Fall 12-30-2020

Cat Tracks – Tracking Wildlife through Crowdsourcing using Firebase

Tracy Ho
San Jose State University

Cat Tracks – Tracking Wildlife through Crowdsourcing using Firebase

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Tracy Ho

Dec 2020

The Designated Project Committee Approves the Master's Project Titled

Cat Tracks – Tracking Wildlife through Crowdsourcing using Firebase

By

Tracy Ho

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2020

Dr. Ben Reed	Department of Computer Science
Dr. Mike Wu	Department of Computer Science
Dr. Anthony Giordano	SPECIES

Acknowledgement

First, I would like to express my deepest gratitude to my advisor, Dr. Ben Reed, for his support and guiding me throughout this entire project. I am grateful that he was willing to allow me to work on a Master's Project and it was an honor having him as an advisor.

Next, I would like to extend my thanks to my committee members for this project, Dr. Mike Wu and Dr. Anthony Giordano, for their encouragement, suggestions, and time towards this project.

Then, I would also like to give a big appreciation to Azael Zamora, Jonathan Kabongo, and Nathan Zadkovsky. Three San José State University undergraduate students who volunteered their time to assist in the research and development of Cat Tracks.

Finally, I am sincerely grateful to all my family and friends for all the support I have received throughout the entire process of my continued education.

Abstract

Many mountain lions are killed in the state of California every year from roadkill. To reduce these numbers, it is important that a system be built to track where these mountain lions have been around. One such system could be built using the platform-as-a-service, Firebase. Firebase is a platform service that collects and manages data that comes in through a mobile application. For the development of cross-platform mobile applications, Flutter is used as a toolkit for developers for both iOS and Android. This entire system, Cat Tracks is proposed as a crowdsource platform to track wildlife, with the current focus on California mountain lions. By building such a system, researchers could use the data to save the lives of many mountain lions.

Keywords – Data collection; data management; file storage; Firebase; Flutter; geolocation; Google Cloud Platform, mobile development; NoSQL

Table of Contents

List of Figures

I. Introduction

It has been stated that roadkill is considered among the biggest killer of mountain lions in the state of California by the California Department of Fish and Wildlife [3]. The California Department of Fish and Wildlife have also stated that more than 100 mountain lions are killed in a year on California roads. The unfortunate effect of this is lowering the mountain lion population and reducing their gene pool diversity [7]. Therefore, to preserve the mountain lion population, it has become increasingly important to track the mountain lions that are roaming around the area and study their movements to give insights into where they may be residing.

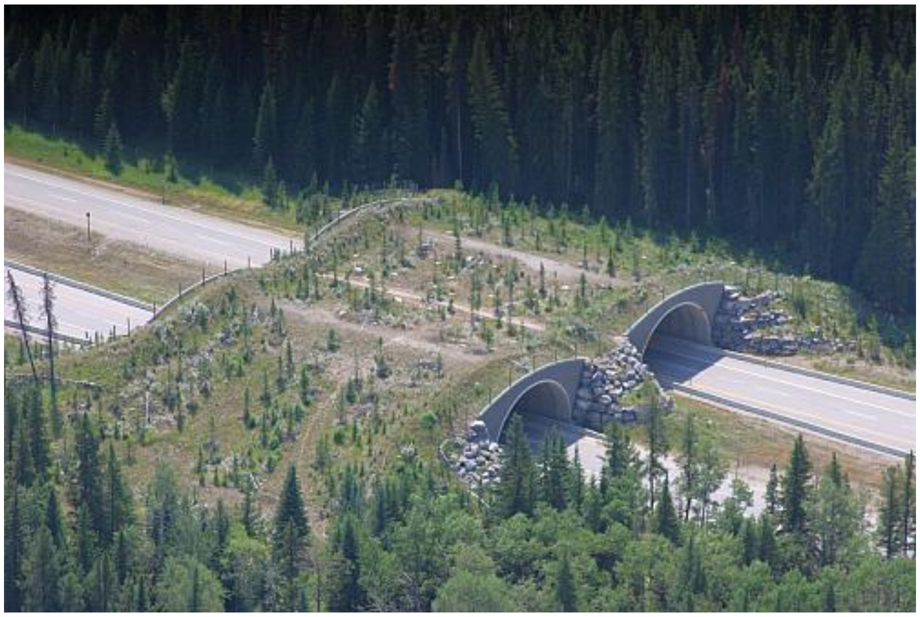


Figure 1. Photo of wildlife overpass. Credit: enjoythesilence.today [12]

This data can be used by wildlife specialists to designate areas for mountain lions to be restricted from the traffic coming in and place wildlife crossings, such as the one shown in Figure 1, in the effort of reducing the amount of roadkill on mountain lions [12].

Therefore, this project, Cat Tracks, is proposed to assist in the preservation of mountain lions. At its core, Cat Tracks is an application with a data collection backend to collect information on animals so researchers may understand what wildlife is encountering traffic, and how they are affected. Cat Tracks aims to provide a platform for crowdsourcing information on animals, with the use case on mountain lions in the state of California.

For this paper, one of the main goals of Cat Tracks is to build and maintain a data collection and management system. Having a well-organized and maintainable system will allow the data to be processed efficiently and developers are able to focus more on developing the apps themselves. Thus, there are two parts to the Cat Tracks project. Building a highly scalable backend system to collect and manage data, and the mobile app itself to perform the actions to obtain the data to be sent and processed to the backend system.

One of the challenges that had to be addressed throughout the development of Cat Tracks was security. There had to be a way to ensure that malicious users were not able to gain access to the data or be able to write in the data that they did not have permission to write into.

Another challenge that had to be considered was the isolation of data. There had to be a way to ensure that users were able to create and delete their own photos, while other users were only allowed to view the photos as well as collect the associated metadata. It is also important that changes to the data are reflected on the app as they are happening.

Privacy is another aspect that had to be considered when developing the backend aspect of Cat Tracks. There had to be a way to make sure the data was only accessed by authorized users that had permission to view the data. For the public collection, there had to be a way to ensure that the data shown was not able to identify the user that uploaded the image in any meaningful way.

Then to go along with privacy concerns, there are two databases used for Cat Tracks. One that was specifically only for an individual user, and another that was open to the public to view any of the data contained in that collection and collect the metadata. Due to the use of two different databases that used the same data, there had to be a way to maintain both databases to make sure that the data was consistent and that the data in the public collection was automatically adding in data as new data was being generated from the users themselves.

This paper is divided into six parts. The first part will discuss the underlying backend technology that was used for this project and what benefits it had to be chosen, along with the research related to this project. The second part will showcase the design of the project such as the architecture used for the overall platform. The third part will explain more in detail about the implementation of Cat Tracks. The fourth part will go into more of the limitations that went into Cat Tracks. The fifth part will discuss some of the future work planned for Cat Tracks. Then the sixth part will conclude the paper and go over future work.

II. Background

Firebase

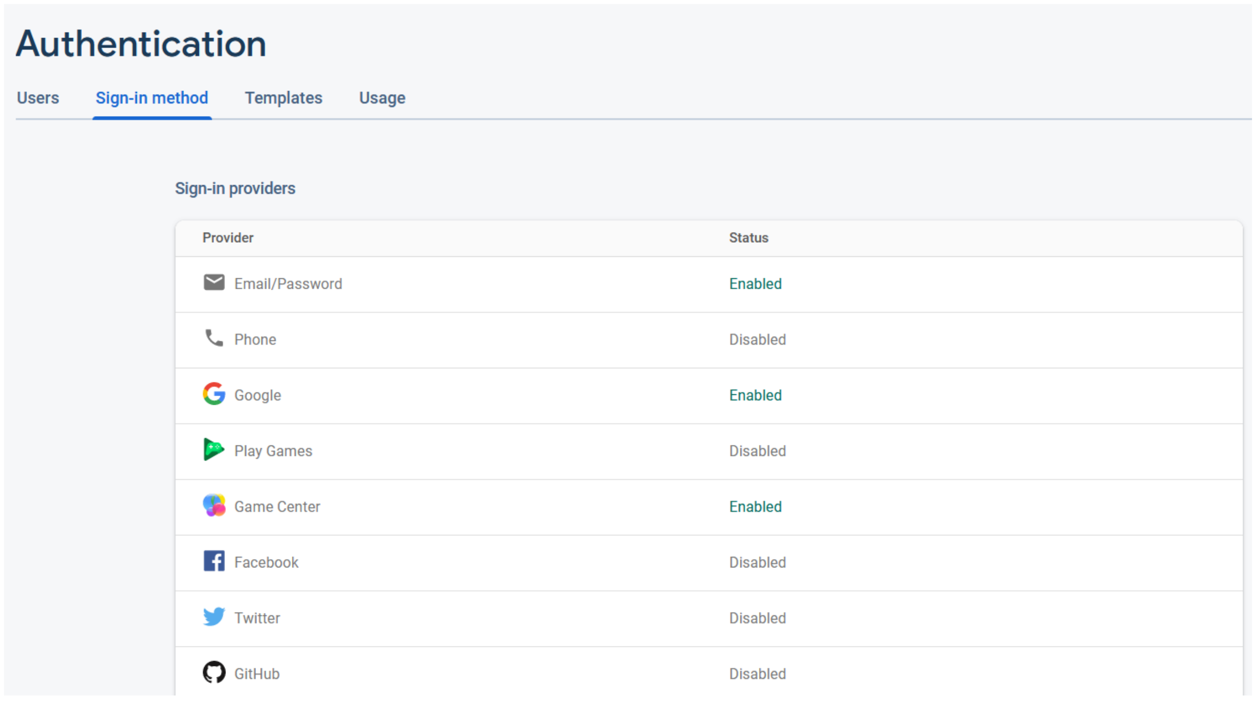
Firebase is a platform from Google to build web and mobile applications [6]. It is a Platform-as-a-Service (PaaS)[15], providing many backend tools that developers can utilize when building their applications. The appeal is that users will not need to build together their own backend technologies to use as part of their development. They can all be provided by Firebase itself. Firebase can handle all the backend technologies so that users may focus more on developing their app.

Due to Firebase providing its own technologies to store data, it can scale to high levels [6]. When developing a web or mobile application, it is important for it to take account of a possible large number of users connecting to the server and storing their own generated data as well. There is potentially no limit to what can be stored from the user.

Some of the many features Firebase provides is its authentication, file storage system, NoSQL database, cloud functions, machine learning tools, and analytics. It can also integrate with other Google products and services such as Google Cloud Platform. For this project, Firebase was chosen as the underlying backend technology, so during development it was not needed to search out different technologies to build a backend infrastructure. It was all provided just by using Firebase itself.

Firebase Authentication

Firebase Authentication provides different ways users can login to an application, including many 3rd-party authentications, such as using Google, Facebook, Twitter, and GitHub. This convenience allows users to have a variety of ways for logging into the application. It is also a secure way to handle authentication since app developers do not need to worry about building their own authentication system. Users also have the convenience of not needing to create an individual account using the standard email and password method. They can use a 3rd-party account that they feel comfortable using as their login.


Authentication

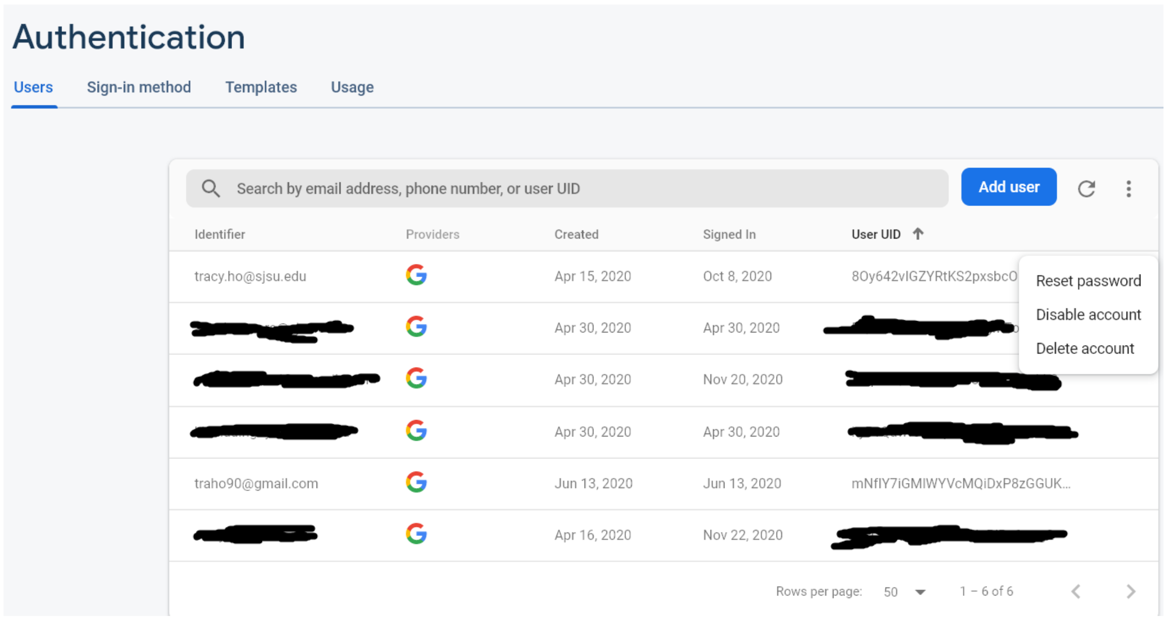
Sign-in providers

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Enabled
Play Games	Disabled
Game Center	Enabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled

Figure 2. List of providers for sign-in methods

As shown in Figure 2, there are a variety of providers for federated logins. The app developers using Firebase can decide which sign-in provider they want to incorporate into their application.

Managing users through Firebase is also convenient. It is simply a list of all the users that have created an account within the application. Furthermore, there is even a way to add users straight into Firebase by just providing an email and password.

