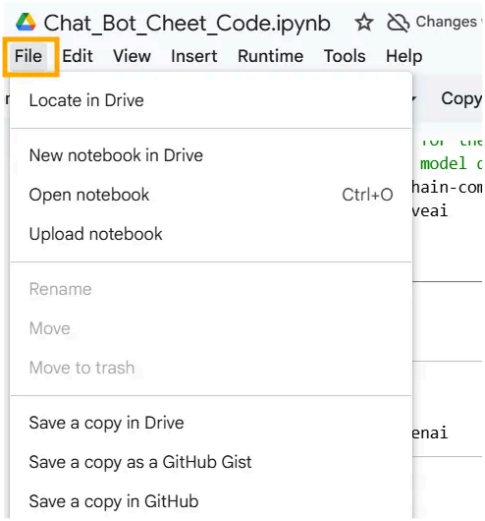
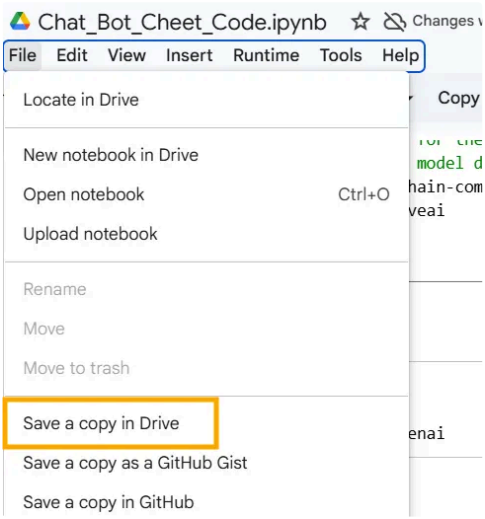
ChatBot with Gemini AI and LangChain Handbook

- ▶ **Prerequisites**
- **Open the below provided Colab link**
  - https://colab.research.google.com/drive/174hMxYiKrZ9dmLcw9Hd-DDpEkuC1qH7H?usp=sharing#scrollTo=SPyLmKojGGwP

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



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



  - 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 all required libraries for the chatbot
# LangChain: Framework for building LLM applications
# Google Generative AI: Python SDK for Gemini API
# Gradio: Creates web interface for the chatbot
# HuggingFace Hub: Optional for model deployment
!pip install -U langchain langchain-community langchain-google-genai
!pip install -U google-generativeai
!pip install -U gradio
!pip install -U huggingface_hub
```

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

```
import os
import gradio as gr
import google.generativeai as genai
```

- **Get the Gemini API Key and set it as environmental variable**
  - **Generate API Key**
    - Go to https://aistudio.google.com/app/api-keys
    - Create a new Secret Key
    - Copy the Secret Key for your use.
  - Replace your Gemini AI API Key with your own API Key

```
GEMINI_API_KEY="...."
genai.configure(api_key=GEMINI_API_KEY)
```

  - Click on the Run button

```
GEMINI_API_KEY="...."
genai.configure(api_key=GEMINI_API_KEY)
```

- **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
gemini_model = genai.GenerativeModel('gemini-1.5-flash')

# Custom LLM wrapper for Gemini
class GeminiLLM:
    def __init__(self, model):
        self.model = model
        self.memory_history = []

    def predict(self, user_message):
        # Build conversation context
        full_prompt = "You are a helpful assistant to answer user queries.\n'
        for msg in self.memory_history:
            full_prompt += f"{msg}\n"
        full_prompt += f"User: {user_message}\nChatbot:"

        # Generate response
        response = self.model.generate_content(full_prompt)
        answer = response.text
```

- **Define a function to generate the response for the question you ask:**
  - From the initialized llm_chain we will predict the response
    - Click on the Run button

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

- **Create a ChatInterface using the Gradio**
  - We are creating the ChatInterface from gradio and providing a function get_text_reponse and also examples
  - Check for other arguments here
  - Click on the Run button to create an interface

```
) = gr.ChatInterface(get_text_response, examples=["How are you doing?","What are your interests?","Which places do you like to
```

