



Unleashed: A personal diary app for tracking walks with your dog

Final Report

CM3203 One Semester Individual Project
40 CREDITS

Amy Shields : c1624302

Supervised by: Dr Alia Abdelmoty
Moderated by: Dr Angelika Kimmig

Abstract

As part of human nature, we like to journal parts of our lives; such as activities that were undertaken that day, places we visited, how this made us feel and where we went. This can include aspects of our daily routines such as exercising and walking our dogs. There is currently not an easy method of tracking these in unison. With the rise of smartphones and mobile technologies this provides a portable and accurate platform to make the experience of tracking our moods alongside our dog walks easier. The aim of this project is to provide a way for users to track their moods on their dog walks in the form of a mobile application.

This project will detail the background research, design, implementation, testing and evaluation of a walking and mood tracking system built into an Android application. A user requirement questionnaire was conducted in order to ensure the application provides relevant functionality to possible end users. Related user test cases were conducted and evaluated. This has assessed the strengths and weaknesses of the project relating them to the original requirements.

Acknowledgements

I would like to thank my supervisor Alia Abeldmoty. Her insightful guidance and continued support throughout the entirety of the project has been unprecedented and was crucial for the progress and success of this project.

I would also like to thank my family for their support and patience throughout this project. Stepping in as volunteers for user testing and being of great help and paramount for the evaluation and development of this project.

Table of Contents

1. Introduction	7
2. Background	8
2.1. Existing Solutions	8
2.1.1. Location Tracking Applications.....	9
2.1.2. Electronic Diary Applications.....	11
2.1.3. Summary of existing solutions results	13
2.2. User Research - Questionnaire	14
2.3. User Research - Emotion.....	15
2.4. User Research - Location	15
2.5. User Research – Personas.....	16
2.6. Project Justification.....	18
2.7. Constraints	18
3. Approach.....	19
3.1. Agile Development Methodology	19
3.2. Learning	20
3.2.1. Android Development Courses & Guides	20
3.2.2. Location	20
3.2.3. Data Storage.....	21
4. Specification.....	22
4.1. Functional Requirements.....	22
4.1.1. Must have	22
4.1.2. Should have.....	23
4.1.3. Could have	23
4.2. Non Functional Requirements	24
4.2.1. Must have	24
4.2.2. Should have.....	25
4.3. Use Cases	25
4.3.1. Use Case Diagram	25
4.3.2. Use Cases	26
5. Design	36
5.1. UML Class Diagram	36
5.1.1. Overview of UML diagram.....	37

5.1.2.	<i>Use Case to UML Mapping</i>	38
5.2.	<i>User Interface Prototypes</i>	40
5.2.1.	<i>Developing for Android</i>	40
5.2.3.	<i>Heuristic Evaluation on Initial prototype</i>	50
5.2.4.	<i>Improved prototype</i>	53
6.	<i>Implementation</i>	58
6.1.	<i>Database design</i>	58
6.2.	<i>Activity Diagrams</i>	64
6.2.1.	<i>User Login actions</i>	64
6.2.2.	<i>Track a walk user actions</i>	65
6.2.3.	<i>Search/filter past walks</i>	65
6.3.	<i>Overview of quality of Design and Implementation</i>	66
6.3.1.	<i>Best practices for Android Development</i>	66
6.4.	<i>Detailed Implementation</i>	68
6.4.1.	<i>Overview of Implementation Structure</i>	68
6.4.2.	<i>User Login, Register & Forgot Password</i>	69
6.4.3.	<i>User Navigation</i>	70
6.4.4.	<i>Adding a mood to a walk</i>	71
6.4.5.	<i>Adding an activity to a walk</i>	72
6.4.6.	<i>Adding an image to a walk</i>	73
6.4.7.	<i>User Location</i>	74
6.4.8.	<i>Showing User's Location on the map</i>	75
6.4.9.	<i>Showing User's added walk elements on the map</i>	75
6.4.10.	<i>Adding a place to a walk</i>	76
6.4.11.	<i>Saving a User's walk</i>	77
6.4.12.	<i>Viewing past walks</i>	77
6.4.13.	<i>Viewing an Individual Walk</i>	79
6.4.14.	<i>Sharing a walk</i>	79
6.4.15.	<i>User's Profile and Statistics</i>	80
6.4.16.	<i>User Help</i>	82
7.	<i>Results</i>	83
7.1.	<i>Test Cases</i>	83
7.1.1.	<i>Acceptance Criteria Results</i>	83
7.2.	<i>Usability Testing</i>	85

7.2.1.	<i>User Tasks</i>	86
7.2.2.	<i>Usability Results Summary</i>	87
8.	<i>Conclusion</i>	89
9.	<i>Future Work</i>	91
10.	<i>Reflection</i>	93
11.	<i>References</i>	95

1. Introduction

The aim of this project is to create a solution which allows dogs walkers to have a simple way of tracking their walks and mood. It intends to be used by regular dog walkers who want more of an insight into their walks, what they did on the walks and how they felt during these walks. The motivation behind this application is to allow users to have an electronic record of their walks which they can interact with being able to discover personal insights into when and where they feel certain moods with their dog.

The main objectives are for the user to be able to:

- Track and log walks that are taken with their dog.
- Track specific details of each walk, such as: route taken, time taken, distance covered.
- Journal the individual's emotions and how the walk made them feel (start, during and end of walk)
- Add details to each walk, such as any places which they visited, and activities performed. E.g. fetch, training
- Attach pictures of the walk to each walk
- Filter/Search past walks by mood/time
- View personalised statistics of users walks

Each walk is intended to be like a 'diary' entry as such which would show the route, moods recorded, activities recorded, places and images. These walks will be saved and easily searched and filtered.

The application intends to be a user friendly, intuitive experience which is easy to use and interact with data. Effective user design principles and Android material design guides will be used in order to create a design which will be used as a basis for implementation.

This report will include all surrounding information regarding the design, implementation, testing and evaluation of this dog walking application.

2. Background

2.1. Existing Solutions

Prior to starting this project, I decided it would be extremely useful to evaluate similar solutions to the problem already. In particular, I wanted to focus on the most popular solutions in order to gauge what makes users want to use these applications. As I intend to create a mobile application for Android, I selected the best rated applications on the Google Play store.

As the problem is multi fragmented with two elements to address, tracking walks (in particular with your dog) and tracking moods, it was difficult to find an exact application which directly addresses the problem. Therefore, I researched into two types of applications: Fitness Applications and Electronic Diary Applications.

I selected several of the most downloaded or highly rated applications and performed an analysis on these. This included a trial use of the application in which I performed common features which are useful and relevant to this project, such as:

- Tracking a walk
- Viewing past walks
- Recording my mood
- Viewing past moods

The results of the investigation are summarised below. Please note the reviews and ratings are taken from the Google Play Store and all information was correct at the time of the investigation.

2.1.1. Location Tracking Applications

Application: MapMyWalk	Rating: 4.5/5 (190k reviews)	Free (in app purchases)
Summary: <p>This is the most recommended walk tracking app advertised on the app store. 40 million members in the community. It is connected to a large health community linked with Under Armour. It uses a simple interface to track walks as well as logging these to share with friends in a community based environment.</p> <p>Free version of the application allows for all features mentioned below as well as additionally you would have personalised training plans, the app would be ad free and live tracking to be shared with your family and friends. The paid subscription is £4.99 a month.</p>		
Features: <ul style="list-style-type: none">○ Track and Map each walk○ Audio feedback throughout the walk○ Routes feature in order to create and discover new routes○ Linking with wearable tech○ Add images/text to each walk○ Analysis of previous walks through a dashboard○ Analyse performance with pace distance, duration, calorie burn○ Challenges		
Reviews: <p>‘Simple to use’</p> <p>‘I wanted an app which would show me where I have walked rather than setting up a route in advance’</p> <p>‘I wish there was a little more map detail’</p> <p>‘I love the challenges feature. It keeps me on track’</p>		
Experience using the app: <p>When first using the application, there were not masses of help features or guidance on how to use the application. The user figures this out through using the application. The application has a minimalist feel to it, with the map and tracking being the sole purpose of the application.</p> <p>Mostly all features are available on the free version of the application, however ads can be visually disruptive and may put users off. The ability to export the workout is only available to paid users as well as mobile coaching.</p> <p>The app allows for automatic tracking of walks as well as a manual entry. However, the manual entry of walk feature is harder to find and hidden behind several menus.</p> <p>There is a strong community feel to the application, with an explore page to look at routes, workouts and motivational images. It also includes a friend’s tab where you can view friends walks and comment/like and share these.</p> <p>The dashboard feature allows for a view of walks for each week/month/year. It also shows the total amount of time walked, calories burnt and walks all together.</p>		
Available at: https://play.google.com/store/apps/details?id=com.mapmywalk.android2&hl=en_GB		

Application: Tractive GPS for Dogs	Rating: 4.3/5 (13k reviews)	Free (in app purchases)
<p>Summary:</p> <p>This is an application which allows for the tracking of your dog, cat or any other animal in real time. However, it also allows for an interactive map in which you can track walks. Tracking walks is a smaller feature on this app as it is usually used for real time tracking.</p> <p>The application itself is free and the 'walking' feature of the application is included in the free version. The application is paired with a wearable collar which is a one off purchase and once connected with the application, users have to pay a monthly fee of £6.99 a month.</p>		
<p>Features:</p> <ul style="list-style-type: none"> ○ Track and Map each walk ○ Add images/text to each walk ○ Share walk with friends and suggest routes ○ Analysis of previous walks through a dashboard – these can be split into lazy, active, dynamic walks ○ Tracking of sleeping behaviour for the dog ○ A virtual fence, if your dog goes beyond this you are alerted ○ Analyse performance with pace distance with dog 		
<p>Reviews:</p> <p>'Love the statistics!'</p> <p>'I would love a shortcut to track a walk immediately'</p> <p>'I like the integration with my dog, it feels personal'</p>		
<p>Experience using the app:</p> <p>The user interface is more cluttered than the usual applications. There is a heavy focus on real time live tracking of the dog, and this can be connected to wearables. The walking feature is not as prominent and quite hard to find. However once started, the user interface is clear and easy to use.</p> <p>There is a profile feature which is more focused on the dog and details of the dog. Icons are not as clear as to what they mean and can cause some ambiguity for the user. Within this app it is clear it is to be used in conjunction with their animal and dog, with a heavy focus on the dog such as the dog's name and breed being used throughout. Similar with MapMyWalk, it is hard to access help features and at times there is no exit from certain screens and not much easier user interaction.</p> <p>The viewing of past walks feature can be hard to find and not as clear as could be, with no sorting or filtering options at hand.</p>		
<p>Available at:</p> <p>https://play.google.com/store/apps/details?id=com.tractive.android.gps&hl=en_GB</p>		

2.1.2. Electronic Diary Applications

Application: Daylio	Rating: 4.7/5 (280k reviews)	Free (in app purchases)
<p>Summary:</p> <p>This is an application which allows you to keep a private journal through the use of emoticons and not actually typing a single line with. It is a two-step entry creation. It is through the use of icons and emojis as well as adding notes as an additional feature. The idea is it should be simple and quick to track your moods and emotions each day in order to analyse these over time.</p> <p>The paid subscription is £2.49 a month which provides additional features such as advanced statistics.</p>		
<p>Features:</p> <ul style="list-style-type: none"> ○ Personalised icons for moods and activities ○ Add images/text to each walk ○ Share statistics with friends ○ Analysis of moods and activities weekly. ○ Create goals ○ Rewards based system 		
<p>Reviews:</p> <p>‘Great easy way to track your life’</p> <p>‘So easy to use and quick’</p> <p>‘Brilliantly customisable’</p> <p>‘The use of emojis makes this so easy to use!’</p>		
<p>Experience using the app:</p> <p>The app is extremely minimalistic and easy to use, it is very intuitive and easy to use and therefore help features are not overly needed. However, some guidance on first time use would be useful.</p> <p>Search feature is useful and allows for easy retrieval for entries. The statistics feature is useful and provides an easy overview of achievements and mood count with goals. There is no way to add photos to each entry. The user must have the premium feature in order to export entries a, have a private pin lock and advanced statistics. To ask the user to pay for privacy is a large ask which is a feature most people would expect in the free version.</p>		
<p>Available at:</p> <p>https://play.google.com/store/apps/details?id=com.tractive.android.gps&hl=en_GB</p>		

Application: Reflectly	Rating: 4.5/5 (20k reviews)	Free (in app purchases)
<p>Summary:</p> <p>Reflectly is used to journal how you are feeling on a daily basis, but with the use of artificial intelligence to give users tools to deal with how they are feeling. It can be used to journal each day through the use of a very simple text and icon based interface. The app currently has 5 million users.</p> <p>Premium version of the application allows for personalised insights and unlimited text entries whereas the free service is just the basic journaling experience. The paid version starts from £9.9 a month.</p>		
<p>Features:</p> <ul style="list-style-type: none"> ○ A basic text diary entry scheme through the use of a simple clean interface ○ Uses a scaling method of 1-5 for certain parts of the entry such as how you are feeling ○ Uses icons as well as text in order to state what activities you completed that day ○ Allows for statistics to be shown (with a minimum of 6 entries) however personalised insights can only be accessed through the paid service. ○ Filtering based off activities/time range and feelings. 		
<p>Reviews:</p> <p>‘Helping me find joy and perspective’</p> <p>‘I love the activities feature, it reminds me what I’ve been up to!’</p> <p>‘Good app, but premium features needed in free version’</p> <p>‘Great app but too expensive’</p>		
<p>Experience using the app:</p> <p>The app takes a clear and guided tour of all features once starting up the application. Each diary entry is populated on the app through 3 clear screens. Each allowing for text or visual input. It has a feature in which it asks the user a profound question each day, such as:</p> <p>‘What was a difficult period of your life that made you stronger as a person?’ this allows for the app to gain insights to the person as a whole and give them personalised insights as to how to help their mental being.</p>		
<p>Available at:</p> <p>https://play.google.com/store/apps/details?id=com.reflectlyApp&hl=en_GB</p>		

2.1.3. Summary of existing solutions results

After trying the applications myself, and reading the reviews made by other users, I discovered several key features from both types of application to keep in mind when designing and implementing my application.

Fitness Applications (for the walk tracking):

- Make the walk feature itself the heart of the application. Users (and I) found the application harder to use if this was not an easy feature to find and use.
- Include statistics in order to review and look back on past walks.
- Make the application centre not only around the user, but their dog too. This could be through the use of the dog's name and it being a dog themed UI.

Electronic Diary Applications:

- Keep the diary entries short and easy to use through the use of icons in order to state their mood. Users reacted well to icons and emoticons in order to show how they feel.
- Users preferred to tag an emotion with emoticons rather than text based methods or using a quantitative scale such as 1-5. It is clear however that by using a limited number of emotions (displayed as emoticons) that it does not represent as accurate a mood as possibly a scale. However, using a quantitative scale could make the user interface messy.
- Include statistics in order to review and look back on past moods.
- Include an activities feature (through the use of icons or short words) for users to input what they have been doing that day (or on particular that walk).

Upon investigation I concluded that both MapMyWalk (Fitness Application) and Daylio (Electronic Diary) were most suited to my problem. MapMyWalk most closely encompasses the walk tracking section of the problem as it is a clear and easy way to track and visualise previous walks. The simplicity of the interface used, the visualisation of walks and its walk centered view is something I wish to take away and apply to the design of my android application. Daylio most closely encompasses the emotion tracking section of the problem. It's clear and simple two step format of retrieving mood alongside the visualisation of previous moods is to be considered during the design phase of my application.

I completed a Heuristic Evaluation of both applications (retrieving and viewing data) using Jakob Nielsen's 10 Usability Heuristics which can be viewed in Appendix: Heuristic Evaluation.

I will be using both the Existing Solutions findings in conjunction with the Heuristic Evaluation throughout the design and implementation of this project in order to make this application as user friendly and useful to the user as possible. I intend to use the scores derived from the Heuristic Evaluation in order to prioritise and implement similar UI elements/features into my application.

2.2. User Research - Questionnaire

In order to have a complete understanding of my intended end users, I met with a local dog walking group in Cardiff. I created a questionnaire which was composed of nine questions. These nine questions were created in order to understand how beneficial an application to track walks and emotion would be those who own a dog and walk this dog on a regular basis. It also was created in order to further refine and understand possible requirements of the system. The questionnaire and the results of the questionnaire can be found in Appendix 2: User Requirements Questionnaire.

Summary of Results:

I discovered from these findings that currently 50% of the respondents use a mobile application to track their walks. Out of the respondents who do not track their walks through a mobile application 75% said they would like to. In addition to this, currently only the 10% of respondents use a mobile application to track their mood, but 60% would like to start, realising the benefits of tracking these details. An interesting discovery was that 90% of the dog owners would like a dog walking app separate to a regular walking application.

This was a great insight into discovering what features would be most used by dog walkers. 100% of the dog walkers said they would find the following features most useful:

- Viewing a tracked route of a walk.
- Tracking how they felt during that walk through a range of emotions.
- Tracking activities taken on the walk.
- Viewing previous walks through different views of each day and week.
- Viewing previous moods related to each walk.
- A goals feature, setting up personalised goals for their dog.

Another finding was that 75% of the dog walkers preferred using a range of emoticons and icons to describe mood and activities taken opposed to a text based entry. Overall, 100% of the dog walkers said they would be more likely to go on dog walks and record emotive experiences through the use of an application like my proposed app.

I will be using the findings from this research when creating my Functional Requirements and throughout the design and development process. This will further enhance my application ensuring that it will be useful to potential end users bringing them functionalities they wish to use most.

2.3. User Research - Emotion

A large feature of this project will be the tracking and displaying of user's emotions. By including a user emotion tag within each walk, this will be able to remind the user how they were feeling on a particular day or on a particular walk. Previously, a user may have had to read through a physical diary in order to remember how they felt and what they were doing to feel this way.

The introduction of emotive tagging to each walk means that the user would have easy and quick retrieval to the emotion tagged itself, instead of using sentiment analysis on a bulk of text to figure out how they felt. Users prefer tagging particular emotions rather than a scale. It is hard, if not nearly impossible to record a range of emotions such as happy, anxious, sad on one qualitative scale. From reviews on the Google Play store and the results from the questionnaire it is clear that users prefer to use emoticons in order to portray their mood, this is mostly due to time reasoning as it takes the average user longer to type the word 'happy' instead of selecting the '😊' emoji which in the users eyes, portrays the same emotion.

Perceiving emotion on a scale is not simple and there is no standardised way of doing so. However, a large majority of users who use online diary applications prefer to use emojis to convey their mood. This idea is reinforced by the findings of research carried out which found that 'In recent years, text emoticons have been profoundly investigated and considered to be the way of conveying thoughts and feelings by simulating nonverbal signs in speech' (Hakami, 2017).

Individuals may have a different perception of what an emoji means however this does not matter to our project as it is a personal diary in which the individual user will create associations with particular emojis and moods. Researchers found that emoticons are useful in strengthening the intensity of a verbal message (Derks et al, 2007).

From investigating the best way to display and retrieve information, the application will be designed to provide the user with a simple, yet effective emoji based interface. It will cover five general emotions which are: 'Amazing', 'Happy', 'Neutral', 'Sad', 'Upset' and 'Horrible'.

2.4. User Research - Location

Using location within an application can offer a more contextual user experience. However, users primary concern to do with location tracking is privacy. Users want to know when the app is collating its location and when it is not. Users also wish to have peace of mind as to where their location data is stored and for how long.

Users of a walk tracking application will know that their location data will be stored and used at some point. However, it is critical to make the user aware of when it will be used. It is also critical to give the user freedom to deny the application access to their location or delete their location data at any given time.

Another user concern is the type of location data collected and if it is of purpose. For example, within Android development you can collect fine location or coarse location. Coarse location gathers less precise location data, useful for weather apps. Fine location gathers precise location data, useful for walking applications. There are trade-offs to consider with which location type to gather. Fine location can be slower to retrieve and can provide more security risks to the user. However, working with precise information is essential when gathering walk data as users will want a precise view of their walk data. Therefore, the application will be collecting fine location data.

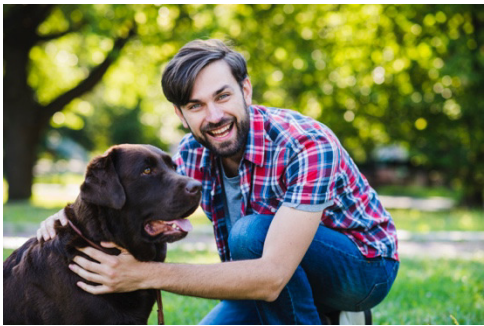
In terms of displaying location data, this is most frequently displayed on a map. The majority of existing applications make use of Google Maps API. This is used to display the current user's location on a clickable and interactive map, instead of text based location data such as your current city 'Cardiff' or latitude and longitude co-ordinates. This rule also applies to places. If the user visits a place, such as Principality Stadium, a marker would be displayed on the interactive map which can be clicked on at the location of the stadium. This is much more user friendly than text based data such as 'Principality Stadium'.

With users enjoying the experience of using an interactive map, in which they can move and pinpoint places/activities, I will consider this in the design of my application in order to adhere to standards and user's familiarity within similar applications.

2.5. User Research – Personas

From the information gained from the typical users of existing solutions, general research into users who track their walks and emotions and the user requirements feedback I have created two personas which are based on the types of users who may use a dog walk and emotion tracking application.

These personas are realistic as they have been derived mostly from the questionnaire (taken from a dog walking group). They represent the user's needs, wants and requirements for the application.

<p>Name: John Dunlop</p> <p>Age: 24</p> <p>Technical Competency:</p> <div data-bbox="209 1659 719 1709" style="border: 1px solid black; background-color: #90EE90; width: 20%;"></div> <p>Quote: "I want an easy way to track my walks with my dog!"</p>	
<p>Description: John walks his dog, Rover, several times a day. He is looking for an application to track his daily walks with Rover to be able to look back and see when/where he walked on certain days. He wants to be able to have a record of what he did with Rover on these walks in order to remember what he was doing on a given day. John currently doesn't use a form of a diary, but he is intrigued to do so, as long as it does</p>	

not take up much time out of his busy day. John wants his personal walk data to be protected and secure.

Goals:

1. Track walks in real time with his dog.
2. Track his mood alongside any activities which were undertaken in a simple manner which is time efficient.
3. View previous walks easily; viewing route, activities and his mood.
4. Have his personal details protected through the use of a login system.

Name: Anna McCabe

Age: 19

Technical Competency:



Quote:

"An app which can help me analyse how my mood changes with my walks would be great"



Description: Anna enjoys walking her dog daily but wants an application which increases her fitness levels by encouraging her to walk with her dog more and make these walks more exciting. Anna is particularly interested in using an application which helps her learn more about herself and her dogs' habits. She wants an easy app to use which shows her previous walks alongside her mood and locations she has been to during the walk.

Anna wants to be able to see how she is feeling in terms of when she is on a walk to possibly improve her mood and routine. Anna wants to be able to share her dog walks with friends.

Goals:

1. View previous walks easily; viewing route, activities and her mood and places been.
2. Determine average emotions for walks and find the places and activities performed where she is most happy, upset, anxious or neutral.
3. Become more excited about walking her dog through the use of the application and increasing her fitness levels by having more visualisation over her walks.
4. Share walks with friends.

2.6. Project Justification

Throughout performing thorough background research into existing applications and speaking with potential end users, it is clear that currently there are not many apps which integrate the walking of your dog alongside recording emotions as a form of an electronic diary.

NHS (2011) research found that 'dog owners are 34% more likely to hit exercise targets' and Sample, I (2009) details that writing about your feelings can leave you feeling happier [2], it is clear that an application which merge these two together could greatly benefit the end user. Users tend to want their personal information in one place, instead of using two apps to track this, this is where the proposed application would fit in.

In order for the application to succeed it would need to provide the user:

- An intuitive design. The application needs to be easy for the user to work with and enjoyable to use.
- The ability to access appropriate features in the application at any time. Users need to have full control over navigation throughout, ensuring any errors can be easily fixed.
- A clean minimalist design which is based off Android Development guides to ensure users are familiar with controls and icons. This will benefit the user to only view relevant information on each screen, avoiding clutter.
- Consistent feedback on the current state of the application. This will allow the user to know when their location is being tracked and when it is not.
- The ability to search, filter and sort past walks in a clear manner which allows the user to correlate walks alongside moods, activities and places.
- Flexibility in using the application. The user should be able to not only select from a pre-set list of activities and moods, but the possibility of adding their own.
- Encourage the user to walk their dog and enjoy the experience of a walk, turning itself into an enjoyable and fun experience which can be looked back on.

2.7. Constraints

There are several constraints which may possibly affected the outcome of this project. I have discussed these in detail alongside methods taken in order to minimise the impact of these.

The most prevalent constraint was the ratio of the scale of this project to the time to do complete it. I had not undertaken a project of this scale before solely or properly developed an Android application from scratch to a working application before. The total timescale of this project is 15 weeks (including a 1 week extension due to illness, making it 16 weeks). It is important to thoroughly plan and set small milestones to ensure a basic functioning system is submitted by the deadline. I used Trello in order to ensure I keep on track with tasks and have milestones set in place so that I had the best opportunity of completing this project to a high standard.

The second constraint is that I had never created an application for the Android platform before. This meant that before the research or design can begin for the implementation, I had to learn the basics of Android application development through the use of Udemy courses. This gave me an insight into what I would be capable to do, helping to make realistic requirements.

The third constraint is the impact in which Covid-19 has had on this project. Unfortunately, the outbreak of the virus was disruptive to the development and design process. I had to move back home, and this limited my resources from which I could obtain at university. This also impacted user usability testing. I had previously intended to perform usability testing on a local dog walking group, however due to government laws regarding social distancing this was not feasible. Usability testing had to be performed on a smaller scale.

