TidyMe - Self Organizing Content Manager
Final Project Report

CS 4243/5493 Large Scale Database Management

The University of Texas at San Antonio

May 1, 2023

Submitted by:

Xandeep Varghese Alexander

Brandon Hernandez

Joy Kiprotich

Lissette Deleon

Supervised by:

Prof. Dr. Ritu Arora

Table of Content

I. What is TidyMe about? **4**
II. Why is it important? **5**
III. Who is the Target audience? **5**
IV. Implementation **6**
Frontend: Flutter Flow UI Development Environment 6
Frontend Design with Wireframes for UI: 6
Backend: 6
Application Architecture: 7
API Calls: 7
Google Language/Text API calls 7
Google Image Annotation API calls 7
Google Video Classification API 8
Google OAuth 2.0 Setup for the Refresh Token access to FlutterFlow API calls 8
1. Setting Google Project as an integration to the FlutterFlow integrated Firebase 8
2. Setting the Scope of the Project for APIs needed to get the access API Code 9
3. Get the refresh token using the API code of the Published google Project 10
4. Setting up the Authentication token request using the refresh token and client ID in the APP for continuous API calls from Flutterflow 10
App Development: 11
FlutterFlow Setup 11
App Framework and Flow 11
Large Data Storage Implementation 12
App Web Pages & Test Cases: 13
V. Implement Searching Feature **17**
VI. Testing **18**
Frontend: 18
Backend: 19
VII. Challenges: **20**
VIII. Team Contributions & Roles: **21**
IX. Final Documentation and Repository Links **22**
1. GitHub repository: 22
2. Final compiled App Link : 22
3. Miro Board Link: 22
4. Final Project Presentation: 22
5. Final Project report Link: 22
6. Final Video Link: 22
References **22**

Appendix.....23

I. What is TidyMe about?

The modern world is data driven and the data that we collect in a day is huge. Over time, it gets tedious to manage such large and varied data manually. This often creates a lot of cluttered data, which requires a great manual effort to organize and categorize by the data owner. In this project we are addressing this everyday concern by creating a cloud storage application that helps users to efficiently organize notes (text, docs), videos and photos by using cutting-edge Artificial Intelligence to classify contents into meaningful categories for the user.



Messy vs. Tidy (Picture courtesy signVOX)

We started off with a plan to organize the Youtube 8 Million Video Dataset to create a self-organizing file system. Then create a similar class/folder in the users' library based on the uploaded video by the user. Thus organizing data for the user's library of images, audio files, and videos.

The application will be able to allow users to log in securely and upload their content securely as a single file (Or multiple files in Batch Mode). The app will also feature the ability to receive text and photos from users. These will also be organized into the same categories or folders created for the users based on the current content database of the user's library of contents (Homepage).

The app will also try to get a keyword or search text from the user to provide relevant videos based on those keyword(s). For this, We will use Google Annotation services for video API, and a Metadata (annotation) of the video generated will be stored for each video or photo or text along with the content in the user's personal folder.

II. Why is it important?

Organizations and individuals have to spend a lot of money and time to organize raw contents. Most of this data will end up in deletion without capturing the essence of the content available in those data. We believe this app is important for such an audience to help them with the following:

- Come up with a data organization logic automatically.
- Come up with a data retention and deletion policy based on the content classification.
- Analyze the trending data across organizations created by multiple individuals collectively.
- Saving a lot of human effort/time and money, which could be directed to more content creation without worrying about organizing the data.

III. Who is the Target audience?

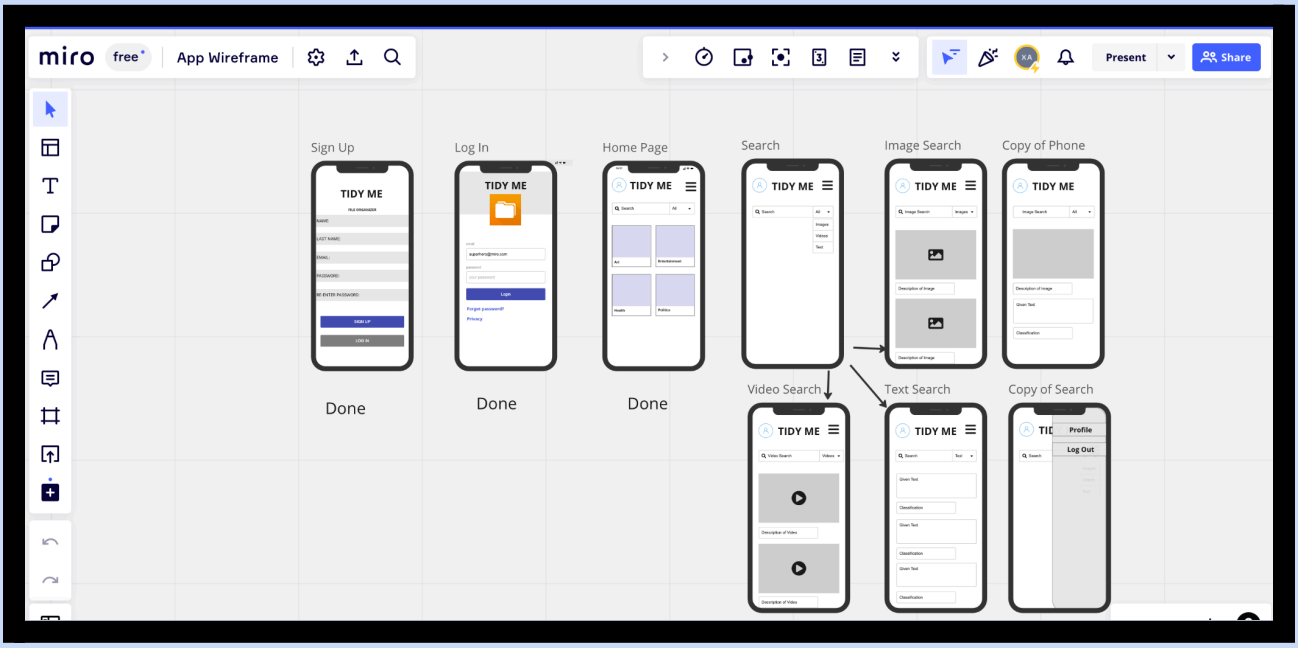
We know that we are creating many memories in the form of digital data such as Text, Images, and Videos throughout our day. For new-generation content creators, this app will help them organize their content to create their story by providing the creators with relevant content from their own data collection based on search by keywords.

