

Title: Personal Daily Meeting Viewer using Google Calendar API.

Use Case: This demo personal project was developed as a lightweight, customizable solution that allows users to view their calendar (e.g. Google Calendar) events for any selected day via a local web interface or could be hosted in AWS cloud.

The solution could be applied to Outlook 365 and somethings in the code needs to be changed for Outlook POC. In my case I have used Google Calendar.

For example:

1. A manager can share a local dashboard with a PA to review meetings for specific dates.
2. A remote engineer can check the daily meetings in a browser without opening Gmail.

Steps:

1. Enable the Calendar API
2. Navigate to the Link: <https://console.developers.google.com/>
3. Login with your personal Gmail account
4. You can create a new project or select an existing project
5. Would recommend a new project, you can delete after the proof of concept



6. Provide a Project Name

A screenshot of the 'Create Project' form in the Google Cloud Platform console. The form has a yellow header bar. It contains three input fields: 'Project name *' with the value 'Personal-Daily-Meeting-Viewer', 'Project ID *' with the value 'personal-daily-meeting-viewer', and 'Location *' with the value 'No organization'. Below the 'Location' field is a 'Browse' button. At the bottom, there is a 'Parent organization or folder' field and two buttons: 'Create' and 'Cancel'. The 'Create' button is highlighted with a red box.

7. Once the project has been created, make sure the project has been selected



8. Now click on Library

9. Locate Google Calendar API



Google Calendar API

Google Enterprise API ⓘ

With the Calendar API, you can display, create and modify calendar events as well as work with many other calendar-related objects, such as calendars or access controls.

10. Enable the API



Google Calendar API

Google Enterprise API

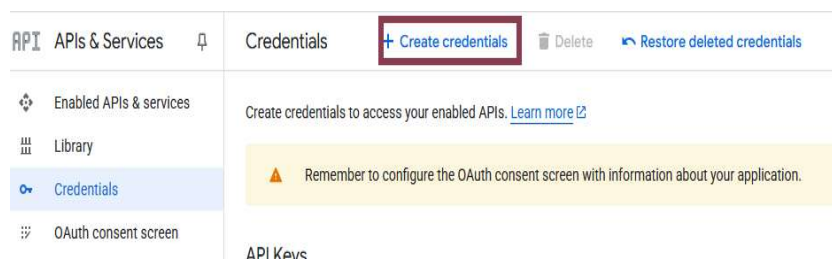
Manage calendars and events in Google Calendar

Enable

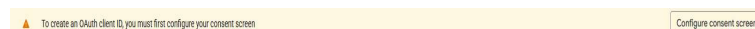
Try this API ↗

11. Now, click on create credentials

12. Select OAuth Client ID (This is personal project)



13. Click on Configure Consent Screen



14. In the overview provide the following name

- App Information
- Your personal email address
- Click on Next

1 App Information

App name *

Personal Daily Meeting Viewer

The name of the app asking for consent

User support email *

ail.com

For users to contact you with questions about their consent. [Learn more](#)

Next

15. This is only test mode, and I will be the only user testing it. So, select external and click on Next

2 Audience

☐ Internal ⓘ

Only available to users within your organization. You will not need to submit your app for verification. [Learn more about user type](#)

☒ External ⓘ

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. [Learn more about user type](#)

Next

16. In the next screen, provide your personal Gmail address

3 Contact Information

Email addresses *

ail.com

These email addresses are for Google to notify you about any changes to your project.

Next

17. Check the box and click on continue

4 Finish

☒ I agree to the [Google API Services: User Data Policy](#)

Continue

18. Click on Create.

19. Now create OAuth Client

20. Select Desktop App

21. Provide a Name

22. Click on Create

← Create OAuth client ID

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information. [Learn more](#) about OAuth client types.

Application type
Desktop app

Name
Personal Daily Meeting Viewer

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

Note: It may take 5 minutes to a few hours for settings to take effect

Create Cancel

23. Download the JSON in a secure area

24. Make a copy

25. Rename one to credentials.json to shorten the file name

26. Now comes the interesting path

27. Open jupyter notebook

28. The credentials json will be in the same path as your jupyter notebook

29. Install the required libraries

- a. `!pip install --upgrade google-api-python-client google-auth-http2 google-auth-oauthlib`
 - i. Http2 will handle the HTTP requests
 - ii. Oauthlib will handle the OAuth token
 - iii. Python client is the API client library

30. Initially, the project demo was to integrate with WhatsApp and for the sandbox to get enabled will take 72 hours. This is a trial version. Will plan next week integration with whatsapp and adopted a different approach

```
!pip install --upgrade google-api-python-client google-auth-http2 google-auth-oauthlib
```

31. The approach was to launch locally and also installed the following library

```
!pip install flask flask-cors
```

32. Run the python code to authenticate with Google Calendar API

```
# Run this to authenticate with Google Calendar API
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
import os

print("Google Calendar API connected.")
```

Google Calendar API connected.

33. Each portion is broken into smaller pieces just like projects to ensure key result is met
34. If you run into access issue or not authorized pop up message. Remember , while creating the project, selected external for testing purpose and not internal. This error is quite common.
- Go to the Google Cloud Console
 - <https://console.cloud.google.com/>
 - Under user type it is set to external and there is no need to publish it because it is a personal project
 - In the Test Users Section – Add your Personal Gmail address. It will be same one you will use when executing the above python code
 - Save the changes
 - Restart the kernel and run the cell again
 - Once successfully executed, output will be Google Calendar API connected successfully
35. Now fetch and display events using the API from today

```
import pytz

# Call the function
today_events = get_today_events(service)

Found 5 event(s) for today:
2025-05-20T07:00:00+05:30 - [redacted] Training
2025-05-20T10:00:00+05:30 - [redacted] Foundation
2025-05-20T13:00:00+05:30 - [redacted] Cloud Certification
2025-05-20T14:00:00+05:30 - [redacted] Assignments
2025-05-20T15:30:00+05:30 - [redacted]
```

36. The today's events are displayed, display the start time and title
37. As mentioned earlier, I am not using WhatsApp messages for the POC, creating a Html form to display the form

```
from flask import Flask, request, jsonify, render_template_string
from threading import Thread
from datetime import datetime, timedelta
import pytz


</html>


@app.route('/')
def index():
```

38. The output will provide a URL. Copy and paste it in the browser.





```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
```

39. Select the date and events are displayed

 **View My Google Calendar Events**



Events for 2025-05-21

2025-05-21T07:00:00+05:30	 Training
2025-05-21T10:00:00+05:30	 Training
2025-05-21T13:00:00+05:30	 Foundation
2025-05-21T15:30:00+05:30	 Assignments