National College of Ireland

BSc. (Honours) in Computing

Software Development

2020/2021

Joey Tatú

15015556

joey.tatu@student.ncirl.ie

Automatic Retinopathy Detection Using Digital Image Processing via a Smart Device

Technical Report

Table of Contents

Annotations

App	Application (e.g. on Android or iPhone)
Retinopathy	Damage to the retina of the eye
Diabetes	Diabetes mellitus
DR	Diabetic retinopathy
DME	Diabetic macular edema
CNN	Convolution neural networks
Ophthalmoscope	Device used by eye doctors to scan the internal of the eye.
Intent	A page within an Android app
Firebase	Google Firebase Realtime Database and Google Firebase Storage

Executive Summary

The purpose of this report is to provide an overview of the project, to explain the main functional and non-functional requirements and to discuss the overall look of the project. Beyond these, how the project is implemented is also considered, as well as explaining the process of testing and evaluation.

The main focus of this project is to detect diabetic retinopathy by using an app on a smart device. Future versions of the app are expected to be able to be connected to a Raspberry Pi. This Raspberry Pi will be connected to a macro lens camera that will be able to take close-up photos of the inside of the eye.

Other future features include connecting to a database (such as a doctor surgery or hospital records). This will allow the generated results from the app to be sent to the database. This database can be used in connection to the patient's online records to keep all their data in one place. Future versions of the app will also be able to update the medical information from the app such as the patient's medical information such as their weight, blood sugars, other illnesses and what medications the patient is taking.

The results of scanning a patient's eye will be able to determine if a patient has diabetic retinopathy (DR) by determining what DR stage the patient is at, with stage I having no DR, stage IV having severe DR and stage V expected to have DR in the future. If stage I is detected, no action would be required. With stages II – V, the app would automatically update the patient's record. This is done by comparing the uploaded image of the eye with a TensorFlow Lite generated model.

The conclusion to the project is that a preliminary test can be completed by a medical professional on a patient before and if further testing of the patient's eyes is needed.

1.0 Introduction

1.1. Background

The project began as a vegan health and fitness app. A proposed feature for this was to take a photo of a person's eye and to determine whether the whites of the eye (sclera) was a shade of yellow. The more yellow the sclera, the more probability of an issue. This version of the project was too simplistic. It's common knowledge that these features had been implemented before in a similar fashion into many different apps. To do a health app would be repetition.

The original idea of checking the sclera was thought of because a family member of the author had sallow skin and yellowing sclera for many years due to previously undiagnosed gut issues. Taking this into consideration, scholarly articles were examined. Some reports had discussed diabetic retinopathy. This is how the idea of creating an app for diabetic retinopathy. creating the idea of scanning for diabetic retinopathy. During researching scholarly articles on sclera heath, diabetic retinopathy (DR) was discovered and detecting this was added to the project, abandoning the vegan health and fitness app.

1.2. Aims

An eye doctor uses an ophthalmoscope or a diabetic retinopathy machine to visually and manually complete an diabetic retinopathy scan. The doctor would need to manually look for issues with the eye. Human error can occur where an issue with the eye can be bypassed or a false positive can occur. The results of the eye exam are manually entered into the eye doctor's computer into the patient's file.

This project aims to not only make this simpler, but to auto-detect issues in the eye, record it and add it to a patient's file, all automatically. The severity or "stage" of DR is then suggested by comparing hundreds of images, via a TensorFlow Lite model, with the new, uploaded image. The eye doctor can create an outcome or treatment, based on these results. It also aims to be able to record many eye exams and then be able to upload these to the patient's file via mobile Internet or Wi-Fi. This test can be completed by a medical professional, not necessarily an eye doctor. This can free up more time for eye doctors to treat many more patients. An example of this in use could be in poverty-stricken areas where time and scanning large number of patients is of the essence.

1.3. Technology

The project is being developed in Android Studio and a future version will use a Raspberry Pi and a camera with a macro lens to obtain the eye test images. Image processing with convolutional neural network (CNN) models will also be used.

Originally, Google LeNet (formally InceptionNet), AlexNet and VGG16 has been considered. These models all have their advantages and disadvantages.

LeNet is specific in categorising the DR stages. AlexNet is a model that, in comparison to older CNNs, has significantly reduced the error rate with setting the result of non-visible neurons to nil, with a 50/50 probability.

VGG16 is more advanced than AlexNet. It does what AlexNet does but adds mire viewable layers and filters for a result with better clarity. LeNet advances AlexNet and VGG16 even further with a more reduced error rate with more layers and filters.

(Wang, et al., 2019) (Rosebrock, 2018)

However, after long hours on doing research on these, no coding examples were found, with little or no mention of how to implement these. TensorFlow Lite was then considered. TensorFlow Lite allows the developer to upload hundreds or thousands of images to generate a model. (Sethi, 2020) The model was created with TensorFlow Lite using Google Colab.

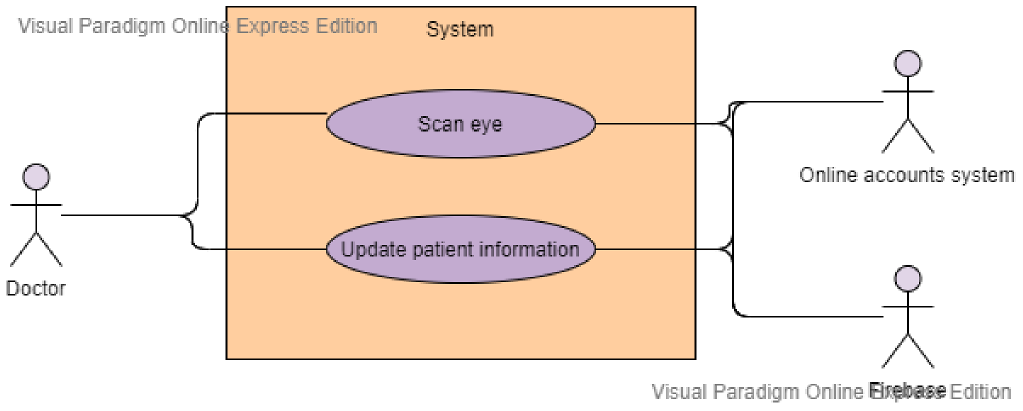
2.0 System

2.1. Requirements

2.1.1. Functional Requirements

This section compiles the functional requirements in ranked order. Functional requirements explain what the project is to overcome.

2.1.1.1.1. Use Case Diagram


Visual Paradigm Online Express Edition
System
Scan eye
Update patient information
Doctor
Online accounts system
Firebase
Visual Paradigm Online Express Edition

2.1.1.1.2. Requirement 1: Scanning eye

This use case describes how the eye in scanned within the app.

2.1.1.1.3. Description & Priority

This is the focus of the app and project. This use case explains to process of how the eye doctor and system scans the eye.

2.1.1.1.4. Use Case

Unique ID: scanEye

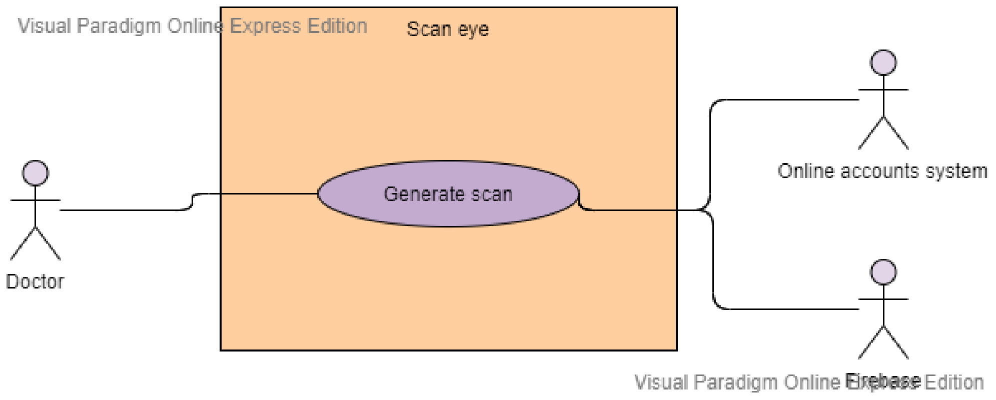
Scope

The scope of this use case is for the eye doctor to scan a patient’s eye, to diagnose a patient with DR.

Description

This use case describes the how an eye doctor scans the eye to diagnose DR in a patient which auto-detects DR and to save the results to a patient's file.

Use Case Diagram


Visual Paradigm Online Express Edition
Scan eye
Generate scan
Doctor
Online accounts system
Firebase
Visual Paradigm Online Express Edition

Flow Description

Precondition

- The device is powered on
- The device is connected to the Internet
- Doctors/eye doctors (user) account controlled by external accounts system (e.g. Microsoft 365 or similar)
- The app is installed on the device
- The Raspberry Pi is connected to the device
- The macro lens camera is connected to the Raspberry Pi
- The authentication and database are connected
- The app is defined as the System
- The device is defined as the external system.

Activation

This use case starts when an eye doctor opens the app.

