Software Design Specifications

for

Web Utility for Time Table Generation
Version 1.0

Prepared by
Group: 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

Document Information

Title:	Web Utility for Time Table Generation
Document Version No: 1.0	
	Document Version Date: 07-April-2025
Prepared By: Team-6	**Preparation Date: 07-April-2025**

Version History

Ver. No.	Ver. Date	Revised By	Description	Filename
1				

Table of Contents

1 Introduction

This Software Design Specification (SDS) document provides a detailed architectural and component-level design for the **Web Utility for Time Table Generation** project. It serves as a bridge between the Software Requirements Specification (SRS) and the actual development process, outlining the structural blueprint and design rationale of the system. The document ensures that all stakeholders have a clear understanding of how the system will be constructed, ensuring consistency, maintainability, and scalability throughout the software lifecycle.

1.1 Purpose

The purpose of this document is to define and communicate the software design of the *Web Utility for Time Table Generation*. It describes the architectural components, data structures, interface designs, algorithms, and their interactions. This document serves as a guide for the development team during implementation and as a reference for testers and maintainers.

The intended audiences include:

- **Developers** – for implementing the modules based on the described architecture.
- **Testers** – for understanding system structure and inter-module dependencies.
- **Project Managers** – to ensure alignment between design and project goals.
- **Institutional Stakeholders** – to review the technical implementation of features described in the SRS.

1.2 Scope

This Software Design Specification applies to the *Web Utility for Time Table Generation* project. It covers the design aspects of all system components — including the front-end (UI/UX), back-end logic (scheduling algorithm), database schema, API design, and authentication mechanisms. The document also identifies design constraints, component responsibilities, and key design decisions that guide implementation.

This document influences:

- Module-level software development
- System architecture
- Integration planning
- Performance optimization
- Security compliance

1.3 Definitions, Acronyms, and Abbreviations

Term	Description
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
DBMS	Database Management System
OAuth 2.0	Open Authorization 2.0 (authentication protocol)
RBAC	Role-Based Access Control
SDS	Software Design Specification
SRS	Software Requirements Specification
UI/UX	User Interface / User Experience

SSR	Server-Side Rendering
JWT	JSON Web Token (for session management)
MongoDB	NoSQL database used for data storage

1.4 References

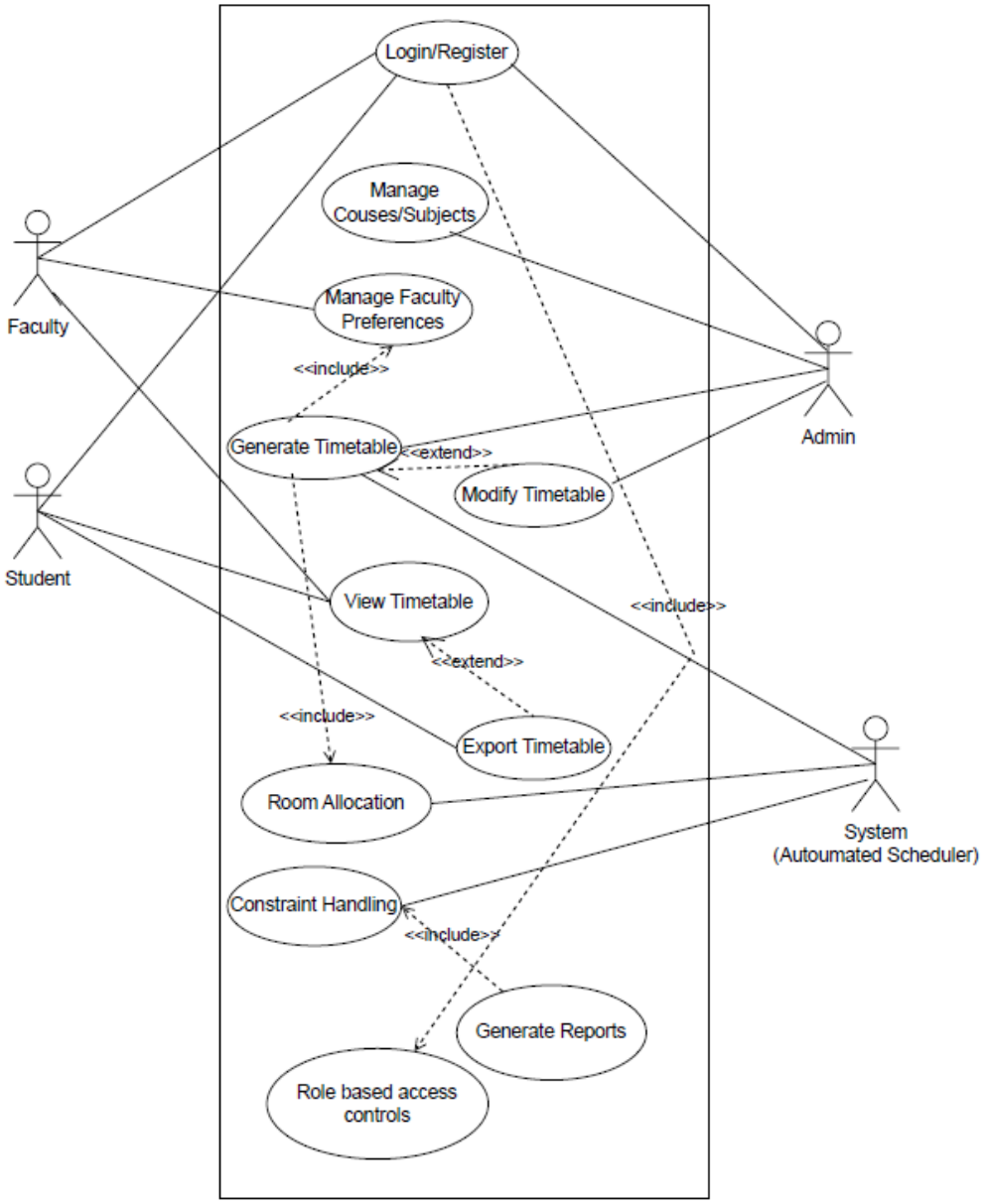
- [Software Requirements Specification – Web Utility for Time Table Generation (Version 1.0)]
- [Statement of Work (SOW) – Web Utility for Time Table Generation]
- IEEE Std 1016-2009 – IEEE Standard for Information Technology – System Design Description
- ReactJS Documentation: https://reactjs.org
- Node.js Documentation: https://nodejs.org/
- MongoDB Documentation: https://www.mongodb.com/docs

2 Use Case View

This section outlines the key use cases that form the backbone of the *Web Utility for Time Table Generation*. These use cases are central to the system's functionality and demonstrate the interactions between users and the system components. The use case diagram and individual descriptions provide insights into the behavioral model of the system and support the identification of major system responsibilities and interactions.

Use Case Diagram

Here's the **Use Case Diagram** that shows interactions between actors (Admin, Faculty, Student)* and the system:



2.1 Use Case

Login/Register

Actors: Admin, Faculty, Student *
Description: Authenticates users and provides access based on roles.
Usage: Essential entry point for any user to interact with the system.

Manage Courses/Subjects

Actors: Admin
Description: Admin can add, edit, or remove courses and subjects offered in the semester.

Manage Faculty Preferences

Actors: Faculty, Admin *
Description: Faculty submits preferred times/slots; Admin can view/override as needed.
Includes: Constraint Handling

Generate Timetable

Actors: Admin
Description: The system creates an optimized timetable based on all data and constraints.
Includes: Room Allocation, Constraint Handling
Extends: Modify Timetable

Modify Timetable

Actors: Admin
Description: Enables manual modifications to the generated timetable post-review.

View Timetable

Actors: Student, Faculty *
Description: Allows users to view their personal or department-wise schedules.
Extends: Export Timetable

Export Timetable

Actors: Student, Faculty *
Description: Enables export of timetables in downloadable formats such as PDF or Excel.

Room Allocation

Actors: System
Description: Automatically allocates classrooms based on room size, availability, and subject type.
Includes: Constraint Handling

Constraint Handling

Actors: System
Description: Manages all scheduling constraints (faculty availability, room conflicts, etc.)

Generate Reports

Actors: Admin
Description: Admin generates analytical reports such as resource usage, faculty load, etc.

Role-Based Access Controls

Actors: System
Description: Ensures that each user accesses only their permitted features/modules.

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

3 Design Overview

The Web Utility for Time Table Generation is structured as a modular, service-oriented web application that separates concerns across the front-end, back-end, and data persistence layers. The system design follows the MVC (Model-View-Controller) pattern and leverages RESTful APIs to ensure interoperability and scalability.
The architecture supports core functionalities such as automated timetable generation, role-based access control, and real-time updates through well-defined interfaces. The design prioritizes modularity, maintainability, and extensibility to support future integration with broader college management systems.

3.1 Design Goals and Constraints

1. Usability – The system should offer a clean and intuitive web interface for all user roles.

2. Automation – The timetable generation must be automatic and free from clashes or conflicts.

3. Security – Role-based access and secure authentication must protect all user data.

4. Scalability – The system should be modular and adaptable for future department-wide use.

1. Web-Only – The application will be accessible only through a web browser, no mobile or desktop versions.

2. Time – The system must be developed and deployed within the academic semester's time frame.

3. Open-Source – Only free, open-source tools and libraries may be used due to budget limits.

4. Hosting – The system must comply with university IT infrastructure and hosting requirements.

3.2 Design Assumptions

Valid Data Input – It is assumed that the admin will enter all course, faculty, and room data accurately and completely before timetable generation.
Static Role Structure – User roles (Admin, Faculty, Student)* are predefined and do not change dynamically within a session.
Stable Network – The system assumes a stable internet connection for both client and server operations during use.

3.3 Significant Design Packages

Auth Module: Handles user login, role-based access control, and JWT authentication.
User Module: Manages Admin, Faculty, and Student profiles, availability, and preferences.
Scheduling Module: Core engine for auto-generating conflict-free timetables based on input constraints.
Data Management Module: CRUD operations for courses, classrooms, departments, and academic sessions.
Timetable Module: Stores, displays, and exports generated timetables in PDF/Excel formats.
Notification Module: Sends alerts for timetable updates or admin approvals (optional/future scope).
API & Database Module: REST API layer and MongoDB interface for structured data access and persistence.

3.4 Dependent External Interfaces

The table below lists the public interfaces this design requires from other modules or applications.

External Module/Application	Using the Functionality/Interface	Description
MongoDB Atlas	Backend Database	Used for storing user details, courses, rooms, timetable data, and schedule requests
React.js Frontend	User Interface	Provides responsive UI screens for Admin, Faculty, and Students
Node.js + Express.js API	Business Logic and API	Handles data processing, CRUD operations, timetable generation logic, and request handling

3.5 Implemented Application External Interfaces (and SOA web services)

The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	Module Implementing the Interface	Functionality/Description
Auth API	User Management Module	Provides API endpoints for user registration, login, logout, and JWT-based token generation
Timetable API	Timetable Management Module	Provides API endpoints for timetable generation, fetching timetable data, and room allocation
Request API	Schedule Request Module	Provides API endpoints for faculty to submit schedule change requests and admin to approve/reject them

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

4 Logical View

This section outlines the detailed design of the system, illustrating how the various modules and classes interact to implement core functionality. The logical architecture is layered to represent increasing levels of detail, beginning with system-level interaction and descending into module-specific internal behaviour.

4.1 Design Model

The system adopts a layered MVC (Model-View-Controller) architecture with clearly separated concerns for user interaction, business logic, and data management. Each major component corresponds to a distinct role in handling user requests and processing scheduling operations.

- Layered Design Overview:

→ Presentation Layer (UI Layer):

Built with React.js
Contains dashboards for Admin, Faculty, and Student users*

→ Controller Layer (API Gateway):

Validates incoming requests
Delegates the processing logic to appropriate service modules based on user role and endpoint
Handles authentication checks, parameter validation, and response formatting

→ Service Layer (Business Logic):

Implements core functionalities such as timetable generation, faculty request handling, and constraint resolution
Includes Admin Services, Faculty Services, Student Services*, and the Scheduling Engine
Interacts with both the controller layer and the data layer

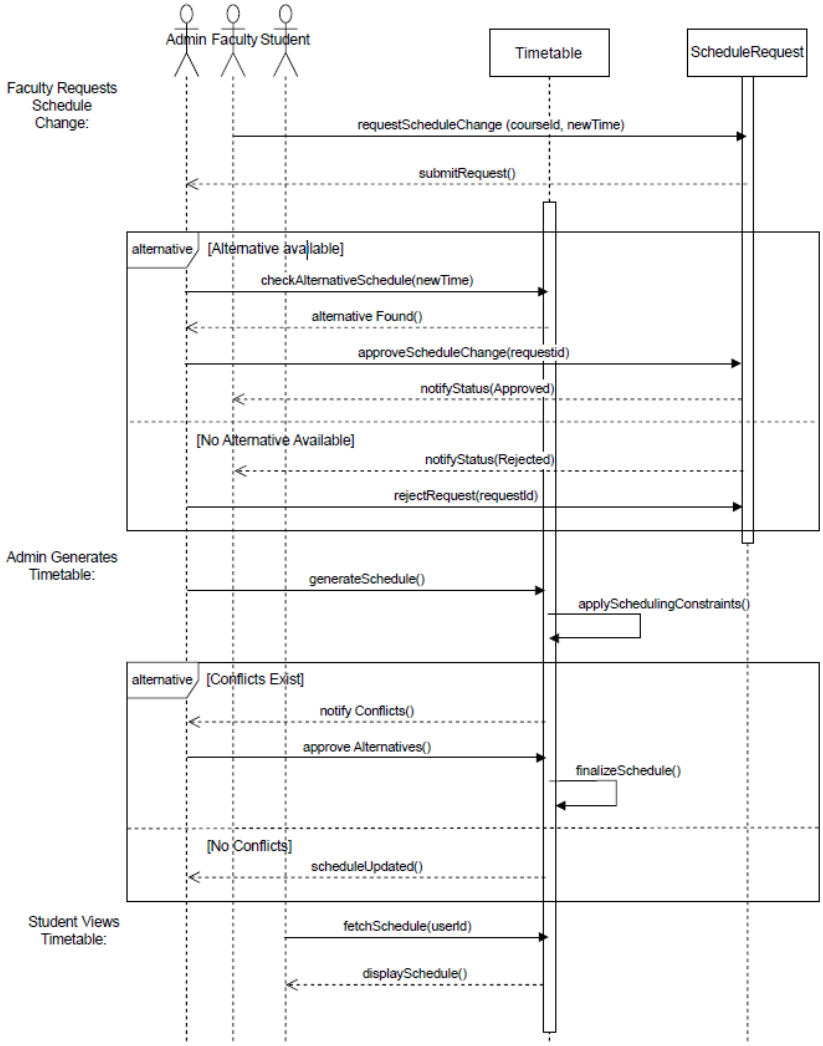
→ Data Access Layer:

Uses MongoDB for data persistence
Maps domain models to documents in collections (e.g., Faculty, Courses, Rooms,
Ensures data consistency and optimized query performance

- Major Classes (Examples):

TimetableGenerator: Applies constraint-solving algorithms to generate optimized schedules
AdminManager: Manages faculty/course/classroom CRUD and overrides
FacultyManager: Handles viewing and requesting changes to faculty schedules
StudentTimetableViewer: Retrieves and formats student-specific schedules
AuthService: Interfaces with OAuth 2.0 or admin created DB credentials to match-u

4.2 Use Case Realization



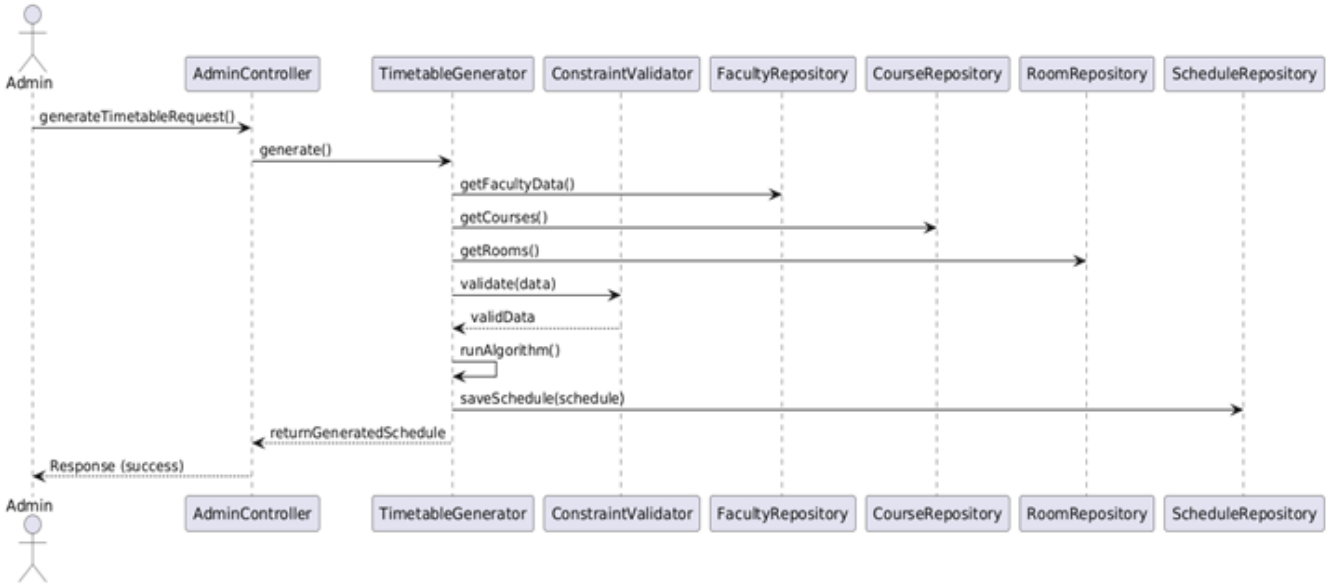
Use Case: Generate Timetable

Actor: Admin

Description: This use case allows the Admin to initiate timetable generation based on current faculty availability, classroom resources, and course data. The scheduling engine processes constraints and generates a conflict-free schedule, which is then stored and made available for viewing.

High Level Sequence Diagram (Module Interaction):

Low Level Sequence Diagram (Class Collaboration):



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

5 Data View

This section describes the system's persistent data design, including how domain objects are modeled and stored in the MongoDB database. Since the timetable generation system relies heavily on structured data—such as faculty availability, course metadata, and classroom constraints—data persistence is essential to ensure accurate scheduling, access control, and traceability.

The application uses MongoDB to manage structured but flexible documents. Each major entity is stored in its own collection and is interlinked using object references or foreign keys (e.g., faculty IDs within courses). These domain entities support the operations of scheduling, viewing, and updating data across Admin, Faculty, and Student modules.

5.1 Domain Model

The domain model consists of several core entities that represent real-world components of the academic scheduling system. These entities are persisted as JSON-like documents in MongoDB collections.

Entity	Attributes	Relationships
Faculty	facultyID, name, email, availability[], department, assignedCourses[]	One-to-Many with Courses, referenced in Timetable
Course	courseID, courseName, facultyID, semester, hoursPerWeek, classroomPreference, department	References Faculty, Timetabled into Rooms
Room	roomID, capacity, type, availableSlots[]	Assigned to Timetable entries
Schedule	timetableID, entries[], generatedBy, generatedAt, status	Includes embedded schedule entries for display
ScheduleEntry	courseID, facultyID, roomID, day, slot, duration	Embedded within Schedule document

User	userID, role (Admin/Faculty/Student), email, passwordHash, name	Used in authentication and access control
ChangeRequest	requestID, facultyID, courseID, requestedChange, status, adminResponse, timestamp	One-to-One with Faculty and Admin

5.2 Data Model (persistent data view)

The Web Utility for Time Table Generation uses a document-based schema, implemented in MongoDB, to store persistent data. Each major entity in the domain model corresponds to a MongoDB collection. Documents within each collection store attributes as key-value pairs, allowing for flexible and scalable data representation.

The system uses embedded documents for tightly coupled data (e.g., ScheduleEntry within Schedule) and referenced documents for loosely coupled data (e.g., Faculty in Course). Indexes are created on commonly queried fields like facultyID, courseID, and timetableID to improve performance.*

5.2.1 Data Dictionary

Field Name	Entity	Data Type	Description
facultyID	Faculty, Course, ScheduleEntry	Integer / String	Unique identifier for each faculty member
name	Faculty, User	String	Full name of the user or faculty
email	User, Faculty	String	Email address used for login and communication
availability	Faculty	Array of TimeSlots	Available time slots for scheduling
assignedCourses	Faculty	Array of courseIDs	Courses taught by the faculty
courseID	Course, ScheduleEntry	Integer / String	Unique identifier for each course
courseName	Course	String	Name of the course
semester	Course	Integer	Semester in which the course is offered
hoursPerWeek	Course	Integer	Number of hours to be scheduled per week
roomID	Room, ScheduleEntry	Integer / String	Unique identifier for each room
capacity	Room	Integer	Number of students the room can accommodate
availableSlots	Room	Array of TimeSlots	Time slots when the room is available
timetableID	Schedule	Integer / String	Unique identifier for a generated timetable
entries	Schedule	Array of ScheduleEntry	Contains all scheduled classes

generatedBy	Schedule	String (userID)	Admin who generated the timetable
generatedAt	Schedule	DateTime	Timestamp when the timetable was created
status	Schedule, ChangeRequest	String (e.g., Active, Pending)	Indicates state of timetable or request
day	ScheduleEntry	String	Day of the week (e.g., Monday, Tuesday)
slot	ScheduleEntry	Integer	Time slot index (e.g., 1st period, 2nd period)
duration	ScheduleEntry	Integer	Duration of the class in number of slots
userID	User	String	Unique identifier for each system user
role	User	Enum (Admin, Faculty, Student)	Defines the user role for access control
passwordHash	User	String (Hashed)	Encrypted password hash stored securely
requestID	ChangeRequest	String / Auto-ID	Unique identifier for a faculty request
requestedChange	ChangeRequest	String	Details of what the faculty wants changed
adminResponse	ChangeRequest	String	Admin's approval or rejection and explanation
timestamp	ChangeRequest	DateTime	Time when the request was submitted

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

6 Exception Handling

This section describes exceptions that are defined within the Web Utility for Timetable Generation System, the circumstances in which they can occur, how the exceptions are logged, and the expected follow-up actions.

Exception Type	Circumstances	Handling Mechanism	Logging and Follow-up Action
Invalid Login Credentials Exception	When a user provides incorrect email or password during login	Display appropriate error message - "Invalid Email or Password"	Error logged in server logs; User prompted to re-enter correct credentials
Database Connection	When the backend fails to establish a connection with MongoDB Atlas	Retry connection up to a defined maximum number of attempts	Error logged in server logs; Admin informed if issue persists

Failure Exception			
Timetable Conflict Exception	When timetable generation encounters overlapping faculty or room allocation	Show error message indicating conflict; Timetable generation halted	Conflict details logged; Admin to resolve manually or modify input data
Unauthorized Access Exception	When a user tries to access resources not allowed for their role (Student accessing Admin routes)	Redirect to Login Page or Access Denied Page	Unauthorized attempt logged with user details for security monitoring
Schedule Request Expiry Exception	When a schedule request is pending beyond the configured expiry time	Automatically mark the request as 'Expired'	Event logged; Notification sent to Faculty about expiry
Room Not Available Exception	When the system fails to allocate a room due to unavailability during timetable generation	Prompt error to Admin for manual room allocation	Logged in server; Admin to allocate room manually
API Request Timeout Exception	When API call exceeds maximum response time due to heavy load	Show error message - "Request Timed Out. Please try again."	Request details logged for performance monitoring and optimization

7 Configurable Parameters

This section describes the configuration parameters used in the Web Utility for Time Table Generation System. These parameters can be easily configured in the .env file or configuration files of the backend application. Some parameters are dynamic and can be changed without restarting the application, while others require a server restart.

This table describes the simple configurable parameters (name / value pairs).

Configuration Parameter Name	Definition and Usage of Parameter	Dynamic?
MONGODB_URI	MongoDB database connection string used to connect the application to the database.	No
PORT	The backend server port number on which the Node.js Express server runs.	No
FRONTEND_API_LINK	The URL link of the frontend React.js application used for API calls from backend to frontend.	No
JWT_SECRET	Secret key used for generating and validating JWT tokens for secure authentication.	No

MAX_RETRY_ATTEMPTS	Maximum number of retry attempts for database connection failure before terminating the server.	Yes
MAX_COURSES_PER_TIMETABLE	Maximum number of courses allowed in a single timetable generation process.	Yes
MAX_ROOM_CAPACITY	Maximum room capacity for allocation during timetable generation. Helps in avoiding overcrowding.	Yes
REQUEST_EXPIRY_TIME	Time duration (in hours) after which a schedule request made by a faculty will automatically expire if not approved by admin.	Yes

8 Quality of Service

This section describes the aspects of the Web Utility for Time Table Generation System related to availability, security, performance, and monitoring in the production environment.

8.1 Availability

The Web Utility for Time Table Generation System is designed to provide high availability for users (Admin, Faculty, and Students)* throughout the academic year. The system will be accessible 24x7 via a web browser.
Design considerations to support availability:
Hosting the application backend on a reliable server with MongoDB Atlas as the database ensures high availability and minimal downtime.
Maintenance tasks like database backups, housekeeping, and cleaning of unused schedule data will be scheduled during off-peak hours to minimize impact on users.
Downtime during timetable generation will be negligible since the generation logic is optimized to execute within seconds for normal workloads.

- Potential availability impacting factors:

Major data loading operations (e.g., adding bulk course or student data) may temporarily slow down the system.
Housekeeping scripts (deleting old requests, cleaning outdated timetable entries) may require short maintenance windows but will be scheduled during non-operational hours.

8.2 Security and Authorization

The system will implement role-based access control to secure both features and sensitive timetable data.
Security features:
JWT (JSON Web Token) based authentication for secure user login sessions.
Admin role will have full access to timetable generation, room management, and handling schedule requests.
Faculty role will have access to view their schedule and request changes.*
Student role will have access to view their respective timetable only.*

- Authorization Design:

User data, passwords, and sensitive information will be securely stored in the database using password hashing techniques.
Unauthorized users attempting to access restricted features will be redirected to the login page with appropriate error messages.
User management and role assignment will be handled by Admin through secure dashboard controls.

8.3 Load and Performance Implications

- Load Projections:

The system is expected to handle data for around 500+ students, 50+ faculty members, and multiple courses in an academic semester.

- Performance Considerations:

Timetable generation logic is optimized to execute within 1-3 seconds for a typical dataset (50 courses).
RESTful API endpoints are designed for efficient CRUD operations with optimized MongoDB queries.
Database table growth (collections like users, courses, rooms, timetable, schedule_requests) will scale based on academic batches and will be monitored regularly.

- Expected Business Transaction Execution Rates:

User Login: Expected response time < 1 second.
Timetable View API: Expected response time < 1 second.
Timetable Generation: Expected execution time 1-3 seconds depending on data size.

8.4 Monitoring and Control

The application will implement logging and monitoring mechanisms to support smooth operations in production.

- Monitoring Features:

Server logs will capture errors, failed API requests, and unexpected exceptions for debugging and maintenance.

MongoDB Atlas built-in monitoring tools will be used to track database performance, query execution time, and storage utilization.

- Controllable Processes:

Automated email or system notifications for Admin when schedule requests are submitted by Faculty.
Monitoring API response times, user activity logs, and schedule generation logs.
Admin dashboard will provide visibility into active users, pending schedule requests, and overall system status.

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