3. Approach

3.1. Agile Development Methodology

In order to account for my lack of experience when creating Android applications, this project will follow an Agile development methodology. The agile methodology will follow an iterative approach to delivering the project on time. Due to the lack of experience, the iterative approach will give me the flexibility to learn and develop at the same time. It will allow the room for additional research into topics and issues when needed.

I will be splitting the project into small sections of work, with each section having a milestone and an expected completion date. By doing this, it will make the project more manageable and ensure that I can deliver the project on time, meeting the expected requirements. The milestones have been set with the intention of firstly creating a MVP (Minimum Viable Product), it is my intention to firstly create an application in which all main functionality is working, and then once that is created, decide on any additional features which could be added. This will ensure that the project is completed on time and all essential requirements and their acceptance criteria are met.

I will take part in regular reviews not only with myself and my milestones but with my supervisor. This will be to ensure that the project is on track and to address any issues if they arise.

3.2. Learning

3.2.1. Android Development Courses & Guides

Selecting a project to develop an Android application from scratch, something I have never done before, seemed like an excellent and intriguing learning opportunity. I decided to undertake this as learning to develop a mobile application is something I have been wanting to do for a while. I specifically chose android development due to several reasons. One being that Android applications can be developed in Java. This is a development language I have several years' experience with, opposed to Swift for iOS. Android is also open source, meaning that the application developed can be better customised. The final reason is that there are many online tutorials on gathering the basics of Android Development using Android Studio.

Due to the tight time schedule I knew I needed to research into Android development and learn quickly yet effectively in order to undertake this project and complete it to a high standard. I completed 'The Complete Android N Developer Course' on Udemy (Udemy 2019). This course gave me a grasp on the basic workings of creating an Android application in Android Studio. The course covered essential parts of the development process, such as using Gradle to import dependencies, activities and how they communicate, xml layout inner workings and requesting permissions. It also provided me with a walkthrough of making an Uber clone application in which parts were relevant to my project; such as location tracking.

Another course in which I followed was a YouTube series 'Simple Login App Tutorial Using Android Studio 2.3.3' (Professor DK, 2017). I found this series alongside the Udemy course to be invaluable to my basic learning of Android application development.

Throughout the project I made use of the Android Development Documentation (Android Developers, 2020). This is the official documentation for Android development and goes into thorough details of each aspect of development. It also provides example code, alongside useful and detailed explanations of what each method is doing and why. An example of where I used this heavily was setting up the location tracking within my application. The official Android Development Documentation provide free courses for developing applications of all standards. I did not become aware of this until much later in the development process. I believe this would have been a better place to start and obtain information opposed to Udemy as it is written and created by working Android employees.

3.2.2. Location

One of the main features of the application is going to be location tracking. This will be location tracking in real time. In order for the application to be able to track the user's location, it will make use of the Google Location Services API (also known as FusedLocationProviderAPI). This is Google's recommended method of obtaining location due to it having high accuracy and its power efficiency on your application. This will be used

in order to retrieve frequent updates of the user's location once granted permissions. The application will be acquiring location data using their GPS provider. This determines the location using satellites and is the most accurate form of location retrieval, opposed to acquiring from their network provider.

In order to visualise the location data, the application will be making use of the Google Maps API. This will be able to display live location data and include all of the useful and familiar features to users that Google Maps has to offer, including markers to pinpoint places and making the map interactive through touch.

The application will be set up so that it is only collecting locational data when on the app. This could be reconfigured and changed, but for security purposes for collecting current location data this will initially be how the application is set up. The application can be easily configured to suit the user's needs, depending on how often it is necessary to retrieve location updates from the user. For example, the API built in method, `setFastestInterval()` could be configured to 5 seconds or 10 seconds.

The Google Maps API has many advanced features and can interact with other API's which could be used to enhance the application. One of which that could be useful within the walk tracking application is the Places API. This would return details about places which are pin pointed by the user on the map. These details can include: user reviews, photos and details such as addresses. This could be an enhancement, time permitting, to enrich the users experience when logging a place on their walk.

3.2.3. Data Storage

In order to store all locational and emotional data for a particular user, it was important to consider the best type of data storage method to select. A primary consideration of mine was the robustness of my data storage method. It needed to be stable and reliable to support the storing and retrieval of the different types of data.

I firstly had to decide between using a NoSQL database or a relational SQL style database. Due to the quick retrieval time needed for this real time application and the flexible nature of the incoming walk data, I decided it was best to use a NoSQL database for this application.

After using MongoDB in my third year of University this seemed like the best choice, however after researching into previous projects which used MongoDB I noticed that they had issues storing images to entries. This is a feature I wanted my application to provide. The Android Development course on Udemy was making use of Google's Firebase as their database. Firebase is a mobile platform that helps developer develop applications. Firebase has a plethora of detailed documentation to help set up and use their services. It has a very easy to use UI to navigate and is cloud hosted. This brings pros and cons as Google is responsible for holding the data, however this makes it easier to set up and develop with. I decided to use Firebase, in particular Cloud Firestore in order to store my data in their

NoSQL database. Firestore is Firebase's cloud hosted, NoSQL database (Firebase Firestore, 2020).

Firebase provides a suite of attractive inbuilt features which made this an easy choice. They provide an easy to set up Authentication method for users on the application, crash reporting to keep the application stable, storage making file storing such as images easy and even a machine learning kit to help your application grow (Firebase, 2020).

4. Specification

4.1. Functional Requirements

4.1.1. Must have

- **FR1:** The system will provide the user its own secure space which only they can access.

Acceptance Criteria:

- User can log in to the system via a log in system with their email and password.
- If the user has not got a profile with the system, the system will provide the user with a method of signing up.
- If the user forgets their password, the system will provide the user with a method of resetting their password.

- **FR2:** The system will allow the user to track and store a walk in real time.

Acceptance Criteria:

- User can start a walk and track their movements in real time
- User can stop the walk
- The walk is stored in a database

- **FR3:** During this walk, the user can optionally include an emotion tag

Acceptance Criteria:

- Before, during and after the walk, the user will have the option to add an emotion tag to the entry. The user will have the option of selecting an emotion tag from a list of pre-set emotion options.

- **FR4:** During this walk, the user can optionally include activity tags

Acceptance Criteria:

- During the walk, the user will have the option to additionally add activities to the entry. The user will have the option of selecting an activity tag from a list of pre-set activity options.

- **FR5:** During this walk, the user can optionally include place tags

Acceptance Criteria:

- During the walk, the user will have the option to additionally add 'places' to the entry. The user will be able to select a place on the map and add a marker will be shown on the map to indicate where they have been.

- **FR6:** During this walk, the user can optionally include a photograph
Acceptance Criteria:
 - During the walk, the user will have the option to additionally add a photograph to the entry. The user will have the option of selecting a photo from their camera roll and attaching this onto the walk.
- **FR7:** The user can view past walk entries
Acceptance Criteria:
 - The user will have the option to view entries of past walks. The user will be able to sort these walk entries by chronological order.
 - The user will have the ability to filter these walks by time.
- **FR8:** The user can view an individual past walk
Acceptance Criteria:
 - The user will have the ability to view an individual past walk. The user will be able to see the route taken, time taken, moods, activities, photograph and places recorded for that walk.
- **FR9:** The user can delete a walk
Acceptance Criteria:
 - The user will have the ability to delete a previous walk.
- **FR10:** The user can view statistics regarding their walks
Acceptance Criteria:
 - The user will have the ability to view statistics, e.g. number of walks, distance covered, average mood. These statistics are subject to change and will be defined during the implementation phase.

4.1.2. Should have

- **FR11:** The user can perform search analysis of the walk entries, displaying emotions that match the provided emojis.
Acceptance Criteria:
 - The user will have the ability to search through past walk entries by selecting the appropriate emoji. The user can search for emotions recorded at the start/on/end of the walk.
- **FR12:** The user can delete their profile, including all location data
Acceptance Criteria:
 - The user will have the ability to delete their profile,. This will delete all personal and location data.

4.1.3. Could have

- **FR13:** The user can share details of a walk with their friends via email/message.
Acceptance Criteria:
 - The user will have the ability to share details of a walk with their friends.

- Friends with or without having the app installed will be able to view the data.
- **FR14:** The user can edit their personal details
Acceptance Criteria:
 - The user will have the ability to edit their personal details such as their name and their dogs name.
- **FR15:** The user will receive notifications to let them know once they have hit their pre inputted target distance for the day
Acceptance Criteria:
 - The user will receive a notification once they have walked the number of miles they have set for themselves to hit for each day.

4.2. Non Functional Requirements

4.2.1. Must have

- **NFR1:** The system will immediately save walk entries into the database
Acceptance Criteria:
 - Once a new walk entry is created, the walk entry is stored in the database once submitted.
- **NFR2:** The user will always have accessibility to their walk entries
Acceptance Criteria:
 - The user will be able to view all of their past walk data to view or delete.
 - If the connectivity to the database is not secure, the user will receive notification of this through a pop up message.
- **NFR3:** The system will be intuitive and easy to use
Acceptance Criteria:
 - The user should be able to navigate the application smoothly and have clean interfaces and icons which are recognisable.
 - The UI will be consistent and appropriate for its purpose.
- **NFR4:** The system will provide a help feature
Acceptance Criteria:
 - The user, if struggling to use the application, will be able to access a help feature which explains the application and how to use it.
- **NFR5:** The system will be reliable
Acceptance Criteria:
 - The application will be able to be ran on 100% of currently supported Android versions.
 - Recycler views will generate contents in no longer than 5 seconds.
 - The application will not crash due to unexpected errors.

4.2.2. Should have

- **NFR6:** The application will allow for future developments

Acceptance Criteria:

- The development of this project will be thoroughly documented in the Implementation section of this report.
- Code will be thoroughly commented and designed in a repeatable way.

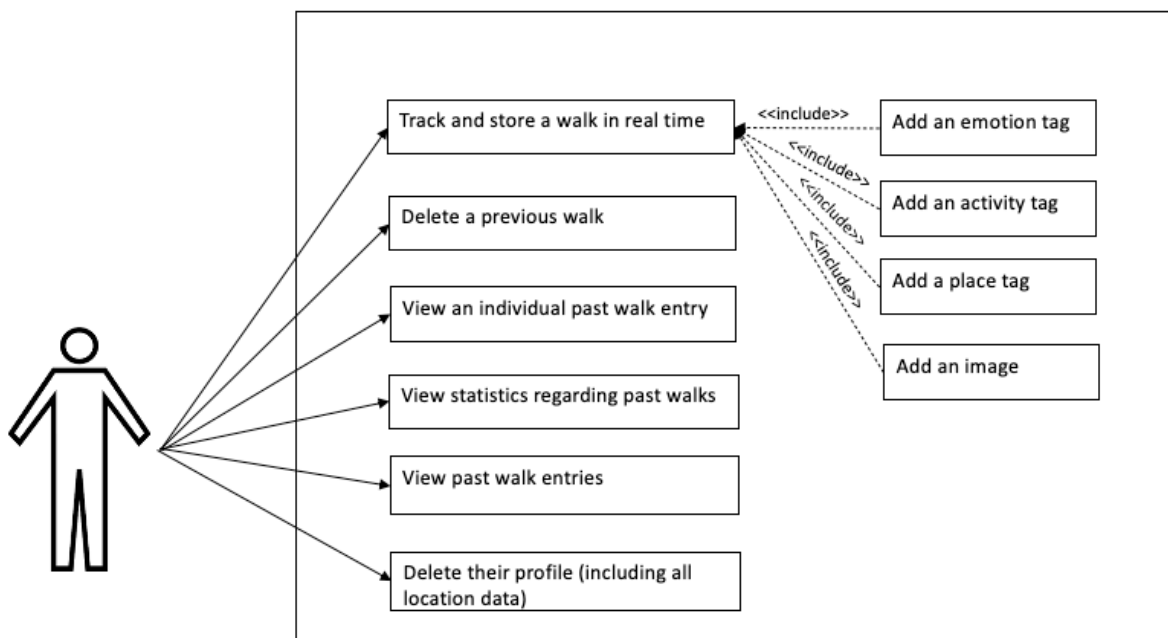
4.3. Use Cases

4.3.1. Use Case Diagram

The following diagram shows the use cases for the system. They are mostly based off of the 'Must have' functional requirements which have been derived from background and user research. The end user is linked to each of these use cases and the diagram shows how the functionality is linked.

Included within the 'Track and store a walk in real time' are the following four optional tasks which the user can perform if they wish to do so:

1. Add an emotion tag (start, during and/or end of walk): one at the start, one during and one at the end.
2. Add an activity tag (during the walk): as many activities as the users wishes.
3. Add a place tag (during the walk): as many places as the users wishes.
4. Add an image (during the walk): one per walk.

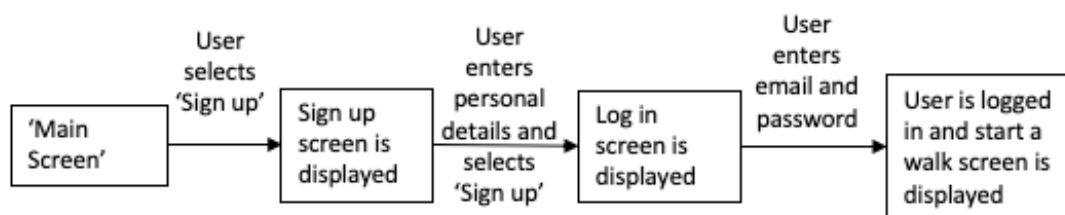


4.3.2. Use Cases

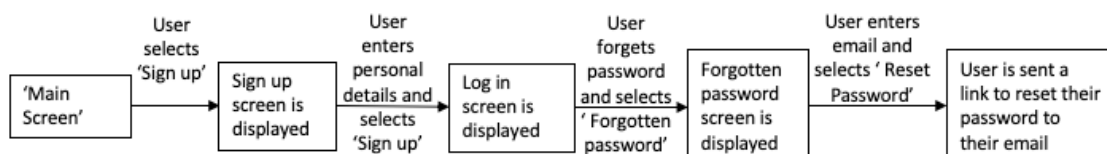
In the following section, I will discuss each use case of the system in detail. I will outline its basic flow, alternative flow and include STN's (State Transition Network) to visualise how the system will work.

Use Case No: UC1 Users must have their own secure space which is password protected.		Type: Must have
Goal:	User can log in to the system via a log in system with their email and password. If the user has not got a profile with the system, the system will provide the user with a method of signing up. If the user forgets their password, the system will provide the user with a method of resetting their password.	
Pre-conditions:	NA – this example will presume the user does not have a profile.	
Basic Flow:	<ol style="list-style-type: none"> 1. User selects the 'Sign up' button 2. User enters their personal details: name, dogs name, email and password <ol style="list-style-type: none"> a. User selects the 'Sign up' button 3. User is directed to the Log in screen. <ol style="list-style-type: none"> a. User enters their email and password and selects the 'Log in' button b. User is logged into the system and is taken to the 'Start a walk' screen 	
Alternative Flow:	3a: User forgets their password. User is taken to 'Forgot password' screen where they can enter their email and get a password reset link.	
Related Use Cases:	All UCs require the user to be logged in.	

Basic Flow STN UC1:

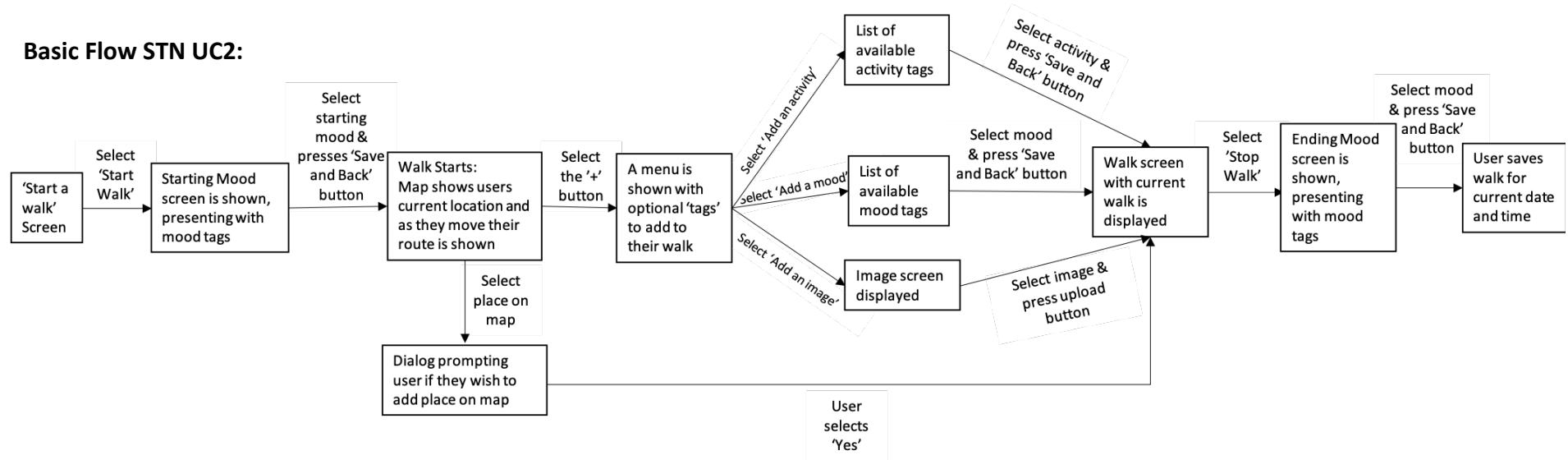


Alternative Flow STN UC1:

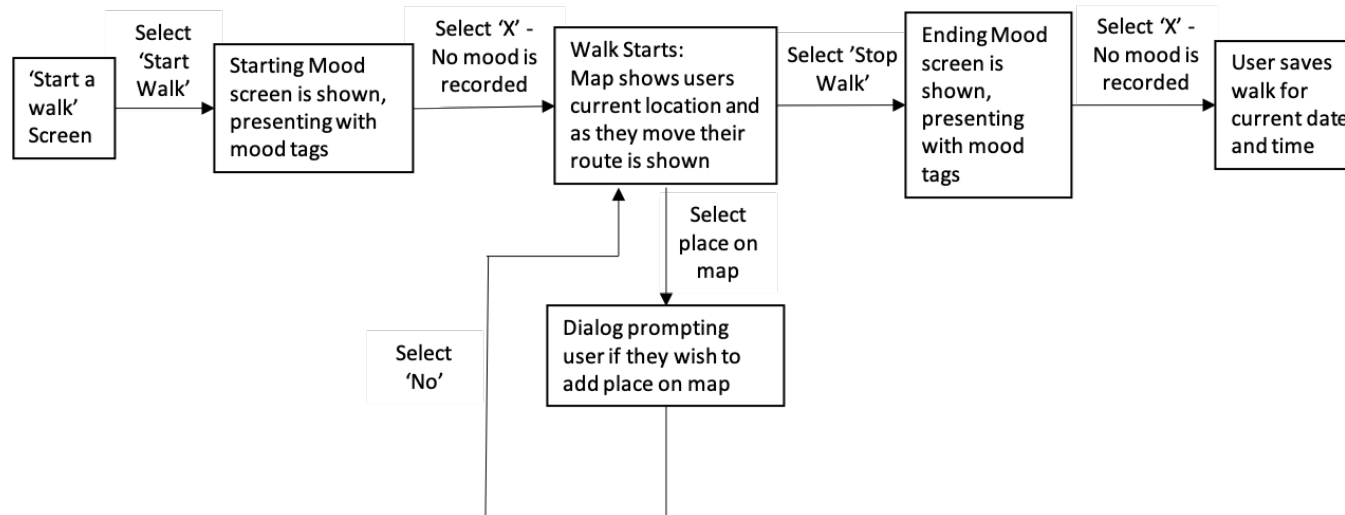


Use Case No: UC2		Type: Must have
Track and save a walk in real time and optionally add content to walk		
Goal:	The user is able to track their walk with their dog in real time. They will be able to see their movements and see the route they have taken so far. They optionally can add the following if they wish to do so: Mood at the start of the walk, mood during, activities during, places on a map, an image and mood at the end of the walk.	
Pre-conditions:	The user has logged into the system with their username and password and is on the 'Start a Walk' screen.	
Basic Flow:	<ol style="list-style-type: none"> 4. User selects the 'Start Walk' button 5. User selects their starting mood from the emotion images presented to them <ol style="list-style-type: none"> a. User selects the 'Save and Back' button 6. User selects the '+' button. <ol style="list-style-type: none"> a. User selects the 'Add an activity' button b. User selects an activity from the activity images presented to them c. User selects the 'Save and Back' button 7. User selects the '+' button. <ol style="list-style-type: none"> a. User selects the 'Add a mood button' b. User selects a mood from the emotion images presented to them c. User selects the 'Save and Back' button 8. User selects the '+' button. <ol style="list-style-type: none"> a. User selects the 'Add an image button' b. User selects the 'Select image' button and selects an image from their library c. User selects the 'Upload image' button and is returned to the main walk screen 9. User selects a place on the map <ol style="list-style-type: none"> a. User is presented with a dialog to ensure they want to add the place to their walk, user selects yes 10. User selects the 'Stop Walk button' 11. User selects their ending mood from the emotion images presented to them <ol style="list-style-type: none"> a. User selects the 'Save and Back' button 12. User selects 'Stop Walk' 13. User saves walk for the current date and time 	
Alternative Flow:	<p>2: User selects the 'X' button and does not record a starting mood.</p> <p>3c: User selects the 'X' button and does not record a starting mood.</p> <p>4c: User selects the 'X' button and does not record a current mood.</p> <p>5c: User selects the 'X' button and does not attach an image.</p> <p>6a: User selects 'No' and the place is not added to the walk.</p> <p>8a: User selects the 'X' button and does not record an ending mood.</p>	
Related Use Cases:	UC3, UC4, UC5, UC6, UC8, UC9, UC10 all require a walk to be made.	

Basic Flow STN UC2:

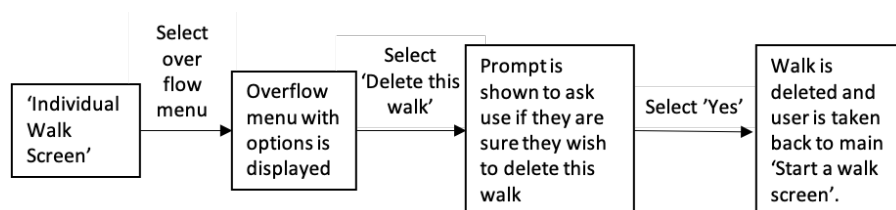


Alternative Flow STN UC2:

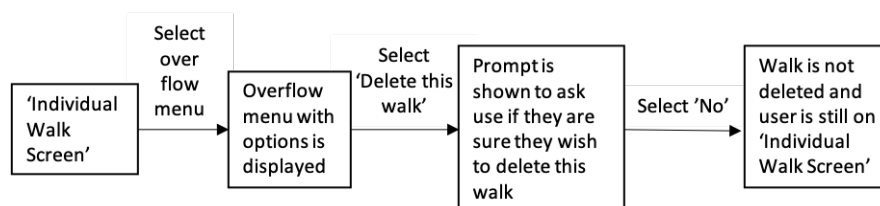


Use Case No: UC3 Delete a previous walk		Type: Must have
Goal:	The user is able to delete a previous walk they have taken. The walk/location data, mood data (if any), activity data (if any), and attached image (if one) will be deleted.	
Pre-conditions:	The user is on the 'Individual Walk' screen for the walk they wish to delete.	
Basic Flow:	<ol style="list-style-type: none"> 1. User selects the overflow menu in the toolbar 2. User selects the 'Delete this walk' option 3. User is prompted with a dialog to ensure they wish to delete that walk. <ol style="list-style-type: none"> a. User selects 'Yes' 4. User is taken back to the main 'Start a walk' screen. 	
Alternative Flow:	2: User selects 'No' and the walk is not deleted.	
Related Use Cases:	UC2 (User must have tracked the walk). UC4 (User must view the walk)	

Basic Flow STN UC3:

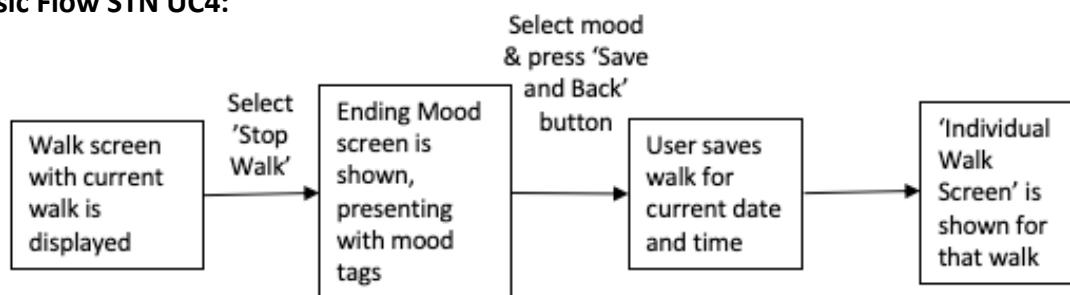


Alternative Flow STN UC3:

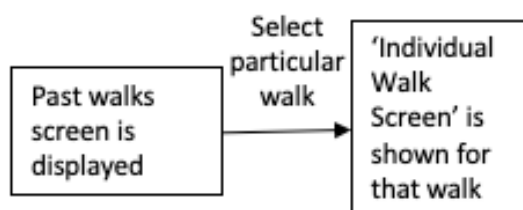


Use Case No: UC4 View an individual walk entry		Type: Must have
Goal:	The user is able to view an individual walk entry. This walk entry will display the route the user took, any moods, activities, places or images attached to the walk. It will also display the distance and time it took the user to walk.	
Pre-conditions:	The user must have completed and saved a walk they wish to view. The screen is accessible from two different screens: <ol style="list-style-type: none"> 1. The user has just completed a walk – after pressing ‘Stop Walk’ and optionally selecting an ending mood; the walk is saved, the user is taken to the individual view of the walk. 2. The user selects the walk to view from the list of past walks they have completed. 	
Basic Flow: *Pre-condition 1*	<ol style="list-style-type: none"> 1. User selects the ‘Stop Walk button’ 2. User selects their ending mood from the emotion images presented to them (optional) <ol style="list-style-type: none"> a. User selects the ‘Save and Back’ button 3. User selects ‘Stop Walk’ 4. User saves walk for the current date and time 5. User is presented with the Individual past walk screen, where they can view all walk details related to that walk. 	
Alternative Flow: *Pre-condition 2*	1: User browses past walks and selects the walk they wish to view by selecting the walk title (time and date).	
Related Use Cases:	UC2 (User must have tracked the walk). UC3 (User can delete the walk). UC6 (User can access from this screen).	

