# One-Day Tour Planner with LLM Integration

## One-Day Tour Planner with LLM Integration

This project is a conversational assistant designed to help users plan a one-day tour. It leverages a variety of agents to provide users with an itinerary, weather information, and local news updates for a personalized travel experience. The application is built using FastAPI for the backend and Streamlit for the frontend, and it uses an LLM (Large Language Model) for natural language understanding and response generation.

Features:

- Interactive Itinerary Planning: Users can input their preferences, and the system will generate a detailed one-day itinerary for a selected city.
- Weather Information: Provides real-time weather updates for the selected city using the OpenWeatherMap API.
- News Updates: Offers local news highlights from the selected city, using the NewsAPI.
- Memory of Preferences: The system retains user preferences across interactions to improve future suggestions.

Technologies Used:

- FastAPI: For building the backend and API endpoints.
- Streamlit: For creating the frontend interface.
- LLM (Large Language Model): We used the LLaMA 3.2 model for generating conversational responses and planning itineraries.
- Neo4j: A graph database for storing user preferences and conversation history, providing context-aware and personalized responses.
- OpenWeatherMap API: To fetch real-time weather data.
- NewsAPI: To fetch recent news related to the selected city.

Project Structure:

# One-Day Tour Planner with LLM Integration

```
.
 app
    agent
       user_interaction_agent.py   # Handles conversation and delegates tasks to other agents
       itinerary_agent.py         # Generates itinerary based on user preferences
       weather_agent.py           # Fetches weather information
       news_agent.py              # Fetches local news
       memory_agent.py            # Stores and retrieves user preferences in Neo4j
    config.py                     # Contains API keys and configuration settings
    main.py                       # Main FastAPI application entry point
 frontend                         # Streamlit frontend for user interaction
    streamlit_app.py
 requirements.txt                 # Project dependencies
 README.md
```

Setup Instructions:

1. Clone the Repository:

```
git clone https://github.com/your-username/one-day-tour-planner.git
cd one-day-tour-planner
```

2. Install Dependencies:

```
pip install -r requirements.txt
```

3. Configure API Keys:

   Create a config.py file in the app directory and add your API keys for OpenWeatherMap and NewsAPI.

4. Run the Neo4j Database

Ensure Neo4j is installed and running. Configure the database to allow connections on localhost:7687.

5. Start the FastAPI Server:

```
uvicorn app.main:app --reload
```

6. Start the Streamlit Frontend:

```
streamlit run frontend/streamlit_app.py
```

Agents:

1. User Interaction Agent: Handles conversation and delegates tasks.

2. Itinerary Generation Agent: Uses the LLaMA model to generate a one-day itinerary.

3. Weather Agent: Retrieves weather data from OpenWeatherMap.

4. News Agent: Fetches recent news from NewsAPI.

5. Memory Agent: Stores user preferences and retrieves them for a personalized experience.

Databases:

The Neo4j graph database is used to store user preferences and conversation history, creating a context-aware and personalized experience.

LLM Integration:

This project uses the LLaMA 3.2 language model for natural language processing, invoked via a local CLI.

APIs:

# One-Day Tour Planner with LLM Integration

1. OpenWeatherMap API: Provides real-time weather information.

2. NewsAPI: Retrieves recent news for the selected city.

Known Issues:

1. Path configuration for the LLaMA model might need adjustments.

2. Neo4j connection issues may occur if the database configuration is incorrect.

Future Improvements:

1. Add support for multi-day itineraries.

2. Enhanced memory management and optimization.

3. More robust error handling.

Contributions:

Contributions are welcome! Please fork the repository, create a branch, and submit a pull request.