CHAPTER 1

INTRODUCTION

1.1 Introduction to the Project

A timetable is a key academic component in any educational institution. Coordinators must consider various factors during scheduling, which becomes particularly stressful in colleges due to multiple departments, diverse courses, and faculty members with varying designations. They must carefully allocate courses, classrooms, and faculty based on course requirements, and patiently construct an efficient timetable to ensure an organized schedule and smooth functioning of the institution. Despite the increasing automation of institutional tasks in colleges, lecture timetables are still often prepared manually due to the inherent complexities of the process.

Timetable scheduling problem is inherently complex due to the numerous interrelated factors and constraints, including teaching plans, courses, faculty and teaching classes, classrooms, and time slots. In universities, the complexity increases significantly during semester scheduling, as it must accommodate multiple student batches, various student groups, elective courses, and common mandatory courses taken by all the students. It is essential to ensure that there are no clashes among student, faculty members during the scheduling process.

Timetable scheduling is widely recognized as a combinatorial optimization problem and a constraint satisfaction problem where the goal is to find a solution that satisfies a predefined set of constraints. These constraints are generally categorized as hard and soft constraints. Hard constraints are mandatory and must be satisfied for a timetable to be feasible, such as preventing classes from clashing in the same room or a teacher having more than one class simultaneously. Soft constraints, on the other hand, represent preferences or desired conditions that ideally should be met but are not strictly required for feasibility. Meanwhile violating soft constraints is acceptable, but an optimal timetable minimizes such violations.

This project aims to address the difficulties of generating timetables by providing an automated scheduling system. Automating the process helps save time, avoid the complexity of manual management, and reduce documentation work. The goal is to generate timetables that are more accurate, precise, and free of human errors. An automated system ensures up-to-date and accurate information.

To achieve automatic timetable generation, the project proposes the use of algorithms such as genetic, heuristic, and resource scheduling. Genetic algorithms are frequently applied to timetabling problems, which are known to be NP-hard optimization problems, because Genetic algorithm is effective for finding optimal or near-optimal feasible solutions among a complex set of variables and constraints. They are based on natural selection and evolution principles and are known for their robustness in solving complex combinatorial problems. Heuristic approaches are also commonly used in timetabling, either independently or as components within algorithms like GA, often focusing on scheduling the most constrained elements first. Resource scheduling is another algorithmic approach listed for addressing these problems. These algorithms incorporate various strategies aimed at improving the efficiency, scalability and reliability of the timetable generation process.

The system will generate the timetables based on various inputs, including the number of course and faculty, faculty workload, semester details, and course priorities. Additional necessary inputs include faculty details, course details (including name and code), workload based on faculty designation, faculty and course allotment based on time slots, and details of theory and lab courses handled by each faculty. Classroom including availability and capacity, are also crucial inputs. By relying on these inputs and utilizing optimization algorithms, the system will generate possible timetables for the working days.

In conclusion, the proposed solution aims to streamline the scheduling process, enhance accuracy, and offer a more efficient, scalable, and reliable approach to timetable generation, ultimately contributing to smoother academic operations and a better teaching-learning experience.

1.2 Introduction to Technology Used

The Web-Based Automatic Timetable Scheduler integrates a set of modern technologies that together enable efficient, accurate, and scalable timetable generation. The core development is carried out in Python, chosen for its simplicity, flexibility, and the wide availability of libraries for optimization, database interaction, and system integration.

The user interface is built using Streamlit, a Python-based framework that allows the rapid creation of interactive web applications without requiring front-end technologies like HTML, CSS, or JavaScript. Through Streamlit, users can easily login, manage academic data, configure faculty preferences, and view generated timetables in a visually clear and user-friendly format. For data management, the system uses a PostgreSQL relational

database, ensuring persistent storage of users, departments, faculty, courses, and preferences. The database layer, provides structured functions for CRUD operations such as adding faculty, updating courses, or retrieving stored preferences. This design ensures secure and reliable handling of institutional data while keeping the system lightweight and portable. At the heart of the scheduler lies the Genetic Algorithm. The Genetic Algorithm is responsible for optimizing timetables by simulating natural selection through operations like selection, crossover, and mutation.

The overall system, integrates the Streamlit UI, PostgreSQL database, and Genetic Algorithm logic into a cohesive workflow. This modular architecture improves maintainability, readability, and scalability of the project. By leveraging these technologies together, the Web-Based Automatic Timetable Scheduler provides a practical and automated solution that reduces manual effort, minimizes human error, and ensures optimized utilization of academic resources.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Timetable scheduling is an NP-hard problem, meaning it involves a high level of computational complexity, and finding an exact solution is often infeasible within a reasonable time for large instances. Therefore, the goal is typically to find an optimal or near-optimal solution within a complex set of variables and constraints. As a result, extensive research has been conducted on applying optimization techniques to address this challenge.

This literature review explores various existing approaches to automated timetable generation, with a particular focus on Genetic Algorithms. It has gained popularity due to their effectiveness in solving NP-hard optimization problems like timetabling. They operate on the principles of natural selection and evolution, using mechanisms such as selection, crossover, and mutation to iteratively improve solutions. Their ability to efficiently explore large and complex search spaces makes them well-suited for generating feasible and optimized timetables under multiple constraints.

2.2 Literature Survey

Han et al. [1] proposed a hybrid approach called POGA-DP, combining Genetic Algorithms with Dynamic Programming (DP) to solve the University Course Scheduling Problem (UCSP), particularly for complex joint-course timetables. The GA component optimizes time slot assignments using a swap-based mutation with a repair mechanism, while DP allocates classrooms to minimize seat wastage and improve utilization. The method achieved significant improvements in scheduling quality (up to 46.99%) and reduced classroom usage by 29.27% compared to standard GA. Tested on data from Beijing Forestry University, the approach outperformed GA, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Producer–Scrounger Method (PSM) across multiple metrics, including fitness and occupancy rate. However, the model lacks real-time adaptability, parameter tuning, cross-institutional testing, and practical integration into university systems, limiting its deployment readiness.

Paramatmuni et al. [2] proposed a web-based timetable generation system using Genetic Algorithms, optimized for scalability and real-time adaptability. The system employs

roulette wheel and tournament selection, single-point crossover, and mutation, with a weighted fitness function evaluating hard and soft constraint violations. It achieved 95% conflict resolution and 90% resource utilization on real-world university data. The system is implemented using Django for the backend and HTML/CSS for the frontend. Despite its efficiency, the model requires manual weight tuning, offers limited support for collaborative user input, and shows reduced performance under extreme resource constraints.

Mahlous et al. [3] introduced a Genetic Algorithm for student timetabling that prioritizes individual preferences such as free days, preferred class times, and grouping with friends. The algorithm uses a binary chromosome representation to encode student-class assignments and applies customized multi-point crossover, mutation, and repair operators. A simulated annealing-based function enhances soft constraint satisfaction while preserving feasibility. The model also integrates parallel processing, fitness caching, and adaptive mutation rates to improve execution time and solution quality. However, it is limited to optimizing student allocations for fixed schedules, lacks a user-facing interface, and has not been deployed in real-time institutional settings.

Alnowaini et al. [4] propose a genetic algorithm-based approach to solve the university course timetabling problem using dynamic chromosomes, which adapt to varying course loads across departments and semesters. The system encodes scheduling variables as tuples of course, time, and room, and uses roulette wheel selection, two-point crossover, and optimized mutation to generate near-optimal schedules, achieving up to 93% conflict-free timetables in under five minutes. While the algorithm effectively satisfies both hard and soft constraints, it suffers from performance issues when scaling beyond 82 courses. However, the need for greater scalability, full automation, and dynamic real-time scheduling in future work.

Gore et al. [5] proposed an automated timetable scheduling system using a Genetic Algorithm to address the limitations of manual timetable generation in educational institutions. The model generates schedules by encoding each class with course, instructor, and room data, using crossover and mutation operators to evolve conflict-free solutions. A fitness function penalizes overlaps and lecture overloads, and the final timetables are exported in Excel format for user access. The system includes GUI modules for faculty and course input, and generates individualized timetables. While being effective at basic conflict resolution, the system does not account for advanced constraints such as faculty preferences, load balancing, or real-time adaptability. Additionally, it lacks scalability

testing on large datasets and offers no support for multi-objective optimization or collaborative scheduling environments.

Ghiridhar et al. [6] proposed a Genetic Algorithm-based system for generating conflict-free timetables tailored to the regulations of A P J Abdul Kalam Technological University. Their approach introduced a conflict operator within each chromosome to dynamically detect and resolve scheduling clashes during evolution. The system addressed both hard constraints, such as subject-hour limits and lab continuity, and soft constraints, like balanced faculty workloads. Mutation was performed through a conflict-aware local descent strategy, outperforming traditional random mutation methods. Implemented as a web application using React and Flask, the model successfully generated valid timetables for six real student batches. However, the system lacked real-time adaptability, multi-objective optimization, and scalability testing across larger institutional datasets, limiting its broader applicability.

Assi et al. [7] proposed a hybrid approach that integrates Genetic Algorithms with Graph Colouring techniques to address the University Timetabling Problem (UTP), modelling courses as graph nodes and conflicts as edges to avoid timeslot overlaps. Their implementation represents each timetable as an array of genes, where each gene contains a course's instructor, room, and timeslot, and uses penalty-based fitness evaluation to quantify violations of hard and soft constraints. Genetic operators, including roulette wheel selection, constraint-aware crossover, and low-probability mutation are complemented by a repair function that resolves infeasible solutions post-crossover. Experimental results demonstrate a 70% reduction in penalties over 150 generations, validating the method's effectiveness in constraint satisfaction. However, the approach is limited by a simplified mutation strategy, narrow crossover scope (only one conflicting pair per generation), and testing on synthetic datasets. Furthermore, the system lacks support for real-world institutional policies, dynamic re-scheduling, and user-defined preferences.

Febrita et al. [8] proposed a modified Genetic Algorithm enhanced with fuzzy time windows to optimize high school timetable scheduling. The approach categorizes subjects by cognitive demand and uses fuzzy logic to assign more mentally intensive subjects to earlier time slots. Chromosomes represent weekly schedules with 210 genes, and a fuzzy-guided mutation prioritizes replacing low-satisfaction time slots. A 2D crossover and a repair mechanism ensure constraint satisfaction and diversity. Despite demonstrating improved fitness scores and convergence compared to standard GA, the system lacks subject frequency control per day, real-time adaptability, multi-department scalability, and a deployable user interface, limiting its application in broader institutional settings.

Sampebatu et al. [9] present a timetable management system utilizing Genetic Algorithms to address the highly constrained, NP-hard nature of university course scheduling. Their system adopts a hybrid GA approach adapted from examination timetabling techniques at the University of Nottingham employing selection and crossover to generate near-optimal timetables based on user-defined constraints such as course units, hall capacities, and lecturer availability. Implemented in Delphi for its performance efficiency, the system includes a random population generator and a crossover module to iteratively improve timetable feasibility. While the method achieved high average fitness scores (up to 0.986), it is limited to 2-unit courses, lacks formal soft constraint optimization, and does not support dynamic rescheduling or real-world institutional benchmarking.

Abdelhalim et al. [10] introduced a Utilization-based Genetic Algorithm (UGA) to solve the university course timetabling problem with a focus on optimizing space utilization alongside satisfying scheduling constraints. Their approach employed a two-dimensional chromosome representation, mapping events to room-timeslot pairs, and generated the initial population using heuristic methods, the authors proposed a utilization-based crossover operator aimed at reallocating underutilized events and a targeted mutation strategy employing local search to optimize room occupancy. A composite fitness function weighted towards occupancy rates, frequency rates, and minimizing scheduling gaps was used to evaluate solutions. The system was tested on real data from Alexandria University also, against two benchmark datasets from the International Timetabling Competition (ITC 2007). The Results demonstrated significant improvements in space utilization, reduced scheduling hours, and better overall timetable quality compared to manual scheduling. However, the study identified gaps in dynamic real-time adaptability and slight computational overhead for medium-sized problems.

