Final Project Report: step-by-step-Todo-app

Prepared by: Bytexl

Educator: Zameer Hussain

# Project Title: step-by-step-Todo-app with Springboot and React

## Executive Summary

This report presents the development of an step-by-step-Todo-app project built using the React Springboot and mySql. The project provides a interface for data upload and real-time visualization, allowing users to manage, display, and interact with data efficiently. Key features of the app include CRUD (Create, Read, Update, Delete) operations, real-time data visualization. This project showcases full-stack development skills and prepares individuals for real-world scenarios in software development.

## 1. Tools and Technologies Utilized

- Frontend: React.js and Material Ui

- Backend: SpringBoot

- Database: MySql

- API Testing: Postman

## 2. Project Overview

This project aims to create a simple Todo application that implements CRUD (Create, Read, Update, Delete) operations. The app will be developed using React for the frontend and Spring Boot for the backend. The project will focus on enabling users to manage their tasks effectively with a simple and user-friendly interface.

## 3. System Requirements

### Software Requirements

- Node.js (with npm)

- React.js (latest version)

- SpringBoot

- MySql Workbench

Compatible browser (Google Chrome or Mozilla Firefox)

### Hardware Requirements

- Minimum RAM: 4GB

- Processor: Dual-core or better

- Operating System: 64-bit

## 4. Functional Requirements

The project must fulfill the following core functions:

- CRUD Operations: Enable the creation, retrieval, updating, and deletion of records.

- Allow users to add, edit, and delete events through an interactive calendar.

## 5. Inputs and Outputs

### Inputs

- User data entered react-web application or postman

### Outputs

- show data according to state.  
- Real-time updates of task.

## 7. System Subcomponents

The project includes the following subsystems:

**Routing:**

* Routing is responsible for managing navigation within the app (e.g., routing between different pages or views, managing URLs, etc.). Popular tools include React Router.

**Data Fetching and API Integration:**

* It is responsible for fetching data from the backend via APIs and displaying that data in the UI. We have use **Axios** .

**API Layer/Service Layer:**

* This layer expose the backend functionality via **APIs** such as RESTful. This is where data processing occurs and is typically where business logic resides.

## 8. Potential for Other Applications

This step-by-step-Todo-app is build to learn the basic of rest api so the students can easily learn how it works and its implementations has a wide range of applications beyond its current implementation.

## 9. Test Case Design

|  |  |  |
| --- | --- | --- |
| Test Case ID | Function | Expected Outcome |
| TC001 | Create a new record | record has been created |
| TC002 | Update a record | record has been updated |
| TC003 | Delete a record | record has been deleted |
| TC004 | Fetch all record | all the record is received |
| TC005 | Mark a record as complete | state of task has been updated |

## 10. Future Enhancements

To improve the system, the following features could be considered for future versions:

- integrate user Login:

- User Based Todo: Implement different levels of user access and permissions to enhance security and flexibility.

- Advanced Filtering: Develop more sophisticated data filtering options, allowing users to filter by multiple parameters or ranges.

## 11. References

- SpringBoot official Documentation  
- React.js and Node.js Official Documentation  
- Nimbus Platform Usage Guidelines

## 12. Project Reflection

### Technical Challenges Encountered

- Data Synchronization: Ensuring real-time updates across the application without performance issues.  
- Error Handling: Developing comprehensive error handling mechanisms across both the frontend and backend.  
- Data Validation: Managing consistency and accuracy of data across forms and APIs.

### Software Engineering Insights

Applying principles of RESTful API design allowed for seamless data exchange between the client and server, simplifying maintenance.

### Personal Development

This project enhanced my skills in full-stack development, particularly in understanding how rest api works with or without database integration.  
I gained practical experience in handling complex, scalable features, from CRUD operations .