FINAL YEAR PROJECT REPORT
BS (COMPUTER SCIENCE)

AUTOMATIC COURSE SCHEDULER

SUBMITTED BY:

ABDUL MAJID	51347
MUHAMMAD GHULAM HAMZA	51228
TAHA MIR	50560

COORDINATOR:
DR. AARIJ MAHMOOD HUSSAAN

FACULTY OF ENGINEERING, SCIENCE AND TECHNOLOGY
IQRA UNIVERSITY, KARACHI
MARCH 2024

FACULTY OF ENGINEERING, SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

FINAL YEAR PROJECT REPORT

BACHELOR OF COMPUTER SCIENCE

ABDUL MAJID 51347

MUHAMMAD GHULAM HAMZA 51228

TAHA MIR 50560

PROJECT:

AUTOMATIC COURSE SCHEDULER

SUPERVISOR:
DR. AARIJ MAHMOOD HUSSAAN

MARCH 2024:

We have approved this manuscript for submission and presentation as fulfillment of Bachelor of Computer Science.

_____

Project Coordinator:

DR. AARIJ MAHMOOD HUSSAAN

Date: _____

ABSTRACT

In today's rapidly evolving digital landscape, the intersection of technology and education holds immense promise for shaping the future of learning. Our project endeavors to harness this potential by developing a sophisticated automated course scheduling system using Python Django framework and genetic algorithm methodology. By seamlessly integrating course data, instructor preferences, and scheduling constraints, our system aims to streamline the arduous task of course scheduling in educational institutions.

The scheduler aims to efficiently allocate courses and teachers while resolving scheduling conflicts to generate optimized schedules.

The system comprises a backend implemented in Python Django, where courses, teachers, and constraints are defined, and a genetic algorithm module is employed to generate schedules. A frontend interface, built with HTML/CSS, enables users to interact with the system, input data, and view generated schedules.

Through the utilization of genetic algorithms, the scheduler optimizes schedules by iteratively evolving and selecting solutions that minimize conflicts and meet specified constraints. This approach enables the scheduler to adapt and improve its solutions over time, effectively addressing the complexities and dynamic nature of course scheduling.

The project contributes to the field of educational scheduling by providing a flexible and efficient solution that automates the scheduling process, reduces manual effort, and optimizes resource utilization. Future enhancements may include the integration of additional optimization techniques and the development of advanced scheduling features to further improve the efficiency and effectiveness of the system.

DECLARATION

The members of this project hereby declare that the work presented in this project, titled “Automatic Course Scheduler” builds upon existing methodologies and techniques within the field of Computer Science. While the core concepts and algorithms used in this project draw from established practices, significant modifications and enhancements have been made to address specific challenges and requirements.

One notable contribution of this project is the implementation of an error-handling function that enables users to detect and resolve conflicts in the generated schedule. This feature not only enhances the usability of the system but also adds a layer of transparency and reliability to the sc

We affirm that this project has not been previously submitted for any academic purpose. Proper acknowledgment has been given to any sources or individuals that have contributed to the development of this project, and all relevant citations and references have been included.

We acknowledge the importance of academic integrity and ethical conduct in scholarly research. Therefore, I attest that the work presented in this project represents our insights, and contributions to the field, while also building upon the collective knowledge and expertise of the academic community.

_____

ABDUL MAJID [51347]

_____

MUHAMMAD [51228]

_____

TAHA AMMAR MIR [50560]

Acknowledgments

We would like to express our sincere gratitude to all those who have contributed to the completion of this project.

First and foremost, we extend my heartfelt appreciation to our coordinator, **Dr. Aarij Mahmood Hussaan**, for their invaluable guidance, support, and encouragement throughout this project. Their expertise, feedback, and constructive criticism have been instrumental in shaping the direction and quality of this work.

We are also indebted to the faculty and staff of Iqra University, whose dedication to academic excellence has provided me with a conducive learning environment and access to resources necessary for conducting research.

We extend our thanks to **Mrs. Saira Khurram**, **Dr. Abdul Ahad Abro**, and **Sir Israr Ali**, who generously shared their knowledge, expertise, and insights during this project. Their input and feedback have been invaluable in overcoming challenges and refining the methodologies employed.

Furthermore, we are grateful to my peers and colleagues for their camaraderie, collaboration, and moral support during the ups and downs of this endeavor. Their encouragement and solidarity have been a source of determination to see this project through to completion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER – 1

1.0 Introduction

The scheduling of courses and teachers is a complex and time-consuming task faced by educational institutions worldwide. Manual scheduling processes often lead to inefficiencies, scheduling conflicts, and suboptimal resource utilization. To address these challenges, the present project introduces an automated course scheduler utilizing the Python Django framework and genetic algorithm methodology.

The primary objective of this project is to develop a system that can efficiently allocate courses to teachers and generate optimized schedules while mitigating scheduling conflicts. By automating the scheduling process, educational institutions can streamline operations, reduce administrative burden, and improve overall efficiency.

The utilization of genetic algorithms offers a robust approach to solving the course scheduling problem. Genetic algorithms simulate the process of selection, mutation and crossover to iteratively generate and refine solutions. This approach enables the scheduler to adapt to changing constraints and optimize schedules over time, leading to improved resource utilization and reduced conflicts.

The system consists of a backend implemented in Python Django, where courses, teachers, and scheduling constraints are defined. A genetic algorithm module is employed to generate schedules based on the input data and constraints. Additionally, a frontend interface built with HTML/CSS provides users with a user-friendly platform to input data, interact with the system, and visualize generated schedules.

One notable feature of this automated course scheduler is the inclusion of an error-handling function that enables users to detect and resolve conflicts in the generated schedule. This feature enhances the usability and reliability of the system by providing transparency and facilitating error resolution.

Through the development and implementation of this automated course scheduler, this project aims to contribute to the field of educational scheduling by providing a flexible, efficient, and scalable approach. By automating the scheduling process and leveraging genetic algorithm methodology, educational institutions can optimize resource allocation, improve scheduling efficiency, and fasten overall academic operations.

1.1 Problem Statement

In today's educational institutions, scheduling courses and teachers is a time-consuming and error-prone task. Manual scheduling processes often lead to conflicts, inefficient resource utilization, and administrative burden. This poses a significant challenge for educational institutions striving to optimize their operations and provide a seamless learning experience for students.

The problem we aim to address with our project is the inefficiency and complexity of course scheduling in educational institutions. Our goal is to develop an automated course scheduling system that can efficiently allocate courses to teachers, assign classrooms, and generate optimized schedules while minimizing conflicts and maximizing resource utilization. By automating the scheduling process, we aim to streamline operations, reduce administrative burden, and enhance overall efficiency in educational institutions.

1.2 Genetic Algorithm

A genetic algorithm is a heuristic search method used to find solutions for optimization problems. It is based on the Darwinian theory of evolution. This technique involves the ultimate selection of the fittest (best timetable) from a randomly created population (chromosomes) of solutions for the timetabling problem where each individual (chromosome) represents a timetable. The optimality (perfection) of a chromosome is evaluated by a fitness function based on hard and soft constraints. Genetic algorithms begin by creating a random population of timetables followed by their evaluation according to defined criteria to select parents (timetables) for the next generation which is expected to produce better timetables by way of crossovers and mutations. The process is repeated until a satisfactory solution is reached.

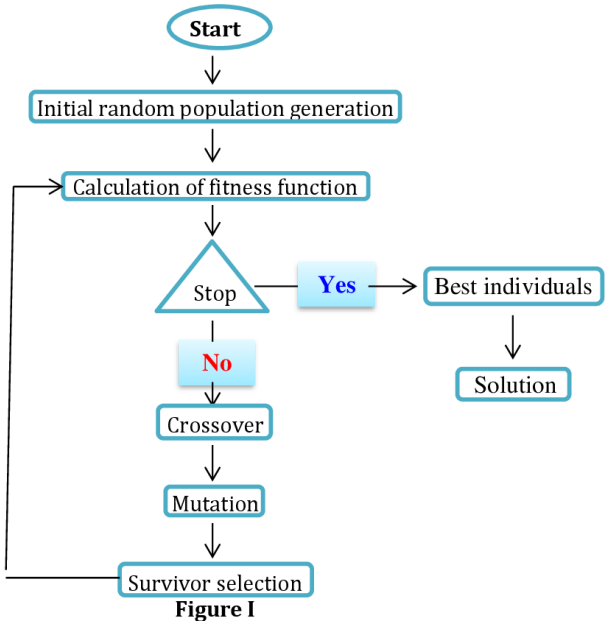


Figure 1 Genetic Algorithm Working

1.2.1 Initialization

Initializing a population of potential solutions. In this case, the solutions represent different course schedules.

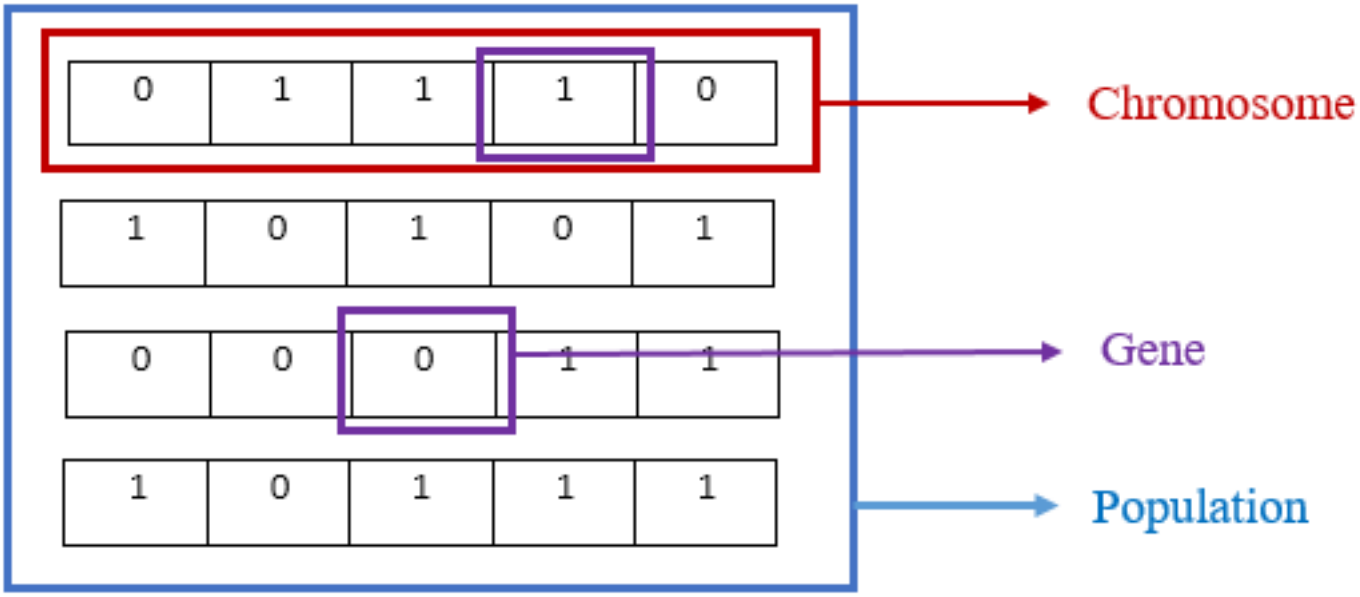


Figure 2 Genetic algorithm initialization

1.2.2 Fitness Function

Fitness Function: Evaluating the fitness of each solution in the population. The fitness function assesses how well each schedule satisfies certain criteria, such as minimizing conflicts or maximizing instructors preferences.

1.2.3 Selection:

Selecting individuals from the current population to serve as parents for producing offspring. Common selection methods include roulette wheel selection, tournament selection, or rank-based selection.

1.2.4 Crossover

Combining genetic material from two parent solutions to produce offspring solutions. This is typically done by selecting a crossover point and exchanging genetic information beyond that point between parents.

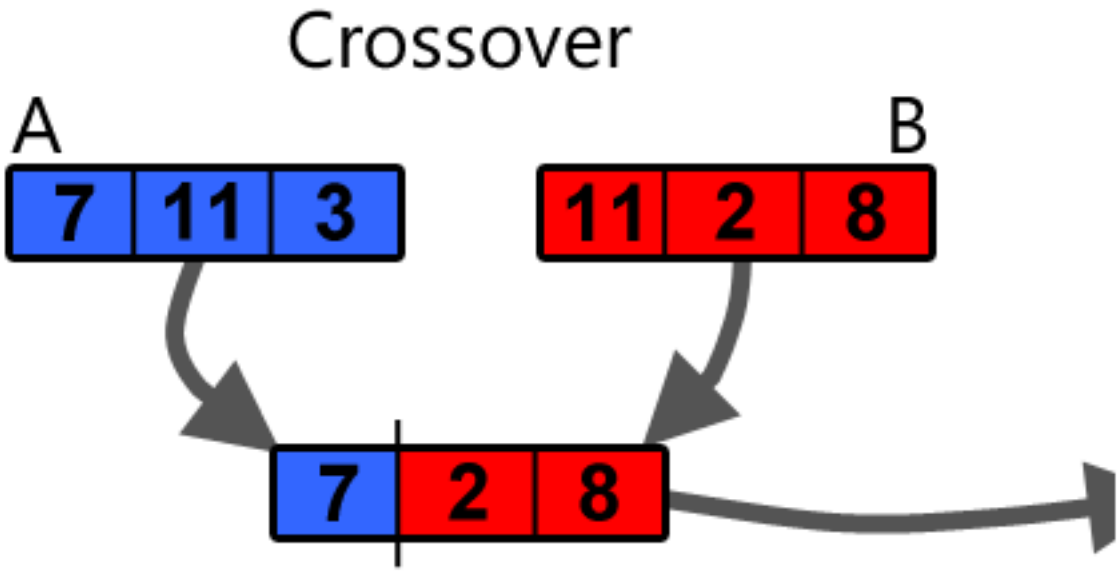


Figure 3 Crossover in Genetic Algorithm

1.2.5 Mutation

Introducing random changes in offspring solutions to maintain genetic diversity within the population. Mutation helps prevent premature convergence to suboptimal solutions.

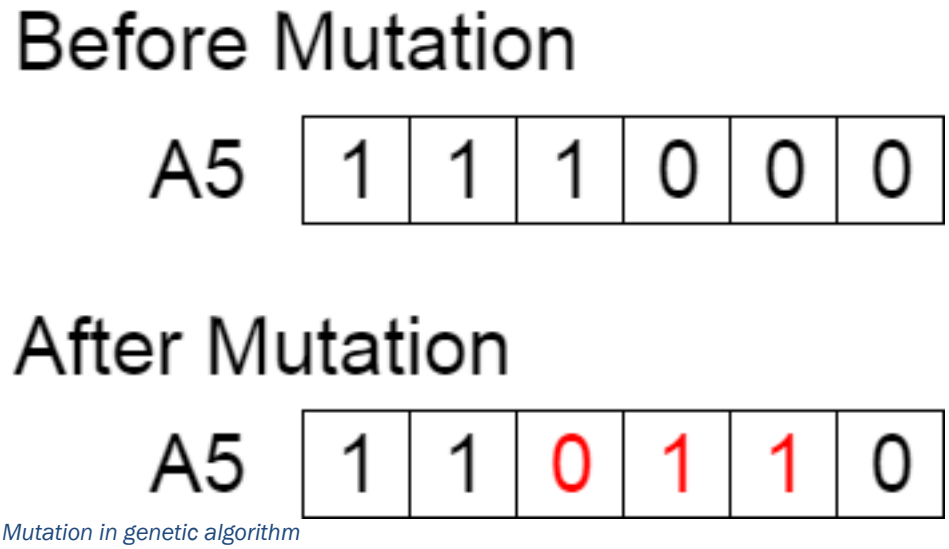


Figure 4 Mutation in genetic algorithm

1.2.6 Replacement

Selecting individuals from the current population and the newly generated offspring population to form the next generation. This process may involve elitism, where the best individuals are preserved in each generation.

Figure 5 Selection in genetic algorithm

1.2.7 Termination Condition

Determining when to stop the algorithm. This could be after a certain number of generations or when a satisfactory solution is found.

1.3 Motivation

The motivation behind this thesis stems from the recognition of the significant challenges and inefficiencies inherent in manual course scheduling processes within educational institutions. As students, educators, and administrators alike have experienced firsthand, the manual scheduling of courses and teachers often leads to scheduling conflicts, suboptimal resource allocation, and increased administrative burden.

By automating the course scheduling process, we aim to address these challenges and improve the overall efficiency and effectiveness of educational operations. The motivation for this thesis lies in the potential impact that an automated course scheduling system can have on streamlining operations, reducing administrative overhead, and enhancing the learning experience for students.

Through the development of an automated course scheduler, we seek to provide a solution that not only alleviates the burden of manual scheduling but also introduces optimization techniques to generate schedules that are efficient, conflict-free, and tailored to the specific constraints and requirements of each educational institution.

Furthermore, the motivation for this thesis also lies in the opportunity to contribute to the broader body of knowledge and research in the field of educational scheduling. By exploring and implementing novel methodologies such as genetic algorithms, we aim to advance the state-of-the-art in course scheduling and provide insights and best practices that can benefit educational institutions worldwide.

Ultimately, the motivation for this thesis is rooted in the belief that by harnessing the power of automation and optimization, we can revolutionize the way educational institutions manage their scheduling processes, leading to improved efficiency, enhanced resource utilization, and a better overall learning experience for students.

1.4 Objective

The objective of this thesis is to design, develop, and implement an automated course scheduling system for educational institutions. Overall, the objective of this thesis is to provide educational institutions with a robust, efficient, and scalable solution for automating the course scheduling process, thereby improving operational efficiency and enhancing the learning experience for students.

The specific objectives include:

1. Designing a user-friendly interface for inputting course, teacher, and scheduling constraint data.
2. Developing a backend system using the Python Django framework to manage course scheduling logic and algorithms.
3. Implementing a genetic algorithm module to generate optimized schedules based on input data and constraints.
4. Integrating error handling functions to detect and resolve conflicts in the generated schedules.
5. Evaluating the performance and effectiveness of the automated course scheduling system through rigorous testing and analysis.
6. Providing recommendations for further improvements and enhancements to the automated course scheduling system based on the evaluation results.

Research Objective:

The research objective of this thesis is to explore and implement novel methodologies, such as genetic algorithms, for automating course scheduling processes in educational institutions. By investigating the effectiveness of these methodologies and their applicability to real-world scheduling challenges, the research aims to advance the understanding of course scheduling optimization techniques and contribute to the body of knowledge in the field.

Academic Objective:

The academic objective of this thesis is to demonstrate the feasibility and effectiveness of automated course scheduling systems in improving operational efficiency and resource utilization within educational institutions. By developing a comprehensive understanding of the underlying principles and algorithms involved in course scheduling automation, the academic objective is to provide valuable insights and recommendations for future research and implementation efforts in the field.

Management Objective:

The management objective of this thesis is to provide educational institutions with a practical and actionable solution for streamlining course scheduling processes. By implementing an automated course scheduling system, the management objective is to reduce administrative burden, minimize scheduling conflicts, and optimize resource allocation, ultimately leading to improved operational efficiency and enhanced academic outcomes within the institution.

1.5 Challenges

1. Genetic Algorithm Complexity: One of the primary challenges of this project is the complexity associated with implementing genetic algorithms for course scheduling. Genetic algorithms are known for their computational intensity, especially when applied to optimization problems like course scheduling, which can be likened to a knapsack problem. The process of generating optimal schedules while considering various constraints and preferences adds another layer of complexity to the algorithm.

2. Data Accessibility: Obtaining accurate and reliable data for course scheduling poses a significant challenge. Educational institutions are often reluctant to share sensitive scheduling data with third-party applications due to privacy concerns and trust issues. This necessitates the development of strategies to collect and utilize data effectively while maintaining trust and compliance with data protection regulations.

3. Quota Limitations: Another challenge is the quota limitations imposed by cloud computing platforms or hosting services for uploading and processing large volumes of data. Since course scheduling involves processing substantial amounts of data, including course offerings, instructor availability, and classroom capacities, the project may encounter difficulties in managing quota arrangements for data uploading and processing, especially if the budget is limited.

4. Algorithm Optimization: Optimizing the genetic algorithm to efficiently handle course scheduling constraints and preferences is a significant challenge. Balancing the trade-offs between computational complexity and solution quality requires careful tuning of algorithm parameters and heuristics. Additionally, addressing edge cases and corner scenarios to ensure robustness and reliability of the scheduling algorithm adds to the challenge of algorithm optimization.

5. User Interface Design: Designing an intuitive and user-friendly interface for inputting scheduling preferences and viewing generated schedules is another challenge. The interface must accommodate diverse user needs and preferences while maintaining simplicity and clarity. Ensuring seamless integration between the backend scheduling engine and frontend user interface adds complexity to the interface design process.

Overall, overcoming these challenges requires a multidisciplinary approach, combining expertise in algorithm design, data management, user interface design, and project management. By addressing these challenges effectively, the project aims to deliver a reliable, efficient, and user-friendly automated course scheduling system that meets the needs of educational institutions and stakeholders.

1.6 Structure of Report

This marks the completion of chapter one of the Automatic Course Scheduler. This chapter highlighted the overall concept of the automatic course scheduler. The introduction section presents an in-depth overview of the concept behind the automated course scheduler. The problem statement identifies the specific challenges addressed by this application and outlines its potential benefits for educational institutions. The motivation section elucidates the driving factors behind

the development of the automated course scheduler, providing insight into the inspiration and rationale for its creation.

As we delve deeper into Chapter One, the objectives of the project become apparent. Here, readers will encounter essential details such as research objectives, academic objectives, and management objectives. Additionally, challenges associated with the development and utilization of the automated course scheduler are addressed. Lastly, the structure of the project report is outlined, providing a roadmap for the subsequent chapters.

1.6.1 Chapter 2: Technology Background

Chapter 2 will offer a comprehensive literature review encompassing the existing body of research relevant to automated course scheduling and the underlying technologies utilized in the project.

1.6.2 Chapter 3: Requirements & Methodology

Chapter 3 will provide an in-depth discussion of the fundamental system models, alongside the presentation of both functional and nonfunctional requirements essential for our project.

1.6.3 Chapter 4: Project Plan & Initial Design

This chapter will encompass detailed project designs aimed at facilitating developers' understanding of the project's implementation and establishing a clear pathway for system development.

1.6.4 Chapter 5: Project Design & Development

Chapter 5 stands as a pivotal chapter, outlining the actual design and implementation phases of the automated course scheduler. It delves into the various stages of design and development.

1.6.5 Chapter 6: Testing

In this chapter, we will construct test cases, including front-end (design testing) and back-end (source code) testing. Additionally, we will employ testing tools to conduct comprehensive testing and integrate the results into the report.

1.6.6 Chapter 7: Conclusion

Chapter 7 serves as the culmination of our work, where we will present our findings, including facts, figures, tables, and graphs.

We will discuss the encountered limitations and challenges and outline future work to be undertaken.

CHAPTER – 2

2. Technology Background

Chapter 2 of this thesis provides a comprehensive exploration of the technological landscape underpinning the automated course scheduling project. Drawing upon extensive literature review and research, this chapter offers insights into the various methodologies, algorithms, and technologies utilized in the development of the automated course scheduling system.

The technology background begins by examining the foundational concepts of course scheduling in educational institutions. This includes an overview of traditional manual scheduling processes, highlighting the inherent challenges and inefficiencies that necessitate the adoption of automated solutions.

Furthermore, the chapter delves into the theoretical underpinnings of genetic algorithms, a key methodology employed in the automated course scheduling project. Genetic algorithms simulate the process of natural selection to iteratively generate optimal solutions, making them particularly well-suited for complex optimization problems such as course scheduling.

