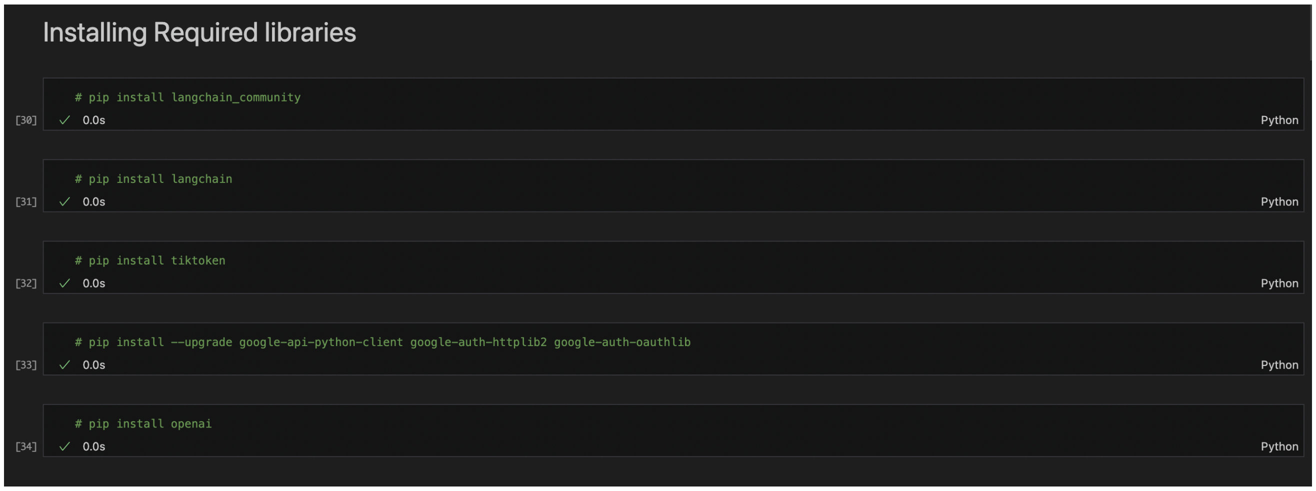
CSC 215 FINAL PROJECT

CALENDER LLM AND CHATGPT

-Lakshmi Tejaswi Devarapalli

Installed libraries:

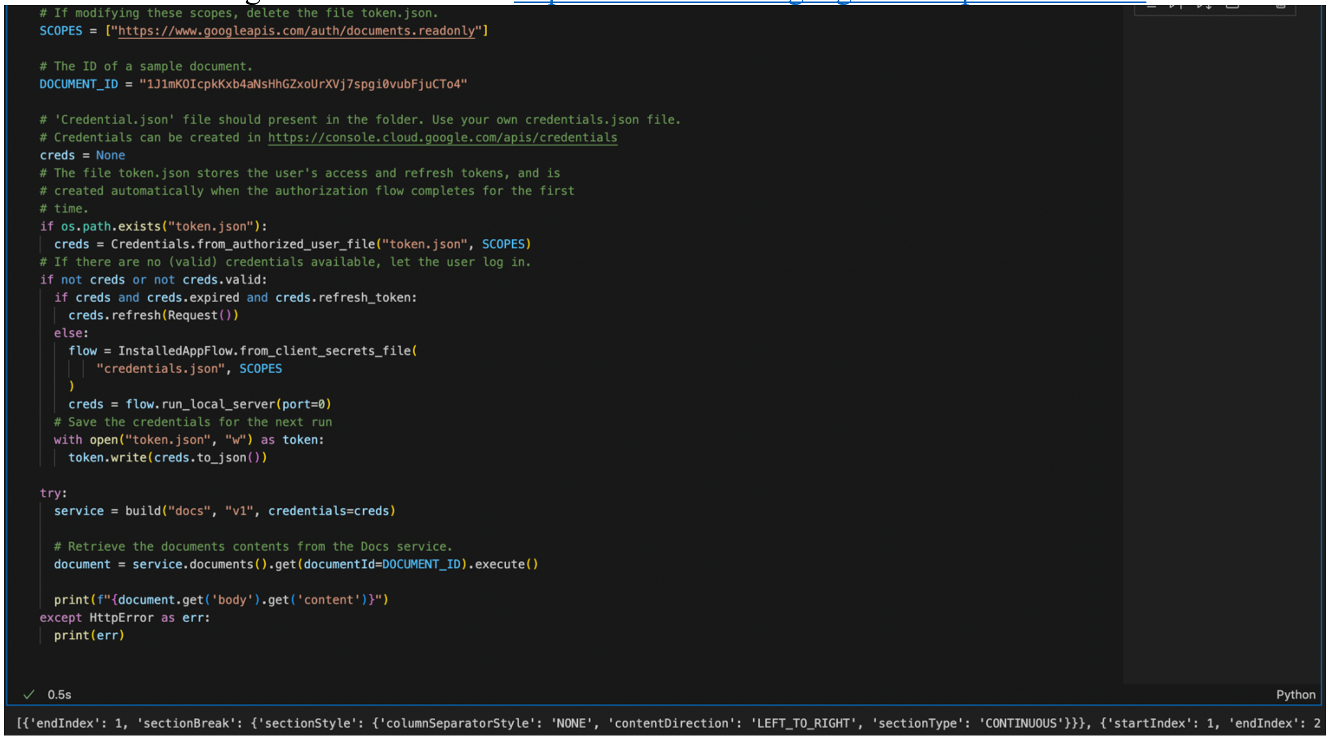


Installing Required libraries

```
[30]  # pip install langchain_community
      ✓ 0.0s                                                          Python

[31]  # pip install langchain
      ✓ 0.0s                                                          Python

[32]  # pip install tiktoken
      ✓ 0.0s                                                          Python

[33]  # pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
      ✓ 0.0s                                                          Python

[34]  # pip install openai
      ✓ 0.0s                                                          Python
```

To Fetch text from Google doc using Google doc API:

Your own 'credentials.json' file should be available in the folder.
Link to create Google API Credentials: https://console.cloud.google.com/apis/credentials



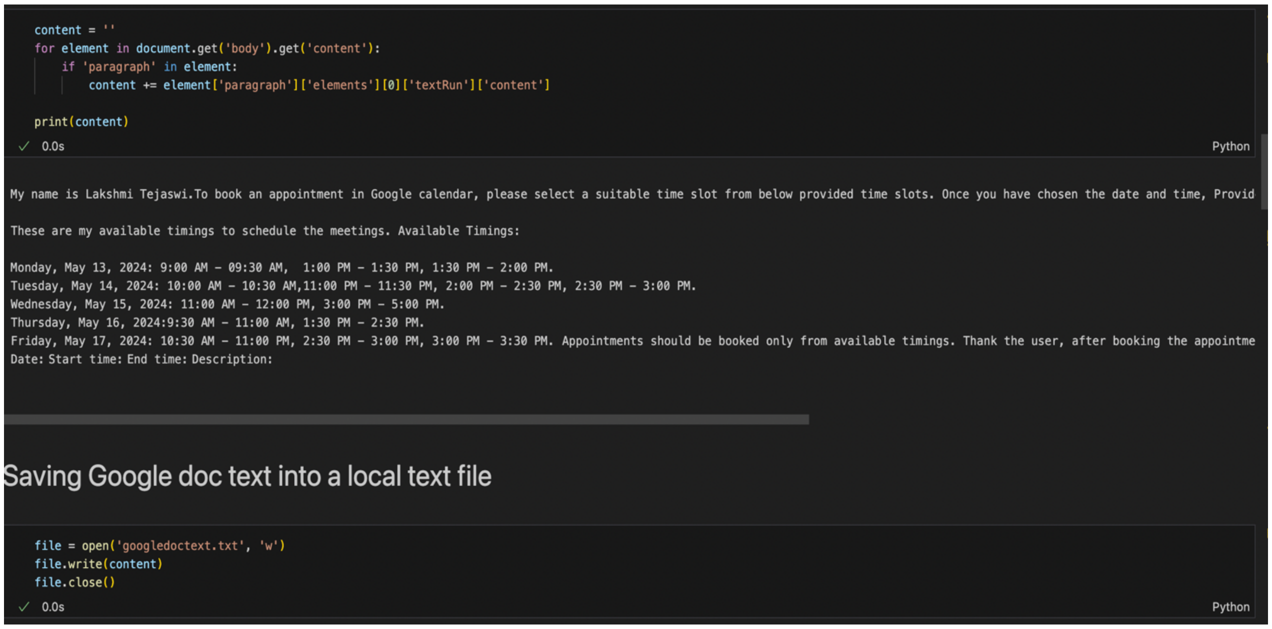
```python
# If modifying these scopes, delete the file token.json.
SCOPES = ["https://www.googleapis.com/auth/documents.readonly"]

# The ID of a sample document.
DOCUMENT_ID = "1J1mKOIcpkKxb4aNsHhGZxoUrXVj7spgi0vubFjuCTo4"

# 'Credential.json' file should present in the folder. Use your own credentials.json file.
# Credentials can be created in https://console.cloud.google.com/apis/credentials
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.json"):
  creds = Credentials.from_authorized_user_file("token.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.json", "w") as token:
    token.write(creds.to_json())

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

  # Retrieve the documents contents from the Docs service.
  document = service.documents().get(documentId=DOCUMENT_ID).execute()

  print(f"{document.get('body').get('content')}")
except HttpError as err:
  print(err)
```

```
✓ 0.5s                                                          Python
[{'endIndex': 1, 'sectionBreak': {'sectionStyle': {'columnSeparatorStyle': 'NONE', 'contentDirection': 'LEFT_TO_RIGHT', 'sectionType': 'CONTINUOUS'}}}, {'startIndex': 1, 'endIndex': 2
```

