AI CHATBOT

Google API Setup

Step 1:

Setting up google cloud console by creating a Google cloud project in this link https://console.cloud.google.com/welcome/new?pli=1. Created credentials in the console for oauth client id. After selecting the necessary options, download the credentials.json file and noted down the client secret id.

Step 2:

Enabled the services required, for our project we need, Google Docs API and Google Calendar API.

Chatgpt API setup

Visited this Openai link https://platform.openai.com/docs/overview and brought a $5 credit for making API requests. Copied the api key from my account to use it in code later.

Google docs creation with content

Created google docs in my gmail account with valid content of my availability.
Link: https://docs.google.com/document/d/1gmXENZiRJ1te56lqNRAARbp_7S8Uto5Kd1p-PUgzABU/edit?usp=sharing

Code Setup

Step 1:

I had setup my Visual Studio for jupyter notebooks. Created a new kernel for this project to install all the necessary libraries.

Step 2:

Installed langchain, openai, langchain_openai for the project setup.

Code Flow

Step 1:

Implemented the Google docs retreival in code with help of credentials.json file downloaded from Google API console. . Got some information on how to implement with this google documentation link : https://developers.google.com/docs

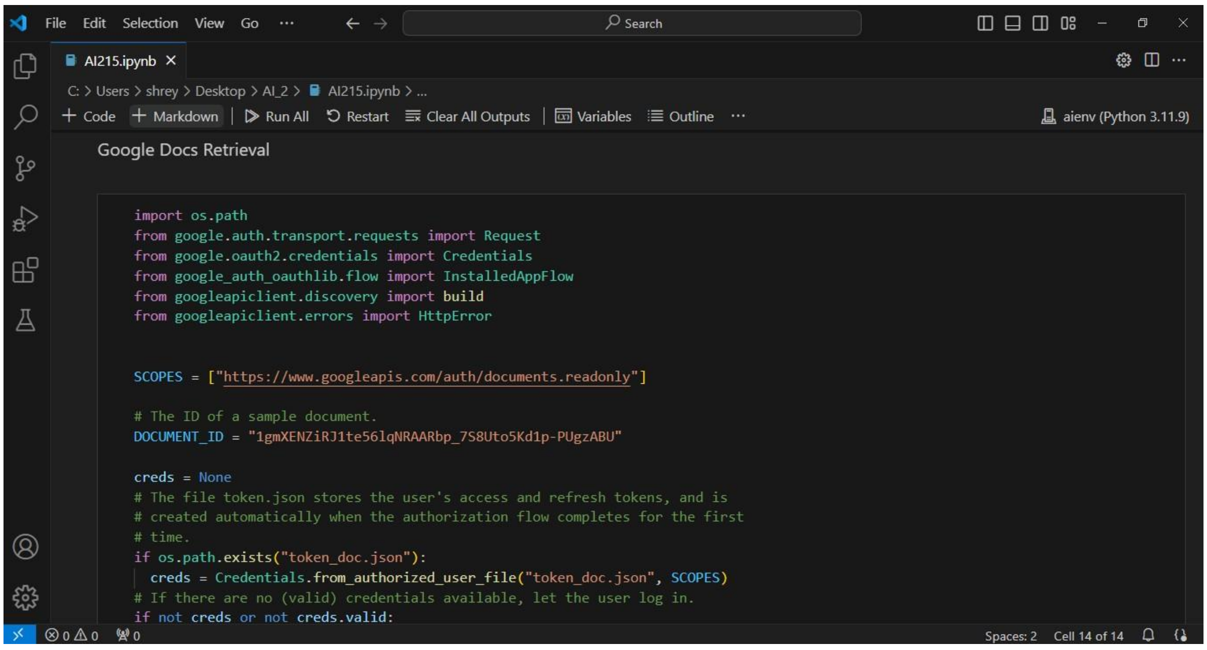
Step 2:

Implemented the Google Calendar event creation in code with help of credentials.json file downloaded from Google API console. Got some information on how to implement with this google documentation link : https://developers.google.com/calendar/api/quickstart/python

Step 3:

Used the Openai API key in code for accessing llm models. Gave the docs to chatgpt code. Made the code interactive with users for querying. When the user enters details to book with time details, the Calendar event gets created. This can be verified on the Google Calendar.

