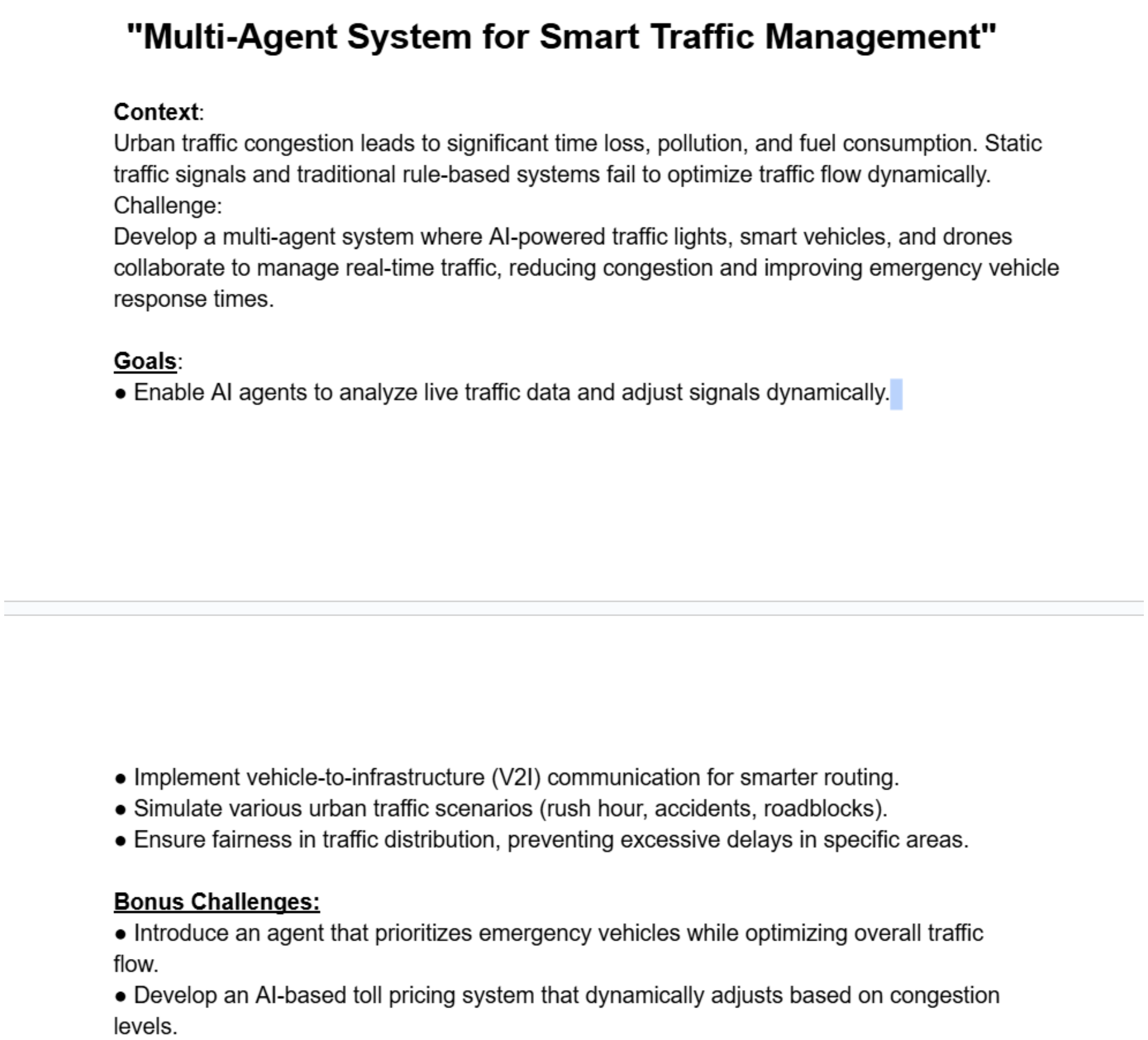
Smart Traffic Management



"Multi-Agent System for Smart Traffic Management"