Basic Flow STN UC4:

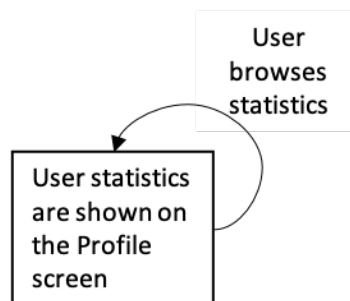


Alternative Flow STN UC4:



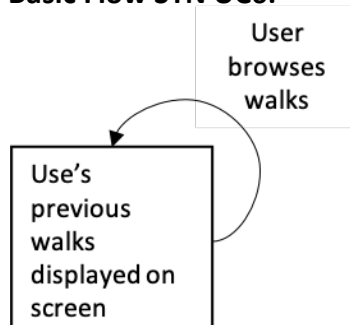
Use Case No: UC5 View statistics regarding past walks		Type: Must have
Goal:	The user is able to view their own personal statistics regarding their walks with their dog. These statistics will include: the number of walks, average distance walked, average starting/ending mood and most common activity.	
Pre-conditions:	The user must be on the 'Profile' screen.	
Basic Flow:	1. User views their personalised statistics on the profile screen.	
Alternative Flow:	NA	
Related Use Cases:	NA	

Basic Flow STN UC5:



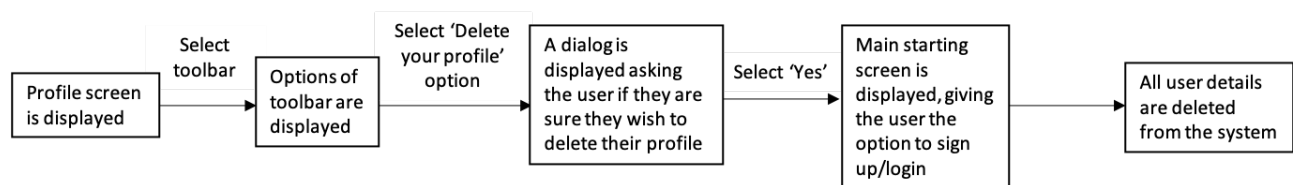
Use Case No: UC6 View past walk entries		Type: Must have
Goal:	The user is able to view all of their previous walks on one screen. The walks will be displayed in descending order. The user can see the time of the walk and the distance covered. The user can click on a walk and view its individual details (UC4).	
Pre-conditions:	The user must have saved at least one walk. User must be on the 'Past walks' screen.	
Basic Flow:	1. User views their past walks on the screen, with most recent walk first.	
Alternative Flow:	NA	
Related Use Cases:	UC2 (User must have tracked a walk).	

Basic Flow STN UC6:

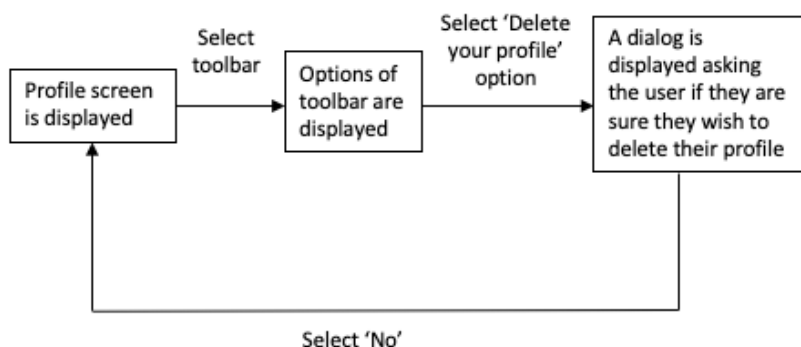


Use Case No: UC7 User can delete their profile (including all location data)		Type: Should have
Goal:	The user is able to delete their profile and account on the system. This will delete all details of the user including all location and personal data.	
Pre-conditions:	The user must have a profile/account on the system. The user must be on the 'Profile' screen.	
Basic Flow:	<ol style="list-style-type: none"> 1. User selects the overflow menu in the toolbar 2. User selects the 'Delete your profile' option 3. User is prompted with a dialog to ensure they wish to delete their profile and all data related to their account. <ol style="list-style-type: none"> a. User selects 'Yes' 4. User is taken back to the main screen allowing the user to sign up/log in. 5. All user details are deleted and removed from the system. 	
Alternative Flow:	3a: User selects 'No' and their account is not deleted.	
Related Use Cases:	NA	

Basic Flow STN UC7:

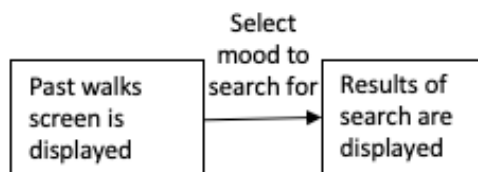


Alternative Flow STN UC7:



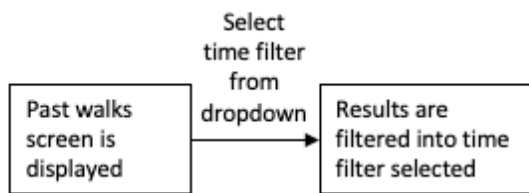
Use Case No: UC8 The user can perform search analysis of the walk entries, displaying emotions that match emoji selected		Type: Should have
Goal:	The user can perform a search of the walk entries, displaying walks that match the emoji selected. The searching/filtering will happen on the 'Past walks' screen, which displays all of the user's past walks. The results will show the walks which have the moods in the selected mood stored within them.	
Pre-conditions:	The user must have tracked and saved at least one walk. The user must have recorded moods on these walks. The user must be on the 'Past walks' screen.	
Basic Flow:	1. User selects the mood icon 2. Results of the search are displayed, which the user can review	
Alternative Flow:	NA	
Related Use Cases:	UC2 (User will need to have tracked a walk to search it). UC6 (User will need walks to search through, searching is done on this page).	

Basic Flow STN UC8:



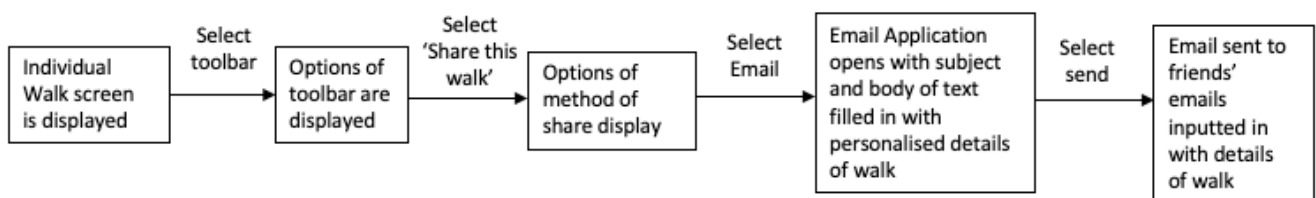
Use Case No: UC9 The user can perform filtering of the walk entries, based on time		Type: Should have
Goal:	The user can perform a filtering of the walk entries, displaying walks that match the time filter selected. The searching/filtering will happen on the 'Past walks' screen, which displays all of the user's past walks. The results will show all walks filtered and sorted into time filter selected. E.g. daily, weekly or monthly.	
Pre-conditions:	The user must have tracked and saved at least one walk. The user must have recorded moods on these walks. The user must be on the 'Past walks' screen.	
Basic Flow:	1. User selects the time filter from dropdown 2. Results of the search are displayed sorted by filter chosen, which the user can review	
Alternative Flow:	NA	
Related Use Cases:	UC2 (User will need to have tracked a walk to search it). UC6 (User will need walks to search through, searching is done on this page).	

Basic Flow STN UC9:



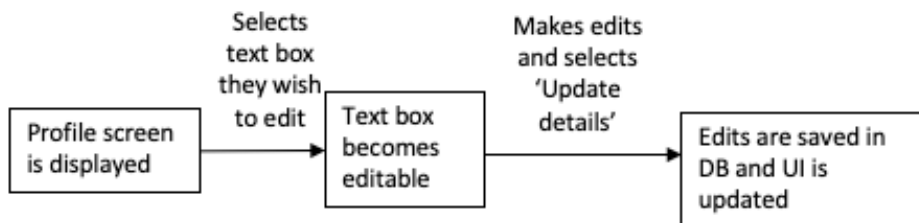
Use Case No: UC10 The user can share details of a walk with their friends via email/message		Type: Could have
Goal:	The user will have the ability to share details of a walk with their friends. Friends with or without having the app installed will be able to view the data. The details will include: walk time and distance.	
Pre-conditions:	The user must have tracked and saved at least one walk. The user must be on the 'Individual Walk' screen.	
Basic Flow:	<ol style="list-style-type: none"> 1. User selects the toolbar overflow menu 2. User selects 'Share this walk' option 3. User selects the method of which they wish to share the walk details. e.g. email / message. For this example it will be the email application. 4. The method app chosen opens with the details of the walk in the body of the text 	
Alternative Flow:	NA	
Related Use Cases:	UC2 (User will need to have tracked a walk to share it). UC4 (User will need to view an individual walk to share).	

Basic Flow STN UC10:



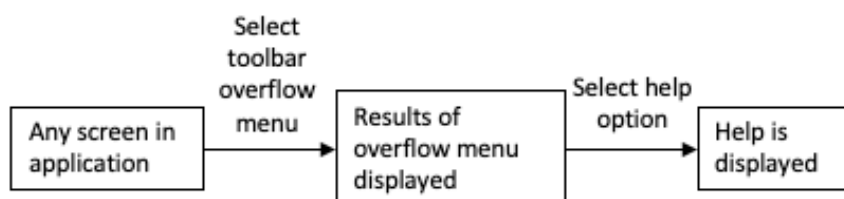
Use Case No: UC11		Type: Could have
The user can edit their personal details		
Goal:	The user will have the ability to edit their personal details such as their name and their dogs name.	
Pre-conditions:	The user must be on the Profile screen.	
Basic Flow:	<ol style="list-style-type: none"> 1. User selects the name textbox which is filled with their name/dog's name 2. User makes the edits they wish 3. User selects 'Update details' button 4. Users changes are made and saved in the database 	
Alternative Flow:	NA	
Related Use Cases:	NA	

Basic Flow STN UC11:



Use Case No: UC12		Type: Should have
The user can access help at any point in the application		
Goal:	The user will have the ability to access help throughout the application	
Pre-conditions:	User must be logged into the system	
Basic Flow:	<ol style="list-style-type: none"> 1. User selects the overflow menu in the toolbar 2. User selects help option 3. User views help section 	
Alternative Flow:	NA	
Related Use Cases:	NA	

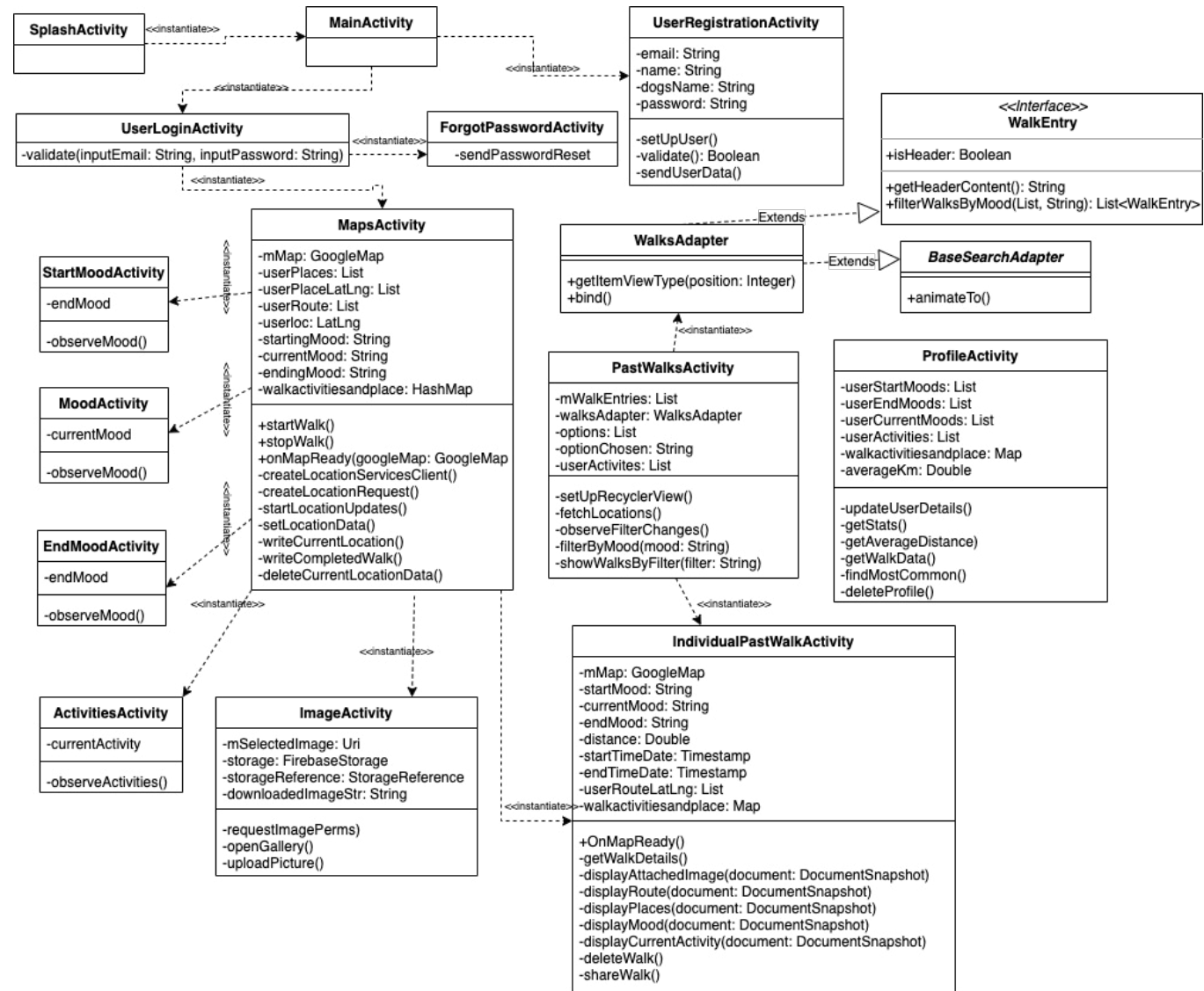
Basic Flow STN UC12:



5.1. UML Class Diagram

The diagram shows the basic structure of the main functional classes in the system.

Note: I have not included all variables and methods within each class in order to simplify the diagram.



5.1.1. Overview of UML diagram

The primary class for starting the user's walk is the **MapsActivity**. This is where the user tracks the walk itself and has access to add additional content to their walk. This is where the user can start and stop their walk.

Once the user starts a walk, the user has access to add additional content to their walk through the use of the following classes: **StartMoodActivity**, **MoodActivity**, **EndMoodActivity**, **ActivitiesActivity** and **ImageActivity**. These classes allow the user to add moods, activities and images to their walks.

All of the users saved walks can be viewed in **PastWalksActivity**. This class will list the users walks and allow the user to search and filter the walks. Past walks will be displayed using a RecyclerView, this is used to display data which elements change at runtime based on user action e.g. filtering. RecyclerView's require an adapter. An adapter's role is to convert an object (e.g. a walk) into a list row item to be inserted. PastWalksActivity uses a **WalksAdapter** class to do so. WalksAdapter extends **BaseSearchAdapter** which is responsible for handling and animating walks and removals of walks in the RecyclerView when filtering.

WalksAdapter also extends **WalkEntry**, this is to determine what type of object is being passed through the RecyclerView and is particularly used when filtering walks. As users must be able to filter walks, for example by Day, means that there will be two types of data passing into the RecyclerView to display, a walk and a header to show the day of filtering. A RecyclerView cannot determine which is a header or a walk without a Boolean check. A RecyclerView will just dump on the screen whatever is passed through it, therefore for formatting reasoning, there needs to be a clarification of what type of object has been passed through.

The following is not on the UML due to simplicity of diagram reasoning: BaseSearchAdapter extends two classes, **BinderViewHolder** and **BinderAdapter**. BinderViewHolder is used to pass/bind walks data to its view holder (used within WalksAdapter) and BinderAdapter provides the basic binding for the RecyclerView, holding a list of generic items in addition to a context (where should be viewed, i.e. PastWalksActivity).

The user can view a past walk in detail through **IndividualPastWalkActivity**. This will allow the user to share and delete a walk also.

Some functions are persistently accessible throughout the interface through the use of the Bottom Navigation Bar. These said functions are the opening of the following activities: PastWalksActivity, MapsActivity and ProfileActivity.

Within most main classes, there are uses of firebaseAuth, firebaseFirestore and firebaseStorage. For simplicity of viewing I have not included these in the UML. There is a `logoutUser()` method accessible from all main classes which logs out the user and takes them back to UserLoginActivity.

5.1.2. Use Case to UML Mapping

Below I will describe how each activity correlates to each use case.

SplashActivity:

- This does not necessarily relate to a Use Case, this is for general usability experience in order to make the starting of the application cohesive and smooth by adding a Splash screen.

MainActivity/UserLoginActivity/ForgotPasswordActivity/UserRegistrationActivity:

- Relates to **UC1**: In order for the user to have their own secure space on the application and Log in to the application.
- MainActivity acts as a landing page for the user to decide to log in or sign up
- UserLoginActivity handles user login
- UserRegistrationActivity handles creating a user login
- ForgotPasswordActivity handles forgotten passwords

MapsActivity/StartMoodActivity/MoodActivity/EndMoodActivity/ActivitiesActivity/ImageActivity:

- Relates to **UC2**: Track and save a walk in real time and optionally add content to walk.
- MapsActivity handles the tracking of the walk itself
- StartMoodActivity, MoodActivity, EndMoodActivity handle adding moods to the walk
- ActivitiesActivity handles adding an activity to the walk
- ImageActivity handles adding an image to the walk

IndividualPastWalkActivity:

- Relates to **UC3, UC4, UC10**
- **UC3**: This page allows the user to delete a past walk
- **UC4**: Allows the user to view an individual past walk
- **UC10**: Allows the user to share a past walk

PastWalksActivity/WalksAdapter/BaseSearchAdapter/WalkEntry:

- Relates to **UC6, UC8, UC9**
- **UC6**: Allows users to view past walk entries
- **UC8 + UC9**: Allows users to filter/search walks based on moods and time filters. WalkAdapter, BaseSearchAdapter and WalkEntry allow for effective searching and animating of RecyclerView

ProfileActivity:

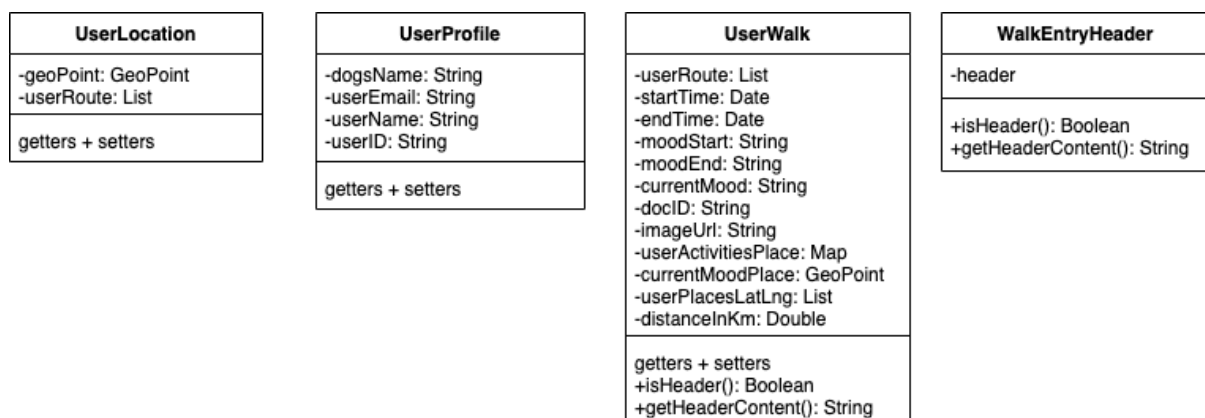
- Relates to **UC5, UC7, UC11**
- **UC5**: Allows users to view statistics regarding past walks (this is displayed on Profile page)
- **UC7**: Allows users to delete their profile
- **UC11**: Allows users to edit personal information

I opted for an object orientated approach to handle data throughout the system. The objects below encapsulate behaviours and data which are manipulated by the system throughout. These are POJO classes as they define how each object should look and allow for the manipulation (getting and setting) of fields within them. I also save the data into the database through the use of objects by saving the entire object into the relevant collection.

For example:

- An instance of UserWalk is created at the start of a walk. Throughout the walk its fields are set. For example, a start mood is selected and the setStartMood method is called on the walk instance. Once the walk is finished the UserWalk object is saved into UserWalk collection in Firebase.
- An instance of UserLocation is created at the start of a walk. Its two fields are continually set every 5 seconds and the UserLocation object is saved in the UserLocation collection in Firebase.
- An instance of UserProfile is created once a user signs up to the application. This allows the application to call methods on the instance of the profile such as getDogsName which will return the dog's name set.

WalkEntryHeader implements the WalkEntry interface (as shown in the UML), this implements the framework set out to determine if an object is a header when passing through the RecyclerView.



Note:

- WalkEntryHeader implements WalkEntry
- UserWalk implements WalkEntry

Also included are five 'Utils' classes. These are: ActivityUtils, DateUtils, HelpUtils, LocationUtils and MoodUtils. Included within these Util's classes are commonly used methods relevant to each task. For example:

- **DateUtils** includes formatting methods for dates which are used within all of the main classes.
- **LocationUtils** includes methods such as converting geopoints to LatLng objects. This is needed as Firebase works with GeoPoints, whereas the Google Location API uses LatLng objects.

- **ActivityUtils** and **MoodUtils** include common methods such as displaying relevant icons on the GoogleMap.
- **HelpUtils** is used for the Help dialog feature throughout the application.
- **MoodUtils** is used in order to set the mood/emotion image on screens in the system.

5.2. User Interface Prototypes

With application usability as a priority, it was essential to create mock-ups/a prototype of the application. This was to be able to visualise what the application would look like and how it would interact with the user. The following mock-ups were created with the intention to allow the user to interact with the application in order to complete and achieve the requirements set out. In order to ensure this was followed through, I kept the functional requirements, use cases and user personas in mind throughout.

I also aimed to apply several principles throughout my design. This included Nielsen's heuristics (Nielsen and Molich, 1990). The CARP principles were also applied. These are contrast, alignment, repetition and proximity. These are essential to create a 'easy to understand presentation of information' [MIT, 2018]. An example of this, is the repeated colour scheme and alignment strategies in order to make the user experience enjoyable and familiar.

I created the mock-ups using software called Balsamiq Cloud. Balsamiq cloud is a web-based user interface design tool for creating wireframes (sometimes called mock-ups or low-fidelity prototypes)' (Balsamiq, 2020). I decided to use this software as I was familiar with it from using it during my HCI coursework. The tool however does have some limitations with its variety of Android icons/tool sets, therefore the designs are missing some visual aspects which are to be implemented. However, I have ensured to explain any aspects which cannot be visualised through the software.

5.2.1. Developing for Android

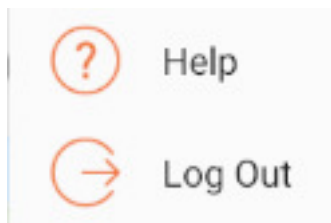
The UI (User Interface) has been designed with several android centric principles in mind. Firstly, I used the Android Developer Guide throughout (Android Developers, 2020). This provided me with general android features which are recommended to implement to make it familiar to the user. Examples of this include using buttons and icons in styles which have a universal meaning to the majority of users. This guide helped in reminding me to keep the user interface clean and simple to use.

I also used the Android Material Design standards throughout (Android Material Guide, 2020). I used this throughout which was useful in ensuring layout, navigation and iconography was in line with other android applications and supports the best practises of user interface design. This will reduce the learning curve in which the user needs to take to become familiar with the application and navigating throughout the system. The material design suggests limited colours within the application to two hues. This is something I have

implemented into the system, through the use of a soft grey and orange to highlight important features.

Example: Using Android Standard Icons throughout the application

This is to create a unified user interface throughout. As stated in the Android material design standards (Android Material Guide, 2020), 'Each icon is reduced to its minimal form, expressing essential characteristics'. Throughout the application basic and simple icons are used.



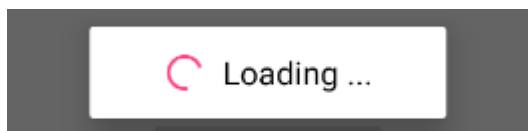
Example: Back button in navigation bar

As stated in the Android Develop guide, 'All Android devices provide a Back button for this type of navigation, so you should not add a Back button to your app's UI'. A back button has not been placed on the final interface when handling main functionalities.