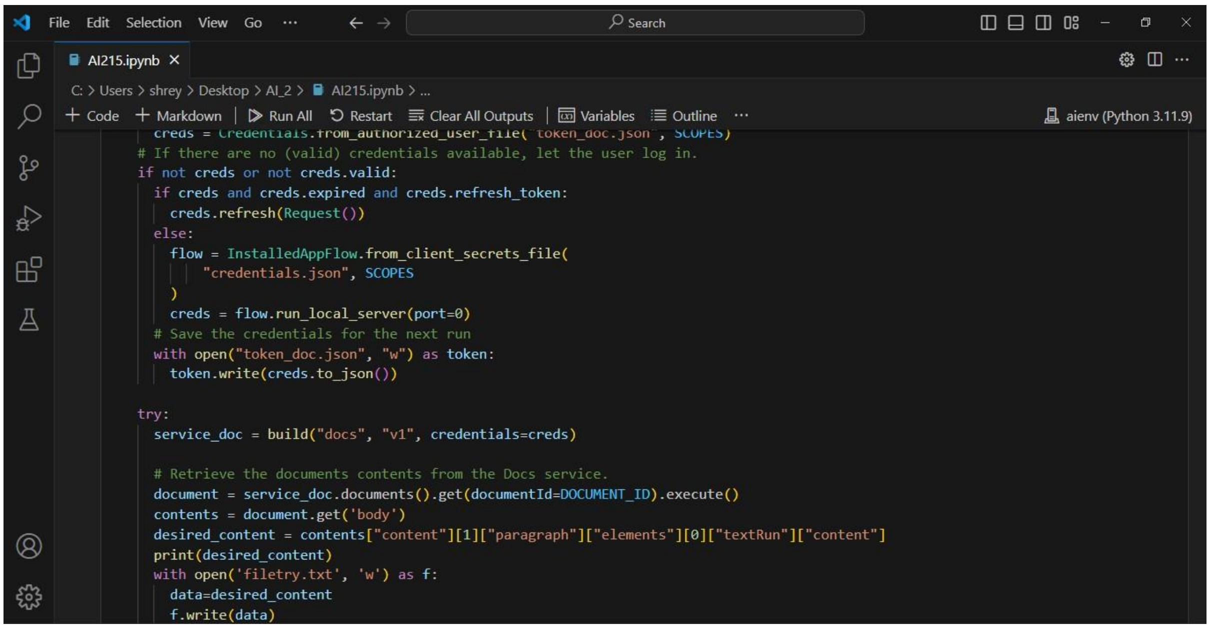
Code screenshots



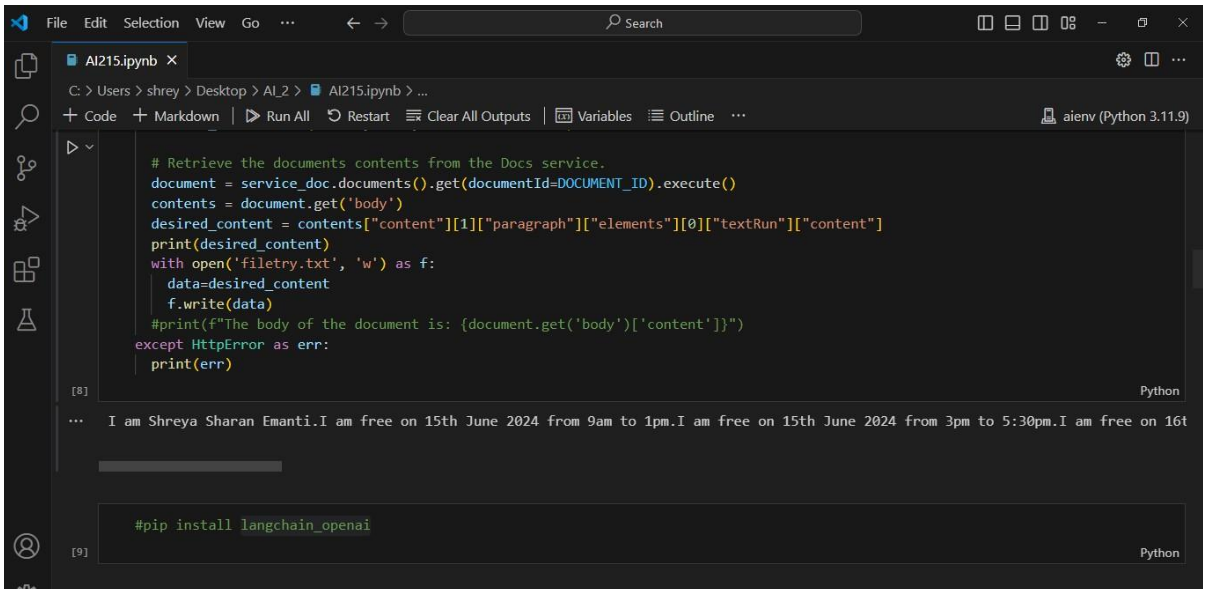
File Edit Selection View Go
Search
AI215.ipynb
C: > Users > shrey > Desktop > AI_2 > AI215.ipynb > ...
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
aienv (Python 3.11.9)
Spaces: 2 Cell 14 of 14

Google Docs Retrieval

```python
import os.path
from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError


SCOPES = ["https://www.googleapis.com/auth/documents.readonly"]

# The ID of a sample document.
DOCUMENT_ID = "1gmXENZiRJ1te56lqNRAARbp_7S8Uto5Kd1p-PUgzABU"

creds = None
# The file token.json stores the user's access and refresh tokens, and is
# created automatically when the authorization flow completes for the first
# time.
if os.path.exists("token_doc.json"):
  creds = Credentials.from_authorized_user_file("token_doc.json", SCOPES)
# If there are no (valid) credentials available, let the user log in.
if not creds or not creds.valid:
```



```
creds = Credentials.from_authorized_user_file("token_doc.json", SCOPES)
# If there are no (valid) credentials available, let the user log in.
if not creds or not creds.valid:
  if creds and creds.expired and creds.refresh_token:
    creds.refresh(Request())
  else:
    flow = InstalledAppFlow.from_client_secrets_file(
        "credentials.json", SCOPES
    )
    creds = flow.run_local_server(port=0)
  # Save the credentials for the next run
  with open("token_doc.json", "w") as token:
    token.write(creds.to_json())

try:
  service_doc = build("docs", "v1", credentials=creds)

  # Retrieve the documents contents from the Docs service.
  document = service_doc.documents().get(documentId=DOCUMENT_ID).execute()
  contents = document.get('body')
  desired_content = contents["content"][1]["paragraph"]["elements"][0]["textRun"]["content"]
  print(desired_content)
  with open('filetry.txt', 'w') as f:
    data=desired_content
    f.write(data)
```



