Department of Computer Science

Individual Project - CS3IP16

Hypothyroid symptoms & fitness tracker

Finley Williams

23013019

Supervisor: Lily Sun

04/06/2018

# 1. ABSTRACT

# 2. ACKNOWLEDGEMENTS

# CONTENTS

[1. ABSTRACT 2](#_Toc515849898)

[2. ACKNOWLEDGEMENTS 3](#_Toc515849899)

[CONTENTS 4](#_Toc515849900)

[3. GLOSSARY OF TERMS AND ABBREVIATIONS 7](#_Toc515849901)

[4. INTRODUCTION 8](#_Toc515849902)

[5. PROBLEMS ARTICULATION 9](#_Toc515849903)

[5.1 Situation as is. 9](#_Toc515849904)

[5.1.1 Situation A (Pen and Paper) 9](#_Toc515849905)

[5.1.2 Situation B (Non-specific apps) 9](#_Toc515849906)

[5.1 Problem statement and Justification. 10](#_Toc515849907)

[5.2 Key stakeholders and their concerns. 10](#_Toc515849908)

[5.2.1 Diagnosed hypothyroid patients 10](#_Toc515849909)

[5.2.2 Suspect hypothyroid patients 10](#_Toc515849910)

[5.4 Technical specification. 11](#_Toc515849911)

[5.4.1 Requirements 11](#_Toc515849912)

[5.4.2 Constraints 11](#_Toc515849913)

[5.5 Potential situations to be. 11](#_Toc515849914)

[5.5.1 Web-app 11](#_Toc515849915)

[5.5.2 Mobile app 12](#_Toc515849916)

[6. LITERATURE REVIEW 12](#_Toc515849917)

[6.1 Medical Background Research 13](#_Toc515849918)

[6.1.1 Symptom Tracking 13](#_Toc515849919)

[6.1.2 Hypothyroidism screening and symptom prediction. 14](#_Toc515849920)

[6.1.3 Barriers to entry for target audience? 15](#_Toc515849921)

[6.2 Existing solutions 16](#_Toc515849922)

[6.2.1 Butterfly [13] 16](#_Toc515849923)

[6.2.2 MySymptoms [14] 17](#_Toc515849924)

[6.2.3 Thyroid Assistant [15] 17](#_Toc515849925)

[6.3 Literature review conclusion 18](#_Toc515849926)

[7. THE SOLUTION APPROACH 19](#_Toc515849927)

[7.1 Platform decided upon 19](#_Toc515849928)

[7.2 Programming languages used 19](#_Toc515849929)

[7.3 Integrated development environment chosen 20](#_Toc515849930)

[7.4 Open source libraries used 20](#_Toc515849931)

[7.4.1 Bullyboo’s circle seekbar 20](#_Toc515849932)

[7.4.2 OpenCSV 20](#_Toc515849933)

[7.4.3 Android GraphView 20](#_Toc515849934)

[7.5 Level of prediction 21](#_Toc515849935)

[7.6 Development process 21](#_Toc515849936)

[7.7 Version control 21](#_Toc515849937)

[8. IMPLEMENTATION 21](#_Toc515849938)

[8.1 Design 21](#_Toc515849939)

[8.2 Tab page viewer 22](#_Toc515849940)

[8.2.1 Design 22](#_Toc515849941)

[8.2.2 Implementation 22](#_Toc515849942)

[8.3 Dashboard (Tab A) 24](#_Toc515849943)

[8.3.1 Initial Design 24](#_Toc515849944)

[8.3.2 Implemented Final Product 26](#_Toc515849945)

[8.3.3 Implementation 27](#_Toc515849946)

[8.4 Intakes (Tab B) Design 31](#_Toc515849947)

[8.4.1 Initial Design 32](#_Toc515849948)

[8.4.2 Refined Design 34](#_Toc515849949)

[8.4.3 Implemented Final Input Tab 35](#_Toc515849950)

[8.4.4 Implementation 36](#_Toc515849951)

[8.5 Labs (Tab C) Design 41](#_Toc515849952)

[8.5.1 Initial Design 41](#_Toc515849953)

[8.5.2 Implemented Final Product 42](#_Toc515849954)

[8.5.3 Implementation 43](#_Toc515849955)

[8.6 Imported third party library’s 47](#_Toc515849956)

[8.7 Layout XML. 47](#_Toc515849957)

[8.8 Android Manifest 47](#_Toc515849958)

[9. TESTING: VERIFICATION AND VALIDATION 48](#_Toc515849959)

[9.1 Test cases 48](#_Toc515849960)

[9.2.1 Tab A tests. 49](#_Toc515849961)

[9.2.2 Tab B tests. 49](#_Toc515849962)

[9.2.3 Tab C tests. 50](#_Toc515849963)

[9.2.4 General Tests. 51](#_Toc515849964)

[9.2 Requirement Validation. 52](#_Toc515849965)

[9.2.1 Requirement Validation. 52](#_Toc515849966)

[9.2.2 Constraint Validation. 53](#_Toc515849967)

[9.2.3 Project Initiation Document Validation. 53](#_Toc515849968)

[9.3 Limits to verification and validation. 55](#_Toc515849969)

[10. DISCUSSION: CONTRIBUTION AND REFLECTION. 55](#_Toc515849970)

[10.1 Test Results discussion and its implications 55](#_Toc515849971)

[10.2 Success analysis 55](#_Toc515849972)

[10.3 Project limitations 56](#_Toc515849973)

[10.4 Reflections 56](#_Toc515849974)

[11. SOCIAL, LEGAL, HEALTH, SAFETY AND ETHICAL ISSUES 56](#_Toc515849975)

[11.1 Social issues. 56](#_Toc515849976)

[11.2 Legal issues. 56](#_Toc515849977)

[11.3 Health and Safety issues. 56](#_Toc515849978)

[11.4 Ethical issues. 57](#_Toc515849979)

[12. CONCLUSION AND FURTHER IMPROVEMENTS 57](#_Toc515849980)

[12.1 Project objective and results conclusion 57](#_Toc515849981)

[12.2 Future work 57](#_Toc515849982)

[REFERENCES 57](#_Toc515849983)

[APPENDICES 59](#_Toc515849984)

[Appendix 1: Original Whiteboard Designs. 59](#_Toc515849985)

[Appendix 2: Project Initiation Document 60](#_Toc515849986)

[Appendix 3: Logbook 67](#_Toc515849987)

# 3. GLOSSARY OF TERMS AND ABBREVIATIONS

**Hypothyroidism** – The shortage of T3 and T4 thyroid hormone which causes the unwanted slowdown of the body’s function causing suffering and damage to the patient’s organs.

**Hyperthyroidism** – The overabundance of T3 and T4 thyroid hormone which causes the unwanted speedup of the body’s function causing suffering and damage to the patient’s organs.

**Subclinical Hypothyroidism** – Mild thyroid failure where TSH is mildly elevated but T3 and T4 are within normal ranges giving some symptoms but not a formal diagnosis of the disease and may not be treated as T3 and T4 are within the normal range.

**TSH –** Thyroid Stimulating Hormone. A hormone created by the pituitary gland to regulate the production of T4 and T3. Low levels of T4 cause a heighten level of TSH production whereas high levels of T4 cause a reduced level of TSH. Typical hypothyroidism has a low level of T4 which results in an extremely elevated level of TSH.

**T4 –** Thyroxine. (Four Iodine molecules) This is converted into triiodothyronine inside the cell when required. T4 is basically a stepping stone for T3.

**T3 –** Triiodothyronine. (Three Iodine molecules) The active form of the thyroid hormone which influences the bodies processes, the most important of which being the regulation of the body’s metabolism.

**Levothyroxine –** Levothyroxine Sodium. A synthetic drug that is identical biologically to T4 allowing the body to convert this into T3 for use within cells.

# 4. INTRODUCTION

By conservative estimates hypothyroidism affects two in one hundred, however many doctors believe the true rate is much higher than this, with the president of the International Hormone Society believing the true number lies between twenty percent and fifty percent of the population having some level of thyroid deficiency.

Hypothyroidism as a condition is characterised by the slowing of almost all bodily functions as the body is short on the hormones primarily used in relation to the metabolism. This “slowing” affects all parts of the body, from the brain to the digestive tract to the muscles causing memory issues and problems concentrating, digestive problems such as constipation, increased risk of other issues like Celiac disease, Crohn’s disease, Graves’ disease, frequent painful muscle cramps and decreased muscle strength.

During May 2017 the writer of this document was diagnosed with hypothyroidism after several months of suffering many of the symptoms. During the period after the diagnosis, there was an attempt to gather as much data about the symptoms as possible. To the writers surprise there was a distinct lack of apps available to aid in this pursuit which led to having to record everything on pen and paper, which was both impractical and not useful when it came to viewing trends and getting a picture of how the sufferer felt in relation to previous time periods.

When it came time to choose a project, inspired by the other apps available for different medical issues, the writer decided to attempt to create a app that could record the data on the issues encountered and present them to the user in a useful manner allowing conclusions and issues to be raised in a timely manner rather than waiting for the symptoms to get progressively worse.

Given this potentially huge section of the population who are either misdiagnosed or undiagnosed, having the ability to predict if thyroid issues are present via an easily installed app would be extremely useful.

A side benefit of having historical records of the symptoms is that due to memory issues associated with hypothyroidism having a record of what is felt greatly reduces the requirement to remember exactly what was felt on a day to day basis when discussing the issues with a doctor.

Given the above the overarching goal of this individual project is:

* To provide a method of easily recording the symptoms on a day to day basis.
* To reduce the memory load on the user by allowing easy reviewing of the data.
* To provide recommendations on what the users should be doing to improve their condition.
* To further reduce the memory load on users by giving reminders to user regarding taking their medication if the medication varies from day to day.

Within this document we will be covering the process from design and research to the final product.

# 5. PROBLEMS ARTICULATION

## 5.1 Situation as is.

The current problem for suffers of hypothyroidism is that if they wish to collect data on their condition there are extremely limited options which give more options than simply collecting the data.

Currently there are no solutions designed specifically with hypothyroidism in mind. This means that if they decide to record the data digitally, they may need to separate over multiple apps

The current situation regarding recording symptom, food and behavioural triggers for hypothyroidism is through one of two different methods.

### 5.1.1 Situation A (Pen and Paper)

The first situation for the recording of data is pen and paper. This involves simply carrying around a notebook or something similar and by writing down whenever anything is noticed that is felt may be significant.

This has several drawback related to its paper based nature:

* Losing or misplacing records is easy: misplacing the notebook could result in multiple months of data being lost, which could be difficult to replace and could majorly affect the long term tracking of the symptoms.
* Examining the data visually comparing symptoms against each other symptom or other triggering factors can be difficult. A graph would have to be drawn or the the data manually transfered into something like excel.
* There is little context to the data making consistency and quantifying the data very difficult. If a headache was recorded one day and a headache again the next day, it may be difficult to tell which was worse a month later.
* Another issue is that it may be a hassle to carry pen and paper around whereas most people constantly carrying their phones or laptop on them at all times.

In regards to pure data collection there is no issues with using pen and paper however whenever difficulties are likely to be encountered if anything more complicated than simply reviewing is to be done to the data.

There is one key advantage to pen and paper which is, the extremely low cost of pen and paper, A potential user is likely to have these in their house already, meaning recording can be started almost as soon as is practical.

### 5.1.2 Situation B (Non-specific apps)

The second situation is using any combination of non-thyroid specific apps. This could be using any number of “generic” medical apps. These can be useful although they are not designed specifically for hypothyroid suffers.

These are varied and each app has a different functionality so it’s very difficult to say what can do what, as none of them have been designed with hypothyroidism in mind it is clear that they may require additional work to make it work for a hypothyroid suffer and may need to use one or more apps to get the functionality the patient may want.

A final issue which generic apps have is if the user has been diagnosed as a sufferer of hypothyroidism it may be distressing or annoying to have various medical apps constantly tell them that they may be at risk of other diseases when they have already been diagnosed with hypothyroidism.

## 5.1 Problem statement and Justification.

Given the wide availability of technology, the best way of recording the data for hypothyroid suffers should not be using pen and paper. There should be a comprehensive digital solution available which allows the user to record their data for later use. Even showing just the basic information recorded chronologically when discussing the issues with a doctor would be an improvement as the current pen and paper solution means that the data will need to be manually ordered and cannot quickly jump to specific parts of the data which is of interest.

The current solution lacks the ability to do anything more than simply record data chronologically without the ability to go back and address previous data or do anything more useful with the data provided.

The ideal problem solution would contain the ability to include more data than just symptoms and provide the ability to visually examine the data compared to other data sets.

This gives the problem statement as:

“*There is no simple solution for tracking symptoms, consumed medication and specific food triggers for analysis and visual investigation/inspection.*”

## 5.2 Key stakeholders and their concerns.

### 5.2.1 Diagnosed hypothyroid patients

The first group of stakeholders is those currently diagnosed with hypothyroidism. This is the group this project will be aimed at.

* Their main concerns consist of accurately tracking their data over a long period of time to gain insights into their condition.
* All medical data must be kept confidential and not shared with third parties, known or unknown without explicit permission from the user.
* Assisting in the regular consumption of medication.
* The cost to use the system must be minimal as a high cost may deter the user from using the system in the first place - especially as some of the existing solutions such as pen and paper have such a low cost to entry.

### 5.2.2 Suspect hypothyroid patients

The second group of stakeholders are those who suspect they are diagnosed with hypothyroidism. This project is not being made for these people, however they may find a definite use for this product even if all functionality is not usable or specific to them.

Their concerns heavily mirror that of diagnosed hypothyroid patients.

* They wish to track their symptoms long term to monitor if they are a suspect hypothyroid suffer.
* All medical data must be kept confidential and not shared with third parties, known or unknown without explicit permission from the user.
* The cost to use the system must be minimal as a high cost may deter the user from using the system in the first place - especially as some of the existing solutions such as pen and paper have such a low cost to entry.
* The system must make accurate recommendations if it suspects that the user’s symptoms are atypical of a non-hypothyroid patient.

## 5.4 Technical specification.

### 5.4.1 Requirements

* The user must be able to complete their data entry for a specific symptom within 10-15 seconds with minimal thought/effort.
* The user must be able to complete the data entry for every available symptom within 2-3 minutes.
* The user must be able to go to a given time period and review their data (both symptoms and consumptions) for that given time period with context to the surrounding time period.
* The user must be able to compare at least two different data sets in context against each other by time to review how one effects the other.
* The data entry system must allow for interval data.
* Some basic advice/information should be given to the user regarding each symptom within the app.
* The system must notify the user and give them useful customizable information regarding their medication.
* The user should be able to monitor the long term blood test data in a separate context.

### 5.4.2 Constraints

* The system platform must allow for the financial situation of users wishing to make use of its features, and thus must take steps to ensure affordability and availability.
* The design must be sufficiently simple that those who are not entirely comfortable with technology are able to use it.
* Personal user data should not be given to a third party.
* Personal user data should be anonymised.
* Where possible, user data should be kept on device.

## 5.5 Potential situations to be.

### 5.5.1 Web-app

This is one potential solution to be. It would involve a web app type system where a user could log on using a web browser on which they could then enter all their issues, symptoms, food intakes etc.

They would then be able to view the data chronologically in the browser which would allow them to see how they are progressing.

Predictions could be made using the user’s data and this could then be conferred to the user.

The advantages of using a web browser is that it allows for a much greater level of flexibility it terms of user interface, however it is likely that the potential user will be using it on either a computer, tablet or even a phone which means that the UI needs to adapt to this in order to remain usable.

Finally, this solution will be using a login system so verification and authentication will be required to make sure that the user is who they say there are, so a password type system would be needed.

Also the user data will need to be stored safely, in compliance with the HIPAA and GDPR and other data protection laws.

A big advantage to this type of system would be that all the user data would be available to us as the user would already be storing their data on our service so using it for things like improving prediction models would be very easy.

### 5.5.2 Mobile app

Another potential solution would be create a mobile app for either android or iOS which would allow the user to enter all their issues such as symptoms, food intakes and other problems.

It would also allow the user to look back chronologically in app to see how they are doing on a day by day basis.

Basic predictions could be made about the users using their data which could encourage them to change their lifestyle choices.

Given the limited “real estate” on mobile devices an effort would need to be made to make to keep the UI simple and allow the user to make sense of what is on the screen.

Using a app for tracking could also allow the user to integrate with existing lifestyle management devices such as Fitbit to allow them to more easily and accurately track the amount of exercise they are doing or digital scales to automate the entry of their weight into the device.

