Indian Institute of Technology Jodhpur

B.Tech Project

Semester IV

Automated Gmail Virtual Assistant

Author:
Mohit Mehta
(B14CS043)
Ashish Sahu
(B14CS009)

Mentor:
Dr. Venkata Ramana Badarla

Abstract

Our aim is to develop a software mainly for the class of students and faculties that reads all the Google mails and extracts the important information (like meeting, classes, tests, assignments etc.) and add all such events in your Google Calendar to give them timely notifications about these cases so that they do not miss out any important life events.
Notifications will be sent via Emails only at a specific time before the event. In this semester, the software is meant to extract the information only from a structured mail and in the next semester it will be made generalized for any type of a mail. The mail can contain any number of lines written in any manner but it should have the following structure somewhere in between:

Event Details: (details regarding event)
Start Date: (the date on which the event is going to start)
End Date: (the date on which the event is going to end)
Start Time: (time at which event is going to start)
End Time: (time at which the event is going to conclude)
Location: (venue i.e. where the event is supposed to happen)

Our software will detect this structure and extract the relevant and useful information. After that it will convert them into the desired format, add them into the calendar and the user will get the timely notifications.

1 Introduction

1.1 Problem Statement

Nowadays, every major process is done through emails. Businessman, faculty and even students do not have enough time to look into all the mails and respond to them. At the end of the day, people want to sort all the important mails so that they do not miss out the important events of their life.

A scanner is required for the mails which would detect the important schedules and add them to the calendar so that the user can get timely notifications. So, we intend to make software which can –

1. Scan all mails and get important information about meetings, assignments, tests, classes etc.
2. Sync the important dates with the calendar.
3. Remind the user at particular intervals.

1.2 Motivation and Scope

In many scenarios, mails about a particular work are received a long time before it is scheduled to be done but due to our lifestyle, we miss the timeline and sometimes forget to do that, which eventually leads to a great loss. So, it becomes very important to have something which can keep track of all important dates in the mail, sync them with calendar and remind us at particular intervals.

2 Literature Survey

- Natural Language toolkit (NLTK) is a leading platform for building Python programs to work with human language data. There is a book on NLTK named "Natural Language Processing with Python" written by Steven Bird, Ewan Klein, and Edward Loper.
- Gmail API created by Google lets us view and manage Gmail mailbox data like threads, messages, and labels.
- Google Calendar API lets us manage the calendars and events.
- ToDoIst is an existing chrome extension which lets us turn our important emails into actionable tasks, add to-do items for projects, set reminders and set deadlines.

3 Technology Used

- Language: Python for feature implementation as well as GUI development.

- Google APIs: Gmail API to get access to the user mails and calendar API to add the event in the calendar.

- Natural Language Toolkit (NLTK): To process natural language using Python.

- Platform: Linux for development. Windows for testing of the software.

4 Methodology

- We created a project named "Automated Gmail Virtual Assistant" in the Google Developers Console and turned on the APIs. OAuth 2.0 protocol is used for authentication and authorization. So, we selected the type of credentials as OAuth Client ID which lets us take user consent so that our software can access the user's data.
- A script is written for the authentication and user consent process which, when the software is used for the first time, opens the browser window automatically to ask the user for the permission to access the mails and calendar (according to the scopes defined in the script) if he is logged in otherwise he is asked to login with his Gmail account first and then asked for permissions.

- If the user approves the access request, then the response contains an authorization code. If the user does not approve the request, the response contains an error message. After the web server receives the authorization code, it can exchange the authorization code for an access token. Now, a service object is build by calling the build function with the name and version of the API(i.e. Gmail API in this case) and the authorized Http object.

- The next step is to use make request to the Gmail API service to read mails using the service object. This was done in mainly three steps:-

- Getting List of Message IDs: Every message in Gmail has a unique Id associated with it. Now, to access a message we need to first get its Id. According to our software requirement, if it is running on the current day, it needs all the messages of the previous day. So, a script is written which gets the Id of all the mails received the previous day.

- Getting the messages using message IDs: Now that we have message Ids of the required messages, using that we can access all the messages through a python script having function which returns the array of messages.

- Converting the messages into readable format: The messages thus obtained are now converted into Multipurpose Internet Mail Extensions (MIME) which is an Internet standard that extends the format of email to support: Text in character sets other than ASCII. Non-text attachments: audio, video, images, application programs etc.

- We, now, have all the messages ready to be processed for extracting relevant data. It is assumed that the messages are structured having the data in prescribed format:-

  Event Details: (details regarding event)
  Start Date: (the date on which the event is going to start)
  End Date: (the date on which the event is going to end)
  Start Time: (time at which event is going to start)
  End Time: (time at which the event is going to conclude)
  Location: (venue i.e. where the event is supposed to happen)

The acceptable formats for the date and time are :-

1. 25th April 2016
2. 25 April 2016

1. 12:00 am
2. 12 am

3. 25th April,2016
4. 25 April,2016
5. April 25,2016
6. 25/04/2016

3. 12:00:00 am
4. 00:00

- The processing is done using the Natural Language Toolkit (NLTK) which is a platform for building Python programs to work with human language data. Using NLTK the messages are processed to get relevant information (start/end date, start/end Time, Location, Event Details).

- The google calendar accepts date and time in a particular format. A python script firstly converts the date and time into the format yyyy-mm-dd and 24 hours format respectively and using that, it is then converted into the format accepted by the calendar API.

- Now there is a function to add new event. It takes all the extracted information from mails and put it into the calendar according to the date and time. There are two cases for this:-

  Case1: If any event is already present in the calendar during the start and end time of the new event, then another script is called for sending the mail to the user telling him about the conflicting events. Eventually, after that the event is added into the calendar.

  Case2: If no event is already present in the calendar during the start and end time of the new event, the new event is directly added to the calendar.

- The reminders for the upcoming events are sent to registered users through mail a specific time before the event.

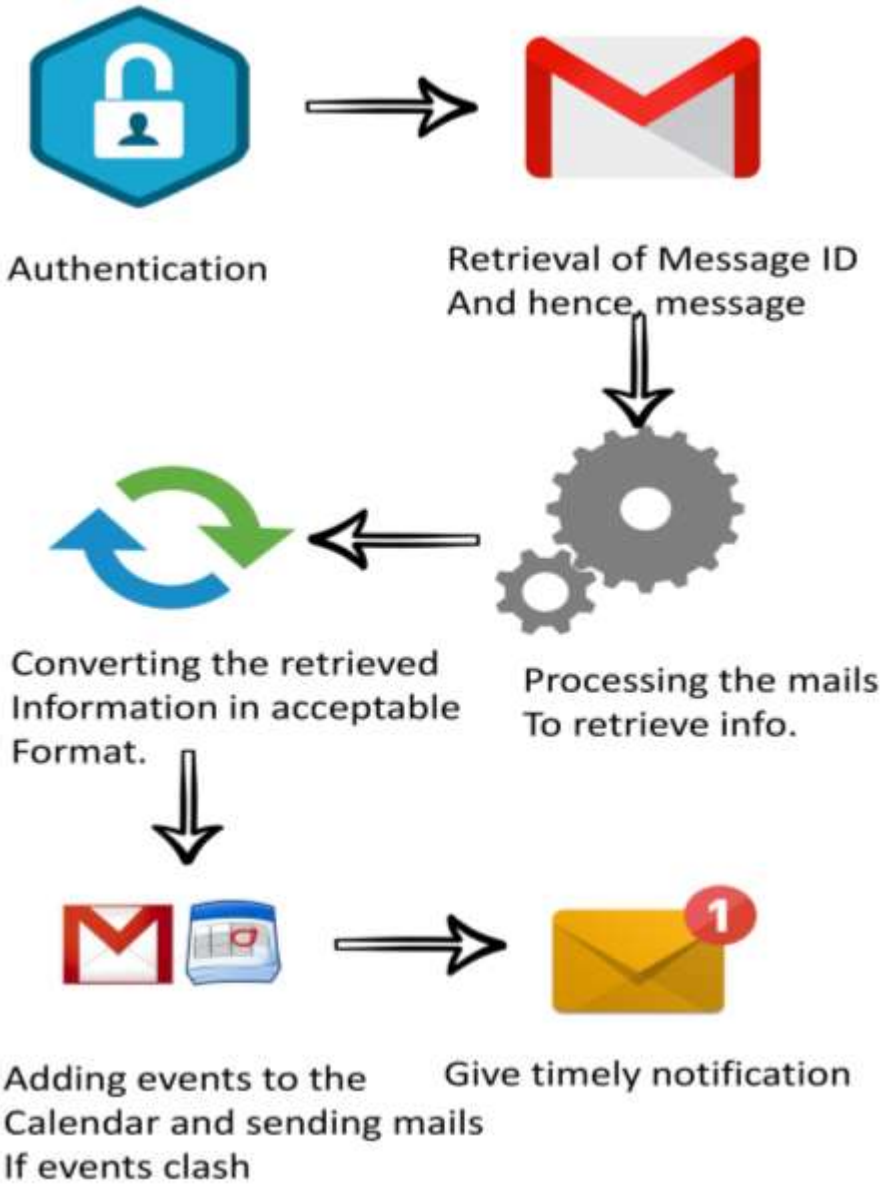

Figure 1Schematic diagram for the methodology used

5 Division of Work

The work was equally divided among both of us in such a way that both of us learn the best out of this project. All the new things were learnt by both of us. Learning python language was must for this project and it was a continuous process. Implementation part was equally divided between both of us. A brief description about the work division among both of us along with the time is illustrated below:

S No	Work	Timeline	Responsible
1	Getting familiar with OAuth 2.0 protocol		Both
2	Getting familiar with Gmail API and its features		Both
3	Authentication and User consent to access his Gmail and Google Calendar		Ashish Sahu
4	Getting familiar with Google Calendar API and its features		Mohit Mehta
5	Write a script to get the message IDs for the required (previous day's) messages		Ashish Sahu
6	Get the messages through message IDs and hence, convert them into MIME format		Ashish Sahu
7	Create a calendar event with dummy data		Mohit Mehta
8	Process the messages to extract important informations		Ashish Sahu
9	Converting the information extracted from the mail in acceptable format using python		Mohit Mehta
10	Linking the important information with the calendar and sending the reminders		Mohit Mehta
11	Python script to send email in case of conflicting events		Mohit Mehta
12	Integrating all the components		Ashish Sahu
13	Creating the GUI and installer for the software		Mohit Mehta
14	Make the software run in the system background		Ashish Sahu

	periodically		
15	Testing the software		Both

6 Experiments and Results

- We used many input test cases following the prescribed structure, differentiated in terms of the text written in the mail and the software worked fine.
- When the defined structure was not followed in the mails, the software could not process the mails for important information.
- Input mails were taken such that there was a clash in timings. The software recognized it and notified the user correctly.
- It is observed that the authentication process takes considerable amount of time as connecting to the google server for authentication is a lengthy process.
- Due to python being a slow language, sometimes when large numbers of mails are encountered the software took considerable time to give output. Though output was correct.
- Then the software was made to run in the system background periodically so that time taken is less and it worked better.
- The software gives best results when there is a high speed internet connection available and all the mails followed correct structure.

7 Conclusion

- This software can be used to keep track of all the important mails and add the important information to the calendar. And hence, give timely notification to the user through mail.

- This software can only work fine if proper structure in the mails is followed.

- The software requires high speed internet connection as the authentication process involves connecting to the google server.

- Due to python being slow language, the software takes considerable amount of time for large data input(above 25 mails).

8 References

[1] *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit (1st ed.) by Steven Bird, Ewan Klein, and Edward Loper and published by O'Reilly.*

[2] *Introduction to Gmail API by George Edwards*

[3] Gmail API overview: [Online] Available at *https://developers.google.com/gmail/api/quickstart/python*

[4] Calendar API overview: [Online] Available at *https://developers.google.com/google-apps/calendar/quickstart/python*

[5] Python Documentation: [Online] Available at *https://docs.python.org/2/tutorial/*

[6] Tkinter Documentation: [Online] Available at *https://docs.python.org/2/library/tkinter.html*