Online Course and Exam Platform – System Design Document

1. Introduction

This document describes the complete **system design** of an **Online Course and Exam Platform** built using **React (Frontend)**, **Spring Boot (Backend)**, and **PostgreSQL (Database)**. The platform enables digital learning, automated examinations, and progress tracking for students, instructors, and administrators.

The document covers: - Functional & Non-Functional Requirements - High-Level Design (HLD) - Low-Level Design (LLD) - Module Breakdown (8 modules: 3 Frontend + 5 Backend) - Database Design - API Design - Exam Engine Design - Security, Scalability & Evaluation Focus

---

2. Requirements Analysis

2.1 Functional Requirements

Admin

- Manage users (students & instructors)
- Approve/reject instructor accounts
- Create categories & courses
- View platform-wide reports

Instructor

- Create & manage courses
- Upload study materials (PDF, video links)
- Create exams (MCQ-based)
- View student performance

Student

- Register & login
- Enroll in courses
- Attempt exams with timer
- View results & progress

Exam Engine

- Timed MCQ exams
- Auto-submit on timeout
- Randomized questions

- Instant evaluation

Reporting

- Student progress reports
- Exam analytics
- Instructor & admin dashboards

2.2 Non-Functional Requirements

- **Scalability**: Handle thousands of concurrent users
- **Performance**: Low-latency exam submission
- **Security**: JWT-based authentication, role-based access
- **Availability**: 99.9% uptime
- **Usability**: Responsive UI for mobile & desktop
- **Maintainability**: Modular architecture

---

3. Technology Stack

Layer	Technology
Frontend	React, HTML, CSS, JavaScript
Backend	Spring Boot, REST APIs
Database	PostgreSQL
Security	Spring Security, JWT
Build	Maven
Deployment	Docker, Cloud VM

---

4. Module Breakdown (8 Modules)

Frontend Modules (3)

1. **Admin & Instructor UI Module**
2. **Student UI Module**
3. **Exam UI Module**

Backend Modules (5)

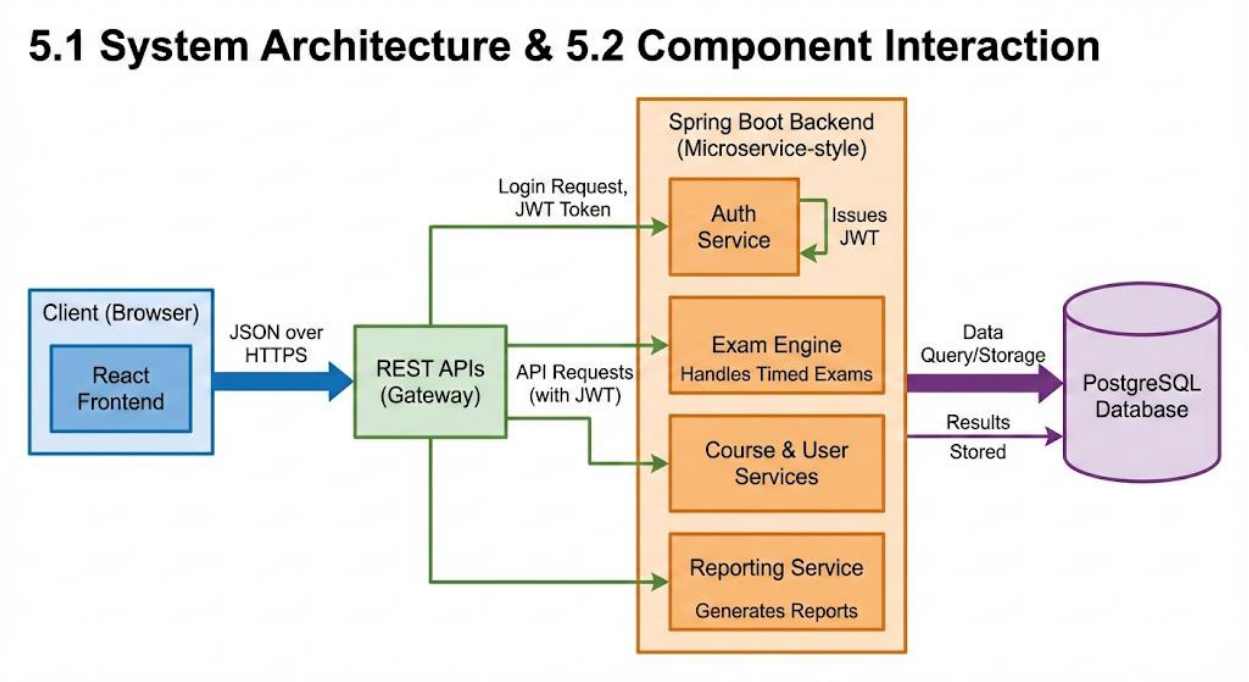
4. **Authentication & User Management Service**
5. **Course & Content Management Service**
6. **Exam Engine Service**
7. **Evaluation & Result Processing Service**
8. **Reporting & Analytics Service**