For big organizations, this app will help them to optimize their space and manage data usage. This app will also help in understanding the trend analysis from the annotations created for each such file in their library. This will help them understand the word cloud trending with new content created in the organization by various people across their organization structure.

IV. Implementation

Frontend: Flutter Flow UI Development Environment

Frontend Design with Wireframes for UI:



Our Miro Dashboard used to create the App UI WireFrame Link

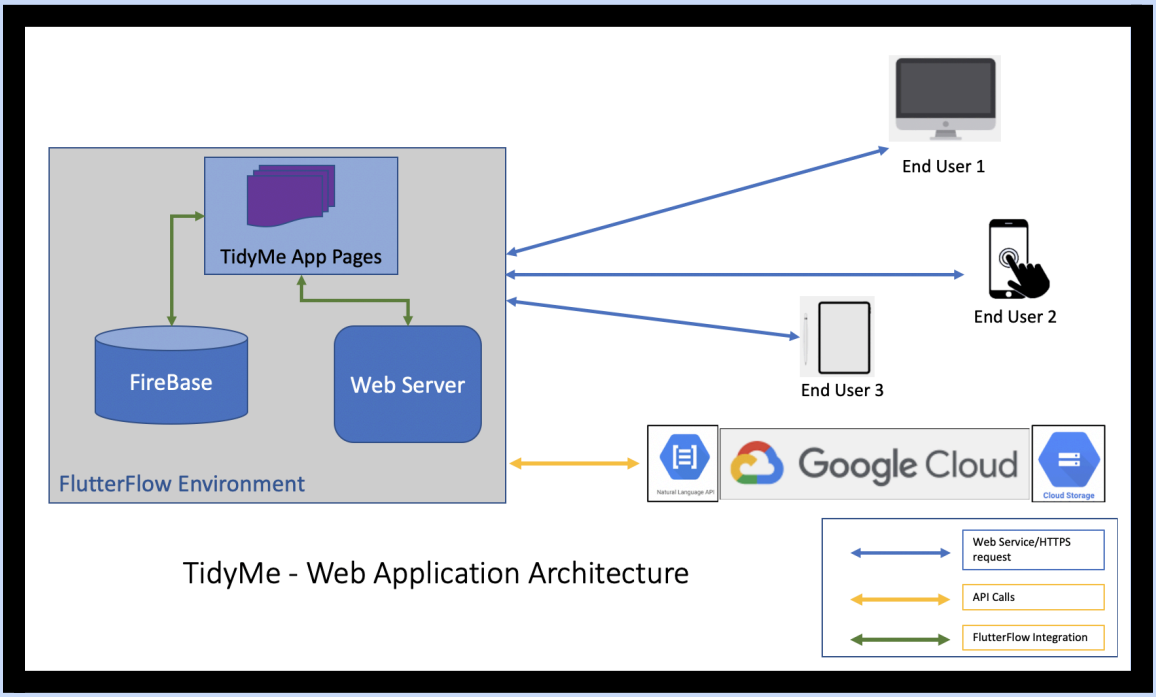
Backend:

1. FireBase DB
2. Google Cloud APIs
3. GitHub for Code backup and Version control repository

Database:

1. Google Drive as Backend for files Storage

Application Architecture:



The above image shows our application architecture explaining the component modules and communication direction.

API Calls:

Google Natural Language/Text API calls

The Google Natural Language API is used to classify text and documents using this app. This app receives the text and then uses a POST REST API method to call the Google API and transfer the needed data. The response of this API call will give a classification of the content of the text based on a google pre-trained text content classifier (version 2) into one of the classes mentioned here. This class is used as the container name to organize the text.

