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](#_Toc515754615)

[2. ACKNOWLEDGEMENTS 3](#_Toc515754616)

[CONTENTS 4](#_Toc515754617)

[3. GLOSSARY OF TERMS AND ABBREVIATIONS 6](#_Toc515754618)

[4. INTRODUCTION 7](#_Toc515754619)

[5. PROBLEMS ARTICULATION 8](#_Toc515754620)

[5.1 Situation as is. 8](#_Toc515754621)

[5.1.1 Situation A (Pen and Paper) 8](#_Toc515754622)

[5.1.2 Situation B (Non-specific apps) 8](#_Toc515754623)

[5.1 Problem statement and Justification. 9](#_Toc515754624)

[5.2 Key stakeholders and their concerns. 9](#_Toc515754625)

[5.2.1 Diagnosed hypothyroid patients 9](#_Toc515754626)

[5.2.2 Suspect hypothyroid patients 9](#_Toc515754627)

[5.4 Technical specification. 10](#_Toc515754628)

[5.4.1 Requirements 10](#_Toc515754629)

[5.4.2 Constraints 10](#_Toc515754630)

[5.5 Potential situations to be. 10](#_Toc515754631)

[5.5.1 Web-app 10](#_Toc515754632)

[5.5.2 Mobile app 11](#_Toc515754633)

[6. LITERATURE REVIEW 12](#_Toc515754634)

[6.1 Medical Background Research 12](#_Toc515754635)

[6.1.1 Symptom Tracking 12](#_Toc515754636)

[6.1.2 Hypothyroidism screening and symptom prediction. 13](#_Toc515754637)

[6.1.3 Barriers to entry for target audience? 14](#_Toc515754638)

[6.2 Existing solutions 15](#_Toc515754639)

[6.2.1 Butterfly [12] 15](#_Toc515754640)

[6.2.2 MySymptoms [13] 16](#_Toc515754641)

[6.2.3 Thyroid Assistant [14] 16](#_Toc515754642)

[6.3 Literature review conclusion 17](#_Toc515754643)

[7. THE SOLUTION APPROACH 18](#_Toc515754644)

[7.1 Platform decided upon 18](#_Toc515754645)

[7.2 Programming languages used 18](#_Toc515754646)

[7.3 Integrated development environment chosen 19](#_Toc515754647)

[7.4 Open source libraries used 19](#_Toc515754648)

[7.4.1 Bullyboo’s circle seekbar 19](#_Toc515754649)

[7.4.2 OpenCSV 19](#_Toc515754650)

[7.4.3 Android GraphView 19](#_Toc515754651)

[7.5 Level of prediction 20](#_Toc515754652)

[7.6 Development process 20](#_Toc515754653)

[7.7 Version control 20](#_Toc515754654)

[8. IMPLEMENTATION 20](#_Toc515754655)

[8.1 Design 20](#_Toc515754656)

[8.2 Tab page viewer 21](#_Toc515754657)

[8.2.1 Design 21](#_Toc515754658)

[8.2.2 Implementation 21](#_Toc515754659)

[8.3 Dashboard (Tab A) 23](#_Toc515754660)

[8.3.1 Initial Design 23](#_Toc515754661)

[8.3.2 Implemented Final Product 24](#_Toc515754662)

[8.3.3 Implementation 25](#_Toc515754663)

[8.4 Intakes (Tab B) Design 29](#_Toc515754664)

[8.4.1 Initial Design 29](#_Toc515754665)

[8.4.2 Refined Design 31](#_Toc515754666)

[8.4.3 Implemented Final Input Tab 32](#_Toc515754667)

[8.4.4 Implementation 33](#_Toc515754668)

[8.5 Labs (Tab C) Design 35](#_Toc515754669)

[8.5.1 Initial Design 35](#_Toc515754670)

[8.5.2 Implemented Final Product 36](#_Toc515754671)