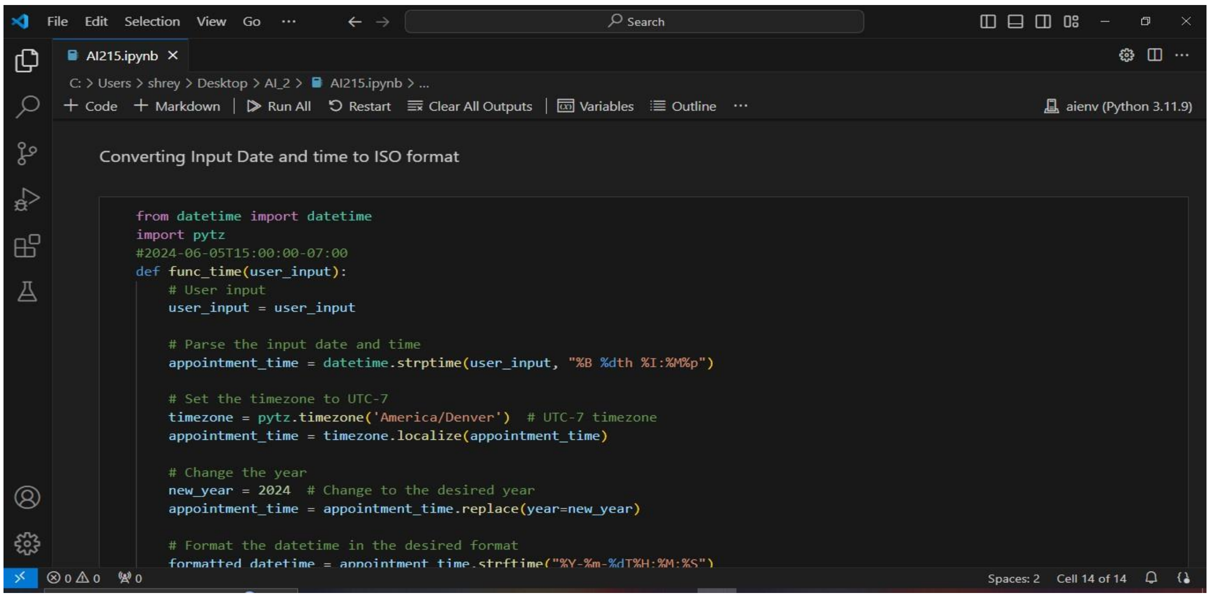
```
  # Retrieve the documents contents from the Docs service.
  document = service_doc.documents().get(documentId=DOCUMENT_ID).execute()
  contents = document.get('body')
  desired_content = contents["content"][1]["paragraph"]["elements"][0]["textRun"]["content"]
  print(desired_content)
  with open('filetry.txt', 'w') as f:
    data=desired_content
    f.write(data)
  #print(f"The body of the document is: {document.get('body')['content']}")
except HttpError as err:
  print(err)
```

[8]

```
I am Shreya Sharan Emanti.I am free on 15th June 2024 from 9am to 1pm.I am free on 15th June 2024 from 3pm to 5:30pm.I am free on 16t
```

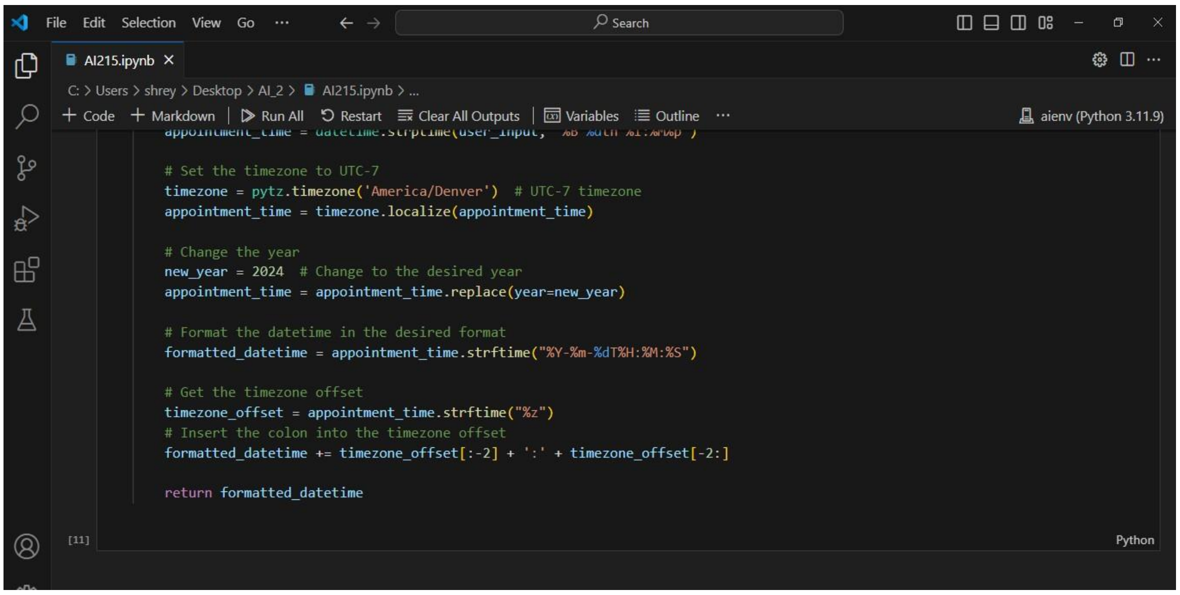
```
#pip install langchain_openai
```

[9]