Example: Progress Dialog

In order to adhere to Nielsen's Usability principle (Nielsen and Molich, 1990), 'Visibility of system status', a progress dialog which is used throughout android application development is used to ensure the user knows when a task is running/processing. An example of this is when the user is logging in.



5.2.2. Initial Prototype

I designed the initial prototype with the user personas, use cases and guidelines in mind. This was intended to be an MVP design. Therefore, it does not follow the use case STNs exactly. I then performed a heuristic evaluation and derived the final prototype which is completely based off use cases, personas and requirements. The final prototype follows the use case STNs exactly.

PTS No: 1



User log in to system.

This screen is where the user can log into the system. They do so by entering their email and their password into the edit text boxes and selecting log in.

The user can also sign up to the application by selecting the 'Sign up here' text which will take the user to PTS2. The user can also select 'Forgotten password' text to send a reset link to their email.

An image placeholder is on this screen. This will be the application logo, once designed.

The orange and grey colour tones will be repeated throughout. Orange will be used to highlight important features. In this case, the login button. This colour differentiation also adheres to the contrast principle in CARP.

All elements on the screen are aligned centrally, and language of features such as 'Log in' will be familiar to the user.

Use Case(s): UC1

PTS No: 2



User registration for the application.

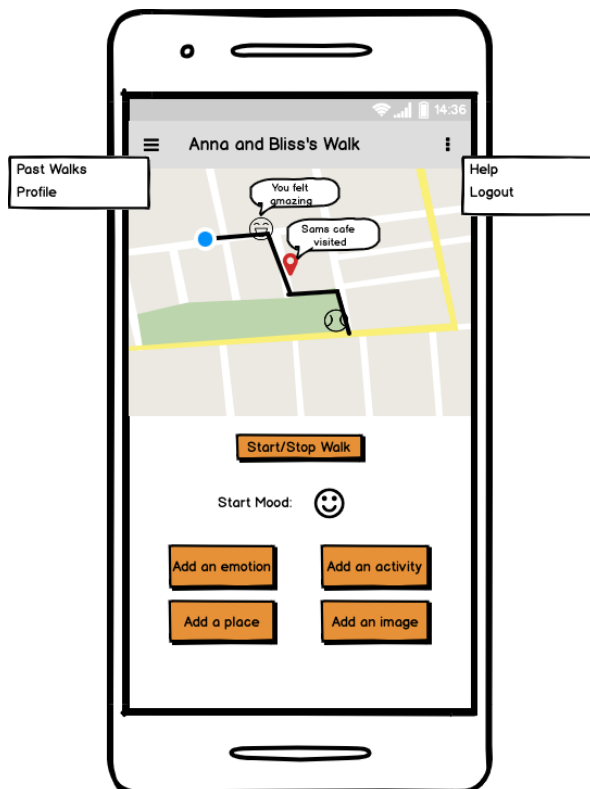
On this screen the user can sign up to the application. It prompts the user to enter their personal details into the edit text boxes.

The logo and colour scheme are consistent with the login screen. With the 'Sign up' button the same colour. Aligning with the repetition principle of CARP.

If the user forgets to enter any details and selects sign up, a toast message will be displayed at the bottom of the screen, prompting the user to enter all details correctly. As following with android material guide, they state toasts should be placed at the bottom of the UI and not in front of any app content.

Use Case(s): UC1

PTS No: 3



Track and store a walk in real time.


This is the main screen in the application. It allows the user to track a walk. The toolbar is personalised with the user's name and dogs name.

The user can navigate throughout the application through the use of the burger menu in the toolbar.

The map is interactive and is a google map. This allows for interactions such as selecting a place and then obtaining directions to that place on the map. These features will be familiar to the user.

The user would press the start button to start a walk (this would then change to stop) once the walk has started. A toast message would show at the bottom of the screen to let the user know the walk has started and once stopped, a toast message appears. This provides consistency.

	<p>The walk itself is like a live 'story'. The user moves, and their route is recorded. They can add emotions, places and activities to the walk and the corresponding icon will be displayed on the map. The corresponding screens in which they can add these are: PTS4, PTS5, PTS6 and PTS7. These icons once pressed will display a popup message with more details.</p> <p>The user can also add an image to the walk, this image will not be displayed on the actual walk screen once attached due to design reasoning.</p> <p>The users start mood is shown, this is obtained from PTS4. PTS4 also covers the screen which is displayed once the user selects the 'Add a mood' option and what is displayed once the user ends the walk.</p> <p>Use Case(s): UC2</p>
--	--

<p>PTS No: 4</p>  <p>Select your current mood:</p> <p>Select your end mood:</p>	<p>Optionally include an emotion tag.</p> <p>This is the screen in which users can select a starting mood. This is shown when the user selects start walk. The user can select the 'x' in the top right corner if they wish to not record a mood.</p> <p>The layout on this screen is consistent in the current mood and ending mood screen. This is to create repetition and make recording moods easy and familiar to the user. The icons were selected to provide the user with a range of moods to select. The user's selection is confirmed. The save button is consistent with the colour scheme.</p> <p>The user is directed back to the walk if they are selecting their starting or current mood. If they are recording current mood, an icon of chosen mood is placed at their current location. If they are recording end mood, they are directed to the recently saved walk.</p> <p>Use Case(s): UC2</p>
---	--

PTS No: 5



Optionally include activity tags.

This is the screen in which users can select an activity. The user can select the 'x' in the top right corner if they wish to not record.

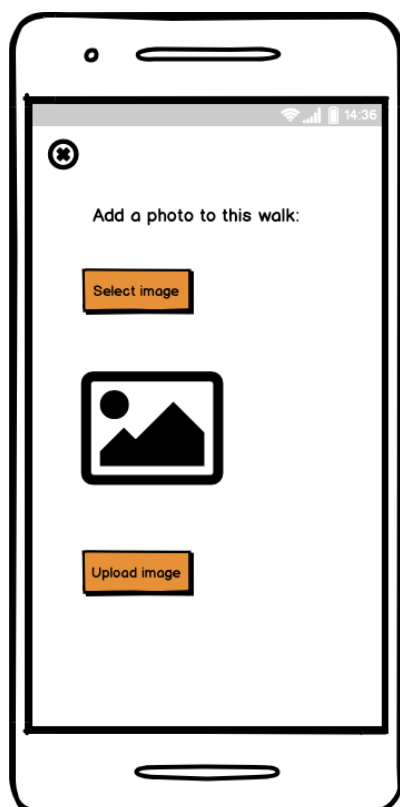
The layout on this screen is consistent in the mood screens. This is to create repetition and make recording details to each walk familiar to the user.

The icons were selected to provide the user with a range of activities to select. The user's selection is confirmed. The save button is consistent with the colour scheme.

Once activity is selected, an icon of chosen activity is placed at their current location.

Use Case(s): UC2

PTS No: 6



Optionally include a photo.

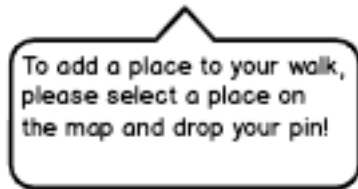
This is the screen in which users can select a photo to attach to the walk. The user can select the 'x' in the top right corner if they wish to not attach.

The user selects an image from their library, the image is previewed to the user and then the user decides to upload the image. This gives the user the chance to change the attached image.

The colour scheme is consistent with the rest of the application.

Use Case(s): UC2

PTS No: 7

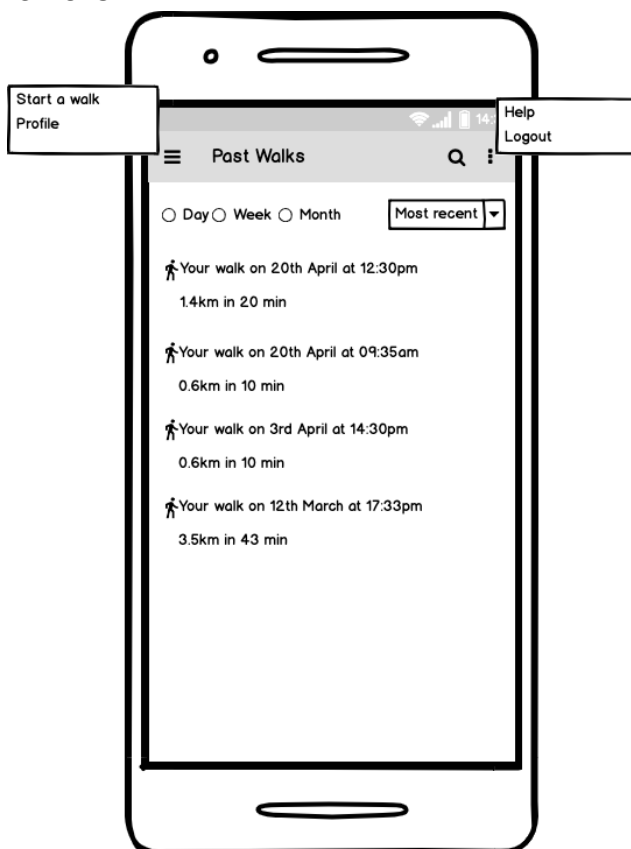


Optionally include places to the walk.

Due to the map being interactive, the best way to add a place is to select a place on the map and drop a pin. To some users this functionality will not be clear, so when a user selects the 'Add a place' button on PTS3, a toast will display providing guidance on how to do so.

Use Case(s): UC2

PTS No: 8



View Past Walk entries.

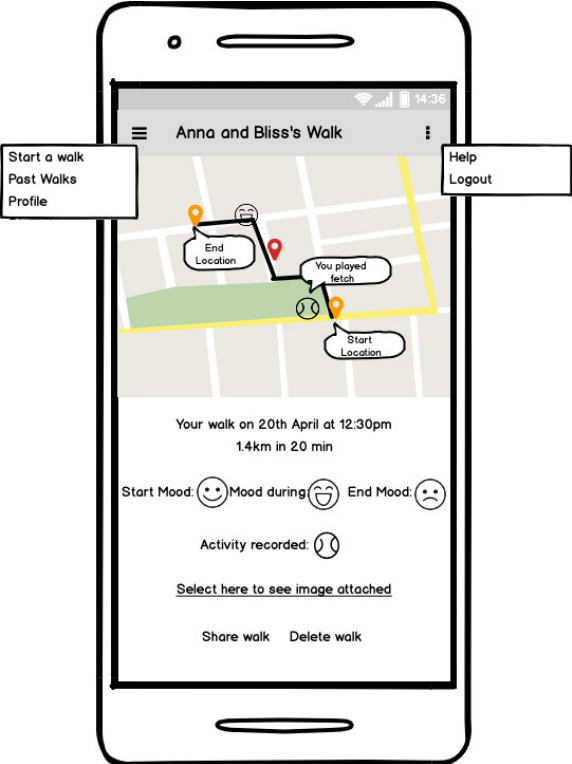
On this screen, the user can view all past saved walks. These walks will be in descending order (most recent first). There will be an option to change the order to least recent.

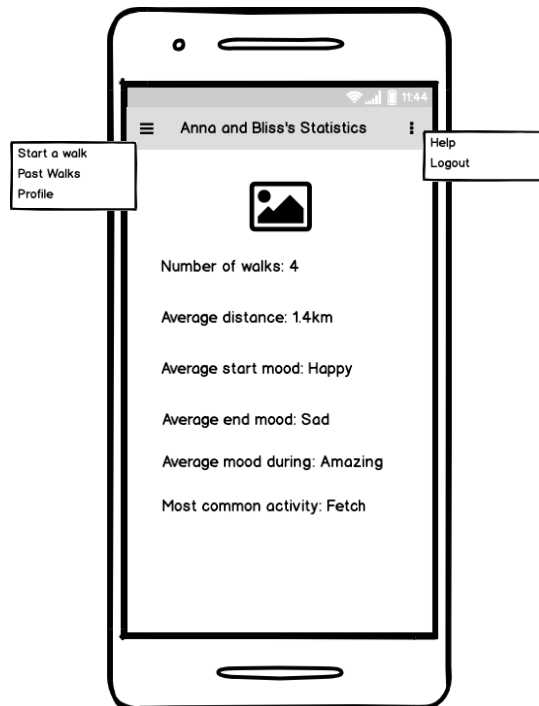
There is also an option to sort the walks by day, week or month. Once clicked this will dynamically change the order of the walks.

Each walk is displayed with its time and the distance to uniquely identify the walk.

The user can search through these walks by mood keywords, e.g. happy. This will filter the walks to only show the user walks which have 'happy' mood attached to them.

	<p>Once clicked, the user will be taken to the individual walk page for that walk with more details on the walk.</p> <p>Use Case(s): UC6, UC8, UC9</p>
--	---

<p>PTS No: 9</p> 	<p>View Individual Past Walk.</p> <p>On this screen the user can view a previous walk in detail. The user can be displayed this screen from two views. One from PTS8, by selecting it from the list of past walks. Or two, immediately after finishing a walk.</p> <p>The view is consistent with PTS3, the map will be the same as when the user finishes the walk. With all the recorded moods and activities presented to the user.</p> <p>The walk time and distance are displayed, alongside all moods. The user can select the link to view the image attached and a simple screen with the attached image will be displayed.</p> <p>The user can also select the share walk or delete walk text to perform that functionality.</p> <p>The toolbar is personalised with the user's details</p> <p>Use Case(s): UC3, UC4, UC10</p>
--	---

PTS No: 10**View statistics.**

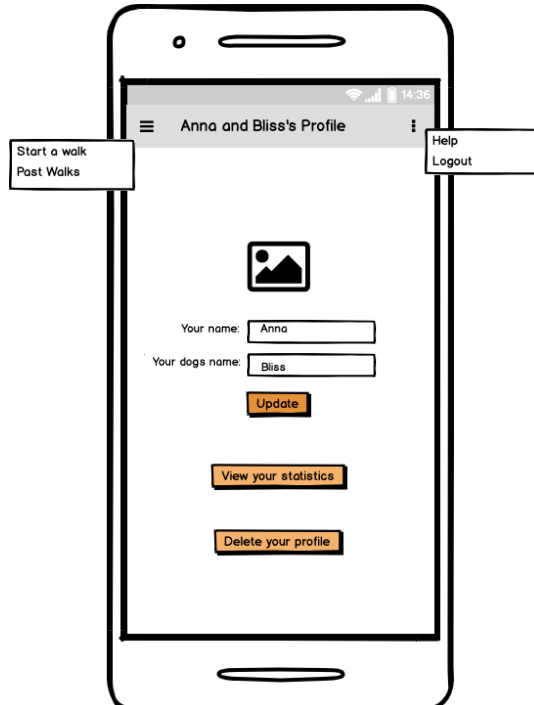
On this screen the user will be able to view their personalised statistics. These are taken from all walks recorded.

This includes number of walks, average distance and average start / end mood. It will give insights into the user's favourite activity too. The toolbar is personalised with the user's details.

This follows the alignment principle of carp and incorporates the application logo.

There is only one method of reaching this screen which is through PTS11.

Use Case(s): UC5

PTS No: 11**Edit profile details.**

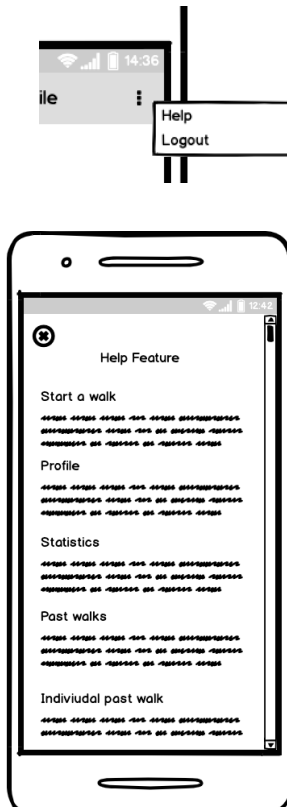
On this screen the user will be able to view their profile, make edits to their personal details, view their stats (PTS10) and delete their profile.

The toolbar is personalised with the user's details and once updated this will dynamically update.

This follows the colour scheme throughout.

Use Case(s): UC5, UC7, UC11

PTS No: 12



Help features.

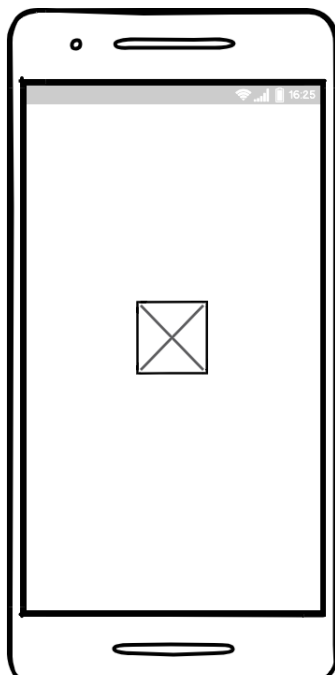
This help feature will be accessible from all main screens.

The help feature will provide an overview help guide for each screen in order to aid the user in case of any issues.

The user can exit the help screen using the 'x' in the top right corner.

Use Case(s): UC12

PTS No: 12



Splash Screen.

This screen will be displayed for several seconds before the starting of the application. This will provide a starting screen which will show while other functionality is loading in the background. It was found by Google, 53% of mobile users will stop using an app that is loading or more than 3 seconds [(Google Marketing Platform, 2020). Therefore, it is important to provide the user with a screen to view while this is loading.

This will be a clean screen with the application logo.

Use Case(s): NA

5.2.3. Heuristic Evaluation on Initial prototype

I was aware that this initial prototype fulfilled the basic requirements of the system and could be used as a basic MVP. However, I wanted to ensure that the application was easy and enjoyable to use. In order to test this, I carried out a heuristic evaluation on the initial prototype using Nielsen's usability principles (Nielsen and Molich, 1990).

In order to have a quantitative assessment of the impact of any usability problems, I used Nielsen's 0-4 rating scale:

0 = I don't agree that this is a usability problem at all

1 = Cosmetic problem only: need not be fixed unless extra time is available on project

2 = Minor usability problem: fixing this should be given low priority

3 = Major usability problem: important to fix, so should be given high priority

4 = Usability catastrophe: imperative to fix this before product can be released

Heuristic Evaluation No: HE1	
Problem	Too many buttons on the screen, drawing focus away from main activity. The four buttons in order for the user to perform functionality on the walk are unaesthetically pleasing to the eye. They take up half of the screen, drawing attention away from the main focus of the app: the walk itself.
Severity Rating	3
Relevant PTS Nos	PTS3
Violated Heuristics	Aesthetic and minimalist design.
Alterations	Remove all four buttons and replace these with a FAB button. A FAB (Floating Action Button) will hide all four options until clicked. This button can be placed over the map and the map can be made bigger to fill more of the screen.

Heuristic Evaluation No: HE2	
Problem	Emotion and activity tags not labelled. The current design does not label what each of the emotions or the activities are. This could cause confusion for the user as to what an icon may mean. When searching or reviewing past walks, users may use incorrect key words due to this misinterpretation.
Severity Rating	3
Relevant PTS Nos	PTS4, PTS5, PTS9
Violated Heuristics	Recognition rather than recall.
Alterations	Add labelling to the emotions and activities in all relevant screens.

Heuristic Evaluation No: HE3	
Problem	When adding a place, there is no confirmation of addition or way to backtrack if accidentally clicked.

	The user could accidentally add a place to the walk without any way of editing or deleting this place.
Severity Rating	4
Relevant PTS Nos	PTS9
Violated Heuristics	Error prevention. Help users recognise, diagnose and recover from errors.
Alterations	Add a dialog which pops up to ensure users are sure they wish to add the place to their walk.

Heuristic Evaluation No: HE4	
Problem	To view attached image of a walk, the user must select a link and navigate to another screen. The user has to perform one too many clicks in order to see the attached image on the walk. The screen with just the image attached on the walk is redundant and this could be incorporated into a better design.
Severity Rating	2
Relevant PTS Nos	PTS9
Violated Heuristics	User control and freedom. Match between system and the real world.
Alterations	Add the attached image on the same screen and make the view scrollable.

Heuristic Evaluation No: HE5	
Problem	To share and delete a walk, the text could be missed as it does not stand out. They are not commonly used features which need to be taking up space on the screen. These two features would not tend which the user will be using in top priority. They use up space on the screen and make it feel cluttered. The two text boxes seem out of place.
Severity Rating	3
Relevant PTS Nos	PTS9
Violated Heuristics	User control and freedom. Aesthetic and minimalist design.
Alterations	Move both options to the overflow menu in the toolbar. Add additional icons in order to become familiar to the user.

Heuristic Evaluation No: HE6	
Problem	When performing delete functionality there is no way of preventing accidental errors. The user could accidentally delete a walk or profile as there is no confirmation of deletion or a second chance for a user to backtrack on their error.
Severity Rating	4
Relevant PTS Nos	PTS9, PTS11

Violated Heuristics	Error prevention. Help users recognise, diagnose and recover from errors.
Alterations	Add a dialog which pops up to ensure users are sure they wish to perform the delete operation.

Heuristic Evaluation No: HE7	
Problem	There is no method of reaching the statistics page, except through the user profile page. The user can not easily reach the statistics page, except for clicking the button on the profile page.
Severity Rating	3
Relevant PTS Nos	PTS10, PTS11
Violated Heuristics	User control and freedom. Match between system and the real world.
Alterations	Either: -Add statistics into the toolbar overflow options and allow it to be accessible from everywhere -Add statistics into the profile page

Heuristic Evaluation No: HE8	
Problem	Both the profile page and statistics page seem 'empty'. Too much white space. The UI is not designed properly as the space seems empty. With the profile page only allowing the user to edit details, does this warrant enough for a whole screen.
Severity Rating	3
Relevant PTS Nos	PTS10, PTS11
Violated Heuristics	Aesthetic and minimalist design. Consistency and standards.
Alterations	Combine both of the profile and statistics pages together.

Heuristic Evaluation No: HE9	
Problem	Help documentation long to read through, provides overview of everything and not field level help, meaning user has to scroll to find relevant details. The help should be more specific to the page the user is on.
Severity Rating	3
Relevant PTS Nos	All
Violated Heuristics	Help and documentation.
Alterations	Make the help feature field level for each page. Make this a dialog to prevent the user having to navigate between more screens.

Heuristic Evaluation No: HE10	
Problem	User may be unclear of where they are in the system/navigation is not consistent with the toolbar.

	The user is not clear (apart from headings) as to where they are in the system. The navigation options with the toolbar seem unnatural and not the easiest to use.
Severity Rating	4
Relevant PTS Nos	All
Violated Heuristics	Visibility of system status. Match between system and the real world.
Alterations	Instead of the navigation through the menu on the toolbar, a bottom navigation menu will be displayed. This will allow the user to access the most important features of the application at all times.


Heuristic Evaluation No: HE11	
Problem	Users have to remember keywords to search/filter mood. The current design means that users have to remember words such as 'happy' or 'sad' to search walks which involve these.
Severity Rating	4
Relevant PTS Nos	PTS8
Violated Heuristics	Recognition rather than recall.
Alterations	Add the emojis to the filter to allow the user to select instead of remembering keywords.


5.2.4. Improved prototype





After reviewing the findings from the heuristic evaluation. Some screens and features had to be modified in order to ensure that the problems found have been resolved. The improved prototype follows the STN diagrams outlined in the use cases.

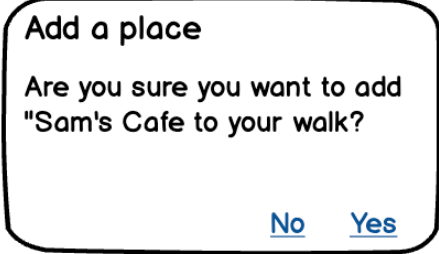
I decided on the name of the application to be "Unleashed" and designed a logo alongside icons which are used throughout the application. I created the logo and icons on Canva.

