# COUNTRY HUB

Learn About Countries


**Visit Via**

**countryhub.com**

## Chosen APIs:

The COUNTRY HUB application integrates the REST Countries API v3 to fetch country-related data. In here, COUNTRY HUB perform several tasks such as searching country by name, searching country by country code, giving the list of countries around the world, filtering countries by region of the country, filtering countries by language of the country and marking countries as favorite countries with the user authentication. The following endpoints were primarily used.

1. **Fetch Country Details:**

`https://restcountries.com/v3.1/all`

2. **Search by country name:**

`https://restcountries.com/v3.1/name/{name}`

3. **Search by country code:**

`https://restcountries.com/v3.1/alpha/{countryCode}`

4. **Filter by country region:**

`https://restcountries.com/v3.1/region/{region}`

5. **Filter by country language:**

`https://restcountries.com/v3.1/lang/{language}`

These endpoints provide the country with details including country name, country code, flags of the countries, population of the countries, region, subregion, capital cities and more.

## Challenges faced and how they were resolved:

1. Inconsistent API Responses:

**Issue**: The REST Countries API v3 sometimes returns data in different structures depending on the query (e.g., language names in different formats).

**Solution**: Implemented utility functions to normalize API responses. For example, language values were transformed from object structures into arrays for consistent rendering.

2. Filtering Overlap:

**Issue**: Dealing with multiple filters (region + language) meant either merging API responses or client-side filtering.

**Solution**: Introduced individual filtering controls and performed client-side filtering after the initial API responses to boost performance as well as user experience.

3. Handling 404 errors and missing data:

**Issue**: Filtering by an invalid country name or an invalid filter causes the API to return a 404 or undefined fields.

**Solution**: Added error handling with default UI messages (e.g., "No countries found") and safe checks for optional fields (i.e., languages, capital, flags).

4. Mocking `react-router-dom` in frontend tests:

**Issue**: Unit tests using Jest failed with "`Cannot find module 'react-router-dom'`" during mocking.

**Solution**: Ensured `react-router-dom` was properly installed and the mock setup used `jest.requireActual` correctly

5. Data Security and Privacy:

**Issue**: Proper handling of sensitive user data was a top priority.

**Solution**: JWT for authentication ensured that only authenticated users would be able to access their favorite countries. Additionally, sensitive data such as login passwords would be encrypted with bcrypt algorithms.

6. Page navigation through Navbar:

**Issue**: Navigating to the signup page does not happen when open the application and directly navigating to the login page by tapping on the signup option in the navbar.

**Solution**: Enabled that the userEffect navigates to the sign-up page by clicking on the signup option when user is not in the logged in state. That meant I ensured that the navigation happened in the logout state as well.

7. Cross-Origin issues (CORS):

**Issue**: When connecting frontend and backend (e.g., for login/favorites), there were CORS errors.

**Solution**: Enabled CORS middleware on the backend server and handled credentials correctly: "include" in fetch requests.

## Outcome:

According to the challenges faced during the development process, COUNTRY HUB successfully provides a dynamic service to the users with,

1. A smooth experience navigating and exploring countries.
2. Functions such as searching, filtering and marking as favorites features.
3. Protected user authentication and data persistence.

The COUNTRY HUB – The rest countries API application is a dynamic web application satisfying user experience and satisfaction. And it proved reliable for this project.