# Geolocated Media Sharing

Undergraduate Honours Thesis

Faculty of Science (Computing Science)

Ontario Tech University

Vital Golub

Supervisor: Dr. Christopher Collins

April 23rd, 2021

# Abstract

Geolocated Media Sharing is a mobile application that allows people to leave memories for themselves or others to discover. By photographing a location or an experience a memory is created. That memory can later be revisited at the location where the original memory was created. Memories can be shared between users of the application and allows for people to share their experiences with one another. In my thesis I discuss how I created this application explaining the design decisions when creating the application, the implementation details, and possible future works to expand upon this application. The thesis has a large focus on the use of geofence geolocation and use of different storage types to reduce latency and data usage.

# Table of Contents

1. Introduction ..... 5

1.1 Motivation ..... 5

1.2 Goals ..... 6

1.3 Overview ..... 6

2. Related Work ..... 6

2.1 Geocaching ..... 7

2.2 Replay AR ..... 8

3. Design Approach ..... 9

3.1 Storage ..... 10

3.2 Geofences ..... 11

3.3 Memories ..... 13

3.4 Notifications ..... 15

4. Implementation ..... 16

4.1 Storage ..... 16

4.2. Geolocation ..... 20

4.3. Memory creation ..... 23

4.4. Notification system ..... 24

5. Future Work ..... 25

6. Conclusion ..... 28

7. References ..... 29

# List of Figures

Figure 1: A user finding a geocache using geocaching application. (https://www.geocaching.com) ..... 7

Figure 2: Replay AR overlaying holiday theme over street image not during holiday. (https://replayar.com/) ..... 8

Figure 3: Geofence representation with the three main geofence events. ..... 11

Figure 4: Memory creation and caption addition screen. ..... 13

Figure 5: Memory grouping system displaying 3 possibilities, within previous, overlap and separate. ..... 14

Figure 5: Notification received after triggering a geofence. ..... 15

Figure 6: Cloud storage directory to store media images. ..... 18

Figure 7: Cloud storage representation of a Geofence ..... 19

Figure 8: Cloud storage representation of a memory ..... 19

Figure 10: Actively monitored geofences in a 2-kilometer radius all other geofences not actively monitored. ..... 22

Figure 11: Geofence displaying issue with notification location versus the location of the media. ..... 26

Figure 12: Possible solutions for more accurate notifications. ..... 27

# 1. Introduction

The objective of this thesis was to design an application that would allow people to leave time capsule like memories in different locations for themselves or others to discover at a later date. The desired effect is a serendipitous notification experience in which the user of the application is prompted with a memory that they have saved in a location around the world. Clicking on this memory notification would enlarge that memory on your screen to display the past memory. This allows people revisit existing memories as well as share memories with others.

## 1.1 Motivation

We often do not think about places we have been to, or experiences we have had, until we look back through our phones or photo albums in an attempt to remember. Using the Geolocated Memory Sharing application people will be able to experience the emotion associated with the memory they experienced in a more fulfilling way. Applications are starting to let users know when their friends have visited a location in the past by giving you a notification. By adding the ability to leave memories in locations, the experience and emotion that was occurring at the time can be revisited, while also still providing the original allure of knowing someone else has visited the same location as well. This application can be used for many things, such as important life events, tourist tips or even restaurant reviews. Leaving a memory of your experience at restaurant to share with others or sharing an interesting fact about a historical object, notifying people who visit those locations as well are just two examples of what is possible. The motivation behind this project was to create a more seamless medium for revisiting memories that have been left around the world.

## 1.2 Goals

The goal of this thesis is to create a mobile application that allows the user to create memories associated with a location by taking a picture. Those memory locations can then later be visited which will provide the user with a notification that will inform them of a memory in that location. The application should have an easy-to-use user interface that supports the ability to take pictures and add captions to images as well as a synchronization system that allows users to share memories with other users.

## 1.3 Overview

In section two, I will explore some related works to describe the inspirations for the thesis as well as look at current implementations of geolocated work. Section three will examine the design approach and design decisions of making the application. I will explain what the application does as well as the important decisions I made a long the way. Section four will describe the implementation of Geolocated Memory Sharing. This will include all the technical implementation as well as describing how I implemented my design plan. And lastly, section five will discuss future works that can be done to improve or modify the application.

# 2. Related Work

In this chapter I have divided the related work into two categories. Both categories focus on applications that have similar features as the Geolocated Media Sharing Application. The first is Geocaching [1] which is an activity that has a mobile application that focuses heavily on

mobile location tracking and geolocation. This application was a large inspiration for the current application. The second application is ReplayAR [3] and it allows people to take a photograph and convert it into an augmented reality experience. Once they have this experience, they can revisit the location that the photograph is from and overlay it on the current location to show a historical view of the location. This application was found during my research and has a similar goal of allowing users to relive memories from the past.

## 2.1 Geocaching

Geocaching [1] was one of the main inspirations for the project. Geocaching is an outdoor recreational activity in which participants use location tracking such as GPS and other navigation techniques to hide and seek containers. Geocaching supports mobile devices using their application, and it is available on Android, IOS, and windows phones. These containers are called geocaches and they are marked by coordinates all around the world [2].