2.1 Background of the technology:

The technology utilized in this project encompasses a modern stack of tools and frameworks tailored to address the complexities of automated course scheduling. Leveraging Python Django for backend development, HTML/CSS for frontend design, and genetic algorithms for optimization, this project integrates cutting-edge technologies to streamline the course scheduling process. These technologies offer robustness, scalability, and flexibility, allowing for efficient management of course assignments, instructor allocations, and scheduling constraints. Additionally, the adoption of cloud computing and database management systems ensures seamless data handling and accessibility, while user-centric design principles guarantee an intuitive and engaging user experience. Overall, the background of the technology used in this project reflects a commitment to innovation and efficiency in educational scheduling solutions.

2.1.1 Visual Studio

Visual Studio was chosen for this project due to its robust integrated development environment (IDE) and comprehensive toolset tailored for web development. With features such as IntelliSense for code completion, debugging capabilities, and seamless integration with version control systems like Git, Visual Studio enhances productivity and facilitates collaborative development. Moreover, its support for various programming languages and frameworks, including Python Django and HTML/CSS, aligns perfectly with the technological requirements of the automated course scheduling project. Overall, Visual Studio offers a seamless and efficient development experience, making it the preferred choice for this endeavor.

2.1.2 Python

Python was selected as the primary programming language for this project due to its versatility, readability, and extensive ecosystem of libraries and frameworks. As a high-level language, Python allows for rapid development and easy integration of complex functionalities, making it well-suited for building the backend logic of the automated course scheduling system. Additionally, Python's support for object-oriented and functional programming paradigms enables the creation of modular and scalable code, facilitating maintenance and future enhancements. Furthermore, Python's popularity within the data science and artificial intelligence communities aligns with the use of genetic algorithms for optimizing course schedules. Overall, Python's combination of simplicity and power makes it an ideal choice for implementing the core functionality of the automated course scheduling project.

2.1.3 Html / CSS

HTML and CSS were chosen for frontend development due to their simplicity, widespread support, and beginner-friendly nature. They allow for rapid prototyping, easy customization, and maintainability, essential for creating an intuitive and visually appealing user interface for the automated course scheduling system.

2.2 Literature Review

The scheduling of courses and teachers in educational institutions presents a multifaceted challenge that has garnered extensive attention in scholarly literature. The subsequent literature review offers a comprehensive examination of pertinent concepts, methodologies, and approaches pertaining to automated course scheduling. Traditional scheduling methods, often reliant on manual processes or basic software tools, are noted for their time-consuming nature, susceptibility to errors, and limited optimization capabilities, resulting in inefficiencies and suboptimal outcomes.

In response, optimization techniques such as genetic algorithms, simulated annealing, and constraint programming have emerged as viable solutions to automate the scheduling process. Among these, genetic algorithms stand out for their ability to generate high-quality schedules by emulating natural selection and evolutionary principles to iteratively refine solutions. Genetic algorithm-based scheduling approaches, frequently employed across various scheduling domains, including education, involve encoding solutions as chromosomes, applying genetic operators like selection, crossover, and mutation, and evaluating solutions based on fitness functions capturing scheduling objectives and constraints.

Illustrative case studies and real-world applications underscore the practical efficacy of automated course scheduling systems in diverse educational contexts, shedding light on the myriad challenges, requirements, and solutions encountered. Ultimately, the literature underscores the pivotal role of automated course scheduling in streamlining operations, alleviating administrative burdens, and elevating academic outcomes within educational institutions. Leveraging optimization methodologies like genetic algorithms, researchers and practitioners can forge effective scheduling systems adept at addressing the intricacies and constraints inherent in educational scheduling environments.

CHAPTER – 3

3.1 Introduction:

In this chapter we will discuss about how much work is done on the development of our project according to the project plan. This chapter will cover the in-detail process and objective of the project. As our app is built on windows that are being reused within the system itself the developers had to take a systematic approach for the app to work smoothly. Our project plan isstrategically planned with the Gantt chart and other organizational tools. Each activity has specific time period allotted according to the complexity of the task, which is why the days in work may vary.

We will also discuss in detail about the Functional, Non-Functional and Hardware requirements of our project. The functional requirement are taken in to full consideration as they are the necessary part in order to get the basic requirement by the project such as, camera detection and learning objective, while on the other hand, the non-functional requirement such as, settings and feedback are also thoroughly planned.

3.2 Project Plan:

Figure 6 Project plan

3.3 Functional Requirements:

1. **Input Information:** The system must allow input of course, teacher, and classroom details.
2. **Constraints:** Users should input constraints and preferences for each course and instructor.
3. **Schedule Generation:** The system should generate schedules for courses and instructors.
4. **Classroom Scheduling**: Ability to generate schedules for each classroom.
5. **Avoid Conflicts:** Ensure no instructor has more than one class simultaneously.
6. **Instructor Preferences:** Allow instructors to specify preferred courses.
7. **Multiple Sessions:** Each course can have multiple sessions, each with an assigned instructor.
8. **Export Functionality:** Ability to export schedules to formats like PDF or CSV.
9. **Conflict Recognition:** Recognize conflicts based on the error handling aspect.
10. **Course Listing:** Display available courses, times, and locations on the interface.
11. **Optimization:** Consider classroom availability for an optimized schedule.
12. **Instructor Preferences:** Accommodate instructor preferences and availability during schedule generation.

3.4 Non-Functional Requirements:

1. **Performance:** The scheduling algorithm must efficiently handle large data sets.
2. **Scalability**: The system should scale to accommodate a large volume of course, instructor, and classroom data.
3. **Security**: Ensure data and user information security.
4. **Reliability**: The system must be reliable and available at all times.
5. **Portability**: Run on various platforms and operating systems.
6. **Support**: Provide dedicated support for technical assistance.

Interoperability: Ability to integrate with existing institution systems or software.

3.5 Hardware Requirements:

The hardware requirements for the automatic course scheduling system are outlined below:

i. Server or Hosting Platform:

The system requires a server or hosting platform to deploy the backend components, such as the database and scheduling algorithms.

Minimum specifications include sufficient processing power, memory (RAM), and storage to accommodate database operations and algorithm computations.

ii. Network Connectivity:

Reliable internet connectivity is essential for accessing the system remotely and facilitating communication between users and the server.

High-speed internet connectivity is recommended to ensure efficient data transfer and system responsiveness.

iii. Client Devices:

Users will access the system through client devices such as desktop computers, laptops, tablets, or smartphones.

Client devices should meet basic requirements for web browsing and accessing web applications.

Compatibility with modern web browsers (e.g., Google Chrome, Mozilla Firefox, Safari) is necessary for optimal performance and user experience.

iv. Storage Devices (Optional):

Additional storage devices may be required for data backup and archival purposes.

Cloud storage solutions can be utilized for storing backups and ensuring data redundancy.

v. Peripheral Devices (Optional):

Peripheral devices such as printers or external monitors may be used for printing schedules or displaying system information, respectively.

vi. Backup and Redundancy Systems:

Implementing backup and redundancy systems is recommended to ensure data integrity and system availability.

vii. Scalability Considerations:

The hardware infrastructure should be scalable to accommodate potential increases in system usage, data volume, and user concurrency.

viii. Security Measures:

Hardware-based security measures, such as firewalls, intrusion detection systems, and encryption protocols, should be implemented to safeguard system assets and protect against unauthorized access or data breaches.

3.6 Summary:

Chapter 3 outlines the functional and non-functional requirements guiding the development of the automatic course scheduling system. It highlights essential capabilities such as inputting course information, handling conflicts, and providing a user-friendly interface. The methodology section discusses Agile principles and iterative development cycles, emphasizing collaboration and adaptability. Overall, this chapter provides a framework for the systematic development of the system, ensuring its reliability and effectiveness in addressing scheduling challenges in educational institutions.

CHAPTER – 4

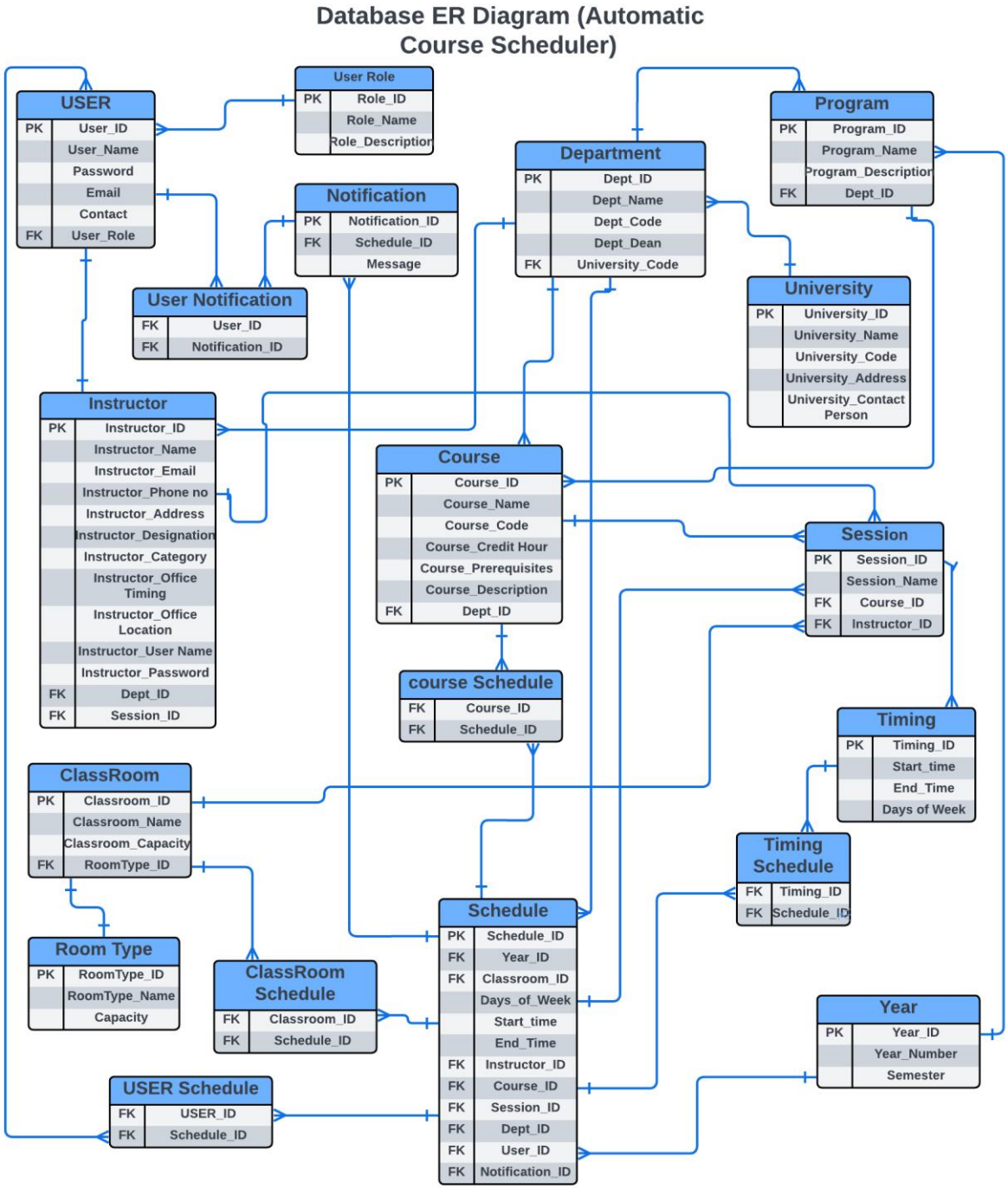
4.1 Introduction:

Chapter 4 delves into the design and specification of the automatic course scheduling system, detailing the data models.

The chapter describes the data models used in the system to represent course information, teacher details, classroom availability, and scheduling constraints. Entity-relationship diagrams (ERDs) or similar visual representations are provided to illustrate the relationships between different data entities and their attributes. The design rationale behind each data model is explained, highlighting considerations such as data integrity, normalization, and efficiency.

4.2 Entity Relationship Diagram:

The Entity Relationship diagram has been displayed below:



4.2 Summary

Chapter 4 focuses on the design and specification of the automatic course scheduling system. The chapter primarily presents the Entity-Relationship Diagram (ERD) depicting the relationships between various data entities such as courses, teachers, classrooms, and scheduling constraints. By showcasing the ERD, the chapter provides a visual representation of the underlying data structure and relationships within the system. This design element serves as a foundational framework for the subsequent phases of system development and implementation. The ERD's simplicity and clarity facilitate understanding and collaboration among project stakeholders, laying the groundwork for further refinement and enhancement of the automatic course scheduling system.

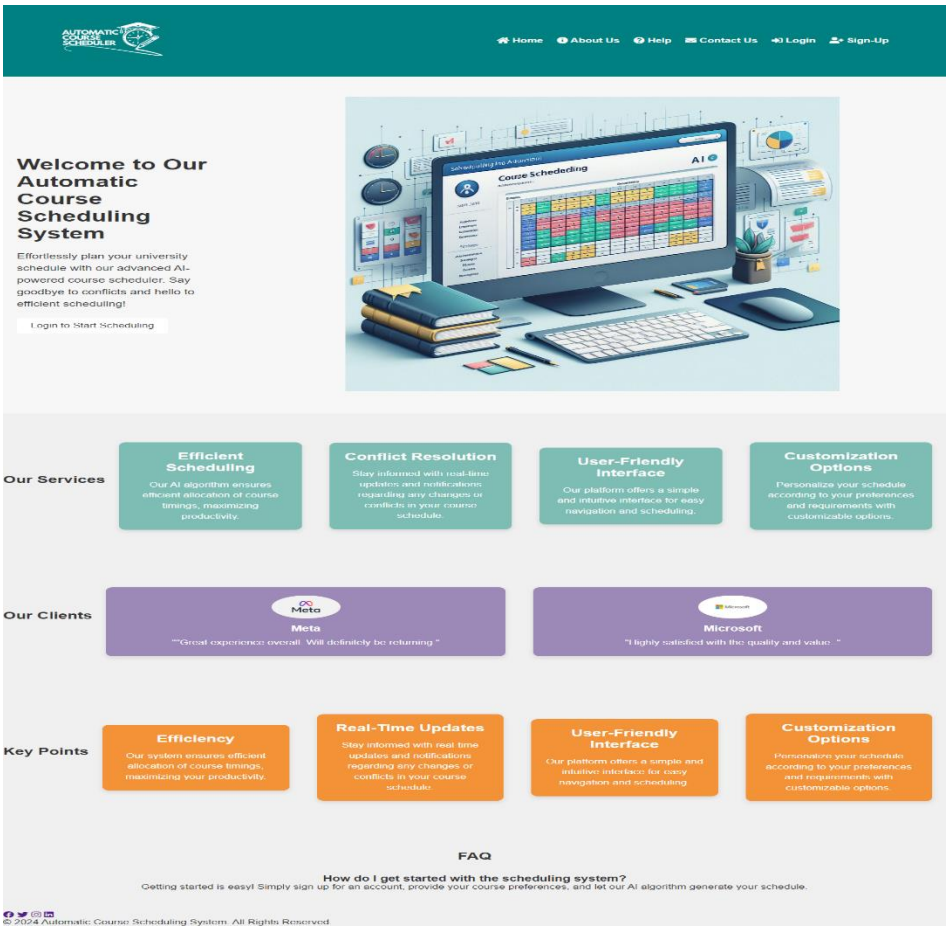
CHAPTER – 5

5.1 Introduction

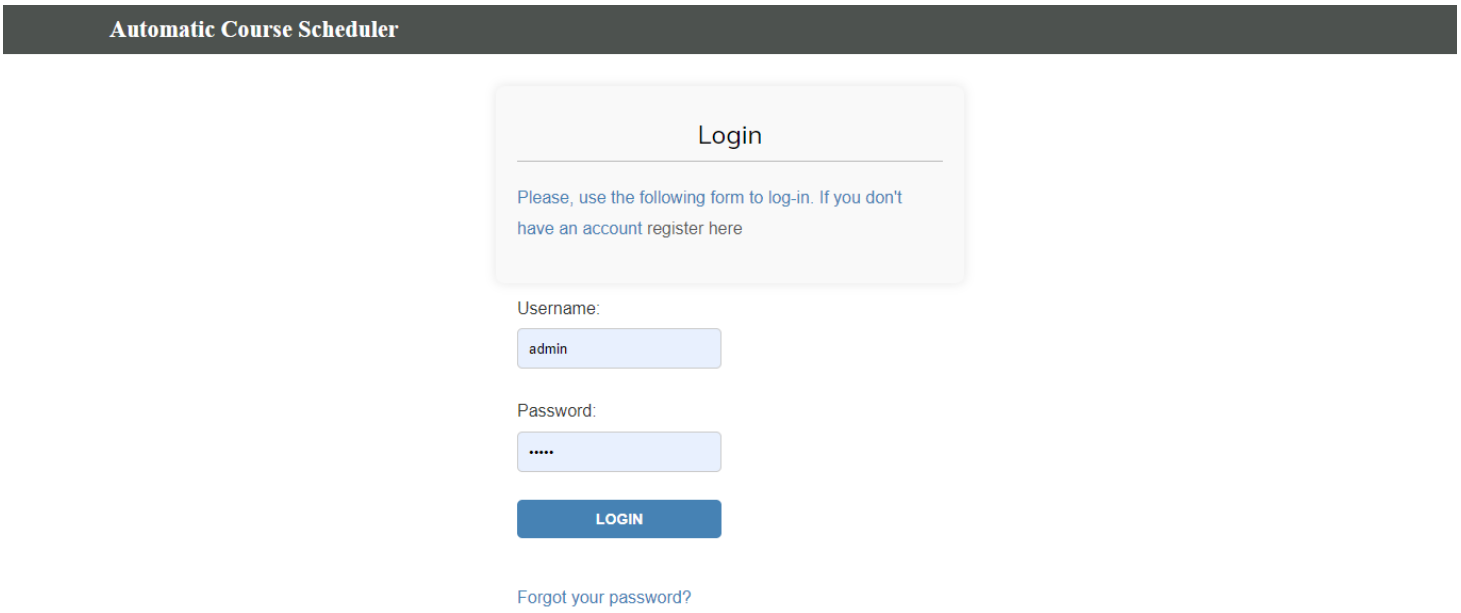
We've integrated several external libraries into our project to enhance its functionality. Additionally, we've included screenshots of the complete system to provide a better visual understanding of how everything works together. These screenshots offer a comprehensive view of our project's user interface and design, allowing for a clearer appreciation of its overall structure and functionality.

5.2 Screenshots

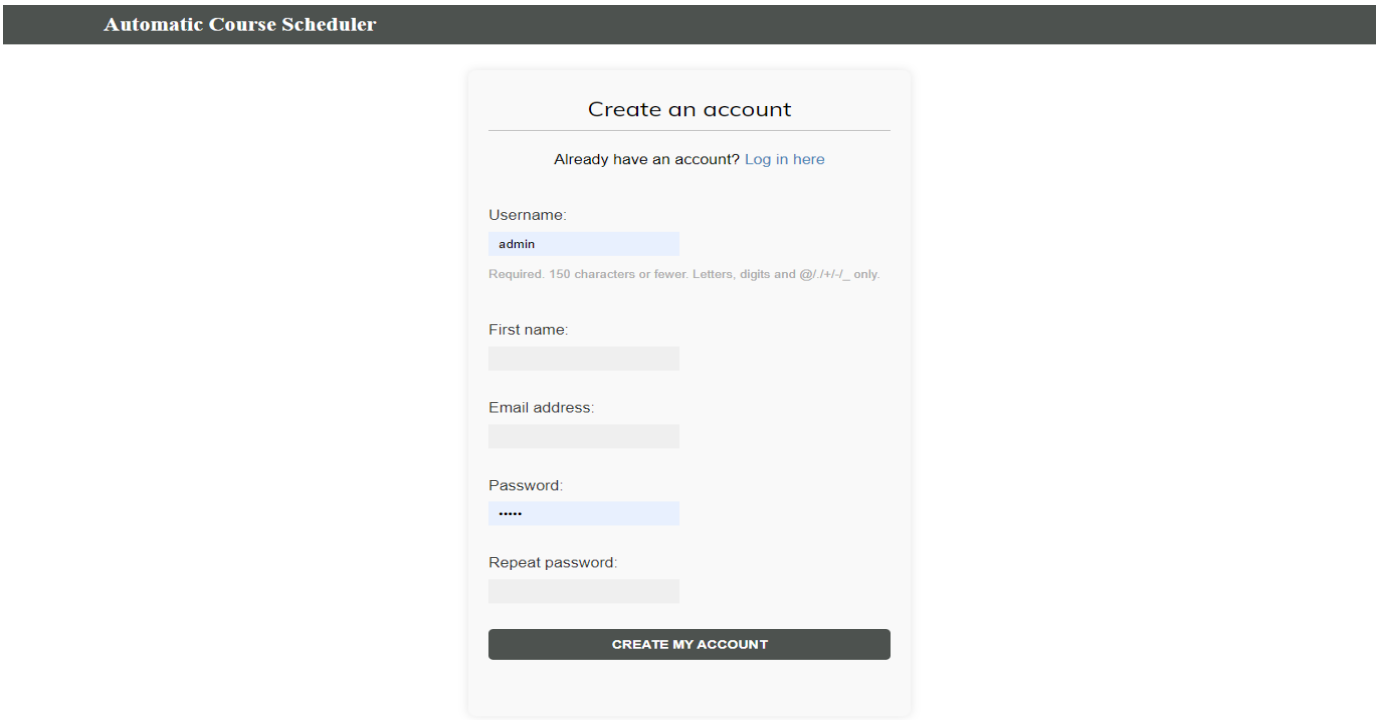
Home page:



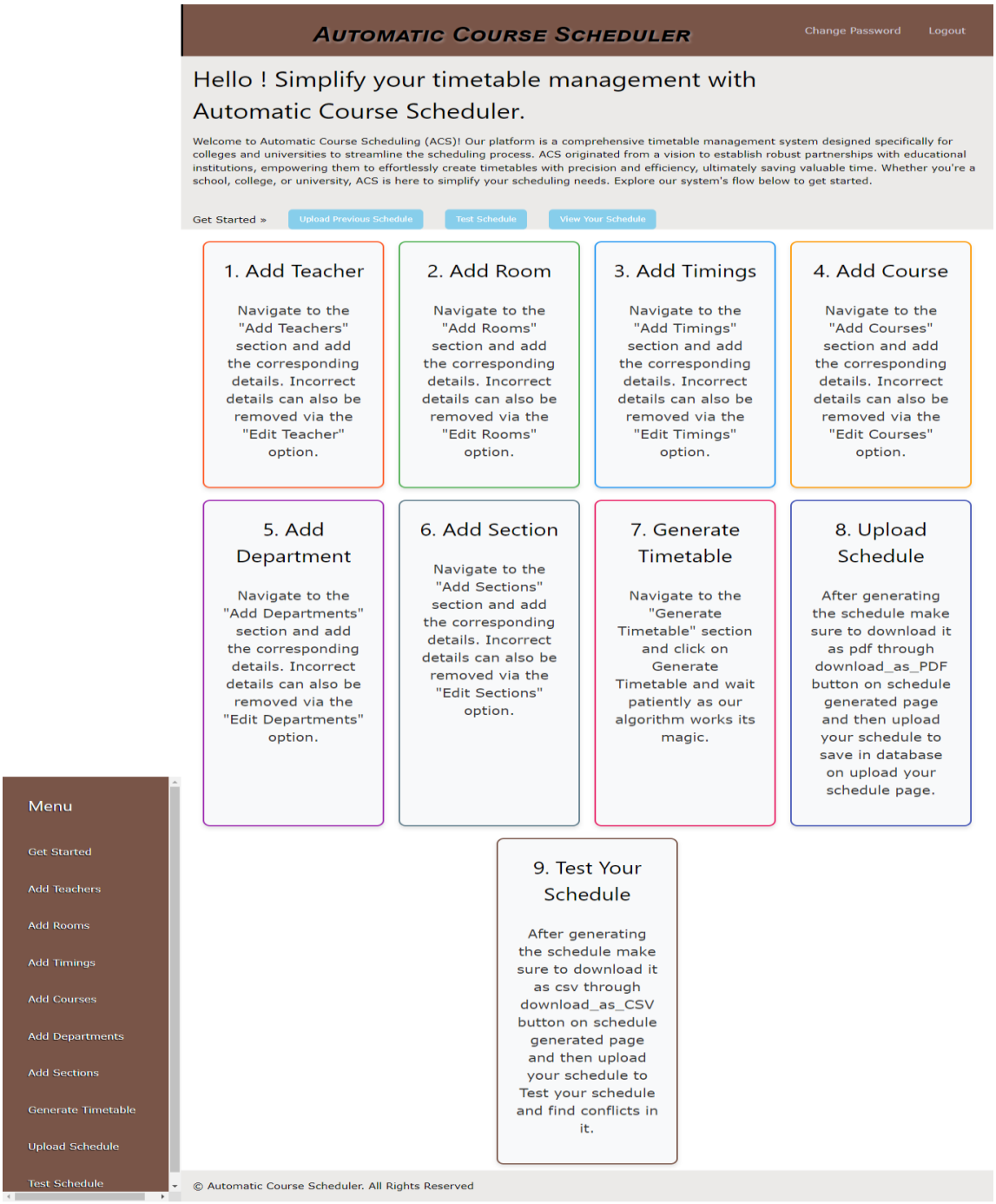
Login Page:



