



Building a Secure MCP Server for Fuel Usage API Access

Ryan McCarthy, Kyle Goodwin, Jacob Musgrave, Christian Cunio, Jackson Haubert, Jacob Chapman,
Department of Computer Science, University of New Hampshire



Introduction

Context :

Palmer Gas and Oil serves thousands of customers, but current systems lack a conversational interface for access to account data. Large Language Models will provide customers with the necessary information through human-like interactions.

Vision :

Create a secure, user friendly, and “exceptionally smart” AI chatbot assistant that will change how customers and representatives interact with the company's technologies.

Value :

Modernize Palmer's customer service platform while creating long-term efficiency for both customers and administrators. A secure and scalable chatbot will reduce call center demands, streamline customer inquiries, and provide accurate information in real-time.

Requirements

Functional Requirements

MCP-Backend Connectivity :

The system must allow the AI model to retrieve real Palmer business data through the MCP Server, a secure middleware layer.

API Proxying :

The MCP Server must proxy requests to the Palmer Backend APIs and return the JSON data to Azure OpenAI in a format that complies with the tool-calling requirements.

Live Connectivity :

The system retrieves real-time fuel inventory data through a tool-calling pipeline between Azure and Palmer APIs.

Non-Functional Requirements

Security :

The MCP Server protects sensitive customer data by acting as an isolation layer between Azure OpenAI and Palmer's backend systems. It uses short-lived, RSA-signed JWTs and a server-to-server shared secret to ensure users can only access their own account data.

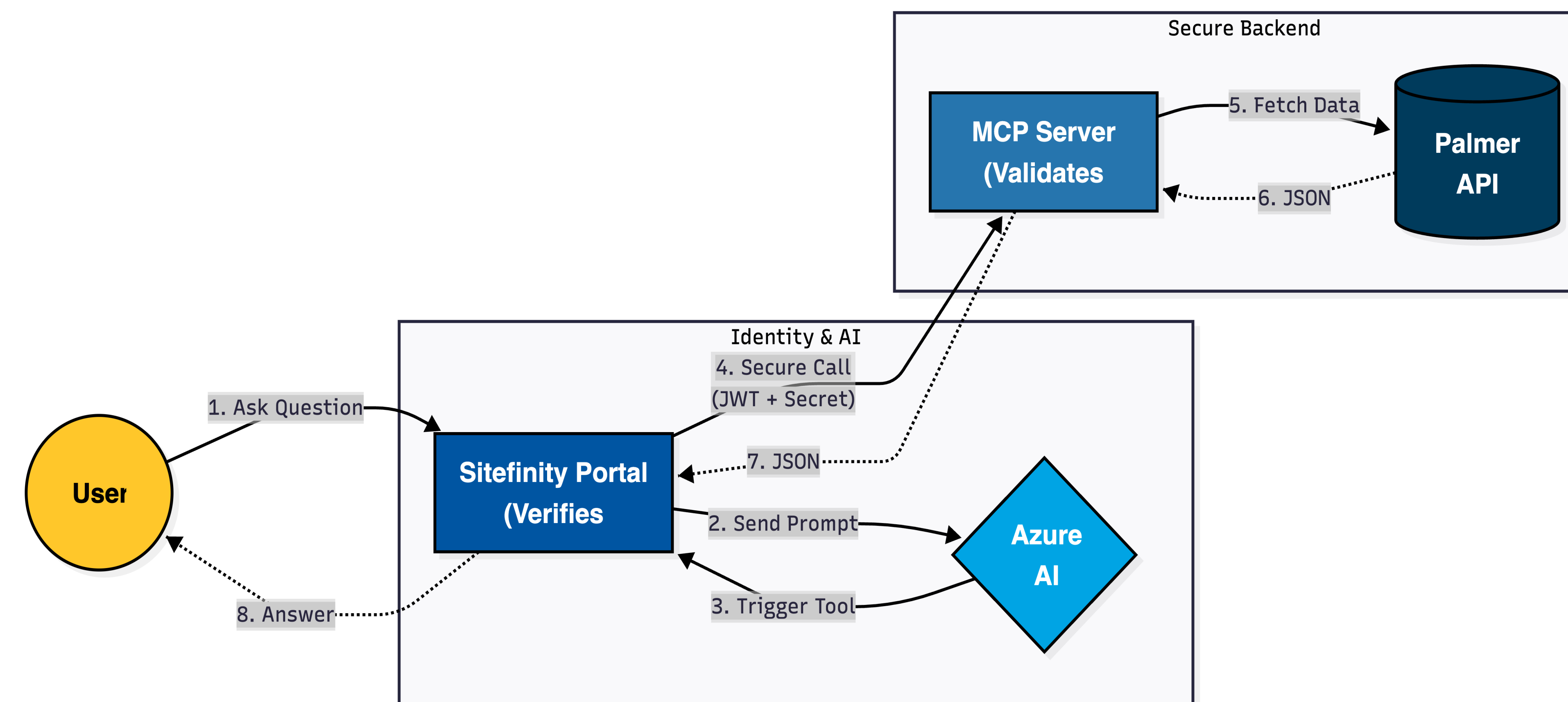
Reliability :

The MCP route used for proxying API calls must respond consistently with accurate HTTP status codes and valid JSON payloads.

