# Introduction

## Problem/goals

* Data sovereignty & access
* Storage & backup
* Tagging system?
* Multilingual search
* Offline use

## Stakeholders

* Researchers – illustrator, animator, linguists, …?
* Josh lives in the middle of nowhere and goes to the city every now and then to upload everything at once
* Consider file size & internet access – proposed solution will vary between users

# Requirements

## Functional

* Back up all files every so often to archive
* Cleanup when files are unlikely to be accessed for a while?
  + ‘Recycle bin’ type situation maybe
* Retrieve files from archive
* Depends on access rules
* Multilingual search
  + Display name even if not accessible?

## Non-functional

* Easy to use with minimal tech knowledge
* Offline solution for users in remote areas

# Proposed solution

## Storage

### Options

* AWS – shitty documentation, slightly cheaper
* GCloud – easier to use, Harry’s used it before maybe?

### Concerns

* Kathrin mentioned issues with ethics etc
* (main options being Amazon and Google)
* Access to data not guaranteed (funding and whatnot)
* Unsure if any better options exist
* Better than bitrot I guess

### Conclusions

* Probably going to go with GCS but hoping to run into better options

### Backup

* GCS can do versioning – keeps X previous versions of files that have been overwritten
* Every month or so upload snapshot of files to archive?
* Can set as cron job for those with decent internet access
* Option to run manually

### Retrieval

* Probably fine not to check local files as they’re already there
* Check dropbox and archive both
* Multilingual search (for later)
* Allow for download if access permits

## Design

### Upload

* Can be just a script that gets run every now and then
* Requirements:
  + Root folder to upload to archive
  + Login details? (for both archive and Dropbox maybe?)

### Download

* May need GUI
* Login page / keep details in env?
* Figure out instructions to get accounts set up from non-owner

|  |  |
| --- | --- |
|  | * Search multilingual by default   (can add settings to change?)   * Download only available if access permits   Things to decide   * Folder or file results * Display unaccessible? |

* Run from a script?
* Figure out how to do that

## Logic

### Args/env

* Login credentials
* Root folder location (to start uploading from)
* Ignore list?
* Cleanup folder location
* Date of last upload?

### Upload

* [start process]
* crontab -e (edit file) crontab -l (to check)
* sudo service cron start/stop
* alternatively, run script when needed
* [refresh credentials]
* (read env file?)
* (set up account login stuff?)
* Upload files to archive
* (opt. upload to dropbox?)

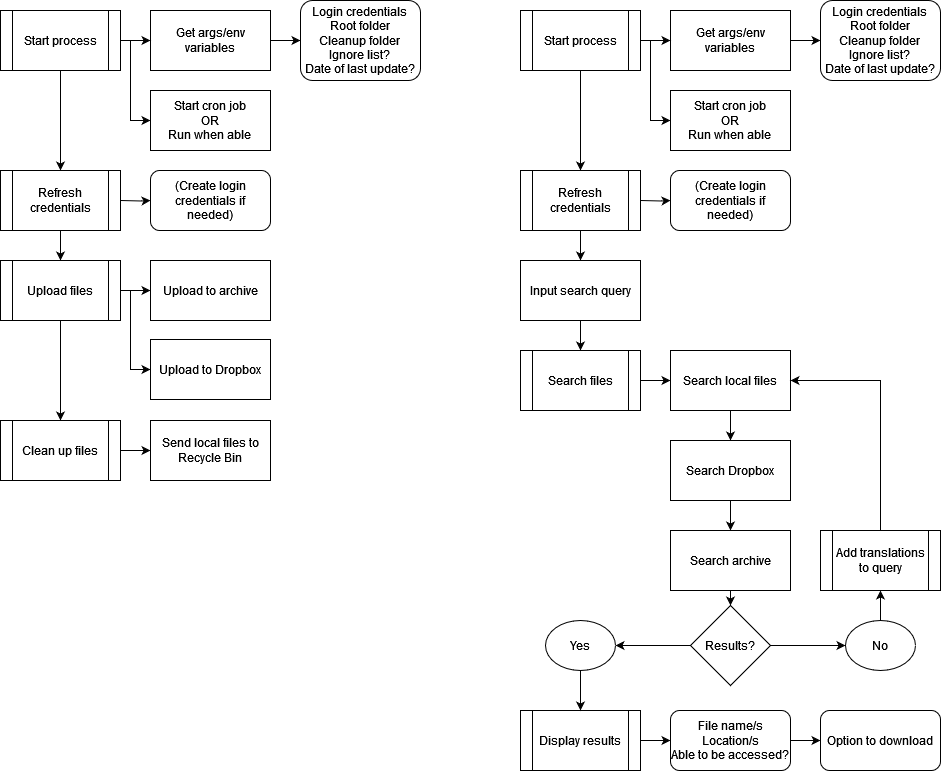
### Download

* Enter search term/s
* (search local files?)
* Search Dropbox files
* Use standard search (black box, up to implementation)
* Get all filenames, iterate through (no idea of efficiency)
* Search archive
* Get names, filter as necessary
* Multi-language search
* Search original string
* Dictionary of lists?

<https://www.emerald.com/insight/content/doi/10.1108/14684520710780458/full/html>

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5430129>

http://epa.niif.hu/02400/02461/00064/pdf/EPA02461\_acta\_polytechnica\_hungarica\_2016\_02\_195-207.pdf



# Considerations

## Storage platform

## Existing workflow

## Internet connection

## Tech experience & running

# Plan

## Tasks

## Testing

## Monitoring

## Timelines

# Implementation

## Google Cloud IAM

* Sign into Google Cloud account
* Install CLI
* Run `gcloud init`
* Create or set a project:
* Create: `gcloud projects create PROJECT\_ID`
* Select: `gcloud config set project PROJECT\_ID`
* Enable resource manager API:

`gcloud services enable cloudresourcemanager.googleapis.com`

* Create local auth credentials: `gcloud auth application-default login`
* Grant roles:

gcloud projects add-iam-policy-binding PROJECT\_ID

--member=”user:EMAIL\_ADDRESS”

--role=ROLE

### Other stuff

#### Install client library (Python)

pip install --upgrade google-api-python-client google-auth google-auth-httplib2

#### Modify roles

<https://cloud.google.com/iam/docs/write-policy-client-libraries>

#### Google Cloud login

`gcloud auth login`

#### Enable IAP (Identity-Aware Proxy)

gcloud iap web enable --resource-type=app-engine --versions=*version*

## Traversal & upload

### Requirements

* Archive folder location
* Authentication completed

### Steps

* os.walk() – get list of all files under archive root
* for all files, upload to archive
* file versioning enabled, so it should keep a certain number before old versions are lost

## Download

### Requirements

* Authentication completed
* GUI

### Steps

* user enters search query
* (maybe swap with next step: determine other language versions of query)
* search local files, Dropbox, and archive
* display results
* if download button pressed, download file/folder (????)

## Cleanup