Need2Want: Personalized Fulfilment Prediction App

CS39440 Major Project Report

Author: Runar Reve (rur7@aber.ac.uk)
Supervisor: Dr. Chuan Lu (cul@aber.ac.uk)

30th April 2020
Version: 1.0 (Release)

This report was submitted as partial fulfilment of a BSc degree in Computer Science and Artificial Intelligence (With Integrated Year in Industry) (GG47)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, U.K.

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name Runar Reve

Date 29th April 2021

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Runar Reve

Date 29th April 2021

Acknowledgements

I am grateful to Dr Chuan Lu for her guidance and support throughout this project. She gave me valuable input on where and how I could improve upon this project.

I would like to thank all my friends and peers for supporting me during my time at university, always inspiring me to do grow into a hopefully better and more intelligent person, despite my flaws and downfalls.

I will be forever thankful of the university who has set up various infrastructure and opportunities for me. The main opportunity was my industrial year placement, where I got the chance to travel to Saudi Arabia and work under Professor Robert Hoehndorf in KAUST, giving me valuable experience to kick-start my professional career.

Abstract

Today, there is a large focus on mental health and tracking various traits, for example to remind to drink enough water or a pedometer apps to encourage to walk a certain number of steps. This is important traits to focus on, but they often lack personalization by adjust to the user's preferences and the capacity of correlating the data to the user's well-being.

This project attempts to tackle this problem by collecting daily data with an android application from the user. The data is split into two sections: wants (subjective traits that user can not directly influence such as happiness) and needs (active traits that the user can influence, such as socializing or activity). This application is not restricted by the traditional pre-set goal, such number of steps taken, but rather ranking their feeling of fulfilment of each need makes this application more subjective to each user.

As people are different from each other, even different from themselves on a day-to-day basis, with the amount of focus needed for the different needs. Making it more feasible to track their daily fulfilment of each need.

Additional to finding correlation in for the user themselves, there is also possibility for each user to opt into shearing their data anonymously for cohort studies. This is only available for authorized users to access the anonymous data and run their own studies.

Contents

List of Figures

List of Tables

Glossary

Before embarking on this report there are a few key words frequently used to more effectively convey subjects that has a slightly different meaning than what they commonly represent.

Word	**Description**	**Examples**
Want	Feelings, Factors one cannot directly control	Happiness, calm, productive
Need	Factors one can control	Movement, sleeping, code
Question	Often a grouping of wants and needs	Users answer the questions

Table 1: Glossary of commonly used word

Chapter 1

Background & Objectives

1.1 Background

Low fulfilment of needs, such as social interaction, have a been studied to have a negative effect on individuals happiness [1]. Happiness also been shown to be correlation to health [2] and not fulfilling the need of socialising has the same effect on health as obesity or smoking [3]. From this one can assume that to fulfil one's needs is important for one's health and well-being.

In addition to fulfil ones need, there are differences between individuals. Depending on various factors such as: age, gender, and background. These varieties can influence what need that has the most impact on improving their well-being. Some might have a better day when they fulfilled their needs of movement, while others might have better day when they have socialized.

By having an accessible and flexible method of tracking and analysing one's wants and needs, can lead to one getting better chance identifying what areas of one's life to focus on to have a better chance of improving their life.

Similar of understanding oneself there is additional potential for understanding population as well with similar techniques. By utilizing the collected data from a broad number of people using the app could lead organizations and institutions to improve their environment, similar to a single person would do for themselves. An example could be for a software company study and identify that their employees are more productive when they are active and happy.

1.2 Background

Several applications have been considered when getting idea and finding niches, such as Samsung Health [4] or Huawei Health [5]. This showed that application normally only monitor a few key features of the users, like steps and calories. Leaving little room for customization, not allowing the user to add other features personalized for them.

The focus of these apps often is to encourage the user to complete an arbitrary pre-set goal. Such as the commonly believed 10 000 steps/day necessary for good health, which has limited scientific basis [6]. These apps allow small tweaking of the goal, but do not consider that humans are dynamic beings, that often changes on a day-to-day basis. A person might go on a long hike, then be satisfied in that area for the next coming days.

There is little consideration on how the features collected affect each other. If there is a method, it is often a set function, calculating for instance the number of calories burnt from number of steps. These functions do not consider that everyone is a bit different. As the example given earlier, people can have different metabolism leading to incorrect number of calories that has been burnt. There is usually no method of learning and personalising the function for each user.

1.3 Requirements Analysis

From the description above there was a clear indication that having a method of understanding oneself can potentially greatly improve one's life in several areas. By combining the areas of data collection, statistics, machine learning, and data visualization, into one application that can clearly communicate to the users to better understand themselves and help them what need to focus on, can potentially affect their well-being over time.

Before any analysis of data can be done, collection of the data is required. By collecting data from the user on regular intervals, ideally at the end of each day. The longer a user inputs data, the more personalized and statistical significance the predictions will become.

The analysis will start off with a *cold start* (little/no starting data, resulting in faulty and overly sensitive predictions). To try to avoid this there should be methods avoiding this, like generating synthetic data. By implementing this data, it will make the prediction less sensitive to each new input when starting the starting using the app.

As the data of any users needs to be stored somewhere. By storing it remotely will allow for any user to easily access their data across multiple devises. This will require authentication and access control to secure that any user only have access to their own data.

When storing data of several user in the same location, there is a potential for setting up methods to study the populations. It should be up to each user if they want to participate and allow their data to be used for any studies. The process of opting in/out should be clear and simple for the normal user and requiring no explanation why they might change their mind. To increase the potential studies that can be done, the user could give more information about themselves, such as age, gender, and ethnicity. The user will in this case have full control what data they want to share and should always have the option to edit or retract any of this information from future potential studies.

1.4 Methodology

The process that has been chosen for this project is a mix of Kanban and FDD (Feature Driven Development). These two methodologies work well together as Kanban can structure what feature to research and develop. As the software develop there is certain to identified additional features to add or removed. Because of the flexibility of agile, it allows to focus on implementing new features to improve the outcome of the project over needing too spend to much time on documenting and restructuring the original design.

To get a better overview on the functionality and what features required, a short planning stage was initiated the first few weeks, compressed version of this can be seen in Appendix: E. During this time, the main design choices was done such as the user interface. Once implementation started a skeleton of the app could quickly be put together, allowing to easily start implementing features and extend upon the project. This initial design set out key milestones when parts should be completed, allowing smaller intermediate goals to strive towards, leading increase the moral and discipline for the developer.

When using the Kanban methodology there should be a Kanban board. This was put up together with the blog/diary to centralize as much of the processes as possible. The philosophy behind keeping as much in one document was that it makes it more easily update on the process without spending too long on locating the specific document/website needed. It can also be a good method of remind to update various sections more often. As a bonus this leads to less physical and digital clutter to manage.

1.4.1 Blog/Diary

It is important to keeping everyone involved or interested in the project updated on the progress and a history of what has been done and when. This project used an Overleaf [7] document to host the blog. Overleaf has many advantages, large amount of support, being freely accessible online, and access control (mainly used for its read-only control) for external users. Blog is accessible through: `www.overleaf.com/read/zhzbjscrfzzq`

1.4.1.1 Blog Structure

A standard format to keep each entry structured and easy to read can more easily get the key points across about progression and plan. This structure also makes it easier to start documenting, as it only need to be filled out with short explanations.

- **Meeting Summary**: Key points from weekly meetings
- **Progression**: What has been done
- **Focus of Next Week**: What will be focused on
- **Problems**: Unexpected delays
- **Ideas**: General ideas on how/what features to add

Chapter 2

Design

2.1 Overall Architecture

Starting a larger project, it can be beneficial to formally structurally the requirements. This allows for an easier communicate on what features can be expected to be in the final product for the customer, developers, and any other parties involved in the process. Figure 2.1 displays what features will be accessible for a normal user. Then from Figure 2.2 one can observe the general process of an expected interaction with the app and how application will handle the various interaction of the part in the system.