Main flow

1. The eye doctor opens the app on the device.
2. The system checks that it can access the Internet by trying to access Google.ie <See A1>
3. The system checks that is can access an example patient file in the database <See A2>
4. The system checks the eye doctor is signed in <See A3 >
5. The eye doctor selects the "Start scan" button
6. The system requests the patient ID number.
7. The eye doctor inputs the patient ID and selects "Submit"
8. The system connects to the Raspberry Pi <See E1>
9. The Raspberry Pi starts the camera. (external, request from system)

10. The system shows the live camera on the screen.
11. The eye doctor moves close to the patient.
12. The eye doctor points the camera to the patient's iris in their eye from 1 – 2 metres.
13. The eye doctor selects "Take photo".
14. The Raspberry Pi records the image.
15. The Raspberry Pi sends the image to the app on the device.
16. The system displays the taken image.
17. The eye doctor selects "Save"
18. The eye doctor selects if the eye is the left or right eye.
19. The system uses a combination of AlenNet, VGGG16 and LeNet to process the image and detect anomalies in the eye.
20. The system puts a grid over the image.
21. The system marks where the anomalies occur with a coloured square around the anomality. (Marked colour is not yellow or red as this may blend with the colour of the eye).
22. The system gets the coordinates of the anomalies and colourises them to match where they are in the image.
23. The system gets the date and time.
24. The system creates a text file and inputs the date, time and results.
25. The system connects to the database
26. The system adds the image and text file to the patient's file in the database using the patient ID as reference.
27. The system displays a message saying the image is saved <See E2>
28. The eye doctor closes the app.

Alternate flow

A1: <No internet on device>

3. The device cannot access Google.ie
4. The system displays a message for the eye doctor to check the internet settings on the device.
   (External: The eye doctor turns on the Wi-Fi and/or mobile internet)
   <Returns to number 2 in Main Flow>

A2: <Cannot access the database>

4. The system connects to the local database on the device
5. Every 30 minutes, the system will try to send the new data to the online database until it succeeds.
6. When it succeeds, the local database is destroyed.
7. <Returns to number 3 in Main Flow>

A3: <Cannot access external accounts system>

5. The system connects to a local account instead
6. Every 30 minutes, the system will try to access the external accounts system until it succeeds.
7. The eye doctors then signs in to their account and any new information is sent to the online database <see A2>
   <Returns to number 4 in Main Flow>

Exceptional flow

E1 : <System cannot connect to Raspberry Pi>

8. The system displays a message saying the Raspberry Pi is not connected Without the Raspberry Pi, the system cannot continue.
9. The system displays an "Exit" button.
10. The eye doctor selects button and the system closes.

Termination

This use case is terminated when the database successfully receives the new information.

Post condition

The system goes into a wait state

2.1.1.2. Requirement 2: Update patient information

This use case describes how the eye doctor or medical professional updates the patient's record.

2.1.1.2.1. Description & Priority

This use case describes how the eye doctor or medical professional updates the patient's medical record in the app that will sync with the patient's files in the Cloud (online storage).

2.1.1.2.2. Use Case

Unique ID: updateRecord

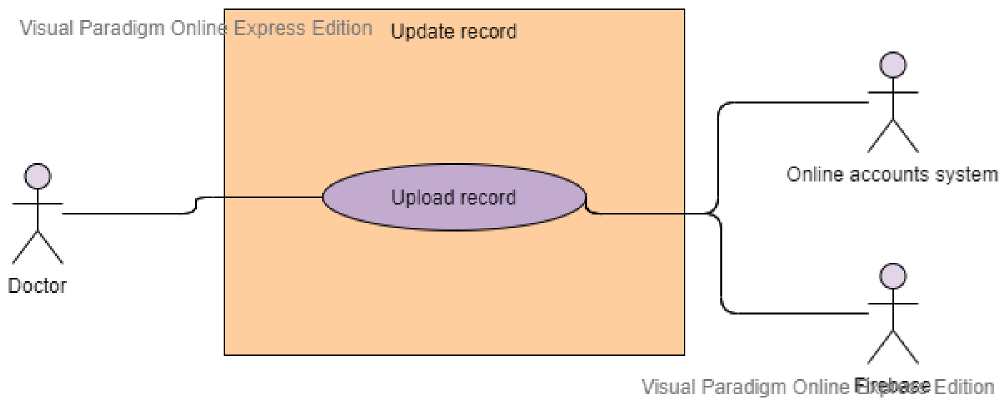
Scope

The scope of this use case is for the eye doctor or medical professional to update the patient's record.

Description

This use case describes the how an eye doctor or medical professional updates the patient's record from the app.

Use Case Diagram


Visual Paradigm Online Express Edition
Update record
Upload record
Doctor
Online accounts system
Firebase
Visual Paradigm Online Express Edition

Flow Description

Precondition

- The device is powered on
- The device is connected to the Internet
- Doctors/eye doctors (user) account controlled by external accounts system (e.g. Microsoft 365 or similar)
- The app is installed on the device
- The authentication and database are connected
- The app is defined as the System
- The device is defined as the external system.
- The doctor in this use case is either the eye doctor or a medical professional.

Activation

This use case starts when an eye doctor opens the app.

Main flow

1. The eye doctor opens the app on the device.
2. The system checks that it can access the Internet by trying to access Google.ie <See A1>
3. The system checks that is can access an example patient file in the database <See A2>
4. The system checks the eye doctor is signed in <See A3>
5. The eye doctor selects the “Start” button
6. The system requests the patient ID number.
7. The doctor inputs the patient ID and selects “Submit”.
8. The system retrieves the patient file from the online database
9. The system stores the database in a local temporary database.
10. The system displays the patient information.
11. The doctor goes through the information, such as age, weight, height and edits them accordingly. <See A4>
12. The doctor selects “New note”. <See A5>
13. The system obtains the device’s date and time.
14. The system inputs the date and time into the new note.

15. The doctor edits the new note accordingly with what the patient has said such as if they are having headaches or if the doctor suggests medication.
16. The doctor selects "Save".
17. The doctor selects Upload.
18. The system adds the updated information to the local database.
19. The system connects to the database and uploads the data. <See E1>
20. The system deletes the local database securely.
21. The system displays a message that the upload is successful.

Alternate flow

A1: <No internet on device>

5. The device cannot access Google.ie
6. The system displays a message for the eye doctor to check the internet settings on the device.
   (External: The doctor turns on the Wi-Fi and/or mobile internet)
   <Returns to number 2 in Main Flow>

A2: <Cannot access the database>

8. The system connects to the local database on the device
9. Every 30 minutes, the system will try to send the new data to the online database until it succeeds.
10. When it succeeds, the local database is destroyed.
11. <Returns to number 3 in Main Flow>

A3: <Cannot access external accounts system>

11. The system connects to a local account instead
12. Every 30 minutes, the system will try to access the external accounts system until it succeeds.
13. The doctor then signs into their account and any new information is sent to the online database <see A2>
    <Returns to number 4 in Main Flow>

A4: <Patient information: No edit needed>

<Returns to number 12 in Main Flow>

A5: <No new note>

<Returns to number 20 in Main Flow>

Exceptional flow

E1: <System cannot connect to online database>

21. The system attempts to send the data every 30 minutes until it is successful
22. <Returns to number 20 in Main Flow>

Termination

This use case is terminated when the database successfully receives the new information.

Post condition

The system goes into a wait state

2.1.2. Data Requirements

All data communication must be encrypted. Raw data is not permitted to be communicated to another device or system.

All passwords must be more than 8 characters with at least one uppercase letter, one lowercase letter, one number and one special character. A check for most used passwords will be implemented. If a password matches one of these simple passwords, it will not be accepted.

All Firebase Databases are encrypted too to ensure the information is secure.

2.1.3. User Requirements

Users should have a general knowledge of smart phones and tablets to use this app. Users should be able to update a social media account, send a message of the likes of WhatsApp and be able to perform a search on the smart device’s browser.

