Software Requirements Specification

for

Web Utility for Time Table Generation

Version 1.0

Prepared by

Group Name: Team-6

Anish Kumar Maganti	**SE22UARI018**	**se22uari018@mahindrauniversity.edu.in**
Sriman Satwik Reddy Chinnam	**SE22UARI166**	**se22uari166@mahindrauniversity.edu.in**
Thodupunury Vamshi Krishna	**SE22UARI198**	**se22uari198@mahindrauniversity.edu.in**
Gopu Venkata Kaashith	**SE22UARI200**	**se22uari200@mahindrauniversity.edu.in**

Instructor:	**Avinash Arun Chauhan**
Course:	**Software Engineering**
Lab Section:	**AI**
Teaching Assistant:	**Nartkannai K**
Date:	**28-02-2025**

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 Introduction

1.1 Document Purpose

This Software Requirements Specification document defines the functional, non-functional, and interface requirements for the Web Utility for Time Table Generation project. The system aims to automate the scheduling process for academic institutions by considering faculty availability, classroom constraints, and course requirements. This document serves as a detailed reference for developers, testers, and stakeholders involved in the project. It outlines the software's purpose, scope, features, and constraints, ensuring alignment between development and institutional needs. This SRS covers the entire system, including the front-end (UI/UX), back-end (API, logic testing), database integration, and role-based access management*. The current version of the document corresponds to Release 1.0, which details the system's core functionalities and technical requirements. Future updates may introduce additional features or optimizations to improve performance and scalability.

1.2 Product Scope

The Web Utility for Time Table Generation is a web-based application designed to automate and optimize academic scheduling for educational institutions. The system leverages constraint-based algorithms to generate conflict-free timetables while considering faculty and classroom availability, and course requirements. The platform is designed to be scalable, efficient, and user-friendly, reducing manual effort and minimizing scheduling conflicts. This system benefits administrators, faculty, and students by providing role-based access to manage, view, and update schedules dynamically. Key objectives include enhancing scheduling accuracy and improving institutional efficiency. By automating the timetable generation process, the system eliminates time-consuming manual adjustments and ensures fair, optimized allocation of resources.

1.3 Intended Audience and Document Overview

This document is intended for various stakeholders involved in the development, implementation, and evaluation of the Web Utility for Time Table Generation. The primary audience includes:

- Developers: To understand the functional and technical requirements for system implementation.
- Testers: To verify that the system meets specified requirements and performs as expected.
- Project Managers: To track development progress and ensure alignment with project goals.
- Clients (Institutional Representatives/Professors): To review the system's capabilities and provide feedback on its effectiveness.
- End Users (Administrators, Faculty, Students)*: To understand how the system operates and interacts with their roles.

This SRS document is structured to provide a comprehensive overview of the system, beginning with an introduction to its purpose, scope, and objectives. It then details specific functional and non-functional requirements, external interfaces, use case models, and system constraints. Readers should start with the introduction and product overview to gain a high-level understanding, followed by functional requirements and system architecture for technical details. Testers and developers should focus on interface and performance requirements, while institutional representatives may prioritize use cases and security considerations.

1.4 Definitions, Acronyms and Abbreviations

Below is a list of abbreviations and acronyms used in this document, sorted alphabetically:
API – Application Programming Interface
CRUD – Create, Read, Update, Delete
DBMS – Database Management System
GUI – Graphical User Interface
NP-hard – A class of computational problems that are at least as hard as the hardest problems in NP (Non-deterministic Polynomial-time)
OAuth 2.0 – Open Authorization 2.0, a protocol for secure authentication
REST – Representational State Transfer, an architectural style for APIs
SRS – Software Requirements Specification
SSR – Server-Side Rendering
UI/UX – User Interface/User Experience

1.5 Document Conventions

The conventions used in this document include:

1.5.1 Formatting Conventions

- The document is written using Times New Roman font, size 11 or 12, with single-line spacing and 1-inch margins on all sides.
- Section and subsection titles follow the numbering format (e.g., 1, 1.1, 1.2, etc.).
- *Italicized text* is used for comments or notes.
- Bold text is used for emphasis on key terms and section headings.
- Tables and diagrams are labeled appropriately for clarity.

