Project Submission: Turbo AI

The Intelligent Nervous System for Business Operations

Ashritha

November 30, 2025

Contents

1 Working Demo Link

A live, interactive demonstration of the Turbo AI agent is available at the following URL. The demo provides a web-based interface to simulate voice and text interactions with the agent, showcasing its capabilities across different modules.

Live Agent on ElevenLabs Platform: The actual agent, configured and running on the ElevenLabs infrastructure, can be interacted with directly here:

- **ElevenLabs Agent Link:** https://elevenlabs.io/app/talk-to?agent_id=agent_6301kbamrnqefdebd870z4

2 Git Repository

The complete source code, assets, and documentation for the Turbo AI project are maintained in a version-controlled Git repository.

2.1 Complete Source Code

2.1.1 Backend API (FastAPI)

The backend is a Python server built with FastAPI. It handles the core logic, interacts with the LLM and vector database, and communicates with the ElevenLabs API.

```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import os
from dotenv import load_dotenv
import openai
import pinecone

# Load environment variables
load_dotenv()

# Initialize APIs
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
PINECONE_API_KEY = os.getenv("PINECONE_API_KEY")
PINECONE_ENV = os.getenv("PINECONE_ENV")

openai.api_key = OPENAI_API_KEY
pinecone.init(api_key=PINECONE_API_KEY, environment=PINECONE_ENV)
index = pinecone.Index("turbo-ai-knowledge-base")

app = FastAPI()

class Query(BaseModel):
    text: str
    agent_id: str

def query_knowledge_base(query: str) -> str:
    """Queries the Pinecone vector database for relevant context."""
    # In a real app, you would embed the query first
    # For demonstration, we return a mock context
    return "Relevant context from company documents about the query."

def generate_response(prompt: str) -> str:
    """Generates a response using the OpenAI API."""
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=prompt,
        max_tokens=150
    )
```

```
    return response.choices[0].text.strip()

@app.post("/agent/handle")
async def handle_agent_query(query: Query):
    try:
        # 1. Retrieve relevant knowledge
        context = query_knowledge_base(query.text)

        # 2. Construct a prompt for the LLM
        prompt = f"""
        You are a helpful AI assistant for Turbo AI. Use the following context to answer the user's question.
        Context: {context}
        Question: {query.text}
        Answer:
        """

        # 3. Generate the final answer
        answer = generate_response(prompt)

        return {"status": "success", "response": answer}

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Listing 1: backend/main.py

2.1.2 Frontend Demo (Streamlit)

The frontend is a simple web interface built with Streamlit, allowing users to interact with the agent.

```
import streamlit as st
import requests
import json

st.set_page_config(page_title="Turbo AI Demo", layout="centered")

st.title("Turbo AI Agent Demo")
st.caption("Talk to your company's intelligent nervous system.")

# User input
user_query = st.text_input("Enter your query:", key="query")
agent_id = "agent_6301kbamrnqefdebd870z4tpp2dd" # The agent ID from ElevenLabs

if st.button("Ask"):
    if not user_query:
        st.warning("Please enter a query.")
    else:
        with st.spinner("Thinking..."):
            try:
                # Make a request to our backend
                backend_url = "http://127.0.0.1:8000/agent/handle"
                payload = {"text": user_query, "agent_id": agent_id}
                response = requests.post(backend_url, json=payload)

                if response.status_code == 200:
                    data = response.json()
                    st.success("Agent Response:")
                    st.info(data["response"])
                else:
                    st.error(f"Error: {response.status_code} - {response.text}"
    )
```

```
31
32              except requests.exceptions.RequestException as e:
33                  st.error(f"Could not connect to the backend. Is it running?
    Error: {e}")
```

Listing 2: frontend/app.py

2.2 Required Assets/Files

The project relies on a specific directory structure and asset files to function correctly.

