Requirement Document for Task Management System

1. Introduction

This document outlines the requirements for a Task Management System. The system will allow users to register, log in, create tasks, view tasks, update tasks, delete tasks, and mark tasks as complete. Additionally, users will be able to log out of the system.

2. Project Overview

The Task Management System is designed to help users manage their tasks efficiently. Users can register and log in to the system, create new tasks, view their tasks, update task details, delete tasks, and mark tasks as complete. The system will also provide a logout functionality to ensure user sessions are securely terminated.

3. Technologies

- **Frontend**:
    - HTML, CSS, JavaScript
    - HTTP Client: Fetch API
- **Backend**:
    - Language: Java
    - Framework: Spring Boot
    - Database: MySQL
    - ORM: Spring Data JPA, Hibernate
    - Build Tool: Maven
    - API Documentation: Open API (Swagger)
- **Other Tools**:
    - Version Control: Git
    - Testing: Junit
    - Logging: Log4J

    - Developer tools: Spring Boot Dev tools, Lombok

4. Functional Requirements

4.1 User Registration

- **Description**: Users should be able to register by providing a username and password.
- **Inputs**: Username, Password, Confirm Password
- **Outputs**: Confirmation of successful registration or error message
- **Preconditions**: The user must not already be registered with the same username.
- **Postconditions**: A new user account is created and stored in the system.

4.2 User Login

- **Description**: Registered users should be able to log in by providing their username and password.
- **Inputs**: Username, Password
- **Outputs**: User session is initiated, and the user is redirected to the task management interface, or an error message is displayed.
- **Preconditions**: The user must be registered and provide correct login credentials.
- **Postconditions**: The user is logged in, and a session is created.

4.3 User Logout

- **Description**: Logged-in users should be able to log out of the system.
- **Inputs**: None
- **Outputs**: User session is terminated, and the user is redirected to the login page.
- **Preconditions**: The user must be logged in.
- **Postconditions**: The user is logged out, and the session is destroyed.

4.4 Task Creation

- **Description**: Logged-in users should be able to create new tasks by providing a title and description.
- **Inputs**: Task Title, Task Description

- **Outputs**: Confirmation of successful task creation or error message
- **Preconditions**: The user must be logged in.
- **Postconditions**: A new task is created and associated with the logged-in user.

4.5 Task Retrieval

- **Description**: Logged-in users should be able to view a list of their tasks.
- **Inputs**: None
- **Outputs**: List of tasks associated with the logged-in user
- **Preconditions**: The user must be logged in.
- **Postconditions**: The user can view their tasks.

4.6 Task Update

- **Description**: Logged-in users should be able to update the details of their tasks.
- **Inputs**: Task ID, Updated Task Title, Updated Task Description
- **Outputs**: Confirmation of successful task update or error message
- **Preconditions**: The user must be logged in and the task must exist.
- **Postconditions**: The task details are updated in the system.

4.7 Task Deletion

- **Description**: Logged-in users should be able to delete their tasks.
- **Inputs**: Task ID
- **Outputs**: Confirmation of successful task deletion or error message
- **Preconditions**: The user must be logged in and the task must exist.
- **Postconditions**: The task is deleted from the system.

4.8 Task Completion

- **Description**: Logged-in users should be able to mark their tasks as complete.
- **Inputs**: Task ID
- **Outputs**: Confirmation of successful task completion or error message
- **Preconditions**: The user must be logged in and the task must exist.

- **Postconditions**: The task status is updated to "Completed".

5. Non-Functional Requirements

- **Performance**: The system should respond to user actions within 2 seconds.
- **Security (Optional)** : User passwords must be stored securely using encryption
- **Usability**: The user interface should be intuitive and easy to navigate.
- **Scalability (Optional)** : The system should be able to handle up to 10,000 concurrent users.
- **Availability**: The system should have an uptime of 99.9%.

6. User Interface Requirements

- **Login Page**: A form for users to enter their username and password.
- **Registration Page**: A form for new users to register by providing a username and password.
- **Task Management Interface**: A dashboard where users can view, create, update, delete, and mark tasks as complete.
- **Logout Button**: A button for users to log out of the system

7. Data Requirements

- **User Data**: Username, Password (encrypted)
- **Task Data**: Task ID, Title, Description, Status, Added Date, Associated User

8. Assumptions

- Users will have a unique username.
- Users will remember their login credentials.
- The system will be accessed via a web browser.

9. Constraints

- The system must comply with data protection regulations.
- The system must be compatible with modern web browsers.

10. Dependencies

- The system depends on a backend server to handle user authentication and task management.
- The system depends on a database to store user and task data.

11. Glossary

- **User**: An individual who uses the Task Management System.
- **Task**: An item created by a user that includes a title, description, and status.
- **Session**: A period during which a user is logged into the system.

