RA2 Technical Interview Coding Challenge

Overview

For this challenge, you will build a simple API that wraps around a free weather API (e.g., OpenWeatherMap). The goal is to create an API that allows users to retrieve weather data for a city from a preset list of cities using an ID.

You are free to choose either FastAPI or Flask for this project, and you are encouraged to follow good software development practices, such as secret management and collaboration-friendly workflows (e.g., using Git). The challenge should take no more than 2-4 hours to complete.

Requirements

Weather endpoint

The first endpoint should take a variable city_id as an argument and return the current weather at that city. Create a preset list of cities in your database. The city_id in the weather endpoint should correspond to a city in this preset list, not directly to the third-party API's city identifier. Log each weather request in a database table, including at minimum: timestamp, requested city_id, and the response status (success/failure).

History endpoint

Create a history endpoint that returns data for the 5 most recent successful requests to the weather endpoint. Include the timestamp, city name, and a summary of the weather data returned for each request.

Details

- Use either SQLite (for simplicity) or PostgreSQL (for a more production-like setup) to store:
    - A table of preset cities with at least city_id and city_name
    - A table for logging weather requests
- Provide the schema for your database tables in your documentation
- Structure your project and code so that it would be easy to collaborate with other developers
- Consider the maintainability and testability of your code
- Implement appropriate error handling for scenarios such as invalid city_id, third-party API failures, and database errors. Document your error handling approach.
- Write unit tests for your API endpoints and any critical functions.
- If you don't have time to implement everything you'd like, make note of the improvements you'd like to make

Bonus

- Provide a Dockerfile so the API can be easily containerized and run in a local environment
- Consider edge cases that might arise and how to handle them

Submission Guidelines

- Push your code to a public or private Git repository (GitHub, GitLab, etc.)
- Include any tests you have written
- Provide documentation that includes:
    - Setup instructions for running your API locally
    - A brief explanation of your project structure
    - Any trade-offs or architectural decisions you made
    - A list of potential improvements or features you would add given more time
- You may use a README.md in your repository or a separate Google Doc for this documentation.

Deadline

Please complete the assignment and share your submission by email in advance of your interview. You will have the opportunity to discuss your work and any challenges you encountered during your interview.