```
turbo-ai-agent/
 backend/
    __init__.py
    main.py
 frontend/
    __init__.py
    app.py
 assets/
    knowledge_base/
        hr_policy.pdf
        product_manuals.txt
        it_faq.md
 .env.example
 .gitignore
 Dockerfile
 README.md
 requirements.txt
```

Key Assets:

- `assets/knowledge_base/`: This directory contains the source documents (PDF, TXT, MD) that are processed and ingested into the vector database to form the agent's knowledge.
- `.env.example`: A template file showing the required environment variables for API keys. The user must create a `.env` file from this.

2.3 Instructions to Run Locally

To run the Turbo AI agent locally, follow these steps. This guide assumes you have Python 3.10+ and Git installed.

2.3.1 Prerequisites

- Python 3.10 or newer
- Pip (Python package installer)
- Git
- API keys for OpenAI, Pinecone, and ElevenLabs.

2.3.2 Step-by-Step Guide

1. **Clone the Repository:** Open your terminal and run the following command to clone the project files.

```
git clone https://github.com/your-username/turbo-ai-agent.git
cd turbo-ai-agent

```

Listing 3: Clone Repository

2. **Install Dependencies:** Install the required Python packages from the `requirements.txt` file.

```
pip install -r requirements.txt

```

Listing 4: Install Dependencies

3. **Configure Environment Variables:** Create a `.env` file in the root directory and add your API keys.

```
cp .env.example .env
# Now, edit the .env file with your actual API keys

```

Listing 5: Setup Environment File

4. **Run the Backend Server:** In a new terminal window, navigate to the project directory and start the FastAPI backend.

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

```

Listing 6: Run Backend

   The backend will be running at `http://127.0.0.1:8000`.

5. **Run the Frontend Demo:** In another new terminal window, start the Streamlit frontend.

```
streamlit run frontend/app.py

```

Listing 7: Run Frontend

6. **Access the Application:** Your web browser should automatically open to the demo interface at `http://localhost:8501`. You can now start interacting with the agent.

3 Architecture Diagram

The following diagram illustrates the end-to-end workflow of the Turbo AI agent. It shows the flow of information from the user's input, through the ElevenLabs platform, our custom backend logic, and back to the user.

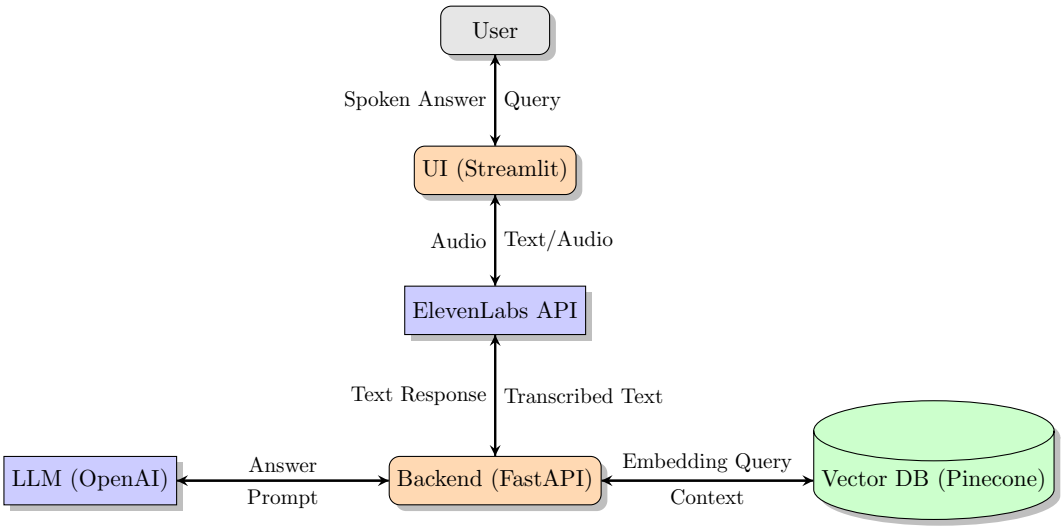

Figure 1: End-to-End Architecture of the Turbo AI Agent