12. Test-Driven Development (TDD) High-Level Plan

Frontend Project Structure

- **login.html**: Contains the login form and a link to the registration page.
- **registration.html**: Contains the registration form and a link to the login page.
- **tasks.html**: Contains the task management form and a list to display tasks.
- **app.js**: Handles form submissions and dynamically updates the task list. It also fetches tasks from the backend.
- **styles.css**: Styles the application to make it visually appealing.

Backend Project Structure

- **src/main/java/com/example/taskmanagement/**
    - **controller/**
        - UserController.java: Controller for handling user-related requests.
        - TaskController.java: Controller for handling task-related requests.
    - **model/**
        - User.java: Model class for user entity.
        - Task.java: Model class for task entity.
    - **repository/**
        - UserRepository.java: Repository interface for user entity.
        - TaskRepository.java: Repository interface for task entity.
    - **service/**

        - UserService.java: Service class for user-related business logic.
        - TaskService.java: Service class for task-related business logic.
    - **TaskManagementApplication.java**: Main application class.
- **src/main/resources/**
    - application.properties: Configuration file for the application.

Backend Step-by-Step Solution

1. **Set Up Spring Boot Project**: Create a new Spring Boot project.
2. **Configure MySQL Database**: Set up MySQL database configuration.
3. **Create Entity Classes**: Define the User and Task Entities.
4. **Create Repository Interfaces**: Create JPA repositories for data access.
5. **Create Service Classes**: Implement business logic in service classes.
6. **Create Controller Classes**: Define REST endpoints in controller classes.
7. **Run the Application**: Test the application.

Step 1: Set Up Spring Boot Project

You can use Spring Initializr to set up your Spring Boot project. Include the following dependencies:

- Spring Web
- Spring Data JPA
- MySQL Driver
- Open API (Swagger)
- Spring Boot Dev tools

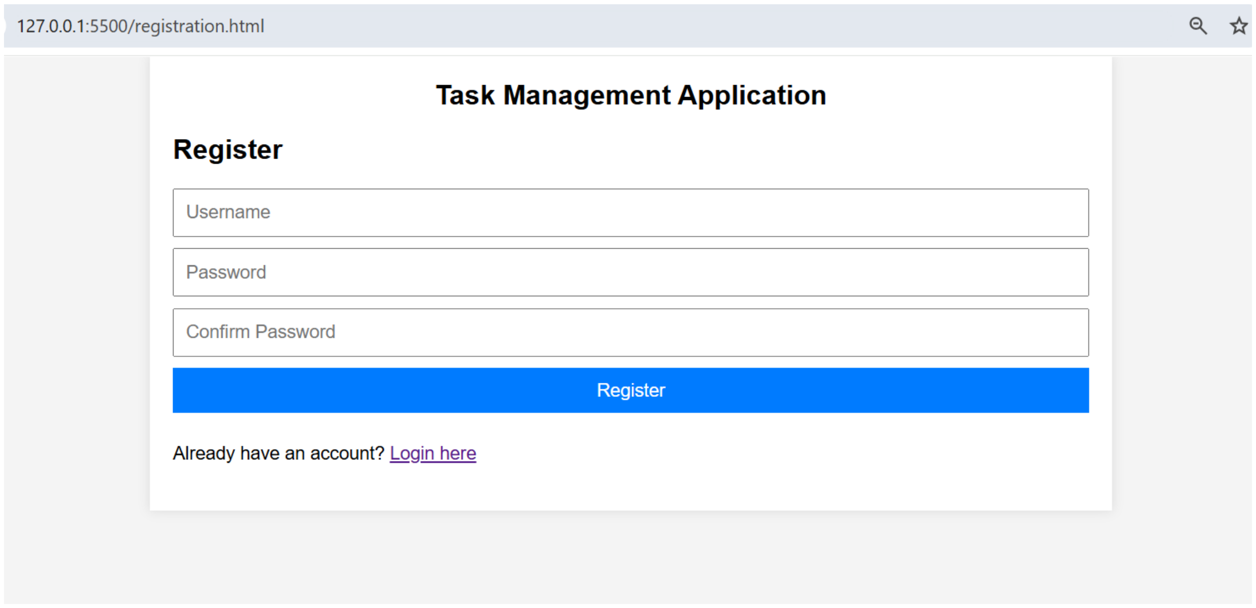
Step 2: Configure MySQL Database

Add the following configuration to your application.properties file:

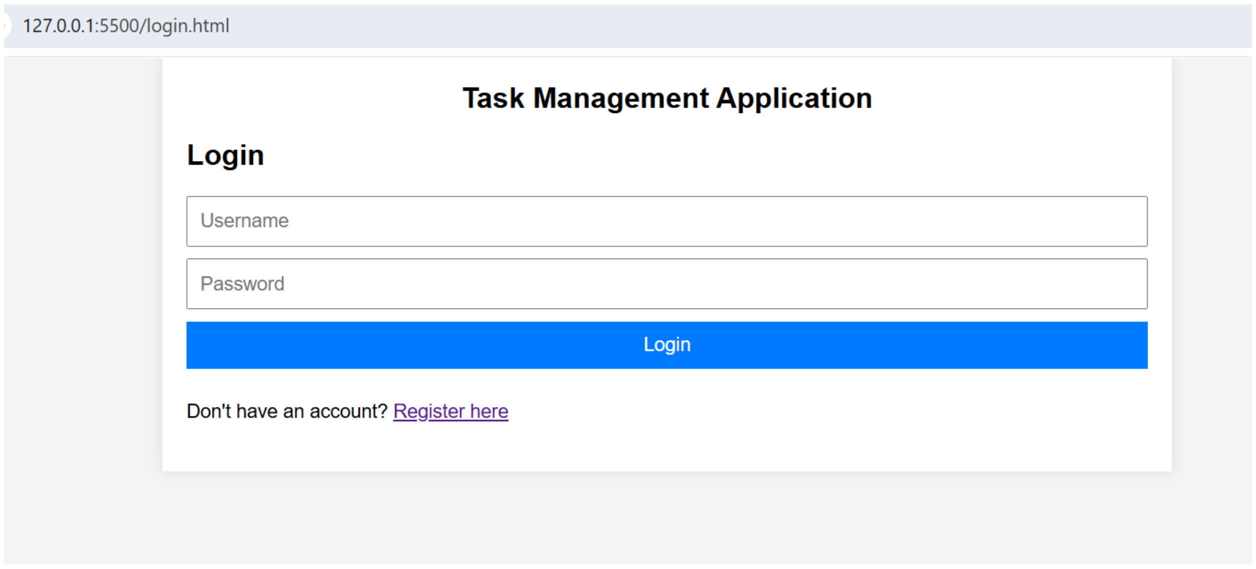
```
spring.datasource.url=jdbc:mysql://localhost:3306/task_management

spring.datasource.username=root
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

User Interface:

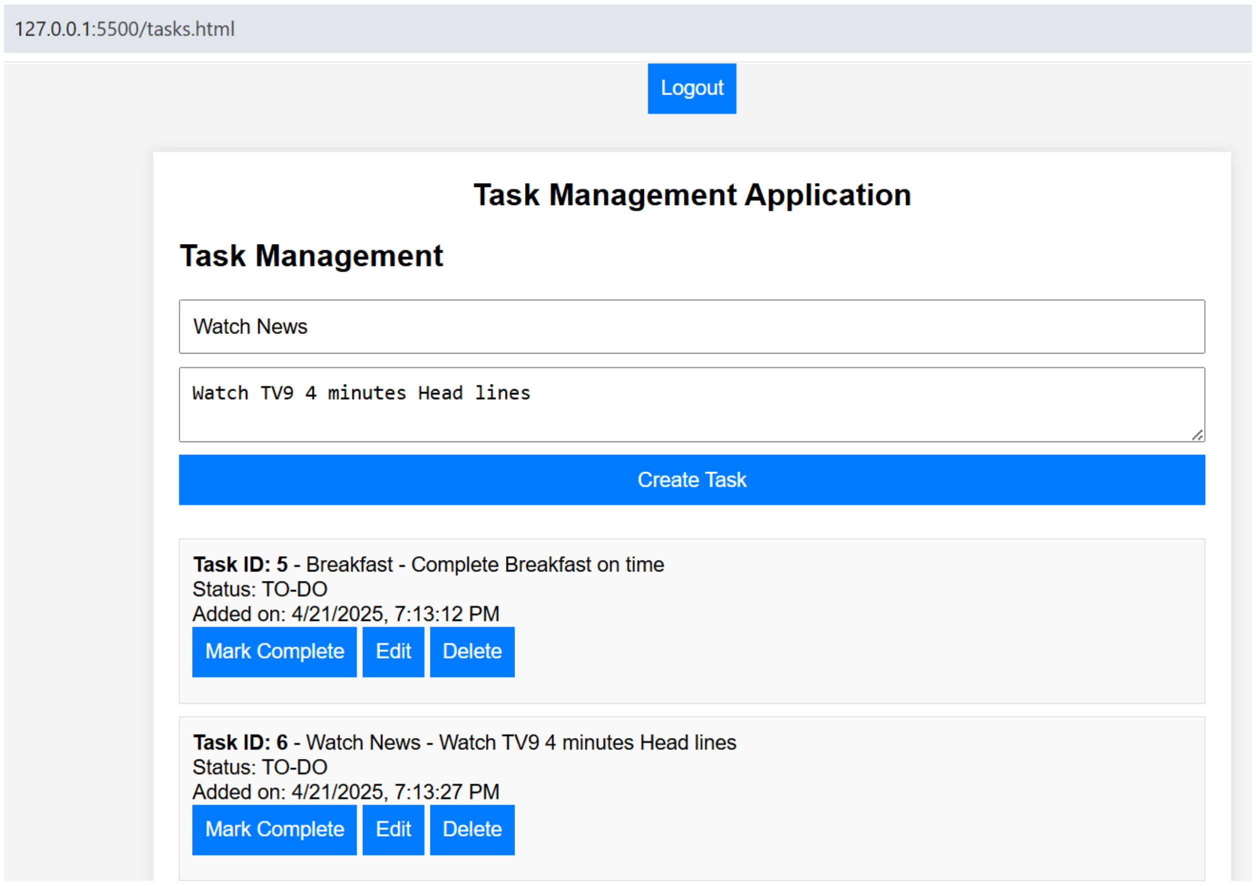
User Registration Screen:


127.0.0.1:5500/registration.html
Task Management Application
Register
Username
Password
Confirm Password
Register
Already have an account? Login here

Login Screen:


127.0.0.1:5500/login.html
Task Management Application
Login
Username
Password
Login
Don't have an account? Register here

Task Management Screen:



127.0.0.1:5500/tasks.html

Logout

Task Management Application

Task Management

Watch News

