Program Title: Online Task Management System

Problem Statement:

You are tasked with building an Online Task Management System using **Spring Boot** that allows users to manage tasks, track their progress, and collaborate with team members. The system will have the following key features:

Key Features:

1. **User Management & JWT Authentication**:
    - The system should allow users to **sign up**, **log in**, and **manage their profile**.
    - Implement **JWT Authentication** for secure login and authorization. Users should receive a JWT token upon successful login that will be used for subsequent requests.
    - The system must support different types of users:
        - **Admin**: Can view and manage all tasks and users.
        - **Manager**: Can create, update, assign, and delete tasks.
        - **Employee**: Can view and update their assigned tasks.
    - Implement **role-based authorization** using JWT tokens to restrict access to certain API endpoints based on user roles.
2. **Task Management**:
    - Users should be able to create, update, and delete tasks. A task will have the following attributes:
        - Title
        - Description
        - Assigned user(s)
        - Due date
        - Status (e.g., Pending, In Progress, Completed)
    - Tasks can be **assigned to multiple users**.
    - A user should be able to filter tasks by status, due date, or assigned user.
3. **API Versioning**:
    - Implement **API versioning** to ensure backward compatibility for future updates. For example:
        - /api/v1/tasks
        - /api/v2/tasks (with some additional features or changes)
    - The system should support **both old and new API versions**.

4. **Real-Time Notifications with WebSockets**:
    - Integrate **WebSockets** for real-time communication between the server and clients.
    - When a task is created, updated, or assigned, the respective users should receive **real-time notifications** about the changes.
    - Display real-time updates for task status changes or comments.
5. **Task Scheduler**:
    - Use **Spring Scheduler** to automatically remind users of upcoming or overdue tasks.
    - The system should send an email notification to users **24 hours before a task's due date**.
    - The email reminder functionality should be integrated and configurable to allow different notification times (e.g., 1 day, 1 hour before due date).
6. **Audit Logging**:
    - Every action taken on the system (such as task creation, deletion, user management, and task assignment) should be logged for audit purposes.
    - Log entries should include the **timestamp**, **user performing the action**, and a brief description of the action taken.
7. **Pagination & Sorting**:
    - Implement **pagination** for fetching tasks, users, and other lists. The results should be returned in a paginated format, e.g., 20 tasks per page.
    - Support **sorting** tasks by various attributes (e.g., due date, status, title).
8. **Swagger Documentation**:
    - Provide **Swagger UI** for the API to allow easy testing and understanding of the available endpoints.
    - Ensure the API endpoints are well-documented with appropriate request/response structures.

Additional Optional Features (For Extra Challenges):

1. **Email Verification**:
    - Implement email verification upon user registration. The user should receive a verification link to activate their account.
2. **File Upload**:
    - Allow users to upload files (such as images or documents) for each task. Store these files securely and allow users to download them.

3. **Task Comments**:
    - Users should be able to add comments to tasks, which will be visible to other users assigned to the task.
4. **User Password Recovery**:
    - Implement password recovery via email (e.g., forgot password flow with a reset token).

Technologies and Tools:

- **Spring Boot** for building the backend.
- **Spring Security** with **JWT** for authentication and authorization.
- **Spring Data JPA** for database interactions (with Hibernate).
- **Spring WebSocket** for real-time communication.
- **Spring Scheduler** for automatic task notifications.
- **MySQL/PostgreSQL** as the database.
- **Swagger** for API documentation.
- **Thymeleaf** or **RESTful** email notifications for reminders.

Deliverables:

1. **Source Code** with a detailed README on how to run the application.
2. **Database Schema** with the tables defined for users, tasks, and logs.
3. **Swagger API Documentation**.

Evaluation Criteria:

- **Code Quality**: Proper use of Spring Boot conventions, clean and modular code.
- **Security**: Proper implementation of JWT Authentication and role-based access control.
- **Real-Time Functionality**: Efficient use of WebSockets for notifications.
- **Scheduler Functionality**: Proper integration of Spring Scheduler for task notifications.
- **Documentation**: Comprehensive API documentation using Swagger and clear README.