Documentation
ChatBot Q&A on Medical Services

Adi Prager

April 16, 2025

1 Project Overview

A stateless microservice-based chatbot that provides personalized Q&A on medical services offered by Israeli HMOs: Maccabi, Meuhedet, and Clalit. It collects user information (name, ID, HMO, tier, etc.) through a conversation, then answers questions based on a knowledge base of HTML files. All session data is stored on the client (frontend).

2 Repository Structure

```
ChatBot_Q-A_on_Medical_Services/
 .venv/
 backend/
  app/
   data/
    phase2_data/
    alternative_services.html
    communication_clinic_services.html
    dentel_services.html
    optometry_services.html
    pragrency_services.html
    workshops_services.html
   services/
    info_collector.py
    llm_client.py
    qa_handler.py
   utils/
    chatbot.log
    html_loader.py
    html_parser.py
    logger.py
    test_html_read.py
    validators.py
   main.py
   api.py
  requirements.txt
 frontend/
  app.py
  requirements.txt
 .env
 readme.md
```

3 Requirements

- Python 3.9+ recommended
- Azure OpenAI account and deployment

- (Optional) `python-dotenv` for environment variable loading

3.1 Environment Variables

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

```
AZURE_OPENAI_API_KEY=<your_azure_api_key>
AZURE_OPENAI_API_BASE=https://<your_resource_name>.openai.azure.com/
AZURE_OPENAI_API_VERSION=2024-02-15-preview
AZURE_DEPLOYMENT_NAME=<azure_deployment_name>
```

Ensure these match your Azure OpenAI resource settings.

4 Installation and Setup

4.1 Clone the Repository

```
git clone https://github.com/adidereviani/ChatBot_Q-A_on_Medical_Services.git
cd ChatBot_Q-A_on_Medical_Services
```

4.2 Create a Python Virtual Environment (optional)

```
python -m venv venv
source venv/bin/activate # On macOS/Linux
# or venv\Scripts\activate # On Windows
```

4.3 Install Backend Dependencies

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

4.4 Install Frontend Dependencies

```
cd ../frontend
pip install -r requirements.txt
```

4.5 Configure Environment Variables

Edit the `.env` file or set them manually for your environment.

5 Running the Application

5.1 Start the Backend

From the project root:

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

This launches FastAPI on `http://127.0.0.1:8000`.

5.2 Start the Frontend (Streamlit Example)

In another terminal:

```
cd frontend
streamlit run app.py
```

By default, it runs at `http://localhost:8501`.

6 Usage Flow

6.1 Info Collection Phase

- The assistant asks how it can help. If relevant, it collects user details (name, ID, gender, age, HMO, membership tier, etc.) via LLM prompts (not forms).
- Once user details are confirmed, it transitions to Q&A.

6.2 Q&A Phase

- The user asks about treatments or services.
- The system consults the knowledge base HTML files in `phase2_data` folder to answer.

6.3 Client-Side Session

- All conversation history and user data stay in `st.session_state` (Streamlit).
- Each `/chat` request includes the entire conversation, so the server is stateless.

7 Key Files

7.1 backend/app/main.py

Creates the FastAPI application and includes the API router.

7.2 backend/app/api.py

Defines `/chat` endpoint, receives phase, `user_info`, `messages`; calls `ask_llm`.

7.3 backend/app/services/llm_client.py

- Loads environment variables
- Builds system prompts (info vs Q&A)
- Calls Azure OpenAI
- Returns the LLM response

7.4 backend/app/utils/html_loader.py

Loads the HTML files from `phase2_data` folder.

7.5 backend/app/utils/logger.py

Configures a file logger to capture requests, responses, and errors.

7.6 frontend/app.py

- Streamlit UI
- Prompts user, keeps session state
- Calls `/chat` endpoint

8 Logging and Error Handling

8.1 Logging

- `logger.py` sets up a file handler (e.g., `chatbot.log`)
- `api.py` logs incoming requests, final responses, and exceptions

8.2 Error Handling

- `try/except` in `api.py`
- `HTTPException` for user errors (400, 422)
- 500-level for server errors

9 Concurrency and Statelessness

- The backend is stateless (no session in memory).
- Each request includes all user data and conversation history.
- FastAPI handles concurrency automatically.

10 Multi-language Support

- The system prompt instructs responding in Hebrew or English based on user input language.
- The knowledge base is mostly Hebrew, but the LLM can translate or respond in English if asked.