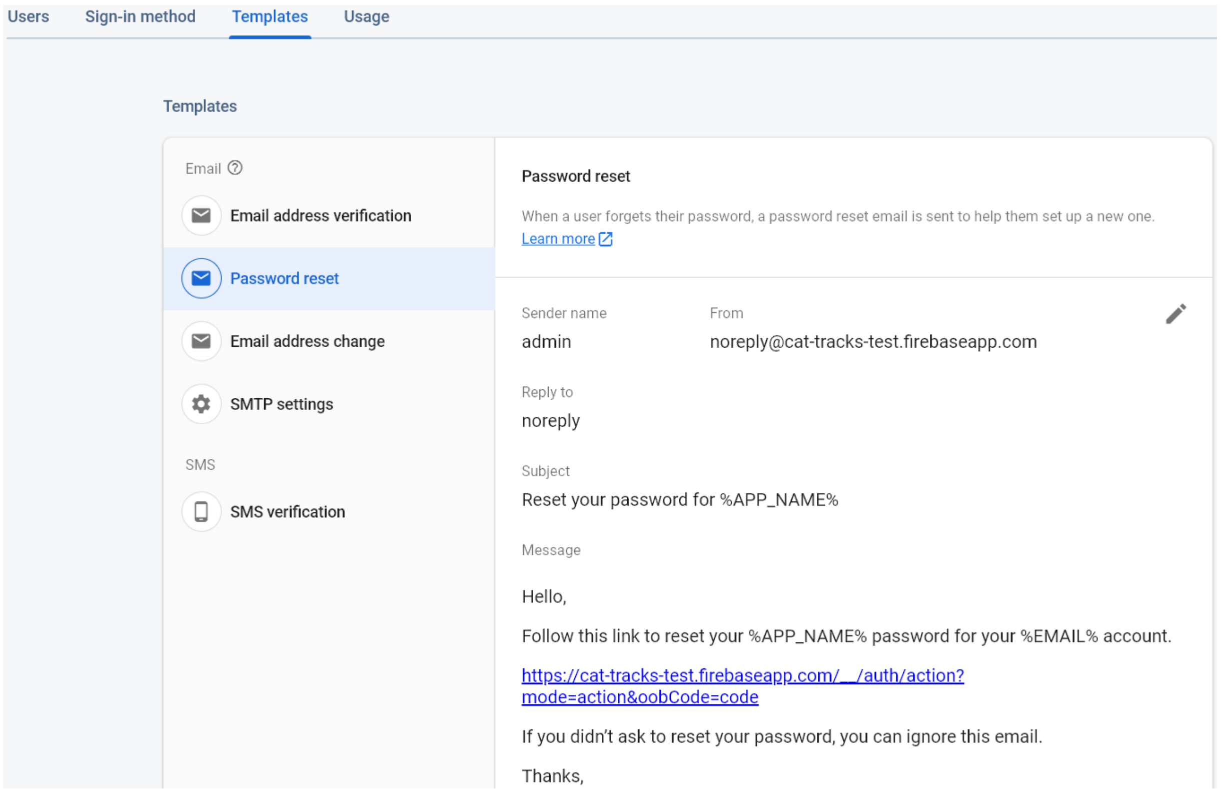

Authentication

Identifier	Providers	Created	Signed In	User UID ↑
tracy.ho@sjsu.edu	G	Apr 15, 2020	Oct 8, 2020	8Oy642vIGZYRtKS2pxsbcO
[redacted]	G	Apr 30, 2020	Apr 30, 2020	[redacted]
[redacted]	G	Apr 30, 2020	Nov 20, 2020	[redacted]
[redacted]	G	Apr 30, 2020	Apr 30, 2020	[redacted]
traho90@gmail.com	G	Jun 13, 2020	Jun 13, 2020	mNfIY7iGMlWYVcMQiDxP8zGGUK...
[redacted]	G	Apr 16, 2020	Nov 22, 2020	[redacted]

Figure 3. List of users for a Firebase system

From this UI as displayed on Figure 3, some basic user account information can be seen, the identifier, which is the user's email or phone number, the provider, which in the case for Cat Tracks are Google accounts, the dates for when the account was created and last signed in, and the User UID so Firebase has a way for the developer to identify and keep track of individual users. With this, app developers can easily manage their users without the need of a separate database management system to store their users' credentials.

One other feature about Firebase Authentication is being able to set email templates to be sent to users. Developers can set templates for different conditions and requests for the email addresses to receive.



Password reset

When a user forgets their password, a password reset email is sent to help them set up a new one.
Learn more

Sender name
admin

From
noreply@cat-tracks-test.firebaseapp.com

Reply to
noreply

Subject
Reset your password for %APP_NAME%

Message

Hello,

Follow this link to reset your %APP_NAME% password for your %EMAIL% account.

https://cat-tracks-test.firebaseapp.com/__/auth/action?mode=action&oobCode=code

If you didn't ask to reset your password, you can ignore this email.

Thanks,

Figure 4. Firebase page to set email templates

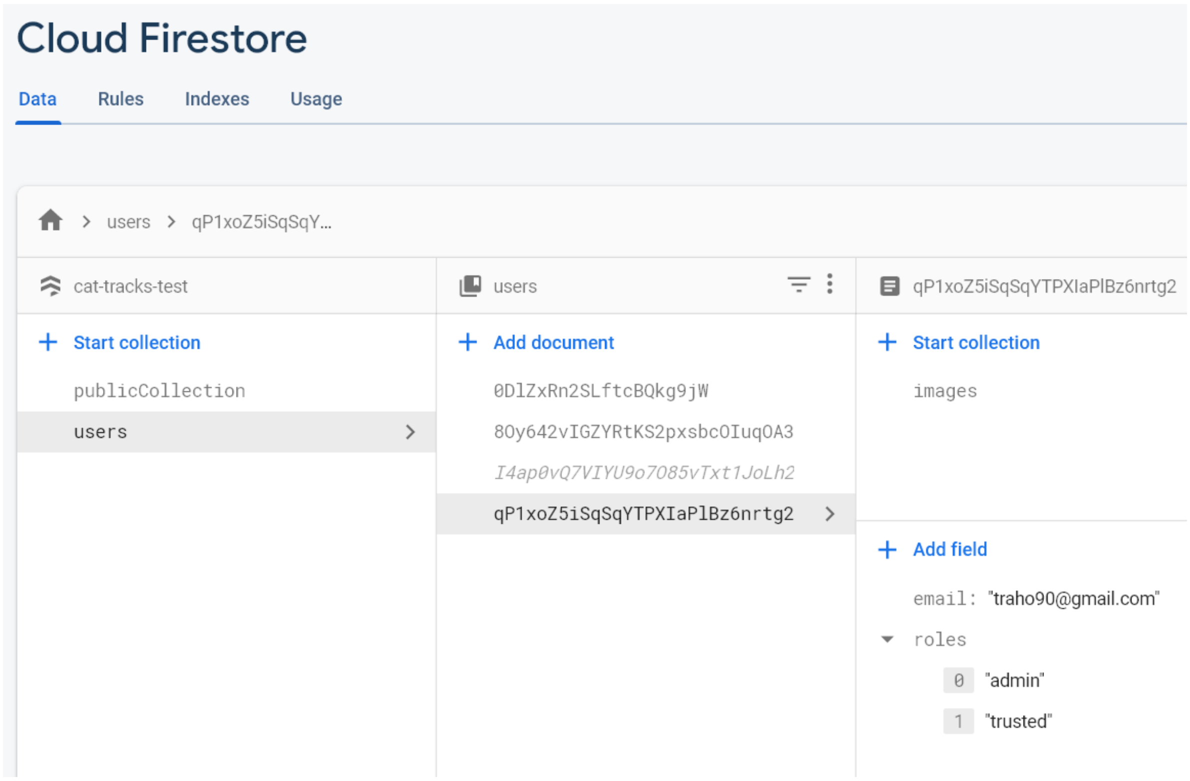
There are a couple different email templates that developers can customize to be sent based on the criteria as shown in Figure 4, with these being basic templates that applications use. The advantage with Firebase provided email templates is that app developers do not need to use a separate email service to send out automated emails. Firebase can send out those emails itself as part of the backend service.

Cloud Firestore

Firebase offers two cloud-based database solutions, Realtime Database and Cloud Firestore. Both database solutions are also of NoSQL databases [6]. The Realtime Database stores data as a large JSON tree, and the structure of the database has a flat design without conforming to any normalization that would be required from a relational database [9]. This makes it ideal to store simple data that do not require complex queries.

However, for the scope of this project, Cloud Firestore was chosen as the database solution. One reason being that Firestore is the newest database developed by Firebase, meaning that it is more likely to be supported with additional updates and new features. Firestore also performs faster queries and scales further than Realtime Database [6]. Since Firestore does not use SQL, that means that SQL injection attacks would not be possible against Firestore [13].

Cloud Firestore stores its data into documents, which are then organized in collections. Each Document may have its own subcollection. This data model creates a flexible, hierarchical data structure for collecting and managing data [6].



publicCollection
users

0DlZxRn2SLftcBQkg9jW
80y642vIGZYRtKS2pxsbcOIuq0A3
I4ap0vQ7VIYU9o7085vTxt1JoLh2
qP1xoZ5iSqSqYTPXIaPlBz6nrtg2

images

email: "traho90@gmail.com"
roles
0 "admin"
1 "trusted"

Figure 5. Cloud Firestore hierarchy of collections and documents

As an example shown in Figure 5, users is a collection of UIDs, which each UID is a document containing fields, such as email and roles, for the data. Each UID also contains images as a subcollection.

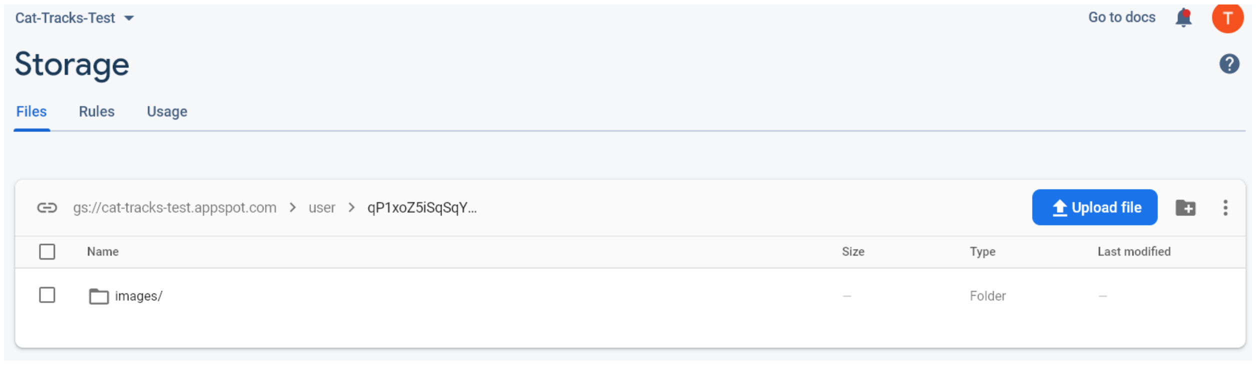
Expressive querying is another capability that Firestore has for managing data [6]. Queries can retrieve a single, specific document or all the documents contained in a collection matching the query parameters. In addition, the queries can include a combination of filtering and sorting to present only a selection of the data. By default, the queries are indexed, as the performance is based on proportion of the result set rather than the data set. To go along with these queries, Firestore provides real-time updates with data synchronization on any connected device, with the design of making efficient single fetch queries.

One feature that Firestore has to offer is the offline support with caching data that the app is using. This means that the app can read, write, and query data even as the device is offline. Then when the device has an internet connection, any changes can be synchronized back to Firestore [6]. Furthermore, Firestore is designed to scale to the vastly large amounts of big data that gets generated by users.

Cloud Storage

As part of their backend services, Firebase also has Cloud Storage as their object storage service [6]. Much like Firebase's database solutions, Cloud Storage is capable of scaling as an app grows larger. When developing with Cloud Storage, the Firebase SDKs for Cloud Storage also contain Google security, making for safe uploads and downloads no matter what the network quality is. It is also integrated with Firebase Authentication so that developers have a simple solution for handling user authentication when using Storage [6]. This ensures that only the appropriate users are viewing files that they can view. Furthermore, the operations for Cloud Storage are robust, meaning that uploads and downloads will resume where they stopped if there is a loss in connection. This will save time and bandwidth on the users of the application.

Firebase Cloud Storage is stored in Google Cloud Storage buckets. Meaning that it integrates with Google Cloud Platform, which allows the application to scale automatically. By using Firebase as the data collection and management system, developers will not need to look for a separate cloud storage system. They would be using Firebase's Cloud Storage, which is through the Google Cloud Platform. Cloud Storage for Firebase itself works much like a file system.



gs://cat-tracks-test.appspot.com > user > qP1xoZ5iSqSqY...

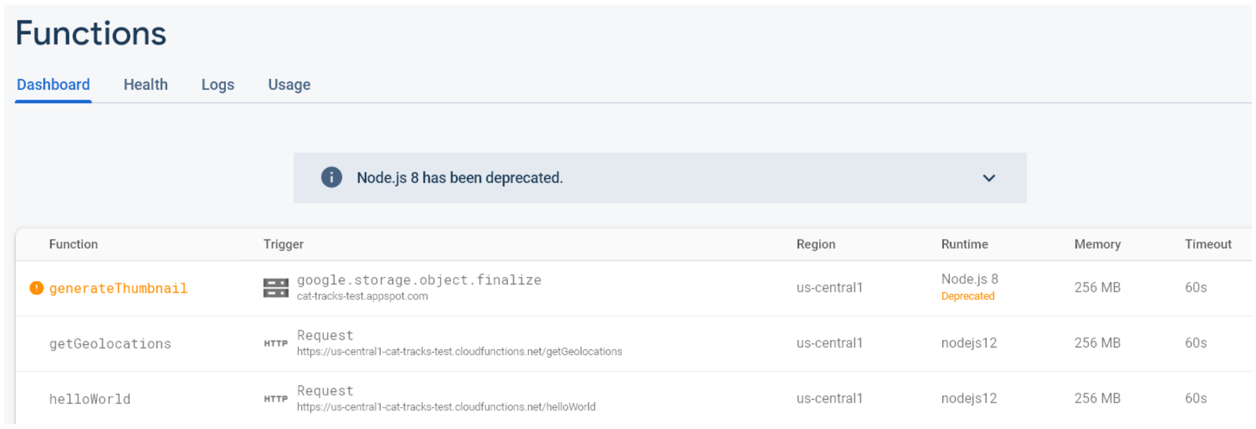
Name	Size	Type	Last modified
images/	–	Folder	–

Figure 6. Storage directory with images folder

It has folders which can contain files or other subfolders to contain additional files, as shown in Figure 6. With this, files can be organized and managed to fit the developer's needs.

Cloud Functions

Cloud Functions is a framework that can run backend code from the cloud as a response to events happening from Firebase or HTTPS requests [6]. Developers can write code in JavaScript or TypeScript and have the code sit in Google's cloud, which runs in a managed environment. The advantage with this approach is that there is no need for a developer to run and manage their own servers to listen for events to happen to activate code, Firebase can handle that instead.



Node.js 8 has been deprecated.

Function	Trigger	Region	Runtime	Memory	Timeout
generateThumbnail	google.storage.object.finalize cat-tracks-test.appspot.com	us-central1	Node.js 8 Deprecated	256 MB	60s
getGeolocations	HTTP Request https://us-central1-cat-tracks-test.cloudfunctions.net/getGeolocations	us-central1	nodejs12	256 MB	60s
helloWorld	HTTP Request https://us-central1-cat-tracks-test.cloudfunctions.net/helloWorld	us-central1	nodejs12	256 MB	60s

Figure 7. Cloud Functions created for Firebase application

Once a Cloud Function is written and pushed into the Google Cloud, Firebase can show the functions that were created on a web page. This lets the developer see what functions are part of the Firebase project on a Dashboard, as shown in Figure 7, any errors as part of the Health of the of the functions, and shows what the activity has been going on from the logs and the usage of the Cloud Functions. With this information on hand, developers can see which functions are being triggered, if there are any issues with them, and how they could adjust for billing.