Mittal et al. [11] addressed the timetable scheduling problem by proposing the use of Genetic Algorithms to automate and optimise the scheduling process. The authors applied Genetic Algorithms where the algorithm initialises the population of guesses, then three operators are applied-selection, crossover and mutation to create an optimal timetable. The system was tested with a real data within the author’s institution. The results demonstrated significant improvements in both efficiency and accuracy compare to manual scheduling. Although the study succeeded in creating a more efficient alternative to manual scheduling, it lacked mechanisms for dynamic adaptability and deeper real-world constraint handling.

Paresh et al. [12] proposed a Genetic Algorithm-based solution to address inefficiencies in manual academic timetable generation by introducing a novel chromosome structure. Each

chromosome encoded multiple genes containing course, faculty, frequency, group, and room data, allowing a more granular and adaptable representation of timetable data. The system incorporated GA operations including one-point crossover and standard mutation, with the fitness function evaluating constraint satisfaction. The model effectively handled both hard constraints and soft constraints, thus successfully generating timetables for Information Technology and Computer Engineering departments. Despite these strengths, the system was limited in scope, lacking scalability across multiple departments or large institutions. Furthermore, faculty preferences and dynamic constraints such as elective course selection or live updates were not addressed.

Othman et al. [13] presented a Genetic Algorithm approach for university course timetabling, ensuring all hard constraints were satisfied while minimizing soft constraint violations. Each chromosome encoded subject, section, instructor, time, and room data. The method employed rank-based selection, single-point crossover, and random mutation, with exclusiveness to retain the best solutions across generations. A fitness function was designed to normalize and weigh both hard and soft constraint violations, covering instructor time preferences, room assignments, course prerequisites, and overload control. Tested on real data from the University of Jordan, the algorithm achieved zero hard constraint violations and significantly reduced instructor overload and scheduling conflicts. However, the model lacked real-time adaptability, sensitivity to constraint weights, and user interaction capabilities, limiting its flexibility and scalability in dynamic environments.

Sapru et al. [14] proposed a Genetic Algorithm based solution for university timetable scheduling, introducing a novel Guided Mutation operator that selectively accepts only beneficial mutations to improve convergence and constraint satisfaction. Chromosomes represent weekly timetables for multiple student groups, encoded in a binary structure that captures room, faculty, subject, and slot activity. A rank-based selection strategy is employed to maintain diversity and avoid premature convergence, while a single-point crossover enhances genetic variation. The guided mutation approach consistently outperformed conventional mutation, leading to faster generation of feasible timetables (fitness = 1.0) across varied test cases. However, the study focused solely on hard constraints, omitting critical soft constraints like instructor preferences or scheduling gaps. Additionally, scalability to large datasets and practical deployment aspects were not addressed.

Colorni et al. [15] proposed a Genetic Algorithm approach to solve the highly constrained school timetabling problem. Their method used a two-dimensional matrix representation of

teachers and timeslots, with a hierarchical objective function focusing first on feasibility, second on management rules, and finally on individual teacher preferences. This method uses customized genetic operators, like a row-based crossover and order-k mutation, along with a genetic repair process to correct infeasibilities. The integration of local search significantly enhanced timetable quality, achieving a substantial reduction in cost compared to manual scheduling and simulated annealing. However, the approach faced scalability challenges due to the computational complexity of the repair process and lacked mechanisms for dynamic timetable adjustments.

2.3 Summary of Literature Survey

Several studies on automated timetable generation, emphasizing methodologies such as Genetic Algorithms and hybrid approaches, are summarized in Table 2.1, which provides an integrated overview of these significant works. Each paper is outlined by its core method, main advantages, and future work. The comparison highlights innovation trends and uncovers common limitations, helping to identify gaps and guide future development in timetable scheduling systems.

Table 2.1: Observations of Literature Survey

Title	Author	Methodology	Advantages	Future Work / Limitations
Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming	Xu Han, Dian Wang	Hybrid approach which is a combination of Dynamic Programming and Genetic algorithm.	Improved quality, reduced room use and outperformed PSO, ACO, PSM.	No cross-institution testing, lacks integration into university systems
Smart Timetable Generation using Genetic Algorithm.	Sahith Siddharth Paramatmuni, Dumpala Yashwanth	Web-based GA with roulette/tournament selection, weighted fitness function;	Fast scalable and provides conflict resolution.	Manual weight tuning, limited collaborative input, reduced performance

	Reddy, Elakurthi Sai Spoorthi, Akhil Dharani, K. Venkatesh Sharma	implemented in Django; outputs Excel files.		under extreme constraints.
Student timetabling genetic algorithm accounting for student preferences.	Ahmed Redha Mahlous, Houssam Mahlous	GA with binary chromosome, simulated annealing based function and adaptive mutation.	Solves student preference satisfaction with adaptive mutation.	No user interface, not deployed in real-time, fixed schedules only.
Genetic algorithm for solving university course timetabling problem using dynamic chromosomes.	Ghazi Alnowaini, Amjad Abdullah Aljomai	Genetic Algorithm using dynamic chromosomes for course-time-room tuples, roulette wheel selection; two-point crossover, optimized mutation.	93% conflict-free, handles varying loads	Poor performance beyond 82 courses, lacks real-time and automation.
Institute Timetable Scheduler.	Bhaven Gore, Disha Shirdhankar, Giriraj Belanekar	GA encoding course, instructor, room and GUI for input, fitness function penalizes clashes/overloads, output in Excel.	GUI interface, conflict-free timetables, Excel sheet export.	No soft constraints, Lacks scalability.
Timetable Generation Using Genetic Algorithm For	Ghiridhar S, Sachin A, Edwin M.T,	Genetic Algorithm with conflict operator; conflict-aware local descent	Custom web app, solved real institution cases,	No multi-objective optimization, lacks scalability

Batches Under Apj Abdul Kalam Technological University	Unnikrishnan K.N	mutation; implemented via React + Flask web app.	effective conflict resolution.	
Genetic algorithm analysis using the graph coloring method for solving the university timetable problem.	Maram Assi, Bahia Halawi, Ramzi A Haraty	Hybrid GA with Graph Colouring and genes encode instructor, room and timeslot.	70% weight reduction and effective infeasibility repair.	Simplified mutation, narrow crossover scope, no real-world testing or user preferences
Modified genetic algorithm for high school time-table scheduling with fuzzy time window.	Ruth Ema Febrita, Wayan Firdaus Mahmudy	Modified GA with fuzzy time windows, cognitive load-based slot assignment, fuzzy-guided mutation.	Better convergence & fitness; prioritizes cognitive load.	Lacks interface and multi-dept scalability.
Timetable Management Using Genetic Algorithms.	Limbran Sampebatu, Aries Kamolan	Hybrid GA adapted from exam timetabling and random population generator	High average fitness, handles hall capacities, lecturer availability.	Supports only 2-unit courses, no soft constraint optimization;
A utilization-based genetic algorithm for solving the university	Esraa A. Abdelhalim, Ghada A. El Khayat	Utilization-based genetic algorithm: focused on optimizing space utilization alongside	Improved space utilization, reduced hours, real data is being used.	Lacks real-time adaptability.

timetabling problem (uga).		satisfying scheduling constraints.		
Automatic timetable generation using genetic algorithm.	Dipesh Mittal, Hiral Doshi, Mohammed Sunasra, Renuka Nagpure.	A Genetic Algorithm was implemented in C# using selection, crossover, and mutation to generate conflict-free timetables based on real departmental data.	Improved efficiency & accuracy vs. manual scheduling.	Lacks dynamic adaptability & deeper real-world constraint handling.
Solving time-table scheduling problem by novel chromosome representation using Genetic algorithm.	Paresh M Chauhan, Kashyap B Parmar, Mahendra B Mendapara	GA with novel chromosome (course, faculty, group, room info) one-point crossover and standard mutation, I/O is managed using XML/HTML interface.	Granular data representation, handles both hard & soft constraints.	Lacks faculty preferences, dynamic constraints, and scalability.
A novel genetic algorithm technique for solving university course timetabling problems.	Othman M. K. Alsmadi, S. Za'er, Dia I. Abu-Al-Nadi, Alia Algsoon	Genetic algorithm which involves rank based selection, with a generation limit of the fitness ≥ 0.99.	Zero hard constraint violations, reduced overload.	No real-time adaptability, lacks sensitivity to constraint weights
Time table scheduling using genetic	Vinayak Sapru, Kaushik	GA with binary chromosome, Guided Mutation,	Avoids getting stuck on bad solutions too	Ignores soft constraints, not scalable.

algorithms employing guided mutation.	Reddy, B Sivaselvan	rank-based selection and single-point crossover.	early, creates complete timetables.	
A genetic algorithm to solve the timetable problem.	Alberto Colorni, Marco Dorigo, Vittorio Maniezzo	Genetic Algorithm with customized genetic operators like row-based crossover, order-k mutation, and a filter.	Reduced time vs. manual & simulated annealing.	Scalability issues, no dynamic adjustment can be made. Lacks faculty preferences.

2.4 Comparison with Existing Systems

Automated timetable scheduling has been the subject of extensive research, with numerous approaches leveraging various optimization algorithms. Traditional manual scheduling methods are prone to inefficiencies, leading to scheduling conflicts, workload imbalances, and increased effort. To address these issues, several advanced computational techniques have been proposed, primarily focusing on Genetic Algorithms due to their ability to efficiently generate optimized timetables under complex constraints.

Existing systems employing GA-based approaches demonstrate improvements in scheduling efficiency. For instance, some models utilize utilization-based genetic algorithms, optimizing space usage while satisfying hard constraints. Others integrate hybrid methods, combining Genetic Algorithms with Dynamic Programming or Fuzzy Logic to refine results further. While these approaches successfully minimize violations of scheduling constraints, they often lack real-time adaptability and user interactivity. Many of these solutions focus on pre-semester timetable generation, requiring manual adjustments for dynamic changes.

2.5 Proposed System

The Web-Based Automatic Timetable Scheduler is designed to address the complexities and inefficiencies associated with manual scheduling in educational institutions. The system automates the process using Genetic Algorithms and heuristic optimization techniques to generate conflict-free and optimized timetables while satisfying predefined

constraints. The primary objective is to ensure an efficient, scalable, and user-friendly solution that accommodates multiple departments, diverse faculty preferences, and dynamic scheduling adjustments.

The system will leverage constraint satisfaction principles, categorizing conditions into hard constraints (mandatory requirements, such as preventing timetable clashes and ensuring faculty availability) and soft constraints (preferences like distributing lectures evenly and minimizing gaps). Unlike existing scheduling solutions, which often require manual intervention for modifications, the system incorporates real-time adaptability, allowing users to make necessary changes dynamically.

2.6 Objectives

The primary objective of the Web-Based Automatic Timetable Scheduler is to streamline the scheduling process for colleges by automating timetable generation through optimization techniques. The system aims to eliminate inefficiencies and human errors in manual scheduling while ensuring scalability and adaptability to diverse academic environments.

The key objectives of the proposed system include:

- To automate the timetable creation process, eliminating the inefficiencies and errors associated with manual scheduling.
- To apply a Genetic Algorithm that optimizes the allocation of courses, and teachers, while satisfying hard constraints and minimizing violations of soft constraints.
- To develop a scalable and efficient scheduling model capable of handling multiple batches, courses, and faculty simultaneously.
- To provide a user-friendly interface that allows users to easily input data, modify schedules, and view the generated timetable.
- To ensure reliability and accuracy in timetable generation by incorporating validation checks and conflict detection mechanisms.

CHAPTER 3

REQUIREMENT SPECIFICATION AND ANALYSIS

3.1 Introduction

An efficient automated timetable scheduling system requires well-defined specifications to ensure smooth implementation. This chapter details the functional, user interface, software and hardware requirements that govern the development of this system.

