SENG440 Assignment 2 Post Mortem

Sam Annand | sga111 | 48562140

02 / 06 / 20

Application Purpose:

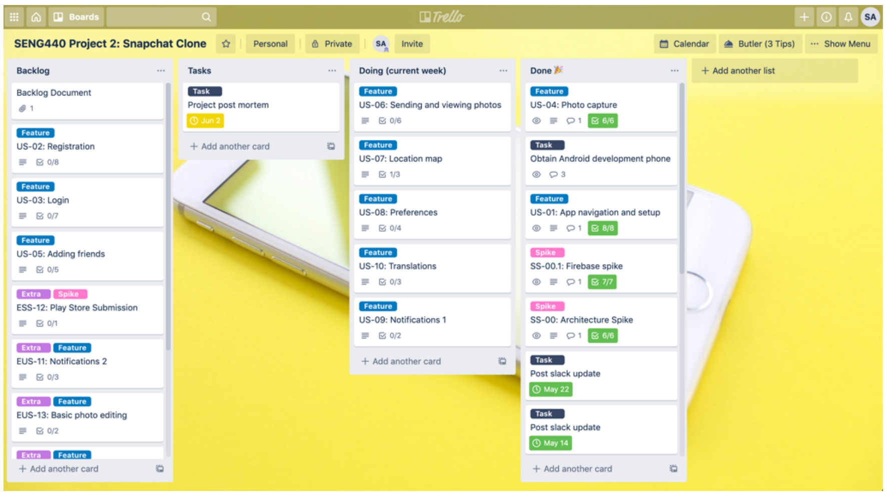
I have created an android application – *Crapchat* – aimed at users who like to take pictures, and send them to their friends as a means of keeping in contact, similar to texting, or Facebook messaging. The main purpose of the application was to clone, rudimentarily, the popular photo messaging mobile application – *Snapchat*. As will be seen in this document, the main purpose of being able to take a picture, and send it to friends has been achieved, however it was not achieved in the most feature rich way. Whilst the purpose of the application remains true, there is still some future work required in order to consider the application a true Snapchat clone.

Development Process:

The development process for Crapchat started over the term 1 break, where I saw a job listing by Snapchat for a Software Engineering position in Sydney. Upon reading the brief for the second mobile computing project, I realised that Snapchat fits the theme of the assignment very well, as it is very connected to the immediate surroundings using a location map of friends, and the camera. It is also connected with other places outside the app with the facility to share memories and talk to other users. This lead to me wanting to creating a rudimentary Snapchat clone, that I could hopefully use to my advantage if I was to apply for a job at Snapchat in the future.

Creating a Snapchat clone was very ambitious of me, and as will be seen throughout this document, unfortunately components of the app had to be left out in order to submit in time. I feel confident that I could add them in for personal benefit in the future. To deal with the ambitious nature of the project, I followed a semi-strict agile development process, specifically Kanban.

As can be seen in the figure below, a Trello board was created and used to manage story backlog items, as well as general admin tasks. I found this helpful up until a point late in the project where the administrative cost was too much and I just needed to get stuck into development.



SENG440 Project 2: Snapchat Clone

Backlog
- Backlog Document — 1
- Feature — US-02: Registration — 0/8
- Feature — US-03: Login — 0/7
- Feature — US-05: Adding friends — 0/5
- Extra, Spike — ESS-12: Play Store Submission — 0/1
- Extra, Feature — EUS-11: Notifications 2 — 0/3
- Extra, Feature — EUS-13: Basic photo editing — 0/2
- Extra, Feature
- Add another card

Tasks
- Task — Project post mortem — Jun 2
- Add another card

Doing (current week)
- Feature — US-06: Sending and viewing photos — 0/6
- Feature — US-07: Location map — 1/3
- Feature — US-08: Preferences — 0/4
- Feature — US-10: Translations — 0/3
- Feature — US-09: Notifications 1 — 0/2
- Add another card

