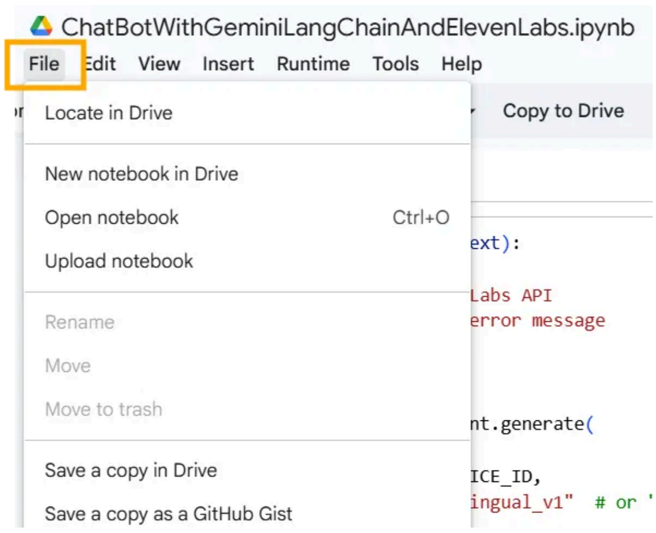
ChatBot with Gemini AI, LangChain and ElevenLabs

- **Prerequisites**
- **Open the below provided Colab link**

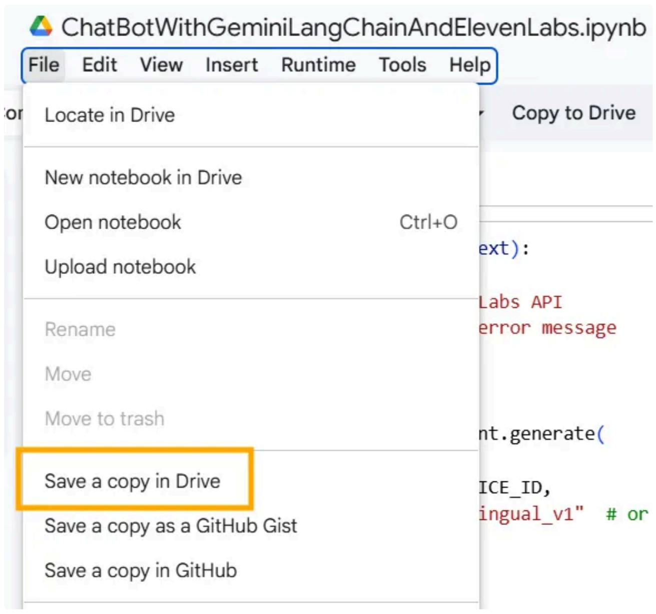
  https://colab.research.google.com/drive/1qo3iWIk2Rv6HnKmqphjIkV9rirDpTrP4?authuser=1

- **Copying Code to your Google Drive**
  - On the top left corner of Google Colab Notebook you can find **File,** click on it.



```
ext):

Labs API
error message


nt.generate(

ICE_ID,
ingual_v1"  # or '
```

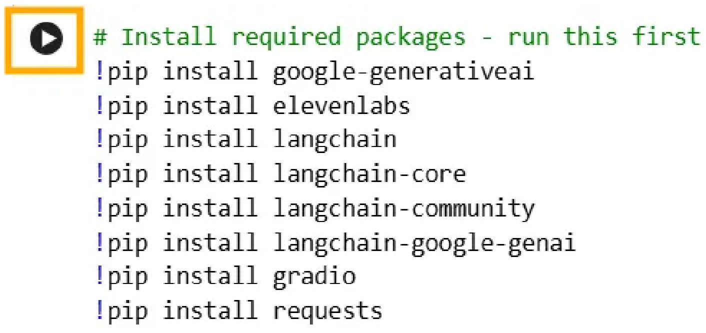
  - Click on **Save a Copy in Drive**



```
ext):

Labs API
error message


nt.generate(

ICE_ID,
ingual_v1"  # or
```