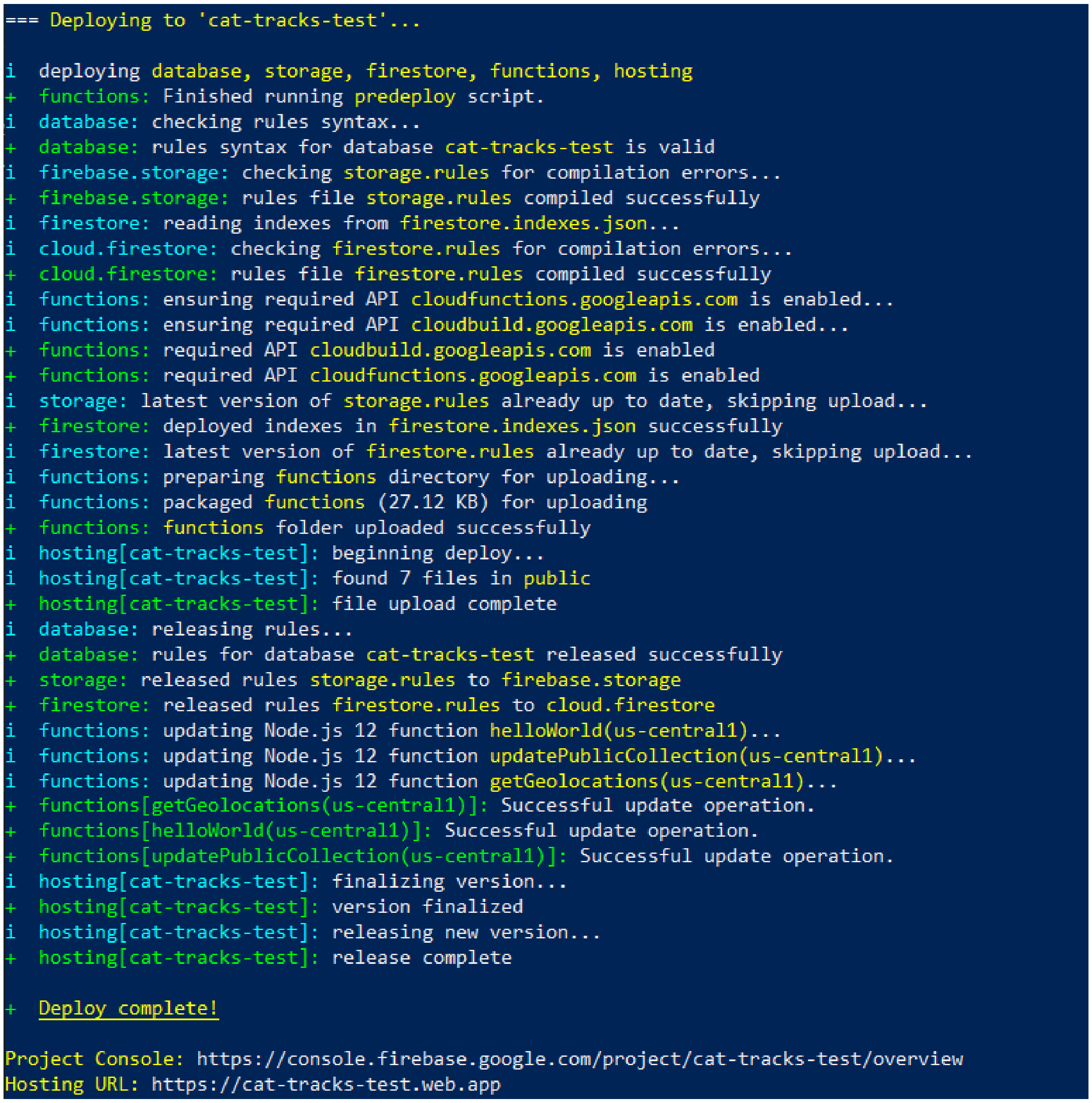
In the end, the reason why Firebase was chosen to be the backend of Cat Tracks was because of its ease of use and convenience. When building a data collection and management system, it would be more efficient to have all the tools and functions contained in a single service, so that it is easier to manage and maintain the data that is coming through Cat Tracks itself.

Firebase Hosting

Firebase provides their own fast and secure hosting for developers of web applications. The ease of use of Firebase Hosting is that developers can deploy their web content from just the command line. It is made so that a single command is all that is needed to deploy the app to Firebase [6].

How it works is that first the Firebase CLI is installed on a developer’s computer. This is a command-line tool that allows developers to deploy and manage their web applications to Firebase through Firebase Hosting. Then when the developer has a directory that they want to deploy to Firebase for their web application, they go to the directory via command line and use the “firebase init” command to set up that Firebase directory to a Firebase project, given that a Firebase project already exists for the developer. From there Firebase will set up all the configurations for the Firebase project, such as the directory of what files will be served publicly and some of the files where the Firebase Security Rules can be read from to be used for Cloud Storage and the Database.

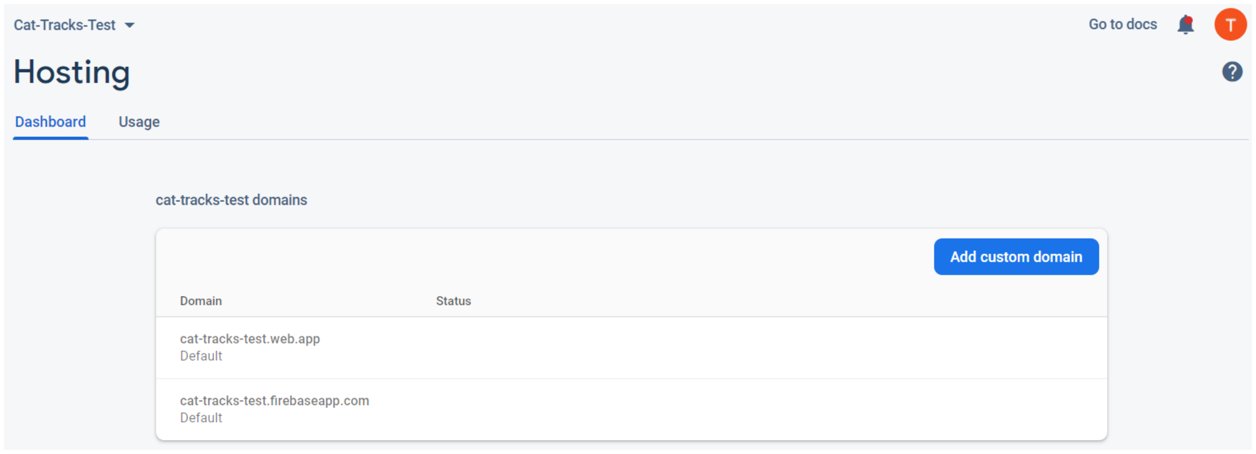
When the developer wants to see their web application in action, there are two actions the developer can take to see it. If the developer merely wants to see it running locally, they can run the “firebase serve” command, where the Firebase CLI will set the web app to run on a localhost. Then the user goes to the localhost to see the web app running through a web browser. This way the developer can test out their web app before having it go live on the net.



```
=== Deploying to 'cat-tracks-test'...

i  deploying database, storage, firestore, functions, hosting
+  functions: Finished running predeploy script.
i  database: checking rules syntax...
+  database: rules syntax for database cat-tracks-test is valid
i  firebase.storage: checking storage.rules for compilation errors...
+  firebase.storage: rules file storage.rules compiled successfully
i  firestore: reading indexes from firestore.indexes.json...
i  cloud.firestore: checking firestore.rules for compilation errors...
+  cloud.firestore: rules file firestore.rules compiled successfully
i  functions: ensuring required API cloudfunctions.googleapis.com is enabled...
i  functions: ensuring required API cloudbuild.googleapis.com is enabled...
+  functions: required API cloudbuild.googleapis.com is enabled
+  functions: required API cloudfunctions.googleapis.com is enabled
i  storage: latest version of storage.rules already up to date, skipping upload...
+  firestore: deployed indexes in firestore.indexes.json successfully
i  firestore: latest version of firestore.rules already up to date, skipping upload...
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (27.12 KB) for uploading
+  functions: functions folder uploaded successfully
i  hosting[cat-tracks-test]: beginning deploy...
i  hosting[cat-tracks-test]: found 7 files in public
+  hosting[cat-tracks-test]: file upload complete
i  database: releasing rules...
+  database: rules for database cat-tracks-test released successfully
+  storage: released rules storage.rules to firebase.storage
+  firestore: released rules firestore.rules to cloud.firestore
i  functions: updating Node.js 12 function helloWorld(us-central1)...
i  functions: updating Node.js 12 function updatePublicCollection(us-central1)...
i  functions: updating Node.js 12 function getGeolocations(us-central1)...
+  functions[getGeolocations(us-central1)]: Successful update operation.
+  functions[helloWorld(us-central1)]: Successful update operation.
+  functions[updatePublicCollection(us-central1)]: Successful update operation.
i  hosting[cat-tracks-test]: finalizing version...
+  hosting[cat-tracks-test]: version finalized
i  hosting[cat-tracks-test]: releasing new version...
+  hosting[cat-tracks-test]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/cat-tracks-test/overview
Hosting URL: https://cat-tracks-test.web.app
```

Figure 8. Firebase CLI with firebase deploy command

However, if the developer wants to have the web application hosted on the internet, then the “firebase deploy” command is used, as Figure 8 shows, where Firebase will host the directory.



cat-tracks-test domains

Domain	Status
cat-tracks-test.web.app Default	
cat-tracks-test.firebaseapp.com Default	

Figure 9. Firebase Hosting Dashboard showing domains for project

From the Firebase Hosting Dashboard, developers can see the website domains that their site is hosted on. By default, Firebase will provide a web.app and firebaseapp.com domain to every project that gets hosted on Firebase, which Figure 9 has an example of for Cat Tracks.

Each site hosted on Firebase is served with a secure connection, since Firebase will provide an SSL certificate to each site deployed. This will ensure that when users access the site, it will be done with a secured connection.

Flutter

The mobile application of Cat Tracks is built using the Flutter UI software development tool kit provided by Google [8]. It runs on the programming language, Dart [4], which is a programming language developed by Google to build multi-platform apps, and Flutter itself is developed to be able to run on both Android and iOS, without the need to have two separate code bases to develop on both mobile platforms. With the market share of mobile phones being predominantly Android and iOS, it would be efficient on the app developers to work with a tool kit and programming language that works natively on both Android and iOS, while also removing the specific limitations of native functionality [2]. With having a single codebase working on both Android and iOS, developers can save time on not needing to develop in two different languages to work with the two different mobile operating systems.

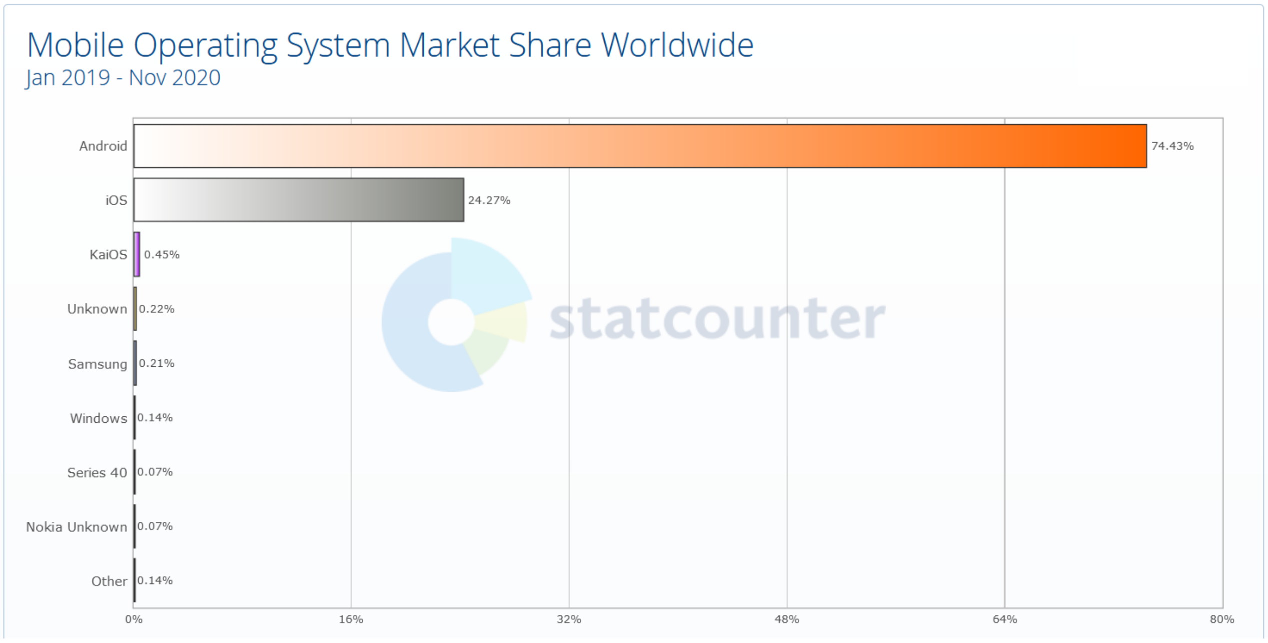


Figure 10. Mobile operating system market share worldwide. Credit: StatCounter [10]

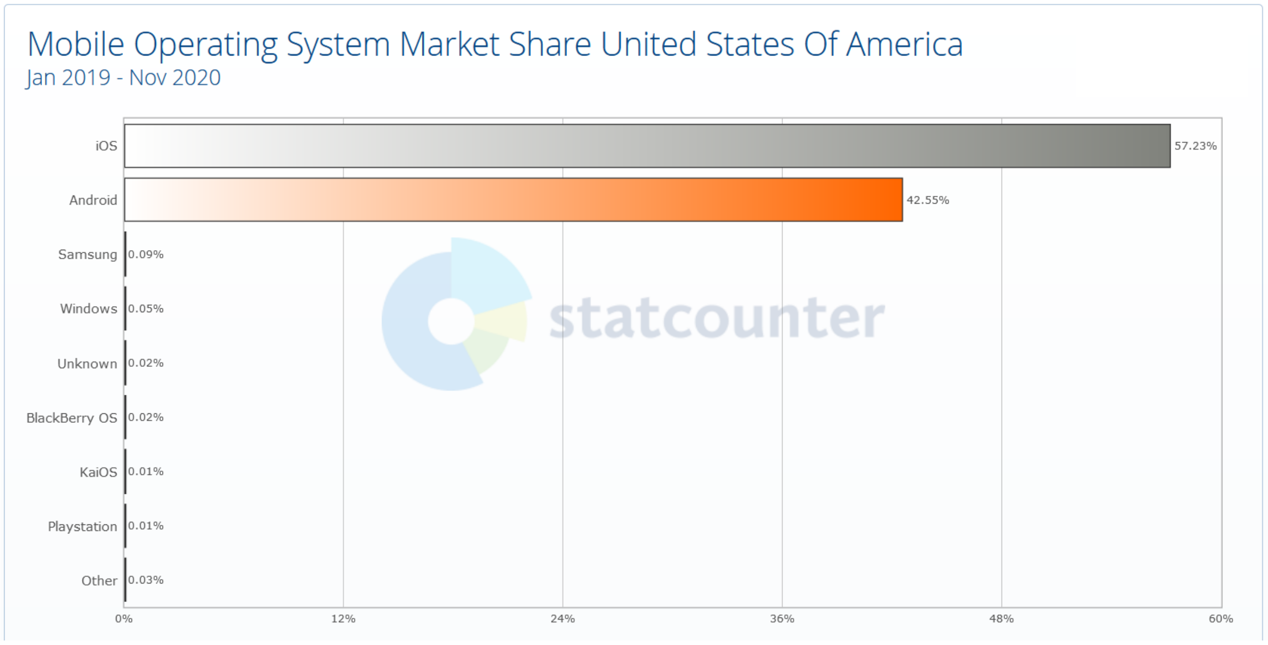


Figure 11. Mobile operating system market share in United States of America. Credit: StatCounter [10]

As shown in both Figures 10 and 11, while the worldwide market share may be predominately Android, the majority of the market share in the US is iOS, but Android is still a significant percentage of the market share [10]. Since the target focus of Cat Tracks is California, a state in the United States of America, it would make more sense to take account of both iOS and Android users, therefore creating a mobile app that works natively on both operating systems.

