.

The venues/classrooms booking coverage includes the lecture theaters, tutorial rooms, Multi-Purpose Hall (MPH) and meeting rooms. The requests from both parties are transmitted to DGS through email mostly, as long as they get approval from the HOD (lecturer) or DSA (students). The venue arrangements are prioritized according to the importance of the event, not only managing it on a first-come first-served basis. There are booking requests that get rejected by DGS, depending on the event appropriateness.

There are much room for improvement on the booking logics, like the booking request is suggested to be sent to a third party for supervision since there are cases where DGS received complaints that the venue is not fully utilised or the event gets cancelled without notifying, causing wastage. Other than this suggestion, it was recommended by Mr Foo that the notification email could be sent to the booking requester and the supervisor to enforce stricter validation and thorough supervision. The booking requester should provide a detailed booking reason as well, so that the DGS could get a better understanding of the venue booking motive and in turn increase the rate of approval.

Regarding the cancellation, it was suggested that the duration of confirmation before the event could be as early as possible. This is because sudden confirmation might lead to insufficient preparation time for the succeeding venue booking requester, especially if the events are formal ones.

To comment, the existing system is not user-friendly due to its inability to integrate each faculty’s timetable into the current system timetable, which increases the DGS workload to compare both timetables side by side when it comes to venue/classroom availability checking. Therefore, it was suggested that the faculty staff be recognized as the system’s end-user as well, so that the timetables are arranged in the same platform.

In terms of satisfaction with the using experience, it is important for DGS to classify the events so that the venue/classroom could be arranged logically. For example, events that tend to be noisier (such as musical performances or society trainings) are given additional attention to not be arranged in lecture theaters during working hours so that other venues/classrooms that have formal occasions would not be affected. Addressing this issue, it is better to provide the controls to DGS staff to choose the highlighting colours, so that the events shown are classified visually.

In terms of usability, it would be better if the venue/classroom booking system could be accessed on mobile devices. This is because DGS staff often get contacted during non-working hours to check the venue/classroom arrangement, which is why sudden checking is needed in this case.

4.2.2 Survey Responses from Lecturers

The survey was released to the UTAR lecturers' community through email blasting with the help of MailMaster and obtained 14 responses at last. The following section describes the findings concluded from the survey.

4.2.2.1 Demographic of the Lecturers

Before starting to discuss the results, the demographic of the lecturers will be analysed first. There are seven departments involved in the survey. There are 5 respondents from the LKCFES, showing the largest rate of 38.5%. The population formed from 4 FAM lecturers (30.8%), 3 FCI lecturers (23.1%) and lastly 1 FMHS lecturer. The following figure shows the participation rate of the lecturers represented by their departments.

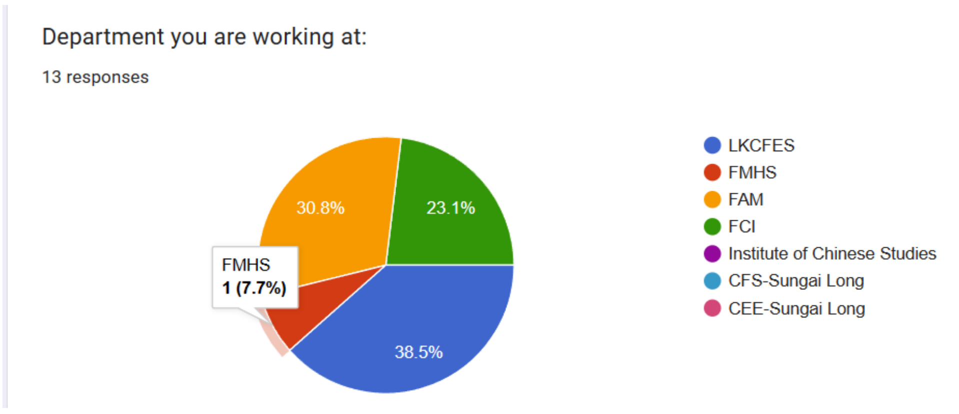


Figure 4.1: Participation rate of the lecturers in terms of their departments

4.2.2.2 Opinions on the current system

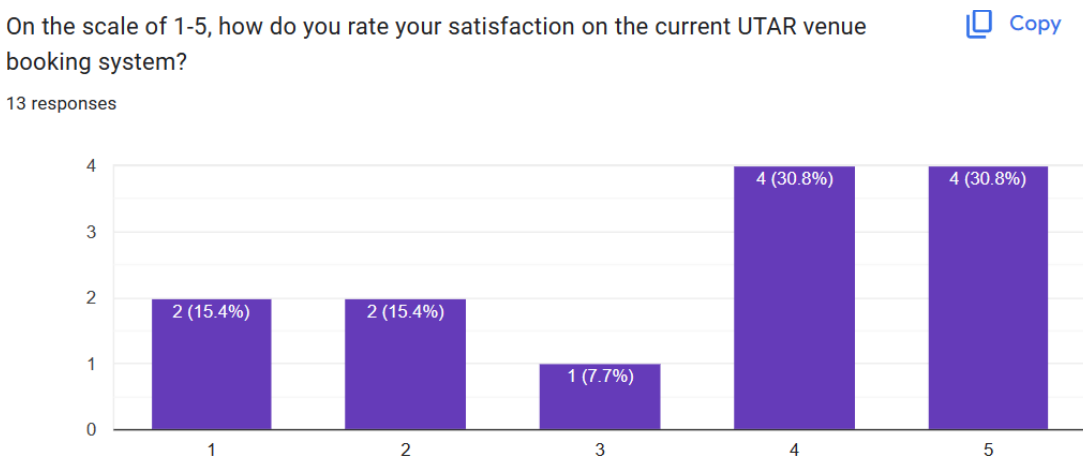
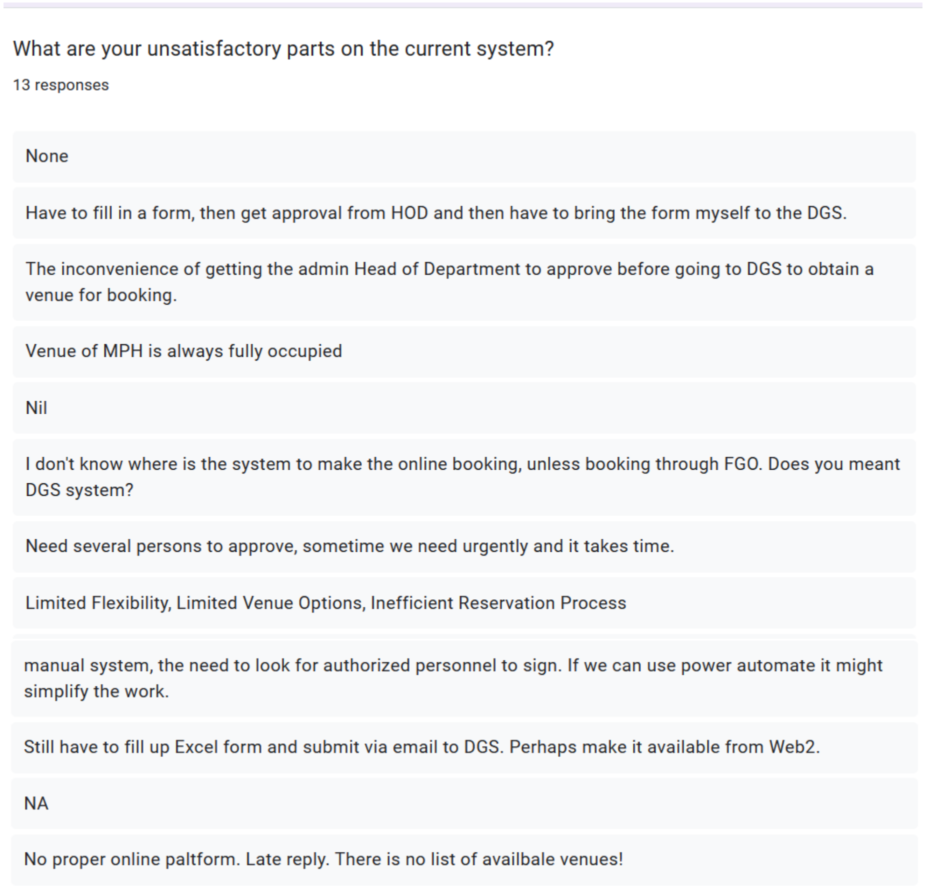


Figure 4.2: Satisfaction rate on the current system

Inquiring the satisfaction rate on the current venue/classroom booking system, with the scale of the number expressing the highest satisfaction rate. From the data collected, it is known that the lecturers are mostly satisfied on the current system, which could be shown by the rating of 30.8% for both 4 and 5 ratings. While there were also lecturers who were not satisfied with the current system, showing the same rate of 15.4% on the scores of 1 and 2.


What are your unsatisfactory parts on the current system?

13 responses

None

Have to fill in a form, then get approval from HOD and then have to bring the form myself to the DGS.

The inconvenience of getting the admin Head of Department to approve before going to DGS to obtain a venue for booking.

Venue of MPH is always fully occupied

Nil

I don't know where is the system to make the online booking, unless booking through FGO. Does you meant DGS system?

Need several persons to approve, sometime we need urgently and it takes time.

Limited Flexibility, Limited Venue Options, Inefficient Reservation Process

manual system, the need to look for authorized personnel to sign. If we can use power automate it might simplify the work.

Still have to fill up Excel form and submit via email to DGS. Perhaps make it available from Web2.

NA

No proper online paltform. Late reply. There is no list of availbale venues!

Figure 4.3: Unsatisfactory opinions on the current system

The next question is a descriptive one, which required the respondents to state their disappointments on the current system so that the project author could elicit the area of improvement. Aside from the 3 three neutral responses, most of the lecturers complained that they need to book the venues in different layers, causing much effort to be spent on waiting for each other’s response to get the booking request transferred to the next party. While complaining about the logical workflows which cause delays, it is known that the functionalities offered by the current system is limited (no venues/classrooms to be viewed) and most of the processes still need to be performed manually. Through this question, the project author is clear that the development of the planned system is a must, and the timing delay issue should be prioritized to boost the venue booking efficiency.

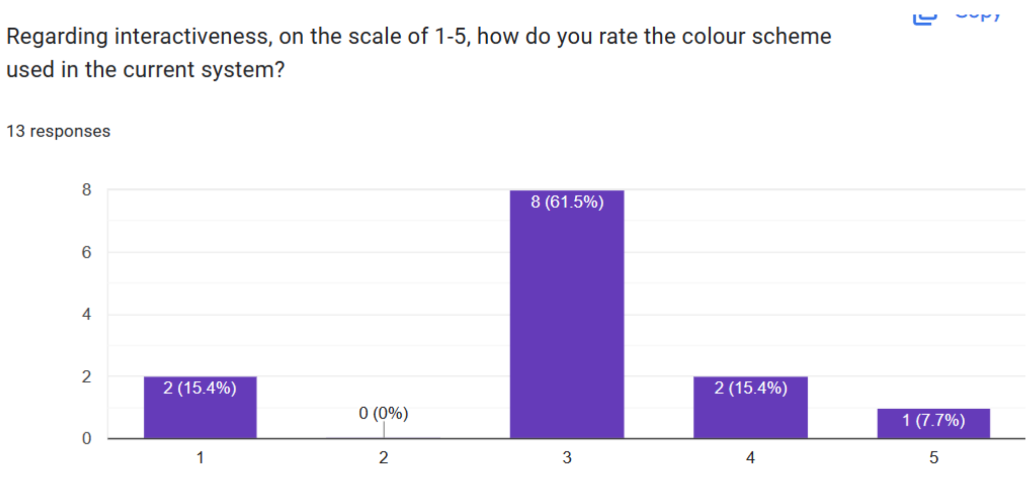


Figure 4.4: Interactiveness rating score for the current system

The following question relates to the Human Computer Interaction Design field, inquiring about the satisfaction rate on the colour scheme for the current system. This question is designed so that the project author could clarify the necessity of redesigning the entire system. From Figure 4.4, most of the lecturers show neutral responses on this issue, followed by another group of lecturers that are quite pleased with the design elements of the current system.

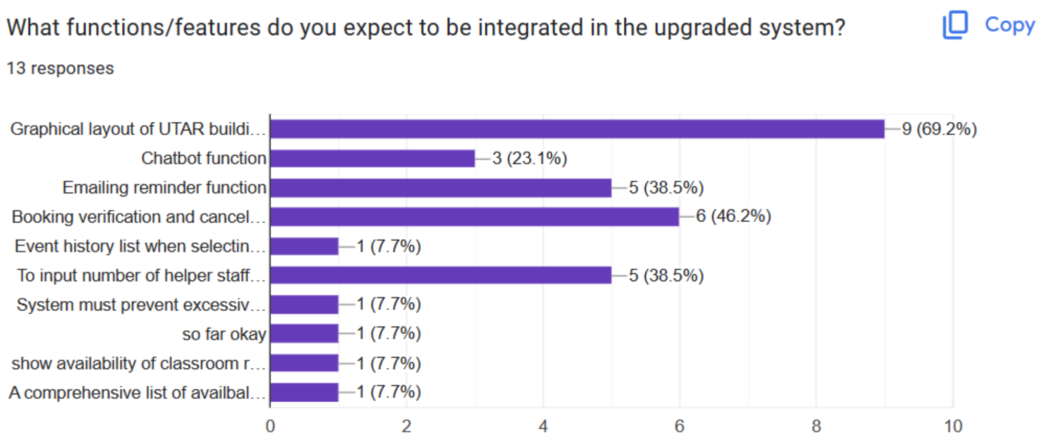


Figure 4.5: Functions expected for the system to be developed

While the following question inquires about the features that the lecturers most wanted for the system to be developed. Refer to Appendix D for the whole view of the selections provided, 69.2% of the participating lecturers shown utmost support on the implementation of a graphical layout for all UTAR venues/classrooms in the form of floor plan, showing the availability status as well. The other features supported are matching what planned by the project author, that is to add the process for verifying the booking and to cancel before the event starts (46.2%), the email reminder function and to get help of helper staff for certain events (38.5%) and the chatting function (23.1%). While this question also enables the lecturers to state their suggestions, there were also creative ideas like providing a comprehensive list of available venues, showing availability status of the venue/classroom directly in the system and preventing excessive booking duration for a venue/classroom that are hugely demanded. The project author appreciates the responses given.

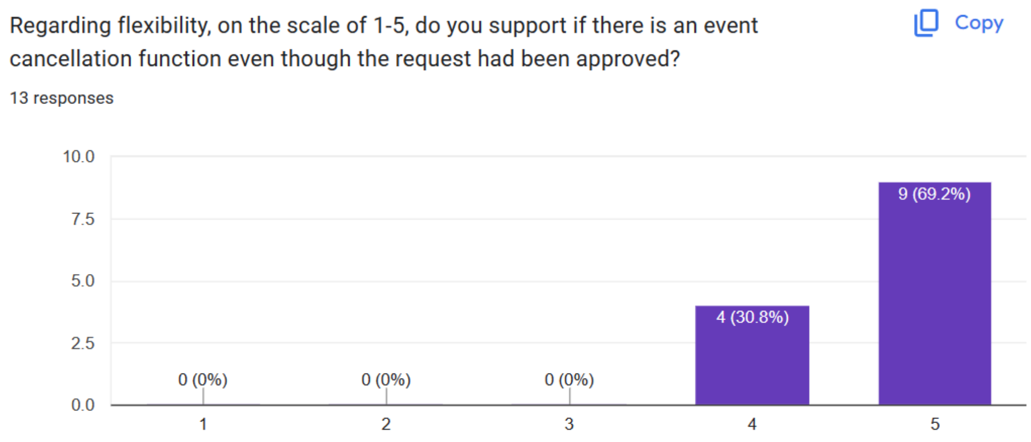


Figure 4.6: Scoring for supporting the cancellation of an event

This question inquired about the lecturers' supportiveness on the essentiality for the event cancellation function although the event has been approved. This question receives positive responses where all lecturers supported this feature, just varies on the degree of supportiveness, which 4 lecturers (30.8%) were likely to support while the other 9 lecturers (69.2%) supported it.

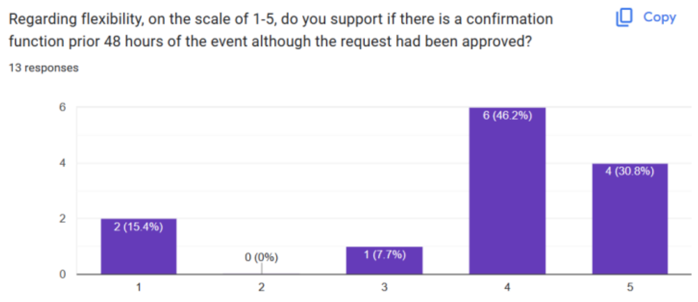


Figure 4.7: Scoring for supporting the confirmation of an event

Regarding the lecturers' supportiveness on the essentiality for prompting the booking requester whether the booked event still carry on although the event starts. While most of the lecturers supported this feature (77%), there is still 1 lecturer that was neutral and not supporting this matter (15.4%). Seeing that this feature mostly receives positive responses, the project author might consider this suggestion as well.

4.2.3 Interview with UTAR Students

The project author had arranged appointments with a total of 9 students respectively on the 11th, 13th, 19th, 25th, 27th and 29th March 2024, which lasted from 15 minutes to 45 minutes approximately. The following paragraphs summarize the findings from the students.

Regarding the venue booking procedures, the communication medium is a hybrid of oral communication and technology, where the students tend to meet up with the DSA staff to submit a venue/classroom booking request, then the DSA staff will notify them of the booking status through email. The students reported that the response duration is unstable, which the response could be within a few hours to 2 working days, depending on the workload of the departments' staff in different stages of a trimester. The inconsistent response duration causes the students to prefer manual communication since the information could be conveyed faster.

Another factor that causes dissatisfaction to rise among the students is that venue conflicts arise upon the event. There were cases in which another party was using the same venue despite it was being booked early. They believe this issue happened due to the lack of booking arrangement coordination between the staff. Consequently, they had to switch to another venue, disrupting the plans. The frequency of having this issue depends on the existence of human errors by the staff in managing the booking requests.

Throughout the interview, the students expressed their expectations of the new system as well. It was suggested notifying the users through Instant Messaging applications, instead of purely email. This is because some would not check their inbox frequently, which led to the neglection of venue booking updates.

Another new feature to be added is the timetable function, which the users could see the availability of each venue/classroom in a certain hour. This reduces the waiting process for the students as originally the DGS staff will need to check the availability of the venue requested then only proceeds to inform them.

It was suggested that those students which had requested for co-curricular reasons must be prioritized, followed by the students which are supported by normal request reasons only. This is because those students who requested for their society events tend to possess a lower misuse risk compared with those students who requested for non-society reasons.

On a scale of 1-5, all interviewed students are supporting the project development though they view this opportunity as a direct approach for them to handle the venue/classroom booking issue.

4.3 Requirements Specification

This section describes the functional requirements and non-functional requirements. Concluded from the survey and interview responses, the features to be delivered to the end-users (functional requirements) and the attributes to be fulfilled by this system (non-functional requirements) will be described in this subsection.

4.3.1 Functional Requirements

The following lists the functional requirements that will be delivered by the UTAR Venue/Classroom Booking Web Portal. To distinguish the targeted users in each functional requirement, the term “all levels of users” refers to the administrative users (DGS staff), HOD staff, lecturers and the students, unless specified wise.

1. The system shall allow all levels of users to log in to the system with their respective account, inclusive of resetting password if forgotten.
2. The system shall allow all levels of users to manage their own profile, including viewing and editing the details.
3. The system shall allow all levels of users to manage their chat messages, including viewing previous conversation messages and continue chatting.
4. The system shall allow all levels of users to manage their booking requests.
5. The system shall allow all levels of users to submit a booking request.
6. The system shall allow all levels of users to view their own venue booking requests.
7. The system shall allow all levels of users to view their booking information in a PDF report.
8. The system shall allow all levels of users to view all events.
9. The system shall allow DGS staff and HOD to add, edit or delete the events.
10. The system shall allow all levels of users to view all UTAR venues/classrooms.
11. The system shall allow the DGS staff to edit all venues/classrooms' information and status.
12. The system shall allow the DGS staff to manage the trimester information, including view, add, edit or delete.
13. The system shall allow the DGS staff to manage the class information, including to add, edit or delete.
14. The system shall allow all levels of users to be notified through SMS and email when it comes to booking information updates.

4.3.2 Non-Functional Requirements

This section specifies the attributes to be applied to the project outcome, which may contribute to enhancing the user experience or strengthening any aspect of the system. The common aspects include performance, scalability, security, usability, maintainability, availability and interoperability (Mudassar and Khan, 2023).

1. Availability requirements
    1.1 The system shall available to all levels of users, providing they have an active Internet connection.
    1.2 The system shall be available to all levels of users to view in desktop-sized platforms and mobile platforms.
2. Usability requirements
    2.1 The system shall allow all levels of users to access the content in English language only.
    2.2 The user interfaces of the system shall be marked with directive links and hovering hints, featured with simplistic logos as well.
    2.3 The system shall allow all levels of users to self-define the Data Tables when showing list of information.
3. Reliability requirements
    3.1 The system shall show popup messages to all levels of users when it comes to status confirmation actions.
    3.2 The system shall check and update all booking request information and venue’s availability status whenever the system loads.
    3.3 The system shall reload the dashboard page every 5 minutes for all levels of users to keep them update of any changes.
4. Security requirements
    4.1 The system shall authenticate registered UTAR email address and password with all levels of users during the login process.
    4.2 The system shall verify the role for all levels of users to prevent authorized functions on certain administrative functions.
    4.3 The system shall authenticate the identity of all levels of users to access pages which contains confidential information about the users.
    4.4 The system shall generate a hashcode for each venue/classroom booking requests to get the booking information secured.
    4.5 The system shall perform validation on user access for every system modules to ensure the confidentiality of information.
5. Performance requirements

5.1 The system shall be in the loading state for not more than 10 seconds.

5.2 The system shall validate and process input from all levels of users according to the business logics and the venue booking Standard Operating Procedures.

4.4 Use Case Modeling

4.4.1 Use Case Diagram

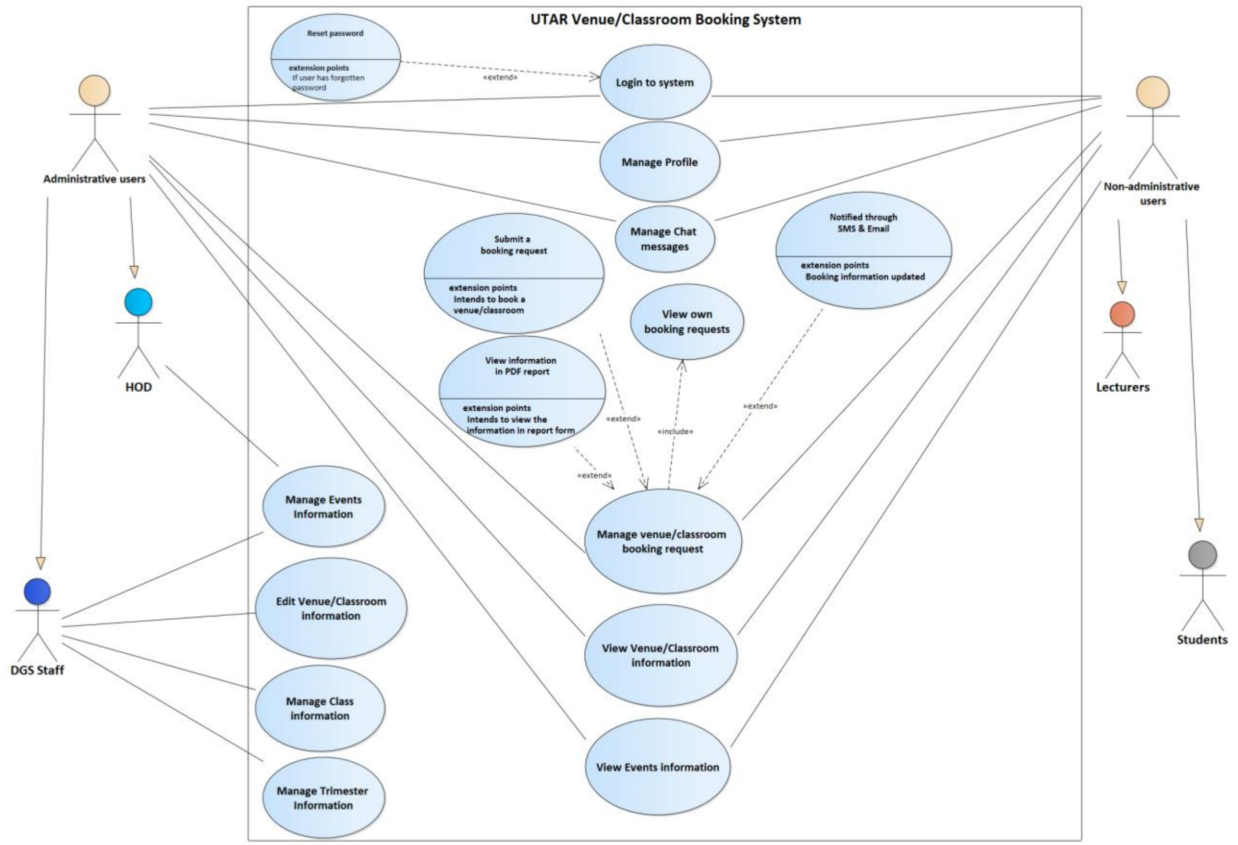

UTAR Venue/Classroom Booking System
Reset password
extension points
If user has forgotten password
«extend»
Login to system
Manage Profile
Manage Chat messages
Submit a booking request
extension points
Intends to book a venue/classroom
Notified through SMS & Email
extension points
Booking information updated
View own booking requests
View information in PDF report
extension points
Intends to view the information in report form
«extend»
«include»
«extend»
«extend»
Manage Events Information
Manage venue/classroom booking request
Edit Venue/Classroom information
View Venue/Classroom information
Manage Class information
View Events information
Manage Trimester Information
Administrative users
HOD
DGS Staff
Non-administrative users
Lecturers
Students

Figure 4.8: Use Case Diagram (UCD) for UTAR Venue/Classroom Booking Web Portal

4.4.2 Use Case Description

Table 4.2: Use Case of Login into system

Use Case Name: Login to system	**ID:** UC01	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to have access to the UTAR Venue/Classroom Booking Web Portal HOD - to have access to the UTAR Venue/Classroom Booking Web Portal Lecturers - to have access to the UTAR Venue/Classroom Booking Web Portal Students - to have access to the UTAR Venue/Classroom Booking Web Portal		
Brief Description: This use case describes how the identified end-users login into the system.		
Trigger: Customer enters the UTAR Venue/Classroom Booking Web Portal main page without having logged in yet.		
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : - Extend : Reset password Generalization: -		
Normal Flow of Events: 1. The end-user enters the main page of the UTAR Venue/Classroom Booking Web Portal. 2. The system requests the end-user to input their UTAR email address, password and to check the ReCAPTCHA form. 3. If the end-user input valid login credentials, V-1 will be executed. 4. If the end-user entered invalid login credentials, V-2 will be executed.		

5. If end-user does not enter any login data but chooses to submit, V-3 will be executed. 6. If the end-user has unrecalled of password, he/she can choose the Forget password link, V-4 will be executed.
Sub-flows: -
Alternate/Exceptional Flows: **V-1: Valid login email address and password** 1. The system approves the login data. 2. The system redirects the end-user to the dashboard interface. **V-2: Wrong login email address and password, or unchecked ReCAPTCHA form** 1. The system rejects the login data. 2. The system prompts the customer with the error message, telling which either of the details are incorrect. 3. The system stays on the same page for the end-user to re-enter the login credentials. 4. The system will follow back the Normal Flow Step-2. **V-3: Blank login credentials entered** 1. The system prompts the customer with the error message. 2. The system stays on the same page for the end-user to re-enter the login credentials. 3. The system will follow back the Normal Flow Step-2. **V-4: Reset password** 1. The system redirects the user to another page to enter their email address.

2. A password renewal link will be sent by the system to the email entered. 3. Based on the link sent in the email, user should enter a new password.

Table 4.3: Use Case of Manage Profile

Use Case Name: View Profile	**ID:** UC02	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to view and manage personal profile information for the Venue/Classroom Booking Web Portal HOD - to view and manage personal profile information for the Venue/Classroom Booking Web Portal Lecturers - to view and manage personal profile information for the Venue/Classroom Booking Web Portal Students - to view and manage personal profile information for the Venue/Classroom Booking Web Portal		
Brief Description: This use case describes how the identified end-users can access their profile for viewing purpose and editing purpose, which the information is limited for UTAR Venue/Classroom Booking Web Portal.		
Trigger: End-users access the profile page by selecting their names on the left sidebar.		
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : - Extend : - Generalization: -		

Normal Flow of Events: 1. The end-user chooses their names displayed on the sidebar after logged in. 2. The end-user is relocated to Profile Viewing interface. 3. If the end-user wishes to edit their contact number or name, S-1 will be executed.
Sub-flows: **S-1: Edit Profile Information** 1. The end-user is redirected to another page which lists the profile information of the user. 2. The end-user performs editing on either the name field or contact number field. 3. The end-user chooses to save before he/she leaves the page.
Alternate/Exceptional Flows:-

Table 4.4: Use Case of Manage Chat messages

Use Case Name: Manage Chat messages	**ID:** UC03	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to refer to chatting histories that have been established and continue chatting. HOD - to refer to chatting histories that have been established and continue chatting. Lecturers - to refer to chatting histories that have been established and continue chatting. Students - to refer to chatting histories that have been established and continue chatting.		
Brief Description:		

This use case describes how the identified end-users can view their chatting histories with another end-user in this system and continue chatting to clarify the ambiguities in the venue booking requests.
Trigger: End-users access the chatting module in the sidebar.
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : - Extend : - Generalization: -
Normal Flow of Events: 1. The end-user chooses the Messages submodule displayed on the sidebar. 2. The end-user is redirected to the View All Messages interface and choose which conversation partner to review the chatting records. 3. If there are no chatting sessions established and user wishes to communicate, then V1 is executed. 4. If the user has been chatting with someone, then he/she can choose any chatboxes to refer the chat messages and continue chatting.
Sub-flows: -
Alternate/Exceptional Flows: **V-1: Start New Chat** 1. In the dashboard page, the end-user chooses the chat function under the Actions column in the Own Request table. 2. The user is redirected to another page which consists of chatting layouts so that both parties could communicate.

Table 4.5: Use Case of Manage venue/classroom booking requests

Use Case Name: Manage venue/classroom booking requests	**ID:** UC04	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to access and read the booking requests that had been requested. HOD - to access and read the booking requests that had been requested. Lecturers - to access and read the booking requests that had been requested personally Students - to access and read the booking requests that had been requested personally		
Brief Description: This use case describes how the end-users can access the venue/classroom booking records that were requested by own profile.		
Trigger: Customer can view the venue/classroom booking requests in the dashboard.		
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : View Own Booking requests Extend : View information in PDF, Submit a Venue Booking Request, Notified through SMS and Email Generalization: -		
Normal Flow of Events: 1. The end-user is redirected to the dashboard page after logging in. 2. For HOD and DGS staff, they could specifically choose the Manage Booking(s) tab in the sidebar. 3. For every level of users, they could read own booking requests in a Data Table layout.		

4. The end-user could filter the booking requests by status through selecting the tabs above the Data Table. 5. For HOD and DGS staff, they could choose to reject or approve the venue booking requests, as described in V-1 and V-2. 6. For own booking requests, if the end-user wishes to view in web interface form, UC06 will be executed. 7. For every levels of end-users that have unapproved booking requests personally, if the end-user wishes to cancel it, then V-3 will be executed.
Sub-flows: -
Alternate/Exceptional Flows: **V-1: Approve venue booking requests** 1. HOD and DGS staff chooses the Approve button for the booking request row. 2. User is prompted if they would really want to approve the selected venue booking request. 3. If user chooses "Yes", then the status for that venue booking request is amended to "Approved". 4. If user chooses "No", then the status for that venue booking request is remaining the same. **V-2: Reject venue booking requests** 1. HOD and DGS staff chooses the Reject button for the booking request row. 2. User is prompted if they would really want to reject the selected venue booking request. 3. If user chooses "Yes", then the status for that venue booking request is amended to "Rejected".

4. If user chooses “No”, then the status for that venue booking request is remaining the same. **V-3: Cancel own booking requests** 1. User could choose the “Cancel” button in the venue booking request row. 2. User is prompted if they would really want to cancel the selected venue booking request. 3. If user chooses “Yes”, then the status for that venue booking request is amended to “Cancelled”. 4. If user chooses “No”, then the status for that venue booking request is remaining the same.

Table 4.6: Use Case of Submit a Booking Request

Use Case Name: Submit a Booking Request	**ID:** UC05	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to submit a venue/classroom booking request. HOD - to submit a venue/classroom booking request. Lecturers - to submit a venue/classroom booking request. Students - to submit a venue/classroom booking request.		
Brief Description: This use case delineates the process of identified end-users to submit venue/classroom booking request.		
Trigger: End-users access the “Booking Request” module in the sidebar, chooses the “New Booking” submodule under it.		
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : - Extend :-		

Generalization: -
Normal Flow of Events: 1. The end-user chooses the New Booking Request submodule under the Booking Request module in the sidebar. 2. The end-user is redirected to the New Booking Request Input page. 3. The end-user needs input the booking reason and choose the right building to start booking. 4. S-1 is executed, followed by S-2. 5. In the process of executing S-2, V-1 will be executed. 6. The end-user inputs other relevant booking data (booking reason, capacity), chooses to submit at the end.
Sub-flows: **S-1: Select Venue Usage Timing** 1. The end-user needs to choose the starting date and end date for the booking event. 2. If the end-user has chosen an invalid date, then the end-user is warned with a popup of date not selected and the reason. 3. The end-user need to choose the starting time and end time for the booking event. 4. If the end-user has chosen an invalid time combination, then the end-user is warned with a popup of date not selected and the reason. **S-2: View and Select Venue/Classroom to book** 1. After the end-user has selected the right building to book, the end-user can surf between the floor plans in the mapping layout. 2. Clicking the venue/classroom will set the venue/classroom chosen for booking.
Alternate/Exceptional Flows: V-1: Booking Validation 1. If the end-user has a confirmed booking on the chosen venue on the filled time and date, then the venue's name would not be showing in the choosing list.

Table 4.7: Use Case of View Own Booking Requests

Use Case Name: View Own Booking Requests	**ID:** UC06	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to review own venue/classroom booking request. HOD - to review own venue/classroom booking request. Lecturers - to review own venue/classroom booking request. Students - to review own venue/classroom booking request.		
Brief Description: This use case describes how the identified end-users can review own venue/classroom booking request information		
Trigger: End-users navigates to the “Your Booking(s)” data table in the dashboard.		
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : - Extend :- Generalization: -		
Normal Flow of Events: 1. If the end-user has submitted a venue/classroom booking request, there will be rows of venue booking information. 2. For those approved yet uncompleted booking request, end-user could tap the row to learn more. 3. After tapping the selected row of request, the end-user is redirected to View Booking Details page that displays all relevant information for the selected booking.		

Sub-flows: -
Alternate/Exceptional Flows:-

Table 4.8: Use Case of View booking information in PDF report

Use Case Name: View booking information in PDF report	**ID:** UC07	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to view venue/classroom booking information in PDF report format. HOD - to view venue/classroom booking information in PDF report format. Lecturers - to view venue/classroom booking information in PDF report format. Students - to view venue/classroom booking information in PDF report format.		
Brief Description: This use case describes how the identified end-users access to each venue/classroom booking information in PDF report that the request had been completed/rejected or cancelled.		
Trigger: End-users navigates to the “Your Booking(s)” data table in dashboard.		
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : -		

Extend : - Generalization: -
Normal Flow of Events: 1. The end-user chooses the "Report" button that resides in the same row in the data table. 2. The end-user is redirected to another tab page, that shows the venue/classroom booking in PDF report. 3. The end-user could save the report. 4. If the end-user amends the booking ID link, then V-1 will be excuted.
Sub-flows: -
Alternate/Exceptional Flows: **V-1: Invalid Booking ID** 1. The end-user amends the booking ID in the URL. **2.** The end-user is prompted with a message "Invalid access.". **3.** The end-user is redirected to the dashboard page.

Table 4.9: Use Case of View UTAR calendar events

Use Case Name: View UTAR calendar events	**ID:** UC08	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to view information for a planned calendar event. HOD - to view information for a planned calendar event. Lecturers - to view information for a planned calendar event.		

Students - to view information for a planned calendar event.
Brief Description: This use case describes how the identified end-users can get the information, like the dates and the event name for the planned calendar events.
Trigger: End-users access the "Events" module in the sidebar.
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : - Extend : - Generalization: -
Normal Flow of Events: 1. The end-user is redirected to the View Event page. 2. The end-user selects the date, relevant information will be shown in the table underneath the calendar to get the end-user updated of the event information.
Sub-flows: -
Alternate/Exceptional Flows:-

Table 4.10: Use Case of Manage calendar events

Use Case Name: Manage calendar events	**ID:** UC09	**Importance Level:** High
Primary Actor: DGS staff	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to review and to add, edit or remove the calendar events.		
Brief Description:		

This use case describes how DGS staff add, edit or remove a planned calendar event.
Trigger: DSA staff access the “Manage Events” module hiding under the “Event” main tab in the sidebar.
Relationships: Association : DGS staff in UTAR Include : - Extend : - Generalization: -
Normal Flow of Events: 1. The DSA staff is redirected to a calendar page, which enables them to surf through dates. 2. If DSA staff wants to add a new calendar event, then S-1 will be executed. 3. If DSA staff wants to remove a calendar event, then S-2 will be executed. 4. If DSA staff wants to edit a calendar event information, then S-3 will be executed.
Sub-flows: **S-1: Add a new event** 1. The DSA staff selects a date. 2. The DSA staff enters event time and event reason for the selected date. 3. The DSA staff needs to choose to save it to add an event officially. **S-2: Remove a calendar event** 1. The DSA staff selects a specifically marked date. 2. The event details will be retrieved and shown on the table underneath the calendar. 3. The DSA staff needs to choose the delete button to clear the information. 4. The DSA staff is prompted to confirm the deletion. **S-3: Edit a calendar event information**

1. The DSA staff selects a specifically marked date. 2. The event details will be retrieved and shown on the table underneath the calendar. 3. The DSA staff needs to choose the "Edit" button. 4. The DSA staff is redirected to the Edit Event page to amend the calendar event details.
Alternate/Exceptional Flows: -

Table 4.11: Use Case of View all UTAR venues/classrooms

Use Case Name: View all UTAR venues/classrooms	**ID:** UC10	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students in UTAR	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to view a venue/classroom information. HOD - to view a venue/classroom information. Lecturers - to view a venue/classroom information. Students - to view a venue/classroom information.		
Brief Description: This use case describes how the identified end-users access to each venue/classroom to gain knowledge (room type, capacity count, previous booking histories) of it and to get aware of its availability status.		
Trigger: End-users access the "Venue/Classroom" module in the sidebar.		
Relationships: Association : DGS staff, HOD, lecturers and students in UTAR Include : - Extend : - Generalization: -		

Normal Flow of Events: 1. The end-user chooses the which building to activate the identical floor plan mappings. 2. The end-user can surf through the mappings to view the venues/classrooms in each floor. 3. If the user wishes to know more about a particular venue/classroom, then S-1 is executed.
Sub-flows: S-1: Particular Venue/Classroom Viewing 1. User selects the venue/classroom in the floor plan mapping. 2. The user is redirected to another page which lists the detailed information for the venue/classroom selected. 3. The user can return to the previous page by selecting the embedded return link, continuing the Normal Flow Step 2.
Alternate/Exceptional Flows:-

Table 4.12: Use Case of Manage all venues/classrooms' information

Use Case Name: Manage all venues/classrooms' status	**ID:** UC11	**Importance Level:** High
Primary Actor: DGS staff	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to perform necessary updates on a particular UTAR venue/classroom.		
Brief Description: This use case describes how DGS staff get access and to update the information for a chosen venue/classroom in both UTAR buildings.		
Trigger: DGS staff access the "Venues/Classrooms" module in the sidebar.		

Relationships: Association : DGS staff in UTAR Include : - Extend : - Generalization: -
Normal Flow of Events: 1. UC06 is executed. 2. If DGS staff intends to make amendments on the venue/classroom information, the staff needs to choose the “Edit” button. 3. DGS staff could edit the facilities column or amend the availability status by toggling the selection in the dropbox. 4. A prompt message “Venue details updated successfully!” is shown, hinting the staff that the information had been updated. 5. The page is being refreshed.
Sub-flows: -
Alternate/Exceptional Flows: -

Table 4.13: Use Case of Manage trimester information

Use Case Name: Manage trimester information	**ID:** UC12	**Importance Level:** High
Primary Actor: DGS staff	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to perform information amendments on the trimester information.		
Brief Description:		

This use case describes how DGS staff get access and to update the information for a chosen trimester.

Trigger:

DGS staff access the Class Scheduling tab in the sidebar.

Relationships:

Association	: DGS staff in UTAR
Include	: -
Extend	: -
Generalization: -	

Normal Flow of Events:

1. If DGS staff wishes to view all added trimester information, he/she will need to access the "View Trimester" under the Class Scheduling tab, S-1 will be executed.
2. If DGS staff wishes to add new trimester information, he/she will need to access the "Add Trimester" under the Class Scheduling tab, S-2 will be executed.
3. If DGS staff wishes to delete a trimester, he/she will need to access the "View Trimester" under the Class Scheduling tab, S-3 will be executed.
4. If DGS staff wishes to edit a trimester's information, he/she will need to access the "View Trimester" under the Class Scheduling tab, S-4 will be executed.

Sub-flows:

S-1: View trimester information

1. DGS staff is shown with data table, listing all added trimester.
2. DGS staff could filter the trimesters by selecting the studying level on the tabs, or just search it in the filter box.
3. DGS staff could tap a trimester row to know the full details of a trimester.
4. DGS staff is redirected to the Edit Trimester page to read all details.

S-2: Add trimester information

1. DGS staff is redirected to Add Trimester page.

2. DGS staff needs to fill the starting date, ending date, the study level and the trimester count. 3. The page is being refreshed with message “Trimester added” is shown to the DGS staff, if the trimester has not been added before. **S-3: Delete trimester** 1. DGS staff is shown with data table, listing all added trimester. 2. DGS staff could select the delete button for each trimester. 3. A prompt message is shown, confirming if the DGS staff wants to delete the selected trimester. 4. If DGS staff chooses “Yes”, then the trimester will be deleted from the database, the page refreshes. 5. If DGS staff chooses “No”, then the trimester will be remaining in the database, and the page remains static. **S-4: Edit trimester information** 1. Execute S-1. 2. DGS staff could only edit the starting date and the ending date of the selected trimester. 3. DGS staff choose the “Save” button once the editing has done. 4. Message “Trimester updated successfully” is shown.
Alternate/Exceptional Flows: -

Table 4.14: Use Case of Manage class information

Use Case Name: Manage class information	**ID:** UC13	**Importance Level:** High
Primary Actor: DGS staff	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to perform information amendments on the class information.		
Brief Description: This use case describes how DGS staff get access and to update the information for a class.		

Trigger:

DGS staff access the Class Scheduling tab in the sidebar.

Relationships:

Association	: DGS staff in UTAR
Include	: -
Extend	: -
Generalization	: -

Normal Flow of Events:

1. If DGS staff wishes to view all added class information, he/she will need to access the "View Scheduled Class" under the Class Scheduling tab, S-1 will be executed.
2. If DGS staff wishes to add new class information, he/she will need to access the "Add Class" under the Class Scheduling tab, S-2 will be executed.
3. If DGS staff wishes to delete a class, he/she will need to access the "View Scheduled Class" under the Class Scheduling tab, S-3 will be executed.
4. If DGS staff wishes to edit a class information, he/she will need to access the "View Class" under the Class Scheduling tab, S-4 will be executed.

Sub-flows:

S-1: View class information

1. DGS staff would need to input the added class name.
2. The result of similar class names will be displayed on the table.
3. DGS staff could view the venue's name for the class, and the date and time values.

S-2: Add class information

1. DGS staff is redirected to Add Class page.
2. DGS staff needs to select the study level, then chooses the right trimester name from the dropdown list.
3. Once the trimester is selected, then the DGS staff proceeds to select the faculty name and the course name under the chosen faculty.

4. DGS staff proceeds to select the weekday and the class mode, whether is it weekly, or having it only on even weeks or odd weeks.
5. DGS staff chooses the starting time and ending time for the class.
6. DGS staff chooses the venue/classroom to run the class.
7. DGS staff chooses "Submit" button at the end.
8. The page refreshes, with the message "Class added successfully" if there are no venue booking requests that clashes with the class timings.

S-3: Delete class

1. S-1 is executed.
2. DGS staff could select the delete button for each class.
3. A prompt message is shown, confirming if the DGS staff wants to delete the selected class.
4. If DGS staff chooses "Yes", then the class session will be deleted, along with the venue booking session to be deleted from the database, the page refreshes.
5. If DGS staff chooses "No", then the class session is unchanged and the venue booking session remains in the database, and the page remains static.

S-4: Edit class information

1. Execute S-1.
2. DGS staff needs to select the "Edit" button for each class row.
3. DGS staff could select the class date and the timings only.
4. Should any chosen timing is clashing with an existing venue booking request, then the DGS staff is prompt with a message that the venue is being occupied.
5. DGS staff could select "Save" button if there are no validation messages shown after editing.
6. The page refreshes, with a message "Class is updated successfully" is shown.

Alternate/Exceptional Flows:

-

Table 4.15: Notified through SMS and Email Use Case

Use Case Name: Notified through SMS and Email	**ID:** UC14	**Importance Level:** High
Primary Actor: DGS staff, HOD, lecturers and students	**Use Case Type:** Real, Detail	
Stakeholders and Interests: DGS staff - to get notified through SMS and Email when any of the booking requests has been updated. HOD- to get notified through SMS and Email when any of the booking requests has been updated. Lecturers- to get notified through SMS and Email when any of the booking requests has been updated. Students- to get notified through SMS and Email when any of own booking requests has been updated.		
Brief Description: This use case depicts the process of every level's end-users gets SMS message or Email notification when the venue/classroom request is being updated.		
Trigger: DGS staff, HOD: a new venue/classroom booking request was submitted. Lecturer: a new venue/classroom booking request was submitted by a student and the lecturer's name was chosen. Student: approval/rejection/updates on own venue/classroom booking request.		
Relationships: Association : - Include : - Extend : - Generalization: -		
Normal Flow of Events: 1. Should any parties update the venue booking information, then the action would trigger the notification function to run.		

2. Based on user tracing, the email and SMS would be sent to the involved end-user.
Sub-flows: -
Alternate/Exceptional Flows: -

4.5 Interface Flow Diagram

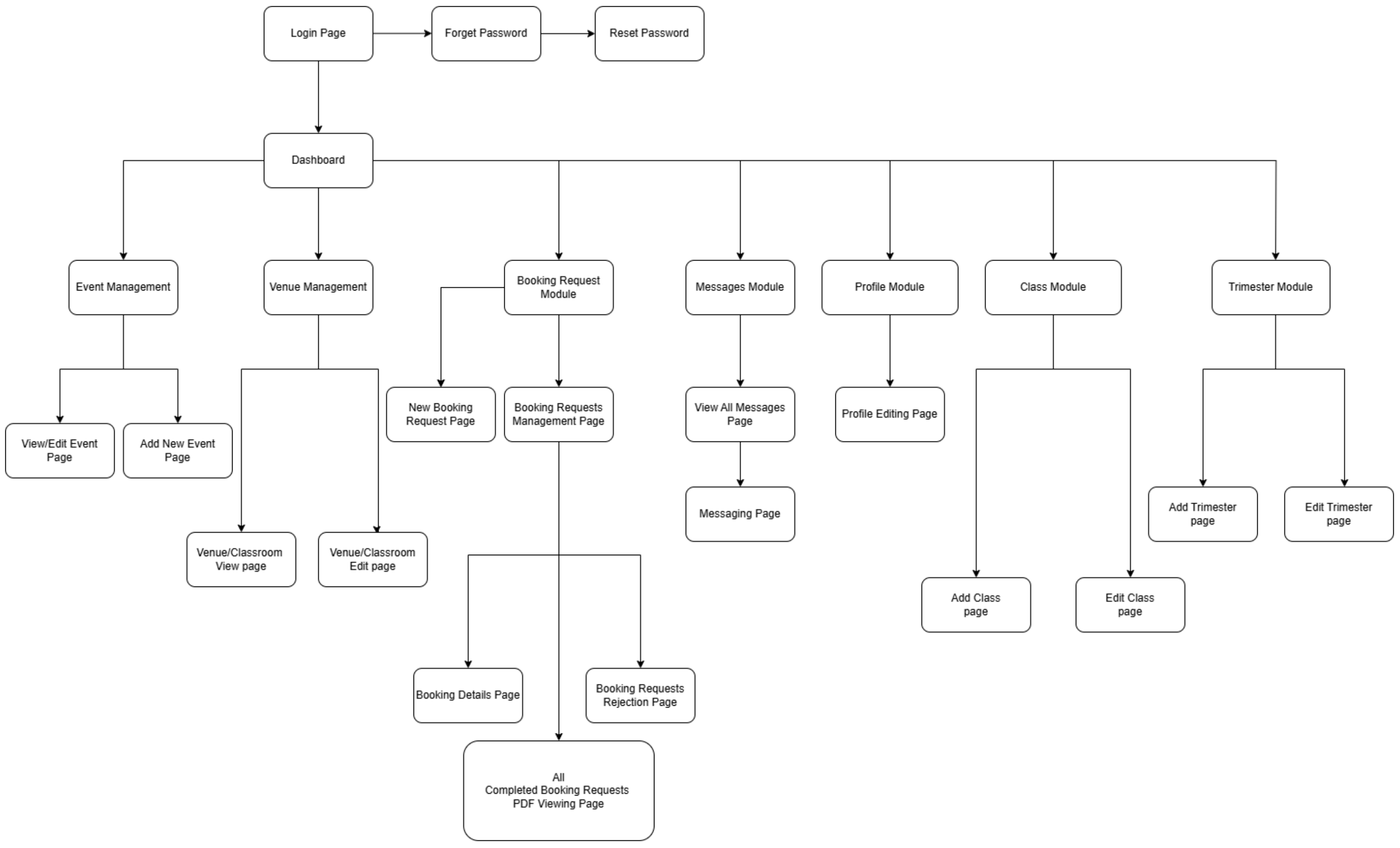


Figure 4.9: Interface Flow Diagram for the DGS staff and HOD users in the UTAR Venue/Classroom Booking Web Portal

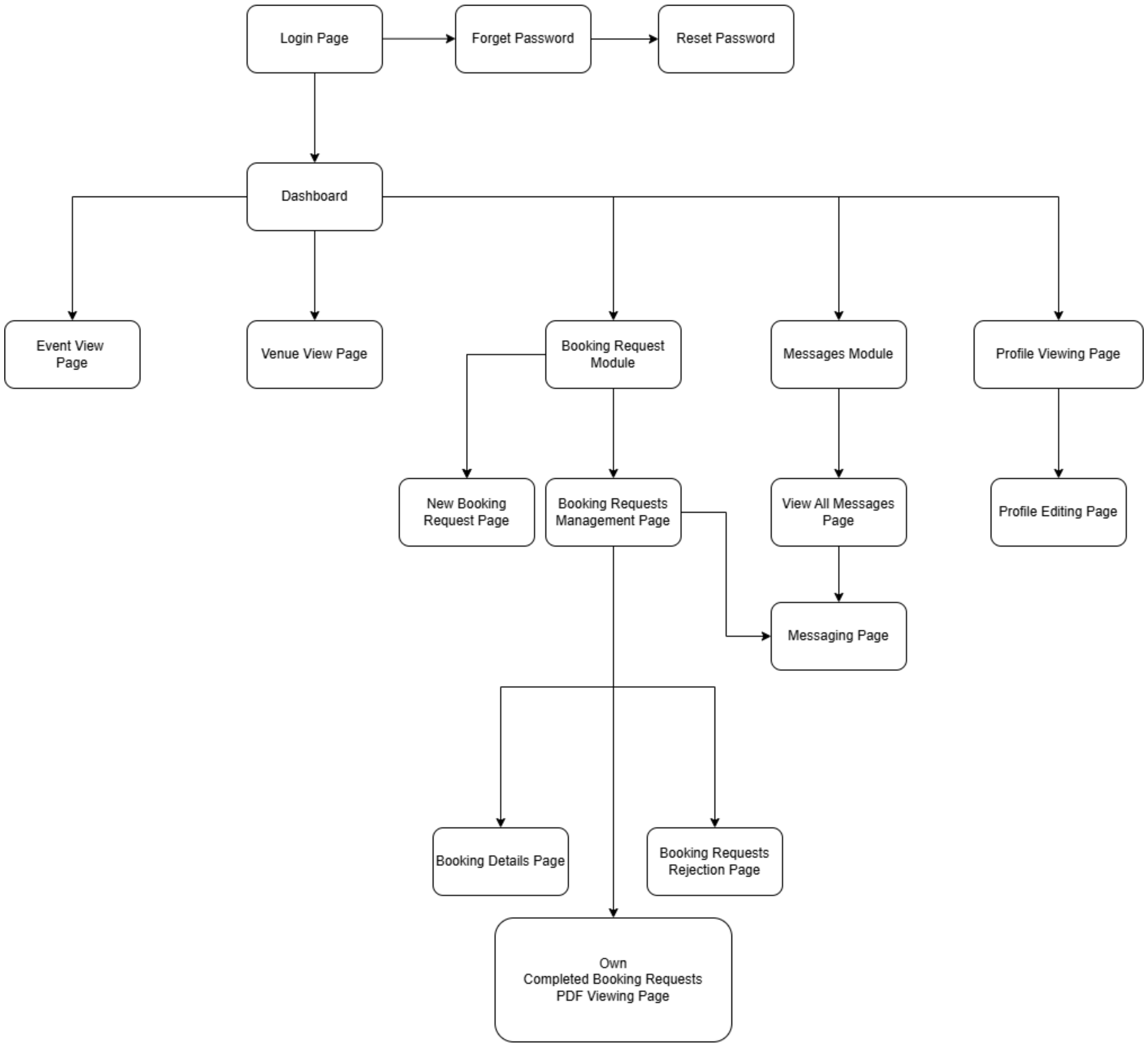


Figure 4.10: Interface Flow Diagram for the non-admin users in the UTAR Venue/Classroom Booking Web Portal

4.6 Prototype Interface

1. Logging in Module

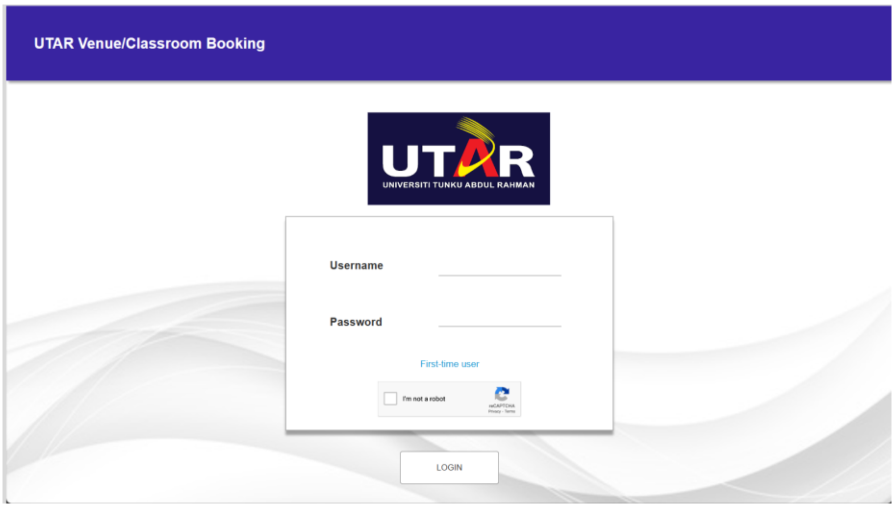
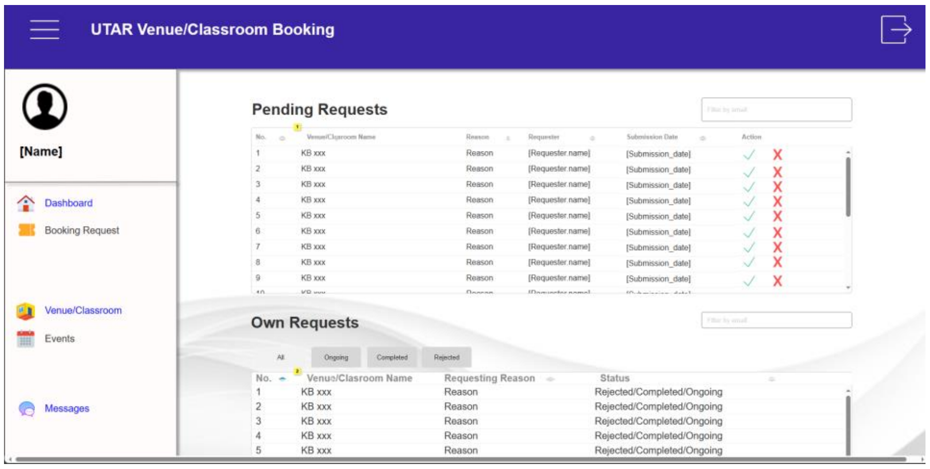


Figure 4.11: Login Page

2. Dashboard



Pending Requests

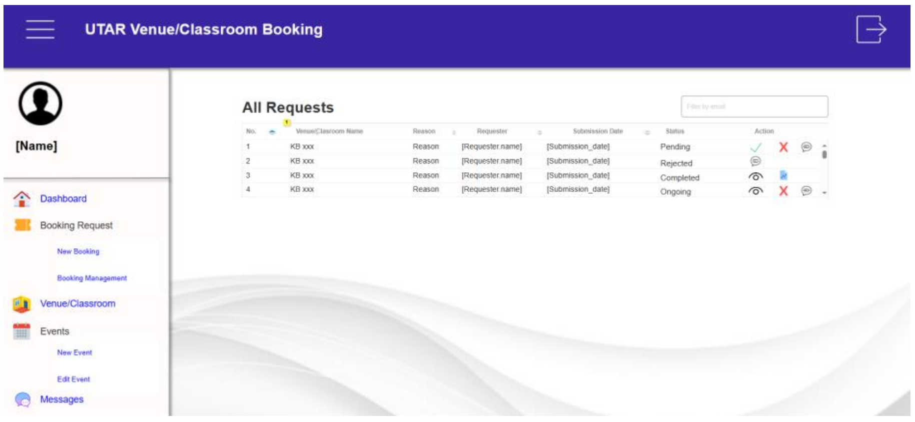
No.	Venue/Claroom Name	Reason	Requester	Submission Date	Action
1	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
2	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
3	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
4	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
5	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
6	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
7	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
8	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗
9	KB xxx	Reason	[Requester.name]	[Submission_date]	✓ ✗

Own Requests

No.	Venue/Clasroom Name	Requesting Reason	Status
1	KB xxx	Reason	Rejected/Completed/Ongoing
2	KB xxx	Reason	Rejected/Completed/Ongoing
3	KB xxx	Reason	Rejected/Completed/Ongoing
4	KB xxx	Reason	Rejected/Completed/Ongoing
5	KB xxx	Reason	Rejected/Completed/Ongoing

Figure 4.12: Dashboard

3. Booking Management

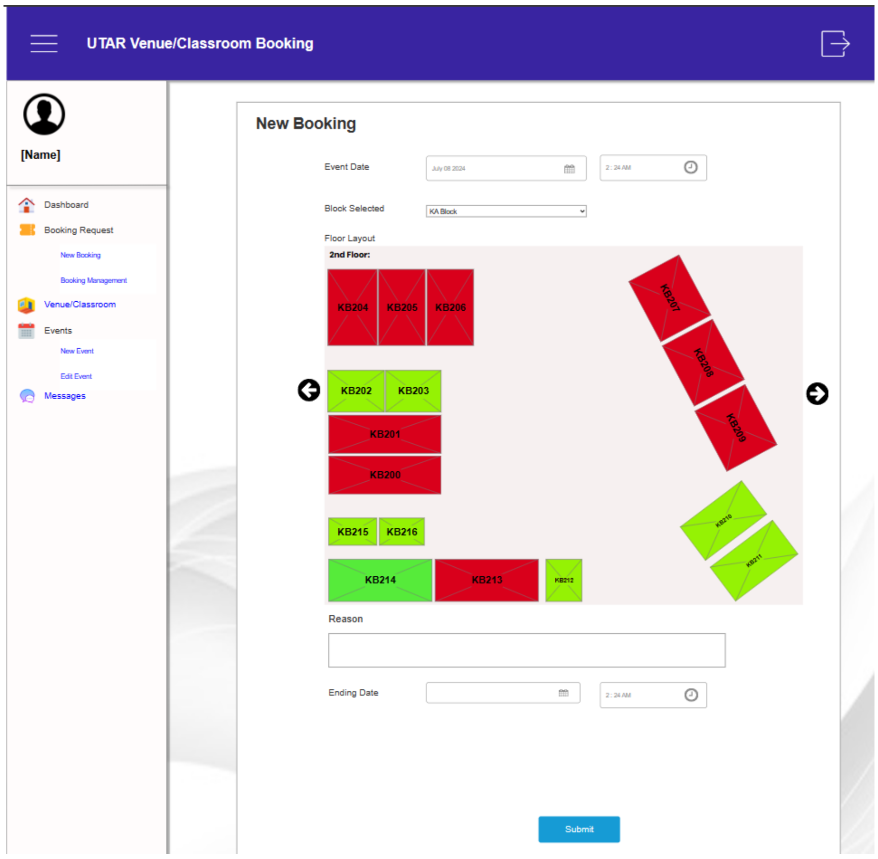


All Requests

No.	Venue/Clasroom Name	Reason	Requester	Submission Date	Status	Action
1	KB xxx	Reason	[Requester.name]	[Submission_date]	Pending	✓ ✗ 💬
2	KB xxx	Reason	[Requester.name]	[Submission_date]	Rejected	💬
3	KB xxx	Reason	[Requester.name]	[Submission_date]	Completed	👁
4	KB xxx	Reason	[Requester.name]	[Submission_date]	Ongoing	👁 ✗ 💬

Figure 4.13: Booking Management Page

4. New Booking Page



New Booking

Event Date July 08 2024 2 : 24 AM

Block Selected KA Block

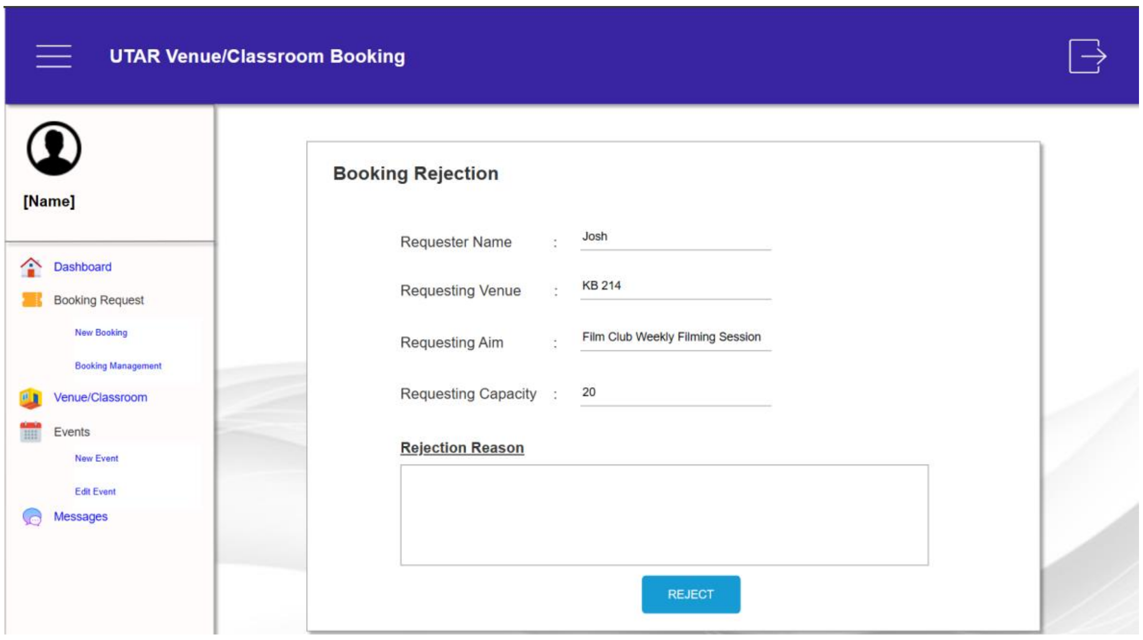
Floor Layout

Reason

Ending Date 2 : 24 AM

Figure 4.14: New Booking Page

5. Booking Rejection Page (only for DGS and HOD/faculty staff)



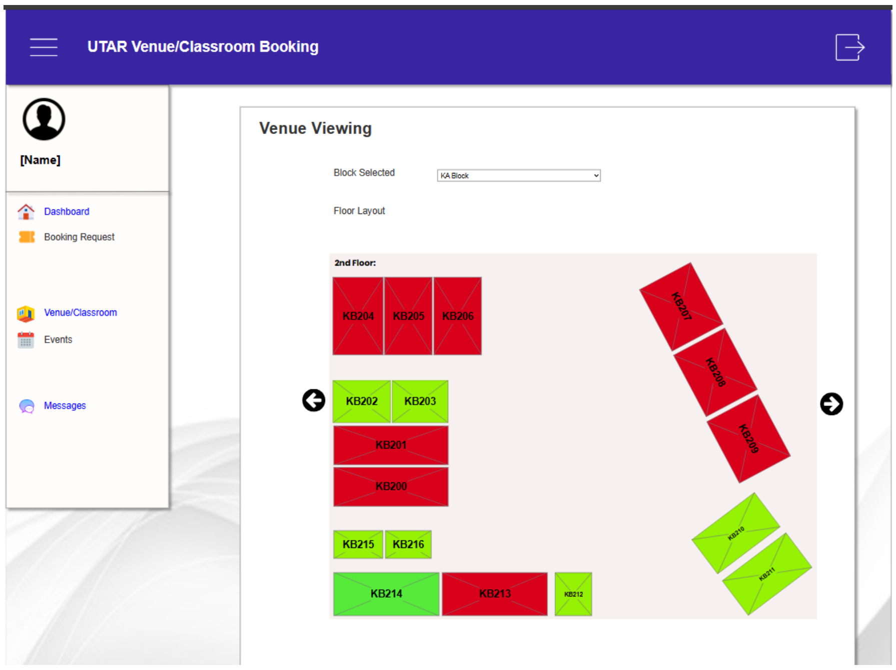
Booking Rejection

Requester Name	:	Josh
Requesting Venue	:	KB 214
Requesting Aim	:	Film Club Weekly Filming Session
Requesting Capacity	:	20

Rejection Reason

Figure 4.15: Booking Rejection Page

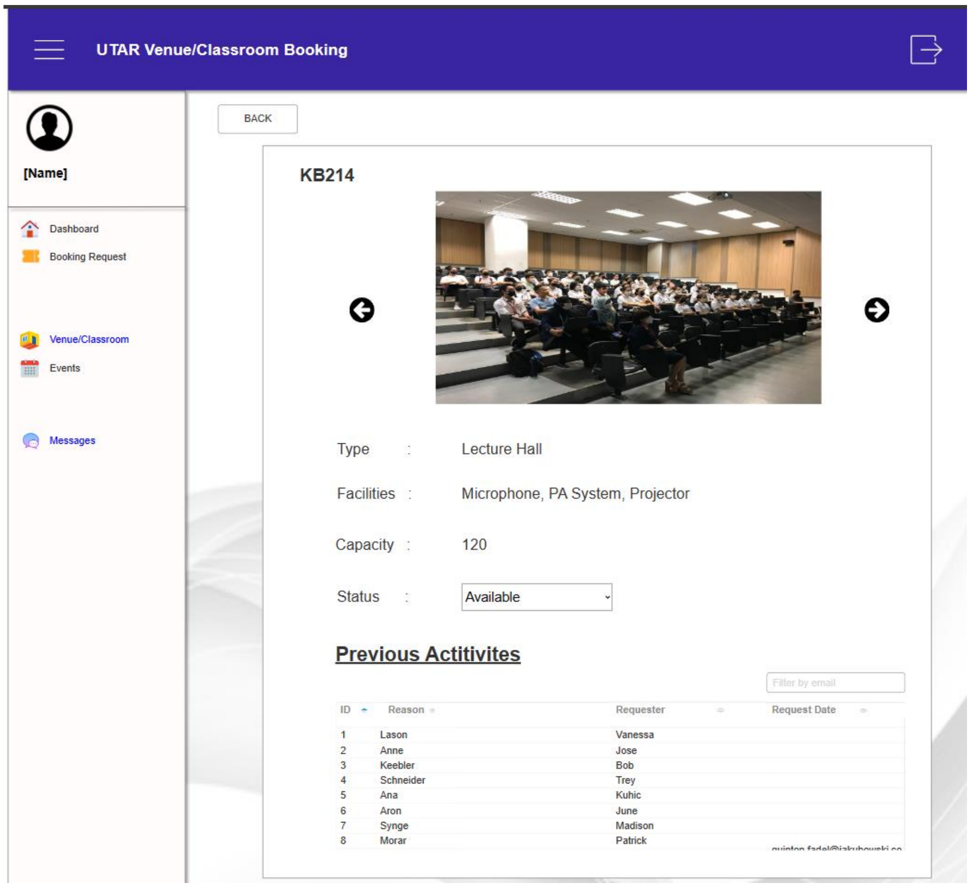
6. Venue Viewing Page



Block Selected: KA Block

Figure 4.16: Venue Viewing Page

7. Venue Details Page



KB214

Type	:	Lecture Hall
Facilities	:	Microphone, PA System, Projector
Capacity	:	120
Status	:	Available

Previous Actitivites

ID	Reason	Requester	Request Date
1	Lason	Vanessa	
2	Anne	Jose	
3	Keebler	Bob	
4	Schneider	Trey	
5	Ana	Kuhic	
6	Aron	June	
7	Synge	Madison	
8	Morar	Patrick	

Figure 4.17: Venue Details Page

8. New Event Page (only for DGS and HOD/faculty staff)

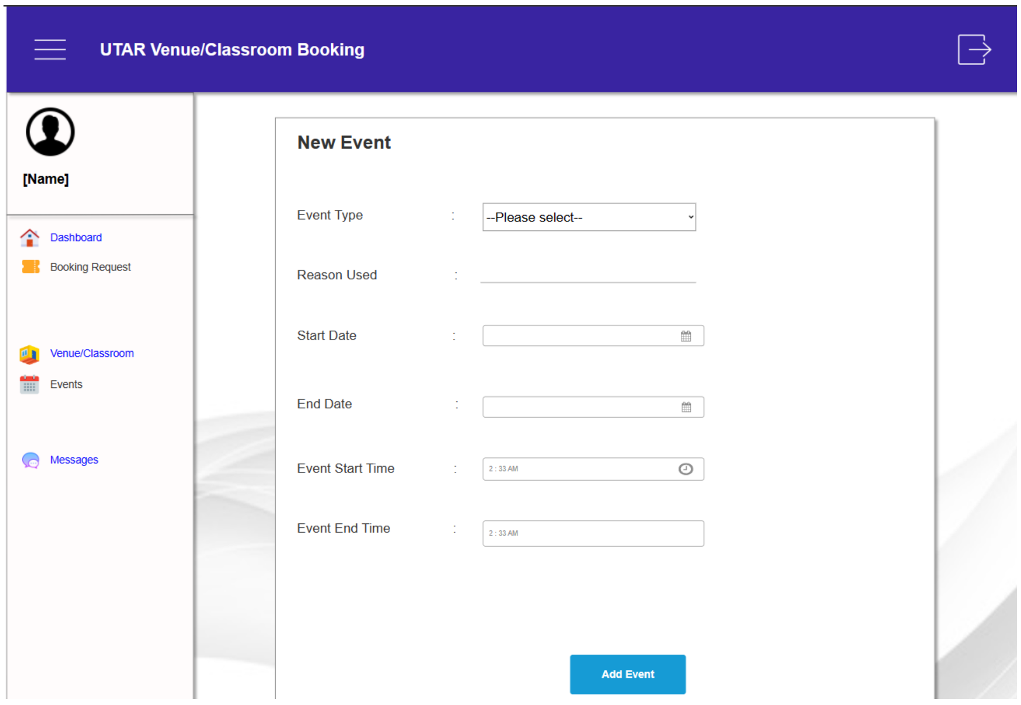


Figure 4.18: New Event Page

9. Event Viewing Page

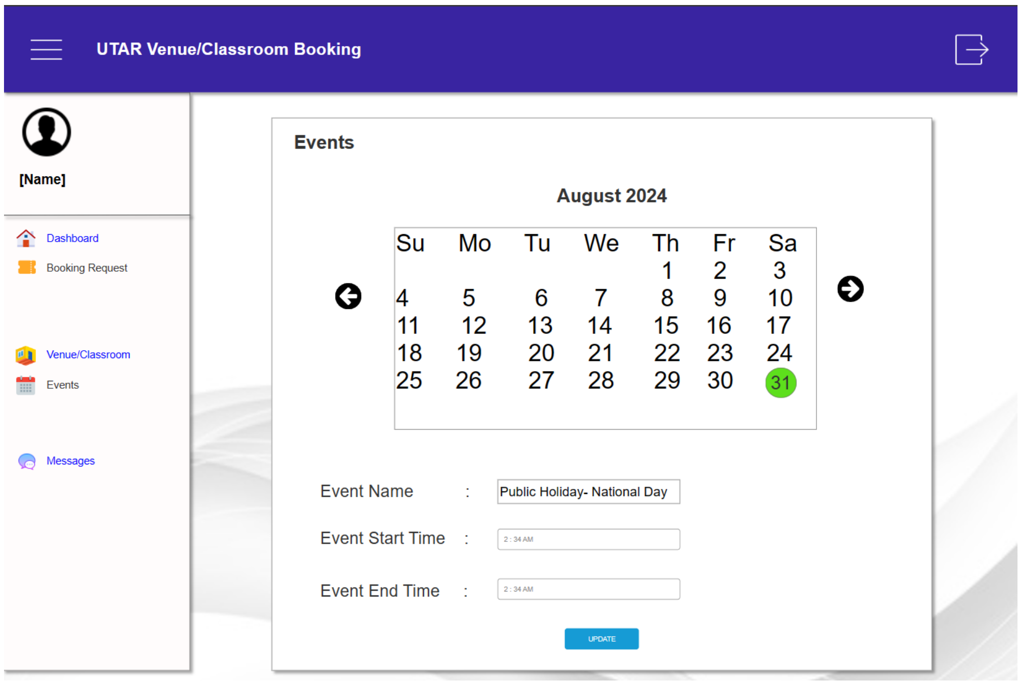
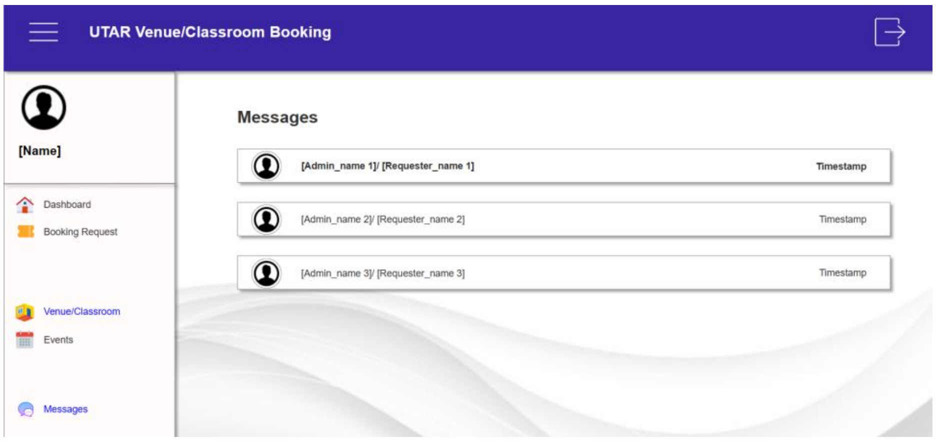


Figure 4.19: Event Viewing Page

10. All Messages Page



Messages

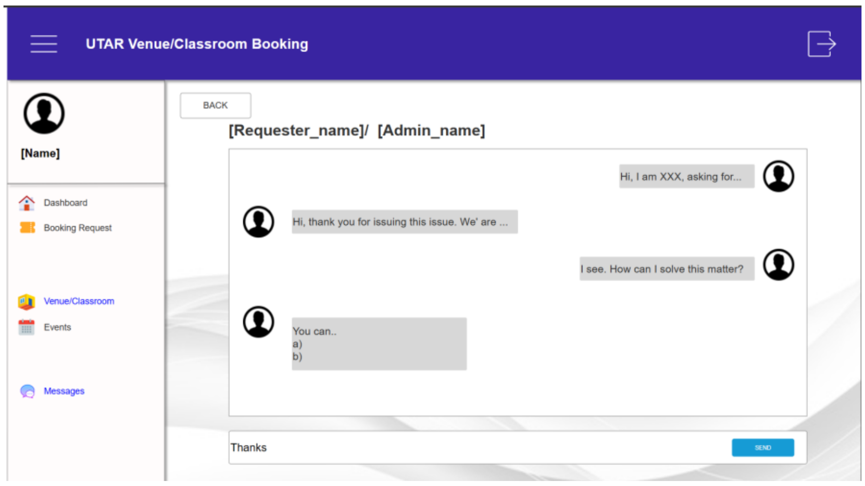
[Admin_name 1]/ [Requester_name 1] Timestamp

[Admin_name 2]/ [Requester_name 2] Timestamp

[Admin_name 3]/ [Requester_name 3] Timestamp

Figure 4.20: All Messages Page

11. Messaging Module



[Requester_name]/ [Admin_name]

Hi, I am XXX, asking for...

Hi, thank you for issuing this issue. We' are ...

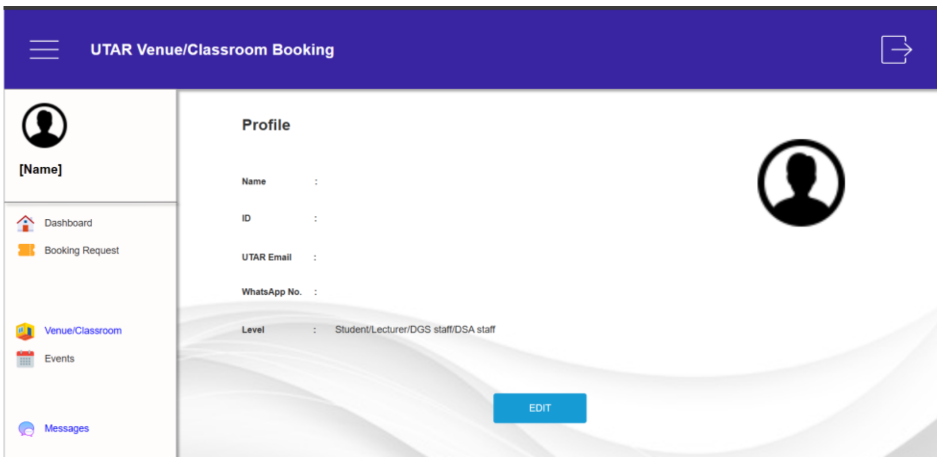
I see. How can I solve this matter?