The choice of device the app would be programmed on could be important however existing technology such as Xamarin [1] or React Native [2] can be used to allow developers to share code and design UIs that are the same on multiple devices allowing the product to be released on multiple devices simultaneously.

Keeping the data secure would be easier on a mobile device as the data can be retained in storage rather than sending them to our server which would allow the user to know that their data is secure. This does however have the drawback in that we cannot build a predictive model using user data. If it is decided that predictive machine learning type of model is required the app could be altered so that the user can choose to send their data to the company so that the model can be developed.

# 6. LITERATURE REVIEW

Within this literature review we will be discussing the existing research within the field of medical application and bio-informatics as well as the existing solutions to the problem defined within the problem statement. The strengths and weaknesses of each will be discussed and where possible the advantages of these will be incorporated into the design of this project.

## 6.1 Medical Background Research

### 6.1.1 Symptom Tracking

Within this section we will be discussing why people track their data and the issues that arise from tracking it.

Within [3] the reasons women track menstrual cycles is examined with an attempt to uncover design opportunities and to extend the understanding of personal informatics tools.

Whilst the menstrual cycle is not a “medical condition” but simply something half the population suffers from. It does however share almost all of the characteristics of hypothyroidism when considering the symptoms. None of the symptoms are binary and have a range of values as well as consistently reoccurring within the patient’s life.

Within [3] the reasons for their tracking of symptoms are: “*(1) be aware of how their body is doing, (2) understand their body's reactions to different phases of their cycle, (3) be prepared, (4) become pregnant, and (5) inform conversations with healthcare providers. Participants were typically motivated by multiple factors.”* [3]

The particularly relevant statements from this are being aware of how the user’s body is doing and informing conversations with the healthcare providers as, as previously mentioned within section 3. The reduction in mental load bearing is of particular relevance to hypothyroid suffers as this condition reduces the ability to remember things.

Interestingly, when deciding how to track their symptoms their first instinct is simply to look for an app with 47% [3] of all participants thinking “*Common sense, there had to be an app for it. There’s an app for everything.*” [3] The fact that there is no such app for hypothyroid suffers shows that there is a definite niche for an application of this nature even just for basic symptom tracking.

Another interesting point raised within this paper [3] is that if predictions are to be included accurate prediction are a must have as predictors will be rapidly abandoned if they are shown to be inaccurate. This will be expanded upon in 6.1.2.

Design of the app itself is important to the surveyed individuals as they say “*Why can't keeping track of my menstruation be a professional and organized task?”* [3]This shows that a clear and professional design is important for the usability and keeping people motivated to be using the application. For the long-term use of the app this is extremely important as retention of the user’s interest is vital; without long-term data entry the app itself is relatively useless as without long-term data very little can be done for the user other than as an aid to memory.

In regards to design from this specific paper [3] there is a strong emphasis on having the design be gender neutral. This will not be an issue in this project is there is no gender bias when making design decisions.

Another key point from [3] is that the process of data entry should be “*professional and organized*”. This should be taken into account when design the UI as a cluttered interface is difficult to use and would greatly increase the difficult for someone who is less technically competent. This should be taken into account as the typical age range for diagnosis hypothyroidism is between forty and fifty years old [4] with approximately 15-18% of women over sixty years old having hypothyroidism.

Discussed within [5] is an examination on how apps can be designed to support sufferers and aid their carers. It also allows us to address barriers to entry that different app designs may come across. The covered conditions are: cancer, diabetes, disability, mental health and wellness.

A point raised by [5] that is of particular interest is that “*Apps should be easy for the patient to use anywhere*.” [5] This is interesting as this desire for usability anywhere means that the complexity of what the user is attempting to do at most point should be limited. When discussing symptom tracking this means that the data entry for the user should be swift and easy otherwise the user may lose interest or simply not find the app useful due to the inability to use it when it’s practical to them.

A reason given within [5] for why patient track data is that it is “*Important to take a long term view and track trends*”. This is especially true for hypothyroid suffers as the condition itself takes several weeks if not months be bought under control so the retention and frequent use of the app is important. This means that effort should be made to increase the ease of which the app is used so that it doesn’t become a chore to use during the period in which there is not enough useable data for the app to be useful.

Overall, the usability and visual design of the app for tracking of symptoms is very important as without a professional, clear look the user is unlikely to use the app in the long term which is where the app is most relevant and has the most benefits to the user. From this the project should be designed with a focus on keeping the app simple and professional so that the user doesn’t feel like it’s a chore.

### 6.1.2 Hypothyroidism screening and symptom prediction.

The field of symptom and condition prediction is discussed within [3] in the context of accuracy and how the accuracy of a prediction can affect the long term use of a predictive app. “*Similar to results in other domains, women abandon inaccurate menstrual tracking apps and search for more accurate alternatives*”. This is important in the context of prediction as if the system makes inaccurate predictions this will just irritate the user and make them search for an alternative [6] to our system especially if the predictions are blatantly wrong and occur frequently.

Another issue raised by [3] is that the users who have atypical symptoms are likely to be missed by a system which is looking for the “average” symptoms of the condition so unless this is accounted for the system may miss many suffers.

A medical whitepaper [7] raises an extremely pertinent point in terms of screening for hypothyroidism in that when attempting to screen for hypothyroidism it should be considered that there is more than one root cause such as autoimmune thyroid disorder or that the patient’s thyroid simply ceased function. This complicates the process of predicating the onset of hypothyroidism as depending on the “type” of hypothyroidism the initial symptoms may wildly vary from a gentle onset to a sudden onset of all symptoms.

Another issue presented in [7] is that if the predicted hypothyroidism is subclinical this could result in an incorrect prediction by the system as the symptoms are present but as a blood test will show T3 and T4 within the normal ranges this will likely result in a non-diagnosis by the doctor, though this depends on the doctor as some endocrinologist believe that subclinical hypothyroidism should be treated [8] much like hypothyroidism. This brings several problems, as if a subclinical suffer is diagnosed by the system as hypothyroid they are likely to speak to a doctor and have blood tests taken, this would result in the prediction being classed as wrong even though they are displaying symptoms and have an elevated TSH. And, as mentioned earlier in this section the users is extremely unlikely to keep using an inaccurate prediction system.

Overall, the main issue in terms of symptoms prediction is that creating an accurate predictor is a huge challenge and if the predictor is inaccurate the user is likely to abandon the software meaning it is likely better to include no predictor at all than to include a shoddily made predictor in an effort to retain users and to build up data. It is possible that the large sets of user data could then be used in future for building models using machine learning methods [9] to create predictions however this brings with it several other problems such as patient confidentiality and the requirements of giving personal data to a third party which may not have a sufficient levels of security to stop anyone using the data. It is extremely unlikely that this level of model creation can be done on device as the large data sets and high computational requirements involved make it difficult to be done for just one person especially on a mobile device.

### 6.1.3 Barriers to entry for target audience?

An interesting point raised by [5] when discussing the barriers to entry to the use of medical apps is that “*The health apps may be too generic, and not relevant to individual patients.*” [5] This shows that whilst they may be seeking an app to their situation, they want the app to be designed for their specific condition rather than a multitude of conditions. It may be that they feel a generic app may be unable to properly support them or that it adds unnecessary additions and work to use as they have to wade through the “other” diseases and conditions features.

The availability of the platform as a barrier to entry is a huge issue as if we were to design a system for iOS device as only 42% [10] of people in the UK would be able to use the system and with only 15.6% [11] of people worldwide. This huge market share disparity shows that should we want the product to be available to a wider world wide audience or the financially challenged, we should aim to develop on a platform where the potential users can afford to access the platform. This could mean either developing an android app as these have huge market shares [12] of around 85.9% allowing practically anyone to use it, and as android devices are so cheap, anyone could buy an android app to use the software.

Alternatively we could develop a web app to be used on a computer or any internet capable mobile device which encompasses practically every device a user is likely to be using. A web app however comes with its own privacy concerns as the data would have to be stored by us and would require large amounts of data protection to keep the data safe.

## 6.2 Existing solutions

### 6.2.1 Butterfly [13]

This iOS app is currently in closed beta and contains many of the features which are aimed to be included within this the system to be.

Butterfly is what was envisioned when this project was decided on however several flaws make it difficult for this app to do the job in a way that is optimal.

There is a wide array of features from tracking thyroid and adrenal medication, medication reminders and monitors when the user’s adherence to the schedule. It also has “over 275” symptoms built in for tracking so that the user can track any number of symptoms they may have.

A good feature it has is keeping track of the user’s lab work all in one place so that they can see the changes over time from one place.

It also looks as though it has a well-designed UI that will make using the app extremely easy and suitable for use however this is all judging from the preview on the website so may be untrue.

Butterfly also contains the ability to set “reminders” which are like alarms but come with a input screen so that the user can be reminded to record data, take medication or anything else that they think they should be inputting regularly.

However, all this functionality comes with two large and significant flaws:

Firstly, this app is not released and is in a fairly small closed beta of only about 500 users. This made it difficult to gain access to it which makes it challenging to fully review its actual performance. This is made worse by the fact the news regarding its release is extremely sparse and not forthcoming with sporadic updates every couple of months saying that they are working on a different feature.

The second larger issue is as of writing this it is iOS only, which puts a large price barrier in front of anyone who might have legitimate use of the program, whereas an android app or web app would be vastly more accessible and affordable for those with the condition which is important as many people struggle hold down normal lives while their conditions are hitting them hard.

Overall Butterfly is a great potential solution however the huge cost barrier to entry of having to own an iPhone is too large as not everyone may be able to afford a high end luxury mobile phone as previously mentioned in section 6.1.

This price however does not detract from vast number of features which butterfly brings to the table. Many of these features had not been considered and will be aimed to be included in the solution which will be created where possible.

### 6.2.2 MySymptoms [14]

This is an App for both android and iOS which is not designed specifically for hypothyroidism however it is extremely applicable to it as it allows the user to track intake of their food, medication, drink, sleep, symptoms and other conditions to track how they are affected.

One thing it does well is that it tracks what symptoms the user has and what they have before them and after and attempts to predict what will cause the problems. Meaning that it works with the user to work out what causes problems which if it correctly guesses the problem the user can remove the substance or issue and see if the symptoms disappears and then bring it up with their doctor later which would be extremely useful.

Beyond this analysis however it does not create reminders for medication consumption or anything of similar nature. It is essentially a smarter, more high tech version of pen and paper. This however is not a bad thing but the potential of this program is wasted when they could easily add so much more to make a much more in depth and useful app. It should be noted though that it was designed as a generic tracker rather than as an aid for a specific condition which means that the generic-ness of the app is by design as it wishes to appeal to a larger user base.

### 6.2.3 Thyroid Assistant [15]

As of writing this paper, on the 13th of May, a thyroid monitoring app of similar design was released onto the android app store. Interestingly this app contains many of the similar features to that which are planned for this project however the execution differs in several ways.

Thyroid assistant allows to the user to track their symptoms day by day through the use of a diary. They can enter food, drinks, symptoms, lab results, hospital/doctor’s appointments, medication, supplements, mood, stress and “others” into the records. These can then be displayed chronologically via the use of a table where any instance of any of these is represented by a logo. Clicking on the logo reveals what it represents.

A major issue with the data entry system of this app is to enter any new item which the user hasn’t entered before they just have to navigate four different submenus and then type in what they are entering. Whilst this does allow for maximum customizability and allows them to record pretty much anything, it also cause the user to spend a lot of time recording data especially if they are trying to record everything. This is important to consider as this app will require the user to record as much as possible to make effective use of it and if the user has a busy life this may not be possible.

Another issue with this app is that the method of viewing the “reports” is through the use of tables. For some categories this is just a chronological list, which is extremely difficult to read where as for the categories with customisable intensity the table has the date as the Y axis with the different symptoms on the X axis. The table itself is easy to use however getting insights from the table is difficult as the logo alone does not tell the user much about what happened that day only that something happened. They have to click on the logo for any useful data from that particular day. It also cannot compare the data in any meaningful way from one logo to another to see if there is any change or noticeable trends.

It does have an extremely useful guide book which gives information on the hypothyroid condition, however even this has issues as the huge blocks of texts contain far too much information to be easily digestible by the user and they will need to read for several minutes to finish one section of the guidebook, which begs the question of why not look up a more reputable and readable guide on hypothyroidism.

Despite its many flaws, this app is attempting to fill a niche which there is no other replacement for and the fact that it is simply and easy to use, if time consuming is a huge advantage for it. It has some issues in regards to using the data it has gathered however for the purposes of data gathering it is sufficient and will greatly reduce the mental load on the user. With some refinement this app could be truly great and there are several features such as the information about hypothyroidism from this which should be incorporated into the system however there are also several issues this app suffers from such as awkward data entry which the proposed system must try and avoid.

## 6.3 Literature review conclusion

Overall, despite the lack of hypothyroid specific bio-informatics material there is a huge surplus of material for bio-informatics in reference to other conditions/diseases. Tracking of menstruation and diabetes in particular have huge amounts of research done on them in relation to the consistent and repeated tracking for prediction and review, which whilst not exactly hypothyroidism, has several qualities which makes the research extremely relevant.

The fact that people want a quick and easy data entry system comes as a not so big surprise as no one wants to use a system which requires huge amounts of time and concentration every day to record data. This is something I must prioritize when designing and implementing the system.

There is a large lack of person level symptom prediction research which is relevant to hypothyroidism outside of “big data” machine learning models attempting to predict if someone has hypothyroidism.

The fact that inaccurate predictions drives users away from an app is extremely useful to know as it means that it needs to be strongly considered when implementing a prediction system as to if it will be accurate enough to be useful or will simply drive away users as it is annoyingly inaccurate.

Implementing a machine learning style system would be ideal for this project however the data protection laws around medical data such as HIPAA [16] and GDPR [17] make it extremely difficult to do this easily and legally.

It would also be extremely difficult to create a machine learning model sufficiently accurate with the authors currently level of knowledge in that field as well as creating a functional data entry and review system. This is especially important as all the effort could be for naught if the model is unable to accurately predict the user’s issues and may even drive away users. With a functional system though, further work could be done to upgrade the system to integrate the initial system to have the ability to work with a machine learning system.

In terms of existing product, all the apps reviewed have at least one or two useful features in them that are worth considering to implement within my system.

From Butterfly, there is the excellent user interface which would keep people using the system as the data entry would be easy to do anytime, anywhere.

MySymptoms has the predictive modelling for an individual user which if it could be imitated would be extremely useful for hypothyroid suffers as if it could predict when the user may have an “episode” of hypothyroid symptoms would be of great use.

Finally, Thyroid Assistant shows how not to design a system for ease of use and ease of review. The lack of thought put into this app shows as reviewing how a user has been for a period of time is extremely difficult. However it does present some interesting ideas of what to include in terms of items to track. Mostly however this is more of a guide of what not to do.

# 7. THE SOLUTION APPROACH

This section will detail the development process that was followed during the creation of the project. The justification for the use of certain technologies, tools and libraries will be given. The knowledge gained from the literature review will be used to inform the decisions made within.

## 7.1 Platform decided upon

The platform chosen for use when creating this project was mobile for the OS Android. Due to a variety of factors which will be expanded upon here.

The main reason as discussed within my literature review is that Android has a huge share of the market with over eight in ten phones running a version of Android. This means that it is extremely likely that any potential user will happen to own and use an Android phone regularly.

The second reason for using android is that the author of this document has a large amount of experience in programming in Java which makes this an obvious choice which will aid in the quick and successful creation of a useable product.

A web app was decided against as the requirements to store user data were deemed unnecessary when an Android device could simply store the data on device.

A standalone desktop style program was quickly ruled out, as the literature review made clear the user wanted to be able to use this app on the go, which a standalone desktop application made very difficult.

## 7.2 Programming languages used

As the platform which has been chosen is android, there are only three potential programming languages which are feasible for use when creating this project. They are: Java, C# and JavaScript.

Given the platform decided upon is only Android this does not require development with multiple platforms in mind. For this reason the programming language used will be Java.

Java is the only real choice as C# or JavaScript could be used if the platform were Android and iOS or Android and Windows phone however as it is only Android, Java is the simplest and easiest option to create the app as multiple tools, IDEs and libraries exist for Java app creation as well as the author being proficient with the language as mentioned within 7.1.

XML will be used side by side with Java to create the UI which the program will run off and the user will interact with.

## 7.3 Integrated development environment chosen

The IDE chosen for the creation of this project is Android studio as this contains everything the project is likely to require from automated test support to live design previews.

The author also has experience with this program which will help smooth the process of development as new tools will not be need to be learnt.

## 7.4 Open source libraries used

### 7.4.1 Bullyboo’s circle seekbar

Given the scope of the project it is unnecessary to create a whole new method of entering data into android. For this reason it was decided to use an available online library to implement what was required by the design.

