A Personal diary application that uses user location and emotion tagging to enrich the user experience

Author: Nathan Greenslade
Supervisor: Alia I Abdelmoty
Moderator: Xianfang Sun
Qualification: BSc Computer Science
Institution: Cardiff University
School: Computer Science and Informatics
Submission Date: 11/05/2018

Abstract

Diaries offer users a way to record their experiences and thoughts as snapshots of their life. They allow for enjoyment through reflection and are often used to track a user's progress and inspire the change they want to achieve. An important aspect of a diary is the ability to record our emotions during particular activities and places, they often tell us more about ourselves then we realised we knew. Mobile technology provides a portable platform on which to make this experience easier and with greater potential functionality, it is the aims of this project to provide a way for users to further achieve the objectives of a diary, recording their daily lives while learning even more about themselves through extra functionality that can be provided though technology.

This project will record the research, design and implementation of a diary application on the Android platform that uses the user's location to tag entries, the user will also be able to tag entries with emotion to allow for the generation of interesting and useful statistics about the user and the locations they visit.

The implementation of the project will be followed by extensive testing and evaluation to assess the strengths and weaknesses of the project and the extent to which the goals and objectives have been met.

Acknowledgements

I would like to thank Alia Abdelmoty, my project supervisor, for her continued support throughout the project. Her help and advice has been crucial for the continued progress of the project which would not have reached completion if it was not for her help. In particular I would like to thank her for her understanding and help concerning personal problems I faced in the early stages of the project.

I would also like to thank my friends and family for their support during the project, thank you to those who volunteered as testers, you have been a source of great help.

Contents

1.0 Introduction

2.0 Background

- **2.1 Location**
- **2.2 Application Research**
- **2.3 Emotion**
- **2.4 Identified Problem**
- **2.5 Constraints**
- **2.6 Methods Used**
  - **2.6.1 Agile / Iterative Project Lifecycle**
- **2.7 Applications and Tools Used**
  - **2.7.1 Android Beginners Course**
  - **2.7.2 Balsamiq**
  - **2.7.3 Android Studio IDE**
  - **2.7.4 Google Android Support Library**
  - **2.7.5 Google Drive**
  - **2.7.6 Visual Paradigm**

3.0 Specification

- **3.1 Personas**
- **3.2 User Requirements**
  - **3.2.1 Functional Requirements**
  - **3.2.2 Non-Functional Requirements**
- **3.3 Changes to Requirements**
- **3.4 Use Cases**

4.0 Design

- **4.1 Use Case Diagram**
- **4.2 Static Architecture**
  - **4.2.1 Package Diagram**
  - **4.2.2 Class Diagram**
- **4.3 Dynamic Behavior**
- **4.4 Information Architecture**
- **4.5 User Interface Designs**

5.0 Implementation
- **5.1 Emotions**
- **5.2 Location**
- **5.3 Database**
- **5.4 User Interface Features Available Across the Application**
- **5.5 Activity Lifecycle**

6.0 Testing, Results and Evaluation
- **6.1 User Interface Design Mock-Ups**
- **6.2 Testing System Usability and Real User Feedback**
- **6.3 System Completeness (Acceptance Criteria)**

7.0 Evaluation

8.0 Future Work
- **8.1 Functionality**
- **8.2 Structure and Implementation**

9.0 Conclusion

10.0 Reflection
- **10.1 Management, Organisation and Communication**
- **10.2 Learning**

1.0 Introduction

The aim of this project is to design, develop and implement a diary application for the Android platform. The diary application will aim to allow the user to achieve everything they could do in a paper-based diary, with added functionality that utilizes mobile technology.

The objectives of this project are to:

- Use user location to automatically give diary entries locational context.
- Offer the user Emotional tags to identify their feelings within an entry.
- Use emotion and location to produce statistics and information that will help the user learn about their routines and lives.

The scope of this project is the complete design, implementation and testing of a system that allows users to utilize mobile technology to achieve the most out of a personal diary application.

User testing will begin at an early stage to ensure designs are usable and that functionality is intuitive. Using the principles and design guides set by “Google Material Design”, along with other interface principles will ensure the correct implementation of the design. Later user-based testing will be used to evaluate the systems strengths and weaknesses, as well as assess to what extent the project aims and objectives have been met.

The report will detail the research and initial preparations that were undertaken early in the project. It will then outline the specification and design of the system that was used to create the working application. This will be followed by a detailed description of the implementation of the application and how each objective was achieved. Finally, the report will detail the testing and evaluation of the application, followed by a description of future work to be undertaken and a conclusion of what the project achieved and the lessons that were learnt.

2.0 Background

2.1 Location

User location has an increasingly important role within mobile technology. The increase in devices and applications that offer and use locational services has opened up functionality and data that has never been available before.

Location allows the user to receive specific content that is more personal to them, based on their location. It also offers the ability to offer location specific applications that use the user's location as a vital part of the service. The increasing use of location is creating a huge amount of user data that contributes to the generation of big data. This data has potentially massive applications for data mining, an area of increasing importance in marketing; retail; government data; scientific study; economic industries; and almost all applications where information is a commodity.

The project aims to create a diary application that uses the user's location to automatically tag entries. The use of locational data demands the consideration of the social and ethical implications that the collection of this data could have on the users.

The user should be aware of the data the application uses to provide its service. Therefore, the application will ask for the user's permission before continuing. The main features of the application will not be able to work without location however it should still be made clear to the user that this is the case.

The application will make the user aware of any background collection of data being collected while they are not running the application. This is for full transparency and the user will have the option to turn off this functionality if they choose to do so however, this would stop the user from being able to use the location history to add entries.

Finally, the user's personal data should not be used by any third parties or for any other reason than the functionality provided to the user. The system will ensure that user data collected is stored locally and that it will not be available to other applications on the device. As a further safe guard against the issues of user data, no connection to the internet will be needed to run the application, with no other services having access to the data it helps ensure its protection.

2.2 Application Research

The following section details applications related to the project that have been researched in order to learn more about the problem. The purpose of this is to determine the target user and explore some existing solutions, as well as learning the shortcomings of these current solutions. This has also helped to highlight the areas of the problem that have not been addressed, which will be the main focus the project is trying to solve.

A wide range of applications have been found that focus on different parts of the problem, while others try to address multiple parts of the problem. These included standard diary applications, mood trackers, location trackers and some that combined different elements, however no application found was using both emotion and location in a diary application or offered analysis of emotion and location together.

The following are the notes made about some of the more interesting application tested:

Daybook (Bighead Techies)

Rating: 4.6

Available at: https://play.google.com/store/apps/details?id=com.bigheadtechies.diary&hl=en

A simple diary application allowing the user to add entries, view entries and use the calendar as a planner.

Simple layout, the calendar view is a good way to order entries chronologically, user can add pictures.

Doesn't really offer much else in terms of functionality.

Reviews

"I was able to open and start writing immediately. It's simple and straight to point… My data is backed up to my Google Drive. Perfect for me!!!!"

"The plain simplicity of the app makes it great to use".

"I wish it was possible to change the theme and colours of the app."

Diary (PerfectlySimple)

Rating: 4.7

Available at: https://play.google.com/store/apps/details?id=com.mydearjournal&hl=en

A simple diary application where you can add and view entries, smileys can be tagged to entries and photos can be added.

Smileys help find related posts but no automation to help this, nice simple design although could be more intuitive. A search option allows the user to find entries that contain a key word or phrase.

No way to order the search for past entries or narrow down searches other than by text, for example can't narrow entries by date or smileys.

Reviews

"Love the little emoji that I can attach to a note and I love the search engine on it."

"Each time I tried to type it would move my keyboard so it was covering up what I was typing"

"I don't like the adverts"

Diary (Rucksack)

Rating: 5.0

Available at: https://play.google.com/store/apps/details?id=com.rucksack.diary&hl=en

A diary that offers more functionality then other simpler designs, weather tags can be added to entries, user can add photos and application has full speech capabilities.

The weather tag is a set of specific weather based smiley and considering their impact on user emotions would be an interesting addition to the project application. As well as offering speech to text capability it also adds great text to speech functionality, this would be a really nice extra to add with a simple one button interface.

Adverts are quite intrusive and the layout is generally quite cluttered due to offering functionality that is possibly unnecessary.

MoodCast Diary (2 App Studio)

Rating: 4.5

Available at: https://play.google.com/store/apps/details?id=com.journey.mood&hl=en

A diary app that offers emotional tagging and runs simple analysis on the data created. User can also create activities tags to define what they were doing in each activity

Nice simple interface, the emotions have the range very positive, positive, neutral, negative and very negative. This is quite easy to quantify but may lack the ability to show the subtle difference between different negative/positive emotions. I like the simple smiley design used in this. A mood chart shows the changing mood over time which is a nice feature, best day (most positive) is a good simple idea. The average positivity function is a great way to show the user how their actions have made them feel this month and also helps the user track whether they are having less or more positive experiences as they progress. User can use Facebook posts as entries, this is a nice little bit of integration with other applications.

The activity tags are a good idea although for the context of the projects aim I think they may overcomplicate the user interface and analysis beyond any real benefit.

Reviews

“Overall I love the stats and the calendar views!”

“I needed a way to log events that made me feel happy or sad and to reflect for cognitive behavioural therapy”

“Maybe add more moods or give the user an option to choose from many other moods to get it more specific….. It needs more statistics”

“Not a wide enough range of moods."

“Some mood tracking apps give too many mood scale options”

“No way to search previous entries though: no search bar. Feels like a monstrous oversight”

Daylio (Daylio)

Rating: 4.8

Available at: https://play.google.com/store/apps/details?id=net.daylio&hl=en

An application that primarily focuses on a user’s emotion at the time of entry, allows the user to create activities and tag entries with these activities as well as use a set of predefined activities.

A nice simple and easy to navigate design. Prompts user to enter emotion first which highlights main concern of the application. Mood count which simply counts the total for each mood is a nice quick piece of useful information. I really like the “Often done together” statistic which shows activities that often done together and the usual mood for these combinations. This is similar to what the project will aim to achieve with locational/emotional analysis in the application.

Activities is a nice idea but probably unnecessary clutter for the projects goals and objectives.

Reviews

“You can set the time it asks, customize and add new activities. I really like this because I can keep track of any negative events I have and it’s easy to search by activity and bring up all instances.”

“Like the way you can back up to Google drive.”

“Gives lots of statistics about how you are feeling which I think is very helpful.”

“Simple, visual, effective! I've always liked the idea of daily journaling but felt that an entry everyday would get monotonous either to write or read back. But Daylio's smilies makes it quick and easy, and short responses are perfectly fine.”

<u>Diaro (Pixel Crater Ltd)</u>

Rating: 4.4

Available at: https://play.google.com/store/apps/details?id=com.pixelcrater.Diaro&hl=en

A diary that allows the user to tag the entry with a location, locations are predefined tags that are generated with help from google maps support, users can select a location and make their own name tag or they can select any of google maps predefined destinations to use as a tag name for the given location. Good use of location but could be more automated in approach. User can define other tags for activities, although these tags are not processed to produce any other data. Another nice feature is a customisable folder system to keep entries of a predefined “type”.

Reviews

“Really good!..... could you please enable the option of moving the app to the SD card? It's getting heavy on the device.”

“Overall, this is an excellent app. My only serious concern is that it takes A LOT of phone storage once you start to use it consistently and are adding photos…… no option to export to an SD card”

“I like the app. But guys, pls understand! At times one may lose all the data and might need to restore. It’s a basic need! All the data, the diary entries, pics, etc collected over an year suddenly lost and can't be brought back”

“Password protected with a PIN number (easier to type than an actual password”

“But can you add voice notes option to the entries, please”

“It would be nice to have a '5 year diary' view where you can see what you did on the same day over the past 5 years.”

“I would like some more editing and writing options, for instance italics, bold lettering, and perhaps some font options and different colours”.

Brief Summary of features and preferences from application Users

Users of diary applications seem to prefer simpler designs with the possibility to customise these with colours or a theme. Any further customisation, such as being able to define personal tags, is also well received and the possibility of tagging activities is well liked. Emotion and location tags are highly well received as key features of many applications. Users prefer to see statistics and information through diagram, charts and other images, mostly because it's more interesting than constant text.

Users like flexibility within their applications, the ability to search by more than one attribute is received positively for example by tags, text or date/time. Text-to-speech and speech-to-text capabilities are a good way to increase the availability of the application and are well received for their novelty value with other users. Being able to store the diary entries on a separate medium also seems to be a big priority for specific users, those that are worried about losing data or filling space on their mobile device like the idea of removable storage of cloud storage.

Some users also like password features offered by some diary applications, although most phones come with this built in it seems popular among parents whose children use their phones. Users enjoy learning things about themselves, in particular the emotional analysis offered by some applications was positively received. Finally, there are a lot of complaints regarding applications that hosted adverts.

2.3 Emotion

Diaries are a largely emotional investment, they capture a moment and usually how we are feeling at specific points in our life. Emotions can also tell us a lot about how the world is affecting us, whether it be a location; an activity; or a particular event. The reason for including emotion tags in this project is to help capture that emotion in entries (without having to read through the text or use sentiment analysis), and to be able to use these emotional tags as the basis to search and learn things about ourselves.

Research of applications already available through Google's play store revealed a lot of information about what users liked and wanted to see in a diary application. In relation to emotion, it is fairly evident that users really like the idea of emotionally tagging entries. This allows them to track and review their moods. It is also apparent that using an image based interpretation, rather than text based one, was more satisfying for most users (for example, using Smileys or other images representing specific emotions). Most users prefer to tag entries with specific emotional types rather than rate how they were feeling on a quantitative scale. It is easier for the user to identify emotions such as sad, happy and angry than to rate how strongly they feel an emotional state. Although quantitative scales offer an easy platform on which to base analysis, it also limits the spectrum of emotions felt, it's hard to consider anger, happiness, sadness and other types on one scale. Any interface to do this would probably make for a more cluttered and complicated experience.

Consider Facebook's reactions, a replacement for their "like" button interface introduced in late 2015, where users had a very positive response to being able to show their feelings towards a post. This re-design has increased the use of the function and has the benefit of using emotion for added information and context. This function is so beneficial to understanding more about the user, that Facebook use reactions in their algorithms for their advert placement and content provided. Part of the popularity of this feature is in its simplicity. Facebook only offers "like"; "love"; "haha"; "wow"; "sad"; and "angry" reactions through visual representation, but these choices still allow users to define very definite feelings towards posts, content and adverts.

Papers, such as “Measures of Emotion: A Review” by Iris B. Mauss and Michael D. Robinson, show us that emotion is not an easy thing to calculate. In reality, calculating someone’s mood is more effective by ignoring our more familiar defined states:

“We suggest that measures of emotional responding appear to be structured along dimensions (e.g., valence, arousal) rather than discrete emotional states (e.g., sadness, fear, anger).”
“Measures of Emotion: A Review”, Iris B. Mauss and Michael D. Robinson

So using such emotion tags isn’t a perfect solution however, a simple sliding scale of 1-5 doesn’t cover a range of emotional factors, whereas the other often give a person’s relative feeling of the state they are in. Measuring such factors would also involve a heavy and busy interface to get an accurate result which we want to avoid. Also, there is the factor that people are often unable to say accurately how the feel with quantitative accuracy (also covered in the paper).

In summary, the application is designed to use a simple emotional interface giving 5 options, these options are “ecstatic”; “happy”; “ok”; “sad”; and “angry”. These cover a variety of the emotional spectrum providing data for emotional analysis but help keep the user interface simple.

2.4 Identified Problem

This next section briefly summarises the problems identified from the research phase of the project and where the implementation of a solution does not exist or could be improved upon.

Diaries are a way for users to record and reflect on their lives by writing entries on their experiences, the problem of pen and paper solutions is that these have to be taken wherever the user goes if they wish to write in the moment. This is not a very practical solution in all circumstances, meaning potential for spontaneous entries is limited. There exists diary applications that offer a solution to this problem on a varying degree of levels, the fact most users will almost always have their mobile phones with them means they have access to their diary at all times.

However, it is the reflection and learning element of a diary that computerized technology, in particular mobile technology, can improve upon. The current progress in data analysis opens up a massive potential to discover patterns and information about our routines and desires that we never would have thought about, and also helps quickly identify the ones we are vaguely aware of.

User location is an increasingly prominent topic of interest, especially in analytical use cases. Users will often have strong emotional responses to particular locations. There is no application that makes effective use of location, there are none that I found that allowed the user to create custom location tags and have an application that was “locationally aware” about where these tags referred to. The project aims to allow users to create their own location tags, making the locations far more relevant and personal to the user. These tags will be used as suggested locations the next time the user makes an entry in the location tags area.

There exists a number of applications that work with emotional tagging. A number of these solutions rely on a sliding scale of happy to sad, which doesn’t really cover the complexity of emotion. There are other applications that use emotion fairly well and provide basic analysis in regards to activity or weather tags. This project also aims to use the location and emotion tags together, to help show where the user associates positive and negative feeling, with the aim of extending this project to tell the users patterns about their life that they may not have even realised existed. This will provide the user with a much more insightful and powerful way to take steps to improve their lives above and beyond their own reflection.

2.5 Constraints

The biggest constraint on the project is time, the project was scheduled for 12 weeks (not including 3 weeks of holiday), this impacted how much could be achieved because research, design, implementation, testing and evaluation all needed to be covered in this period. During the project I also had other work commitments including a coursework project for my emerging technologies module that took approximately 2 weeks to complete. Illness and family matters that came up during the coursework also meant adjusting my work schedule. To manage time, a schedule was created as part of the initial plan and an appropriate time frame was given to the majority of the projects major tasks.

I had no experience with android development going into the project, although fairly comfortable with java, the use of xml files for layout meant this was a new way to code with java that took some getting used to. As well as slowing down the implementation process, it also made the project harder to plan as I was unsure as to exactly what could be achieved on the platform and how, especially in the time frame allowed. A number of learning materials and guides were considered and used to lessen the impact of this inexperience on the project.

The physical resources available to complete the project also constrained progress on the project. Android studio is not available on the lab computers within the university, this meant I had to complete all implementation and testing on my home laptop. To make matters worse my laptop broke during the early stages of the project making it harder to learn android and implement a solution, specifically the screen broke so I had to connect my laptop to a monitor and the keyboard did not work (requiring use of a usb keyboard).

This second problem caused one of the biggest constraints of the project, the x86 emulator used to emulate android projects runs on a kernel drive called HAXIM. To enable this requires going into the BIOS, to access this I need to use the keyboard on boot up, however my usb keyboard does not power up until windows starts running. This slowed down the testing of implemented features in the application as these had to be loaded onto a physical device every time. It also meant I had no access to test the application on devices that I could not physically find myself, as a result this application could not be tested on tablets and had limited testing on phones of different models and sized screens.

2.6 Methods Used

2.6.1 Agile / Iterative Project Lifecycle

The project followed an agile / iterative approach generally, however, some initial research and planning was undertaken as a foundation for the rest of the project. This initial phase was more in line with the waterfall life cycle. Some early research was carried out on what solutions to the problem already exist, the typical user, how emotion tags could be used, how user location could be used and further definition of the projects aims. This research resulted in the development of the applications requirements and some rough initial use cases and flows that helped get a feel for the system. This research and specification phase of the project was undertaken, alongside learning how to use android studio and how to implement basic functionality in the first 3-5 weeks of the project.

