SMART STUDENT TIMETABLE PLANNER

BY

WONG XIN TONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my FYP supervisor, Dr Tan Joi San, and my FYP moderator, Ts Dr Ku Chin Soon, for their invaluable guidance, constructive feedback, and continuous support throughout the development of Smart Student Timetable Planner. Their expertise and encouragement have been instrumental in the successful completion of this project. I would also like to extend my gratitude to the lecturers and staff of UTAR for providing the necessary resources and facilities that greatly assisted my work. Lastly, I am sincerely thankful to my family and friends for their unwavering support, understanding, and motivation during the FYP.

ABSTRACT

Timetable planning is a crucial yet challenging task for university students, as traditional manual methods are often time-consuming, prone to errors, and lack collaborative support. Students frequently face difficulties in avoiding timetable clashes, managing personal preferences, and coordinating with peers, which can lead to inefficiencies and added stress. To address these issues, this project introduces the Smart Student Timetable Planner, a system developed to streamline academic scheduling by providing both automated and manual timetable management options. The objectives of this project are to generate conflict-free and customizable schedules, enable real-time collaboration among students, and offer administrative tools for maintaining course information. The project scope encompasses features such as secure login, course selection with conflict detection, timetable history, comparison between auto-generated and manual schedules, collaboration modules, and export functionality. To achieve these objectives, the system adopts the Rapid Application Development (RAD) methodology, ensuring iterative design, prototyping, and user feedback integration throughout the process. The system is implemented using Node.js with Express for server-side development, HTML, CSS, and JavaScript for the frontend, and Socket.IO for real-time collaboration. Course data is managed in CSV format, parsed into JSON for fast processing, while sessionStorage and localStorage handle user data within active sessions. A Genetic Algorithm forms the core scheduling engine, generating optimized timetables that respect both hard constraints, such as avoiding clashes, and soft constraints, such as personal preferences.The final output of this project is a functional web-based timetable planner that successfully enhances scheduling efficiency, reduces the likelihood of errors, and improves the overall academic planning experience. With its flexible design, collaborative features, and administrative integration, the Smart Student Timetable Planner demonstrates significant potential as a scalable solution for modern university scheduling needs.

Area of Study (Minimum 1 and Maximum 2): Genetic Algorithm, Web-based application

Keywords (Minimum 5 and Maximum 10): Smart Timetable Planner, Academic Scheduling, Real-Time Collaboration, Conflict-Free Timetable, Student-Centered System, Automated Scheduling.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GAs	Genetic Algorithms
UCT	University Course Timetable
SHO	Spotted Hyena Optimizer
SA	Simulated Annealing
UniTime	University Timetabling System
TTAP	Timetable Arranging Program
GUI	Graphic User Interface
RAD	Rapid Application Development

Chapter 1
Introduction

In this chapter, the project presents the background and motivation of the work, the contributions to the field, and the outline of the project. This project aims to develop an effective Smart Student Timetable Planner that enables students to arrange their own personal timetables according to the courses and timeslots offered by the university. Unlike a static timetable, this planner allows students to select and organize their classes in a way that best fits their preferences and constraints, while still adhering to the university's assigned classrooms and available timeslots.

The Smart Student Timetable Planner is designed to support students in building their academic schedules by ensuring that selected courses do not overlap in time and that all required sessions are included. The main objective is to provide students with a clear and manageable timetable that reflects their chosen courses without conflicts, helping them attend all necessary lectures, tutorials, and practicals within the semester. This approach shifts the focus to the student's individual planning process, where convenience and flexibility play an important role in timetable management.

Timetable planning is an essential part of a student's academic journey, as every semester presents different course offerings and unique scheduling constraints. Traditionally, students rely on manual methods, such as checking course lists and cross-referencing timeslots, which can be time-consuming and prone to mistakes. By introducing a smart planner, the process becomes more efficient, reducing the likelihood of clashes and saving students from the stress of repeated rescheduling.

The project highlights the importance of a Smart Student Timetable Planner as a personal academic management tool. By leveraging automated features and conflict-checking mechanisms, the planner enhances the scheduling process and helps students' overall university experience by ensuring clarity, convenience, and grater satisfaction in managing their study schedules.

1.1 Problem Statement and Motivation

I Difficulty in Course Selection and Scheduling

Students often face significant challenges in selecting and scheduling their courses for a new academic term. The process of manually arranging a timetable that fits all required courses, elective options, and personal commitments can be overwhelming. Students struggle to avoid conflicts, such as overlapping class times or courses scheduled back-to-back in distant locations, which can make it impossible to attend all desired classes. These difficulties can lead to suboptimal course selections, delays in academic progress, and increased stress as students try to balance their academic and personal lives.

II Lack of Real-Time Collaboration in Schedule Planning

Students often face difficulties when trying to collaborate with peers, advisors, or faculty members in real-time while planning their schedules. The current system typically does not support real-time updates and collaborative decision-making, which are essential for group projects, study groups, or coordinated course selections. This lack of collaboration tools can lead to miscommunication, scheduling conflicts, and missed opportunities for joint learning experiences. As a result, students may find it challenging to align their schedules with those of their peers, hindering group work and reducing the overall effectiveness of their academic planning.

The motivation behind addressing the lack of real-time collaboration in schedule planning and the difficulties in course selection stems from the need to improve students' academic planning efficiency and effectiveness. Traditional scheduling systems often fall short in providing dynamic and collaborative tools, leading to scheduling conflicts, inefficiencies, and academic delays. By developing a system that supports real-time collaboration, students will be able to coordinate schedules with peers, advisors, and faculty, reducing conflicts and enhancing collaborative learning. Additionally, the introduction of a Smart Student Timetable Planner will offer personalized recommendations and flexible scheduling options, helping students avoid overlapping classes and manage their workloads more effectively. Integration with existing university systems will ensure consistency and streamline academic management, ultimately contributing to a more user-centered, efficient, and stress-free scheduling experience for students.

1.2 Research Objectives

- **To develop a tool that streamlines course selection and resolves scheduling conflicts for students.**

  The system will incorporate features for automatic conflict detection and resolution, identifying issues such as overlapping class times or class scheduled in distant locations and providing solutions to avoid these conflicts. It will offer dynamic scheduling options, allowing students to easily modify their course selections and adjust their schedules as needed. A visual timetable representation will be included in present schedules intuitively. These enhancements aim to reduce scheduling challenges, lower stress levels, and improve overall academic efficiency.

- **To facilitate real-time collaboration and communication among users.**

  Real-time collaboration features will allow multiple users to work on the timetable concurrently, with changes being instantly visible to all collaborators. This will enhance teamwork and ensure that all modifications are synchronized. Real-time communication will play a crucial role in keeping all users in the same session updated about schedule changes, new bookings, or cancellations, thereby reducing miscommunications and enhancing overall efficiency.

1.3 Project Scope and Direction

The project aims to develop a comprehensive Smart Student Timetable Planner with the following key aspects:

I. Login and Logout Module

This module provides secure access to the timetable system, where students log in using their credentials. It manages authentication, ensuring that only authorized users can access the platform. Personalized features such as saved timetables, course preferences, and collaboration sessions are tied to the student's profile.

II. Course Selection and Scheduling Module

Students can select their intake, trimester, and courses for the semester. The system assists in building schedules by:

- Automatically detecting and preventing class conflicts.
- Allowing students to set personal time constraints.
- Enabling the selection of preferred class groups and timeslots when multiple options are available.

III. View Timetable Module

This module enables students to view their finalized timetable after scheduling. The timetable will show the details of the courses such as the course code, course name, days, time, room allocated and preferred groups according to students' preferences.

IV. Real-Time Collaboration Module

This feature enables students to collaborate with peers while planning schedules. It supports:

- Real-time updates when timetable changes occur.
- Shared editing of timetable slots in a collaborative workspace.
- Built-in chat for discussion and decision-making.

This module is particularly useful for group project planning or ensuring friends share compatible schedules.

V. Save and History Module

Students can save their timetables into a history log for later reference. This allows them to revisit previous scheduling attempts, compare alternatives, and restore timetables without redoing the process.

VI. Comparison Module

This module enables students to compare manually created timetables with automatically generated ones. By presenting both side-by-side, students can evaluate the efficiency, conflict resolution, and flexibility of different scheduling approaches.

VII. Export and Feedback Module

The system allows students to export their finalized timetables in IMG or CSV formats for offline access and sharing. In addition, a feedback feature is included where users can provide comments and suggestions. This input supports continuous system improvement and bug fixing.

VIII. Administrative Module

Administrators are responsible for maintaining up-to-date course information each semester. They can:

- Upload new course schedules and classroom allocations.
- Update the timeslot or make corrections when changes occur.
- Review student feedback to enhance system reliability and address issues promptly.

1.4 Contributions

The Smart Student Timetable Planner contributes by developing an intelligent platform that assists students in generating valid and personalized course schedules based on selected courses and individual time constraints. By integrating a Genetic Algorithm, the system optimizes the generation timetable to avoid clashes and maximize scheduling flexibility. It enhances the course registration experience through a user-friendly interface, real-time data handling, and storage using sessionStorage. Additionally, it offers both manual and automatic scheduling modes to suit different user preferences. This project also establishes a strong foundation for future enhancements such as exporting timetable, improving students' academic planning efficiency.

1.5 Report Organization

This report is organized into seven chapters. Chapter 1 provides an introduction and background to the Smart Student Timetable Planner project. Chapter 2 presents a literature review, discussing similar projects and existing systems to highlight their strengths and limitations. Chapter 3 describes the system methodology and approaches, explaining the methodology used and presenting the system design through activity diagrams, flowcharts, use-case diagrams, and detailed use-case descriptions. Chapter 4 explains the system design in

greater detail, including program development aspects. Chapter 5 focuses on system implementation, covering the hardware and software setup, user interface design with screenshots, and the issues and challenges encountered during implementation. Chapter 6 presents system evaluation and discussion, including the experimental results of the GA and a comparison between the generated timetables and the real-world timetables. Finally, Chapter 7 concludes the report by summarizing the project achievements and outlining directions for future work.

Chapter 2
Literature Review

In this chapter, similar projects and existing systems related to Smart Student Timetable Planner are reviewed. The reviews aim to provide an overview of the objectives behind these systems, the development processes involved, and to identify their achievements, limitations, and suggestions for future enhancements. Additionally, existing solutions are analyzed, summarizing their strengths and weaknesses for comparative purposes, thereby highlighting opportunities for improvement and innovation in the field of course timetabling.

2.1 Similar Projects

2.1.1 Personal Course Timetabling for University Students based on Genetic Algorithm

In this paper, the author presents a solution to the Personal University Course Timetabling (PUCT) problem, which is framed as an individual-oriented variation of the more widely studied course timetable problem. Unlike institutional scheduling, which must consider room capacities, lecturer availability, and global optimization across an entire university, PUCT focuses on generating a feasible and optimized timetable for a single student. The system accepts as input the official university course catalog, where times and course offerings are predetermined, and produces personalized timetables that satisfy both academic requirements and student preferences. The core objective is to provide students with a smart timetable planner that relieves them of the manual burden of arranging courses and ensures that their chosen schedule is both conflict-free and academically sound. By applying a genetic algorithm, the author demonstrates how evolutionary computations can effectively search the large solution space of possible course combinations, yielding timetables that align with students' curricular progression and workload preferences.

The system is designed around a Genetic Algorithm (GA), chosen for its suitability in solving combinatorial optimization problems such as timetabling. Each candidate timetable is encoded as a binary chromosome, where each gene corresponds to a course offering from the catalog. A gene value of 1 signifies inclusion of the course in the student's timetable, while 0 indicates

exclusion. This representation is straightforward yet effective, as the catalog already specifies the day and time of each course, allowing the system to directly detect conflicts between selected courses. The GA employs classical operators including tournament selection, n-point crossover, bit-flip mutation, and elitism to evolve populations of candidate solutions. The fitness function evaluates each timetable based on both hard constraints, such as avoiding overlapping class times, meeting minimum and maximum credit requirements, and not exceeding weekly workload; and soft constraints, such as adhering to suggested curricular progression and allowing students to advance in preferred courses. By adjusting the weights assigned to these criteria, the system can balance feasibility with personalized optimization, ensuring that students receive timetables that are both valid and tailored to their academic needs.

A significant strength of this project lies in its student-centric approach. By narrowing the focus to individual timetables rather than institutional allocation, the system provides immediate practical utility to students seeking efficient ways to plan their courses. The use of binary encoding ensures computational efficiency, enabling the GA to scale to catalogs with many offerings while remaining simple enough to be implemented in student-facing applications. Another notable strength is the flexibility of the fitness function, which allows for the incorporation of both institutional academic requirements and subjective student preferences. Furthermore, empirical evaluation with 25 students demonstrated high levels of satisfaction, with the majority rating the automatically generated timetables as superior to or least equivalent to manually designed schedules.

Despite these contributions, the project is not without limitations. The binary encoding assumes that each course offers a fixed and unique timeslot, which restricts flexibility when courses provide multiple tutorial or laboratory sections. The dataset used for evaluation was relatively small and drawn from a single academic program, which raises concerns regarding the generalizability of results to larger and more complex institutions. Additionally, the weighted-sum approach adopted for multi-criteria optimization, while straightforward, can observe trade-offs between competing objectives; a Pareto-based evolutionary algorithm might offer students a richer set of timetable options that balance different preferences. Finally, the system does not explicitly address the dynamic nature of course catalogs, where sections may be added, removed, or rescheduled during registration periods, limiting its applicability in real-time environments.

2.1.2 Web-Based Personalized University Timetable for UiTM Students Using Genetic Algorithm

The UiTM project presents a web-based personalized timetable generator developed to assist students in creating their own schedules from the university's published course lists and timeslots. The motivation behind the system arises from the difficulty students face during course registration, particularly repeat students who often struggle to fit required subjects into their timetables without overlaps. Unlike institutional-level timetabling, which attempts to allocate courses, lecturers, and room across the entire university, this system is tailored to the student's perspective. By leveraging GA, the system automatically generates feasible schedules based on the courses chosen by an individual student. The objective is to eliminate timetable clashes, reduce the manual trial-and-error process, and provide students with a faster and smarter way to plan their semester. The project therefore directly contributes to the development of a personalized student timetable planner rather than a university-wide scheduling tool.

The system architecture integrates GA into a web application interface, providing students with an accessible platform to generate timetables. Students first select the courses they wish to register for from the course catalog. Each course has several groups (lectures, tutorials, or labs) with predefined timeslots already determined by the university. The GA represents potential solutions as chromosomes, where each gene corresponds to a selected course and its assigned group. The algorithm evaluates candidate solutions through a fitness function that prioritizes conflict-free scheduling and adherence to university-imposed rules such as credit limits. GA applies standard operators, crossover, mutation, and selection, to evolve a population of candidate timetables, eventually converging towards valid solutions. Importantly, the system also allows students to "lock" certain courses or groups they prefer, giving them some control while the GA optimizes the rest of the schedule. The final timetable is displayed to the student in a clear and structured format, ready to be used for registration.

One of the primary strengths of this project is its student-centered orientation. Instead of tackling the highly complex institutional timetable, which involves balancing multiple global constraints, the system narrows its scope to the individual student's needs, making it both practical and computationally feasible. This focus ensures that the GA runs efficiently,

producing personalized timetables within reasonable time frames. Another strength is the integration of user preferences; by allowing students to lock specific courses or groups, the system combines the flexibility of manual planning with the efficiency of algorithmic optimization. The web-based interface is also an advantage, as it improves accessibility for students and eliminates the need for specialized software. Finally, the project is validated with real use cases, showing that it can reduce the stress and inefficiency students often experience during registration periods.

Despite its advantages, the project has some limitations. The most notable limitation is its reliance on static course lists and timeslots provided by the university. If the catalog changes, such as when classes are canceled, timeslots are updated, or new sections are added. The system does not incorporate real-time adjustments, which could reduce its practicality in dynamic registration environments. Another limitation is the binary representation of course groups, which assumes that each course offering is pre-defined and fixed; the design may restrict flexibility when students wish to choose among multiple tutorial or practical groups, or when group availability changes suddenly. Additionally, the fitness function primarily focuses on clash detection and basic workload balance but does not fully incorporate more nuanced student preferences, such as avoiding early morning classes, reducing long gaps between sessions, or creating compact daily schedules. Lastly, while the GA approach is effective for small to medium sized, its performance and scalability in handling very large course catalogs or thousands of students simultaneously remains untested in this project.

2.1.3 Heuristic Algorithm for a Personalized Student Timetable

The paper presents a system for generating personalized student timetables using a heuristic algorithm. Unlike institutional timetabling, where the university must assign courses, rooms, and lectures across the entire institution, this project focuses on the individual student's perspective. The university first produces a raw timetable in which all lectures and practical sessions are predefined and assigned times and locations. The challenge for students arises when courses offer multiple seminars or practical groups, often leading to overlapping sessions and inefficient use of their weekly schedules. The system therefore aims to optimize the student's timetable by eliminating conflicts and arranging sessions in a way that reduces idle time on campus. This makes the project highly relevant as a personal student timetable planner,

supporting flexibility for students with part-time jobs, extracurricular activities, or preferences for compact schedules.

The system applies to a heuristic algorithm that balances efficiency with practicality. Each student's set of courses is treated as input, with multiple possible session groups available for selection. The heuristic works by incrementally building a conflict-free timetable, giving priority to non-overlapping sessions and compact arrangements that minimize the number of separate campus visits. This approach is computationally lightweight compared to methods such as GA, enabling fast generation of timetables even for large datasets. Importantly, the design assumes that all lectures are compulsory and already fixed by the university; the heuristic therefore primarily selects alternative tutorials, seminars, or practical sessions to tailor the schedule for each individual student. This simplified but effective model ensures that the algorithm focuses only on personal student optimization, without being burdened by institutional resource allocation.

A major strength of this project is its student-centric scope. By limiting the problem to personal timetable, the system avoids the extreme complexity of university-wide scheduling while directly addressing the challenges students face during course registration. The use of a heuristic algorithm provides speed and practically, generating near-optimal timetables in real-time. Another strength is its explicit consideration of student lifestyle needs, particularly reducing campus idle hours and improving schedule compactness, which directly improves satisfaction. The algorithm also scales well, since it does not require global optimization across thousands of students but only needs to handle one student's dataset at a time.

The project also has limitations that restrict its broader applicability. First, the reliance on a pre-existing raw timetable means it cannot adapt if the university makes frequent changes to a course schedule during registration periods. Second, the heuristic focuses mainly on avoiding conflicts and reducing idle time but does not incorporate softer preferences, such as avoiding morning classes, preferring specific days off, or prioritizing certain instructors. Compared to multi-objective approaches like GA, the heuristic may also produce fewer diverse timetable options, limiting student choice. Furthermore, because evaluation was not tested extensively across different institutions or datasets, questions remain about its generalizability beyond the specific case studied.

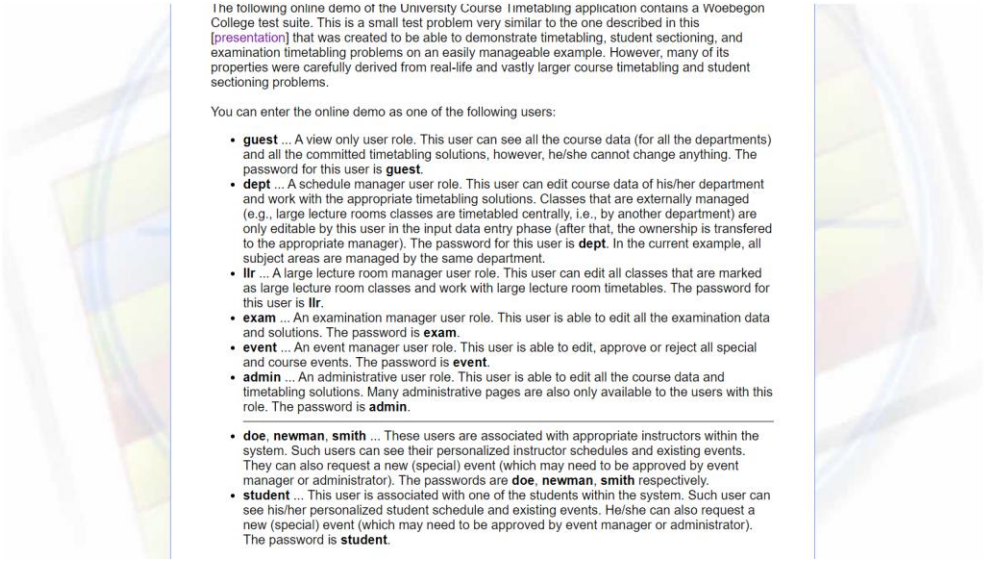
*2.*2 Existing Systems

2.2.1 University Timetabling System (UniTime) [2]

The UniTime system, as detailed on its official website, aims to provide comprehensive solutions for academic scheduling challenges, including course timetabling, examination scheduling, and student scheduling. The primary objective of UniTime is to streamline and optimize the scheduling process for educational institutions, ensuring efficient use of resources while meeting the diverse needs of students, faculty, and administrative staff. This is achieved through a flexible and extensible scheduling platform that accommodates various institutional constraints and preferences.

The development process of UniTime involves several key stages. Initially, a thorough requirements analysis is conducted to understand the specific scheduling needs and constraint-based and optimization techniques, to generate feasible schedules. These algorithms are designed to handle complex constraints such as room capacities, time preferences, and course conflicts. The development process also includes extensive testing and validation to ensure the generated schedules are practical and efficient. UniTime's modular architecture allows for customization and integration with other institutional systems, enhancing its adaptability and functionality.

UniTime has achieved significant success in providing robust scheduling solutions to numerous educational institutions worldwide. Its achievements include the ability to generate conflict-free schedules that optimize the use of available resources, such as classrooms and faculty time, while accommodating a wide range of constraints and preferences. The system's user-friendly interface and comprehensive reporting tools facilitate easy management and adjustment of schedules. Moreover, UniTime's open-source nature encourages community collaboration and continuous improvement, leading to a highly adaptable and evolving platform.



The following online demo of the University Course Timetabling application contains a Woebegon College test suite. This is a small test problem very similar to the one described in this [presentation] that was created to be able to demonstrate timetabling, student sectioning, and examination timetabling problems on an easily manageable example. However, many of its properties were carefully derived from real-life and vastly larger course timetabling and student sectioning problems.

You can enter the online demo as one of the following users:

- **guest** ... A view only user role. This user can see all the course data (for all the departments) and all the committed timetabling solutions, however, he/she cannot change anything. The password for this user is **guest**.
- **dept** ... A schedule manager user role. This user can edit course data of his/her department and work with the appropriate timetabling solutions. Classes that are externally managed (e.g., large lecture rooms classes are timetabled centrally, i.e., by another department) are only editable by this user in the input data entry phase (after that, the ownership is transfered to the appropriate manager). The password for this user is **dept**. In the current example, all subject areas are managed by the same department.
- **llr** ... A large lecture room manager user role. This user can edit all classes that are marked as large lecture room classes and work with large lecture room timetables. The password for this user is **llr**.
- **exam** ... An examination manager user role. This user is able to edit all the examination data and solutions. The password is **exam**.
- **event** ... An event manager user role. This user is able to edit, approve or reject all special and course events. The password is **event**.
- **admin** ... An administrative user role. This user is able to edit all the course data and timetabling solutions. Many administrative pages are also only available to the users with this role. The password is **admin**.

- **doe**, **newman**, **smith** ... These users are associated with appropriate instructors within the system. Such users can see their personalized instructor schedules and existing events. They can also request a new (special) event (which may need to be approved by event manager or administrator). The passwords are **doe**, **newman**, **smith** respectively.
- **student** ... This user is associated with one of the students within the system. Such user can see his/her personalized student schedule and existing events. He/she can also request a new (special) event (which may need to be approved by event manager or administrator). The password is **student**.

Figure 2.2.1.1 A list of different identity for user to choose from which interface they wish to refer.

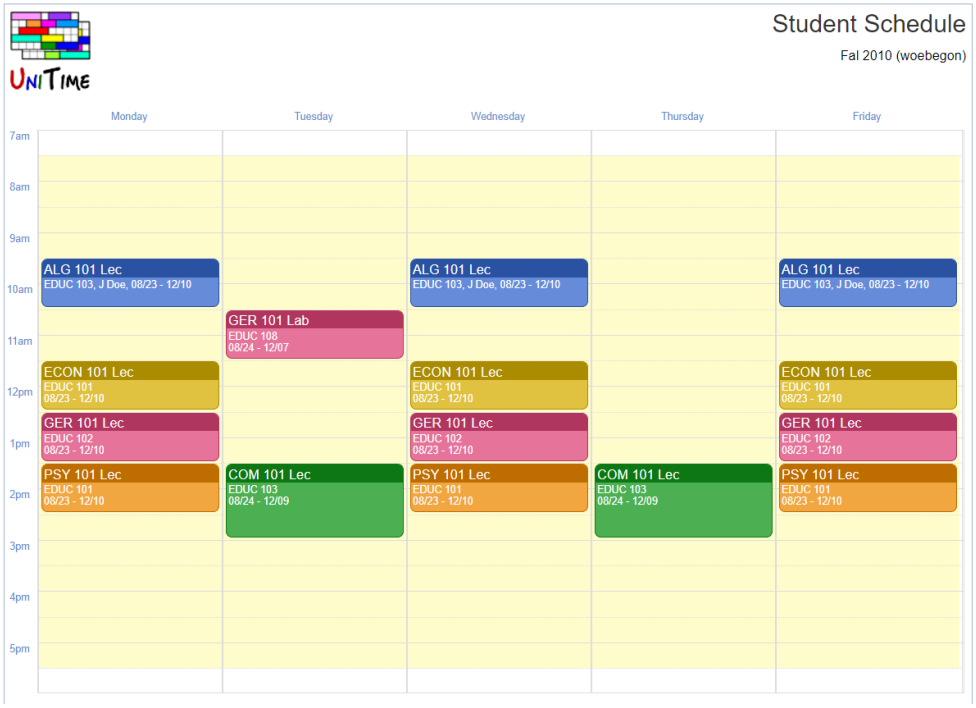


Figure 2.2.1.2 Sample demo of student schedule

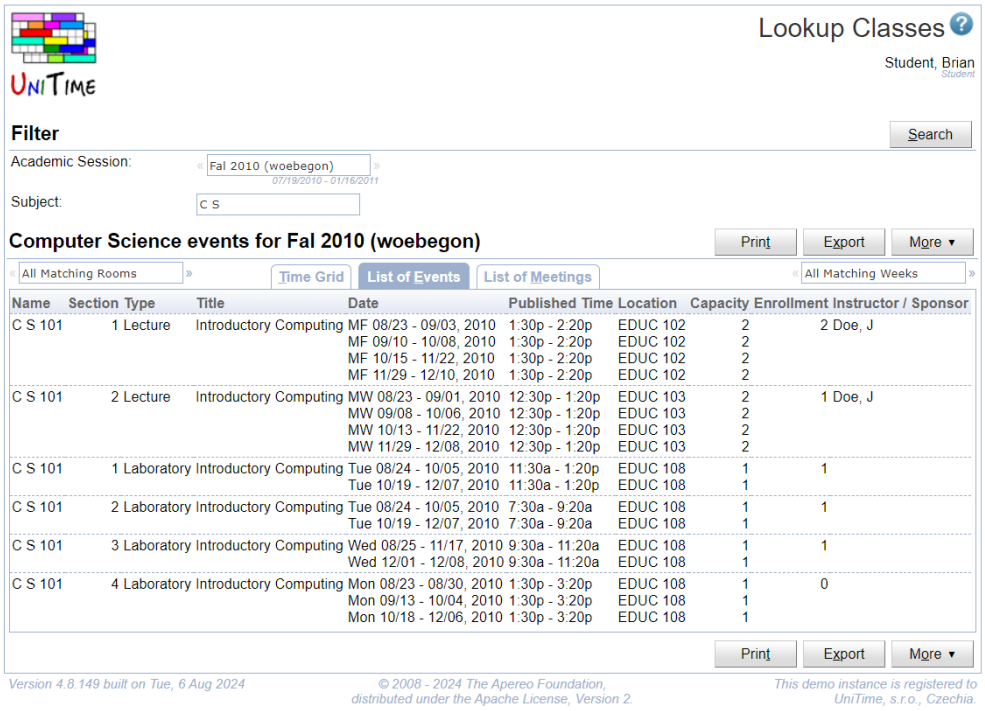


Figure 2.2.1.3 Lookup Classes Page

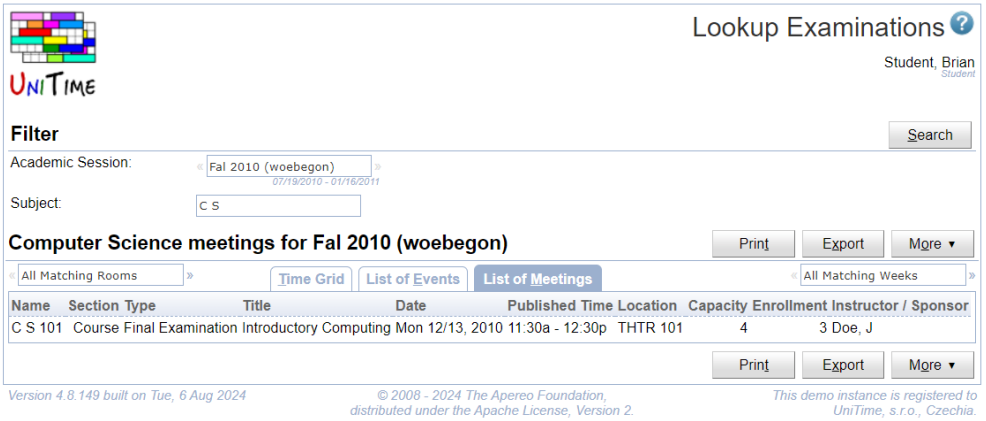


Figure 2.2.1.4 Lookup Examination Page

The UniTime website showcases several strengths and weaknesses in its design and functionality. One of the key strengths is its comprehensive and user-friendly interface, which provides detailed information about the system's features, capabilities, and use cases. This makes it easy for potential users to understand how UniTime can meet their scheduling needs. The website also offers extensive documentation, tutorials, and community support which are invaluable resources for new users and institutions looking to implement the system. Additionally, the open-source nature of UniTime is prominently highlighted, encouraging collaboration and continuous improvement within the user community.

However, the website has some weaknesses, including a somewhat cluttered layout that can make navigation challenging for first-time visitors. Important information may be buried under multiple layers of menus, making it less accessible. Furthermore, while the technical documentation is comprehensive, it might be overwhelming for users without a strong technical background, potentially deterring them from fully engagement with the platform. Overall, while the UniTime website is a valuable resource with extensive information and support, improvements in layout and accessibility could enhance user experience and engagement.

In conclusion, UniTime offers a sophisticated and flexible solution to the challenges of academic scheduling. Its development process integrates advanced optimization techniques and extensive testing to produce high-quality schedules. While the system has notable achievements in resource optimization and user adaptability, it also faces limitations related to complexity and initial setup requirements. Nonetheless, UniTime's ongoing development and open-source nature position it as a leading tool in the field of academic scheduling.

2.2.2 Timetable Arranging Program (TTAP) – UTAR [6]

The Timetable Arranging Program (TTAP) is a timetable scheduling program that was designed by a student named Wong Jia Hau which is a Faculty of Information Communication and Technology student from Universiti Tunku Abdul Rahman (UTAR). The objective of this program is to help the UTAR students to arrange timetables smoothly and easily by implementing automated scheduling concepts to the timetable.

This system represents a significant advancement in university course-scheduling through its comprehensive suite of features and user-centric design. One of the system's most notable features is its user-friendly interface, which is designed to simplify the complex task of scheduling. The graphical user interface (GUI), as illustrated in the tutorial GIF, enables users to interact intuitively with the system. The design approach minimizes the learning curve and facilitates ease of data entry, schedule visualization, and adjustment, making the scheduling process more accessible to users regardless of their technical proficiency.

Another feature of the TTAP-UTAR system is its automated timetable generation capability. By employing sophisticated algorithms, the system automates the creation of timetables, effectively reducing the manual effort involved in scheduling. This automation ensures that

timetables are generated with optimal consideration for various constraints such as room availability, instructor schedules, and course requirements. The system's ability to handle these constraints and generate conflict-free timetables enhances the efficiency of resource utilization within the university, thereby addressing one of the key challenges in academic scheduling. Conflict detection and resolution is another integral feature of the TTAP-UTAR system. During the timetable generation process, the system actively identifies potential scheduling conflicts and provides solutions or alternatives to resolve them. This proactive approach not only improves the overall quality of the timetable but also reduces the likelihood of disruptions caused by scheduling conflicts, ensuring a smoother ad more effective scheduling process.

The system's customizable constraints further contribute to its flexibility and adaptability. Users can define and adjust constraints based on their specific institutional requirements, such as room capacities, course prerequisites, and instructor availability. This customization allows the system to accommodate diverse scheduling needs and adapt to changing conditions, making it a versatile tool for various educational environments.

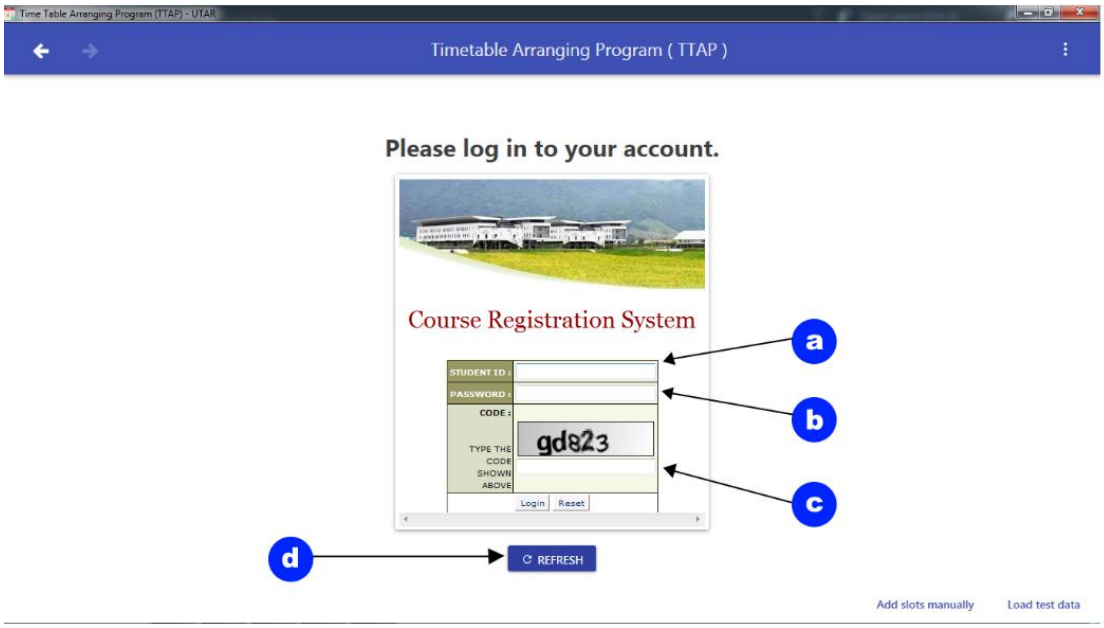


Figure 2.2.2.1 Login page for students

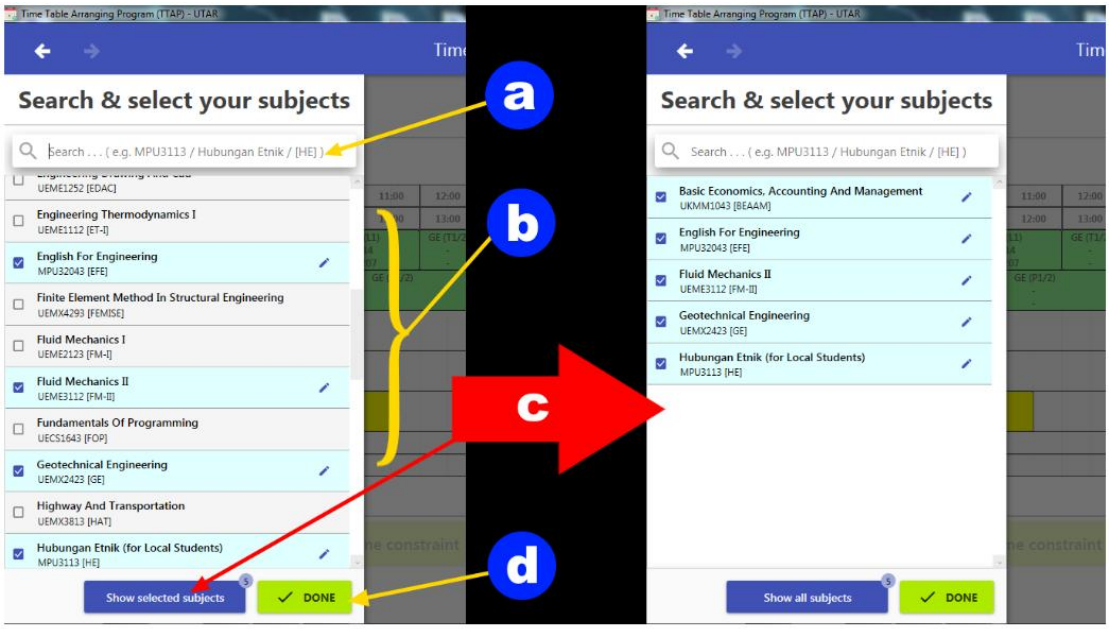


Figure 2.2.2.2 Subjects are listed for the students to choose from, and selected subjects will be shown after choosing.

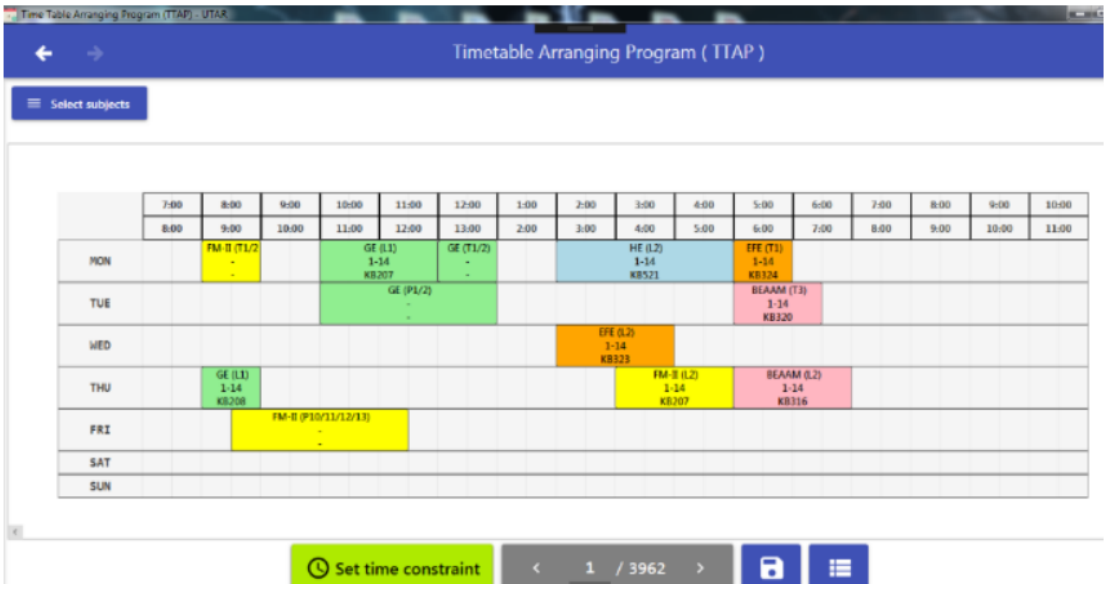


Figure 2.2.2.3 Set the time constraints

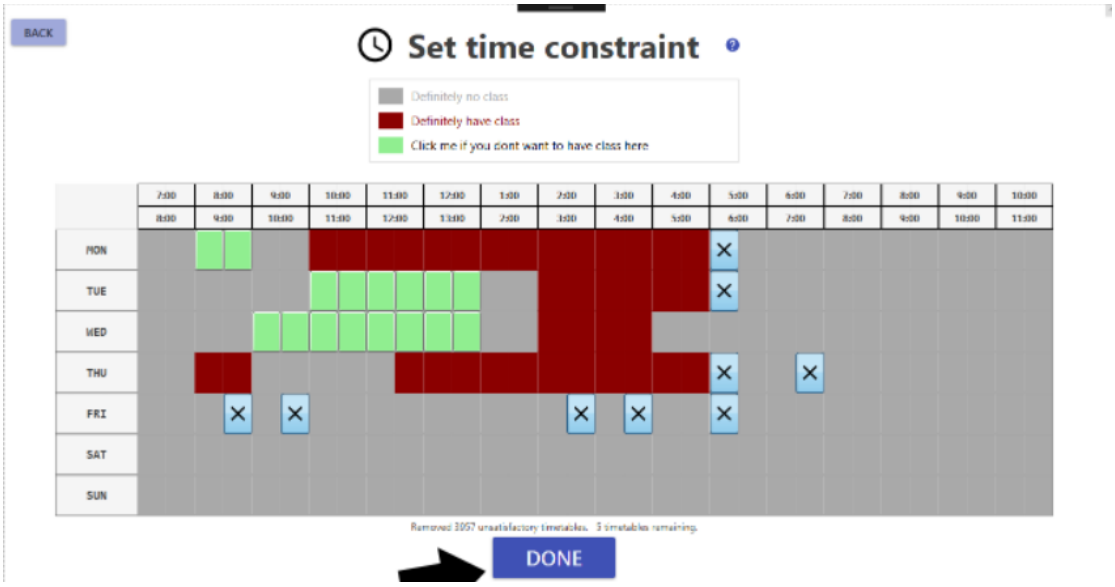


Figure 2.2.2.4 Choose preferable timeslots

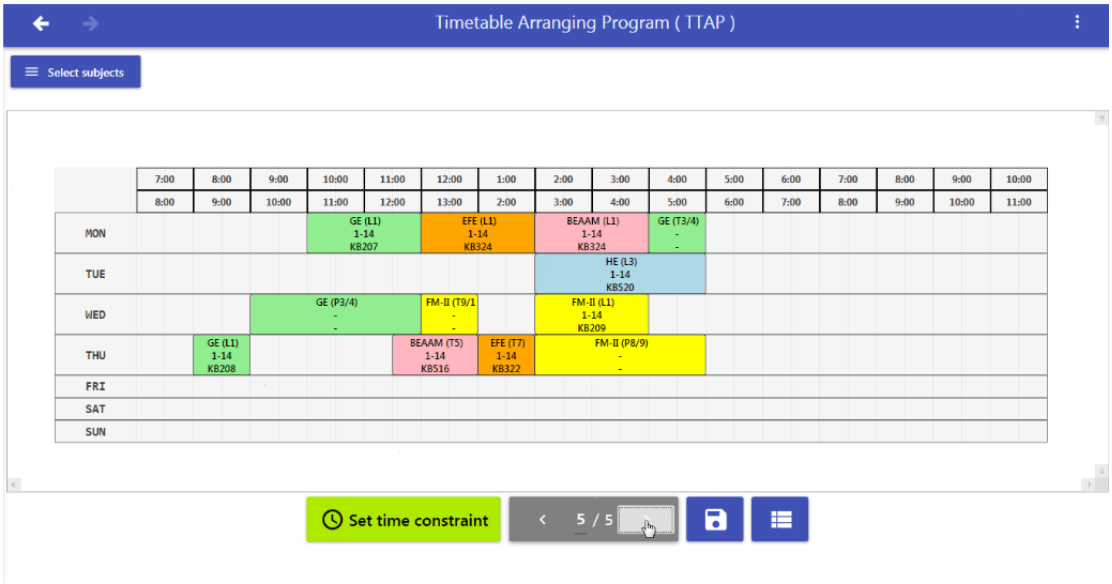


Figure 2.2.2.5 Choose preferable combination timetable

In terms of strengths, the TTAP-UTAR system excels in enhancing usability through its intuitive GUI design. This user-centric approach improves efficiency by making the scheduling process more manageable for users with varying levels of technical expertise. Additionally, the system's ability to optimize resource utilization through automated timetable generation and conflict resolution leads to more efficient scheduling and better management of university resources. The flexibility and customization offered by the system further enhances its value, as institutions can tailor the system to their unique needs and policies.

However, the TTAP-UTAR system is not without its challenges. The complexity of constraints management can be a significant drawback, especially when dealing with many constraints. Balancing various constraints requires careful consideration and may become time-consuming, potentially complicating the scheduling process. Additionally, the system may face scalability issues as the number of courses and constraints increases, which could impact its performance and efficiency in large institutions. Lastly, the system's dependencies on accurate data are a critical factor for its effectiveness. Inaccurate or incomplete input data can lead to suboptimal scheduling outcomes, necessitating manual adjustments to correct any errors and ensure the timetables meet institutional requirements.

In conclusion, the TTAP-UTAR system offers a robust solution for university course scheduling, with its user-friendly interface, automated timetable generation, and customizable constraints. Its strengths in usability, resource optimization, and flexibility make it a valuable tool for educational institutions. Nevertheless, addressing challenges related to constraints management, scalability, and data accuracy is essential for enhancing the system's overall effectiveness and ensuring its continued success in academic scheduling.

2.2.3 TimeEdit [7]

TimeEdit is a feature-rich scheduling and resource management system tailored for educational institutions and organizations. Its intuitive simplifies the creation, management, and viewing of schedules, featuring drag-and-drop functionality for easy modifications. The system's advanced scheduling algorithms automate the process, ensuring conflict free timetables by considering constraints like room availability, course requirements, and instructor preferences. Real-time updates ensure that any changes are immediately reflected across the system, maintaining up-to-date information and preventing scheduling conflicts.

Additionally, TimeEdit offers robust reporting tools, allowing users to generate detailed reports on resource utilization, scheduling conflicts, and other key metrics, aiding informed decision-making and resource optimization. Its integration capabilities with other institutional systems, such as Student Information Systems (SIS) and Learning Management Systems (LMS), ensure seamless data exchange and enhance operational efficiency. The system's high level of

customization enables institutions to tailor it to their specific needs, accommodating unique scheduling requirements, room layout, and its institutional policies.