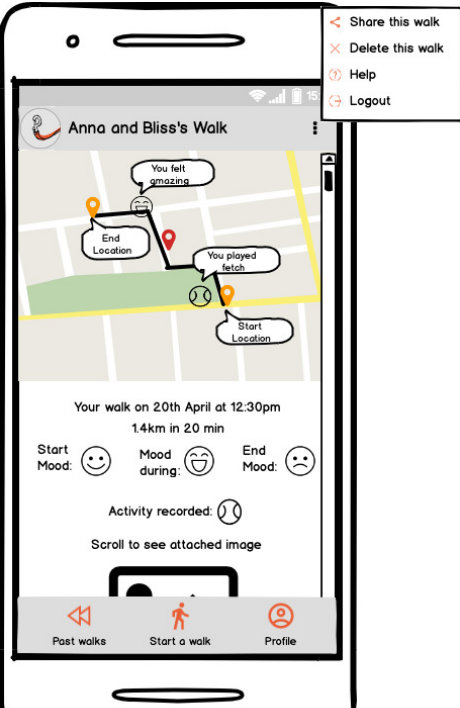


Related Heuristic No: HE10	Navigation Bar
	<p>The toolbar options of navigation have been replaced with a constant bottom navigation toolbar. This follows Androids navigation standards which removes the navigation menu from the toolbar all together and giving the use access to top level destinations that need to be accessible from anywhere in the app.</p> <p>As highlighted by Armstrong (2017), navigation drawers are less convenient and less accessible, unlike navigation bars.</p> <p>The navigation bar is in the 'Thumb zone' which is easy to reach with the thumb when the device is held in once hand (Ingram, 2016).</p>


Related Heuristic No: HE1	Start a walk screen
	<p>The map itself takes up most of the screen in this new design, focusing on the primary task.</p> <p>A FAB (floating action button) is used and positioned in the bottom right corner as suggested in the Android material guide. I also have used a '+' sign as suggested in the guide. A fab button is used to draw focus to the most important features on the screen. Once selected the four options of adding to your walk is shown. Icons have been used and text so that there is no confusion for the user.</p> <p>The toolbar is now populated with the applications logo which ties in with the colour scheme.</p> <p>The start mood has a label attached to reinforce what mood they inputted.</p>

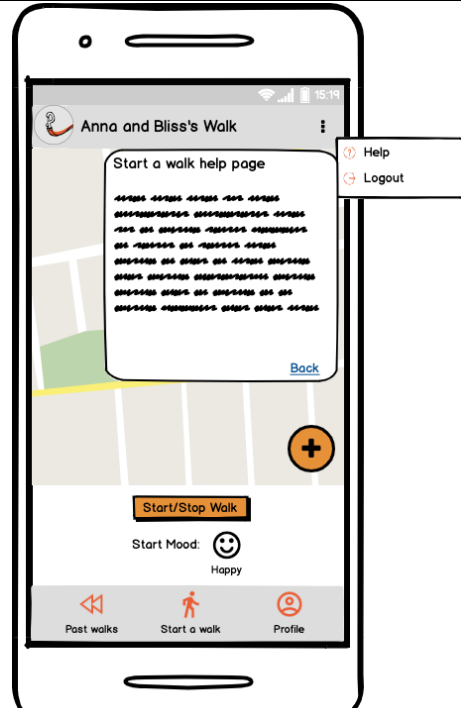
Related Heuristic No: HE2	Emotion and activity screens
 <p>Amazing Happy Neutral Sad Horrible</p> <p>Selected mood:  Happy</p>  <p>Cafe Met with friends Fetch Training Off lead walking</p> <p>Selected activity:  Fetch</p>	<p>The emotions have been labelled with appropriate labelling in order to remove ambiguity to the user.</p> <p>The same has been done with the activities.</p>

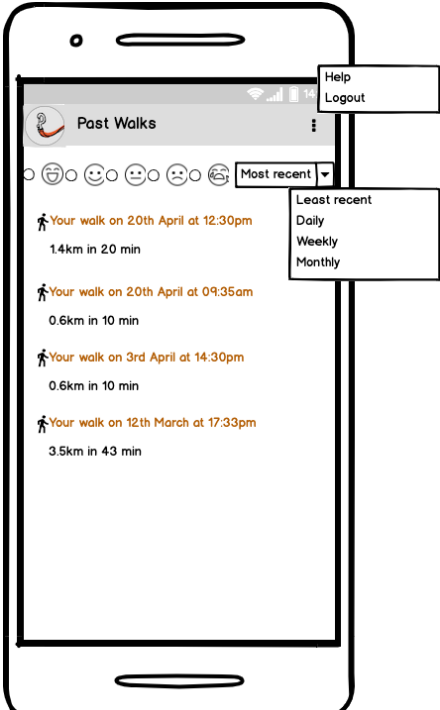
Related Heuristic No: HE3	Adding a place
	<p>Once the user selects a place on the map, a confirmation dialog appears which gives the user the option if they wish to add the place to their walk or not.</p>

<p>Related Heuristic No: HE4, HE5</p> 	<p>Individual walk screen</p> <p>Instead of the user having to click a link to view the attached image, the user can scroll on the individual walk screen in order to see their image.</p> <p>The image and delete options have been moved to the overflow menu in the toolbar. The toolbar now includes appropriate icons which follow the colour scheme and theme.</p>
---	---

<p>Related Heuristic No: HE6</p> <div data-bbox="272 1167 727 1429"> <p>Delete this walk</p> <p>Are you sure you want to delete this walk?</p> <p>No Yes</p> </div> <div data-bbox="256 1518 732 1794"> <p>Delete your profile</p> <p>Are you sure you want to delete your profile?</p> <p>No Yes</p> </div>	<p>Delete functionality</p> <p>When the user selects the delete a walk or delete profile option, dialogs are displayed to ensure the user wishes to perform the delete. This gives the user the opportunity to decide against.</p>
---	---

Related Heuristic No: HE7, HE8	Profile screen
	<p>The profile screen has been redesigned completely. A version of the application logo is displayed.</p> <p>The screen now has the user's statistics displayed on the profile screen. This eliminates the user having to navigate through screens to find their details. This also fills out the profile screen with appropriate details.</p>

Related Heuristic No: HE9	Help screen
	<p>The help feature has been redesigned to be field level help. Each page will have a different help page in the form of a dialog, therefore limiting the number of screens the user has to work through to complete tasks.</p>

Related Heuristic No: HE11	Past walks screen
	<p>The past walk screen has been redesigned in order to allow the user to select moods through radio button options.</p> <p>The other filters have been moved into the spinner object.</p> <p>The search feature has been removed as this is redundant now.</p> <p>The screen has also incorporated the UI colour scheme.</p>

6. Implementation

The following section will detail how the application has been developed. I will explain the main functionality of the application through discussing the database design, Activity diagrams, an overview of design and implementation and include code excerpts. I will detail how the main functionality of the system works within the code and how this has been achieved using the technologies that have been previously chosen.

6.1. Database design

Using Firebase's Firestore database, a NoSQL database allows for quick access to the user's entries and for flexibility in the fields included in the documents. This is particularly useful for this system as there are many optional elements which can be added to a walk such as moods and activities, which very likely could be null if the user decides not to store these; therefore, the schema of the database had to be flexible.

The schema consists of two primary collections, structured as detailed below:

1. UserProfile:

This collection holds all of the user's personal details. Each user has their own document which is uniquely identified by their userID. This userID is generated once a user signs up. This collection holds the user's name and dog's name which are to be used for personalised views throughout the application.

Collection: UserProfile

Document: random_userID

Fields:

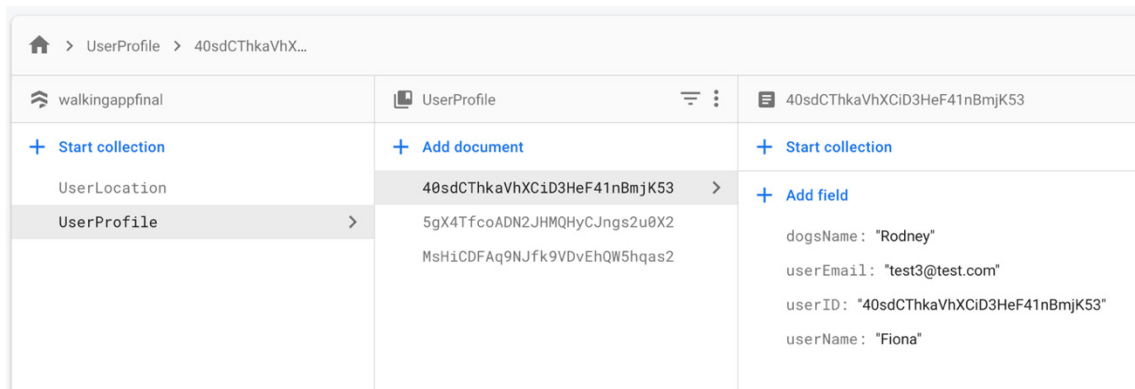
dogsName: string

userEmail: string

userID: string

userName: string

Example of this in Firebase:



2. UserLocation:

This collection holds all location data relevant to the user. Each user has their own document, identified by their unique userID. Within this document there are fields which collect the current walk location data. This will be the route and current point (geoPoint). These details are used for the live map during the walk, once the walk has stopped these two fields will be deleted.

Within the users document there is a nested collection: UserWalks. UserWalks is a collection which holds all of the user's previous walks and relevant details. Each walk is uniquely identified by a walkID. Each walk is saved into the database after the walk is stopped (i.e. the user presses 'stop walk'). The fields within the walk document are described below.

Collection: UserLocation

Document: random_userID

Fields:

geoPoint: geopoint

userRoute: array<geopoint>

Nested Collection: UserWalks

Document: random_walkID

Fields:

currentMood: string

currentMoodPlace: geopoint

distanceInKm: number

docID: string

endTime: timestamp

```
header: boolean
headerContent: string
imageUrl: string
moodEnd: string
moodStart: string
startTime: timestamp
userActivitiesPlace: hashmap<string, geopoint>
userPlacesLatLng: array<geopoint>
userRoute: array<geopoint>
```

The moods of the user are held in string fields, these are: currentMood, moodStart and moodEnd. The reasoning for keeping these in strings (hence one mood can be recorded) has been decided through user research, with users usually only wishing to record one mood during the walk.

currentMoodPlace is stored to reference where the current mood was set. Ideally, this would have been stored in a Hash Map with the currentMood string. This however is not feasible with how I planned to implement the searching and filtering of moods. When filtering moods I used get methods on each instance of a UserWalk, e.g. walk.getMoodStart() in order to call the collect() method on the start mood, end mood and current mood and transform these into a list of walks which the RecyclerView can display. I found it hard to obtain just the key of the Hash Map e.g. the mood string and therefore due to time constraints opted to keep each mood in a string.

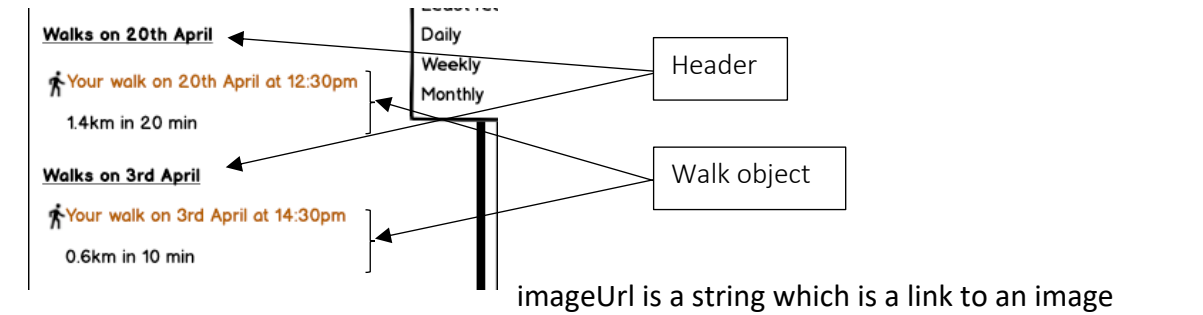
distanceInKm refers to the total distance of the walk. docID is the random_walkID referring to the walk. This is used as a reference as to where to store the current walk data in UserLocation once the walk has ended. endTime and startTime refer to the times related to the walk.

A design decision was made to make userActivitiesPlace a hashmap of string (activity) and geopoint (coordinates of activity), instead of splitting it into a string and geoPoint (like that of currentMood). This means user activities cannot be searched or filtered, however the user can add more than one activity to the walk. This made sense to me as a user is more likely to perform several activities on a walk than several moods.

userPlacesLatLng is an array of coordinates of any places which the user adds to the walk. These are added via selecting a place on the map and dropping a pin. userRoute is the entire route which was taken by the user, the coordinates are collected every 5 seconds until the user selects stop walk.

header and headerContent are used for easier filtering of walks. Within the application, it will use a RecyclerView to display walks on a Past Walks display screen. As part of the design, if the user selects for example, to view the walks by a 'Daily' view, this means that there will be two types of data passing into the RecyclerView to display, a walk and a header to show the day of filtering. A RecyclerView cannot determine which is a header or a walk without a Boolean check. A RecyclerView will just dump on the screen whatever is passed through it, therefore for formatting reasoning, there needs to be a clarification of what type

of object has been passed through. Below is an example design of what a header and walk objects will look like in the application.



imageUrl is a string which is a link to an image attached to the walk. The image is stored in the 'Storage' section of Firebase. This is Firebase's image storing method. The data is stored in a Google Cloud Storage bucket, an exabyte scale object storage solution with high availability and global redundancy (Firebase, 2020).

Example of this in Firebase Firestore Database:

UserLocation collection showing a live walk.

Figure 1 shows the UserLocation collection whilst a walk has been started. This walk is for the User 5gX4TfcoADN2JHMQHyCJngs2u0X2. This shows their current geoPoint (where they currently are) which will update every 5 seconds. This is used to display the users current location on the map. userRoute shows the route in which they are currently walking, which updates every 5 seconds. This is used to display the route they are walking on the map.

Figure 1:

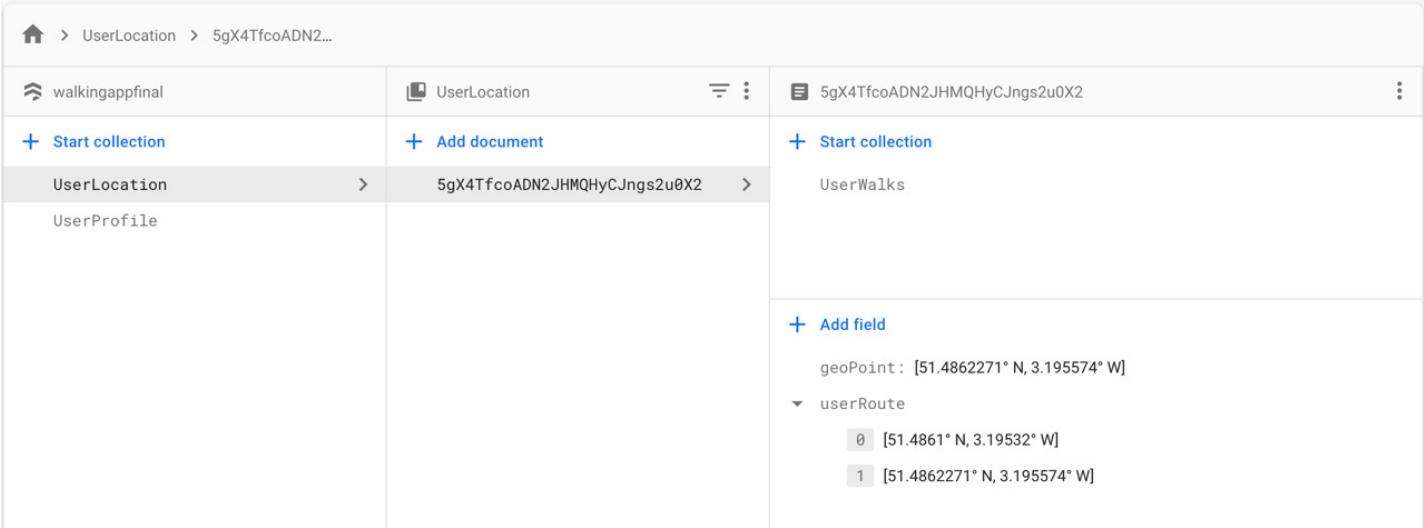


Figure 1: Live Walk

UserWalks showing a saved walk.

Figure 2 shows a saved walk in the database. This saved walk has an docID of GulbU7SGIrCR61pAJvOT and all relevant saved details can be seen.

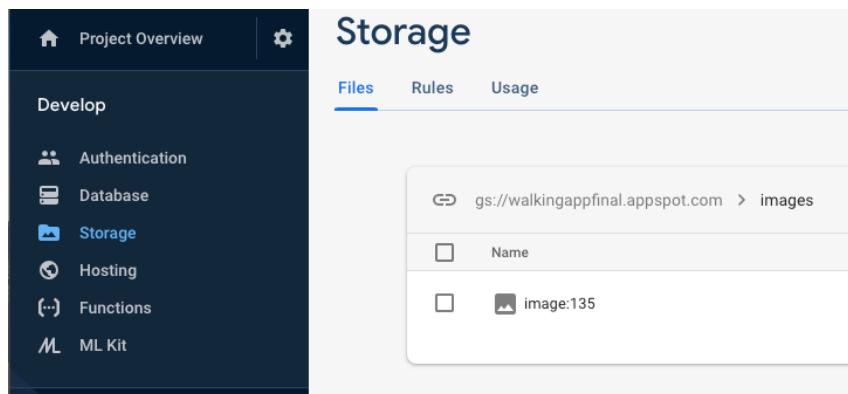
This includes for example the currentMood set was “Sad”, the walk started at 16:41 on the 28th April and ended at 16:42 on the 28th April. The imageUrl link has been set to where the image is stored in Firebase Storage. The userActivitiesPlace shows the user performed two activities on their walk, Café and Training and the respective places these were set. userPlacesLatLng shows the user has saved a place to the walk with its coordinates. userRoute shows an array of the entire walk route, which was updated every 5 seconds. header is also set to false as this is a walk object and will be treated as a walk object when passed through the RecyclerView.

Figure 2:

UserLocation > 5gX4TfcoADN2JHMQHyCJngs2u0X2 > UserWalks > GulbU7SGIrCR6...		
5gX4TfcoADN2JHMQHyCJngs2u0X2	UserWalks	GulbU7SGIrCR61pAJvOT
+ Start collection	+ Add document	+ Start collection
UserWalks >	0QTubyn1tJPYfFTKNUjx	+ Add field
	1F0xTpE9tzo6a97RWvLR	currentMood: "Sad"
	9z1cBmxUkUdOLOxGeR2D	currentMoodPlace: [51.4837379° N, 3.1912993° W]
	GulbU7SGIrCR61pAJvOT >	distanceInKm: 0.4084986529570341
+ Add field	LrunXc1MU0cw0D7IUcIb	docId: "GulbU7SGIrCR61pAJvOT"
This document has no data	hSX0FUgk84PC2VbEN8rD	endTime: 28 April 2020 at 16:42:00 UTC+1
		header: false
		headerContent: null
		imageUrl: "https://firebasestorage.googleapis.com/v0/b/walkingappfinal.appspot.com/o/media?alt=media&token=4ee03e95-6a85-4a94-b463-7578fed84bd3"
		moodEnd: "Amazing"
		moodStart: "Happy"
		startTime: 28 April 2020 at 16:41:00 UTC+1
		userActivitiesPlace
		Cafe: [51.4841507° N, 3.1919499° W]
		Training: [51.4862107° N, 3.1955414° W]
		userPlacesLatLng
		0 [51.4872589718578° N, 3.191453218460083° W]
		userRoute
		0 [51.4835733° N, 3.19157° W]
		1 [51.4835733° N, 3.19157° W]
		2 [51.4835733° N, 3.19157° W]
		3 [51.4837379° N, 3.1912993° W]
		4 [51.4841507° N, 3.1919499° W]

Figure 2: A saved walk

Storage (for images) Firebase:



Regarding security access to the database and storage, this can be set easily through 'Rules'. As this is a personal project I will be developing in test mode which allows read and write access without authentication. However, if this was a real-life project, these would be altered accordingly.

Rules would be altered so that each authenticated user can only access their data and their images held in Storage. By changing the read/write/modify rules to authenticated users and myself this would mean that only these users would be able to view and access their data.

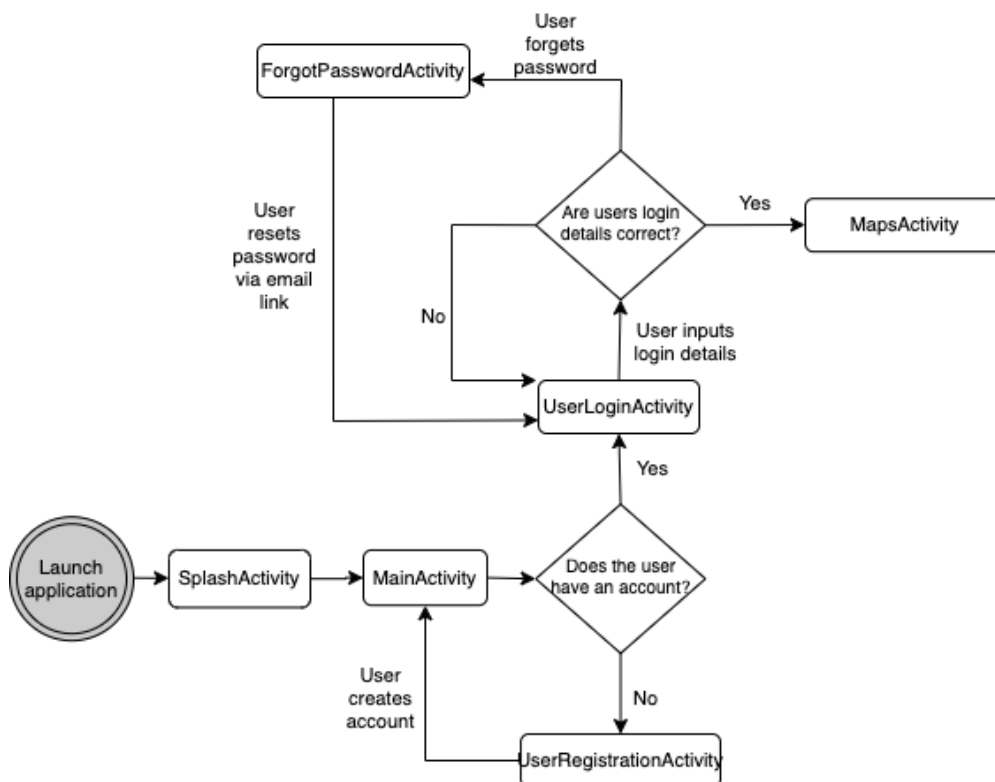
6.2. Activity Diagrams

In order to visualise the flow of users actions within the application, I have included Activity diagrams below which detail the many flows which involve several user actions.

Note: The navigation bar is available in all main activities (MapsActivity, PastWalksActivity, ProfileActivity and IndividualWalkActivity) and allow user access to MapsActivity, PastWalksActivity and ProfileActivity. This is not reflected in the diagram for simplicity. The user can also select the 'Back' button on their device at any time, again not portrayed on the system for simplicity.

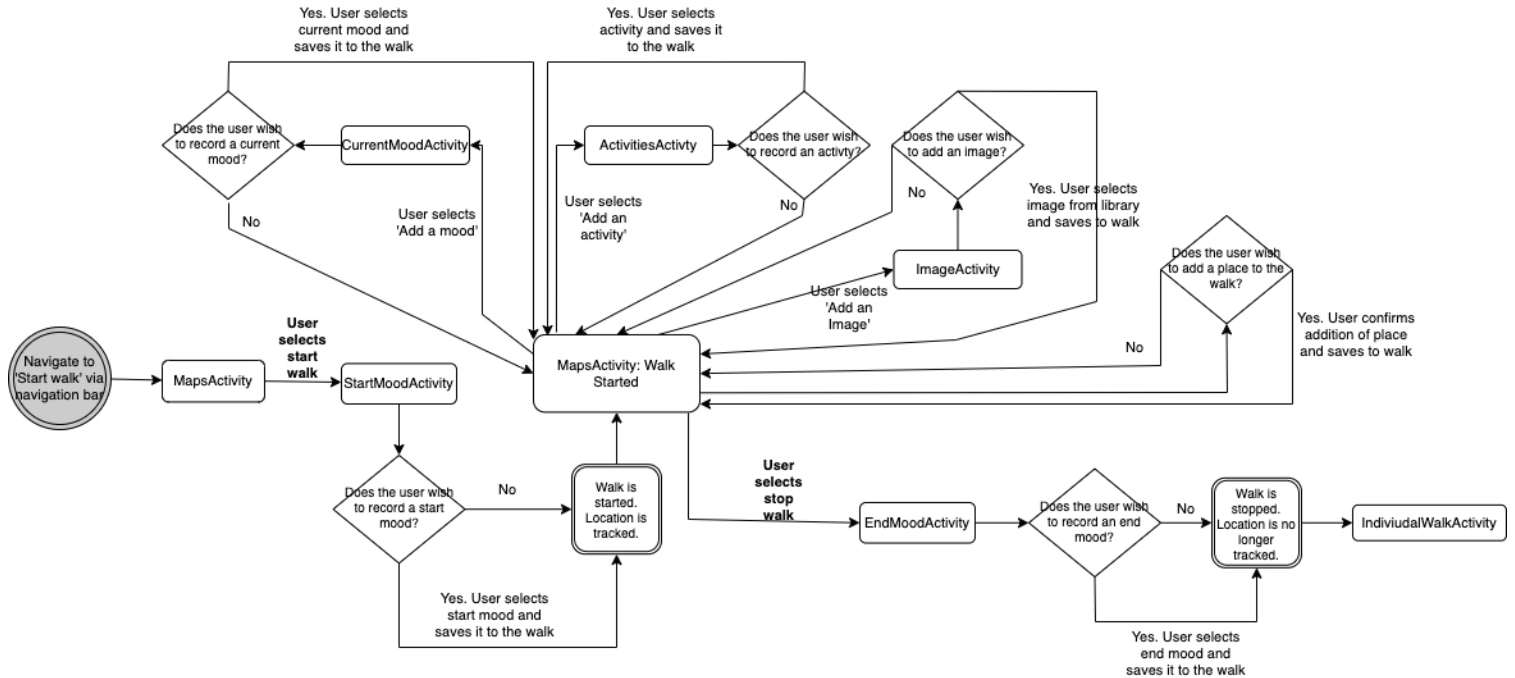
6.2.1. User Login actions

This takes the user from opening the application to MapsActivity (where they can start a walk).



