Functional and Non-Functional Requirements Documentation for EduLearn LMS

Shivaramakrishna Aravapally, Surya Prakash Reddy Pachika

September 3, 2024

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to outline the functional and non-functional requirements for the EduLearn Learning Management System (LMS) application. This document outlines the architecture and design of a microservices-based system. The project aims to create a scalable, maintainable, and secure platform that supports various functionalities such as student management, teacher management. Each functionality is encapsulated within its microservice, which communicates with others via RESTful APIs.

1.2 Scope

This document covers the requirements for the LMS application, which is designed to facilitate digital learning by enabling teachers to manage classes, assignments, materials, and sessions, and allowing students to access learning resources, submit assignments, and view grades.

2 Functional Requirements

2.1 Overview

This section details the functional requirements of the LMS application, describing the interactions between users (teachers and students) and the system.

2.2 User Registration and Authentication

- The system shall allow new users to register as teachers by providing their name, email address, username, and password.
- The system shall store teacher credentials, student credentials in their respective microservice.
- Students and teachers additional information is stored in their respective microservice.
- The system shall allow registered teachers to log in using their username and password.
- The system shall allow teachers to add students in the class by providing the credentials to students.

2.3 Teacher Dashboard

- The system shall allow teachers to view a dashboard displaying all the classes they are teaching.
- The system shall allow teachers to create new classes by providing class ID, name, and description.
- The system shall allow teachers to add students to their classes, with the student details (name, email, password) automatically assigned to the respective class ID.
- The system shall allow teachers to view the details of a specific class by selecting it from the dashboard.
- The system shall allow teachers to upload assignments, materials, videos content and can also schedule the sessions.

2.4 Student Dashboard

- The system shall allow students to view their assigned classes and access materials, assignments, and session details for each class.
- The system shall provide a Digital Library feature where students can search for educational content and view related YouTube videos within the platform.
- The system shall allow students to submit assignments for their classes.
- The system shall allow students to view their assignment submissions and receive feedback from teachers.

2.5 Assignment Management

- The system shall allow teachers to create and upload assignments, including assignment name, description, file link, and deadline.
- The system shall allow students to view and submit assignments for their classes.
- The system shall track the number of assignments submitted by students and allow teachers to view all submissions for a specific assignment.

2.6 Material and Video Management

- The system shall allow teachers to upload materials (documents, PDFs) and videos for each class.
- The system shall ensure that materials and videos are linked to specific classes, and only students enrolled in those classes can access them.

2.7 Session Management

- The system shall allow teachers to create and schedule sessions, providing session name, date, time, link, and description.
- The system shall display the number of sessions scheduled for each class on the class dashboard.
- The system shall allow students to view upcoming sessions and access session links from their dashboards.

3 Non-Functional Requirements

3.1 Performance Requirements

- The system shall respond to user actions (e.g., loading dashboards, submitting assignments).
- The system shall support concurrent users without performance degradation.

3.2 Security Requirements

- Passwords shall be stored securely using strong encryption methods (e.g., bcrypt).
- The system shall implement role-based access control (RBAC) to ensure that users can only access resources appropriate to their role (teacher or student).

3.3 Usability Requirements

- The system shall have a user-friendly interface designed with Tailwind CSS for consistency and ease of use.
- The user interface shall be intuitive, allowing users to navigate between different sections (e.g., dashboard, assignments) with minimal effort.

3.4 Reliability Requirements

- The system shall maintain an uptime of 99.9% to ensure availability for users.
- All user data, including assignments and materials, shall be regularly backed up to prevent data loss.

3.5 Scalability Requirements

- The system shall scale horizontally to accommodate an increase in the number of users and data volume without affecting performance.
- The system shall allow easy addition of new features and modules, such as adding new types of content (e.g., quizzes).

3.6 Maintainability Requirements

- The system shall follow modular design principles to ensure that different components (e.g., user authentication, material management) can be maintained independently.
- The codebase shall adhere to industry-standard coding conventions and include comprehensive documentation to facilitate future maintenance and updates.

