COLLEGE CODE :9111

COLLEGE NAME :SRM MCET

DEPARTMENT :CSE

STUDENTNMROLLNO: 57F1376148A7CD9AC484D1EC1AE55C09

DATE :29/09/2025

Completed the project named as

Phase__4 TECHNOLOGY PROJECT

NAME :SINGLE PAGE APPILICATION

SUBMITTED BY,

NAME:HARSHINI .S
MOBILENO:6381280399

IBM-FE –Single Page Application

Phase 4- Enhancements & Deployment

🎬 **Focus:** Developing a modern web application using front-end technologies (like React, Angular, or Vue.js), likely incorporating **IBM's design systems** (e.g., **Carbon Design System**) for consistent UI/UX.

🎬 **Key-Points:** The entire application loads a single HTML page and dynamically updates content using JavaScript, providing a fast, fluid, and **desktop-like user experience**.

Additional Features

- **Focus:** Implementing new functionalities beyond the application's **Minimum Viable Product (MVP)** or initial scope.
- **Key-Points:** This phase involves requirements gathering, design, and development of new components, business logic, or third-party integrations (e.g., adding user authentication, a complex data visualization dashboard, or a payment gateway).

3. UI/UX Improvements

- **Focus:** Enhancing the **User Interface (UI)** aesthetic and the **User Experience (UX)** flow, usability, and accessibility.
- **Key-Points:** Activities include **A/B testing**, conducting user interviews, refining visual design (typography, color palette), improving **responsiveness** across different devices, and ensuring compliance with **accessibility standards (WCAG)**.

4. API Enhancements

- **Focus:** Modifying, optimizing, or expanding the **Backend Application Programming Interfaces (APIs)** that the front-end SPA consumes.
- **Key-Points:** This often involves improving **API response times**, adding new endpoints to support front-end features, updating data models, ensuring **data security**, and implementing features like **caching** or **rate limiting**.

5. Performance & Security Checks

- **Focus:** Auditing the application's speed, efficiency, stability, and protection against threats.
- **Key-Points - Performance:** Use tools like **Lighthouse** or WebPageTest to measure metrics (e.g., **First Contentful Paint, Time to Interactive**),

optimize asset loading (lazy loading, compression), and minimize bundle size.

- **Key-Points - Security:** Conduct **vulnerability scanning**, address common web vulnerabilities (**OWASP Top 10**), ensure secure data transmission (**HTTPS**), and implement proper **CORS** and **authentication/authorization** mechanisms.

6. Testing Of Enhancements

- **Focus:** Verifying that all new features, UI/UX changes, and API enhancements function correctly, meet requirements, and haven't introduced regressions.
- **Key-Points:** Includes different testing types: **Unit Tests** (for small code units), **Integration Tests** (for system components working together), **End-to-End (E2E) Tests** (simulating user flows), and **User Acceptance Testing (UAT)**.

7. Deployment (netlify, Vercel, or cloud Platform)

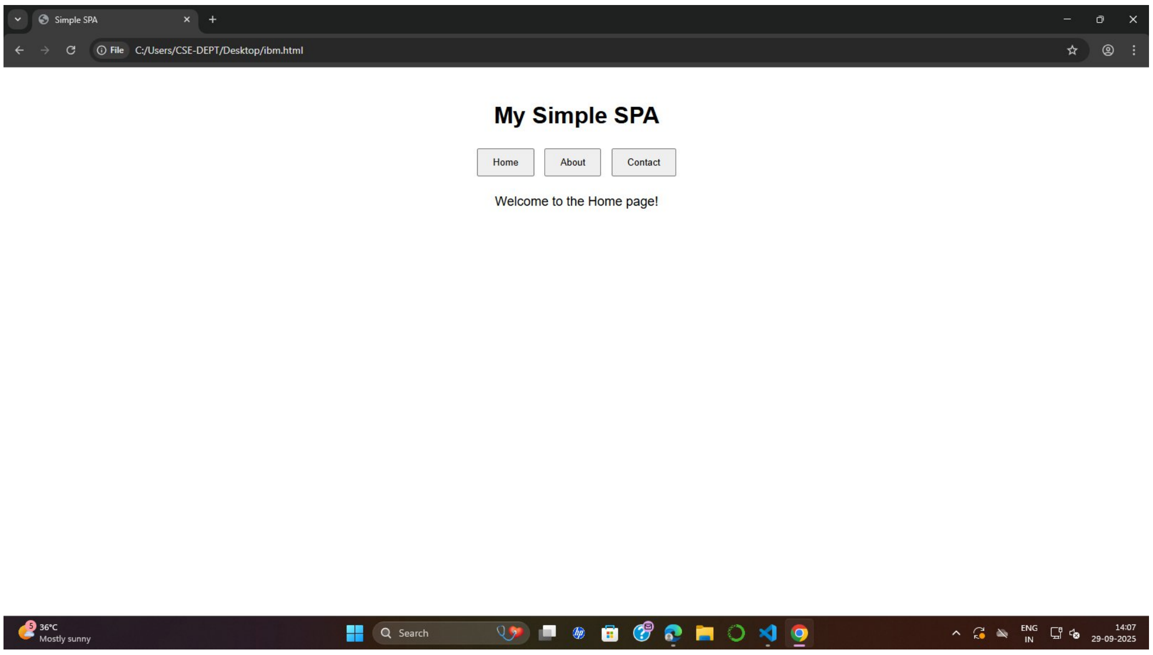
- **Focus:** Making the final, tested application available to end-users on a reliable hosting service.
- **Key-Points:** This involves **Continuous Integration/Continuous Deployment (CI/CD)** setup. Platforms like **Netlify** or **Vercel** are popular for static SPAs due to their ease of use, global CDN, and automatic build processes, while a more robust **cloud platform** (like IBM Cloud, AWS, Azure, or GCP) might be used for the backend APIs.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Simple SPA</title>
   <style>
      body { font-family: Arial, sans-serif; text-align: center; padding: 20px; }
      nav button { margin: 5px; padding: 10px 20px; }
      #content { margin-top: 20px; font-size: 18px; }
   </style>
</head>
<body>
   <h1>My Simple SPA</h1>
   <nav>
      <button onclick="showPage('home')">Home</button>
      <button onclick="showPage('about')">About</button>
```

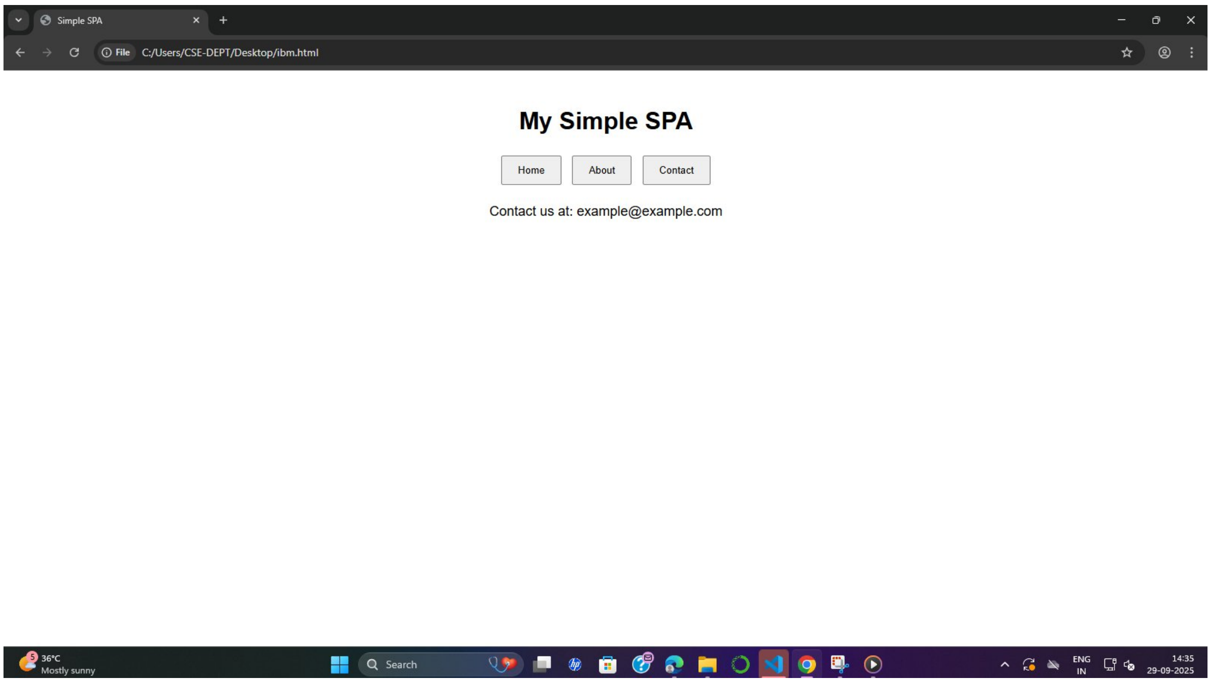
```
        <button onclick="showPage('contact')">Contact</button>
    </nav>
    <div id="content">Welcome to the Home page!</div>

