AI-Powered Pizza Ordering Assistant

Welcome to the AI-Powered Pizza Ordering Assistant — a conversational AI chatbot that helps users place customized pizza orders through a simple, natural language interface powered by the Gemini API. The assistant supports voice interaction and delivers structured order summaries in multiple formats.

Project Overview

This project demonstrates how to build an AI-driven pizza ordering assistant using Python and Gradio for a web-based chat interface. Users can:

- Interact naturally via text or voice
- Specify dietary preferences (e.g., vegan, gluten-free)
- Customize pizza orders with dynamic menu options
- Receive a final JSON summary of their order for confirmation
- Have orders saved in JSON, CSV, XML, and SQLite formats

The system architecture emphasizes modular design, clear separation of frontend and backend, and easy extensibility.

Repository Architecture

```
AI_assistant_project/
│
├── backend/                          # Core backend logic and data handling
│   ├── dialog_agent.py               # Conversation flow and AI interaction
logic
│   ├── gemini_client.py              # Gemini API communication client
│   ├── order_saver.py                # Order persistence in multiple formats
│   ├── utils.py                      # Utility functions (parsing, validation,
formatting)
│   ├── menu.json                    # Pizza menu data file (JSON)
│   ├── request_counter.txt          # Tracks number of order sessions
│
├── frontend/                         # Frontend launcher scripts for user
interaction
│   ├── run_gradio.py                 # Launches the Gradio text-based web UI
│   └── voice_chatbot.py              # Voice input/output interface
│
├── docs/                            # Documentation, flowcharts, and screenshots
│   ├── flowchart/                   # Flowchart images and source files
│   ├── screenshots/                 # UI and conversation screenshots
```

```
│   ├── Mermaidlive.png            # Mermaid flowchart image
│   ├── README.md                  # Detailed project documentation
│   └── report.pdf                 # Full project report document
│
├── tests/                        # (Optional) Automated test scripts
│
├── .env                          # Environment variables (API keys)
[gitignored]
├── .gitignore                    # Specifies files/folders to ignore in Git
├── requirements.txt              # Python package dependencies
├── chat_transcript.txt           # Sample chat log from an interaction
├── screenshot.png                # General UI screenshot for documentation
└── README.md                    # This file — project overview and setup
```

Features

- **Natural language interaction** via Gradio web chat and voice assistant
- AI-driven order handling with Gemini API integration
- Dynamic menu navigation supporting dietary restrictions
- Final order export in JSON, CSV, XML, and SQLite for persistence
- Voice-enabled interface for accessibility and hands-free use
- Lightweight architecture with no mandatory database dependency

---

Setup Instructions

1. **Clone the Repository**

```
git clone <your_repo_link>
cd AI_assistant_project
```

1. **Create and Activate Python Virtual Environment**

```
python -m venv venv
source venv/bin/activate         # Windows: venv\Scripts\activate
```

1. **Install Required Dependencies**

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

1. **Configure Gemini API Key**

Create a `.env` file in the project root directory:

```
GEMINI_API_KEY="Insert_Your_API_Key_Here"
```

1. **Run the Gradio Chat Interface**

```
python frontend/run_gradio.py
```

Open your browser and navigate to: http://127.0.0.1:7860

1. **(Optional) Run the Voice Assistant**

```
python frontend/voice_chatbot.py
```

How It Works

- User inputs a pizza order via text or voice.
- `dialog_agent.py` handles conversation flow, building prompts for the Gemini API.
- `gemini_client.py` sends prompts and retrieves responses from Gemini.
- Responses are shown in Gradio UI or voiced back through `voice_chatbot.py`.
- Once the order is finalized, `order_saver.py` exports the data into multiple formats.
- `utils.py` supports input parsing and order validation.
- The system tracks session counts in `request_counter.txt`.

Documentation & Resources

- Flowcharts and technical diagrams are stored under `docs/flowchart/`
- Screenshots of the chat flow and UI are available in `docs/screenshots/`
- A detailed project report is provided in `docs/report.pdf`
- Example chat logs can be found in `chat_transcript.txt`

Future Improvements

- Integrate payment processing and user authentication
- Add multi-language support
- Implement persistent database backend (e.g., PostgreSQL)
- Enhance voice assistant with advanced NLP capabilities

License

This project is licensed under the MIT License.

Thank you for exploring the AI-Powered Pizza Ordering Assistant!\ Feel free to open issues or submit pull requests for improvements.