Done 🎉
- Feature — US-04: Photo capture — 1 — 6/6
- Task — Obtain Android development phone — 3
- Feature — US-01: App navigation and setup — 1 — 8/8
- Spike — SS-00.1: Firebase spike — 1 — 7/7
- Spike — SS-00: Architecture Spike — 1 — 6/6
- Task — Post slack update — May 22
- Task — Post slack update — May 14
- Add another card

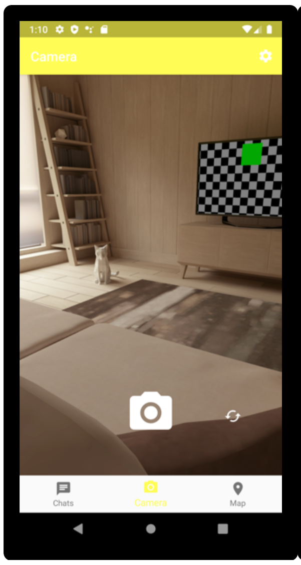
The stories that can be seen in the figure came from a backlog document that I created *prior* to any implementation. As is the requirement, these user stories have been appended to the end of this post mortem document. It should be noted that in some cases, minor changes were made along the way, mostly in the interest of time. These changes can occasionally be seen as comments on the Trello board (which the markers are able to request access to if needed), however have not been updated on this backlog of user stories. This was a conscious decision, as these stories still reflect the product that would have been created in ideal circumstances, and I did not want to take from that. I also wanted to keep them the same in case I do come back to this project and polish it up for personal benefit.

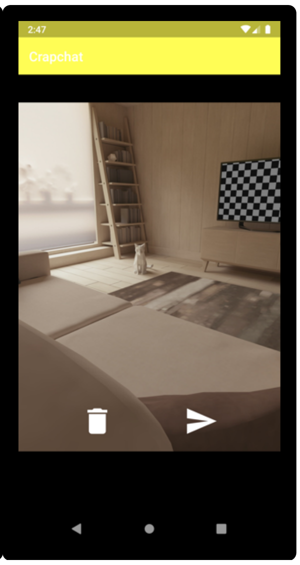
Having completed the development of Crapchat, I feel pleased that I was able to at least hold true to the purpose of the application, as well as frustrated that I was not able to properly implement some key features of Snapchat such as adding friends, and sending them photos *within* the Crapchat app chats view. At the end of the day, this came down to time and the ambitious nature of the app requiring a backend. On a positive note, all grade bearing requirements were met and the overall purpose is present.

Requirement enumeration:

1. **Interact with the nearby physical world in some way. This interaction might rely on sensors, GPS, the camera, or Bluetooth to create a local area network.**

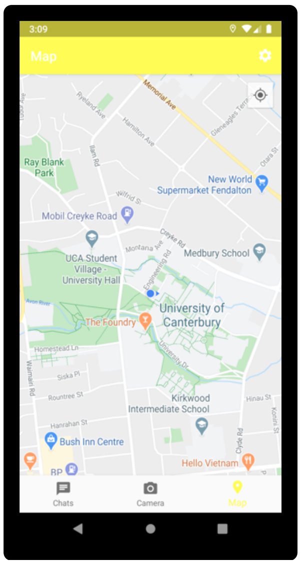
   Crapchat firstly interacts with the nearby physical world by using the device's camera. The modern CameraX library was used to create a camera view that is a central part of the application. The user can press the camera icon to snap a picture from the view finder, which is automatically saved to the device unless they delete it when the option is presented after taking the picture. The user also has the option to switch between front and rear facing cameras.





