Project: Online Exam Portal

1. Introduction

This document provides the Low-Level Design (LLD) for an **Online Exam Portal** aimed at streamlining the creation, management, and evaluation of online assessments.

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

2. Module Overview

2.1 Admin Module

- Create and manage exams, questions, and user roles.

2.2 User Module

- User registration, login, and profile management.

2.3 Exam Management Module

- Attempt exams, view results, and provide feedback.

2.4 Question Bank Module

- Manage a repository of categorized questions.

2.5 Analytics and Reporting Module

- Generate performance reports and insights.

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 through REST APIs for managing exams, users, and reports.
- The backend handles database operations and processes results and analytics.

4. Module-Wise Design

4.1 Admin Module

Features:

- Create, update, and delete exams and questions.
- Assign roles (e.g., Examiner, Student).

Data Flow:

- Admin submits exam details via the frontend.
- Backend validates and stores the data in the database.
- Confirmation is displayed on the frontend.

Entities:

- **Exam**:
    - ExamID
    - Title
    - Description
    - Duration
    - TotalMarks

4.2 User Module

Features:

- User registration and login.
- View and update profiles.

Data Flow:

- User submits details via the frontend.
- Backend validates, encrypts sensitive data, and stores it in the database.

Entities:

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

4.3 Exam Management Module

Features:

- Display exam questions and record responses.
- Calculate and display results.

Data Flow:

- User selects an exam via the frontend.
- Backend fetches questions and stores user responses.
- Results are calculated and displayed to the user.

Entities:

- **Response**:
    - ResponseID
    - ExamID

    - UserID
    - QuestionID
    - Answer
    - MarksObtained

4.4 Question Bank Module

Features:

- Manage questions, including categories and difficulty levels.
- Import/export questions in bulk.

Data Flow:

- Admin adds or imports questions via the frontend.
- Backend validates and stores the questions in the database.

Entities:

- **Question**:
    - QuestionID
    - Text
    - Category
    - Difficulty
    - CorrectAnswer

4.5 Analytics and Reporting Module

Features:

- View individual and aggregate performance reports.
- Export data for analysis.

Data Flow:

- Admin selects report criteria via the frontend.
- Backend processes the data and returns it to the frontend.

Entities:

- **Report**:
    - ReportID
    - ExamID
    - UserID
    - TotalMarks
    - PerformanceMetrics

5. Deployment Strategy

5.1 Local Deployment

- Frontend: Served using local servers (e.g., ng serve for Angular or equivalent for React).
- Backend: Deployed locally using Spring Boot or ASP.NET Core.
- Database: Local instance of the relational database for testing.

6. Database Design

6.1 Tables and Relationships

1. **Exam**
    - Primary Key: ExamID
2. **User**
    - Primary Key: UserID
3. **Question**
    - Primary Key: QuestionID
4. **Response**
    - Primary Key: ResponseID
    - Foreign Keys: ExamID, UserID, QuestionID
5. **Report**
    - Primary Key: ReportID
    - Foreign Keys: ExamID, UserID

7. User Interface Design

7.1 Wireframes:

- Admin Dashboard
- User Registration/Login Page
- Exam Attempt Page
- Results and Analytics Dashboard

8. Non-Functional Requirements

8.1 Performance

- The portal must handle 500 concurrent users.

8.2 Scalability

- Designed for multiple institutes and large user bases.

8.3 Security

- Implement secure authentication and access control.

8.4 Usability

- Ensure an intuitive and responsive user interface.