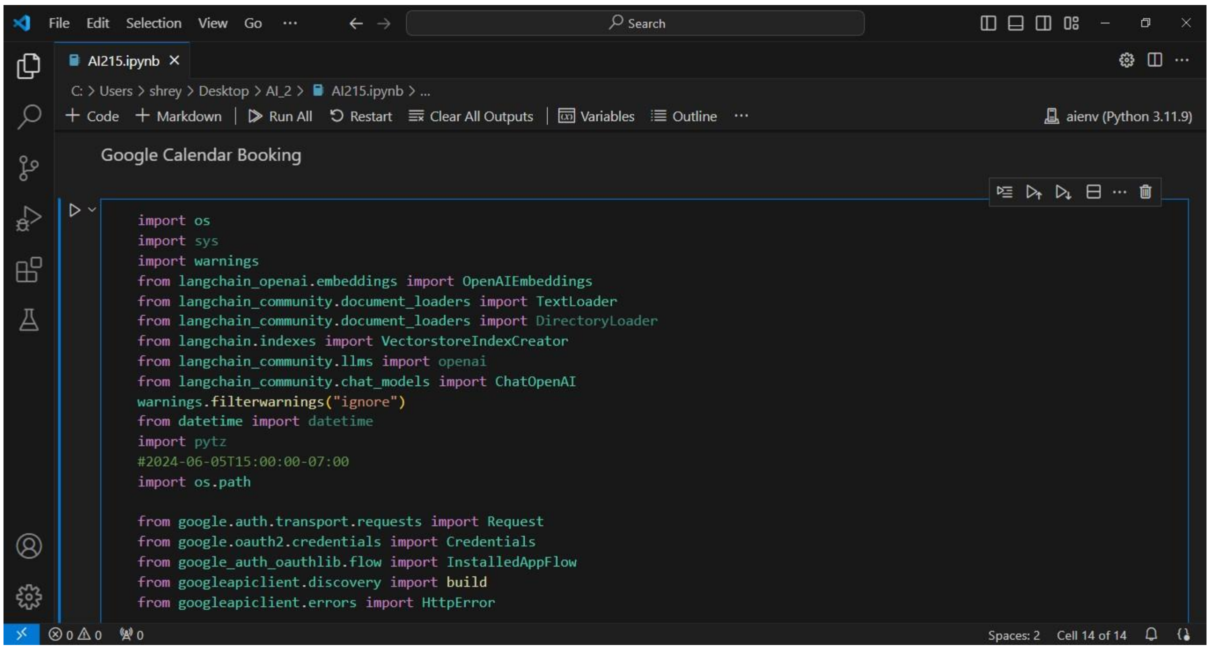


Converting Input Date and time to ISO format

```
from datetime import datetime
import pytz
#2024-06-05T15:00:00-07:00
def func_time(user_input):
    # User input
    user_input = user_input

    # Parse the input date and time
    appointment_time = datetime.strptime(user_input, "%B %dth %I:%M%p")

    # Set the timezone to UTC-7
    timezone = pytz.timezone('America/Denver')  # UTC-7 timezone
    appointment_time = timezone.localize(appointment_time)

    # Change the year
    new_year = 2024  # Change to the desired year
    appointment_time = appointment_time.replace(year=new_year)

    # Format the datetime in the desired format
    formatted_datetime = appointment_time.strftime("%Y-%m-%dT%H:%M:%S")
```