5. High-Level Design (HLD)

5.1 System Architecture

Client (Browser) → React Frontend → REST APIs → Spring Boot Backend (Microservice-style) → PostgreSQL Database

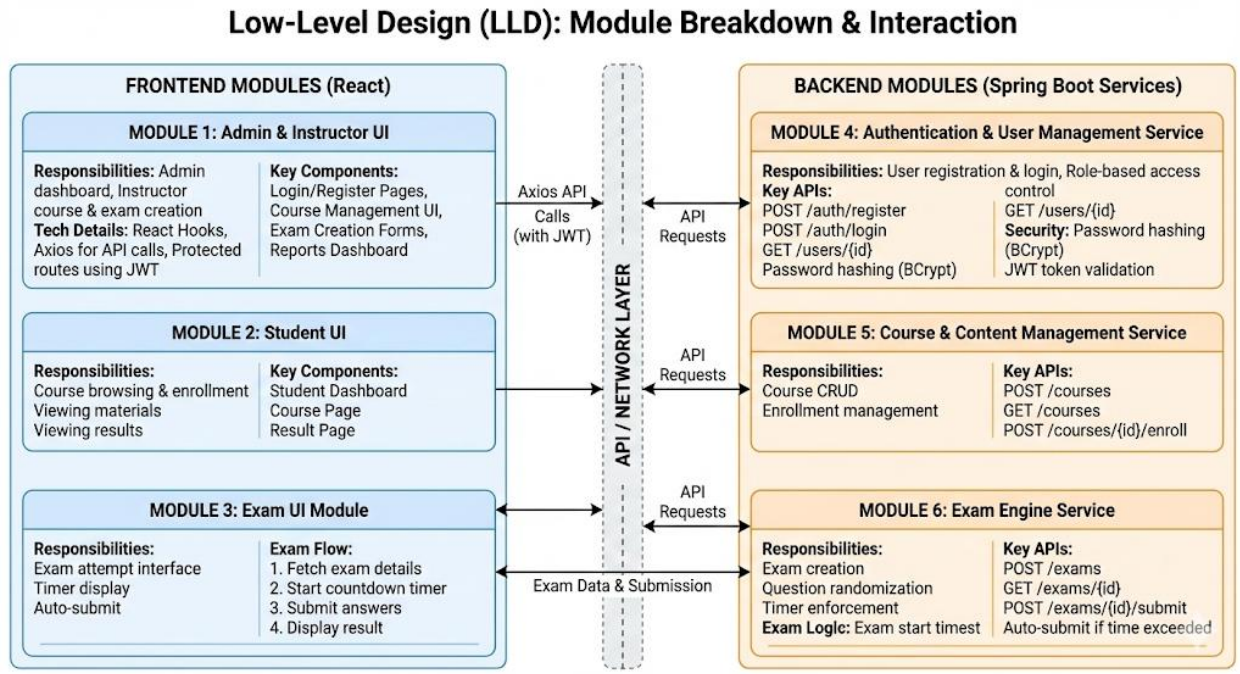
All communication between frontend and backend happens via **JSON over HTTPS**.