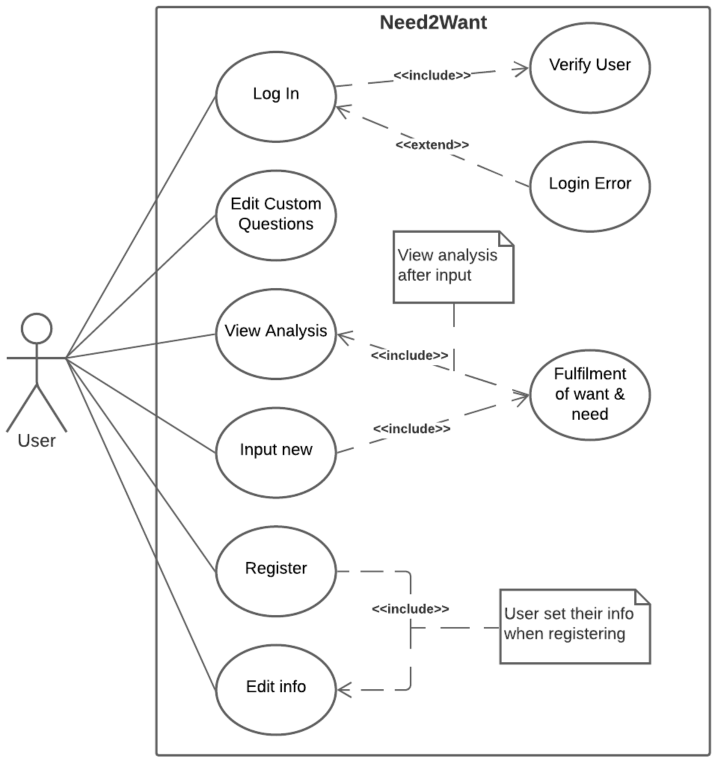


Figure 2.1: Use Case Diagram

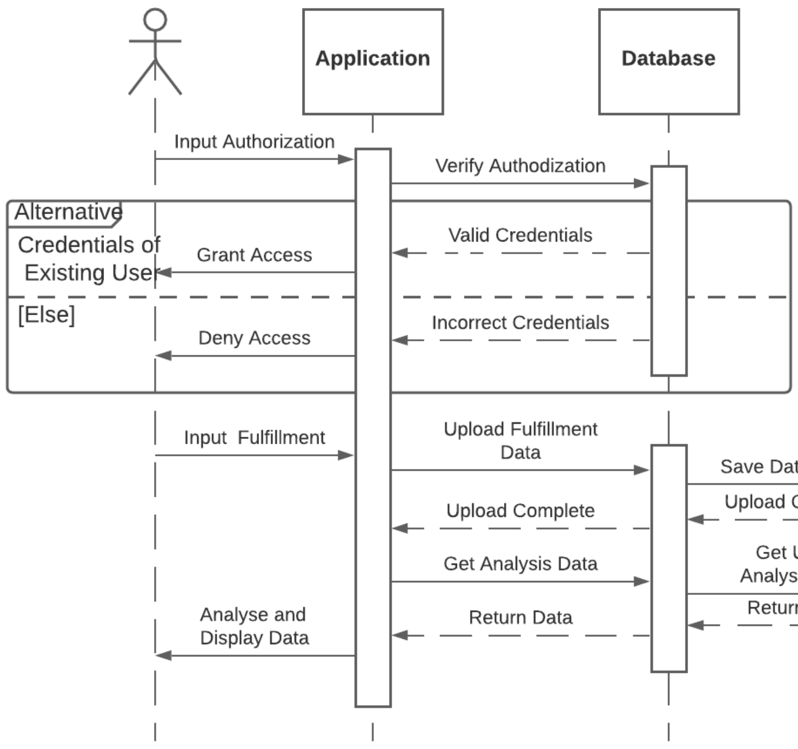


Figure 2.2: Sequence Diagram

2.2 Main features

This section will look closer at the different key features throughout the system and discuss them more in-depth and how they interact between each other. Figure 2.3 gives a simplified overview of the interaction and life cycle for users using the app.

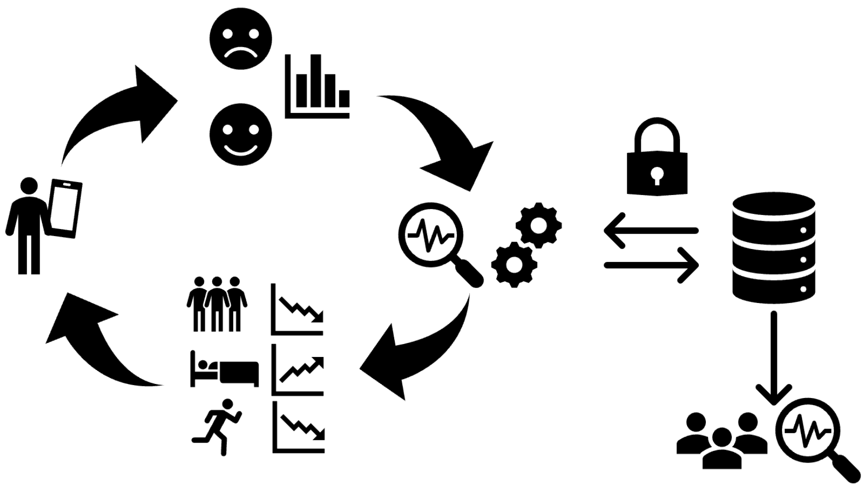


Figure 2.3: Life cycle for the user

2.2.1 User Input

A user accesses the project through an android application on their phone. Android has been chosen for its vast numbers of useful open-source libraries and support from the community. The user is expected to come back to the app on regularly basis to input their daily mood, ideally coming back at the end of each evening to reflect on their day.

Figure 2.4: User Input

From the analysis the user can get a better insight into themselves and what they can focus upon to have the most improvement for what they want to focus on. As the user uses the app it improves the prediction on them to fit themselves even more.

2.2.2 Rating system

An easy and comprehensive method for collect the data of wants and needs from the user is key. That is why a short survey style has been chosen, as most people are already familiar with this method from many other fields. It also keeps the questions in a standardised form. Minimising time spent in the app which reduces frustration user might encounter.

Figure 2.5: Rating inputs

A hand full of questions has been selected as default for everyone:

- **Happiness**: Want
- **Sleep**: Need
- **Movement**: Need
- **Social**: Need

Allowing for the user to add their own questions can make the experience be more personal and flexible to their desire. Some examples of new questions to add are: ”Productivity” (want) and ”Code” (need).

2.2.3 Analysis

Raw data is useless on its own. Therefor there needs to be methods to analyse the data gathered from surveying the user. By finding correlation between the wants and needs one can get a better understanding of the relation between them. The 3 method that has been used are as followed:

Figure 2.6: Analysing data

2.2.3.1 Pearson correlation coefficient

A well-known method of finding correlation between two datasets is to use Pearson correlation coefficient. By run the user’s data through the formula underneath one can calculating the correlation between the different wants and needs. As the data from surveying are all normalized to the same range, Pearson is ideal as it compared correlation with the raw values.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2(y_i - \bar{y})^2}} \tag{1}$$

2.2.3.2 Spearman's rank correlation coefficient

Like Pearson, Spearman find the correlation between two datasets. What separates the two is that Spearman normalizes the data from the raw value into ranked values based on significance. This makes it more ideal to compare two different sets of data which is not in similar ranges. Because the data used in this project is mostly in the same range, and the range is not significant, Spearman correlation is not the most optimal for the normal use cases. Except it prepares the system to allow calculating the surveyed data with other extensions, such as steps count from a pedometer.

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2 - 1))} \tag{2}$$

2.2.3.3 Linear Regression

The two previous methods give a good indicator on how correlated two sets are, but it does not give an indication on how much they affect each other. For example, x = [1,2,3,4] and y = [2,4,6,8] will output a correlating of 1 for both methods, but will not tell the user that dataset y increases by 2 for each increase in x, $y = 2x + 0$. This method is important for the application as it indicates to the user how much a want is affected by each need.

Linear regression will assume that the relation between the two given lists is linear and will starts off with an arbitrary linear function. Then it will be calculating the overall error rate when trying to calculate on the given data. By manipulating the different factors in the function, based on a set learning rate, and recalculating the new error rate for the modified function. Then selecting the best fitted (lowest error rate) and repeat the process for a pre-set number of epochs (numbers of iterations), or an optimal function has been found. For more information and extracted code see the Appendix C.

Linear regression was the chosen regression as it gives the most comprehensive output for any user to understand. A logistic or polynomial regression might fit the data better but would be harder for the user to extract meaningful information out of it.

2.2.4 Data Storage

As this project will collect data from its user it is necessary to store this information somewhere, both accessible and securely. The ideal method would be to allow for storing the data both locally and remotely, allowing for use of the app even when not connected to the internet or problems with the remote storage server.

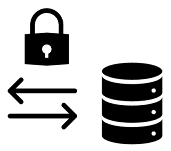


Figure 2.7: Secure Data Storage

For the local storing the data locally, an SQLite database [8] is sufficient to be stored on the device. It is the default database for Android development and is sufficiently documentation for what was needed (No need for any fancy querying, as only one user is planned for on each device). Local storage relies on the security of the device and the data would not be accessible from anywhere except for the used phone.

Then as the app matures there will be a push to set up a remote database. This will remove the limitation of one unique user per phone and accessing a user on multiple devices. Azure [9] and Firebase [10] was the main remote storage services researched. From this research Firebase was selected, as this has more support and documentation for Android development. The IDE used also has inbuilt features to assist connecting Firebase to the project.