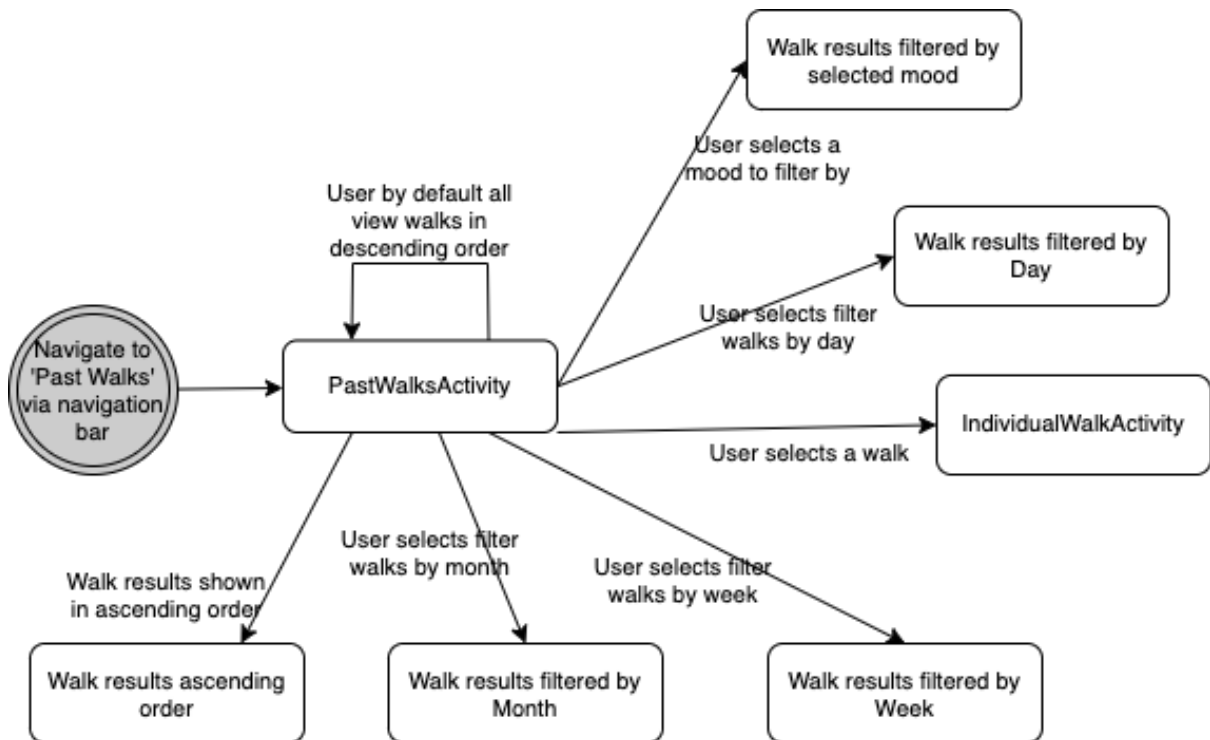
6.2.2. Track a walk user actions

User can decide against adding extra content to the walk even after selecting the adding content screen.



6.2.3. Search/filter past

walks



6.3. Overview of quality of Design and Implementation

Throughout the design and implementation of the project, it was important to ensure that code quality and extensibility was at the forefront of my mind. This was achieved through simple yet effective methods.

Instead of repeating code in several classes, I put commonly used methods into their own classes. I created five Utils classes; HelpUtils, LocationUtils, ActivityUtils, MoodUtils and DateUtils. This increased code readability and reuse. When working with the RecyclerView, I used an interface to handle the binding of walks to the view and to determine if the object passed through the RecyclerView to the display was a walk or a filtering header. The interface implements polymorphism as it is enforcing the code to handle different types of data e.g. walk or a header of a walk. As a reminder, a header being the type of filter applied if the user decides to filter walks, e.g. 'Walks in April' would be a header in the RecyclerView if the user selected the monthly filter.

I decided to use an Object Orientated approach throughout, using objects and creating instances of these objects. The objects used within the system are: UserLocation, UserWalk and UserProfile. I use the getters and setters within these classes throughout in order to manipulate data and set personal data.

I ensured there was a principle of high cohesion and low coupling throughout. High cohesion was implemented into the system by ensuring that each class is focused on what it is doing only and not broad and include methods which are not relevant. I ensured each class had a sole purpose, e.g. splitting start, current and end mood activities into three classes. This has made the code more reusable and maintainable. I enforced loose/low coupling through ensuring that classes are not overly dependent on each other. This would mean that a change in one class would rarely affect a change in another. This was implemented through the use of using mainly private variables and non-public methods throughout each class when possible.

6.3.1. Best practices for Android Development

Regarding best practices in Android Development, I followed several of the best practises set out by Futurice (2019). I used Gradle throughout and configured my build using the Android developers guide (Android Developers, 2020) and used a third party library to avoid 'reinventing the wheel' when handling Images in my application. I used Gradle in order to implement the core features within the system such as location services, map services and Firestore services. I used styles throughout to avoid duplicate attributes in layout XMLs, increasing readability. Another best practice is to use Android Studio as my main IDE, this is recommended as it is developed and updated by Google.

The final practice I followed was using the recommended Java package structure. This allows for simpler navigation as most related classes are in one package and promotes encapsulation. It also allows for my code to be structured in a more logical manner. Below the application structure, which is split into java files and resources files which hold the following:

Java

activities: holds all activity classes

adapters: holds all adapters (used for RecyclerView)

pojo: holds all object orientated java classes such as Object definition classes

utils: holds several classes with commonly used methods

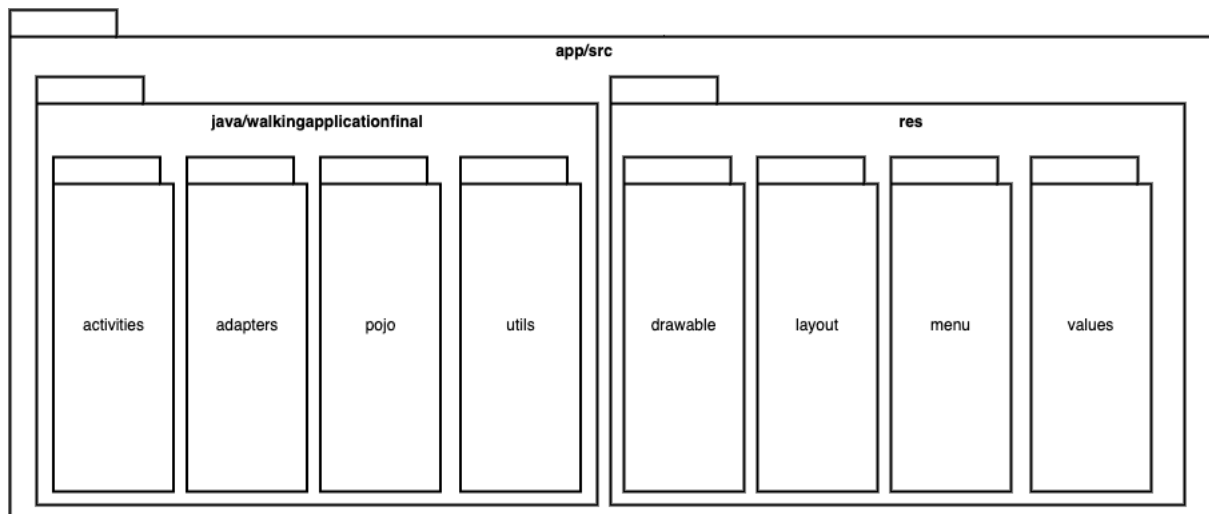
Res

drawable: holds all icons and images used in UI

layout: holds all layout xml files which are used to set out structure of activities

menu: holds all menu xml files used to define toolbar menu and bottom nav menu

values: holds colors, styles and dimension xml files



6.4. Detailed Implementation

In this section I will detail and demonstrate the actual development of the system. I will discuss core features and how they were implemented through the use of Java.

6.4.1. Overview of Implementation Structure

To the right is a screenshot of the classes within Android Studio.

All java files have been grouped together under the java folder and stored within packages.

The activities package stores all activities which are used throughout the system. These interact with each other as the user passes through different screens/activities.

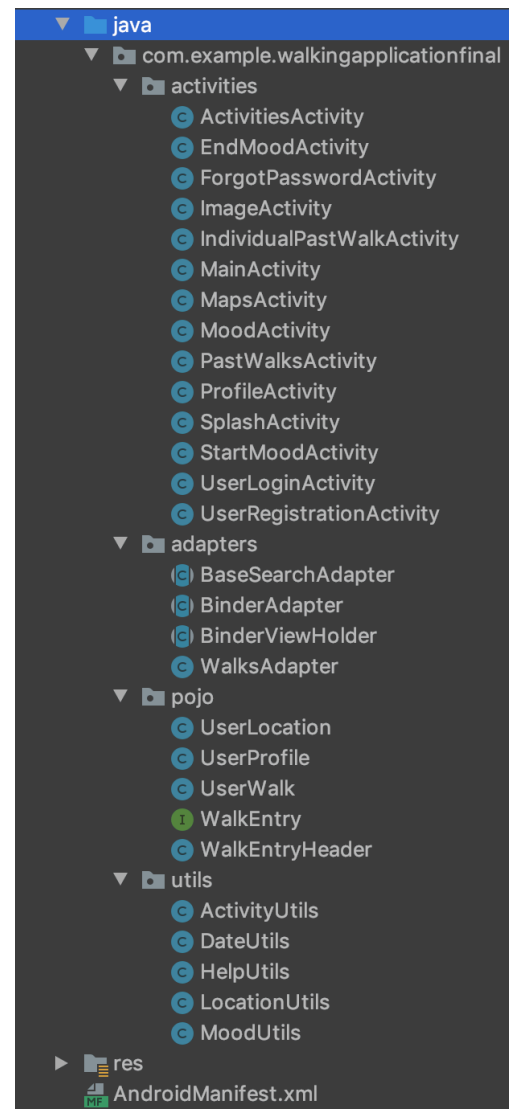
The adapters package stores the adapters used within the system. These are mostly used for the filtering and searching of past walks (PastWalksActivity) as it uses a RecyclerView to display data.

The pojo package stores all the object definition classes which are instantiated throughout the system. E.g. a new instance of UserProfile is created when a user registers to the application in UserRegistrationActivity, or a new instance of UserWalk is created when a user starts a new walk in MapsActivity.

The utils package holds five java classes which contain commonly used methods throughout several classes. This increases code reusability and accessibility. By using these classes it ensures a level of high cohesion due to each class/activity having a specific purpose and not containing a large number of methods which start to over complicate the class and its purpose. Instead of this, there are classes which all have specific purposes and these can interact.

Under the res folder, it contains all of the resources needed for the interface design, including drawable images, xml layout files, colours and styles.

The AndroidManifest.xml is used to provide essential information about the application to the Android operating system. This includes permissions needed for the application, in this case e.g. access to location when on the app. It also is used to store important information such as the API key to use the Google Maps API.



6.4.2. User Login, Register & Forgot Password

It was key that each user has their own secure space within the system to store their walks and personal details. This was achieved through the use of a login system. In order for the

```
firebaseAuth.signInWithEmailAndPassword(inputName, inputPassword).addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
    @Override  
    public void onComplete(@NonNull Task<AuthResult> task) {  
        if(task.isSuccessful()){  
            progressDialog.dismiss();  
            Toast.makeText( context: UserLoginActivity.this, text: "Login Success.", Toast.LENGTH_SHORT).show();  
            startActivity(new Intent( packageContext: UserLoginActivity.this, MapsActivity.class));  
        } else{  
            Toast.makeText( context: UserLoginActivity.this, text: "Login failed, Please enter the correct details.", Toast.LENGTH_SHORT).show();  
            progressDialog.dismiss();  
        }  
    }  
});
```

user to Log in to the system, there is a communication made with the database through FirebaseAuth. This is to ensure authentication of users on the system. For this system, I opted for email and password based authentication as this would be most accessible to the end users, instead of possibly a Facebook or Google authentication method. Below shows how the FirebaseAuth instance allows for sign into the system through the use of the signInWithEmailAndPassword method.

If the user does not already have an account in the system, they can sign up. This functionality was achieved again through FirebaseAuth. This was achieved through the createUserWithEmailAndPassword method.

```
firebaseAuth.createUserWithEmailAndPassword(user_email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
    @Override  
    public void onComplete(@NonNull Task<AuthResult> task) {  
        if (task.isSuccessful()) {  
            Toast.makeText( context: UserRegistrationActivity.this, text: "Registration Successful, get ready to walk " + dogsname + " ! ", Toast.LENGTH_LONG).show();  
            sendUserData();  
            //Send user back to login:  
            startActivity(new Intent( packageContext: UserRegistrationActivity.this, MainActivity.class));  
        } else {  
            Log.w( tag: "createUser:", msg: "createUser failure", task.getException());  
            Toast.makeText( context: UserRegistrationActivity.this, text: "Registration Not Successful", Toast.LENGTH_SHORT).show();  
        }  
    }  
});
```

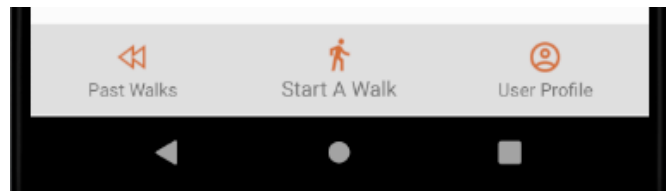
If the user has forgotten their password, through the use of FirebaseAuth, a password reset email is sent to the user. This is achieved through the sendPasswordResetEmail method.

```
firebaseAuth.sendPasswordResetEmail(email).addOnCompleteListener(new OnCompleteListener<Void>() {  
    @Override  
    public void onComplete(@NonNull Task<Void> task) {  
        Toast.makeText( context: ForgotPasswordActivity.this, text: "An email has been sent. Please check your emails and reset your password.", Toast.LENGTH_SHORT).show();  
        startActivity(new Intent( packageContext: ForgotPasswordActivity.this, UserLoginActivity.class));  
    }  
});
```

6.4.3. User Navigation

As decided upon in the User Interface Design, a bottom navigation bar has been implemented into the system. This allows the user to navigate between the three main activities in the system: MapsActivity (Start a walk), ProfileActivity (User Profile) and PastWalksActivity (Past Walks).

When the user has selected and is on the activity, the icon and text grow slightly bigger to indicate to the user they are that screen.



The navigation bar was achieved through the use of the BottomNavigationView. Using the setNavigationItemSelectedListener, it is easy to control the implications of users clicks. The example below is the bottomNavigationView on the MapsActivity, this has a Boolean check on each listener to ensure a walk is not in progress.

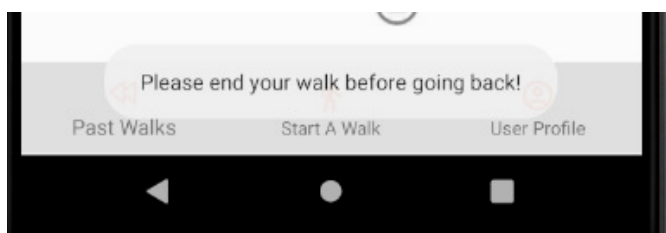
Due to the nature of how a walk is stored, the user is on a walk they can only leave the MapsActivity screen to add content to their walk (e.g. add a mood/activity). This is so that the system is up to date, as PastWalks and ProfileActivity rely on walk data and each walk must be completed before viewing.

```
public void setUpNavBar(){
    bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
            switch (menuItem.getItemId()){
                case R.id.action_pastwalks:
                    if(walkStarted == true){
                        Toast.makeText( context: MapsActivity.this, text: "Please end your walk before viewing past walks!", Toast.LENGTH_SHORT).show();
                    } else {startActivity(new Intent( packageContext: MapsActivity.this, PastWalksActivity.class)); }
                    break;

                case R.id.action_profile:
                    if(walkStarted == true){
                        Toast.makeText( context: MapsActivity.this, text: "Please end your walk before viewing your profile!", Toast.LENGTH_SHORT).show();
                    } else {startActivity(new Intent( packageContext: MapsActivity.this, ProfileActivity.class));}
                    break;
            }
            return true;
        }
    });
}
```

In order to minimise errors and crashes within the application with the user pressing the Android 'Back' button on their device, I added this Boolean check to ensure the walk is stopped before exiting this screen.

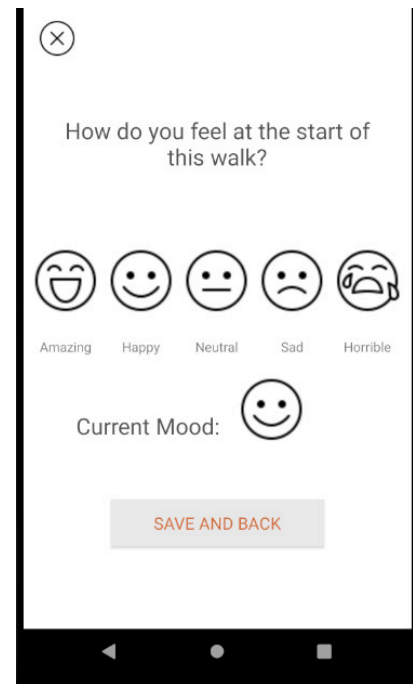
Shown is the example Toast which is displayed if the user tries to go 'Back' using the Android Back button, when already started a walk.



6.4.4. Adding a mood to a walk

There are three activities used to record the users emotion/mood. All of which operate in a similar fashion but are called at separate times. Each class has an observeMood() function. This function is used to set the mood variable to the mood chosen by the user and to set the 'currentMood' image to that the user has selected, for user confirmation of selection.

```
//Listening to mood changes
private void observeMood() {
    View.OnClickListener onClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            tvCurrentMood.setVisibility(View.VISIBLE);
            ivCurrentMood.setVisibility(View.VISIBLE);
            switch (v.getId()) {
                case R.id.extrahappyicon:
                    ivCurrentMood.setImageResource(R.drawable.extrahappy);
                    startCurrentMood = "Amazing";
                    break;
                case R.id.happyicon:
                    ivCurrentMood.setImageResource(R.drawable.happy);
                    startCurrentMood = "Happy";
                    break;
                case R.id.neutralicon:
                    ivCurrentMood.setImageResource(R.drawable.neutral);
                    startCurrentMood = "Neutral";
                    break;
                case R.id.sadicon:
                    ivCurrentMood.setImageResource(R.drawable.sad);
                    startCurrentMood = "Sad";
                    break;
                case R.id.supersad:
                    ivCurrentMood.setImageResource(R.drawable.supersad);
                    startCurrentMood = "Horrible";
                    break;
            }
        }
    };
}
```



The user selects an image and this image is translated into a String value which is passed through back to the MapsActivity using the putExtra method with Intents. An instance of the Intent class is created, the extra string is added to it and the result is set. If the user selects the 'Exit button or the 'Back' button on the Android itself, the starting/current/ending mood is set to null.

```
btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.putExtra( name: "currentStartMood", startCurrentMood);
        setResult(RESULT_OK, intent);
        finish();
    }
});
```

The 'result' is then read in the MapsActivity in the onActivityResult() method. In this method, a switch statement determines which screen the user has come from. The getStringExtra method is called on the incoming Intent and this receives the extra string, i.e. The currentStartMood and then sets the mood using the setMoodStart method on the

UserWalk instance already created if the resultCode (ensuring no transmission errors) is set to ok.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    switch (requestCode) {
        case REQUEST_STARTING_MOOD:
            if (resultCode == RESULT_OK) {
                startingMood = data.getStringExtra( name: "currentStartMood");

                if (startingMood != null){
                    userWalk.setMoodStart(startingMood);
                }
                setUpStartScreen();
            }
            break;
    }
}
```

If the user decides to record a mood during the walk, the chosen emoji will be shown on the interactive walk map where they recorded this mood, to remind them that this is where they felt that certain mood.

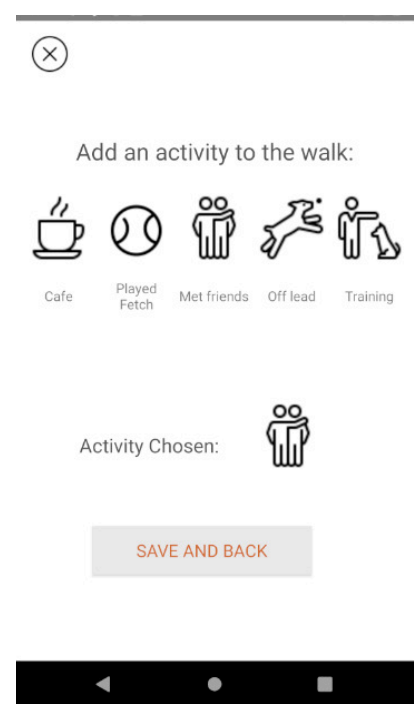
One limitation of the implementation of this, is that the user is only able to select one mood at the start, during and after a walk. I faced issues in implementing a method to allow the user to store several moods during the walk due to these having to be stored in a HashMap and struggling to make this work when filtering and searching walks. With more time and future enhancements it would be useful to allow the user to select several moods to more accurately depict how they felt during that walk.

6.4.5. Adding an activity to a walk

The ActivitiesActivity screen operates in a similar fashion to the three mood screens. It also uses an observing method, observeActivities() which uses a switch statement. It also passes through an intent back to the MapsActivity which is discussed above.

The user can add as many activities to the walk as they wish, and the relevant icon which is chosen is displayed on the map where they performed the activity.

On reflection, this was not implemented in the most elegant form. Due to the similarities in these four classes, they could have been refactored into one dynamic activity which changes based on what the user wishes to add to their walk. With more time, this could have been achieved through the use of passing Strings between the intent to determine the type of element the user wished to add to the walk.

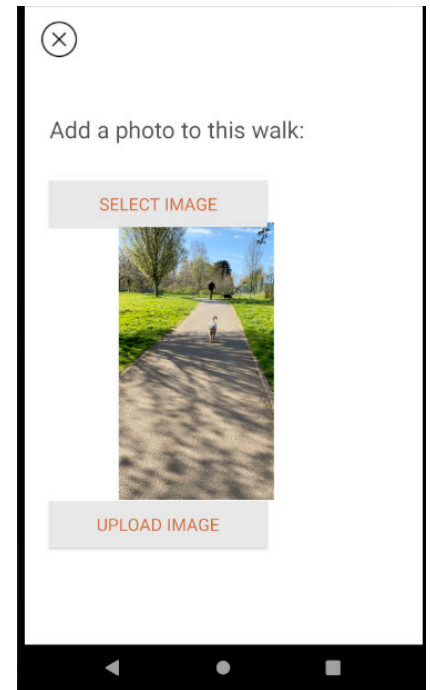


6.4.6. Adding an image to a walk

In order to allow the user to select and upload an image from their library to the walk, it's important to first request permissions to access the user's gallery.

In order to display the selected image to the user I used a third party library called Glide. Glide is 'a fast and efficient image loading library for Android focused on smooth scrolling' (Bumpteck, 2017). Glide deals with the compression and decompression of the images. This is triggered via switch statement checking the request code for REQUEST_OPEN_GALLERY, which checks permissions. This code shows how glide loads an image (mSelectedImage) into an ImageView (ivUpload).

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    switch (requestCode) {
        case REQUEST_OPEN_GALLERY:
            if (resultCode == RESULT_OK) {
                mSelectedImage = data.getData();
                Glide.with( activity: this)
                    .load(mSelectedImage)
                    .into(ivUpload);
            }
            break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```



When uploading the image to Firestore Storage, I created an instance of the StorageReference class (storageReference) and called the putFile method to put the image into the Storage section of Firebase. I then take the URI (url of where the image is stored) and pass this back to MapsActivity so that it can be set to the UserWalk instance and then updated in Firebase once the walk has finished.

```
//Uploads to firebase storage
private void uploadPicture() {
    if (mSelectedImage == null) {
        Toast.makeText( context: this, text: "Please select an image first.", Toast.LENGTH_LONG).show();
        return;
    }
    //Reference of where image to be stored:
    StorageReference ref = storageReference.child("images/" + mSelectedImage.getLastPathSegment());
    ref.putFile(mSelectedImage).continueWithTask(task -> {
        if (!task.isSuccessful())
            throw task.getException();
        // Continue with the task to get the download URI
        return ref.getDownloadUrl();
    }).addOnCompleteListener(task -> {

        if (task.isSuccessful()) {
            Uri downloadUri = task.getResult();
            //String to be put into walk:
            downloadedImageStr = downloadUri.toString();

            //Send ImageURI back to Maps:
            Intent intent = new Intent();
            intent.putExtra( names: "imageURL", downloadedImageStr);
            setResult(RESULT_OK, intent);
            finish();
        } else {
            Toast.makeText( context: this, text: "Image could not be saved.", Toast.LENGTH_LONG).show();
        }
    });
}
```

6.4.7. User Location

The main functionality of the application revolves around users location. In order to obtain the users location, the application must first request permission to the users *fine* location. Fine location is a more precise location tracking of the user, which is needed for the accuracy of this application.

```
//Request Permissions:
//If permission not granted, request permissions:
if (ContextCompat.checkSelfPermission( context: MainActivity.this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED){
    ActivityCompat.requestPermissions( activity: MainActivity.this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, requestCode: 1);
}
//If location permission already granted start walk:
else {
```

In order to accurately and correctly set up my location tracking within the application, I followed the guidelines set out by Android Developers regarding Location (Android Developers, 2020). I used the Google Services Location API – FusedLocationProvider in order to retrieve the devices last known location (which is equivalent to the current location). I call the `getLastLocation` on the created instance of `FusedLocationProvider`.

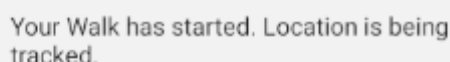
To store parameters for requests to the `FusedLocationProvider`, I created a `LocationRequest`. This is where I can decide how often my application receives location updates.