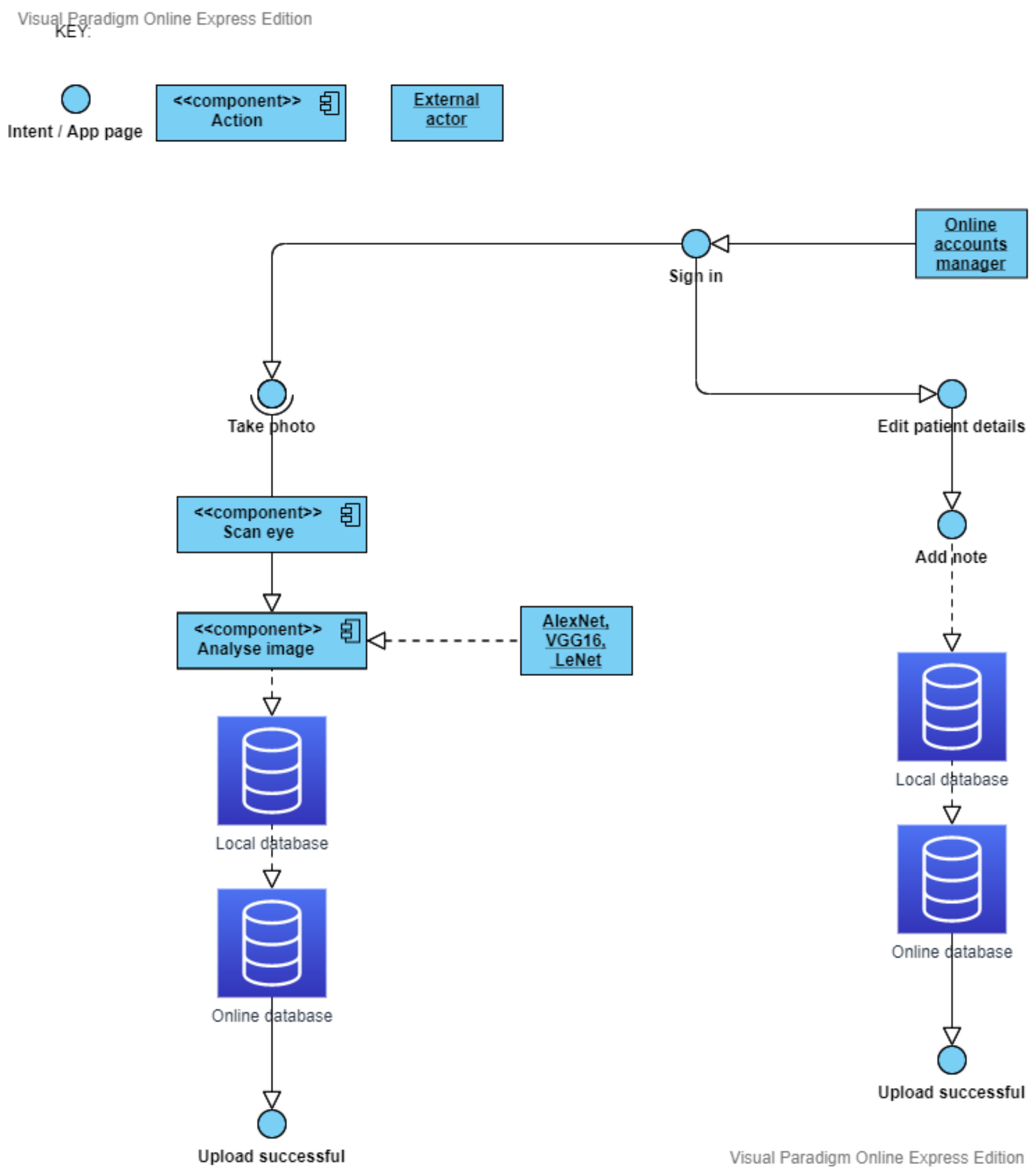
2.1.4. Usability Requirements

The app itself will be straightforward and simple to use.

Accessibility feature are included. These include making the text larger or smaller, using neutral colours in standard mode and high-contrast mode for those with visual impairments.

A text-to-speech feature will also be implemented, with sensitive data requiring a confirmation (e.g. tap the button/area again) before being read out.

2.2. Design & Architecture



2.3. Implementation

Input from User

When the user is inputting text, a text field will be used. This is recognised by the system using the variable name.

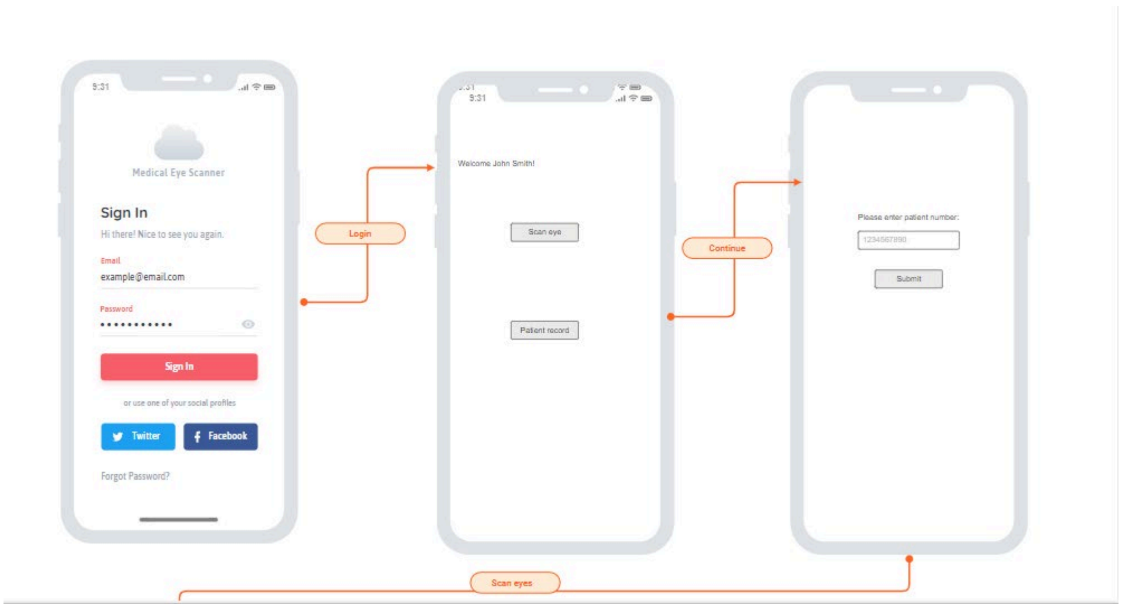
Database connections

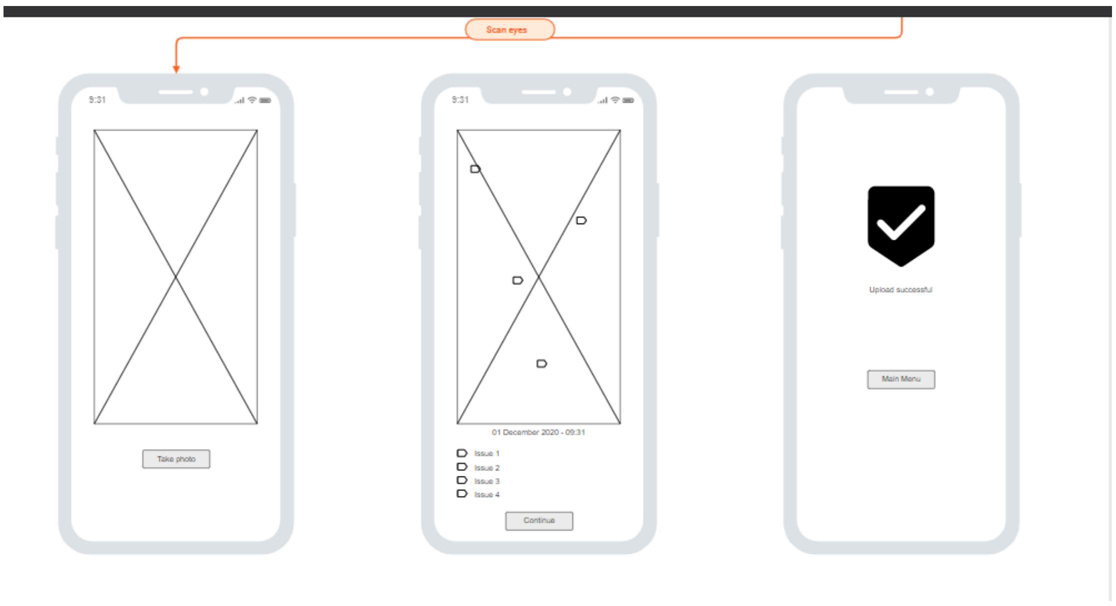
An example of how tables in the database would be used is the Records tables.

For the doctor to see the patient's record, the system will read the information in the database using API and JSON (a machine-readable format). This ensures (e.g.) all profile layouts are the same for each User.

2.4. Graphical User Interface (GUI)

These GUI examples have been created before the app creation.





2.5. Testing

Eye scanning: This will be tested with eye exam images from online. One group will have no issues, and the other group will have issues. If these are correctly identified by the system, the test passes.

Update patient record: A test patient record will be created. If the test record can be successfully edited and updated, the test passes.

All testing has been completed by the developer.

<u>**Usability testing:**</u>

"Visibility of system status"

With this test, the system would ensure the doctors are aware of what is happening on the app. A response with an acceptable period is required. (Nielsen, 1994)

- The app informs the user what it is doing, but without being overly complicated.

"Match between system and the real world"

Regarding language, it should be simplistic and to a level of the artists and clients understanding. Technical jargon should be avoided. (Nielsen, 1994)

- The app contains Basic and Learning English only. Difficult or words that are not well known have been avoided.

"User control and freedom"

Sometimes a doctor may access a page and edit something they didn't want to. An "emergency exit" to the likes of the home page is to be implemented. Buttons that revert and obvert the user's action will also be implemented. (Nielsen, 1994)

- The Sign Out and Revoke access buttons act on behalf of this. Simply pressing the back button on the device does this as well.

"Consistency and standards"

Clear, understandable words will be used throughout the app. An example such as "edit" and "modify" generally means the same thing. Only one of these will be used (Edit). This will be implanted throughout the app. (Nielsen, 1994)

- Consistency of words are used throughout. Facebook uses "Log out" instead of "Sign out". However, in the app, "Sign out" has been used.

"Error prevention" and "Recognition rather than recall"

It may not always be possible to avoid error messages. However, a better designed app will avoid making the error at all. Forms that are in the website will be checked before they are sent to the database. The app will also provide "tips and tricks" to assist the doctor in using the app. (Nielsen, 1994)

It's important for the doctor not to recall data from one form to another. Such information will be either obtained from the database or stored in temporary files on the device.