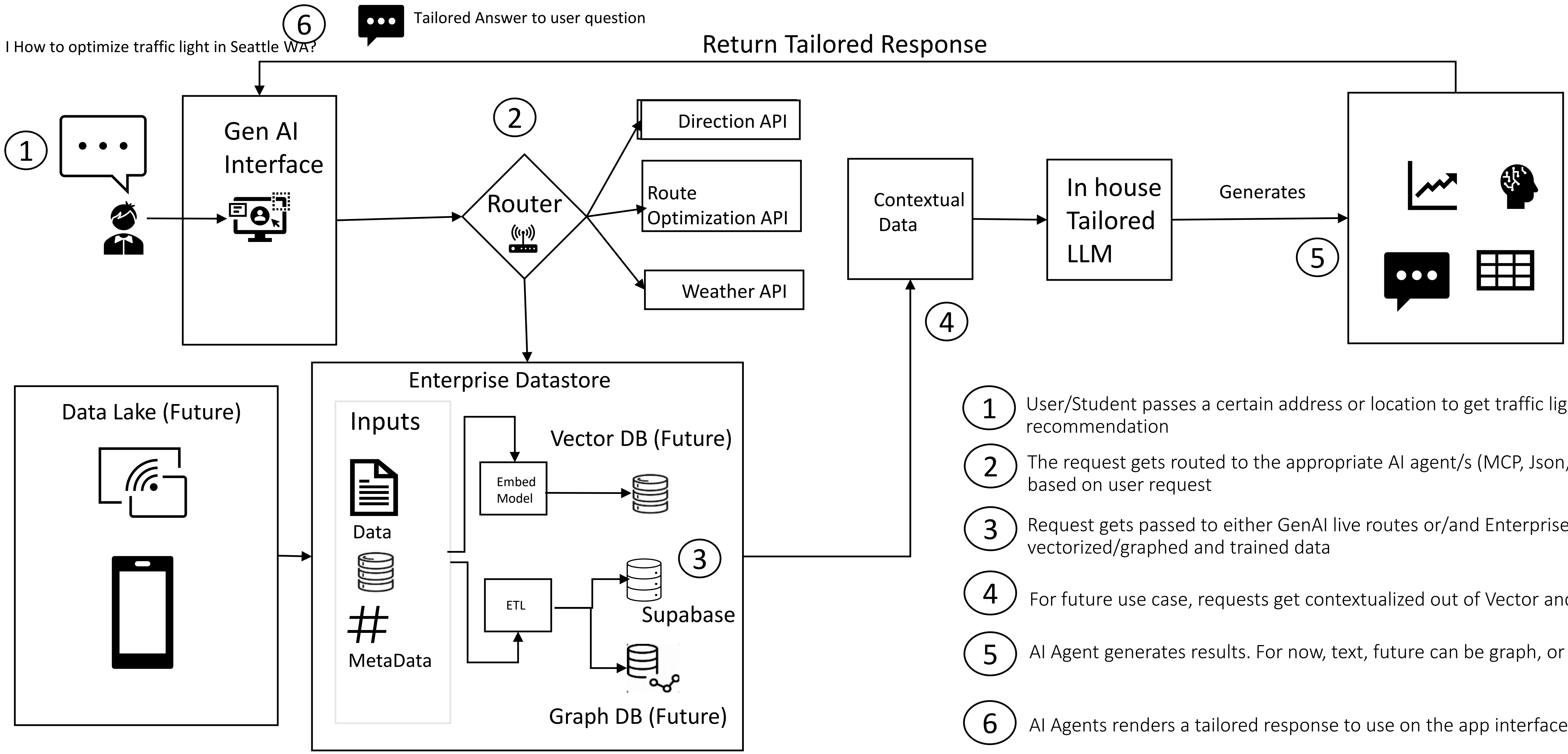
Context:
Urban traffic congestion leads to significant time loss, pollution, and fuel consumption. Static traffic signals and traditional rule-based systems fail to optimize traffic flow dynamically.
Challenge:
Develop a multi-agent system where AI-powered traffic lights, smart vehicles, and drones collaborate to manage real-time traffic, reducing congestion and improving emergency vehicle response times.

Goals:
● Enable AI agents to analyze live traffic data and adjust signals dynamically.

● Implement vehicle-to-infrastructure (V2I) communication for smarter routing.
● Simulate various urban traffic scenarios (rush hour, accidents, roadblocks).
● Ensure fairness in traffic distribution, preventing excessive delays in specific areas.

Bonus Challenges:
● Introduce an agent that prioritizes emergency vehicles while optimizing overall traffic flow.
● Develop an AI-based toll pricing system that dynamically adjusts based on congestion levels.