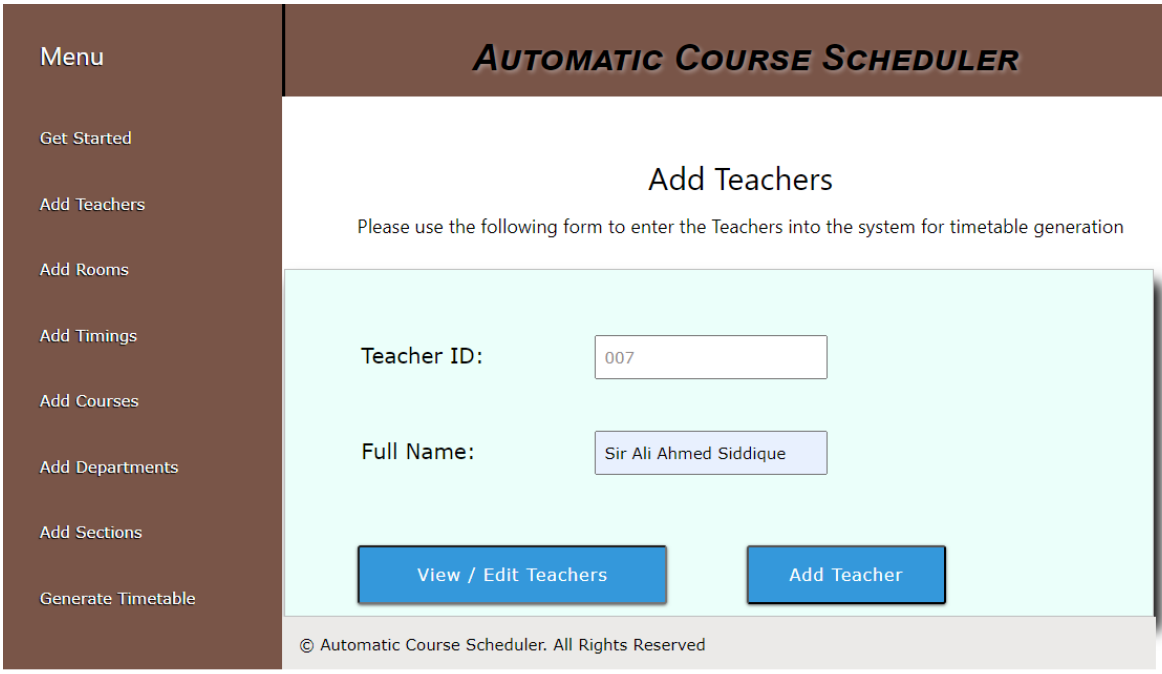
SignUP page:



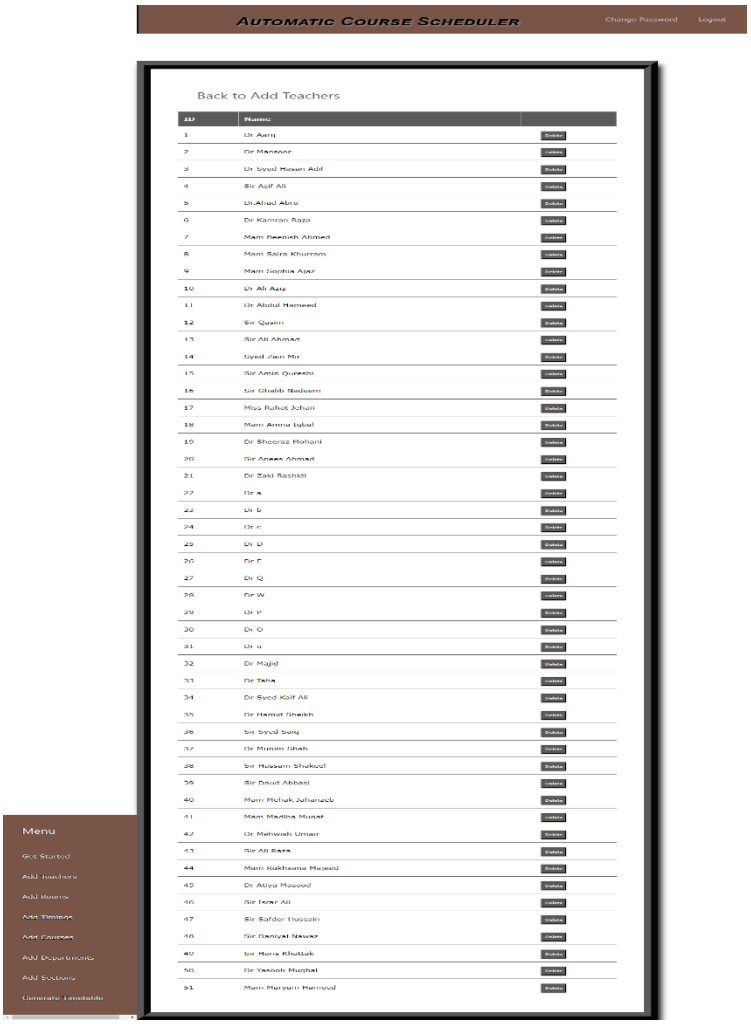
Admin Dashboard:



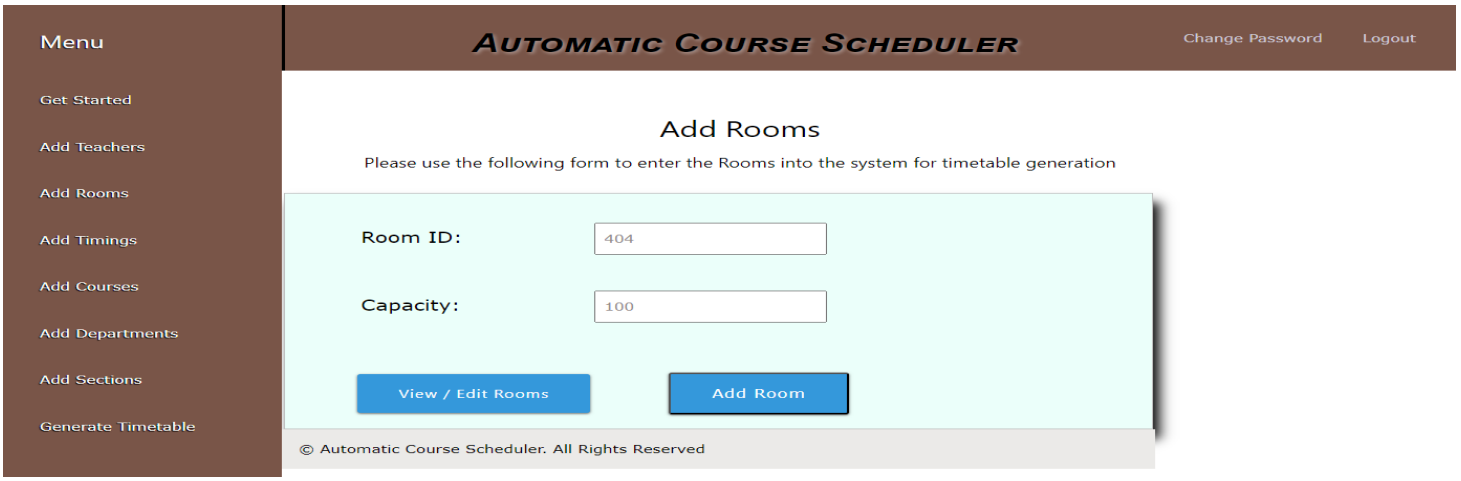
Add Instructor Page:


Menu
AUTOMATIC COURSE SCHEDULER
Change Password
Logout
Get Started
Add Teachers
Add Rooms
Add Timings
Add Courses
Add Departments
Add Sections
Generate Timetable
Add Teachers
Please use the following form to enter the Teachers into the system for timetable generation
Teacher ID:
007
Full Name:
Sir Ali Ahmed Siddique
View / Edit Teachers
Add Teacher
© Automatic Course Scheduler. All Rights Reserved

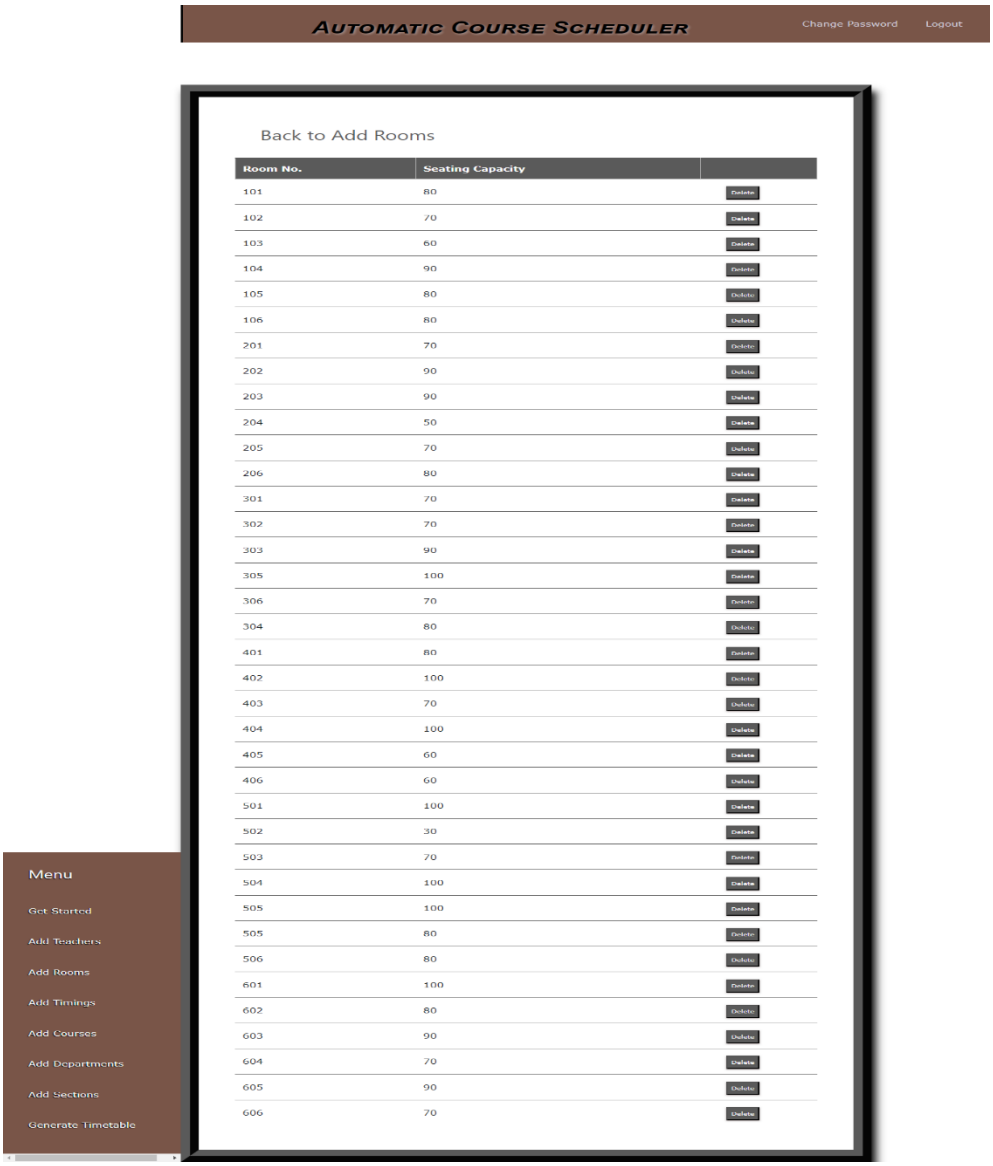
List of instructors:



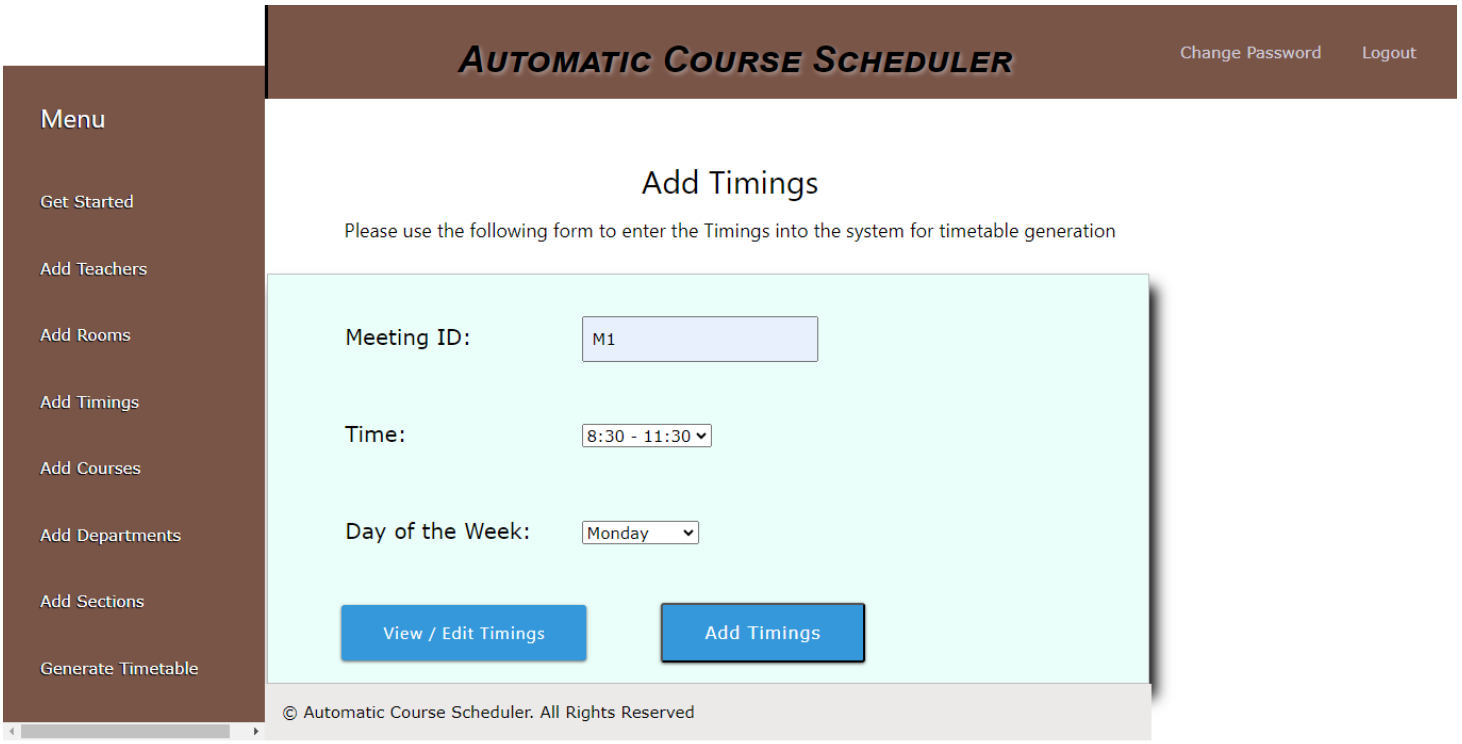
Add Room page:


Menu
AUTOMATIC COURSE SCHEDULER
Change Password
Logout
Get Started
Add Teachers
Add Rooms
Add Timings
Add Courses
Add Departments
Add Sections
Generate Timetable
Add Rooms
Please use the following form to enter the Rooms into the system for timetable generation
Room ID:
404
Capacity:
100
View / Edit Rooms
Add Room
© Automatic Course Scheduler. All Rights Reserved

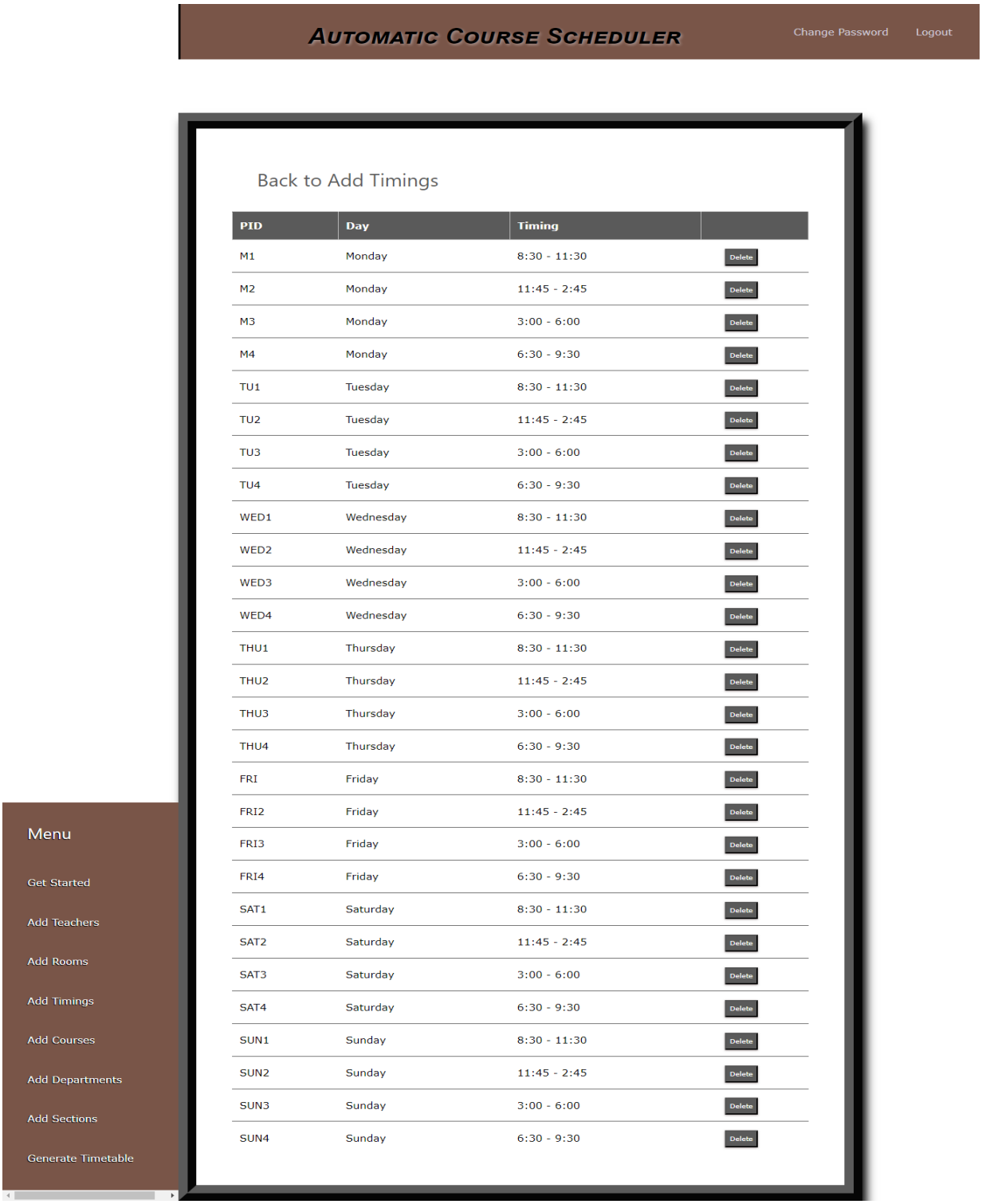
List of Classrooms:



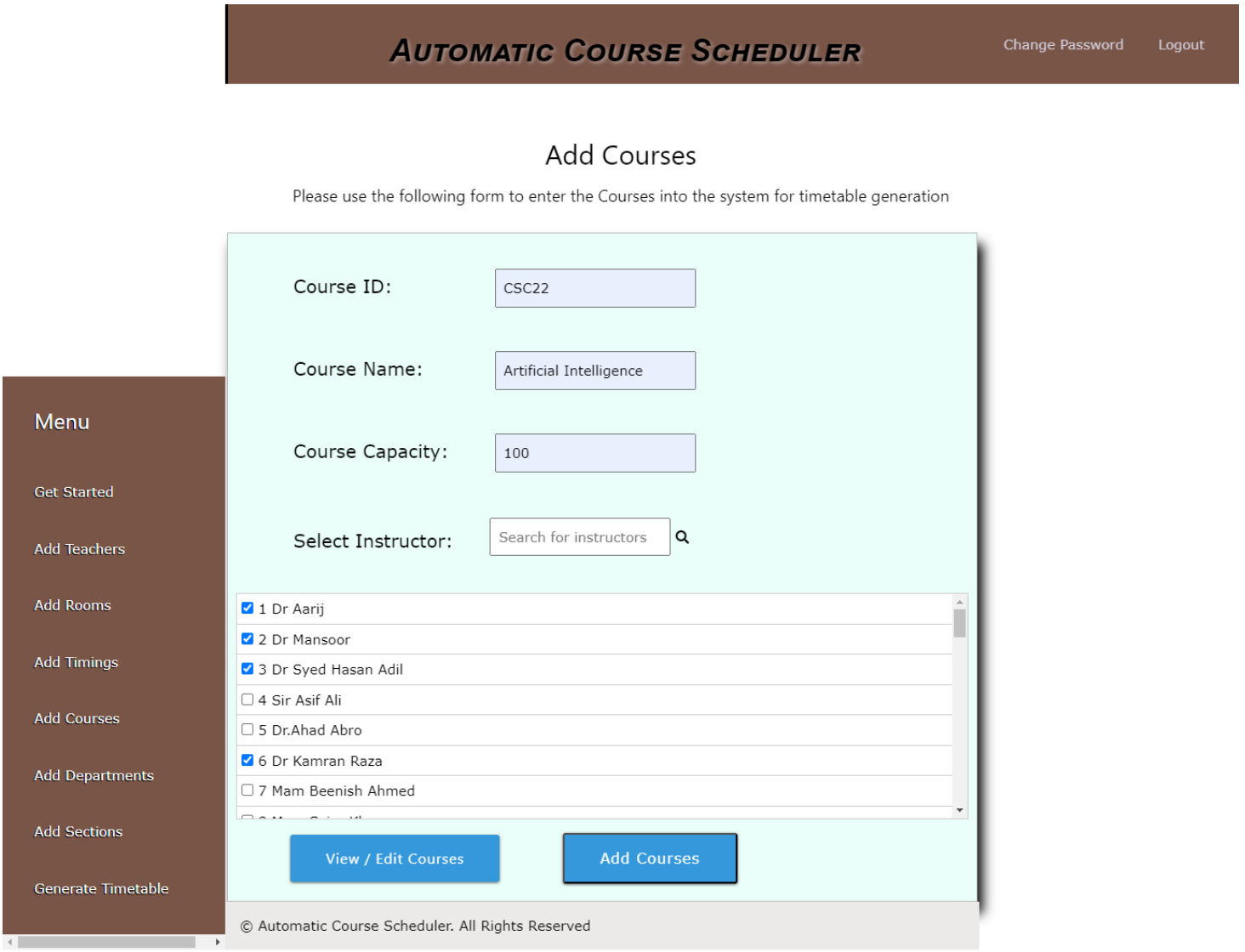
Add Meeting timing page:



