1. Introduction

This document outlines the Low-Level Design (LLD) for an **Online Learning Management System** (LMS), enabling instructors to create courses, manage students, and track learning progress. The system allows students to enroll, take quizzes, access resources, and view 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

- Manage course creation, editing, deletion, and categorization.
- Assign instructors to courses and set course prerequisites.

2.2 User Registration and Profile Management Module

- Allows users (students and instructors) to register, log in, and manage profiles.

2.3 Enrollment and Access Management Module

- Students can enroll in courses and access learning materials.

2.4 Quiz and Assessment Module

- Instructors create quizzes and assessments. Students can take quizzes and receive feedback.

2.5 Progress Tracking and Reports Module

- Track student progress across courses, including quiz scores and completed lessons.

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 for course browsing, enrollment, quiz management, and profile management.
- The backend handles user authentication, course creation, enrollment, and quiz/assessment management.

4. Module-Wise Design

4.1 Course Management Module

Features:

- Instructors can create, update, and delete courses.
- Course metadata, such as title, description, syllabus, and prerequisites.

Data Flow:

- Instructors initiate course creation/update requests via the frontend.
- The backend validates input, stores course data in the database, and links the course to the appropriate instructor.

Entities:

- **Course**:
    - CourseID
    - Title
    - Description
    - Syllabus
    - InstructorID
    - Prerequisites

4.2 User Registration and Profile Management Module

Features:

- Registration for students and instructors.
- Profile management for both students and instructors.

Data Flow:

- Users register via the frontend, providing their details (name, email, role).
- The backend validates user input and stores user details in the database.
- Users can update their profiles through the frontend interface.

Entities:

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

4.3 Enrollment and Access Management Module

Features:

- Students can enroll in available courses.
- Manage student access to learning materials.

Data Flow:

- Students browse available courses and submit an enrollment request via the frontend.
- Backend processes enrollment requests and links the student to the chosen course.

Entities:

- **Enrollment**:
  - EnrollmentID
  - CourseID
  - UserID (Student)
  - EnrollmentDate

4.4 Quiz and Assessment Module

Features:

- Instructors create quizzes with multiple-choice or short-answer questions.
- Students take quizzes and get feedback.

Data Flow:

- Instructors create quizzes via the frontend, providing questions and possible answers.
- Students attempt quizzes via the frontend, with answers submitted to the backend for evaluation.

Entities:

- **Quiz**:
  - QuizID
  - CourseID
  - Title
  - QuestionList
  - TotalMarks
- **Answer**:
  - AnswerID
  - QuizID
  - UserID
  - Response
  - Marks

4.5 Progress Tracking and Reports Module

Features:

- Track student performance across courses.
- Generate reports based on quiz scores and lesson completion.

Data Flow:

- Students' quiz attempts and completed lessons are tracked in the backend.
- Reports are generated on demand by the instructor or system for each student.

Entities:

- **Progress**:
  - ProgressID
  - UserID (Student)
  - CourseID
  - CompletedLessons
  - QuizScores

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 testing and development.

6. Database Design

6.1 Tables and Relationships

1. **Course**
   - Primary Key: CourseID
   - Foreign Key: InstructorID
2. **User**
   - Primary Key: UserID
3. **Enrollment**
   - Primary Key: EnrollmentID
   - Foreign Key: UserID, CourseID
4. **Quiz**
   - Primary Key: QuizID
   - Foreign Key: CourseID
5. **Answer**
   - Primary Key: AnswerID
   - Foreign Key: QuizID, UserID
6. **Progress**
   - Primary Key: ProgressID
   - Foreign Key: UserID, CourseID

7. User Interface Design

7.1 Wireframes:

- Course Listing Page
- Course Enrollment Page
- Student Dashboard
- Instructor Dashboard
- Quiz Attempt Page
- Progress Report Page

8. Non-Functional Requirements

8.1 Performance

- The system should be able to support up to 500 concurrent users.

8.2 Scalability

- The design should scale easily to accommodate increasing courses, users, and data.

8.3 Security

- Ensure secure user authentication, authorization, and data encryption.

8.4 Usability

- The system should be intuitive, responsive, and user-friendly across devices.