5.2 Component Interaction

1. User logs in → Auth Service issues JWT
2. React stores JWT and sends it in headers
3. Backend validates role (Admin/Instructor/Student)
4. Requests routed to respective services
5. Exam Engine handles timed exams
6. Results stored & reports generated

6. Low-Level Design (LLD)


Low-Level Design (LLD): Module Breakdown & Interaction
FRONTEND MODULES (React)
MODULE 1: Admin & Instructor UI
Responsibilities: Admin dashboard, Instructor course & exam creation
Tech Details: React Hooks, Axios for API calls, Protected routes using JWT
Key Components: Login/Register Pages, Course Management UI, Exam Creation Forms, Reports Dashboard
MODULE 2: Student UI
Responsibilities: Course browsing & enrollment Viewing materials Viewing results
Key Components: Student Dashboard Course Page Result Page
MODULE 3: Exam UI Module
Responsibilities: Exam attempt interface Timer display Auto-submit
Exam Flow: 1. Fetch exam details 2. Start countdown timer 3. Submit answers 4. Display result
Axios API Calls (with JWT)
API / NETWORK LAYER
API Requests
API Requests
API Requests
Exam Data & Submission
BACKEND MODULES (Spring Boot Services)
MODULE 4: Authentication & User Management Service
Responsibilities: User registration & login, Role-based access control
Key APIs: POST /auth/register POST /auth/login GET /users/{id} Password hashing (BCrypt)
GET /users/{id}
Security: Password hashing (BCrypt) JWT token validation
MODULE 5: Course & Content Management Service
Responsibilities: Course CRUD Enrollment management
Key APIs: POST /courses GET /courses POST /courses/{id}/enroll
MODULE 6: Exam Engine Service
Responsibilities: Exam creation Question randomization Timer enforcement
Exam Logic: Exam start timest
Key APIs: POST /exams GET /exams/{id} POST /exams/{id}/submit Auto-submit if time exceeded

Module 1: Admin & Instructor UI (Frontend)

Responsibilities

- Admin dashboard
- Instructor course & exam creation

Key Components

- Login/Register Pages
- Course Management UI
- Exam Creation Forms
- Reports Dashboard

Tech Details

- React Hooks
- Axios for API calls
- Protected routes using JWT

Module 2: Student UI (Frontend)

Responsibilities

- Course browsing & enrollment

- Viewing materials
- Viewing results

Key Components

- Student Dashboard
- Course Page
- Result Page

Module 3: Exam UI Module (Frontend)

Responsibilities

- Exam attempt interface
- Timer display
- Auto-submit

Exam Flow

1. Fetch exam details
2. Start countdown timer
3. Submit answers
4. Display result

Module 4: Authentication & User Management Service (Backend)

Responsibilities

- User registration & login
- Role-based access control

Key APIs

- POST /auth/register
- POST /auth/login
- GET /users/{id}

Security

- Password hashing (BCrypt)
- JWT token validation

Module 5: Course & Content Management Service (Backend)

Responsibilities

- Course CRUD
- Enrollment management

Key APIs

- POST /courses
- GET /courses
- POST /courses/{id}/enroll

Module 6: Exam Engine Service (Backend)

Responsibilities

- Exam creation
- Question randomization
- Timer enforcement

Exam Logic

- Exam start timestamp stored
- Server validates submission time
- Auto-submit if time exceeded

Key APIs

- POST /exams
- GET /exams/{id}
- POST /exams/{id}/submit

Module 7: Evaluation & Result Processing Service (Backend)

Responsibilities

- MCQ evaluation
- Score calculation
- Result storage

Evaluation Flow

1. Fetch correct answers
2. Compare responses
3. Calculate score

4. Store result

Module 8: Reporting & Analytics Service (Backend)

