**Project Overview**

This project involves creating two separate projects: a landing page and a weather app. The landing page is built with pure JavaScript, and the weather app uses React. The main objective is to load the weather app inside the landing page when clicking a button. Additionally, the landing page should display some weather details even after the weather app is closed.

**Project Breakdown**

**1. Landing Page (Pure JavaScript)**

**- Functionality:**

- Display a button labeled "Load Weather App."

- When clicking the button, the weather app (built with React) should load dynamically into the landing page.

- After closing the weather app, display basic weather information such as temperature, weather conditions, and last updated time on the landing page.

**- Implementation:**

- Use vanilla JavaScript and HTML/CSS to build the landing page.

- Load the weather app as an iframe or dynamically inject the React app into a container on the landing page.

- Use `localStorage` to save the weather details after the weather app is closed so that they can be displayed on the landing page.

**2. Weather App (React)**

**- Functionality:**

- Display the user’s current location using the browser’s Geolocation API.

- Fetch and display weather details such as temperature, humidity, wind speed, and weather conditions using a weather API (e.g., OpenWeatherMap API).

- Show a forecast for the next few days.

- On app closure, save the weather details to `localStorage` to be displayed on the landing page.

**- Implementation:**

- Use React and appropriate hooks (e.g., `useEffect`, `useState`) to handle API requests and manage component state.

- Use a service like `Axios` or the Fetch API to get weather data.

- Implement unit tests using Jest and React Testing Library.

- Implement end-to-end tests using a framework like Cypress or Puppeteer.

**3. Deployment**

**- GitHub Actions:**

- Set up a CI/CD pipeline using GitHub Actions to automate the build, test, and deployment process.

- Ensure that unit and e2e tests run on every pull request.

- Deploy the landing page and weather app to Firebase Hosting.

**- Firebase Hosting:**

- Configure Firebase for hosting both the landing page and the weather app.

- Use Firebase’s automatic deployment feature to deploy the projects from GitHub.

**4. Documentation**

**- Design Decisions:**

- Create a `design\_decisions.txt` file to document assumptions and reasoning behind the architectural and technical decisions made during the project.

- Include the steps to run the project locally, details on the deployment process, and instructions on how to test the project.

**- README File:**

- Provide an overview of the project, instructions on how to set up and run the projects, links to GitHub repositories, and Firebase-hosted URLs.

**Steps to Implement the Solution**

**1. Set Up the Repositories:**

- Create two separate repositories on GitHub: one for the landing page and one for the weather app.

- Share these repositories with the provided email.

**2. Landing Page Development:**

- Build the landing page using HTML, CSS, and vanilla JavaScript.

- Implement the button to load the weather app dynamically.

- Set up basic tests for the landing page.

**3. Weather App Development:**

- Create a React app using `create-react-app`.

- Implement the features for fetching and displaying the current weather and forecast.

- Set up unit and e2e tests.

**4. Integration:**

- Implement the dynamic loading of the weather app on the landing page.

- Ensure weather details are displayed on the landing page after the weather app is closed.

**5. CI/CD Setup:**

- Configure GitHub Actions for automatic testing and deployment.

- Deploy both projects to Firebase.

**6. Testing:**

- Write and run unit tests for both the landing page and the weather app.

- Write and run e2e tests to ensure the entire flow works as expected.

**7. Documentation:**

- Write the `design\_decisions.txt` file.

- Update the README files with all necessary information, including GitHub and Firebase links.

**8. Submission:**

- Share the GitHub repositories with the provided email.

- Include the Firebase-hosted URLs in the README files.

This plan covers the essential steps and best practices to complete the project efficiently and according to the requirements.