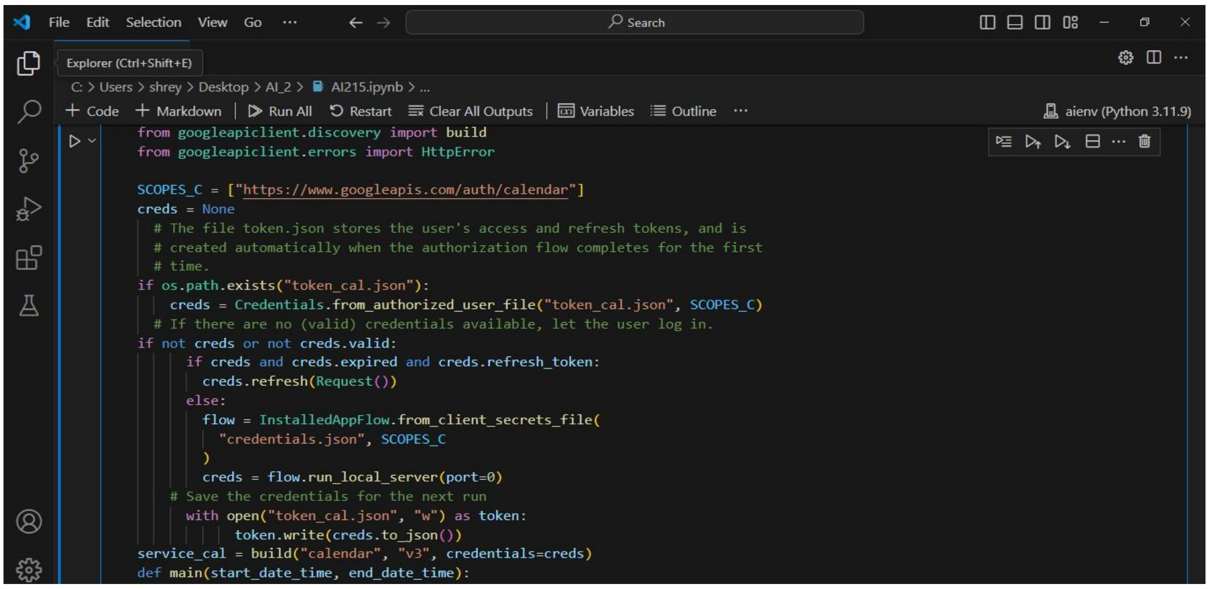

File Edit Selection View Go
Search
AI215.ipynb
C: > Users > shrey > Desktop > AI_2 > AI215.ipynb > ...
Code Markdown Run All Restart Clear All Outputs Variables Outline
aienv (Python 3.11.9)
[11]
Python

```python
    appointment_time = datetime.strptime(user_input, "%B %dth %I:%M%p")

    # Set the timezone to UTC-7
    timezone = pytz.timezone('America/Denver')  # UTC-7 timezone
    appointment_time = timezone.localize(appointment_time)

    # Change the year
    new_year = 2024  # Change to the desired year
    appointment_time = appointment_time.replace(year=new_year)

    # Format the datetime in the desired format
    formatted_datetime = appointment_time.strftime("%Y-%m-%dT%H:%M:%S")

    # Get the timezone offset
    timezone_offset = appointment_time.strftime("%z")
    # Insert the colon into the timezone offset
    formatted_datetime += timezone_offset[:-2] + ':' + timezone_offset[-2:]

    return formatted_datetime
```

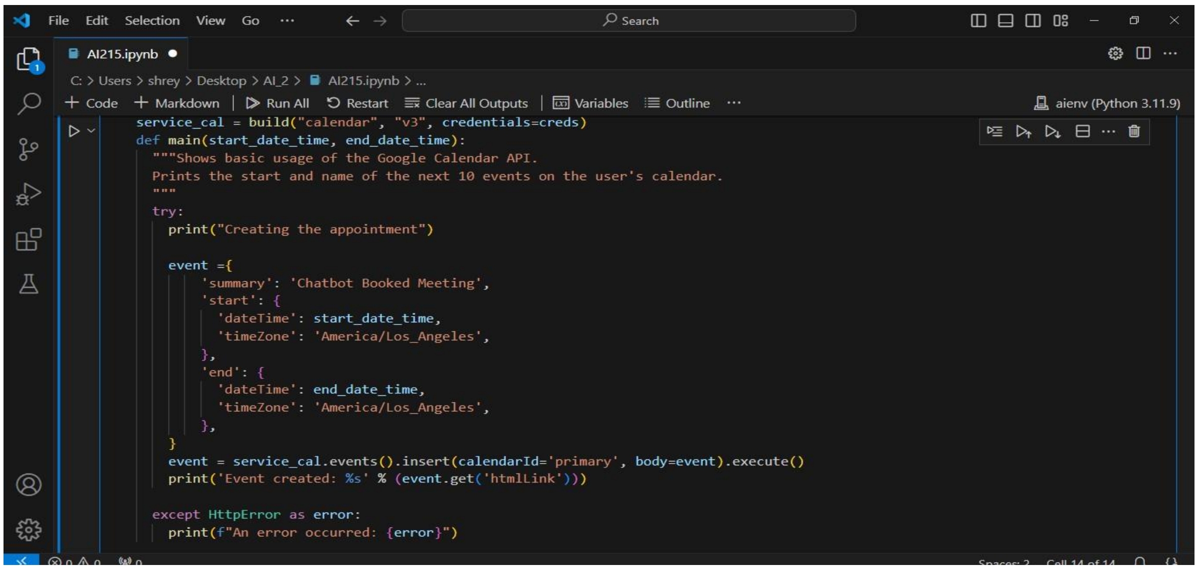

File Edit Selection View Go
Search
AI215.ipynb
C: > Users > shrey > Desktop > AI_2 > AI215.ipynb > ...
Code Markdown Run All Restart Clear All Outputs Variables Outline
aienv (Python 3.11.9)
Spaces: 2 Cell 14 of 14