Google Image Annotation API calls

The Google Vision API is used to classify images into meaningful classes, and these classes are used to assign a folder name to the images. The class with the highest confidence value is taken to get the final class of the images. If there is no class with proper confidence, the value “others” is given as the folder name for that image.

Google Video Classification API

The Google Video Intelligence Annotation API was used to annotate the video. This is a batch process, and the first API call is to submit the video to the batch job called project operations to start annotation extraction from the video files. A second API call will be used to get the status of the operations to get the annotation labels. A category of a label with highest confidence value is mapped as the class of the video.

Google OAuth 2.0 Setup for the Refresh Token access to FlutterFlow API calls

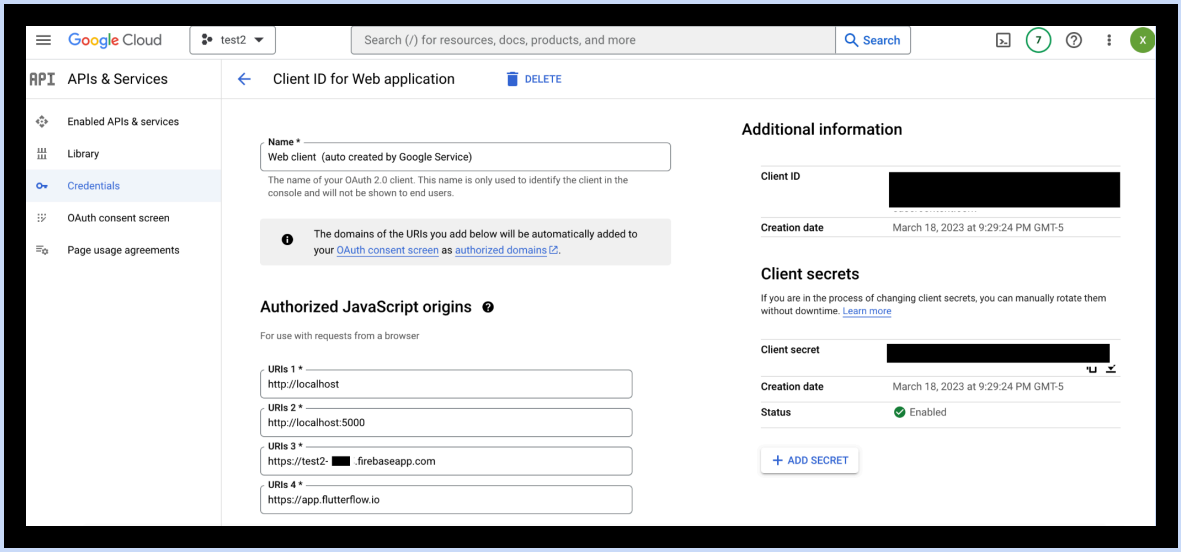
For multiple successful and interactive API calls from Flutter Flow App to Google Cloud, it is important to have a valid authentication token available to the App engine at all times. This is one of the most challenging parts of this App implementation.

The Google OAuth 2.0 refresh token obtained for an application endpoint (like our Flutter Flow App) was completed by implementing the below **four-step** process. The four-step process could not be added here in the documentation directly as a screenshot, as this includes keys that can open the app and the billing account for unauthorized personal Google account access of the members. [The screenshots are for informational purposes only.]

1. **Setting Google Project as an integration to the FlutterFlow integrated Firebase**

Created a new Project in Google Cloud and had the redirect URI along with the Authorized origin set to Flutterflow.io app urls. Enabled the above Classification APIs for this new project in Google Cloud project so that the APIs can access content and call these APIs without any restriction. The project helps to define the billing rates, Scope of the APIs available for use from the flutterFlow app.

A web client ID will be created by default for this project. We need to upgrade the web client ID credentials to have the redirect URI and authorized URI as FLutterFlow.io app. This will create the Client_ID and Client_Secret key needed for the API REST calls (GET, PUT, PUSH methods)



Web Client setup in Goole Project

2. **Setting the Scope of the Project for APIs needed to get the access API Code**

In order to access the APIs, we need to set up the scope of the access as needed for the APIs we will be using. In our case, we had the scope "https://www.googleapis.com/auth/cloud-platform" covering all the Classification APIs scope we needed. The scope has to be obtained my referring the Scope reference documentation in google cloud for the APIs that need to be added to complete the App.

Cloud Video Intelligence API, v1

Scopes	
https://www.googleapis.com/auth/cloud-platform	See, edit, configure, and delete your Google Cloud data and see the email address for your Google Account.