[8.5.3 Implementation 36](#_Toc515754672)

[9. TESTING: VERIFICATION AND VALIDATION 37](#_Toc515754673)

[10. DISCUSSION: CONTRIBUTION AND REFLECTION. 37](#_Toc515754674)

[11. SOCIAL, LEGAL, HEALTH, SAFETY AND ETHICAL ISSUES 37](#_Toc515754675)

[12. CONCLUSION AND FURTHER IMPROVEMENTS 37](#_Toc515754676)

[REFERENCES 37](#_Toc515754677)

[APPENDICES 38](#_Toc515754678)

# 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 your 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 you have thyroid issues via an easily installed app seems like a useful app to exist.

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

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

* To provide a method of easily recording your 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 with you and whenever you notice something you feel may be significant just writing it down.

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 your symptoms.
* Examining the data visually comparing and symptoms against each other symptoms or other trigger factors can be difficult. You would have to draw a graph or transfer the data into something like excel.
* There is little context to the data making consistency and quantifying the data very difficult. If you recorded a headache 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 you are likely always carrying a phone or laptop on you at all times.

In regards to pure data collection there is no issues with using pen and paper however whenever you want to do anything more with the data other than simply review it becomes significantly more problematic.

There is one key advantage to pen and paper which is, the extremely low cost of pen and paper, you are likely to have these in your house already, meaning you can start recording almost as soon as you’d like.

### 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 you want.

A final issue which generic apps have is if you have been diagnosed as a sufferer of hypothyroidism it may be distressing or annoying to have various medical apps constantly tell you that you may be at risk of other diseases when you 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 you have recorded chronologically when discussing the issues with a doctor would be an improvement as the current pen and paper solution means you will need to manually order all the data and cannot quickly jump to specific parts of the data you wish to look at.

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 as you could log on using a web browser on which you could then enter all your issues, symptoms, food intakes etc.

You would then be able to view the data chronologically in the browser which would allow you to see how you 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 you to enter all your issues such as symptoms, food intakes and other problems.

It would also allow you to look back chronologically in app to see how you’re 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 you 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 you can retain data in storage rather than sending them to our server which would allow you to know that your 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 you are 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 you have to consider 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 your 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 requirement of giving personal data to a third party which may not have a sufficient levels of security to stop anyone using your data. It is extremely unlikely that this level of model creation can be done on your 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 your adherence to the schedule. It also has “over 275” symptoms built in for tracking so that you can track any number of symptoms you may have.

A good feature it has is keeping track of your lab work all in one place so that you 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 you can be reminded to record data, take medication or anything else that you think you 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 you to track intake of food, medication, drink, sleep, symptoms and other conditions to track how they affect you.

One thing I think it does well is that it tracks what symptoms you have and what you have them after and attempts to predict what will cause you problems. Meaning that it works with you to work out what causes problems which if it correctly guesses the problem you can remove the substance or issue and see if the symptoms disappears and then bring it up with your doctor 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 you to track your symptoms day by day through the use of a diary. You 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 you haven’t entered before you have to navigate four different submenus and then type in what you are entering. Whilst this does allow for maximum customizability and allows you 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 you to record as much as possible to make effective use of it and if you have 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 you much about what happened that day only that something happened. You have to click on the logo for any useful data from that particular day. You 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 you will need to read for several minutes to finish one section of their 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 not big surprise as you do not want 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 you 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 though put into this app shows as reviewing how you’ve been for a period of time is extremely difficult. However it does present some interesting ideas 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. This was for a multitude of reasons 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 some version of android which means that it is extremely likely that any potential user will happen to own and use regularly an android phone.

The second reason for the decision to use android is that the author of this document has a large amount of experience in programming in Java which makes this an obvious choice as this 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, 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 within 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 Java 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 had 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 seekbar in android studio however the key difference for 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 the seekbar in 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 your own 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 to 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.

Their 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].

