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.

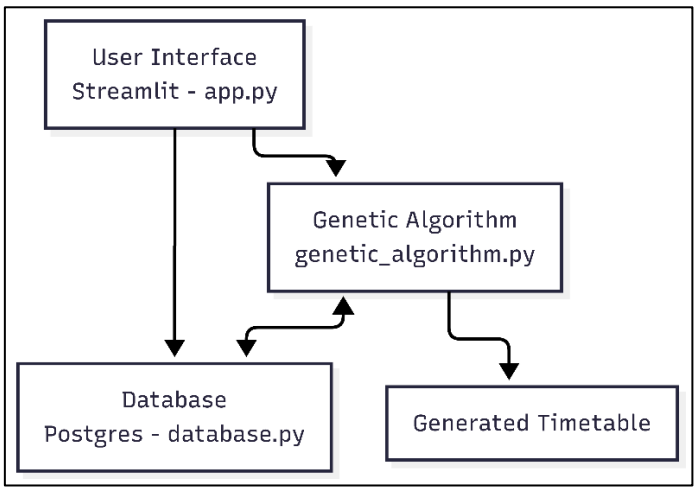

Figure 4.1: System Architecture of Web application

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 (chromosomes), 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.

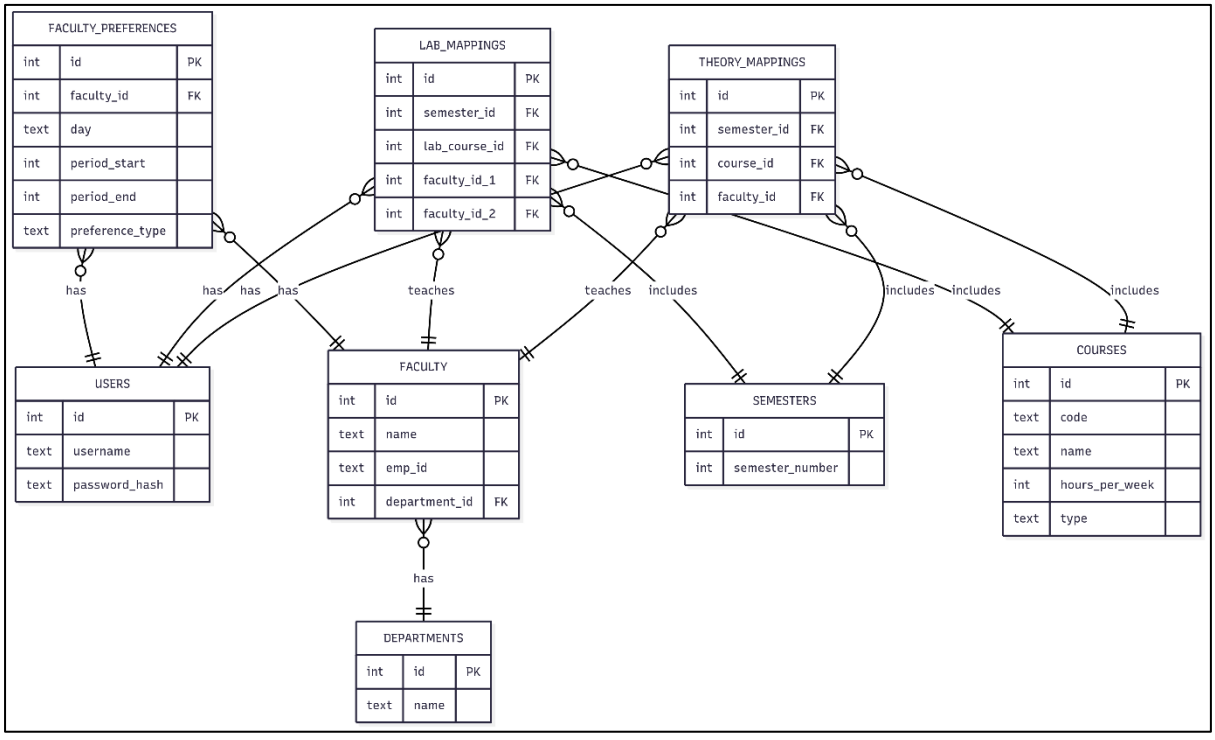

Figure 4.2: ER Diagram

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.