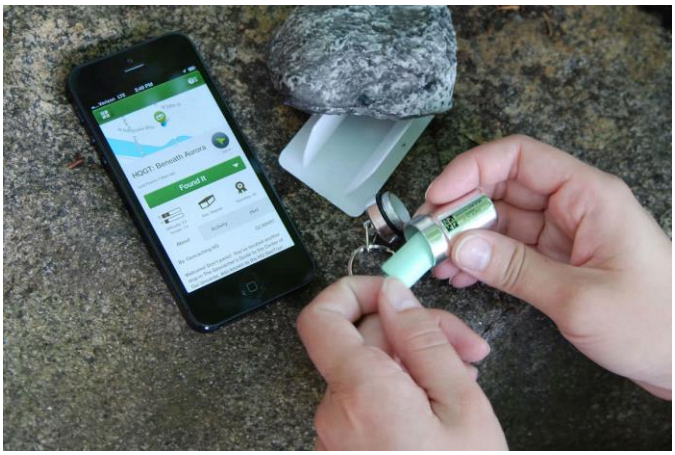

*Figure 1 A user finding a geocache using geocaching application. (https://www.geocaching.com)*

Figure 1 shows a typical cache. They are usually a small waterproof container, and they contain a small logbook. The logbook allows the participants that found the geocache to sign their code

as well as the date that they found the geocache [2]. After a cache is found, the participants put the cache back exactly where they found it and the next group to find it will also sign the logbook which creates an endless cycle. Although there are many types of geocaches, these are the most common. Geocaching was a great example of a geolocation system that logs where users have been as well as a mobile application to display where to go.

## 2.2 Replay AR

Replay AR is an application created by ReplayAR, Inc that is available on iOS and Android. The application has two main uses. The first use is to be able to create any image into an augmented reality experience. That AR image can then be interacted with in the app and even allows users to create props [3].

Figure 2 Replay AR overlaying holiday theme over street image not during holiday. (https://replayar.com/)

The feature that related more closely to Geolocated Memory Sharing is the ability to overlay the augmented reality image on the current world as seen in Figure 2. Replay AR allows you to freeze your memories where and when they happened so if you revisit them any time later you

can view them again. Figure 2 is displaying this by showing holiday decorations from the past on a street that is no longer decorated. It overlays the past on the present day.

# 3. Design Approach

This project was written for mobile devices because of the need to be able to travel with a device that has location sensors, a camera, and the ability to receive notifications. When choosing what mobile platform to use I chose to use Flutter [4] specifically targeting its use for Android, which offers native app performance. Although Flutter allows for cross platform implementation, I developed strictly for Android. This was because of the different limitations with background location tracking and notifications. iOS restricts the ability to send notifications while the phone is in a sleeping state where the application is minimized or closed. As well as the strict limitations for location tracking and geofence storage.

Flutter also has a unique programming language called Dart [5]. Dart allows for hot reloads which drastically increased developments speeds when making many small modifications. Flutter is also a widget-based language that allows for quick UI creation and modifications. With this, it allowed me to focus more energy on design decisions related to geolocation and storage constraints which will be discussed further in sections three and four.

## 3.1 Storage

The storage system uses both local and cloud storage alternating between which one is used depending on the situation. I created a system where it will cache things locally to avoid querying the cloud data. This was done to avoid unnecessary data usage and the cost attributed with multiple Google Firebase [6] queries. The reason I used a cloud Firestore [7] database is because I needed a system where I could store memories and geofences online while being able to distribute these new memories and geofences to currently active user phones. This way all users would have access to all the public memories.

By storing information locally, I can also avoid latency. When pulling from the cloud during key events such as taking a picture or entering a geofence there is latency involved which will create a large delay for the user. By creating the geofences ahead of time and storing them locally on the phone and storing the information about which geofences were triggered through a local cooldown table on the phone, I was able to avoid the latency involved by querying Firestore when walking through a geofence. I was also able to offload authentication and memory grouping to local storage as well. By creating an authenticated flag and date when the last memory was modified locally, I avoided the latency for the log in process and did not require a cloud query when adding to a memory.

## 3.2 Geofences

Originally when I started this project, I created the app using active location tracking. There were many issues with this approach but the primary one was Flutter for Android only allowed one location query every 10 minutes while the phone was closed, this limit did not exist with geofences.