The design and implementation phase followed a far more iterative process, a large factor in this

was a lack of experience with developing android applications, experience with using android applications meant early design could be drawn up in balsmiq, but these were continually changed and refined as I discovered how to implement more and more of the systems features. To ensure early progress was made, a basic application was built and tested with more features added over time to ensure there was a working implementation within the project time frame, as a result certain features were not implemented until the final few weeks of the project.

2.6.2 Google Material Design

Most guides and tutorials focussing on Android development are usually influenced by and reference the Google Material design guides for user interface design. Most professional developers recommend these guides to conform to the high standards that are expected from applications in the Android community.

Personal experience and application research highlighted a high level of user interface design amongst applications related to the project, in particular, familiarity and standard methods to carry out regular functions such as navigation; displaying multiple contents on a page; and user input; help keeps user interfaces intuitive. These standards have been matched where possible within the design of the application.

The Google Material Design guides help enforce a standard of design that keeps functionality simple to use, lowers users learning time by making the interface intuitive and keeps elements tidy and in "logical sense". It also helps keep applications accessible by making sure element sizes and colours make the application usable by those with poor sight; hearing; and other disabilities; as well as coping with different languages and social implications. The well-tested and informed design principals of the Material design guide have been followed where possible.

2.7 Applications and Tools Used

2.7.1 Android Beginners Course

One of the first challenges associated with this project was learning the basics of the android platform. I have some experience in Java and a little in XML but this was a new way of using both, along with learning the basics of gradle build and project manifests.

A number of courses were considered including those offered at Udemy and Coursera, however a course provided by Udacity was ultimately chosen due to it being free and its recommendation by the google developer's web pages.

"Developing Android Apps", available at https://classroom.udacity.com/courses/ud851, is a 60 hour course that introduced me to all the basic concepts of android including the file structure, Gradle, the project manifest, layout files in xml and the bulk work of applications provided by java. It gave me an introduction to activities, intents, views, layouts and basic material design principles.

As well as this course the Android developers guides available through the google built web pages were also incredibly useful, in particular the design and docs pages.

The material design web references were also a great source of design inspiration and guidance.

Learning the Location Services

A large part of the implementation involved connecting the application to Googles location Services to establish the user's current location. Due to the relative success I had with the "Developing

Android Apps" course collaboration between Udacity and Google I decided to try the course they provide for Location Services. The course "Google Location Services on Android" available at: https://classroom.udacity.com/courses/ud876-1 seemed perfect at first but I soon found out it was using a deprecated set of API's, I had a similar problem with small parts of the previous course but was able to find work arounds by using newer API's. However although investing quite a bit of time in the course I eventually switched to another tutorial as newer API's worked quite differently to those used in the course.

I ended up using the Youtube video "Android FusedLocationProviderAPI Deprecated - How to use the FusedLocationProviderClient - Part 1" created by "north border tech training" available at: https://www.youtube.com/watch?v=4bNT-2r8NOA&t=731s. This used the up-to-date API's and was a great video to learn the Location basics, much of which was the foundation for my final code in the project.

2.7.2 Balsamiq

During the design phase, it was necessary to construct visual designs that would detail the user interface to give the applications implementation focus and purpose. Using wireframe software allowed the design to show the layout and look of the application, as well as basic functionality and flow between activities. A few options were considered for this task, there was a trade-off between available functionality, available resources, ease of use, flexibility and cost.

I considered a number of online options for wireframe software such as wireframe.cc and mockflow, however Balsamiq was the final choice. I was familiar with Balsamiq having used it for my HCI coursework in second year, it provided an option that avoided a large learning curve. There is not a huge range of different items to use but it covers most of the basic features, however it allows the designer to create new features and import these into the design suite. Importantly, the use of these basic items is quick and easy with a fair amount of options to create simple designs that fulfil material design guidelines. It offers a way to link buttons to tasks and activities allowing the designer to create mock up designs for human testers to trial. This might not be as close to the "finished application" feel that other software supplies, but a huge advantage of this simplicity is the ability to change designs quickly based on feedback and test the designs iteratively. Using Balsamiq was a great way to get quick feedback on design features and layouts, as well as get quick suggestions on what testers felt the application was missing. The first mock-ups were produced with just 2-3 activities to get feedback on general layout. As more features were added and tested, the design was updated to reflect changes and additions. The design shown later in the design section is the final iteration of a wireframe design used. Notes about steps towards this design regarding the progression are provided, as well as an overview of the main features added or changed after the design phase to improve the user interface.

2.7.3 Android studio IDE

"Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps".

This is available free to help encourage professional design that follows the material design guidelines, specifically built by google, it helps avoid issues that may be associated with 3rd parts IDE's and is reliably up to date with changes to the android system. This was chosen as a

recommendation from several guides to learning android, in particular the course I chose uses this IDE and so do most other tutorial videos available on the internet.

2.7.4 Google Android Support Library

The Android support libraries offer functionality that is supported by multiple API versions, they supply classes that require no version checking of the user's android version as this is handled automatically. This helps maintain cross version functionality and helps enlarge the target audience the application can reach. The libraries also provide additional classes not found in the standard android libraries, particularly those that offer support across multiple devices

2.7.5 Google drive

Google drive was used as a free way to upload and access documents during the design and implementation process of the project. Materials that were related to and would help with the planning phase of the project, were uploaded to a shared drive by my supervisor and the drive also proved useful when formatting this project report and seeking advice from my supervisor.

2.7.6 Visual Paradigm

Visual Paradigm has been used to create a number of the diagrams and visual aids within this report. It was chosen partly because it is freely available within the university's windows lab. The software is also very easy to use, pretty flexible to get customised results and creates professional looking diagrams and images.

3.0 Specification

3.1 Personas

In order to better understand the requirements of the application, two personas that detail two main types of user of the application have been identified. These personas do not represent a real person but describe the main thoughts and goals of the type of people that will use the software. This helps define what the application needs to achieve to meet user expectations and helps avoid writing self-focused software.

The first persona is based on a person that will use the application as an application based replacement for a pen and paper diary. The second details a person that is hoping to use the diary to learn something about themselves to improve their moods and routines.

Name: Amy Smith

Status: Paperback Diary User

"I really want an easy way to be able to add to my diary where ever I am, whatever I'm doing."

Amy is a 22 year old Student who has kept a paperback diary since she was 11. She loves being able to record her thoughts and reflect on her day and also read back and remember her life experiences from previous months and years. Amy would like an easier way to keep her diary with her at all times, allowing her to make entries more quickly. Amy would like an application on her mobile phone that allows her to keep track of her life wherever she goes and add entries at the end of the day, that are automatically ordered by date and time. Amy loves being able to look back at particular moments in her life at a certain time, location or when she felt specific emotions or milestones. She would also like a more interactive way to keep track of entries, in particular she would like to track locations and tag posts with emotions so she can look specifically for entries when she was happy, sad or otherwise. Amy would also be happy to customise the diary or individual entries such as

different themes or text colour/styles and sizes. A concern Amy has with switching to an application is the data she has lost before when her phone has broken, she would feel her memories were far safer if she had a way to back them up.

Amy's Goals:

- Amy wants a diary application that is quick and easy to use as a substitute for her paperback diary
- Amy wants to be able to look through past entries with minimum effort ordered by date/time.
- Amy wants to be able to search for diary entries from a given location or had a particular emotional response from her.
- Customize the diary to make it look and feel more personal.
- Ability to back up diary entries in case of emergency

Name: Peter Green

Status: Reflective application user that wishes to improve their routines and life.

"I'd like a way to track my location and emotions, to identify the positive and negatives effects that events in my life are having."

Peter is a 31 year old professional who feels like he struggles to organise his life to get the most out of his time. He has a busy working life and a number of daily activities and routines and often feels he is not using his time effectively and that this is effecting his satisfaction and personal mental health. Peter wants a relatively quick and easy way to record his daily routines, his thoughts and to track his moods and locations. He hopes that this will allow him to reflect on his routines and make positive changes to his life that make him feel more fulfilled. Peter would like to be able to easily see how his daily activities are having an effect on him and would also like to be given back some feedback, regarding patterns of positive emotion and possibly highlight areas in his life that are causing stress and other negative emotions.

Peter's Goals:

- Peter wants an application that he can use to keep track of the daily events in his life.
- Peter would like to be able to see how his actions are having an impact on his mood by tracking his emotional status during particular activities.
- Peter would like to get feedback on what events and locations appear to impact positively on his life and those that seem to impact negatively.

3.2 User requirements

The following section details a set of initial requirements that were created within the first few weeks of the project. The requirements were carefully chosen to avoid the need for large changes during the project and detail all the requirements the user has but were based on underlying assumptions. Due to the agile method used during the project, some slight changes were made to these requirement based on new knowledge gained during the planning and research phase of the project. These requirement changes have been detailed below the initial list of requirements.

3.2.1 Functional Requirements

Must Have

1. The application will allow the user to add a diary entry at any time.

- The entry will automatically be tagged with the user's current location.
- The entry will automatically be tagged with the current time and date.

Acceptance criteria:

- The application allows the user to create an entry and saves this in a database local to the application.
- The application is able to receive the user's current location.
- The application offers location suggestions within 50m of the user's current location.
- The user can add a new location tag for the current location.
- The entry is tagged with a timestamp automatically.

2. The application will allow the user to add entries by viewing the locations they have previously visited at a given time/date.

- The user will be able to select a time/date from a list of locations previously visited.
- The user will then be able to add an entry for that specific time.

Acceptance criteria:

- The application collects user location at regular time intervals throughout the day.
- The application presents location history to the user.
- The application allows the user to add entries with location and time automatically filled with the relevant location history data.

3. The user will be able to emotionally tag diary entries with a pre-defined set of emotional tags.

Acceptance criteria:

- The user is presented a set of 5 emotion tags to tag their entry with.
- This emotion choice is saved with the entry.

4. The user will be able to view previous diary entries.

Acceptance criteria:

- Diary entries are presented within the home screen.
- The user can view a specific diary entry by pressing on it.

5. The application will have functions to analyse and display statistics to the user. (This will be more strongly defined later in the projects planning phase).

Acceptance criteria:

- The user can view the top 5 locations tagged with a specific emotion.
- The user can see the emotion count for a location.

Should Have

6. The user should be able to edit entries after they are saved.

Acceptance criteria:

- The user can edit the text content of a diary entry after it has been submitted.

7. The user should be able to delete entries after they are saved.

Acceptance criteria:

- The user can delete any entry saved in the database.
- The user is asked to confirm deletion before entries are removed.

8. The user should be able to search for diary entries by text content.

Acceptance criteria:

- The user can search for diary entries by text.

9. The user should be able search for entries by location.

Acceptance criteria:

- The user can select a location from a list of location tags, the user can view all entries tagged with this location

10. The user should be able to search for entries by emotional tag

Acceptance criteria:

- The user can select an emotional tag, all entries tagged with this emotion are displayed

11. The user should have the ability to back up the diary contents via cloud storage or SD CARD

- To save phone memory.
- In case a phones memory is lost.

Acceptance criteria:

- The user can export their data to a SD CARD or Cloud Storage.

Could Have

13. The application could show the user the diary entries that relate to the current day one week, month or one+ year ago.

14. The application could allow the user to select previous entries by use of a GUI calendar.

15. The user could have the ability to change the applications "Theme".

16. The User could have the ability to change diary entry settings such as text colour, font, size etc.

17. The user could have the ability to attach photos to entries.

18. The user could have the ability to add weather tags to entries.

19. The user could have the ability to add activity tags to entries.

20. The user could have the ability to add text entries via speech to text recognition.

21. The application could read text entries back to the user via text to speech capabilities.

22. The application could use social media such as Facebook and Twitter to dynamically generate entries.

Will Not

12. Contain Adverts.

Acceptance criteria:

- The application contains no adverts.

3.2.2 Non-Functional requirements

1. Diary entries should be ordered by time/date as default.

Acceptance criteria:

- Diary entries are displayed in descending (newest first) time order.

2. Application should have a simple easy to use layout.

Acceptance criteria:

- Application text entry does not obscure relevant text box.
- Application is never in a state where the user can't access the home screen.
- Application adheres to the Android Material Design standards where possible and does not breach them.
- The application has an average System Usability Scale score above 68%.
- The application reacts to the user's orientation in a responsive manner.
- The application scales well to larger devices such as tablets.

3. Will be reliable

Acceptance criteria:

- The application does not crash due to unexpected errors more than is reasonably acceptable (The application does not crash more than 1% of the time).
- All patterns/statistics generated process and display a result within 2 seconds.
- Recycler views generate contents within a second.
- Users take no longer than a couple of minutes to complete application functions (Add Entry, View Entry, Delete Entry, Edit Entry and View Patterns/statistics) during testing.
- The application can access the user's location at least 95% of the time.
- The application can be run on all currently supported android versions, this includes every android version from 5.0(lollipop) upwards.

Will Not

4. Require access to the internet to carry out important functions.

Acceptance criteria:

- The application does not attempt to access the internet.

3.3 Changes to Requirements

Must Have

1. The application will allow the user to add a diary entry at any time.

- The entry will automatically offer location choices based on users previous location tags.
- The entry will automatically be tagged with the current time and date.

Acceptance criteria:

- The application allows the user to create an entry and saves this in a database local to the application.
- The application is able to receive the user's current location.
- The application offers location suggestions within 50m of the user's current location.
- The user can add a new location tag for the current location.
- The entry is tagged with a timestamp automatically.

Later in the design stage, it was noted in a meeting with the supervisor that automatically tagging location would require the application to pull qualitative information about the location, regarding its human readable name such as an address. This has a couple of issues. Firstly, using such a service as google maps API to do this would require internet access, something me and my supervisor agreed was preferably avoidable, especially for a main function/use case.

Secondly, there may be areas where a recognisable address would not be found using such an API, also an address as a location tag might not be very meaningful to the user. As a result, it was decided that user defined tags would have a lot more meaning to a user, and make the application more personal, something important for those that use a diary. The application automatically gets the users current location. This is then compared to the location tags previously defined, those close to the current location co-ordinates are offered as suggestions. The user can also create a new location tag at this point that users the current latitude and longitude as the exact location.

3. The user will be able to emotionally tag diary entries with a pre-defined set of emotional tags.

- Ecstatic
- Happy
- Ok
- Sad
- Angry

Acceptance criteria:

- The user is presented a set of 5 emotion tags to tag their entry with.
- This emotion choice is saved with the entry.

The emotion tags used have now been defined

4. The user will be able to view previous diary entries.

- From the main activity where they will be displayed in a scrollable view.
- By clicking on a particular entry on the main activity the user can view the whole entry in detail.

Acceptance criteria:

- Diary entries are presented within the home screen.
- The user can view a specific diary entry by pressing on it.

The way entries will be viewed is further defined to avoid ambiguity.

5. The application will have functions to analyse and display statistics to the user, these will include:

- Top 5 locations with a particular emotion. The user will be able to choose an emotion and see the 5 Locations that have the highest count of this emotion.

- Location emotion count. The user will be able to select a location and see how many times each emotion is tagged with that location and a percentage breakdown of ratio

Acceptance criteria:

- The user can view the top 5 locations tagged with a specific emotion.
- The user can see the emotion count for a location.

Two pattern/statistics that will be available have been defined.

Should Have

1. The user should be able to edit entry text after the entry is saved.

Acceptance criteria:

- The user can edit the text content of a diary entry after it has been submitted.

The original requirement implied all details in the entry could be changed. A user may make spelling/grammar mistakes or may remember more details at a later time and so may want to edit the contents. However, it is logically unnecessary to change the location and time as these would not have changed from the initial time of writing the entry.

3.4 Use Cases

Use Case 1 - Add a diary entry

The user will be able to add a diary entry by selecting the add entry button, they can then add text entry and an emotional tag and save the entry. The application will check the user's location and offer suggestions automatically based on previous user defined location tags and/or offer the user to add a new location tag for this location. Time and date will be added automatically based on the data generated by the devices time/date.

Pre Conditions: User is in the main activity, Location is switched on.

Basic Flow

1. Select "add entry" button

The user selects the add entry button from the main activity and is taken to the "add entry" activity.

2. Select a location tag

The application shows the user, in a drop down menu, a selection of previously defined locations that are within a 50m range of the current user location. The user selects one of these choices as their location for the entry.

3. Write entry

The user taps the text entry field and writes their diary entry using built-in android keyboard. The keyboard does not interrupt or cover the text field allowing the user to see what they are typing to help avoid errors.

4. Select emotional tag

The user selects an emotional tag from the available choices via a radio button configuration.

5. Select “Finish Entry”

The user presses the “Finish Entry” button to save the entry.

Alternate Flows

1. Cancel entry

The user cancels the diary entry by pressing the back button, the back button is available at every activity and takes the user back to the previous activity. The diary entry will be discarded and not saved.

2. Emotional tag, Location Tag or Entry text not entered or selected

If the user forgets or misses out a field in the add entry activity then they will be presented with a pop up upon pressing “Finish Entry” that informs the user what is missing.

3. Close application

The user closes the application by shutting down the application from the app management control and the application finishes.

4. Add new Location Tag (Use Case 2)

At Step 2 the user chooses to add a new location tag instead of choose a pre-selected location choice.

5. Select an option from the tab menu bar.

Rather than press back, the user selects one of the options available from the tab menu that is available in all activities.

Use Case 2 - Add New Location

The user needs to tag this entry with a new Location that they haven’t added before, they can choose to “Add new Location!” in the location drop down menu which reveals a text area to write a new location name. Saving the entry adds the new location to the database.

Pre Conditions: User is in the add entry activity, Location is switched on

Basic Flow

1. Select “Add new Location!” from the location drop down menu

User selects the “Add new Location!” option from the locations drop down menu, this reveals a previously hidden text space.

2. Type location name into the text area revealed

The user types in a location name for this location in the text area that appears after selecting “Add new Location!” When the entry is saved the location tag will automatically be added along with its latitude and longitude (taken from the current user’s location) to the location tag database.

Alternate Flows

1. “Add new Location!” is already selected in the location drop down menu.

This will happen when there are no previous locations added that fall within 50 m of the user’s current location, therefore the user will automatically have to create a new location.

Use Case 3 - Add a diary entry for a location in the location history

The user can add an entry for a time/date and location that has happened in the past, the application tracks the user at regular intervals allowing them to add entry’s for these locations later in case they are busy. The user uses a calendar interface to pick a day and then chooses a time/location to write an entry for.

Pre Conditions: User is in an application activity.

Basic Flow

1. Select location history tab

The user selects the locational history tab from the menu bar, this takes them to the calendar selection activity.

2. Select a day from the calendar display

The user selects a day from the calendar interface to choose the day they would like to see location tracking for.

3. Select a Location/Time Instance

The user is presented with a list of time and the locations they were located, at that time, for example a location for every hour of the day. The user then selects one of these to create and entry for that time/date and location.

4. Add entry (Use case 1 without the first step)

The user is at the add entry activity and enters the details necessary to add an entry, then saves this.

Basic Flow

1. Press back button

The user can press the back button built into the android system at any time, this will take the user back to the previous activity. At the entry screen it will take them back to the locations and times activity, at this activity the back button will take them to the calendar selection activity and at this activity it will take the user back to whatever activity they were in when they selected the location history tab.