2.2.5 Storage Structure

To structure the data is important to keep the data accessible to read and write. Local storage does not need to much structure, as SQLite takes care of each data class and stores it in various independent tables, as there is no relation between the different types of data except for the user. There is no need for any formal structure as by design can only host one users' information.

Remote storage in Firebase realtime database requires more structure, as each user's data is stored together with all the other user data. Firebase uses a JSON structure, a simplified structure of it can be seen in Figure 2.8. C An example of it with data can found in Appendix F.

```
--user
    --$USER-ID$
        --customNeed
            --$CUSTOM-QUESTION$
        --data
            --$INPUT-DATE$
                --$INPUT-ANSWERS$
        --userInfo
            --$INFORMATION-ANSWERS$
```

Figure 2.8: Structure in Firebase

2.2.6 Data visualisation

Explanation and simplification of the analysis needs to be communicated to the user, as they will most likely not understand the output of the analysis. Structuring and visualising this will allow for better understanding.

Figure 2.9: Visualise analysis

2.2.6.1 Line chart

One of the best methods of displaying the changes over time is by use of line charts. It can quickly communicate approximately how many data points are inputted and visually find similarities between two different charts.

The library identified to be the best method of plotting these charts is *MPAndroidChart* by PhilJay, as this seems to be the most user-friendly documentation and tutorials. Allowing for several different types of charts with extensive number of functionalities.

2.2.7 Natural Language

A user-friendly method of communicating the analysis is to transform it into a sentence. This can guide new users to quicker learn what the other visualisation method tries to convey and spell out for them what seems to have the most effect on their wants.

2.2.8 Simple List

The simplest method is to list out the data in a simple manner. If the user is already familiar with the statistical method used and layout of the activity, they can more quickly identify what feature and statistics they want to see.

2.2.9 Cohort Studies

By collecting all this data in one place there is an opportunity to learn more about the population and cohort within that population. Results that can lead various sectors to put entities in place or encourage people to do define actions resulting in an enhancement of a certain wanted traits. Examples could be for a company to encourage movement to enhance the happiness or productivity of their employees.

Figure 2.10: Cohort Studies

An administration tool that can access and download the data of users, that has opted in and allow their data to be shared. This mus be protected by needing to authenticate that data is not accessed by unauthorized people.

The data will be downloaded as two CSV (comma-separated values) files. One to contain additional given information about each permitted user, to allow for more specific cohort studies. The additional information will be method of grouping cohorts, gender, age, ethnicity, etc. The other containing all the data inputs of each user in file one.

There will not be any identification to identify the specific user, such as name or email address, only a unique ID to correlate between the two files.

Then to make the data accessible to the masses there will be a method of regularly posting cohort studies. There are a few methods identified, setting up and host a website with the data, a simple automated twitter bot that posts the study regularly to twitter (Similar to the Twitter account GWASbot [11]), or accessing through the application. The latter allowing easy access for the users of the app to compare themselves to similar people or other groups.

2.3 User Interface

A large section of any mobile application is the user interface. By having a logical method to manoeuvre between the different activities is important for the user to not become frustrated. A not optimal UI can make influence the success of keep the user coming back.

Finding a good colour combination is essential, as colours has been shown be an important role to influence our mood and perception of any situations [12]. Orange has been chosen as it has shown to give the individual a sense of excitement and enthusiasm [13, 14]. Encouraging the user to act on from the feedback they get to improve their life.

- **Login**: Figure 2.11 Start-up activity to log in users.
- **Register**: Figure 2.12 Register new users.
- **Main Hub**: Figure 2.13 Register new users.

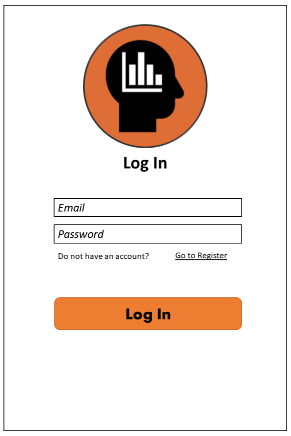


Figure 2.11: Log in

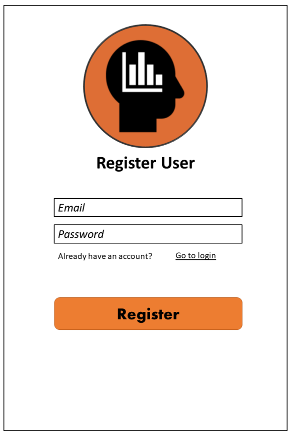


Figure 2.12: Register

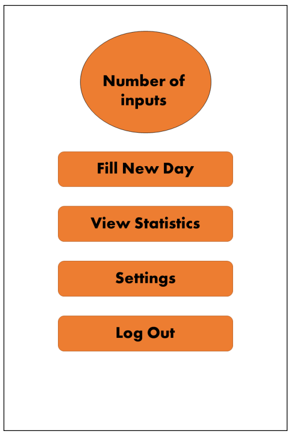


Figure 2.13: Main Hub

- **Input Wants**: Figure 2.14 Scale fulfilment of wants.
- **Input Wants**: Figure 2.15 Scale fulfilment of wants.
- **Statistics**: Figure 2.16 Present analysis of data visually.

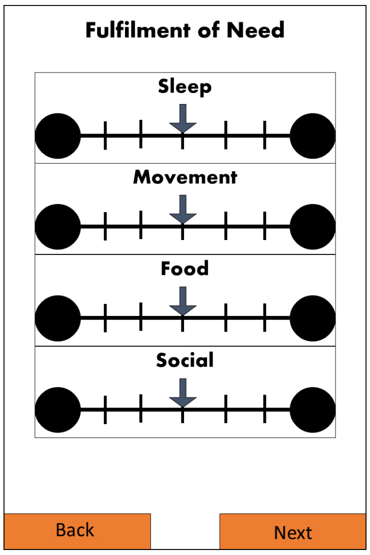


Figure 2.14: Rating want

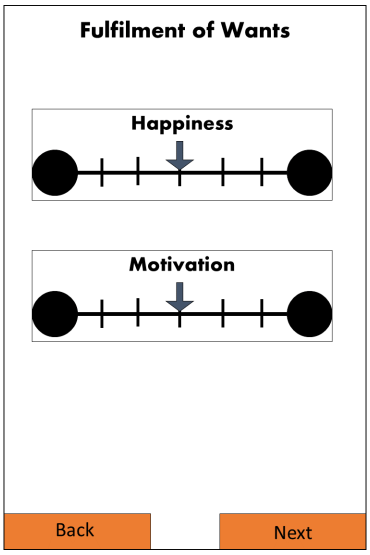


Figure 2.15: Rating need

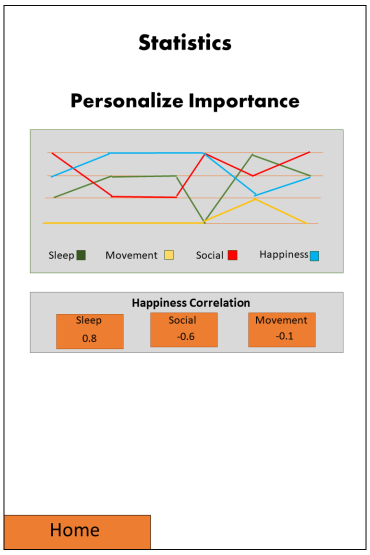


Figure 2.16: Statistics

- **Settings**: Figure 2.17 Hub for settings and more advanced use.
- **Custom Questions**: Figure 2.18 Allow to add custom wants and needs.
- **User Information**: Figure 2.19 View and edit information about the user.

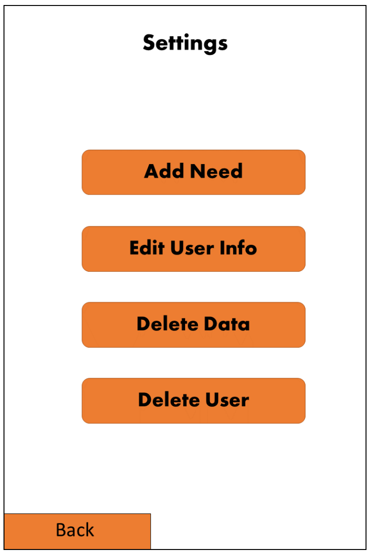


Figure 2.17: Settings

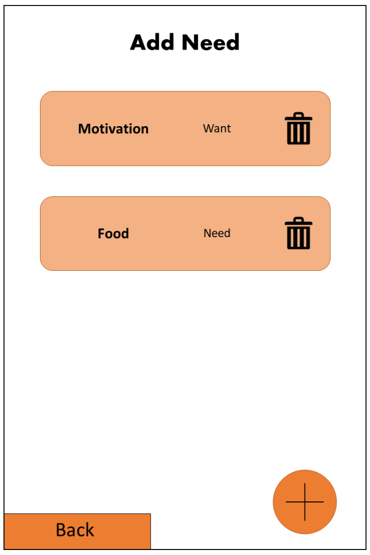


Figure 2.18: Add needs

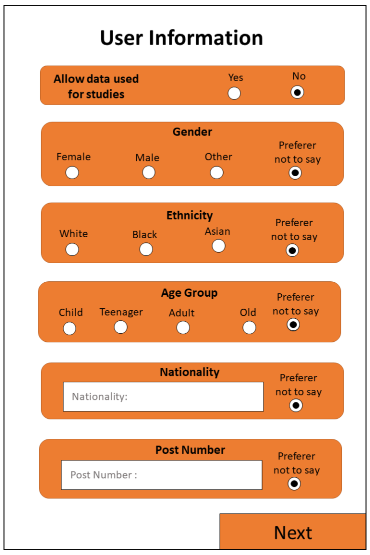


Figure 2.19: Edit user info

2.4 Implementation Tools

There will be used several tools throughout this project to produce the described software. This section will describe the tools used and considered.

2.4.1 Languages

Java has for a long time been the main go to language for programming android applications. But in recent years the new language Kotlin has been getting attraction as the successor of the established Java. Kotlin, version 1.4.30, chosen for its more modern syntax, more up-to-date documentation, and it is becoming the standard language for android development.

The administration program to download data of users to study do not require any UI and any complex mechanism, a simple text-based interface will be sufficient. Therefor Python3 is the chosen language, for its quick development and large backlog of useful libraries.

2.4.2 Development Environment

The most popular development environment when developing an android application is the IDE Android Studio [15], version 4.1.2. As this IDE is the most popular method to developer makes most documentation and tutorials fitted toward it.

The programs developed with python are on the smaller side of programs and do not require any more structure than one file, therefore the simple text editor Vim [16] is sufficient for this.

2.4.3 Build tool

For this project Gradle [17] was used to keep track of all the dependencies used the correct versions and automate the build the project. It was chosen as it was the default build tool for android development, therefor packages documentation assumed and focused their documentation on Gradle. Minimum SDK allowed is version 26, Android 8.0 (Oreo), as some functionality restricted this from being set lower.

2.5 GitHub

GitHub [18] was used for its version control and backing up the source code for the project. This made it more efficient to roll back to a previous version of the software in case of any problematic instances, such as a major bug or data corruption. GitHub also has the feature of access control, permitting access to only wanted users, useful to keep source code private during development. It can control access to any supervisors or other parties who want to keep track of the progress. It also gives the advantage of simultaneously working on the source code between multiple workstations.The Github can be accessed through this ULR: `https://github.com/RunarReve/MoodTrackingApp`

2.6 Custom additional Needs/Wants

To make the user experience more personalized and flexible to each user, the application will allow them to add their own wants and need. It will be stored separated from all the other but still accessible when required by the application.

When adding or removing a custom need/want, the system will be sturdy enough so that the analysis does not break. When it is analysing two questions with different number of inputs, it should ignore both data-points when one of them are missing. Meaning once a new custom need/want is added it is not statistically reliable, but will improve over time.

2.7 Authentication

Security is key part for avoiding data breaches and user getting access to data they are not permitted. Authorisation methods are provided by Firebase [10] which allows for sign in from multiple methods, Email & Password, Google, Facebook, etc. This project currently only focuses on Email & Password.

To prohibit access between users the database has set up ”rules” to restrict access on user’s data between users. These rules can be seen in figure 3.1.

2.8 Admin Storage Connection

For authorised personnel to download the users’ data in a structured format there is an additional python program to access and download data of users, who has allowed their data to be shared for external studies. This is done by using the tool Pyrebase4 [19] to be able to access Firebase [10] services through Python3.

2.9 Mid-Project Demo

About halfway the project was showcased for its mid-project demonstration to display the progression up until that point. The feedback from this touched upon several points that could be improve or implemented. The most significant inputs were to add more different tests, such as a larger focus on testing the UI, and to research into and include features from available sensors from the user’s device, for example a pedometer.

Chapter 3

Implementation

There are always obstacles encountered when implementing software. These might change the functionality of the software or take more time than expected. Here are some of the major obstacles that was unforeseen an changed the design.

The path to main software, the android application, will be: $Application/MoodTrack$. While the additional admin software's path: $Application/Need2Want\_AdminTool$. Both from the project repository.

3.1 Storage Method

In the original design for storing data was to have storage split between local (SQLite) and remote (Firebase) storage. This was to prevent the reliance on internet connection and allowing the user to choose when to allow data upload, for example only when connected to Wi-Fi.

This caused some design problems and security concerns. The design would become more complex as the application need to keep track of what both databases and make sure both were up-to-date simultaneously. Like the structural design it could make the user experience more frustrating, as the user might not easily understand or care for these advantages of multiple storage methods. In regard to security, authorising the user would be a significant problem, as the design would require storing the user's authentication or bypass this step when offline, both raising significant security concerns. There would also be no easy method to fully delete all the data ever collected on the user if they would like to delete all their data. As previous data could still be stored locally on other devices that the user had previously used.

Therefore, throughout the development this dual storage method was scraped and reliance on only the remote storage method became the only storage method for the final version. It will result in the user not being available to log their day when they are not connected to the internet. This will not cause any major restriction, as the expected use-case is for the user to use the app at the end of the day connected to Wi-Fi.

3.1.1 Database Connection Issues

The process of connecting an app to Firebase is well documented. But during implementation an issue arose. The project was designed to use the newly available Belgian server. This was chosen for its closer proximity to expected userbase and hosted in a more stable and trusted country with more stricter policies regarding data protection than the location default server, the US.

The problem arose from connecting the application to the server before setting up the realtime database, used for remote storage. As a generated setup JSON file would not contain information for the correct address of the non-default server. Resulting in error from attempting to connect to a non-existing address in the default server.

Once identified this it was a simple fix by regenerating the JSON file and replace it with the old file in the project then rebuilding the project.

3.2 Authentication

Controlling access so each user only has access to their own data is important for security, as well to avoid external parties accessing the data. Originally the method of hard coding the source code such that users was not able to request any data than their own. This is acceptable for local database, as access to others' data is impossible, but remote databases can be accessed through other medias. One could access the database is by reverse engineering the application and requesting all data from database.

Fortunately, Firebase has a method of controlling access at the server-side, called rules. This requires any query to the database to come with an authenticated token, then the token is checked against the rules to allow/deny the query. Viewed the rules in Figure 3.1.

There are several methods of declaring an administrator user. Firebase allows setting special token on users, to declare various features, such as permission. As this project is not designed to allow more than a handful administrators and each new admin needs to be manually evaluated. The simplest method is to update the rules with the new admins user ID and give them read permission on the top level, as seen in Figure 3.1 replacing *$$ADMINID$$* with admins ID.

```
{"rules": {
    //Admin user
    ".read":  "'$$ADMINID$$' == auth.uid",
    "user": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
} } } }
```

Figure 3.1: Firebase Realtime Database Rules

3.3 Cold Starts

One of the biggest problems with collecting and analysing data is to not have enough data to start off with. In the beginning the analysis will be heavily skewed towards the first few inputs, making the results not reliable and overly sensitive. As more inputs are given the less this is a problem, as the analysis get more data to understand the user.

That is why as a small additional project was to attempt to solve this problem. The path for this program is: $Application/SyntheticGenerator$ from the project repository. By creating a python program to generate a dataset of synthetic data, allowing users to use this synthetic data to add extra data to themselves and make the start-up less sensitive to the first few inputs.

It was originally generating the data-points at random and did not have any relation between the different features of wants and needs. It was improved to consider the relation between these features. By first generating the inputs for needs with a frequency of values based on standard deviation that allowed for randomness, a set centre-point of values then diminishing frequency the further deviating from set centre-point. Then the values for want can be calculated based on a linear formula between each need and want.

As the user register on the application, they can choose to include this generated synthetic data as their own input to have a less sensitive start analysis.

As a secondary usage of the synthetic data is to test if the application is able to pick up on the relations between wants and needs set in generated data. From analysing a set of generated data indicates that it is able to observe the relation intended, but because the data of its large amount of randomness it is not perfect. The generator could benefit from more development.

3.4 Sensor Data Integration Issues

From feedback from mid-project demonstration discussed that implementing device sensors into the device could provide a more richer user experience.

The first method attempted was reading the default pedometer of the user's phone. This method appeared to be problematic and unstable. It would not be accurate and would only have an accumulative counter since the device last bootup. Resulting in no easy method of getting an accurate read of the number of steps each day.