This library implements a seekbar much like the stock version in Android studio. The key difference with this seekbar is that it is circular which allows it to make better use of the space available within the design as well as having more options within a smaller amount of space.

Using a circular seekbar is also very intuitive and very quick which assists in fulfilling the requirement in section 5.4.1 to be able to enter data for a specific symptom within ten to fifteen seconds.

### 7.4.2 OpenCSV

The choice was made to include OpenCSV in an effort to improve the reliability of the application as, whilst managing CSVs is generally consider quite easy, it was decided to use an open source library to both write and read to the CSVs used to store the user’s data.

This is due to the fact any CSV reading or writing code being written by the author is likely to include bugs or miss edge cases which begs the question as to why write original code which may cause issues when open source, reliable code exists that can do the same job but better.

### 7.4.3 Android GraphView

Android GraphView was included as, much like the circular seekbar, it was decided that the creation of whole new graphing tools for android was entirely outside of the scope of the program, therefore open source libraries were found which did the job which was required.

In this instance, it was chosen to use Android graph as it is a well-documented fully featured open source library which is capable of completing all the requirement which existed for it.

## 7.5 Level of prediction

Finally, the level of prediction available to the user within the app will be set to a minimal level, simply warning the user when they are doing activities which may harm them, eating the wrong thing to often or suffering from symptoms too frequently. The predictor will only attempt to correct these lifestyle choices, suggest dietary changes or recommend discussing the symptoms with a doctor as the inaccurate predictions can lead to users abandoning the application as discussed within 6.1.2.

There is a technical element to this as an accurate prediction model would require both technical skills which the author of this document does not have as well as far more data than is currently available to be used to train and test a model as also discussed within 6.1.2.

## 7.6 Development process

The development process followed was standard agile development being followed for every feature. This resulted in one feature being developed, tested and “finished” before moving onto the next feature.

This feature by feature method of creation suited the author well as this meant that all the attention could be a given to a set task until it was finished. Testing was then carried out to ensure the feature worked as planned. Where required the feature would be fixed. Maintenance was not a greatly important part of the process in reference to this project however if this project is to be released to the world at large this will become a large portion of the time put into the project.

## 7.7 Version control

Throughout the development of the project GitHub was used as version control. This is to both keep a backup of the code and to assist in regression testing as if a feature breaks, the working code can be referred to, and either replaced or corrected to get a feature working again.

The GitHub used for this project can be found at [18].

## 7.8 Solution Approach Summary

# 8. IMPLEMENTATION

Within this section, the product will go from a solution approach to a problem state to a fully functional product which fulfils the goals of the problem statement. Any design issues encounter will also be discussed with justification for the final decision made. Finally, where possible the strengths of the existing products will be incorporated and any possible weaknesses avoided.

## 8.1 Design

As this project is focused on the recording of data from the user for use in monitoring and predicting their symptoms, there are a relatively few methods of which are greatly different from each other.

The designs presented within this section are copies of the original designs made on at the beginning of the project. Due to them being on a whiteboard, they have been digitalised to improve readability.

The whiteboards designs can be seen in appendix 1.

## 8.2 Use of XML

Through the implementation, the XML layouts will be referred to. These layouts are XML documents which specifies exactly which element goes where, where its borders are, what colour it is, etcetera.

A sample of one of these XML documents will be shown below in Figure 1.



Figure 1: XML Document Snippet

As shown all the graphical elements of the design is recorded within this document. As every java class within this project has one which will be extremely similar to this one. For this reason the XML document for every class will not be provided.

## 8.3 Tab page viewer

### 8.3.1 Design

As show in the following sections, Tabs are used to display the programming and separate the sheets into 3 distinct categories.

These categories are:

* The dashboard: This is where the user can view their data and if they choose, get recommendations on what to do next.
* The Intakes: This is where the data is inputted into the system and information will be given to the user, should they ask for it, regarding the symptom they are currently entering data for.
* The Labs: This tab gives the user a long term view of the user’s lab results as well as setting alarms to take medication.

### 8.3.2 Implementation

The tab layout is implement using a modified version of the standard android view pager as shown below in Figure 1Figure 2.

**public class** NonSwipeableViewPager **extends** ViewPager {  
  
 **public** NonSwipeableViewPager(Context context) {  
 **super**(context);  
 setMyScroller();  
 }  
  
 **public** NonSwipeableViewPager(Context context, AttributeSet attrs) {  
 **super**(context, attrs);  
 setMyScroller();  
 }  
  
 @Override  
 **public boolean** onInterceptTouchEvent(MotionEvent event) {  
 *// Catches and rejects the touch event motion.* **return false**;  
 }  
  
 @Override  
 **public boolean** onTouchEvent(MotionEvent event) {  
 *// Catches and rejects the touch event motion.* **return false**;  
 }

Figure 2: A Snippet from NonSwipeableViewPager.Java responsible for the interception of swipe events from the user.

This code snippets shows the section in NonSwipeableViewPager.Java which is responsible for the interception of the user’s swipe motions when using MainActivity.Java’s viewpager.

This modified version is designed to not allow the user to swipe between the different tabs as this interferes with the function of the graph as the motions to zoom and pan the graph are interpreted as attempts to change the current tab. The same is true for the nested viewpagers within Tab B. These will be discussed within section 8.4.

The touch events are overwritten by whenever a motion event is detect the usual methods for interpreting the motion event is ignored and these new methods are called as the “scroller” has been set to this new class.

Instead of changing anything, the overwritten method always returns false meaning it will never detect a swipe or motion event which could cause the viewpager to change tab.

**private** SectionsPagerAdapter **mSectionsPagerAdapter**;  
  
NonSwipeableViewPager **mViewPager**;  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 *// Create the adapter that will return a fragment for each of the three  
 // primary sections of the activity.* **mSectionsPagerAdapter** = **new** SectionsPagerAdapter(getSupportFragmentManager());  
  
 *// Set up the ViewPager with the sections adapter.* **mViewPager** = (NonSwipeableViewPager) findViewById(R.id.***container***);  
 **mViewPager**.setAdapter(**mSectionsPagerAdapter**);  
 TabLayout tabLayout = (TabLayout) findViewById(R.id.***tabs***);  
  
 **mViewPager**.addOnPageChangeListener(**new** TabLayout.TabLayoutOnPageChangeListener(tabLayout));  
 tabLayout.addOnTabSelectedListener(**new** TabLayout.ViewPagerOnTabSelectedListener(**mViewPager**));  
  
 **int** fragmentId = getIntent().getIntExtra(**"Fragment\_ID"**,0);  
 **mViewPager**.setCurrentItem(fragmentId);  
  
}

@Override  
 **public** Fragment getItem(**int** position) {  
 *//This creates and returns a copy of the fragment that is responsible for the  
 //viewpager to display to the user.* **switch** (position){  
 **case** 0:  
 Tab\_a tab\_a = **new** Tab\_a();  
   
 **return** tab\_a;  
 **case** 1:  
 Tab\_b tab\_b = **new** Tab\_b();  
 **return** tab\_b;  
 **case** 2:  
 Tab\_c tab\_c = **new** Tab\_c();  
 **return** tab\_c;  
 }  
 **return null**;  
  
 }  
  
 @Override  
 **public int** getCount() {  
 *// Show 3 total pages.* **return** 3;  
 }

Figure 3: A snippet from MainActivity.java responsible for the initialisation and management of the Tab layout.

This viewpager is responsible for the handling of the three main tabs and returns the appropriate fragment (Tab) to the user. The majority of the code responsible for the managing of the tab and the initialisation and creation of the NonSwipableViewPager is shown above in Figure 2.

As displayed above, the ViewPager is initialised and linked to the container which is set in XML in activity\_main.xml. This container, hold everything within the app except the TabLayout.

Once the ViewPager is initialised it is linked to the adapter.

The TabLayout is then linked to the Tab layout which was also defined in activity\_main.xml. This is the graphical part which the user can click on to change tabs. This must be outside of the container as it cannot call the function inside the main activity if the container is a parent of the tab layout.

Finally, two listeners are created:

The first listener is set so that that the viewpager listens for when the tab layout changes tab, this means that on the tab changing the current page will also change.

The second and final listener is from the tab layout to the viewpager. This means that if for any reason the viewpager current page is change the TabLayout will also change to reflect that.

Finally, the initial page is set to be the dashboard. This causes Tab A to be created which will be explained within section 8.3.

## 8.4 Dashboard (Tab A)

### 8.4.1 Initial Design

In Figure 3 below the initial designs for the dashboard of the application are shown.

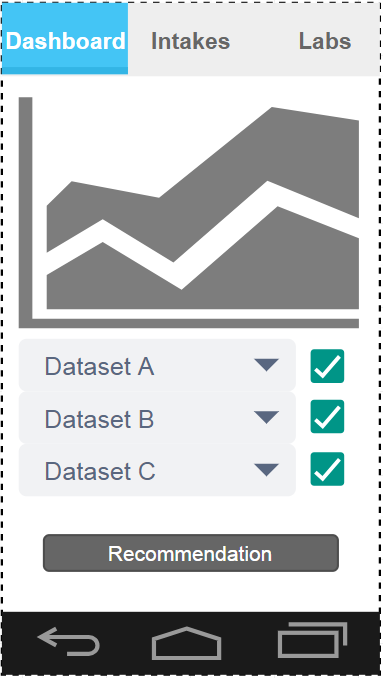


Figure 4: Dashboard Tab Initial Design

The GraphView at the top will start with the current date selected.

It is scrollable and zoom able which allows the user to see an overview of their continuous condition or deep dive into a specific time and view the day by day data for a select period.

Each of the dropdowns labelled: Dataset A, Dataset B and Dataset C allow the user to select the data set, either symptoms or intakes that they wish to inspect.

The checkbox on the right had side of the drop down when enabled shows the graph for the equivalent dataset and when disabled hides that graph on the dataset.

This enables the user view up to three different datasets in any combination they wish.

This ability to review thee separate datasets in context fulfils two of my technical requirements:

* “*The user must be able to go to a given time period and review their data (both symptoms and consumptions) for that given time period with context to the surrounding time period.*”
* “*The user must be able to compare at least two different data sets in context against each other by time to review how one effects the other.*”

The “Recommendation” button when clicked will prompt the user with a list of recommendations they could make to improve their health or tell them that they should speak to their doctor

As discussed within section 7.5 the level of prediction is very low as inaccurate prediction can drive away users. So this low level quality of life improvements is all the will be used in terms of predictions and recommendations to the user.

### 8.4.2 Implemented Final Product

Below in Figure 4 is the implemented final design.

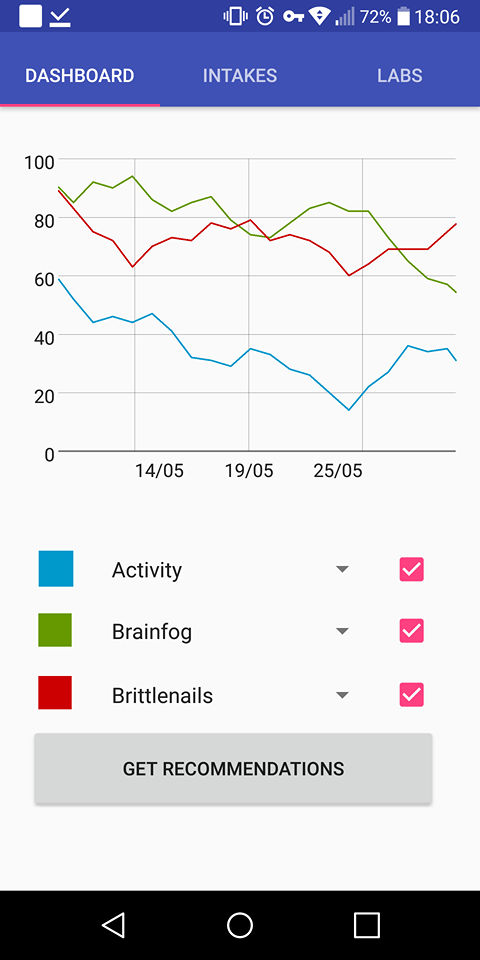


Figure 5: Implemented Dashboard Tab

When compared to the design it is apparent that the features have been carried through from the design to the implementation.

The one main addition being that of the coloured squares next to the dropdowns to allow the user to know which line is which data.

The UI fairly clean and the functionality very obvious.

An example of the pop up which occurs when the recommendation button is pressed is shown below in Figure 5.

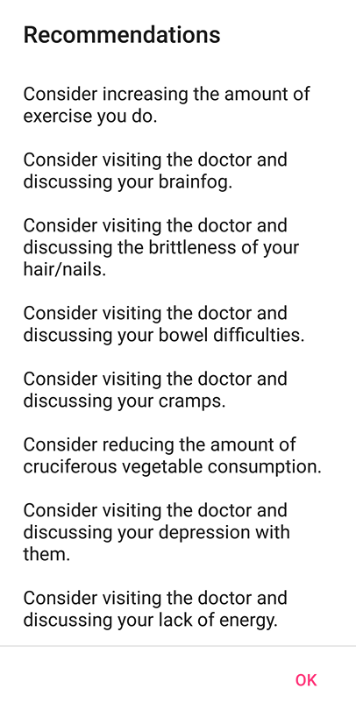


Figure 6: Recommendation Pop Up

In a real life example this popup would likely have far fewer recommendations. However, as this example is using randomly generated data, it has a random number of recommendations, which is why there are so many. The functionality is very clear though and this also is fairly unobtrusive and only is really displayed if the user chooses to ask the device for recommendations.

It was designed this was so that those who may be irritated by the inaccuracy of the prediction or simply do not care, do not have to see this data.

### 8.4.3 Implementation

The graph was implemented using GraphView as stated in section 7.4.3.

The data for the graph points is read from file by an OpenCSV reader as mentioned in 7.4.2. This goes through the CSV line by line adding the data into an ArrayList, then this ArrayList is iterated over, each line then becomes a data point which is assigned to a series.

This entire process is completed by the method getDataPoint. This method accepts a string as a parameter which is the file name of the csv it will be reading from and returns an array of data points as a series.

This method is shown below in Figure 6.

**private** DataPoint[] getDataPoint(String File) {  
 Context context = getContext();  
   
 *//Converts the passed string into the filename for the two exceptions.* **if** (File.equals(**"Loss of libido"**))  
 {  
 File = **"LossOfLibido"**;  
 }  
 **if** (File.equals(**"Pins and Needles"**))  
 {  
 File = **"PinsAndNeedles"**;  
 }  
   
 *//Constructs the file path from the string* String FileName=**"/"** + File+**".csv"**;  
 ArrayList<String[]> List = **new** ArrayList<>();  
  
 **try** {  
  
 *//Opens the reader using the filepath* CSVReader reader = **new** CSVReader(**new** FileReader(context.getFilesDir().getPath().toString() +

FileName), **'\t'** ,**'"'**,0);   
 String[] nextline;  
  
 *//Gets a single line of the csv and iterates until the line run out* **while** ((nextline = reader.readNext()) != **null**) {  
 **if** (nextline != **null**) {  
 List.add(nextline); *//Adds a line to the arraylist* }  
  
 }  
 }**catch**(IOException ie) {  
 ie.printStackTrace();  
 }  
 DataPoint[] dp = **new** DataPoint[List.size()]; *//Creates a appropriately sized array of data points* **for**(**int** i = 0; i<List.size();i++)  
 **try** {  
 String[] Temp = List.get(i); *//Gets a single line* Date date = **sdf**.parse(Temp[1]); *//Parses the string into date format. "DD/mm"* DataPoint D = **new** DataPoint(date.getTime(),Double.*parseDouble*(Temp[0]));

*//Creates a new data point using the parsed date and the data from the line.* dp[i] = D; *//Adds the point ot the array of points* } **catch** (java.text.ParseException e) {  
 e.printStackTrace();  
 }  
 **return** dp; *//Returns the array of datapoints*}

Figure 7: Tab\_A’s getDataPoint Method

The first thing this method does is correct input strings as there are two strings that can be passed to the function that do not direct the reader to an existing csv. This is due to the fact that the string is directly passed from the current value of the drop downs in Tab A.

The next step is to construct the file path of the csv. This is done by adding a ‘/’ to the beginning of the string and a “.csv” to the end of the string.

An array list of String[] is initialised. This is used to store the values read from the csv.

The method then initialises the reader passing it the file path as constructed above.

This reader is then iterated over placing every line it reads into the array list until nextLine is null. This signifies the end of the file which ends the loop.

Next, an array of data points is initialised to the size of the ArrayList of lines read from the csv.