4 Technologies Used

The project employs the following technologies:

- **Programming Language**: Java 17 for backend services.
- **Frameworks**:
  - Spring Boot for building microservices.
  - Spring Security for authentication and authorization.
  - Spring Cloud Netflix Eureka for service discovery.
  - Spring Cloud Gateway for API routing.
- **Database**: MongoDB for storing persistent data.
- **API Communication**: RESTful APIs using Spring Web.
- **Frontend**: React Framework for building the user interface.
- **Containerization**: Docker for containerizing microservices.

5 Design

5.1 Architectural Overview

The architecture is based on a microservices pattern, where each service is responsible for a specific capability. Each microservice runs independently, communicating with others via HTTP REST APIs. The key components include:

- **API Gateway**: Acts as a single entry point for all client requests, routing them to the appropriate microservice.
- **Student Service**: Manages the Student data retrives the data from teacher service relates to class related information.
- **Teacher Service**: Handles Teacher operations such as registration, login, provide access to student, upload materials and other content, and other class management.

5.2 System Interactions

This section describes how different components interact with each other. The interactions are illustrated using a wireframe diagram that shows the flow between different pages and components of the system.

- The dashboard allows users to log in.
- After logging in, teachers or students can access their profiles and dashboards accordingly.
- Teacher create class add student in class and manage the course related content.
- Students can access the class content uploaded by their teachers.

