Week 15: Assignment Work - Phone Book (Frontend)

Day 1: UI Design and Component Composition

The Architecture of a Contact Manager

The Phone Book application was designed as a high-interactivity Single Page Application (SPA). The goal was to create a clean, intuitive interface for managing a large volume of personal and professional contacts.

- **Component Structure**: The UI was broken down into a ContactForm for data entry, a SearchBar for filtering, and a ContactList for display.
- **Modular Design**: Each contact is represented by a ContactItem component, ensuring that the logic for editing or deleting a specific individual is encapsulated.
- **Semantic HTML**: Used <section> and <article> tags to ensure the phone book remains accessible and SEO-friendly.

Day 2: Advanced State Management for Contact Lists

Managing Dynamic Data Arrays

Unlike simpler projects, the Phone Book requires managing an array of objects that is constantly changing as users add, edit, or remove entries.

- **State Initialization**: Used the useState hook to initialize an empty array that would eventually hold objects containing name, phoneNumber, email, and category.
- **Reference Types**: Because arrays are reference types in JavaScript, we ensured that every update used the spread operator to maintain immutability.

- **Initial Data Load**: Leveraged the useEffect hook to trigger an API call to the backend upon the initial render to populate the list.

Day 3: Search Functionality and Real-Time Filtering

The Logic of the Search Bar

A critical feature of the Phone Book is the ability to find contacts instantly.

- **Controlled Inputs**: The search bar was implemented as a controlled component, where the value is tied to a React state variable.
- **The Filter Method**: JavaScript's .filter() method was used to create a "filtered view" of the contact list based on the user's input string.
- **Case Insensitivity**: To improve user experience, the search logic converts both the search query and the contact names to lowercase before comparison.
- **Performance**: Used the "Lifting State Up" pattern to ensure the search query in the SearchBar component could filter the list in the ContactList component.

Day 4: Form Validation and User Feedback

Ensuring Data Integrity

Data entry for a phone book requires strict validation to prevent invalid entries from reaching the database.

- **Validation Rules**: Implemented checks to ensure the "Name" field is not empty and the "Phone Number" follows a specific numeric pattern.
- **Error States**: Created a local error state to display red validation messages to the user if they attempt to submit an incomplete form.
- **Loading Indicators**: Added boolean state variables to show a "Saving..." message while the frontend waits for a response from the backend API.

Day 5: CRUD Operations on the Frontend

Integrating Interactivity

- **Adding Contacts**: Developed a handler function that takes the form data and sends a POST request to the backend, then updates the local state to show the new contact immediately.
- **Deleting Contacts**: Implemented a "Delete" button on each ContactItem. Clicking this triggers a DELETE request and then filters out that specific ID from the React state to update the UI without a page refresh.
- **Editing Logic**: Set up a "Select for Edit" flow where clicking a contact populates the ContactForm with that person's current details using the useState hook.