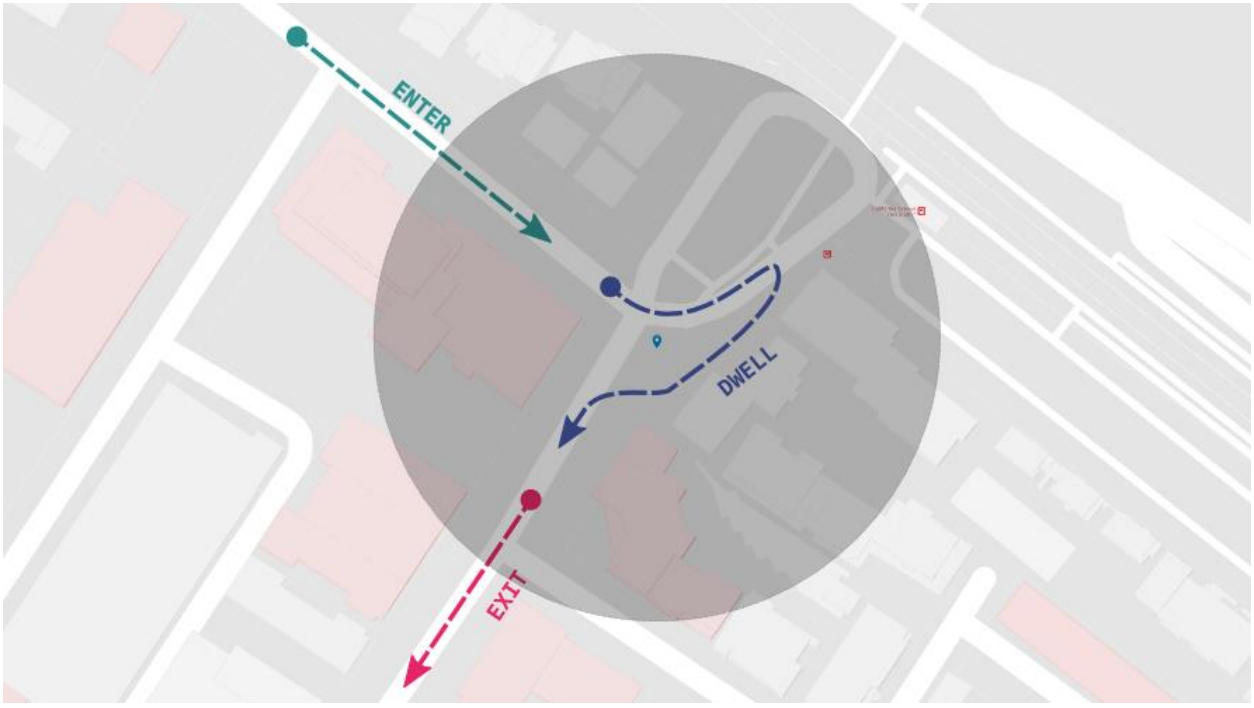


*Figure 3 Geofence representation with the three main geofence events.*

Geofences are created with three events [8]. An `enter` event, `exit` event and a `dwell` event as shown in Figure 3. The `enter` event was the perfect event which allowed me to offload the primary system of figuring out when a location was entered for the first time to the operating system. When location tracking, the difficulty was creating a system that understood that it has already triggered a notification without sending multiple. Although active location tracking is more accurate it was also consuming very large amounts of the battery. Monitoring geofences

lowers battery consumption and the system I use to monitor Geofences through `FlutterBackgroundGeolocation` does not actively enable location services. It will only enable location services when the phone is deemed to be nonstationary.

Geofences also have a radius that needed to be configured to work best with this application. The minimum the size of a geofence that will trigger every time, as recommended by Android, is 150 to 200 meters [8]. Through testing I determined that by setting the geofence priority to the highest priority I was able to shrink the geofence size to 50 meters. Because users of the application are using mobile data, a radius of 50 meters is more than accurate enough. For future work, after a geofence is triggered, I can turn on active location tracking and display the notification when they are even closer to the original memory location. Using Geofences, I no longer had to reinvent the wheel and was able to use the native Android Geofence systems as part of the Android SDK to allow for more efficient location triggers.

## 3.3 Memories

The memory system is the backbone of the thesis application. Memories are what users create when they interact with the application. A memory stores media as well as a reference to the location in which they were created.

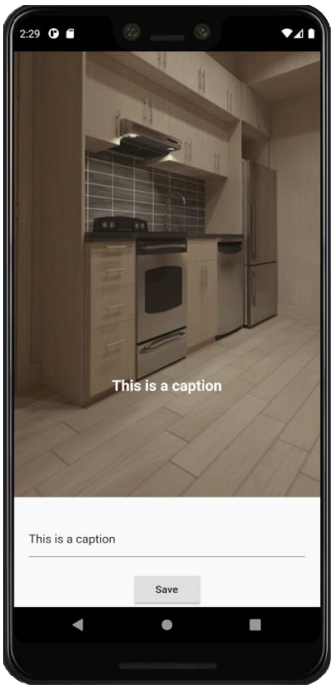


*Figure 4 Memory creation and caption addition screen.*

Memories are created after an image is taken and the save button is clicked. The user interface for this process is displayed in Figure 4. When a picture is taken the user has the option to add a caption and click save. The caption system was created so that users can add some reminders or personalization to their media. Allowing users to add a caption can better remind the user what emotion they were feeling when they first created the memory to then trigger a similar emotion when seeing the memory again. In the original version of the application, I did not include a save button. The original philosophy was to have all images taken through the

application to be created into memories to create an unpredictable experience. After initial testing I realised not all pictures should turn into memories and by adding a save button I created a confirmation screen. This was to ensure that only pictures that are important to the user or that the user wants to share are being added to the memory system.

I also ran into a problem where many geofences and memories were created in the same area within a short interval of time. Because of this I created a system that will group memories together to avoid bothering a user with multiple notifications in the same area for similar memories. Multiple memories created during the same trip or location are grouped into one memory which allows the user to get notified of all the memories in one notification.

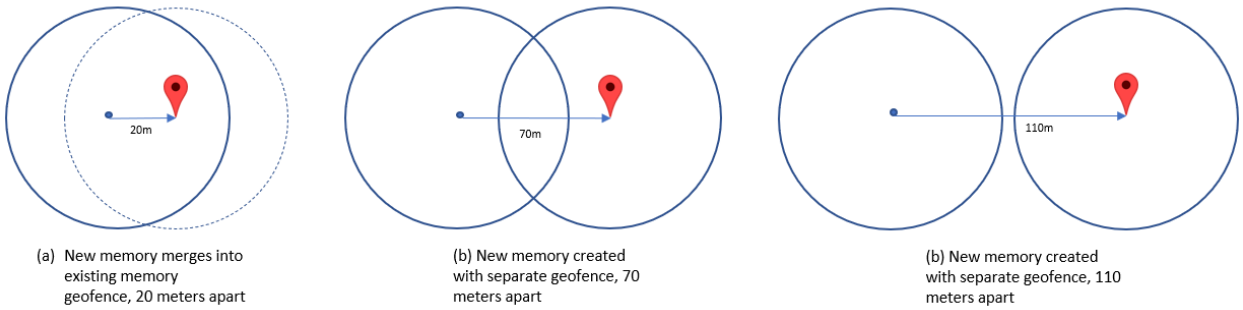


*Figure 5 Memory grouping system displaying 3 possibilities, within previous, overlap and separate.*

When a user creates a memory within 50 meters of the previous memory while also being within a 20-minute window of the previous memory creation those memories will be grouped together. Figure 5 shows this process by outlining 3 scenarios of overlap which can lead to memories being grouped or memories remaining separate. The initial memory time is also saved which adds another condition. If the initial memory is more than two hours apart from the current memory that is being created in the same area, it will create a separate memory and geofence. This is to signify that a large enough period of time has passed, and this is a new experience.

## 3.4 Notifications

The notification system is what decides whether a user should be notified of a memory that was just triggered. It is also responsible for sending a notification with the relevant information to the user. It listens for when that notification is clicked to display to the user the memory they were notified of.

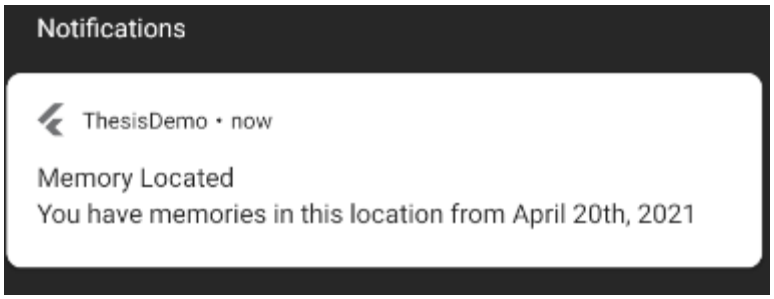


*Figure 6 Notification received after triggering a geofence.*

The notifications have a very basic title explaining that a memory was located as well as a message body that displays the date the memory was created as seen in Figure 5.

The system to decide when notifications should be shown is the largest section of future work for the application. The current system decides to show memories based off the cultural tradition to recognize anniversaries. When a memory is created there is an arbitrary amount of time before the memory becomes eligible to notify users. This time is currently one year. After one year a memory will be eligible for users to receive a notification. The reason for this was because there needs to be sufficient time in between when the memory was created, and the notification received that the feeling is serendipitous.

The notifications also have a cool down system. The cool down ensures that a user is not notified of the same memory multiple times in one trip or multiple trips in a short time period.

The cool down period is currently one year to play off the cultural traditions of recognizing anniversaries. This cooldown is per user and not a global cooldown. Currently everyone is eligible for all notifications as long as the cooldown time has passed. Once a user receives a notification for a memory, other users are still eligible to receive notifications for that specific memory. The notification system was designed to provide a feeling or surprise and wonder to the user. Trying to establish a similar emotion as what the user felt when they initially created the memory.

# 4. Implementation

In this chapter I discuss the implementation details of the application. This chapter is split into four different sections. The first section discusses how the storage was implemented. It discusses three different storage types, cloud storage, local database storage and XML document storage. All these storage systems are used to allow for reduced latency and data usage. The next section discusses Geofencing and geolocation. It explains which plugins are used for the geofences and geolocation as well as the techniques used to share geofences across multiple mobile devices. The next section discusses memory creation. The memory creation sections describe the process in which memories are created and how they are grouped together. Lastly there is the notification section. This section explains how notifications are created as well as the system which decides if a user should receive a notification.

## 4.1 Storage

There are three storage methods within the application: cloud storage, local SQLite storage, and shared preferences. Shared Preferences [9] is an XML style document stored locally on the phone. It is built into Android and allows storage of primitive data, such as Strings, Booleans, and Integers, in a key value pair system. I used the shared preference system to

store the session of the last user by saving their username and their current logged in status. The shared preference system also stores information about the last date the phone created a memory and the location of the last memory. The information is mainly pairs of Strings but the location is two pairs of String to Double to represent longitude and latitude. The login information is stored for the purpose of checking if there was a previous session on App start-up to perform an automatic login and the last memory information is stored so that it can be later used during memory creation. The SQLite storage as mentioned in the design approach was added to avoid using Cloud storage for information that can be stored locally to save on data usage and remove latency. The SQLite database has two tables. One table stores local memories and one table stores which memories a current user has recently seen by storing the unique ID of the memory that was seen. These unique ids are stored as individual rows where the columns contain the current user id and the unique id of the memory.

The last kind of storage that is being used is cloud storage. For cloud storage I am using Google's Firebase [6]. More specifically I am using three features: Firebase Firestore [7], Firebase Authentication [10] and Firebase Storage [11]. These features required `firebase_core`, `cloud_firestore`, and `firebase_auth` flutter plugins to be enabled. Firebase authentication is used to ensure that authentication using the users email and password is correct. This service will not only keep all of the passwords encrypted it allows for base user account modifications like password recovery and modification. Using the authentication system allowed me to distinguish which user is currently triggering a memory to better determine which user should receive the cooldowns. The next cloud storage system I am using is Firebase Storage.