# 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 recording and predicting their symptoms there are a relatively few methods of which are greatly different from each other. Most of the changes you are likely to see will be encountered in the choice made when designing the system.

## 8.2 Tab page viewer

### 8.2.1 Design

As you will see 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 you a long term view of the user’s lab results as well as setting alarms to take medication.)

### 8.2.2 Implementation

The tab layout is implement using a modified version of the standard android view pager as you can see below in .

**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 : 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 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 : 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 .

As you can see 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 you 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.

## 8.3 Dashboard (Tab A)

### 8.3.1 Initial Design

In below you can see my initial designs for the dashboard of the application.

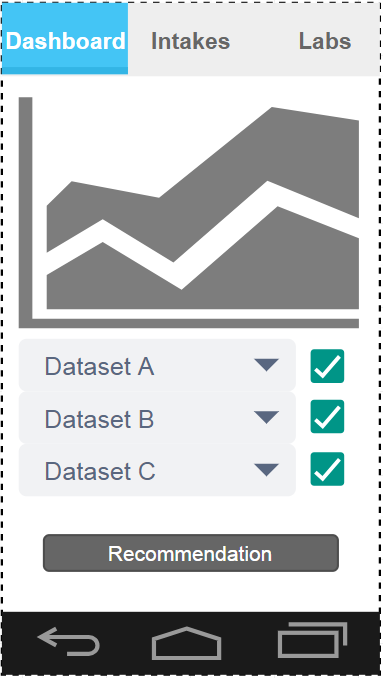


Figure 3: Dashboard Tab Initial Design

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

It is scrollable and zoom able which allows the user to see a 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 you to select the data set, either symptoms or intakes that you 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.

### 8.3.2 Implemented Final Product

Below in is the implemented final design.

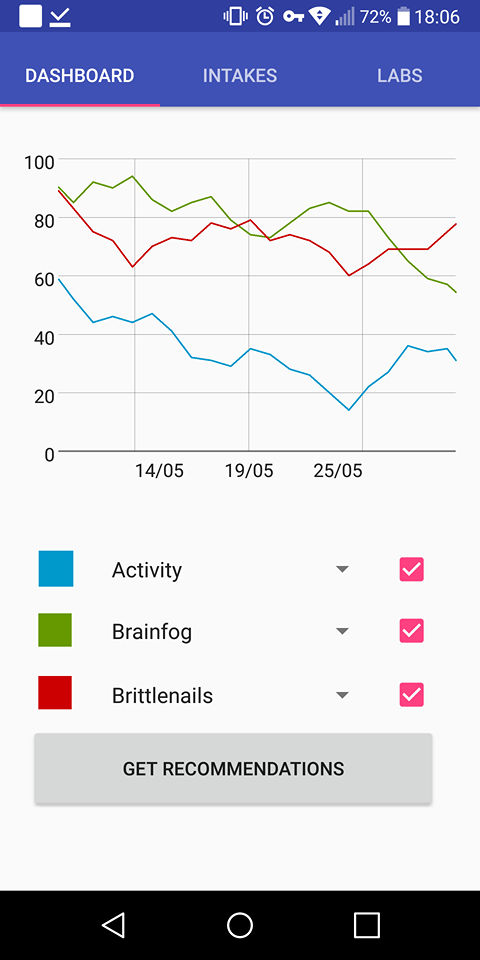


Figure : Implemented Dashboard Tab

As you can see most of the features have been carried through from the design.

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 .

