Automated Passport Image Downloader using Google Apps Script - by Odubiyi Azeez

# 1. Introduction

This project demonstrates an automation solution built using Google Apps Script to download image files from Google Drive links listed in a Google Sheet. The goal is to streamline the process of retrieving passport images for unique user IDs, eliminating manual downloads and preventing duplicate storage.

2. Problem Statement

The making of the Oyo Kopa magazine, one of which is the preparation of corps members' profile which involves their passport sized photograph. This solution address the repeated task of manually downloading passports from Google drive.

# 2. Objective

To automate the downloading of passport images from a Google Sheet that contains unique IDs and their corresponding Google Drive shareable links, saving them into a designated folder in Google Drive while handling retries and skipping duplicates.

# 3. Tools and Technologies Used

- Google Sheets

- Google Apps Script

- Google Drive API

- JavaScript (Apps Script flavor)

# 4. Key Features

- Reads data from Google Sheets (unique IDs and links).

- Downloads images to a specific folder on Google Drive.

- Skips already downloaded files to avoid duplicates.

- Retries failed downloads up to 3 times.

- Uses both direct download links and file ID-based methods.

# 5. Methodology

The script reads rows from the spreadsheet starting from row 2. It processes images in batches (10 per execution) to avoid hitting Google Apps Script’s execution time limit. If a file fails to download, the script retries up to 3 times and optionally falls back to accessing the file by ID using the DriveApp service. Execution continues in scheduled chunks via time-based triggers.

# 6. Script Flow

1. `startDownloadPassports`: Initializes row tracking and triggers the first batch.

2. `downloadPassports`: Downloads a batch of images and schedules the next batch if needed.

3. `downloadWithRetries`: Attempts to fetch and save the image up to 3 times.

4. `transformToDownloadUrl`: Converts Google Drive links to direct download links.

5. `fileExists`: Checks if a file already exists to avoid duplicates.

6. `tryGdownAlternative`: Accesses image directly via file ID as fallback.

# 7. Limitations

- Google Apps Script execution time is limited to 6 minutes (for consumer accounts).

- Files must have public access or the script must have permission.

- Google Drive links must be properly formatted.

# 8. Conclusion

This solution effectively automates the tedious task of downloading and managing image files from Google Sheets. It handles large datasets by scheduling executions in manageable batches, ensuring efficient and error-resistant operation.

Report generated on: 2025-06-23 21:19:29

# 9. Script Appendix and Explanation

## 9.1 startDownloadPassports

function startDownloadPassports() {  
 var scriptProperties = PropertiesService.getScriptProperties();  
 scriptProperties.setProperty('LAST\_PROCESSED\_ROW', '1');  
 downloadPassports();  
}

Initializes the download process by setting the starting row for processing and calling the main function `downloadPassports()`. This is usually run manually to begin the automation cycle.

## 9.2 downloadPassports

function downloadPassports() {  
 var scriptProperties = PropertiesService.getScriptProperties();  
 var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
 var lastRow = sheet.getLastRow();  
 var dataRange = sheet.getRange(2, 1, lastRow - 1, 2);  
 var data = dataRange.getValues();  
 var destinationFolder = DriveApp.getFoldersByName('ibnb2').next();  
 var lastProcessedRow = parseInt(scriptProperties.getProperty('LAST\_PROCESSED\_ROW'), 10);  
 var batchSize = 10;  
  
 for (var i = lastProcessedRow; i < lastProcessedRow + batchSize && i < data.length; i++) {  
 var uniqueId = data[i][0];  
 var passportLink = data[i][1];  
 if (!fileExists(destinationFolder, uniqueId)) {  
 downloadWithRetries(uniqueId, passportLink, destinationFolder);  
 } else {  
 Logger.log('File for ' + uniqueId + ' already exists. Skipping download.');  
 }  
 }  
  
 if (i < data.length) {  
 scriptProperties.setProperty('LAST\_PROCESSED\_ROW', i.toString());  
 ScriptApp.newTrigger('downloadPassports').timeBased().inMinutes(1).create();  
 } else {  
 scriptProperties.deleteProperty('LAST\_PROCESSED\_ROW');  
 Logger.log('Download completed for all rows.');  
 }  
}

Handles the main download logic in batches. Checks if the file already exists to avoid duplicates, downloads new files, and sets up a trigger to continue in the next cycle if more rows exist.

## 9.3 downloadWithRetries

function downloadWithRetries(uniqueId, passportLink, destinationFolder) {  
 var retries = 3;  
 var success = false;  
  
 for (var attempt = 0; attempt < retries; attempt++) {  
 try {  
 var response = UrlFetchApp.fetch(transformToDownloadUrl(passportLink), {muteHttpExceptions: true});  
 if (response.getResponseCode() == 200 && isImage(response.getBlob())) {  
 var fileBlob = response.getBlob();  
 var file = destinationFolder.createFile(fileBlob.setName(uniqueId + '.jpg'));  
 Logger.log('Successfully downloaded ' + file.getName());  
 success = true;  
 break;  
 }  
 } catch (e) {  
 Logger.log('Error downloading ' + uniqueId + ' on attempt ' + (attempt + 1) + ': ' + e.message);  
 }  
 Utilities.sleep(2000);  
 }  
  
 if (!success) {  
 Logger.log('All retry attempts failed for ' + uniqueId + '. Trying gdown alternative.');  
 tryGdownAlternative(uniqueId, passportLink, destinationFolder);  
 }  
}

Attempts to download each image up to three times. If all retries fail, it switches to using the Google Drive file ID directly via the `tryGdownAlternative()` function.

## 9.4 transformToDownloadUrl & extractFileIdFromUrl

function transformToDownloadUrl(url) {  
 var fileIdMatch = url.match(/[-\w]{25,}/);  
 if (fileIdMatch) {  
 var fileId = fileIdMatch[0];  
 return 'https://drive.google.com/uc?export=download&id=' + fileId;  
 }  
 return url;  
}  
  
function extractFileIdFromUrl(url) {  
 var fileIdMatch = url.match(/[-\w]{25,}/);  
 return fileIdMatch ? fileIdMatch[0] : null;  
}

`transformToDownloadUrl` converts a shared Google Drive link into a direct download link.  
`extractFileIdFromUrl` is used to extract the file ID for use with the Google Drive API fallback.

## 9.5 isImage & fileExists

function isImage(blob) {  
 var contentType = blob.getContentType();  
 return contentType.startsWith('image/');  
}  
  
function fileExists(folder, uniqueId) {  
 var files = folder.getFilesByName(uniqueId + '.jpg');  
 return files.hasNext();  
}

`isImage` checks if the fetched file is a valid image using its MIME type.  
`fileExists` checks if the file with the given unique ID already exists in the destination folder.

## 9.6 tryGdownAlternative

function tryGdownAlternative(uniqueId, passportLink, destinationFolder) {  
 try {  
 var fileId = extractFileIdFromUrl(passportLink);  
 if (fileId) {  
 var file = DriveApp.getFileById(fileId);  
 var blob = file.getBlob();  
 destinationFolder.createFile(blob.setName(uniqueId + '.jpg'));  
 Logger.log('Successfully downloaded ' + uniqueId + '.jpg using gdown alternative.');  
 }  
 } catch (e) {  
 Logger.log('Error using gdown alternative for ' + uniqueId + ': ' + e.message);  
 }  
}

Fallback function to download the image using DriveApp API if direct URL fetching fails. It accesses the file using its Google Drive file ID and saves it with the appropriate filename.