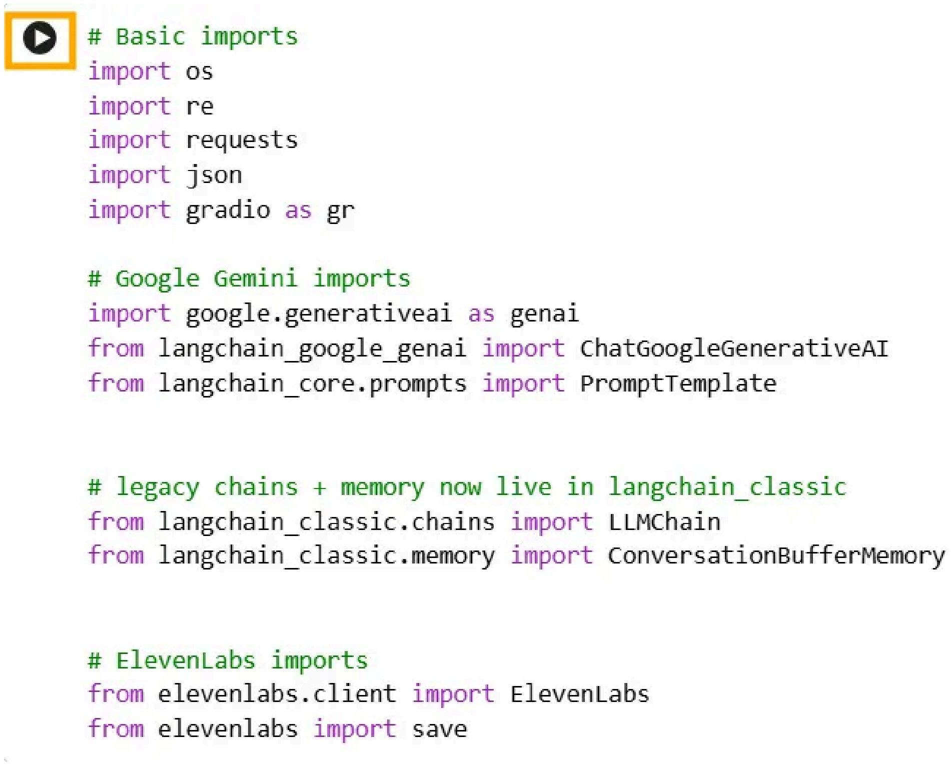
  - If you are not logged in to your Google Account, please log into it.
  - Once you are successfully logged in a new Google Colab Notebook with the given code will be opened

- **Click on the Run button to Install the Packages**



```
# Install required packages - run this first
!pip install google-generativeai
!pip install elevenlabs
!pip install langchain
!pip install langchain-core
!pip install langchain-community
!pip install langchain-google-genai
!pip install gradio
!pip install requests
```

- **Click on the Run button to import the required things to build the application**



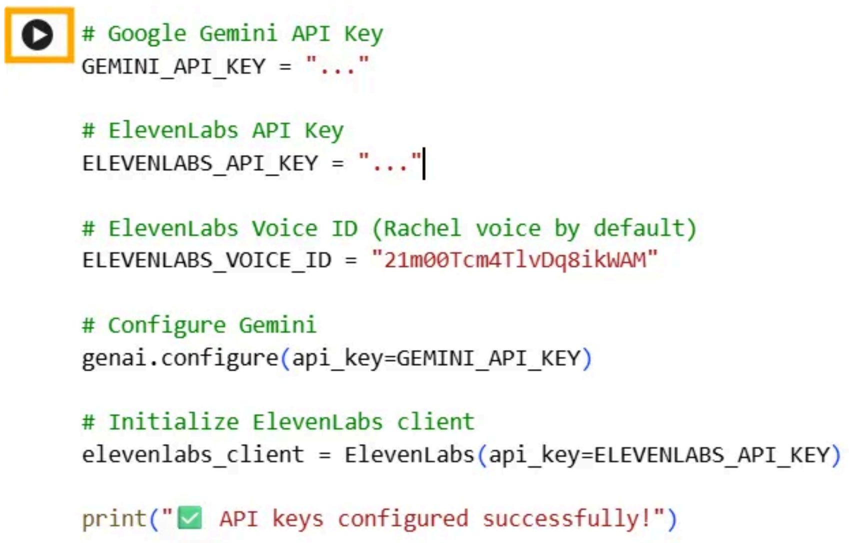
```
# Basic imports
import os
import re
import requests
import json
import gradio as gr

# Google Gemini imports
import google.generativeai as genai
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_core.prompts import PromptTemplate


# legacy chains + memory now live in langchain_classic
from langchain_classic.chains import LLMChain
from langchain_classic.memory import ConversationBufferMemory


# ElevenLabs imports
from elevenlabs.client import ElevenLabs
from elevenlabs import save
```