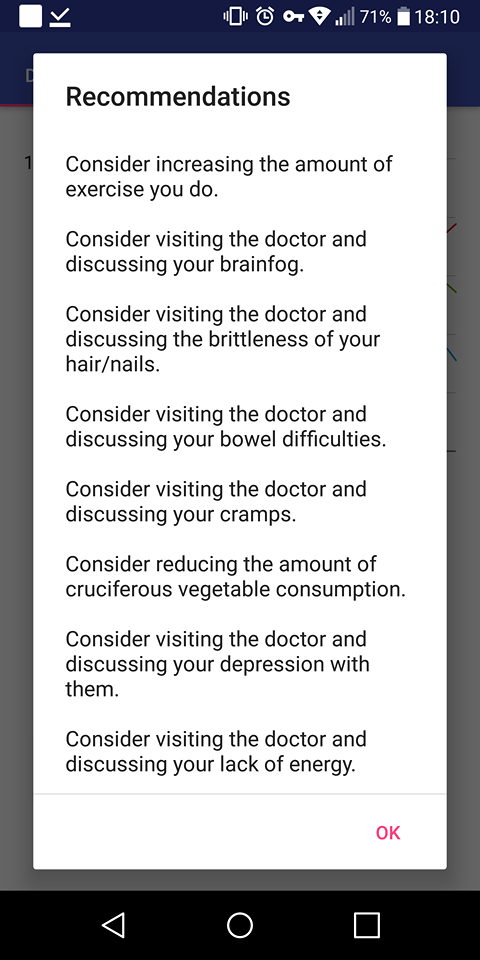


Figure : Recommendation Pop Up

In a real life example this popup would likely have less recommendations as this example is using randomly generated data, so has a lot of recommendations. However the functionality is very clear 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.

### 8.3.3 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 a 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 datapoint 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 .

**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 : 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 : 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.

#### 8.3.3.1 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 : A snippet of the code from the recommendation case statement.

Above in , 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.

## 8.4 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.

### 8.4.1 Initial Design

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

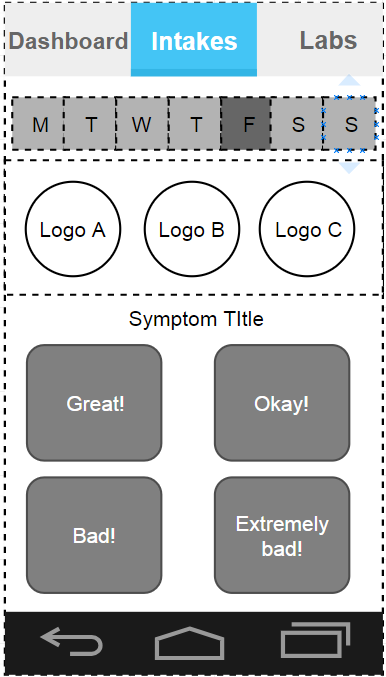


Figure : Input Tab Initial Design

The first section is the week overview. This section shows you the current week you are 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 you to quickly navigate to the required symptom. You can then tap the icon and it will move you to that input field.

The third and final section is the data entry section. This will consist of four buttons which will allow you 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 you can only choose one of four values when entering 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.

### 8.4.2 Refined Design

Below in is the redesigned intakes tab.

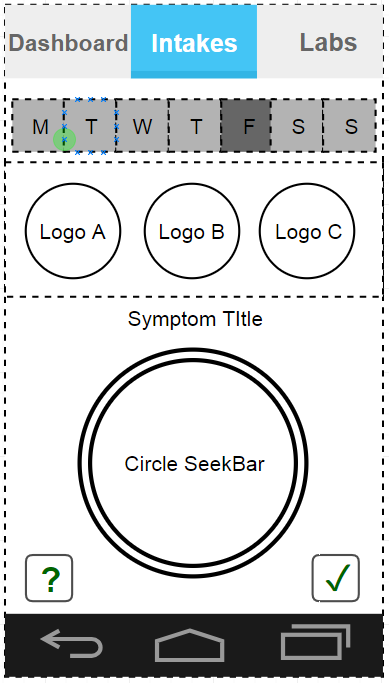


Figure : Input Tab Improved Design

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.