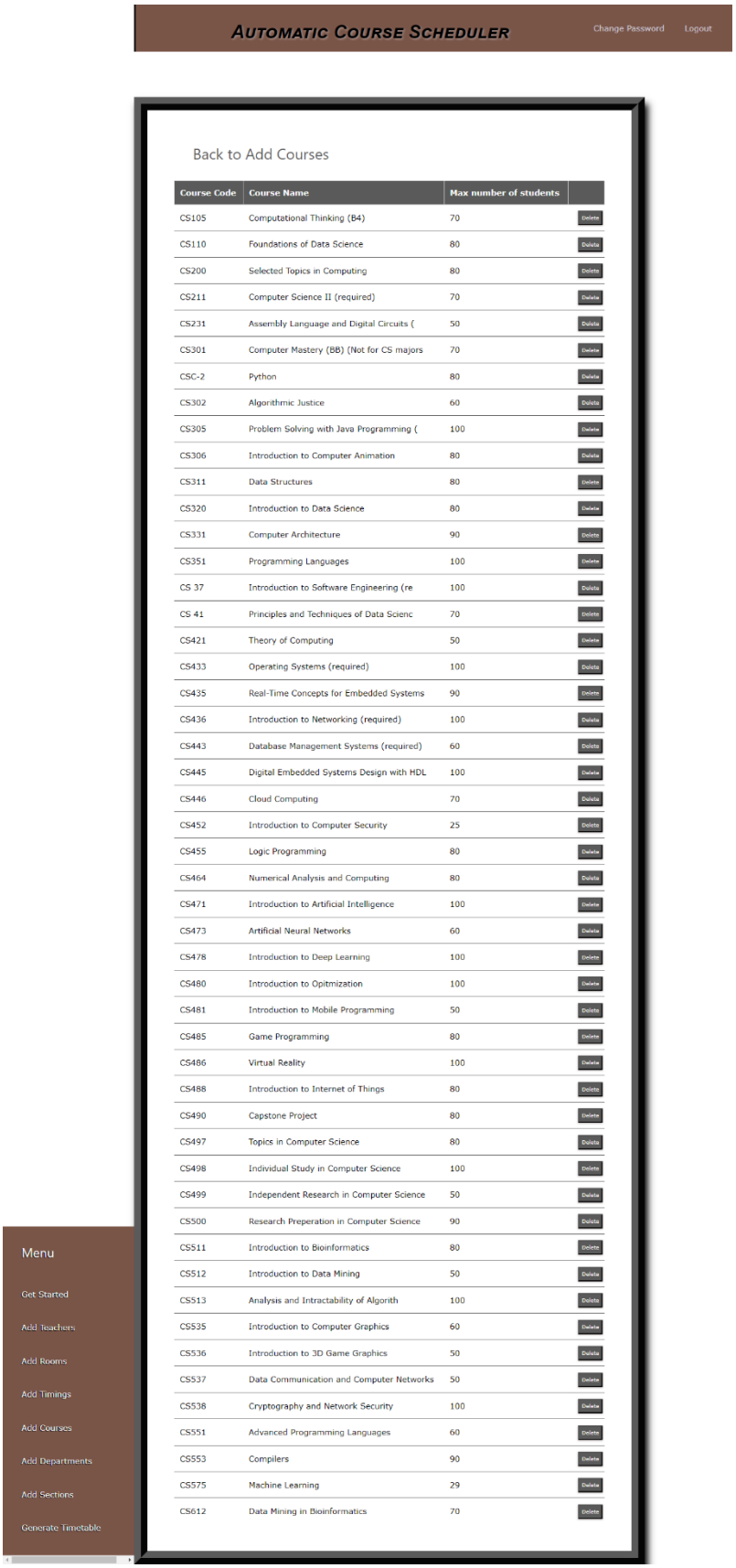
List of Meeting Timings:



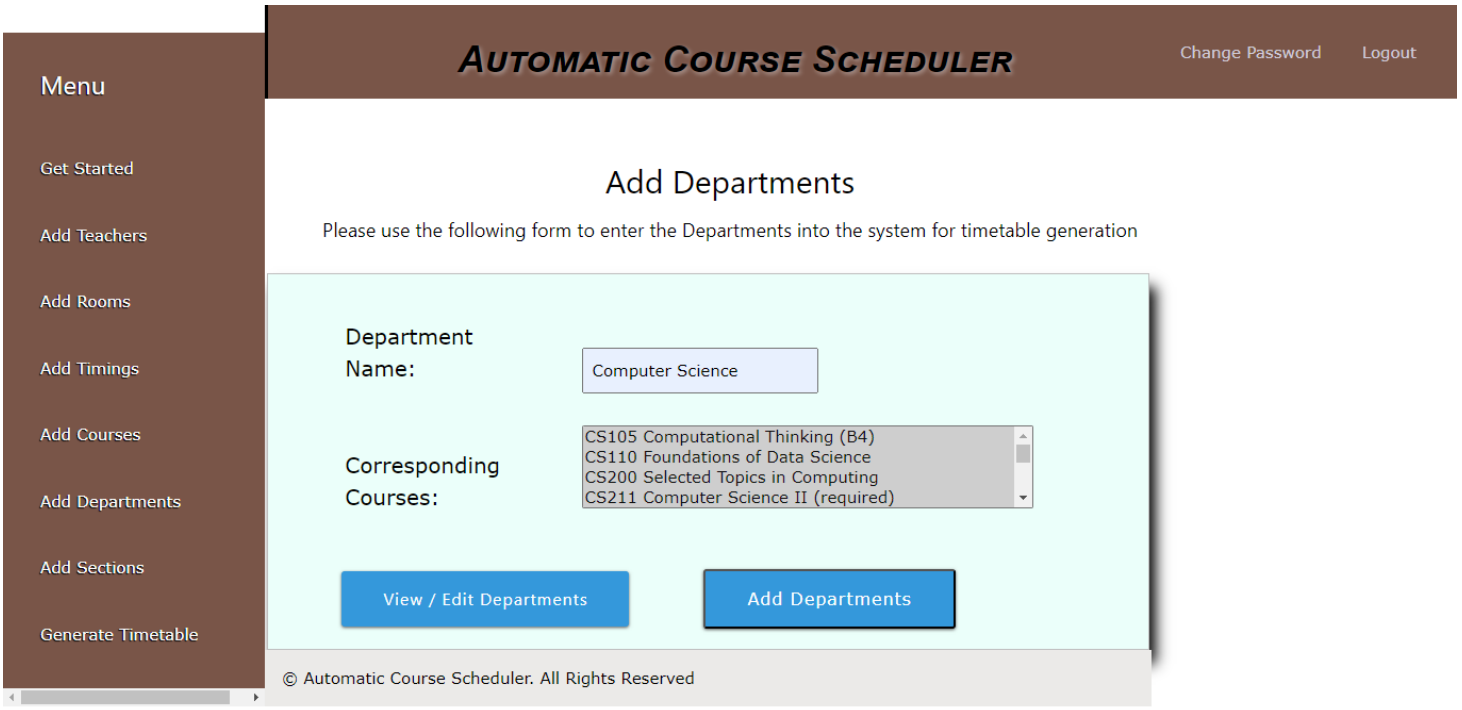
Add Courses page:



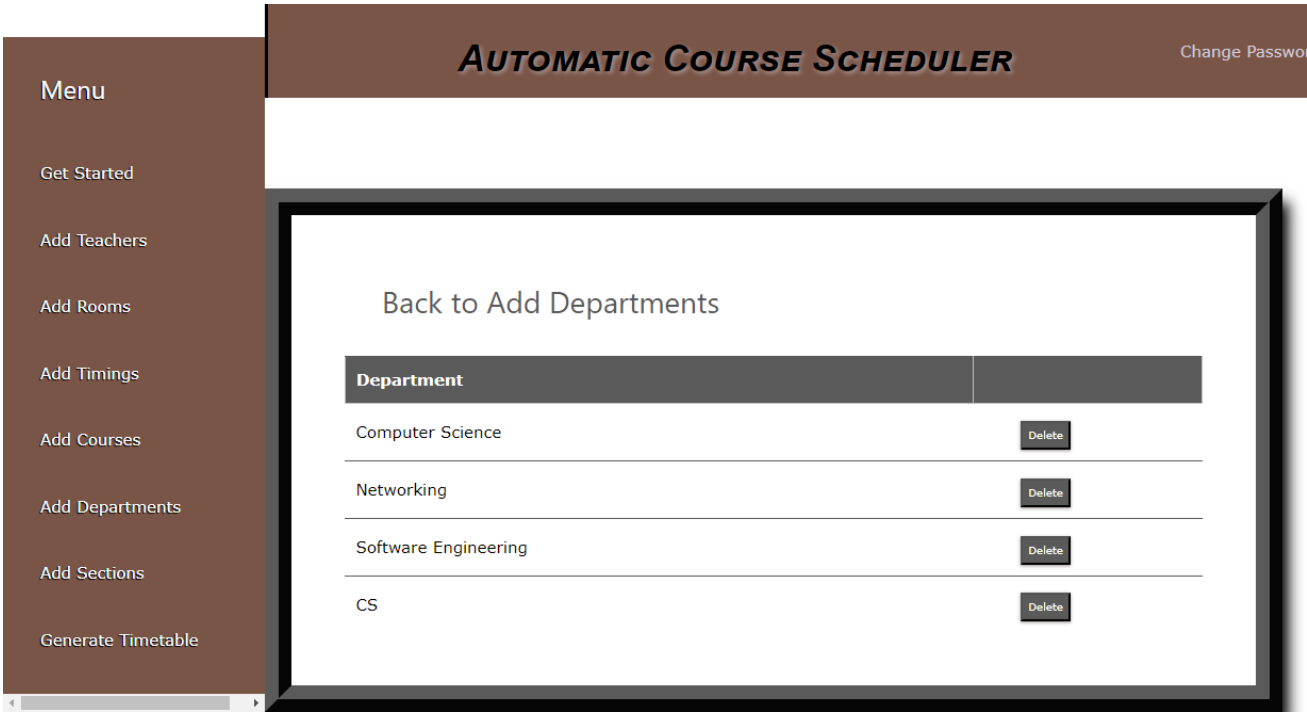
List of added courses:


AUTOMATIC COURSE SCHEDULER
Change Password
Logout
Back to Add Courses
Course Code
Course Name
Max number of students
CS105 Computational Thinking (B4) 70 Delete
CS110 Foundations of Data Science 80 Delete
CS200 Selected Topics in Computing 80 Delete
CS211 Computer Science II (required) 70 Delete
CS231 Assembly Language and Digital Circuits (50 Delete
CS301 Computer Mastery (BB) (Not for CS majors 70 Delete
CSC-2 Python 80 Delete
CS302 Algorithmic Justice 60 Delete
CS305 Problem Solving with Java Programming (100 Delete
CS306 Introduction to Computer Animation 80 Delete
CS311 Data Structures 80 Delete
CS320 Introduction to Data Science 80 Delete
CS331 Computer Architecture 90 Delete
CS351 Programming Languages 100 Delete
CS 37 Introduction to Software Engineering (re 100 Delete
CS 41 Principles and Techniques of Data Scienc 70 Delete
CS421 Theory of Computing 50 Delete
CS433 Operating Systems (required) 100 Delete
CS435 Real-Time Concepts for Embedded Systems 90 Delete
CS436 Introduction to Networking (required) 100 Delete
CS443 Database Management Systems (required) 60 Delete
CS445 Digital Embedded Systems Design with HDL 100 Delete
CS446 Cloud Computing 70 Delete
CS452 Introduction to Computer Security 25 Delete
CS455 Logic Programming 80 Delete
CS464 Numerical Analysis and Computing 80 Delete
CS471 Introduction to Artificial Intelligence 100 Delete
CS473 Artificial Neural Networks 60 Delete
CS478 Introduction to Deep Learning 100 Delete
CS480 Introduction to Opitmization 100 Delete
CS481 Introduction to Mobile Programming 50 Delete
CS485 Game Programming 80 Delete
CS486 Virtual Reality 100 Delete
CS488 Introduction to Internet of Things 80 Delete
CS490 Capstone Project 80 Delete
CS497 Topics in Computer Science 80 Delete
CS498 Individual Study in Computer Science 100 Delete
CS499 Independent Research in Computer Science 50 Delete
CS500 Research Preperation in Computer Science 90 Delete
CS511 Introduction to Bioinformatics 80 Delete
CS512 Introduction to Data Mining 50 Delete
CS513 Analysis and Intractability of Algorith 100 Delete
CS535 Introduction to Computer Graphics 60 Delete
CS536 Introduction to 3D Game Graphics 50 Delete
CS537 Data Communication and Computer Networks 50 Delete
CS538 Cryptography and Network Security 100 Delete
CS551 Advanced Programming Languages 60 Delete
CS553 Compilers 90 Delete
CS575 Machine Learning 29 Delete
CS612 Data Mining in Bioinformatics 70 Delete
Menu
Get Started
Add Teachers
Add Rooms
Add Timings
Add Courses
Add Departments
Add Sections
Generate Timetable

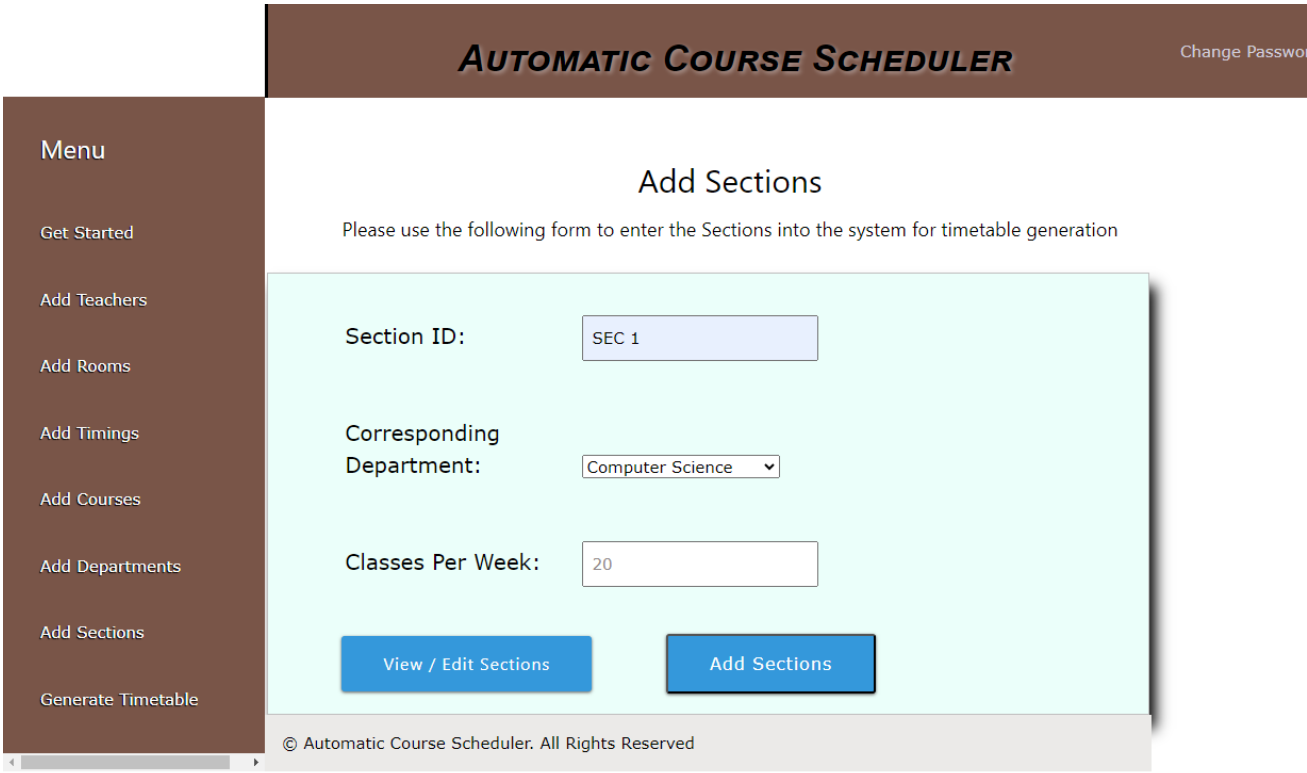
Add Department page:



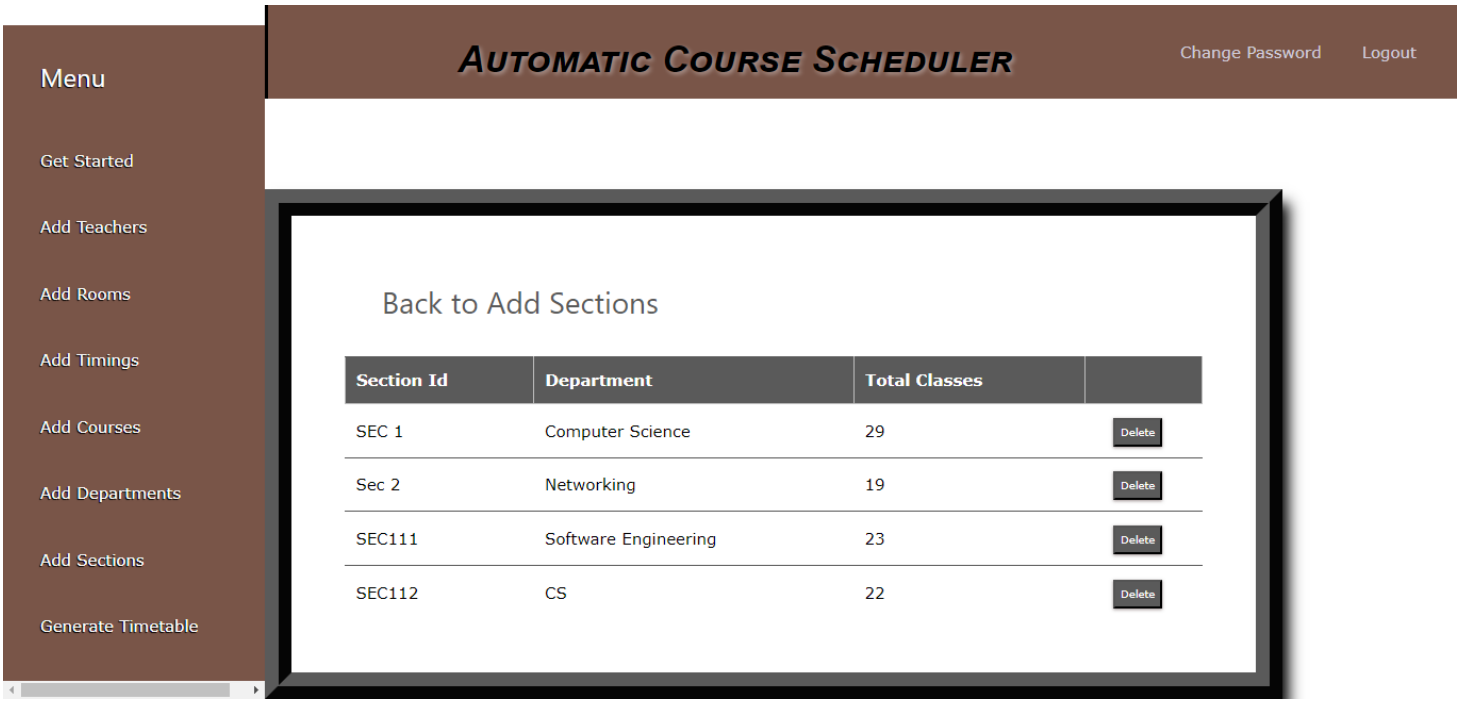
View list of department :



Add Sections page:



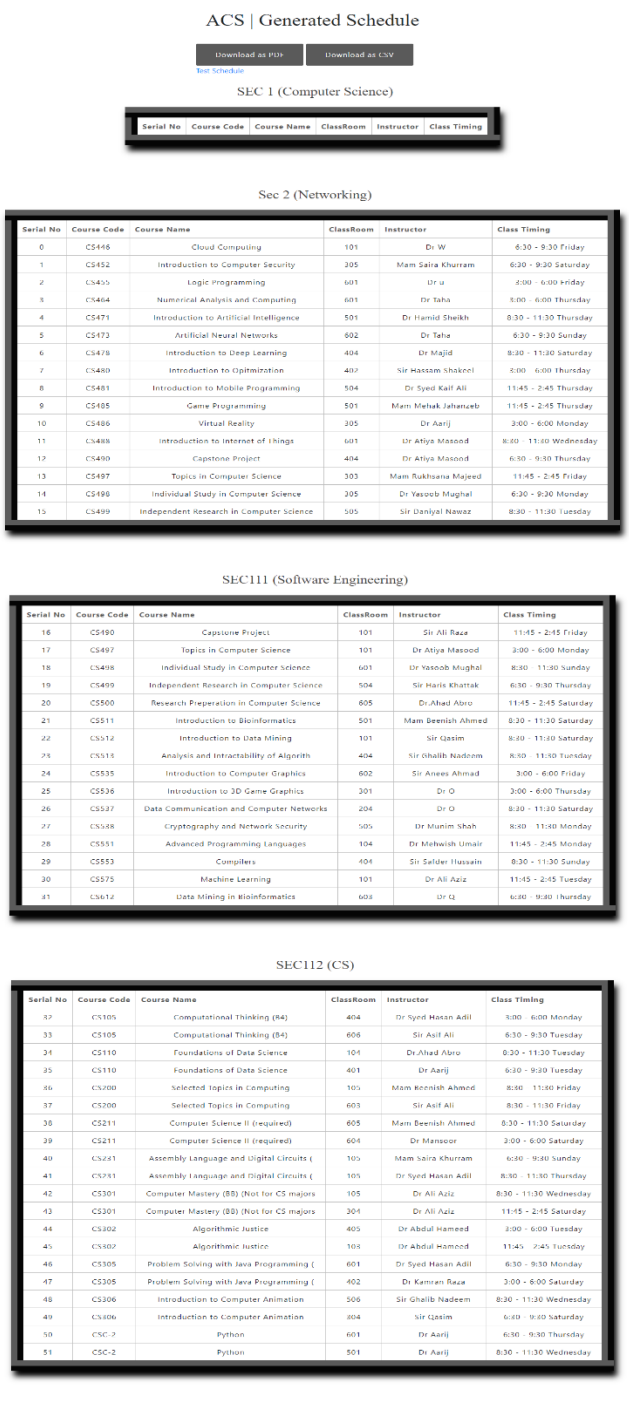
View list of Sections:



Generate TimeTable Page:


AUTOMATIC COURSE SCHEDULER
Change Password
Logout
Automatic Course Scheduler | Generate Timetable
Please make sure you have entered the correct information for the Teachers, Rooms, Timings, Courses, Sections and Departments into the system. If you have not entered all the information yet please refer below on how to go about feeding all the information into the system. Once confirmed please click on the "Generate Timetable" button below and wait patiently as Automatic Course Scheduler generates your timetable for you.
Generate Timetable»
1. Add Teacher
Navigate to the "Add Teachers" section and add the corresponding details. Incorrect details can also be removed via the "Edit Teacher" option.
2. Add Room
Navigate to the "Add Rooms" section and add the corresponding details. Incorrect details can also be removed via the "Edit Rooms" option.
3. Add Timings
Navigate to the "Add Timings" section and add the corresponding details. Incorrect details can also be removed via the "Edit Timings" option.
4. Add Course
Navigate to the "Add Courses" section and add the corresponding details. Incorrect details can also be removed via the "Edit Courses" option.
5. Add Department
Navigate to the "Add Departments" section and add the corresponding details. Incorrect details can also be removed via the "Edit Departments" option.
6. Add Section
Navigate to the "Add Sections" section and add the corresponding details. Incorrect details can also be removed via the "Edit Sections" option.
7. Generate Timetable
Navigate to the "Generate Timetable" section and click on Generate Timetable and wait patiently as our algorithm works its magic.
8. Upload Schedule
After generating the schedule make sure to download it as pdf through download_as_PDF button on schedule generated page and then upload your schedule to save in database on upload your schedule page.
9. Test Your Schedule
After generating the schedule make sure to download it as csv through download_as_CSV button on schedule generated page and then upload your schedule to Test your schedule and find conflicts in it.
Menu
Get Started
Add Teachers
Add Rooms
Add Timings
Add Courses
Add Departments
Add Sections
Generate Timetable
© Automatic Course Scheduler. All Rights Reserved

: View Generated Schedule;



ACS | Generated Schedule

Download as PDF | Download as CSV

Test Schedule

SEC 1 (Computer Science)

Serial No	Course Code	Course Name	ClassRoom	Instructor	Class Timing

Sec 2 (Networking)

Serial No	Course Code	Course Name	ClassRoom	Instructor	Class Timing
0	CS446	Cloud Computing	101	Dr W	6:30 - 9:30 Friday
1	CS452	Introduction to Computer Security	205	Mam Saira Khurram	6:30 - 9:30 Saturday
2	CS455	Logic Programming	601	Dr u	3:00 - 6:00 Friday
3	CS464	Numerical Analysis and Computing	601	Dr Taha	3:00 - 6:00 Thursday
4	CS471	Introduction to Artificial Intelligence	501	Dr Hamid Sheikh	8:30 - 11:30 Thursday
5	CS473	Artificial Neural Networks	602	Dr Taha	6:30 - 9:30 Sunday
6	CS478	Introduction to Deep Learning	404	Dr Majid	8:30 - 11:30 Saturday
7	CS480	Introduction to Opitmization	402	Sir Hassam Shakeel	3:00 - 6:00 Thursday
8	CS481	Introduction to Mobile Programming	504	Dr Syed Kaif Ali	11:45 - 2:45 Thursday
9	CS485	Game Programming	501	Mam Mehak Jahanzeb	11:45 - 2:45 Thursday
10	CS486	Virtual Reality	305	Dr Aarij	3:00 - 6:00 Monday
11	CS488	Introduction to Internet of Things	601	Dr Atiya Masood	8:30 - 11:30 Wednesday
12	CS490	Capstone Project	404	Dr Atiya Masood	6:30 - 9:30 Thursday
13	CS497	Topics in Computer Science	303	Mam Rukhsana Majeed	11:45 - 2:45 Friday
14	CS499	Individual Study in Computer Science	305	Dr Yasoob Mughal	6:30 - 9:30 Monday
15	CS499	Independent Research in Computer Science	505	Sir Daniyal Nawaz	8:30 - 11:30 Tuesday

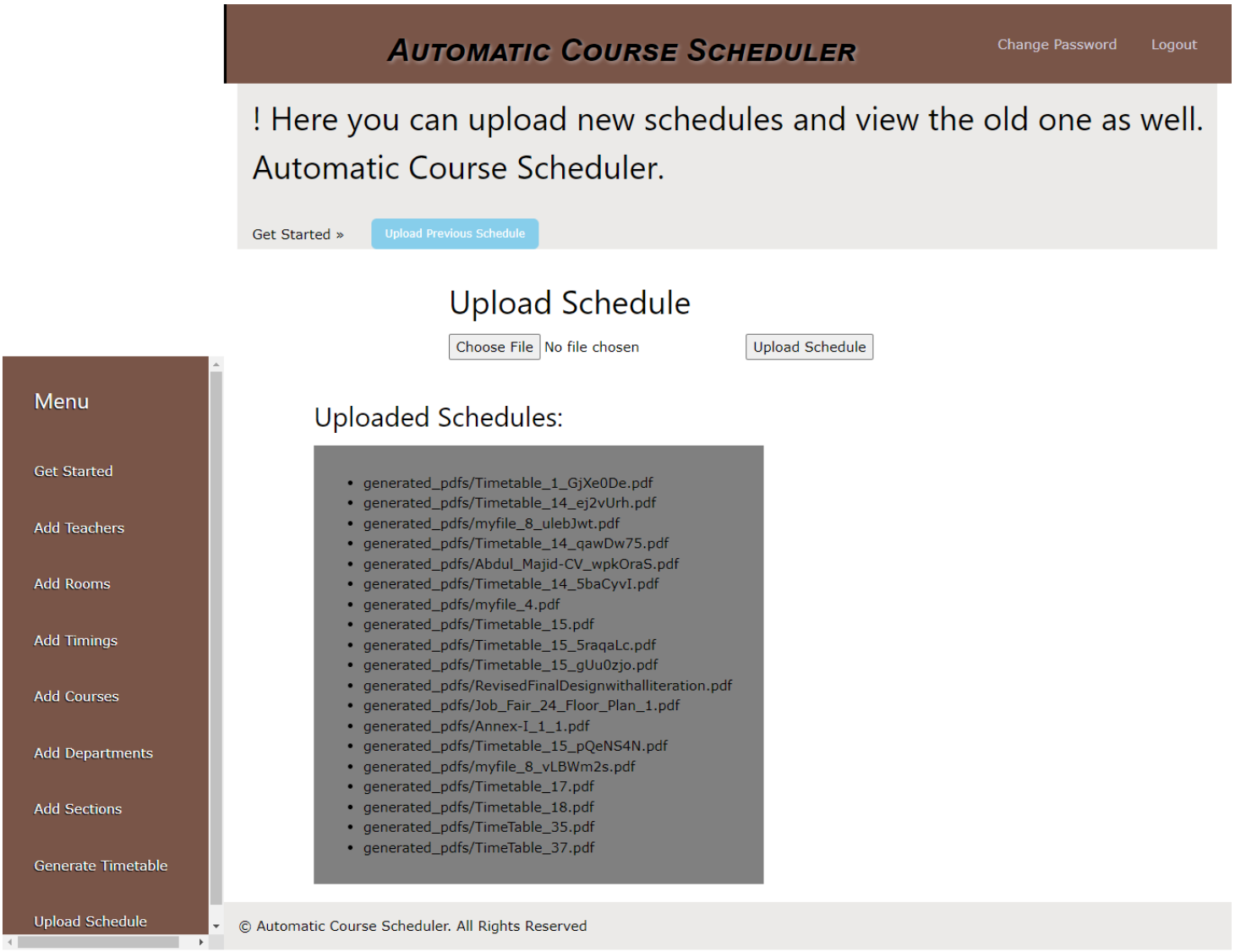
SEC111 (Software Engineering)

Serial No	Course Code	Course Name	ClassRoom	Instructor	Class Timing
16	CS490	Capstone Project	101	Sir Ali Raza	11:45 - 2:45 Friday
17	CS497	Topics in Computer Science	101	Dr Atiya Masood	3:00 - 6:00 Monday
18	CS498	Individual Study in Computer Science	601	Dr Yasoob Mughal	8:30 - 11:30 Sunday
19	CS499	Independent Research in Computer Science	504	Sir Haris Khattak	6:30 - 9:30 Thursday
20	CS500	Research Preperation in Computer Science	605	Dr.Ahad Abro	11:45 - 2:45 Saturday
21	CS511	Introduction to Bioinformatics	501	Mam Beenish Ahmed	8:30 - 11:30 Saturday
22	CS512	Introduction to Data Mining	101	Sir Qasim	8:30 - 11:30 Saturday
23	CS518	Analysis and Intractability of Algorith	404	Sir Ghalib Nadeem	8:30 - 11:30 Tuesday
24	CS535	Introduction to Computer Graphics	602	Sir Anees Ahmad	3:00 - 6:00 Friday
25	CS536	Introduction to 3D Game Graphics	301	Dr O	3:00 - 6:00 Thursday
26	CS537	Data Communication and Computer Networks	204	Dr O	8:30 - 11:30 Saturday
27	CS548	Cryptography and Network Security	505	Dr Munim Shah	8:30 - 11:30 Monday
28	CS551	Advanced Programming Languages	104	Dr Mehwish Umair	11:45 - 2:45 Monday
29	CS553	Compilers	404	Sir Safder Hussain	8:30 - 11:30 Sunday
30	CS575	Machine Learning	101	Dr Ali Aziz	11:45 - 2:45 Tuesday
31	CS612	Data Mining in Bioinformatics	603	Dr Q	6:30 - 9:30 Thursday

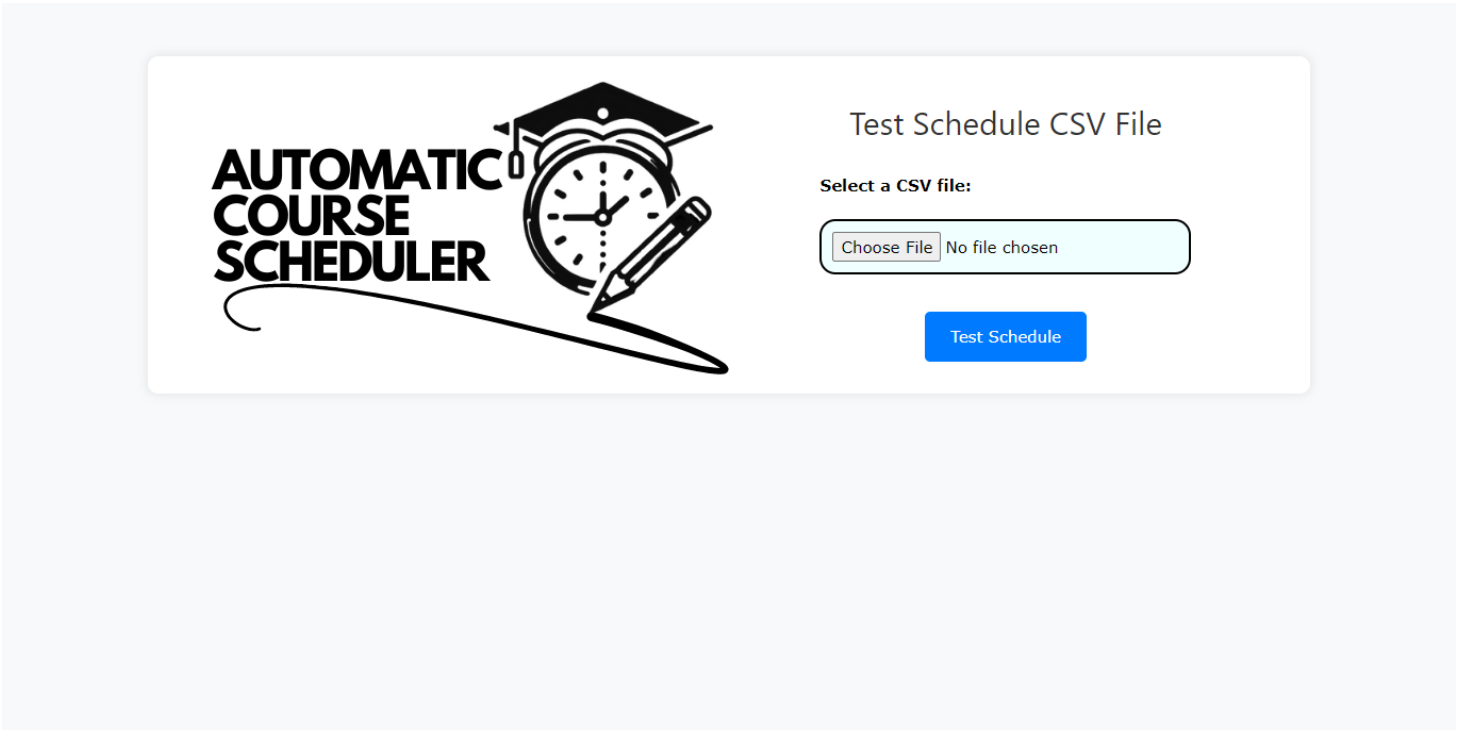
SEC112 (CS)

Serial No	Course Code	Course Name	ClassRoom	Instructor	Class Timing
32	CS105	Computational Thinking (B4)	404	Dr Syed Hasan Adil	3:00 - 6:00 Monday
33	CS105	Computational Thinking (B4)	606	Sir Asif Ali	6:30 - 9:30 Tuesday
34	CS110	Foundations of Data Science	104	Dr.Ahad Abro	8:30 - 11:30 Tuesday
35	CS110	Foundations of Data Science	401	Dr Aarij	6:30 - 9:30 Tuesday
36	CS200	Selected Topics in Computing	105	Mam Beenish Ahmed	8:30 - 11:30 Friday
37	CS200	Selected Topics in Computing	603	Sir Asif Ali	8:30 - 11:30 Friday
38	CS211	Computer Science II (required)	605	Mam Beenish Ahmed	8:30 - 11:30 Saturday
39	CS211	Computer Science II (required)	604	Dr Mansoor	3:00 - 6:00 Saturday
40	CS231	Assembly Language and Digital Circuits (	105	Mam Saira Khurram	6:30 - 9:30 Sunday
41	CS231	Assembly Language and Digital Circuits (	105	Dr Syed Hasan Adil	8:30 - 11:30 Thursday
42	CS301	Computer Mastery (B8) (Not for CS majors	105	Dr Ali Aziz	8:30 - 11:30 Wednesday
43	CS301	Computer Mastery (B8) (Not for CS majors	304	Dr Ali Aziz	11:45 - 2:45 Saturday
44	CS302	Algorithmic Justice	405	Dr Abdul Hameed	3:00 - 6:00 Tuesday
45	CS302	Algorithmic Justice	103	Dr Abdul Hameed	11:45 - 2:45 Tuesday
46	CS305	Problem Solving with Java Programming (	601	Dr Syed Hasan Adil	6:30 - 9:30 Monday
47	CS305	Problem Solving with Java Programming (	402	Dr Kamran Raza	3:00 - 6:00 Saturday
48	CS306	Introduction to Computer Animation	506	Sir Ghalib Nadeem	8:30 - 11:30 Wednesday
49	CS306	Introduction to Computer Animation	304	Sir Qasim	6:30 - 9:30 Saturday
50	CSC-2	Python	601	Dr Aarij	6:30 - 9:30 Thursday
51	CSC-2	Python	501	Dr Aarij	8:30 - 11:30 Wednesday

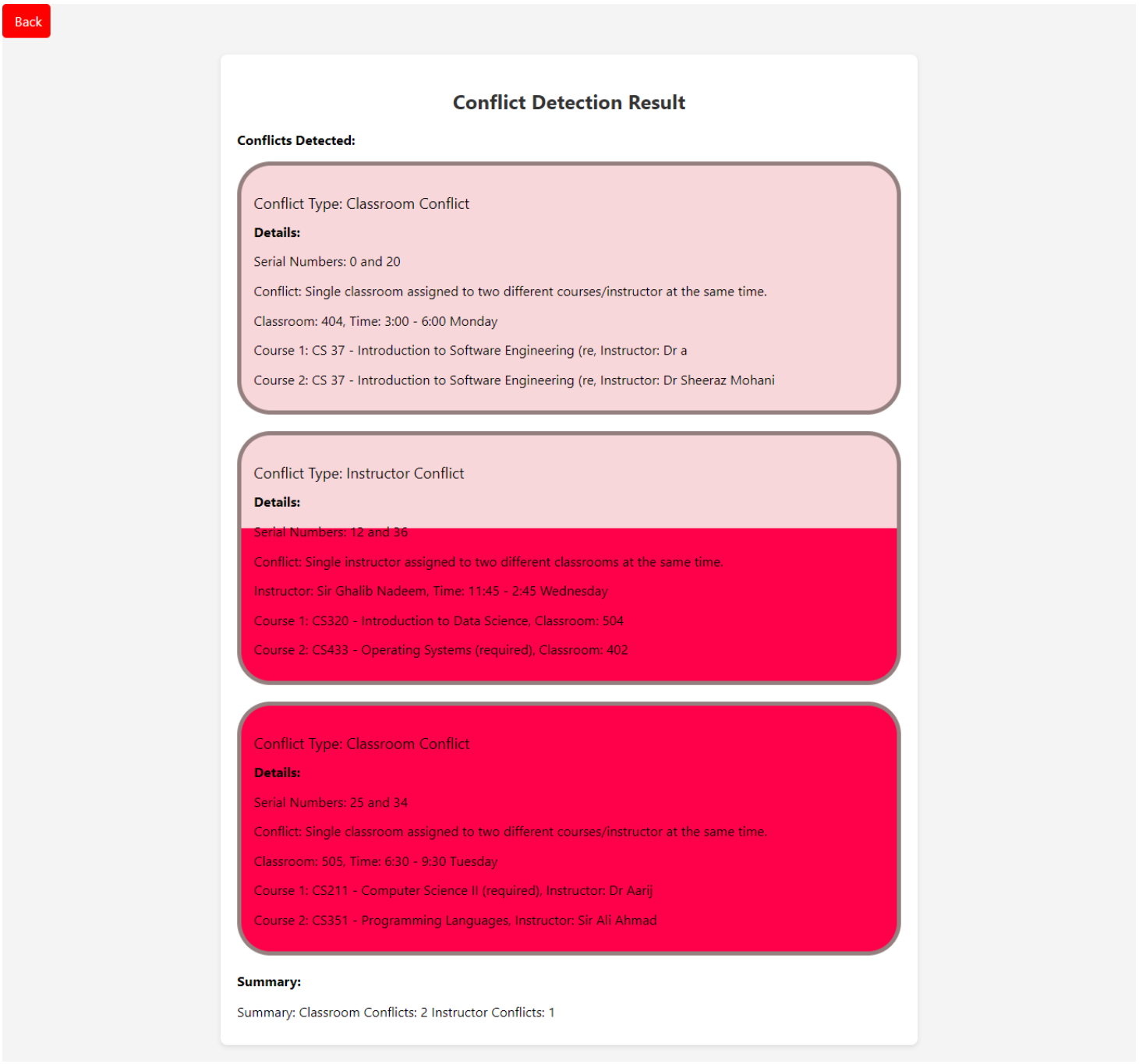
: Upload Schedule Pdf:



: Upload CSV for conflicts Resolution:



View Conflicts



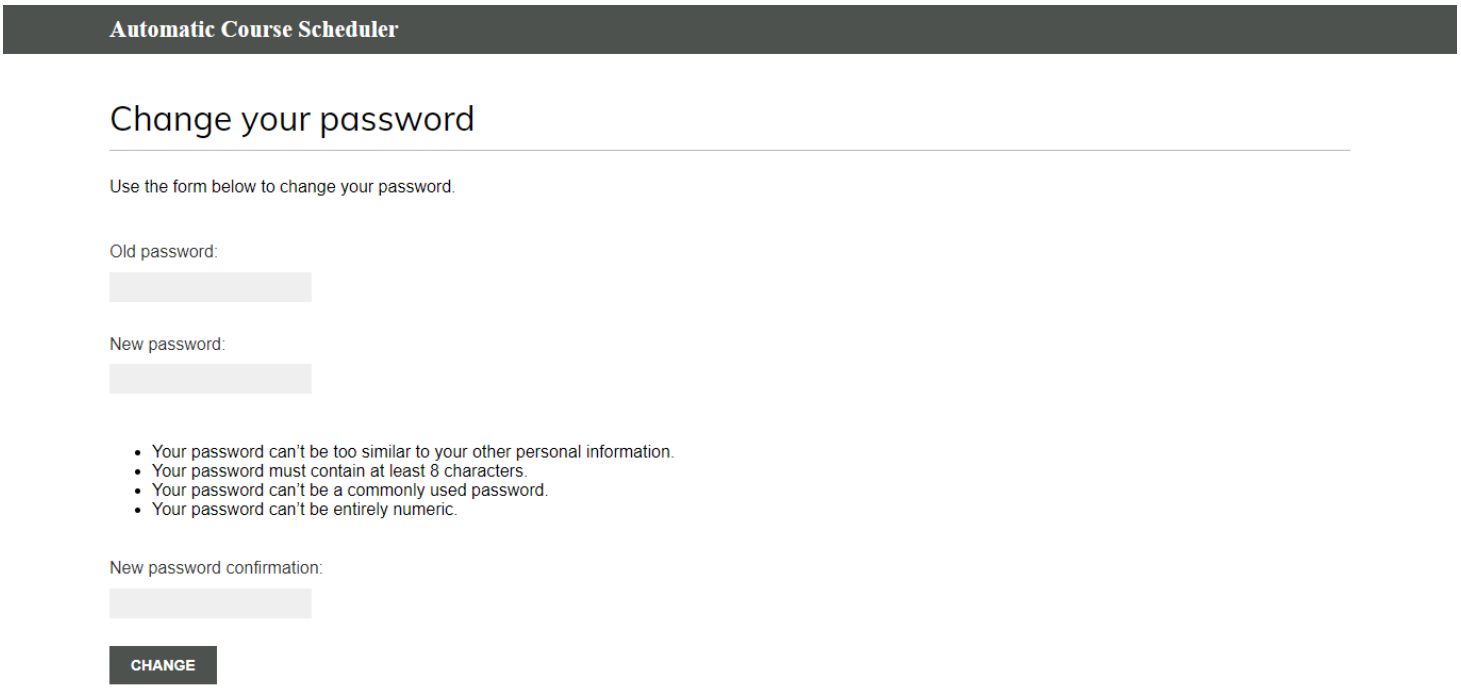
View Schedule

View Schedule

Sort by: Serial No Sort

Serial No	Course Code	Course Name	Class Room	Instructor	Class Timing
			504	Dr P	
			501	Dr Sheeraz Mohani	
			303	Dr P	
			203	Dr Hamid Sheikh	
			402	Dr Syed Kaif Ali	
			101	Dr Aarij	
			601	Sir Syed Saiq	
			402	Sir Daud Abbasi	
			402	Dr Taha	
			602	Mam Madiha Munaf	
			601	Dr Hamid Sheikh	
			501	Mam Madiha Munaf	
			305	Sir Ali Raza	
			605	Dr Atiya Masood	
			402	Dr Yasoob Mughal	
			303	Sir Haris Khattak	
			605	Dr Atiya Masood	
			304	Sir Daniyal Nawaz	
			505	Dr Yasoob Mughal	
			103	Sir Daniyal Nawaz	
			203	Dr.Ahad Abro	
			504	Mam Beenish Ahmed	
			302	Sir Qasim	
			601	Dr Sheeraz Mohani	
			105	Dr b	
			605	Dr P	
			406	Dr Taha	
			305	Sir Syed Saiq	
			501	Sir Ali Raza	
			202	Sir Israr Ali	
			106	Dr Sheeraz Mohani	
			403	Dr u	
			605	Sir Asif Ali	
			105	Sir Asif Ali	
			106	Dr Aarij	
			104	Dr Kamran Raza	
			605	Mam Beenish Ahmed	
			506	Sir Asif Ali	
			402	Dr Mansoor	
			602	Dr Aarij	
			604	Sir Asif Ali	
			305	Sir Asif Ali	
			305	Dr Ali Aziz	
			601	Mam Sophia Ajaz	
			404	Mam Sophia Ajaz	
			604	Dr Ali Aziz	
			402	Sir Qasim	
			305	Dr Kamran Raza	
			206	Syed Zain Mir	
			305	Dr Syed Hasan Adil	
			501	Dr Aarij	
			605	Dr Aarij	

Change Password



: Logout window:

Logged out

You have been successfully logged out. You can log-in again or go back to the Homepage.