- **Assign API Keys**

  Generate API Key

  - Go to https://aistudio.google.com/app/api-keys
  - Create a new Secret Key
  - Copy the Secret Key for your use.

  - **ElevenLabs API Key**
    - Sign up at https://elevenlabs.io/
    - Go to "Developer" section
    - Copy your API key from the "API Key" section
  - Replace your Gemini AI API Key and ElevenLabs API Key with your own API Key



```
# Google Gemini API Key
GEMINI_API_KEY = "..."

# ElevenLabs API Key
ELEVENLABS_API_KEY = "..."

# ElevenLabs Voice ID (Rachel voice by default)
ELEVENLABS_VOICE_ID = "21m00Tcm4TlvDq8ikWAM"

# Configure Gemini
genai.configure(api_key=GEMINI_API_KEY)

# Initialize ElevenLabs client
elevenlabs_client = ElevenLabs(api_key=ELEVENLABS_API_KEY)

print("✅ API keys configured successfully!")
```

- **Set Gemini AI API Key to Environmental Variables**
  - Click on the Run button to set it as a Key

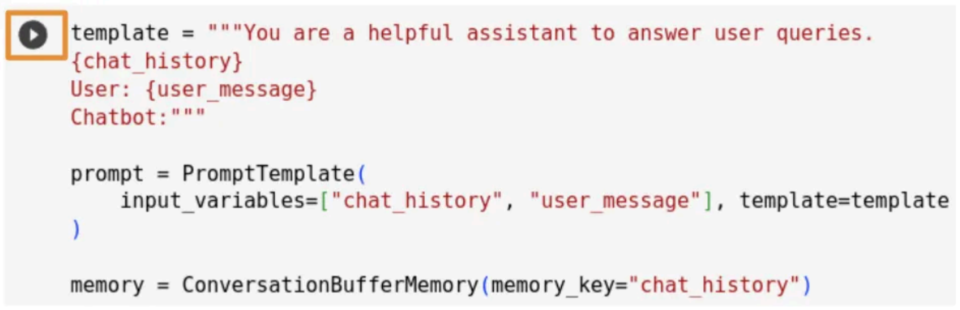
- **Assign your Voice to a variable and set the URL to a variable**
    - For Default/Pre-made Voices:
        1. Login to ElevenLabs at https://elevenlabs.io/
        2. Go to the Voices section from the left sidebar
        3. Click on the "Default Voices" tab
        4. Browse through the available voices (Rachel, Sarah, Clyde, etc.)
        5. Click on the three dots menu (⋮) next to any voice you want to use
        6. Select "Copy voice ID" from the dropdown menu
        7. The Voice ID will be copied to your clipboard
        8. Paste this ID as the value for `ELEVENLABS_VOICE_ID` in your code
    - For Custom/Cloned Voices:
        1. Login to ElevenLabs at https://elevenlabs.io/
        2. Go to the Voices section from the left sidebar
        3. Click on the "My Voices" tab
        4. Find your custom or cloned voice
        5. Click on the three dots menu (⋮) next to your voice
        6. Select "Copy voice ID"
        7. Paste this ID as the value for `ELEVENLABS_VOICE_ID` in your code
    - Copy the voice and update it in the below cell

```
# ElevenLabs Voice ID (Rachel voice by default)
ELEVENLABS_VOICE_ID = "21m00Tcm4TlvDq8ikWAM"
```

