SE3040 – Application Frame Work

The **Country Explorer** application uses the REST Countries API to fetch real-time data about countries. This API provides comprehensive details such as:

- Country name
- Flag image
- Capital
- Region and subregion
- Population
- Languages
- Country code (used for navigation)

Challenges faced and how it solved

1. Inconsistent API response structure

- **Challenge:** Some country data fields like `capital`, `languages`, or `subregion` were missing or undefined in certain API responses.
- **Solution:** Used optional chaining and fallback handling in React components.

2. Handling API errors gracefully

- **Challenge:** If a country name was mistyped or didn't exist, the API call failed.
- **Solution:** Used try-catch and axios.get(...).catch(...) to display user-friendly error messages using window.alert or inline feedback.

3. Cross-origin requests (CORS) during development

- **Challenge:** Occasionally, the API would throw CORS errors when accessed from localhost.
- **Solution:** Used a CORS proxy (when needed) or configured the browser for local testing. Most issues resolved once served over a proper dev server.

4. Testing async API calls

- **Challenge:** Unit testing components that use axios and useEffect was tricky.
- **Solution:** Used jest.mock('axios') and @testing-library/react's waitFor() to properly mock and await API responses in test files.

5. Routing dynamic details page

- **Challenge:** Creating a dynamic route using React Router (/country/:code) and accessing the code via useParams.
- **Solution:** Configured Route components and used useParams() inside the CountryDetailsPage component to extract and use the code to fetch data.