3.2 Functional Requirements

The system must fulfil several key functional requirements to support automatic timetable generation effectively:

3.2.1 Timetable Generation

The system automatically generates conflict-free schedules, considering faculty workload, and course allotment. Allow adjustments based on faculty preferences such as preferred/blocked teaching slots.

3.2.2 Constraint Handling

Hard Constraints (must be strictly satisfied):

- No clashes between classes scheduled.
- No faculty assigned to multiple classes at the same time.
- Courses assigned according to department requirements.

Soft Constraints (should be optimized but can be violated if necessary):

- Minimizing gaps in timetable.
- Distributing lectures evenly throughout the week.

3.2.3 Data Input & Management

Faculty Details: Names, designations, availability.

Course Details: Course names, codes, types (lecture/lab), credit hours.

Department and Semester Details: Name of department, active semesters.

3.3 User Interface Requirements

For ease of use, the system's interface should be:

Intuitive: Designed for quick navigation and clear data visualization.

Accessible: Mobile-friendly for easy timetable access on various devices.

Graphical: Interactive visual timetables for better readability.

Customizable: Users should be able to modify preferences dynamically.

3.4 Software Requirements

The successful development and deployment of the automatic timetable scheduler rely on a robust and well-defined software environment. These software requirements ensure that the system operates efficiently, remains secure, and supports scalability.

- **Programming Language:** The system is built using Python 3.10, a versatile and powerful programming language widely used for web development. Python's extensive library ecosystem and compatibility with Streamlit make it an ideal choice for this project.
- **Framework:** Streamlit is an open-source Python library designed to simplify the creation and sharing of custom web applications, particularly for machine learning and data science projects. It allows developers to build interactive data apps using only Python, eliminating the need for traditional web development knowledge. Streamlit is widely used for building interactive dashboards and data exploration tools, Machine learning model demonstrators and explainability tools, Data-driven web applications and reporting tools, Chatbot interfaces and NLP applications, and Scientific and engineering applications.
- **Web Browser:** The system is designed to be compatible with modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge. This ensures a consistent user experience across different platforms, including Windows, macOS, and Linux. The use of responsive web design techniques further ensures usability on mobile and tablet devices.

- **Operating System:** The system is platform-independent and can run on any OS that supports Python and Streamlit. Commonly used operating systems include:
  - Linux (Ubuntu): Suitable for development and production due to its stability and extensive community support.
  - Windows and macOS: Suitable for local development and testing.

3.5 Hardware Requirements

Processor: Intel Core i5 and above

System Type: 64-Bit Operating System

HDD: 100GB and above

RAM: 4-GB RAM and above

3.6 Non-Functional Requirements

Non-functional requirements describe the quality characteristics that determine the overall performance and user experience of the system. These requirements ensure that the timetable generation process is not only efficient but also reliable, user-friendly, and robust under varying operational conditions.

3.6.1 Performance

The system must efficiently generate complete and conflict-free timetables within a reasonable time frame, even when processing large datasets involving multiple departments, faculty members, and courses. The underlying Genetic Algorithm shall be optimized for computational efficiency to ensure scalability and responsiveness. Furthermore, the application must maintain stability and performance consistency during peak loads, ensuring that the timetable generation and modification processes remain smooth and uninterrupted.

3.6.2 User-Friendly

The user interface shall be designed to be intuitive, simple, and accessible for users with minimal technical expertise. All menus and input sections shall include clear labels and concise instructions to minimize user errors. The system shall also provide real-time feedback through success messages, and error notifications, guiding users effectively

through each operation. Ease of navigation and a visually clear layout are essential to promote seamless user interaction and improve the overall user experience.

3.6.3 Reliability

Reliability is critical to ensure that the generated timetables are accurate, consistent, and free from conflicts under all operational conditions. The system shall incorporate robust data validation mechanisms to prevent invalid or incomplete entries during data input. Comprehensive error-handling procedures will ensure that scheduling errors are identified and resolved efficiently.

CHAPTER 4

SYSTEM DESIGN

4.1 Introduction

System Design describes the overall structure of the Web-Based Automatic Timetable Scheduler for Colleges. It translates requirements into a modular architecture, data models, user interfaces and processing workflow so that the system implemented is robust, maintainable, and extensible. Design decisions emphasize simplicity for users, correctness of generated timetables enforcing hard constraints, and respect for faculty preferences as soft constraints.

4.2 System Architecture

System architecture is an essential part of development, defining how different components such as the user interface, database, and algorithm interact with each other, ensuring smooth data flow and reliable communication. A well-structured architecture improves modularity, scalability, and maintainability, making it easier to manage and extend the system in the future. Here in web based automatic timetable scheduler, system architecture is essential to clearly organize the flow between data storage, optimization logic, and user interaction, resulting in efficient and conflict-free timetable generation.

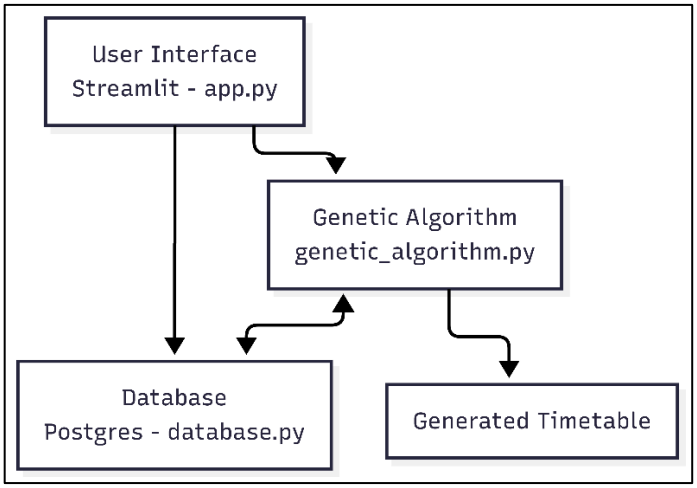

Figure 4.1: System Architecture of Web application

The system architecture of the web application, as illustrated in Figure 4.1, is designed using a modular three-layer architecture to ensure clarity, scalability, and efficient management of the scheduling process. The frontend consists of a Streamlit-based web

interface that provides a simple and intuitive platform for user interaction, allowing users to access the system seamlessly. The backend, developed in Python, manages the core application logic and facilitates communication with the PostgreSQL database, ensuring reliable and persistent data storage. Finally, the Genetic Algorithm module is responsible for generating and optimizing academic timetables, producing conflict-free schedules that satisfy institutional constraints.

4.3 User Interface

The application uses Streamlit as the UI framework. Streamlit provides a single-file-like app structure that integrates form inputs, tables, and buttons with minimal boilerplate and lets users immediately see changes in a live web app. The UI is responsible for data entry such as departments, semesters, faculty, courses, theory & lab mappings, faculty preference entry, and triggering timetable generation.

- **Authentication:** Users register and log in via username/password forms. Successful authentication enables access to management panels.

- **Department & Semester Management:** CRUD operations for departments and semester entries, with instant table views reflecting updates.

- **Faculty Management:** Add, edit, and delete faculty records. Each entry includes name, employee ID validation, and department association.

- **Course Management:** Define theory and lab courses, specifying hours per week. Institution rules enforce that labs span two consecutive periods.

- **Course–Faculty Mapping:** Assign theory courses to one faculty member and labs to two faculty members. These mappings are the GA's primary input.

- **Faculty Preferences:** Specify blocked or preferred time windows for each faculty member. Preferences feed into the GA's fitness evaluation.

- **Timetable Generation & Display:** A "Generate Timetable" button triggers the GA. Progress feedback is shown via a Streamlit progress bar, and final timetables render as interactive tables.

4.4 Scheduling Logic

The core scheduling logic is implemented as a Genetic Algorithm in genetic_algorithm.py. The GA creates populations of candidate timetables, evaluates them using a penalty-based fitness function, and uses selection, crossover, and mutation to evolve better timetables. The GA is tuned to satisfy hard constraints and soft constraint thus reducing penalties.

- **Chromosome representation:** A chromosome represents a list of scheduled class instances, with each gene encoding one scheduled class, including its assigned day, period, faculty, and type (lab or theory). This structure supports both hard and soft constraint evaluations. Each chromosome encodes a full academic timetable as a list of scheduled class genes.

- **Fitness evaluation:** Fitness evaluation quantifies the quality of a timetable by using penalty-based scoring system. Hard constraints incur high penalties to ensure feasibility, while soft constraints guide the optimization toward preferred scheduling patterns. Hard constraints must be strictly enforced, while soft constraints are preferred but flexible.

- **Population management:** introduces random changes in a chromosome by reassigning new time slots to individual classes with a small probability. This maintains diversity in the population and enables broader search of the solution space.

4.5 Database

The database has been designed using PostgreSQL, a robust relational database management system. The schema is normalized and uses primary keys, foreign keys to maintain consistency and avoid redundancy. The database consists of multiple interrelated tables that store information about users, departments, faculty, courses, and their mappings. The overall structure of these tables, ensuring data consistency and integrity to support conflict-free timetable generation, is detailed in Table 4.1 which presents database schema description.

Table 4.1: Database Schema Description

Table Name	Attributes	Purpose
users	id, username, password_hash	Stores login credentials of the system users.

departments	id, name	Stores information about different academic departments.
semesters	id, semester_number	Maintains details of the current semesters.
faculty	id, name, emp_id, department_id	Stores detailed information of faculty members.
courses	id, code, name, hours_per_week, type	Contains information about all courses.
theory_mappings	id, semester_id, course_id, faculty_id	Defines the relationship between theory courses, the semesters, and the faculty assigned.
lab_mappings	id, semester_id, lab_course_id, faculty_id_1, faculty_id_2	Defines the relationship between lab courses, semesters, and the faculty members assigned.
faculty_preferences	id, faculty_id, day, preference_type	Stores faculty preferences regarding their availability for classes.

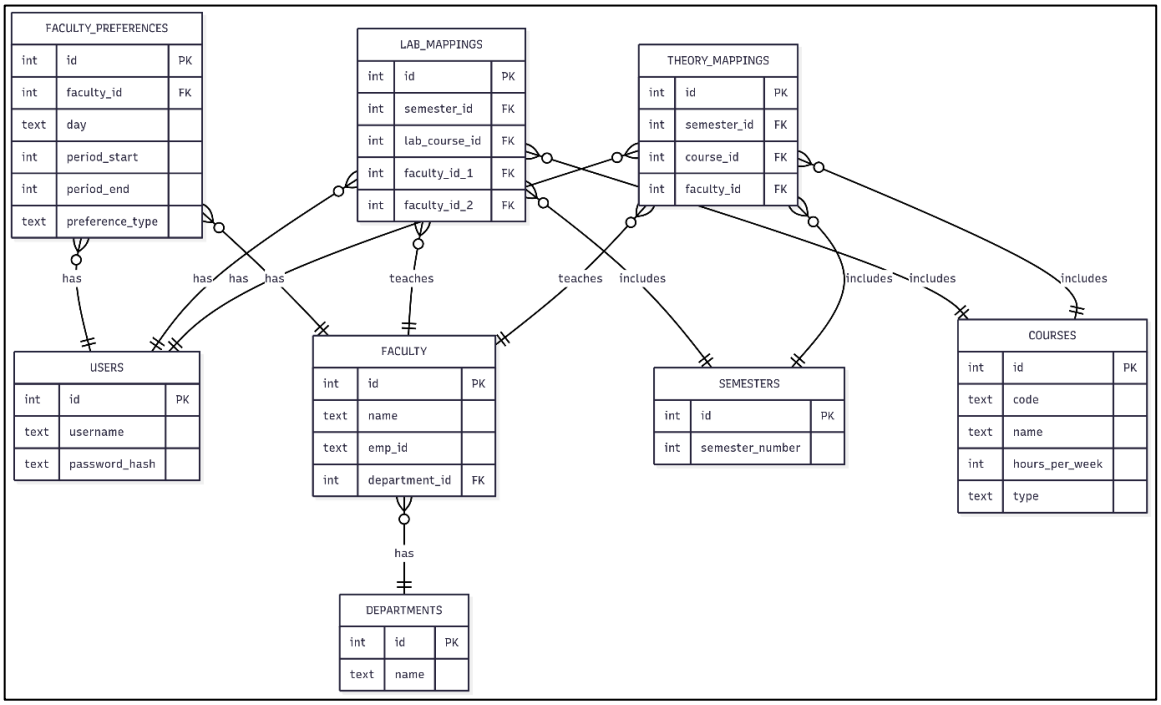

Figure 4.2: ER Diagram

The relationships among various academic entities are depicted in Figure 4.2, where each department comprises multiple faculty members, and each faculty member can be assigned to one or more courses. Courses are linked to specific semesters, ensuring that courses are scheduled in the correct academic term. Theory and lab courses are mapped separately to faculty through the theory_mappings and lab_mappings entities, allowing for both individual and team teaching. Additionally, the faculty_preferences entity captures each faculty member's availability, which the scheduling system considers while generating the timetable. The user's entity ensures secure access to the system by managing login credentials. Together, these relationships form a cohesive structure that maintains data consistency and supports the generation of conflict-free academic timetables.

4.6 Flow Chart

The flowchart represents the logical flow of the system and it offers a clear, structured, and easy-to-understand overview of the process. It provides overview of the sequence of operations, decision points, and iterative cycles involved in timetable generation. This helps to simplify abstract concepts, making the working of the system accessible improving the overall readability and highlights the step-by-step execution logic, and serve as an effective tool for future reference.

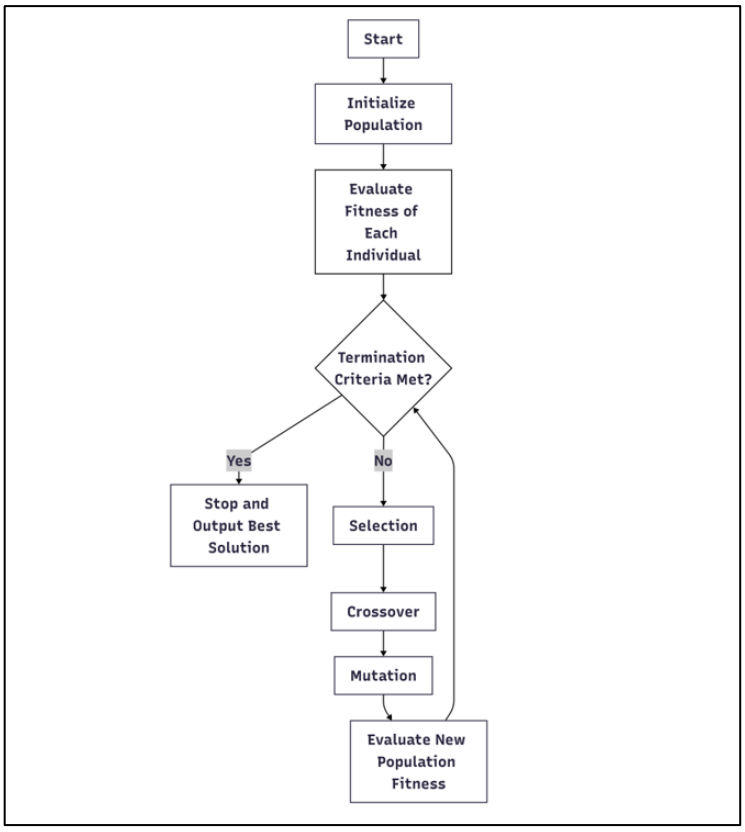

Figure 4.3: Flowchart of Genetic Algorithm Process

The working cycle of the Genetic Algorithm used in the timetable scheduler is depicted in Figure 4.3. The process begins with the initialization of a random population, where each individual represents a possible timetable arrangement. Once initialized, the fitness of each individual is evaluated based on how well the timetable satisfies both hard and soft scheduling constraints. After evaluation, the algorithm checks the termination condition, which is typically defined as reaching the maximum number of generations or achieving a timetable with an acceptable fitness score. If the termination condition is not met, the GA proceeds with selection, where the fittest individuals are chosen as parents. These parents undergo crossover, combining parts of their structure to create offspring solutions, followed by mutation, which introduces small random variations to maintain diversity in the population. The new population is then evaluated again, and the cycle continues iteratively. Once the termination criteria are satisfied, the algorithm stops and outputs the best solution, which represents the most optimal timetable generated during the evolutionary process.

4.7 Sequence Diagram

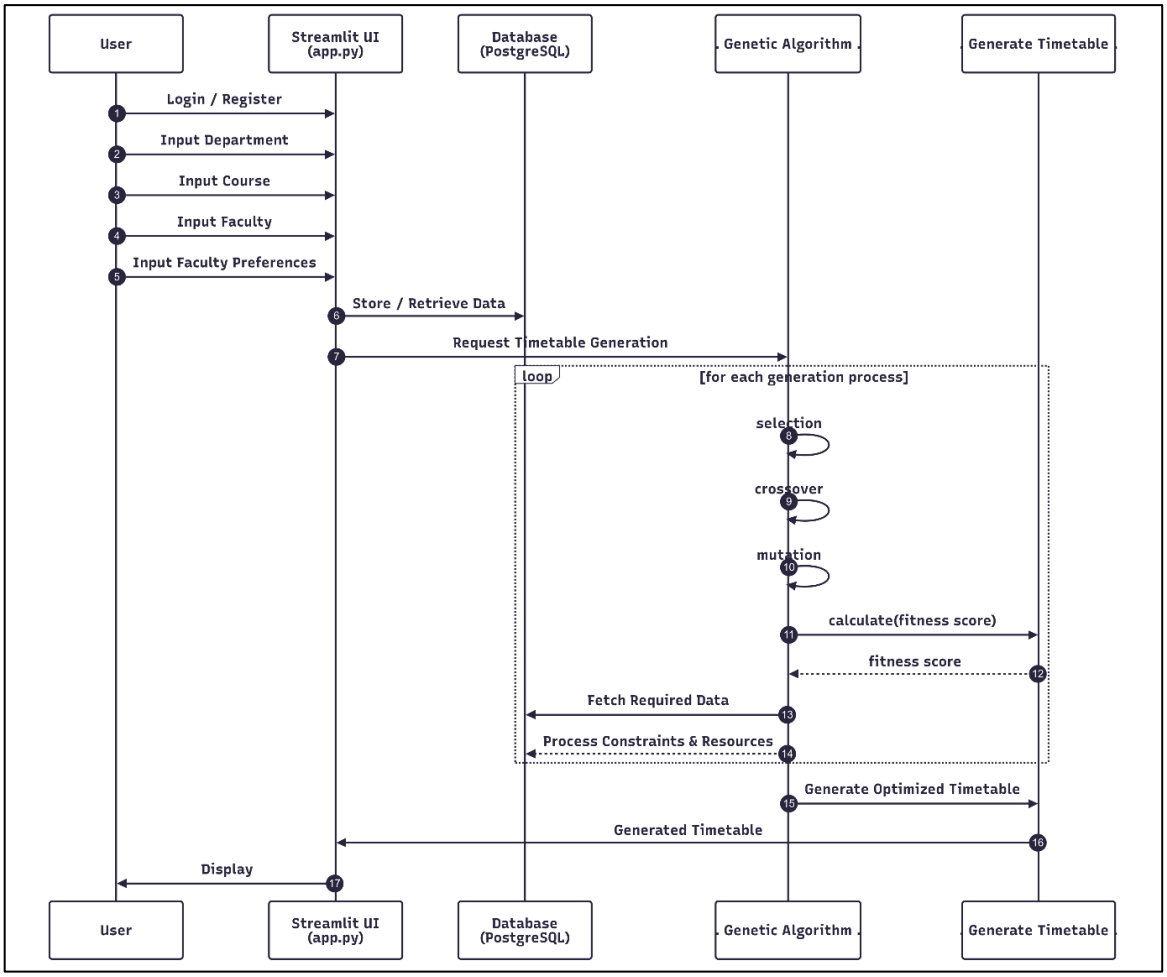

Figure 4.4: Sequence Diagram of the Web-Based Automatic Timetable Scheduler

The workflow of the system, including the step-by-step interaction between the user, interface, database, genetic algorithm, and the generated timetable, is illustrated in Figure 4.4, which presents the sequence diagram of the Web-Based Automatic Timetable Scheduler.

The process begins when the user logs into the system by providing valid credentials through the interface. Upon successful login, the user is granted access to system functionalities. The user enters essential academic inputs such as department, course, faculty, and faculty preferences. These inputs are processed through the stream lit user interface, which stores and retrieves data from the PostgreSQL database to ensure all required academic details and constraints are available.

Once the data is stored and reviewed, the user requests timetable generation. The genetic algorithm works iteratively by performing selection, crossover, and mutation operations to evolve multiple candidate solutions and calculate their fitness scores. Throughout this process, the algorithm interacts with the database to fetch resources and validate constraints, ensuring that no conflicts exist. Finally, the optimized timetable is generated and returned to the user via the UI, where it is displayed in an organized and accessible format.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Introduction

System Implementation refers to the process of transforming the proposed design and architecture into a fully functional application that can be used by users for automated timetable generation. The Web-Based Automatic Timetable Scheduler, implementation involves the development of a Streamlit-based user interface through which users interact with the system, the integration of a Genetic Algorithm module that performs the optimization and scheduling logic, and the PostgreSQL database that persistently stores institutional data such as departments, courses, semesters, faculty records, and individual preferences.

5.2 System Workflow Overview

The system's workflow provides a comprehensive view of how the Web-Based Automatic Timetable Scheduler operates from start to finish. The sequence of processes begins with secure user authentication, followed by the configuration of departments, semesters, courses, and faculty. The workflow also covers course–faculty mappings, the integration of faculty preferences, and the preparation of scheduling data before execution of the Genetic Algorithm. Each stage is explained to show how the user interface, database, and scheduling logic interact seamlessly. By following this structured process, the system ensures that the generated timetables are accurate, optimized, and aligned with both institutional requirements and faculty constraints.

5.2.1 User Authentication

The system begins with a secure login and registration mechanism developed in Streamlit. New users can register by providing a username and password, while returning users can directly login. To protect user credentials, all passwords are encrypted using the bcrypt hashing algorithm before being stored in the users table of the database. During login, the input is validated against the stored records, and upon successful authentication, the session state is updated, granting the user access. This ensures that only authorized users are allowed to generate timetables, thereby safeguarding institutional data and preventing misuse of the system.

5.2.2 Department, Semester, Faculty, and Course Management

Once authenticated, the user configures the basic academic structure of the institution. This includes managing departments, semesters, faculty, and courses. Departments can be dynamically added, updated, or removed, while semesters are recorded as numeric values (e.g., 3, 5, 7). Faculty members are registered with their name, faculty ID, and department, ensuring proper identification and categorization. Courses are defined as either theory or lab, with weekly hours assigned for each. To maintain academic integrity, lab courses are always fixed at two consecutive periods. All these details are stored in database tables, and their management is facilitated through CRUD operations such as add department, update faculty, and delete course. This structured approach allows the scheduler to build a reliable foundation for timetable generation.

5.2.3 Course–Faculty–Semester Mapping

The next stage involves mapping courses to faculty and semesters to establish teaching responsibilities. Theory courses are assigned to exactly one faculty member, while lab courses are mapped to two different faculty members to reflect the collaborative nature of practical sessions. The system implements strict validation checks to prevent errors such as duplicate entries or invalid mappings. By ensuring uniqueness and consistency, these mappings provide a structured link between academic courses, faculty availability, and semester requirements.

5.2.4 Faculty Preferences

To make the scheduling process realistic and acceptable to staff, the system incorporates faculty preferences and availability. Faculty members can specify blocked slots, which indicate the times they are unavailable, and preferred slots, which highlight their favourable teaching hours. These preferences are stored in the faculty preferences table and directly influence the optimization process carried out by the Genetic Algorithm. By honouring these preferences, the scheduler produces timetables aligned with the institutional workflow and faculty satisfaction.

5.2.5 Data Loading for Scheduling

Before the scheduling process begins, all necessary data is consolidated through the data loading stage. The function load_all_data() retrieves records from multiple database tables, while get_classes_to_schedule() prepares ScheduledClass objects that represent individual course-to-semester assignments. Each object carries attributes such as faculty, semester,

workload, and course type, ensuring that the scheduler operates with a complete and structured dataset. This preprocessing step is critical because it transforms static database entries into dynamic scheduling units suitable for optimization.

5.2.6 Genetic Algorithm Execution

At the core of the timetable scheduler lies the Genetic Algorithm, which generates optimized and conflict-free timetables. The GA begins with population initialization, where a random set of candidate timetables is created. Each timetable, or chromosome, consists of multiple Scheduled Class instances randomly assigned to valid days and slots. These timetables are then evaluated using the Timetable Fitness class, which applies both hard constraints and soft constraints. Hard constraints include ensuring no semester overlaps, preventing faculty double-booking, blocking unavailable hours, and enforcing consecutive slots for lab sessions. Soft constraints, such as honouring preferred teaching hours and avoiding labs during lunch breaks, are considered less strictly but still influence fitness scores.

Following fitness evaluation, the algorithm applies tournament selection to identify high-quality candidate timetables as parents. Through crossover, portions of two parent solutions are exchanged to create new child timetables, combining beneficial features from both. To maintain diversity, mutation introduces small random changes, such as moving a class to another valid slot. This cycle of selection, crossover, mutation, and evaluation continues iteratively across multiple generations. The algorithm continuously tracks the best timetable until the termination condition is met, either by convergence or by reaching the maximum number of generations. At this point, the fittest timetable is selected as the final output.

5.2.7 Display of Timetable

The final timetable is formatted and presented to the user through the Streamlit interface. The generated schedule is displayed in a grid or tabular layout, making it easy to review allocations for accuracy and consistency. users can verify that all faculty, courses, and semesters have been properly scheduled according to institutional requirements and faculty preferences. This final stage ensures that the implementation not only generates optimized timetables but also delivers them in a user-friendly and actionable format.

5.3 Formula

The efficiency of the Web-Based Automatic Timetable Scheduler largely depends on the fitness function, which serves as the mathematical representation of timetable quality. The fitness function evaluates each candidate timetable generated by the Genetic Algorithm and assigns it a score based on how well it satisfies the defined constraints. A higher fitness score indicates a timetable that is free of conflicts and aligned with institutional requirements, while lower scores correspond to timetables with multiple violations. The general representation of the function can be expressed as:

$$Fitness = -\sum PenaltiesFitness$$

Here, penalties are applied whenever a constraint is violated. By minimizing penalties, the system indirectly maximizes timetable quality. The function differentiates between hard constraints and soft constraints. This balance ensures that the Genetic Algorithm evolves towards solutions that are both feasible and practical.

Hard Constraints:

- **Semester and Faculty Collision:** A penalty is applied if two classes are scheduled in the same semester at the same time, or if a faculty member is assigned to two places simultaneously. These collisions are unacceptable and heavily penalized.
- **Blocked Hours:** Faculty members can specify time slots when they are unavailable. If a class is assigned during these periods, the timetable receives a significant penalty.
- **Invalid Lab Allocation:** Lab sessions must always be scheduled in two consecutive periods. Any violation of this rule results in a penalty.

Soft Constraints:

- **Faculty Preferences Not Met:** Faculty members may request preferred teaching slots. If these preferences are ignored, the fitness function imposes a smaller penalty compared to hard constraints, ensuring flexibility.
- **Lab Scheduled Across Breaks:** Lab sessions that overlap with lunch or recess breaks are discouraged. While not strictly invalid, such allocations reduce timetable practicality and are therefore penalized.

The penalty scheme is designed such that hard constraints incur high penalties to strongly discourage infeasible timetables. For example, semester or faculty time conflicts result in

a penalty of +1000, while invalid lab durations or improper scheduling of labs outside consecutive slots impose a penalty of +500. On the other hand, soft constraints are assigned smaller penalties to provide flexibility without compromising feasibility. Ignoring a faculty's preferred teaching slot results in a penalty of +10, while scheduling labs across lunch or recess breaks adds a penalty of +5.

The overall fitness score is the negative sum of these penalties. A timetable with fitness ≥ 0 indicates that no hard constraints have been violated, making it feasible. Among feasible solutions, the one with the least number of soft constraint violations is considered the most optimal. This design of the fitness function ensures that the scheduler consistently prioritizes conflict-free timetables while still accommodating institutional preferences and practical usability.

5.4 Implementation Flow

The implementation flow of the Web-Based Automatic Timetable Scheduler provides a high-level overview of how the system functions from start to finish. It highlights the sequence of steps followed by the user, beginning with login, data entry, and preference management, and ending with timetable generation and display. This flow ensures that all essential tasks are performed in an organized order, integrating user interaction with the algorithm to deliver a complete scheduling solution.

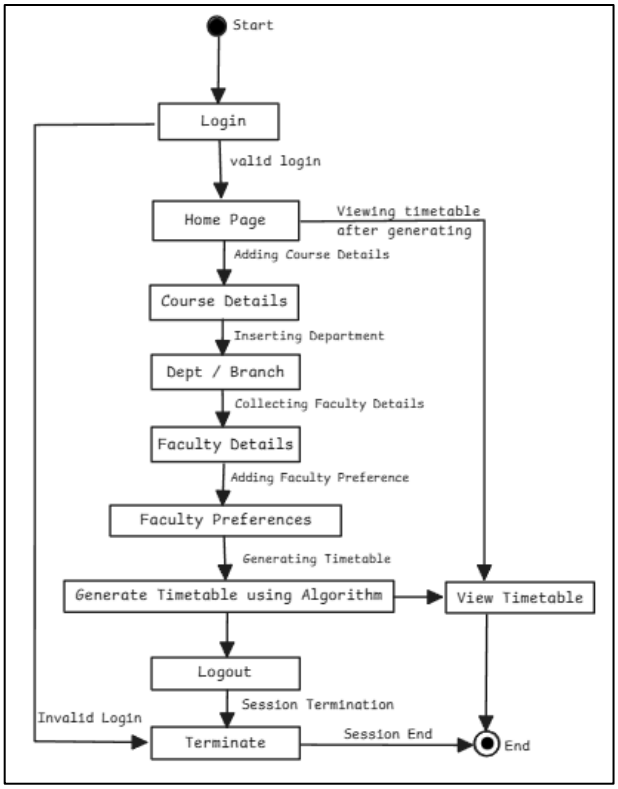

Figure 5.1: Implementation of the Web-Based Automatic Timetable Scheduler

The Data flow of the Web-Based Automatic Timetable Scheduler Figure 5.1, illustrates the overall sequence of operations carried out by the system, beginning from login and ending with timetable generation and display. The process starts with the user attempting to log in. If the login is valid, the user is directed to the home page where different configuration tasks can be performed. These tasks include entering course details, inserting department information, collecting faculty details, and adding faculty preferences. Once all required data is entered, the system proceeds to the next stage where the timetable is generated using the Genetic Algorithm. The generated timetable is then formatted and displayed to the user for verification and use. In case of invalid login, the system immediately terminates the session to prevent unauthorized access. After viewing the timetable, the user has the option to log out, which ends the session securely. This workflow ensures that the scheduling system operates in a logical, step-by-step manner, integrating user interaction, data entry, algorithm execution, and output visualization into a seamless process. The accompanying flowchart clearly represents these interactions, making the flow of implementation easy to understand at a high level.

CHAPTER 6

SYSTEM TESTING

6.1 Introduction

System testing is one of the most crucial phases of the software development lifecycle. It ensures that the complete and integrated system works as intended, meeting both functional and non-functional requirements. For the web-based automatic timetable scheduler, system testing was particularly important since the system directly impacts academic planning, scheduling efficiency, and faculty management.

The primary objectives of system testing were to validate that the scheduler generates conflict-free timetables, handles faculty availability constraints, adheres to the requirement of sequential slot allocation, and provides a user-friendly interface. A combination of unit testing, integration testing, functional testing, and overall system testing was employed to comprehensively evaluate the scheduler's performance and reliability.

6.2 Unit Testing

Unit testing involves testing individual components or modules of the system in isolation to ensure they function correctly. Here, unit testing focused on two major components: the Python back-end modules and the Streamlit-based front-end interface.

The Python back-end formed the core of the scheduler, handling data retrieval, validation, and timetable generation using the Genetic Algorithm.

- **Fitness Function:** Ensured that penalties for conflicts faculty overlap, and lab schedules were correctly applied.
- **Database Operations:** Verified that data retrieval functions such as fetching faculty details, course mappings, and preferences worked correctly without returning null or incorrect values.
- **Algorithmic Components:** Tested crossover and mutation functions to confirm that new candidate timetables were valid and maintained scheduling rules.

The Streamlit interface provided the user interaction. Unit tests for this component focused on input validation and display correctness. Specific tests included:

- **Form Validation**: Verified that all mandatory fields (e.g., faculty name, course selection, semester details) were required before submission.
- **Error Messages**: Checked that clear, user-friendly messages were displayed for invalid inputs (e.g., missing course mapping or duplicate entries).
- **Output Display**: Confirmed that generated timetables were displayed correctly within the Streamlit dashboard, maintaining clarity and alignment.

6.3 Integration Testing

Integration testing was carried out once individual modules were validated, with the objective of ensuring that the modules interacted seamlessly. The focus areas included:

- **Database and Algorithm Integration**: Ensured that real-time faculty-course data from the database was correctly passed into the scheduling algorithm.
- **Algorithm and Interface Integration**: Verified that generated timetables were correctly displayed on the Streamlit interface without mismatched or missing data.

6.4 Functional Testing

Functional testing validated the system against the defined requirements. The main areas tested were:

- **Conflict-Free Allocation**: The system correctly prevented double-booking of faculty across different batches.
- **Faculty Availability**: The scheduler respected unavailability constraints, ensuring no allocations during blocked slots.
- **User Interaction**: Verified that users could easily input data, generate timetables, and export results with minimal steps.

Each function was tested, including typical academic data and simulated different scenarios. The results indicated that the system consistently fulfilled functional requirements.

6.5 System Testing

System testing encompassed the evaluation of the scheduler as a whole in a real-world environment. Unlike unit and integration testing, which focused on individual modules,

system testing validated the complete functionality and performance of the integrated application.

- The scheduler could handle multiple batches and faculty members simultaneously without conflict.
- Timetable generation was completed within a reasonable time frame, even for larger datasets.
- Error handling mechanisms, such as invalid input warnings, worked effectively to prevent system crashes.

The results of system testing demonstrated that the timetable scheduler was stable and reliable.

6.6 Test Cases

To validate the functionality and performance of the Web-Based Automatic Timetable Scheduler, test cases was developed. Each test case outlines the input conditions, expected output, and actual results observed during tests are illustrated in Table 6.1. The successful execution of all test cases confirms the system's correctness, reliability, and readiness for deployment.

Table 6.1: Test Cases

Test Case No.	Description	Input	Expected Output	Actual Output	Result
1	Faculty availability constraint	Faculty A unavailable on Monday Slot 2	Faculty A not scheduled in Slot 2	Faculty A not scheduled in Slot 2	Passed
2	Conflict-free scheduling across semesters	Faculty B assigned to multiple semesters	Faculty B not double-booked in same slot	Faculty B scheduled in separate slots	Passed
3	Invalid input handling	Faculty entry without course mapping	Error message, input rejected	Error message displayed, input rejected	Passed

4	Performance test	9 faculty, 3 Semester, 20 courses	Timetable generates within few minutes/ seconds	Generated in 6.4 seconds	Passed

6.7 Summary of Testing

System testing verified that all components of the Web-Based Automatic Timetable Scheduler for Colleges functioned correctly and cohesively. The system consistently generated conflict-free timetables, handled faculty constraints effectively, and maintained reliable performance. The system has proven to be robust, accurate, and efficient. It meets all its design and functional objectives, making it fully suitable for real-world academic scheduling scenarios. The successful completion of testing provides confidence that the scheduler can be deployed for institutional use, ensuring automation, accuracy, and improved productivity in timetable generation.

