### **Approach to Each Task**

#### **Task 1: Setting up the React Project**

**Approach:**

* **Framework Selection**: Choose React for its component-based architecture, which allows for reusable components, and Vite for its fast build times and modern development features.
* **Initialization**: Initialized the project using create-vite to set up the basic project structure and install necessary dependencies.

**Personal Insight:** Using Vite over Create React App provides a significant boost in development speed, which is crucial for iterative design and testing.

#### **Task 2: Integrating Weather API**

**Approach:**

* **API Research**: Evaluated multiple weather APIs and selected WeatherAPI for its comprehensive data and ease of use.
* **API Integration**: Implemented API calls using asynchronous functions in React to fetch weather data based on user input.

**Personal Insight:** Ensuring proper error handling and loading states is critical for providing a seamless user experience during data fetch operations.

#### **Task 3: Building the Weather Dashboard UI**

**Approach:**

* **Component Design**: Designed components such as WeatherStory, EventPlanner, and FarmerDashboard to modularize the UI.
* **CSS Styling**: Used CSS Grid and Flexbox to create a responsive layout, ensuring the dashboard is user-friendly on various devices.

**Personal Insight:** Adopting a mobile-first approach in CSS styling ensures that the application is accessible and functional on smaller screens, which is often a primary mode of access for users.

#### **Task 4: Making Components Scrollable**

**Approach:**

* **Scrollable Main Container**: Ensured the main container (.container) is scrollable, while internal components like WeatherStory and EventPlanner expand to fit their content without individual scroll bars.
* **CSS Adjustments**: Used overflow: hidden for internal components and overflow: auto for the main container.

**Personal Insight:** Balancing between a clean UI and functionality can be challenging. Ensuring the main container handles scrolling provides a better user experience by avoiding nested scroll bars.

### **Sketches or Mockups**

Before implementing the final design, several sketches and mockups were created to visualize the layout and unique features:

#### **Initial Sketch:**

* **Header**: Title and input fields for city and date.
* **Main Container**: Divided into two columns: left for weather details and stories, right for event planning.
* **Weather Details**: Display forecast data with icons and relevant weather information.
* **Event Planner**: Calendar integration with weather context for planning activities.

#### **Mockup Highlights:**

* **User-Centric Elements**:
  + Clear and intuitive form input for city and date.
  + Immediate display of relevant weather information post data fetch.
  + Integrated event planning with visual cues about weather conditions.

## **Usage**

1. Enter the city name and select a travel date.
2. Click "Get Forecast" to fetch the weather data.
3. View the weather details, add weather stories, and plan events using the calendar.

## **Design Process**

### **Approach to Each Task**

1. **Setting up the React Project**:
   * Initialized with Vite for a modern development setup.
   * Used React for its component-based architecture.
2. **Integrating Weather API**:
   * Selected WeatherAPI for comprehensive data.
   * Implemented asynchronous data fetching with error handling.
3. **Building the Weather Dashboard UI**:
   * Designed modular components.
   * Used CSS Grid and Flexbox for a responsive layout.
4. **Making Components Scrollable**:
   * Ensured the main container is scrollable.
   * Adjusted CSS to prevent individual scroll bars on internal components.