    - Click on the Run button to set the values

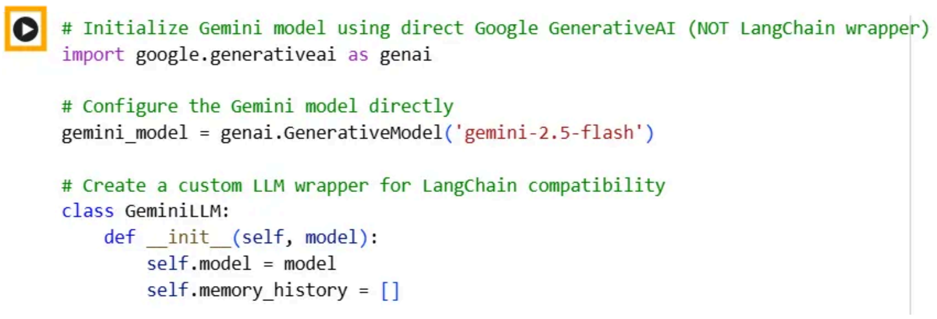
```
# ElevenLabs Voice ID (Rachel voice by default)
ELEVENLABS_VOICE_ID = "21m00Tcm4TlvDq8ikWAM"
```

- **Assigning the values for template, prompt, and memory**
  - You can update the first line of the template provided
  - Click on the Run button



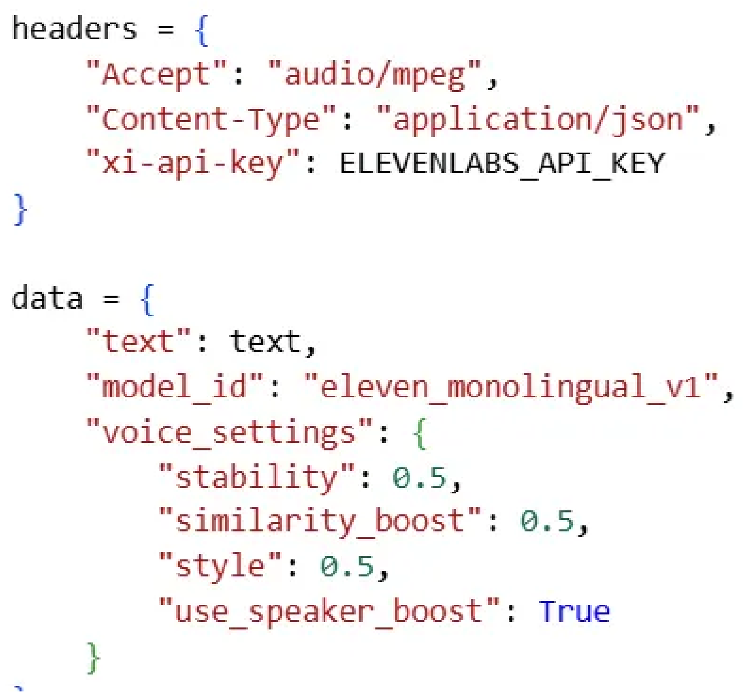
```
template = """You are a helpful assistant to answer user queries.
{chat_history}
User: {user_message}
Chatbot:"""

prompt = PromptTemplate(
    input_variables=["chat_history", "user_message"], template=template
)

memory = ConversationBufferMemory(memory_key="chat_history")
```

- **Initializing LLM Chain using Gemini AI**
  - Using Gemini AI we are creating an LLMChain
  - Click on the Run button