CHAPTER 7

EXPERIMENTAL RESULTS AND SCREENSHOTS

7.1 Introduction

The Web-Based Automatic Timetable Scheduler for Colleges was experimentally evaluated to validate its performance, accuracy, and operational reliability. The experimental phase involved executing the system with institutional data to verify its capability to automatically generate optimized, conflict free timetables while satisfying all defined academic constraints. Screenshots captured during execution illustrate the complete workflow of the system, including data entry, faculty-course mapping, timetable generation, and final result visualization. These experimental outcomes highlight the effectiveness of the Genetic Algorithm in resolving scheduling complexities and demonstrate the seamless integration between the algorithmic logic and the Streamlit based interface. Overall, the results confirm that the developed system performs efficiently, ensuring usability, accuracy, and scalability in real world academic environments.

7.2 Results and Screenshots

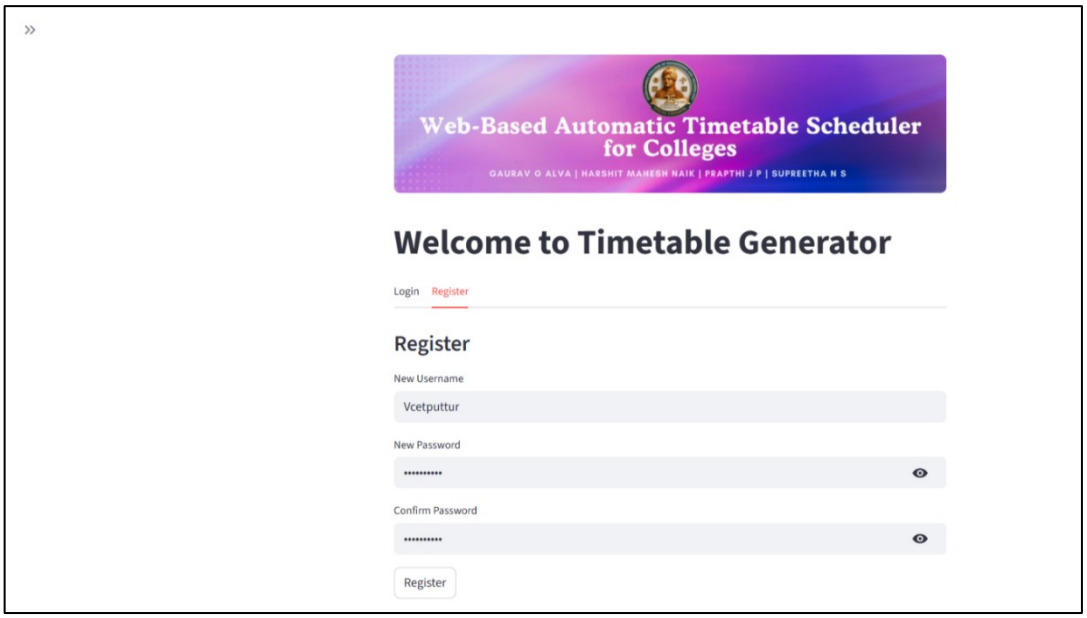

Figure 7.1: User Registration Page

The user registration page of the Web-Based Automatic Timetable Scheduler, as illustrated in Figure 7.1, serves as the entry point for users to register, login and begin interacting with the system.

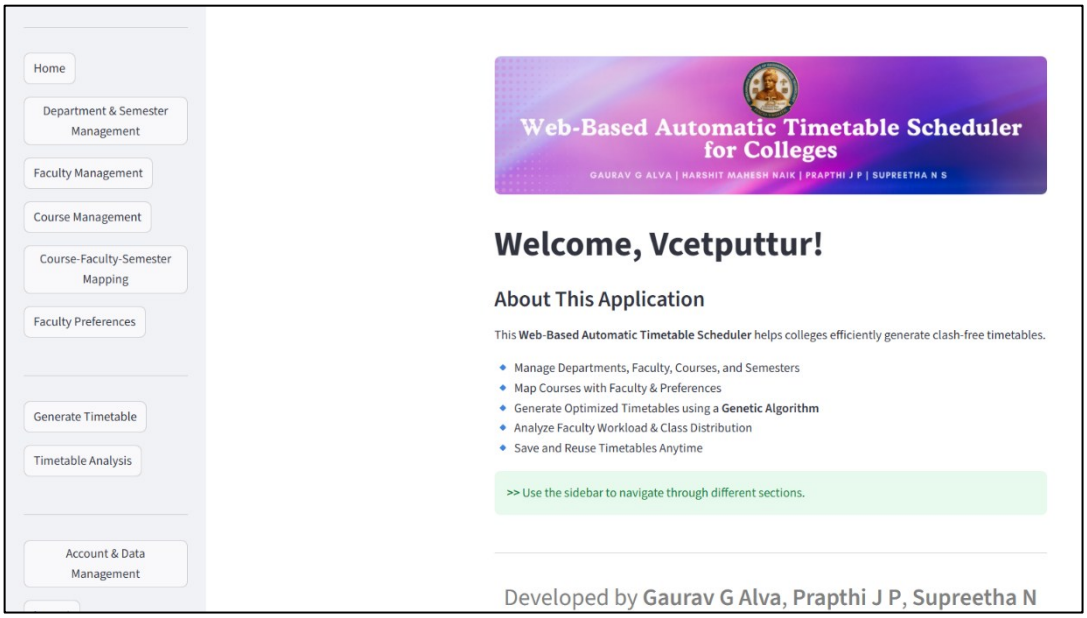

Figure 7.2: User Welcome Page

An overview of the application is shown in Figure 7.2, outlines its core functionalities such as faculty and course management, timetable generation, and analysis. A sidebar is displayed on the left to facilitate easy navigation across different modules.

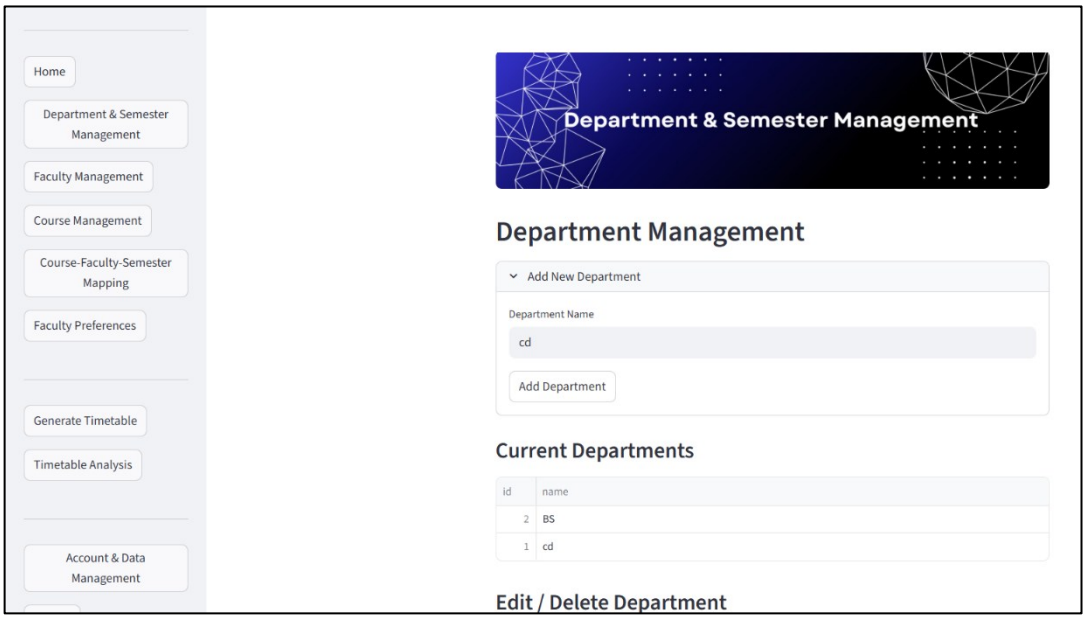

Figure 7.3: Department and Semester Management Page

The Department and Semester Management Page, as shown in Figure 7.3, displays options for adding, viewing, and modifying department and semester details. The interface includes a structured input field for creating new departments and a tabular section for managing existing entries, ensuring organized handling of academic departments.

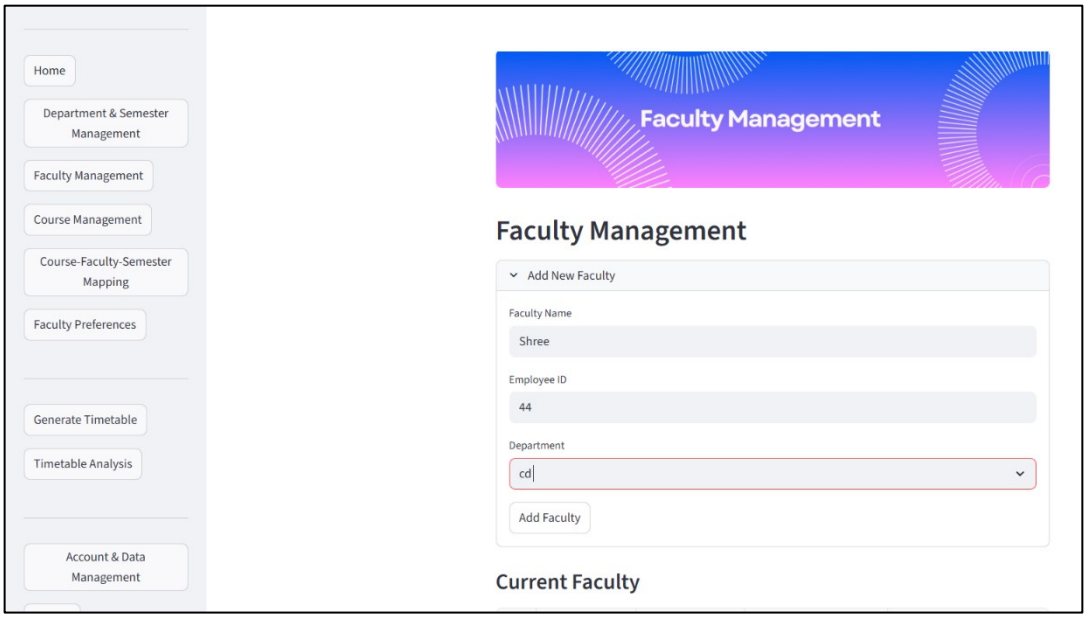

Figure 7.4: Faculty Management Page

The faculty management page allows users to add new faculty details such as name, employee ID, and department as shown in Figure 7.4. It also provides a table to view and manage existing records, ensuring systematic maintenance of faculty information.