2. Close application

The user closes the application by shutting down the application from the app management control and the application finishes.

3. Select an option from the tab menu bar.

The user selects one of the options available from the tab menu that is available in all activities, this will take them to the associated activity.

Use Case 4 - View diary entry

The main activity will display a scrollable view of all entries in chronological order, in order to view an entry the user simply has to tap the entry to bring that entry up in full screen in a separate activity.

Pre Conditions: User is in the main activity.

Basic Flow

1. Scroll through entries

The user can scroll through entries by touch to find the entry they would like to view

2. Tap chosen entry

The user taps the entry which takes them to the view entry activity. This activity displays the time, date, emotional tag and location of the entry above the full entry's text in a scrollable view (if scrolling required due to large amount of text).

3. Press back to go back to previous activity.

The user can then return to the previous activity by using the back button.

Alternate Flows

1. Close application

The user closes the application by shutting down the application from the app management control and the application finishes.

2. Select an option from the tab menu bar.

Rather than press back, the user selects one of the options available from the tab menu that is available in all activities.

Use Case 5 - View Patterns and Statistics

The user can view various patterns and statistics regarding their emotional response to locations and how their routines are affected by this.

Pre Conditions: User is in an application activity.

Basic Flow

1. Select "Patterns" tab in the menu bar

The Tab Menu Bar Is available from all activity's and is a link to the apps main activities. The user selects the tab "Patterns" which opens the patterns activity.

2. Scroll through patterns/statistics

The use can browse various statistics and patterns such as top 5 locations for an emotion or the emotion count of a particular location. All these stats are contained in a scroll view for easy browsing containing them in one activity.

Alternate Flows

1. Close application

The user closes the application by shutting down the application from the app management control and the application finishes.

2. Select an option from the tab menu bar.

Rather than press pattern, the user selects one of the other options available from the tab menu that is available in all activities.

View an Emotions top 5 locations

Pre Conditions: User is in the patterns activity.

Basic Flow

1. Scroll to “Emotions top 5”

Scroll to the “Emotions top 5” statistic section.

2. Select an Emotion

The user can select any of the 5 emotions available from a drop down menu to see the top 5 locations that are associated with that emotion.

Alternate Flows

1. Close application

The user closes the application by shutting down the application from the app management control and the application finishes.

2. Select an option from the tab menu bar.

The user selects one of the options available from the tab menu that is available in all activities.

Use Case 6 - View a Locations Emotion count

Pre Conditions: User is in the patterns activity.

Basic Flow

1. Scroll to “Emotions Count”

Scroll to the “Emotions Count” statistic section.

2. Select a Location

The user can select a Locational tag from a drop down menu to see how many times they have tagged this location with each individual emotion.

Alternate Flows

1. Close application

The user closes the application by shutting down the application from the app management control and the application finishes.

2. Select an option from the tab menu bar.

The user selects one of the options available from the tab menu that is available in all activities.

Use Case 7 - Delete a saved entry

The user can delete a diary entry by viewing it then selecting the delete button, a pop up text will ask the user to confirm this choice.

Pre Conditions: User is at the main activity.

Basic Flow

1. Scroll through entries

The user can scroll through entries by touch to find the entry they would like to view

2. Tap chosen entry

The user taps the entry which takes them to the view entry activity. This activity displays the time, date, emotional tag and location of the entry above the full entry's text in a scrollable view (if scrolling required due to large amount of text).

3. Select delete button

User presses the delete button, this brings up a pop up text that asks the user to confirm the action "Are you sure you want to delete this entry?"

4. Select Yes button

The current activity closes and returns to the main activity removing the entry.

Alternate Flows

1. User selects "No"

The deletion is cancelled and the user returns to viewing the entry.

2. User Presses back

The deletion is cancelled and the user returns to viewing the entry

Use Case 8 - Edit a saved entry

The user can select an entry and edit the text of the entry, the user may want to do this to change mistakes such as spelling or grammar or they may have remembered extra details they would like to add.

Pre Conditions: User is at the main activity.

Basic Flow

1. Scroll through entries

The user can scroll through entries by touch to find the entry they would like to view

2. Tap chosen entry

The user taps the entry which takes them to the view entry activity. This activity displays the time, date, emotional tag and location of the entry above the full entry's text in a scrollable view (if scrolling required due to large amount of text).

3. Select Edit Entry Button

User presses the edit entry button that leads to the edit entry activity, this is essentially the same activity but the entry's text is editable.

4. Edit Text

The user changes the entry's text where they see appropriate.

5. Select "Save" Button

Once the user is happy with their edit they press the save button to save the changes. This closes the edit entry activity and takes the user back to the view entry screen updated with the new changes.

4.0 Design

4.1 Use Case Diagram

The following is a use case diagram that shows the interaction of the user with the various use cases in the system as well as their relations to each other and elements within the system, namely the databases.

The user is directly linked with all use cases, all these use cases are performed/initiated by the user.

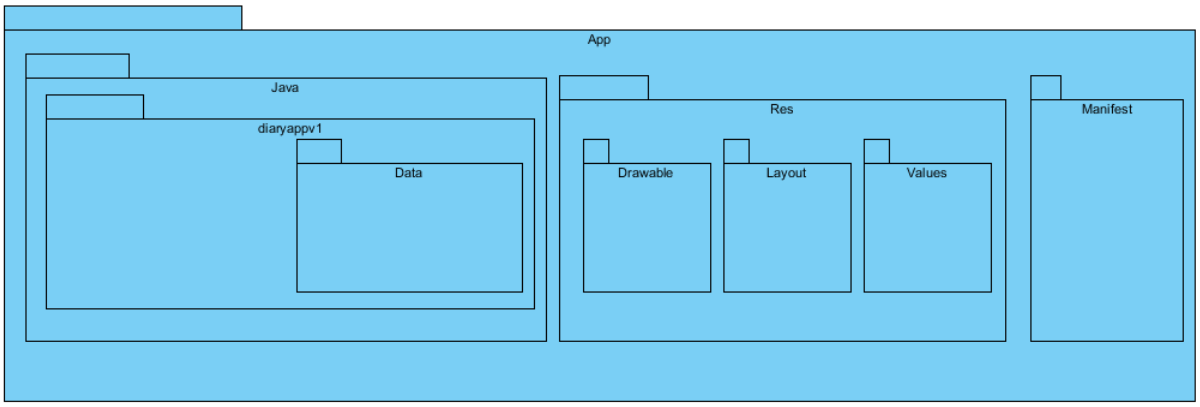
Viewing an emotions top 5 locations and viewing a locations emotion count is part of the greater "View Patterns and statistics" use case. "Add new location" likewise, is part of the greater use case "Add entry".

The application is highly linked to two core databases, one that handles user location tags and one that handles the storage of diary entries. Direction of the relationship shows which of these use cases receive data from a database, for example "View Diary Entry" and which use cases change the database's state, for example "Delete a saved entry".

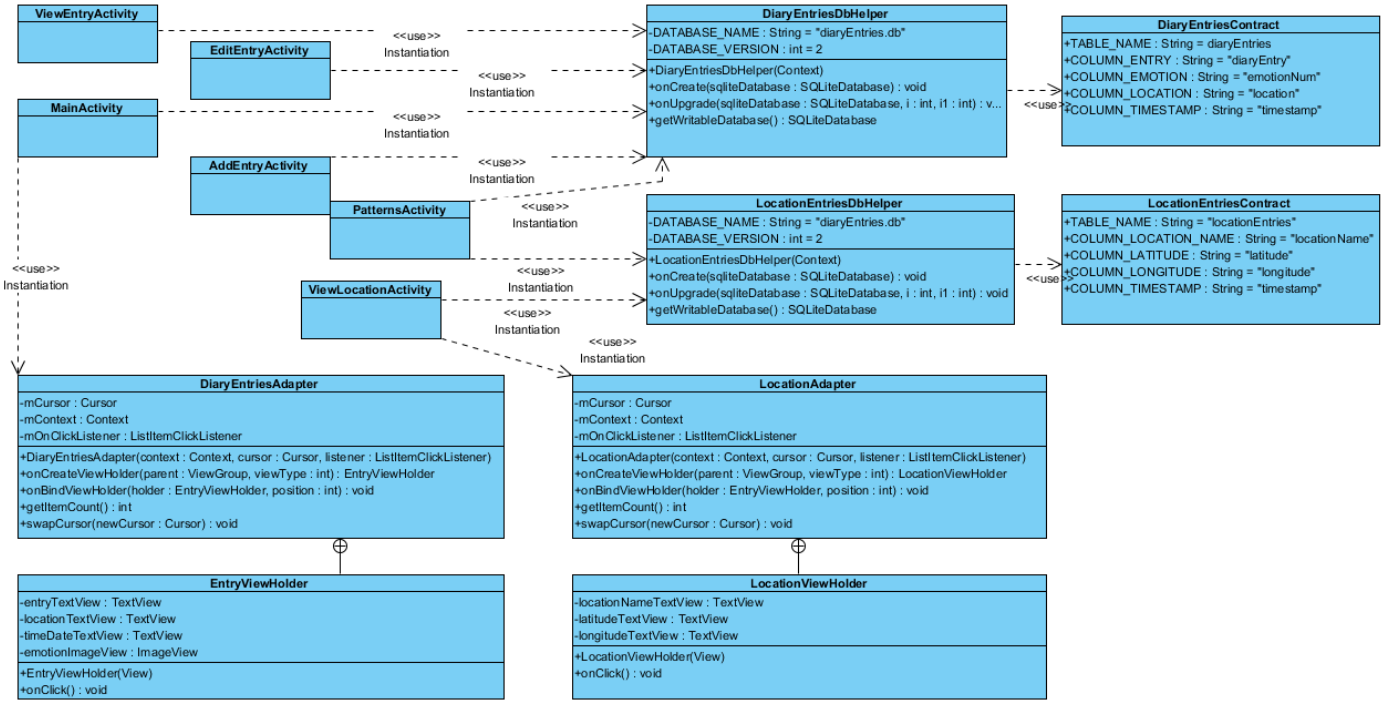
The use case "add a diary entry for a location in location history" and its supporting database locationHistory.db (coloured in red) are not present within the final application. However this was an important use case in the design of the application so I have added these to show what the system should have included.

4.2 Static Architecture

4.2.1 Package diagram



4.2.2 Class diagram



4.3 Entity relationship diagram

The following diagram shows the design of the database tables used in the system. It should be noted that the location history table was not implemented in the final application, but would have been used with the implementation of the “add a diary entry for a location in the location history” use case. It has been included in the diagram to demonstrate the structure it would have followed had this use case been implemented on time.

“locationEntries” stores the users location tags they have created. It is used in “add entry” to get all location tags, the locations within 50m of the users current location are used to populate a dropdown menu used by the user to choose a location tag. It is also used in by “patterns activity” to display location options to the user in a similar manner. This avoids the need to search through all diary entries every time we need a list of current location tags.

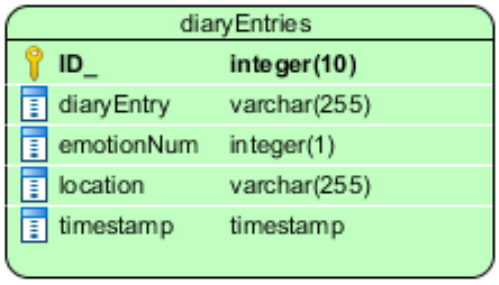
“diaryEntries” stores the data used to represent the user’s diary entry. It is used by “View diary entry” and “view patterns” which accesses entry data from this table. “add a diary entry”, and “delete a save entry” add and remove entries (rows) within diary entries tables and “Edit a saved entry” both receives data from the table and has the ability to change the “diaryEntry” column of an entry.

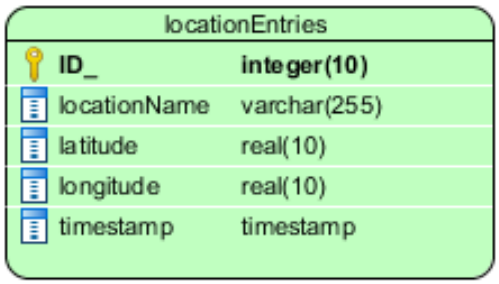
ID_ is never used in location entries by the application, a better primary key would have been locationName, this would eliminate an unnecessary column and would solve the following issue. Currently locationName does not have to be unique. This is a logical floor in the system and may allow the user to add the same location tag twice without realising. This is not a major issue but it is inefficient to have multiple copies of the same location in the database.

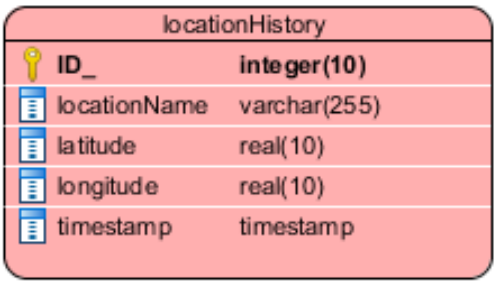
The applications methods currently ensure that any new location tags created for an entry are added to the locations database. Therefore, created entries cannot have a location that is not available from locations database. In future it would be better to ensure this with the use of foreign key constraints upon “location name” and “location” and adding the two tables to the same database to enforce this.

locationHistory would be used by the “add a diary entry for a location in location history” use case. Location history would deliver this activity the locational history of the user, for which entries can be added. “locationName” would be a location name found within the locationEntries table, or Null if this place had not been defined with a tag yet.

It should be noted that by default, column lengths are not defined in sqlite. The application needs to enforce these restrictions itself as unfortunately, the Visual Paradigm software used to create the diagram needs a length as input so I just used default values but they mean nothing. Another notable difference between sqlite and other relational database designs is it only offers 5 entry types. So in the below diagram, varchar actually refers to a type called TEXT. The last thing to note is the timestamp, timestamp is actually stored as TEXT, but can also be stored as a REAL or INTEGER type. Sqlite does not use static typing so there is nothing actually ensuring data integrity for type, this must be enforced by the application methods accessing and changing the data. The actual definition of timedate column is handled by the command DATETIME DEFAULT CURRENT_TIMESTAMP, this defines the data as type DATETIME, which as mentioned isn’t an actual sqlite type. sqlite when confronted with illegitimate types converts them to the closest possible type. When an entry is inserted the DEFAULT CURRENT_TIMESTAMP dictates that the default value for time date is taken from the devices current datetime.



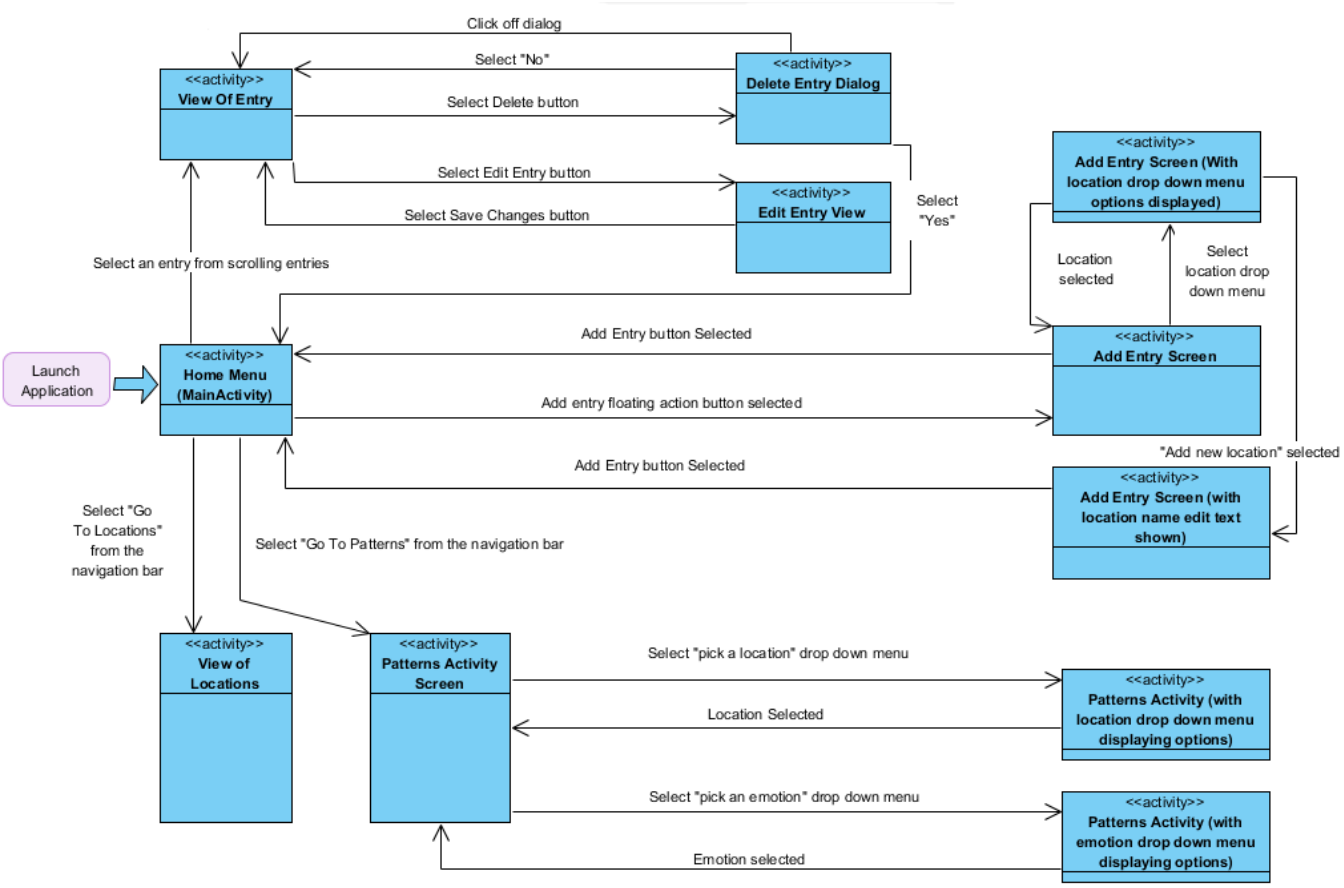




4.4 Dynamic Behaviour

State Diagram

Below is a state transition diagram that shows the flow of user actions within the application, it should be noted that the navigation bar is available in every state and therefore you can navigate to the Home Screen, Patterns Screen and locations Screen in these states. To keep the diagram clearer I have avoided labelling these actions accept for the Home Screen state, however these screens are available from all states except “delete entry dialog” and “edit entry” state.



4.5 Information architecture

The way the application displays data and information was an important consideration during the design phase. The information architecture effects the way users can view and interact with diary entries as well as other data related to the application. In Richard Saul Wurman's book, Information Anxiety, he mentions 5 main ways to organise data and information via the mnemonic LATCH (Location, Alphabetical, Time, Category and Hierarchy). With this in mind, the different types of user interactions with the data have been considered against each of these methods of organising data.

Diary entries have been listed by time and date on the home page. This is the best way to introduce some order when casually viewing entries without giving prejudice to any "type" of entry. For casual browsing and viewing of entries, it is a safe assumption that the user will be more interested in the most recent entries so they have been listed in reverse order. In the future, it would be helpful to allow the user to select a date from a "calendar selection" allowing the user to narrow the entries shown to those matching the date selected. This would be particularly beneficial to users that have used the app for a long time and have many entries.