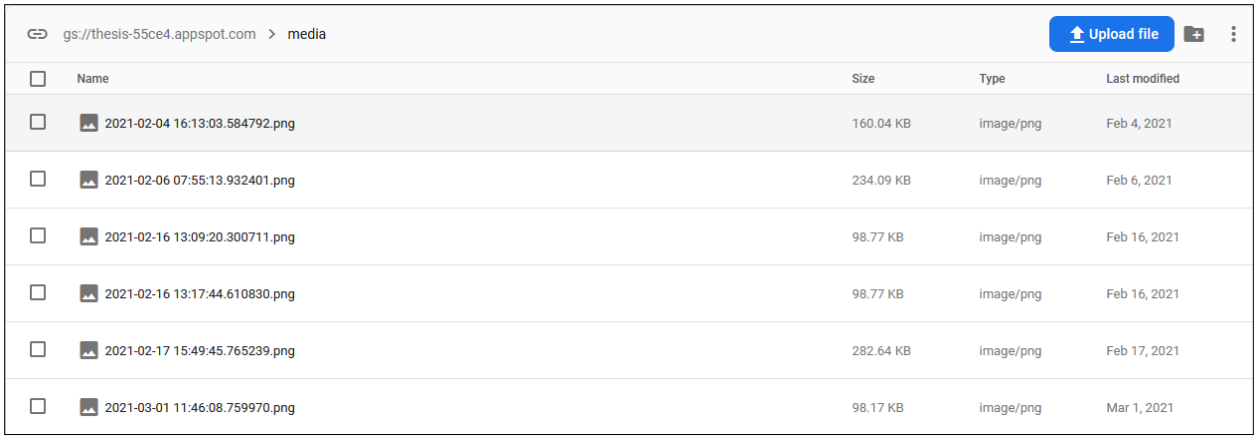


*Figure 7 Cloud storage directory to store media images.*

As seen in Figure 6 I create a folder to store all the media that is being created for different memories. Memories that the user creates can also be stored in their local file system, where the application will prioritise pulling from local data before pulling from the cloud. Because of the layout of the cloud storage, I recreate the same file path in the local system so the process of grabbing the media is similar. When a memory is created the media will be saved on the cloud to ensure that the memories can be shared among other users using the application. The last things I needed to store where the geofences, the memories and the information about the geofence even that was triggered. This is all done through cloud Firestore. Firestore is a NoSQL document database that allows multiple phones to store, query and synchronize data [7]. I created three different collections within the Firestore.

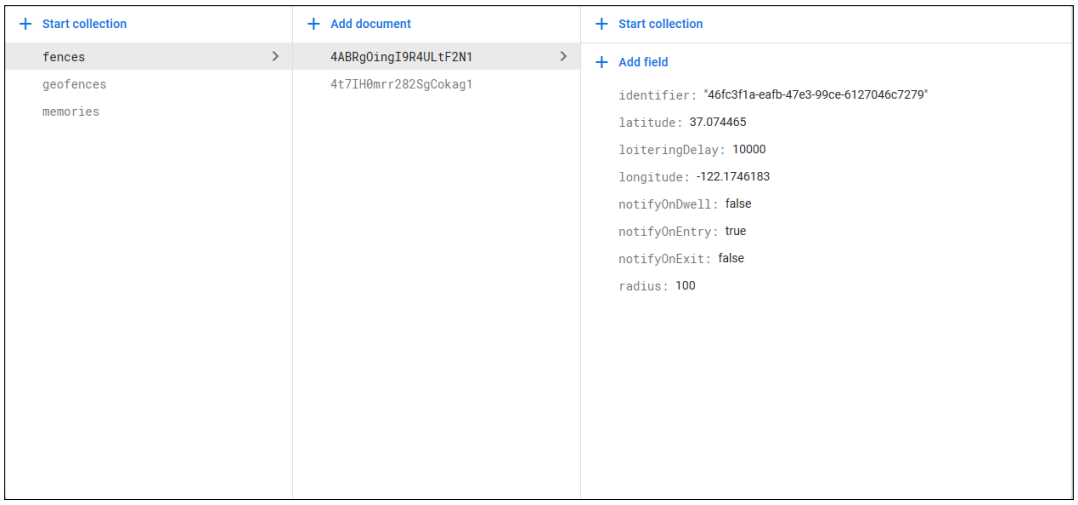


*Figure 8 Cloud storage representation of a Geofence*

The first collection is called *fences* which stores a simple representation of a Geofence as seen in Figure 7. It contains a v4 unique ID as its identifier as well as the geofence location, size, and trigger events. That unique identifier is then used in the next collection called *memories*.

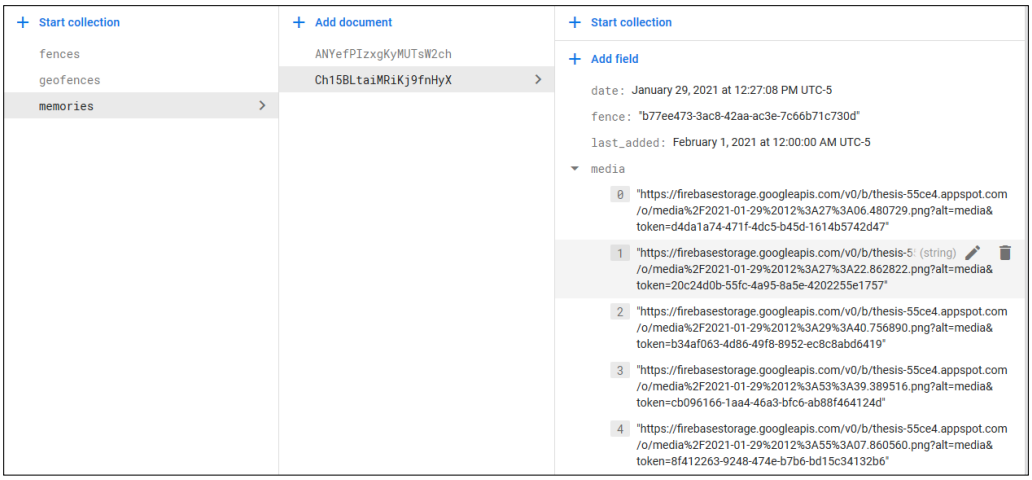


*Figure 9 Cloud storage representation of a memory*

Figure 8 shows the memories collection which contains memories that are connected to a Geofence. Within that memory document it will contain the Geofence identifier as well as the media, the date the initial memory was created and the last time the memory was modified. The

media is an array that takes reference locations to the previously mentioned media folder in the storage section of firebase. By maintaining references to both the media and the geofence identifier I can receive all of the information I need from a query comparing the current geofence triggered ID to the memory ID. The third collection is the geofence collection. This collection displays all of the data concerning when a geofence event is triggered. The information stored includes a movement flag, the odometer data, a timestamp, heading, latitude and longitude coordinates as well as altitude, battery levels and speed.

The synchronization is done by creating a stream and listening directly to the Firestore. By listening to the memories collection of documents, when a document is modified or a new document is added, all phones that are currently active will also add that memory to their local storage. Doing this allows me to share geofences between users. The phone will also pull from storage during boot to ensure that it is up to date on all the latest memories that were created. Using cloud storage paired with local storage allowed me to create a system where I use the best storage type for the specific job that needed to be accomplished.

## 4.2. Geolocation

The next thing that I needed to implement was the system that would notify the phone that a user is in a current location. I ended up using geofencing and using the `FlutterBackgroundGeolocation` implementation of location tracking and geofencing. `FlutterBackgroundGeolocation` by Transistor Software [12] is a battery-conscious location tracking and geofencing module for Flutter. The plugin uses motion detection API's such as the accelerometer, gyroscope, and magnetometer to determine if a device is stationary or moving. When the plugin determines that the phone is stationary it will fully disable the phones' location

services to save on battery. The plugin has many configuration options and needs to be configured in the initial state of your primary widget. One of the main features to establish is the distance filter. This will determine how often the device will check for its location based on the distance it has travelled. In the case of my application, I have the distance filter set to one meter. The reason for this is the primary application being walking about required a more accurate location filter. I also set the background geolocation to not terminate when the app is closed `stopOnTerminate: false`. This allows me to continue to record geofence notifications even if the application has been turned off. I set my desired accuracy to the highest possible accuracy and ensure that the specific geofence accuracy is also set to the highest. Setting the geofence accuracy higher allows me to lower the radius of the geofence for more precise notifications. And lastly, I set the background geolocation to start when the device boots to ensure that the plugin initializes right as the phone turns on [13].

Once the plugin was initialized, I needed to create an event listener to be able to listen to when a geofence has been triggered. The geofence UUID is the same as the memory associated with it so I was able to query the memory database with the information provided from the geofence event. Now that geofences were being listened to, I also needed to create the geofences.

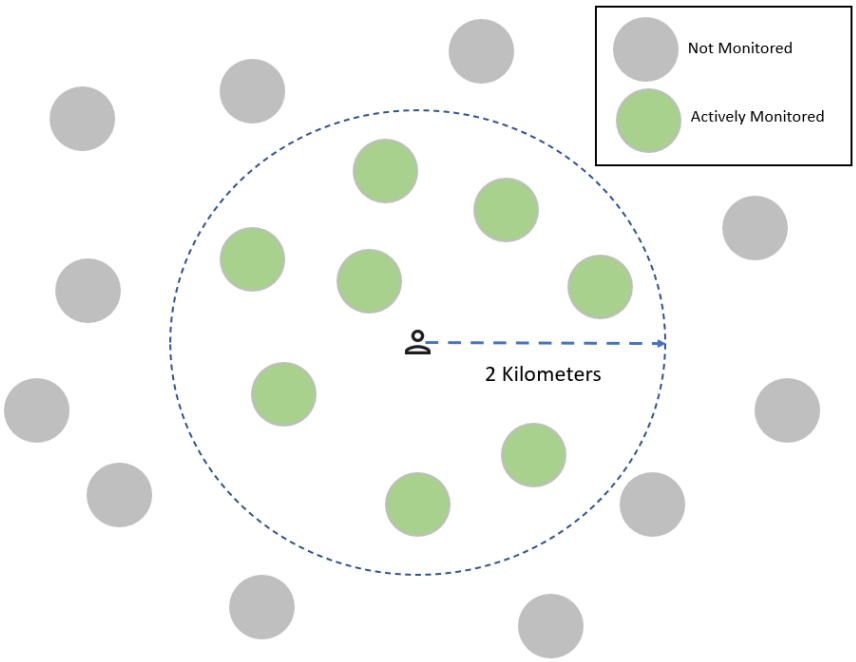


*Figure 10 Actively monitored geofences in a 2-kilometer radius all other geofences not actively monitored.*

Android has built in Geofence support and with the `FlutterBackgroundGeolocation` plugin I was able to create an infinite amount of geofences while only actively monitoring geofences in a 2-kilometer radius around the device as seen in Figure 10. Because Android has a limitation of 100 actively monitored geofences, a geospatial query is done to ensure that the closest 100 geofences are being monitored. All geofences outside of the radius are deemed inactive and are no longer being monitored. The geofences needed a latitude and longitude location as well as a radius and a trigger to listen to. To obtain the location for the geofence I would check the previous locations that have been recorded, if there was no location recorded, I would preform another location check and create a geofence that has a radius of 150 meters around the last location that only listens to the enter event. Using the `FlutterBackgroundGeolocation` plugin allowed me to use the native geolocation features within Android in a nice wrapper.

## 4.3. Memory creation

Memory creation was a pivotal part of the application. I needed an easy system to create a memory that would have some sort of meaning. As an initial plan I needed to create a system that would allow me to open a base camera screen with the ability to preview the image after it was taken. To do this I implemented the flutter Camera plugin. I then pass the image created from the camera into a new widget of the image to display the preview image while saving the image. Because of the active location tracking I can take the last location that was logged and start the process of creating the memory. By saving locations as the phone moves, I can avoid the unnecessary processing time of finding the current geolocation while taking the picture. This was an issue in my first implementation because of the long wait times during the camera snap.