The second method was using the API Google Fit, which allows access to the user's daily step count. This is more stable than the first method and separates the steps from each day, and do not restarts counter upon bootup.

But Google Fit requires more than expected setup and needed to be connect to a specific google account. From research this would not be manageable, with the time allocated. As this was not being a feature originally designed for this had to be scraped to focus on more imminent features. It was reduced to a simple editable text-box with a button to retrieve number of steps from first method (accumulated steps since bootup) or by edit/input their own step manually. This can be found when entering a new input of the day, but can in generally be ignored as it is mainly a placeholder for further implementation of pedometer.

3.5 UI Implementation

Implementation of the user interface for the mobile application went as expected. By first setting up a manoeuvrable skeleton app allowed for easy separation the activities and a visual indicator to what needed to be done and in what order. This allowed for more focused and streamlined implementation for each feature.

Library used to make the application more flexible and efficient was Recycleview and its widget of same name. It allows for flexible lists, like the survey list of wants or needs, more efficiently. This allows the user to add more custom questions without reducing performance too much.

The next section, Section 3.5.1, contains screenshot of the finished product, created based on the design described in Section 2.3.

3.5.1 UI Screenshot

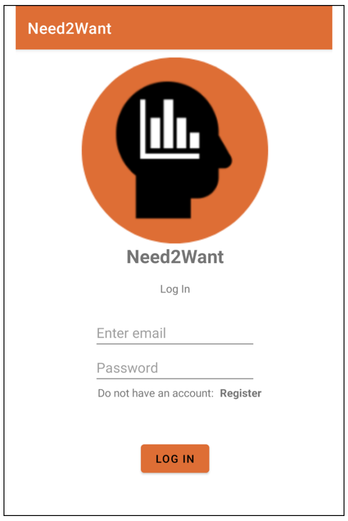


Figure 3.2: Log in

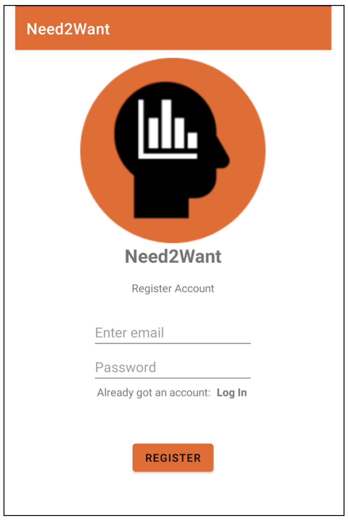


Figure 3.3: Register

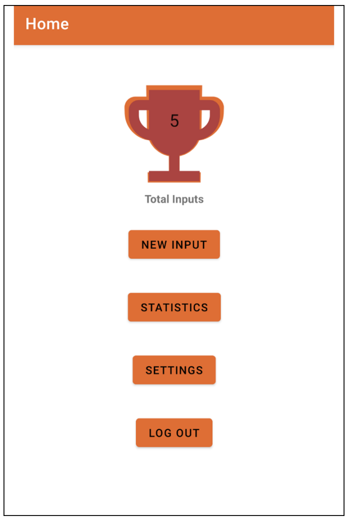


Figure 3.4: Main Hub

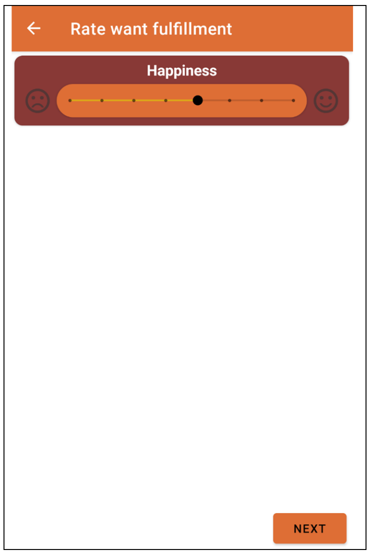


Figure 3.5: Rating want

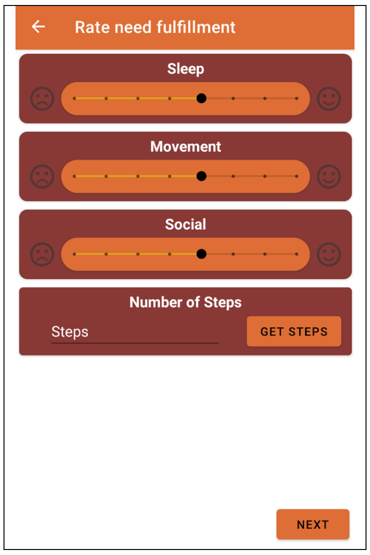
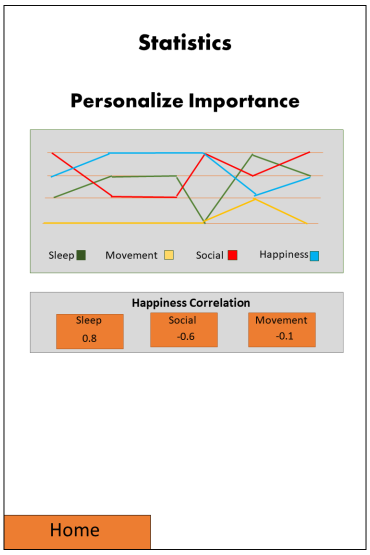


Figure 3.6: Rating need



Happiness Correlation

Sleep	Social	Movement
0.8	-0.6	-0.1

Figure 3.7: Statistics

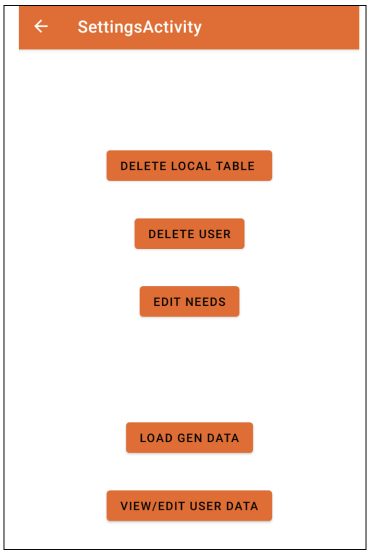
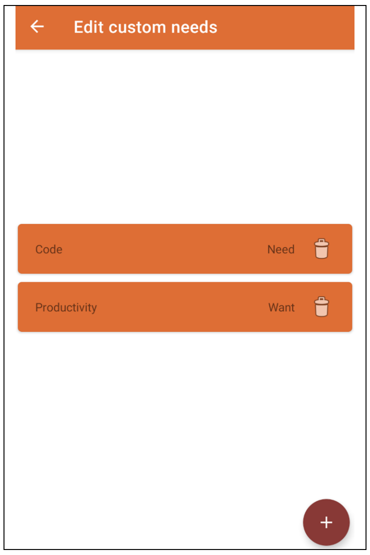
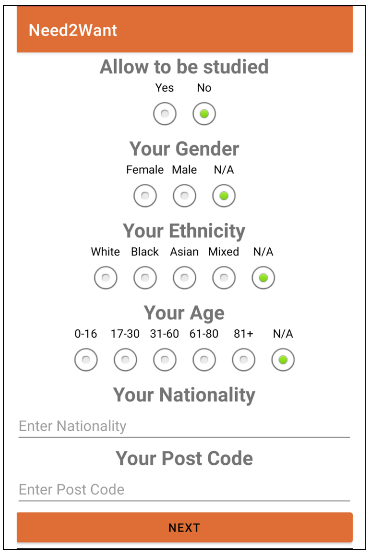


Figure 3.8: Settings



Code	Need
Productivity	Want

Figure 3.9: Add needs



Allow to be studied: Yes [✓ No]

Your Gender: Female Male [✓ N/A]

Your Ethnicity: White Black Asian Mixed [✓ N/A]

Your Age: 0-16 17-30 31-60 61-80 81+ [✓ N/A]

Your Nationality: Enter Nationality

Your Post Code: Enter Post Code

Figure 3.10: Edit user info

Chapter 4

Testing

4.1 Overall Approach to Testing

The approach to testing throughout this project was mixed, as not all features can be easily tested separately from the larger system. But when possible, if it had predictable input and output, then these features could be developed with a more TDD approach. This made the development quicker as multiple test-cases could be tested automatically, tuning the algorithms until they fit all test cases. It also made it more comprehensive to find bugs, improve, and refactor it, as every change could easily check if it broke any functionality.

When the application was manoeuvrable, UI tests was created to make sure all functionality worked as expected. Testing for buttons taking the user to the right activities, feature worked as it should, and identify if there are any instances where the app crashes.

These methods of testing make it quick and reliably be sure when working software is functional that it stays functional.

