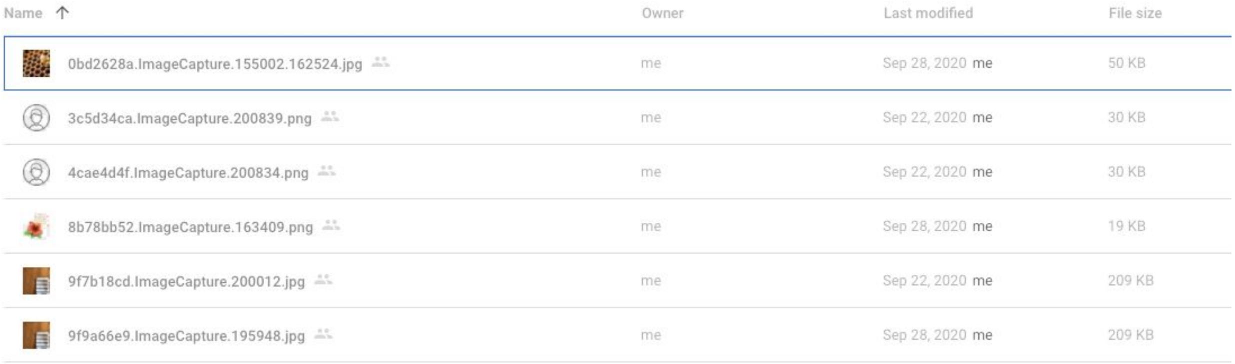
Find Duplicate Images using AppsScript

NOTE: This is not an official Google Product or solution

Intro

Appsheet allows you to upload the same image (or file) over and over again, and the system will mark each image with a unique filename, e.g. something like the following:



Name ↑	Owner	Last modified	File size
0bd2628a.ImageCapture.155002.162524.jpg	me	Sep 28, 2020 me	50 KB
3c5d34ca.ImageCapture.200839.png	me	Sep 22, 2020 me	30 KB
4cae4d4f.ImageCapture.200834.png	me	Sep 22, 2020 me	30 KB
8b78bb52.ImageCapture.163409.png	me	Sep 28, 2020 me	19 KB
9f7b18cd.ImageCapture.200012.jpg	me	Sep 22, 2020 me	209 KB
9f9a66e9.ImageCapture.195948.jpg	me	Sep 28, 2020 me	209 KB

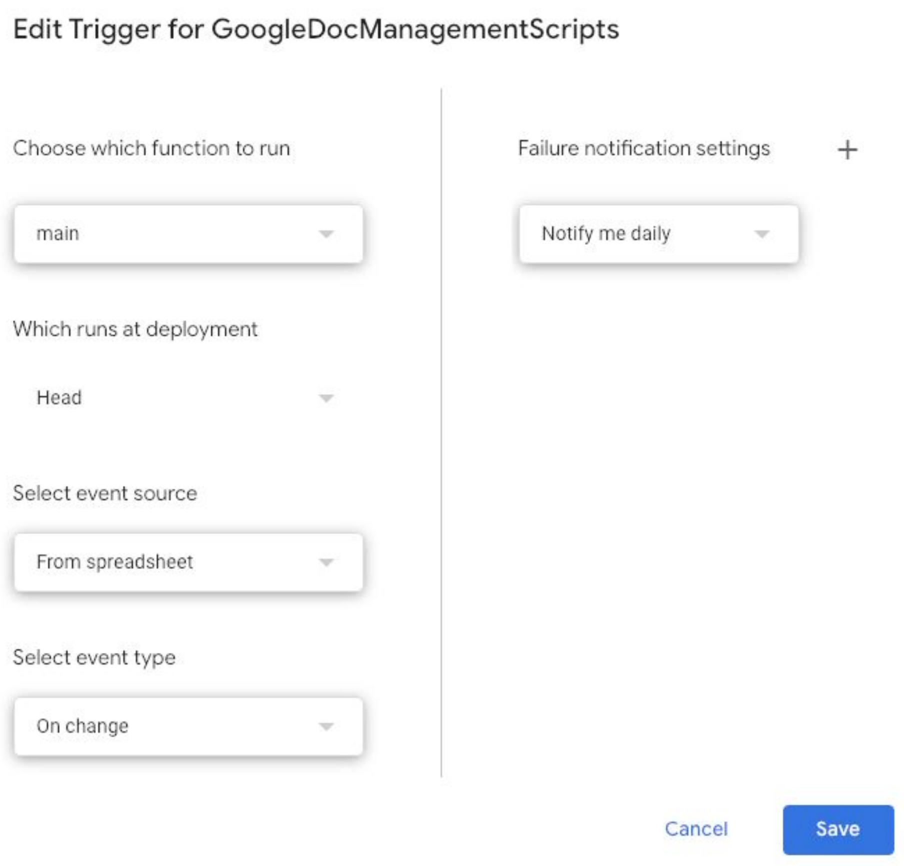
This is all by design and expected behavior. Customers who desire to identify exactly identical images can use this document and its reference implementation to build a solution that marks images as duplicates for future deletion or archiving purposes.