With both Flutter and Firebase being developed by Google, they are compatible with working together, with Flutter having its libraries make use of the Firebase platform [8]. The way Flutter works is that each part of the UI is considered a widget. On a screen, there may be several different widgets that each serve a different function. One key to working with Flutter is handling the states of widgets. In Flutter, a widget can be either Stateless or Stateful. The main difference is whether the state of the widget will change. For example, a Text widget that merely displays a message that does not change would be in a Stateless widget because the text itself is not changing. Whereas a Google Map widget that displays a map on the screen would be a Stateful widget, since the map would always be changing, such as when new markers are placed on the map.

According to Qadir et al. [13], Flutter is very secure. They noted that Flutter's security vulnerabilities have not been fully discovered which means that security exploits have not been found in Flutter yet. Since Flutter receives regular updates, any possible security holes also get fixed in newer updates, so keeping Flutter updated to the latest version is an effective measure of making sure that the application is secured. Another way to ensure security with Flutter is the ease of being able to update dependencies. A developer merely needs to make sure that the dependencies are updated regularly to the latest version as the latest version will also have updated security fixes. A range of versions for dependencies can also be given to make sure that the dependencies are updated to the latest versions.

Related Works

There has been on-going research and development regarding both mobile applications and tracking, along with using Firebase as part of the infrastructure for their application. In [11] Mutiawani et al. proposed an Anti-Theft Vehicle Tracking System built on Android that used the Firebase database to track the location of the vehicle and the owner of the vehicle. Their application measured the distance between the user and their vehicle while being able to monitor it through Google Maps. In [9] Li et al. designed an Internet of Things system called JustIoT, which used Firebase as their backend database, such as for roles with different kinds of users, and subsequently the user authentication that Firebase provides. Darweesh et al. [5] presented how the distributed capabilities of Firebase on mobile nodes can lead to better image processing performance. They showed how Firebase makes communication easier for a mobile device using master and slave nodes. In [14] Rajappa et al. developed an Android application using Firebase for user authentication and its database to store data obtained from pinging beacons. In [2], Boukhary et al. proposed a Flutter Clean Architecture to deal with some of the state management issues that Flutter has. Where an issue they saw with Flutter was how the widget tree of Flutter's UI could cause performance problems when a change in one widget could cause the rest of the widgets in the children or nested widgets to render as well. Qadir et al. [13] developed a cargo tracking system using Flutter that used Firebase as their backend server. They emphasized the importance of cross-platform development, that developing with a single codebase can reduce cost and time of development.

III. Design

The whole idea behind Cat Tracks is that it is a data collection and management system done through crowdsourcing. It is built through the Firebase platform from Google, which serves as an infrastructure to maintain the data provided by the users. Firebase provides its own database structure, called Firestore, which is used to store and track the metadata of images that are uploaded. Such metadata contains information on the images such as the geolocation of where the image was taken, and user-provided tags, both of which can be used for querying the data for insight of what kind of activities are coming from the users. Firebase also provides its own Cloud Storage for where user images are stored in the Cloud. These images can be used to appear on a map of where the images are taken to give a visualization of where the mountain lions may be roaming in the area.

For the simplicity of user authentication, it was decided to use Google login as a federated login for Cat Tracks. This way, there is no need to develop and maintain user authentication, as it can be provided by Google itself given that the user has their own Google account. Otherwise, it would be required for the user to create a Google account to use the Cat Tracks application.

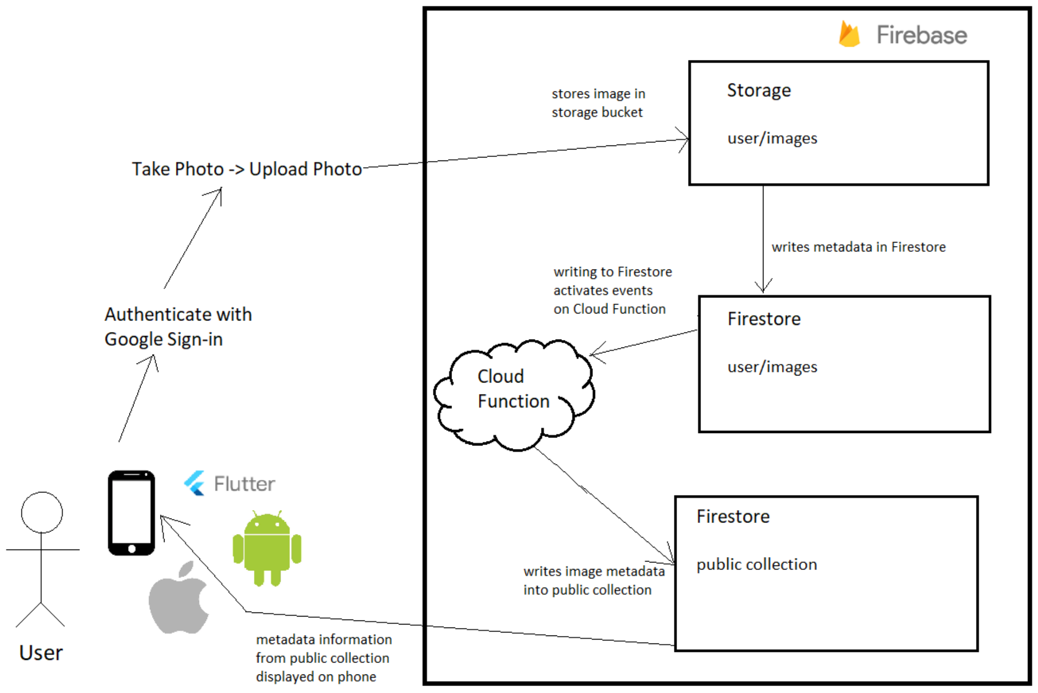
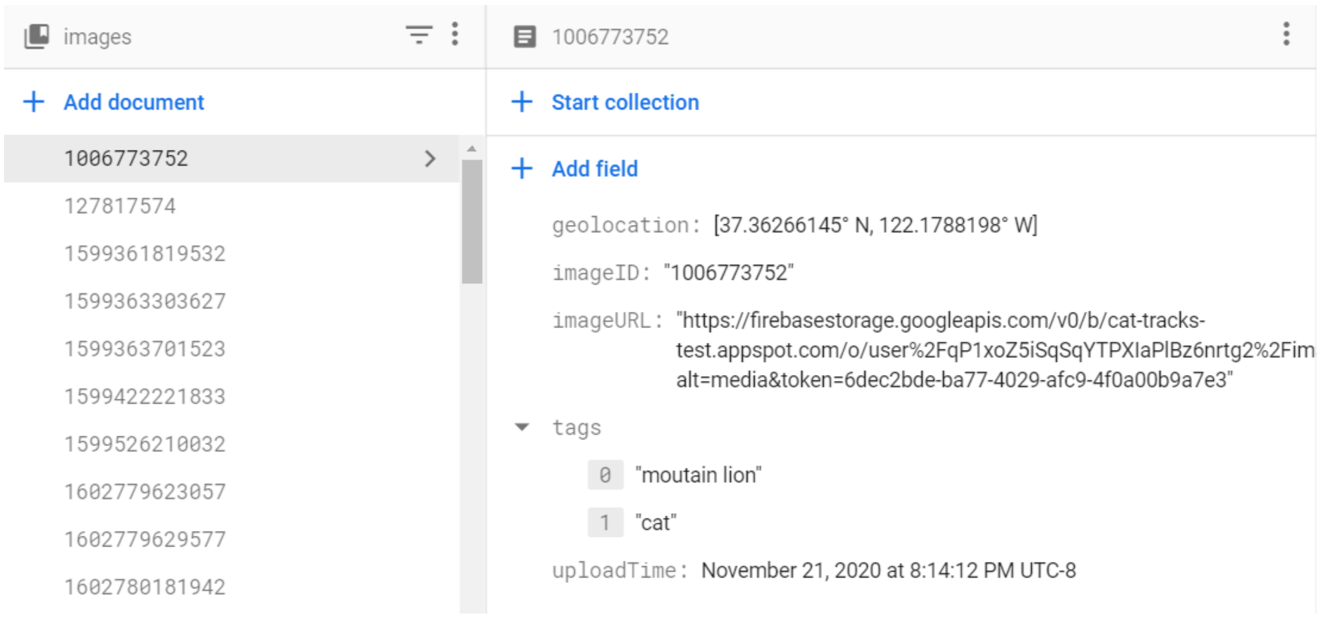

Firebase
Storage
user/images
stores image in storage bucket
Take Photo -> Upload Photo
writes metadata in Firestore
writing to Firestore activates events on Cloud Function
Firestore
user/images
Authenticate with Google Sign-in
Cloud Function
Flutter
Firestore
public collection
writes image metadata into public collection
User
metadata information from public collection displayed on phone

Figure 12. Overall architecture design of Cat Tracks

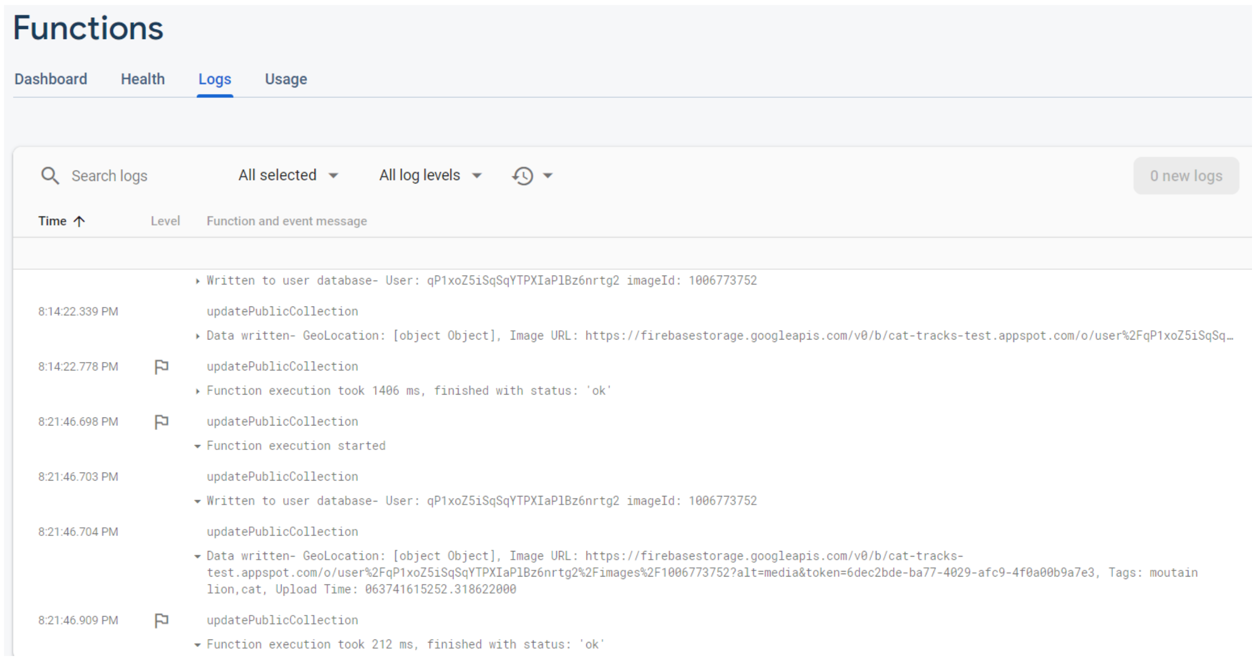
Cat Tracks itself is a mobile application, which the overall design and procedure is shown in Figure 12. It works with a user logging into the application through their Google account. Then they take a picture of the mountain lion. Afterward, they input the tags they feel are appropriate for the image they have taken and then upload the image. Where it will be processed through the Google Cloud Service and stay in the image repository.



```
1006773752
127817574
1599361819532
1599363303627
1599363701523
1599422221833
1599526210032
1602779623057
1602779629577
1602780181942
```

```
1006773752
geolocation: [37.36266145° N, 122.1788198° W]
imageID: "1006773752"
imageURL: "https://firebasestorage.googleapis.com/v0/b/cat-tracks-
          test.appspot.com/o/user%2FqP1xoZ5iSqSqYTPXIaPlBz6nrtg2%2Fim
          alt=media&token=6dec2bde-ba77-4029-afc9-4f0a00b9a7e3"
tags
    0 "moutain lion"
    1 "cat"
uploadTime: November 21, 2020 at 8:14:12 PM UTC-8
```

Figure 13. Metadata of a sample image in Firestore



```
Time ↑          Level   Function and event message

                        ▸ Written to user database- User: qP1xoZ5iSqSqYTPXIaPlBz6nrtg2 imageId: 1006773752
8:14:22.339 PM            updatePublicCollection
                        ▸ Data written- GeoLocation: [object Object], Image URL: https://firebasestorage.googleapis.com/v0/b/cat-tracks-test.appspot.com/o/user%2FqP1xoZ5iSqSq…
8:14:22.778 PM            updatePublicCollection
                        ▸ Function execution took 1406 ms, finished with status: 'ok'
8:21:46.698 PM            updatePublicCollection
                        ▾ Function execution started
8:21:46.703 PM            updatePublicCollection
                        ▾ Written to user database- User: qP1xoZ5iSqSqYTPXIaPlBz6nrtg2 imageId: 1006773752
8:21:46.704 PM            updatePublicCollection
                        ▾ Data written- GeoLocation: [object Object], Image URL: https://firebasestorage.googleapis.com/v0/b/cat-tracks-
                          test.appspot.com/o/user%2FqP1xoZ5iSqSqYTPXIaPlBz6nrtg2%2Fimages%2F1006773752?alt=media&token=6dec2bde-ba77-4029-afc9-4f0a00b9a7e3, Tags: moutain
                          lion,cat, Upload Time: 063741615252.318622000
8:21:46.909 PM            updatePublicCollection
                        ▾ Function execution took 212 ms, finished with status: 'ok'
```

Figure 14. Log of cloud function activity

During the process of uploading the image, the metadata of the images will be written to Firestore, which can be used for querying the data, as seen in Figure 13. After that, a Cloud Function will be triggered to write into a public collection that can be used to query the data for the public to see, as displayed on the log activity in Figure 14. This may entail showing a map of the images that have been collected so far that users can see where the mountain lions have been spotted.

