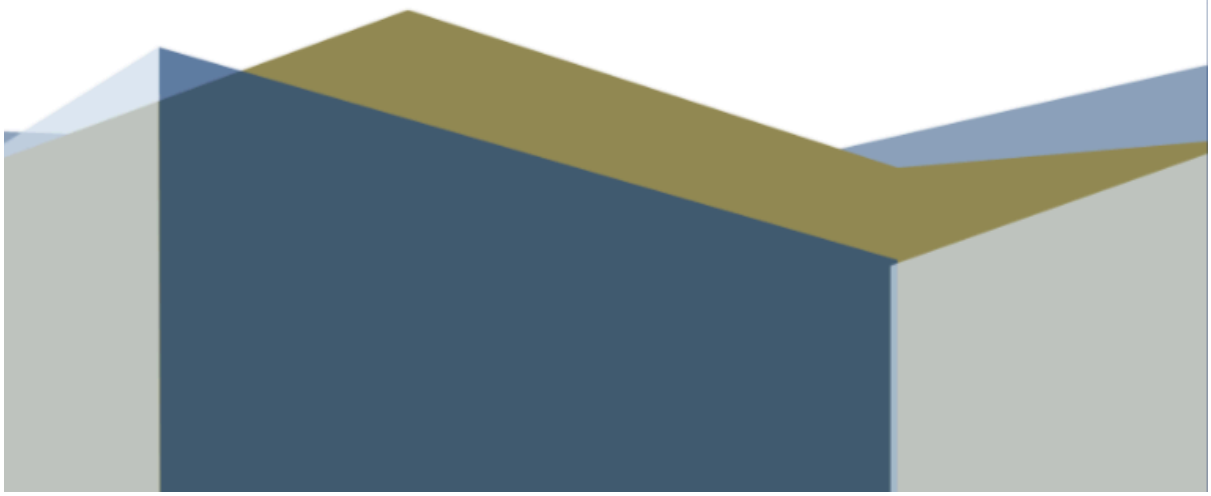
FINDATION APPLICATION – PROJECT REPORT

FACULTY OF COMPUTING & ENGINEERING

A MOBILE APPLICATION WHICH ALLOWS A CUSTOMER TO PICK A FOUNDATION SHADE BASED ON THE COLOUR OF THEIR SKIN.

VERSION 1.0

Courtney Rainey
B00632350
BSc Computing Science
Dr. Don McFall
27th April 2017



ABSTRACT (446 words – 500_MAX)

As the battle to find the perfect foundation shade that works continues and technologies evolve, the cosmetic industry has lacked in the amount of tools and services they have created. They have not provided resources to firstly, combat and optimise sales from this perfect match issue and secondly, utilise their huge growth potential and market opportunity that is available to the billion- dollar industry in the ever growing IT revolution.

Deciding on a foundation formula is a challenging problem. Do you pick sheer, medium or full coverage? However, deciding on the shade selection is an even bigger and complex process. What is deemed to be a good match in the foundation bottle often shows up completely differently once applied to your skin. Since the foundation product is one of the best and most favoured cosmetic inventions of all time, it is finally time to create a software product that aids in this decision making process and provides foundation customers with expert help.

Catching your beguilingly un-bespoke foundation shade match at a counter in a department store is another seemingly stressful and daunting experience. World wide cosmetic retailers have tried to demystify this process with the help of a trained expert and a handheld device that provides you with an exact shade in seconds. The next step to this is to revolutionise this process in the form of a mobile application. This concept provides a solution to tireless shopping trips with the need to wait in long queues to embarrassingly obtain foundation advice from local experts.

This project differs from many of the already packaged, specialised cosmetic applications on the mobile market. The aim of the Findation Application is to eliminate the need for in store colour matching advice services and replace these with a mobile application which has the ability to provide customers with a match to a foundation product that is unique to their skin tone. The Findation Application is an emerging cosmetic technology that fits into the upcoming growth strategy of the cosmetic market. Unlike any other product currently available, this product has the potential to be the future of the make up industry which is used by individuals who are looking for a personal make up match assistant from their mobile device.

The application that has been implemented is a fully functional Android application. It provides a whole range of features such as shade matching, filtering of images, product browsing and foundation location services. This document details all the work that has been carried out from the project planning, elicitation of product requirements, execution of development, implementation of test strategies and many more phases that have enabled this project to be a success.

1. INTRODUCTION

1.1. Problem Statement

This project motivation stems from the developers own personal interest in cosmetics and the daunting experience of getting colour matched in store. According to Sephora, the average woman buys the wrong colour foundation seven times before finding the right shade suitable to her skin tone. This is extremely problematic and costly. A particular foundation shade may have been suitable in store but may not have been suitable when it was applied at home in different lighting. As your skin tone can change throughout the seasons or from the application of fake tan, consumers will be able to make their own judgement on the appropriate shade with the use of this proposed application. With this project the developer is hoping to achieve an application which is niche to the market and if this project is successful it may have commercial potential.

1.2. Project Aim

The aim of this project is to eliminate the need for in store colour matching advice services and replace these with a mobile application, which has the capability to match customer skin colour/tone with a range of suitably coloured foundation shades across all cosmetic brands. Therefore, allowing the customer to purchase a makeup product remotely

1.3. Project Objectives

1) To research the current market and evaluate the currently available cosmetic colour matching applications to determine the most desirable features of a cosmetic application before establishing requirements.
2) To establish a requirement specification which identified both user and product requirements from gathering feedback from stakeholders through the use of online questionnaires and personable interviews.
3) To research the current technology available and pick the most suitable technology stack for developing the mobile application
4) To generate an exhaustive design document of the proposed mobile application
5) To designed and implement a mobile application which meets the demands of the customer and is relevant to the current cosmetic market
6) To extensively perform validation and verification based testing of the application and document the outputs in a test article indicating the pass or failure of each test
7) Evaluate and document the progress of the application weekly throughout the lifecycle of the project.
8) Ensure the full lifecycle of the project has been carried out and detailed in the completed final report by the 27th April 2017

1.4. Software Lifecycle Methodology followed

This project was developed using a modified waterfall model.

2. REQUIREMENTS CONTROL DOCUMENT

2.1. Requirements Gathering Methods

Requirements for this project were gathered using different techniques involving the identified stakeholders: Unstructured interviews were carried out with the mentor and a company (Boots) who has developed an initial interest in the project. Also questionnaires were distributed among the 20 potential users identified.

2.2. Requirements Evolution (500 words)

As the software methodology chose, modified waterfall, does not allow for requirements change, the final list of requirements is the same as the initial list.

2.3. Final Requirements (table 3 pages)

Tables 1 and 2 show the final set of un-prioritised functional and non-functional requirements for the system.

Requirement ID	Description	Rationale
F01	The application shall allow the user to capture a photograph of their skin.	This requirement enables a photograph to be taken by the user for the skin match to take place.
F02	The application shall allow the user to place a foundation order from a third party seller.	This requirement enables a user to purchase the matched foundation shade from an online seller.
F03	The application shall allow the user to sign up to the system using google authentication.	This requirement enables the user to sign up to use the system. The user must provide log in credentials when using the system.
F04	The application shall allow the user to sign in to their user account.	This requirement enables the user to access the applications content after they have signed in.
F05	The application shall display the users colour match profile history.	This requirement enables the user to view a back dated profile of their match data to be able to remember the products they have purchased and potentially re order the product.
F06	The application shall display the most popular foundation products.	This requirement displays the most popular foundation products on the market.
F07	The application shall provide the user with a high end and low end product.	This requirement displays a high end and low end product which enables a customer to purchase a product within their price range.
F08	The application shall provide a digital image from each foundation product.	This requirement allows the user to visualise the product they are buying and the packaging it comes in.
F09	The application shall allow a user to watch YouTube videos of cosmetic tutorials.	This requirement allows the user to watch beauty bloggers to gain advice on applying the foundation.
F10	The user shall be able to search across all make up brands.	This requirement allows the user to see what each cosmetic brand has to offer.
F11	The application shall allow the user to enhance the colour of their skin through the use of filters.	This requirement allows the user to modify their skin tone to predict what shade they may require if they go on holiday.

F12	The application shall provide the user with 5 colour matched products.	This requirement allows the user to select from a range of products they could potentially match with.
F13	The application shall be able to pick up the users GPS location.	This requirement allows the application to pick up the current location of the customer.
F14	The application shall allow the user to logout.	This requirement allows the user to log out of the application.
F15	The application shall provide the user with a higher and lower skin tone shade.	This requirement allows the user to be able to go a shade darker or a shade lighter if they have applied fake tan.
F16	The application shall provide the user with tool tips.	This requirement provides the user with hints and tips on how to use the system.
F17	The application shall provide the user with the closest place to purchase the product in store.	This requirement provides the user with the closest stockist to be able to purchase the foundation product.
F18	The application shall allow the user to take a make up quiz.	This requirement allows the user to take a quiz to determine her skin type.
F19	The application shall provide the user with product 'dupes.'	This requirement provides the user with a less expensive version of the product.
F20	The application shall provide product ingredient details.	This requirement allows the user to view the ingredient details.

Table 1: Functional Requirements

Requirement ID	Requirement Type	Description	Rationale
NF01	SECURITY	The product must ensure users are aware of it collecting their personal data.	This requirement is necessary to ensure the data complies with the Data Protection Act
NF02	SECURITY	The user data should be encrypted.	This requirement is necessary to ensure the data complies with the Data Protection Act and the Computer Misuse Act.
NF03	LOOK & FEEL	The system must comply with the companies branding standards.	This is to provide consistency and to create a strong brand
NF04	USABILITY	The interface should be easy to learn and navigation, buttons, headings and help/error messages are simple to understand.	This is to ensure that the user can use the GUI without any prior training
NF05	USABILITY	The product must be able to be used by with public with no prior training.	Providing an app that is accessible by all.
NF06	PERFORMANCE	A valid login transaction response shall occur within 2 seconds of the request when the system architecture is under normal and peak transaction loads.	This is a normal response time to prevent the end user from having to wait to access the system. This should prevent customer frustration.
NF07	PERFORMANCE	The application must provide the user with a colour match within 5 seconds.	This is a normal response time ensuring good customer satisfaction.
NF08	OPERATIONAL	The application is available on an Android platform across different versions.	This will maximise availability to a wide customer base.
NF09	SAFETY	The application will only provide products that have been clinically tested.	This is a preventative measure in order to prevent skin irritations.
NF10	CULTURAL	The application shall provide a wide range of products for all skin tones enabling multicultural provision.	This system will be all encompassing ensure everyone's needs are met.
NF11	LEGAL	The app shall make users comply to its End User License Agreement (EULA) before using it.	This conforms with the legality of the user making use of this application.
NF12	USABILITY	The quick guide should be context sensitive and help the user achieve common tasks.	No prior training will be required to make use of this app.
NF13	USABILITY	Error messages should display how to recover from the error.	This informs the end user of what they are doing wrong.
NF14	SECURITY	The password must be a minimum of 8 characters and contain an upper case letter, a special character and a number.	The Computer Misuse Act applies here and should prevent fraudulent activity.
NF15	MAINTAINABILITY	The application must conform to Google Java coding standards.	To ensure the app is developed to a high standard.
NF16	MAINTAINABILITY	The application must conform to Androids best practices.	This is to ensure that the app will function across all android platforms.
NF17	MAINTAINABILITY	The application must conform to Androids technical design standards.	This ensures that the application meets the GUI standards.
NF18	LOOK AND FEEL	The application should be non-gender specific.	Creating the app to appeal to as broad a market as possible.

Table 2: Non-Functional Requirements

3. SYSTEMS DESIGN

3.1. APPROACH TO DESIGN

This chapter is the foundation in which the implementation is derived from. A Model View Control high level guide has been used to gather design ideas for implementation. The chapter includes database designs, class structure, design patterns, mock screens and activity diagrams.

3.2. SYSTEM ARCHITECTURE

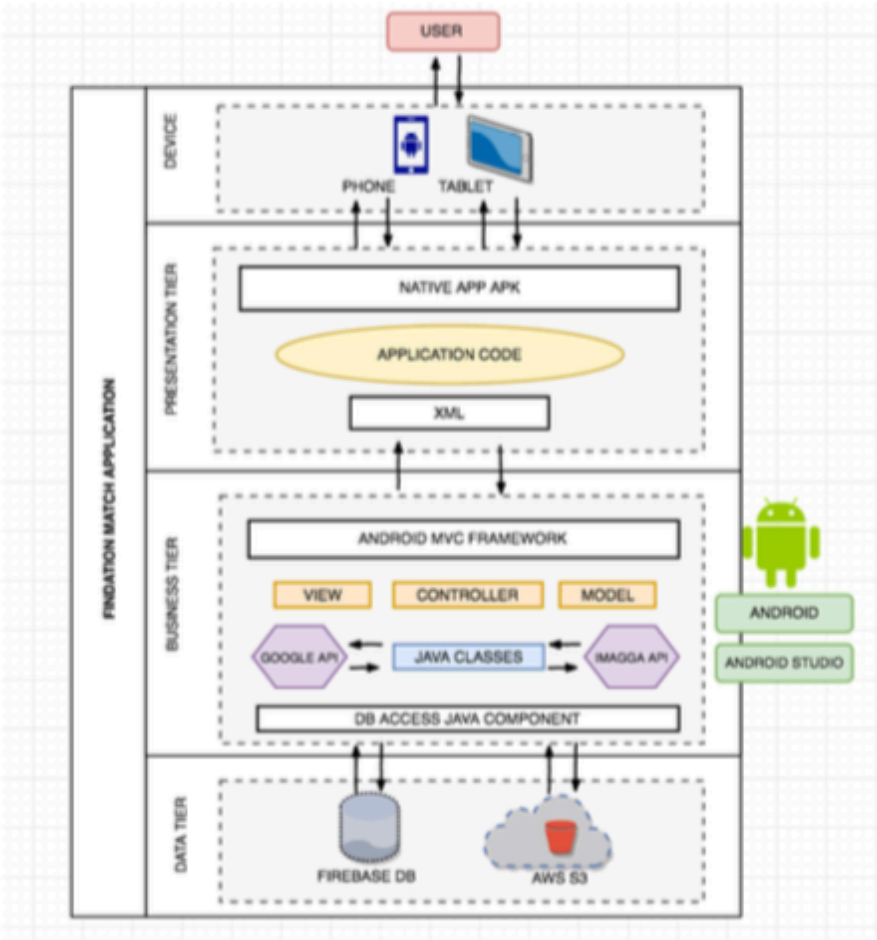


Figure 1: System Architecture

Figure 1 shows the system architecture. From there, it can be seen that the user will be able to use the android application on their Android device whether it be an Android phone or an Android tablet.

There are two devices present in the architecture diagram, an android mobile and an android tablet. These devices will run the Android application provided it has Android Lollipop (5.0) version or above and because the the application is required to be constantly communicating with the cloud based database and images need to be retrieved from Amazon Web services, the device needs to have internet access whether it be via WIFI or a mobile network providing 4G/3G.

A FIREBASE database will be used for the users to be able to log in and sign up to the application. This database will also hold the user's credentials and the user will be able to view their user profile and the history of their foundation matches. This database will also hold all the cosmetic information about each particular foundation shade and the brand it represents.

GOOGLES API & IMAGGA API will be essential parts of this systems architecture. The Java classes of the android application will communicate with them directly. The Java class will request the location of the user from the Google API and the Google API will provide the current location of the user. It will also send back the closest cosmetic shops available to the customer. Additionally, the Java class will request colour information about a particular image to the Imagga API and it will return colour matching information to the Java layer.

Amazon Web Services will play a crucial role in retrieving the colour matching data. The Imagga API requires the image to be posted to it in the form of a URL. Therefore, the image that the user takes on the device needs to be hosted on the cloud and returned as a URL so it can be posted to the Imagga API. Using AWS enables this and this is why it is represented in the data tier of the system architecture model.