My Google doc link:
https://docs.google.com/document/d/1J1mKOIcpkKxb4aNsHhGZxoUrXVj7spgi0vubFjuCTo4/edit

Documented code for Google doc API: https://developers.google.com/calendar/api/quickstart/python

Getting only the text from google doc service into a variable and saving it to local text file:



```
content = ''
for element in document.get('body').get('content'):
    if 'paragraph' in element:
        content += element['paragraph']['elements'][0]['textRun']['content']

print(content)
```

✓ 0.0s Python

```
My name is Lakshmi Tejaswi.To book an appointment in Google calendar, please select a suitable time slot from below provided time slots. Once you have chosen the date and time, Provid

These are my available timings to schedule the meetings. Available Timings:

Monday, May 13, 2024: 9:00 AM - 09:30 AM,  1:00 PM - 1:30 PM, 1:30 PM - 2:00 PM.
Tuesday, May 14, 2024: 10:00 AM - 10:30 AM,11:00 PM - 11:30 PM, 2:00 PM - 2:30 PM, 2:30 PM - 3:00 PM.
Wednesday, May 15, 2024: 11:00 AM - 12:00 PM, 3:00 PM - 5:00 PM.
Thursday, May 16, 2024:9:30 AM - 11:00 AM, 1:30 PM - 2:30 PM.
Friday, May 17, 2024: 10:30 AM - 11:00 PM, 2:30 PM - 3:00 PM, 3:00 PM - 3:30 PM. Appointments should be booked only from available timings. Thank the user, after booking the appointme
Date: Start time: End time: Description:
```

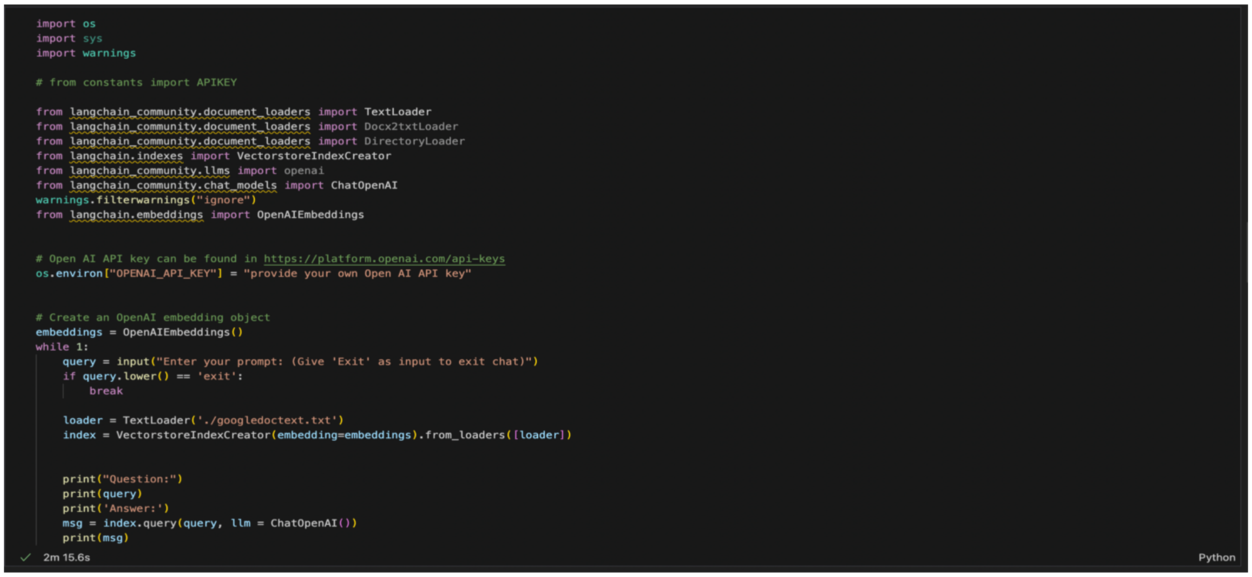
Saving Google doc text into a local text file

```
file = open('googledoctext.txt', 'w')
file.write(content)
file.close()
```

✓ 0.0s Python

Creating our own chatgpt based on the data in our document:

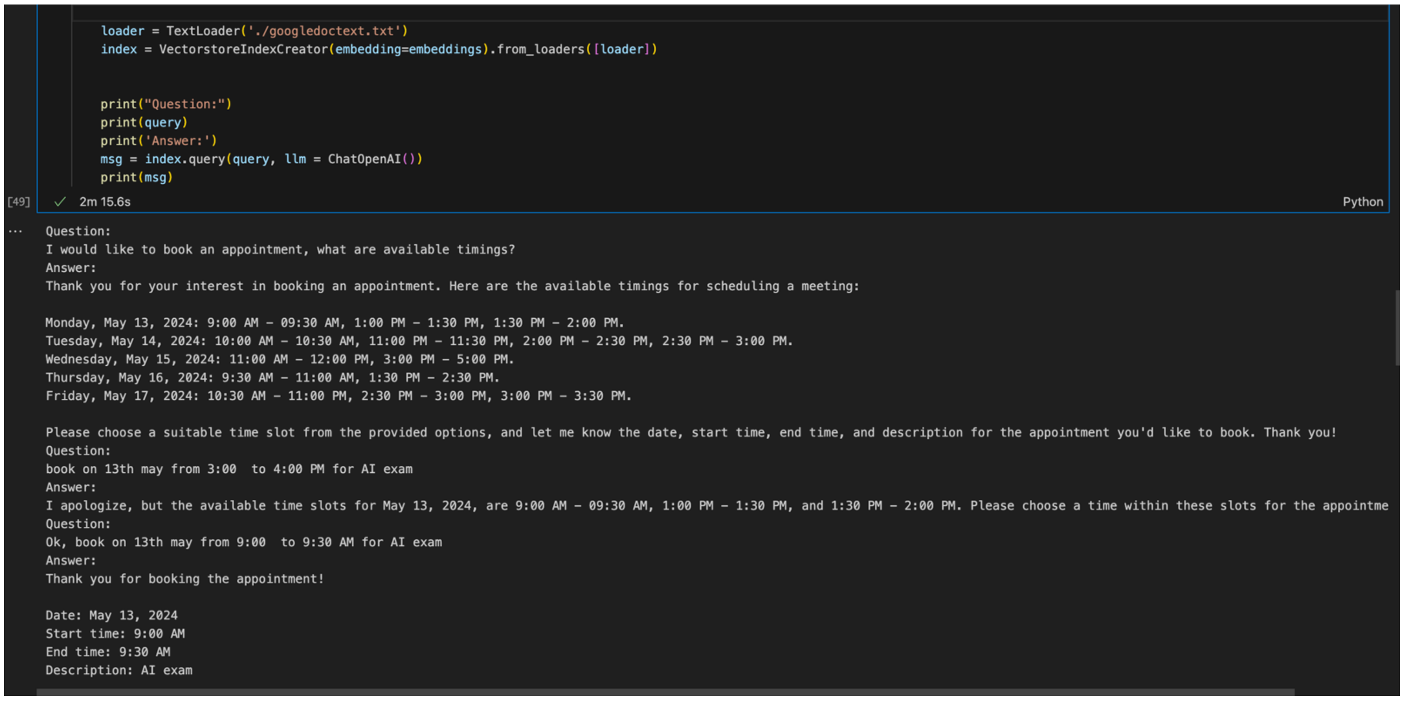
Provide your own Open AI API key in the below code.
Open AI API key billing Link: https://platform.openai.com/settings/organization/billing/overview
Open AI API key Link: https://platform.openai.com/api-keys



```
import os
import sys
import warnings

# from constants import APIKEY

from langchain_community.document_loaders import TextLoader
from langchain_community.document_loaders import Docx2txtLoader
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 langchain.embeddings import OpenAIEmbeddings


# Open AI API key can be found in https://platform.openai.com/api-keys
os.environ["OPENAI_API_KEY"] = "provide your own Open AI API key"


# Create an OpenAI embedding object
embeddings = OpenAIEmbeddings()
while 1:
    query = input("Enter your prompt: (Give 'Exit' as input to exit chat)")
    if query.lower() == 'exit':
        break

    loader = TextLoader('./googledoctext.txt')
    index = VectorstoreIndexCreator(embedding=embeddings).from_loaders([loader])


    print("Question:")
    print(query)
    print('Answer:')
    msg = index.query(query, llm = ChatOpenAI())
    print(msg)
```

✓ 2m 15.6s Python

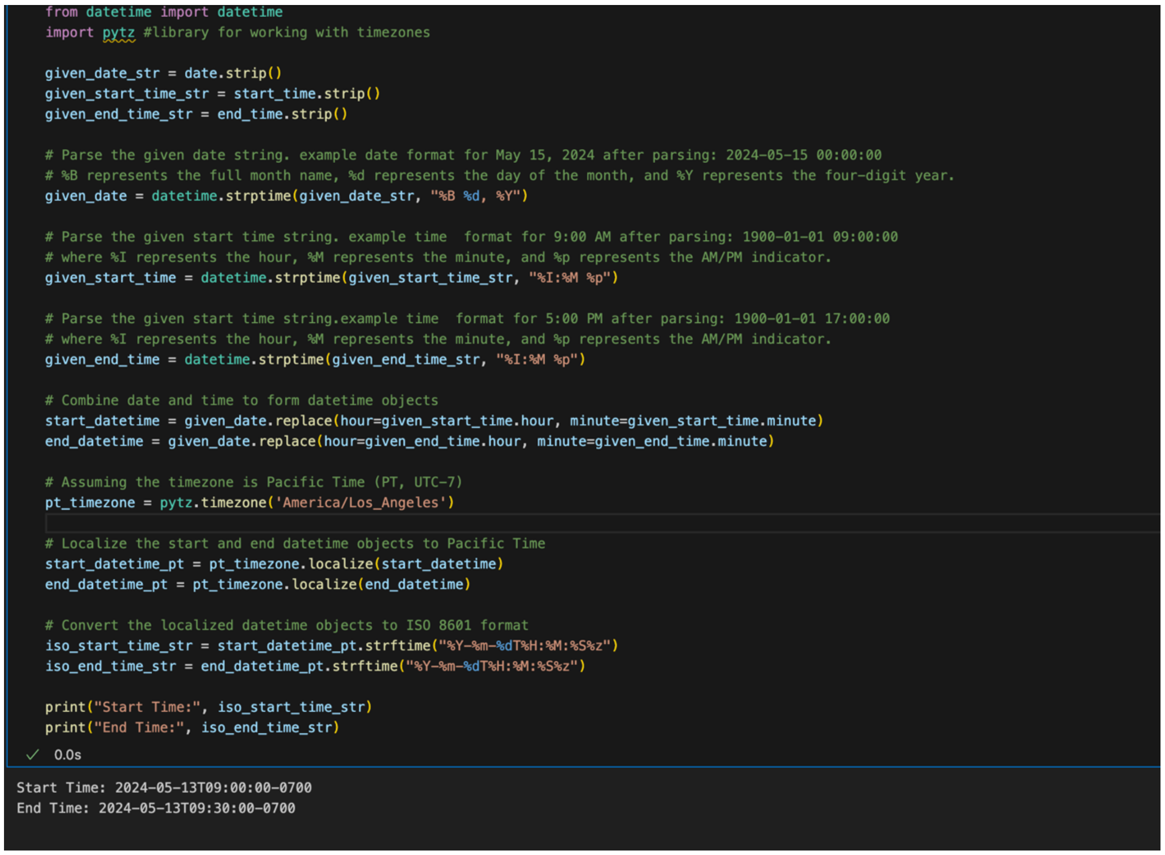
Example Chat to book an appointment:



```
    loader = TextLoader('./googledoctext.txt')
    index = VectorstoreIndexCreator(embedding=embeddings).from_loaders([loader])


    print("Question:")
    print(query)
    print('Answer:')
    msg = index.query(query, llm = ChatOpenAI())
    print(msg)
[49]  ✓ 2m 15.6s                                                                    Python
```

```
Question:
I would like to book an appointment, what are available timings?
Answer:
Thank you for your interest in booking an appointment. Here are the available timings for scheduling a meeting:

Monday, May 13, 2024: 9:00 AM - 09:30 AM, 1:00 PM - 1:30 PM, 1:30 PM - 2:00 PM.
Tuesday, May 14, 2024: 10:00 AM - 10:30 AM, 11:00 PM - 11:30 PM, 2:00 PM - 2:30 PM, 2:30 PM - 3:00 PM.
Wednesday, May 15, 2024: 11:00 AM - 12:00 PM, 3:00 PM - 5:00 PM.
Thursday, May 16, 2024: 9:30 AM - 11:00 AM, 1:30 PM - 2:30 PM.
Friday, May 17, 2024: 10:30 AM - 11:00 PM, 2:30 PM - 3:00 PM, 3:00 PM - 3:30 PM.

Please choose a suitable time slot from the provided options, and let me know the date, start time, end time, and description for the appointment you'd like to book. Thank you!
Question:
book on 13th may from 3:00  to 4:00 PM for AI exam
Answer:
I apologize, but the available time slots for May 13, 2024, are 9:00 AM - 09:30 AM, 1:00 PM - 1:30 PM, and 1:30 PM - 2:00 PM. Please choose a time within these slots for the appointme
Question:
Ok, book on 13th may from 9:00  to 9:30 AM for AI exam
Answer:
Thank you for booking the appointment!

Date: May 13, 2024
Start time: 9:00 AM
End time: 9:30 AM
Description: AI exam
```