- All images, buttons and input text forms are checked they have the required information before continuing.
- Examples:
    - The user's Google ID is checked it to see if it is present on the relevant page, and if it's not, the user is sent back to the login screen.
    - The Patient ID is needed to continue to upload or take an image. If this is not present, the user is taken to the patent ID form.
    - Both of these are stored in temporary files on the device and cleared when requested.

"Flexibility and efficiency of use"

An importance is to allow users who are new to the app to go through the app at their own pace. This would not be beneficial for a doctor who has visited the website many times. The app will accommodate both types of doctors. (Nielsen, 1994)

- The app is simple and straightforward to the user, but there is much going on behind the scenes.

"Aesthetic and minimalist design"

Text data must not be over complicated with unrelated data. Over-complicated text will be completely avoided on the app. (Nielsen, 1994)

"Help users [recognise], diagnose, and recover from errors"

If an error message is show, it needs to be simple and not have any code or complicated language. If possible, a way to resolve the issue should also be present. (Nielsen, 1994)

- An example of this if the patient ID needs to be four to eight numbers. If the user does not enter the right amount of numbers, an error shows stating same.

"Help and documentation"

A help section will be added to the app. The action that the doctor is trying to do should be clear and understandable, and not contain excessive text. (Nielsen, 1994)

2.6. Evaluation

The "end-user" testing will be completed by the developer. The developer will explore the app and see how everything feels as well as measure the usability, based on Nielson's "10 Usability Heuristics for User Interface Design" (1994).

Heuristic evaluation:

The developer has evaluated the app. The app is simple and straight-forward and any errors that do show are simple and easy to understand. The end-user may find the app a bit "clunky" as the login and inserting the patient ID could have been put on one page.

Performance Testing:

All performance testing has been done in Junit. All texts where done 5 times and the result below is the average.

The average speeds when using the app are:

- Login Activity to Patient Activity using Google login: 2.203 seconds. (excluding web browser activity)
- Login Activity to Patient Activity using Facebook login: 2.958 seconds. (excluding web browser activity)
- Patient Activity to Image Activity: 0.694 seconds.
- Image Activity – Camera Activity loading – 0.321 seconds.
- Image Activity – Gallery Activity loading – 0.503 seconds.

- Image Activity to Results Activity: Image sent through model and obtained results: 4.036 seconds.
- Results Activity – Submit to file (database) – 0.012 seconds.
- Results Activity – Send to doctor (email) loading – 0.394 seconds.
- Image Activity to Patient Activity – (Patient ID not present) – 0.102 seconds.
- Results Activity to Patient Activity – (Patient ID not present) – 0.134 seconds.
- Patient Activity to Login Activity – (User ID not present) – 0.523 seconds.
- Image Activity to Login Activity – (User ID not present) – 0.198 seconds.
- Results Activity to Login Activity – (User ID not present) – 0.209 seconds.
- Main Activity to Login Activity – (Not previously signed in) – 0.129 seconds
- Main Activity to Patient Activity – (Not previously signed in) – 0.235 seconds

Load and stress testing

The above was retested with 90% of the device's RAM active (excluding the app). The app ran slower. The results are as follows:

- Login Activity to Patient Activity using Google login: 4.029 seconds. (excluding web browser activity) **(+1.826 seconds)**
- Login Activity to Patient Activity using Facebook login: 4.920 seconds. (excluding web browser activity) **(+1.962 seconds)**
- Patient Activity to Image Activity: 1.938 seconds. **(+1.244 seconds)**
- Image Activity – Camera Activity loading – 1.289 seconds. **(+0.968 seconds)**
- Image Activity – Gallery Activity loading – 2.093 seconds. **(+1.590 seconds)**
- Image Activity to Results Activity: Image sent through model and obtained results: 6.983 seconds. **(+2.947 seconds)**
- Results Activity – Submit to file (database) – 1.091 seconds. **(+1.079 seconds)**
- Results Activity – Send to doctor (email) loading – 0.992 seconds. **(+0.598 seconds)**
- Image Activity to Patient Activity – (Patient ID not present) – 1.220 seconds. **(+1.118 seconds)**
- Results Activity to Patient Activity – (Patient ID not present) – 1.922 seconds. **(+1.788 seconds)**
- Patient Activity to Login Activity – (User ID not present) – 2.001 seconds. **(+1.478 seconds)**
- Image Activity to Login Activity – (User ID not present) – 1.899 seconds. **(+1.701 seconds)**
- Results Activity to Login Activity – (User ID not present) – 1.982 seconds. **(+1.773 seconds)**
- Main Activity to Login Activity – (Not previously signed in) – 1.632 seconds **(+1.503 seconds)**
- Main Activity to Patient Activity – (Not previously signed in) – 1.112 seconds **(+0.877 seconds)**

Acceptable increase is +2.000 seconds except for Image Activity to Results Activity which is +3.000 seconds.

Endurance Testing

A macro was created to automate tapping and going through the app from start to finish. This app ran for 24 hours, non-stop.

The app slowed down a slight amount. It was like the load and stress testing results but with a difference of ±0.500 seconds on average.

Spike testing

Spike testing was done similarly to the endurance testing, but with the device's RAM being increased to 90% for 10 minutes at 30-minute intervals.

The results were like the load and stress testing, but the results had a difference of ±0.125 seconds on average.

Volume testing

The Results page was duplicated into another app with that page only. Every 30 seconds, the Results page would send random text to represent the results to the database.

The developer created an app with a button that would click every 5 seconds to add a new patient with example data of their personal details, medical conditions, DR scan results, a long string of random text to represent notes. This was sent to the database until the database was full (unpaid allowance.

Both these ran for 9 hours and 34 minutes (Max data reached for free allowance after this time).

In the last 10 to 15 minutes before the database refused any more inputs, the Results page became slightly sluggish with a max increase of +10.797 seconds. This is an acceptable level, as the developer set +15 seconds as a limit.

3.0 Conclusions

The advantages of the project are that an eye doctor or a medical professional would be able to scan a person's eye and determine if there is an issue. This could free up an eye doctor's time, so that another medical professional could scan an eye without medical training. This would free up the eye doctor's time and they would only need to look at the confirmed cases or if a patient requests a second opinion.

4.0 Further Development or Research

If given more time and resources, this app could be developed to scan the likes of cuts and bruises and to record a person's temperature. The Raspberry Pi and macro camera would also be implemented.

5.0 References

Burggraf, C., 2020. *Is It Vegan?.* [Online]
Available at: https://play.google.com/store/apps/details?id=net.isitvegan.androidfree
[Accessed 5 November 2020].

HappyCow, 2020. *Find Vegan Restaurants & Vegetarian Food- HappyCow.* [Online]
Available at: https://play.google.com/store/apps/details?id=com.hcceg.veg.compassionfree
[Accessed 5 November 2020].

Higgins, J. P., 2016. *Smartphone Applications for Patients' Health,* Tucson: The American Journal of Medicine.

Kim, E. J. et al., 2019. Treatment of Diabetic Macular Edema. *Current Diabetes Reports,* 68(2019), pp. 1-10.

Kirange, D. K. et al., 2019. Diabetic Retinopathy Detection and Grading Using Machine Learning. *International Journal of Advanced Trends in Computer Science and Engineering,* 8(6), pp. 3570-3576.

Leap Fitness Group, 2020. *Home Workout - No Equipment.* [Online]
Available at:
https://play.google.com/store/apps/details?id=homeworkout.homeworkouts.noequipment
[Accessed 5 November 2020].

National Eye Insitute, 2020. *Macular Edema.* [Online]
Available at: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/macular-edema#:~:text=Macular%20edema%20is%20the%20build,and%20thicken%2C%20which%20distorts%20vision.
[Accessed 18 November 2020].

Nielsen, J., 1994. *10 Usability Heuristics for User Interface Design.* [Online]
Available at: https://www.nngroup.com/articles/ten-usability-heuristics/
[Accessed 10 December 2020].

Oyibo, K., Ifebude, B., Adaji, I. & Vassileva, J., 2019. Investigation of the Perceived Persuasive Features. *4th International Workshop on Personalizng Persuasive Technologies.*

Rosebrock, A., 2018. *LeNet – Convolutional Neural Network in Python.* [Online]
Available at: https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/
[Accessed 18 November 2020].

Sethi, A., 2020. *Build your Own Object Detection Model using TensorFlow API.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2020/04/build-your-own-object-detection-model-using-tensorflow-api/
[Accessed 13 May 2021].

Tatú, J., 2019. *Body Branding Bookings,* Dublin: National College of Ireland.

Tatú, J., 2019. *Body Branding Bookings (3B) - Project Proposal,* Dublin: National College of Ireland.

Wang, X., Lu, Y., Wang, Y. & Chen, W.-B., 2019. *Diabetic Retinopathy Stage Classification using Convolutional Neural Networks,* Virginia: Virginia State University.

Wray, M., 2020. *Germany, France push to end male chick 'shredding' in European Union.* [Online]
Available at: https://globalnews.ca/news/6420754/male-chick-culling/
[Accessed 2 November 2020].

