AI Agent for Operating a Backend System Using Natural Language

Introduction

This project focuses on developing an AI agent that a user can interact with using natural language and that will act on the user's behalf to operate a backend system. For the purposes of this project the backend system could be a simple key value store. But the same principles could be used to operate any other backend system or API.

Operations on the backend system could involve inserting key-value pairs, updating them, or deleting them.

The user might request of the agent: "Insert an entry with key "my.test" and value "my.value". The system would make use of an LLM system to determine what action should be performed, in this case the intended request is: action=insert, key=my.test, value=my.value. The agent would then perform the insert for the user.

Task 1: Prompt design

Design prompts that can be passed to an LLM system along with the user's natural language sentence and that will respond in a structured format such as JSON. The response should indicate what action is requested such as insert, update, or delete, as well as what key and value are involved.

Deliverables:

1. Prompt or prompts to be used
2. Sample requests and responses that illustrate the various actions that can be performed. This might include limitations on the types of requests that result in usable responses.
3. Possibly sample code showing how to programmatically send the requests to the LLM and receive the response.

Task 2: Backend system design

Design a key-value store that can be used as the backend system to be manipulated by the agent. The store could be either in-memory or persist on disk. The table or table-like object should include the following:

Key
Value
Created datetime
Updated datetime

Ideally there would be some way to track changes to the store for auditing purposes

Deliverables:

1. Design of backend key-value store
2. Possibly sample code showing how to create and operate the backend store

Task 3: User interface design

Design the user interface that explains how a user will interact with the system. The user interface could be a command line interface or a web interface.

Deliverables:

1. UI/UX design doc

Task 4: Implementation

Implement a demo of the system. Demo should take user sentences, construct the prompt, send it to the LLM, receive the LLM response, parse it, perform the requested action, then provide some response to the user. Any language can be used.

Deliverables:

1. Code
2. Demo (ideally documented)

Task 5: Testing

Test the system as much as possible. Document the passing and failing test cases. Are failing scenarios handled well? What improvements might be needed.

Deliverables:

1. Test plan doc
2. Test results doc

Suggestions

You might use an LLM api such as Gemini, which will have sample code to make requests using various languages. There are many other LLM options that will probably work.