Security

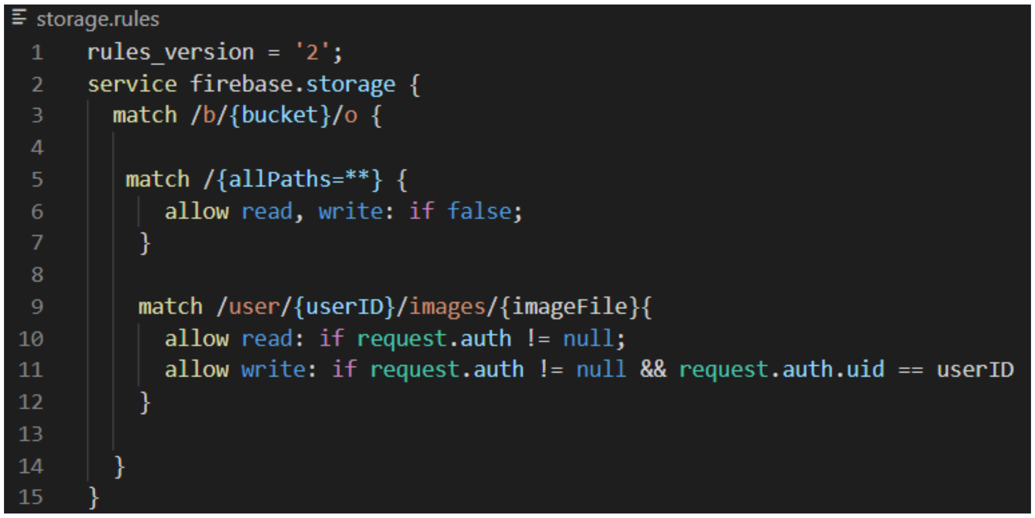

storage.rules
```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {

   match /{allPaths=**} {
      allow read, write: if false;
    }

     match /user/{userID}/images/{imageFile}{
      allow read: if request.auth != null;
      allow write: if request.auth != null && request.auth.uid == userID
    }

  }
}
```

Figure 15. Rules used for storage in Cat Tracks

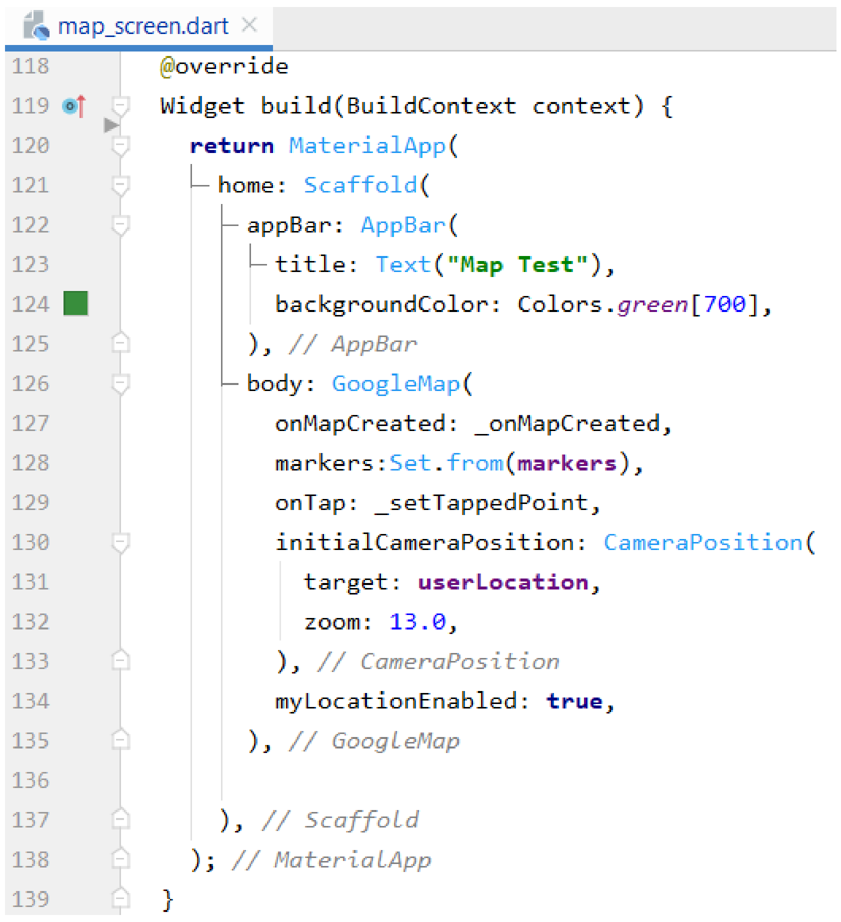
Security is handled by Firebase's own security system called "Firebase Security Rules" for read and write access on the data, as shown in Figure 15. It has been set up to grant both read and write permissions to logged-in users, so they may see their own data. The Rules are applied to Firestore and Storage separately. This way access to either of them are not dependent on the rule settings of the other. Rules are also set up to prevent unauthorized access. It ensures that only authorized users may access their own data for both Storage and Firestore, and certain roles can access other data. For instance, someone with an admin role may have access to any of the data stored in the database. All in all, Firebase Security Rules are an extensible and granular way to secure an app's data. Depending on the case, the rules can be simple or complex based on the sensitivity of the data carried by the app itself.

Along with the Rules in place for Firebase Security, there is also a need to enforce that only valid data is uploaded to the server. There may be cases of malicious users trying to get access to data they do not have access to or try to upload "junk" or inappropriate images to the server, which can then lead to issues when users use the application and see illegitimate images or data appear on the map. One such way to circumvent these malicious users is to make it so only “trusted” users have their data shown to other users. When users have been verified that the images they have uploaded are valid they may be marked as "trusted" users and the images they have uploaded to the server can be shown to any other user. A user marked as “trusted” will simply be a role assigned to them from Firebase itself.

One consideration that had to be accounted for during development was rate limits. Firebase is set up to charge based on the number of reads that are done on the application. Therefore, it is important to set a limit to how much a user could upload for some time. This was especially needed for the possibilities of bots using the applications and making rapid read or writes through the application, which would have driven up the cost charged by Google.

Design with Flutter

The mobile app of Cat Tracks was designed using Flutter. By using Flutter, each component on the screen is made up of a widget to control an action of the application. With Cat Tracks, a widget may be a button to log the user in or out of the application, or it could be another button used to open the camera on the phone to take a picture, and then another widget may be a text box to add tags for a taken photo. A widget of a map is also used to locate where different mountain lions have been found in the area.



```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: Text("Map Test"),
        backgroundColor: Colors.green[700],
      ), // AppBar
      body: GoogleMap(
        onMapCreated: _onMapCreated,
        markers:Set.from(markers),
        onTap: _setTappedPoint,
        initialCameraPosition: CameraPosition(
          target: userLocation,
          zoom: 13.0,
        ), // CameraPosition
        myLocationEnabled: true,
      ), // GoogleMap

    ), // Scaffold
  ); // MaterialApp
}
```

Figure 16. Dart code of Flutter with Google Map widget

With the use of Flutter, the entire app is written in the Dart programming language [4], which makes it so the Cat Tracks mobile app can run on both iOS and Android without the need of two different code bases for both platforms. This can be shown in Figure 16, with a GoogleMap widget written in Dart. Any slight differentiation between the Operating Systems can be handled in an individual folder for iOS and Android. One convenience with Flutter is that all the dependencies can be obtained just by including the dependency in a pubspec.yaml file, in which a command with Flutter can be used to get those dependencies. Then a file only needs to import the dependency to be able to use it.

Metadata

To keep track of the data for the images uploaded to the server, there are metadata information taken from each image as it is being uploaded. In Firestore, the following metadata are kept track of for each user in a User collection:

- geolocation: Which are the coordinates of the device when the picture was taken
- imageID: The ID provided to the image on upload, which is a hash based on the time the image was uploaded
- imageURL: The URL of the image itself after it is uploaded, which can be retrieved throughout the system of Cat Tracks
- tags: An array of user provided tags to describe the image
- uploadTime: The time of when the image was uploaded to the system

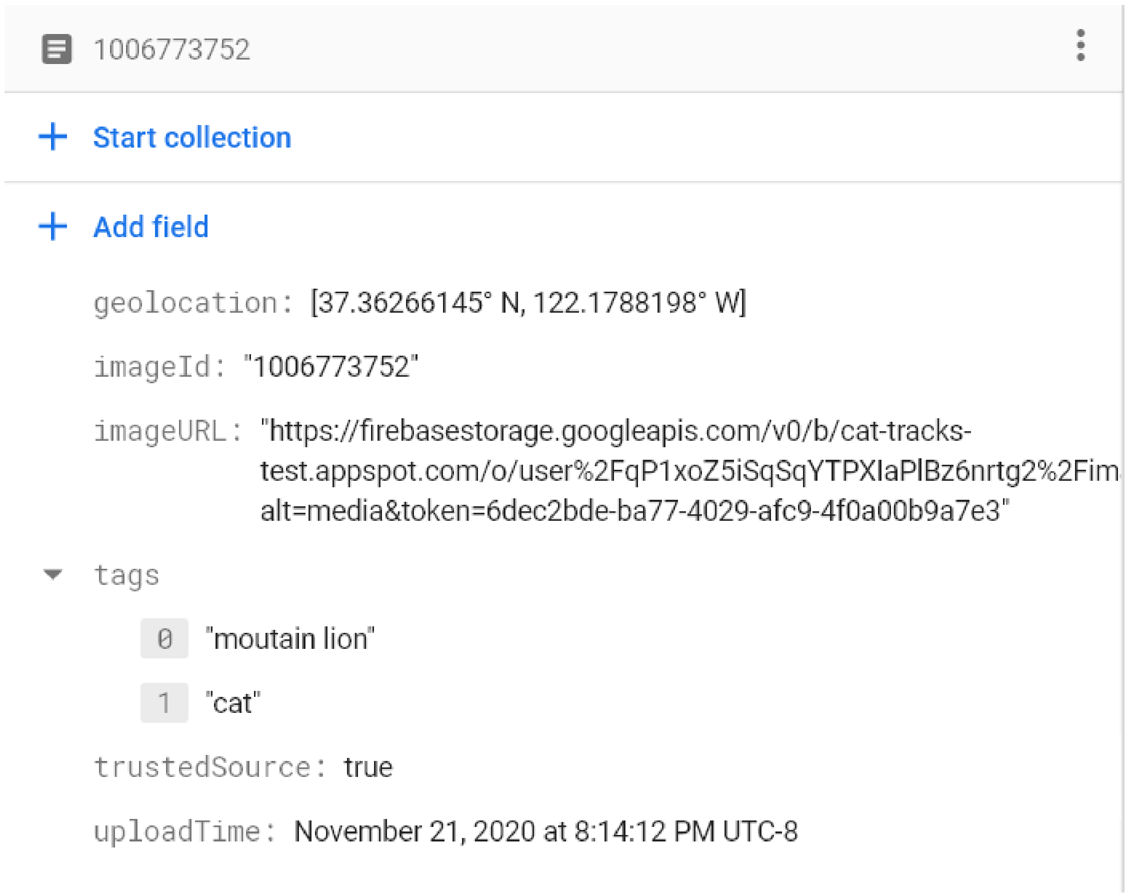
In the Public Collection, the metadata for each image is the same as in the except that there is one extra metadata kept track of:

- trustedSource: A Boolean that that marks whether the image was uploaded by a user with a “trusted” role

Keeping track of the metadata is an integral part of Cat Tracks being a system for data collection and management. With the data collected, a query can be made based on the metadata to observe the kind of activity that has been going throughout the system. This will be important as researchers can view where mountain lions have been found and around the time they have been spotted. If researchers notice a cluster of mountain lions spotted in an area nearby traffic, they can take action to reduce the likelihood of a mountain lion having a collision with a vehicle.

The metadata also has uses throughout Cat Tracks itself. The imageID can let the system know what image is being processed. This is also to identify each unique image that gets uploaded to Cat Tracks. The imageURL is the location of the image itself on the internet. Anytime a file gets uploaded to Firebase a public download URL is given to that file. However, by design from Firebase these files are freely available for the purpose of file sharing to the public. It is up to the developer to be able to revoke the download URLs to prevent the files from being out in the public. This is only for public access to files. For private access to the files, the Firebase Security Rules would still have to be obeyed for access to the files.

For the tags metadata, these are the tags users are providing when they upload an image. In Firestore, the tags are stored as an array of strings. What these tags can be used for is that when searching for a particular type of wildlife, there can be a search by tags that will show all the images containing a particular tag. When it comes to statistical purposes, a look up on how many pictures have a tag associated with them may also be used with the tags. The uploadTime shows the time that the photo was uploaded. The purpose of this is to show the users when a photo was uploaded to Cat Tracks. This can be used to show how recent an animal was spotted when the image is taken and uploaded. In the case of mountain lions, the data for upload time can give time frames of when people spotted mountain lions around the area. If there were a sudden increase of mountain lions appearing within a particular time frame, wildlife researchers could investigate the kind of activities that went on and if the mountain lions were spotted in areas with high traffic.


1006773752

```
geolocation: [37.36266145° N, 122.1788198° W]
imageId: "1006773752"
imageURL: "https://firebasestorage.googleapis.com/v0/b/cat-tracks-
          test.appspot.com/o/user%2FqP1xoZ5iSqSqYTPXIaPlBz6nrtg2%2Fim
          alt=media&token=6dec2bde-ba77-4029-afc9-4f0a00b9a7e3"
tags
    0  "moutain lion"
    1  "cat"
trustedSource: true
uploadTime: November 21, 2020 at 8:14:12 PM UTC-8
```

Figure 17. Metadata of an image in Firestore public collection

On the public collections, which is the collection available for the public to see such as on the map, there is an additional field as can be seen in Figure 17, which is a Boolean for trustedSource. What this is supposed to signal is that the uploader of this image has the role of “trusted”. How this can be used is that users can set a toggle to only view images that have the same from a trusted source or they could see all images that have been on the public collection regardless of whether or not it came from a trusted source. This way there is a significantly less chance that the images shown came from malicious users if the user chooses to only view images that have been uploaded by trusted users.

IV. Implementation

Cat Tracks is developed as a mobile app. While it can work on iOS, it will be shown as an Android app in the screenshots below.

