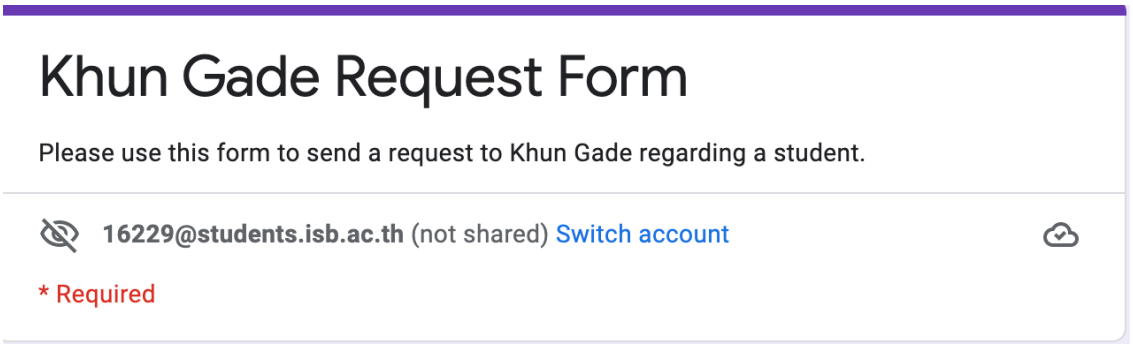
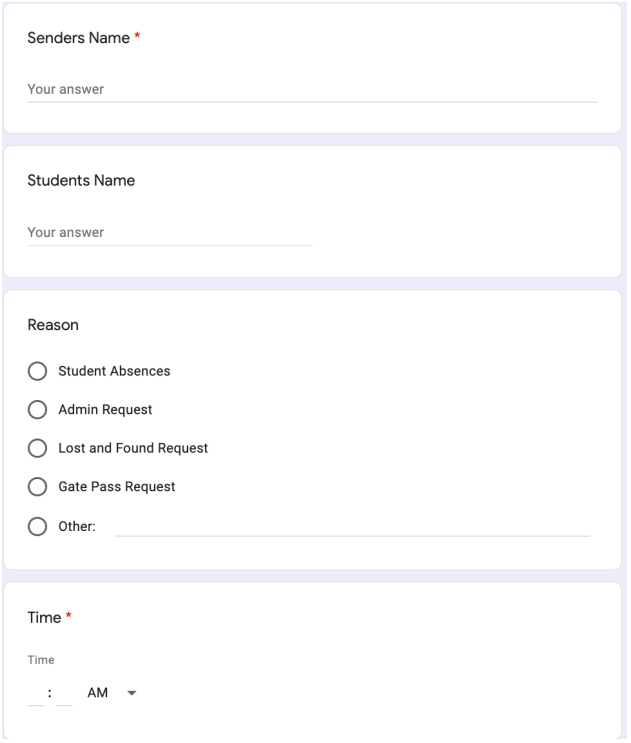
Descriptions of Functions Criterion C

Introduction

Google scripting is an application development program that allows users to create applications which google workspace apps can be integrated into. For example: Google Docs, Google Spreadsheets, Google Forms etc. In the case of this project, google scripting allows an school that has always been using Google as their domain, to automate tasks and lessen possible inefficiencies when completing certain tasks.



A Google Form was used to collect data for the requests as it is easy to gather data as well compatible with sending data to a google spreadsheet. The google form helps complete success criteria 1,3,5,6.The google form only allows certain types of data based on certain parameters. For example Sender's Name must be a worded answer that mitigates any chances for an error when retrieving the data. Moreover, making it so the client does not have to sort it herself.