Using the Reference Design

This reference design requires some familiarity with Google AppsScript. A summary of the steps required to explore and study this reference design:

- Copy the sample Appsheet app located here into your Appsheet account.

- Immediately open Google Drive and locate the folder where this app and its content was deployed to. Find the Google Sheet called “Google Doc”, open it, then go to the Tools menu and choose “Script Editor”.
- Copy the AppsScript script located in this gist into the script editor.
- You will need to make one single change: the FolderID of the location where these apps’ images are uploaded to.
- Set up a trigger for this script: it should run on all changes to the Google Sheet, e.g:



Edit Trigger for GoogleDocManagementScripts

Choose which function to run

main

Which runs at deployment

Head

Select event source

From spreadsheet

Select event type

On change

Failure notification settings

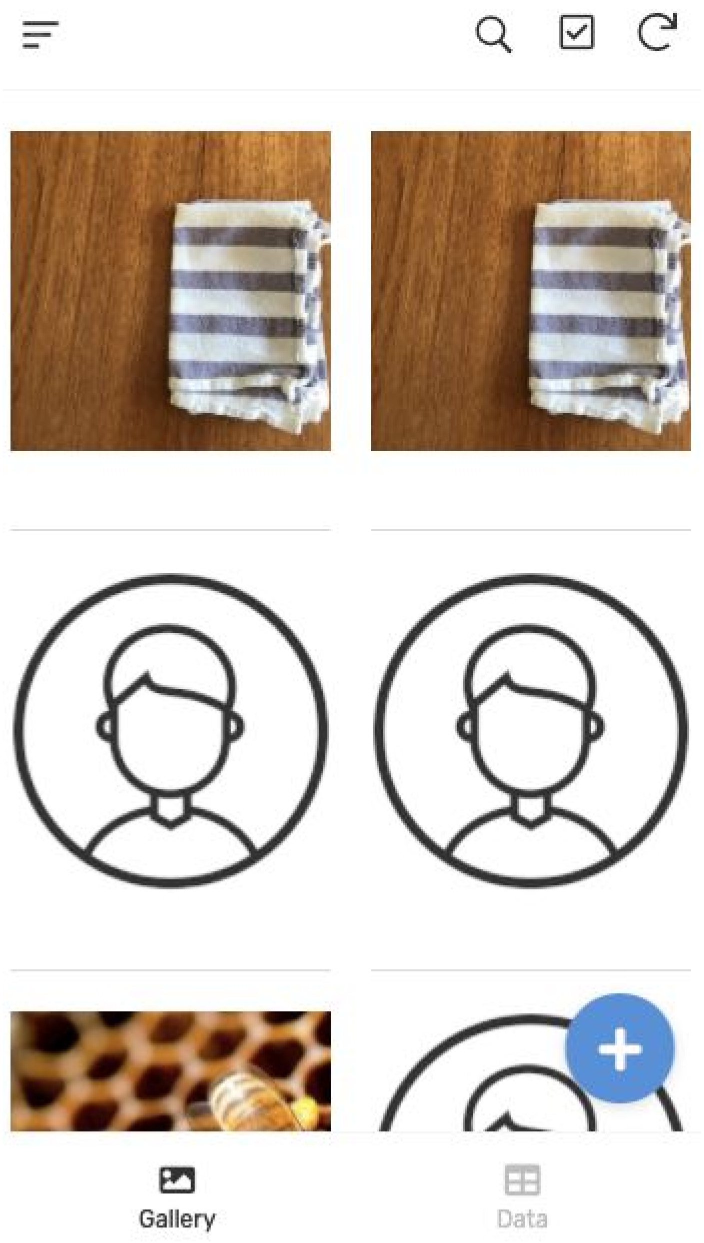
Notify me daily

- In the above image, we called our script “GoogleDocManagementScripts”. Yours can be named whatever you like.
- Save your work and test using the Appsheet app.

Expected Behavior and How It Works

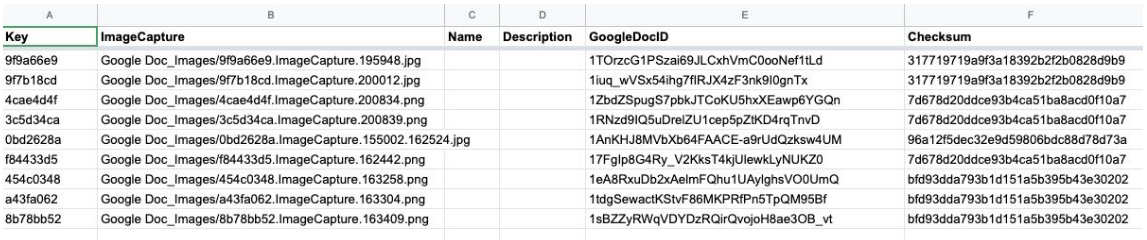
Part One - the creation of the checksum/digest value

Open the Appsheet app. Notice that you create a new record and upload an image (or take a photo on your smartphone):



Go ahead and do this twice using the same exact image. If you configured your AppsScript script correctly, it will run each time you add a record to the sheet, and then it will:

- Insert the Google Drive File ID back into your Google Sheet
- Run the AppsScript utility function called computeDigest on the image and return a checksum/string back into your Google Sheet, e.g.:

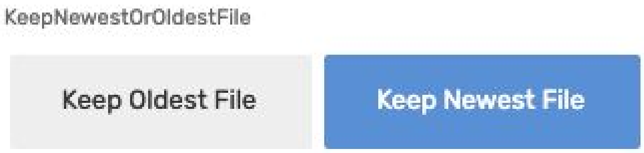


A	B	C	D	E	F
Key	ImageCapture	Name	Description	GoogleDocID	Checksum
9f9a66e9	Google Doc_Images/9f9a66e9.ImageCapture.195948.jpg			1TOrzcG1PSzai69JLCxhVmC0ooNef1tLd	317719719a9f3a18392b2f2b0828d9b9
9f7b18cd	Google Doc_Images/9f7b18cd.ImageCapture.200012.jpg			1iuq_wVSx54ihg7flRJX4zF3nk9l0gnTx	317719719a9f3a18392b2f2b0828d9b9
4cae4d4f	Google Doc_Images/4cae4d4f.ImageCapture.200834.png			1ZbdZSpugS7pbkJTCoKU5hxXEawp6YGQn	7d678d20ddce93b4ca51ba8acd0f10a7
3c5d34ca	Google Doc_Images/3c5d34ca.ImageCapture.200839.png			1RNzd9IQ5uDrelZU1cep5pZtKD4rqTnvD	7d678d20ddce93b4ca51ba8acd0f10a7
0bd2628a	Google Doc_Images/0bd2628a.ImageCapture.155002.162524.jpg			1AnKHJ8MVbXb64FAACE-a9rUdQzksw4UM	96a12f5dec32e9d59806bdc88d78d73a
f84433d5	Google Doc_Images/f84433d5.ImageCapture.162442.png			17Fglp8G4Ry_V2KksT4kjUlewkLyNUKZ0	7d678d20ddce93b4ca51ba8acd0f10a7
454c0348	Google Doc_Images/454c0348.ImageCapture.163258.png			1eA8RxuDb2xAelmFQhu1UAylghsVO0UmQ	bfd93dda793b1d151a5b395b43e30202
a43fa062	Google Doc_Images/a43fa062.ImageCapture.163304.png			1tdgSewactKStvF86MKPRfPn5TpQM95Bf	bfd93dda793b1d151a5b395b43e30202
8b78bb52	Google Doc_Images/8b78bb52.ImageCapture.163409.png			1sBZZyRWqVDYDzRQirQvojoH8ae3OB_vt	bfd93dda793b1d151a5b395b43e30202

