Azure OpenAI Medical Phone Appointments

An intelligent phone-based appointment scheduling system powered by Azure OpenAI GPT-4 and Twilio, featuring real-time voice interactions and natural language processing.

Features • Architecture • Getting Started • Configuration • Development • Contributing

Features

- Real-time voice interactions using GPT-4
- Phone-based appointment scheduling
- Natural language understanding
- Automated voice responses
- Appointment information collection
- SMS confirmations
- Real-time availability checks
- Multi-language support

Architecture

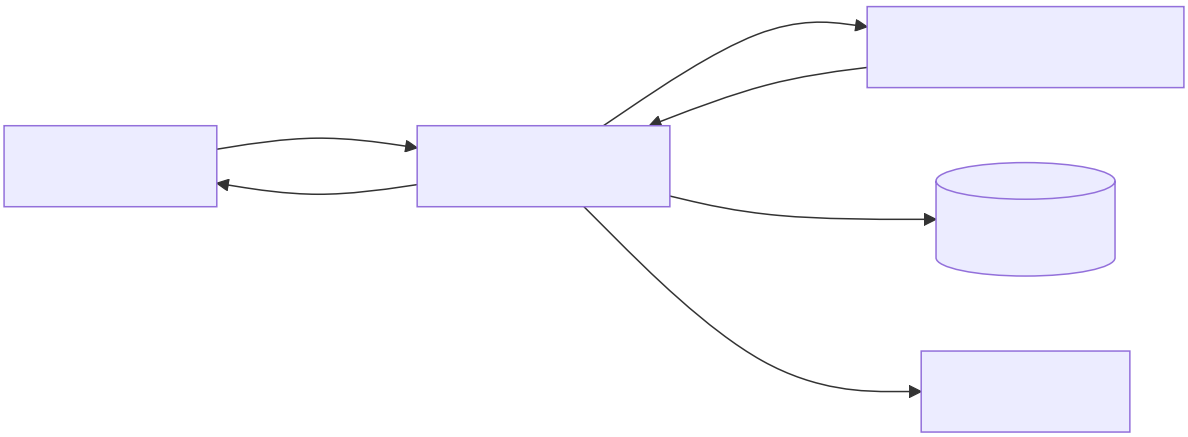

Figure 1: diagram

Data Flow

Incoming Call

Tech Stack

```
Backend Framework: FastAPI
AI Model: Azure OpenAI GPT-4o-Realtime
Voice Services: Twilio
Database: Azure SQL
Runtime: Python 3.12
WebSocket: asyncio/websockets
Deployment: Azure App Service
```

Quick Start

1. Clone the repository

```
git clone https://github.com/yourusername/azure-openai-phone-appointments.git
cd azure-openai-phone-appointments

## Set up environment variables
cp .env.example .env
```

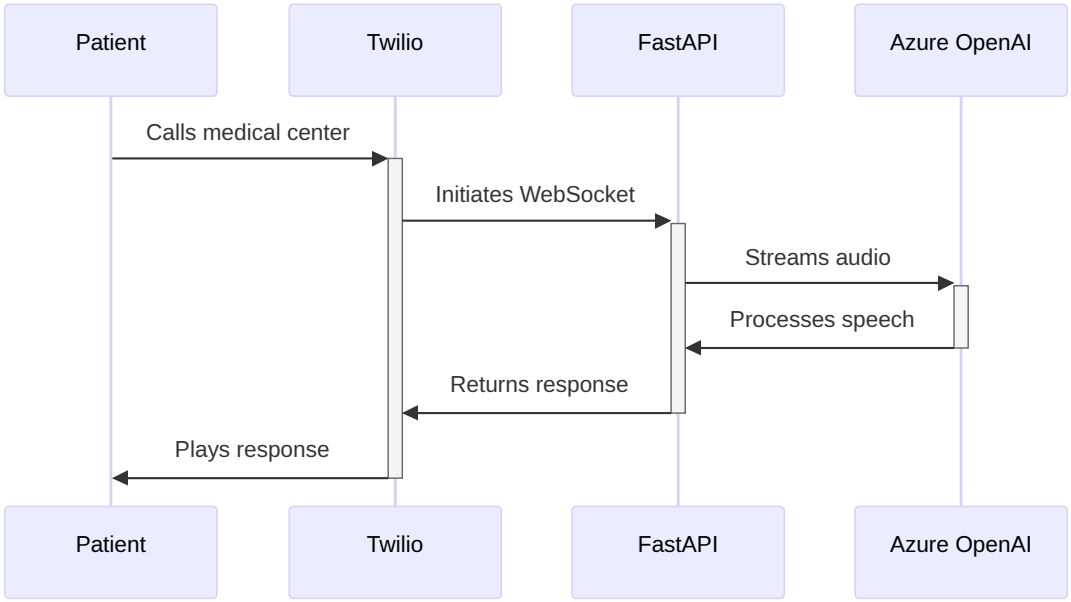

Figure 2: diagram

```
## Configure the following in .env:
OPENAI_API_KEY=your_azure_openai_key
OPENAI_API_ENDPOINT=your_azure_endpoint
PORT=5050

## 3. Install dependencies
pip install -r requirements.txt

## 4. Run the application
python app.py
```

Configuration

The system uses several key configurations:

```
SYSTEM_MESSAGE = """
You are an AI assistant acting as a medical center receptionist.
Tasks:
1. Greet callers
2. Collect appointment information:
   - Doctor's name
   - Caller's name
   - Phone number
   - Appointment date/time
"""
```

API Endpoints

Security Features

- Secure WebSocket connections
- API key authentication

- Input validation
- HIPAA-compliant communications

WebSocket Communication

The application handles bi-directional audio streaming:

- Incoming audio from Twilio
- Processed responses from Azure OpenAI
- Real-time speech detection
- Low-latency voice interactions

Key Components

1. Call Handler

```
@app.api_route("/incoming-call", methods=["GET", "POST"])
async def handle_incoming_call(request: Request):
    # Handles incoming Twilio calls

## 2. Media Stream
```python
@app.websocket("/media-stream")
async def handle_media_stream(websocket: WebSocket):
    # Manages real-time audio streaming
```

Contributing

1. Fork the repository
2. Create a feature branch
3. Commit changes
4. Push to the branch
5. Open a Pull Request

Support

For support, please open an issue in the repository or contact

License

This project is licensed under the MIT License - see the LICENSE file for details.

Explore AI Resources

- **AiMegos**: Discover Vamsi Kethu's AI Leadership
- **AgentsInfinite**: Where Vision Meets Velocity

Support

- Email: contactus@agentsinfinite.com
- Discord: Join our server
- Twitter: @agentsinfinite

Made with by VamK2 @ Aimegos | AgentsInfinite
Website • Documentation • Blog
```