6.0 Appendices

This section contains information that is supplementary to the main body of the report.

6.1. Project Plan

National College of Ireland

Project Proposal

Automatic Retinopathy Detection Using Digital Image Processing via a Smart Device

November 2020

BSc. (Honours) in Computing

Software Development

2020/2021

Joey Tatú

15015556

joey.tatu@student.ncirl.ie

Contents

Annotations

App	Application (e.g. on Android or iPhone)
Retinopathy	Damage to the retina of the eye
Diabetes	Diabetes mellitus
DR	Diabetic retinopathy
DME	Diabetic macular edema
CNN	Convolution neural networks
Intent	A page within an Android app
Firebase	Online services such as databases and authenticators provided by Google.

1. Objectives

General

The objective of this project is to create an app that detects retinopathy. This will be created as an Android application using Android Studio.

Checking eye health

The user will be able to use the in-built camera on their Android phone to scan their eyes to be able to detect retinopathy and to determine their health. The scan will be completed using artificial intelligence and image processing. It is expected that this app will be provided to medical professionals to assist in their diagnosis of a patient.

Features

Apart from checking the patient's retinopathy, the app will be able to record patient's general Health, for example; blood sugars, blood pressure, etc. The medical professional will be able to monitor the patient's health and determine if the retinopathy is improving or worsening.

Database

The database will be implemented via Google Firebase Realtime Database. This is a No-SQL database that is generally used for Android apps.

The main tables in the database will be Patients, Health Information and Retinopathy Results.

The Patients table will contain general information on the patient such as their name, age, and medication they are on. The Health Information table will contain the patient's Health status such as any illnesses, blood sugars, blood pressure, weight and height. The Retinopathy Results table will contain the results of the retinopathy scan, the damage of the retina, the white to yellow discolouration, as well as the date and time the scan was taken.

Artificial Intelligence

Artificial intelligence and image processing will be used to scan the patient's eyes to detect retinopathy. Using convolution neural networks (CNNs) for image processing, issues with the retina will be detected and analysed.

2. Background

The idea for this Project began as a vegan health and fitness Android app that would take an image of a person's eyes and only detect the white to yellow ratio of the sclera. The original idea was founded in late August 2020. As a *de facto*, there are many fitness and health apps available. A health app was proposed for this project, but it was felt that this idea has been exhausted with little success for less popular apps.

The author decided to become vegan in mid-September after they saw that male chicks are destroyed shortly after birth. This is known as "chick culling". (Wray, 2020) This was a last straw for them, and they decided to become vegan. While personally eliciting information from switching to a vegan diet, there was not that much information was provided. The information that was provide was just recipes and very general information about veganism. The author felt at a loss on how to correctly transition to veganism.

Being vegan, one asks themselves: "Can I eat this?" After trying a few vegan Android apps (described more in Research below), there was no solid information from these apps on whether a product is suitable for vegan. The main answer that was received was "Not sure". But that was an impasse, as there was no connection to where information could be retrieved to get information on whether the food is Ok for the vegan diet. This project is to redesign vegan, health and fitness apps that are currently available and go above and beyond with improvements. This is how the idea of a vegan health app was discovered.

The idea behind identifying the sclera was founded due to a family member of the author having issues with their gut. This caused the person's sclera to become a tint of yellow and their skin to become sallow. The author thought that a health app with scanning the sclera would be a beneficial idea.

As the project idea progressed, the innovation for the Project was lacking. Creating a general health and fitness app has been done many times previously. It was decided to change the project slightly and to focus on detecting retinopathy in a patient's eyes using artificial intelligence and image processing.

3. Research

Diabetic retinopathy

A widespread disease across the globe is diabetes. Diabetes is caused by the body being unable to create insulin. Without treatment, this causes the blood sugars in the body to become high. People who have diabetes for a very long time can get diabetic retinopathy (DR). (Kirange, et al., 2019)

DR is a condition where the blood vessels in the retinas become damaged. This causes the blood vessels to leak which can result in a person becoming visually impaired. All types of diabetes can result in DR with those having the condition for more than two decades being at a higher risk. Signs that a person has DR is shown with irregularities to the retina. Such irregularities are fluffy white spots in the retina known as "cotton wool spots", small white or yellow tinted build-ups that look waxy or shiny are known as "hard exudates". Haemorrhaging or microaneurysms can also occur. (Kirange, et al., 2019)

Another issue that can occur with retinopathy is diabetic macular edema (DME). DME is an accumulation of fluid in the macula. The macula is a part of the middle of the retina. (National Eye Insitute, 2020) DME can cause a patient's vision to become blurred. It can also cause metamorphopsia, which is a vision impairment that causes straight lines, such as those in a grid, to appear round or curved. Other viewing issues can occur such as seeing colours incorrectly and having issues with reading. (Kim, et al., 2019)

Image processing

Classification of DR stages:

"Stage I: No diabetic retinopathy": A retina that has no DR.

"Stage II: Mild non-proliferative diabetic retinopathy": A retina that has microaneurysms only and on other signs of DR.

"Stage III: Moderate non-proliferative diabetic retinopathy": A retina that can have either microaneurysms, minor haemorrhaging, hard exudates or cotton wool spots.

"Stage IV: Severe non-proliferative diabetic retinopathy": A retina that can have a lot of haemorrhaging, significant constriction and dilation of venules (small blood vessels in the retina), abnormal branching of venules or existing venules dilating.

"Stage V: Proliferative diabetic retinopathy": Natural formation of blood vessels and bleeding into the fluid in the macula.

(Wang, et al., 2019)

Techniques

A group of neural networks is known as "convolution neural networks (CNNs)". CNNs are established to be successful in image processing and cataloguing. Some of the enhancements with CNNs are that the blurriness in the image can be detected as well as making the image clearer and being able to detect edges.

Three of the most-modern CNN designs are AlexNet, VGG16 and InceptionNet (now Google's LeNet), specifically for DR stage categorising.

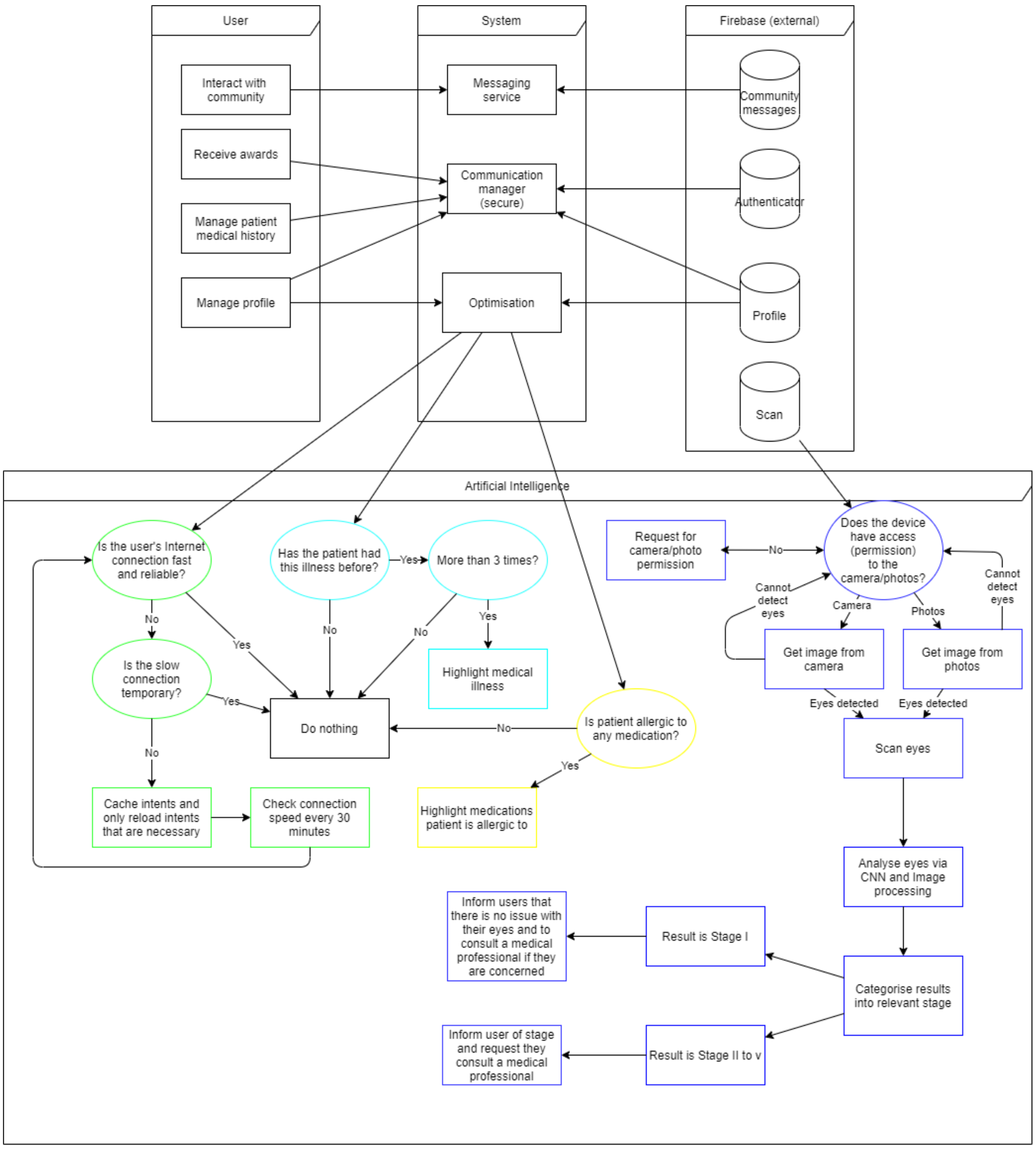
AlexNet is a model by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton, with Hinton being Krizhevsky's Ph.D. consultant. From older CNNs, AlexNet improved the error rate speed as older CNNs have significantly increased error rates in comparison. It can set the result of non-visible neurons to nil, with a 50/50 probability.

