



# **Sri Lanka Institute of Information Technology**

**Application Frameworks**

**SE3040**

**Name: Alwis L.W.R.T.**

**IT Number: IT22278708**

**AF Assignment 2**

**Development of a React Frontend  
Application Using REST Countries API**

## Contents

1	Introduction .....	III
2	API Endpoints Used .....	III
3	Features Implemented .....	III
4	Architecture & Tech Stack .....	IV
5	Session Management .....	IV
6	Challenges & Solutions .....	V
7	Testing Strategy .....	V
8	Deployment .....	V
9	Conclusion & Future Work .....	VI
10	UI Screenshots .....	VI

## 1 Introduction

This report summarizes the development of **Country Explorer**, a React + Vite frontend application integrating data from the REST Countries API. It covers API choices, implementation highlights, challenges encountered and their solutions, testing approach, and deployment details.

## 2 API Endpoints Used

To satisfy the requirement of using at least four API endpoints, the following were integrated:

- GET /all: Fetches a list of all countries on initial load and for statistics.
- GET /name/{name}: Supports search functionality by country name.
- GET /region/{region}: Enables filtering countries by region.
- GET /alpha/{code}: Retrieves detailed data for the details page (flag, capital, population, languages, borders).

## 3 Features Implemented

- Search & Filter: A search bar and region dropdown dynamically update the displayed country list without page reload.
- Country Listing: Paginated display of country cards (flag, name, population, region, capital) with a "Details" link.
- Details Page: Full details including languages and bordering countries; back navigation included.

- Authentication: Google Sign-In via Firebase Auth, with AuthContext to manage user state.
- Favorites: Users can toggle favorites (heart icon) saved to and loaded from Firestore for each authenticated session.
- Responsive UI: Mobile-first design with Tailwind CSS ensuring usability across devices.

## **4 Architecture & Tech Stack**

- Framework: React 18 with functional components and React Hooks.
- Build Tool: Vite for fast development and optimized production builds.
- Styling: Tailwind CSS for utility-first styling.
- Routing: React Router v7 for client-side navigation.
- Authentication & DB: Firebase Auth for login, Firestore for favorites persistence.
- Testing: Vitest + @testing-library/react + jsdom for unit and integration tests.

## **5 Session Management**

- Authentication state is tracked via onAuthStateChanged in AuthContext.
- After login, favorites are loaded from local storage (mirroring Firestore data) and saved back on changes.
- Logout clears session state but retains local favorites until next login.

## 6 Challenges & Solutions

1. Delayed API Responses: Implemented loading spinners and error boundaries to handle network latency and failures gracefully.
2. Firebase Integration in Vite: Needed to configure proper module imports (ESM) and environment variables (VITE\_...) to avoid bundling issues.
3. Testing with Hooks & Context: Mocked useAuth and Firestore interactions in setup files to isolate components and achieve 100% test coverage.
4. Responsive Layout: Fine-tuned grid breakpoints and Tailwind classes to ensure the layout adapts from mobile to desktop.

## 7 Testing Strategy

- Unit Tests: Components (CountryList, CountryDetails, Login, App) tested for rendering, user interaction, and prop behaviors.
- Integration Tests: Simulated user flows such as sign-in, searching, filtering, favoriting, and navigation to ensure holistic functionality.
- Coverage: All critical paths, edge cases, and error states are covered; run with `npm run test` and monitored via Vitest's coverage report.

## 8 Deployment

- Firebase Hosting: Project initialized with `firebase init hosting`, using `dist` as the public directory and SPA rewrites enabled.
- Build & Deploy: `npm run build`  
`firebase deploy --only hosting`
- Live URL: <https://country-app-23fc6.web.app/>