Maintainability :

API routing, HttpClient configuration, and endpoint definitions must be modular and easily replaceable as Palmer expands its backend.

Design



Chat UI (Frontend) :

web-based interface embedded into the Palmer website where customers and administrators will submit natural language queries. The UI handles message display and communication but does not perform any business logic.

Azure OpenAI :

The language model interprets user intent and determines what backend data is needed. When needed Azure invokes the MCP tools to retrieve account information. The model converts the JSON responses into human readable answers.

MCP Server :

The MCP server acts as a secure proxy layer between Azure OpenAI and Palmer's live APIs. It validates authentication tokens, maps requests to the correct endpoints, and returns JSON. This is to prevent direct LLM access to sensitive backend data.

Palmer Backend API :

Existing Palmer services provide real business data such as fuel inventory data. The APIs are not directly exposed to the LLM but instead all the requests are mediated through the MCP server.

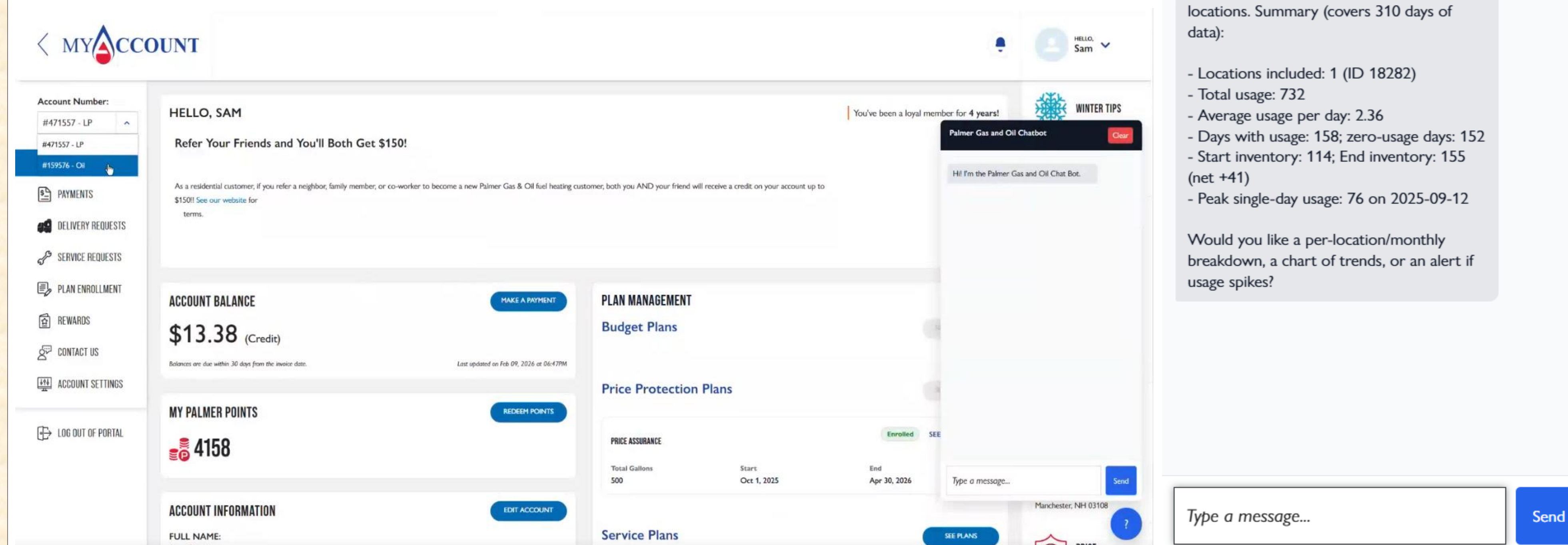
UI

Chat Widget :

Standard chatbot corner widget that opens the chatbot UI and allows for the chat window to be hidden during regular use.

Chat Window :

Directly facilitates conversation between the end-user and the AI chatbot. This is where all interactions between the user and the chatbot occur. The chatbot is limited to only sending replies in this window.



Implementation and Testing

MCP Server Deployment :

A custom .NET service that wraps Palmer API endpoints as callable tools, exposed via Swagger/OpenAPI for automatic discovery.

Integration Pipeline :

Successfully connected Azure OpenAI → MCP Server → Palmer Azure APIs to retrieve live fuel inventory data using structured tool calls.

Rigorous Validation :

Reliability was confirmed through xUnit-based integration tests that verify proper connectivity, JWT authentication, and JSON payload accuracy.

MCP/API Testing :

Verify that all MCP server endpoints are reachable and return correct response data in the proper format from the Palmer Azure APIs

Authentication Testing :

Generate real JWT tokens to authenticate test queries to the MCP server endpoints, verifying correct access levels for specific credentials.

Multi-Device Testing :

Access the live website from a multitude of devices to verify the useability of the AI chatbot across different formats.

Evaluations and Conclusions

Technical Success :

Successfully established end-to-end backend pipeline needed for AI assistant.

Goal Alignment :

The architecture supports real business data and secure tool-calling, proving the viability of the MCP middleware.

Useability :

AI assistant produces meaningful content related to user requests

Acknowledgments

We would like to give a sincere thank you to :

- Our Project Sponsor, Derek Barka, for all of his support and guidance throughout this project.
- Our Project Advisor, David Benedetto, for his advice, feedback, and continued support throughout the development of this project.