5.3 Summary

In this chapter, we showcase screenshots of our complete system, offering a comprehensive visual overview of its user interface and functionality. These images provide insight into how our project looks and operates, allowing for a better understanding of its design and features. By including these screenshots, we aim to offer readers a clearer perspective on the overall structure and layout of our system.

CHAPTER – 6

6.1 Introduction:

In Chapter 6, the focus shifts towards the crucial testing phase of the automatic course scheduling software. This phase plays a pivotal role in ensuring that the developed system meets the desired standards of functionality, performance, and reliability. The chapter outlines a comprehensive set of test cases meticulously designed to evaluate different aspects of the software's behavior.

The testing process encompasses various dimensions, including input validation, schedule generation, conflict resolution, export functionality, and user interface usability. Each test case is meticulously crafted to assess specific functionalities and features of the software, ensuring that it operates flawlessly under different scenarios and conditions.

By systematically executing these test cases, the chapter aims to validate the accuracy and effectiveness of the automatic course scheduling system. Through rigorous testing, any potential issues, bugs, or inconsistencies can be identified and addressed promptly, thus enhancing the overall quality and robustness of the software.

6.2 Test Cases:

Test Case 1:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.1	**Test Type**	Functionality
Test Case Description	1. To test that login button works. 2. To test that signup button works.		
Test Steps	1. Click on login button. 2. Click on signup button.		
Expected Result	1. Click on login redirect to login page. 2. Click on signup redirect to sign up.		
Actual Result	1. Click on login redirect to login page. 2. Click on signup redirect to sign up.		
Pass/Fail	Pass		

Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 2:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.2	**Test Type**	Functionality
Test Case Description	1. To test that the Register new account text on Login screen. 2. To test that Login here text on Register screen.		
Test Steps	1. Click on text login here 2. Click on text register here		
Expected Result	1. Open registration screen 2. Open Login screen		
Actual Result	1. Open registration screen. 2. Open login screen.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 3:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.3	**Test Type**	Functionality
Test Case Description	To check that text fields in Register Screen must be mandatory to register a unique use.		
Test Steps	Click on Create my Account.		
Expected Result	Please fill out the required fields.		
Actual Result	Please fill out the required fields.		
Pass/Fail	Pass		

Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 4:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.4	**Test Type**	Functionality
Test Case Description	To check password and match password function works.		
Test Steps	Click on Create my Account.		
Expected Result	Password didn't match.		
Actual Result	Password didn't match.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 5:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.5	**Test Type**	Functionality
Test Case Description	To check create my account button creates the account of user		
Test Steps	Click on Create my Account.		
Expected Result	User account get created.		
Actual Result	User account get created.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 6:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.6	**Test Type**	Functionality
Test Case Description	To check that text fields in Login Screen must be mandatory to login into co generation panel.		
Test Steps	Click on Login Button.		
Expected Result	User login into in the schedule generation page.		
Actual Result	User login into in the schedule generation page.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 7:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.7	**Test Type**	Functionality
Test Case Description	To check that wrong credential must not login into system.		
Test Steps	Click on Login Button.		
Expected Result	Enter correct username and password.		
Actual Result	Enter correct username and password.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 8:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler

Test Case Id	1.8	**Test Type**	Functionality
Test Case Description	To check the Add Instructors functions works properly and instructors are added into database.		
Test Steps	Click on Add instructor.		
Expected Result	Instructor is added into database.		
Actual Result	Instructor is added into database.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 9:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.9	**Test Type**	Functionality
Test Case Description	To check the Add Instructors text input fields must be mandatory to add instructor.		
Test Steps	Click on Add instructor.		
Expected Result	Please fill the require field.		
Actual Result	Please fill the require field.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 10:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.10	**Test Type**	Functionality
Test Case Description	To check the view instructor button works and it show the already added		

	instructors.
Test Steps	Click on View/Edit instructor.
Expected Result	Display all the added instructors.
Actual Result	Display all the added instructors.
Pass/Fail	Pass
Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 11:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.11	**Test Type**	Functionality
Test Case Description	To check that added instructors can be deleted from the list of instructors..		
Test Steps	Click on delete button.		
Expected Result	Instructor is deleted from the database.		
Actual Result	Instructor is deleted from the database.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 11:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.11	**Test Type**	Functionality
Test Case Description	To check that back button which redirect to add instructor panel from view instructor panel works.		
Test Steps	Click on Back to add instructors.		
Expected Result	Redirect to add instructor page.		
Actual Result	Redirect to add instructor page.		

Pass/Fail	Pass
Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 12:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.12	**Test Type**	Functionality
Test Case Description	To check the Add Room functions works properly and Rooms are added into database.		
Test Steps	Click on Add Room.		
Expected Result	Room is added into database.		
Actual Result	Room is added into database.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 13:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.13	**Test Type**	Functionality
Test Case Description	To check the Add Room text input fields must be mandatory to add Room.		
Test Steps	Click on Add Room.		
Expected Result	Please fill the require field.		
Actual Result	Please fill the require field.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		

Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 14:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.14	**Test Type**	Functionality
Test Case Description	To check the view room button works and it show the already added rooms.		
Test Steps	Click on View/Edit room.		
Expected Result	Display all the added room.		
Actual Result	Display all the added room.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 15:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.15	**Test Type**	Functionality
Test Case Description	To check that added room can be deleted from the list of room..		
Test Steps	Click on delete room button.		
Expected Result	Room is deleted from the database.		
Actual Result	Room is deleted from the database.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 16:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.16	**Test Type**	Functionality
Test Case Description	To check that back button which redirect to add room panel from view room panel works.		
Test Steps	Click on Back to add room.		
Expected Result	Redirect to add room page.		
Actual Result	Redirect to add room page.		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 17:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.17	**Test Type**	Functionality
Test Case Description	1. To check the Add Meeting time functions works properly and meeting time are added into database. 2. To check all the required fields must be filled. 3. To check view meeting time button works and show already added meeting times. 4. To check meeting times can be deleted from the view panel. 5. To check that back button which redirect to add meeting panel from view meeting time panel works.		
Test Steps	1. Click on Add meeting time. 2. Click on Add meeting time. 3. Click on View meeting time. 4. Click on delete meeting time 5. Click on back to add meeting time button.		
Expected Result	1. Meeting time is added into database. 2. Please fill the require field. 3. Display all the added Meeting times. 4. Meeting time is deleted from the database.		

	5. Redirect to add meeting time page.
Actual Result	1. Meeting time is added into database. 2. Please fill the require field. 3. Display all the added Meeting times. 4. Meeting time is deleted from the database. 5. Redirect to add meeting time page.
Pass/Fail	Pass
Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 18:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.18	**Test Type**	Functionality
Test Case Description	1. To check the Add Courses functions works properly and Courses are added into database. 2. To check all the required fields must be filled. 3. To check view Courses button works and show already added Courses. 4. To check instructor selection box select the instructors. 5. To check search instructor function searching the instructor. 6. To check Courses can be deleted from the view panel. 7. To check that back button which redirect to add Courses from view Course panel works.		
Test Steps	1. Click on Add Course. 2. Click on Add course. 3. Click on View courses. 4. Click on tick box. 5. Search inn search box. 6. Click on delete Course. 7. Click on back to add Course button.		
Expected Result	1. Courses is added into database. 2. Please fill the require field. 3. Display all the added Courses. 4. Show tick in the selected instructor box.		

	5. Display name of instructor who is being searched. 6. Course is deleted from the database. 7. Redirect to add courses page.
Actual Result	1. Courses is added into database. 2. Please fill the require field. 3. Display all the added Courses. 4. Show tick in the selected instructor box. 5. Display name of instructor who is being searched. 6. Course is deleted from the database. 7. Redirect to add courses page.
Pass/Fail	Pass
Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 19:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.19	**Test Type**	Functionality
Test Case Description	1. To check the Add Department functions works properly and Departments are added into database. 2. To check all the required fields must be filled. 3. To check view Department button works and show already added Departments. 4. To check Department corresponding courses are selected to add in department. 5. To check Department can be deleted from the view panel. 6. To check that back button which redirect to add department from view department panel works.		
Test Steps	1. Click on Add Department. 2. Click on Add Department. 3. Click on View department. 4. Select multiple courses. 5. Click on delete department. 6. Click on back to add department button.		
Expected Result	1. Department is added into database.		

	2. Please fill the require field. 3. Display all the added Departments. 4. Show Selection on the selected Courses box. 5. Department is deleted from the database. 6. Redirect to add Departments page.
Actual Result	1. Department is added into database. 2. Please fill the require field. 3. Display all the added Departments. 4. Show Selection on the selected Courses box. 5. Department is deleted from the database. 6. Redirect to add Departments page.
Pass/Fail	Pass
Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 20:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.20	**Test Type**	Functionality
Test Case Description	1. To check the Add Section functions works properly and Sections are added into database. 2. To check all the required fields must be filled. 3. To check view Sections button works and show already added Sections. 4. To check corresponding department drop down menu show the added departments. 5. To check Sections can be deleted from the view panel. 6. To check that back button which redirect to add sections from view sections panel works.		
Test Steps	1. Click on Add Section. 2. Click on Add Section. 3. Click on View Section. 4. Click on Corresponding Department. 5. Click on delete sections. 6. Click on back to add section button.		

Expected Result	1. Section is added into database. 2. Please fill the require field. 3. Display all the added Sections. 4. Show Corresponding Department in menu. 5. Section is deleted from the database. 6. Redirect to add Section page.
Actual Result	1. Section is added into database. 2. Please fill the require field. 3. Display all the added Sections. 4. Show Corresponding Department in menu. 5. Section is deleted from the database. 6. Redirect to add Section page.
Pass/Fail	Pass
Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 21:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.21	**Test Type**	Functionality
Test Case Description	To check generate timetable button generates the timetable and display it after the generation.		
Test Steps	Click on Generate Timetable.		
Expected Result	The generated timetable is shown after the generation		
Actual Result	The generated timetable is shown after the generation		
Pass/Fail	Pass		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 22:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.22	**Test Type**	Functionality
Test Case Description	To check generate timetable button Show the loading bar after the click event perform on it.		
Test Steps	Click on Generate Timetable.		
Expected Result	The loading bar display before the generated timetable shows.		
Actual Result	In title bar loading icon is shown but not on web page.		
Pass/Fail	Fail		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 23:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.23	**Test Type**	Functionality
Test Case Description	To download the generated table in pdf and in csv format by clicking on download button.		
Test Steps	1. Click on Download as PDF. 2. Click on download as CSV.		
Expected Result	1. The PDF file is downloaded. 2. The CSV file is downloaded.		
Actual Result	1. The PDF file is downloaded. 2. The CSV file is downloaded.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 24:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.24	**Test Type**	Functionality
Test Case Description	To check that pdf file is can be uploaded into database through upload pdf.		
Test Steps	1. Click on choose file 2. Click on upload button.		
Expected Result	1. Open windows explorer to select the file. 2. The file save in database and display below.		
Actual Result	1. Open windows explorer to select the file. 2. The file save in database and display below.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 25:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.25	**Test Type**	Functionality
Test Case Description	To check that CSV file is uploading to check the schedule constrains.		
Test Steps	1. Click on choose file 2. Click on upload button.		
Expected Result	1. Open windows explorer to select the file. 2. The file save in database and display schedule conflicts.		
Actual Result	3. Open windows explorer to select the file. 4. The file save in database and display schedule conflicts.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 26:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.26	**Test Type**	Functionality
Test Case Description	To check that after CSV file uploaded it shows the conflicts.		
Test Steps	Click on Test Schedule.		
Expected Result	Showing the schedule conflicts in the upload CSV file.		
Actual Result	Showing the schedule conflicts in the upload CSV file.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 27:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.27	**Test Type**	Functionality
Test Case Description	To verify that the conflicts are valid and appearing in the CSV file as well.		
Test Steps	Click on Test Schedule button.		
Expected Result	If there are conflicts then show them on web page or show that no conflict detected.		
Actual Result	If there are conflicts then show them on web page or show that no conflict detected.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 28:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler

Test Case Id	1.28	**Test Type**	Functionality
Test Case Description	To make sure that it display what type of conflicts are there and show there description with detail and summary from the uploaded CSV file.		
Test Steps	Click on Test Schedule button.		
Expected Result	Display conflict information		
Actual Result	Display conflict information		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 29:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.29	**Test Type**	Functionality
Test Case Description	To check that a user can change his\her password by the use of change password page.		
Test Steps	Click on Change Password.		
Expected Result	Password has been change successfully.		
Actual Result	Password has been change successfully.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 30:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.30	**Test Type**	Functionality
Test Case Description	To check that user text input fields must be mandatory to fill.		

Test Steps	Click on Change Password.
Expected Result	Please fill out this field.
Actual Result	Please fill out this field.
Pass/Fail	PASS
Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 31:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.31	**Test Type**	Functionality
Test Case Description	To check that user text input fields must be mandatory to fill.		
Test Steps	Click on Change Password.		
Expected Result	Please fill out this field.		
Actual Result	Please fill out this field.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 32:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.32	**Test Type**	Functionality
Test Case Description	To check that if user has given wrong old password it must show that password can’t be change.		
Test Steps	Click on Change Password.		
Expected Result	Your old password was entered incorrectly. Please enter it again.		
Actual Result	Your old password was entered incorrectly. Please enter it again.		
Pass/Fail	PASS		

Date Prepared	March 15, 2024
Date Run	March 22, 2024
Prepared By	Abdul Majid
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

TEST CASE 33:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.33	**Test Type**	Functionality
Test Case Description	To check that user new password and new password confirmation does not miss match.		
Test Steps	Click on Change Password.		
Expected Result	The two password fields didn't match.		
Actual Result	The two password fields didn't match.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		
Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza		

TEST CASE 34:

Requirement Reference	1	**Project Name**	Automatic Course Scheduler
Test Case Id	1.34	**Test Type**	Functionality
Test Case Description	To check that click on logout button make sure user is logged out of the system.		
Test Steps	Click on logout.		
Expected Result	User get logged out.		
Actual Result	User get logged out.		
Pass/Fail	PASS		
Date Prepared	March 15, 2024		
Date Run	March 22, 2024		
Prepared By	Abdul Majid		

Tested By	Taha Ammar Mir and Muhammad Ghulam Hamza

6.3 Summary:

Chapter 6 delves into the pivotal testing phase of the automatic course scheduling software. Through a systematic approach, various test cases were meticulously designed and executed to assess the functionality, performance, and reliability of the system. The tests encompassed input validation, schedule generation, conflict resolution, export functionality, and user interface usability. Each test case aimed to validate specific aspects of the software, ensuring its accuracy and effectiveness under diverse scenarios.

By rigorously testing the system, potential issues and inconsistencies were identified and addressed, thereby enhancing its overall quality and robustness. The outcomes of the testing phase provide valuable insights into the software's performance and readiness for deployment in real-world educational environments.

Overall, the error handling feature proved to be a valuable addition to the automated course scheduler testing results demonstrate the effectiveness, reliability, and performance of the automated course scheduler in generating optimized schedules for educational institutions. The scheduler's functionality, compliance with requirements, and user acceptance validate its suitability for practical deployment and use in real-world scheduling scenarios.

.

CHAPTER – 7

7.1 Introduction:

Chapter 6 serves as the culmination of the final year of the project, providing a comprehensive overview of the work undertaken, achievements made, and areas for future exploration. The chapter begins by summarizing the key findings and outcomes of the project, highlighting the successful development and implementation of the automatic course scheduling software. It outlines the methodology employed, including the use of Python Django framework and genetic algorithms, to address the complexities of course scheduling in educational institutions.

Moreover, Chapter 6 delves into the challenges encountered during the project, ranging from data acquisition and validation to algorithm optimization and user interface design. These challenges, though significant, were overcome through collaborative efforts and iterative refinement of the software.

Furthermore, the chapter discusses the limitations of the current implementation, such as scalability issues with large datasets and the need for additional optimization in certain algorithmic aspects. Despite these limitations, the software demonstrates promising results and lays a solid foundation for future enhancements and developments.

In conclusion, Chapter 6 outlines the roadmap for future work, including plans for further optimization, scalability improvements, and integration of additional features based on user feedback. It underscores the project's ongoing evolution and commitment to addressing the evolving needs of educational institutions in automating course scheduling processes. Through continued research, development, and collaboration, the automatic course scheduling software aims to remain at the forefront of educational technology, facilitating efficient and optimized scheduling solutions for academic institutions worldwide.

7.2 System Limitations and Challenges:

The automatic course scheduler is an innovative solution aimed at streamlining the scheduling process in educational institutions. Designed to automate and optimize course assignments and teacher schedules, it offers significant advantages in efficiency and resource utilization. However, like any technological solution, the automatic course scheduler also faces its own set of limitations and challenges. In this section, we will explore these limitations and challenges to provide a comprehensive understanding of the system's constraints and areas for improvement.

- Scalability issues with large datasets may lead to performance degradation.
- Optimization algorithms, such as genetic algorithms, require extensive tuning and optimization.
- Data acquisition and validation present challenges in obtaining accurate and up-to-date data.
- Ensuring data quality and integrity is essential for minimizing scheduling conflicts.
- User interface design and usability may pose challenges for user adoption and acceptance.
- Designing an intuitive and user-friendly interface requires careful consideration of user needs.
- Ongoing efforts are needed to address these challenges and improve system functionality and

performance.

7.3 Future Work:

Future work on the automatic course scheduler project encompasses several potential enhancements and features aimed at further improving its functionality and user experience. Some of the key areas for future development include:

- **Instructor Panel:**

  Introducing a dedicated instructor panel within the system to provide instructors with visibility into their assigned courses, schedules, and any relevant updates. This panel could offer personalized dashboards for instructors to view their course assignments, scheduling preferences, and classroom allocations. Additionally, interactive features such as calendar views and notification settings could enhance instructor engagement and facilitate effective communication between instructors and administrators.

- **Email Notifications:**

  Implementing email notification capabilities to alert instructors about any changes or updates to their schedules in real-time. By integrating email notifications into the system, instructors can receive timely updates on schedule modifications, class cancellations, or new course assignments directly to their email inbox. This feature enhances communication and ensures that instructors stay informed and up-to-date on their teaching commitments without the need to frequently check the system.

- **Enhanced Reporting and Analytics:**

  Enhancing the reporting and analytics capabilities of the system to provide administrators and stakeholders with valuable insights into scheduling trends, resource utilization, and performance metrics. By incorporating advanced reporting tools and data visualization techniques, the system can generate comprehensive reports and analytics dashboards to support informed decision-making and strategic planning. This includes analyzing historical scheduling data, identifying patterns, and optimizing scheduling processes to improve overall efficiency and effectiveness.

- **Integration with Learning Management Systems (LMS):**

  Exploring opportunities to integrate the automatic course scheduler with existing learning management systems (LMS) used by educational institutions. Seamless integration with LMS platforms allows for synchronized data exchange, streamlined workflows, and centralized management of course scheduling and academic resources. This integration enhances interoperability and facilitates a cohesive ecosystem for managing educational operations.

7.4 Conclusion:

In conclusion, the development of the automatic course scheduler represents a significant advancement in educational scheduling technology. By leveraging genetic algorithms and modern web technologies, this project has demonstrated the feasibility and effectiveness of automating course scheduling processes in educational institutions. The system's ability to generate optimized schedules while considering various constraints and preferences has the potential to streamline operations, improve efficiency, and enhance academic outcomes. Despite the challenges and limitations encountered during the development process, the project has laid the foundation for future enhancements and advancements in educational scheduling systems. Moving forward, continued research and development efforts will focus on refining the system, incorporating user feedback, and

addressing emerging needs to ensure its continued success and impact in the field of education.

REFERENCES

[1]. Achini Herath, Genetic Algorithm For University Course Timetabling Problem Published 2017, Computer Science https://www.semanticscholar.org/paper/Genetic-Algorithm-For-University-Course-Timetabling-Herath/ec0047d00d1a84a31ed731a92809e6b90b22a0c3?utm_source=direct_link

[2]. A Genetic Algorithm Based University Timetabling System, E. Burke, D. Elliman, Rupert F. Weare, Published 1994, Computer Science, Education https://www.semanticscholar.org/paper/A-Genetic-Algorithm-Based-University-Timetabling-Burke-Elliman/cdfa597af238c36a565463228636fa0f459301e9?utm_source=direct_link

[3]. COURSE SCHEDULING OPTIMIZATION SYSTEM MODELING WITH GENETIC ALGORITHMS, Eka Yulia Sari, Dina Yulina Heriyani, dan Titik Rahmawati, Published 2023, Engineering https://www.semanticscholar.org/paper/(COURSE-SCHEDULING-OPTIMIZATION-SYSTEM-MODELING-Sari-Heriyani/1b6ec9a14a931fdc1214792b590c99fc8ef3c76b?utm_source=direct_link

[4]. Implementation of Genetic Algorithm to academic scheduling system, Hanny Prastya Hariyadi, Triyanna Widiyaningtyas,S. Sendari, Published in IEEE Region 10 Conference 1 November 2016, Computer Science https://www.semanticscholar.org/paper/Implementation-of-Genetic-Algorithm-to-academic-Hariyadi-Widiyaningtyas/b1c3e5bccaea6885f20cb1e70ca5a6eafae57d9e?utm_source=direct_link

[5]. Design and Application of an Improved Genetic Algorithm to a Class Scheduling System, Xiangliu Chen, Xiaofeng Yue, R. Li, A. Zhumadillayeva, Ruru Liu, Published in International Journal of Emerging Technologies in Learning (iJET), 12 January 2021, Computer Science https://www.semanticscholar.org/paper/Design-and-Application-of-an-Improved-Genetic-to-a-Chen-Yue/a5a262e0926b442dc04a0abf93441b59010f5d0f?utm_source=direct_link

[6]. Knowledge-based genetic algorithm for university course timetabling problems, H. Kanoh, Y. Sakamoto, Published in Int. J. Knowl. Based Intell. Eng. Syst , 1 December 2008, Computer Science, Int. J. Knowl. Based Intell. Eng. Syst. https://www.semanticscholar.org/paper/Knowledge-based-genetic-algorithm-for-university-Kanoh-Sakamoto/93db0eade382959f7501cde6b4ddc6d4a1fd4528?utm_source=direct_link

[7]. School timetabling for quality student and teacher schedules, Theodore Birbas, S. Daskalaki, E. Housos, Published in Journal of Scheduling 1 April 2009, Education, Mathematics, https://www.semanticscholar.org/paper/School-timetabling-for-quality-student-and-teacher-Birbas-Daskalaki/472aae346a7a79a9a6fbefc90cfc09389e620d50?utm_source=direct_link