### 8.4.3 Implemented Final Input Tab

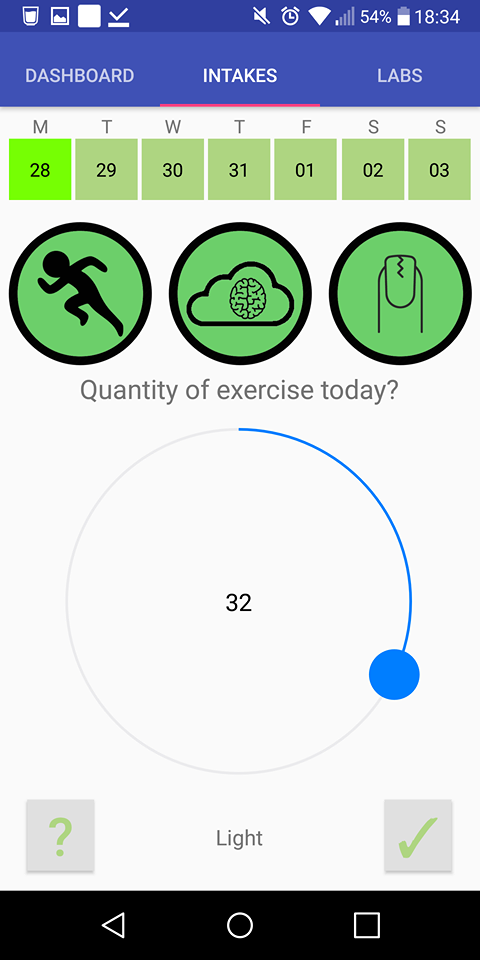


Figure : Implemented Intake Tab

### 8.4.4 Implementation

#### 8.4.4.1 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 : Snippet from Tab\_B.Java responsible for the week's overview.

The code shown above in Figure 12, 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 : SetPage method and Tab\_B.java's onCreate and listeners.

In Figure 13 above, you can see 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

