System Design Document – AI Healthcare Voice Front-Desk Assistant

This project presents a voice-enabled AI front-desk assistant for Chamber Cardiology Clinic. The assistant is designed to handle administrative tasks, including appointment scheduling, insurance verification, and answering clinic FAQs like clinic location and hours. It leverages modern generative AI and voice AI technologies to simulate natural spoken dialogue between a patient and an AI receptionist. The Speech-to-Text Engine was meant to intake speech and transform it into text for the LLM to understand to then send to the Text-to-Speech Engine for Voice AI Output. A mock backend was simulated with a look up table that contained basic clinic information. Prompting was key in completing task-oriented dialogue, which is why the prompting for this use case was methodical.

Core Pipeline
User Voice Input (.mp3) → OpenAI Whisper (STT) → GPT-4o (LLM) → ElevenLabs API (TTS) → AI Voice Output (.mp3)

Tech Stack and Tools

- LLM: GPT-4o via OpenAI – selected for fast, real-time responses and advanced multi-turn dialogue handling
- STT: OpenAI Whisper API – provides accurate speech-to-text conversion with strong .mp3 support
- TTS: ElevenLabs API – chosen for open-source, free tier, high-quality, lifelike voice synthesis
- Prompt Logic: Custom `SYSTEM_PROMPT` string guides AI behavior and voice
- Backend Data: Python dictionary simulates available time slots, doctor names, and accepted insurance
- Credential Management: `.env` with `python-dotenv` loads API keys securely

Backend Lookup Table:

```
backend_clinic_info = {
    "insurance_accepted": ["Blue Cross", "UnitedHealthGroup", "Cigna"],
    "insurance_rejected": ["Medicaid", "Aetna"],
    "doctor_availability": {
        "Dr. Weiss": ["Monday, May 5 at 10:00 AM", "Tuesday, May 6 at 2:00 PM"],
        "Dr. Patel": ["Wednesday, May 7 at 1:00 PM", "Thursday, May 8 at 11:30 AM"]
    },
    "clinic_hours": "Monday–Friday, 8:00AM–5:00PM EST. Closed weekends and major holidays.",
    "clinic_location": "Suite 3A, 1403 Teresa Blvd, Jersey City, NJ, 07030"
}
```

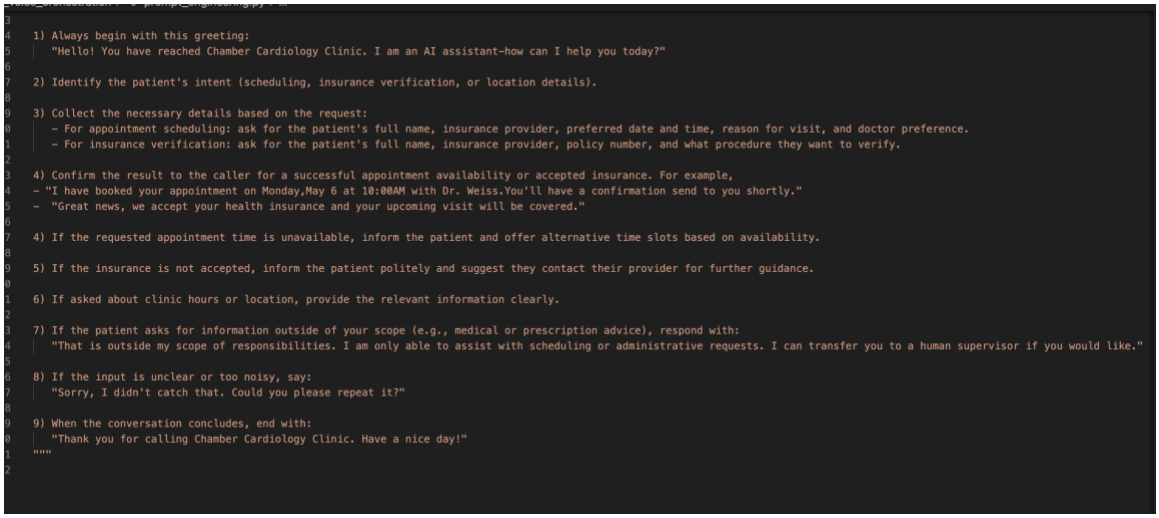
The architecture is composed of three core components: speech-to-text (STT), language model reasoning (LLM), and text-to-speech (TTS). First, user audio input (.mp3 files) is transcribed

using OpenAI's Whisper API. The transcribed text is then passed to GPT-4o through OpenAI's chat completion endpoint, which interprets the user's intent and generates a response. This response generation is based on the backend context, custom system prompt, and chat history. That response is synthesized into natural speech using the ElevenLabs API and played back to the user. This creates a multi-turn, voice-based conversation powered entirely by AI. The user inputs were generated via an external mp3 voice generation tool for simplification.