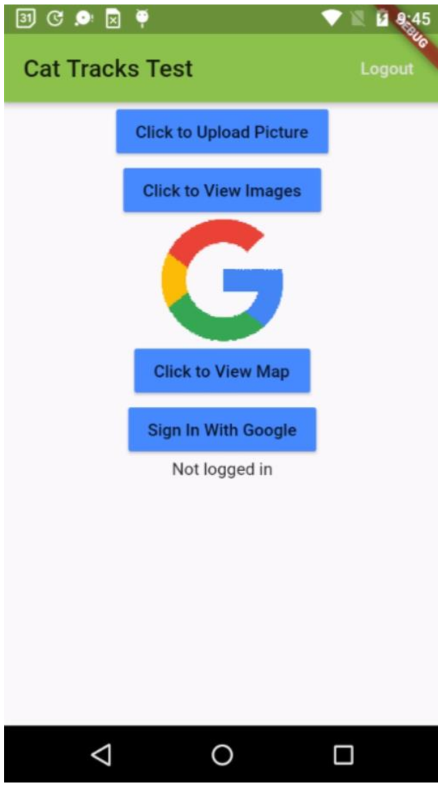
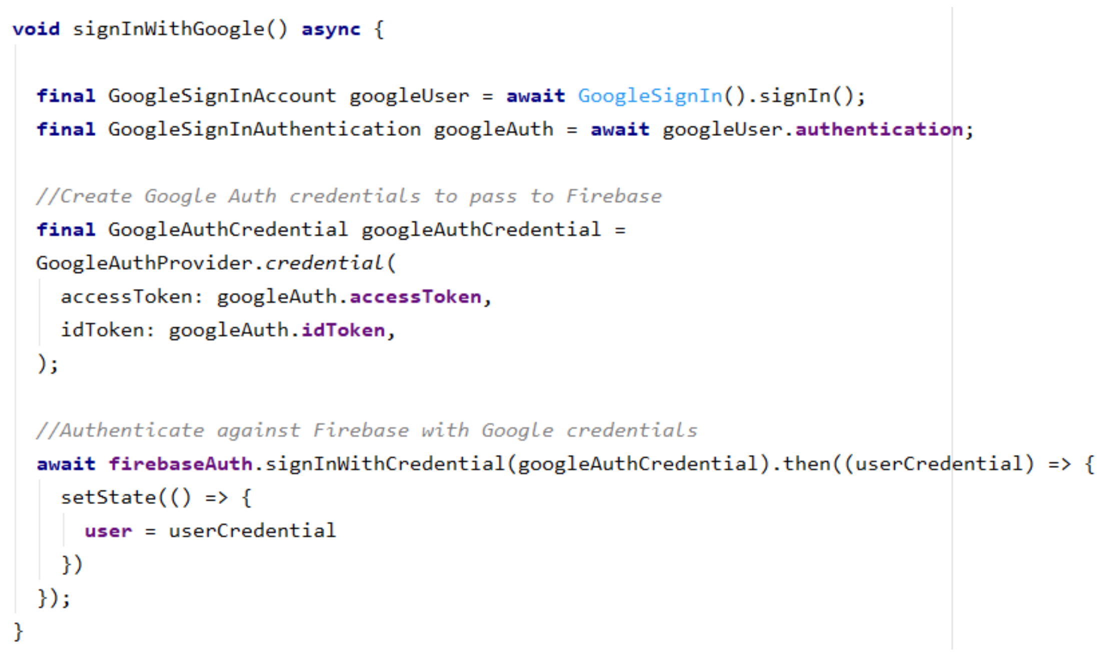


Figure 18. Cat Tracks home screen

When launching the app, users will be presented with a home screen as shown in Figure 18, where they will need to log in with their Google account by entering their own Google account credentials. For the ease of creating and managing accounts on Firebase, Cat Tracks currently only allows for Google logins during this stage of development. Users will not be able to take and upload or view any photos until they have logged in.



```
void signInWithGoogle() async {

  final GoogleSignInAccount googleUser = await GoogleSignIn().signIn();
  final GoogleSignInAuthentication googleAuth = await googleUser.authentication;

  //Create Google Auth credentials to pass to Firebase
  final GoogleAuthCredential googleAuthCredential =
  GoogleAuthProvider.credential(
    accessToken: googleAuth.accessToken,
    idToken: googleAuth.idToken,
  );

  //Authenticate against Firebase with Google credentials
  await firebaseAuth.signInWithCredential(googleAuthCredential).then((userCredential) => {
    setState(() => {
      user = userCredential
    })
  });
}
```

Figure 19. Method to sign-in with Google account credentials

When a user tries to login to their Google account, they will be taken to a screen to either select their Google account or to enter their credentials to login to their account. In the process, as shown in Figure 19, the mobile application will go through an asynchronous method to get the user authenticated with their Google account. When the user's account has been authenticated, the Firebase Auth will use their Google account and set the application's credentials to their Google account's credentials. This will then set the state as the user being authenticated and logged in, which the user will stay logged in until they either logout or close out of the app.

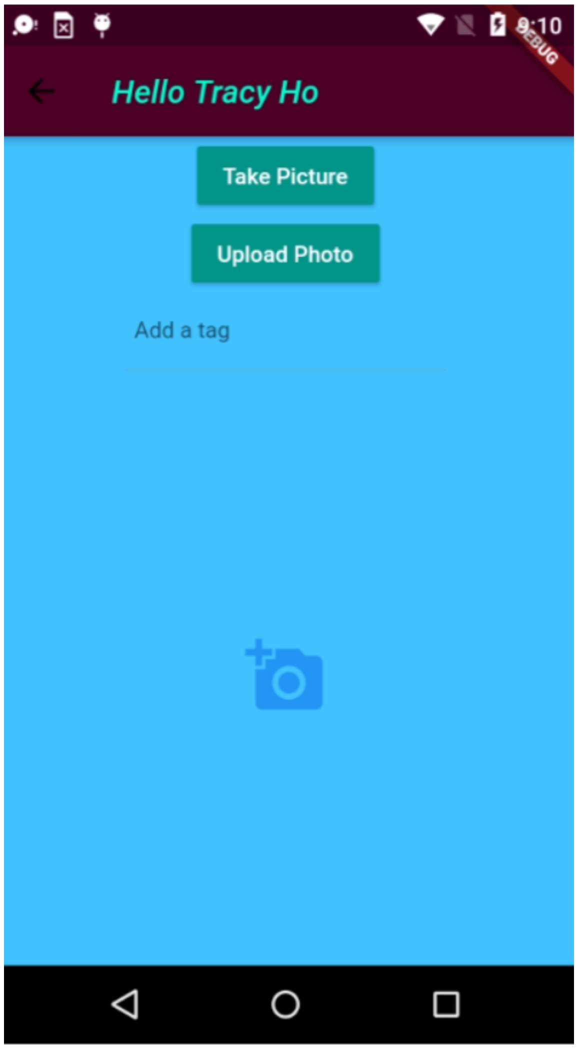


Figure 20. Screenshot of picture upload screen

After a user has logged in, they can access a screen, as shown in Figure 20, where they can click on a button to open the camera on their device to take a photograph of their immediate surroundings.

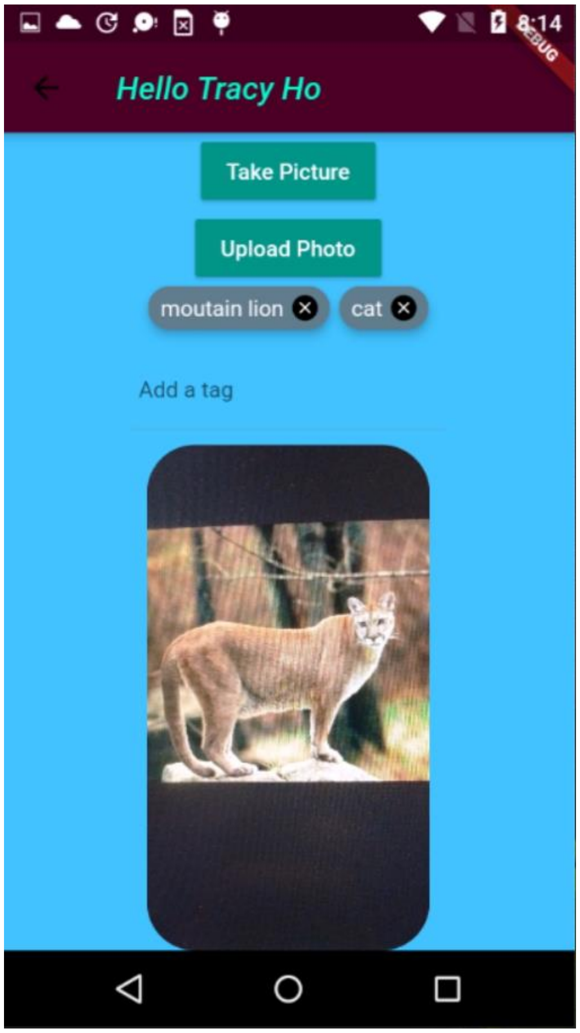
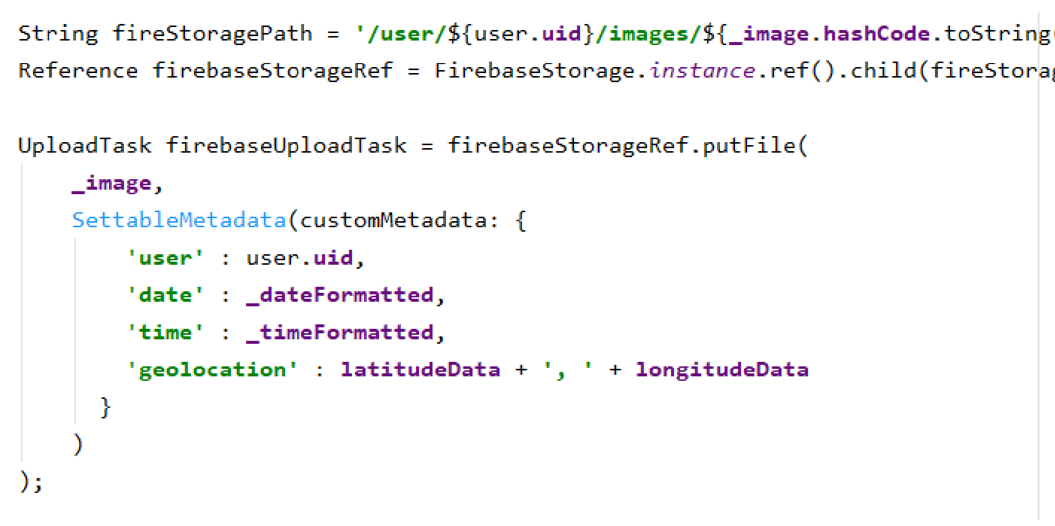


Figure 21. Image upload screen

Once the photo is taken and accepted, they will be brought back to the screen and a preview of the image they have taken will be shown on screen as seen in Figure 21. From there they can add any tags they feel is appropriate for the photo.



```
String fireStoragePath = '/user/${user.uid}/images/${_image.hashCode.toString()}';
Reference firebaseStorageRef = FirebaseStorage.instance.ref().child(fireStoragePath);

UploadTask firebaseUploadTask = firebaseStorageRef.putFile(
    _image,
    SettableMetadata(customMetadata: {
        'user' : user.uid,
        'date' : _dateFormatted,
        'time' : _timeFormatted,
        'geolocation' : latitudeData + ', ' + longitudeData
      }
    )
);
```

Figure 22. Firebase Cloud Storage path of uploaded images

Afterward, they can upload the photo to be stored inside a storage bucket. In Firebase, the Storage breakdown for files is through a file path of /user/{userID}/images/{imageID}, much like the storage path reference shown in Figure 22, and in these folders are the photos taken by each user.

These photos are used to track all the different mountain lions found roaming the area. This can also present possibilities about whether mountain lions have been found in locations near high traffic. The pictures used could also potentially identify the mountain lions that may have been in the area.

For each photo taken, there is some metadata information that gets taken to associate the photo with. Such information is the geolocation of where the device was when the photo is taken, which requires the user to enable and allow the Cat Tracks application to use location data. Other metadata information taken are the user ID, image ID, image URL, and user-inputted tags, and the time the photo was uploaded. While Firebase Storage does allow the addition of metadata to be given on uploaded files, it is recommended from the Firestore developers that the metadata be stored in a database to be tracked with the uploaded files rather than using the metadata attributes of a file uploaded to Storage [6].