After the picture is taken and the image is created, and the location is now saved I start the process of creating the memory object. If the user then decided that they would like to save the memory I provide them the option to add their caption and click the save button. The image editing is using a Stack object with the order of the caption being higher than the image that was taken. Once the save button is clicked, the image size is processed and a new image is saved by taking the screens current state and saving it into a new image, with or without a new added caption. Before the memory is created, I check to see if there are any existing memories in the area that are still available for this media to be added to them, I do this by querying the existing memories comparing their times, the user that created them and the locations to ensure that the query only returns suitable memories. If a suitable memory is found I take the newest memory in the list and modify the storage for the existing memory to ensure that the new media is added to the array in the Firestore document. This was all done asynchronously because of the increased amount of time required by a more complex query of the NoSQL Firestore. While this task is finishing the background, the user is once again directed to the camera screen allowing them to

create more memories. This system was created to be easily expandable but still simple for a first-time user to understand.

## 4.4. Notification system

The notification system is what would bring the app a sense of serendipity and provide an implicit interaction to the experience. The notification system was created using `flutter_local_notifications`. This is a flutter plugin that allows me to display different types of notifications. For Android it supports multiple image notifications as well as basic notifications. To use the notification plugin, I created an initialization as well as a listener that will listen to when a notification is clicked. It was important that after a notification was select the application would open showing the current memory to the user. To do this I created a `BehaviorSubject<String>` stream. `BehaviorSubject` streams are from the React library [14] for Flutter. It allowed me to create a stream that will emit the current value, or latest value that the stream has. Behavior subjects are good to represent values over time which why I chose to use one to represent the notifications. Because multiple notifications are being sent out over time, I wanted to be able to find the notification payload easily through a stream that I am subscribing too. After the stream was subscribed to, I needed to display the current memory from the payload. The payload is determined when the notification is being sent out and, in my case, it is always a singular UUID in its string representation.

The next step was creating the actual notifications and their trigger conditions. The notifications are created when a geofence event is triggered, but only if that memory is not currently on a cool down for that specific user. The notification is created with a simple and readable title “Memory Located” to inform the user in a glance that they have stumbled upon a memory. If the notification is expanded it will display that the user has memories in the current

location from a specific date, and if it is only one image it will display that image. The goal with this system is to allow it to carry a small payload while still being able to display all the information I needed. Originally, I was encoding images straight into the payload and quickly ran out of payload room as the max string length is 1024 characters. Using this notification system, it also allows me to send notifications through Firebase messaging which will allow me to send notifications even when my application is in a hibernation or off state.

# 5. Future Work

While creating the thesis applications many options for future work became apparent. Due to time constraints many of these features could not be added. Some notable features would be ability to use different types of media to memory creation and augmented reality displays for the memories. The application only supports images, some other possible memory types could be videos, drawings, sound bites and letters. The storage system allows for any media type to be stored but the UI would need to be modified to allow for displaying these memories. Augmented reality was also originally a design goal for implementation. Implementing an augmented reality overlay would require a different system for memory displaying. One option would be to remove the main screen and create a camera screen that would allow for augmented reality overlays. Both additions would require UI changes that would modify the visual elements of the applications.

Another opportunity for future work is to increase the accuracy in the location tracking. Because I am using geofences the trigger event happens at the edge of the configured radius of the circle.

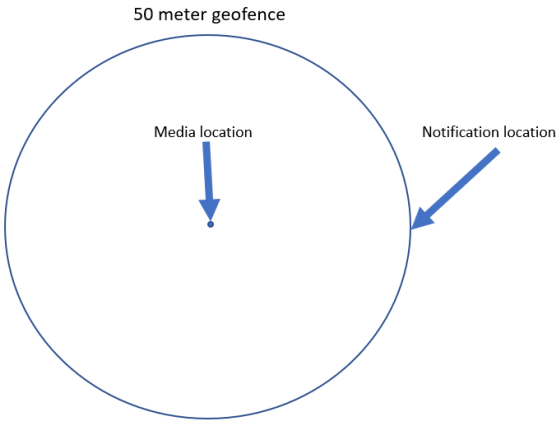


*Figure 11 Geofence displaying issue with notification location versus the location of the media.*

As seen in Figure 11 the default radius is 50 meters which means in the worst-case scenario you are 50 meters away from the location of your memory when you get a notification.

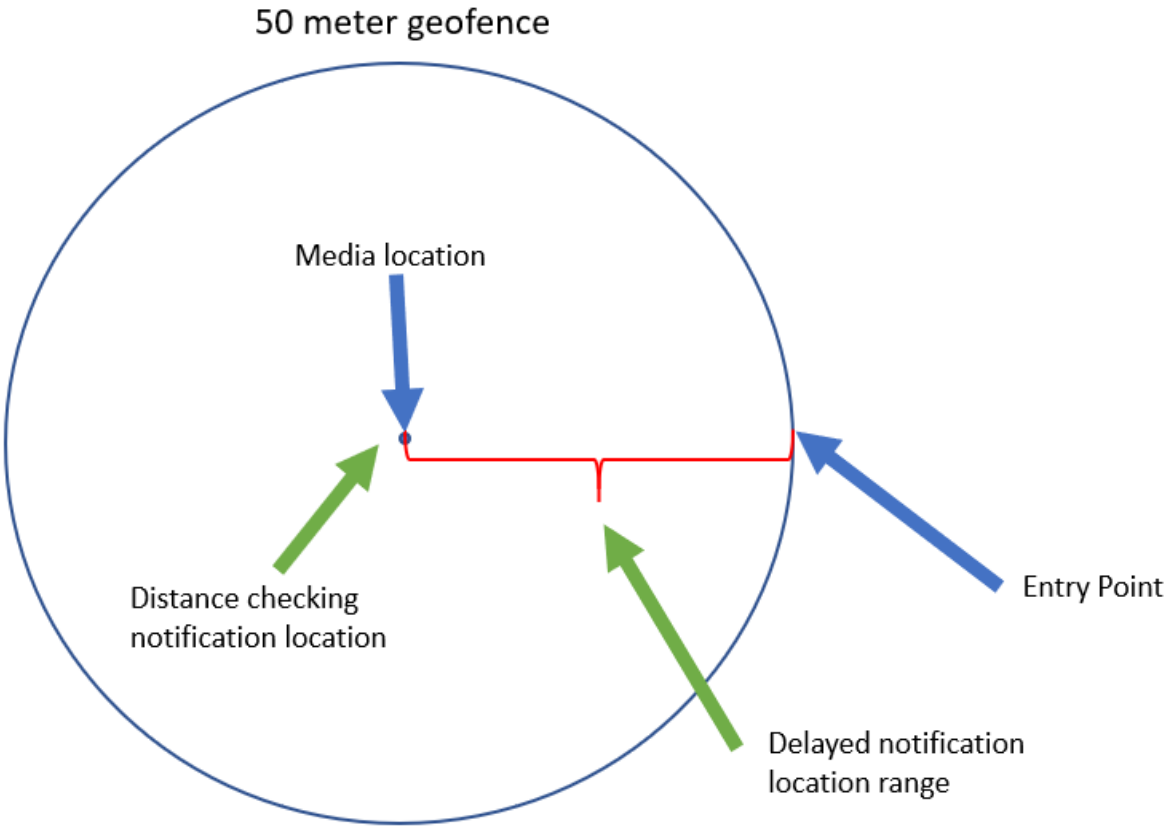


*Figure 12 Possible solutions for more accurate notifications.*

There are two possible options shown in Figure 12. The first option is delayed notifications. By knowing the heading and the speed I can create an estimate for the delay of how long it will take a user to reach the center of the geofence and delay the notification by that that time to create a more accurate notification. The second options shown in Figure 12 is active location distance tracking. I would be to transition from strictly monitoring geofences to active location tracking when enter a geofence. Monitoring distance while active location tracking to find a location that is closer to the center of the geofence would require a system which either looks for a distance close enough to the original memory location or to monitor the change in distance. If the distance is monitored and it is determined that the user is moving toward the centers the centre, once they reach a threshold and start walking away from the center, that is when a notification should be displayed. I think there is a lot of potential in making the notifications more accurate.

More accurate notifications would also allow for a more seamless experience with augmented reality overlays.

The larger opportunity for future work is understand what a user will find interesting. Right now, I only use a time-based system to determine what memories should be displayed but a system that is more sophisticated will allow for a better user experience. The goal would be to determine what a user is currently interested and scan existing memories to better understand which memories should be set into an active state. An example of this would be to look at previous images taken through the camera app to show similar images/memories, understanding milestones would help as well. Other information that would be useful is milestones such as birthdays, holidays, or anniversaries. Being able to harness all of this information to better determine what a user will find interesting would allow for me to show users more personalized memories.

# 6. Conclusion

The Geolocated Memory Sharing project is an application that allows users to leave memories of an experience associated with a location for themselves or others discover. This thesis was created as a mobile application which allows users to take pictures to create a memory and receive a mobile notification when they revisit that location of memory creation. The system has four main design and implementation sections, the storage, the notifications, the geolocation, and the memories. The system stores information locally as well as the cloud and implements geofencing to notify users when they revisit a memory location. This system attempts to produce a serendipitous feeling when they receive a memory notification.

# 7. References


[1] Geocaching. (n.d.). Get the free Official Geocaching app and join the world's largest treasure hunt. Retrieved April 23, 2021, from https://www.geocaching.com/

[2] Brigitte, /., Writer, /., Sarah, /., & Kettler, /. (2021, April 06). Learn. Retrieved April 23, 2021, from https://www.geocaching.com/blog/category/learn/

[3] The future of preserving the past. (n.d.). Retrieved April 23, 2021, from https://replayar.com/

[4] Flutter. (n.d.). Retrieved April 24, 2021, from https://flutter.dev/

[5] Dart programming language. (n.d.). Retrieved April 23, 2021, from https://dart.dev/

[6] Google Firebase. (n.d.). Retrieved April 23, 2021, from https://firebase.google.com/

[7] Cloud Firestore | Firebase. (n.d.). Retrieved April 23, 2021, from https://firebase.google.com/docs/firestore

[8] Geofences : Android Developers. (n.d.). Retrieved April 23, 2021, from https://developer.android.com/training/location/geofencing

[9] Shared Preferences. (n.d.). Retrieved April 24, 2021, from https://developer.android.com/training/data-storage/shared-preferences

[10] Firebase authentication. (n.d.). Retrieved April 23, 2021, from https://firebase.google.com/docs/auth

[11] Cloud storage for Firebase. (n.d.). Retrieved April 23, 2021, from https://firebase.google.com/docs/storage

[12] Mobile geolocation experts. (n.d.). Retrieved April 23, 2021, from https://www.transistorsoft.com/

[13] Flutter_background_geolocation. (n.d.). Retrieved April 23, 2021, from https://pub.dev/documentation/flutter_background_geolocation/latest/index.html

[14] React: Dart package. (2021, February 17). Retrieved April 23, 2021, from https://pub.dev/packages/react