**COLLEGE CODE : 9623**

**COLLEGE NAME:** Amrita College Of Engineering And

Technology, Erachakulam

**DEPARTMENT :** COMPUTER SCIENCE ENGINEERING

**STUDENT NM- ID : aut**962323104119

**ROLL NO :** 962323104119

**DATE :** 27/10/2025

**C**ompleted **T**he **P**roject **N**amed **A**s **P**hase**-5 T**echnology **(**PROJECT DEMONSTRATION AND DOCUMENTATION for Single Page Application)

**PROJECT NAME : SINGLE PAGE APPLICATION**

**S**ubmitted **B**y**,**

**NAME:** VISHAL.V (962323104119)

MOBILE NO: +91 8489633498

**Project Overview:**

A Single Page Application (SPA) is a **web app that loads a single HTML page** and **dynamically updates** content as the user interacts with it.  
This project demonstrates how modern web technologies like **HTML, CSS, JavaScript**, and frameworks such as **React or Angular** can be used to create responsive and efficient web applications.

The app will allow users to perform **CRUD (Create, Read, Update, Delete)** operations, manage data efficiently, and navigate between sections without a full page refresh.

**Key points:**

* SPA loads only **one main HTML page**.
* **Content updates dynamically** without reloading the page.
* Provides a **fast and smooth user experience**.
* Uses **JavaScript frameworks** like React, Angular, or Vue.
* Communicates with the server using **AJAX or Fetch API**.
* Uses **client-side routing** for navigation between pages.

**Essential Topics Covered:**

* API integration (REST or GraphQL),
* Component-based architecture,
* Data handling and storage methods,
* Responsive web design,
* Performance optimization,
* Testing and debugging,
* Version control and deployment and etc.

**1. Project Setup:**

* Create project using React or Angular CLI
* Install dependencies: react-router-dom, axios
* Set up folder structure: components/, pages/, App.js
* Create reusable components (Header, Footer, PrivateRoute)
* Configure routes for Home, Login, Dashboard
* Implement login logic with local storage
* Add form validation for user inputs

**2. Components Used:**

* Header, Footer
* Home, Login, Dashboard pages
* PrivateRoute for login protection
* Form components for input and validation
* Loader/Alert components (optional)

### **3.Routing :**

* SPA routing controls navigation **without reloading the page**.
* Uses **client-side routing** to update views dynamically.
* Common libraries: **React Router, Angular Router, Vue Router**.
* Supports **protected routes** for logged-in users.
* Benefits: **fast navigation, seamless UX, and state preservation**.

### **4.Login Protection Logic :**

* Protects certain pages so only **authenticated users** can access them.
* On login, store status in **local storage, session, or state**.
* Use a **PrivateRoute** component to check login before rendering.
* If **not logged in → redirect to login page**.
* Ensures **secure access** and improves **user experience**.

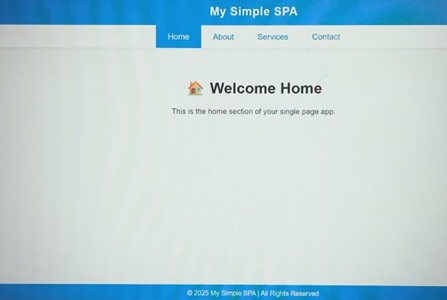
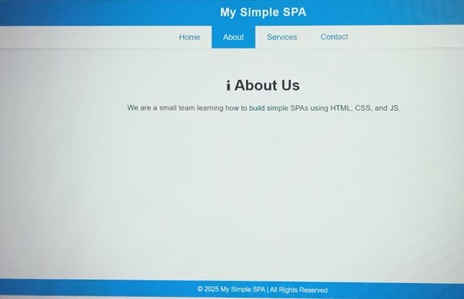
**5. Testing and Validation:**

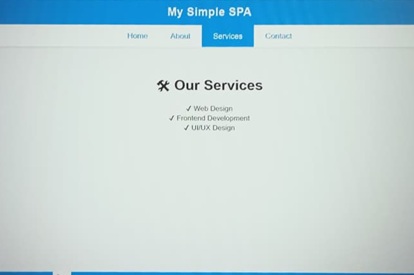
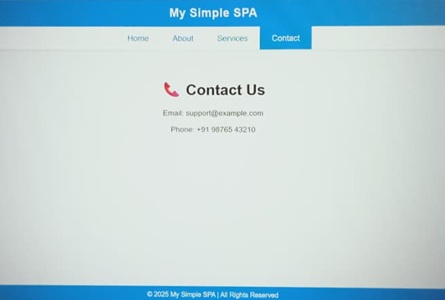
* Validated login form for correct input
* Protected routes redirect unauthorized users to login
* Components render correctly on navigation
* Application tested for responsiveness

**OUTCOME:**

* **Secure access** to protected pages via login protection.
* **Reusable and modular components** for easier maintenance.
* Improved **user experience, performance, and responsiveness** across devices.

**SCREENSHOTS:**

**1.] 2.]**

**3.] 4.]**

**CHALLENGES AND SOLUTION:**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| **Initial Load Time** | Use **code splitting** and **lazy loading** |
| |  | | --- | | **SEO Issues** | | |  | | --- | | **Use server-side rendering (SSR) or pre-rendering** | |
| |  | | --- | | **Unauthorized Access** | | |  | | --- | | **Implement login protection / PrivateRoute** | |
| |  | | --- | | **State Management** | | |  | | --- | | **Use Redux, Context API, or NgRx** | |  | |

**5. GitHub Link:**

### **https://github.com/hhayahaya655-ui/single-page-application.git**

### **Conclusion:**

The SPA project demonstrates **efficient navigation, secure access, and dynamic content updates**. It showcases modern web development practices using **component-based architecture, routing, login protection, and validation**.