Smart Traffic Management High-level Process Overview



1 User/Student passes a certain address or location to get traffic light recommendation

2 The request gets routed to the appropriate AI agent/s (MCP, Json, API) based on user request

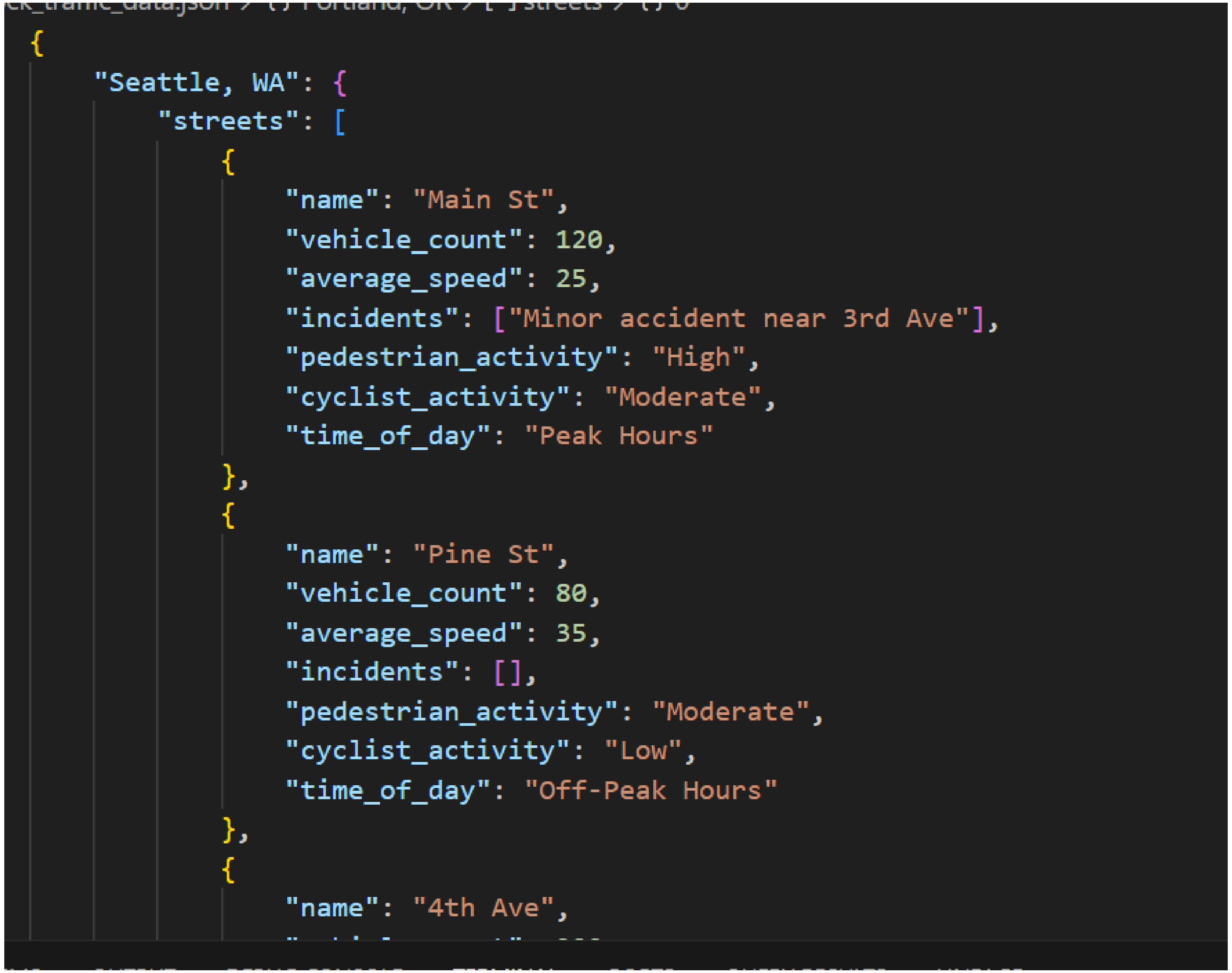
3 Request gets passed to either GenAI live routes or/and Enterprise DB for vectorized/graphed and trained data