You can..
a)
b)

Thanks

Figure 4.21: Messaging Page

12. Profile Viewing Page

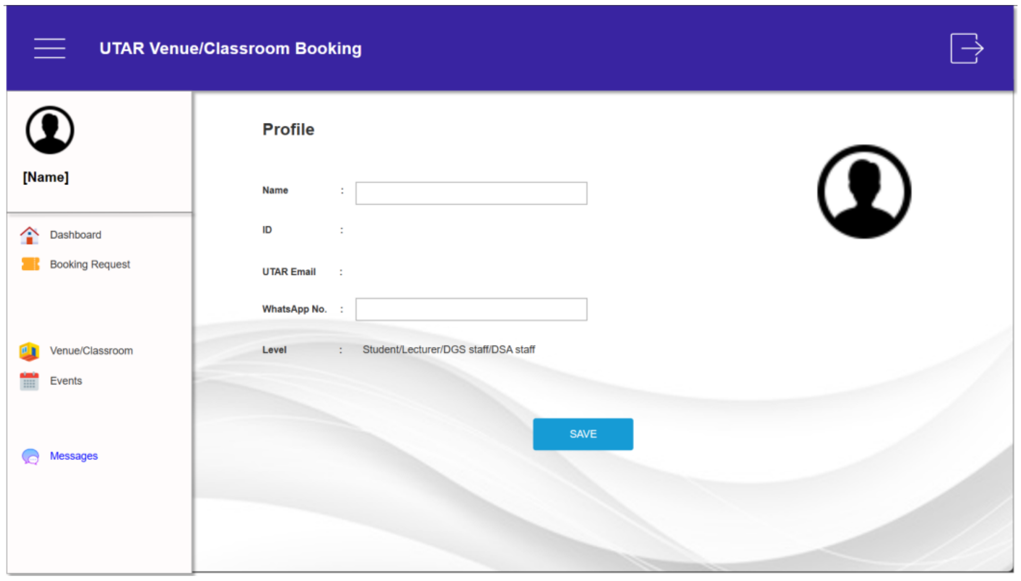


Profile

Name	:	
ID	:	
UTAR Email	:	
WhatsApp No.	:	
Level	:	Student/Lecturer/DGS staff/DSA staff

Figure 4.22: Profile Page

13. Profile Editing Page



Profile

Name	:	
ID	:	
UTAR Email	:	
WhatsApp No.	:	
Level	:	Student/Lecturer/DGS staff/DSA staff

Figure 4.23: Profile Editing Page

4.7 Summary

This chapter provided a technical view of the UTAR Venue/Classroom Booking Web Portal. The functional and non-functional requirements had stated all attributes and the processes to achieve them, getting the end-users prepared for how the system would perform at the end. Snapshots from the prototype were included as well in this chapter to assist them in having a glance at how the end system could look like, also giving the project author a preliminary concept on constructing the system. With this to be said, this chapter acts as a reference for the project author in development, after summarizing all requirements gathered.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

This chapter delineates the system architecture of the UTAR Venue/Classroom Booking Web Portal, which is inclusive of frontend and backend frameworks used. Besides that, levels of data flow diagram will be included to show the overall flow of information across the entire system. Database schema which serves as the data handling backbone will be described as well in another section.

5.2 System Architecture Design

The UTAR Venue/Classroom Booking Web Portal implements the three-tier architecture to maintain the layout. In short, it is an architecture which decomposes an application into three layers that are described as the following. The Presentation layer is the interface part where the users could interact with the elements built, like the forms and the data shown. Though the users interact, the application layer consists of the business logic of the entire system, to perform any processing or validation based on the users' actions. Any third-party libraries also lie here as it could be a core part of the system. Since the data needs to be kept, there is a data layer linking with the application layer to keep track of the information flow, whether to store, amend and process the data according to the business logics coded. The following diagram depicts the implementation of the architecture in this system developed.

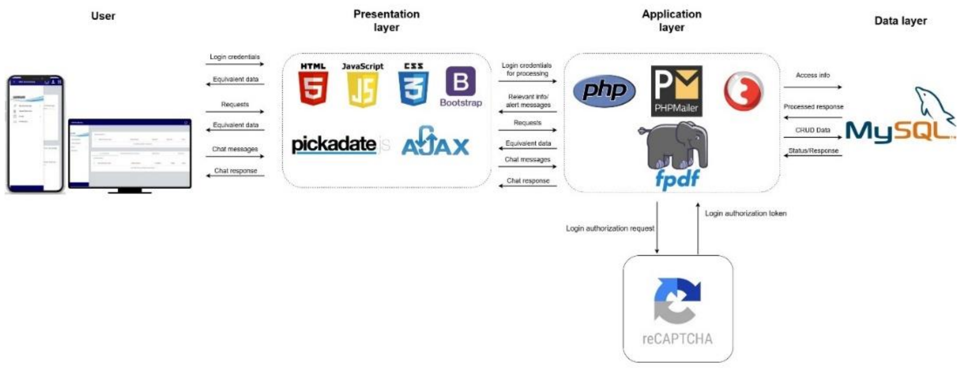


Figure 5.1: Three-tier architecture for UTAR Venue/Classroom Booking Web Portal

Being designed in the way that users could use the system through mobile devices or computers, the UTAR Venue/Classroom Booking Web Portal’s presentation layer is presented through HTML format, filled with Bootstrap elements that are customized with the help of CSS. Since the elements could be changed according to the users’ actions, this could be achieved through JavaScript. Working with JavaScript, AJAX could change the data presented asynchronously by working with the application layer and data layer. Meaning to say, this reduces the need of the user to refresh the page to get updated. To quote an example, XMLHttpRequest is adopted in JavaScript for the chatting function so that the chat messages could be sent and retrieved between the layers with ease. PickaDate.js is a third-party library used here, to reshape the date input elements instead of the traditional layouts.

The application layer is the business logics coded with PHP. Since this application is sensitive to logged-in user’s identity, validation needs to be done by working along with the data layer to check the user’s role before accessing the function. Based on the data retrieved, PHP could respond by either allowing the user to proceed or stopping the user.

Each user has a unique identity and it has to be handled with care when it comes to logging into the system. To secure user session, ReCAPTCHA form checking is implemented in this case. This measure is taken to prevent automated malicious attacks, as it works by requiring the user to choose the right pictures that match the title given. With all the right pictures chosen, it grants a valid Boolean to the authentication mechanism, allowing the users to log in.

While this system has communication features to notify the users via email, PHPMailer is implemented in this case. Though there are scenarios to send email at times, it requires a clear cut of situations to trigger this function. Working along with the data tier, it needs to get the right port number of the system first so that it could send an email to the fetched user’s email address. Regarding the SMS, it works by implementing the Bulk360 SMS API. The working logics is similar to the emailing function, just that it requires the system owner to register for a Bulk360 account, and it needs to be installed in the project directory through npm package installing method.

Another notable library is the FPDF, which works with PHP to generate PDF reports. Technically, it comprises of the functions to generate a PDF report by allowing the project author to define the document properties, such as the column, spacing, font styles and the text to be displayed. Considering that one of the functions in the system is to generate venue usage report, this library is chosen as the provider for this service.

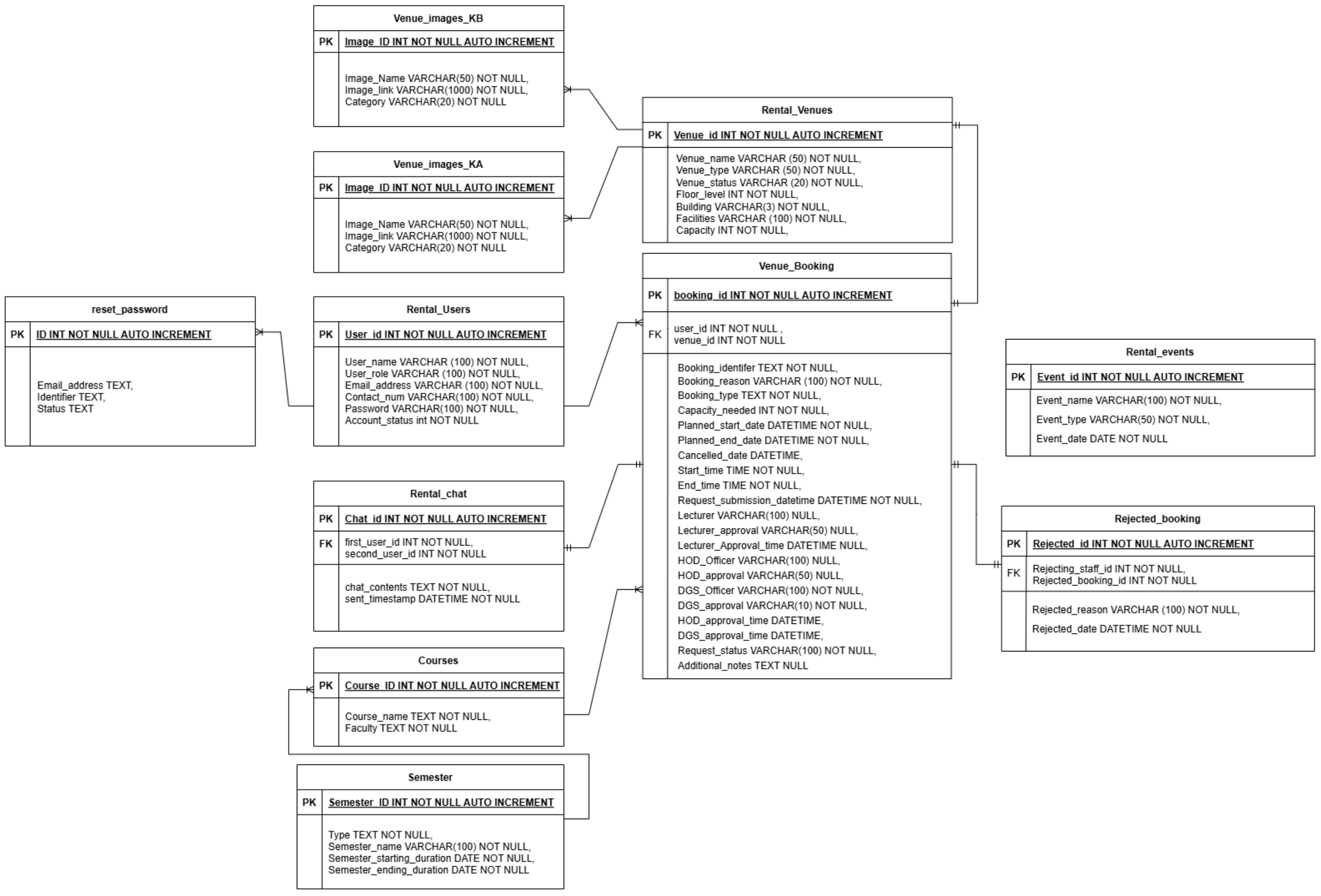
Mentioning that the system for this project is hosted locally through XAMPP, phpMyAdmin is readily available as the data layer. Working on the basis of SQL commands, the learning curve is low compared to real-time databases. For this system, it stores the information for the venue booking details, the users' information, chatting history and so on. Data retrieval has played a major role in this system as users' role are prioritized when it comes to validation. Other than that, this database approach is useful enough for data transactions as well.

5.3 Database Architecture

Being implemented in phpMyAdmin, there are ten tables built for the system developed. Known as one of the relational databases, this section will express the exact database structure that relies in the Data Layer.

5.3.1 Database Schema

In this system, most of the tables share the same attribute, showing the real-life interaction of a single entity. Known as multiplicity, a single entity can have different properties. For example, a user can submit multiple venue booking requests at a time. The following diagram depicts the database schema for the UTAR Venue/Classroom Booking Web Portal.



Venue_images_KB
PK Image_ID INT NOT NULL AUTO INCREMENT
Image_Name VARCHAR(50) NOT NULL, Image_link VARCHAR(1000) NOT NULL, Category VARCHAR(20) NOT NULL
Venue_images_KA
PK Image_ID INT NOT NULL AUTO INCREMENT
Image_Name VARCHAR(50) NOT NULL, Image_link VARCHAR(1000) NOT NULL, Category VARCHAR(20) NOT NULL
Rental_Venues
PK Venue_id INT NOT NULL AUTO INCREMENT
Venue_name VARCHAR (50) NOT NULL, Venue_type VARCHAR (50) NOT NULL, Venue_status VARCHAR (20) NOT NULL, Floor_level INT NOT NULL, Building VARCHAR(3) NOT NULL, Facilities VARCHAR (100) NOT NULL, Capacity INT NOT NULL,
reset_password
PK ID INT NOT NULL AUTO INCREMENT
Email_address TEXT, Identifier TEXT, Status TEXT
Rental_Users
PK User_id INT NOT NULL AUTO INCREMENT
User_name VARCHAR (100) NOT NULL, User_role VARCHAR (100) NOT NULL, Email_address VARCHAR (100) NOT NULL, Contact_num VARCHAR(100) NOT NULL, Password VARCHAR(100) NOT NULL, Account_status int NOT NULL
Venue_Booking
PK booking_id INT NOT NULL AUTO INCREMENT
FK user_id INT NOT NULL , venue_id INT NOT NULL
Booking_identifer TEXT NOT NULL, Booking_reason VARCHAR (100) NOT NULL, Booking_type TEXT NOT NULL, Capacity_needed INT NOT NULL, Planned_start_date DATETIME NOT NULL, Planned_end_date DATETIME NOT NULL, Cancelled_date DATETIME, Start_time TIME NOT NULL, End_time TIME NOT NULL, Request_submission_datetime DATETIME NOT NULL, Lecturer VARCHAR(100) NULL, Lecturer_approval VARCHAR(50) NULL, Lecturer_Approval_time DATETIME NULL, HOD_Officer VARCHAR(100) NULL, HOD_approval VARCHAR(50) NULL, DGS_Officer VARCHAR(100) NOT NULL, DGS_approval VARCHAR(10) NOT NULL, HOD_approval_time DATETIME, DGS_approval_time DATETIME, Request_status VARCHAR(100) NOT NULL, Additional_notes TEXT NULL
Rental_events
PK Event_id INT NOT NULL AUTO INCREMENT
Event_name VARCHAR(100) NOT NULL, Event_type VARCHAR(50) NOT NULL, Event_date DATE NOT NULL
Rental_chat
PK Chat_id INT NOT NULL AUTO INCREMENT
FK first_user_id INT NOT NULL, second_user_id INT NOT NULL
chat_contents TEXT NOT NULL, sent_timestamp DATETIME NOT NULL
Rejected_booking
PK Rejected_id INT NOT NULL AUTO INCREMENT
FK Rejecting_staff_id INT NOT NULL, Rejected_booking_id INT NOT NULL
Rejected_reason VARCHAR (100) NOT NULL, Rejected_date DATETIME NOT NULL
Courses
PK Course_ID INT NOT NULL AUTO INCREMENT
Course_name TEXT NOT NULL, Faculty TEXT NOT NULL
Semester
PK Semester_ID INT NOT NULL AUTO INCREMENT
Type TEXT NOT NULL, Semester_name VARCHAR(100) NOT NULL, Semester_starting_duration DATE NOT NULL, Semester_ending_duration DATE NOT NULL

Figure 5.2: Database Schema

5.3.2 Database Dictionary

This section provides a detailed description of each database table and its attributes. Do refer to Figure 5.2 to know the naming for each table.

5.3.2.1 Venue booking requests table

Being named as Venue_Booking table, this table is responsible for storing venue/classroom booking request details. To ensure the uniqueness of each booking request established, the venue selected and the usage starting and ending time with date values formed a chain by enabling them to become unique key, preventing duplicated entries as well.

Table 5.1: Data dictionary for Venue_Booking table

Name of attribute	**Representation**	**Key**	**Type**	**Default Value**

booking_id	The identity number for each specific booking request.	Primary	Integer, not null	Auto incremental
user_id	The user ID who has submitted the booking request.	Foreign	Integer, not null	-
venue_id	The venue ID selected for booking.	Foreign	Integer, not null	-
Booking_identifier	A randomly generated string, used to identify each specific booking request.	-	Text (11), not null	-
Booking_reason	Reason to use the venue.	-	Varchar(100), not null	-
Booking_type	To differentiate whether is the request class events or not.	-	Text, not null	-
Capacity_needed	The expected capacity of participants for the venue usage.	-	Integer, not null	-

Planned_start_date	The starting date for the venue usage.	-	DateTime, not null	-
Planned_end_date	The ending date for the venue usage.	-	DateTime, not null	-
Cancelled_date	The date if the booking request gets cancelled.	-	DateTime	-
Start_time	Starting time for the venue usage.	-	Time, not null	-
End_time	Ending time for the venue usage.	-	Time, not null	-
Request_submission _datetime	The timestamp where the venue booking request is submitted	-	DateTime, not null	-
Lecturer	Selected lecturer's name if student submitted a request.	-	Varchar (100), null	-
Lecturer_approval	Selected lecturer's approval decision on the request.	-	Varchar(50), null	-
Lecturer_approval_time	The timestamp where the selected	-	DateTime, null	-

	lecturer decide on the request.			
HOD_Officer	HOD staff name if any of them approves/rejects a request.	-	Varchar(100), null	-
HOD_approval	HOD's decision on the request.	-	Varchar(50), null	-
DGS_Officer	DGS staff name if any of them approves/rejects a request.	-	Varchar(100), not null	-
DGS _approval	DGS's decision on the request.	-	Varchar(50), not null	-
HOD_approval_time	The timestamp where the DGS staff decide on the request.	-	DateTime	-
DGS_approval_time	The timestamp where the DGS staff decide on the request.	-	DateTime	-
Request_status	Concluded status based on lecturer, HOD, DGS on the request.	-	Varchar(100), not null	-
Additional_notes	Add-ons by the DGS or HOD once the request is approved but to complete.	-	Text,null	-

5.3.2.2 Venues table

Being named as Rental_Venues table, this table is responsible for storing details for each venue/classroom.

Table 5.2: Data dictionary for Rental_Venues table

Name of attribute	**Representation**	**Key**	**Type**	**Default Value**
Venue_id	The identity number for each specific venue/classroom.	Primary	Integer, not null	Auto incremental
Venue_name	Naming for each venue/classroom.	-	Varchar(50), not null	-
Venue_type	Indicator for the type of venue/classroom.	-	Varchar(50), not null	-
Venue_status	Availability status of the venue/classroom.	-	Varchar(20), not null	-
Floor_level	The floor level where the venue/classroom is located at.	-	Integer, not null	-
Building	The building name where the venue/classroom is located at.	-	Varchar(3), not null	-
Facilities	The facilities that resides in the venue/classroom.	-	Varchar(100), not null	-

Capacity	The capacity supported by the venue/classroom.	-	Integer, not null	-

5.3.2.3 Venues' images table

Being named as Venue_images_*block name*, this table is responsible for storing pictures for each building block.

Table 5.3: Data dictionary for Venue_images_KA and Venue_images_KB table

Name of attribute	**Representation**	**Key**	**Type**	**Default Value**
Image_ID	The identity number for each specific image.	Primary	Integer, not null	Auto incremental
Image_Name	The name for each image.	-	Varchar(50), not null	-
Image_link	The URL link that stores the image.	-	Varchar(1000), not null	-
Category	The building block's name where the image belongs to.	-	Varchar(20), not null	-

5.3.2.4 User table

Being named as Rental_Users, this table is responsible for storing details for each user that uses this system.

Table 5.4: Data dictionary for Rental_Users table

Name of attribute	**Representation**	**Key**	**Type**	**Default Value**

User_id	The identity number for each user.	Primary	Integer, not null	Auto incremental
User_name	Full name for each user.	-	Varchar(125), not null	-
User_role	Identity for each user.	-	Varchar(30), not null	-
Email_address	User's UTAR email address.	-	Varchar(135), not null	-
Contact_num	User's contact number.	-	Varchar(20), not null	-
Password	User's encrypted password.	-	Varchar(550), not null	-
Account_status	User's account status.	-	Integer, not null	1

5.3.2.5 Chat table

Being named as Rental_chat, this table is responsible for storing details for each piece of message in the system.

Table 5.5: Data dictionary for Rental_ chat table

Name of attribute	**Representation**	**Key**	**Type**	**Default Value**
Chat_id	The identity number for each message.	Primary	Integer, not null	Auto incremental
first_user_id	User ID for the specific message sent.	Foreign	Integer, not null	-
second_user_id	User ID for the specific message sent.	Foreign	Integer, not null	-

chat_contents	Text contents that were sent by the user.	-	Text, not null	-
sent_timestamp	Sending timestamp for each messages.	-	DateTime, not null	-

5.3.2.6 Events table

Being named as Rental_events, this table is responsible for storing details for each calendar event in the system.

Table 5.6: Data dictionary for Rental_events table

Name of attribute	**Representation**	**Key**	**Type**	**Default Value**
Event_id	The identity number for each calendar events.	Primary	Integer, not null	Auto incremental
Event_name	The name for each calendar events.	-	Varchar(100), not null	-
Event_type	The type for each calendar events.	-	Varchar(50), not null	-
Event_date	The date for each calendar events.	-	Date, not null	-

5.3.2.7 Rejected requests table

Being named as Rejected_booking, this table is responsible for storing details for each rejected venue booking requests.

Table 5.7: Data dictionary for Rejected_booking table

Name of attribute	Representation	Key	Type	Default Value
Rejected_id	The identity number for each rejected request.	Primary	Integer, not null	Auto incremental
Rejected_staff_id	The HOD or DGS staff user ID that rejects this request.	Foreign	Integer, not null	-
Rejected_booking_id	The specific venue booking request ID that is being rejected.	Foreign	Integer, not null	-
Rejected_reason	The cause of disapproving the venue booking request.	-	Varchar(100), not null	-
Rejected_date	The disapproval date of the request.	-	DateTime, not null	-

5.3.2.8 Courses table

Being named as Courses, this table is responsible for storing details for courses for each study programme, which could be related to preserving venues for class events.

Table 5.8: Data dictionary for Courses table

Name of attribute	Representation	Key	Type	Default Value

Course_id	The identity number for each course.	Primary	Integer, not null	Auto incremental
Course_name	Name for the specific course.	-	Text, not null	-
Faculty	The name of faculty where the course belongs to.	-	Text, not null	-

5.3.2.9 Semester table

Being named as Semester, this table is responsible for storing details for each trimester, which could be used to pre-fetch dates for reserving venues for classes.

Table 5.9: Data dictionary for Semester table

Name of attribute	**Representation**	**Key**	**Type**	**Default Value**
Semester_ID	The identity number for each trimester.	Primary	Integer, not null	Auto incremental
Type	The level of study for the specific trimester.	-	Text, not null	-
Semester_name	The trimester name, differ from the level of study and the timing.	-	Varchar (100), not null	-
Semester_starting _duration	The starting date for the trimester.	-	Date, not null	-
Semester_ending _duration	The ending date for the trimester.	-	Date, not null	-

5.3.2.10 Password reset table

Being named as reset_password, this table is responsible for storing details for each reset password attempt, including the attempted email address, the identifier value for each attempt and the status for each attempt.

Table 5.10: Data dictionary for Reset password table

Name of attribute	Representation	Key	Type	Default Value
ID	The identity number for each reset password attempt.	Primary	Integer, not null	Auto incremental
Email address	The email address that tries to reset password.	-	Text, not null	-
Identifier	A randomly generated string, used for validation for each attempt.	-	Text, not null	-
Status	Boolean status for each reset attempt.	-	Text, not null	-

5.4 Data Flow Diagram

This section provides a detailed view of the internal relationships for the UTAR Venue/Classroom Booking Web Portal. Starting from the Context Diagram, it narrates the actions between each level of end-users and any external libraries with the system developed. Followed by the Level 0 Data Flow Diagram, which decomposes the general system into specific processes and the information flow.

5.4.1 Context Diagram

Describing the relationships between the system with the potential involved parties, the system is represented in the eclipse, while connects with the external parties that are in squares. For this project, there will be administrative users (HOD, DGS) and non-administrative users (lecturers and students) and external libraries that might have their APIs implemented in it.

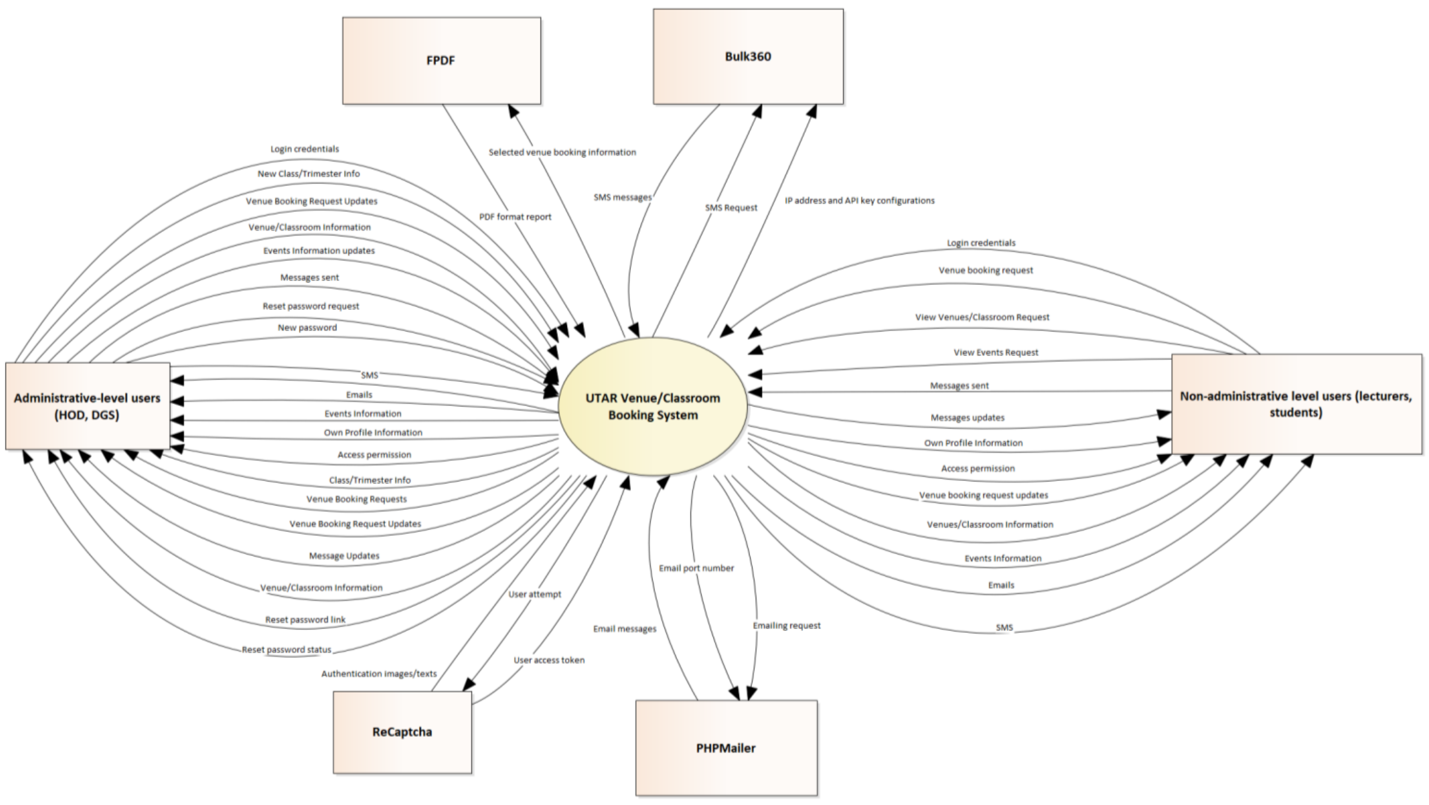


Figure 5.3: Context Diagram

5.4.2 Level 0 Data Flow Diagram

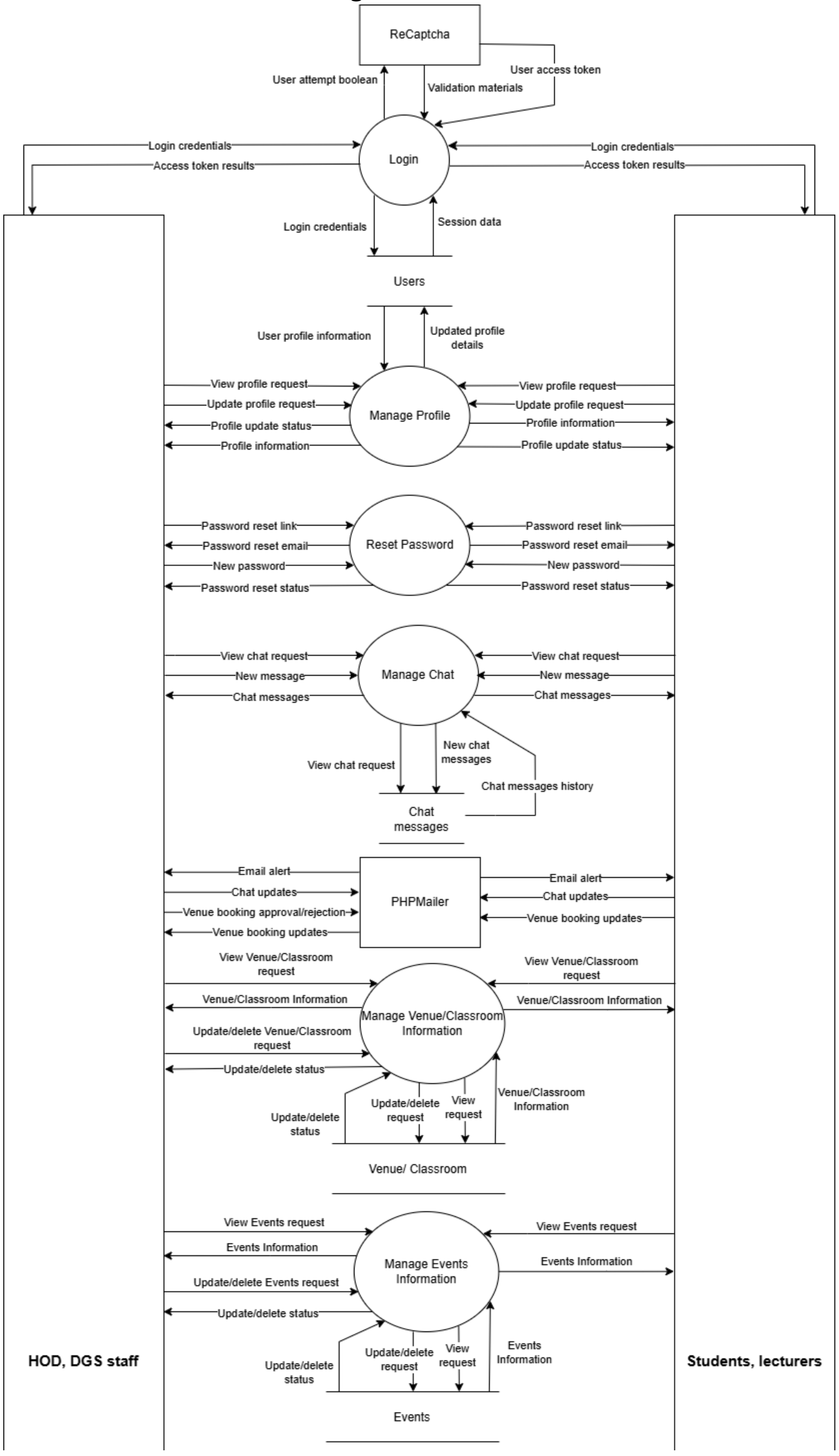


Figure 5.4: Level 0 DFD for the system developed, first part

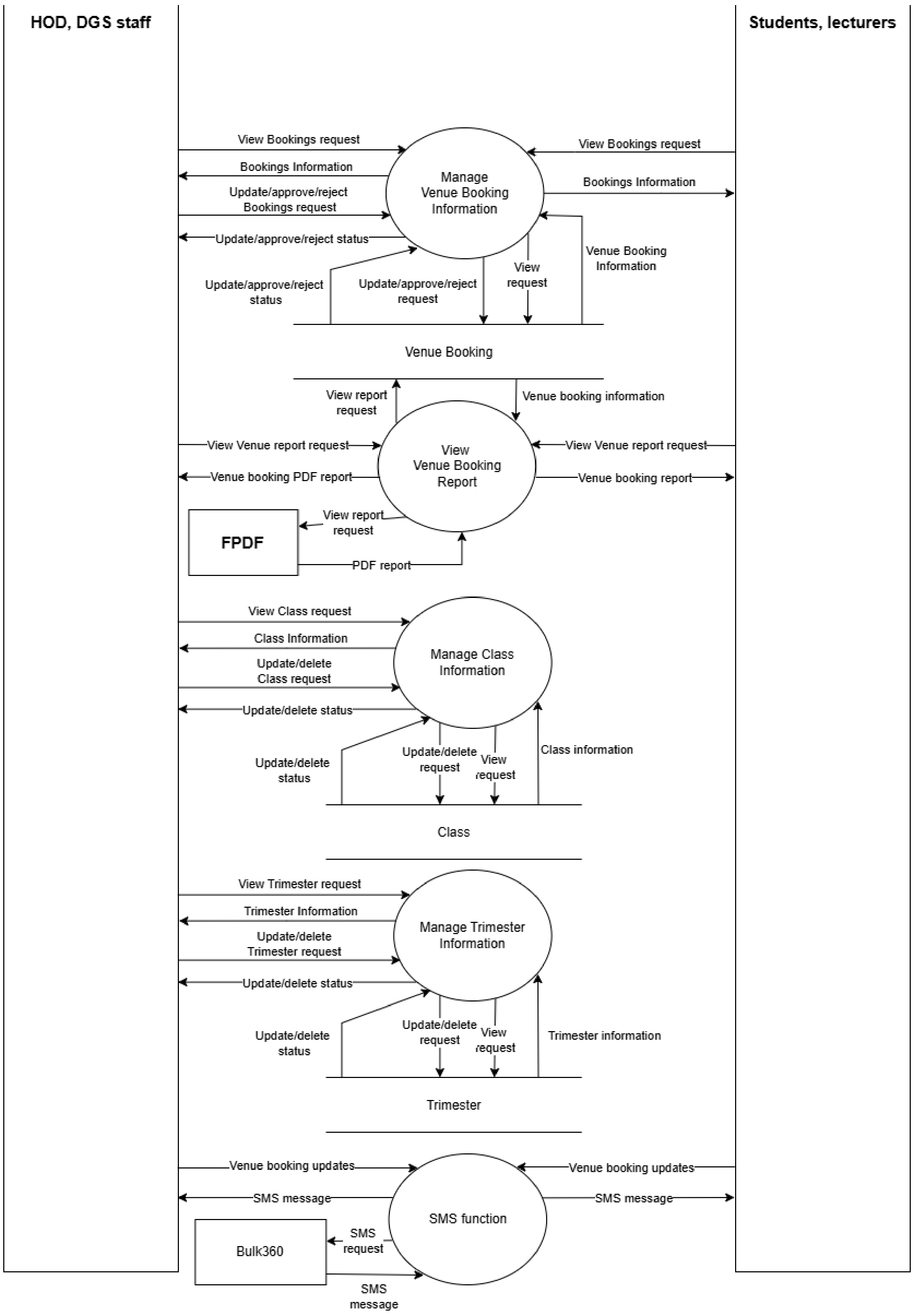


Figure 5.5: Level 0 DFD for the system developed, second part

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Introduction

This segment describes the implementation details of the developed UTAR Venue/Classroom Booking Web Portal, after featuring all requirements into a specialized system architecture. Technically, though the system by the name will just serve CRUD purposes, but it has notable features which general venue booking systems would not be equipped with.

Firstly, section 6.2 will specify the system modules that are usable by which role of end-users. Section 6.3 will discuss the general processes like login/logout, venue/classroom booking and viewing and events management. Section 6.4 will discuss the PDF report generation. Section 6.5 will explore the notification function, which consists of Bulk360 SMS API and PhpMailer. Sequentially, section 6.6 delves into the Messaging module where it is an instant messaging function built internally. Section 6.7 will cover any security mechanisms, inclusive of accessing and data validation. Section 6.8 will explain the functions desired by the DGS staff, which will be the class and trimester information management. Lastly, there would be section 6.9 to explain how the system is to be used in mobile devices, followed by a concluding summary.

6.2 System Modules

The end-users for this system are categorized into four groups, that will be the administrative side (DGS staff and HOD) and the non-administrative side (lecturers and students). Since some functions are related to managerial tasks, a few submodules could be only accessed by administrative users.

Table 6.1: System modules

End-users	**Module**	**Aim of module**
DGS staff	Access Authentication	Login and reset password

	Profile Management	View and update profile information
	Venue/Classroom booking	Submit a new venue/classroom usage request
	Manage Venue/Classroom booking	- Approve or reject the booking requests submitted - Add reminder notes for those approved booking requests - View requests' information in web interface or PDF report format
	Manage Venue/Classroom details	- To view on the selected venue/classroom details - To edit on the selected venue/classroom facilities and availability status
	Manage events details	- To view event details based on selected date - To add, edit and delete a selected event
	Message	To view conversation with other users and chatting

	Manage class information	- To search and view added class information - To add a new class session for venue /classroom booking - To edit a selected class information - To delete an added class session for venue /classroom booking
	Manage trimester information	- To search and view added trimester information - To add a trimester for class arrangement - To edit a selected trimester information - To delete an added trimester for class arrangement
HOD	Access Authentication	Login and reset password
	Profile Management	View and update profile information
	Venue/Classroom booking	Submit a new venue/classroom usage request
	Manage Venue/Classroom booking	- Approve or reject the booking requests submitted - Add reminder notes for those approved booking requests

		- View requests' information in web interface or PDF report format
	View Venue/Classroom details	- To view on the selected venue/classroom details
	View events details	- To view event details based on selected date
	Message	To view conversation with other users and chatting
Lecturers	Access Authentication	Login and reset password
	Profile Management	View and update profile information
	Venue/Classroom booking	Submit a new venue/classroom usage request
	Manage Venue/Classroom booking	- Approve or reject the booking requests submitted - View requests' information in web interface or PDF report format
	View Venue/Classroom details	- To view on the selected venue/classroom details
	View events details	- To view event details based on selected date

	Message	To view conversation with other users and chatting
Students	Access Authentication	Login and reset password
	Profile Management	View and update profile information
	Venue/Classroom booking	Submit a new venue/classroom usage request
	Manage Venue/Classroom booking	- Cancel uncompleted booking requests - View requests' information in web interface or PDF report format
	View Venue/Classroom details	- To view on the selected venue/classroom details
	View events details	- To view event details based on selected date
	Message	To view conversation with other users and chatting

6.3 General Processes

This section discusses the general functions that are commonly used among the DGS staff, HOD, lecturers and the students. There will be a minor difference in the venue/classroom-related functions between the administrative users and the non-administrative users.

6.3.1 Login/logout

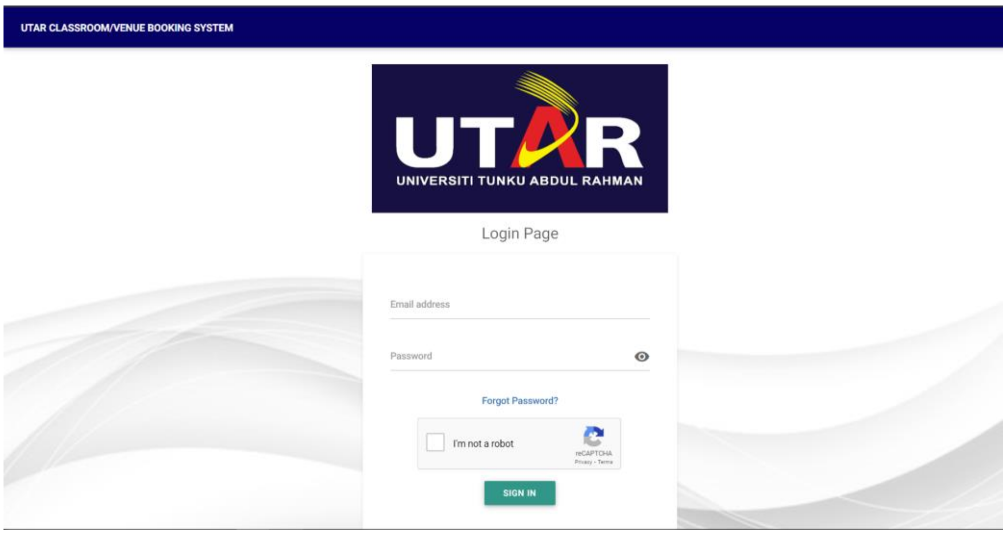
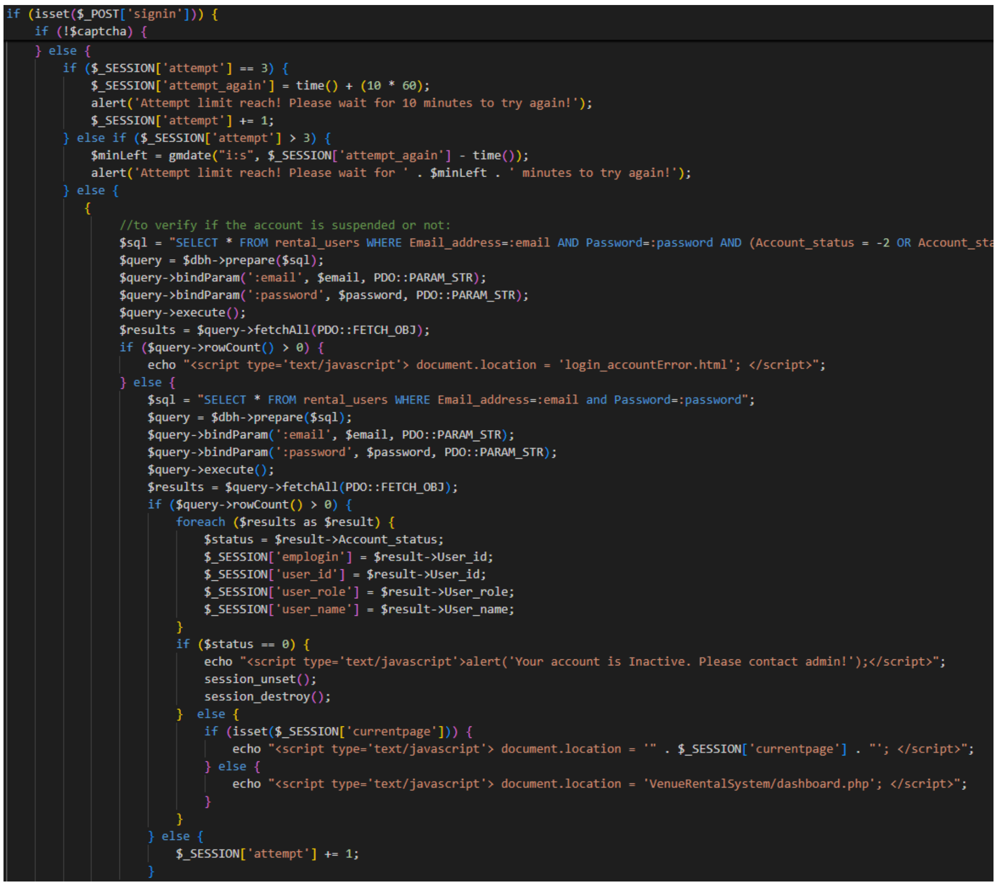


Figure 6.1: Login page

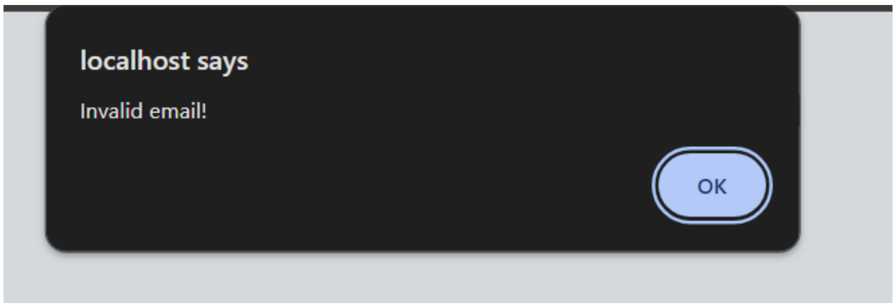
When any levels of the end-user accesses the system, they must log in with their UTAR email address to prevent unauthorized parties from simply accessing, followed by checking the ReCAPTCHA form. Once the user has entered the right credentials, then the user will be granted access with the session data assigned with own user information.



```php
if (isset($_POST['signin'])) {
    if (!$captcha) {
    } else {
        if ($_SESSION['attempt'] == 3) {
            $_SESSION['attempt_again'] = time() + (10 * 60);
            alert('Attempt limit reach! Please wait for 10 minutes to try again!');
            $_SESSION['attempt'] += 1;
        } else if ($_SESSION['attempt'] > 3) {
            $minLeft = gmdate("i:s", $_SESSION['attempt_again'] - time());
            alert('Attempt limit reach! Please wait for ' . $minLeft . ' minutes to try again!');
        } else {
            {
                //to verify if the account is suspended or not:
                $sql = "SELECT * FROM rental_users WHERE Email_address=:email AND Password=:password AND (Account_status = -2 OR Account_sta
                $query = $dbh->prepare($sql);
                $query->bindParam(':email', $email, PDO::PARAM_STR);
                $query->bindParam(':password', $password, PDO::PARAM_STR);
                $query->execute();
                $results = $query->fetchAll(PDO::FETCH_OBJ);
                if ($query->rowCount() > 0) {
                    echo "<script type='text/javascript'> document.location = 'login_accountError.html'; </script>";
                } else {
                    $sql = "SELECT * FROM rental_users WHERE Email_address=:email and Password=:password";
                    $query = $dbh->prepare($sql);
                    $query->bindParam(':email', $email, PDO::PARAM_STR);
                    $query->bindParam(':password', $password, PDO::PARAM_STR);
                    $query->execute();
                    $results = $query->fetchAll(PDO::FETCH_OBJ);
                    if ($query->rowCount() > 0) {
                        foreach ($results as $result) {
                            $status = $result->Account_status;
                            $_SESSION['emplogin'] = $result->User_id;
                            $_SESSION['user_id'] = $result->User_id;
                            $_SESSION['user_role'] = $result->User_role;
                            $_SESSION['user_name'] = $result->User_name;
                        }
                        if ($status == 0) {
                            echo "<script type='text/javascript'>alert('Your account is Inactive. Please contact admin!');</script>";
                            session_unset();
                            session_destroy();
                        }  else {
                            if (isset($_SESSION['currentpage'])) {
                                echo "<script type='text/javascript'> document.location = '" . $_SESSION['currentpage'] . "'; </script>";
                            } else {
                                echo "<script type='text/javascript'> document.location = 'VenueRentalSystem/dashboard.php'; </script>";
                            }
                        }
                    } else {
                        $_SESSION['attempt'] += 1;
                    }
```

Figure 6.2: Login function

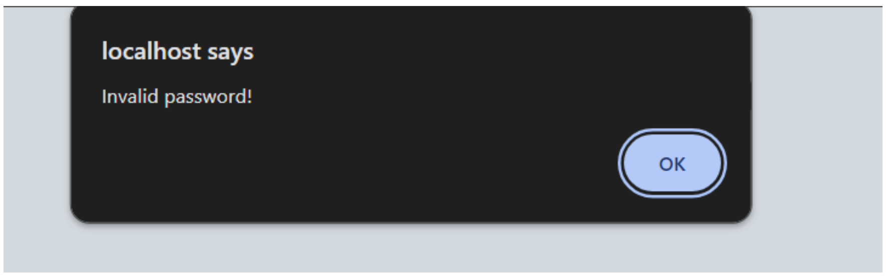
Checking with the database to verify the identity, in case any of the email address or the password is invalid, then the system will alert the user regarding the right thing to be aware. This works by fetching users' data once the sign-in button is clicked. The following diagrams show the example of the alert messages and the code implementation.



localhost says

Invalid email!

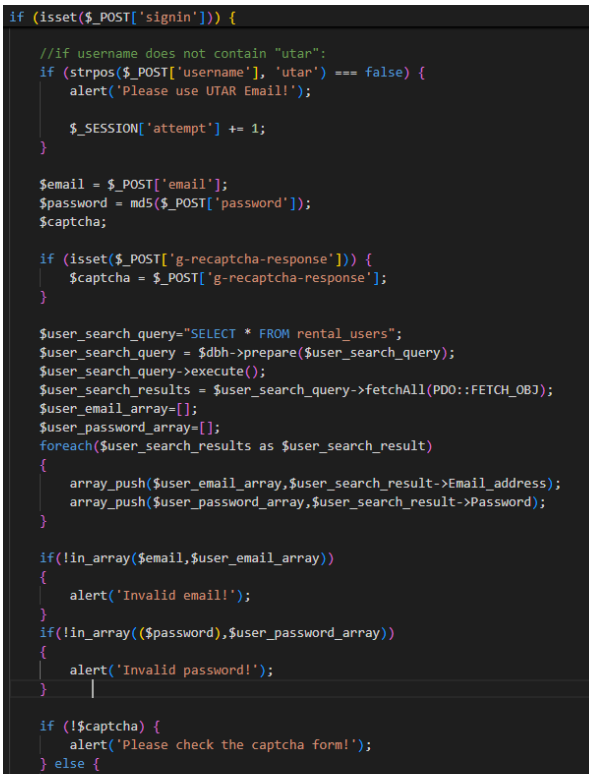
Figure 6.3: Message if the user has entered an invalid email address



localhost says

Invalid password!

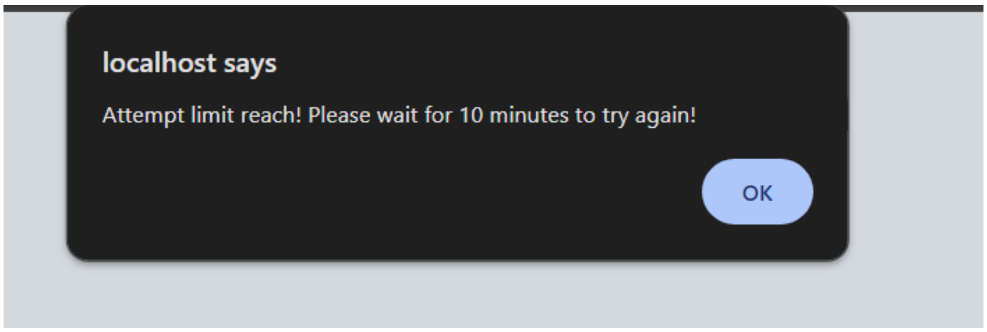
Figure 6.4: Message if the user has entered an invalid password



```
if (isset($_POST['signin'])) {

    //if username does not contain "utar":
    if (strpos($_POST['username'], 'utar') === false) {
        alert('Please use UTAR Email!');

        $_SESSION['attempt'] += 1;
    }

    $email = $_POST['email'];
    $password = md5($_POST['password']);
    $captcha;

    if (isset($_POST['g-recaptcha-response'])) {
        $captcha = $_POST['g-recaptcha-response'];
    }

    $user_search_query="SELECT * FROM rental_users";
    $user_search_query = $dbh->prepare($user_search_query);
    $user_search_query->execute();
    $user_search_results = $user_search_query->fetchAll(PDO::FETCH_OBJ);
    $user_email_array=[];
    $user_password_array=[];
    foreach($user_search_results as $user_search_result)
    {
        array_push($user_email_array,$user_search_result->Email_address);
        array_push($user_password_array,$user_search_result->Password);
    }

    if(!in_array($email,$user_email_array))
    {
        alert('Invalid email!');
    }
    if(!in_array(($password),$user_password_array))
    {
        alert('Invalid password!');
    }

    if (!$captcha) {
        alert('Please check the captcha form!');
    } else {
```

Figure 6.5: Verifying function of the email address and password input

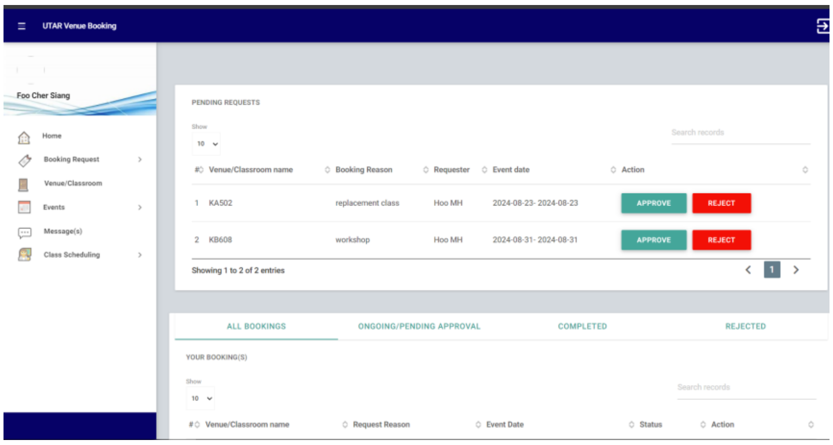
Every user has given three attempts to login, else if the user fails to give a valid credential, then the user will be banned from logging into the system for 10 minutes. This was achieved by setting the attempt count and utilizing the *time()* function in PHP, shown in the top lines in Figure 6.2.


localhost says

Attempt limit reach! Please wait for 10 minutes to try again!

Figure 6.6: Locking function for invalid login credentials

The end-user will be directed to the dashboard, which serves as the home page for the UTAR Venue/Classroom Booking Web Portal.



PENDING REQUESTS

Show 10

Search records

#	Venue/Classroom name	Booking Reason	Requester	Event date	Action
1	KA502	replacement class	Hoo MH	2024-08-23- 2024-08-23	APPROVE REJECT
2	KB608	workshop	Hoo MH	2024-08-31- 2024-08-31	APPROVE REJECT

Showing 1 to 2 of 2 entries

YOUR BOOKING(S)

Show 10

Search records

#	Venue/Classroom name	Request Reason	Event Date	Status	Action

Figure 6.7: Dashboard page once successfully login

Logout-wise, the function is triggered once the end-user clicks the logout button at the header, to reset the session name with blanks and the logged-in duration. In the end, the session is meant to be depleted.

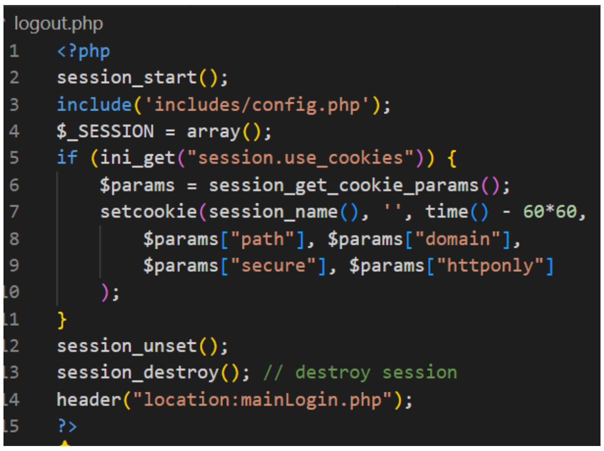

logout.php
```php
<?php
session_start();
include('includes/config.php');
$_SESSION = array();
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 60*60,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}
session_unset();
session_destroy(); // destroy session
header("location:mainLogin.php");
?>
```

Figure 6.8: Logout function

6.3.2 Reset password

The user could renew the password, by selecting the Forget Password directive link on the login page, if they had forgotten it. With the clicking action, the user is directed to enter their email address.

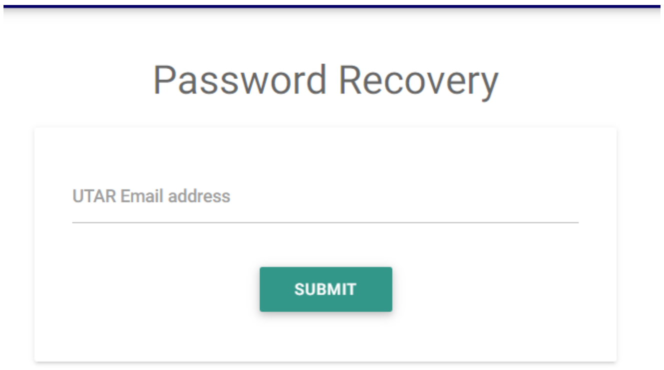


Figure 6.9: Page for the Forget Password link

Upon the user enters the email address where the password is forgotten, there is a checking to verify if there is a matching address in the users' database. Else, an error message of invalid address will be shown.

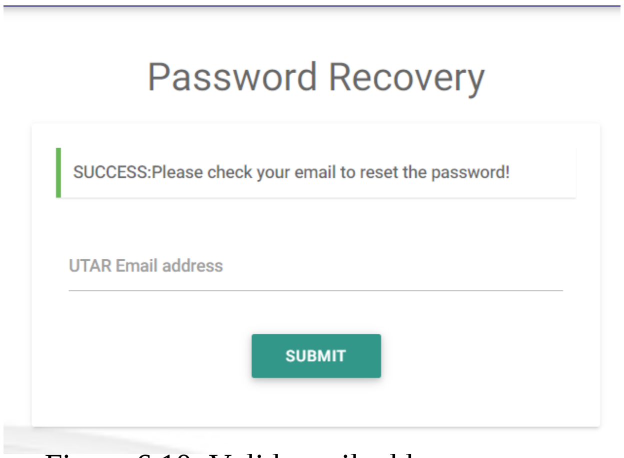

SUCCESS:Please check your email to reset the password!

Figure 6.10: Valid email address message

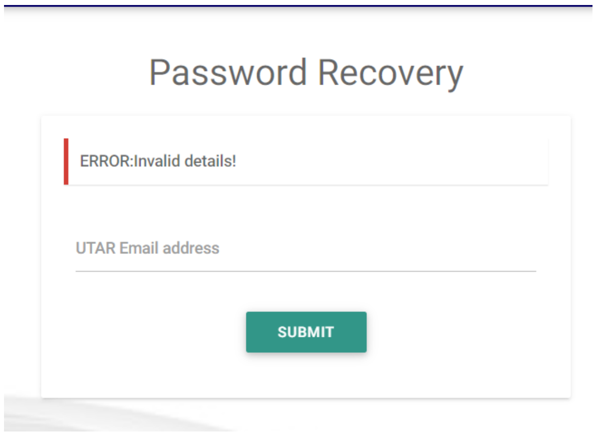
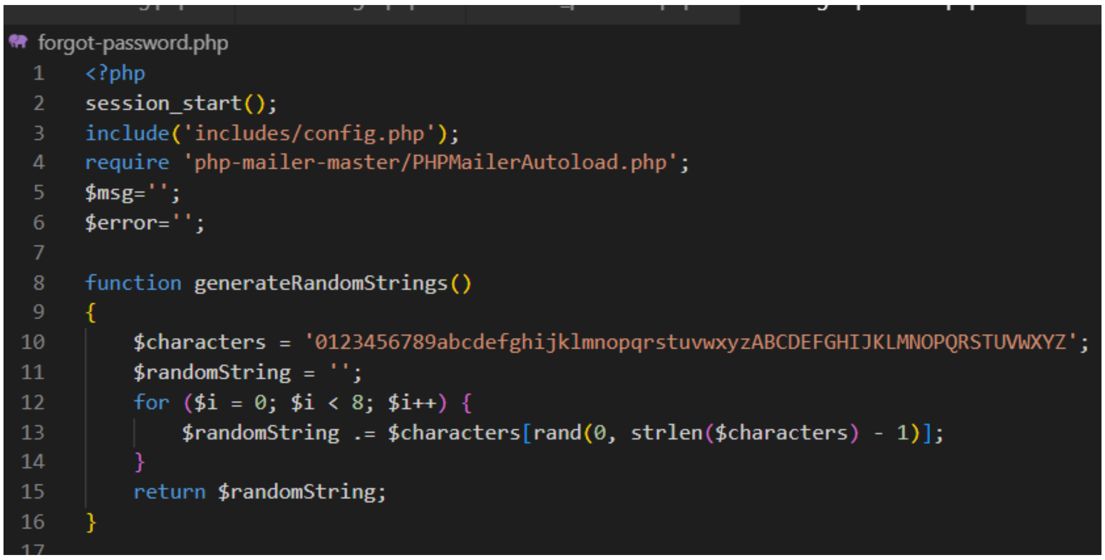

ERROR:Invalid details!

Figure 6.11: Invalid email address message

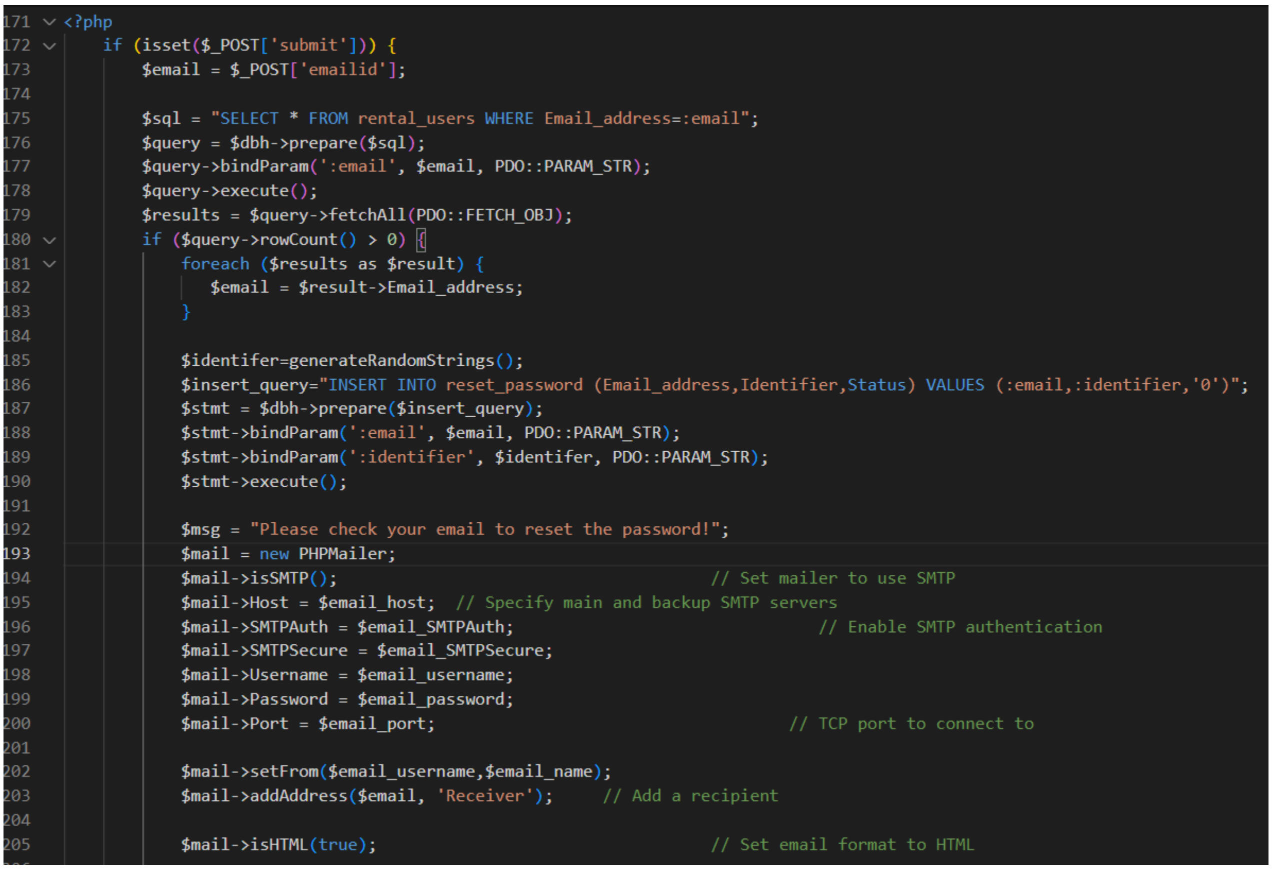
Once the email address entered is valid, then a unique identifier value which consists of 8 characters is generated, attached along with the email address to insert the database. Each attempt has a status value as well, to verify if the attempt has been updated through resetting the password. Do refer to the technical implementations in the following figures.



forgot-password.php

```php
<?php
session_start();
include('includes/config.php');
require 'php-mailer-master/PHPMailerAutoload.php';
$msg='';
$error='';

function generateRandomStrings()
{
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $randomString = '';
    for ($i = 0; $i < 8; $i++) {
        $randomString .= $characters[rand(0, strlen($characters) - 1)];
    }
    return $randomString;
}
```

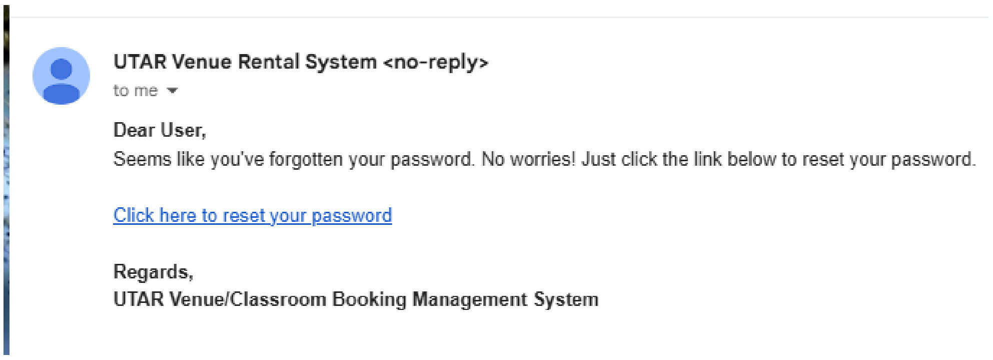
Figure 6.12: Technical implementation of generating identifier value



```php
<?php
    if (isset($_POST['submit'])) {
        $email = $_POST['emailid'];

        $sql = "SELECT * FROM rental_users WHERE Email_address=:email";
        $query = $dbh->prepare($sql);
        $query->bindParam(':email', $email, PDO::PARAM_STR);
        $query->execute();
        $results = $query->fetchAll(PDO::FETCH_OBJ);
        if ($query->rowCount() > 0) {
            foreach ($results as $result) {
                $email = $result->Email_address;
            }

            $identifer=generateRandomStrings();
            $insert_query="INSERT INTO reset_password (Email_address,Identifier,Status) VALUES (:email,:identifier,'0')";
            $stmt = $dbh->prepare($insert_query);
            $stmt->bindParam(':email', $email, PDO::PARAM_STR);
            $stmt->bindParam(':identifier', $identifer, PDO::PARAM_STR);
            $stmt->execute();

            $msg = "Please check your email to reset the password!";
            $mail = new PHPMailer;
            $mail->isSMTP();                                    // Set mailer to use SMTP
            $mail->Host = $email_host;  // Specify main and backup SMTP servers
            $mail->SMTPAuth = $email_SMTPAuth;                              // Enable SMTP authentication
            $mail->SMTPSecure = $email_SMTPSecure;
            $mail->Username = $email_username;
            $mail->Password = $email_password;
            $mail->Port = $email_port;                                  // TCP port to connect to

            $mail->setFrom($email_username,$email_name);
            $mail->addAddress($email, 'Receiver');      // Add a recipient

            $mail->isHTML(true);                                    // Set email format to HTML
```

Figure 6.13: Technical implementation of address verification and email preparation

With the help of PHPMailer, the input email account will receive a resetting password email, specified with a unique identifier value. Once the password had been updated, then the identifier status is updated to unusable anymore. The identifier value limits the user from updating their password anonymously, whether it has been updated or the value is not found in the database.

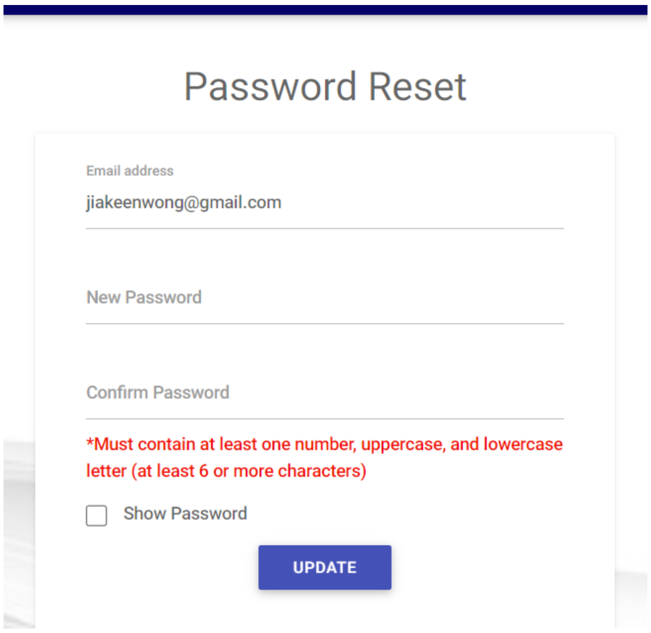

UTAR Venue Rental System <no-reply>
to me

Dear User,
Seems like you've forgotten your password. No worries! Just click the link below to reset your password.

Click here to reset your password

Regards,
UTAR Venue/Classroom Booking Management System

Figure 6.14: Email sent with the reset password link


Password Reset

Email address
jiakeenwong@gmail.com

New Password

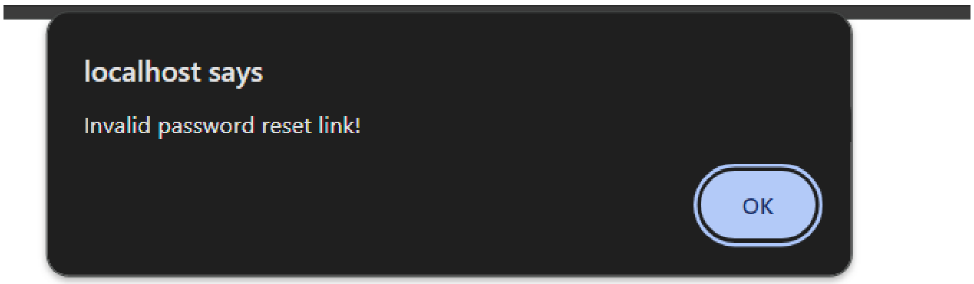
Confirm Password

*Must contain at least one number, uppercase, and lowercase letter (at least 6 or more characters)

Show Password

UPDATE

Figure 6.15: Interface for resetting password if the link is valid

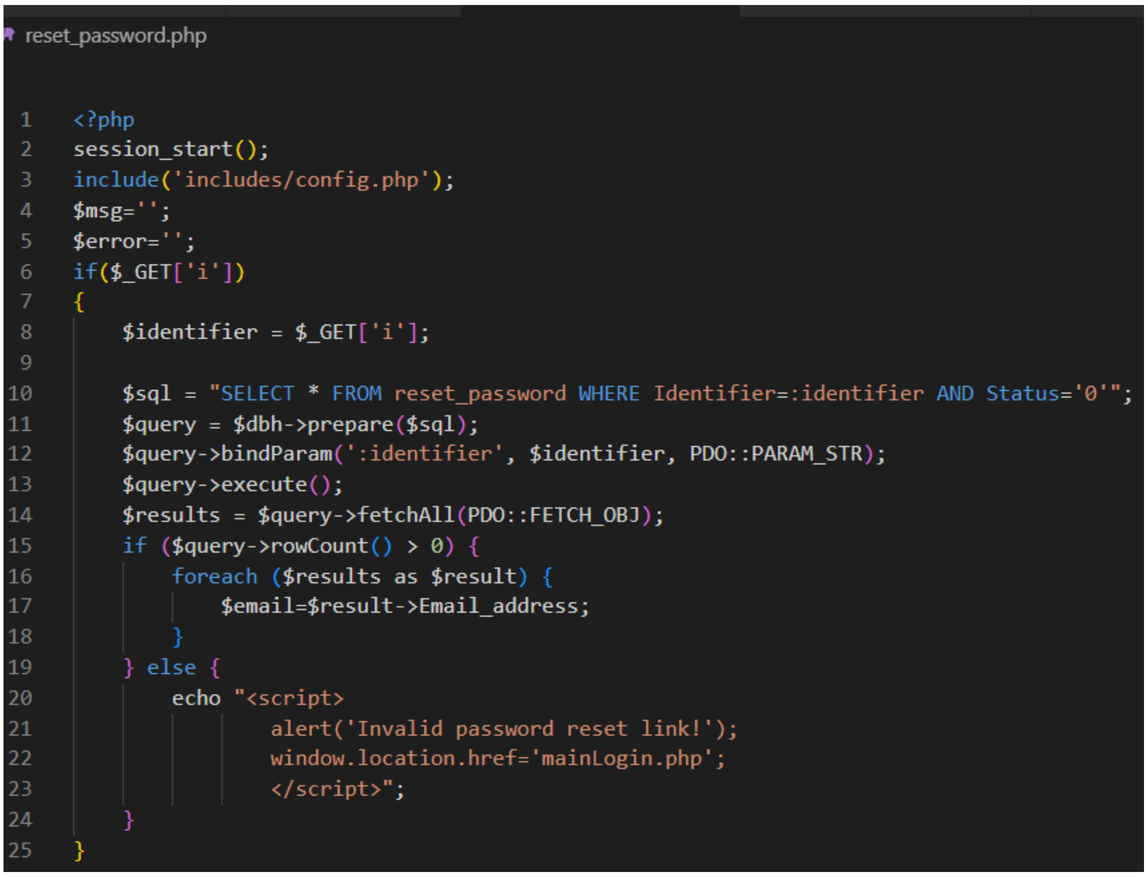

localhost says

Invalid password reset link!

OK

Figure 6.16: Invalid reset password link

The checking of identifier value on the reset password page is based on the status and the existence of such value in the database. Once accessing the page, the checking will be performed. Matching both scenarios described, if there are no data records retrieved, then the access will be prohibited.



reset_password.php

```php
<?php
session_start();
include('includes/config.php');
$msg='';
$error='';
if($_GET['i'])
{
    $identifier = $_GET['i'];

    $sql = "SELECT * FROM reset_password WHERE Identifier=:identifier AND Status='0'";
    $query = $dbh->prepare($sql);
    $query->bindParam(':identifier', $identifier, PDO::PARAM_STR);
    $query->execute();
    $results = $query->fetchAll(PDO::FETCH_OBJ);
    if ($query->rowCount() > 0) {
        foreach ($results as $result) {
            $email=$result->Email_address;
        }
    } else {
        echo "<script>
                alert('Invalid password reset link!');
                window.location.href='mainLogin.php';
                </script>";
    }
}
```

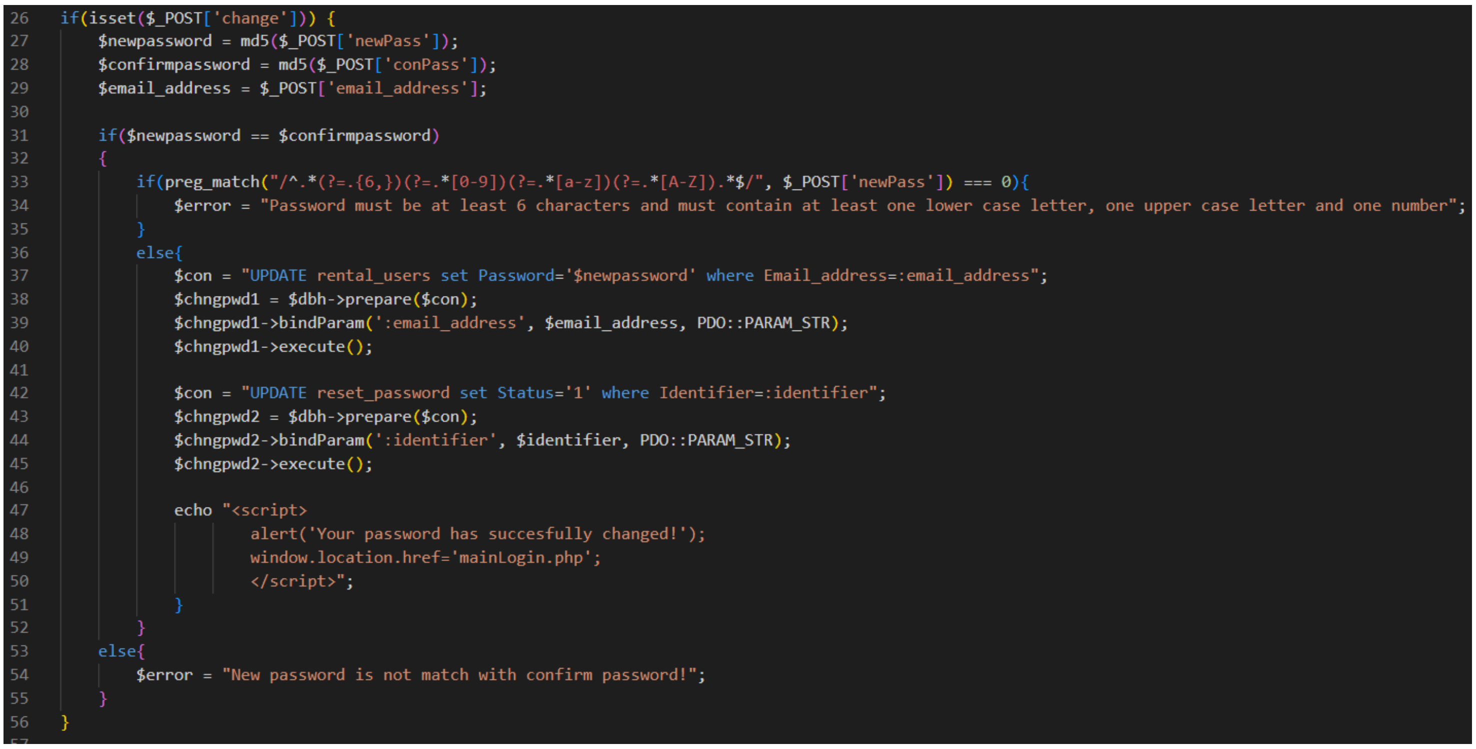
Figure 6.17: Checking on the identifier value on reset password page

The updated password needs to follow the validation rule as follows, so that the new password is considered valid, else validation messages will be shown on spot to remind the users:

1. The new password must match with the confirmation password
2. The new password must be alphanumeric, with at least an uppercase letter combining with a lowercase letter
3. The minimum length of the new password must be 6 characters

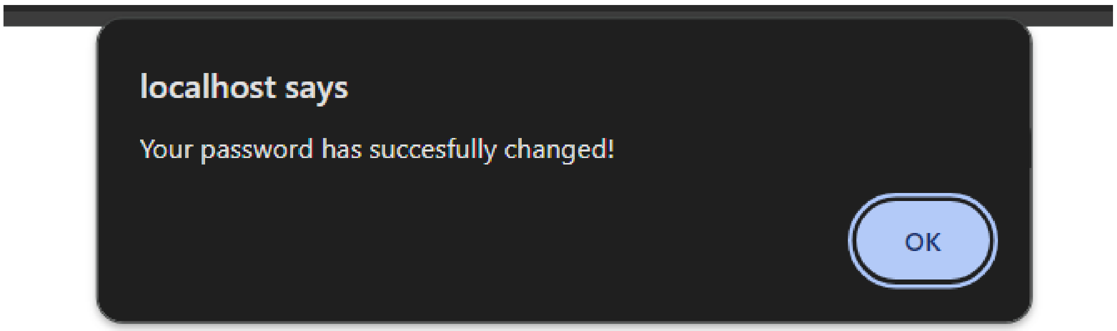
The new password values to be encrypted in hash values using the *md5()* function. Once the action is done, it updates the password value for the attempted user in the users' database. It triggers another update on the reset password database as well, so that the attempt for the unique identifier value is updated to completed.

A message of a successful update will be shown to the user at the end.



```php
if(isset($_POST['change'])) {
    $newpassword = md5($_POST['newPass']);
    $confirmpassword = md5($_POST['conPass']);
    $email_address = $_POST['email_address'];

    if($newpassword == $confirmpassword)
    {
        if(preg_match("/^.*(?=.{6,})(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).*$/", $_POST['newPass']) === 0){
            $error = "Password must be at least 6 characters and must contain at least one lower case letter, one upper case letter and one number";
        }
        else{
            $con = "UPDATE rental_users set Password='$newpassword' where Email_address=:email_address";
            $chngpwd1 = $dbh->prepare($con);
            $chngpwd1->bindParam(':email_address', $email_address, PDO::PARAM_STR);
            $chngpwd1->execute();

            $con = "UPDATE reset_password set Status='1' where Identifier=:identifier";
            $chngpwd2 = $dbh->prepare($con);
            $chngpwd2->bindParam(':identifier', $identifier, PDO::PARAM_STR);
            $chngpwd2->execute();

            echo "<script>
                    alert('Your password has succesfully changed!');
                    window.location.href='mainLogin.php';
                    </script>";
            }
        }
    else{
        $error = "New password is not match with confirm password!";
    }
}
```

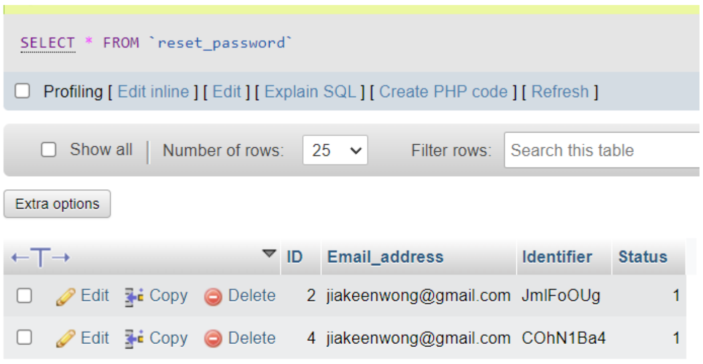
Figure 6.18: Technical implementation of update action



localhost says

Your password has succesfully changed!

