Routing and SPA

IIHT

Time To Complete: 10 to 12 hr

CONTENTS

1 Project Abstract

Modern web applications often follow the Single Page Application (SPA) model, allowing users to navigate between different views without full page reloads. This project introduces **React Router**—a library for handling routing in React applications. You will build a personal task management app that includes multiple pages (Home, Task List, and About) and allows smooth navigation using client-side routing. This project helps to understand the fundamentals of building multi-page experiences in a single-page architecture using declarative routing.

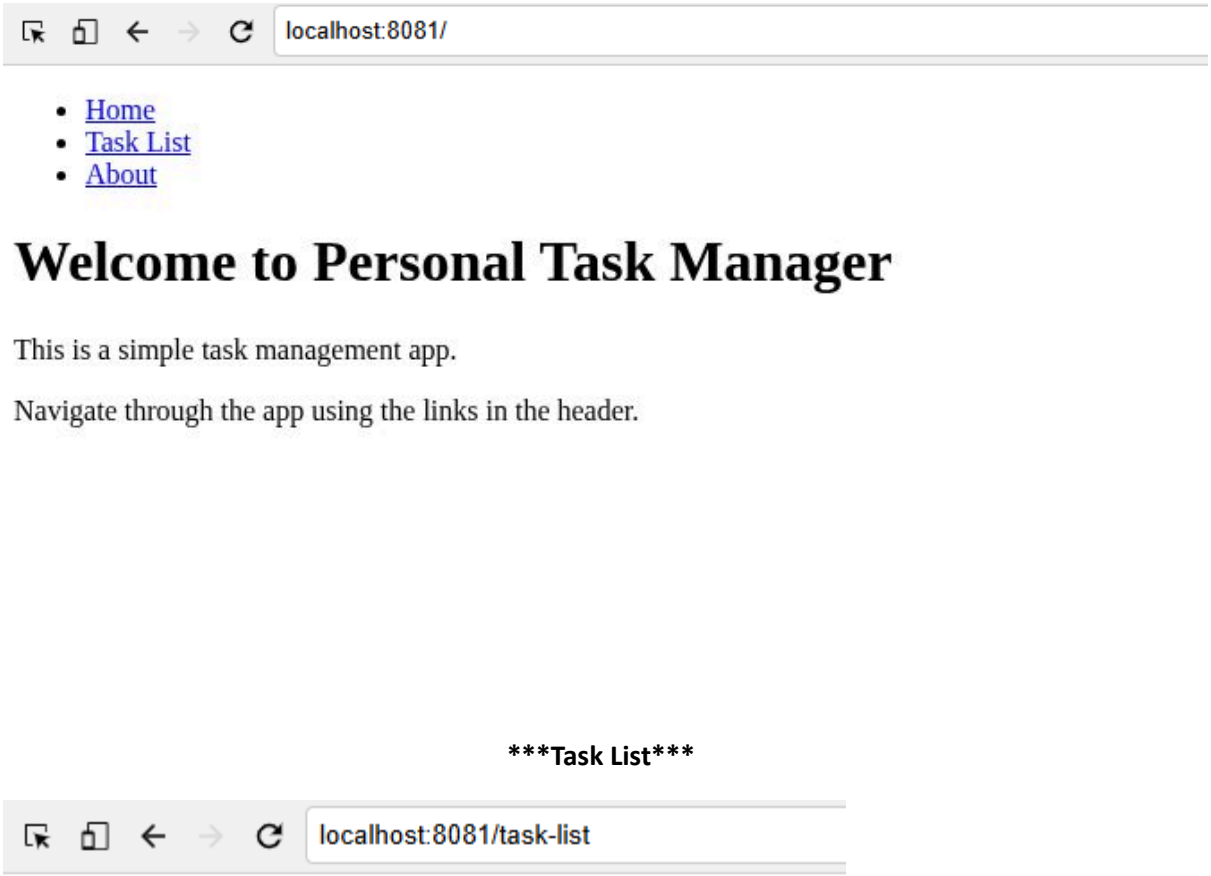
2 Problem Statement

You are required to create a React Single Page Application (SPA) with three main views: **Home**, **Task List**, and **About**. The application should use `react-router-dom` for client-side routing, allowing users to navigate without reloading the page. A shared `Header` component will provide navigation links. The `App` component will manage route definitions using `Route` and `Switch`. Each view is a separate component displaying its respective content. The goal is to demonstrate how to implement navigation, route handling, and modular page structuring in a React SPA.

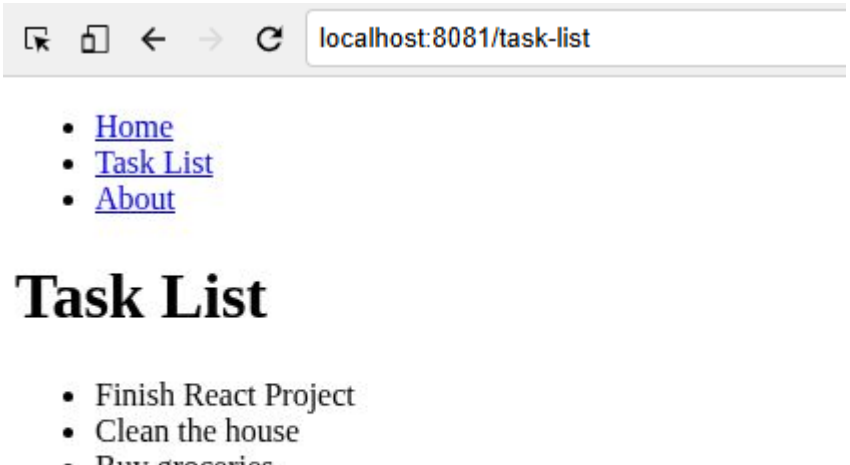
3 Proposed Routing and SPA Application Wireframe

UI needs improvisation and modification as per given use case and to make test cases passed.

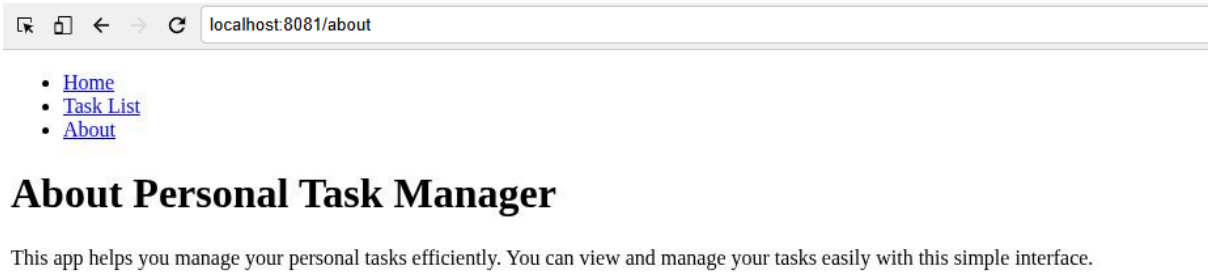
3.1 ScreenShots



Task List



About



4 BUSINESS-REQUIREMENT:

As an application developer, develop the Routing and SPA (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Welcome Page	As a user I should be able to visit the welcome page as the default page. Acceptance criteria: 1. Display a consistent **header** across all pages with navigation links. 2. Route definitions: ● `/` → Home page ● `/task-list` → Task List page ● `/about` → About page 3. Page content should change **without refreshing the browser**. 4. Clicking each link should update the view dynamically using React Router. **Component Overview:** **AppComponent:** 1. Wraps the entire application layout. 2. Uses `<Switch>` and `<Route>` from `react-router-dom` to handle routing.

		3. Defines three routes: ● `/` renders `Home` ● `/task-list` renders `TaskList` ● `/about` renders `About` 4. Renders the `Header` component above the routed content. **Header Component:** 1. Renders navigation using `<Link>` components: ● **Home** ● **Task List** ● **About** 2. Links update the route **without reloading the page**. **HTML Structure & Navigation Flow:** 1. **Header:** ● Use a `<nav>` element containing a `<ul>` list. ● Each `<li>` should include a navigation link using the `<Link>` component. ● The links should point to: ➔ `/` with the label **"Home"** ➔ `/task-list` with the label **"Task List"** ➔ `/about` with the label **"About"** 2. **Main Page Container (`App.js`):** ● Use a `<main>` tag to wrap the content that changes based on the route. ● Inside this, define each `<Route>` with a `path` and `component` using a `<Switch>` statement: ➔ Route `/` should display the **Home** page. ➔ Route `/task-list` should display the **Task List** page. ➔ Route `/about` should display the **About** page. 3. **Pages:** ● **Home Page:** ➔ Use a `<h1>` tag with the heading: **Welcome to Personal Task Manager**

		➔ Add a `<p>` paragraph with the description: **This is a simple task management app.** ➔ Add another `<p>` paragraph with navigation info: **Navigate through the app using the links in the header.** ● **Task List Page:** ➔ Use a `<h1>` tag with the heading: **"Task List"** ➔ Add a `<ul>` element to display a list of tasks. ➢ Each `<li>` inside should contain a task name like: ● **Finish React Project** ● **Clean the house** ● **Buy groceries** ● **About Page:** ➔ Use a `<h1>` tag with the heading: **About Personal Task Manager** ➔ Add a `<p>` paragraph with the description: **This app helps you manage your personal tasks efficiently.** ➔ Add another `<p>` paragraph with additional info: **You can view and manage your tasks easily with this simple interface.** **Dynamic Routing Behavior:** 1. Clicking on any navigation link updates the route using **React Router's SPA mechanism**. 2. Page does **not reload** — content is dynamically swapped. 3. The route updates the **URL bar**, and the corresponding component renders instantly. **** Kindly refer to the screenshots for any clarifications. ****

5 CONSTRAINTS

1. You should be able to press the "TAB" key and "SHIFT + TAB" to navigate from top field to bottom field and vice-versa.

6 MANDATORY ASSESSMENT GUIDELINES

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to
   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. This editor Auto Saves the code.
4. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.
   Note: The application will not run in the local browser
7. You can follow series of command to setup React environment once you are in your project-name folder:
   a. npm install -> Will install all dependencies -> takes 10 to 15 min.
   b. npm run start -> To compile and deploy the project in browser. You can press the <Ctrl> key while clicking on localhost:4200 to open the project in the browser -> takes 2 to 3 min.
   c. npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.
8. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.