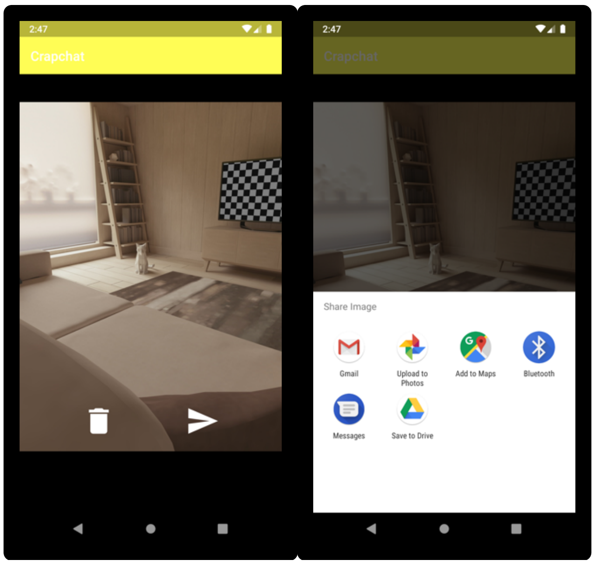
2. **Interact with the nearby physical world in some additional way.**

   Crapchat secondly interacts with the nearby physical word using the devices GPS system. A Google maps view is available to the user that utilizes the GPS services on their device to display where they are on the map. In the future, this will be expanded to also show the users Crapchat friends on the map. It should be noted that this feature specifically uses GPS, as opposed to other locations services, such as IP address location.



3. **Provide a facility for "openness" – for a user to interoperate with entities beyond their phone or beyond your app. This could take many forms. You could allow the user to export a backup of your app's data that could be imported by someone else or on another device. You could provide support for deep-linking, such that certain patterns of URLs would trigger your app. You could allow the user to share achievements via texting or social media. If your idea for this feature situates your app within the larger community, it probably qualifies.**

Initially, as can be seen in the user stories attached to this document, for this requirement I was hoping to have users, that could add friends, and then send them photos within the Crapchat app. This proved too ambitious, and as such a comprise has been made that simply allows the user to send the photo to their friend using existing apps on their device – even the real Snapchat!



It should be briefly noted that email and text are the only logical options presented above simply because the emulator does not have other apps installed by default such as Facebook, Instagram, or Snapchat.

4. **Gracefully handle configuration changes, not losing any of the user's data**

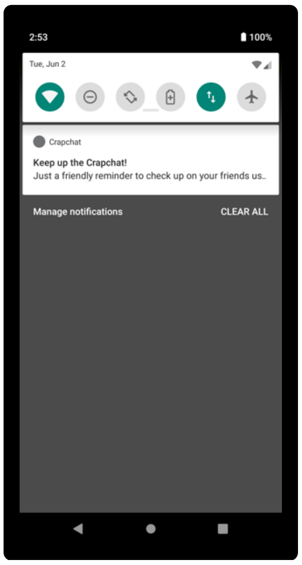
   All configuration changes are handled and user data is not lost. Before restricting the screen rotation, explained shortly, I had to make a change to the preference screen on rotation as it was causing the application to crash. This was achieved by swapping a <fragment> tag to a <FragmentContainerView> tag. I also use a custom room datastore to save the preferences immediately, with no chance of state being lost. I also have empty ViewModel classes for the three main screens of the application. These would be used whenever mutable data was required on the views, however this was not needed in the end. A good example of this can actually be seen in my first project where ViewModels are used in conjunction with LiveData to prevent data loss during configuration changes.

5. **Use a local database to persist data, preferably using Room instead of raw SQLite**

   I have used a Room database in my application as a custom datastore for the preferences used in my preference screen explained shortly. Related code can be found in the *sga111.seng440.crapchat.room* package, as well as the *sga111.seng440.crapchat.ui.settings* package.

6. **Send the user notifications related to your app in some way.**

Provided that the user has enabled the notifications setting in the preferences screen mentioned below, Crapchat will send a daily notification at 9:00 am every day telling the user to make sure they check up on their friends using the app. Topical given current circumstances!

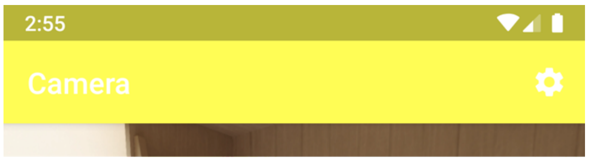


Keep up the Crapchat!
Just a friendly reminder to check up on your friends us..