4 For future use case, requests get contextualized out of Vector and Graph DB

5 AI Agent generates results. For now, text, future can be graph, or visual

6 AI Agents renders a tailored response to use on the app interface

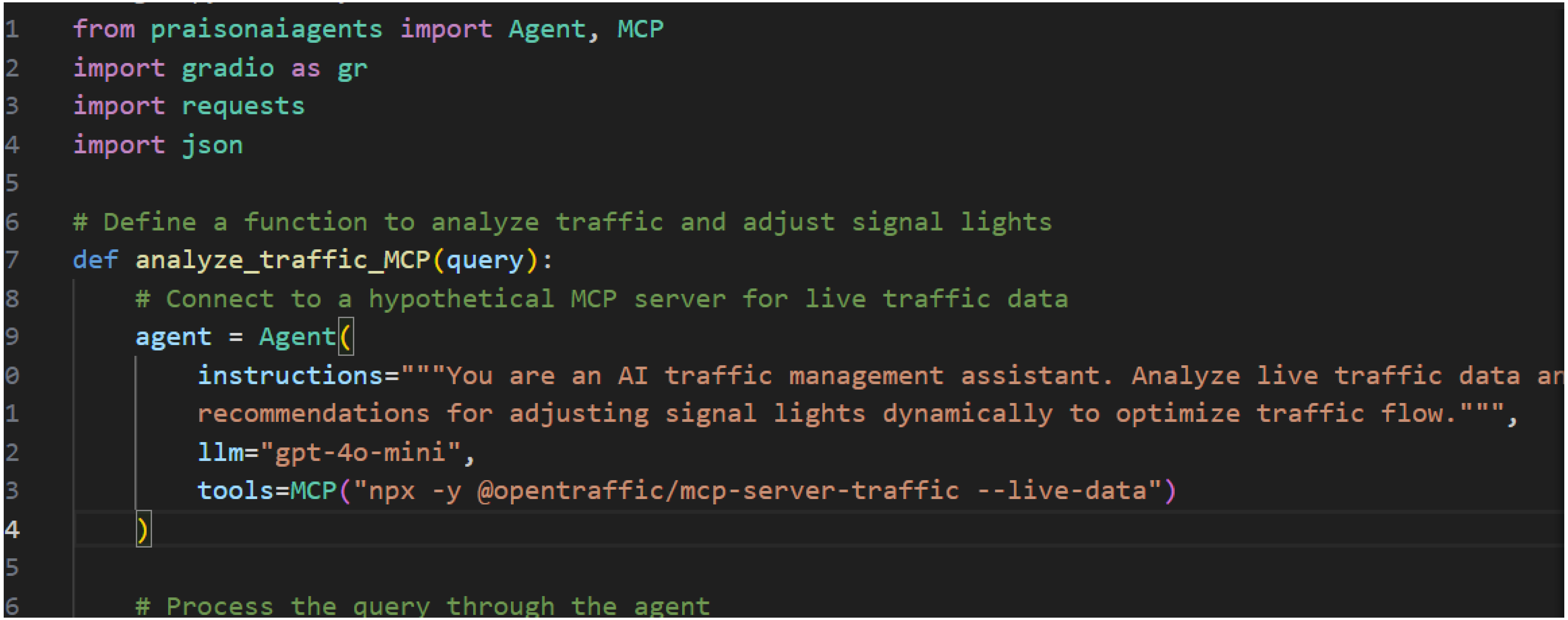
Data Samples



```
ck_traffic_data.json > {} Portland, OR > [ ] streets > {} 0
{
    "Seattle, WA": {
        "streets": [
            {
                "name": "Main St",
                "vehicle_count": 120,
                "average_speed": 25,
                "incidents": ["Minor accident near 3rd Ave"],
                "pedestrian_activity": "High",
                "cyclist_activity": "Moderate",
                "time_of_day": "Peak Hours"
            },
            {
                "name": "Pine St",
                "vehicle_count": 80,
                "average_speed": 35,
                "incidents": [],
                "pedestrian_activity": "Moderate",
                "cyclist_activity": "Low",
                "time_of_day": "Off-Peak Hours"
            },
            {
                "name": "4th Ave",
```