```
protected void createLocationRequest(){
    locationRequest = LocationRequest.create();
    locationRequest.setInterval(10000);
    locationRequest.setFastestInterval(5000);
    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}
```

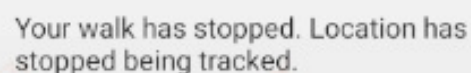
I have set up my location to receive location updates at the fastest every 5 seconds. I also have set the priority to `PRIORITY_HIGH_ACCURACY`, this requests the most precise location possible. The location services are more likely to use GPS to determine it's location. The combination of this accuracy and the fast update interval of 5 seconds provides updates that are accurate to within a few feet which is crucial for this application showing the users location in real time.

As the application receives updates of the current location, it writes this to Firestore to the `UserLocation` collection, where it is used to create a live tracking interactive map.

Due to the sensitive nature of tracking location, it is incredibly important to be as transparent to the user as possible with when the application is tracking their location. As found in a study by Clark, E (2019), 31% of people don't feel comfortable with location tracking as they value privacy. In order to allow the user to know when the location is being tracked, a toast is shown to the user that the walk has started and location is being tracked. A toast is also shown when location has stopped being tracked and the walk has stopped.



Your Walk has started. Location is being tracked.



Your walk has stopped. Location has stopped being tracked.

Past Walks

Start A Walk

User Profile

6.4.8. Showing User's Location on the map

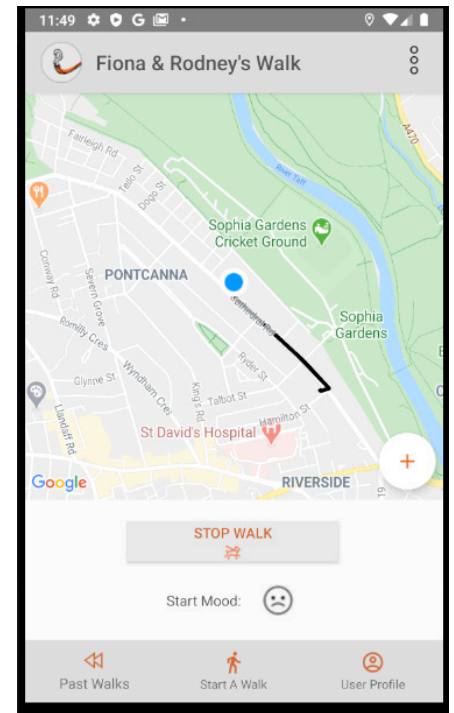
In order to retrieve the users current location and route taken (to show the path taken), I add a listener to a reference to the UserLocation collection which is being updated every 5 seconds with the users location. I update the current location icon with the users location as they move and add this geopoint to a temporary array which is used to draw a Polyline of the users route as they move.

```
private void showLocation(){
    //Get current location data:
    DocumentReference docRef = firebaseFirestore.collection( collectionPath: "UserLocation").document(firebaseAuth.getUid());

    docRef.addSnapshotListener(new EventListener<DocumentSnapshot>() {
        @Override
        public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable FirebaseFirestoreException e) {
            if (e != null) {
                Log.w( tag: "retrieveLocation()", msg: "Listen failed.", e);
                return;
            }
            //If walk has started and has written to DB:
            if(walkStarted==true){
                if (snapshot != null && snapshot.exists()) {
                    //current location:
                    GeoPoint stringgeo = (GeoPoint) snapshot.get("geoPoint");
                    //GoogleMaps API uses LatLngs:
                    userloc = LocationUtils.convertToLatLng(stringgeo);
                    //Temp array for polyline:
                    userTempRoute.add(userloc);

                    //draws current route on maps:
                    Polyline polyline1 = mMap.addPolyline(new PolylineOptions().clickable(true).add(userloc));
                    polyline1.setPoints(userTempRoute);

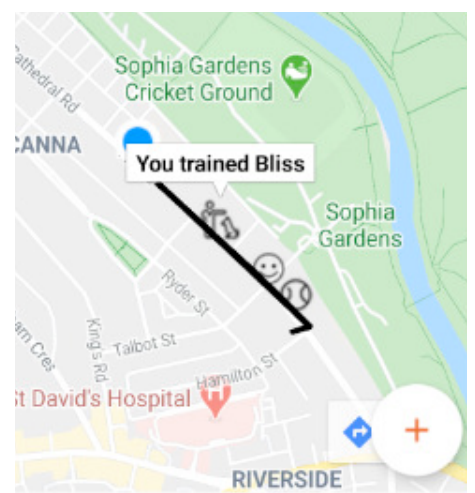
                    //Google Map Updating Current Location :
                    currentlocmarker.remove();
                    BitmapDescriptor icon = BitmapDescriptorFactory.fromResource(R.drawable.currentloc);
                    currentlocmarker = mMap.addMarker(new MarkerOptions().position(userloc).title("Your Location").icon(icon));
                    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userloc, 15));
                }
            }
            else {
                Log.d( tag: "retrieveLocation()", msg: "Current data: null");
            }
        }
    });
}
```



6.4.9. Showing User's added walk elements on the map

To make the walk an interactive experience, when the user adds a current mood or an activity a marker which denoted the mood/activity chosen is displayed on the map where they set it. This is achieved through the use of a switch statement which sets appropriate icon on the map at the users current location. The icon titles are personalised for the activity and the dog name.

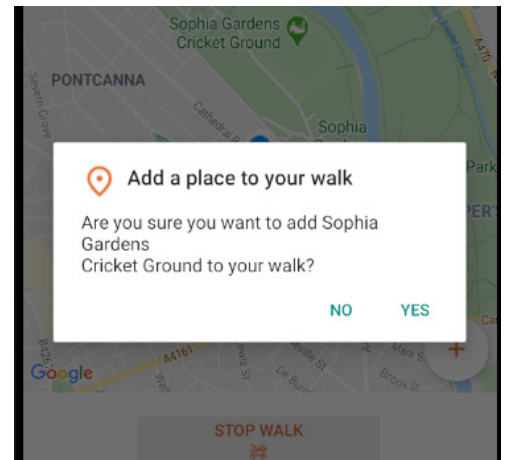
Due to the use of GoogleMaps API, once an element is clicked on the map, a directions icon is shown which provides the user with the option to get directions to that location. In this example, where the user trained their dog.



6.4.10. Adding a place to a walk

In order to add a place to a walk, I make use of Google Maps API. Through using this API, it comes with many useful prebuilt methods and features. As the map is interactive and clickable, in order for the user to add a place to their walk they can click on a place. I use the `onPoiClick()` method in order to handle when a user does so on the map. In order to ensure they wish to add the place to the walk, I have included a dialog which gives the user to opportunity to change their mind.

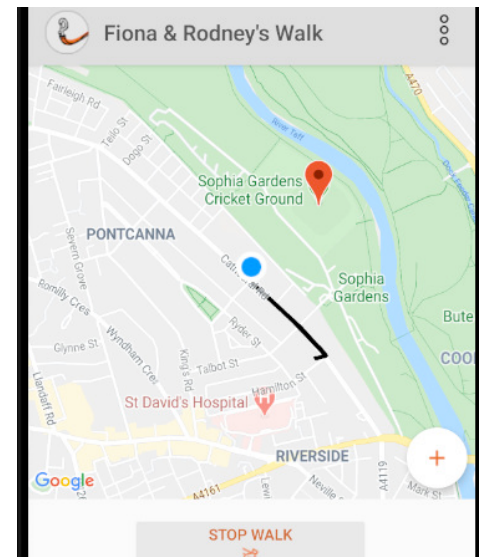
After selecting the place, the place is marked with a 'marker' to indicate to the user that they visited this place. I add the co-ordinates of the place to an array which will be used when the user is viewing this walk at a later time. The use of the array is due to the fact the user can add as many places as they wish to each walk.



```
@Override
public void onPoiClick(PointOfInterest poi) {
    //need to have a check if they want to add it to the place:
    new AlertDialog.Builder(context: this).setTitle("Add a place to your walk").setMessage("Are you sure you want to add " + poi.name + " to your walk?")
    //Yes:
    .setPositiveButton(text: "Yes", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            userPlaces.add(poi.name);

            //adding to array to show places clicked when on Individual Walk:
            GeoPoint placeLatLng = LocationUtils.convertToGeoPoint(poi.latLng);
            userPlaceLatLng.add(placeLatLng);

            String placeName = poi.name;
            mMap.addMarker(new MarkerOptions().position(new LatLng(poi.latLng.latitude, poi.latLng.longitude)).title("Place visited: " + placeName));
            //No:
        }).setNegativeButton(text: "No", listener: null)
        .setIcon(R.drawable.place).show();
    }
}
```



One limitation of the implementation of this is that it allows the user to add a place anywhere visible on the map. With more time I would have implemented a 'boundary' feature as such to ensure that the place they are adding is within a predefined radius to ensure that the place was 'on' their walk.

If more time allowed, this could be more integrated with the Google Places API, in order to view reviews, leave reviews and see opening times for places.

6.4.11. Saving a User's walk

As the walk comes to a finish, all elements are set to the instance created of the UserWalk object. This object is then saved into Firebase. This is achieved via a set() method on a reference to the Database. After the walk is saved, the current location data of the user is deleted in order to comply with GDPR laws and that data not being needed anymore.

```
//Write entire walk to DB (UserWalks collection):
private void writeCompletedUserWalk(){
    if(!walkactivitiesandplace.isEmpty()){
        userWalk.setUserActivitiesPlace(walkactivitiesandplace);
    }
    if(userRoute!=null){
        //UserWalk entire route:
        userWalk.setUserRoute(userRoute);

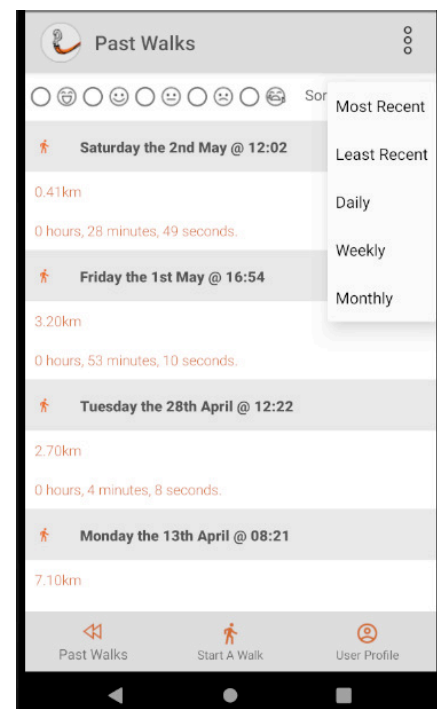
        //Create reference to walk:
        DocumentReference walkref = firebaseFirestore.collection( collectionPath: "UserLocation").document(firebaseAuth.getUid())
            .collection( collectionPath: "UserWalks").document(userWalk.getDocId());

        //Setting into Firebase:
        walkref.set(userWalk).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if(task.isSuccessful()){ Log.d( tag: "writeCompletedUserWalk:", msg: "writeRouteLocations: \ninserted user walk into database "); }
            }
        });
    } else { Log.d( tag: "writeRouteLocations:", msg: "route null"); }
    //Delete current location data as no longer needed:
    deleteCurrentLocationData();
}
```

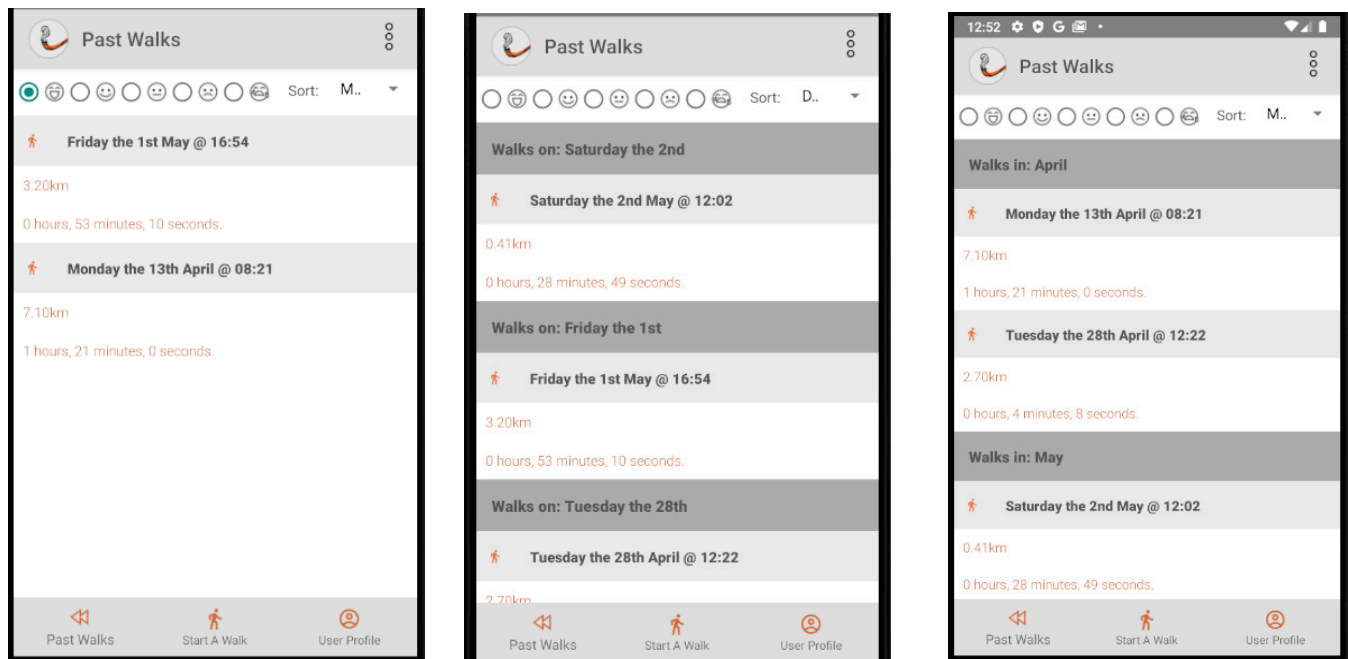
6.4.12. Viewing past walks

To view all walks that a user has previously made I used a RecyclerView. This is used to display a scrolling list of dynamic elements. The walks are automatically displayed in ascending order (most recent walk first). Andress (2019) explained that there are 5 different ways to view data (Location, Alphabetical, Time, Category and Hierarchy), I tried to incorporate this in my views and ensured they were organised by time and when a time filter is applied, a header explaining the filter is shown.

This is a scrollable view. This screen is dynamic and has several filters in which the user can use to narrow down their walks. These include filtering by mood, in which users can use the radio buttons at the top of the screen to select, or they can use the drop down menu to select a time based filter.



Below I have included screenshots of three example views, from left to right: Walks filtered with 'Amazing' mood, walks filtered by day, walks filtered by month.



Due to the dynamic nature of the RecyclerView, filtering the walks which also displayed a header e.g. 'Walks in: April', proved to be one of the more complicated features to implement. I created a WalkEntry class, which is passed through the RecyclerView. This was used as two types of object are being passed through the RecyclerView, both a header and a walk (UserWalk). The application had to be able to determine what items were walks and headers for formatting reasons. Below, I have shown my method of filtering by Day. I create a map which consists of a pair of integers (day of walk which is represented as a number and the month) and a list of UserWalks. I stream this map and filter if the walkEntry is not header and group by the day. This is then passed through to the RecyclerView. This method is used for both Monthly and Weekly views.

```
private void showWalksByFilter(String filter) {
    //List of Walk Entries:
    List<WalkEntry> walkEntries = new ArrayList<>();

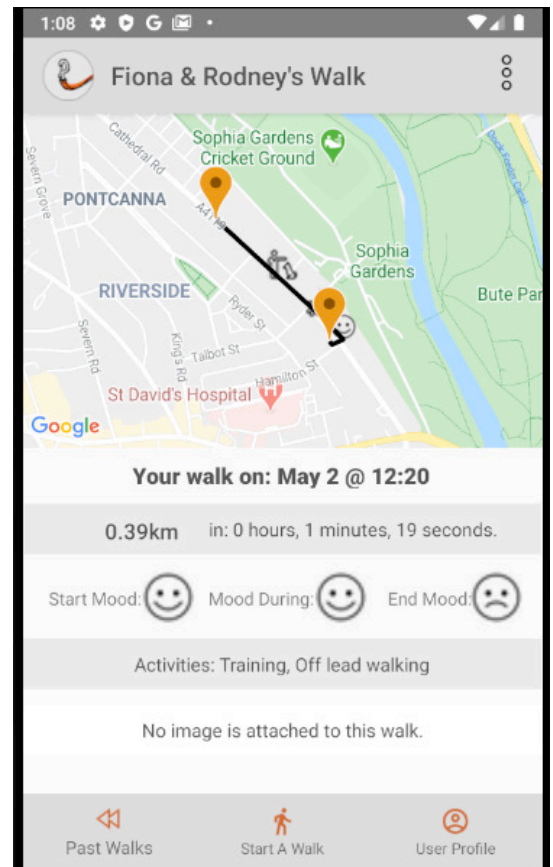
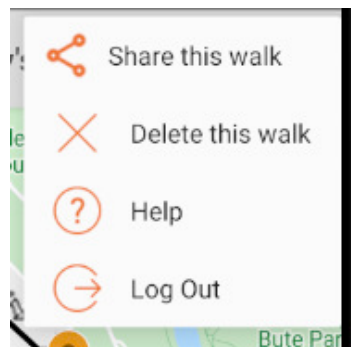
    switch (filter) {
        case ("Daily"):
            Map<Pair<Integer, Integer>, List<UserWalk>> data = mWalkEntries.stream()
                //filter walk entries that are not headers:
                .filter(walkEntry -> !walkEntry.isHeader())
                //make them into User Walk objects:
                .map(walkEntry -> (UserWalk) walkEntry)
                //group by day of walk:
                .collect(Collectors.groupingBy(DateUtils::getDayOfWeek));

            data.entrySet()
                .forEach(entry -> {
                    Pair<Integer, Integer> dayPair = entry.getKey();
                    List<UserWalk> userWalks = entry.getValue();
                    WalkEntryHeader header = new WalkEntryHeader("Walks on: " + DateUtils.getTextDay(dayPair.first)
                        + " the " + dayPair.second + DateUtils.getDaySuffix(dayPair.second));
                    walkEntries.add(header);
                    walkEntries.addAll(userWalks);
                });
            break;
    }
}
```

6.4.13. Viewing an Individual Walk

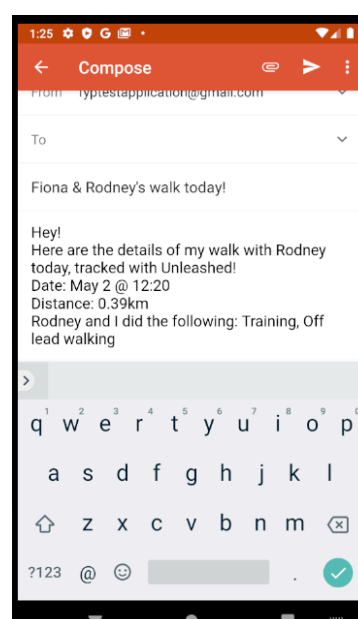
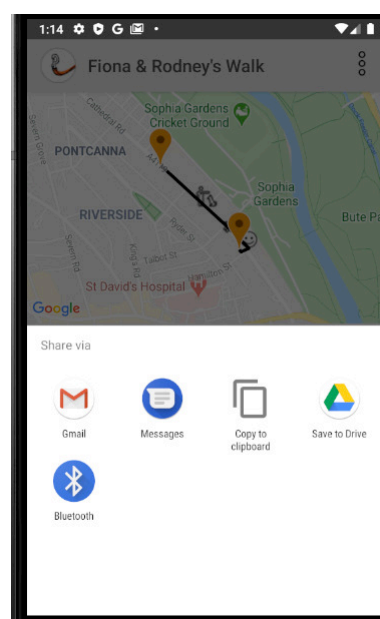
Users can either select a walk from the RecyclerView or after completing a walk is shown a summary of their walk. This is to show the user the walk details and all additional information attached to the walk. This information is simply pulled from Firebase using a 'get' command. The elements are then placed on the screen into the appropriate text, image and map views. The view is scrollable if the user has attached an image.

The user can delete this walk, which will delete all information from Firestore. They can also share the walk with friends. These options are accessible from the overflow menu in the toolbar in the top right.



6.4.14. Sharing a walk

When the user selects the option to share their walk, they are offered several methods of doing so. In the example below, I have selected the 'Gmail' option. As it can be seen the user's Gmail application opens and a pre-set email subject and body of text is created and displays details of the users walk. This is dynamic and changes if the user has performed activities or not.



This is implemented by creating an instance of the Intent class of type 'ACTION_SEND' meaning this intent is to send data. The intent's type is set to plain text as it's primary purpose is to send information of type plain text. The text body is built and added this to the sharing intent to pass onto the method of sharing. The method createChooser then determines the method of sharing e.g. email application or message.

```
//Shares the walk with friends via email:
private void shareWalk(){

    Date startDate = DateUtils.timestampToDate(startTimeDate);
    LocalDate localDate = startDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

    //create intent which main purpose is to send information of type text
    Intent sharingIntent = new Intent(android.content.Intent.ACTION_SEND);
    sharingIntent.setType("text/plain");

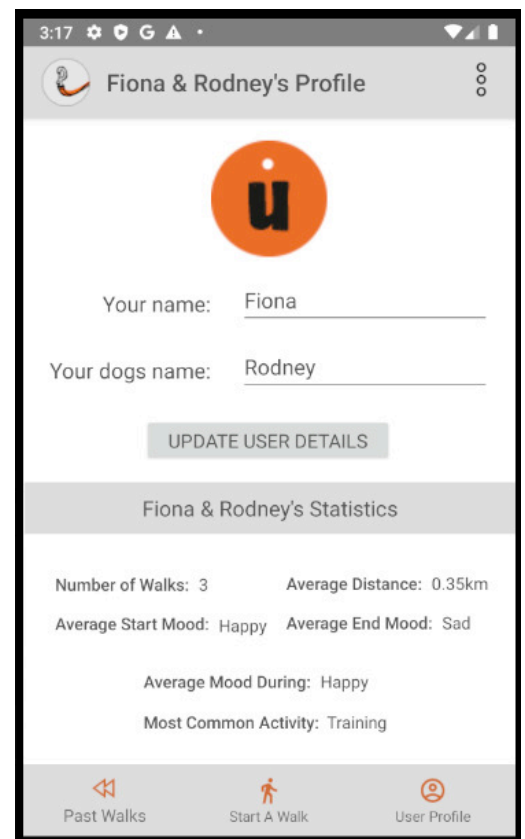
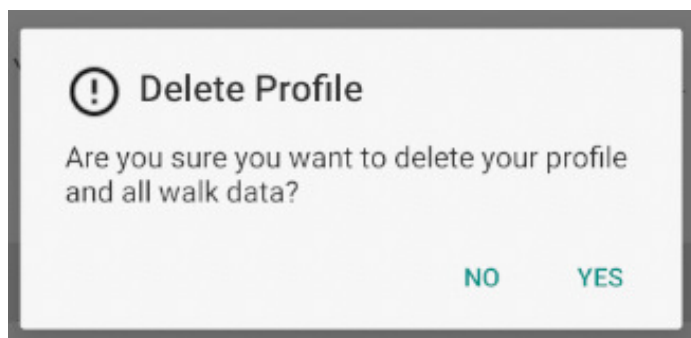
    //Detail body:
    String shareBody = "Hey! \n Here are the details of my walk with " + dogsname +
        " today, tracked with Unleashed! \n Date: " + DateUtils.getMonth(localDate) + " " + DateUtils.getDayOfMonth(localDate) + DateUtils.getTimeOfWalk(startDate) +
        "\n Distance: " + LocationUtils.convertDistance(distance);
    sharingIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, username + " & " + dogsname + "'s walk today!");

    if(activityoutput!=null){
        outputactivities = "\n " + dogsname + " and I did the following: " + activityoutput;
        sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT, value: shareBody + outputactivities);
    } else {
        sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
    }
    startActivity(Intent.createChooser(sharingIntent, title: "Share via"));
}
```

6.4.15. User's Profile and Statistics

On the profile page, the user can edit their personal details and can view their own personalised statistics. The users details are updated through the update() method made on a reference to the UserProfile collection. Once updated all personalised views on the application will be updated too.

The user can delete their profile from this page by selecting the overflow menu toolbar. They are asked for confirmation of deletion via a Dialog.



Once confirmed, the profile is deleted and the user is sent back to the MainActivity and a Toast is displayed to confirm to the user the profile has been deleted. The deletion is executed via the delete() method on both the UserProfile collection and the UserLocation collection in Firebase.

Regarding the statistics, the total number of walks they have walked which is calculated from counting the number of walks stored on Firebase for that particular user. The average distance is calculated by cumulating the distances on all the walks and finding the average based on how many walks they have saved. Finding the average start, current, end and activity is all calculated using the same methodology.

All walks are looped through and an array is filled with all recorded start moods, current moods, end moods and activities. Then on each array, a stream operation is performed and a map is created for each string (mood or activity) and the number of occurrences. The key which maps to the largest occurrence is denoted as the most common. The string (mood or activity) is then displayed in the appropriate TextView on the screen.

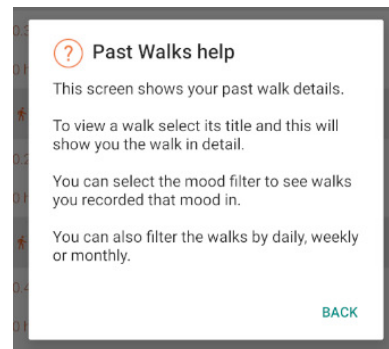
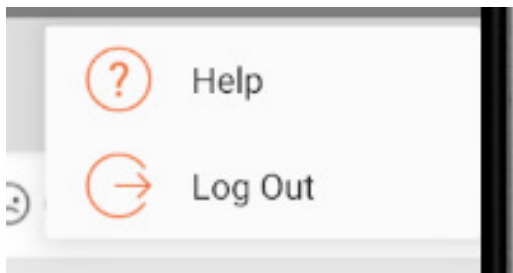
Currently, if there are two most commonly occurring moods or activities it is sorted by default by alphabetical order. This could be further enhanced to display both all of the most common elements but for simplicity of the screen and minimalist design this was decided against. Below shows the method in order to find the most common start mood. This methodology is applied to current, end and activities.

```
//finds most common mood/activity:
private void findMostCommon(){
    //Find most common Start Mood
    if(!userStartMoods.isEmpty()){
        String mostRepeatedStart = userStartMoods.stream() Stream<String>
            .collect(Collectors.groupingBy(w -> w, Collectors.counting())) Map<String, Long>
            .entrySet() Set<Map<K, V>.Entry<String, Long>>
            .stream() Stream<Map<K, V>.Entry<String, Long>>
            .max(Comparator.comparing(Map.Entry::getValue)) Optional<Map<K, V>.Entry<String, Long>>
            .get() Map<K, V>.Entry<String, Long>
            .getKey();

        tvPulledAverageStartMood.setText(mostRepeatedStart);
    } else {
        tvPulledAverageStartMood.setText("NA");
    }
}
```

6.4.16. User Help

It was important for help to be accessible in all main screens in the application. Help is found in a consistent place in all in the overflow menu of the toolbar. Help is given in the form of a dialog and is specific to each screen. This is handled via a switch case method in which two parameters are passed through to find out which screen the user is on (String) and therefore show the correct help on that screen (Context). An instance of the AlertDialog class is created and a title and message is set. An icon for 'help' is attached to each AlertDialog in order to indicate to the user the manner of the Dialog.



```
public static void showHelp(Context context, String screen) {
    switch (screen) {
        case "Profile":
            new AlertDialog.Builder(context).setTitle("Profile help")
                .setMessage("This screen shows your personal details. " +
                    "\n\nTo update your details, change as you wish and select update." +
                    "\n\nYour statistics are shown on this screen. " +
                    "\nThese automatically update as you walk and enter more details. \n" +
                    "\nIf a statistic is shown as 'NA' this means we need you to walk more and record these details!")
                .setNegativeButton( text: "Back", listener: null).setIcon(R.drawable.help).show();
            break;
        case "Past Walks":
            new AlertDialog.Builder(context).setTitle("Past Walks help")
                .setMessage("This screen shows your past walk details. " +
                    "\n\nTo view a walk select its title and this will show you the walk in detail." +
                    "\n\nYou can select the mood filter to see walks you recorded that mood in." +
                    "\n\nYou can also filter the walks by daily, weekly or monthly. \n")
                .setNegativeButton( text: "Back", listener: null).setIcon(R.drawable.help).show();
            break;
    }
}
```

7. Results

The following section will detail a thorough testing of the system through test cases to ensure functionality and real user testing for usability.

7.1. Test Cases

It is necessary to verify whether the system functions are in compliance with the requirements. In order to test and review this and the completeness of the application a set of test cases were created. These tests were carried out as accurate to the test case template as possible to ensure no bias in testing the system. In order to test the capabilities of the system, three types of data were used in the tests:

1. **Valid test data:** To check if the system accepts and handles data that it should do
2. **Invalid test data:** To check if the system correctly handles invalid values, showing relevant messages to the user
3. **Absent data:** To check if the system handles blank field values, showing relevant messages to the user

For full test cases, see Appendix 3 Test Cases.

Out of 19 tests, 17 passed. This is a functionality test pass rate of a high 89%. This depicts that the majority of the requirements initially outlined for the system were met. It also depicts that the requirements were implemented to a high standard, handling valid, invalid and absent data. Out of the original 14 functional requirements, 17 completely passed, 1 had a partial failure and 1 failed.

7.1.1. Acceptance Criteria Results

7.1.1.1. Functional Requirements

- **FR1:** The system will provide the user its own secure space which only they can access: **PASS**
- **FR2:** The system will allow the user to track and store a walk in real time: **PASS**
- **FR3:** During this walk, the user can optionally include an emotion tag: **PASS**
- **FR4:** During this walk, the user can optionally include activity tags: **PASS**
- **FR5:** During this walk, the user can optionally include place tags: **PASS**
- **FR6:** During this walk, the user can optionally include a photograph: **PASS**
- **FR7:** The user can view past walk entries: **PARTIAL FAIL**

- **FR8:** The user can view an individual past walk: **PASS**
- **FR9:** The user can delete a walk: **PASS**
- **FR10:** The user can view statistics regarding their walks: **PASS**
- **FR11:** The user can perform search analysis of the walk entries, displaying emotions that match the provided emojis: **PASS**
- **FR12:** The user can delete their profile, including all location data: **PASS**
- **FR13:** The user can share details of a walk with their friends via email/message: **PASS**
- **FR14:** The user can edit their personal details: **PASS**
- **FR15:** The user will receive notifications to let them know once they have hit their pre inputted target distance for the day: **FAIL**

Regarding the failures, FR7 had a partial failure due to filtering by time views. The actual filtering by time is implemented and the user can filter by Newest walk first, oldest walk first, Day, Week or Month. However, when filtering by Week or Month, the walks are not automatically sorted into descending order (Most recent walk first). This is successfully implemented for Daily walks. Due to time constraints this feature could not be implemented. It was not as straight forward as previously thought it would be and this is due to the RecyclerView and how it handles week dates and month dates as Calendar objects.

FR15 was not implemented in time. However, this was a 'Could have' requirement which through previous communication had been decided this would only be implemented if enough time was left in the project. With more time, this feature could be implemented relatively easily with a daily target in KM being set as the user registers to the system and then once a walk is completed each day, a check could be made to see if the walks(s) total is more than the target.

I do not believe that either of these two tests which failed effect the main functionality of the system and therefore it can be concluded that the vast majority of the functional requirements of the system have been met.

7.1.1.2. Non-Functional Requirements

The non-functional requirements of the system have been tested non-directly through the means of the Functional test cases.

- **NFR1:** The system will immediately save walk entries into the database: **PASS:** Users can view a walk immediately after saving it. Example in TC4.
- **NFR2:** The user will always have accessibility to their walk entries: **PASS:** Users can view and delete walks. Example in TC7, TC8, TC11.

- **NFR3:** The system will be intuitive and easy to use: **PASS:** After using Android Developer Material guides throughout design. This is confirmed in User usability testing below.
- **NFR4:** The system will provide a help feature: **PASS:** Users can view help screens throughout. Example in TC19.
- **NFR5:** The system will be reliable: **PASS:** Application has been developed to support 100% of Android Versions. Recycler view in TC7 did not take longer than 5 seconds to generate.
- **NFR6:** The application will allow for future developments **PASS:** Code has been commented throughout and designed in a repeatable manner. Camel case variables and methods have been used throughout to make this consistent.

7.2. Usability Testing

It was originally hoped and planned to test the application on a set of real potential users whom accurately depict the user personas. This would have been achieved by testing the application on the dog walking group whom participated in the original User Requirements Questionnaire. COVID-19 had a large effect on the reality of this. Due to the social distancing measures put in place and the government law of only allowing one hour of exercise a day, this meant that usability testing had to be performed on a much smaller scale confined to users in my home, this consisted of 3 users. This still aligns with some aspects of the user personas, as each user is a regular dog walker and therefore still a valuable exercise to perform.

There of course are limitations regarding this style of testing, with much fewer test subjects this will impact the test results quality. It is also likely, as the test subjects are all from the same home and background with the same dog there may be some bias in how they walk their dog. However, this was discussed and asked of the users to answer the questions as honestly as possible.

A test was created which consisted of 9 tasks and after each task the user was encouraged to think out loud regarding the usability and how easy they found the task. The tasks were based off the Use Cases and User Personas for the application.

The user was given the opportunity after each task to add any additional comments as to how they found navigation of the application, the user interface and functionality as a whole. As each user was performing the task, I documented their thoughts/any difficulties they had or general comments. Up until this point, the application had only been tested on one device (Google Pixel 2) and an emulator during development and Functional testing.

Tests were carried out on the following three devices and operating systems:

1. Google Pixel 4: Android 10
2. Samsung Galaxy S20: Android 10

3. OnePlus 6: Android 9

7.2.1. User Tasks

- **UST1: Create a new user login**
Explanation: Create a new user login with provided details. This is key to ensure users can create a login for the application.
Time allocated: 1 minute
- **UST2: Log in to the system**
Explanation: Create a new user login with provided details. This is key to ensure users can create a login for the application a starting point.
Time allocated: 30 seconds
- **UST3: Go for a walk and record added extras**
Explanation: This is the main purpose of the application. The walk itself with the dog and being able to add extras to the walk. It is essential to see how the user finds adding extras and interacting with the map. I provided the user with test data to make this process easier and ensure all features are tested.
Time allocated: Roughly 10 minutes (as long as the walk takes)
- **UST4: Share recent walk just completed via email**
Explanation: This is essential for the user to interact with the walk summary page. This test will check to see if the user can share the walk with friends.
Time allocated: 1 minute 30 seconds
- **UST5: Review past walk entries for a given date**
Explanation: This is essential for the user to interact with the past walk page. This will provide the user with all previous walks. This test is key to see how the user interacts with the filters and how easily they can navigate the sorting of the walks. For efficiency I provided the user with 3 walks to find.
Time allocated: 1 minute 30 seconds
- **UST6: Review past walk entries for a given mood**
Explanation: This is essential for the user to interact with the past walk page regarding moods. This test is key to see how the user interacts with the mood filters and how easily they can navigate the sorting of the walks. For efficiency I provided the user with 3 moods to search with.
Time allocated: 1 minute
- **UST7: Delete a previous walk**
Explanation: This is essential for the user to interact with the walks themselves. For efficiency (and the fact there are several users) I provided a walk to the user at the time of the test to delete. This test checks the user's familiarity with walks and how to interact with an individual walk.
Time allocated: 45 seconds

- **UST8: Edit personal details**

Explanation: This test will ask the user to change the username and dogs name on the account. This will check the user can interact with the profile page and make the appropriate changes.

Time allocated: 1 minute

- **UST9: Review statistics of previous walks**

Explanation: This test will ask the user to find and review 3 provided statistics. This will ensure the user understands the statistic format.

Time allocated: 30 seconds

For the test and full results, please see Appendix 4 User Usability Testing.

7.2.2. Usability Results Summary

The 3 users who performed the tests had varying degrees of technical ability using mobile phone applications. 2 users (User 1 and User 2) are very confident in using applications and are in the age range of 16-24. User 2 would track their dog walks already using a walk tracking application and therefore a great tester. User 3 is not as competent with their technical ability, does not currently track walks and is in the age range of 50-55.

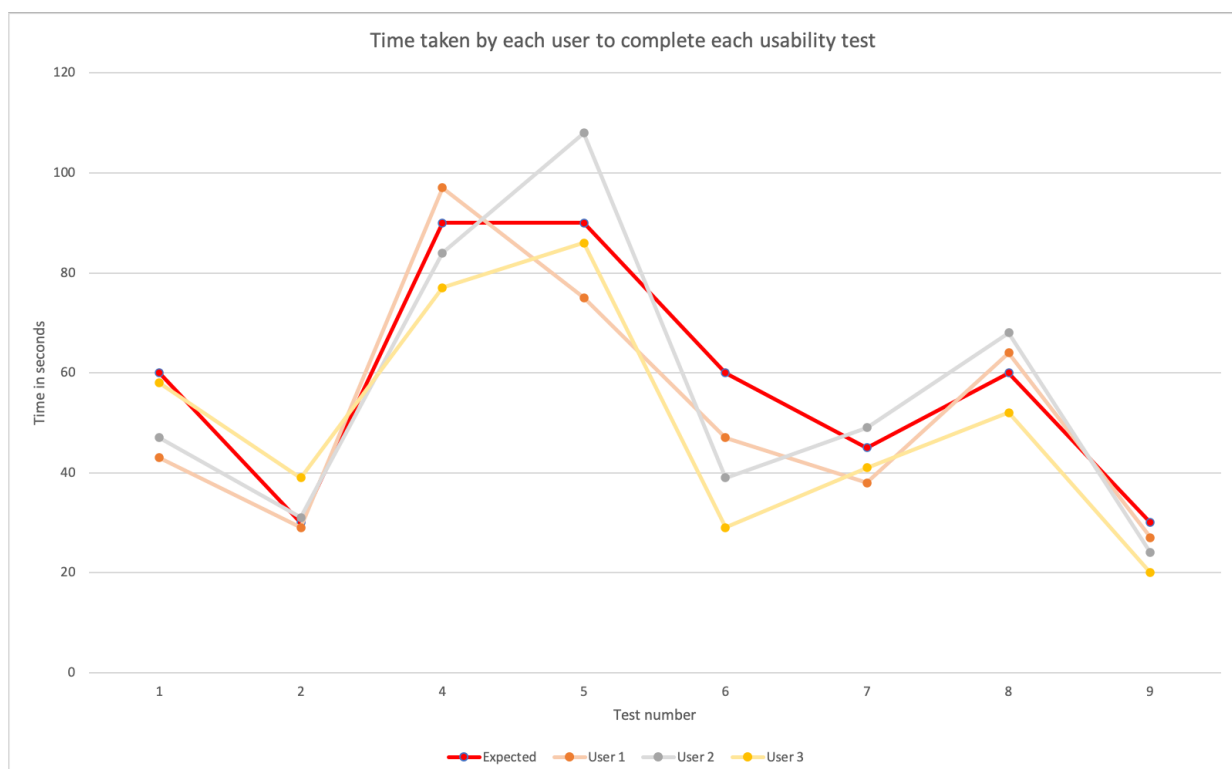
Out of the 27 tasks performed, 25 passed. 2 tasks partially failed. 1 of these was regarding adding a place to the walk. User 1 thought that by pressing the 'Add a place' button in the FAB (Floating Action Menu) this would allow the user to drop a pin or type a place but then was prompted to select a place on the map. After the prompt understood this feature but thought it could be explained better. The other failure was when a user selected the wrong activity and then could not edit or delete this from the walk. This is a flaw in the system to not be able to edit or delete an activity/mood/image/place from a walk. Users can only delete an entire walk.

Overall, all three users found the application simple and easy to use. This was the most common comment regarding the system. They thought it was intuitive and easy to navigate through the use of the bottom navigation. They liked the colour scheme and logo incorporation. Users liked the fact that the dog was incorporated into the design and personalised views such as 'Sarah and Baxter's Walk'. Users liked the filtering by mood, one did question if filtering by mood applied to just start mood/current mood/end mood but thought all moods was the best approach which is what is implemented. Users liked the story like feature when tracking a walk live, with icons appearing where they selected their content. Users found the sharing and deletion of walks easy and overall had very little issues with adding content to walks and filtering walks.

Users had some suggestions on how to improve the application or features they would find useful. One of these was the ability to add more than one current mood or one mood for each time asked. One user didn't like the different screens to add content and would prefer this is one cohesive popup style screen. One found it slightly tricky to find the filtering methods of walks. All users did make a comment on the statistics and thought a more

engaging statistics page would be useful which provided averages over certain weeks/months to track their moods and activities over time. Although did like the simple nature of the statistics. Another suggestion was to make the extra content screens only have one exit point instead of two ('x' in top of screen or save with a null value), one user suggested a popup confirmation that no mood/activity would be recorded.

In terms of the time it took each user to complete the tasks, they mostly were around the time limit set out. Below is a graph which shows the time taken by each user for each task relative to the expected time result. Please note task 3 is not included due to this being the length of the walk itself.



It can be seen that most tasks are completed in and around the expected time. User 2 has some issues with task 5 and 8. User 2 is the user with the least amount of technical ability.

These findings are invaluable and great foundation to start future work on and a place to begin to create an even more intuitive and useful application to end users.

8. Conclusion

This project intended to offer dog walkers a simple way of tracking their walks and mood. The Android application was intended to be used by regular dog walkers who want more of an insight into their walks, what they did on the walks and how they felt during these walks. Each walk was intended to be like a 'diary' entry as such which would show the route, moods recorded, activities recorded, places and images. On top of this, some users may be wishing to effectively and easily search and filter their walks. They may wish to see what walks they were happiest on or in turn saddest on, when these were, what they did and where they were.

The project's main aim was to provide a more intuitive and collective application for those who journal but wish to tie this in with their dog walks and daily exercise throughout the day. Currently, there are applications for journaling moods and tracking dog walks but there is no simple and effective application to merge these two ideas. The application intended to provide:

- Track and log walks that are taken with their dog.
- Track specific details of each walk, such as: route taken, time taken, distance covered.
- Journal the individual's emotions and how the walk made them feel (start, during and end of walk)
- Add details to each walk, such as any places which they visited, and activities performed. E.g., fetch, training
- Attach pictures of the walk to each walk
- Filter/Search past walks by mood/time
- View personalised statistics of users walks

All of the main functionality that was set out to be implemented was implemented in time. Regarding the User Persona goals, all of the goals set out for the two personas were achieved.

There are some limitations with the ordering of past walks when using certain filters (Weekly and Monthly) however the sorting was implemented. Personalised statistics were implemented into the system in time, however after some feedback potential end users would prefer these statistics to be more intuitive and useful. The statistics currently are most common start/end mood and during, as well as most common activity and average distance. Through usability testing, I found users would prefer more of a time based statistic feature, such as average mood that week or correlations of moods to activities and places. Another limitation of the current implementation is the limited searching ability for activities and places, this could be incorporated into an advanced search or filtering feature of past walks.

One of the original requirements (could have requirement) was not met. This is the implementation of the notification feature which would indicate to the user when they have walked a pre inputted number of km for that day. This feature was not implemented due to

time constraints but due to it being a could have requirement this has not had any impact on the basic functionality of the application.

During the design and implementation, a focus was made on usability and making the application as easy and intuitive for the user to interact with. I followed various guides set out by Android throughout and conducted a user interface review on a prototype, which led me to find usability issues and implement features which make the application easier to use such as the bottom navigation bar.

Overall, the implemented system met nearly all of the expectations set out and did so in a structured and defined manner. The additional features could be implemented with more time and research and the system has been designed to allow for the expansion of features easily.

9. Future Work

The application has been completed to a high standard, achieving nearly all of the goals I set out to. However, there are a few features implemented in the system which have not been completed to as high a standard as previously wished. These features would be the first modifications to be made given more time. On top of these features, additional features and views which could be implemented were discovered through user usability testing. By implementing these fixes and new features improvements could be made to the current system in order to improve the user experience as a whole.

Firstly, given more time one of the first improvements to make would be to fix the bug which causes the time filters (Weekly/Monthly) to not display walks in ascending order. This would make it easier and quicker for users to find specific walks. Functional Requirement 15 was not implemented in time and this was a notification based feature which would notify the user when they have reached a certain pre inputted target. This feature seemed popular in the User Requirement Questionnaire and would therefore be useful to implement and test.

Through the implementation phase, as time went on I did realise that there are some database design flaws. There could be a slight tweak in the way the startMood, currentMood and endMood are stored in the system. These moods could be stored in a Hash Map instead of strings. The Hash Map could store the string value of the mood as the key and the location it was set at as the this would allow for multiple moods and locational data stored too. This would be an improvement on the current schema which holds currentMood as a string and currentMood place as a GeoPoint. Therefore less fields are needed.

Through usability tests and speaking to potential end users, several features or modifications to existing features were brought up. One common feature which was brought up on several occasions is a more personalised statistics page. The application currently provides statistics of a simple and text form. However, an improvement of this would be more statistical views of these walks such as moods per week/month or recommendations of walks which make the user most 'happy'. This would make the application even more personalised and make these statistics more useful to the end user. A further improvement to the application would be a more advanced searching system. This could search for activities and places and would allow the user to have a control over searching abilities.

An important feature which would be useful to be implemented would be allowing the user to edit a walk live. This would be tremendously useful for the user in case of any mistakes made adding content to the walk. This would improve the tracking of the walk itself. A further improvement to the tracking of the walk itself would be to allow the user to add several moods instead of one mood. This is because human moods are complex, and humans can feel several moods at once and wish to record these.

Finally, in order to make the experience of the application as personal to users as possible, more personalisation could be incorporated throughout. For example, users could choose their own colour scheme or set of emojis and activities. It would be useful for the user to be able to add their own activities as it is common for users to do very different activities with their dogs.

10. Reflection

This entire project has been a tremendous learning experience in all aspects. The project has been invaluable for both technical skills and softer self-management skills. Both of these skills are vital for progressing my personal development and capabilities as well as working within the software development sector.

Firstly, this project has allowed me to progress and enhance my previous Java skills. It allowed me to put into practice techniques and skills learnt from my previous years at University, e.g. object orientated concepts as well as skills picked up from my year on industry. I have a new found appreciation and passion for Android Development. Creating an Android application was something I had never done before and from this project I have been able to pick up on a new skill and hobby. Starting the project with zero previous Android Development skills did prove to be a struggle, I underestimated the time it would take to understand with the more advanced features of Android development. I followed tutorials online through Udemy which provided me with a good overview, but advanced features were not always relevant. Only at a slightly later stage in the development progress did I discover CodeLabs tutorials which cover advanced features which were more relevant to this project. These tutorials are also more up to date and would have been a better use of time.

Another issue I faced was in the later stage of development, I was making small changes in order to finish off the implementation of some features. As set out in my initial report, I decided it was best to write the report as I implemented features in order to keep on track. These small code changes at times would mean some design structure would need to change or the way the application ran would change. This then meant I had to continually change my report as I went along, which in turn took up time. Going forward in future projects, I would aim to have the implementation completed slightly earlier and therefore alleviate too many changes to the report in a later stage of the project.

Regarding the organisation of the project, overall I think this was kept to a high standard. I followed my plan relatively closely as set out in the initial report and as I was following an Agile approach, this allowed for continual review of requirements and allow me to make additions or changes when necessary as research and implementation took place. I made use of a Trello board to ensure I was on hitting my personal deadlines for features and report writing and this was an invaluable tool to do so. Communication with my supervisor was consistent throughout and this was crucial to the success of this project, with weekly meetings and email communication throughout so that progress of the project was clear at all times. In future projects, I will ensure there is a high level of communication and self-reflection throughout. Setting personal targets and deadlines were key in delivering this on time and to a high standard, as well as consistent reviews with my supervisor.

Version control (GitHub) was not used consistently throughout the project. Luckily, there was no outages or accidental deletions, however this I do know was a risk. I created backups roughly every few days into a zip file with my latest report and application source code. I did

however use OneDrive to write my report on to ensure a backup was always present. In future projects, I would definitely make more use of version control to limit the chances of any deletions or mistakes, this is something I was lucky with for this project, but I have taken away the risk of losing material and would not allow this to happen again.

Overall, this project has helped me in my determination and dedication. I had set high standards and ambitious objectives, this kept me motivated and interested throughout. I have thoroughly enjoyed seeing an idea on paper come to fruition as an application. This has been exceptionally rewarding and a great learning experience being able to take away key fundamental methods to put in place in my future learning and career which will help me produce work to a high standard.

11. References

Armstrong, Sarah. 2017. Android Navigation. Available at: <<https://medium.com/curated-by-versett/featurecrushfriday-android-navigation-ab4329f1cfd4>> [Accessed 1 Apr].

Andress, J. 2019. Foundations Of Information Security.

Android Developers. 2020. Android Developer Guide. Available at: <<https://developer.android.com/>> [Accessed 15 Jan 2020].

Android Material Guide. 2020. Material Design. Available at: <<https://developer.android.com/>> [Accessed 18 Mar 2020].

Balsamiq. 2020. Introduction to Balsamiq Cloud. Available at: <<https://balsamiq.com/wireframes/cloud/docs/intro/>> [Accessed 17 Mar].

Bumpteck. 2017. Glide V4. Github. Available at: <<https://github.com/bumpteck/glide/wiki>> [Accessed 13 Apr].

Clark, Emily. 2019. Do people trust apps that track their location?. The Manifest. Available at: <<https://themanifest.com/mobile-apps/do-people-trust-apps-track-their-location>> [Accessed 25 Apr]

Derks, D, Bos, A and von Grumbkow, J, 2007. Emoticons and Online Message Interpretation. Social Science Computer Review, 26(3), pp.379-388.

Firebase Firestore. 2020. Firebase. Available at: <<https://firebase.google.com/products/firestore>> [Accessed 25 Jan].

Firebase Firestore a. 2020. Compound queries in Cloud firestore. Available at: <<https://firebase.google.com/docs/firestore/query-data/queries>> [Accessed 14 Apr].

Futurice. 2019. Android Best Practices. Github. Available at: <<https://github.com/futurice/android-best-practices>> [Accessed 15 Apr].

Google Marketing Platform, 2020. Marketing and Analytics Resources. Available at: <<https://marketingplatform.google.com/about/resources/>> [Accessed 21 Mar].

Hakami, S.A. (2017). The Importance of Understanding Emoji: An Investigative Study: University of Birmingham.

Ingram Samantha. 2016. The Thumb Zone: Designing For Mobile Users. Available at: <<https://www.smashingmagazine.com/2016/09/the-thumb-zone-designing-for-mobile-users/>> [Accessed 21 Mar].

MIT. 2018. Reading 13: Graphic Design. Available at: <<http://web.mit.edu/6.813/www/sp18/classes/13-graphic-design/>> [Accessed 13 Feb].

NHS, 2011. Dog Walking 'Is Good Exercise'. [online] nhs.uk. Available at:
<<https://www.nhs.uk/news/lifestyle-and-exercise/dog-walking-is-good-exercise/>>
[Accessed 23 Jan 2020].

Nielsen, J, and Molich, R. (1990). Heuristic evaluation of user interfaces, *Proc. ACM CHI'90 Conf.* (Seattle, WA, 1-5 April), 249-256.

Professor DK. 2017. Simple Login App Tutorial. Available at:
<https://www.youtube.com/watch?v=IF5m4o_CuNg> [Accessed 14 Jan 2020].

Sample, I. 2009. A Diary Makes You Happier And Helps Brain Cope With Emotional Upsets, Psychologists Say. [online] the Guardian. Available at:
<<https://www.theguardian.com/science/2009/feb/15/psychology-usa>> [Accessed 23 Jan 2020].

Udemy. 2019. The Complete Android N Developer Course. [online] Available at:
<<https://www.udemy.com/course/complete-android-n-developer-course/>> [Accessed 9 Jan 2020].