An Android App for Skin Lesion Tracking

Pedro Schulze Navarrete

Fourth Year Project Report

Computer Science with Management

School of Informatics

University of Edinburgh

2019

Abstract

In this project, an Android application with the purpose of helping users with early detection of skin cancer is developed. The concept is based around enabling users to easily compare photos of their skin spots, providing them with the tools to detect changes in appearance which could indicate a potentially dangerous skin lesion.

The project focuses on the successful completion of the main phases in the mobile app development process. These stages include design, implementation, user testing, evaluation, and eventually deployment to the Google Play Store. Part of the project also involves developing and launching a website to provide users with further guidance on how to use the application.

Acknowledgements

Many thanks to my supervisor, Bob Fisher, for his continuous support and patience throughout the project, always providing extensive yet timely feedback on all areas of the project.

I would also like to thank Daniel Man for developing the iOS version of the app and his contribution to the *TrackYourSpot* website.

Lastly, a big thank you to my family and friends for their unconditional support throughout the course of the project.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Pedro Schulze Navarrete*)

Contents

Chapter 1

Introduction

1.1 Objectives

With the overall task being the development of an Android application to monitor skin spots, we can start by breaking the tasks down further, grouping the three core goals into Technical, Evaluation and Publishing tasks.

1. **Technical Tasks:**
    1.1. Allow the user to add and name new spots, additionally, the user should be able to add new images to a spot.
    1.2. A feature to compare two images of a spot on the same screen.
    1.3. The ability to email both pictures to a doctor from within the app.
    1.4. Educational information screens towards skin cancer signs and usage of the app.
2. **Evaluation Tasks:**
    2.1. Performing Alpha and Beta testing phases for bug fixing and app improvements.
    2.2. Carry out a usability analysis with known human factor assessment methodologies.
3. **Publishing Tasks:**

3.1. Development of the *TrackYourSpot* website to accommodate for both the iOS and Android versions of the app.

3.2. Publishing a stable version of the app on the Android app store.

1.2 Definition of Terms

Some of the language used in this report can appear confusing for people unfamiliar with the Android platform. These terms are used continually throughout the report so it is recommended to pay close attention to the following definitions:

- **Google Play Store** - The Android application store/market, users can choose from over 2.5 million apps to download (*Android and Google Play statistics, development resources and intelligence*, 2019).
- **Activity** - An application component, displayed as a single screen to the user. Each screen has its own activity. Activities contain the code that interacts with the UI.
- **Fragment** - Similar to an activity, but a fragment only occupies part of the screen. Multiple fragments can be used in a screen, and fragments can be swapped within an activity.
- **XML Layout** - Each Activity has its own XML file controlling the visual appearance of the screen. The code inside the activity can interact with elements described in the XML file.
- **Android API** - In an Android context, it refers to the version number of the operating system. For example, API 23 stands for the *Marshmallow* Android version.
- **SDK** - Formally refers to a Software Development Kit, however, in the Android context, it is also used to refer to the Android version. For example, the *minimumSDK* of an app is the minimum Android version required to run the app.
- **Intent** - A messaging object to request an action from another app component. Intents are used to change from one activity to another, among others.
- **APK** - The package file format used to distribute and install an app. This file can be uploaded to the Google Play Store.

For full definitions and examples, please refer to the Android docs (*Developer Guides — Android Developers*, n.d.).

Throughout the report, Activities will occasionally be referred to by their implementation class name, for example, *CompareScreen* refers to the spot comparison screen, while *SpotImageList* refers to the screen displaying all the images of a same spot.

1.3 Related Work

This section will evaluate similar android apps in the Google Play Store, finding possible flaws with them and justifying the gap in the market for our application. After scouting the Google Play Store, three main apps seem to be dominating the skin spot tracking market. As labelled on the app store, they are:

- Miiskin - Melanoma Skin Cancer
- Molexplore "Skin Cancer App"
- SkinVision - Detect Skin Cancer

Each subsection below will analyse them individually, evaluating the respective advantages and disadvantages of each by testing out the different apps.

1.3.1 Miiskin - Melanoma Skin Cancer

With over 100,000 downloads and 1000+ reviews, it is no surprise that this app appears as the first result when searching for "Skin Cancer" in the Google Play Store. The app averages 4.2/5 stars and has a premium subscription of £2.99 a month or £23.49 a year. Figure 1.1 shows a screenshot of the app listing on the Google Play Store. The app allows taking photos of moles, setting up reminders to update a mole, and comparing two photos of a mole side by side.

Advantages:

- The user can pinpoint the exact location of body spots, this helps with identifying different spots in the same body part. Sometimes spots can look very similar, if the user adds a photo to the wrong spot, it would hinder the comparison task. This model prevents this from happening.

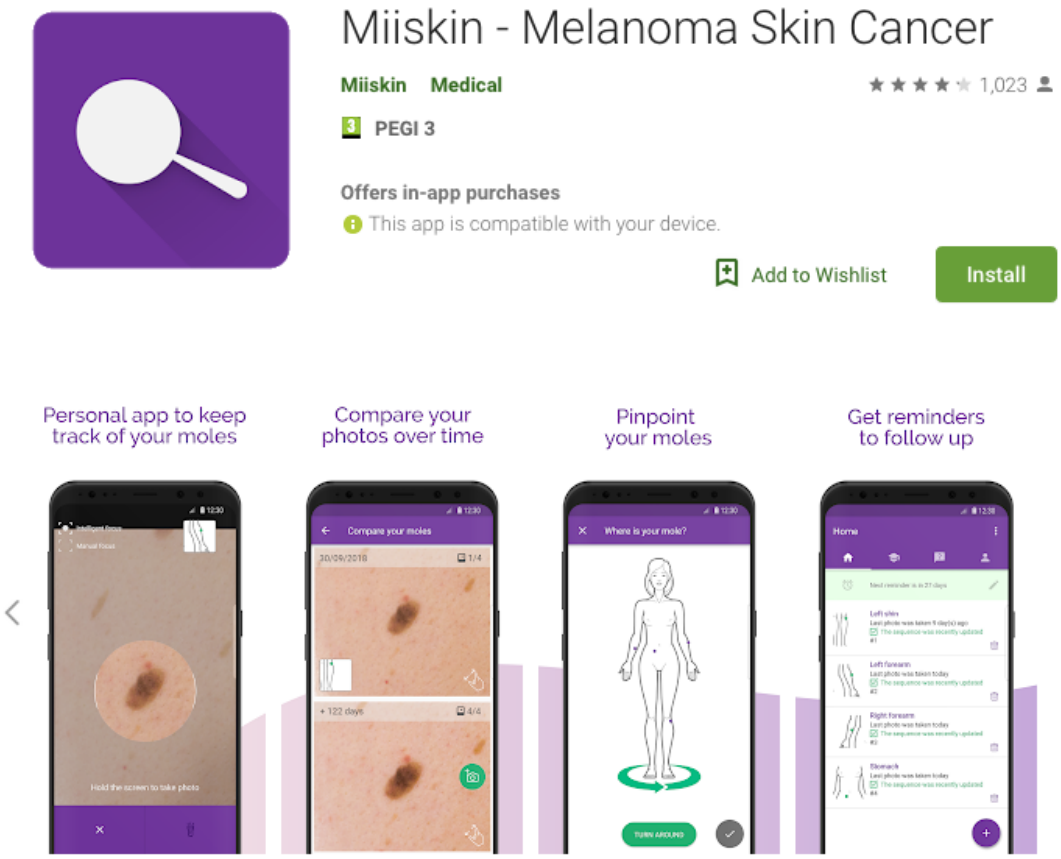


Figure 1.1: MiiSkin Google Play Store's Listing

- Ability to zoom into pictures while comparing them. This addresses the issue of users cropping images to incorrect sizes, as they can just zoom in on this screen instead.
- Reminders to take photos of a spot through push notifications. This is a particularly useful feature, as it could make the difference between one-time app users and regular users, updating their spots once every few weeks.

Disadvantages:

- While marketed as a free app, the app is only usable through a trial version, requiring payment details before registering. The user's photos become inaccessible once the free trial version runs out, frustrating many of its users.
- Users are forced to register if they want to use the app, this can put many people off who just want to quickly try out the app, as they might only be interested in the offline features of the app.

1.3.2 Molexplore "Skin Cancer App"

Second in the list, one can find this skin spot tracking app. The app has 50,000 downloads and 347 reviews (Figure 1.2), averaging 4/5 stars. The app offers similar core features as MiiSkin, it allows pinpointing the exact location of spots, comparing images side by side and offering general skin health information. The app is completely free to use.

Advantages:

- The app provides very helpful insight on what to look for when inspecting skin spots, for example, it offers advice on interpreting the color, symmetry, borders or diameter of a spot.
- An alternative way of comparing spots is available, it's called "Overlap Compare". This feature blends both images together to a user specified degree. This feature would be particularly useful in detecting changes in size or shape of a spot.
- The app provides real time UV radiation index information custom to the user's location. While it isn't essential for the process of adding and comparing images, it could be useful information for the user.

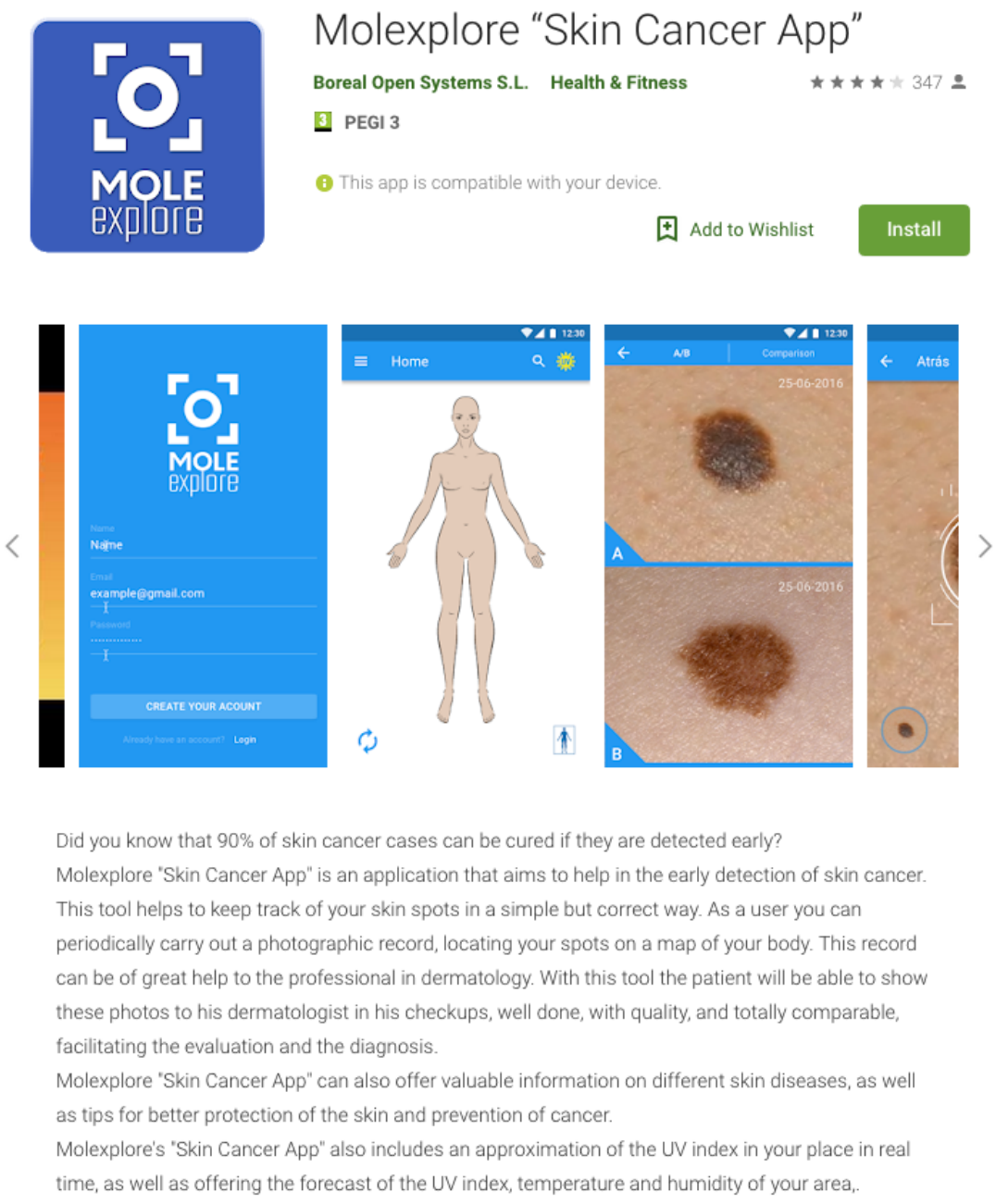


Figure 1.2: Molexplore Google Play Store’s Listing

Disadvantages:

- The app's UI can seem a bit clunky or "old". Adding or comparing spots seems to require more steps than the two previous applications.
- Even if the app doesn't offer any online features, it also requires registration before use. This is once again a drawback for some users.
- Recent user reviews have complained about frequent crashes in different areas of the app. This is an indicator of a buggy implementation or a lack of maintenance from the Molexplore team.

1.3.3 SkinVision - Detect Skin Cancer

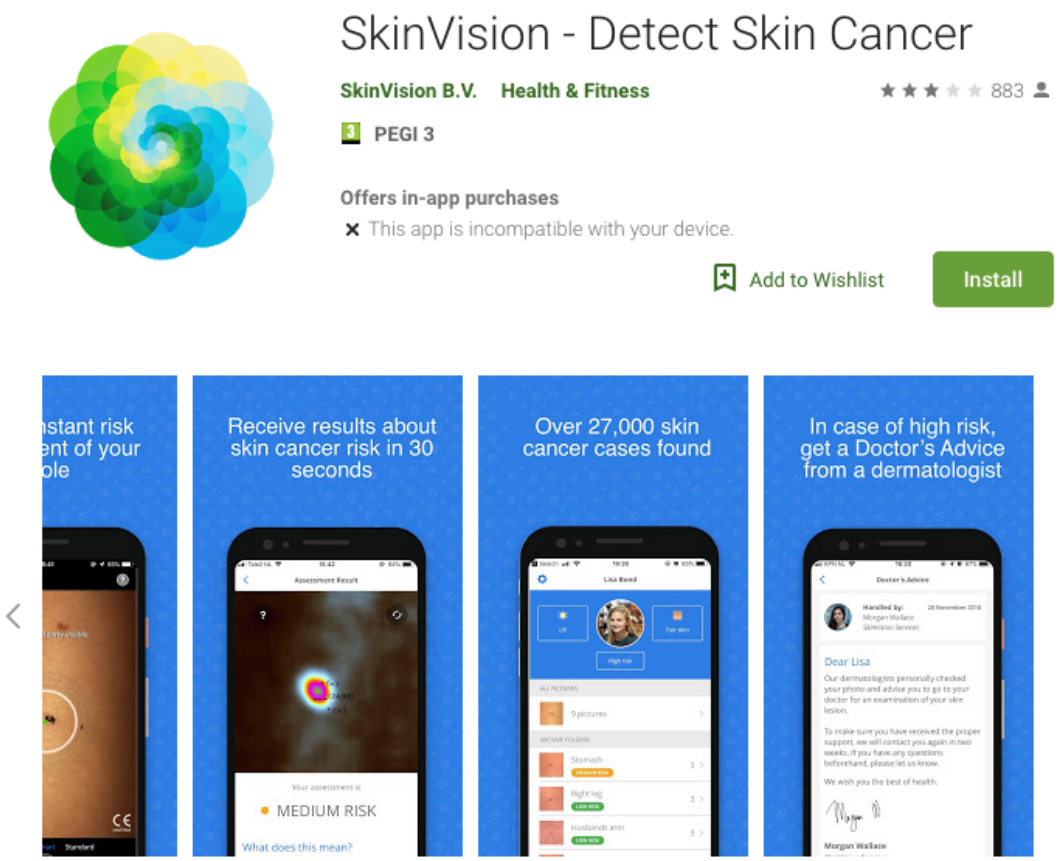
This app also surpasses 100,000 downloads. It averages 3/5 stars with 883 reviews (Figure 1.3). Contrarily to the previous two apps, this app provides a slightly different service. The app offers a "clinically validated algorithm" that actually detects instances of skin cancer. The app's business model is also quite unique. It offers one free "Smart Check", giving the user a risk indication on a photo of a skin spot, risky spots are also inspected by their team of dermatologists to provide further advice. After this free trial, users can choose to pay £3.99 for one individual "Smart Check" or pay £21.99 a year for an unlimited amount.

Advantages:

- The app provides an actual diagnosis with a spot risk assessment and doctor attention if required. This is an incredibly valuable feature which will be realistically impossible to compete with.
- Email, SMS and push notification reminders are available for any previously added spot, this goes the extra mile in preventing the user from just "swiping away" a notification.

Disadvantages:

- The obvious main issue of this app is the price. A payment of £3.99 for a single check is a very expensive price to pay for tracking a spot. Given the service they claim to offer, it is clear the app is targeted more towards people seeking for medical attention or heavily worried about particular skin lesions.



SkinVision helps you and your family with the early detection of skin cancer. In the past 6 years, with our clinically validated algorithm, we have detected more than 27,000 cases of melanoma and other common skin cancers globally, helping ensure that thousands are able to get treatment in time.

SkinVision is the first Class 1 medical device using clinically-proven technology that is trusted by insurance companies worldwide for the early detection of skin cancer. SkinVision has partnerships with leading health insurers, skin cancer clinics and research universities in Australia, Germany, the Netherlands, and New Zealand.

ORCHA, the world's leading independent provider of health and care app reviews has assessed SkinVision as Level 4 Regulated with a score of 89%, the highest of any skin cancer app. This score is calculated on data security, clinical assurance and user experience.

Figure 1.3: SkinVision Google Play Store's Listing

1.3.4 Conclusion

It would be unreasonable to aim for an app of SkinVision's characteristics, as the both the legal implications and scope would be unrealistic for an undergraduate project such as this one. Additionally, given that our target user is more of a casual user looking to monitor his skin spots, it would be sensible to assume SkinVision in a different category. Furthermore, even if all three apps are developed and maintained by a team of developers, comparing our app to Miiskin and Molexplore would be more logical.

From the evaluations of both apps, the following five points will prove to be decisive if the app aims to compete in the skin spot tracking app market:

1. **Free Business Model** - The app will offer users a completely free way of monitoring their skin spots, no ads or in-app purchases will be offered. This business model will be exactly the same as Molexplore's, so the app's success will depend on its other features.
2. **Good Information Screens** - The app will have to offer relevant and insightful information screens about skin spot tracking.
3. **Intuitive UI** - The app's UI will have to be easy to use and intuitive. Making the app easy to understand is the first step to attract new users.
4. **Email Functionality** - The email functionality is not offered by any of the other apps. This feature could somewhat emulate SkinVision's medical attention feature, even if user's would have to input their own doctor's email address.
5. **App Stability** - Needless to say, the app being bug-free and compatible with as many devices as possible will be very important in ensuring users don't delete the app.

1.4 Report Structure

The content of this report will be the following:

- **Chapter 2 - Design:** A collection of User Interface app designs, justification for design decisions of the main screens and specific UI features, as well as interactions used within the app.

- **Chapter 3 - Architecture and Implementation:** This chapter will delve into the technical challenges of the task. Including choices for object architecture, alternative and/or unsuccessful approaches, flow of data and core algorithms.
- **Chapter 4 - Deployment:** Everything involved with publishing and branding of the app. It also contains a section dedicated to the *TrackYourSpot* website.
- **Chapter 5 - User Testing and Evaluation:** Code Testing phases and exploring the human factor element of the project. Analysis and results of usability testing methods and evaluation of these are also included.
- **Chapter 6 - Conclusion:** Overall result of the project, completion of initial goals and future work.

Chapter 2

Design

2.1 Main Screens

This chapter will describe the design decision for each of the screens that can be reached within the app. Each subsection will address how the design progressed over time and any alternative designs considered. Each screen's final design is justified through also analysing its benefits and limitations. In order, the main screens analysed below are:

1. **Body Screen** - App home screen for body part selection
2. **Information Screens** - Tutorial screen for first time users
3. **Old Spot Screen** - List of spots added to a chosen body part
4. **Camera and Cropping Screen** - Screens used to add new spot photos
5. **Spot Naming Screen** - Screen to name a new spot to be saved
6. **Spot Image List Screen** - List of all photos taken of a spot
7. **Comparing Spots Screen** - Comparison screen for two side by side spot images
8. **Email Screen** - Screen for emailing selected images to a doctor

2.1.1 Body Screen

This screen acts as the app homepage, it is the first screen displayed to the user (excluding the first time use tutorial). This screen is the result of a series of design refactors that occurred within the first few weeks of development. These are explained in depth in section 2.2.2.

The final design for the body screen is displayed in Figure 2.1. The toolbar includes an "i" button, which refers the user to the information screens (Section 2.1.2). Pressing a body part button such as "Right Arm" would take the user to the old spot list screen (Section 2.1.3). To switch between the front and back body perspectives, the user can simply swipe the screen or tap the "front" and "back" body tabs.

Designs for the actual body image changed throughout the whole duration of the project. This came as a result of beta testing, usability tests and preference tests. The evolution of the screen is displayed in Figures 2.2-2.6. Designs are numbered 1-5 and a short justification for each approach is included below.

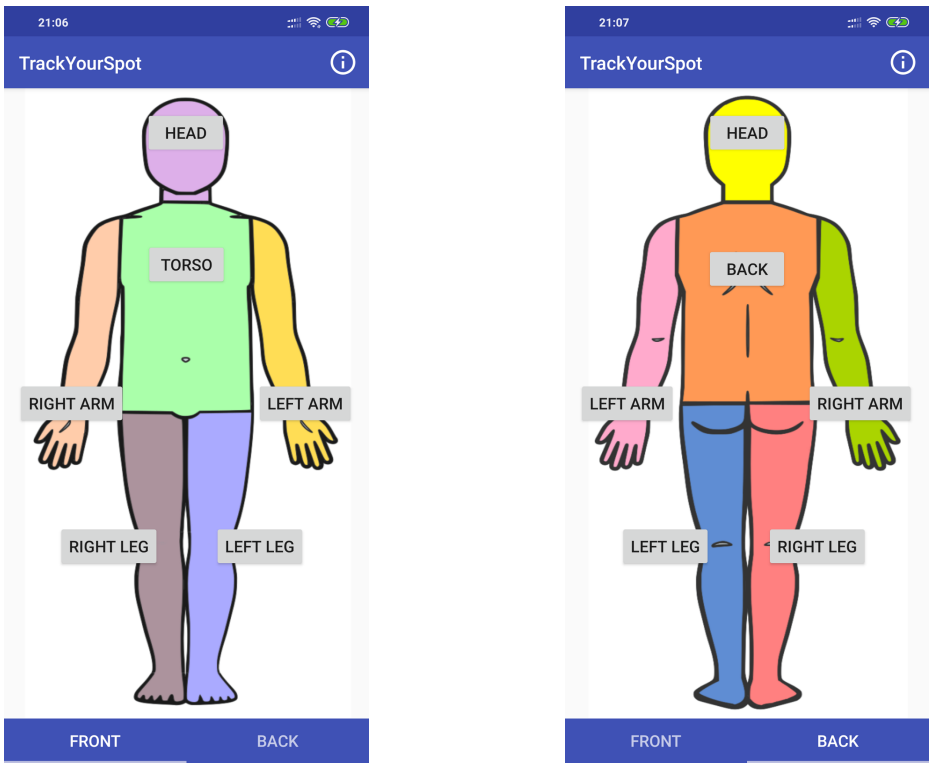
1. **Design Iteration 1** - Used as a placeholder for the eventual official screen, this image was simply used as a background to make the app more graphical during early stages of development. The same image was used for front and back.

2. **Design Iteration 2** - Due to limited artistic skills, a decision was made to find an external source for the body images. After browsing through the iStock photo library (Service offering copyright free images), this body outline was chosen. It provided a back and front side perspective with a simple, clean look. Consent was requested through the university's image graphic design department.

3. **Design Iteration 3** - Getting closer to the final design, the changes to this design had the goal of making the image and app more gender neutral, this was done by widening the hips and removing the pectoral marks. This process required learning and using *Inkscape* (Vector graphics editor).

4. **Design Iteration 4** - It was decided to add some indication of where the app separates body parts, this would become important if the buttons were hidden, and it would make it even more clear that spots are saved under different body parts.

5. **Design Iteration 5** - At this stage, there was the idea of adding color to the



(a) Front Body Screen

(b) Back Body Screen

Figure 2.1: App homepage body screen

different body parts, indicating the separations even more, and replicating the design of an unfinished prototype app by Maunik Desai. The choice of which final design to use would be left in the hands of end-users, this would be done via usability evaluations and preference testing.