Cloud Vision API, v1

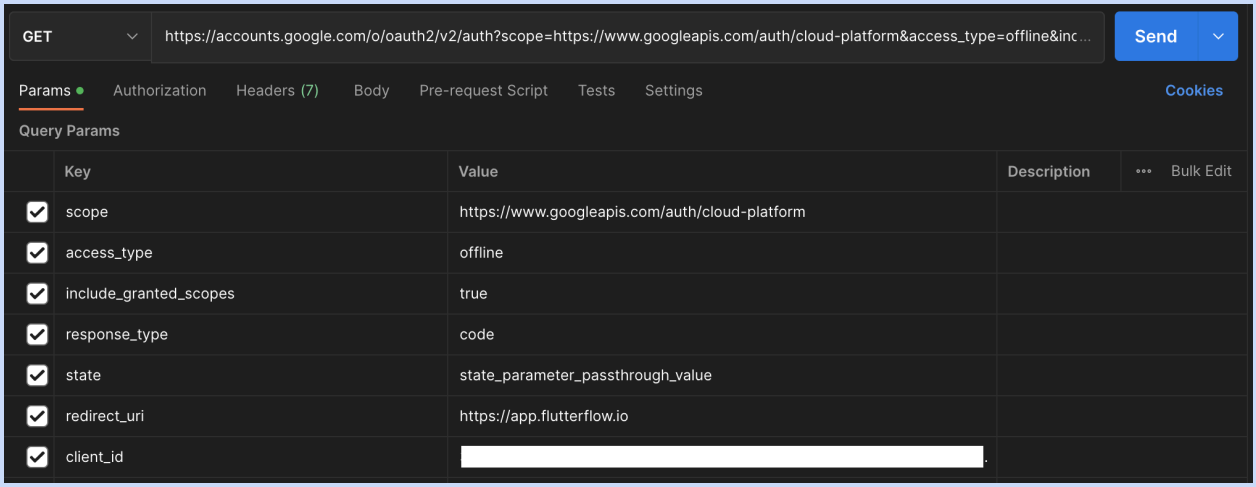
Scopes	
https://www.googleapis.com/auth/cloud-platform	See, edit, configure, and delete your Google Cloud data and see the email address for your Google Account.
https://www.googleapis.com/auth/cloud-vision	Apply machine learning models to understand and label images

Cloud Natural Language API, v1

Scopes	
https://www.googleapis.com/auth/cloud-language	Apply machine learning models to reveal the structure and meaning of text
https://www.googleapis.com/auth/cloud-platform	See, edit, configure, and delete your Google Cloud data and see the email address for your Google Account.

3. **Get the Refresh Token using the API Code of the Published google Project**

We then create an API code (need to create a new code for each request) for the same. It is also important at this point to have the google project published to production. To publish to production we need to define the access and privacy policies to google to get the publishing approved for the project. PostMan App was used to get these API Codes.



4. **Setting up the Authentication token request using the refresh token and client ID in the APP for continuous API calls from Flutterflow**

A GET REST API method is used to fetch the Authentication Token each time as needed using the above refresh key, Client_ID, and Client_Secret code. This API process needs to be called before calling any APIs to have the access tokens refreshed each time.

App Development:

FlutterFlow Setup

Our application's front-end design was created using FlutterFlow. After everyone downloaded the trial version, we shared a FlutterFlow Project named LSDM for collaborative App development. To satisfy our design criteria and produce the intended layout, we created a visual board UI Wireframe in Miro. Then using FlutterFlow's drag-drop App Development interface, we added components to match the blueprint we built on **Miro**.

App Framework and Flow

The App framework has three main components:

1. App Page Navigations and Flow control
2. FireBase DB - FlutterFlow integration to support faster user logins & Contents accessed from Firebase DB collections
3. REST API calls to Google Cloud Storage Buckets and Google Machine Learning/Classification APIs

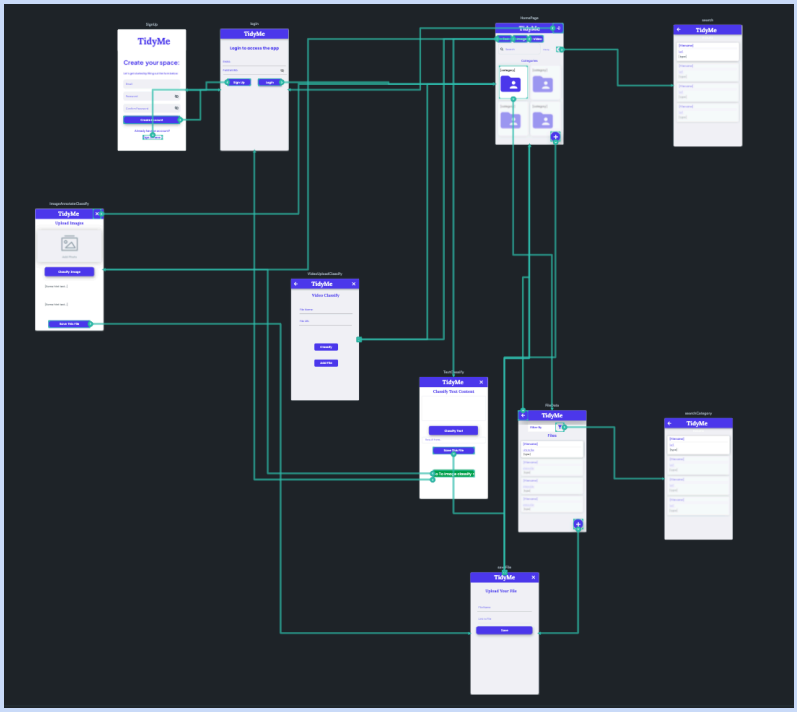
The App Page Navigations are the results of multiple iterative approaches to accomplish all the features required for the App and were designed on a visual collaborative board called Miro Board.