```
Watch TV9 4 minutes Head lines
```

Create Task

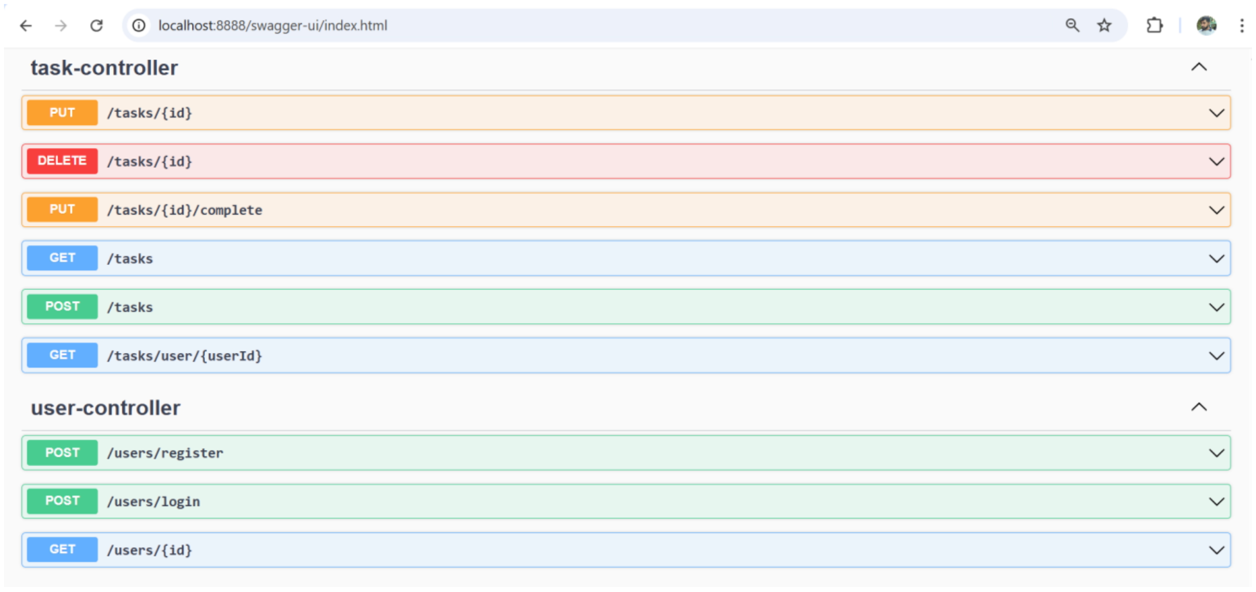
Task ID: 5 - Breakfast - Complete Breakfast on time
Status: TO-DO
Added on: 4/21/2025, 7:13:12 PM

Mark Complete | Edit | Delete

Task ID: 6 - Watch News - Watch TV9 4 minutes Head lines
Status: TO-DO
Added on: 4/21/2025, 7:13:27 PM

Mark Complete | Edit | Delete

Backend APIs:



localhost:8888/swagger-ui/index.html

task-controller

```
PUT     /tasks/{id}
DELETE  /tasks/{id}
PUT     /tasks/{id}/complete
GET     /tasks
POST    /tasks
GET     /tasks/user/{userId}
```

user-controller

```
POST    /users/register
POST    /users/login
GET     /users/{id}
```

Note: Feel free to implement additional features and enhance the functionality of the application.