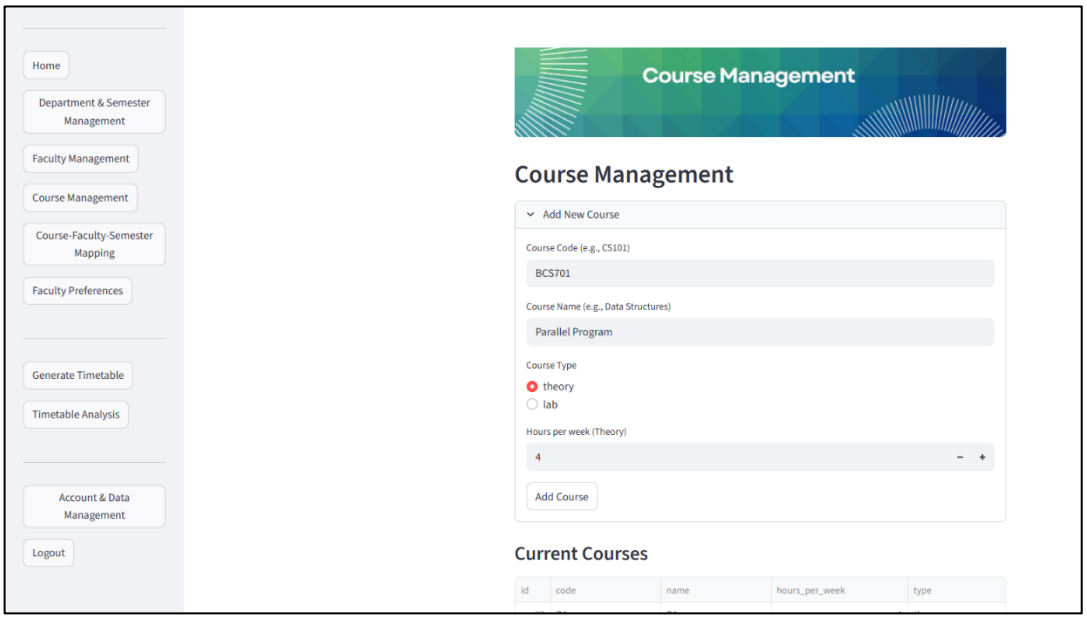

Figure 7.5: Course Management Page

The Course Management page enables users to add and manage course details such as course code, name, type, and weekly hours as shown in Figure 7.5. The section provides structured input fields and a tabular view for maintaining accurate and organized course information.

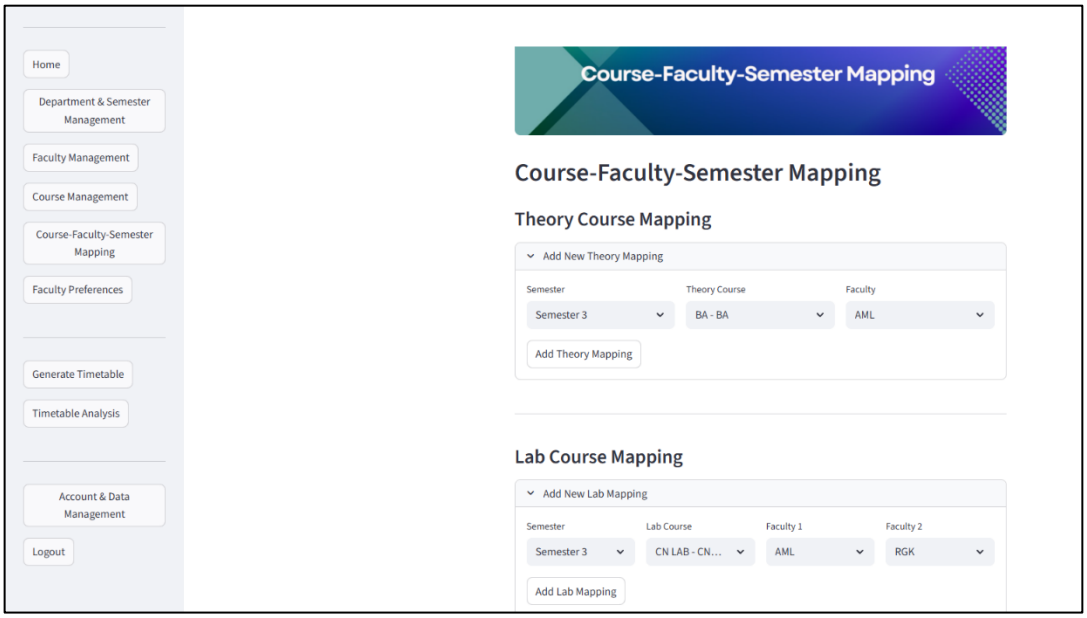

Figure 7.6: Course Faculty and Semester Mapping Page

The Course–Faculty–Semester Mapping page, shown in Figure 7.6, allows users to map faculty members to their respective theory and lab courses across semesters, ensuring accurate assignment of courses and faculty. It maintains proper linkage between academic courses, semesters, and faculty members in the scheduling process.

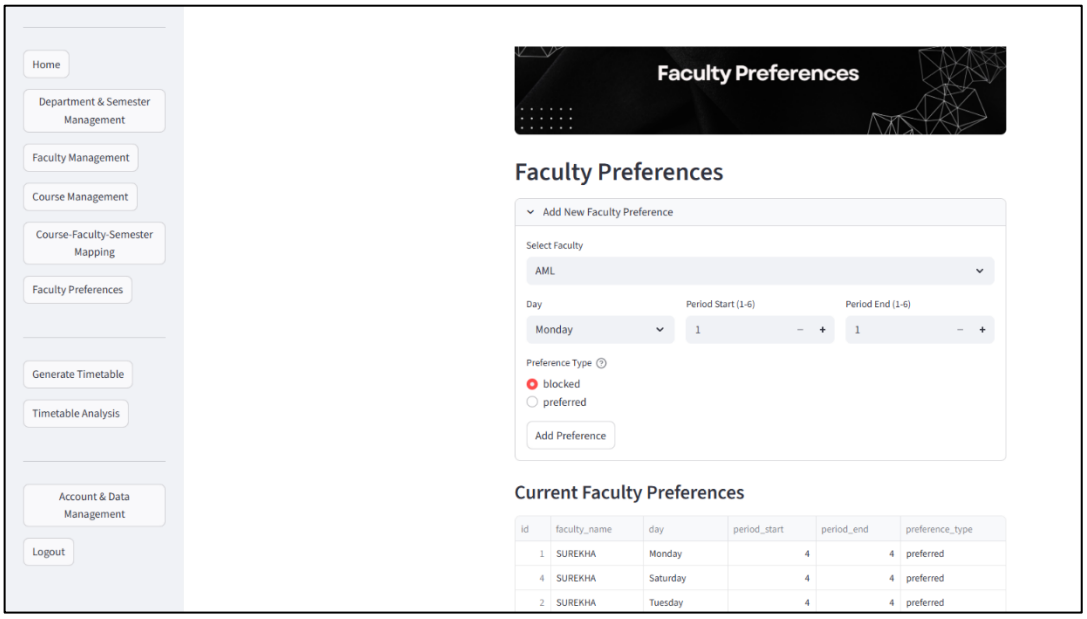

Figure 7.7: Faculty Preference Page

Faculty preferences indicating preferred and blocked time slots are illustrated in Figure 7.7. This information is utilized by the scheduling algorithm to ensure that timetable generation aligns with individual faculty availability and institutional constraints.

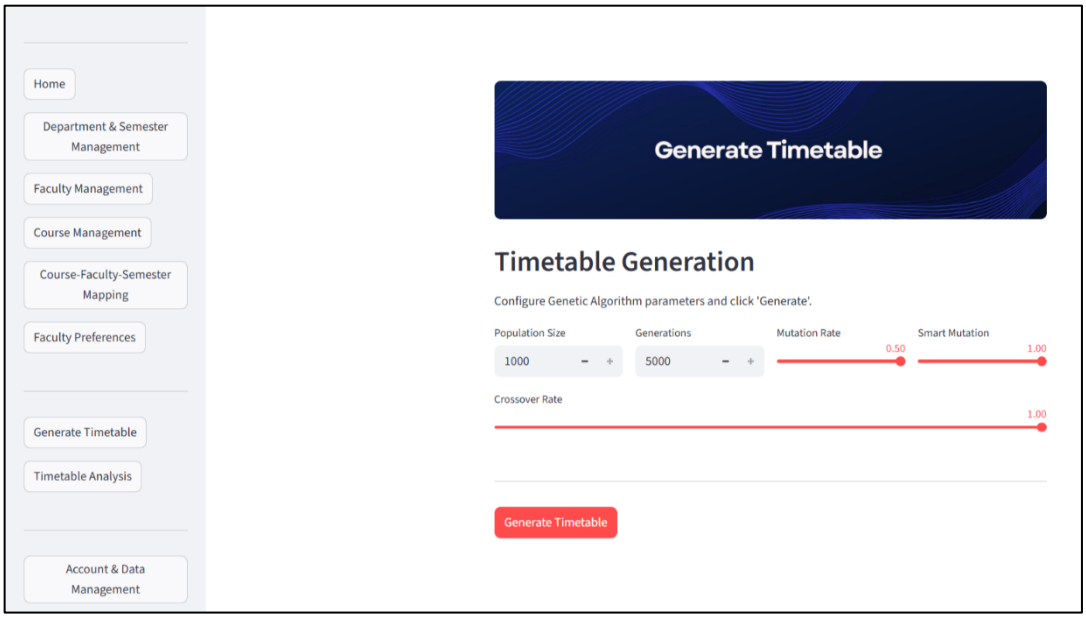

Figure 7.8: Timetable Generation and Display Page

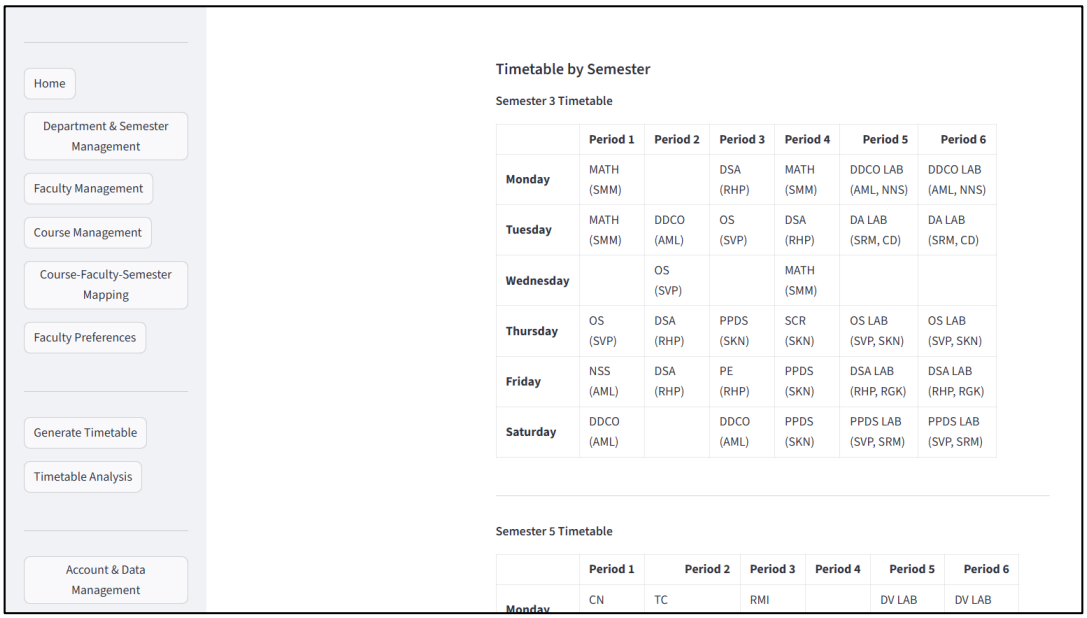

Figure 7.9: Generated Timetable

The configuration and execution of the scheduling algorithm are shown in Figure 7.8, where parameters such as population size, number of generations, mutation rate, and crossover rate are adjusted before generating the timetable. The corresponding results are illustrated in Figure 7.9, displaying the finalized timetables for different semesters. These results confirm that the scheduler successfully produces conflict-free timetables while maintaining balanced faculty and course allocations.