VGG16 is an advanced version of AlexNet. Its advantage is how the image is processed adding more viewable layers and filters for better clarity.

LeNet advances AlexNet and VGG16 even further with a reduced error rate and adding more viewable layers and filters.

(Wang, et al., 2019) (Rosebrock, 2018)

4. Technical Approach


User
Interact with community
Receive awards
Manage patient medical history
Manage profile
System
Messaging service
Communication manager (secure)
Optimisation
Firebase (external)
Community messages
Authenticator
Profile
Scan
Artificial Intelligence
Is the user's Internet connection fast and reliable?
No
Yes
Is the slow connection temporary?
Yes
No
Cache intents and only reload intents that are necessary
Check connection speed every 30 minutes
Has the patient had this illness before?
Yes
More than 3 times?
No
No
Yes
Do nothing
Highlight medical illness
Is patient allergic to any medication?
No
Yes
Highlight medications patient is allergic to
Request for camera/photo permission
No
Does the device have access (permission) to the camera/photos?
Cannot detect eyes
Camera
Photos
Cannot detect eyes
Get image from camera
Get image from photos
Eyes detected
Eyes detected
Scan eyes
Analyse eyes via CNN and Image processing
Categorise results into relevant stage
Result is Stage I
Result is Stage II to v
Inform users that there is no issue with their eyes and to consult a medical professional if they are concerned
Inform user of stage and request they consult a medical professional

(Larger version available on request)

These are the main aspects of the app:

Patient / Medical profiles

The patients table will be the account information on the app. No actual names will be used. Instead either the likes of a patient's doctor/hospital number will be used instead.

The medical profile will contain a staff number and password for the medical professional to log in and use the app.

Patient information and profiles will be stored in Firebase Realtime Database and will be authenticated using Firebase Authentication.

Illness querying

The image of a patient’s eye will be uploaded to the Firebase Storage with identifying illnesses (such as haemorrhages, fluid on the macula, etc.). A list of identified illnesses will also be displayed. The medical professional can either tap on the issue on the image or tap on the item in the list to gain more information on the illness detected

Health

Another part of the app will consider the patient’s health and fitness. This will contain the user’s medical history and any previously detected illness of the retina.

The patient’s medical history will be stored in a NoSQL database online using Firebase Realtime Database.

Scanning and checking retina

The medical professional or patient can take a photo or upload an image of the eye. The flashlight on the phone will be used to obtain a clearer image. The image will be checked via a CNN and then uploaded into Firebase Storage.

Version control and methodologies

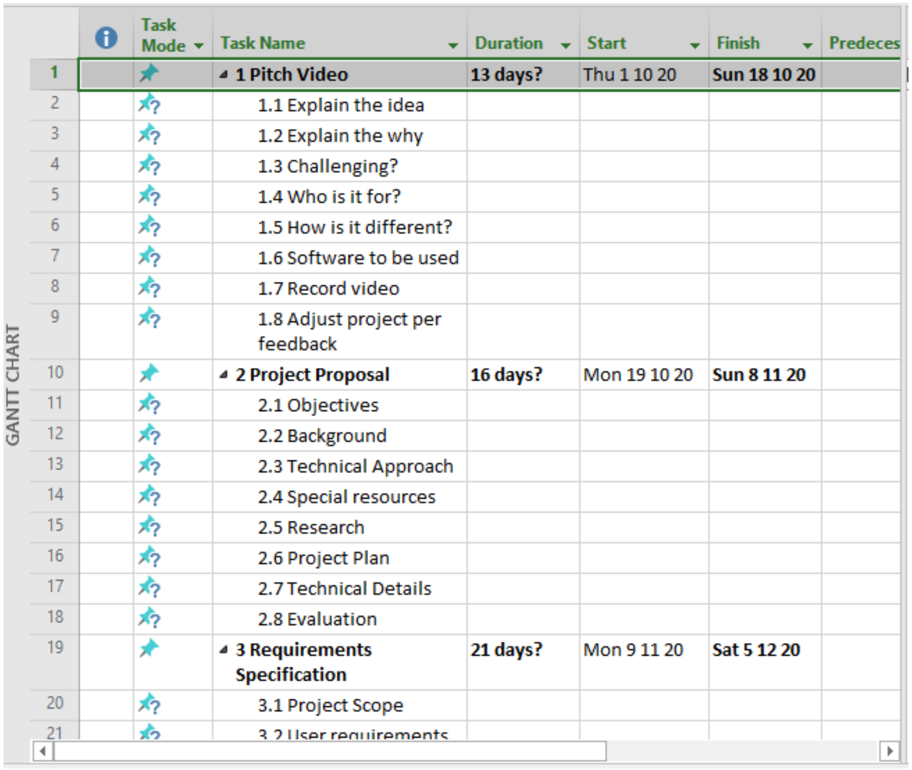
Version control will be handed using a GitHub repository and will be synced using Git Bash. The version control will be located online at https://github.com/JoeyTatu/Software_Project_2020_21.

A mix of Kanban and Scrum will be used for the methodologies.

5. Special Resources Required

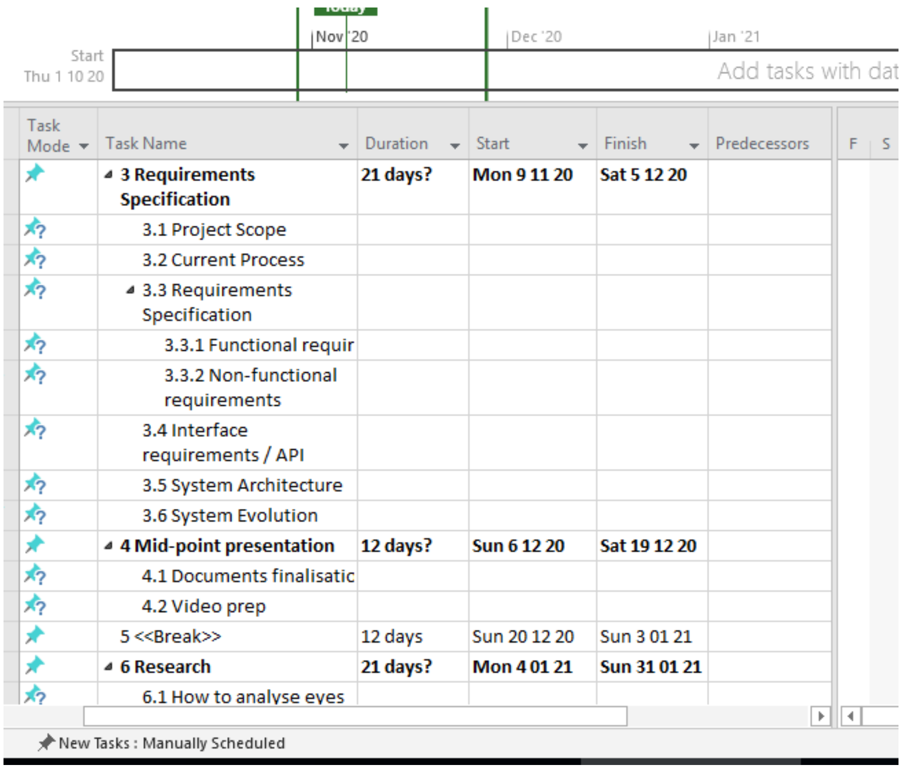
This Project will be completed in Android Studio using Kotlin coding. Implementation of external code required will be implemented in Android Studio. An example of this is using Picasso for simplifying the process of inserting images into intents and sections of the app.