4.2 Test Devices

Phone	Resolution	Android Version
Pixel 2 (Emulated)	1920x1080 (441 ppi)	9 (Pie)
Huawei P20 (Physical)	2244x1080 (428 ppi)	8.1 (Oreo)
Huawei P20 (Physical)	2244x1080 (428 ppi)	9 (Pie)

Table 4.1: Devices used for testing

4.3 Automated Testing

Automated tests are a reliable method of testing functionality without the input of a tester. By automatically running multiple tests it can easily identify if something breaks or changes unexpectedly.

The automated tests can be accessed from the project repository: $Application/MoodTrack/app/src/test/java$ (Unit & Integrating tests) and $Application/MoodTrack/app/src/androidTest/java/com/rever/moodtrack$ (UI tests). In these locations there is a suit to automatically run all the different separated tests.

4.3.1 Unit Tests

Unit tests were used extensively where features could easily be extracted out from the larger system and have predictable inputs and outputs. This led to the main bulk of the unit test focused on the app's correlation features, such as Spearman and Pearson Correlation. The testing framework used was Junit4 [20].

By isolating each step of these features allowed for more easily make sure each step performed as it should and identify where the method was faulty. It allowed for not requiring fitting the feature into the larger system before knowing if it was working. Saving time with evaluating if the algorithm was correct, as one did not need to boot up the application.

Similarly testing was done for testing the linear regression feature. But as this is a learning method and the outcome is not always expected to be a single ideal output based on the input, as it is heavily based on the parameters used. This was solved by making the assertions when testing less strict, allowing for a small range of deviancy around the expected ideal output.

These test for regression allows a quick method to tweak the parameters and find the best fit. Testing what specific learning rates and numbers of epochs would give the best result in a respectable time. Additionally, refactoring and improvement on the method was made quicker to identify the change in results.

4.3.2 Integration Testing

As the unit tests above only test with the raw ideal inputs (arrays of integers), there is a need for tests similar to how it is expected to be when connected into the larger system. Testing how it handles expected data and data with no input, missing inputs, or other problematic instances. By simulating expected inputs allows for a fully functional and complete feature before attaching it to the larger system, confirming that it will not break the larger system when it is connected. If something breaks in the system later, these integration test can narrow the search significantly to find the bug, by identifying if there is a problem with the functionality or in the interactable sections of the application.

4.3.3 User Interface Testing

As this was an android application project there needs to be a large focus on testing the interface. Making sure that all the buttons led to the expected activities, the functionality was as it should, and the app not unexpectedly breaking. The tool for testing the UI was the espresso testing framework [21], developed by Google.

The application is based on different users creating accounts and signing in. Therefore, the general design for each test is to create a unique user for each test case. Ones the test case is complete it will delete the user. This ensure that each test is working independent from each other. But with this structure there can occur a problem if a test fails and do not properly delete its user, leading the test users needing to be manually deleted.

Espresso is often a bit eager to proceed with the tests and do not really want to wait until all data has been retrieved from a database. Espresso has method to avoid this, called Espresso Idling Resource, which sets up the code such that it can keep track of all processes ongoing and waits until jobs are pending. But this required much restructuring of the code, and therefor was replaced with a primitive ”Wait for X milliseconds”. It is not the most efficient method as it slows down testing, but is sufficient for the expected number of tests needed.

The espresso framework can be limited and struggles to test more advanced UI. This seems to be a problem when using a rank-bar inside a RecycleView, mainly used when user is answering their wants and needs for the day. Most likely this is caused by espresso not easily locating and interacting with features that may vary the interface with more complex widgets. Therefor it was concluded that the easiest way of testing was with some manual tests.

4.4 Manual Testing

As described earlier, there is instances where automatic tests are not easily possible. The focus on these tests is to test that rank-bar in RecycleView is intractable and processed by the application, when filling out the forum of questions.

Additionally, there is test to document that the administration tool is correctly connecting to Firebase server. The tests created is designed to be able during runtime. As the admin tool relies on data input from the android app, it requires the tester to simultaneously use both programs during testing. Increasing the complexity of the test significantly.

The procedure for the manual tests can be viewed in the Appendix D.

4.5 Generated Data Test

An informal method of testing how the application on ”real” data, prior to user tests, is to use the synthetically generated data used to tackle the cold starts, described in Section 3.3. It can test if the app can detect the correct correlation from relations set in generating data. This is only an informal method, as the generator is highly random and noisy, only used as an indicator for proof of concept.

Chapter 5

Critical Evaluation

5.1 Original Objective

This project was initially vague, main requirement was to be a mobile application regarding mood or well-being. This gave the freedom and liberty to develop and lead the project to my own ideas. Therefor the extended objective was set.

Close to all the requirements set was met. But as some of these was more optimistic and would require more time and resources than expected was not completed to the degree initially planned for. Nonetheless, there were also areas that exceeded expectations. Where the learning more allowed for expansion and or implementation of features.

Some of the examples given as background (Section 1.1) could be described differently. As the apps used for inspiration was more fitness related and the ideas of more personalization would be difficult for these. Even though these examples are not fully equal, it helps to convey the idea of a more personal user experience.

5.2 Preparation and Research

The first few weeks of starting this project was spent planning what was needed to solve the set objective and researching what was needed. As it has been a few years since developing an android application and never used Kotlin, this was the focus for the first few weeks. By running several tutorials to get the basics of the framework and language. As I got the handle of the basics, I could move on to the more advanced functionalities that was needed.

As the project developed and I got a more understanding of languages I discovered new functionalities to add that would improve the project. I also learnt better practises and methods of improving it, if the output would outweigh the developing time cost.

5.3 Time Management

The time management during this project was close to optimal regarding my original plan. During planning I set several key milestones set to specific dates and most of these was completed on time or ahead of schedule. An example to this was the milestone of Mid-Project-Demo where I planned for having the basic of the functionalities of the app working and presentable, but little in regards of security.

The success of the time management I contribute to the planning stage, as I managed to split the project into manageable milestones (Appendix E). This made it easier to keep discipline as I had intermediate goals to work towards. The second was to utilize my blog/diary as much as possible. By regularly writing what has been done made me see progress, even in weeks that felt slow and less productive. If there was no progression on the functionality of the app, it often was because the time was spent researching and learning or doing important refactoring. I also kept as much as possible centralised, such as having the Kanban board in the same document as the blog/diary. This allowed for less documents and websites to keep track of, and I could easily check what features I could do next after writing about the latest feature.

5.4 Testing

The testing strategy used throughout the development was to easily verify if an algorithm worked as expected, for example Spearman correlation, then add more tests to try to break it. Testing also gave the ability to develop the feature independently from the larger system until it could easily be implemented.

I would like to add more test coverage, as the current tests are mainly testing the main features. There are still some more unusual instances that could require some more tests. Similar each test is only tested in optimal physical situations and having tests that simulated unexpected situation could help improve the quality coverage, for instance to tests for device losing connection to the internet.

Another method of improving the testing would be to increase the diversity of phones, screen seizes, and android versions used for testing. As all the tests was run on an emulated phone: Pixel 2, and my personal phone: Huawei P20, both with a somewhat new android version.

In my opinion the current test is close to acceptable for the size of the project. It covers most of the interactions a user will have with the application. But there is always room for improvement and additional tests.

5.5 Hindsight is 20/20

There are several areas one would have approached differently at the end of a project. For me this was no different.

The major difference if I had to do this project again would be to take a module or class that formally trained me for android development. I do believe that my attempt was well executed, but there are several areas my knowledge is lacking, due to the only learning what was needed. For instance, an area that I skipped was on fragments, as it is not ideal to create your whole application with only activities. As it would take a significant time to refactor the whole project into using fragments, that would have minimal user notice, and would most certainly break some of the functionality. Therefore, some of the less ideal practises was not refactor, as the benefits would not outweigh the time required.

5.6 Future work

There are a few areas that throughout implementation there was identified several features that could enhance the user experience (such as communication with generated sentences) showed to require more time and resources than was available.

Analysing and publishing the cohort data would require setting up a dedicated server or device to regularly analyse the available data collected. As with the current system requires a device to download all the needed data to analyse. A dedicated server could calculate these analyses the data on all users to then publish the data, reducing the number of downloads needed and processing in the app. Putting a script on this server to download and analyse the data, then automatically publish it to a website, twitter bot, or back to Firebase, allowing the application to display it for users using the app.

The second major thing that would be ideal for future work would be to properly implement a method of reading the users step counter. With some more time and research the issue described in Section 3.4, a more sophisticated method could be added, either through the Google Fit API, or the less ideal method of calculating step based on $TotalStep - LastCalculatedStep$.

Additional feature of push notification could help and remind the user to fill out the survey at the end of the day. Allowing them to set if they would like and when to receive the notification.