    <script>
        function showPage(page) {
            const content = document.getElementById('content');
            if(page === 'home') {
                content.innerHTML = 'Welcome to the Home page!';
            } else if(page === 'about') {
                content.innerHTML = 'This is a simple SPA example created using
HTML, CSS, and JS.';
            } else if(page === 'contact') {
                content.innerHTML = 'Contact us at: example@example.com';
            }
        }
    </script>
</body>
</html>
```

OUTPUT:



My Simple SPA

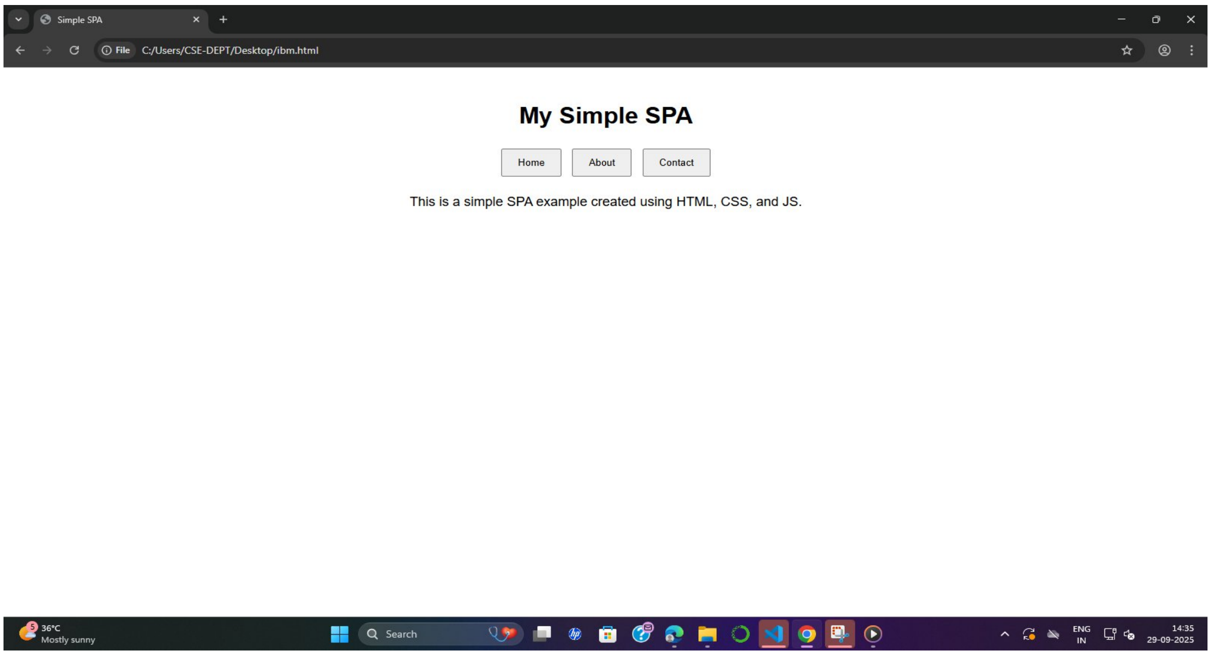
Welcome to the Home page!


Simple SPA

C:/Users/CSE-DEPT/Desktop/ibm.html

My Simple SPA

Contact us at: example@example.com


Simple SPA

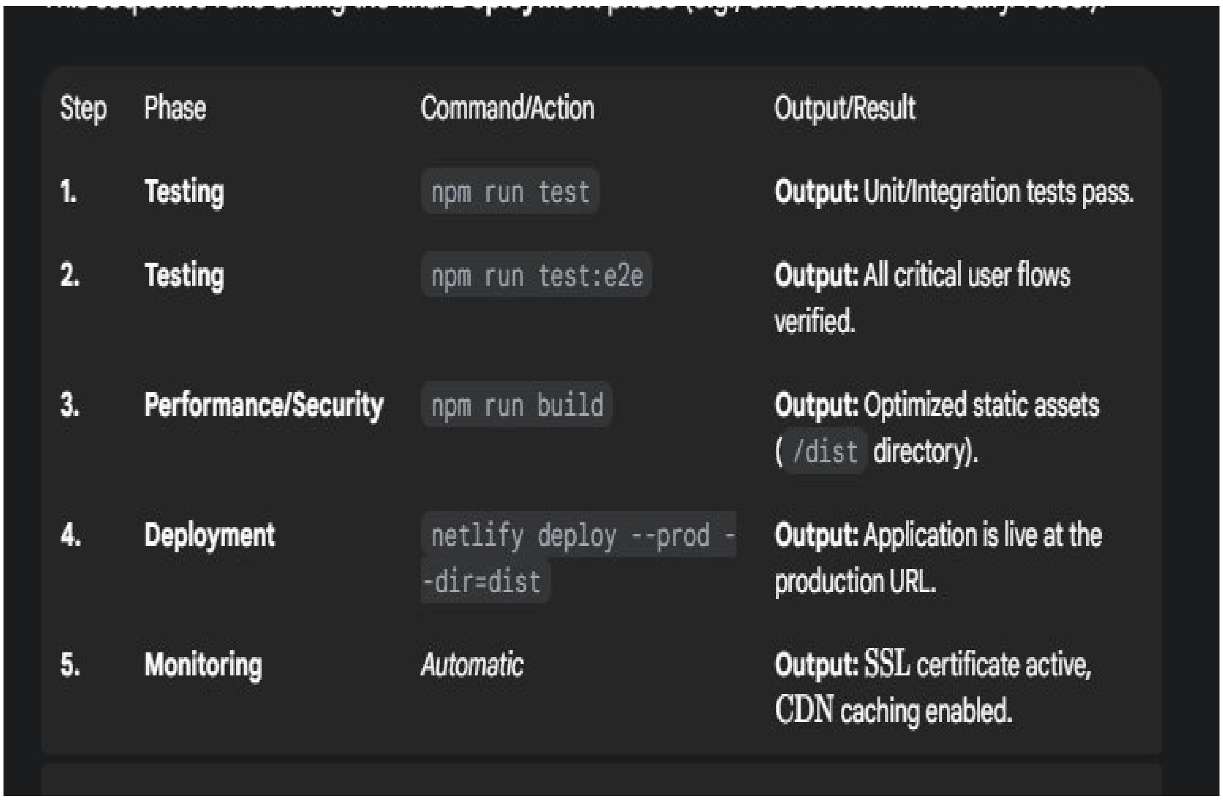
C:/Users/CSE-DEPT/Desktop/ibm.html

My Simple SPA

This is a simple SPA example created using HTML, CSS, and JS.

Deployment Command Sequence (The Execution Program)

This sequence runs during the final **Deployment** phase (e.g., on a service like Netlify / Vercel).



[illegible]

Step	Phase	Command/Action	Output/Result
1.	**Testing**	`npm run test`	**Output:** Unit/Integration tests pass.
2.	**Testing**	`npm run test:e2e`	**Output:** All critical user flows verified.
3.	**Performance/Security**	`npm run build`	**Output:** Optimized static assets (`/dist` directory).
4.	**Deployment**	`netlify deploy --prod --dir=dist`	**Output:** Application is live at the production URL.
5.	**Monitoring**	*Automatic*	**Output:** SSL certificate active, CDN caching enabled.

DRIVE LINK:

https://drive.google.com/file/d/1Er-6ozdGLcYJQPpneEjHJgnR0cLewMKd/view?usp=drivesdk

https://drive.google.com/file/d/1ak1MbTV2PbmnyWcvB2Eveh-Y8neQe09x/view?usp=drivesdk

GITHUB LINK:

https://github.com/samiksharajesh5-stack/Single-page-application-phase-1.git

https://github.com/samiksharajesh5-stack/single-pageapplication-phase-2.git

https://github.com/samiksharajesh5-stack/Single-phase-application-phase-3.git

Conclusion:

The development and deployment of the IBM-FE-Single Page Application (SPA) is a structured process that emphasizes speed, quality, and user-centric design, integrated with robust back-end support.

The core conclusion is that successful completion of this project is achieved through the **iterative and collaborative fulfillment of several critical requirements**:

1. **Modern Architecture:** The project establishes a flexible, dynamic **IBM-FE-SPA** foundation (using technologies like React/Carbon Design System) capable of delivering a fast, desktop-like user experience.
2. **Feature and Quality Integration:** The **Additional Features** and **API Enhancements** phases ensure the application meets evolving business needs with responsive and efficient data handling.
3. **User-Centricity:** Dedicated focus on **UI/UX Improvements** guarantees the application is not only functional but also highly usable, accessible (WCAG compliant), and visually aligned with IBM's design standards.
4. **Assurance and Stability:** Rigorous **Performance & Security Checks** combined with comprehensive **Testing Of Enhancements** (Unit, E2E, UAT) ensure the final product is secure, fast, and free of critical regressions before reaching users.
5. **Efficient Delivery:** The final **Deployment** via modern platforms (Netlify, Vercel, or Cloud) leverages **CI/CD** pipelines for rapid, automated, and reliable releases, resulting in a live application that is consistently available and high-performing.