- Interesting side note: the reason we have to call "computeDigest" is because although the Google Drive API has a built in checksum method, Google AppsScript does not expose this method. Instead in our code we have to:
    - Get the image from Google Drive
    - Represent it as a byte array
    - Run computeDigest on this byte array
    - Take the result of the previous step and convert any negative values to positive values
    - Return this final string - this is our "checksum" or file "digest".

Part Two - An Appsheet report which identifies duplicate images and marks them as such

- Now, back in the app, take note of an Appsheet Report called "Check Data for Dupes". This is meant to run once a day or weekly for **each row in the table**.
- Also note in the app, on the upper left menu, a user defined choice: should we keep the newest file as the non-dupe, or, should we keep the oldest file as the non-dupe?



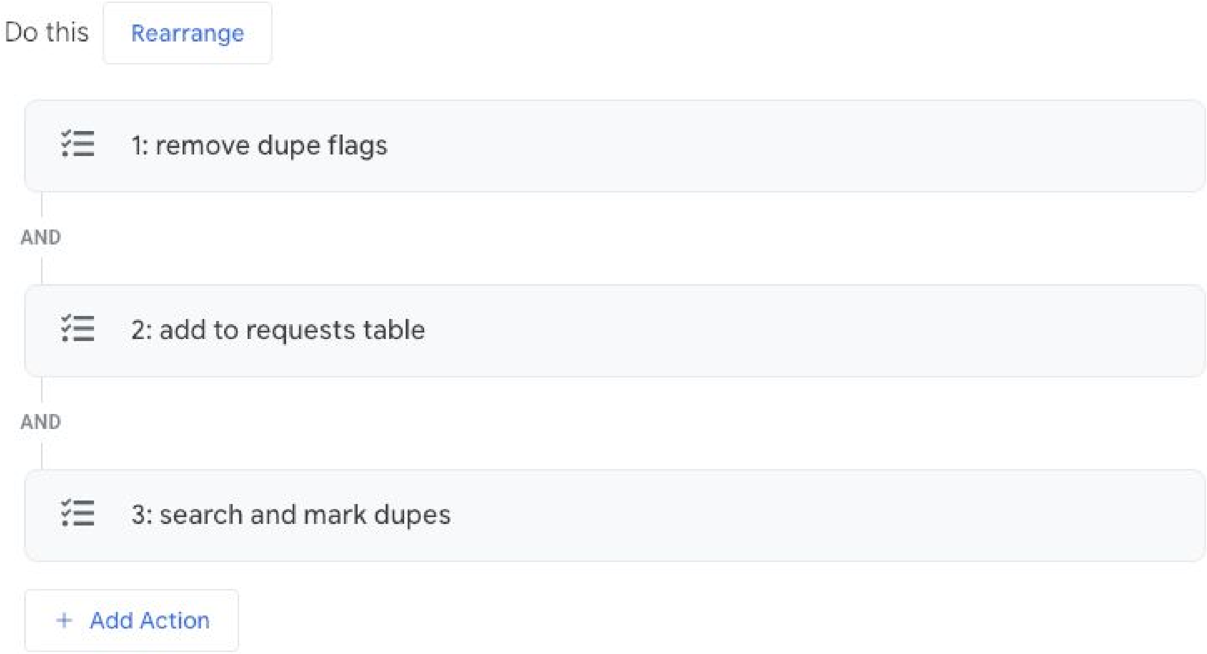
KeepNewestOrOldestFile

Keep Oldest File [✓ Keep Newest File]