TimeEdit excels in resource management, ensuring optimal allocation of rooms, equipment, and personnel, reducing downtime, and improving utilization. The system is accessible from various devices, including desktops, tablets, and smartphones, providing users the flexibility to manage and view schedules on the go. User roles and permissions allow administrators to control access levels, ensuring data security and efficient task delegation. Automated notifications keep users informed about schedule changes, upcoming events, or resource bookings, reducing the likelihood of missed appointments or double bookings. Through these comprehensive features, TimeEdit aims to streamline scheduling and resource management, making the process more efficient and less error prone.

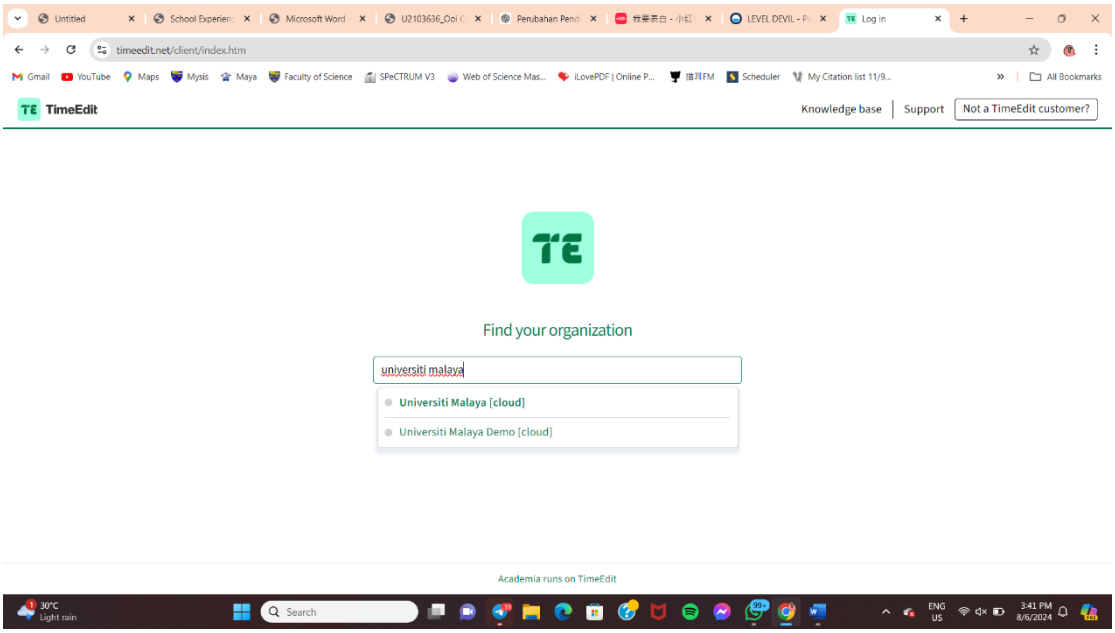


Figure 2.2.3.1 Users must search out the organization

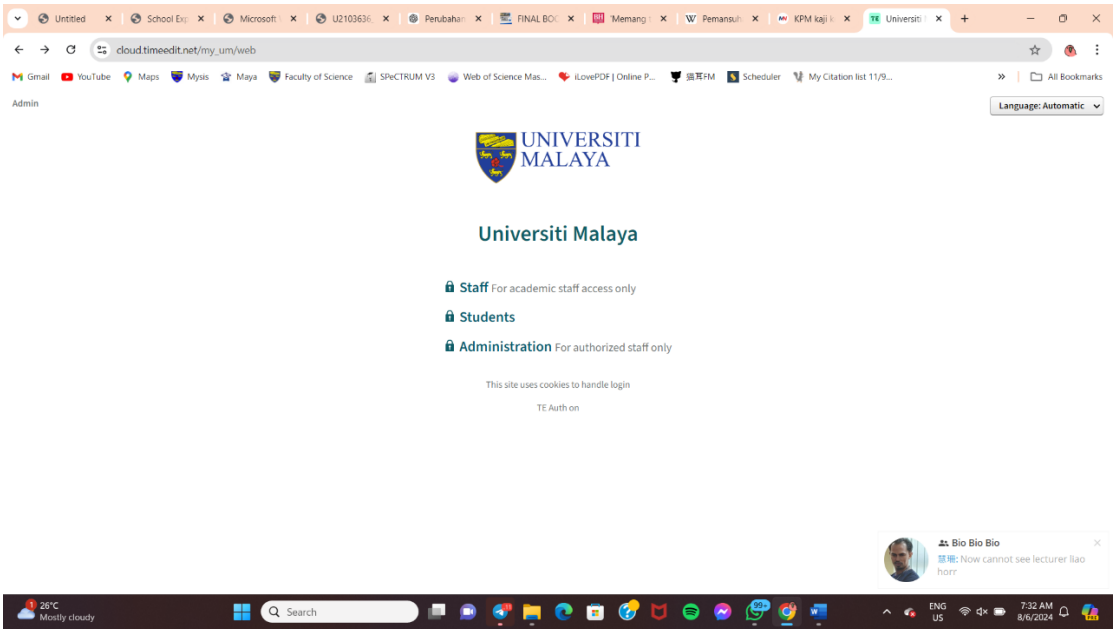


Figure 2.2.3.2 Login page for students, staff and administrators

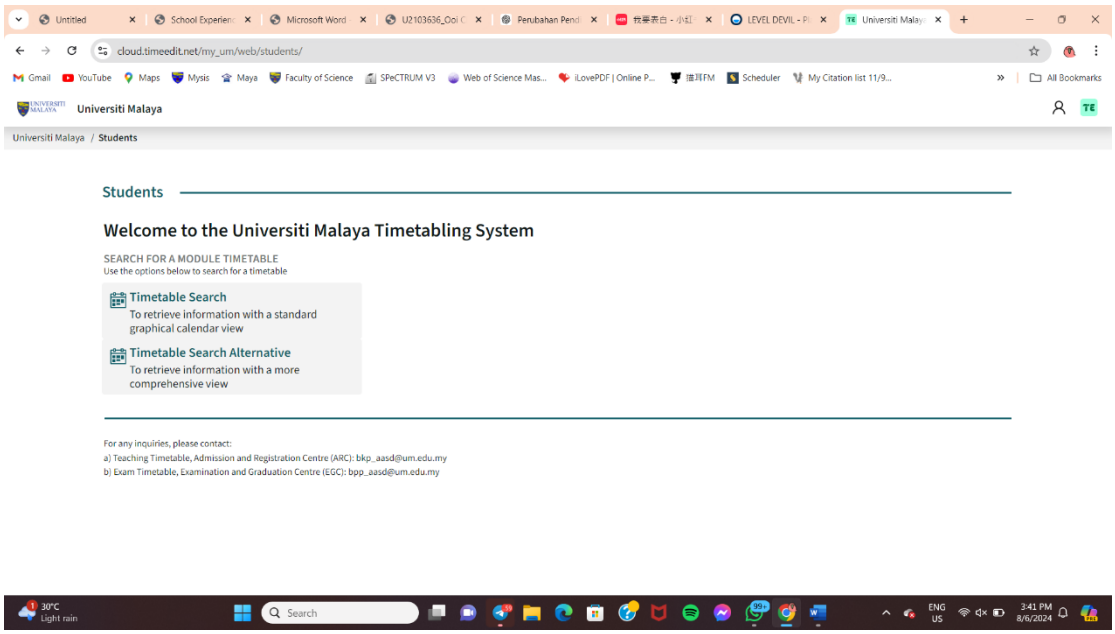


Figure 2.2.3.3 Welcome page of the timetabling system

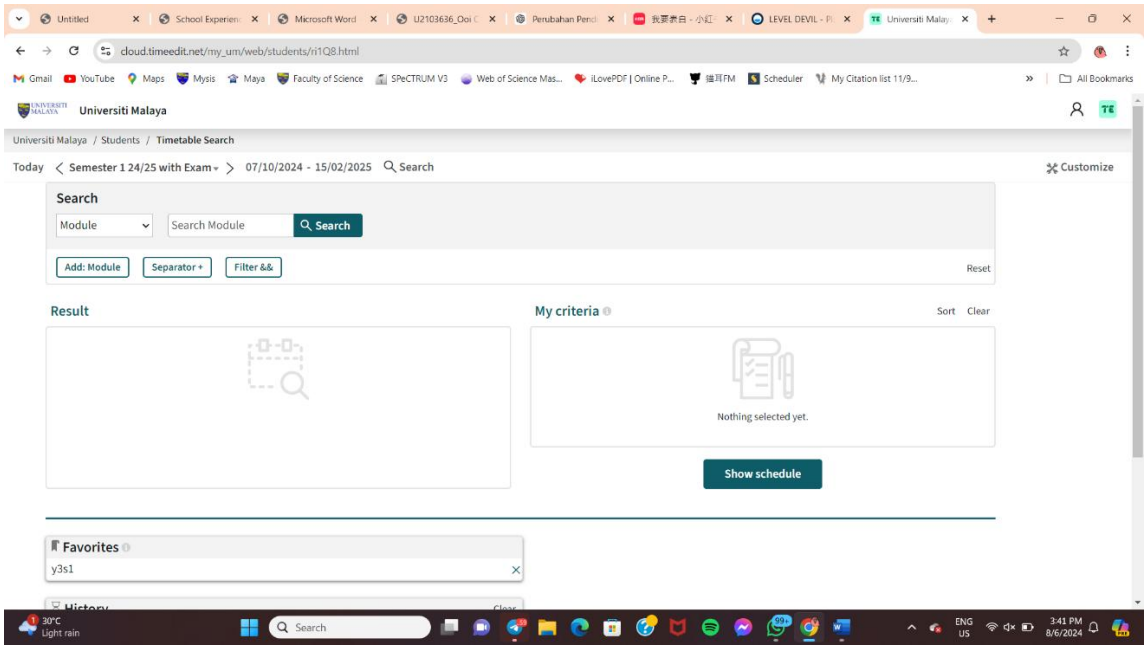


Figure 2.2.3.4 Search module page

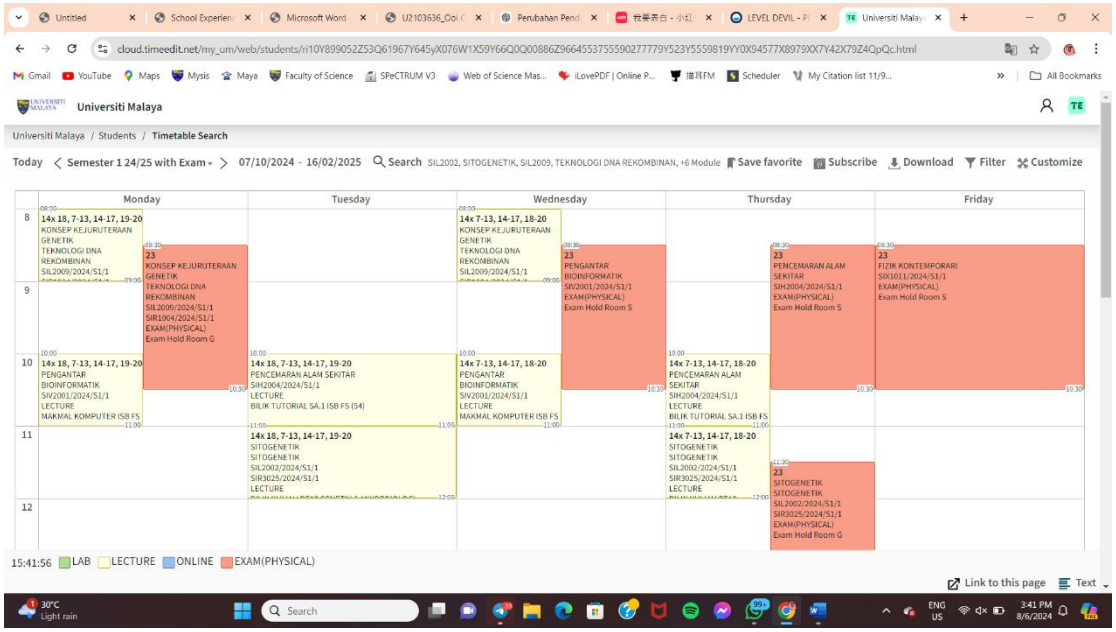


Figure 2.2.3.5 Student timetable

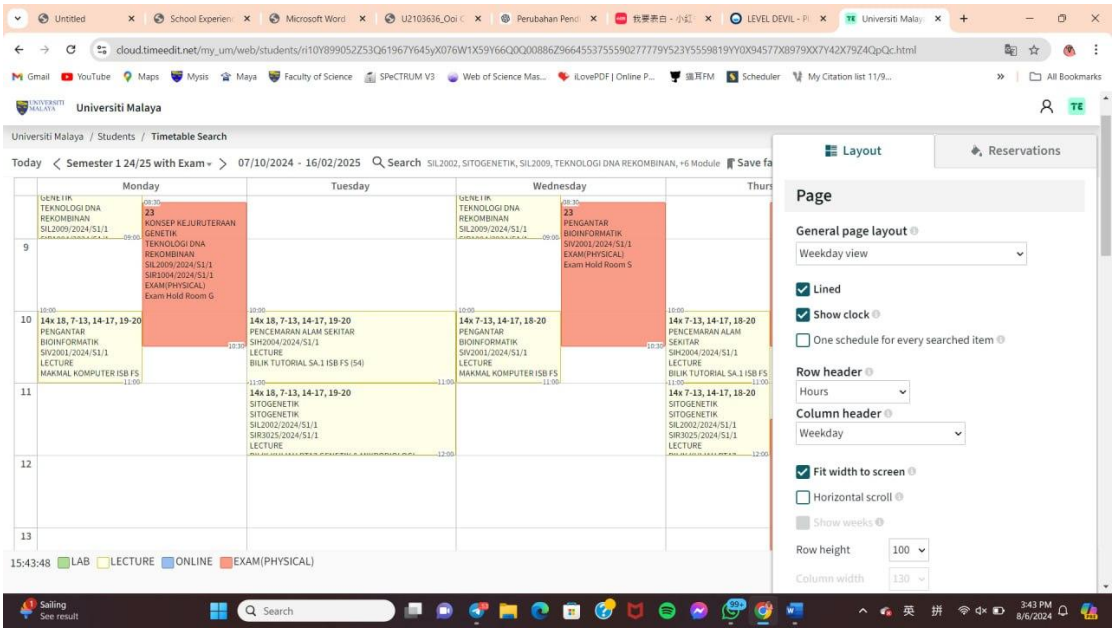


Figure 2.2.3.6 Details to show in the timetable

One of the primary strengths of TimeEdit is its user-friendly interface, which simplifies the complex process of scheduling for both administrators and users. Its advanced algorithms and automation capabilities significantly reduce the time and effort required to create and manage schedules, making it a valuable tool for instituions with diverse and dynamic scheduling needs. Additionally, the system's ability to integrate with other institutional software ensyres a seamless flow of information and enhances overall oeprational efficiency. The customizable nature of TimeEdit allows instituions to tailor the system to their specific requirements, ensuring that it meets unique scheduling challenges effectively.

However, TimeEdit also has its weaknesses. Despite its robust features, the initial setup and customization process can be complex and time-consuming, requiring a significant investment of time and resources. Additionally, while the system is designed to be user-friendly, some users may still face a learning curve, particularly those who are not tech-savvy. Furthermore, the reliance on internet connectivity for real-time updates may pose challenges in areas with unstable or limited internet access.

In conclusion, TimeEdit stands out as a powerful and versatile scheduling and resource management system, designed to meet the complex needs of educational institutions and other organizations. Its user-friendly interface, adavanced scheduling algorithms, and real-time updates streamline the scheduling process, ensuring efficiency and accuracy. The robust

reporting tools, integration capabilities, and high customization options allow institutions to optimize their resources and tailor the system to teir specific requirements. Despite some initial setup challenges and a learning curve for non-tech-savvy users, TimeEdit's comprehensive features, including effective resource management, accesibility, user roles, and automated notifications, make it an invaluable tool for improving operational efficiency and reducing scheduling conflicts. Overall, TimeEdit's strengths significantly outweigh its weaknesses, making it a highly recommend solution for any institution seeking to enhance its scheduling and resource management process.

2.3 Summary

In this chapter, three similar projects named "Personal Course Timetabling for University Students based on Genetic Algorithm", "Web-Based Personalized University Timetable for UiTM Students Using Genetic Algorithm" and "Heuristic Algorithm for a Personalized Student Timetable" has been reviewed. For the first paper models personal timetabling as a combinatorial optimization problem and applies a GA to produce conflict-free schedules that balance hard constraints such as credit limits with soft preferences like compactness and academic progression. The second paper builds on similar GA principles but delivers a practical, web-based system integrated with UiTM's course catalog, enabling students to lock preferred groups, save and export timetables, and benefit from administrative updates. In contrast, the last paper emphasizes speed and efficiency, using a lightweight method to select non-conflicting seminar and practical groups while reducing idle time on campus, though with less flexibility for incorporating diverse student preferences. Together, these works demonstrate complementary strengths in accuracy, usability, and efficiency, contirbuting significantly to the development of smart, student-centered timetable planners.

Three exisitng systems, including University Timetabling, Timetable Arrangement Program – UTAR and TimeEdit are reviewed, and the strengths and weaknesses are listed out. Their strengths and weaknesses are then summarized and concluded in Table 2.3.1.

Table 2.3.1 Comparison of Strengths and Weaknesses of Existing Systems

Feature	**University Timetabling**	**TTAP - UTAR**	**TimeEdit**
User Interface	Often user-friendly, but varies	Intuitive and user-freindly	User-friendly interface with drag-and-drop
Optimization Algorithms	Advanced, handles complex constraints	Efficient algorithms for diverse constraints	Advanced algorithms for conflict-free-scheduling
Resource Management	Optimizes use of resources	Effective conflict detection and resolution	Optimal allocation of rooms, equipment, personnel
Flexibility	Open-source, encourages collaboration	Customizable to institution needs	Highly customizable to specific requirements
Initial Setup	Time consuming, requires expertise	Time-consuming, requiring technical knowledge	Complex and time-consuming setup
Computational Complexity	High for large datasets	Significant computational resources needed	Requires significant computational resources
Usability	Can vary, some systems complex	Complexity of features might overwhelm users	Learning curve for non-tech-savvy users
Customization	Often requires extensive adjustments	Needs substantial customization and fine-tuning	High level of customization required
Training and support	May require significant training	Subtantial training and support needed	Significant training and support needed

Chapter 3

System Methodology/Approach OR System Model

In this chapter, the technologies and methodology to develop the project are implemented. The activity diagram, flowchart, use case diagram and descriptions are done in this chapter to show the overview of the system. The timeline of the project is also planned to deliver the project on time.

3.1 Methodology

The methodology proposed for the project is Rapid Application Development (RAD). RAD adopts traditional Software Development Life Cycle (SDLC) phases to accelerate program development. This approach enables the project to be iterated and updated continuously throughout the development process, emphasizing rapid prototyping and incorporating user feedback. [8]

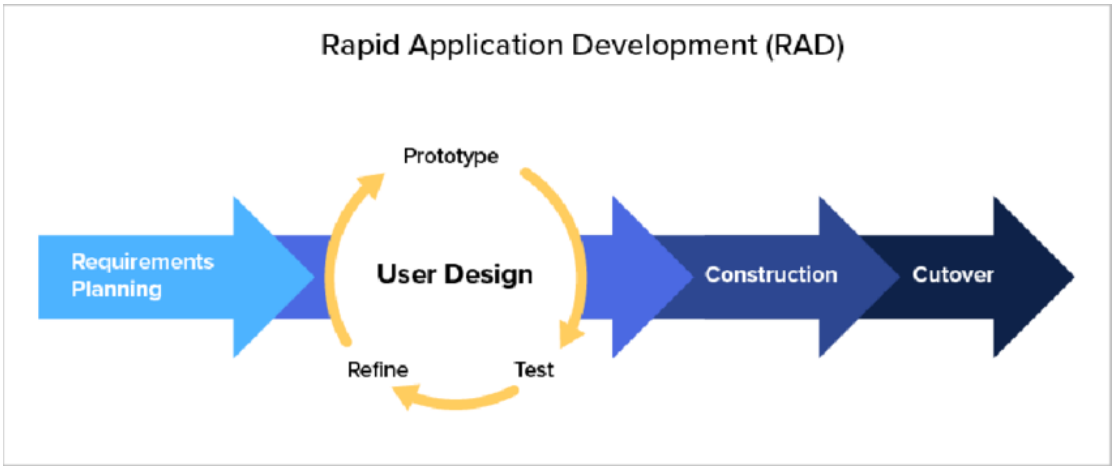


Figure 3.1.1 RAD Methodology

3.1.1 Requirements Planning

The Requirements Planning phase is the initial stage of the development process for the Smart Student Timetable Planner. This phase begins with the identification of key problem statements

related to the current timetable system. The objective of the project is clearly outlined, focusing on developing a robust timetable system that addresses identified issues.

During this phase, a comprehensive review of existing literature and systems is conducted. This review aims to analyze the strengths and weaknesses of current solutions and to identify the potential requirements needed for the new system. This phase also includes defining the project's objectives, specifying hardware and software requirements, and creating a detailed project timeline. By setting clear requirements and understanding existing solutions, this phase lays the groundwork for a well-informed and structured development process.

3.1.2 User Design

The User design phase focuses on developing prototypes to visualize and refine the system's user interface and functionality. HTML and CSS are used to build structural and visual elements, while JavaScript and jQuery bring interactivity for dynamic course selection and timetable management. JSON, localStorage and sessionStorage are applied to manage course data and user choices, ensuring that timetable information is stored and easily retrievable during active sessions. Socket.IO is integrated into the prototypes to demonstrate real-time collaboration, enabling users to view updates and communicate instantly while arranging timetables.

At this stage, the main modules are represented through interactive prototypes. These include login or logout, course selection with conflict detection and time constraints, timetable viewing, collaboration with session chat, saving timetables to history, comparing manual and auto-generated schedules, exporting timetables, and administrative course management. Prototypes are tested with sample users to gather feedback on usability and functionality. Based on this feedback, the designs are iteratively improved to ensure a user-friendly, intuitive, and functional system before progressing to full implementation.

3.1.3 Construction and Feedback Phase

The Construction and Feedback phase transforms the prototypes into a fully working timetable system. In this stage, the system modules are implemented and integrated using HTML, CSS, and JavaScript for the frontend, while JSON, sessionStorage, localStorage are used to manage timetable data. Socket.IO is applied to handle synchronization and broadcasting of timetable updates during collaborative sessions. All modules, from secure login and conflict-free

scheduling to exporting timetables and managing administrative updates, are developed and linked together into a cohesive system.

Testing and feedback are crucial elements of this phase. Functionality testing ensures that modules perform as intended, usability testing checks that the system is intuitive for students, interface testing evaluates design consistency, and compatibility testing validates performance across devices and browsers. Feedback collected from test users is used to iteratively improve the system, refining features, fixing issues, and enhancing overall reliability. By the end of this phase, the timetable system is stable, accurate, and aligned with user expectations, ready to support students in managing their personal schedules efficiently.

3.1.4 Finalize Product and Implementation Phase

During the Finalize Product and Implementation phase, it focuses on ensuring the system is fully operational and ready for deployment. This involves comprehensive testing to verify that all modules function correctly, with a particular emphasis on functionality, usability, interface consistency, and cross-browser compatibility.

Once testing is complete, detailed documentation is prepared, covering system design, operation, database structure, functionalities, and testing results. This phase culminates in the system's deployment, marking the transition from development to active use, with the project now complete and ready for launch.

3.1.5 Maintenance and Evaluation

This phase ensures that the system remains functional and continues to meet users' needs for post-deployment. This phase involves ongoing activities to support and improve the system.

System performance and user feedback are continuously monitored to identify any issues or areas for improvement. Support teams address reported problems and help as needed. Regular updates are made to fix bugs, enhance performance, and introduce new features based on user feedback. Additionally, post-deployment evaluations are conducted to assess the system's effectiveness and gather insights into future improvements. The outcome of this phase is a well-maintained system that evolves to meet the university's needs and maintains high performance and relevance.

3.2 System Design Diagram

3.2.1 System Design Flowchart

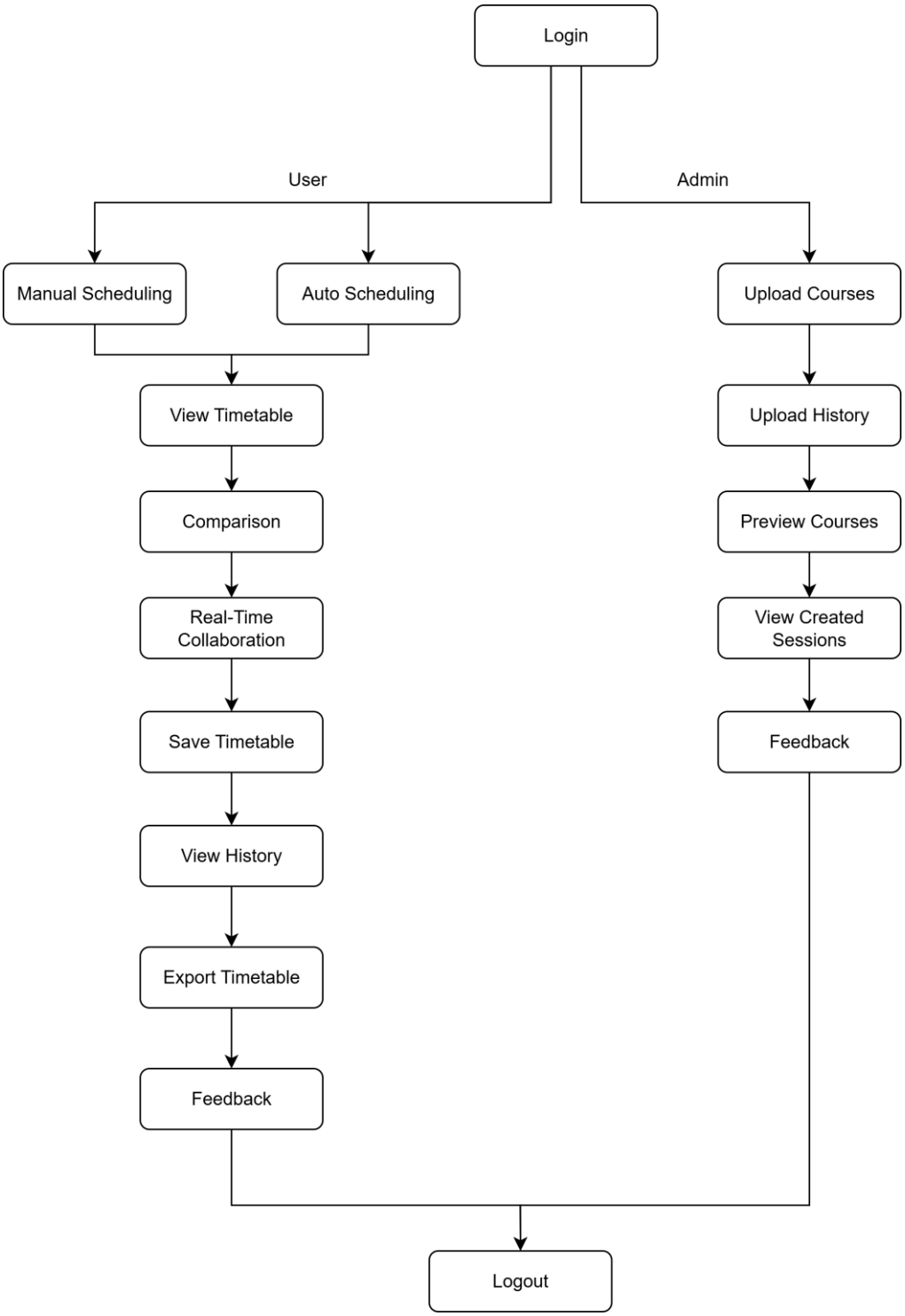


Figure 3.2.1 System Overview Design Flowchart

3.2.2 Use Case Diagram

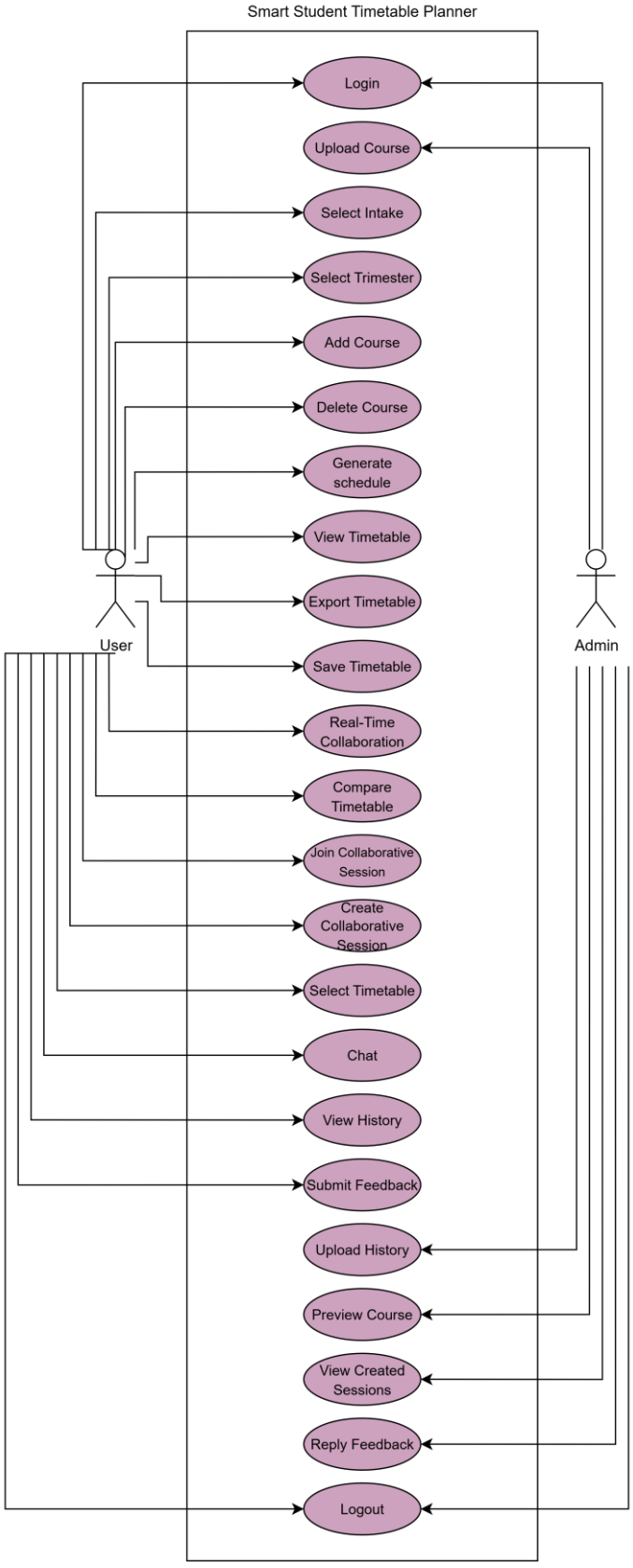


Figure 3.2.2 Use Case Diagram of Smart Student Timetable Planner

3.2.3 Use Case Description

Table 3.2.3.1 Login Use Case Description

Use Case ID	00001
Use Case Name	Login
Brief Description	Users and administrators are allowed to access the platform by logging into their account with valid credentials.
Actor	User, Administrator
Trigger	Go to the login page of the platform.
Precondition	1. Accounts must exist in the system and have not logged in.
Normal flow of events	1. User navigates to a login page and inputs username and password. 2. The system sends a POST request to /login with the credentials. 3. Backend verifies if the credentials are valid. 4. If valid, the system stores the session in sessionStorage and redirects to the main page. 5. User is redirected to their dashboard according to their role (student/admin).
Sub Flows	-
Alternate Flows	2a. If credentials are invalid, error is displayed, and user is asked to try again.

Table 3.2.3.2 Logout Use Case Description

Use Case ID	00002
Use Case Name	Logout
Brief Description	Allows a logged-in user to securely exit the system, ending their current session and preventing unauthorized access to personal data.
Actor	User, Administrator
Trigger	The user selects the “Logout” option from the navigation menu.
Precondition	1. The user must be logged into the system.

Normal flow of events	1. User clicks on the “Logout” button. 2. The system terminates the current session. 3. The system clears session data from storage. 4. The user is redirected to the login page.
Sub Flows	-
Alternate Flows	-

Table 3.2.3.3 Select Intake Use Case Description

Use Case ID	00003
Use Case Name	Select Intake
Brief Description	User selects their academic intake in the system. The selected intake determines the courses and timeslots available for that student.
Actor	User
Trigger	User chooses to begin the course scheduling process.
Precondition	1. The user is logged into the system. 2. The system has intake and course data uploaded by the administrator.
Normal flow of events	1. The system displays a list of available intakes. 2. The user selects their intake from the dropdown. 3. The system retrieves and displays the course offered for the selected intake. 4. The user proceeds to select their trimester and courses.
Sub Flows	-
Alternate Flows	-

Table 3.2.3.4 Select Trimester Use Case Description

Use Case ID	00004
Use Case Name	Select Trimester
Brief Description	User selects the trimester in which they are registering for courses. The system will then filter and display only the courses and timeslots available for the chosen trimester.
Actor	User

Trigger	The user logs into the system and proceeds to schedule their timetable, requiring trimester selection before course registration.
Precondition	1. The user is logged into the system. 2. The system has intake and course data uploaded by the administrator.
Normal flow of events	1. The user navigates to the course scheduling module. 2. The system prompts the students to select a trimester. 3. The user selects the desired trimester from the dropdown list. 4. The system retrieves and displays all courses offered in that trimester. 5. The user proceeds to course selection.
Sub Flows	3a. If the user changes their trimester selection, the system refreshes the available courses accordingly.
Alternate Flows	2a. If no trimester data isa available, the system displays an error message.

Table 3.2.3.5 Add Course Use Case Description

Use Case ID	00005
Use Case Name	Add Course
Brief Description	Allows user to add a course into their timetable for the selected trimester. The system ensures that the chosen course does not conflict with other registered courses and meets the trimester's requirements.
Actor	User
Trigger	User clicks "Add Course" on a listed course.
Precondition	1. The user must be logged into the system. 2. The user must have selected a trimester. 3. The course must be available in selected trimester and not already added.
Normal flow of events	1. User clicks "Add Course" button after choosing lectures and desired practical/tutorial slot. 2. The system checks for duplicates. 3. If valid, the course is added to the selected course list.

	4. The UI updates to show the course in the selected list.
Sub Flows	2a. If the course requires compulsory sessions, the system prompts the user to select a preferred group. 3a. If the course has multiple groups, the system prompts the user to select a preferred group.
Alternate Flows	2a. If the course is already added, a warning message is displayed.

Table 3.2.3.6 Delete Course Use Case Description

Use Case ID	00006
Use Case Name	Delete Course
Brief Description	Allows user to remove a previously added course from their personal timetable for the selected trimester. The system updates the timetable and ensures that all related sessions for that course are removed.
Actor	User
Trigger	User clicks “Delete” on a course from their timetable.
Precondition	1. The user must be logged into the system. 2. The user must have selected a trimester. 3. At least one course must be present in the user’s timetable.
Normal flow of events	1. The user navigates to their timetable view. 2. The system displays all courses currently added to the timetable. 3. The user selects a course to delete. 4. The system prompts the student to confirm the deletion. 5. The user confirms the deletion. 6. The system removes the selected course from the timetable. 7. The updated timetable is displayed to the student.
Sub Flows	-
Alternate Flows	4a. If the user cancels the confirmation, the course remains in the timetable.

Table 3.2.3.7 View Timetable Use Case Description

Use Case ID	00007
Use Case Name	View Timetable
Brief Description	Allows user to view their current timetable for the selected trimester. The timetable displays all registered courses with details such as course code, course name, day, time, allocated room, and selected group.
Actor	User
Trigger	The user selects the option to view their timetable by clicking "View Timetable" button.
Precondition	1. The user must be logged into the system. 2. The user must have selected a trimester. 3. At least one course should be registered in the timetable.
Normal flow of events	1. The user clicks the "View Timetable" button. 2. The system retrieves the user's timetable data for the selected trimester. 3. The system generates and displays the timetable in a structured format. 4. The user reviews the timetable.
Sub Flows	-
Alternate Flows	2a. If no courses are registered, the system displays alert message. 2b. If timetable data cannot be retrieved due to a system error, the system display error message.

Table 3.2.3.8 Export Timetable Use Case Description

Use Case ID	00008
Use Case Name	Export Timetable
Brief Description	Allows user to export their finalized timetable into different formats (IMG, CSV) for offline use, printing, or sharing.
Actor	User
Trigger	The user selects the "Export Timetable" button from the system.
Precondition	1. The user must be logged into the system.

	2. The user must have already generated a timetable.
Normal flow of events	1. The user navigates to the “Export Timetable” button. 2. The system prompts the user to choose a file format. 3. The user selects the desired format. 4. The system generates the timetable file in the selected format. 5. The system prompts the students to download or save the file. 6. The student successfully saves or downloads the file.
Sub Flows	-
Alternate Flows	2b. If no timetable is available, the system displays: “No timetable found to export.” 4a. If the file generation fails, the system displays an error message: “Export failed. Please try again.”

Table 3.2.3.9 Save Timetable Use Case Description

Use Case ID	00009
Use Case Name	Save Timetable
Brief Description	Allows the user to save a generated or manually created timetable into their personal history for future reference, comparison, or retrieval after logging in again.
Actor	User
Trigger	Users click on “Save Timetable” button.
Precondition	1. The user must be logged into the system. 2. The user must have a valid timetable generated or created.
Normal flow of events	1. The user views their generated or created timetable. 2. The user clicks on the “Save Timetable” button. 3. The system stores the timetable in the student’s history with the given label and timestamp. 4. The system stores the timetable in the user’s history with the intake, trimester and timestamp. 5. The system displays a confirmation message: “Timetable saved successfully.”
Sub Flows	-

Alternate Flows	2a. If no timetable is available, the system displays: “No timetable available to save.” 4a. If the saving process fails, the system displays: “Unable to save timetable. Please try again.”

Table 3.2.3.10 Select Time Constraints Use Case Description

Use Case ID	000010
Use Case Name	Select Time Constraints
Brief Description	User selects unavailable time slots to avoid conflicts in auto-scheduling.
Actor	User
Trigger	1. Students click on unavailable time in the input section in the auto-scheduling page.
Precondition	1. The user must be logged into the system. 2. User must be in auto-scheduling mode.
Normal flow of events	1. User marks unavailable time slots on the timetable interface. 2. System saves these constraints locally in sessionStorage. 3. During auto-scheduling, GA filers out sessions that overlap with constraints.
Sub Flows	-
Alternate Flows	2a. If no constraints are added, all-time slots are considered available.

Table 3.2.3.11 Generate Schedules Use Case Description

Use Case ID	000011
Use Case Name	Generate Schedules
Brief Description	User uses Genetic Algorithm to generate valid timetables.
Actor	User
Trigger	Click on “Generate Schedule” button.
Precondition	1. At least one course is selected.
Normal flow of events	1. User clicks “Generate Schedule”. 2. GA runs through a population of potential schedules.

	3. Valid timetables are checked against constraints (availability, no clash). 4. The system displays a paginated list of valid schedules.
Sub Flows	-
Alternate Flows	-

Table 3.2.3.12 Compare Timetable Use Case Description

Use Case ID	000012
Use Case Name	Compare Timetable
Brief Description	Enables the user to compare a manually created timetable with one generated automatically by the system's Genetic Algorithm. The comparison highlights differences in course session allocation, conflicts, and timetable efficiency to help the user decide which option suits them best.
Actor	User
Trigger	Click on "Comparison" tab.
Precondition	1. The user must be logged into the system. 2. At least one manual timetable and one auto-generated timetable must be available in the system.
Normal flow of events	1. The user navigates to the timetable comparison feature. 2. The system prompts the students to select one manual timetable and one auto-generated timetable. 3. The system displays timetable up-and-down. 4. The user reviews the comparison results.
Sub Flows	-
Alternate Flows	2a. If only one type of timetable is available, the system shows: "No available timetable."

Table 3.2.3.13 Join Collaborative Session Use Case Description

Use Case ID	00013
Use Case Name	Join Collaborative Session
Brief Description	Allows a user joins an existing collaboration session to work with peers on merging and adjusting timetable in real time. Once joined, the user can view the shared timetable, submit their own, and participate in collaborative decision-making.
Actor	User
Trigger	User selects the option to join a collaborative session and enters a valid session ID and username.
Precondition	1. The user is logged into the system. 2. A collaboration session must exist. 3. The user has at least one generated timetable available to submit.
Normal flow of events	1. The user navigates to the "Merging" section. 2. The user enters a session ID, password and username. 3. The system validates the session ID and confirms availability. 4. The user successfully joins the session. 5. The system displays the list of participants currently in the session. 6. The shared timetable is displayed to the user. 7. The user can now submit their timetable and edit the merged timetable.
Sub Flows	2a. If the session requires a password or authentication, the system prompts the user to enter the credentials. 5a. When a new user is joined, the system notifies all other participants: "[Username] has joined the session."
Alternate Flows	3a. If the session ID is invalid, the system notifies the student: "Invalid session ID, please try again." 4a. If connection to the session fails due to a network error, the student is prompted to retry joining. 6a. If no timetable has been merged yet, the system displays a message: "Waiting for users to submit timetables."

Table 3.2.3.14 Create Collaborative Session Use Case Description

Use Case ID	00014
Use Case Name	Create Collaborative Session
Brief Description	User initiates a new collaborative session that allows multiple users to join, submit their timetables, and work together on merging and editing timetables in real time.
Actor	User
Trigger	The user selects the option to create a new collaboration session.
Precondition	1. The user is logged into the system. 2. The user has access to the collaboration feature.
Normal flow of events	1. The user navigates to the "Merging" section. 2. The user selects "Create Session". 3. The system generated a unique session ID. 4. The session is initialized and made available for other users to join.
Sub Flows	3a. If required, the student may set a password or session name before creation.
Alternate Flows	-

Table 3.2.3.15 Real-Time Collaboration Use Case Description

Use Case ID	00015
Use Case Name	Real-Time Collaboration
Brief Description	Allow multiple users to collaborate on timetable planning within a shared session. Users can submit their generated timetables to be merged, adjust tutorial/practical slots, and see updates broadcast in real time. The system ensures synchronized views across all participants, supporting interactive decision-making and conflict resolution.
Actor	User
Trigger	A user joins a collaborative session and submits timetable to merge.
Precondition	1. The user must be logged into the system.

	2. A collaboration must be created or joined. 3. At least one timetable must be selected or generated before merging.
Normal flow of events	1. The user enters a collaboration session. 2. The user selects their generated timetable and submits it to merge with others. 3. The system merges submitted timetables using the GA and displays the shared timetable. 4. The user clicks on tutorial/practical slot in the shared timetable. 5. The system highlights available slot to shift the tutorial/practical. 6. The user selects an alternative slot to shift the tutorial/practical. 7. The updated timetable is broadcast simultaneously to all users in the same session. 8. All users see the synchronized timetable updates in real time.
Sub Flows	5a. If the user only wants to check available slots without shifting, they can click outside the highlighted area to close the options. 6a. If multiple students propose edits simultaneously, the system manages conflicts by applying a “last change wins”.
Alternate Flows	2a. If no timetable is selected/generated, the system notifies the student: “Please generate or select a timetable before submitting to the session.” 3a. If the merge fails due to incomplete submissions, the system prompts: “Waiting for other users to submit their timetables.” 7a. If network or synchronization issues occur, the system retries broadcasting updates until all participants are synchronized.

Table 3.2.3.16 Select Timetable Use Case Description

Use Case ID	00016
Use Case Name	Select Timetable
Brief Description	Allows user to select one of their generated timetables to submit for merging in a collaborative session. The selected timetable becomes the student's contribution to the shared merged timetable.
Actor	User
Trigger	The user clicks on a generated timetable and chooses the option to submit it for merging.
Precondition	1. The user is logged into the system. 2. The user has generated at least one timetable. 3. The user has joined or created a collaborative session.
Normal flow of events	1. The system displays a list of generated timetables for the user. 2. The user selects one timetable from the dropdown. 3. The system previews the selected timetable. 4. The user submits the selection. 5. The system stores the selected timetable as the student's submission for merging.
Sub Flows	2a. User can preview each timetable before selecting. 4a. If user changes their mind, they can change the selection.
Alternate Flows	1a. If no timetable has been generated, the system notifies: "No timetables available." 5a. If submission fails due to a system error, the system displays: "Submission failed, please try again."

Table 3.2.3.17 Session Chat Use Case Description

Use Case ID	00017
Use Case Name	Session Chat
Brief Description	Allows user within the same collaborative session to exchange messages in real time. The chat supports coordination, discussion of timetable adjustments, and decision-making among participants.
Actor	User

Trigger	The user types a message in the chat box and clicks “Send”.
Precondition	1. The user is logged into the system. 2. The user has joined a collaborative session.
Normal flow of events	1. The user enters a collaborative session. 2. The system displays a chat box linked to the session. 3. The user types a message into the chat input box. 4. The user clicks Send. 5. The system broadcasts the message to all users in the same session. 6. Other participants see the message in their chat box instantly.
Sub Flows	5a. The system timestamps each message and displays the sender’s name.
Alternate Flows	4a. If the message is empty and Send is clicked, the system ignores the action.

Table 3.2.3.18 View History Use Case Description

Use Case ID	00018
Use Case Name	View History
Brief Description	Enables user to access previously saved timetables. They can filter and view timetables by mode (manual, auto, merged), intake, and trimester.
Actor	User
Trigger	User navigates to the My History section.
Precondition	1. The user is logged into the system. 2. At least one timetable has been saved to history.
Normal flow of events	1. The user opens the My History section. 2. The system displays all saved timetables with metadata (mode, intake, trimester, timestamp, label). 3. The user applies filters if needed. 4. The system updates the displayed list accordingly. 5. The user selects a timetable to view. 6. The system displays the timetable details.

Sub Flows	2a. The system provides sorting options.
Alternate Flows	2b. If no timetable is saved, the system displays: "No saved timetables found." 5a. If the selected timetable file is corrupted or unavailable, the system notifies: "Unable to load timetable."

