Brief Implementation for Case Study: Streamlining Operation with Google Script

Calendar Event Handling
Upon creation of a Google Calendar event (using timer triggers for production):

- First initializing the participants Participant A (the active user in my case) and Participant B (the first index in event list).
- We then create a Google Drive folder (using the given naming convention).
- Then saving the handoff details such as the folder ID and emails in Script Properties recognized by the event ID.
- Finally generating a pre-populated Google Form link containing the event ID and email it to Participant B. In this case I used a sample google form with event ID field for testing purposes.

Key Functions:

- **handleCalendarEvent(e)**
  1. Reads the Calendar event data from the passed event object or a dummy event for testing in my case.
  2. Creates and shares the Drive folder and stores data using the key format "handoff_" + eventID.
  3. Constructs the pre-populated form link using FORM_URL and EVENT_ID_KEY.
  4. Finally Emails the link to Participant B.

- **testHandleCalendarEvent()**
  – A dummy function for testing the above process without a real calendar trigger.

- **Implementation Steps:**

- Set **FORM_URL** and **EVENT_ID_KEY** according to your form details.
- Run **testHandleCalendarEvent()** to test the CalendarEvent and this should create a folder in Drive and email the pre-populated link to your designated Participant B email.

Processing Form Submissions to Generate a PDF

- **Purpose:**
  When Participant B submits the form with prepopulated event ID the script will:
  - Retrieve all responses from the form submission and look for Event ID response to match.
  - Create a PDF from the form responses and save it in the folder.
  - Send notification emails to both Participant A and Participant B.
- **Key Functions:**

  - **onFormSubmit(e)**
    1. Automatically triggered when the form is submitted. To make sure this happens the script should be bound to the form.
    2. Loops through each Item Response to build a text summary and find the Event ID.
    3. Retrieves handoff data from Script Properties using the Event ID as the key and calls createPdfFromText to create a PDF file containing the form data.
    4. Saves the PDF in the correct folder and sends emails.
  - **createPdfFromText(text, title)**
    1. Creates a temporary Google Doc with the provided text.
    2. Converts the document to a PDF blob.

- **Implementation Steps:**

- Bind this Apps Script project to your Google Form.
- Set up an **on form submit trigger** and choose "From form" as the event source and "On form submit" as the event type.
- Ensure your form contains a question for "Event ID" that will be automatically pre-populated by the link. We can also hide the question for production purposes.
- Test and verify the process by submitting the form.
- Check your emails and Folder for the verification.

3. Debugging and Verification

- **Logging:**
  This code maximizes logging techniques to smoothen the execution process.

- **Script Properties:**
  – In the Apps Script editor, check **File > Project properties > Script properties** to ensure that the handoff data is stored under the correct key and also verify the oauthscopes in **appsscript.json** (manifest file) to check for permission scopes.

- **Folder and File Check:**
  – Copy the folder ID from the logs and paste it into your browser URL to make sure the PDF I saved correctly.

- **Email Verification:**
  – Ensure that both Participant A and Participant B receive the notification emails.

This setup demonstrates the asked handoff process using Google Calendar, Google Drive and Google Forms via Google Apps Script.

Commented Code:

Code.gs:

```
// Form URL and Event Id as key for verification
var FORM_URL = "https://docs.google.com/forms/d/e/1FAIpQLSfl4lcSj6pW4Drmsq-dYUuQDAoyRSZww1JGcWCvUQ2tCoPheQ/viewform";
var EVENT_ID_KEY = "entry.839337160";

function handleCalendarEvent(e) {
  try {
    // Error handling for missing Calendar Event
    if (!e || !e.calendarEvent) {
      throw new Error("Missing 'calendarEvent' or not triggered properly.");
    }
    var event = e.calendarEvent;
    var eventId = event.getId();

    // I am going to choose participant A as the active user and participant B as my secondary email for testing purposes. The participant B will be chosen from a guest list just in case we need more emails.
    var participantAEmail = Session.getActiveUser().getEmail();
    var participantAName = "participantA";

    var guestList = event.getGuestList();
    if (guestList.length === 0) {
      throw new Error("No guest found in the event.");
    }

    // To make it simple I just chose the first index of my guest list as my participantB
    var participantB = guestList[0];
    var participantBEmail = participantB.getEmail();
    var participantBName = "participantB";

    // Now creating the folder with the given naming convention and I will use the driveapp library to create the folder.
    var folderName = participantAName + " " + participantBName + " Handoff";
    var folder = DriveApp.createFolder(folderName);

    // Sharing the folder with participant A and saving the handoff details.
    folder.addEditor(participantAEmail);

    var handoffData = {
      folderId: folder.getId(),
      participantAEmail: participantAEmail,
      participantBEmail: participantBEmail
    };
    PropertiesService.getScriptProperties().setProperty("handoff_" +
eventId, JSON.stringify(handoffData));
    // I am logging just to see if it executed
    Logger.log("Folder created for the event: " + eventId);

    // Creating a pre-populated form link to match the event id and
emailing to the participant B.
    var prepopulatedLink = FORM_URL + "?" + EVENT_ID_KEY + "=" +
encodeURIComponent(eventId);
    var emailSubject = "Please Complete the form given below:";
    var emailMessage = "Dear " + participantBName + ",\n\n" +
      "Please complete the form by clicking the link below:\n" +
      prepopulatedLink + "\n\n" +
      "Regards,\n\n" +
      participantAName;
    MailApp.sendEmail(participantBEmail, emailSubject, emailMessage);
    // Logging and error checking
    Logger.log("Pre-populated form link emailed to Participant B: " +
participantBEmail);

  } catch (error) {
    Logger.log("Error in handleCalendarEvent: " + error.message);
  }
}

// I have created a Dummy function below for testing the above function.
For production purposes we have a simulate a proper calendar event using
the above function using a timer trigger.
function testHandleCalendarEvent() {
  var dummyEvent = {
    calendarEvent: {
      getId: function () { return EVENT_ID_KEY; },
      getGuestList: function () {
        return [{
          getEmail: function () { return "My secondary email"; }, // I used
my secondary email for testing (gmail.com)
          getName: function () { return "ParticipantB"; }
        }];
      }
    }
  };
  handleCalendarEvent(dummyEvent);
}

// Onsubmit function to get details from form. Just to let you that this
apps script project is binded to my test form.
function onFormSubmit(e) {
  try {
    //getting the item responses
    var formResponse = e.response;
    var itemResponses = formResponse.getItemResponses();
    var responseText = "";
    var eventId = null;

    // Getting the text response for each question
    itemResponses.forEach(function (itemResponse) {
      var question = itemResponse.getItem().getTitle();
      var answer = itemResponse.getResponse();
      responseText += question + ": " + answer + "\n";

      // Here we are looking for the event id to match with the event id
given in the testHandleCalendarEvent
      if (question.toLowerCase().indexOf("event id") !== -1) {
        eventId = answer;
        Logger.log("Found event: " + eventId);
      }
    });

    if (!eventId) {
      throw new Error("Event ID not in the form please recheck");
    }

    // We get and retrieve the handoff details using event id and logging.
    var handoffDataString =
PropertiesService.getScriptProperties().getProperty("handoff_" + eventId);
    if (!handoffDataString) {
      throw new Error("No data found for : " + eventId);
    }
    var handoffData = JSON.parse(handoffDataString);
    Logger.log("Handoff data found: " + handoffDataString);

    // Create a PDF and saving it in the folder using a helper function.
    var pdfTitle = "Handoff Document between Participant A and Participant
B";
    var pdfBlob = createPdfFromText(responseText, pdfTitle);
    Logger.log("PDF created: " + pdfTitle);
    var folder = DriveApp.getFolderById(handoffData.folderId);
    folder.createFile(pdfBlob);

    // Emailing both participants.
    var subject = "Completion of the Handoff Procedure";
    var message = "Hello participants ,\n\nThe handoff process has been
completed. The folder is saved in Participant A's Drive
Folder.\n\nRegards.";
    MailApp.sendEmail(handoffData.participantAEmail, subject, message);
    MailApp.sendEmail(handoffData.participantBEmail, subject, message);
    Logger.log("Email sent.");

  } catch (error) {
    Logger.log("Error in onFormSubmit: " + error.message);
  }
}
```

```javascript
//Helper function to create pdf
function createPdfFromText(text, title) {
  // Creting a google doc temporarily
  var doc = DocumentApp.create(title);
  var body = doc.getBody();
  body.appendParagraph(text);
  doc.saveAndClose();

  // Converting the document to a PDF blob file.
  var docFile = DriveApp.getFileById(doc.getId());
  var pdfBlob = docFile.getAs("application/pdf").setName(title + ".pdf");

  // Deleting the document later.
  DriveApp.getFileById(doc.getId()).setTrashed(true);
  return pdfBlob;
}
```

appsscript.json:

```json
{
  "timeZone": "America/New_York",
  "dependencies": {},
  "exceptionLogging": "STACKDRIVER",
  "oauthScopes": [
    "https://www.googleapis.com/auth/forms.currentonly",
    "https://www.googleapis.com/auth/drive",
    "https://www.googleapis.com/auth/script.external_request",
    "https://www.googleapis.com/auth/spreadsheets",
    "https://www.googleapis.com/auth/script.send_mail",
    "https://www.googleapis.com/auth/documents"
  ]
}
```