Software Requirements Specification (SRS)

Timetable Management System

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This Software Requirements Specification (SRS) defines all functional and non-functional requirements for the Timetable Management System—a full-stack web application built with React.js (frontend), Node.js/Express (backend), MongoDB (database), and optional AI microservices.

1.2 Document Conventions

- **Bold text** denotes major features or sections
- `Code-like formatting` denotes API paths or attribute names

1.3 Intended Audience

- System developers & engineers
- Project managers
- QA testers

- Educational institution planners

1.4 Scope

The system automates academic timetable creation, detects conflicts, manages schedules, and supports AI-based optimization.

1.5 Definitions

- **AI Optimization:** Algorithmic scheduling improvements
- **RBAC:** Role-Based Access Control

2. PROJECT OVERVIEW

2.1 Background

Manual timetable creation is error-prone and inefficient, especially for large educational institutions.

2.2 System Objectives

- Automate timetable creation
- Detect conflicts immediately
- Provide real-time updates
- Optimize scheduling using AI
- Support thousands of users reliably

2.3 Product Features

- Automated timetable generation
- Manual editing and session dragging
- Conflict detection
- Timetable publication
- Notifications
- Multi-role access control

3. FUNCTIONAL REQUIREMENTS

3.1 User Management

- Registration, login, logout
- Password recovery
- RBAC (Admin, Lecturer, Student)
- Profile management

3.2 Timetable Generation

Admin/Lecturer inputs:

- Department
- Semester
- Courses
- Lecturer availability
- Room constraints

System outputs:

- Draft timetable
- AI-assisted scheduling suggestions (optional)

3.3 Timetable Management

- Create, edit, delete timetables
- Modify individual sessions
- Drag-and-drop scheduling
- Validate timetable changes

3.4 Conflict Detection

Detects:

- Lecturer double-booking
- Room double-booking

- Time overlaps
- Capacity issues

3.5 Notification System

Notifies users about:

- Conflicts
- Updates
- Published timetables

3.6 Reporting / Exporting

Export formats:

- PDF
- Excel
- Printable HTML

3.7 Role-Based Access Control

Admin:

- Manage full system

  Lecturer:
- View & update availability

  Student:
- Read-only timetable access

4. NON-FUNCTIONAL REQUIREMENTS

4.1 Performance

- Responses ≤ 2 seconds
- 500+ concurrent users
- DB queries ≤ 150 ms
- Timetable generation ≤ 10 seconds

4.2 Security

- Bcrypt password hashing
- JWT authentication
- Sanitization against XSS, injection
- HTTPS enforced
- Encrypted data at rest

4.3 Usability

- Clean, intuitive UI
- Mobile responsive
- Color-coded timetable
- Helpful error messages

4.4 Scalability

- Microservice-ready
- Kubernetes for horizontal scaling

4.5 Reliability & Availability

- 99% uptime
- Auto backups every 24 hours

4.6 Maintainability

- Modular components
- Service-layer architecture

4.7 Portability

- Works with all modern browsers

5. USER INTERFACE REQUIREMENTS

5.1 Landing Page

UI Elements

- Title, branding
- System description
- Buttons: Login, Register
- Footer

Functionality

- No authentication needed
- Routes to login/register
- Preloads major assets

5.2 Login Page

UI Elements

- Email, password fields
- Remember me checkbox
- Forgot Password link

Functionality

- Validates credentials
- Redirects based on role
- Stores secure JWT

5.3 Dashboard

Admin Dashboard

- Statistics
- Quick links
- Activity logs

Lecturer Dashboard

- Schedule
- Availability editing
- Conflict alerts

Student Dashboard

- Timetable display
- Course list
- Notifications

5.4 Timetable Generation Interface

- Dropdowns for department, semester
- Course selection
- Lecturer availability
- Room constraints
- AI suggestion toggle
- Generate button

Function Flow

1. User enters requirements
2. System validates
3. AI engine suggests optimal layout
4. Conflict engine checks mistakes
5. Draft is created

5.5 Timetable Management View

UI Elements

- Grid layout
- Drag-and-drop classes
- Color-coded blocks
- Side info panel

Functionality

- Real-time conflict checking
- Manual edits
- Version control
- Publish button

5.6 Conflict Warning Interface

Displays:

- Overlapping sessions
- Room/lecturer conflicts
- Missing constraints

Provides:

- Fix suggestions
- Auto-resolve option

5.7 User Profile Page

- Update password
- Edit personal information
- Notification settings
- Availability management (lecturers only)

5.8 System Workflow & Navigation Diagram

```
Landing Page
     ↓
Login Page → Forgot Password
     ↓
Dashboard
     ↓
Admin: Management Pages
Lecturer: Availability/Schedule
Student: View timetable
     ↓
Timetable Generation
     ↓
Draft Timetable → Conflict Detection
     ↓
Management View (Edit & Publish)
     ↓
Notifications Sent
```

6. SYSTEM ARCHITECTURE

6.1 High-Level Architecture

Frontend (React) ↔ Backend API (Express) ↔ Database (MongoDB)
Optional AI Microservice (Python/FastAPI)

6.2 Frontend Architecture

- React functional components
- Context API / Redux
- Axios for HTTP requests

6.3 Backend Architecture

- Express routers
- Controller → Service → Repository

- JWT middleware

6.4 AI Microservice

- FastAPI (Python)
- Uses OR-Tools for optimization

6.5 Deployment Architecture

- Docker containers
- Nginx reverse proxy
- GitHub Actions CI/CD

7. TECHNOLOGY STACK

7.1 Frontend

React.js, Tailwind CSS, Axios

7.2 Backend

Node.js, Express.js

7.3 Database

MongoDB, Mongoose ORM

7.4 AI Components

Python, Scikit-learn, OR-Tools

7.5 Infrastructure

Docker, Kubernetes, Nginx

8. AI FEATURES

8.1 AI-Based Timetable Optimization

Uses:

- Lecturer availability
- Room type
- Time constraints

8.2 Pattern Learning

Learns:

- Room usage
- Lecturer load

8.3 Load Balancing

Prevents lecturer overloading

8.4 Smart Room Assignment

Matches rooms by:

- Size
- Type
- Proximity

9. DATABASE DESIGN

9.1 ER Diagram (Text-Based)

```
Users ---< Timetables >--- TimetableEntries
Courses ---<
Rooms ---<
```

```
Lecturers ---<
```

9.2 Collections & Attributes

Tables:

- Users
- Courses
- Rooms
- Timetables
- TimetableEntries

(Attributes already defined)

10. API ENDPOINTS

Authentication

```
POST /api/auth/login
 POST /api/auth/register
```

Timetables

```
POST /api/timetables
 GET /api/timetables
 PUT /api/timetables/:id
 DELETE /api/timetables/:id
```

Conflicts

```
POST /api/conflicts
```

11. SECURITY CONSIDERATIONS

11.1 Authentication

- JWT access/refresh tokens

11.2 Authorization

- RBAC middleware

11.3 Data Validation

- Joi validation schema

11.4 Encryption

- Bcrypt
- HTTPS

11.5 Common Attack Protection

- Rate limiting
- CORS
- CSRF tokens
- Helmet.js headers

12. TESTING AND VALIDATION

12.1 Unit Testing

- Jest
- React Testing Library

12.2 Integration Testing

- Supertest

12.3 Performance Testing

- JMeter
- Locust

12.4 User Acceptance Testing

12.5 Security Testing

- Vulnerability scanning
- Penetration testing