# Project: Online Education Platform

## 1. Introduction

This document outlines the Low-Level Design (LLD) for an **Online Education Platform** that provides a comprehensive learning management system (LMS) for students and instructors. The platform allows students to enroll in courses, participate in assignments, track their progress, and communicate with instructors. Instructors can create courses, upload content, and manage student progress.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks for backend development.

## 2. Module Overview

### 2.1 Course Management Module

- Allows instructors to create and manage courses, including lessons, quizzes, and assignments.
- Students can view course details and enroll.

### 2.2 User Management Module

- Manages user profiles, roles (student, instructor, admin), and permissions.
- Handles user authentication and registration.

### 2.3 Assignment and Quiz Management Module

- Manages assignments and quizzes for students to complete as part of their coursework.
- Includes grading and feedback features.

### 2.4 Communication Module

- Enables interaction between students and instructors through forums, messaging, and announcements.
- Provides real-time notifications for updates related to courses and assignments.

### 2.5 Reporting and Analytics Module

- Generates reports on student progress, course completion, grades, and engagement.
- Provides insights into student performance and course effectiveness.

# 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 via REST APIs to fetch course data, submit assignments, and manage user interactions.
- The backend processes requests, interacts with the database for CRUD operations, and returns responses to the frontend.

# 4. Module-Wise Design

## 4.1 Course Management Module

**Features**:

- Instructors can create, update, and manage courses.
- Students can browse courses, view course details, and enroll.

**Data Flow**:

- Instructors interact with the frontend to create/update course details.
- The backend processes these requests, updates the database, and returns the appropriate responses.

**Entities**:

- **Course**:
  - CourseID
  - Title
  - Description
  - InstructorID
  - Category

## 4.2 User Management Module

**Features**:

- Registration, login, and authentication for students and instructors.
- Role-based access control to determine permissions.

**Data Flow**:

- Users register and log in via the frontend.
- The backend authenticates the user, retrieves profile data, and authorizes access based on roles.

**Entities**:

- **User**:
  - UserID
  - Name
  - Email

    - Role (Student, Instructor, Admin)
    - Password

### 4.3 Assignment and Quiz Management Module

**Features**:

- Instructors can create assignments and quizzes with questions.
- Students can submit completed assignments and take quizzes.

**Data Flow**:

- Instructors add assignments and quizzes through the frontend.
- The backend stores the assignments and quizzes in the database.
- Students can submit assignments and take quizzes via the frontend, with results processed by the backend.

**Entities**:

- **Assignment**:
    - AssignmentID
    - CourseID
    - Title
    - DueDate
    - TotalMarks
- **Quiz**:
    - QuizID
    - CourseID
    - Question
    - AnswerOptions
    - CorrectAnswer

### 4.4 Communication Module

**Features**:

- Students and instructors can post and reply to messages in forums.
- Real-time notifications for important updates (new course material, assignment deadlines).

**Data Flow**:

- Students and instructors interact through the frontend, sending messages or posting to forums.
- The backend processes communication requests and updates the database.
- Notifications are generated and sent to users based on predefined triggers.

**Entities**:

- **ForumPost**:
    - PostID
    - CourseID
    - UserID (Student/Instructor)
    - Message
    - Timestamp
- **Notification**:

  - NotificationID
  - UserID
  - Message
  - Type (Assignment Due, New Course, Announcement)

### 4.5 Reporting and Analytics Module

**Features**:

- Provides reports on student grades, course progress, and engagement metrics.
- Generates data visualizations for performance tracking.

**Data Flow**:

- Admins and instructors can generate reports via the frontend.
- The backend aggregates and processes data, returning reports to the frontend.

**Entities**:

- **Report**:
  - ReportID
  - ReportType (Grade, Progress, Engagement)
  - Data

## 5. Deployment Strategy

### 5.1 Local Deployment

- Frontend: Use ng serve for Angular or equivalent local servers for React.
- Backend: Deploy the REST API locally using Spring Boot or ASP.NET Core.
- Database: Set up a local instance of MySQL/PostgreSQL/SQL Server for development.

## 6. Database Design

### 6.1 Tables and Relationships

1. **Course**
   - Primary Key: CourseID
   - Foreign Key: InstructorID
2. **User**
   - Primary Key: UserID
   - Foreign Key: None
3. **Assignment**
   - Primary Key: AssignmentID
   - Foreign Key: CourseID
4. **Quiz**
   - Primary Key: QuizID
   - Foreign Key: CourseID
5. **ForumPost**
   - Primary Key: PostID
   - Foreign Key: CourseID, UserID
6. **Notification**

    - Primary Key: NotificationID
    - Foreign Key: UserID

## 7. User Interface Design

**7.1 Wireframes**:

- Course Catalog Page
- Course Enrollment and Details Page
- Assignment Submission Page
- Quiz Interface
- Messaging/Forum Page
- Admin Dashboard for Reports

## 8. Non-Functional Requirements

### 8.1 Performance

- The platform should be capable of handling up to 300 concurrent users, including students and instructors.

### 8.2 Scalability

- The system should be designed to scale horizontally, supporting thousands of users and courses.

### 8.3 Security

- Secure user authentication with password hashing and two-factor authentication.
- Role-based access control to ensure only authorized users can access certain features.

### 8.4 Usability

- The user interface should be responsive, with fast page load times and an intuitive navigation structure.