Figure 6.19: Successfully updated message shown to the user



```
SELECT * FROM `reset_password`
```

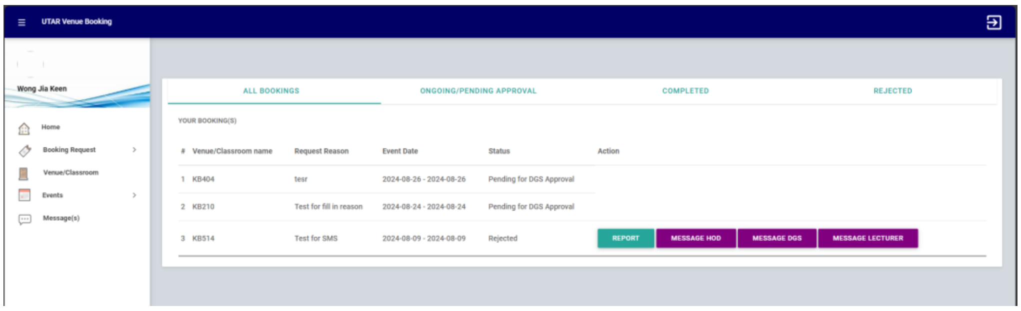
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

	ID	Email_address	Identifier	Status
Edit Copy Delete	2	jiakeenwong@gmail.com	JmlFoOUg	1
Edit Copy Delete	4	jiakeenwong@gmail.com	COhN1Ba4	1

Figure 6.20: Preview of reset password database

6.3.3 Dashboard

Upon logging into the system, the end-user is directed to the homepage, which is normally known as the dashboard in this project. Constructed by data tables, the end-users could see the booking requests' information, depending on their status. The one in the figure 6.7 is for the DGS staff, HOD and lecturers, which they could see the requests which await for their approval decision and their own booking requests as well. While for the students, they could only view their booking requests, as represented by Figure 6.21.

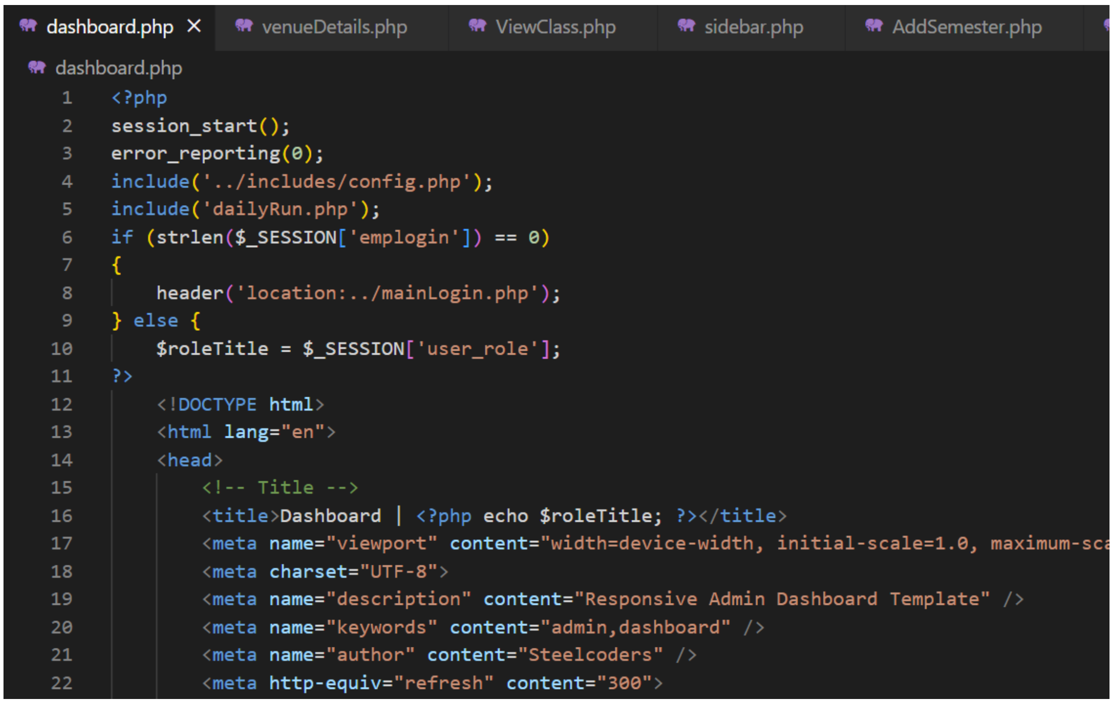


YOUR BOOKING(S)

#	Venue/Classroom name	Request Reason	Event Date	Status	Action
1	KB404	tesr	2024-08-26 - 2024-08-26	Pending for DGS Approval	
2	KB210	Test for fill in reason	2024-08-24 - 2024-08-24	Pending for DGS Approval	
3	KB514	Test for SMS	2024-08-09 - 2024-08-09	Rejected	REPORT MESSAGE HOD MESSAGE DGS MESSAGE LECTURER

Figure 6.21: Student dashboard page

To solve the problem of information overlooked, the dashboard page is set to be reloading every 5 minutes so that the latest updates could be shown on the dashboard. Do observe the 22nd line in the figure 6.22.



```
1   <?php
2   session_start();
3   error_reporting(0);
4   include('../includes/config.php');
5   include('dailyRun.php');
6   if (strlen($_SESSION['emplogin']) == 0)
7   {
8       header('location:../mainLogin.php');
9   } else {
10      $roleTitle = $_SESSION['user_role'];
11  ?>
12      <!DOCTYPE html>
13      <html lang="en">
14      <head>
15          <!-- Title -->
16          <title>Dashboard | <?php echo $roleTitle; ?></title>
17          <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-sca
18          <meta charset="UTF-8">
19          <meta name="description" content="Responsive Admin Dashboard Template" />
20          <meta name="keywords" content="admin,dashboard" />
21          <meta name="author" content="Steelcoders" />
22          <meta http-equiv="refresh" content="300">
```

Figure 6.22: Technical implementation of reloading every 5 minutes

6.3.4 Venue/Classroom Viewing

6.3.4.1 Mapping

If the end-user wishes to know more about the venue/classroom details, he/she could refer to the Venue/Classroom tab that resides in the sidebar. After being directed to the viewing page in Figure 6.24, the end-user could choose the right building to view.

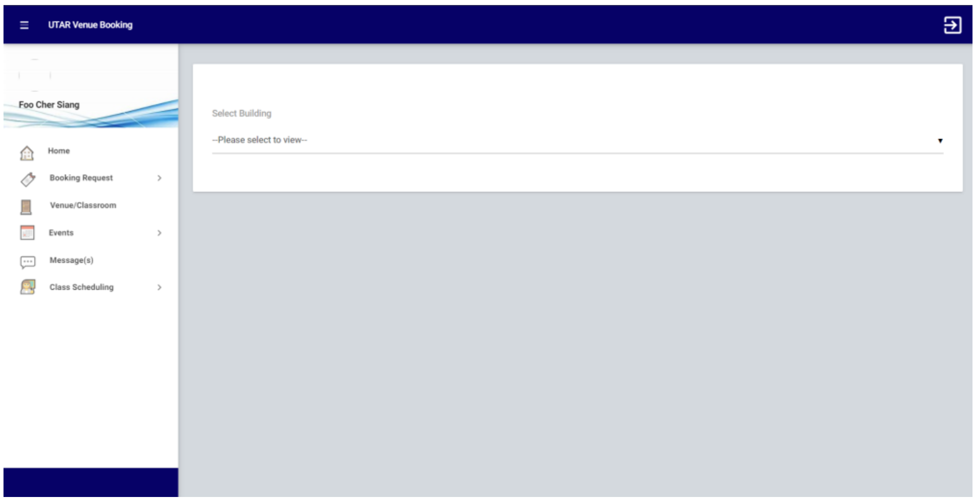


Figure 6.23: Venue/Classroom Viewing page

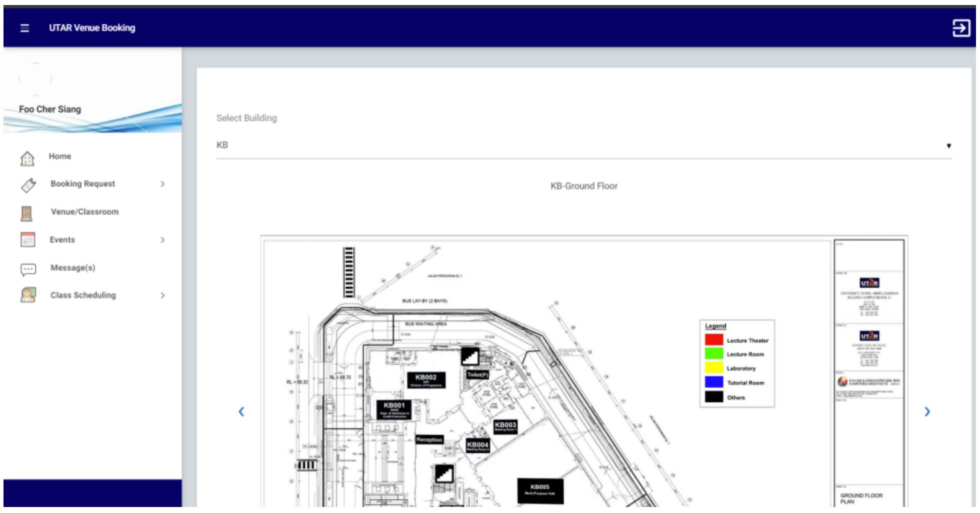
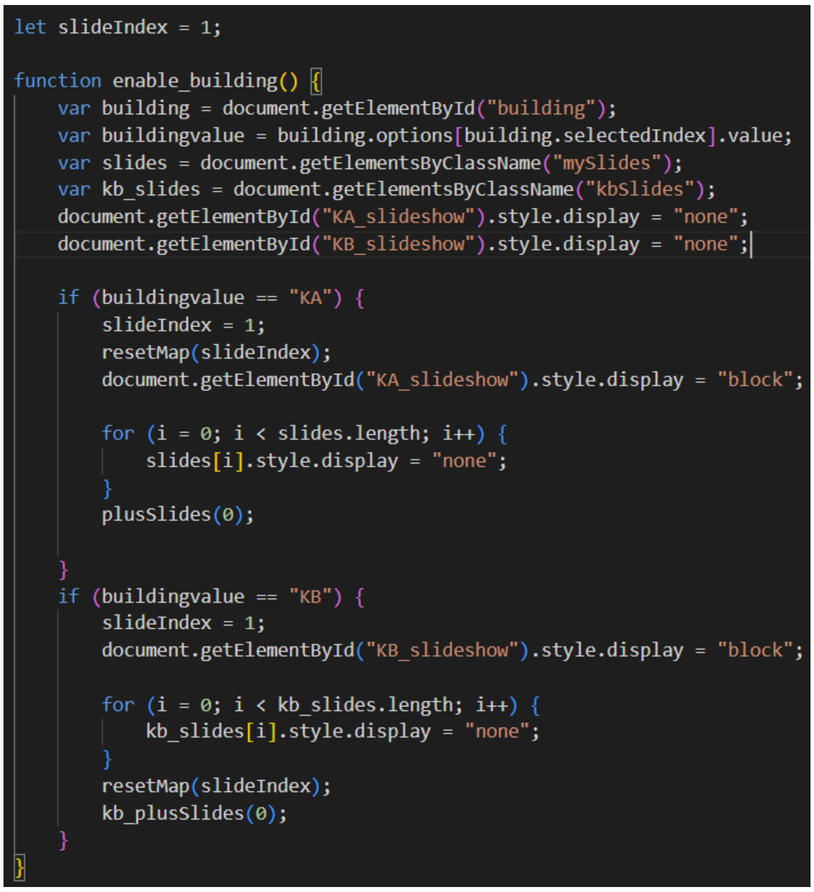


Figure 6.24: Venue/Classroom Viewing page with map

After choosing the right building from the drop-down list, then the equivalent maps will be shown in Figure 6.24. This is achieved through external JavaScript that reads the input choice of the end-user, then perform amendments on the elements to be displayed.

The maps are images that are fetched from the database, being placed in a division that is declared hidden initially. Once the JavaScript reads the changes on the building names, then relevant division would be displayed. This function could be defined in Figure 6.25.

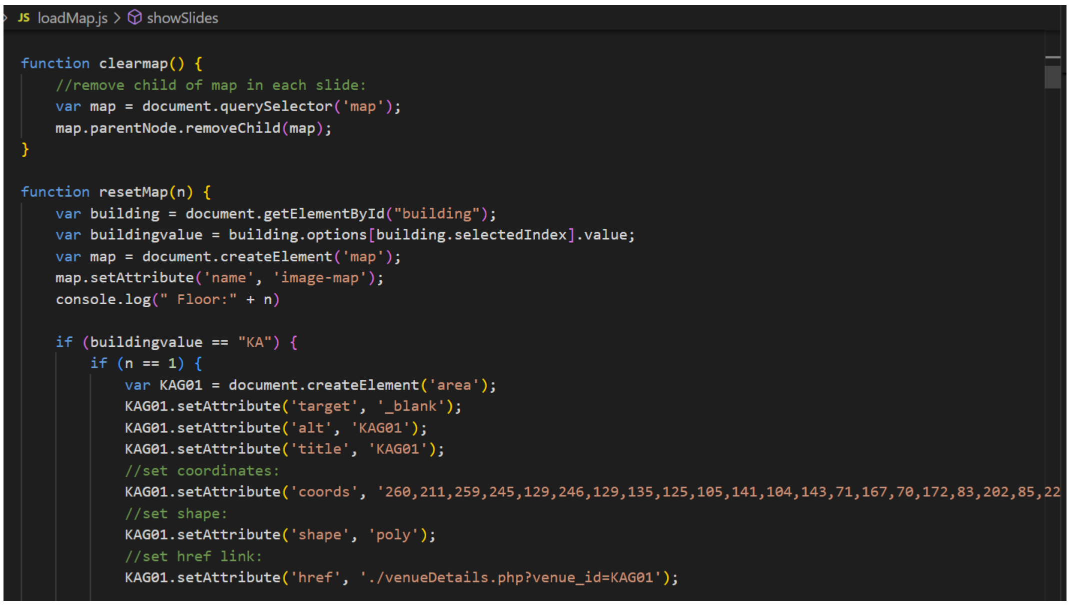


```
let slideIndex = 1;

function enable_building() {
    var building = document.getElementById("building");
    var buildingvalue = building.options[building.selectedIndex].value;
    var slides = document.getElementsByClassName("mySlides");
    var kb_slides = document.getElementsByClassName("kbSlides");
    document.getElementById("KA_slideshow").style.display = "none";
    document.getElementById("KB_slideshow").style.display = "none";

    if (buildingvalue == "KA") {
        slideIndex = 1;
        resetMap(slideIndex);
        document.getElementById("KA_slideshow").style.display = "block";

        for (i = 0; i < slides.length; i++) {
            slides[i].style.display = "none";
        }
        plusSlides(0);

    }
    if (buildingvalue == "KB") {
        slideIndex = 1;
        document.getElementById("KB_slideshow").style.display = "block";

        for (i = 0; i < kb_slides.length; i++) {
            kb_slides[i].style.display = "none";
        }
        resetMap(slideIndex);
        kb_plusSlides(0);
    }
}
```

Figure 6.25: Switching display of division to show maps

To simulate the maps to be displayed in a self-controlled slideshow mode, the maps are being set to display with the use of index position under the range of number of maps. The index was set to default to 1, then if the end-user wants to navigate to other maps, they can choose the “next” or “previous” button. The clicking of these buttons may trigger an addition or a subtraction of index number respectively. The index number is passed as a parameter to another function that runs a loop: where the slides loop in the total number of maps would not be shown, whereas the slide that its index hits the number will be shown.

The maps are working with certain areas segmented as a clickable link, to redirect the end-users to the venue viewing page. This could be achieved by setting the area with the ‘area’ tag, which requires coordinates to be drafted on each image map. As the maps change with user actions, so each navigating button will activate the function to clear the map regions, and to set up coordinates for transitioned maps. This could be represented in Figure 6.26.



```
function clearmap() {
    //remove child of map in each slide:
    var map = document.querySelector('map');
    map.parentNode.removeChild(map);
}

function resetMap(n) {
    var building = document.getElementById("building");
    var buildingvalue = building.options[building.selectedIndex].value;
    var map = document.createElement('map');
    map.setAttribute('name', 'image-map');
    console.log(" Floor:" + n)

    if (buildingvalue == "KA") {
        if (n == 1) {
            var KAG01 = document.createElement('area');
            KAG01.setAttribute('target', '_blank');
            KAG01.setAttribute('alt', 'KAG01');
            KAG01.setAttribute('title', 'KAG01');
            //set coordinates:
            KAG01.setAttribute('coords', '260,211,259,245,129,246,129,135,125,105,141,104,143,71,167,70,172,83,202,85,22
            //set shape:
            KAG01.setAttribute('shape', 'poly');
            //set href link:
            KAG01.setAttribute('href', './venueDetails.php?venue_id=KAG01');
```

Figure 6.26: Navigating button’s function once pressed

6.3.4.2 Viewing

With each region clicked in the maps, the end-user is redirected to the venue viewing page. The page displays images, information like facilities, venue name and type and the current availability status. This page is also featured with a data table, showcasing the recent events that took place in the selected venue, limiting to 5. The following figure provides a view of this page.

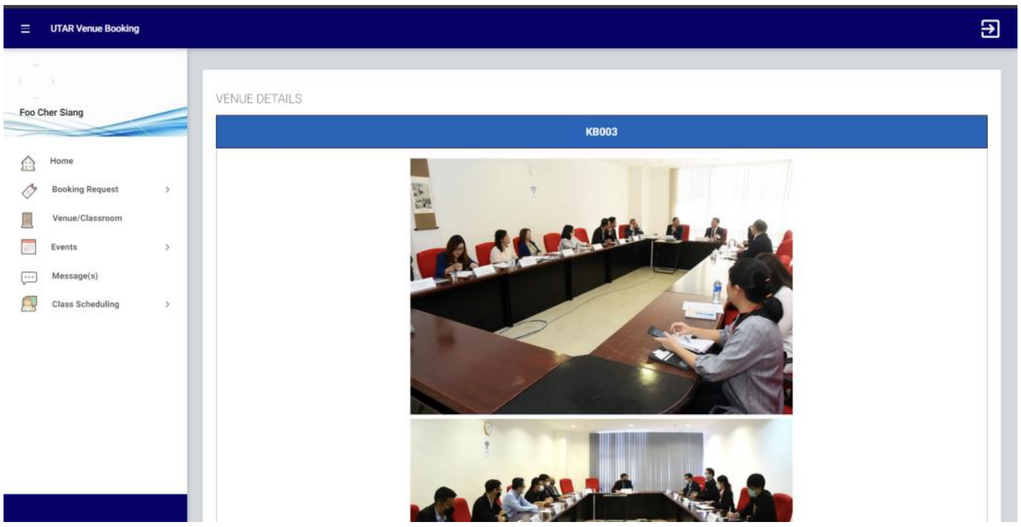
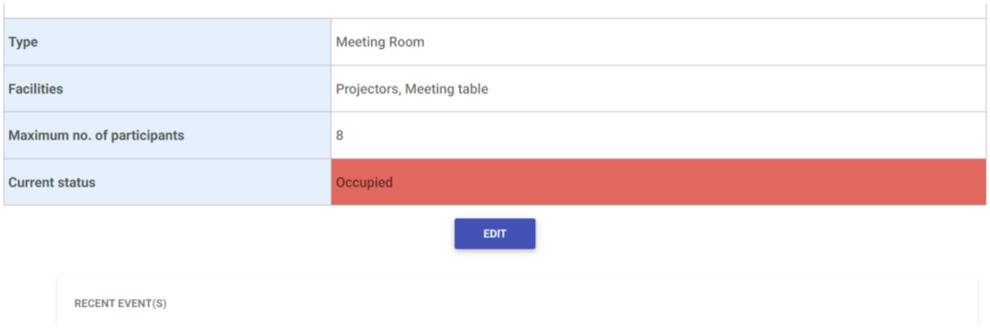


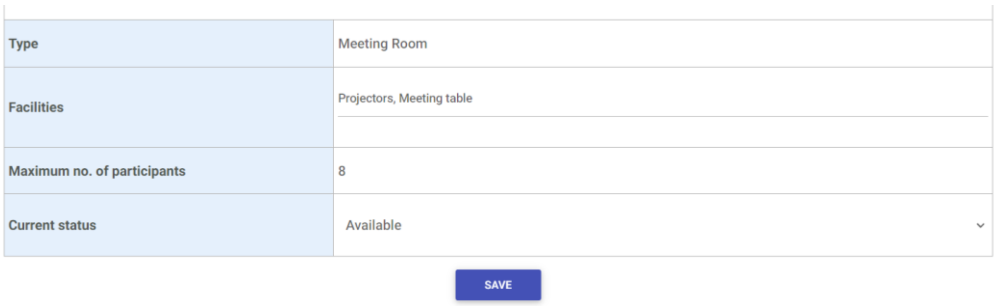
Figure 6.27: Venue viewing page

The DGS staff has the right to edit the venue's information, so the DGS staff has an edit button to edit the details. Since the details are presented in the table form, the edit button temporarily sets the table's contents to be editable, limited to facilities and the availability status only. Of course, the staff needs to save to keep performed amendments. The following figures portray the transitioning of the page.



Type	Meeting Room
Facilities	Projectors, Meeting table
Maximum no. of participants	8
Current status	Occupied

Figure 6.28: Edit button before activated

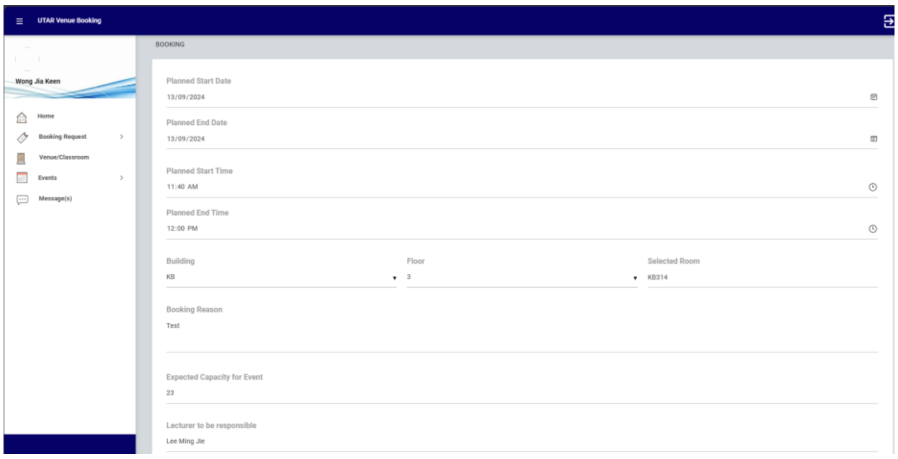


Type	Meeting Room
Facilities	Projectors, Meeting table
Maximum no. of participants	8
Current status	Available

Figure 6.29: Save button

6.3.5 Venue/Classroom booking

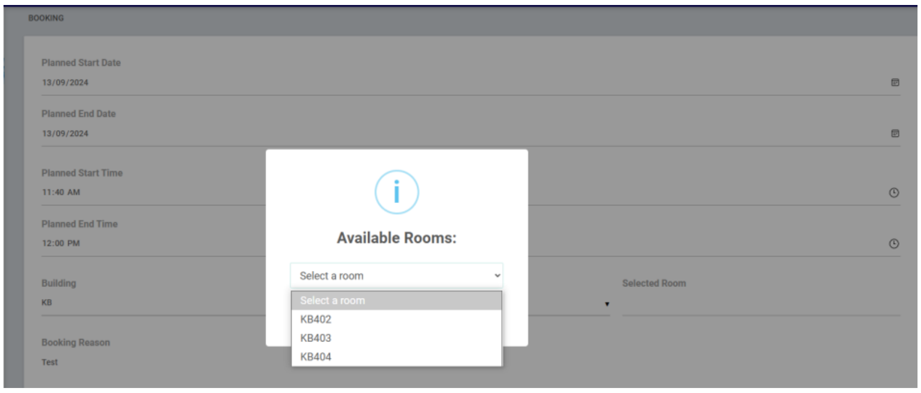
As the main point of setting up this system, any level of end-users could submit a venue/classroom usage request. This could be found in the Add Booking page which hides under the "Booking Request" tab in the sidebar. Students need to additionally select a lecturer since students' requests need to be supervised.



Planned Start Date	13/09/2024	
Planned End Date	13/09/2024	
Planned Start Time	11:40 AM	
Planned End Time	12:00 PM	
Building: KB	Floor: 3	Selected Room: KB314
Booking Reason	Test	
Expected Capacity for Event	23	
Lecturer to be responsible	Lee Ming Jie	

Figure 6.30: Venue/classroom booking page

Regardless of any levels, the end-users need to enter the starting and ending date, time values first. Based on the timing values chosen, then the division to choose venues will appear. Available venues' name will be fetched in a list through AJAX, once user has chosen the right building block and right floor. Technically, onchange event needs to be implemented on the timing inputs so that the data could be fetched from time to time.


BOOKING

Planned Start Date
13/09/2024

Planned End Date
13/09/2024

Planned Start Time
11:40 AM

Planned End Time
12:00 PM

Building
KB

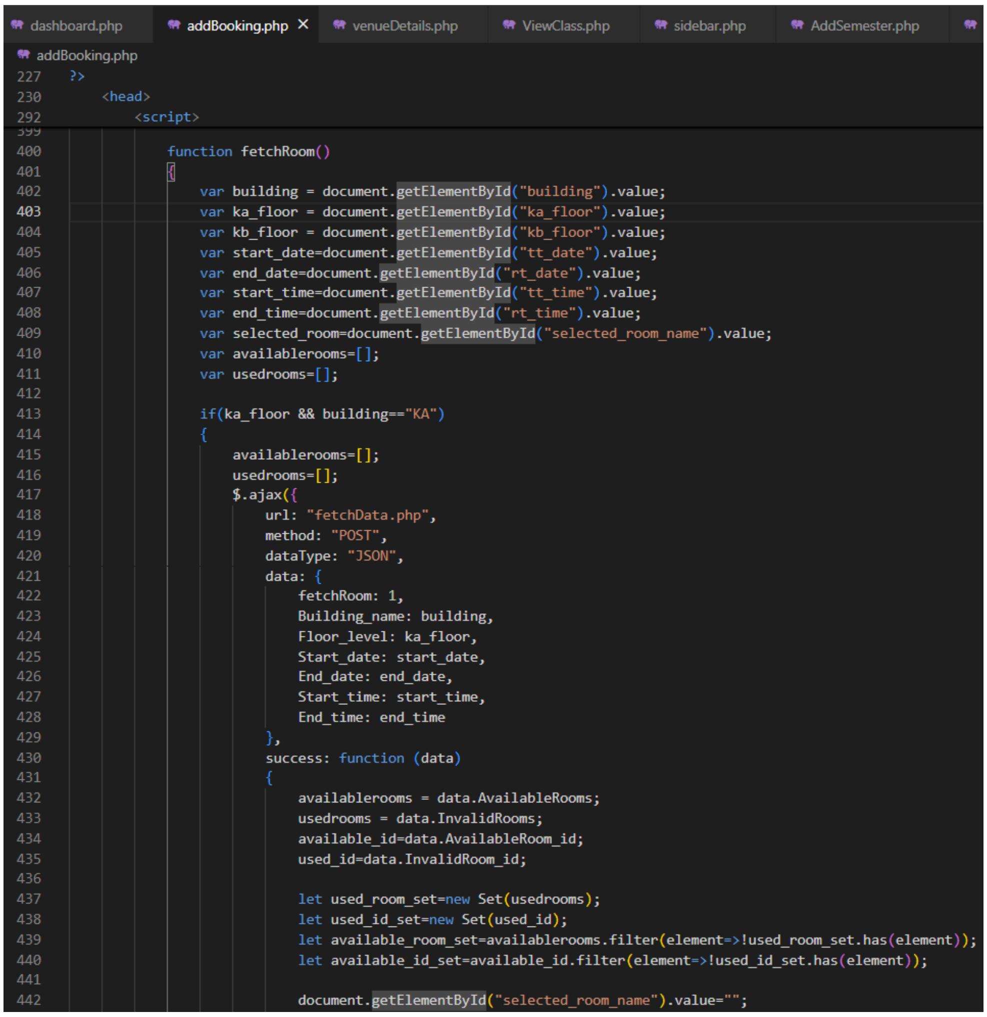
Selected Room

Booking Reason
Test

Available Rooms:

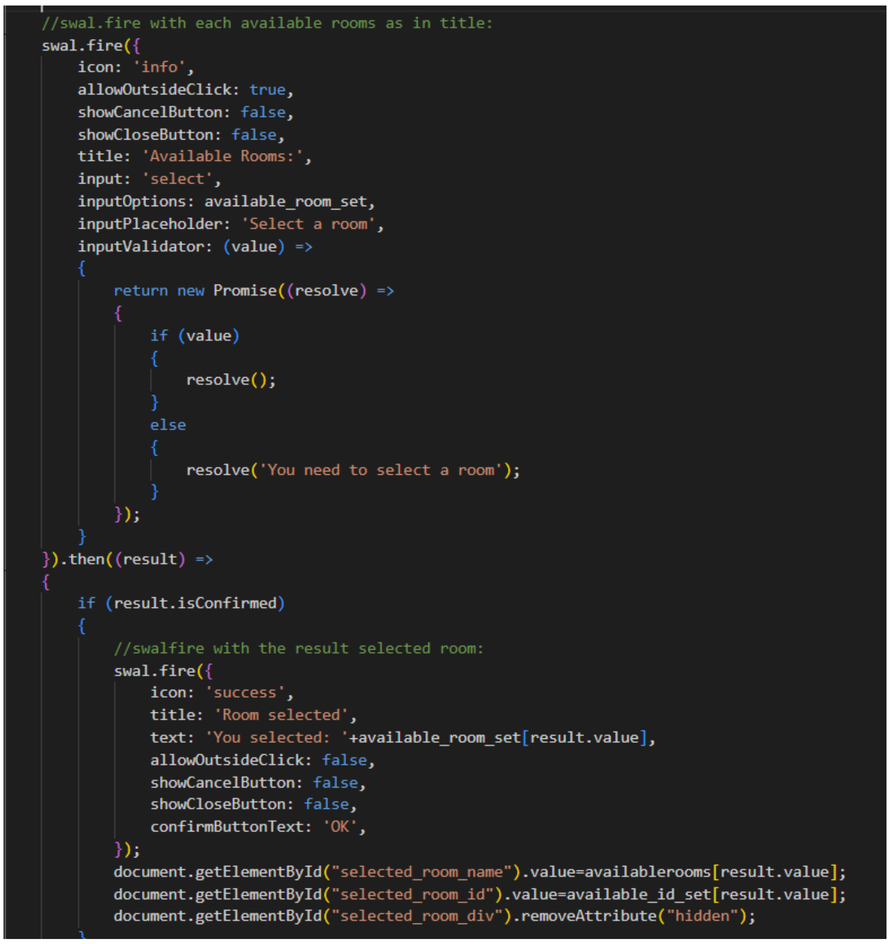
Select a room
- Select a room
- KB402
- KB403
- KB404

Figure 6.31: Choosing the venues



```
?>
    <head>
        <script>
            function fetchRoom()
            {
                var building = document.getElementById("building").value;
                var ka_floor = document.getElementById("ka_floor").value;
                var kb_floor = document.getElementById("kb_floor").value;
                var start_date=document.getElementById("tt_date").value;
                var end_date=document.getElementById("rt_date").value;
                var start_time=document.getElementById("tt_time").value;
                var end_time=document.getElementById("rt_time").value;
                var selected_room=document.getElementById("selected_room_name").value;
                var availablerooms=[];
                var usedrooms=[];

                if(ka_floor && building=="KA")
                {
                    availablerooms=[];
                    usedrooms=[];
                    $.ajax({
                        url: "fetchData.php",
                        method: "POST",
                        dataType: "JSON",
                        data: {
                            fetchRoom: 1,
                            Building_name: building,
                            Floor_level: ka_floor,
                            Start_date: start_date,
                            End_date: end_date,
                            Start_time: start_time,
                            End_time: end_time
                        },
                        success: function (data)
                        {
                            availablerooms = data.AvailableRooms;
                            usedrooms = data.InvalidRooms;
                            available_id=data.AvailableRoom_id;
                            used_id=data.InvalidRoom_id;

                            let used_room_set=new Set(usedrooms);
                            let used_id_set=new Set(used_id);
                            let available_room_set=availablerooms.filter(element=>!used_room_set.has(element));
                            let available_id_set=available_id.filter(element=>!used_id_set.has(element));

                            document.getElementById("selected_room_name").value="";
```

Figure 6.32: Function to read the input values first

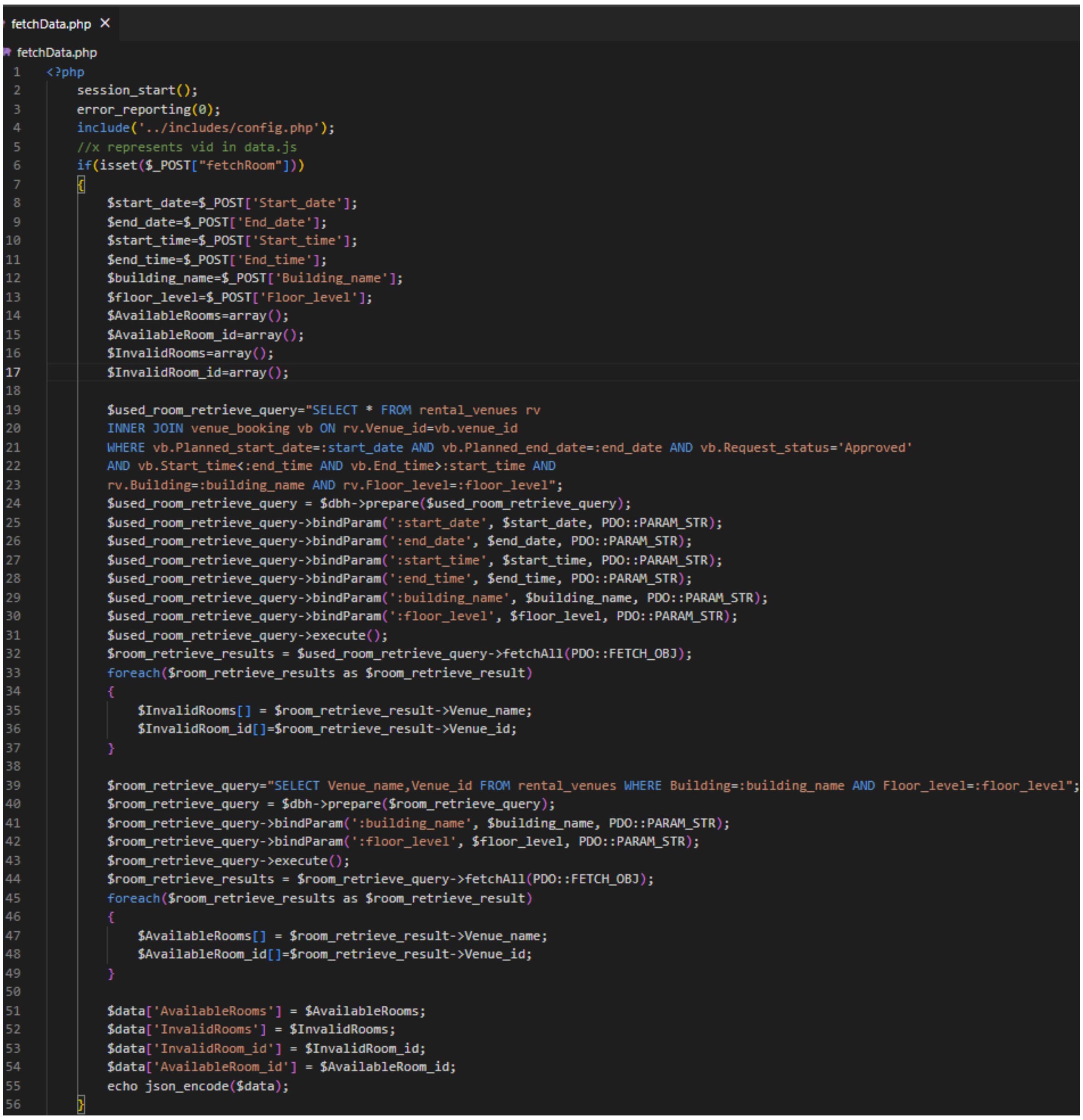


```
//swal.fire with each available rooms as in title:
swal.fire({
    icon: 'info',
    allowOutsideClick: true,
    showCancelButton: false,
    showCloseButton: false,
    title: 'Available Rooms:',
    input: 'select',
    inputOptions: available_room_set,
    inputPlaceholder: 'Select a room',
    inputValidator: (value) =>
    {
        return new Promise((resolve) =>
        {
            if (value)
            {
                resolve();
            }
            else
            {
                resolve('You need to select a room');
            }
        });
    }
}).then((result) =>
{
    if (result.isConfirmed)
    {
        //swalfire with the result selected room:
        swal.fire({
            icon: 'success',
            title: 'Room selected',
            text: 'You selected: '+available_room_set[result.value],
            allowOutsideClick: false,
            showCancelButton: false,
            showCloseButton: false,
            confirmButtonText: 'OK',
        });
        document.getElementById("selected_room_name").value=availablerooms[result.value];
        document.getElementById("selected_room_id").value=available_id_set[result.value];
        document.getElementById("selected_room_div").removeAttribute("hidden");
```

Figure 6.33: Popup shown to choose the venues

Noticing that the end-users need to select the venues in a pop-up method, this is achieved through implementing the SweetAlert library, whose aim is to enhance interactiveness. Fetching data wise, AJAX is used to exchange data asynchronously with another PHP page. Named as “fetchData.php”, the page consists of many separated cases where it communicates with the PhpMyAdmin to get data returned. Refer to Figure 6.33 to learn about the technical implementation.

To filter the available venues, a data separation has been done in this stage. Firstly, all venues will be fetched into an array first, regardless of it is being booked during that timing. Next, the venues that are being booked during that timing will be fetched into another array. Utilizing the array.filter() in PHP, the available rooms array will get all the booked venues’ name removed. That is how only the available venues will be shown in the selection list. Refer to Figure 6.32 to gain understanding of this function.



fetchData.php ×

fetchData.php

```php
<?php
    session_start();
    error_reporting(0);
    include('../includes/config.php');
    //x represents vid in data.js
    if(isset($_POST["fetchRoom"]))
    {
        $start_date=$_POST['Start_date'];
        $end_date=$_POST['End_date'];
        $start_time=$_POST['Start_time'];
        $end_time=$_POST['End_time'];
        $building_name=$_POST['Building_name'];
        $floor_level=$_POST['Floor_level'];
        $AvailableRooms=array();
        $AvailableRoom_id=array();
        $InvalidRooms=array();
        $InvalidRoom_id=array();

        $used_room_retrieve_query="SELECT * FROM rental_venues rv
        INNER JOIN venue_booking vb ON rv.Venue_id=vb.venue_id
        WHERE vb.Planned_start_date=:start_date AND vb.Planned_end_date=:end_date AND vb.Request_status='Approved'
        AND vb.Start_time<:end_time AND vb.End_time>:start_time AND
        rv.Building=:building_name AND rv.Floor_level=:floor_level";
        $used_room_retrieve_query = $dbh->prepare($used_room_retrieve_query);
        $used_room_retrieve_query->bindParam(':start_date', $start_date, PDO::PARAM_STR);
        $used_room_retrieve_query->bindParam(':end_date', $end_date, PDO::PARAM_STR);
        $used_room_retrieve_query->bindParam(':start_time', $start_time, PDO::PARAM_STR);
        $used_room_retrieve_query->bindParam(':end_time', $end_time, PDO::PARAM_STR);
        $used_room_retrieve_query->bindParam(':building_name', $building_name, PDO::PARAM_STR);
        $used_room_retrieve_query->bindParam(':floor_level', $floor_level, PDO::PARAM_STR);
        $used_room_retrieve_query->execute();
        $room_retrieve_results = $used_room_retrieve_query->fetchAll(PDO::FETCH_OBJ);
        foreach($room_retrieve_results as $room_retrieve_result)
        {
            $InvalidRooms[] = $room_retrieve_result->Venue_name;
            $InvalidRoom_id[]=$room_retrieve_result->Venue_id;
        }

        $room_retrieve_query="SELECT Venue_name,Venue_id FROM rental_venues WHERE Building=:building_name AND Floor_level=:floor_level";
        $room_retrieve_query = $dbh->prepare($room_retrieve_query);
        $room_retrieve_query->bindParam(':building_name', $building_name, PDO::PARAM_STR);
        $room_retrieve_query->bindParam(':floor_level', $floor_level, PDO::PARAM_STR);
        $room_retrieve_query->execute();
        $room_retrieve_results = $room_retrieve_query->fetchAll(PDO::FETCH_OBJ);
        foreach($room_retrieve_results as $room_retrieve_result)
        {
            $AvailableRooms[] = $room_retrieve_result->Venue_name;
            $AvailableRoom_id[]=$room_retrieve_result->Venue_id;
        }

        $data['AvailableRooms'] = $AvailableRooms;
        $data['InvalidRooms'] = $InvalidRooms;
        $data['InvalidRoom_id'] = $InvalidRoom_id;
        $data['AvailableRoom_id'] = $AvailableRoom_id;
        echo json_encode($data);
    }
```

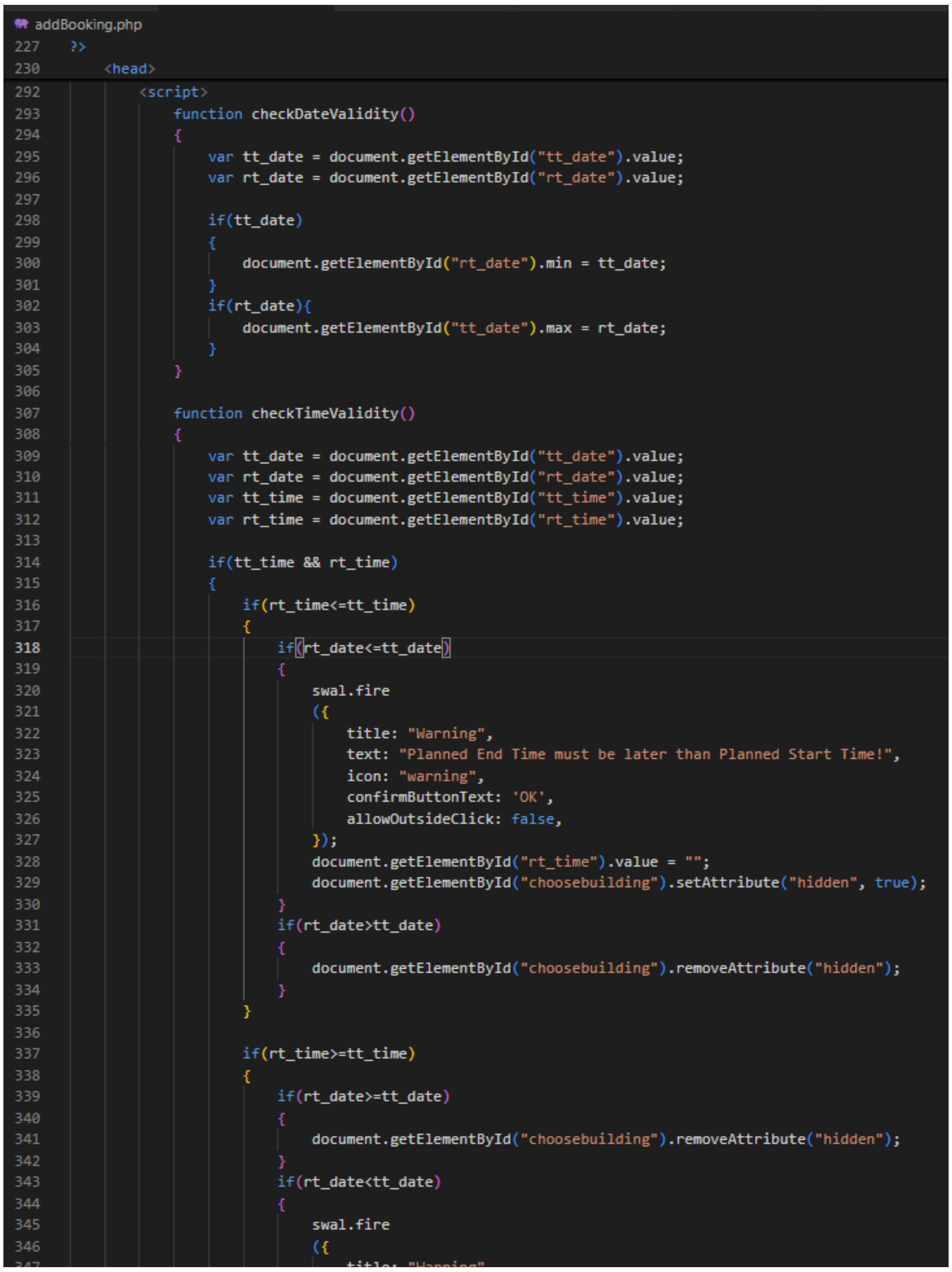
Figure 6.34: fetchData.php

On-spot input data validation has been done in this case as well, ensuring there are no logic conflicts arising before submitting a venue booking request. This had been done through JavaScript onchange functions, reading the dates, time values, booking reason and capacity. The validation criteria could be referred to as follows:

Table 6.2: Validation of booking data

Time values	**Date values**
- The ending time must be later than the starting time, within the same date	- The ending date must be later than the starting time

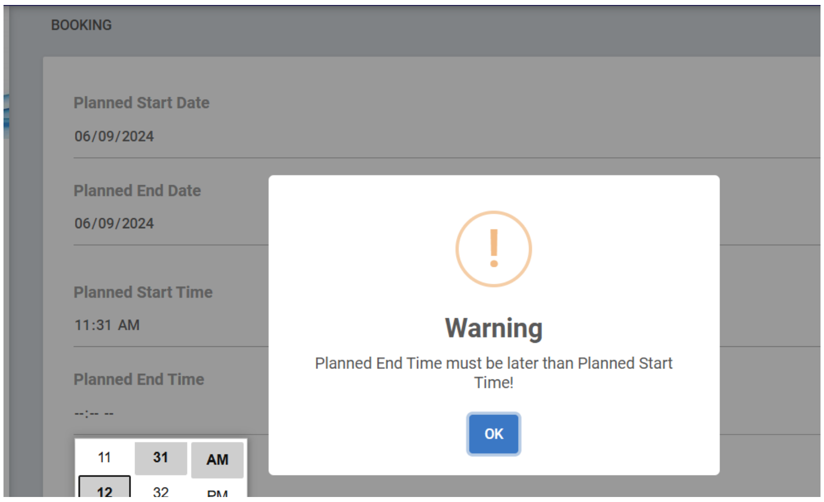
- There are no time values chosen - Choosing the current time on today's date	- There are no date values chosen - The chosen date is lesser than 5 days from current date
Selected room	**Lecturer's name** (for student)
- No rooms selected	- No lecturer's name in the column
Venue capacity	
- Input capacity exceeded the fetched venue's capacity count	



addBooking.php

```
    ?>
        <head>
            <script>
                function checkDateValidity()
                {
                    var tt_date = document.getElementById("tt_date").value;
                    var rt_date = document.getElementById("rt_date").value;

                    if(tt_date)
                    {
                        document.getElementById("rt_date").min = tt_date;
                    }
                    if(rt_date){
                        document.getElementById("tt_date").max = rt_date;
                    }
                }

                function checkTimeValidity()
                {
                    var tt_date = document.getElementById("tt_date").value;
                    var rt_date = document.getElementById("rt_date").value;
                    var tt_time = document.getElementById("tt_time").value;
                    var rt_time = document.getElementById("rt_time").value;

                    if(tt_time && rt_time)
                    {
                        if(rt_time<=tt_time)
                        {
                            if(rt_date<=tt_date)
                            {
                                swal.fire
                                ({
                                    title: "Warning",
                                    text: "Planned End Time must be later than Planned Start Time!",
                                    icon: "warning",
                                    confirmButtonText: 'OK',
                                    allowOutsideClick: false,
                                });
                                document.getElementById("rt_time").value = "";
                                document.getElementById("choosebuilding").setAttribute("hidden", true);
                            }
                            if(rt_date>tt_date)
                            {
                                document.getElementById("choosebuilding").removeAttribute("hidden");
                            }
                        }

                        if(rt_time>=tt_time)
                        {
                            if(rt_date>=tt_date)
                            {
                                document.getElementById("choosebuilding").removeAttribute("hidden");
                            }
                            if(rt_date<tt_date)
                            {
                                swal.fire
                                ({
```

Figure 6.35: Technical implementation on validation functions


BOOKING

Planned Start Date
06/09/2024

Planned End Date
06/09/2024

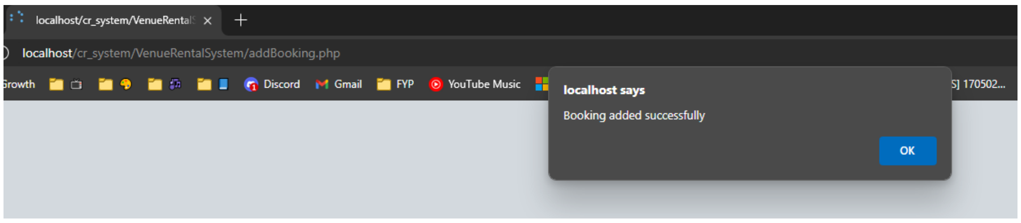
Planned Start Time
11:31 AM

Planned End Time
--:-- --

Warning
Planned End Time must be later than Planned Start Time!

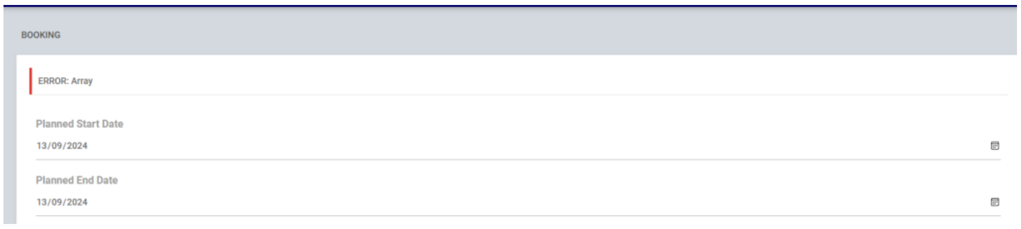
Figure 6.36: Example of validation

After submitting a valid booking request, then the user will be alerted with a message that booking has been added successfully, triggering the function of sending emails to the DGS admin and chosen lecturer (for student only), at the end bringing back the end-user to the same page.


localhost/cr_system/VenueRentalSystem/addBooking.php

localhost says
Booking added successfully

Figure 6.37: Message of booking added


BOOKING

ERROR: Array

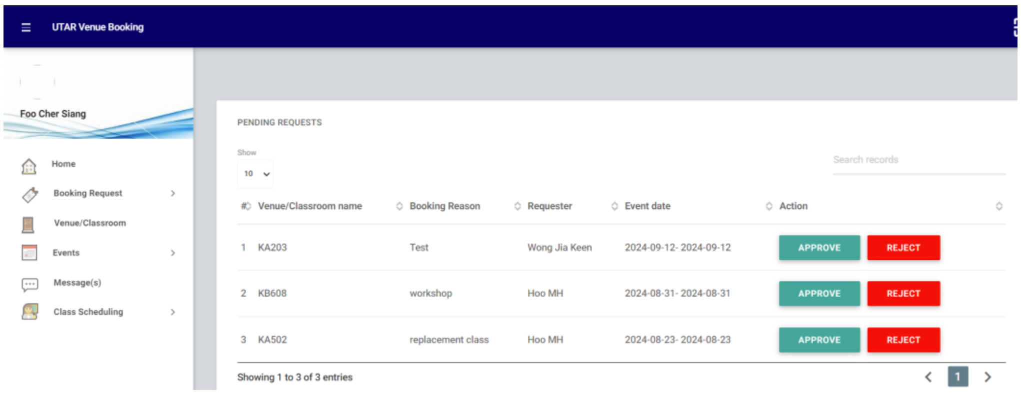
Planned Start Date
13/09/2024

Planned End Date
13/09/2024

Figure 6.38: Failed scenario of booking due to data incompletion

6.3.6 Venue/Classroom booking management

Upon submitting the venue/classroom booking request, the initial status will be “Pending for approval from DGS and HOD”. The ultimate approval decision lies on the DGS side, however if the HOD rejects, then the request is directly being rejected. These approval actions could be done by real-time interactions in the interface.



PENDING REQUESTS

Show 10

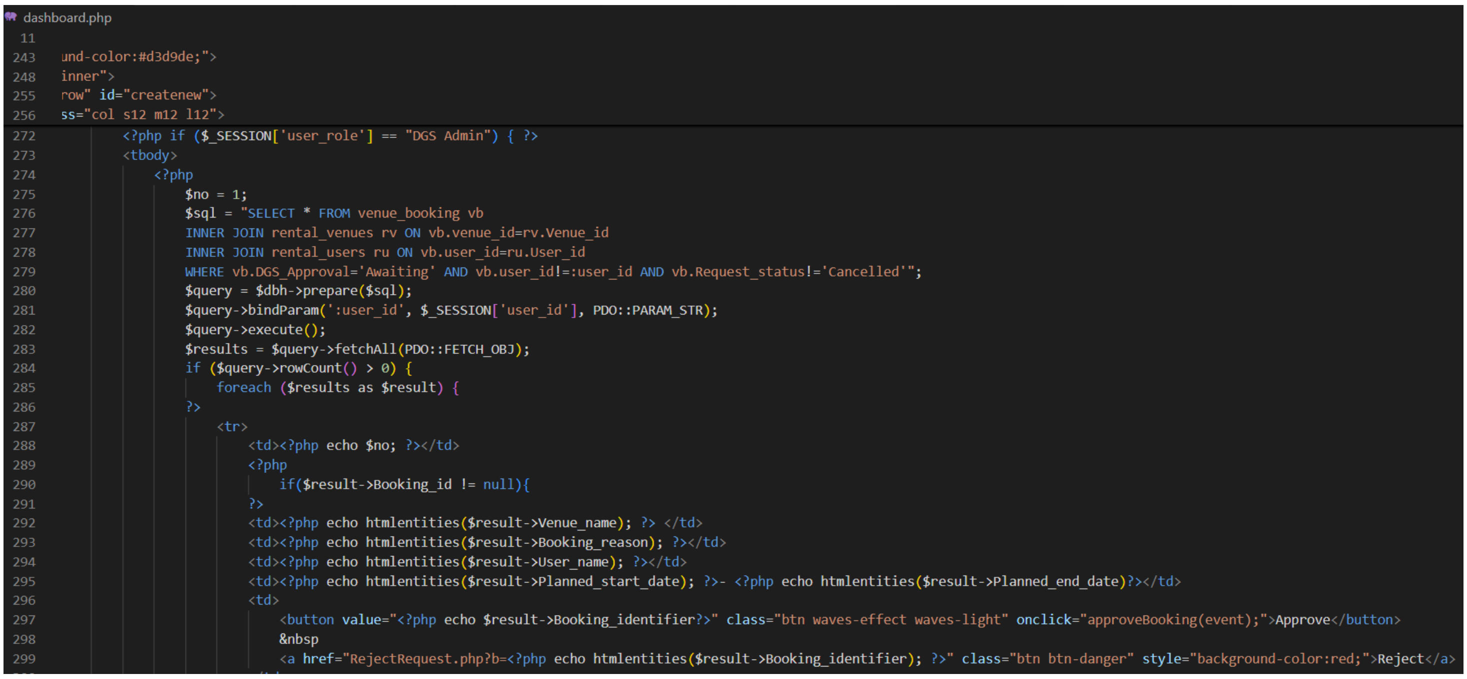
Search records

#	Venue/Classroom name	Booking Reason	Requester	Event date	Action
1	KA203	Test	Wong Jia Keen	2024-09-12- 2024-09-12	APPROVE REJECT
2	KB608	workshop	Hoo MH	2024-08-31- 2024-08-31	APPROVE REJECT
3	KA502	replacement class	Hoo MH	2024-08-23- 2024-08-23	APPROVE REJECT

Showing 1 to 3 of 3 entries

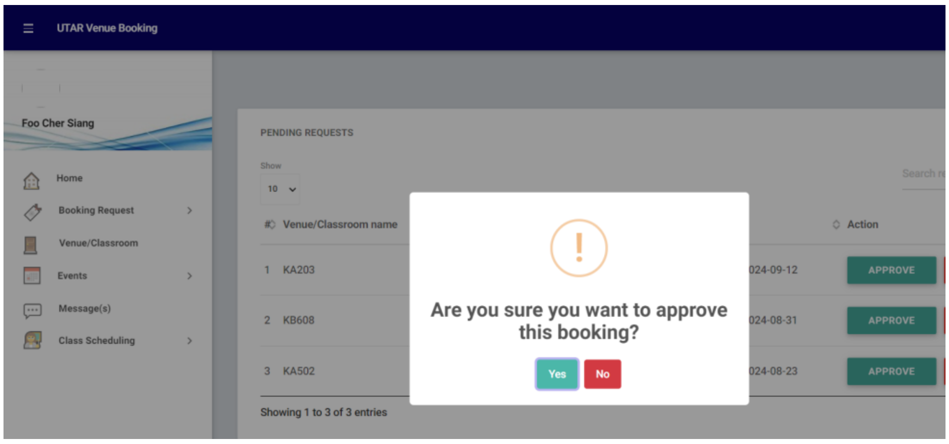
Figure 6.39: Dashboard interface whose DGS staff could manage booking requests

The booking management interaction and logic works the same for DGS staff, HOD and lecturers. The approve and reject buttons carry the identifier value for the specific booking request in the same row. If the approve button is clicked, then a pop-up would be displayed, confirming the actions. Again, this is done by utilizing the SweetAlert library. The following figure depicts the technical operation of the buttons.


dashboard.php

```
 11
243   und-color:#d3d9de;">
248   inner">
255   row" id="createnew">
256   ss="col s12 m12 l12">
272           <?php if ($_SESSION['user_role'] == "DGS Admin") { ?>
273           <tbody>
274               <?php
275                   $no = 1;
276                   $sql = "SELECT * FROM venue_booking vb
277                   INNER JOIN rental_venues rv ON vb.venue_id=rv.Venue_id
278                   INNER JOIN rental_users ru ON vb.user_id=ru.User_id
279                   WHERE vb.DGS_Approval='Awaiting' AND vb.user_id!=:user_id AND vb.Request_status!='Cancelled'";
280                   $query = $dbh->prepare($sql);
281                   $query->bindParam(':user_id', $_SESSION['user_id'], PDO::PARAM_STR);
282                   $query->execute();
283                   $results = $query->fetchAll(PDO::FETCH_OBJ);
284                   if ($query->rowCount() > 0) {
285                       foreach ($results as $result) {
286                   ?>
287                       <tr>
288                           <td><?php echo $no; ?></td>
289                           <?php
290                               if($result->Booking_id != null){
291                           ?>
292                           <td><?php echo htmlentities($result->Venue_name); ?> </td>
293                           <td><?php echo htmlentities($result->Booking_reason); ?></td>
294                           <td><?php echo htmlentities($result->User_name); ?></td>
295                           <td><?php echo htmlentities($result->Planned_start_date); ?>- <?php echo htmlentities($result->Planned_end_date)?></td>
296                           <td>
297                               <button value="<?php echo $result->Booking_identifier?>" class="btn waves-effect waves-light" onclick="approveBooking(event);">Approve</button>
298                               &nbsp
299                               <a href="RejectRequest.php?b=<?php echo htmlentities($result->Booking_identifier); ?>" class="btn btn-danger" style="background-color:red;">Reject</a>
```

Figure 6.40: Fetching the booking request data with buttons in lines 297, 299



PENDING REQUESTS

#	Venue/Classroom name		Action
1	KA203	024-09-12	APPROVE
2	KB608	024-08-31	APPROVE
3	KA502	024-08-23	APPROVE

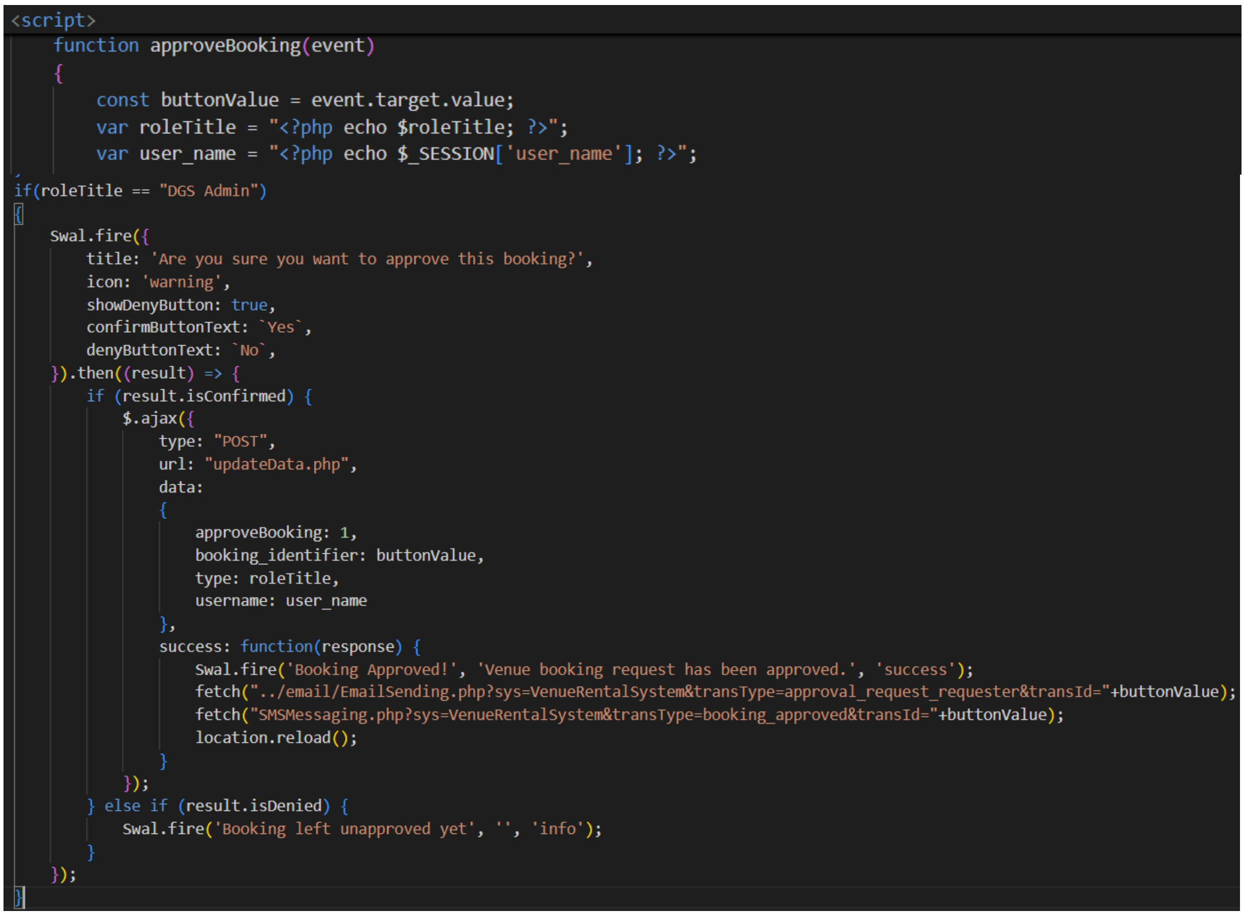
Showing 1 to 3 of 3 entries

Are you sure you want to approve this booking?

Yes No

Figure 6.41: Approval pop-up

Approve action would send status updates on the booking request in the database. The approval person’s name, approval timing and approval decision would be recorded in the Venue_booking’s table. Alike the room fetching function described previously, this update could be achieved through AJAX. The following figure describes the technical implementation for the approval decision.

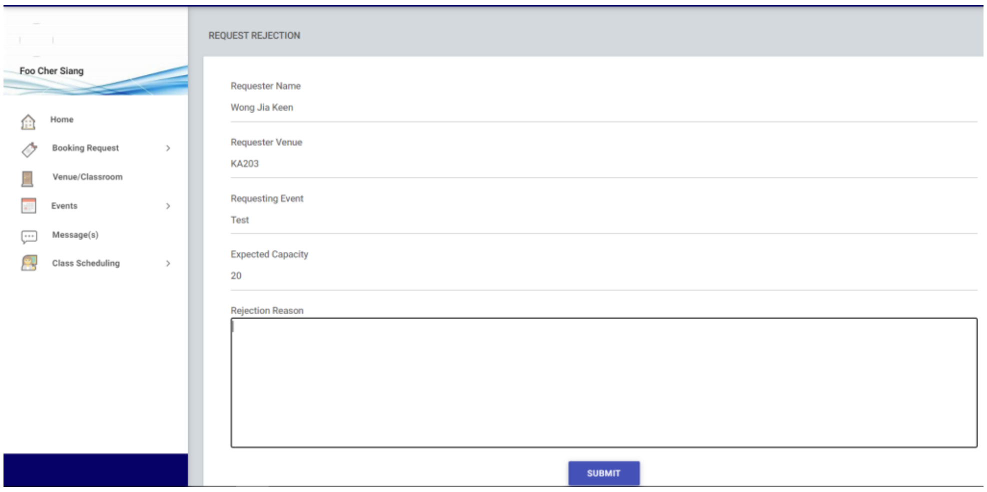


```
<script>
    function approveBooking(event)
    {
        const buttonValue = event.target.value;
        var roleTitle = "<?php echo $roleTitle; ?>";
        var user_name = "<?php echo $_SESSION['user_name']; ?>";
if(roleTitle == "DGS Admin")
{
    Swal.fire({
        title: 'Are you sure you want to approve this booking?',
        icon: 'warning',
        showDenyButton: true,
        confirmButtonText: `Yes`,
        denyButtonText: `No`,
    }).then((result) => {
        if (result.isConfirmed) {
            $.ajax({
                type: "POST",
                url: "updateData.php",
                data:
                {
                    approveBooking: 1,
                    booking_identifier: buttonValue,
                    type: roleTitle,
                    username: user_name
                },
                success: function(response) {
                    Swal.fire('Booking Approved!', 'Venue booking request has been approved.', 'success');
                    fetch("../email/EmailSending.php?sys=VenueRentalSystem&transType=approval_request_requester&transId="+buttonValue);
                    fetch("SMSMessaging.php?sys=VenueRentalSystem&transType=booking_approved&transId="+buttonValue);
                    location.reload();
                }
            });
        } else if (result.isDenied) {
            Swal.fire('Booking left unapproved yet', '', 'info');
        }
    });
}
```

Figure 6.42: Technical implementation of booking approval

At the same time, the booking requester would get notified of the approval via SMS and email. This is done through PhpMailer and Bulk360 SMS API which would be described in section 6.4.

However, the rejection could redirect the end-user to the rejection page, where he/she needs to fill in the rejection reason. Once submitted, then the booking request is updated as "Rejected", also adding another entry of data into the "Rejected_booking" table.



Requester Name
Wong Jia Keen

Requester Venue
KA203

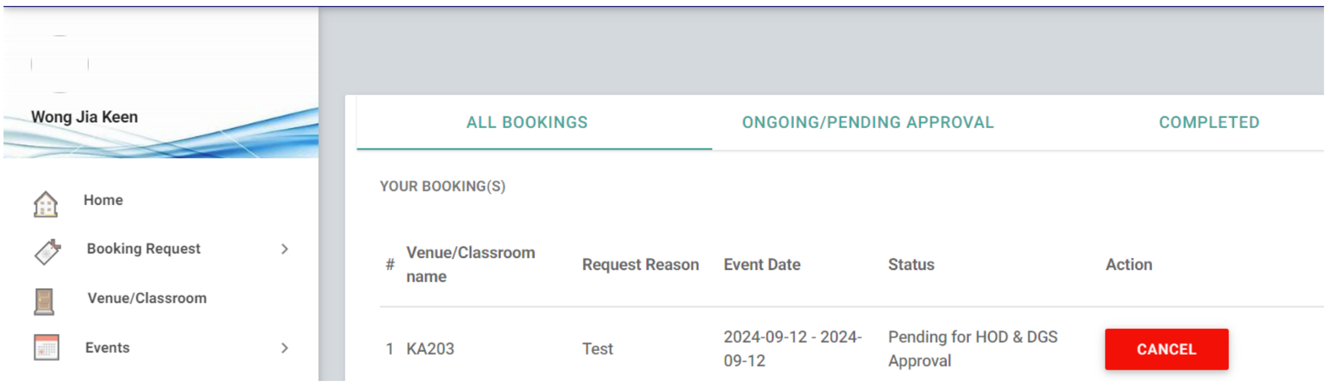
Requesting Event
Test

Expected Capacity
20

Rejection Reason

Figure 6.43: Rejection page

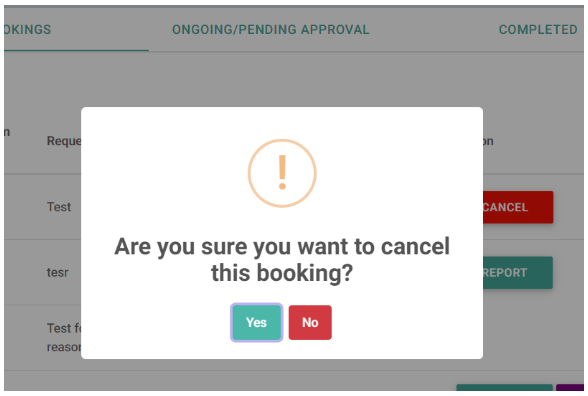
For managing own requests, the booking requester could choose to cancel their uncompleted booking request. It works like the approve button. Once cancelled, the status for the booking request would be updated to 'Cancelled' and an notification email will be sent to the DGS staff.



YOUR BOOKING(S)

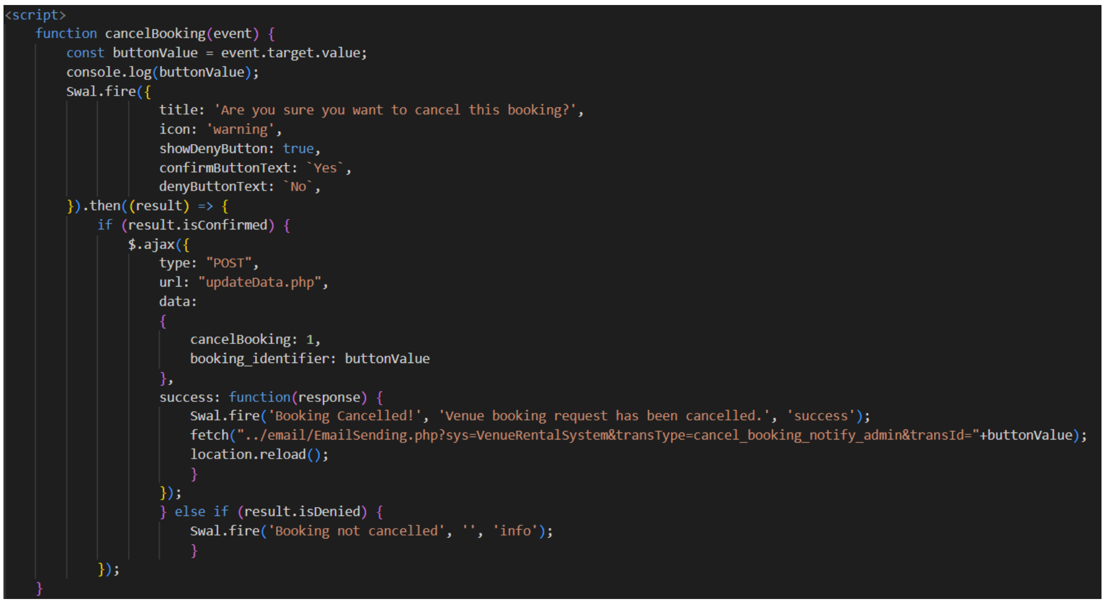
#	Venue/Classroom name	Request Reason	Event Date	Status	Action
1	KA203	Test	2024-09-12 - 2024-09-12	Pending for HOD & DGS Approval	CANCEL

Figure 6.44: Cancel button



Are you sure you want to cancel this booking?

Figure 6.45: Cancel pop-up



```
<script>
    function cancelBooking(event) {
        const buttonValue = event.target.value;
        console.log(buttonValue);
        Swal.fire({
                        title: 'Are you sure you want to cancel this booking?',
                        icon: 'warning',
                        showDenyButton: true,
                        confirmButtonText: `Yes`,
                        denyButtonText: `No`,
        }).then((result) => {
            if (result.isConfirmed) {
                $.ajax({
                    type: "POST",
                    url: "updateData.php",
                    data:
                    {
                        cancelBooking: 1,
                        booking_identifier: buttonValue
                    },
                    success: function(response) {
                        Swal.fire('Booking Cancelled!', 'Venue booking request has been cancelled.', 'success');
                        fetch("../email/EmailSending.php?sys=VenueRentalSystem&transType=cancel_booking_notify_admin&transId="+buttonValue);
                        location.reload();
                        }
                    });
                    } else if (result.isDenied) {
                        Swal.fire('Booking not cancelled', '', 'info');
                        }
            });
    }
```

Figure 6.46: Technical implementation of cancel booking

6.3.7 Events

There are requirements stated that some parties are unaware of which date has calendar events that may cause inconveniences in booking UTAR venue/classrooms. Therefore, this is the point of such a feature being implemented in this system.

6.3.7.1 Viewing

Accessible by every level of end-users, this function could be found in the “View Event” page hiding under the “Event” tab in the sidebar. This function enables the end-user to check the calendar to view any calendar events.

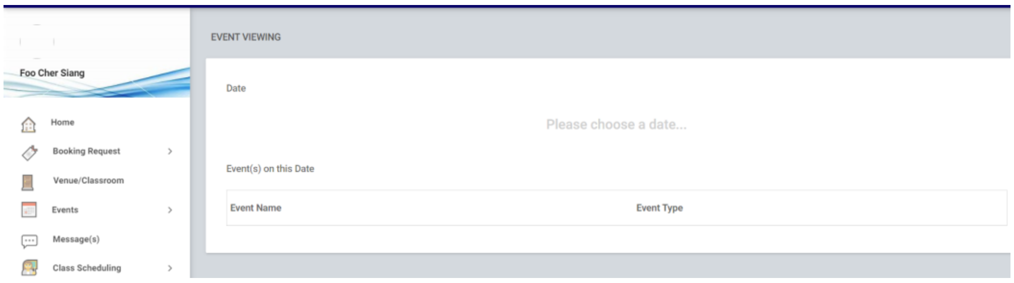


Figure 6.47: Interface of events viewing

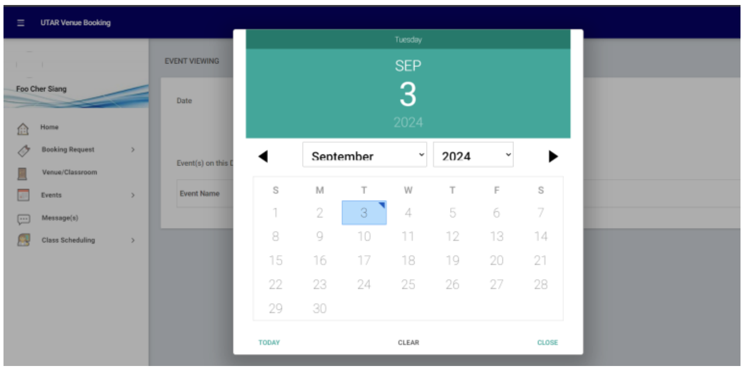
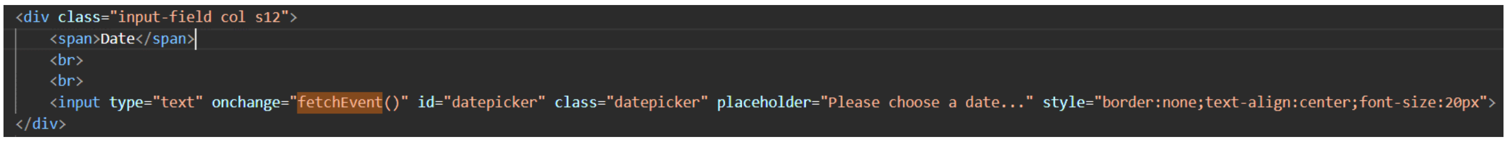


Figure 6.48: Calendar interface

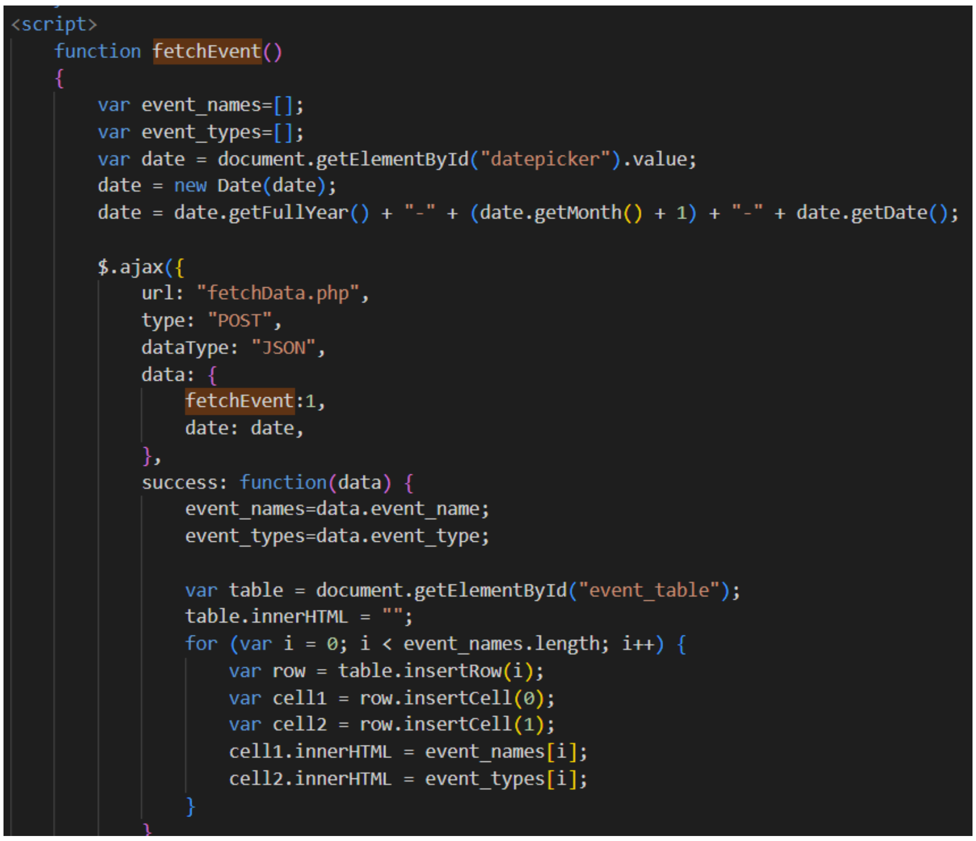
To enhance the level of interaction, the pickadate.js library was implemented, replacing the traditional calendar to choose dates function. After a date is chosen, then any relevant events added to this date will be displayed on the table underneath, showing the event name and the event type.

The fetching data is done through AJAX with external PHP script information interexchange. Since the date values could vary based on user input, then onchange() function is suitable to apply to the date input element. Through reading the changing date value in JavaScript, the date values are sent to the script to check with the database. Immediate data fetching is performed in this way, referring to the following figures.