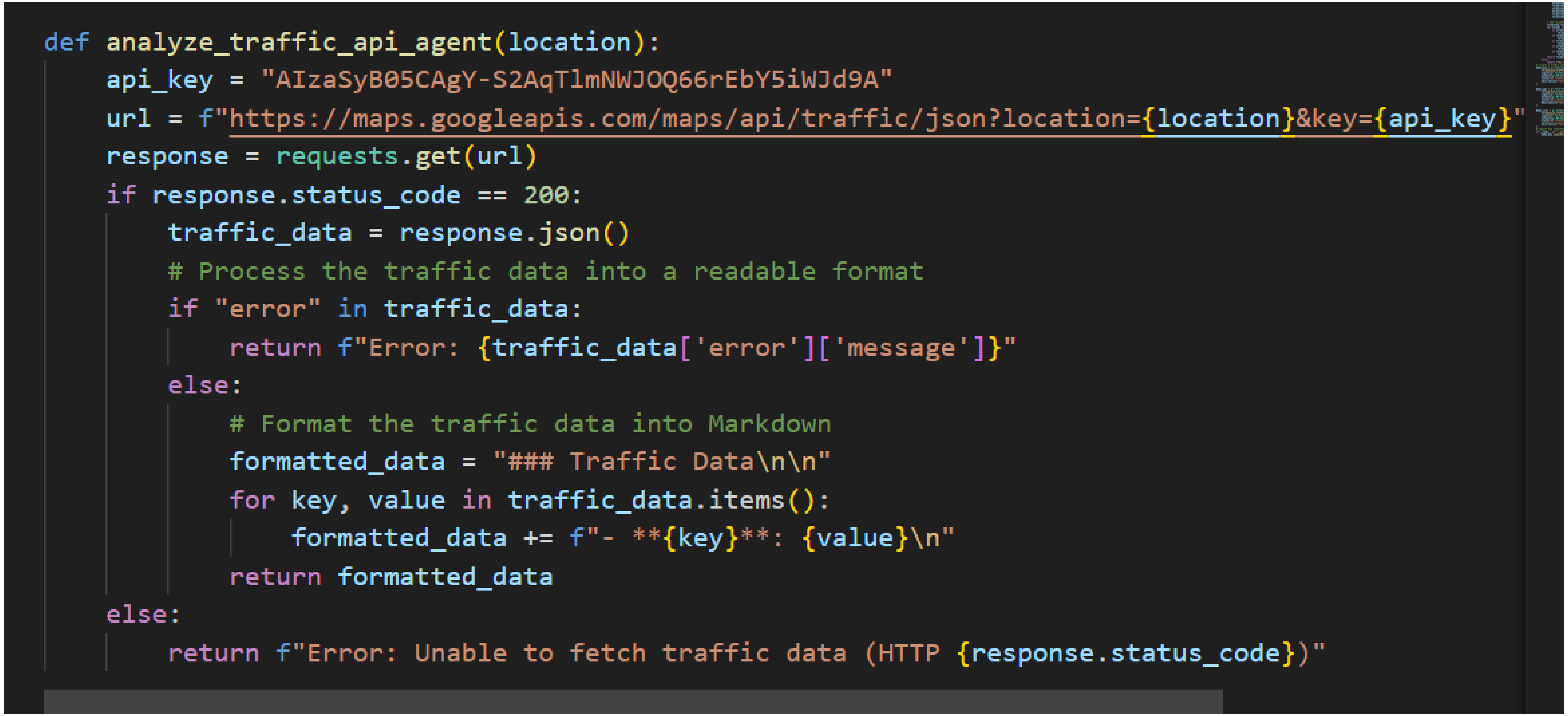
Key Code Snippet - Leveraging MCP

Connecting to MCP server for traffic live data



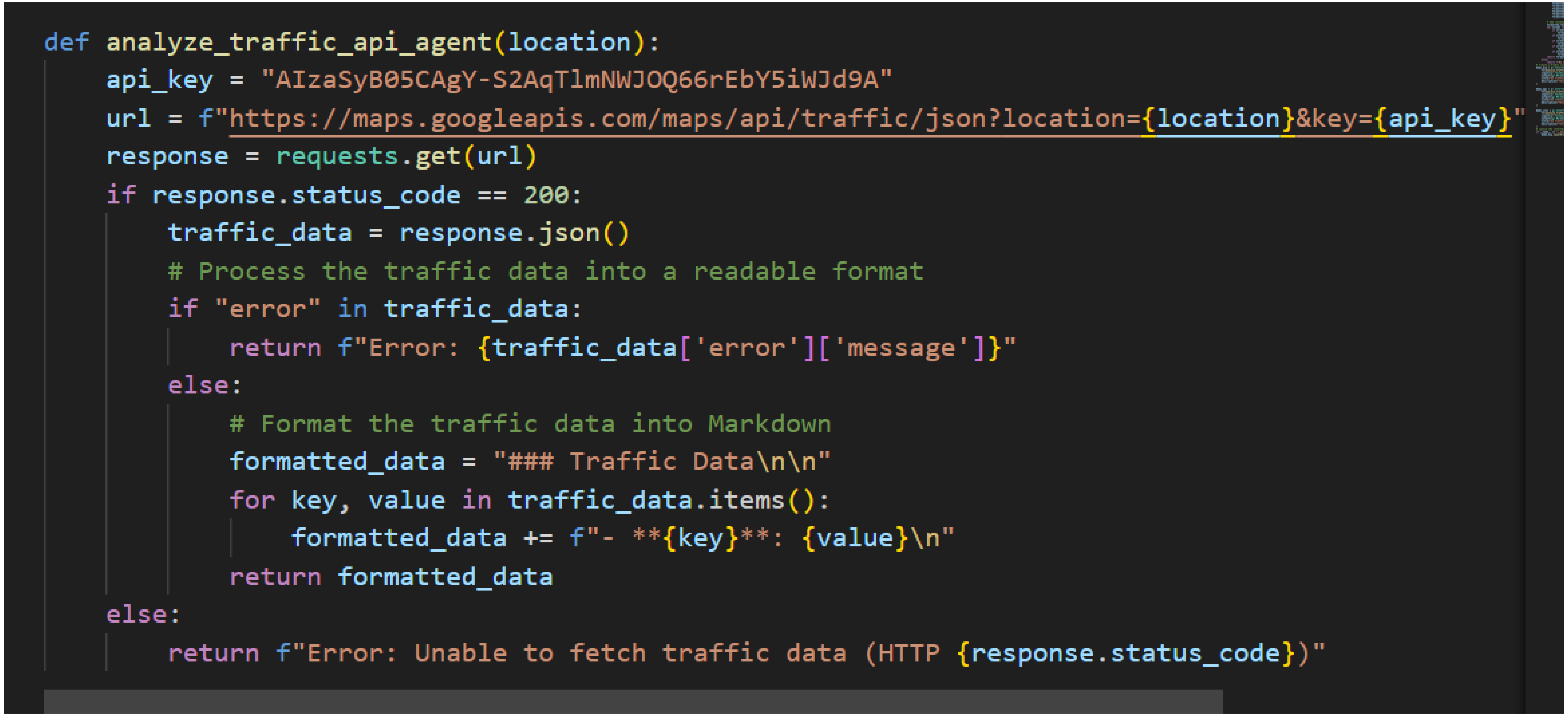
```python
from praisonaiagents import Agent, MCP
import gradio as gr
import requests
import json

# Define a function to analyze traffic and adjust signal lights
def analyze_traffic_MCP(query):
    # Connect to a hypothetical MCP server for live traffic data
    agent = Agent(
        instructions="""You are an AI traffic management assistant. Analyze live traffic data ar
        recommendations for adjusting signal lights dynamically to optimize traffic flow.""",
        llm="gpt-4o-mini",
        tools=MCP("npx -y @opentraffic/mcp-server-traffic --live-data")
    )

    # Process the query through the agent
```