Table 3.2.3.19 Submit Feedback Use Case Description

Use Case ID	00019
Use Case Name	Submit Feedback
Brief Description	Allows user to submit feedback to the administrator regarding the timetable system. Users can also view responses from the admin and mark the feedback as read once reviewed.
Actor	User
Trigger	User navigates to the Feedback section and chooses to submit new feedback or view existing feedback.
Precondition	1. The user is logged into the system. 2. The feedback feature is enabled in the system.
Normal flow of events	1. The user navigates to the Feedback section. 2. The user clicks Submit Feedback. 3. The user writes a message and submits it. 4. The system saves the feedback and notifies the admin. 5. The user can later return to the Feedback section to see the admin's reply. 6. The user clicks Mark as Read after reviewing the reply. 7. The system updates the feedback status to Read.
Sub Flows	5a. The system displays both pending feedback and replied feedback in separate sections.
Alternate Flows	3b. If the feedback message is empty, the system prevents submission. 4a. If the system fails to save due to server error, system displays: "Unable to submit feedback."

Table 3.2.3.20 Upload Course Use Case Description

Use Case ID	00020
Use Case Name	Upload Course
Brief Description	Admin uploads course details in the system for a new semester. The uploaded file ensures that students have the latest course offerings available when planning their timetables.
Actor	Administrator
Trigger	Admin navigates to the admin site dashboard and selects upload course.
Precondition	1. The admin is logged into the system. 2. A valid course file is prepared according to the required format.
Normal flow of events	1. The admin navigates to the dashboard section. 2. The admin clicks the Upload Course. 3. The system prompts the admin to select a course file. 4. The admin selects the file and uploads it. 5. The system validates the file format and contents. 6. If the file is valid, the system saves the course details. 7. The system confirms: “Course upload successful!” 8. The courses are now parsed and available for students in the timetable planner.
Sub Flows	5a. The system displays a preview of the uploaded courses for admin confirmation.
Alternate Flows	4a, If no file is selected, the system cancels the upload and prompts: “Please select a file to upload.” 5c. If the format is invalid, the system displays: “Invalid file format.” 6a. If the validation fails, the system displays an error message with details for correction.

Table 3.2.3.21 View Upload History Use Case Description

Use Case ID	00021
Use Case Name	View Upload History
Brief Description	Admin can view the history of previously uploaded course files, including the timestamp of each upload and the corresponding filename. This allows tracking and verification of past course updates.
Actor	Administrator
Trigger	Admin navigates to the Upload History section from the dashboard.
Precondition	1. The admin is logged into the system. 2. At least one course file has been uploaded previously.
Normal flow of events	1. The admin navigates to the Upload History page. 2. The system retrieves all past upload records from the database. 3. The system displays the records on a table, showing: • Upload timestamp (date and time). • Filename of uploaded course file. 4. The admin reviews the history for verification.
Sub Flows	-
Alternate Flows	2a. If no history exists, the system displays: “No course uploads found.”

Table 3.2.3.22 Preview Courses Use Case Description

Use Case ID	00022
Use Case Name	Preview Courses
Brief Description	Admin can preview uploaded course details before finalizing them into the system. This ensures correctness and completeness of course data.
Actor	Administrator
Trigger	The admin selects the Preview Courses option after uploading a course file.
Precondition	1. The admin is logged into the system.

	2. A course file has been uploaded successfully.
Normal flow of events	1. The admin uploads a course file. 2. The system processes and parses the uploaded file. 3. The admin clicks Preview Courses. 4. The system displays the list of parsed courses with details. 5. The admin reviews the course details for accuracy.
Sub Flows	-
Alternate Flows	2a. If uploaded file is invalid or corrupted, the system shows an error message: “Invalid file format. Please upload a valid course file.” 4c. If no course data is found in the uploaded file, the system shows: “No course details available for preview.”

Table 3.2.3.23 View Created Sessions Use Case Description

Use Case ID	00023
Use Case Name	View Created Sessions
Brief Description	Admin can view all collaborative sessions created by users, including session details such as creation date and time, and the number of participants currently in each session.
Actor	Administrator
Trigger	The admin selects the View Created Sessions option in the admin panel.
Precondition	1. The admin is logged into the system. 2. At least one session has been created by users.
Normal flow of events	1. The admin navigates to the Created Sessions page in the admin panel. 2. The system retrieves all existing collaborative session records from the database. 3. The admin reviews the list of sessions.
Sub Flows	-
Alternate Flows	-

Table 3.2.3.24 Reply to Feedback Use Case Description

Use Case ID	00024
Use Case Name	Reply to Feedback
Brief Description	Admin can manage and reply to student feedback. The admin site provides three sections (Pending, Read, Replied) to help organize feedback and ensure timely responses.
Actor	Administrator
Trigger	The admin selects the Feedback from the dashboard.
Precondition	1. The admin is logged into the system. 2. At least one feedback has been submitted.
Normal flow of events	1. The admin navigates to the Feedback page. 2. The system displays feedback organized into three sections: • Pending: New feedback that has not yet been replied. • Read: Feedback that has been opened and read. • Replied: Feedback that has been responded to. 3. The system displays feedback details (username, message). 4. The admin types a reply and submits it. 5. The system sends the reply to the respective users and moves the feedback to the Replied section.
Sub Flows	-
Alternate Flows	2b. If no feedback exists, the system show: “No feedback available.”

3.3 Timeline

3.3.1 Overview

The timeline of the Smart Student Timetable Planner project is planned in accordance with the selected development methodology, and it spans several structural phases to ensure systematic completion.

In this initial phase, the proposal for the project was reviewed thoroughly. The problem statement and project objectives were clearly identified, followed by defining the project scope to ensure all goals can be achieved efficiently. A literature review of similar systems and websites was conducted to analyze their strengths and weaknesses. Based on this research, the tools and technologies for development were chosen. Furthermore, the project timeline was

outlined, system use cases were written, and the initial database scheme was designed and created. This phase served as the foundational stage for the rest of the development process.

This phase focused on the construction of the system's core functionalities. It began with the development of the login module, for the authentication of students and administrators. Following this, modules such as the Manual and Auto Scheduling Modules, the Genetic Algorithm and the Comparison Module were developed and completed. In parallel, FYP 1 documentation and presentation preparation were carried out.

In the User Design phase also includes future development in FYP 2 of advanced features like the Real-Time Collaboration Module, Export Timetable Module, and Administrative Module, which are scheduled for completion by mid-August.

In the final stage, the entire system will undergo thorough software testing to ensure all features function correctly and the system is free of major bugs or performance issues. During this time, the FYP 2 Report will be written and refined for submission. The project will conclude with the FYP 2 Presentation to both the supervisor and the moderator.

3.3.2 Gantt Chart

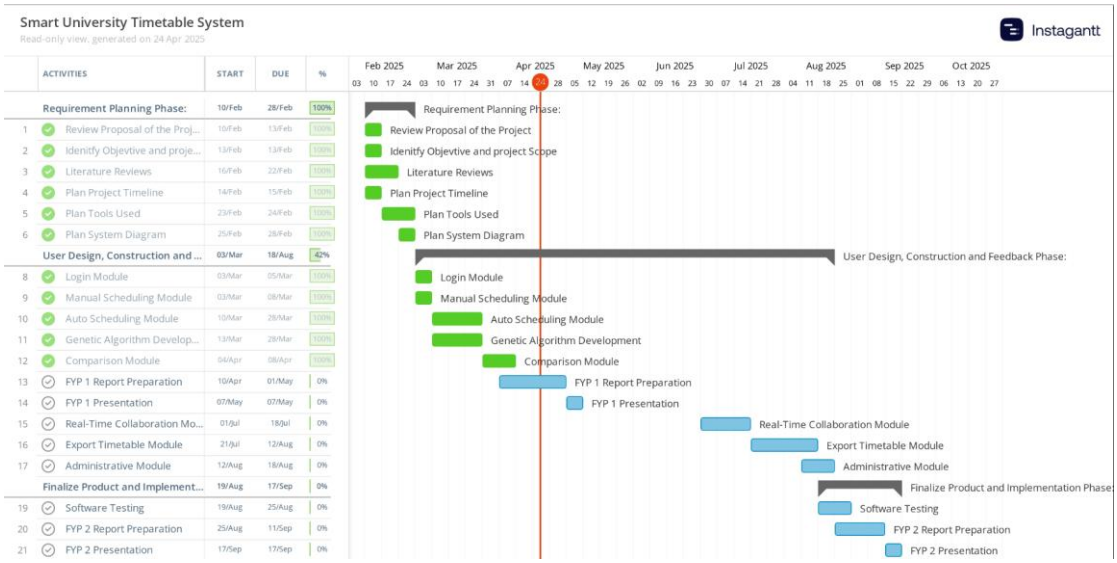


Figure 3.3.2.1 Gantt Chart of the Project Timeline

3.4 Summary

This project utilizes the Rapid Application Development (RAD) methodology, enabling flexible, iterative development with ongoing testing and easy adaptation of features throughout the process. The tools and technologies used for both front-end and back-end development are clearly outlined. Diagrams such as the activity and use case diagrams, along with detailed descriptions, illustrate the functionalities available to students and administrators. The system's development is described based on the implemented functions in the code. A Gantt chart is used to organize the project timeline, ensuring efficient time management and smooth progress without delays.

Chapter 4

System Design

In this chapter focuses on the program development which includes the server-side development, login and logout function development, manual scheduling, auto scheduling, timetable comparison, real-time collaboration, view timetable history, feedback, and for administrative module, which includes upload courses, view upload history, preview courses, view created sessions and managing feedback module.

2.1 Program Development

4.1.1 Server-side Development

The backend of the Smart Student Timetable Planner is implemented using Node.js with the Express framework, offering a lightweight and modular server architecture. The package.json file defines project metadata and dependencies such as express for routing, cors for cross-origin requests, and papaparse/csv-parser for reading and parsing CSV files. These modules enable the server to deliver static files, handle API requests, and process course data efficiently. The project is configured with a start script that launches server.js, which serves as the entry point of the application.

The core server logic in server.js begins by reading a master course list (masterlist.csv) that contains detailed course session information. This data is parsed with papaparse, transformed into structured JSON, and stored in memory to allow rapid API access. Each course entry includes lecture and tutorial/practical sessions, with lectures treated as compulsory components. Several REST API endpoints are provided to support timetable construction and data access. For instance, /api/courses returns the parsed course dataset, /api/update-session allows updates to session details such as timing or group allocation, and /generateTimetable runs a simplified Genetic Algorithm (GA) to generate non-conflicting timetables based on selected courses.

The GA implementation on the server initializes a random population of candidate timetables and applies basic evaluation rules to avoid conflicts. While currently simplified, the framework is designed to be extended with mutation, crossover, and fitness evaluation functions for more sophisticated optimization of scheduling. Static resources such as login.html and auto-

scheduling.html are served directly via Express middleware, integrating backend logic with the client-side interface.

Overall, the backend of the Smart Student Timetable Planner demonstrates a modular and extensible design. By combining structured CSV parsing, efficient API delivery, and a foundation for GA-based scheduling, it provides a robust backbone for supporting intelligent academic timetable generation and management.

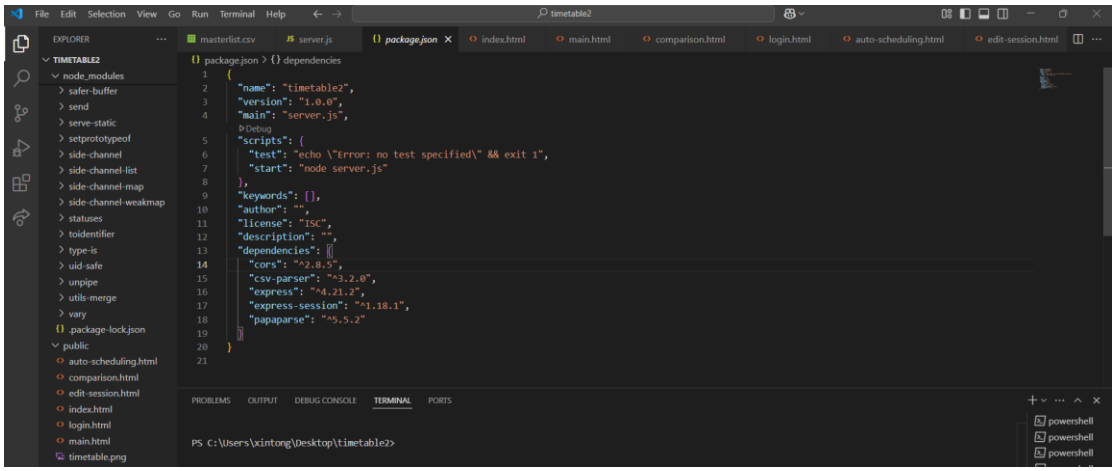


Figure 4.1.1.1 package.json

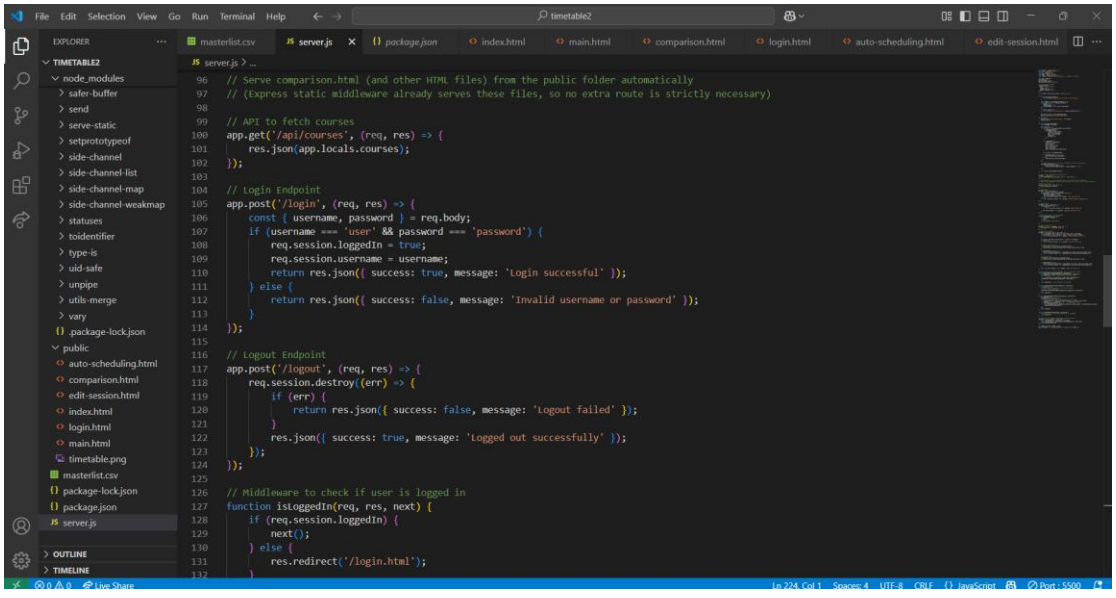


Figure 4.1.1.2 server.js

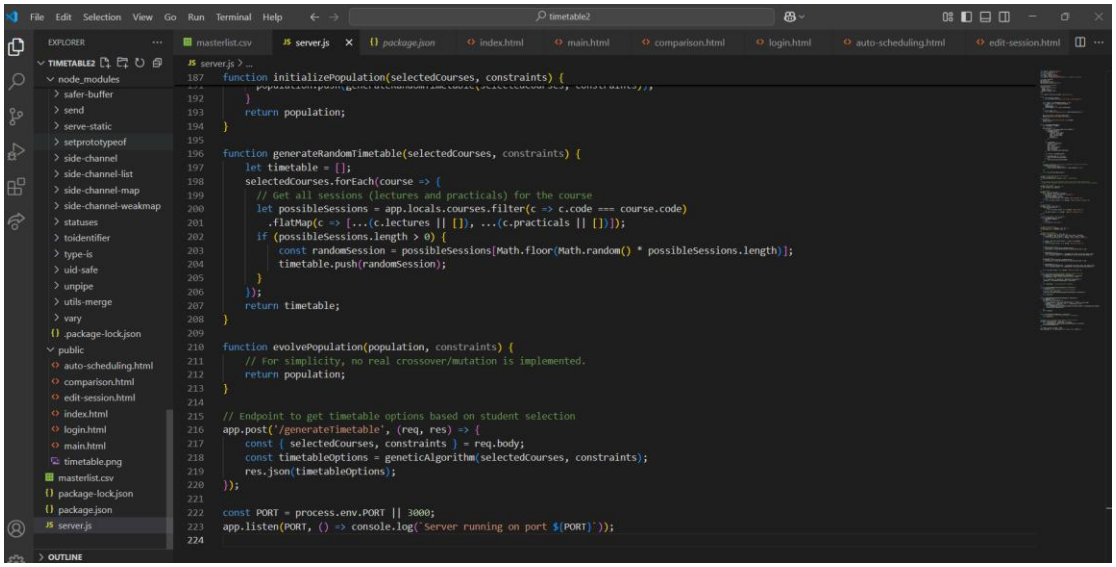


Figure 4.1.1.3 server.js

4.1.2. Login Function Development

The login functionality of the Smart Student Timetable Planner was developed to provide a secure yet seamless user authentication process. The interface is designed using HTML and CSS, ensuring a clean and responsive layout across devices. It features input fields for the username and password, role selection buttons, and a login button, all styled with padding, shadows, and rounded corners for clarity and usability. A consistent colour scheme and sticky navigation bar enhance the overall user experience, while the form is centrally aligned for accessibility on both desktop and mobile screens.

From a functional perspective, the login process is powered by a JavaScript script that manages form submission through an event listener. When a user submits the form, the script prevents the default page reload and instead collects the entered credentials together with the selected role. These details are sent via a POST request to the /login endpoint using the Fetch API. The server responds with a JSON object that indicates whether authentication was successful. Upon success, the username and role are stored in sessionStorage, and the user is redirected to either the main student page (main.html) or the admin page (admin.html) depending on their role. If the login fails, an error message is dynamically displayed, providing immediate feedback without requiring a page refresh.

On the server side, the login logic is implemented using Node.js and the Express framework. The /login endpoint verifies the submitted username, password, and role against predefined

credentials. For successful logins, a JSON response is returned containing the username and role, allowing the client to manage access control on the frontend. Logout functionality is also provided through a /logout endpoint, which destroys the user's session and clears the authentication cookie before redirecting back to the login page. Middleware functions such as isAuthenticated and isLoggedIn are implemented to demonstrate session-based access control, ensuring that only authenticated users can access protected routes like main.html or other secure resources.

Overall, the login and logout modules combine a responsive client-side design with a lightweight server-side authentication mechanism. This structure provides a solid and extensible foundation for secure user access management, while maintaining ease of use for both students and administrators.

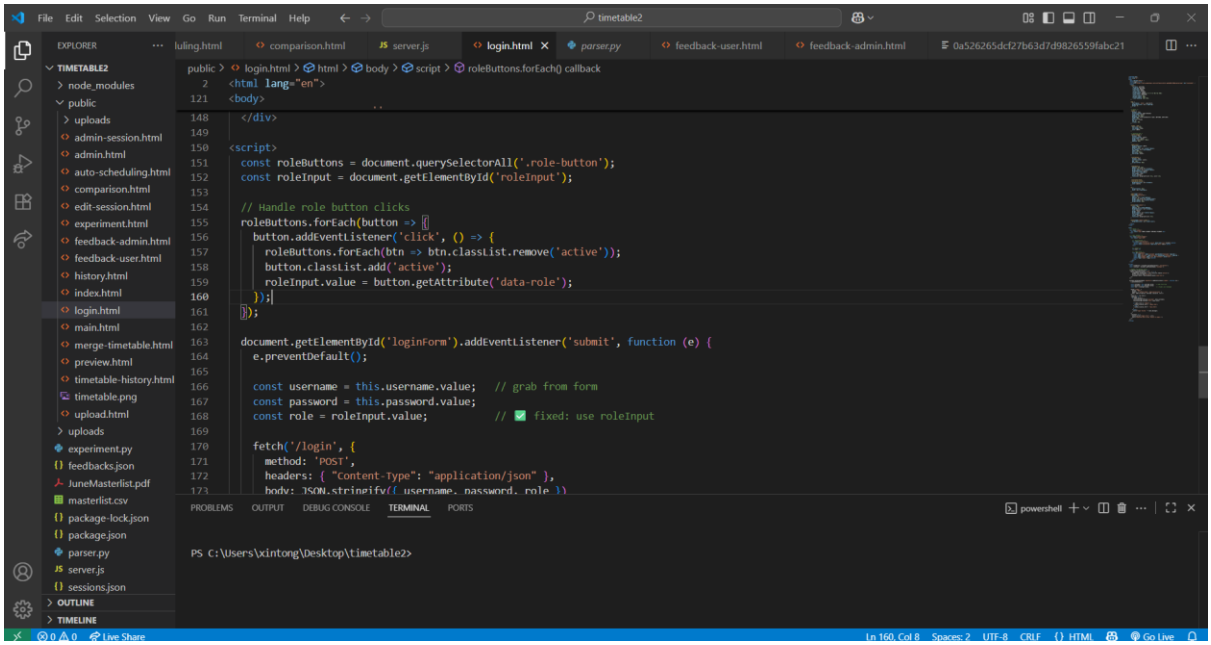


Figure 4.1.2.1 Login.html

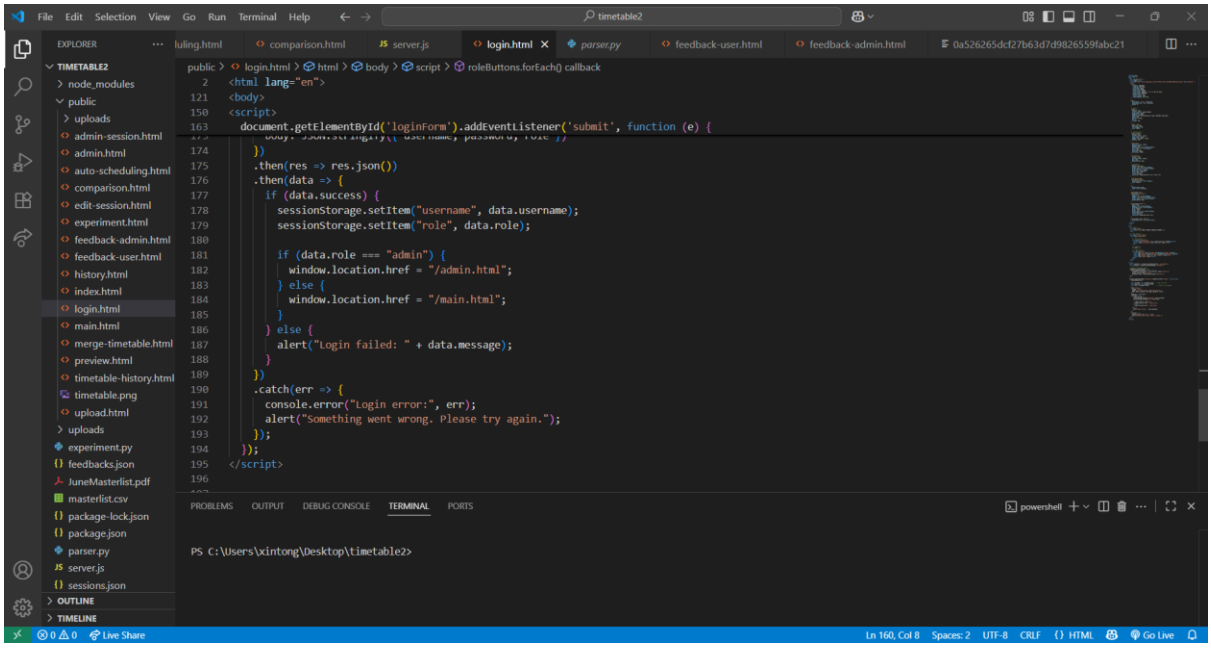


Figure 4.1.2.2 Login.html

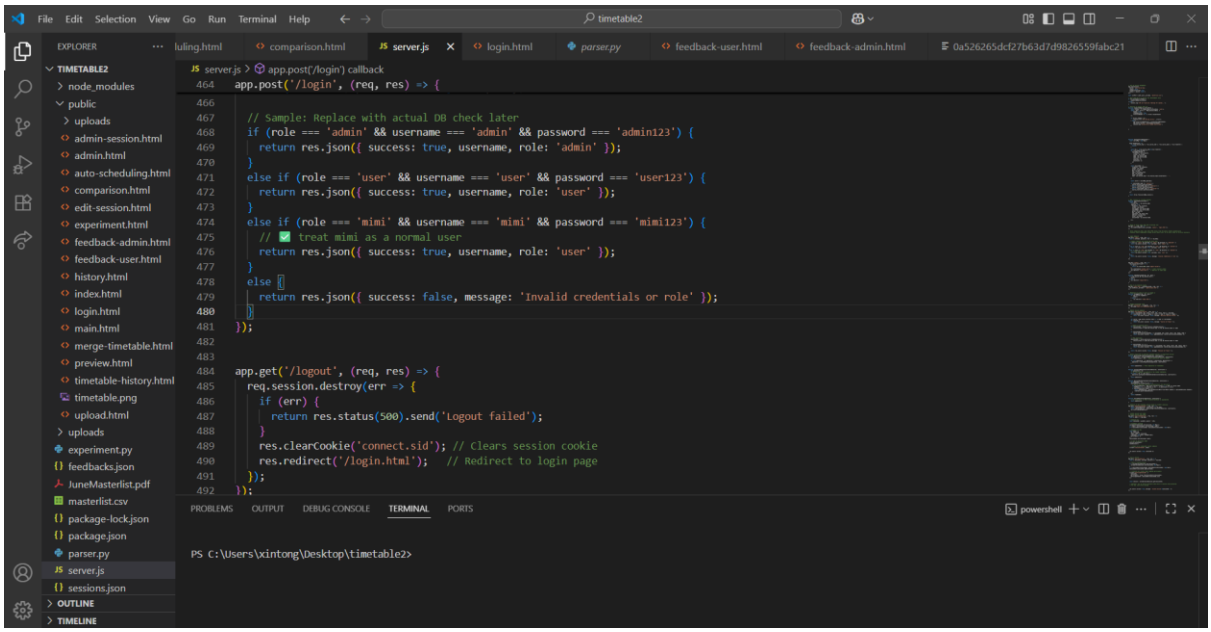


Figure 4.1.2.3 Login and Logout endpoint in server.js

4.1.3 Manual Scheduling Development

The manual scheduling module in the Smart Student Timetable Planner provides students with an interactive way to manage their course schedules. This system allows students to manually select courses choose specific sessions (lectures, tutorials, and practicals) and ensure that their schedule is free from conflicts. The design of the module ensure flexibility, while also maintaining academic rules and guidelines, making it a robust tool for course management.

When the page is first loaded, the script checks if the student has a previously saved schedule in their browser's sessionStorage. If a saved schedule exists, it retrieves the data and displays it in the schedule table, allowing the student to pick up where they left off. If there is no saved schedule, the student starts with an empty schedule. After checking for a saved schedule, the script fetches all available course data from an API endpoint (/api/courses). The list of courses is stored locally, allowing it to be filtered based on the trimester selected by the student.

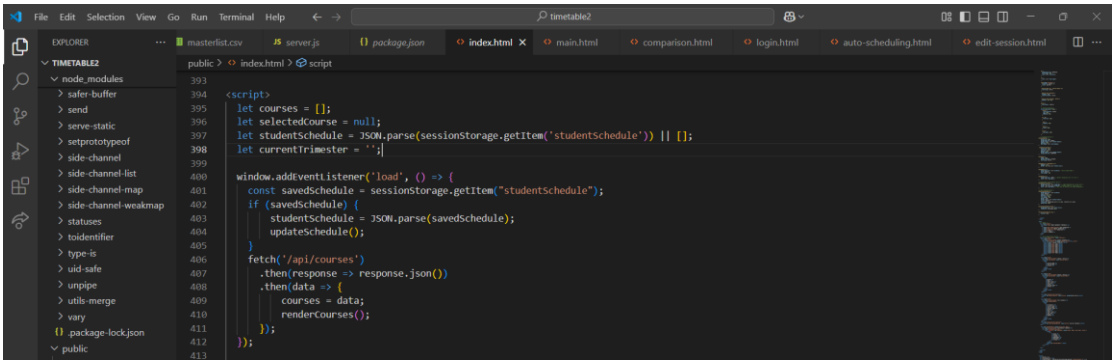
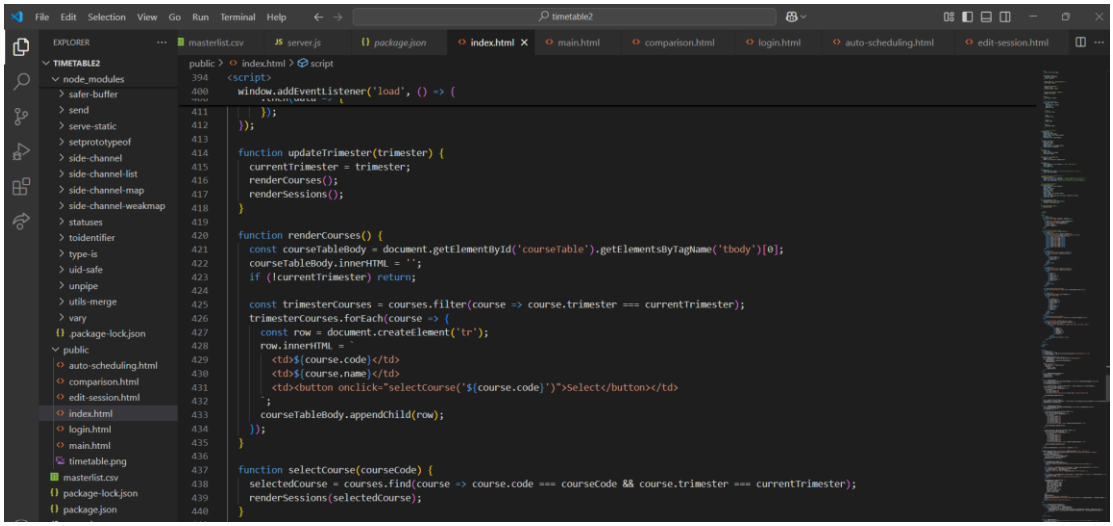


Figure 4.1.3.1 Loading courses from API

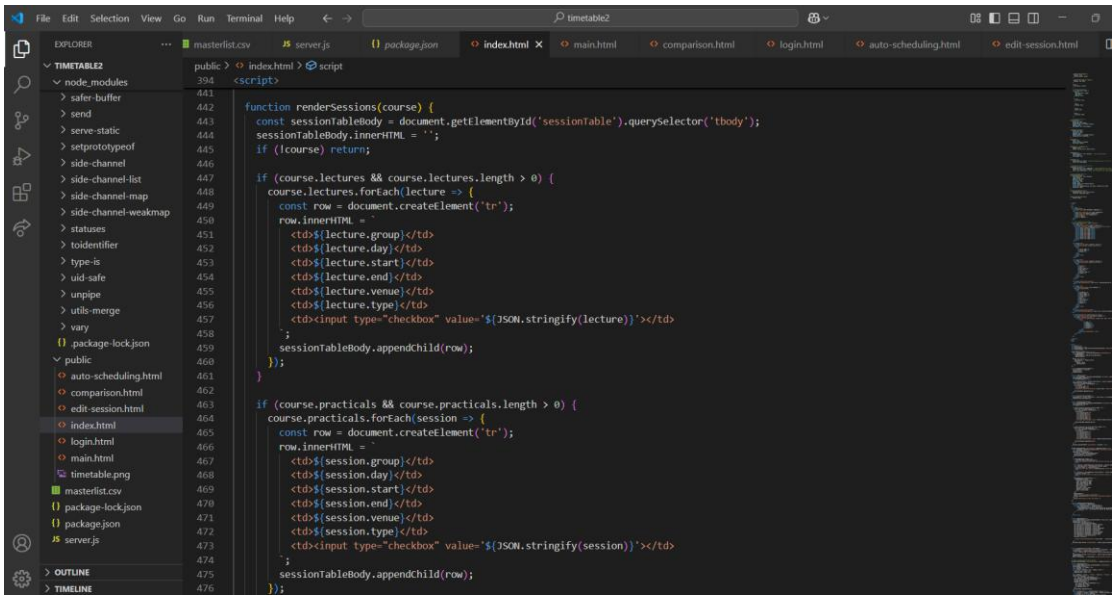
The system works based on trimesters, so the first step is for the student to select a trimester. This is done using the updateTrimester function. Once a trimester is selected, the system updates the course list to only show the courses that are offered in the chosen trimester. The list is displayed in a table format, where each course is shown with a "Select" button. When the student clicks on "Select" course, the system displays the different sessions (lectures, tutorials, and practicals) available for that course.



```
window.addEventListener('load', () => {
  });
});

function updateTrimester(trimester) {
  currentTrimester = trimester;
  renderCourses();
  renderSessions();
}

function renderCourses() {
  const courseTableBody = document.getElementById('courseTable').getElementsByTagName('tbody')[0];
  courseTableBody.innerHTML = '';
  if (!currentTrimester) return;

  const trimesterCourses = courses.filter(course => course.trimester === currentTrimester);
  trimesterCourses.forEach(course => {
    const row = document.createElement('tr');
    row.innerHTML = `
      <td>${course.code}</td>
      <td>${course.name}</td>
      <td><button onclick="selectCourse('${course.code}')">Select</button></td>
    `;
    courseTableBody.appendChild(row);
  });
}

function selectCourse(courseCode) {
  selectedCourse = courses.find(course => course.code === courseCode && course.trimester === currentTrimester);
  renderSessions(selectedCourse);
}
```

Figure 4.1.3.2 Update Trimester function

Upon selecting a course, the renderSessions function is called to display the available sessions for that course. Each session is displayed in a table with important information like the group, day, time, venue, and session type (lecture, tutorial, or practical). The student can then choose the session(s) they wish to add for by checking a checkbox next to the session. This interactive session selection allows students to pick and choose their preferred time slots.



```
function renderSessions(course) {
  const sessionTableBody = document.getElementById('sessionTable').querySelector('tbody');
  sessionTableBody.innerHTML = '';
  if (!course) return;

  if (course.lectures && course.lectures.length > 0) {
    course.lectures.forEach(lecture => {
      const row = document.createElement('tr');
      row.innerHTML = `
        <td>${lecture.group}</td>
        <td>${lecture.day}</td>
        <td>${lecture.start}</td>
        <td>${lecture.end}</td>
        <td>${lecture.venue}</td>
        <td>${lecture.type}</td>
        <td><input type="checkbox" value='${JSON.stringify(lecture)}'></td>
      `;
      sessionTableBody.appendChild(row);
    });
  }

  if (course.practicals && course.practicals.length > 0) {
    course.practicals.forEach(session => {
      const row = document.createElement('tr');
      row.innerHTML = `
        <td>${session.group}</td>
        <td>${session.day}</td>
        <td>${session.start}</td>
        <td>${session.end}</td>
        <td>${session.venue}</td>
        <td>${session.type}</td>
        <td><input type="checkbox" value='${JSON.stringify(session)}'></td>
      `;
      sessionTableBody.appendChild(row);
    });
```

Figure 4.1.3.4 Render Sessions function

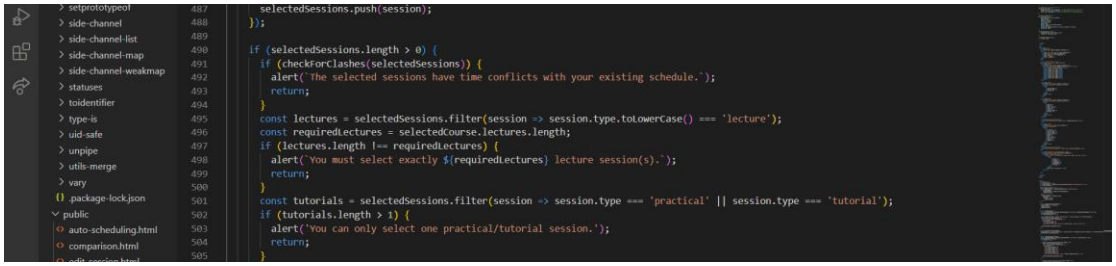
Once the student has selected their desired sessions, they can click the "Add Course" button to add the sessions to their schedule. However, before the registration is finalized, several important checks are performed:

1. Clash Detection: The system checks whether the newly selected sessions conflict with any existing sessions in the student's schedule. This is done by comparing the start and end times of the selected sessions with those already scheduled, ensuring there are no overlaps.
2. Lecture Session Validation: The system checks that the student has selected the required number of lecture sessions for the course. If a course requires multiple lectures, the student must select all of them; Otherwise, they will not be able to add for that course.
3. Practical/Tutorial Validation: The system ensures that the student selects only one practical or tutorial session for a given course. This rule is enforced to ensure that the student does not register for multiple overlapping practicals or tutorials.

If all checks pass, the selected sessions are added to the studentSchedule array, which stores the student's current timetable. This updated schedule is then displayed in the scheduled table, and the session data is saved in sessionStorage for future use.

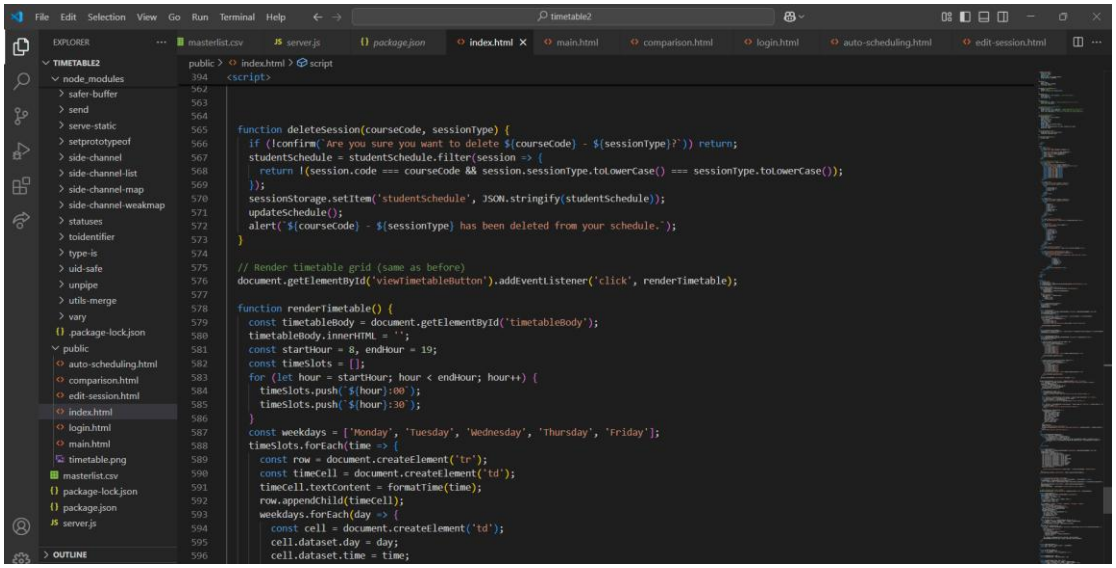
The system also offers a "View Timetable" feature, which allows the student to see their entire schedule in a grid format. The timetable span from 8:00AM to 7:00PM, with time slots in 30-minute intervals. The system dynamically generates rows and columns for each time slot, representing the days of the week (Monday to Friday). As sessions are added to the timetable, they are placed in the corresponding time slots, with the length of the session determining how many rows it will span. For example. A 2-hour session will occupy 4 rows in the timetable, and the system merges these cells to visually represent continuous sessions. The timetable is sorted by day and time, ensuring a clear and easy-to-read display. Any overlapping sessions are handled by removing redundant cells. Providing a clean, uncluttered view.

Each session that the student has registered for is displayed in the schedule table with a "Delete" button next to it. If the student wishes to remove a session, they can click this button, which prompts a confirmation message. If confirmed, the session is removed from the studentSchedule array, and the updated schedule is saved back to sessionStorage. The schedule table is then re-rendered to reflect the changes, allowing the student to manage their course load flexibly.



```
    selectedSessions.push(session);
  });

  if (selectedSessions.length > 0) {
    if (checkForClashes(selectedSessions)) {
      alert(`The selected sessions have time conflicts with your existing schedule.`);
      return;
    }
    const lectures = selectedSessions.filter(session => session.type.toLowerCase() === 'lecture');
    const requiredLectures = selectedCourse.lectures.length;
    if (lectures.length !== requiredLectures) {
      alert(`You must select exactly ${requiredLectures} lecture session(s).`);
      return;
    }
    const tutorials = selectedSessions.filter(session => session.type === 'practical' || session.type === 'tutorial');
    if (tutorials.length > 1) {
      alert('You can only select one practical/tutorial session.');
      return;
    }
```

Figure 4.1.3.4 Check Clashes Function



```
function deleteSession(courseCode, sessionType) {
  if (!confirm(`Are you sure you want to delete ${courseCode} - ${sessionType}?`)) return;
  studentSchedule = studentSchedule.filter(session => {
    return !(session.code === courseCode && session.sessionType.toLowerCase() === sessionType.toLowerCase());
  });
  sessionStorage.setItem('studentSchedule', JSON.stringify(studentSchedule));
  updateSchedule();
  alert(`${courseCode} - ${sessionType} has been deleted from your schedule.`);
}

// Render timetable grid (same as before)
document.getElementById('viewTimetableButton').addEventListener('click', renderTimetable);

function renderTimetable() {
  const timetableBody = document.getElementById('timetableBody');
  timetableBody.innerHTML = '';
  const startHour = 8, endHour = 19;
  const timeSlots = [];
  for (let hour = startHour; hour < endHour; hour++) {
    timeSlots.push(`${hour}:00`);
    timeSlots.push(`${hour}:30`);
  }
  const weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'];
  timeSlots.forEach(time => {
    const row = document.createElement('tr');
    const timeCell = document.createElement('td');
    timeCell.textContent = formatTime(time);
    row.appendChild(timeCell);
    weekdays.forEach(day => {
      const cell = document.createElement('td');
      cell.dataset.day = day;
      cell.dataset.time = time;
```

Figure 4.1.3.5 Delete Session and View Timetable Function

The Export Timetable feature was developed to allow students to generate portable copies of their manually created schedules for external use and reference. Two export formats are supported: PNG images and CSV files. The PNG export is implemented using the html2canvas library, which captures the rendered timetable grid as an image. This enables students to download a visually identical copy of their timetable, which can be stored on devices, shared with peers, or printed for offline reference. The CSV export option, on the other hand, provides a structured text-based representation of the timetable. By converting the schedule into comma-separated values, students can open and further process their timetables in spreadsheet applications such as Microsoft Excel or Google Sheets. This ensures compatibility for students who may prefer to analyse or reorganize their timetable data in tabular form. Both export options are accessible through a dropdown menu integrated into the interface, providing flexibility in how students preserve and share their schedules.

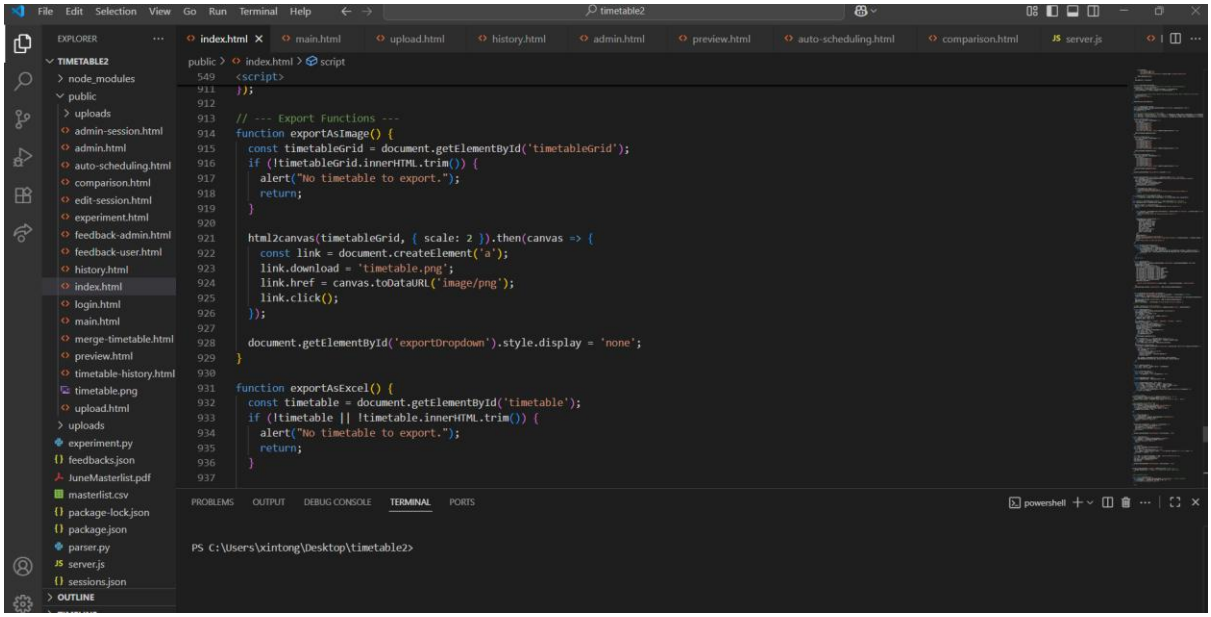


Figure 4.1.3.6 Export Timetable

The Save to History functionality extends the system's usability by maintaining a personal record of previously generated timetables. When students choose to save a timetable, the timetable grid is first captured as an image using html2canvas. The resulting image, along with metadata such as the timetable label, intake, and trimester, is sent to the backend through the /api/user/history/save endpoint. The server then stores this information in a user-specific history structure, which is retrievable even after the user logs out and logs back in. This feature ensures continuity, as students can revisit and review their past schedules without having to recreate them from scratch. It also provides a mechanism for comparison between multiple timetable versions, supporting better decision-making in course and session selection. By combining image-based storage with contextual metadata, the Save to History feature offers both visual clarity and organizational efficiency in timetable management.

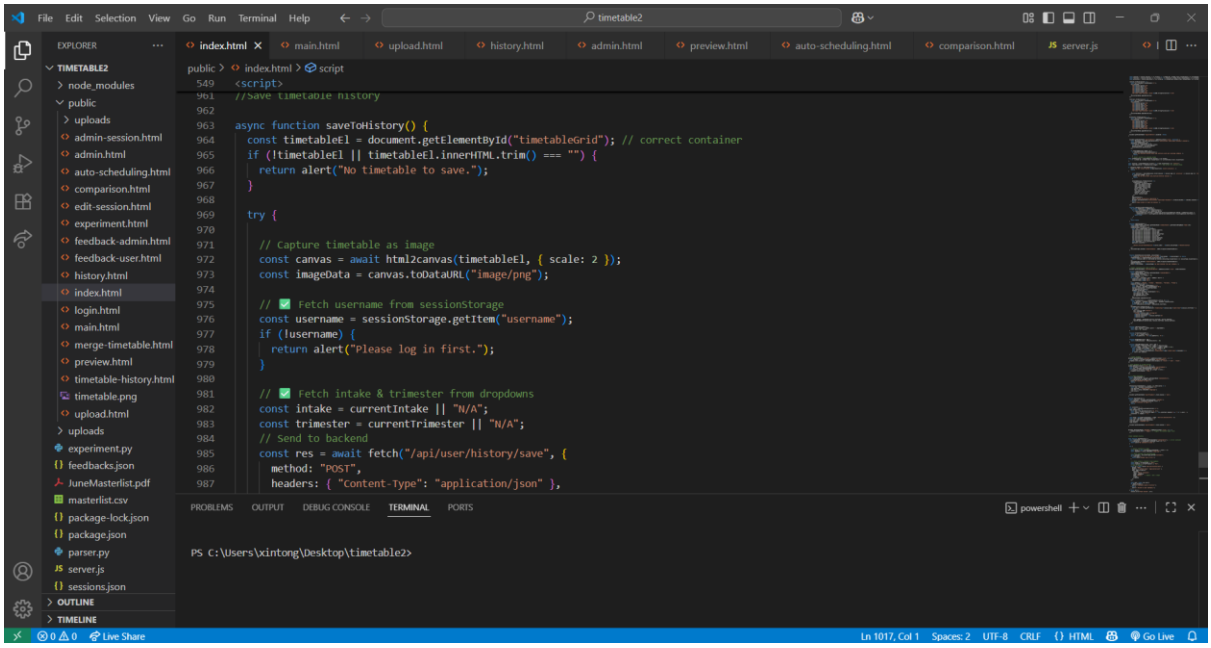


Figure 4.1.3.7 Save Timetable to History

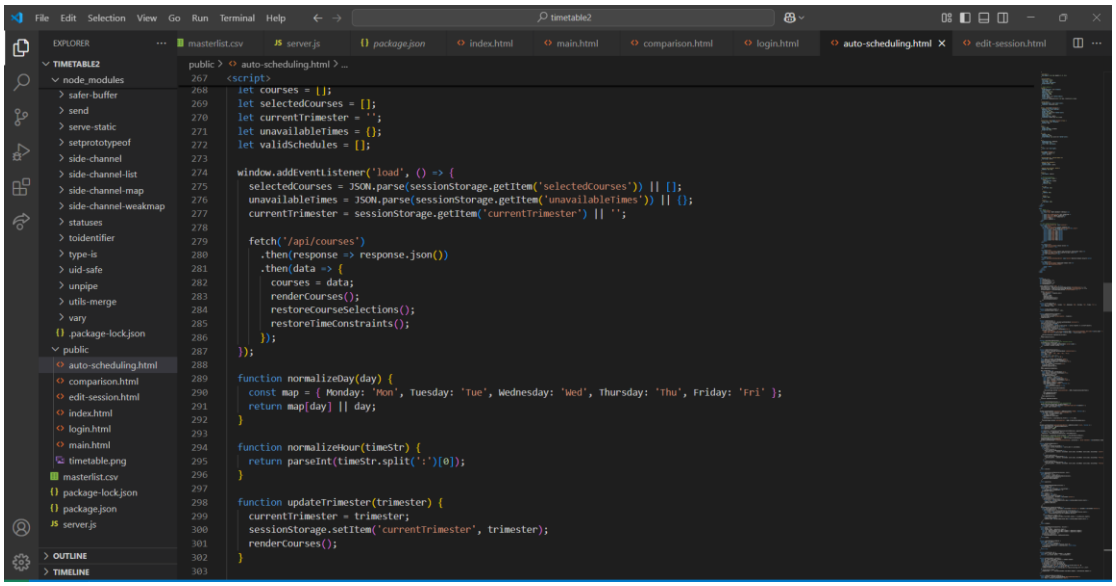
4.1.4 Auto Scheduling Development

In the auto scheduling section, it is designed to provide students with an automated way to generate valid and personalized course schedules based on selected courses and individual time constraints. This file forms the user interface where students can interact with the scheduling system, input their preferences, and visualize the generated timetables. The system is tailored to address common student needs such as avoiding scheduling conflicts, respecting unavailable time slots, and ensuring complete course registration within a given academic trimester.

Upon loading, the HTML page retrieves various data stored in the browser's sessionStorage, which may include the selected courses, the current trimester, and the student's unavailable time slots. This data is used to render course options dynamically for the selected trimester, allowing students to mark the courses they wish to enrol in. Each course selection is captured and stored to maintain consistency across user sessions and page refreshes. Moreover, the page incorporates a time constraint selection interface, implemented as a grid of checkboxes that represent time slots across weekdays and working hours. This grid allows students to specify their unavailable periods during the week, such as when they may have part-time jobs or personal commitments.

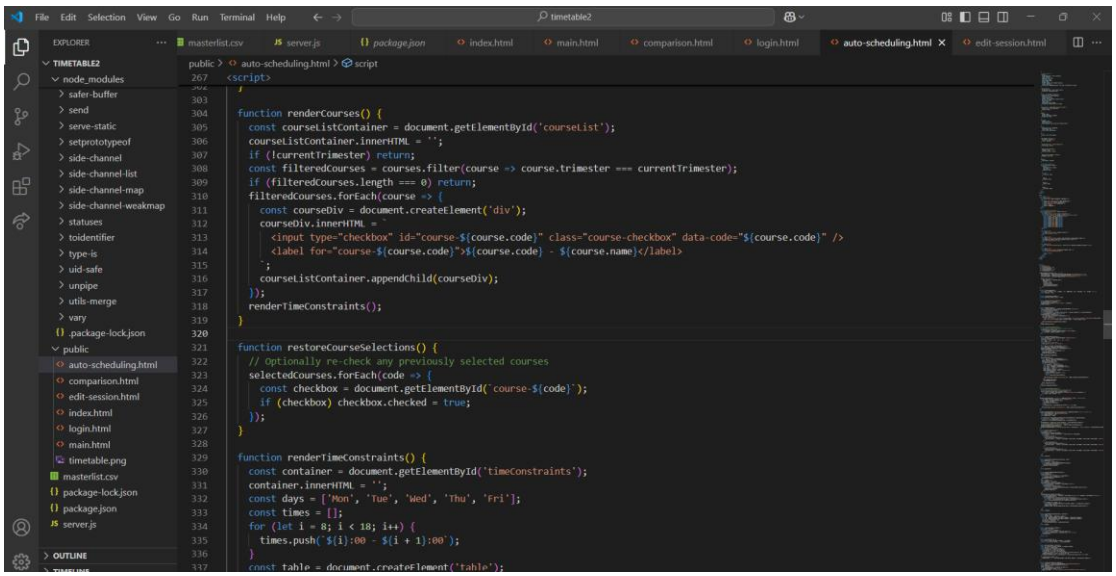
A critical feature of this page is the "Generate Schedule" button, which initiates the scheduling process. When clicked, the system collects all the relevant data, selected courses and

unavailable times, and processes it through a Genetic Algorithm (GA). The algorithm then generates multiple valid timetable combinations that meet all defined constraints. These schedules are displayed to the user in a structured timetable format, with course session visually arranged in a grid representing days and times. Each session is color-coded and labelled to indicate the course code, session type (lecture, tutorial, or practical), and location details.



```
let courses = [];
let selectedCourses = [];
let currentTrimester = '';
let unavailableTimes = {};
let validSchedules = [];

window.addEventListener('load', () => {
  selectedCourses = JSON.parse(sessionStorage.getItem('selectedCourses')) || [];
  unavailableTimes = JSON.parse(sessionStorage.getItem('unavailableTimes')) || {};
  currentTrimester = sessionStorage.getItem('currentTrimester') || '';

  fetch('/api/courses')
    .then(response => response.json())
    .then(data => {
      courses = data;
      renderCourses();
      restoreCourseSelections();
      restoreTimeConstraints();
    });
});

function normalizeDay(day) {
  const map = { Monday: 'Mon', Tuesday: 'Tue', Wednesday: 'Wed', Thursday: 'Thu', Friday: 'Fri' };
  return map[day] || day;
}

function normalizeHour(timeStr) {
  return parseInt(timeStr.split(':')[0]);
}

function updateTrimester(trimester) {
  currentTrimester = trimester;
  sessionStorage.setItem('currentTrimester', trimester);
  renderCourses();
}
```

Figure 4.1.4.1 Update Trimester and Fetch Courses from API



```
function renderCourses() {
  const courseListContainer = document.getElementById('courseList');
  courseListContainer.innerHTML = '';
  if (!currentTrimester) return;
  const filteredCourses = courses.filter(course => course.trimester === currentTrimester);
  if (filteredCourses.length === 0) return;
  filteredCourses.forEach(course => {
    const courseDiv = document.createElement('div');
    courseDiv.innerHTML = `
      <input type="checkbox" id="course-${course.code}" class="course-checkbox" data-code="${course.code}" />
      <label for="course-${course.code}">${course.code} - ${course.name}</label>
    `;
    courseListContainer.appendChild(courseDiv);
  });
  renderTimeConstraints();
}

function restoreCourseSelections() {
  // Optionally re-check any previously selected courses
  selectedCourses.forEach(code => {
    const checkbox = document.getElementById(`course-${code}`);
    if (checkbox) checkbox.checked = true;
  });
}

function renderTimeConstraints() {
  const container = document.getElementById('timeConstraints');
  container.innerHTML = '';
  const days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri'];
  const times = [];
  for (let i = 8; i < 18; i++) {
    times.push(`${i}:00 - ${i + 1}:00`);
  }
  const table = document.createElement('table');
```

Figure 4.1.4.2 Render Course and Render Time Constraints function

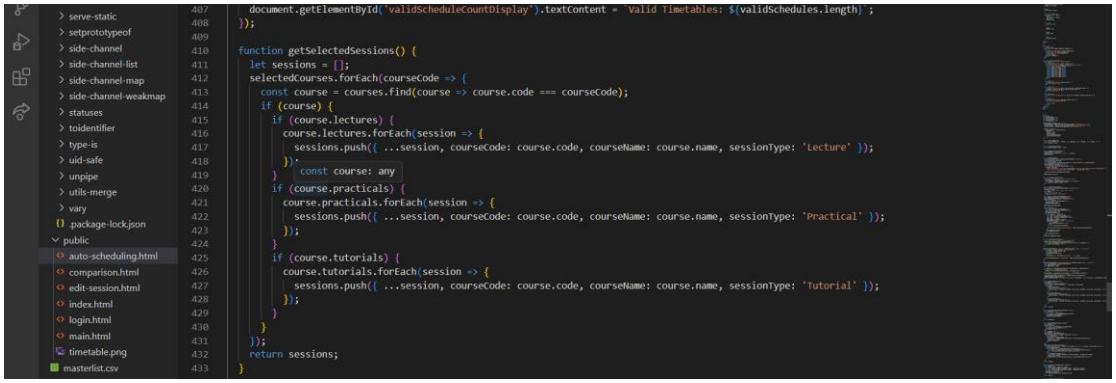


Figure 4.1.4.3 Get Selected Sessions Function

The Export Timetable feature allows students to preserve their generated timetables in two different formats: PNG images and CSV files. When the export as image option is chosen, the timetable grid is first cloned and placed in an off-screen container to ensure a clean layout. The html2canvas library is then used to capture the table and render it into a high-resolution canvas. This canvas is converted into a downloadable PNG file, enabling students to save, print, or share their timetable in its original visual form. In contrast, the CSV export option provides a structured representation of the data timetable. The timetable is processed into a grid covering all hours and weekdays, with each session represented by its course code and type. The grid is converted into comma-separated values and packaged as a downloadable file that can be opened in applications such as Microsoft Excel or Google Sheets. By offering both visual and data-driven exports, the system ensures flexibility, catering to students who prefer either a quick visual reference or a structured dataset for further analysis.

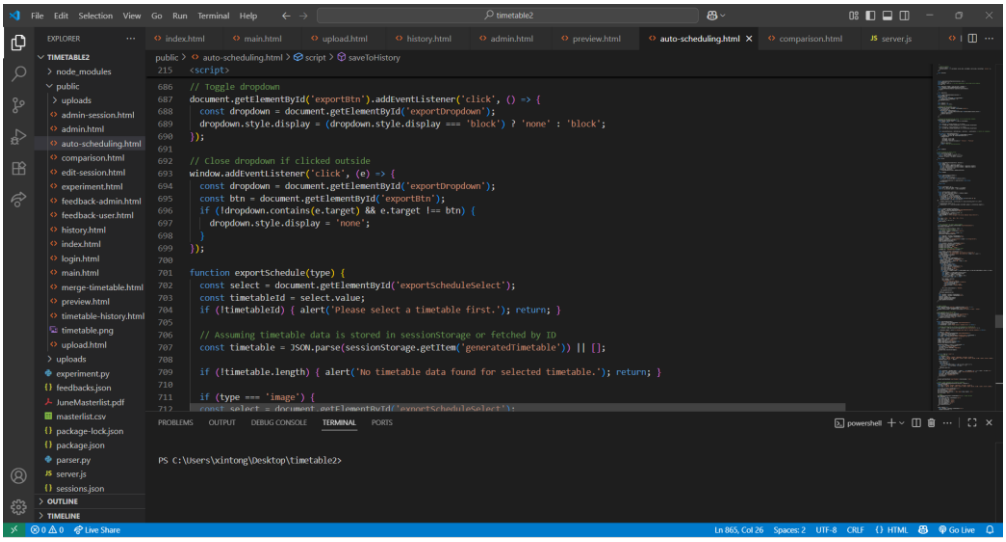


Figure 4.1.4.4 Export Timetable

The Save to History feature enhances the planner by maintaining a record of timetables within the system itself. When students choose to save a timetable, the application checks for a valid user session before processing each generated schedule. For every timetable, the grid is cloned and captured into a high-quality image using html2canvas. This image, together with metadata such as username, label, intake, trimester, and mode, is transmitted to the backend via the /api/user/history/save endpoint. The server then stores the data in a user-specific history record, ensuring timetables are linked to the correct account. Saved timetables remain accessible even after the user logs out and logs back in, allowing students to revisit or compare multiple versions without having to regenerate them. By combining visual accuracy with contextual metadata, this feature provides both continuity and organization, supporting better decision-making in timetable management.

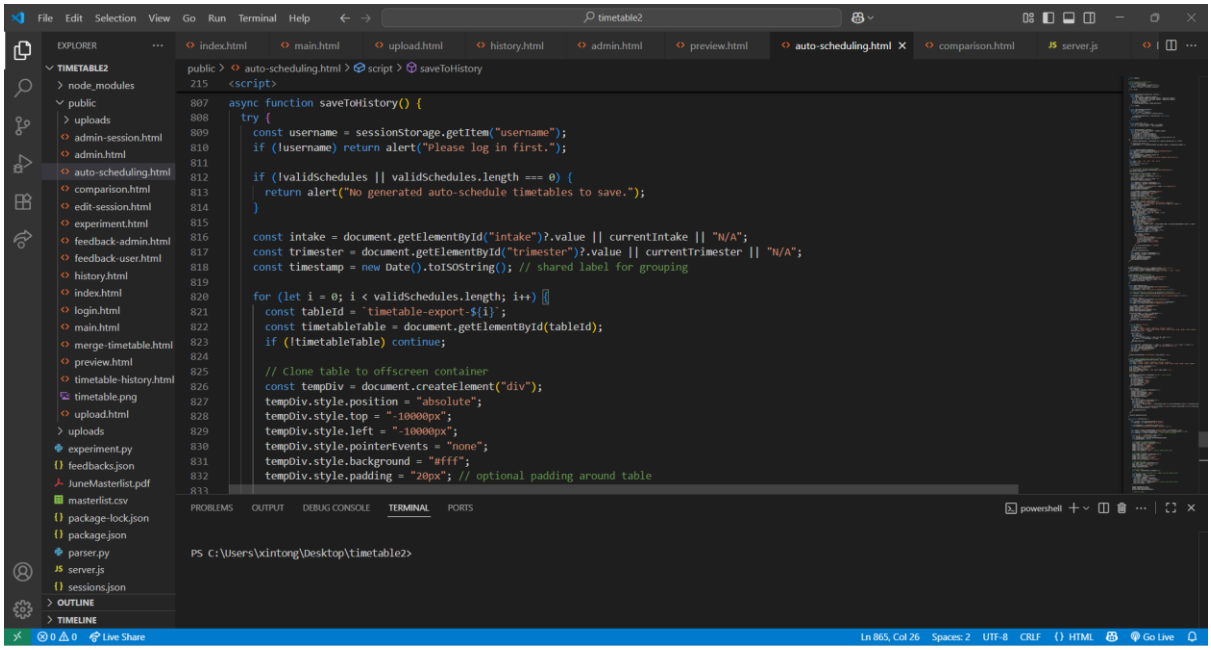


Figure 4.1.4.5 Save Timetable to History

4.1.5 Genetic Algorithm Development

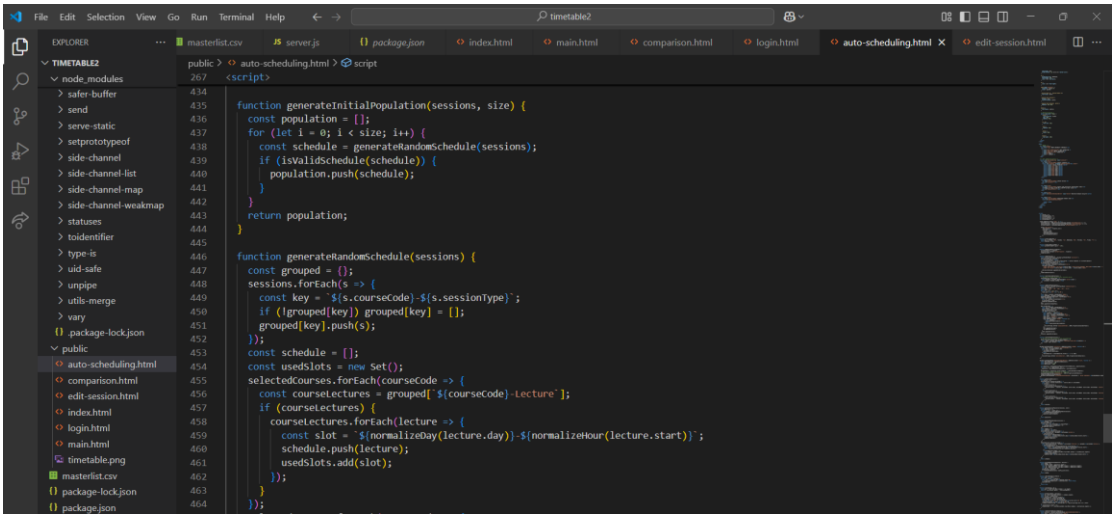
The core logic for timetable generation relies on a Genetic Algorithm (GA), a search heuristic inspired by the process of natural selection. The GA operates by iteratively improving a population of candidate solutions. In this case, course schedules, based on fitness criteria that reflects the validity of a schedule. This evolutionary approach is well-suited to complex scheduling problems, where the solution space is vast and includes numerous constraints.

The process begins with the generation of an initial population of random schedules. Each schedule is a combination of lecture sessions (which are mandatory for each course) and either one tutorial or one practical session. The function generateInitialPopulation() is responsible for creating these candidate schedules. It ensures that each schedule includes all required lecture sessions and exactly one non-lecture session for each course, provided there is no conflict with the student's unavailable time or with other sessions in the schedule.

The fitness of each schedule is evaluated using the isValidSchedule() function. This function ensures that no two sessions in a schedule overlap and that none of the sessions conflict with the student's declared unavailable slots. Unlike traditional GAs that assign a numeric fitness score, this implementation uses a binary evaluation where a schedule is either valid or invalid. This simplification is effective in pruning the search space and focusing only on feasible timetables.

The evolution of the population occurs through a combination of crossover and mutation operations, implemented within te evolvePopulation () function. In each generation, the algorithm selects pairs of valid parent schedules and produces offspring by combining their session lists at a randomly selected crossover point. This recombination helps in exploring new combinations of session arrangements. To introduce diversity and avoid premature convergence, the algorithm also applies mutations to some schedules. The mutate () function randomly replaces a session with an alternative option of the same type, provided the change maintains schedule validity.

The evolution process continues for a fixed number of generations, during which new valid schedules are produced and stored. At the end of the evolution process, a pool of valid and optimized schedules is available. These schedules are then stored in sessionStorage and rendered on the interface using the renderAllSchedules() function. The visual representation includes up to five valid schedules displayed as weekly timetables, each showing course sessions in their appropriate time slots. Each session is clearly labelled with its code, type, group, and venue, allowing students to make informed decisions.

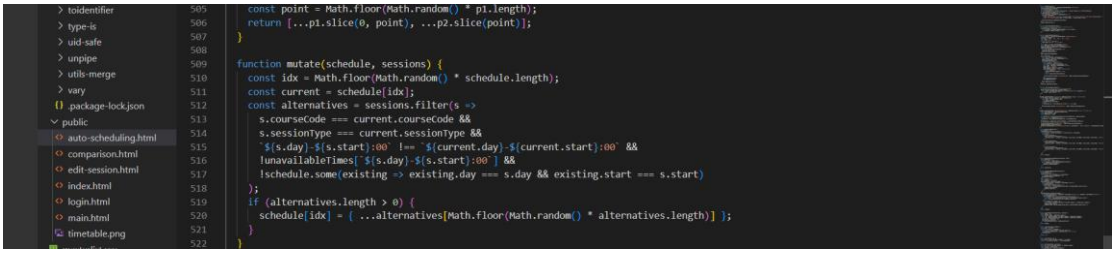


```
function generateInitialPopulation(sessions, size) {
  const population = [];
  for (let i = 0; i < size; i++) {
    const schedule = generateRandomSchedule(sessions);
    if (isValidSchedule(schedule)) {
      population.push(schedule);
    }
  }
  return population;
}

function generateRandomSchedule(sessions) {
  const grouped = {};
  sessions.forEach(s => {
    const key = `${s.courseCode}-${s.sessionType}`;
    if (!grouped[key]) grouped[key] = [];
    grouped[key].push(s);
  });
  const schedule = [];
  const usedSlots = new Set();
  selectedCourses.forEach(courseCode => {
    const courseLectures = grouped[`${courseCode}-Lecture`];
    if (courseLectures) {
      courseLectures.forEach(lecture => {
        const slot = `${normalizeDay(lecture.day)}-${normalizeHour(lecture.start)}`;
        schedule.push(lecture);
        usedSlots.add(slot);
      });
    }
  });
```

Figure 4.1.5.1 Generate Initial Population and Generate Random Schedule function

```
  return schedule;
}

function evolvePopulation(population, sessions) {
  const newPop = [];
  while (newPop.length < population.length) {
    const p1 = population[Math.floor(Math.random() * population.length)];
    const p2 = population[Math.floor(Math.random() * population.length)];
    let child = crossover(p1, p2);
    mutate(child, sessions);
    if (isValidSchedule(child)) newPop.push(child);
  }
  return newPop;
}

function isValidSchedule(schedule) {
  const used = new Set();
  for (const session of schedule) {
    const slot = `${session.day}-${session.start}:00`;
    if (unavailableTimes[slot] || used.has(slot)) return false;
    used.add(slot);
  }
  return true;
}

function crossover(p1, p2) {
  const point = Math.floor(Math.random() * p1.length);
  return [...p1.slice(0, point), ...p2.slice(point)];
}
```

Figure 4.1.5.2 Evolve Population and Crossover function



```
  const point = Math.floor(Math.random() * p1.length);
  return [...p1.slice(0, point), ...p2.slice(point)];
}

function mutate(schedule, sessions) {
  const idx = Math.floor(Math.random() * schedule.length);
  const current = schedule[idx];
  const alternatives = sessions.filter(s =>
    s.courseCode === current.courseCode &&
    s.sessionType === current.sessionType &&
    `${s.day}-${s.start}:00` !== `${current.day}-${current.start}:00` &&
    !unavailableTimes[`${s.day}-${s.start}:00`] &&
    !schedule.some(existing => existing.day === s.day && existing.start === s.start)
  );
  if (alternatives.length > 0) {
    schedule[idx] = { ...alternatives[Math.floor(Math.random() * alternatives.length)] };
  }
}
```

Figure 4.1.5.3 Mutate function

4.1.6 Timetable Comparison Development

This module serves the specific function of visually comparing a student's manually registered timetable against a set of auto-generated timetables. This page acts as a decision-support tool for students by allowing them to evaluate and compare different scheduling options before finalizing their course registration. It is an integral component of the Smart University

Timetable System, which offers both manual and Genetic Algorithm (GA)-based scheduling modes.

Several utility functions are defined to support time parsing and data normalization across different session formats. These functions ensure compatibility between manual and auto-generated session data, as two types may use slightly different field names. Function such as getSessionDay, getSessionStart, getSessionEnd, and getSessionCourseCode abstract away these differences, providing a unified interface for retrieving session attributes. Additionally, time-handling utilities like toMinutes and formatHour are employed to calculate durations and display time ranges in a readable format.

This page loads both the manual and aut-generated schedules from sessionStorage, where they were previously stored during the scheduling processes in other parts of the system. The loadComaprison() function serves as the entry point when the page is loaded, retrieving and parsing these stored schedule arrays for further processing and rendering.

The core rendering logic is encapsulated in the renderTimetableGrid() function, which constructs a visual timetable in HTML table format based on a given schedule array. The function generated a grid where each row represents an hourly time slot, and each column corresponds to a weekday. For each cell in the grid, it checks whether a course session is scheduled to start at that day and hour. If so, the session is inserted into the grid with appropriate rowSpan values to reflect its duration in hours.

To handle overlapping sessions and avoid duplicate rendering, the script maintains an internal cellOccupied map. This structure tracks cells that are already occupied by longer sessions and ensures that merged cells spanning multiple rows are rendered correctly without interference. Each session is labelled with details including the course code, session type, group number, start and end times, and venue. These details are styled with varied font sizes to maintain clarity and compactness within each cell.

Two distinct rendering pathways are defined for manual and auto-generated timetables. The renderManualTimetable() function specifically loads the student's registered schedule from

sessionStorage under the key “studentSchedule” and renders it using the timetable grid function. If no such data is found, a fallback message “No registered timetable found” is displayed.

On the other hand, auto-generated schedules are handled by the renderAutoTimetables() function. This function introduces pagination logic to support the display of multiple valid schedules generated by the Genetic Algorithm. Only a maximum of five timetables is rendered, and the user can navigate through them using “Next” and “Previous” buttons. The currently displayed timetable is labelled to distinguish it from others. Each time a new schedule is rendered, the pagination status is updated using updatePagination() to reflect the current page number and enable or disable the navigation buttons as needed.

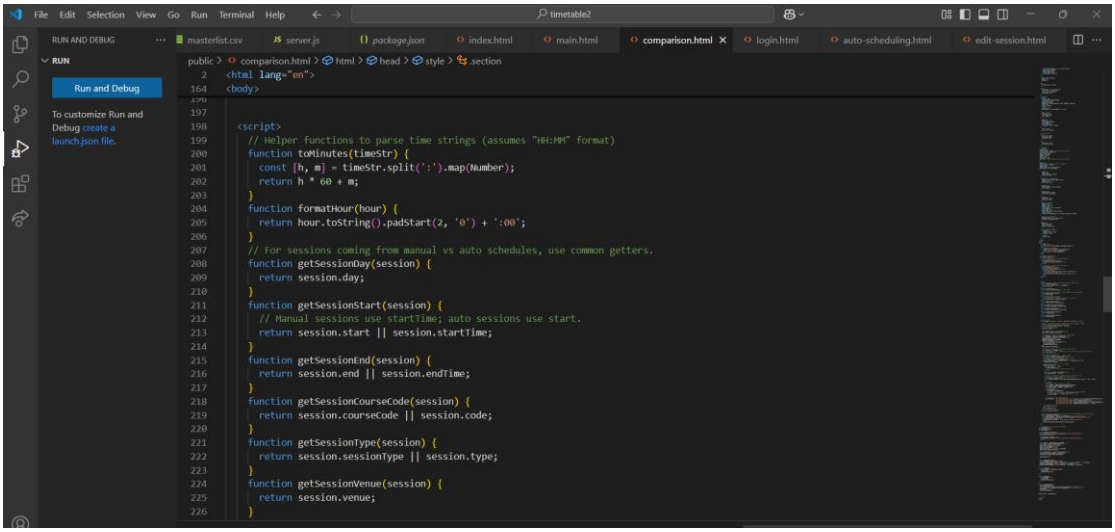
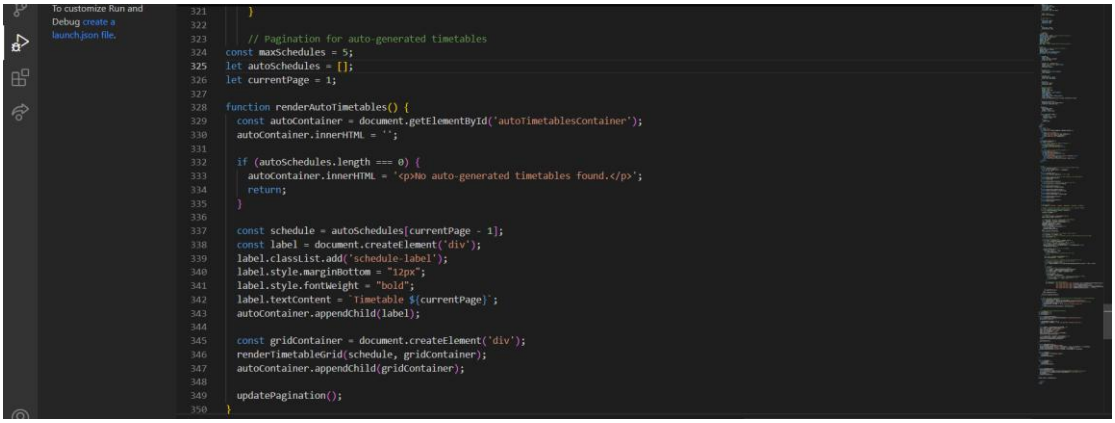


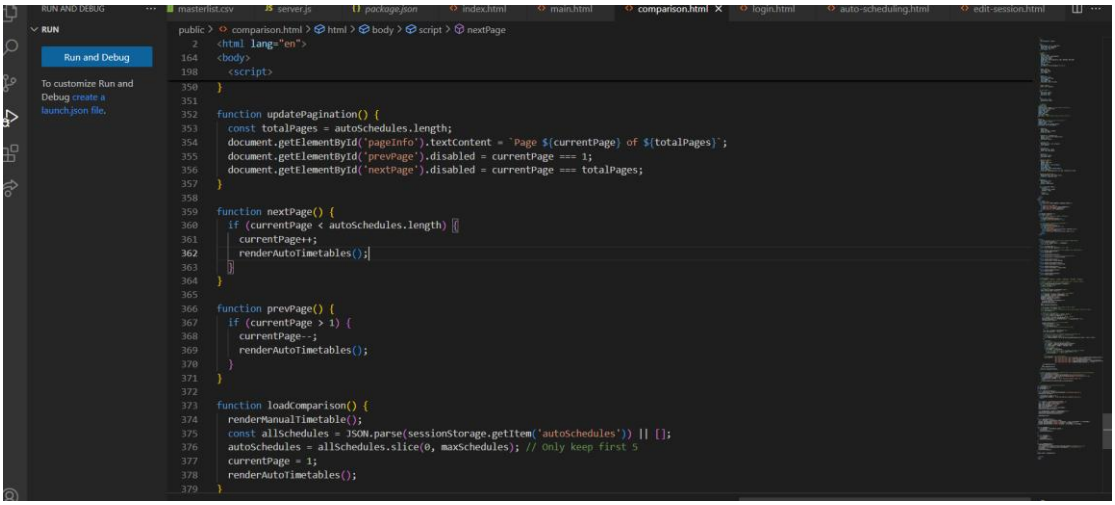
Figure 4.1.6.1 Normalize Getters Function

Figure 4.1.6.2 Render Manual Timetable



```
    }

    // Pagination for auto-generated timetables
const maxSchedules = 5;
let autoSchedules = [];
let currentPage = 1;

function renderAutoTimetables() {
  const autoContainer = document.getElementById('autoTimetablesContainer');
  autoContainer.innerHTML = '';

  if (autoSchedules.length === 0) {
    autoContainer.innerHTML = '<p>No auto-generated timetables found.</p>';
    return;
  }

  const schedule = autoSchedules[currentPage - 1];
  const label = document.createElement('div');
  label.classList.add('schedule-label');
  label.style.marginBottom = "12px";
  label.style.fontWeight = "bold";
  label.textContent = `Timetable ${currentPage}`;
  autoContainer.appendChild(label);

  const gridContainer = document.createElement('div');
  renderTimetableGrid(schedule, gridContainer);
  autoContainer.appendChild(gridContainer);

  updatePagination();
}
```

Figure 4.1.6.3 Render Auto Generated Timetable



```
function updatePagination() {
  const totalPages = autoSchedules.length;
  document.getElementById('pageInfo').textContent = `Page ${currentPage} of ${totalPages}`;
  document.getElementById('prevPage').disabled = currentPage === 1;
  document.getElementById('nextPage').disabled = currentPage === totalPages;
}

function nextPage() {
  if (currentPage < autoSchedules.length) {
    currentPage++;
    renderAutoTimetables();
  }
}

function prevPage() {
  if (currentPage > 1) {
    currentPage--;
    renderAutoTimetables();
  }
}

function loadComparison() {
  renderManualTimetable();
  const allSchedules = JSON.parse(sessionStorage.getItem('autoSchedules')) || [];
  autoSchedules = allSchedules.slice(0, maxSchedules); // Only keep first 5
  currentPage = 1;
  renderAutoTimetables();
}
```

Figure 4.1.6.4 Update Pagination and Load Comparison functions

4.1.7 PDF Parser Development

This program was developed to automate the conversion of timetable data from PDF format into a structured CSV file that can be easily processed by the Smart Student Timetable Planner system. It begins by handling command-line arguments to ensure that both the input PDF and output CSV file paths are provided. Using the pdfplumber library, the program opens the PDF file and scans each page for tables. Every table is then broken down into rows, and only non-empty rows are collected. During this process, empty cells are replaced with blank strings, and extra whitespace is trimmed to keep the data clean and consistent. This ensures that only meaningful data is extracted while avoiding formatting issues caused by messy PDF structures.

Once the data has been gathered, the program assumes the first row of the extracted table is the header, while the remaining rows are treated as the dataset. A pandas DataFrame is created from this structure, which provides a reliable way to handle tabular data. Finally, the

DataFrame is exported to a CSV file without including row indices, making the output ready for integration into the timetable system. This approach not only simplifies the process of converting complex PDFs into usable data but also ensures that the information is standardized, clean, and accessible for further operations such as scheduling, searching, and conflict checking in the application.

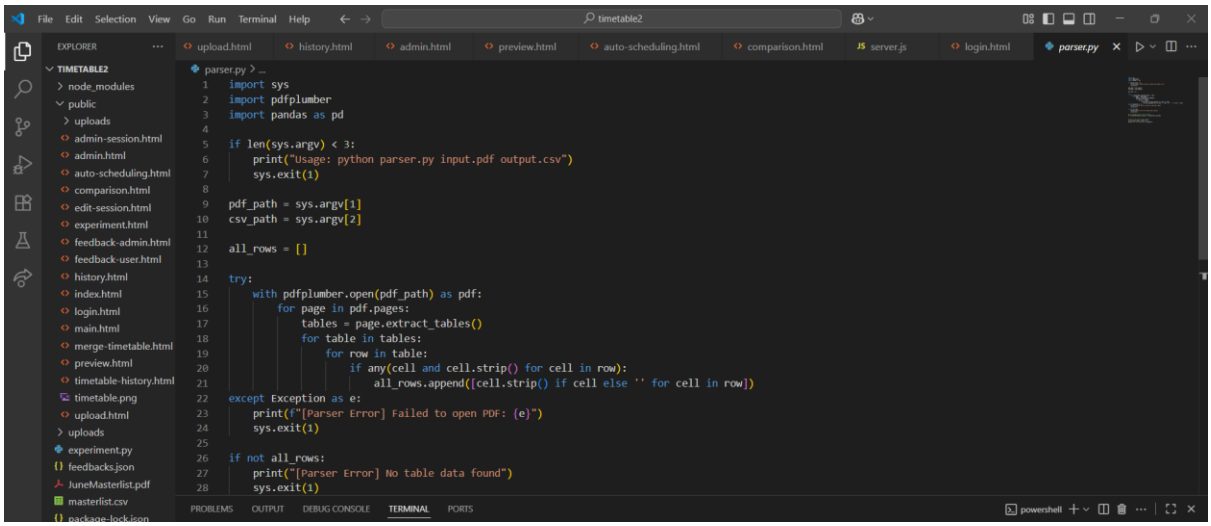


Figure 4.1.7.1 PDF Parser in parser.py

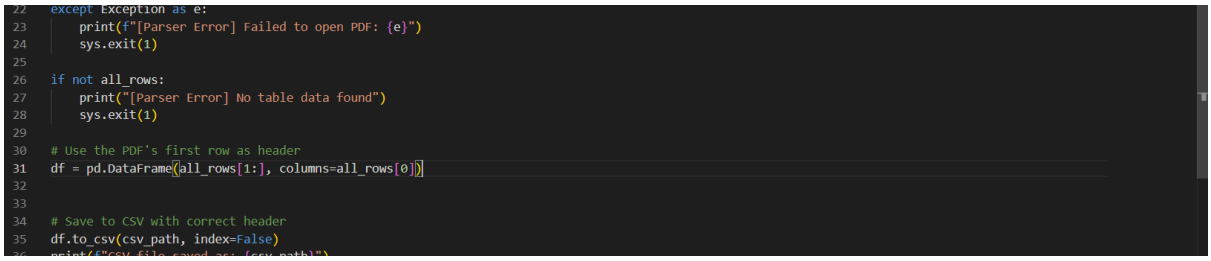


Figure 4.1.7.2 PDF Parser in parser.py

4.1.8 Real-time Collaborative Development

This client-side script implements the real-time collaboration UI for the merged timetable feature. It connects to the backend via Socket.IO, lets users create/join collaborative sessions, submit their personal timetables, preview others' timetables, and participate in an interactive merge process where users (or the admin) can click to assign alternative timeslots for tutorial/practical sessions. The script keeps a local copy of course/session data, maintains session state in sessionStorage, renders both full merged grids and compact mini-previews, and provides export / save-to-history capabilities (image snapshot via html2canvas).

When the page loads the script opens a persistent socket with const socket = io(). User and session identity are retrieved from sessionStorage (username, collabSessionId). Global datasets include window.courses (the unified course/session masterlist fetched from /api/courses), allSessions (the server-provided list of merged/updated sessions), mergedTimetable (the timetable currently displayed), and window.allUserTimetables for storing each user's submitted timetables. sessionSubmissions (a Map) and some small state variables (mode, validSchedules, userTimetables) are used to track submission status and ephemeral UI state. This layout favors client-side rendering and immediate feedback while relying on the server for authoritative session synchronization and synthesis.

The client both listens for and emits events. Important inbound events include:

- sessionState — server sends full session state (allSessions + submissions) so the client can do a full sync and render the timetable and submission list.
- sessionSubmissionsUpdated — updates the per-user submission indicator UI.
- allUserSubmissions — provides all users' submitted timetables; used to build the preview gallery.
- updateUserSubmission and updateTimetable — incremental events when a single user submits or when the synthesized timetable is updated; the client updates mini-previews and re-renders the merged grid respectively.
- userJoined, selectedTimetable, chatMessage — UI notifications for presence, selection announcements, and chat.

Outbound events emitted by the client include:

- userJoined — announces the client has joined a session (username + sessionId).
- updateSessionCell — when a user chooses a new slot for a session, the client sends the updated session object to the server so it can persist and broadcast it.
- submitTimetable and API POST to /api/saveTimetable — when a user submits their chosen timetable; the client both POSTs to the REST endpoint (persistence) and emits a socket message so the session can be notified immediately.
- selectedTimetable — the user notifies peers which auto-schedule index they selected as "their timetable."

This combination of REST + socket usage provides both reliable persistence (HTTP) and low-latency notifications (Socket.IO).

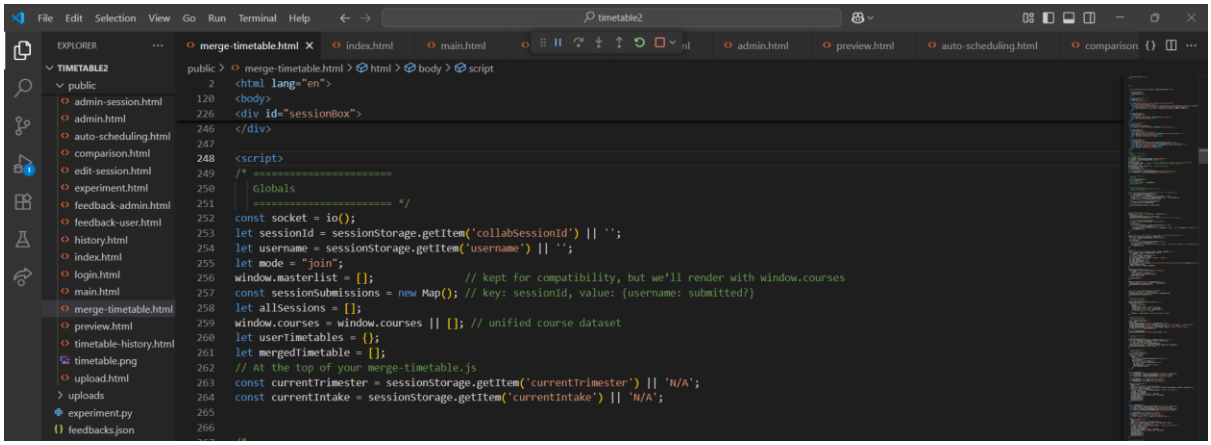


Figure 4.1.8.1 Architecture and Main Data Structures

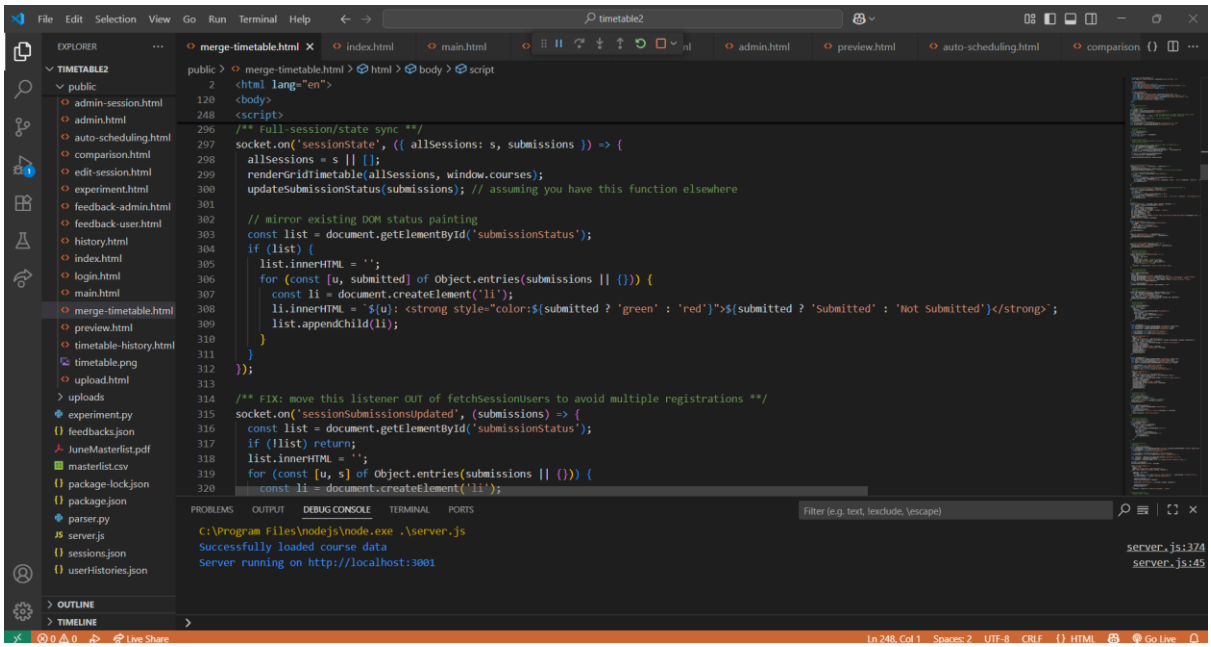


Figure 4.1.8.2 Architecture and Main Data Structures

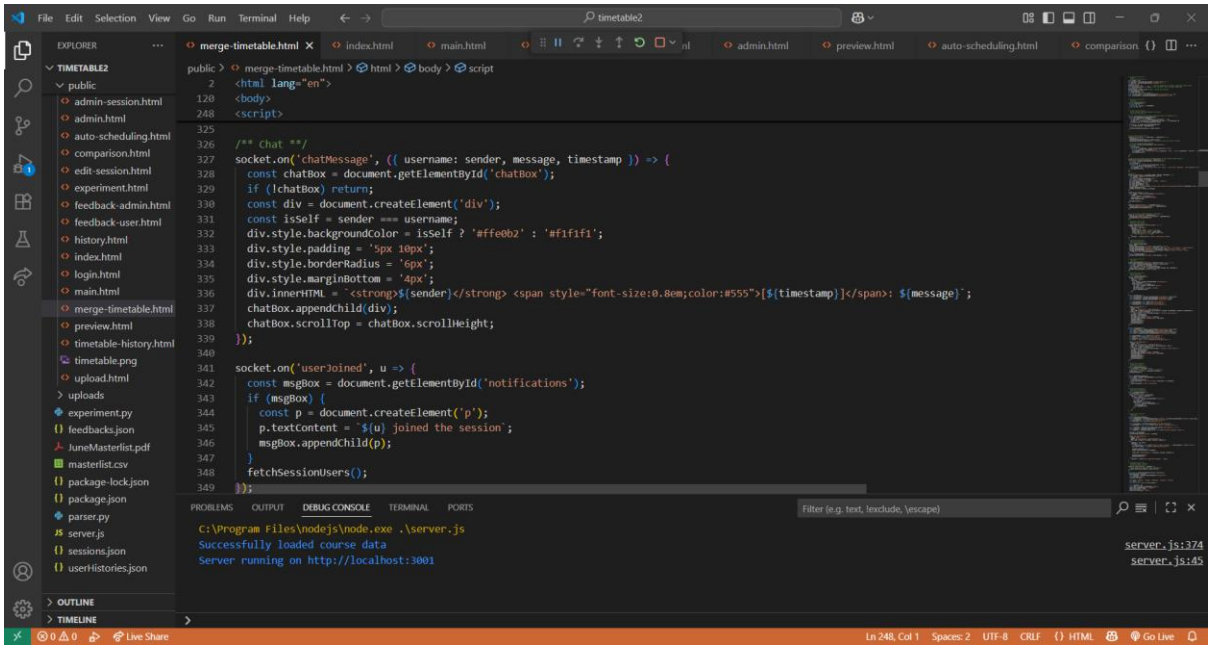


Figure 4.1.8.3 Architecture and Main Data Structures

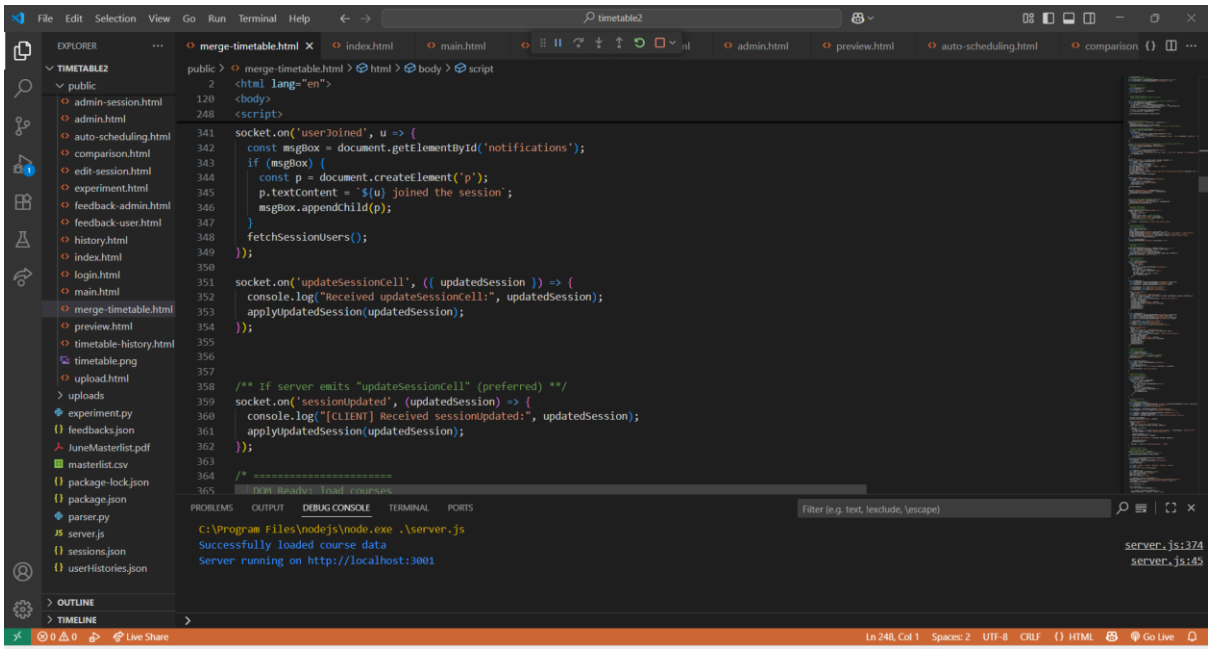


Figure 4.1.8.4 Architecture and Main Data Structures

renderGridTimetable(sessions, masterlist) builds a visual grid weekly calendar (Mon–Fri, 08:00–18:00). Sessions are placed into hour rows, and exact vertical positioning and sizing are computed using the helper timeToMinutes() so fractional starts (e.g., 08:30) and durations are reflected visually. Cells are marked occupied in cellOccupied to prevent overlapping DOM cells. Each session block is colored by status (merged, exclusive, conflict) to visually

communicate which items were agreed on, exclusive to a single user, or conflicted. Non-lecture sessions are interactive: clicking a tutorial/practical open highlights of *available slots* where that session can be moved.