A for loop is then entered from zero to the size of the ArrayList. For each index in the ArrayList, the value is assigned to a temporary variable. The data from the file is parsed according to “Day/Month” format. A new data point is then created using the parsed date and the value from the ArrayList.

This data point is then assigned to the array of data points at the same position as its index in the ArrayList.

This array of data points is then returned.

**spinner**.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
 @Override  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  
 **graphView**.removeSeries(**series**); *//Removes the current series if it exists* **if** (**spinner** != **null** && **spinner**.getSelectedItem()!= **null** ) *//Ensures spinner has something*

*selected* {  
 **series**=**new** LineGraphSeries<>(getDataPoint(**spinner**.getSelectedItem().toString())); *//Gets the*

*selected csvs data points* **series**.setColor(getResources().getColor(android.R.color.***holo\_blue\_dark***)); *//Sets the colour*

*of the series* **graphView**.addSeries(**series**); *//Adds the series to the graph* } **else** {  
 }  
 }  
 @Override  
 **public void** onNothingSelected(AdapterView<?> parent) { }  
});

Figure 8: Tab\_A's Spinner Listeners

This method listens to when the spinner has an item selected. When the listener is triggered the onItemSelected method is called, which removes the current series from the graph.

It then gets the string from the spinner and passes that to the getDataPoint method discussed above, which returns a series of data points.

This series is then assigned a colour and added to the graph.

This method is the same for all spinners within Tab A except each spinner has an assigned colour.

The check box by each spinner has a simple function. When unchecked it disables the spinner and removes its series from the graph. On being checked in adds the series back into the graph and enables the spinner.

**graphView**.getGridLabelRenderer().setLabelFormatter(**new** DefaultLabelFormatter(){  
 @Override  
 **public** String formatLabel(**double** value, **boolean** isValueX) {  
 **if**(isValueX)  
 {  
 **return sdf**.format(**new** Date((**long**) value));  
 } **else  
 return super**.formatLabel(value, isValueX);  
 }  
});

Figure 9: Override Axis Format

In Figure 8, the method which the axis was changed from long to a date formatted (dd/MM). This works by overriding the formatting label function and forcing it to return a date instead of a long. This results in the X axis now being a series of chronological dates which is obviously useful for reviewing data.

#### 8.4.3.1 Get Recommendation button

Button Rec = (Button) Fragment.findViewById(R.id.***REC***);  
Rec.setOnClickListener(**new** View.OnClickListener()  
{  
  
 @Override  
 **public void** onClick(View v) {  
 String output = **"\n"**; *//Initialises the string as a line return character* String[] ResourceNames2 = getResources().getStringArray(R.array.***Resulttypes***); *//gets an array of all possible strings for the input* **for** (String s: ResourceNames2) *//Iterates over the list of strings* {  
 **try** {  
 *//Corrects the input strings to be the usable strings* **if** (s.equals(**"Loss of libido"**))  
 {  
 s = **"LossOfLibido"**;  
 }  
 **if** (s.equals(**"Pins and Needles"**))  
 {  
 s = **"PinsAndNeedles"**;  
 }  
 String FileName = **"/"** + s + **".csv"**;*//Turns string into file path  
  
 //Creates the file reader used to get the data.* CSVReader reader = **new** CSVReader(**new** FileReader(context.getFilesDir().getPath().toString() +

FileName), **'\t'**, **'"'**, 0);  
 String[] nextline;  
 **int** linecount=0;*//Initialises the line reader as 0.* **boolean** outputted = **false**;  
 **int** Count = 0;

int NumLines = 0;

ArrayList<String[]> List = **new** ArrayList<>();  
  
 **try** {  
  
 *//Opens the reader using the filepath* CSVReader reader2 = **new** CSVReader(**new** FileReader(context.getFilesDir().getPath().toString()

+ FileName), **'\t'** ,**'"'**,0);  
 *//Gets a single line of the csv and iterates until the line run out* **while** ((nextline = reader.readNext()) != **null**) {  
 **if** (nextline != **null**) {

NumLines++;}  
  
 }  
 }**catch**(IOException ie) {  
 ie.printStackTrace();  
 }  
  
 **while** ((nextline = reader.readNext()) != **null**) *//Iterates over each line until the lines run*

*out* {  
 **if** (nextline != **null**&&linecount>=NumLines-20) *//For every line which isn't the last*

*line and is in the last 20 days.* {  
 **switch**(s){ *//Case statement passed the string from the array of possible strings* **case "Activity"**: *//For this string* **if**(Double.*parseDouble*(nextline[0])<40) *//if the value is less than 40* {  
  
 Count++; *//Count increases* **if**(Count>=8) *//If count is greater than 8* {  
 **if** (outputted==**false**) *//If it hasn't already outputted.* {  
 output = output + **"Consider increasing the amount of exercise**

**you do.\n\n"**; *//Adds a string to the output string*

outputted = **true**; *//Flags as already outputted.* }  
 }  
 }  
 **break**;

Figure 10: A snippet of the code from the recommendation case statement.

Above in Figure 9 is a snippet of the code that gives recommendation to the users regarding their lifestyle choice and food consumption.

It creates an array of strings of all possible data types recorded by the program.

It also constructs the file path for each of these data types by appending a ‘/’ to the front and “.csv” to the end.

The number of entries in the CSV is calculated by counting the number of times the reader reads the file until it hits the end.

It then iterates over every string within the array of strings, reading the associated csv file line by line. As the predictor only wants to take recent events into account, there is an if statement which causes only the most recent twenty days of data to be considered.

The number of instances beyond a threshold are then counted. If this this number breaks a set threshold, this triggers the recommendation message.

A message is then concatenated onto the end of the output string and a Boolean flag set to true to stop the message being added multiple times.

Each recorded data type has its own thresholds which have been set based on the messaged displayed within Tab B when entering data compared to the recommendations given by the NHS or British Thyroid foundation for an adult.

## 8.5 Intakes (Tab B) Design

During the creation the intake tab, there was difficulty in designing the UI for the intakes tab to allow useful data for comparison as the initial design gave only 4 potential outcomes which made the comparison of the data difficult. It was also not particularly attractive nor intuitive to use as each of the buttons was a colour coded logo based on the page logo.

This section will discuss the difficulties had when designing this page and highlight the changes made and why these changes were required.

### 8.5.1 Initial Design

Below in Figure 10 is the initial design for the layout of my intakes tab. This is comprised of three main sections.

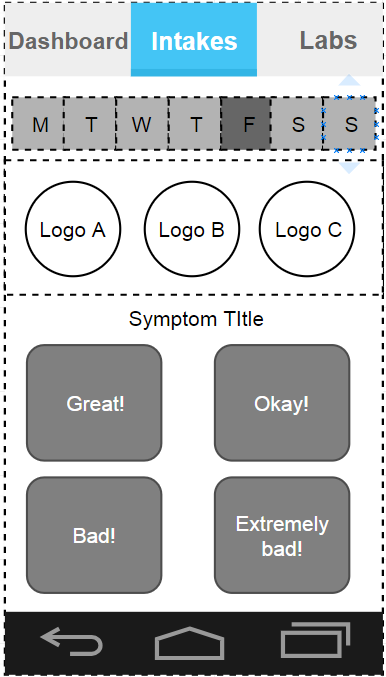


Figure 11: Input Tab Initial Design

The first section is the week overview. This section shows the user the current week the user is in both by day and date and then highlights the current day.

Whilst this is not useful to the user in terms of hypothyroidism and is not designed based on any of the technical specifications, it is useful for keeping context of the current day in relation to the rest of the week and general time management.

The second section is the Logo A, B and C section. This is a separate section which will be swipe-able to allow the user to quickly navigate to the required symptom. They can then tap the icon and it will move them to that input field.

The third and final section is the data entry section. This will consist of four buttons which will allow the user to press this to enter this data into the app. This section is the part where the main issues regarding this original design occurred from.

The fact that only choose one of the four values can be entered when collecting data means that it is in conflict with one of the technical requirements mentioned in section 5.4.1:

* “*The data entry system must allow for interval data.*”

This is requirement is not satisfied as the value is represented by a written value which cause the data to be nominal, meaning the difference between two given values is not quantifiable.

This is a sizable issue as without be able to quantify the difference between values make the inspection and analysis of the data difficult.

It also does not provide the user with any information about the specific symptom that the user’s page is currently on. This is another requirement mentioned in 5.4.1 which is not met by this design:

* “*Some basic advice/information should be given to the user regarding each symptom within the app.”*

Admittedly this information could be provided on one of the other tabs, however this is the most suitable place for this information as the user may wish to quickly review what a symptom is if they are unsure or how much of a certain food they are allowed to eat.

These two conflicts led to this tab being redesign to conform to the requirements mentioned previously within this section.

Despite these issues this design did fulfil the two other requirements given in 5.4.1:

* “*The user must be able to complete their data entry for a specific symptom within 10-15 seconds with minimal thought/effort.*”
* “*The user must be able to complete the data entry for every available symptom within 2-3 minutes.*”

There requirements were fulfilled as the process of clicking a button is incredibly easy. This ease of entry should be attempted to be replicated within the redesign of this page.

### 8.5.2 Refined Design

Below in Figure 11 is the redesigned intakes tab.

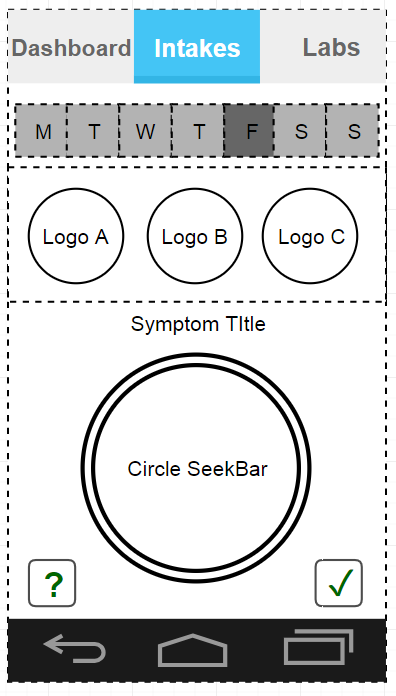


Figure 12: Input Tab Improved Design

The mains changes made to this tab was to remove the four button data entry system and replace it with Bullyboo’s circular seekbar. This enabled the user to enter any values from zero to one hundred for each symptom making this data type interval in compliance with the requirement from 5.4.1.

Another significant changes made were to include a submit button as unlike the previous solution the data should not be submitted when the circle is tapped as this will likely submit the wrong data, so for reason a submit button was added to allow the user to then select the value of their data and then submit when they are happy.

The final significant change made was to include a help button. This button pops up a dialog box with useful information about the selected page.

These two changes now bring this tab into compliance with the requirements list in the 5.4.1:

* “*Some basic advice/information should be given to the user regarding each symptom within the app.”*
* “*The data entry system must allow for interval data.*”

Because this now satisfies these requirements, this design was kept.

### 8.5.3 Implemented Final Input Tab

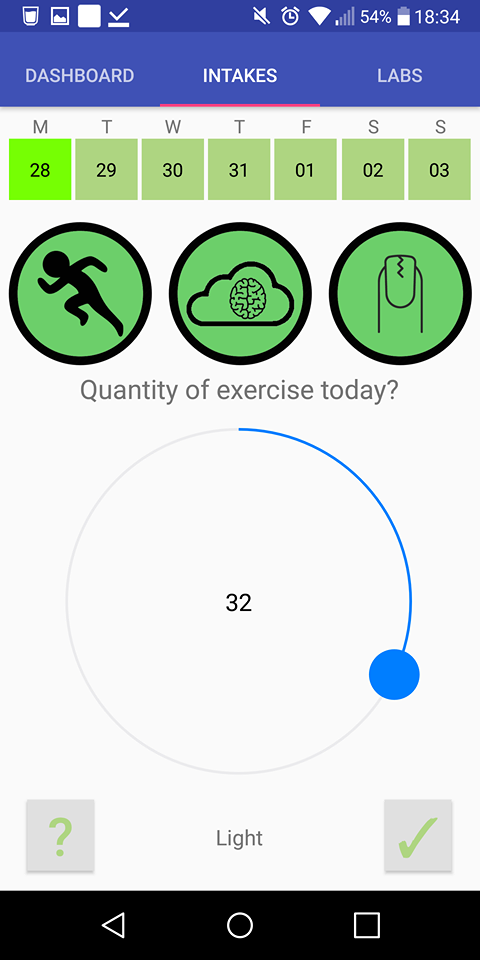


Figure 13: Implemented Intake Tab

As shown can see in Figure 12 the design has been closely mirrored with only one major change. This change is that of adding a line of text to give context to the value the user will be entering.

Without this the user has difficulty knowing if high values are good or if high values are bad. It also helps the user enter the right data as it gives them some ideas of what the normal value or right value for that specific page may be.

### 8.5.4 Implementation

#### 8.5.4.1 Week overview

Date date = **new** Date(); *//Gets current date and time*Calendar c = Calendar.*getInstance*(); *//Creates an instance of calendar*c.setTime(date);*//Sets the calendars time to be now***int** dayOfWeek = c.get(Calendar.***DAY\_OF\_WEEK***); *//Gets the current day of the week as a number*Calendar now = Calendar.*getInstance*(); *//creates another instance of calendar*SimpleDateFormat format = **new** SimpleDateFormat(**"dd"**); *//date format for the output*String[] days = **new** String[7]; *//Seven size array***int** Day = -now.get(GregorianCalendar.***DAY\_OF\_WEEK***) + 2; *//add 2 if your week start on monday*now.add(Calendar.***DAY\_OF\_MONTH***, Day); *//Adds a day to the calendar.***for** (**int** i = 0; i < 7; i++) *//Iterates seven times to fill the current week*{  
 days[i] = format.format(now.getTime()); *//Gets the time from the current calendar day in the*

*pre-determined format* now.add(Calendar.***DAY\_OF\_MONTH***, 1); *//This iterates the calendar day*}  
  
**final** TextView Sunday = (TextView) Fragment.findViewById(R.id.***Sunday***); *//Links the textview to the XML*

*textview*Sunday.setText(days[6]); *//sets the text of the textview*Sunday.setBackgroundColor(getResources().getColor(R.color.***LightGreen***)); *//Sets the colour of the text*

**final** TextView Monday = (TextView) Fragment.findViewById(R.id.***Monday***);  
Monday.setText(days[0]);  
Monday.setBackgroundColor(getResources().getColor(R.color.***LightGreen***));  
  
**final** TextView Tuesday = (TextView) Fragment.findViewById(R.id.***Tuesday***);  
Tuesday.setText(days[1]);  
Tuesday.setBackgroundColor(getResources().getColor(R.color.***LightGreen***));  
  
**final** TextView Wednesday = (TextView) Fragment.findViewById(R.id.***Wednesday***);  
Wednesday.setText(days[2]);  
Wednesday.setBackgroundColor(getResources().getColor(R.color.***LightGreen***));  
  
**final** TextView Thursday = (TextView) Fragment.findViewById(R.id.***Thursday***);  
Thursday.setText(days[3]);  
Thursday.setBackgroundColor(getResources().getColor(R.color.***LightGreen***));  
  
**final** TextView Friday = (TextView) Fragment.findViewById(R.id.***Friday***);  
Friday.setText(days[4]);  
Friday.setBackgroundColor(getResources().getColor(R.color.***LightGreen***));  
  
**final** TextView Saturday = (TextView) Fragment.findViewById(R.id.***Saturday***);  
Saturday.setText(days[5]);  
Saturday.setBackgroundColor(getResources().getColor(R.color.***LightGreen***));

**switch** (dayOfWeek)

{  
 **case** 1:  
 Sunday.setBackgroundColor(getResources().getColor(R.color.***HighlightGreen***)); *//Sets the colour*

*of the box for the current day to be a different colour.*  
 **break**;  
 **case** 2:  
 Monday.setBackgroundColor(getResources().getColor(R.color.***HighlightGreen***));  
 **break**;  
 **case** 3:  
 Tuesday.setBackgroundColor(getResources().getColor(R.color.***HighlightGreen***));  
 **break**;  
 **case** 4:  
 Wednesday.setBackgroundColor(getResources().getColor(R.color.***HighlightGreen***));  
 **break**;  
 **case** 5:  
 Thursday.setBackgroundColor(getResources().getColor(R.color.***HighlightGreen***));  
 **break**;  
 **case** 6:  
 Friday.setBackgroundColor(getResources().getColor(R.color.***HighlightGreen***));  
 **break**;  
 **case** 7:  
 Saturday.setBackgroundColor(getResources().getColor(R.color.***HighlightGreen***));  
 **break**;  
}

Figure 14: Snippet from Tab\_B.Java responsible for the week's overview.

