Team Member 1: User Management Module

Assigned Tasks

1. **User Role Management**
    - Implement role-based access control using **Spring Security**.
    - Create a UserController with endpoints for:
        - **User Registration**: Create a user and assign roles.
        - **User Login**: Authenticate users.
        - **Profile Management**: Fetch and update user profile details.
    - Mock the user data using in-memory data structures (e.g., HashMap) instead of a database.
2. **Testing**
    - Write **JUnit tests** for:
        - User registration and login.
        - Ensuring unauthorized users cannot access restricted endpoints.
        - Role-based access validation.

Endpoints

- POST /register - Register a user.
- POST /login - Login a user.
- GET /profile/{id} - View profile.
- PUT /profile/{id} - Update profile.

Team Member 2: Course Management Module

Assigned Tasks

1. **Course Handling**
    - Implement CourseController with endpoints for:
        - Course creation (Instructor only).
        - Fetching all courses.
        - Enrolling students in courses (Student only).
        - Fetching enrolled students for a course (Admin/Instructor only).
    - Create a Course service to manage:
        - Adding mock data for courses (e.g., ArrayList).
        - Managing course enrollments in-memory.
2. **Testing**
    - Write **JUnit tests** for:
        - Course creation by valid instructors.
        - Enrollment of valid students.
        - Fetching course lists and enrolled students.

Endpoints

- POST /courses - Create a course (Instructor only).
- GET /courses - Fetch all courses.
- POST /courses/{id}/enroll - Enroll in a course (Student only).
- GET /courses/{id}/students - View enrolled students (Admin/Instructor).

Team Member 3: Assessment & Grading Module

Assigned Tasks

1. **Quiz and Assignment Management**
    - Implement AssessmentController with endpoints for:
        - Creating quizzes and assignments (Instructor only).
        - Submitting assignments (Student only).
        - Grading assignments and quizzes (Instructor only).
    - Use mock structures for storing:
        - Question banks for quizzes.
        - Assignment submissions by students.
    - Add functionality to randomize quiz questions during retrieval.
2. **Testing**
    - Write **JUnit tests** for:
        - Creating quizzes with randomized questions.
        - Submitting and retrieving assignments.
        - Grading functionality for instructors.

Endpoints

- POST /assessments/quizzes - Create a quiz.
- POST /assessments/quizzes/{id}/attempt - Attempt a quiz.
- POST /assessments/assignments - Create an assignment.
- POST /assessments/assignments/{id}/submit - Submit an assignment.

- POST /assessments/assignments/{id}/grade - Grade an assignment.

Team Member 4: Attendance, Notifications, and Performance Tracking Module

Assigned Tasks

1. **Attendance Management**
    - Implement AttendanceController with endpoints for:
        - Generating OTP for a lesson (Instructor only).
        - Submitting OTP to mark attendance (Student only).
    - Manage attendance using mock structures.
2. **Notifications**
    - Implement NotificationController for:
        - Sending system notifications (e.g., enrollment confirmation, grades posted).
        - Fetching notifications (unread/all).
3. **Performance Tracking**
    - Create a PerformanceService for:
        - Tracking grades, attendance, and assignment completion.
4. **Testing**
    - Write **JUnit tests** for:
        - Attendance marking and validation.
        - Notification fetching.
        - Performance data retrieval.

Endpoints

- POST /attendance/{lessonId}/generate - Generate OTP (Instructor).

- POST /attendance/{lessonId}/submit - Submit OTP (Student).
- GET /notifications - Fetch notifications.
- GET /performance/{studentId} - View student performance.