3.3. INTERFACE DESIGN

3.3.1. MATERIAL DESIGN

Material design is a conceptual design philosophy which was established by Google. It is deemed as one of the most influential visual guides for mobile design. It offers an extensive list of recommendations and guidelines for "visual, motion and interaction design across different platforms and devices." (Android Developer, 2017) Android includes support for material design applications and a material design specification which documents the components and functionality available for each API level. It provides elements such as themes, widgets and API custom shadows and animations. Considering this application will be used on both a mobile and a tablet platform, material design will support applications developed for multiple platforms and for different screen sizes. A theme has been selected for the design of this application which can be illustrated in the mock up screens in the next section. This theme adheres to the material design guidelines and provides a consistent look and feel to the application. The colours that have been selected to design the application have been selected from Googles Material Design Colour Palette which recommends colour recognition patterns for developers to use in their application.

3.3.2. HCI CONSIDERATIONS

When designing the user interface of an application it is advised to adhere to HCI guidelines. For this project, Schneidermans Eight Golden Rules have been used. By referring to these 8 golden rules, the initial user interface designs for the application have been designed. Table 3 describes how each principle has been achieved.

THE PRINCIPLES	QUESTIONS TO CONSIDER	COMPLETE?
[1] Strive for consistency	Is the style of this element contained across the whole application? Is content placed in the correct location according to the applications hierarchy? **- Googles material UI design has been implemented to ensure this principle has been enacted.**	✓
[2] Enable frequent users to use shortcuts	How do you make the process quicker and easier for users? Is there need to consider more experienced users? **- The home menu has a list of the most popular functions of the application to ensure this principle has been enacted.**	✓
[3] Offer informative feedback	Does the user know where they are in the process? How is feedback being communicated to the user? **- The application provides feedback to the user when they are being provided with a colour match to ensure this principle has been enacted.**	✓
[4] Design dialogue to yield closure	Does the user have to guess? Is it clear and obvious enough for the intended audience? **- The application provides users with instructions when using the application to ensure this principle has been enacted.**	✓
[5] Offer simple error handling	Has everything been done to ensure that the error will not happen? If the user makes an error, how easy it for them to fix? **- Data validation has been implemented throughout the application to ensure that this principle has been enacted.**	✓
[6] Permit easy reversal of action	How many steps does it take for the user to reverse their actions? How to users detect the possibility of reversal? **- The user is able to quickly go back at all times of the application to ensure that this principle has been enacted.**	✓
[7] Support internal locus of control	Does the user feel in control at touch points of the application? How can you make the user feel in control? **- Direct action by the user manipulates data stored in the database to ensure that this principle has been enacted.**	✓
[8] Reduce short term memory load	Do you have to remember things to understand what is going on? How can you help the user recall? **- Each separate function is segregated to prevent confusion and ensure that this principle has been enacted.**	✓

Table 3: Schneidermans Golden Rules applied to the system design

3.3.3. MOCK SCREENS

Figures 2 through 4 show the mock up screens that will make up the application. Figure 2(a) This is the mock screen for the login screen. On this screen the user can sign in to the application or they can click on the sign up button to be able to register. This will be the first screen the user can perform action on when they open the Findation Application.

Figure 2(b) shows the mock screen for the home screen. This is the first screen the user will see when they have successfully logged into the system. It contains a menu of the main functionality of the application in which the user can click on each button to explore.

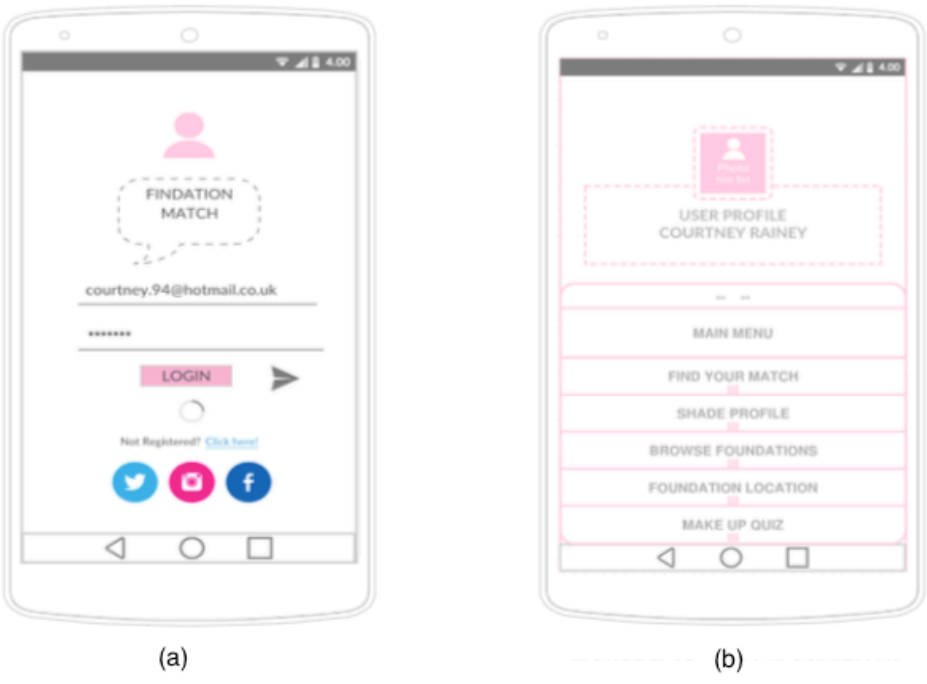


Figure 2: (a) Login Screen. (b) Home Screen

Figure 3(a) is the mock screen for the image capture process. When the user clicks on the, "find your match" button on the main menu the application will open the devices camera which will allow the user to capture an image.

Figure 3(b) is the mock screen for the match results. Here the user is provided with 5 foundation products which they have matched with. These results will be processed through the Java layer based on data obtained from the Imagga API

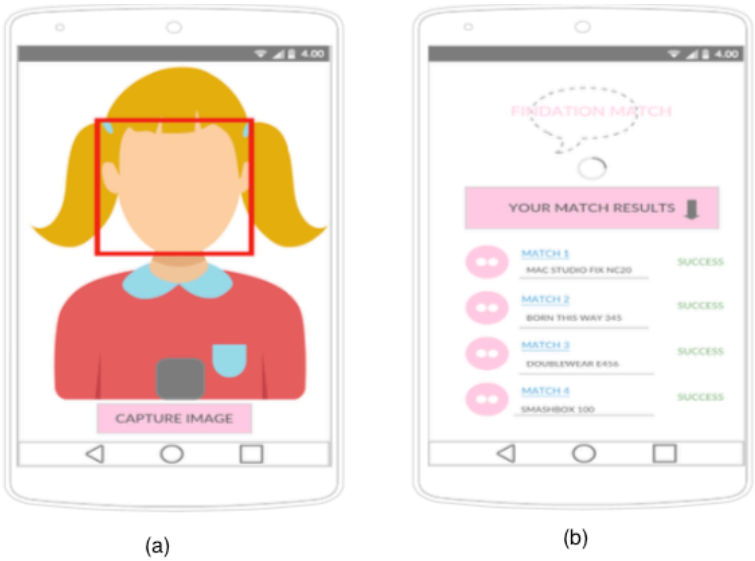


Figure 3(a)Image Capture Process. (b) Match Results Screen

Figure 4(a) is the mock screen for results of the store locations closest to the user's location. These results will be obtained from Googles API and presented to the user with markers on

Figure 4(b) is the mock screen for the user's profile which can be accessed from the main menu. It will provide information about the user's match history in a graphical format.

Figure 4: (a) Store Locator (b) User Profile Interface

3.4. DATA SUPPORT DESIGN

3.4.1. CONSIDERATION FOR SECURITY AND DATA VALIDATION

3.4.2. NO SQL DATABASE DESIGN

Firebase is a NoSQL database which stores data in JSON. The schema for the database of the application was designed with different entities required for the application which have been separated into different objects. Figure 5 is break down of the different objects, the data they contain and how and why they are used in the application. Having no prior experience writing NoSQL schemas, these initial designs are subject to change. However due to the nature of the NoSQL based database, changes can be easily made to tweak the database until the needs of the system have been fulfilled. If a SQL database had of been implemented, it would have been more difficult to implement changes during development.

USER OBJECT

```
{
    "users": {
        "4e238dc1-cr27-4d5d-a7a3-8812a7a3f8b0": {
            "firstName": "Joe",
            "lastName": "Bloggs",
            "email": "bloggs-joe@email.ulster.ac.uk",
            "ST05": true
        }
        "4e245de3-eg13-ff5d-f7f3-321a6d5g4t0": {
            "firstName": "Elaine",
            "lastName": "Veitch",
            "email": "Veitch-e13@email.ulster.ac.uk",
            "ST08": true
        }
}
```

SKINTONE OBJECT

```
{
    "skintone": {
        "ST01": {
            "skinType": "Fair",
            "NC20", true,
            "Fair-Violet", true
        }
        "ST02": {
            "skinType": "Ivory",
            "Peach", true,
            "c87", true
        }
        "ST03": {
            "skinType": "Vanilla",
            "Icecream", true,
            "Sunkissed", true
        }
    }
}
```

PRODUCT OBJECT

```
{
    "products": {
        "MatchMaster": {
            "coverage": "full",
            "MAC":true
        }
        "Mineralize Moisture": {
            "coverage": "full",
            "MAC":true
        }

        "Luminous Weightless": {
            "coverage": "full",
            "NARS":true
        }

        "Born This Way": {
            "coverage": "medium",
            "Too Faced":true
        }

    }
}
```

SHADE OBJECT

```
{
    "shades": {
        "NC20": {
            "shadeType": "Matte",
            "MAC":true

        } "Sunkissed": {
            "shadeType": "Dewy",
            "Rimmel":true
        }
        "E90": {
            "shadeType": "Dewy",
            "Chanel":true
        }
    }
}
```

BRAND OBJECT

```
{
    "brands": {

        "Mac": {
            "BrandCategory": "High-end"
        }
        "NARS": {
            "BrandCategory": "High-end"
        }
        "Rimmel": {
            "BrandCategory": "Drug Store"
        }
        "Benefit": {
            "BrandCategory": "High-end"
        }
    }
}
```

PROFILE OBJECT

```
{
    "Profile": {
        "P01": {
            "Uname": "Crainey94",
            "LastMatchColour": "N45",
            "LastMatchMonth": "Decemeber",
            "FavouriteBrand": "Too Faced",
            "4e238dc1-cr27-4d5d-a7a3-8812a7a3f8b0", true
        }
        "P02": {
            "Uname": "Electric89",
            "LastMatchColour": "N12",
            "LastMatchMonth": "July",
            "FavouriteBrand": "Nars",
            "4e245de3-eg13-ff5d-f7f3-321a6d5g4t0", true
        }
    }
}
```

Figure 5: Database Objects

USER OBJECT

This object will be used to store and retrieve the user information. It will store a unique ID for every customer who has signed up to the system. It will also store the users first and last names and the email address they have provided. Each customer will also be assigned a skin tone which will be stored in the user object to make it easier to provide the customer with a colour match. This table is necessary for security purposes but also for personalisation of the user experience when using the application.

SKINTONE OBJECT

This object will be used to store information regarding skin tones. In order for the matching to take place, each user will be assigned a skin tone. This skin tone will then be mapped to particular foundation shades. The particular skin tone type will also be stored.

PRODUCT OBJECT

This object will be used to store information on the particular products that are potential match recommendations. Each product will be stored including a reference to its particular brand. The type of coverage that this product provides will also be stored.

SHADE OBJECT

This object will be used to store information about the particular shades associated with each product. Each product will have a minimum of six shades associated with it. It will also contain data for the type of shade finish that the product gives. For example, a matte or dewy finish.

BRAND OBJECT

This object will be used to store information about each cosmetic brand that will be included in the application. Each brand will also be categorised into a high end or low end brand category. This will be based on the affordability of a product. When a product is displayed as a match to a customer it will recommend both a high and low end product.

PROFILE OBJECT

This object will be used to store information about each user's profile information. It will include a personalised username for the user. It will also keep track of their last colour match and at what month in the year this match was made. The user will also be able to personalise their profile by selecting their favourite brand. The unique user ID which the user will be assigned when they register the application will be mapped to the user profile.

3.4.3. OBJECT RELATIONSHIPS

An Entity Relationship Diagram (ERD) can still be used to track data and relations in a NoSQL schema less database. Modelling the data that is expected in the system is still recommended. The ER diagram for these objects can be seen in Figure 6.

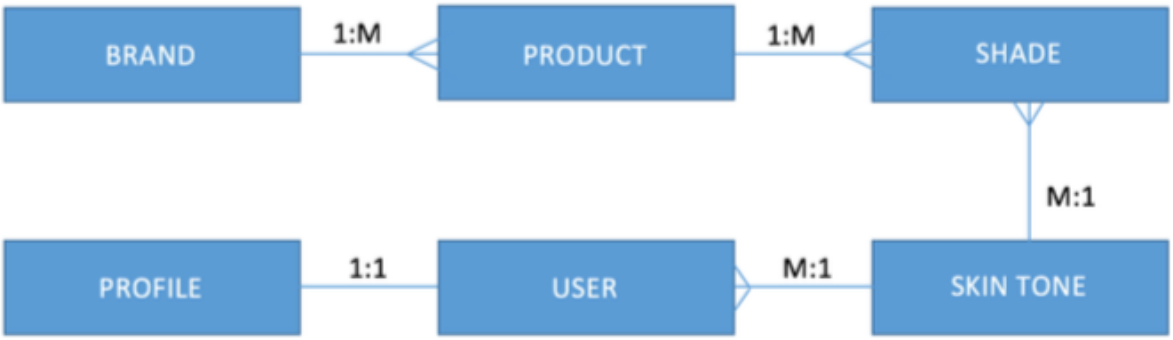


Figure 6: Application ERD Diagram

3.5. USER INTERACTION DESIGN

3.5.1. USE CASE MODELLING

The first stage of the design process was to create a use case diagram for the system. This diagram shows all the proposed functions a user can perform in the system as well as how particular functions extend other functions. This use case modelling technique shows how users can attain goals by executing particular system tasks. Figure 7 shows the use case diagram for the Findation application. From this diagram, the main functionality of the system can be derived. A use case diagram was beneficial for this system because it allowed the developer to cross check the functionality against the requirements to make sure that the required functionality had been incorporated in the design of the system

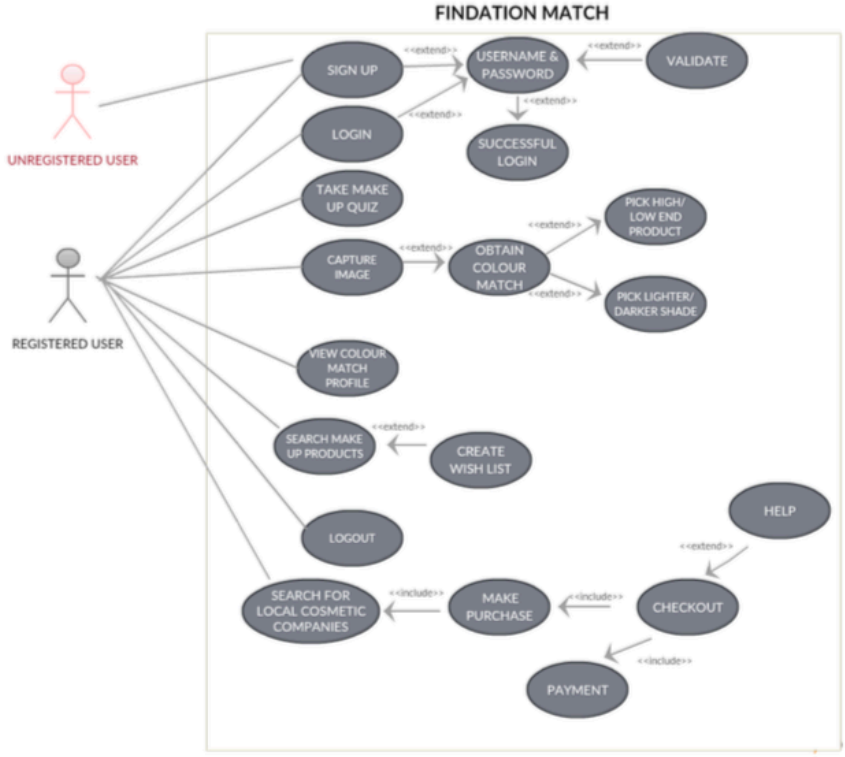


Figure 7: FIndition Application Use Case Model

3.5.2. ACTIVITY DIAGRAMS

Most applications require knowledge on the identity of a user for firstly, security reasons but secondly, to personalise the user experience across all devices. To enable this, registration and authentication will be implemented using Firebases authentication. This authentication provides, “back-end services, easy-to-use SDKs, and ready-made UI libraries to authenticate users,” for the Findation application. (Firebase, 2017) It supports authentication by using email and passwords but also provides authentication from popular providers like Google, Facebook and Twitter. This Authentication services integrates with other Firebase services such as cloud messaging and analytics tools. It also leverages industry standards from OAuth 2.0 which can be easily integrated to customise the backend of this application

Figure 8 demonstrates the process of the user sign up through an activity diagram. It is assumed that the user will have already downloaded the application and has opened it. After the splash screen has loaded, the login screen will appear. Statistically more people return to login to an application rather than the single sign up process. Therefore, the first screen that loads is the login screen. From here, the user has access to a sign up button located on this page in which they can press to sign up to the application. This is the start of the sign up process.
Once the sign up screen is displayed, the user has to enter their email and password. As the authentication has been implemented using Firebase, a request needs to be made to its server to establish a connection. These sign up credentials are then passed to Firebase to allow for a check to be made to ensure that the user email does not already exist. If this email address has been used before then an error message will be displayed to the user. This error message notifies the user that the email address is in use and offers the user the ability to sign up again using a different email address. Ostensibly, if the users email address does not exist then the email and password will be stored in the user object of the NoSQL Firebase database. This will allow the user to login again in the future. After the user account has been successfully created, the user will remain logged in and will be able to access the home screen which displays the menu options of the application.

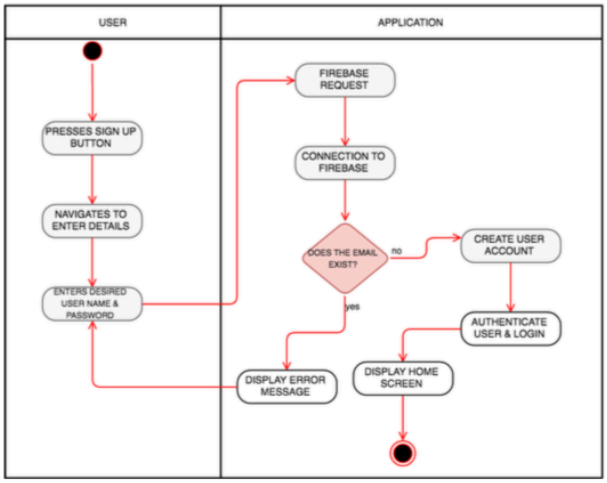


Figure 8: Sign-up Activity Diagram

LOGIN

Figure 9 demonstrates the login process activity diagram. The user firstly opens the application and the splash screen will be displayed for 5 seconds. The user then gets redirected from this to the Login screen. The user enters their email and password and presses the login button. A request is then made to firebase authentication services to establish a connection. Validation will then be performed to ensure that the user account exists within the user's object in the Firebase database. If the user enters an email and password combination that does not exist, then an error message will be displayed informing the user that their credentials are invalid and to try again. On the contrary, if the email and password details are correct and they match the account details that exist in the Firebases database within the user's object then the user will be successfully authenticated and will be logged into the application. This will direct the user to the home screen which displays the menu options of the application.

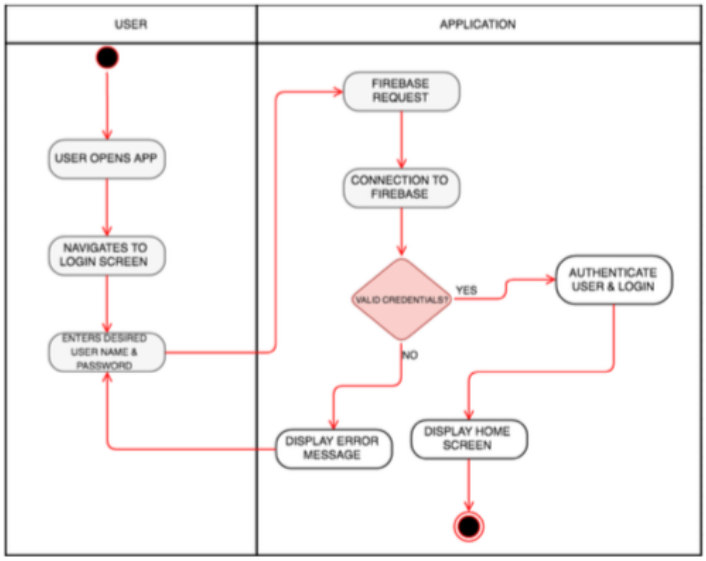


Figure 9: Login Activity Diagram

USER PROFILE

Figure 10 displays the user profile activity. When the user has successfully logged into the application they will be directed to the home screen. On this home screen is a menu which displays the main features of the application. One of these features enables the user to view their user profile. If the user presses the profile button they will be directed to the user profile page. A firebase request is then made to establish a connection to its database. If the user is successfully logged in and has been authenticated correctly, then the user results will be displayed in visual representations such as pie charts. The data displayed will be based on the user's latest matches and most frequently purchased foundations. The user should not be able to navigate to this part of the application if they have not successfully logged in but a check will be put in place to ensure that the user credentials must be revalidated. If this is unsuccessful then no results will be displayed to the user on the user profile page.

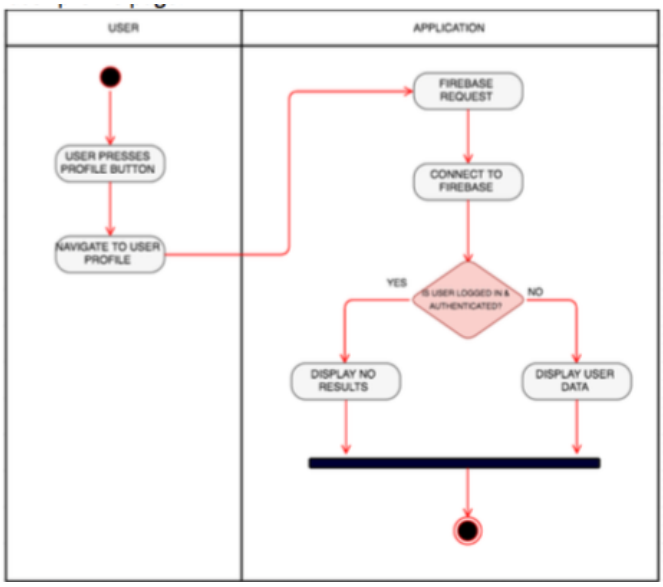


Figure 10: User Profile Activity Diagram

COLOUR MATCH

Figure 11 displays the colour match activity. This process documents the main feature of the application and the most highly prioritised requirement. After the user has opened the application and has successfully logged into the application they will be directed to the home screen which contains a menu displaying the main functionality of the application. If the user clicks the “Findation Match” button the camera will open on the Android device. The user is then required to take a photo of their skin and press the upload button. A request to Amazons Web Services is then created where the image is uploaded to the S3 bucket. This step is required in the process so the image can be hosted and retrieved in an URL format. This is necessary as the Imagga API requires the image to be posted to it in this format. Once the image has been retrieved it is sent to Imagga. In the unlikely event that the API does not return any data then the user will be requested to take another photo. Ostensibly, when data is returned from the API it is then mapped to particular foundations through Java logic and the matches are displayed to the user.

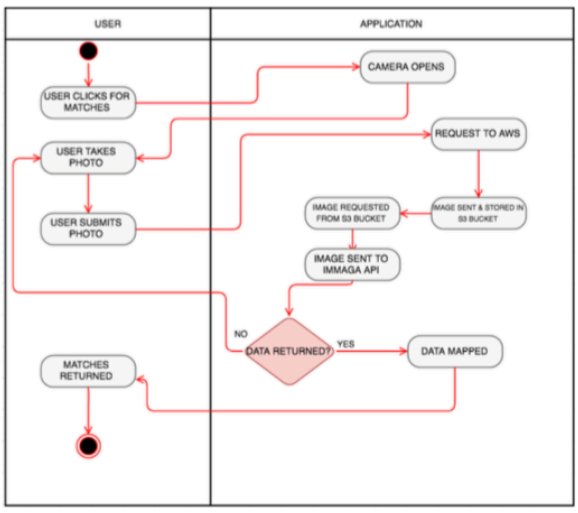


Figure 11:Colour Match Activity Diagram

FOUNDATION LOCATION

Figure 7.8 displays the foundation location activity. After the user has opened the application and has successfully logged in to the application they will be directed to the home screen which contains a menu which displays the main features of the application. One feature which will be accessible to the user is the ability to search for a nearby store where they can purchase a foundation product. When the user clicks on the button they are navigated to the location page. The application then asks the user to grant permission to use their location details retrieved through GPS. If the user rejects this request, then an error message will be displayed to the user notifying them that they need to turn on their location services to be able to proceed with the request. If the user gives the application permission to use their location details, then an additional request is made to Googles API services. It will then return the nearest shops to the user based on their location data.

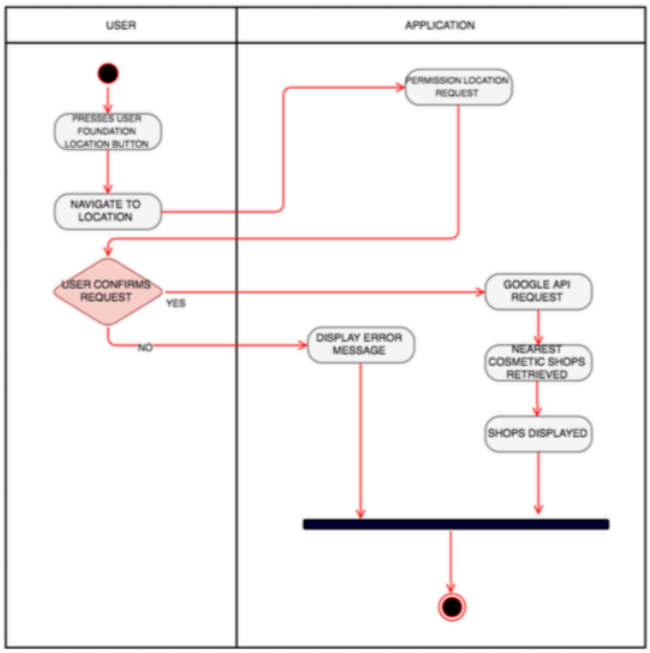


Figure 12: Foundation Location Activity Diagram

4. IMPLEMENTATION

This section goes into more detail about how the design was achieved with code breakdown listings. It highlights the more important features which have been implemented and selected through the prioritised requirements list. It also highlights any design issues that have been encountered.

NOTE FROM MODULE COORDINATOR: This report sample does not contain the currently reqired structure of the report due for submission in 2017/18.

4.1 Summary and Rationale for tools, languages, databses, APIs, frameworks

NOTE FROM MODULE COORDINATOR :Not provided in this sample as was not part of the required submission, instead it is implicit on the explanation of the functionality below

4.2 Evidence of use of version control

NOTE FROM MODULE COORDINATOR :Not provided in this sample as was not part of the required submission

4.3 Summary of the Volume of Code produced

NOTE FROM MODULE COORDINATOR :Not provided in this sample as was not part of the required submission

4.4 System Walkthrough

NOTE FROM MODULE COORDINATOR: This sample does not provide a proper walkthrough of the system, instead it talks through implementing the functionality of the system. For 2017-18 we are looking for a combination of walkthrough with details of implementation code as needed – the limit is 5 pages so needs to be properly summarised

TAKING A PHOTO

A requirement for the Findation Application was to allow the user to capture a photograph of their skin. This photograph is an important aspect of the application as it is used to provide the user with a foundation match. In order to achieve this, this application needs to make use of its built in camera facility on the device. When the user clicks on the camera icon, the onClickListener calls the dispatchTakePictureIntent() method. This method creates a new instance of a camera Intent. An intent is Androids way of delegating actions to other applications, in this instance, the camera application on the hardware device. An intent has three processes; the intent itself, a call to an external activity and code to handle the image data when it returns to an activity. The startActivityForResult() method is protected by a condition. This condition calls the resolveActivity() method which returns the first activity component available to handle the intent. This is a compulsory check because if you call the method startActivityForResult() using an intent that no application can handle, it will crash. Therefore, the condition has been added to ensure the result is not null and the intent is ready to use.

As the application needs to save the image from the user, the code is required to save the image to a file so it can be used. When saving a file in Android, the Android Camera application saves a full size photograph of the image. It is compulsory in this method to provide a fully qualified file name where the application should save the photograph. If the activity is available and not null, then the intent creates an image file for the image to be saved to. An IO exception has been used in case an error occurs when the file is created. If the file is created successfully then the camera intent places the image, using the control of the extra output variable, to the specified file URI path. A URI is a String of characters used to identify a resource. The activity is then started and a request is made to the user to request the image capture.

```
/**
 * This method is used to create the picture intent which is required to capture
 * an image. This image will be an image of the users skin tone.
 */
private void dispatchTakePictureIntent() {
    //Creating the camera intent and provoking the action of capturing an image.
    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    //ResolveActivity method required to ensure activity is available
    //This prevents the application from crashing.
    if (cameraIntent.resolveActivity(getPackageManager()) != null) {
        // Creating the file where the photo should be stored.
        File photoFile = null;
        try {
            //Calling the method to create the image file.
            photoFile = createImageFile();
        } catch (IOException ex) {
            // An error will have occurred when creating the file.
            Log.i(TAG, "IOException");
        }
        // Only continue if the file has been created successfully.
        if (photoFile != null) {
            cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photoFile));
            startActivityForResult(cameraIntent, REQUEST_IMAGE_CAPTURE);
        }
    }
}
```

The createImageFile() method which is called within the try of the try catch statement within the dispatchTakePictureIntent() method is used to create the image file where the image is stored to. The method firstly creates the image file name. This file name consists of the image timestamp, which is created when the file is created. This time stamp is then prefixed with a “JPEG_” stamp. The file storage directory is also created to get the users file directory of photographs on their device. The temporary file path for the image is then created using the image file name as the prefix, the file type, “.jpg” as the suffix and the file directory path. The current photo path is then created by retrieving this absolute path for the image. This file is used with the action view intent.

```
/**
 * This method is used to create the image file.
 * It creates a file with a timestamp in a JPG format.
 * @return image
 * @throws IOException
 */

private File createImageFile() throws IOException {
    // Creates the image file name.
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    // Adds the time stamp to the JPEG_ prefix
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES);
    //Create the temporary file
    File image = File.createTempFile(
            imageFileName,  // prefix
            ".jpg",         // suffix
            storageDir      // directory
    );

    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = "file:" + image.getAbsolutePath();
    mUploadedImage = image.getAbsolutePath();
    System.out.println(mUploadedImage);
    return image;
}
```

CODE LISTING 8.3 – CREATING IMAGE FILE

Once the startActvityForResult() method has been executed then the onActivityResult() method begins. This is where the activity receives the result from the intent and if it is successful it carries out processing the image. The image request code, result code and intent data are passed in the methods parameters. If the request code equates to the requested image success and the result is successful, then the image is retrieved from the media store and set in the image view. This allows the user to check they are happy with the photo. The uploadToAWS() method is then called. This method is to be discussed later on in this chapter. Its purpose is to store the image in an Amazon Web Services storage bucket so it can be hosted and retrieved and sent to Imagga to retrieve a result.

```
/**
 * This method is used to process the image in an activity for a result.
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        try {
            //Retrieving the image from the current photo path.
            mImageBitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(),
                    Uri.parse(mCurrentPhotoPath));
            //Setting the image in the image view to be displayed to the user.
            mImageView.setImageBitmap(mImageBitmap);
            //Displaying the upload button for the user to upload for a result.
            findViewById(R.id.upload).setVisibility(View.VISIBLE);
            //Uploading the image to amazon web services s3 bucket.
            uploadImageToAWS();
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("An IO exception occurred");
        }
    }

}
```

CODE LISTING 8.4 – PROCESSING THE RESULT FROM THE CAMERA INTENT

FILTERING A PHOTO

One of the requirements of the application was to grant the user the ability to filter images. This enables the user to edit the photo allowing them to enhance or adjust their skin tone. The user may want to do this in case they would like a match to a darker or paler colour. The use of filtering also helps combat the constraint of lighting when obtaining a colour match. To enable this filtering functionality to be incorporated a Photo Editor SDK was used within the Findation Application. The first thing required to do was instantiate the SDK so its methods can be used to Filter the images. This was done by embedding a new application within the Findation Application. When initialising the SDK, the license agreement is picked up from within the assets folder which contains an API key to be used to authenticate the developer.


```java
//Importing the SDK's functionality.
import ly.img.android.PESDK;

/**
 * This class is used to initialise an instance of the SDK.
 */
public class Application extends android.app.Application {

    //Overwriting Androids onCreate method to instantiate the SDK
    @Override
    public void onCreate() {
        super.onCreate();
        //Passing in this calls the license agreement.
        PESDK.init(this);
    }

}
```

CODE LISTING 8.5 – INSTANTIATING THE PHOTO FILTER SDK

From section 8.1.2 we can see what methods are called which enable the user to take a photograph. If the user decides to filter the image, then its own Imgly Intent is created. The user is then able to apply filters and adjust the brightness and contrast of an image. In the filter class the onActivityResult() method which can be seen in Appendix A5, enables the user to add the result file and the source type of the image to their image gallery on their phone. Checks have been put in place to ensure that the result path and the source path do not equal null. If the image checks pass the not null criteria, then the uploadForMatchActivity is started and the MatchActivity is ended. If the image is not found or the result is cancelled by the user, then the user can retake the photo. Toasts are also displayed to the user which illustrate the file location in which the image has been stored. When the user saves the file to their phone, it is also stored to AWS for later use with the Imagga API. These methods are discussed in 8.1.5 and 8.1.6 of this chapter.

8.1.4 SPLASH SCREEN

A splash screen has been implemented to welcome the user to the application. It is displayed on the screen for 5 seconds. Once the 5 seconds has elapsed, a handler is used to start the Login Activity and close the splash screen. The full Activity can be found in Appendix A6 alongside the UI of the splash screen.

FIGURE 8.1 – FILTER FEATURE

```
    /* This is a new handler to start the Login Activity.
     * This handler also closes the splash screen after the time has elapsed..*/
    new Handler().postDelayed(new Runnable(){
        @Override
        public void run() {
            /* Create an Intent that will start the Menu-Activity. */
            Intent mainIntent = new Intent(SplashActivity.this,EmailPasswordActivity.class);
            SplashActivity.this.startActivity(mainIntent);
            SplashActivity.this.finish();
        }
    }, SPLASH_DISPLAY_LENGTH);
}
```

CODE LISTING 8.6 – SNIPPET OF SPLASH SCREEN METHOD

8.1.5 PHOTO UPLOAD TO AMAZON WEB SERVICES

In order for the photograph that the user has captured on their device to be able to be uploaded to the Imagga API, this pre process of uploading and hosting the image on Amazon Web Services was vital. This is because the Imagga API can only accept an image URL to perform the colour extraction on. Amazon S3 is cloud storage for the images. In order to upload the photo data, the developer needs to firstly create a bucket in one of the AWS regions. Any number of objects can then be uploaded to this unique bucket. As the image captured by the user on the device was stored in a local file directory on the phone, the next step was to upload the image to the Findation AWS S3 bucket so that it could then be downloaded in the URL format and accessed by the colour matching API. A method called uploadImageToAWS()was implemented within the MatchActivity class to generate this step. This method is an Asynchronous Task which is an abstract class provided by Android. It gives the developer the ability to perform heavy processing tasks in the background and keeps the UI thread light. This enables the application to be more responsive. Android applications run

on a single thread when they are launched. Due to this single thread model, tasks that take longer to fetch a response, such as a network call to AWS, are carried out this way. The three generic types used in this task all return String objects.
The doInBackground() method of this Asynchronous Task is where the S3 bucket name and key are set. Am amazon client is then instantiated with the developer's unique API key being passed in the parameters. This establishes a connection to Amazon S3. The region of the bucket is then set to Western EU. This is an important step in setting up the bucket. Amazon S3 creates the bucket in the specified region purposefully to, "optimise latency, minimise costs and also ensure adherence to address regulatory requirements." (AWS,2017) The next step in the method is putting the uploaded image object captured by the user into the existing bucket name and creating a new file within this bucket. It is also important that you set this buckets control list access to public so it can be used to download the image and be accessed by the application. Once this has been completed the final URL is established. This URL is used by the uploadToAPI() method which posts the image to Imagga for the colour extraction process to take place. An example of this final URL created in the AWS bucket is, "https://findationmatch.s2.amazonaws.com/JPG_20170311.png."

```
@Override
protected String doInBackground(String... arg0) {
    try {
        //Setting the bucket name and key.
        String existingBucketName = "findationmatch";
        String keyName = "image_name";

        //Creating the S3 client and passing the unique API key credentials.
        AmazonS3Client s3Client1 = new AmazonS3Client(new BasicAWSCredentials("AKIAJF7REEIFDH2GZRJA",
                "ou8l524IpqQ1S+iG9cMiI5q/LdQvSpNa3BFvzPT1"));
        //Print out ensuring the connection has been established
        //System.out.println("Connected to Amazon S3.");
        //Getting the region in which the bucket has been set.
        s3Client1.setRegion(Region.getRegion(Regions.EU_WEST_1));
        //Placing the image object in the S3 bucket.
        PutObjectRequest por = new PutObjectRequest(existingBucketName,
                keyName + ".png", new File(mUploadedImage));
        //Making the object public
        por.setCannedAcl(CannedAccessControlList.PublicRead);
        s3Client1.putObject(por);

        //Establishing the final URL
        _finalUrl = "https://" + existingBucketName + ".s3.amazonaws.com/" + keyName + ".png";

    } catch (Exception e) {
        //Writing error to log.
        e.printStackTrace();
    }
    return null;

}
```

CODE LISTING 8.7 – UPLOADING IMAGE TO AWS

UPLOADING IMAGE TO IMAGGA API

The Imagga API is the colour extraction mechanism used within this project to be able to analyse and extract the predominant skin tone from the images taken by the user. In order to make a request in Android to this application another Asynchronous Task needed to be used. Just like the Asynchronous task used to upload the image to AWS, the main thread in Android, the UI thread which handles the most crucial parts of the application cannot be blocked by additional coding activities. The network requests to the Imagga API would block the main thread so this uploadToApi() method must be asynchronous. This Asynchronous Tasks parameters that are used when the task has been executed are void, the progress parameters

return an integer and the results of the background computation tasks return a string. Before any of the four steps of the Asynchronous task are executed the Image URL which is to be sent to the API is set. This is the final URL which was populated with the Amazon S3 bucket details from the previous Asynchronous task. The developers unique key and API secret are also set. These credentials where obtained by the developer when they signed up to Imagga's data plan. The doInBackground() method of this task is where the imageRequest() method is executed. For this tasks code implementation please refer to Appendix A8.
The imageRequest() method which is called in the uploadImageToApi() method is used to create an HTTP request using the open-source Unirest library. Unirest is a set of lightweight HTTP libraries maintained by Mashape. Unirest enables multiple REST methods to be implemented. These methods enable HTTP requests to any API endpoint. This has been adopted to make the colour extraction request to Imagga. A GET request is made to Imagga which passes the image URL, the API key and the API secret. This request is set to the HTTP response and assigned to a stack. The stack is able to retrieve the JSON object from the response and convert this to a string which will be parsed in the colour matching process to obtain the colour prediction data.

```
// Image Request method making the HTTP request to Imagga
public void imageRequest() throws JSONException, UnirestException {

    //This code segment uses the open-source library, Unirest.
    //Assigning the GET request to the HTTP response
    HttpResponse<JsonNode> stack = Unirest.get("https://api.imagga.com/v1/colors")
            .queryString("url", imageUrl)
            .basicAuth(apiKey, apiSecret)
            .header("Accept", "application/json")
            .asJson();
    //Setting the JSON object to a string and assigning it to the results variable.
    _result = stack.getBody().getObject().toString(2);

}
```

CODE LISTING 8.8 – UNIREST REQUEST TO IMAGGA API

As explained, the response of the colours API endpoint is a JSON objet which has two main keys. The first key is the 'results' array which contains the colour results for the successfully analysed image. Within this results array there is information about the colour analysis of the photograph. The results are broken down into background colours and foreground colours. For the purpose of this application, the developer was only interested in the foreground colours as it will contain the facial image. This foreground colours array contains values for 5 different image colours. The developer is interested in the image colour with the highest percentage value as that will refer to the skin colour. Each image colour displays values for the RGB, hexadecimal code, percentage colour and colour variance. The RGB and html code values are required for this application. To illustrate a successful results array, please refer to Appendix A9 for an example. The second key is the "unsuccessful" array which contains the image URL of the unprocessed image and information regarding the reasoning as to why the processed image was unsuccessful.

COLOUR MATCHING PROCESS
The first part of this process required the JSON Object data returned from the Imagga API to be parsed. The JSON data from Imagga was assigned to a result variable at the end of the

ImageRequest() method which can be seen in Code Listing 8.8. This result variable had to be instantiated as the parent object to enable access to its JSON Arrays. Within this parent object was the results array. This results array contained another JSON object named, “info.” This “info” JSON object contained both the background_colors and the foreground_colors arrays. The developer needed to make use of the foreground_colors array as this is the part of the photograph were the user image is obtained. Both the html_code and the percentage strings values within this array are required. The three elements returned represent three foreground colours. For example, the three foreground colours in a facial image represent, the skin, the eyes and the hair colour. A calculation is then performed to check what element out of the three elements returned from the API has the highest percentage colour. The element that has the highest percentage colour represents the skin, as the skin represents the largest proportion of a facial image. In Imagga, the first element of the foreground_colors array always represents the highest percentage cover. Therefore, the first elements, html_code needs to be retrieved. This html_code value contains the hexadecimal colour value for that particular part of the image. This hexadecimal colour value is required to map the image to a particular skin tone set which represents particular foundation shade colours. The method used to obtain this data result can be displayed in Code Listing 8.9 below:

```
//Retrieving different JSON Objects and JSONArraysFrom the Imagga Results
JSONObject parentObject = new JSONObject(_result);
//Getting the results Array
JSONArray resultsArray = parentObject.getJSONArray("results");
//Getting the info Object. An JSON object cannot be gotten as a string
//and therefore you must get the object by its element id.
JSONObject infoObject = resultsArray.getJSONObject(0).getJSONObject("info");
//Getting the foreground_colors JSONArray
JSONArray jsonArray = infoObject.getJSONArray("foreground_colors");

//For loop to search through the foreground_colors array and get the
//JSON object data
//html_code and percentage strings are required fom the result
for (int i = 0; i < jsonArray.length(); ++i) {
    JSONObject resultsObject = jsonArray.getJSONObject(i);
    String html_result = resultsObject.getString("html_code");
    String percentage_result = resultsObject.getString("percentage");
    System.out.print(html_result);
    System.out.print(percentage_result);
}
```

CODE LISTING 8.9 – RETRIEVING HTML CODE VARIABLE FROM IMAGGA RESULTS

The second part of the process enables the HTML code with its hexadecimal value to be mapped to a particular set of skin tones that are stored in the database. There are 15 skin tones. Each skin tone represents particular products and shades. For example, the code represented in the Figure below illustrates a forground_color array element. This elements hexadecimal value is “b89d6b” which falls into the skin tone group 2 range. Within this skin tone group, there are multiple products. The user is recommended 5 products within this group. For example, MAC foundation shade “NC15” or L’Oreal True match foundation shade “vanilla”. These results are displayed to the user.

SKIN TONE GROUP 2

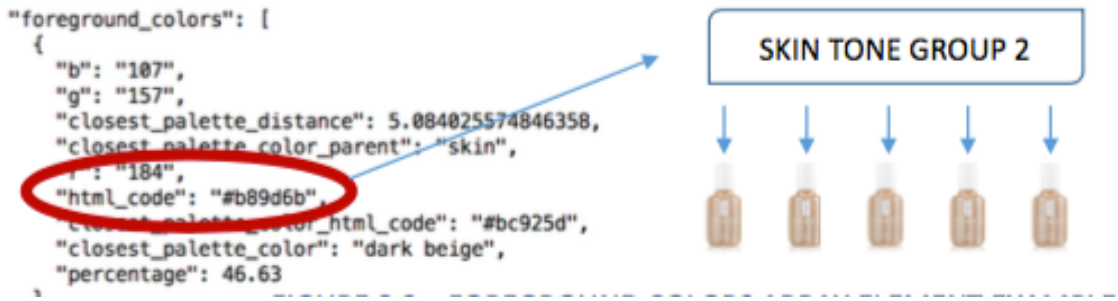


FIGURE 8.2 – FOREGROUND COLORS ARRAY ELEMENT EXAMPLE

GOOGLE MAPS API

A requirement of the project was to implement a feature that would allow the application to provide the user with the closest place to buy a foundation product in store. To achieve this, the Google Maps API was implemented to pick up the user's current location and provide the user with cosmetic companies that stock foundation products within a 10km radius. When the user clicks on the Foundation Location button from the main menu, a user request permission is prompted. Here, the user has the choice to share their location services with the application or not. If the permission is denied, then an error message is prompted and the user is redirected back to the main menu. The functionality that depends on this permission also becomes disabled. If the permission is granted, then the Google API client is built.

Once permission has been granted and the client instantiated, a method is called to check that Googles play services are available. If its available, then an instance of these services are created. If Google play services is not installed on the device, the user will be prompted to install It inside the Support Map Fragment. The Support Map Fragment manipulates the map once it is available and this call back is triggered when the map is ready to be used. The map is then created and the type of Google Map is set. To keep the map fresh and updated the "normal" google map type has been selected rather than the satellite, terrain or hybrid options. Once Google Play services has been initialised then the setMyLocationEnable() method is set to true which enables the users current location. This location layer continuously draws on indication of a user's current location and bearing and displays the UI control that allows a user to interact with their location. With this enabled, the map fragment has been populated and the camera vision zooms in on the position of the map that the user is located. At this location, the user is represented by a purple dot marker.

```
@Override
public void onMapReady(GoogleMap googleMap) {
    //Setting the Map variable to Google Maps.
    //Setting the type of map to Google's "normal" type.
    mMap = googleMap;
    mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);

   //Initialising Google Plays services.
    if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        //Checking the location permission has been granted in the Android Manifest file
        // and that the User has granted permission
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            //Building the Google API client.
            buildGoogleApiClient();
            //Enabling the users location.
            mMap.setMyLocationEnabled(true);
        }
    }
    //If Google play services has already been initialised then,
    // build the client and enable the users location.
    else {
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
    }
```

CODE LISTING 8.10 – LOADING GOOGLE MAPS WITH USER LOCATION LAYER

Located in the left hand corner of the map is a button named, “Local Cosmetic Retailers.” When this button is clicked, the map is cleared. This is to ensure that every time the button is clicked, the current place markers have been removed. When the map is cleared, the getURL() method then passes in the doubles longitude and latitude and the String nearby place instantiates a new String Builder which takes the API request URL as its parameter. It then appends the longitude and latitude to the location alongside the proximity radius which is set to 10km and the nearby place type. In this instance, the near by place type which is most relevant to purchase cosmetic products is a “pharmacy,” and therefore this is the place ID which is being passed. The last piece of data which is appended to the URL is the developer’s personal API key. This key validates the developers request. This method returns the full URL as a String.

```
/** This method is used to pass in the data values required to make the request to the
 * Google Maps API. Its variables are latitude, longitude and nearby place ID.
 */

private String getUrl(double latitude, double longitude, String nearbyPlace) {
    //StringBuilder is created
    StringBuilder googlePlacesUrl = new
            StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");
    //Appending the location, radius, type and API key data.
    googlePlacesUrl.append("location=" + latitude + "," + longitude);
    googlePlacesUrl.append("&radius=" + PROXIMITY_RADIUS);
    googlePlacesUrl.append("&type=" + nearbyPlace);
    googlePlacesUrl.append("&sensor=true");
    //The developers unique API Key.
    googlePlacesUrl.append("&key=" + "AIzaSyATuUiZUkEc_UgHuqsBJa1oqaODI-3mLs0");
    Log.d("getUrl", googlePlacesUrl.toString());
    //Returning the full URL as a string.
    return (googlePlacesUrl.toString());
}
```

CODE LISTING 8.11 – GETTING THE API URL

An instance of the getNearByPlacesData class is then created and both the map fragment and the URL returned from the getURL() method are then passed in as two values of the data transfer array. A “Toast” is then shown to the user that the near by cosmetic retailers are available.

```
Button btnPharmacy = (Button) findViewById(R.id.btnPharmacy);
btnPharmacy.setOnClickListener(new View.OnClickListener() {
    //Assigning the pharmacy string the Place ID type which is pharmacy.
    String Pharmacy = "pharmacy";
    //OnClickMethod
    @Override
    public void onClick(View v) {
        //Log to ensure the button has been clicked
        Log.d("onClick", "Button is Clicked");
        //Clearing the map of an existing makers.
        mMap.clear();
        //Passing the data parameters to the URL.
        String url = getUrl(latitude, longitude, Pharmacy);
        Object[] DataTransfer = new Object[2];
        DataTransfer[0] = mMap;
        DataTransfer[1] = url;
        Log.d("onClick", url);
        //New instance of getNearbyPlacesData
        GetNearbyPlacesData getNearbyPlacesData = new GetNearbyPlacesData();
        //Passing the map and url data
        getNearbyPlacesData.execute(DataTransfer);
        Toast.makeText(MapsActivity.this,"Nearby Cosmetic Retailers",
                Toast.LENGTH_LONG).show();
    }
});
```

CODE LISTING 8.12 – ON CLICK METHOD TO GET THE NEARBY COMSETIC PLACES

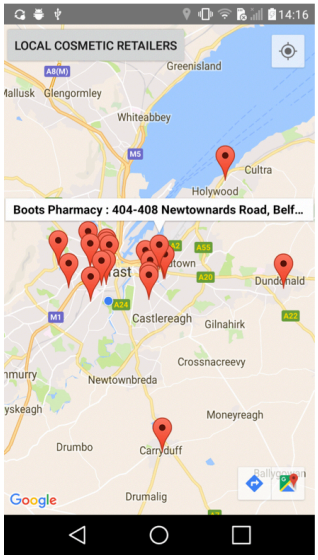
The GetNearbyPlacesData class extends an Asynchronous Task. This enables proper and easy use of the UI thread. It allows the developer to perform background operations and publish results on the UI thread without having to manipulate threads or handlers. Asynchronous Tasks are ideal for short and quick operations which is perfect for sending the API request to the Google Maps API and getting the nearest places result. An Asynchronous task is defined by 3 generic types; Params, Progress and Result. In this instance, the parameters sent to the task upon execution is an Object, the progress published during the background computation is of the String type and the result of the background computation is also of the String type.

In the doInBackground() task, an instance of the Download URL class is initialised. In this class the HTTP URL connection is created and data is read from the URL. This is data being returned from Google Maps API which returns the places information as an JSON object. Once this data has been downloaded it is assigned to a String before it is parsed.

```
@Override
protected String doInBackground(Object... params) {
    try {
        mMap = (GoogleMap) params[0];
        url = (String) params[1];
        //Instiansitiating the DownloadUrlClass.
        DownloadUrl downloadUrl = new DownloadUrl();
        //Assigning the data returned from Google Maps to the Google places data.
        googlePlacesData = downloadUrl.readUrl(url);
    } catch (Exception e) {
        Log.d("GooglePlacesReadTask", e.toString());
    }
    //returning the data as a String.
    return googlePlacesData;
}
```

CODE LISTING 8.13 – STEP 1 OF NEARBY PLACES ASYNCHRONOUS TASK

In the onPostExecute() Task an ArrayList of nearby places is created and set to null. The data is then parsed from the JSON object and the longitude, latitude, place name and vicinity is obtained. This parsed result is then assigned to the ArrayList of nearby places. A showNearbyPlaces() method is then called which takes in this Array List in its methods parameter. Marker options are then created for each place returned in the results locating its position and providing a title for each vicinity. These are placed on the map and are represented by a red Bitmap Descriptor Factory Marker. The map camera is also moved from the current position to the position of the closest venues and it zooms in on these place locations. Now the user can successfully view cosmetic places close by.




```
@Override
protected void onPostExecute(String result) {
    //Creating a nearbyPlaces Array list and setting it to null.
    List<HashMap<String, String>> nearbyPlacesList = null;
    //Creating the dataParser.
    DataParser dataParser = new DataParser();
    //Assigning the parsed data result to the List.
    nearbyPlacesList =  dataParser.parse(result);
    //Calling the nearby places method to mark the data points on the map.
    ShowNearbyPlaces(nearbyPlacesList);
}
```

Throughout the implementation of this project there have been many areas of software development that the developer has had to learn and acquire new skills in using the picked technology stack. One significant challenge the user had when developing the application was ensuring that all features and aspects of the code were suitable for the particular API version selected. This was not the case when implementing the Unirest HTTP request to the Imagga API. No result data was being returned, despite no errors in the code. The reason this was happening is because the Unirest library and the Maven request only run with Version 5.0 of Android and the device that the application was running on was outdated. A solution to this problem was to run the application on multiple platforms when developing the application. This was done on an Android tablet and an Android Device. Another problem encountered was the users lack of experience with using both Firebase and AWS services. This led to difficulties in implementing the colour matching aspect as the developer had to spend additional hours researching and undertaking tutorials on these technologies to be able to implement the required functionality.

4.5 Security Implementation.

The system has a requirement to keep the user data encrypted. It also has the required functionality that allows the user to sign up, sign in and sign out. Another non-functional requirement that this system implements is the ability for a login transaction to occur within 1 second of the request, when the system is under normal and peak transaction loads. The system fulfils these requirements with its use of the software package Firebase for all of the applications user authentication and login functionality. Firebase also handles encrypting the user data in transit since it its sent via HTTPS.

SIGN IN

Then signIn() method is called when the sign in button has been clicked. The username and password variables are passed in the method signature. These are the credentials the user has entered. There is a check to ensure that the form has been validated. The validateForm() method checks if the email field and password field are empty. This method can be found in Appendix A2. If the form has been validated properly, then the method continues to execute. The Firebase Authenticator calls the signInWithEmailAndPassword() method. If the sign in fails, an error message is displayed to the user. If sign in succeeds, then the authentication state listener will be notified and logic to handle the signed in user is handled by the listener. The main menu Intent is then created and the user is redirected to the main menu page.

SIGN UP

When the user account is being created in the createAccountMethod() the same validateForm() method that was used in the signin() method is used to check that the email and password have been inputted correctly and that both values have not been left blank. The Firebase Authenticator calls the createAccountWithEmailAndPassword() method which creates a new user with their email and password. If the email does not exist in the Firebase database, then this method will be a success and the user will be created and granted access to the application. If the user is unsuccessful, then they will be redirected to enter in their credentials again for another email address. Figure XX shows a snippet of the code used on the createAccount method.

```
/**
 * This method is used to create a new user Account with an email and a password.
 * @param email
 * @param password
 */
private void createAccount(String email, String password) {
    Log.d(TAG, "createAccount:" + email);
    if (!validateForm()) {
        return;
    }
```

5. SYSTEM TESTING AND VERIFICATION

5.1. TESTING STRATEGY

Testing is a crucial yet somewhat neglected aspect of building a software product. Testing was necessary to ensure the Findation Application was reliable, robust and could run independently. Therefore, it is important to have a rigorous test driven strategy in place which outlines this projects testing approach. As the developer was using the Revised Waterfall method, testing was carried out after implementation was complete. However, as the Revised Waterfall method has a more flexible structure, implementation could have been revised if necessary after the test phase was complete. It was decided to have a test/feedback step for each feature and phase of implementation. The majority of the testing carried out for this project is white box testing. This approach is used to generate test cases that cover the adequacy criterion for all code elements identified by the approach. White box testing is applicable to lower levels of testing: Unit Testing and Integration Testing. These testing methods are described in the next section.

Unit testing is used to ensure that small component functionality is consistent with the requirements. As a result, it helps to improve the overall software quality. (Jamro, 2015) In Android development either local test units or instrumented unit tests can be carried out. Local unit tests avoid the need for the application to be ran on an emulated or physical device as they make use of the JVM instead. This ensures a faster execution time which can quickly provide results for specific tests. Mockito is a framework which enables the mocking of simple Android dependencies and as a result was integrated into the project to aid unit testing. A unit test called *testLoginUser()* was created to test on a small login component. After unit testing is concluded, the individual code components that have been tested are collated and validated using another set of test drivers. This is referred to as Integration Testing. Its objective is to ensure that, “all components interact and interface correctly with each other, that is, have no interface mismatches.” (Hartmann J., Imoberdorf C., Meisinger M., 2000)

5.2. SYSTEM VERIFICATION

Table 4 shows all the test carried out on the system. Each scenario is described and expected and actual results presented along with a pass/fail mark for each test.

TEST#	TEST SCENARIO	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
#1.0 OPEN THE APPLICATION & TIME HOW LONG IT TAKES THE SPLASH SCREEN TO LOAD	CAN THE APPLICATION BE OPENED?	5 SECONDS & THE LOGIN SCREEN SHOULD BE DISPLAYED	5 SECONDS & THE LOGIN SCREEN IS DISPPLAYED	PASS
#2.0 CUSTOMERS MUST BE ABLE TO LOGIN IN. PLEASE LOCATE TO LOGIN SCREEN.	CAN THE TESTER CLICK THE LOG IN BUTTON?	TESTER HAS THE ABILITY TO LOGIN.	TESTER DOES HAVE THE ABILITY TO LOG IN	PASS
#3.0 LOCATE THE LOGIN SCREEN. ENTER AN EMAIL BUT DO NOT ENTER A PASSWORD AND HIT THE LOGIN BUTTON.	EMAIL: RAINEY-C12 @EMAIL.ULSTER.AC.UK PASSWORD: NULL	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER TO ENTER A PASSWORD	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER TO ENTER A PASSWORD	PASS
#4.0 LOCATE THE LOGIN SCREEN. DO NOT ENTER AN EMAIL BUT ENTER A PASSWORD AND HIT THE LOGIN BUTTON.	EMAIL: NULL PASSWORD: TEST123	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER TO ENTER AN EMAIL	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER TO ENTER AN EMAIL	PASS
#5.0 LOCATE THE LOGIN SCREEN. DO NOT ENTER AN EMAIL AND DO NOT ENTER A PASSWORD AND HIT THE LOGIN BUTTON.	EMAIL: NULL PASSWORD: NULL	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER TO ENTER BOTH AN EMAIL AND A PASSWORD	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER TO ENTER BOTH AN EMAIL AND A PASSWORD	PASS
#6.0 LOCATE THE LOGIN SCREEN. ENTER AN INCORRECT EMAIL AND A CORRECT PASSWORD	EMAIL: RAINEY @EMAIL.ULSTER.AC.UK PASSWORD: TEST123	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE USER THAT THE EMAIL IS NOT VALID	AN ERROR MESSAGE IS DISPLAYED ALERTING THE USER THAT THE EMAIL IS NOT VALID	PASS
#7.0 LOCATE THE LOGIN SCREEN. ENTER AN CORRECT EMAIL AND AN INCORRECT PASSWORD	EMAIL: RAINEY-C12 @EMAIL.ULSTER.AC.UK PASSWORD: TEST3	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER THAT THE PASSWORD IS INCORRECT	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER THAT THE PASSWORD IS INCORRECT	PASS

#8.0 LOCATE THE LOGIN SCREEN. ENTER AN INCORRECT EMAIL AND AN INCORRECT PASSWORD	EMAIL: RAINEY @EMAIL.ULSTER.AC.UK PASSWORD: TEST3	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER THAT THE EMAIL & PASSWORD ARE INCORRECT	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER THAT THE EMAIL & PASSWORD ARE INCORRECT	PASS
#9.0 LOCATE THE LOGIN SCREEN. ENTER A CORRECT EMAIL AND A CORRECT PASSWORD	EMAIL: RAINEY-C12 @EMAIL.ULSTER.AC.UK PASSWORD: TEST123	THE TESTER SHOULD SUCESFFULY LOGIN INTO THE SYSTEM AND THE MAIN MENU SHOULD BE DISPLAYED	THE TESTER HAS SUCCESSFULLY LOGGED INTO THE SYSTEM AND THE MAIN MENU IS DISPLAYED	PASS
#10.0 CUSTOMERS MUST BE ABLE TO SIGN UP. PLEASE LOCATE TO SIGN UP SCREEN.	CAN THE TESTER CLICK THE SIGN UP BUTTON?	TESTER HAS THE ABILITY TO SIGN UP.	TESTER DOES HAVE THE ABILITY TO SIGN UP	PASS
#11.0 LOCATE THE SIGN UP SCREEN. ENTER AN EMAIL BUT DO NOT ENTER A PASSWORD AND HIT THE SIGN UP BUTTON.	EMAIL: RAINEY-C12 @EMAIL.ULSTER.AC.UK PASSWORD: NULL	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER TO ENTER A PASSWORD	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER TO ENTER A PASSWORD	PASS
#12.0 LOCATE THE LOGIN SCREEN. DO NOT ENTER AN EMAIL BUT ENTER A PASSWORD AND HIT THE SIGN IN BUTTON.	EMAIL: NULL PASSWORD: TEST123	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER TO ENTER AN EMAIL	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER TO ENTER AN EMAIL	PASS
#13.0 LOCATE THE SIGN UP SCREEN. DO NOT ENTER AN EMAIL AND DO NOT ENTER A PASSWORD AND HIT THE SIGN UP BUTTON.	EMAIL: NULL PASSWORD: NULL	AN ERROR MESSAGE SHOULD BE DISPLAYED ALERTING THE TESTER TO ENTER BOTH AN EMAIL AND A PASSWORD	AN ERROR MESSAGE IS DISPLAYED ALERTING THE TESTER TO ENTER BOTH AN EMAIL AND A PASSWORD	PASS
#14.0 LOCATE THE SIGN IN SCREEN. ENTER AN INVALID EMAIL AND A VALID PASSWORD	EMAIL: RAINEY.COM PASSWORD: TEST123	AN ERROR MESSAGE SHOULD BE DISPLAYED TO THE TESTER ALERTING THEM THAT THEY HAVE NOT ENTERED A VALD EMAIL	AN ERROR MESSAGE IS DISPLAYED TO THE TESTER ALERTING THEM THAT THEY HAVE NO ENTERED A VALID EMAIL	PASS

#15.0 LOCATE THE SIGN IN SCREEN. ENTER AN VALID EMAIL AND AN INVALID PASSWORD	EMAIL: RAINEY-C12 @EMAIL.ULSTER.AC.UK PASSWORD: HELLO**	AN ERROR MESSAGE SHOULD BE DISPLAYED TO THE TESTER ALERTING THEM THAT THEY HAVE NOT ENTERED A VALID EMAIL	AN ERROR MESSAGE IS DISPLAYED TO THE TESTER ALERTING THEM THAT THEY HAVE NOT ENTERED A VALID EMAIL	PASS
#16.0 LOCATE THE SIGN IN SCREEN. ENTER AN INVALID EMAIL AND AN INVALID PASSWORD	EMAIL: RAINEY.COM PASSWORD: HELLO**	AN ERROR SHOULD BE DISPALYED TO THE TESTED ALERTING THEM THAT THEY HAVE NOT ENTERED A VALID EMAIL OR PASSWORD	AN ERROR MESSAGE IS DISPLAYED TO THE TESTER ALERTING THEM THAT THEY HAVE NOT ENTERED A VALID EMAIL OR PASSWORD	PASS
#17.0 LOCATE THE SIGN IN SCREEN. ENTER A CORRECT EMAIL AND A CORRECT PASSWORD	EMAIL: RAINEY-C12 @EMAIL.ULSTER.AC.UK PASSWORD: TEST123	THE TESTER SHOULD HAVE SUCCESSFULLY CREATED AN ACCOUNTIN AND THE MAIN MENU SHOULD BE DISPLAYED	THE TESTER ACCOUNT HAS BEEN SUCCESSFULLY CREATED AND THE MAIN MENU IS DISPLAYED	PASS
#18.0 CUSTOMERS MUST BE ABLE TO LOG OUT. PLEASE LOCATE TO LOG OUT BUTTON	CAN THE TESTER CLICK THE LOG OUT BUTTON?	TESTER HAS THE ABILITY TO LOG OUT AND THE APPLICATION IS TERMINATED.	THE APPLICATION IS TERMINATED AND THE TESTER IS LOGGED OUT WHEN THEY HIT THE LOG OUT BUTTON	PASS
#19.0 CUSTOMERS MUST BE ABLE TO TAKE A PHOTOGRAPH. CLICK ON THE FIND A MATCH BUTTON.	CAN THE TESTER TAKE A PHOTOGRAPH SUCCESSFULLY?	THE CAMERA SHOULD OPEN AND THE TESTER SHOULD BE ABLE TO TAKE A PHOTO	THE CAMERA OPENS AND THE TESTER CAN TAKE A PHOTO	PASS
#20.0 CUSTOMERS MUST BE ABLE TO FILTER A PHOTOGRAPH. TAKE A PHOTOGRAPH ON THE DEVICE AND APPLY THE SEPIA FILTER.	CAN THE TESTER CLICK ON THE SEPIA FILTER?	THE SEPIA FILTER SHOULD BE APPLIED TO THE IMAGE	THE SEPIA FILTER IS APPLIED TO THE IMAGE	PASS
#21.0 CUSTOMERS MUST BE ABLE TO UPLOAD THE PHOTOGRAPH. HIT ON THE UPLOAD BUTTON	CAN THE TESTER CLICK ON THE UPLOAD BUTTON TO UPLOAD A PHOTO?	WHEN THE TESTER HITS THE UPLOAD BUTTON THE PHOTO SHOULD BE UPLOADED TO THE IMAGGA API	THE PHOTO IS UPLOADED TO THE IMAGGA API	PASS
#22.0 HOW LONG DOES IT TAKE FOR THE CUSTOMER TO BE DISPLAYED WITH A MATCH?	RECORD THE TIME TAKEN	THE USER SHOULD RETRIEVE THE MATCH WITHIN 5 SECONDS.	THE USER RETRIEVE THE MATCH WITHIN 3 SECONDS	PASS

#23.0 HOW MANY MATCHES HAVE BEEN RETURNED?	COUNT THE NUMBER OF MATCHES GIVEN TO THE TESTER	5 MATCHES SHOULD BE DISPLAYED	A TOTAL OF FIVE MATCHES ARE DISPLAYED	PASS
#24.0 CUSTOMER MUST BE ABLE TO VIEW THEIR USER PROFILE. HIT ON THE SHADE PROFILE BUTTON.	TESTER HITS THE SHADE PROFILE BUTTON	THE USER PROFILE IS DISPLAYED WITH THE TEST DATA	THE TESTER IS ABLE TO SEE THEIR PROFILE	PASS
#25.0 CUSTOMER MUST BE ABLE TO TAKE A MAKE UP QUIZ. CLICK ON THE MAKE UP QUIZ BUTTON.	TESTER HITS THE MAKE UP QUIZ BUTTON	THE TESTER SHOULD BE ABLE TO TAKE A MAKE UP QUIZ	THE TESTER TOOK THE MAKE UP QUIZ SUCCESFULLY	PASS
#26.0 CUSTOMER MUST BE ABLE TO BROWSE FOUNDATION PRODUCTS. CLICK ON THE BROWSE PRODUCTS BUTTON.	TESTER HITS THE BROWSE FOUNDATION PRODUCTS	FOUNDATION BRANDS ALONGSIDE THEIR PRODUCTS SHOULD BE DISPLAYED	FOUNDATION PRODUCTS ARE DISPLAYED AND THE TESTER CAN LOOK THROUGH THE PRODUCTS	PASS
#27.0 CUSTOMER MUST BE ABLE TO SEE THEIR CURRENT LOCATION ON GOOGLE MAPS. CLICK ON THE FOUNDATION LOCATION BUTTON.	TESTER HITS THE FOUNDATION LOCATON BUTTON.	THE TESTER SHOULD BE RESPRESENTED BY A PURPLE MARKER ON THE GOOGLE MAP.	THE TESTER IS REPRESENTED BY A PURPLE MARKER ON THE GOOGLE MAP	PASS
#28.0 CUSTOMER MUST BE ABLE TO SEARCH FOR LOCAL COSMETIC COMPANIES WITHIN THEIR AREA. CLICK ON COSMETIC RETAILERS BUTTON ON GOOGLE MAPS.	TESTER HITS THE COSMETIC RETAILER BUTTON LOCATED ON THE MAP	THE LIST OF ALL COSMETIC STORES WITHIN A 10KM RADIUS SHOULD APPEAR ON THE MAP REPRESENTED BY RED MARKERS.	RED MARKERS APPEAR ON THE RED MAP WHICH REPRESENT ALL THE LOCAL COSMETIC STORES WITHIN A 10KM RADIUS.	PASS

Table 4: System Verification – Test Results

Based on the Verification tests results presented in table 4, it can be concluded that the systems has passed all the tests.

6. SYSTEM VALIDATION

6.1. RATIONALE FOR USER EVALUATION AND ACCEPTANCE TESTING

The application was subject to a full usability evaluation. This process was carried out by three potential end users of the system who all had different technical capability levels. They were asked to evaluate the main functional criteria of the application. The developer observed their actions when carrying out a set of predefined tasks to see if and where they encountered problems or experienced confusion. This type of evaluation helps to ensure an application is fit for purpose by the end user and defines any issues of the UI that could indicate what improvements could be made to enhance the user experience.
When deciding on the user profile for the selection of participants to undertake the testing, the developer felt that users who are keen enthusiasts of cosmetics and wear foundation was sufficient criteria. The user's IT skills were also questioned which would determine which type of users with a certain level of IT ability this application was fit for. Due to confidentiality reasons the end users wanted to remain unidentified but the background information for each user an be discussed

Since it was anticipated that not all end users owned an Android device, the testing was carried out in person using the Developers Laptop and Android Devices: Tesco Hudl running Android version 4.0 and LG Phone running Android version 5.0. A series of questions were produced to ensure that the core functionality of the application was tested thoroughly by each participant and that each participant was carrying out and experiencing the same aspects of the app. These questions were derived from the projects product functions which were identified in chapter 5, the requirements chapter. Relating these questions to the high level product functions of the system enabled the developer to be able to evaluate if the application had been developed to meet its intended purpose. A "Think aloud protocol," was used with the end users. Before starting, each participant was asked if they could think out loud while they were using the application. The developer suggested that participants should be vocal in what they are looking at, thinking, feeling and doing. This is a popular technique when carrying out usability testing

6.2. USER EVALUATION RESULTS

Table 5 shows the user evaluation results.

QUESTION	PARTICIPANT 1 (ADVANCED IT SKILLS) SOFTWARE DEVELOPER	PARTICIPANT 2 (BASIC IT SKILLS) ACCOUNTANT	PARTICIPANT 3 (MODERATE IT SKILLS) MAKE UP ARTIST
[1] Splash Screen Observations	The splash screen is very appealing to the eye and definitely grabs the user's attention. It represents what the application is.	Its fabulous. I love the image used. It makes the application look exciting and very up to date with the fashion trends of today.	I like the photograph that was used. However, I would be inclined to think that the image represented nail polish with the woman's colourful nails standing out.
[2] Can you sign In?	You cannot access the application if you do not sign in. This is the first screen to load after the initial splash screen.	Yes – the first thing I did was sign in.	Yes – I found the sign up process easy to use. I thought you may be able to sign in with my Twitter or Facebook account though.
[3] Can you sign up?	Yes – A user can sign up if the, "Not a member? Sign up" link at the bottom of the screen.	It took me about a minute to locate where the button was to click to sign up. I thought it would be more clear on the first screen.	Yes – You can sign up with ease.
[4] Can you reset your password?	Yes – there is a reset password button on the main page which you can click and an email is sent to your email address to reset your password. This was a nice touch.	You can but I cant even remember my email account password to use this feature.	This feature is available to users when they click the link.
[4] Can you logout?	There is a log out button available on the main menu where you can log out.	Yes - you can log out of the application.	Yes – it is possible to log out of the application. However, I think this should be available at all times.
[5] Main Menu Observations	The main menu is well designed. I particularly like the use of icons to represent each particular feature.	The main menu made use of a good layout and it was not over cluttered that the user felt over whelmed when using the application.	I like the personal touch on the main menu. It represented the features of the application appropriately.
[6] What were you first drawn to?	Getting my match of course, it's the most exciting feature of the application.	I wanted to find my foundation location. I was intrigued by this menu item so I clicked on this first on the main menu. I was keen to explore this applications features.	I was looking forward to browsing the foundations that were available.

[7] What happens when you click on the, "Find your perfect Match button?"	The camera opens up which allows you to take a photograph of yourself. Your permission to use your devices camera is required.	I hit no to the permission request and had to re-click the button. However the camera appears where you can take a photo.	The camera appears where the user can take a photo of their face to be able to obtain a match.
[8] What happens when you click on the "Your shade profile button"?	My user profile is shown with match data. I like the layout of this page.	You can see my information.	My own personal account is shown with my details.
[9] What happens when you click on the, "Browse foundation button?"	You are able to search through lots of foundation products.	I could spend hours looking through the foundation products that are provided when you click this button.	Lots of brands with their products and shades are listed.
[10] What happens when you click on the, "Foundation Location Button?"	My location data is collected and you can see my position on the map.	Google Maps opens up.	I am pin pointed on a map which looks just like Google Maps.
[11]What happens when you click on the, "Make Up Quiz Button?"	You are able to take a make up quiz.	I can take a make up quiz as the button suggests.	The option to take a make up quiz is provided.
[12] Can you edit your image? Can you apply filters to your image?	Absolutely love this feature. Yes, there are so many filters to choose from. You can also adjust the brightness and the contrast.	You can do this. However, as I have never used features like this on an application before, it took some getting used to. However, I spent quite a while using this feature.	A very appropriate feature which is perfect for this application.
[13] What do these filters achieve?	You can do so much with these features but its very appropriate for fixing the lighting on photographs.	It makes the picture look better.	This feature is really important if an individual applies fake tan or intends on applying it and wants to be matched to an appropriate shade.
[14] Can you upload your image to gain a colour match?	Yes – there is an upload button. You can also save your edited image to your gallery.	It took me a while to understand what I was doing but this functionality is available.	It wasn't clear that you were about to obtain a match.
[15] Was a match result provided?	Yes – I was provided with 5 match results.	I got a match, yes!	Yes, a list of foundation products where displayed that match my photo.

[16] Was your location picked up by the image? Was this an accurate representation?	As soon as I clicked on the Foundation Location button, the map zoomed in to my particular location and I was represented in real time with a purple marker.	I was a little purple marker on the screen. Yes, my location was picked up and I was able to see my local shopping centre on the map so my location was definitely correct.	The map zoomed into my current location on a normal map view. You could see my position on the map with a marker.
[17] What happens when you click on the "cosmetic retailer button" on the map?	Red markers appear of local cosmetic companies that are near to me. The map position also changes.	I saw exactly were I was on the map and lots of places I recognised beside me.	Lots of red markers appear within about 8 miles from my location representing all the available cosmetic retailers close by.
[18] What happens when you click on the red marker on the map?	You are provided with information about these cosmetic companies. I think more information should be provided.	A white box appears with the shops address.	You get details about local cosmetic retailers and their exact street address appears. More information would be appropriate.
[19] Can you browse foundation products?	Yes – There was a good range of foundation products for me to browse. However, I think the user of additional search functionality or criteria could be implemented.	Yes – I could scroll through lots of products.	Yes – There was sufficient products for me to browse. You could maybe have an advanced search option.
[20] Do you feel like enough foundation products have been incorporated in this application?	Yes - I think that there was an appropriate amount of both high end and low end make up brands. It gave the user the choice.	Yes – I was slightly overwhelmed with the range of products. Less is more!	No – I would like to see more brands represented in this application. Even a broader range of shades in the brands that you have represented.
[21] Comments – Theme and colour scheme of the application	The theme was very appropriate and represented the brand extremely well. It was very modern.	Its very glamorous look to the application and I think it suits us foundation wearers perfectly.	I feel that the application was designed for females. However it does represent the foundation brand and it does have the look and the feel.
[22] Comments – Layout of the application	Application was easy to use. I liked the layout of the main menu. It highlighted the key functionality of the application to the user.	It was relatively easy to use. I could not find the sign up button for ages. I feel that its positioning could be improved.	The application was easy to use. I think it has logically been designed.

[23] Comments - Could you easily sign in/login/logout?	I had absolutely no issues with signing up, logging in or logging out. This applications process have adopted a standard approach when using mobile applications.	I could not locate the sign up button on the main screen. I think this should be made more clear. I had no issues signing up or logging out of the application.	Yes, I could easily perform this functionality. I liked the way you had to sign in to keep your user profile data safe.
[24] Comments – Suggestions on Google Maps.	I liked the Google Map feature. However, one suggestion would be to add different map views to suit different user preference.	I thought the use of location services was very beneficial and is something I would use.	I would like to see additional information about each the locations provided.
[25] Comments – Suggestions on match process.	The process was methodical and enabled the user to understand the match process. I would like to see additional information about the foundation products.	For someone who doesn't use applications very often, I was able to understand what was going on. The only thing that I was a bit confused on was the filters because I have never used them before.	The match process was thought through well. It was logical and easy for the user to understand. The filter features were slick and modern and gave the application a fun felt edge.
[26] Comments – With 1 being highly unsatisfied and 10 being highly satisfied, on what do you rate the match result produced?	**7** – I am satisfied with the match process. I feel that this application has serious potential to be expanded on.	**9** – I am highly delighted with my match results and cannot wait to try them out.	**5** - I am not satisfied with the results provided. As I am trained to match customers in store in my profession, I know that the match required is not appropriate for me.
[27] Comments - Having purchased make up before, has it been valuable/acceptable?	I have purchased the foundation product that was recommended for me in that exact colour – I am happy with this result. I think it is an acceptable match.	I am highly fascinated with this applications ability to pick a match. I have not purchased a wide range of foundation products but I will definitely purchase my match. It seems to be a good result.	This application does not represent the scale of products available on the market and therefore the match that I was provided with has not suited by needs. This is probably due to my profession. For your average foundation wearer, this application meets their needs.
[28] Comments – Favourite Feature	The match – I was keen	Google Maps - It was a good idea to implement this.	Filtering the photo – This was the best feature.
[29] Comments – Least Favourite feature	I don't like the quiz – Don't see the point of it.	I don't like the quiz feature.	Google Maps – I didn't really see the need for it.

[30] Comments – Suggestions for improvements	The ability for the user to be able to try on the products through the user of a Bitmoji or possibly in the future through a virtual reality type mechanism.	The ability for the user to buy a product through the application would be an appealing feature to include. The user would not have to go elsewhere to look up the product to purchase it.	This application could incorporate more than just foundation products. The addition of cosmetic products such as concealers and lipsticks could increase the scalability of this application.
[31] Comments – Additional Comments	Good luck for the future with this application.	Thank you. I have really enjoyed being part of this process.	All the best in the completion of this dissertation. I hope to be using this application one day. It would have a big impact on the cosmetic market.
	NOTES		
PARTICIPANT 1	The most computer literate. Finds navigating through the application easy. Enthusiastic about the filters being applied to photos. Was able to sign in and sign up to the application instantly. Quick reactions when carrying out tasks.		
PARTICIPANT 2	The least computer literate. Took 1 minute to locate the Sign Up Button. Slowest reactions when carrying out tasks. Had to be shown how to apply filters. Enthusiasm towards the match process.		
PARTICIPANT 3	Lack of enthusiasm in Google Maps. Finds the application is more geared towards woman. More than moderate IT skills observed. Steadily improved in awareness. Enthusiastic in developing this application and using it in his profession.		

Table 5: User Evaluation Results

6.3. REQUIREMENTS EVALUATION & RESULST

As the development stage was being carried out it was decided that a few of the lowest priority requirements would not be implemented due to the time constraints. This decision was discussed with the stakeholders and it was concluded that eliminating these requirements would not have an overall effect on the overall product quality. The status of these requirements was updated on Trello and categorised into a “Will Not Implement” section. This was to distinguish between which requirements were to be worked on and which requirements were not. There was a total of 20 initial Functional Requirements, 17 of these requirements were implemented and 3 of these were not. Table 6 contains the functional requirements not implemented along with a rationale as to why.

ID	DESCRIPTION	STATUS	RATIONALE
F02	The application shall allow the user to place a foundation order from a third party seller.	Will not Implement	There was no time to implement this feature. The development effort for this requirement would extend the initial delivery of the software.
F19	The application shall provide the user with product dupes.	Will not implement	This feature was simply a surprise and delight feature and would have required the developer to spend time matching particular foundation dupes based on their own personal judgement as there is no list available. This would have been a very time consuming process for a small and not needed feature.
F20	The application shall provide product ingredients to the user.	Will not implement	This feature would have required the developer to input large amounts of product data ingredients list into the database. This would have been a very time consuming task which was not required for such a low priority feature.

Table 6: Functional Requirements not implemented

Non-functional requirements differ slightly in the sense they cannot capture the criteria needed to sign off the requirement as complete. A non-functional evaluation was done to counteract this where each requirement was tested in a black box fashion. The results of this can be seen in table 7

ID	DESCRIPTION	TYPE	EVALUATION METHOD	RESULT
NF06	A valid login transaction response shall occur within 2 seconds of the request when the system is under normal and peak transaction loads.	Performance	The login process time was monitored using the Firebase console.	99% of login requests took less than 2 seconds to complete under normal and peak transaction load times.
NF07	The application must provide the user with a colour match within 5 seconds.	Performance	The process was carried out over 50 times and the time recorded to receive a match was recorded and averaged.	100% of colour matches were provided within this 5 second window.
NF08	The application is available on an Android platform across different versions.	Operational	The application should be tested on more than one device running different versions of Android.	The application was tested on both an Android phone and an Android tablet one running Android 5.0 and the other Android 6.0
NF14	The password must be a minimum of 8 characters and contain an upper case letter, a special character and a number.	Security	Authentication mock data was used to test that this was not possible with validation checks.	The user cannot login to the system without creating their password this way.

Table 7: Non-Functional Requirement Evaluation

6.4. CONSIDERATION FOR FUTURE WORK

As the Findation application is a working prototype the system has been built for extension. There are some features that could have been implemented outside the scope of this project to further enhance the applications capabilities. In addition to these desirable features there are also improvements to the functionality within this application that could improve the overall product quality. These future improvements and desirable additional features include:

ADDITIONAL FEATURES

[1] **Offline support**: The current application relies on a constant internet connection. However, there is some functionality provided by the Firebase API to enable certain parts of the application to be used when the user is offline. The user data could be stored locally on the device enabling the user to still login into their account and view their profile.
[2] **Virtual Reality Mechanism**: This feature could be implemented to provide the user with the in store experience they desire when purchasing a foundation product. It would also enable the user to try on the foundation shades in different lighting.
[3] **Ability to purchase the product**: This feature could be implemented to provide the user easy access from the one application to purchase the foundation instead of navigating to the product website. This would save time and also enable the user to track the purchased product information.
[4] **Expand on products**: The current application only offers matching services for foundation products. However, additional make up products such as concealers, lipsticks and eye shadows can be matched or suggested for particular skin tones. The applications scope could be extended to provide matches or recommendations on all cosmetics.

FUTURE IMPROVEMENTS

[1] **Advanced colour matching API**: The Imagga API provided sufficient information for the colour matching process to take place. However, there are more scalable used colour matching technologies available such as LTU which would have provided more accurate results.
[2] **More extensive user profile**: It would be more suitable to create a more in depth user profile which could be linked to the user's social media account where they could receive updates on latest products and share them with their friends. Pigmentation analysis within this user profile could be recorded and represented on graphs so the user can visualise their skin data.
[3] **Development for iOS:** As this product was created solely to be ran on the Android platform the customer base was limited. This product could also be developed for the iOS development platform which would increase the products market capability.
[4] **Matching Algorithm:** A look up was implemented to map particular foundation shades to the user's skin tone to carry out the matching process. However, in hindsight, a more effective approach would be to implement a matching algorithm which is based on hexadecimal values. This would provide a more appropriate match instead of using the developer's judgement on predicting the corresponding hexadecimal values to particular foundation shades.
[5] **Additional Unit and Integration Testing:** The developer would have liked to have implemented more unit and integration testing but due to the time limitations of the project this was not feasible. However, with the use of more unit and integration testing, the Java code would be more robust and there is less potential for bugs to arise in the software.

7. CONCLUSIONS

7.1. CRITICAL APPRAISAL OF THE PROJECT

The scope of the project was large and at the beginning of the project it seemed daunting to the developer due to its large feature set. The scope of this project would have not been feasible without a concrete development methodology, appropriate project planning and an extensive test plan to prove the quality of the features and document evidence that both the functional and non functional requirements have been satisfied. All of the project objectives where completely fulfilled. However, due to the time constraint of this project and pressure from additional course modules, some requirements where only partially fulfilled or completely unfulfilled. This did not directly impact the end software product as the features that were not successfully implemented had an extremely low priority rating.

In terms of project outcomes, the development of this application and the implementation of new technologies that the developer had no previous experience with made the whole experience extremely beneficial. It enhanced the developer's software development skillset and their ability to plan, manage and evaluate a product. This skill set will be invaluable to the developers' future career within the IT industry. The additional use of computing theories and Engineering standards, especially the use of the CMMI-DEV model of best practices, helped to gauge the quality of the processes enacted to create the Findation Application. The end software product is a fully functional android application which conforms to Androids best practices and Googles Java coding standards. It is a modern idea which has large market opportunity and remediates cosmetic users of the frustrations identified in chapter one and two of this report

7.2. REFLECTION ON PROJECT PLAN

There is a paramount importance to reflect on the processes and technologies used in this project. This reflection enables the developer to progress, learn from what has been created, to synthesise and define what comes next. A reflection has been carried out on the software methodology, and technology stack

<u>METHODOLOGY</u>

The Revised Waterfall lifecycle model was enacted throughout this project. The sequential and defined steps of this lifecycle model enabled the developer to set strict timeframes for each defined step from the planning to the evaluation stage. A lengthy timeframe assigned to the requirements elicitation stage helped reduce poor requirements management and a requirements creep, keeping the scope of the project within its expected time scale. With this defined process in place and a strong initial Volere requirements specification established there were no requirements change requests, despite a change request procedure being enabled. By adhering to areas of the CMMI, more particularly its Requirements Development (RD), Requirements Management (REQM) and its Decision Analysis and Resolution (DAR) process areas, the quality of the work product produced was to a high standard. DAR also enabled the selection and decision of all aspects of this project. The HOOD

capability model enabled for a set of requirements that adhered to a quality baseline check and with the use of Kano Analysis and Relative Weighting, the voice of the customer was heard. When these engineering standards are not enacted, there is no set procedure or process for carrying out parts of the development methodology which leads to indecisions and mistakes.

On reflection of the choice of development methodology, the Revised Waterfall method was an appropriate selection due to its rigid structure. Its logical step wise process was also appropriate for an individual project. It also was the best selection for this developer as they like to work through tasks in this logical way, completing each step sequentially. However, in hindsight, the PXP methodology would have been another good choice for this project. This is due to its flexible nature and its large capacity to extend and improve the project to become as maintainable as possible. Its rigorous testing ethos would have been extremely beneficial in the testing phase of this project and using its test driven development approach throughout development would have enabled testing to be carried out throughout the lifecycle rather than solely at the end of implementation. This would have ensured that every feature was well written, well tested and furthered the confidence of the developer and stakeholders in the end product. The completion of user stories in small chunks also gives a good indication of project progress with the application growing after every sprint iteration. The comparison of methodologies in chapter 4 of this project definitely aided in the selection of the appropriate methodology choice and the developer would use this comparison again if implementing a future project.

TECHNOLOGIES

Extensive research was carried out into the appropriate technology stack to be used in the development of the Findation Application. The use of Firebase, Google Maps, and Colour extraction APIs includes some of the most popular and comprehensive technologies used today in native mobile application development. Both the developer and the stakeholders agreed that this application would not have been suitable to be implemented as a web application due to the need of the camera facility from a mobile device. The selection to develop the application for the Android platform was a successful decision due to the developer's familiarity and experience with this technology. However, in hindsight only developing the application for Android limited the user base. Another downfall of implementing this technology was the need to purchase additional hardware, an android device. The developer is an Apple advocate and had no access to a device which was able to run the appropriate Android version required for the development of this application.

AWS provided a cheap and powerful way to store the images in the cloud to enable the colour match to be performed by the Imagga API. It also gave the developer experience in creating an S3 storage bucket which is a widely used and powerful way of storing objects in the industry. There was never any downtime for this service which satisfied the need of this applications availability. Firebase provided a successful way to deal with user authentication and the availability of data in real time. It also removed the need for referential integrity and strict keying like SQL

databases. This prevented data persistence errors when the data was required to be changed within the system to calculate the colour match. As this product is a prototype application, the use of the Imagga API was an appropriate selection as it enabled the required functionality to be carried out and only cost £10 a month to implement. However, there are larger and costlier colour extraction APIs available which are more accurate and would provide a larger range of colour palettes. This would have increased the end user's satisfaction with the end product as it would have provided more accurate results. On reflection, for this projects purpose the project could not have been completed in a better way than by using its selected technology stack. The developer's familiarities with these technologies ensured that the product produced was to the best of the developer's ability. This enabled them to produce the best version of the application they could within the given time frame.

7.3. REFLECTION ON APPROPRIATENESS OF TIME/EFFORT ESTIMATION

As the Revised Waterfall methodology was adhered to in this project, each stage of the life cycle model was broken up into different work activities. The original project plan that was created for the initial report was used for the proposal, planning, analysis and design stages of the project and the project deadlines set in this schedule were adhered to. However, the implementation start date which was set in the initial plan had to re-evaluated after the Christmas period as development did not start on the 12th of December as originally stated. This was due to the prioritisation of exams. A revised Gantt chart was created on the 20th of January which can be found in Appendix X. The new project deadlines were achievable and everything stayed on track throughout the rest of the project with this revised project plan.

The developer had a lot of additional work to carry out while working on the project but the revised development times were managed accordingly and the stakeholders and mentor deadlines were met. The stakeholders were updated on progress via the Trello tracking board. Story point metrics were assigned to each task alongside an estimated development time. This enabled steady progress throughout the full implementation stage. The Gantt chart allocated plenty of time to complete the development within a 60-day period.

Considering that the project required a large amount of time management with meetings, reviews and evaluations, a large degree of discipline was required. The developer's ability to use their time effectively and prioritise tasks has greatly improved and this soft skill set will be invaluable to the developer in the future. Overall, the developer feels that the project was managed steadily. However, one criticism on the management of the project would be the developers lack of planning on specific features. It would have been more beneficial to implement one feature at a time rather than start implementing parts of multiple features at once. This would have reduced the larger work loads nearing the end of the implementation stage and the long hours that were involved in this period could have been prevented. This could have been mitigated more appropriately by following an agile work flow. This work flow would have allocated work accordingly to particular sprint iterations.

8. BIBLIOGRAFY/REFERENCES

Agile Alliance. (2017). *What is Agile Software Development?*. [online] Available at: https://www.agilealliance.org/agile101/ [Accessed 6 Apr. 2017].

Agilemethodology.org. (2008). *The Agile Movement*. [online] Available at: http://agilemethodology.org/ [Accessed 14 Apr. 2017].

Amazon Web Services (2017). *Amazon Simple Storage Service (S3) — Cloud Storage — AWS*. [online] Amazon Web Services, Inc. Available at: https://aws.amazon.com/s3/ [Accessed 3 Apr. 2017].

Amazon Web Services (2017). *What is AWS? - Amazon Web Services*. [online] Amazon Web Services, Inc. Available at: https://aws.amazon.com/what-is-aws/ [Accessed 5 Apr. 2017].

Ambler, S. (2014). *Constraints: An Agile Introduction*. [online] Agilemodeling.com. Available at: http://www.agilemodeling.com/artifacts/constraint.htm [Accessed 4 Apr. 2017].

Android (2014). *Android – History*. [online] Android. Available at: https://www.android.com/intl/en_uk/history/#/marshmallow [Accessed 10 Apr. 2017].

Android (2017). *android.database.sqlite | Android Developers*. [online] Developer.android.com. Available at: https://developer.android.com/reference/android/database/sqlite/package-summary.html [Accessed 8 Apr. 2017].

Android (2017). *Configure Your Build | Android Studio*. [online] Developer.android.com. Available at: https://developer.android.com/studio/build/index.html [Accessed 8 Apr. 2017].

Android (2017). *Drag and Drop | Android Developers*. [online] Developer.android.com. Available at: https://developer.android.com/guide/topics/ui/drag-drop.html [Accessed 8 Apr. 2017].

Android Developers (2017). *Dashboards | Android Developers*. [online] Developer.android.com. Available at: https://developer.android.com/about/dashboards/index.html [Accessed 2 Apr. 2017].

Android Developers (2017). *Device Compatibility | Android Developers*. [online] Developer.android.com. Available at: https://developer.android.com/guide/practices/compatibility.html [Accessed 2 Apr. 2017].

Android, M. (2017). *Material Design for Android | Android Developers*. [online] Developer.android.com. Available at: https://developer.android.com/design/material/index.html [Accessed 14 Apr. 2017].

Anon, (2017). *Kano Analysis Model*. [online] Available at: http://people.ucalgary.ca/~design/engg251/First%20Year%20Files/kano.pdf [Accessed 2 Apr. 2017].

App.smartsheet.com. (2017). *Smartsheet*. [online] Available at: https://app.smartsheet.com/b/home [Accessed 2 Apr. 2017].

Atlantic Systems Guild (2017). *Experience*. [online] Volere.co.uk. Available at: http://www.volere.co.uk/experience.htm [Accessed 7 Apr. 2017].

Berry, S. and Thomas, R. (2016). *Use SMART objectives to focus goals, plans and performance*. [online] Project Smart. Available at: Available from: https://www.projectsmart.co.uk/use-smart-objectives-to-focus-goals-plans-and-performance.php [Accessed 6 Apr. 2017].

BRUEGGE, B. (2006). *Software Lifecycle Models*. 1st ed.

Bunting, N. (2015). Approaches to Reflection. [Blog] *The IB Community Blog*. Available at: http://blogs.ibo.org/blog/2015/10/31/approaches-to-reflection/ [Accessed 16 Apr. 2017].

CDC (n.d.). *Change Management Plan*. [online] www2a.cdc.gov. Available at: https://www2a.cdc.gov/cdcup/library/templates/CDC_UP_Change_Management_Plan_Template.doc [Accessed 8 Apr. 2017].

CMMI for DEV. (2014). 1st ed. Kornwestheim: Kugler Maag Cie.

Cogswell, J. (2014). *Google's Android Studio vs. Eclipse: Which Fits Your Needs? - Dice Insights*. [online] Dice Insights. Available at: http://insights.dice.com/2014/03/19/googles-android-studio-vs-eclipse-fits-needs/ [Accessed 6 Apr. 2017].

Cohn, M. (2004). *Project Advantages of User Stories as Requirements*. [online] Mountain Goat Software. Available at: https://www.mountaingoatsoftware.com/articles/advantages-of-user-stories-for-requirements [Accessed 8 Apr. 2017].

Collofello, J. (1998). *Introduction to Software Verification and Validation*. 1st ed. Arizona: Carnegie Mellon University Software Engineering Institute.

Cosmeticsbusiness.com. (2013). *Boots UK rolls out revolutionary system of colour matching No7 cosmetic foundations with skin tones of customers using X-Rite technology*. [online] Available at: https://www.cosmeticsbusiness.com/news/article_page/Boots_UK_rolls_out_revolutionary_system_of_colour_matching_No7_cosmetic_foundations_with_skin_tones_of_customers_using_XRite_technology/85993 [Accessed 1 Apr. 2017].

Creately.com. (2017). *Online Diagram Software to draw Flowcharts, UML & more | Creately*. [online] Available at: https://creately.com/ [Accessed 3 Apr. 2017].

Curtis, S. (2014). *How technology is transforming the cosmetics industry*. [online] Telegraph.co.uk. Available at: http://www.telegraph.co.uk/technology/news/11146752/How-technology-is-transforming-cosmetics.html [Accessed 4 Apr. 2017].

Davis, A., Dieste, O., Hickey, A., Juristo, N. and Moreno, A. (2006). Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review. *14th IEEE International Requirements Engineering Conference (RE'06)*.

Decision Analysis and Resolution (DAR) and its use and benefit in projects and org level as per CMMI. (2013). [Blog] *CMMI Consultant Blog*. Available at: http://www.cmmiconsultantblog.com/cmmi-faqs/decision-analysis-and-resolution-dar-and-its-use-and-benefit-in-projects-and-org-level-as-per-cmmi/ [Accessed 4 Apr. 2017].

Dekkers, C. (2013). *Fundamentals of Software Metrics in Two Minutes or Less*. [online] QSM SLIM-Estimate. Available at: http://www.qsm.com/blog/2013/fundamentals-software-metrics-two-minutes-or-less [Accessed 2 Apr. 2017].

Developer.android.com. (2017). *Dashboards | Android Developers*. [online] Available at: https://developer.android.com/about/dashboards/index.html [Accessed 3 Apr. 2017].

Developer.android.com. (2017). *Device Compatibility | Android Developers*. [online] Available at: https://developer.android.com/guide/practices/compatibility.html [Accessed 3 Apr. 2017].

Developer.apple.com. (2014). *Core OS Layer*. [online] Available at: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1 [Accessed 2 Apr. 2017].

Draw.io. (2017). *Flowchart Maker & Online Diagram Software*. [online] Available at: https://www.draw.io [Accessed 2 Apr. 2017].

Fakhruddin, H. (2015). *Eclipse vs Android Studio: Which IDE Is Better For Android Developers?*. [online] Teknowledge Software : iPhone Application Development Company India. Available at: http://teks.co.in/site/blog/eclipse-vs-android-studio-ide-better-android-developers/ [Accessed 7 Apr. 2017].

Firebase (2017). *Firebase Authentication | Firebase*. [online] Firebase. Available at: https://firebase.google.com/docs/auth/ [Accessed 14 Apr. 2017].

Firebase (2017). *Structure Your Database | Firebase*. [online] Firebase. Available at: https://firebase.google.com/docs/database/web/structure-data [Accessed 8 Apr. 2017].

Flylib.com. (2008). *Look and Feel Requirements: Type 10 - Mastering the Requirements Process (2nd Edition)*. [online] Available at: http://flylib.com/books/en/4.445.1.138/1/ [Accessed 3 Apr. 2017].

Fowler, M. and Highsmith, J. (2001). *The Agile Manifesto*. [online] Available at: http://dimsboiv.uqac.ca/8INF851/web/part1/introduction/The_Agile_Manifesto.pdf [Accessed 3 Apr. 2017].

Goguen, J. and Linde, C. (n.d.). Techniques for requirements elicitation. *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*.

Gosling, J., Joy, B., Steele, G. and Bracha, G. (2000). *The Java Language Specification*. 2nd ed. California: Addison Wesley.

Hood, C., Wiedemann, S., Fichtinger, S. and Pautz, U. (2008). *Requirements management*. 1st ed.

Hulse, A. (2015). *How to write Performance Requirements with Example | 1202Performance*. [online] 1202performance.com. Available at: http://www.1202performance.com/atricles/how-to-write-performance-requirements-with-example/ [Accessed 2 Apr. 2017].

Imagga.com. (2017). *Imagga - powerful image recognition APIs for automated categorization & tagging*. [online] Available at: https://imagga.com/ [Accessed 2 Apr. 2017].

Imgly (2017). *Getting started with our PhotoEditor SDK for Android.*. [online] Photo Editor SDK. Available at: https://www.photoeditorsdk.com/documentation/android/getting-started [Accessed 13 Apr. 2017].

ITO, K., MATSUSHIRO, N. and YOBAYASHI, Y. (1998). Colour Matching Technology. *OKI Technical Review*, 64.

Jamro, M. (2015). *POU-Oriented Unit Testing of IEC 61131-3 Control Software*. 5th ed. Poland.

Lacey, M. (2012). *Prioritization - Relative Weighting - Karl Weigers*. [online] Msdn.microsoft.com. Available at: https://msdn.microsoft.com/en-us/library/hh765981(v=vs.120).aspx#Weigers [Accessed 2 Apr. 2017].

Lenfield, K. (2015). *Everyone is a supermodel*. 1st ed. Lucky Pineapple Books.

Levy, D. (2010). *Requirements Traceability -- Why so important?*. [online] Gatherspace.com. Available at: http://www.gatherspace.com/static/requirements_traceability.html [Accessed 7 Apr. 2017].

Liles, J. (2012). *Methods for Eliciting - Not Gathering - Requirements*. [online] Batimes.com. Available at: https://www.batimes.com/articles/methods-for-eliciting-not-gathering-requirements.html [Accessed 4 Apr. 2017].

Linux For you (2009). *A developers first look at Android*. pp.48-50.

Loreal-paris.co.uk. (2017). *Shade Genuis: Foundation SkinTone Matching*. [online] Available at: http://www.loreal-paris.co.uk/products/make-up/apps/shade-genius [Accessed 5 Apr. 2017].

Lui, J. (2016). *Formal Requirements Specification Week 6 Part 2*.

Lui, J. (2016). *Requirements Engineering Process*.

Malik, P. (2014). *What are Project Assumptions?*. [online] Pmbypm.com. Available at: http://www.pmbypm.com/what-are-assumptions/ [Accessed 3 Apr. 2017].

McFall, D. (2016). *COM583-Week 3 Soft Process Models_Part II*.

McFall, D. (2016). *Requirements Development*.

McFall, D. (2017). *Measurement*.

Mercado, Logalbo, Shidara, McKenna, McGuinness, Pardo and Owen (2017). *Patent US9549602 - Foundation makeup and concealer composition*. [online] Google Books. Available at: https://www.google.com/patents/US9549602 [Accessed 5 Apr. 2017].

Mysql.com. (2017). *MySQL :: MySQL Workbench*. [online] Available at: https://www.mysql.com/products/workbench/ [Accessed 1 Apr. 2017].

Ndegwa, A. (2016). *What is a Web Application?*. [online] Maxcdn.com. Available at: https://www.maxcdn.com/one/visual-glossary/web-application/ [Accessed 5 Apr. 2017].

Nielsen, J. (1993). *Usability engineering*. 1st ed. [Place of publication not identified]: Morgan Kaufmann.

NO7 MATCH MADE SERVICE. (2013). [Blog] *30SomethingMel*. Available at: http://www.30somethingmel.co.uk/spotlight-no7-match-made-service/ [Accessed 6 Apr. 2017].

OfCom (2015). *The UK is now a smartphone society*. [online] Ofcom. Available at: https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2015/cmr-uk-2015 [Accessed 8 Apr. 2017].

Oracle (2017). *MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL?*. [online] Dev.mysql.com. Available at: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html [Accessed 6 Apr. 2017].

Ostrand, T. (2002). White-Box Testing. *Encyclopedia of Software Engineering*.

Patten, N. (2016). Boots No7 New Match Made App Review. [Blog] *NiaPattenLooks*.

PENALBA, J. (2010). *Requirements Engineering Process*.

PIOUS, K. (2013). Techniques for Eliciting Quality Requirements – Observation. [Blog] *Cap Tech Blog*. Available at: https://www.captechconsulting.com/blogs/techniques-for-eliciting-quality-requirements--observation [Accessed 5 Apr. 2017].

PTG Media (2017). *Volere Shell Description*. [image] Available at: http://ptgmedia.pearsoncmg.com/images/chap2_9780321815743/elementLinks/2_9_volere_snow_card_alt.jpg [Accessed 4 Apr. 2017].

Readfaster.com. (2017). *Education Quotes - A Great Collection of Quotes Relating to Education - The Literacy Company*. [online] Available at: http://www.readfaster.com/educationquotes.asp [Accessed 2 Apr. 2017].

Rouse, M. (2007). *What is waterfall model? - Definition from WhatIs.com*. [online] SearchSoftwareQuality. Available at: http://searchsoftwarequality.techtarget.com/definition/waterfall-model [Accessed 8 Apr. 2017].

Rozanski, N. and Woods, E. (2017). *Stakeholders*. [online] Software systems Architecture. Available at: http://www.viewpoints-and-perspectives.info/home/stakeholders/ [Accessed 7 Apr. 2017].

Siddiqui, S. (2016). *Metrics for Requirements Engineering and Automated Requirements Tools*. [online] Available at: http://www.academia.edu/1297947/Metrics_for_Requirements_Engineering_and_Automated_Requirements_Tools [Accessed 4 Apr. 2017].

Software Testing Fundamentals (2017). *Differences Between Black Box Testing and White Box Testing – Software Testing Fundamentals*. [online] Softwaretestingfundamentals.com. Available at: http://softwaretestingfundamentals.com/differences-between-black-box-testing-and-white-box-testing/ [Accessed 14 Apr. 2017].

Software Testing Help. (2016). *Defect Density Guide*. [online] Available at: Software Testing Help. Available from: http://www.softwaretestinghelp.com/defect-density/ [Accessed 4 Apr. 2017].

Software-quality-assurance.org. (2017). *CMMI - Verification (VER) Process Area*. [online] Available at: http://www.software-quality-assurance.org/cmmi-verification.html [Accessed 1 Apr. 2017].

Solutions, N. (2017). *Distributed Agile Development Process | Agile Methodology*. [online] Netsolutionsindia.com. Available at: https://www.netsolutionsindia.com/distributed-agile [Accessed 2 Apr. 2017].

Storkel, S. (2002). *An Introduction to the Eclipse IDE - O'Reilly Media*. [online] Onjava.com. Available at: http://www.onjava.com/pub/a/onjava/2002/12/11/eclipse.html [Accessed 5 Apr. 2017].

TechTarget (2008). *How to effectively elicit user interface requirements*. [online] SearchSoftwareQuality. Available at: http://searchsoftwarequality.techtarget.com/answer/How-to-effectively-elicit-user-interface-requirements [Accessed 3 Apr. 2017].

TechTerms (2017). *OOP (Object-Oriented Programming) Definition*. [online] Techterms.com. Available at: https://techterms.com/definition/oop [Accessed 8 Apr. 2017].

TinEye (2017). *About - TinEye*. [online] Tineye.com. Available at: https://tineye.com/about [Accessed 5 Apr. 2017].

TinEye (2017). *MulticolorEngine: Color Search and Extraction*. [online] Services.tineye.com. Available at: https://services.tineye.com/MulticolorEngine [Accessed 8 Apr. 2017].

TortiseSVN (2017). *Preface*. [online] Tortoisesvn.net. Available at: https://tortoisesvn.net/docs/nightly/TortoiseSVN_en/tsvn-preface.html#tsvn-preface-about [Accessed 12 Apr. 2017].

Tutor2u (2012). *Critical Path Analysis*.

Varma, P. (2016). *What is Firebase?*. [online] Quora. Available at: https://www.quora.com/What-is-firebase [Accessed 2 Apr. 2017].

Weinswig, D. (2016). Beauty Shopping Augmented With New Mobile Apps. *Forbes*. [online] Available at: https://www.forbes.com/sites/deborahweinswig/2016/07/28/beauty-shopping-augmented-with-new-mobile-apps/#26b5b61b56e0 [Accessed 3 Apr. 2017].

Wells, D. (1999). *Extreme Programming Rules*. [online] Extremeprogramming.org. Available at: http://www.extremeprogramming.org/rules.html [Accessed 5 Apr. 2017].

Wibas.com. (2015). *Process and Product Quality Assurance (PPQA) (CMMI-DEV)*. [online] Available at: https://www.wibas.com/cmmi/process-and-product-quality-assurance-ppqa-cmmi-dev [Accessed 5 Apr. 2017].

Wibas.com. (2015). *Validation (VAL) (CMMI-DEV)*. [online] Available at: https://www.wibas.com/cmmi/validation-val-cmmi-dev [Accessed 2 Apr. 2017].

Wiegers, K. (2012). *Validating Requirements - Enfocus Solutions Inc*. [online] Enfocus Solutions Inc. Available at: http://enfocussolutions.com/validating-requirements/ [Accessed 11 Apr. 2017].

Wodehouse, C. (2017). *SQL vs. NoSQL: What's the difference?*. [online] Upwork. Available at: https://www.upwork.com/hiring/data/sql-vs-nosql-databases-whats-the-difference/ [Accessed 7 Apr. 2017].

WONG, E. (2017). *Shneiderman's Eight Golden Rules Will Help You Design Better Interfaces*. [online] The Interaction Design Foundation. Available at: https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces [Accessed 18 Apr. 2017].

Woodhouse, C. (2017). *What is Java? Server-Side Programming Language - Android - Object-Oriented Programming*. [online] Hiring | Upwork. Available at: https://www.upwork.com/hiring/development/the-java-platform/ [Accessed 8 Apr. 2017]

.

Woorisol.kyungpook.ac.kr. (n.d.). *Chapter 7. Requirements Definition and Specification*. [online] Available at: http://woorisol.kyungpook.ac.kr/lab/prof/softeng/ch7.htm [Accessed 10 Apr. 2017].

Workbreakdownstructure.com. (2014). *What is a Work Breakdown Structure*. [online] Available at: https://www.workbreakdownstructure.com [Accessed 5 Feb. 2017].

Yeates, S. (2013). *What is version control? Why is it important for due diligence?*. [online] Oss-watch.ac.uk. Available at: http://oss-watch.ac.uk/resources/versioncontrol [Accessed 10 Apr. 2017].

YEOMANS, M. (2013). *Global beauty market to reach $265 billion in 2017 due to an increase in GDP*. [online] CosmeticsDesign.com USA. Available at: http://www.cosmeticsdesign.com/Market-Trends/Global-beauty-market-to-reach-265-billion-in-2017-due-to-an-increase-in-GDP [Accessed 1 Apr. 2017].