```
!pip install langchain

!pip install openai

!pip install gradio

!pip install huggingface_hub import os import re import

requests import json import gradio as gr from

langchain.chat_models import ChatOpenAI from langchain

import LLMChain, PromptTemplate from

langchain.memory import ConversationBufferMemory

OPENAI_API_KEY="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"

PLAY_HT_API_KEY="85c5bd1039f646e788af7fb646b113b8"

PLAY_HT_USER_ID="t1FGti3Pn2RVkLOhaE1uCToAfsq1"

os.environ["OPENAI_API_KEY"] = OPENAI_API_KEY

play_ht_api_get_audio_url = "https://play.ht/api/v2/tts"

PLAY_HT_VOICE_ID="s3://voice-cloning-zero-shot/5cdffea7-96b5-4a65-
9f1ca8f665851135/mohammed-faizal/manifest.json"

template = """You are an enthusiastic high school student passionate about science and
exploration. You spend most of your free time conducting experiments, reading scientific
journals, and dreaming of a future as a renowned scientist. Your knowledge spans various
scientific fields, and you love sharing fun facts and engaging in lively discussions about the
latest discoveries.

{chat_history}

User: {user_message} Chatbot:""" prompt = PromptTemplate(

input_variables=["chat_history", "user_message"], template=template

)
```

```
memory =

ConversationBufferMemory(memory_key="chat_history")

llm_chain = LLMChain(     llm=ChatOpenAI(temperature='0.5',

model_name="gpt-3.5-turbo"),     prompt=prompt,     verbose=True,

memory=memory,

)

headers = {

     "accept": "text/event-stream",

     "content-type": "application/json",

     "AUTHORIZATION": "Bearer "+ PLAY_HT_API_KEY,

     "X-USER-ID": PLAY_HT_USER_ID

} def

get_payload(text):

return {


   "text": text,

   "voice": PLAY_HT_VOICE_ID,

   "quality": "medium",

   "output_format": "mp3",

   "speed": 1,

   "sample_rate": 24000,

   "seed": None,
    "temperature": None

  }

def get_generated_audio(text):

payload = get_payload(text)

generated_response = {}

  try:

      response = requests.post(play_ht_api_get_audio_url, json=payload,

headers=headers)       response.raise_for_status()       generated_response["type"]=

'SUCCESS'       generated_response["response"] = response.text   except

requests.exceptions.RequestException as e:

      generated_response["type"]= 'ERROR'

      try:

        response_text = json.loads(response.text)

        if response_text['error_message']:

          generated_response["response"] = response_text['error_message']

        else:

          generated_response["response"] = response.text

except Exception as e:

        generated_response["response"] = response.text

except Exception as e:
    generated_response["type"]= 'ERROR'
generated_response["response"] = response.text
return generated_response def extract_urls(text):
    # Define the regex pattern for URLs     url_pattern =
r'https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+[/\w\.-]*'
    # Find all occurrences of URLs in the
text     urls = re.findall(url_pattern, text)
return urls def
get_audio_reply_for_question(text):
  generated_audio_event = get_generated_audio(text)
  #From get_generated_audio, you will get events in a string format, from that we need
to extract the url   final_response = {
      "audio_url": '',
      "message": ''
  }
  if generated_audio_event["type"] == 'SUCCESS':
    audio_urls = extract_urls(generated_audio_event["response"])
if len(audio_urls) == 0:
      final_response['message'] = "No audio file link found in generated event"
    else:       final_response['audio_url'] =
audio_urls[-1]   else:
   final_response['message'] =

generated_audio_event['response']   return final_response def

download_url(url):

   try:

       # Send a GET request to the URL to fetch the content

final_response = {

           'content':'',

           'error':''

       }

       response = requests.get(url)

       # Check if the request was successful (status code 200)

if response.status_code == 200:

           final_response['content'] = response.content

       else:

           final_response['error'] = f"Failed to download the URL. Status code:
{response.status_code}"     except Exception as e:

final_response['error'] = f"Failed to download the URL. Error: {e}"

return final_response def get_filename_from_url(url):

   # Use os.path.basename() to extract the file name from the

URL     file_name = os.path.basename(url)     return file_name

def get_text_response(user_message):
   response = llm_chain.predict(user_message =
user_message)     return response def
get_text_response_and_audio_response(user_message):
   response = get_text_response(user_message) # Getting the reply from Open AI
audio_reply_for_question_response = get_audio_reply_for_question(response)
final_response = {
      'output_file_path': '',
      'message':''
   }
   audio_url = audio_reply_for_question_response['audio_url']
if audio_url:
output_file_path=get_filename_from_url(audio_url)
download_url_response = download_url(audio_url)
audio_content = download_url_response['content']       if
audio_content:         with open(output_file_path, "wb") as
audio_file:
        audio_file.write(audio_content)
final_response['output_file_path'] = output_file_path
     else:
        final_response['message'] = download_url_response['error']
   else:
      final_response['message'] =

audio_reply_for_question_response['message']     return final_response def

chat_bot_response(message, history):

    text_and_audio_response =

get_text_response_and_audio_response(message)     output_file_path =

text_and_audio_response['output_file_path']     if output_file_path:

      return (text_and_audio_response['output_file_path'],)

    else:

      return text_and_audio_response['message']


demo = gr.ChatInterface(chat_bot_response,examples=["How are you doing?","What are
your interests?","Which places do you like to visit?"])

if __name__ == "__main__":

    demo.launch() #To create a public link, set `share=True` in `launch()`. To enable errors
and logs, set `debug=True` in `launch()`.

from huggingface_hub import

notebook_login notebook_login() from

huggingface_hub import HfApi api =

HfApi()

HUGGING_FACE_REPO_ID = "mohammedfaizal/VoiceAssistant"

%mkdir /content/ChatBotWithOpenAILangChainAndPlayHT

!wget -P  /content/ChatBotWithOpenAILangChainAndPlayHT/ https://s3.ap-
south1.amazonaws.com/cdn1.ccbp.in/GenAI-
Workshop/ChatBotWithOpenAILangChainPlayHT2/app.py
```

```
!wget -P /content/ChatBotWithOpenAILangChainAndPlayHT/ https://s3.ap-
south1.amazonaws.com/cdn1.ccbp.in/GenAI-
Workshop/ChatBotWithOpenAILangChainPlayHT/requirements.txt

%cd /content/ChatBotWithOpenAILangChainAndPlayHT

api.upload_file(

path_or_fileobj="./requirements.txt",

path_in_repo="requirements.txt",

   repo_id=HUGGING_FACE_REPO_ID,

   repo_type="space")

api.upload_file(

path_or_fileobj="./app.py",

path_in_repo="app.py",

repo_id=HUGGING_FACE_REPO_I

D,


   repo_type="space"))          print(prompt)          memory          =
ConversationBufferMemory(memory_key="chat_history") llm_chain
= LLMChain( llm=ChatOpenAI(

        temperature='0.5', model_name="gpt-3.5-turbo"),
         prompt=prompt,
        verbose=True,
        memory=memory,
)

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

```
demo = gr.ChatInterface(get_text_response, examples=["How are you doing?","What
are your interests?","Which places do you like to visit?"]) if name == " main ":

        demo.launch(share = True)
```