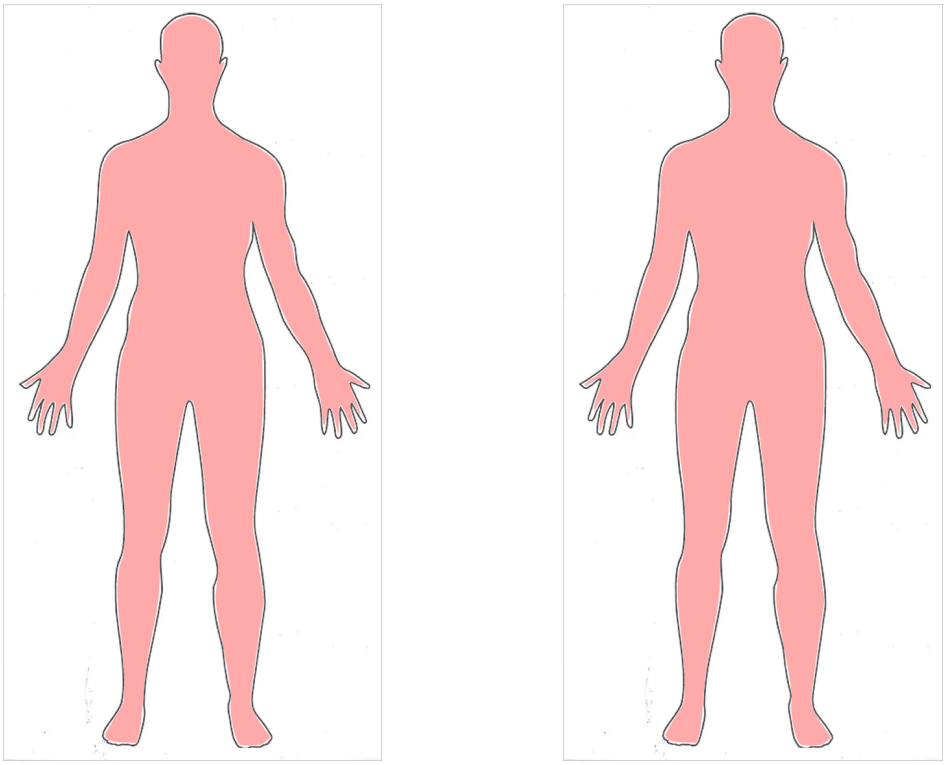


Figure 2.2: Iteration 1 Body Designs

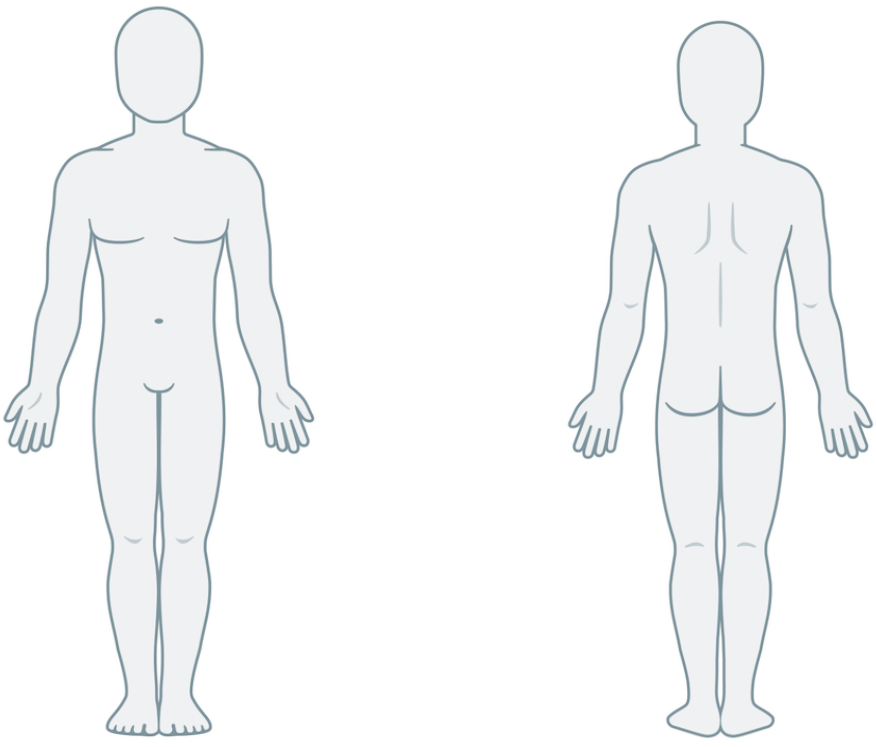


Figure 2.3: Iteration 2 Body Designs

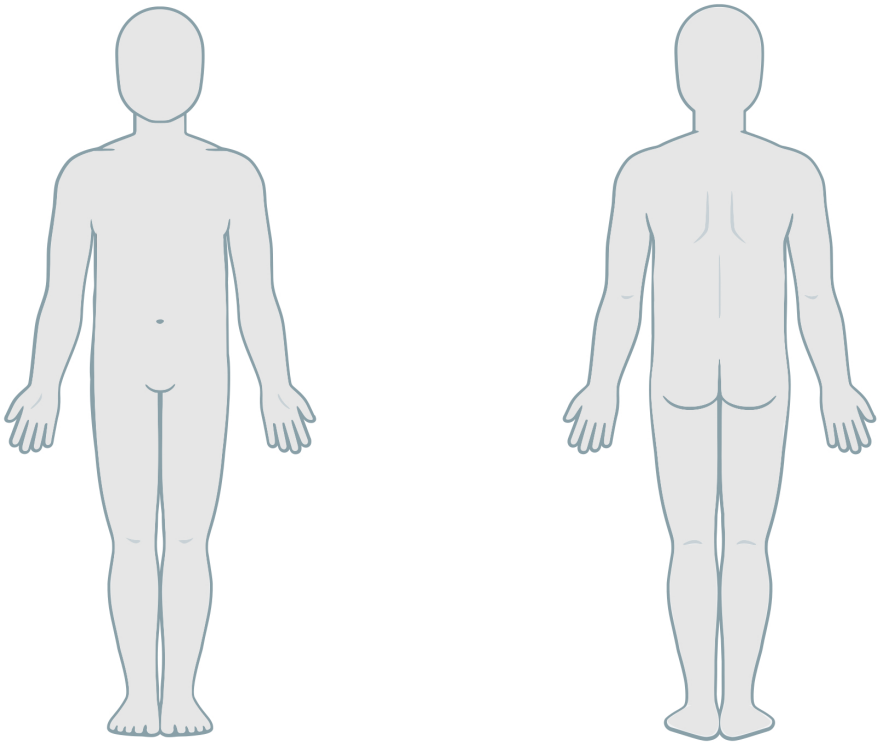


Figure 2.4: Iteration 3 Body Designs

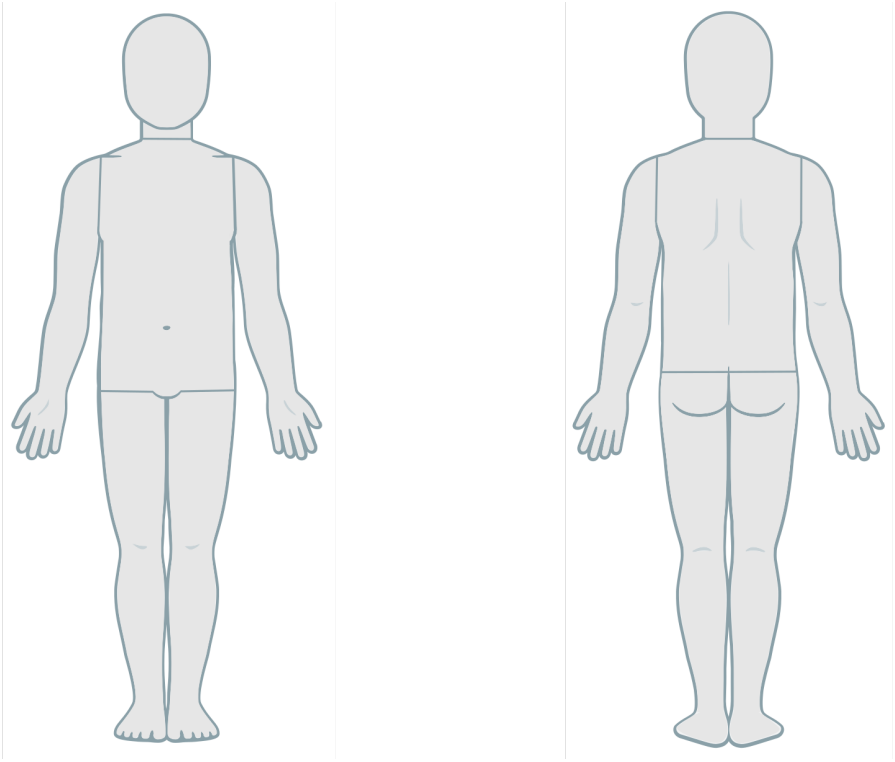


Figure 2.5: Iteration 4 Body Designs

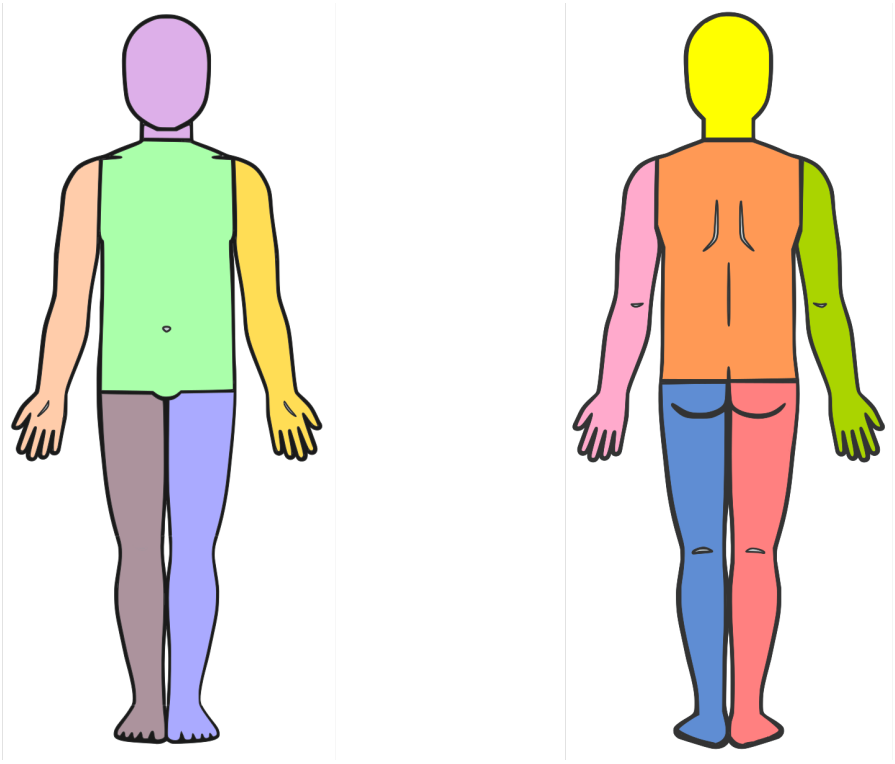


Figure 2.6: Iteration 5 Body Designs

2.1.2 Information Screens

As the title suggests, the information screens provide some background information on how to use the app in a safe manner. The screens in question are displayed in Figure 2.7. In order, the screens provide the following information concerning the following points:

1. Danger of Skin Cancer
2. Changes in Appearance
3. Preventing Skin Cancer
4. Consulting a Doctor

Ideally, the tutorial would also contain guidance on operating within the app, however, this would go against material design guidelines (*Onboarding*, n.d.), which suggest keeping tutorial screens short. A well-designed app user interface should require minimal tutorial help for the app to be used.

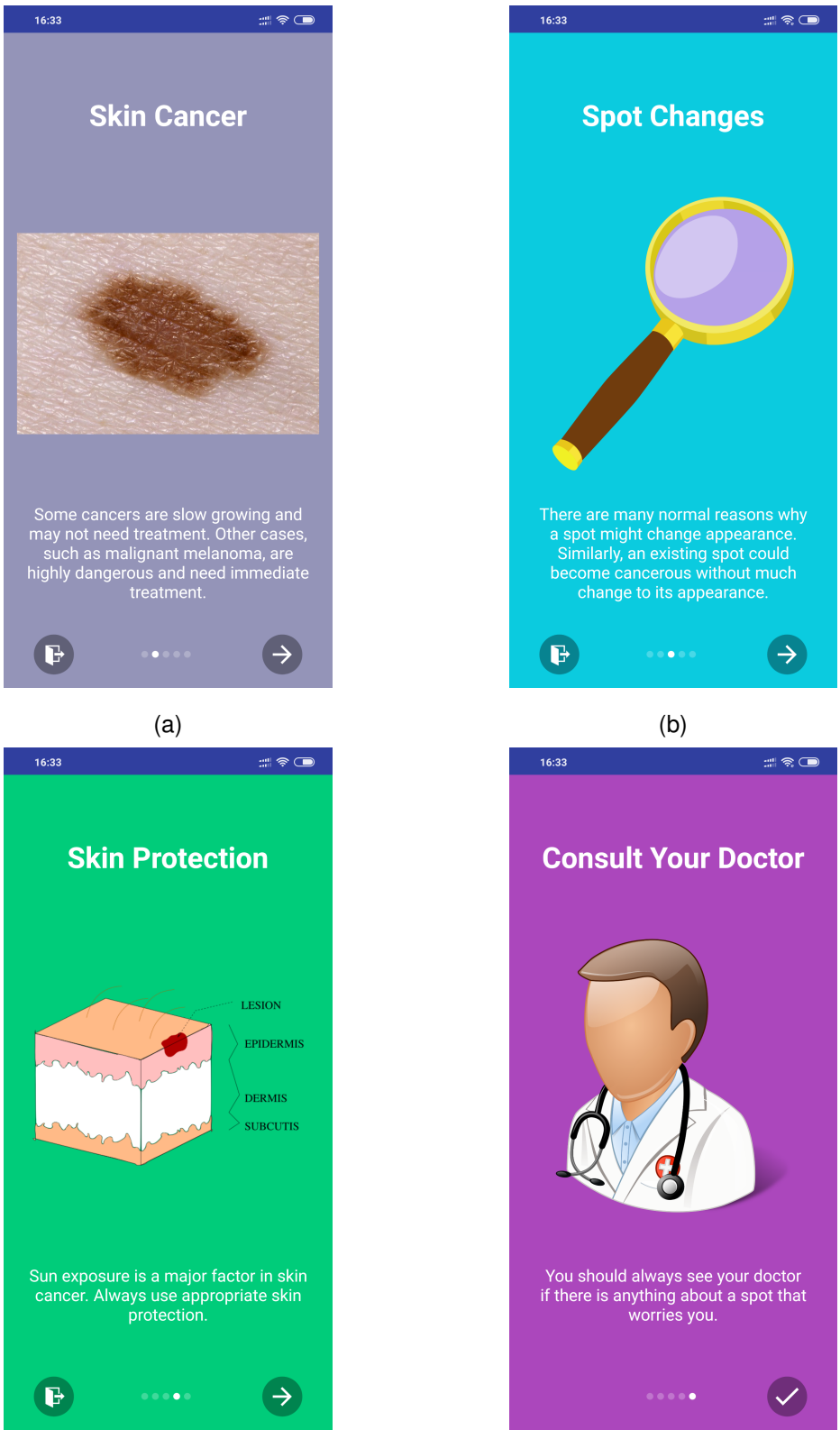
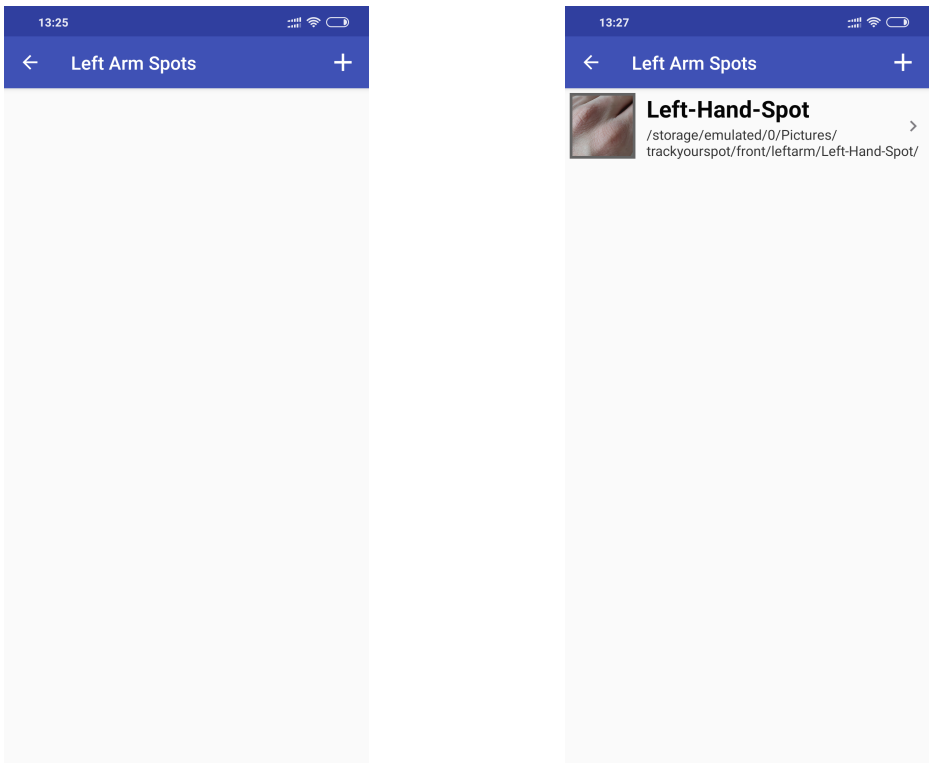


Figure 2.7: Providing background information to the user through Information Screens



(a) Empty Spot List

(b) Left arm spot list after adding a spot

Figure 2.8: Old Spot Screen for the Left Arm

2.1.3 Old Spot Screen

More accurately, this screen can be described as the screen displaying all previously added spots by the user in a selected body part. For example, if the user previously added the spots "Worrying Spot" and "Small normal spot" on their right arm, then these would be displayed in a list format on this screen. Each list item shows the spot name, a thumbnail of the spot, and the path of the spot in storage. From this screen, the user can:

- Add a new spot : By pressing the "+" button on the top right corner of the screen
- Open a previously added spot: By selecting the spot from the displayed list of spots
- Return to the body screen: By pressing the "←" arrow button on the top left corner of the screen

2.1.4 Camera and Cropping Screen

These successive screens are displayed on two occasions: When a user elects to add a new spot, and when the user adds a new image to an old spot (spot update). This allows users to take a photo of a spot (Figure 2.9), zoom in to a close up distance, and crop the image to a squared 1:1 aspect ratio. This ensures consistency across all images, so that all photos are of similar characteristics for easy comparison. The camera screen is provided by each user's default camera app, while the cropping screen is implemented through the uCrop Android Library (Yalantis, 2019).

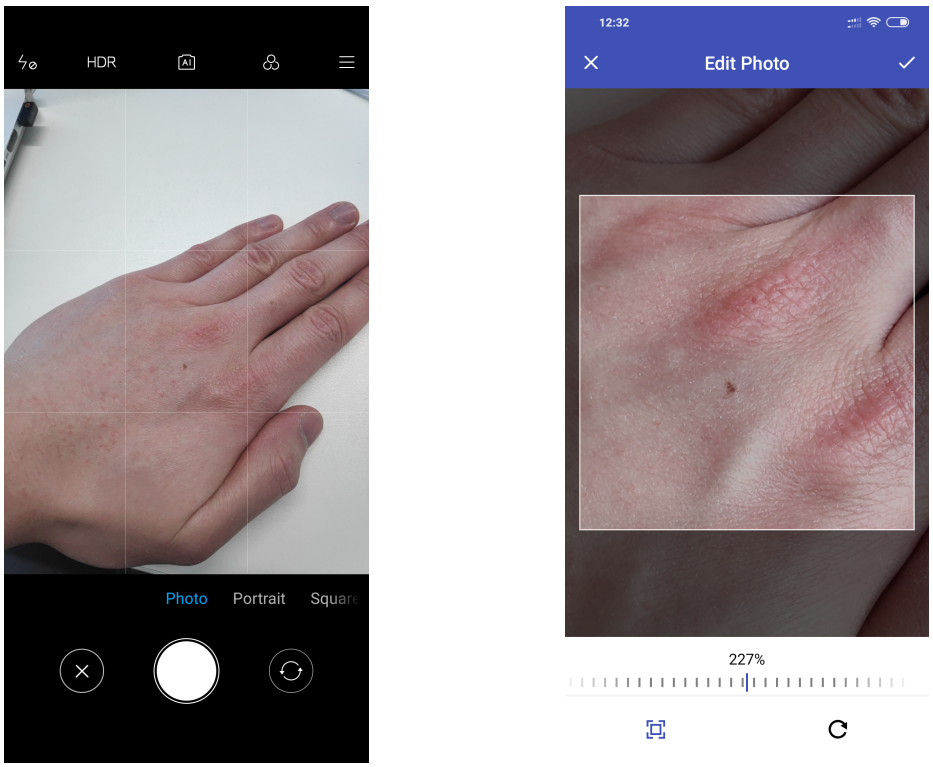
Using the Camera and uCrop API's means we are quite limited in customizing the appearance and design of these screens. The alternative would involve creating custom Activities and screen layouts. However, this would require a very large amount of effort for a relatively low reward. Benefits would include drawing a "mask" or "frame" around the camera, so that the user knows how to better position the spot in the photo or cropping. This would also require implementing back-end storage manipulating functions for both the camera and cropping activities. On the contrary, using the APIs already provides a very flexible and user friendly UI, hence the decision was made to use these.

Once the camera is open, the user can repeatedly take photos until they are happy with the image. This is indicated through the "✓" or "x" symbols. Subsequently, the cropping activity opens the image. The cropping screen allows zooming in and rotating the image, both operated by a slider at the bottom of the screen. Once again, the user can proceed to save the photo with the "✓" button, or cancel the action through the "x" button. This returns the app to the old spot or spot image list screens.

2.1.5 Spot Naming Screen

This screen is used to give a name to a new spot, this name is useful to better identify a spot when adding newer images in the future. Figures 2.10a and 2.10b show the very first initial design, later evolving into designs 2.10c and 2.10d (Showing and hiding the keyboard respectively).

This screen contains a text input field to name the spot, a large preview of the photo that has just been taken, and two aligned buttons labelled "Cancel" and "Confirm". Clicking either button would take the user back to the *OldSpotList* screen. However,



(a) Camera Screen after selecting to add a new spot

(b) Cropping Screen after zooming in to the chosen spot

Figure 2.9: Camera and Cropping Screens

the confirm button would successfully add the new spot to the list, while the cancel button deletes the image and returns to the previous spot list.

To justify the final design, the advantages and disadvantages of both designs can be compared. The initial design appears to be very basic, lacking an option to cancel the process of adding the spot. The design makes little use of screen size, leaving much of the display blank and keeping the image preview small in size. This could prove to be an issue with tiny spots or low quality images. The new design in Figures 2.10c and 2.10d addresses these issues accordingly, providing a much larger, half-screen view of the image, allowing even very small spots to be clearly visible (assuming the image has been cropped sensibly). The text entry field now has its own container, separating more clearly the different components of the screen. Following official Android design guidelines for image positioning (*Imagery*, n.d.), the image has been anchored to the top of the screen, with text right below it, this also makes the app more consistent with common image or social networking apps. Furthermore, the image and name container will always be visible even with the keyboard displayed. Lastly, a “Cancel”

button gives the user flexibility to restart the spot adding process if they are unhappy with the image. This can also be done with the "←" arrow button on the top left corner.

2.1.6 Spot Image List Screen

This screen displays a list of all taken photos of a spot (Figures 2.11a and 2.11b), this enables the user to compare them over time, spotting any differences in appearance. Under the toolbar, a list of images is displayed. Each list item shows a thumbnail for each particular image, the name of the image, and the date on which the image was taken. The toolbar contains multiple action buttons, a back button to return to the previous screen, a "Compare" button to open the comparison contextual toolbar, and a "+" button to add a new photo to the list. Clicking on an item in the list would open a full-screen view of the image for better inspection.

When the user presses the first "Compare" button, the contextual action bar opens. This enters the user into selection mode, where tapping on any of the images highlight the image, indicating it has been selected. If two images are selected, the "Compare" button lightens up, taking the user to the side-by-side comparison screen.

When it comes to displaying the list of images, alternative design decisions for selecting spots to compare will be discussed in section 2.1.7. However, we can justify the choice of what information is displayed on each item. The thumbnail of each image is critical for the user to know which images they are selecting to compare, the name of each image contains the "JPEG" together with the timestamp of the photo. Arguably, this format could be more user friendly by including the name of the spot. This would require extra image renaming methods in the app implementation, but would further help the user identify images.

2.1.7 Comparing Spots Screen

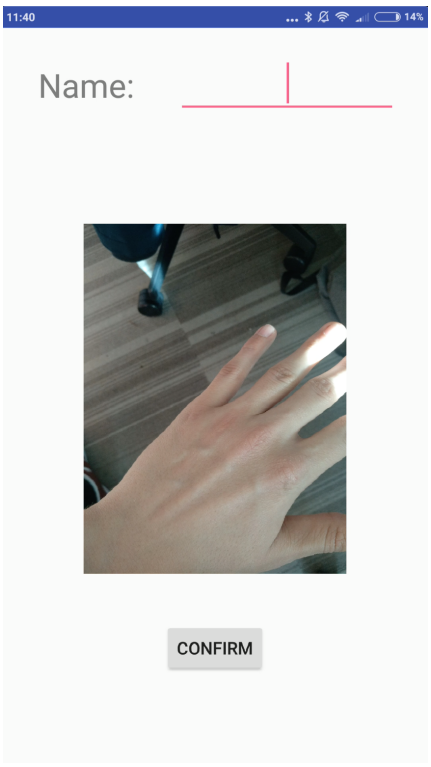
The spot comparison screen (Figure 2.12) allows the user to closely inspect the differences between two previously selected images. It is the main feature that differentiates the app from a typical photo gallery app, also allowing the user to email both images to a doctor or user specified address.

A lot of thought was put into what the best way to compare images was. Two main approaches were considered. Figures 2.12a and 2.12b show the initial design, where all images are displayed in a "CarouselView", imitating a slideshow where the user can swipe left and right to compare all images. The main advantages and disadvantages of this design are highlighted below:

- **Smooth and Convenient** - Comparing images through swiping feels intuitive and natural to users.
- **Memory Loss** - Continually loading all images of a spot can quickly exhaust the app's memory budget (*Handling bitmaps — Android Developers*, n.d.). With certain older devices, issues still arise even after using efficient bitmap decoding libraries such as *Glide*.

On the other hand, Figures 2.11b and 2.12c show the second (and final) design for comparing images. This approach requires images to be selected in the spot image list screen and displays them statically one above the other. Each image takes up 50% of the screen. Thoughts about this approach:

- **Image Count** - It is unlikely for users to save more than 5 images of a spot. The appearance of spots usually takes several months or years to change. This means that the user will not necessarily need to compare *all* the images of a spot, but the very first and last images would suffice.
- **Closer Zoom** - Loading only two images avoids any memory leaks, and removing the swipe interaction means images can take up the whole screen, enabling the user to inspect and compare the spots more easily.
- **Clunky Feel** - Selecting spots through a selection menu can feel more cumbersome to some users.