It was hoped that the user would be able to narrow viewable entries by accessing a list of locational tags they have created, selecting one of these to return the diary entries tagged at this location. This would be helpful when the user wants to see entries for a specific location. It would be helpful to present the locations in alphabetical order to help the user find a predetermined location. Some

work was made towards this solution by giving users access to the location tags they had created, but full implementation was stopped due to time constraints on the project.

It would also be a good idea to give the user the ability to perform similar searches by emotional tag (categorical). However, an assumption has been made that a user will be far more interested in the places in their routines, rather than randomly looking for entries of a particular emotion. In the patterns activity, the user can see the top 5 locations at which they feel a selected emotion. This hierarchical ordering is helpful in showing the importance of particular locations and the entries associated with them. Users that want to learn about their emotional responses to particular locations can reflect on the most prominent locations that invoke an emotion and possibly discover why this is by viewing the relevant entries.

In regards to narrowing searches for entries that contain specific words or phrases, it would be beneficial in future to create a function to allow the user to search entries based on a string of text.

4.6 User Interface designs

The UI is designed to make the application as usable and as aesthetically pleasing as possible. Keeping the application tidy and ordered helps prevent any confusion of function. By sticking to the android design standards (Google material design) where possible and preserving continuity throughout the application, it helps keep the user familiar and reduces any learning curve associated with the applications features.

Using standard Android UI components such as spinners and buttons, especially in styles that have specific universal meaning helps give familiarity to the application, the specifics of which, will be covered in this section.

The UI is designed to limit the amount of clicks the user must do to complete a task, this is to help avoid the user feeling lost within the application. Where the user must click a number of times (3-4), the controls are as intuitive as possible using standard android design principles, this guides the user towards their goal more quickly and lowers any learning curve for the applications features.

The CARP (Contrast, Alignment, Repetition and Proximity) design principles have also been considered and implemented within the design.

Colour

Contrast in colour is particularly important within the application. It keeps the UI interesting and also highlights the importance of main features and separates functions of different roles. It also avoids hard to read “blending” of words/features and background/surroundings.

Google material design suggests limiting bold colour to 2 separate hues, a primary colour and one other “contrast” secondary colour. The primary colour consists of a set of 3 different tones to allow differentiation between elements and the secondary colour is used sparingly for highlighting important objects or text. The colours in the initial wireframe mock ups were a first draft and used for getting a rough feel and look. They later changed but followed the same rules. The primary colour consists of 100; 500; and 800 (referring to the “tone”); and a secondary colour was used for highlighting. White is also used to help elements such as text areas stand out and remain clear.

To avoid unnecessary confusion for colour blind users, I have stayed away from a Red-Green combination and the rarer Blue-yellow combination to ensure the application is accessible to all potential users. The applications also uses the primary colour carefully to avoid confusion for older

uses or those users who struggle to differentiate shade and may struggle to read text on a similar background, in most cases a white background for text is used to help avoid this problem.

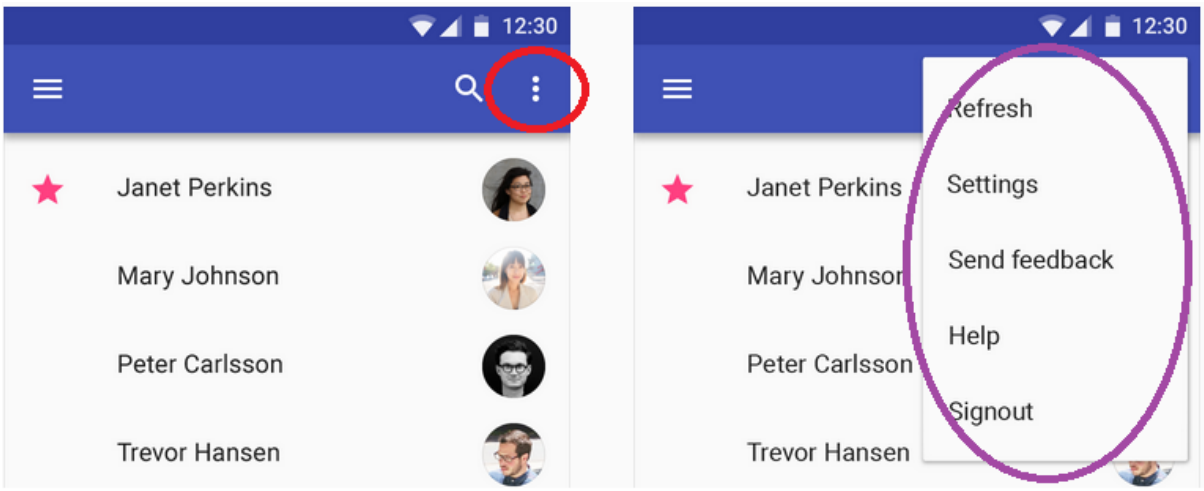
Navigation Bar

The navigation bar was designed to provide easier navigation around the applications main functions and also helps to highlight these main features/activities within the application. The navigation bar offers access to the Home screen (Main Activity), the patterns screen and the locations screen, the current activity is highlighted from other choices to allow the user to easily know exactly where they are in the application.

In the initial interface designs this was only available in some activities. This has now been updated to be in all activities (accept edit entry) to help user navigation across the entire application and to ensure consistency.

Menu

In the top right corner of each interface wireframe design there is an area reserved for what was initially planned to be a “hamburger” style menu. Below is an example of this style of menu taken from the material design website.



https://material.io/guidelines/components/menus.html#menus-usage

This would have contained items such as settings allowing the user to change themes of the application and entries, as well as provide quick access to other extras. Unfortunately due to time constraints these features were never implemented so this menu has been left out of the final application with hope that it could be completed in future.

Main Activity (Home screen)

Allows the user to scroll through their entries or access other areas of the application through the navigation bar.

Floating Action Button (FAB) – A floating action button has been used as an "add entry" button to add new entries, FAB's are used to draw focus to the application most important features. The FAB has also been coloured with the "accent" colour to help emphasise its importance within the activity as the main function. The FAB uses a universally standard symbol for add (+), in fact the + sign used is taken from the android design library and is commonly used and recommended in the material design guides. This helps avoid needing to put the text "add entry" into the button saving space and making the application look less cumbersome. With the context of the application, users should be able to easily identify what the add button does, it therefore helps keep a sleeker design without confusing the user about its function.

Later in the implementation phase it was decided to move the FAB to the bottom right of the screen (as can been seen in the final application screen shot). This makes the button easier to reach with your thumb when using a device such as a phone, it also looked visually better with the changes made to the diary entry views (mentioned below) and obscures less important information in this position.

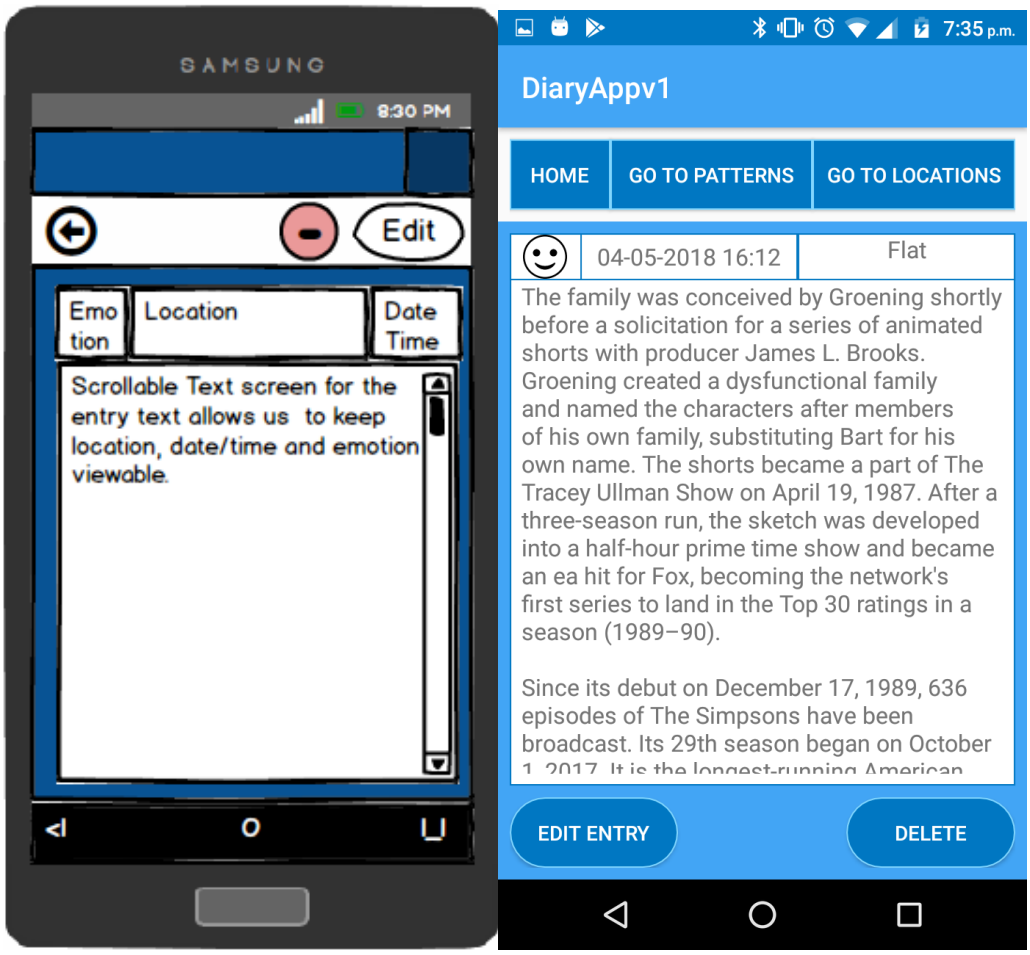
Scrolling View – The diary entries are displayed to the user in a scrolling view, the size, alignment and grouping of each entry is mirrored for each entry. This continuity highlights that these are separate but equal items within the display. A scroll view of all entries makes it easy for users to casually browse through their entries.

The layout of the views used to hold entries changed slightly in implementation. Emotion, time/date and location are now positioned horizontally above the entry text, this still preserves the design features, however it makes the data easier to display. The delete buttons made the scroll view quite cluttered so were removed to avoid this, users can still delete entries by swiping them (after which a

dialog appears to confirm deletion). Although this isn't an obvious feature, it is there for users who become "experts" as a way to quickly delete entries, other users can delete entries in the View Entry Activity below.

View Entry Activity

This activity allows the user to view their entries in detail, it also offers the ability to delete and edit the text of entries.



The contents of the diary entry are grouped together in close proximity to show they make up the entry as a whole, however the emotion date/time and location attributes have been aligned to group them further. This gives prominence to the main focus of the entry, the text.

Scroll view - used to surround this entry's text content. This ensures that the user can scroll through larger entries and still always be able to see the navigation bar (added in implementation) and other functions such as edit and delete as well as always being able to see the entries attributes (emotion, time/date and location).

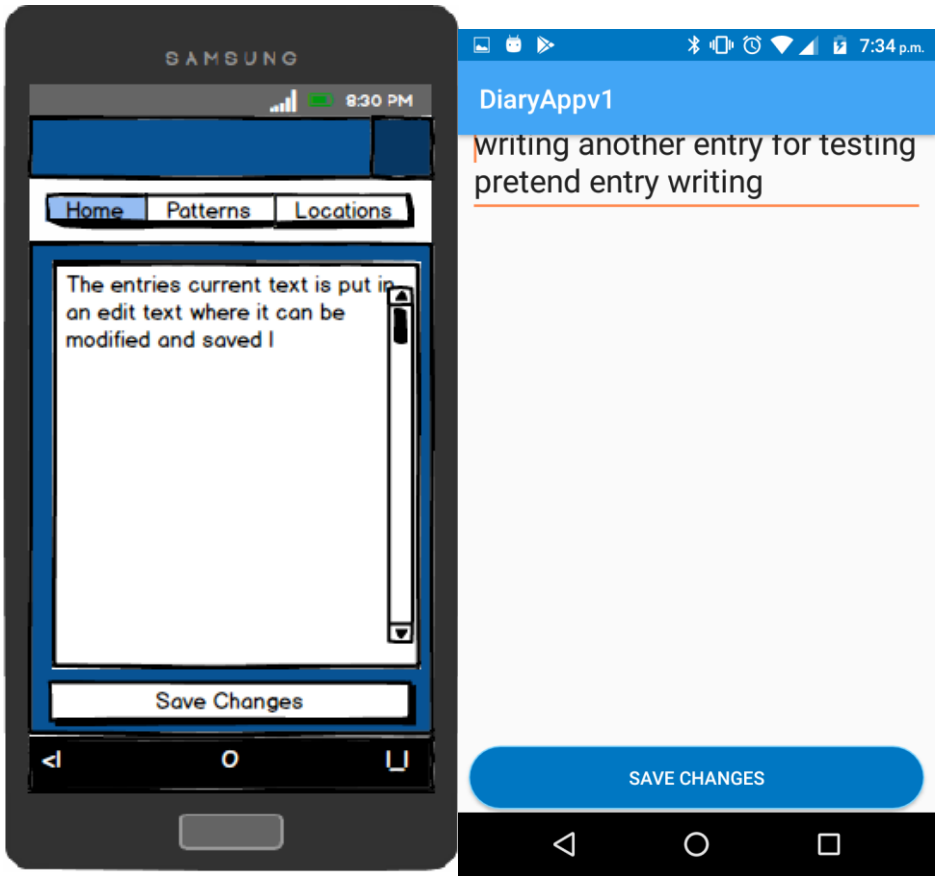
The **delete button** was initially designed to mirror the add button with a minus (-) sign. However, during feedback while testing using design wireframe mock-ups, it was noted that this wasn't entirely clear. One participant thought that you may need to select **edit entry** to delete the entry. As a result it was decided to make this button clearer by having a text displaying "Delete". To highlight these new features to the user they were also made larger. After adding the navigation bar to the design in the implementation phase, the top of the screen looked too cluttered. The edit entry and delete buttons were moved to the bottom of the screen, mainly for aesthetic reasons and to avoid

blending of the buttons but also make the buttons easier to reach with the users thumb for quicker navigation.

Initially the designs had a **back button** specifically built for the application however, Google recommends that navigating back to a previous activity be handled in a more intuitive way. All android versions currently have a back button built into the platform interface. When questioned about how to navigate back to the menu users, seemed to instinctively know they could use this inbuilt function rendering the back button unnecessary. As a result, this was removed.

<u>Edit Entry</u>

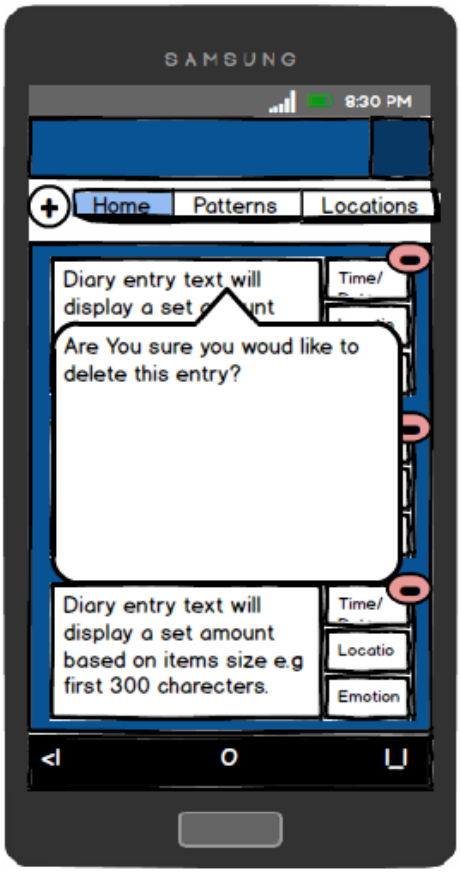
Edit Entry button allows the user to edit their entry's text, an assumption has been made that users may want to edit the text of their entries if they make spelling/grammar mistakes or they think of things to add after completing the entry. However it is logically unlikely that a user would ever need to change the time/date or location of the entry or that they would change their emotional response to an entry.

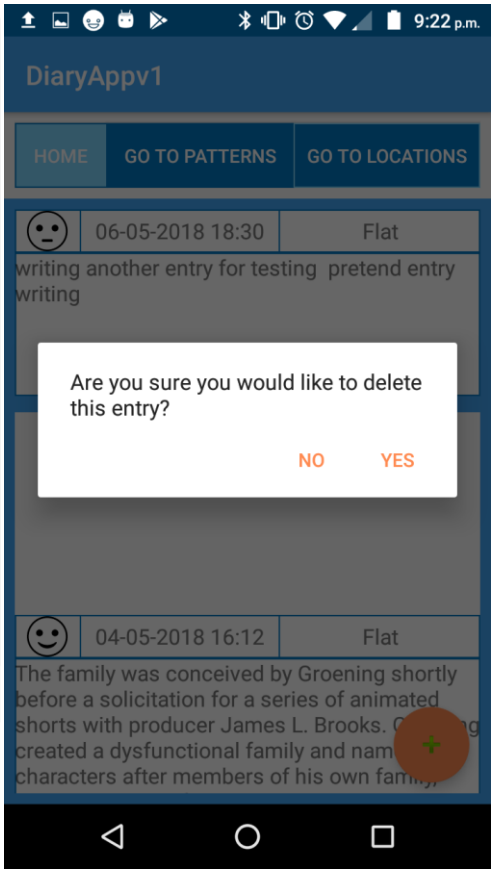


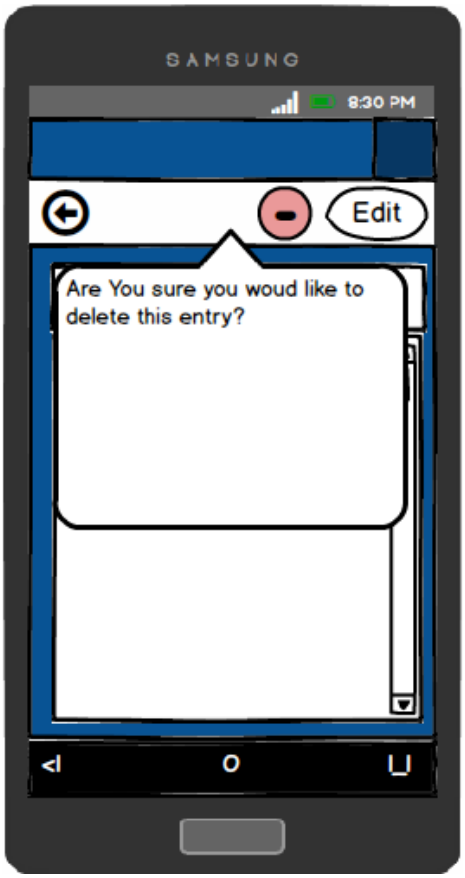
A simple layout avoids any confusion in function, user can edit the entry's text and then choose "Save Changes" to finish. The navigation bar was removed in the final application. It is unlikely that user will go to edit an entry and then want to navigate elsewhere, if they made a mistake they can just use the back button. The navigation bar made the UI more complicated then it needed to be in this activity and the approach to this activity was to make it as minimalistic as possible to focus on its one and only function.

