# Projects steps :

- **Working of the Project** – Explain how the project functions.
- **Project Architecture** – Describe the structure and components of the project.
- **Build from Scratch** – Develop the project step by step.
- **Deploy the Project** – Make the project live and operational.

✅ Answer user queries based on preloaded company documents.

✅ Accept **text-based** or **voice-based** inputs and provide AI-generated responses.

✅ Convert **speech-to-text** (using **Whisper**) and **text-to-speech** (using **ElevenLabs**).

✅ Translate queries and responses to different languages.

✅ Handle document uploads and extract relevant information.

---

## Tools and Technologies Used

### 1️⃣ Streamlit

Used for building the chatbot UI (User Interface).

### 2️⃣ LangChain

Used to integrate OpenAI's GPT model and **ConversationalRetrievalChain** for intelligent Q&A.

### 3️⃣ OpenAI GPT (via LangChain)

Handles the natural language processing for chat responses.

### 4️⃣ FAISS (Facebook AI Similarity Search)

Manages the **vectorstore** for document-based retrieval, allowing AI to answer questions based on uploaded files.

### 5️⃣ Whisper (Speech-to-Text)

Transcribes voice input into text, enabling voice queries.

### 6️⃣ ElevenLabs (Text-to-Speech)

Converts AI-generated responses into natural-sounding audio.

### 7️⃣ Deep Translator (Google Translator API)

Enables multi-language support by translating user input and AI responses.

### 8️⃣ PyMuPDF (fitz)

Extracts text from uploaded PDF files.

### 9️⃣ Sounddevice & SciPy

Records audio input for speech-to-text conversion.

### 🔟 Dotenv

Manages API keys securely using environment variables.

---

## Flow of the Project

**1️⃣ User Input**

- The user can type a query or record their voice.
- If voice input is used, **Whisper** transcribes it into text.

**2️⃣ Processing the Query**

- If the query is not in English, it is translated using **Google Translator**.
- The system then searches for relevant information in the **FAISS vectorstore**.
- The **GPT model (via LangChain)** processes the query and generates a response.

### 3️⃣ Generating the Response

- The AI response is translated back to the user's language (if necessary).
- The chatbot displays the response as text.
- The response is converted into speech using **ElevenLabs** for an audio output option.

### 4️⃣ User Interaction & Document Uploads

- Users can upload documents (PDF, TXT, MD, HTML) to enhance AI knowledge.
- Extracted text is stored in the **FAISS vectorstore** for retrieval.

### 5️⃣ Conversation Memory

- **Session state** tracks chat history for context-aware responses.
- Users can reset the conversation if needed.

# Architecture Overview

The system consists of multiple components working together in the following flow:

## 1. User Input Handling

- Users can input queries in two ways:
  - **Text input** via the Streamlit UI
  - **Voice input** using Whisper ASR (Automatic Speech Recognition)

## 2. Text Processing & Translation

- If the query is not in English, it is translated using **Google Translator**.
- The processed query is sent to the retrieval-based AI model.

## 3. Knowledge Retrieval & AI Response Generation

- A **FAISS vector store** is used for storing and retrieving company-related documents.
- **LangChain ConversationalRetrievalChain** fetches relevant documents and interacts with OpenAI's **GPT-4 Turbo** model to generate an appropriate response.
- The response is translated back to the user's preferred language if needed.

### 4. Text-to-Speech (TTS) Output

- If the user has enabled **audio output**, the response is converted to speech using **ElevenLabs API**.

### 5. UI & Interaction Management

- The Streamlit UI displays chat history, document upload options, and language selection.
- Users can upload documents (PDF, TXT, MD, HTML), which are processed using **PyMuPDF (fitz)** and stored in the FAISS vector store for future queries.
- The chatbot maintains conversation history for context-aware responses.