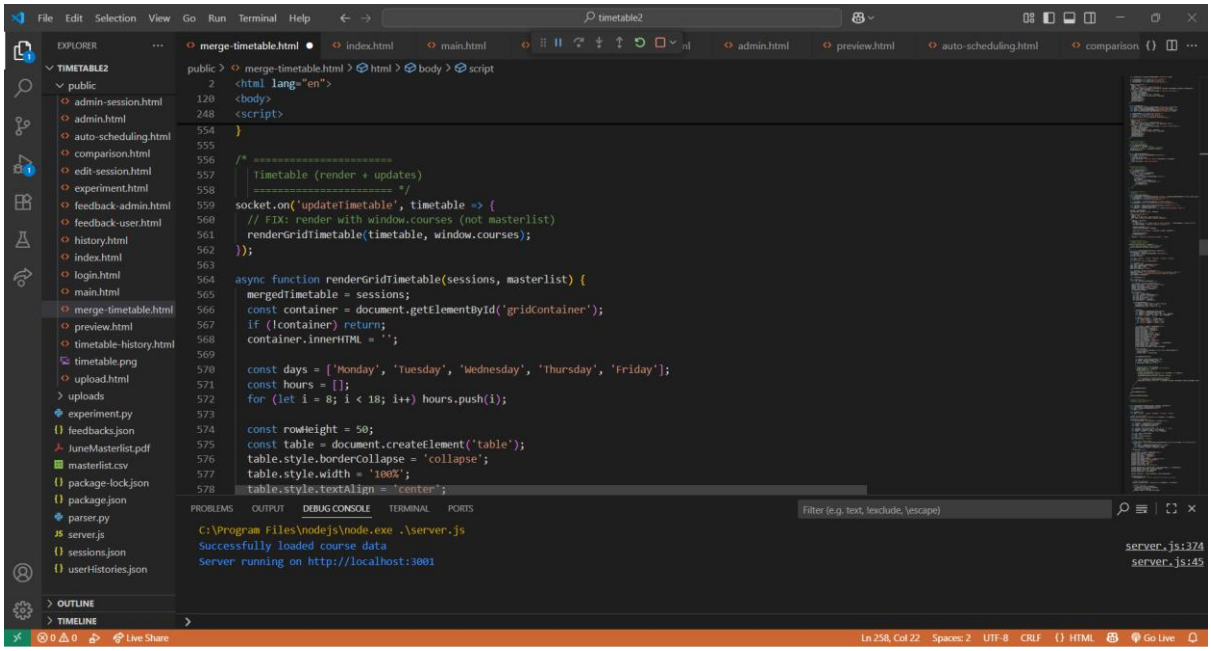


Figure 4.1.8.5 Timetable rendering & interaction model.

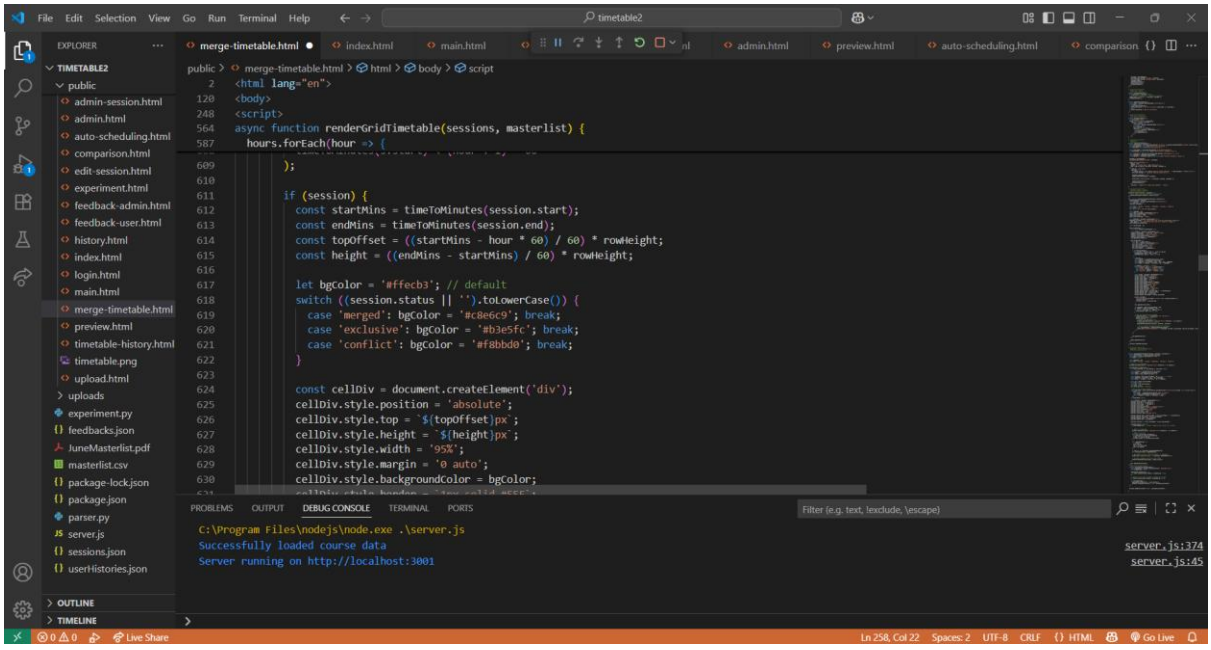


Figure 4.1.8.6 Timetable rendering & interaction model.

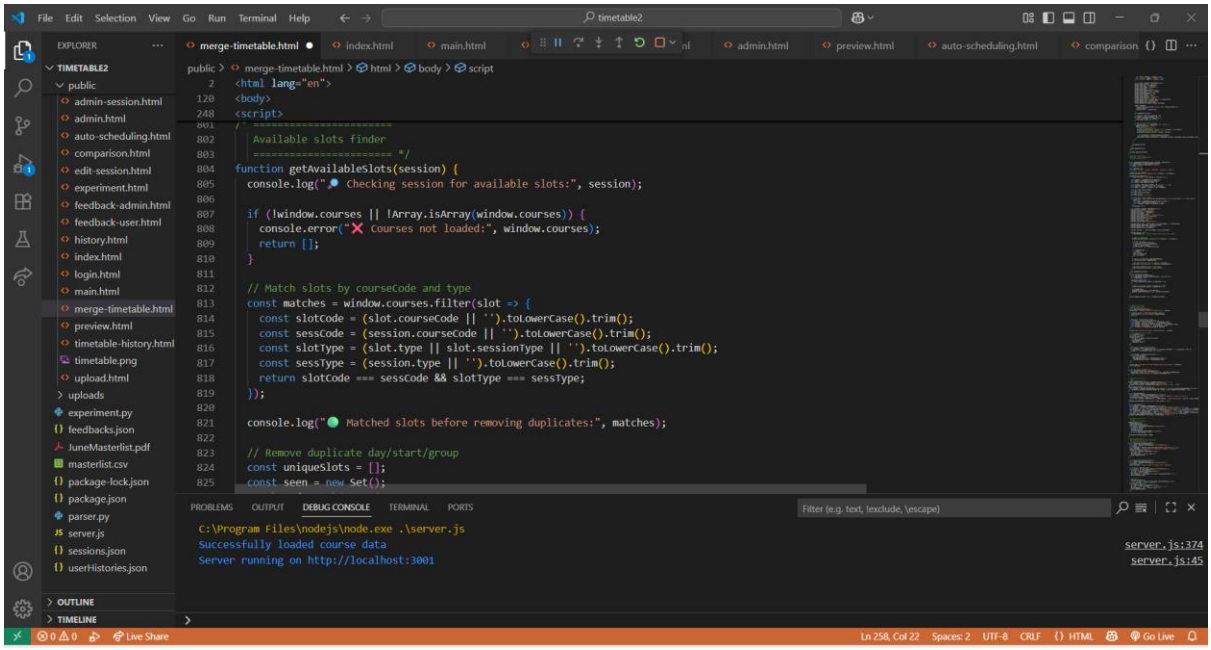


Figure 4.1.8.7 Timetable rendering & interaction model.

When a user clicks a non-lecture session, the script calls getAvailableSlots(session) to compute candidate slots and then highlightAvailableSlots(session, container, sessionsArr) to render clickable highlight overlays. getAvailableSlots searches window.courses for sessions matching courseCode and type, deduplicates by day-start-group, and filters out slots that clash with any compulsory lecture for that course — ensuring a tutorial/practical is never placed overlapping a lecture. The highlight overlays are positioned precisely in the timetable grid (using top offsets and heights derived from minute arithmetic) and have click handlers that: update the local sessionsArr with the newly chosen day/start/end; emit updateSessionCell to the server to save and broadcast the change; and re-render the timetable locally for immediate feedback. An outside-click listener removes highlight overlays when the user clicks away.

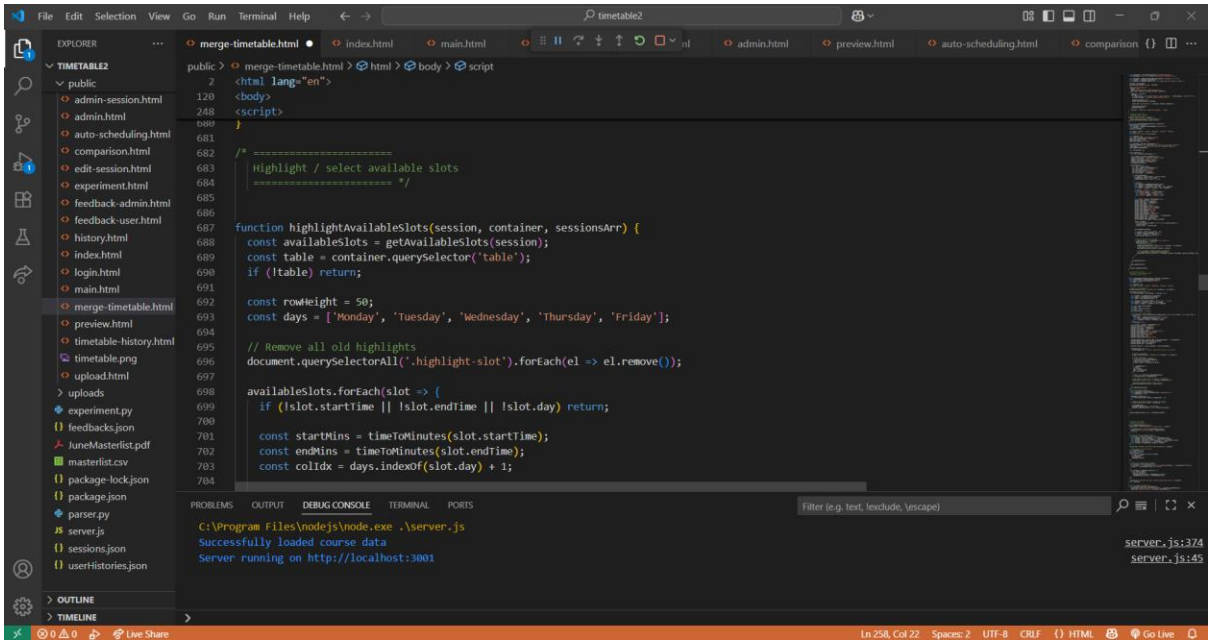


Figure 4.1.8.8 Finding and highlighting available slots

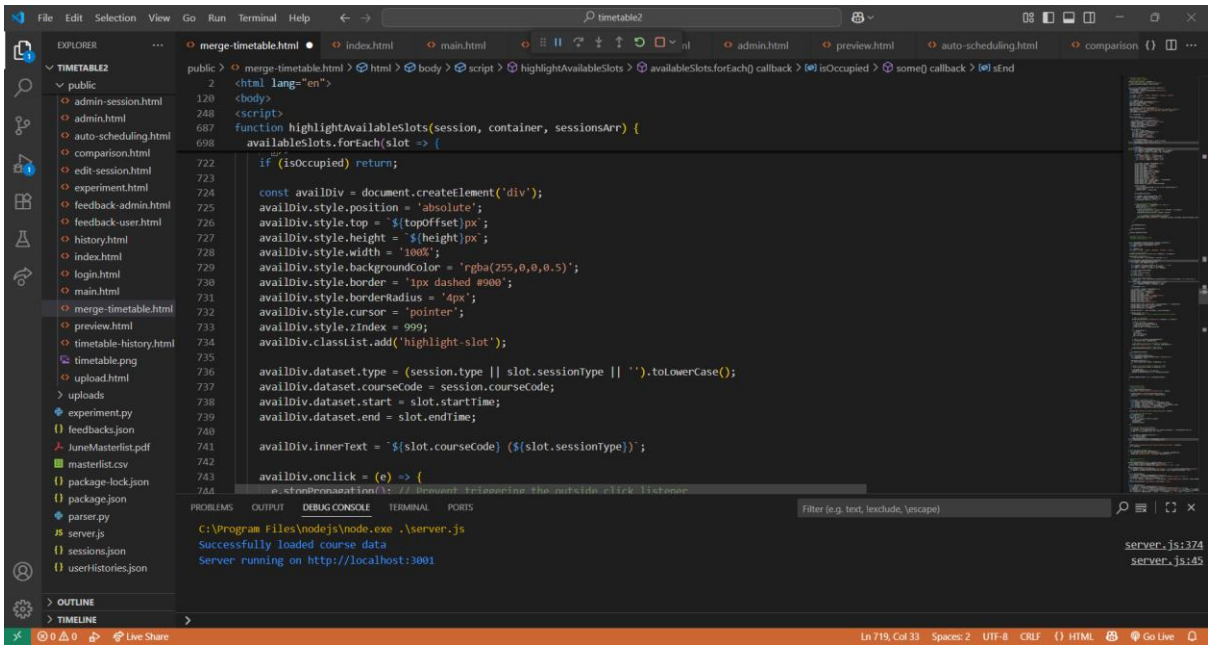


Figure 4.1.8.9 Finding and highlighting available slots

Users save/submit their chosen timetable by calling submitActualTimetable() which pulls a selected timetable from sessionStorage and POSTs it to /api/saveTimetable. After the server persists the timetable, it emits updateUserSubmission to all clients; the client adds a submitted flag and shows mini previews. The server keeps track of which users have submitted and (when all required submissions are present) synthesizes a merged timetable and broadcasts it with

updateTimetable (or via the /api/synthesizeTimetable endpoint). The client's socket.on('updateTimetable') handler receives the authoritative merged result and calls renderGridTimetable so all participants see the same merged calendar.

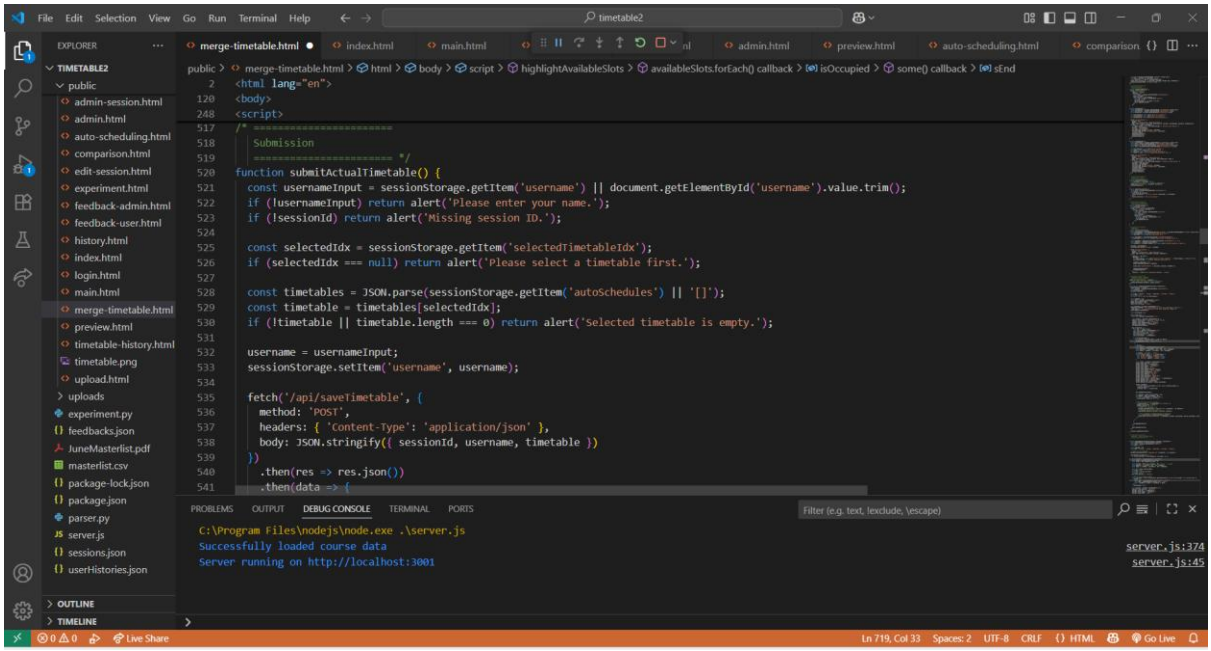


Figure 4.1.8.10 Submission and synthesis flow.

The script populates a user timetable selector (populateUserTimetableSelect) from sessionStorage (auto schedules) so a user can pick which auto-schedule they want to submit. The preview system (renderAllUserPreviews, renderMiniTimetable, renderMiniPreview) displays compact visual summaries of each user's submitted timetable. The preview code handles both array-of-sessions and array-of-arrays shapes, and it keeps the preview UI in sync when allUserSubmissions or updateUserSubmission arrives.

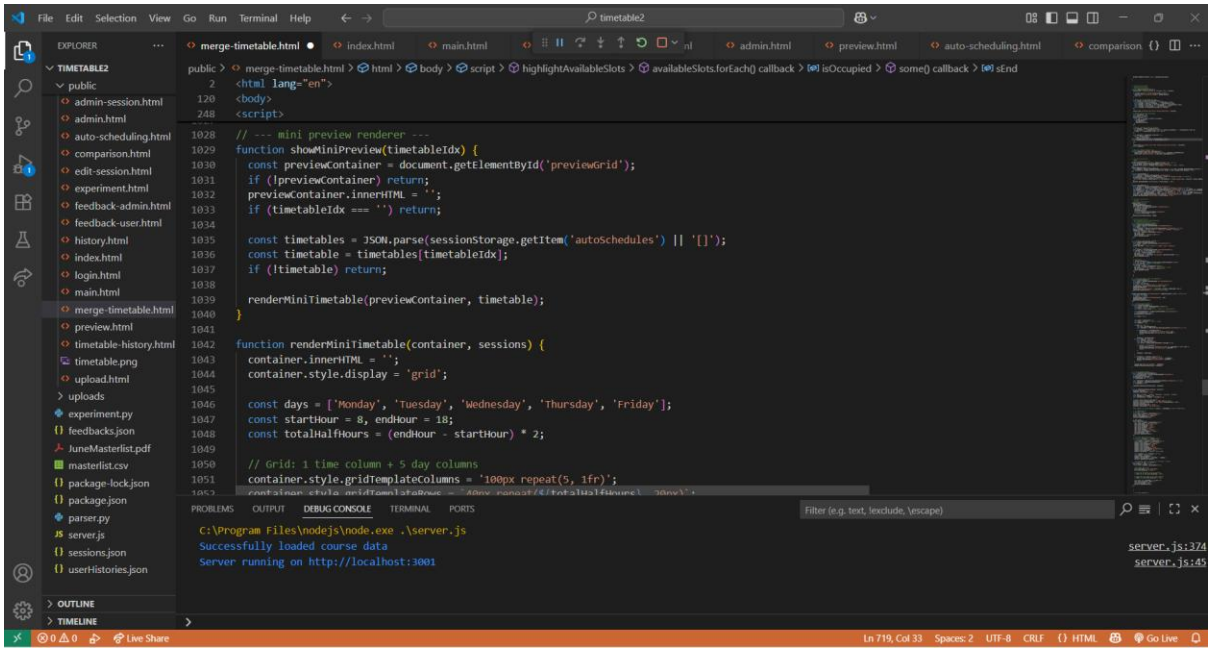


Figure 4.1.8.11 Previews, selection UI and mini rendering.

The merged timetable UI can be exported as PNG or CSV. For PNG exports and history saves the script uses html2canvas to capture the gridContainer into an image (high resolution on save flows), then either triggers a download (export) or POSTs the data URL to /api/user/history/save with metadata (username, label, mode: "merge", intake/trimester) to persist the snapshot in the user's history. Saving to history is protected by checks (session joined + non-empty grid) and returns user feedback on success/failure.

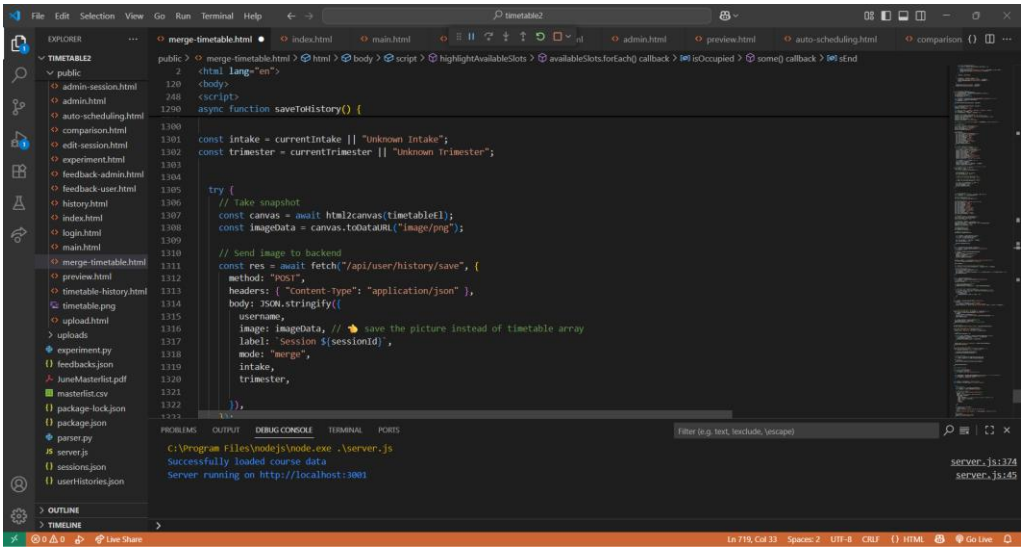


Figure 4.1.8.12 Save to History.

4.1.9 View Timetable History

The Timetable History module was developed to allow students to manage their previously saved timetables in a structured and interactive way. This functionality integrates both client-side rendering and server-side data management to provide features such as filtering, grouping, viewing, and deleting saved schedules. When the page loads, the loadHistory() function is executed, retrieving the logged-in username from session storage and sending a request to the backend API endpoint (/api/user/history/:username). The response is parsed into JSON, and timetables are dynamically rendered inside the historyContainer. If no timetables exist, the system displays a user-friendly message to indicate that no records are available.

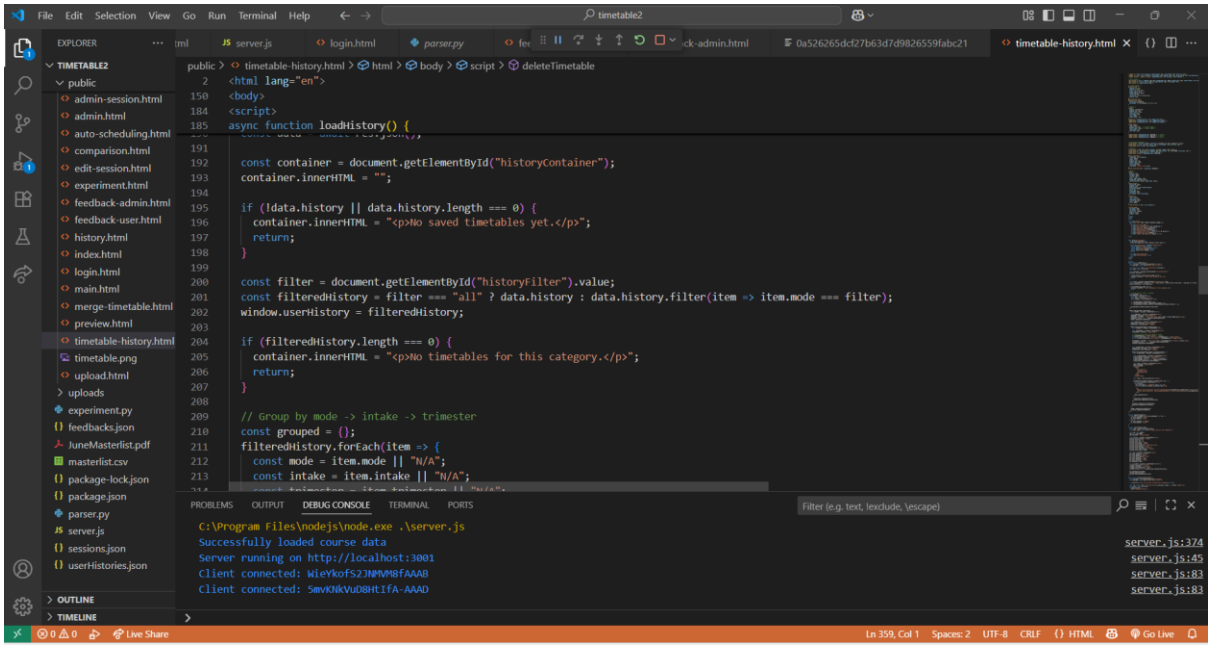


Figure 4.1.9.1 Load History function

A key part of the development involved grouping saved timetables by mode, intake, and trimester to make browsing intuitive. This grouping ensures that students can easily distinguish between timetables generated using manual or auto scheduling modes, across different intakes and trimesters. For example, the script dynamically creates expandable sections using collapsible panels (toggleCollapse() function), enabling a hierarchical view. At the lowest level, each timetable is displayed inside a table containing metadata such as its label and save timestamp, along with action buttons for viewing and deletion.

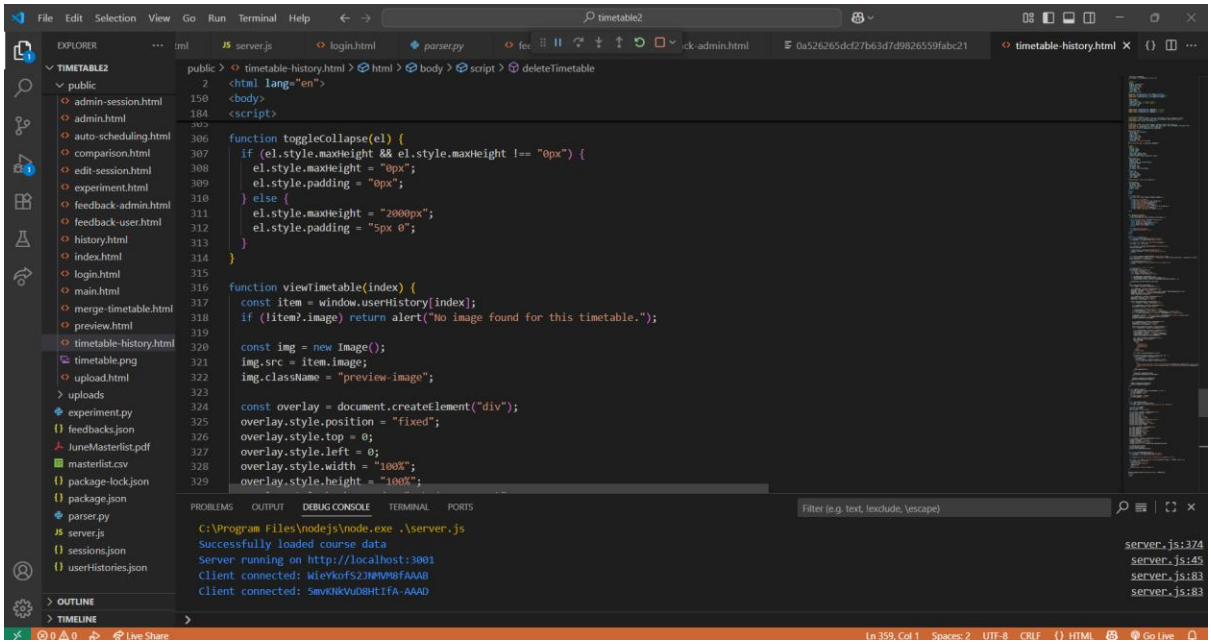


Figure 4.1.9.2 Toggle Collapse function

The view timetable feature was designed for quick preview without navigating away from the history page. This is achieved through the viewTimetable() function, which displays the saved timetable image inside a modal overlay. The modal is styled with semi-transparent background shading and a centered preview box, providing a focused display of the timetable. The image is retrieved directly from the saved data (item.image) and scaled appropriately to fit different screen sizes. A close button allows students to dismiss the preview, reinforcing usability across both desktop and mobile platforms.

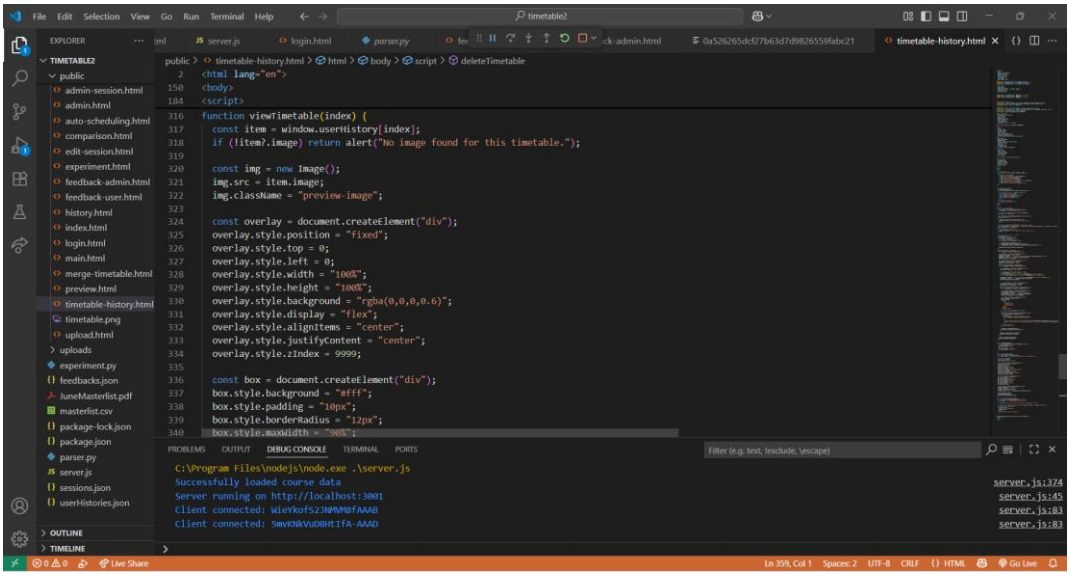


Figure 4.1.9.3 View Timetable function

Additionally, the delete timetable function (deleteTimetable()) enhances user control over saved data. When triggered, the system confirms the deletion action for safety before sending a DELETE request to /api/user/history/:username/:index. The backend processes the request, removes the timetable entry, and returns a JSON response. If successful, the history view is reloaded to reflect the updated state in real time. This development choice ensures that students always see an up-to-date view of their saved timetables without needing a manual page refresh.

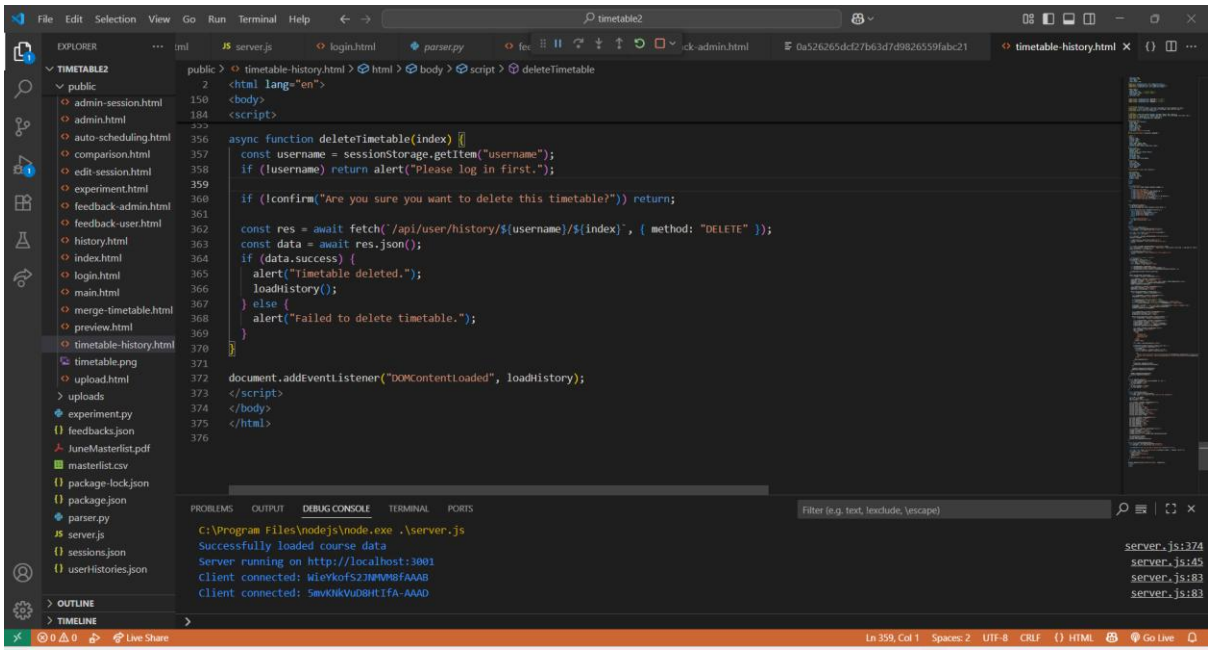


Figure 4.1.9.4 View Timetable function

4.1.10 User Feedback

The User Feedback module was developed to allow students to communicate directly with administrators by submitting feedback, reporting issues, or suggesting improvements for the Smart Student Timetable Planner. The frontend is built with HTML, CSS, and JavaScript, styled to maintain a consistent theme with the rest of the system. The interface contains two key sections: a feedback submission form and a feedback history section where users can view administrator replies. The submission form includes fields for username, feedback type (general, bug report, or suggestion), and a text area for the message. Once submitted, the feedback is sent to the backend using a POST request (/api/feedback). To enhance user experience, the system displays a floating “Thanks for your feedback!” box using CSS animations, confirming successful submission without requiring a page refresh.

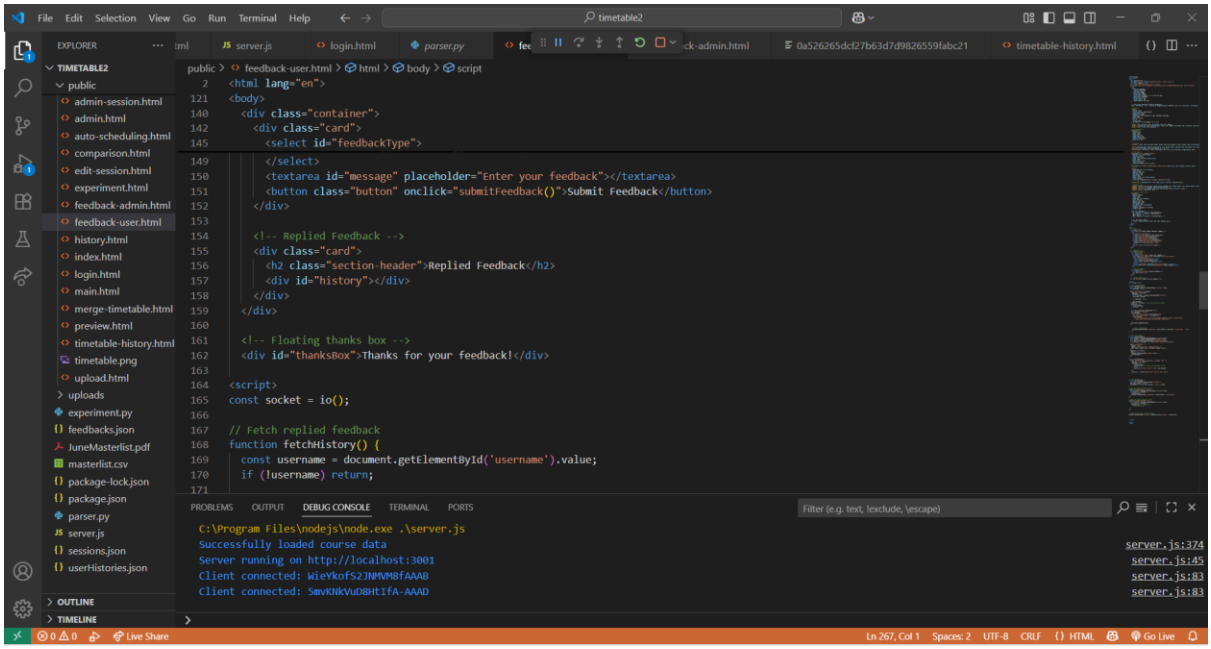


Figure 4.1.10.1 Alert Float Box

To provide users with continuous engagement, the module also retrieves previously replied feedback through a GET request (/api/feedback/:username). The replies are dynamically rendered within the history section, showing both the user's original message and the administrator's response. A red notification dot appears on the navigation bar whenever unread replies exist, ensuring users are aware of new updates. Each feedback entry includes an option to mark replies as read, which sends a request (/api/user/feedback/:id/read) to the server and updates both the history view and the notification indicator. This interaction enhances usability by helping students manage and track their communication history with administrators.

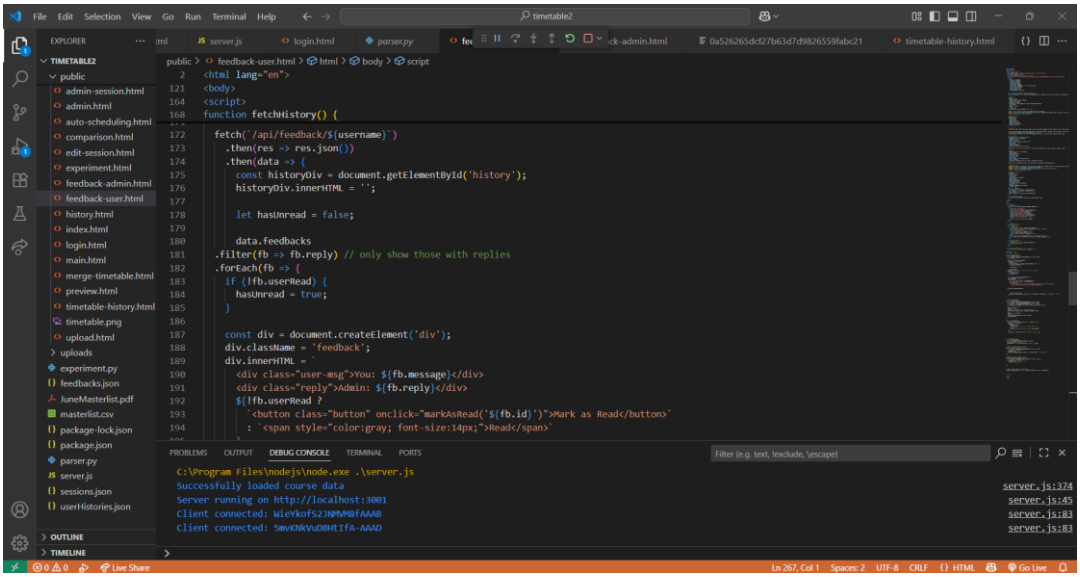


Figure 4.1.10.2 Get Request

The module also integrates real-time updates using Socket.IO, enabling immediate synchronization when administrators reply to feedback. This eliminates the need for users to manually refresh the page. When an admin posts a reply, the feedbackReplied socket event automatically triggers an update of the user's feedback history and reactivates the red notification dot. Similarly, when a user marks a feedback reply as read, a feedbackUserRead event ensures the UI is updated across connected clients. This design ensures responsive, interactive, and collaborative communication between students and administrators, strengthening system reliability and transparency.

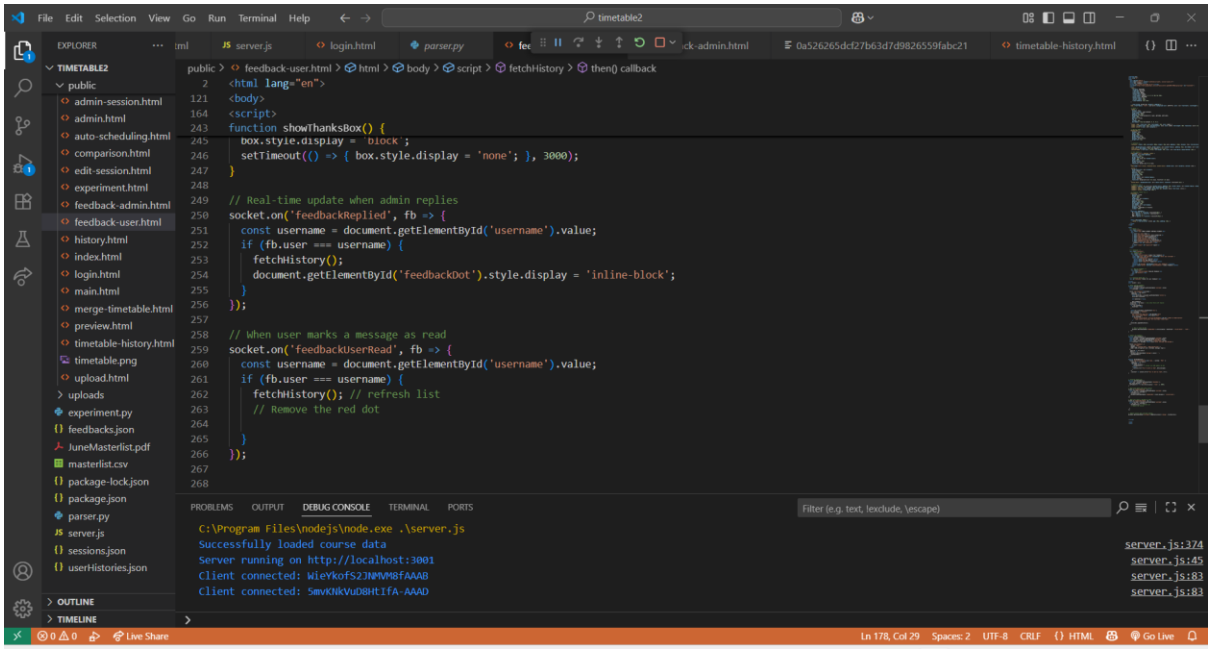


Figure 4.1.10.3 Real-Time Updates

4.1.11 Upload Course

The Upload Course module was developed to streamline the process of importing course data into the Smart Student Timetable Planner. Instead of requiring manual entry of courses, administrators can upload an official course timetable in PDF format. The frontend was implemented using HTML, CSS, and JavaScript to maintain a consistent design with the rest of the platform. The interface provides a custom file upload selector styled with a drag-and-drop appearance, alongside a dynamic progress bar that tracks each stage of the upload process. Once a file is selected, the chosen filename is displayed, and upon submission, the file is sent to the server using an XMLHttpRequest POST request to the /upload route.