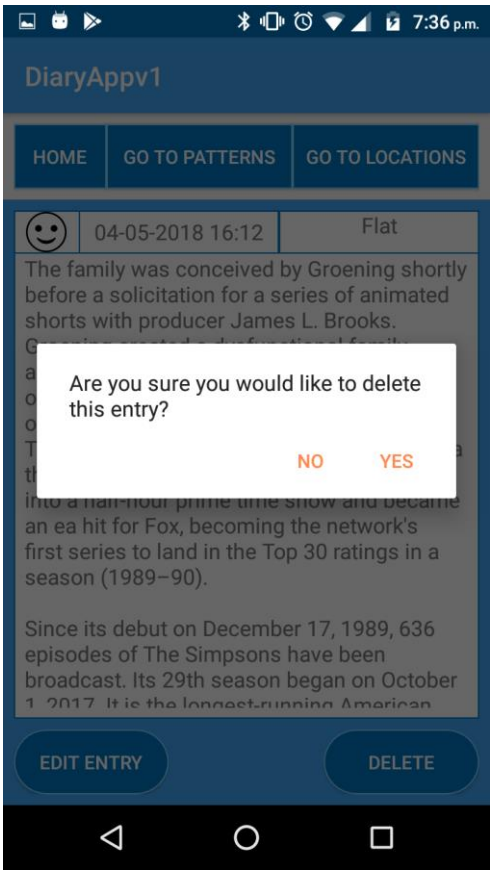
Dialog confirmations for Delete

Dialogs were used to make sure the user confirms the deletion of any data. This avoids accidental deletion of data but and also ensures the user knows what they are doing within the activity. The nature of dialogs makes them hard to miss when they pop up. They are large and intrusive enough to demand the users attention. Also, the user can't confirm a positive action if they accidentally click off the dialog so there is no chance of accidental deletion. Dialogs conform to material design, ensure error correction and are a standard part of the android packages. Below you will see the initial design and the implementation for deletion in both the home screen and the view entry screen.



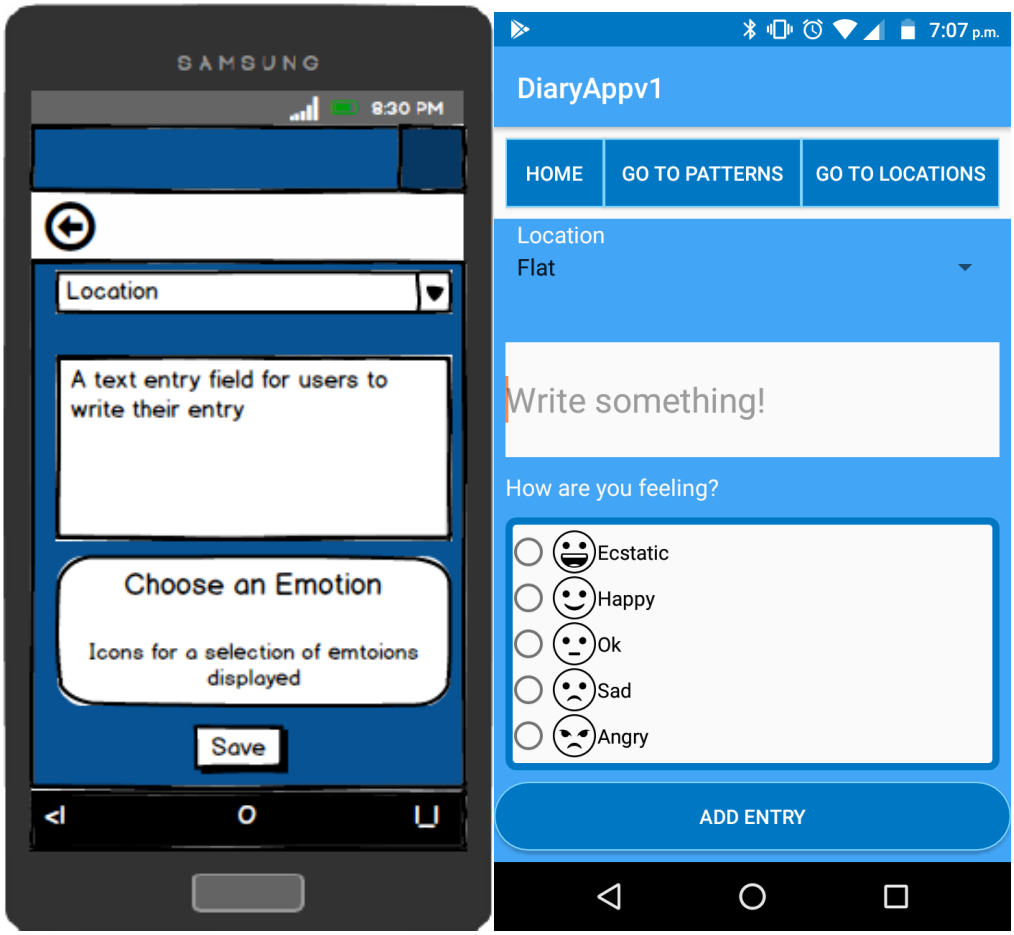






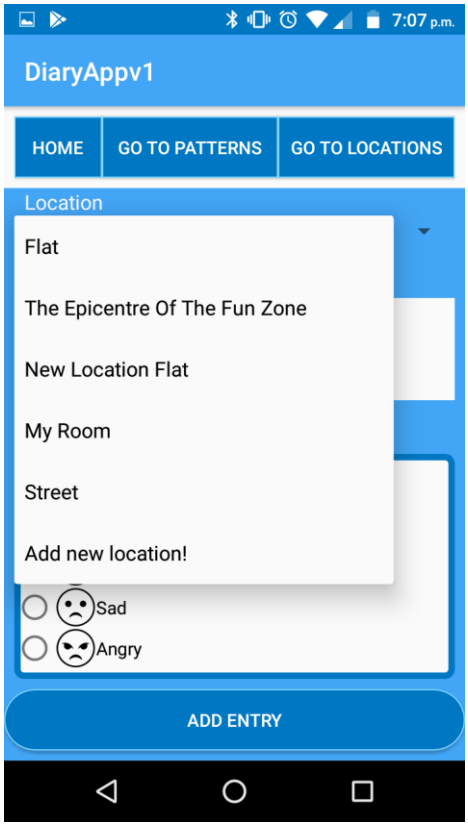
Add Entry Activity

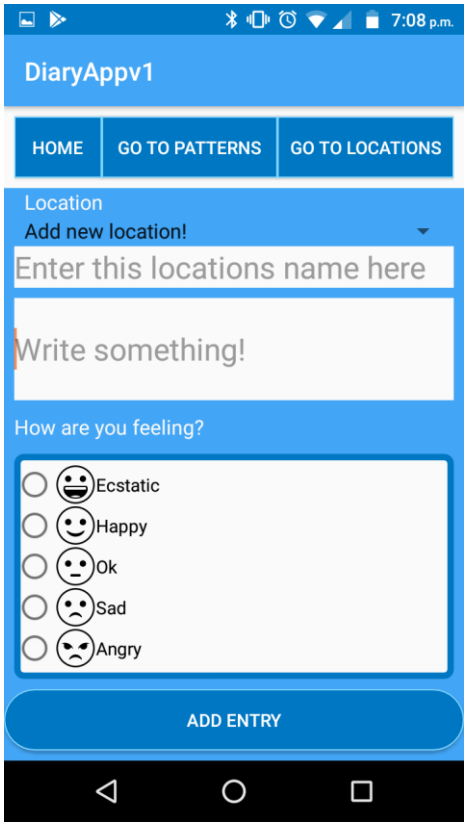
This activity allows the user to add an entry to their diary, the user's current location is determined and then this is compared to any previous location tags the user has created to see which ones are within proximity of the current location. If such tags exist they are displayed to the user as options to select in a drop down menu, if there are no previous locations close by then the user can create a new location that uses the user's current latitude and longitude to define the location. The user can always add a new location even if suggestions are presented by choosing the "Add new location" option from the drop down menu, this then reveals a hidden text field in which to type the location name. Users then write an entry and choose an emotion to tag the entry with.



Above you can see the initial wireframe design and its implementation in the application.

Below left you can see the use of the drop down menu for locations and below right you can see the "Add new location!" option reveals a hidden edit text, hiding this area unless being used helps keep the interface easier, simpler and generally better aesthetically.





Limiting the use of text fields for user input and making use of different input element types helps keep separation between different functions of user input as well as make the interaction a little less repetitive.

Spinner - User location tags are offered within a drop down menu to help save space and group together the functionality.

The main input is the **diary's entry text**, this is the only text field initially present apart from in specific situations.

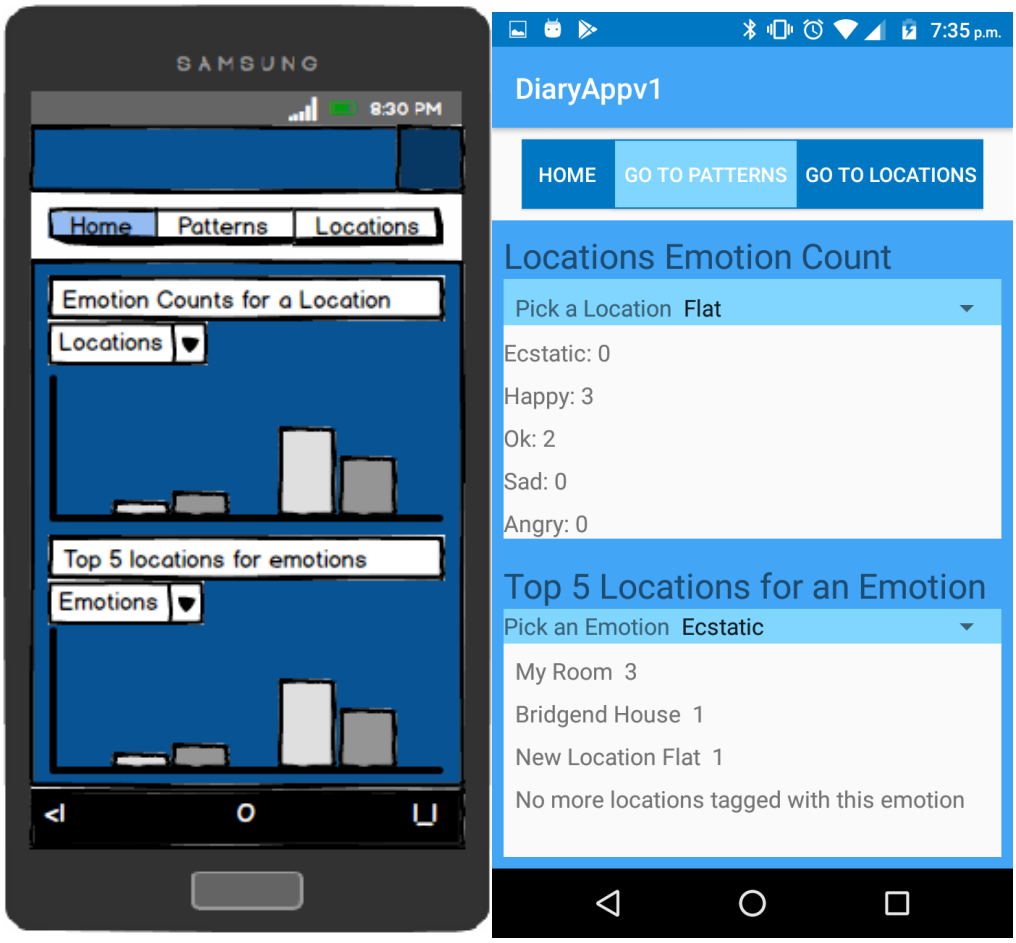
Radio Button - The emotion tag selection is done via radio button selection, this groups the elements together and avoids any ambiguity between input types. Radio buttons allow the application to ensure entry is correctly formatted and limited to a number of options of which only one can be selected, this avoids affecting the processing of posts when producing analysis and patterns.

Alignment of the various inputs that make up an entry shows unity and keeps things tidy.

Save button - It is a material design suggestion to save user input automatically (As a draft or as incomplete final) in case the back button is pressed or the user needs to close the app unexpectedly. I believe it will be unhelpful to the user to automatically save unconfirmed entries, if a user decides to come back later they can always save and then edit the entry later. The user may change their mind about making an entry mid-way through, automatically saving this would mean automatically creating the entry which may spam the user's diary with unfinished unwanted entries.

Patterns Activity

The patterns activity displays statistics to the user, the user can edit different inputs such as location and emotion to refine what they learn.



Each set of data and the buttons or spinners used to refine it are grouped together in close proximity to show their relation. This also helps separate the different patterns and statistics apart.

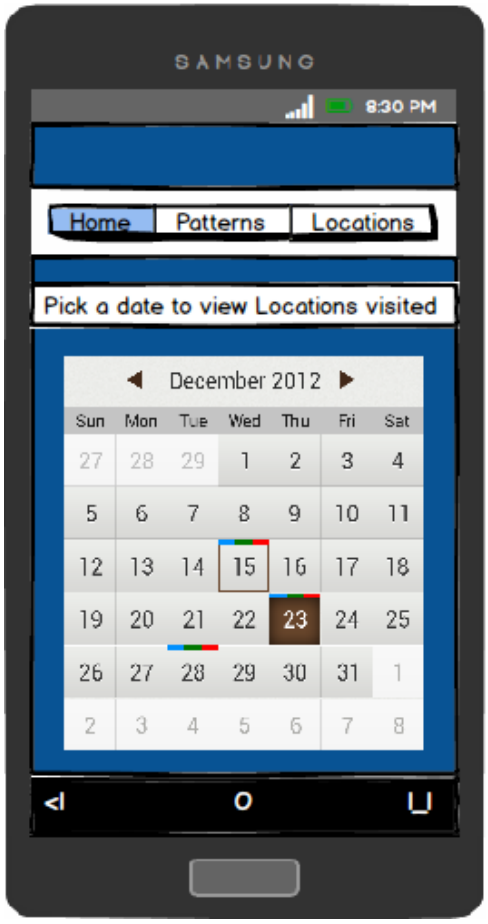
Unfortunately, in the final application the bar/pie charts planned were not implemented, they have been replaced with text displays due to time constraints on the project. The application elegantly handles events where data is missing from the application. For example, above right you can see that only 3 locations (5 entries) have been tagged with the “Ecstatic” emotion tag thus far, the application displays an explanation to the user in order to avoid confusion. In the case where no entries had been tagged with the emotion another text would come up saying no entries had used this tag yet.

Locations Activity

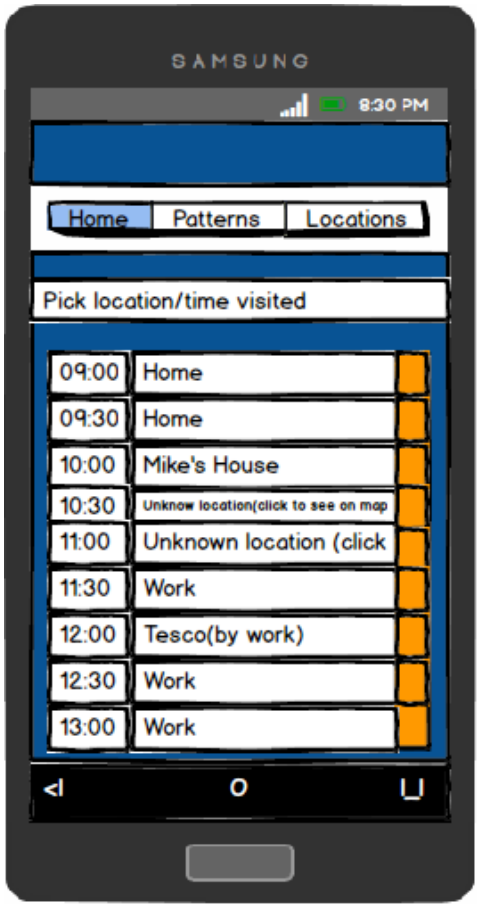
The Locations activity was originally designed to be a way to create entries at a later point by choosing a date/time and the location recorded at that time, then they would be able to add an entry with these values automatically filled. Background collection of the user’s location at set intervals of time would be used to do this. On the first page a user picks a date, then picks a location/ time on the second page from the background data collected for that date, the user can

then see this location on a map for reference and then choose to add an entry for this location or just add a new location tag for this location.

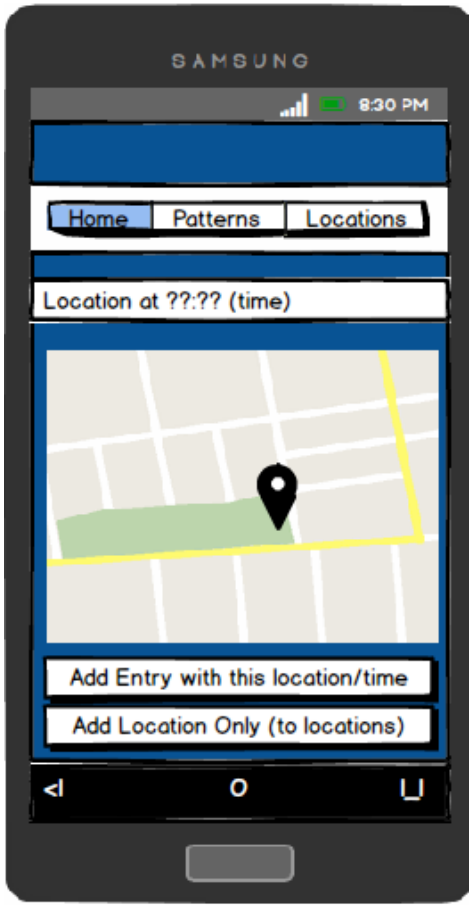
This feature was not implemented due to time constraints on the project, as a result the activity just displays the list of locational tags the user has created while creating entries. This feature would be the primary use case to develop in the future. The following images show the design envisioned originally.



The User can only use the navigation bar or select a date (other than press back or close the application), the process was created to follow a wizard/tunnel like design frame to guide the user through the process.



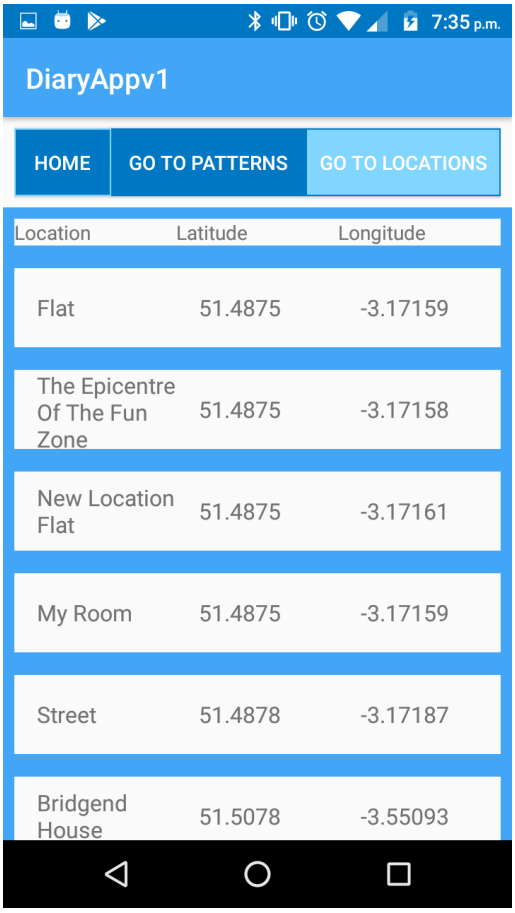
The next part of this directed interface involves picking a location/time from the options recorded for the selected date. **Alignment** of time with the location vertically suggests these elements are together due to the table like manner, alignment of all the locations shows these all provide similar information. Locations may be unknown if a location for that position has not been defined before, in the next step the users position is shown on the map to help the user create a location tag.