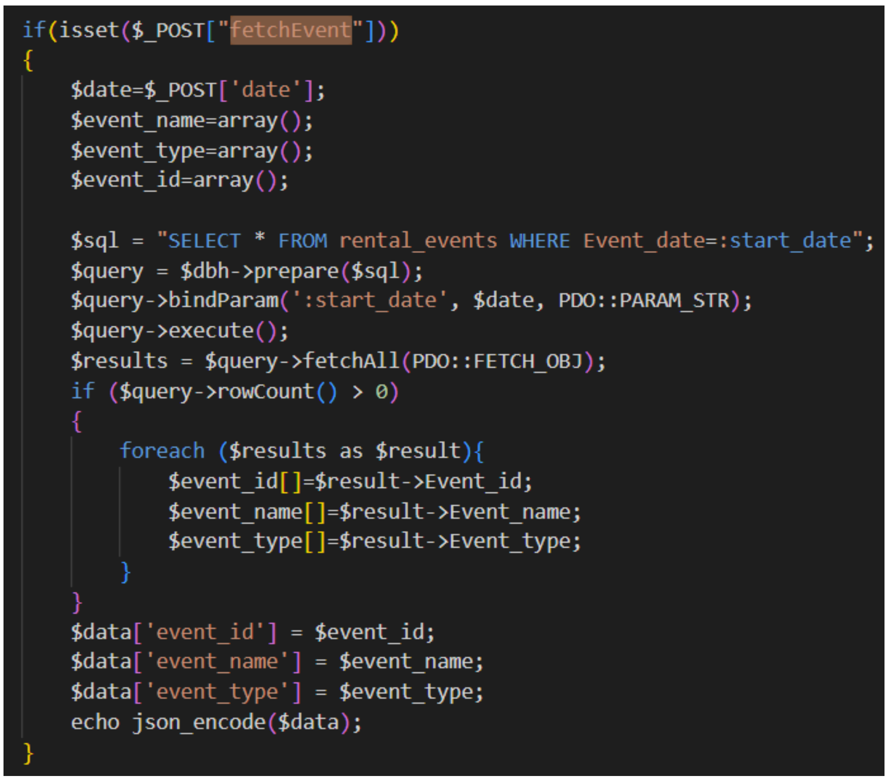
```
<div class="input-field col s12">
    <span>Date</span>
    <br>
    <br>
    <input type="text" onchange="fetchEvent()" id="datepicker" class="datepicker" placeholder="Please choose a date..." style="border:none;text-align:center;font-size:20px">
</div>
```

Figure 6.49: Onchange function implemented on the input element



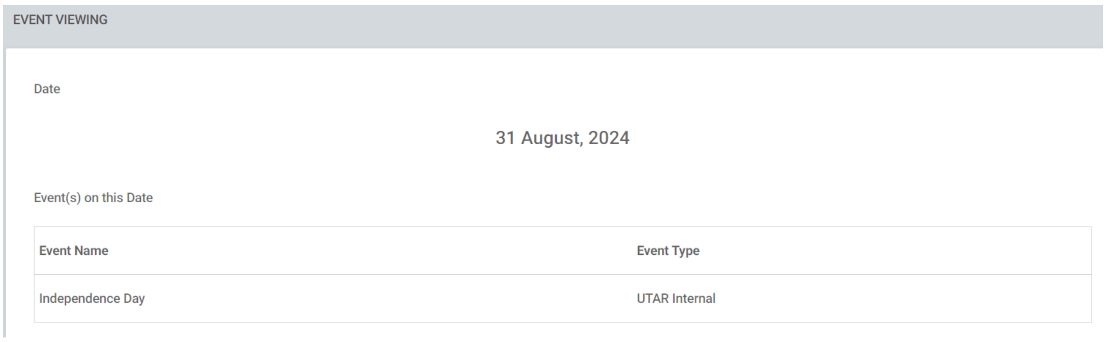
```
<script>
    function fetchEvent()
    {
        var event_names=[];
        var event_types=[];
        var date = document.getElementById("datepicker").value;
        date = new Date(date);
        date = date.getFullYear() + "-" + (date.getMonth() + 1) + "-" + date.getDate();

        $.ajax({
            url: "fetchData.php",
            type: "POST",
            dataType: "JSON",
            data: {
                fetchEvent:1,
                date: date,
            },
            success: function(data) {
                event_names=data.event_name;
                event_types=data.event_type;

                var table = document.getElementById("event_table");
                table.innerHTML = "";
                for (var i = 0; i < event_names.length; i++) {
                    var row = table.insertRow(i);
                    var cell1 = row.insertCell(0);
                    var cell2 = row.insertCell(1);
                    cell1.innerHTML = event_names[i];
                    cell2.innerHTML = event_types[i];
                }
```

Figure 6.50: Javascript function that reads the date and fetches events' information



```
if(isset($_POST["fetchEvent"]))
{
    $date=$_POST['date'];
    $event_name=array();
    $event_type=array();
    $event_id=array();

    $sql = "SELECT * FROM rental_events WHERE Event_date=:start_date";
    $query = $dbh->prepare($sql);
    $query->bindParam(':start_date', $date, PDO::PARAM_STR);
    $query->execute();
    $results = $query->fetchAll(PDO::FETCH_OBJ);
    if ($query->rowCount() > 0)
    {
        foreach ($results as $result){
            $event_id[]=$result->Event_id;
            $event_name[]=$result->Event_name;
            $event_type[]=$result->Event_type;
        }
    }
    $data['event_id'] = $event_id;
    $data['event_name'] = $event_name;
    $data['event_type'] = $event_type;
    echo json_encode($data);
}
```

Figure 6.51: fetchData.php that searches for events' information asynchronously



EVENT VIEWING

Date

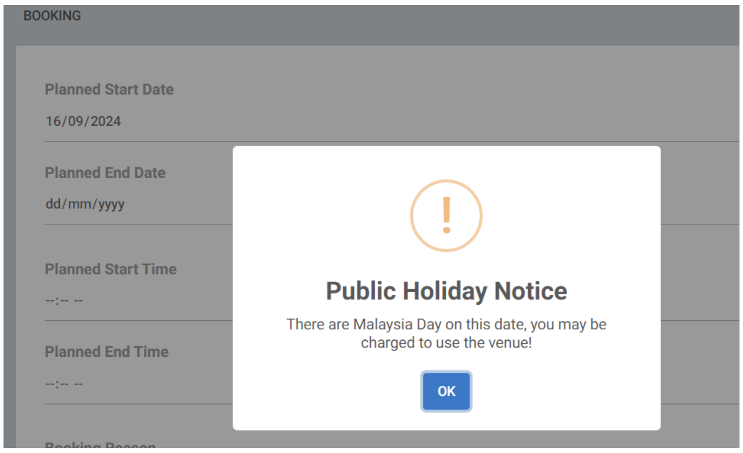
31 August, 2024

Event(s) on this Date

Event Name	Event Type
Independence Day	UTAR Internal

Figure 6.52: Example of event data shown

To relate to the add venue/classroom usage request, whenever the user chooses the dates that have calendar events, then the user is warned with a pop-up, hinting that there would be an event to be held on the date. The reminder appears as it could affect the approval decision if the user chooses to book a venue on that date, rejection due to rescheduling might happen in this case.



BOOKING

Planned Start Date
16/09/2024

Planned End Date
dd/mm/yyyy

Planned Start Time
--:-- --

Planned End Time
--:-- --

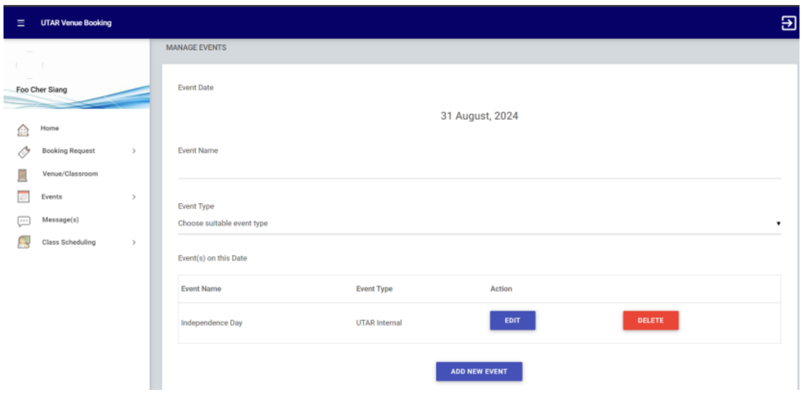
Public Holiday Notice

There are Malaysia Day on this date, you may be charged to use the venue!

Figure 6.53: Events relating to add booking

6.3.7.2 Management

Accessible by only DGS staff and HOD, this function could be found in the “Manage Event(s)” page hiding under the “Event” tab in the sidebar.



MANAGE EVENTS

Event Date

31 August, 2024

Event Name

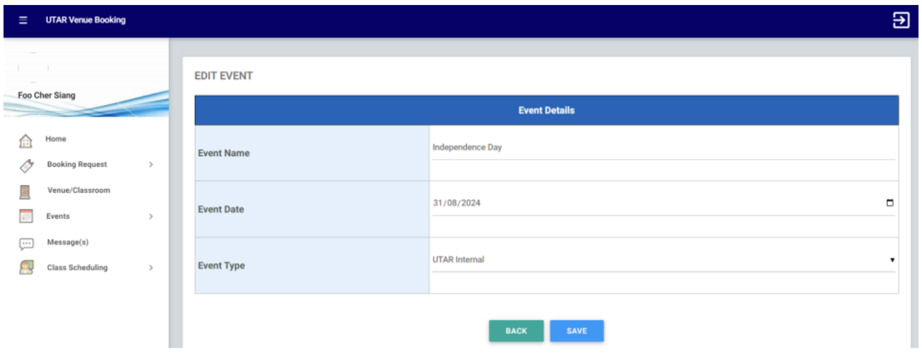
Event Type
Choose suitable event type

Event(s) on this Date

Event Name	Event Type	Action	
Independence Day	UTAR Internal	EDIT	DELETE

Figure 6.54: Manage event interface

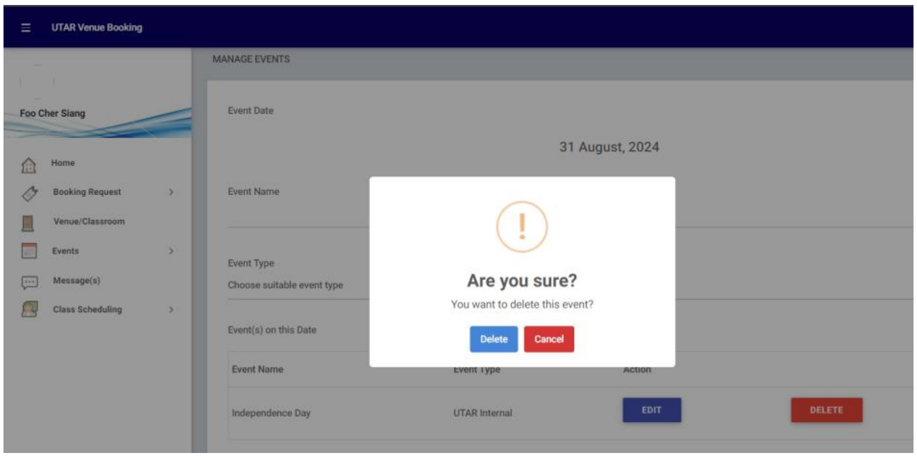
Event viewing could be performed on this page as well, working like the Event Viewing page, just that there are edit and delete buttons presented in the table. Working like the booking request management, each button hold the unique ID value for the fetched event. Editing will lead the end-user to the editing page, whereas choosing to delete will trigger the pop-up as a confirmation.



EDIT EVENT

Event Details	
Event Name	Independence Day
Event Date	31/08/2024
Event Type	UTAR Internal

Figure 6.55: Event editing page



Event Date

31 August, 2024

Event Name

Event Type

Choose suitable event type

Event(s) on this Date

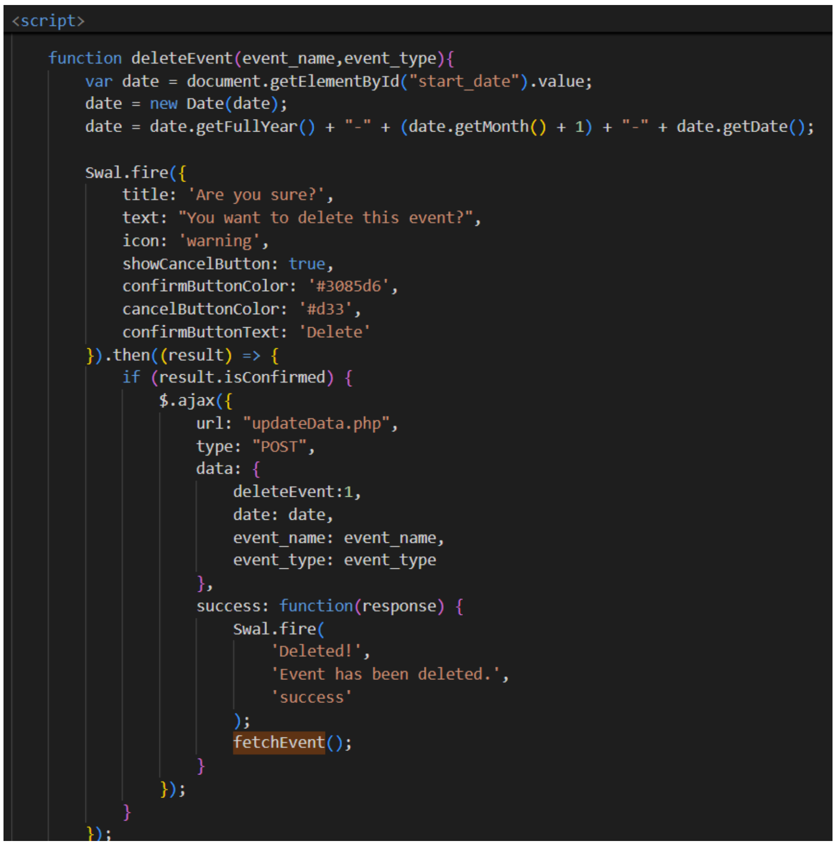
Event Name	Event Type	Action
Independence Day	UTAR Internal	EDIT DELETE

Are you sure?

You want to delete this event?

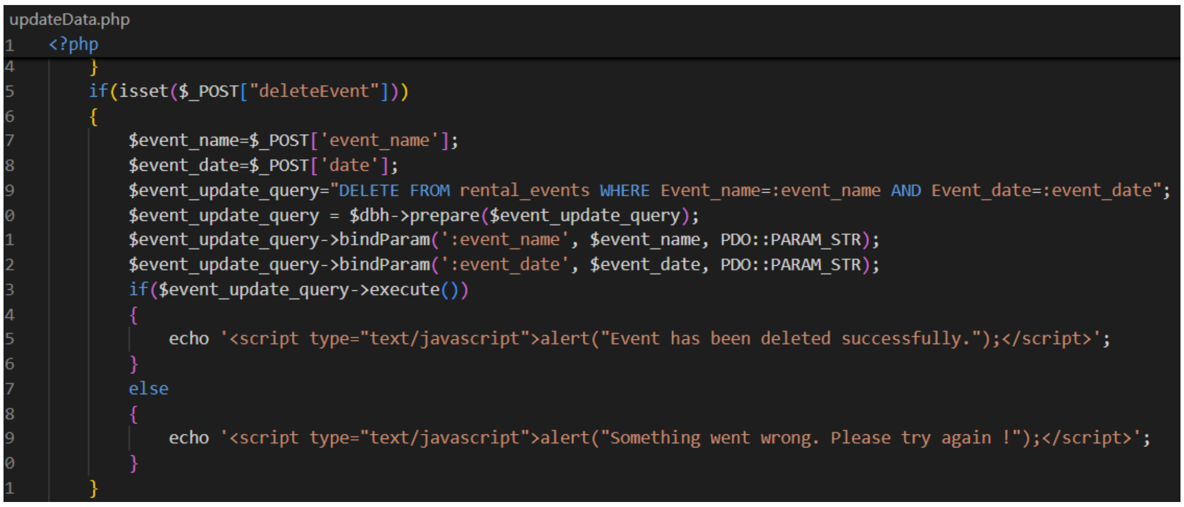
Figure 6.56: Event deletion pop-up

The event deletion works in AJAX. By default, the cancel button in this page carries the event name and the event type. Whenever the cancel button is clicked, then it triggers the JavaScript function to read the values and display the SweetAlert pop-up. Upon confirmation, it works with the external updateData.php file to request the database to be deleted. After deletion, the event fetching function is reloaded again, to make sure the deletion effect takes place. The following figures depict the operation of event deletion.



```
<script>
    function deleteEvent(event_name,event_type){
        var date = document.getElementById("start_date").value;
        date = new Date(date);
        date = date.getFullYear() + "-" + (date.getMonth() + 1) + "-" + date.getDate();

        Swal.fire({
            title: 'Are you sure?',
            text: "You want to delete this event?",
            icon: 'warning',
            showCancelButton: true,
            confirmButtonColor: '#3085d6',
            cancelButtonColor: '#d33',
            confirmButtonText: 'Delete'
        }).then((result) => {
            if (result.isConfirmed) {
                $.ajax({
                    url: "updateData.php",
                    type: "POST",
                    data: {
                        deleteEvent:1,
                        date: date,
                        event_name: event_name,
                        event_type: event_type
                    },
                    success: function(response) {
                        Swal.fire(
                            'Deleted!',
                            'Event has been deleted.',
                            'success'
                        );
                        fetchEvent();
                    }
                });
            }
        });
```

Figure 6.57: Delete event JavaScript function



```
updateData.php
<?php
    }
    if(isset($_POST["deleteEvent"]))
    {
        $event_name=$_POST['event_name'];
        $event_date=$_POST['date'];
        $event_update_query="DELETE FROM rental_events WHERE Event_name=:event_name AND Event_date=:event_date";
        $event_update_query = $dbh->prepare($event_update_query);
        $event_update_query->bindParam(':event_name', $event_name, PDO::PARAM_STR);
        $event_update_query->bindParam(':event_date', $event_date, PDO::PARAM_STR);
        if($event_update_query->execute())
        {
            echo '<script type="text/javascript">alert("Event has been deleted successfully.");</script>';
        }
        else
        {
            echo '<script type="text/javascript">alert("Something went wrong. Please try again !");</script>';
        }
    }
```

Figure 6.58: External PHP file running deletion

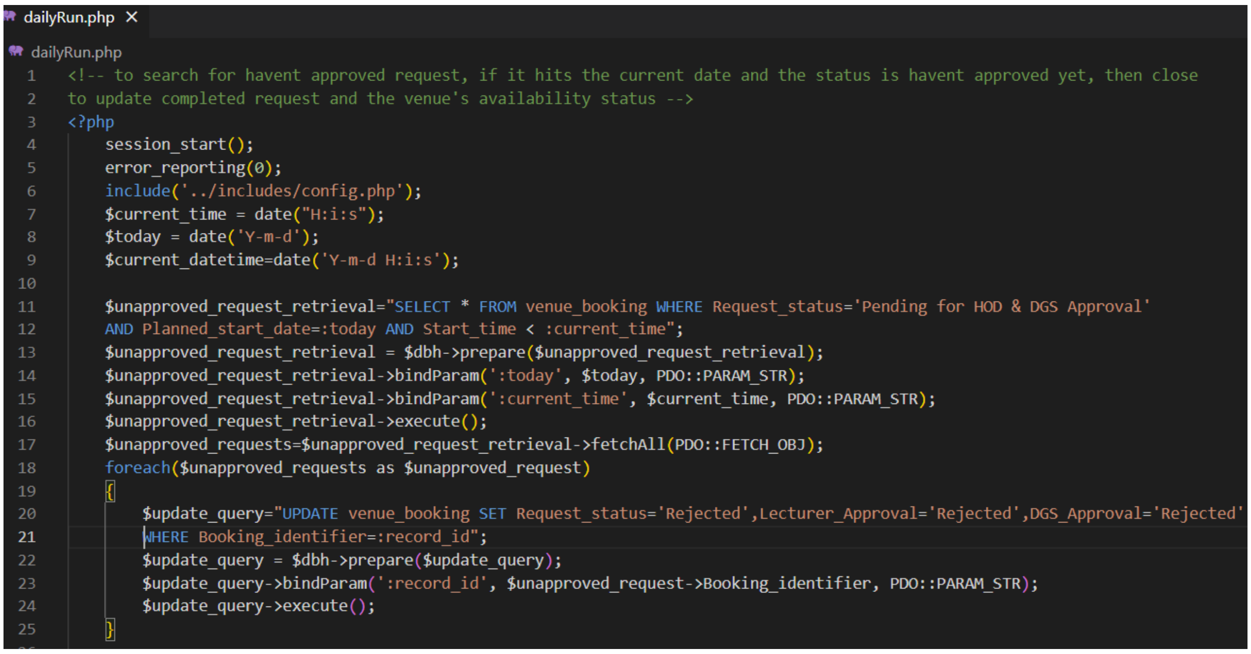
If the DGS staff or HOD wants to add an event, they could just enter the information in the input columns. Again, the date, name and the event type for the event to be added will be validated. Whenever one of the details is left empty, then the data insertion is failed. In detail, if the end-user fails to provide the event name or date, or does not choose the event type, then relevant error messages will be displayed to the user.

6.3.8 Autorun

This section relates to the 3.2 non-functional requirements listed in the Reliability aspect. It performs data checking and updating in the background whenever the system is loaded. This idea is to automate manual tasks that do not involve human review. To keep such tasks continue running, the functions listed in this section are declared in an external PHP file, that is included in every interface in this project.

6.3.8.1 Rejecting sudden venue booking requests

There are situations where an end-user may submit a venue/classroom usage request urgently that the DGS staff or HOD may overlook to miss it for approval decision. To perform this, the database needs to check if are there any venue/classroom requests where its status is still waiting for pending, plus if the request is on the current date and the usage's starting time is over the current time. If there is a record that matches all these conditions, then the record will be retrieved and updated for being rejected by both parties. Refer to the following figure to learn about the technical implementation.



dailyRun.php

```php
<!-- to search for havent approved request, if it hits the current date and the status is havent approved yet, then close
to update completed request and the venue's availability status -->
<?php
    session_start();
    error_reporting(0);
    include('../includes/config.php');
    $current_time = date("H:i:s");
    $today = date('Y-m-d');
    $current_datetime=date('Y-m-d H:i:s');

    $unapproved_request_retrieval="SELECT * FROM venue_booking WHERE Request_status='Pending for HOD & DGS Approval'
    AND Planned_start_date=:today AND Start_time < :current_time";
    $unapproved_request_retrieval = $dbh->prepare($unapproved_request_retrieval);
    $unapproved_request_retrieval->bindParam(':today', $today, PDO::PARAM_STR);
    $unapproved_request_retrieval->bindParam(':current_time', $current_time, PDO::PARAM_STR);
    $unapproved_request_retrieval->execute();
    $unapproved_requests=$unapproved_request_retrieval->fetchAll(PDO::FETCH_OBJ);
    foreach($unapproved_requests as $unapproved_request)
    {
        $update_query="UPDATE venue_booking SET Request_status='Rejected',Lecturer_Approval='Rejected',DGS_Approval='Rejected'
        WHERE Booking_identifier=:record_id";
        $update_query = $dbh->prepare($update_query);
        $update_query->bindParam(':record_id', $unapproved_request->Booking_identifier, PDO::PARAM_STR);
        $update_query->execute();
    }
```

Figure 6.59: Background-running script that checks for urgent venue booking

6.3.8.2 Updating venues' status

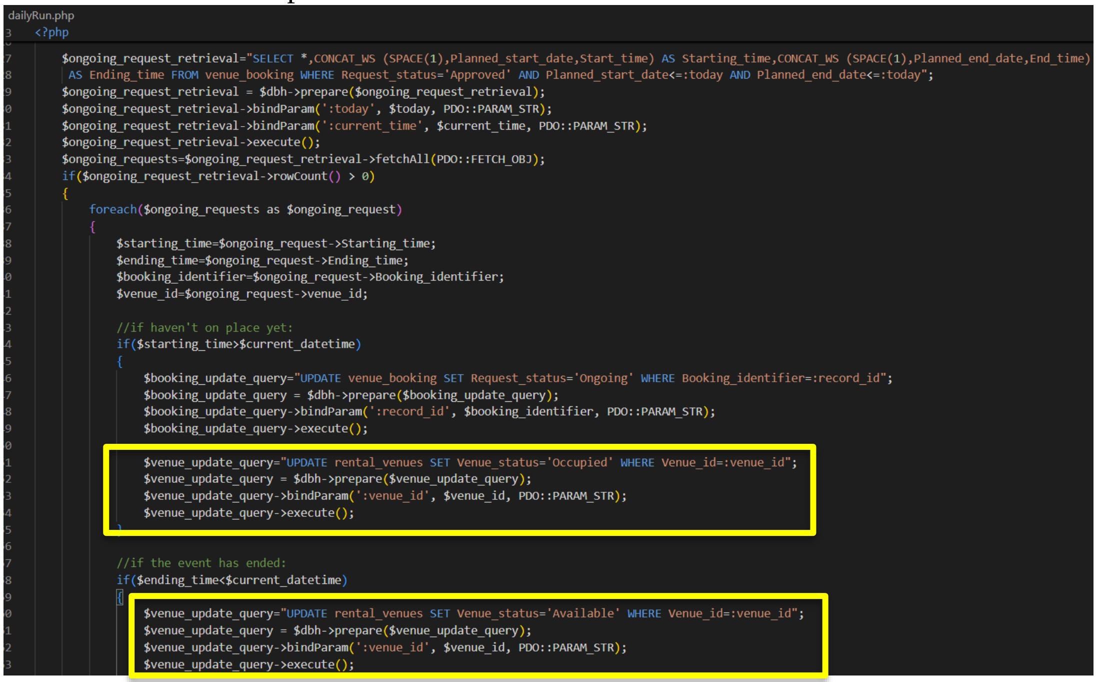
Another scenario will be the venue/classroom usage request is on the current date and it has been approved by both parties to be ready for use. With these conditions, the DGS staff needs to constantly monitor the event status then to update the venue/classroom availability status, which consumes additional effort to do so. Since the conditions are listed, then a database checking could be performed.

Based on the records retrieved, if the usage timing has not hit the current time yet, then the venue/classroom's availability status will be updated to available. Whereas if the timing is exactly or exceeds the current time, then the availability status will be updated to occupied. Refer to the yellow parts drawn in Figure 6.48 to learn about the technical implementation for this case.

6.3.8.3 Updating venue booking request's status

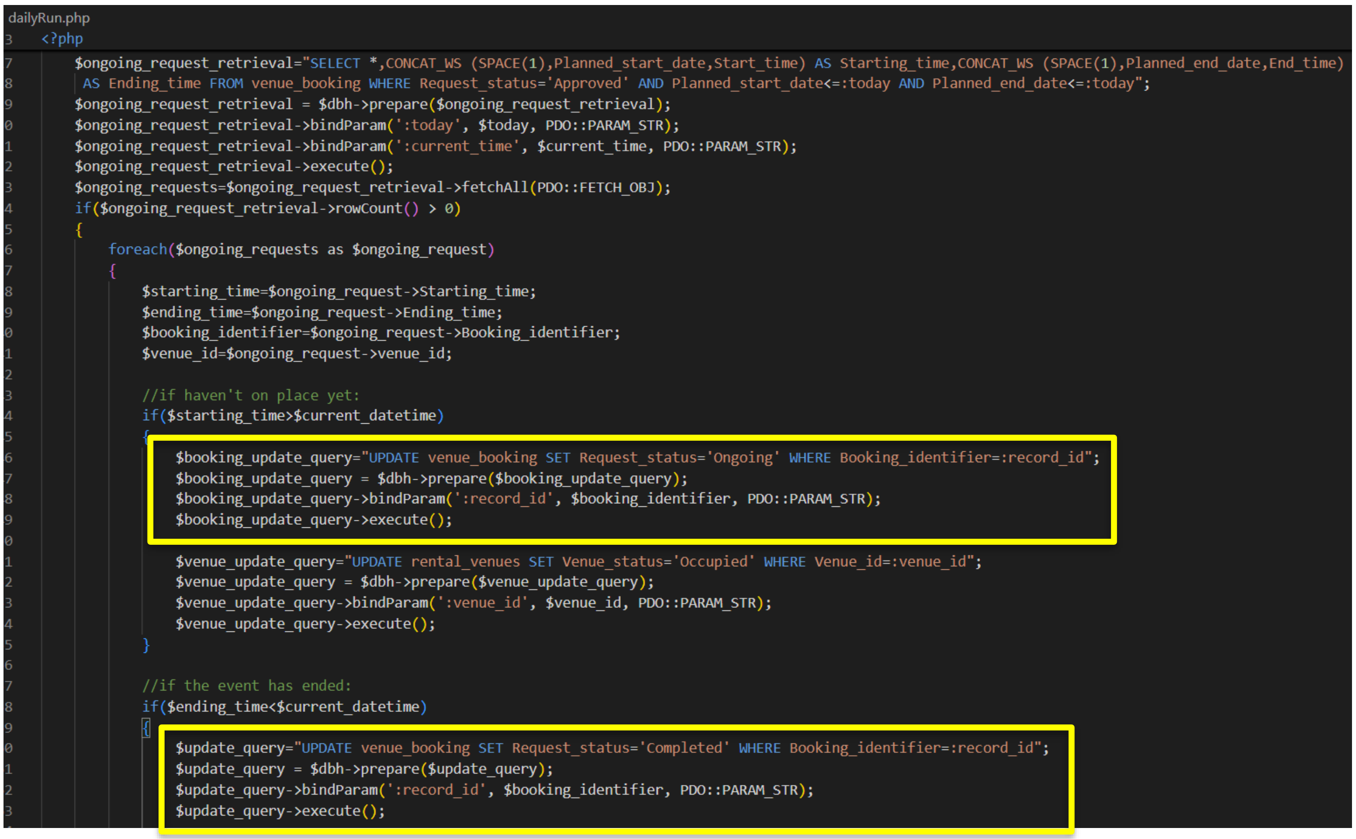
In the same scenario which was described in section 6.3.7.2, the DGS staff may need to update the request's status manually if the event has been completed. This could be done with the background running as well.

Based on the records retrieved, if the usage timing has not hit the current time yet, then the request's status will be updated to ongoing. Whereas if the timing is exactly or exceeds the current time, then the availability status will be updated to completed. Refer to the yellow parts drawn in Figure 6.49 to learn about the technical implementation for this case.



```
dailyRun.php
<?php

    $ongoing_request_retrieval="SELECT *,CONCAT_WS (SPACE(1),Planned_start_date,Start_time) AS Starting_time,CONCAT_WS (SPACE(1),Planned_end_date,End_time)
     AS Ending_time FROM venue_booking WHERE Request_status='Approved' AND Planned_start_date<=:today AND Planned_end_date<=:today";
    $ongoing_request_retrieval = $dbh->prepare($ongoing_request_retrieval);
    $ongoing_request_retrieval->bindParam(':today', $today, PDO::PARAM_STR);
    $ongoing_request_retrieval->bindParam(':current_time', $current_time, PDO::PARAM_STR);
    $ongoing_request_retrieval->execute();
    $ongoing_requests=$ongoing_request_retrieval->fetchAll(PDO::FETCH_OBJ);
    if($ongoing_request_retrieval->rowCount() > 0)
    {
        foreach($ongoing_requests as $ongoing_request)
        {
            $starting_time=$ongoing_request->Starting_time;
            $ending_time=$ongoing_request->Ending_time;
            $booking_identifier=$ongoing_request->Booking_identifier;
            $venue_id=$ongoing_request->venue_id;

            //if haven't on place yet:
            if($starting_time>$current_datetime)
            {
                $booking_update_query="UPDATE venue_booking SET Request_status='Ongoing' WHERE Booking_identifier=:record_id";
                $booking_update_query = $dbh->prepare($booking_update_query);
                $booking_update_query->bindParam(':record_id', $booking_identifier, PDO::PARAM_STR);
                $booking_update_query->execute();

                $venue_update_query="UPDATE rental_venues SET Venue_status='Occupied' WHERE Venue_id=:venue_id";
                $venue_update_query = $dbh->prepare($venue_update_query);
                $venue_update_query->bindParam(':venue_id', $venue_id, PDO::PARAM_STR);
                $venue_update_query->execute();
            }

            //if the event has ended:
            if($ending_time<$current_datetime)
            {
                $venue_update_query="UPDATE rental_venues SET Venue_status='Available' WHERE Venue_id=:venue_id";
                $venue_update_query = $dbh->prepare($venue_update_query);
                $venue_update_query->bindParam(':venue_id', $venue_id, PDO::PARAM_STR);
                $venue_update_query->execute();
```

Figure 6.60: To update the venue/classroom's availability status

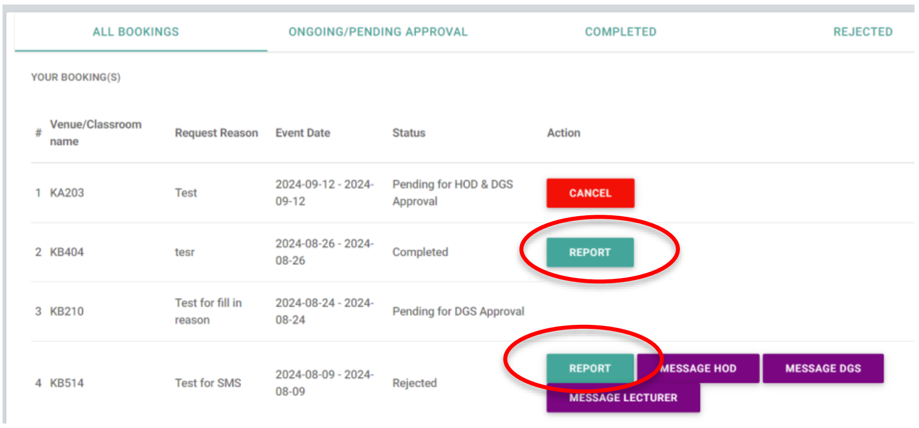


```
dailyRun.php
    <?php
        $ongoing_request_retrieval="SELECT *,CONCAT_WS (SPACE(1),Planned_start_date,Start_time) AS Starting_time,CONCAT_WS (SPACE(1),Planned_end_date,End_time)
         AS Ending_time FROM venue_booking WHERE Request_status='Approved' AND Planned_start_date<=:today AND Planned_end_date<=:today";
        $ongoing_request_retrieval = $dbh->prepare($ongoing_request_retrieval);
        $ongoing_request_retrieval->bindParam(':today', $today, PDO::PARAM_STR);
        $ongoing_request_retrieval->bindParam(':current_time', $current_time, PDO::PARAM_STR);
        $ongoing_request_retrieval->execute();
        $ongoing_requests=$ongoing_request_retrieval->fetchAll(PDO::FETCH_OBJ);
        if($ongoing_request_retrieval->rowCount() > 0)
        {
            foreach($ongoing_requests as $ongoing_request)
            {
                $starting_time=$ongoing_request->Starting_time;
                $ending_time=$ongoing_request->Ending_time;
                $booking_identifier=$ongoing_request->Booking_identifier;
                $venue_id=$ongoing_request->venue_id;

                //if haven't on place yet:
                if($starting_time>$current_datetime)
                {
                    $booking_update_query="UPDATE venue_booking SET Request_status='Ongoing' WHERE Booking_identifier=:record_id";
                    $booking_update_query = $dbh->prepare($booking_update_query);
                    $booking_update_query->bindParam(':record_id', $booking_identifier, PDO::PARAM_STR);
                    $booking_update_query->execute();

                    $venue_update_query="UPDATE rental_venues SET Venue_status='Occupied' WHERE Venue_id=:venue_id";
                    $venue_update_query = $dbh->prepare($venue_update_query);
                    $venue_update_query->bindParam(':venue_id', $venue_id, PDO::PARAM_STR);
                    $venue_update_query->execute();
                }

                //if the event has ended:
                if($ending_time<$current_datetime)
                {
                    $update_query="UPDATE venue_booking SET Request_status='Completed' WHERE Booking_identifier=:record_id";
                    $update_query = $dbh->prepare($update_query);
                    $update_query->bindParam(':record_id', $booking_identifier, PDO::PARAM_STR);
                    $update_query->execute();
```

Figure 6.61: To update the venue booking request status

6.4 PDF report generation function

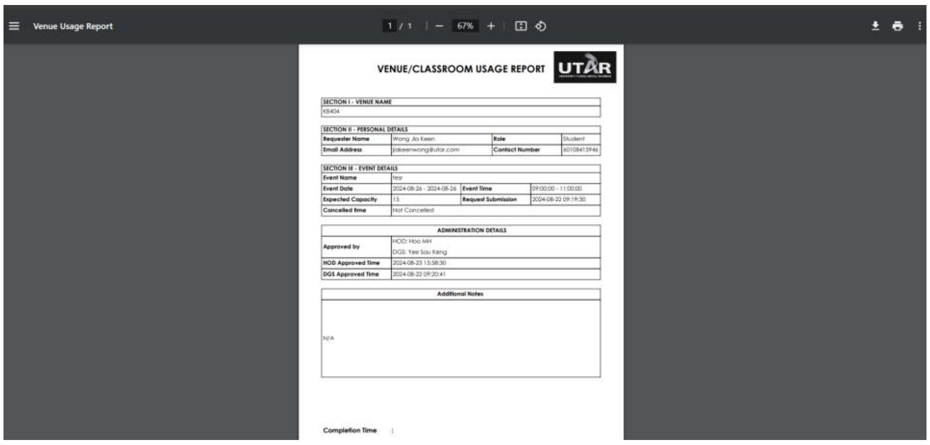
Relating to section 6.3.5 about venue/classroom booking management, this function aims to generate a report based on the booking information. Utilizing the FPDF library, only completed (inclusive of rejected and cancelled) booking requests information could be reviewed in PDF format. This could be done by validating the status of the booking request when accessing.



YOUR BOOKING(S)

#	Venue/Classroom name	Request Reason	Event Date	Status	Action
1	KA203	Test	2024-09-12 - 2024-09-12	Pending for HOD & DGS Approval	CANCEL
2	KB404	tesr	2024-08-26 - 2024-08-26	Completed	REPORT
3	KB210	Test for fill in reason	2024-08-24 - 2024-08-24	Pending for DGS Approval	
4	KB514	Test for SMS	2024-08-09 - 2024-08-09	Rejected	REPORT MESSAGE HOD MESSAGE DGS MESSAGE LECTURER

Figure 6.62: Booking requests that could only be view with report are shown with the green 'Report' button, being circled with red


Venue Usage Report

VENUE/CLASSROOM USAGE REPORT

SECTION I - VENUE NAME
KB404

SECTION II - PERSONAL DETAILS			
Requester Name	Wong Jia Keen	Role	Student
Email Address	jiakeenwong@utar.com	Contact Number	60108415946

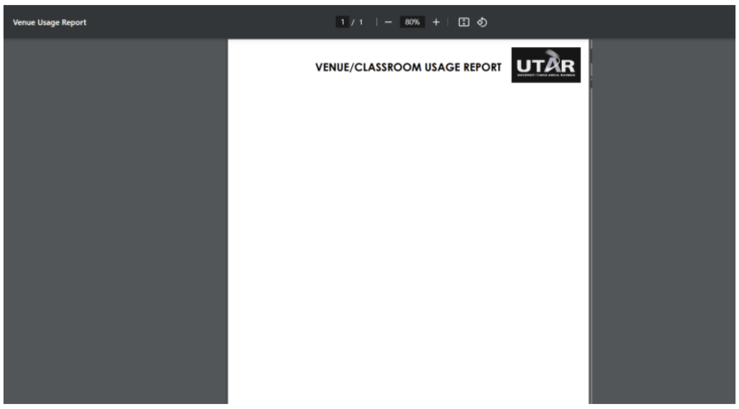
SECTION III - EVENT DETAILS			
Event Name	test		
Event Date	2024-08-26 - 2024-08-26	Event Time	09:00:00 - 11:00:00
Expected Capacity	15	Request Submission	2024-08-22 09:19:30
Cancelled time	Not Cancelled		

ADMINISTRATION DETAILS	
Approved by	HOD: Hoo MH DGS: Yee Sou Keng
HOD Approved Time	2024-08-23 15:58:30
DGS Approved Time	2024-08-22 09:20:41

Additional Notes
N/A

Completion Time :

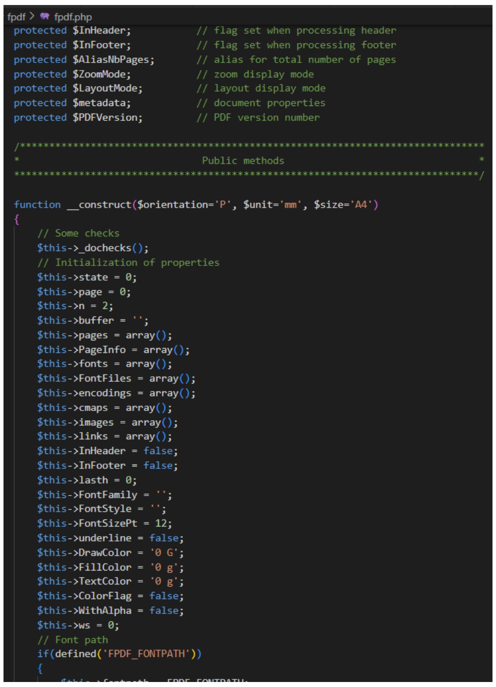
Figure 6.63: Valid booking request shown in PDF


Venue Usage Report

VENUE/CLASSROOM USAGE REPORT

Figure 6.64: Invalid booking request shown in PDF

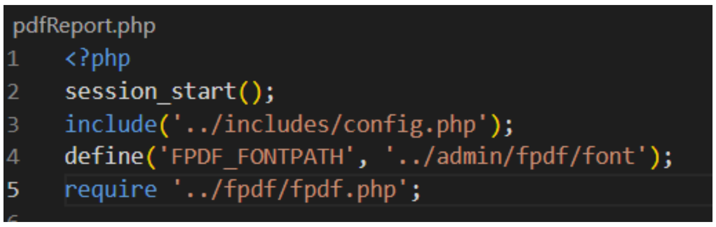
In terms of technical implementation, the library configuration script must be placed in the project folder, existing in the form of PHP file which could be found in the official website repository. The script contains classes that declares public variables and functions that could be extended in the target file.


fpdf > fpdf.php

```php
protected $InHeader;          // flag set when processing header
protected $InFooter;          // flag set when processing footer
protected $AliasNbPages;      // alias for total number of pages
protected $ZoomMode;          // zoom display mode
protected $LayoutMode;        // layout display mode
protected $metadata;          // document properties
protected $PDFVersion;        // PDF version number

/*******************************************************************************
*                               Public methods                                 *
*******************************************************************************/

function __construct($orientation='P', $unit='mm', $size='A4')
{
    // Some checks
    $this->_dochecks();
    // Initialization of properties
    $this->state = 0;
    $this->page = 0;
    $this->n = 2;
    $this->buffer = '';
    $this->pages = array();
    $this->PageInfo = array();
    $this->fonts = array();
    $this->FontFiles = array();
    $this->encodings = array();
    $this->cmaps = array();
    $this->images = array();
    $this->links = array();
    $this->InHeader = false;
    $this->InFooter = false;
    $this->lasth = 0;
    $this->FontFamily = '';
    $this->FontStyle = '';
    $this->FontSizePt = 12;
    $this->underline = false;
    $this->DrawColor = '0 G';
    $this->FillColor = '0 g';
    $this->TextColor = '0 g';
    $this->ColorFlag = false;
    $this->WithAlpha = false;
    $this->ws = 0;
    // Font path
    if(defined('FPDF_FONTPATH'))
    {
```

Figure 6.65: FPDF configuration script file

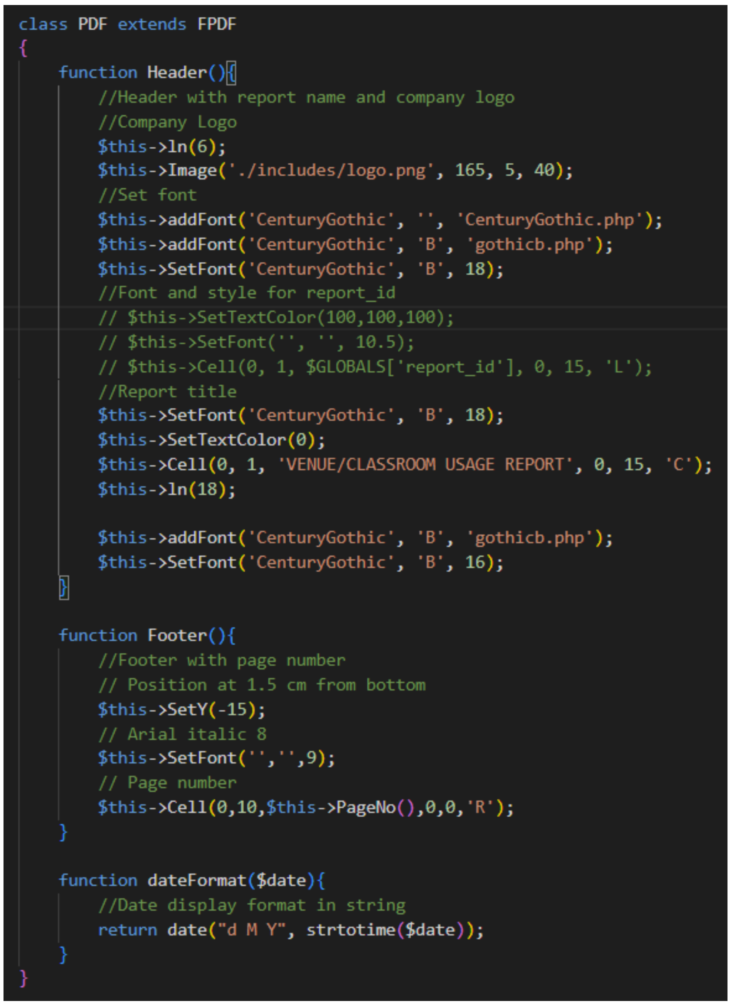
Next, the project author needs to declare a PHP page to configure it to display information, by including the downloaded source configuration file at the PHP page.



```
pdfReport.php
<?php
session_start();
include('../includes/config.php');
define('FPDF_FONTPATH', '../admin/fpdf/font');
require '../fpdf/fpdf.php';
```

Figure 6.66: Inclusion of FPDF configuration script file

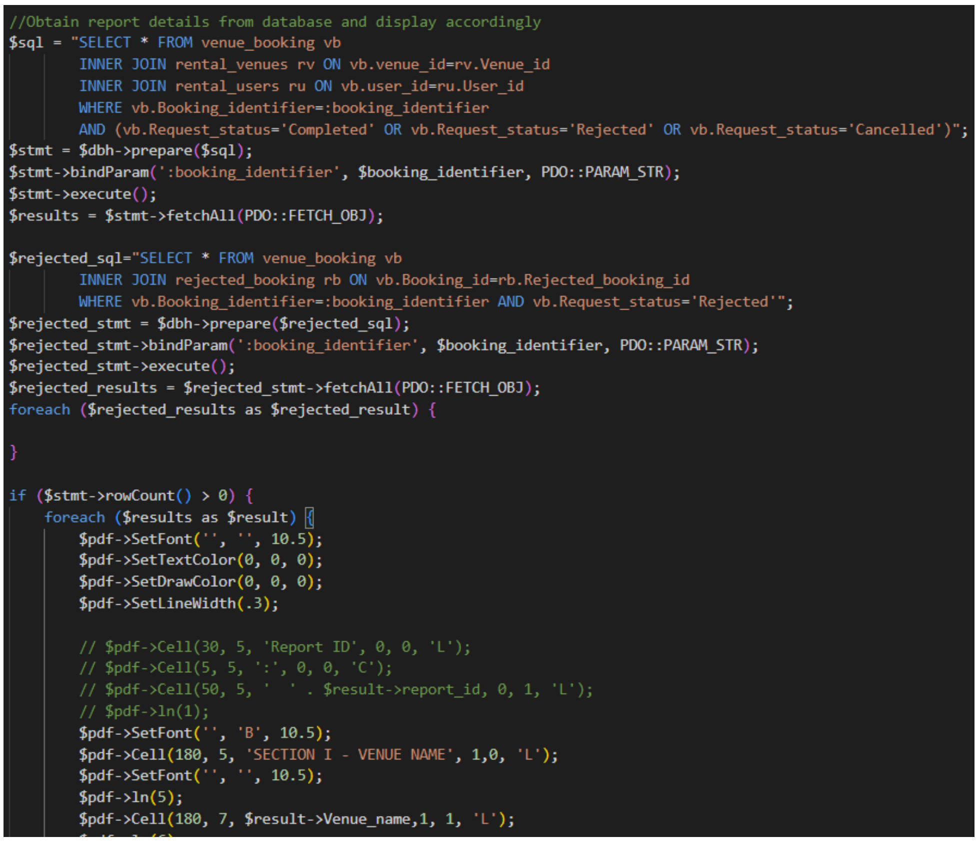
In the new page built, the project author has to extend the functions from the configuration file to customize the properties to be shown. For example, the font styles and the header and footer contents.



```
class PDF extends FPDF
{
    function Header(){
        //Header with report name and company logo
        //Company Logo
        $this->ln(6);
        $this->Image('./includes/logo.png', 165, 5, 40);
        //Set font
        $this->addFont('CenturyGothic', '', 'CenturyGothic.php');
        $this->addFont('CenturyGothic', 'B', 'gothicb.php');
        $this->SetFont('CenturyGothic', 'B', 18);
        //Font and style for report_id
        // $this->SetTextColor(100,100,100);
        // $this->SetFont('', '', 10.5);
        // $this->Cell(0, 1, $GLOBALS['report_id'], 0, 15, 'L');
        //Report title
        $this->SetFont('CenturyGothic', 'B', 18);
        $this->SetTextColor(0);
        $this->Cell(0, 1, 'VENUE/CLASSROOM USAGE REPORT', 0, 15, 'C');
        $this->ln(18);

        $this->addFont('CenturyGothic', 'B', 'gothicb.php');
        $this->SetFont('CenturyGothic', 'B', 16);
    }

    function Footer(){
        //Footer with page number
        // Position at 1.5 cm from bottom
        $this->SetY(-15);
        // Arial italic 8
        $this->SetFont('','',9);
        // Page number
        $this->Cell(0,10,$this->PageNo(),0,0,'R');
    }

    function dateFormat($date){
        //Date display format in string
        return date("d M Y", strtotime($date));
    }
}
```

Figure 6.67: Extension of functions in own file

To set the information displayed, a data fetching needs to be performed on this page. From the queries returned, the project author could build a table to achieve a clear segregation of data displayed. The tables are built by defining the cell size, the next column, the font styles and of course the contents to be shown.



```php
//Obtain report details from database and display accordingly
$sql = "SELECT * FROM venue_booking vb
        INNER JOIN rental_venues rv ON vb.venue_id=rv.Venue_id
        INNER JOIN rental_users ru ON vb.user_id=ru.User_id
        WHERE vb.Booking_identifier=:booking_identifier
        AND (vb.Request_status='Completed' OR vb.Request_status='Rejected' OR vb.Request_status='Cancelled')";
$stmt = $dbh->prepare($sql);
$stmt->bindParam(':booking_identifier', $booking_identifier, PDO::PARAM_STR);
$stmt->execute();
$results = $stmt->fetchAll(PDO::FETCH_OBJ);

$rejected_sql="SELECT * FROM venue_booking vb
        INNER JOIN rejected_booking rb ON vb.Booking_id=rb.Rejected_booking_id
        WHERE vb.Booking_identifier=:booking_identifier AND vb.Request_status='Rejected'";
$rejected_stmt = $dbh->prepare($rejected_sql);
$rejected_stmt->bindParam(':booking_identifier', $booking_identifier, PDO::PARAM_STR);
$rejected_stmt->execute();
$rejected_results = $rejected_stmt->fetchAll(PDO::FETCH_OBJ);
foreach ($rejected_results as $rejected_result) {

}

if ($stmt->rowCount() > 0) {
    foreach ($results as $result) {
        $pdf->SetFont('', '', 10.5);
        $pdf->SetTextColor(0, 0, 0);
        $pdf->SetDrawColor(0, 0, 0);
        $pdf->SetLineWidth(.3);

        // $pdf->Cell(30, 5, 'Report ID', 0, 0, 'L');
        // $pdf->Cell(5, 5, ':', 0, 0, 'C');
        // $pdf->Cell(50, 5, '  ' . $result->report_id, 0, 1, 'L');
        // $pdf->ln(1);
        $pdf->SetFont('', 'B', 10.5);
        $pdf->Cell(180, 5, 'SECTION I - VENUE NAME', 1,0, 'L');
        $pdf->SetFont('', '', 10.5);
        $pdf->ln(5);
        $pdf->Cell(180, 7, $result->Venue_name,1, 1, 'L');
```

Figure 6.68: Technical implementation of data fetching with the self-defining tables

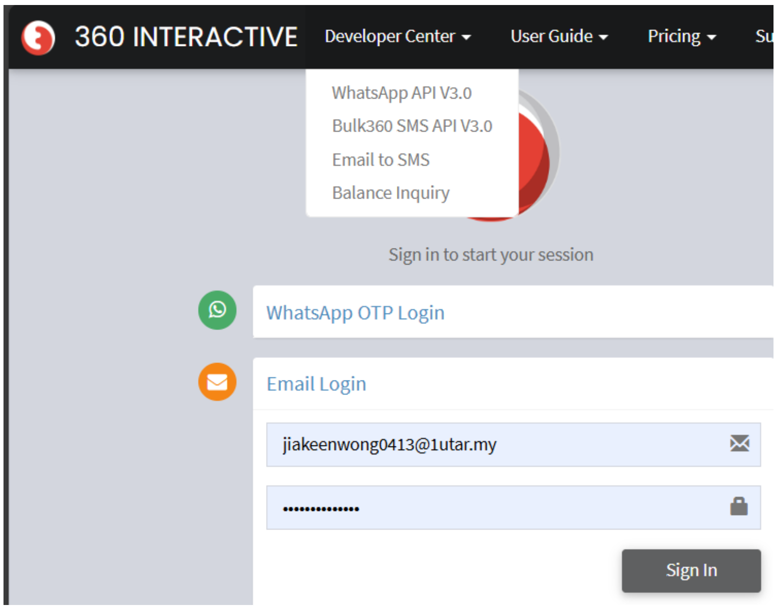
6.5 Notification

This section describes the notifying function for the UTAR Venue/Classroom Booking Web Portal. As there are requirements stating that they prefer an instant notifying function instead of just email, then SMS could be a choice to accomplish this objective. The notification works on various scenarios, including the venue/classroom booking request updates and the messaging updates.

6.5.1 SMS

This feature is being implemented using the API provided by Bulk360, a Malaysian company that provides professional web development and design outsourcing services. In detail, their products include the communication APIs (WhatsApp, SMS and Email to SMS), Search Engine Optimization (SEO) and

IT outsourcing. To start, the project author needs to register an account on the product portal first, as a developer.



Sign in to start your session

WhatsApp OTP Login

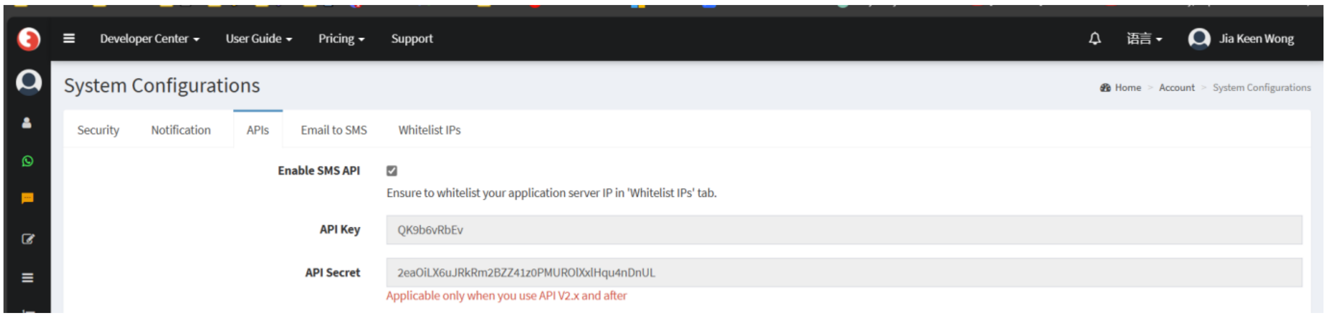
Email Login

jiakeenwong0413@1utar.my

••••••••••••••

Figure 6.69: A glance at the APIs and login screen of Bulk360 panel

After logging into the developer panel, the SMS API must be activated first. With the activation action, the project author could get to know the API key and secret hashcode to be pasted in the configuration file. Next, the project author needs to list the IP address for the system, in order to integrate the API provided with the system developed. Since the system is self-hosted currently, then 0.0.0.0 would be the universal IP address to use in this case.



System Configurations

Enable SMS API ✓

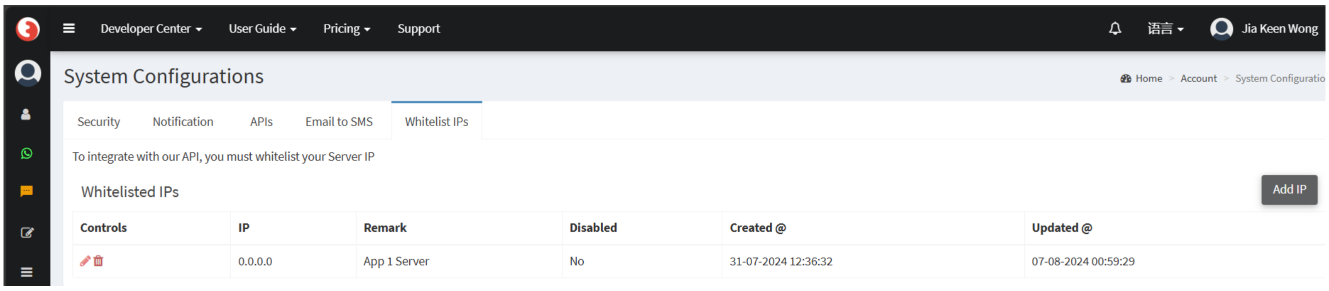
Ensure to whitelist your application server IP in 'Whitelist IPs' tab.

API Key: QK9b6vRbEv

API Secret: 2eaOiLX6uJRkRm2BZZ41z0PMUROlXxlHqu4nDnUL

Applicable only when you use API V2.x and after

Figure 6.70: Activation of SMS API in Bulk360 panel



System Configurations

Security | Notification | APIs | Email to SMS | Whitelist IPs

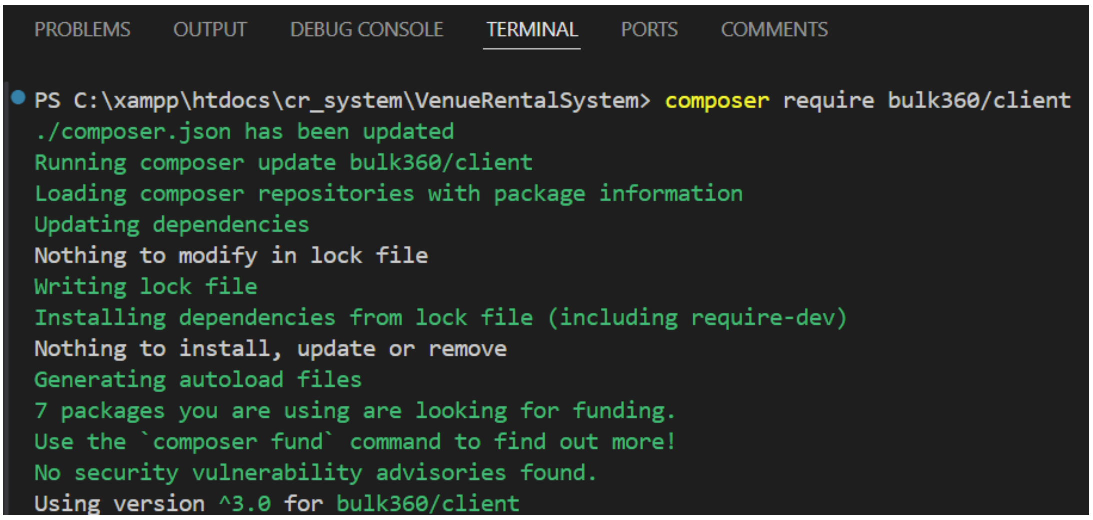
To integrate with our API, you must whitelist your Server IP

Whitelisted IPs

Controls	IP	Remark	Disabled	Created @	Updated @
	0.0.0.0	App 1 Server	No	31-07-2024 12:36:32	07-08-2024 00:59:29

Figure 6.71: IP address listing in Bulk360 panel

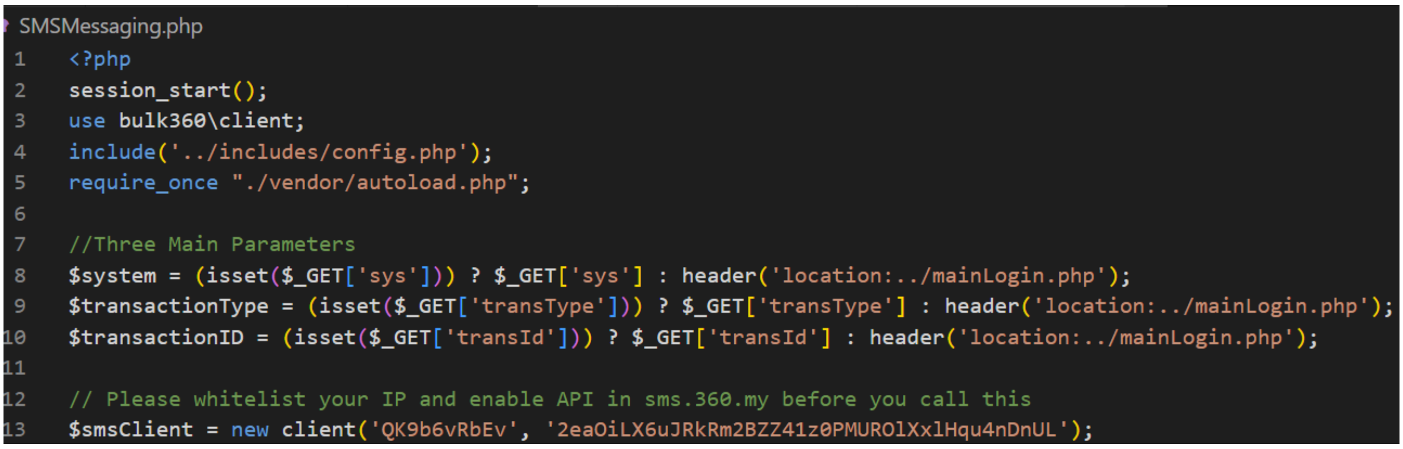
After starting the configurations at the developer panel, the project author needs to install the package files in the project folder. This could be done by using Composer commands.



```
PS C:\xampp\htdocs\cr_system\VenueRentalSystem> composer require bulk360/client
./composer.json has been updated
Running composer update bulk360/client
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
7 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.
Using version ^3.0 for bulk360/client
```

Figure 6.72: Installation guide for Bulk360 SMS API

After installation, a new PHP file would need to be declared to define the SMS scenarios and list the configurations. In this project, the file is named as SMSMessaging.php. Do notice the 13th line in Figure 6.73, there would be the API key and secret hashcode pasted from the developer panel, so that it could identify the developer identity.


SMSMessaging.php

```php
<?php
session_start();
use bulk360\client;
include('../includes/config.php');
require_once "./vendor/autoload.php";

//Three Main Parameters
$system = (isset($_GET['sys'])) ? $_GET['sys'] : header('location:../mainLogin.php');
$transactionType = (isset($_GET['transType'])) ? $_GET['transType'] : header('location:../mainLogin.php');
$transactionID = (isset($_GET['transId'])) ? $_GET['transId'] : header('location:../mainLogin.php');

// Please whitelist your IP and enable API in sms.360.my before you call this
$smsClient = new client('QK9b6vRbEv', '2eaOiLX6uJRkRm2BZZ41z0PMUROlXxlHqu4nDnUL');
```

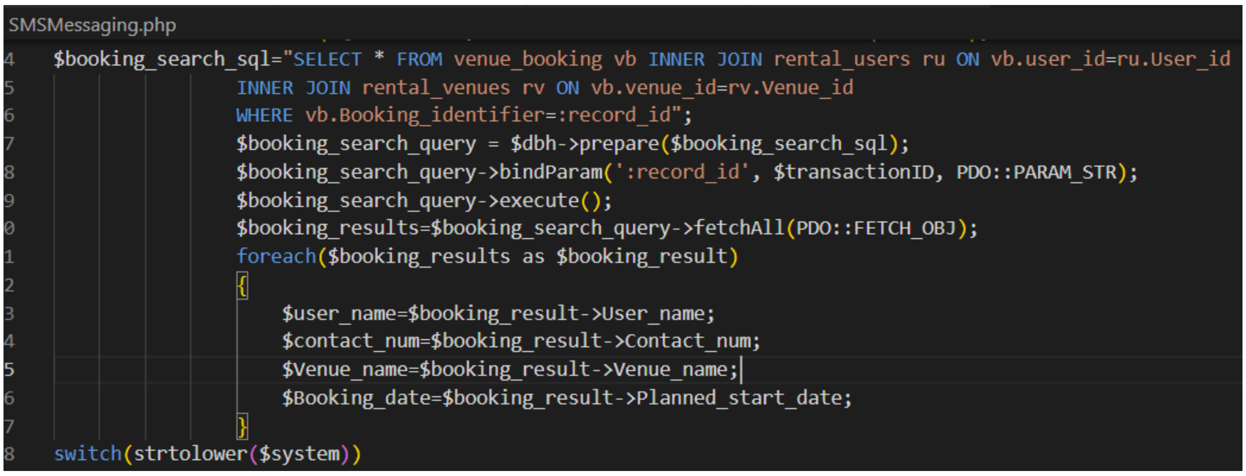
Figure 6.73: Declaration of SMS configurations in SMSMessaging.php

To explain the variables, $system is used to define which type of system is it, for this case it will be “venuerentalsystem”, while $transactionType would be the scenario name, whether is it relating to booking request updates or other information updates. The $transactionID would represent the piece of information affected, so that data fetching could be performed with the database. With the data fetching, the exact row of information could be retrieved and completes the sending content.

There would be scenarios that trigger the SMS sending function, that will be when:

1. A venue/classroom booking request has been approved by DGS staff.
2. A venue/classroom booking request has been rejected by DGS staff or HOD.
3. A venue/classroom booking request’s information has been updated by DGS staff or HOD.

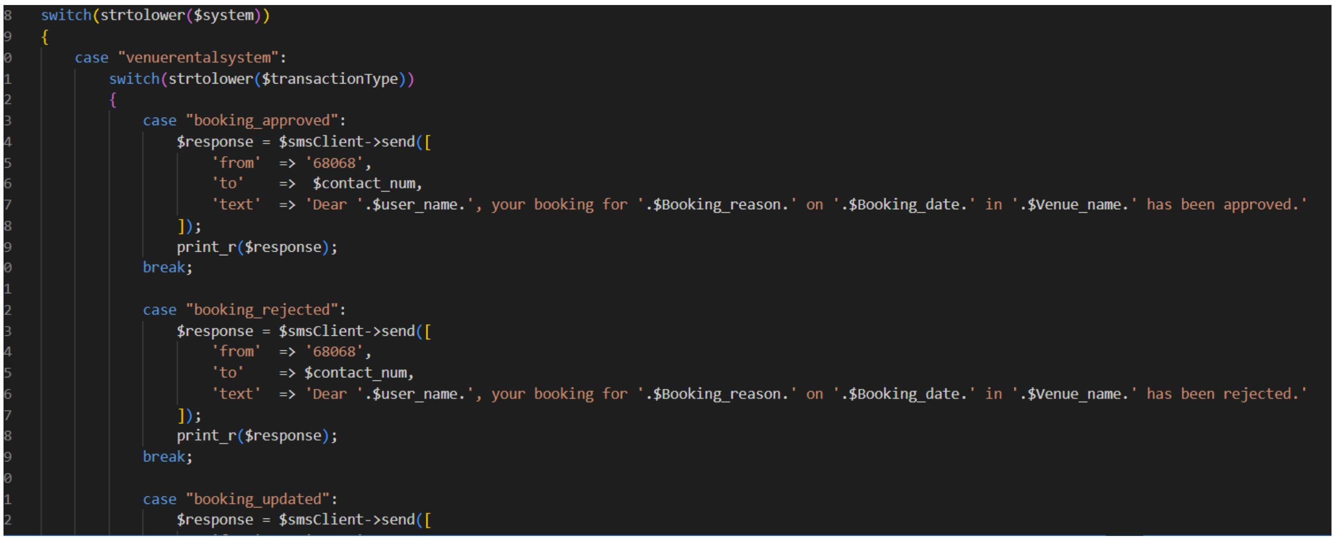
Declaration of scenario types and data fetching needs to be done before sending SMS, so that the contents sent are compatible with the updates. Case switching is used in this case. In detail, the data fetched includes the user name, the booking requester’s contact number, the reserved venue/classroom’s name and the usage date. The last 2 pieces of information are vital since the booking requesters could get an understanding of which booking was made, without recalling.



SMSMessaging.php

```php
$booking_search_sql="SELECT * FROM venue_booking vb INNER JOIN rental_users ru ON vb.user_id=ru.User_id
                INNER JOIN rental_venues rv ON vb.venue_id=rv.Venue_id
                WHERE vb.Booking_identifier=:record_id";
                $booking_search_query = $dbh->prepare($booking_search_sql);
                $booking_search_query->bindParam(':record_id', $transactionID, PDO::PARAM_STR);
                $booking_search_query->execute();
                $booking_results=$booking_search_query->fetchAll(PDO::FETCH_OBJ);
                foreach($booking_results as $booking_result)
                {
                    $user_name=$booking_result->User_name;
                    $contact_num=$booking_result->Contact_num;
                    $Venue_name=$booking_result->Venue_name;
                    $Booking_date=$booking_result->Planned_start_date;
                }
switch(strtolower($system))
```

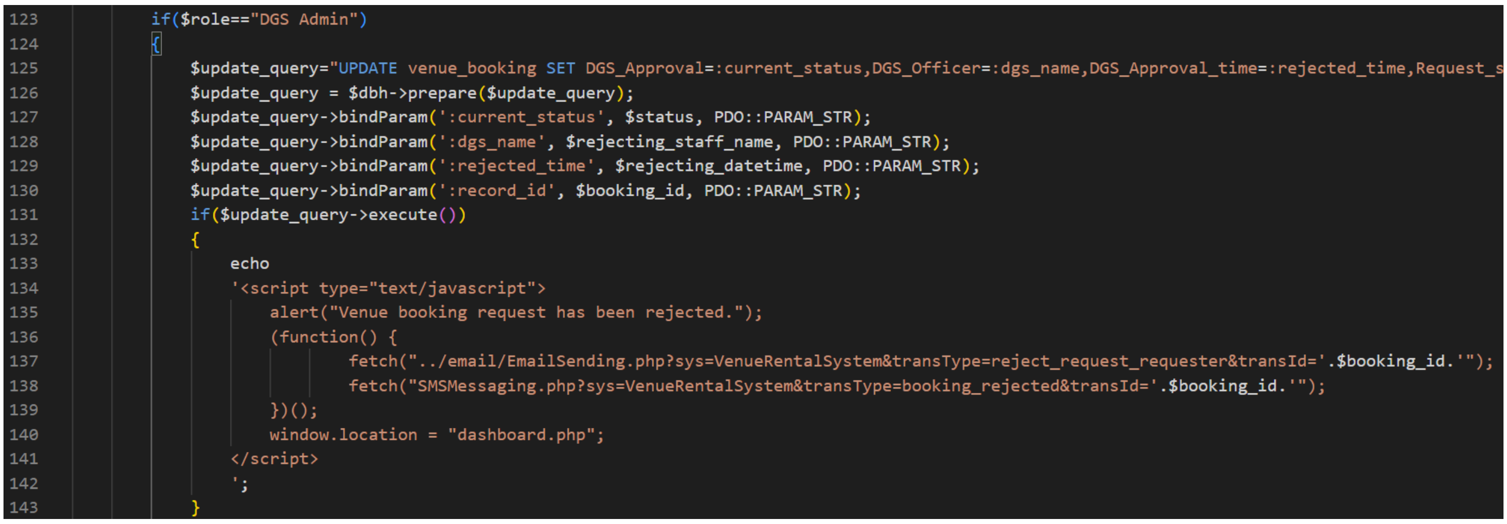
Figure 6.74: Data fetching in SMSMessaging.php



```php
switch(strtolower($system))
{
    case "venuerentalsystem":
        switch(strtolower($transactionType))
        {
            case "booking_approved":
                $response = $smsClient->send([
                    'from'  => '68068',
                    'to'    =>  $contact_num,
                    'text'  => 'Dear '.$user_name.', your booking for '.$Booking_reason.' on '.$Booking_date.' in '.$Venue_name.' has been approved.'
                ]);
                print_r($response);
            break;

            case "booking_rejected":
                $response = $smsClient->send([
                    'from'  => '68068',
                    'to'    => $contact_num,
                    'text'  => 'Dear '.$user_name.', your booking for '.$Booking_reason.' on '.$Booking_date.' in '.$Venue_name.' has been rejected.'
                ]);
                print_r($response);
            break;

            case "booking_updated":
                $response = $smsClient->send([
```

Figure 6.75: Scenario types declared

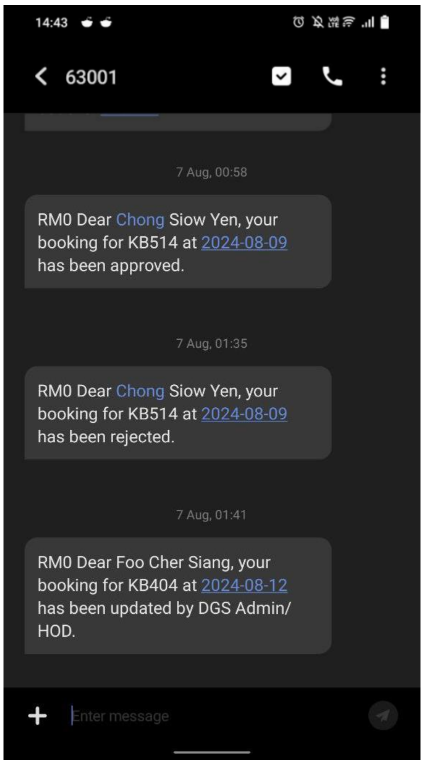
In those functional pages, the SMS function gets triggered if any $_POST events or liberally, data insertion or updating happens. This could be declared by calling the JavaScript function in PHP tags, where *fetch()* contains the parameters that call the link to the SMS sending PHP page, along with the variables that notify the system about the system type, transaction type and which scenario it belongs to. The 138th line in the following figure demonstrates how the calling took place if any updates happened.



```
if($role=="DGS Admin")
{
    $update_query="UPDATE venue_booking SET DGS_Approval=:current_status,DGS_Officer=:dgs_name,DGS_Approval_time=:rejected_time,Request_s
    $update_query = $dbh->prepare($update_query);
    $update_query->bindParam(':current_status', $status, PDO::PARAM_STR);
    $update_query->bindParam(':dgs_name', $rejecting_staff_name, PDO::PARAM_STR);
    $update_query->bindParam(':rejected_time', $rejecting_datetime, PDO::PARAM_STR);
    $update_query->bindParam(':record_id', $booking_id, PDO::PARAM_STR);
    if($update_query->execute())
    {
        echo
        '<script type="text/javascript">
            alert("Venue booking request has been rejected.");
            (function() {
                fetch("../email/EmailSending.php?sys=VenueRentalSystem&transType=reject_request_requester&transId='.$booking_id.'");
                fetch("SMSMessaging.php?sys=VenueRentalSystem&transType=booking_rejected&transId='.$booking_id.'");
            })();
            window.location = "dashboard.php";
        </script>
        ';
    }
```

Figure 6.76: Example of calling SMS function when information gets updated

The SMS sending function is working in a freemium version, where every developer gets a free trial quota that is worth RM5. Each SMS sent consumes around RM0.10, regardless of the length of messages sent. After that, the developer will need to be charged to use the service. Therefore, it is urged that the implementation party to consider this service as it works with the Telco to send SMS. The following figure demonstrates the receiving of SMS messages:



7 Aug, 00:58

RM0 Dear Chong Siow Yen, your booking for KB514 at 2024-08-09 has been approved.

7 Aug, 01:35

RM0 Dear Chong Siow Yen, your booking for KB514 at 2024-08-09 has been rejected.

7 Aug, 01:41

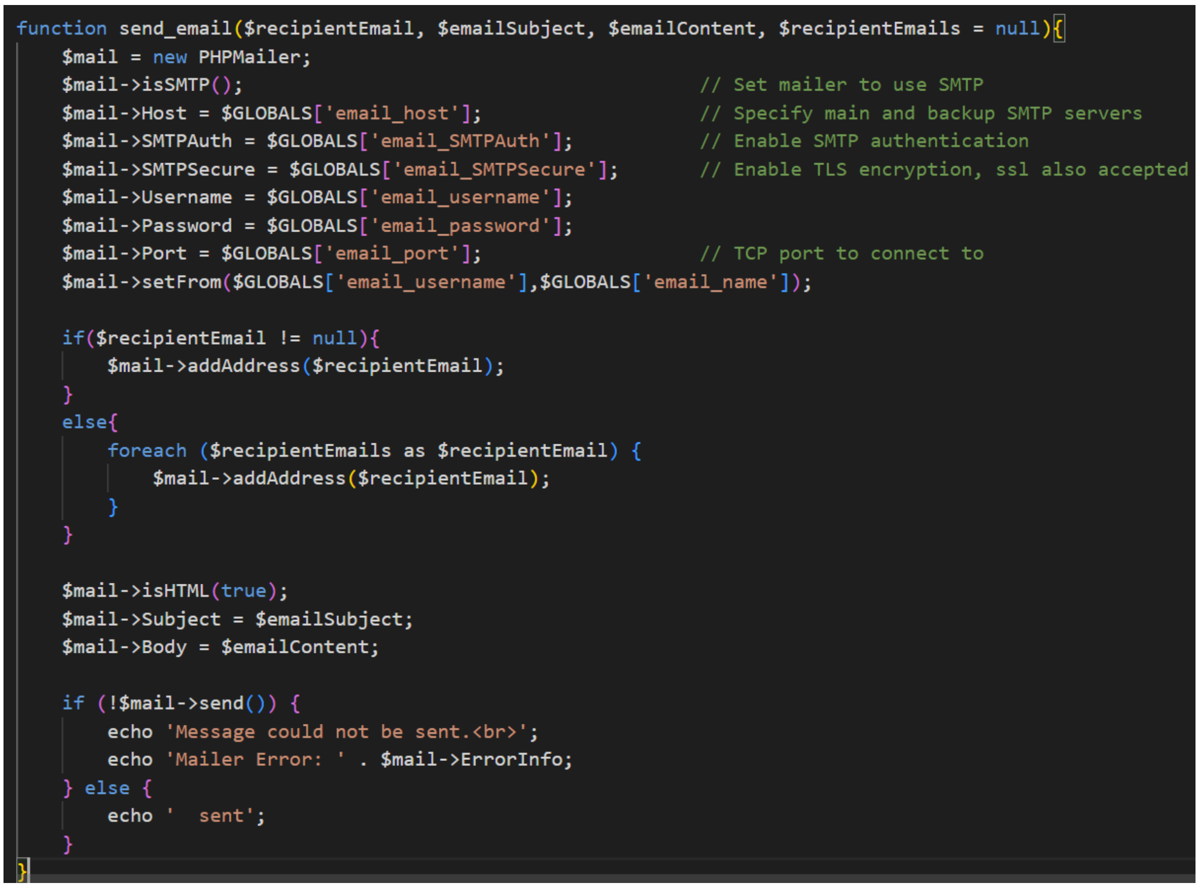
RM0 Dear Foo Cher Siang, your booking for KB404 at 2024-08-12 has been updated by DGS Admin/ HOD.

Figure 6.77: Receiving of SMS messages

6.5.2 Email

Serves as another notifying function, the emailing in this project is achieved through using PHPMailer, an external email library that is intended to replace the mail() function in PHP (Bointon, 2019). Firstly, the project author needs to run the composer function to install the dependencies in the project folder first. Though this is an open-source project, there is no need to register a developer account and to verify the usage key like the Bulk360 SMS.