The FireBase DB- FlutterFlow integration is the backbone for this application, which supports faster user login and user account setup to access the secured customer files and profiles. Each user can access only their uploaded contents.

All the Artificial Intelligence/Content Classification functionalities of this app are achieved using secured and authenticated API calls to Google Cloud API for Natural Language, Vision and Video Intelligence provided by the Google Cloud. The details of these API calls are already defined in the API call section of this document.

TidyMe Page Navigation Flow

The figure below shows the page navigation flow and controls for the different pages in this app.



Large Data Storage Implementation

Data Collection:

1. For video we used Youtube Robust 8 Mil dataset
2. For Images, we used Kaggle's image dataset
3. For text, we used the BBC dataset
4. Caltech 256 Image dataset, has 30K+ images classified into 257 classes
5. Human Action short videos Dataset, UCF Center of Research in Computer Vision.

Classification:

Once data was collected, we used Google's API to test classification and annotation.

1. Video Classification
2. Image Classification
3. Text Classification

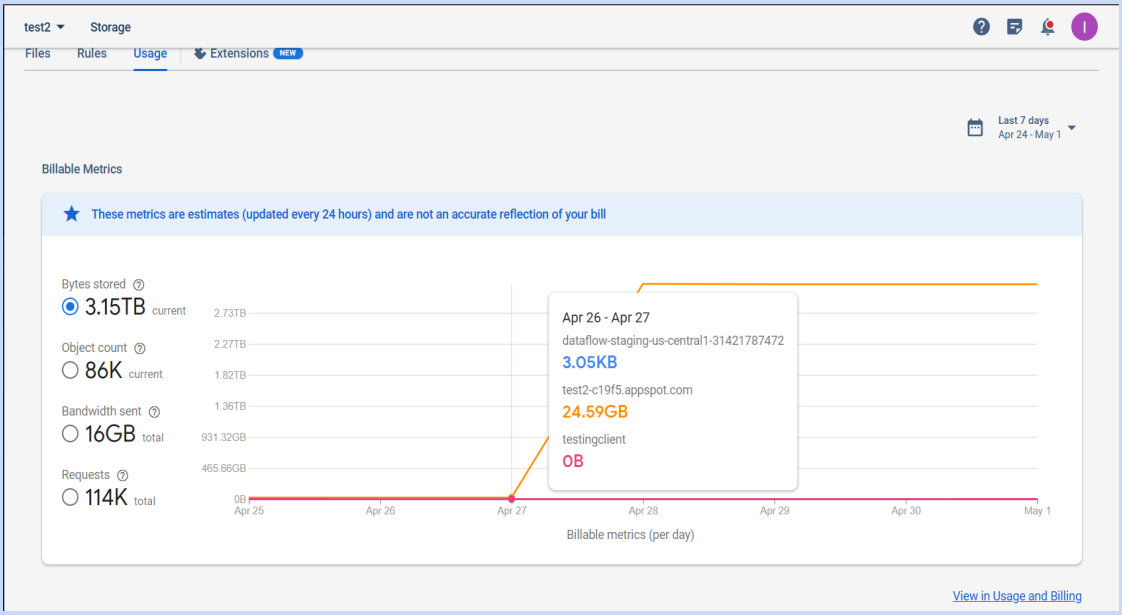
Building Application:

1. FlutterFlow
2. Firebase
3. PostMan App

Storage:

We stored the data within a bucket called **test2-c19f5.appspot.com in Google Cloud Storage.**

Cloud



Computing Services | Google Cloud 3.15TB stored data

FireBase Uploads for App-specific collections

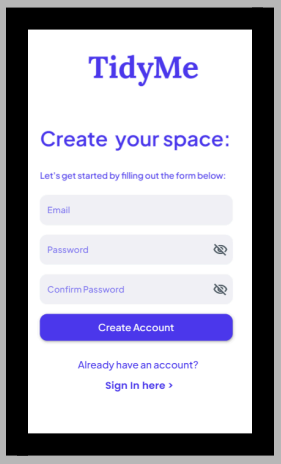
TidyMe, a cloud storage application that utilizes API to categorize videos, images, and text. We finalized on Firebase DB for the app database because it provides authentication, is compatible with Flutterflow and Google Cloud. Firebase also allows everyone from our group to work and view the Firebase console simultaneously. The team built 3 Firebase Collections (similar to Tables): **Users, Categories, and File Data**. The User database stores email and passwords for safe login. The categories database contains a list of categories of specific files that have already been classified. File Data stores user categorization and search annotations.

Google Storage Bucket upload

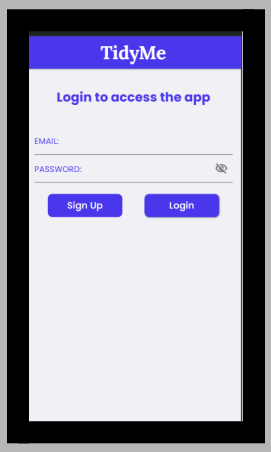
Our team picked Google Cloud Bucket because of its scalable storage, security, high availability, facilitates group work, and because we could obtain 300 dollar credits by signing up. Our cloud

storage stored Youtube and Kaggle picture datasets. We used Google's API video, image, and text classification to simplify user data categorization. Firebase storage moved files to the Google Cloud bucket.