The location is presented to the user on a map in case the location is unknown to the application, in this case the user can create a new tag for the location and added to the database, either while adding an entry or using the add location only function. This page will take the user to the add entry activity with the relevant information already filled.

The location, time and date are shown above the application to avoid the user having to remember their previous selections.

The Image below shows the current locations activity, this just displays a list of location tags the user has created along with their coordinate positions. The layout uses alignment and proximity to group the data for each location and separate individual locations. The list is scrollable but the label bar at the top that denotes the columns stays in place at all times.


7:35 p.m.
DiaryAppv1
HOME
GO TO PATTERNS
GO TO LOCATIONS
Location
Latitude
Longitude
Flat
51.4875
-3.17159
The Epicentre Of The Fun Zone
51.4875
-3.17158
New Location Flat
51.4875
-3.17161
My Room
51.4875
-3.17159
Street
51.4878
-3.17187
Bridgend House
51.5078
-3.55093

5.0 Implementation

The following section details the specific implementation of the main features and parts of the application. The structuring of this section separates the functionality into a logical flow. First, the main components are explained including emotion tags, handling location and the database structure. Then, the functions will be discussed in an order that will take the reader through the application in a task focussed manner.

5.1 Emotions

When implementing the emotional tags in the application, the aim was to keep the interface simple and leave any complexity to the methods that process the data while making the information produced as beneficial as possible.

Designs have been kept simple for the user. users like a visual representation wherever possible to make the interface more aesthetically pleasing. Smileys are a popular form of visually representing emotion, they are intuitive and familiar to the majority of android users. To avoid confusion between meaning, the textual meaning of each smiley is also shown when choosing an emotion while creating an entry. The images chosen for these smileys are a set encouraged by the google design guides, a reference to where they were created can be found below. The designs are open source and it is only asked that an acknowledgement be given to the original source, although not legally necessary they are listed for transparency.

By Emoji One, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=59731229 – ecstatic

By Emoji One, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=59733597 – happy

By Emoji One, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=59732564 – ok

By Emoji One, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=59732415 – sad

By Emoji One, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=59733705 – angry

In order to reduce the storage demands of collected user entries and to make emotion tags easier to work with when searching or analysing data patterns, a reference to the emotion tag chosen is stored as an integer value. Each of the 5 emotions is referenced by a unique integer between 1 and 5. Integers are far smaller to store in a database then the image of the smiley associated with that emotion, when carrying out tasks, it is computationally inexpensive to do operations on integers, as opposed to comparing strings or dealing with image files. As a result, the operations under emotion data within the application run relatively quickly. When displaying the emotions in entries to the user a simple switch statement is used to define which image is necessary to substitute in its place. This means only 5 images need to be stored in the application resource folder, rather than 100's of the same 5 images in a database table.

```
switch(emotion) {
    case 1:
        holder.emotionImageView.setImageResource(R.drawable.ic_ecstatic);
        break;
    case 2:
        holder.emotionImageView.setImageResource(R.drawable.ic_happy);
        break;
    case 3:
        holder.emotionImageView.setImageResource(R.drawable.ic_ok);
        break;
    case 4:
        holder.emotionImageView.setImageResource(R.drawable.ic_sad);
        break;
    case 5:
        holder.emotionImageView.setImageResource(R.drawable.ic_angry);
        break;
}
```

5.2 Location

The application makes use of googles Location Service API to handle current location and also manipulate locational data objects. The Location services are used in the Add Entry activity, any use of the locational information thereafter is obtained from the relevant database table. First, the application must be granted permission by the user to use the devices location. This is achieved with a declaration in the applications manifest shown in the following highlighted line of code.

AndroidManifest.xml

manifest

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="greenslade.nathan.diaryappv1">
4
5      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
6
7      <application
8          android:allowBackup="true"
9          android:icon="@mipmap/ic_launcher"
```

The primary use of the location services is during the applications add entry activity. When this activity starts up, a location request is created. The LocationRequest is a data object that contains quality of service parameters, in this case setting the accuracy to high and the interval to 1000ms, for request to the FusedLocationProvider. The FusedLocationProviderClient is the class that interacts with the FusedLocationProvider.

AddEntryActivity.java

AddEntryActivity

```
89      //set up locaton requests
90      mLocationRequest = LocationRequest.create();
91      mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
92      mLocationRequest.setInterval(1000);
93      mFusedLocationProviderClient = LocationServices.getFusedLocationProviderClient( activity: this);
```

We can use the FusedLocationProviderClient to get the last known location but what we are really interested in is getting location updates. To receive location updates the application uses a

LocationCallback to handle responses from the FusedLocationProvider, by implementing the onLocationResult method the LocationCallback handles location results received. The location object has a number of data, however for the purposes of this application, only the current latitude and longitude are needed to determine the users current location. For the purpose of the activity, there is only need to get one location update, once the application has the latitude and longitude of the current location the method stops further unnecessary location updates.

```
//set up location callback to update location
mLocationCallback = (LocationCallback) onLocationResult(locationResult) → {
        super.onLocationResult(locationResult);
        for (Location location : locationResult.getLocations()) {
            //update latitude/longitude
            if (location != null) {
                latitude = location.getLatitude();
                longitude = location.getLongitude();
                stopLocationUpdates();
            }
        }
```

We start receiving location updates by calling startLocationUpdates() method within the onResume() method of the activity. This calls the requestLocationUpdates method on our FusedLocationProviderClient. Calling this in onResume ensures that the location request is made when the activity begins, it also ensures that if the user leaves the application, the location is checked and updated in case the user has moved in the time it has taken to come back to the activity.

```
private void startLocationUpdates() {
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSIO
        mFusedLocationProviderClient.requestLocationUpdates(mLocationRequest, mLocationCallback,  looper: null);
    } else {
        //request permission
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, MY_PERMISSION_REQUEST_FINE_LOCATION);
        }
    }
}
```

This is how the applications determines the current latitude and longitude, but the LocationCallback also handles some other work for the add entry activity, including updating some of the user interface elements. The following code is also held within the onLocationResult method.

```
adapter = new  ArrayAdapter<String>( context: AddEntryActivity.this, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
for (int i = 0; i < mCursor.getCount(); i++) {
    float[] results = new float[1];
    mCursor.moveToPosition(i);
    double eLatitude = Double.parseDouble(mCursor.getString(mCursor.getColumnIndex(LocationEntriesContract.LocationEntry.COLUMN_LATITUDE)));
    double eLongitude = Double.parseDouble(mCursor.getString(mCursor.getColumnIndex(LocationEntriesContract.LocationEntry.COLUMN_LONGITUDE)));
    Location.distanceBetween(latitude, longitude, eLatitude, eLongitude, results);
    double result = results[0];
    if (result < MIN_DISTANCE_FROM_OTHER_NAMED_LOATIONS){
        adapter.add(mCursor.getString(mCursor.getColumnIndex(LocationEntriesContract.LocationEntry.COLUMN_LOCATION_NAME)));
    }
}
adapter.add("Add new location!");
mSpinner.setAdapter(adapter);
```

When the user starts the add entry activity, the activity finds the current location, the onLocationResult method then compares this location's latitude and longitude to the locations found within the locations table that holds all the previous locational tags the user has created.

Cursors hold the result set retuned by a database query. mCursor is a set of the user's previously defined locations, the method loops through each previous location and checks to see which locations are within a 50m radius of the current location using the distanceBetween method of the Location class. The results of this are then added to an adapter which displays these results in the UI as a spinner. If the user creates a new location tag, either because there are no previous locations within 50m or they choose to define a new location at this exact point, the current latitude and longitude variables used to hold the location results data are used to define latitude and longitude.

There are 2 reasons why the application checks for locations within 50m rather than a more exact match:

1. Some mobile devices have a less accurate sensor for determining location, in order for the application to effectively suggest a previous location, it needs to have a bit of leeway with the accuracy.
2. Some users may wish a wider area to be under one tag, for example a work place may be a big office building, in which case the user's location at "work" may vary quite significantly but they still may want to use the same locational tag. By suggesting all tags within a proximity and allowing the user to choose which to use it gives them flexibility, they may define a location to be over a larger area but still have the flexibility to have several locations close together. For example each room of a house, and have the option to select which one they choose.

A future improvement could be to allow the user to define this radius in a settings menu, this would allow for appropriate customisation if the user wishes their tags to suggest what district, town, city or even country they are in instead of a more local scale.

5.3 Database

Two sqlite database tables are used to store different related data, accessed by various methods within the application, these are "diaryEntries" and "locationEntries".

diaryEntries stores the data needed to create and rebuild a diary entry within the application. The tables columns consists of columns "ID_", "diaryEntry", "emotionNum", "location" and "timestamp". ID_ is an integer value that acts as a primary key and auto increments on creation of a new entry, this ID is used to refer to and manipulate specific rows (entries) within the table. The diaryEntry row is a string that holds the users actual written entry. emotionNum is an integer value that represents an emotion, storing an integer representation saves the space needed to store the emotion for a given entry. Location is just a string that refers to the user defied location tag of the entry, this is just a reference to the name of the location tag used as location tag data is held within a separate table. Timestamp just refers to data using the java.sql.Timestamp class so we can keep track of the time and dates of the entries automatically as they are added.

Storing the entry as separate types of data rather than a "diary entry" object gives more flexibility in the way the data can be queried. One benefit of this is the application can query specific columns of an entry rather than retrieve a whole object, which means working with smaller data types when you only need a specific part of the entry such as emotion. It also means the application can make use of sql commands to narrow results returned for a given task, so when we need entire entry information, we can return them based on an attribute constraint rather then pull all entries and then process the information within the current activity that needs it.

locationEntries stores the data for user defined location tags, it consists of the columns “ID_”, “locationName”, “latitude”, “longitude” and “timestamp”. ID_ is again used as a primary key and timestamp refers to the time of creation of the location tag. locationName is a string that represents the user defined name for a particular location. Latitude and longitude columns are both double values that are taken from the location objects manipulated within the application, effectively they are the way we define where locations are in relation to each other.

The locationEntries table Is primarily used when the user enters the add entry activity, the activity requests the users current location and this is compared to the values within the database table to establish if this location is (or at least might be) a previously visited location the user has already made an entry for. Housing this data in a separate table voids the necessity to search through all previous diary entries in search of a match to the current location. This is more efficient as the table will contain less rows then that of the diaryEntries.

The location held within the diaryEntries table is just a string that refers to the user generated name they have given the location (identified by the longitude and latitude in the locationEntries table). For the purpose of displaying an entry and its location, this is all the information needed. The entry doesn’t have to be “aware” of the actual position. However, by storing the same string name as the locations name in the table containing location tags we can use this name as a key if we needed to join the two tables if the entry needed to reference its actual physical location. This is helpful for future development of functionality for example such as showing on a map where entries were written.

DiaryEntriesDbHelper and LocationEntriesDbHelper extends the SQLiteOpenHelper class. This class is used as a helper class for creation of databases and version management. The classes are responsible for opening the database and handle creation of the database if it does not exist yet (for example, if the user has not opened the application before). It also deals with transactions to update the database when entries (DiaryEntriesDbHelper) or location tags (LocationEntriesDbHelper) are added by the application. Several activities within the application use these classes, having a separate class to contain them helps abstract the details of database creation and helps avoid replication of the same code across multiple activities.

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {

    // Create a table to hold diary entries
    final String SQL_CREATE_DIARYENTRY_TABLE = "CREATE TABLE " + DiaryEntry.TABLE_NAME + " (" +
            DiaryEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
            DiaryEntry.COLUMN_ENTRY + " TEXT NOT NULL, " +
            DiaryEntry.COLUMN_EMOTION + " INTEGER NOT NULL, " +
            DiaryEntry.COLUMN_LOCATION + " TEXT NOT NULL, " +
            DiaryEntry.COLUMN_TIMESTAMP + " TIMESTAMP DEFAULT CURRENT_TIMESTAMP" +
            "); ";

    sqLiteDatabase.execSQL(SQL_CREATE_DIARYENTRY_TABLE);
}
```

```java
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {

    // Create a table to hold location tags
    final String SQL_CREATE_LOCATIONENTRY_TABLE = "CREATE TABLE " + LocationEntry.TABLE_NAME + " (" +
            LocationEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
            LocationEntry.COLUMN_LOCATION_NAME + " TEXT NOT NULL, " +
            LocationEntry.COLUMN_LATITUDE + " REAL NOT NULL, " +
            LocationEntry.COLUMN_LONGITUDE + " REAL NOT NULL, " +
            LocationEntry.COLUMN_TIMESTAMP + " TIMESTAMP DEFAULT CURRENT_TIMESTAMP" +
            "); ";

    sqLiteDatabase.execSQL(SQL_CREATE_LOCATIONENTRY_TABLE);
}
```

The SQLiteOpenHelper class requires the new classes to override the onCreate and onUpgrade methods. onCreate handles creating the database itself

onUpgrade handles upgrading the database when the schema changes. This requires keeping a reference to the database version via a final integer value, which must be incremented if changes are made to the schema. In this case, when the schema is updated the table is dropped and a new one created. This loses any data held in the old table but for the purposes of this project this is acceptable. When early changes to the schema were made, it was not a problem to create new entry data to test the system with. This schema should now remain as it is for the duration of the applications life so there is no need to worry about this disadvantage with onUpgrade. Users of the application will not have to worry about the table structure changing so this will never directly affect them.

If certain features were to be added to the application in future, there is a slight possibility that the tables may need an updated schema. For example, if more information was added to the diary entry attributes, such as weather or activity tags, then it is reasonable to assume the diary entries table would change. If this were a working application where updates were expected, this would be a concern. However, if released, the onUpgrade method could be modified to try and retain previous entries.

Having these helper classes means we only need to change the code within this class to change the database structure of all activities that use this database. This makes the application far more easily extensible than if the database creation code was hidden within each activity respectively, requiring us to change every activities implementation.

5.4 User Interface Features Available Across the Application

The following sub section details features used in multiple parts of the application, these features are important in the creation of interface elements across the application.

Navigation Bar

A custom built navigation bar has been built to provide easy movement between the applications features, it allows functionality similar to tabs. The bar shows the user the main areas of the application giving the user insight into the tasks they can perform as well as its primary use for easier navigation within the application.

The bar uses the primary colour tones, the lighter tone is used to highlight the current screen the user is on (if any), the bar uses 3 different xml drawable files to create the layout.

menubarbackground.xml | menubarbackgroundalldark.xml | menubarbackgroundhighlight.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle" >
    <solid android:color="@color/colorPrimaryDark" />
    <stroke android:width="1dip" android:color="@color/colorPrimaryLight"/>
</shape>
```

menubarbackground.xml | menubarbackgroundalldark.xml | menubarbackgroundhighlight.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle" >
    <solid android:color="@color/colorPrimaryDark" />
    <stroke android:width="1dip" android:color="@color/colorPrimaryDark"/>
</shape>
```

menubarbackground.xml | menubarbackgroundalldark.xml | menubarbackgroundhighlight.xml

shape | stroke

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle" >
    <solid android:color="@color/colorPrimaryLight" />
    <stroke android:width="1dip" android:color="@color/colorPrimaryDark"/>
</shape>
```

The Navigation bar has slightly different settings in most activity's but is always made up of a linear layout (horizontal alignment) containing 3 buttons all styled individually with a parent styling for the linear layout.

Intents

Intents are used across the application to transition from one activity to another. The navigation bar for example uses intents to switch to the next activity (when triggered by pressing on a particular button/part of the bar). The intent below is an example of a method called by the navigation bar and activity's triggering another. This function is called by the appropriate button when pressed.

```java
//go to activity for viewing patterns
public void goToPatterns(View view) {
    Context context = MainActivity.this;
    Class destinationActivity = PatternsActivity.class;
    Intent startViewPatterns = new Intent(context, destinationActivity);
    startActivity(startViewPatterns);
}
```

Intents can also use the putExtra method to send a message to the proceeding activity. This is used in the view entry activity to communicate to the edit entry activity which entry is to be edited.

```java
public void goToEdit(View view) {
    Context context = ViewEntryActivity.this;
    Class destinationActivity = EditEntryActivity.class;
    Intent startEditEntry = new Intent(context, destinationActivity);
    String msg = String.valueOf(message);
    startEditEntry.putExtra(EXTRA_MESSAGE, msg);
    startActivity(startEditEntry);
}
```

Edit activity then uses getIntent to get the intent that created it and uses getStringExtra to get the message sent with the intent. In case of multiple messages, the intents putExtra method takes a variable name for the message, along with the submitted data. This variable name is a static final string to avoid typing errors.

```
Intent intent = getIntent();
int message = Integer.parseInt(intent.getStringExtra(MainActivity.EXTRA_MESSAGE));
```

The integer message is then used to tell the Cursor the position it should be at to retrieve the right entry. (Cursors are described next in this sub section).

Cursor

The application uses three different cursors to provide read access to the data sets returned by database queries, this is used by the application to set the data when displaying diary entries and location tag information. A cursor is also passed to an adapter when using the recycler view to set the data for each view.

getAllLocations returns a cursor used by AddEntryActivity and PatternsActivity. This returns all location tags in the locations table. getAllEntries returns a cursor used by MainActivity, ViewEntryActivity and EditEntryActivity returning all diary entries in the diary entries table. Returning the full table was a simpler query to write however ViewEntryActivity and EditEntryActivity only need access to one specific entry. A simple improvement to the application could be making a new cursor that contains just the entry needed which would decrease the memory used to process and store the results.

```
private Cursor getAllEntries() {
    return mDb.query(
            DiaryEntriesContract.DiaryEntry.TABLE_NAME,
            columns: null,
            selection: null,
            selectionArgs: null,
            groupBy: null,
            having: null,
            DiaryEntriesContract.DiaryEntry.COLUMN_TIMESTAMP
    );
}
```

PatternsActivity uses a cursor created by a more complex sql query shown below. This query returns, for a particular predetermined “emotion”, the 5 locations with the highest count of that emotion within all entries, ordered from highest to lowest. This is then used in the Patterns Activity to display the top 5 locations for a particular emotion.