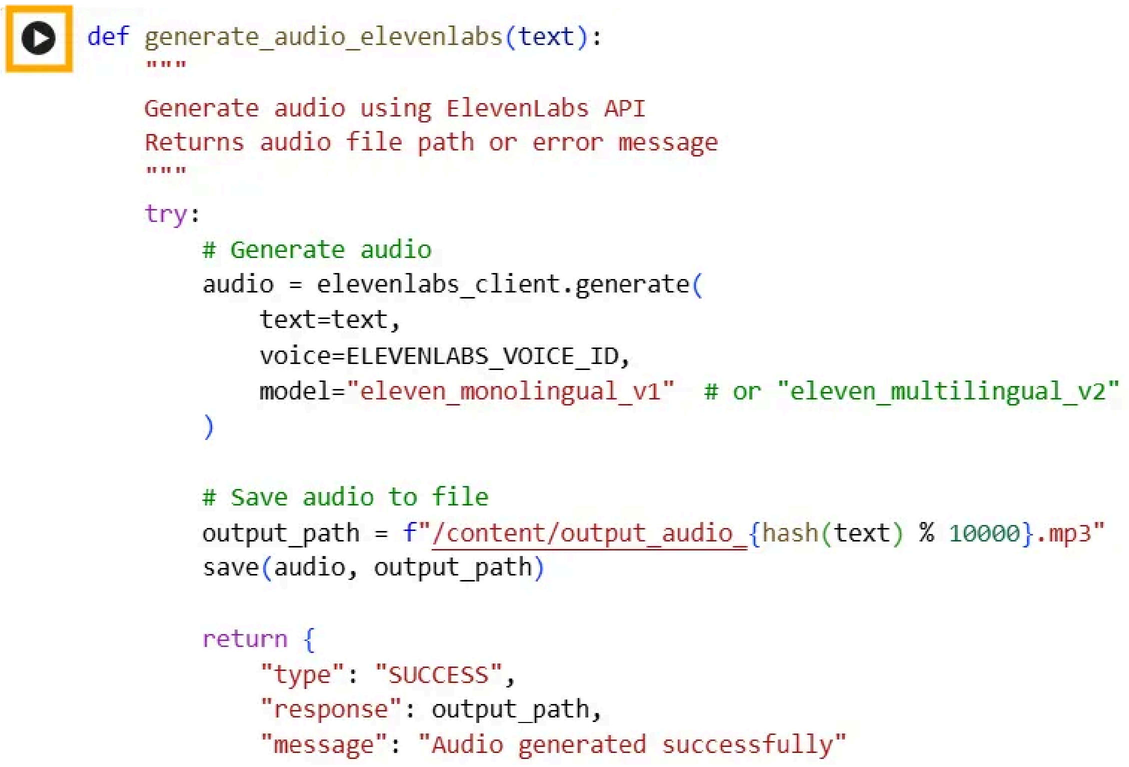
```
# Initialize Gemini model using direct Google GenerativeAI (NOT LangChain wrapper)
import google.generativeai as genai

# Configure the Gemini model directly
gemini_model = genai.GenerativeModel('gemini-2.5-flash')

# Create a custom LLM wrapper for LangChain compatibility
class GeminiLLM:
    def __init__(self, model):
        self.model = model
        self.memory_history = []
```

- **Setting headers**



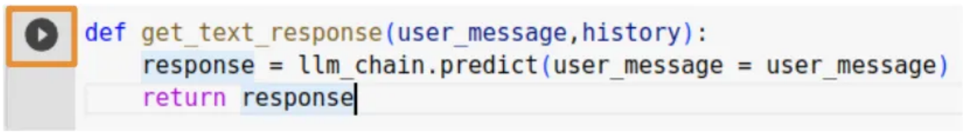
```
headers = {
    "Accept": "audio/mpeg",
    "Content-Type": "application/json",
    "xi-api-key": ELEVENLABS_API_KEY
}

data = {
    "text": text,
    "model_id": "eleven_monolingual_v1",
    "voice_settings": {
        "stability": 0.5,
        "similarity_boost": 0.5,
        "style": 0.5,
        "use_speaker_boost": True
    }
```