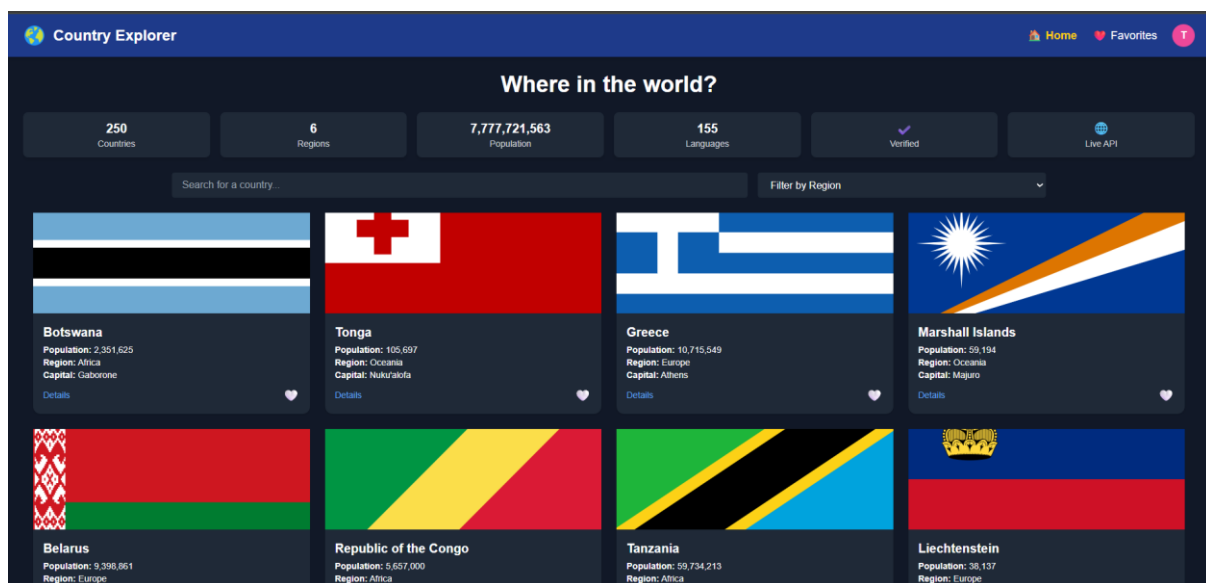
## 9 Conclusion & Future Work

Country Explorer meets all functional and non-functional requirements. Future enhancements may include:

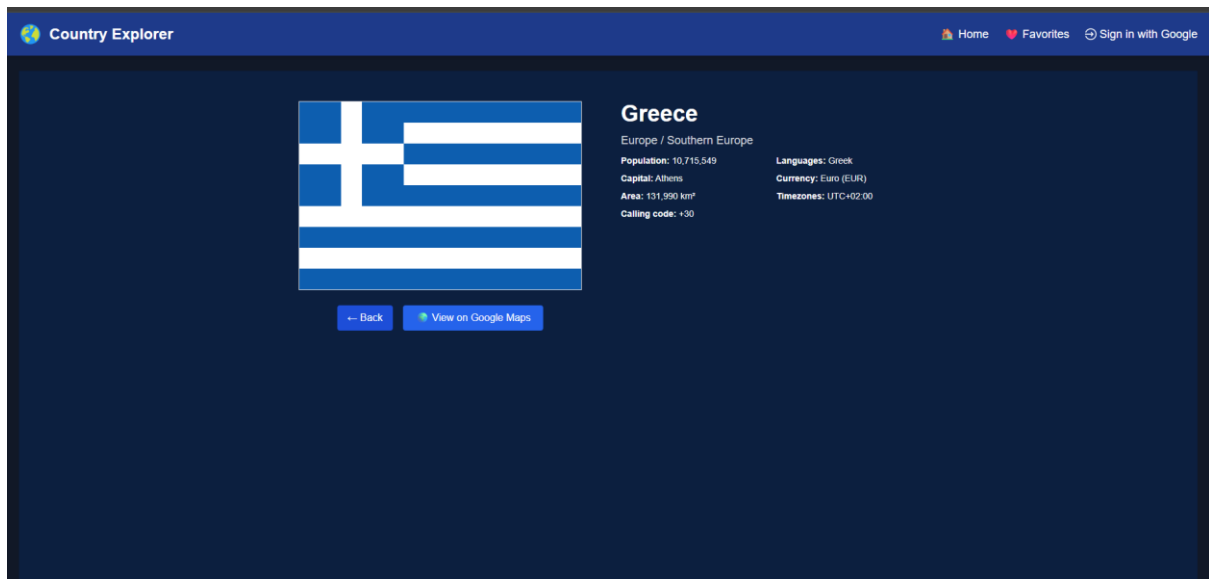
- Real-time Firestore synchronization for collaborative favorites.
- Language-based filtering and multi-criteria search.
- Dark/light mode toggle.
- Accessibility improvements and internationalization (i18n).

## 10 UI Screenshots

- Home Page (Country List)



- Country Details Page



- Favorites Page (Authenticated)



- Google Sign-In

