Project: E-Learning Platform

1. Introduction

The purpose of this document is to provide a detailed Low-Level Design (LLD) for the E-Learning Platform. The platform supports interactive online learning by enabling course management, student enrollment, assessments, and performance tracking. It uses a REST API-based backend architecture and Angular or React for the frontend.

This design supports both Java (Spring Boot) and .NET (ASP.NET Core) frameworks.

2. Module Overview

The project consists of the following modules:

2.1 User Management

Handles registration, authentication, and role-based access for students and instructors.

2.2 Course Management

Allows instructors to create, update, and publish courses with multimedia content.

2.3 Enrollment Management

Enables students to enroll in courses and track their progress.

2.4 Assessment and Evaluation

Supports quizzes, assignments, and grading mechanisms.

2.5 Notifications and Alerts

Sends reminders for deadlines, course updates, and announcements.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React
- **Backend**: REST API-based architecture
- **Database**: Relational Database (MySQL/PostgreSQL/SQL Server)

3.2 Component Interaction

- The frontend communicates with the backend API for all operations.
- The backend handles business logic and interacts with the database.
- Notifications are sent via email, SMS, or displayed on the platform.

4. Module-Wise Design

4.1 User Management Module

4.1.1 Features

- Register as a student or instructor.
- Login and manage user profiles.

4.1.2 Data Flow

1. Users interact with the frontend to register/login.
2. Frontend sends user data to the REST API.
3. Backend authenticates users and interacts with the database.
4. Responses are sent back to the frontend for display.

4.1.3 Entities

- **User**
  - UserID
  - Name
  - Role (Student/Instructor)
  - Email
  - Password

4.2 Course Management Module

4.2.1 Features

- Create, update, and delete courses.
- Add multimedia content and resources to courses.

4.2.2 Data Flow

1. Instructors create courses via the frontend.
2. Frontend sends course data to the backend API.
3. Backend saves the course details to the database.
4. Courses are displayed to students.

4.2.3 Entities

- **Course**
  - CourseID
  - Title
  - Description
  - ContentURL
  - InstructorID

4.3 Enrollment Management Module

4.3.1 Features

- Students enroll in courses.
- Track enrollment status and progress.

4.3.2 Data Flow

1. Students request enrollment via the frontend.
2. Frontend sends enrollment requests to the backend API.
3. Backend updates the database and returns confirmation.

4.3.3 Entities

- **Enrollment**
    - EnrollmentID
    - StudentID
    - CourseID
    - Progress

4.4 Assessment and Evaluation Module

4.4.1 Features

- Create quizzes and assignments.
- Submit and grade assessments.

4.4.2 Data Flow

1. Instructors create assessments via the frontend.
2. Students submit assessments through the frontend.
3. Backend processes submissions and stores results in the database.
4. Grades are displayed to students.

4.4.3 Entities

- **Assessment**
    - AssessmentID
    - CourseID
    - Type (Quiz/Assignment)
    - MaxScore
- **Submission**
    - SubmissionID
    - AssessmentID
    - StudentID
    - Score

4.5 Notifications and Alerts Module

4.5.1 Features

- Notify students of upcoming deadlines and course updates.
- Notify instructors of submissions and queries.

4.5.2 Data Flow

1. Backend generates notifications based on events (e.g., new submissions).
2. Notifications are sent via email/SMS or displayed in the frontend.

5. Deployment Strategy

5.1 Local Deployment

- **Frontend**: Local Angular/React servers for development.
- **Backend**: REST API deployed using Spring Boot/ASP.NET Core.
- **Database**: Local database instance for development.

6. Database Design

6.1 Tables and Relationships

- **User**
  - Primary Key: UserID
- **Course**
  - Primary Key: CourseID
  - Foreign Key: InstructorID
- **Enrollment**
  - Primary Key: EnrollmentID
  - Foreign Keys: StudentID, CourseID
- **Assessment**
  - Primary Key: AssessmentID
  - Foreign Key: CourseID
- **Submission**
  - Primary Key: SubmissionID
  - Foreign Keys: AssessmentID, StudentID

7. User Interface Design

7.1 Wireframes

- **Student Dashboard**: View enrolled courses and progress.
- **Instructor Dashboard**: Manage courses and assessments.
- **Assessment Page**: Submit quizzes and assignments.

8. Non-Functional Requirements

8.1 Performance

- System must handle up to 200 concurrent users during peak hours.

8.2 Scalability

- Designed to scale for production environments.

8.3 Security

- Ensure secure login, role-based access control, and encrypted data storage.

8.4 Usability

- User interface must be intuitive and responsive for all users.

9. Assumptions and Constraints

9.1 Assumptions

- The platform will operate in a local environment during development.

9.2 Constraints

- No third-party cloud integrations in the initial phase.