Converting Time to ISO format:

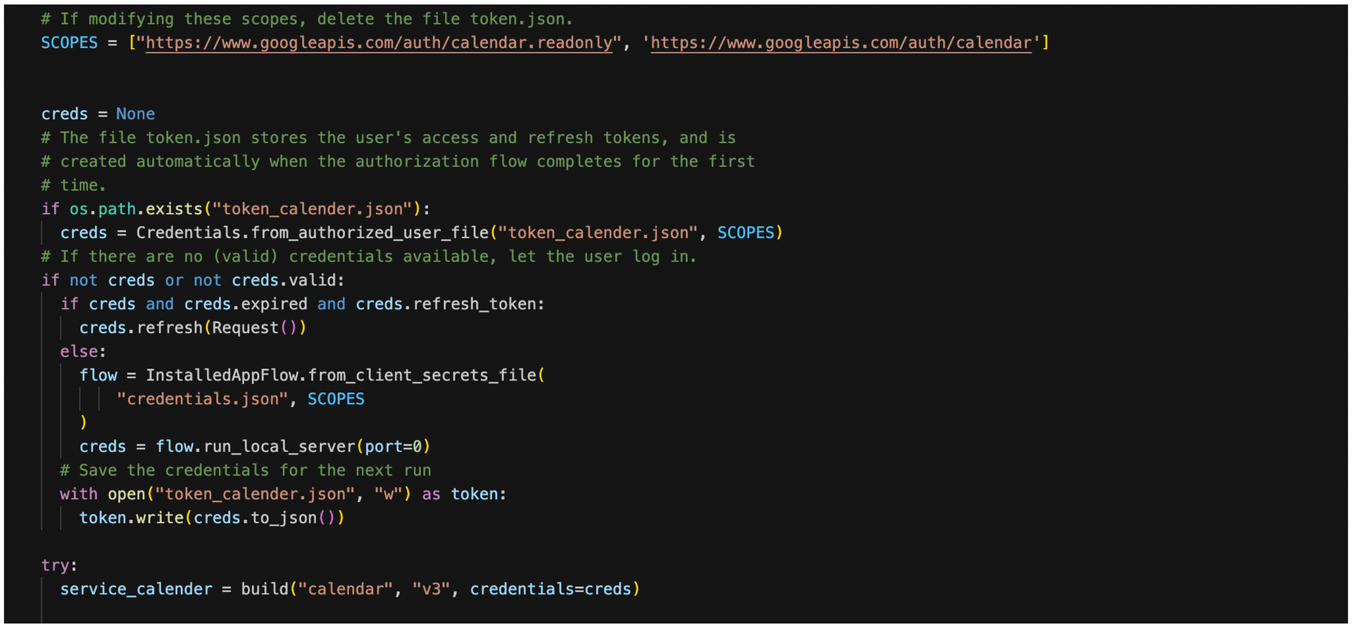


```python
from datetime import datetime
import pytz #library for working with timezones

given_date_str = date.strip()
given_start_time_str = start_time.strip()
given_end_time_str = end_time.strip()

# Parse the given date string. example date format for May 15, 2024 after parsing: 2024-05-15 00:00:00
# %B represents the full month name, %d represents the day of the month, and %Y represents the four-digit year.
given_date = datetime.strptime(given_date_str, "%B %d, %Y")

# Parse the given start time string. example time  format for 9:00 AM after parsing: 1900-01-01 09:00:00
# where %I represents the hour, %M represents the minute, and %p represents the AM/PM indicator.
given_start_time = datetime.strptime(given_start_time_str, "%I:%M %p")

# Parse the given start time string.example time  format for 5:00 PM after parsing: 1900-01-01 17:00:00
# where %I represents the hour, %M represents the minute, and %p represents the AM/PM indicator.
given_end_time = datetime.strptime(given_end_time_str, "%I:%M %p")

# Combine date and time to form datetime objects
start_datetime = given_date.replace(hour=given_start_time.hour, minute=given_start_time.minute)
end_datetime = given_date.replace(hour=given_end_time.hour, minute=given_end_time.minute)

# Assuming the timezone is Pacific Time (PT, UTC-7)
pt_timezone = pytz.timezone('America/Los_Angeles')

# Localize the start and end datetime objects to Pacific Time
start_datetime_pt = pt_timezone.localize(start_datetime)
end_datetime_pt = pt_timezone.localize(end_datetime)

# Convert the localized datetime objects to ISO 8601 format
iso_start_time_str = start_datetime_pt.strftime("%Y-%m-%dT%H:%M:%S%z")
iso_end_time_str = end_datetime_pt.strftime("%Y-%m-%dT%H:%M:%S%z")

print("Start Time:", iso_start_time_str)
print("End Time:", iso_end_time_str)
✓ 0.0s
```