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

```
if __name__ == "__main__":
    demo.launch(debug=True) #To create a public link,
```

- **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)
```

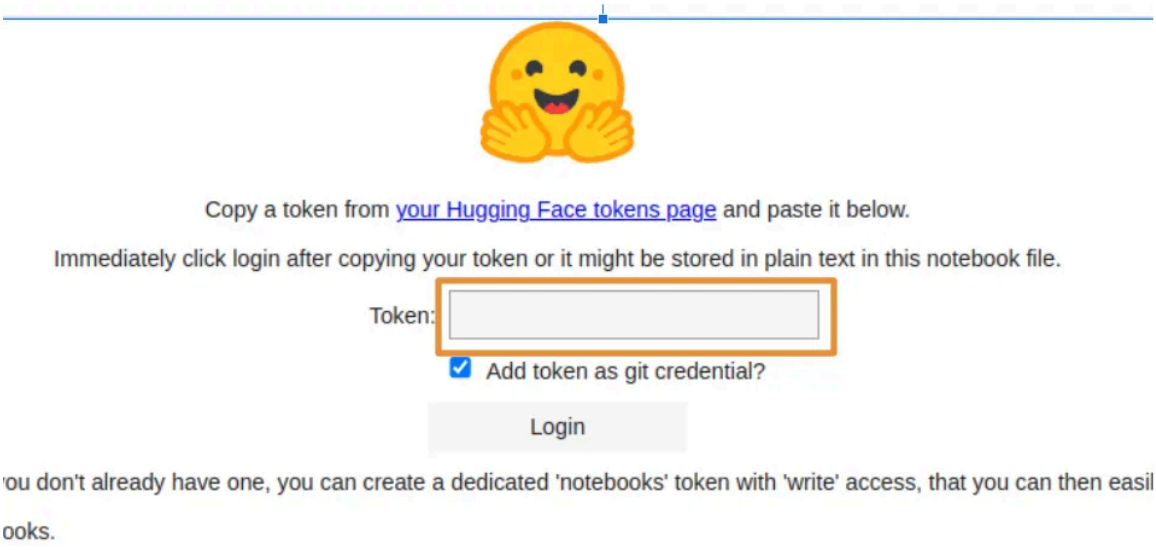
```
def get_text_response(user_message,history): try: response =
llm_chain.predict(user_message = user_message) except
Exception as e: print("Error:", e) try: print("Error:",
e.error.message) response = "Failed to reply: " +
e.error.message except Exception as e: response = "Failed to
reply" return response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

Publish your code to Hugging face

- **Login to Hugging Face from Google Colab**
  - Create a Hugging Face token and Copy
    - Login to Hugging Face https://huggingface.co/
    - Open https://huggingface.co/settings/tokens
    - Click on New token
    - Add a name for the Token
    - Choose a Role for the Token whether to Read or Write
    - Choose to Write and click on Create
  - Click on the Run button to enter Hugging Face Token

```
from huggingface_hub import notebook_login

notebook_login()
```



  - Now paste the hugging Face token in the textbox provided
- **Create HuggingFace API to push code from Google Colab**
  - Click on the Run button to create API

```
from huggingface_hub import HfApi
api = HfApi()
```

- **Adding Hugging Face Repo ID**
  - Copy Hugging Face Repo ID by opening the Hugging Face Repo Created

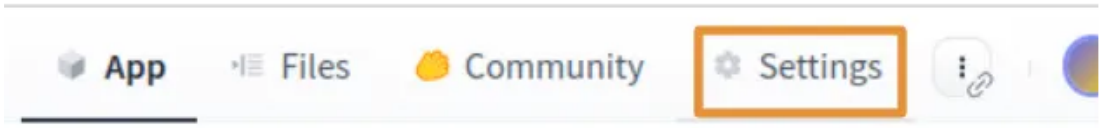
  - Replace your Repo ID

```
HUGGING_FACE_REPO_ID = "<Hugging Face User Name/Repo Na
```