To enhance user experience, the module includes visual feedback mechanisms that guide the administrator through the entire process. The progress bar is updated in real time, showing the percentage of the file uploaded. Once the upload is complete, the status automatically switches to Processing, and upon successful parsing by the backend, it changes to Upload & Parsing Complete! with a green highlight. In the event of errors—such as invalid files or connection failures—the system provides clear alerts and a red failure status, ensuring administrators understand the problem and can retry immediately.

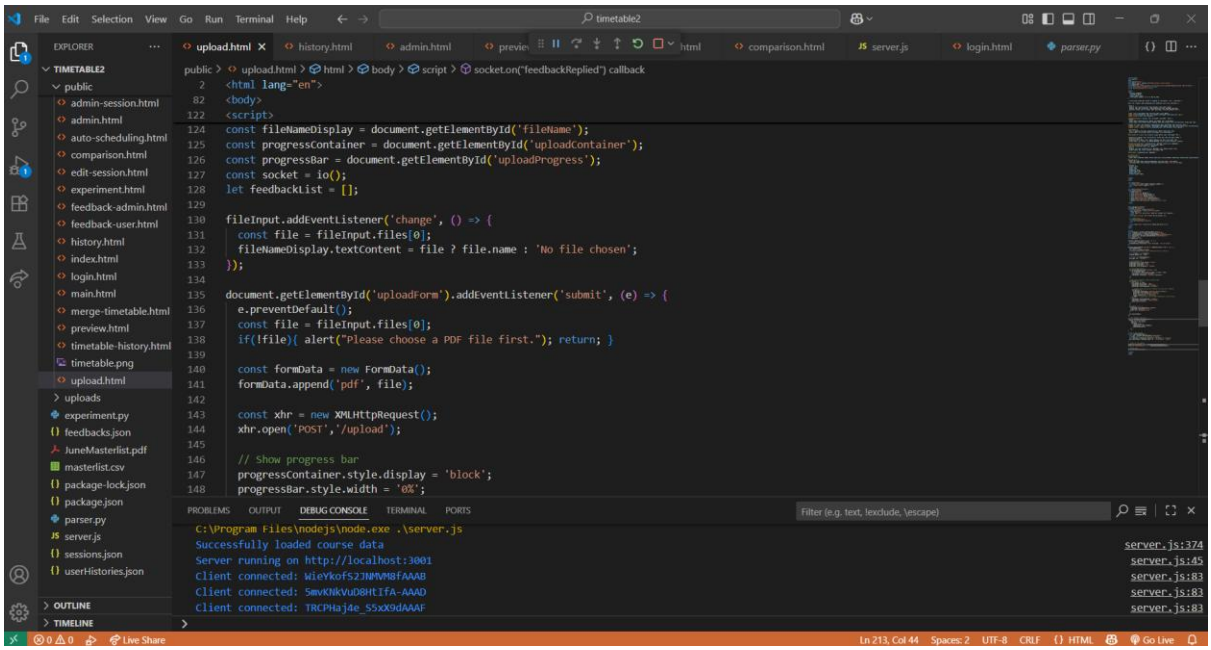


Figure 4.1.11.1 upload.html

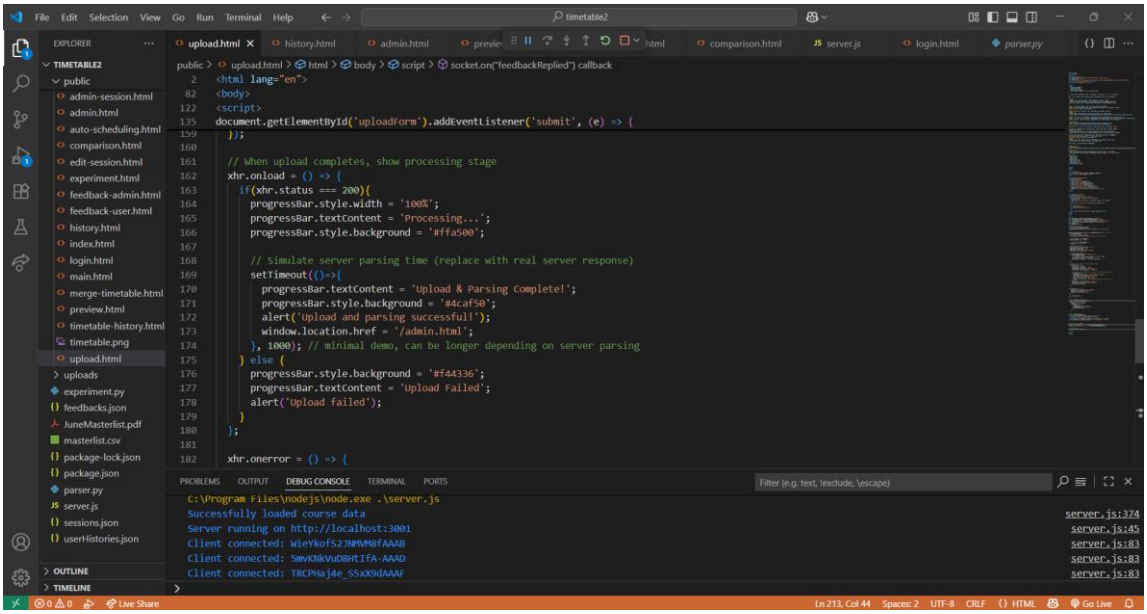


Figure 4.1.11.2 upload.html

4.1.12 Upload Course History

The Admin Upload History module was designed to provide administrators with a clear and organized record of all course files that have been uploaded into the Smart Student Timetable Planner. This feature ensures transparency and accountability by allowing admins to track the details of every upload, including the file name and the exact time it was submitted. The interface is structured with HTML and styled using CSS to align with the system's overall dashboard layout, ensuring consistency across all administrative modules. A responsive design approach was used so that the history table adapts seamlessly to various screen sizes, enabling administrators to review records on both desktop and mobile devices.

The upload history is dynamically populated by fetching data from the backend through a GET request to the /api/history endpoint. Once the data is retrieved, JavaScript generates table rows containing each file's name and upload timestamp, which are displayed in a user-friendly format using toLocaleString() for readability. This automated retrieval removes the need for manual record-keeping, ensuring that every upload event is captured and displayed in real time. By storing and presenting this information, the system ensures that administrators can easily verify past uploads, identify errors, and confirm that course data has been successfully parsed into the system.

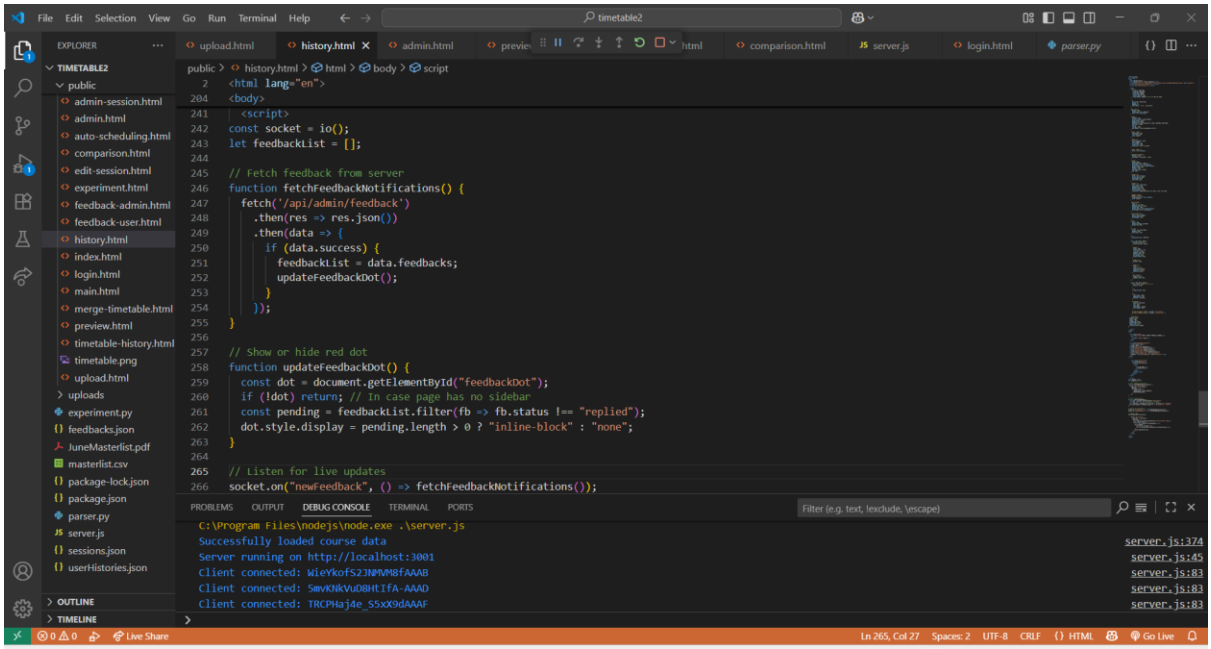


Figure 4.1.12.1 history.html

4.1.13 Preview Courses

The script begins by fetching course data from the backend using the /api/courses endpoint, and it also retrieves the latest upload information through /api/history. Once the data is successfully fetched, the script dynamically populates the table body with course records, ensuring only a subset of rows (30 per page) is displayed at a time. Pagination is managed through renderTable() and updatePagination(), which allow administrators to navigate between pages efficiently without overloading the interface with large amounts of data. This ensures smooth performance and readability when previewing potentially extensive course datasets.

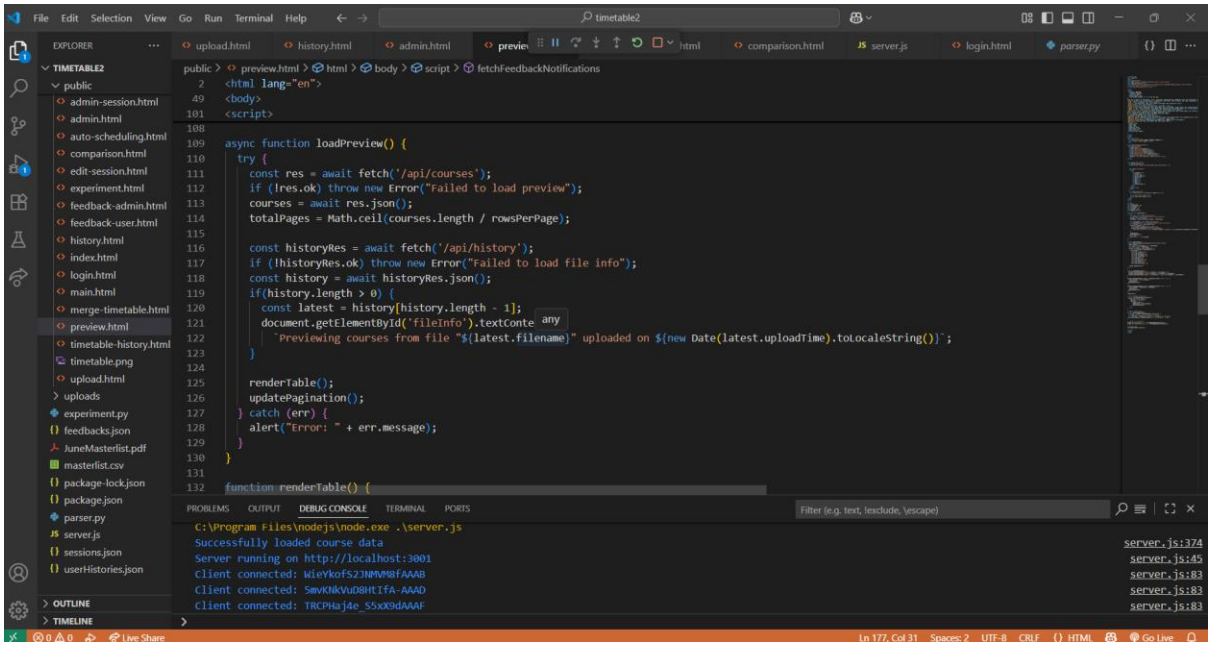


Figure 4.1.13.1 preview.html

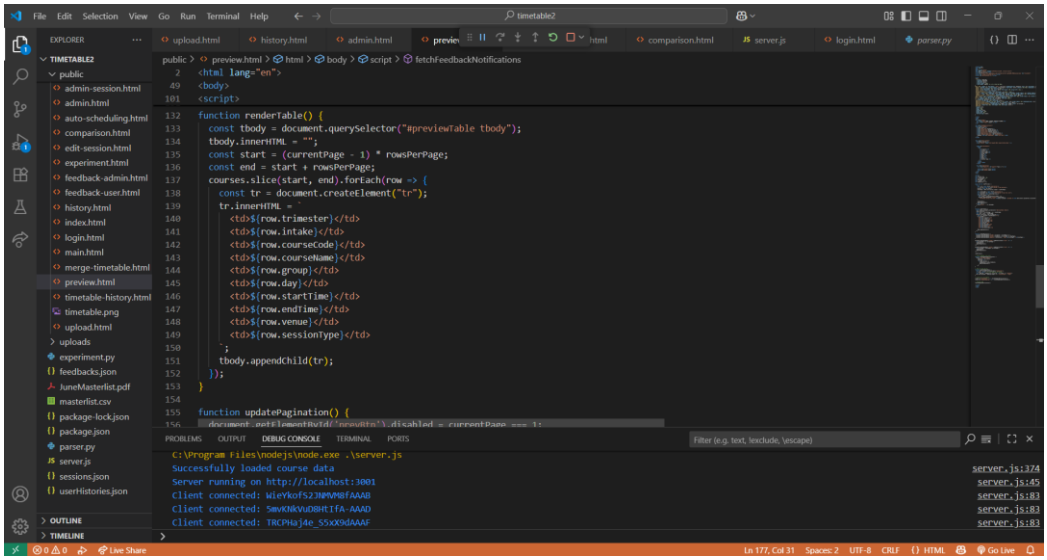


Figure 4.1.13.2 preview.html

4.1.14 View Created Sessions

The Created Sessions module was developed to provide administrators with an overview of all collaborative sessions created within the Smart Student Timetable Planner. The interface is structured with a navigation bar, a sidebar menu, and a main content area. The sidebar organizes key admin functions such as course uploads, history tracking, feedback, and session management, ensuring a consistent and intuitive navigation experience. The main content area displays a dynamically populated table that lists session details, including the session name, unique ID, creation time, and participant list with count. The table is generated through a JavaScript function (loadSessions()), which retrieves session data via the /api/sessions endpoint. This ensures that the administrator always has access to the most up-to-date information stored on the server.

In addition to static retrieval, the script integrates real-time updates using Socket.IO, allowing the session table to refresh dynamically without requiring a page reload. When a new session is created, the "sessionCreated" event immediately inserts the session record into the table, while the "participantJoined" event updates the participant count and list when new users join a session. This real-time synchronization ensures administrators maintain an accurate view of ongoing collaborations. Moreover, the module incorporates a feedback notification system, where the script fetches pending feedback from /api/admin/feedback and toggles a red dot indicator to alert administrators of unread or unreplied messages. The notification state is further kept in sync with "newFeedback" and "feedbackReplied" socket events, ensuring administrators can respond promptly to user concerns. Together, these features enable seamless monitoring of system activity, providing both visibility and responsiveness within the administrative workflow.

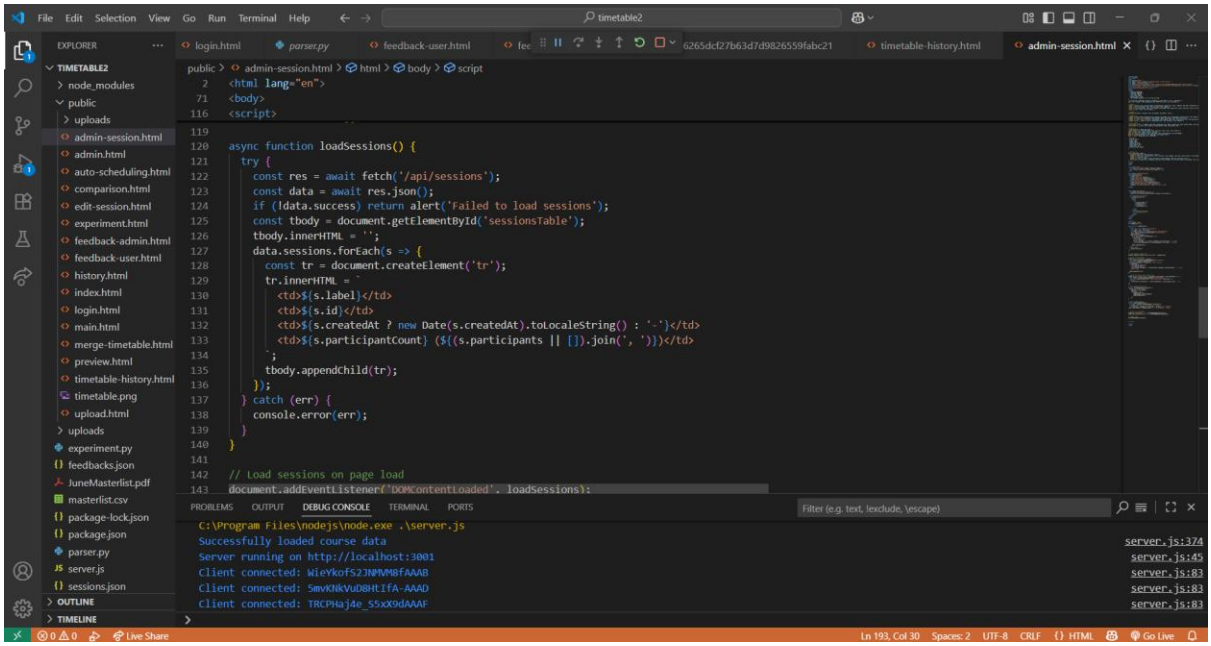


Figure 4.1.14.1 admin-session.html

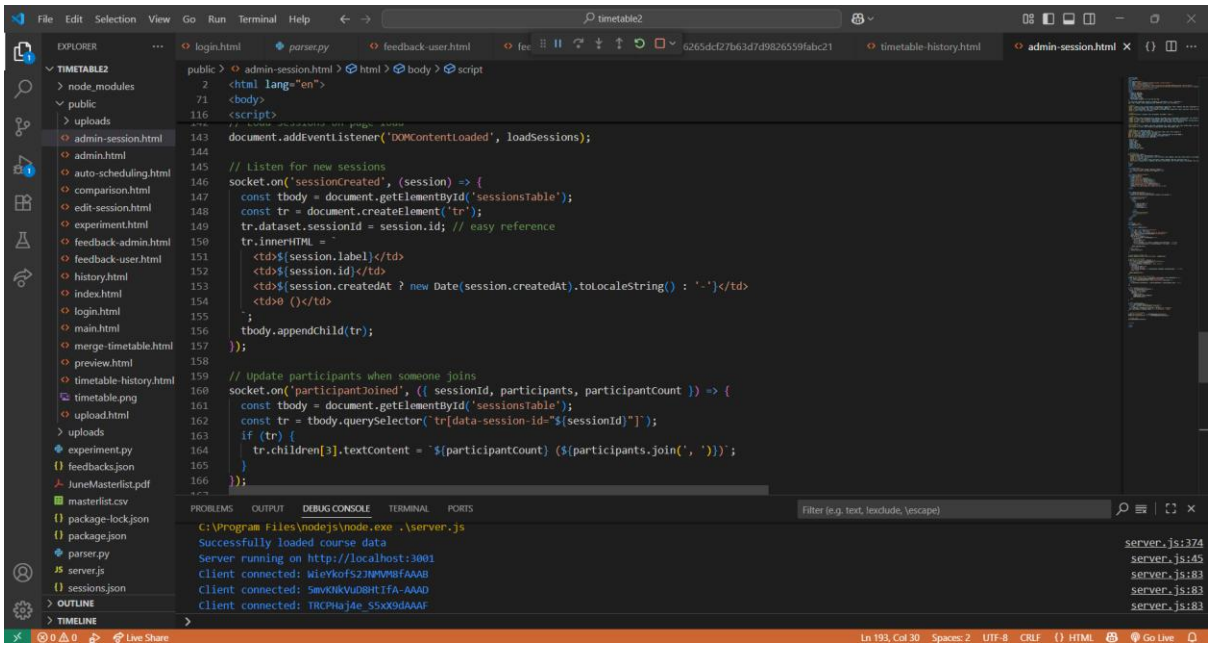


Figure 4.1.14.2 admin-session.html

4.1.15 Admin Feedback

The Feedback module was developed to enable administrators to efficiently review, respond, and organize user feedback within the Smart Student Timetable Planner. The layout consists of a sidebar navigation for accessing other admin functions, a filter bar for narrowing feedback by type or status, and a structured feedback table grouped into Pending, Read, and Replied sections. The table is dynamically populated by fetching records from the server through the

/api/admin/feedback endpoint. Feedback entries include details such as the user, feedback type, message content, status, and available actions. Pending feedback provides administrators with Reply and Mark as Read options, while read and replied entries are displayed for reference. A badge indicator is also included in the sidebar, showing the number of pending items so administrators always remain aware of unresolved feedback.

It also enhances interactivity by integrating modal-based reply handling and real-time updates via Socket.IO. When administrator clicks “Reply,” a modal window is triggered, allowing the reply to be composed and submitted through POST request (/api/admin/feedback/:id/reply). Replies are then immediately reflected in the table and broadcast to connected clients using the "feedbackReplied" socket event. Similarly, marking feedback as read triggers POST request (/api/admin/feedback/:id/read) and synchronizes updates across clients with a "feedbackRead" event. Socket listeners ("newFeedback", "feedbackReplied", "feedbackRead") ensure that changes made by one administrator are instantly reflected on all connected sessions, maintaining a consistent and collaborative view. Filters further allow administrators to refine results based on feedback type or status, improving searchability in large datasets. This modular approach to design ensures scalability, responsiveness, and clear communication flow between users and administrators, significantly strengthening the system’s support and feedback loop.

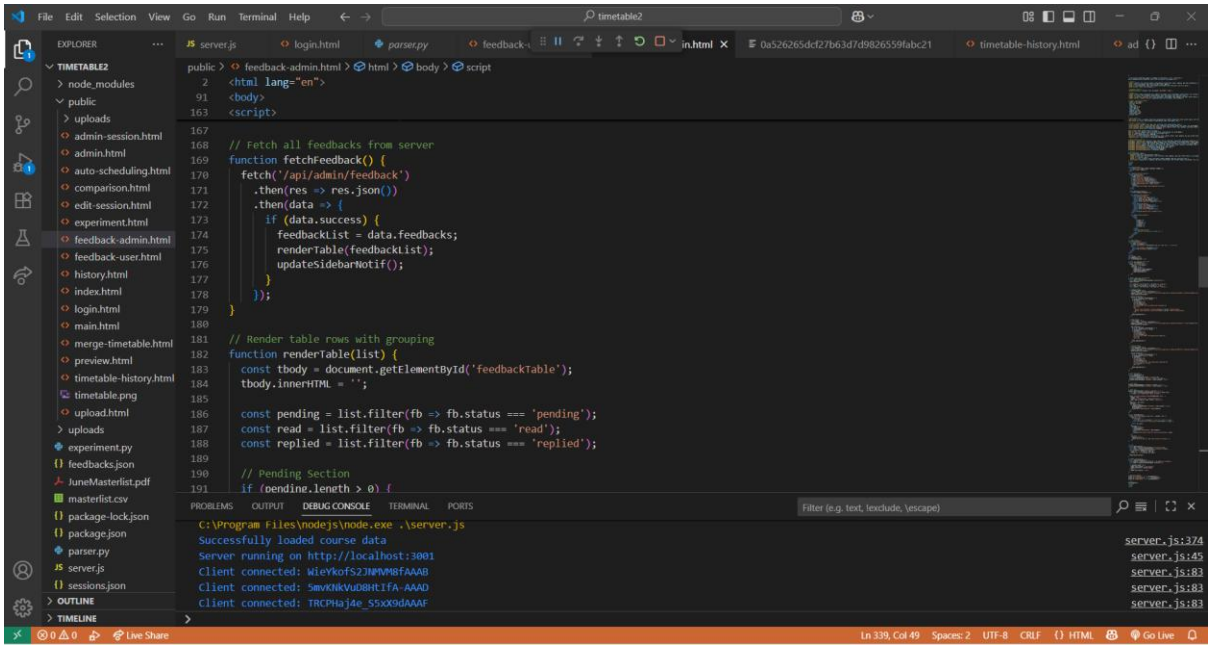


Figure 4.1.15.1 feedback-admin.html

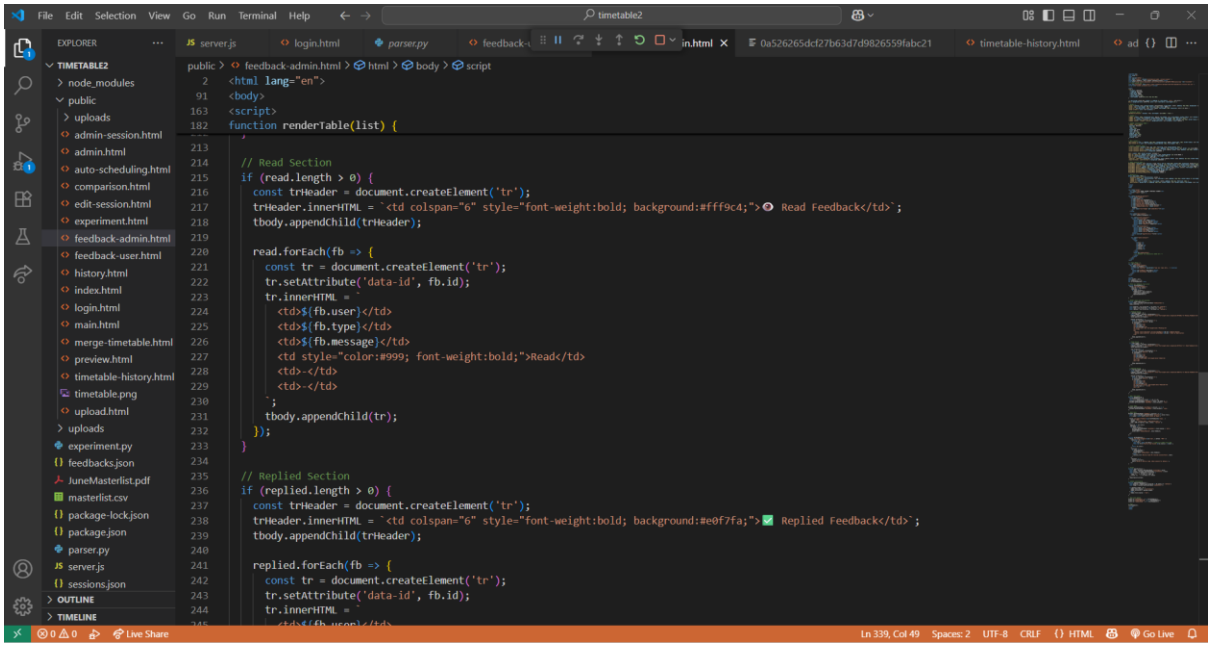


Figure 4.1.15.2 feedback-admin.html

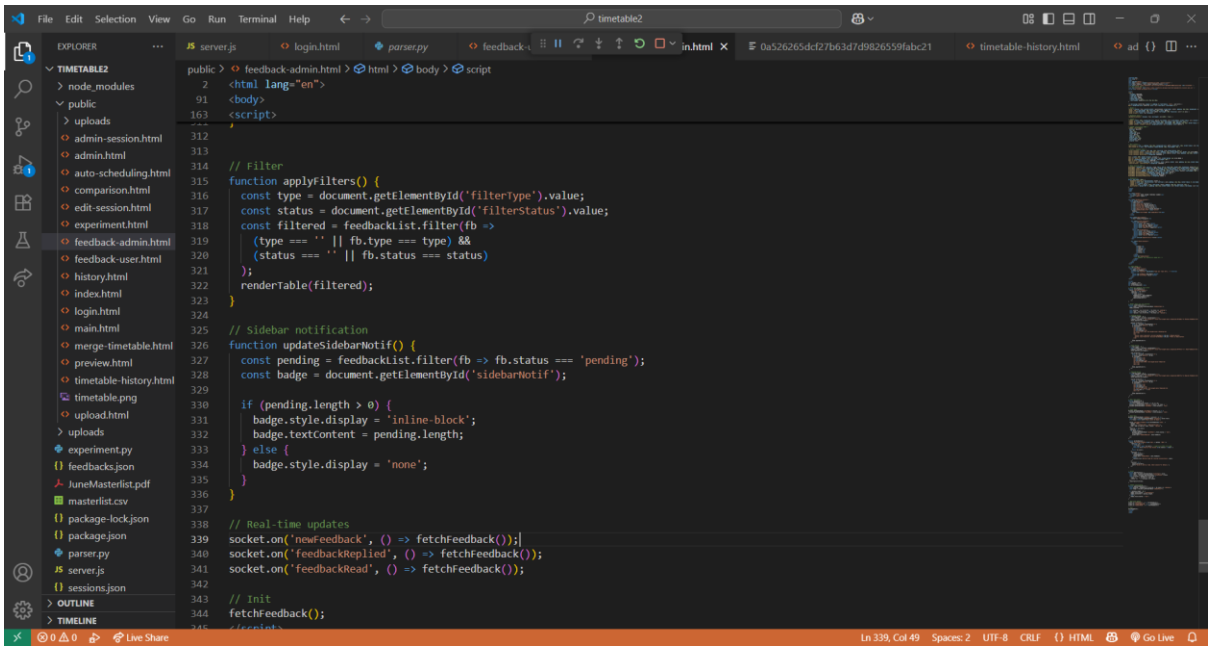


Figure 4.1.15.3 feedback-admin.html

4.2 Summary

The overall program development of the Smart Student Timetable Planner integrates multiple modules—login, scheduling, feedback, and collaboration—designed to provide a seamless and interactive user experience. The front end was implemented using HTML, CSS, and JavaScript for intuitive interfaces, while the backend relied on Node.js, Express, and Socket.IO to manage

real-time updates, data storage, and communication between clients and the server. Key modules such as the Genetic Algorithm-based auto-scheduling ensure conflict-free timetable generation, the collaborative merging feature supports real-time session sharing, and the feedback system enables structured communication between students and administrators. Together, these developments emphasize usability, reliability, and responsiveness, resulting in a robust system that simplifies course planning and enhances student–admin engagement.

Chapter 5

System Implementation

In this chapter, it covers the hardware and software setup, user interfaces, and challenges faced during development. The system runs in a client-server environment using Node,js, Express, and Socket.IO for real-time processing, with data handled through CSV and JSON integration. User interfaces were designed to be clear and responsive, featuring student modules for scheduling, merging and feedback, and an admin dashboard for course and feedback management. Key challenges included developing the GA in Auto Scheduling and ensuring real-time synchronization in timetable merging, preventing conflicts, and managing highlighted modifiable slots.

5.1 Hardware Setup

Table 5.1.1 Hardware Components and Requirements

Description	Specifications
Model	HP Pavilion Laptop 15-eg2xxx
Processor	12th Gen Intel ® Core™ i7-1255U
Graphic	NVIDIA GeForce MX550
Memory	8.00 GB (7.68 GB usable)

5.2 Software Setup

Table 5.2.1 Software Components and Requirements

Description	Specifications
Source Code Editor	Visual Studio Code, Google Colab
Programming Language	Python, Java, Node.js, HTML, CSS, JavaScript, JSON,Express, Socket.IO, papaparse
Database	MySQL
Operating System	Windows 11 Home Single Language
Documentation	Microsoft Office

5.3 User Interface

5.3.1 Login Page

When a user accesses the URL of the website, it will redirect them to this page with a modern login interface for the Smart Student Timetable Planner. The user interface of this login page is designed to be simple, clean, and user-friendly, with a focus on clarity and ease of navigation. At the top, a sticky navbar with the system's name, "SMART STUDENT TIMETABLE PLANNER", provides consistent branding and remains visible as the user scrolls. The login form itself is presented inside a centered card with rounded corners and a subtle shadow, giving it a modern and professional look. Users can select their role, either Student or Admin, through clearly styled toggle buttons at the top of the card, with the active role highlighted in orange for quick recognition. Below, the login form features neatly spaced input fields for username and password, styled with rounded edges in a bright orange that matches the theme, with hover effects for interactivity. Overall, the interface emphasizes accessibility, role clarity, and a visually appealing layout that guides users smoothly through the login process.

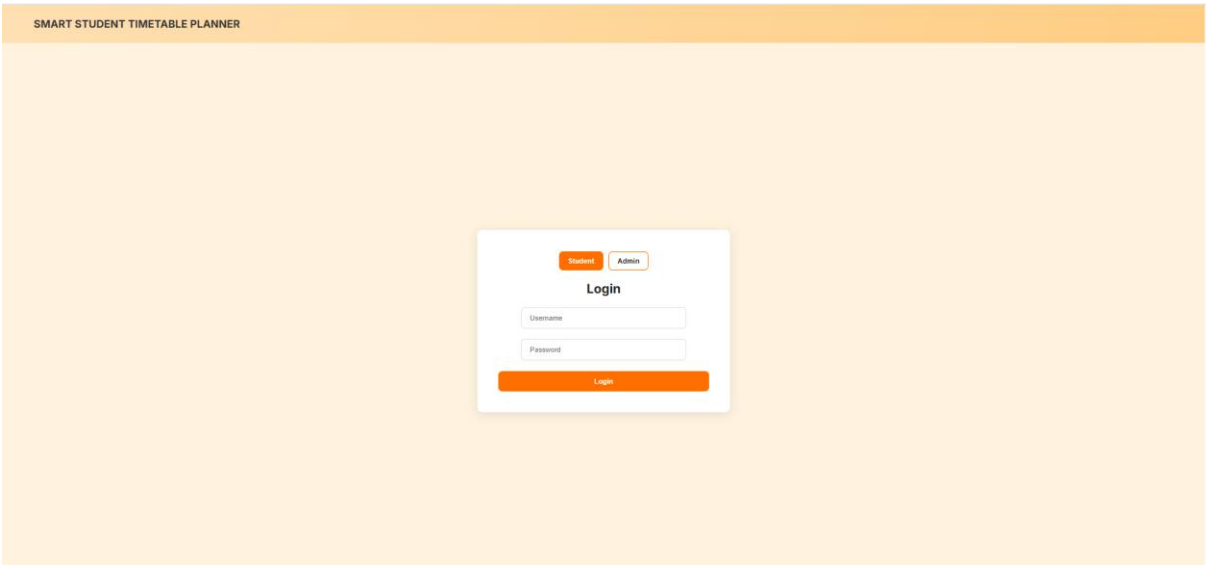


Figure 5.3.1.1 Login Page

5.3.2 Main Page

The user interface of this main page is designed to be modern, clean, and welcoming, providing users with quick access to the main features of the Smart Student Timetable Planner. At the top, a sticky navigation ban spans the full width of the page with the system's name on the left and neatly arranged links on the right, allowing users to move easily between pages such as Home, Auto Scheduling, Manual Scheduling, Comparison, Merging, History, Feedback, and

Logout. The hero section dominates the page with a bright gradient background and a split layout: on the left, bold headings and descriptive text introduce the platform, while two prominent orange buttons guide users directly to either Manual or Auto Scheduling. On the right, a large illustrative image of a timetable ensuring that on smaller screens the navbar and hero section stack vertically for readability and accessibility. Overall, the interface is intuitive, visually engaging, and structured to highlight the platform's core scheduling features.

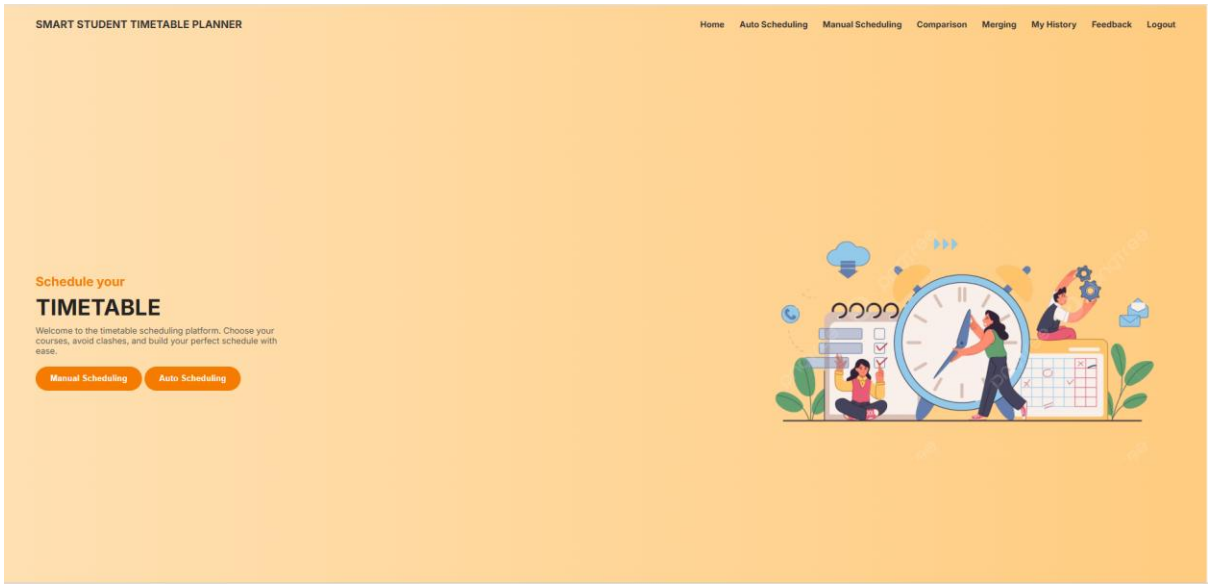


Figure 5.3.2.1 Main Page

5.3.3 Manual Scheduling Page

This page is designed to provide students with a structured and interactive way to register and organize their courses. At the top, a navigation bar spans across the screen, displaying the system's title on the left and quick links on the right to different sections such as Home, Auto Scheduling, Manual Scheduling, Comparison, Merging, History, Feedback, and Logout. This ensures consistency and allows users to easily switch between different features in the system.

The main content is arranged into card-based sections, each guiding the student through the registration process step by step. The first section allows students to select their intake month using a dropdown menu, ensuring that course availability is filtered based on the correct academic intake. Below that, another dropdown enables users to select their trimester, further narrowing down the list of courses to those relevant for that period. The layered filtering design minimizes confusion and keeps the selection process straightforward.

Once the trimester is chosen, the interface presents a course table listing available courses with their codes and names, along with an action button to select a course. After a course is selected, a session selection table appears, displaying details such as group, day, time, venue, and type of session. Each session includes a checkbox, allowing students to choose the required lectures and either one tutorial or practical session. A clearly styled "Add Course" button enables students to confirm their choices, which then get added to their personal schedule.
The selected sessions are displayed in the schedule table, which lists all registered courses with full details including trimester, course code, session type, group, day, time, and venue. Each entry includes a modify option, giving users flexibility to remove or adjust sessions if necessary. To provide a visual overview, a weekly timetable grid is displayed below, where all registered sessions are mapped according to their times and days. This dual representation, detailed table and visual grid, helps students cross-check for potential clashes and view their timetable in a familiar calendar-like format.

Finally, below the timetable, the interface includes a button group for exporting and saving schedules. The export button provides a dropdown menu to export the timetable as an image or Excel file, while a dedicated save button allows students to save their timetable to their history for future reference. These features are styled consistently with the overall theme, ensuring that students can not only build their schedules but also store and share them conveniently. Overall, the manual scheduling interface combines clarity, step-by-step guidance, and flexibility to give students full control over constructing their academic timetable.

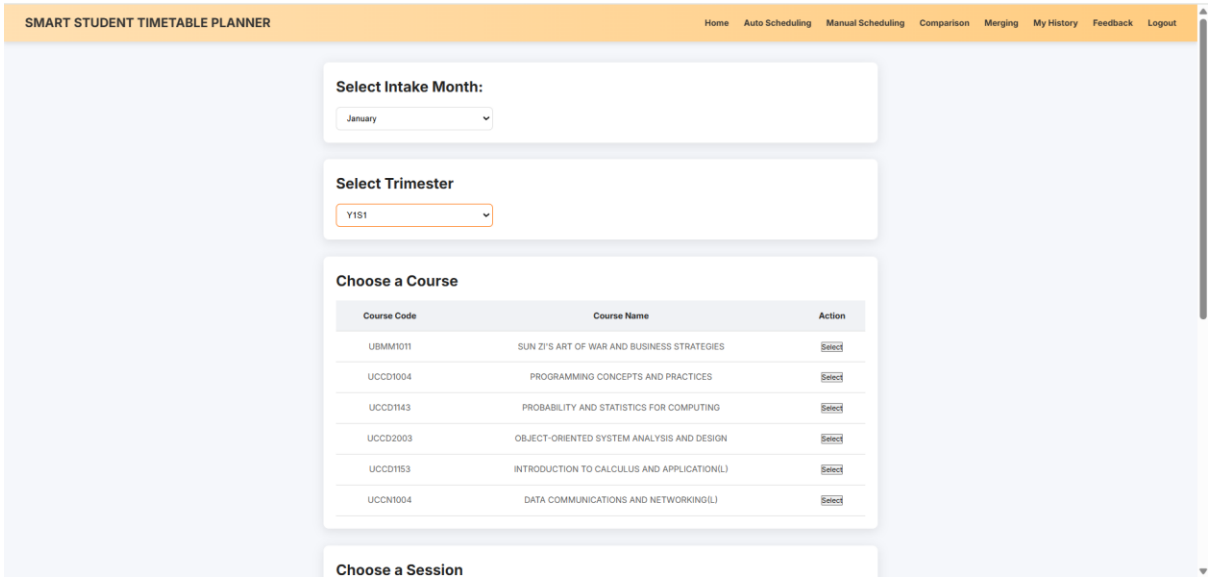


Figure 5.3.3.1 Select Intake, Trimester and Course List Available

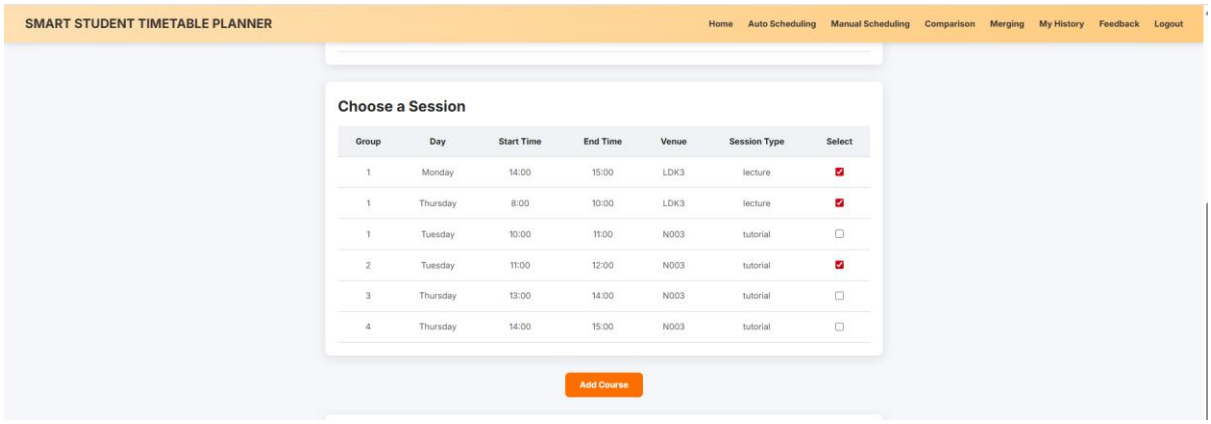


Figure 5.3.3.2 Session Available for the Selected Course

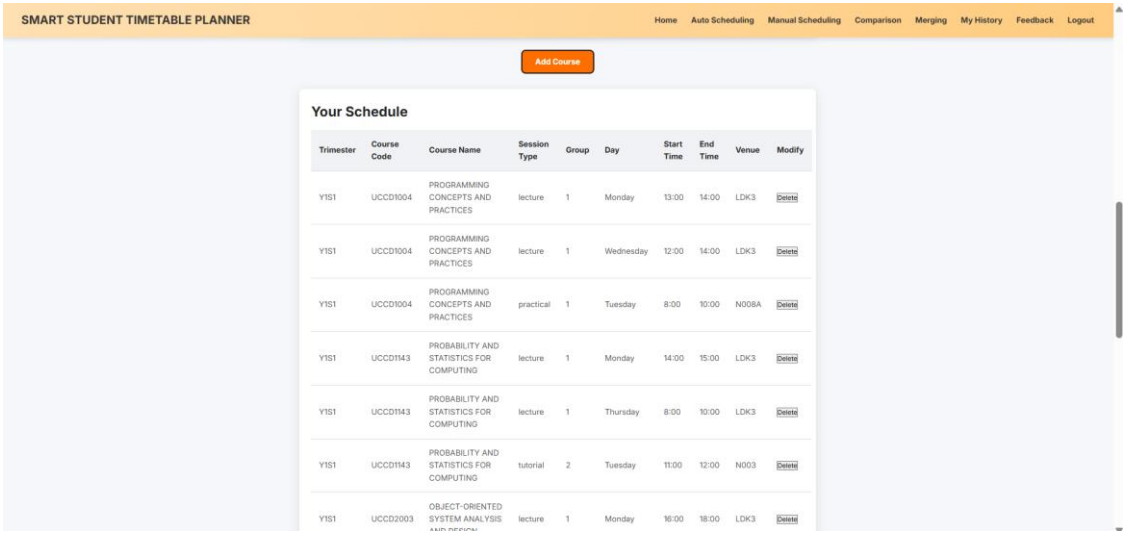


Figure 5.3.3.3 Schedule for Selected Course

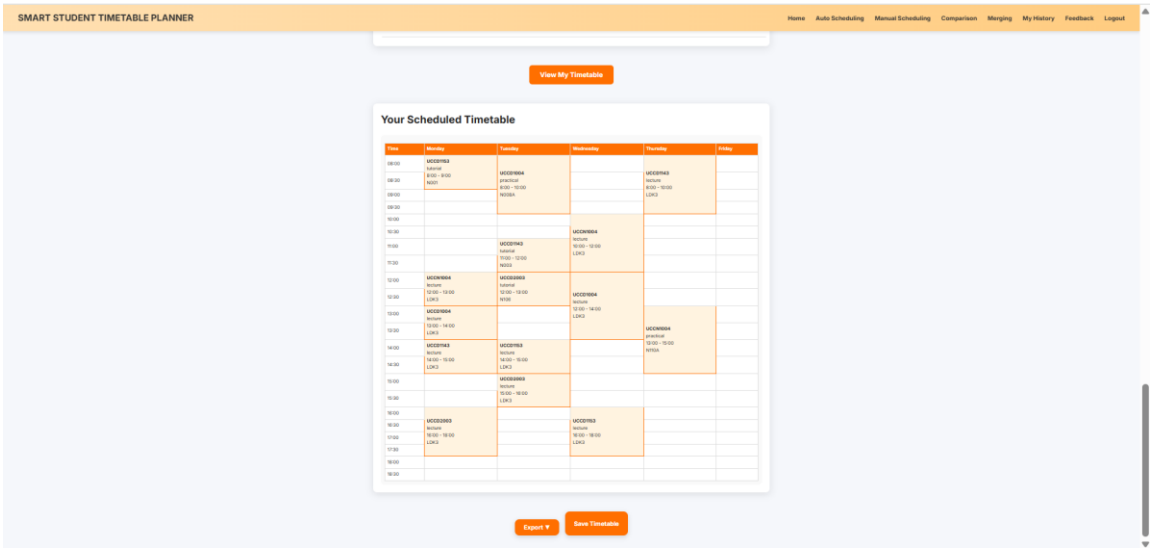


Figure 5.3.3.4 Timetable View, Export and Save Timetable Button

5.3.4 Auto Scheduling Page

The Auto Scheduling Page is designed to provide students with an automated and efficient way of generating valid timetables using a GA. The interface maintains a consistent navigation bar at the top, offering quick access to all main sections of the platform such as Home, Manual Scheduling, Comparison, Merging, History, Feedback, and Logout. This ensures users can easily navigate across different features without confusion. The main content is structured within card-based sections, each clearly labeled with headers to guide the user step by step through the scheduling process.