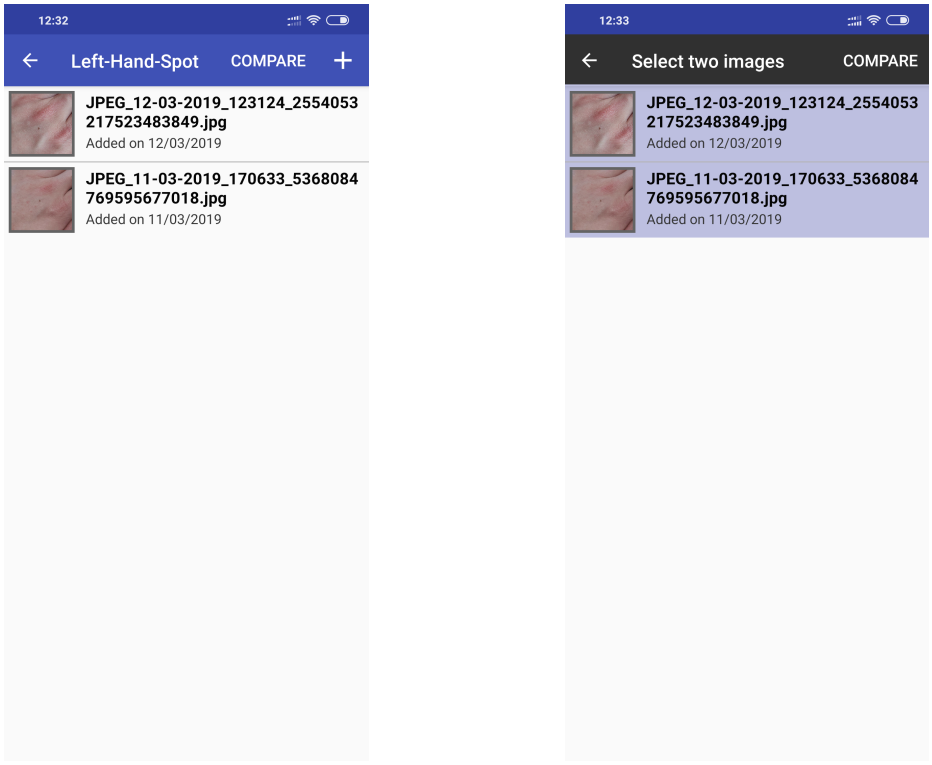
(a) Initial *AddSpot* Activity screen

(b) Initial *AddSpot* Activity Screen showing the keyboard

(c) Redesigned *AddSpot* Activity Screen hiding the keyboard

(d) Redesigned *AddSpot* Activity after inputting text

Figure 2.10: Progression of the *AddSpot* Activity screens



(a) List of images of a spot

(b) Selecting images to compare

Figure 2.11: *SpotImageList* Activity showing image selection for comparison

2.1.8 Email Screen

This screen is used to email the compared images to a doctor or another specified email address. There is no control over the design of this screen, as the "Email" button on the comparison screen simply opens a new email on the default email app on the user's device. The app automatically attaches both selected images and inserts a template text body (Figure 2.13).

2.2 Interactions

2.2.1 Navigation

The app's navigation overview can be observed in Figure 2.15, to summarise the interaction between the previously described screens: the app's home screen is the front perspective of the body screen. However, if the app has never been used before, the Information Screens will be displayed, giving the user a simple introduction to the app with some background information. From the home screen, the user can click on a body part to open the old spot screen for that body part, this would display a list of spots. The user can either add a new spot (Starting the Camera-Cropping Process) or view a spot by tapping it. Tapping takes the user to the spot image list for the selected spot, following a similar design to the old spot screen, but in this case showing all the images for a particular spot. The user can select two images to compare (as shown in the compare screen), or once again start the Camera-Cropping process to add a new image of the spot. In the compare screen, the user can also choose to email both images to a specified email address.

As a recall from the technical requirements of the project, the goal was to develop an app with the following features:

1. The user can add and name new spots. Subsequently, the user can add new images to a spot.
2. The user is able to compare two images of a spot on the same screen.
3. The ability to email both pictures to a doctor from within the app.
4. Educational information screens towards skin cancer signs and usage of the app.

From a design perspective, all the core features have been accounted for, relying the success of the project on how successful we are at implementation and evaluation of the app.

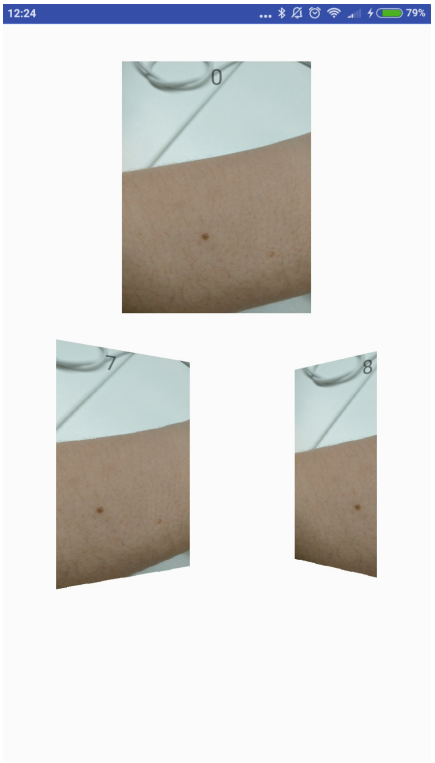
2.2.2 Navigation Redesign

The app's home screen and overall navigation was altered midway through the project. Initially, the app would welcome the user through a typical home screen menu. Figure 2.14a shows the layout of this home screen. The first 3 buttons would all lead to the body screen, after which either the camera or list of spots would be displayed (depending on the main menu option selected). As would seem apparent, this design is very redundant, as all three options would essentially lead the user to the same screens. This lead to re-factoring the whole design of the app, shifting towards a body interface home-screen (Figure 2.15). From this point, actions to add spots, add photos, or compare spots had to be implemented within the *OldSpotScreen* and *SpotImageScreen* Activities, since the user wouldn't be selecting these from the main menu. Benefits and drawbacks of this approach include:

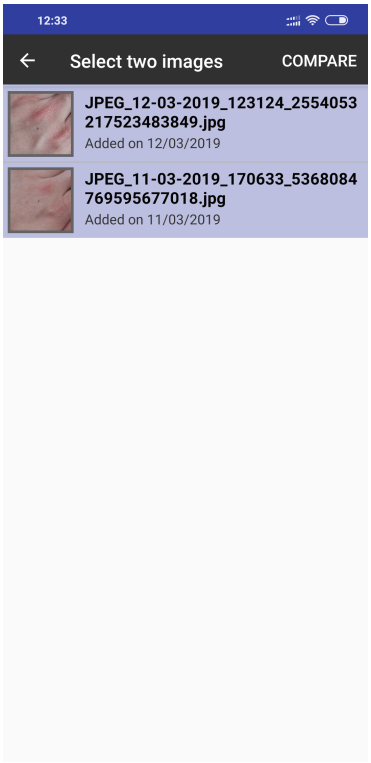
- **Decrease redundancy** - The user doesn't have to go through the same 2 screens for every different action, saving time and delivering a more efficient interface
- **Loss of clarity** - Having a main menu with options has its merits, as the user knows exactly what action they are currently doing, preventing them from getting lost within the app. This is particularly true with older users.
- **Flexibility** - The user can easily switch between actions without returning to the main menu, for example, if a user wanted to add a new spot A, then add an updated photo of a different spot B, and finally compare two images of spot B, they could easily do this by entering the different body part spot lists, without requiring the user to return to the main menu screen three times.

Overall, the conclusion was that the main menu design in Figure 2.14 would only be suitable if users only completed 1-2 within every app use. With this unrealistic expectation, it was decided to shift to the body home screen design.

(a) Display of initial compare screen design



(b) Comparing images can be done through swiping left and right



(c) Final compare screen image selection

(d) Final design for comparing spot images

Figure 2.12: Progression of the Spot Comparison Screen

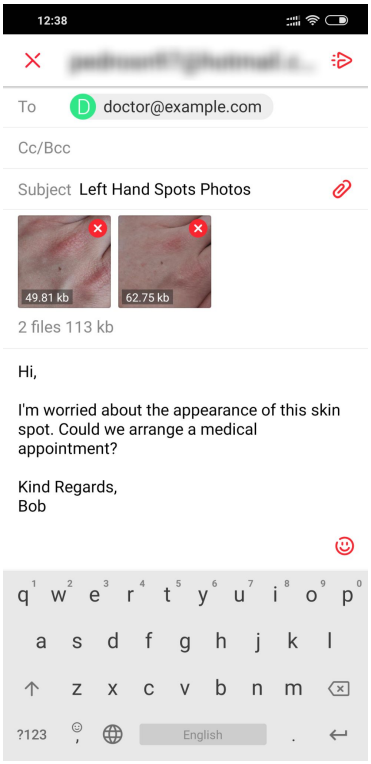
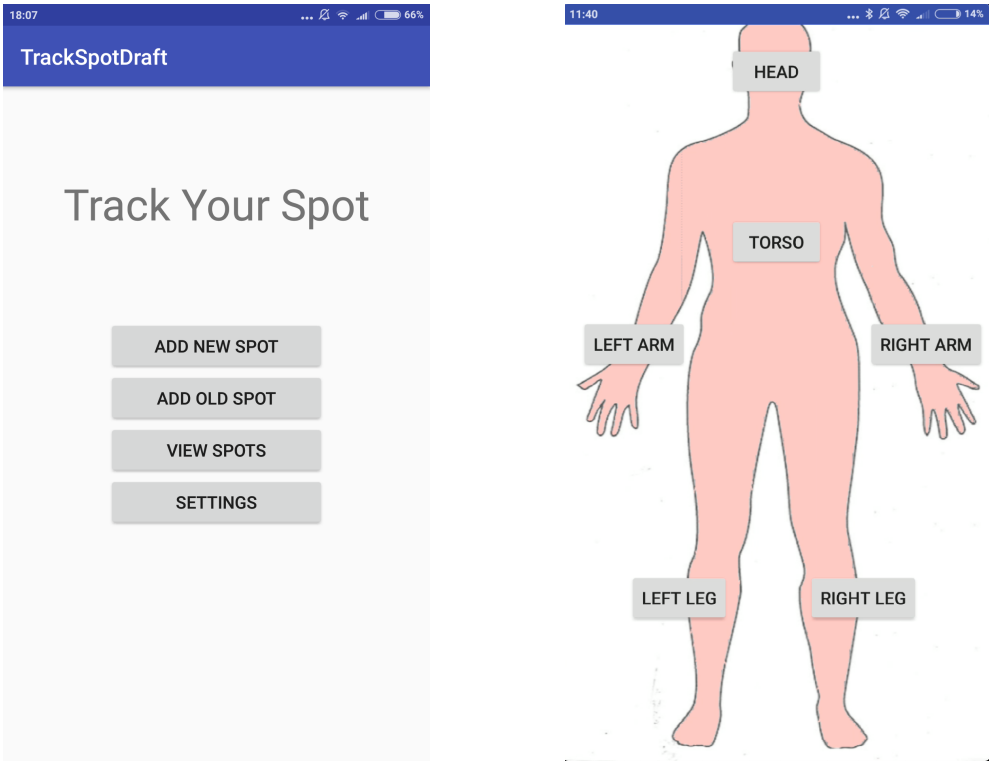


Figure 2.13: Email Screen through the *myMail* Android App



(a) First app draft home screen

(b) First app draft body screen

Figure 2.14: Navigation menu of first app draft

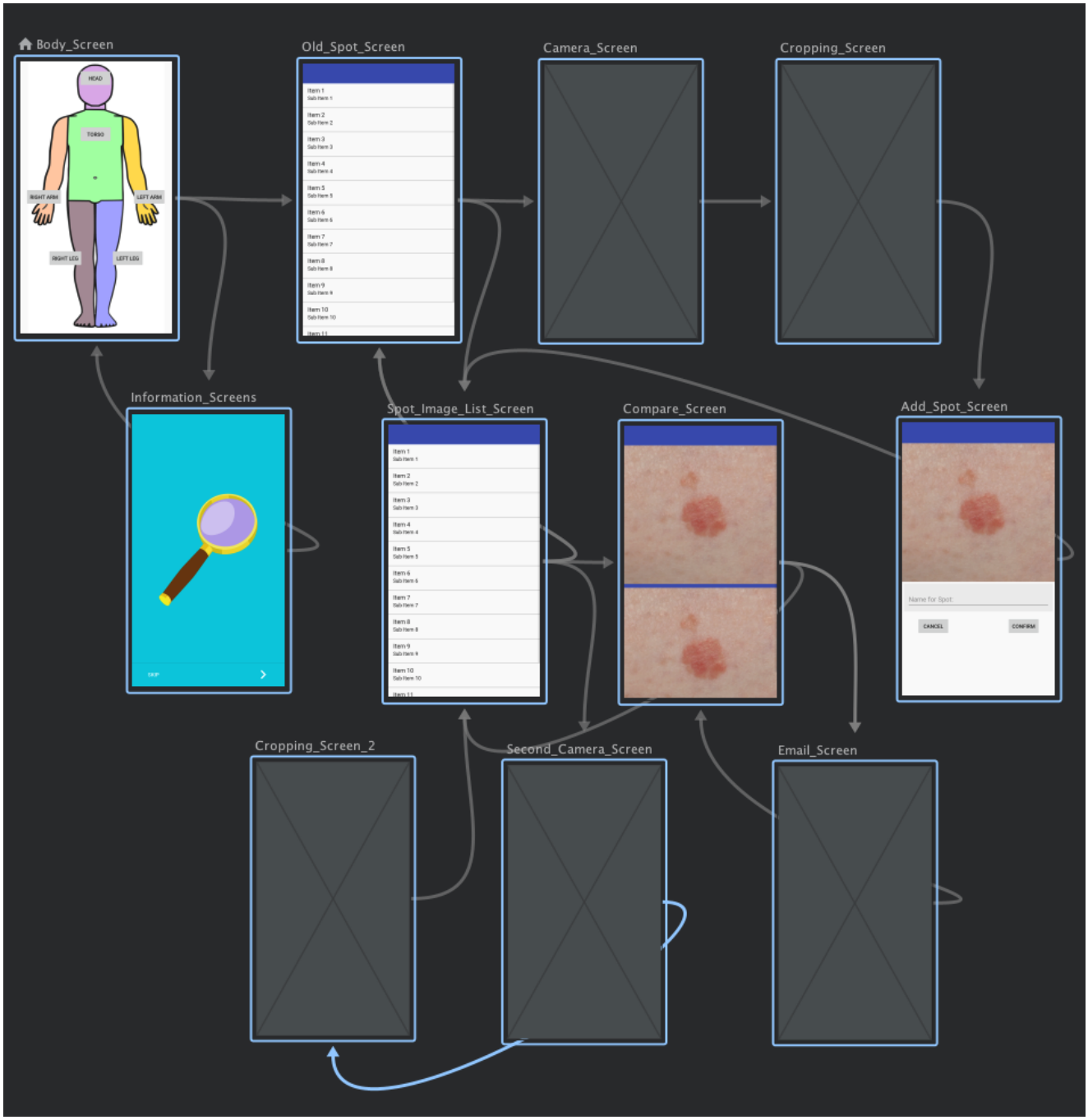


Figure 2.15: Navigation Graph

Chapter 3

Architecture and Implementation

3.1 Main Components

This section encompasses the architecture behind the app. Each screen will include information about methods and algorithms used, accompanied by Activity UML diagrams to better visualise what the code does for the user and how it interacts with the app's UI. Note that Android's default "back" button allows the user to always return to a previous screen in the app. To avoid cluttering, obvious "back" operations have been omitted from UML diagrams. In addition, note that code listings only contain subsets of the code, a lot of the less "interesting" code has been omitted from the report.

The language of choice is *Kotlin*, the leading programming language in Android app development (Sagar, 2019). Kotlin is a fully Android-focused language running on the *Java Virtual Machine*. It is extensively supported by *Android Studio*, the official Android IDE. Both Kotlin and Android Studio have been developed by *JetBrains*. For these reasons, the option of developing the app in Java was quickly dismissed. Any relevant libraries used within activities can be found in the bibliography. It might be useful to re-read Section 1.2 for a better understanding of the technical terms used throughout this section.

3.1.1 App Opening and Information Screens

In Android development, we use *Shared Preferences* as a way of saving the user's app data to be used in the future. In our implementation, we don't necessarily need a

heavy use of shared preferences, as images are directly saved and scanned from storage (see Section 3.3). However, we can utilise them to save small user settings or specific variables. In the app's homescreen, we use this tool to check if the user has already completed the tutorial in the past. Observe code listing 3.1, the code first checks if the user has just returned from another screen, selecting the right tab "front" or "back" from where the user started their action, this code is irrelevant if the app has just been opened. On lines 11-15, we make use of a boolean "TutorialDone" to track if the user has already completed the tutorial information screens, this is accomplished through the `getPreferences`, `getBoolean` and `putBoolean` methods.

As displayed in Figure 3.1, the user can return to the body screen by completing or skipping the tutorial. Contrarily, pressing the "i" button allows the user to retake the tutorial.

Listing 3.1: Displaying Information Screens

```
1  //Check if the user has pressed the back button
2  selectedBodySide = if (intent.hasExtra("selectedBodySide")) {
3      intent.getStringExtra("selectedBodySide")
4  } else {
5      "front"
6  }
7  createTabs(selectedBodySide) //Displays front and back tabs
8  //Android way of retrieving previous app data
9  val sharedPref = this.getPreferences(Context.MODE_PRIVATE) ?: return
10 val isTutorialDone = sharedPref.getBoolean("TutorialDone", false)
11 if (!isTutorialDone) {
12     //If app first time use, show tutorial
13     sharedPref.edit().putBoolean("TutorialDone", true).apply()
14     val intent = Intent(this, IntroActivity::class.java)
15     startActivity(intent)
16 }
```

3.1.2 CameraOpeningActivity

To explain the implementation of the *OldSpotList* and *SpotImageList* activities, we first have to break down the *CameraOpeningActivity* class. This class was created so that the two mentioned activities could share the `createImageFile` and `dispatchTakePictureIntent` methods, avoiding any redundant code. `createImageFile()` is used for creating a temporary empty image file in storage for the image to be taken, and `dispatchTakePictureIntent()` starts the actual camera app to save a new picture into this image file. The code has been adapted

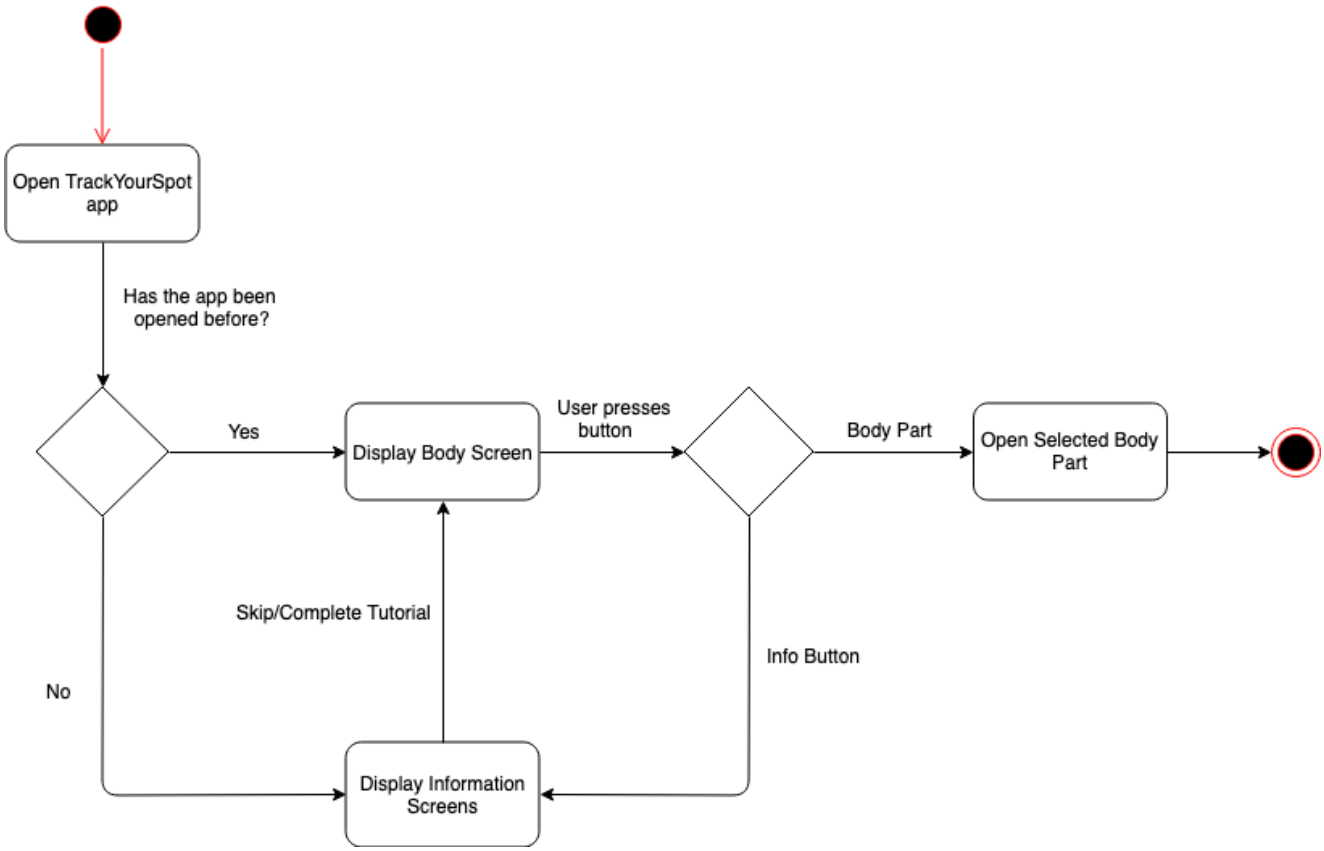


Figure 3.1: UML Activity Diagram for Info Screens

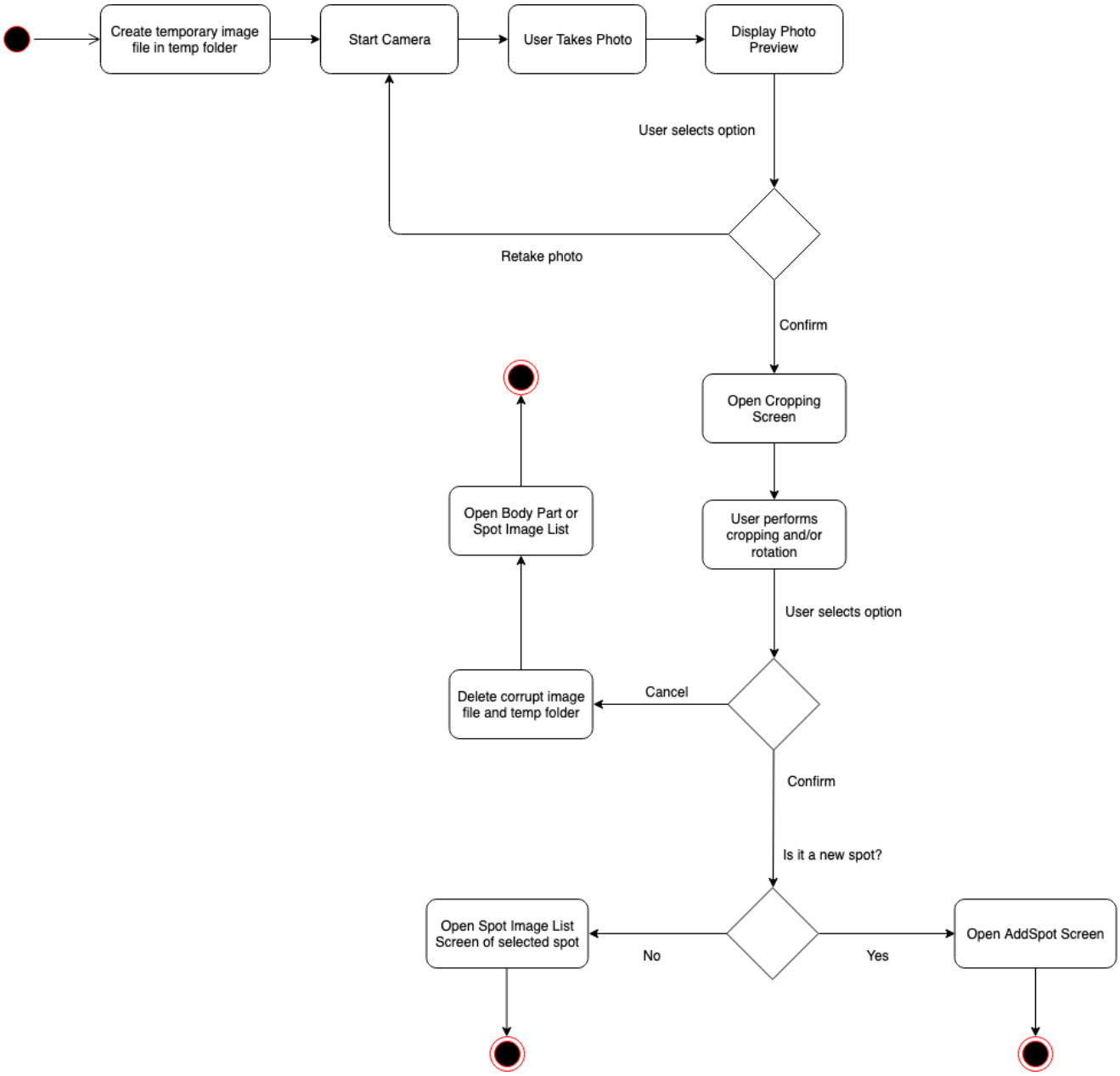


Figure 3.2: UML Activity Diagram for the Camera Activity

from the Android docs (*Take photos — Android Developers*, n.d.), as these methods are widely used across any Photo taking app, comments have been added for clarification.