6. Project Plan



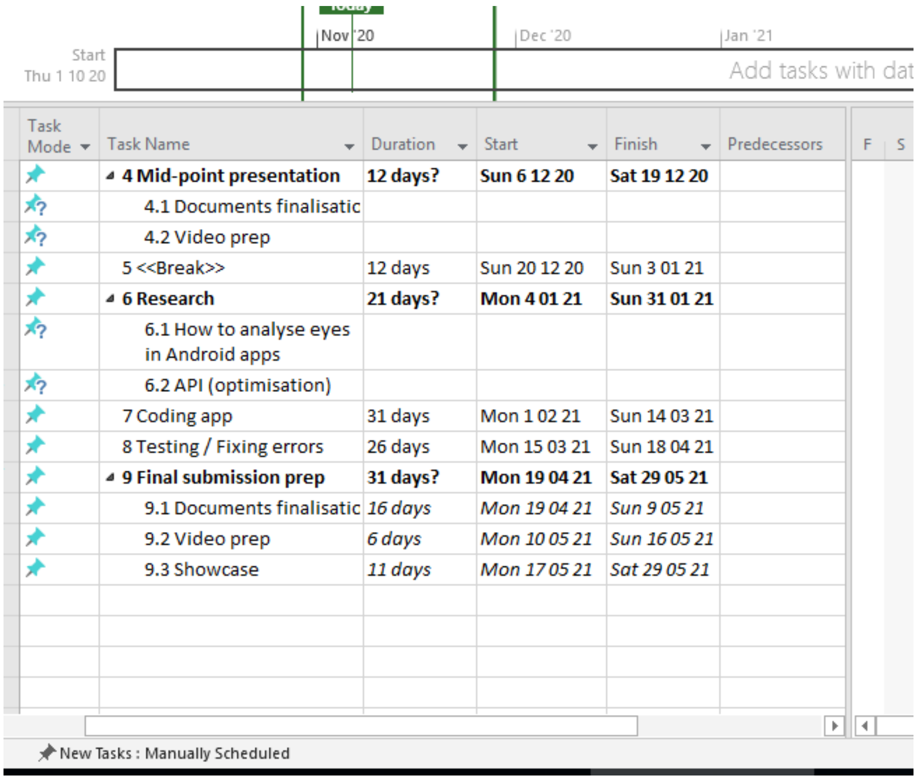
GANTT CHART

	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predeces
1		📌	**◢ 1 Pitch Video**	**13 days?**	Thu 1 10 20	**Sun 18 10 20**	
2		📌?	1.1 Explain the idea				
3		📌?	1.2 Explain the why				
4		📌?	1.3 Challenging?				
5		📌?	1.4 Who is it for?				
6		📌?	1.5 How is it different?				
7		📌?	1.6 Software to be used				
8		📌?	1.7 Record video				
9		📌?	1.8 Adjust project per feedback				
10		📌	**◢ 2 Project Proposal**	**16 days?**	Mon 19 10 20	**Sun 8 11 20**	
11		📌?	2.1 Objectives				
12		📌?	2.2 Background				
13		📌?	2.3 Technical Approach				
14		📌?	2.4 Special resources				
15		📌?	2.5 Research				
16		📌?	2.6 Project Plan				
17		📌?	2.7 Technical Details				
18		📌?	2.8 Evaluation				
19		📌	**◢ 3 Requirements Specification**	**21 days?**	Mon 9 11 20	**Sat 5 12 20**	
20		📌?	3.1 Project Scope				
21		📌?	3.2 User requirements				



Task Mode	Task Name	Duration	Start	Finish	Predecessors	F	S
📌	**◢ 3 Requirements Specification**	**21 days?**	**Mon 9 11 20**	**Sat 5 12 20**			
📌?	3.1 Project Scope						
📌?	3.2 Current Process						
📌?	◢ 3.3 Requirements Specification						
📌?	3.3.1 Functional requir						
📌?	3.3.2 Non-functional requirements						
📌?	3.4 Interface requirements / API						
📌?	3.5 System Architecture						
📌?	3.6 System Evolution						
📌	**◢ 4 Mid-point presentation**	**12 days?**	**Sun 6 12 20**	**Sat 19 12 20**			
📌?	4.1 Documents finalisatic						
📌?	4.2 Video prep						
📌	5 <<Break>>	12 days	Sun 20 12 20	Sun 3 01 21			
📌	**◢ 6 Research**	**21 days?**	**Mon 4 01 21**	**Sun 31 01 21**			
📌?	6.1 How to analyse eyes						

📌 New Tasks : Manually Scheduled



Today

Nov '20 Dec '20 Jan '21

Start
Thu 1 10 20

Add tasks with dat

Task Mode	Task Name	Duration	Start	Finish	Predecessors	F	S
📌	**4 Mid-point presentation**	**12 days?**	**Sun 6 12 20**	**Sat 19 12 20**			
📌?	4.1 Documents finalisatic						
📌?	4.2 Video prep						
📌	5 <<Break>>	12 days	Sun 20 12 20	Sun 3 01 21			
📌	**6 Research**	**21 days?**	**Mon 4 01 21**	**Sun 31 01 21**			
📌?	6.1 How to analyse eyes in Android apps						
📌?	6.2 API (optimisation)						
📌	7 Coding app	31 days	Mon 1 02 21	Sun 14 03 21			
📌	8 Testing / Fixing errors	26 days	Mon 15 03 21	Sun 18 04 21			
📌	**9 Final submission prep**	**31 days?**	**Mon 19 04 21**	**Sat 29 05 21**			
📌	9.1 Documents finalisatic	*16 days*	*Mon 19 04 21*	*Sun 9 05 21*			
📌	9.2 Video prep	*6 days*	*Mon 10 05 21*	*Sun 16 05 21*			
📌	9.3 Showcase	*11 days*	*Mon 17 05 21*	*Sat 29 05 21*			

📌 New Tasks : Manually Scheduled

(Project plan file available on request)

7. Technical Details

The Project will be developed in Android, using the coding language Kotlin. The database and user authentication will be handled by Firebase. Firebase Realtime Database is a NoSQL database. Some of the libraries to be implemented will be Google Play Services for adverts and location services, Picasso to assist in inserting images more easily and Circle Image View to create rounded corners on buttons, images and the like.

8. Evaluation

A Testing Report will be generated after a section is completed to identify and correct errors.

Some examples of Unit Testing, Integration Testing and Performance Testing include:

Unit Testing

Assumption: The user has not opened the app before.

Scenario 1: Create Patient profile

- Can the user successfully connect to the database?
- Does the user have internet access?
  - If the user has not got internet access, the test fails.
- Is the data securely sent and retrieved from the database?
  - An example patient information will be created

    - Before sending to the database, an external programme will copy and try to read the data.
    - If data can be read, the test fails.
    - For a successful test, the data should not be viewable by any other program or person, except the database where it's being inserted or retrieved from.
- Can the medical professional successfully add a patient to the database?
    - The test profile mentioned above will be retrieved and displayed in an intent on the app for testing purposes.
    - A call to the database requesting the profile information will be called
    - If the correct profile information is displayed, the test passes.

Integration testing

Assumption: The patient information has previously been created.

Scenario 1: Medical professional accessing patient information

- Integration between a medical professional (test user) and patient information (test patient).
    - The test user inserts recent history into the test patient.
    - The app creates graphs to show how the patient has improved or worsened.
    - The updated test patient information is sent to the database.
    - The updated test patient information is retrieved from the database and put into the test intent
    - If the correct test patient information is displayed with the patient ID, the test passes.

Performance Testing

Performance testing will be completed using JUnit.

The following will be tested:

Response time

- The response time must be less than 4 secs with 500 users accessing the database at the same time.
- Check the response time of the app/database when a user's Internet connection is slow or limited.
- Check response time when the load condition is low, medium and heavy.

Users and database

- Check what the maximum number of users accessing the app and database before it becomes unresponsive.
- With 500 records being sent and received to the database at one time, check the execution time. The limit would be 10 seconds.

(Tatú, 2019)

References

Burggraf, C., 2020. *Is It Vegan?.* [Online]
Available at: https://play.google.com/store/apps/details?id=net.isitvegan.androidfree
[Accessed 5 November 2020].

HappyCow, 2020. *Find Vegan Restaurants & Vegetarian Food- HappyCow.* [Online]
Available at: https://play.google.com/store/apps/details?id=com.hcceg.veg.compassionfree
[Accessed 5 November 2020].

Higgins, J. P., 2016. *Smartphone Applications for Patients' Health,* Tucson: The American Journal of Medicine.

Kim, E. J. et al., 2019. Treatment of Diabetic Macular Edema. *Current Diabetes Reports,* 68(2019), pp. 1-10.

Kirange, D. K. et al., 2019. Diabetic Retinopathy Detection and Grading Using Machine Learning. *International Journal of Advanced Trends in Computer Science and Engineering,* 8(6), pp. 3570-3576.

Leap Fitness Group, 2020. *Home Workout - No Equipment.* [Online]
Available at:
https://play.google.com/store/apps/details?id=homeworkout.homeworkouts.noequipment
[Accessed 5 November 2020].

National Eye Insitute, 2020. *Macular Edema.* [Online]
Available at: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/macular-edema#:~:text=Macular%20edema%20is%20the%20build,and%20thicken%2C%20which%20distorts%20vision.
[Accessed 18 November 2020].

Nielsen, J., 1994. *10 Usability Heuristics for User Interface Design.* [Online]
Available at: https://www.nngroup.com/articles/ten-usability-heuristics/
[Accessed 10 December 2020].

Oyibo, K., Ifebude, B., Adaji, I. & Vassileva, J., 2019. Investigation of the Perceived Persuasive Features. *4th International Workshop on Personalizng Persuasive Technologies.*

Rosebrock, A., 2018. *LeNet – Convolutional Neural Network in Python.* [Online]
Available at: https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/
[Accessed 18 November 2020].

Sethi, A., 2020. *Build your Own Object Detection Model using TensorFlow API.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2020/04/build-your-own-object-detection-model-using-tensorflow-api/
[Accessed 13 May 2021].

Tatú, J., 2019. *Body Branding Bookings,* Dublin: National College of Ireland.