Notification scheduling code has been abstracted into a Util object, and is called upon creating the MainActivity (if a notification hasn't already been scheduled). A preference change listener is also set up on the preference screen to immediately remove a scheduled notification or create one based on the setting change.

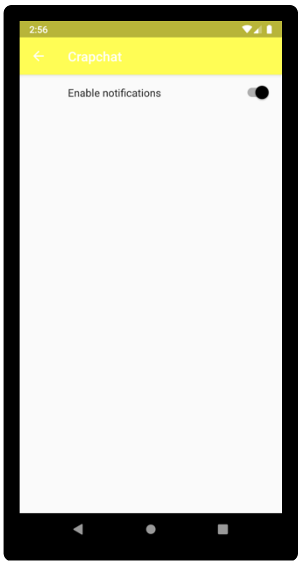
7. **Integrate an action bar in at least one activity.**

   The application's MainActivity, which holds the three main fragments for the navigation view, has a permanently visible action bar with a settings icon that will present the preferences screen to the user when clicked.



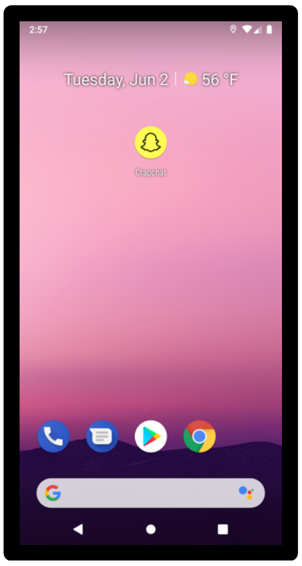
8. **Provide a preference screen using the modern AndroidX Preference Library.**

   As mentioned above, on clicking the settings icon the action bar present in the MainActivty of the application, a preference screen is presented to the user. This screen was created using the AndroidX Preference library, and contains the option for the user to turn on or off notifications. This preference is read when the application starts to determine whether or not a notification needs to be scheduled, and there is also a change listener on the preference screen to immediately schedule or remove scheduled notifications when the preference changes.



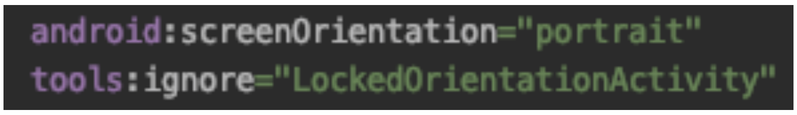
9. **Add a multi-resolution launcher icon.**

   Given I was attempting a Snapchat clone, a multi-resolution launcher icon resembling the Snapchat icon was added to the application, and can be seen in the figure below.



10. **Support both landscape and portrait orientations in all views – unless your content demands a fixed orientation, as in a game. In other words, all widgets should be able to be made fully visible in either orientation. This may happen automatically given your layout manager, or you may use a ScrollView, or you may specify two separate layouts.**

    During the development of Crapchat, I used constraint layouts throughout, resulting in this requirement being met, for the most part, automatically. As mentioned earlier, I did have to make a change to the preference screen XML first. As I continued doing this, I noticed that the landscape mode did not suit the style of the app. Snapchat, the inspiration for this app, also does not support landscape in its views. As a result, I have since restricted the orientation into portrait mode using statements seen in the figure below in all activities defined in the manifest.



```
android:screenOrientation="portrait"
tools:ignore="LockedOrientationActivity"
```