The code shown above in Figure 13 is the code responsible for the week’s overview.

It gets the current date and time, inputs it into a calendar and then works out what the current day is in terms of an integer value from one to seven.

This number is then used within the calendar function to provide the calendar dates for each of the days in the week.

This date is then set sequentially for each of the seven TextViews representing each of the days of the week.

Finally the current day is passed into a case statement which highlights today’s date in bright green.

#### 8.4.4.2 Upper and Lower viewpager.

**public void** SetPage(**int** Pos)

{  
 **inputPager**.setCurrentItem(Pos, **true**);  
}  
  
  
@Nullable  
@Override  
**public** View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState)  
{  
 View Fragment = inflater.inflate(R.layout.***tab\_b***, container, **false**);  
  
 **final** ViewPager viewPager = (ViewPager) Fragment.findViewById(R.id.***TypeSelector***);  
 **adapterViewPager** = **new** PageAdapterSelector(getChildFragmentManager());  
 viewPager.setAdapter(**adapterViewPager**);  
  
 **final** ViewPager viewPager2 = (ViewPager) Fragment.findViewById(R.id.***InputSelector***);  
 **adapterViewPager2** = **new** PagerAdapterInput(getChildFragmentManager());  
 viewPager2.setAdapter(**adapterViewPager2**);  
 **inputPager** = viewPager2; *//Links the "inputPager" with the ViewPager2 allowing the SetPage method*

*to be called by the child fragment in viewPager.* viewPager2.addOnPageChangeListener(**new** ViewPager.OnPageChangeListener() {  
 @Override  
 **public void** onPageScrolled(**int** position, **float** positionOffset, **int** positionOffsetPixels) {}  
  
 @Override  
 **public void** onPageSelected(**int** position)  
 {  
 viewPager.setCurrentItem((**int**) Math.*floor*(position/3)); *//Keeps the upper viewpager in sync*

*with the current shown lower fragment* }  
  
 @Override  
 **public void** onPageScrollStateChanged(**int** state) { }  
 });

Figure 15: SetPage method and Tab\_B.java's onCreate and listeners.

The code shown in Figure 14 above is the SetPage method and the main body of Tab\_B class.

The SetPage method is used by the fragments which are within viewpager to set the current page within the viewpager2. This allows the user to navigate the symptoms by taping on the logos.

The onPageChangeListener on viewPager2 allows for viewpager to stay in sync with the viewpager2. As when the user swipes through more than 3 pages this will cause the top viewpager to switch page to match it.

Finally, the main body of this code is responsible for the initialisation of the two nested viewpagers which display both the input fragments and the “trio” fragments. Both of these will be expanded upon within section 8.4.4.3 and 8.4.4.4 respectively.

#### **8.4.4.3 Input fragments**

#### 8.4.4.3.1 Help text

Button Help = (Button) view.findViewById(R.id.***help***);  
Help.setOnClickListener(**new** View.OnClickListener()  
{  
  
 @Override  
 **public void** onClick(View v) {  
 AlertDialog alertDialog = **new** AlertDialog.Builder(getContext()).create();  
 alertDialog.setTitle(**"Exercise and activity!"**);  
 alertDialog.setMessage(**"Exercise releases endorphines, boosting your metabolism and fights weight gain.\n\nIntensity isn't important, just that you do it. "**);  
 alertDialog.setButton(AlertDialog.***BUTTON\_POSITIVE***, **"OK"**,  
 **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 dialog.dismiss();  
 }  
 });  
 alertDialog.show();  
 }  
});

Figure 16: Input Fragment Example Help text Code Snippet

Above in Figure 15, is the code responsible for the creation of the help text alertDialog.

It is triggered by an OnClickListener which has been set to listen to the appropriate button based on the Input Fragment’s XML layout. Figure 17 shows the generated output dialog.

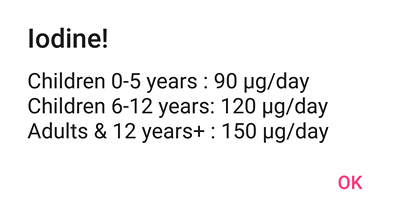


Figure 17: Example help text for iodine input

#### 8.4.4.3.2 Writing data to file.

Button Submit = (Button) view.findViewById(R.id.***button4***);  
Submit.setOnClickListener(**new** View.OnClickListener()  
{  
 @Override  
 **public void** onClick(View v)  
 {  
 Context context = getContext();  
 String[] LastDateLine;  
 String LastDate=**null**;  
 List<String[]> File = **null**;  
  
 **try** {  
 CSVReader reader = **new** CSVReader(**new** FileReader(context.getFilesDir().getPath().toString() + FileName), **'\t'**, **'"'**, 0);  
 File = reader.readAll();  
 LastDateLine =File.get(File.size()-1); *//Gets the last line of a file.* LastDate = LastDateLine[1]; *//Gets the last date within the file* }  
 **catch** (IOException ie)  
 {  
 ie.printStackTrace();  
 }  
  
  
  
 **if** (!(LastDate.equals(StringDate))) *//If the last date is not today, Write New* {  
 **try** {  
 CSVWriter writer = **new** CSVWriter(**new** FileWriter(context.getFilesDir().getPath().toString() + FileName, **true**), **'\t'**);  
 String Enter = Input.getValue() + **","** + StringDate;  
 String[] entries = Enter.split(**","**);  
 writer.writeNext(entries); *//Writes a line to the end* Toast.*makeText*(context, **"Data Submitted for "** + StringDate, Toast.***LENGTH\_SHORT***).show();  
 writer.close();  
 } **catch** (IOException ie) {  
 ie.printStackTrace();  
 }  
 }  
 **else** *//Otherwise, delete and replace the last line as that is todays data.* {  
 **try**{  
 CSVWriter writer = **new** CSVWriter(**new** FileWriter(context.getFilesDir().getPath().toString() + FileName, **false**), **'\t'**);  
 File.remove(File.size()-1); *//Deletes the last line* writer.writeAll(File);  
 writer.close();  
 CSVWriter writer2 = **new** CSVWriter(**new** FileWriter(context.getFilesDir().getPath().toString() + FileName, **true**), **'\t'**);  
 String Enter = Input.getValue() + **","** + StringDate;  
 String[] entries = Enter.split(**","**);  
 writer2.writeNext(entries); *//Writes a line to the end* writer2.close();  
 Toast.*makeText*(context, **"Data Resubmitted for "** + StringDate, Toast.***LENGTH\_SHORT***).show();  
 } **catch** (IOException ie) {  
 ie.printStackTrace();  
 }  
 }  
 }  
});

Figure 18: Code responsible for the Input Fragments CSV writing.

The code above in Figure 17 is responsible for each of the Input Fragments.

This code loads the entire relevant CSV into memory. It then goes to the very last entry, and sets LastDate to be equal to the last entry within the CSV.

LastDate is then compared to the current date. If it’s the same, the last line in the file is deleted and the remaining CSV is then written to file, and then today’s new value is appended to the bottom. This value is gained from calling the method getValue to this fragments circle spinner.

If the values are different, today’s new value is simply appended to the bottom.

For both outcomes, a Toast is created which notifies the user that their data has been either submitted or resubmitted.

#### 8.4.4.3.3 Circle Spinner

**final** CircleSeekBar Input = (CircleSeekBar) view.findViewById(R.id.***circleSeekBar***);  
**final** TextView output = (TextView) view.findViewById(R.id.***Readout***);  
Input.setValue(0); *//Starts the spinner at 0.*Input.setOnValueChangedListener(**new** CircleSeekBar.OnValueChangedListener() {  
 @Override  
 **public void** onValueChanged(**int** i) {  
 **int** v = Input.getValue(); *//Changes the text below the spinner to give context to the value they are entering.* **if** (v <= 25) {  
 output.setText(**"Sedentary"**);  
 }  
 **else if** ( v > 25 && v <=50)  
 {  
 output.setText(**"Light"**);  
 }  
 **else if** ( v > 50 && v <= 75)  
 {  
 output.setText(**"Moderate"**);  
 }  
 **else if** ( v > 75 )  
 {  
 output.setText(**"Intense"**);  
 }  
  
  
 }  
});

Figure 19: CircleSeekBar Implementation

In Figure 18, the circleSeekBar is linked to its XML representation within the UI and initialised to 0.

The text that gives context to the value within the circle seekbar is also linked to its XML.

A listener is then created which detects whenever the circleSeekBar changes value. On this listener being triggered, the current value is compared to 4 thresholds. Depending on which band it end up in, causes the text underneath to change.

This process is the same for all of the Input fragments with exception of the weight fragment, which constantly updates to be the value of the spinner with “KG” appended onto it.

#### 8.4.4.4 High level navigation fragments.

ImageButton button1 = (ImageButton) view.findViewById(R.id.***EnergyLogo***);  
button1.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 FragmentManager fm = getFragmentManager();  
 Tab\_b fragment = (Tab\_b)getParentFragment();  
 fragment.SetPage(0);  
 }  
});

Figure 20: Calling another class’s function from a child fragment

Above in Figure 19 is the method in which the SetPage function of Tab B is called from the nested upper fragments.

This is done by using the fragment manager to assign the current parent fragment (Tab B) to be assigned to a local Tab B fragment. As java passes by reference any change made to the local fragment will be also made to the parent fragment.

The local fragment then has the SetPage method called which is passed an integer as the page to be set too. This causes the lower fragment to switch to the given integer page.

## 8.6 Labs (Tab C) Design

### 8.6.1 Initial Design

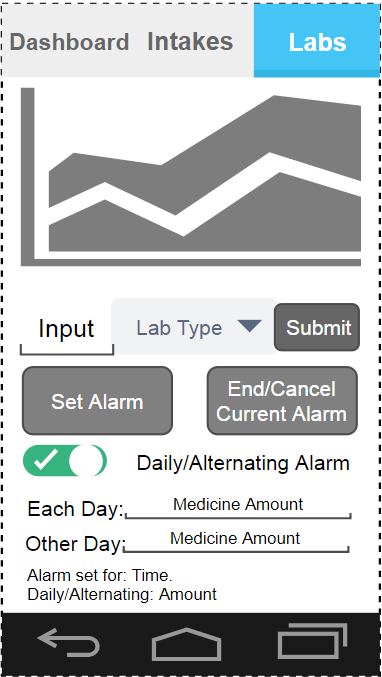


Figure 21: Lab Tab Initial Design

Above in Figure 20 is the initial design for Labs tab. This contains two main elements.

The first main element is the graph and associated data entry.

This consists of the Graph which will show the long term blood tests results to the user.

The second element is the data entry. This allows the user to submit their blood test results. The designs of this data entry is not as important as the intakes tab as the user is less likely to be using it more than once every two to three months. Meaning that a drop down and box to type in the value is an acceptable way of doing it.

The second main element of this tab is the alarm section. This consist of the set alarm button which will prompt the user with a Time Picker Dialog to set the time for the alarm.

The other button for the alarm is to end or cancel the current alarm.

The other parts of this section are there so the user can enter the medication they must take, which will then be displayed to them at the time the alarm goes off.

Finally the two lines at the bottom show the user what the current alarm time is and what if any medication they will be taking.

### 8.6.2 Implemented Final Product

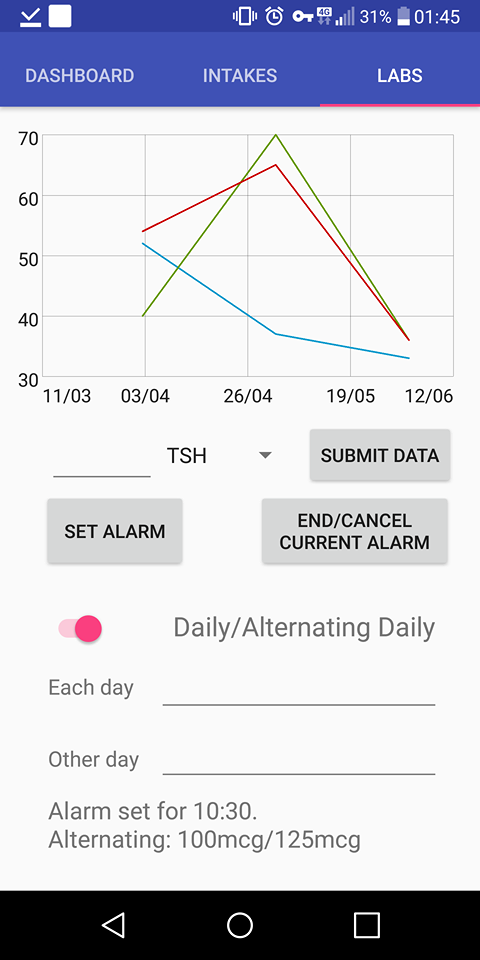


Figure 22: Implemented Labs Tab

As shown in Figure 21, the implemented copy closely fits the original design.

Upon implementation it became apparent this UI has some issues in clarity and usability however given this page will not be used nearly as much as the other two this is not a huge issue.

However it is usable enough that a redesign was deemed unnecessary and would be a waste of time that could be spent elsewhere.

Figure 22 below shows the time picker dialog that is created when the set alarm button is pressed.

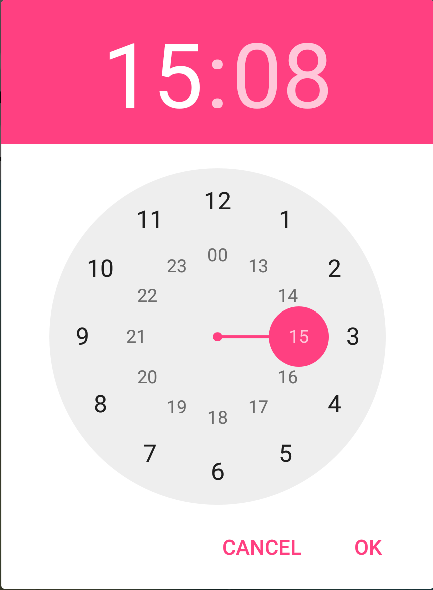


Figure 23: The Time Dialog Picker Pop up.

### 8.6.3 Implementation

**alarmManager** = (AlarmManager) context.getSystemService(Context.***ALARM\_SERVICE***);  
**final** Intent intent = **new** Intent(context, Alarm\_Receiver.**class**);

**final** Button button = (Button) Fragment.findViewById(R.id.***button***);  
button.setOnClickListener(**new** View.OnClickListener(){  
**public void** onClick(**final** View v) {  
  
 **timePickerDialog** = **new** TimePickerDialog(context, **new** TimePickerDialog.OnTimeSetListener() {  
 @Override  
 **public void** onTimeSet(TimePicker timePicker, **int** selectedHour, **int** selectedMinute) { *//Creates the dialog* **hour** = selectedHour; *//Gets current hour* **minute** = selectedMinute; *//Current minute* calendar.set(Calendar.***HOUR\_OF\_DAY***, **hour**); *//Set the calendar to now using previous data.* calendar.set(Calendar.***MINUTE***, **minute**);  
  
 intent.putExtra(**"extra"**, **true**); *//modifies the intent to contain extra data.* **alarmIntent** = PendingIntent.*getBroadcast*(context, 0, intent, PendingIntent.***FLAG\_UPDATE\_CURRENT***); *//Creates a pending intent using the data from the intent* **if** (calendar.getTimeInMillis() < System.*currentTimeMillis*())  
 {  
 calendar.add(Calendar.***DATE***, 1); *//Stops the alarm going off instantly if the time has already based. Means it goes off then next day instead.* }  
  
 String date = String.*valueOf*(calendar.get(Calendar.***HOUR\_OF\_DAY***)); *//Creates stingg of the hour and the minute.* String date2 = String.*valueOf*(calendar.get(Calendar.***MINUTE***));  
  
 **alarmManager**.setInexactRepeating(AlarmManager.***RTC\_WAKEUP***, calendar.getTimeInMillis(),AlarmManager.***INTERVAL\_DAY***,**alarmIntent**); *//Creates a repeating alarm using hte pending intent created earlier.* Context context = getContext();  
  
 Toast.*makeText*(context, **"Alarm set for: "** + date + **":"** + date2, Toast.***LENGTH\_SHORT***).show(); *//Notifies the user the alarm has been set.*

}

Figure 24: Alarm Intent and Time Picker Dialog

In Figure 23 the code for the creation of the alarms from the time picker dialog is shown.

Firstly, an intent is created to be sent to the Alarm\_Receiver.Java class. This class will be further explained in 8.5.3.2.

A time picker dialog is then created to select an hour and time.

Next, the calendar is set to the current time. So that the shown time picker will default to the time as the user opens it.