Responsibilities

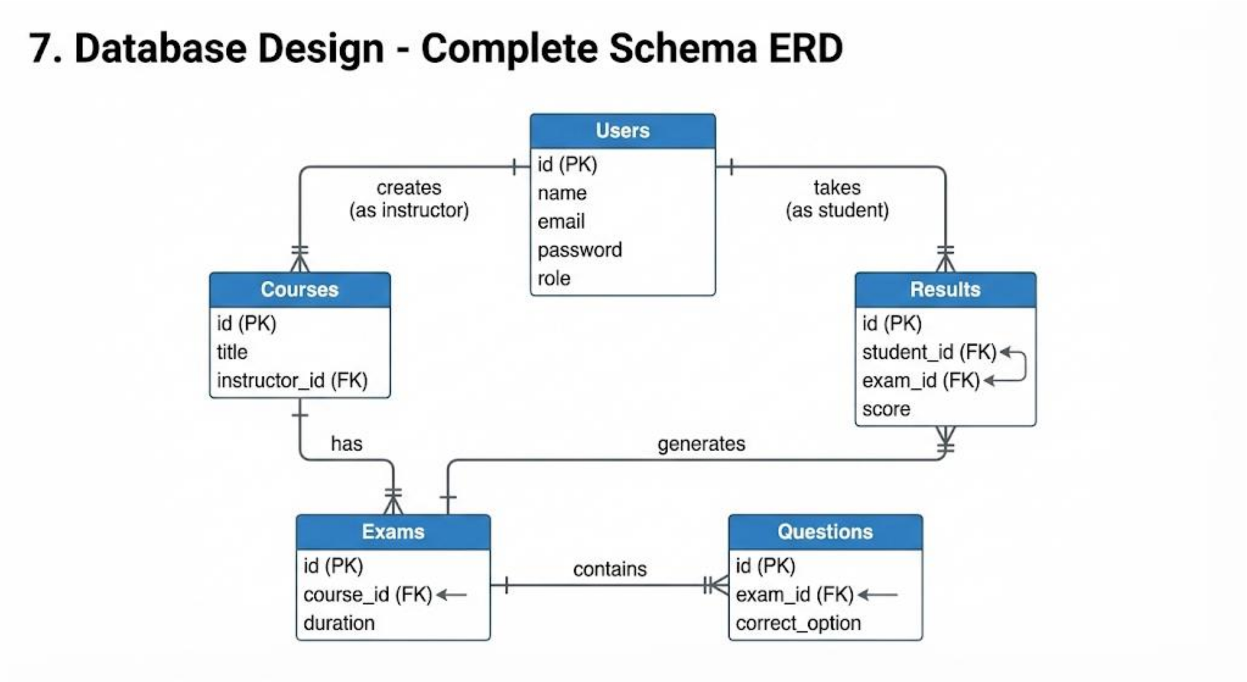
- Generate reports
- Track progress

Reports

- Student performance report
- Course completion rate
- Exam difficulty analysis

---

7. Database Design



Key Tables

Users

- id (PK)
- name
- email
- password
- role

Courses

- id (PK)
- title
- instructor_id (FK)

Exams

- id (PK)
- course_id (FK)
- duration

Questions

- id (PK)
- exam_id (FK)
- correct_option

Results

- id (PK)
- student_id (FK)
- exam_id (FK)
- score

---

8. Security Design

- JWT Authentication
- Role-based authorization
- HTTPS
- Input validation
- SQL Injection protection (JPA)

---

9. Scalability & Performance

- Stateless backend services
- Database indexing
- Horizontal scaling with load balancer
- Caching frequently accessed data

---

10. Evaluation Focus (as per Project)

- **MCQ + Timer Exam Engine**: Accurate & cheat-resistant
- **Backend Design Quality**: Clean layered architecture
- **UI/UX**: Responsive, intuitive dashboards

11. Conclusion

This system design ensures a **scalable, secure, and modular online learning platform** suitable for academic evaluation, real-world deployment, and future feature expansion (AI proctoring, live classes, etc.).