Software Requirements Specification

For

TimeTable Generator

Version <1.0.0>

Prepared by

Group No.: 16

Alasyam Likhith	SE22UCSE021
C Shanmukha Padma Kumar	SE22UCSE061
Chakilam Kunal	SE22UCSE063
Dachepally Sathwik	SE22UCSE073
Gopishetty Sathwik	SE22UCSE103
Kurnala Karthik	SE22UCSE146

Instructor	Dr. Vijay Rao Duddu
Course	Software Engineering
Lab Instructors	Mrs. Swapna
Date	10.03.2025

Table of Contents

1. Introduction

1.1 Document Purpose

This Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for the Time Table Generator system. The purpose of this document is to provide a comprehensive overview of the system's capabilities, constraints, interfaces, and design goals to ensure that all stakeholders—including developers, testers, system administrators, and end-users—have a unified understanding of the system's behavior and expectations. It serves as a foundation for system design, development, and validation.

1.2 Product Scope

The Time Table Generator system is designed to automate the generation, management, and modification of academic timetables for educational institutions. The primary goal is to minimize human errors, reduce scheduling conflicts, and enhance administrative efficiency. The system allows faculty to request time slots, and these requests are forwarded to the admin for approval or rejection. Approved updates are reflected in the timetable, and notifications are sent accordingly. The sequence diagram provided demonstrates this workflow clearly—from request initiation to final schedule update and notifications.

Key features include:

- Automated generation of timetables based on inputs such as faculty availability, room allocation, and course requirements.
- Faculty time slot request submission and tracking.
- Admin-controlled approval/rejection of slot requests.
- Notification system for informing faculty of request status.
- Integration with institution calendars.

1.3 Intended Audience and Document Overview

The intended audience of this document includes:

- **Developers** – To understand the system's requirements and develop features accordingly.
- **Testers/QA Engineers** – To develop test cases based on functional and non-functional requirements.
- **System Administrators** – To manage deployment and server environments.
- **Faculty and Administrative Staff** – As end-users to interact with the system.
- **Project Managers** – To track progress and ensure that requirements are being met.

This document is organized into multiple sections that cover:

- Functional and non-functional requirements
- Use cases and system interactions
- External interfaces
- Design constraints
- Performance, security, and usability aspects

1.4 Definitions, Acronyms, and Abbreviations

Term	Definition
SRS	Software Requirements Specification
GUI	Graphical User Interface
Admin	Administrator responsible for managing timetable requests
Faculty	Academic staff members who request time slots and deliver lectures
Timetable System	The software system responsible for generating and managing timetables
Notification Module	Component responsible for sending updates to users
Request Time Slot	Action initiated by faculty to propose changes in schedule
Deployment	Using Vercel and Railway for deployment

1.5 Document Conventions

- All requirements are labeled using identifiers such as **FR-XX** for Functional Requirements and **NFR-XX** for Non-Functional Requirements.
- Diagrams follow UML (Unified Modeling Language) notations.
- Sequence flows and component roles are based on standard software engineering practices.
- Terms such as “shall”, “should”, “must” follow standard RFC 2119 terminology indicating requirement strength.

1.6 References and Acknowledgments

- IEEE SRS Standard (IEEE 830-1998)
- Project Team's Inputs (Sequence Diagram, Project Goals)
- Railway and Vercel Deployment
- Internal Timetable Management Practices from academic institutions
- Appreciation to faculty and admin staff who participated in requirements elicitation

2. Overall Description

2.1 Product Overview

The **Time Table Generator System** is an automated scheduling solution designed to generate, manage, and update academic timetables for educational institutions. It acts as a central system that coordinates between **faculty**, **timetable administrators**, and **notification services** to streamline the timetable management process.

This system addresses manual workload, reduces scheduling conflicts, and improves responsiveness by integrating a **request-and-approval workflow**, as captured in the system's **sequence diagram**:

- Faculty members can initiate **time slot requests**.
- The **timetable system** forwards the request to the **admin**.
- The **admin approves/rejects** the request.
- The updated schedule is communicated back and a **notification is sent**.

2.2 Product Functionality

Based on the workflow you provided (refer to the sequence diagram), the **Time Table Generator System** will offer the following functionalities:

- **Time Slot Request Submission:**
    - Faculty members initiate time slot change requests via the system interface.
- **Request Forwarding:**
    - The timetable system forwards the request to the admin automatically.
- **Request Approval/Rejection:**
    - Admin users evaluate and act upon the received requests.
- **Notification Dispatch:**
    - Upon admin decision, the system sends notifications (acceptance or rejection) to the respective faculty.
- **Timetable Update:**
    - If approved, the system updates the master timetable accordingly and reflects the changes to all stakeholders.
- **Audit and Logging:**
    - All request transactions and admin actions are logged for accountability.

- **CI/CD Deployment Workflow:**
    - Changes to the system are version controlled and deployed via GitHub pipelines to Vercel and Railway Instances.
- **System Feedback Loop:**
    - Faculty are informed about request acknowledgments and status in near real-time.

2.3 Design and Implementation Constraints

The design and deployment of the Time Table Generator system must consider the following constraints:

- **Technology Stack Constraint:**
    - The backend must be implemented using JavaScript (Node.js as per your setup).
    - Linux-based deployment environment is assumed.
- **Deployment Platform:**
    - System will be deployed and managed using **Railway and Vercel**.
- **Authentication Constraint:**
    - The system must have role-based access control to differentiate between faculty and admin users.
- **System Responsiveness:**
    - Notifications must be delivered in near real-time with minimal latency.
- **Database Selection Constraint:**
    - The system must support schedule conflict checking and fast retrieval of timetable data, influencing database schema design.

2.4 Assumptions and Dependencies

The development and deployment of the system are based on the following assumptions and dependencies:

Assumption / Dependency	Description
Faculty and Admin Internet Access	All users will have access to the system over the internet/intranet.
Role-based Authentication System	A basic authentication mechanism (login credentials) is assumed for each user role.

Notification Service	System assumes integration with an internal or external notification mechanism (e.g., email, dashboard alerts).
Server Availability	The system will run on AWS cloud infrastructure, with expected uptime dependent on EC2 instance availability.
GitHub Integration	The development and deployment cycle depends on Github repositories and runners for Railway and vercel.
Institutional Data Input	Initial course, faculty, room, and session data will be manually fed or imported during setup.
Linux Knowledge Requirement	Deployment and maintenance assume basic Linux system administration knowledge.

3. Specific Requirements

3.1 External Interface Requirements

This section identifies the external interfaces that the Time Table Generator System interacts with.

3.1.1 User Interfaces

- **Faculty Interface:**
    - GUI for submitting time slot requests.
    - View status of submitted requests.
    - Receive notifications.
- **Admin Interface:**
    - GUI for reviewing, approving, or rejecting time slot requests.
    - View full schedule and pending request list.
- **Notification Dashboard:**
    - A notification interface (either pop-up messages or email logs) displaying request status updates to users.

3.1.2 Hardware Interfaces

- No specialized hardware interface required.
- The system is intended to be accessed through standard computing devices (e.g., desktops, laptops).

3.1.3 Software Interfaces

- **Web Browser:** User interfaces are accessible via standard modern browsers (Chrome, Firefox, Edge).
- **Database Interface:** Backend uses a database system (MongoDB(NoSQL)) to store timetables, user credentials, requests, and logs.
- **Notification System Integration:** System will interact with external/internal notification APIs for sending request status messages.
- **GitHub/Railway/Vercel:** Integrated Vercel and Railway pipelines used for software versioning and deployment.

3.1.4 Communication Interfaces

- HTTP/HTTPS protocols for secure client-server communication.
- SMTP/notification API for sending request notifications.
- SSH for admin-level backend access and deployment.

3.2 Functional Requirements

The major functional requirements derived from the system workflow and sequence diagram are:

ID	Functional Requirement Description
FR1	Faculty must be able to log in and request a time slot.
FR2	Timetable system should forward the request to the Admin automatically.
FR3	Admin must receive and review the forwarded request.
FR4	Admin must be able to approve or reject the request.
FR5	The system must notify faculty of request status (approval/rejection).
FR6	Upon approval, the system must automatically update the schedule.
FR7	Faculty must be able to view updated schedule after change.
FR8	All requests and updates must be logged in the database.
FR9	Railway and Vercel must be used to automate deployment updates.
FR10	Notifications must be sent promptly after admin action.

3.3 Use Case Model

Below is the **Use Case Model** derived from your sequence diagram and workflow. Each primary actor and their interaction with the system are detailed below.

Use Case 1: Request Time Slot (Faculty)

Item	Description
Actor	Faculty
Description	Faculty submits a request to change or add a time slot.
Precondition	Faculty must be logged into the system.
Basic Flow	Faculty → Request Time Slot → Timetable System
Postcondition	Request is forwarded to Admin.

Use Case 2: Forward Request (Timetable System)

Item	Description
Actor	Timetable System
Description	System forwards faculty request to Admin.
Trigger	Request is submitted by faculty.
Outcome	Admin receives the request. Notification service is triggered.

Use Case 3: Approve/Reject Request (Admin)

Item	Description
Actor	Admin
Description	Admin reviews and takes action on the time slot request.
Precondition	Admin is logged in and has request list.
Basic Flow	Admin → Approve/Reject Request
Postcondition	Decision is sent to Timetable System and Faculty is notified.

Use Case 4: Send Notification (Notification System)

Item	Description
Actor	Notification Module
Description	Notify faculty of admin's decision.
Trigger	Admin action completed.
Outcome	Faculty is notified of approval/rejection.

Use Case 5: Update Schedule (Timetable System)

Item	Description
Actor	Timetable System
Description	Update schedule based on admin approval.
Trigger	Admin approves request.
Outcome	Schedule is updated and saved in the database.

4. Other Non-functional Requirements

4.1 Performance Requirements

The system must fulfill the following performance requirements:

- **PR1:** The time slot request operation (from faculty to timetable system) must be processed and forwarded to admin within **2 seconds** under standard load conditions.
- **PR2: Admin decision updates (Approve/Reject)** must reflect in the schedule and notify the concerned faculty within **3 seconds** of action.
- **PR3:** The system should be able to **handle concurrent requests from at least 50 faculty members** simultaneously without system lag or crash.
- **PR4: Notification dispatch** should occur within **2 seconds** of the admin's action.
- **PR5:** Schedule updates must be reflected in the faculty interface in **real-time or within 3 seconds** of admin approval.

These timings are essential to ensure smooth communication between modules and avoid bottlenecks in time-critical academic scheduling environments.

4.2 Safety and Security Requirements

The system must ensure the safety of user data and prevent unauthorized access or modification. The following security measures are applied:

- **SR1: Authentication & Authorization:**
    - Role-based access control (RBAC) must be enforced.
        - Only **faculty can submit requests**.
        - Only **admins can approve/reject requests and modify the timetable**.
- **SR2: Data Security:**
    - All communication must be encrypted using **HTTPS protocols**.
    - Database access must be secured with **encrypted credentials** and **restricted IP access**.
    - Only authenticated users can access their respective interfaces (Faculty/Admin).

- **SR3: Backup and Recovery**
- Daily automated database backups must be configured using **MongoDB Atlas Backup** features.
- In case of system failure or data corruption, the system must allow **recovery within 30 minutes** using the most recent MongoDB backup snapshot.
- Regular testing of **backup restore functionality** should be conducted to ensure data integrity.

SR4: Secure Deployment

- The **deployment pipelines on Vercel and Railway** must ensure that only tested and validated code is deployed on production.
- Only **authorized contributors** (via GitHub access controls) can trigger or approve deployments.
- Deployment configurations must be version-controlled and securely stored in **environment variable settings of Vercel and Railway** dashboards.
- Role-based access control must be implemented for managing deployment access securely.

4.3 Software Quality Attributes

The Time Table Generator System must conform to the following software quality attributes:

Quality Attribute	Description
Reliability	The system must be available at least **99% of the time**, with minimal failures during working hours.
Availability	System must be accessible 24/7, especially during timetable planning and adjustment periods.
Usability	User interfaces (for both Faculty and Admin) must be intuitive and require minimal training.
Maintainability	Modular code design and GitLab CI/CD setup ensure ease of system maintenance, updates, and bug fixes.
Scalability	System architecture must support future scaling to handle additional departments or campuses.
Portability	The system should be deployable across different Linux-based AWS instances without requiring major configuration changes.
Interoperability	Timetable system can be integrated with other academic systems (e.g., attendance tracking, course management) in the future using APIs.
Auditability	Every request, approval, update, and notification must be logged for audit and rollback purposes if required.

5. Other Requirements

5.1 System Deployment Requirements (Updated)

- The system must be deployed using **Vercel** for frontend hosting and **Railway** for backend server deployment. The database is managed via **MongoDB Atlas**. Deployment workflows are managed through **GitHub-based integration**, ensuring that only tested and approved code from designated branches is deployed. Environment variables and configurations are securely managed through the **Vercel and Railway dashboards**. Monitoring and logging features provided by Railway should be used to track system health and performance.

- **5.2 Logging and Monitoring:**
    - All significant system activities (e.g., request submission, approval/rejection, notification dispatch) must be logged with timestamps and user roles.
    - Logs must be stored in a secured location (e.g., S3 bucket) with **daily log rotation** and **auto-archive** enabled.
- **5.3 Compatibility Requirements:**

    - The system must work across modern browsers: **Chrome, Firefox, Edge, Safari**.
    - Compatible with **desktop and mobile views** (responsive design).
- **5.4 Notification Requirements:**
    - System must send **real-time alerts or email notifications** when a request is forwarded or a decision is made.
    - Notification service should be extendable to SMS or mobile app push notifications in future versions.

Appendix A – Data Dictionary

Field Name	Description	Data Type	Source
Faculty_ID	Unique identifier for each faculty member	Integer/String	Faculty DB
Request_ID	Unique ID for time slot request	Integer	Generated by system
Time_Slot	Requested time slot (e.g., Mon 10:00–11:00)	String	Faculty input
Request_Status	Status: Pending / Approved / Rejected	Enum/String	Admin input
Admin_ID	Identifier for admin handling the request	Integer	Admin DB
Notification_ID	Unique ID for system notifications	Integer	System-generated
Notification_Type	Type: Request Sent / Approved / Rejected	Enum/String	System-generated
Timestamp	Date and time of action or notification	DateTime	System log
Schedule_Update_Log	Record of modifications made to the timetable	Text/Log	Timetable System

---

Appendix B – Group Log

Date	Member Name	Task Completed	Status
01-Feb-2025	Shanmukha Padma Kumar Challa, Likhith, Kunal	Initial project setup and role allocation	Completed
03-Feb-2025	Likhith, D.Sathwik, Karthik	Functional Requirement Listing and Sequence Diagram	Completed
07-Feb-2025	Shanmukha Padma Kumar Challa, G. Sathvik	SRS Template Structuring and Draft Content	Completed
16-Feb-2025	Entire Team	System Design Review and Role Testing	Completed
24-Feb-2025	Entire Team	Finalization of Use Cases and Workflow Diagram	Completed
02-Mar-2025	Shanmukha Padma Kumar Challa	Deployment Pipeline Configuration & Testing	Completed
04-Mar-2025	Entire Team	Documentation and Final SRS Preparation	Completed