* Help text

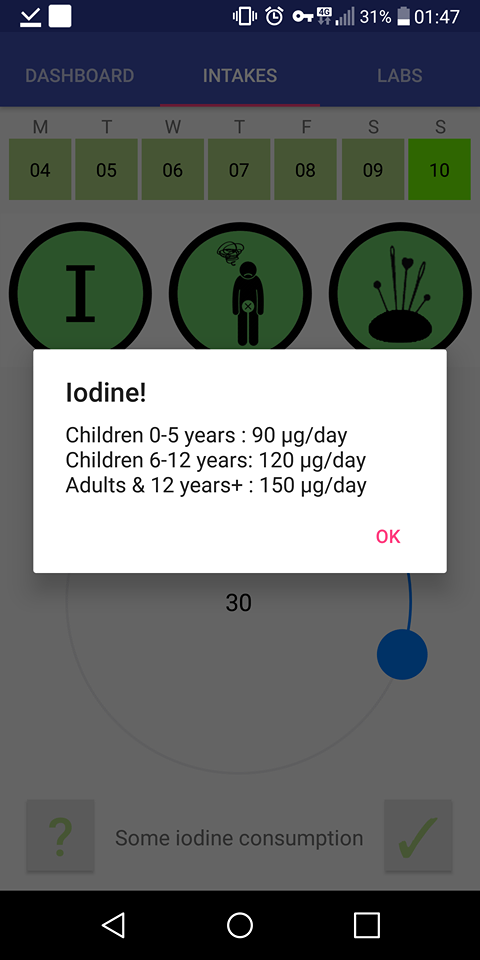


Figure : Example help text for iodine input

* CSV writing
* “what your level means text” – Text under the circle spinner

#### 8.4.4.4 High level navigation fragments.

* Logo’s made by hand
* Function calls

## 8.5 Labs (Tab C) Design

### 8.5.1 Initial Design

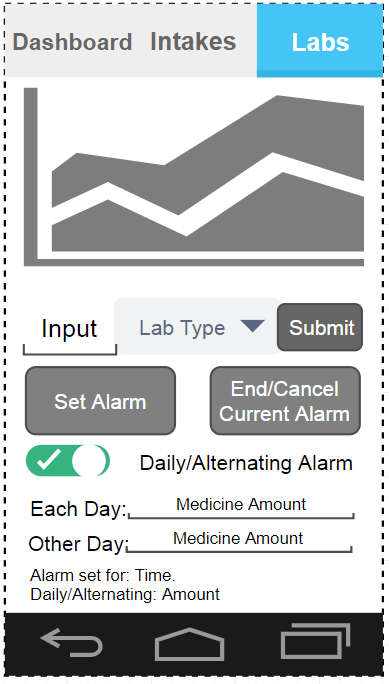


Figure : Lab Tab Initial Design

### 8.5.2 Implemented Final Product

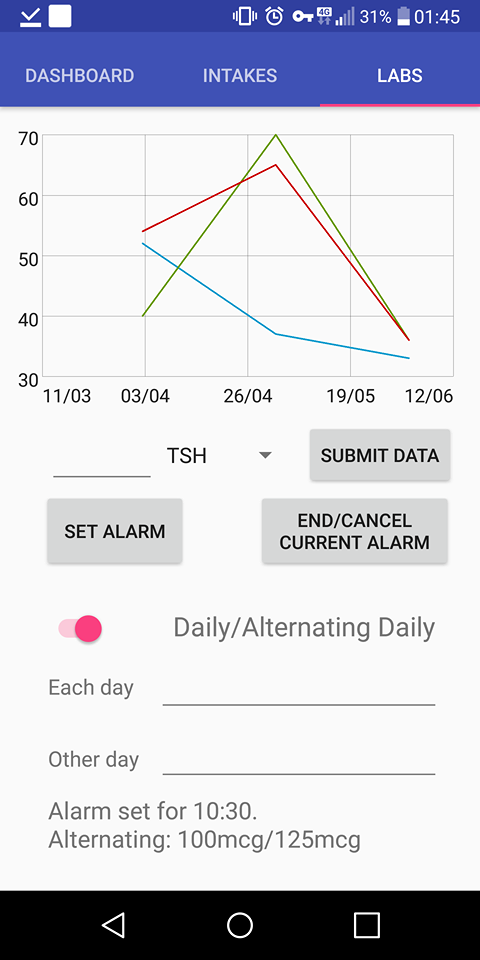


Figure : Implemented Labs Tab

### 8.5.3 Implementation

#### 8.5.3.1 Time picker Dialog

Below in Figure 17

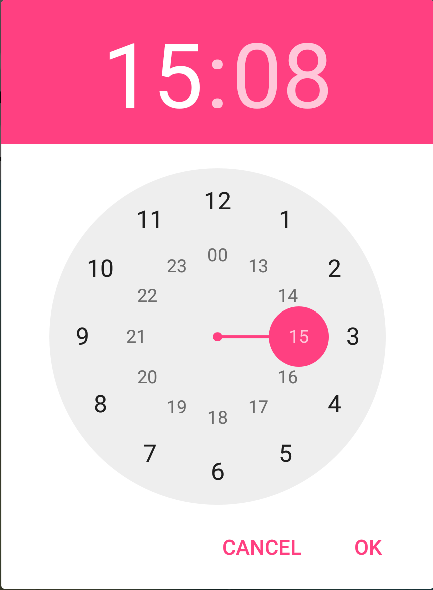


Figure 17: The Time Dialog Picker Pop up.

#### 8.5.3.2 AlarmReciever

#### 8.5.3.3 AlarmPlayer

#### 8.5.3.4 GraphView

# 9. TESTING: VERIFICATION AND VALIDATION

# 10. DISCUSSION: CONTRIBUTION AND REFLECTION.

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

# 12. CONCLUSION AND FURTHER IMPROVEMENTS

# 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