****

**Assignment No : 4**

## 4. Create a SPA with React.js that includes multiple components, state management, and routes. Integrate with a public API to fetch and display data.

**Objective:**

* To create a Single Page Application (SPA) that interacts with the unofficial ISRO API.
* To implement multiple endpoints: Spacecrafts, Launchers, Customer Satellites, and Centres.
* To demonstrate the use of React for building a responsive and interactive web application.
* To practice working with external APIs and managing state in a React application.

**Theory:**

1. **React and Component-Based Architecture:** The project utilizes React's component-based structure, promoting reusability and separation of concerns. Each API endpoint is represented by its own component.
2. **State Management:** The application uses React's useState hook for local state management, demonstrating how to handle and update data within functional components.
3. **Side Effects and Data Fetching:** React's useEffect hook is employed to manage side effects, particularly for fetching data from the ISRO API when components mount.
4. **Routing in Single Page Applications:** React Router is used to implement client-side routing, allowing for navigation between different views without full page reloads.
5. **API Integration:** The project showcases how to integrate external APIs into a React application, handling asynchronous operations and managing loading states.
6. **Responsive Design:** CSS is utilized to create a responsive layout that adapts to different screen sizes, enhancing user experience across devices.

**Implementation Details:**

1. **Component Structure:** Separate components are created for each API endpoint (Spacecrafts, Launchers, CustomerSatellites, Centres), promoting a modular and maintainable codebase.
2. **Data Fetching:** Each component implements its own data fetching logic using the useEffect hook, demonstrating how to handle API calls and update component state.
3. **Error Handling:** The application includes basic error handling to manage potential API failures or network issues, improving the robustness of the application.
4. **User Interface:** A clean and intuitive UI is implemented using vanilla CSS, showcasing how to create appealing layouts without relying on external UI libraries.
5. **Navigation:** A navigation bar is implemented to allow users to easily switch between different data views, demonstrating the use of React Router for SPA navigation.

**Conclusion:**

1. **React's Efficiency:** The project demonstrates React's efficiency in building interactive UIs, particularly in how it manages updates and re-renders only the necessary components.
2. **API Integration:** The application showcases a practical approach to integrating external APIs in a React application, handling asynchronous operations, and managing application state based on API responses.
3. **Scalability:** The component-based structure and use of React Router provides a scalable foundation, making it easy to add new features or additional API endpoints in the future.
4. **Performance Considerations:** While React's virtual DOM provides efficient updates, care must be taken when fetching and rendering large datasets to ensure optimal performance.
5. **Learning Outcomes:** This project provides practical experience in several key areas of modern web development, including working with APIs, state management in React, and implementing responsive designs.
6. **Future Enhancements:** Potential improvements could include implementing more advanced state management solutions (e.g., Context API or Redux) for larger-scale applications, adding data caching to reduce API calls, or implementing more advanced error handling and user feedback mechanisms.

**GitHub Repo with Source Code :**

[FSDL Assn 4](https://github.com/RudradevArya/FSDL/tree/main/?tab=readme-ov-file#assn-4)

<https://github.com/RudradevArya/FSDL/edit/main/4_SPA_with_React>

**Output (Screenshots):**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated