# React JS Tasks

## Task 1: Context Setup

1. \*\*Run the Application\*\*  
- Run your `hamro-sansar` app and start the fake API server.

2. \*\*Create Context Folder\*\*  
- Create a new folder named `contexts`.

3. \*\*Create CitiesContext.jsx\*\*  
- Inside the `contexts` folder, create a file named `CitiesContext.jsx`.

4. \*\*Develop the Context\*\*  
- In `CitiesContext.jsx`, start by creating a new context and a function named `CitiesProvider`.

5. \*\*Refactor App.jsx\*\*  
- Move the necessary `useState` and `useEffect` logic from `App.jsx` to `CitiesContext.jsx`. Eliminate all prop drilling from `App.jsx`.

6. \*\*Export and Use CitiesProvider\*\*  
- Export `CitiesProvider` from `CitiesContext.jsx`. Wrap your `App.jsx` with the `CitiesProvider` as demonstrated in the previous class.

## Task 2: Consume Context

1. \*\*Use the CitiesContext\*\*  
- Consume the `CitiesContext` wherever it is required in the app.

2. \*\*Create a Custom Hook\*\*  
- Build a custom hook to simplify working with the `CitiesContext`.

## Task 3: City View Component

1. \*\*Introduce Global State\*\*  
- Create a new state named `currentCity` in `CitiesContext.jsx` to make it global.

2. \*\*API Integration\*\*  
- Add `useState` logic in `CitiesContext.jsx` and handle API calls with the endpoint: `http://localhost:8000/cities/[city.id]`.

3. \*\*Create Method\*\*  
- Develop a method named `getCity` with an `id` parameter to fetch city details. Pass `getCity` and `isLoading` as values in the provider.

4. \*\*Update CityItem.jsx\*\*  
- In `CityItem.jsx`, implement a missing `<Link>`:  
```jsx  
<Link className={styles.cityItem} to={`${id}?lat=${position.lat}&lng=${position.lng}`}></Link>  
```

5. \*\*Style Active City\*\*  
- Use the `currentCity` state to highlight the active city with the className:  
```jsx  
className={`${styles.cityItem} ${id === currentCity.id ? styles['cityItem--active'] : ''}`}  
```

## Task 4: Back Button

1. \*\*Add Back Button\*\*  
- Implement a back button in `City.jsx` similar to the one already integrated in `Form.jsx`.

2. \*\*Create Common Component\*\*  
- Refactor the back button into a common component to be reused in both `City.jsx` and `Form.jsx`.

## Task 5: Map Integration

1. \*\*Integrate react-leaflet\*\*  
- Use `react-leaflet` to implement a map in `Map.jsx`.

2. \*\*Review Documentation\*\*  
- Refer to the official documentation, go through the installation process and quick setup guide, and address any issues.