Project Overview:

Title: **Advanced Course Management System**

Create a simple course management system where administrators, instructors, and students have different roles and permissions. The system will allow instructors to create courses, manage student enrollment, and record grades. Administrators should manage users and permissions. Students can view their enrolled courses and grades.

Requirements:

1. **Authentication and Authorization**:
    - Implement a role-based system with three user types: **Admin**, **Instructor**, and **Student**.
    - Use Laravel's built-in authentication and authorization mechanisms.
    - Ensure only admins can manage users, instructors can manage their courses and students, and students can only access their own course details and grades.
    - Use policies or gates to restrict access to certain functionalities.
2. **Database Schema**:
    - Design and implement the following database tables:
        - **Users**: (for authentication, including roles)
        - **Courses**: (title, description, instructor_id, etc.)
        - **Enrollments**: (student_id, course_id, enrollment_date)
        - **Grades**: (student_id, course_id, grade)
    - Relationships:
        - A user can be an **Instructor** or **Student**.
        - A course belongs to an **Instructor**.
        - Students can enroll in multiple courses.
        - A course has many students, and students receive grades for each course.
3. **Course Management (for Instructors)**:
    - Instructors can create, update, and delete courses.
    - Instructors can view the list of students enrolled in their courses.
    - Implement a feature where an instructor can upload a CSV file to enroll students in a course (implement CSV import logic).
    - After enrolling students, the instructor can update grades for each student in their courses.
4. **Student Dashboard**:
    - Students can view a dashboard showing:
        - List of courses they are enrolled in.
        - Grades for each course.
5. **Admin Panel**:
    - Admin can view, add, edit, and delete users (instructors and students).
    - Admin can assign roles to users (instructor or student).
6. **Complex Queries**:

    - Write an optimized query to fetch the top 5 students (based on average grade) enrolled in a specific course.
    - Write a query to get the list of courses that have the highest average grade for a specific instructor.
7. **Notifications**:
    - When a student is enrolled in a course or when their grade is updated, send them an email notification using Laravel's built-in notification system.
    - Implement a cron job to send a weekly summary email to instructors with a list of students and their current grades for each course.
8. **Code Quality and Optimization**:
    - Use Eloquent for relationships but also demonstrate when and how you would optimize queries using DB::raw or joins for performance in certain areas (e.g., fetching students with average grades).
    - Use caching (e.g., `Redis` or `file`) to store the list of enrolled students for a course to minimize database hits on repeated requests.
    - Demonstrate use of services or repositories to abstract complex business logic.

Project Deliverables:

1. **Functional Application**: A working Laravel project with the described features.
2. **Database Design**: A SQL file with the schema or migrations for creating the database structure.
3. **Documentation**: A README file describing how to set up the project, including:
    - Installation steps
    - Database seeding commands (if any)
    - How to run the cron job for weekly email summaries
    - Any challenges faced and how they were resolved
4. **Code Quality**: The project will be reviewed for clean, maintainable code, proper use of Laravel features (policies, middleware, notifications, etc.), and SQL optimization.