CMJD - Comprehensive Master Java Developer
Course work – Batch 110/111

Assignment 1: Backend Development with Spring Boot and MySQL

Task: Research Project Tracker for an Educational Institute

Objective

Develop and implement business logic that meets the requirements of a Research Project Tracker System, enabling users to create, manage, and monitor academic research projects within an educational environment. This system should include **authentication, project management, and role-based access control.**

Technology Stack

• Backend Framework: Spring Boot
• Database Layer: Spring Data JPA with MySQL
• Security Layer: Spring Security with JWT (JSON Web Token) Authentication
• Build Tool: Maven
• Version Control: GitHub (repository link required upon submission)

Task Flow

Spring Boot Setup

• Create a Spring Boot project with relevant dependencies:
  - Spring Web
  - Spring Data JPA
  - Spring Security
  - MySQL Driver
  - Lombok (optional for cleaner code)
• Configure the data persistence environment by integrating database connectivity.

• Suggested package structure (You may modify it as necessary)

```
lk.ijse.cmjd.researchtracker
 ├─ auth/
 ├─ project/
 ├─ milestone/
 ├─ document/
 ├─ user/
 ├─ config/
 └─ common/
```

Entities and Enumerations

Include all relevant properties and enumerations listed below.

1. Project

Represents an academic or institutional research project.

Fields:
• id (String) - Primary key
• title (String) - Project title
• summary (String) - Short description of the project
• status (Enum as Status) - One of: PLANNING, ACTIVE, ON_HOLD, COMPLETED, ARCHIVED
• pi (User) - Linked Principal Investigator
• tags (String) - Comma-separated tags (e.g., “AI, environment”)
• startDate (LocalDate) - Project start date
• endDate (LocalDate) - Expected completion date
• createdAt (LocalDateTime) - Audit field
• updatedAt (LocalDateTime) - Audit field

2. Milestone

Represents key stages or deliverables of a research project.

Fields:
• id (String) - Primary key
• project (Project) - Associated project
• title (String) - Milestone title
• description (String) - Notes or task details
• dueDate (LocalDate) - Deadline date
• isCompleted (Boolean) - Completion flag
• createdBy (User) - User who created the milestone

3. Document
Represents research-related files or reference materials uploaded to the system.

Fields:
• id (String) - Primary key
• project (Project) - Linked project
• title (String) - File name or label
• description (String) - Notes or summary
• urlOrPath (String) - File URL or server path
• uploadedBy (User) - Linked user who uploaded the file
• uploadedAt (LocalDateTime) - Timestamp of upload

4. User
Represents any registered system user (Admin, Principal Investigator, or Research Member).

Fields:
• id (String) - Primary key
• username (String) - Unique username or email
• password (String) - Encrypted password
• fullName (String) - User's full name
• role (Enum as UserRole) - One of: ADMIN, PI, MEMBER, VIEWER
• createdAt (LocalDateTime) - Account creation timestamp

Authentication and Authorization
Implement secure, token-based authentication using JWT and Spring Security.

Requirements:
• Sign Up (Registration): Allow new users to register (default role: MEMBER).
• Sign In (Login): Authenticate users and issue JWT tokens.
• Authorization:
  - ADMIN: Full system access (manage users, projects, milestones, documents).
  - PI: Manage own projects and associated members.
  - MEMBER: Create and update milestones or upload documents.
  - VIEWER: Read-only access to public project data.

Security Features:
• Stateless JWT authentication
• Password hashing (BCrypt)
• Role-based endpoint authorization
• Custom exception handling for unauthorized access

Controller Layer

Organize endpoints following RESTful standards:

AuthenticationController:

- POST /api/auth/signup – Register new user
- POST /api/auth/login – Authenticate and issue JWT

ProjectController:

- GET /api/projects – List all projects
- GET /api/projects/{id} – View project details
- POST /api/projects – Create new project (Only for PI or Admin)
- PUT /api/projects/{id} – Update project details
- PATCH /api/projects/{id}/status – Update project status
- DELETE /api/projects/{id} – Delete project (Only for Admin)

MilestoneController:

- GET /api/projects/{id}/milestones – List milestones for a project
- POST /api/projects/{id}/milestones – Add milestone
- PUT /api/milestones/{id} – Update milestone
- DELETE /api/milestones/{id} – Delete milestone

DocumentController:

- GET /api/projects/{id}/documents – List project documents
- POST /api/projects/{id}/documents – Upload new document
- DELETE /api/documents/{id} – Delete document (Only for Admin or PI)

UserController:

- GET /api/users – List all users (Only for Admin)
- GET /api/users/{id} – View user profile
- DELETE /api/users/{id} – Delete user (Only for Admin)

Version Control (VCS)

- Use GitHub for repository management.
- Maintain clear commit messages.
- Include source code, SQL schema or JPA configuration, and a README.md file.
- Use GitHub and share the repository link when submitting the task.

Deliverables

1.Fully functional backend API built with Spring Boot and JWT.
2. Working authentication and authorization mechanisms.
3. MySQL database schema and data persistence.
4. Clean, well-documented code with comments.
5. GitHub repository link and README file.

Expected Outcome

Upon completing this coursework, students will be able to:

• Design and implement a multi-entity CRUD application with proper data relationships.
• Secure APIs using Spring Security and JWT.
• Apply role-based access control in a research collaboration context.
• Practice clean architecture and best coding practices.
• Demonstrate proficiency in using Spring Boot, JPA, and MySQL for enterprise-grade application development.