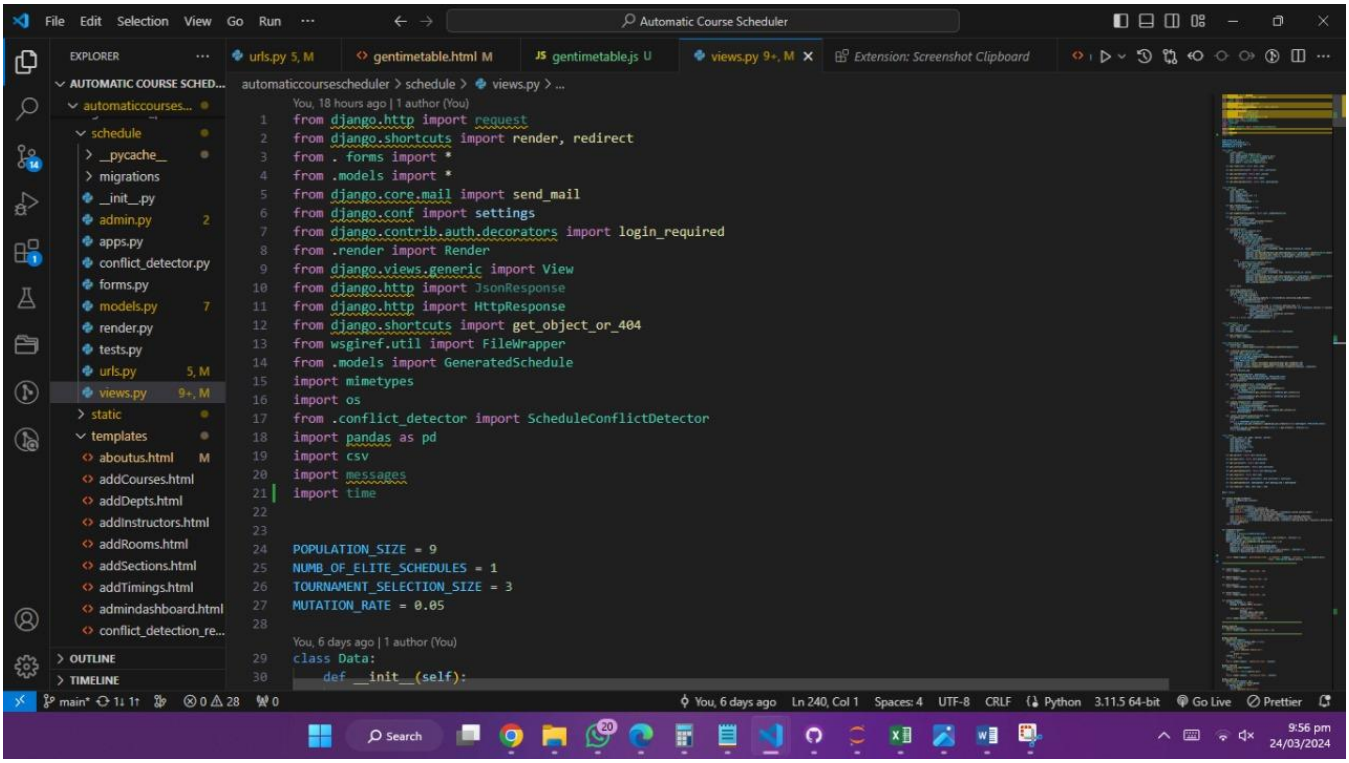
[8]. Automatic Timetable Generation using Genetic Algorithm, D. Mittal, H. Doshi, Mohammed Sunasra, Renuka Nagpure, Published 28 February 2015, Computer Science, International Journal of Advanced Research in Computer and Communication Engineering https://www.semanticscholar.org/paper/Automatic-Timetable-Generation-using-Genetic-Mittal-Doshi/a08fddd492660c1dd49c0fd001d2d570373220f0?utm_source=direct_link

[9]. K. F. Man, K. S. Tang, and S. Kwong, Genetic Algorithms: Concepts and Applications, IEEE Trans. Ind. Electron. 43 (5), 519 – 534, (1996) Genetic algorithms: concepts and applications [in engineering design] | IEEE Journals & Magazine | IEEE Xplore

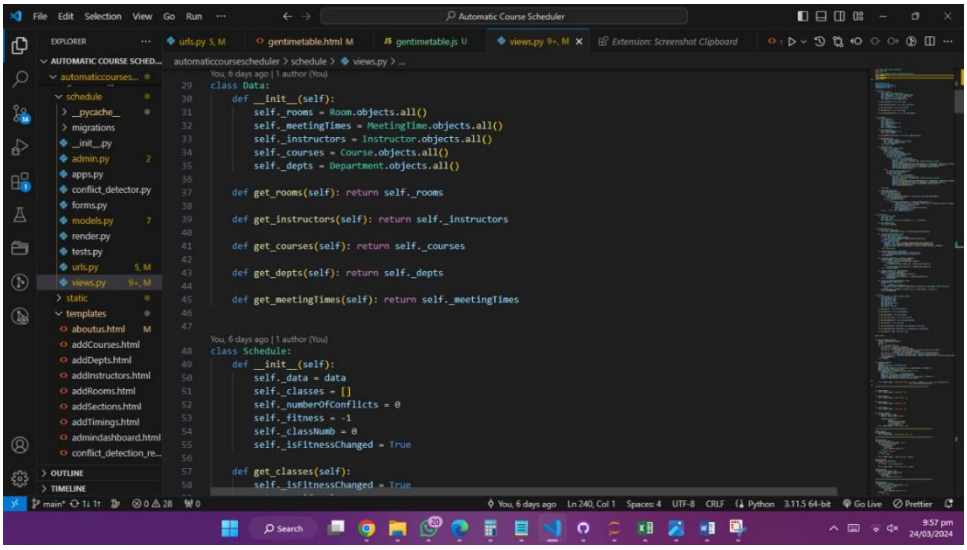
[10]. T. Jain, N. Jamil, Genetic Algorithm Approach to Time Tabling Problem, European Journal of Business and Management , 7(4) 7-11, (2015) Genetic Algorithm as a General Approach to Time Tabling Problem | Jain | European Journal of Business and Management (iiste.org)

APPENDIX:

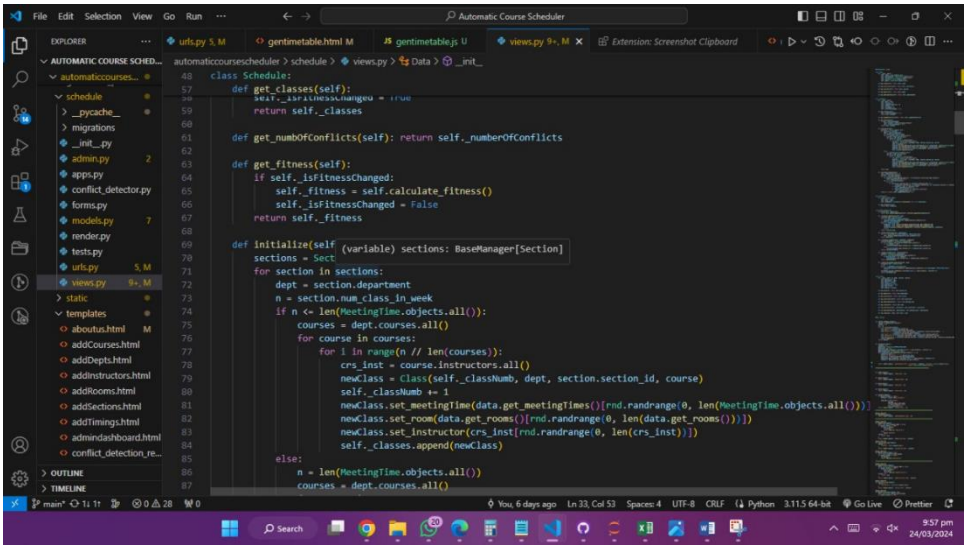
Working Of the Algorithm



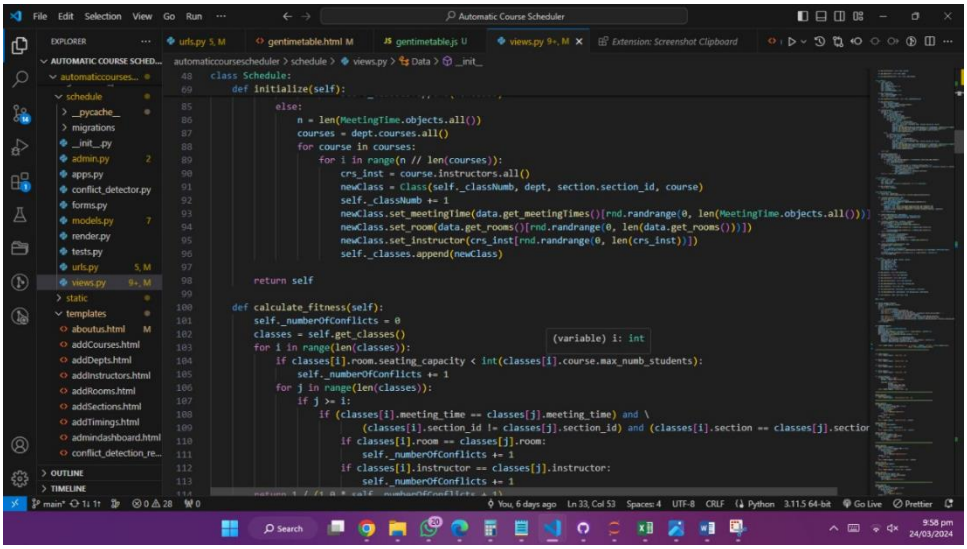
```
from django.http import request
from django.shortcuts import render, redirect
from . forms import *
from .models import *
from django.core.mail import send_mail
from django.conf import settings
from django.contrib.auth.decorators import login_required
from .render import Render
from django.views.generic import View
from django.http import JsonResponse
from django.http import HttpResponse
from django.shortcuts import get_object_or_404
from wsgiref.util import FileWrapper
from .models import GeneratedSchedule
import mimetypes
import os
from .conflict_detector import ScheduleConflictDetector
import pandas as pd
import csv
import messages
import time


POPULATION_SIZE = 9
NUMB_OF_ELITE_SCHEDULES = 1
TOURNAMENT_SELECTION_SIZE = 3
MUTATION_RATE = 0.05

class Data:
    def __init__(self):
```



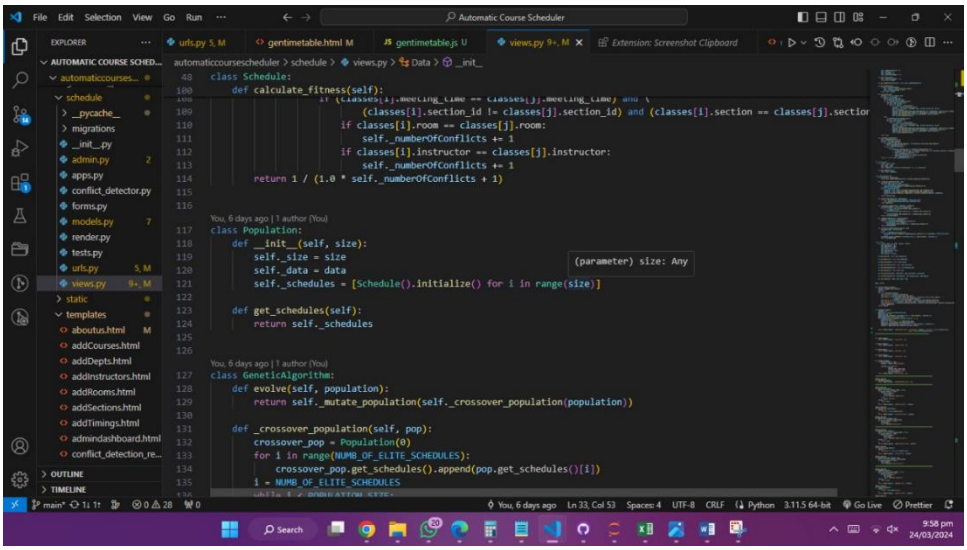
```
class Data:
    def __init__(self):
        self._rooms = Room.objects.all()
        self._meetingTimes = MeetingTime.objects.all()
        self._instructors = Instructor.objects.all()
        self._courses = Course.objects.all()
        self._depts = Department.objects.all()

    def get_rooms(self): return self._rooms

    def get_instructors(self): return self._instructors

    def get_courses(self): return self._courses

    def get_depts(self): return self._depts

    def get_meetingTimes(self): return self._meetingTimes


class Schedule:
    def __init__(self):
        self._data = data
        self._classes = []
        self._numberOfConflicts = 0
        self._fitness = -1
        self._classNumb = 0
        self._isFitnessChanged = True

    def get_classes(self):
        self._isFitnessChanged = True
```



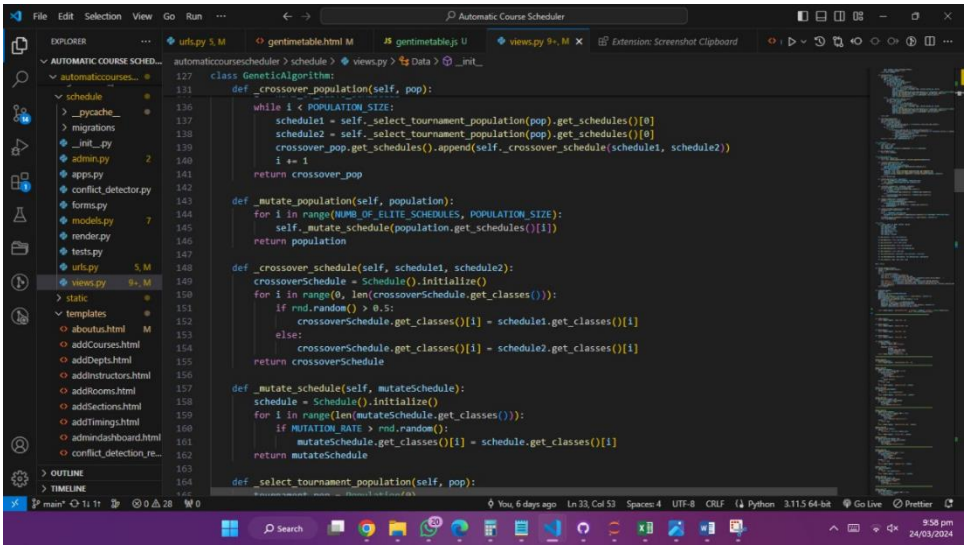
```
class Schedule:
    def get_classes(self):
        return self._classes

    def get_numbOfConflicts(self): return self._numberOfConflicts

    def get_fitness(self):
        if self._isFitnessChanged:
            self._fitness = self.calculate_fitness()
            self._isFitnessChanged = False
        return self._fitness

    def initialize(self (variable) sections: BaseManager[Section]
        sections = Sect
        for section in sections:
            dept = section.department
            n = section.num_class_in_week
            if n <= len(MeetingTime.objects.all()):
                courses = dept.courses.all()
                for course in courses:
                    for i in range(n // len(courses)):
                        crs_inst = course.instructors.all()
                        newClass = Class(self._classNumb, dept, section.section_id, course)
                        self._classNumb += 1
                        newClass.set_meetingTime(data.get_meetingTimes()[rnd.randrange(0, len(MeetingTime.objects.all()))])
                        newClass.set_room(data.get_rooms()[rnd.randrange(0, len(data.get_rooms()))])
                        newClass.set_instructor(crs_inst[rnd.randrange(0, len(crs_inst))])
                        self._classes.append(newClass)
            else:
                n = len(MeetingTime.objects.all())
                courses = dept.courses.all()
```



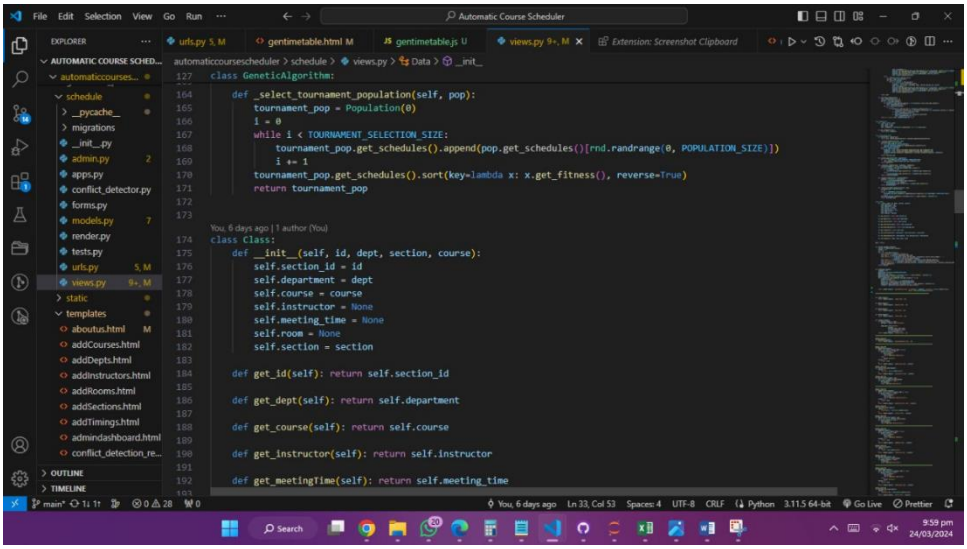
```
class Schedule:
    def initialize(self):
            else:
                n = len(MeetingTime.objects.all())
                courses = dept.courses.all()
                for course in courses:
                    for i in range(n // len(courses)):
                        crs_inst = course.instructors.all()
                        newClass = Class(self._classNumb, dept, section.section_id, course)
                        self._classNumb += 1
                        newClass.set_meetingTime(data.get_meetingTimes()[rnd.randrange(0, len(MeetingTime.objects.all()))])
                        newClass.set_room(data.get_rooms()[rnd.randrange(0, len(data.get_rooms()))])
                        newClass.set_instructor(crs_inst[rnd.randrange(0, len(crs_inst))])
                        self._classes.append(newClass)

        return self

    def calculate_fitness(self):
        self._numberOfConflicts = 0
        classes = self.get_classes()
        for i in range(len(classes)):                    (variable) i: int
            if classes[i].room.seating_capacity < int(classes[i].course.max_numb_students):
                self._numberOfConflicts += 1
            for j in range(len(classes)):
                if j >= i:
                    if (classes[i].meeting_time == classes[j].meeting_time) and \
                            (classes[i].section_id != classes[j].section_id) and (classes[i].section == classes[j].sectior
                        if classes[i].room == classes[j].room:
                            self._numberOfConflicts += 1
                        if classes[i].instructor == classes[j].instructor:
                            self._numberOfConflicts += 1
```



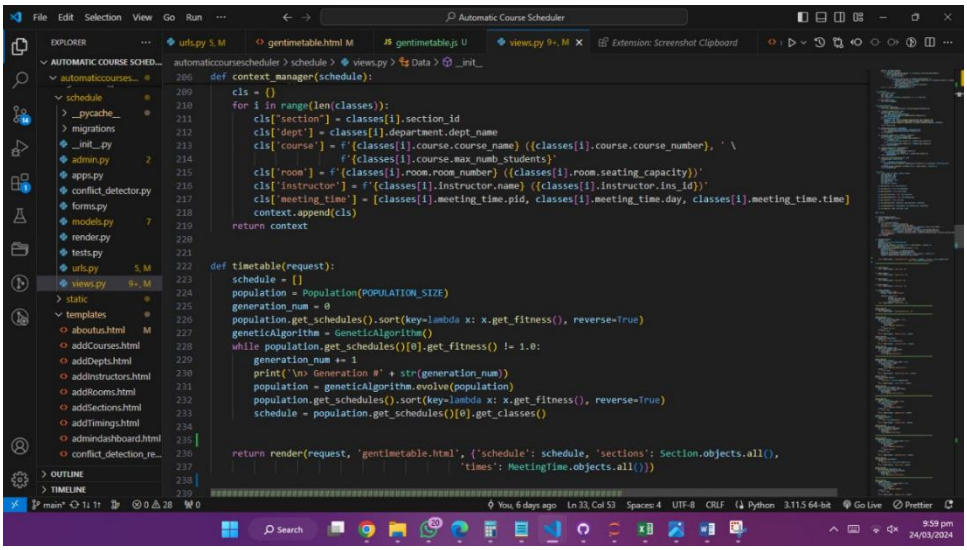
```
class Schedule:
    def calculate_fitness(self):
                    (classes[i].section_id != classes[j].section_id) and (classes[i].section == classes[j].sectior
                if classes[i].room == classes[j].room:
                    self._numberOfConflicts += 1
                if classes[i].instructor == classes[j].instructor:
                    self._numberOfConflicts += 1
        return 1 / (1.0 * self._numberOfConflicts + 1)


You, 6 days ago | 1 author (You)
class Population:
    def __init__(self, size):
        self._size = size                                (parameter) size: Any
        self._data = data
        self._schedules = [Schedule().initialize() for i in range(size)]

    def get_schedules(self):
        return self._schedules


You, 6 days ago | 1 author (You)
class GeneticAlgorithm:
    def evolve(self, population):
        return self._mutate_population(self._crossover_population(population))

    def _crossover_population(self, pop):
        crossover_pop = Population(0)
        for i in range(NUMB_OF_ELITE_SCHEDULES):
            crossover_pop.get_schedules().append(pop.get_schedules()[i])
        i = NUMB_OF_ELITE_SCHEDULES
```



```
class GeneticAlgorithm:
    def _crossover_population(self, pop):
        while i < POPULATION_SIZE:
            schedule1 = self._select_tournament_population(pop).get_schedules()[0]
            schedule2 = self._select_tournament_population(pop).get_schedules()[0]
            crossover_pop.get_schedules().append(self._crossover_schedule(schedule1, schedule2))
            i += 1
        return crossover_pop

    def _mutate_population(self, population):
        for i in range(NUMB_OF_ELITE_SCHEDULES, POPULATION_SIZE):
            self._mutate_schedule(population.get_schedules()[i])
        return population

    def _crossover_schedule(self, schedule1, schedule2):
        crossoverSchedule = Schedule().initialize()
        for i in range(0, len(crossoverSchedule.get_classes())):
            if rnd.random() > 0.5:
                crossoverSchedule.get_classes()[i] = schedule1.get_classes()[i]
            else:
                crossoverSchedule.get_classes()[i] = schedule2.get_classes()[i]
        return crossoverSchedule

    def _mutate_schedule(self, mutateSchedule):
        schedule = Schedule().initialize()
        for i in range(len(mutateSchedule.get_classes())):
            if MUTATION_RATE > rnd.random():
                mutateSchedule.get_classes()[i] = schedule.get_classes()[i]
        return mutateSchedule

    def _select_tournament_population(self, pop):
```

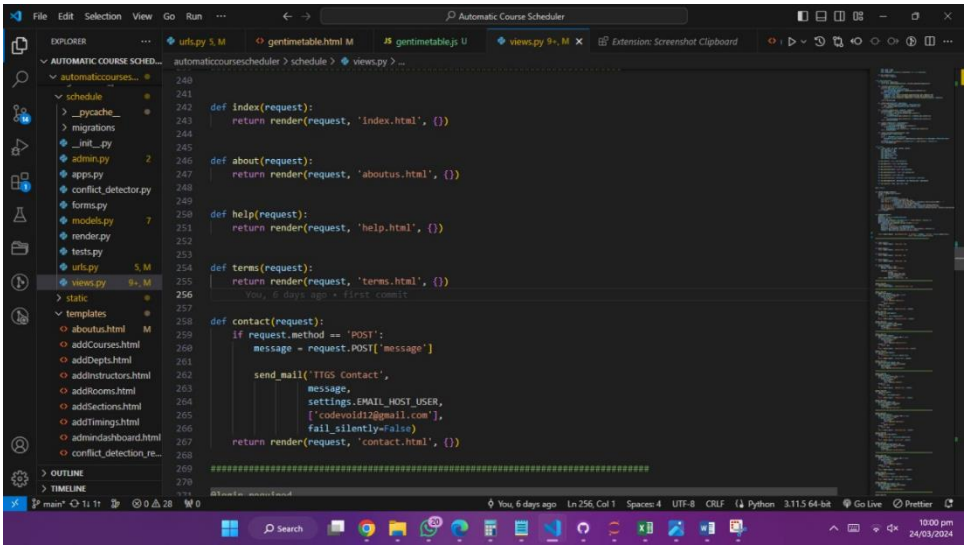


```
    def _select_tournament_population(self, pop):
        tournament_pop = Population(0)
        i = 0
        while i < TOURNAMENT_SELECTION_SIZE:
            tournament_pop.get_schedules().append(pop.get_schedules()[rnd.randrange(0, POPULATION_SIZE)])
            i += 1
        tournament_pop.get_schedules().sort(key=lambda x: x.get_fitness(), reverse=True)
        return tournament_pop


You, 6 days ago | 1 author (You)
class Class:
    def __init__(self, id, dept, section, course):
        self.section_id = id
        self.department = dept
        self.course = course
        self.instructor = None
        self.meeting_time = None
        self.room = None
        self.section = section

    def get_id(self): return self.section_id

    def get_dept(self): return self.department

    def get_course(self): return self.course

    def get_instructor(self): return self.instructor

    def get_meetingTime(self): return self.meeting_time
```