Take the *OldSpotList* activity as an example. The user first selects a body part. They are then presented with the list of spots from that selected body part. Once the user presses the "+" button, the `dispatchTakePictureIntent()` method would be called (Code Listing 3.2). The spotDirectory argument contains the path of where the photo will be saved, this could be something similar to:

`/storage/emulated/0/Pictures/TrackYourSpot/front/torso/worryingspot/`

In line 8, the `createImageFile()` method first tries to create a temporary image file at the given path. It uses the current date and time to create a unique timestamp string (Code Listing 3.3). This ensures no two images will have the same name. Next, the method checks if the spotDirectory path exists in storage, creating them through the `mkdirs()` method if needed. Once we have the directory, it creates the temporary file by concatenating the "JPEG_" prefix, the timestamp, and the ".jpg" file extension. Back to the `dispatchTakePictureIntent` method, we need to find the *URI* of the photoFile. The URI variable is passed on to the Intent in charge of starting up the Camera. For compatibility reasons, we have to check the device's Android version, as SDKs older than Lollipop require a different way of attaching intent data to the camera intent.

To see how this code interacts with the app's UI and navigation screens, take a look at Figure 3.2. The diagram also shows the paths the app takes if the user cancels the camera or cropping processes midway through. Since the camera process is based around temporary image files, these have to be dealt with accordingly.

Listing 3.2: Starting the Camera

```
fun dispatchTakePictureIntent(spotDirectory: String) {
        Intent(MediaStore.ACTION_IMAGE_CAPTURE).also {
            takePictureIntent ->
            // Ensure that there's a camera activity to handle the intent
            takePictureIntent.resolveActivity(packageManager)?.also {
                // Create the File where the photo should go
                val photoFile: File? = try {
                    createImageFile(spotDirectory)
                } catch (ex: IOException) {
                    ... //Display error message
                }
                // Continue only if the File was successfully created
                photoFile?.also {
                    //Get the image file's URI
                    val photoURI: Uri = FileProvider.getUriForFile(
                            this, "my.package.name.provider", it)
                    //For compatibility issues, we need to check the device's
                    //Android version and add write permissions accordingly.
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                        takePictureIntent.addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
                    } else {
                        val clip = ClipData.newUri(contentResolver, "clipData", photoURI)
                        takePictureIntent.clipData = clip
                        takePictureIntent.addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
                    }
                    //Open Camera App
                    takePictureIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
                    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI)
                    startActivityForResult(takePictureIntent, OldSpotScreen.REQUEST_TAKE_PHOTO)
                }
            }
        }
    }
```

Listing 3.3: Creating an Image File

```
private fun createImageFile(spotDirectory: String): File {
        //Used for creating a unique name for the image file
        val timeStamp: String = SimpleDateFormat("dd-MM-yyyy_HHmmss").format(Date())
        //If needed, create the folder for the spot.
        val newDir = File(spotDirectory)
        if(!newDir.exists()) newDir.mkdirs()
        //Create a temporary image file
        return File.createTempFile(
                "JPEG_${timeStamp}_",
                ".jpg",
                newDir
        ).apply {
            //Saves the image path in String format for later use.
            currentFullPhotoPath = absolutePath
        }
```

3.1.3 Old Spot List Screen

The old spot list screen extends the abstract class *CameraOpeningActivity*. The code in this activity accomplishes the following tasks (See UML Diagram in Figure 3.3):

1. Load all saved spots from storage
2. Display spots with thumbnails and other info in a list format
3. Start camera automatically or after the user's button click, checking permissions accordingly

To display the list of spots of a selected body part, we run the algorithm in code listing 3.4. This algorithm scans through directories and reads every spot name and path repeatedly. It first initialises empty arrays for spotNames, spotDetails, spotPaths and thumbnails (Lines 1-4). Using the `walk()` method, it loops through every file inside the `../TrackYourSpot/bodypart/` directory. Setting the `maxDepth` parameter to 1 ensures only the top level folders inside these are read, in other words, only the spot names, but nothing inside these directories. Index 0 contains the root folder, therefore we choose to ignore it as it does not provide a spotName. Inside each of these spotName folders, we perform a 1-depth walk again, this time selecting only the file at index 1, which would contain the first image taken of a spot. We then use this image's path to display a thumbnail for each spot.

Listing 3.4: Loading Old Spots

```
var spotNames = emptyArray<String>() //Spot Names
var spotDirectories = emptyArray<String>() //Spot paths without jpg file
var spotThumbnails = emptyArray<Bitmap>() //Spot thumbnail paths
var numberOfSpots = 0 //Counter for later use
File(spotListDirectory).walk().maxDepth(1).forEachIndexed { index, file ->
    if (index != 0) { //Ignore the root folder
        numberOfSpots++
        val spotName = file.toString().removePrefix(spotListDirectory)
        spotNames += spotName //Add spot to list of spot names
        spotDirectories += "$spotListDirectory$spotName/" //Add directory
        val filesInFolder = File("$spotListDirectory/$spotName/").walk().maxDepth(1).toList()
        //Grab first image of each spot folder
        spotThumbnails += if (filesInFolder.size>1) {
            BitmapFactory.decodeFile(filesInFolder[1].absolutePath)
        } else { //Show question mark icon if image is missing.
            BitmapFactory.decodeResource(this.resources, R.drawable.questionmark)
        }
    }
}
```

3.1.4 Spot Image List

This activity also extends the CameraOpeningActivity class, similarly to the *OldSpotScreen*. This screen also makes use of the camera when a user elects to add a new photo of an existing spot. Figure 3.4 depicts the process of the opening the screen, with the different paths available to the user. When the user opens the spot image list of a particular spot, all the photos of a spot are read from storage. This algorithm is displayed in Code Listing 3.5. In a similar way to the algorithm in Code Listing 3.4, spots are first loaded from storage with the `walk()` method. This creates a list of `imageFiles`, each of which we decode into an image path and thumbnail. We also initialise an array `selectedSpotBooleanList` to keep track of which images are being selected by the user (to compare).

Following the UML Diagram, the user can press the "+" button to add a new photo. If the goal is to compare two images, the user can either:

- Press on an image for more than 2 seconds
- Click the "Compare" button on the toolbar

Both of these options would launch the *Contextual Action Bar*, this is a toolbar that substitutes the default toolbar when a specific action takes place. It allows the user to

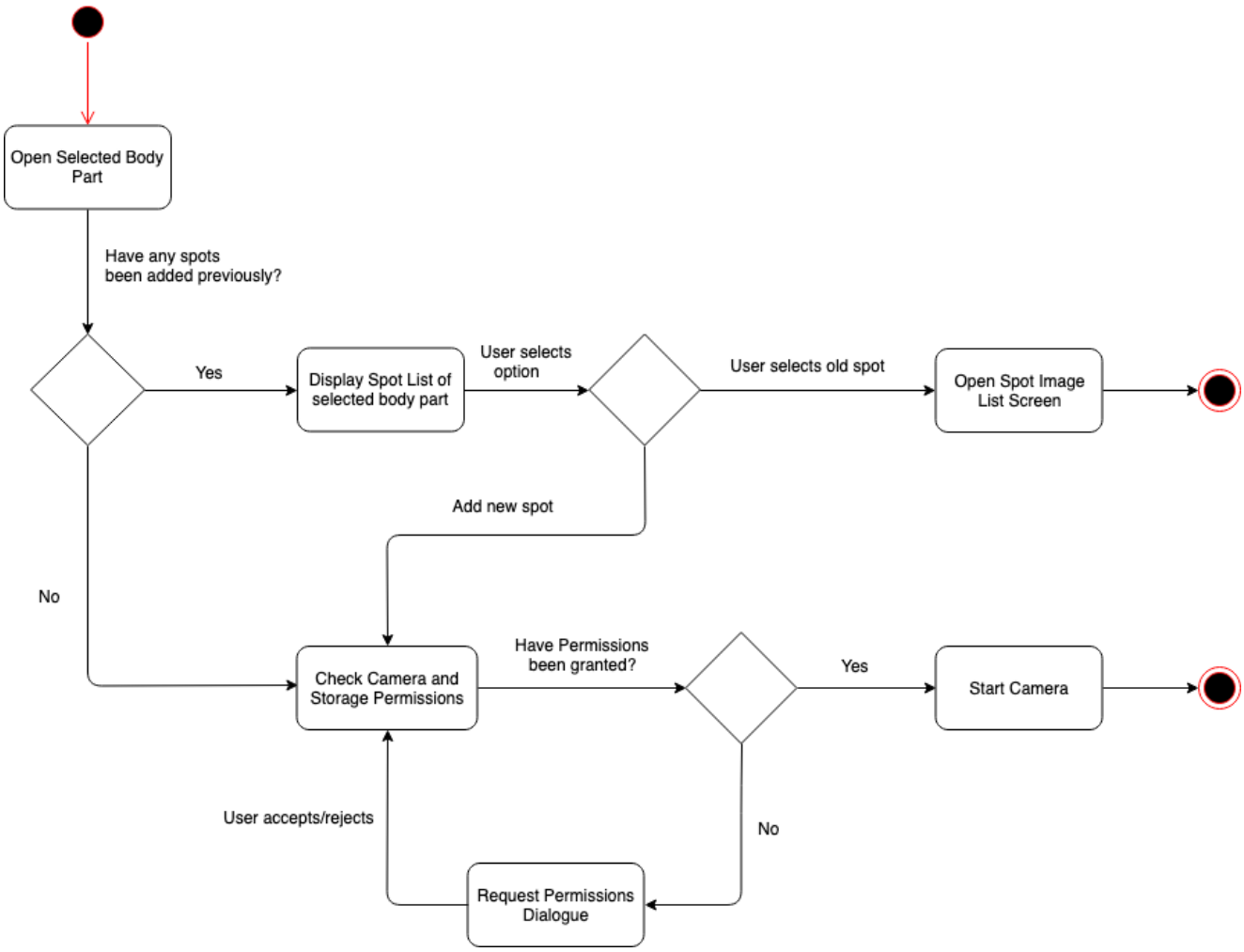


Figure 3.3: UML Activity Diagram for the Old Spot's screen

select images from the list. If exactly two images are selected, a "Compare" button lights up, indicating the user can compare the images.

Listing 3.5: Loading spot images to compare

```
//Initialise arrays for each photo's path and thumbnails
var fullImagePaths = emptyArray<String>()
var imageThumbnails = emptyArray<Bitmap>()
//Loop through images in the selected spot's directory
val imageFiles =  File(spotDirectory).walk().maxDepth(1).toList()
for (i in 2..imageFiles.size) {
    fullImagePaths += imageFiles[i-1].toString()
    imageThumbnails += BitmapFactory.decodeFile(imageFiles[i-1].absolutePath)
}
numberOfImages = fullImagePaths.size
//Initialise dynamic array to keep track of which images are selected
val selectedSpotBooleanList = arrayOfNulls<Boolean>(fullImagePaths.size)
//By default, no images are selected
Arrays.fill(selectedSpotBooleanList, java.lang.Boolean.FALSE)
```

3.1.5 AddSpot Screen

When the user chooses to add a new spot to a body part, they first take a photo, crop it to the desired dimensions, and is then redirected to this screen, the *AddSpot* Activity. As shown in Figure 3.5, they are asked to input a name for this spot, displaying appropriate error messages if the name violates any of the following properties:

- Spot name is not left blank
- Spot name is under 20 characters
- Spot name contains only letters and numbers

This ensures stability when spot folders are created in storage, as any unrecognised characters (e.g. @, !, $) could easily cause errors. The rules were devised following advice on folder naming conventions from (Santaguida, 2011). If the name satisfies these conditions, the spot folder is created in storage, substituting spaces for dashes ("-") to further comply with good practices.

The algorithm in charge of this task is shown in Code Listing 3.6. Running through the algorithm; a `photoDirectory` string is created, populating it by replacing the "temp" text with the body side, body part, and given spot name. This is the first step to move the image file from the temporary folder into a new folder named after the new spot. Next, a directory `newDir` is created at this path (making sure it doesn't exist yet through

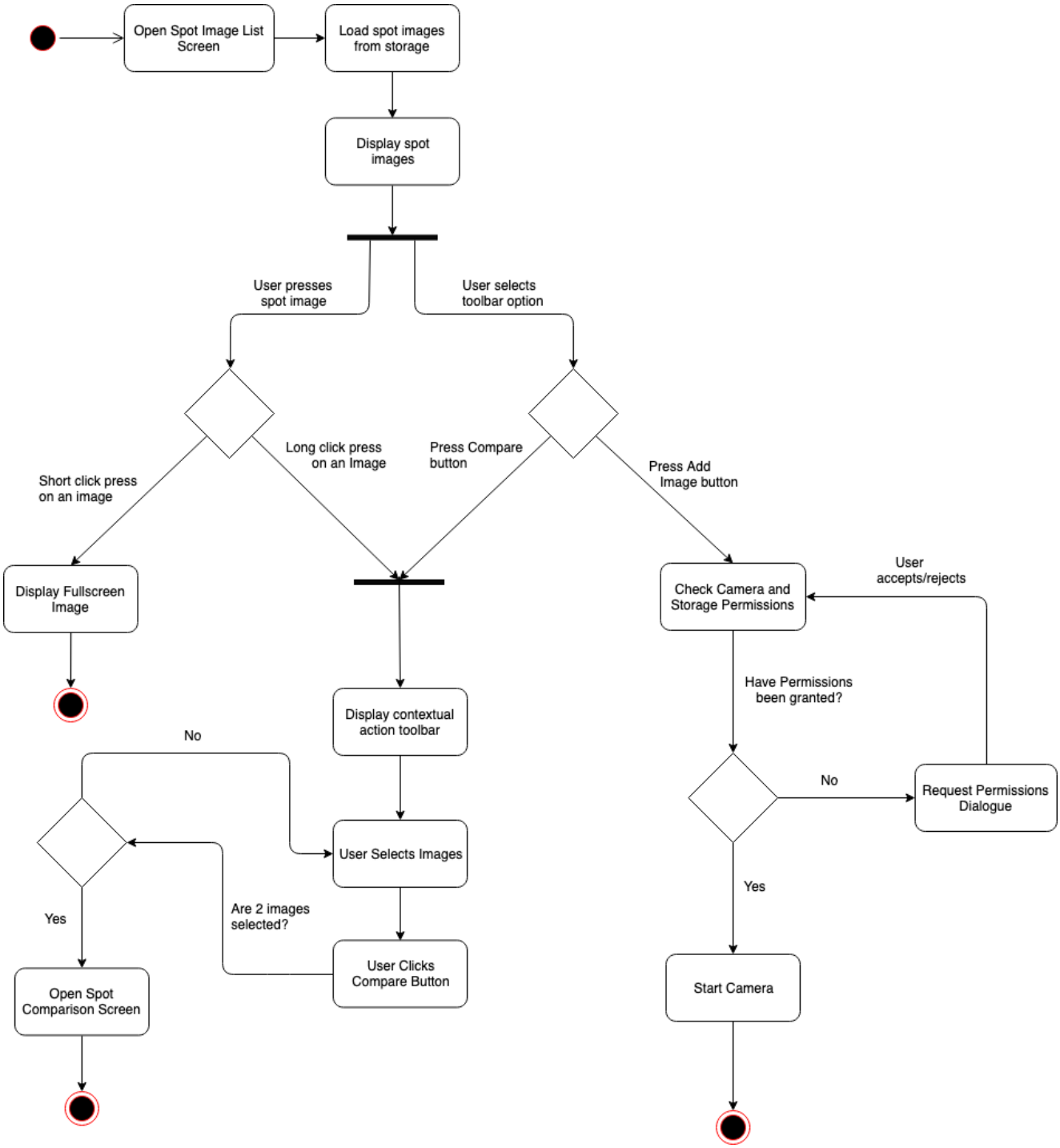


Figure 3.4: UML Activity Diagram for the Spot Image List

the `exists()` command). Once the folder is created, we proceed to move the image by modifying its path with the `renameTo` method. We then delete the (now empty) `tempfolder`, as the file has been successfully moved.

Listing 3.6: Adding a Spot

```
private fun moveImageFile(spotImageName: String,
newSpotName: String, selectedBodySide: String, selectedBodyPart: String) {
    val photoDirectory = fullPhotoPath.removeSuffix(spotImageName)
    //Move image from temp folder to a new *newSpotName* folder
    val newDirPath =
    photoDirectory.replace("temp", "$selectedBodySide/$selectedBodyPart/$newSpotName")
    val newDir = File(newDirPath)
    //Check if the directory exists already, create it otherwise
    if(!newDir.exists()) newDir.mkdirs()
    //Make app available in the device's Gallery
    Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE).also { mediaScanIntent ->
        val f = File(fullPhotoPath)
        mediaScanIntent.data = Uri.fromFile(f)
        sendBroadcast(mediaScanIntent)
        //Rename the file to the new path
        f.renameTo(File(newDirPath + spotImageName))
        //Delete the temporary folder
        val tempFolder = File(photoDirectory)
        deleteRecursive(tempFolder)
    }
}
```

3.1.6 Compare Screen

The workflow of the compare screen is very straightforward. Following the UML diagram in Figure 3.6, once the two images are displayed, the user's only options are to go back to the image list or email the two photos. If the user wanted to email the photos, they would press the email button, at which point the default email app is initiated, automatically creating a new message with the two attachments. Once the user sends the email, the compare screen opens again, displaying a *Toast* message (short pop-up message that fades away after 2 seconds).

3.2 Data Flow

Most mobile applications require constantly sending information from one screen to another. This is done through the use of *Intents*. In plain terms, an Intent is an intention

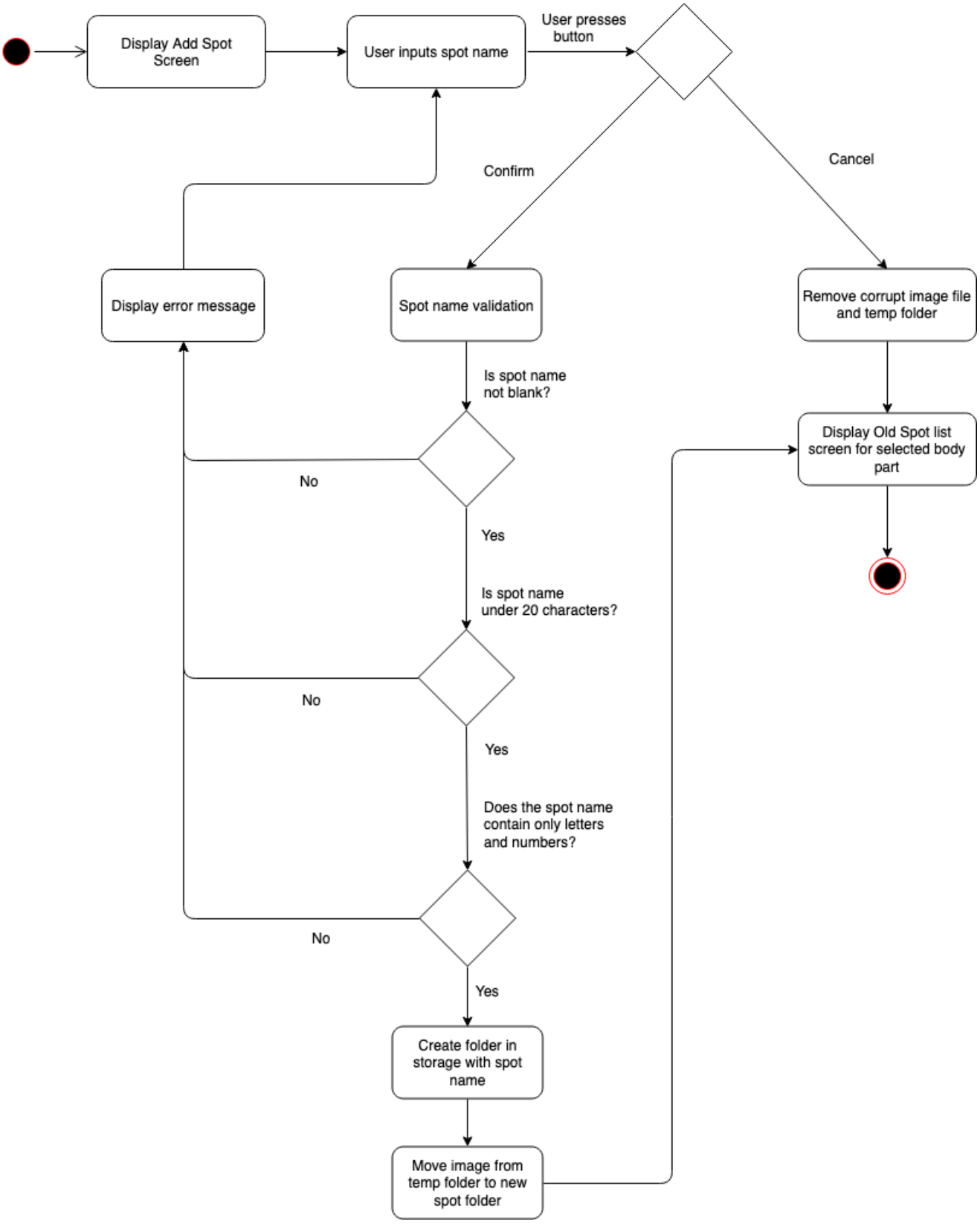


Figure 3.5: UML Activity Diagram for the *AddSpot* Activity

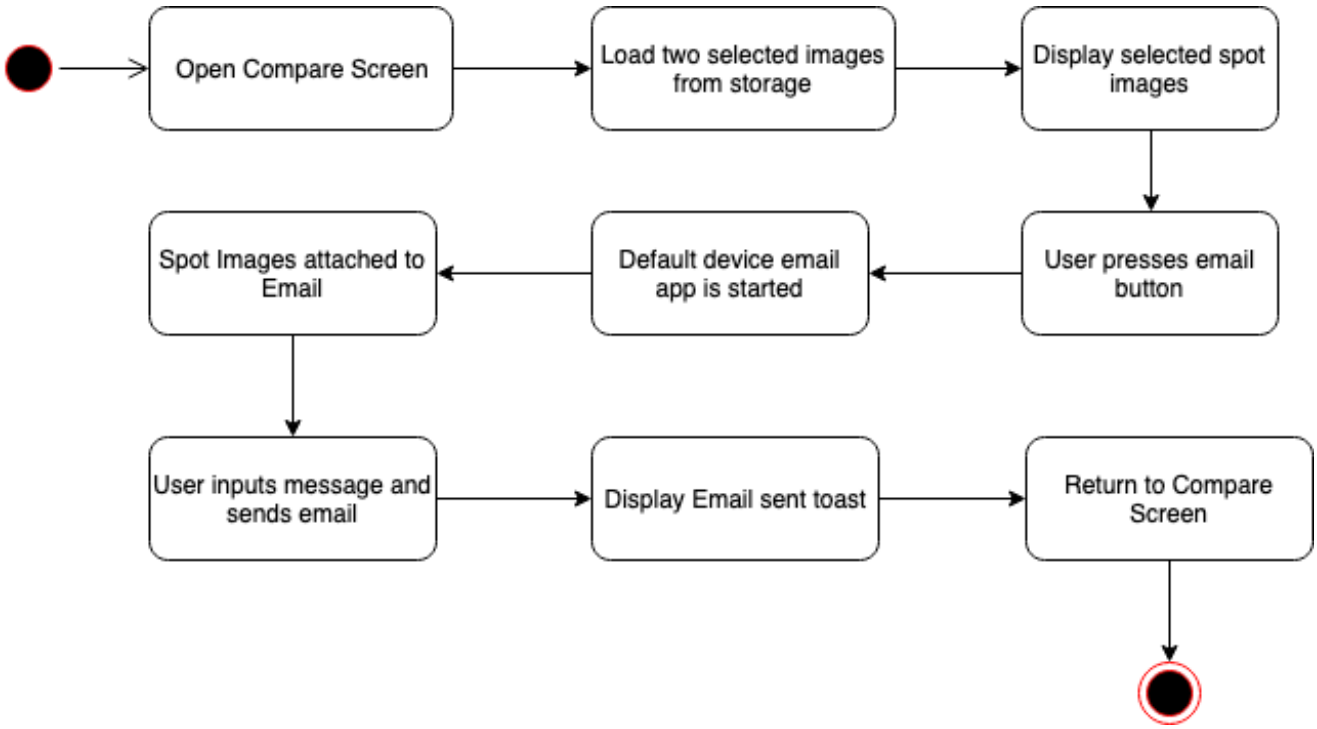


Figure 3.6: UML Activity Diagram for the *CompareScreen Activity*

to perform an action, or in other words, "a messaging object you can use to request an action from another app component" (*Intents and Intent Filters — Android Developers*, n.d.). Intents can have data extras, these are key-value pairs such as strings or booleans. In the *TrackYourSpot* context, the main data being attached to intents is:

- **selectedBodySide** - "front" or "back"
- **selectedBodyPart** - "head", "back", "torso", "leftarm", "rightarm", "leftleg", "rightleg"
- **spotName** - User defined spot name such as "Spot1"
- **spotDirectory** - Directory of a spot in storage such as "/storage/emulated/0/Pictures/TrackYourSpot/front/torso/Spot1/"
- **spotListDirectory** - Directory of a body part in storage such as "/storage/emulated/0/Pictures/TrackYourSpot/back/head/"
- **spotImageName** - File name of a particular image such as "JPEG_20190113_133157.jpg"
- **fullPhotoPath** - Full path of an image such as "/storage/emulated/0/Pictures/TrackYourSpot/front/torso/Spot1/JPEG_20190113_133157.jpg"

An example intent can be found in Code Listing 3.7. In that particular example, the intent takes the user from the *OldSpotList* to the *AddSpot* screen. Note how variables and intent data labels are equal. This has been done to ensure consistency across screens and make sure it is clear what every variable represents in different screens.

Listing 3.7: Intent Dataflow for the *AddSpot class*

```
1 val intent = Intent(this, AddSpot::class.java)
2 intent.putExtra("selectedBodySide", selectedBodySide)
3 intent.putExtra("selectedBodyPart", selectedBodyPart)
4 intent.putExtra("spotListDirectory", spotListDirectory)
5 intent.putExtra("spotImageName", spotImageName)
6 intent.putExtra("fullPhotoPath", currentFullPhotoPath)
7 startActivity(intent)
```

3.3 File Structure

The file structure for the app is relatively simple. Initially, images are stored in the default picture directory defined by each device. To identify this path, a File object needs to be created by calling the following method:

`getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)`

This file's path can then be identified by calling the inbuilt `getAbsolutePath()` method from the File.java class.