As discussed in Section 3.3 about synthetic data generator to tackle cold starts. The generator is working as intended, producing data-set with correlation in the data, but there is room for improvement. It could be made more advanced and flexible, for instance having relation between each data-point of same need, imitating more of a flow in timeline, rather than each data-point being independent. Integration into the application as a feature could be possible, allowing each user to change various factors of the generated data (number of data point, relation, etc.) to better fit themselves and what they desire.

5.7 Evaluation Summary

In my opinion I believe that work done for this project is adequate regarding my prior knowledge and the time given to complete it. There are always areas that could have been improved and made more efficient. From the original plan put forth there were some features that might have been a bit too optimistic, but even though these features were not fully implemented I managed to put in place methods that replace it and that could evolve if given more time.

Annotated Bibliography

[1] V. L. Buijs, B. F. Jeronimus, G. M. Lodder, N. Steverink, and P. de Jonge, “Social needs and happiness: A life course perspective,” *Journal of Happiness Studies*, pp. 1–26, 2020.

Article explaining how well-being is linked with sufficient amount of social activity.

[2] A. Steptoe, “Happiness and health,” *Annual review of public health*, vol. 40, pp. 339–359, 2019.

Article explaining relations between happiness and health.

[3] J. Holt-Lunstad, T. B. Smith, M. Baker, T. Harris, and D. Stephenson, “Loneliness and social isolation as risk factors for mortality: a meta-analytic review,” *Perspectives on psychological science*, vol. 10, no. 2, pp. 227–237, 2015.

Article describing similarities between loneliness and a higher mortality.

[4] Samsung Electronics Co., Ltd., “Samsung health [mobile app],” April 2015, accessed February 2021.

A health application used for inspiration and finding features to improve upon.

[5] Huawei Internet Service, “Huawei health [mobile app],” March 2017, accessed February 2021.

A health application used for inspiration and finding features to improve upon.

[6] I.-M. Lee, E. J. Shiroma, M. Kamada, D. R. Bassett, C. E. Matthews, and J. E. Buring, “Association of Step Volume and Intensity With All-Cause Mortality in Older Women,” *JAMA Internal Medicine*, vol. 179, no. 8, pp. 1105–1112, 2019. [Online]. Available: https://doi.org/10.1001/jamainternmed.2019.0899

Study touching on the commonly followed arbitrary daily step goal of 10.000 steps do not have any scientific basis (comes from a marketing strategy by Japanese pedometer manufacturer, as the symbol for 10.000: 万 is like a running figure).

[7] John Hammersley and John Lees-Miller, “Overleaf,” https://www.overleaf.com/, 2012.

Cloud based LaTeX [24] editor. Released under the AGPLv3 License.

[8] Dwayne Richard Hipp, "SQLite," https://www.sqlite.org/index.html, 2000.

Relational database, default database for Android development used to save data locally on device. Released under Public domain.

[9] Microsoft, "Azure," https://azure.microsoft.com/, 2008.

Cloud service for computing and storage. SDK client released as Open source.

[10] Google and Firebase Inc, "Firebase," https://firebase.google.com/, 2012.

Platform/service for creating mobile and web applications, used for its free remote storage and authorisation methods.

[11] A. Ganna, "Twitter bot: Gwasbot," https://twitter.com/SbotGwa, July 2018, accessed February 2021.

Example of method of regularly publishing analysis through a Twitter bot.

[12] B. J. Babin, D. M. Hardesty, and T. A. Suter, "Color and shopping intentions: The intervening effect of price fairness and perceived affect," *Journal of business research*, vol. 56, no. 7, pp. 541–551, 2003.

Article on colours and their effect on consumer behaviours.

[13] Kendra Cherry, "Color psychology: Does it affect how you feel?" www.verywellmind.com/color-psychology-2795824, 05 2020, last accessed March 2021.

Blog post giving an overview what various colours makes us feel.

[14] ——, "The color psychology of orange," www.verywellmind.com/the-color-psychology-of-orange-2795818, 10 2019, last accessed March 2021.

Blog post on the psychology of orange and its relation to happiness, enthusiasm, and more.

[15] Google and JetBrains, "Android Studio Version 4.1," https://developer.android.com, 2014.

The official development environment for Google's Android operating system.

[16] Bram Moolenaar, "Vim," https://www.vim.org/, 1991.

Texted editor. Used for smaller programs and scripts. Released under the Free-software license.

[17] Hans Dockter, Adam Murdoch, Szczepan Faber, Peter Niederwieser, L. andRene Gröschke, and Daz DeBoer, "Gradle," https://gradle.org/, 2007.

Automated build tool. Released under the Apache License 2.0.

[18] Chris Wanstrath, P. andTom Preston-Werner, and Scott Chacon, "GitHub," https://github.com/, 2008.

Online Git [23] repository used version control service and storage of source code.

[19] Nick Horvath (AKA nhorvath), "Pyrebase4, Version 4.4.3," https://pypi.org/project/Pyrebase4/, 2020.

Python tool to easily access Firebase [10] via on Python3. It is available through pip or GitHub. Released under MIT License.

[20] Kent Beck, Erich Gamma, David Saff, and Kris Vasudevan, "JUnit 4," https://github.com/junit-team/junit4, 2014.

Unit Testing Framework for Java. Released under the Eclipse Public License 2.0.

[21] Google, "Espresso," https://developer.android.com/training/testing/espresso.

User interface testing framework. Released under Creative Commons Attribution 2.5.

[22] Lucid Software Inc., "Lucidchart," https://www.lucidchart.com, 2008.

Web-based chard and diagram creation platform.

[23] Linus Torvalds and Junio C Hamano, "GitHub," https://git-scm.com/, 2005.

Software for version control. Released under GPLv2, LGPLv2.1, and other licenses.

[24] Leslie Lamport, "LaTeX," https://www.latex-project.org//, 1984.

Software system for document preparation. Released under LaTeX Project Public License (LPPL).

[25] Philipp Jahoda, "PhilJay: MPAndroidChart v3.1.0," https://github.com/PhilJay/MPAndroidChart, 2014.

Library to easily make charts in Java and Kotlin. Mainly used for its line graph. Released under Apache License, Version 2.0.

Appendices

The following appendices are not directly needed to get a understanding of the project. It is an location where sections can be expanded and give additional insight into less necessary information.

Appendix A

Third-Party Code and Libraries

This appendix is to list the parts that was not created for this project. It will exclude the most common, e.g. *sys* for python, as these can be considered default for any program.

RecyclerView: Library that adds the RecyclerView widget that handles recyclable lists, useful to make list of unknown lengths viewable and efficient (only loading in the items that is on the screen). It is part of the Android Jetpack suit of libraries.

Cardview: adds card widget for UI. Version 1.0.0

Navigation: Makes navigation in the app easier. Version used 2.3.3

Room: Needed to use SQL Lite. Version used 2.2.6

Lifecycle: Perform actions based on changes in lifecycle. Version used 2.2.0

MPAndroidChart [25]: Library to easily make various charts for android applications with Java and Kotlin. Mainly used for its line graph. Version used 3.1.0

Firebase [10]: Needed to access features of Firebase, mainly for its realtime database and authentication. Version used 26.8.0

JUnit [20]: Framework for testing, used for unit and integration tests. Version used 4.13

Truth: Makes the syntax for testing easier. Version used 1.0.1

Espresso [21]: Framework for testing UI. Version used 3.3.0

numpy: Python library to include more mathematical functions, used for its random number generator/selector

json: Python library used to interact with JSON structured data more easily.

Pyrebase4 [19] - Python tool to easily access firebase via Python3. Version used 4 4.4.3

Appendix B

Ethics Submission

The ethics form is copied as text.

Reference number: **19187**

10/03/2021

For your information, please find below a copy of your recently completed online ethics assessment

Next steps

Please refer to the email accompanying this attachment for details on the correct ethical approval route for this project. You should also review the content below for any ethical issues which have been flagged for your attention

Staff research - if you have completed this assessment for a grant application, you are not required to obtain approval until you have received confirmation that the grant has been awarded.

Please remember that collection must not commence until approval has been confirmed.

In case of any further queries, please visit www.aber.ac.uk/ethics or contact ethics@aber.ac.uk quoting reference number 19187.

Assesment Details

AU Status

Undergraduate or PG Taught

Your aber.ac.uk email address

rur7@aber.ac.uk

Full Name

Runar Reve

Please enter the name of the person responsible for reviewing your assessment.

Neil Taylor

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment

nst@aber.as.uk

Supervisor or Institute Director of Research Departme

cs

Module code (Only enter if you have been asked to do so)

CS39440

Proposed Study Title

Mood Monitoring App: Personalized Prediction For Essential Fulfillment

Proposed Start Date

25 January 2021

Proposed Completion Date

1 June 2021

Are you conducting a quantitative or qualitative research project?

Mixed Methods

Does your research require external ethical approval under the Health Research Authority?

No

Does your research involve animals?