```
String sql = "SELECT " + DiaryEntriesContract.DiaryEntry.COLUMN_LOCATION + ", "
        + DiaryEntriesContract.DiaryEntry.COLUMN_EMOTION +
        " , COUNT(" + DiaryEntriesContract.DiaryEntry.COLUMN_EMOTION +
        ") as Total FROM " + DiaryEntriesContract.DiaryEntry.TABLE_NAME + " WHERE "
        + DiaryEntriesContract.DiaryEntry.COLUMN_EMOTION + " = "+ emotion +
        " GROUP BY " + DiaryEntriesContract.DiaryEntry.COLUMN_LOCATION + ", "
        + DiaryEntriesContract.DiaryEntry.COLUMN_EMOTION + " ORDER BY COUNT(" +
        DiaryEntriesContract.DiaryEntry.COLUMN_EMOTION + ") DESC LIMIT 5";
Cursor exCursor = mDb.rawQuery(sql,  selectionArgs: null);
```

Recyclerview

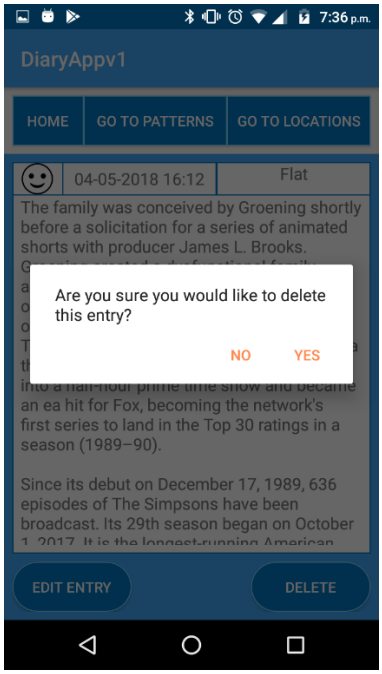
In the application, to view a list of the entries or the locations we have to populate that list with many items. For diary entries, this involves loading in the image files for emotions, as well as assigning onclickhandlers to view specific entries. To avoid delays due to processing time or running out of memory (if we had a lot of entries), the application uses a recycler view to display entries in MainActivity and locations in ViewLocationsActivity. This allows us to display a list of the same item type, as a series of views populated with different data sets (entries).

DiaryEntriesAdapter (in MainActivity) and LocationAdapter (in ViewLocationActivity) extend RecyclerView.Adapter. Recycler view uses the adapter to provide it with new views. The Adapter binds data from our database tables to the views via a cursor (that returns all entries or location tags from their respective tables) in the onBindViewholder method. The adapter sends the views to the RecyclerView as ViewHolders, and these are then displayed the in the recycler view. We then just need to set the layout with the layout manager in the respective activity using the RecyclerView using setLayoutManager. The application uses linear layouts as this gives the most natural way to scroll through large lists.

In Main Activity, the recycler view has an Item Touch Helper attached to it. By creating an item touch helper and overriding the onSwipe method the activity can react to entries being swiped. This action initiates the dialog created to handle deletion of entries. If the dialog is responded to positively, the entry is deleted from the database, otherwise it is ignored. In either case, we swap the recycler views cursor to refresh the view. This function offers “expert” users an easier method to delete entries, the dialogue ensures that inexperienced users or accidental use does not unintentionally delete entries.

Dialogs

Dialogs were used within the application to ensure an adequate response to the request for the deletion of data. To ensure the user does not lose data through accidental deletion, the user is prompted to confirm their choice. Upon selection of the delete button in View Entry, or upon swiping an entry in the main activity (another way to delete entries), the user is prompted with a pop up menu in the centre of the screen. The use can either choose to confirm this choice or cancel it by either selecting no or pressing off the dialog.



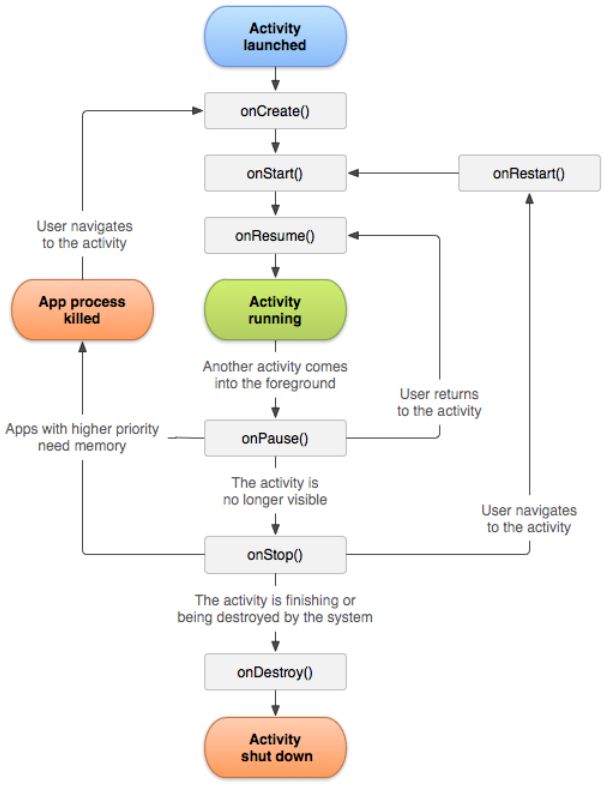
The code below shows the implementation of the dialog for the View Entry's delete button. In the onclickListener, an alert dialog is created with AlertDialog.Builder, several settings are then set. Set Cancelable allows the user to cancel the action by clicking off the dialog anywhere on the screen.

The message for the dialog is then set and a negative and positive button are set for the dialog. The dialogs positive button carries through the action whereas the negative button just cancels the action returning the user to the View Entry activity.

```
    deleteButton = findViewById(R.id.delete_button);
    deleteButton.setOnClickListener((view) → {
            AlertDialog.Builder builder = new AlertDialog.Builder( context: ViewEntryActivity.this);

            builder.setCancelable(true);
            builder.setMessage("Are you sure you would like to delete this entry?");

            builder.setNegativeButton("No", (dialogInterface, i) → {
                    dialogInterface.cancel();
            });