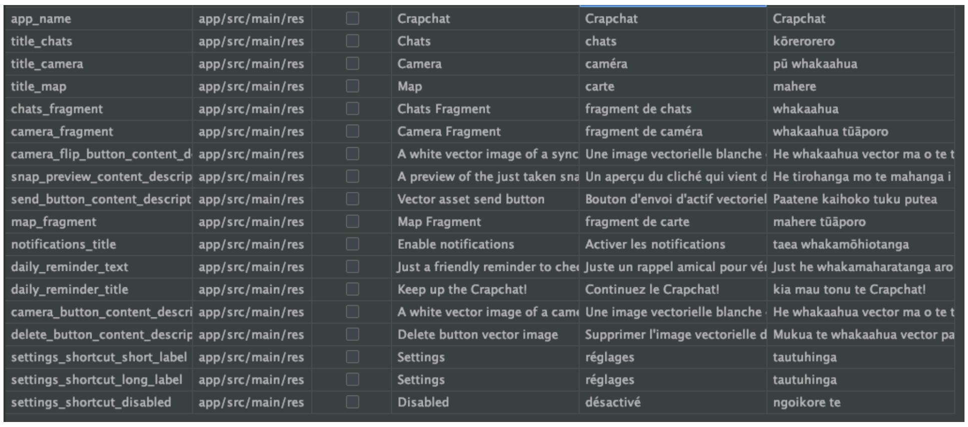
If the marker is interested in checking this requirement was actually met before doing this, they need only to remove these statements from the manifest.

11. Use string resources for all static text on the user interface.

String resources have been used for all static text on the UI, as well as some non user facing text such as image content descriptions. The defaults for these strings can be found in the strings.xml file.

12. Provide default definitions for your string resources in English. Provide definitions for one other language. (Use your favourite online translator if necessary.)

As seen in the figure below, string translations have been provided for Māori and French. The reason there is two translations is that I first wanted to do Māori, however then realised that Android does not yet support Māori, or at least the version I used didn't.



app_name	app/src/main/res	☐	Crapchat	Crapchat	Crapchat
title_chats	app/src/main/res	☐	Chats	chats	kōrerorero
title_camera	app/src/main/res	☐	Camera	caméra	pū whakaahua
title_map	app/src/main/res	☐	Map	carte	mahere
chats_fragment	app/src/main/res	☐	Chats Fragment	fragment de chats	whakaahua
camera_fragment	app/src/main/res	☐	Camera Fragment	fragment de caméra	whakaahua tūāporo
camera_flip_button_content_d	app/src/main/res	☐	A white vector image of a sync	Une image vectorielle blanche	He whakaahua vector ma o te t
snap_preview_content_descrip	app/src/main/res	☐	A preview of the just taken sna	Un aperçu du cliché qui vient d	He tirohanga mo te mahanga i
send_button_content_descript	app/src/main/res	☐	Vector asset send button	Bouton d'envoi d'actif vectoriel	Paatene kaihoko tuku putea
map_fragment	app/src/main/res	☐	Map Fragment	fragment de carte	mahere tūāporo
notifications_title	app/src/main/res	☐	Enable notifications	Activer les notifications	taea whakamōhiotanga
daily_reminder_text	app/src/main/res	☐	Just a friendly reminder to che	Juste un rappel amical pour vé	Just he whakamaharatanga aro
daily_reminder_title	app/src/main/res	☐	Keep up the Crapchat!	Continuez le Crapchat!	kia mau tonu te Crapchat!
camera_button_content_descri	app/src/main/res	☐	A white vector image of a cam	Une image vectorielle blanche	He whakaahua vector ma o te t
delete_button_content_descrip	app/src/main/res	☐	Delete button vector image	Supprimer l'image vectorielle d	Mukua te whakaahua vector pa
settings_shortcut_short_label	app/src/main/res	☐	Settings	réglages	tautuhinga
settings_shortcut_long_label	app/src/main/res	☐	Settings	réglages	tautuhinga
settings_shortcut_disabled	app/src/main/res	☐	Disabled	désactivé	ngoikore te

13. Write a set of *user stories* for your app and include them in your post mortem. In addition, if you have someone available in your "bubble", have them test your app and record their feedback.