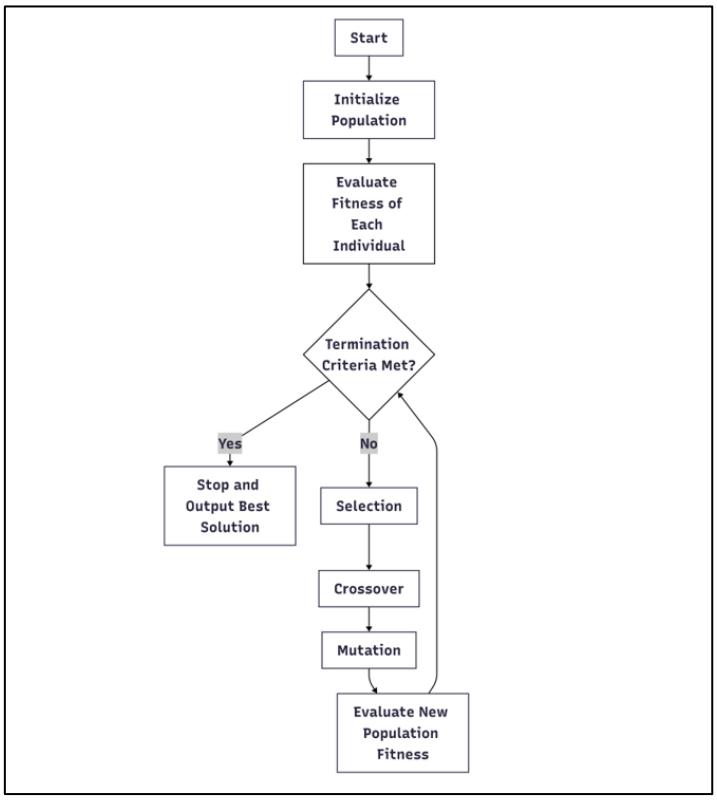

Figure 4.3: Flowchart of Genetic Algorithm Process

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

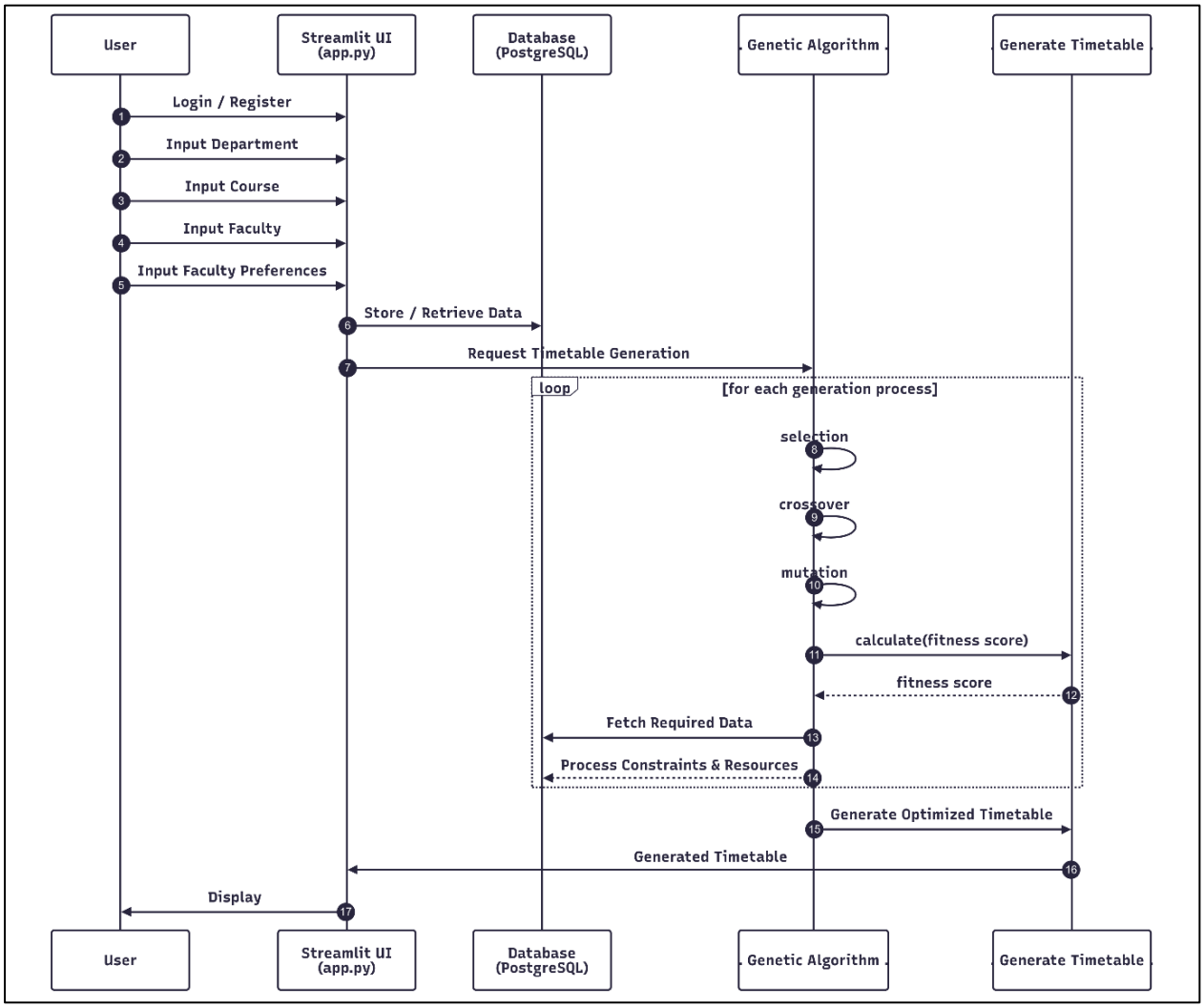

Figure 4.4: Sequence Diagram of the Web-Based Automatic Timetable Scheduler

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.