•Software Design Document

1. Introduction

This document outlines the design of a DevOps application, which facilitates users to manage their projects. It also provides administrative functionalities to manage projects and tasks and assign them to specific developers. The application aims to provide a user-friendly interface for Team Leader to divide tasks over the registered developers.

2. System Overview

The system consists of two main components:

• User Interface: Provides an interface for users to interact with the application.

• Backend Server: Handles requests from the user interface, performs necessary operations, and communicates with the database.

3. Functional Requirements

3.1 User (Team Leader-Developer) Functionality

1. User Registration (POST): Users can register by providing necessary details.

2.User Login (POST): Registered users can log in to the application.

3. User Logout (POST): Users can log out of the application.

4. List Tasks (GET): Users can view the list of all tasks.

5. View comments (GET): Users can see all comments of all tasks.

3.2 Team Leader Functionality

1. Add project (POST): Team Leader can add new project to the database by providing its title.

2. Update project (PUT): Team Leader can update existing project title.

3. Delete project (DELETE): Team Leader can delete project from the database.

4. Add Tasks (POST): Team Leader can add new tasks to projects

5.assign tasks (POST): Team Leader can assign developers to specific tasks.

5. List of projects (GET): Team Leader can view the list of projects.

6. List of tasks (GET): Team Leader can view the list of all tasks.

7. List of users (GET): Team Leader can view the list of developers.

3.3 Developer Functionality

1. User Registration (POST): Users can register by providing necessary details.

2. User Login (POST): Registered users can log in to the application.

3. User Logout (POST): Users can log out of the application.

4. Update task status (PUT): Developer can update task status that is assigned to him.

5. accept or reject assigning to project (POST): Users can accept or reject assigning to project.

6. attach file (POST): Users can attach file when task status is done.

4. Architecture Overview

The application follows a client-server architecture:

- Client Side: Implemented using a web application.
- Server Side: Implemented using a RESTful API server.

4.1 Frontend Architecture

The front-end architecture includes components for user registration, login, main dashboard, tasks dashboard, projects list.

4.2 Backend Architecture

The backend architecture comprises the following components:

• Controller Layer: Handles incoming requests and delegates them to appropriate service classes.

• Service Layer: Contains business logic for processing requests and interacts with the data access layer.

• Data Access Layer: Communicates with the database to perform CRUD operations on projects and tasks data.

5. Database Schema

The database schema consists of tables to store information about users, movies, favorites, booking details, and booked movies.

5.1 User Table

- Table Name: **user**
- Columns:
  - id (Primary Key, Auto-generated): Unique identifier for each user.
  - email: Email address of the user.
  - password: Password of the user.
  - FirstName: First name of the user.
  - LastName: Last name of the user.
  - RoleID: Role of the user (e.g., TeamLeader, Developer).

5.2 Role Table

- Table Name: **Roles**
- Columns:
  - id (Primary Key, Auto-generated): Unique identifier for each role.
  - Role(foreign Key >>Table: Role): Role type for user.

5.3 Projects table

- Table Name: **Projects**
- Columns:
    - id (Primary Key, Auto-generated): Unique identifier for each project.
    - title: project’s title.

5.4 Tasks table

- Table Name: **Tasks**
- Columns:
    - id (Primary Key, Auto-generated): Unique identifier for each task.
    - title: Task’s title.
    - Description: Task’s description.
    - status: Task’s status (e.g., Backlog, Running, Done).
    - ProjectID(foreign Key >>Table: Projects): project’s id that the task is referred to.

5.5 Project Status table

- Table Name: **ProjectStatus**
- Columns:
    - id (Primary Key, Auto-generated): Unique identifier for each project.
    - id (Primary Key, Auto-generated): Unique identifier for each user.
    - status: project’s status.

5.6 Comments table

• Table Name: **Comments**

• Columns:

  • id (Primary Key, Auto-generated): Unique identifier for each comment.

  • UserId (foreign Key >>Table: User): User's id that makes the comment.

  • TaskID (foreign Key >>Table: Task): Unique identifier for each user.

  • comment : comment body

5.6 Assigned Tasks table

• Table Name: **AssignedTasks**

• Columns:

  • UserId (Primary Key, Auto-generated): Unique identifier for each User.

  • TaskId (foreign Key >>Table: User): Task's id that that is to be assigned.

  • attachment: attachment add by the assigned developer when the task status is done.

6. Technologies Used

• Backend Framework: Spring boot

• Database: MySQL

Authentication:

1- User Authentication Process:

• Describe the process of how users are authenticated within the system.

• Mention the use of role-based authentication where users provide their credentials (username and password) through a login form.

• Explain how the authentication process is initiated when a user attempts to access a secured resource or endpoint.

2- Custom Success Handler:

• Explain the purpose of the custom success handler (CustomSuccessHandler) used in the authentication configuration.

• Mention that the success handler is responsible for redirecting authenticated users to the appropriate page based on their role (e.g., TeamLeader-page or Developer-page).

3- User Details Service:

• Highlight the usage of a custom user details service (CustomUserDetailsService) for loading user-specific data during authentication.

• Mention that the user details service retrieves user details (e.g., username, password, RoleId) from the database.

4- Logout Configuration:

• Describe the configuration for handling user logout.

• Mention the invalidation of the HTTP session, clearing of authentication, and redirection to the login page after logout.

7. Conclusion

This software design document provides a comprehensive overview of the Project management application's design, including its functionality, architecture, database schema, and technologies used. It serves as a guideline for development and maintenance activities throughout the project lifecycle.