The interface begins with intake month selection and trimester selection, both provided in neatly styled dropdown menus. Once chosen, the system dynamically updates the course list to display only the relevant options. The course selection section presents a list of available courses for the chosen trimester, while the time constraints section allows users to specify unavailable slots, ensuring the generated timetable respects personal preferences. A prominent “Generate Schedule” button is provided to initiate the timetable generation process.

After schedules are generated, the results are displayed in the Generated Schedule section, which includes a detailed timetable table along with a set of functional controls. Users can choose from multiple generated schedules using a dropdown menu, and then either export the timetable in image or Excel format or save it directly to their history for later use. The export

button includes a dropdown design for clear and accessible output options, while the save button is highlighted in green to emphasize successful storage. Overall, the Auto Scheduling page provides a structured, interactive, and user-friendly interface that simplifies complex timetable generation into an accessible step-by-step workflow.

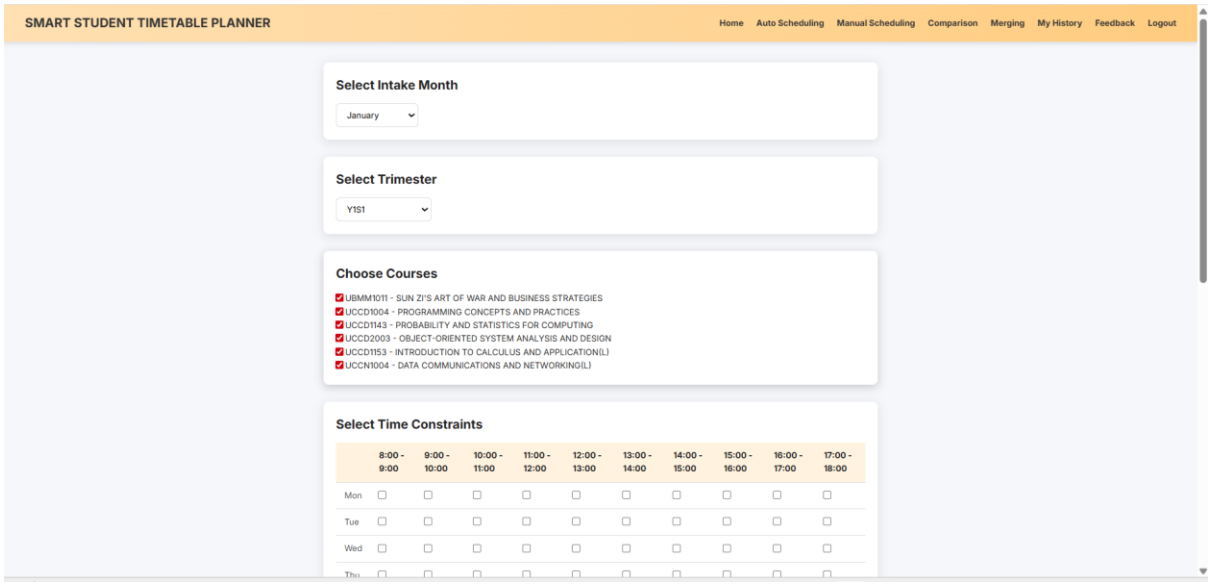


Figure 5.3.4.1 Select Intake, Trimester, and Course Available

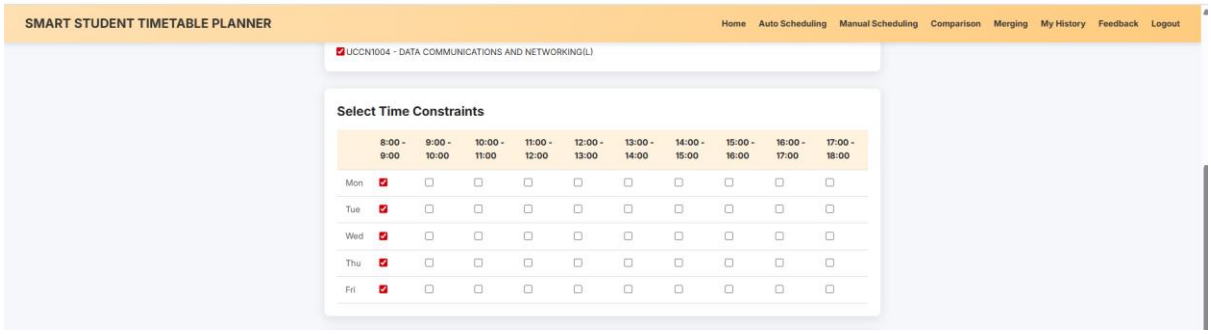


Figure 5.3.4.2 Select Time Constraints

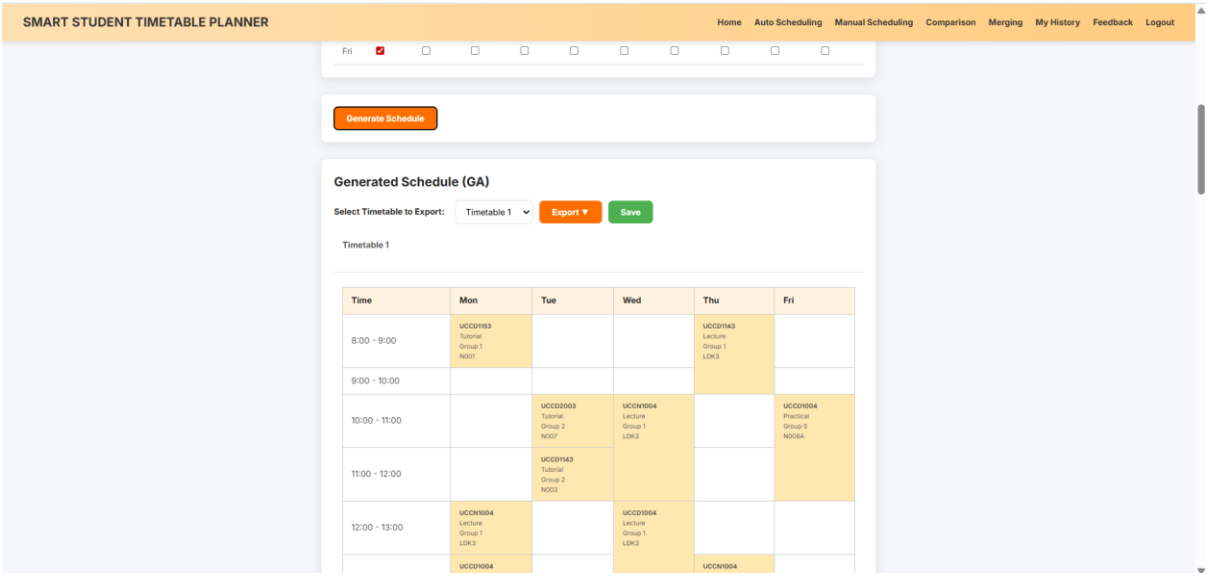


Figure 5.3.4.3 Generate Schedule, Export and Save buttons and Generated Timetables

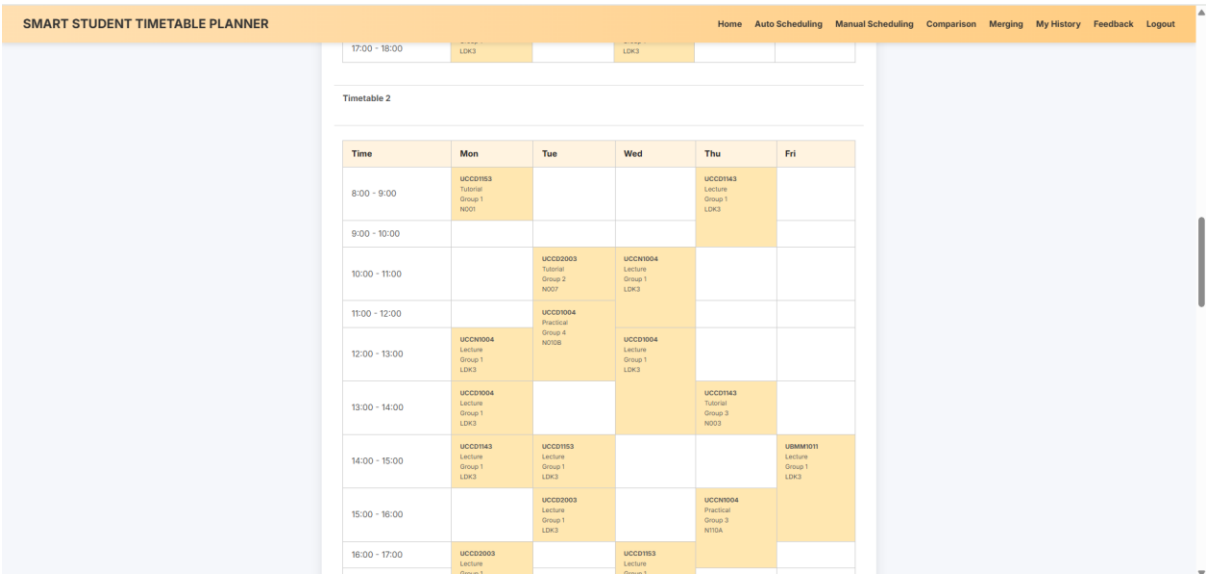


Figure 5.3.4.4 Generated Timetables

5.3.5 Timetable Comparison Page

The Timetable Comparison page is structured to allow students to directly evaluate the differences between their manually created timetable and the automatically generated alternatives. At the top, the navigation bar provides consistently with the rest of the system, ensuring quick access to other key features such as Home, Manual Scheduling, Auto Scheduling, Merging, History, Feedback, and Logout. Below the navigation bar, a clear page header titled “Timetable Comparison” introduces the purpose of the page, reinforcing its role in helping students make informed scheduling decisions.

The main content is divided into two clearly defined sections, displayed within a card-based layout for readability. On the left, the Manual Scheduling Timetable section presents the students manually registered timetable, allowing them to review their chosen courses and timeslots. On the tight, the Auto-Generated Timetable section displays timetables produced through the system's Genetic Algorithm, offering alternatives valid scheduling options. To manage multiple possible schedules, a pagination feature is included at the bottom of the auto-generated section, allowing users to navigate between pages using Previous and Next buttons while the current page number is displayed for reference.

The timetables themselves are rendered in neatly bordered tables with colored headers and clear labeling, ensuring that session details are easy to compare immediately. The use of consistent formatting between the manual and auto-generated schedules supports straightforward side-by-side evaluation. Overall, the interface is designed for clarity, efficiency, and direct comparison, helping students determine whether to retain their manually created schedule or adopt an optimized alternative generated by the system.

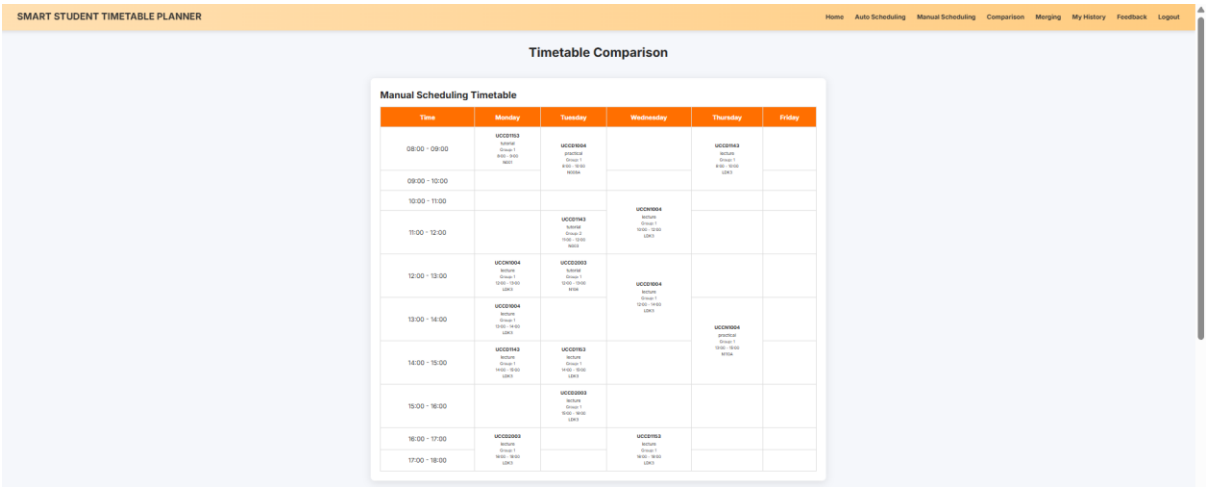


Figure 5.3.5.1 Timetable Comparison Page

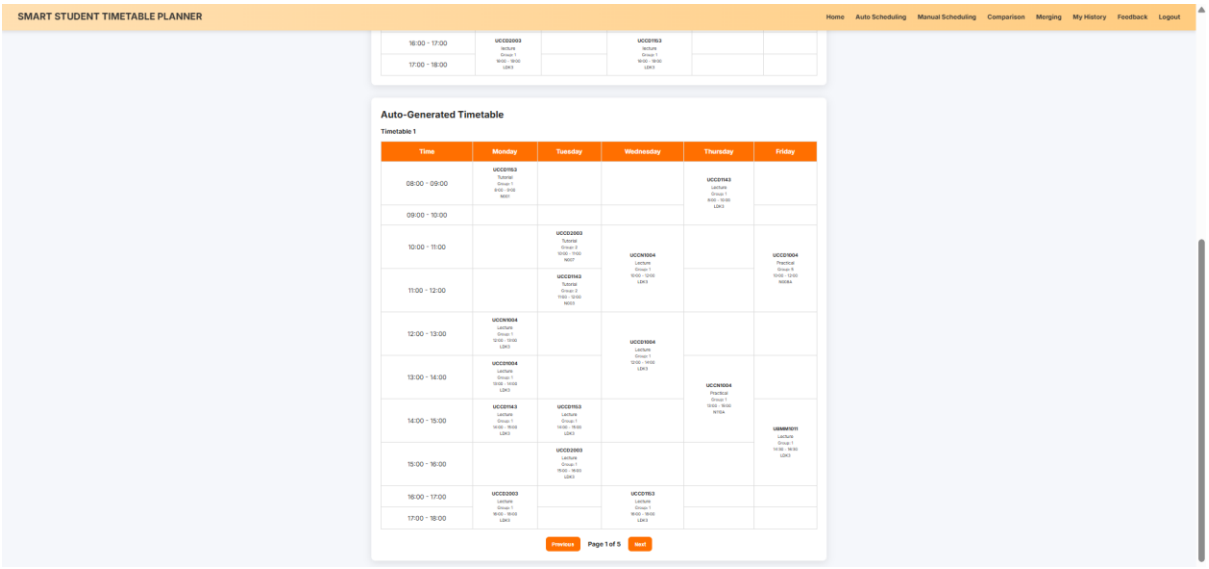


Figure 5.3.5.2 Timetable Comparison Page

5.3.6 Merging Timetable Page

The Merging Timetable Page is designed to provide students with a collaborative platform where multiple users can combine their individual timetables into a shared schedule. At the top of the page, a responsive navigation bar is included to ensure consistency with the rest of the system. It features the system's logo on the left and quick links to key pages such as Home, Auto Scheduling, Manual Scheduling, Comparison, Merging, History, and Feedback on the right. This keeps navigation intuitive and accessible.

The main container adopts a clean, card-style layout with rounded edges and subtle shadows to maintain readability and separation from the background. Within the container, the session management section is prominently placed, giving users options to join an existing collaborative session or create a new session. The session status panel beneath these buttons provides real-time feedback on whether the user is currently connected to a session. This interaction is supported by a floating modal box that appears for session creation or joining, featuring input fields for session name, password, and participants limits.

The timetable visualization itself is structured in a grid layout that organizes days and timeslots clearly. Each cell uses color-coded highlights to indicate the type of entry: green for merged sessions, pink for conflicts, and blur for exclusive sessions. A legend is provided above the grid to ensure users can easily interpret the meaning of each color. Beyond this static visualization, the design also introduces highlighted slots that can be shifted interactively.

When users click on a practical or tutorial session, the system highlights alternative available timeslots where the session could be moved. This provides participants with direct control over resolving clashes by selecting alternative slots without needing to regenerate the timetable entirely.

Additional interaction elements further enhance collaboration. Users can select their personal timetable from a dropdown list, preview it in a compact mini-grid format, and submit it to the session. A participants list is shown in real-time, reflecting who has joined the session and which timetables have been submitted. Notifications and submission status are displayed to keep all users updated on progress.

To support teamwork and communication, a session chat panel is integrated at the bottom of the page. This live chat allows participants to coordinate decisions and resolve conflicts while merging. Export functionality is also included, allowing the finalized timetable to be downloaded either as an image or Excel file for personal use or sharing. Finally, a "Save" button enables students to store their merged timetable into the system's history for later reference.

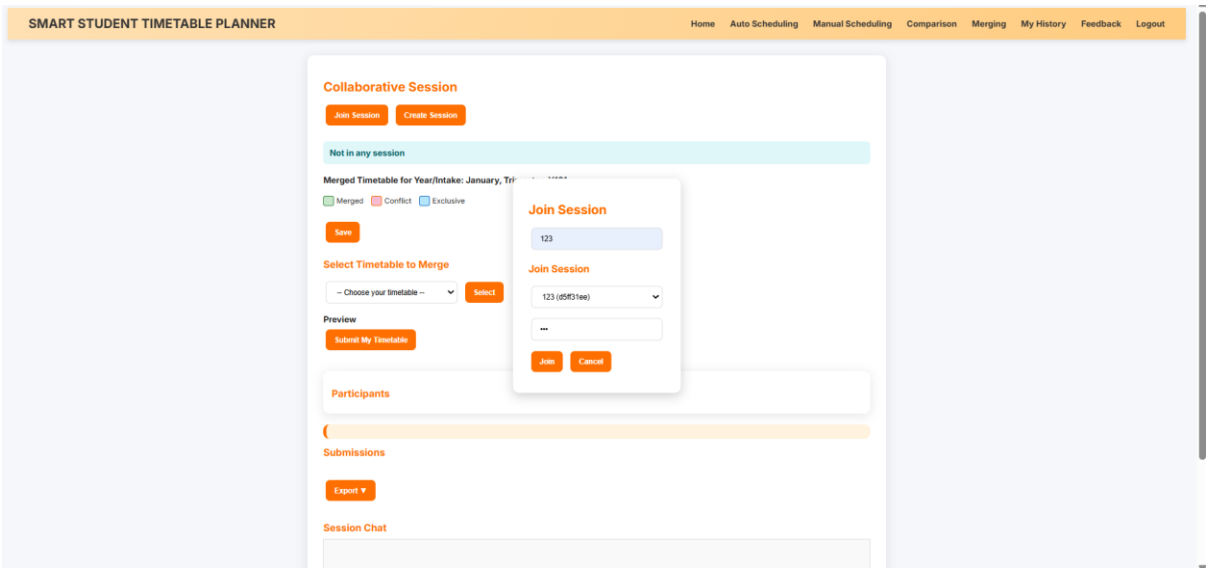


Figure 5.3.6.1 Join Collaborative Session

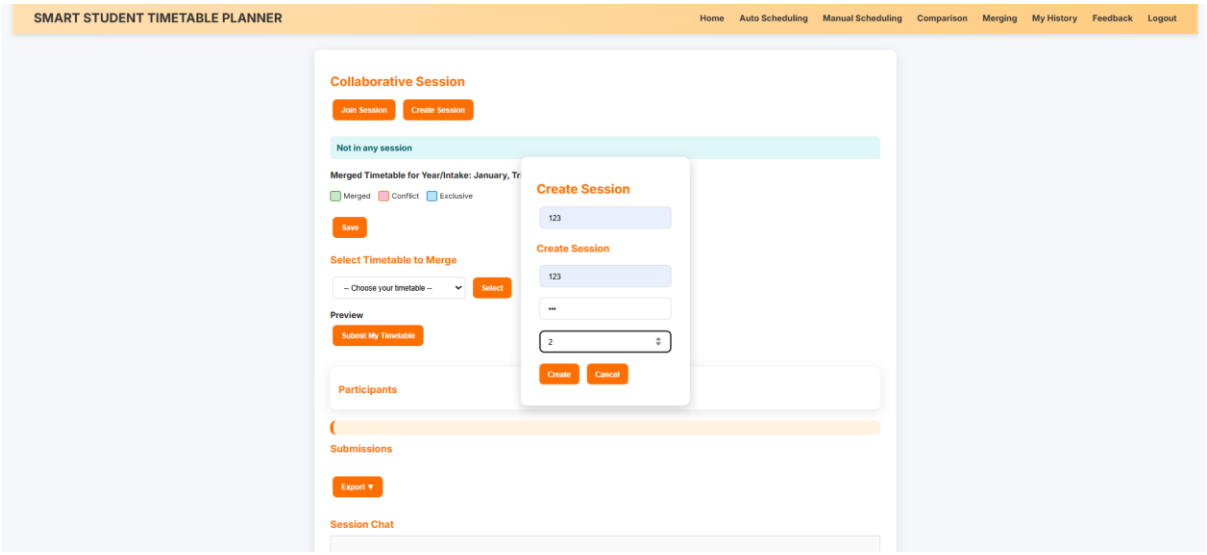


Figure 5.3.6.2 Create Collaborative Session

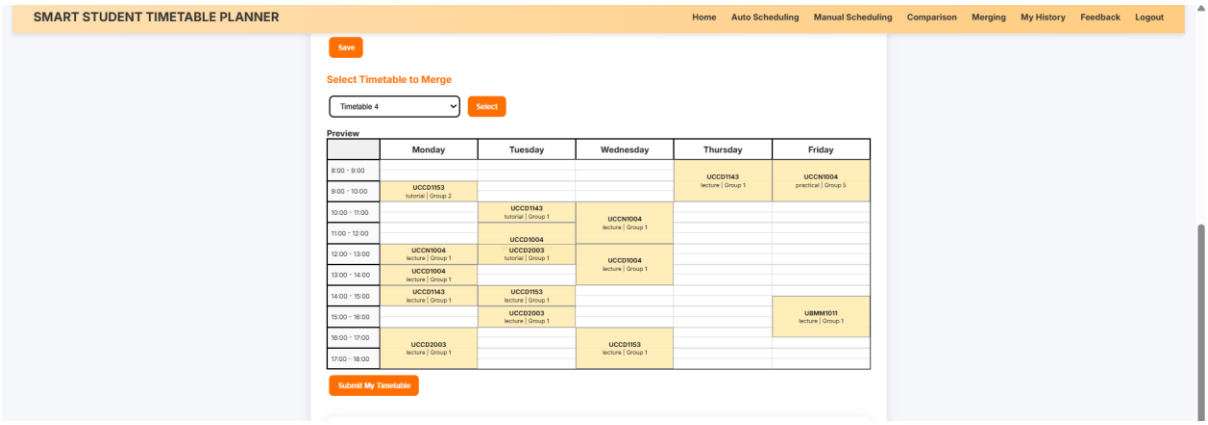


Figure 5.3.6.3 Preview Selected Timetable

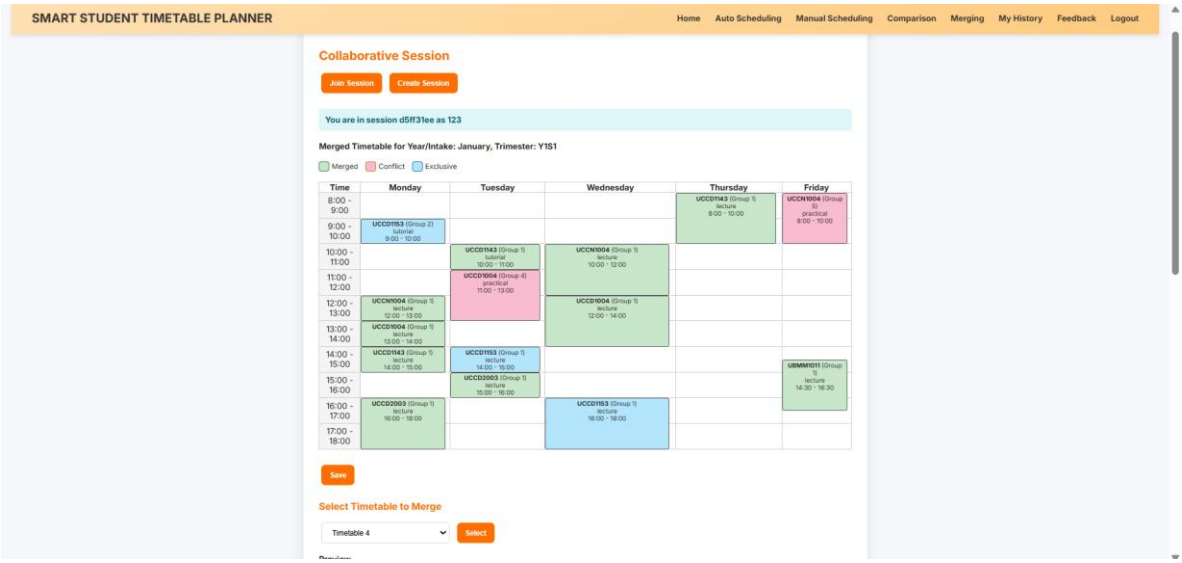


Figure 5.3.6.4 Preview Merged Timetable

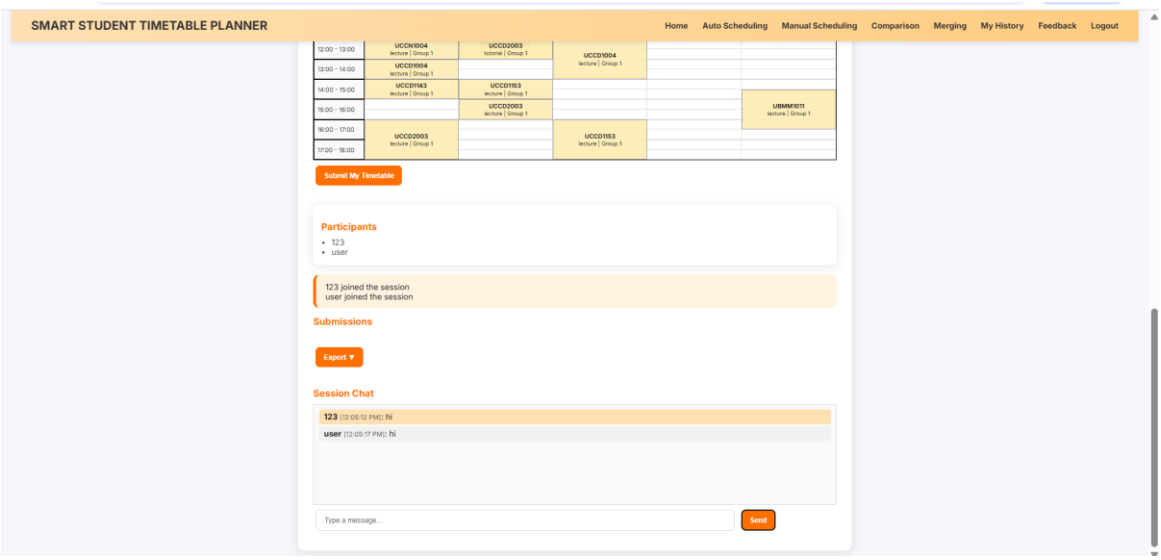


Figure 5.3.6.5 Session Chat

5.3.7 Timetable History Page

The Timetable History Page provides students with a structured and user-friendly interface to manage their previously saved timetables. A consistent navigation bar at the top ensures seamless movement across modules such as Auto Scheduling, Manual Scheduling, Comparison, and Merging, while clearly highlighting the active page. This consistent design enhances usability and creates a smooth transition across different features of the platform.

The main content area is organized into collapsible sections that group timetables by mode (manual, auto, or merged), then further by intake and trimester. Each timetable entry displays key details such as its label and the date and time it was saved. Color-coded badges help distinguish between timetable types, allowing users to identify schedules immediately. A filter option above the list also enables users to refine their view based on scheduling mode, improving efficiency in locating specific timetables.

To support quick interaction, each timetable entry includes action buttons for viewing or deleting. Viewing opens a model overlay that displays a high-quality preview of the timetable, while the delete option allows users to remove unnecessary entries with confirmation. This combination of grouping, filtering, and previewing ensures the page balances clarity with interactivity, giving students full control over their scheduling history.

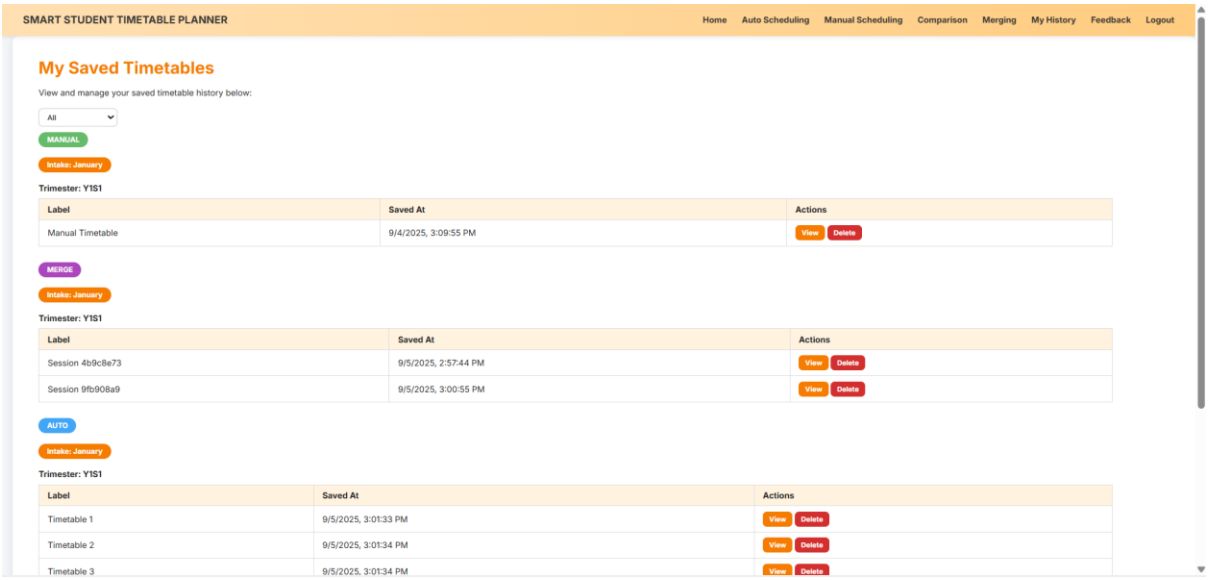


Figure 5.3.7.1 Timetable History Page

5.3.8 User Feedback Page

The User Feedback Page is designed to provide students with an intuitive and interactive navigation bar at the top, ensuring easy access to other modules such as scheduling, merging, comparison, and history, while clearly highlighting the active feedback section. The layout follows a card-based structure, giving the page a clean and organized look. The first card contains the feedback submission form, where users can enter their username, select a feedback type (general, bug, or suggestion), and input their message in a text area. A clearly styled submission button completes the form, supported by a floating “thanks” box that briefly appears after successful submission to reinforce user acknowledgment.

Below the submission form, the page displays the Replied Feedback section, which organizes admin responses into Unread and Read categories. Unread feedback entries are highlighted with a light background to draw attention, and each entry includes the original message, the admin’s reply, and a “Mark as Read” button for managing visibility. Once marked as read, entries automatically shift into the collapsible Read section, which can be expanded or collapsed on demand. A red notification dot in the navigation bar provides a real-time visual cue for users when new replies are received, improving responsiveness and ensuring students do not miss updates.

The interface also integrates real-time interactivity through Socket.IO, allowing admin replies and user actions such as marking messages as read, to update the page instantly without requiring a refresh. This design choice enhances engagement and streamlines communication between students and administrators.

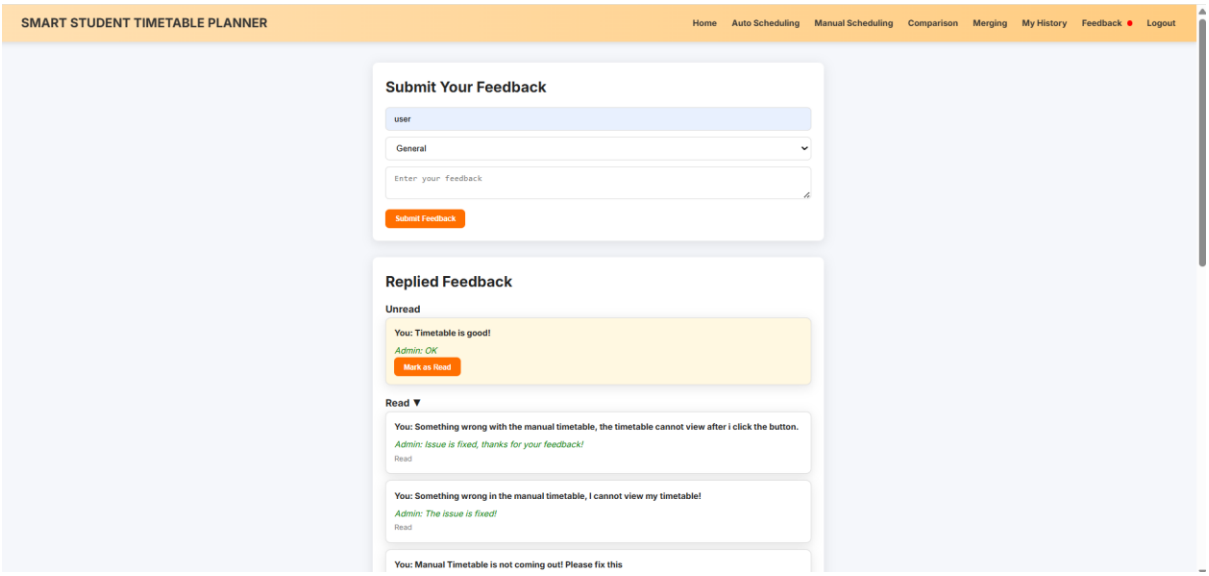


Figure 5.3.8.1 User Feedback Page

5.3.9 Admin Dashboard

The Admin Dashboard Page provides administrators with a centralized and intuitive interface to manage the Smart Student Timetable Planner. The page adopts a clean, card-based layout with a navigation bar at the top and left-aligned sidebar for quick access to core features, including course uploads, history, previews, created sessions, and feedback. Visual consistency is maintained through a warm orange theme, rounded cards, and responsive design, ensuring usability across devices. Quick-action cards in the main content area allow admins to instantly access key functions such as uploading courses, viewing upload history, and previewing parsed data, with supporting statistics to provide context.

To enhance system monitoring, the dashboard integrates a bar chart using Chart.js to visualize uploads per month, helping administrators track activity trends over time. Below, a recent uploads table lists the latest course files with timestamps, ensuring transparency and easy verification of updates. Real-time interactivity is incorporated through Socket.IO, enabling instant notification for new or pending feedback, indicated by a dynamic red dot in the sidebar.

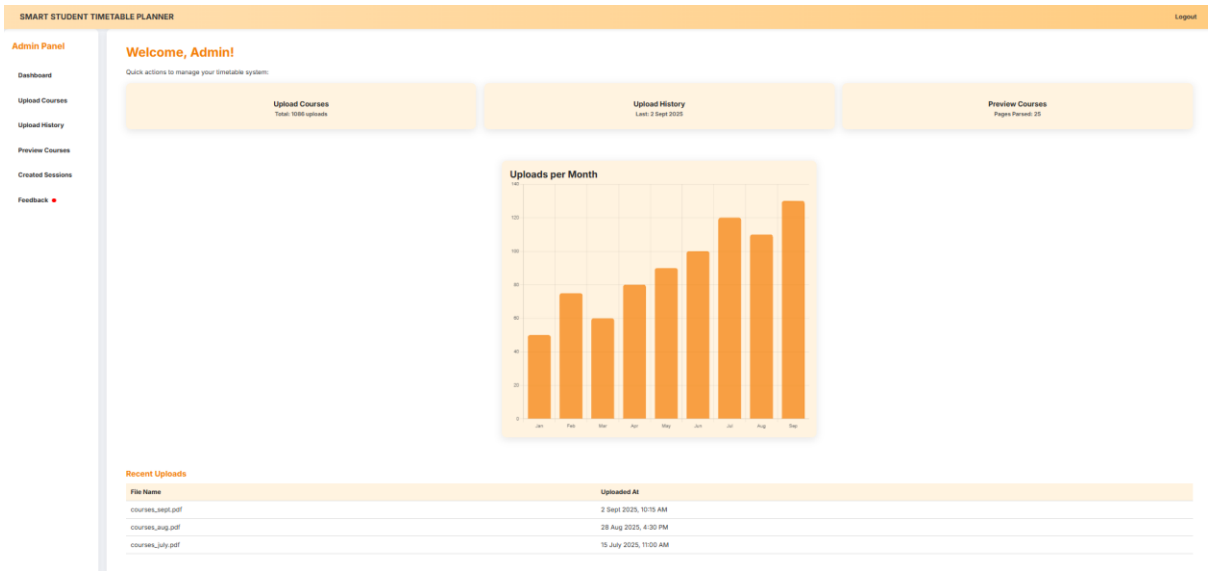


Figure 5.3.9.1 Admin Dashboard Page

5.3.10 Admin Upload Course Page

The Admin Upload Course Page provides a simple and efficient interface for administrators to upload PDF files containing course schedules. The page adopts a consistent layout with the main navigation bar at the top and a sidebar on the left for quick access to dashboard functions such as course uploads, history, previews, sessions, and feedback. The main content area highlights the upload functionality, featuring a custom file selection box with a dashed border design that emphasized drag-and-drop or click-to-select interactions. Once a file is selected, the filename is displayed for confirmation, ensuring transparency before submission.

To enhance user experience, the interface integrates a progress bar that visually tracks the upload and parsing process. The bar dynamically updates from initial upload to processing and completion, with clear color-coded states for progress, success, or error. This feedback mechanism ensures that administrators remain informed about the status of their uploads in real time. Additionally, real-time notification indicators in the sidebar alert admins of pending feedback, ensuring that while managing uploads, they can also stay responsive to student input.

Figure 5.3.10.1 Admin Upload Course Page

5.3.11 Admin Upload Course Page

The Upload History Page provides administrators with a structured and user-friendly interface to review past course uploads. The page follows the same consistent layout as other admin pages, with a top navigation bar for logout access and a left-hand sidebar offering quick navigation to dashboard functions such as uploading, previewing courses, viewing sessions, and managing feedback. A notification indicator is also embedded in the feedback menu item, ensuring that pending student feedback remains visible while administrators manage upload records.

At the center of the interface, the main content area displays a clear table listing uploaded course file alongside their upload timestamps. This tabular design allows administrators to quickly verify successful uploads and track submission timelines. The table is styled with alternating rows, borders, and responsive behaviors to ensure readability across devices. For smaller screens, the layout adapts by transforming rows into block displays with labels, preserving clarity on mobile. Overall, the design emphasized accessibility, consistency, efficiency, enabling administrators to manage and verify uploaded course data with ease.

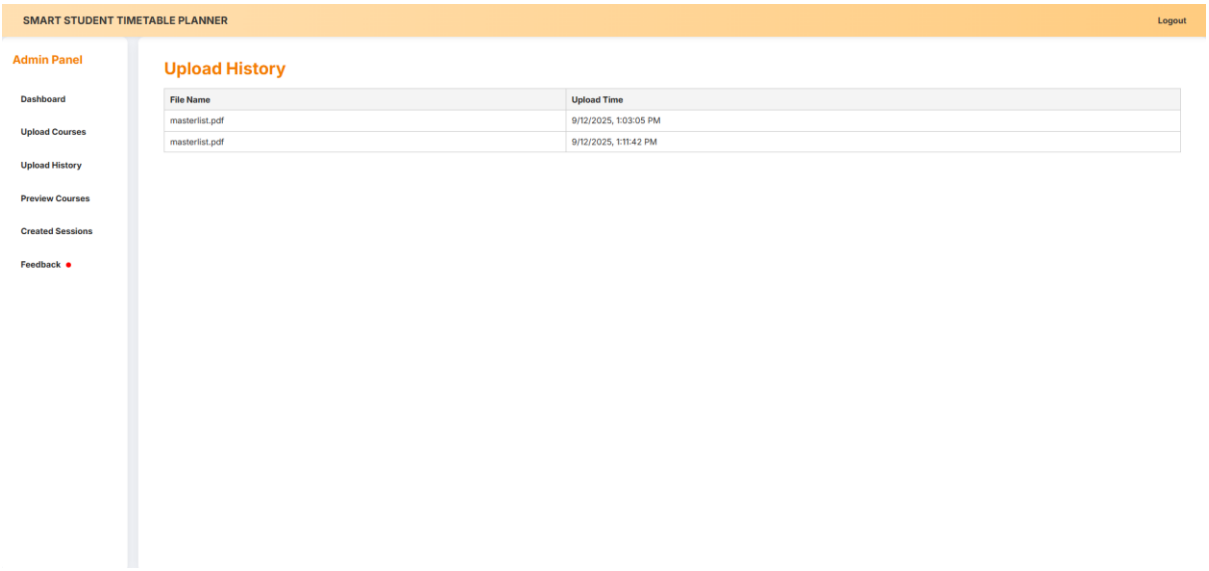


Figure 5.3.11.1 Upload History Page

5.3.12 Preview Courses Page

The Preview Courses Page provides administrators with a structured and efficient interface for reviewing uploaded course data. The design follows a consistent layout used across the admin panel, featuring a top navigation bar for system-wide access and a sidebar for quick navigation to modules such as dashboard, upload, history, sessions, and feedback. A notification indicator in the feedback menu item ensures that pending student responses remain visible while the admin is managing course data. The main content area displays the most recently uploaded file information at the top, ensuring clarity on the source of the previewed data.

The core of the interface is a tabular course preview, where course details such as trimester, intake, code, name, group, day, start and end time, venue, and session type are displayed in a clean, column-based layout. To improve usability, the table integrates pagination controls at the bottom, allowing administrators to navigate through large datasets in batches of 30 rows. The pagination includes both “Prev” and “Next” buttons with state-based enabling and disabling, as well as a page info label for clarity.