An intent is created with an extra Boolean payload. This will be used later to create the alarm.

A pending intent is then created using the previously created intent. This will be used to set the alarm manager to go off in the future rather than now.

The time the user has selected is then checked against the current time. If the time has already passed a day is added to it so that it goes off the next day at the specified time.

A repeating inexact alarm is then set using the android alarm manager, the created pending intent along with the time the user specified.

A toast is then shown to the user so they know an alarm has been set at their specified time.

button2.setOnClickListener(**new** View.OnClickListener(){  
 **public void** onClick(**final** View v) {  
 **alarmManager**.cancel(**alarmIntent**);  
  
 intent.putExtra(**"extra"**, **false**);  
  
 getContext().sendBroadcast(intent);  
 }  
  
  
});

Figure 25: Alarm Stop Cancel Button Code

#### Figure 24 shows the code to stop or cancel the alarm.

First, a listener is attached to a previously defined button.

Then a cancel intent is sent to the alarm manager which was used in Figure 23, an intent is then created with the Boolean payload set to false which indicates to the alarm that the sound should be stopped.

This intent is the broadcast.

#### 8.5.3.2 AlarmReciever

**public class** Alarm\_Receiver **extends** BroadcastReceiver {  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
  
 **boolean** flag = intent.getExtras().getBoolean(**"extra"**);  
  
 Intent service\_intent = **new** Intent(context, AlarmPlayer.**class**);  
  
 service\_intent.putExtra(**"extra"**, flag);  
  
 context.startService(service\_intent);  
 }  
}

Figure 26: Alarm\_Receiver.Java

The Alarm\_Receiver class is used to receive the intents that are created by the pending intent within the code in Figure 23.

Upon the time of the alarm being reached, an intent will be sent to this class. This class will extract the Boolean payload and then create a new service intent including the Boolean payload which is sent to the AlarmPlayer class.

Finally, start service is called passing it the created service intent which will cause the service within AlarmPlayer to be started.

This class is extremely important as without it the alarm would only go off when the app was running. When the alarm is set like this, the alarm can be set and still go off even if the app is closed or the phone has been restarted as the intent will force this code to run, which in turn forces the alarm to go off.

#### 8.5.3.3 AlarmPlayer

**if** (!**this**.**isRunning** && startId == 1) *//Alarm not running and should be playing*{

**media\_song** = MediaPlayer.*create*(**this**, R.raw.***kids***); *//Creates a media player* **media\_song**.start(); *//Starts the media player* **this**.**isRunning** = **true**;  
 startId = 0;  
  
 *//Creates the notification manager and sets the context to allow the user to be notified regarding what to take.* NotificationManager notificationManager = (NotificationManager) getSystemService(***NOTIFICATION\_SERVICE***);  
 Intent intent\_Tab\_c = **new** Intent(**this**.getApplicationContext(), Tab\_c.**class**);  
 PendingIntent pendingIntent\_main\_actvity = PendingIntent.*getActivity*(**this**, 0, intent\_Tab\_c,0);  
  
 Calendar date = Calendar.*getInstance*();  
 **int** day = date.get(Calendar.***DAY\_OF\_YEAR***);  
  
 **if**(alternate.equals(**"true"**)) *//For each alternate day* {  
 **if** (day%2==0) *//for each even calendar day* {  
 Notification AlarmPopup = **new** Notification.Builder(**this**)  
 .setContentTitle(**"Take your Medication"**)  
 .setContentText(daily) *//Normal Notification* .setContentIntent(pendingIntent\_main\_actvity)  
 .setAutoCancel(**true**)  
 .setSmallIcon(R.drawable.***thyroidlogo***)  
 .build();  
   
 notificationManager.notify(0, AlarmPopup);  
 } **else** { *//every odd day* Notification AlarmPopup = **new** Notification.Builder(**this**)  
 .setContentTitle(**"Take your Medication"**)  
 .setContentText(alt) *//Alt notification* .setContentIntent(pendingIntent\_main\_actvity)  
 .setAutoCancel(**true**)  
 .setSmallIcon(R.drawable.***thyroidlogo***)  
 .build();  
   
 notificationManager.notify(0, AlarmPopup);  
 }  
 }  
 **else** *//For every day* {  
 Notification AlarmPopup = **new** Notification.Builder(**this**)  
 .setContentTitle(**"Take your Medication"**)  
 .setContentText(daily) *//Normal Notification* .setContentIntent(pendingIntent\_main\_actvity)  
 .setAutoCancel(**true**)  
 .setSmallIcon(R.drawable.***thyroidlogo***)  
 .build();  
   
 notificationManager.notify(0, AlarmPopup);  
 }  
  
}  
**else if** (**this**.**isRunning** && startId == 0) *//Is running and stop pressed*{  
 **media\_song**.stop();  
 **media\_song**.reset();  
 **this**.**isRunning** = **false**;  
 startId = 0;  
}  
**else if** (!**this**.**isRunning** && startId == 0) *//Not running and stop pressed.*{  
 **this**.**isRunning** = **false**;  
 startId = 0;  
}  
**else if** (**this**.**isRunning** && startId == 1) *//Is running and set pressed.*{  
 **this**.**isRunning** = **true**;  
 startId = 1;  
  
} **else** {}

Figure 27: Alarm Player Snippet: Causes Alarm to play based on the Boolean payload

Figure 26 shows the code use to start and stop the alarm as required.

When the Boolean payload is true and the alarm is currently not going off, this causes an if statement to be entered. This statement is responsible for the creation of the sounds and displaying the notification in the status bar.

It creates a MediaPlayer and sets it to running.

Depending on if the alarm is set to alternate day or every day, one of two things happen.

For every day, a status bar notification is then created and passed the data regarding the notification text which has been read from file.

For alternate day the same status bar notification is created however, based on if the calendar is odd or even it will use the alternate or daily data and create the corresponding notification text.

If isRunning is true and the flag is false, this means that the user wants the playing alarm to end. This occurs when the user pressed the end/cancel alarm button.

This causes the media to be stopped and reset to the beginning as well as the flag for isRunning to be set to false. This means it is ready to accept another alarm.

The other two cases are there, just in case the user pressed set alarm whilst an alarm is running and in case the user cancels an alarm when no alarm is running.

#### 8.5.3.4 GraphView

To see how the GraphView overrode the axis to have it be by date please view Figure 8.

## 8.7 Imported third party library’s

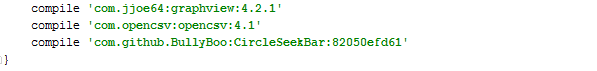


Figure 28: Snippet from Build.Gradle to include the 3rd Party Library's

Modification of the Build.Gradle was used to import the third party libraries as shown in the snippet Figure 27.

## 8.8 Android Manifest

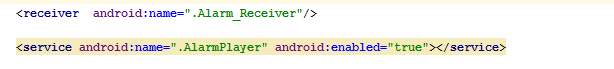


Figure 29: Snippet from the android manifest.

Figure 28 shows the modifications made to the android manifest. “<receiver/>” allows the interclass communication through the use of intents. This is required otherwise android will be block the intents from being received.

“<Service/>” is required to allow the AlarmPlayer to run long term in the background. Without this the class will stop running as soon as the app is closed.

# 9. TESTING: VERIFICATION AND VALIDATION

## 9.1 Test cases

The testing for this product has been broken into four logical sections. Each section has its own tests with any overlapping tests being included in the fourth section “general tests”.

The four sections are:

* Tab A tests.
* Tab B tests.
* Tab C tests.
* General tests.

The test plans for each section are ran through for each major feature added to the product. When a feature is added more tests were added to the plan. This meant a slow build up in the amount of tests for the entire system until completion.

Due to the relatively small scale, it is easily affordable timewise to run through every single test whenever a major change is made.

In an ideal world these would have been made into automated unit tests, however this was considered a waste of resources that could have been better spent elsewhere as the time taken to automate the process would far exceed the time taken to simply do the tests manually.

For all tests, random test data was generated for any field that required data to be tested. This is so the recommendation button has something to use to create its recommendations and the graph has data to display.

Five types of testing was done on the product:

* Blackbox testing.
* Regression testing.
* Exploratory testing.
* Platform testing.
* Acceptance testing.

Black box testing was carried out extensively with the majority of the tests done within the four sections being Blackbox tests. This is the main core of the testing and these tests passing is extremely important as if these are failing, large amounts of the application are broken.

Regression testing has also been undertaken with the test plans being run through for each feature being added to the application. As mentioned earlier, due to time constraints and the cost-benefit analysis, automating the regression test was deemed unnecessary and therefore all testing was done by hand.

Exploratory testing was done in a very extremely informal process as it just devolved into the tester attempting to break the system by any means necessary, this managed to catch a few bugs. The notable ones being the two below from general testing:

* “*If the drop downs are disabled in tab A and then another tab is switched to and then back to tab A do the lines remain as the checkbox’s indicate*”
* “*If data is entered into the alarm fields and the “Daily/Alternating” is set to a desired setting. Does this remain when the tab is switched?*”

Platform testing was done by running the app and completing the test plans on multiple devices and checking if the functionality still works as well as visually inspecting the UI to see if the change in resolution and aspect ratio has affected it. Unfortunately the author only has two device available to platform test on: A Google Nexus 6P and a Google Nexus 4. Fortunately both of these showed no problems and the app ran perfectly, however further testing must be done on more devices to fully platform test this app.

Finally, Acceptance testing was done against my requirements as well as loosely against my PID. This is expanded upon in 9.2. This meant that when designing a feature the user’s requirements were kept in mind at all time and upon starting a feature it had to be considered if the feature would support or aid in validating any of the user’s requirements. Consequently, upon a feature being completed it was compared the requirement it set out to validate and if it successfully validated the requirement only then could that test be passed. If the feature did not validate a user requirement it was reconsidered, reworked or adjusted until it did.

### 9.2.1 Tab A tests.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tab A** | | | |
| **Test Action** | **Expected Result** | **Observed Result** | **Pass/Fail** |
| Swipe along the graph in Tab A. | Pans in direction of Swipe. | Pans in direction of Swipe. | Pass |
| Pinch to zoom the graph in Tab A. | Zooms in on the pinched section. | Zoomed successfully on the pinched area. | Pass |
| Three Lines enabled. | Shows 3 lines of different colours | Shows three lines. | Pass |
| Two Lines enabled. | Shows 2 lines of different colours | Shows two lines. | Pass |
| One line enabled. | Shows 1 lines of different colours | Shows one line. | Pass |
| Check the top check box. | Corresponding line is removed. | Blue line disappears. | Pass |
| Check the middle check box. | Corresponding line is removed. | Green line disappears. | Pass |
| Check the bottom check box. | Corresponding line is removed. | Red line disappears. | Pass |
| Switch item in top dropdown. | Line switches to selected line. | Data in corresponding line changed. | Pass |
| Switch item in middle dropdown. | Line switches to selected line. | Data in corresponding line changed. | Pass |
| Switch item in bottom dropdown. | Line switches to selected line. | Data in corresponding line changed. | Pass |
| Press recommendation button. | Recommendation dialog box pops up | Text dialog pops up with recommendations on it. | Pass |
| Press “ok” on recommendation button | Dialog should close | Dialog closes | Pass |
| Exit help dialog via the back button. | Dialog should close | Dialog closes | Pass |

Table 1: Tab A Test cases

### 9.2.2 Tab B tests.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tab B** | | | |
| **Test Action** | **Expected Result** | **Observed Result** | **Pass/Fail** |
| Navigation fragment swipe left. | Trio to the left is moved into focus. | Switches to the left trio. | Pass |
| Navigation fragment swipe right. | Trio to the right is moved into focus. | Switches to the right trio. | Pass |
| Data input fragment can swipes left. | Left fragment is swiped left into focus. | Switches to the corresponding left fragment. | Pass |
| Data input Fragment can swipe right. | Right fragment is swiped left into focus. | Switches to the corresponding right fragment. | Pass |
| For every three swipes right of the data input fragment the navigation fragment will automatically switch to the next trio on the right. | Navigation fragment switches to the next on the right after three right swipes. | Navigation fragment switched to the next on the right after three right swipes. | Pass |
| For every three swipes left of the data input fragment the navigation fragment will automatically switch to the next trio on the left. | Navigation fragment switches to the next on the left after three left swipes. | Navigation fragment switched to the next on the left after three left swipes. | Pass |
| Move the CircleSeekBar | Circle seekbar is adjusted to the user value and the text below switches based on the thresholds. | Circle seekbar adjusts to user input and bottom texts adjusts to the input. | Pass |
| Press submit button | Data should show up in the graphs and a toast pops up showing user data has been submitted. | Data appears in the graphs and a toast pops up showing the user has submitted. | Pass |
| Repress the submit button | Toast pops up showing user data has been resubmitted. | Data replaces current data in today’s date and a popup occurs which notifies the user it’s been resubmitted. | Pass |
| Press the help button. | An alert dialog is created with help text. | An alertDialog is created containing help text. | Pass |
| Press the “ok” button on the help dialog. | The alert is closed. | The alert closed. | Pass |
| Exit help dialog via the back button. | The alert is closed. | The alert closed. | Pass |

Table 2: Tab B Test cases.

### 9.2.3 Tab C tests.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tab C** | | | |
| **Test Action** | **Expected Result** | **Observed Result** | **Pass/Fail** |
| Submit a “normal” data value. | Data is entered into the graph. | The data is inputted into the graph. | Pass |
| Submit a “” data value. | Nothing should happen | Nothing happens. | Pass |
| Press the set alarm button with a string in “Each day” for daily. | When time selected text should appear at bottom of screen. | Text appears at the bottom of the screen with the selected time and string. | Pass |
| Press the set alarm button without a string in “Each day” for daily. | When time selected the default text should appear at bottom of screen. | Default text appears at the bottom of the screen with the selected time and the string is empty. | Pass |
| Press the set alarm button with strings in “Each day” and “Other day” for alternating. | Text Dialog should appear, when time selected text should appear at bottom of screen. | Text appears at the bottom of the screen with the selected time and strings. | Pass |
| Press the set alarm button without strings in “Each day” and “Other day” for alternating. | Text Dialog should appear, when time selected text should appear at bottom of screen. | Default text appears at the bottom of the screen with the selected time and strings are empty. | Pass |
| Set Daily/Alternating switch to Daily | Each day should be enabled, but other day should be disabled. | Each day stays enabled. Other day becomes disabled. | Pass |
| Set Daily/Alternating switch to Alternating | Both each day and other day should be enabled. | Both each day and other day become enabled. | Pass |
| Attempt to write in “other day” when toggle is set to daily. | Should be unable to select it. | Unable to be selected. | Pass |
| Attempt to write in “other day” when toggle is set to Alternating. | Should be able to select and type in the field. | Able to select and type into field. | Pass |
| Press “End/Cancel” when no alarm running | Nothing should occur. | Nothing happens. | Pass |
| Press “End/Cancel” when alarm running | Current alarm should end. | Current alarm ends. | Pass |
| Press “End/Cancel” when alarm set. | Alarm should be cancelled. | Scheduled alarm is cancelled. | Pass |
| Press “End/Cancel” when no alarm set. | Nothing should occur. | Nothing happens. | Pass |

Table 3: Tab C Test cases

### 9.2.4 General Tests.

|  |  |  |  |
| --- | --- | --- | --- |
| **General Tests** | | | |
| **Test Action** | **Expected Result** | **Observed Result** | **Pass/Fail** |
| Click the Apps Icon | The App opens. | The app opened. | Pass |
| Click Tab A (Dashboard) | Dashboard opens. | Dashboard opened. | Pass |
| Click Tab B (Intakes) | Intakes opens. | Intakes opened. | Pass |
| Click Tab C (Labs) | Labs opens. | Labs opened. | Pass |
| Alarm goes off at pre-set time | Alarm to go off at pre-set time. | Alarm went off within the minute of the pre-set time.  Functionally the same but not perfect.  Inexact alarms do this to save battery. | Pass |
| Switch to the app using the overview button. | App should resume function to previous position. | App resumed function to previous position. | Pass |
| Exit the app using the back button. | App should exit to home screen. | Exited to home screen. | Pass |
| Exit the app using the home button. | App should exit to home screen. | Exited to home screen. | Pass |
| If the drop downs are disabled in tab A and then another tab is switched to and then back to tab A do the lines remain as the checkbox’s indicate. | Lines should behave as they do when normally disabling and enabling. | Lines behave as normal. | Pass |
| If data is entered into the alarm fields and the “Daily/Alternating” is set to a desired setting. Does this remain when the tab is switched? | Data should remain within the boxes and the toggle should stay the same. | Data remained within the boxes and the toggle stayed the same as before. | Pass |

Table 4: General Test that do not fall into a Tab's category.

