

AUTOMATIC TIMETABLE GENERATOR



A DESIGN PROJECT REPORT

submitted by

SANJAY N

SANTHOSH J

VIGNESHWARAN R

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

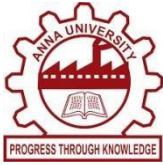
COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

DECEMBER, 2024



AUTOMATIC TIMETABLE GENERATOR

A DESIGN PROJECT REPORT

submitted by

SANJAY N (811722104130)

SANTHOSH J (811722104133)

VIGNESHWARAN R (811722104181)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

DECEMBER, 2024

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM-621112

BONAFIDE CERTIFICATE

Certified that this project report titled **“AUTOMATIC TIMETABLE GENERATOR”** is bonafide work of **SANJAY N (811722104130), SANTHOSH J (811722104133), VIGNESHWARAN R (811722104181)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr A Delphin Carolina Rani M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram-621 112

SIGNATURE

Mrs G Rajendra Kannammal,M.E.,

SUPERVISOR

Assistant Professor

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram-621 112

Submitted for the viva-voice examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We jointly declare that the project report on “**AUTOMATIC TIMETABLE GENERATOR**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of Bachelor Of Engineering. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of Bachelor Of Engineering.

Signature

SANJAY N

SANTHOSH J

VIGNESHWARAN R

Place:Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in debtness to our institution “**K RAMAKRISHNAN COLLEGE OF TECHNOLOGY**”, for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project with full satisfaction.

We whole heartily thank **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Mrs. G RAJENDRA KANNAMMAL, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his in calculable suggestions, creativity, assistance and patience which motivated us to carry our this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

The Automatic Timetable Generator is an innovative Java-based application developed to streamline and automate the scheduling process for educational institutions. Built using Java layouts such as JFrame and Swing, the system utilizes the Greedy Randomized Scheduling Algorithm to assign courses, professors, and time slots efficiently, ensuring conflict-free and optimized timetables. Its modular design includes features to create and manage batches, courses, and professors, along with dynamic options to view and delete entries. The intuitive interface enables administrators to navigate seamlessly and generate schedules without relying on external databases, enhancing operational efficiency. This project addresses the challenges of manual scheduling by offering a user-friendly solution that minimizes errors and administrative workload. The absence of database integration makes it lightweight, while the algorithm ensures flexibility and adaptability to institutional requirements. This documentation details the methodology, technical specifications, and user interface design, highlighting the potential for future enhancements, such as database support and export functionality, making it a transformative tool in timetable management.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
	1.1 Background	1
	1.2 Overview	2
	1.3 Problem Statement	2
	1.4 Objective	3
	1.5 Implication	3
2	LITERATURE SURVEY	4
3	SYSTEM ANALYSIS	7
	3.1 Existing System	7
	3.2 Proposed System	8
	3.3 Block Diagram for Proposed System	9
	3.4 Process Cycle	9
	3.5 Flowchart	10
	3.6 Activity Diagram	11

4	MODULES	12
	4.1 Module Description	12
	4.1.1 View Page Module	12
	4.1.1.1 Functionality And Workflow	13
	4.1.1.2 User Interface Design	14
	4.1.1.3 Algorithm Integration	14
	4.2 Create Batch Module	15
	4.2.1 Batch Creation	15
	4.2.2 Course Assignment	16
	4.3 Create Course Module	16
	4.4 Create Professor Module	18
	4.4.1 Assign Professor To Course	20
	4.5 Delete Module	21
	4.5.1 Delete Course	22
	4.5.2 Delete Professor	23
	4.5.3 Delete Batch	24
5	SYSTEM SPECIFICATION	26
	5.1 Hardware Requirements	26
	5.2 Software Requirements	26
	5.2.1 Programming Language: JAVA	26
	5.2.2 Development Environment	27
	5.2.3 Java Libraries	28
	5.2.3.1 JFrame	29

5.2.3.2 Swing	30
5.2.4 Java Runtime Environment	31
6 METHODOLOGY	32
6.1 Algorithm Methodology	32
6.1.1 Algorithm Steps	33
6.2 User Interaction Mechanism	34
6.3 System Architecture and Integration	35
6.4 Development Environment and Tools	36
7 CONCLUSION AND FUTURE ENHANCEMENT	37
7.1 Conclusion	37
7.2 Future Enhancement	38
APPENDIX-1	39
APPENDIX-2	51
REFERENCES	53

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.1	Flow of Timetable Generation	1
3.1	Usecase diagram	9
3.2	Process cycle	9
3.3	Flow of Control	10
3.4	Action Sequence structure of Timetable Generator	11
4.1	View Page	13
4.2	Create Batch Page	16
4.3	Create Course Page	17
4.4	Create Professor Page	19
4.5	Delete Page	21

APPENDIX - 2

2.1	Execution of Code	51
2.2	Generated Timetable	51
2.3	Generated Timetable for Professors	52

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
GUI	Graphical User Interface
JRE	Java Runtime Environment
JVM	Java Virtual Machine
IDE	Integrated Development Environment
HCI	Human-Computer Interaction
CRUD	Create, Read, Update, Delete
RGB	Red, Green, Blue (Color Model)
IoT	Internet of Things
AI	Artificial Intelligence
VS Code	Visual Studio Code
API	Application Programming Interface
SWING	Standard Widget Toolkit in Java
MVC	Model-View-Controller (Design Pattern)
UI	User Interface
OOP	Object-Oriented Programming
RAM	Random Access Memory
CPU	Central Processing Unit
UML	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Timetable management is a critical yet challenging task for educational institutions. It involves coordinating multiple factors, including teacher availability, student schedules, classroom capacity, and institutional regulations. The manual process of timetable creation is often labour-intensive, time-consuming, and susceptible to errors. These errors may include scheduling conflicts, inefficient use of resources, or failure to meet specific requirements, such as maintaining teacher workload balance or ensuring students have manageable schedules. As institutions grow and educational programs become more diverse, the complexity of creating accurate and efficient timetables increases exponentially. Traditional methods struggle to accommodate dynamic scenarios, such as last-minute changes in staff availability or room allocations. These challenges underline the necessity for an automated solution that simplifies the process while maintaining flexibility and accuracy. The Automatic Timetable Generator project is designed to address these challenges by leveraging advancements in computational techniques and algorithmic design. Developed in Java, the application introduces a systematic approach to timetable generation using a Greedy Randomized Scheduling Algorithm. This algorithm employs a combination of greedy principles—assigning available slots optimally—and randomization to introduce diversity and reduce the likelihood of repetitive or rigid schedules.

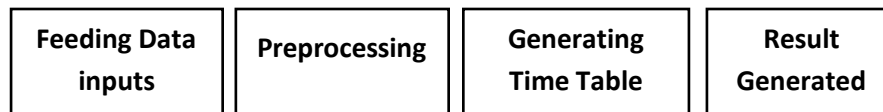


Figure 1.1: Flow of Time Table Generation

1.2 OVERVIEW

Timetable management in educational institutions is a complex process that involves allocating resources such as teachers, classrooms, and time slots efficiently while avoiding conflicts. The Automatic Timetable Generator addresses these challenges by automating the scheduling process, ensuring accuracy, and saving valuable administrative time. Developed using Java, the system leverages a Greedy Randomized Scheduling Algorithm to create flexible and efficient timetables. The algorithm combines a greedy approach for quick assignment of slots with randomization to produce diverse and conflict-free schedules. This project aims to alleviate the challenges of manual scheduling, which is prone to errors, overlaps, and inefficiencies. By automating the process, the system minimizes human effort while adhering to institutional constraints such as teacher availability, course priorities, and classroom capacity. The Automatic Timetable Generator is equipped with a user-friendly interface, making it accessible for administrators to input data, review generated timetables, and export the results. The tool is adaptable, capable of accommodating last-minute changes and updates without requiring a complete rescheduling effort.

1.3 PROBLEM STATEMENT

Creating timetables for educational institutions is a complex and time-consuming task that involves coordinating various elements, such as teacher availability, classr from scratch. These challenges underscore the need for an automated solution that can handle the complexities of timetable creation, ensuring accuracy, improving resource utilization, and reducing the overall time and effort required for scheduling. The Automatic Timetable Generator addresses these issues by providing a reliable, efficient, and flexible solution for generating conflict-free timetables with minimal human intervention

1.4 OBJECTIVE

The Automatic Timetable Generator aims to automate the creation of conflict-free timetables for educational institutions. The system utilizes a Greedy Randomized Scheduling Algorithm to efficiently allocate teachers, classrooms, and courses while meeting institutional constraints. Its key objectives are to reduce the time and effort involved in manual scheduling, eliminate errors, and improve resource utilization. The tool provides an easy-to-use interface for administrators to input data, generate schedules, and make adjustments as needed. Ultimately, the project seeks to enhance scheduling efficiency, ensure flexibility in adapting to changes, and minimize administrative workload.

1.5 IMPLICATION

The Automatic Timetable Generator has significant implications for educational institutions by streamlining the scheduling process, reducing administrative workload, and minimizing human error. It ensures efficient resource utilization, optimizing the use of classrooms and teaching staff. The system's adaptability allows for quick adjustments to accommodate unforeseen changes, such as teacher absences or course modifications. By automating the scheduling process, the tool enhances operational efficiency, allowing administrators to focus on other critical tasks. This solution also promotes fairness in workload distribution, ensuring that teachers are not overburdened while maintaining an equitable timetable for all stakeholders. When done manually, this process is prone to errors, including overlapping schedules, inefficient use of classrooms, and imbalanced workloads for instructors. As the size and complexity of the institution grow, the manual scheduling process becomes increasingly difficult to manage, leading to inefficiencies and increased administrative burden. Additionally, modifying existing schedules to accommodate changes, such as teacher absences or classroom reallocations, often requires starting the process

CHAPTER 2

LITERATURE SURVEY

1 . TITLE : Survey Paper on Automatic Timetable Generator

**AUTHOR : Ankit Pounikar, Hrushikesh Bhandage, Nupur Dalvi, Tanvi Borade,
S. H. Lokhande**

YEAR : 2023

In today's literate world it is very difficult to create time table manually. Timetables are to be created uniquely for all branches and years respectively. It becomes a very hectic, time consuming, and needs manpower for preparing the timetables manually. In some cases, this process becomes complex when any staff is on a leave or needs to be substituted. It would be convenient if an algorithm creates timetable which will save a lot of time and reduce the load and pressure on the person doing the job. Using software to do the job saves a lot of time and can also create timetables for complex situations. It will also avoid any human error like: subject clash, vacant slots.