Google Calendar Booking

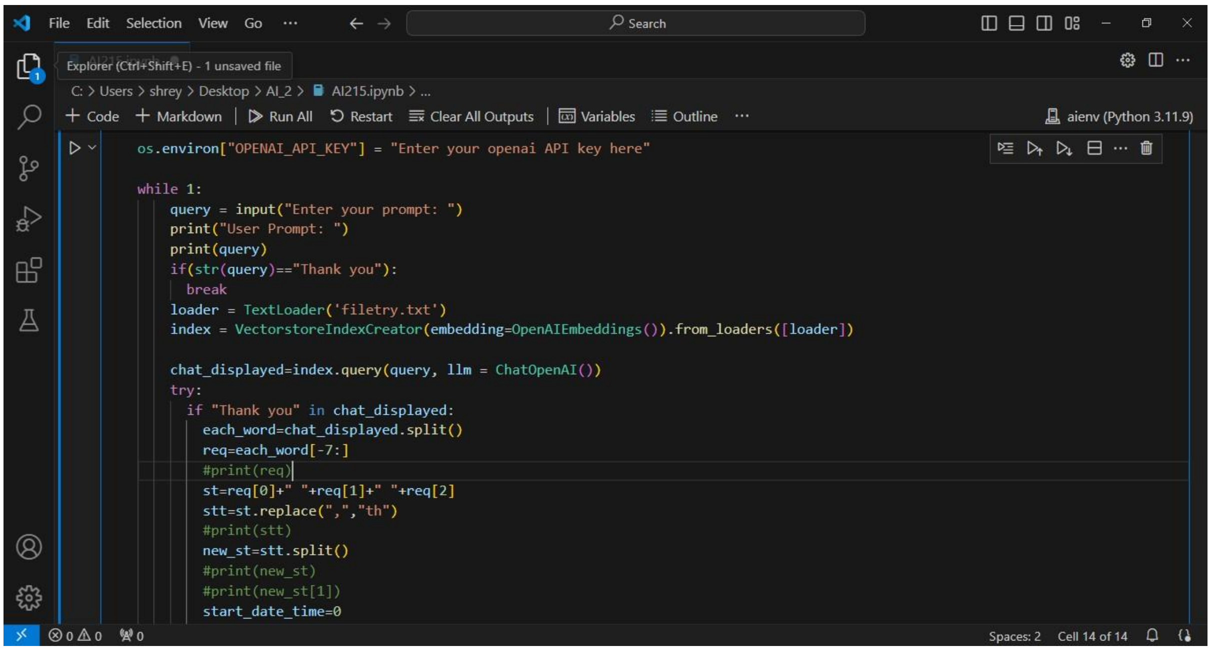
```python
import os
import sys
import warnings
from langchain_openai.embeddings import OpenAIEmbeddings
from langchain_community.document_loaders import TextLoader
from langchain_community.document_loaders import DirectoryLoader
from langchain.indexes import VectorstoreIndexCreator
from langchain_community.llms import openai
from langchain_community.chat_models import ChatOpenAI
warnings.filterwarnings("ignore")
from datetime import datetime
import pytz
#2024-06-05T15:00:00-07:00
import os.path

from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
```


File Edit Selection View Go
Search
Explorer (Ctrl+Shift+E)
C: > Users > shrey > Desktop > AI_2 > AI215.ipynb > ...
Code Markdown Run All Restart Clear All Outputs Variables Outline
aienv (Python 3.11.9)

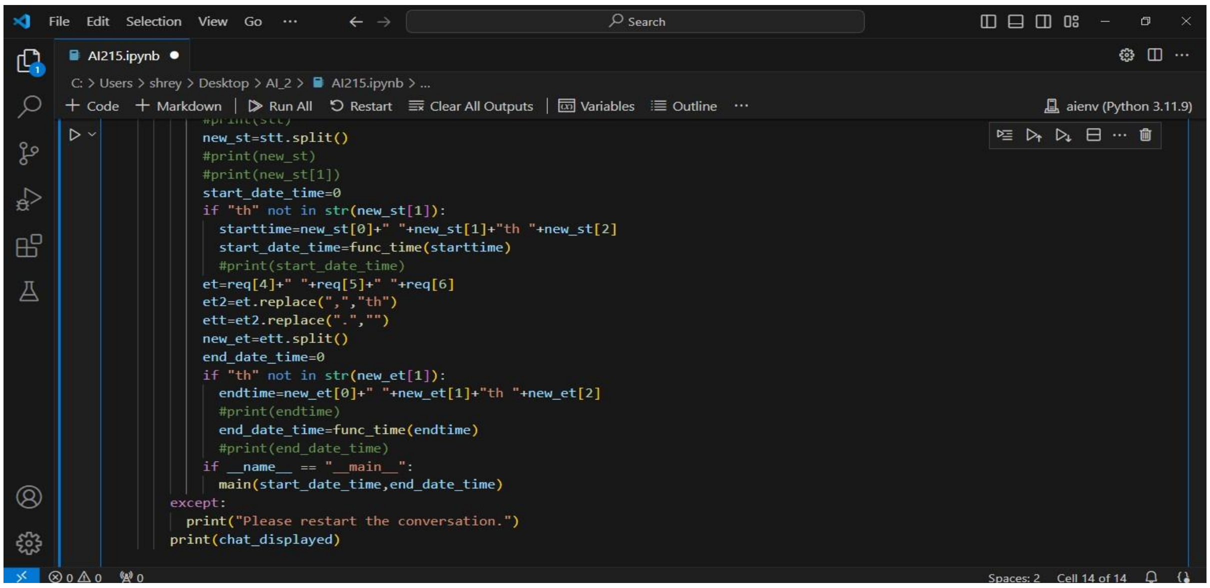
```python
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

SCOPES_C = ["https://www.googleapis.com/auth/calendar"]
creds = None
  # The file token.json stores the user's access and refresh tokens, and is
  # created automatically when the authorization flow completes for the first
  # time.
if os.path.exists("token_cal.json"):
    creds = Credentials.from_authorized_user_file("token_cal.json", SCOPES_C)
  # If there are no (valid) credentials available, let the user log in.
if not creds or not creds.valid:
      if creds and creds.expired and creds.refresh_token:
        creds.refresh(Request())
      else:
        flow = InstalledAppFlow.from_client_secrets_file(
          "credentials.json", SCOPES_C
        )
        creds = flow.run_local_server(port=0)
    # Save the credentials for the next run
      with open("token_cal.json", "w") as token:
          token.write(creds.to_json())
service_cal = build("calendar", "v3", credentials=creds)
def main(start_date_time, end_date_time):
```


