Voice-Driven Chatbot: Summary, Setup, and Design

Summary of Findings:

I have created a voice-driven chatbot application with the following key components:

1. **Functionality:** The chatbot listens to user voice input, converts it to text, and provides predefined responses based on keyword matching from a set of personal narrative responses.

2. **Technology Stack:**

- Python as the programming language
- Streamlit for the web interface
- SpeechRecognition library for audio processing
- OpenAI API (though currently only used for its API key setup)
- PyAudio for microphone access

3. **Current Implementation:**

- The bot responds to 5 predefined questions about personal history, strengths, growth areas, misconceptions, and boundary-pushing
- Uses simple keyword matching rather than full AI conversation
- Has a clean, user-friendly interface with microphone input

Setup Instructions

Prerequisites

1. Python 3.7 or higher installed

2. A microphone connected to your computer

3. Stable internet connection (for speech-to-text conversion)

Installation Steps

1. Clone the repository (if applicable) or create a project folder with your files

2. Set up a virtual environment(recommended):

“

python -m venv venv

On Windows use: venv\Scripts\activate

“

3. Install dependencies:

“

pip install -r requirements.txt

“

Note: If you encounter issues with PyAudio installation on Windows, try:

“

pip install pipwin

pipwin install pyaudio

“

4. Set up your OpenAI API key:

- Create a `.env` file in your project root

- Add your OpenAI API key:

“

API_KEY=your_api_key_here

“

5. Run the application:

“

streamlit run app.py

“

6. Access the application:

- The app will automatically open in your default browser at `http://localhost:8501`
- If not, manually navigate to the URL shown in your terminal

Usage Instructions:

1. Click the "**Speak to Voice-Chatbot**" button
2. Allow microphone access when prompted by your browser
3. Speak clearly one of the predefined questions:
   - "life story"
   - "superpower"
   - "grow"
   - "misconception"
   - "boundaries"
4. The app will convert your speech to text and display the matching response

Approach and Design Decisions:

Technical Approach

1. Modular Architecture:
   - Separated concerns with `utils.py` for business logic and `app.py` for UI/UX
   - Environment variables for sensitive configuration (API keys)

2. Speech Processing:
   - Used SpeechRecognition library with Google's speech-to-text API for reliable conversion
   - Implemented error handling for audio processing failures

3. Response System:
   - Currently using simple keyword matching with regex for immediate responses
   - Structured predefined responses in a dictionary for easy maintenance

UI/UX Decisions

1. Streamlit Framework:
   - Chosen for rapid development of data apps with minimal frontend code
   - Provides built-in components for audio processing and clean UI

2. Visual Design:
   - Clean, minimalist interface focusing on core functionality
   - Added a sidebar with chatbot information for context
   - Green button styling for clear call-to-action
   - Robot icon for personality

3. User Guidance:

- Clearly listed predefined questions to set proper expectations
- Visual feedback during listening ("Listening..." message)
- Display of both user input and bot response for transparency

Future Improvement Opportunities

1. Enhanced Conversation:

- Integrate actual OpenAI API for more natural conversations beyond predefined Q&A
- Implement conversation memory for context-aware responses

2. Error Handling:

- More robust error recovery for microphone issues
- Fallback input method (text input) when voice fails

3. Performance:

- Local speech-to-text options to reduce internet dependency
- Caching for frequent queries

4. UI Enhancements:

- Animation during listening/speaking
- Response audio playback (text-to-speech)
- Mobile responsiveness improvements

5. Configuration:

- Make predefined questions configurable via external file/database
- Add language selection options

The current implementation provides a solid foundation for a personal narrative chatbot that can be expanded in numerous directions while maintaining its simple, voice-first interaction model.