---

## Technology Stack

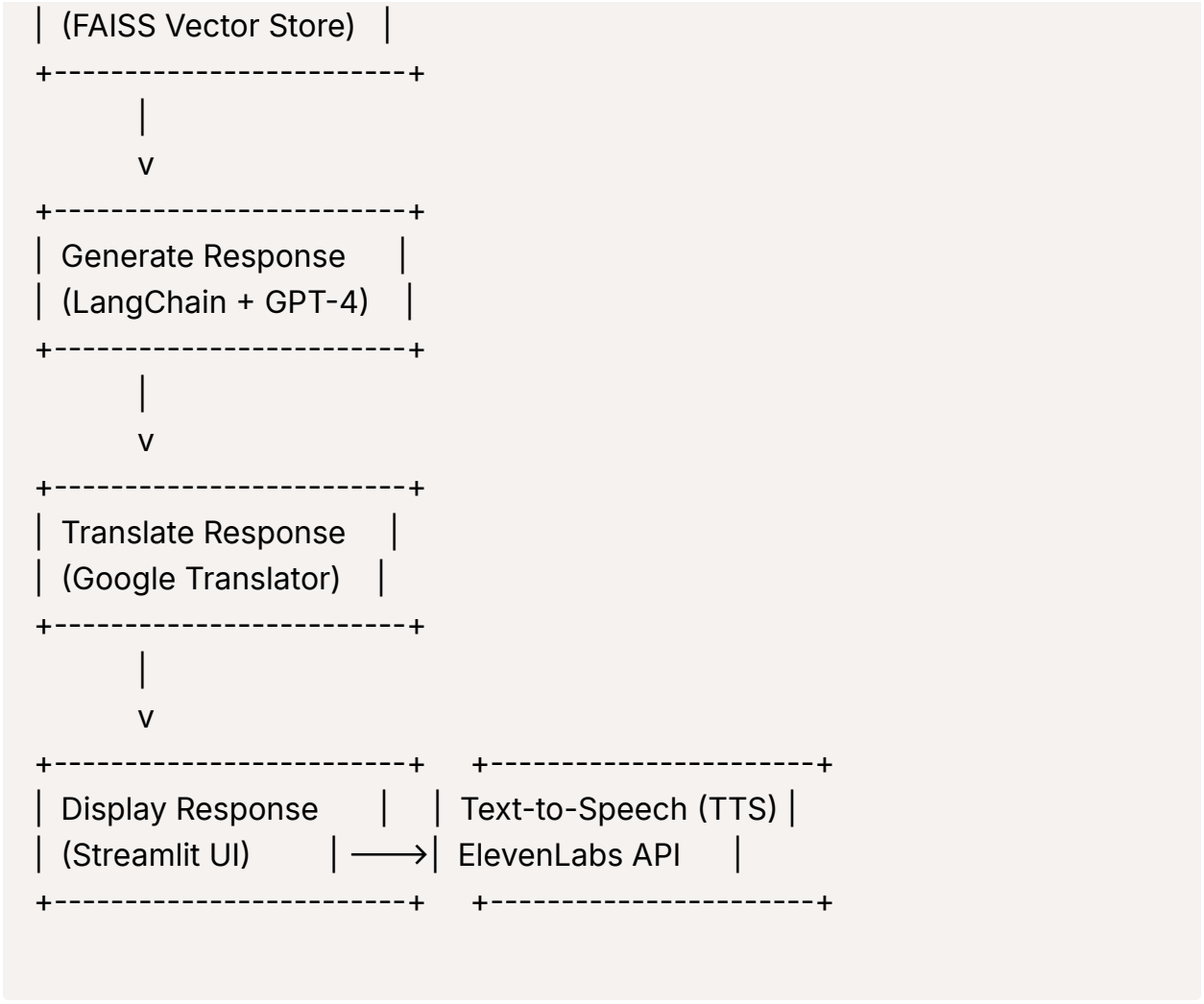
| Component | Tool/Library |
|---|---|
| Frontend UI | **Streamlit** |
| AI Model | **OpenAI GPT-4 Turbo** (via LangChain) |
| Speech-to-Text | **Whisper ASR** |
| Text-to-Speech | **ElevenLabs API** |
| Translation | **Google Translator API** |
| Vector Store | **FAISS (Facebook AI Similarity Search)** |
| Document Processing | **PyMuPDF (fitz) for PDF, text parsing for TXT/MD/HTML** |
| Environment Management | **dotenv (for API keys)** |

| | |
|---|---|
| Audio Recording | **sounddevice, scipy.io.wavfile** |

## Flow Diagram

Here’s a high-level architectural flow diagram:

```
plaintext
CopyEdit
+------------------------+
|     User Input         |
| (Text / Voice Query)   |
+-----------+------------+
            |
            v
+------------------------+
|  Speech-to-Text (ASR)  |
|  Whisper AI Model      |
+------------------------+
            |
            v
+------------------------+
|  Language Detection    |
|  & Translation         |
|  (Google Translator)   |
+------------------------+
            |
            v
+------------------------+
| Query Understanding    |
| (LangChain + GPT-4)    |
+------------------------+
            |
            v
+------------------------+
| Knowledge Retrieval    |
```




## How It Works (Step-by-Step Flow)

1. **User provides input**
   - If text, it is directly processed.
   - If voice, it is recorded and transcribed by **Whisper AI**.
2. **Language detection & translation**
   - If the query is not in English, it is translated using **Google Translator**.
3. **Query processing & AI response**
   - The chatbot fetches **relevant context** from FAISS Vector Store.
   - Uses **GPT-4 Turbo (LangChain)** to generate an answer.

4. **Response translation (if needed)**
    - If the user's preferred language is not English, the response is translated.
5. **Output response to the user**
    - **Text response** appears in the chat window.
    - **Audio response (optional)** is generated via **ElevenLabs API**.
6. **Maintain conversation history**
    - The system keeps track of user queries for context-aware interactions.
7. **Document upload & knowledge updates**
    - Users can upload **PDF/TXT/MD/HTML** files.
    - Extracted text is stored in **FAISS vector store** for future queries.