2 . TITLE : AUTO TIMETABLE GENERATOR

AUTHORS : Mrunal Gaikwad, Adwait Gaikwad, Mukesh Chaudhary, Dhanshree

YEAR : 2022

The program we are proposing is aimed at an intelligent auto generate planning system specific to the field of education. In the construction of an accurate and high-quality timetable there are barriers that need to be allowed namely access to classrooms, students, teachers, courses, time spaces etc. These are annoying factors that contribute to the challenges of similar production. Based on the uploaded information, the system will generate a class schedule automatically with customized configuration for each user.

3 . TITLE : Timetable Generation System

AUTHOR : Gaurav Kumar, Aishwarya Raut, Apeksha Patel

YEAR : 2021

In colleges after admission lots of changes happens regarding teachers, subjects, rules, timing and most important time table. Making of new Time table for different timings and new teachers is very difficult work which increases the human work load. If we have an automated system which provide us a time table by providing the information of teachers, subjects etc. will help us to make automated timetable. Proposed system of our project will help to generate it automatically and also it is time saving. It prevents the complexity of setting and managing Timetable manually. we are going to use genetic algorithm in our project to reduce these difficulties of generating timetable. Various inputs like number of subjects, teachers, workload of a teacher, semester, priority of subject will be given to this automated system. With the help of these inputs, it will generate possible time tables for working days of the week for teachers. For online assignment submission of students, by using online method will enable faster and easier submission for students as well as teachers. The virtual queue will be there instead of physical queues. It will ensure an effective management of queues.

4 . TITLE : A STUDY ON AUTOMATIC TIMETABLE GENERATOR

AUTHOR : Parkavi A

YEAR : 2018

Time table generation is tedious job for educationalist with respect to time and man power. Providing a automatic time table generator will help to generate time table automatically. Proposed system of our project will help to generate it automatically also helps to save time. It avoids the complexity of setting and managing Timetable manually. In our project we are going to use algorithms like genetic, heuristic, resource scheduling to reduce these difficulties of generating timetable. These algorithms incorporate a numeral of strategy, aimed to improve the operativeness of the search operation. The system will take various inputs like number of subjects, teachers, workload of a teacher, semester, priority of subject. By relying on these

inputs, it will generate possible time tables for working days of the week for teaching faculty. This will integrate by making optimal use of all resources in a way that will best suit the constraints

5 . TITLE : Timetable generation algorithms based on Genetic Algorithm

AUTHORS : Anisha Jain, Ganapathy S C Aiyer, Harshita Goel

YEAR : 2015

The problem of timetable scheduling is described as a highly constrained NP-hard problem. It is known as the timetabling problem by most researchers. A lot of complex constraints need to be addressed for development of an efficient algorithm to solve this problem. In this paper, we present a comparison among the different techniques that have been developed for timetable generation using Genetic Algorithm and heuristic algorithm.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The traditional method of timetable creation in educational institutions and organizations is predominantly manual. Administrators and staff spend hours or even days compiling data, considering various constraints, and attempting to create schedules that avoid conflicts. While this process has been in use for decades, it is not without its shortcomings. Below are some key characteristics of the existing system:

MANUAL INPUT AND SCHEDULING:

Schedules are typically created using tools like spreadsheets or pen and paper. This process involves manually cross-referencing teacher availability, classroom capacities, and student group requirements.

STATIC CONSTRAINTS HANDLING:

While some software tools may assist with simple scheduling, they cannot adapt to dynamic constraints, such as last-minute changes.

LIMITED OPTIMIZATION:

The manual process often results in suboptimal timetables that do not maximize resource utilization or consider preferences effectively.

ERROR-PRONE NATURE:

Human errors, such as overlapping classes, resource allocation conflicts, or forgotten constraints, are common in manually created timetables.

3.2 PROPOSED SYSTEM

The Automatic Timetable Generator is a Java-based solution designed to automate the creation of timetables for educational institutions. The proposed system leverages a Greedy Randomized Scheduling Algorithm to efficiently allocate resources such as teachers, classrooms, and time slots while ensuring conflict-free schedules. By addressing the limitations of manual scheduling methods, the system simplifies the process and reduces administrative workload.

The system operates by allowing administrators to input essential data, including teacher availability, course details, and institutional constraints. The algorithm then processes this data to generate a timetable that meets predefined conditions, ensuring optimal resource utilization and balanced workloads for staff. The randomness introduced in the scheduling process ensures flexibility and diversity, avoiding repetitive patterns in generated timetables.

Key features of the proposed system include:

- **Automated Slot Assignment:** Efficiently assigns courses, teachers, and rooms to available time slots.
- **Conflict Resolution:** Identifies and eliminates scheduling conflicts dynamically.
- **Flexibility and Adaptability:** Allows for updates and changes without requiring a complete rescheduling effort.
- **User-Friendly Interface:** Simplifies data entry, timetable generation, and result review.

By providing an intuitive, reliable, and efficient solution, the proposed system aims to improve the overall process of timetable management, ensuring accuracy, scalability, and adaptability in diverse educational settings.