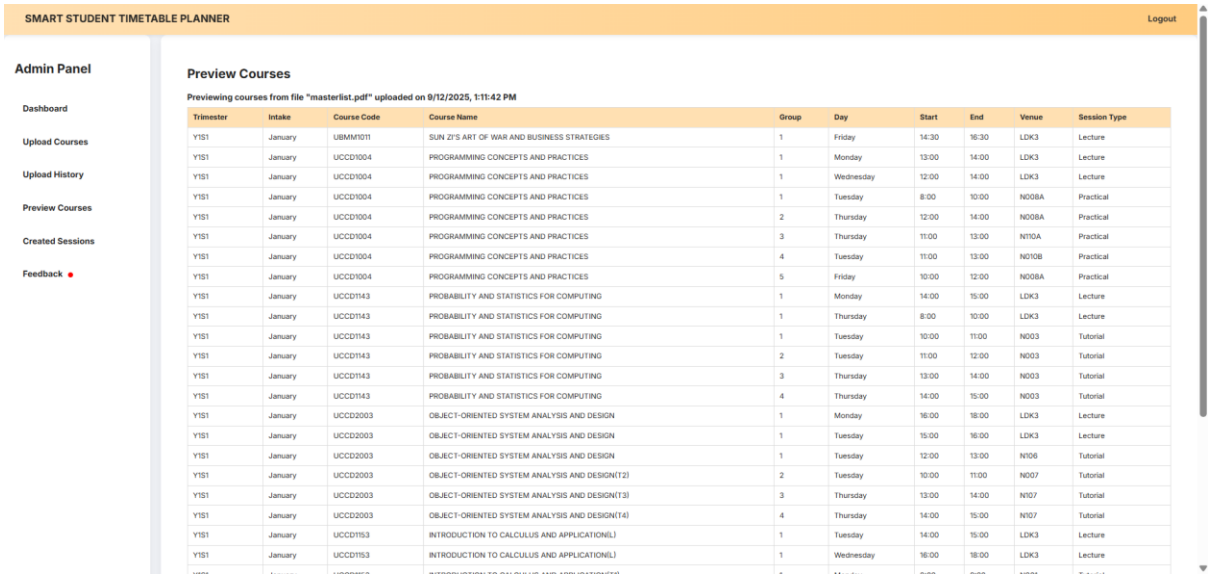


Figure 5.3.12.1 Preview Courses Page

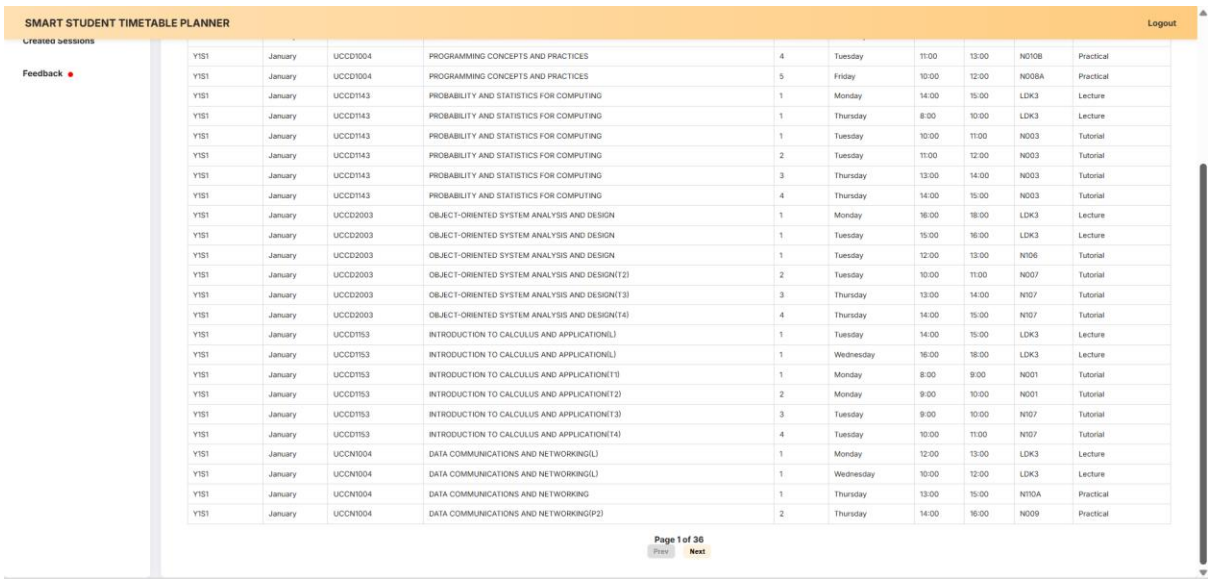


Figure 5.3.12.2 Pagination Button in Preview Courses Page

5.3.13 View Created Sessions Page

The View Created Sessions page provides administrators with a clear interface to monitor collaborative scheduling sessions within the system. The layout follows the admin panel's consistent design, beginning with a navigation bar at the top for logout access and a sidebar for quick navigation to core features such as dashboard, upload, history, preview, and feedback, with a notification indicator highlighting pending feedback. The main content area focuses on displaying all active and past sessions, beginning with a header and brief description for context, followed by a structured sessions table.

The table presents essential session details in a simple, column-based format, including session name, session ID, creation time, and the list of participants with a count. Dynamic updates are integrated through real-time Socket.IO events: newly created sessions appear instantly, while participants count update automatically as users join. This ensures that administrators can monitor session activity without manual refresh, keeping the interface responsive, transparent, and efficient for managing collaborative timetable sessions.

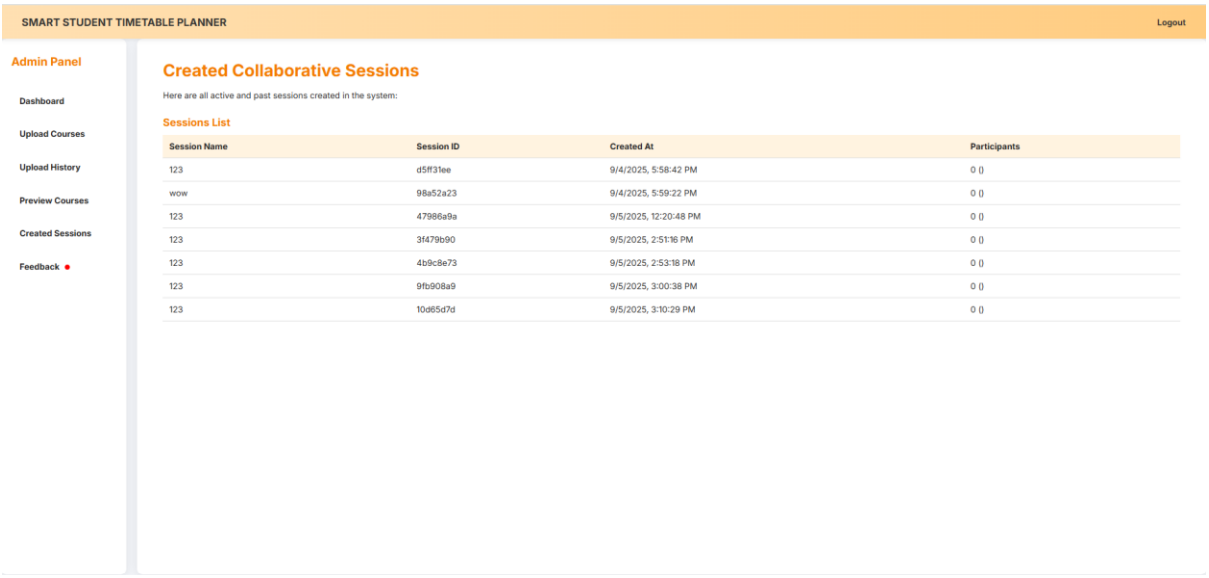


Figure 5.3.13.1 View Created Sessions Page

5.3.14 Admin Feedback Page

The Admin Feedback Page is designed to streamline the management of user feedback within the Smart Student Timetable Planner. The layout follows the standard admin dashboard design, with a top navigation bar for logout and a sidebar for quick access to different features. A notification badge on the sidebar highlights the number of pending feedback items, ensuring administrators are alerted to new or unresolved issues. The main content area is centered around a feedback management table, where feedback is organized into sections, Pending, Read, and Replied, each visually differentiated with colored headers for quick recognition.

Administrators are provided with interactive controls to handle feedback efficiently. A filter bar at the top allows sorting by type (general, bug, suggestion) and status, enabling focused management. Pending feedback items include action buttons to mark as read or open a modal for direct replies, where responses can be typed and sent back to users. Once replied, the

feedback is updated dynamically in the table and tracked in real time using Socket.IO, keeping the interface responsive and up to date. This design ensures feedback is systematically categorized, easily traceable, and manageable in a clear, user-friendly environment.

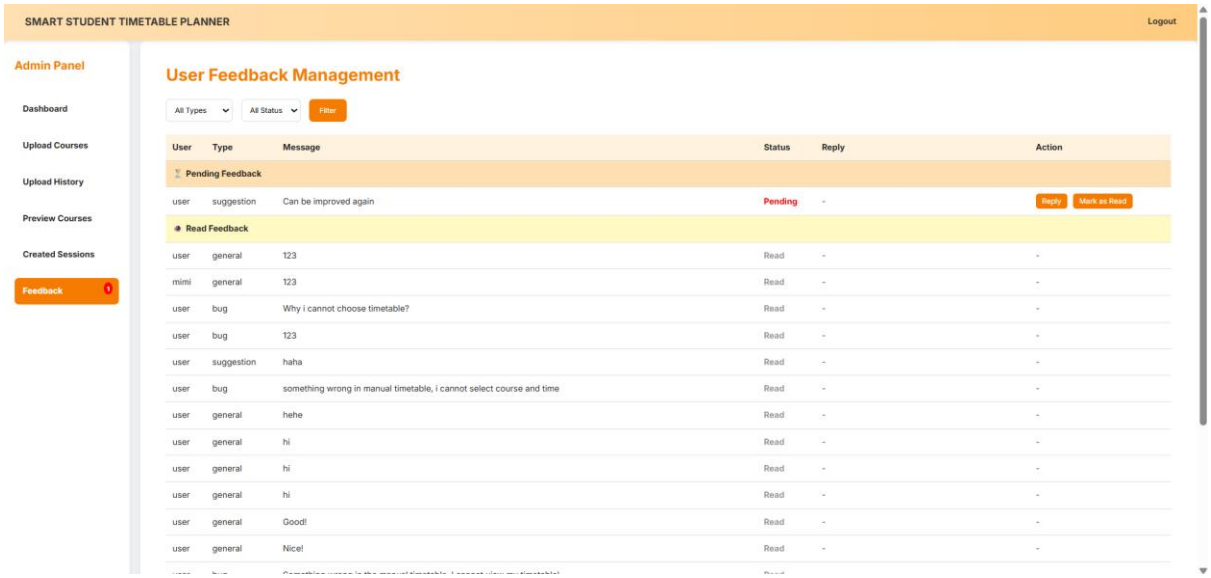


Figure 5.3.14.1 Pending and Read section in Admin Feedback Page

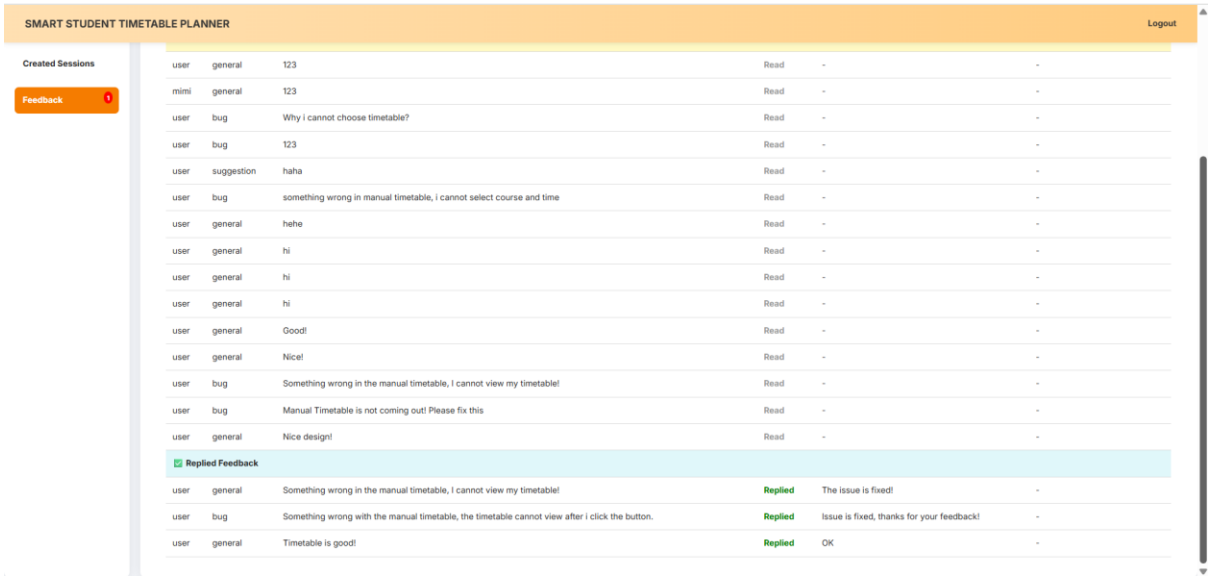


Figure 5.3.14.2 Replied section in Admin Feedback Page

5.4 Implementation Issues and Challenges

There are some issues and challenges faced during the development of the Smart Student Timetable Planner.

- **Challenges in Developing the Genetic Algorithm for Auto Scheduling**

  Implementing a Genetic Algorithm (GA) for course scheduling presents several technical and logical challenges. One major issue involves defining a suitable chromosome structure that accurately represents possible timetable configurations, including lectures, tutorials, and practicals. Designing a fitness function that balances multiple constraints, such as session conflicts, time availability, and course type requirements, is another complexity. Additionally, ensuring that the selection, crossover, and mutation processes maintain timetable validity without introducing conflicts or incomplete schedules requires careful algorithm tuning. The random nature of GAs can also lead to inconsistent results or excessive computation time when dealing with large course datasets, requiring optimization strategies such as elitism, constraint filtering, or population pruning.

- **Challenges in Implementing the Merging Timetable Page**

  Another issue faced during the implementation of the merging timetable page is ensuring real-time synchronization and consistency across all users in a session. When multiple users submit or adjust their timetables simultaneously, conflicts may arise in merging sessions, especially when handling overlapping slots or choosing between tutorial and practical alternatives. Another challenge is designing a clear and interactive interface that highlights modifiable slots while preventing accidental shifts of fixed lecture sessions. Additionally, maintaining efficient performance of the genetic algorithm for schedule synthesis while ensuring smooth updates on the client side posed difficulties, particularly in managing state transitions and reflecting changes instantly across all connected users.

5.5 Summary

In summary, the system implementation chapter details the setup and execution of the Smart Student Timetable Planner, highlighting its client-server architecture, responsive user interfaces, and real-time collaboration features. It emphasizes how hardware and software components were integrated to support both student and admin functionalities, while also addressing challenges such as synchronization, conflict management, and optimization of the genetic algorithm. This implementation ensures the system is practical, user-friendly, and capable of handling dynamic scheduling requirements effectively.

Chapter 6

System Evaluation and Discussion

In this chapter focused on two main aspects: the experimentation with the Genetic Algorithm (GA) and the comparison between real-world timetables and system-generated timetables. The GA experiments examined how parameters such as population size, crossover, and mutation rates influenced the quality and diversity of valid schedules, ensuring compulsory lectures were included and practical/tutorial sessions were optimally allocated without conflicts. In parallel, the comparison with actual university timetables confirmed the system's accuracy and reliability, showing that generated schedules closely matched real-world arrangements while offering greater flexibility and efficiency.

6.1 Experiment on Genetic Algorithm

6.1.1 Overview

This experiment investigated the use of a Genetic Algorithm (GA) to optimize student timetable scheduling for a set of Year 1 Semester 1 (Y1S1) university courses. The goal is to automatically generate feasible combinations of course sessions that minimize schedule conflicts and adhere to student-defined preferences. GA evolves a population of candidate timetables using biologically inspired operations such as crossover, mutation, and selection, to iteratively improve solution quality.

A fitness function is used to evaluate each timetable in the population based on the number of conflicts and violations of constraints. The system also employs early stopping criteria, either when a desired fitness level is reached or when improvement stagnates over several generations.

6.1.2 Code Structure and Functionality

The system begins by uploading a sample dataset from the masterlist.csv which contains course offerings. It filters the data for the target trimester (Y1S1) and selected courses. Lecture sessions are fixed and treated as compulsory, whereas tutorial or practical sessions are variable and subject to scheduling.

Figure 6.1.2.1 Experiment Settings and Filter Trimester

Each GA individual is a list of integers, each indicating the selected index of an optional session for one of the courses. For example, an individual [0,1,2,0,1,0] represents one specific combination of tutorial or practical sessions for the six selected courses. The fitness function evaluates the quality of a timetable by applying penalties for:

- Class overlaps: Sessions that occur at overlapping times on the same day.
- Friday sessions: Student prefers a free Friday.
- Classes before 9:00 AM: Early classes are undesirable.

These constraints are incorporated by converting session times to minutes and calculating overlaps. The function returns a fitness score inversely proportional to total penalties. This means the fitness ranges from near 0 (worst case with many conflicts) to near 1 (ideal schedule). However, due to the fixed nature of lecture sessions, some penalties may be unavoidable.

Figure 6.1.2.2 Fitness function for constraints

6.1.3 Genetic Algorithm Design

An initial population of 20 individuals is randomly generated using available optional sessions for each course.

The tournament selection method chooses the best individual out of a randomly sampled group size of 3. This method balance exploration and exploitation by favouring fitter individuals while allowing weaker ones a chance.

```
def tournament_select(pop, fits, k=3):
    out = []
    for _ in pop:
        aspirants = random.sample(list(zip(pop, fits)), k)
        out.append(max(aspirants, key=lambda x: x[1])[0].copy())
    return out
```

Figure 6.1.3.1 Tournament Selection function

Crossover involves selecting a random point in the genome and swapping gene segments between two parents to produce two offspring. The operation introduces new combinations of session choices, which helps the algorithm explore the solution space.

```
def crossover(a, b):
    pt = random.randint(1, len(a) - 1)
    return a[:pt] + b[pt:], b[:pt] + a[pt:]
```

Figure 6.1.3.2 Crossover function

Mutation randomly alters genes (session selections) with a given probability (mut_rate). This introduced diversity and helps prevent the population from getting stuck in local optima.

```
def mutate(gen, rate):
    for i in range(len(gen)):
        if random.random() < rate and optionals[SELECTED[i]]:
            gen[i] = random.randrange(len(optionals[SELECTED[i]]))
    return gen
```

Figure 6.1.3.3 Mutate function

```
        # evolve
        pop = tournament_select(pop, fits)
        next_pop = []
        for i in range(0, len(pop), 2):
            p1, p2 = pop[i], pop[(i+1) % len(pop)]
            if random.random() < cx_rate:
                c1, c2 = crossover(p1, p2)
            else:
                c1, c2 = p1.copy(), p2.copy()
            next_pop += [c1, c2]
        pop = [mutate(ind, mut_rate) for ind in next_pop]
```

Figure 6.1.3.4 Evolve function

There are two stopping criteria are used to terminate the GA:

- Fitness Threshold: If the best fitness in a generation exceeds 0.95, the algorithm stops early, indicating a near-optimal timetable.
- Stagnation: If there is no improvement in the best fitness for 15 consecutive generations, the algorithm halts to save computation time.

```python
# 4) GA runner with early stopping
def run_ga(pop_size=20, gens=100, cx_rate=0.7, mut_rate=0.1,
           threshold=0.95, patience=15):
    pop = init_pop(pop_size)
    best_hist, avg_hist = [], []
    best_so_far, stagnant = 0, 0

    for g in range(gens):
        fits = [fitness(ind) for ind in pop]
        best, avg = max(fits), sum(fits)/len(fits)
        best_hist.append(best)
        avg_hist.append(avg)

        # threshold stopping
        if best >= threshold:
            print(f"✓ Reached threshold {threshold} at gen {g}")
            break
        # stagnation stopping
        if best > best_so_far:
            best_so_far, stagnant = best, 0
        else:
            stagnant += 1
            if stagnant >= patience:
                print(f"■ Stopped—no improvement for {patience} gens at gen {g}")
                break
```

Figure 6.1.3.5 GA with stopping criteria

6.1.4 Experimental Results and Graph Analysis

The algorithm was executed with five different crossover and mutation configurations, where cx = crossover rate and mu = mutation rate:

- Set A: cx = 0.7, mu = 0.01
- Set B: cx = 0.7, mu = 0.1
- Set C: cx = 0.9, mu = 0.1
- Set D: cx = 0.9, mu = 0.3
- Set E: cx = 0.9, mu = 0.5

Each configuration was run with a population of 20 for a maximum of 100 generation. The results are visualized in two plots:

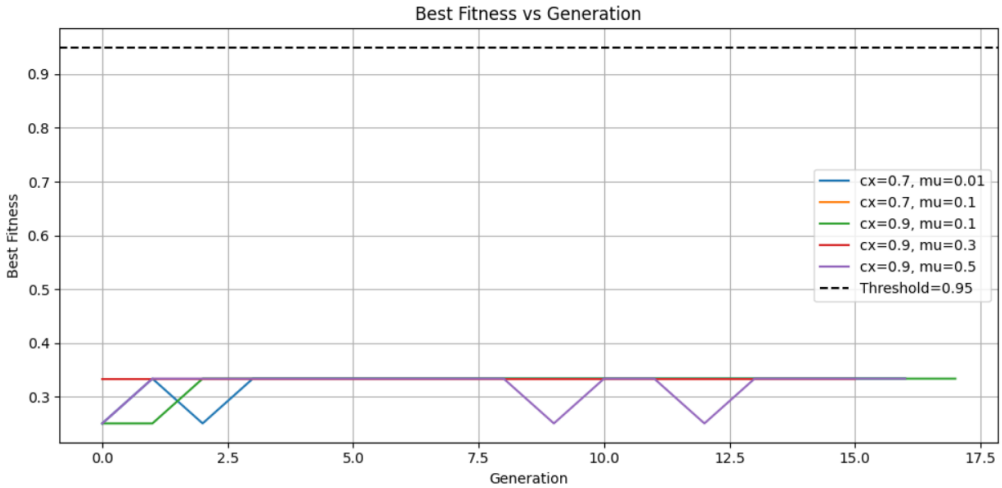


Figure 6.1.4.1 Graph of Best Fitness vs Generation

The graph depicting best fitness versus generation illustrates the highest fitness score achieved in each generation for various configurations of crossover and mutation rates. A key

observation is that none of the Genetic Algorithm (GA) runs reached the preset threshold fitness value of 0.95, which was defined as a desirable quality benchmark for an optimal timetable. Instead, most configurations plateaued between 0.25 and 0.35, indicating a premature convergence to suboptimal solutions.

This stagnation in fitness can be attributed largely to hard constraints imposed by fixed lecture sessions. Lecture slots are not subject to the GA's evolutionary operators, such as mutation and crossover and therefore remain unchanged throughout the optimization process. These sessions may inherently contain conflicts such as overlapping with one another or with available practical/tutorial sessions, or they may occur during undesirable times, such as Fridays or before 9:00AM, both of which incur penalties in the fitness evaluation. As a result, even the best possible combination of optimal sessions cannot entirely offset these penalties, effectively capping the maximum achievable fitness.

Furthermore, configurations involving higher mutation rates exhibit greater fluctuations in best fitness values across generations. This indicates instability in solution quality, where beneficial traits discovered in earlier generations may be frequently disrupted due to excessive mutation. Consequently, while high mutation promotes exploration, it also increases the risk of the population diverging from promising regions of the search space.

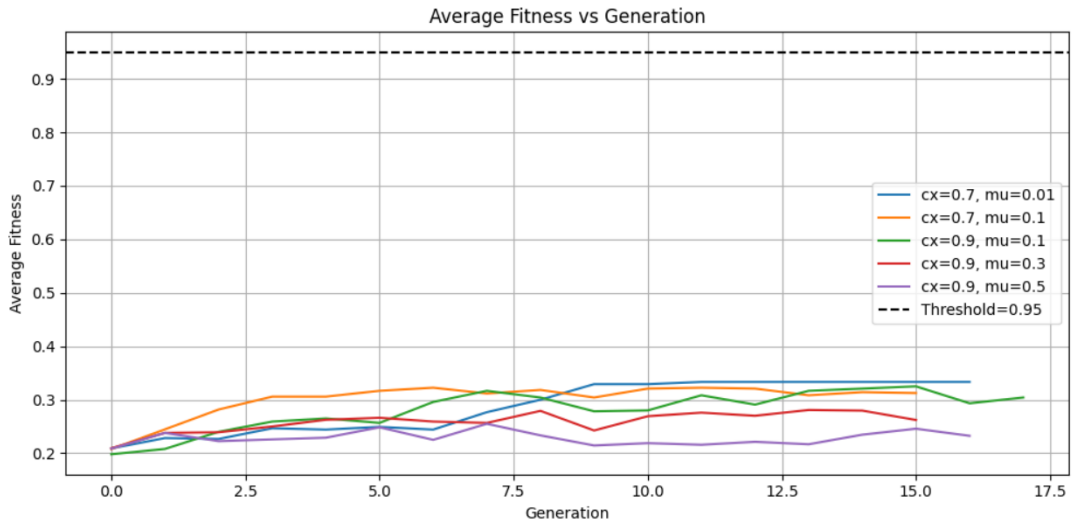


Figure 6.1.4.2 Graph of Fitness vs Generation

The graph showing average fitness across generation offers insights into the overall performance and evolutionary stability of the population under each configuration. In general, all tested parameter sets demonstrate incremental improvements in the early generations, suggesting that the GA can initially identify and propagate advantageous genes.

Configuration employing moderate mutation rates (mu = 0.1) tend to yield smoother and more consistent improvements in average fitness. This balance allows for controlled exploration of the solution space while preserving good solutions over generations. In contrast, high mutation rates introduce significant randomness into the population, resulting in erratic fitness trajectories and an overall failure to converge toward optimal or even moderately good solutions. The inconsistency is especially problematic in this context due to the small solution space constrained by fixed sessions.

In summary, lower to moderate mutation rates strike a better balance between exploration and exploitation, encouraging the algorithm to gradually refine solutions without discarding valuable traits. Excessive mutation, while theoretically increasing diversity, ultimately hinders convergence by constantly altering promising genes and destabilizing the population.

6.1.5 Interpretation of GA Parameters

Crossover Rate (cx) enables recombination of potentially beneficial traits (session selections) between individuals. High crossover rate (0.9) promotes exploration but relies on the presence of good genes to mix effectively. Without diverse or high-quality genes in the population, crossover alone is insufficient for fitness improvement.

Table 6.1.5.1 Summary of the Experimental Results

Set	Crossover Rate (cx)	Mutation Rate (mu)	Best Fitness	Average Fitness	Stability
A	0.7	0.01	Low	Very slow increase	Stable but suboptimal
B	0.7	0.1	Moderate	Steady upward trend	Stable
C	0.9	0.1	Good	Fast improvement early	Stable
D	0.9	0.3	Moderate	Fluctuating gains	Slightly unstable
E	0.9	0.5	Poor	Random fluctuations	Unstable

6.2 Comparison of Real-World Timetable and Generated Timetable

6.2.1 Overview

In this section, the generated timetable using Smart Student Timetable Planner system was compared directly with the official timetable using university system. The primary purpose of this comparison was to verify the accuracy and reliability of the system in replicating actual schedules without errors, mismatches, or missing sessions.

The main objective of this testing was to determine whether the Smart Student Timetable Planner can produce a timetable identical to the official real-world timetable. This involved verifying that the courses, session types such as lectures, tutorials, and praticals, and their corresponding time slots were all consistent with the university's published schedule. The test also aimed to confirm that no extra classes or shortened sessions were generated by the system.

6.2.2 Test Environment and Methodology

The testing was conducted on a standard laptop with 8GB of RAM and an intel i7 processor, using the web-based Smart Student Timetable Planner application. The test was performed on

the Google Chrome browser, while the reference timetable was obtained from the university's official academic portal. This ensured a consistent and reliable environment for testing and comparison.

The testing procedure began by logging into the Smart Student Timetable Planner and generating a timetable for the chosen trimester using the auto-scheduling function. The official timetable for the same trimester was retrieved using the auto-scheduling function. The official timetable for the same trimester was retrieved from the university portal. Both timetables were carefully compared by examining course codes, session types, allocated days and times, session durations. Any potential differences, such as missing or additional sessions, incorrect time slots, or mismatched durations, were noted. Screenshots of both timetables were captured to provide supporting evidence for the analysis.

6.2.3 Test Data

The data used for testing included two sources: the timetable generated by Smart Student Timetable Planner system and the official real-world timetable from the university portal. Both contained the same set of courses for the selected trimester, which allowed a direct one-to-one comparison. Two official real-world timetables from the university portal were collected from two respondents, which are: Wong Xin Tong, Y3S3, and Elaine Chung Hui Lin, Y3S3. From these two respondents' timetables, the system has generated two timetables using the Smart Student Timetable Planner as below:

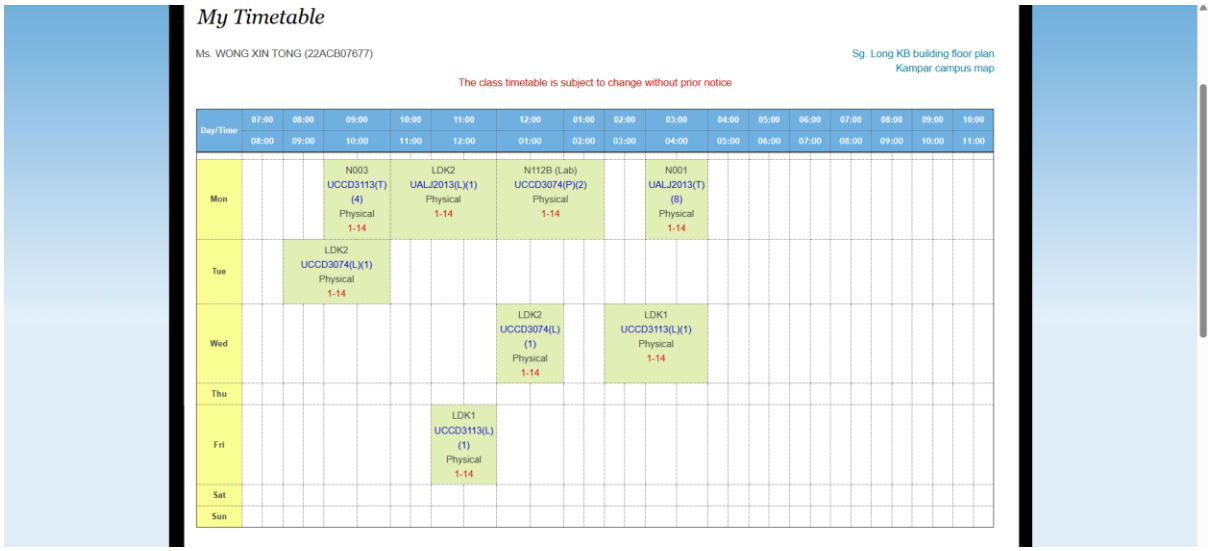


Figure 6.2.3.1 Wong Xin Tong's real-world timetable from official university portal

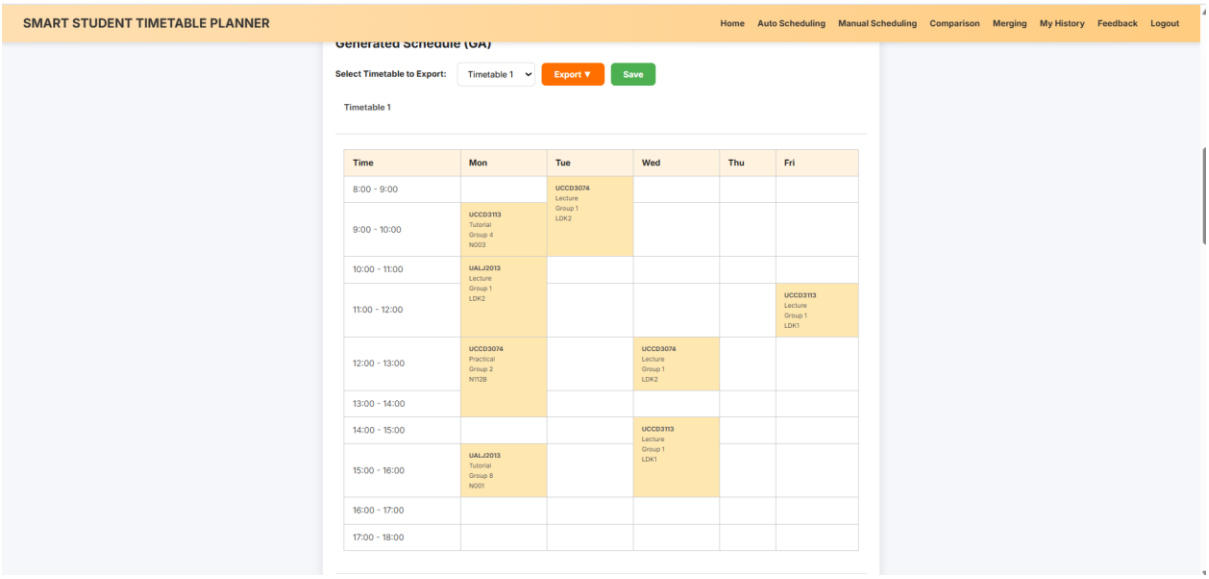


Figure 6.2.3.2 Wong Xin Tong's generated timetable from Smart Student Timetable Planner

Day/Time	07:00 08:00	08:00 09:00	09:00 10:00	10:00 11:00	11:00 12:00	12:00 01:00	01:00 02:00	02:00 03:00	03:00 04:00	04:00 05:00	05:00 06:00	06:00 07:00	07:00 08:00	08:00 09:00	09:00 10:00	10:00 11:00
Mon			N003 UCCD3113(T) (4) Physical 1-14							N006 UCCD3064(L)(1) Physical 1-14						
Tue			N006 UCCD3064(L) (1) Physical 1-14													
Wed								LDK1 UCCD3113(L)(1) Physical 1-14		LDK1 UALB1003(L)(3) Physical 1-14						
Thu		N112A (Lab) UCCD3064(P)(2) Physical 1-14		N002 UALB1003(T) (2) Physical 1-14												
Fri					LDK1 UCCD3113(L) (1) Physical 1-14											
Sat																
Sun																

Figure 6.2.3.3 Elaine's real-world timetable from official university portal

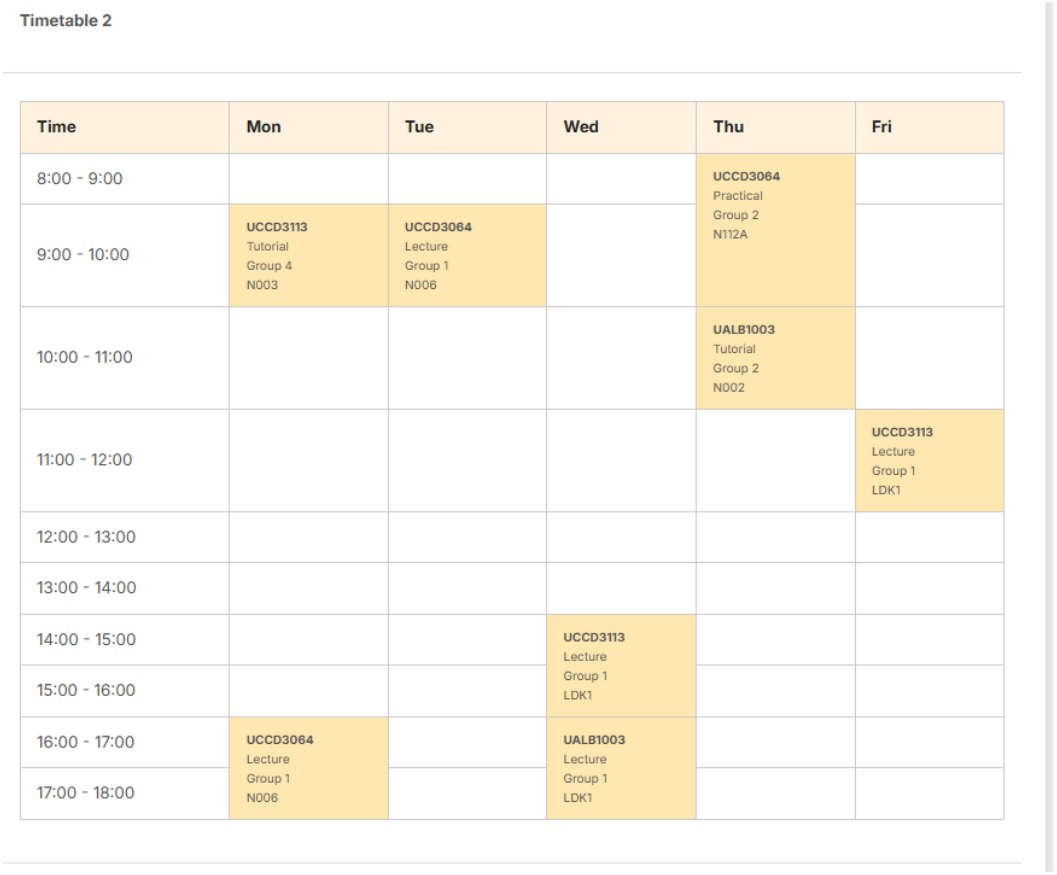


Figure 6.2.3.2 Elaine's generated timetable from Smart Student Timetable Planner

6.2.4 Testing Results

The results showed that the system-generated timetable was similar to the real-world timetable. All courses were correctly placed at the expected times, with no missing or extra sessions detected. The lectures, tutorials, and practical sessions matched perfectly in terms of course codes, session types, allocated days, and time durations. This demonstrated the system's ability to reproduce accurate timetables that are consistent with official schedules.

Table 6.2.4.1 Wong Xin Tong's Comparison Timetable's Results

Course Code	**Session Type**	**Real-World Timetable (Day/Time)**	**Generated Timetable (Day/Time)**	**Match**
UCCD3113	Lecture	Thu 14:00-16:00	Thu 14:00-16:00	✓
UCCD3113	Lecture	Fri 11:00-12:00	Fri 11:00-12:00	✓

UCCD3113	Tutorial	Mon 9:00-10:00	Mon 9:00-10:00	✓
UCCD3074	Lecture	Tue 8:00-10:00	Tue 8:00-10:00	✓
UCCD3074	Lecture	Wed 12:00-13:00	Wed 12:00-13:00	✓
UCCD3074	Practical	Mon 12:00-14:00	Mon 12:00-14:00	✓
UALJ2013	Lecture	Mon 10:00-12:00	Mon 10:00-12:00	✓
UALJ2013	Tutorial	Mon 15:00-16:00	Mon 15:00-16:00	✓

Table 6.2.4.2 Elaine's Comparison Timetable's Results

Course Code	Session Type	Real-World Timetable (Day/Time)	Generated Timetable (Day/Time)	Match
UCCD3113	Lecture	Thu 14:00-16:00	Thu 14:00-16:00	✓
UCCD3113	Lecture	Fri 11:00-12:00	Fri 11:00-12:00	✓
UCCD3113	Tutorial	Mon 9:00-10:00	Mon 9:00-10:00	✓
UCCD3064	Lecture	Mon 16:00-18:00	Mon 16:00-18:00	✓
UCCD3064	Lecture	Tue 8:00-10:00	Tue 8:00-10:00	✓
UCCD3064	Practical	Thu 8:00-10:00	Thu 8:00-10:00	✓
UALB1003	Lecture	Wed 16:00-18:00	Wed 16:00-18:00	✓
UALB1003	Tutorial	Thu 10:00-11:00	Thu 10:00-11:00	✓

6.2.5 Summary

In summary, the system testing confirmed that the Smart Student Timetable Planner accurately generates timetables that are identical to the real-world schedules. The comparison revealed no mismatches, ensuring that the system's scheduling logic, session mapping, and conflict

handling are working correctly. This outcome validates the system's capability to deliver reliable and precise timetables, demonstrating that it meets the functional requirements and can be trusted as a dependable scheduling tool for students.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

The Smart Student Timetable Planner has successfully addressed the challenges students face in planning their academic schedules. Using Genetic Algorithm, the system ensures that students are provided with conflict-free timetables that align with their personal preferences and academic requirements. This not only reduces the time and effort required for manual scheduling but also minimizes the risk of errors such as overlapping classes or missed sessions.

In addition to conflict resolution, the system supports both manual and automated scheduling, giving students the flexibility to customize their timetables according to individual needs. The collaborative module further enhances the planning experience by enabling real-time coordination among students, which is particularly useful for group projects or shared study plans. These features demonstrate the project's effectiveness in combining technical innovation with practical usability.

The project also contributes to academic management by incorporating features such as timetable history, comparison between auto-generated and manuals schedules, and export functions. These modules provide long-term benefits by ensuring students can keep records of past schedules and evaluate alternative planning strategies. The inclusion of administrative tools, such as uploading course data and responding to feedback, ensures the system remains relevant and sustainable within a university environment.

Overall, the system meets its primary objectives and provides a strong foundation for future enhancements. It proves that integration optimization algorithms with user-friendly interfaces can significantly improve academic planning, reducing stress for students while also supporting institutional efficiency.

7.2 Recommendations

Although the Smart Student Timetable Planner has achieved its goals, there are several areas where improvements can be made. One key recommendation is to enhance the Genetic Algorithm to handle larger datasets more efficiently. As student intakes and course offerings expand, optimizing the algorithm will ensure that timetable generation remains fast and accurate without compromising system performance.

Another area for improvement lies in collaborative scheduling. While the current module enables students to work together in real time, it could be extended with intelligent conflict-resolution suggestions. For instance, the system could recommend optimal adjustment when two or more students encounter scheduling clashes, thereby simplifying decision-making during group planning.

Integration with official university systems is also highly recommended. By linking the planner directly with the student portal and course registration database, data consistency would be improved, and administrative workloads would be reduced. This would create a seamless experience for students, eliminating the need for duplicate data entry and ensuring that timetables remain aligned with official course offerings.

Furthermore, expanding the platform to mobile applications would greatly improve accessibility. Students could check, update, and share their timetable on-the-go, making the system a more convenient part of their daily academic routines. Regular updates driven by student feedback should also be incorporated to ensure the system evolves alongside user expectations and technological trends.

By implementing these recommendations, the Smart Student Timetable Planner can transition from a prototype-level project into a fully integrated academic scheduling ecosystem. This evolution will not only maximize its value to students but also contribute meaningfully to efficiency of the institution’s academic management processes.

REFERENCES

[1] Mohd Asyraf Ruslaan, Zalmitah Zakaria, "Univeristy Course Timetabling System For Part-Time Students," Research Gate, September 2019. [Online]. Available: https://www.researchgate.net/publication/337601679_University_Course_Timetabling_System_For_Part-Time_Students. [Accessed 19 July 2024].

[2] T. Müller, "UniTime : Course Timetabling & Management," [Online]. Available: https://www.unitime.org/uct_courses.php#:~:text=The%20primary%20objective%20behind%20course,with%20the%20course%20of%20interest.. [Accessed 19 July 2024].

[3] B. Sunuami, René Arnulfo García-Hernández, and Yulia Ledeneva, "Personal Course Timetabling for University Students based on Genetic Algorithm," International Journal of Combinatorial Optimization Problems and Informatics., vol. 12, no. 3, pp. 32–49, Sep. 2021, doi: https://doi.org/10.61467/2007.1558.2021.v12i3.237. [Accessed 1 September 2025].

[4] J. Mohd and A. Fadzil, "Web based personalized university timetable for UiTM students using genetic algorithm / Mohd Radhi Fauzan Jamli and Ahmad Firdaus Ahmad Fadzil - UiTM Institutional Repository," Uitm.edu.my, Oct. 2024, doi: https://ir.uitm.edu.my/id/eprint/106030/1/106030.pdf. [Accessed 1 September 2025].

[5] D. Cimr and J. Hynek, "Heuristic Algorithm for a Personalised Student Timetable," *Lecture Notes in Computer Science*, pp. 79–88, 2018, doi: https://doi.org/10.1007/978-3-319-98446-9_8. [Accessed 1 September 2025].

[6] J. H. Wong, "TTAP-UTAR : Timetable Arranging Problem - UTAR," GitHub, 30 August 2017. [Online]. Available: https://github.com/wongjiahau/TTAP-UTAR. [Accessed 06 August 2024].

[7] "TimeEdit.net,2024," [Online]. Available: https://www.timeedit.net/ . [Accessed 04 August 2024].

[8] T. Kissflow, "RAD Methodology | Rapid Application Development Phases," Kissflow, 05 April 2024. [Online]. Available: https://kissflow.com/application-development/rad/rapid-application-development-methodology-essentials/. [Accessed 06 August 2024].

[9] Hayat Alghamdi, Tahani Alsubait, Hosam Alhakami, Abullah Baz, "A Review of Optimization Algorithms for University Timetable Scheduling," December 2020. [Online]. Available: https://etasr.com/index.php/ETASR/article/view/3832/2387. [Accessed 05 August 2024].

POSTER

SMART STUDENT TIMETABLE PLANNER

Timetable Scheduling is a critical process for students to ensure they can enroll in their required courses without conflicts. This project introduces a Smart Student Timetable Planner that automates the scheduling process, helping students generate optimized, clash-free timetables quickly and efficiently.

OBJECTIVES

- Develop a smart tool to streamline course selection and automatically resolve scheduling conflicts, offering dynamic schedule adjustments and intuitive visual timetables.
- Facilitate real-time collaboration and communication, enabling synchronized timetable planning with instant updates and notifications for all users.

MOTIVATION

Improve academic planning by addresing the challenges of course selection and lack of real-time collaboration.

PROBLEMS

- Difficulty in Course Selection and Scheduling
- Lack of Real-Time Collaboration in Schedule Planning

METHODS

- RAD Methodology
  - Requirements Planning
  - User Design
  - Constructive and Feedback
  - Finalise and Implementation
- Node.js, HTML, CSS, JavaScript and Python for main development.
- Visual Studio Code as IDE to develop the website

FUNCTIONS

- Login
- Manual Scheduling
- Auto Scheduling
- Timetable Comparison
- Timetable Export
- Real-Time Collaboration
- View Timetable History
- Administration Modules

CONCLUSION

The Smart Student Timetable Planner successfully streamlines course selection, resolves scheduling conflicts, and enhances real-time collaboration among students, advisors, and faculty. By offering dynamic scheduling tools, personalized recommendations, and a user-friendly interface, the system improves academic planning efficiency and reduces stress, creating a more seamless and effective scheduling experience.

Done By: Wong Xin Tong
Supervised By: Dr. Tan Joi San