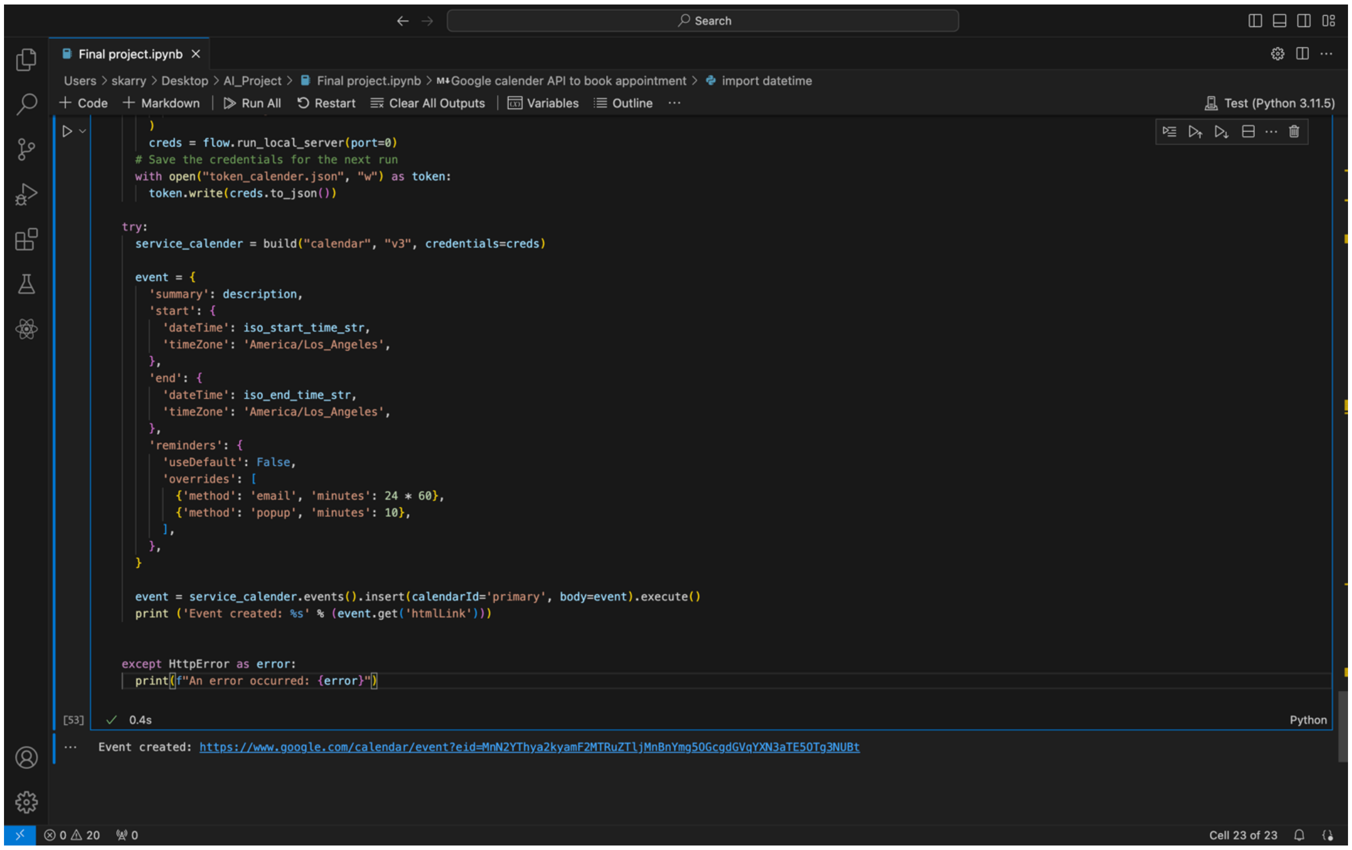
```
Start Time: 2024-05-13T09:00:00-0700
End Time: 2024-05-13T09:30:00-0700
```

Google Calendar API to book an Event:

Documented code Google calendar API: https://developers.google.com/calendar/api/quickstart/python



```python
# If modifying these scopes, delete the file token.json.
SCOPES = ["https://www.googleapis.com/auth/calendar.readonly", '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_calender.json"):
  creds = Credentials.from_authorized_user_file("token_calender.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_calender.json", "w") as token:
    token.write(creds.to_json())

try:
  service_calender = build("calendar", "v3", credentials=creds)
```