3.3 BLOCK DIAGRAM OF PROPOSED SYSTEM

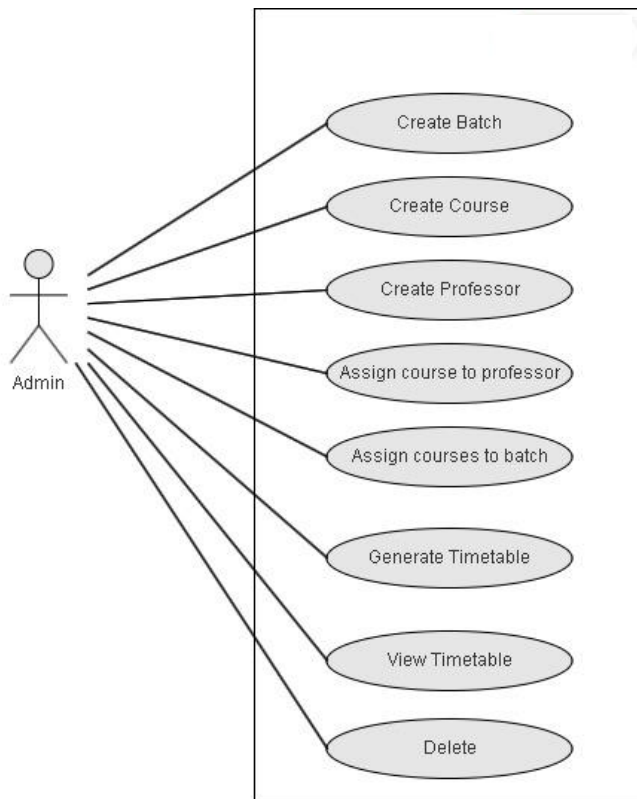


Figure 3.1:Usecase Diagram

3.4 PROCESS CYCLE OF PROPOSED SYSTEM

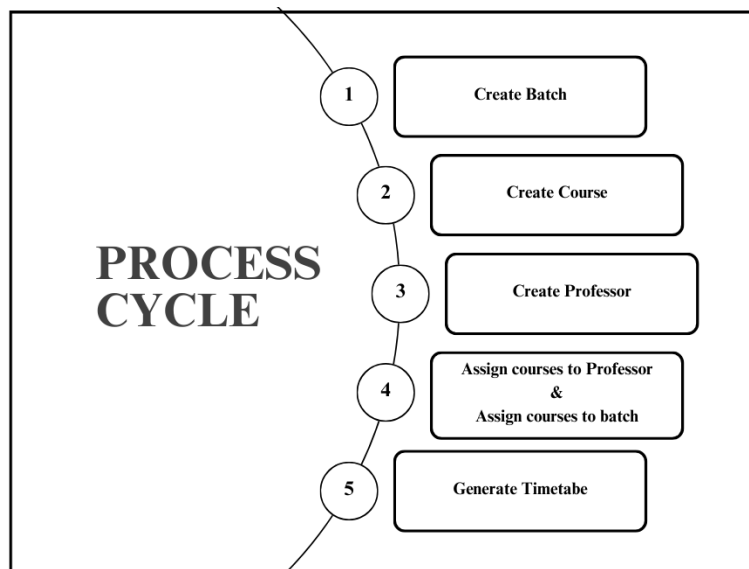


Figure 3.2: Process cycle

3.5 FLOWCHART

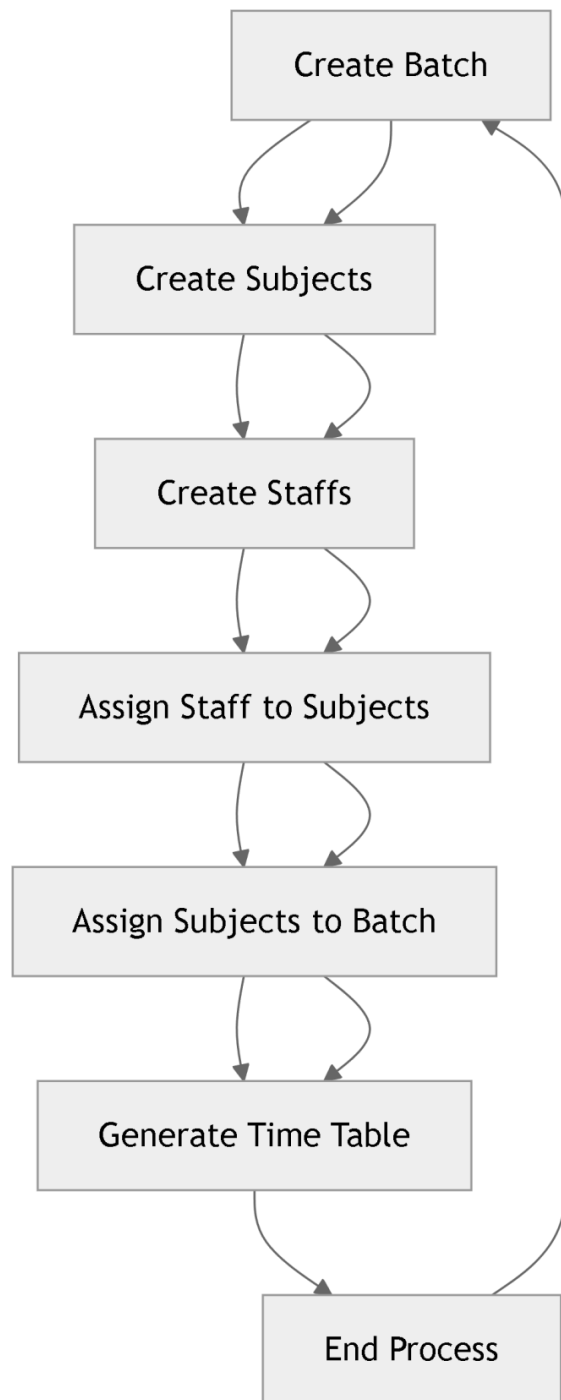


Figure 3.3: Flow of Control

3.6 ACTIVITY DIAGRAM

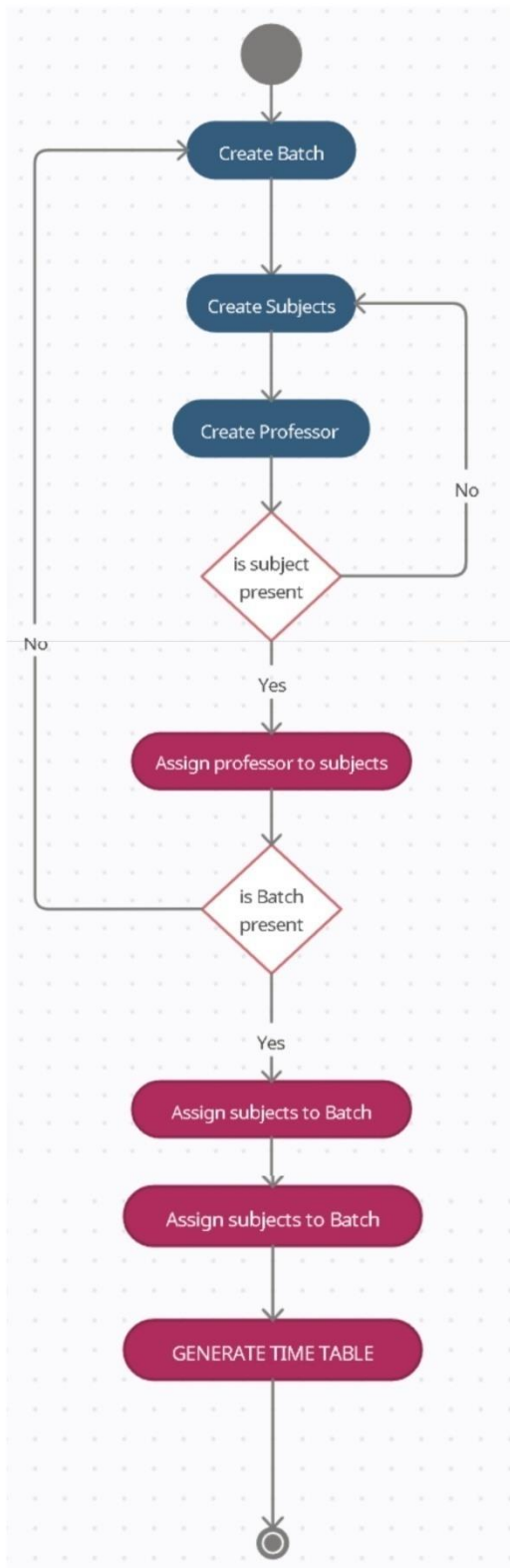


Figure 3.4: Action Sequence Structure of Timetable Generator

CHAPTER 4

MODULES

4.1 MODULE DESCRIPTION

The Automatic Timetable Generator consists of five main modules, each designed to handle specific functionalities essential for efficient timetable creation and management. Below is a description of each module:

1. View Page Module
2. Create Batch Module
3. Create Course Module
4. Create Professor Module
5. Delete Module

4.1.1 VIEW PAGE MODULE

The View Page Module is a critical component of the Automatic Timetable Generator project, serving as the primary interface for users to generate and view timetables efficiently. This module is thoughtfully designed to provide an intuitive, user-friendly, and interactive platform that caters to the scheduling needs of administrators, professors, and other stakeholders. It allows users to seamlessly generate comprehensive timetables for specific batches or individual professors, ensuring that schedules are clear, well-organized, and free from conflicts.

The module eliminates the manual effort traditionally required in timetable creation, leveraging automation to provide accurate and efficient results. By simply selecting the desired criteria—such as a batch or professor name—users can instantly access a detailed timetable that includes information such as courses, time slots, and assigned professors. This functionality streamlines schedule management, reduces errors, and enhances productivity.

In addition to its core functionality, the View Page Module serves as a hub for real-time timetable updates. Any changes made in other modules, such as the creation or deletion of batches, courses, or professors, are dynamically reflected on the view page. This ensures that

users always have access to the most up-to-date schedules without the need for manual updates. Overall, this module plays a pivotal role in maintaining the accuracy, flexibility, and usability of the timetable generation system.

	1	2	3	4	5	6	7	8
Monday	TOC(RAJAVARMA...	MCAD(SOWMIYA)		AI(SARAVANAN)	LIB(SOWMIYA)		PE(KARTHIKA)	
Tuesday	PE(KARTHIKA)	MCAD(SOWMIYA)	DM(MARIA)			TOC(RAJAVARMA...	DM(MARIA)	TOC(RAJAVARMA...
Wednesday		PE(KARTHIKA)	DM(MARIA)	AI(SARAVANAN)	PE(KARTHIKA)			OOAD(VIGNESH)
Thursday		FREE HOUR(-)		DM(MARIA)	DM(MARIA)			OOAD(VIGNESH)
Friday	OOAD(VIGNESH)		MCAD(SOWMIYA)					MCAD(SOWMIYA)
Saturday	DM(MARIA)		TOC(RAJAVARMA...	AI(SARAVANAN)	OOAD(VIGNESH)	TOC(RAJAVARMA...		

Figure 4.1: View Page

4.1.1.1 FUNCTIONALITY AND WORKFLOW

1. Batch Timetable Generation

The primary function of the View Page Module is to generate and display the timetable for a selected batch. Administrators can choose a batch from a dropdown menu or search bar, and the system generates the corresponding timetable.

- **User Input:** The user selects the desired batch from a predefined list of batches.
- **Timetable Display:** The system dynamically displays a structured timetable that includes course details, assigned professors, and the specific time slots for each course.
- **Conflict-Free Scheduling:** The underlying algorithm ensures that no overlapping or conflicting schedules are displayed for a batch.

2. Professor Timetable Viewing

This feature allows users to view the schedule for individual professors. By selecting a professor's name, the system retrieves and displays their assigned courses, along with the respective time slots. This functionality ensures that professors' workloads and schedules are transparent and manageable.

- **User Input:** The user selects a professor's name from a dropdown or search bar.
- **Output:** The timetable for the selected professor is displayed, showing their assigned courses and corresponding time slots.
- **Error Handling:** If a professor has no assigned courses, the system provides an appropriate message to notify the user.

4.1.1.2 USER INTERFACE DESIGN

The View Page has been designed with simplicity and efficiency in mind. The interface includes the following elements:

- **Dropdown Menus:** Allow users to select batches or professors quickly.
- **Search Functionality:** Enables users to locate specific batches or professors by typing their names.
- **Timetable Display:** Organized in a tabular format for clear and concise visualization of schedules. Rows represent time slots, while columns represent days of the week.
- **Real-Time Feedback:** Any changes made to batches or professor assignments are reflected instantly in the displayed timetable.

4.1.1.3 ALGORITHM INTEGRATION

The Greedy Randomized Scheduling Algorithm powers the timetable generation in the View Page Module. When a batch or professor is selected:

- The system retrieves data regarding courses, professors, and available time slots.
- The algorithm assigns courses to slots based on availability and constraints, such as professor availability and batch-specific requirements.

- The results are displayed in real-time, ensuring the generated timetable adheres to the defined rules and avoids conflicts.

4.2 CREATE BATCH MODULE

The Create Batch Module is a cornerstone of the Automatic Timetable Generator, designed to provide a user-friendly and centralized interface for administrators to efficiently define and manage student groups. These groups, referred to as "batches," represent collections of students who share a common set of courses and follow a similar schedule. This module simplifies the batch creation process, allowing administrators to organize students systematically and link them to their respective academic curriculum. By enabling the seamless assignment of courses to batches, this feature ensures that all relevant subjects are accurately grouped, laying a strong foundation for the subsequent generation of error-free and well-structured timetables.

This module not only helps in maintaining logical organization within the system but also enhances scheduling efficiency by ensuring that every course required for a batch is appropriately allocated during timetable generation. Additionally, it reduces manual effort and the risk of errors by providing automated validation checks, such as preventing duplicate batch names or ensuring that courses are assigned to at least one batch. As a pivotal part of the system, the Create Batch Module plays a crucial role in aligning the academic structure with the scheduling algorithm, ensuring the generation of conflict-free and optimized timetables tailored to the specific needs of each batch.

4.2.1 BATCH CREATION


This feature allows administrators to create new batches with unique identifiers, ensuring clarity and preventing duplication.

- **Batch Name Input:** Administrators can input a unique name or code for each batch. This helps in easy identification during timetable generation and prevents overlap.
- **Batch Validation:** The system ensures that the entered batch name is unique, checking against existing batches to avoid duplication. If a duplicate is detected, the system prompts the user to modify the input.

4.2.2 COURSE ASSIGNMENT

Courses can be easily assigned to batches, defining the curriculum each group will follow.

