Lucra

Take-Home Test for Mid-Level Backend Developer

Objective:
Develop a backend system using Python, Flask, and MySQL that provides API endpoints for an AI-powered feature. The backend should handle interactions with a database and potentially integrate with an AI service or model to process data.

Requirements:

1. API Development:
   - Develop a Flask application with a set of API endpoints that interact with a MySQL database and an AI service/model.
   - Implement RESTful principles in API development, ensuring endpoints are well-structured and intuitive.

2. Database Integration:
   - Design and implement a MySQL database schema relevant to the application's functionality (e.g., storing user data, AI inputs/outputs).
   - Implement CRUD operations through the API to interact with the database.

3. AI Integration:
   - Integrate an external AI API or a Python AI library to process data. For example, use an NLP library to analyze text data or an image processing library for image data.
   - Ensure the AI component is modular and interacts with the rest of the application through well-defined interfaces.

4. Authentication and Authorization:
   - Implement basic authentication and authorization to secure the API endpoints. Consider using Flask extensions like Flask-HTTPAuth or Flask-JWT.

5. Error Handling and Logging:
   - Develop robust error handling to manage and respond to exceptions gracefully.
   - Implement logging to track the application's behavior and potential issues.

6. Testing:
   - Write unit and integration tests for your API endpoints and business logic.
   - Ensure tests cover various scenarios and edge cases.

7. Documentation:
   - Document your API endpoints using tools like Swagger or Postman. Provide clear instructions on how to interact with the API.
   - Include a README file in the GitHub repository with:
     - Detailed setup and run instructions for the Flask application.
     - An overview of the API endpoints and their functionalities.
     - Information on the AI integration and how it's used within the application.

8. GitHub Repository:
   - Push your application code to a public GitHub repository.
   - Ensure the repository includes all necessary files (excluding virtual environments and sensitive credentials) and instructions for setting up and running the application.

Submission:

- Provide the URL to the GitHub repository containing your project.
- The repository should have a clear README with setup instructions and detailed API documentation.

Evaluation Criteria:

- **Functionality:** The API functions correctly, handling various requests and responses as expected.
- **Code Quality:** Code is clean, well-organized, and follows Pythonic conventions.
- **Database Design:** Efficient and logical use of MySQL, with well-designed schema and interactions.
- **AI Integration:** Effective use of AI to enhance the application's backend functionalities.
- **Security:** Implementation of basic security measures for API access.
- **Error Handling and Logging:** Comprehensive error handling and logging for troubleshooting and monitoring.
- **Testing:** Thorough testing demonstrating reliability and robustness of the application.
- **Documentation:** Clear and detailed documentation for setting up the project and interacting with the API.