1.5.2 Naming Conventions

- Variable and database field names follow CamelCase notation (e.g., FacultyID, ClassroomSchedule).
- API endpoints follow RESTful naming conventions, using lowercase and hyphens (e.g., /get-timetable).
- User roles are defined as Admin, Faculty, and Student* with case-sensitive naming for consistency.

1.6 References and Acknowledgments

This SRS document refers to the following supporting documents and resources:

Statement of Work (SOW) – Web Utility for Time Table Generation

1. Defines the project scope, objectives, timeline, and deliverables.

IEEE Standard for Software Requirements Specifications (IEEE 830-1998)

1. Provides guidelines for structuring and writing SRS documents.

*Student and faculty dashboards will be available in the next release.

2 Overall Description

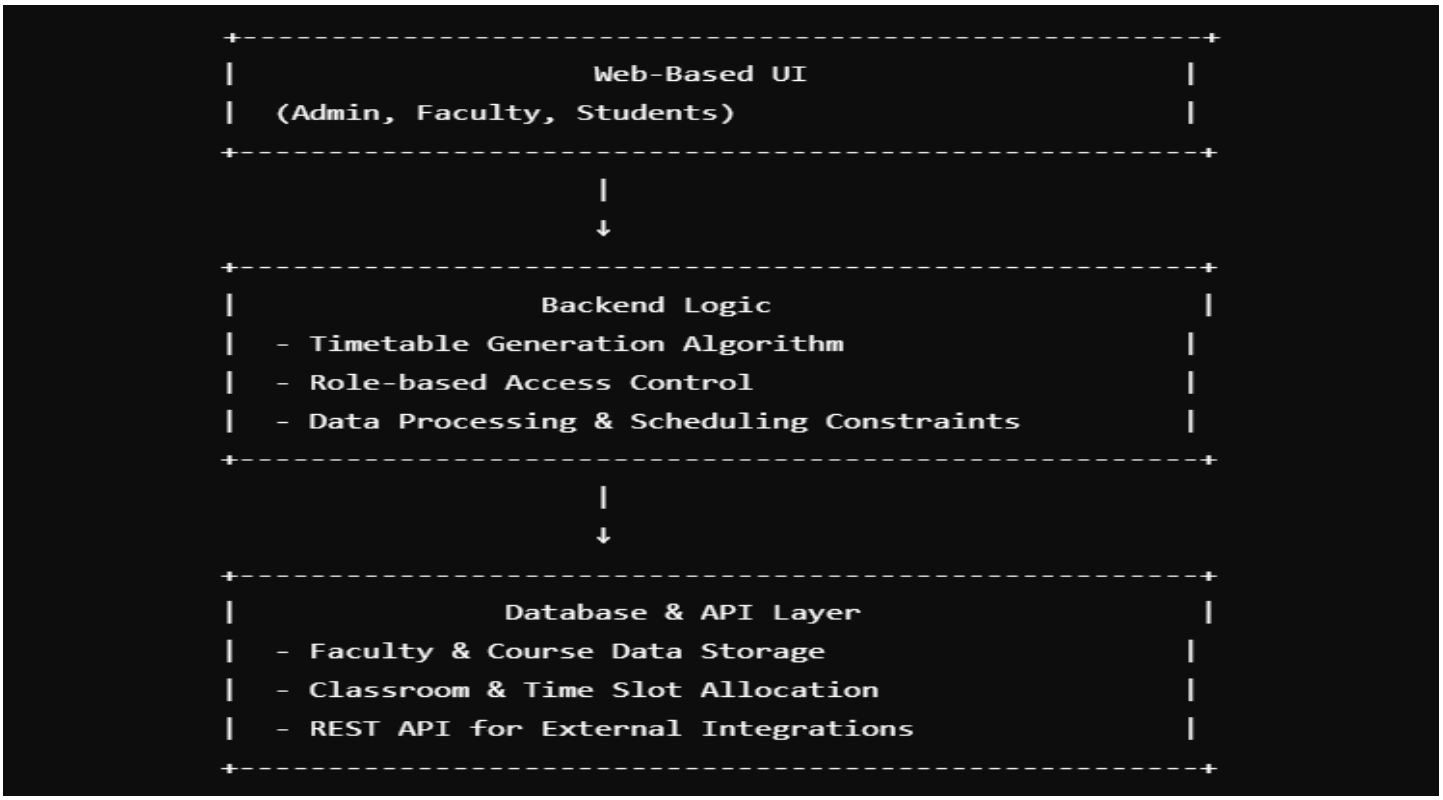
2.1 Product Overview

The Web Utility for Time Table Generation is a new, standalone software product designed to automate academic scheduling for educational institutions. It is developed as an independent solution but can be integrated with existing institutional databases to fetch faculty availability, course details, and classroom resources. The system addresses inefficiencies in manual timetable creation, reducing scheduling conflicts and optimizing resource allocation using constraint-based algorithms.

This system consists of three primary components:

1. User Interface (UI) – A web-based dashboard allowing administrators, faculty, and students to interact with the system.
2. Backend Logic – A scheduling algorithm that processes constraints and generates optimized timetables.
3. Database & API Layer – A cloud-hosted database (MongoDB/Postman) storing faculty, course, and room details, with APIs enabling real-time data exchange.

It is designed for scalability and efficiency, supporting multiple institutions with role-based access. The frontend ensures ease of use along with React.js, while the Node.js/Python backend handles complex scheduling logic.



This architecture enables seamless interaction between users and the scheduling system, ensuring real-time updates and an optimized experience for educational institutions.

2.2 Product Functionality

The Web Utility for Time Table Generation provides the following major functions:

- Automated Timetable Generation – Creates conflict-free academic schedules based on faculty availability, classroom capacity, and course constraints.
- Role-Based Access Control – Allows different user roles (Admin, Faculty, Students)* with specific permissions.
- Faculty and Student Access* – Faculty can view their assigned schedules, while students can check their class timetables.
- Real-Time API Integration – Syncs data with institutional databases for accurate scheduling.