            builder.setPositiveButton("Yes", (dialogInterface, i) → {
                    deleteEntry();
            });
            builder.show();
    });
}
```

5.5 Activity Lifecycle

Understanding the activity lifecycle helps ensure the application reacts the way we want to various states the activity could be in. The callbacks provided by the android platform allow the application to react to events, such as another application or activity taking focus on the screen or the activity becoming full screen again, after the user has returned from another application or task. See below for a useful diagram showing these stages taken from the Android Developer documentation.



The application reacts to some of these callbacks to handle any tasks we need to ensure the activity is providing the service it should be. All activities must override the onCreate method to react to their creation and usually compromise of building some sort of User Interface and the methods needed to generate these dynamically.

The MainActivity overrides onRestart to ensure that the recycler view displaying the user's entries is up-to-date. There may be entries added; deleted; or changed from the diary in activities that proceed this activity. When returning to this activity, it is important to make sure the entries displayed reflect the true state of the diary and its contents. This is achieved by swapping out the cursor within the DiaryEntriesAdapter used for the recycler view. The same process happens with the LocationAdapter and viewLocationActivity, on restart the LocationAdapter, swaps its cursor to ensure that any locations added to the diary application are now present in the recycler view.

viewEntryActivity overrides the onResume method to ensure the diary entry's text is correct and up-to-date. When the user finishes editing an entry, they will be returned to this activity. The onResume method retrieves the latest entry text replacing the previous entry's text to reflect the latest changes.

addEntryActivity also overrides the onResume method to handle situations where the location information for the user may have changed. There are two cases where this might be important; if the user goes to the add entry screen but decides to go to another application; or simply ignores the application for a while they may return to the application later wanting to add an entry but be in a new location. This allows the application to check the location again to ensure the current location is used. There is also the chance that connection to the LocationService is unavailable when entering the activity. However, if the user returns to the application when services may have resumed or even when fine location is turned on, the application will automatically try and attain the Location once more.

addEntryActivity also overrides Onpause to avoid unnecessary processing when the activity is not in use, stopLocationUpdates is called to halt the location services. The activity should automatically stop location updates when a location has been received, however if the service is unreachable, the activity may keep calling this method. If the user leaves the applications activity it is unnecessary to keep trying to find a location so the activity stops this process.

6.0 Testing, Results and Evaluation

The following section will provide an overview of the testing used during the project and the results obtained. This will be followed by an evaluation of the system as a whole. The testing of the system has been split up into three main sections, each corresponding to a different stage of the project.

6.1 User Interface Design Mock-Ups

During the early stages of the project, a user interface was designed to provide vision for the application. It detailed basic functionality and was used as a benchmark for early development while I was still becoming familiar with the android platform. As more progress was made on the application, activity screens were added to represent the next stages of development and some changes were made to previous screens as part of the iterative process. When the last iteration of the designs was finished, the user interface was tested for usability and general feedback.

The wireframe designs were built in balsamiq, links were added between activities and elements to represent as closely as possible the actual navigation and transition that would be seen in the application.
User testing involved a number of friends and family who regularly use mobile devices that run android. I asked each of them to perform a series of tasks on the balsamiq web application. User interaction was simulated by clicking on the appropriate parts of the user interface. I also asked them to comment what they were doing and why and what they would do with any non-interactive elements, such as text edits. Testers were given an introduction to the applications aims and objectives and then were asked to perform a series of unaided tasks from the home screen (main activity). After the application test was over, I asked the testers for any feedback they could think of regarding the application mock-up they had just used.
Questions covered in the test covered the majority of use cases, the results can be found in the appendix.

Results

Most testers seemed fairly happy with the applications and most use cases were carried out successfully.

The delete button repeatedly received negative comments and results. Some testers found the functionality of deleting an entry confusing, while other found the interface ugly, especially on the home page. To combat any ambiguity and to make the home page tidier, the delete button was removed from the home page replaced by a swipe to delete interface. The delete button in the view entry activity has been changed to say delete to avoid confusion.

The navigation bar has been added to most activities as discussed in the design section, this was a lack of attention to detail and pointed out by a number of testers.

Testers did not use the implemented back button much and opted for the standard android built in function. The apps back button has been removed to avoid irrelevancy and adhere to Google Material design.

Testers seemed to like the theme and colour of the application however one tester wanted more functionality. It was noted that the “Patterns” option in the navigation bar was a bit undescriptive

and some testers did not know what it would do until they selected it, however users found out very quickly upon using the application what it was for and I feel this is not a concern.

One function that seems to be fairly poorly designed is the ability to add entries for location history, unfortunately time constraints stopped this feature being implemented but if this was implemented in the future this part of the interface would need to be redesigned. Testers commented that more assistance was needed, possibly adding descriptions to the start of each activity may help with this or a tutorial for first time use.

6.2 Testing System Usability and Real User Feedback

After the implementation phase was complete, a series of testing and evaluation was conducted to understand the strengths and limitations of the application. Unfortunately, a number of features that were originally planned for the application were not implemented due to time constraints. The main focus of testing was to establish the usability of the system that had been implemented, in order to establish the extent of what had been achieved and to create a foundation for any future work to take place with the application.

The testing focussed on the system usability scale (SUS). This was chosen based on its previous success with similar projects at Cardiff University within the past. The project was running late at this stage and something quick was preferential, this helped reduce time costs but was also beneficial for keeping the interest of human testers who were doing this for free as a favour. It is also easy to implement, the questions and scoring are predefined and this avoids the need to construct a test while also being seen as solid and dependable by the industry. A copy of the test can be found in the appendix.

Testers had the application installed on their personal devices. However, to ensure accuracy of the scores, testers should have used the same device to ensure no other factors, such as changes in modal and system version had an impact on usability. Personal devices were used because the application had only been tested on one device at this point. This was seen as an opportunity to ensure the application worked on many different devices and had no platform specific bugs. Phone models and android versions used include (Samsung Galaxy A5 – Android 7.0 nougat, Motorola Moto e3 – Android 6.0 Marshmallow, Motorola Moto e2 – Android 5.0 Lollipop).

The testers were briefed on the purpose and aims of the application and then given a set of tasks to perform, similar to those found in the interface mock-up testing. These tasks were observed and noted while being carried out. Testers were then given a week to use the application as they wished, hopefully this would give them a better sense of any learning curve, the longevity of interest in using the application and report any bugs they experienced. As well as the SUS test, I asked testers to note any issues they had with the system, any bugs they experienced and what they liked about the application.

Results

Question Number	Answer	Score
1	3	2
2	1	4
3	5	4
4	1	4
5	4	3
6	2	3
7	5	4
8	2	3
9	5	4
10	1	4
	total	35
	sus score	87.5

Question Number	Answer	Score
1	4	3
2	1	4
3	5	4
4	1	4
5	5	4
6	1	4
7	5	4
8	1	4
9	5	4
10	2	3
	total	38
	sus score	95

Question Number	Answer	Score
1	5	4
2	1	4
3	5	4
4	1	4
5	4	3
6	1	4
7	4	3
8	1	4
9	4	3
10	2	3
	total	36
	sus score	90

Question Number	Answer	Score
1	4	3
2	3	2
3	3	2
4	3	2
5	5	4
6	1	4
7	3	2
8	2	3
9	3	2
10	4	1
	total	25
	sus score	62.5

Question Number	Answer	Score
1	1	0
2	2	3
3	4	3
4	1	4
5	3	2
6	2	3
7	4	3
8	3	2
9	4	3
10	2	3
	total	26
	sus score	65

Test observation and tester feedback shows a similar experience was seen by all on their personal devices which suggests there are no platform specific bugs in the application (at least not for the modals/platform versions being used). This is not a conclusive test but is an encouraging confirmation that the application should work on most if not all android devices currently in circulation.
In general no major bugs seem to exist in the system, two users identified one flaw that was witnessed on 3 occasions and has also been seen through “field testing” of the application. This was an issue with the device attaining the phones location, given the context of when these issues occurred it seems this is down to the particular devices not being able to find user location, this is an inevitable issue with location awareness when in areas of low reception/connectivity and occurs with many applications. An improvement to the system for the future would be to handle this circumstance better, maybe by displaying a message to the user to explain why the location is not working. Some users may not realise why the application is not getting location and may assume this is a bug as opposed to a problem with the devices location.

The SUS scores are very positive and seem to represent a high quality of usability, the scores recorded were 95, 87.5, 90, 62.5 and 65 which have a mean of 80. Any score over 80.3 is considered

excellent and likely to be easy to use and recommended to friends so the average is very close to this ideal category. This suggests the overall usability of the system is pretty good

The average score for an SUS test is 68, any score below this is considered to be highlighting some potentially serious usability issues. Two tester scores were just below this, some particular question scores stand out as being problematic and probably the cause of these deviations. Question 1 regarding wanting to use the system frequently scored fairly low with 2 testers, given their feedback I think this is largely down to a lack of features that were implemented. Testers commented that if all the features originally planned were implemented they would be very likely to use and recommend the application to friends and family. One tester seemed to find the application less usable then others, and a lot of the testers felt that question 10 "I needed to learn a lot of things before I could get going with the system" Was fairly agreeable. Feedback by all testers showed that they felt the application was really easy to use but most agreed that some of the features were not initially obvious and that some sort of tutorial would help really settle them in to the application straight away.

In summary testers seemed to agree that the application itself was easy to use and fairly easy to learn but felt a tutorial or help system would help decrease the learning curve of one or two application features. Testers liked the layout and colour of the application and found the application helpful and fun, all testers agreed that more features would improve the application. In particular all testers wanted to see more patterns as they thought this was the best part of the application, most testers wold like a way to search or order entries by text, location or emotion. Testers agreed that adding an ability to create entries later based on location history would really improve the functionality of the application. These are all features previously considered for the application and a great foundation for future work to build upon.

6.3 System Completeness (Acceptance Criteria)

The following sub section is an overall review of the reliability and completeness of the system in regards to the requirements met, the functionality offered and the extent to which it achieves the goals and objectives of the initial problem.

Acceptance criteria results

Must Have

Functional Requirement 1

- The application allows the user to create an entry and saves this in a database local to the application.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.
- The application is able to receive the user's current location.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.
- The application offers location suggestions within 50m of the user's current location.
  Acceptance criteria met: True
  Confirmed during development testing.
- The user can add a new location tag for the current location.

Acceptance criteria met: True
Confirmed during development testing and SUS testing.

- The entry is tagged with a timestamp automatically.
  Acceptance criteria met: True
  Confirmed during development testing.

Functional Requirement 2.

- The application collects user location at regular time intervals throughout the day.
- **Acceptance criteria met: False**
  Not implemented in time.
- The application presents location history to the user.
  Acceptance criteria met: False
  Not implemented in time.
- The application allows the user to add entries with location and time automatically filled with the relevant location history data.
  Acceptance criteria met: False
  Not implemented in time.

Functional Requirement 3

- The user is presented a set of 5 emotion tags to tag their entry with.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.
- This emotion choice is saved with the entry.
  Acceptance criteria met: True
  Confirmed during development testing.

Functional Requirement 4

- Diary entries are presented within the home screen.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.
- The user can view a specific diary entry by pressing on it.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.

Functional Requirement 5

- The user can view the top 5 locations tagged with a specific emotion.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.
- The user can see the emotion count for a location.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.

Should Have

Functional Requirement 6

- The user can edit the text content of a diary entry after it has been submitted.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.

Functional Requirement 7

- The user can delete any entry saved in the database.
  Acceptance criteria met: True
  Confirmed during development testing
- The user is asked to confirm deletion before entries are removed.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing

Functional Requirement 8

- The user can search for diary entries by text.
  Acceptance criteria met: False
  Not implemented in time.

Functional Requirement 9

- The user can select a location from a list of location tags, the user can view all entries tagged with this location
  Acceptance criteria met: False
  Not implemented in time.

Functional Requirement 10

- The user can select an emotional tag, all entries tagged with this emotion are displayed
  Acceptance criteria met: False
  Not implemented in time.

Functional Requirement 11

- The user can export their data to a SD CARD or Cloud Storage.
  Acceptance criteria met: False
  Not implemented in time.

Will Not

Functional Requirement 12

- The application contains no adverts.
  Acceptance criteria met: True
  No adverts were implemented

Non-Functional requirements

Non-Functional Requirement 1

- Diary entries are displayed in descending (newest first) time order.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.

Non-Functional Requirement 2

- Application text entry does not obscure relevant text box.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing.
- Application is never in a state where the user can't access the home screen.
  Acceptance criteria met: True (accept in edit entry activity)
  Confirmed during development testing and SUS testing.

  Note: Technically this criteria was not met, but in late development it was decided that edit text entry should be made as minimalist as possible, users can easily navigate back to the previous task with the back button and it is unlikely that this would cause an issue if the user wanted to get to home, user testing showed users intuitively how to react in this activity. Therefore it is accurate to say that the application was made accessible and that navigation within the application is complete.

- Application adheres to the Android Material Design standards where possible and does not breach them.
  Acceptance criteria met: True
  This criteria is slightly subjective because of the "where possible" phrase, however as shown in the design section the majority of elements follow the google material design suggestions. The application doesn't breach any of the "rules" that the material design guides recommend not breaking.
- The application has an average System Usability Scale score above 68%.
  Acceptance criteria met: True
  Confirmed during SUS testing.
- The application reacts to the user's orientation in a responsive manner.
  Acceptance criteria met: False
  Separate xml files for landscape orientation need to be developed, it was confirmed during post development testing that the applications layout reacts badly to landscape view.
- The application scales well to larger devices such as tablets.
  Acceptance criteria met: False
  Material design and basic android principle suggest that separate xml files are created to handle larger devices, due to time constraints these were never implemented. The application should still be functional and react fairly well to larger devices, however I did not have such a device available to test this and without dedicated xml files to make sure the app scales well we can't be sure.

Non-Functional Requirement 3

- The application does not crash due to unexpected errors more than is reasonably acceptable (The application does not crash more than 1% of the time).
  Acceptance criteria met: True
  Confirmed during post development testing and SUS testing. All testers reported no crashes, during post development no crashes occurred so it can only be concluded that the applications crash rate is 0% based on current data. Most testers used the application on at least 5 separate occasions and post development testing has meant opening and using the application a minimum of 84 times (calculated as using the application for 3 weeks at least 4 times a day).
- All patterns/statistics generated process and display a result within 2 seconds.
  Acceptance criteria met: True
  Confirmed during post development testing and SUS testing.
- Recycler views generate contents within a second.
  Acceptance criteria met: True
  Confirmed during post development testing and SUS testing.
- Users take no longer than a couple of minutes to complete application functions (Add Entry, View Entry, Delete Entry, Edit Entry and View Patterns/statistics) during testing.
  Acceptance criteria met: True
  Confirmed during SUS testing.
- The application can access the user's location at least 95% of the time.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing. A reported 3 issues with location during SUS testing and 1 issue in post development testing. This should account for time when the user's location services have been inaccessible. Each tester opened the application a minimum of 5 times during their time with the device (so a total of at least 25 uses across the user testing) with at least 84 uses in post development testing that is 4 issues for a minimum 109 attempts. At least a 96.33% success rate.
- The application can be run on all currently supported android versions, this includes every android version from 5.0(lollipop) upwards.
  Acceptance criteria met: True
  Confirmed during development testing and SUS testing. During SUS testing the application was tested on android version 5, 6 and 7. Post development testing was mainly done on a device running android 6 but a device running android 8 was borrowed to confirm functionality was working.

Will Not

Non-Functional Requirement 4

Acceptance criteria:

- The application does not attempt to access the internet.
  Acceptance criteria met: True
  No calls to internet based services are used.

7.0 Evaluation

The finished application has achieved the majority of the basic goals and aims of the project, all must requirements have been satisfied except Functional requirement 5. This requirement refers to the ability for users to create diaries from their location history at a later date. This is the biggest issue with the application and was not finished due to a lack of development on background collection of user location. This would be the starting point for any future work done on the application.

The application does however offer the user a way to create diary entries, which are stored locally to the device. These diary entries can be tagged with emotion and are automatically tagged with the time/date at creation. The application checks the user's location and offers a personalised way to tag this location. These locations can be used to automatically fill the location name for an entry in future. The user can view entries through a scrolling view on the home page or choose one of these entries to view in full screen. The user can also view some statistics related to the entries emotional and locational content by using the patterns activity, this could be another area to improve upon. Testers loved the patterns activity but wanted more statistics and patterns available. The application also lets the user delete and edit entries.

The SUS testing provided good results for the systems usability. This is a very encouraging and although the system has some areas that would need work in future to enable full functionality of the original aims and objectives, the test shows that the basic features that have been implemented in a fairly effective manner. The current process is producing good results. This is due to careful attention to design conventions such as Google's Material Design, and real user testing to see how users actually react to the application. The system is also robust, there have been no reports of crashes and the locational services have been very reliable, any issues with location have been attributed to the phones connection to GPS services.

The Agile method used has suited the project and without it the project may not have produced the results it did. The initial planning and design started off in a fairly waterfall like manner to provide an over view of what needed to be achieved in implementation. However, requirements; flows; and designs; were regularly updated at later stages to reflect discussions with the supervisor, feedback from user testers and developments in implementation. The iterative approach to the implementation was important, particularly as I was learning more about android development as I proceeded through the project. This meant redeveloping certain aspects to which I found better solutions. The agile approach allowed the project to focus on getting a working version, before proceeding to improve the application by adding functionality one requirement at a time. In summary, the agile development process ensured that a working application was delivered by the deadline, it also helped refine the interface which has resulted in great usability results.

Android was a good choice for a mobile application of this nature, it has the largest user base and a wealth of different packages and API's, making the platform very accessible to a wide audience and easy to develop with due to the amount of support available. Care has been taken to ensure the application works across the currently supported android platforms, android studio recommends a minimum SDK support, the minimum was set to SDK 15, this supports 99.7% of current devices. The platform also offers the application the ability to work in a number of different devices and sizes, although care was taken to ensure layouts worked on different size screens, future development should include specific xml files for landscape screen orientation (which currently produces unusable results) and tablet sized devices.

Designs have followed the material design guidelines, further development and investment in this could produce an even better user interface. No rules were broken within the design guides, however certain additional design concerns such as shadows, side menu bars and other features could be added to make a more complete interface, especially with further development of system functionality. The biggest floor in the implementation and design of the system is the lack of design modelling used. Due to this being my first android project, much of the process was trial and error and resulted in a lot of awkwardly put together code just to learn how it worked. Code was tidied up during the course of the project and some modular methods can be found within the system however the lack of strong modelling means the system is not as extendable as it could be. In future development or new projects I would try to follow the modal-view-presenter approach throughout the application.

8.0 Future Work

The following section details the potential continuation of the project with improvements to the current system, as well as functionality that should be added to improve the overall user experience. These have been collected into functionality, user interface and Structure and Implementation.

8.1 Functionality

Location history

Functional requirement 5 was the ability for the user to add an entry for a location they had previously been, based on location history. The main reason for this feature not being implemented was the need to find a suitable way to collect the user's location data in the background, without needing to have the application open. Implementing this function would be one of the most important pieces of functionality to add to the application post project.

Implementation of this feature would need to consider some ethical and functional issues. The task should not be intrusive and it should also be transparent, users should be aware that a background task is running before it starts. Limiting the power usage would also be a preferred solution, achieved through limiting the amount of times through the day the location call is made, this could even be set by the user and allows them to personalise their experience.

This task was achieved by previous projects of a similar nature, this would be a great foundation for the applications background location gathering, maybe even avoiding the need to create any new code.

The initial user interface designed for the "Add entry by location history" functionality had a fairly negative response from testers. This would almost certainly be improved with a few minor changes, for example changing the option "Locations" to "Location history" would probably make the functionality of this activity more obvious. The application could either have some useful hints or even provide a quick tutorial for first time users to speed up any learning time.

One concern with the original designs would be how to display the location to the user. This would probably require an address or place name to be generated for new locations, or the application could display a location on a map like in the initial design. In either case this would require access to web based services, the obvious choice for these would be the Google Maps API's. The benefit of such a feature would probably justify the use of internet service however, ethical considerations would need to be taken, ensuring the user was fully aware of the access being used and what this was for.

Improve data analysis and patterns

User testers like the patterns activity but most felt this could go further, the application could easily add some extra simple pieces of information to the activity giving the user more to look at. Current and future statistics could also add more options for customisability, for example changing the time range of entries analysed to last week or last month.

The vision from the start of this project was that this application could be extended to show much more complicated and obscured information and patterns. Adding modules that do complex datamining on the entries would be the ultimate goal to really show the users useful information and patterns they never even thought might exist.

Currently the application focuses on the casual diary user's objectives. However, more complex emotion processing would help with the aim of giving users information that would allow them to make alterations to change their life. This could offer many things but an example would be the ability to suggest to a user what time they prefer visiting locations, or even suggesting daily routines that would help give a positive improvement to mood.

Search function

The user would really benefit from being able to search for or categorise entries, especially when the user has created many entries.

The user should have a feature present from the home screen allowing them to search for entries containing a particular word or group of text allowing them to find specific entries quickly.

The ability to group or limit entries by location or emotion would also be a great function to introduce, categorisation of entries would be very easy to implement. The application could use the same basic frame used for the standard recycler view and adjust the SQL query used to return results to the cursor. The SQL query would simply use a variable defining the emotion or location to filter results, a "WHERE emotion/ location = variableName" type addition to the original statement used would achieve this.

In regards to displaying the choices, changing the location tags to be displayed in alphabetical order would also be a helpful and quick change to the user interface.

Application Personalisation

In "add entry" locational tags are suggested based on those within a 50m range of the user's current location. It would be beneficial to some users to be able to increase and decrease this range to make the locations suggested broader or narrower. For example, if a user travelled a lot and simply wanted to tag entries by town or country, the radius needed would be much larger to link entries with the same regional location. Allowing the user to personalise this range would allow them to get a more personal experience from their diary.

The user should feel the diary is personal to them, being able to change user interface settings such as colour, text size and themes would help achieve this. This would be fairly easy to implement using a "burger style" material design friendly menu to control settings, and these would then be used to switch in and out appropriate xml files defining different styles. Using shared preferences should be an easy and memory efficient way to keep track of the user's preferences.

User Interface Development

The system testing suggests the application is very usable, this was helped by following the Google material designs. Further development of the UI in future should ensure even more compliance with these guidelines.

The application handles different sized screens by using adaptive layouts, however it does not handle orientation and has not been tested on tablet sized devices. In order to solve the orientation issues and ensure optimized size specific layouts more xml layout files should be developed to handle this. Android allows the creation of multiple xml layouts for the same activity, these files are then given specific name endings to denote different contexts, the platform substitutes the appropriate layout for the activity based on the context. This can be used to handle orientation,

specify optimisations for specific screen sizes and also create layouts customised for specific counties in order to change culture specific symbols and offer extras language options.

8.2 Structure and Implementation

The application was written with the focus of achieving functionality over modularity. When implementation of the system started, personal development with android was still being undertaken. A lot of the initial coding was my first experience with particular functionality and standard classes, this lack of experience with android lead to some of the code being less extensible than would be preferred. Android encourages a modal-view-presenter approach to the structure, some parts of the system represent a more modular approach for example an adapter is used to get data from the modal (database), this is then given to the recycler views that present them within the applications activity. Redevelopment of the application should make sure MVP is further implemented within the system providing a more extensible and easy to work with system.

The two database tables used by the system are currently stored in two separate databases, there is no need for this to be the case and it would actually be more beneficial for the two tables to be separate tables within the same database. The two tables are used by different parts of the system, the part that uses the location table was developed separately and added at a later stage. This would be an easy change and would make a lot more sense if implemented.

The view entry activity currently uses a cursor containing all entries within the database, for efficiency it would be better to rewrite the query used to populate the cursor to return the one entry that is needed rather then return all and point to the relevant entry.

9.0 Conclusion

The project aimed to create a diary application that used user location and emotional tagging to create an application that was personal and could help the user learn something about themselves.

The project involved researching relevant information and factors to fully understand the problem, as well as the best approach to solving it. It also involved the design of a user interface that promoted usability and likeability and designing a system that could support the functionality of the application. The main scope of the project involved the implementation of the application based on initial designs, tester feedback, advice from my supervisor and late developments and lessons learned during the project. The system was then tested and evaluated to assess the success of the project as a whole.

The application meets the majority of the major requirements that were set as the project began. The application does not collect users locational history so implementation of entries based on location history was unsuccessful, this is the only major use case not fulfilled from the original vision of the application. The application was successful in achieving the projects main aim, the fusion of location and emotion to create a personal and helpful diary application. The user can create entries and view these at any time, these entries can be deleted and edited and basic data analysis provides users with interesting facts about their entries and their relationship with locations.

Location is used to assess the user's current location, the user can customise these location with their own names rather than rely on impersonal generations from services such as Google Maps. The application then uses this to suggest location to the user next time they write an entry. The application offers a range of emotions to tag entries with so users can see where they are often happy, sad or angry with the applications analysis.

Users can also see a breakdown of how a place effects them via an emotion count. Most importantly the application has laid the ground work for further development into exploring user emotion and location, possibly with the extension of time/date to assess user routines. I would like to further develop the application with the vision of more powerful data processing to help users that suffer depression, anxiety or other mental health issues see the patterns that affect them positively or negatively as well as help identify the barriers their life presents them with.

The project involved researching the ethical and social implications of the technology, not just the physical technological limits. Project development processes were researched and an agile/iterative method chosen, this complimented a user centred approach and helped create a highly usable interface, it also ensured an initial working system was available early in the implementation phase ensuring basic functionality was present.

The application has been tested through its design, implementation and evaluation phases. Testing of this nature ensured a user focussed design and a stable system. Interface testing established an early benchmark for the system and development testing ensure the system was robust upon completion. SUS testing helped ascertain the usability of the finished application while post implementation testing checked the system for bugs. Further development should follow this iterative and constant testing process to ensure higher quality software.

In summary the project has met most of its main aims and objectives to some degree, the key areas that should be implemented or improved in future have been discussed in the future work section. One of the main success of the project is the testing and focus on usability of the interface, future

development should follow this pattern. With a large part of the project having been spent learning android development the focus was not on creating an organised modal, future effort should be focused on modelling the system into a cleaner more extendable system.

10.0 Reflection

The following section reflects on how the project was handled personally and what was learned through the process.

10.1 Management, Organisation and Communication

The management of the project was handled with varying success, however, I feel I have learned a lot in areas I could have improved. Initial planning and preparation for the project was fairly strong, as I had started research into the various aspects of the project early and a plan was put in place with overall reasonable time constraints.

The biggest issue with managing the projects time scale was not accounting enough time for learning android development. This process took a number of weeks and was ongoing when implementation on the application had already begun. This lead to some implementation being changed at a later date causing further delay. Personal issues and health also set the project back a number of weeks. I feel this time was recovered from however, it is one of the main reasons for certain features being absent. In future, I would reserve some time for unexpected delays such as the latter, the 3 week period break should have been reserved in such a manner rather than factored into implementation time.

Communication between myself and my supervisor was fairly good, as meetings were held almost weekly and these meetings really helped to focus the weeks work. The feedback on progress offered was also vital and had it not been for these meetings, the projects progression would probably have been far slower. Google Drive was used to share valuable pieces of background information, as well as help for formatting and planning the report. I used Google Drive as a repository for report progress to get feedback on what had been done. Regular e-mails also helped keep both parties up to date with developments.

I tried to record my thoughts and notes during the entirety of the project for later use, although this was really helpful when writing the report, it could have been organised far better and will try to do so in future. Some early parts of the report were started (and completed) at the time of carrying out the relative tasks, for example a lot of my research was documented within the first few weeks. I failed to document the development of the implementation at the time, in future especially when following an agile method I will document pieces of the system when they are created. It would be far easier to make changes then try to explain the entire system at a later date, in some cases this was more than a month after certain features were implemented.

Version control was implemented to a degree, after new parts of the system were implemented I saved a legacy file in case of emergency and to track progress for the report. In the future I will document this process better and also this process would be far more efficient using Github.

One final issue I had during the later stages of the project was stopping the implementation phase at a realistic time. Wanting to complete as much as possible, I was changing minor parts of the system in the last few weeks. This put a huge time constraint on the latter stages of the report writing and the user testing. In future, it would be far better to stick to a deadline date at which all implementation stopped to ensure that the project was properly tested and documented.

10.2 Learning

The project has been a great opportunity to learn a lot of hard and soft skills and techniques, these skills will help with my future personal development, in particular the project skills learned should help with future personal projects and my future career in the field.

The largest hard skill I have learned is an overall understanding of the android system and developing applications with the Android Studio IDE. Previously I had some experience with Java through my course in University, this helped understand some of the basics but was fairly different to anything I have done in Android. I now feel confident that I would be comfortable developing these skills in future and I also feel like it has added to my experience with java programming.

I have also learnt a lot about user interface design and usability, although I have had previous experience with design on modules within the course this was the first time I had looked at design guides in a detailed way. There is a lot of room for further knowledge about Google's Material Design, but I feel confident in the basics and feel developing this in the future will help in many areas that will involve design generally, not just for Android platform.

During second year of university, I was tasked with leading the team during the group project, although this helped with planning and organising this project, this was the first time I had to plan, track and implement the entire system on my own. Dealing with time management was a lot harder when trying to implement the bulk of a system myself. The project has developed skills to handle both organisation and management coinciding with implementation. This is the largest software I have ever made and have never made anything with so much functionality, it has given me vital experience with user interfaces having only designed two previously for single activity basic programs.

I have had to learn to manage setbacks due to personal issues and illness and address the effect this has had on the project. My supervisor has been a great source of advice and focus during these times, it has helped me learn to handle the situation more effectively and focus on what needs to be achieved next, rather than worry about the effect on the project as a whole. This has led to a much better coping strategy in such situations that I will use in the future.

Other skills that have improved during the project include research, communication, time management and organisation skills. The process of writing the report, in particular the implementation section, has really highlighted the importance of creating modular systems. I found that describing the system was complex and at times over complicated, this has highlighted that the application may be harder for someone else to extend.

References (sorry didn't have time to do this properely)

Location research, implementation and google maps

Android location - https://developer.android.com/training/location/

Location-based apps present opportunities and data challenges - http://searchcrm.techtarget.com/feature/Location-based-apps-present-opportunities-and-data-challenges

Google maps and android development - https://developers.google.com/maps/documentation/android-api/start

And - https://developers.google.com/maps/documentation/android-api/

Udacity course - https://classroom.udacity.com/courses/ud876-1

youtube vid - https://www.youtube.com/watch?v=4bNT-2r8NOA

Android Development Courses and information

Information on where to start - https://lifehacker.com/i-want-to-write-android-apps-where-do-i-start-1643818268

Android studio guide(invaluable as a quick over view before jumping into trying it out - https://www.androidauthority.com/android-studio-tutorial-beginners-637572/

Android development information from android (where I started but wanted something a bit more interactive - https://developer.android.com/training/basics/firstapp/index.html

Udacity 2 week course (not used) - https://eu.udacity.com/course/android-basics-user-interface--ud834

Udacity course - https://classroom.udacity.com/courses/ud851

Android developer's docs - https://developer.android.com/docs/

DESIGN and implementation

Richard Saul Wurman, Information Anxiety (LATCH)

Android material design - https://material.io/guidelines/material-design/introduction.html#

Android developers design - https://developer.android.com/design/index.html

Activity lifecycle (with diagram) - https://developer.android.com/guide/components/activities/activity-lifecycle

tabs - https://material.io/guidelines/components/tabs.html

navigation drawer - https://developer.android.com/training/implementing-navigation/nav-drawer?authuser=4

menu - https://material.io/guidelines/components/menus.html#

colour picker - https://material.io/color/#!/?view.left=0&view.right=0

Dialogs

https://developer.android.com/guide/topics/ui/dialogs

https://www.youtube.com/watch?v=7vWoi8j5vL4

Emotion

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2756702/ Iris B. Mauss and Michael D. Robinson

Measures of emotion: A review

Facebook- facebook.com

Forbes - https://www.forbes.com/sites/amitchowdhry/2017/03/02/facebook-confirms-emoji-reactions-affects-your-news-feed/#430155c213bf - Facebook Confirms Emoji 'Reactions' Affects Your News Feed

Applications/resources used

Balsamiq - https://balsamiq.com/

Wireframe.cc - https://wireframe.cc/

MockPlus https://www.mockplus.com/

Article - https://www.shopify.co.uk/partners/blog/wireframing-tools-2017

Android Studio - https://developer.android.com/studio/

CONCLUSION EVALUATION

System Usability Scale - https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/

Sus picture - https://www.researchgate.net/figure/SUS-Questionnaire-Source-Brooke-J-SUS-A-Quick-and-Dirty-Usability-Scale_fig4_300327246