Once we're located inside the absolute directory, pictures in .jpg format are saved in a `../bodySide/bodyPart/spotName` folder hierarchy, where bodySide can be one of "front" or "back", bodyPart refers to the selected body part of a spot, it can take any of the following labels:

- **Front** - head, torso, rightarm, leftarm, rightleg, leftleg
- **Back** - head, back, rightarm, leftarm, rightleg, leftleg

The spotName is the user defined name given to the spot at the time of adding the first image. Spot folders are created on demand, so the `../front/leftarm/worryingspot` folder will only be generated when the user takes a picture of a spot and gives it a name. Therefore, a typical folder structure could look like Figure 3.7.

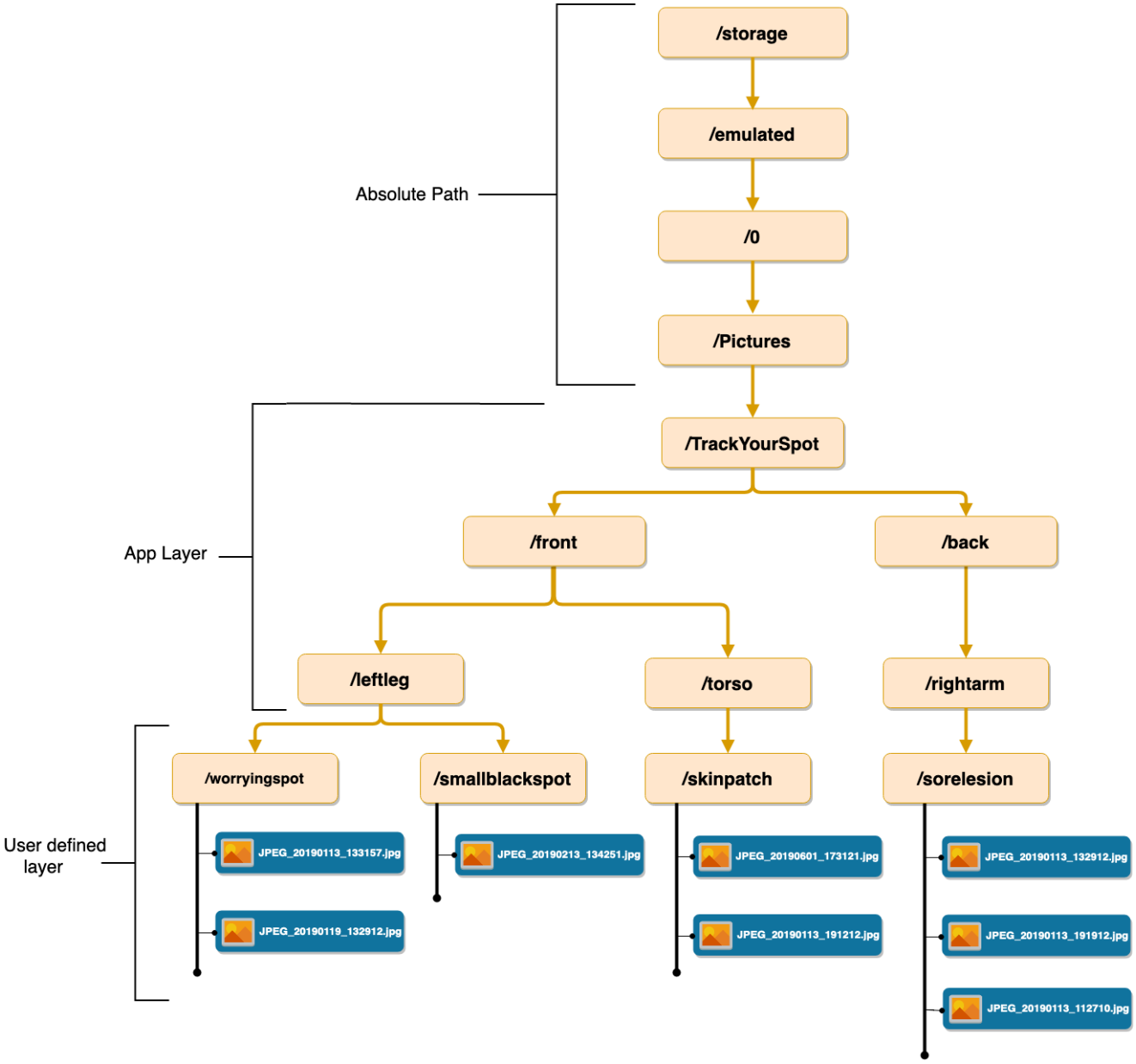


Figure 3.7: A typical folder hierarchy showing folder layers within the user's device

Another component of this file hierarchy are `/temp` folders. These temporary folders are only generated when pictures are not saved correctly due to the user closing the app without naming the spot, causing the spot to not be saved to the correct location.

Chapter 4

Deployment

4.1 Google Play Store

This section will dive into the development and administrative tasks required to officially deploy the app to the public. This task only commenced once Beta Testing was declared complete. Recall the way we managed app releases through Alpha and Beta testing, deployment to production was done in a very similar way. The Android Developer Console facilitated this task through a "Push to Production" button on the latest beta release of the app. Before the app was pushed to production, a series of forms and administrative tasks were required.

Firstly, an app title and descriptions were needed (Figure 4.1). These would be displayed on the app's download page, with the goal of persuading the user to click on the app and download it. After testing stages, it was decided to add the "Free Skin Cancer Tracker" label to the name. This is purely an advertisement instrument to ensure the app appears in as many searches as possible.

Secondly, the app's nature required the creation of a privacy policy (Figure 4.2). This is usually optional, but apps that require storage or camera permissions have to include one. An app privacy policy generator (Nishant Srivastava, n.d.) was used for this task. The file was hosted on a *Github* website repository to ensure it was permanently available.

Thirdly, the app's pricing details had to be defined (Figure 4.3). As determined previously, the app would follow a free business model, with no in app purchases or ads

either. All 144 available countries would be included in the deployment list.

In fourth place, a content rating questionnaire had to be filled in (Figure 4.4). App content ratings intend to help parents identify potentially objectionable content that exists within an app. In the case of our app, the PEGI 3 age tag was delivered, this means Google determined the app to be suitable for all age groups.

Lastly, the app's graphics had to be uploaded (Figure 4.5). An app's graphics include:

- App screenshots
- Highest resolution icon
- Feature graphic (banner)

The icon and banner were designed with the Inkscape vector graphics editor, combining different layers of copyright-free cliparts.

Once all the described forms were complete, the app was submitted to the Google Play Store. Four hours later, the app was successfully approved and officially deployed (Figure 4.6). If you recall the app listing screenshot from Beta testing (Figure 5.5), the app title had an "(Unreleased)" tag. This tag is no longer shown.

4.2 Website

As part of the project, a website had to be developed to provide guidance on how to carry out all the core tasks of the app. To get started, a *Bootstrap* business website template was downloaded from *StartBootstrap* (Bootstrap, n.d.). Bootstrap is an HTML, CSS and Javascript framework to develop responsive websites (Twbs, 2019). This template was adapted to include links for all the tasks to be explained. Links for the Google Play Store and Android App Store versions of the app were also included. Figures 4.7 - 4.10 show a selection of screenshots of the finished website. The full website can be found at `www.trackyourspot.com`.

This part of the project was also done by the iOS *TrackYourSpot* developer, my contributions are described below:

- Page formatting and design of the home screen

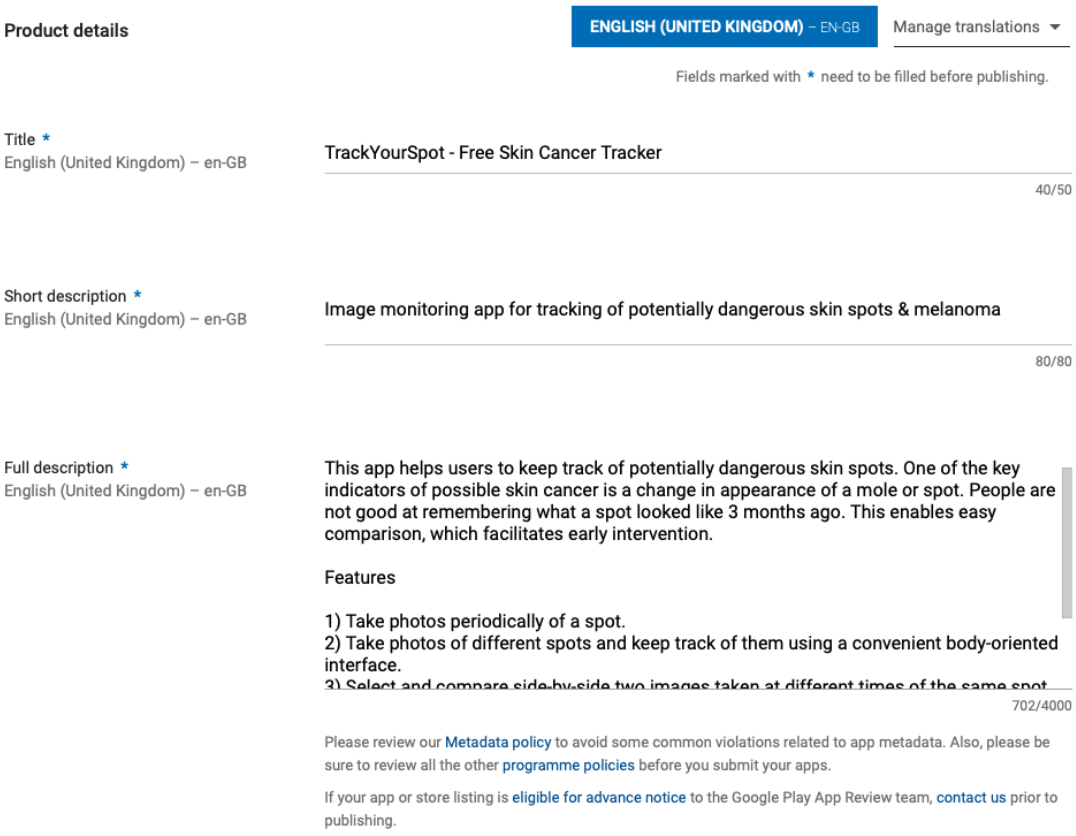


Figure 4.1: Google Play Store App Details

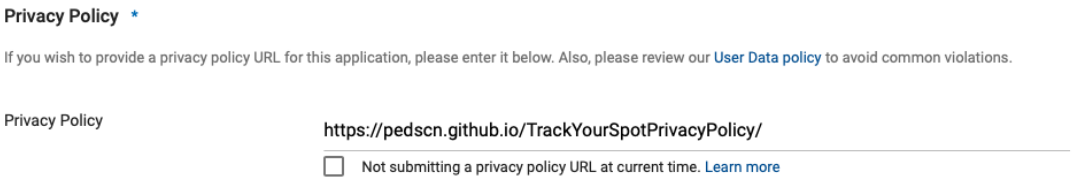


Figure 4.2: Google Play Store Privacy Policy Form

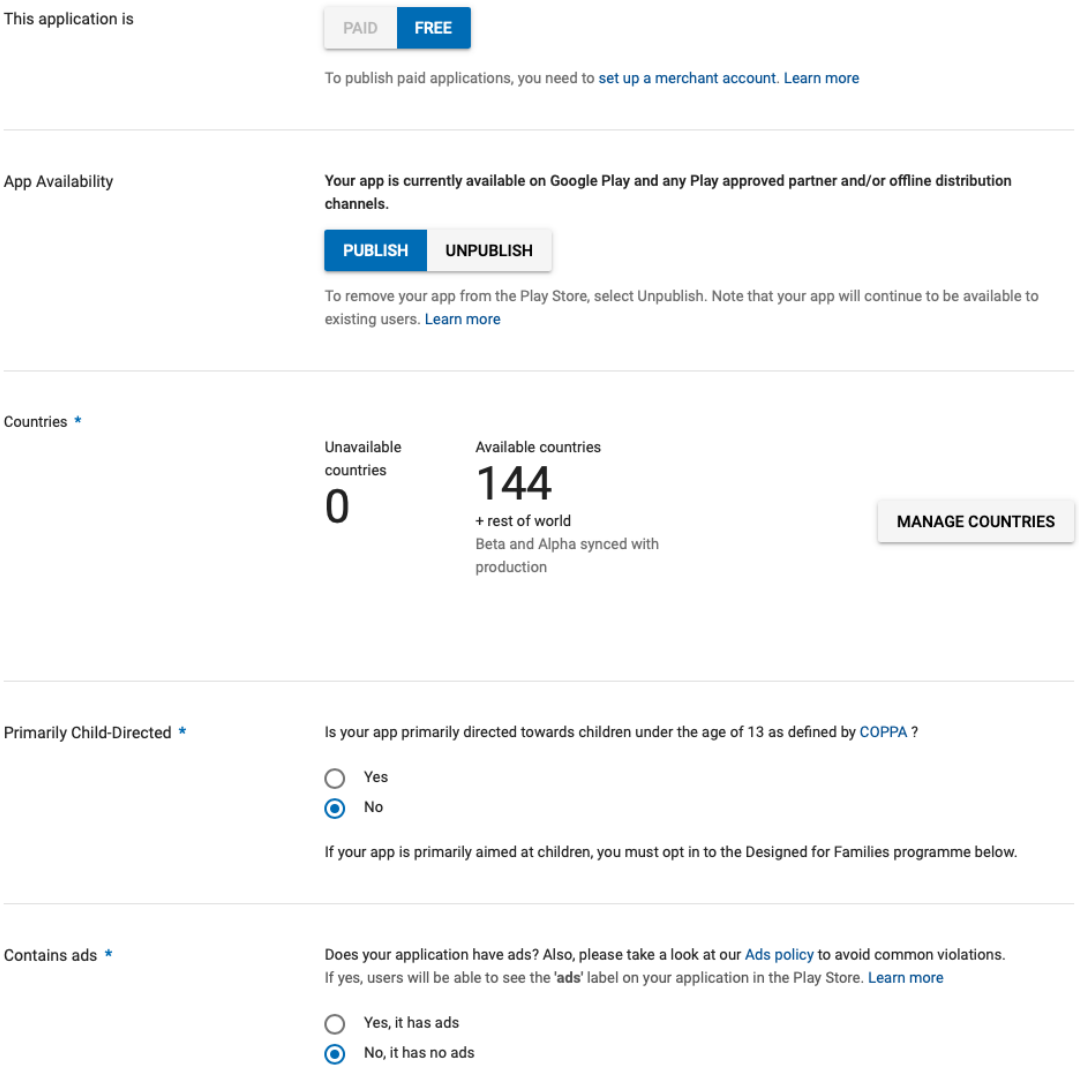


Figure 4.3: *TrackYourSpot* Deployment Settings

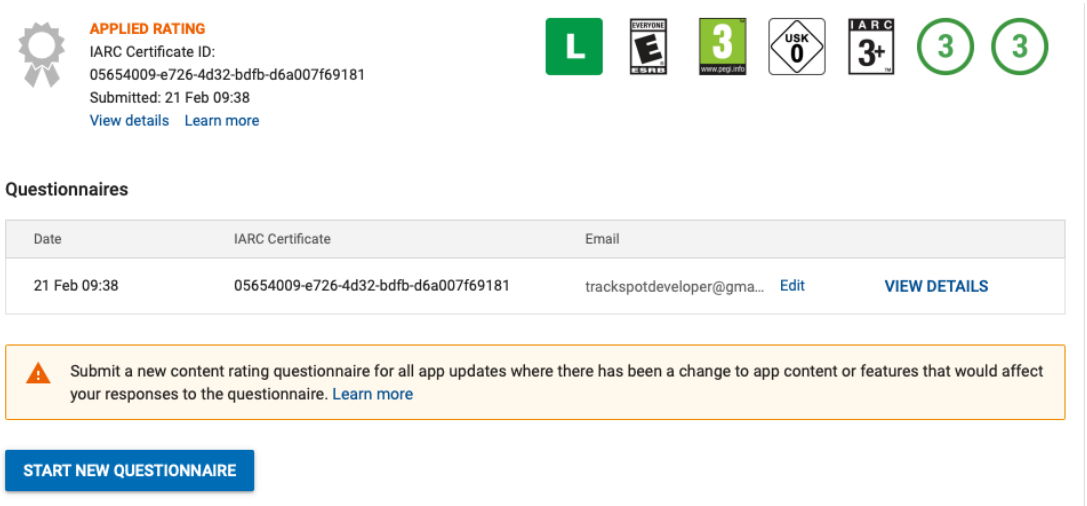


Figure 4.4: Google Play Store Content Rating

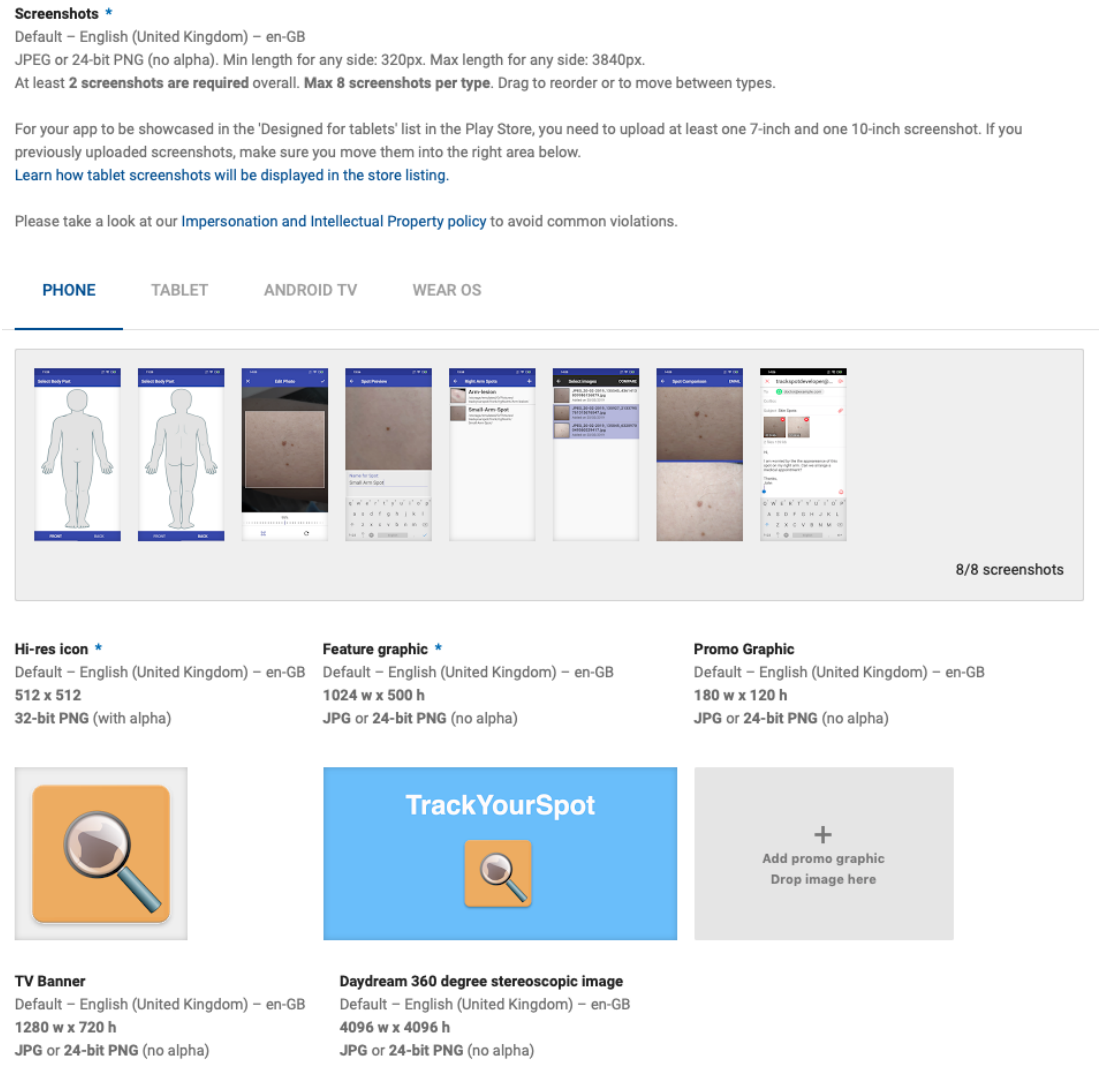


Figure 4.5: Google Play Store Image Uploads

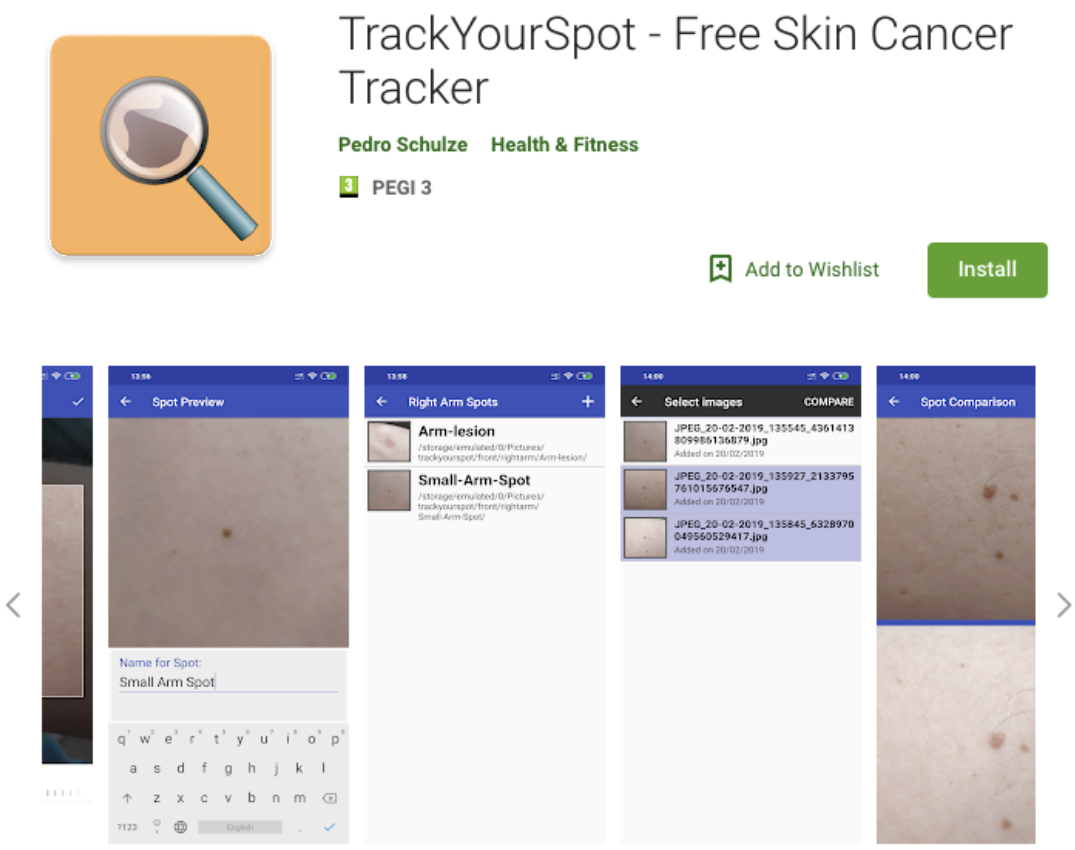


Figure 4.6: *TrackYourSpot* app officially available to the public

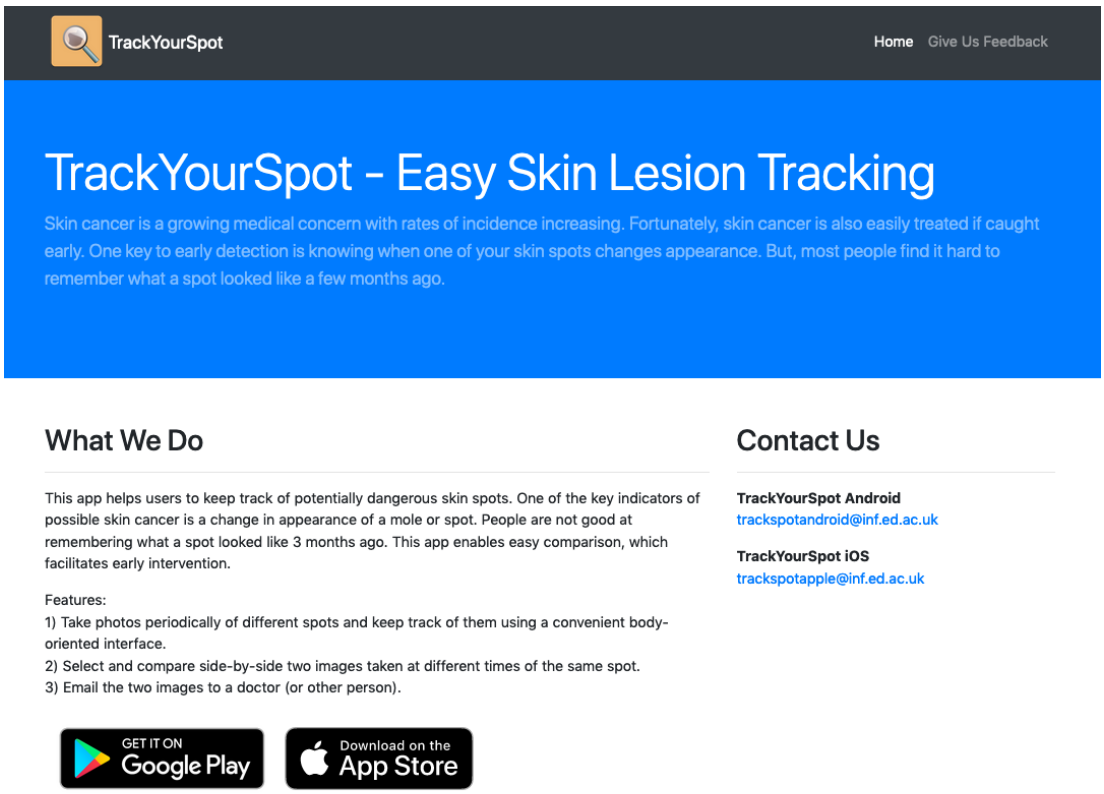


Figure 4.7: *TrackYourSpot* website homepage

- Finding suitable copyright free cliparts to be used as icons for the home screen (Figure 4.8)
- Modifying HTML code to describe steps to be completed for multiple tasks
- Attaching Android screenshots for each step

