Candidate Brief

Role Overview

As part of our hiring process, we want to evaluate your skills in designing and developing a clean, functional, and well-structured application. This task will simulate a real-world scenario and allow us to assess your ability to deliver solutions that align with best practices and demonstrate your technical proficiency.

Task Description

You are tasked with creating a REST API for a car rental service. The API should fulfill the following requirements:

1. **Endpoints**:
    - **List Available Cars**: Provide an endpoint that lists all available cars for a specific date.
    - **Create a Booking**: Provide an endpoint to create a booking for a car on a given date.
2. **Data**:
    - You may define the necessary data for the application yourself. The data does not need to be overly complex, and you may omit any unnecessary details such as images.
    - For the moment, use a file-based storage mechanism (e.g., JSON file) for simplicity. There's no need to set up a database, we might get to that later on.
3. **Technical Requirements**:
    - Use any framework or tools of your choice (Python based), though we primarily work with **FastAPI** and encourage its use if you are familiar with it.
    - Include basic **logging** to capture key application events such as:
        - Successful and failed booking attempts.
        - Queries for available cars.

- Include basic **unit tests** or **integration tests** for at least one endpoint.
- Ensure your solution is simple yet adheres to best practices, including clear structure, clean code, and appropriate use of architectural patterns.
- Containerization of your application using Docker is optional but highly appreciated.

4. **Deliverables**:
   - A fully functional REST API with the specified endpoints.
   - A public repository (e.g., GitHub) containing:
     - The complete codebase.
     - A clear commit history that reflects your working process.
   - Include logging outputs in the application runtime.
5. **Documentation**:
   - Include a brief explanation (in the README file) of your design choices and reasoning. The explanation does not need to be exhaustive but should be clear and concise.
   - Test cases with instructions on how to run them.

Guidelines

- Keep the task scoped to approximately **2-3 hours of work**.
- While simplicity is encouraged, ensure the solution is robust and demonstrates your skillset effectively.
- Use good coding practices, such as clean code, modular design, and clear commit messages.
- Avoid overcomplicating the solution where a straightforward approach suffices.

Evaluation Criteria

Your submission will be evaluated based on the following aspects:

1. **Functionality**:
   - Do the provided endpoints meet the requirements?

2. **Code Quality**:
    - Is the code clean, maintainable, and readable?
    - Are industry best practices followed?
3. **Testing**:
    - Are basic tests included and functional?
4. **Logging**:
    - Are key application events logged effectively?
    - Do logs provide meaningful information for debugging and monitoring?
5. **Design**:
    - Does the solution demonstrate good architectural principles?
    - Is the project structure logical and modular?
6. **Documentation**:
    - Does the README file clearly explain your design choices?
    - Is it easy to understand how to run and test the application?
7. **Optional Enhancements**:
    - Dockerization of the application.
    - Additional thoughtful features or optimizations.