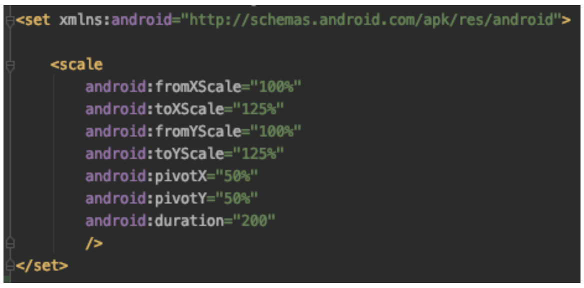
As mentioned earlier in this document, the unedited user stories that were created at the start of the development process can be found appended to the end of this document. Unfortunately due to submitting close to the deadline, user feedback from people within my bubble was fairly minimal.

- "Wow that looks amazing, definitely A+ material"

- “Seems like a nightmare to develop an app with camera functionality, well done!”
- “I’m going to stick with normal Snapchat”
- “Help, I can’t find the dog ears filter!”

14. Incorporate an animation into your UI, preferably one specified in XML. This time around, animated GIFS and fragment transitions don’t count.

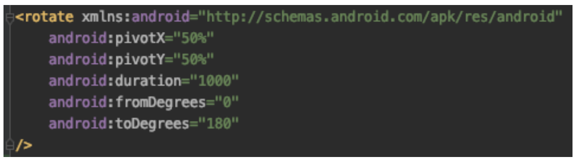
Two simple XML defined animations can be found in Crapchat. The first is an animation on the camera icon when taking a photo.



```xml
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <scale
        android:fromXScale="100%"
        android:toXScale="125%"
        android:fromYScale="100%"
        android:toYScale="125%"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="200"
        />
</set>
```

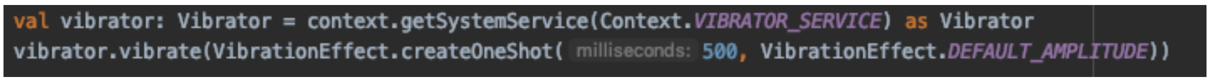
The second is a spin animation on the camera flip icon when swapping between front and rear facing cameras.



```xml
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="1000"
    android:fromDegrees="0"
    android:toDegrees="180"
/>
```

15. Incorporate some other Android feature not mentioned above into your app. (something different from the extra feature in project 1)

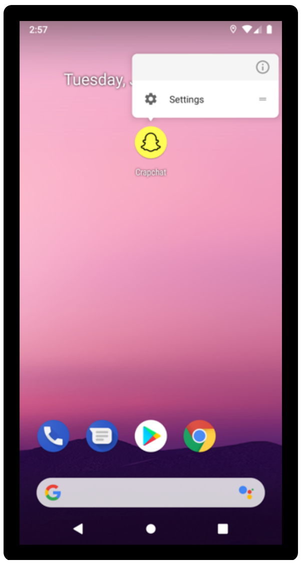
As an extra feature, I have implemented vibration when the user receives a notification from Crapchat.



```kotlin
val vibrator: Vibrator = context.getSystemService(Context.VIBRATOR_SERVICE) as Vibrator
vibrator.vibrate(VibrationEffect.createOneShot( milliseconds: 500, VibrationEffect.DEFAULT_AMPLITUDE))
```

16. Incorporate some other Android feature not mentioned above into your app. (something different from the extra feature in project 1)

As a second extra feature, I have implemented an app shortcut from the app icon. If the user long-presses on the icon, a menu presents them with the option to open the preferences screen of the application, to quickly disable or enable notifications.



17. **Share a plan for your app by the end of Friday (1 May), week 8 in a post on #project2 in Slack – before you've written any code or created any layouts. Include hand-drawn sketches or wireframes.**

    https://seng440-2020.slack.com/archives/CU1QVDF9V/p1588327153075700

18. **Share an update of your work by the end of Friday, week 9 in a post on #project2 in Slack. Include screenshots.**

    https://seng440-2020.slack.com/archives/CU1QVDF9V/p1588936503144700

19. **Share an update of your work by the end of Friday, week 10 in a post on #project2 in Slack. Include screenshots.**

    https://seng440-2020.slack.com/archives/CU1QVDF9V/p1589454038160400

20. **Share an update of your work by the end of Friday, week 11 in a post on #project2 in Slack. Include screenshots.**

    https://seng440-2020.slack.com/archives/CU1QVDF9V/p1590132511259900

Possible extra features for requirements 15 and 16:

As there doesn't seem to be a clear definition for extra feature, I have included a brief list of features present in Crapchat below that I believe could also be considered extra features for requirements 15 and 16.