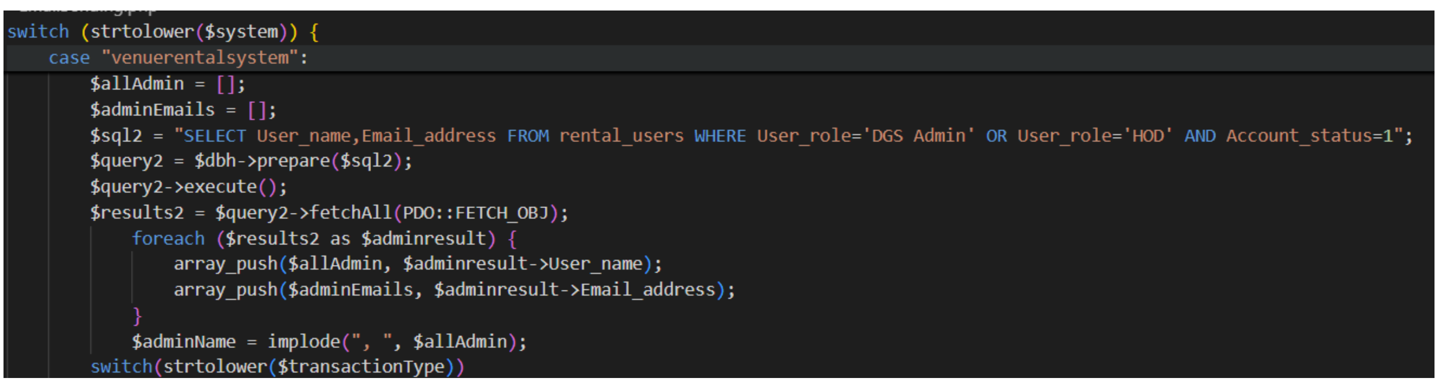
Email declaration page is needed to be built in the project folder. The variables declared are the same as presented in figure 6.61. While working for the SMS function does not need to write the sending function, it is necessary to self-define a sending function in this case.



```php
function send_email($recipientEmail, $emailSubject, $emailContent, $recipientEmails = null){
    $mail = new PHPMailer;
    $mail->isSMTP();                                          // Set mailer to use SMTP
    $mail->Host = $GLOBALS['email_host'];                     // Specify main and backup SMTP servers
    $mail->SMTPAuth = $GLOBALS['email_SMTPAuth'];             // Enable SMTP authentication
    $mail->SMTPSecure = $GLOBALS['email_SMTPSecure'];         // Enable TLS encryption, ssl also accepted
    $mail->Username = $GLOBALS['email_username'];
    $mail->Password = $GLOBALS['email_password'];
    $mail->Port = $GLOBALS['email_port'];                     // TCP port to connect to
    $mail->setFrom($GLOBALS['email_username'],$GLOBALS['email_name']);

    if($recipientEmail != null){
        $mail->addAddress($recipientEmail);
    }
    else{
        foreach ($recipientEmails as $recipientEmail) {
            $mail->addAddress($recipientEmail);
        }
    }

    $mail->isHTML(true);
    $mail->Subject = $emailSubject;
    $mail->Body = $emailContent;

    if (!$mail->send()) {
        echo 'Message could not be sent.<br>';
        echo 'Mailer Error: ' . $mail->ErrorInfo;
    } else {
        echo '  sent';
    }
}
```

Figure 6.78: Email-sending function

To explain, a new instance needs to be established everytime when attempting to send an email. Next, the email configurations are retrieved from the global session variables, including the port name, the accessing credentials fetched from the database. This function supports multiple receiver as well, as long as the recipients' email address are not empty. The email subject and content's variables are declared here as well, but not exactly defining what would be sent. The last few rows representing the exception error messages, if occurred.



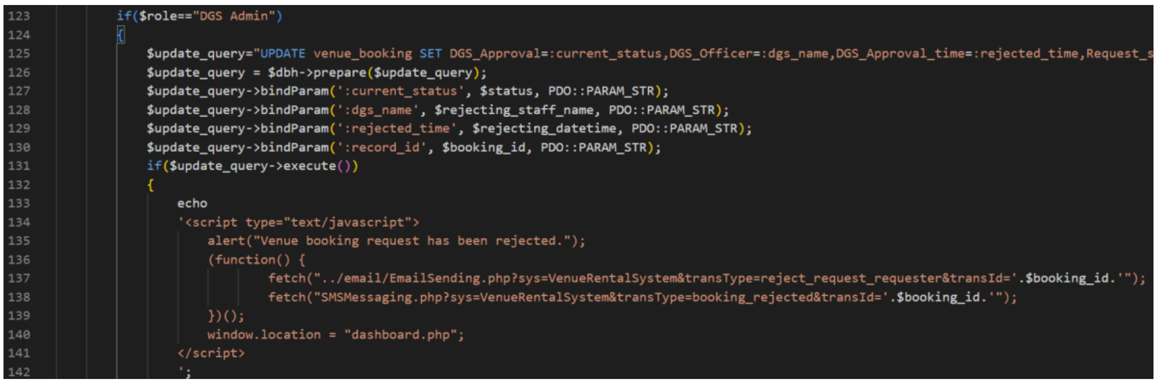
```
switch (strtolower($system)) {
    case "venuerentalsystem":
        $allAdmin = [];
        $adminEmails = [];
        $sql2 = "SELECT User_name,Email_address FROM rental_users WHERE User_role='DGS Admin' OR User_role='HOD' AND Account_status=1";
        $query2 = $dbh->prepare($sql2);
        $query2->execute();
        $results2 = $query2->fetchAll(PDO::FETCH_OBJ);
            foreach ($results2 as $adminresult) {
                array_push($allAdmin, $adminresult->User_name);
                array_push($adminEmails, $adminresult->Email_address);
            }
            $adminName = implode(", ", $allAdmin);
        switch(strtolower($transactionType))
```

Figure 6.79: Recipients' email declaring

Figure 6.79 retrieves all recipients' email addresses from the database, if they are active DGS staff or HODs. Since there may be multiple email addresses to send, they are stored in an array, including the names as well. The scenarios that trigger the emailing function are as follows:

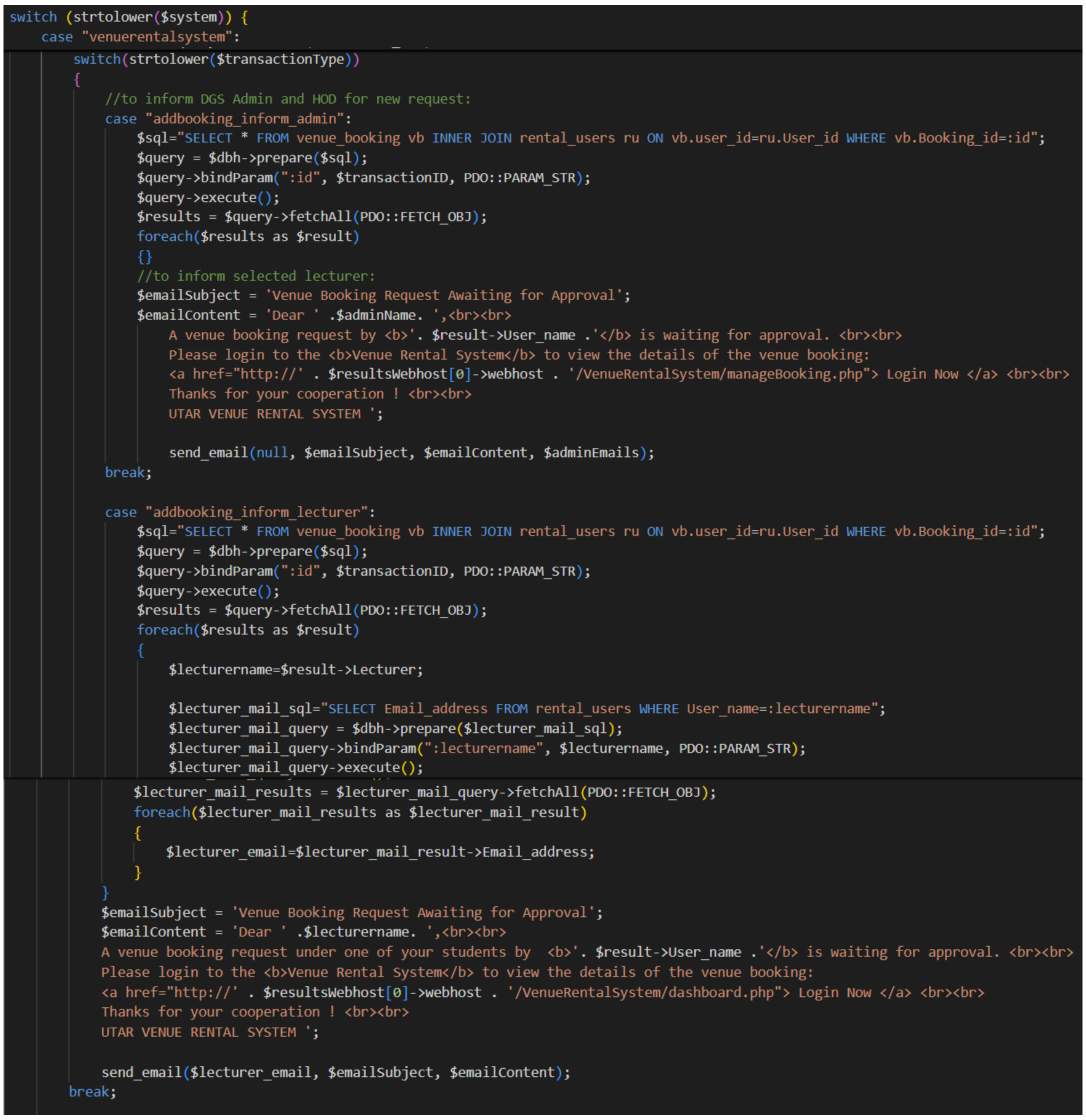
1. When someone submits a venue/classroom usage request, it reminds the DGS staff and HOD.
2. When a student submits a venue/classroom usage request, it reminds the lecturer chosen.
3. When the booking request gets approved, then the booking requester is notified.
4. When the booking request gets rejected, then the booking requester is notified.
5. When the booking request is already approved and added with notes by the DGS staff or HOD, then the booking requester is notified.
6. When the booking request gets cancelled by the booking requester, then the DGS staff and HOD gets notified.
7. When someone sent a message in the system, then the recipient gets notified.

In those functional pages, the email function gets triggered if any $_POST events or liberally, data insertion or updating happens. This could be declared with calling the JavaScript function in PHP tags, where *fetch()* contains the parameters that call the link to the email sending PHP page, along with the variables that notify the system about the system type, transaction type and which scenario it belongs to. The 137th line in the following figure demonstrates how does the calling took place if any updates.



```
123         if($role=="DGS Admin")
124         {
125             $update_query="UPDATE venue_booking SET DGS_Approval=:current_status,DGS_Officer=:dgs_name,DGS_Approval_time=:rejected_time,Request_s
126             $update_query = $dbh->prepare($update_query);
127             $update_query->bindParam(':current_status', $status, PDO::PARAM_STR);
128             $update_query->bindParam(':dgs_name', $rejecting_staff_name, PDO::PARAM_STR);
129             $update_query->bindParam(':rejected_time', $rejecting_datetime, PDO::PARAM_STR);
130             $update_query->bindParam(':record_id', $booking_id, PDO::PARAM_STR);
131             if($update_query->execute())
132             {
133                 echo
134                 '<script type="text/javascript">
135                     alert("Venue booking request has been rejected.");
136                     (function() {
137                             fetch("../email/EmailSending.php?sys=VenueRentalSystem&transType=reject_request_requester&transId='.$booking_id.'");
138                             fetch("SMSMessaging.php?sys=VenueRentalSystem&transType=booking_rejected&transId='.$booking_id.'");
139                     })();
140                     window.location = "dashboard.php";
141                 </script>
142                 ';
```

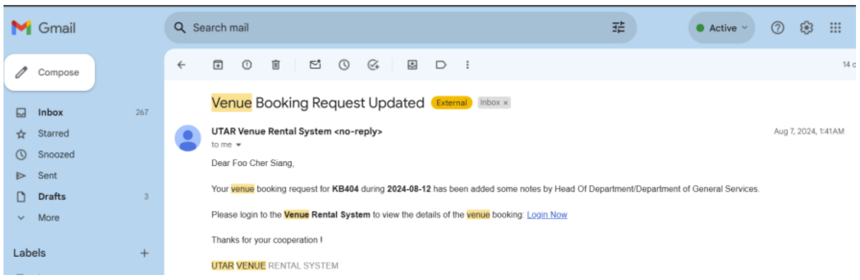
Figure 6.80: Example of calling email function



```
switch (strtolower($system)) {
    case "venuerentalsystem":
        switch(strtolower($transactionType))
        {
            //to inform DGS Admin and HOD for new request:
            case "addbooking_inform_admin":
                $sql="SELECT * FROM venue_booking vb INNER JOIN rental_users ru ON vb.user_id=ru.User_id WHERE vb.Booking_id=:id";
                $query = $dbh->prepare($sql);
                $query->bindParam(":id", $transactionID, PDO::PARAM_STR);
                $query->execute();
                $results = $query->fetchAll(PDO::FETCH_OBJ);
                foreach($results as $result)
                {}
                //to inform selected lecturer:
                $emailSubject = 'Venue Booking Request Awaiting for Approval';
                $emailContent = 'Dear ' .$adminName. ',<br><br>
                    A venue booking request by <b>'. $result->User_name .'</b> is waiting for approval. <br><br>
                    Please login to the <b>Venue Rental System</b> to view the details of the venue booking:
                    <a href="http://' . $resultsWebhost[0]->webhost . '/VenueRentalSystem/manageBooking.php"> Login Now </a> <br><br>
                    Thanks for your cooperation ! <br><br>
                    UTAR VENUE RENTAL SYSTEM ';

                    send_email(null, $emailSubject, $emailContent, $adminEmails);
            break;

            case "addbooking_inform_lecturer":
                $sql="SELECT * FROM venue_booking vb INNER JOIN rental_users ru ON vb.user_id=ru.User_id WHERE vb.Booking_id=:id";
                $query = $dbh->prepare($sql);
                $query->bindParam(":id", $transactionID, PDO::PARAM_STR);
                $query->execute();
                $results = $query->fetchAll(PDO::FETCH_OBJ);
                foreach($results as $result)
                {
                    $lecturername=$result->Lecturer;

                    $lecturer_mail_sql="SELECT Email_address FROM rental_users WHERE User_name=:lecturername";
                    $lecturer_mail_query = $dbh->prepare($lecturer_mail_sql);
                    $lecturer_mail_query->bindParam(":lecturername", $lecturername, PDO::PARAM_STR);
                    $lecturer_mail_query->execute();
                    $lecturer_mail_results = $lecturer_mail_query->fetchAll(PDO::FETCH_OBJ);
                    foreach($lecturer_mail_results as $lecturer_mail_result)
                    {
                        $lecturer_email=$lecturer_mail_result->Email_address;
                    }
                }
                $emailSubject = 'Venue Booking Request Awaiting for Approval';
                $emailContent = 'Dear ' .$lecturername. ',<br><br>
                A venue booking request under one of your students by  <b>'. $result->User_name .'</b> is waiting for approval. <br><br>
                Please login to the <b>Venue Rental System</b> to view the details of the venue booking:
                <a href="http://' . $resultsWebhost[0]->webhost . '/VenueRentalSystem/dashboard.php"> Login Now </a> <br><br>
                Thanks for your cooperation ! <br><br>
                UTAR VENUE RENTAL SYSTEM ';

                send_email($lecturer_email, $emailSubject, $emailContent);
            break;
```

Figure 6.81: Demonstration of scenario #1 and #2

Regardless of any notification methods used, the system takes time to load once the notification function is triggered, but not more than 10 seconds. The following figure shows the receiving of email at the recipient side.



Venue Booking Request Updated External Inbox ×

UTAR Venue Rental System <no-reply>
to me
Aug 7, 2024, 1:41AM

Dear Foo Cher Siang,

Your venue booking request for **KB404** during **2024-08-12** has been added some notes by Head Of Department/Department of General Services.

Please login to the **Venue Rental System** to view the details of the venue booking: Login Now

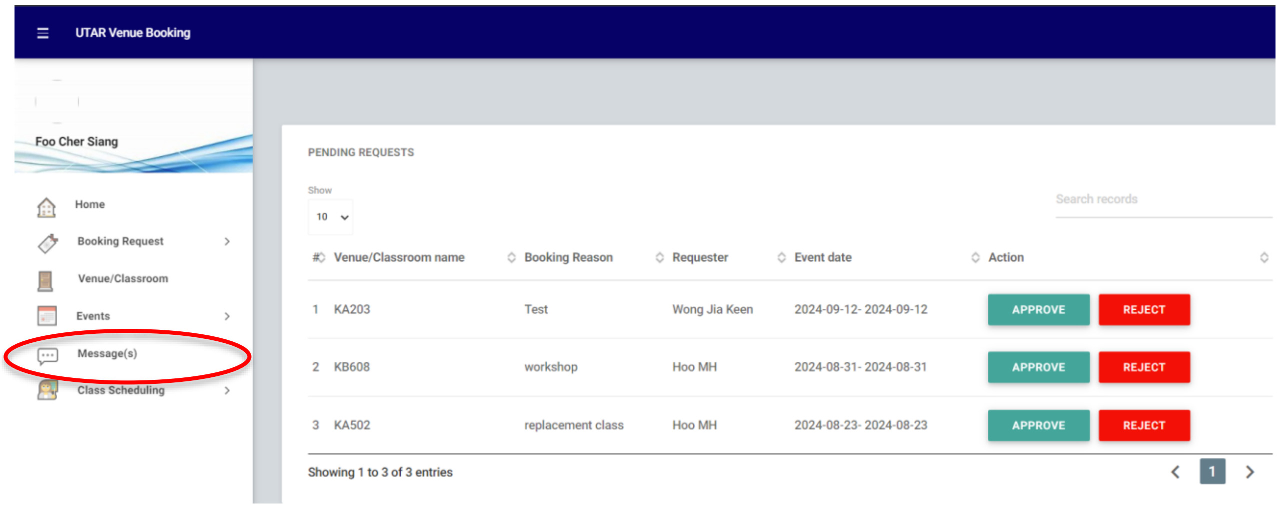
Thanks for your cooperation !

UTAR VENUE RENTAL SYSTEM

Figure 6.82: Email receiving

6.6 Instant Messaging

To convey information faster, instant messaging has been a communication trend. This function is featured in this system, accessible by all levels of end-users. This feature is useful especially when it comes to venue/classroom booking clarification purpose. To view the conversation histories, the end-users could trace to the Message(s) tab located in the sidebar.



PENDING REQUESTS

Show 10

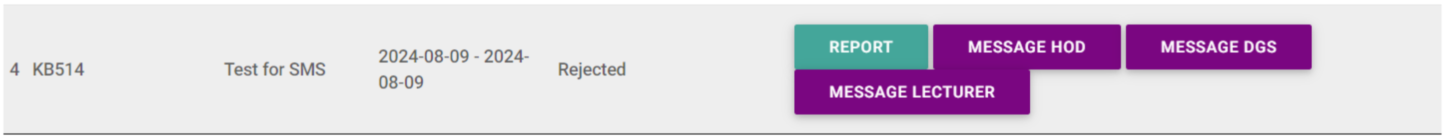
Search records

#	Venue/Classroom name	Booking Reason	Requester	Event date	Action
1	KA203	Test	Wong Jia Keen	2024-09-12- 2024-09-12	APPROVE REJECT
2	KB608	workshop	Hoo MH	2024-08-31- 2024-08-31	APPROVE REJECT
3	KA502	replacement class	Hoo MH	2024-08-23- 2024-08-23	APPROVE REJECT

Showing 1 to 3 of 3 entries

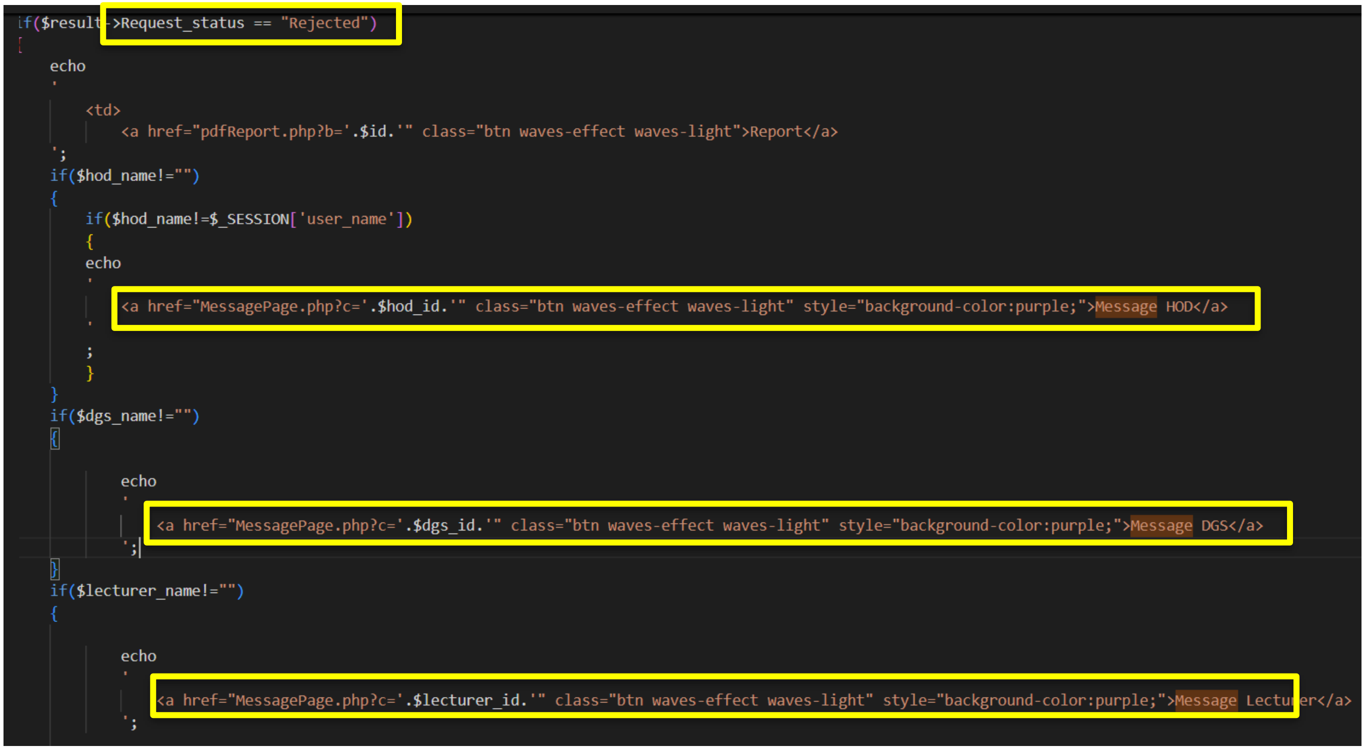
Figure 6.83: Message module that is red-circled

However to start a conversation, the end-user needs to submit a venue/classroom booking request. As the main aim of getting this feature built is for booking clarification purpose only, therefore the end-users cannot simply start a conversation with any end-users. Through submitting a venue booking request, the parties involved will be the DGS staff that holds the ultimate decision, HOD and the lecturers (if it was submitted by students). The booking requester could only start chatting with the parties once they had approve/reject such request. This could be done with simulating the status and to only show the chatting buttons when the status matches the condition.



4	KB514	Test for SMS	2024-08-09 - 2024-08-09	Rejected	REPORT MESSAGE HOD MESSAGE DGS MESSAGE LECTURER

Figure 6.84: Example of student's request that can message three parties

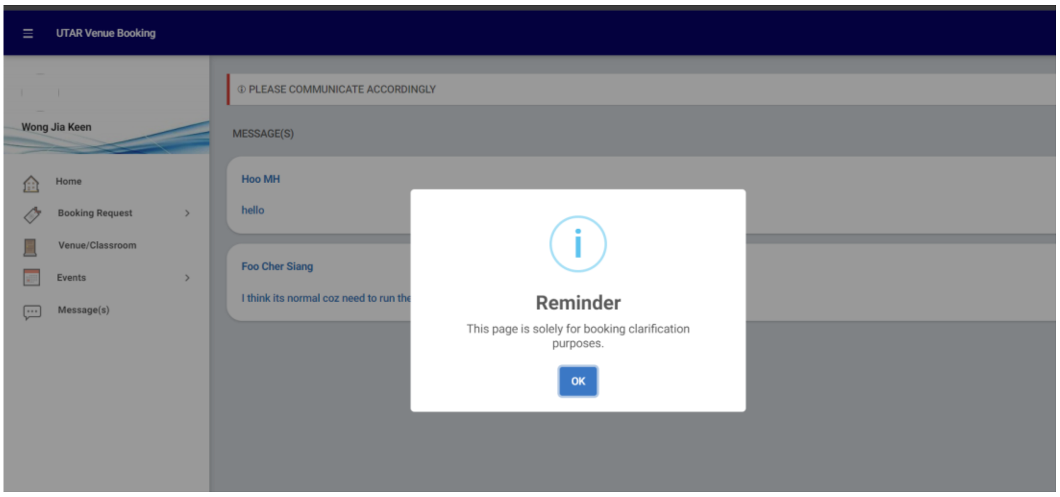


```
if($result->Request_status == "Rejected")
{
    echo
    '
        <td>
            <a href="pdfReport.php?b='.$id.'" class="btn waves-effect waves-light">Report</a>
    ';
    if($hod_name!="")
    {
        if($hod_name!=$_SESSION['user_name'])
        {
        echo
        '
            <a href="MessagePage.php?c='.$hod_id.'" class="btn waves-effect waves-light" style="background-color:purple;">Message HOD</a>
        '
        ;
        }
    }
    if($dgs_name!="")
    {

            echo
            '
                <a href="MessagePage.php?c='.$dgs_id.'" class="btn waves-effect waves-light" style="background-color:purple;">Message DGS</a>
            ';
    }
    if($lecturer_name!="")
    {

            echo
            '
                <a href="MessagePage.php?c='.$lecturer_id.'" class="btn waves-effect waves-light" style="background-color:purple;">Message Lecturer</a>
            ';
```

Figure 6.85: Technical implementation of the relationship of booking request's status with the presence of chatting links

6.6.1 Overview of Messages

If the end-users had already established a conversation with another, then he/she could directly to the Message(s) tab. There would be tabs of person who had chatted before, once the end-user has accessed the page. Before that, the end-users would get a reminder message, to remind them to use this feature wisely. This could be performed with the implementation of SweetAlert upon accessing.

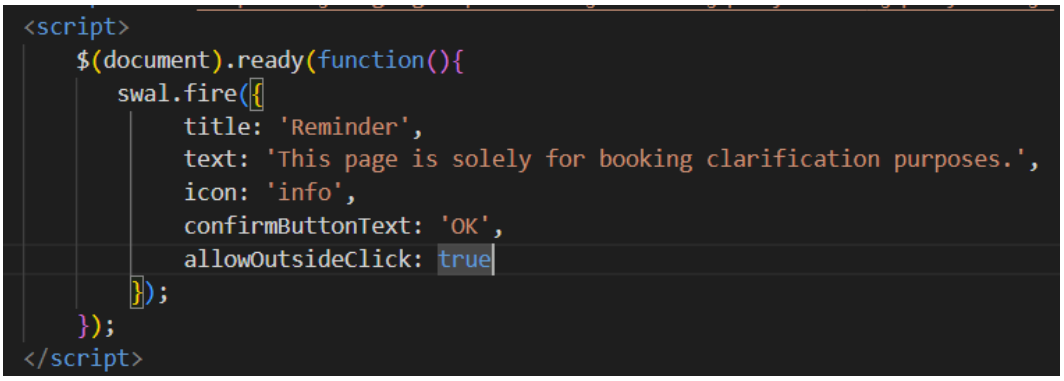


Hoo MH
hello

Foo Cher Siang
I think its normal coz need to run the

Reminder
This page is solely for booking clarification purposes.

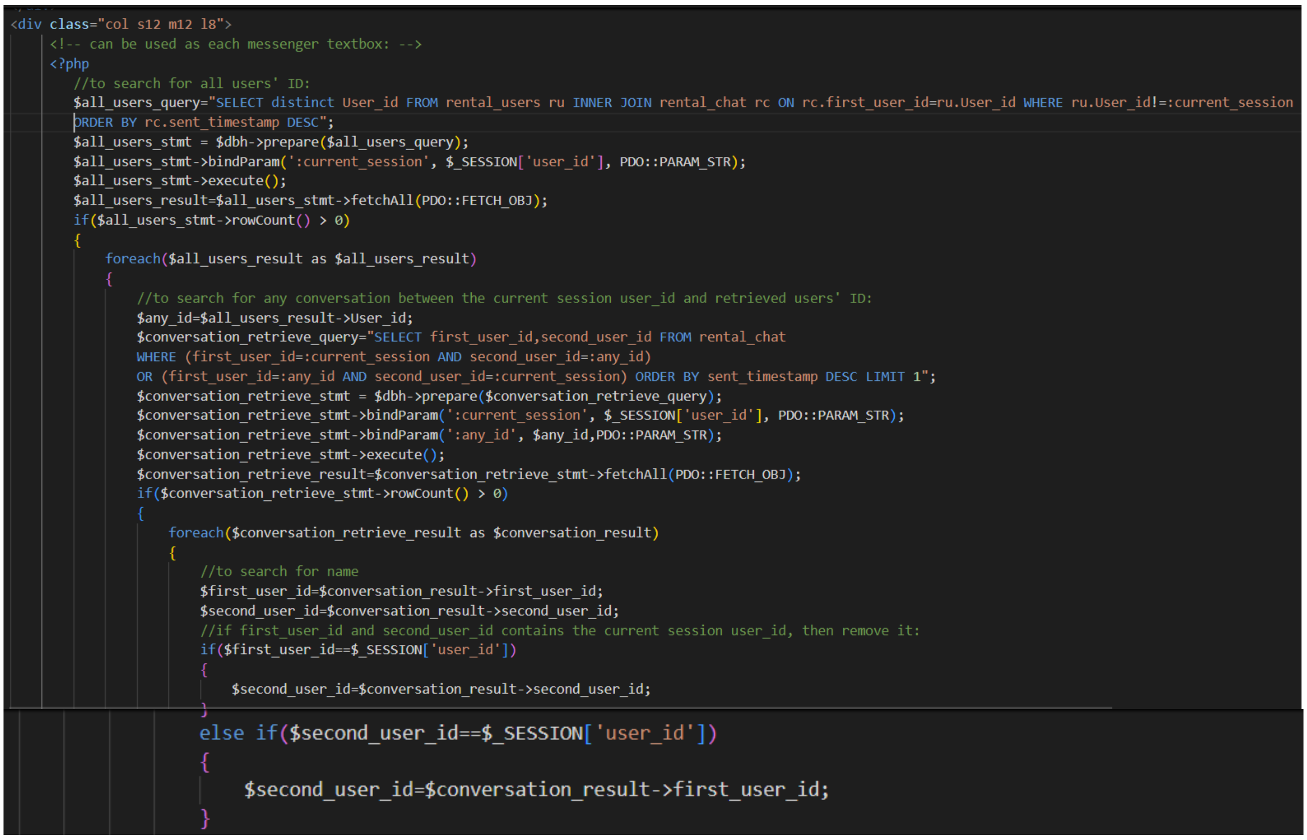
Figure 6.86: Reminder message once message page is accessed



```
<script>
    $(document).ready(function(){
        swal.fire({
            title: 'Reminder',
            text: 'This page is solely for booking clarification purposes.',
            icon: 'info',
            confirmButtonText: 'OK',
            allowOutsideClick: true
        });
    });
</script>
```

Figure 6.87: Technical implementation of reminder message

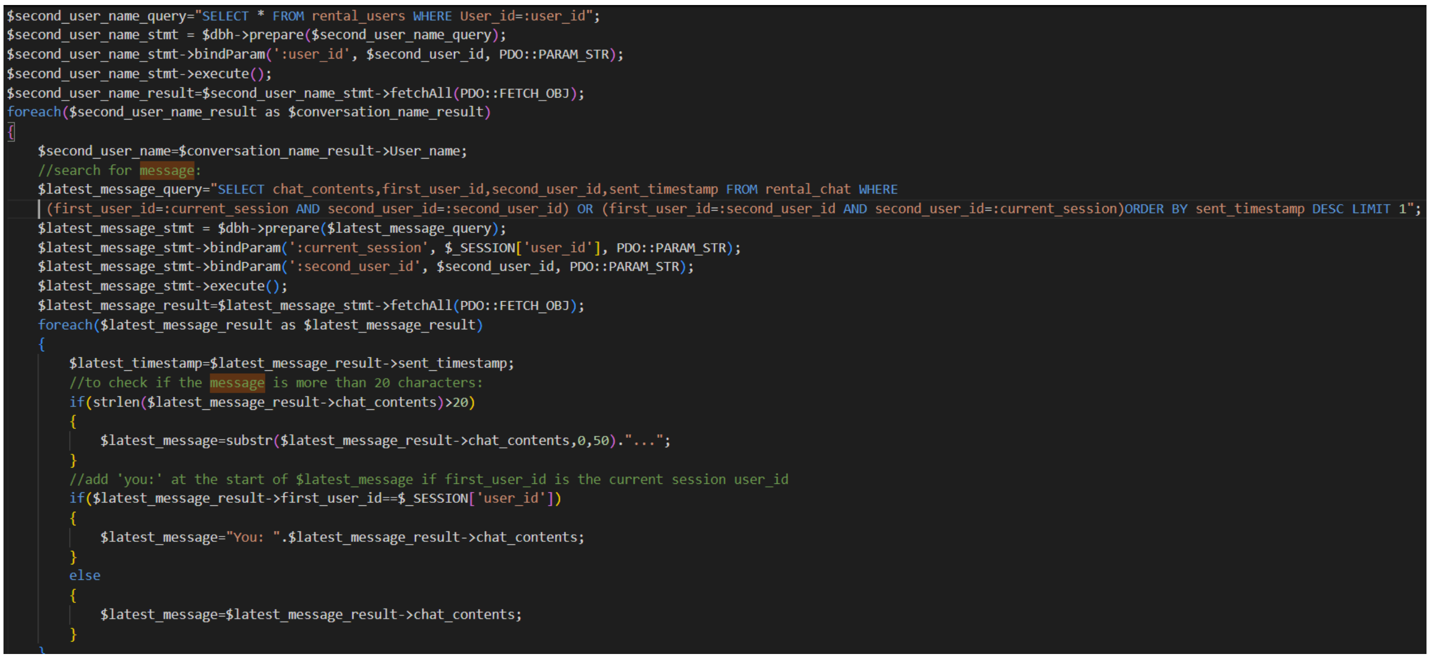
To explain the web display structure, firstly the page needs to perform searching for all users' ID, then the IDs fetched will be used to search if the current session's user has any conversation record with the fetched IDs or not. If the records truly exist, then both user's name will need to be fetched so that it could be displayed on each conversation tabs for identifying. Refer to figure 6.88 to know the technical implementation.



```
<div class="col s12 m12 l8">
    <!-- can be used as each messenger textbox: -->
    <?php
        //to search for all users' ID:
        $all_users_query="SELECT distinct User_id FROM rental_users ru INNER JOIN rental_chat rc ON rc.first_user_id=ru.User_id WHERE ru.User_id!=:current_session
        ORDER BY rc.sent_timestamp DESC";
        $all_users_stmt = $dbh->prepare($all_users_query);
        $all_users_stmt->bindParam(':current_session', $_SESSION['user_id'], PDO::PARAM_STR);
        $all_users_stmt->execute();
        $all_users_result=$all_users_stmt->fetchAll(PDO::FETCH_OBJ);
        if($all_users_stmt->rowCount() > 0)
        {
            foreach($all_users_result as $all_users_result)
            {
                //to search for any conversation between the current session user_id and retrieved users' ID:
                $any_id=$all_users_result->User_id;
                $conversation_retrieve_query="SELECT first_user_id,second_user_id FROM rental_chat
                WHERE (first_user_id=:current_session AND second_user_id=:any_id)
                OR (first_user_id=:any_id AND second_user_id=:current_session) ORDER BY sent_timestamp DESC LIMIT 1";
                $conversation_retrieve_stmt = $dbh->prepare($conversation_retrieve_query);
                $conversation_retrieve_stmt->bindParam(':current_session', $_SESSION['user_id'], PDO::PARAM_STR);
                $conversation_retrieve_stmt->bindParam(':any_id', $any_id,PDO::PARAM_STR);
                $conversation_retrieve_stmt->execute();
                $conversation_retrieve_result=$conversation_retrieve_stmt->fetchAll(PDO::FETCH_OBJ);
                if($conversation_retrieve_stmt->rowCount() > 0)
                {
                    foreach($conversation_retrieve_result as $conversation_result)
                    {
                        //to search for name
                        $first_user_id=$conversation_result->first_user_id;
                        $second_user_id=$conversation_result->second_user_id;
                        //if first_user_id and second_user_id contains the current session user_id, then remove it:
                        if($first_user_id==$_SESSION['user_id'])
                        {
                            $second_user_id=$conversation_result->second_user_id;
                        }
                        else if($second_user_id==$_SESSION['user_id'])
                        {
                            $second_user_id=$conversation_result->first_user_id;
                        }
```

Figure 6.88: Technical implementation of searching

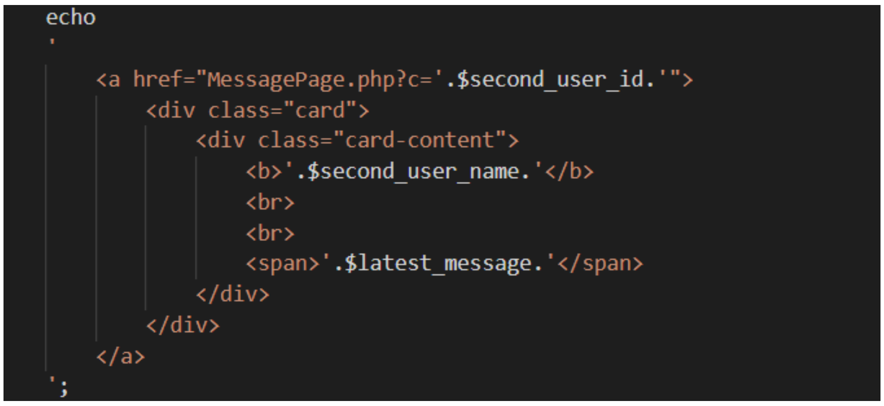
Regarding the message, the messages will be searched for each conversation partner, limiting to only the latest messages to be fetched. Utilizing the *strlen()* function to check the messages' length, if the message's length is more than 20 characters, then PHP will use the *substr()* function to cut the messages length to only 50 characters allowed, following by the '…' sign, indicates more contents written. Another notable feature is that if the latest message is sent by the current session's user, then a "You:" term will be placed on the opening of the latest message. This is considered as a non-functional requirement for the end-users to differentiate if the latest message was sent by which party.



```php
$second_user_name_query="SELECT * FROM rental_users WHERE User_id=:user_id";
$second_user_name_stmt = $dbh->prepare($second_user_name_query);
$second_user_name_stmt->bindParam(':user_id', $second_user_id, PDO::PARAM_STR);
$second_user_name_stmt->execute();
$second_user_name_result=$second_user_name_stmt->fetchAll(PDO::FETCH_OBJ);
foreach($second_user_name_result as $conversation_name_result)
{
    $second_user_name=$conversation_name_result->User_name;
    //search for message:
    $latest_message_query="SELECT chat_contents,first_user_id,second_user_id,sent_timestamp FROM rental_chat WHERE
    (first_user_id=:current_session AND second_user_id=:second_user_id) OR (first_user_id=:second_user_id AND second_user_id=:current_session)ORDER BY sent_timestamp DESC LIMIT 1";
    $latest_message_stmt = $dbh->prepare($latest_message_query);
    $latest_message_stmt->bindParam(':current_session', $_SESSION['user_id'], PDO::PARAM_STR);
    $latest_message_stmt->bindParam(':second_user_id', $second_user_id, PDO::PARAM_STR);
    $latest_message_stmt->execute();
    $latest_message_result=$latest_message_stmt->fetchAll(PDO::FETCH_OBJ);
    foreach($latest_message_result as $latest_message_result)
    {
        $latest_timestamp=$latest_message_result->sent_timestamp;
        //to check if the message is more than 20 characters:
        if(strlen($latest_message_result->chat_contents)>20)
        {
            $latest_message=substr($latest_message_result->chat_contents,0,50)."...";
        }
        //add 'you:' at the start of $latest_message if first_user_id is the current session user_id
        if($latest_message_result->first_user_id==$_SESSION['user_id'])
        {
            $latest_message="You: ".$latest_message_result->chat_contents;
        }
        else
        {
            $latest_message=$latest_message_result->chat_contents;
        }
```

Figure 6.89: Technical implementation of retrieved message overview

With the data ready, the conversation tabs are shown in <div> with unique styling, showing along the conversation partner’s name and the latest message shown.

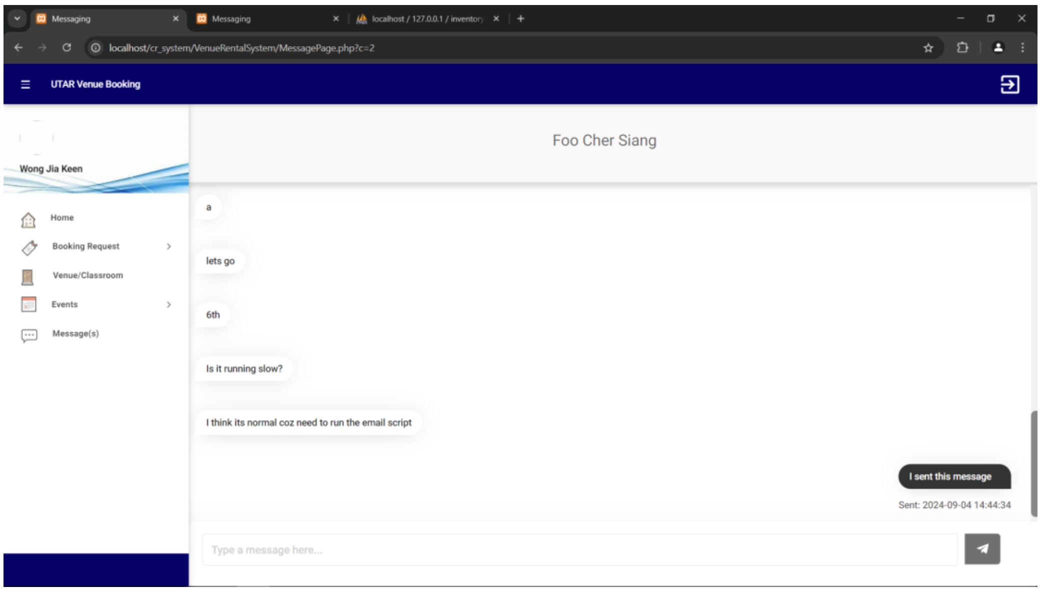


```php
echo
'
    <a href="MessagePage.php?c='.$second_user_id.'">
        <div class="card">
            <div class="card-content">
                <b>'.$second_user_name.'</b>
                <br>
                <br>
                <span>'.$latest_message.'</span>
            </div>
        </div>
    </a>
';
```

Figure 6.90: Each conversation blocks

6.6.2 Individual Message page

Once clicking into the individual message page, there are two parts to explore with, that is to retrieve the previous chat messages and the operation of sending.



Foo Cher Siang

a

lets go

6th

Is it running slow?

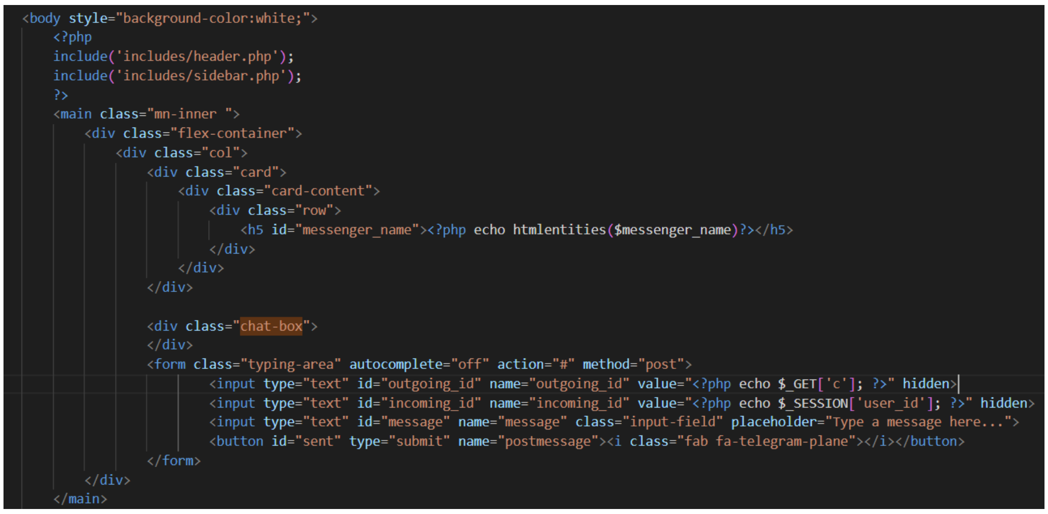
I think its normal coz need to run the email script

I sent this message

Sent: 2024-09-04 14:44:34

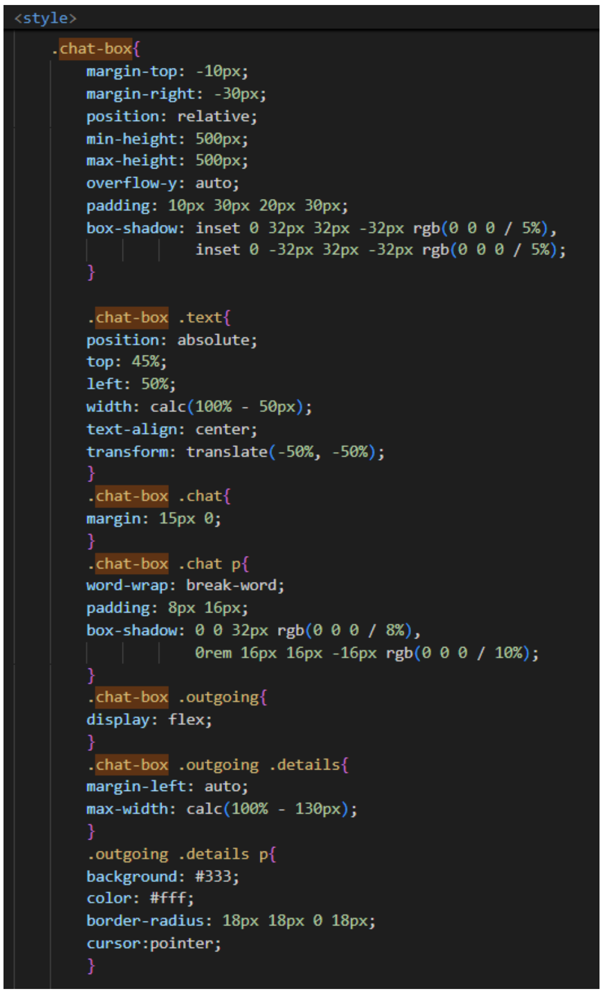
Figure 6.91: Specific conversation page

To explain the structure, the conversation partner's name is displayed in the form of header in a <div>. Styling is applied on the <div> so that the shadow effect is present. Whereas for the chat messages are displayed in a <div> as well, with a customized height and positioning, and the messages are divided into incoming and outgoing ones, differentiated in the positioning as well and the colours of the background. The typing area is built in the <form>, but consists of invisible elements, which are the sender and receiver's ID. This is to ensure the message sent could be stored in the database, along with the user ID.



```
<body style="background-color:white;">
    <?php
    include('includes/header.php');
    include('includes/sidebar.php');
    ?>
    <main class="mn-inner ">
        <div class="flex-container">
            <div class="col">
                <div class="card">
                    <div class="card-content">
                        <div class="row">
                            <h5 id="messenger_name"><?php echo htmlentities($messenger_name)?></h5>
                        </div>
                    </div>
                </div>

                <div class="chat-box">
                </div>
                <form class="typing-area" autocomplete="off" action="#" method="post">
                        <input type="text" id="outgoing_id" name="outgoing_id" value="<?php echo $_GET['c']; ?>" hidden>
                        <input type="text" id="incoming_id" name="incoming_id" value="<?php echo $_SESSION['user_id']; ?>" hidden>
                        <input type="text" id="message" name="message" class="input-field" placeholder="Type a message here...">
                        <button id="sent" type="submit" name="postmessage"><i class="fab fa-telegram-plane"></i></button>
                </form>
        </div>
    </main>
```

Figure 6.92: Technical implementation of the structure of chatting page



```
<style>
    .chat-box{
        margin-top: -10px;
        margin-right: -30px;
        position: relative;
        min-height: 500px;
        max-height: 500px;
        overflow-y: auto;
        padding: 10px 30px 20px 30px;
        box-shadow: inset 0 32px 32px -32px rgb(0 0 0 / 5%),
                    inset 0 -32px 32px -32px rgb(0 0 0 / 5%);
        }

        .chat-box .text{
        position: absolute;
        top: 45%;
        left: 50%;
        width: calc(100% - 50px);
        text-align: center;
        transform: translate(-50%, -50%);
        }
        .chat-box .chat{
        margin: 15px 0;
        }
        .chat-box .chat p{
        word-wrap: break-word;
        padding: 8px 16px;
        box-shadow: 0 0 32px rgb(0 0 0 / 8%),
                    0rem 16px 16px -16px rgb(0 0 0 / 10%);
        }
        .chat-box .outgoing{
        display: flex;
        }
        .chat-box .outgoing .details{
        margin-left: auto;
        max-width: calc(100% - 130px);
        }
        .outgoing .details p{
        background: #333;
        color: #fff;
        border-radius: 18px 18px 0 18px;
        cursor:pointer;
        }
```

Figure 6.93: Styling for the chatting page

The entire chat contents work through the JavaScript function, being placed externally and declared at the script section for the individual chatting page.

The retrieval and sending of messages are working through XMLHttpRequest object, or in abbreviation known as XHR. To explain, it is an instance used to interact with the servers, without the need of users to refresh the page to get the contents loaded and sent to the server. As this instance could be handled asynchronously or synchronously, therefore the developer needs to declare the boolean of either true (for asynchronous) or false when initiating communication request with the targeted server via the *open()* method (MDN contributors, n.d.). The operation could be represented with the following figure.

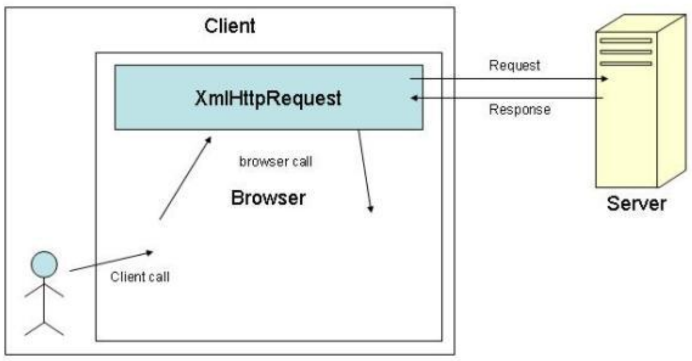
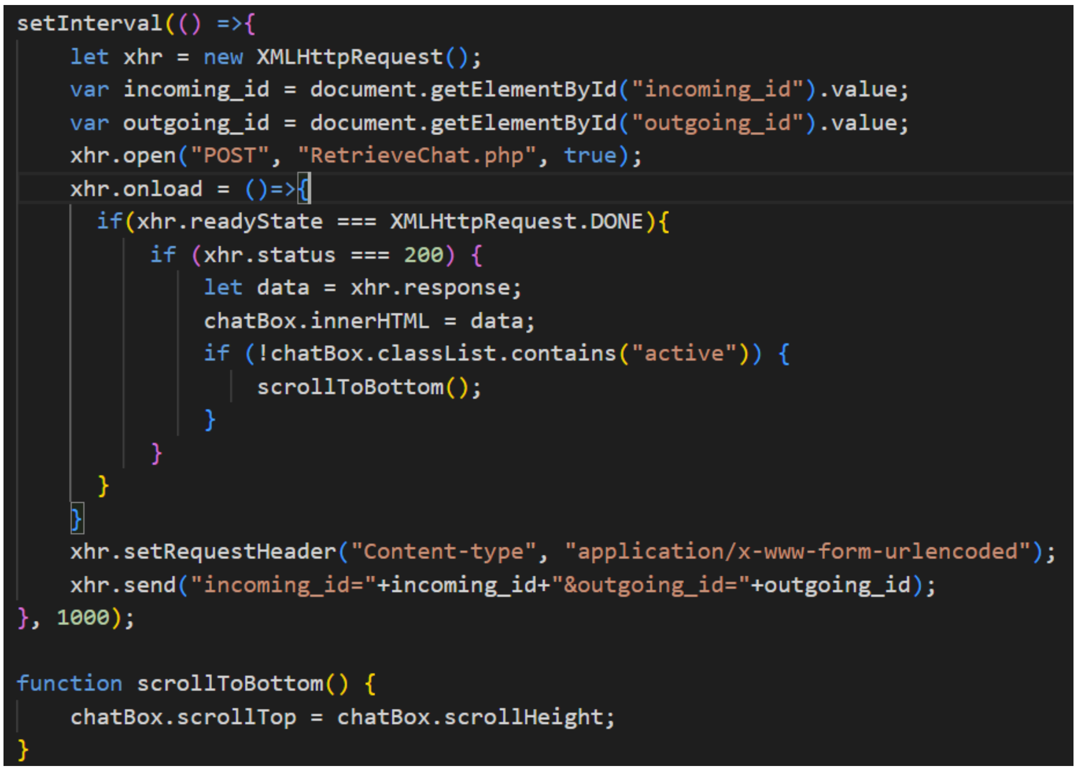


Figure 6.94: Graphical representation of how XMLHttpRequest works

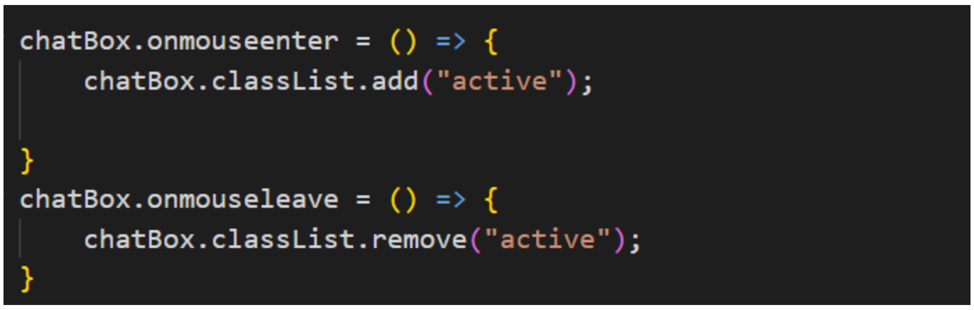
6.6.2.1 Retrieval of chat messages

For the case of retrieving chat messages, the JavaScript function needs to open a XMLHttpRequest, then send both user's ID with a POST request to another PHP page, that is dedicated for retrieving chat histories. While to retrieve the chat messages without the need of reloading the page, the sending action occurs every second constantly. To keep the user stay at the end of chatting page since the messages are displaying from top to bottom following the sent timing, a micro function is implemented here where if the user's cursor is not in the chatting area, then the chatting area will be scrolled to the bottom automatically.



```
setInterval(() =>{
    let xhr = new XMLHttpRequest();
    var incoming_id = document.getElementById("incoming_id").value;
    var outgoing_id = document.getElementById("outgoing_id").value;
    xhr.open("POST", "RetrieveChat.php", true);
    xhr.onload = ()=>{
      if(xhr.readyState === XMLHttpRequest.DONE){
          if (xhr.status === 200) {
              let data = xhr.response;
              chatBox.innerHTML = data;
              if (!chatBox.classList.contains("active")) {
                  scrollToBottom();
              }
          }
      }
    }
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.send("incoming_id="+incoming_id+"&outgoing_id="+outgoing_id);
}, 1000);

function scrollToBottom() {
    chatBox.scrollTop = chatBox.scrollHeight;
}
```

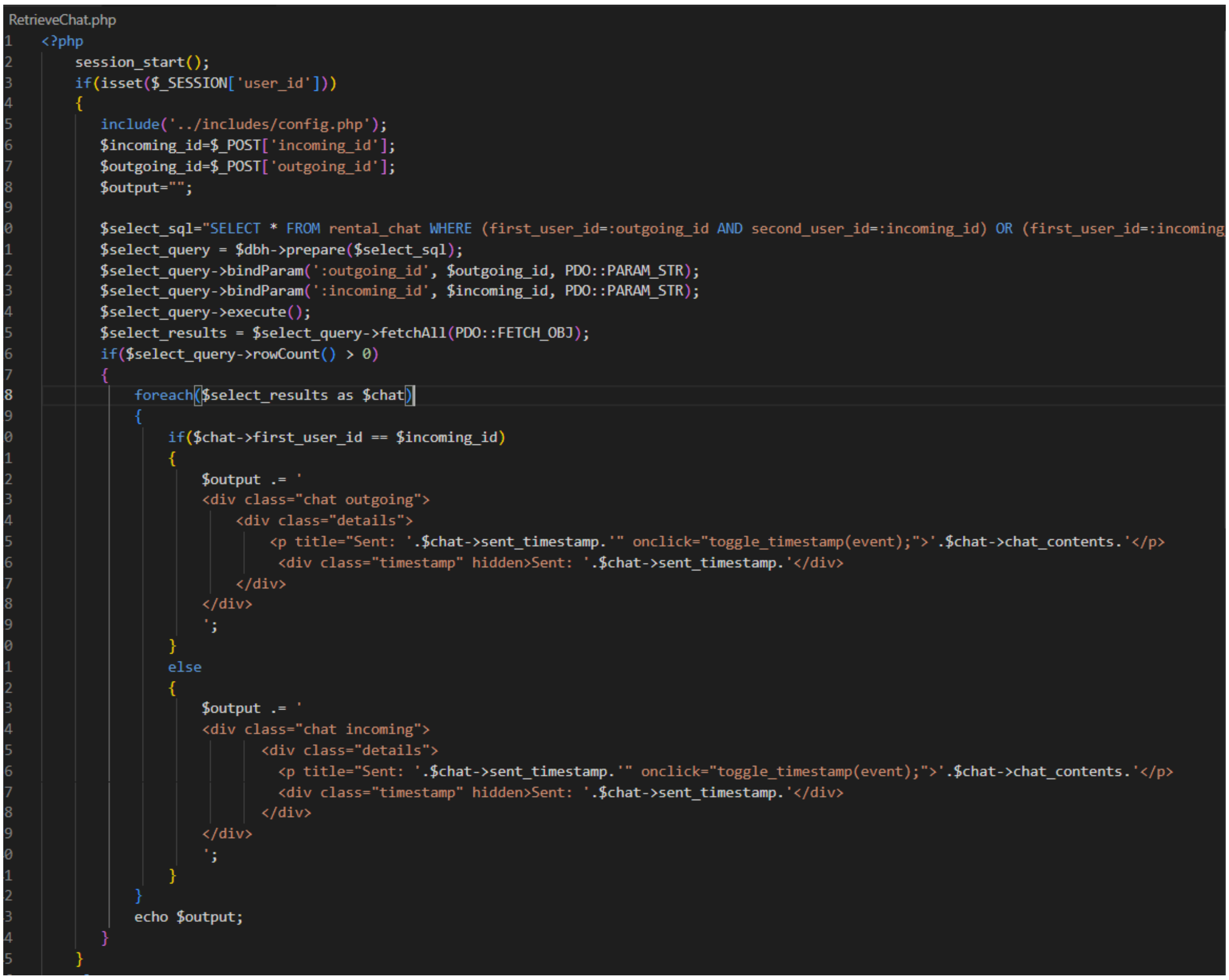
Figure 6.95: Technical implementation of retrieving chat messages



```
chatBox.onmouseenter = () => {
    chatBox.classList.add("active");

}
chatBox.onmouseleave = () => {
    chatBox.classList.remove("active");
}
```

Figure 6.96: Technical implementation of user staying at the chatting area or not

For the PHP page that runs the chat messages retrieval, usual interaction with the SQL database is necessary, just that the output is presented in the form of the HTML DOM (Document Object Model) method, which means amendment of HTML elements in the parent page based on those retrieved messages.



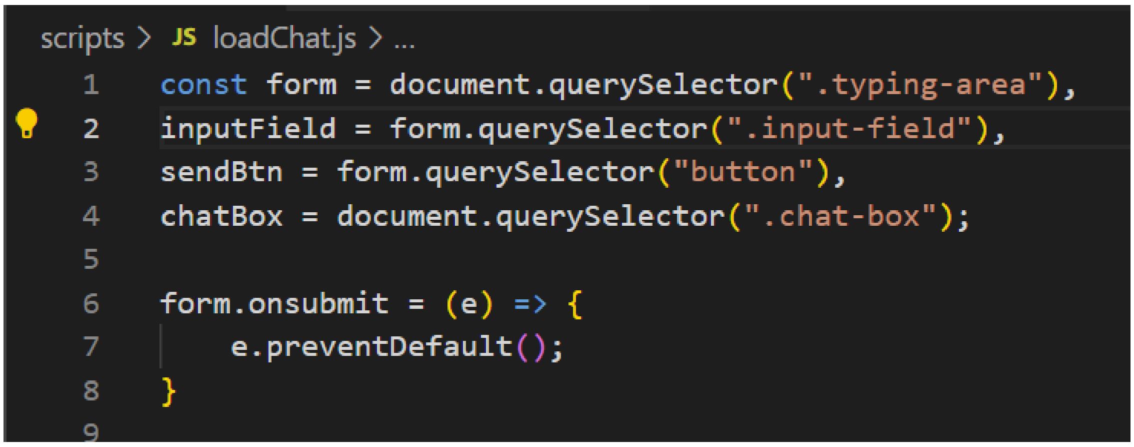
RetrieveChat.php

```
<?php
    session_start();
    if(isset($_SESSION['user_id']))
    {
        include('../includes/config.php');
        $incoming_id=$_POST['incoming_id'];
        $outgoing_id=$_POST['outgoing_id'];
        $output="";

        $select_sql="SELECT * FROM rental_chat WHERE (first_user_id=:outgoing_id AND second_user_id=:incoming_id) OR (first_user_id=:incoming
        $select_query = $dbh->prepare($select_sql);
        $select_query->bindParam(':outgoing_id', $outgoing_id, PDO::PARAM_STR);
        $select_query->bindParam(':incoming_id', $incoming_id, PDO::PARAM_STR);
        $select_query->execute();
        $select_results = $select_query->fetchAll(PDO::FETCH_OBJ);
        if($select_query->rowCount() > 0)
        {
            foreach($select_results as $chat)
            {
                if($chat->first_user_id == $incoming_id)
                {
                    $output .= '
                    <div class="chat outgoing">
                        <div class="details">
                            <p title="Sent: '.$chat->sent_timestamp.'" onclick="toggle_timestamp(event);">'.$chat->chat_contents.'</p>
                             <div class="timestamp" hidden>Sent: '.$chat->sent_timestamp.'</div>
                        </div>
                    </div>
                    ';
                }
                else
                {
                    $output .= '
                    <div class="chat incoming">
                           <div class="details">
                             <p title="Sent: '.$chat->sent_timestamp.'" onclick="toggle_timestamp(event);">'.$chat->chat_contents.'</p>
                             <div class="timestamp" hidden>Sent: '.$chat->sent_timestamp.'</div>
                           </div>
                    </div>
                    ';
                }
            }
            echo $output;
        }
    }
```

Figure 6.97: The database interaction to retrieve the chat messages

6.6.2.2 Send chat messages

Mentioning that the chatting area is built in <form>, it enables the contents typed to be sent with just a click of sent button. However, in usual forms, the page would get refreshed once the contents had been sent. To prevent this from happening, firstly the project author must do something to stop the reloading action whenever any contents are posted.



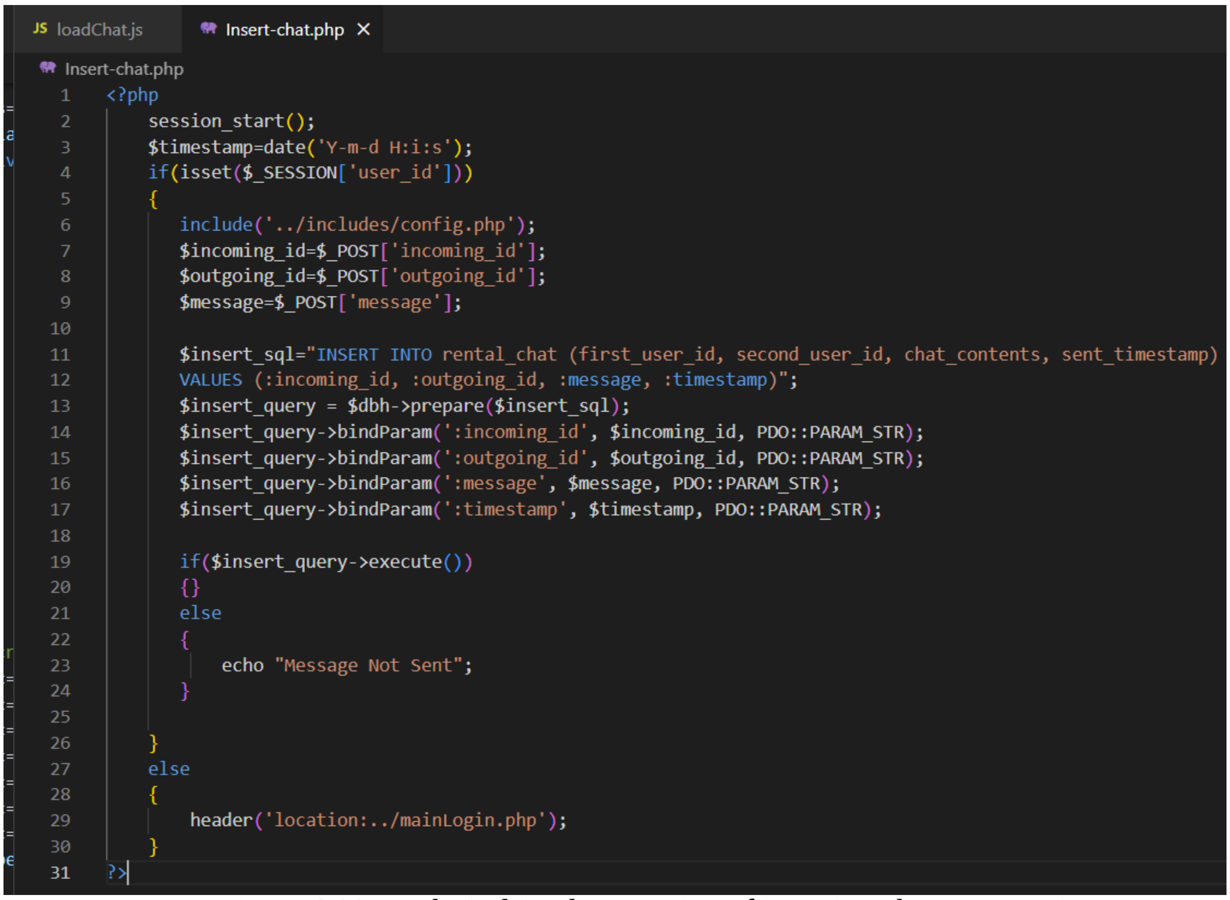
```
scripts > JS loadChat.js > ...
const form = document.querySelector(".typing-area"),
inputField = form.querySelector(".input-field"),
sendBtn = form.querySelector("button"),
chatBox = document.querySelector(".chat-box");

form.onsubmit = (e) => {
    e.preventDefault();
}
```

Figure 6.98: Technical implementation of stopping the page from reloading

In figure 6.98, declaration of variables in the first 4 lines are to inform JavaScript which elements to beware of. In the 6th line, this function tells the JavaScript to stop the page from propagating to other pages when the form submits any content. Reloading is counted as another form of propagating.

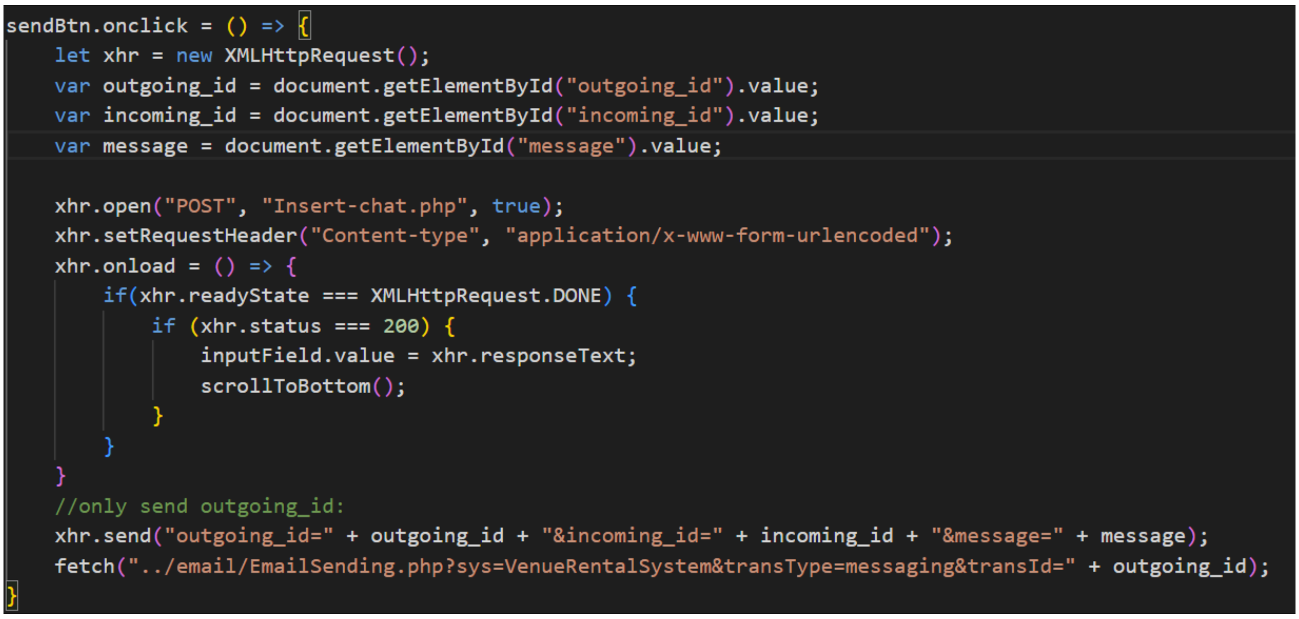
Regarding the send message operation, do recall that the <form> area consists of both users' ID and the typed contents. Therefore in this JavaScript that constantly waiting for user's action, whenever the sent button is clicked, then a new XHR object is initiated and sends a POST request as well. Working much similar with the retrieval of chat messages, but the targeted PHP file is different. A new PHP page is declared, specifically for sending chat messages only. Refer to figure 6.82 to learn more.

The variables sent through POST request are the users' ID and the chat contents. Moving to the newly declared PHP page, it performs a database interaction to inject the variables to store in it. The following diagram depicts the database operation for the PHP page.



```php
<?php
    session_start();
    $timestamp=date('Y-m-d H:i:s');
    if(isset($_SESSION['user_id']))
    {
        include('../includes/config.php');
        $incoming_id=$_POST['incoming_id'];
        $outgoing_id=$_POST['outgoing_id'];
        $message=$_POST['message'];

        $insert_sql="INSERT INTO rental_chat (first_user_id, second_user_id, chat_contents, sent_timestamp)
        VALUES (:incoming_id, :outgoing_id, :message, :timestamp)";
        $insert_query = $dbh->prepare($insert_sql);
        $insert_query->bindParam(':incoming_id', $incoming_id, PDO::PARAM_STR);
        $insert_query->bindParam(':outgoing_id', $outgoing_id, PDO::PARAM_STR);
        $insert_query->bindParam(':message', $message, PDO::PARAM_STR);
        $insert_query->bindParam(':timestamp', $timestamp, PDO::PARAM_STR);

        if($insert_query->execute())
        {}
        else
        {
            echo "Message Not Sent";
        }

    }
    else
    {
        header('location:../mainLogin.php');
    }
?>
```

Figure 6.99: Technical implementation of inserting chat contents into the database



```javascript
sendBtn.onclick = () => {
    let xhr = new XMLHttpRequest();
    var outgoing_id = document.getElementById("outgoing_id").value;
    var incoming_id = document.getElementById("incoming_id").value;
    var message = document.getElementById("message").value;

    xhr.open("POST", "Insert-chat.php", true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.onload = () => {
        if(xhr.readyState === XMLHttpRequest.DONE) {
            if (xhr.status === 200) {
                inputField.value = xhr.responseText;
                scrollToBottom();
            }
        }
    }
    //only send outgoing_id:
    xhr.send("outgoing_id=" + outgoing_id + "&incoming_id=" + incoming_id + "&message=" + message);
    fetch("../email/EmailSending.php?sys=VenueRentalSystem&transType=messaging&transId=" + outgoing_id);
}
```

Figure 6.100: Technical implementation of XHR to send chat messages when send button is clicked

The clicking of send button does not only send the messages, but also triggers the email notification function to remind the recipient as well.

6.7 Security mechanisms

This section describes the 4.4 and 4.5 non-functional requirements for the Security aspect. It could be possible if any of the users would have the intention to access other piece of information that are not related to themselves, and it is dangerous if they perform any amendments on the data. Therefore, this system has also tried possible measures to prevent such cases to happen by validating each users' status before accessing into the page. The following table summarizes the modules for validation.

Table 6.3: Modules for validation

Module	**Validation Modes**
All modules	Users had to log in the modules
Profile management	Users accessing other users' profile
Booking request rejection	- If the user's identity is not DGS staff or HOD - If the booking ID is null - If the booking is still awaiting for approval yet - If the booking had been rejected once by the same DGS staff or HOD
Booking PDF report viewing	- If the user's identity is not DGS staff/HOD or the booking requester - Booking request ID not found in database
Venue viewing	- Venue's name is not found in the database
Message	- If user attempts to chat with themselves
Event editing	- If user is not DGS staff/HOD
Trimester & Class Management	Users had to be DGS staff

6.7.1 Venue/Classroom Booking Request Identifier

Initially, each venue/classroom booking request submitted possess a unique ID value. However since the system uses POST values to access it for further management, the ID values shown on the URL link could be a confidentiality concern. Refer to the figure 6.89 to get a better understanding on the situation described.

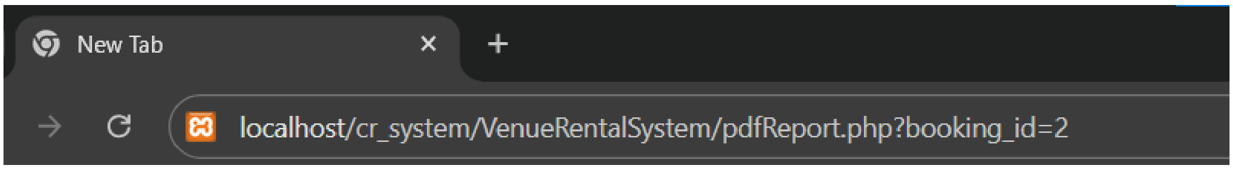

New Tab
localhost/cr_system/VenueRentalSystem/pdfReport.php?booking_id=2

Figure 6.101: Initial method to access booking request information

Due to this reason, the project author refers to the YouTube link, that is to encrypt the information with a certain length of characters. Firstly, the *booking_id* in the URL link needs to be amended so that it could be vague.

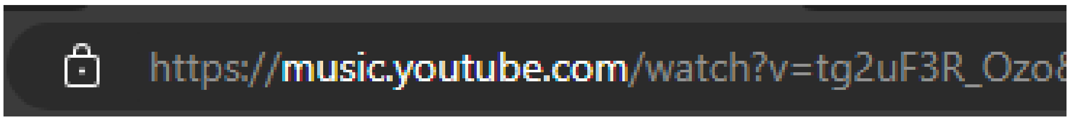
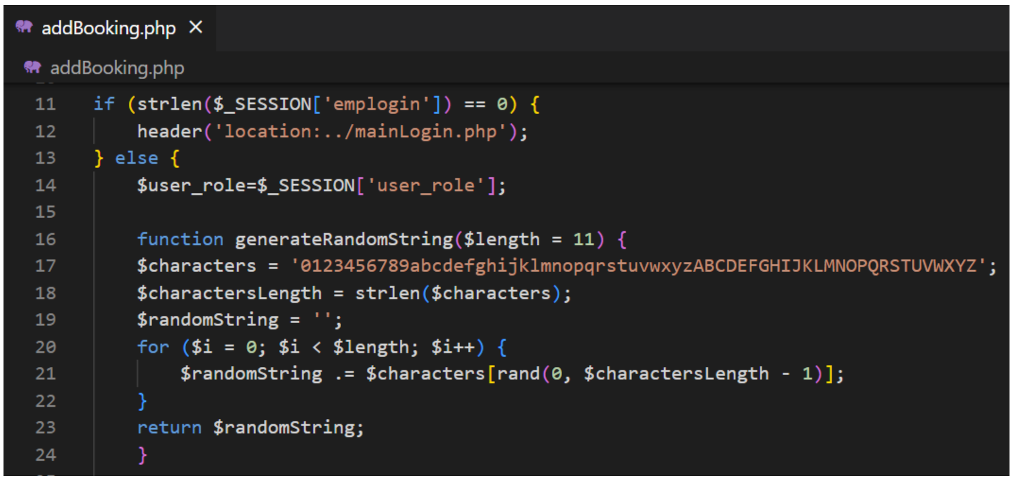


Figure 6.102: Example of YouTube links to encrypt each content

To generate the randomized string of characters when submitting venue/classroom usage request, a function which generates the string needs to be declared first. Depending on the requirements, the characters defined in this project consists of numeric and alphabetical characters, inclusive of uppercase and lowercase as well. Using the for-loop function that iterates the defined length size, the output string will be generated with looping the characters randomly, by decreasing the defined length size by 1 every time each loop counts completed. It was until the loop count reduces to zero, then the string is considered completed for generation. That is how the randomized string generated for identifying each venue usage request, instead of just numeric ID values.

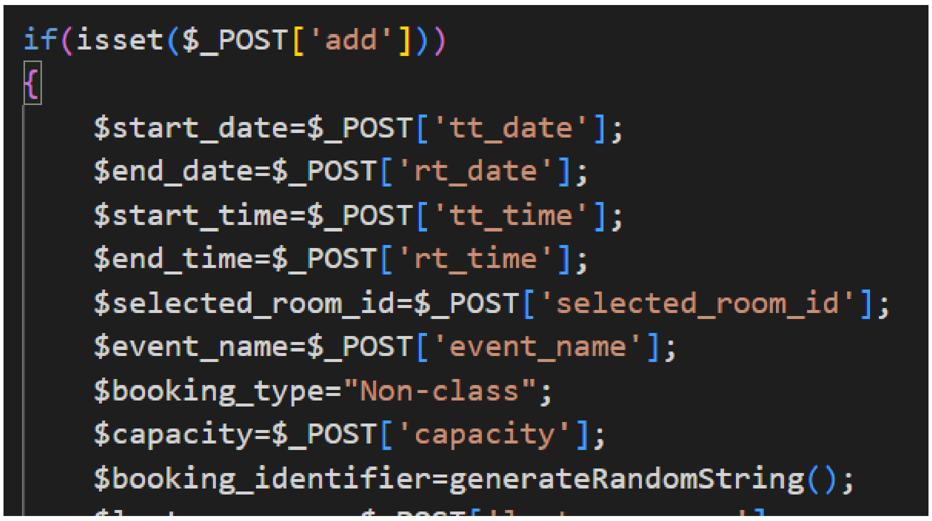


addBooking.php

```php
if (strlen($_SESSION['emplogin']) == 0) {
    header('location:../mainLogin.php');
} else {
    $user_role=$_SESSION['user_role'];

    function generateRandomString($length = 11) {
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomString;
    }
```

Figure 6.103: Generation function of randomized characters

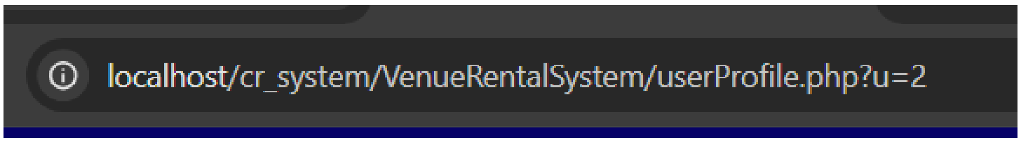
Each time when a booking request is initiated, the identifier value would be called in such method:



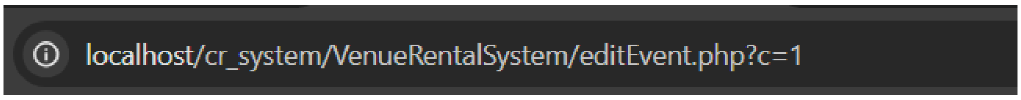
```php
if(isset($_POST['add']))
{
    $start_date=$_POST['tt_date'];
    $end_date=$_POST['rt_date'];
    $start_time=$_POST['tt_time'];
    $end_time=$_POST['rt_time'];
    $selected_room_id=$_POST['selected_room_id'];
    $event_name=$_POST['event_name'];
    $booking_type="Non-class";
    $capacity=$_POST['capacity'];
    $booking_identifier=generateRandomString();
```

Figure 6.104: Calling the generation function

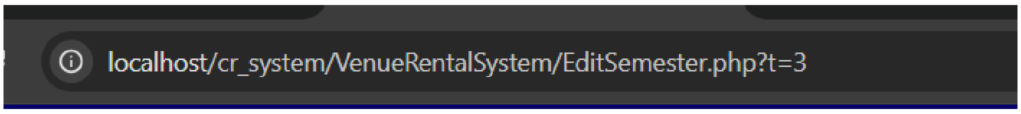
Although the generation function only applies to booking request, but the accessing rule is widely applied on other functional areas, like profile accessing, venue information accessing, trimester information accessing, class information accessing and so on. As the class information relates with the venue booked, so the identifier value used for class info are the same with the venue/classroom usage request.