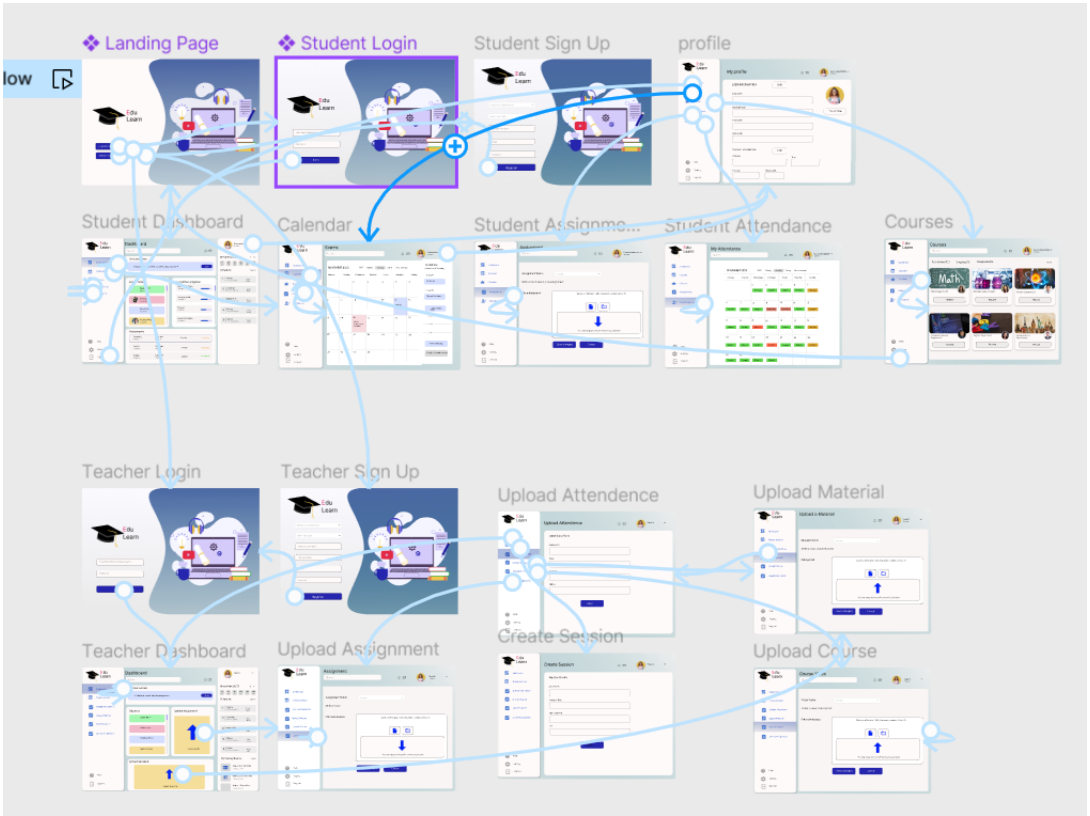

Figure 1: Wireframe

5.3 Use Case Diagram

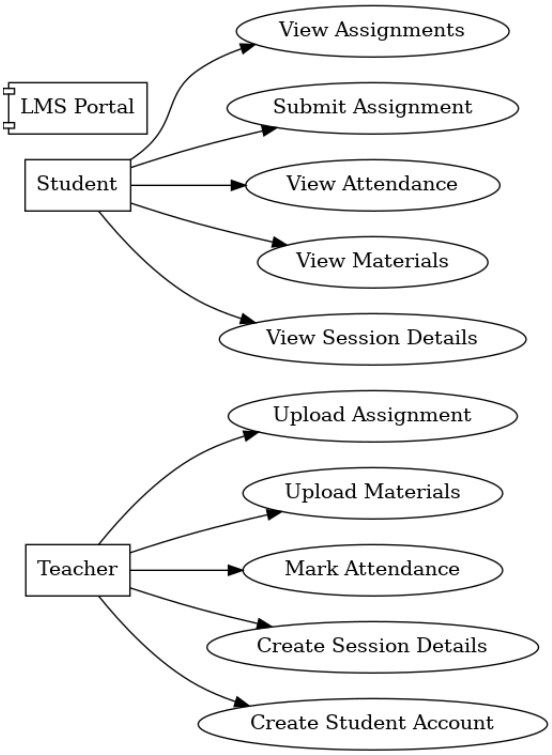

Figure 2: UseCase Diagram

5.4 Database Architecture

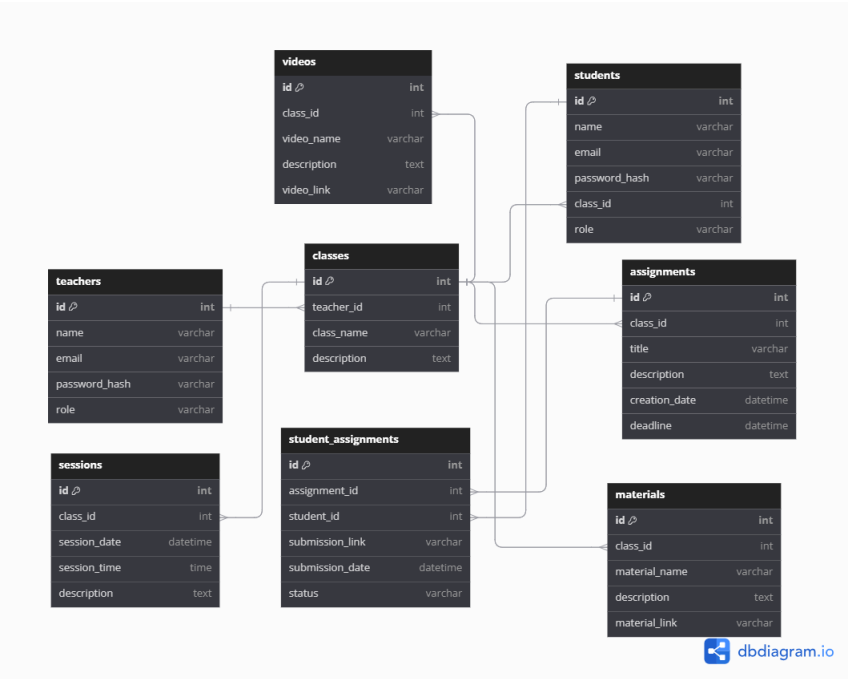

Figure 3: Database Schema

6 Conclusion

In conclusion, the Learning Management System (LMS) Portal is designed to address the significant challenges posed by the shift to digital learning. By providing a structured platform for managing assignments, sessions, materials, and attendance, the LMS portal supports both teachers and students in maintaining organization, motivation, and engagement. The focus on scalability, usability, and security ensures that the system can grow with the needs of its users, making it a valuable tool in the modern educational landscape.