- Constraint Handling and Optimization – Uses algorithms to ensure optimal scheduling while adhering to academic constraints.
- Interactive UI/UX – Provides a user-friendly web-based interface for easy navigation and timetable viewing.
- Performance Monitoring – Ensures efficient timetable generation and API response times.
- Secure Authentication – Uses **OAuth 2.0** for secure login and user authentication.

2.3 Design and Implementation Constraints

The development of the Web Utility for Time Table Generation is subject to several constraints that affect design choices, technology selection, and system implementation. These constraints ensure compatibility, maintainability, and optimal performance of the system.

2.3.1 Methodological Constraints

- The system design must follow the COMET (Concurrent Object Modeling and Architectural Design Method) for software architecture and development. This method emphasizes object-oriented and real-time design, ensuring modularity and maintainability.
    - Reference: Gomaa, H. (2016). "Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures." Cambridge University Press.
- The Unified Modeling Language (UML) must be used for system modeling, including class diagrams, use case diagrams, and sequence diagrams.

    - Reference: Booch, G., Rumbaugh, J., & Jacobson, I. (2005). "The Unified Modeling Language User Guide." Addison-Wesley.

2.3.2 Technological Constraints

- Front-End: The system must be developed using React.js, ensuring a responsive UI
- Back-End: The scheduling logic must be implemented using Node.js, ensuring efficient API processing.
- Database: The system must use MongoDB for structured data storage and fast query execution.
- API Integration: The system must expose and consume REST APIs for institutional database synchronization using Postman
- Cloud Deployment: The system will be deployed using Docker, allowing scalability and containerized execution.

2.3.3 Performance and Hardware Constraints

- The timetable generation process should complete within 30 seconds for institutions with up to 50 courses.
- API response times must not exceed 2 seconds for real-time data interactions.
- The system should function smoothly on modern web browsers (Chrome, Firefox, Edge).

2.3.4 Security and Compliance Constraints

- User authentication must be implemented using OAuth 2.0, ensuring secure access.
- Sensitive user data (faculty schedules, student details) must be encrypted in storage and transit.
- The system must comply with institutional data privacy policies to protect user information.

2.4 Assumptions and Dependencies

The successful development and deployment of the Web Utility for Time Table Generation depend on several assumptions and external factors. Any deviation from these assumptions may impact system functionality, development time-lines, or integration feasibility.

2.4.1 Assumptions

- Institutional Data Availability – The institution will provide accurate and up-to-date data, including faculty availability, course details, and classroom allocations.
- Cloud or Server Deployment – The system will be hosted on a cloud platform (e.g., GCP) or institutional servers, ensuring remote access.
- User Technical Proficiency – End users (Admins, Faculty, Students)* will have basic technical knowledge to operate the web application.
- Stable API Connectivity – Institutional databases will allow seamless API integration without frequent downtimes or access restrictions.
- Timetable Constraints Defined – The institution will clearly define scheduling constraints (e.g., max classes per faculty, room usage policies) to allow proper algorithm optimization.

2.4.2 Dependencies

- Third-Party Authentication (**OAuth 2.0**) – The system relies on **OAuth 2.0** for secure authentication and role-based access management.
- Database System (MongoDB) – The system depends on a NoSQL (MongoDB) for storing timetable data.
- React & Tailwind for Front-End – The UI depends on React.js, meaning any major updates to these frameworks may require adjustments.
- Node.js/Python for Back-End – The scheduling and API logic depend on Node.js or Python, requiring compatibility with the selected database.

**Student and faculty dashboards will be available in the next release.*

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The system will provide an interactive web-based interface accessible via a browser. It will have role-based dashboards for administrators, faculty, and students*, ensuring a seamless user experience.

Main User Interface Components

Admin Dashboard

1. **Functionality**: Allows admins to manage faculty, classrooms, courses, and scheduling constraints.
2. **Interaction**:
    1. Forms for data entry (e.g., faculty availability, room capacities).
    2. "Generate Timetable" button to initiate scheduling.

Faculty Dashboard*

1. **Functionality**: Allows faculty to view their assigned schedules and make requests for changes.
2. **Interaction**:
    1. Table format displaying daily and weekly schedules.
    2. Filter/search options for courses and rooms.

Student Dashboard*

1. **Functionality**: Enables students to view their course schedules.
2. **Interaction**:
    1. Simple timetable display with real-time updates.
    2. Mobile-friendly interface for accessibility.

Login & Authentication Page

1. **Functionality**: Secure login using **OAuth 2.0** authentication.
2. **Interaction**:
    1. Email/password or single sign-on (SSO).
    2. Role-based redirection after login.

3.1.2 Hardware Interfaces

The Web Utility for Time Table Generation is a web-based application, meaning it primarily interacts with hardware through standard computing devices and network infrastructure. The system does not rely on specialized hardware components but must ensure compatibility with commonly used devices.

Supported Hardware Interfaces

Client Devices (Used by Admins, Faculty, and Students)

1. Desktop Computers (Windows)
2. Laptops

Server Infrastructure (Hosting and Computation)

1. Cloud-based or on-premise servers for API hosting and database storage
2. Docker-based containerization for deployment and scalability

3.1.3 Software Interfaces

The Web Utility for Time Table Generation interacts with several software components to ensure smooth functionality, data exchange, and system integration.

External Software Components

Institutional Database System

1. The system connects with an existing institutional database (MongoDB) via RESTful APIs to fetch and update faculty schedules, course details, and classroom availability.

Authentication Service (OAuth 2.0)

1. Secure authentication and role-based access control are implemented using **OAuth 2.0**, allowing users to log in with institutional credentials.

Cloud Deployment Platform

1. The system is hosted on a Docker-based cloud environment or GCP for scalability and remote access.

Postman API Testing

1. All API endpoints are tested using Postman, ensuring smooth integration between the front-end and back-end services.

Internal Software Components

1. **Front-End (React.js)**
    1. Provides an interactive UI for users to manage and view schedules.
2. **Back-End (Node.js/Python)**
    1. Handles timetable generation logic and API requests.

3.2 Functional Requirements

Functional requirements define the specific behavior and capabilities the Web Utility for Time Table Generation must implement. These include user interactions, system processes, and external integrations required for the system to function effectively.

F1: Timetable Generation

- The system shall generate conflict-free academic timetables based on predefined constraints such as faculty availability, classroom capacity, and course schedules.
- The system shall allow manual adjustments to the generated timetable by administrators.
- The timetable generation process shall complete within 30 seconds for institutions with up to 50 courses.

F2: User Authentication & Role Management

- The system shall provide secure authentication via **OAuth 2.0**.
- The system shall allow role-based access for Admin, Faculty, and Students *with specific permissions.
- The system shall restrict unauthorized users from accessing restricted functionalities.

F3: Admin Dashboard & Management

- The system shall allow administrators to add, modify, and delete faculty, courses, and classroom allocations.
- The system shall provide a dashboard for administrators to manage scheduling constraints.
- The system shall allow administrators to override generated timetables if necessary.

*F4: Faculty Dashboard & Schedule Viewing**

- The system shall allow faculty members to view their assigned schedules.
- The system shall allow faculty to request schedule modifications, subject to admin approval.

*F5: Student Access & Timetable Viewing**

- The system shall allow students to view their personal class schedules.
- The system shall provide a mobile-friendly timetable display.

F6: API Integration for Institutional Data

- The system shall integrate with institutional databases via REST APIs to fetch faculty availability, course details, and classroom allocations.
- The system shall synchronize data in real time with external systems.

F7: Performance & Optimization

- The system shall optimize timetable scheduling using constraint-based algorithms.
- The system shall ensure API response times do not exceed 2 seconds.

F8: Security & Data Protection

- The system shall encrypt user credentials and sensitive data in storage and transit.

* Student and faculty dashboards will be available in the next release.

3.3 Use Case Model

- **Actors**: Admin, Faculty, Student*
- **Priority**: High (Ensures proper schedule generation and role-based access)
- **Preconditions**: Users must be registered and logged in. The system must have course and faculty data available.
- **Postconditions**: A timetable is generated, modified, and made accessible to all relevant users. System constraints (faculty preferences, room allocation) are enforced.
- **Basic Flow**:
  1. User Login & Access Control*
     - Admin, faculty, and students log in based on role-based access control.
     - Admins have full access, faculty can modify preferences, and students can only view/export timetables.
  2. Timetable Generation

- The automated scheduler processes course requirements, faculty availability, and room allocations.
- System constraints (e.g., no overlapping schedules, room capacity) are applied.

3. Modification & Constraint Handling
   - Admins and faculty can modify generated timetables.
   - Constraints such as faculty preferences, course combinations, and room assignments are handled.
4. Viewing & Exporting
   - Students and faculty can view the final timetable.
   - Users can export the schedule in different formats (PDF, Excel, etc.).
5. Report Generation
   - The system generates reports on scheduling efficiency, faculty workload, and resource allocation.

- **Exceptions**:
  1. Conflicting Faculty Preferences
     - The system alerts the admin if a faculty preference cannot be accommodated.
  2. Room Overbooking
     - If multiple classes are scheduled for the same room, the system suggests an alternative or requires admin approval.
  3. System Downtime
     - If the system is down, users cannot generate or modify timetables until it is restored.

*Student and faculty dashboards will be available in the next release.

4 Other Non-functional Requirements

4.1 Performance Requirements

The Web Utility for Time Table Generation must meet the following performance requirements to ensure efficiency, scalability, and responsiveness under different usage scenarios.

P1. Timetable Generation Time

- The system shall generate a conflict-free timetable within 30 seconds for institutions with up to 50 courses.
- For larger institutions, timetable generation shall complete efficiently using optimized scheduling algorithms.

P2. API Response Time

- The system shall process API requests and return responses within 2 seconds under normal server load conditions.
- Under peak load (e.g., multiple timetable requests), API latency shall not exceed 5 seconds.

P3. User Interface Responsiveness

- The dashboard and timetable pages shall load within 3 seconds on standard internet connections (10 Mbps and above).
- The system shall maintain smooth interaction for viewing, filtering, and exporting timetables without noticeable delays.

P4. Data Synchronization

- The system shall update schedules and reflect changes within 5 seconds of modification by an administrator.

P5. Concurrent Users Support

- The system shall support at least 100 concurrent users without performance degradation.

4.2 Safety and Security Requirements

The Web Utility for Time Table Generation must ensure the security and privacy of institutional data while preventing unauthorized access and system vulnerabilities. The following safety and security measures must be implemented:

S1. User Authentication and Access Control

- The system shall implement **OAuth 2.0** authentication to ensure secure login.
- Role-based access control (RBAC) shall be enforced to limit permissions based on user roles (Admin, Faculty, Student).
- Failed login attempts shall trigger an account lockout after seven unsuccessful attempts, requiring administrator intervention for unlocking.

S2. Data Encryption and Protection

- All user credentials and sensitive data (faculty schedules, student details) shall be encrypted using AES-256 in storage and TLS 1.2+ in transit.
- Personally Identifiable Information (PII) shall be masked where necessary to prevent unauthorized access.

S3. Data Integrity and Backup

- The system shall implement automated daily backups to prevent data loss in case of server failure.
- In case of a detected data breach or corruption, the system shall trigger an automated rollback to the latest valid backup.

S4. Protection Against Unauthorized Modifications

- Faculty members* shall only be able to request schedule changes, but final modifications must be approved by the admin.
- System logs shall record all modifications to timetables, including timestamps and user details, for audit purposes.

4.3 Software Quality Attributes

The Web Utility for Time Table Generation must meet several software quality attributes to ensure a reliable, maintainable, and adaptable system. The following quality attributes define measurable criteria that will impact the system's usability and effectiveness.

4.3.1 Reliability

- The system shall maintain 99.5% uptime, ensuring minimal downtime for users.

- Timetable generation must produce consistent and accurate schedules, adhering to faculty availability and classroom constraints.
- Automated error logging and failure recovery mechanisms shall be implemented to detect and resolve issues within 5 minutes of failure detection.

4.3.2 Adaptability

- The system shall allow modifications to scheduling constraints (e.g., faculty availability) without requiring code changes.
- The system shall support future expansion, allowing additional academic departments or institutions to be integrated seamlessly.

4.3.3 Maintainability

- The codebase shall follow modular design principles, ensuring ease of updates and debugging.
- API documentation will be provided to simplify future integrations with external systems.
- The system shall have unit and integration tests covering at least 80% of the codebase, ensuring stability during updates.

4.3.4 Usability

- The system shall have a user-friendly UI, ensuring that administrators, faculty, and students can navigate easily.
- The interface will be optimized for desktop to provide accessibility on various devices.

4.3.5 Interoperability

- The system shall support REST API communication to integrate with institutional databases.
- Data exchange formats shall follow JSON/CSV/PDF standards, ensuring compatibility with third-party systems.
- The system shall be compatible with major web browsers (Chrome, Firefox, Edge) without requiring additional plugins.

*Student and faculty dashboards will be available in the next release.

Appendix – Data Dictionary

The following table tracks all variables, states, and functional requirements used in the Web Utility for Time Table Generation system. It includes constants, state variables, inputs, and outputs along with their descriptions, related operations, and requirements.

Variable/State*	Description	Possible Values/States	Related Operations	Functional Requirements
Faculty_ID	Unique identifier for each faculty member	Integer (Auto-generated)	Create, Read, Update, Delete (CRUD)	F2, F3, F4
Course_ID	Unique identifier for each course	Integer (Auto-generated)	CRUD	F3
Classroom_ID	Unique identifier for each classroom	Integer (Auto-generated)	CRUD	F3

Variable/State*	Description	Possible Values/States	Related Operations	Functional Requirements
User_Role	Defines user role (Admin, Faculty, Student)	"Admin", "Faculty", "Student"	Role-based access	F2
Timetable_ID	Unique identifier for each generated timetable	Integer (Auto-generated)	Create, Update, View	F1, F5
Schedule	Stores generated timetable details	JSON Object	Generate, Modify, View	F1, F3, F5
Faculty_Availability	Stores available time slots for faculty members	List of time slots	Check constraints, Modify	F6
API_Endpoint	RESTful API for integrating institutional data	URL String	Fetch, Update Institutional Data	F6
Login_Credentials	Stores user authentication details	Hashed String (Encrypted)	Authentication (OAuth 2.0)	S1, S2
Request_ID	Tracks faculty modification requests	Integer (Auto-generated)	Create, Approve, Reject	F4
API_Response_Time	Measures API latency for real-time operations	Time in milliseconds	Optimize Performance	P2

*The variable names are temporary and might be changed with further use-cases.

*Student and faculty dashboards will be available in the next release.