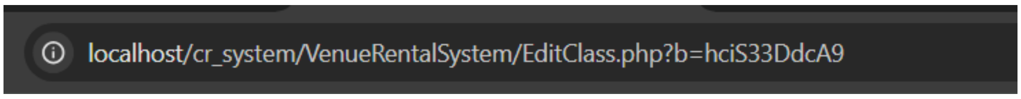
localhost/cr_system/VenueRentalSystem/userProfile.php?u=2



localhost/cr_system/VenueRentalSystem/editEvent.php?c=1



localhost/cr_system/VenueRentalSystem/EditSemester.php?t=3



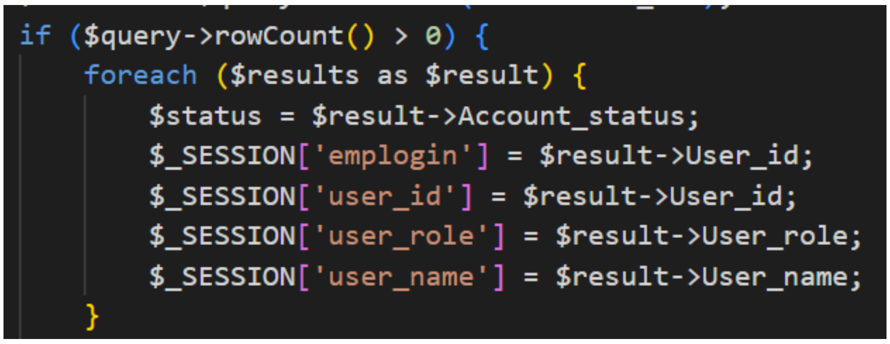
localhost/cr_system/VenueRentalSystem/EditClass.php?b=hciS33DdcA9

Figures 6.105: Accessing values of each piece of information

6.7.2 Accessing validation rules

Due to authority level that possesses accessing restrictions on certain functionalities, validation needs to be done on verifying the user's identity. Another point of validating is that users may attempt to intrude information pages that are not related, which is considered as an offence of confidentiality. This section describes the technical implementation of how to perform the validation.

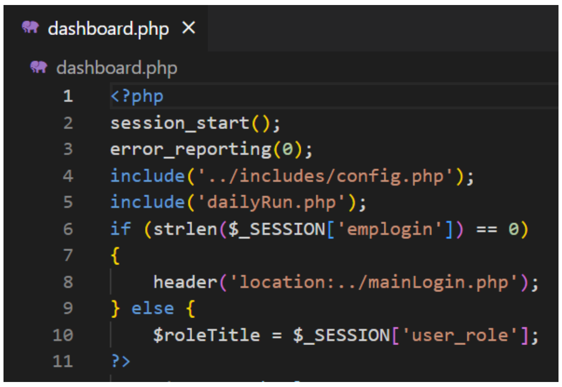
As users logged into the system, the session data holds their ID, status level and their names, being fetched from the database. The session data will be used for validation later.



```
if ($query->rowCount() > 0) {
    foreach ($results as $result) {
        $status = $result->Account_status;
        $_SESSION['emplogin'] = $result->User_id;
        $_SESSION['user_id'] = $result->User_id;
        $_SESSION['user_role'] = $result->User_role;
        $_SESSION['user_name'] = $result->User_name;
    }
```

Figure 6.106: Technical implementation of storing session data when login

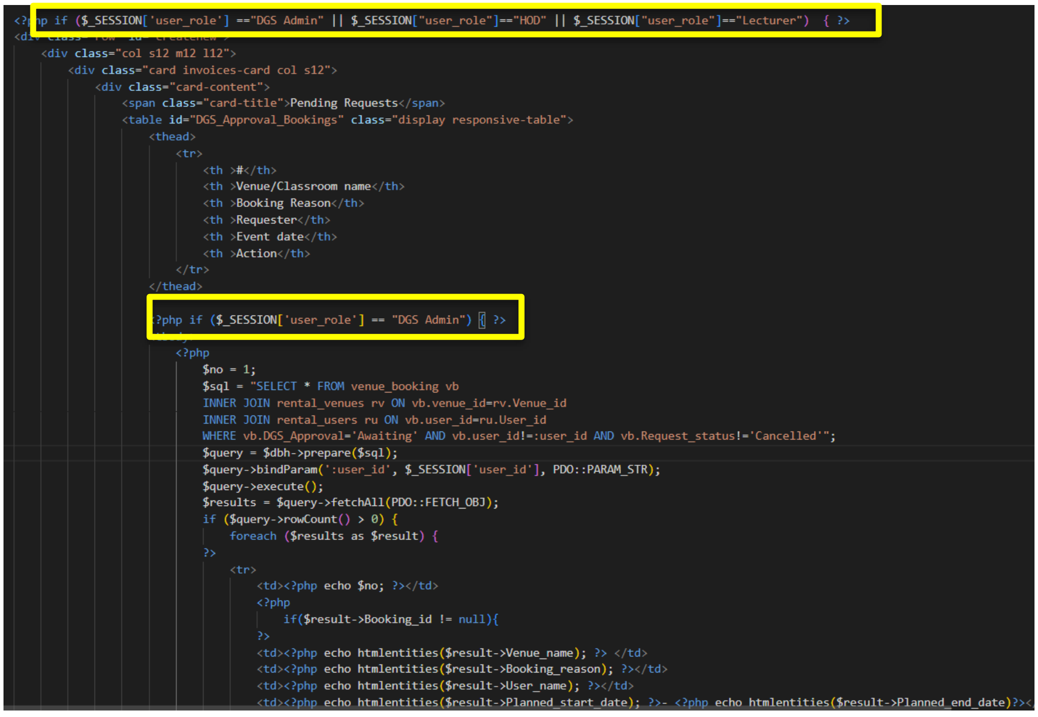
To check if the user has logged in or not, PHP will check the string length for the session data ‘emplogin’. As mentioned this data will be filled with user ID whose length would not be 0, then the login is considered valid. Refer to the 6th line in Figure 6.95 to know the operation.



```
dashboard.php ×
dashboard.php
1   <?php
2   session_start();
3   error_reporting(0);
4   include('../includes/config.php');
5   include('dailyRun.php');
6   if (strlen($_SESSION['emplogin']) == 0)
7   {
8       header('location:../mainLogin.php');
9   } else {
10      $roleTitle = $_SESSION['user_role'];
11  ?>
```

Figure 6.107: Technical implementation of validating whether the user has logged in or not

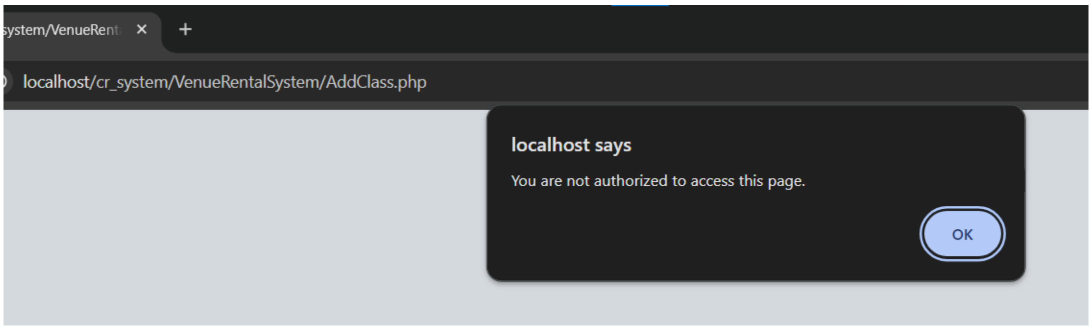
Based on the $roleTitle demonstrated in 10th line in figure 6.90, that indicates the authority level of the logged in user. Certain piece of information will be shown according to the authority level, like DGS staff and HOD could know which requests are pending for approval. Do notice the yellow squared part in figure 6.96 to know about this operation.



```
<?php if ($_SESSION['user_role'] =="DGS Admin" || $_SESSION["user_role"]=="HOD" || $_SESSION["user_role"]=="Lecturer")  { ?>
[illegible]
    <div class="col s12 m12 l12">
        <div class="card invoices-card col s12">
            <div class="card-content">
                <span class="card-title">Pending Requests</span>
                <table id="DGS_Approval_Bookings" class="display responsive-table">
                    <thead>
                        <tr>
                            <th >#</th>
                            <th >Venue/Classroom name</th>
                            <th >Booking Reason</th>
                            <th >Requester</th>
                            <th >Event date</th>
                            <th >Action</th>
                        </tr>
                    </thead>

                    <?php if ($_SESSION['user_role'] == "DGS Admin") { ?>
                        <?php
                            $no = 1;
                            $sql = "SELECT * FROM venue_booking vb
                            INNER JOIN rental_venues rv ON vb.venue_id=rv.Venue_id
                            INNER JOIN rental_users ru ON vb.user_id=ru.User_id
                            WHERE vb.DGS_Approval='Awaiting' AND vb.user_id!=:user_id AND vb.Request_status!='Cancelled'";
                            $query = $dbh->prepare($sql);
                            $query->bindParam(':user_id', $_SESSION['user_id'], PDO::PARAM_STR);
                            $query->execute();
                            $results = $query->fetchAll(PDO::FETCH_OBJ);
                            if ($query->rowCount() > 0) {
                                foreach ($results as $result) {
                            ?>
                                <tr>
                                    <td><?php echo $no; ?></td>
                                    <?php
                                        if($result->Booking_id != null){
                                    ?>
                                    <td><?php echo htmlentities($result->Venue_name); ?> </td>
                                    <td><?php echo htmlentities($result->Booking_reason); ?></td>
                                    <td><?php echo htmlentities($result->User_name); ?></td>
                                    <td><?php echo htmlentities($result->Planned_start_date); ?>- <?php echo htmlentities($result->Planned_end_date)?><
```

Figure 6.108: Displaying of information according to the status of the logged-user

Similarly, the authority level of users used for validation are also used in accessing booking request rejection page, event management, trimester and class management pages.



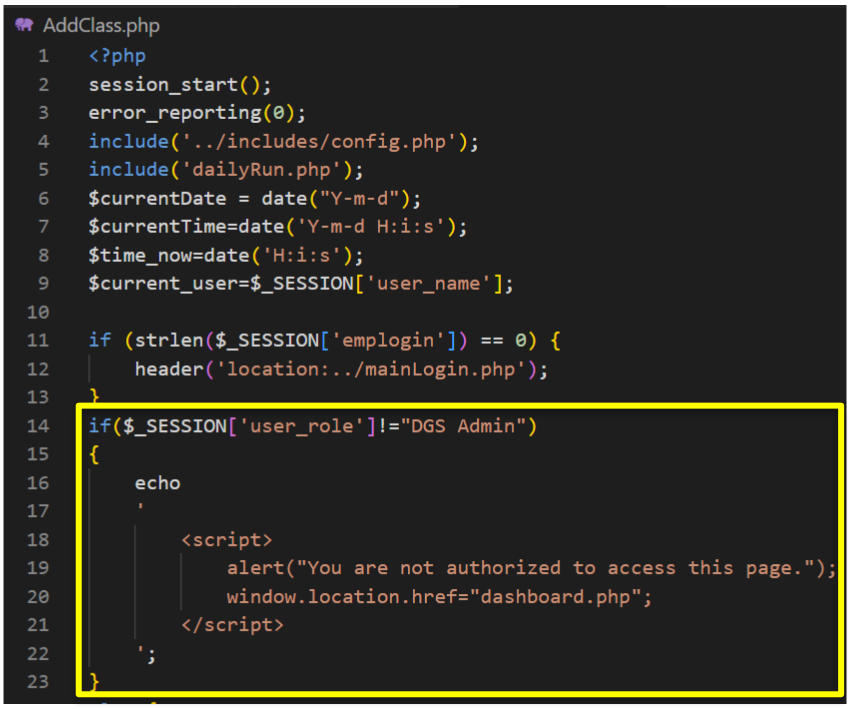
ystem/VenueRent ×

localhost/cr_system/VenueRentalSystem/AddClass.php

localhost says

You are not authorized to access this page.

Figure 6.109: Alert message if users access the page that should not be

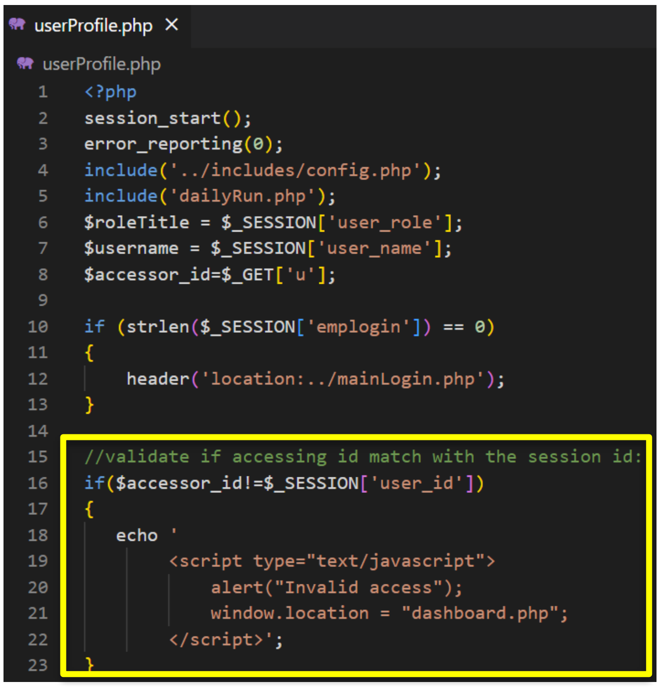


AddClass.php

```
<?php
session_start();
error_reporting(0);
include('../includes/config.php');
include('dailyRun.php');
$currentDate = date("Y-m-d");
$currentTime=date('Y-m-d H:i:s');
$time_now=date('H:i:s');
$current_user=$_SESSION['user_name'];

if (strlen($_SESSION['emplogin']) == 0) {
    header('location:../mainLogin.php');
}
if($_SESSION['user_role']!="DGS Admin")
{
    echo
    '
        <script>
            alert("You are not authorized to access this page.");
            window.location.href="dashboard.php";
        </script>
    ';
}
```

Figure 6.110: Technical implementation of user access validation based on authority level

Subsequently, if the page is not related to current logged in user, a checking could be done as well to prohibit the user from accessing. Using the case of profile viewing, every user could only view their own profile. A comparison of the current session data with the accessing data could be performed, like what presented in figure 6.111. The effects are likely the same as in figure 6.109.



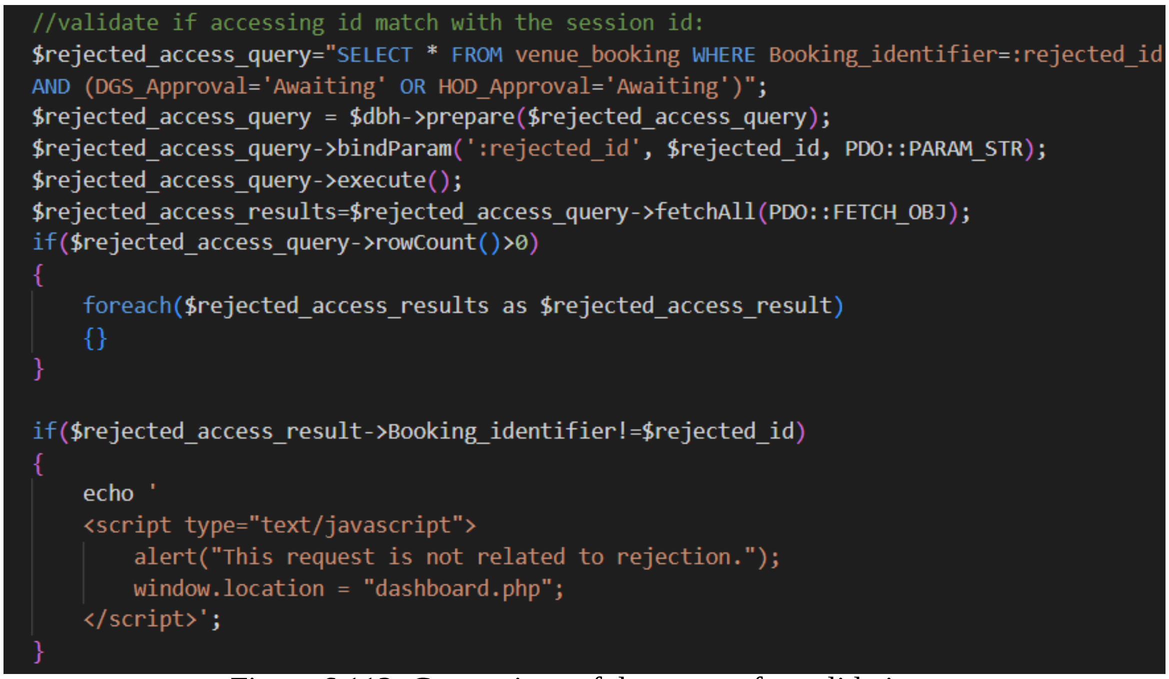
userProfile.php ×

userProfile.php

```php
<?php
session_start();
error_reporting(0);
include('../includes/config.php');
include('dailyRun.php');
$roleTitle = $_SESSION['user_role'];
$username = $_SESSION['user_name'];
$accessor_id=$_GET['u'];

if (strlen($_SESSION['emplogin']) == 0)
{
    header('location:../mainLogin.php');
}

//validate if accessing id match with the session id:
if($accessor_id!=$_SESSION['user_id'])
{
   echo '
        <script type="text/javascript">
            alert("Invalid access");
            window.location = "dashboard.php";
        </script>';
}
```

Figure 6.111: Comparison of data for validating purpose

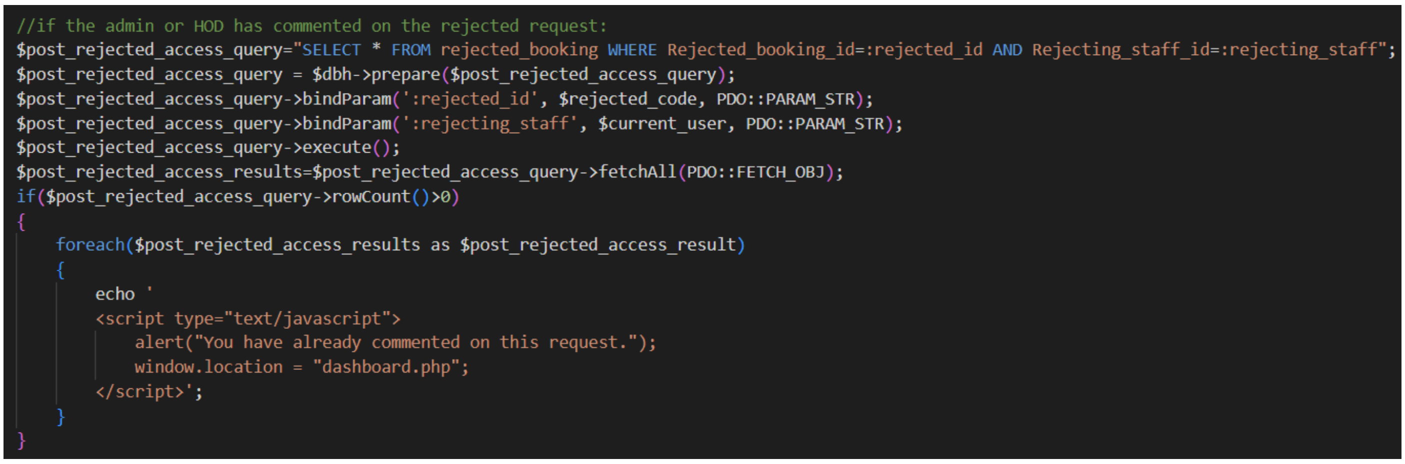
Status of information could limit the access as well. Taking the case of rejecting venue/classroom booking request, requests that have been rejected since early or cancelled should not be presenting in this page anymore. For the example presented in Figure 6.112, data fetching needs to be done on the accessing booking request, to verify the approval status. If the request is still pending, then the rejection page should be appeared, so that the DGS admin or HOD could reject it.



```php
//validate if accessing id match with the session id:
$rejected_access_query="SELECT * FROM venue_booking WHERE Booking_identifier=:rejected_id
AND (DGS_Approval='Awaiting' OR HOD_Approval='Awaiting')";
$rejected_access_query = $dbh->prepare($rejected_access_query);
$rejected_access_query->bindParam(':rejected_id', $rejected_id, PDO::PARAM_STR);
$rejected_access_query->execute();
$rejected_access_results=$rejected_access_query->fetchAll(PDO::FETCH_OBJ);
if($rejected_access_query->rowCount()>0)
{
    foreach($rejected_access_results as $rejected_access_result)
    {}
}

if($rejected_access_result->Booking_identifier!=$rejected_id)
{
    echo '
    <script type="text/javascript">
        alert("This request is not related to rejection.");
        window.location = "dashboard.php";
    </script>';
}
```

Figure 6.112: Comparison of data status for validating purpose

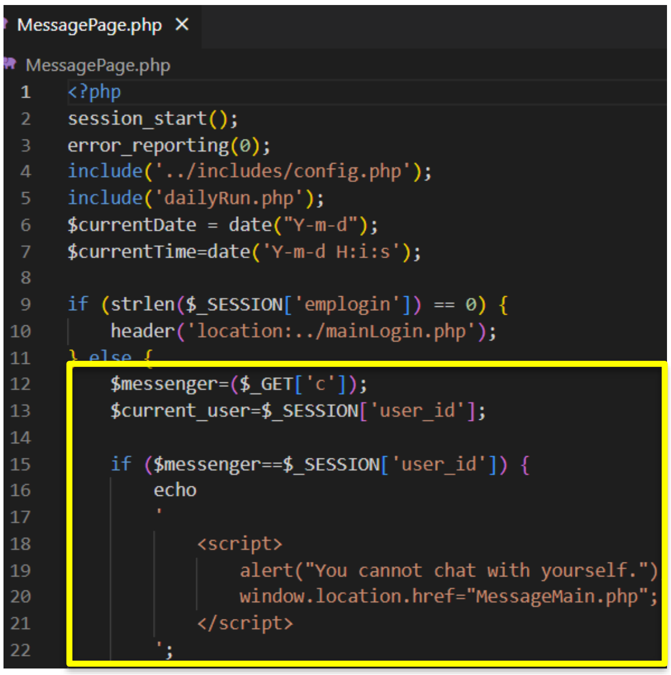
Else, if the booking request has been rejected by the current logged-in user, then the page should not be accessed anymore for the booking request.



```php
//if the admin or HOD has commented on the rejected request:
$post_rejected_access_query="SELECT * FROM rejected_booking WHERE Rejected_booking_id=:rejected_id AND Rejecting_staff_id=:rejecting_staff";
$post_rejected_access_query = $dbh->prepare($post_rejected_access_query);
$post_rejected_access_query->bindParam(':rejected_id', $rejected_code, PDO::PARAM_STR);
$post_rejected_access_query->bindParam(':rejecting_staff', $current_user, PDO::PARAM_STR);
$post_rejected_access_query->execute();
$post_rejected_access_results=$post_rejected_access_query->fetchAll(PDO::FETCH_OBJ);
if($post_rejected_access_query->rowCount()>0)
{
    foreach($post_rejected_access_results as $post_rejected_access_result)
    {
        echo '
        <script type="text/javascript">
            alert("You have already commented on this request.");
            window.location = "dashboard.php";
        </script>';
    }
}
```

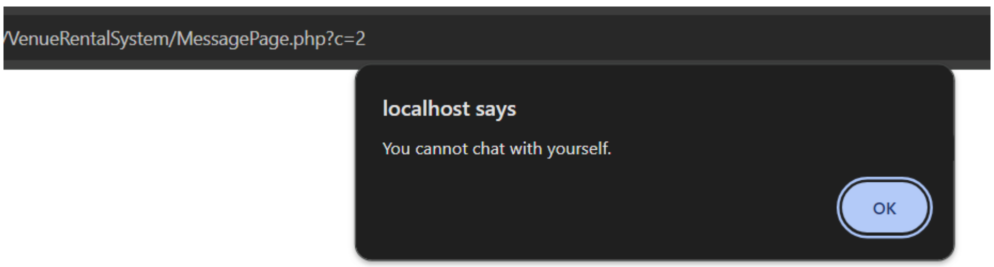
Figure 6.113: Comparison of booking request status for validating purpose

In the case of instant messaging module, chatting with own is impossible, as it serves no purpose and waste the data storage. This could happen if the end-user is attentive enough to edit the links in the URL. Again, this validation could be done by comparing the current logged in user's ID and the conversation partner's ID. If both are the same, then the user would be warned with a message, redirecting them to the message viewing page as well.



```php
<?php
session_start();
error_reporting(0);
include('../includes/config.php');
include('dailyRun.php');
$currentDate = date("Y-m-d");
$currentTime=date('Y-m-d H:i:s');

if (strlen($_SESSION['emplogin']) == 0) {
    header('location:../mainLogin.php');
} else {
    $messenger=($_GET['c']);
    $current_user=$_SESSION['user_id'];

    if ($messenger==$_SESSION['user_id']) {
        echo
        '
            <script>
                alert("You cannot chat with yourself.")
                window.location.href="MessageMain.php";
            </script>
        ';
```

Figure 6.114: Comparison of user ID for validation purpose

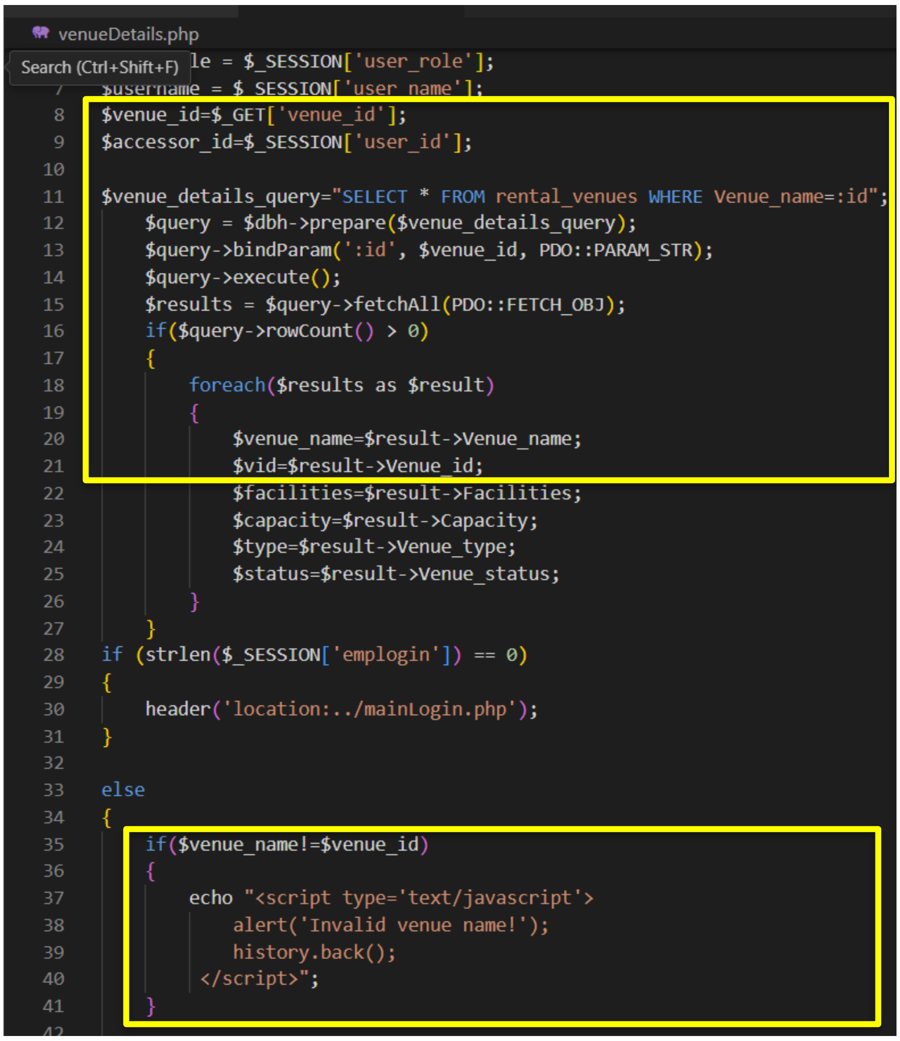


localhost says

You cannot chat with yourself.

Figure 6.115: Alert message if the user amends the ID in the URL link

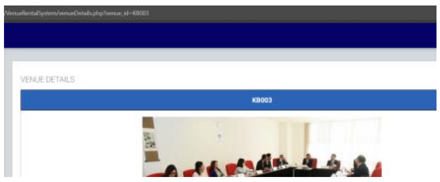
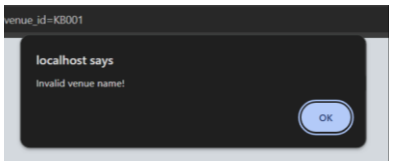
For the case if the user attempts to access data that are not available in the database, the validation also does its work here. By checking the accessing name with the pre-fetch data, PHP could decide the access right of the user. If such data exists, then the user could access. In contrast, users could not access the page. Taking the case of searching venue's name, such processing is shown in figure 6.116 and the effects are presenting in table 6.3.



```php
le = $_SESSION['user_role'];
$username = $_SESSION['user_name'];
$venue_id=$_GET['venue_id'];
$accessor_id=$_SESSION['user_id'];

$venue_details_query="SELECT * FROM rental_venues WHERE Venue_name=:id";
    $query = $dbh->prepare($venue_details_query);
    $query->bindParam(':id', $venue_id, PDO::PARAM_STR);
    $query->execute();
    $results = $query->fetchAll(PDO::FETCH_OBJ);
    if($query->rowCount() > 0)
    {
        foreach($results as $result)
        {
            $venue_name=$result->Venue_name;
            $vid=$result->Venue_id;
            $facilities=$result->Facilities;
            $capacity=$result->Capacity;
            $type=$result->Venue_type;
            $status=$result->Venue_status;
        }
    }
if (strlen($_SESSION['emplogin']) == 0)
{
    header('location:../mainLogin.php');
}

else
{
    if($venue_name!=$venue_id)
    {
        echo "<script type='text/javascript'>
            alert('Invalid venue name!');
            history.back();
        </script>";
    }
```

Figure 6.116: Data fetching and validation

Table 6.4: Comparison of valid data and invalid data for accessing

	Valid data	**Invalid data**
Effect(s)	User is shown with information.	- An alert pops out, telling invalid information. - User is relocated to the previous page.
Example	 /VenueRentalSystem/venueDetails.php?venue_id=KB003 	 venue_id=KB001 localhost says Invalid venue name!

6.8 DGS desired functions

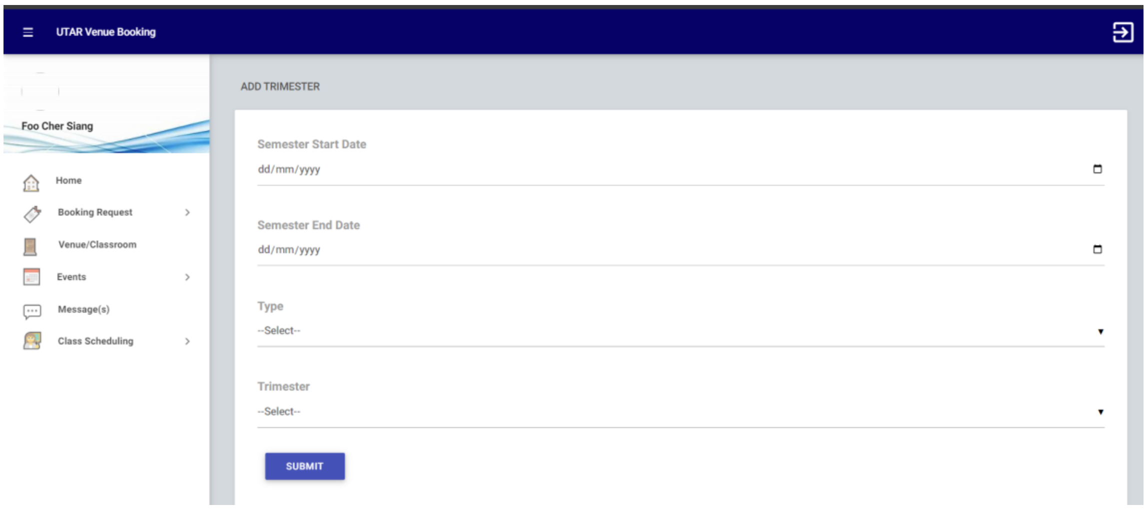
This section is an additional requirement from the DGS staff, since the following functions are part of the venue/classroom allocation. Since venues/classrooms are mostly used for classes, therefore DGS staff would need to add a trimester information first, then proceed to add a class information for that added trimester.

6.8.1 Trimester information management

This functionality aims to let the DGS staff to manage the trimester related information, which the actions inclusive of add new trimester information, view the added trimester and edit or delete an existing trimester.

6.8.1.1 Add Trimester

This functionality could be traced in the "Add Trimester" under the Class Scheduling tab that hides in the sidebar. Once entering this page, the DGS staff could see an input form that requires to fill in the starting and ending date for the trimester, the study level for the trimester and the trimester count, since there are three trimesters in UTAR calendar.



ADD TRIMESTER

Semester Start Date
dd/mm/yyyy

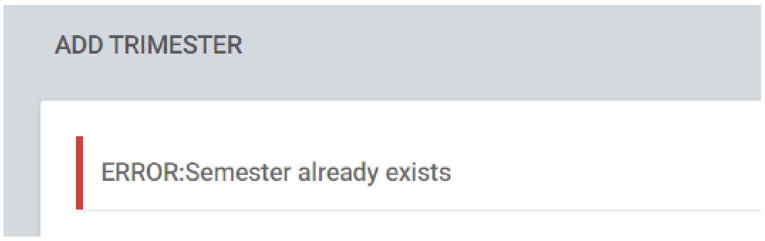
Semester End Date
dd/mm/yyyy

Type
--Select--

Trimester
--Select--

Figure 6.117: Input form for Add Trimester page

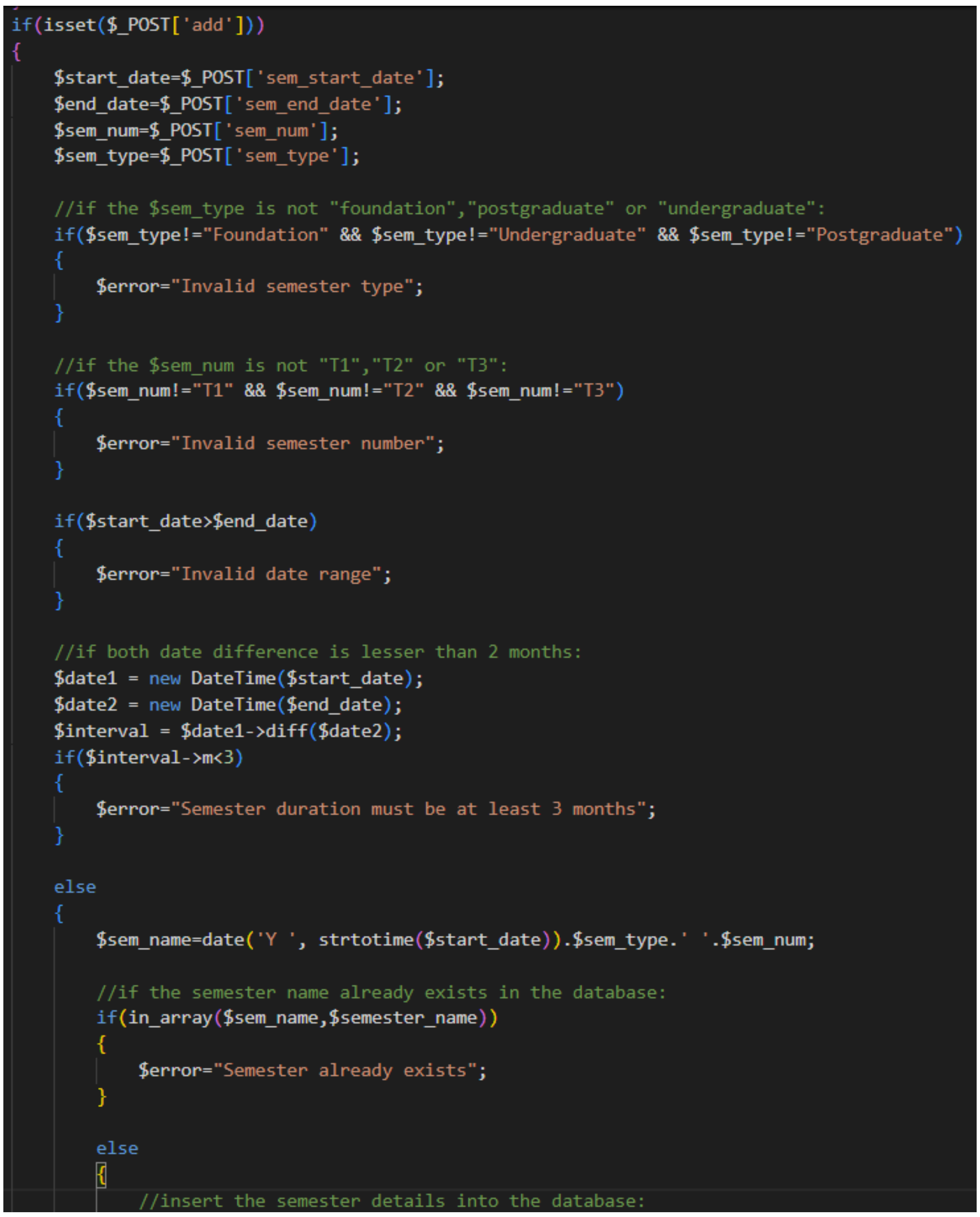
Input data validations had been done in this page, which could be summarized from the following table. Storing in the database table named ‘semester’, each trimester record added is a unique combination of trimester name, and the date values. Therefore, there should no be duplicate records of trimester exist in the database. Else, the DGS staff would be reminded with a message of such trimester has been exist. Input data validations would also remind the DGS staff regarding the particulars to be aware of.



ADD TRIMESTER

ERROR:Semester already exists

Figure 6.118: Error message if DGS staff has entered a duplicated trimester record



```php
if(isset($_POST['add']))
{
    $start_date=$_POST['sem_start_date'];
    $end_date=$_POST['sem_end_date'];
    $sem_num=$_POST['sem_num'];
    $sem_type=$_POST['sem_type'];

    //if the $sem_type is not "foundation","postgraduate" or "undergraduate":
    if($sem_type!="Foundation" && $sem_type!="Undergraduate" && $sem_type!="Postgraduate")
    {
        $error="Invalid semester type";
    }

    //if the $sem_num is not "T1","T2" or "T3":
    if($sem_num!="T1" && $sem_num!="T2" && $sem_num!="T3")
    {
        $error="Invalid semester number";
    }

    if($start_date>$end_date)
    {
        $error="Invalid date range";
    }

    //if both date difference is lesser than 2 months:
    $date1 = new DateTime($start_date);
    $date2 = new DateTime($end_date);
    $interval = $date1->diff($date2);
    if($interval->m<3)
    {
        $error="Semester duration must be at least 3 months";
    }

    else
    {
        $sem_name=date('Y ', strtotime($start_date)).$sem_type.' '.$sem_num;

        //if the semester name already exists in the database:
        if(in_array($sem_name,$semester_name))
        {
            $error="Semester already exists";
        }

        else
        {
            //insert the semester details into the database:
```

Figure 6.119: Technical implementation of validating input data when adding a trimester

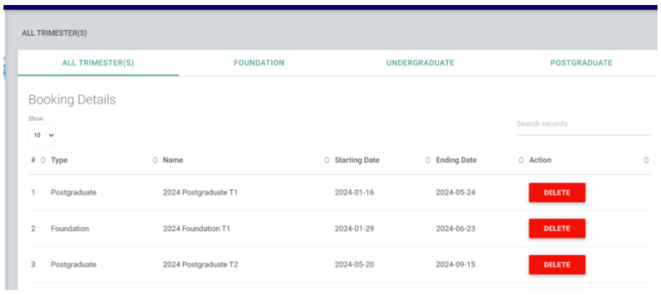
Table 6.5: Input data validations on Adding Trimester

Aspect	**Validation Scenario**	**Effect**
Date	Starting date is later than ending date	- Error message of “Invalid date range” appears. - Trimester information not added.

	Duration between both date values are lesser than 3 months	- Error message of “Trimester duration must be at least 3 months” appears. - Trimester information not added.
Semester count	Semester count number is not chosen	- Error message of “Invalid trimester number” appears. - Trimester information not added.
Level of study	Level of study is not chosen	- Error message of “Invalid trimester type” appears. - Trimester information not added.
Duplicated record	Adding the information which it already exists in the database	- Error message of “Trimester already exists” appears. - Trimester information not added.

6.8.1.2 View Trimester

With the trimester information added in section 6.8.1.1, the DGS staff could view the added trimester information in the “View Trimester” page that hides under the Class Scheduling in the sidebar.



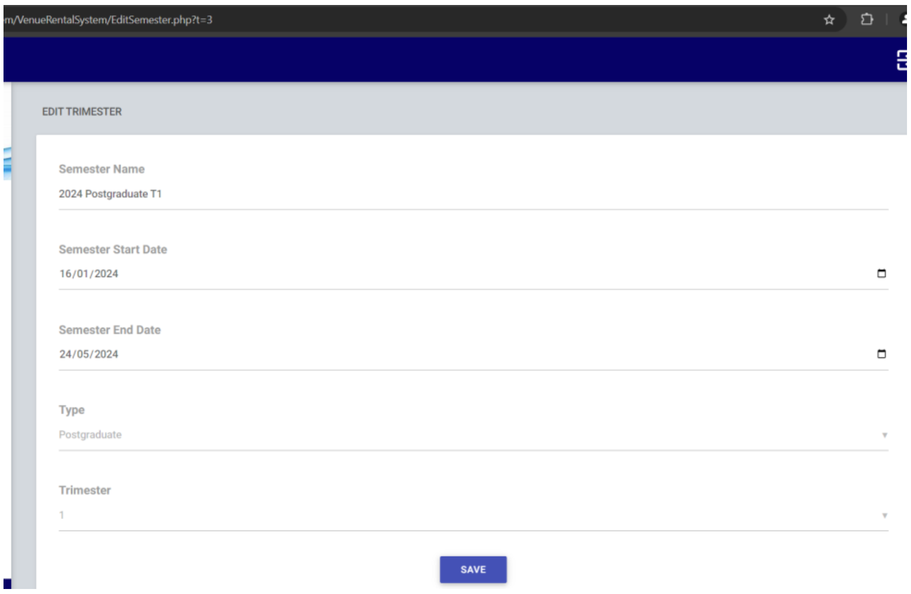
#	Type	Name	Starting Date	Ending Date	Action
1	Postgraduate	2024 Postgraduate T1	2024-01-16	2024-05-24	DELETE
2	Foundation	2024 Foundation T1	2024-01-29	2024-06-23	DELETE
3	Postgraduate	2024 Postgraduate T2	2024-05-20	2024-09-15	DELETE

Figure 6.120: View Trimester interface

The trimesters are being listed in the Data Table, which were categorized into tabs according the study level (foundation, undergraduate and postgraduate). The DGS staff could also utilize the search box to search for any trimesters, other than just scrolling from pages to pages. The data table lists the study level, the trimester name and the starting date and ending date values.

6.8.1.3 Edit & Delete Trimester

To edit a trimester information, the DGS staff could click on a trimester row to redirect to the editing page. Refer to the following figure to edit the trimester information.


m/VenueRentalSystem/EditSemester.php?t=3

EDIT TRIMESTER

Semester Name
2024 Postgraduate T1

Semester Start Date
16/01/2024

Semester End Date
24/05/2024

Type
Postgraduate

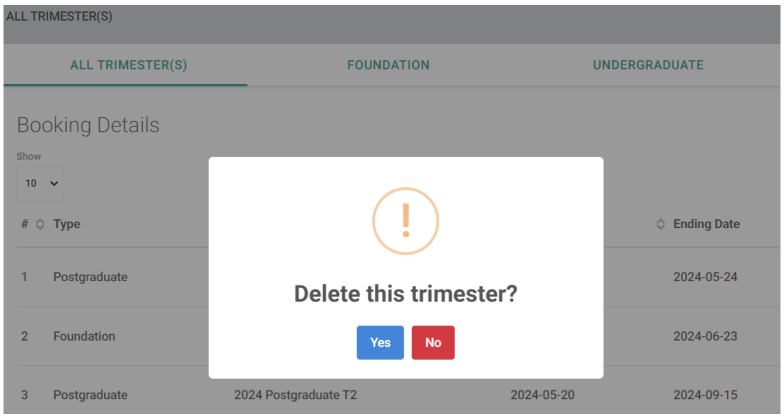
Trimester
1

SAVE

Figure 6.121: Edit Trimester interface

The DGS staff could only edit the starting date and ending date for a trimester. The date validation in the Add Trimester also works in this page as well. If the amendments are successful, then the message of “Trimester updated successfully” should be shown to the DGS staff after clicking the “Save” button.

To delete, the DGS staff has to click the delete button in the View Trimester page. Working like the booking request rejection function, the button in the row carries the identifier value for the exact row of trimester. Using JavaScript, once the button is clicked, then the SweetAlert pop-up will show up, confirming the deletion action. If the action is confirmed, it works with the external PHP script via AJAX to interact with the database to remove the exact trimester.


ALL TRIMESTER(S)

ALL TRIMESTER(S) | FOUNDATION | UNDERGRADUATE

Booking Details

Show 10

#	Type			Ending Date
1	Postgraduate			2024-05-24
2	Foundation			2024-06-23
3	Postgraduate	2024 Postgraduate T2	2024-05-20	2024-09-15

Delete this trimester?

Yes No

Figure 6.122: Delete trimester pop-up

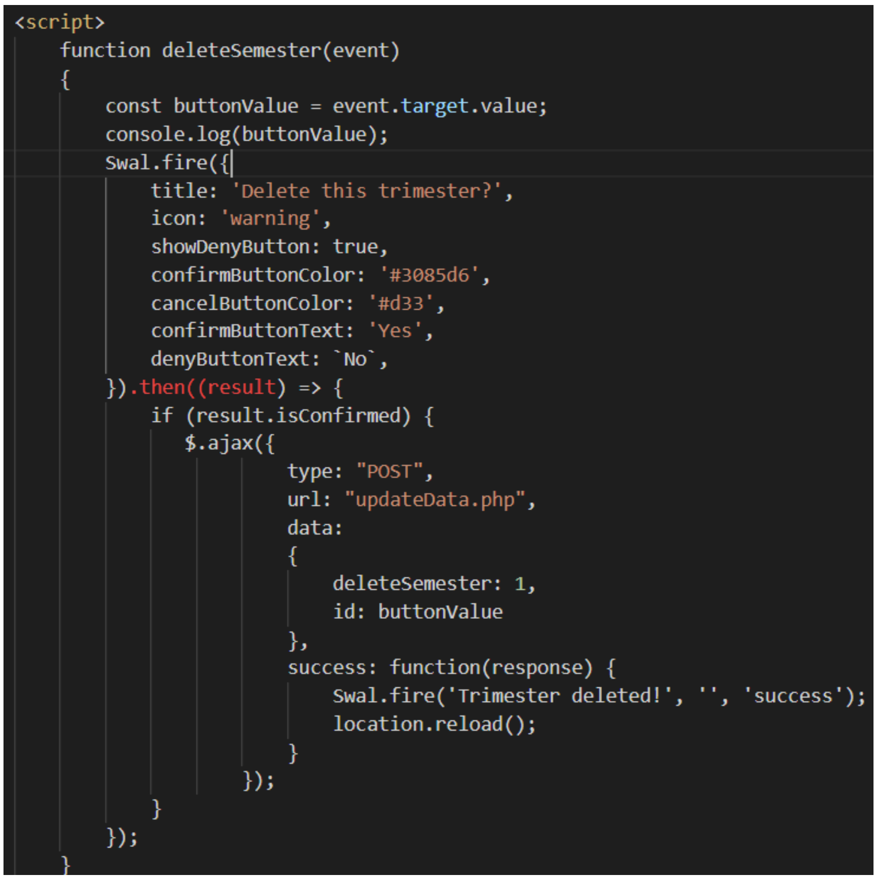

```
<script>
    function deleteSemester(event)
    {
        const buttonValue = event.target.value;
        console.log(buttonValue);
        Swal.fire({
            title: 'Delete this trimester?',
            icon: 'warning',
            showDenyButton: true,
            confirmButtonColor: '#3085d6',
            cancelButtonColor: '#d33',
            confirmButtonText: 'Yes',
            denyButtonText: `No`,
        }).then((result) => {
            if (result.isConfirmed) {
               $.ajax({
                            type: "POST",
                            url: "updateData.php",
                            data:
                            {
                                deleteSemester: 1,
                                id: buttonValue
                            },
                            success: function(response) {
                                Swal.fire('Trimester deleted!', '', 'success');
                                location.reload();
                            }
                        });
            }
        });
    }
```

Figure 6.123: Delete trimester JavaScript function for clicking delete button

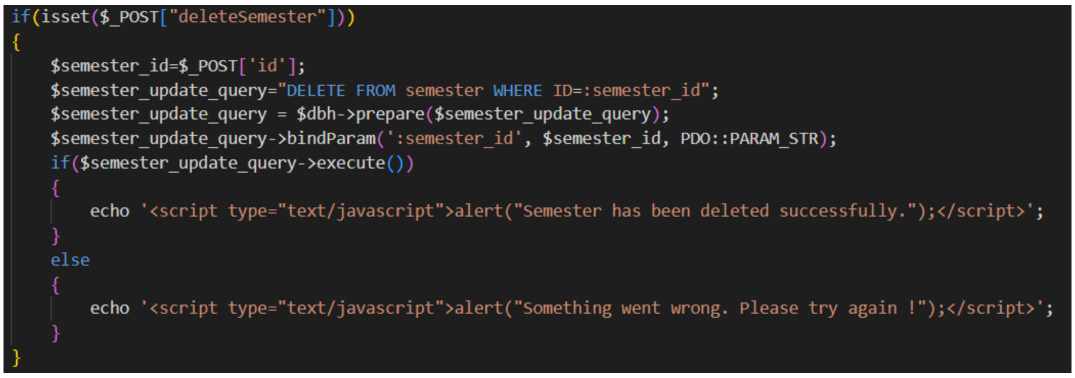

```
if(isset($_POST["deleteSemester"]))
{
    $semester_id=$_POST['id'];
    $semester_update_query="DELETE FROM semester WHERE ID=:semester_id";
    $semester_update_query = $dbh->prepare($semester_update_query);
    $semester_update_query->bindParam(':semester_id', $semester_id, PDO::PARAM_STR);
    if($semester_update_query->execute())
    {
        echo '<script type="text/javascript">alert("Semester has been deleted successfully.");</script>';
    }
    else
    {
        echo '<script type="text/javascript">alert("Something went wrong. Please try again !");</script>';
    }
}
```

Figure 6.124: Delete trimester PHP function

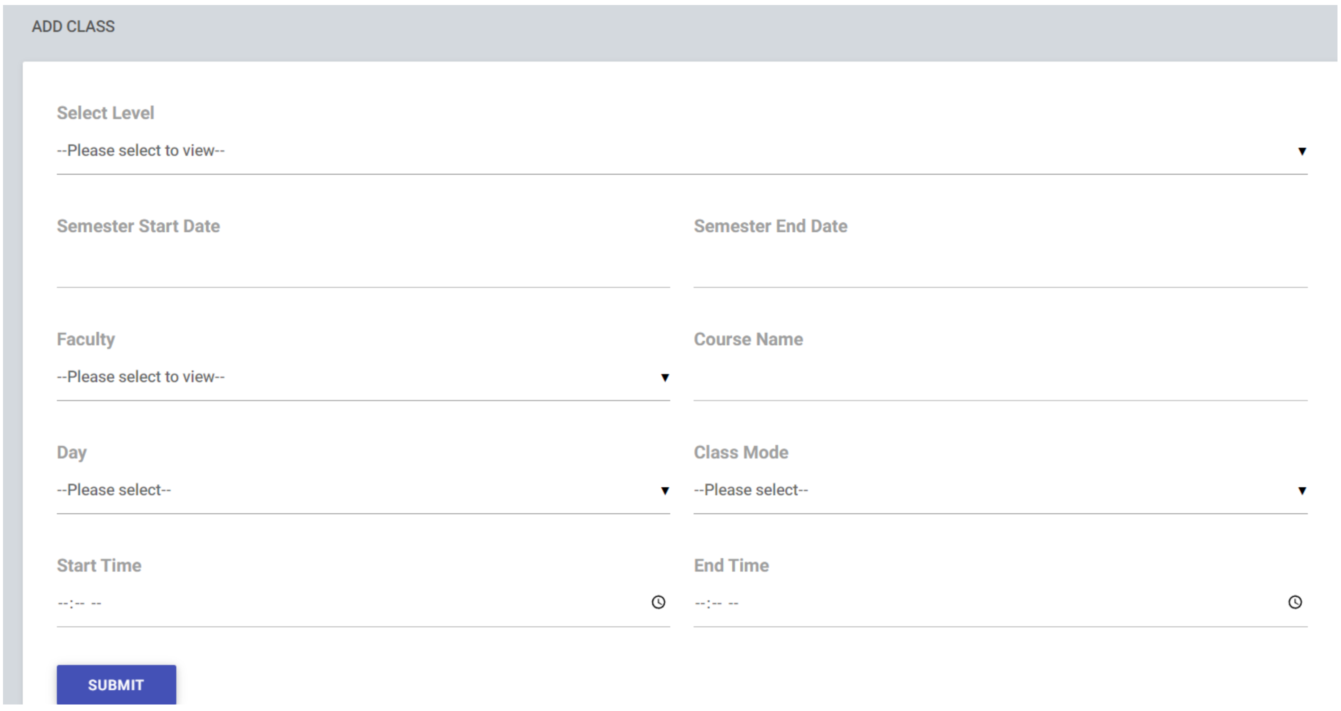
6.8.2 Class information management

This functionality aims to let the DGS staff to manage the class related information, which the actions inclusive of add new class information, view the added class and edit or delete an existing class information.

6.8.2.1 Add Class

This functionality could be traced in the "Add Class" under the Class Scheduling tab that hides in the sidebar. Once entering this page, the DGS staff could see an input form that needs to select the study level first. Based on the study level chosen, then equivalent trimesters would be fetched for the users to choose.

The trimester chosen finalizes the date range, which classes are meant to be held during that date range. Next, the DGS staff needs to choose the right faculty and the course name from the dropdown list. The DGS staff might proceed to choose the weekday to hold the class. On the class mode section, the DGS staff might need to choose if the class is going to be held weekly, or in even week and odd week. The class mode chosen will decide the class dates along with the starting and ending time values. Lastly, the DGS staff would need to choose the venue/classroom as demonstrated in the Add Booking page.


ADD CLASS

Select Level
--Please select to view--

Semester Start Date | Semester End Date

Faculty
--Please select to view--

Course Name

Day
--Please select--

Class Mode
--Please select--

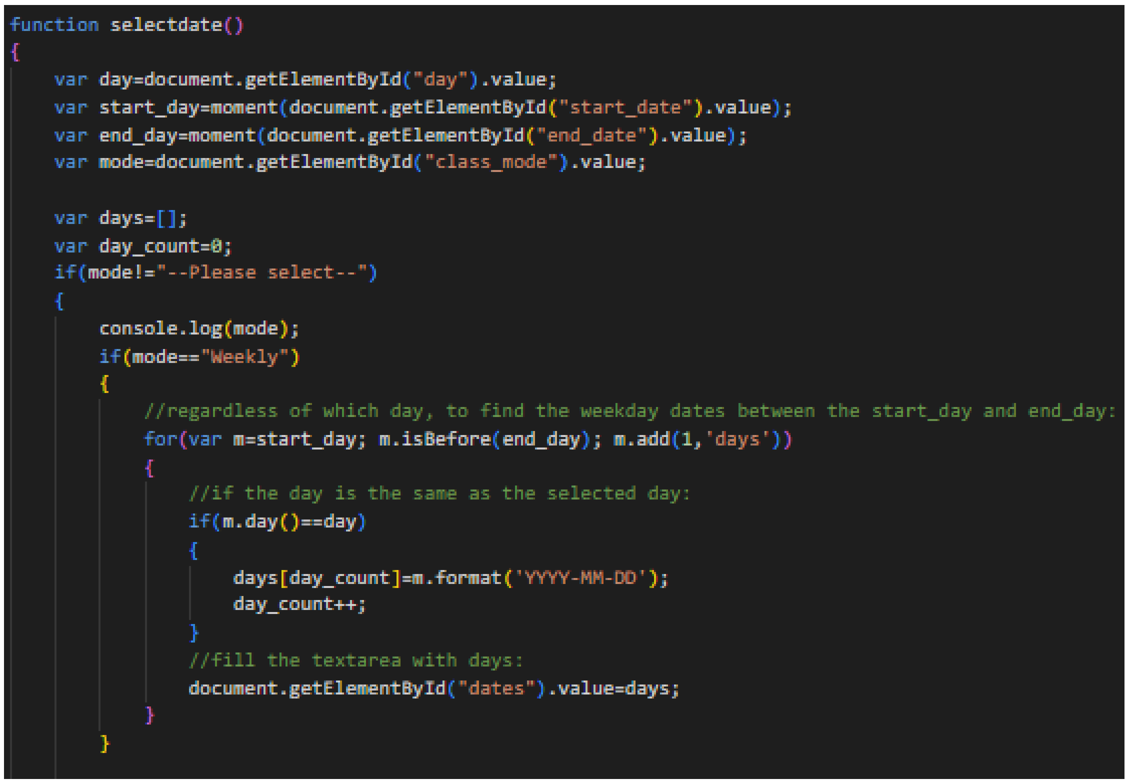
Start Time
--:-- --

End Time
--:-- --

SUBMIT

Figure 6.125: Add Class interface

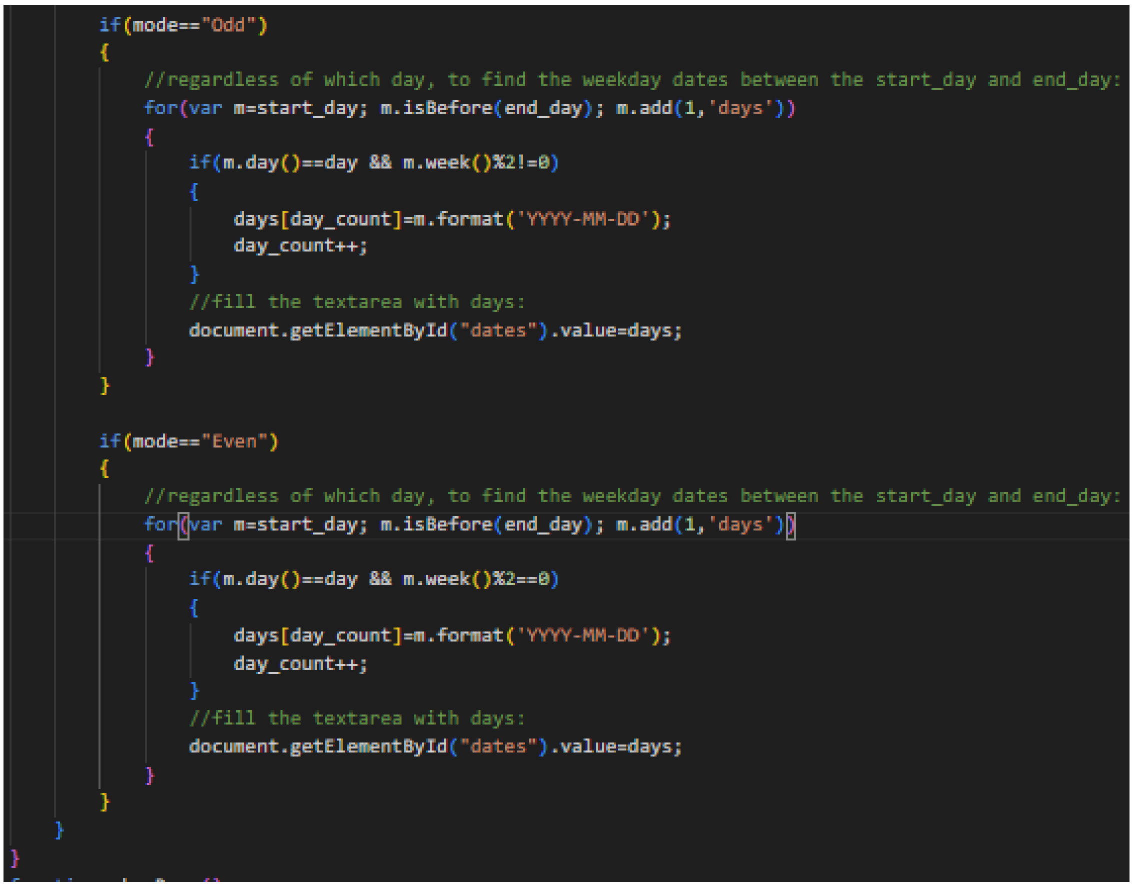
Technically, JavaScript is constantly ready to read the selection changes for starting and ending date values, to filter the date values for classes. First, it sets a loop between the trimester's starting date and ending date values, incrementing one day for each iteration. Based on the weekday chosen, it sets the weekday's dates into a date array if the iterating date's weekday value matches the weekday chosen. It sets one of the invisible sections in the form with the dates filtered, so that the venue/classroom is booked for those dates.



```
function selectdate()
{
    var day=document.getElementById("day").value;
    var start_day=moment(document.getElementById("start_date").value);
    var end_day=moment(document.getElementById("end_date").value);
    var mode=document.getElementById("class_mode").value;

    var days=[];
    var day_count=0;
    if(mode!="--Please select--")
    {
        console.log(mode);
        if(mode=="Weekly")
        {
            //regardless of which day, to find the weekday dates between the start_day and end_day:
            for(var m=start_day; m.isBefore(end_day); m.add(1,'days'))
            {
                //if the day is the same as the selected day:
                if(m.day()==day)
                {
                    days[day_count]=m.format('YYYY-MM-DD');
                    day_count++;
                }
                //fill the textarea with days:
                document.getElementById("dates").value=days;
            }
        }
```

Figure 6.126: Technical implementation of filtering dates for classes

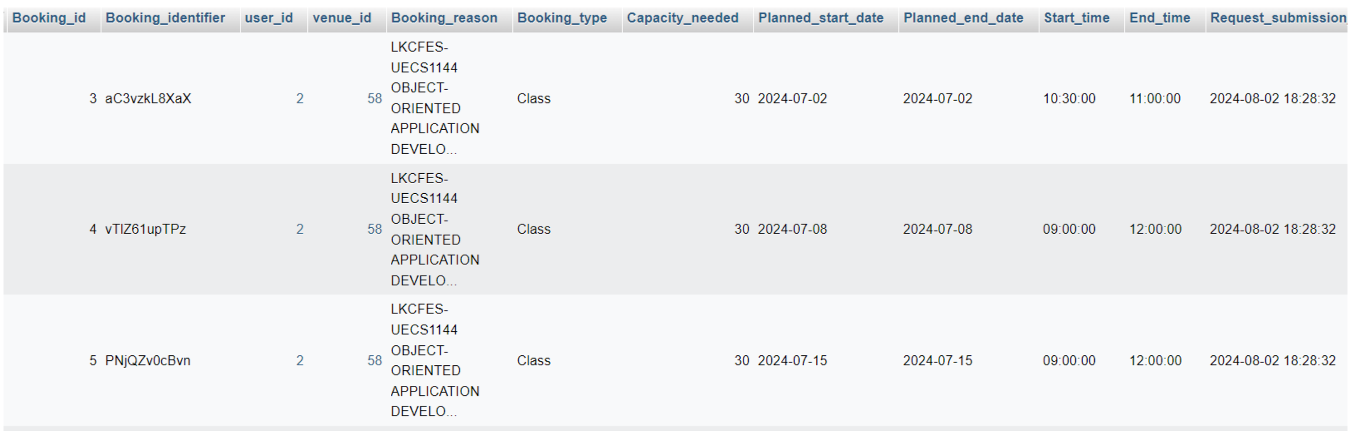
For odd week classes, other than making sure the weekday's date is matching, it also counts the week number by modulo operating it by 2. If there is remainder, then it is considered as an odd-week date. In contrast, even weeks do not have any remaining values even though the week count is modulo operated by 2. This is how the odd-week and even-week dates filtering works. Do refer to Figure 6.127 to know this operation.



```
        if(mode=="Odd")
        {
            //regardless of which day, to find the weekday dates between the start_day and end_day:
            for(var m=start_day; m.isBefore(end_day); m.add(1,'days'))
            {
                if(m.day()==day && m.week()%2!=0)
                {
                    days[day_count]=m.format('YYYY-MM-DD');
                    day_count++;
                }
                //fill the textarea with days:
                document.getElementById("dates").value=days;
            }
        }

        if(mode=="Even")
        {
            //regardless of which day, to find the weekday dates between the start_day and end_day:
            for(var m=start_day; m.isBefore(end_day); m.add(1,'days'))
            {
                if(m.day()==day && m.week()%2==0)
                {
                    days[day_count]=m.format('YYYY-MM-DD');
                    day_count++;
                }
                //fill the textarea with days:
                document.getElementById("dates").value=days;
            }
        }
    }
}
```

Figure 6.127: Technical implementation of filtering dates for odd-week and even-week

The trimester names, course names and the venues/classrooms are being fetched from database through JavaScript via AJAX, since it needs to read the selection choice on the study level and the faculty respectively. Since it is related to adding new venue/classroom usage, then the function of generating random strings described in section 6.7.1 is implemented in this page as well.

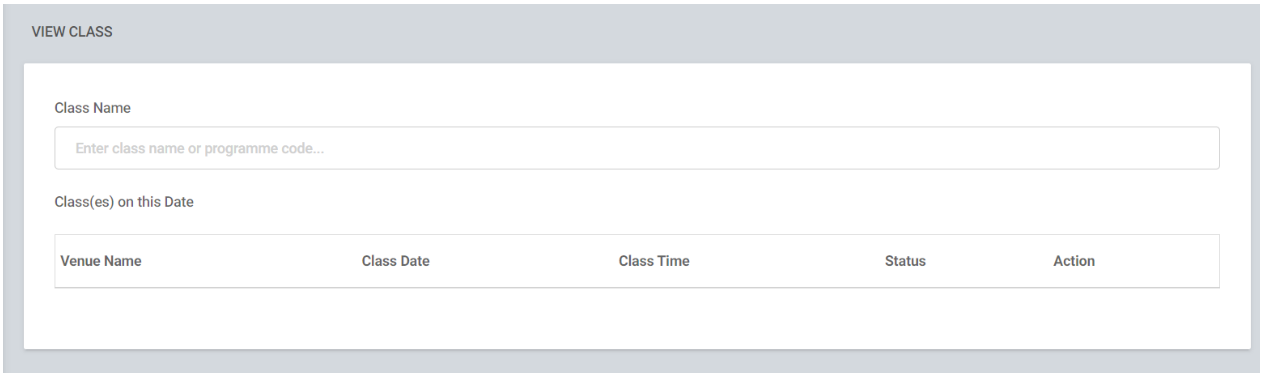


Booking_id	Booking_identifier	user_id	venue_id	Booking_reason	Booking_type	Capacity_needed	Planned_start_date	Planned_end_date	Start_time	End_time	Request_submission
3	aC3vzkL8XaX	2	58	LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELO...	Class	30	2024-07-02	2024-07-02	10:30:00	11:00:00	2024-08-02 18:28:32
4	vTIZ61upTPz	2	58	LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELO...	Class	30	2024-07-08	2024-07-08	09:00:00	12:00:00	2024-08-02 18:28:32
5	PNjQZv0cBvn	2	58	LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELO...	Class	30	2024-07-15	2024-07-15	09:00:00	12:00:00	2024-08-02 18:28:32

Figure 6.128: Overview of venue booking’s database for classes added

6.8.2.2 View Class

With the class information added in section 6.8.2.1, the DGS staff could view the added class information in the “View Scheduled Class” page that hides under the Class Scheduling in the sidebar.



VIEW CLASS

Class Name

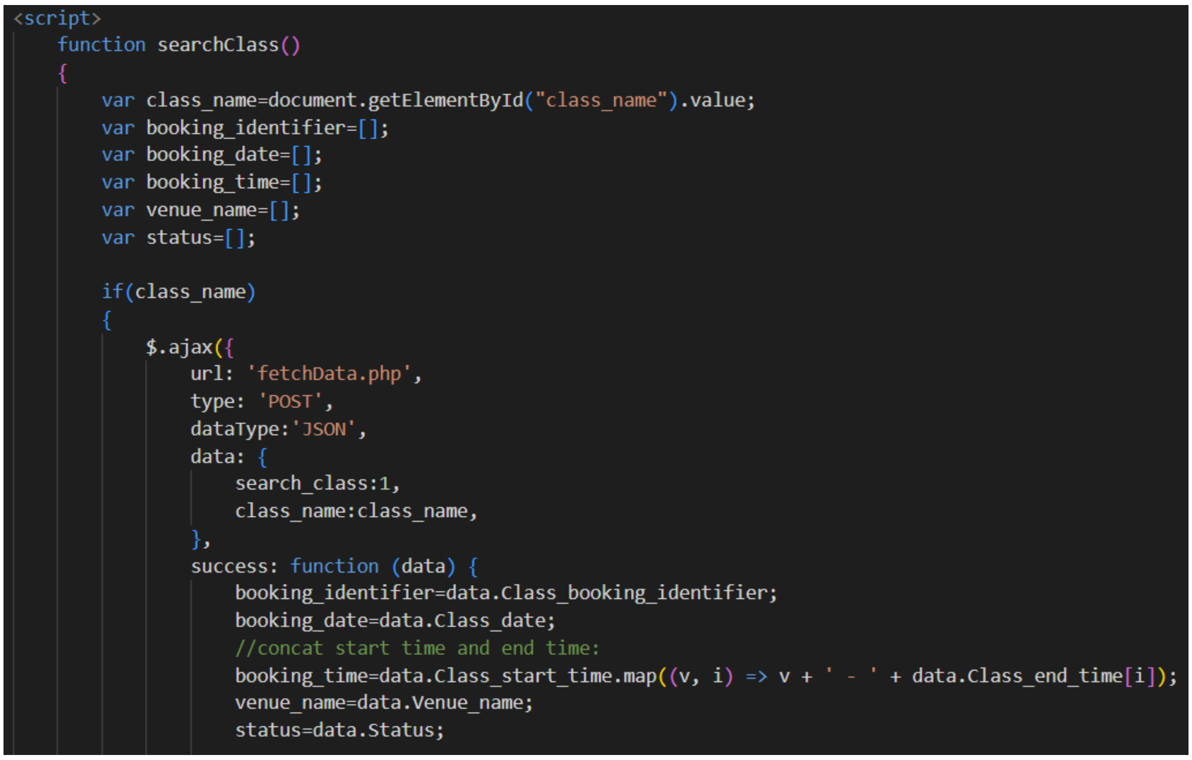
Enter class name or programme code...

Class(es) on this Date

Venue Name	Class Date	Class Time	Status	Action

Figure 6.129: View Class interface

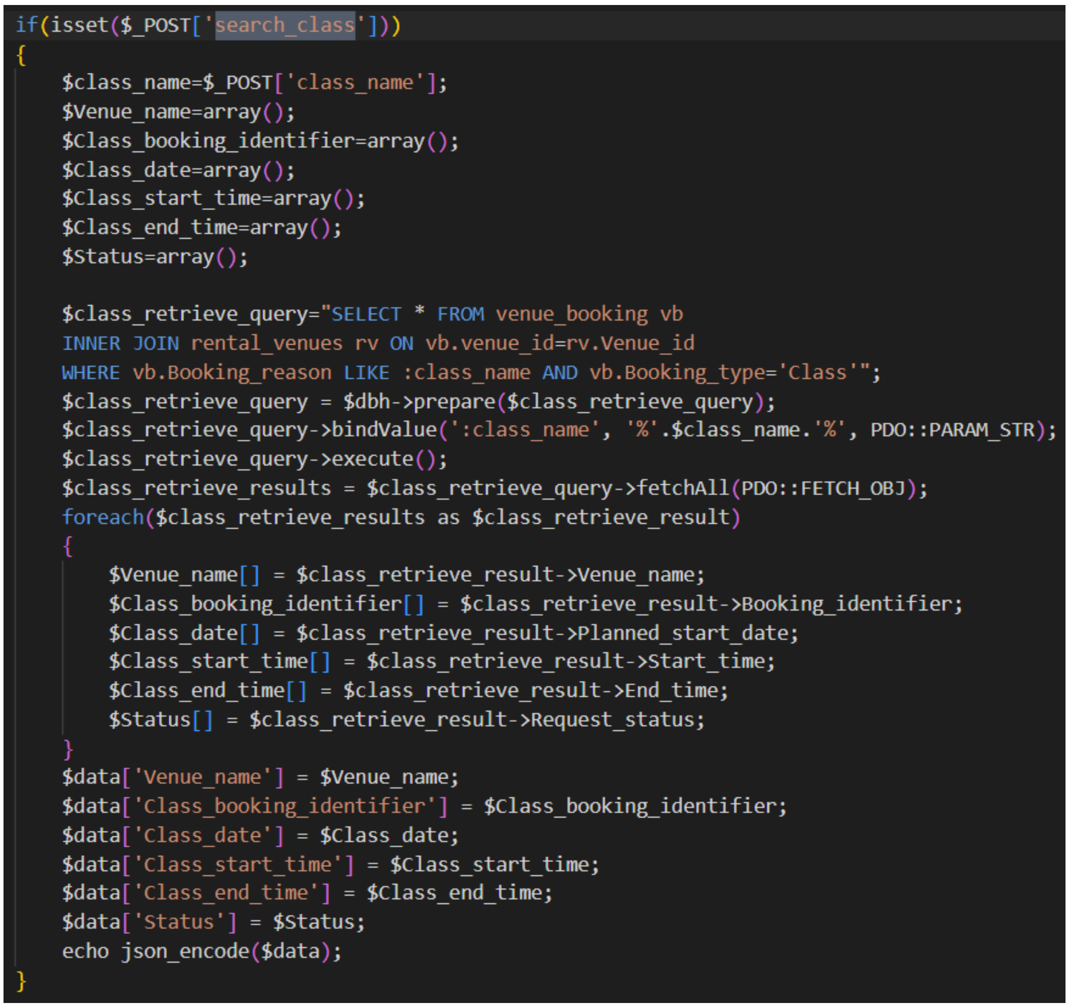
Since there could be many classes added, therefore the DGS staff is needed to type the course name or programme code so that relevant class information would be displayed on the table below. Initially, the table shows nothing because there is no class information input yet. To constantly reads the changes on the input column, *oninput()* function needs to be implemented on the input box, working with a function declared in JavaScript.



```
<script>
    function searchClass()
    {
        var class_name=document.getElementById("class_name").value;
        var booking_identifier=[];
        var booking_date=[];
        var booking_time=[];
        var venue_name=[];
        var status=[];

        if(class_name)
        {
            $.ajax({
                url: 'fetchData.php',
                type: 'POST',
                dataType:'JSON',
                data: {
                    search_class:1,
                    class_name:class_name,
                },
                success: function (data) {
                    booking_identifier=data.Class_booking_identifier;
                    booking_date=data.Class_date;
                    //concat start time and end time:
                    booking_time=data.Class_start_time.map((v, i) => v + ' - ' + data.Class_end_time[i]);
                    venue_name=data.Venue_name;
                    status=data.Status;
```

Figure 6.130: Technical implementation of reading input column changes

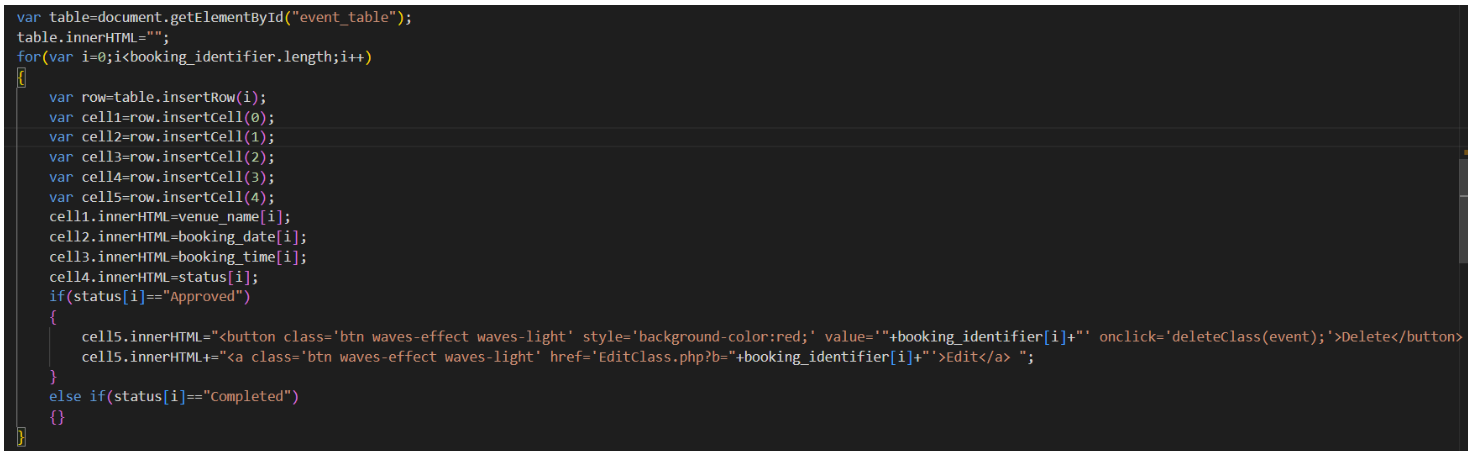
As long as there are characters input on the input column, then it works with AJAX to exchange data asynchronously with the external PHP file, that is the fetchData.php, projecting the characters input. In the database interaction, the SQL operation of LIKE was used in this case, so that any database records that its properties are alike with the keyword will be fetched. Values for venue's name, booking request's randomly generated identifier, class date and time values and booking request's status will be the output for the database interaction. Do refer to figure 6.131 to learn about the operation.



```php
if(isset($_POST['search_class']))
{
    $class_name=$_POST['class_name'];
    $Venue_name=array();
    $Class_booking_identifier=array();
    $Class_date=array();
    $Class_start_time=array();
    $Class_end_time=array();
    $Status=array();

    $class_retrieve_query="SELECT * FROM venue_booking vb
    INNER JOIN rental_venues rv ON vb.venue_id=rv.Venue_id
    WHERE vb.Booking_reason LIKE :class_name AND vb.Booking_type='Class'";
    $class_retrieve_query = $dbh->prepare($class_retrieve_query);
    $class_retrieve_query->bindValue(':class_name', '%'.$class_name.'%', PDO::PARAM_STR);
    $class_retrieve_query->execute();
    $class_retrieve_results = $class_retrieve_query->fetchAll(PDO::FETCH_OBJ);
    foreach($class_retrieve_results as $class_retrieve_result)
    {
        $Venue_name[] = $class_retrieve_result->Venue_name;
        $Class_booking_identifier[] = $class_retrieve_result->Booking_identifier;
        $Class_date[] = $class_retrieve_result->Planned_start_date;
        $Class_start_time[] = $class_retrieve_result->Start_time;
        $Class_end_time[] = $class_retrieve_result->End_time;
        $Status[] = $class_retrieve_result->Request_status;
    }
    $data['Venue_name'] = $Venue_name;
    $data['Class_booking_identifier'] = $Class_booking_identifier;
    $data['Class_date'] = $Class_date;
    $data['Class_start_time'] = $Class_start_time;
    $data['Class_end_time'] = $Class_end_time;
    $data['Status'] = $Status;
    echo json_encode($data);
}
```

Figure 6.131: Technical implementation of database interaction

The output will be projected on the table underneath the input column, through JavaScript DOM operations. Along the same row will be showing the delete and edit buttons that holds the identifier value for the venue/classroom booking request. Refer to figure 6.132 for the operations.