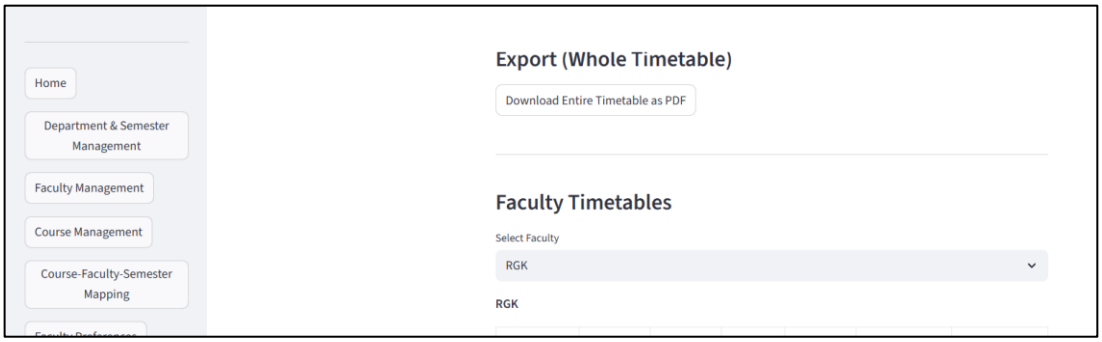

Figure 7.10: Export Timetable & Faculty Timetable

Timetable_20251112_0633

Semester 3

Day	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6
Monday	MATH (SMM)		DSA (RHP)	MATH (SMM)	DDCO LAB (AML, NNS)	DDCO LAB (AML, NNS)
Tuesday	MATH (SMM)	DDCO (AML)	OS (SVP)	DSA (RHP)	DA LAB (SRM, CD)	DA LAB (SRM, CD)
Wednesday		OS (SVP)		MATH (SMM)		
Thursday	OS (SVP)	DSA (RHP)	PPDS (SKN)	SCR (SKN)	OS LAB (SVP, SKN)	OS LAB (SVP, SKN)
Friday	NSS (AML)	DSA (RHP)	PE (RHP)	PPDS (SKN)	DSA LAB (RHP, RGK)	DSA LAB (RHP, RGK)
Saturday	DDCO (AML)		DDCO (AML)	PPDS (SKN)	PPDS LAB (SVP, SRM)	PPDS LAB (SVP, SRM)

Figure 7.11: Downloaded Timetable

Timetable_20251112_0633-Faculty-RGK

Faculty: RGK

Day	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6
Monday		TC (Sem 5)				
Tuesday	TC (Sem 5)					
Wednesday						
Thursday				TC (Sem 5)		
Friday					DSA LAB (Sem 3)	DSA LAB (Sem 3)
Saturday	TC (Sem 5)					

Figure 7.12: Downloaded Faculty Timetable

The timetable represents the final phase of the scheduling process. The export functionality allows users to download the generated timetables in PDF format as shown in Figure 7.10, for record keeping and academic use. The downloaded timetable as shown in Figure 7.11, displays the finalized schedule for each semester, organized systematically by day, slot, and course allocation. In contrast, the faculty specific timetable shown in Figure 7.12 presents individual schedules, detailing sessions assigned to each faculty across the week.

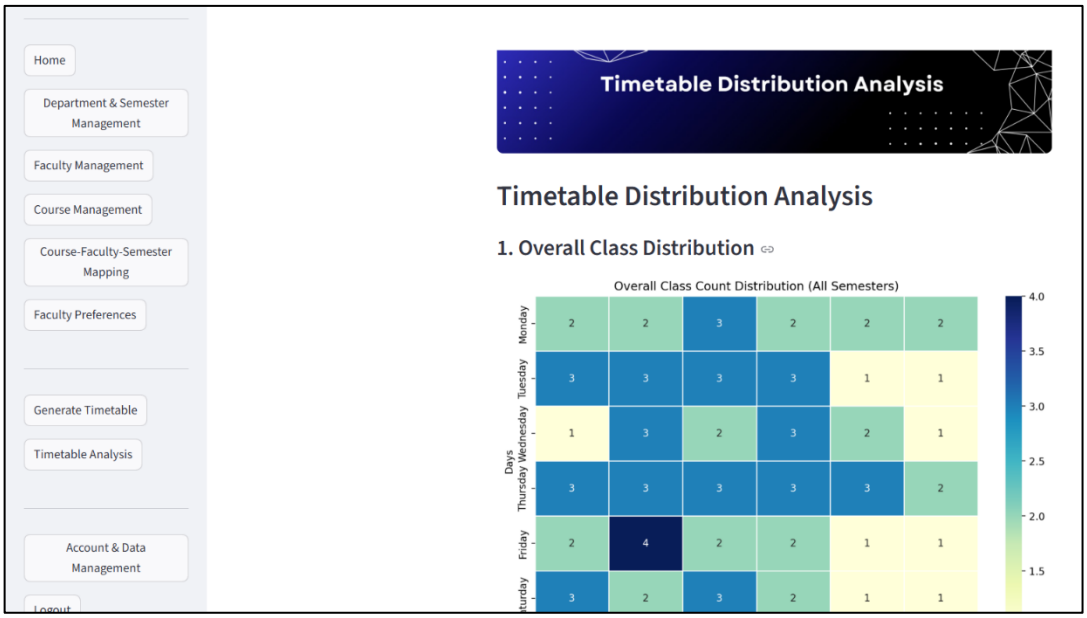

Figure 7.13: Timetable Analysis

The Timetable Analysis page visualizes the overall class count distribution across all semesters as illustrated in Figure 7.13 along with class distribution per semester. The graphical heatmap representation aids in understanding faculty workload balance, peak academic hours, and slot utilization efficiency.

CHAPTER 8

CONCLUSION AND SCOPE FOR FUTURE ENHANCEMENT

8.1 Conclusion

The Web-Based Automatic Timetable Scheduler for Colleges successfully automates the complex and time-consuming process of academic timetable creation. By integrating a Genetic Algorithm with a user-friendly web interface built using Streamlit and Python, the system efficiently generates conflict-free schedules while adhering to institutional and faculty constraints.

The scheduler effectively manages multiple parameters such as faculty availability, course distribution, and, ensuring optimized slot allocation and fairness in workload distribution. Through extensive testing, the system has proven to be reliable, accurate, and efficient in handling real-world academic data.

Overall, the project achieves its primary objectives of minimizing manual intervention, improving scheduling accuracy, and enhancing academic productivity. It serves as a practical solution for colleges and schools seeking to transition from manual to automated timetable generation systems.

8.2 Scope for Future Enhancement

In the future, Web-Based Automatic Timetable Scheduler for Colleges can be integrated with existing Academic Management Systems (AMS) or Enterprise Resource Planning (ERP) tools used by educational institutions. It can also be enhanced with advanced features such as dynamic rescheduling to automatically adjust the timetable in real-time during unforeseen events like faculty leave, holidays, or institutional changes. Additionally, the development of a mobile application using frameworks like Flutter would make the system more accessible and user-friendly for faculty and students.

These enhancements would collectively transform the scheduler into a more adaptive, data-driven, and institution wide solution, capable of meeting the dynamic scheduling needs of modern academic environments.

REFERENCES

[1] Han, Xu, and Dian Wang. "Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming". *Algorithms* 18, no. 3 (2025): 158.

[2] Paramatmuni, Sahith Siddharth, Dumpala Yashwanth Reddy, Elakurthi Sai Spoorthi, Akhil Dharani, and K. Venkatesh Sharma. "Smart Timetable Generation using Genetic Algorithm". *Macaw International Journal of Advanced Research in Computer Science and Engineering* 10, no. 1s (2024): 204-215.

[3] Mahlous, Ahmed Redha, and Houssam Mahlous. "Student timetabling genetic algorithm accounting for student preferences". *PeerJ Computer Science* 9 (2023): e1200.

[4] Alnowaini, Ghazi, and Amjad Abdullah Aljomai. "Genetic algorithm for solving university course timetabling problem using dynamic chromosomes". In *2021 International Conference of Technology, Science and Administration (ICTSA)*, pp. 1-6. IEEE, 2021.

[5] Gore, Bhaven, Disha Shirdhankar, and Giriraj Belanekar. "Institute Timetable Scheduler". *International Research Journal of Engineering and Technology (IRJET)* Volume: 07 Issue: 08 Aug 2020 (2020).

[6] Ghiridhar, S., A. Sachin, M. T. Edwin, and K. N. Unnikrishnan. "Timetable Generation Using Genetic Algorithm for Batches Under APJ Abdul Kalam Technological University". *International Journal of Computer Science and Mobile Computing*, Vol.9 Issue.6, June- 2020, pp. 82-91 (2020).

[7] Assi, Maram, Bahia Halawi, and Ramzi A. Haraty. "Genetic algorithm analysis using the graph coloring method for solving the university timetable problem". *Procedia Computer Science* 126 (2018): 899-906.

[8] Febrita, Ruth Ema, and Wayan Firdaus Mahmudy. "Modified genetic algorithm for high school time-table scheduling with fuzzy time window". In *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, pp. 88-92. IEEE, 2017.

[9] Sampebatu, Limbran, and Aries Kamolan. "Timetable Management Using Genetic Algorithms". *Widya Teknik* 15, no. 2 (2016): 67-72.

[10] Abdelhalim, Esraa A., and Ghada A. El Khayat. "A utilization-based genetic algorithm for solving the university timetabling problem (uga)". *Alexandria Engineering Journal* 55, no. 2 (2016): 1395-1409.

[11] Mittal, Dipesh, Hiral Doshi, Mohammed Sunasra, and Renuka Nagpure. "Automatic timetable generation using genetic algorithm". *International Journal of Advanced Research in Computer and Communication Engineering* 4, no. 2 (2015): 245-248.

[12] Chauhan, Paresh M., Kashyap B. Parmar, and Mahendra B. Mendapara. "Solving time-table scheduling problem by novel chromosome representation using Genetic algorithm". In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pp. 1-6. IEEE, 2015.

[13] Alsmadi, Othman M. K., S. Za'er, Dia I. Abu-Al-Nadi, and Alia Algsoon. "A novel genetic algorithm technique for solving university course timetabling problems". In *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*, pp. 195-198. IEEE, 2011.

[14] Sapru, Vinayak, Kaushik Reddy, and B. Sivaselvan. "Time table scheduling using genetic algorithms employing guided mutation". In *2010 IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1-4. IEEE, 2010.

[15] Colorni, Alberto, Marco Dorigo, and Vittorio Maniezzo. "A genetic algorithm to solve the timetable problem". *Politecnico di Milano, Milan, Italy TR* (1992): 90-060.

PERSONAL PROFILE

Prof. CHAITHANYA D

Assistant Professor, Department of CSE (Data Science), Vivekananda College of Engineering and Technology, Puttur, 574203.

Education qualification: B. E, M. Tech

Area of Interests: Data Mining & Data Analytics

Phone: 9164561651

Email: chaithanyad.cd@vcetputtur.ac.in

GAURAV G ALVA

USN: 4VP22CD019

Phone: 7259130756

Email: gauravalva.me@gmail.com

Address: Laxmi Kripa, Kaipa House, Bellipadi Village, Kodimbady Post, Puttur, D.K 574325

HARSHIT MAHESH NAIK

USN: 4VP22CD022

Phone: 7975517211

Email: harshitnaik762@gmail.com

Address: Banavani road Sirsikar colony Sirsi "Pratiksha building" Sirsi UK 581401

	**PRAPTHI J P** USN: 4VP22CD037 Phone: 9632530837 Email: prapthijaineera@gmail.com Address: Jaineera House, c/o Prasad J P, Niduvatti Village, Kalooru Post, Madikeri, Kodagu 571201
	**SUPREETHA N S** USN: 4VP22CD058 Phone: 9535965726 Email: supreethans4002@gmail.com Address: Nalalu House, Rekhya Post and Village, Belthangady Taluk, D.K 574229

CONFERENCE ATTENDED

Prof. Chaithanya D, Mr. Gaurav G Alva, Ms. Prapthi J P, Ms. Supreetha N S, Mr. Harshit M Naik, "Web-Based Automatic Timetable Scheduler for Colleges", Proceedings of 3rd International Conference DHRISHTI – 2025, held at Vivekananda College of Engineering and Technology, Puttur on 09/08/2025, ISBN: 9878-93-343-5348-8, 2025