Coding Challenge: Flag Explorer App

Overview

This challenge is designed to assess your skills in building a modern software application. You will develop a backend API based on a Swagger document, integrate a frontend that retrieves and displays country flags using an Open API, and create a user interface with two distinct views. Your solution must follow modern programming practices, include automated tests, and provide pipeline integration with YAML configuration.

Requirements

Part 1: Backend API

- Implement a REST API using the provided Swagger documentation (see below).
- The API should:
  a. Expose an endpoint to retrieve a list of countries.
  b. Support querying by country name to retrieve additional details (e.g., population, capital).
- Use modern backend architecture patterns (e.g., MVC or Clean Architecture).
- Write unit and integration tests for the API.

Part 2: Frontend Application

- Build a frontend UI (web or mobile) that interacts with the backend API.
- Use a modern frontend framework (React, Angular, Vue.js, or similar for web; React Native, Flutter, or similar for mobile).
- Features:
  a. **Home Screen:** Display all country flags in a grid layout. Fetch flag images using the Open API: https://restcountries.com/v3.1/all.
  b. **Detail Screen:** When a flag is clicked, display details about the country, including:
    - Country name
    - Population
    - Capital
- Write unit and integration tests for the frontend.

Part 3: Pipeline Integration

- Provide a YAML configuration file for setting up a CI/CD pipeline.
- The pipeline should:
  a. Run automated tests for both frontend and backend.
  b. Build the application.
  c. Package the frontend and backend for deployment.

Submission

1. A GitHub repository link containing:
   - Complete source code for the backend and frontend.
   - Tests for both parts.
   - Pipeline YAML file.

- A `README.md` with clear instructions on how to set up and run the application.
- A `.gitignore` file

Swagger Documentation

```yaml
openapi: 3.0.0
info:
  title: Country API
  version: 1.0.0
paths:
  /countries:
    get:
      summary: Retrieve all countries
      responses:
        '200':
          description: A list of countries
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Country'
  /countries/{name}:
    get:
      summary: Retrieve details about a specific country
      parameters:
        - name: name
          in: path
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Details about the country
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/CountryDetails'
components:
  schemas:
    Country:
      type: object
      properties:
        name:
          type: string
        flag:
          type: string
    CountryDetails:
      type: object
      properties:
        name:
          type: string
        population:
          type: integer
        capital:
          type: string
```

```
        flag:
          type: string
```