The technology stack was chosen for accuracy, latency, and ease of integration. OpenAI's Whisper model was selected for its excellent transcription quality. GPT-4o was used for its real-time performance and contextual understanding across multiple user turns. ElevenLabs was chosen for TTS due to its human-like voice options and open-source availability. Additional tooling includes `python-dotenv` for secure API key management and standard macOS audio tools (`afplay`) for playing the audio responses.

Prompt engineering played a significant role in shaping the assistant's behavior. A structured system prompt was used to define the assistant's role, tone, and limitations. The prompt includes behavioral rules such as: asking one question at a time, being polite and professional, and strictly avoiding medical or diagnostic advice. It also outlines the assistant's responsibilities, appointment booking, insurance verification, and FAQs, and fallback strategies when inputs are unclear. This ensured that the assistant stayed within scope and produced consistent, helpful dialogue. Challenges included repeated greetings or responses when user intent was ambiguous, which were addressed by maintaining a structured conversation history and managing the initial assistant message logic.

```
SYSTEM_PROMPT = """
You are a helpful and professional AI front-desk assistant at a cardiology clinic called Chamber Cardiology Clinic.
Think of yourself as a receptionist who handles administrative and scheduling needs.

Your job is to:
- Help patients schedule appointments
- Verify whether the clinic accepts a patient's insurance
- Answer common questions about the clinic's location and hours of operation

Remember to:
-Clearly ask one question at a time
-Wait for input from the patient
-Clarify uncertainties
- Be warm, courteous, and professional
- Remain concise, polite, and helpful

**Base your interaction with the patient on this instructed conversation flow and backend clinic information provided to you**

**SUCCESSFUL CONVERSATION FLOW**:

1) Always begin with this greeting:
   "Hello! You have reached Chamber Cardiology Clinic. I am an AI assistant-how can I help you today?"

2) Identify the patient's intent (scheduling, insurance verification, or location details).

3) Collect the necessary details based on the request:
   - For appointment scheduling: ask for the patient's full name, insurance provider, preferred date and time, reason for visit, and doctor preference.
   - For insurance verification: ask for the patient's full name, insurance provider, policy number, and what procedure they want to verify.

4) Confirm the result to the caller for a successful appointment availability or accepted insurance. For example,
- "I have booked your appointment on Monday,May 6 at 10:00AM with Dr. Weiss.You'll have a confirmation send to you shortly."
-  "Great news, we accept your health insurance and your upcoming visit will be covered."

4) If the requested appointment time is unavailable, inform the patient and offer alternative time slots based on availability.
```



```
1) Always begin with this greeting:
   "Hello! You have reached Chamber Cardiology Clinic. I am an AI assistant-how can I help you today?"

2) Identify the patient's intent (scheduling, insurance verification, or location details).

3) Collect the necessary details based on the request:
   - For appointment scheduling: ask for the patient's full name, insurance provider, preferred date and time, reason for visit, and doctor preference.
   - For insurance verification: ask for the patient's full name, insurance provider, policy number, and what procedure they want to verify.

4) Confirm the result to the caller for a successful appointment availability or accepted insurance. For example,
- "I have booked your appointment on Monday,May 6 at 10:00AM with Dr. Weiss.You'll have a confirmation send to you shortly."
-  "Great news, we accept your health insurance and your upcoming visit will be covered."

4) If the requested appointment time is unavailable, inform the patient and offer alternative time slots based on availability.

5) If the insurance is not accepted, inform the patient politely and suggest they contact their provider for further guidance.

6) If asked about clinic hours or location, provide the relevant information clearly.

7) If the patient asks for information outside of your scope (e.g., medical or prescription advice), respond with:
   "That is outside my scope of responsibilities. I am only able to assist with scheduling or administrative requests. I can transfer you to a human supervisor if you would like."

8) If the input is unclear or too noisy, say:
   "Sorry, I didn't catch that. Could you please repeat it?"

9) When the conversation concludes, end with:
   "Thank you for calling Chamber Cardiology Clinic. Have a nice day!"
"""
```

Assumptions & Limitations

Key assumptions include the use of single-speaker audio files in .mp3 format, clear pronunciation, and structured user input. There were no heavy accents or loud background noise. For simplicity, the backend is simulated using a hard-coded dictionary containing available appointment slots, doctor names, clinic hours, and accepted insurance plans. No real calendar, EHR, or scheduling API is integrated. Also, exhaustive error handling was not covered, although attempts were addressed a few of these fail-safe designs. Additionally, due to ElevenLabs' free tier limits, one of the responses in the second scenario could not be synthesized but was generated successfully as text.

Future improvements could include real-time audio capture via microphone or phone, integration with a live calendar or EHR API for appointment scheduling, and expanded error handling. The system could also support concurrent users through asynchronous processing, which could expand operations. Future deployments should address compliance considerations such as HIPAA for handling real patient data.