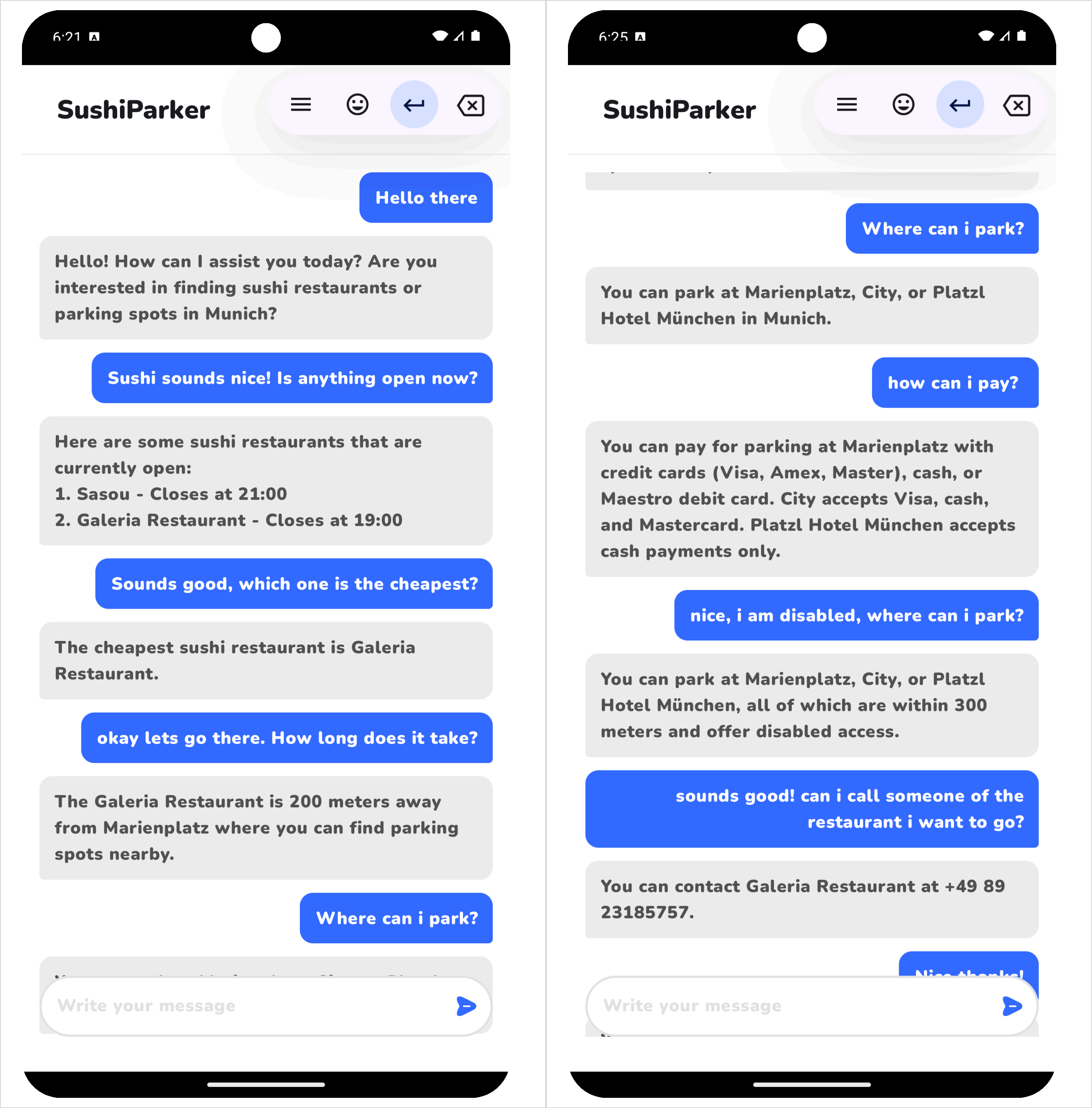
Munich Sushi & Parking Assistant

An AI-powered assistant that helps users find sushi restaurants and parking spots in Munich. The assistant uses GPT-3.5-turbo to understand user queries and provide relevant information from a mock database.

Android Jetpack Compose App

Inspired by HariAgus/TokenApp



Hello there

Hello! How can I assist you today? Are you interested in finding sushi restaurants or parking spots in Munich?

Sushi sounds nice! Is anything open now?

Here are some sushi restaurants that are currently open:
1. Sasou - Closes at 21:00
2. Galeria Restaurant - Closes at 19:00

Sounds good, which one is the cheapest?

The cheapest sushi restaurant is Galeria Restaurant.

okay lets go there. How long does it take?

The Galeria Restaurant is 200 meters away from Marienplatz where you can find parking spots nearby.

Where can i park?

Where can i park?

You can park at Marienplatz, City, or Platzl Hotel München in Munich.

how can i pay?

You can pay for parking at Marienplatz with credit cards (Visa, Amex, Master), cash, or Maestro debit card. City accepts Visa, cash, and Mastercard. Platzl Hotel München accepts cash payments only.

nice, i am disabled, where can i park?

You can park at Marienplatz, City, or Platzl Hotel München, all of which are within 300 meters and offer disabled access.

sounds good! can i call someone of the restaurant i want to go?

You can contact Galeria Restaurant at +49 89 23185757.

Nice, thanks!

Setup

1. Clone the repository
2. Create a virtual environment and activate it:

```
python -m venv .venv
source .venv/bin/activate  # Linux/Mac
.venv\Scripts\activate     # Windows
```

3. Install requirements

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

4. Install the project in editable mode:

```
pip install -e .
```

5. Create a `.env` file in the root directory with your OpenAI API key:

```
OPENAI_API_KEY=your_api_key_here
```

Running the Project

There are three ways to interact with the assistant:

1. Jupyter Notebook

Explore the detailed implementation and test different scenarios:

```
application.ipynb
```

2. Terminal Chat

Start an interactive chat session in the terminal:

```
python src/main_terminal.py
```

3. FastAPI Endpoint

Run a local API server:

```
python src/main_api.py
```

The API will be available at `http://localhost:8000`

- Swagger UI: `http://localhost:8000/docs`

Features

- Dynamic function calling with instructor
- Checks users intend and selects response model
- Support for both sushi restaurant and parking queries (can switch in one session)
- Information about:
  - Restaurant locations, hours, and reviews
  - Parking spot availability and pricing
  - Distance from current location
  - Payment methods

Project Structure

- `application.ipynb`: Detailed walkthrough of the assistant's functionality
- `src/main_terminal.py`: Terminal-based chat interface
- `src/main_api.py`: FastAPI server implementation
- `src/services/`: Core business logic and data handling
- `src/models/`: Pydantic models for type safety
- `src/data/`: Mock database JSON files