- **Course Selection Interface:** Administrators can choose courses for a batch using an intuitive multi-select dropdown or checkbox-based system.
- **Real-Time Updates:** Any changes in the course list, such as new courses added or old ones deleted via the Create Course Module, are dynamically reflected here.
- **Course Details:** Each course includes relevant metadata like credit points or duration, helping users understand the workload associated with the batch.



The screenshot shows a web application interface for creating a batch. At the top, there is a navigation bar with tabs: Home, View, Create Batch, Create Course, Create Professor, and Delete. The 'Create Batch' tab is currently selected. Below the navigation bar, the main content area has a yellow background. It contains two sections. The first section is labeled 'Name' and has a text input field containing 'CSE C' and a blue 'Create' button. The second section is labeled 'Courses' and has two dropdown menus. The first dropdown menu is labeled 'Python' and the second is labeled 'Gayatri'. To the right of these dropdowns is a blue 'ADD' button.

Figure 4.2: Create Batch Page

4.3 CREATE COURSE MODULE:

The Create Course Module is a pivotal component of the Automatic Timetable Generator system, enabling administrators to define, manage, and organize courses within the academic structure. This module allows the creation of new courses and assigns essential details such as course names, credit points, duration, and course codes. The efficient management of courses is crucial for building accurate and well-structured timetables, as the course data directly impacts the scheduling process. When an administrator accesses the Create Course module, they are presented with a user-friendly form that prompts them to enter key information about the course.

The course name and code are mandatory fields, ensuring that each course is uniquely identifiable. Credit points are an essential component of course creation, as they indicate the weight or significance of the course, which can affect how much time is allotted for that course during timetable generation..

The Create Course Module also includes the ability to assign courses to specific batches. Once a course is created, administrators can select from a list of available batches and assign the course to one or more batches. This helps in organizing the courses according to the student groups, ensuring that each batch's timetable reflects the courses they need to attend. Along with batches, the module also facilitates the assignment of professors to courses. Administrators can select the appropriate professor for each course, ensuring that instructors are aligned with their respective subjects. To maintain data integrity and avoid errors, the Create Course Module incorporates validation checks. These checks are designed to ensure that no duplicate courses are entered into the system. For example, if an administrator tries to create a course with the same name or course code as an existing one, the system will prevent the creation of that course and prompt the user to modify the details. Similarly, all required fields, such as course name and credit points, must be filled before submission to ensure that no incomplete data is entered into the system.

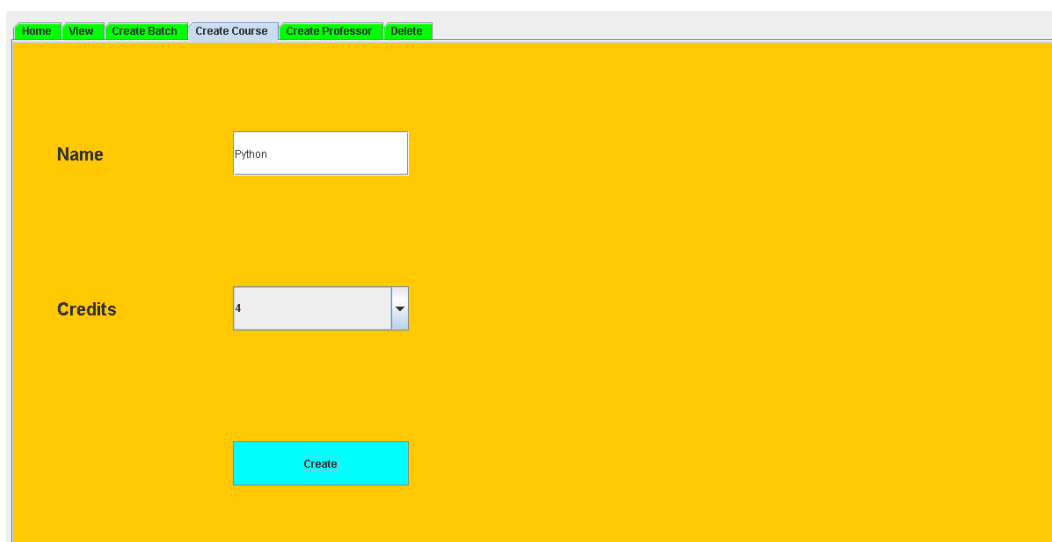
The image shows a web application interface for creating a new course. At the top, there is a navigation bar with several tabs: 'Home', 'View', 'Create Batch', 'Create Course' (which is currently selected), 'Create Professor', and 'Delete'. The main content area has a yellow background. It contains two form fields: 'Name' with a text input field containing the word 'Python', and 'Credits' with a dropdown menu currently showing the value '4'. Below these fields is a red 'Create' button.

Figure 4.3: Create Course Page

In addition to creating and managing courses, the module also allows administrators to edit or delete existing courses. If a course needs to be modified, such as changing the name, credit points, or professor assignment, the system provides an option to update the course details. Deleting outdated or unnecessary courses is equally simple, helping maintain a clean and up-to-date list of courses within the system.

Since the Automatic Timetable Generator does not rely on a database, all course data is temporarily stored within the system during the session. This data can be accessed at any time during the course creation process and is used when generating the final timetables for batches. Although this method is not persistent across system restarts, it is sufficient for the project's requirements, allowing for real-time management and organization of course information.

The Create Course Module plays an essential role in the overall timetable generation process. It ensures that the courses are correctly defined, associated with the relevant batches and professors, and validated for accuracy. The data entered through this module directly contributes to generating an accurate, conflict-free timetable. It serves as a key building block that links courses to batches and professors, providing the necessary foundation for an optimized scheduling system. By ensuring that courses are properly created and managed, this module enhances the efficiency of the timetable generation, making it easier for administrators to organize and manage academic schedules.

4.4 CREATE PROFESSOR MODULE:

The Create Professor Module in the Automatic Timetable Generator system serves as a key component for managing and organizing professor-related information within the application. This module is designed to provide the administrator with a simple and efficient interface to input, edit, and maintain the profiles of professors. The system's success heavily depends on accurate professor data, as it is essential for correctly associating professors with specific courses and ensuring that the timetable is logically structured. Through this module, the administrator can add new professors to the system, assign them to particular courses, and update or delete their records as needed.

When adding a professor, the module prompts the administrator to input relevant details such as the professor's name, email address, academic department, and any other identifying information required by the system. This ensures that each professor's profile is complete and unique. Once the professor is added, the system allows the administrator to assign specific courses to that professor. This step is crucial because the timetable needs to accurately reflect which professors will be teaching which courses. By linking the professor to their courses, the module helps avoid any scheduling conflicts and ensures that each course is adequately staffed.



Figure 4.4: Create Professor Page

Additionally, the module facilitates easy modifications to professor profiles. If there are any changes in the professor's information or if they are reassigned to different courses, the administrator can quickly update the details. This flexibility ensures that the system remains accurate and up to date, even if there are last-minute changes in staff assignments or contact details.

Furthermore, the Create Professor Module helps ensure the accuracy of the timetable. As the system progresses to generate the timetable, the professor's assignments are used to populate the schedule, ensuring that each professor is allocated the appropriate time slots for their courses. If a professor's details change or a new professor is added, the system automatically updates the timetable accordingly.

Overall, the Create Professor Module is essential for managing professor information within the timetable system. It not only supports the creation and maintenance of professor profiles but also integrates this data into the timetable generation process, ensuring that all courses are taught by the appropriate faculty members and that the generated timetable is accurate and functional. This module plays a vital role in the smooth operation of the system, making it a core feature of the Automatic Timetable Generator project.

4.4.1 ASSIGN PROFESSOR TO COURSE:

The process of assigning a professor to a course in the Automatic Timetable Generator system is a crucial step in ensuring the accurate creation of a timetable. Once a professor is added to the system, the administrator can assign specific courses to that professor, linking their profile with the courses they are responsible for teaching. This assignment is done through a simple and intuitive interface, where the administrator selects the courses from a predefined list and associates them with the professor's profile.

The assignment ensures that each course is taught by a qualified professor, and it also enables the system to properly allocate time slots and avoid any scheduling conflicts. The system requires that the professor's expertise and availability align with the courses they are assigned to. For example, if a professor specializes in a particular subject, they will be assigned to teach that course, ensuring that the content is delivered effectively.

Once the professor is linked to the course, this information is integrated into the system, allowing it to be used when generating the timetable. During the timetable generation process, the system uses this data to ensure that each course is scheduled with the correct professor, factoring in their availability and the needs of the students. By assigning professors to courses, the system can allocate specific time slots for each class, preventing double-booking or overloading any individual professor.

4.5 DELETE MODULE:

The Delete Module in the Automatic Timetable Generator system is a crucial component for ensuring that the data within the system remains current and accurate. As time progresses, certain data such as courses, professors, or batches may become obsolete or irrelevant due to various reasons—such as course cancellations, changes in staff, or the conclusion of a batch's curriculum. This module offers administrators an easy and efficient way to remove such outdated information from the system, preventing it from interfering with the timetable generation process.

Without the ability to delete unnecessary or incorrect data, the system could become cluttered with outdated entries, leading to potential conflicts or errors in the scheduling process. For example, if a course is deleted but not properly removed from the system, the timetable might still show it as an available option, causing confusion for students, professors, and administrators. Similarly, if a professor's profile is not deleted when they leave the institution, their name might still appear when trying to assign courses, leading to incorrect timetable assignments.

The screenshot shows a web interface for deleting data. At the top is a navigation bar with links: Home, View, Create Batch, Create Course, Create Professor, and Delete. The main content area has a yellow background and contains three sections, each with a dropdown menu and a red 'Delete' button:

- Delete Course**: The dropdown menu shows 'FREE HOUR'.
- Delete Professor**: The dropdown menu shows 'RAJAVARMAN'.
- Delete Batch**: The dropdown menu shows 'CSE C'.

Figure 4.5: Delete Page

The Delete Module ensures that only relevant and current information is used in the generation of timetables. It works by providing administrators with an intuitive interface where

they can select and delete courses, professors, or batches as needed. Once the deletion is confirmed, the system automatically removes all associated data, including any references to the deleted entities within the timetable scheduling, thus maintaining the integrity of the system.