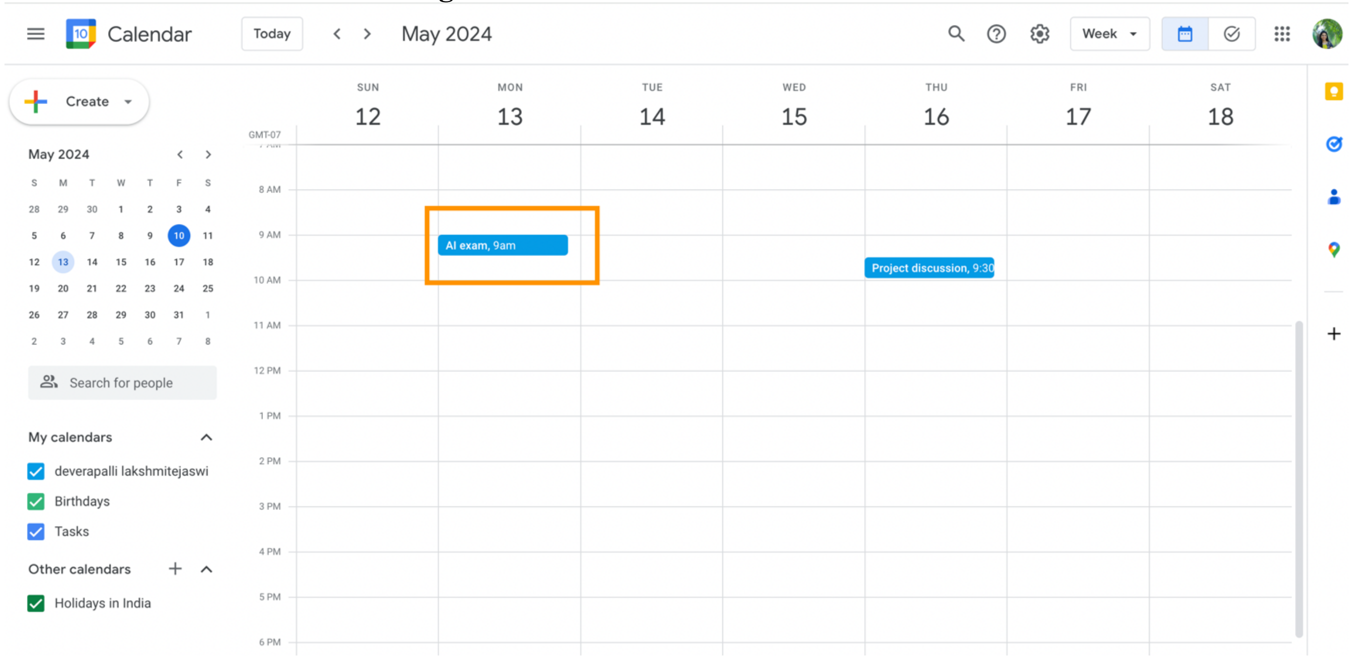


Users > skarry > Desktop > AI_Project > Final project.ipynb > Google calender API to book appointment > import datetime

```python
    )
    creds = flow.run_local_server(port=0)
  # Save the credentials for the next run
  with open("token_calender.json", "w") as token:
    token.write(creds.to_json())

try:
  service_calender = build("calendar", "v3", credentials=creds)

  event = {
    'summary': description,
    'start': {
      'dateTime': iso_start_time_str,
      'timeZone': 'America/Los_Angeles',
    },
    'end': {
      'dateTime': iso_end_time_str,
      'timeZone': 'America/Los_Angeles',
    },
    'reminders': {
      'useDefault': False,
      'overrides': [
        {'method': 'email', 'minutes': 24 * 60},
        {'method': 'popup', 'minutes': 10},
      ],
    },
  }

  event = service_calender.events().insert(calendarId='primary', body=event).execute()
  print ('Event created: %s' % (event.get('htmlLink')))


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

```
[53]  ✓ 0.4s

Event created: https://www.google.com/calendar/event?eid=MnN2YThya2kyamF2MTRuZTljMnBnYmg5OGcgdGVqYXN3aTE5OTg3NUBt
```

Created Event Reflected in Google Calendar:


Calendar
Today
May 2024
Week
Create
May 2024
S M T W T F S
Search for people
My calendars
deverapalli lakshmitejaswi
Birthdays
Tasks
Other calendars
Holidays in India
GMT-07
SUN 12
MON 13
TUE 14
WED 15
THU 16
FRI 17
SAT 18
8 AM
9 AM
10 AM
11 AM
12 PM
1 PM
2 PM
3 PM
4 PM
5 PM
6 PM
AI exam, 9am
Project discussion, 9:30