Automating Data Collection from KoboToolbox to Google Sheets and saving files to Google Drive

Introduction

In many data collection scenarios, especially in field research, automating the flow of data from collection tools like KoboToolbox directly to Google Sheets and Google Drive can streamline processes and save significant time. This automation is particularly useful when working with multiple data entries that may include attachments (like PDFs or images) that need to be stored securely and organized for further analysis.

By using Google Apps Script, we can:

- Pull form responses from KoboToolbox,
- Save structured data into Google Sheets for easy access and processing,
- Automatically download any attachments and store them in a designated Google Drive folder.

The following script demonstrates how to accomplish this. After executing the script, each form submission will have a corresponding row in Google Sheets and, if attachments exist, they will be stored in Google Drive, with a link to each file recorded in the sheet.

Code

```
function getData() {
  var formId = "xxxxxxxxxxxxxxxx"; // Replace with your form ID
  var apiUrl = "https://eu.kobotoolbox.org/api/v2/assets/" + formId + "
      /data/?format=json";
  var apiToken = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"; // Replace with your
       API token
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var driveFolderId = "xxxxxxxxxxxxxxxxxxxxxx"; // Replace with your
      Google Drive folder ID
  var headers = {
    "Authorization": "Token␣" + apiToken
  };
  var response = UrlFetchApp.fetch(apiUrl, { headers: headers });
  var data = JSON.parse(response.getContentText());

  // Clear existing sheet content to avoid duplicates
  sheet.clear();

  // Get all unique keys (fields) directly for headers
  var firstRecord = data.results[0];
  var keys = Object.keys(firstRecord);
  sheet.appendRow([...keys, "Drive␣File␣ID"]); // Append headers
      directly and add a column for Drive File ID

  var folder = DriveApp.getFolderById(driveFolderId);

  for (var i = 0; i < data.results.length; i++) {
    var rowData = [];
    var record = data.results[i];

    // Collect data for each key, matching header names
    keys.forEach(key => {
      rowData.push(record[key] || ""); // Add each field's data or an
          empty string if null
    });

    // File handling based on specific variable for filename
    if (record._attachments && record._attachments.length > 0) {
      var fileName = record["Media"]; // Use the specific variable name
      var pdfUrl = "https://kc-eu.kobotoolbox.org/media/original?
          media_file=" + encodeURIComponent(record._attachments[0].
          filename);

      // Check if the file with the same name already exists
      var fileExists = false;
      var files = folder.getFilesByName(fileName);
      if (files.hasNext()) {
        fileExists = true;
        var existingFile = files.next();
        rowData.push(existingFile.getId()); // Reference existing file
            ID in sheet
      }

      // Download and save new file if it doesn't already exist
      if (!fileExists) {
        var pdfResponse = UrlFetchApp.fetch(pdfUrl, { headers: headers
            });
        var blob = pdfResponse.getBlob().setName(fileName);
        var newFile = folder.createFile(blob);
        rowData.push(newFile.getId()); // Save new file ID in sheet
      }
    } else {
      rowData.push("No␣attachment"); // Indicate no file if there is no
           attachment
    }

    // Append the row to the sheet
    sheet.appendRow(rowData);
  }
}
```

Listing 1: Full Script for Fetching and Saving KoboToolbox Data to Google Sheets and files to Google Drive

Explanation of Each Part of the Script

Define Variables

- **Form ID (formId)**: Identifies the specific form in KoboToolbox you want to get data from.
- **API URL (apiUrl)**: Combines the base URL of KoboToolbox's API with the form ID to specify where to retrieve data.
- **API Token (apiToken)**: Serves as a unique key to allow access to your KoboToolbox account securely.
- **Sheet and Folder**:
  - **sheet** refers to the Google Sheet where data will be saved.
  - **driveFolderId** identifies the specific Google Drive folder where attachments will be stored.

Authentication Headers

This code sets up a header with the API token. Think of this as a way to "log in" programmatically so KoboToolbox recognizes your account and gives you access to the data.

Fetching Data

- `UrlFetchApp.fetch(apiUrl, { headers: headers })` makes a request to KoboToolbox, which responds with the form data.
- This response is then parsed (`JSON.parse(response.getContentText())`) into a format the script can work with.

Clear the Existing Sheet Content

`sheet.clear()` erases all existing content from the Google Sheet to prevent duplicate data.

Extract and Set Headers

- It reads the first record's keys (or field names) to automatically generate column headers.
- `sheet.appendRow([...keys, "Drive File ID"])` creates a row at the top of the sheet with each field's name and an extra column for storing Google Drive file IDs.

Saving Each Record to a Row in Google Sheets

For each record (or row of data) in the form results, the code:

- Creates an empty array called `rowData`.
- Goes through each field name (`keys`) and checks if that field has a value for the current record.
- If there's no data for a field, it fills in an empty space (`""`).
- Adds each field's value to `rowData`.

File Handling and Attachment Downloads

- If the record includes an attachment (in `_attachments`), the code:
  - Uses a specific field (`Media` in this case) to set the attachment's file name in Google Drive.
  - Checks if a file with that name already exists in the folder (to avoid duplicates).
- If there's no existing file, it:
  - Fetches the attachment and saves it in the specified Google Drive folder.
  - Adds the Google Drive file ID to `rowData` to track it.

Add Row to Google Sheets

Finally, it appends `rowData` as a new row in the Google Sheet.

Each time the script runs, it fetches all data from KoboToolbox, clears and refreshes the Google Sheet with new entries, and saves any new attachments in Google Drive while avoiding duplicates.