In addition to preventing data clutter, the Delete Module also plays a role in safeguarding data integrity. It ensures that all necessary relationships between entities (such as courses, professors, and batches) are properly maintained when deletions are made. This prevents orphaned records or broken links within the system. For example, if a course is deleted, any professors who were assigned to that course will no longer appear in the system as associated with it, and any batch assignments for that course will also be appropriately updated or removed.

Overall, the Delete Module is essential for the smooth functioning of the Automatic Timetable Generator, ensuring that the system remains clean, accurate, and up-to-date. It minimizes the risk of errors, reduces unnecessary data clutter, and helps the system adapt to changes in staffing or course offerings, ultimately facilitating a more efficient and accurate timetable generation process.

Functionality of the Delete Module:

The Delete Module is designed with simplicity and efficiency in mind. It allows administrators to delete specific records by selecting them from a list. The key operations of the module include:

1. Delete Course
2. Delete Professor
3. Delete Batch

4.5.1 DELETE COURSE:

The Delete Course Module works by allowing the administrator to select a course from the system's list of available courses and remove it. Here's how the process typically works:

1. **Selecting a Course for Deletion:** The administrator is provided with a user-friendly interface where they can browse through the existing courses in the system. The list of

courses is usually displayed with basic information, such as course name, code, credit points, and the professor assigned to the course. The administrator selects the course that needs to be deleted.

2. **Confirmation of Deletion:** Before the course is permanently deleted, the system prompts the administrator for confirmation. This step is crucial to prevent accidental deletions, as removing a course from the system can affect various parts of the timetable, including course assignments to professors and batches. The confirmation ensures that the administrator is certain about their decision.
3. **Deleting the Course:** Once confirmed, the course is deleted from the system. This includes removing the course record from the list of available courses, and any associated information (such as professor assignments or batch schedules) is updated accordingly. If any batch or professor had the course scheduled, these records are either removed or adjusted to reflect the changes.
4. **Impact on Timetable Generation:** After the deletion of the course, the system ensures that the course no longer appears in the timetable generation process. This means that no student will be assigned to a non-existent course, and no professor will be scheduled to teach a course that has been removed from the system. Any affected timetables are automatically updated to reflect this change.
5. **Data Integrity:** The Delete Course Module works in conjunction with other modules in the system to ensure data integrity. When a course is deleted, any references to that course in other parts of the system are also removed, such as assignments to professors or inclusion in batches. This prevents orphaned data or broken links within the system, ensuring that all records are consistent.

4.5.2 DELETE PROFESSOR:

The Delete Professor Module works by providing a straightforward interface where administrators can search for and select the professor to be deleted. Here's how the process typically works:

1. **Selecting a Professor for Deletion:** The administrator is presented with a list of all professors currently registered in the system. This list may include the professor's name,

designation, courses assigned, and other relevant information. The administrator selects the professor to be deleted from this list.

2. **Confirmation of Deletion:** To avoid accidental removal of important data, the system asks for confirmation before proceeding with the deletion. The administrator must confirm that they want to delete the professor and all associated data. This step is critical, as deleting a professor can have implications on course assignments, batch schedules, and the overall timetable generation process.
3. **Deleting the Professor:** Once confirmed, the system deletes the professor's record from the system. This includes removing the professor's profile, their assigned courses, and any references to them in batches or other parts of the system. If the professor had any courses assigned to them, those courses are either reassigned to other professors or removed from the available course list, depending on the requirements.
4. **Impact on Timetable Generation:** After the deletion of the professor, the system ensures that the professor is no longer considered when generating timetables. Any scheduled classes that involved this professor are either rescheduled or removed from the timetable, ensuring that the timetable is updated and reflects only active professors.
5. **Data Integrity:** The Delete Professor Module ensures data integrity by removing all references to the deleted professor in other parts of the system. For example, if a professor is deleted, the system ensures that no references to that professor remain in batch assignments or course schedules. This prevents orphaned data and ensures that the system remains consistent and up-to-date.

4.5.3 DELETE BATCH:

The Delete Batch Module in the Automatic Timetable Generator system is an essential feature for managing and maintaining the accuracy of batch-related data within the system. This module allows administrators to remove batches that are no longer needed or relevant, ensuring that only active and up-to-date batch information is utilized in the timetable generation process. The Delete Batch Module works by enabling administrators to select and remove batches from the system. The following steps outline how this process typically occurs:

1. **Selecting a Batch for Deletion:** The administrator can access a list of all existing batches in the system, which includes details such as the batch name, associated courses, professors, and student groups. The administrator selects the batch they wish to delete from this list.
2. **Confirmation of Deletion:** To prevent accidental deletion of important data, the system requires confirmation before proceeding with the deletion. The administrator must explicitly confirm that they want to delete the selected batch. This step ensures that the deletion is intentional and prevents mistakes that could affect the timetable or scheduling process.
3. **Deleting the Batch:** After the administrator confirms the action, the system deletes the selected batch. This involves removing the batch from the system, including its associated course schedules, professors, and any student enrollment data tied to the batch. If the batch was part of the timetable generation process, the system ensures that the batch is no longer included in any future timetable calculations.
4. **Impact on Timetable Generation:** Once a batch is deleted, the system automatically updates the timetable to ensure that any courses or schedules associated with the deleted batch are removed. The deletion also ensures that no references to the batch remain in the system, preventing any errors or inconsistencies during timetable generation.
5. **Data Integrity:** The Delete Batch Module ensures the integrity of the system by removing all traces of the deleted batch from related modules and data. For example, if a batch is deleted, the courses and professors associated with that batch are either reassigned to other batches or removed from the system, as needed. This prevents the system from maintaining unnecessary or outdated data.

CHAPTER 5

SYSTEM SPECIFICATION

5.1 HARDWARE SPECIFICATION:

1. **Processor:** 2 GHz or higher (Intel Core i3/i5 or AMD Ryzen)
2. **RAM:** 4 GB RAM or more for smooth performance
3. **Storage:** 500 MB or more (for ease of operation and storage of project files)
4. **Operating System:** Windows 7 or higher, macOS, or Linux

5.2 SOFTWARE REQUIREMENTS:

1. **Programming Language: Java** (version 8 or higher)
2. **Development Environment:** Visual Studio Code or any IDE (Eclipse, or NetBeans)
3. **Java Libraries:**
 - **JFrame** (for GUI development)
 - **Swing** (for creating and managing the user interface components)
4. **Java Runtime Environment (JRE): JRE 8 or higher** to run Java applications

5.2.1 PROGRAMMING LANGUAGE: JAVA

Java, a widely used and versatile programming language, plays a crucial role in the development of the Automatic Timetable Generator project. The language's platform independence is one of the key reasons for its adoption in this project, as it allows the application to run seamlessly across different operating systems such as Windows, macOS, and Linux without requiring modifications. The project is built using Java SE (Standard Edition), which provides a rich set of libraries and APIs essential for developing desktop applications. In this project, Java is used for both the core logic and the user interface. The JFrame library, which is part of the Swing toolkit, is utilized to create the graphical user interface (GUI). Swing allows for

the creation of custom, visually appealing, and interactive interfaces. By using Java's event-driven programming model, the system enables intuitive user interactions, such as selecting a batch, adding courses, and generating timetables with simple mouse clicks and keyboard inputs.

Java's object-oriented programming (OOP) features, such as inheritance, polymorphism, and encapsulation, make it ideal for managing the complex relationships between various components of the timetable generator—such as courses, batches, and professors. These OOP concepts ensure that the code is modular, maintainable, and easy to extend in the future, should additional features be needed. The core logic of the project, which involves the Greedy Randomized Scheduling Algorithm, is implemented using Java. This algorithm efficiently assigns courses to time slots, while the integration of randomness ensures the flexibility and optimization of the timetable. Java's strong capabilities for handling data structures and logic make it well-suited for such algorithmic processes.

Overall, Java provides the necessary tools and flexibility for developing a robust, cross-platform application like the Automatic Timetable Generator, ensuring smooth performance, ease of use, and scalability.

5.2.2 DEVELOPMENT ENVIRONMENT: VISUAL STUDIO CODE

Visual Studio Code (VS Code) served as the primary development environment for the Automatic Timetable Generator project, playing a key role in streamlining the development process. As a lightweight, yet powerful, source-code editor, VS Code offered numerous features that significantly enhanced productivity during the project's development.

One of the main advantages of using VS Code for this project is its customizability. With a wide array of extensions and plugins available, VS Code allowed for easy integration of Java support, enabling features like syntax highlighting, auto-completion, and debugging tailored for Java development. This helped maintain clean, efficient, and error-free code, making it much easier to implement complex logic such as the Greedy Randomized Scheduling Algorithm for generating timetables.

The integrated Terminal in VS Code allowed for easy compilation and execution of Java programs directly within the editor, eliminating the need to switch between multiple tools or command-line interfaces. Additionally, the Git integration within VS Code helped manage version control and track changes to the project, ensuring smooth collaboration and control over the project's progress. By directly linking to Git repositories, VS Code simplified tasks such as commit, push, and pull, making version management easier and more efficient.

VS Code's intuitive user interface, coupled with its debugging capabilities, also proved to be an essential tool in troubleshooting issues. Features such as breakpoints, step-through debugging, and the debug console helped identify and resolve errors in real-time, ensuring that the application functioned as expected throughout its development.

Furthermore, VS Code supported smooth integration with other Java-based tools, libraries, and frameworks, which were useful in developing the core functionality of the project. This, combined with the lightweight nature of the editor, allowed for faster load times and more responsive development, leading to a quicker turnaround in implementing new features and making adjustments to the existing codebase.

Overall, VS Code was instrumental in developing the Automatic Timetable Generator, providing a flexible, efficient, and user-friendly environment that significantly improved the development process and ultimately contributed to the successful completion of the project.

5.2.3 JAVA LIBRARIES:

In the Automatic Timetable Generator project, several Java libraries have been utilized to enhance functionality and simplify the development process. These libraries are essential for building the graphical user interface (GUI), managing user interactions, and ensuring that the application is efficient, responsive, and easy to use.