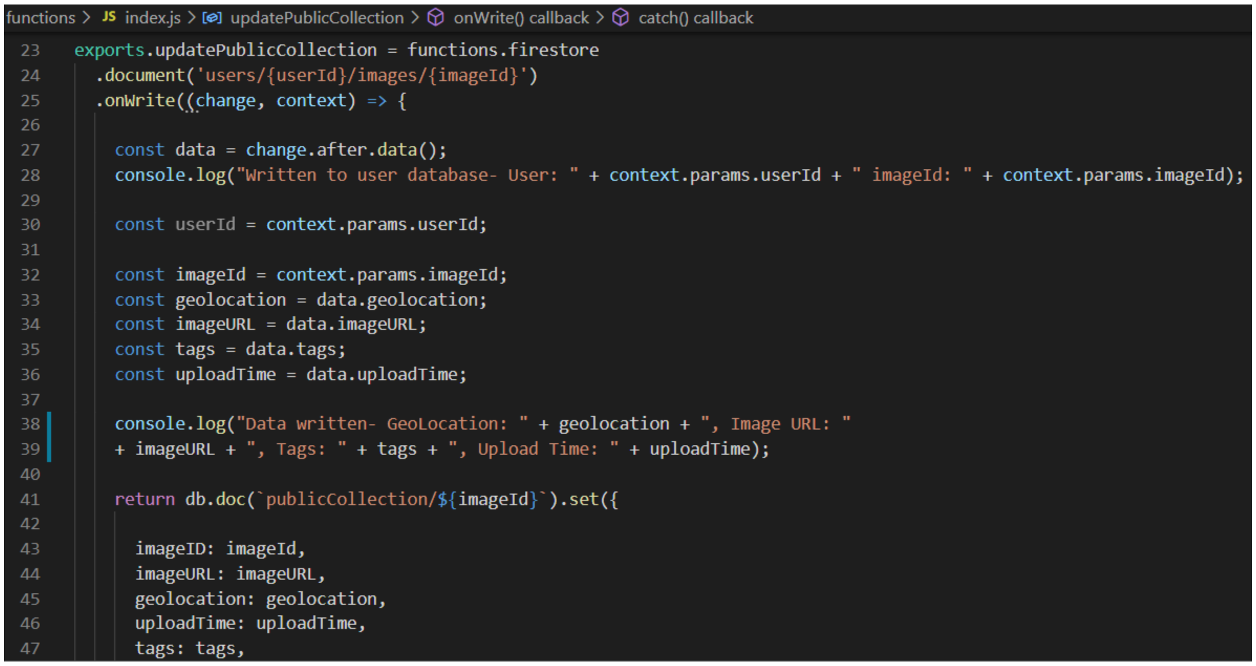
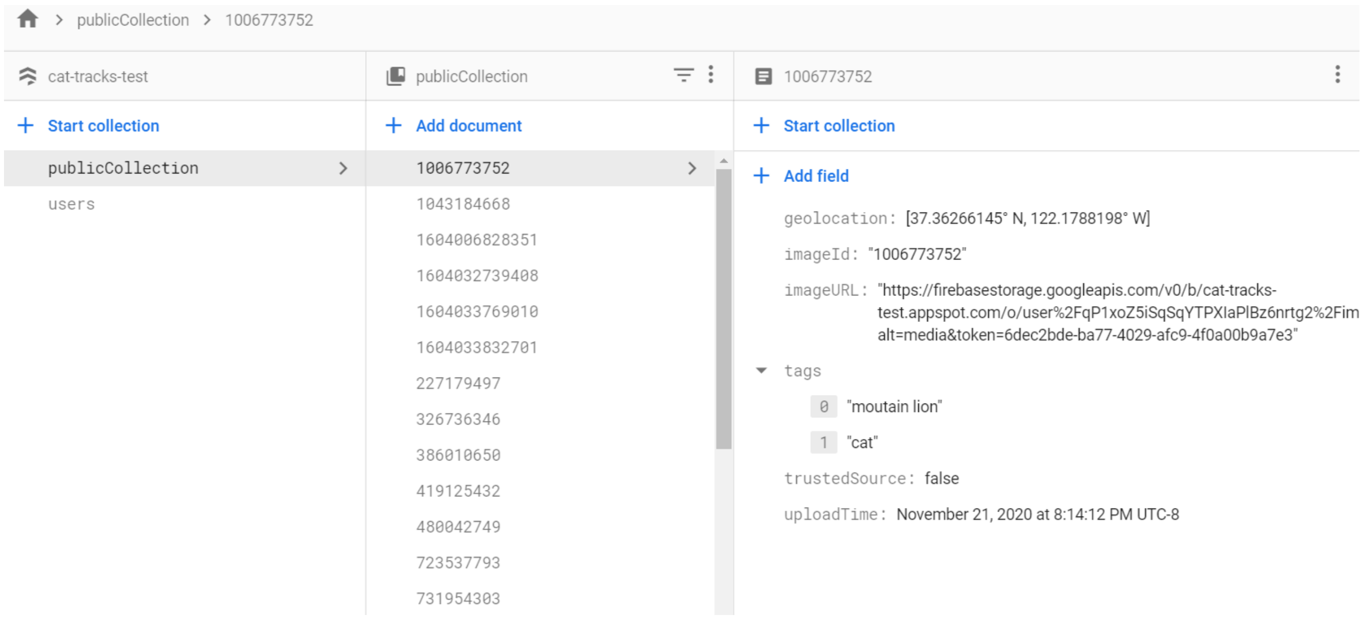

```
functions > JS index.js > updatePublicCollection > onWrite() callback > catch() callback
exports.updatePublicCollection = functions.firestore
  .document('users/{userId}/images/{imageId}')
  .onWrite((change, context) => {

    const data = change.after.data();
    console.log("Written to user database- User: " + context.params.userId + " imageId: " + context.params.imageId);

    const userId = context.params.userId;

    const imageId = context.params.imageId;
    const geolocation = data.geolocation;
    const imageURL = data.imageURL;
    const tags = data.tags;
    const uploadTime = data.uploadTime;

    console.log("Data written- GeoLocation: " + geolocation + ", Image URL: "
    + imageURL + ", Tags: " + tags + ", Upload Time: " + uploadTime);

    return db.doc(`publicCollection/${imageId}`).set({

      imageID: imageId,
      imageURL: imageURL,
      geolocation: geolocation,
      uploadTime: uploadTime,
      tags: tags,
```

Figure 23. JavaScript code of Cloud Function for writing in public collection



cat-tracks-test	publicCollection
publicCollection	1006773752
users	1043184668
	1604006828351
	1604032739408
	1604033769010
	1604033832701
	227179497
	326736346
	386010650
	419125432
	480042749
	723537793
	731954303

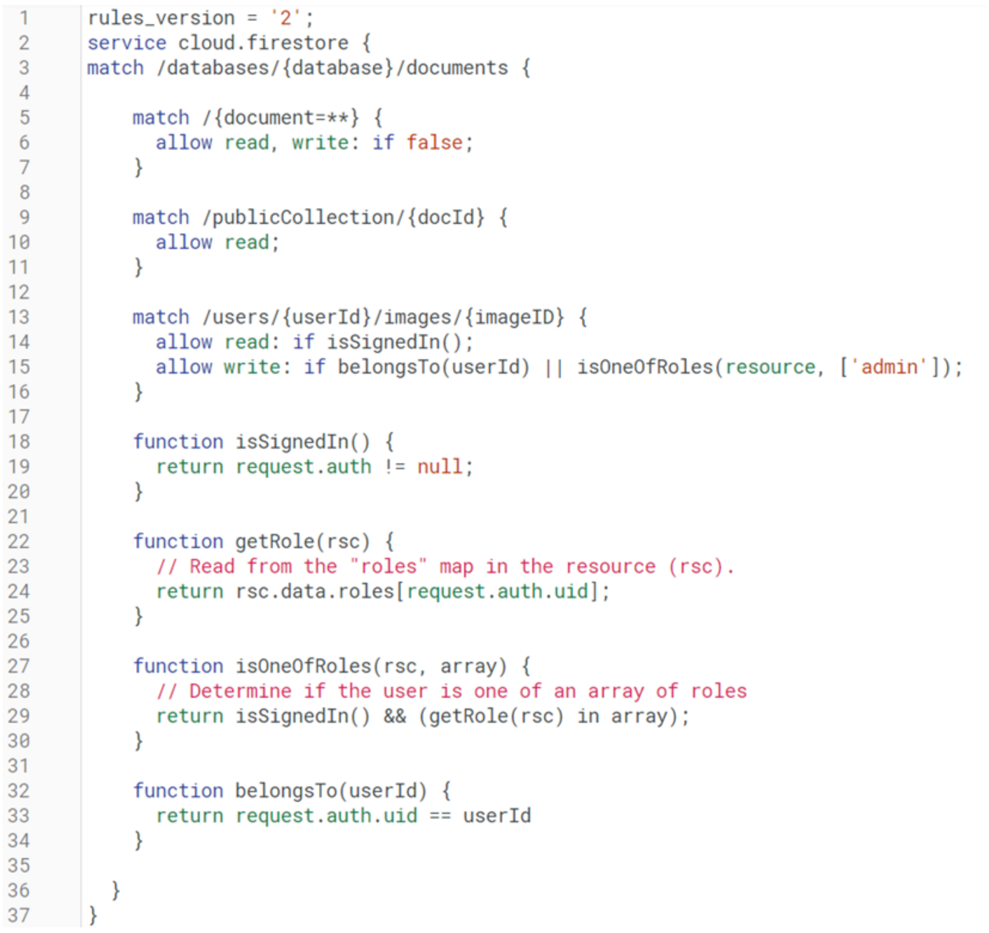
```
geolocation: [37.36266145° N, 122.1788198° W]
imageId: "1006773752"
imageURL: "https://firebasestorage.googleapis.com/v0/b/cat-tracks-test.appspot.com/o/user%2FqP1xoZ5iSqSqYTPXIaPlBz6nrtg2%2Fimalt=media&token=6dec2bde-ba77-4029-afc9-4f0a00b9a7e3"
tags
    0 "moutain lion"
    1 "cat"
trustedSource: false
uploadTime: November 21, 2020 at 8:14:12 PM UTC-8
```

Figure 24. Firestore sample of public collection

After uploading inside the bucket, information on the photo will also be written to a Firestore entry, and then after writing to a Firestore entry, a Cloud Function, as shown in Figure 23, will activate to write the same metadata from the photo into a public collection. The public collection, as shown in Figure 24, is created to be seen by anyone. The main purpose of this public collection is that everyone may be able to see the kind of activities going on with the Cat Tracks application and users can see where pictures have been taken and where mountain lions can be found roaming on a map. This public collection will include the image URL of the photo so that it can be referenced on a map to be viewed.

Firebase Security Rules

Both Firebase Storage and Firestore have a different set of rules for the Cat Tracks application.



```
rules_version = '2';
service cloud.firestore {
match /databases/{database}/documents {

    match /{document=**} {
      allow read, write: if false;
    }

    match /publicCollection/{docId} {
      allow read;
    }

    match /users/{userId}/images/{imageID} {
      allow read: if isSignedIn();
      allow write: if belongsTo(userId) || isOneOfRoles(resource, ['admin']);
    }

    function isSignedIn() {
      return request.auth != null;
    }

    function getRole(rsc) {
      // Read from the "roles" map in the resource (rsc).
      return rsc.data.roles[request.auth.uid];
    }

    function isOneOfRoles(rsc, array) {
      // Determine if the user is one of an array of roles
      return isSignedIn() && (getRole(rsc) in array);
    }

    function belongsTo(userId) {
      return request.auth.uid == userId
    }

  }
}
```

Figure 25. Firebase Security Rules on Firestore for Cat Tracks

In both cases, the very first rule set is making it so that on all paths, "allow read, write: if false;" as seen in Figure 25, this enforces a strict rule that no access is allowed on the data unless another criterion is fulfilled to grant the user access to see the data. This is a simple security measure to prevent any unauthorized access to the data.

When it comes to Storage, a simple check is used to see if a user is logged in to allow reads on a user's image files. Then for writes to the user's storage bucket, such as uploading photos, there is a check to see if the user is logged in and if the user making the upload is the same as its assigned UserID.

For Firestore, there are a different set of security rules used to grant read and write access to the data. First, the Public Collection is set up, so anyone has read access. Due to the nature of publicly available data, it is important not to have too many restrictions on the data that anyone is free to look at and retrieve. This way anyone can see what kind of activity has been going on with the Cat Tracks application and where users have spotted mountain lions roaming the area.

Now for individual user data, Firebase Security Rules have been set up so that only authorized users can access the data. This means that the only users that can see a specific user's data are the user themself when they log in or users that have been given the "admin" role.

Map

With the geolocation taken from the metadata of uploaded images stored in the public collection, the map screen will display the points that have been collected on the map. When the map screen starts up, the app will take the geolocation of all the images in the public collection and for each image, it will place a marker on the map for that image. This way users can see firsthand what kind of activities have been going on with tracking wildlife, and what kind of wildlife has been spotted. In this case, users can see where mountain lions have been found in the area. The map screen itself does not require a user to login to see the activity because it is pulling data from a public collection database and that data is available for anyone to see.

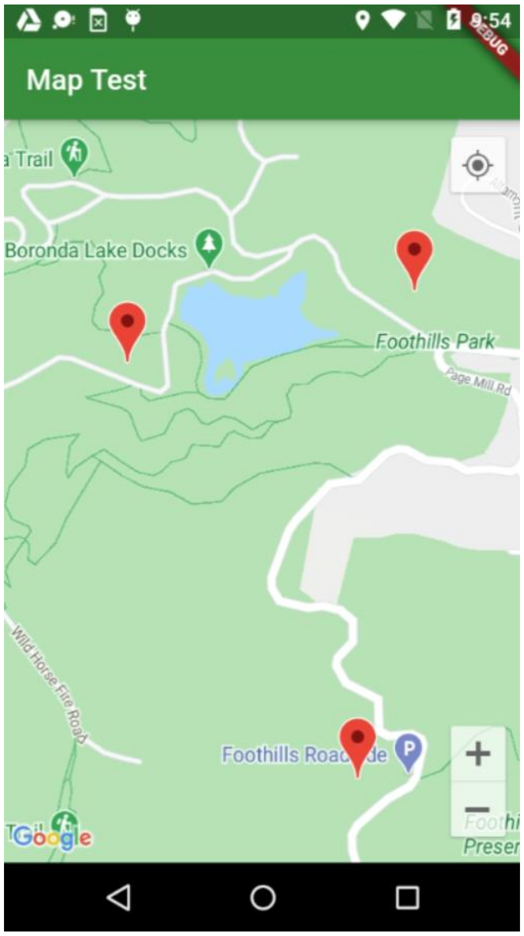


Figure 26. Map with markers to show the activity of where wildlife was spotted

In the current state, the Map Screen will place a marker for every image coordinate that is in the Firestore public collection as Figure 26 has shown. Users can scroll through the map and see where markers have been placed on the map. With Cat Tracks, the focus is on the state of California and mountain lions. From looking at the markers, users can see what other users are reporting to where they have found mountain lions roaming in the area. From the markers themselves, it can be seen if the reported locations are around any areas with traffic nearby.

Web Application

There is also a Web Application part of Cat Tracks. The web app portion works very similarly to the Map Screen of the mobile app. It takes from Firebase, the coordinates of all the images from Firestore and places markers of each of them on the map.