AI215.ipynb
C: > Users > shrey > Desktop > AI_2 > AI215.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline
aienv (Python 3.11.9)
Spaces: 2 Cell 14 of 14

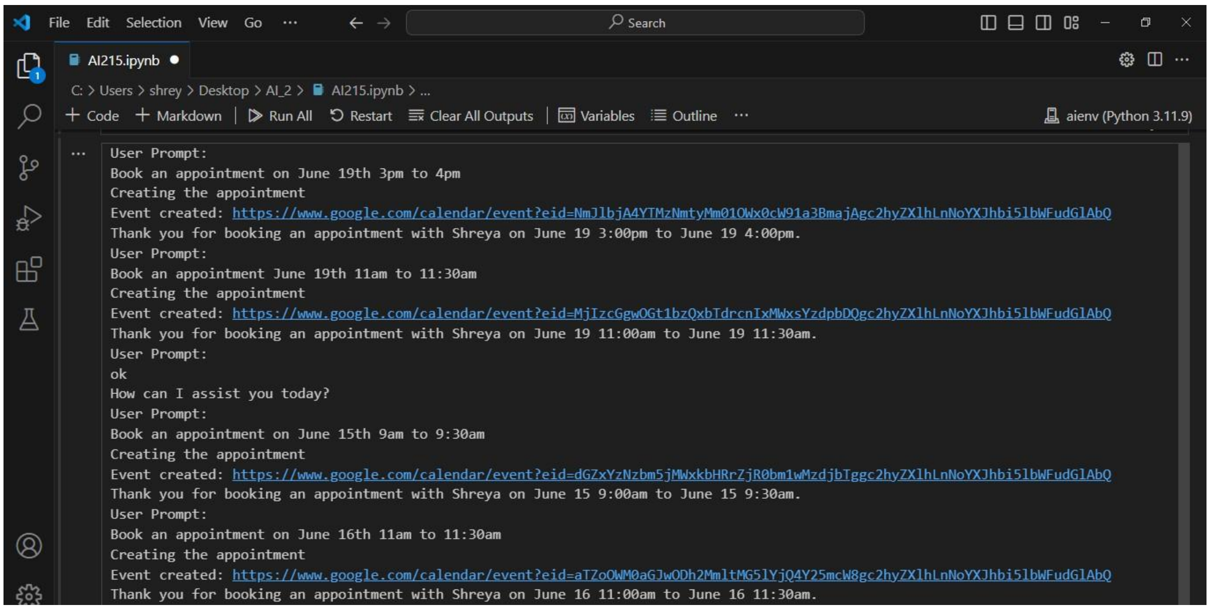
```
service_cal = build("calendar", "v3", credentials=creds)
def main(start_date_time, end_date_time):
  """Shows basic usage of the Google Calendar API.
  Prints the start and name of the next 10 events on the user's calendar.
  """
  try:
    print("Creating the appointment")

    event ={
        'summary': 'Chatbot Booked Meeting',
        'start': {
          'dateTime': start_date_time,
          'timeZone': 'America/Los_Angeles',
        },
        'end': {
          'dateTime': end_date_time,
          'timeZone': 'America/Los_Angeles',
        },
    }
    event = service_cal.events().insert(calendarId='primary', body=event).execute()
    print('Event created: %s' % (event.get('htmlLink')))

  except HttpError as error:
    print(f"An error occurred: {error}")
```


Explorer (Ctrl+Shift+E) - 1 unsaved file
C: > Users > shrey > Desktop > AI_2 > AI215.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline
aienv (Python 3.11.9)
Spaces: 2 Cell 14 of 14

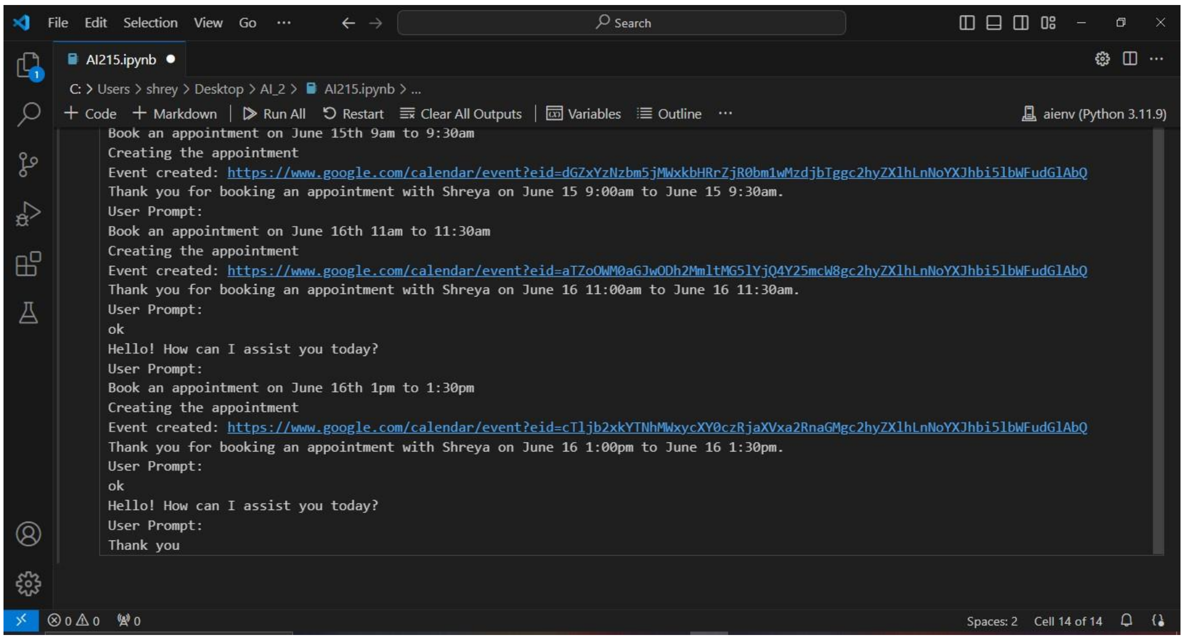
```
os.environ["OPENAI_API_KEY"] = "Enter your openai API key here"

while 1:
    query = input("Enter your prompt: ")
    print("User Prompt: ")
    print(query)
    if(str(query)=="Thank you"):
      break
    loader = TextLoader('filetry.txt')
    index = VectorstoreIndexCreator(embedding=OpenAIEmbeddings()).from_loaders([loader])

    chat_displayed=index.query(query, llm = ChatOpenAI())
    try:
      if "Thank you" in chat_displayed:
        each_word=chat_displayed.split()
        req=each_word[-7:]
        #print(req)
        st=req[0]+" "+req[1]+" "+req[2]
        stt=st.replace(",","th")
        #print(stt)
        new_st=stt.split()
        #print(new_st)
        #print(new_st[1])
        start_date_time=0
```