1. JFrame (for GUI development),
2. Swing (for creating and managing the user interface components)

5.2.3.1 JFrame:

In the Automatic Timetable Generator project, JFrame played a crucial role as the foundation for building the graphical user interface (GUI). JFrame, part of the Swing library in Java, is used to create the main window of a desktop application. As a subclass of `java.awt.Frame`, it serves as the container for other graphical components like buttons, text fields, labels, and panels. It allows developers to create and organize the user interface components in a structured and visually appealing manner. JFrame simplifies the creation of windows in desktop applications by providing a flexible container that can hold various UI elements. This class is an essential part of creating any graphical user interface in Java, and it is especially useful in applications that require a structured window layout, such as the Automatic Timetable Generator. In this project, JFrame was utilized to create the main window that holds the interface for generating timetables, creating batches, courses, and professors, and managing data entries.

The JFrame class provides several important methods for customizing and controlling the window's behavior. Methods such as `setTitle()` are used to set the title of the window, while `setSize()` determines the dimensions of the window. Furthermore, the `setDefaultCloseOperation()` method enables developers to specify the action when the user closes the window, ensuring proper termination of the application. JFrame also works hand-in-hand with layout managers such as `BorderLayout`, `FlowLayout`, and `GridLayout`, which organize the components inside the frame in a structured and predictable manner. These layout managers help to automatically arrange components based on the specified layout, ensuring that the interface remains consistent across different screen sizes and resolutions.

By utilizing JFrame, the Automatic Timetable Generator not only benefits from a reliable and standardized window structure but also leverages its capabilities to create a dynamic, interactive, and user-friendly application. The JFrame class is integral in ensuring that the GUI is visually appealing and responsive to the user's needs, making it an essential part of the project's development.

5.2.3.2 SWING:

The Swing library in Java is a powerful tool for creating graphical user interfaces (GUIs) that are rich, interactive, and highly customizable. In the Automatic Timetable Generator project, Swing was utilized extensively to manage and render the user interface components. Swing, a part of Java's Java Foundation Classes (JFC), provides a wide range of components like buttons, labels, text fields, tables, and combo boxes that are essential for building dynamic and user-friendly interfaces.

One of the primary advantages of Swing is its platform independence, which allows it to create consistent and functional user interfaces that work seamlessly across all operating systems without modification. This was particularly beneficial in the Automatic Timetable Generator, as the application needed to be flexible enough to run on different systems like Windows, macOS, and Linux. By using Swing, the project ensured that the GUI would look and function the same across all platforms, providing a uniform user experience regardless of the environment.

Swing components are designed to be highly customizable. Each component can be customized in terms of appearance, behavior, and interactivity. For example, the project uses JButton, JTextField, and JComboBox to enable users to interact with the application by selecting batches, professors, and courses, as well as entering relevant data. Swing's Look-and-Feel feature allows developers to apply different visual themes to the user interface, making it possible to design an application with a modern and visually appealing look. By grouping related components within JPanel containers, the project was able to keep the interface organized and easy to navigate. JPanel acts as a container for placing other components like buttons, labels, and text fields, ensuring that these elements are displayed in a logical and structured manner within the JFrame. This feature is especially useful when creating complex interfaces, such as the one in the Automatic Timetable Generator, where different sections (e.g., creating batches, courses, professors) need to be logically separated for ease of use.

One of the most important features of Swing is its event-driven model. In Swing, user interactions such as mouse clicks, keyboard presses, and menu selections trigger events that the

application can respond to. For instance, when the user selects a batch or professor from a drop-down menu or clicks on a button, an event is generated, and the application processes the event accordingly. This event handling mechanism allows the Automatic Timetable Generator to react to user inputs dynamically, providing real-time feedback and making the application interactive. Swing's event-handling model ensures that users can create timetables, add courses, and manage professors with ease. By using Swing, the project ensures that the application is not only functional but also provides an interactive environment for users to input, manage, and view data effectively. Swing's versatility in designing complex user interfaces makes it an essential part of the development process, ensuring that the Automatic Timetable Generator delivers a smooth and user-friendly experience for both administrators and users alike.

5.2.4 JAVA RUNTIME ENVIRONMENT (JRE):

The Java Runtime Environment (JRE) plays a critical role in the functioning of the Automatic Timetable Generator project. The JRE is a software package that provides the necessary libraries, components, and tools to run Java applications. It includes the Java Virtual Machine (JVM), core libraries, and supporting files required for executing Java programs. In this project, the JRE is essential for executing the Java code that forms the backend of the timetable generation system, ensuring the correct functioning of the application on the user's machine.

When the Automatic Timetable Generator application is launched, the JRE is responsible for interpreting the compiled Java code and running it in the form of bytecode. The JRE facilitates cross-platform compatibility, meaning the application can run on any operating system (Windows, macOS, Linux) that has the appropriate version of the JRE installed. This eliminates the need to worry about specific hardware or operating system requirements, making the project accessible to a wider range of users.

CHAPTER 6

METHODOLOGY

6.1 ALGORITHM METHODOLOGY

The Greedy Randomized Scheduling Algorithm is at the core of this project, enabling automated timetable generation with a focus on efficiency and adaptability. This algorithm combines a greedy approach, which prioritizes immediate optimization, with randomized logic to introduce variability. The primary goal is to allocate time slots to courses in a way that maximizes resource utilization while avoiding repetitive or overlapping schedules.

The algorithm operates in a series of steps:

1. **Input Gathering:** It begins by collecting data about available time slots, courses, professors, and batch requirements. These inputs are essential for defining constraints and possibilities within the timetable.
2. **Greedy Allocation:** In each iteration, the algorithm selects a course that needs scheduling and attempts to assign it to the earliest available and valid time slot based on the professor's and batch's availability. This ensures priority is given to slots that fulfill the most constraints.
3. **Randomization:** To prevent deterministic patterns and ensure fairness, a layer of randomness is applied. For instance, if multiple time slots are equally valid for a course, one is chosen at random, introducing diversity in the generated timetables.
4. **Conflict Resolution:** The algorithm checks for scheduling conflicts, such as double-booking professors or overlapping courses for the same batch. If conflicts arise, adjustments are made dynamically to resolve them.

By integrating greedy logic with randomization, this methodology creates balanced timetables that are both efficient and equitable. The adaptability of this approach allows the system to handle diverse scenarios, such as accommodating unexpected schedule changes or new course additions.

6.1.1 ALGORITHM STEPS:

1. Initialize Data Structures:

Create a list of all available time slots for each day of the week. Store the list of courses, professors, and batches. Create an empty timetable structure to store the final schedule.

2. Input Constraints:

Gather user input for the total number of courses, professor availability, and batch preferences. Define constraints, such as: A professor cannot teach two classes at the same time. A batch cannot attend more than one course in a given time slot. Certain courses may require specific time slots.

3. Sort Courses by Priority:

Sort the courses based on their credit points or the number of hours required per week. High-priority courses are scheduled earlier to ensure their allocation in the timetable.

4. Greedy Allocation:

For each course in the sorted list: Check the availability of the assigned professor. Check the availability of the batch. Find the earliest available time slot that satisfies all constraints. Assign the course to the selected time slot.

5. Introduce Randomization:

If multiple time slots are valid for a course, randomly select one to introduce diversity and prevent predictable patterns. This randomness ensures fairness and avoids unbalanced schedules.

6. Conflict Detection and Resolution:

After assigning each course, check for conflicts: If a professor is double-booked, reassign the conflicting course. If a batch is double-booked, adjust the schedule by swapping time slots. Repeat the allocation process for unresolved conflicts.

7. Repeat Until Completion:

Continue assigning courses until all have been scheduled or no valid time slots remain.

8. Output Timetable:

Once all courses are scheduled, generate the timetable for each batch and professor. Display the results in the View Page Module for user review.

6.2 USER INTERACTION MECHANISM

User interaction is a cornerstone of the project, ensuring that the system is intuitive and accessible to administrators with minimal training. Built using Java Swing and JFrame, the interface provides a visually appealing and functional environment for performing essential tasks like batch creation, course assignment, and timetable generation. Each module within the system is designed with user-centric principles:

- **Create Batch Module:** Administrators are guided through the process of defining batches, assigning courses, and organizing students into logical groups. Real-time validation ensures that data input errors are minimized, enhancing accuracy.
- **Create Course Module:** Users can add new courses, specify their credit points, and integrate them into the system with ease. The interface highlights any inconsistencies, such as duplicate course names, enabling immediate corrections.

- **Create Professor Module:** Professors are added to the system, with options to assign them to specific courses. Dropdown menus and auto-suggestions streamline the process of data entry.
- **View Page Module:** This module enables users to generate and view timetables for both batches and individual professors. Interactive filters and selection options make it easy to customize the display, providing an efficient way to analyze and share schedules.

The consistent design and intuitive layout across all modules reduce the learning curve and empower users to navigate the system confidently.

6.3 SYSTEM ARCHITECTURE AND INTEGRATION:

The system architecture is modular and robust, designed to ensure seamless interaction between different components. Unlike systems that rely on complex databases, this project uses Java's powerful data structures to store and manage information effectively. Arrays, lists, and hash maps are employed to organize data related to courses, professors, and batches, ensuring quick access and manipulation. The modular design ensures that each function, such as batch creation or timetable viewing, operates independently while sharing data with other modules through well-defined interfaces. This approach facilitates scalability, allowing for the addition of new features without disrupting existing functionalities. For example, when generating a timetable:

1. **Data Retrieval:** Information about professors, courses, and batches is fetched from the respective modules.
2. **Logic Processing:** The Greedy Randomized Scheduling Algorithm processes the data to allocate time slots.
3. **Output Delivery:** The timetable is displayed dynamically in the **View Page Module**, ensuring clarity and ease of understanding.

The architecture's modularity also aids in debugging and maintenance, as individual components can be updated or replaced without affecting the entire system.

6.4 DEVELOPMENT ENVIRONMENT AND TOOLS

The development of the Automatic Timetable Generator was carried out in Visual Studio Code (VS Code), a modern and versatile integrated development environment (IDE). The choice of VS Code as the development environment significantly influenced the project's success by providing a feature-rich yet lightweight platform for Java application development.