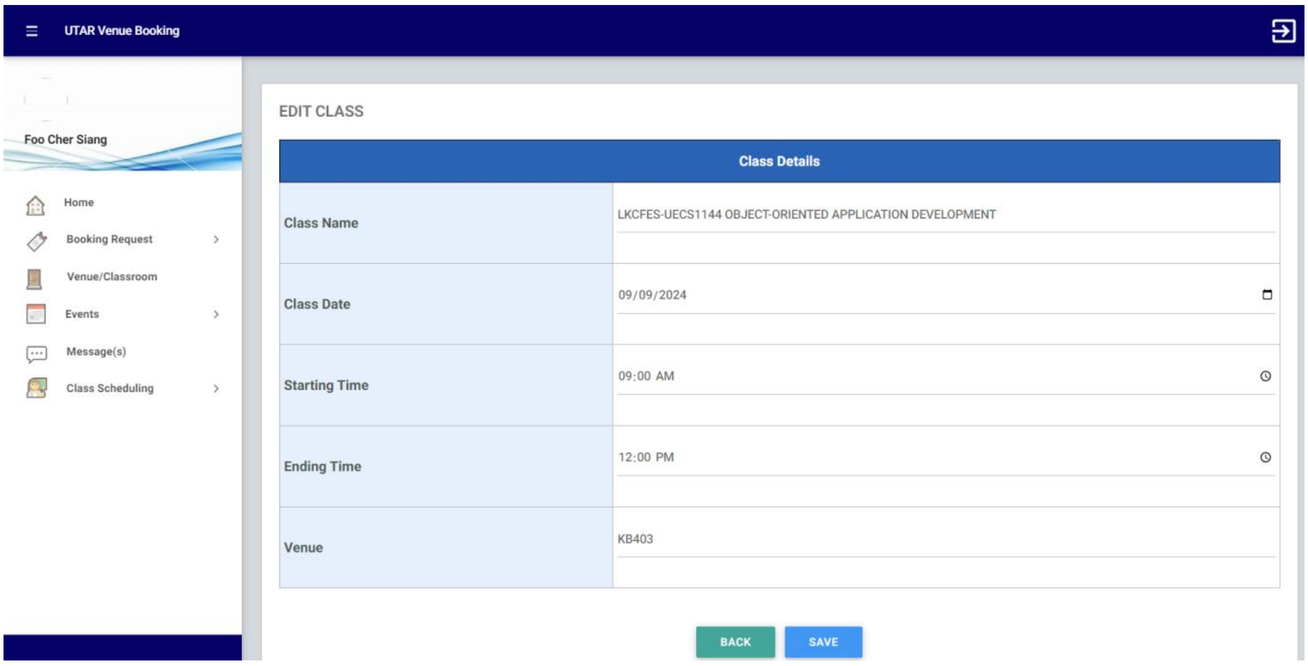


```
var table=document.getElementById("event_table");
table.innerHTML="";
for(var i=0;i<booking_identifier.length;i++)
{
    var row=table.insertRow(i);
    var cell1=row.insertCell(0);
    var cell2=row.insertCell(1);
    var cell3=row.insertCell(2);
    var cell4=row.insertCell(3);
    var cell5=row.insertCell(4);
    cell1.innerHTML=venue_name[i];
    cell2.innerHTML=booking_date[i];
    cell3.innerHTML=booking_time[i];
    cell4.innerHTML=status[i];
    if(status[i]=="Approved")
    {
        cell5.innerHTML="<button class='btn waves-effect waves-light' style='background-color:red;' value='"+booking_identifier[i]+"' onclick='deleteClass(event);'>Delete</button>
        cell5.innerHTML+="<a class='btn waves-effect waves-light' href='EditClass.php?b="+booking_identifier[i]+"'>Edit</a> ";
    }
    else if(status[i]=="Completed")
    {}
}
```

Figure 6.132: Technical implementation of projecting class information output

6.8.2.3 Edit & Delete Class

To edit a class (venue/class request) information, the DGS staff could click on the edit button. Accessing validation is performed on this page, since it works based on the venue/classroom request booking identifier value. The figure below shows the interface of editing the class information.

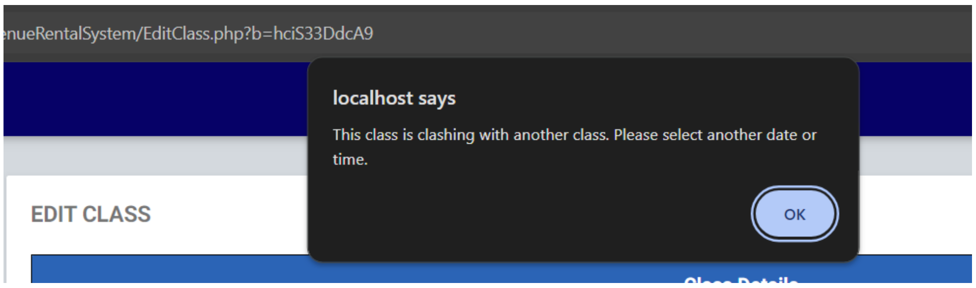


EDIT CLASS

Class Details	
Class Name	LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELOPMENT
Class Date	09/09/2024
Starting Time	09:00 AM
Ending Time	12:00 PM
Venue	KB403

Figure 6.133: Editing class information interface

The DGS staff could only edit the class date and the timing values. To prevent venue/classroom usage conflicts, the editing values are compared with the existing venue/classroom records in the database. This could be done by having the *onchange()* function implemented on the time input columns. Again, it checks through AJAX with the external PHP page to check if the venue/classroom is being used with the edited values or not. If there truly exist a record, then the DGS staff is immediately alerted with a message. The time values are then reset again. All these editing values do not need to be saved to undergo the checking. Once the editing has been done without problems, then the class information is considered updated successfully.



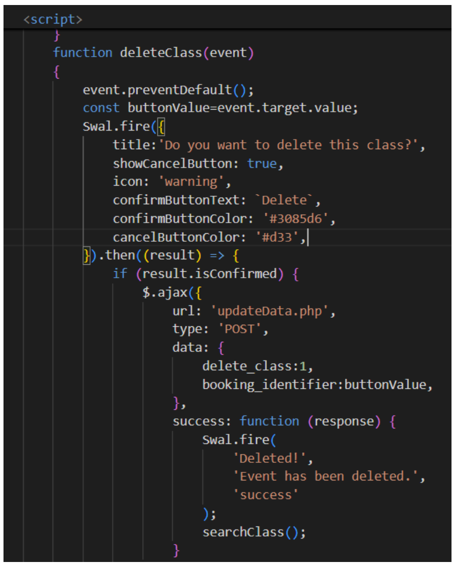
localhost says

This class is clashing with another class. Please select another date or time.

Figure 6.134: Alert message if editing info conflicts with existing record

Timing inputs has been validated as well, as long as the starting time does not exceed the ending time value, like what was described in the previous sections.

The deletion of class works the same with the trimester information. Each delete button holds the unique identifier value for the class information, triggering it will show the SweetAlert pop-up, confirming with the DGS staff on the actions. The confirmation will remove the row of class information, and to delete the venue/classroom usage request in the database.



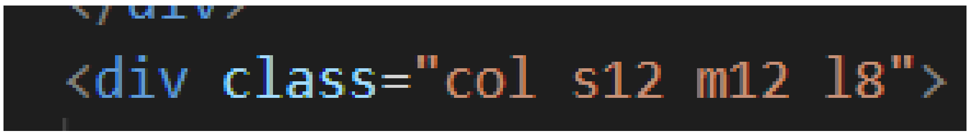
```
<script>
    }
    function deleteClass(event)
    {
        event.preventDefault();
        const buttonValue=event.target.value;
        Swal.fire({
            title:'Do you want to delete this class?',
            showCancelButton: true,
            icon: 'warning',
            confirmButtonText: `Delete`,
            confirmButtonColor: '#3085d6',
            cancelButtonColor: '#d33',
        }).then((result) => {
            if (result.isConfirmed) {
                $.ajax({
                    url: 'updateData.php',
                    type: 'POST',
                    data: {
                        delete_class:1,
                        booking_identifier:buttonValue,
                    },
                    success: function (response) {
                        Swal.fire(
                            'Deleted!',
                            'Event has been deleted.',
                            'success'
                        );
                        searchClass();
                    }
```

Figure 6.135: JavaScript function for deleting class button

6.9 Mobile interface

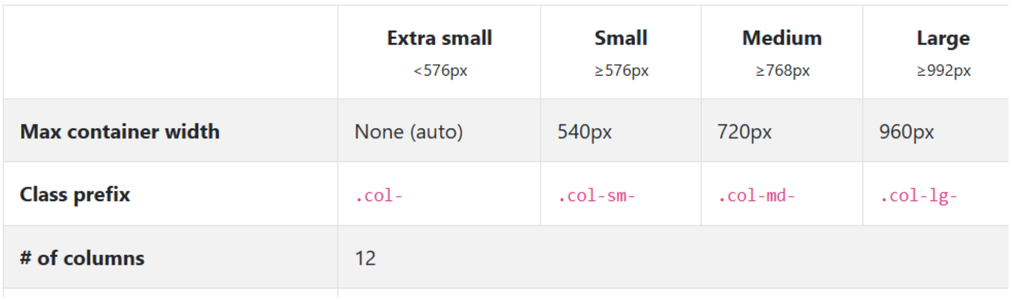
Matching the 1.2 non-functional requirement specified in the Availability section, the entire system could be presented in the mobile interface. This could be achieved with Bootstrap class configurations for each HTML element.

According to the official documentation, the Bootstrap elements are working in the flexbox concept and the maximum size for it will be 12 (Bootstrap, -). As it matters with the screen sizing, the dimension size needs to be specifically declared at the class section. Taking View Class page as an example, since the entire screen is responsive according to the end-users' device, then the <div> which serves as the parent element should be declared with the equivalent size when it comes to working with different sizes of device. For 's' that means small devices, 'm' refers to medium-sized devices and lastly l will be representing large-sized devices. These three sizes got their dimensions listed as what presented in figure 6.136.



```
<div class="col s12 m12 l8">
```

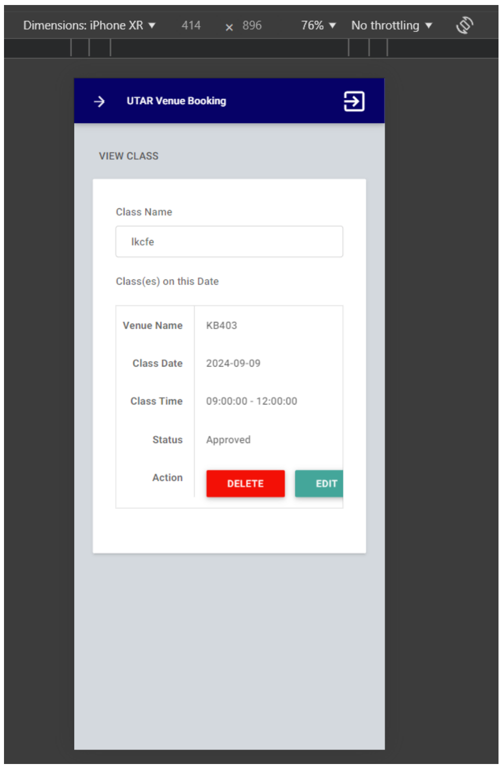
Figure 6.136: Technical implementation of sizes declaring



	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px
Max container width	None (auto)	540px	720px	960px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`
# of columns	12			

Figure 6.137: Different sizes of devices with their dimensions (Bootstrap, -).

For tables which tend to get out of shape due to restricted screen size, it needs to be declared with a tag 'display-responsive-table' so that the contents are still recognizable though the device size changes drastically. The following figure shows the comparison of tables showing in desktop-sized devices and mobile-sized devices.



Dimensions: iPhone XR ▾ 414 × 896 76% ▾ No throttling ▾

UTAR Venue Booking

VIEW CLASS

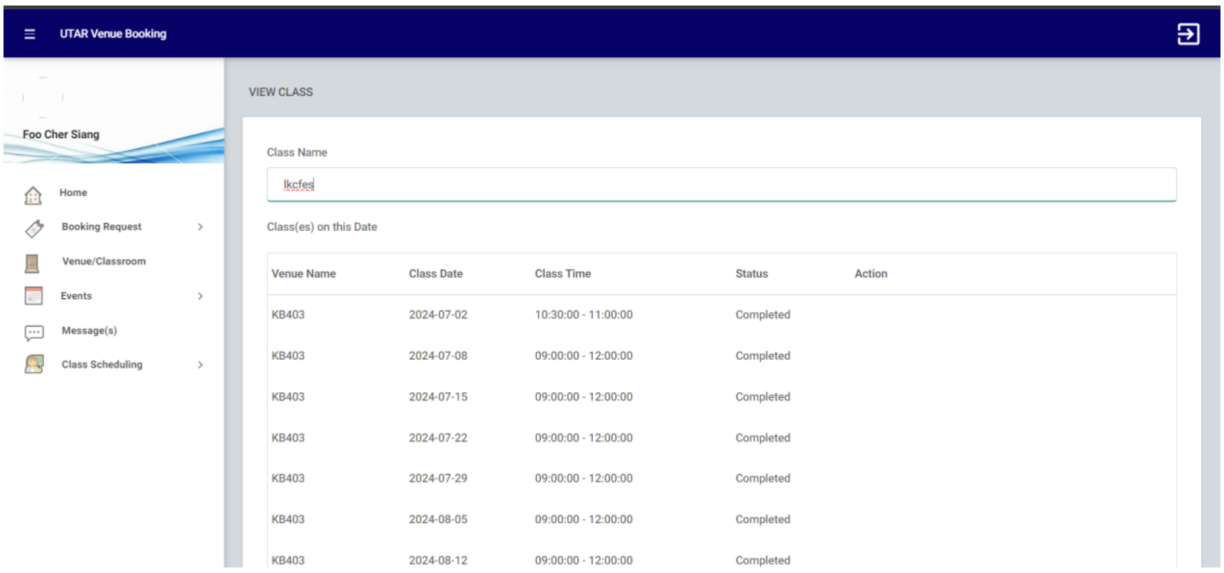
Class Name

lkcfe

Class(es) on this Date

Venue Name	KB403
Class Date	2024-09-09
Class Time	09:00:00 - 12:00:00
Status	Approved
Action	DELETE EDIT

Figure 6.138: View Class page demonstrating in iPhone XR size



VIEW CLASS

Class Name

lkcfes

Class(es) on this Date

Venue Name	Class Date	Class Time	Status	Action
KB403	2024-07-02	10:30:00 - 11:00:00	Completed	
KB403	2024-07-08	09:00:00 - 12:00:00	Completed	
KB403	2024-07-15	09:00:00 - 12:00:00	Completed	
KB403	2024-07-22	09:00:00 - 12:00:00	Completed	
KB403	2024-07-29	09:00:00 - 12:00:00	Completed	
KB403	2024-08-05	09:00:00 - 12:00:00	Completed	
KB403	2024-08-12	09:00:00 - 12:00:00	Completed	

Figure 6.139: View Class page demonstrating in desktop-sized devices

6.10 Summary

This chapter has demonstrated all system modules, covering all notable requirements. The developed system has strived to hit the functionalities in terms of core problems (venue/classroom usage) and communication aspects. Therefore, this system is ready for testing, which will be described in the next chapter.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

Testing phase aims of examining functionality and to detect presence of unexpected defects. Unit Testing, User Acceptance Test and Usability Testing are conducted for this project so that it verifies the workability of the upgraded UTAR Venue/Classroom Booking Web Portal.

In this chapter, the processes and the outcomes for conducting the testing will be discussed.

7.2 Test Execution

Unit Testing was executed once the system has marked an end on the development phase, followed by the User Acceptance Test and Usability Testing which these two testings were carried at the same time. Inclusive of Mr. Foo and his colleagues from DGS who is responsible of venue allocation, partly of the interviewees and the lecturers who had participated in the online survey during the requirements elicitation phase had been invited to test the system physically. The testing duration varies, however it took a week to conduct the last two types of testing. While some defects were discovered during the User Acceptance Test, immediate fixes were done to remedy. Minor refinements were done on the system as well, based on the suggestion from the testers which responded on spot.

7.3 Unit Testing

Unit Testing examines each component of the system developed with the project author planning the test cases, assessing the correctness of the function developed and to observe the flaws if the testing results are not matched as expected.

7.3.1 Unit Test Cases for General Cases

This subsection discusses the testing on the functionalities that every possible end-user could use, regardless of their identity as a student, lecturer, Head of

Department or other administrative role like staff from Department of General Services. The functions tested inclusive of login, profile management, add/cancel booking, events checking, venue checking and messaging.

Table 7.1: Unit Test Case for User Login

Test Case #	1	**Test Case Name**	Login		
Test Case Summary	To test the user could login				
Pre-Conditions	User possess a valid UTAR account, with selecting reCAPTCHA pictures.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Login with registered UTAR email address and valid password values	1. Enter registered UTAR email address 2. Enter password values 3. Check reCAPTCHA	Email address: dgsadmin@utar.edu.my Password attempted: 123	Login session established and relocated to dashboard.	Login session established and relocated to dashboard.	PASS

Login with non-UTAR email address and password values	1. Enter non-UTAR email address 2. Enter password values 3. Check reCAPTCHA	Email address: jiakeenwong@gmail.com Password:123	Alert is shown with message "Please use UTAR email"	Alert is shown with message "Please use UTAR email"	PASS
Login with non-registered UTAR email address	1. Enter anonymous UTAR email address 2. Enter password values 3. Check reCAPTCHA	Email address:test@utar.edu.my Password:123	Alert is shown with message "Invalid email".	Alert is shown with message "Invalid email".	PASS
Login with valid UTAR email address but wrong password values	1. Enter UTAR email address 2. Enter wrong password values	Email address:jiakeenwong@utar.com Password:ewedddw	Alert is shown with message "Invalid password".	Alert is shown with message "Invalid password".	PASS

	3. Check reCAPTCHA 4. Click Login button				
Login account without checking reCAPTCHA	1. Enter registered UTAR email address 2. Enter password 3. Tap Login button	Email address: dgsadmin@utar.edu.my Password attempted: 123	Alert is shown with message “Please check the captcha form!”.	Alert is shown with message “Please check the captcha form!”.	PASS
Login account with blanks on email or password column	1. Leave either email or password columns empty. 2. Click Login button	Email: Password:	Input reminder appears on the blank columns, with message “Please fill out this field.”	Input reminder appears on the blank columns, with message “Please fill out this field.”	PASS

Table 7.2: Unit Test Case for Reset Password

Test Case #	2	**Test Case Name**	Reset Password		
Test Case Summary	To test if user could reset password				
Pre-Conditions	User clicks the "Forgot password?" link before login to the page				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Enter invalid email address	1. User is in the Forget Password page where it requests the user to enter their registered email address.	Email address: jiakeenwong0413@yahoo.com	- Page reloads, with an error message displaying "Invalid details!". - No reset password email received.	- Page reloads, with an error message displaying "Invalid details!". - No reset password email received.	Pass

	2. User simply enters an invalid address. 3. User presses Submit button.				
Enter valid email address	1. User is in the Forget Password page. 2. User fills in their registered UTAR email address. 3. User submits.	Email address: jiakeenwong@gmail.com	- Page reloads, with a success message displaying “Please check your email to reset the password! - Reset password email received.	- Page reloads, with a success message displaying “Please check your email to reset the password! - Reset password email received.	Pass
Enter Reset Password page with valid link	1. User receives the reset password email. 2. User selects the directive hyperlink,	Identifier link: JmlFoOUg	-Reset password page displayed to the user.	-Reset password page displayed to the user.	Pass

	which was listed in the email. 3. User is directed to Reset Password page.				
Enter Reset Password page with invalid link	1. User receives the reset password email. 2. User selects the directive hyperlink, which was listed in the email. 3. User is directed to Reset Password page. 4. User edits the identifier link in the URL section.	Identifier link: JmlFoOUg3	-Showing alert message of “Invalid password reset link!”. - User is relocated to login interface.	-Showing alert message of “Invalid password reset link!”. - User is relocated to login interface.	Pass

Valid new password	1 User is in the reset password page. 2. User follows the reset hint in the page, entering a new password and the confirmation password. 3. User clicks the Change button.	New password: Testing@123 Confirmation password: Testing@123	- Page reloads, with message "Your password has successfully changed!" displayed. - User is redirected back to the login page.	- Page reloads, with message "Your password has successfully changed!" displayed. - User is redirected back to the login page.	Pass
Invalid new password	1 User is in the reset password page. 2. User omits the reset hint in the page, entering a new password and	New password: Testing@ Confirmation password: Testing@	- Page reloads, with message "Password must be at least 6 characters and must contain at least one lower case letter, one upper case	- Page reloads, with message "Password must be at least 6 characters and must contain at least one lower case letter, one upper case	Pass

	the confirmation password. 3. User clicks the Change button.		letter and one number!" displayed. - User stays on the page.	letter and one number!" displayed. - User stays on the page.	
New password not equivalent with confirmation password	1 User is in the reset password page. 2. User follows the reset hint in the page, entering a new password. 3. User simply types a string on the confirmation password column.	New password: Testing@123 Confirmation password: aadadwdw	- Page reloads, with message "New password is not match with confirm password!" displayed. - User stays on the page.	- Page reloads, with message "New password is not match with confirm password!" displayed. - User stays on the page.	Pass

	4. User clicks the Change button.				

Table 7.3: User Profile Unit Test Case

Test Case #	3	**Test Case Name**	User Profile Access		
Test Case Summary	To test user could access own profile page				
Pre-Conditions	User can be accessed to the UTAR Venue/Classroom Booking Web Portal.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Access profile page using own user ID	1. Successful login to UTAR Venue/Classroom Booking Web Portal	User ID: designated own user ID	Redirected to User Profile page with table showing user details.	Redirected to User Profile page with table showing user details.	PASS

	2. Select Profile logo on header bar				
Access profile page with other ID	1. Successful login to UTAR Venue/Classroom Booking Web Portal 2. Edit the user_id on the URL link to user Profile page	User ID: random user ID, not same as the logged in user	Alert with message “Invalid access” is shown, then user is redirected back to dashboard page.	Alert with message “Invalid access” is shown, then user is redirected back to dashboard page.	PASS

Table 7.4: Unit Test Case for User Profile Edit

Test Case #	4	**Test Case Name**	User Profile Details Editing
Test Case Summary	To test if user could edit their profile details		
Pre-Conditions	User can be accessed to the UTAR Venue/Classroom Booking Web Portal and enters their Profile page.		
Prepared & Executed By	Wong Jia Keen		

Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Edit contact number	1. Successful login to UTAR Venue/Classroom Booking Web Portal 2. Successful access to profile page 3. Select "Edit" button	Contact number: 60108415946	- Page reloads with header message "SUCCESS : Profile updated successfully!" is shown. - Contact number for the profile is updated.	- Page reloads with header message "SUCCESS : Profile updated successfully!" is shown. - Contact number for the profile is updated.	PASS
Edit contact number with blanks	1. Successful login to UTAR Venue/Classroom Booking Web Portal 2. Successful access to profile page	Contact number: -	Alert message "Please fill out this field." appears.	Alert message "Please fill out this field." appears.	PASS

	3. Select "Edit" button				

Table 7.5: Unit Test Case for Add Booking

Test Case #	5	**Test Case Name**	Add Booking		
Test Case Summary	To test if user could submit a new venue/classroom booking request.				
Pre-Conditions	User can be accessed to the UTAR Venue/Classroom Booking Web Portal and enters the Add Booking page through the directive links in the sidebar				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Student to add booking	1. Enter date for event starting and ending.	Start date: 17/08/2024 Ending date: 17/08/2024 Start time: 4:00PM Ending time: 5:00PM	- Page reloads with header message "SUCCESS:Booking	- Page reloads with header message "SUCCESS:Booking	PASS

	2. Enter time for starting and ending. 3. Select building, floor and room to book. 4. Enter booking reason. 5. Enter expected capacity during the event. 6. Enter lecturer's name.	Building: KB Floor: 5 Room: KB510 Booking reason: Test Capacity:20 Lecturer's name: Lee Ming Jie	added successfully" is shown. - New booking data is added to database.	added successfully" is shown. - New booking data is added to database.	
Lecturer's name is not valid (Student)	1. Enter event start date and ending date. 2. Enter start time and ending time.	Start date: 17/08/2024 Ending date: 17/08/2024 Start time: 4:00PM Ending time: 5:00PM Building: KB Floor: 5	- Page reloads with header message "ERROR:Lecturer name is not found in the database!" is shown.	- Page reloads with header message "ERROR:Lecturer name is not found in the database!" is shown.	PASS

	3. Select building, floor and room to book. 4. Enter booking reason. 5. Enter expected capacity during the event. 6. Enter lecturer’s name.	Room: KB509 Booking reason: Invalid lecturer test Capacity:15 Lecturer’s name: zz	- Data is not inserted to database.	- Data is not inserted to database.	
Lecturer/Head of Department/DGS staff to add booking	1. Enter event start date and ending date. 2. Enter start time and ending time. 3. Select building, floor and room to book.	Start date: 17/08/2024 Ending date: 17/08/2024 Start time: 8:00PM Ending time: 9:00PM Building: KB Floor: 2 Room: KB208	- Page reloads with header message “SUCCESS:Booking added successfully” is shown. - New booking data is added to database.	- Page reloads with header message “SUCCESS:Booking added successfully” is shown. - New booking data is added to database.	PASS

	4. Enter booking reason. 5. Enter expected capacity during the event.	Booking reason: COcu activity Capacity:20			
Starting date is later than ending date	1. Enter event start date and ending date. 2. Enter start time and ending time.	Start date: 18/08/2024 Ending date: 17/08/2024 Start time: 4:00PM Ending time: 5:00PM	- Pop-up alert shown, with message "Warning: Planned End Date must be later than Planned Start Date!" - Ending date input column resets.	- Pop-up alert shown, with message "Warning: Planned End Date must be later than Planned Start Date!" - Ending date input column resets.	PASS
Starting time is later than ending time on a same day	1. Enter event start date and ending date.	Start date: 17/08/2024 Ending date: 17/08/2024 Start time: 5:00PM Ending time: 4:00PM	- Pop-up alert shown, with message "Warning:	- Pop-up alert shown, with message "Warning:	PASS

	2. Enter start time and ending time.		Planned End Time must be later than Planned Start Time!” - Ending time input column resets.	Planned End Time must be later than Planned Start Time!” - Ending time input column resets.	
Select a venue that is already been booked on that exact timing	1. Enter event start date and ending date. 2. Enter start time and ending time. 3. Select building, floor and room to book.	Start date: 17/08/2024 Ending date: 17/08/2024 Start time: 4:00PM Ending time: 5:00PM Building: KB Floor: 5 Room: KB510	- Intended room name is not shown in the dropdown list to select.	- Intended room name is not shown in the dropdown list to select.	PASS
Select a venue that is already been booked during that timing	1. Enter event start date and ending date.	Start date: 17/08/2024 Ending date: 17/08/2024 Start time: 4:00PM Ending time: 6:00PM	- Intended room name is not shown in the dropdown list to select.	- Intended room name is not shown in the dropdown list to select.	PASS

	2. Enter start time and ending time. 3. Select building, floor and room to book.	Building: KB Floor: 5 Room: KB510			
Enter a capacity amount that is beyond the capacity supported by the room selected	1. Enter event start date and ending date. 2. Enter start time and ending time. 3. Select building, floor and room to book. 4. Enter booking reason. 5. Enter expected capacity during the event.	Start date: 17/08/2024 Ending date: 17/08/2024 Start time: 4:00PM Ending time: 5:00PM Building: KB Floor: 5 Room: KB511 Booking reason: Test Capacity:100	- Pop-up alert shown, with message “Warning: Your capacity has exceeded the maximum capacity of this venue!” - Capacity input column resets.	- Pop-up alert shown, with message “Warning: Your capacity has exceeded the maximum capacity of this venue!” - Capacity input column resets.	PASS

Booking date that has 5 days difference with today	1. Select any starting date or ending date that has 5 days difference with today.	Start date/Ending date: 15/8/2024	Pop-up displayed, with message “You're advised to book at least 5 days in advance!”.	Pop-up displayed, with message “You're advised to book at least 5 days in advance!”.	PASS
Blanks on any input column	1.Select “Submit” button.	Start date: Ending date: Start time: Ending time: Booking Reason:	Alert on the input fields, with hints “Please fill out this field”.	Alert on the input fields, with hints “Please fill out this field”.	PASS

Table 7.6: Unit Test Case for Cancel Booking

Test Case #	6	**Test Case Name**	Cancel Booking		
Test Case Summary	To test if user could cancel a submitted yet uncompleted venue booking request				
Pre-Conditions	User had submitted a venue/classroom booking request, that it has not ongoing yet.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Cancel Booking	1. User successfully login. 2. In the dashboard, user selects the “Cancel” button on the request row, that is displayed in the	Cancel button is selected for booking reason “Cocu activity” that has the status “approved” but the event is not ongoing yet.	Confirming popup with message “Are you sure you want to cancel this booking?” is shown. - Selecting “Yes”:	Confirming popup with message “Are you sure you want to cancel this booking?” is shown. - Selecting “Yes”:	PASS

	"Your Booking(s)" data table.		Dashboard page reloads with the status updated with "Cancelled". Data is updated in the database. - Selecting "No": Popup is closed, the status for that request remains the same. Data remains the	Dashboard page reloads with the status updated with "Cancelled". Data is updated in the database. - Selecting "No": Popup is closed, the status for that request remains the same. Data remains the	

			same in the database.	same in the database.	

Table 7.7: Unit Test Case for Booking Report Viewing

Test Case #	7	**Test Case Name**	Booking Report Viewing		
Test Case Summary	To test if user could view report for their own booking request.				
Pre-Conditions	The venue booking must be under the current user's identity.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Report Viewing using correct user ID	1. After successfully logged in to the system, select "Report" button for the booking request	Booking identifier: 1JbazQvQg9S Booking requester: Foo Cher Siang	Report is shown.	Report is shown.	PASS

	row in the "Your Booking(s)" data table.	Current session user: Foo Cher Siang			
Report Viewing with anonymous booking ID	1. After successfully logged in to the system, select "Report" button for the booking request row in the "Your Booking(s)" data table. 2. Edit the booking identifier strings in the URL link	Booking identifier: 1JbazQvQg9p Booking requester: Foo Cher Siang Current session user: Foo Cher Siang	Alert with message "Invalid ID" is shown, user is redirected back to dashboard page.	Alert with message "Invalid ID" is shown, user is redirected back to dashboard page.	PASS

Table 7.8: Unit Test Case for Venue Viewing

Test Case #	8	**Test Case Name**	Venue Viewing		
Test Case Summary	To test if user could view a venue/classroom details.				
Pre-Conditions	User can be accessed to the UTAR Venue/Classroom Booking Web Portal and enters the Venue Viewing page then select a venue to view.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
View Venue	1. Select Building Block which the venue/classroom resides. 2. Surf through maps of floor plans of the building selected.	Building: KB Venue: KB207	A new tab with KB207 details is shown.	A new tab with KB207 details is shown.	PASS

	3. Select the venue to be view.				
Invalid Venue	1. Edit the venue name in the URL link that the venue is not recognised in the database.	Venue name: KB015	Alert message of “Invalid venue name!” is shown, then user is redirected back to the previous page.	Alert message of “Invalid venue name!” is shown, then user is redirected back to the previous page.	PASS

Table 7.9: Unit Test Case for Events Checking

Test Case #	9	**Test Case Name**	Events Checking
Test Case Summary	To test if user is able to check the calendar events in the “View Event” page.		
Pre-Conditions	User has logged in to the UTAR Venue/Classroom Booking Web Portal and chooses the View Event page under the Events tab in the sidebar.		
Prepared & Executed By	Wong Jia Keen		

Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
View Event	1. Choose a date in the date input column.	Date: 30/8/2024	The event table under the date input column is updated with the event name(s) and the event type(s).	The event table under the date input column is updated with the event name(s) and the event type(s).	PASS
Integrate with Add Booking function	1. Choose a date that has a event marked in the Add Booking page.	Date: 30/8/2024	Pop-up alert displayed, with message showing that the event(s) names on that date, warning the users.	Pop-up alert displayed, with message showing that the event(s) names on that date, warning the users.	PASS

Table 7.10: Unit Test Case for Messaging

Test Case #	10	**Test Case Name**	Messaging		
Test Case Summary	To test if user is able to start a conversation in the Message Page.				
Pre-Conditions	User has logged in to the UTAR Venue/Classroom Booking Web Portal and chooses the Message(s) page.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Access Existing Chat	1. Enter Message interface by selecting the Message(s) tab in the sidebar. 2.Select any one of the chat(s) boxes		The user is taken to chat interface for the chosen contact.	The user is taken to chat interface for the chosen contact.	PASS
Access Own Chatting ID	1.In the Message Page, edit the user ID	Current session user ID: 2 Edited ID to access: 2	Alert with message “You cannot chat	Alert with message “You cannot chat	PASS

	in the URL link with own user ID.		with yourself' is shown, the user is redirected to the Message Main Page.	with yourself' is shown, the user is redirected to the Message Main Page.	
Access to anonymous user	1.In the Message Page, edit the user ID in the URL link with other user ID, as long as not own user ID.	Current session user ID: 2 Edited ID to access: 21	Alert with message "You are not allowed to chat with this user" is shown, the user is redirected to the Message Main Page.	Alert with message "You are not allowed to chat with this user" is shown, the user is redirected to the Message Main Page.	PASS
Sent a message	1.Enter a valid chatting page. 2. Type a message in the input column underneath, then click "Send" icon.	This is a test message	The chatting interface is reloaded to the bottom with the latest message input.	The chatting interface is reloaded to the bottom with the latest message input.	PASS

Sent a message in another language	1.Enter a valid chatting page. 2. Type a message in the input column underneath, then click “Send” icon.	“Yes, later 我會試試”	The chatting interface is reloaded to the bottom with the latest message input, displaying in a normal order without Unicode errors.	The chatting interface is reloaded to the bottom with the latest message input, displaying in a normal order without Unicode errors.	PASS

7.3.2 Unit Test Case for Administrative Functions

This subsection discusses the testing on the functionalities that only DGS staff could use, since some functions are only available for them. The functions tested inclusive of venue details editing, event management and semester, class management under the class scheduling main function.

Table 7.11: Unit Test Case for Venue Editing

Test Case #	11	**Test Case Name**	Venue Checking
Test Case Summary	To test if administrative staff from Department of General Services could edit a venue/classroom details.		
Pre-Conditions	User selects one of the venue to be view and selects the “Edit” button.		

Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Amend Venue's facilities details	1. Choose "Edit" button. 2. User adds/removes details in the Facilities input column.	Building: KB Venue: KB207 Facilities: Microphone	- Page reloads with alert message "SUCCESS : Details updated successfully !" is shown. - Facility data is updated for that venue in the database.	- Page reloads with alert message "SUCCESS : Details updated successfully !" is shown. - Facility data is updated for that venue in the database.	PASS
Edit Venue's Status	1. Choose "Edit" button. 2. User changes the status based on the	Building: KB Venue: KB207 Current Status: Occupied	- Page reloads with alert message "SUCCESS : Details updated	- Page reloads with alert message "SUCCESS : Details updated	PASS

	options provided in dropdown list.		successfully !” is shown. - Status for the venue is updated in the database.	successfully !” is shown. - Status for the venue is updated in the database.	

Table 7.12: Unit Test Case for Event Adding

Test Case #	12	**Test Case Name**	Event Adding		
Test Case Summary	To test if administrative staff from Department of General Services could add a new calendar event				
Pre-Conditions	User is in the Manage Event(s) page, which could be directed by choosing the link hiding under the “Events” tab in the sidebar.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**

Add new calendar event	1. Select a date in date input column. 2. Enter event name. 3. Choose event type, either it is a public holiday or an UTAR internal event. 4. Select "Add new Event" button.	Date: 16/9/2024 Event Name: Malaysia Day Event Type: Public Holiday	- Page reloads with alert message "SUCCESS : Event added successfully!" is shown. - New event data is added to the database.	- Page reloads with alert message "SUCCESS : Event added successfully!" is shown. - New event data is added to the database.	PASS
Add duplicated calendar event	1. Select a date which is already initiated with event(s). 2. Add the same event name and type that is already been displayed on the	Date: 16/9/2024 Event Name: Malaysia Day Event Type: Public Holiday	- Page reloads with alert message "ERROR:Something went wrong. Please try again" is shown. - Data is not added in the database.	- Page reloads with alert message "ERROR:Something went wrong. Please try again" is shown. - Data is not added in the database.	PASS

	event(s) table underneath. 3. Select "Add new event" button.				
Blank input on Event Name	1. Select a date. 2. Leave the event name input column blank. 3. Choose either one of the event type. 4. Select "Add new event" button.	Date: 14/8/2024 Event Name: Event Type: Public Holiday	Event Name input column field appears message "Please fill out this field".	Event Name input column field appears message "Please fill out this field".	PASS
Blank input on Event Date	1. Leave date input column blank. 2. Enter an event name on the input column.	Date: Event Name: a Event Type: Public Holiday	- Page reloads with alert message "ERROR:Please select a date" is shown.	- Page reloads with alert message "ERROR:Please select a date" is shown.	PASS

	3. Choose either one of the event type. 4. Select "Add new event" button.		- Data is not added to database.	- Data is not added to database.	
Blank input on Event Type	1. Select a date in date input column. 2. Enter event name. 3. Leave event type dropbox untouched. 4. Select "Add new Event" button.	Date: 14/8/2024 Event Name: a Event Type:	- Page reloads with alert message "ERROR:Invalid event type. Please try again" is shown. - Data is not added to the database.	- Page reloads with alert message "ERROR:Invalid event type. Please try again" is shown. - Data is not added to the database.	PASS

Table 7.13: Unit Test Case for Event Editing

Test Case #	13	**Test Case Name**	Event Editing		
Test Case Summary	To test if administrative staff from Department of General Services could edit details for an existing event				
Pre-Conditions	User is in the Manage Event(s) page, then to select an event which has already been build on the selected date.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Amend event's name	1. Choose a date in date input column. 2. Select "Edit" button on the Event(s) table on the fetched event's name row.	Date: 16/9/2024 Event Name: Independence Day Event Type: Public Holiday	- Page reloads with alert message "SUCCESS : Event updated successfully!" is shown.	- Page reloads with alert message "SUCCESS : Event updated successfully!" is shown.	PASS

	3. User is redirected to the Event Editing page. 4. User selects the event name input column, then make changes. 5. User selects “Save” button after changing the name of event.		- Name of event is amended, saved in database.	- Name of event is amended, saved in database.	
Edit event’s date	1. Select a date in date input column. 2. Select “Edit” button on the Event(s) table on the fetched event’s name row.	Date: 31/9/2024 Event Name: Independence Day Event Type: Public Holiday	- Page reloads with alert message “SUCCESS : Event updated successfully!” is shown.	- Page reloads with alert message “SUCCESS : Event updated successfully!” is shown.	PASS

	3. User is redirected to the Event Editing page. 4. User selects the event date input column, then make changes. 5. User selects “Save” button after changing the event’s name.		- Event’s date is updated in the database.	- Event’s date is updated in the database.	
Edit event’s type	1. Select a date in date input column. 2. Select “Edit” button on the Event(s) table on the fetched event’s name row.	Date: 31/8/2024 Event Name: Independence Day Event Type: UTAR Internal	- Page reloads with alert message “SUCCESS : Event updated successfully!” is shown.	- Page reloads with alert message “SUCCESS : Event updated successfully!” is shown.	PASS

	3. User is redirected to the Event Editing page. 4. User selects the event type dropbox column, then make changes. 5. User selects "Save" button after changing the event's name.		- Event's type is amended, saved in database.	- Event's type is amended, saved in database.	
Blank input on Event Name	1. In the Event Editing page, keep the event name input column blank. 2. User selects "Save" button after making sure the	Date: 31/8/2024 Event Name: Event Type: Public Holiday	- Input column field for event name appears message "Please fill out this field.".	- Input column field for event name appears message "Please fill out this field.".	PASS

	event name column is empty.		- Name of event remains the same in database.	- Name of event remains the same in database.	
Deleted input for Event Date	1. In the Event Editing page, highlight the event date column and delete it. 2. User selects "Save" button after conducted the first step.	Date: dd/8/2024 Event Name: a Event Type: Public Holiday	- Input column field for event name appears message "Please enter a valid value. The field is incomplete or has an invalid date.". - Event's date is not updated in the database.	- Input column field for event name appears message "Please enter a valid value. The field is incomplete or has an invalid date.". - Event's date is not updated in the database.	PASS
Edit existing event to another duplicated event details	1. In the Event Editing page, change the event name, event date and event	Date: 16/9/2024 Event Name: Malaysia Day	- Page reloads with alert message "ERROR:Something	- Page reloads with alert message "ERROR:Something	PASS

	type to the events details that are existing. 2. User selects "Save" button after conducted the first step.	Event Type: Public Holiday	went wrong. Please try again" is shown. - Data is not saved in the database.	went wrong. Please try again" is shown. - Data is not saved in the database.	

Table 7.14: Unit Test Case for Event Deletion

Test Case #	14	**Test Case Name**	Event Deletion		
Test Case Summary	To test if administrative staff from Department of General Services could remove an existing event.				
Pre-Conditions	User selects one of the venue to be view and selects the "Delete" button.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**

Selecting "Delete" upon delete	1. In the Event Management page, enter the date which was initialized with event(s). 2. For the fetched event in the Event table, select "Delete" button for the event to be deleted. 3. Select "Delete" for the popup displayed.	Event Date:30/8/2024 Event Name: Selangor Agong Bday	- Page reloads with popup showing confirming message "Event has been deleted.". - Event has been deleted in the database.	- Page reloads with popup showing confirming message "Event has been deleted.". - Event has been deleted in the database.	PASS
Selecting "Cancel" upon delete	1. In the Event Management page, enter the date which was initialized with event(s).	Event Date:30/8/2024 Event Name: Selangor Agong Bday	- Popup closed, with event selected remains displayed on the table.	- Popup closed, with event selected remains displayed on the table.	PASS

	2. For the fetched event in the Event table, select “Delete” button for the event to be deleted. 3. Select “No” for the popup displayed.		- Event remains in the database.	- Event remains in the database.	

Table 7.15: Unit Test Case for Add Semester

Test Case #	15	**Test Case Name**	Add Semester
Test Case Summary	To test if administrative staff from Department of General Services could add a new semester.		
Pre-Conditions	User has to be DGS admin and access to the Add Semester page under the Class Scheduling tab in the sidebar.		
Prepared & Executed By	Wong Jia Keen		

Test Summary	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass / Fail)
Add new semester	1. User selects the starting date and the ending date for the new semester. 2. User selects which type of semester it is. 3. User selects which semester count for the new semester to be added.	Semester Start Date: 29/1/2024 Semester End Date: 23/6/2024 Type: Undergraduate Semester: 1	- Page reloads with header message “SUCCESS: Trimester added successfully” appears. - Data for new semester has been added in database.	- Page reloads with header message “SUCCESS: Trimester added successfully” appears. - Data for new semester has been added in database.	PASS
Add a duplicated semester	1. User selects the starting date and the ending date for the new semester. 2. User selects which type of semester it is.	Semester Start Date: 29/1/2024 Semester End Date: 23/6/2024 Type: Undergraduate Semester: 1	- Page reloads with header message “ERROR: Trimester already exists” appears.	- Page reloads with header message “ERROR: Trimester already exists” appears.	PASS

	3. User selects which semester count for the new semester to be added.		- Data is not added in the database.	- Data is not added in the database.	
Semester that is lesser than 3 months	1. User selects the starting date and the ending date for the new semester. 2. User selects which type of semester it is. 3. User selects which semester count for the new semester to be added. 4. The duration between the start date and the end date	Semester Start Date: 29/1/2024 Semester End Date: 24/3/2024 Type: Undergraduate Semester: 1	- Page reloads with header message “ERROR: Semester duration must be at least 3 months” appears. - Data is not added in the database.	- Page reloads with header message “ERROR: Semester duration must be at least 3 months” appears. - Data is not added in the database.	PASS

	is lesser than 3 months.				

Table 7.16: Unit Test Case for Edit Semester

Test Case #	16	**Test Case Name**	Edit Semester		
Test Case Summary	To test if administrative staff from Department of General Services could edit an added semester details.				
Pre-Conditions	User has to be DGS admin and access to the View Semester page, then selects one of the semester to edit.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Edit an existing semester	1. User selects the semester to edit. 2. User is taken to the Edit Semester interface with the	Semester Chosen: 2024 Postgraduate T1 Semester Start Date: 16/1/2024	- Page reloads with header message "SUCCESS: Trimester updated	- Page reloads with header message "SUCCESS: Trimester updated	PASS

	information fetched for the semester selected. 3. User could edit the start date and ending date only. 4. User selects “Save” button after confirming the changes.	Semester End Date:24/5/2024	successfully” appears. - Data for the semester is updated in the database.	successfully” appears. - Data for the semester is updated in the database.	
Start date is greater than end date	1. User selects the semester to edit. 2. User is redirected to the Edit Semester page with the details fetched for the semester selected.	Semester Chosen: 2024 Postgraduate T1 Semester Start Date: 16/1/2024 Semester End Date:24/12/2023	- Alert message on the ending date input field, showing “Value must be 16/1/2024 or later”. - Data is not updated in the database.	- Alert message on the ending date input field, showing “Value must be 16/1/2024 or later”. - Data is not updated in the database.	PASS

	3. User could edit the start date and ending date only. 4. User to input start date that is greater than the ending date. 5. User selects "Save" button after confirming the changes.				

Table 7.17: Unit Test Case for Semester Deletion

Test Case #	17	**Test Case Name**	Semester Deletion
Test Case Summary	To test if administrative staff from Department of General Services could remove an added semester.		
Pre-Conditions	User has to access to the View Semester page and selects Delete button on any semester that is intended for deleting.		

Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Selecting "Delete" upon delete	1. In Semester Viewing page, select the Delete button along the semester row that is intended for deleting. 2. Select "Delete" for the popup displayed.	Semester Name: 2024 Undergraduate T1	- Page reloads with popup showing confirming message "Trimester has been deleted.". - Trimester data has been deleted from database.	- Page reloads with popup showing confirming message "Trimester has been deleted.". - Trimester data has been deleted from database.	PASS
Selecting "No" upon delete	1. In the Semester Viewing page, select the Delete button along the semester	Semester Name: 2024 Postgraduate T1	- Popup closed, with semester selected remains displayed on the table.	- Popup closed, with semester selected remains displayed on the table.	PASS

	row that is intended for deleting. 2. Select “Delete” for the popup displayed.		- Trimester data remains in the database.	- Trimester data remains in the database.	

Table 7.18: Unit Test Case for Adding Class

Test Case #	18	**Test Case Name**	Adding Class		
Test Case Summary	To test if administrative staff from Department of General Services could book a venue for the entire semester through adding a class.				
Pre-Conditions	User has to access to the Add Class page and selects details for adding a class that is intended for reserving a venue throughout the entire semester.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Adding a class session	1. Select the trimester level,	Level: Undergraduate Start Date: 28/10/2024	- Page reloads with venue added into	- Page reloads with venue added into	PASS

	whether it is Foundation, Undergraduate or Postgraduate. 2. Select Faculty. 3. Select the course name after the faculty chosen. 4. Select the weekday that the class resides to. 5. Select the week mode for the class, whether is it ongoing weekly, even week or odd week.	Ending Date: 2/2/2025 Faculty: FMHS Course Name: UMFA2062 Pharmacology for Physiotherapy Day: Tuesday Class Mode: Weekly Start Time: 8:15 AM End Time: 12:00 PM Block Name:KB Floor: 2nd floor Selected Room: KB208	database, with header messaging showing “SUCCESS:Class has been added successfully!”. - New class data has been added in the database.	database, with header messaging showing “SUCCESS:Class has been added successfully!”. - New class data has been added in the database.	

	6. Select starting time and ending time for the class. 7. Select Building Block, Floor and which venue/classroom does the class going to reserve. 8. Select "Submit" button.				
Blanks on any input field column	1. Select the trimester level, whether it is Foundation, Undergraduate or Postgraduate.	Level: Start Date: Ending Date: Faculty: Course Name: Day:	- Input field column shows message "Please fill out this field.".	- Input field column shows message "Please fill out this field.".	PASS

	2. Select Faculty. 3. Select the course name after the faculty chosen. 4. Select the weekday that the class resides to. 5. Select the week mode for the class, whether is it ongoing weekly, even week or odd week. 6. Select starting time and ending time for the class. 7. Select Building Block, Floor and which	Class Mode: Start Time: 8:15 AM End Time: 12:00 PM	- No trimester data is added into the database.	- No trimester data is added into the database.	

	venue/classroom does the class going to reserve. 8. Select "Submit" button.				

Table 7.19: Unit Test Case for Edit Class

Test Case #	19	**Test Case Name**	Edit Class		
Test Case Summary	To test if administrative staff from Department of General Services could edit date and timing for a particular class.				
Pre-Conditions	User has to access to the View Class page and selects Edit button on any class that needs to be amended.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**

Valid edit on class date.	1. In Edit Class page, change the date value beside the Class Date column. 2. Select "Save" after changing.	Class Name: LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELOPMENT Class Date:2/7/2024	- Page reloads with data updated, header showing confirming message "SUCCESS : Class updated successfully !". - Class date is updated in the database.	- Page reloads with data updated, header showing confirming message "SUCCESS : Class updated successfully !". - Class date is updated in the database.	PASS
Invalid edit on the class date	1. In the Edit Class page, delete the date value beside the Class Date column. 2. Select "Save" after changing.	Class Name: LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELOPMENT Class Date:dd/7/2024	- Date input column field appears message "Please enter a valid value. The field is incomplete or has an invalid date".	- Date input column field appears message "Please enter a valid value. The field is incomplete or has an invalid date".	PASS

			- Class date is not updated in the database.	- Class date is not updated in the database.	
Valid edit on the class starting time	1. In the Edit Class page, ensure the start time value is not greater than the ending time value. 2. Select "Save" after changing.	Class Name: LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELOPMENT Start Time: 10:00AM End Time: 1:30PM	- Page reloads with data updated, header showing confirming message "SUCCESS : Class updated successfully !". - Starting time is updated in the database.	- Page reloads with data updated, header showing confirming message "SUCCESS : Class updated successfully !". - Starting time is updated in the database.	PASS
Starting Time value is greater than Ending Time value	1. In the Edit Class page, ensure the start time value is greater	Class Name: LKCFES-UECS1144 OBJECT-ORIENTED	- Alert message displays immediately,	- Alert message displays immediately,	PASS

	than the ending time value. 2. Select "Save" after changing.	APPLICATION DEVELOPMENT Start Time: 5:30PM End Time: 12:00PM	showing that "End time must be greater than start time!". - Time is not updated in the database.	showing that "End time must be greater than start time!". - Time is not updated in the database.	
Edit to a duplicated class value	1. In the Edit Class page, change the date and time value to the existing venue booking value with the same venue name. 2. Select "Save" after changing.	Class Name: LKCFES-UECS1144 OBJECT-ORIENTED APPLICATION DEVELOPMENT Class Date: 8/7/2024 Start Time: 5:30PM End Time: 12:00PM	- Alert message pops out, with message "This class is clashing with another class. Please select another date or time.". - Input columns for date and start time and ending time get reset.	- Alert message pops out, with message "This class is clashing with another class. Please select another date or time.". - Input columns for date and start time and ending time get reset.	PASS

			- No info is updated in the database.	- No info is updated in the database.	

Table 7.20: Unit Test Case for Class Deletion

Test Case #	20	**Test Case Name**	Class Deletion		
Test Case Summary	To test if administrative staff from Department of General Services could remove an added class.				
Pre-Conditions	User has to access to the View Class page and selects Delete button on any class that is intended for deleting.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Selecting "Delete" upon delete	1. In Class Viewing page, select the Delete button along	Class Name: LKC Class Date Selected: 2024-07-02	- Page reloads with popup showing confirming message	- Page reloads with popup showing confirming message	PASS

	the class row that is intended for deleting. 2. Select "Delete" for the popup displayed.	Class Time: 09:00:00 - 12:00:00	"Class has been deleted.". - Class data is deleted in the database.	"Class has been deleted.". - Class data is deleted in the database.	
Selecting "No" upon delete	1. In the Semester Viewing page, select the Delete button along the semester row that is intended for deleting. 2. Select "Delete" for the popup displayed.	Semester Name: 2024 Postgraduate T1	- Popup closed, with class selected remains displayed on the table. - Class data remains in the database.	- Popup closed, with class selected remains displayed on the table. - Class data remains in the database.	PASS

Table 7.21: Unit Test Case for Accessing Validation

Test Case #	21	**Test Case Name**	Accessing Validation		
Test Case Summary	To test if lecturers and students users are prohibited from accessing management functions.				
Pre-Conditions	Students and lecturers has to access the management functional pages by editing the links after logged into the UTAR Venue/Classroom Booking Web Portal.				
Prepared & Executed By	Wong Jia Keen				
Test Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass / Fail)**
Accessing "Add Class"	1. Current user is a	URL: http://localhost/cr_system/VenueRentalSystem/AddClass.php	- Page reloads with	- Page reloads with	PASS

functional page	student/lecturer or HOD. 2. User edits the URL link, attempting to access the Add Class page.		popup showing alerting message "You are not authorized to access this page.". - User is relocated to the dashboard page.	popup showing alerting message "You are not authorized to access this page.". - User is relocated to the dashboard page.	

Accessing “Edit Class” functional page	1. User is logged in as a student/lecturer or HOD. 2. User edits the URL link, attempting to access the Edit Class page.	URL: http://localhost/cr_system/VenueRentalSystem/EditClass.php?b=aC3vzkL8XaX	- Page reloads with popup showing alerting message “You do not have access to this page.”. - User is relocated to the dashboard page.	- Page reloads with popup showing alerting message “You do not have access to this page.”. - User is relocated to the dashboard page.	PASS

Accessing "View Class" functional page	1. User is logged in as a student/lecturer or HOD. 2. User edits the URL link, attempting to access the View Class page.	URL: http://localhost/cr_system/VenueRentalSystem/ViewClass.php	- Page reloads with popup showing alerting message "You are not authorized to access this page". - User is relocated to the	- Page reloads with popup showing alerting message "You are not authorized to access this page.". - User is relocated to the	PASS

			dashboard page.	dashboard page.	
Accessing “Edit Event” functional page	1. User is logged in as a student or lecturer. 2. User edits the URL link, attempting to access the Edit Event page.	URL: http://localhost/cr_system/VenueRentalSystem/editEvent.php?c=1	- Page reloads with popup showing alerting message “You are not authorized to access this page.”.	- Page reloads with popup showing alerting message “You are not authorized to access this page.”.	PASS

			- User is relocated to the dashboard page.	- User is relocated to the dashboard page.	
Accessing "Edit Semester" functional page	1. User is logged in as a student or lecturer. 2. User edits the URL link, attempting to access the Edit Semester page.	URL: http://localhost/cr_system/VenueRentalSystem/EditSemester.php?t=3	- Page reloads with popup showing alerting message "You cannot access this page".	- Page reloads with popup showing alerting message "You cannot access this page".	PASS

			- User is relocated to the dashboard page.	- User is relocated to the dashboard page.	
Accessing “View Semester” functional page	1. User is logged in as a student/lecturer or HOD. 2. User edits the URL link, attempting to access the View Semester page.	URL: http://localhost/cr_system/VenueRentalSystem/ViewSemester.php	- Page reloads with popup showing alerting message “Invalid access”. - User is relocated to the	- Page reloads with popup showing alerting message “Invalid access”. - User is relocated to the	PASS

			dashboard page.	dashboard page.	
Accessing "Event Management" functional page	1. User is logged in as a student or lecturer. 2. User edits the URL link, attempting to access the Event Management page.	URL: http://localhost/cr_system/VenueRentalSystem/EventManagement.php	- Page reloads with popup showing alerting message "You are not authorized to access this page.".	- Page reloads with popup showing alerting message "You are not authorized to access this page.".	PASS

			- User is relocated to the dashboard page.	- User is relocated to the dashboard page.	
Accessing “Manage Booking” functional page	1. User is logged in as a student or lecturer. 2. User edits the URL link, attempting to access the Manage Booking page.	URL: http://localhost/cr_system/VenueRentalSystem/manageBooking.php	- Page reloads with popup showing alerting message “You cannot access this page!.”.	- Page reloads with popup showing alerting message “You cannot access this page!”.	PASS

			- User is relocated to the dashboard page.	- User is relocated to the dashboard page.	
Accessing “Reject Booking” functional page	1. User is logged in as a student or lecturer. 2. User edits the URL link, attempting to access the Reject Booking page.	URL: http://localhost/cr_system/VenueRentalSystem/RejectRequest.php?b=hQW5lFpPq9S	- Page reloads with popup showing alerting message “You do not have access to this page”.	- Page reloads with popup showing alerting message “You do not have access to this page”.	PASS

			- User is relocated to the dashboard page.	- User is relocated to the dashboard page.	
Accessing "Reject Booking" functional page with invalid booking request ID	1. User edits the URL link, attempting to access the Reject Booking page, with an invalid venue booking ID.	URL: http://localhost/cr_system/VenueRentalSystem/RejectRequest.php?reject=6	- Page reloads with popup showing alerting message "Invalid request.". - User is relocated to the	- Page reloads with popup showing alerting message "Invalid request.". - User is relocated to the	PASS

			dashboard page.	dashboard page.	

7.4 Usability Testing

Usability Testing requires certain potential users to try out the project outcome, intending to examine the usage experience of the entire system developed. From the feedback gained by the invited testers, refinements could be conducted to leverage the parts to be improved.

In this case, ten participants which their identity ranged from UTAR students, lecturers, HOD and administrators from DGS were involved in this testing. They explored the system with little guidance from the project author and were required to fill the System Usability Scale questionnaire, which consists of 10 usage-related questions. On a scale of 1-5 which signifies the degree of agreement, the testers need to mark a score for each question. The scores will be summed, representing the users' impression of the usability issues on the system. Referring to the questionnaire questions in Appendix A, the following table describes the scores collected for each tester.

Table 7.22: User Satisfaction Scale and results

Participants.	Score (s)										Total	Percentage
	1	2	3	4	5	6	7	8	9	10		
1	5	2	5	1	4	1	5	1	5	1	38	95
2	4	2	4	2	4	2	4	1	4	2	31	77.5
3	5	2	4	2	4	2	4	2	5	1	33	82.5
4	4	2	4	3	4	1	5	3	5	2	31	77.5
5	4	3	5	3	5	2	5	2	5	4	30	75
6	5	1	5	1	5	1	5	1	5	1	40	100
7	5	1	4	3	5	1	3	1	5	2	35	87.5
8	3	1	4	3	4	2	4	2	4	3	28	70
9	4	3	4	2	4	3	3	2	4	3	26	65
10	4	3	4	2	4	3	4	2	4	2	28	70
Average Score (percentage)		80%										
Grade		A										

Referring to the figure 2.23, the system developed had achieved the SUS grade of A- since the score is 80%. Therefore, this project has met the objective #4 of reaching 80% in the usability testing, meaning it is widely accepted.

7.5 User Acceptance Test

Conducted along System Usability Testing, this testing measures the operability of the functions built in the UTAR Venue/Classroom Booking Web Portal. Guided by the project author, the testers need to go through the scenarios that involved all possible functionalities in the system. Since there are two levels of potential end-users (administrative: DGS and HOD and non-administrative: lecturers and students) which featured with more management functions, there will be more test columns for the former level testers.

7.5.1 User Acceptance Test for Lecturers and Students

Table 7.23: Non-administrative level User Acceptance Test results

Tester No. / Test	1	2	3	4	5	6	7
Login (Successful and Failed)	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Profile (View, Edit)	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Venue Booking (Add, View, Cancel, View in report)	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Venue Viewing	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Event Viewing	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Messaging (View, Chat)	Pass	Fail	Pass	Pass	Pass	Pass	Pass
Notification Function	Pass	Pass	Pass	Pass	Pass	Pass	Pass

The feedback collected from the non-administrative testers are focusing on the functionalities. It was discovered that if the system is used under UTAR wired connection, then some functionalities are functioning abnormally, like some wordings are unable to display. It was believed this happened due to connection firewall set. Considering this system to be used in UTAR community, this restriction must be given attention.

Other than that, some suggested that certain functionalities needed to be improved due to vague meaning of the features. Especially the contents in the SMS notification, it could be written clearer so that the booking requesters could fully understand the updates with just a glimpse. The map for the View Venue function could be simpler, as the featured one is used from the existing floor plans which contains many architecture study elements, considered unnecessary in this case. Form refilling is also a notable issue, as it forces users to retype once the users omitted a particular when using an input function.

7.5.2 User Acceptance Test for DGS Admin and HOD

Table 7.24: Administrative level User Acceptance Test results

Tester No. / Test	8	9	10
Login (Successful and Failed)	Pass	Pass	Pass
Profile (View, Edit)	Pass	Pass	Pass
Venue Booking (Add, View, Cancel, View in report, Approval, Rejection)	Pass	Pass	Pass
Venue Management (Viewing, Editing)	Pass	Pass	Pass

Event Management (Viewing, Editing, Deleting)	Pass	Pass	Pass
Messaging (View, Chat)	Pass	Pass	Pass
Scheduling Management (Adding, Editing, Deleting)	Pass	Pass	Pass
Class Management (Adding, Editing, Deleting)	Pass	Pass	Pass
Notification Function	Pass	Pass	Pass

Regarding the feedback from the administrative users, the HOD stated that the requirements for the dashboard could be clarify further as the information to be displayed on the dashboard could be different for every administrative user. The venue request management logics could be remodified since it could involve users from more departments in the future. Refer to the 8th participant's opinions in Appendix J to learn more about the feedback.

Based on the feedback from one of the DGS staff, the staff stated that the system developed had fulfilled her needs than expectations. There are many suggestions that hope the system could improve, such as the function to remind the booking requesters to cancel the venue bookings they had submitted since long time ago. Else, there are still many improvements on the Standard Operating Procedures (SOP), like to extend the end-users from other departments since they are handling the co-curriculum activities that could relate with the venue usage arrangement. This could be beyond from the project scope but still a notable point when this system gets deployed in the future. Refer to the 9th and 10th participants' views in Appendix J to learn more about the feedback.

Regardless of the users' level, everyone preferred this system than the existing one, since it facilitates many tasks that needs to be done manually, like the check venues' availability function and the overlooking function. Furthermore, it automates the communication between the users when it comes to venue booking purpose, which speeds up the efficiency as well. However, there are still amendments to be made on, which is the SMS contents, and the interface needs to be simplified further as some testers are not aware of the contents when testing.

CHAPTER 8

CONCLUSION AND RECOMMENDATIONS

8.1 Conclusion

Throughout months of planning and utilizing the Waterfall development approach, the development of the UTAR Venue/Classroom Booking Web Portal has marked an end, implementing all functions proposed since early and it has accomplished a user satisfaction rate of 80%. This system is expected to enhance the efficiency of venue booking scenario in UTAR as the existing mechanism had been critiqued for poor information retrieval and most of the processes are carried out manually, which leaves a major space of improvement.

The UTAR Venue/Classroom Booking Web Portal is equipped with the fundamental functions to support venue booking in UTAR Sungai Long campus, inclusive of the venue reservation, view previous reservation records, check venues' condition. Functions that are indirectly relevant to venue booking are included as well, like the events checking feature. To automate the communication between the end-users and to convey booking information faster, emailing function and SMS notification were featured to alert them anytime. Since the end-users for this system are classified according to their status, management functions are only accessible by administrative level users. As this system is heavily dependent on user input, all levels of users is capable to execute CRUD operations in a controlled environment.

This project has fulfilled the goal of setting a system that serves the venue booking purpose with the planned resources in a clearly defined project timeline. It was expected that the project outcome could benefit the UTAR community as what stated in the objectives in the future.

8.2 Achievement of Objectives

One of the project objectives is to publish an independent yet upgraded venue/classroom booking system. Since the existing venue booking system is attached under the UTAR Staff Services intranet, this project had achieved the aim by developing a sophisticated system that is solely for UTAR parties (students, lecturers, Head of Department, Department of General Services) to perform this usage, which support functionalities like venue booking, events checking and any venue booking clarification matters. The project outcome had addressed all mentioned problems by featuring the right functionalities, such as to view the venue availability instantly and to gain knowledge of the venues' facilities. With this to be said, all UTAR parties are eligible to enjoy the benefit of modern venue booking system in UTAR community.

Another objective is to achieve a basic satisfaction score of 80% through conducting System Usability Testing. Serving as a practical system which needs to cater venue booking usage, eight potential end-users with different identities are invited to test the functionalities and rate their usage experience in a scoring method. The feedback gained in this testing aided in future refinement as well.

Eliciting the scope for enhancement is vital for this development project as it was considered an upgrade on the existing venue booking system. Therefore, this project has met the third objective at the planning stage, that is to conduct a requirements and satisfaction study on the current venue/classroom booking system and to identify the areas for improvement. This is to ensure the problems stated by the current end-users are collected in a logical yet systematic method. Questionnaire distribution to the UTAR lecturers and manual interviewing are the methods selected to gain understanding on end-users' preference. Suggested solutions collected during this phase could be used as a motivation to the project author to implement in the system.

8.3 Limitation

The development project faced some limitations that restrain the system's efficiency. First, the SMS functionality operates based on an amount of subscription fee, which could be considered as a basic operating cost. The service fee could be higher if the end-users suggested that there should be more scenarios that trigger this function. More SMS scenarios indicate that the messaging quota could be consumed faster.

Another restriction is that this development project is unorganized in terms of structuring. Comparing to modern web development frameworks such as Laravel or Vue.js which emphasize the MVC (Model-View-Controller) concept to organize the files based on their functionality, the UTAR Venue/Classroom Booking Web Portal is developed with the pure web development method. Using PHP as the backbone, there is only little file classification, depending on the project author's intention to structure the scripts. With this to be said, refactoring could be a challenging task if the project author collaborates with other developers to work on this project.

The entire system rarely focuses on data encryption and protection. Although it was mentionable that accessing validation are performed on all pages which every user will be checked if they are eligible to access the redirected page, but since there are many SQL scripts running on background, this means that injection attacks could occur if this system is open to the public and being targeted by the attackers.

8.4 Recommendation

Regarding the SMS functionality, it was suggested that the instant alerting function could be considered wisely by the stakeholders as this idea in fact is beneficial than the emailing function as it could be noticed by the end-users in time. It is a feasible idea as the subscription fee is worth the service quality. Since this development project applies the SMS API function provided by Bulk360, the lowest package for this service only costs RM110 for 1000 SMS sent. The amount provided is sufficient for the UTAR venue booking case.

The technical limitation arises due to choice of framework chosen. Considering the project scope and complexity, it is advised to reframe this project in modern web development frameworks as mentioned before so that the overall project structure is well organized and aids in technical reading for future maintenance. With the choice of framework chosen being switched, this also gives more chance to tighten the confidentiality of the data as certain frameworks had been built in with data protection measures. For example, Laravel protects the website from SQL injection attacks by using *whereRaw* methods for information fetching or disabling error reporting function to prevent the attackers to stand a chance to exploit the input scenarios.

REFERENCES

Anil Kumar, S. V, Bawge, G. and Kumar, V., 2021. *An Overview of Industrial Revolution and Technology of Industrial 4.0*. [online] *International Journal of Research in Engineering and Science (IJRES) ISSN*, Available at: <https://www.ijres.org/papers/Volume-9/Issue-1/3/L09016471.pdf> [Accessed 7 April 2024].

Amanda Onion, Missy Sullivan, Matt Mullen, Christian Zapata, 2010. *The Invention of the Internet.* [Online] Available at: <https://www.history.com/topics/inventions/invention-of-the-internet> [Accessed 27 March 2024].

Au, K.Y., 2024. *Opinions on current UTAR venue/classroom booking mechanism* [Interview] (25 March 2024).

Ben, S., 1986. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*.

Bointon, M., 2019. *ReadME for PHPMailer.* [Online] Available at: https://github.com/PHPMailer/PHPMailer/ [Accessed 4 September 2024].

Bootstrap, -. *Grid system.* [Online] Available at: https://getbootstrap.com/docs/4.0/layout/grid/ [Accessed 5 September 2024].

Brooke, J., 1995. *SUS: A quick and dirty usability scale*. [online] Available at: <https://www.researchgate.net/publication/228593520>.

Dennis, A., Barbara, H.W. and David, T., 2009. *Systems Analysis Design UML Version 2.0 An Object-Oriented Approach Third Edition*. 3rd ed. John Wiley & Sons Inc.

Elmasri, R. and Navathe, S.B., 2016. *Fundamentals of Database Systems SEVENTH EDITION*. 7th ed. Pearson.

Foo., C. S., 2024. *Opinions and Procedures for UTAR Venue/Classroom Booking System* [Interview] (5 April 2024).

Garba, M. and Abubakar, H., 2020. *A Comparison of NoSQL and Relational Database Management Systems (RDBMS)*. [online] *KASU JOURNAL OF MATHEMATICAL SCIENCES (KJMS)*, E-Copy. Available at: <http:/www.journal.kasu.edu.ng/index.php/kjms>.

Lee, Z.Q., 2024. *Opinions on current UTAR venue/classroom booking mechanism* [Interview] (27 March 2024).

Lewis, J.R., 2018. *Item Benchmarks for the System Usability Scale*. [online] Available at: <https://www.researchgate.net/publication/330225055>.

Liew, L., 2024. *Opinions on current UTAR venue/classroom booking mechanism* [Interview] (14 March 2024).

MDN contributors, n.d. *Using XMLHttpRequest- mdn web_docs.* [Online] Available at: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest_API/Using_XMLHttpRequest
[Accessed 4 September 2024].

Microsoft, 2023. *Looking back at 10 years of Microsoft 365 making history.* [Online] Available at: <https://www.microsoft.com/en-us/microsoft-365-life-hacks/stories/looking-back-ten-years-microsoft-365>
[Accessed 7 April 2024].

Mudassar, S. and Khan, A., 2023. *What is a Non-Functional Requirement? Reference: Software Requirements Engineering What is a Non-Functional*

Requirement? [online] Available at: <https://www.researchgate.net/publication/371904469>.

Mumu, J., Tanujaya, B., Charitas, R. and Prahmana, I., 2022. Likert Scale in Social Sciences Research: Problems and Difficulties. *FWU Journal of Social Sciences*, 16(4), pp.89–101. https://doi.org/10.51709/19951272/Winter2022/7.

Ng, J.Y., 2024. *Opinions on current UTAR venue/classroom booking mechanism* [Interview] (29 March 2024).

Ong, Z.Y., 2024. *Opinions on current UTAR venue/classroom booking mechanism* [Interview] (11 March 2024).

R. Burke Johnson and Larry Christensen, 2014. *Educational Research: Quantitative, Qualitative, and Mixed Approaches*. 5th ed. Sage.

Roger, S.P., 2010. *Software Engineering: A Practitioner's Approach*. 7th ed. [online] New York: McGraw-Hill. Available at: <https://www.mlsu.ac.in/econtents/16_EBOOK-7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman_.pdf> [Accessed 25 March 2024].

Soo, J. L. H., 2024. *Opinions on current UTAR venue/classroom booking mechanism* [Interview] (11 March 2024).

Sukamolson, S., 2007. *Fundamentals of quantitative research*. [online] Available at: <https://www.researchgate.net/publication/242772176_Fundamentals_of_quantitative_research> [Accessed 11 March 2024].

Teng, Y. 2024. *Opinions on current UTAR venue/classroom booking mechanism* [Interview] (19 March 2024).

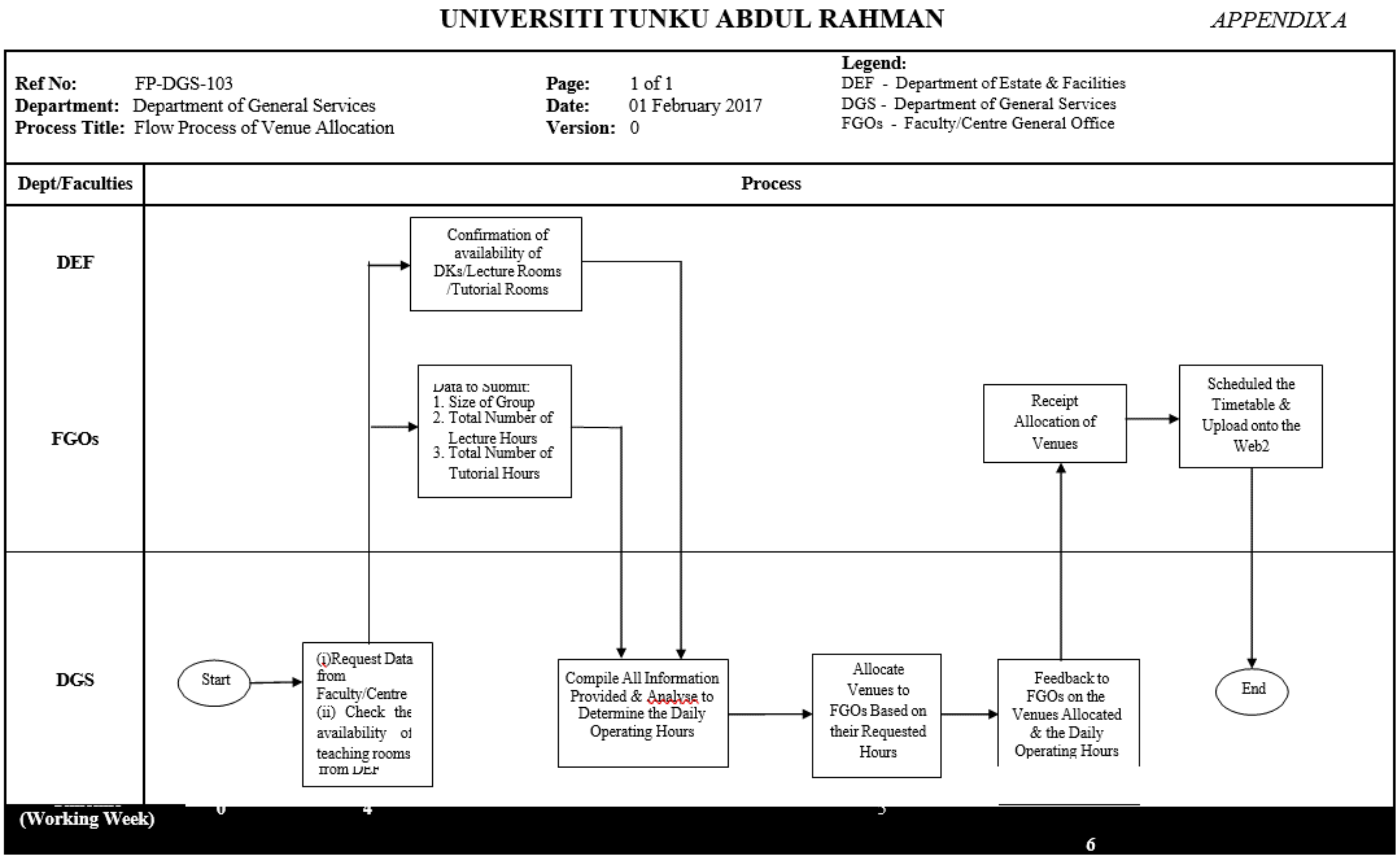
UTAR, n.d. *Rental Rates for UTAR Facilities (Sungai Long).* [Online] Available at: <https://utar.edu.my//documents/RentalRates_facilities_SungaiLong_20230331.pdf>
[Accessed 7 April 2024].

UTAR, 2024a. *Sg. Long KB building floor plan*. [online] Available at: <https://portal.utar.edu.my/stuIntranet/slkbfloorplan/index.jsp> [Accessed 7 April 2024].

UTAR, 2024b. *UTAR Sungai Long Campus*. [online] Available at: <https://study.utar.edu.my/sungai-long-campus.php> [Accessed 7 April 2024].

APPENDICES

Appendix A: Workflows on Booking of Venues/Classrooms



UNIVERSITI TUNKU ABDUL RAHMAN *APPENDIX A*

Ref No: FP-DGS-103
Department: Department of General Services
Process Title: Flow Process of Venue Allocation

Page: 1 of 1
Date: 01 February 2017
Version: 0

Legend:
DEF - Department of Estate & Facilities
DGS - Department of General Services
FGOs - Faculty/Centre General Office

Dept/Faculties
Process
DEF
FGOs
DGS
Confirmation of availability of DKs/Lecture Rooms /Tutorial Rooms
Data to Submit:
1. Size of Group
2. Total Number of Lecture Hours
3. Total Number of Tutorial Hours
Receipt Allocation of Venues
Scheduled the Timetable & Upload onto the Web2
Start
(i)Request Data from Faculty/Centre (ii) Check the availability of teaching rooms from DEF
Compile All Information Provided & Analyse to Determine the Daily Operating Hours
Allocate Venues to FGOs Based on their Requested Hours
Feedback to FGOs on the Venues Allocated & the Daily Operating Hours
End
(Working Week)
0
4
5
6

Appendix B: Guideline on Booking of Teaching Room

1. <u>Objective</u>

   The document serves to provide the necessary guidelines for booking of teaching room for replacement class, mid-term test, in-house training, meetings, club & societies activities, faculties' events, etc.

2. <u>Scope</u>

   The guideline applies to all lecture halls, lecture rooms & tutorial rooms.

3. <u>Process Flow</u>

   Please refer to attached Appendix A.

4. <u>Timeline Booking of Teaching Rooms</u>

   4.1 Booking of lecture hall, lecture room and tutorial room will only be allocated after the students' timetables are finalized. Please refer to the below timeline:

   (a) Long Trimester (14 weeks or 12 weeks)

   - From 3rd week onwards of the trimester

   (b) Short Trimester (7 weeks)

   - From 2nd week onwards of the trimester

   4.2 For faculty/department which would like to conduct any activities before the above said timeline, they need to refer and submit the booking form to DGS for venue confirmation. The approval on the use of the venue is based on the availability of the venue.

   4.3 Booking through phone or email is not accepted.

5. <u>Submission of Booking Form to DGS</u>

   5.1 Booking form must be submitted at least five (5) working days in advance from the booking date.

   5.2 To ensure teaching rooms' bookings are processed on time, DGS need four (4) working days from the date of receipt to process the booking forms.

5.3 All available time slots or venues for booking are on first-come-first-serve basis.

5.4 To ensure the teaching rooms are fairly utilized, any venues which required to be booked more than 10 days in advance from the booking date will not be processed by DGS unless under special circumstances with justification and approval from Dean/Director/Administrative Head of Department.

5.5 Bookings on Public Holiday required approval from Dean/Director/Administrative Head of Department as it will incur operational cost.