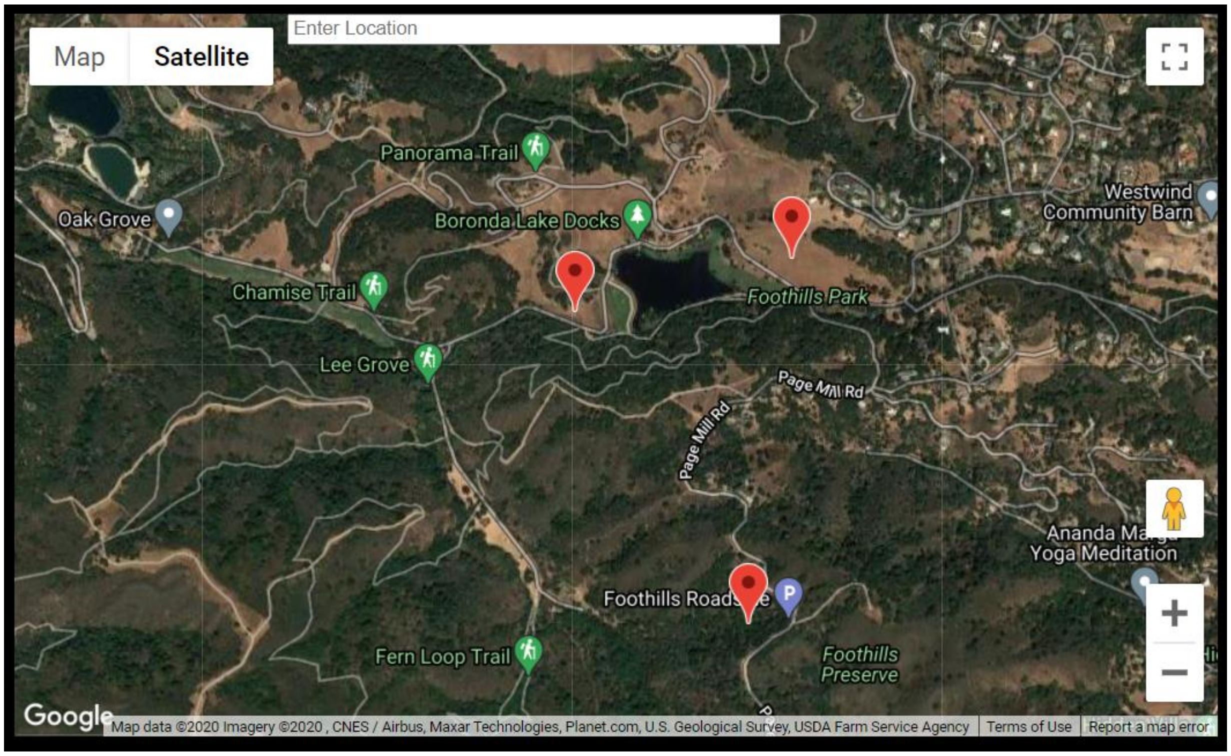
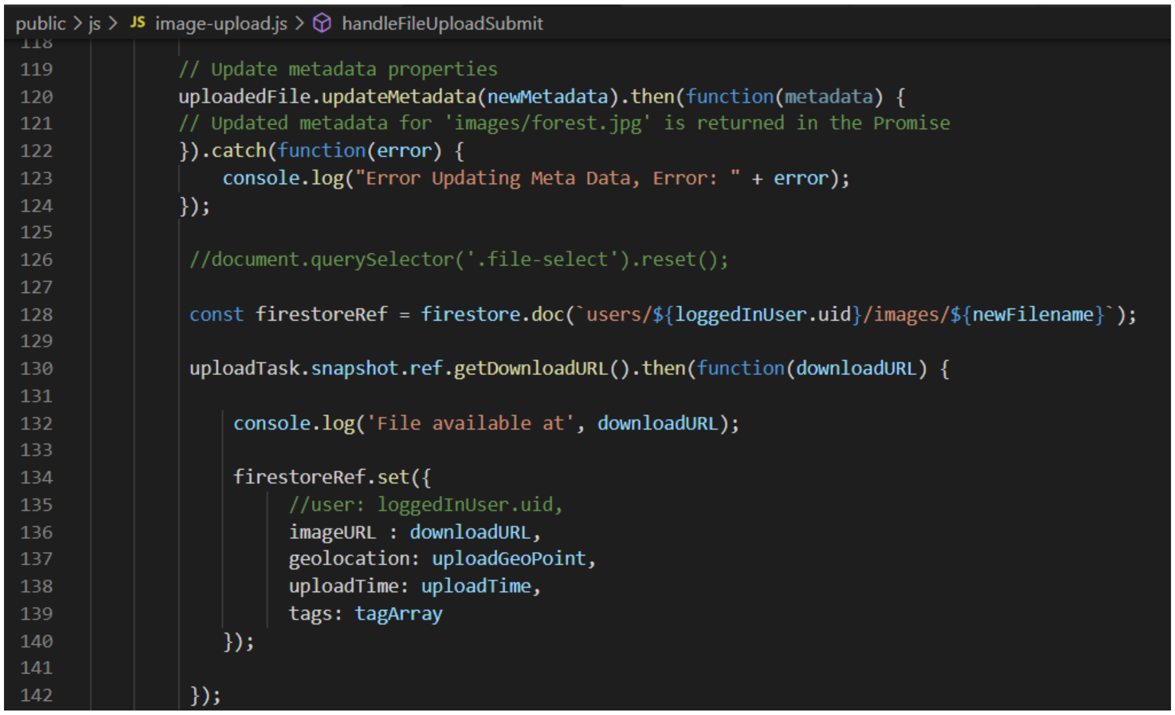


Figure 27. Google Map on web application with satellite

With the web application, users do not need to launch the mobile app on their mobile devices to view the activities that have been going on through the Cat Tracks system. They can go on the web application to see where wildlife has been spotted. For researchers, this will help them get a better view of what animals have been reported to be seen. For the state of California, a look at the map with a satellite view on a bigger screen, as an example displayed on Figure 27, can give a better overview to where mountain lions have been spotted in the area. If there have been multiple sightings of mountain lions near areas with traffic, then actions can be taken to prevent traffic collisions on mountain lions.


public > js > image-upload.js > handleFileUploadSubmit

```
            // Update metadata properties
            uploadedFile.updateMetadata(newMetadata).then(function(metadata) {
            // Updated metadata for 'images/forest.jpg' is returned in the Promise
            }).catch(function(error) {
                console.log("Error Updating Meta Data, Error: " + error);
            });

             //document.querySelector('.file-select').reset();

             const firestoreRef = firestore.doc(`users/${loggedInUser.uid}/images/${newFilename}`);

             uploadTask.snapshot.ref.getDownloadURL().then(function(downloadURL) {

                 console.log('File available at', downloadURL);

                 firestoreRef.set({
                     //user: loggedInUser.uid,
                     imageURL : downloadURL,
                     geolocation: uploadGeoPoint,
                     uploadTime: uploadTime,
                     tags: tagArray
                 });

             });
```

Figure 28. JavaScript snippet for Firestore functionality

During development of Cat Tracks, additional use of the Web Application was to test out the functionality of Firebase to ensure that the backend service was working as intended. Such functionality testing included logging in through Google sign-in to test Firebase Authentication, and then creating a web page to upload photos and JavaScript files to test out the functionalities of Cloud Storage and Firestore, along with getting geolocation data as part of the metadata on images for Firestore as shown in Figure 28. Testing these functionalities is what gave a better understanding for the same Firebase functions when developing the mobile app.

Image Viewer

On Cat Tracks, there is an Image Viewer Screen where users can go to see what images they have uploaded to Cat Tracks.

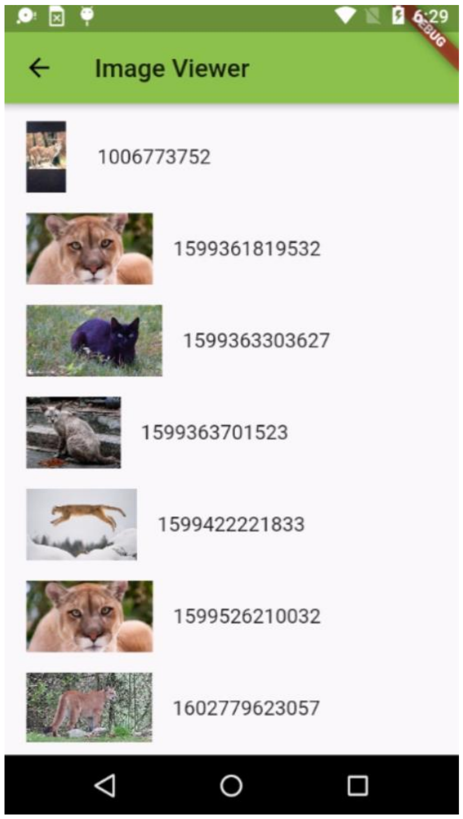


Figure 29. Image viewer screen

From the image uploading process, when a user uploaded an image to Cat Tracks, a download URL was generated and written to the metadata associated with that image on Firestore. In the Image Viewer Screen, the mobile application takes each of the download URLs that were written from the images that a user has uploaded and displays the images on a list through a List View. The List View, as shown on Figure 29, contains a small image of each image they have uploaded along with the image name that was generated through Cat Tracks from the upload. This way users can scroll through the screen to see what images they have personally uploaded to Cat Tracks.

The implementation of this screen was to get all the documents for a user's uploaded images and check which ones contained an imageURL field as part of its metadata. Then for each of these images display the image using the image URL on a row along with its image name. Making sure that an image document contained an image URL is a necessary step to ensure that an image does show up on a list.

V. Limitations During Development

During development of Cat Tracks, some limitations were encountered. It is worth mentioning that both Firebase and Flutter are relatively new technologies that are still undergoing development changes.

So, one difficulty encountered is how soon some of the libraries for both Firebase and Flutter became deprecated. Flutter itself is still a relatively new development kit, having its first stable release in December 2018. Thus, when major changes are put out from Flutter releases, some of the documentation for Flutter from a few years ago would be deprecated and obsolete in favor of the more recent releases. Sometimes help found from the past 2 years may not have been useful with the current time because those were using outdated library versions, now deprecated in the present time in favor of the current library, which may have completely different functions or even variables from what used merely 2 years ago.

When it comes to Firestore, queries are not very robust for querying subcollections. The reason why there is a Public Collection used for queries anyone can make on the data is that at the time of this writing, there is not a way to query multiple subcollections of the image metadata on Firestore for each user. Meaning that there was not a way to query multiple users' subcollection, which in the case of Cat Tracks, would be to get the geopoints for all the different images. Thus, this leads to data being replicated for the sake of the public being able to view activity going on throughout Cat Tracks. Another limitation also with Firestore is that it is not able to calculate a radius for the geolocation, meaning it is not able to populate a radius of all the different geopoints of where people have uploaded pictures. The way around this is to use a range for latitude and longitude, but this would be a robust way to include all the points within a radius that contain an image.

An additional limitation with Firebase was Cloud Storage. On the mobile app for Cat Tracks, there was no way to display all the images from a user's Storage. The images were displayed using the download

URL of the associated Firestore metadata for each of the user’s stored images. This has a chance of inconsistency problems because it assumes that when an image was uploaded to Storage, the download URL was also written to Firestore. However, if there were any issues between the image getting uploaded to Cloud Storage and writing the download URL to Firestore, there could be a chance where the download URL was not written to Firestore. Therefore, the image would not appear on the mobile app when retrieve all the user’s images. The Image Viewer screen on the mobile application relies on the image URL existing for each image uploaded to Firebase with how it takes from a user’s image collection and uses the imageURL field to display the image on the screen along with its associated metadata. If there happens to not be an entry for the image written to Firestore, the image will not show up on the Image Viewing screen.

However, on the web application for Firebase, it is possible to display all the images taken from the Storage that a user has. Thus, this is more of a limitation on the Firebase library for Flutter rather than Firebase itself.

VI. Future Work

Some future works can be done to Cat Tracks. One idea is to implement a “karma point” system where users can “like” an image to accumulate the amount of likes for a user. Once a user reaches a certain threshold for the number of likes, they can become a trusted user.

Another possible addition is adding more features to the web application aspect of Cat Tracks. The development of Cat Tracks was more focused on the mobile application, with the nature of it being an app that people would use on their phones to take pictures of wildlife and upload it to the net. So, development on a web application for Cat Tracks was limited to just showing a map and testing out Firebase functionality to make sure that the backend infrastructure of Firebase was working properly. Such as uploading simple pictures to make sure Cloud Storage and Firestore were working as intended and to see how Firebase works in general.

Cat Tracks has been built in such a way that additional features can be added to it. One possible addition is adding classification so when a user goes to upload a picture there can be tag suggestions to be included as part of the metadata based on what other users have tagged for similar-looking animals. Besides, in the future, if the technology for recognizing the same animal improves, it may even be used to track when an individual animal has been spotted and where it was last found. This would truly give a better sense of tracking wild mountain lions. Going along with this, another possibility is similar to what Ahn et al. [1] used, where they had participants manually classify images with the goal of having every image on the internet labeled. They made it as a game with two players trying to guess what the other player is labeling the image as, and the image itself has a count of how many times it has been labeled with a word. For Cat Tracks, there can be a screen using the public collection where users can manually label the image with words, they feel is appropriate with it.

Another future idea is the ability of importing or exporting data from other systems such as Zooniverse [16], which is a collaborative research web portal, where users participate in assisting the studies made by other researchers. Not only could there be an exchange of data with Zooniverse, but there may also even be the possibility of getting users to help with the labeling images uploaded to Cat Tracks.

Furthermore, when 5G becomes more commonly used throughout the world, there may be more advantages to enhance the performance of Cat Tracks by using the new technologies with the increase in cellular network speed. Since Cat Tracks is a platform that uses crowdsourcing, it can be extended to include many other new features.

VII. Conclusion

This paper proposed a crowdsourced data collection and management system called Cat Tracks, that is built on Firebase from Google. The purpose of this system was to create a way to track where mountain lions were found roaming in the area to reduce the amount of roadkill for mountain lions, which is considered the biggest killer of mountain lions in the state of California. This paper served as a case study of researching newer technologies that may enhance building a system for the purpose of crowdsourcing and collecting and managing data. This comes from both using Firebase as the backend service and using Flutter for developing the mobile app to work with Firebase. Firebase provided a platform-as-a-service used to collect and manage data obtained using the mobile application of Cat Tracks. The mobile app of Cat Tracks was built using Flutter from Google, which is a cross-platform UI toolkit, made so development of the app only needed one codebase. The emergence of cross-platform development is so developers can reduce time on developing apps by only having one codebase instead of multiple individual codebases for each different platform. So rather than having separate code for iOS and Android, it can be developed and run natively on both by using Flutter. As technology improves over time, there is hope that a system like Cat Tracks may contribute to the preservation of wildlife.

References

[1] L. Ahn and L. Dabbish. 2004. Labeling images with a computer game. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04). Association for Computing Machinery, New York, NY, USA, 319–326. DOI:https://doi-org.libaccess.sjlibrary.org/10.1145/985692.985733

[2] S. Boukhary and E. Colmenares, "A Clean Approach to Flutter Development through the Flutter Clean Architecture Package," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 1115-1120, doi: 10.1109/CSCI49370.2019.00211.

[3] “California’s Deadly Roads,” Mountain Lion Foundation. 2016-12-27. https://mountainlion.org/newsstory.php?news_id=1731

[4] Dart programming language. https://dart.dev/

[5] A. Darweesh, A. Abouelfarag and R. Kadry, "Real Time Adaptive Approach for Image Processing Using Mobile Nodes," 2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Barcelona, 2018, pp. 158-163, doi: 10.1109/W-FiCloud.2018.00031.

[6] Firebase. https://firebase.google.com/

[7] P. Fimrite. “California mountain lions dying on highways, but face an even bigger threat,” San Francisco Chronicle. 2019-9-20. https://www.sfchronicle.com/environment/article/California-mountain-lions-dying-on-highways-but-14454323.php

[8] Flutter. https://flutter.dev/

[9] W. Li, C. Yen, Y. Lin, S. Tung and S. Huang, "JustIoT Internet of Things based on the Firebase real-time database," 2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE), Hsinchu, 2018, pp. 43-47, doi: 10.1109/SMILE.2018.8353979.

[10] Mobile Operating System Market Share. StatCounter Global Stats https://gs.statcounter.com/os-market-share/mobile/

[11] V. Mutiawani, S. Rahmany and T. F. Abidin, "Anti-theft Vehicle Monitoring and Tracking Android Application Using Firebase as Web Service," 2018 International Conference on Electrical Engineering and Informatics (ICELTICs), Banda Aceh, 2018, pp. 72-77, doi: 10.1109/ICELTICS.2018.8548842.

[12] N. Ortego. “Animals are using these wildlife overpasses, and it’s saving lives.” Enjoy The Silence. 2017-9-11. https://enjoythesilence.today/2017/09/11/animals-are-using-these-wildlife-overpasses-and-its-saving-lives/

[13] A. M. Qadir and P. Cooper, "GPS-based Mobile Cross-platform Cargo Tracking System with Web-based Application," *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, Beirut, Lebanon, 2020, pp. 1-7, doi: 10.1109/ISDFS49300.2020.9116336.

[14] A. Rajappa, A. Upadhyay, A. S. Sabitha, A. Bansal, B. White and L. Cottrell, "Implementation of PingER on Android Mobile Devices Using Firebase," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 698-703, doi: 10.1109/Confluence47617.2020.9058306.

[15] “What is PaaS? – Platform as a Service.” Salesforce.com. https://www.salesforce.com/paas/overview/

[16] Zooniverse. https://www.zooniverse.org/