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.