The function “onFormSubmit” is the most integral part of the program, by running this function it subsequently runs two other functions in “autoSort” and “autoEmail” that will be discussed later. The function gathers the form responses by “form.getResponses” which retrieves the object. Through formResponses.length the length of the object is retrieved. In order to get the latest response, it is the tail end of the array minus one. Shown in line 9. Next we get the item’s of these responses which are just the headings generated from the google form. Then a for loop is run to look at each response that is given. Similar things are done in both line 14 and 15 where data is getting retrieved. Finally in line 20 answers are moved to an array.

```
1   function onFormSubmit(event) {
2
3     record_array = []
4
5     var form = FormApp.openById('17HC4rKA3591q04DiMn2bTZGV8RTWwm-dcx6oaOZpRkI');
6     var formResponses = form.getResponses();
7     var formCount = formResponses.length;
8
9     var formResponse = formResponses[formCount - 1];
10    var itemResponses = formResponse.getItemResponses();
11
12    for (var j = 0; j < itemResponses.length; j++) {
13    var itemResponse = itemResponses[j];
14      var title = itemResponse.getItem().getTitle();
15      var answer = itemResponse.getResponse();
16
17      Logger.log(title);
18      Logger.log(answer);
19
20      record_array.push(answer);
21    }
```

In order to store the data gathered from the google form a function called “AddRecord” must be created. What this function does is, after line 28 when it opens the spreadsheet by the URL, it’ll go to the specific sheet determined by the name. Then the function will append or add the data correspondingly to the name of the header, in the first available row.

```
23    AddRecord(record_array[0], record_array[1], record_array[2], record_array[3]);
24
25  }
26
27  function AddRecord(senders_name, students_name, reason,) {
28    var url = 'https://docs.google.com/spreadsheets/d/1nIHGQE8PiuMpQGzjgT4i8w1mrzhpPsBNHtQRGAGp8zI/edit#gid=0';
29    var ss= SpreadsheetApp.openByUrl(url);
30    var dataSheet = ss.getSheetByName("Khun Gade's Form");
31    dataSheet.appendRow([senders_name, students_name, reason, new Date()]);
```

Next is the autoSort function.which is shown by the code below.

```
function autosort (sort){
  var url = 'https://docs.google.com/spreadsheets/d/1nIHGQE8PiuMpQGzjgT4i8w1mrzhpPsBNHtQRGAGp8zI/edit#gid=0';
  var ss= SpreadsheetApp.openByUrl(url);
  var dataSheet= ss.getSheetByName("Khun Gade's Form")
  const lastRow = dataSheet.getLastRow();
  const lastCol = dataSheet.getLastColumn();
  let range = dataSheet.getRange(2, 3 , lastRow - 1, lastCol);
  range.sort([
    {column: 3, ascending: true},]);

}
```

Through declaring an object instance in the form of getRange (row,column,lastRow-1,lastCol); From range.sort we identify that we would like to sort column: 3 which is column with all the requests by ascending alphabetical order (A-Z). Putting administrative requests first as well as completing success criteria 9.

Word Count: 536

```
50  var emailTemplate = 'https://docs.google.com/document/d/1el2SQ6uPD1yQ9UswOeWk5yfSAGiSePXieoVzoHbepGg/edit?usp=sharing';
51  var emailSubject = 'Request Has been Created';
52
53  /**
54   * Sends a customized email for every response on a form.
55   *
56   * @param {Object} e - Form submit event
57   */
58  function autoEmail(e) {
59    var responses = e.namedValues;
60    var email = "16229@students.isb.ac.th";
61    Logger.log('; responses=' + JSON.stringify(responses));
62
63    MailApp.sendEmail({
64      to: email,
65      subject: emailSubject,
66      htmlBody: createEmailBody(),
67    });
68    Logger.log('email sent to: ' + email);
```

Finally the last function “autoEmail” which is a function that once the form is submitted it automatically sends an email to the recipient. Done together with the onformsubmit trigger. First two variables are declared. The first one being a google document that contains the content of an email along with the second variable containing the email subject. Next if we look at the function “autoEmail” where it’s parameter is e which is an object. The amount of values of responses given are the same amount of emails that will be sent out shown in line 58. Next in Line 61 JSON.stringify turns a javascript object into a JSON String. Then a call to a separate function is made which is MailApp.sendEmail through using Gmail.