Key benefits of VS Code in this project include:

1. **Ease of Java Integration:** With plugins like the Java Extension Pack, VS Code offers comprehensive support for writing, debugging, and running Java applications. The IDE provided seamless access to libraries like Swing and JFrame, which were integral to the project's GUI development.
2. **Code Management:** Features like syntax highlighting, code completion, and error detection enhanced productivity by reducing coding errors and speeding up development.
3. **Version Control:** Built-in Git support allowed the team to track changes, manage versions, and collaborate efficiently.
4. **Customization:** VS Code's extensive plugin ecosystem enabled the integration of tools specific to project needs, such as code formatters and linters, ensuring consistent coding practices.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The Automatic Timetable Generator signifies a significant leap forward in the automation of scheduling tasks for educational institutions, addressing the complexities and inefficiencies inherent in traditional manual methods. This innovative project leverages the Greedy Randomized Scheduling Algorithm, which combines the precision of a systematic approach with the adaptability of randomization to produce optimal timetables. The algorithm ensures fairness in resource allocation, effectively managing constraints such as available professors, course timings, and classroom capacities. As a result, the system minimizes conflicts and maximizes the efficient use of institutional resources. Developed entirely in Java, the project utilizes JFrame and Swing to create a visually appealing and intuitive graphical user interface (GUI). These Java libraries not only provide a robust foundation for the application but also enhance user experience by enabling seamless navigation and interactive functionality. The system's modular architecture divides its functionalities into distinct yet interconnected components, such as creating and managing batches, assigning professors to courses, and handling deletions. This modularity simplifies the overall operation, making it highly user-friendly and easily maintainable.

By automating the scheduling process, the system significantly reduces administrative workload, freeing up valuable time for academic staff to focus on more strategic tasks. Additionally, the elimination of human error in timetable creation improves the overall efficiency and reliability of the schedules, ensuring that classes run smoothly and without unnecessary conflicts. The project also fosters transparency, as users can view and verify the generated timetables with ease. In essence, the Automatic Timetable Generator is not merely a technological achievement but a practical tool that holds the potential to transform how educational institutions manage their scheduling needs. It exemplifies how thoughtful application of technology can simplify complex processes, improve productivity, and contribute to the overall efficiency of academic management.

7.2 FUTURE ENHANCEMENT

The Automatic Timetable Generator has laid a strong foundation for efficient scheduling, with immense potential for growth. Integrating a database system would enable persistent storage for batches, courses, and professors, enhancing practicality. Adding export options (PDF, Excel, Word) would simplify distribution and printing. A user access control system could improve security, offering role-based permissions for administrators, professors, and students. Custom constraints for specific time slots or requirements would add flexibility, while a mobile application could provide on-the-go accessibility.

APPENDIX – 1

SOURCE CODE

Main.java

```
import java.awt.*;

import java.util.*;

import java.awt.event.*;

import javax.swing.*;

import java.awt.image.*;

import java.io.*;

import javax.imageio.ImageIO;


class Main {

    public static void main(String[] args) throws IOException {

        Map<String, Batch> B = new HashMap<String, Batch>();

        Map<String, Course> CO = new HashMap<String, Course>();

        Map<String, Professor> PR = new HashMap<String, Professor>();

        JFrame f = new JFrame();

        JFrame s = new JFrame();
```

```

JPanel home = new JPanel();

JLabel l1 = new JLabel("Welcome to Timetable Generator");

l1.setBounds(250, 50, 850, 200);

l1.setFont(new Font("TimesRoman", Font.BOLD, 35));

BufferedImage image = ImageIO.read(new File("timetable.jpeg"));

JLabel label = new JLabel(new ImageIcon(image));

label.setSize(600, 500);

label.setBounds(300, 20, 500, 600);

home.setLayout(null);

home.setBackground(Color.orange);

home.add(label);

home.add(l1);

JPanel view = new JPanel();

String[] Timet = { "Select", };

String[] profs = { "Select" };

String[] credits = { "1", "2", "3", "4", "5", "6" };

String[] course = { "Select" };

JComboBox cr = new JComboBox(credits);

JComboBox cb = new JComboBox(Timet);

```

```

JComboBox pf = new JComboBox(profs);

JComboBox co = new JComboBox(course);

JTabbedPane t = new JTabbedPane();

JLabel vl = new JLabel("Timetable");

vl.setBounds(530, 5, 250, 70);

vl.setFont(new Font("TimesRoman", Font.BOLD, 20));

final String timet[][] = { { "Monday", "", "", "", "", "", "", "", "" },

    { "Tuesday", "", "", "", "", "", "", "", "" },

    { "Wednesday", "", "", "", "", "", "", "", "" },

    { "Thursday", "", "", "", "", "", "", "", "" },

    { "Friday", "", "", "", "", "", "", "", "" },

    { "Saturday", "", "", "", "", "", "", "", "" } };

String column[] = { "", "1", "2", "3", "4", "5", "6", "7", "8" };

JTable jt = new JTable(timet, column);

jt.setSize(1000, 100);

JScrollPane sp = new JScrollPane(jt);

sp.setSize(1000, 150);

sp.setBounds(100, 150, 1000, 125);

cb.setBounds(475, 70, 100, 20);

```

```

pf.setBounds(600, 70, 100, 20);

cb.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        for (int i = 0; i < 6; i++) {

            for (int j = 0; j < 8; j++) {

                timet[i][j + 1] = B.get(cb.getItemAt(cb.getSelectedIndex()).toString()).a[i][j];

            }

        }

        view.revalidate();

        view.repaint();

    }

});

view.setLayout(null);

view.setBackground(Color.orange);

view.add(vl); view.add(cb); view.add(pf);

jt.setVisible(true);

sp.setVisible(true);

view.add(sp);

JPanel delete = new JPanel();

```

```
JLabel d1 = new JLabel("Delete Course");

d1.setBounds(50, 50, 500, 50);

d1.setFont(new Font("TimesRoman", Font.BOLD, 30));

JLabel d2 = new JLabel("Course");

d2.setFont(new Font("TimesRoman", Font.BOLD, 20));

d2.setBounds(50, 125, 200, 50);

JComboBox d3 = new JComboBox(course);

d3.setBounds(250, 125, 200, 50);

JButton d4 = new JButton("Delete");

d4.setBounds(550, 125, 200, 50);

d4.setBackground(Color.cyan);

JLabel d5 = new JLabel("Delete Professor");

d5.setFont(new Font("TimesRoman", Font.BOLD, 30));

d5.setBounds(50, 200, 500, 50);

JLabel d6 = new JLabel("Professor");

d6.setFont(new Font("TimesRoman", Font.BOLD, 20));

d6.setBounds(50, 275, 200, 50);

JComboBox d7 = new JComboBox(profs);

d7.setBounds(250, 275, 200, 50);
```

```
JButton d8 = new JButton("Delete");

d8.setBounds(550, 275, 200, 50);

d8.setBackground(Color.cyan);

JLabel d9 = new JLabel("Delete Batch");

d9.setFont(new Font("TimesRoman", Font.BOLD, 30));

d9.setBounds(50, 350, 500, 50);

JLabel d10 = new JLabel("Batch");

d10.setFont(new Font("TimesRoman", Font.BOLD, 20));

d10.setBounds(50, 425, 200, 50);

JComboBox d11 = new JComboBox(profs);

d11.setBounds(250, 425, 200, 50);

JButton d12 = new JButton("Delete");

d12.setBounds(550, 425, 200, 50);

d12.setBackground(Color.cyan);

delete.setBackground(Color.orange);

delete.setLayout(null);

delete.add(d1); delete.add(d2);

delete.add(d3); delete.add(d4); delete.add(d5); delete.add(d6); delete.add(d7);

delete.add(d8); delete.add(d9); delete.add(d10); delete.add(d11); delete.add(d12);
```

```
JPanel createBatch = new JPanel();

JLabel cb1 = new JLabel("Name");

cb1.setFont(new Font("TimesRoman", Font.BOLD, 20));

cb1.setBounds(50, 100, 200, 50);

JTextField cb2 = new JTextField();

cb2.setBounds(250, 100, 200, 50);

JLabel cb3 = new JLabel("Courses");

cb3.setFont(new Font("TimesRoman", Font.BOLD, 20));

cb3.setBounds(50, 275, 200, 50);

JComboBox cb4 = new JComboBox(course);

cb4.setBounds(250, 275, 200, 50);

JComboBox cb7 = new JComboBox(profs);

cb7.setBounds(550, 275, 200, 50);

JButton cb5 = new JButton("ADD");

cb5.setBounds(850, 275, 200, 50);

cb5.setBackground(Color.cyan);

JButton cb6 = new JButton("Create");

cb6.setBounds(550, 100, 200, 50);

cb6.setBackground(Color.cyan);
```



```

cb4.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        cb7.removeAllItems();

        cb7.addItem("Select");

        for (int i = 0; i <
CO.get(cb4.getItemAt(cb4.getSelectedIndex()).toString()).getProfsSize(); i++) {

cb7.addItem(CO.get(cb4.getItemAt(cb4.getSelectedIndex()).toString()).getProfsItem(i));

        }

    });

createBatch.setLayout(null);

createBatch.setBackground(Color.orange);

createBatch.add(cb1); createBatch.add(cb2); createBatch.add(cb3);

createBatch.add(cb4); createBatch.add(cb5); createBatch.add(cb6); createBatch.add(cb7);

JPanel createProfessor = new JPanel();

JLabel cp1 = new JLabel("Name");

cp1.setFont(new Font("TimesRoman", Font.BOLD, 20));

cp1.setBounds(50, 100, 200, 50);

JTextField cp2 = new JTextField();

```

```

cp2.setBounds(250, 100, 200, 50);

JLabel cp3 = new JLabel("Course");

cp3.setFont(new Font("TimesRoman", Font.BOLD, 20));

cp3.setBounds(50, 275, 200, 50);

JComboBox cp4 = new JComboBox(course);

cp4.setBounds(250, 275, 200, 50);

JButton cp5 = new JButton("Add");

cp5.setBackground(Color.cyan);

cp5.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        if (cp4.getSelectedIndex() != 0) {

            CO.get(cp4.getItemAt(cp4.getSelectedIndex()).toString()).addProfs(cp2.getText());

            PR.get(cp2.getText()).addCourses(cp4.getItemAt(cp4.getSelectedIndex()).toString());

            JOptionPane.showMessageDialog(s, "The course " +
cp4.getItemAt(cp4.getSelectedIndex()).toString() + " has been assigned to " + cp2.getText());

        } else {

            JOptionPane.showMessageDialog(s, "Select a valid course!");

        }

    }

});

cp5.setBounds(550, 275, 200, 50);