Key Code Snippet – Connecting using API calls



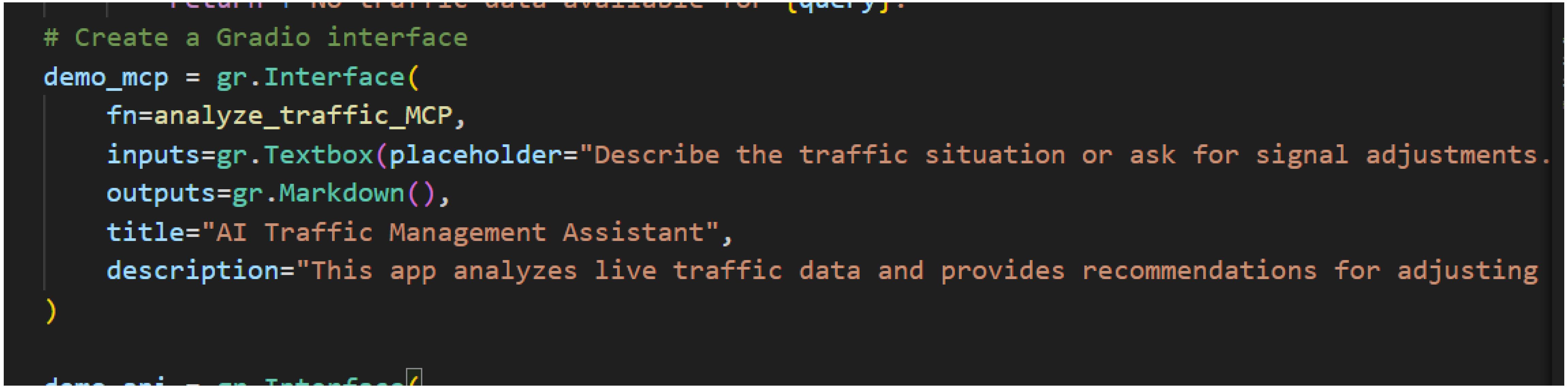
```python
def analyze_traffic_api_agent(location):
    api_key = "AIzaSyB05CAgY-S2AqTlmNWJOQ66rEbY5iWJd9A"
    url = f"https://maps.googleapis.com/maps/api/traffic/json?location={location}&key={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        traffic_data = response.json()
        # Process the traffic data into a readable format
        if "error" in traffic_data:
            return f"Error: {traffic_data['error']['message']}"
        else:
            # Format the traffic data into Markdown
            formatted_data = "### Traffic Data\n\n"
            for key, value in traffic_data.items():
                formatted_data += f"- **{key}**: {value}\n"
            return formatted_data
    else:
        return f"Error: Unable to fetch traffic data (HTTP {response.status_code})"
```

Key Code Snippet – Connecting using API calls



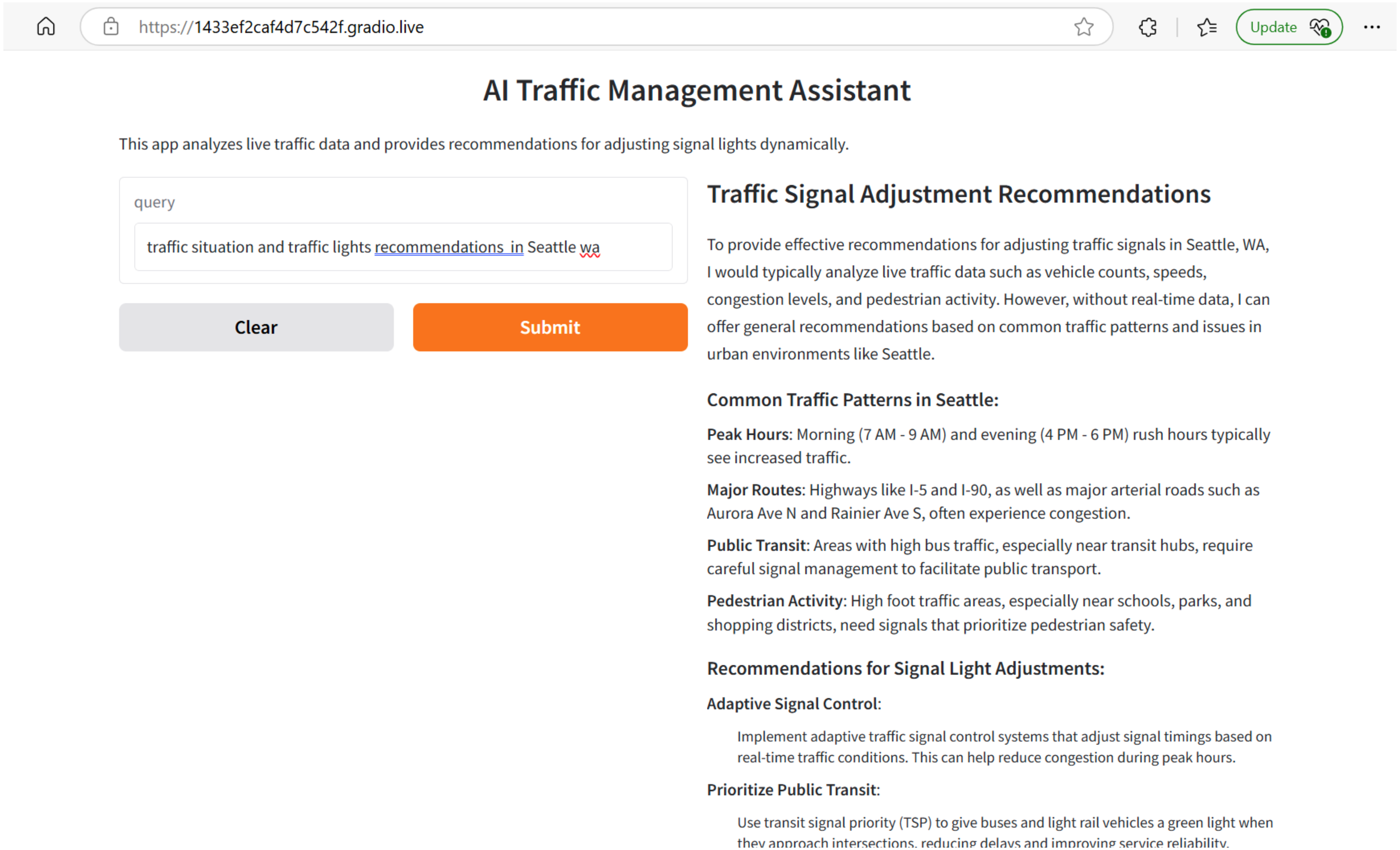
```python
def analyze_traffic_api_agent(location):
    api_key = "AIzaSyB05CAgY-S2AqTlmNWJOQ66rEbY5iWJd9A"
    url = f"https://maps.googleapis.com/maps/api/traffic/json?location={location}&key={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        traffic_data = response.json()
        # Process the traffic data into a readable format
        if "error" in traffic_data:
            return f"Error: {traffic_data['error']['message']}"
        else:
            # Format the traffic data into Markdown
            formatted_data = "### Traffic Data\n\n"
            for key, value in traffic_data.items():
                formatted_data += f"- **{key}**: {value}\n"
            return formatted_data
    else:
        return f"Error: Unable to fetch traffic data (HTTP {response.status_code})"
```

Key Code Snippet – Rendering simple interface



```
        return f"No traffic data available for {query}."
# Create a Gradio interface
demo_mcp = gr.Interface(
    fn=analyze_traffic_MCP,
    inputs=gr.Textbox(placeholder="Describe the traffic situation or ask for signal adjustments.
    outputs=gr.Markdown(),
    title="AI Traffic Management Assistant",
    description="This app analyzes live traffic data and provides recommendations for adjusting
)

demo_api = gr.Interface(
```

Interface Screenshots



https://1433ef2caf4d7c542f.gradio.live

Update

AI Traffic Management Assistant

This app analyzes live traffic data and provides recommendations for adjusting signal lights dynamically.

query

traffic situation and traffic lights recommendations in Seattle wa

Clear

Submit

Traffic Signal Adjustment Recommendations

To provide effective recommendations for adjusting traffic signals in Seattle, WA, I would typically analyze live traffic data such as vehicle counts, speeds, congestion levels, and pedestrian activity. However, without real-time data, I can offer general recommendations based on common traffic patterns and issues in urban environments like Seattle.

Common Traffic Patterns in Seattle:

Peak Hours: Morning (7 AM - 9 AM) and evening (4 PM - 6 PM) rush hours typically see increased traffic.

Major Routes: Highways like I-5 and I-90, as well as major arterial roads such as Aurora Ave N and Rainier Ave S, often experience congestion.

Public Transit: Areas with high bus traffic, especially near transit hubs, require careful signal management to facilitate public transport.

Pedestrian Activity: High foot traffic areas, especially near schools, parks, and shopping districts, need signals that prioritize pedestrian safety.

Recommendations for Signal Light Adjustments:

Adaptive Signal Control:

Implement adaptive traffic signal control systems that adjust signal timings based on real-time traffic conditions. This can help reduce congestion during peak hours.

Prioritize Public Transit:

Use transit signal priority (TSP) to give buses and light rail vehicles a green light when they approach intersections, reducing delays and improving service reliability.

Future Enhancements

Scaling and Optimization Options

1. Horizontal Scaling with load balancer and Elastic Pool technologies needed to scale out the application to handle millions of requests.
2. Vector and Graph DB to optimize AI agent customization per user.
3. Use of Data lake/Delta Lake/OneLake technologies for future advanced analytics and AI/ML model experimentation.
4. Embedding analytics visualization tools for more enhanced and self-service capabilities.
5. Security: for example, Using MFA and data encryption for PII
6. CDN: Zone/Region architecture for optimal availability