  - Click on Run button to assign hugging Face Repo ID

```
HUGGING_FACE_REPO_ID = "<Hugging Face User Name/Repo Na
```

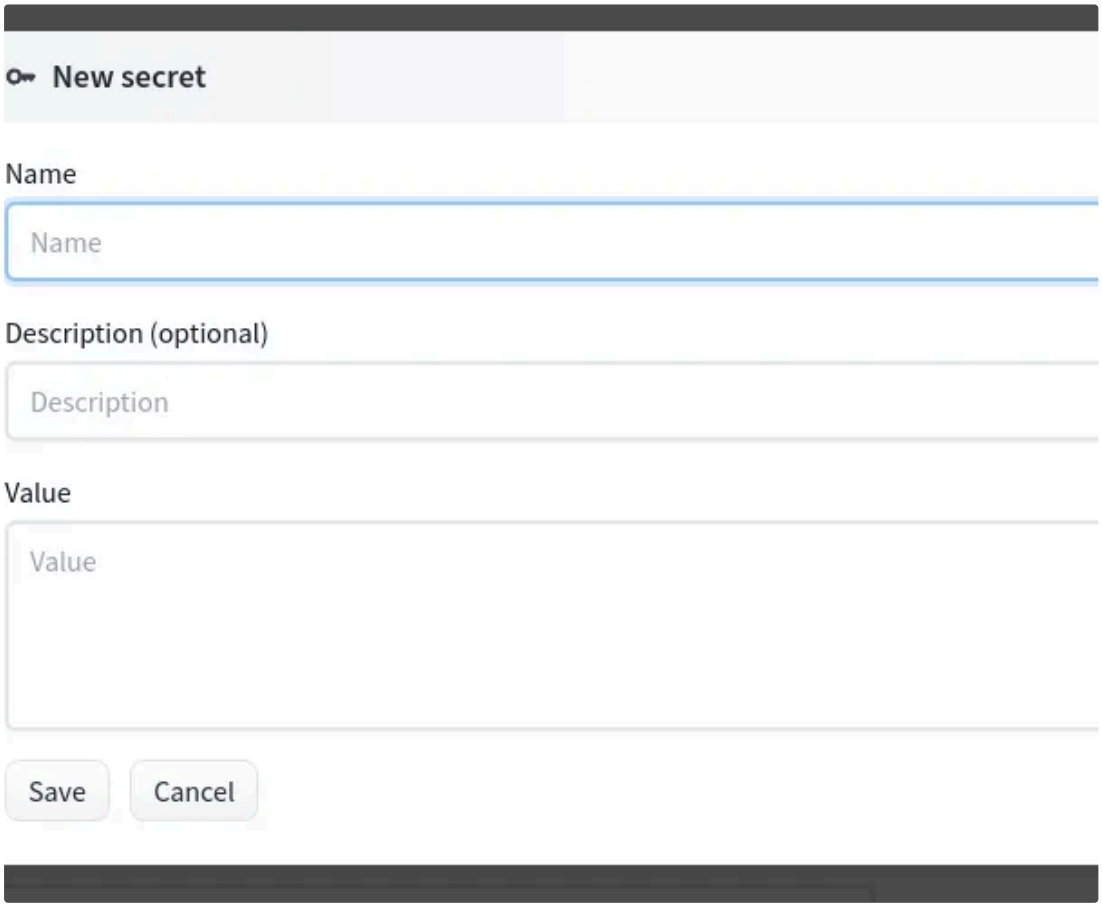
- **Add GEMINI_AI_API_KEY in Hugging Face secrets**
  - Click on the Settings Button



  - Go to the **Variables and secrets** section

  - Click on New Secret to create New Secrets

  - Enter Name and Value and Click on Save



- **Load files App and Requirements file**
  - Click on the Run button to download files

  - You should see the files here