Project Requirement Document

Project Title: Java Web Application with Spring Boot

Overview

This project aims to give students hands-on experience with **Spring Boot** and the development of a **RESTful API**, focusing on back-end development and integration with front-end libraries or frameworks. Students will implement **authentication** (login and signup) and **CRUD operations** for managing data, building a complete web application. They are should integrate the API with any front-end framework (React, Angular, Vue, etc.) or build a mobile interface using React Native or similar.

The project is designed to solidify concepts of web services, Spring Boot, security, and database interaction while encouraging creativity in the choice of frontend technology.

Objectives

- **Apply intermediate to advanced Java programming concepts** using Spring Boot.
- Gain practical experience in **RESTful API design and development**.
- Implement **authentication and authorization** using modern security mechanisms.
- Design and implement **CRUD operations** for managing resources.
- Integrate the back-end API with any **front-end technology** (React, Vue, Angular, etc.).
- Optional: Explore integration with mobile platforms (React Native).

Project Scope

Students will build a **Spring Boot application** that serves as the back-end for a web or mobile application. The project must meet the following requirements:

1. **Authentication & Authorization:**
    - Implement a secure **login and signup** system.
    - Ensure **password encryption** using hashing algorithms (e.g., BCrypt).
    - Allow role-based access control (e.g., Admin, User).

2. **CRUD Operations:**
    - Create, Read, Update, Delete (CRUD) functionality for one or more entities (e.g., users, products, posts).
    - Use a relational or non Relational database (MySQL, PostgreSQL,Nosql, MongoDB etc.) for persistent storage.
    - Implement **data validation** and error handling.
3. **RESTful API:**
    - Build a RESTful API using **Spring Boot**.
    - Endpoints should be well-structured and follow RESTful principles.
    - Secure the API using **JWT (JSON Web Tokens)** or OAuth 2.0.
4. **Frontend Integration:**
    - Integrate the API with any front-end framework or library (React, Angular, Vue).
    - Provide basic UI/UX for login, signup, and CRUD operations.
    - Optional: Integration with a mobile interface (React Native).

Additional Project Options (Alternative to Spring Boot RESTful API)

Students may choose **one** of the following alternatives if they wish to explore other Java technologies:

1. **Desktop Application with JavaFX:**
    - Build a **JavaFX application** that includes login/signup functionality and CRUD operations.
    - Use **SQLite** , **MySQL** or NoSQL for data storage.
    - Implement **multi-threading** for background tasks (e.g., loading data, processing requests).
2. **Microservices with Spring Cloud:**
    - Implement a set of **microservices** using **Spring Cloud**.
    - Each microservice should handle different responsibilities (e.g., user management, product catalog, orders).
    - Use a **message queue** like RabbitMQ or Kafka to enable communication between services.
3. **WebSocket-based Chat Application:**
    - Build a **real-time chat application** using **WebSockets** and Spring Boot.

    - Implement user authentication for chat participants.
    - Use WebSocket connections to allow real-time messaging between users.
4. **E-commerce Platform:**
    - Develop a basic **e-commerce platform** with product listings, user authentication, and an order system.
    - Integrate payment gateways using Java libraries (mocking payments is acceptable).

Technologies to be Used

- **Java 11/17+**
- **Spring Boot** for back-end
- **Spring Security** for authentication
- **Hibernate/JPA** for database interaction
- **MySQL/PostgreSQL** for the database
- **JWT** or **OAuth 2.0** for securing API endpoints
- **Any front-end library/framework** (React, Vue, Angular, or React Native for mobile)

Deliverables

1. **Source Code:**
    - Complete source code for the back-end (Spring Boot) and front-end (if integrated).
2. **Technical Documentation:**
    - Description of the API endpoints and their functionality.
    - Instructions on how to set up and run the project locally (README file).
    - Documentation for front-end integration (if applicable).
3. **Database Schema:**
    - SQL script for setting up the database schema.
    - ER diagram showing the relationships between entities.
4. **Presentation:**
    - A brief presentation (10 minutes) demonstrating the application and its functionality.

- Discuss challenges faced and solutions implemented.

---

1. **Documentation and Presentation**
   - Write a clear and complete technical documentation.
   - Ability to explain the project during the presentation.

Resources

- Spring Boot Official Documentation: https://spring.io/projects/spring-boot
- Hibernate Documentation: https://hibernate.org/orm/documentation/5.4/
- JWT Java Library: https://github.com/jwtk/jjwt
- MySQL Database: https://www.mysql.com/
- Frontend Library/Frameworks: React, Angular, Vue (Optional)