Country Dashboard Project Documentation

Project Overview

The Country Dashboard is a full-stack web application designed to display information about countries using data from the REST Countries API. It includes backend APIs for fetching and processing country data and a frontend for displaying and interacting with the data. The application enables users to view, search, filter, and compare country details, offering visual insights with charts and maps.

Features

1. **Backend (Node.js, Express, TypeScript):**
- Fetch data from the REST Countries API.
- Provide the following endpoints:
- GET /countries: Fetch a list of all countries.
- GET /countries/:code: Fetch detailed information about a single country using its code.
- GET /countries/region/:region: Fetch countries filtered by a specific region.
- GET /countries/search: Search for countries by name, capital, region, or time zone.
- Implement data caching to reduce redundant API calls.
- Handle API errors (e.g., invalid country codes, server errors).
- Process data to extract key fields like name, population, flag URL, region, and currency.

2. **Frontend (React/Next.js, TypeScript):**

- Display a list of countries with key information such as name, flag, region
- Allow users to search countries by name or filter by region or time zone.
- Show detailed country information, including population, currency, and languages.
- Provide a **compare feature** to view side-by-side comparisons of two countries.
- Include visualizations:
- **Charts**: Compare populations of selected countries using a bar chart.
- **Maps**: Display the location of a country on a map using latitude and longitude.
- Ensure a responsive and clean UI/UX with frameworks like Tailwind CSS.
- Display loading and error states for better user experience.

Tech Stack

1. **Frontend**:
- Framework: React with Next.js
- Language: TypeScript
- CSS Framework: Tailwind CSS
- Libraries:
- React-Leaflet for maps.
- Chart.js for data visualization.

2. **Backend**:
- Framework: Express.js
- Language: TypeScript
- REST Countries API as the data source.
- Axios for making API calls.

3. **Deployment and CI/CD**:
- GitHub Actions for CI/CD pipeline (optional).

Setup Instructions

Prerequisites:
- Node.js (v16 or above)
- npm or yarn
- Git installed on the local machine

Backend Setup:

1. Clone the repository: https://github.com/satya1105/PWC.git
   cd country-dashboard-backend
2. Install dependencies:
   npm install
3. Run the backend server:
   npm run start
4. Backend will be running on http://localhost:3001.

Frontend Setup:

1. Navigate to the frontend folder:
   cd country-dashboard-frontend
2. Install dependencies:
   npm install
3. Run the frontend server:
   npm run dev
4. Open the app in the browser at http://localhost:3000.

API Endpoints

Backend Endpoints:

1. **Get all countries:**
- GET /countries
- Response: List of all countries with basic details.

2. **Get country details:**
- GET /countries/:code
- Response: Detailed information about a specific country.

3. **Get countries by region:**
- GET /countries/region/:region
- Response: List of countries within a specific region.

4. **Search countries:**
- GET /countries/search
- Query Params:
- name: Search by country name.
- capital: Search by capital city.
- region: Search by region.
- time zone: Search by time zone.

Key Components

Frontend Components:

1. **Country Card**:
- Displays country name, flag, region, and population.
2. **Search Bar**:
- Allows users to search for countries by name.
3. **Filter Dropdown**:
- Provides options to filter countries by region or time zone.
4. **Comparison Chart**:
- Displays a bar chart comparing populations of two selected countries.

Deployment

- Deploy the frontend on platforms like Vercel
- Use GitHub Actions to automate the build and deployment process.

Future Enhancements

- Add user authentication for saving favourite countries.
- Include more advanced visualizations (e.g., line charts for historical population data).
- Enable offline support with service workers.
- Optimize backend performance with advanced caching techniques.