API Integration and Data Flow

API Integration:

1. Weather API:

- The application integrates with the Tomorrow.io weather API to fetch real-time and forecast weather data.
- Two endpoints are used:
- Real-Time Weather: ‘https://api.tomorrow.io/v4/weather/realtime’
- Forecasted Weather: ‘https://api.tomorrow.io/v4/weather/forecast’
- API key is included in the request headers for authentication.

2. Axios Library:

- The Axios library is used to make asynchronous HTTP requests to the weather API.
- The ‘weatherRequest’ function is a helper function that handles making API requests with appropriate configurations.

Data Flow:

1. State Management:

- The component uses React ‘useState’ hooks to manage various states:
- ‘realTimeWeather’: Holds real-time weather data.
- ‘forecastWeather’: Holds forecasted weather data.
- ‘loading’: Indicates whether data is currently being loaded.
- ‘userLocation’: Holds the user’s location.
- ‘inputValue’: Holds the value entered by the user in the location input field.

2. Effect Hook for Data Fetching:

- The ‘useEffect’ hook is used to handle asynchronous operations and data fetching.
- When the component mounts or when ‘userLocation’ changes, it initiates the following:
- Fetches real-time weather data using the ‘fetchRealTimeWeather’ function.
- Fetches forecasted weather data using the ‘fetchForecastWeather’ function.
- If the user location is not available, it attempts to retrieve it using geolocation.

3. Weather Data Update:

- Upon successful API requests, the corresponding weather data is updated in the component’s state using the ‘setRealTimeWeather’ and ‘setForecastWeather’ functions.

4. User Location Handling:

- If the user denies geolocation access or if the browser doesn’t support it, an error message is displayed, and the user is prompted to enter the location manually.

5. User Interface (UI) Updates:

- The UI is divided into sections for real-time weather, forecasted weather, and user preferences.
- Chakra UI components are used to structure and style the UI elements.
- The temperature, description, and date/time details are displayed for both real-time and forecasted weather.

6. User Input Handling:

- The user can enter a location manually in the input field.
- The input value is stored in the ‘inputValue’ state.
- Clicking the "Fetch" button triggers the ‘handleButtonClick’ function, setting the ‘userLocation’ state to the entered value.

7. Loading Spinner:

- A loading spinner is displayed while the data is being fetched, providing feedback to the user.