- You can manually run the report at any time using the UX and designer. You should upload some deliberate duplicate images to test.
- Some very clever query logic is in the condition field for this report:

```
AND
(
[PossibleDupe] <> "DUPE",
count(select(Google Doc[Key],[Checksum] = [_THISROW].[Checksum])) > 1,
if(any(globals[KeepNewestOrOldestFile]) = "Keep Oldest File",
[Key] <> MINROW("Google Doc","CreationDate",[Checksum] = [_THISROW].[Checksum]),
[Key] <> MAXROW("Google Doc","CreationDate",[Checksum] = [_THISROW].[Checksum])
)
)
```

- The above is an expensive query - on thousands and thousands of records it might take a little while. This is why we have built it as an Appsheet Report as opposed to marking a field or table with this same logic.
- When you run the report, it will find the duplicate images and mark the PossibleDupe column with “DUPE” by calling a series of Appsheet actions,
- E.g. here are the actions we call:

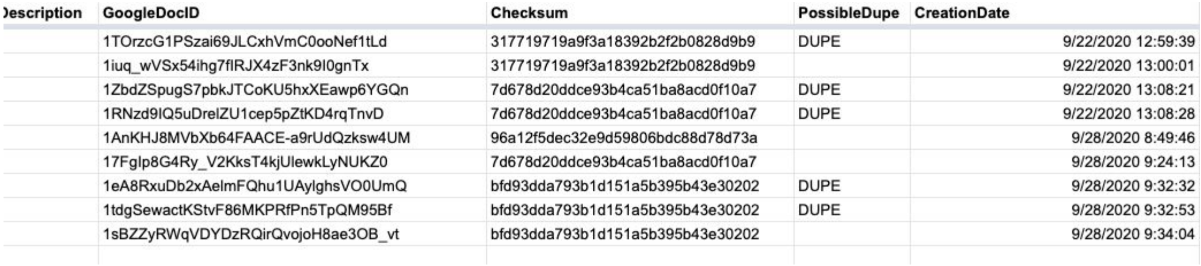


1: remove dupe flags

2: add to requests table

3: search and mark dupes

- And here is the Google Sheet output after the report has run:



escription	GoogleDocID	Checksum	PossibleDupe	CreationDate
	1TOrzcG1PSzai69JLCxhVmC0ooNef1tLd	317719719a9f3a18392b2f2b0828d9b9	DUPE	9/22/2020 12:59:39
	1iuq_wVSx54ihg7flRJX4zF3nk9I0gnTx	317719719a9f3a18392b2f2b0828d9b9		9/22/2020 13:00:01
	1ZbdZSpugS7pbkJTCoKU5hxXEawp6YGQn	7d678d20ddce93b4ca51ba8acd0f10a7	DUPE	9/22/2020 13:08:21
	1RNzd9IQ5uDrelZU1cep5pZtKD4rqTnvD	7d678d20ddce93b4ca51ba8acd0f10a7	DUPE	9/22/2020 13:08:28
	1AnKHJ8MVbXb64FAACE-a9rUdQzksw4UM	96a12f5dec32e9d59806bdc88d78d73a		9/28/2020 8:49:46
	17Fglp8G4Ry_V2KksT4kjUlewkLyNUKZ0	7d678d20ddce93b4ca51ba8acd0f10a7		9/28/2020 9:24:13
	1eA8RxuDb2xAelmFQhu1UAylghsVO0UmQ	bfd93dda793b1d151a5b395b43e30202	DUPE	9/28/2020 9:32:32
	1tdgSewactKStvF86MKPRfPn5TpQM95Bf	bfd93dda793b1d151a5b395b43e30202	DUPE	9/28/2020 9:32:53
	1sBZZyRWqVDYDzRQirQvojoH8ae3OB_vt	bfd93dda793b1d151a5b395b43e30202		9/28/2020 9:34:04

If you are paying close attention, you will note that we are also copying these DUPE records to a separate Google Sheet called “Deletion Requests”. This is so that you, the designer, can now take further action on these files.

From here, you can start to “do something” such as remove these records, or iterate through the image content, removing the images as desired. Quite deliberately, we have ***not*** included any deletion activities or destructive examples in this reference design, as we are not in the business of making it easy to delete our customers’ content.