- Use of CameraX library as opposed to just normal camera
- Implementation of a custom data store for the AndroidX preference library
- Saving photos to external storage, with the ability to delete
- Use of bottom navigation menu
- Use of FileProvider

- Use of GoogleMapsApi in conjunction with requirement 2 to display location on a Google Map.

SENG440 Project 2 User Story Backlog

Sam Annand
08/05/2020

Setting the scene:

For the second project of SENG440 I will be attempting to create a simple Snapchat clone. This will involve the basic user flow of being able sign up, take a picture, and send it to a friend. The friend can then view the photo, only once, and also send a photo back. They should also be able to see the location of their friends on a map. This workflow is very simple as it will be all that is required to check the grade bearing requirements off as outlined in the brief, as well as complete a minimum usable product. Time permitting, I would like to add more features (taken from Snapchat). I have included these features in this backlog in a separate section so there is a clear distinction between priority stories, and extra time stories.

Introducing the **User**:

For the sake of time and simplicity, the many possible users of this application have been grouped into **User**, having the following vague and generalised properties:

- Own an Android device running Android 9 Pie (API 29) or later.
- Device is able to connect to the internet at all times
- Device has a front and rear-facing cameras
- Has some friends that are keen to start sharing photos amongst each other
- User and friends of user are okay with sharing their location data by default

More traits about the user will be inferred as one reads through the user-centric stories below.

Core User Stories:

US (User Story) *SS (Spike Story)*

SS-00: Architecture Spike

As a developer, I want to take some time to investigate the architectural requirements of this application, documenting any plans, and requesting any resources needed, so that I can have a solid foundations upon which to develop the app.

Acceptance Criteria:

1. Have investigated existing solutions architecture
2. Document planned architecture
3. Request resources, or set up required resources
4. Made note of any possible roadblocks

5. Sufficiently tested any immediate blockers to ensure they will not be project blocking

SS-00.1: Firebase spike

As a developer, separate but related to the architecture spike, I want to take a specific dive into investigating what firebase is, and how it may be able to help me.

Acceptance criteria:
1. Understand how firebase may or not be able to help me achieve the user stories involved with this app
2. Update documentation related to architecture spike accordingly

US-01: App navigation and setup

As a user, I want to be able to navigate through a basic version of the application using a bottom navigation menu, so that I can find the map, my unopened photos, and a the camera.

Acceptance criteria:
1. User can tap a custom designed icon to use the app
2. User can navigate between the three main views (camera, map, and snaps) using some kind of bottom navigation
3. The middle view, and also the default screen to greet the user when opening the app, is the camera view

US-02: Registration

As a user, I want to be able to register an account with the application, so that I can begin using the application

Acceptance criteria:
1. When the app is opened, if the user is not logged in a view is shown with the option to register or log in
2. A user must provide their full name, an email, a username, and a password to register.
3. There username must be one word and not contain any strange characters
4. There email must not already exist in the system
5. The username must not already be taken by another user
6. An unregistered user is unable to use the application
7. Upon successful registration, the user is able to fully use the application

US-03: Login

As a user, I want to be able to login to an existing account I have with the system, so that I can use the application.

Acceptance criteria:

1. When the app is opened, if the user is not logged in a view is shown with the option to register or log in
2. A user can provide either their email or username when logging in
3. The user must provide the correct password to log in successfully
4. Upon successful log in, the user is able to fully use the application
5. Log in information should be saved so that the user remains logged in even if they completely close the application
6. Use a local Room database to persist session data

US-04: Photo capture

As a user, I want to be able to capture a photo, so that I can send it to my friend when that functionality exists.