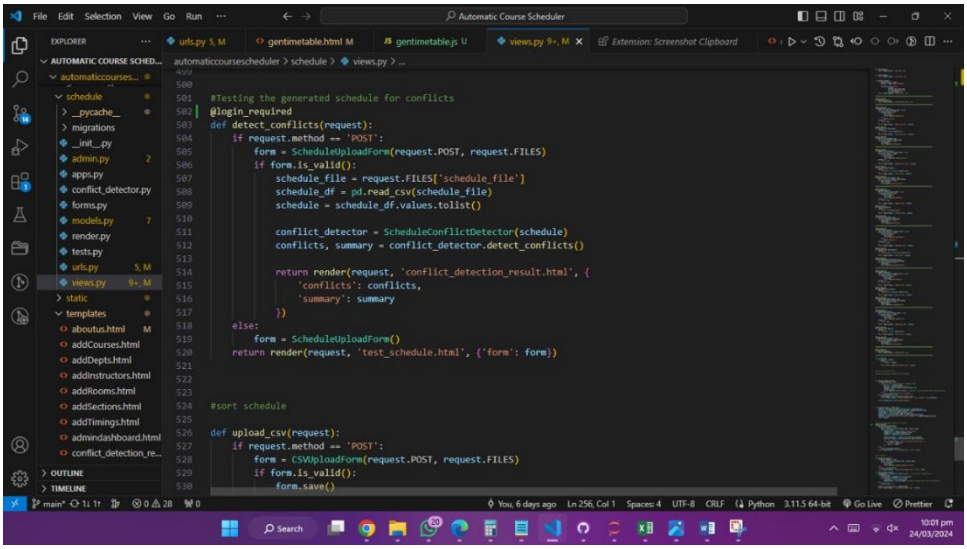
```
def context_manager(schedule):
    cls = {}
    for i in range(len(classes)):
        cls["section"] = classes[i].section_id
        cls['dept'] = classes[i].department.dept_name
        cls['course'] = f'{classes[i].course.course_name} ({classes[i].course.course_number}, ' \
                        f'{classes[i].course.max_numb_students}'
        cls['room'] = f'{classes[i].room.room_number} ({classes[i].room.seating_capacity})'
        cls['instructor'] = f'{classes[i].instructor.name} ({classes[i].instructor.ins_id})'
        cls['meeting_time'] = [classes[i].meeting_time.pid, classes[i].meeting_time.day, classes[i].meeting_time.time]
        context.append(cls)
    return context


def timetable(request):
    schedule = []
    population = Population(POPULATION_SIZE)
    generation_num = 0
    population.get_schedules().sort(key=lambda x: x.get_fitness(), reverse=True)
    geneticAlgorithm = GeneticAlgorithm()
    while population.get_schedules()[0].get_fitness() != 1.0:
        generation_num += 1
        print('\n> Generation #' + str(generation_num))
        population = geneticAlgorithm.evolve(population)
        population.get_schedules().sort(key=lambda x: x.get_fitness(), reverse=True)
        schedule = population.get_schedules()[0].get_classes()


    return render(request, 'gentimetable.html', {'schedule': schedule, 'sections': Section.objects.all(),
                                                 'times': MeetingTime.objects.all()})
```

Views For Pages:



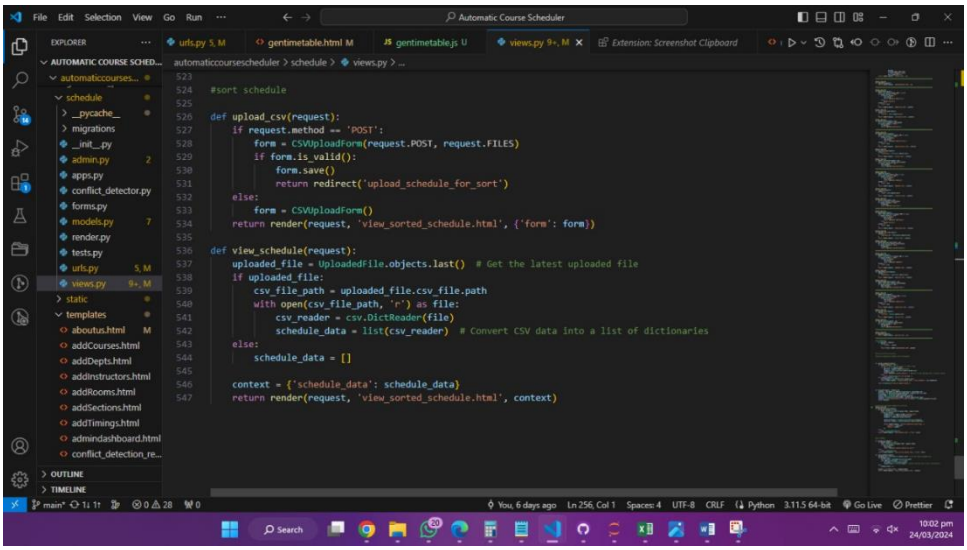
```
def index(request):
    return render(request, 'index.html', {})


def about(request):
    return render(request, 'aboutus.html', {})


def help(request):
    return render(request, 'help.html', {})


def terms(request):
    return render(request, 'terms.html', {})
      You, 6 days ago • first commit


def contact(request):
    if request.method == 'POST':
        message = request.POST['message']

        send_mail('TTGS Contact',
                  message,
                  settings.EMAIL_HOST_USER,
                  ['codevoid12@gmail.com'],
                  fail_silently=False)
    return render(request, 'contact.html', {})

#########################################################################
```

Conflict Recognizing Module:



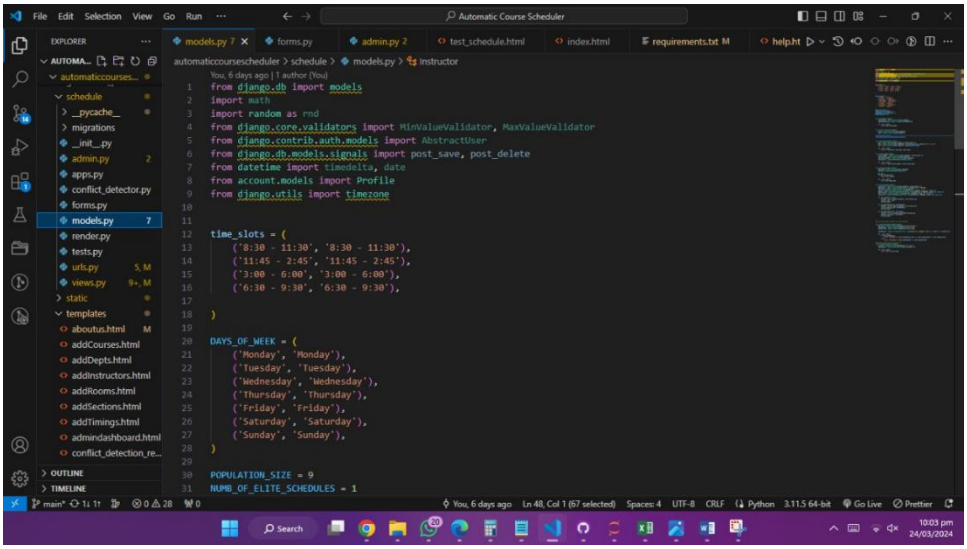
```
#Testing the generated schedule for conflicts
@login_required
def detect_conflicts(request):
    if request.method == 'POST':
        form = ScheduleUploadForm(request.POST, request.FILES)
        if form.is_valid():
            schedule_file = request.FILES['schedule_file']
            schedule_df = pd.read_csv(schedule_file)
            schedule = schedule_df.values.tolist()

            conflict_detector = ScheduleConflictDetector(schedule)
            conflicts, summary = conflict_detector.detect_conflicts()

            return render(request, 'conflict_detection_result.html', {
                'conflicts': conflicts,
                'summary': summary
            })
    else:
        form = ScheduleUploadForm()
    return render(request, 'test_schedule.html', {'form': form})



#sort schedule

def upload_csv(request):
    if request.method == 'POST':
        form = CSVUploadForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
```

View Generated Schedule:



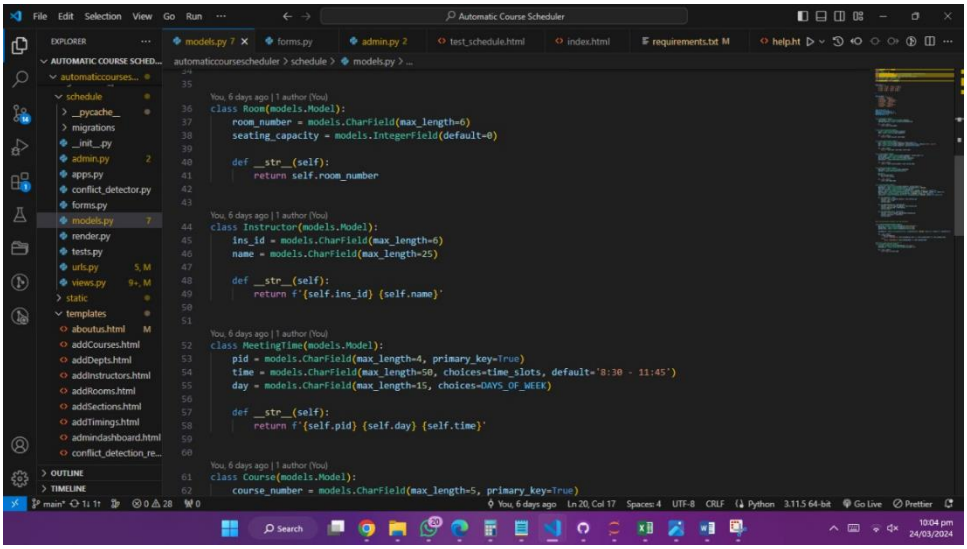
```
#sort schedule

def upload_csv(request):
    if request.method == 'POST':
        form = CSVUploadForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('upload_schedule_for_sort')
    else:
        form = CSVUploadForm()
    return render(request, 'view_sorted_schedule.html', {'form': form})

def view_schedule(request):
    uploaded_file = UploadedFile.objects.last()  # Get the latest uploaded file
    if uploaded_file:
        csv_file_path = uploaded_file.csv_file.path
        with open(csv_file_path, 'r') as file:
            csv_reader = csv.DictReader(file)
            schedule_data = list(csv_reader)  # Convert CSV data into a list of dictionaries
    else:
        schedule_data = []

    context = {'schedule_data': schedule_data}
    return render(request, 'view_sorted_schedule.html', context)
```

Database Model 1:



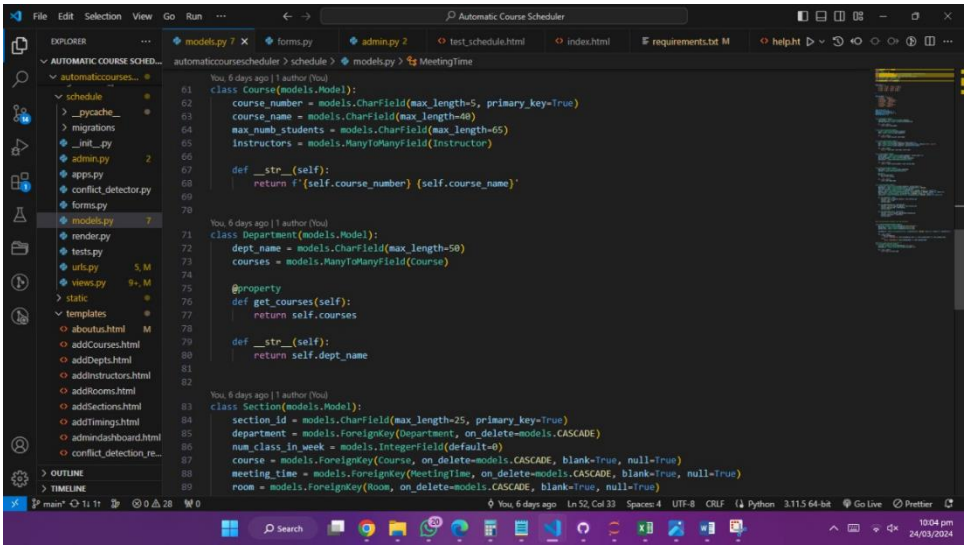
```
from django.db import models
import math
import random as rnd
from django.core.validators import MinValueValidator, MaxValueValidator
from django.contrib.auth.models import AbstractUser
from django.db.models.signals import post_save, post_delete
from datetime import timedelta, date
from account.models import Profile
from django.utils import timezone


time_slots = (
    ('8:30 - 11:30', '8:30 - 11:30'),
    ('11:45 - 2:45', '11:45 - 2:45'),
    ('3:00 - 6:00', '3:00 - 6:00'),
    ('6:30 - 9:30', '6:30 - 9:30'),
)

DAYS_OF_WEEK = (
    ('Monday', 'Monday'),
    ('Tuesday', 'Tuesday'),
    ('Wednesday', 'Wednesday'),
    ('Thursday', 'Thursday'),
    ('Friday', 'Friday'),
    ('Saturday', 'Saturday'),
    ('Sunday', 'Sunday'),
)

POPULATION_SIZE = 9
NUMB_OF_ELITE_SCHEDULES = 1
```

Database Model 2:



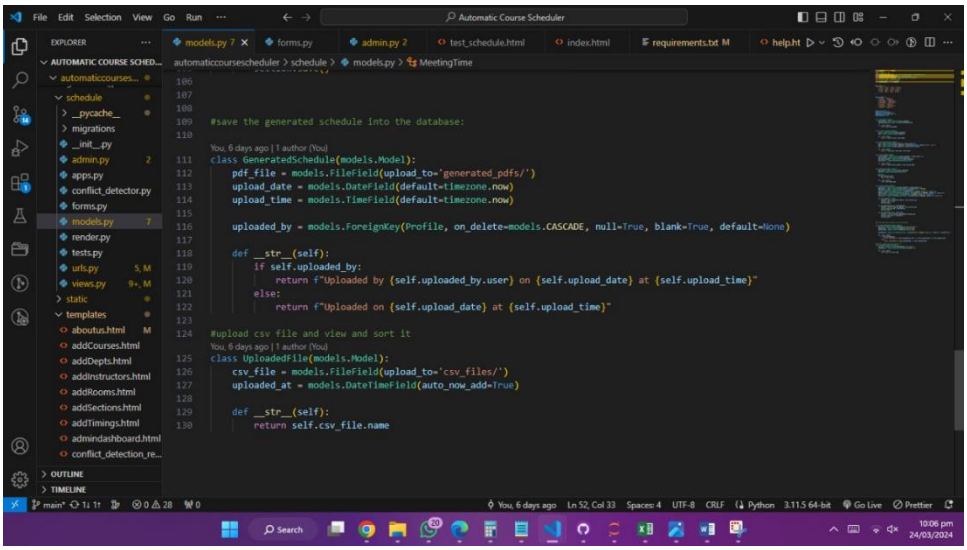
```
class Room(models.Model):
    room_number = models.CharField(max_length=6)
    seating_capacity = models.IntegerField(default=0)

    def __str__(self):
        return self.room_number


class Instructor(models.Model):
    ins_id = models.CharField(max_length=6)
    name = models.CharField(max_length=25)

    def __str__(self):
        return f'{self.ins_id} {self.name}'


class MeetingTime(models.Model):
    pid = models.CharField(max_length=4, primary_key=True)
    time = models.CharField(max_length=50, choices=time_slots, default='8:30 - 11:45')
    day = models.CharField(max_length=15, choices=DAYS_OF_WEEK)

    def __str__(self):
        return f'{self.pid} {self.day} {self.time}'


class Course(models.Model):
    course_number = models.CharField(max_length=5, primary_key=True)
```

Database Model 3:

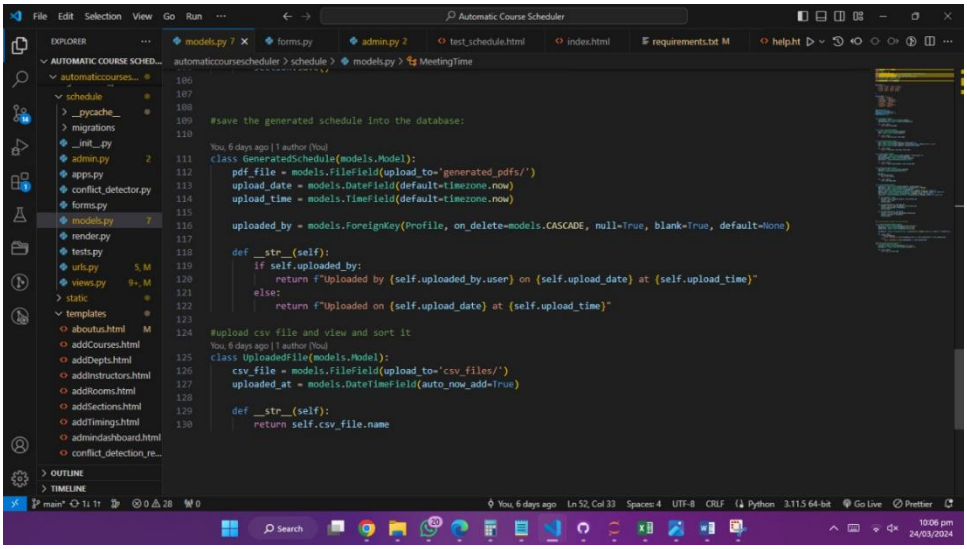


```
class Course(models.Model):
    course_number = models.CharField(max_length=5, primary_key=True)
    course_name = models.CharField(max_length=40)
    max_numb_students = models.CharField(max_length=65)
    instructors = models.ManyToManyField(Instructor)

    def __str__(self):
        return f'{self.course_number} {self.course_name}'


class Department(models.Model):
    dept_name = models.CharField(max_length=50)
    courses = models.ManyToManyField(Course)

    @property
    def get_courses(self):
        return self.courses

    def __str__(self):
        return self.dept_name


class Section(models.Model):
    section_id = models.CharField(max_length=25, primary_key=True)
    department = models.ForeignKey(Department, on_delete=models.CASCADE)
    num_class_in_week = models.IntegerField(default=0)
    course = models.ForeignKey(Course, on_delete=models.CASCADE, blank=True, null=True)
    meeting_time = models.ForeignKey(MeetingTime, on_delete=models.CASCADE, blank=True, null=True)
    room = models.ForeignKey(Room, on_delete=models.CASCADE, blank=True, null=True)
```

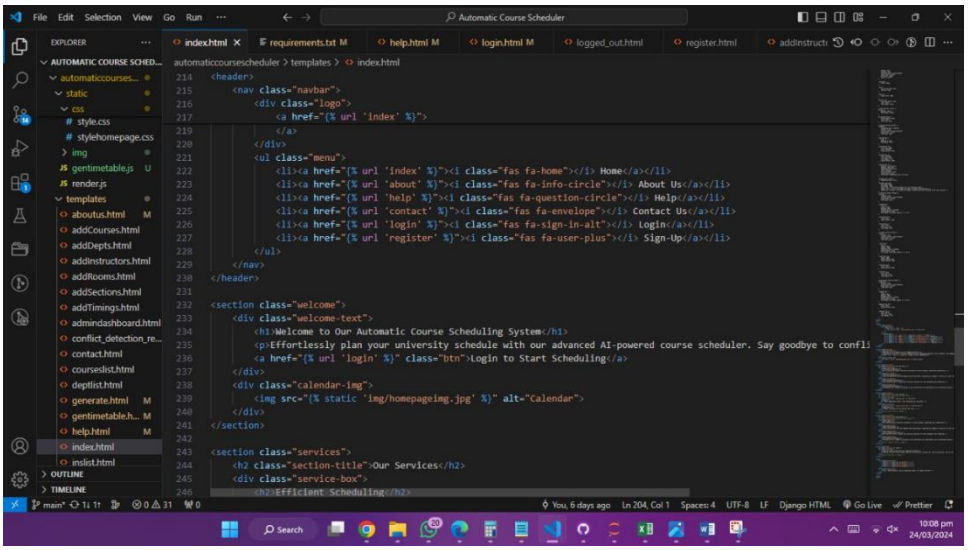
Database Model 4:



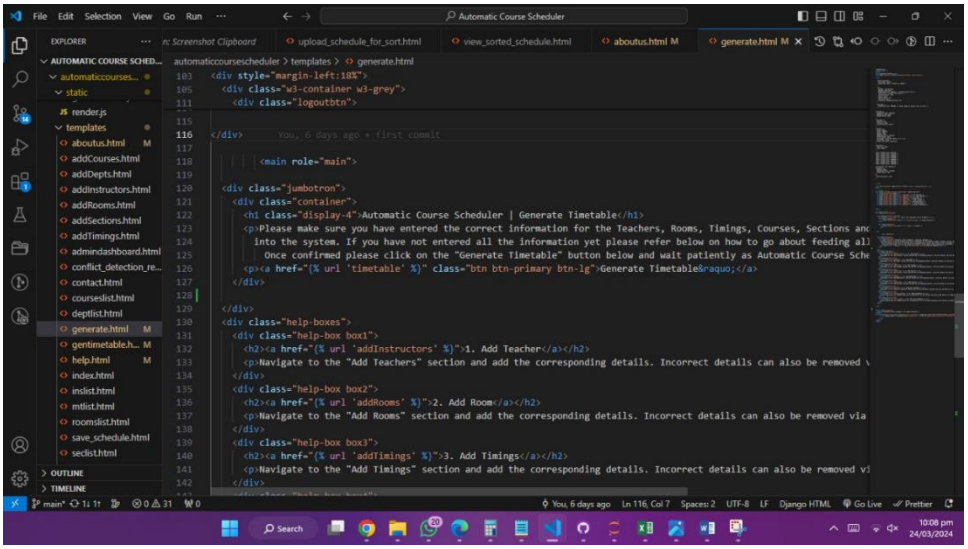
CSS Design:



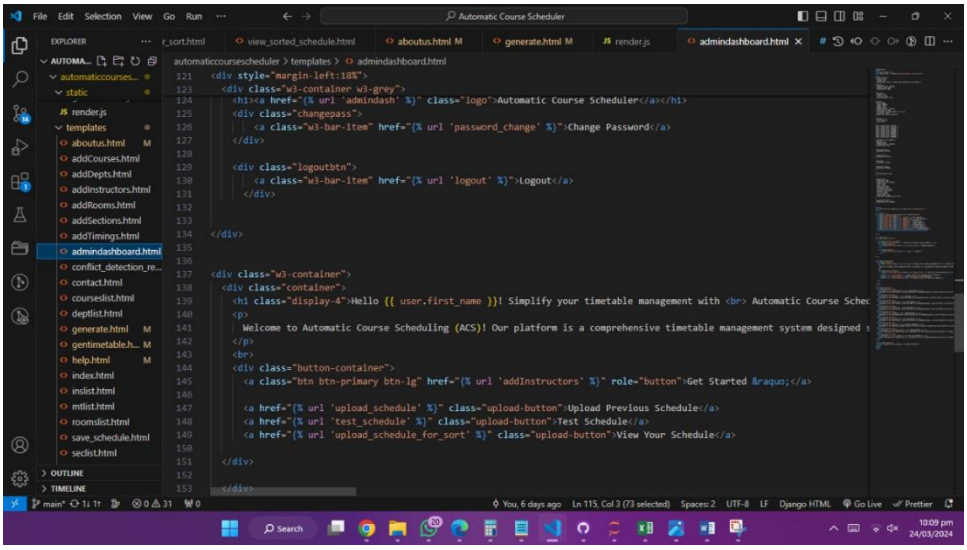
Homepage:



Schedule Generation Age:



Admin Dashboard:



Generated Schedule Page: