**SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY**



SE3040 – Application Frameworks

2024 – Assignment01

**University Timetable Management System**

**BSc (Hons) in Information Technology Specialized in Software Engineering**

Contents

[Student Details 3](#_Toc162200447)

[Introduction 4](#_Toc162200448)

[Scope and Requirements 5](#_Toc162200449)

[Functional Requirements 5](#_Toc162200450)

[Non-Functional Requirements 5](#_Toc162200451)

[Initial Directory Structure 6](#_Toc162200452)

[Technologies 8](#_Toc162200453)

[Backend Development 8](#_Toc162200454)

[Security 8](#_Toc162200455)

[Database Management 8](#_Toc162200456)

[Testing Requirements 9](#_Toc162200457)

[Unit Testing 9](#_Toc162200458)

[Security Testing 10](#_Toc162200459)

[Performance Testing 10](#_Toc162200460)

# Student Details

Student No: IT21203176

Student Name: Dias A.H.J.S.S.

Student Email: it21203176@my.sliit.lk

# Introduction

The University Timetable Management System optimizes scheduling in universities, prioritizing efficiency, security, and user experience. It provides tailored functionalities for students, faculty, and administrative staff, ensuring smooth operation and satisfaction.

In response to the dynamic nature of university schedules, the system prioritizes flexibility and ease of use, allowing authorized users to create, modify, and query class timetables seamlessly. By leveraging RESTful API architecture, built on Express JS (Node.js), it ensures smooth communication between various components while adhering to industry-standard design principles.

The system prioritizes security with robust authentication and authorization mechanisms, utilizing JWT for secure login and session management. Administrators can easily manage courses, faculty assignments, and enrollments. Timetable management tools enable seamless scheduling adjustments as necessary. Moreover, the system provides tools for room and resource booking, ensuring optimal utilization of facilities while preventing conflicts or overlaps. Students benefit from the ability to enroll in courses and access their personalized timetables, while faculty members and administrators gain insights into enrollment data for effective academic planning and management. To enhance user engagement and communication, the system incorporates notification and alert systems, delivering timely updates on timetable changes, room allocations, and important announcements.

The system emphasizes compliance with non-functional requirements, particularly focusing on security, database design, code quality, and documentation. By adhering to best practices and leveraging modern technologies, it aims to revolutionize scheduling processes in educational institutions, promoting efficiency, transparency, and collaboration among stakeholders.

# Scope and Requirements

The scope of the project encompasses the development of a secure RESTful API for managing the timetable system of a university. The system will facilitate the creation, modification, and querying of class schedules for students, faculty, and administrative staff, ensuring secure access, data integrity, and user-friendly interfaces. Below are the requirements gathered from stakeholders:

## Functional Requirements

1. User Roles and Authentication
2. Course Management
3. Timetable Management
4. Room and Resource Booking
5. Student Enrollment
6. Notifications and Alerts

## Non-Functional Requirements

1. Security
2. Database Design
3. Code Quality and Documentation
4. Error Handling and Logging

These requirements will serve as the foundation for the development and implementation of the University Timetable Management System, ensuring it meets the needs and expectations of all stakeholders involved.

# Initial Directory Structure

ASSIGNMENT-01-IT21203176/backend

│

├── controllers/

│ ├── authController.js

│ ├── courseController.js

│ ├── timetableController.js

│ ├── locationResourceController.js

│ ├── enrollmentController.js

│ └── notificationController.js

│

├── middleware/

│ └── authMiddleware.js

│

├── models/

│ ├── Course.js

│ ├── Enrollment.js

│ ├── LocationResources.js

│ ├── Notification.js

│ ├── TimetableSession.js

│ └── User.js

│

├── routes/

│ ├── authRoute.js

│ ├── courseRoute.js

│ ├── enrollmentRoute.js

│ ├── locationResourceRoute.js

│ └── timetableRoute.js

│

│

├── tests/

│ ├── auth.test.js

│ ├── course.test.js

│ ├── timetable.test.js

│ ├── locRes.test.js

│ └── enroll.test.js

│

├── .env

│

├── server.js

└── package.json

controllers/: Holds controller files responsible for handling business logic for different modules.

middleware/: Contains middleware functions, particularly for JWT authentication.

models/: Includes files defining Mongoose schemas.

routes/: Defines route handlers for different endpoints related to authentication, course management, timetable management, enrollments and bookings.

tests/: Contains test files that are used for unit testing of the application

server.js: Main entry point for the application where server setup and middleware configurations are defined.

.env: Store environment variables that are required for configuring the application.

# Technologies

For the development of the University Timetable Management System, a combination of technologies and frameworks was employed to ensure robustness, scalability, and security. The overview of the technologies utilized in various components of the project is as follows:

## Backend Development

**Express.js (Node.js):** Employed as a minimalist web framework for Node.js, Express.js provided a robust set of features for developing the applications, chosen for its simplicity, flexibility, and performance in building RESTful APIs.

## Security

**Role-Based Access Control (RBAC):** Implemented to define multiple user roles (Admin, Faculty, Student) with varying access levels, RBAC ensured that each role had access to specific endpoints and functionalities based on their authorization level.

**JSON Web Tokens (JWT):** Utilized for secure authentication and session management, JWTs facilitated a compact and self-contained way to transmit information between parties in the form of a JSON object, ensuring cryptographic signing to verify authenticity.

## Database Management

**MongoDB:** Utilized as a NoSQL document-oriented database, MongoDB was selected for its scalability, flexibility, and seamless integration with Node.js applications, enabling efficient storage and management of data related to timetables, courses, users, and bookings.

**Development Tools**

**Postman:** Utilized as a popular collaboration platform for API development, Postman simplified the process of designing, testing, and documenting APIs, facilitating efficient testing of API endpoints and generation of API documentation.

**Visual Studio Code:** Visual Studio Code provided essential features for Node.js development, including syntax highlighting, debugging, and version control integration.

**Git:** Utilized as a distributed version control system, Git enabled tracking of changes in the source code during development, version control, and code management.

By integrating these technologies and adhering to industry best practices, we developed a secure, scalable, and user-friendly RESTful API for the University Timetable Management System, ensuring code quality, maintainability, and compliance with all requirements.

# Testing Requirements

## Unit Testing

Unit testing for all controllers using the Jest testing library, particularly focusing on Express.js components. Each controller's functionality has been thoroughly tested to ensure its behavior in isolation. The testing approach involves creating individual test suites for each controller, encompassing various test cases to cover different scenarios. The Jest framework provides the necessary tools for mocking dependencies, setting up test data, and making assertions about the expected behavior of the controllers.

**user.test.js**: This file contains unit tests for the functionalities provided by the **userController**. The tests cover scenarios such as user creation, retrieval, updating, and deletion. Additionally, it might include tests for authentication and authorization mechanisms implemented within the user controller.

**course.test.js**: Within this file, unit tests for the **courseController** are implemented. The tests focus on functionalities related to managing courses, such as creation, retrieval, updating, and deletion of courses. It might also include tests for course enrollment, validation, and any other business logic associated with courses.

In the **enroll.test.js** file, tests are conducted for the **enrollmentController** module. This includes verifying the enrollment of students in courses (**enrollStudent** function) and retrieving enrollments for a specific student (**getStudentEnrollments** function).

Similarly, the **locRes.test.js** file encompasses tests for the **locationResourceController** module. These tests cover functionalities such as creating, retrieving, updating, and deleting location resource bookings.

Lastly, the **timetable.test.js** file contains tests for the **timetableController** module. This includes testing the creation, retrieval, updating, and deletion of timetable sessions, as well as fetching timetable sessions for a particular student.

Each test suite follows a structured approach, beginning with mocking relevant dependencies using **jest.mock**. Subsequently, it describes the controller function being tested and provides individual test cases using **it** blocks. Each test case sets up the necessary mock data, invokes the controller function under test, and asserts the expected behavior using **expect** statements.

Overall, the unit testing strategy ensures the correctness and reliability of the application's backend controllers, contributing to the overall quality and stability of the software system.

## Security Testing

Security testing was conducted on the University Timetable Management System's RESTful API using OWASP ZAP and Postman. These tools helped identify and address security vulnerabilities in line with industry standards.

The testing process involved configuring Postman and OWASP ZAP to communicate effectively, enabling the transfer of API requests from Postman to OWASP ZAP for analysis and testing. This was achieved by setting up proxy configurations in both tools, establishing a seamless flow of requests for evaluation.

Once the proxy configurations were properly set up, API requests were sent from Postman to OWASP ZAP for examination. This allowed for the simulation of various attack scenarios to assess the robustness of the API against common security threats.

## Performance Testing

Performance testing for University Timetable Management System conducted using Postman. This evaluates the API's performance under various loads to ensure it can handle multiple requests simultaneously without significant latency.

A screenshot of a computer

Description automatically generated