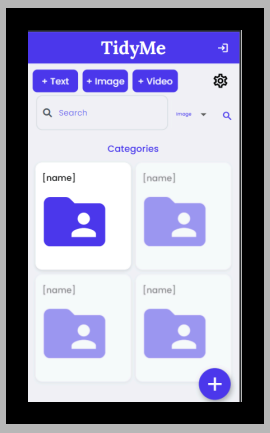
App Web Pages & Test Cases:

<table>
<tr>
<td>
</td>
<td><u>Sign Up:</u>

- Consist of three input fields (email, password, and password confirmation)
- Redirects to login button if user has an account already

<u>Peer Review Test Cases:</u>
- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame</td>
</tr>
<tr>
<td>
</td>
<td><u>Login:</u>
- Consist of two input field (email and password)
- Redirects to Sign Up Page if user doesn't have an account

<u>Peer Review Test Cases:</u>
- Test that the textfields (email and password) are visible and can receive input value
- Login button invokes the correct api call to authenticate
- API</td>
</tr>
</table>

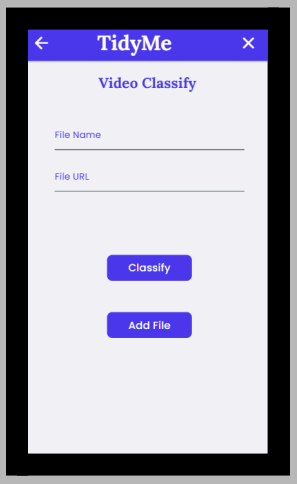


Home Page:

- Has a searchbar and dropdown menu
- you can upload your file
- User is able to view personalized profile
- Has a logout option
- Has a setting button

Peer Review Test Cases:

- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame
- Dispose of the search bar controller when page is not in use



Video Classification:

- Filename and google cloud storage bucket URL are needed for the uploaded file.
- Click on 'Classify' submits a video annotation job with Google Video Intelligence API. After a wait/delay time, another API call is used to retrieve the job results to the screen
- Click Add File to add that file to TidyMe personal Homepage

Peer Review Test Cases:

- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame

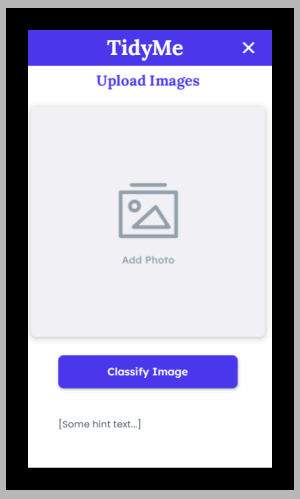
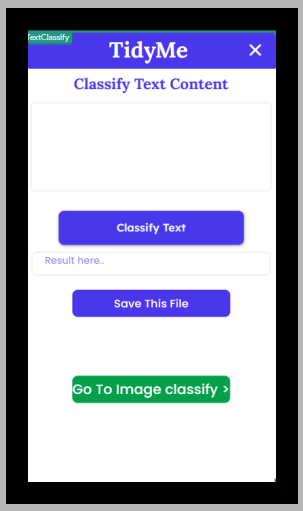


Image Classification:

- The image files are to be uploaded from the local library on the end user device.
- The Classify Image button will call Google Cloud Vision API and get all annotations for the Image.
- The highest confidence value annotation will be used as the image class and displayed below and a file is added to the storage.

Peer Review Test Cases:

- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame

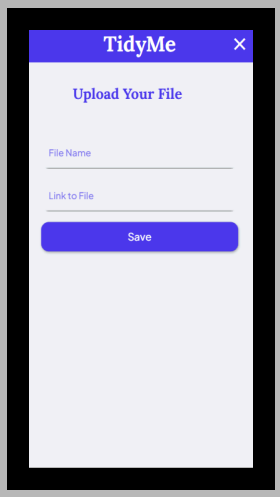


Text Classification

- User is prompted to put in what text they wish to classify
- Upon pressing 'Classify Text', api calls are made that return the different annotations associated with the given text.
- On the backend, we can extract a main category and a list of remaining annotations.