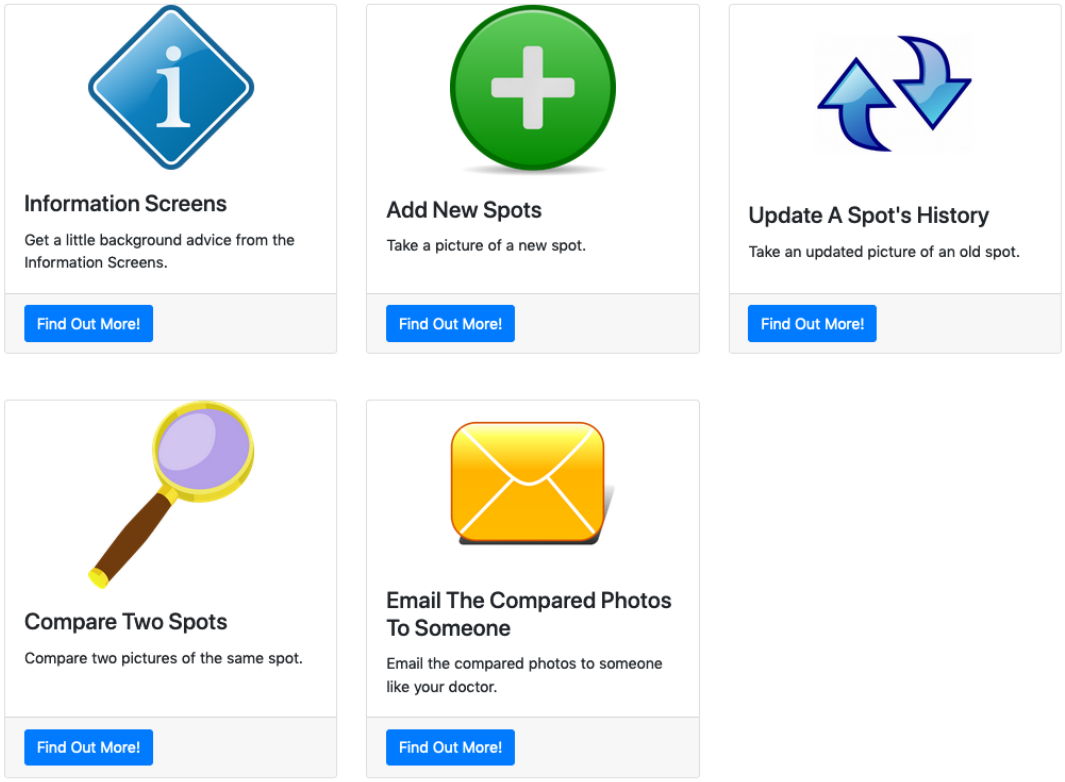


Figure 4.8: *TrackYourSpot* website homepage (continued)

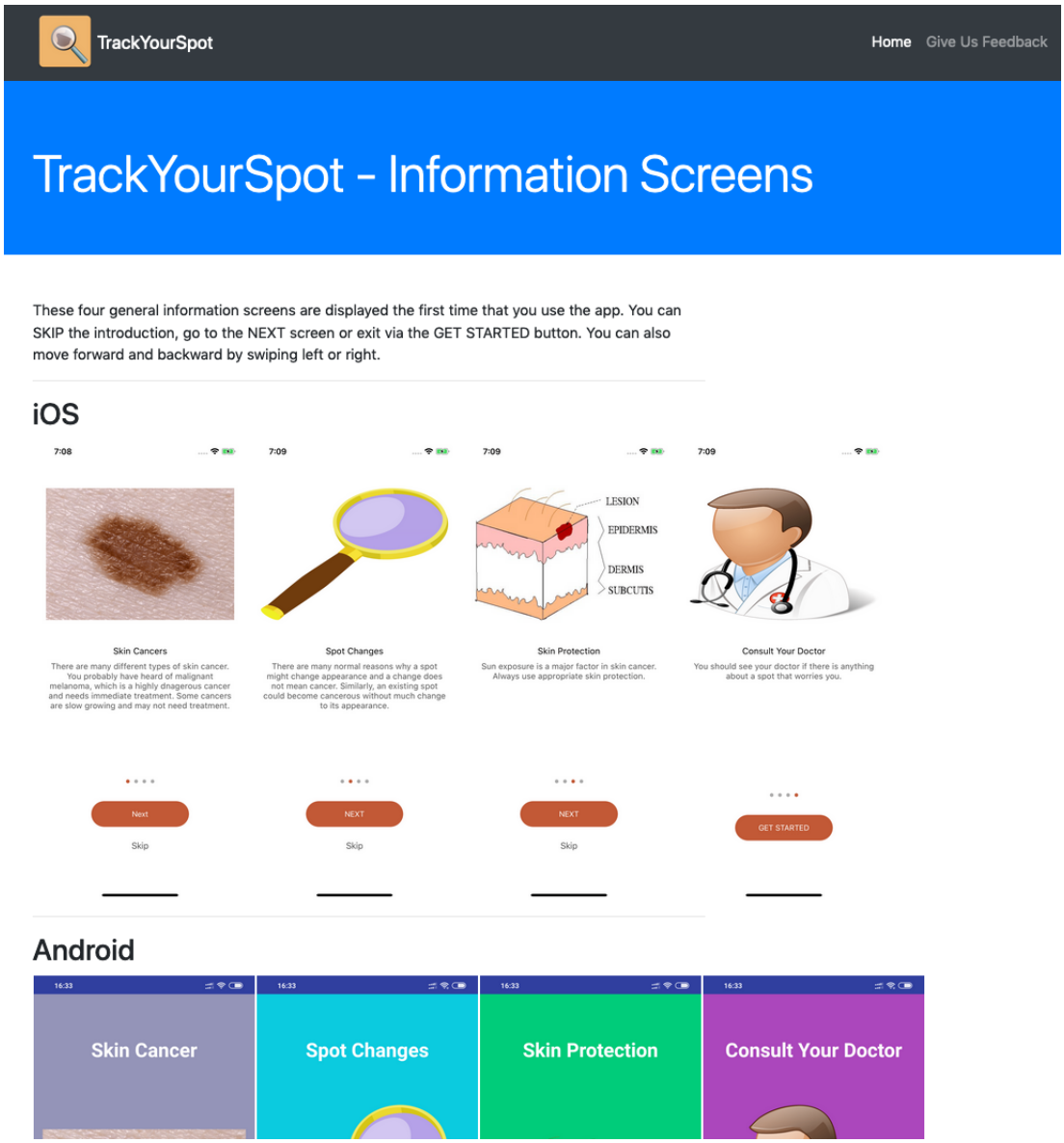


Figure 4.9: *TrackYourSpot* website information screens guidance

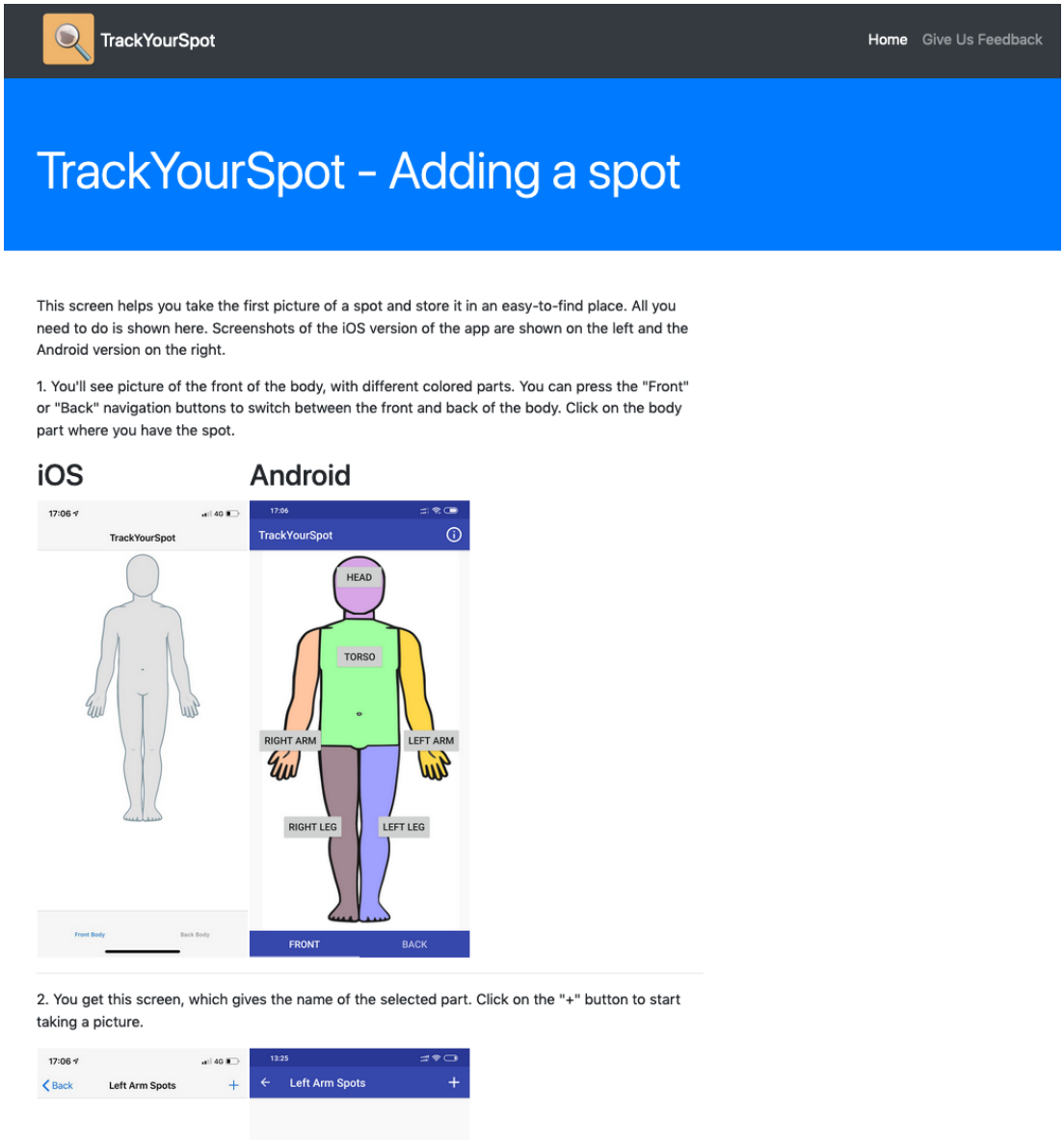


Figure 4.10: *TrackYourSpot* website adding a spot guidance

Chapter 5

Testing and Evaluation

Testing of the app was carried out in four main stages. Including both technical functionality tests and user design evaluations. They can be summarised into the following:

- **Internal Testing** - Also known as "pre-alpha", this stage was carried out for both major and minor bug fixes. It prepared the app to be further tested by external testers.
- **User Acceptance Testing** - This stage was used for minor bug fixes. The app was exposed to users in a controlled, private circle (alpha testers) and later to the general public in the beta stage.
- **Preference Testing** - This stage took place by surveying users on their preferred interface between two different design concepts for multiple screens in the app.
- **Usability Evaluation** - Usability testing involved assessing the user experience through both qualitative and quantitative measures. These included interviews, think-aloud sessions and system usability scale reports.

5.1 Internal Testing

Once the final stages of development were completed, the internal testing phase commenced. Internal testing's main focus was around testing multiple SDK versions and particularly the minimum SDK at which the app remains stable. Other areas of focus involved testing the appearance of the UI across different screen sizes and resolutions. This testing stage helped set up for the larger scale alpha and beta tests, as we could

get a good idea of the device range we were targeting and it ensured the core features of the app were fully functional and accessible. Internal Testing took place across 10 different devices, ranging between physical devices and emulators. Table 5.1 shows the list of devices and operating systems that were tested at this stage:

Xiaomi Pocophone F1 - API 28 (Pie)
Motorola Moto E - API 19 (KitKat)
Xiaomi Mi Max 2 - API 25 (Nougat)
Lenovo Zuk Z1 - API 23 (Marshmallow)
Google Pixel - API 21 (Lollipop)
Google Pixel - API 19 (KitKat)
Nexus 6 - API 21 (KitKat)
Nexus 6 - API 21 (Lollipop)
3.7 inch FWVGA display slider emulator - API 25 (Nougat)
Nexus S - API 23 (Marshmallow)

Table 5.1: Table of devices and operating systems used for testing

As seen by Figure 5.1, the screen sizes vary from 3.7 inches to 6.4 inches, with resolutions from 540 x 960 pixels to 1080 x 2246 pixels. This gives us a perfect range to assess how the UI adapts to different screens. This revealed a bug where small screens (smaller than 4 inches) did not display the left and right leg buttons. This was due to incorrect layout constraints in the XML file for the Body Screen. The app was also tested across different versions of the Android platform. Any new Application project on Android Studio will by default target the latest Android SDK and, as of this moment, have a minimum SDK of 23 (Marshmallow). It is the developer's job to decide whether this minimum SDK version should be lowered or not, depending on compatibility issues of APIs used in the application. Without adequate testing of SDK versions, some users would be able to download the app with risk of crashes and malfunctioning.

After multiple tests progressively lowering the SDK version, it was observed that issues started to arise with the KitKat operating system. This was expected, as the Android update documentations for APIs 19 and 21 clearly state behaviour changes for Permissions (*Android 5.0 Behavior Changes — Android Developers*, n.d.), reading from storage, or bitmap handling (*Android 4.4 APIs — Android Developers*, n.d.) . All of which are done within the app. The CameraOpeningActivity() classes were crashing

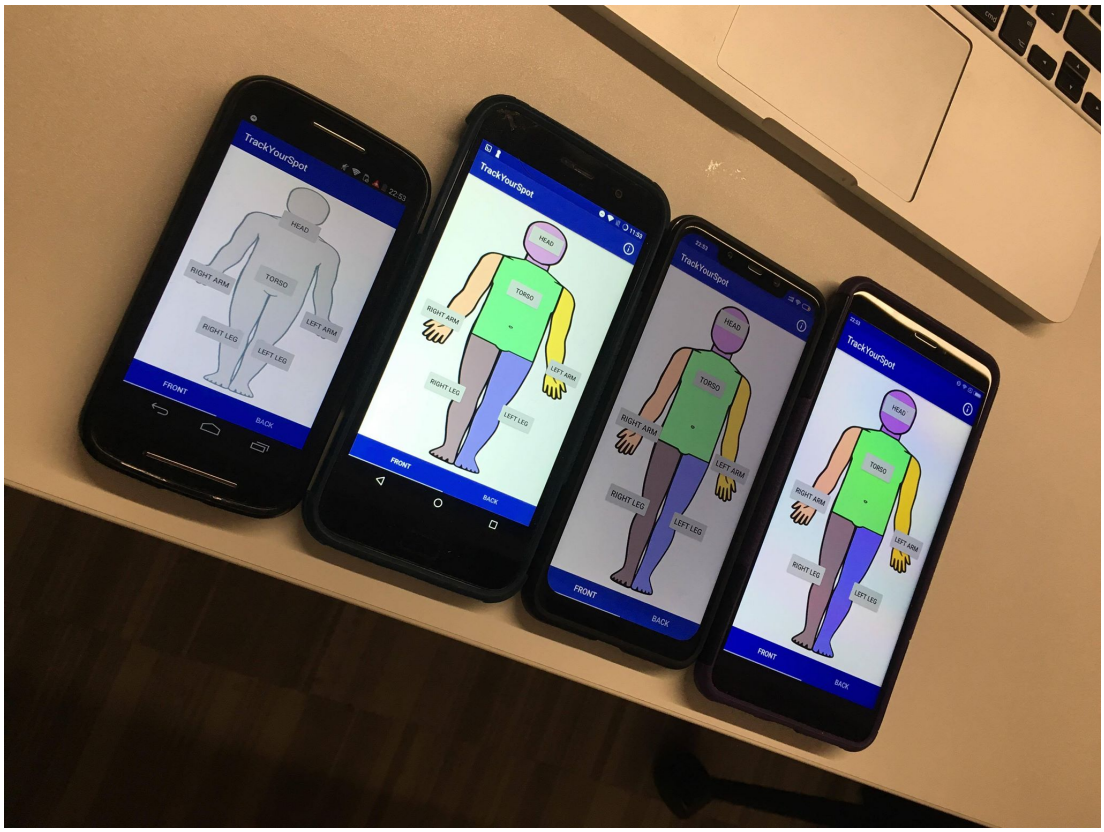


Figure 5.1: *TrackYourSpot* app being tested across different devices

due to a multitude of problems with Camera/Storage permissions, Intent data flows and File URI's. Some of these issues could be fixed by adding a simple if statement checking the current Android version of the device, for example:

Listing 5.1: Checking the Android Build Version

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
    takePictureIntent.addFlags
    (Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
} else {
    val clip = ClipData.newUri(contentResolver, "clipData", photoURI)
    takePictureIntent.clipData = clip
    takePictureIntent.addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
}
```

These workarounds made the app stable in some API 19 devices, such as the Google Pixel, however other same OS devices would still run into crashes, and hence it would be unwise to make the app available for these. Under less time constraints, it would be ideal to improve support for older Android versions, however as shown by Google (*Distribution dashboard — Android Developers*, n.d.) on their publicly available distribution dashboard (Figure 5.2). Devices with API's under 21 only account for 11.1% of all devices, meaning our app is still available for almost 90% of devices. At this point it was decided to set the minimumSDK to 21, and focus on other areas of the project. However, as the app's main user base will not have have up to date Android versions, the usage stats will determine whether developing the apps compatibility becomes a priority again.

5.2 User Acceptance Testing

User acceptance can be defined as the last stage of the software testing process. At this stage, the app is exposed to real end-users. The two test stages in User Acceptance Testing were:

1. Alpha Testing
2. Beta Testing

These tests were smoothly supported by Google's Android Developer Console platform, allowing for easy deployment from one to another and user friendly controls to

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%

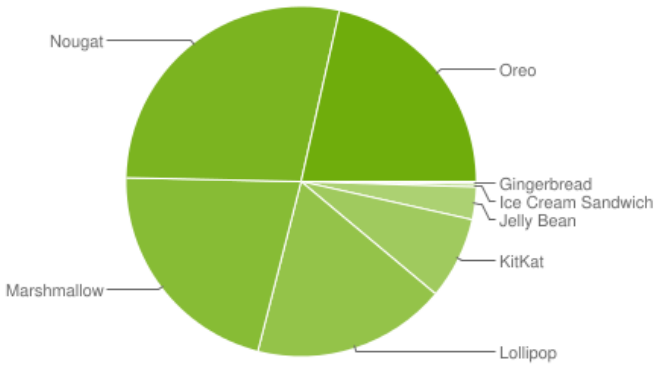


Figure 5.2: Android Device Distribution of Platforms

manipulate these. The two sections below will delve into the details of these, describing bugs and issues that arised in the process.

5.2.1 Alpha Testing

After the app was confirmed to be stable across the range of Android versions and screen sizes in the internal testing stage, the app was published on the Google Play Store under a private URL (Figure 5.3).

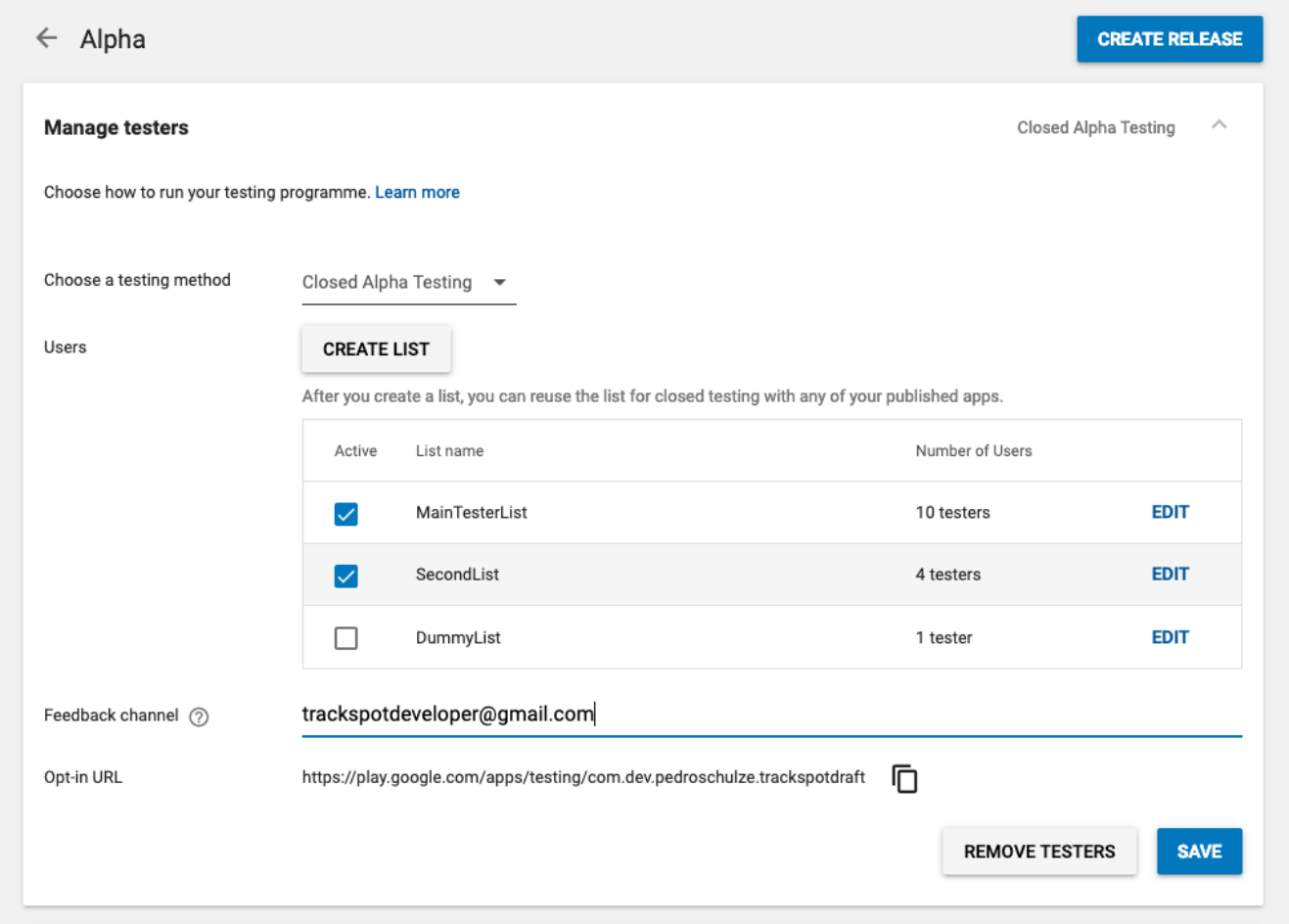


Figure 5.3: Alpha Testing Tester selection, feedback channel and download URL

Any testers could be simply added to the tester mailing list and sent the URL to the *TrackYourSpot* application to download on their devices. As Google Play Developer's console dictated, all testers were required to have Gmail accounts, somewhat limiting the scale at which alpha testing could be carried out. Despite this limitation, 14

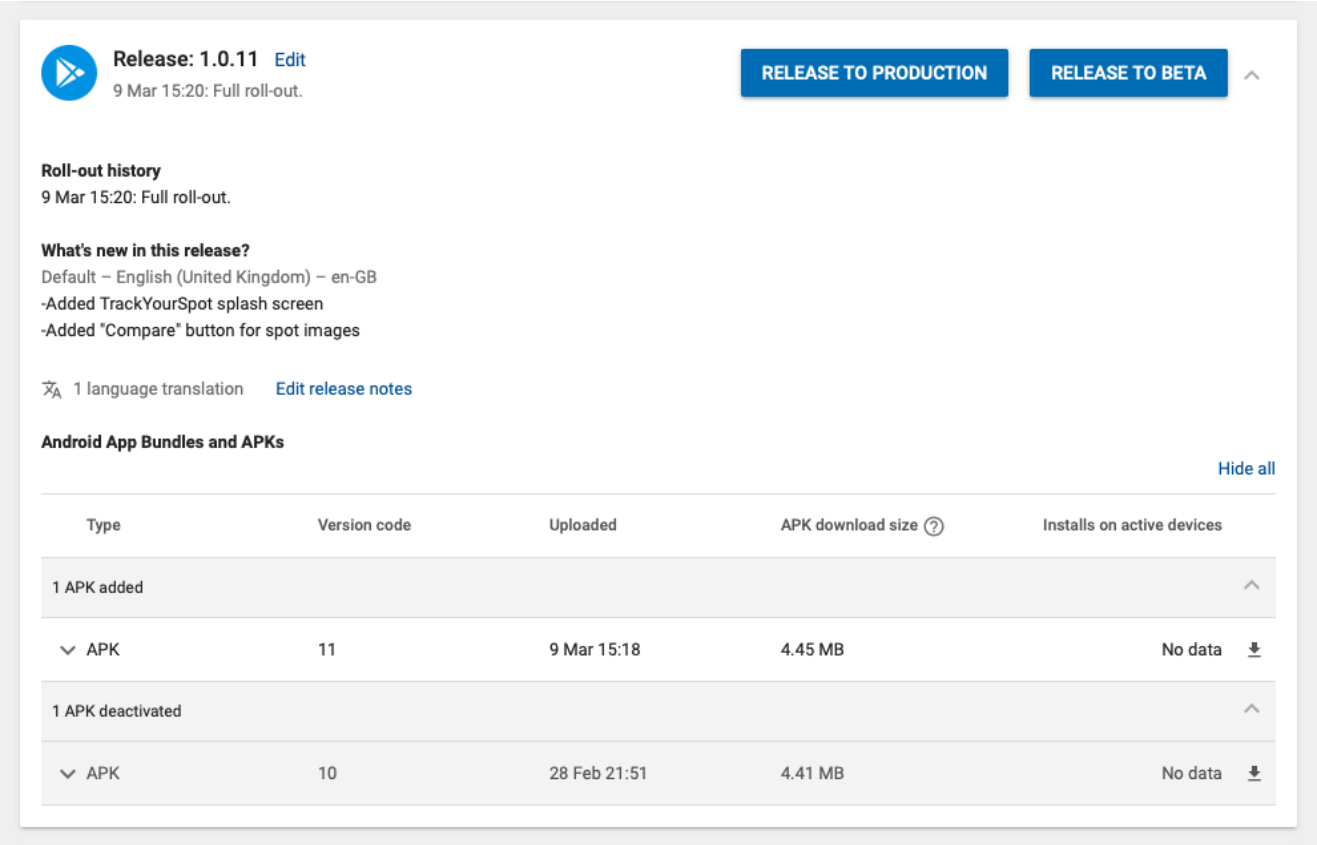


Figure 5.4: Alpha Release APK list

testers were selected throughout alpha testing. The group was mainly composed of penultimate and final year Computer Science students at The University of Edinburgh.

It was clear that the testers's background and computer literacy would be ideal to spot any hidden or convoluted bugs that escaped Internal testing. This resulted in many small fixes and 11 updates (Figure 5.4) during the 2 weeks of alpha testing. The main functionality bugs encountered were:

- Not handling the "temp" folder or corrupt image files accordingly.
- App crashes and unwanted interactions if the Camera or Cropping screens were closed throughout the process.
- App crashes if the user deletes the only photo of a spot.

5.2.2 Beta Testing

Once alpha stage was complete, an app release was pushed to the beta stage. This proved to be a significant milestone for the project, as in an Android application development context, the app was now fully available to the public. Android apps in beta stage are fully available to everyone (likewise to a production-level finished app), with the difference of the app having an "Unreleased" tag next to its name (Figure 5.5) and the user being able to send direct feedback to the developer. This meant the app was now ready and fully available for anyone and the project would be able to receive feedback and reviews from anonymous people.

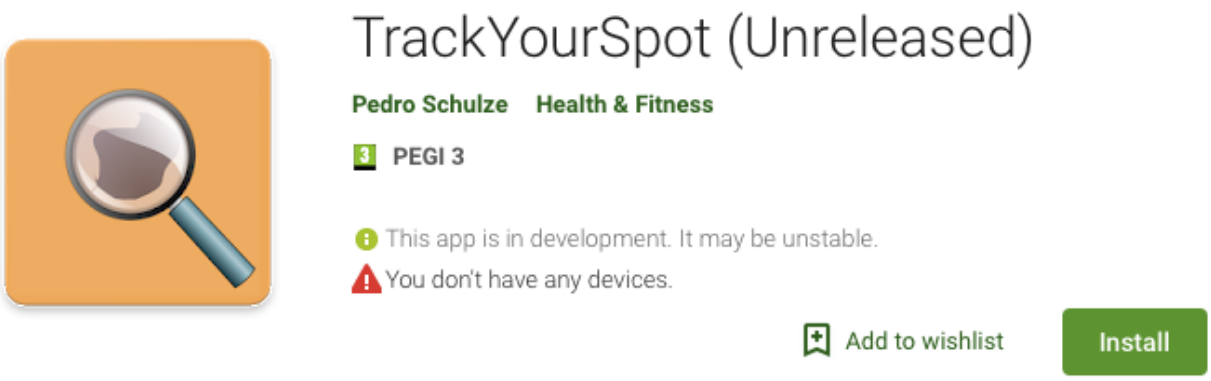


Figure 5.5: Google Play Store's display of *TrackYourSpot* in Beta testing

Before this point, it was crucial to properly execute both the internal and alpha testing stages, as any significant bugs or incompatibilities would have been disastrous at these early stages of an app's release. As described by (Joorabchi, Mesbah, & Kruchten, 2013), "If users like an app, they download and start using it. If not, they delete it and move on immediately. If they really like it, they rank it high; Or if they really dislike it, they go on social media and complain. Thus, a low-quality release can have devastating consequences for mobile developers.". This quote refers to the official release of an app, but Android's user ratings are available since the beta release, and hence this milestone carries an equal responsibility.

Issues discovered during beta stage include:

- Information screen text overflowing the screen on smaller devices
- After an update a typo was found in the body part labels of the *OldSpotScreen*

- App crashes if a spot is given the same name as a previously saved spot

5.3 Preference Testing

An initial plan was made to carry out *A/B Testing* with multiple app design options. The process can be defined as randomly assigning users into segments, displaying a different design variant to each and comparing them based on "Metrics of interest, ranging from runtime performance to implicit and explicit user behaviors and survey data" (Kohavi, Longbotham, Sommerfield, & Henne, 2009). This could be neatly done in parallel with the think-aloud study, however, as the project developed, it was clear that there were limitations and constraints to taking this approach:

- **Nature of design screens being assessed** - The main design features being assessed were trivial visual or appearance changes on a single screen. These ranged from a choice of colours to button visibility and/or position. If the design decisions being assessed significantly changed how a core task is carried out, such as adding or comparing a spot, then completing A/B tests would make more sense.
- **Response sample size** - Due to the limited availability of elderly candidates (Over 50 years old) for the think aloud study, the results for design choices would be more biased towards the younger users which were participating in the study, therefore not giving an accurate representation of the superior design in the eyes of more realistic end users (older individuals).

The factors above suggested that trading part of the study's formality for more plentiful results and statistics would be beneficial. This incited to take a more general approach, arriving at the concept of *Preference Testing*. In psychology, a preference test can be defined as an analysis where consumers are asked to express a preference between rival products (M.S., 2015). In the area of software design, we can translate it to presenting users with multiple visual designs to determine which one they prefer and why. This was done through the method of an online questionnaire. Reasons for this include:

- **Statistics Analysis** - Availability of predefined statistics visualization, filtering, and analysis tools for design preference questions. A straightforward comparison of designs can be done by measuring preference votes.
- **Time and Participation Numbers** - Once the survey's final version was com-

plete, it was evidently easy to share and collect responses. The survey would take 2 minutes to complete and ensure that a much higher volume of opinions was collected.

- **Qualitative Feedback** - Allowing for both open ended and inclusion of demographic questions would provide useful data in terms of why users prefer design A/B.

The questionnaire form can be found in Appendix A. The choice of questions had been carefully chosen after an initial pilot survey had been designed. This initial design was lacking background information on the project. In addition, the questions weren't always clear and often had ambiguous choices or answers. The reliability of the survey was evaluated, putting particular emphasis on whether the questions being asked were actually understood, following the advice of using very simple language and avoiding little-known words (Gendall, 1998). Open ended questions were kept to a minimum and Likert-type answer scales (Boone & Boone, 2012) were used.

5.3.1 Results

The survey was circulated through the 4th year undergraduate Informatics group, varied groups on social networks and friends and family of everyone involved in the project. As a result, a total of 46 responses were collected. Figures 5.6 - 5.13 show answers to both demographic and design questions in different charts. We can analyse the results below:

1. **Please specify your age**
   Over 2/3 of the survey participants were between the ages of 18 and 30. This is far from ideal, as the app's target audience is the middle-aged to older population. One of the biggest challenges of the project's testing and evaluation stage was collecting data from older users. This is clearly shown by this data. Nonetheless, 32.61% of the responses were from people over the age of 30. This combination of figures should provide some interesting answers to the rest of the survey and make an attempt at balancing answers across all age groups.

2. **Please specify your gender**
   More than 3/4 of the answers came from the male gender, this had to be expected when the survey was circulated through a university group with male dominated

demographics. However, this turned out to be an advantage for the study, as according to research by (Fisher & Geller, 2013), "men are 55 percent more likely to die of melanoma than women" between the ages of 15-39, converting the answers of male individuals in that age group into very valuable participants for the study.

3. **What smartphone operating system do you regularly use?**
   56.52% of users are regular iPhone users and 43.48% use Android. As the two clearly dominating operating systems in the market, this is completely expected. These stats don't mean much on their own but they will become useful to understand the design preferences of users, for example iPhone users might dislike more Android specific designs and viceversa.

4. **What do you consider the most important in app design? (Tick all that apply)**
   This question acts as a build up to help in understanding the results of questions 6-7. "Ease of Use" was the most popular option, with over 80% of participants including it on their list of preferences. At 65% and 63%, we have "Clean look" and "Intuitiveness". On the other side, "Colors and Patterns" and "Text and button size" seem to be less important to users, being only selected 18% and 24% of the time. This should allow for good predictions on what design users will prefer in questions 6 and 7.

5. **How likely are you to use an app to monitor your skin spots?**
   over 58% of participants claim they are unlikely or very unlikely to use this kind of app. These numbers do seem a bit low, however most likely expected from a group of mainly 20-25 year old student. Looking back, the phrasing of the question could have been improved by adding "in the future", since it is likely that some participants wouldn't use it at this point in time, but more likely in the future. Another option would be to rephrase the question to "How likely are you to recommend this app to an elderly relative?", since it would provide better feedback on whether the concept and design of the app are adequate.

6. **Rank these 4 designs from best to worst (Best=1, Worst=4)**
   Looking at the 4 designs being assessed (Appendix A), users seemed to rank Design C as the best over 52% of the time, this makes it a clear favourite compared to Design D, which was only voted as the favourite 26% of the time. Designs A

and B performed worse, being only voted as the favourite 10.87% of the time. It is worth noting that designs A and B seemed to be popular second choices, with 31% and 33% frequencies. Analysing the designs, the popularity of Design C seems logical with respect to the answers to question 4. Most users preferred a clean and intuitive look, while dismissing the need of labelled buttons.

7. **Which of the two designs (A or B) would you prefer using?**
   The results to this question were the least expected, over 80% of participants preferred Design B over A (see Appendix A). The design in A tries to emulate official Android design guidelines, displaying an action float button. Design B is closer to iOS guidelines, displaying the button on the right corner of the toolbar. Knowing around 57% of users are iPhone users (Question 3 results), it was predicted that design B would come out slightly on top. With the clear popularity of design A, it is obvious that replacing the "+" button on the toolbar with a floating action button is the right choice.

8. **Do you have any comments about these designs or the concept of the app?**
   Figure 5.13 shows a few responses that participants provided for this question. Overall, the feedback was very useful, most of which can be used to further improve the app's design. A few interesting ideas people suggested include:

   - Pick better color schemes for the body images, also altering the opacity
   - Let users choose the exact position of a skin spot
   - Increase the size of the toolbar buttons
   - Display a dialogue if no spots have been added

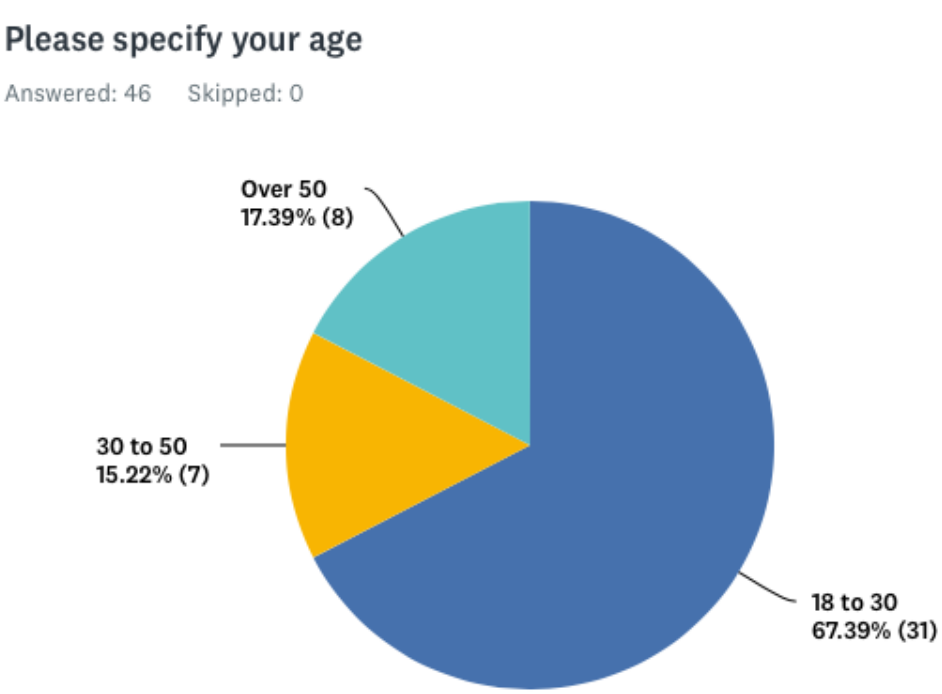


Figure 5.6: Question 1 results of the Preference Testing Survey

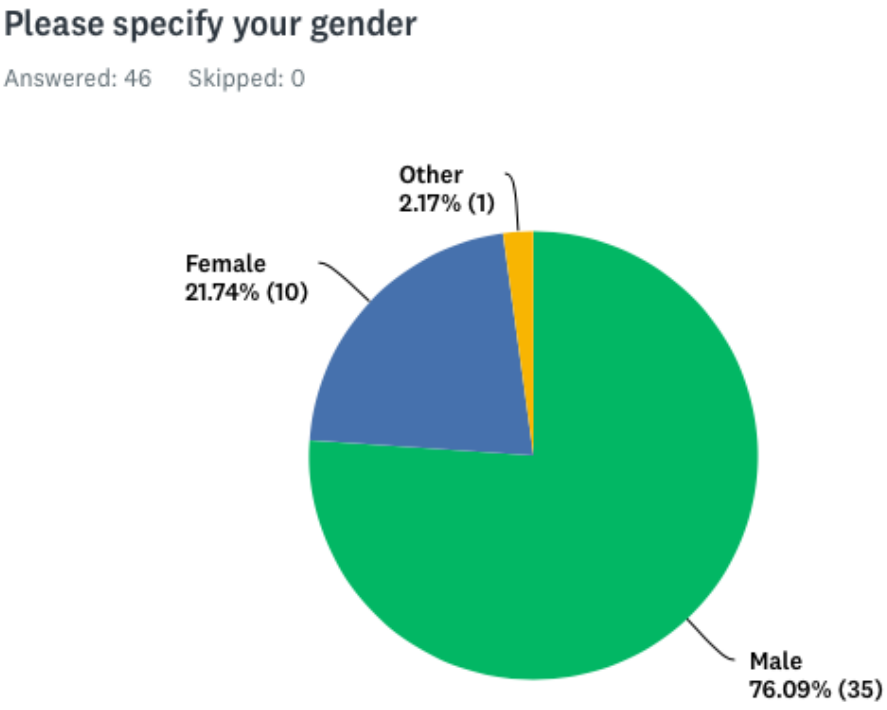


Figure 5.7: Question 2 results of the Preference Testing Survey

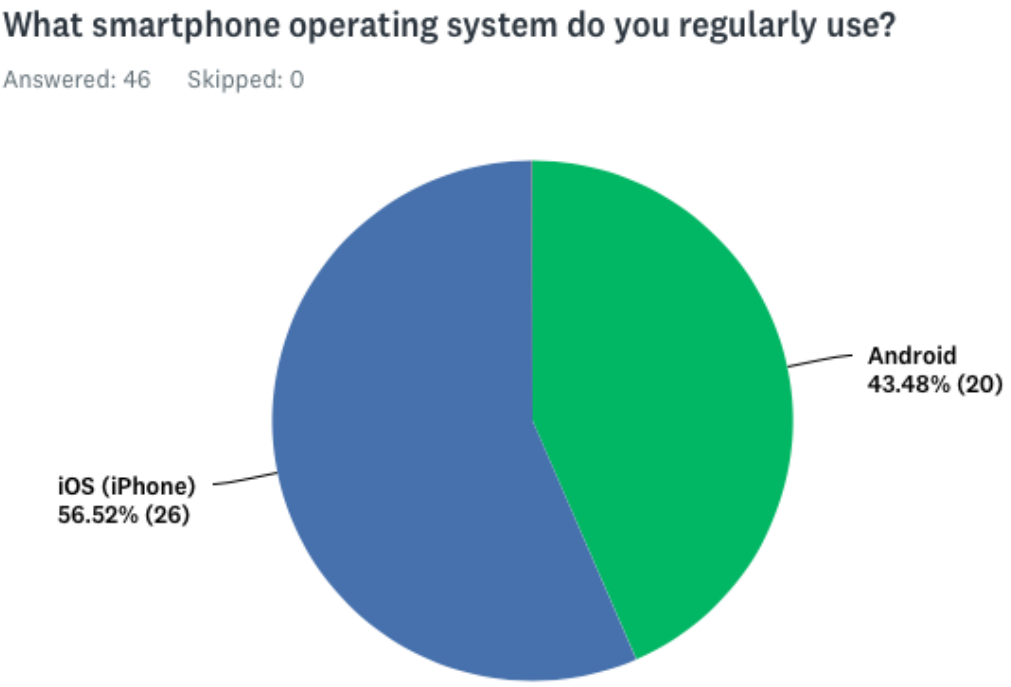


Figure 5.8: Question 3 results of the Preference Testing Survey

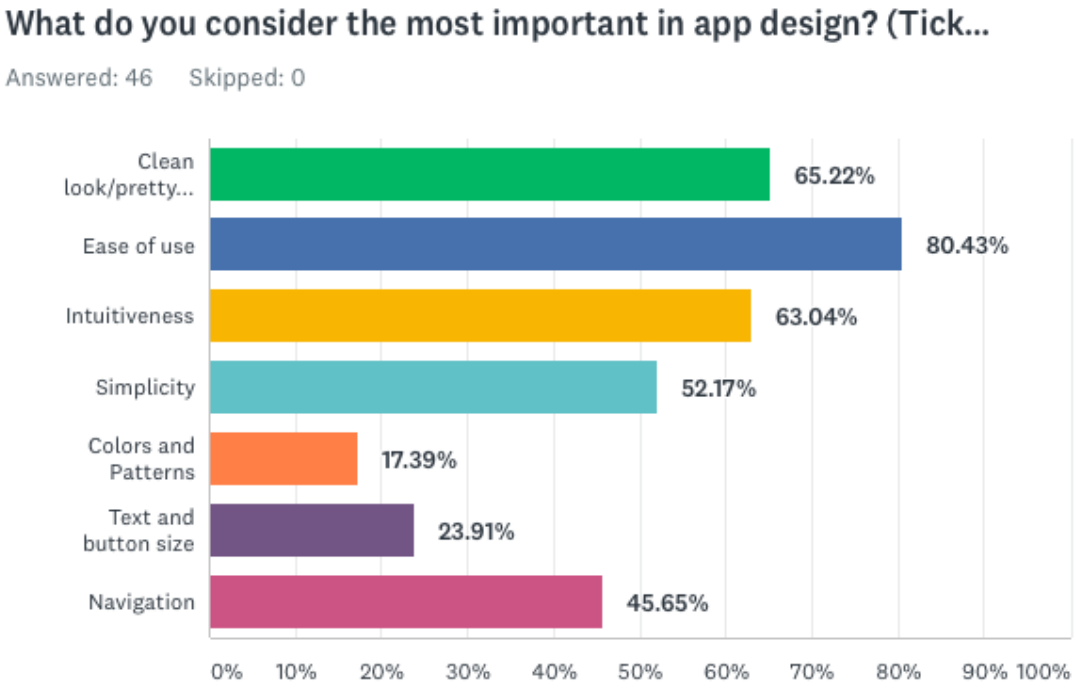


Figure 5.9: Question 4 results of the Preference Testing Survey

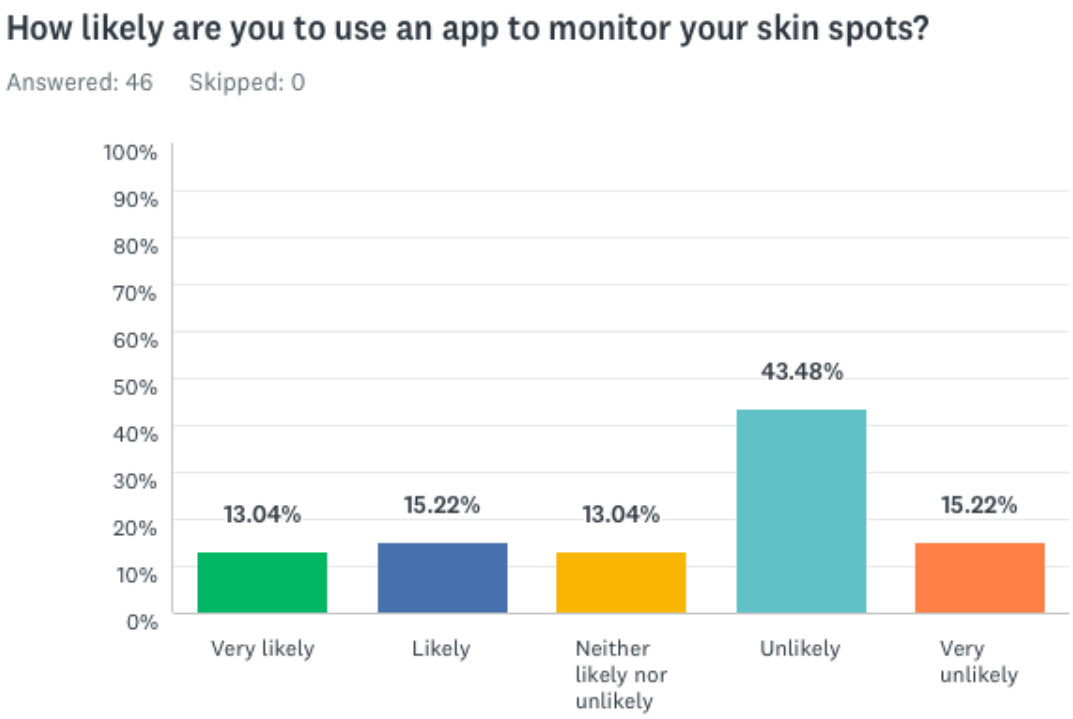


Figure 5.10: Question 5 results of the Preference Testing Survey

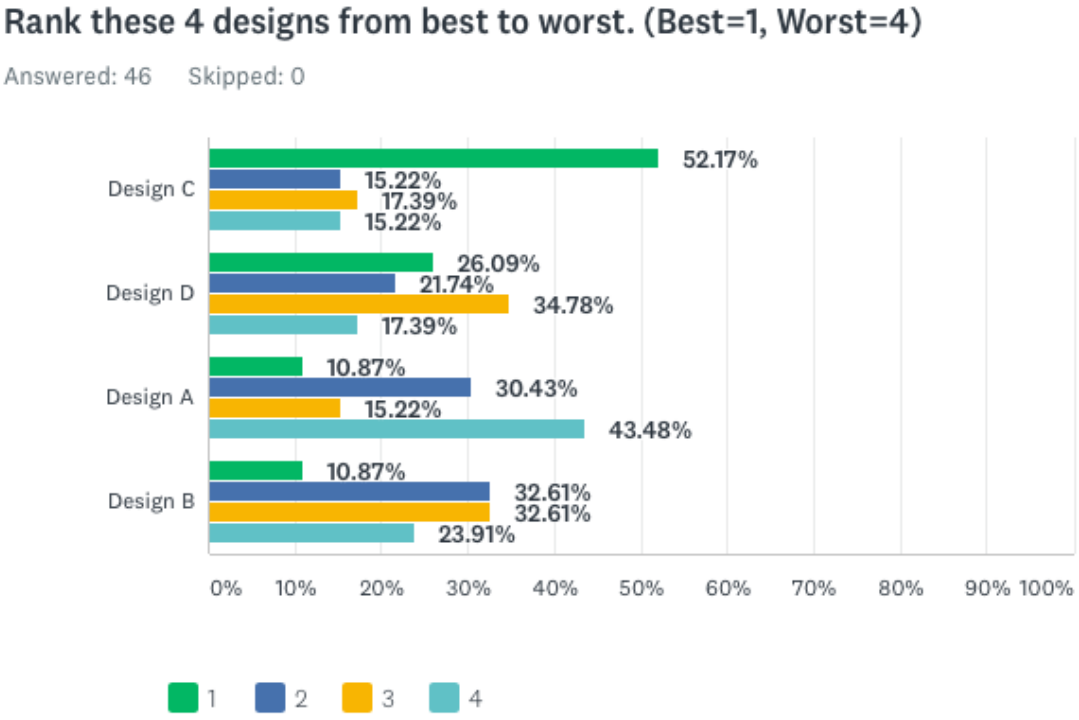


Figure 5.11: Question 6 results of the Preference Testing Survey

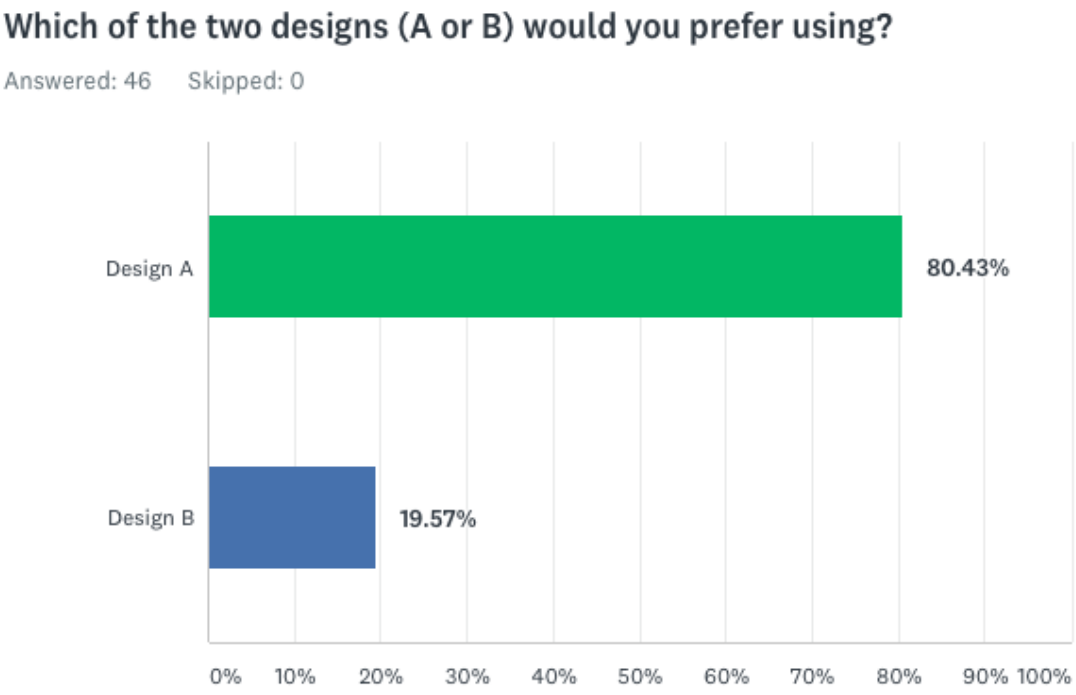


Figure 5.12: Question 7 results of the Preference Testing Survey

Figure 5.13: Question 8 results of the Preference Testing Survey

5.4 Usability Evaluation

The evaluation of the app's user design and experience was carried out through a series of popular methodologies in UI evaluation. Three usability studies were carried out, with the goal of revealing problems that would go unrealised with a single usability evaluation (Tselios, Avouris, & Komis, 2008). Prof. Tselios' findings declare that the value of combining usability studies lies on providing "different but synergistic perspectives on various aspects of the user interface". While this claim is based on evaluating educational applications, one would assume this translates to most design evaluations. The three studies performed were:

- Think-Aloud
- System Usability Scale Evaluation
- Interviews

The individual studies are explained in-depth below. The justification and execution of these is emphasized, with the results of all three studies summarized at the end of the section.

5.4.1 Think-Aloud Sessions

The chosen study was a *Concurrent Think-aloud*, where users are asked to walk through tasks and and articulate what they are doing, as opposed to a *Retrospective Think-aloud*, where they complete the tasks in silence while being recorded (Hanington & Martin, 2012). The main problems with the concurrent think-aloud approach are:

- **Unnatural feeling** - Thinking aloud while using an app often seems unnatural for users. This means they will often unconsciously leave some thoughts out, making it impossible to know everything that goes through the user's mind (Rubin & Shirk, 1996).
- **Lack of quantitative measures** - Asking users to think aloud while using the app significantly slows down the process, making it useless to measure variables such as time taken on task. The useful data that can be collected from this study revolves around *where* and *why* problems exist.

Despite these drawbacks, the choice of this study can be justified in several ways.

Firstly, the low cost of the study, as only a notepad and an Android device are needed. Secondly, a low number of participants will usually highlight most of the issues with the design being evaluated. Thirdly, the tasks being assessed were considered to be of easy-intermediate difficulty. As denoted by (Charters, 2003), think-aloud studies should be done with systems that require "more than an automatic response but should not be cognitively overwhelming.", hence suggesting think-aloud to be appropriate for this project.

Before the study could commence, an ethics review of the study had to be submitted to the school's ethics board. Once approval was granted, participants were asked if they would like to take part in the study. Each of them would then have to agree to sign a consent form, they would also be shown an information sheet before they started the study (See Appendices B and C). The think-aloud script was then read to participants. Subsequently, they were asked to carry out these four tasks one step at a time:

1. Add a new spot in your **front right arm**. Give it the name "Small spot"
2. Add a new image to the previously created spot "Small spot"
3. Compare two images of the previously added "Small spot".
4. Without exiting the *TrackYourSpot* app, email two images of the previously added "Small spot" to the following address: trackspotandroid@gmail.com

While carrying out these tasks, the device's sound recorded app was used in the background, this allowed for audio files to be revisited at a later stage. Participants were constantly reminded to talk out loud, while the researcher took notes of their comments and behaviour. The researcher was not allowed to answer any questions and the participant had to decide when they believed a task was complete. Each instance of the study would take between 20-30 minutes on average.

5.4.2 System Usability Scale

To achieve a quantitative measure of usability for the app, the *System Usability Scale* was used. As John Brooke describes it (Brooke et al., 1996), it is "robust and reliable", and a powerful evaluation tool that "correlates well with other subjective measures of usability". The SUS form used in question can be found in Appendix D. This form was presented to participants after their participation on the think-aloud study. The

effectiveness of this evaluation method relies on the participant having fresh memory of the tasks and features of the app.

Once every participant completes the form, a score is calculated between 0-100. Even numbered questions receive a value of 5 minus the scale position, odd numbered questions are simply the scale position minus 1. The sum of these scores multiplied by 2.5 returns the SUS Score.

While the SUS evaluation system has been widely used for a variety of research projects, its effectiveness can still be criticised, particularly when applied to projects such as this one. These issues could be:

- **Lack of Adaptability** - While the form's questions can generally be modified and adapted to your project, it is not clear whether the changes will alter the validity of the SUS score, weighting system, or the overall study.
- **Imprecise Terms** - Statements such as *I would like to use the system frequently* or *needed to learn a lot* can have different meanings to different users. In addition, an average app would consider usage of once a month an infrequent time frame, however, for a skin spot monitoring app, this would be very frequent.
- **Simple Language** - Terms such as "cumbersome" or "various functions in the system were well integrated" were used in the form. As a general rule, technical terms should be avoided and language used should be easily understood by users with no technical knowledge (Harrison, 2007).

5.4.3 Interviews

Following the completion of the SUS form, users were asked a series of interview questions about the overall usability of the app. Typically, one of *Interviews* or *Focus Groups* is used as a qualitative measure of usability. Following the decision to carry out think-aloud studies, performing interviews seemed like the obvious choice. Their flexibility and ability to be used together with other evaluation methods was worthwhile. Performing interviews after another usability activity allows to better understand the issues and results in depth (Courage & Baxter, 2005). The following five questions were asked:

1. Which tasks did you find particularly easy to follow?

2. Which tasks did you find more challenging?
3. Are there any changes you would make to the design of the app?
4. Was the website useful in helping you learn how to use the app?
5. Did you find the information displayed on the information screens relevant?

The interview followed a *semi-structured* type design, where the set of questions above were used as a starting point. Followup questions and deviations were made frequently whenever participants mentioned an interesting idea or concern. Answers to questions were also recorded and written notes were made when appropriate.

5.4.4 Results

A total of 9 participants agreed to take part in the study. The feedback collected from the first few participants was particularly insightful, after that, the value of the study started diminishing as users started to point out the same issues over and over again. At this point, it was decided to not continue the study, but perhaps repeat it in the future once the issues have been addressed. All 9 participants agreed to take part in the study. This involved signing the consent form, participating in the Think-Aloud and Interview stages, and finally filling in the System Usability Scale Form. We will analyse the results of these studies individually.

In the case of the Think Aloud study, the analysis of the data was done by listening to the recordings and reviewing the notes taken at each experiment. Both of these were analyzed to determine if there were any patterns in how users interacted with the app. This consisted of the problems where users got stuck on or how they went about solving problems. The main issues highlighted for each task were:

1. **Add a new spot in your front right arm. Give it the name "Small spot"**
   Users found this task relatively straightforward, they clicked on the right arm straight away, proceeded to take a photo of a spot and crop it, knowing that they had completed the task when returning to the main menu. Occasionally the participant would add the button to the back "Right Arm" instead, quickly realising they hadn't paid enough attention to the task. Participants commented on the clarity of the UI while completing this task. As a small issue, some participants appeared to not zoom in enough when cropping the photo, this would make the

spot harder to inspect when comparing it on task 4.

2. **Add a new image to the previously created spot "Small spot"**
   This task was completed with more difficulties than the previous one. The most common mistake was participants adding a completely new spot to the right arm instead of adding a new photo to the existing spot. Participants showed difficulty in realising the "Small spot" list item represented a folder that could be clicked on, instead assuming it was simply an image. Some users realised their mistake when trying to complete the next task, as they weren't able to compare two separate spots. It was also pointed out that the button for adding a new photo looked too similar to the button for adding a new spot, implying these should be clearly differentiable.

3. **Compare two images of the previously added "Small spot".**
   The success of this task was very dependent on whether task 2 was carried out correctly, the participants that added the 2 photos correctly usually had no issues comparing them. Some participants realised they had to add another photo to be able to compare, and 1 participant required guidance to be able to complete the task. Another participant thought opening the images individually and swiping left and right was enough to compare them, and declared the task finished before actually comparing them side by side. Most participants pointed out that the compare screen was missing an option to zoom in to the images, this was particularly the case when they didn't crop the images to an adequate zoom. This suggests there could be better guidance on how to take photos.

4. **Without exiting the TrackYourSpot app, email two images of the previously added "Small spot" to the following address: trackspotandroid@gmail.com**
   Compared to the other tasks, this task was completed very easily, users spotted the "Email" button on the compare screen straight away and proceeded to email both photos to the address. A few participants pointed out the lack of a dialogue message to know if the email has been sent. Overall this task was completed very successfully.

After the think-aloud study, participants were asked to fill in the SUS Form. The score for each participant was calculated individually. The overall result was positive, with the average SUS score being 73.6/100. The highest score given was 87.5 and the lowest was 55. According to the government's SUS scale interpretation guidelines (Affairs,

2013), results above 68 are above average, indicating there is room for improvement but nevertheless a good performance of the app in terms of usability.

Once the SUS Form was completed, users were asked a series of questions as part of the interview stage. Responses to each question are analysed below:

1. **Which tasks did you find particularly easy to follow?**
   Participants agreed that tasks 1 and 4 were the easiest, with some participants pointing they were all easy enough.

2. **Which tasks did you find more challenging?**
   As depicted in the think-aloud results, most participants pointed out task 2 was not clear enough and hence more difficult. One participant gave an interesting idea of including a folder icon next to each item in the list, indicating it could be opened.

3. **Are there any changes you would make to the design of the app?**
   Some very interesting responses were provided to this question, a few ideas pointed out:

   - Showing multiple spot images per thumbnail, making it clear the spot can contain multiple photos
   - Changing the zoom gesture to a pinch zoom
   - Add more guidance text for each screen. e.g. “Click + to add a spot” under the toolbar
   - Add password protection to the app

4. **Was the website useful in helping you learn how to use the app?**
   Users pointed out they only skimmed through the website, checking out the tutorial for one or perhaps two tasks. Most participants claim it wasn’t particularly useful in this case because they hadn’t tried the app before reading through the site. Nevertheless, they complimented the idea of having a website in case some people find it useful.

5. **Did you find the information displayed on the information screens relevant?**
   Overall, users found the information useful, particularly mentioning the information on skin spot appearance changes and consulting a doctor. Participants thought it was a good idea to make sure the user knows the app is not a sub-

stitute for medical attention. A few participants mentioned the tutorial could include more guidance on actually using the app.

Chapter 6

Conclusion

6.1 Completion of Initial Objectives

The goal of the project was to develop, evaluate and deploy an Android app with the purpose of helping users with early detection of skin cancer. The project's focus revolved around the successful completion of the main phases in mobile app development.

As per the technical tasks, the app allows adding new spots or new photos of a spot. The app also allows comparing two images of a spot side by side. These images can then be emailed to a user specified email address from within the app. Information screens are also presented to users on the app's home screen. To accommodate these core features, the design and implementation stages were carried out, delving into the process and decisions behind the app's UI and implementation.

Secondly, the app's evaluation was completed through user testing and usability evaluations. This process was done through stages of internal, Alpha, and Beta testing. The app's usability was evaluated through the use of think-aloud studies, interviews, system usability scale assessments and preference tests.

Thirdly, the publishing tasks of the app included deploying a website to guide users in performing all of the core features of the app. Lastly, a stable version of the app was published on the Google Play Store, acting as a milestone for the successful completion of the project.

6.2 Future Work

With the unique nature of app development, it can be sometimes difficult to deem a project as complete. There are always new ideas to be implemented, and the rapid growth of the Android platform will guarantee to bring new bugs and incompatibilities to every app. Leaving this aside, if the project was to be continued, the following areas would be sensible to explore:

- **Reminder Notifications** - Implementing reminders for the user to take a new photo of a spot. These could range from push notifications to emails or SMS messages.
- **Accomodate for Mistakes** - Among others, this could include: allowing users to modify a photo's crop, deleting, moving, or renaming spots.
- **In-App Help** - Providing information on app actions and buttons on each screen of the app. This could be done with libraries such as *TourGuide* (Rong, 2018).
- **Image Matching Function** - Exploring the image processing side of the project. Developing a feature that takes two images of a spot and makes them more similar, this facilitates the comparison process. A starting point would be to automatically adjust rotation, scale and colors of both images.

Appendix A

Preference Testing Survey

TrackYourSpot - Design Survey

TrackYourSpot is an app that helps users to keep track of potentially dangerous skin spots. It allows taking photos periodically of different spots and keeping track of them using a convenient body-oriented interface. Further details can be found here:

https://pedscn.github.io/trackyourspotweb/

This survey will take about 2 minutes and it will help determine the app's design. Thank you for taking part!

Next

Powered by SurveyMonkey

See how easy it is to create a survey.

Privacy & Cookie Policy

* 1. Please specify your age

- ◯ Under 18
- ◯ 18 to 30
- ◯ 30 to 50
- ◯ Over 50

* 2. Please specify your gender

- ◯ Male
- ◯ Female
- ◯ Other

* 3. What smartphone operating system do you regularly use?

- ◯ Android
- ◯ iOS (iPhone)
- ◯ Other (please specify)

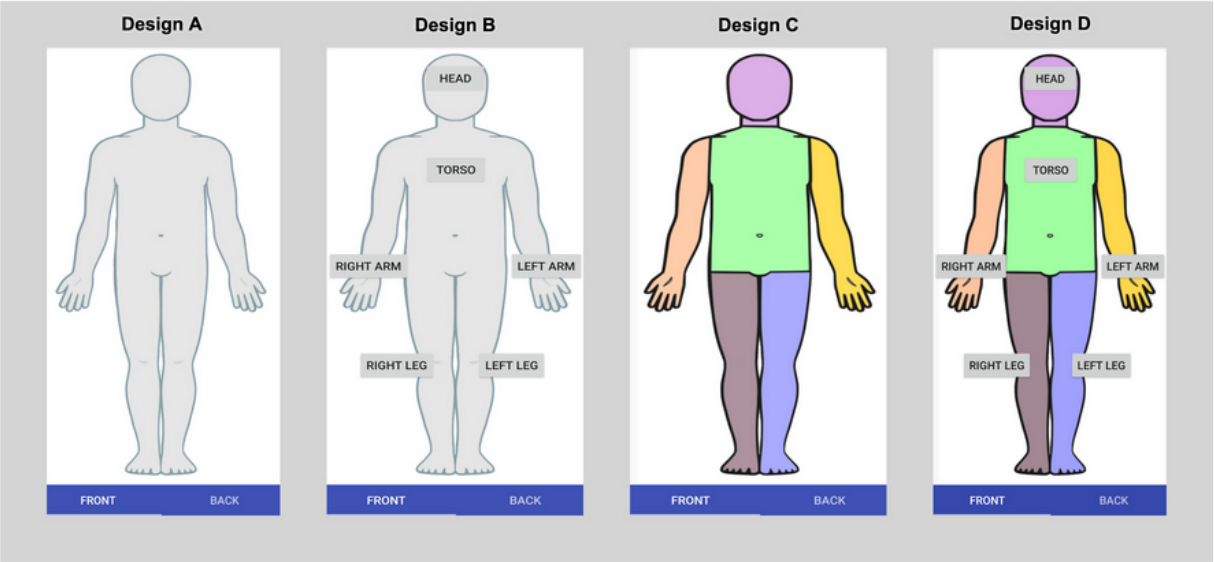
* 4. What do you consider the most important in app design? (Tick all that apply)

- ☐ Clean look/pretty appearance
- ☐ Easy of use
- ☐ Intuitiveness
- ☐ Simplicity
- ☐ Colors and Patterns
- ☐ Text and button size
- ☐ Navigation

* 5. How likely are you to use an app to monitor your skin spots?

- ◯ Very likely
- ◯ Likely
- ◯ Neither likely nor unlikely
- ◯ Unlikely
- ◯ Very unlikely

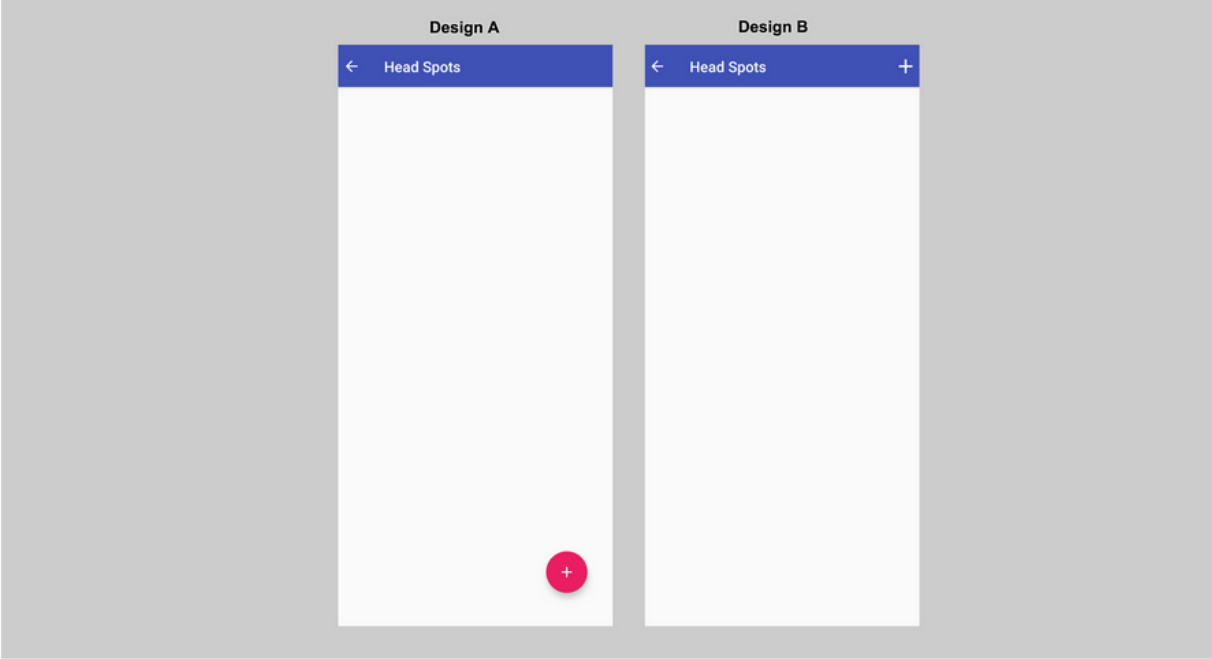
Observe the following four screen designs. This screen is used to select a body part to add a picture of a skin spot. To choose a body part, the user would simply touch it or press the button. Please answer the question below.



* 6. Rank these 4 designs from best to worst. (Best=1, Worst=4)

- Design A
- Design B
- Design C
- Design D

Observe the two designs below. This screen is used to add a picture of a new spot, this is done through the "+" button. Please answer the question below.



* 7. Which of the two designs (A or B) would you prefer using?

- ◯ Design A
- ◯ Design B

8. Do you have any comments about these designs or the concept of the app?

Prev Done

Appendix B

Usability Consent Form

Participant number:_____

Participant Consent Form - #3468

Project title:	**TrackYourSpot**
Principal investigator (PI):	**Pedro Schulze, Daniel Man**
Researcher:	**Pedro Schulze, Daniel Man**
PI contact details:	s1527484@sms.ed.ac.uk, s1535383@sms.ed.ac.uk

Please tick yes or no for each of these statements.

		Yes	No
1.	I confirm that I have read and understood the Participant Information Sheet for the above study, that I have had the opportunity to ask questions, and that any questions I had were answered to my satisfaction.		
2.	I understand that my participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.		
3.	I agree to being audio recorded.		
4.	I consent to my anonymised data being used in academic publications and presentations.		
5.	I understand that my anonymised data can be stored for a minimum of two years		
6.	I allow my anonymised data to be used in future ethically approved research.		
7.	I agree to take part in this study.		

Name of person giving consent	Date dd/mm/yy	Signature

Name of person taking consent	Date dd/mm/yy	Signature

THE UNIVERSITY of EDINBURGH informatics

Appendix C

Participant Information Sheet

Participant Information Sheet

Project title:	**TrackYourSpot**
Principal investigator:	**Pedro Schulze, Daniel Man**
Researcher collecting data:	**Pedro Schulze, Daniel Man**

This study was certified according to the Informatics Research Ethics Process, RT number #3468. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

The project is carried out by:

-Pedro Schulze (s1527484@sms.ed.ac.uk)

-Daniel Man (s1535383@sms.ed.ac.uk)

The project is supervised by:

-Bob Fisher (rbf@inf.ed.ac.uk)

What is the purpose of the study?

We are final year undergraduate students at The University of Edinburgh. As part of our dissertation project, we are developing a mobile app called TrackYourSpot. The goal of the app is to help users track changes in the appearance of their skin spots, by allowing the user to periodically take pictures of a spot and compare them.

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study at any time, without giving a reason. Your rights will not be affected. If you wish to withdraw, contact the PI. We will stop using your data in any publications or presentations submitted after you have withdrawn consent. However, we will keep copies of your original consent, and of your withdrawal request.

THE UNIVERSITY of EDINBURGH informatics

What will happen if I decide to take part?

In this study we are investigating how people carry out basic tasks on the TrackYourSpot app so that we can better understand how to improve the user interface and overall experience with the app. We will ask you to speak aloud while carrying out a series of tasks that will be given to you. After these tasks, we will ask you a few simple questions on your experience with the app. You can leave the study at any time. There is no compensation for participating in this study. All the data that we collect will be treated anonymously, with any pictures being taken with the app during the study being deleted straight away. Audio will be recorded with a mobile device. The session will last approximately 30 minutes.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher/research team (Daniel Man, Pedro Schulze and Bob Fisher).

Who can I contact?

If you have any further questions about the study, please contact the lead researchers,

-Pedro Schulze (s1527484@sms.ed.ac.uk)

-Daniel Man (s1535383@sms.ed.ac.uk)

If you wish to make a complaint about the study, please contact:
inf-ethics@inf.ed.ac.uk
When you contact us, please provide the study title and detail the nature of your complaint.

THE UNIVERSITY of EDINBURGH informatics

Appendix D

System Usability Scale Form

Participant ID: _____

TrackYourSpot - System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

This questionnaire is based on the System Usability Scale (SUS), which was developed by John Brooke while working at Digital Equipment Corporation.

References

Affairs, A. S. f. P. (2013, Sep). *System usability scale (sus).* Department of Health and Human Services. Retrieved from https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

Android 4.4 apis — android developers. (n.d.). Retrieved 2019-03-25, from https://developer.android.com/about/versions/android-4.4.html

Android 5.0 behavior changes — android developers. (n.d.). Retrieved 2019-03-25, from https://developer.android.com/about/versions/android-5.0-changes.html

Android and google play statistics, development resources and intelligence. (2019, Apr). Retrieved from https://www.appbrain.com/stats

Boone, H. N., & Boone, D. A. (2012). Analyzing likert data. *Journal of extension*, *50*(2), 1–5.

Bootstrap. (n.d.). *Free bootstrap themes, templates, snippets, and guides.* Retrieved 2019-03-21, from https://startbootstrap.com/

Brooke, J., et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4–7.

Charters, E. (2003). The use of think-aloud methods in qualitative research an introduction to think-aloud methods. *Brock Education Journal*, *12*(2).

Courage, C., & Baxter, K. (2005). *Understanding your users: A practical guide to user requirements methods, tools, and techniques*. Gulf Professional Publishing.

Developer guides — android developers. (n.d.). Retrieved 2019-03-22, from https://developer.android.com/guide

Distribution dashboard — android developers. (n.d.). Retrieved 2019-03-20, from https://developer.android.com/about/dashboards

Fisher, D. E., & Geller, A. C. (2013). Disproportionate burden of melanoma mortality in young us men: the possible role of biology and behavior. *JAMA dermatology*,

149(8), 903–904.

Gendall, P. (1998). A framework for questionnaire design: Labaw revisited. *Marketing Bulletin-Department of Marketing Massey University*, *9*, 28–39.

Handling bitmaps — android developers. (n.d.). Retrieved 2019-03-21, from https://developer.android.com/topic/performance/graphics

Hanington, B., & Martin, B. (2012). *Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions*. Rockport Publishers.

Harrison, C. (2007). Tip sheet on question wording. *Harvard University Program on Survey Research*, 1–4.

Imagery. (n.d.). Retrieved 2019-02-10, from https://material.io/design/communication/imagery.html

Intents and intent filters — android developers. (n.d.). Retrieved 2019-03-12, from https://developer.android.com/guide/components/intents-filters.html

Joorabchi, M. E., Mesbah, A., & Kruchten, P. (2013). Real challenges in mobile app development. In *2013 acm/ieee international symposium on empirical software engineering and measurement* (pp. 15–24).

Kohavi, R., Longbotham, R., Sommerfield, D., & Henne, R. M. (2009). Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery*, *18*(1), 142.

M.S., P. (2015, Jul). *What is preference test? definition of preference test (psychology dictionary).* Retrieved 2019-03-25, from https://psychologydictionary.org/preference-test/

Nishant Srivastava, n. (n.d.). *App privacy policy generator.* Retrieved 2019-03-21, from https://app-privacy-policy-generator.firebaseapp.com/

Onboarding. (n.d.). Retrieved 2019-02-14, from https://material.io/design/communication/onboarding.htmlself-select-model

Rong, T. J. (2018, May). *worker8/tourguide.* Retrieved 2019-03-25, from https://github.com/worker8/TourGuide

Rubin, J., & Shirk, H. N. (1996). Handbook of usability testing: How to plan, design, and conduct effective tests. *Journal of Technical Writing and Communication*, *26*(1), 97–106.

Sagar, P. (2019, Mar). *Java vs. kotlin: Which one will be the best in 2019? - dzone java.* Retrieved from https://dzone.com/articles/java-vs-kotlin

-which-one-will-be-the-best-in-2019

Santaguida, V. (2011, Nov). *Taking folder and file naming best practices to the next level.* Retrieved from http://www.exadox.com/en/articles/advanced-folder-and-file-naming-best-practices

Take photos — android developers. (n.d.). Retrieved 2019-03-07, from https://developer.android.com/training/camera/photobasics

Tselios, N., Avouris, N., & Komis, V. (2008, 03). The effective combination of hybrid usability methods in evaluating educational applications of ict: Issues and challenges. *Education and Information Technologies*, *13*, 21. doi: 10.1007/s10639-007-9045-5

Twbs. (2019, Apr). *twbs/bootstrap.* Retrieved 2019-03-26, from https://github.com/twbs/bootstrap

Yalantis. (2019, Mar). *Yalantis/ucrop.* Retrieved 2019-03-20, from https://github.com/Yalantis/uCrop