AI215.ipynb
C: > Users > shrey > Desktop > AI_2 > AI215.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline
aienv (Python 3.11.9)
Spaces: 2 Cell 14 of 14

```
        new_st=stt.split()
        #print(new_st)
        #print(new_st[1])
        start_date_time=0
        if "th" not in str(new_st[1]):
          starttime=new_st[0]+" "+new_st[1]+"th "+new_st[2]
          start_date_time=func_time(starttime)
          #print(start_date_time)
        et=req[4]+" "+req[5]+" "+req[6]
        et2=et.replace(",","th")
        ett=et2.replace(".","")
        new_et=ett.split()
        end_date_time=0
        if "th" not in str(new_et[1]):
          endtime=new_et[0]+" "+new_et[1]+"th "+new_et[2]
          #print(endtime)
          end_date_time=func_time(endtime)
          #print(end_date_time)
        if __name__ == "__main__":
          main(start_date_time,end_date_time)
    except:
      print("Please restart the conversation.")
    print(chat_displayed)
```



```
User Prompt:
Book an appointment on June 19th 3pm to 4pm
Creating the appointment
Event created: https://www.google.com/calendar/event?eid=NmJlbjA4YTMzNmtyMm01OWx0cW91a3BmajAgc2hyZXlhLnNoYXJhbi5lbWFudGlAbQ
Thank you for booking an appointment with Shreya on June 19 3:00pm to June 19 4:00pm.
User Prompt:
Book an appointment June 19th 11am to 11:30am
Creating the appointment
Event created: https://www.google.com/calendar/event?eid=MjIzcGgwOGt1bzQxbTdrcnIxMWxsYzdpbDQgc2hyZXlhLnNoYXJhbi5lbWFudGlAbQ
Thank you for booking an appointment with Shreya on June 19 11:00am to June 19 11:30am.
User Prompt:
ok
How can I assist you today?
User Prompt:
Book an appointment on June 15th 9am to 9:30am
Creating the appointment
Event created: https://www.google.com/calendar/event?eid=dGZxYzNzbm5jMWxkbHRrZjR0bm1wMzdjbTggc2hyZXlhLnNoYXJhbi5lbWFudGlAbQ
Thank you for booking an appointment with Shreya on June 15 9:00am to June 15 9:30am.
User Prompt:
Book an appointment on June 16th 11am to 11:30am
Creating the appointment
Event created: https://www.google.com/calendar/event?eid=aTZoOWM0aGJwODh2MmltMG5lYjQ4Y25mcW8gc2hyZXlhLnNoYXJhbi5lbWFudGlAbQ
Thank you for booking an appointment with Shreya on June 16 11:00am to June 16 11:30am.
```



```
Book an appointment on June 15th 9am to 9:30am
Creating the appointment
Event created: https://www.google.com/calendar/event?eid=dGZxYzNzbm5jMWxkbHRrZjR0bm1wMzdjbTggc2hyZXlhLnNoYXJhbi5lbWFudGlAbQ
Thank you for booking an appointment with Shreya on June 15 9:00am to June 15 9:30am.
User Prompt:
Book an appointment on June 16th 11am to 11:30am
Creating the appointment
Event created: https://www.google.com/calendar/event?eid=aTZoOWM0aGJwODh2MmltMG5lYjQ4Y25mcW8gc2hyZXlhLnNoYXJhbi5lbWFudGlAbQ
Thank you for booking an appointment with Shreya on June 16 11:00am to June 16 11:30am.
User Prompt:
ok
Hello! How can I assist you today?
User Prompt:
Book an appointment on June 16th 1pm to 1:30pm
Creating the appointment
Event created: https://www.google.com/calendar/event?eid=cTljb2xkYTNhMWxycXY0czRjaXVxa2RnaGMgc2hyZXlhLnNoYXJhbi5lbWFudGlAbQ
Thank you for booking an appointment with Shreya on June 16 1:00pm to June 16 1:30pm.
User Prompt:
ok
Hello! How can I assist you today?
User Prompt:
Thank you
```