Acceptance criteria:

1. The camera view constantly shows a preview of what the camera is seeing, as in it is always read to take a photo
2. There is a circle at the bottom of the view that when pressed, will take a photo, and display it in the view
3. For the meantime, the user should be able to save the photo to their device, with the knowledge that, in the future, this will be replaced with the ability to send the photo to a friend.
4. The user should be able to switch between the front facing, and rear facing camera.
5. Photo button should animate when taking a picture

US-05: Adding friends

As a user, I want to be able to add my friends as connections, so that I can send them the photos I am capturing in the application.

Acceptance criteria:

1. An action bar should be present on the unopened snaps view with the option to add a friend
2. A user be able to search for users by their name or username
3. A user should have the option to add a user that comes up in the search results
4. Users do not have to accept friend requests, they are automatically accepted

US-06: Sending and viewing photos

As a user, once I have captured a photo, I should have the option of selecting friends to send the photo to, so that we can communicate and share memories.

Acceptance criteria:

1. A user can send a photo to as many friends as they like

2. When a user sends a photo to a friend, a new item appears in their unopened snaps view
3. The friend can open the photo they received to view it once
4. The friend cannot open the photo if they have already opened it
5. Photos should no longer be stored on the device

US-07: Location map

As a user, I should be able to see the location of my friends on a map, so that I know if they are okay.

Acceptance criteria:
1. A users location on the shared map should be updated when the app is opened, and when the app is closed
2. Location should be obtained using GPS (as opposed to IP address location)

US-08: Preferences

As a user, I should be able to access and set various preferences, so that I can tailor my experience with the app to my specific needs regarding privacy, accessibility and so on.

Acceptance criteria:
1. A user should be able to access a preferences menu from the action bar on the unopened snaps view
2. The preferences menu should contain options for language, location on map, text size, etc.
3. The preferences screen should use the AndroidX preference library

US-09: Notifications 1

As a user, I should receive some basic notifications from my app to keep me up to date, and remind me of various things.

Acceptance criteria:
1. The application should notify the user one per day reminding them to keep in touch with their friends

US-10: Translations

As a user, I should be able to view the application in a different language, so that I can continue to improve my Māori.

Acceptance criteria
1. User should be able to view application in Māori
2. User should be able to change language through preferences menu, as well as through the device as normal

Extra Stories:

EUS (Extra User Story) *ESS (Extra Spike Story)*

EUS-11: Notifications 2

As a user, I should receive more advanced notifications, so that I know when my friends have sent me photos to look at, or a new person has added me.

Acceptance criteria:

1. A user should receive a notification when a friend sends them a photo
2. A user should receive a notification when another user adds them as a friend
3. A user should receive a notification when their friend screenshots a photo that they have sent them

ESS-12: Play Store Submission

As a developer, I want to release my application on the Android Play Store, so that potential users can download the application, opening up opportunities for monetization.

Acceptance criteria:

1. Users can download the application on the Play store

EUS-13: Basic photo editing

As a user, I should be able to edit the photo I have just taken, so that I can convey more meaning to my friends.

Acceptance criteria:

1. User should be able to add text on top of a photo
2. User should be able to apply a colour based filter to a photo

EUS-14: Videos

As a user, I want to be able to send videos to my friends, to convey moments that can't be conveyed with a. single photo.

Acceptance criteria:

1. A user can send videos to their friends, just as they can send photos

EUS-15: Face mapping filters

As a user, I want to be able to apply a face filter, to make myself look silly for a laugh.

Acceptance criteria:

1. A user can select from a range of face filters and apply them to their face before taking a picture

EUS-16: Stories

As a user with a large following, I want to be able to post photos to my ‘story’, so that I do not have to send thousands of individual snapchats.

Acceptance criteria:

1. A user can send a photo to their story just like they can send it to their friends
2. A photo lasts on a story for 24 hours
3. Stories are shown on the unopened snaps view in their own portion of the screen