Peer Review Test Cases:

- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame

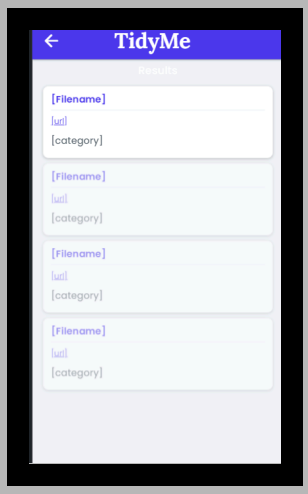


Saving files:

- From classification, we saved the type, category as well as annotations associated with the given file (text, image or video)
- User is prompted to enter the file name and url which is a link to where the file is stored.
- Saving stores all associated information into our database (user, filename, url, category, type and annotation)

Peer Review Test Cases:

- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame

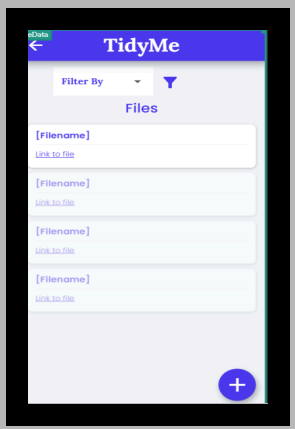


Searching files:

- When searching for a file, they will all appear in a list
- The individual file will display information like the name, the link to the file itself, and a classification tag
- Clicking the link will take the user to the file
- Each user can only see their own files and only have access to their own library

Peer Review Test Cases:

- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame



Filtering files:

- Filtering of the files is also allowed to make it easier when searching for them
- They can be filtered by Text, Images, and Videos
- It has a drop down menu to allow the user to select the format of the file they want to search
- Clicking the link will also take the user to the chosen file

Peer Review Test Cases:

- All text Input field receive text values
- Button Click Invokes the Actions and API correctly
- Navigation controls implemented correctly
- All navigation pages linked as per WireFrame

V. Implement Searching Feature

The Home Page's backend is a database with available categories based on the files that the user has classified and saved. To load available categories as folders, we queried our Categories database displaying said categories that are specific to the user with a limit of 10 folders being displayed in case the user has an extensive category list tied to them. This page offers search functionality with image, text, and video being available for filtering results. This is supported by our file data database. The schema for this database is the File name, user, URL, category, type, and annotation, with all of them being strings and annotation being a list of strings. The nature of the schema provides for easy searching by looking for the search term in the list of annotations and displaying a list of files that have that term. The user can also access files in the folders provided on the homepage, and within those folders, they are able to filter the results by file type that is image, video or text.

We implemented upload functionality for the user after classifying said files and the backend for this is also the file data database. To achieve this, we initialized two state variables for our application that are annotations, which is a list of strings, and currentCategory, which is a string. Both of these variables are not persisted, thus enabling them to change for every user as well as every action. Upon successful annotation/classification from the API call, the result is passed to two custom functions as a parameter. The first custom function splits the result, which is delimited by a forward slash '/', and returns a list that sets the annotations variables. The second function splits the result as well but only returns the value at index 0 which then sets our current category variable. The user is only prompted to enter the file name as well as a URL to the location of the file, and a new entry in our file data database is made for that user. It is important to emphasize that files cannot be saved if the annotation/classification API is unsuccessful since we will not be able to extract category or annotation data which are required fields.

VI. Testing

To ensure we maintain high quality, we have a very detailed iterative testing plan which requires each of the completed development tasks to be handed over to another developer to conduct initial peer-review comments. Once these peer-review comments are rectified, the developer has to complete the final testing to ensure all functionality and requirements are met for that module. The final integration testing and demo run was completed in a group session with all members and feedback was given to the original developers to complete and the final integration testing was completed.

Frontend:

Section	Requirement	Appropriate modules implemented?
Start Screen		
	Allow new users to the Sign Up Account page.	☑ ~~yes~~
	Allows users to redirect to the login page	☑ ~~Yes~~
Sign Up		
	Allows new users to create a new account	☑ ~~yes~~
	Allow new users to register email and password to use the application	☑ ~~yes~~
	Allows user to enter their log in (when redirected)	☑ ~~yes~~
Login		
	Allow returning users to enter their email and password to login	☑ ~~yes~~
	Allow users to log in to the application with the correct email address and password combination	☑ ~~yes~~
	Allow new users to be redirected to Sign Up page	☑ ~~yes~~
HomePage		
	Allows to view the user's current files that have been save with the users account	☑ ~~yes~~
	Allows user to navigate to Text, images, and Video classification page	☑ ~~yes~~
	Allows to search files	☑ ~~yes~~
	Allows users to view their currently saved files associated with their account.	☑ ~~yes~~
	Allows user to logout (redirected to sign up page)	☑ ~~yes~~
	Allows user to navigate to upload image page	☑ ~~yes~~
Video Classification		
	Allows user to classify a file	☑ ~~yes~~
	Allows user to add video file	☑ ~~yes~~

	Allows user redirect to the previous page	☑ ~~yes~~
	Allows User to Log Out	☑ ~~yes~~
Image Classification		
	Allows user to upload image	☑ ~~yes~~
	Allows users to classify image	☑ ~~yes~~
	Allows users to logout (redirected to login)	☑ ~~yes~~
Text Classification		
	Allows users to classify a textfile	☑ ~~yes~~
	Allows users to Save a file	☑ ~~yes~~
	Allows users to navigate to image classification	☑ ~~yes~~
Saving Files		
	Allows user to upload file name	☑ ~~yes~~
	Allows user to Save file	☑ ~~yes~~
	Allows user to logout	☑ ~~yes~~
Searching Files		
	Allows user to search	☑ ~~yes~~
	Allows user to go back to the previous page the user was on	☑ ~~yes~~
	Allows user to logout	☑ ~~yes~~
Filtering Files		☑ ~~yes~~
	Allows user to filter image	☑ ~~yes~~
	Allows user to filter video	☑ ~~yes~~
	Allows use to filter text	☑ ~~yes~~
	Allows user to go back to the previous page the user was on	☑ ~~yes~~
	Allows user to logout	☑ ~~yes~~

Backend:

Section	Requirement	Appropriate modules implemented?
Data transaction		
	Creates new users from Tidy Me app (Verify at Backend)	☑ ~~yes~~
	Confirms User information from Frontend (for Existing users)	☑ ~~yes~~
	Upload images from Frontend (verify at Backend)	☑ ~~yes~~
	Able to upload text from Frontend (verify at Backend)	☑ ~~yes~~

	Able to upload video from Frontend (verify at Backend)	☑ ~~yes~~
	New files get assigned to the users application (verified at backend)	☑ ~~yes~~
	Allow users to enter secured password (hashed)	☑ ~~yes~~
Required Table		
	Firebase collections are established to support the application's data storage needs with appropriate fields and data types.	☑ ~~yes~~
	Create the fields as per the requirements specified. Also verified the datatypes	☑ ~~yes~~

VII. Challenges:

1. Getting Google Cloud - Refresh Token setup for the App had a lot of difficulty controlling the scope needed for all APIs in the APP. It took a toll on all the members of the team to complete setting up the integration of the FlutterFlow project to google cloud, setting the scope of the API and data controls, and publishing apps in the google cloud to authorize an endpoint to make a lot of API calls to the Google cloud and receive responses.

2. App development using REST API documentation was a challenge and to get all headers in the format required by the app needed effort to go beyond Google documentation and multiple youtube video lectures on the APIs. We also had to chat with the google and FlutterFlow support teams to get align our API call parameters to have successful

3. Controlling the API responses to receive the highest confident classification and implementing a logic to finalize on a class to be added to the folder name of each file.

4. Google Cloud Storage Bucket APIs are not giving back the bucket storage link of the files uploaded. We have to go with first uploading the data to Google cloud and using those google utils link to map the uploads to google file locations.

5. Being able to collaborate on FlutterFlow as it only allows one user to edit at a time without subscription.

VIII. Team Contributions & Roles:

Joy Kiprotich (Project Leader)
Lissette Deleon (Secondary Project Leader)
Brandon Hernandez
Xandeep Varghese Alexander

Contributions	Team
Git configuration	Joy
FlutterFlow Setup	Lissette
Firebase Setup	Brandon
Google cloud Setup	Xandeep
Data Collection	Joy
Upload Image Dataset	Lissette
Upload video Dataset	Xandeep
Upload audio Dataset	Brandon
Sign Up Page	Lissette
Login Page	Joy
Home Page	Brandon
Video Classification	Xandeep
Image Classification	Joy
Text Classification	Lissette
Saving Files	Brandon
Searching Files	Joy
Filtering Files	Xandeep

IX. Final Documentation and Repository Links

1. GitHub repository:

https://github.com/joyee-c/LSDM

2. Final compiled App Link :

(For running the code, sign up with an account and access the App)
https://app.flutterflow.io/run/enQjGoUsrBVoUBRkVWPH

3. Miro Board Link:

https://miro.com/app/board/uXjVMfYskcU=/?share_link_id=239112030360

4. Final Project Presentation:

LSDM- Powerpoint

5. Final Video Link:

https://www.youtube.com/watch?v=6j9aHk2kmaw

References

1. Google Cloud Storage and API Authorization - OAuth 2.0 protocol, https://developers.google.com/identity/protocols/oauth2, Google Inc
2. Video Classification API (Google Video Intelligence Annotation API), https://cloud.google.com/video-intelligence/docs/annotate-video-command-line, Google Inc.
3. Image Classification Algorithm and API (Google Vision Image Annotation), https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate, Google Inc.
4. Text Classification Algorithm (Google Natural Language), https://cloud.google.com/natural-language/docs/classifying-text, Google Inc.
5. Google Cloud Scope by API reference, https://developers.google.com/identity/protocols/oauth2/scopes, Google Inc.
6. Miro Visual Collaborative Platform, https://miro.com/, RealtimeBoard, Inc.
7. YouTube-8M Segments dataset, https://research.google.com/youtube8m/, Google Inc.
8. FlutterFlow Documentation, https://docs.flutterflow.io/, Google Inc.
9. GitHub Documentation, https://docs.github.com/en, GitHub Inc.
10. Postman API App, https://www.postman.com/, PostMan
11. Action Recognition Dataset, https://www.crcv.ucf.edu/data/UCF101.php, UCF Center for Research in Computer Vision.
12. Caltech 256 Image Dataset, https://www.kaggle.com/datasets/jessicali9530/caltech256, Caltech.

Appendix

Documentation & Demo

Documentation is a collective team effort and each developer completed the respective sections then final proofreading was also completed by all team members in a group discussion and corrections were applied to complete the final Project report, Project Demo & Project slides.