- **Define functions to fetch Audio Response from ElevenLabs**



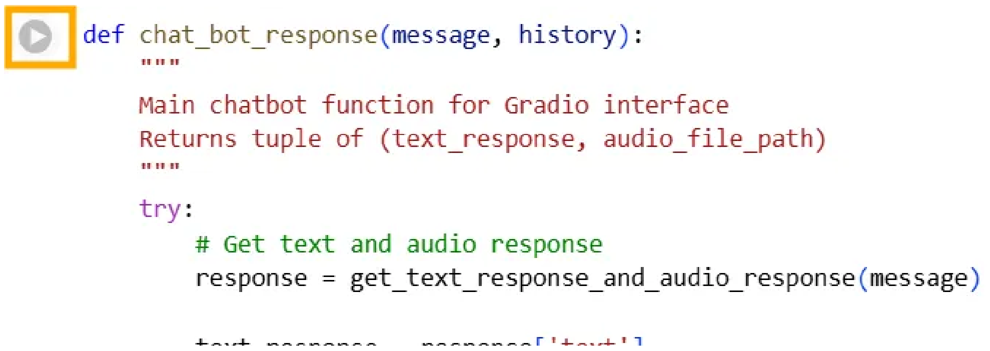
```python
def generate_audio_elevenlabs(text):
    """
    Generate audio using ElevenLabs API
    Returns audio file path or error message
    """
    try:
        # Generate audio
        audio = elevenlabs_client.generate(
            text=text,
            voice=ELEVENLABS_VOICE_ID,
            model="eleven_monolingual_v1"  # or "eleven_multilingual_v2"
        )

        # Save audio to file
        output_path = f"/content/output_audio_{hash(text) % 10000}.mp3"
        save(audio, output_path)

        return {
            "type": "SUCCESS",
            "response": output_path,
            "message": "Audio generated successfully"
```

- **Define a function to generate the response for the question you ask:**
    - Click on the Run button



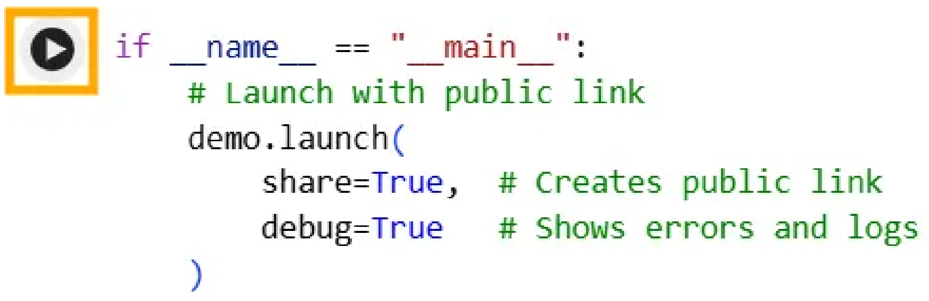
```python
def get_text_response(user_message,history):
    response = llm_chain.predict(user_message = user_message)
    return response
```

- **Define a function to generate the Audio response and Create a ChatInterface using the Gradio**
    - We are defining a **chat_bot_response** to return either an audio response format if audio file is fetched properly or returns a string response if it is an error
    - We are creating the ChatInterface from gradio and providing a function **chat_bot_response** and also examples
    - Check for other arguments here
    - Click on the Run button to create an interface



```
def chat_bot_response(message, history):
    """
    Main chatbot function for Gradio interface
    Returns tuple of (text_response, audio_file_path)
    """
    try:
        # Get text and audio response
        response = get_text_response_and_audio_response(message)
```

- **Launch your ChatBot with Gradio AP**
    - Click on the Run button to launch the App



```
if __name__ == "__main__":
    # Launch with public link
    demo.launch(
        share=True,  # Creates public link
        debug=True   # Shows errors and logs
    )
```

- **Now you can try asking questions in your ChatBot**

- **If you are getting any errors:**
    - Keep print statements to identify the issue
    - To identify the error you are getting please add **debug=True** while launching the gradio app.

```
if __name__ == "__main__": demo.launch(debug=True)
```

    - your can add try and except block to handle the errors

```
def get_text_response(user_message,history): try: response =
llm_chain.predict(user_message = user_message) except
Exception as e: print("Error:", e) try: print("Error:",
e.error.message) response = "Failed to reply: " +
e.error.message except Exception as e: response = "Failed to
reply" return response
```