## 9.2 Requirement Validation.

Within the section the requirements given in 5.4.1 and 5.4.2 are validated against.  
Each requirement and constraint is given a pass or a fail and a justification for why.

### 9.2.1 Requirement Validation.

|  |  |
| --- | --- |
| **Requirement** | **Pass/Fail and justification?** |
| The user must be able to complete their data entry for a specific symptom within 10-15 seconds with minimal thought/effort. | Pass. As the selecting a value on a circle can be done within 10 second easily. “Minimal thought/effort” is a vague requirement however I would still classify it as a pass as it is still extremely easy. |
| The user must be able to complete the data entry for every available symptom within 2-3 minutes. | Pass. Since each input takes roughly 10 seconds this fulfils this requirement. But only just. Since they can all be filled within 3 minutes. |
| The user must be able to go to a given time period and review their data (both symptoms and consumptions) for that given time period with context to the surrounding time period. | Pass. The graph enables the user to jump to any given time where they have recorded data. |
| The user must be able to compare at least two different data sets in context against each other by time to review how one effects the other. | Pass. The user can compare from one to three different data sets. |
| The data entry system must allow for interval data. | Pass. The user enters integers which are interval data, it could be argued these are ratio data as they have an absolute zero point. Regardless they still fulfil this requirement. |
| Some basic advice/information should be given to the user regarding each symptom within the app. | Pass. Advice is given to the user within Tab B if the user presses the help button. |
| The system must notify the user and give them useful customizable information regarding their medication. | Pass. The customizable alarm feature allows the user to enter whatever information they wish to see regarding their medication on a daily or alternate daily schedule. |
| The user should be able to monitor the long term blood test data in a separate context. | Pass. The long term blood test data is available in Tab C for the users viewing. |

Table 5: Requirements compared to the product.

### 9.2.2 Constraint Validation.

|  |  |
| --- | --- |
| **Constraint** | **Pass/Fail and justification?** |
| The system platform must allow for the financial situation of users wishing to make use of its features, and thus must take steps to ensure affordability and availability. | Pass. The platform is iOS which is both extremely cheap and extremely common increasing the likelihood that the user already owns a device capable of using this application. |
| The design must be sufficiently simple that those who are not entirely comfortable with technology are able to use it. | Pass. The minimalistic design is fairly easy to use. Tab C could do with some work however Tabs A and B are both usable.  This is a poorly worded constraint as it is difficult to quantify the simplicity of a design. |
| Personal user data should not be given to a third party. | Pass. No data leaves the device meaning no data is given to any party other than the devices storage. |
| Personal user data should be anonymised. | Pass. No user data is stored in a personal context. The stored data is simply a list of values and dates. |
| Where possible, user data should be kept on device. | Pass. No data leaves the device. |

Table 6: Constraints compared to product.

### 9.2.3 Project Initiation Document Validation.

For all quotes and the specification mentioned in 9.2.3.3 see appendix 2 “Project initiation document”.

#### 9.2.3.1 Brief app purpose

In the PID, a brief purpose of the project was given as “*A mobile and/or Web app to allow suffers of hypothyroidism to monitor their health and symptoms.”*

From this brief sentence we have fully satisfied its requirements as a mobile app was made that allows the user to monitor their hypothyroid symptoms and health.

#### 9.2.3.2 Project objectives

In the PID the project objective was given as:

*“Create mobile and/or Webapp to track general fitness (Weight, BMI, Macros, Exercise, etc) and diary of symptoms of hypothyroidism with the aim of tracking patterns to better help control your symptoms as well as reminders of the consumption of hypothyroid medication.”*

From this project objectives we can say we have mostly satisfied the requirements as we have implemented everything except for the macro and BMI tracking. All other features of note mentioned within this have been satisfied.

#### 9.2.3.3 Initial product specification.

This list of specifications was taken from the appendix 2, the project initiation document. Each specification is given a pass or a fail and justification why.

|  |  |
| --- | --- |
| **Specification** | **Pass/Fail and justification.** |
| Calorie tracking | Fail, weight was deemed a sufficient measure of continuously monitoring the effects of hypothyroidism on weight gain for this reason calories per day was discarded. |
| Medication Timing | Pass, the app notifies the user to take their medication at specific times. |
| Symptom Diary. | Pass, the app records the symptoms in a chronological manner. |
| Graphs and Analytics. | Pass, the app lets the user view their symptoms in a chronological in a graph as well as performing basic analytics such as recommendations to see a doctor or make lifestyle changes. |
| Hyperthyroid tracker. | Fail, quickly on in development the decision was made to focus on a hypothyroid only tracker as to cover both would lose its specialisation towards hypothyroidism reducing its usefulness for its core audience to attract those who are hyperthyroid.  If this is required a separate app could be made that would be far more specialise to this role. |
| Other medication Tracking (such as paracetamol or ibuprofen) | Fail, the amount of conflicts with other medicine is limited however this is a notable feature that is missing as commonly consumed drugs can interfere if taken too soon after taking levothyroxine. |
| BMI, Exercise tracking and weight. | Pass, Weight and exercise are both tracked. BMI is not tracked however weight and exercise are sufficient to monitor the effects of hypothyroidism on weight gain. |
| Medication timing (when the medication was consumed.) | Fail. The app does not monitor the timing of medication consumption. It does however notify the user at specific times to take their medication however it does not collect data on if it has been taken or not. |
| Macro Tracking. | Fail. Was deemed unnecessary early on in development, however one macro (Iodine) is tracked as this is significant to the hypothyroid condition. |
| Automatic analytics. | Pass. Automatic analytics are run on the data. Probably not to the level envisioned by the PID however the feature is there so it fulfils this requirement. |

Table 7: Project Initiation Document initial specification

## 9.3 Limits to verification and validation.

One issue encountered whilst validating the project was poorly written requirements and constraints. In some cases the ambiguity left a lot unsaid within the requirement or constraint which made it difficult to decide if the requirement had been met.

Fortunately the person who wrote the requirements also created the project so any missed detail could be decided upon by same person who set the requirements, however in a real life project this would be a much greater issue.

# 10. DISCUSSION: CONTRIBUTION AND REFLECTION.

## 10.1 Test Results discussion and its implications

Overall the though testing of the application has greatly improved its resilience to use and its reliability. It is impossible to say if the product is free from errors and bugs however the testing has definitely had a beneficial effect on the product.

In terms of improvements made by specific test blocks, Tables 1, 2 and 3 greatly improved the reliability of their individual tabs by ensuring the functionality behaved fully as expected.

Whereas the general test done by Table 4 ensured that all the individual parts merged together in a comfortable seamless experience that people have come to expect from apps.

The requirement and constraints testing and validation done within Tables 5 and 6 was extremely valuable as it shows that the project fulfils its task in terms of the technical requirements and specifications.

## 10.2 Success analysis

As shown in Tables 5 and 6, all current technical requirements were passed showing that the project completed what was required of it.

However as Table 7 shows, only half of the ten specifications given in the product initiation document were passed showing the project did not manage to deliver on all that it aspired to be. This however is not unreasonable as the product initiation documents goals were very broad, loosely defined and not fully reasoned, but merely a list of features that would be good to have.

Finally as shown in 9.2.3.1 and 9.2.3.2, the project managed to fully realise the purpose and objectives from the product initiation document, so whilst all specifications from the product initiation document were not met, the overarching goal of the project has been met.

When the project is compared to the problem statement in 5.1, “*There is no simple solution for tracking symptoms, consumed medication and specific food triggers for analysis and visual investigation/inspection”.* It can be confidently said that this statement has had a functional and reasonable solution made for it. This again shows the success of this project.

The points made above lead to the opinion that the project has met its goals, even if it did not fill the lofty ambitions made during the project initiation document.

## 10.3 Project limitations

There are several limitations with app as it currently stands:

Firstly, there is no security what so ever for any of the medical data stored on this device which is a huge issue that would need to be rectified if this were ever to be released publicly.

Next, despite the best efforts of the developer, the developer is not a qualified endocrinologist so the advice given and suggestions made could be incorrect.

Finally, creating a smarter prediction model will minimal data than runs on device is extremely difficult to achieve, so the level of prediction and analytics being made was extremely limited.

## 10.4 Reflections

Overall this project has proved technically and in terms of user design to be a challenge.

The awkwardness of android fragments and viewpagers causing headache after headache as they refused to work together properly. Intents, pending intents and service intents also proved difficult get properly functional as it required a lot of fiddling with the manifest and gradle to get work as required.

In terms of user design, the limited real estate available to android developers proved difficult as can be shown by the implementation of Tab C which is not entirely satisfactory and uncomfortable cramped.

One huge lesson learnt from this project is the importance of generic solutions and reusable code as huge amounts of time were spent modify and fixing bugs in the input fragment classes, but because they were implemented as standalone classes the bug had to be fixed once for every input type available to the user. Even using copy and paste this still takes time, especially when copy and paste cannot do it and the fix requires changing to match that of the input fragment.

In terms of learning experiences, the author’s confidence in writing java has hugely improved. As mentioned earlier the author already has some proficiency but the confidence in the ability is now much higher as well as understand android intra-process communication and how the various intents and pending intents work.

Another lesson learnt was how to use XML to represent more than data as throughout this project, XML has been used to represent the various UI and graphical interfaces. As previously the author only used XML to represent data.

# 11. SOCIAL, LEGAL, HEALTH, SAFETY AND ETHICAL ISSUES

As stated in the project initiation document there are no social, legal, health and safety or ethical concern associated with this project.

No personal data of the author or anyone else has been disclosed to anyone during the process of this project.

All included code not written by the author is either open sourced or under the apache license removing all ethical concerns regarding included code.

No medical decision are being made for the user, only recommendations which were specifically chosen to be on the safer side. Before any public release these would be checked with a doctor and endocrinologist to remove both the legal liability as well as the ethical issue of giving out incorrect or inaccurate medical advice.

From a social standpoint, there is no risk of this app altering society as it is designed to augment and aid the doctors ability to understand their patient and to stop them negatively impacting on their condition through lifestyle choices.

There was no use of dangerous hardware or exposure to other physical hazards during the creation of this project.

# 12. CONCLUSION AND FURTHER IMPROVEMENTS

## 12.1 Project objective and results conclusion

The primary aim of this project was to create an application for suffers of hypothyroidism to manage and monitor their symptoms. As well as this the application was also to include basic levels of assistance in regards to help the user makes lifestyle choices which could affect their condition and provide information about what they may be feeling.

Overall the resulting application fully delivered upon these goals and delivers a solution which will allow the use to track their data chronologically in a manner which allows them to see their data in context to the surrounding data for any given period of time they have records for. It provides advice regarding advantageous lifestyle changes that can be made and advises when a doctor should be contracted as the symptoms are reoccurring to frequently. It also provides information to the user regarding what a specific symptom is and what they may feel further improving their own knowledge of how their body work and how their condition will affect it. Finally it provides reminders to take medication and can be configured to say which medication the user must take on which day greatly reducing the mental strain on the user. To conclude, these features deliver on all the requirements within the objective and the problem statement in a manner that whilst it will not cure anyone, but will at least slightly improve their quality of life and understanding of their condition.

## 12.2 Future work

There is a huge amount of future work that could be done to improve or introduce useful features to this application.

The first improvement to be made would be to include a login system and the encryption of user data, as currently all the data is stored in plaintext which is a huge issue for medical data and would be required for compliance with HIPAA, GDPR and other data protection laws.

Once security has been included I would then consider including other more “useful” features. The first of which would be personalization such as using the login name to talk to directly to the user or ask the user to specify their height upon sign up so that their BMI can be calculated and displayed alongside the height graph.

Another useful feature might be to have a companion website which, given the user has a login and their data is encrypted could then be uploaded to the website for better predictions, more insightful analytics or other useful features. The data being on this site would also open it to being used to create better prediction models for the users as this would give larger sample sizes to work with.

The next improvement to be made greatly relies upon having a large number of user’s data to hand but, if the quantity of data is avaible to a machine learning tool could be created to learn the patterns of symptoms that people go through to give actual predictive insights into the world of hypothyroidism. This level of prediction could go to the level of analysing user’s lab levels whilst knowing their dosage and predicting when symptoms may cease occurring or if their dosage should be adjusted as well as predicating based on their current symptoms if they are likely to have episodes of tiredness or fatigue.

Another improvement to be made was briefly touched upon during the reflections but improving the code reusability and use of generics would greatly improve the maintainability as well as the ability to expand this app to include more symptoms, more factor and record more data, easier. As well as this, it would greatly improve the ability of the developer to make changes and manage the app long term.

Yet another improvement that could be made would be to create an automated system of testing, even if just for the simple tests as this would again, improve the maintainability and help with the long term life span of this application.

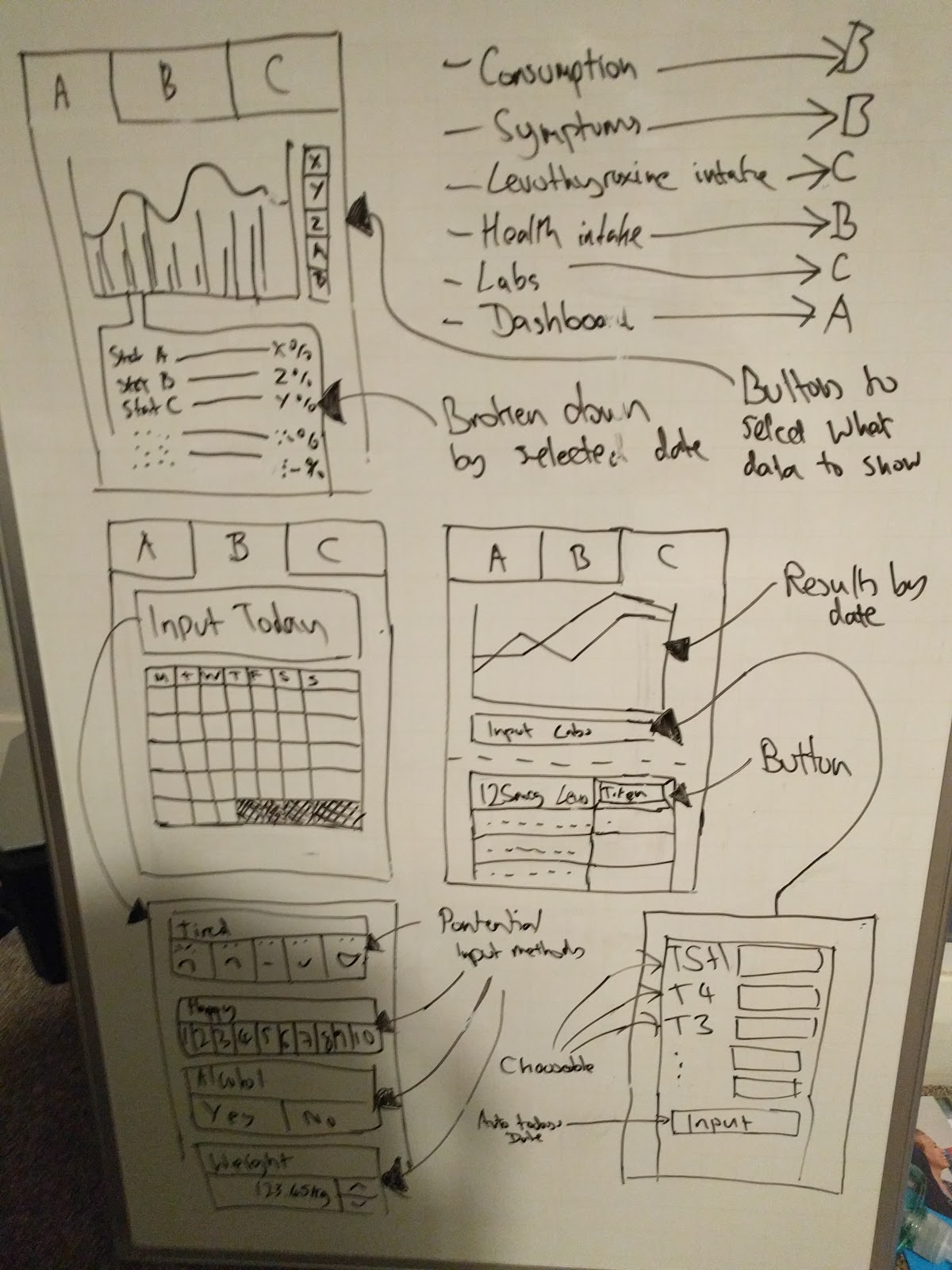
Finally, working with an endocrinologist to fix any medical inaccuracies, improve the “help” text and advice as well as give experts insights into the field that I may have missed.

