Technical homework

Hey there aspiring Python developer! 🐍 Are you ready to embark on an exciting mission with Recog? We're diving into the fascinating world of AWS, Python, and healthcare AI. Imagine using AWS Lambda to create a super-smart assistant that interacts with Language Models to assist medical professionals in diagnosing patients! Let's make it happen! 🚀

Challenge: Super Smart Medical Assistant Lambda

You're about to create a Python AWS Lambda function using AWS SAM (Serverless Application Model). Our goal is to build a smart medical assistant that communicates with ChatGPT (a powerful language model) to assist doctors in diagnosing patients based on symptoms provided.

Problem Statement

You're tasked with building a Python Lambda function that interfaces with an external ChatGPT API. The Lambda should take in a set of symptoms as input and generate a potential diagnosis for a patient. You'll need to mock the API during testing and ensure proper logging and event parsing using AWS Lambda Powertools.

Specifications

1. **Input**: The Lambda should accept a JSON payload containing a list of symptoms. Each symptom should be a string.

Example Input:

```
{
  "symptoms": ["fever", "headache", "cough"]
}
```

2. **Output**: The Lambda should respond with a JSON object containing a diagnosis generated by ChatGPT based on the provided symptoms.

Example Output:

```
{
  "diagnosis": "You might have a common cold. Make sure to rest and stay hydrated."
}
```

3. **Integration**: The Lambda should integrate with the ChatGPT API to generate the diagnosis.
4. **Testing**:
    - Write unit tests to mock the API using appropriate testing frameworks.
    - Write end-to-end tests using AWS SAM with mock events.
5. **Event Format**: The Lambda should be able to handle events in the AWS API Gateway V2 format.
6. **Logging and Parsing**: Utilize AWS Lambda Powertools for logging and event parsing.

Deliverables

Your mission is not complete without delivering the goods! Here's what we're expecting you to deliver:

1. **GitHub Repository**
    - Create a GitHub repository with a meaningful name related to the project.
    - The repository should host the entire project, including code, tests, and documentation.
2. **AWS SAM Structure**
    - Implement the AWS SAM structure to organize your serverless application.
    - Include the necessary AWS SAM configuration files.
3. **Lambda Function Code**

- Write the Python code for the Lambda function that interfaces with the ChatGPT API to generate diagnoses based on symptoms.
- Ensure the code adheres to best practices and follows the requirements mentioned earlier.

4. **Unit Tests**
    - Write comprehensive unit tests to validate the functionality of your Lambda function.
    - Mock the ChatGPT API calls to isolate and test the different components of your code.
5. **End-to-End Tests**
    - Implement end-to-end tests using AWS SAM with mock events to verify the complete functionality of the Lambda function.
6. **Documentation**
    - Provide detailed documentation describing the project, how to set it up locally, and how to deploy it on AWS using SAM.
    - Include instructions on running tests, handling dependencies, and integrating with the ChatGPT API.
7. **README File**
    - Craft a well-structured README file with an overview of the project, setup instructions, and any other relevant information for developers and contributors.

Submission Instructions

1. **GitHub Repository**
    - Share the link to the GitHub repository with us.
2. **Readme Updates**
    - Ensure that your README is updated with clear instructions for setup, testing, and usage of the application.
3. **Documentation Review**

- Confirm that your documentation covers all necessary aspects of the project and is easy to follow.

Final Considerations

- **ChatGPT Integration**: Emphasize integration itself over the accuracy of the ChatGPT prompt or its output. The key is to successfully set up and mock the integration for testing purposes.
- **Deadline Flexibility**: If the project isn't complete by the deadline, that's okay! Submit what you have and be prepared to discuss your progress and intentions during the review.
- **End-to-End Functionality**: Prioritize achieving end-to-end functionality. Aim for a functional solution that receives input and produces the desired output. Quality over quantity!

Remember, it's about delivering a solid proof of concept. Let's aim for a robust integration and a glimpse of the potential of this medical assistant. Good luck! 🚀

Resources

- AWS SAM Documentation
- Local Testing with SAM
- ChatGPT API Documentation
- AWS Lambda Powertools Documentation

Good luck on this journey to revolutionize healthcare with AI magic!