```

```
JButton cp6 = new JButton("Create");

cp6.setBounds(550, 100, 200, 50);

cp6.setBackground(Color.cyan);

createProfessor.setLayout(null);

createProfessor.setBackground(Color.orange);

createProfessor.add(cp1); createProfessor.add(cp2); createProfessor.add(cp3);

createProfessor.add(cp4); createProfessor.add(cp5); createProfessor.add(cp6);

JPanel createCourse = new JPanel();

JLabel cc1 = new JLabel("Name");

cc1.setFont(new Font("TimesRoman", Font.BOLD, 20));

cc1.setBounds(50, 100, 200, 50);

JTextField cc2 = new JTextField();

cc2.setBounds(250, 100, 200, 50);

JLabel cc3 = new JLabel("Credits");

cc3.setFont(new Font("TimesRoman", Font.BOLD, 20));

cc3.setBounds(50, 275, 200, 50);

JComboBox cc4 = new JComboBox(credits);

cc4.setBounds(250, 275, 200, 50);

JButton cc5 = new JButton("Create");
```

```

cc5.setBounds(250, 450, 200, 50);

cc5.setBackground(Color.cyan);

cc5.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        d3.addItem(cc2.getText());

        cb4.addItem(cc2.getText());

        cp4.addItem(cc2.getText());

        CO.put(cc2.getText(), new Course());

        CO.get(cc2.getText()).setName(cc2.getText());
CO.get(cc2.getText()).setCredits(Integer.parseInt(cc4.getItemAt(cc4.getSelectedIndex()).toString()));

        JOptionPane.showMessageDialog(s, "A new course has been created" + "\n" +
"Course: " + cc2.getText() + "\n" + "Credits: " +
Integer.parseInt(cc4.getItemAt(cc4.getSelectedIndex()).toString()));

    }

});

createCourse.setLayout(null);

createCourse.setBackground(Color.orange);

createCourse.add(cc1); createCourse.add(cc2); createCourse.add(cc3);

createCourse.add(cc4); createCourse.add(cc5);

```

```
t.add("Home", home);

t.add("View", view);

t.add("Create Batch", createBatch);

t.add("Create Course", createCourse);

t.add("Create Professor", createProfessor);

t.add("Delete", delete);

t.setBackground(Color.green);

f.add(t);

f.setSize(1500, 800);

t.setBounds(50, 50, 1200, 600);

f.setLayout(null);

f.setVisible(true);

} }
```

APPENDIX - 2

SCREENSHOT'S

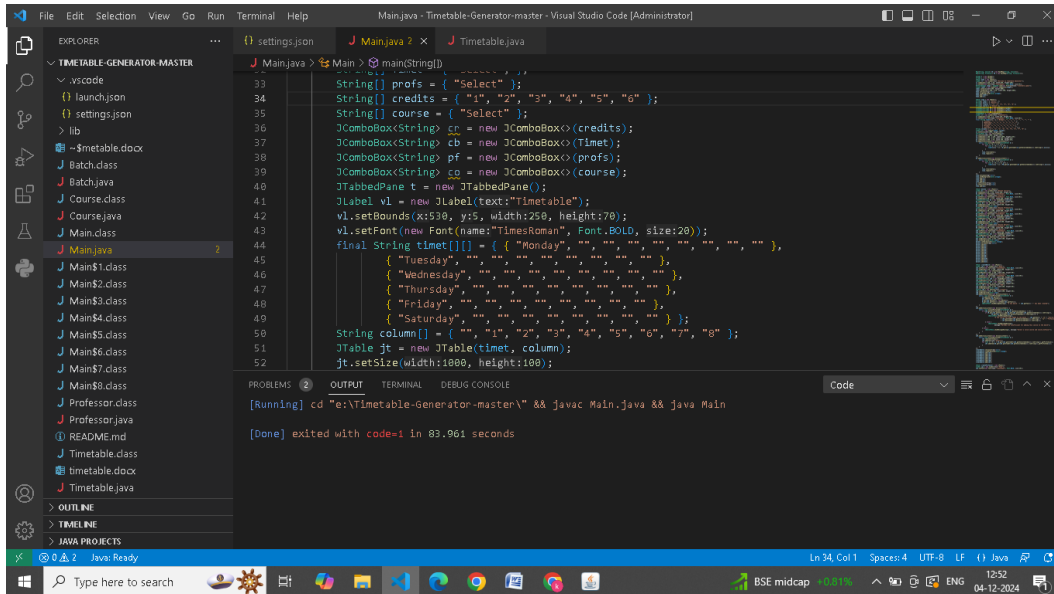


Figure 2.1: Execution of code

	1	2	3	4	5	6	7	8
Monday	TOC(RAJAVARMA)	MCAD(SOWMIYA)		AI(SARAVANAN)	LIB(SOWMIYA)		PE(KARTHIKA)	
Tuesday	PE(KARTHIKA)	MCAD(SOWMIYA)	DM(MARIA)			TOC(RAJAVARMA)	DM(MARIA)	TOC(RAJAVARMA)
Wednesday		PE(KARTHIKA)	DM(MARIA)	AI(SARAVANAN)	PE(KARTHIKA)			OOAD(VIGNESH)
Thursday		FREE HOUR(-)		DM(MARIA)	DM(MARIA)			OOAD(VIGNESH)
Friday	OOAD(VIGNESH)		MCAD(SOWMIYA)					MCAD(SOWMIYA)
Saturday	DM(MARIA)		TOC(RAJAVARMA)	AI(SARAVANAN)	OOAD(VIGNESH)	TOC(RAJAVARMA)		

Figure 2.2: Generated Timetable

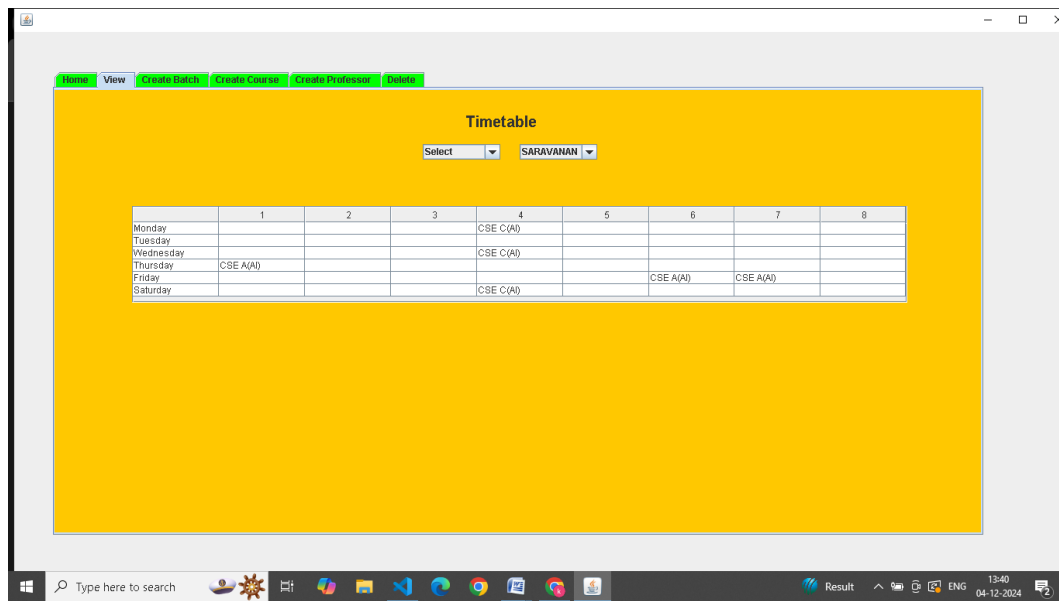


Figure 2.3: Generated Timetable for Professors

REFERENCES

1. B. M. Leung, K. T. Ng, and T. K. Dey, "An algorithmic approach to automated class scheduling in educational institutions," *Journal of Educational Technology Systems*, vol. 49, no. 3, pp. 413–430, Jul. 2020.
2. A. S. Gendreau, F. H. Lheureux, and M. C. Rousseau, "Greedy randomized adaptive search procedures for the university course timetabling problem," *Computers & Operations Research*, vol. 31, no. 9, pp. 1577–1591, 2004.
3. P. E. Sanderson, "A survey of the computational complexity of scheduling algorithms for educational timetables," *European Journal of Operational Research*, vol. 102, no. 2, pp. 236–246, 1997.
4. L. J. K. Goh, W. J. L. Poon, and R. Y. Chan, "Optimizing scheduling algorithms for educational institutions: A comprehensive approach," *Journal of Scheduling*, vol. 22, no. 5, pp. 553–570, Nov. 2019.
5. M. O'Neill and C. C. de Carvalho, "Heuristic methods for automatic timetable generation using greedy algorithms," *International Journal of Computer Science & Information Technology*, vol. 16, no. 4, pp. 68–75, 2017.
6. A. P. Singh, "An overview of timetable scheduling algorithms in higher educational institutions," in *Proceedings of the 5th International Conference on Information and Knowledge Management*, 2019, pp. 107–116.
7. R. S. Blodget, C. L. A. Mann, and T. J. Brown, "Java programming for educational tools and simulations: An application-based approach," *IEEE Transactions on Education*, vol. 56, no. 3, pp. 324–330, 2013.
8. L. Z. Sharma, M. V. Yadav, and H. G. Krishnan, "Swing-based Java GUI design and development for interactive software applications," *Journal of Computer Science and Technology*, vol. 21, no. 2, pp. 182–189, Mar. 2020.
9. G. S. Webb, "Java-based educational timetabling systems: Development and implementation," *International Journal of Software Engineering & Applications*, vol. 5, no. 7, pp. 23–35, 2014.