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.

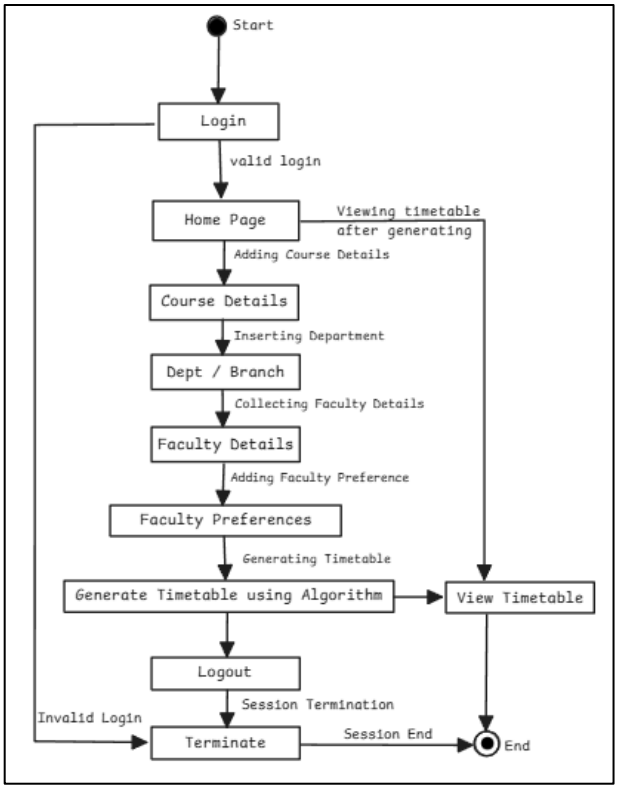

Figure 5.1: Implementation of the Web-Based Automatic Timetable Scheduler

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.