Tatú, J., 2019. *Body Branding Bookings (3B) - Project Proposal,* Dublin: National College of Ireland.

Wang, X., Lu, Y., Wang, Y. & Chen, W.-B., 2019. *Diabetic Retinopathy Stage Classification using Convolutional Neural Networks,* Virginia: Virginia State University.

Wray, M., 2020. *Germany, France push to end male chick 'shredding' in European Union.* [Online]
Available at: https://globalnews.ca/news/6420754/male-chick-culling/
[Accessed 2 November 2020].

6.2. Reflective Journals

6.2.1. October 2020

Student name: **Joey Tatú - 15015556**
Programme: **BSc (Hons) in Computing – Software Development (PT)**
Month: **October 2020 (Week 1 – 6)**

My Achievements

I came up with a Project idea and put it into my Project Pitch and uploaded it.

My Reflection

Pitch Video:

I needed to keep my Pitch Video under 5 minutes. I feel my Project Pitch was a bit lacking. I do think I expressed it well, but I think it was lacking a bit. Lacking in the sense that I don't think it met the requirements for the Project. However, I hope that when I speak with my Project Supervisor, they can point me in the right direction and help me direct it to the correct format and to do what is required.

Supervisor:

My first Project Supervisor meeting is on 2 November 2020.

Intended Changes

Next week, I am planning on spending most of my time completing the Project Proposal. This is due on next Sunday (8 November). After this, I plan to begin working on the Requirement Specifications.

Supervisor Meetings

Date of Meeting:
Items discussed:
Action Items:

6.2.2. November 2020

Student name: **Joey Tatú - 15015556**

Programme: **BSc (Hons) in Computing – Software Development (PT)**

Month: **November 2020 (Week 6 – 10)**

My Achievements

I finished a version of my Project Proposal. I was focusing on creating a health/fitness app that scanned people's eyes to determine their heath. After a discussion with my supervisor Paul, the Project was altered to focus more on image processing and retinopathy (i.e. damage to the retina).

I have made a start on my Requirements Specification as well.

My Reflection

While I do fully understand Paul Stynes' (supervisor) recommendations, I really thought I had a good idea with the vegan health and fitness app I originally planned. But after speaking with Paul, internally I felt kind of silly with my original, possibly over-simplistic project. But with Paul's feedback and help, I was able to flesh it out a bit and make it from a second-year project to a final year project. To be honest, I never even knew what retinopathy was before starting this. That's a good thing, it's heightens the complexity a bit.

Intended Changes

I plan to finish the Requirements Specification by the end of the first week in December (Week 10) and hopefully have Paul sign off on it on the following Monday in our group supervisor meeting.

The second week in December (Week 11) will focus on writing up the Technical Report and starting the prototype in preparation for the Midpoint submission on 22 December.

Supervisor Meetings

16 November 2020

Items discussed: After a group discussion, I gained some insight into what other students were doing and reflecting on the input from Paul, our supervisor. After the group chat, Paul and I had a one-on-one conversation to discuss my Project. Paul reviewed my Project Proposal and explained that it's simple to create a website/app, and innovation for my Project was lacking.

Action Items: Paul suggested I convert my Project to a website or an app that focuses on retinopathy and image processing. This helped me inflate and mould my Project to where the innovation was sufficient. Paul also suggested to look at reports and papers that are recent, i.e. from 2018 onwards. If I implement these suggestions, there can be little argument to a lack of innovation for the Project.

Paul also said that it doesn't matter how the Project is implemented, whether a website or an app, but either a Raspberry Pi or a smart phone (specifically the in-built camera on the smart phone) could be used.

23 November 2020

Items discussed: The Project Proposal was discussed and signed off

Action Items: Paul explained that if I wanted to, I could add more resources regarding eye health, if I wanted to. He said it was a good proposal as it is, but the extras would give it a boost. With being in Week 9 (of 12), I felt I should press on with the Requirements Specification instead, as I don't want to run out if time and not have something complete for the Midpoint submissions.

30 November 2020

Items discussed: Paul and the group discussed what has been done in the past week. I explained that I had started the Requirements Specification and I expect to have it completed by the end of the week.

Action Items: Paul and I agreed to continue working on the Requirements Specifications and to email him a copy when I have it complete.

6.2.3. December 2020

Student name: **Joey Tatú - 15015556**

Programme: **BSc (Hons) in Computing – Software Development (PT)**

Month: **December 2020 (Week 10 – 14)**

My Achievements

The Mid Point deliverables were due on 22 December. I focused on finishing my Technical Report, Prototype and the video presentation, as well as keeping my GitHub repo updated. I kept in touch with my supervisor Paul to keep him informed on what I was doing.

My Reflection

I feel that I was rushing a lot and possibly stressing myself out a bit. I found it difficult to keep my attention focused on completing my documents. But when I kept pushing myself, I was able to complete it to a standard I was happy with.

Intended Changes

In January, I would like to do a lot more research that I was not able to complete last Semester as creating the documents took most of my time. I would like also to explore other ways to implement the app and adjust the documents accordingly. I plan to focus on app coding/tinkering in February and to focus on testing in March.

Supervisor Meetings

7 December 2020

Items discussed: I explained to Paul (supervisor) that I was working on the Technical Report and I expect to have that finished by Wednesday and to start working on the Prototype for the Midpoint Deliverables after that.

Action Items: I am working on my Technical Report, expected to be complete for Wednesday and after that to begin on the Prototype for the Midpoint.

6.2.4. January 2021

Student name: **Joey Tatú - 15015556**

Programme: **BSc (Hons) in Computing – Software Development (PT)**

Month: **January 2021 (Week 15 – 18)**

My Achievements

This month, I focused on learning about Raspberry Pis and macro cameras, mostly from YouTube tutorials and looking at coding for same.

My Reflection

I think I focused too much on looking at said tutorials when really, I should have been looking at machine learning.

Intended Changes

After speaking with my supervisor, I realised I was focusing on the wrong aspects of the project. For February, I want to get through the machine learning and start properly coding in March.

Supervisor Meetings

1 February 2021

Items discussed: Paul and I discussed the midpoint and about my midpoint marks where he said I lacked with the prototype, which I kind of expected.

Paul also asked me how I was scanning eyes for the diabetic retinopathy and suggested machine learning

Action Items: I am going to focus on research on machine learning for the next week.

6.2.5. February 2021

Student name: **Joey Tatú - 15015556**

Programme: **BSc (Hons) in Computing – Software Development (PT)**

Month: **February 2021**

My Achievements

This month I focused on researching the different model creations for how the image would reference.

My Reflection

I ended up deciding to use TensorFlow Lite as this seems the most straightforward and less complicated way to do.

Intended Changes

After talking with my supervisor, Paul said that working with TensorFlow Lite was probably easier than trying to create a model from scratch, which would severely hinder my time.

Supervisor Meetings

1 February 2021

Items discussed: Paul and I discussed TensorFlow Lite and sent me articles to look towards when creating the model.

Action Items: I am going over the contents of what Paul sent.

6.2.6. March 2021

Student name: **Joey Tatú - 15015556**

Programme: **BSc (Hons) in Computing – Software Development (PT)**

Month: **March 2021**

My Achievements

This month, I focused on trying to upload around 40,000 images to my GitHub to use for the learning file. It didn't go well.

My Reflection

After failing multiple times on the above, I've decided to upload only 1,500 files instead into each Stage.

Intended Changes

I am planning to get the app completed by mid April, then I will be completing the testing and finalising the documents towards the end of the month.

Supervisor Meetings

There were no supervisor meetings this month due to schedule conflicts between my supervisor and myself.

6.2.7. April 2021

Student name: **Joey Tatú - 15015556**

Programme: **BSc (Hons) in Computing – Software Development (PT)**

Month: **April 2021**

My Achievements

This month, I focused on creating the app in Android Studio. I have a few "Old versions" of the MainActivity, but I feel these have been extremely important.

My Reflection

With the multiple re-edits and trying to ensure parts of the code were working for different Android versions, I tried to implement Microsoft Office, Google and Facebook sign in and log outs. Microsoft do not make it easy and I was beginning to invest too much time into trying to get it to work.

I moved onto the Facebook login. This was easier and more straightforward, but I noticed that every 72 hours my API key would expire as I am not publishing my app and Facebook declares this as being in "development mode". In this mode API keys expire. The code in the app works and is present, but I fear that whoever is examining my code, will not be able to get it to work due to the API key expiring. I commented the Facebook code out and moved onto the Google sign in.

With the Google sign in, it was much easier. I got it working and was able to get details from the Google user. I did notice that when the app closed, the user had to sign in again. So the Google sign in first checks if there are two saved files for the Google ID and the display name. If they are not present or are null/empty, the user has to sign in. If there are already present, the login screen is bypassed. This is similar to the sign out where the files are set to empty or " " before going back to the main screen.

I was also working on the main body of the app itself to get the camera/file upload working and comparing it with my dataset.

Intended Changes

With the above said, I am planning to get the app finalised by getting the online database sorted. This is to complete within the first week of May, which also includes testing. After which, I will finalise the documents ready for submission.

Supervisor Meetings

There were no supervisor meetings this month due to schedule conflicts between my supervisor and myself.

However, I emailed my progress to my supervisor on a regular basis.