Spreadsheet Programming

Anilesh Bansal (22B0928)

Abstract

This is the Report for the CS104 Endsem project on Spreadsheet programming.

1 Description

This project is about using Google sheets API to read a spreadsheet and take some actions based on the data. A google form is made available which aims to mimic a virtual queue which can be used to schedule appointments and assign tokens to the form-fillers at a Visa Application Office.
Users who have filled the form receive a custom Gmail which includes their name, assigned token number, time and date of appointment among other things.

2 Implementation and Customization

The project was implemented using Google App scripts linked with SpreadsheetApp and MailApp class to take data from a spreadsheet and send emails.
I made a google form which takes in the Email Address, Name, Preferred Location, Preferred Slot and Gender of the form-filler and stores them in a spreadsheet with the sheetname "Form Responses" . Then I linked a App script Project (code_prefslot.gs) to this spreadsheet.
For the entire project, data is handled through the Header titles like "Name" and "Preferred Slot" instead of using column indexes [1]. This is to ensure that the code works even if we later on decide to take more information from the form-fillers, which may not be directly involved in assigning the tokens.
The code has been commented at various places to help in understanding. The various functions and their uses in the project are as follows:

2.1 removeDuplicates()

A copy of the sheet "Form Responses" is made and is renamed to "NoDuplicates" [2]. Then the code checks for duplicate entries by maintaining a uniqueData[ ] array and adding a element in it only if the same (Email Address, Name) do not exist in it already. In the end it writes this array to "NoDuplicates". This ensures that only the first response corresponding to a (Email Address, Name) is considered. [3]

2.2 TokenAssigner()

This function creates a new column "Assigned Token" in the sheet and gives out tokens by making Location-specific queues and taking preferred slot into consideration.
The array locationData[i] contains data who have their chosen location as location[i]. Then iterating over all locations, we start assigning tokens. Note that according to the problem statement, only the first 6 responses to that location should be assigned a token. So for the first 6 responses, if the preferred slot is unoccupied, the person is assigned that token and it's updated in the array tokens[ ]. If the slot is occupied/ the person has no preference, the person is pushed to the array unallocated[ ]. Once this is done for the first 6 responses, the unallocated[ ] data is assigned tokens over the remaining slots and its updated in the tokens[ ] array.
The rest of the people are given the token "-".
The "Assigned Token" are updated in the sheet "NoDuplicates" as well using the Write() function.

1	Email Address	Name	Preferred Location	Preferred Slot	Gender	Assigned Token
20	anileshbansal@gmail.cor	Gwen1	Chennai	11:30 AM - 12:00 PM	F	4
21	anileshbansal@gmail.cor	Gwen2	Chennai	No Preference	F	2
22	anileshbansal@gmail.cor	Gwen3	Chennai	11:00 - 11:30 AM	M	3
23	anileshbansal@gmail.cor	Gwen4	Chennai	10:00 - 10:30 AM	F	1
24	anileshbansal@gmail.cor	Gwen5	Chennai	10:00 - 10:30 AM	F	5
25	anileshbansal@gmail.cor	Gwen6	Chennai	10:00 - 10:30 AM	F	6
26	anileshbansal@gmail.cor	Gwen7	Chennai	10:00 - 10:30 AM	F	-

For example for the above data, first tokens 4, 3 and 1 are assigned for Gwen1, Gwen3 and Gwen4 according to their preference. Gwen2, Gwen5 and Gwen6 are pushed to unallocated[ ]. Then the unallocated array is traversed and they are assigned the remaining tokens 2, 5 and 6 respectively. all the remaining persons(i.e Gwen7) is given the token "-".

2.3 sendEmails()

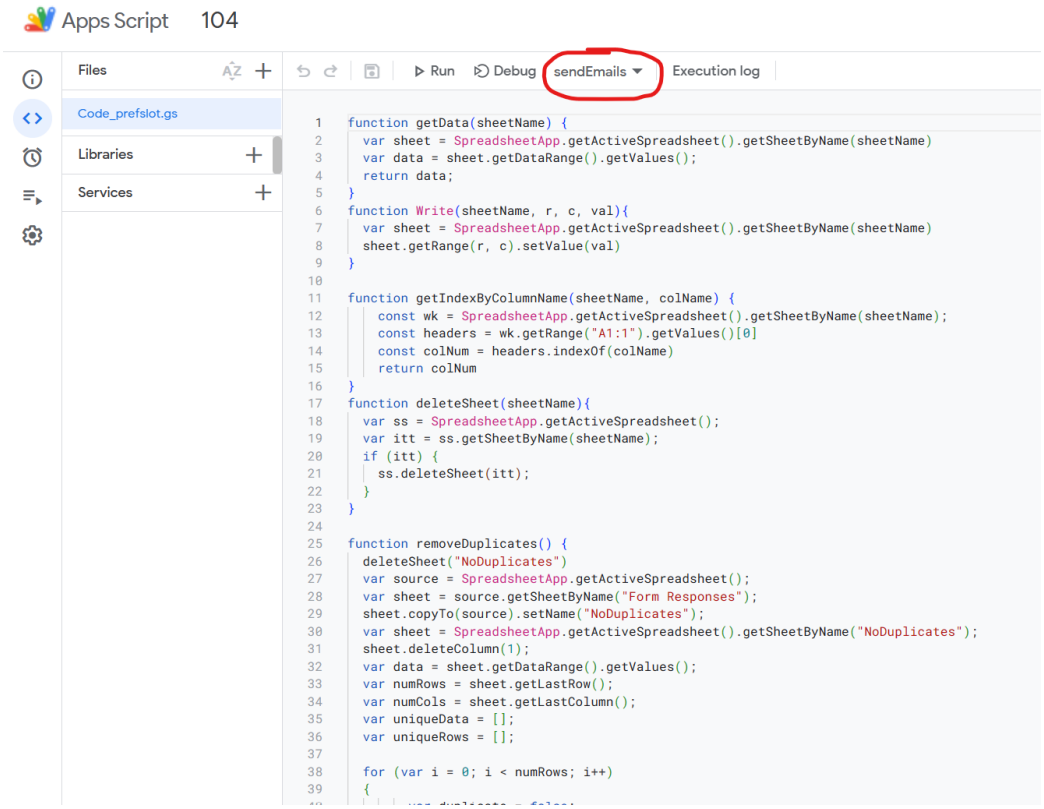
This is the main function of the code that is executed. It first calls removeDuplicates() and TokenAssigner() so now we have a sheet "NoDuplicates" with duplicates handled and Tokens already assigned. The function first takes in the emailSubject and emailBody from the sheet "Templates" in the same spreadsheet. For each row of the data, it takes in the name, email, location and assigned token from the sheet and sends a custom mail with these. It also includes the current date and calculated time slot of the appointment from assigned token number.
The mail is sent using the MailApp.sendEmail() [4]command.
To customize the email, Utilities.formatString() [5] function is used. It Performs sprintf-like string formatting using % style format strings. The emailBody is sent as an htmlBody instead of plain string to include features like Bold, italics and line-breaks like in HTML [6] as I was unable to do bold in plain string format.

3 Set-up and Running the Code

All the relevant links can be found below: They should be accessible using any email ID.

1. Google form - Form maker's link : To edit the information asked from the form fillers.
2. Google form - Responder's link : To add data to the spreadsheet
3. Spreadsheet containing Responses : This spreadsheet contains the Form Responses and the Templates for Email. It also contains the NoDuplicates sheet which is created on execution of the code. It can also be opened by Google Form Makers Link > Settings > View In Sheets
4. Apps Script Code : Can also be accessed using Spreadsheet > Extensions > Apps Script > Choose Project : 104

To execute the code, open the Spreadsheet linked above. Also open the App Script Project "104" using either the link or the steps given above. Make sure to choose the function sendEmails() in the dropdown menu as shown in the red circle and code_prefslot.gs as the file and press RUN.



The script will require Authorization to send emails and access data on running. In the pop-up Authorization Required, click "Review Permissions" > Choose Google Account > Advanced > Go to 104 (unsafe) > Allow. This will authorize the script and it should start running.

In case there is some issue with Google Account authentication and it is not running, login using the following account -
Gmail Account : cs104anilesh@gmail.com
Password : computerscience104

In case this doesn't work either, import into google spreadsheets the sample.xlsx file. Go to Extensions > App Scripts and add the file code_prefslot.gs. Again choose sendEmails() and press RUN.

References

[1] Get column index by column name. https://stackoverflow.com/questions/36346918/get-column-values-by-column-name-not-column-index.

[2] Spreadsheet api documentation. https://developers.google.com/apps-script/reference/spreadsheet/spreadsheet-app.

[3] Check and remove duplicate entries using google script. https://levelup.gitconnected.com/how-to-use-google-script-to-check-and-remove-duplicate-entries-in-google-shee

[4] Send email for every row in a google sheet. https://spreadsheet.dev/send-an-email-for-every-row-in-a-google-sheet.

[5] Documentation for utilities.formatstring(). https://developers.google.com/apps-script/reference/utilities/utilities#formatString(String,Object...).

[6] How to bold text using google apps script. https://stackoverflow.com/questions/9442375/how-to-bold-specific-text-using-google-apps-script.

[7] Google sheets apps scripts basics. https://spreadsheet.dev/learn-coding-google-sheets-apps-script.

[8] Patrick W. Daly Helmut Kopka. *A Guide to LaTeX*. Addison-Wesley, 2004.