No

Does your research involve human participants?

Yes

Are you completing this form for your own research?

Yes

Does your research involve human participants?

Yes

Institute

IMPACS

Please provide a brief summary of your project (150 word max)

Project is about producing an android application that allows the user to rate their fulfillment of wants (e.g. happiness, productivity) and needs (e.g. sleeping, social) their day. Then the app will process the input and previous inputs to predict with statistics and machine learning how each need correlated to each want. This can thereby give the user and indication what need to put more effort into to feel more fulfilled in their wants. In addition to the personal prediction the user can opt-in to be a part of a cohort study, where their input will be allowed to be studied, like the personal study. This can identify correlations in a cohort of users. Allowing the users to see their differences to the larger population.

I can confirm that the study does not involve vulnerable participants including participants under the age of 18, those with learning/communication or associated difficulties or those that are otherwise unable to provide informed consent?

Yes

I can confirm that the participants will not be asked to take part in the study without their consent or knowledge at the time and participants will be fully informed of the purpose of the research (including what data will be gathered and how it shall be used during and after the study). Participants will also be given time to consider whether they wish to take part in the study and be given the right to withdraw at any given time.

Yes **I can confirm that there is no risk that the nature of the research topic might lead to disclosures from the participant concerning their own involvement in illegal activities or other activities that represent a risk to themselves or others (e.g. sexual activity, drug use or professional misconduct).**

Yes

I can confirm that the study will not induce stress, anxiety, lead to humiliation or cause harm or any other negative consequences beyond the risks encountered in the participant's day-to-day lives.

Yes

Please include any further relevant information for this section here:

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?

Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?

Yes

Does the research pose more than minimal and predictable risk to the researcher?

Not applicable

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?

No

Please include any further relevant information for this section here:

My project is focused on the application, and not any data collected

Is your research study related to COVID-19?

No

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.

Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.

Yes

Please include any further relevant information for this section here:

Appendix C

Code Examples

This section contains code extract that is not needed to understand the larger project, but it can give a bit more insight into what is being discussed in the report.

3.1 Linear Regression

Linear regression is a supervised machine learning algorithm that predicts the best fitting function for two lists. Linear regression assumes that the relation between the two lists is linear and will repeatedly change the function until it cannot find any more optimal function, or surpass a pre-set iterations, often called epochs.

```
//Linear regression to get function "f(x)=m*x + n"
fun regression(yList: List<Double>, xList:List<Double>)
:List<Double>{
    val learningRate = 0.001 //Tuning of learning rate
    val epochs = 10_000      //Number of iterations
    var m = 0.0 //Start position of tilt
    var n = 0.0 //Start position of start

    //Find avrg deviancy between prediction and given list
    fun averageDeviancy(predictionList: List<Double>)
    :Double{
        var tot = 0.0
        (0..predictionList.size-1).forEach {
            tot += (predictionList[it] - yList[it]).pow(2)
        }
        //Returns avrg deviancy of function
        return sqrt(tot)/predictionList.size
    }

    (0..epochs).forEach{
        //Calculate for all
```

```
            val calculationList = mutableListOf<List<Double>>()
            \\List of all possible changes of current function
            val listOfChanges =  listOf(
                listOf(m,n), //Default
                //Change only one variable
                listOf(m+learningRate,n),
                listOf(m-learningRate,n),
                listOf(m,n+learningRate),
                listOf(m,n-learningRate),
                //Change both variables
                listOf(m+learningRate,n+learningRate),
                listOf(m+learningRate,n-learningRate),
                listOf(m-learningRate,n-learningRate),
                listOf(m-learningRate,n+learningRate),
            )

            //-------CALCULATE-AVERAGE-DEVIANCY---------------
            listOfChanges.forEach { vars ->
                calculationList.add(
                    listOf(averageDeviancy(
                        xList.map {(vars[0] * it) + vars[1]}
                    ), vars[0], vars[1])
                )
            }

            //-------COMPARE-AND-SET-TO-BEST-DEVIANCY---------
            var best = 1000.0
            calculationList.forEach{
                if(it[0] < best){ //If the best, do a change
                    best = it[0]
                    m    = it[1]
                    n    = it[2]
                }
            }
            //If current is optimal, return it
            if(best == calculationList[0][0])
                return listOf(m,n)
        }
        return listOf(m,n)
    }
```

Appendix D

Manual Testing

Document for manually test parts not easily possible to test automatically. Admin account will be replaced after the 30th June 2021 to protect from unwanted future data breaches.

Test ID	**Description**	**Steps**	**Inputs**	**Expected Result**	**Pass /Fail**
MT1	Android App Test rank bar and application gets correct input	1. Register user with given info 2. *New Input*. enter survey as given 4. View stat activity is as ”Expected Results” 5. To settings and delete user	**TEST ACCOUNT**: Email: *test@mt1.test* Password: *123456* All personal info as: N/A or similar **Forum input** Happiness: Max Sleep: Min Movement: Max	Statistic activity: *List of inputs* (last section) only containing: Happiness:7 Movement: 7 Sleep: 0 Social: 4	Pass
MT2	Admin Tool Test that admin tool can connect and download csv files from database	1. Log in with admin account 2. Check the two files has been created in current dir 3. Delete the files	**ADMIN ACCOUNT** Email: *admin@admin.com* Password: *654321*	Expected files: dataInfo.csv userInfo.csv	Pass

Table D.1: Manual tests, Part 1

Test ID	Description	Steps	Inputs	Expected Result	Pass /Fail
MT3	Admin Tool (Adm) Android App (App) Only download data that has user permitted to be shared	1.(App)Register test account 2. *New Input*, default settings 3.(Adm)Log into admin tool 4. Search *userInfo.csv* for *MT3TEST* 5. Expected 1. Result 6.(App)To settings, edit info, change to allow study, save 7.(Adm) Repeat 3. and 4. 8. Expected 2. Result 9. Delete files 10.(App) Settings and delete user	**TEST ACCOUNT** Email: *test@mt3.test* Password: *123456* Do not allow study, nationality: *MT3TEST* **ADMIN ACCOUNT** Email: *admin@admin.com* Password: *654321* *MT3TEST* used to identify test account	1.Result: No Match 2.Result: One Match	Pass

Table D.2: Manual tests, Part 2

Appendix E

Initial Plan

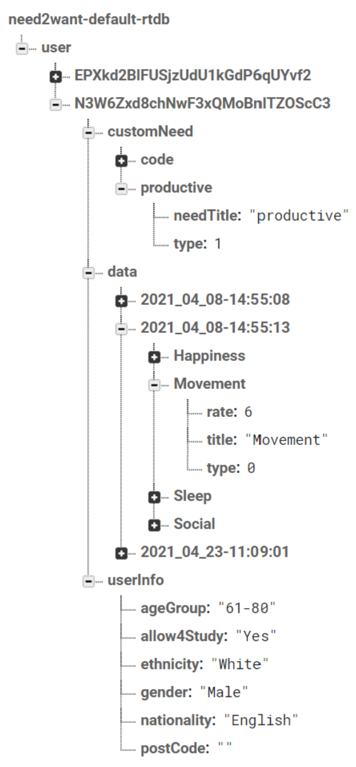
This section contains the brief plan that was used to plan for how long sections should take and when various features could be expected.

- **Week 1-2**: Research, set up project (GitHub, Diary, Repository), and planning (e.g. UI).
- **Week 3**: Put together skeleton app based on planned UI to allow for implementation of coming features.
- **Week 4-6**: Add main functionalities, surveying wants and needs and some basic statistics. A local database should be in place to store the data.
- **Week 7**: A functional app should be produced allowing for simple personal use. Ready to showcase for the Mid-Project Demo.
- **Week 8**: Implement new statistical methods.
- **Week 9-11**: Implement remote database and.
- **Week 11**: Adding authentication to application .
- **Week 12**: Method of analysing cohort data.
- **Week 13-14**: Fleshing out code and documentation, making everything ready for submission.

Appendix F

Firebase data example

An example of the Firebase realtime database is structured, filled with some example data.



```
need2want-default-rtdb
 - user
    + EPXkd2BIFUSjzUdU1kGdP6qUYvf2
    - N3W6Zxd8chNwF3xQMoBnITZOScC3
       - customNeed
          + code
          - productive
               needTitle: "productive"
               type: 1
       - data
          + 2021_04_08-14:55:08
          - 2021_04_08-14:55:13
             + Happiness
             - Movement
                  rate: 6
                  title: "Movement"
                  type: 0
             + Sleep
             + Social
          + 2021_04_23-11:09:01
       - userInfo
            ageGroup: "61-80"
            allow4Study: "Yes"
            ethnicity: "White"
            gender: "Male"
            nationality: "English"
            postCode: ""
```

Figure F.1: Example of how data is stored in Firebase