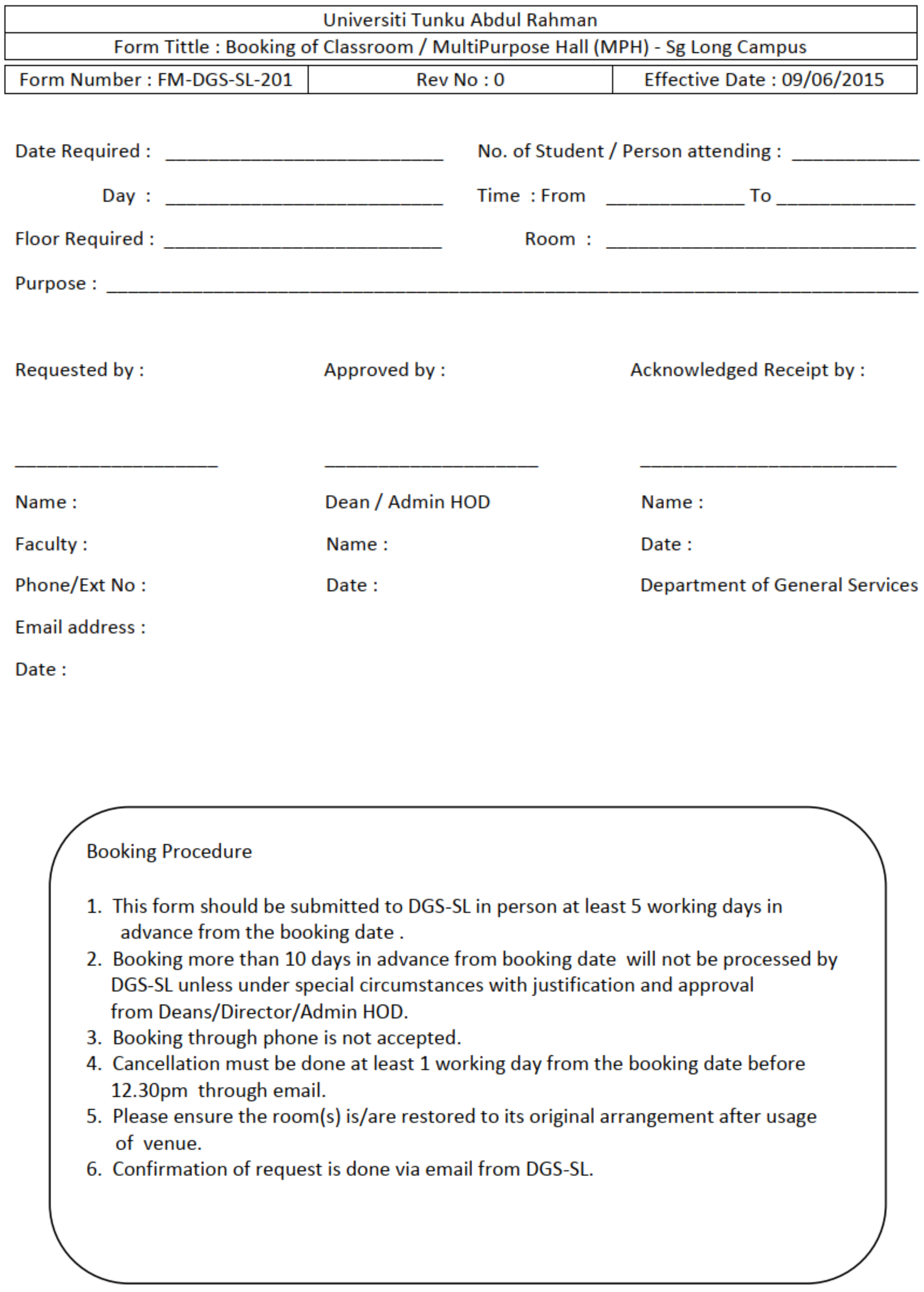
6. <u>Booking on weekdays after 6pm and on weekends (for UTAR Kampar Campus only)</u>

6.1 All bookings on weekdays after 6pm and on weekends will be allocated to FBF Block and Lecture Complex I (> 100 pax) due to energy saving.

7. <u>Booking Cancellation</u>

7.1 Any cancellation of booking, the requester must inform DGS at least one (1) working day from the booking date before 12.30pm through email to optimize the use of space.

Appendix C: UTAR Venue Application Form



Universiti Tunku Abdul Rahman		
Form Tittle : Booking of Classroom / MultiPurpose Hall (MPH) - Sg Long Campus		
Form Number : FM-DGS-SL-201	Rev No : 0	Effective Date : 09/06/2015

Date Required : ____________________________ No. of Student / Person attending : ____________

Day : ____________________________ Time : From _____________ To _____________

Floor Required : ____________________________ Room : _______________________________

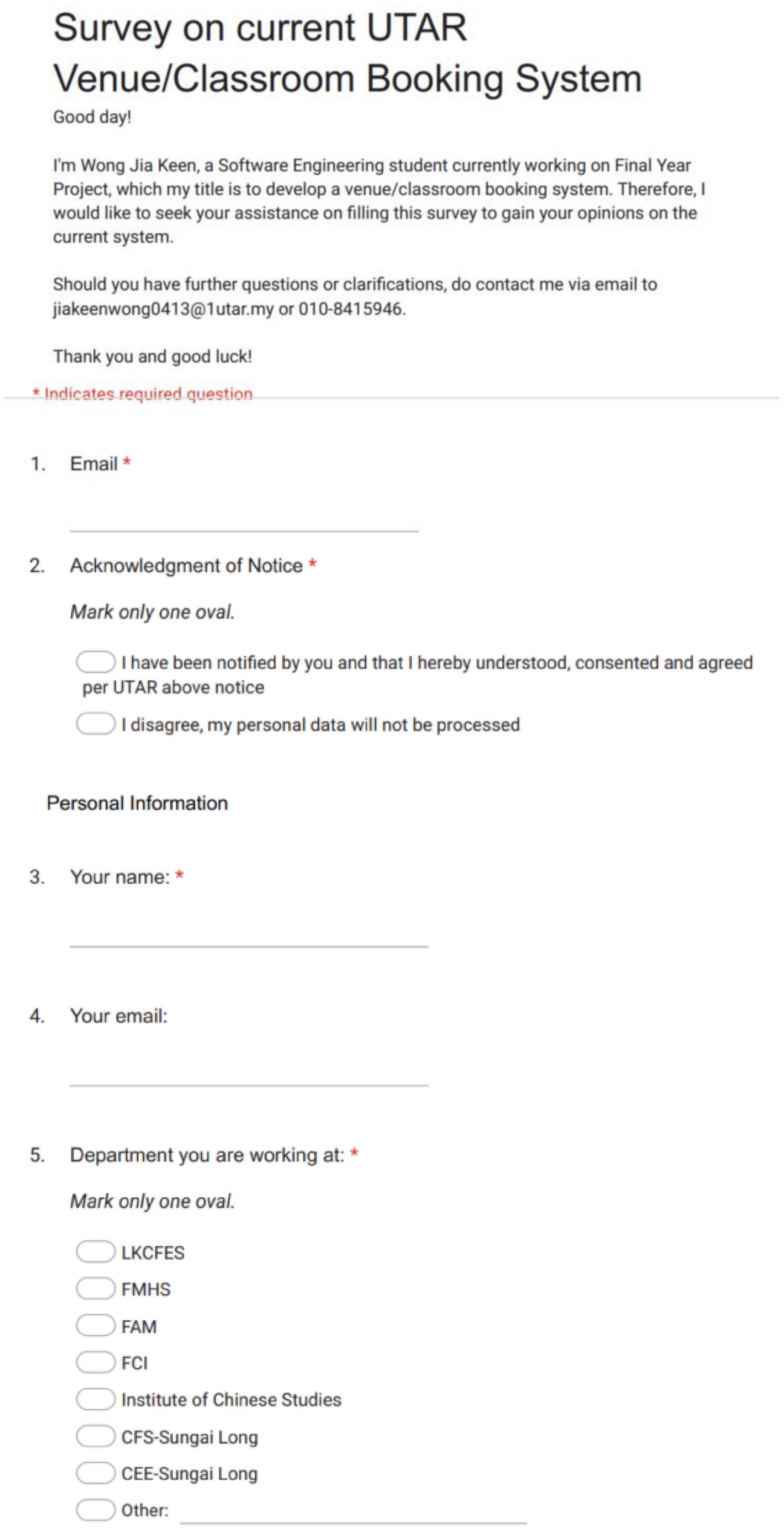
Purpose : ____

Requested by :	Approved by :	Acknowledged Receipt by :
____________________	_____________________	_________________________
Name :	Dean / Admin HOD	Name :
Faculty :	Name :	Date :
Phone/Ext No :	Date :	Department of General Services
Email address :		
Date :		

Booking Procedure

1. This form should be submitted to DGS-SL in person at least 5 working days in advance from the booking date .
2. Booking more than 10 days in advance from booking date will not be processed by DGS-SL unless under special circumstances with justification and approval from Deans/Director/Admin HOD.
3. Booking through phone is not accepted.
4. Cancellation must be done at least 1 working day from the booking date before 12.30pm through email.
5. Please ensure the room(s) is/are restored to its original arrangement after usage of venue.
6. Confirmation of request is done via email from DGS-SL.

Appendix D: Online Survey Questions



Survey on current UTAR Venue/Classroom Booking System

Good day!

I'm Wong Jia Keen, a Software Engineering student currently working on Final Year Project, which my title is to develop a venue/classroom booking system. Therefore, I would like to seek your assistance on filling this survey to gain your opinions on the current system.

Should you have further questions or clarifications, do contact me via email to jiakeenwong0413@1utar.my or 010-8415946.

Thank you and good luck!

* Indicates required question

1. Email *

2. Acknowledgment of Notice *

   Mark only one oval.

   ◯ I have been notified by you and that I hereby understood, consented and agreed per UTAR above notice

   ◯ I disagree, my personal data will not be processed

Personal Information

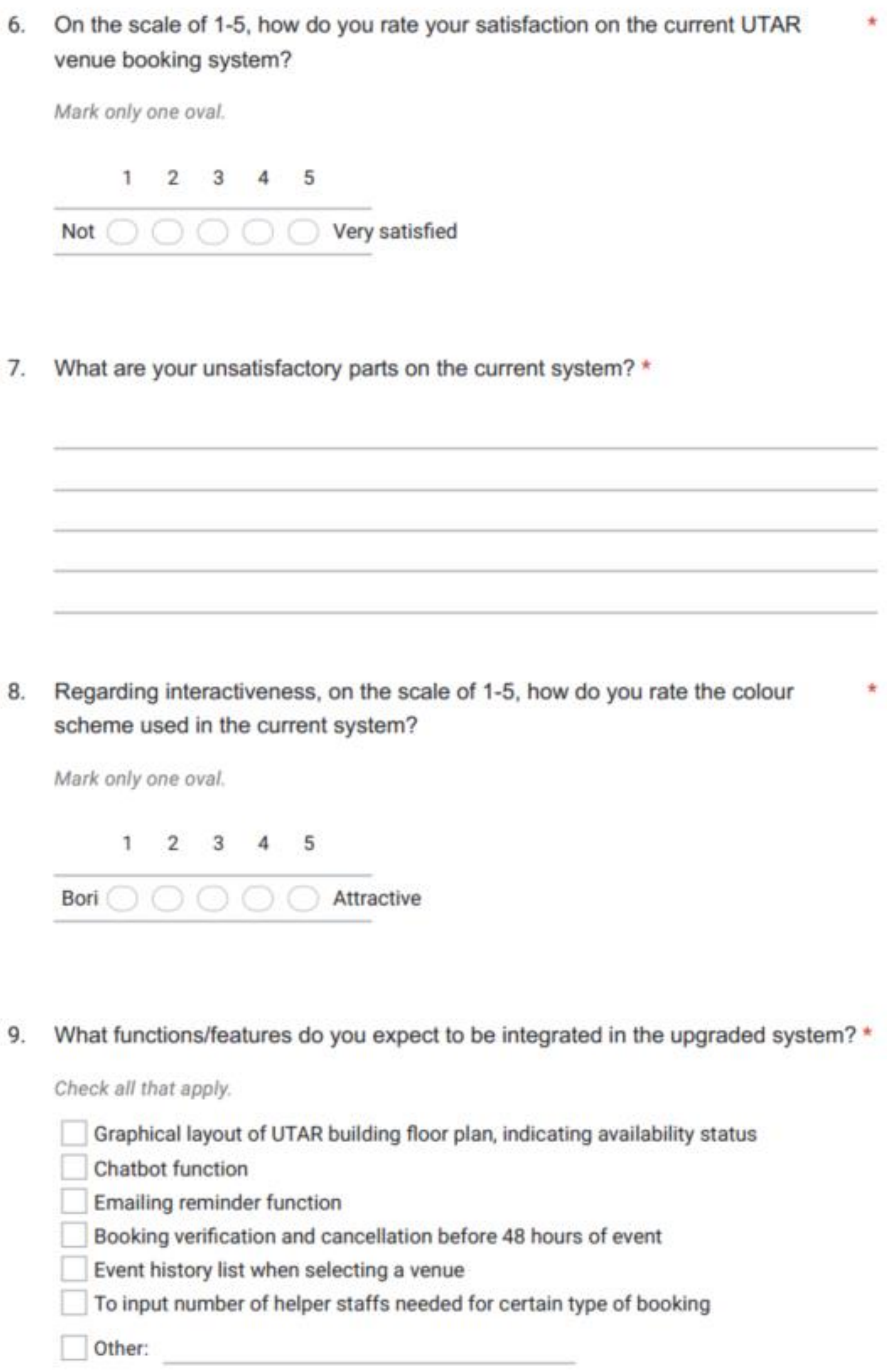
3. Your name: *

4. Your email:

5. Department you are working at: *

   Mark only one oval.

   ◯ LKCFES
   ◯ FMHS
   ◯ FAM
   ◯ FCI
   ◯ Institute of Chinese Studies
   ◯ CFS-Sungai Long
   ◯ CEE-Sungai Long
   ◯ Other:



6. On the scale of 1-5, how do you rate your satisfaction on the current UTAR venue booking system? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very satisfied

7. What are your unsatisfactory parts on the current system? *

8. Regarding interactiveness, on the scale of 1-5, how do you rate the colour scheme used in the current system? *

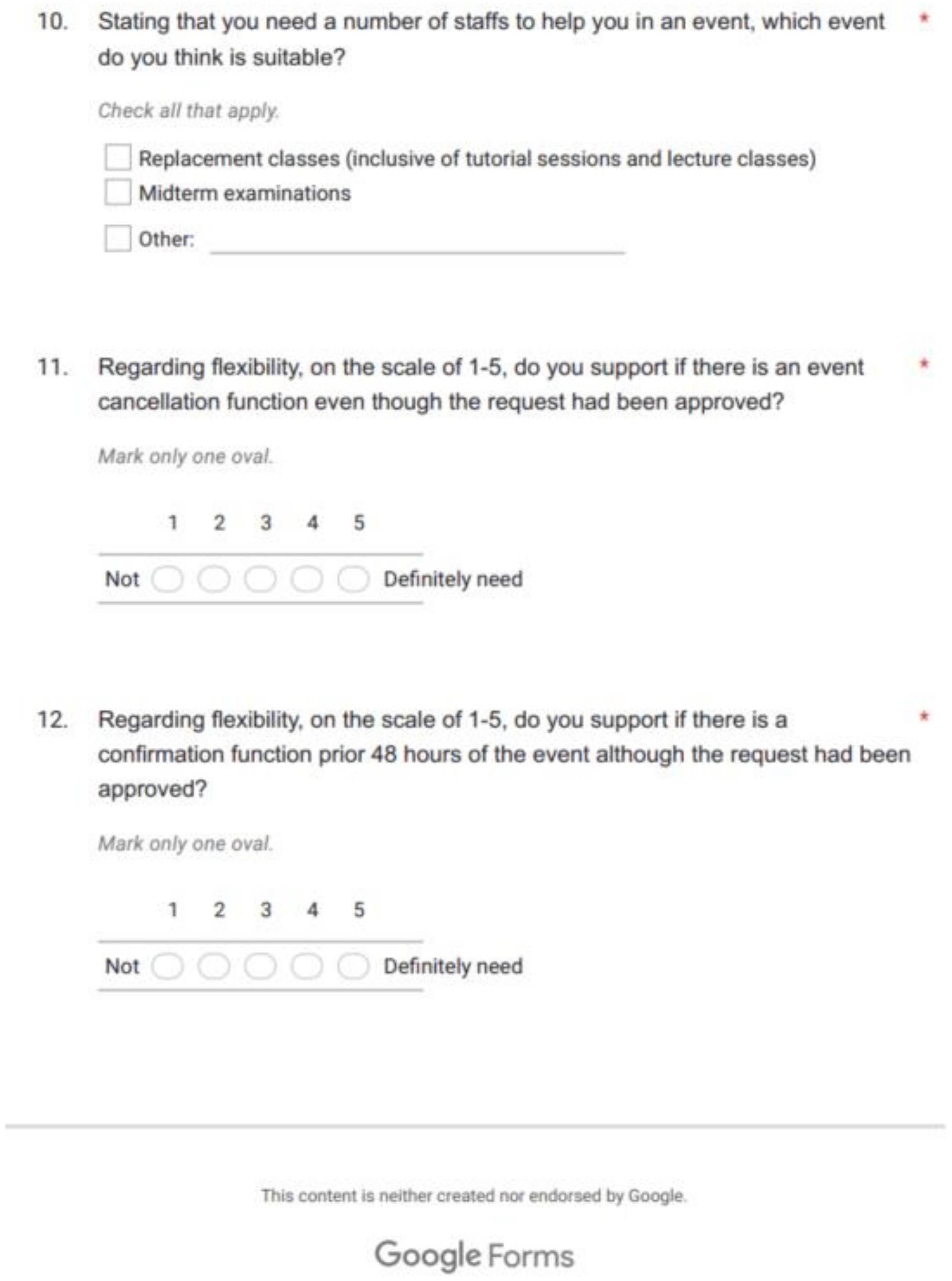
Mark only one oval.

	1	2	3	4	5	
Bori	◯	◯	◯	◯	◯	Attractive

9. What functions/features do you expect to be integrated in the upgraded system? *

Check all that apply.

- [ ] Graphical layout of UTAR building floor plan, indicating availability status
- [ ] Chatbot function
- [ ] Emailing reminder function
- [ ] Booking verification and cancellation before 48 hours of event
- [ ] Event history list when selecting a venue
- [ ] To input number of helper staffs needed for certain type of booking
- [ ] Other: ______________________



10. Stating that you need a number of staffs to help you in an event, which event do you think is suitable? *

Check all that apply.

- ☐ Replacement classes (inclusive of tutorial sessions and lecture classes)
- ☐ Midterm examinations
- ☐ Other: ____________________

11. Regarding flexibility, on the scale of 1-5, do you support if there is an event cancellation function even though the request had been approved? *

Mark only one oval.

	1	2	3	4	5	
Not	○	○	○	○	○	Definitely need

12. Regarding flexibility, on the scale of 1-5, do you support if there is a confirmation function prior 48 hours of the event although the request had been approved? *

Mark only one oval.

	1	2	3	4	5	
Not	○	○	○	○	○	Definitely need

This content is neither created nor endorsed by Google.

Google Forms

Appendix E: Interview Questions and responses with the DGS staff

1. Please describe the current venue booking process on the administrative functions and booking functions. Do mention if there are any specific constraints.

Ans: Lecturers to submit a booking form then get faculty approve, transfer to DGS to further check the availability. If got venue, will approve then notify the requester thru email. Else if the venue is not available, then DGS will suggest another venue. For students, there will be one-day events or throughout the trimester ones. The venue coverage inclusive the lecture theatres, tutorial rooms, Multi-Purpose Hall and meeting rooms.

DSA is managing the clubs and societies event approval, but at the end DSA transfer the venue booking task to DGS as well. Regarding the conflicts, there is no guarantee on no human errors due to possible overlook. However, the department is open for any complaints since they are looking for improvements. The department can accept any booking requests in the form of email or manually, as long as they get their approval from the faculty (lecturer) or DSA (students). Other than prioritizing the importance of the event, most likely the event is approved according the first-come first-serve basis.

Regarding rejection, DGS will check first instead of directly rejecting the request. They will check the proposal first.

2. Do you think the existing system still can cater the normal usage? Is there any dissatisfaction with the existing system and the flow?

Ans: Requested since long ago, since the functionalities can be further improved. It will be better if the request is supervised by another party since there’s case where the DGS received complaints that the venue is not fully utilised or the event gets cancelled without notifying, causing wastage. The email can be sent to the requester and the supervisor as well so that both parties get notified. The reason to be validated is to get a clear understanding on the aim of running an event.

Regarding the further cancellation, it is suggested that the cancellation confirmation duration to be as early as possible. This is because the preparation is time consuming, which the suggested one week could be insufficient for the planned requester to prepare.

3. Regarding user-friendly to use, do you think the existing system has achieved this aspect well? Any suggestions to improve this sector?

Ans: No, cause the venues/classrooms are not viewable and that will be another system. In terms of working process, DGS might need to compare the faculty's class timetable with their own timetable manually to perform venue availability checking.

4. Regarding interactiveness, what do you think regarding the existing system's colour scheme? What colour scheme do you expect for the upgraded system?

Ans: It is better if the activity elements are categorized based on colours, like those events which are noisier can be marked with darker tone colours to indicate that it is better to not have activities surrounding that venue. The colour schemes are better to let ourselves decide.

5. What new features do you expect if there is a customized yet upgraded system?

Ans: Faculty to add the course timetables, then only to let DGS to manage the venue afterwards.

6. In terms of accessibility, do you think mobile access is needed? Since it is targeted for web-based system.

Ans: Yes, since sometimes the staff get contacted in the non-working hours to check the venue availability, therefore it will be better if the system is able to be accessed through mobile devices, reducing the staff's effort to boot the computer to check the arrangement.

Appendix F: Interview Questions with the students

1. Please describe the venue booking process. Do mention if there are any specific constraints.

2. Is there any dissatisfaction with the existing system/workflow?

3. If there is a system specifically developed to cater this function, on a scale of 1-5, how much would you support it?

4. What new features do you expect if there is a customized system?

5. Do you think is it suitable to implement for all UTAR students? Regardless of their status as an executive in societies.

Appendix G: Students' interview responses

Name: Jacky Soo Li Heng

Position: Chairperson

Contact Number: 013-835 8008

Email address: jackyjacky063@gmail.com

1. After confirming the activity to be organized and the venue, the secretary will get the venue application form manually and will submit through email to the Department of Student Affairs (DSA) staff, regardless of the activity scale. The society will get notified through email as well. However when it comes to booking many venues at a same time, then our club advisor will need to take charge of the venue booking task.

   There are times where the booking request gets rejected due to the venue that the Architecture Society wants is unavailable. The society will need to think of another method to solve this issue.

   Regarding the constraints, the DSA staff requires our society to state the expected number of participants to join the event, when it comes to booking university venues for large-scale event. If the number of participants does not match the venue/classroom capacity, then there is a higher chance for the request to get rejected.

2. For the mode of communication, our society prefers to contact the staff physically rather than through email, as the information could be conveyed faster. Normally our society will inform the DGS staff through WhatsApp to schedule an appointment, then the society representative will approach with them and deal with the venue booking matters within a day.

3. Our society is highly supportive of this development, definitely a rating of 5.

4. As long as the features implemented can boost the venue booking efficiency, then our society will support. If to specify one, I think it should be implementing a function for the requesters to see the availability status of all venues/classrooms in UTAR.

5. To my opinion, I think it is better for conducting a survey with all UTAR students since some students are actively engaged in co-curricular activities. I believe most of the students will support this development.

Name: Ong Zi Yang

Position: Chairperson of Wushu Society

Contact Number: 018-573 0590

Email address: ongziyang02@1utar.my

1. Our club had used many venues in UTAR for running our activities, like the lecture theaters, concourse areas, Multi-Purpose Hall and the entrance area of KB building. Our club may need to contact with DSA to book the venues. Our club may skip the process for booking concourse areas and entrance area since only lecture theaters are needed to undergo the process. Our club will state the venue name to DSA upon our request, but the results are not fixed, depending on the availability status of the venue on that activity duration.

Our club deals with the DSA staff physically, submitting the application form on spot.

2. Since our activities are carrying out in the weekly basis, therefore it could be a troublesome for our committee to meet the DSA staff regularly to book the venues.

3. Our club supports the development for this system.

4. Our club hopes that the system to be developed supports the function of long-term booking, since our regular activities will be the weekly trainings in every

weekday. Therefore, our club really hopes this function to be implemented, so that our club does not need to book the venues repetitively.

5. It could be good news for the students with no co-curricular needs, but it could in turn be bad news for us as the society executives, since this will increase the unavailability rate for the venues. So, our society thinks it is better if the students with co-curricular reasons are prioritized than the other students with normal reasons only.

Name: Louis Liew You Jun

Position: Vice Chairperson of Journalism in Chinese Media Society and Taoist Society

Contact Number: 017-852 8614

Email address: gumeng248@gmail.com

1. Since our activities are mostly formal like conducting workshops or talks which need venues that could fit many audiences, our society will enquire the DGS staff regarding the selected venue's availability on the event date, then we only proceed to draft the event proposal. Normally, we would deal this venue booking issue before 2 weeks of the event we planned. Our society committee prefers to communicate with the DGS staff manually, since the information could be delivered in an easier manner.

We would specify our event requirements every time booking, so that the DGS staff could assign another venue that fulfills the same condition as the venue our society initially selected. If our booking requests get rejected, normally the DGS staff would notify us about the unavailability, and we would directly accept the suggested venue since the committee do not have much time to bargain the venue selection matter.

2. Firstly, the society committee need to appoint a time to meet the DGS staff to state our booking request. Additionally, we are unable to select the venue by our own means since the approval is controlled by the DGS staff.

3. Our society would give a rating of 5, since we quite support the development of this system.

4. It would be better if a system is developed that supports the viewing of the availability status of each venue/classroom at which date and the timing, so that it reduces our committee effort to meet the DGS staff every time when it comes to booking a venue for our activity. This also enables our event committee to have sufficient time to replan the event.

Another suggestion is that each venue is supported with descriptive pictures so that we can know the venue layout. This also reduces the society committee's effort to survey on the planned venue and its facilities provided. Speaking about the facilities, it is better to list all facilities available in the venue, which we got cases that the facilities are malfunctioning, causing the event's flow to get disrupted.

5. Not suitable, since some students will misuse it for their own purpose, which could cause inconvenience for other societies that want to book a venue for running their activities.

Name: Teng Yun

Position: Chairperson of Choir Society

Contact Number: 016-647 7498

Email address: tengyun18@gmail.com

1. Our society tends to book the venues at the start of the trimester since the activities are held weekly. Our society will state our venue booking request to the DGS staff through WhatsApp, eventually they will send an email to notify the venue had been booked successfully.

2. There are cases where we found that another party was using the venue that our society had booked earlier. While the party insists that they had booked the venue too, this could cause our activity plans to get disrupted for not having the

right place to carry on. I believe this issue exists because the DSA did not coordinate with the DGS regarding the venue booking status and the availability. This issue happens frequently on last two trimesters, while having lower rate of occurrence for last trimester and this trimester.

There is another problem that some venues were open for use but the DGS states that there is a booking for the venue. Upon our checking, this happens due to the unnotified cancellation of the event to the DGS staff. With this to be said, a venue was wasted due to their neglect.

3. Our society will give a rating of 5.

4. Our society hopes that there will be a system that enables the society committee to view the availability status of each venue/classroom at which timing so that we can directly book the venue in the system instead of communicating with the departmental staff. Another thing is that it is better to state out the facilities provided in each venue/classroom, since our society's activities are about performing and these facilities are important for us. The feature of flexible cancellation upon one week of event is better to include as well.

5. To my opinion, it is fine to open the system to all UTAR students, but to limit some functionalities for them. Perhaps, they could just view the availability status of each venue/classroom, so that they are aware of the status and our society committee does not need to warn them if we found someone who is in the venue which we already booked it since early.

Name: Au Kit Yeng

Position: Head of Public Relations Department

Contact Number: 011-3951 9619

Email address: angel0709yeng@1utar.my

1. Our society's activities are mostly film shooting and requires us to prepare a proposal for the DSA because our context are mostly in the university. Our society will contact the department before 2 weeks of film shooting through email. When it comes to scenarios where the email is not responded more than 2 working days, then our society committee will approach physically. Normally, it will take 2 working days to get our request email replied.

2. To my opinion, I think the frequency of replying to the messages could be improved to be more responsive. It will be better if the communication could be done online since at times our society committee fails to meet them physically.

3. I think it is worthy to give a try, I will give it a rating of 4.

4. Since our activity will be held in both KA and KB buildings, our society hopes that the system will provide a feature that enables us to see the layouts for all venues/classrooms for these two buildings. This feature enables us to decide which venue is more suitable for filming by just referring to the system's pictures, instead of walkthrough both blocks to compare the condition of each venue/classroom.

5. I think the system could only open to those students who are executives from the co-curricular societies will do. It depends on these students' experience to use the system then to decide whether this system is appropriate to open to all UTAR students.

Name: Laurenz Lee

Position: Chairperson of Film Club

Contact Number: 013-369 3808

Email address: laurenzlee@1utar.my

1. For the screening activities since will be held at KB entrance and lecture theaters, our society will send the event proposal to our DSA officer through email just before the start of the trimester. The response duration varies,

depending on which timing our society sent the request, but we tend to sent early since our request could get replied in quick.

Normally our booking request will get responded in 2 days' time, but the situation becomes difficult if we had change on plans. Sometimes our society need to contact the departmental staff immediately through WhatsApp to get assistance from them.

2. There are times where our society faced situations like there are another events running in the venue that we had booked at the same time, which our society need to delay our screening since the particular venue our society booked has the best facilities to support the screening experience.

Another annoying problem is that our society committee needs to refer to our own calendar when it comes to selecting the date for running our activity. As the university has many planned events and the campus might close during public holidays, we need to specifically be aware of these dates and uncertainty arises since at times we are not sure about the university arrangement on these dates available.

3. I would give full 5 pertaining this matter, because I think as the system progresses, it sure will make the venue booking job easier for the departmental staff and the booking requesters.

4. Since our society faced situations like there are clashing of events at the same time in the venue we had booked, I think the system should feature something that shows if the event is still on the schedule or no. Referring to the second problem faced, I think the booking process will be a lot easier if there is an accessible calendar embedded in the system, showing which dates are the public holidays, which dates are preserved for university events and so for.

Another thing to suggest is the system to show the availability status of the venue/classroom.

5. I think it is nice to open this system to all UTAR students, but I believe at the same time there will be issues where I foresee the society committees will have a shortage of venues to book as some students could book a large venue solely for practicing their presentation. If to open to the general population of students, I think they will need to go through a certain party to review their request, like to review the necessity of booking such venue and the feasibility.

Name: Ng Jia Ying

Position: Event Committee of Chinese Language Society

Contact Number: 017-398 9285

Email address: njy2106632@1utar.my

1. Representing the society, I will send requesting email to the DSA. I will need to meet them physically as well when it comes to getting extra equipment from them for running the activity. They tend to respond my request email in 1 day.

2. There are times where our society faced situations like there are another events running in the venue that we had booked at the same time, which our society need to further communicate with them if their event is nearly to an end or no. This is because I am booking the venue under DSA's approval, but another party stated that they get DGS approval too. I believe both departments lack coordination when it comes to event management which relates further to the venue allocation.

Additionally, our event gets rejected but the rejection reason could be vague, which causes confusion.

3. 5.

4. I think it will be better if the departments can coordinate the venue booking issues properly, to prevent clashing of events at first. Technically, both

department staff can be added as the system's user so that both parties can view the timetable and managing the availability status of the venue/classroom.

Our society hopes that the system will provide a feature that enables us to see the layouts for all venues/classrooms for these two buildings. This feature enables us to decide which venue is more suitable for our event by referring to the system's pictures, since the layout design for certain venue/classroom is not standardized.

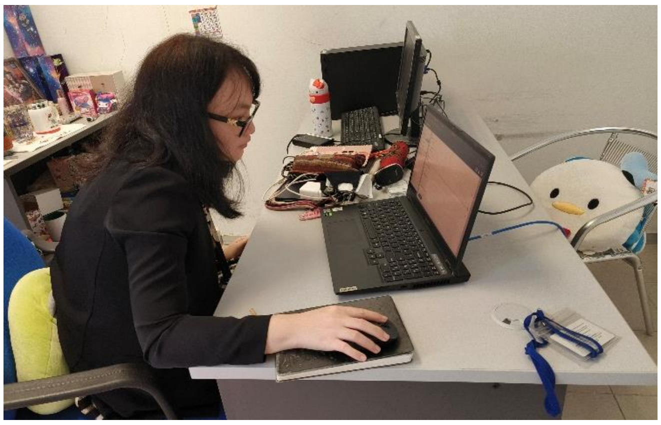
Another feature suggested is to prompt the booking requester whether if the planned event continues or not. Since some of our events may postpone, therefore it is better to confirm with the status of the event so that other society could use the venue. The best timing is to prompt the booking requester before 1 week of the event, so that the succeeding party got enough time to prepare for their event.

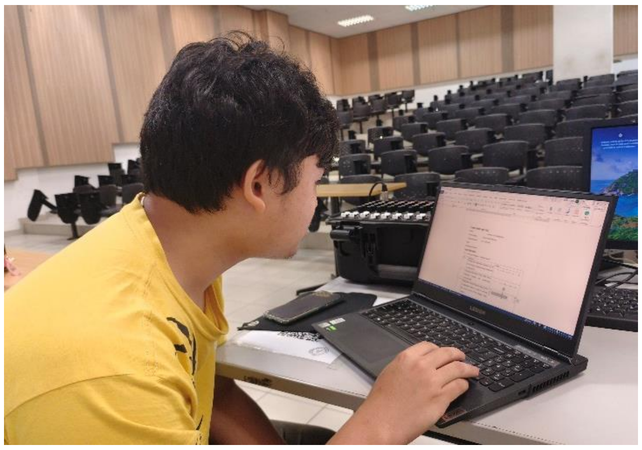
In terms of communication channel, I think to utilize WhatsApp along with email to get both parties informed with the venue booking updates, since some will overlook the email and some may not check their inbox frequently.

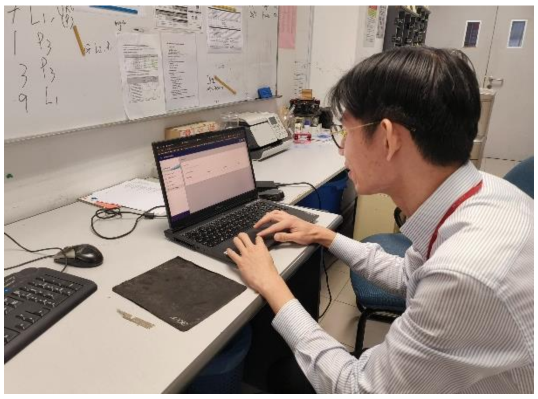
5. I think the situation will be complex if open to all UTAR students. To my opinion, I think what if we segregate the system into both versions, like one is dedicated for the co-curricular use, and one is for the students with non-co-curricular reasons to use, since the method of treating such booking requests will be handled differently.

Else, those students who book for normal usage will need their lecturer's approval for running their own activity. This will be a safer method since the lecturer is aware and takes partly responsible to review and approve the student's booking motive.

Appendix H: Usability Testing with participants







Appendix I: System Usability Scale responses

Name	:	Ker Ding Wei
Email Address	:	2103383@1utar.my
Contact Number	:	0132583426
Role	:	Student
Participant Number	:	1

<u>**Score Distribution**</u>

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1 2 3 4 **5**
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	1 **2** 3 4 5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1 2 3 4 **5**
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	**1** 2 3 4 5
5	I found the notification function of the UTAR Venue Booking System (Email and SMS) really helps in updating me regarding the progress of venue booking request.	1 2 3 **4** 5
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	**1** 2 3 4 5
7	I am confident that the UTAR Venue Booking System could solve my	1 2 3 4 **5**

	venue usage problems with ease when it is urgent.	
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	**1** 2 3 4 5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1 2 3 4 **5**
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	**1** 2 3 4 5

Name	: Dr. Hooi Chee Mei
Email Address	: hooicm@utar.edu.my
Contact Number	: 019-4102732
Role	: Lecturer
Participant Number	: 2

<u>Score Distribution</u>

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1 2 3 **4** 5
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	1 **2** 3 4 5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1 2 3 **4** 5

4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	[✓ 2]	3	4	5
5	I found the notification function of the UTAR Venue Booking System (email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	[✓ 4]	5
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	1	[✓ 2]	3	4	5
7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1	2	3	[✓ 4]	5
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	1	[✓ 2]	3	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	[✓ 4]	5
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	1	[✓ 2]	3	4	5

Name	: Er Pek Hoon Winnie
Email Address	: erph@utar.edu.my
Contact Number	: 012-6771767
Role	: Lecturer
Participant Number	: 3

<u>**Score Distribution**</u>

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score				
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	3	[✓ 4]	5
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	1	[✓ 2]	3	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	[✓ 4]	5
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	[✓ 2]	3	4	5
5	I found the notification function of the UTAR Venue Booking System (email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	[✓ 4]	5
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	1	[✓ 2]	3	4	5
7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1	2	3	[✓ 4]	5
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	1	[✓ 2]	3	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	[✓ 4]	5
10	I found the overall UTAR Venue Booking System is quite confusing to	1	[✓ 2]	3	4	5

	use, which needs certain amount of time to learn, adapt and master it.	

Name	: Teng Yun
Email Address	: tengyun18@1utar.my
Contact Number	: 0166477498
Role	: Student
Participant Number	: 4

Score Distribution

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score				
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	3	**4**	5
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	1	**2**	3	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	**4**	5
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	2	**3**	4	5
5	I found the notification function of the UTAR Venue Booking System (email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	**4**	5
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	**1**	2	3	4	5
7	I am confident that the UTAR Venue Booking System could solve my venue	1	2	3	4	**5**

	usage problems with ease when it is urgent.					
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	1	2	**3**	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	4	**5**
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	1	**2**	3	4	5

Name	:	Ng Hao Xiang
Email Address	:	nhx526@1utar.my
Contact Number	:	0182591396
Role	:	Student
Participant Number	:	5

<u>Score Distribution</u>

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score				
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	3	**4**	5
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	1	2	**3**	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	4	**5**
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	2	**3**	4	5

5	I found the notification function of the UTAR Venue Booking System (email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	4	**5**
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	1	**2**	3	4	5
7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1	2	3	4	**5**
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	1	**2**	3	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	4	**5**
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	1	2	3	**4**	5

Name	: Ting Jen Ching
Email Address	: tingjc@utar.edu.my
Contact Number	: 0168785119
Role	: Lecturer
Participant Number	: 6

<u>Score Distribution</u>

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score				
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	3	4	**5**

2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	**1**	2	3	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	4	**5**
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	**2**	3	4	5
5	I found the notification function of the UTAR Venue Booking System (email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	4	**5**
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	**1**	2	3	4	5
7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1	2	3	4	**5**
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	**1**	2	3	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	4	**5**
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	**1**	2	3	4	5

Name	: Laurenz Lee Zheng Qian
Email Address	: laurenzlee@1utar.my
Contact Number	: 013-3693808

Role : Student

Participant Number : 7

Score Distribution

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score				
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	3	4	**5**
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	**1**	2	3	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	**4**	5
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	2	3	**4**	5
5	I found the notification function of the UTAR Venue Booking System (email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	4	**5**
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	**1**	2	3	4	5
7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1	2	**3**	4	5
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	**1**	2	3	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	4	**5**

10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	1	**2**	3	4	5

Name : Hoo Meei Hao

Email Address : hoomh@utar.edu.my

Contact Number : 012-7044188

Role : Head of Department, DIECS

Participant Number : 8

Score Distribution

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score				
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	**3**	4	5
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	**1**	2	3	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	**4**	5
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	2	**3**	4	5
5	I found the notification function of the UTAR Venue Booking System (Email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	**4**	5
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	1	**2**	3	4	5

7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1	2	3	**4**	5
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	1	**2**	3	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	**4**	5
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	1	2	**3**	4	5

Name	: Yee Sau Keng
Email Address	: yeesk@utar.edu.my
Contact Number	: 012-2879148
Role	: Venue Booking person in charge, DGS
Participant Number	: 9

<u>Score Distribution</u>

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score				
1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	3	**4**	5
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	1	2	**3**	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	**4**	5

4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1 **2** 3 4 5
5	I found the notification function of the UTAR Venue Booking System (Email and SMS) really helps in updating me regarding the progress of venue booking request.	1 2 3 **4** 5
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	1 2 **3** 4 5
7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1 2 **3** 4 5
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	1 **2** 3 4 5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1 2 3 **4** 5
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	1 2 **3** 4 5

Name	: Foo Cher Siang
Email Address	: foocs@utar.edu.my
Contact Number	: +603-9086 0288 (Ext 212)
Role	: Head of DGS
Participant Number	: 10

<u>Score Distribution</u>

(1 – 5, Strongly Disagree – Strongly Agree)

No.	Question	Score

1	I found the information displayed in the UTAR Venue Booking System dashboard is informative.	1	2	3	**4**	5
2	I found the functions of the UTAR Venue Booking System are performing inconsistently.	1	2	**3**	4	5
3	I found the UTAR Venue Booking System is informative when it comes to viewing the venues in the UTAR Sungai Long campus.	1	2	3	**4**	5
4	I found the interface for the UTAR Venue Booking System is dull and unattractive to use.	1	**2**	3	4	5
5	I found the notification function of the UTAR Venue Booking System (Email and SMS) really helps in updating me regarding the progress of venue booking request.	1	2	3	**4**	5
6	I found the information confidentiality of the UTAR Venue Booking System is unreliable.	1	2	**3**	4	5
7	I am confident that the UTAR Venue Booking System could solve my venue usage problems with ease when it is urgent.	1	2	3	**4**	5
8	I found there are not enough functionalities in the UTAR Venue Booking System, and they are insignificant.	1	**2**	3	4	5
9	I would recommend the UTAR Venue Booking System to any UTAR parties.	1	2	3	**4**	5
10	I found the overall UTAR Venue Booking System is quite confusing to use, which needs certain amount of time to learn, adapt and master it.	1	**2**	3	4	5

Appendix J: User Acceptance Testing responses

Participant #1:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 22-08-2024
Name	: Ker Ding Wei

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	**Test Title**	**Expected Result**	**Result**	**Comments**
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	
Profile Management	View Own Profile	User could view their own profile details.	Working	
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	
	Accessing	User could access their own profile page and not accessing other users'.	Working	

Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	Working	
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	
	Report Viewing	User could review their venue usage request in PDF report format.	Working	
Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	
Event Management	View all calendar events	User could check if any events through the calendar function.	Working	
Messaging	Instant Messaging	User could send messages with the management (HOD/DGS/lecturers) when it comes to booking clarification.	Working	
	Check own Messages	User could view the person they had a conversation with.	Working	

Additional Feedback

Please provide any additional comments or feedback on the field below:

Allow the users to view the recent events even though they have not chosen a date.

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff/Student Name	:	Ker Ding Wei
Signature	:	*ker*
Date	:	22 August 2024

Participant #2:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 22 August 2024
Name	: Dr. Hooi Chee Mei

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	Test Title	Expected Result	Result	Comments

Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	working	N/A
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	working	N/A
Profile Management	View Own Profile	User could view their own profile details.	working	N/A
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	working	N/A
	Accessing	User could access their own profile page and not accessing other users'.	working	N/A
Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	working	N/A
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	working	N/A
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	working	N/A
	Report Viewing	User could review their venue usage request in PDF report format.	working	N/A
Venue Viewing	View all Venues in UTAR Sungai	User could view all venues in UTAR Sungai Long campus that are available for booking.	working	N/A

	Long campus			
Event Management	View all calendar events	User could check if any events through the calendar function.	working	N/A
Messaging	Instant Messaging	User could send messages with the management (HOD/DGS/lecturers) when it comes to booking clarification.	Not shown	N/A
	Check own Messages	User could view the person they had a conversation with.	Not shown	N/A

Additional Feedback

Please provide any additional comments or feedback on the field below:

When it is connected to UTAR Wi-Fi, some functions of the system are not working properly.

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff/Student Name	: Dr. Hooi Chee Mei
Signature	: HCM
Date	: 22 August 2024

Participant #3:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 22-08-2024
Name	: Er Pek Hoon, Winnie

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	**Test Title**	**Expected Result**	**Result**	**Comments**
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	N/A
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	N/A
Profile Management	View Own Profile	User could view their own profile details.	Working	N/A
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	N/A
	Accessing	User could access their own profile page and not accessing other users'.	Working	N/A
Venue Booking Request	Add a new Venue	User could submit a new venue usage request.	Working	Will be good if we can see if other

	Booking Request			venues are taken
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	N/A
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	N/A
	Report Viewing	User could review their venue usage request in PDF report format.	Working	N/A
Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	Maybe have a separate tab for us to view
Event Management	View all calendar events	User could check if any events through the calendar function.	Working	N/A
Messaging	Instant Messaging	User could send messages with the management (HOD/DGS/lecturers) when it comes to booking clarification.	Working	N/A
	Check own Messages	User could view the person they had a conversation with.	Working	N/A

Additional Feedback

Please provide any additional comments or feedback on the field below:

Hope the management takes up this system

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff/Student Name	:	Er Pek Hoon, Winnie
Signature	:	Er Pek Hoon, Winnie
Date	:	22 August 2024

Participant #4:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 22 August 2024
Name	: Teng Yun

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	Test Title	Expected Result	Result	Comments
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	

	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	Maybe can show which part is wrong (ie. Username wrong/password wrong)
Profile Management	View Own Profile	User could view their own profile details.	Working	
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	
	Accessing	User could access their own profile page and not accessing other users'.	Working	
Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	Working	
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	
	Report Viewing	User could review their venue usage request in PDF report format.	Working	
Venue Viewing	View all Venues in	User could view all venues in UTAR	Working	

	UTAR Sungai Long campus	Sungai Long campus that are available for booking.		
Event Management	View all calendar events	User could check if any events through the calendar function.	Working	
Messaging	Instant Messaging	User could send messages with the management (HOD/DGS/lecturers) when it comes to booking clarification.	Working	
	Check own Messages	User could view the person they had a conversation with.	Working	

Additional Feedback

Please provide any additional comments or feedback on the field below:

- User needs to refill everything whenever one criterion is omitted

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff/Student Name	:	Teng Yun
Signature	:	Yun
Date	:	22 August 2024

Participant #5:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 22.08.2024
Name	: NG HAO XIANG

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	**Test Title**	**Expected Result**	**Result**	**Comments**
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	
Profile Management	View Own Profile	User could view their own profile details.	Working	
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	
	Accessing	User could access their own profile page and not accessing other users'.	Working	

Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	Working	
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	
	Report Viewing	User could review their venue usage request in PDF report format.	Working	
Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	
Event Management	View all calendar events	User could check if any events through the calendar function.	Working	
Messaging	Instant Messaging	User could send messages with the management (HOD/DGS/lecturers) when it comes to booking clarification.	Working	

	Check own Messages	User could view the person they had a conversation with.	Working	

Additional Feedback

Please provide any additional comments or feedback on the field below:

N/A

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff/Student Name	:	NG HAO XIANG
Signature	:	NG HAO XIANG
Date	:	22.08.2024

Participant #6:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 22-08-2024
Name	: Ting Jen Ching

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	Test Title	Expected Result	Result	Comments

Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	
Profile Management	View Own Profile	User could view their own profile details.	Working	
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	
	Accessing	User could access their own profile page and not accessing other users'.	Working	
Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	Working	
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	
	Report Viewing	User could review their venue usage request in PDF report format.	Working	

Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	Venue viewing within the same tab
Event Management	View all calendar events	User could check if any events through the calendar function.	Working	
Messaging	Instant Messaging	User could send messages with the management (HOD/DGS/lecturers) when it comes to booking clarification.	Working	
	Check own Messages	User could view the person they had a conversation with.	Working	

Additional Feedback

Please provide any additional comments or feedback on the field below:

N/A

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff/Student Name	:	Ting Jen Ching
Signature	:	Ting Jen Ching
Date	:	22 August 2024

Participant #7:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 23-08-2024
Name	: Laurenz Lee Zheng Qian

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	**Test Title**	**Expected Result**	**Result**	**Comments**
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	
Profile Management	View Own Profile	User could view their own profile details.	Working	
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	
	Accessing	User could access their own profile page and not accessing other users'.	Working	
Venue Booking Request	Add a new Venue	User could submit a new venue usage request.	Working	

	Booking Request			
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	
	Report Viewing	User could review their venue usage request in PDF report format.	Working	
Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	I didn't check every single venue but the second floor one worked. I also think a simpler map would be clearer
Event Management	View all calendar events	User could check if any events through the calendar function.	Working	Date button could be clearer
Messaging	Instant Messaging	User could send messages with the management (HOD/DGS/lecturers) when it comes to booking clarification.	Working	
	Check own Messages	User could view the person they had a conversation with.	Working	

Additional Feedback

Please provide any additional comments or feedback on the field below:

I think knowing who rejected your booking and who wrote the notes could be good for clarification later. I also think that the SMS when details of the booking are changed could be clearer on what is changed or give a link to view the changes.

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Student Name	:	Laurenz Lee Zheng Qian
Signature	:	*LAURENZ*
Date	:	23/08/2024

Participant #8:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 23.8.2024
Name	: Hoo Meei Hao

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	**Test Title**	**Expected Result**	**Result**	**Comments**
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	
Profile Management	View Own Profile	User could view their own profile details.	Working	
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	
	Accessing	User could access their own profile page and not accessing other users'.	Working	
Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	Working	
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	

	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	
	Report Viewing	User could review their venue usage request in PDF report format.	Working	
	Request Management	User could approve/reject non-management users' requests if they found it invalid.	Working	
Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	
	Edit venues' information	User could edit the information for all venues in UTAR Sungai Long campus that are available for booking.	Working	
Event Management	View all calendar events	User could view any events through	Working	

		the calendar function.		
	Edit events information	User could edit any events' information.	Working	
	Delete event	User could delete any existing events.	Working	
Messaging	Instant Messaging	User could send messages with the booking requesters when it comes to booking clarification.	Working	
	Check own Messages	User could view the person they had a conversation with.	Working	
Scheduling Management	Add Trimester Schedule	User could add information for a new trimester.	Working	
	Edit Trimester Schedule	User could edit the starting date and ending date for an existing trimester.	Working	
	Delete Trimester Schedule	User could delete an existing trimester.	Working	

Class Management	Add Class info with reserving venue	User could add a class information and reserve a venue that is specifically for that class throughout the entire trimester selected.	Working	
	Edit Class info	User could edit a specific added class time and date.	Working	
	Delete Class Info	User could delete a specific added class for a reserved venue.	Working	

Additional Feedback

Please provide any additional comments or feedback on the field below:

- suggest to input capacity of the room requested to determine the venue available that show in the list of option. - dashboard should be different depends on the role of person. - approval and reject buttons appear in the list need to show significant meaning. This highlight the person to make approval first before the next.

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff Name : Hoo Meei Hao

Signature : [illegible]

Date : 23.08.2024

Participant #9:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project : UTAR Venue/Classroom Booking Web Portal
Conducted Date : 22/08/2024
Name : Yee Sau Keng

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	**Test Title**	**Expected Result**	**Result**	**Comments**
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	N/A
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	N/A

Profile Management	View Own Profile	User could view their own profile details.	Working	N/A
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	N/A
	Accessing	User could access their own profile page and not accessing other users'.	Working	N/A
Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	Working	N/A
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	N/A
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the	Working	N/A

		date they requested.		
	Report Viewing	User could review their venue usage request in PDF report format.	Working	N/A
	Request Management	User could approve/reject non-management users' requests if they found it invalid.	Working	N/A
Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	N/A
	Edit venues' information	User could edit the information for all venues in UTAR Sungai Long campus that are available for booking.	Working	N/A

Event Management	View all calendar events	User could view any events through the calendar function.	Working	N/A
	Edit events information	User could edit any events' information.	Working	N/A
	Delete event	User could delete any existing events.		
Messaging	Instant Messaging	User could send messages with the booking requesters when it comes to booking clarification.	Working	N/A
	Check own Messages	User could view the person they had a conversation with.	Working	N/A
Scheduling Management	Add Trimester Schedule	User could add information	Working	N/A

		for a new trimester.		
	Edit Trimester Schedule	User could edit the starting date and ending date for an existing trimester.	Working	N/A
	Delete Trimester Schedule	User could delete an existing trimester.		
Class Management	Add Class info with reserving venue	User could add a class information and reserve a venue that is specifically for that class throughout the entire trimester selected.	Working	N/A
	Edit Class info	User could edit a specific added class time and date.	Working	N/A
	Delete Class Info	User could delete a specific added class	Working	N/A

		for a reserved venue.		

Additional Feedback

Please provide any additional comments or feedback on the field below:

The online booking system is better than I thought. I can receive application and cancellation notifications from applicants. It is also recommended to add a "remind applicants of the use date" command. This is useful for some applicants who have already made venue reservations a long time ago, but may cancel the event or meeting due to certain factors, and forget to cancel the venue. This is especially for MPH/Conference Room/Meeting Room reservations.

As for the venue management system, it can be convenient for certain factors to block the venue, such as regular class schedule/final exam/MUET exam/replacement & supplementary exam.

The only problem is that I cannot see an overview of all venues (MPH/Conference Room/Meeting Room/Classroom) in the same system. This is because the classroom management DGS must work with Faculties at the same time. At present, DGS reserves all classrooms for Faculties before the start of each new semester, and they will arrange the new class schedules and then put it into Web2. DGS then uses copy and paste from Web2 to put it into Google Spreadsheet. Therefore, DGS refers to Google Spreadsheet to see an overview of all classroom usage, while MPH/Conference Room/Meeting Room refers to Web2.

For student venue online booking, the purpose should be included for Club & Societies OR Assignment Presentation. If it is related to Club & Societies, the student's application also needs to be approved by the relevant DSA officer, and then submitted to the DSA supervisor for approval, and finally to the DSG for approve and venue arrangement. If it is a Student Assignment Presentation, the student's application also needs to be approved by the relevant Faculty's Advisor/Lecture, and then submitted to the Faculty Dean for approval, and finally to the DSG for signature and venue arrangement.

Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff Name	: Yee Sau Keng
Signature	: Yee Sau Keng
Date	: 27/08/2024

Participant #10:

UTAR Venue/Classroom Booking Web Portal – User Acceptance Testing (UAT) Form

Project	: UTAR Venue/Classroom Booking Web Portal
Conducted Date	: 23.8.2024
Name	: Foo Cher Siang

Instructions

Please follow the test cases below and provide your feedback in the 'Comments' section. Indicate whether the test case passed or failed and provide any additional observations.

Legend: HOD- Head of Department; DGS- Department of General Services

Section	**Test Title**	**Expected Result**	**Result**	**Comments**
Login and Authentication	Login with valid credentials	User logs in and is directed to the dashboard.	Working	
	Login with invalid credentials	User is shown with an alert message and is asked to retry.	Working	

Profile Management	View Own Profile	User could view their own profile details.	Working	
	Edit Profile Details	User could edit their contact number, as long as it is not blank.	Working	
	Accessing	User could access their own profile page and not accessing other users'.	Working	
Venue Booking Request	Add a new Venue Booking Request	User could submit a new venue usage request.	Working	
	View Own Booking Request	User could review their venue usage request once it is approved by lecturer/ HOD/ DGS administrator.	Working	
	Cancel Own Booking Request	User could cancel their venue usage request if it is before the date they requested.	Working	It would be useful if user can edit the booking if the booking is yet recommended / approved
	Report Viewing	User could review their	Working	

		venue usage request in PDF report format.		
	Request Management	User could approve/reject non-management users' requests if they found it invalid.	Working	
Venue Viewing	View all Venues in UTAR Sungai Long campus	User could view all venues in UTAR Sungai Long campus that are available for booking.	Working	
	Edit venues' information	User could edit the information for all venues in UTAR Sungai Long campus that are available for booking.	Working	
Event Management	View all calendar events	User could view any events through the calendar function.	Working	
	Edit events information	User could edit any events' information.	Working	
	Delete event	User could delete any existing events.	Working	Requester should receive

				notification if his/her event has been deleted
Messaging	Instant Messaging	User could send messages with the booking requesters when it comes to booking clarification.	Working	
	Check own Messages	User could view the person they had a conversation with.	Working	
Scheduling Management	Add Trimester Schedule	User could add information for a new trimester.	Working	
	Edit Trimester Schedule	User could edit the starting date and ending date for an existing trimester.	Working	
	Delete Trimester Schedule	User could delete an existing trimester.	Working	
Class Management	Add Class info with	User could add a class information	Working	

	reserving venue	and reserve a venue that is specifically for that class throughout the entire trimester selected.		
	Edit Class info	User could edit a specific added class time and date.	Working	
	Delete Class Info	User could delete a specific added class for a reserved venue.	Working	

Additional Feedback

Please provide any additional comments or feedback on the field below:

Overall, the venue booking system is useful and user friendly.

Requester can view the venue availability, select the venue based on the capacity. Requester and DGS can also communicate with each other through the system for clarification

As for DGS, it will be useful if DGS can view the overall venue usage.(refer to attachment)

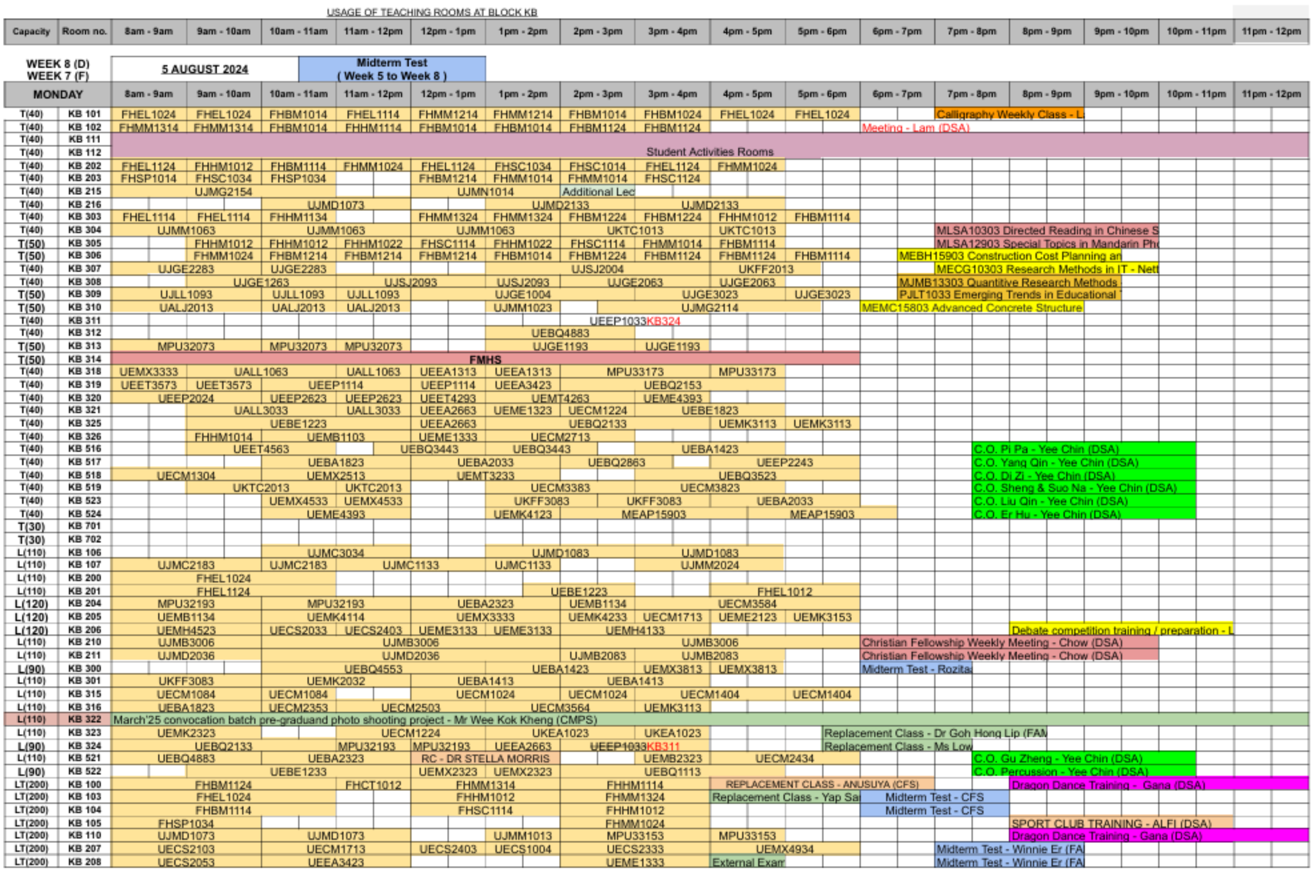
Signature

By signing below, I acknowledge that I have completed the User Acceptance Test for the UTAR Venue/Classroom Booking Web Portal and provided honest feedback based on my usage experience.

Staff Name : Foo Cher Siang

Signature	: [illegible]
Date	: 23.08.2024

APPENDIX K: Attachment from Mr. Foo as a reference for venue usage



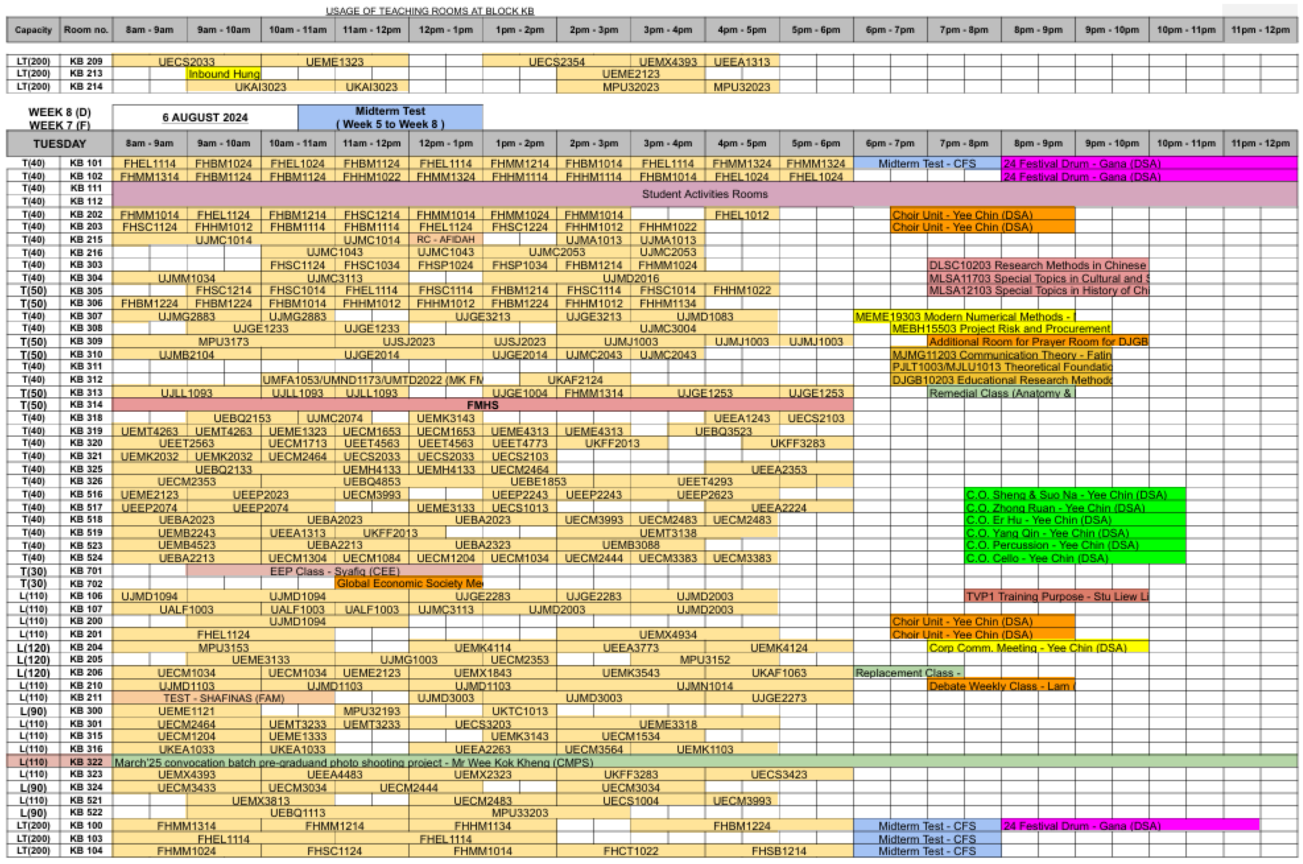
USAGE OF TEACHING ROOMS AT BLOCK KB

WEEK 8 (D)
WEEK 7 (F)

5 AUGUST 2024

Midterm Test
(Week 5 to Week 8)

Capacity	Room no.	8am - 9am	9am - 10am	10am - 11am	11am - 12pm	12pm - 1pm	1pm - 2pm	2pm - 3pm	3pm - 4pm	4pm - 5pm	5pm - 6pm	6pm - 7pm	7pm - 8pm	8pm - 9pm	9pm - 10pm	10pm - 11pm	11pm - 12pm
MONDAY		8am - 9am	9am - 10am	10am - 11am	11am - 12pm	12pm - 1pm	1pm - 2pm	2pm - 3pm	3pm - 4pm	4pm - 5pm	5pm - 6pm	6pm - 7pm	7pm - 8pm	8pm - 9pm	9pm - 10pm	10pm - 11pm	11pm - 12pm
T(40)	KB 101	FHEL1024	FHEL1024	FHBM1014	FHEL1114	FHMM1214	FHMM1214	FHBM1014	FHBM1024	FHEL1024	FHEL1024		Calligraphy Weekly Class - L				
T(40)	KB 102	FHMM1314	FHMM1314	FHBM1014	FHHM1114	FHBM1014	FHBM1014	FHBM1124	FHBM1124			Meeting - Lam (DSA)					
T(40)	KB 111	Student Activities Rooms															
T(40)	KB 112																
T(40)	KB 202	FHEL1124	FHHM1012	FHBM1114	FHMM1024	FHEL1124	FHSC1034	FHSC1014	FHEL1124	FHMM1024							
T(40)	KB 203	FHSP1014	FHSC1034	FHSP1034		FHBM1214	FHMM1014	FHMM1014	FHSC1124								
T(40)	KB 215	UJMG2154				UJMN1014		Additional Lec									
T(40)	KB 216			UJMD1073			UJMD2133		UJMD2133								
T(40)	KB 303	FHEL1114	FHEL1114	FHHM1134		FHMM1324	FHMM1324	FHBM1224	FHBM1224	FHHM1012	FHBM1114						
T(40)	KB 304	UJMM1063		UJMM1063		UJMM1063		UKTC1013		UKTC1013			MLSA10303 Directed Reading in Chinese S				
T(50)	KB 305		FHHM1012	FHHM1012	FHHM1022	FHSC1114	FHHM1022	FHSC1114	FHMM1014	FHBM1114			MLSA12903 Special Topics in Mandarin Pho				
T(50)	KB 306		FHMM1024	FHBM1214	FHBM1214	FHBM1214	FHBM1014	FHBM1224	FHBM1124	FHBM1124	FHBM1114	MEBH15903 Construction Cost Planning an					
T(40)	KB 307	UJGE2283		UJGE2283			UJSJ2004			UKFF2013			MECG10303 Research Methods in IT - Nett				
T(40)	KB 308		UJGE1263		UJSJ2093		UJSJ2093	UJGE2063		UJGE2063		MJMB13303 Quantitive Research Methods					
T(50)	KB 309	UJLL1093		UJLL1093	UJLL1093	UJGE1004			UJGE3023		UJGE3023	PJLT1033 Emerging Trends in Educational					
T(50)	KB 310	UALJ2013		UALJ2013	UALJ2013		UJMM1023	UJMG2114				MEMC15803 Advanced Concrete Structure					
T(40)	KB 311							UEEP1033KB324									
T(40)	KB 312						UEBQ4883										
T(50)	KB 313	MPU32073		MPU32073	MPU32073		UJGE1193		UJGE1193								
T(50)	KB 314	FMHS															
T(40)	KB 318	UEMX3333	UALL1063		UALL1063	UEEA1313	UEEA1313	MPU33173		MPU33173							
T(40)	KB 319	UEET3573	UEET3573	UEEP1114		UEEP1114	UEEA3423	UEBQ2153									
T(40)	KB 320	UEEP2024		UEEP2623	UEEP2623	UEET4293	UEMT4263		UEME4393								
T(40)	KB 321		UALL3033		UALL3033	UEEA2663	UEME1323	UECM1224	UEBE1823								
T(40)	KB 325		UEBE1223			UEEA2663	UEBQ2133			UEMK3113	UEMK3113						
T(40)	KB 326		FHHM1014	UEMB1103		UEME1333	UECM2713										
T(40)	KB 516		UEET4563		UEBQ3443		UEBQ3443		UEBA1423				C.O. Pi Pa - Yee Chin (DSA)				
T(40)	KB 517			UEBA1823		UEBA2033		UEBQ2863		UEEP2243			C.O. Yang Qin - Yee Chin (DSA)				
T(40)	KB 518	UECM1304		UEMX2513		UEMT3233			UEBQ3523				C.O. Di Zi - Yee Chin (DSA)				
T(40)	KB 519		UKTC2013		UKTC2013		UECM3383		UECM3823				C.O. Sheng & Suo Na - Yee Chin (DSA)				
T(40)	KB 523			UEMX4533	UEMX4533		UKFF3083	UKFF3083		UEBA2033			C.O. Liu Qin - Yee Chin (DSA)				
T(40)	KB 524			UEME4393			UEMK4123	MEAP15903		MEAP15903			C.O. Er Hu - Yee Chin (DSA)				
T(30)	KB 701																
T(30)	KB 702																
L(110)	KB 106			UJMC3034			UJMD1083		UJMD1083								
L(110)	KB 107	UJMC2183		UJMC2183	UJMC1133		UJMC1133		UJMM2024								
L(110)	KB 200	FHEL1024															
L(110)	KB 201	FHEL1124					UEBE1223			FHEL1012							
L(120)	KB 204	MPU32193		MPU32193		UEBA2323		UEMB1134	UECM3584								
L(120)	KB 205	UEMB1134		UEMK4114		UEMX3333		UEMK4233	UECM1713	UEME2123	UEMK3153						
L(120)	KB 206	UEMH4523		UECS2033	UECS2403	UEME3133	UEME3133	UEMH4133						Debate competition training / preparation - L			
L(110)	KB 210	UJMB3006		UJMB3006				UJMB3006				Christian Fellowship Weekly Meeting - Chow (DSA)					
L(110)	KB 211	UJMD2036		UJMD2036				UJMB2083	UJMB2083			Christian Fellowship Weekly Meeting - Chow (DSA)					
L(90)	KB 300			UEBQ4553			UEBA1423		UEMX3813	UEMX3813		Midterm Test - Rozita					
L(110)	KB 301	UKFF3083		UEMK2032		UEBA1413		UEBA1413									
L(110)	KB 315	UECM1084		UECM1084		UECM1024		UECM1024	UECM1404		UECM1404						
L(110)	KB 316	UEBA1823		UECM2353	UECM2503		UECM3564		UEMK3113								
L(110)	KB 322	March'25 convocation batch pre-graduand photo shooting project - Mr Wee Kok Kheng (CMPS)															
L(110)	KB 323	UEMK2323			UECM1224		UKEA1023		UKEA1023		Replacement Class - Dr Goh Hong Lip (FAM						
L(90)	KB 324	UEBQ2133			MPU32193	MPU32193	UEEA2663	~~UEEP1033~~KB311			Replacement Class - Ms Low						
L(110)	KB 521	UEBQ4883		UEBA2323		RC - DR STELLA MORRIS			UEMB2323	UECM2434			C.O. Gu Zheng - Yee Chin (DSA)				
L(90)	KB 522		UEBE1233			UEMX2323	UEMX2323	UEBQ1113					C.O. Percussion - Yee Chin (DSA)				
LT(200)	KB 100	FHBM1124			FHCT1012	FHMM1314		FHHM1114		REPLACEMENT CLASS - ANUSUYA (CFS)				Dragon Dance Training - Gana (DSA)			
LT(200)	KB 103	FHEL1024				FHHM1012		FHMM1324		Replacement Class - Yap Sa		Midterm Test - CFS					
LT(200)	KB 104	FHBM1114				FHSC1114		FHHM1012				Midterm Test - CFS					
LT(200)	KB 105	FHSP1034						FHMM1024						SPORT CLUB TRAINING - ALFI (DSA)			
LT(200)	KB 110	UJMD1073		UJMD1073			UJMM1013	MPU33153		MPU33153				Dragon Dance Training - Gana (DSA)			
LT(200)	KB 207	UECS2103		UECM1713		UECS2403	UECS1004	UECS2333		UEMX4934			Midterm Test - Winnie Er (FA				
LT(200)	KB 208	UECS2053		UEEA3423				UEME1333		External Exam			Midterm Test - Winnie Er (FA				



USAGE OF TEACHING ROOMS AT BLOCK KB

Capacity	Room no.	8am - 9am	9am - 10am	10am - 11am	11am - 12pm	12pm - 1pm	1pm - 2pm	2pm - 3pm	3pm - 4pm	4pm - 5pm	5pm - 6pm	6pm - 7pm	7pm - 8pm	8pm - 9pm	9pm - 10pm	10pm - 11pm	11pm - 12pm
LT(200)	KB 209	UECS2033		UEME1323			UECS2354		UEMX4393	UEEA1313							
LT(200)	KB 213		Inbound Hung					UEME2123									
LT(200)	KB 214		UKAI3023		UKAI3023			MPU32023		MPU32023							

WEEK 8 (D)
WEEK 7 (F)

6 AUGUST 2024

Midterm Test
(Week 5 to Week 8)

TUESDAY		8am - 9am	9am - 10am	10am - 11am	11am - 12pm	12pm - 1pm	1pm - 2pm	2pm - 3pm	3pm - 4pm	4pm - 5pm	5pm - 6pm	6pm - 7pm	7pm - 8pm	8pm - 9pm	9pm - 10pm	10pm - 11pm	11pm - 12pm
T(40)	KB 101	FHEL1114	FHBM1024	FHEL1024	FHBM1124	FHEL1114	FHMM1214	FHBM1014	FHEL1114	FHMM1324	FHMM1324	Midterm Test - CFS		24 Festival Drum - Gana (DSA)			
T(40)	KB 102	FHMM1314	FHBM1124	FHBM1124	FHHM1022	FHMM1324	FHHM1114	FHHM1114	FHBM1014	FHEL1024	FHEL1024			24 Festival Drum - Gana (DSA)			
T(40)	KB 111	Student Activities Rooms															
T(40)	KB 112																
T(40)	KB 202	FHMM1014	FHEL1124	FHBM1214	FHSC1214	FHMM1014	FHMM1024	FHMM1014		FHEL1012		Choir Unit - Yee Chin (DSA)					
T(40)	KB 203	FHSC1124	FHHM1012	FHBM1114	FHBM1114	FHEL1124	FHSC1224	FHHM1012	FHHM1022			Choir Unit - Yee Chin (DSA)					
T(40)	KB 215	UJMC1014			UJMC1014	RC - AFIDAH		UJMA1013	UJMA1013								
T(40)	KB 216			UJMC1043		UJMC1043	UJMC2053		UJMC2053								
T(40)	KB 303			FHSC1124	FHSC1034	FHSP1024	FHSP1034	FHBM1214	FHMM1024								
T(40)	KB 304	UJMM1034		UJMC3113			UJMD2016						DLSC10203 Research Methods in Chinese				
T(50)	KB 305		FHSC1214	FHSC1014	FHEL1114	FHSC1114	FHBM1214	FHSC1114	FHSC1014	FHHM1022			MLSA11703 Special Topics in Cultural and S				
T(50)	KB 306	FHBM1224	FHBM1224	FHBM1014	FHHM1012	FHHM1012	FHBM1224	FHHM1012	FHHM1134				MLSA12103 Special Topics in History of Chi				
T(40)	KB 307	UJMG2883		UJMG2883		UJGE3213		UJGE3213	UJMD1083			MEME19303 Modern Numerical Methods - I					
T(40)	KB 308		UJGE1233		UJGE1233			UJMC3004				MEBH15503 Project Risk and Procurement					
T(50)	KB 309	MPU3173			UJSJ2023		UJSJ2023	UJMJ1003		UJMJ1003	UJMJ1003		Additional Room for Prayer Room for DJGB				
T(50)	KB 310	UJMB2104		UJGE2014			UJGE2014	UJMC2043	UJMC2043			MJMG11203 Communication Theory - Fatin					
T(40)	KB 311											PJLT1003/MJLU1013 Theoretical Foundatio					
T(40)	KB 312			UMFA1053/UMND1173/UMTD2022 (MK FM			UKAF2124					DJGB10203 Educational Research Methodo					
T(50)	KB 313	UJLL1093		UJLL1093	UJLL1093		UJGE1004	FHMM1314	UJGE1253		UJGE1253		Remedial Class (Anatomy & I				
T(50)	KB 314	FMHS															
T(40)	KB 318		UEBQ2153	UJMC2074		UEMK3143				UEEA1243	UECS2103						
T(40)	KB 319	UEMT4263	UEMT4263	UEME1323	UECM1653	UECM1653	UEME4313	UEME4313	UEBQ3523								
T(40)	KB 320	UEET2563		UECM1713	UEET4563	UEET4563	UEET4773	UKFF2013		UKFF3283							
T(40)	KB 321	UEMK2032	UEMK2032	UECM2464	UECS2033	UECS2033	UECS2103										
T(40)	KB 325	UEBQ2133			UEMH4133	UEMH4133	UECM2464			UEEA2353							
T(40)	KB 326	UECM2353		UEBQ4853			UEBE1853		UEET4293								
T(40)	KB 516	UEME2123	UEEP2023		UECM3993		UEEP2243	UEEP2243	UEEP2623				C.O. Sheng & Suo Na - Yee Chin (DSA)				
T(40)	KB 517	UEEP2074	UEEP2074			UEME3133	UECS1013			UEEA2224			C.O. Zhong Ruan - Yee Chin (DSA)				
T(40)	KB 518	UEBA2023		UEBA2023		UEBA2023		UECM3993	UECM2483	UECM2483			C.O. Er Hu - Yee Chin (DSA)				
T(40)	KB 519	UEMB2243		UEEA1313	UKFF2013			UEMT3138					C.O. Yang Qin - Yee Chin (DSA)				
T(40)	KB 523	UEMB4523		UEBA2213		UEBA2323		UEMB3088					C.O. Percussion - Yee Chin (DSA)				
T(40)	KB 524	UEBA2213		UECM1304	UECM1084	UECM1204	UECM1034	UECM2444	UECM3383	UECM3383			C.O. Cello - Yee Chin (DSA)				
T(30)	KB 701		EEP Class - Syafiq (CEE)														
T(30)	KB 702				Global Economic Society Me												
L(110)	KB 106	UJMD1094	UJMD1094			UJGE2283		UJGE2283	UJMD2003				TVP1 Training Purpose - Stu Liew Li				
L(110)	KB 107	UALF1003		UALF1003	UALF1003	UJMC3113	UJMD2003		UJMD2003								
L(110)	KB 200		UJMD1094									Choir Unit - Yee Chin (DSA)					
L(110)	KB 201	FHEL1124						UEMX4934				Choir Unit - Yee Chin (DSA)					
L(120)	KB 204	MPU3153				UEMK4114		UEEA3773		UEMK4124			Corp Comm. Meeting - Yee Chin (DSA)				
L(120)	KB 205		UEME3133		UJMG1003		UECM2353		MPU3152								
L(120)	KB 206	UECM1034		UECM1034	UEME2123	UEMX1843		UEMK3543		UKAF1063		Replacement Class -					
L(110)	KB 210	UJMD1103		UJMD1103		UJMD1103		UJMN1014					Debate Weekly Class - Lam (				
L(110)	KB 211	TEST - SHAFINAS (FAM)				UJMD3003	UJMD3003			UJGE2273							
L(90)	KB 300	UEME1121			MPU32193		UKTC1013										
L(110)	KB 301	UECM2464		UEMT3233	UEMT3233	UECS3203		UEME3318									
L(110)	KB 315	UECM1204		UEME1333			UEMK3143	UECM1534									
L(110)	KB 316	UKEA1033		UKEA1033		UEEA2263		UECM3564	UEMK1103								
L(110)	KB 322	March'25 convocation batch pre-graduand photo shooting project - Mr Wee Kok Kheng (CMPS)															
L(110)	KB 323	UEMX4393		UEEA4483		UEMX2323		UKFF3283		UECS3423							
L(90)	KB 324	UECM3433		UECM3034	UECM2444			UECM3034									
L(110)	KB 521		UEMX3813			UECM2483		UECS1004		UECM3993							
L(90)	KB 522		UEBQ1113			MPU33203											
LT(200)	KB 100	FHMM1314		FHMM1214		FHHM1134			FHBM1224			Midterm Test - CFS		24 Festival Drum - Gana (DSA)			
LT(200)	KB 103	FHEL1114			FHEL1114							Midterm Test - CFS					
LT(200)	KB 104	FHMM1024		FHSC1124		FHMM1014		FHCT1022		FHSB1214		Midterm Test - CFS					