Full Stack Developer Challenge

This is the technical code challenge for the full stack engineer position.

Description

The CitiBike Trip System Data provides records of trips taken by users of the CitiBike bike-sharing program in New York City. The data includes trip start and end times, stations/locations, and the user's bike number among other fields.

What should you do?

You work for a company that analyzes the bikers behaviour in New York City. Your task is to provide a tool (that can be run in a Production environment) to the Data Engineering team to be able to download the different CitiBike Trip datasets.

Tasks

You need to perform the following tasks:

1. **Backend** (**API** with FastAPI)
    a. Provide an endpoint that takes the year and month (month is optional) as parameters and it downloads the specific datasets based on the input parameters
    b. Provide a Swagger UI so the Data Engineering team can use it for documentation purposes and know how to use it.
    c. No authentication is required for accessing the API.

2. **Frontend** (**React** is preferable)
    a. Swagger UI is good for internal purposes, the Data Engineering team will be using it. But you need to provide a simple UI to the client where the users can download the different CitiBike Trip datasets. You can use for this functionality the endpoint implemented in the previous step.
    b. No login page is required.

3. **Optional (Bonus)**
    a. Implement an endpoint that is able to extract a specific dataset (you can filter by year and month. It is up to you), parse it and return as an output to answer the next questions:
        i. What days of the week are most rides taken on?
        ii. Top 5 stations with the most starts (showing # of starts)
    b. Make available this functionality in the UI. Feel free to show the results in a table, chart, … whatever you consider is better.

Deliverables

- **Repository:** It would be nice if you can create a public repository in your Github account or similar, push the code there and share it with us
- **Documentation:** The repository should include a documentation file in Markdown explaining:
    - How you approached the problem
    - Assumptions or decisions made
    - Steps to run the code.
    - If you have implemented **Task 3** please explain the approach you have followed.
- **Submission Guidelines:** Once you finish the challenge, ping us via the contact email we have provided.
- **Final Steps:** Once we take a look at it (maybe we can ask some questions about your decisions) we will schedule something with you to present the solution to the team.

Questions to resolve

- Consider the next scenario: The application has been a success and is being used by many clients, both synchronously through the web platform you developed and asynchronously through an event-driven system. As a result, the API is now starting to receive millions of requests per day.
    - How would you ensure the solution scales and the API has a low latency? You don't need to implement anything related to this.
    - How and what would you monitor if you had to go to production with your solution? You don't need to implement anything related to this.
- As the application has been a success, now we need to implement an authentication method for accessing the API.
    - What authentication method should we use and why? (There is no good answer, just explain any method or framework that you know and mention its benefits and disadvantages: JWT, OAUTH ..etc). You don't need to implement anything related to this.
    - Describe the changes needed at the front-end level to be able to authenticate with the API properly. You don't need to implement anything related to this.

NOTE: Please include this as part of your documentation file generated in the deliverables.

Evaluation Criteria

- What questions do you ask to clarify the requirements
- Correctness and completeness of the solution.
- How do you approach the problem

- Code Quality
- Data Validation
- Error Handling
- Coverage tests