# REFERENCES

|  |  |
| --- | --- |
| [1] | Microsoft, “Xamarin,” [Online]. Available: https://www.xamarin.com/. |
| [2] | Facebook, “React Native,” [Online]. Available: http://www.reactnative.com/. |
| [3] | N. B. L. J. H. K. E. A. J. S. L. R. P. J. F. J. A. K. a. S. A. M. Daniel A. Epstein, “Examining Menstrual Tracking to Inform the Design of Personal Informatics Tools,” [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5432133/. |
| [4] | Thyroid UK, “ThyroidUK.com,” [Online]. Available: http://www.thyroiduk.org.uk/tuk/about\_the\_thyroid/hypothyroidism.html. |
| [5] | PatientView and myhealthapps.net, TICBioMed, [Online]. Available: http://www.digitalezorg.nl/digitale/uploads/2015/08/GET-project.-What-patients-and-carers-need-in-apps.pdf. |
| [6] | A. P. J. F. S. A. M. Daniel A. Epstein, “A Lived Informatics Model of Personal Informatics,” Computer Science & Engineering, Human Centerd Design and Engineering DUB group, University of Washington, [Online]. Available: http://www.depstein.net/pubs/depstein\_ubi15.pdf. |
| [7] | E. Marc B. Stone and Robert B. Wallace, “Medicare Coverage of Routine Screening for Thyroid Disfunction,” [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK221544/pdf/Bookshelf\_NBK221544.pdf. |
| [8] | M. T. McDermott, “Subclinical Hypothyroidism Is Mild Thyroid Failure and Should be Treated,” [Online]. Available: https://academic.oup.com/jcem/article/86/10/4585/2848862. |
| [9] | C. W. a. F. D. Daniele Ravì, “Deep Learning for Health Informatics,” Imperial College London, [Online]. Available: https://ieeexplore.ieee.org/document/7801947/. |
| [10] | Statistia, “Great Britain iPhone market share.,” Statista, [Online]. Available: https://www.statista.com/statistics/288870/market-share-of-apple-iphone-in-great-britain/. [Accessed 2018]. |
| [11] | Statista, “Global market share of iPhones.,” Q1 2018. [Online]. Available: https://www.statista.com/statistics/216459/global-market-share-of-apple-iphone/. |
| [12] | Statista, “Global market share held by the leading smartphone operating systems,” [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. |
| [13] | Salubrious Ltd, “Butterfly,” Salubrious Ltd, [Online]. Available: http://getbutterfly.net/. |
| [14] | SkyGazer Lab Ltd, “Google Play Store,” SkyGazer Lab Ltd, [Online]. Available: https://play.google.com/store/apps/details?id=com.mhs.mysymptoms&hl=en\_GB. |
| [15] | K. Ton, “Thyroid Assistant,” [Online]. Available: https://play.google.com/store/apps/details?id=com.ikdong.health.thyroid. |
| [16] | US Department of Health & Human Services, “Summary of the HIPAA Security Rule,” [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulation. |
| [17] | EU Parliament, “GDPR Compliance and Regulation,” [Online]. Available: https://www.eugdpr.org/the-regulation.html. |
| [18] | F. Williams, “ThyroidTracker GitHub Repository,” [Online]. Available: https://github.com/Awkwardowl/ThyroidTracker. |
| [19] | M. P. M. M. Dr. G. Rasitha Banu MCA., “PREDICTING THYROID DISEASE USING DATAMINING TECHNIQUE,” [Online]. Available: https://www.researchgate.net/publication/308983859\_PREDICTING\_THYROID\_DISEASE\_USING\_DATAMINING\_TECHNIQUE. |

# APPENDICES

## Appendix 1: Original Whiteboard Designs.

****

## Appendix 2: Project Initiation Document

Individual Project (CS3IP16)

Department of Computer Science

University of Reading

Project Initiation Document

**PID Sign-Off**

|  |  |
| --- | --- |
| **Student No.** | **23013019** |
| **Student Name** | **Finley Williams** |
| **Email** | [**Mb013019@student.reading.ac.uk**](mailto:Mb013019@student.reading.ac.uk) |
| **Degree programme** (BSc CS/BSc IT) | **Computer Science with year in industry.** |
|  |  |
| **Supervisor Name** | **Dr Lily Sun** |
| **Supervisor Signature** |  |
| **Date** | **27/09/2017** |

**SECTION 1 – General Information**

**Project Identification**

|  |  |
| --- | --- |
| **1.1** | **Project ID**  (as in handbook) |
|  | - |
| **1.2** | **Project Title** |
|  | Hyper/Hypo-thyroid symptoms & fitness tracker |
| **1.3** | **Briefly describe the main purpose of the project in no more than 25 words** |
|  | **A mobile and/or Web app to allow suffers of hypothyroidism to monitor their health and symptoms.** |

**Student Identification**

|  |  |
| --- | --- |
| **1.4** | **Student Name(s), Course, Email address(s)**  e.g. Anne Other, BSc CS, a.other@student.reading.ac.uk |
|  | 23013019  F.williams@student.reading.ac.uk |

**Supervisor Identification**

|  |  |
| --- | --- |
| **1.5** | **Primary Supervisor Name, Email address**  e.g. Prof Anne Other, a.other@reading.ac.uk |
|  | Prof Lily Sun  lily.sun@reading.ac.uk |
| **1.6** | **Secondary Supervisor Name, Email address**  Only fill in this section if a secondary supervisor has been assigned to your project |
|  |  |

**Company Partner (only complete if there is a company involved)**

|  |  |
| --- | --- |
| **1.7** | **Company Name** |
|  | N/A |
| **1.8** | **Company Address** |
|  | N/A |
| **1.9** | **Name, email and phone number of Company Supervisor or Primary Contact** |
|  | N/A |

**SECTION 2 – Project Description**

|  |  |
| --- | --- |
| **2.1** | **Summarise the background research for the project in about 400 words. You must include references in this section but don’t count them in the word count.** |
|  | I will need to research into both web based applications as well as research into the creation of android and/or iOS applications. This would be both so that I can decide which is most suitable for my project as well as then learn the required elements for the final project creation itself.  I will also need to put in additional research into hypothyroidism itself to make sure that everything I include will be of use.  Some of my research is based on personal experience as a suffer of Hypothyroidism.  I attended a group meet for suffers of hypothyroidism in Bristol who will also be available have I any requirements which I am in doubt of or require a “professional opinion” on.  Also if I require any market research or significant user data provided I can plan far enough ahead I can request that the Hypothyroid suffer group fill out questionnaires or test the product or equivalent.  A final thing that I may need to research if I get “ahead of schedule” in regards to the project would be how to automate the analysis of data and to see if I can “predict” the symptoms of suffers based on inputted data. |
| **2.2** | **Summarise the project objectives and outputs in about 400 words.** These objectives and outputs should appear as tasks, milestones and deliverables in your project plan. In general, an objective is something you can do and an output is something you produce – one leads to the other. |
|  | Create mobile and/or Webapp to track general fitness (Weight, BMI, Macros, Exercise, etc) and diary of symptoms of hypothyroidism with the aim of tracking patterns to better help control your symptoms as well as reminders of the consumption of hypothyroid medication.  General Objectives   1. Research Existing methods of Track symptoms/fitness. 2. Research into the options in which the solution can be implemented. 3. Design a solution to the problem. 4. Implement the solution. 5. Test the Solution. 6. Writing-up the final report.   Milestone/Deliverables.   1. Create an App that can track and store the various symptoms of hypothyroidism. 2. Add the ability to track consumption of medication, Alcohol and other activities with the aim of tracking patterns and seeing what works for you. 3. Track Calories, Weight and other health metrics as well as graphs showing this data on a time basis. 4. Reminders and Timers for Levothyroxine (Eating and Caffeine) 5. Track Hyperthyroid symptoms in similar way to Hypothyroidism 6. Perform analytics on the data (Stretch goal) |
| **2.3** | **Initial project specification - list key features and functions of your finished project.** Remember that a specification should not usually propose the solution. For example, your project may require open source datasets so add that to the specification but don’t state how that data-link will be achieved – that comes later. |
|  | Calorie tracking.  Medication Timing (Food and Caffeine)  Symptom Diary.  Graphs and Analytics.  Hyperthyroid tracker.  Other medication Tracking (EG paracetamol or ibuprofen etc)  BMI, Exercise tracking and weight.  Medication timing (when the medication was consumed.)  Macro Tracking.  Automatic analytics. |
| **2.4** | **Describe the social, legal and ethical issues that apply to your project. Does your project require ethical approval?** |
|  | I cannot think of any Social, legal or ethical issues which would apply to my project. |
| **2.5** | **Identify and lists the items you expect to need to purchase for your project. Specify the cost (include VAT and shipping if known) of each item as well as the supplier.** e.g. item 1 name, supplier, cost |
|  | N/A  Unless I can get a dev phone which seem unnecessary but useful. |
| **2.6** | **State whether you need access to specific resources within the department or the University e.g. special devices and workshop** |
|  | Android device would be useful but not essential. I have two functioning android devices of my own I could use to develop on. |

**SECTION 3 – Project Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **3.1** | **Project Plan**  Split your project work into sections/categories/phases and add tasks for each of these sections. It is likely that the high-level objectives you identified in section 2.2 become sections here. The outputs from section 2.2 should appear in the Outputs column here. Remember to include tasks for your project presentation, project demos, producing your poster, and writing up your report. | | |
| **Task No.** | **Task description** | **Effort**  **(weeks)** | **Outputs** |
| **1** |  |  |  |
| 1.1 | **Background Research** | 3 | General research on hypothyroidism and things I need to include |
| 1.2 | Research of alternatives | 1 | A general overview of similar products available. |
|  |  |  |  |
| **2** | **Analysis and design** |  |  |
| 2.1 | Design a GUI | 2 | A UI design. |
| 2.2 | Design Calendar (Data Structure and look) | 1 | A design for the calendar |
| 2.3 | Design function to allow addition of data to each calendar day. | 1 | A design for the submission of data. |
| 2.4 | Design function to allow easy update of when medication is taken and the notification when you can eat and then when you can drink caffeine. | 1 | A design for the medication tracker. |
| 2.5 | Design graphs for the display of health metrics. | 1 | A design for the graphs |
| 2.6 | Design calorie counter. | 1 | A design for the calorie counter. |
|  |  |  |  |
| **3** | **Develop prototype** | 10 | **A working prototype** |
| 3.1 | Bug fixs | 3 | Less bugs within the program. |
| 3.2 | Decide on platform, programming language | … | A language and platform chosen. |
|  |  |  |  |
| **4** | **Testing, evaluation/validation** |  |  |
| 4.1 | Unit testing | 3 | More known bugs and problems. |
|  | Verification Testing | 3 | More known bugs and problems. |
|  | System Testing | 3 | More known bugs and problems. |
|  | Performance Testing | 3 | More known bugs and problems. |
|  | Validation Testing | 3 | More known bugs and problems. |
| 4.2 | Other forms of testing (e.g. user acceptance testing) | 4 | More known bugs and problems. |
|  |  |  | … |
| **5** | **Assessments** |  |  |
| 5.1 | write-up project report (20/4) | 6 | Project Report |
| 5.2 | produce poster (23/2) | 0.5 | Poster |
|  | Demo and presentation | 0.5 |  |
|  |  |  |  |
| **TOTAL** | **Sum of total effort in weeks** | **28 week until 16/4**  **30.5** |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SECTION 4 - Time Plan for the proposed Project work** | | | | | | | | | | | | | |
| For each task identified in 3.1, please *shade* the weeks when you’ll be working on that task. You should also mark target milestones, outputs and key decision points. To shade a cell in MS Word, move the mouse to the top left of cell until the curser becomes an arrow pointing up, left click to select the cell and then right click and select ‘borders and shading’. Under the shading tab pick an appropriate grey colour and click ok. | | | | | | | | | | | | | |
| **Project stage** | **START DATE: 2/10/2017****Project Weeks 28** | | | | | | | | | | | | |
| 0-3 | 3-6 | 6-9 | 9-12 | 12-15 | 15-18 | 18-21 | 21-24 | 24-27 | 27-30 | 30-33 | 33-36 | 36-39 |
| 1 Background Research |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Research of alternatives |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 Analysis/Design |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design a GUI |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design Calendar |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design calendar data addition method |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design medication consumption timer. |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design graphs for the display of health metrics. |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design calorie counter. |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 Develop prototype. |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Bug fixes |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Decide on platform and language. |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 Testing, evaluation/validation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Unit testing |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Verification Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |
| System Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Performance Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Validation Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 Assessments |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Project Report |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Project Poster |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Demo and Presentation |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Risk Assessment Form**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assessment Reference No.** |  | **Area or activity assessed:** | **Final project.** |
| **Assessment date** |  |
| **Persons who may be affected by the activity (i.e. are at risk)** | **Finley Williams** |

**SECTION 1: Identify Hazards -** *Consider the activity or work area and identify if any of the hazards listed below are significant (tick the boxes that apply).*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Fall of person (from work at height) |  |  | Lighting levels |  |  | Use of portable tools / equipment |  |  | Vehicles / driving at work |  |  | Hazardous fumes,  chemicals, dust |  |  | Occupational stress |  |
|  | Fall of objects |  |  | Heating & ventilation |  |  | Fixed machinery or lifting equipment |  |  | Outdoor work / extreme weather |  |  | Hazardous biological agent |  |  | Violence to staff / verbal assault |  |
|  | Slips, Trips & Housekeeping |  |  | Layout , storage, space, obstructions |  |  | Pressure vessels |  |  | Fieldtrips / field work |  |  | Confined space / asphyxiation risk |  |  | Work with animals |  |
|  | Manual handling operations |  |  | Welfare facilities |  |  | Noise or Vibration |  |  | Radiation sources |  |  | Condition of Buildings & glazing |  |  | Lone working / work out of hours |  |
| 1. **55** | Display screen equipment |  |  | Electrical Equipment |  |  | Fire hazards & flammable material |  |  | Work with lasers |  |  | Food preparation |  |  | Other(s) - specify |  |

**SECTION 2: Risk Controls** *- For each hazard identified in Section 1, complete Section 2.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Hazard No**. | Hazard Description | Existing controls to reduce risk | **Risk Level** (tick one) | | | Further action needed to reduce risks |
|  | High | Med | Low | *(provide timescales and initials of person responsible)* |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Name of Assessor(s)** | |  | **SIGNED** | | | |
| **Review date** | |  |

|  |  |  |
| --- | --- | --- |
| **Health and Safety Risk Assessments** – continuation sheet | **Assessment Reference No** |  |
|  | **Continuation sheet number:** |  |

**SECTION 2 continued: Risk Controls**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hazard No**. | Hazard Description | | Existing controls to reduce risk | **Risk Level** (tick one) | | | | Further action needed to reduce risks |
|  | High | | Med | Low | *(provide timescales and initials of person responsible for action)* |
|  |  | |  |  | |  |  |  |
|  |  | |  |  | |  |  |  |
|  |  | |  |  | |  |  |  |
| **Name of Assessor(s)** | |  | | | **SIGNED** | | | |
| **Review date** | |  | | |

## Appendix 3: Logbook

|  |  |
| --- | --- |
| Date | Work Done |
| 03/1/18 | Creation of Tabbing interface. |
| 27/2/18 | Initial commit to github. |
| 03/3/18 | Prototype input screen (PoC) created. |
| 28/3/18 | Embedded ViewPager within Tab B. |
| 06/4/18 | Fixed both viewpagers. Now both pages turn. |
| 09/4/18 | Created Logos for current list of symptoms/inputs. |
| 10/4/18 | Created fragment for each input page, updated viewpager and created more logos. |
| 15/4/18 | Implement alarmreceiver and media player for alarm. |
| 16/4/18 | Alarms intents now functional. Alarms now can be set and go off within minute of set time.  Android Graph implemented. Date replaced in the X axis. |
| 10/5/18 | Graphs created in Tab A.  Tab B inputs now record data to .csv’s |
| 11/5/18 | Implements zooming/panning + multiple chose able lines. |
| 12/5/18 | Replaced 4 button system with a circular seeker wheel. |
| 21/5/18 | Fixed strings and input fields. |
| 22/5/18 | Increased the coverage of the inputs fields. (Created new logos, fragments etc) |
| 25/5/18 | Redone Alarm UI  You can now click the logo to navigate to the other fragments page.  Alarm now remembers set time. |
| 26/5/18 | Help text button gives useful advice. |
| 27/5/18 | Recommendation engine now suggests when you may have a problem. |

