Colbi: A Colourblindness-Themed Android Application

Andrius Girdžius

MInf Project Report
Master of Informatics
School of Informatics
University of Edinburgh

2021

Abstract

Colorblindness affects nearly 8% of male and 0.5% of female population and, depending on the condition type, can introduce severe difficulties in everyday life. There are existing software solutions to dealing with the effects of the condition, but they lack functionality, are inaccessible or require the use of multiple tools.

This project introduces *Colbi* - a new mobile *Android* solution related to colourblindness, now published in the *Google Play Store* and implemented in *Kotlin*. The app features 4 core components, namely colour identification (naming), smart recolouring (*daltonization*), simulation and deficiency assessment tools, as well as a friendly and a minimal user interface.

After design and implementation stages were complete, an evaluation study was run that concluded that user opinions are overall positive on both the functionality and the usability aspects. However, further evaluation should be done with more colourblind participants to truly measure the impact of the proposed solution.

Acknowledgements

I want to thank my supervisor Boris Grot for the continuous encouragement during the project for all my ideas - I had plenty of creative freedom which I am grateful for.

Thank you to my family and friends who were understanding, supportive and overall motivating throughout all years of my studies - I truly do not think I would be the person I am today without you.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Colour vision deficiencies (*CVD*s), also known as colourblindness, affect approximately 8% of male and 0.5% of female population and cause difficulties discriminating certain hues of colours [34]. The condition is, in most cases, inherited, although there are certain types of the deficiency that can be acquired as well due to age, diseases and other risk factors.

Depending on the severity, the condition can significantly affect an individual's everyday life, causing issues with simple tasks such as discriminating between raw and cooked meat, identifying ripe fruits, picking clothes, reading certain study material, and more [21]. Although awareness of the condition is increasing, there are still cases of undiagnosed or unsuspected colourblindness which can cause unnecessary confusion to the individual, thus early diagnosis, typically verified during optometry examinations, is preferred [19].

Unfortunately, there is no cure for most colour vision deficiencies [19]. There are ways to help ease the effects of the condition, such as wearing specially-designed glasses as *EnChroma* [25] or asking your friends and family for support, but these can be expensive and availability may vary [19]. In contrast, most people ($> 80\%$) these days already own a smart mobile device, so mobile applications are one of the most effective and accessible platforms to develop potential solutions for raising awareness of, and improving the life of individuals with, colourblindness [31].

1.2 Research Goals

The primary objective of this project is to *design, implement and evaluate a mobile colourblindness-themed application*. To aid the process throughout the duration of the project, the following research questions are to be considered:

RQ1. How to design and implement a user-friendly, accessible and useful mobile application related to colourblindness? What features should it include? How could it improve over the existing solutions?

RQ2. What is the extent of impact, usefulness and usability of the implemented prototype application?

RQ3. How to improve the proposed solution by utilising the feedback from user evaluation studies?

1.3 Contribution Summary

To partially address *RQ1*, a simple and minimalist interface for an app related to colourblindness was designed using *Figma* [29]. The design includes four core features: colour naming, recolouring to increase discrimination of confusing colours, *CVD* simulations and assessment for common *CVD* types.

To further address *RQ1*, the prototype designed was implemented in *Kotlin* [39] for the *Android* OS, published on the *Google Play Store* [12]. The app features 4 swappable colour naming dictionaries, 3 filters for recolouring using the *LMS Daltonization* algorithm [57] and simulation filters for all colourblindness types, as well as *CVD* testing using *Ishihara* [62] methodology.

To address *RQ2* and *RQ3*, the implementation was tested with 11 participants using the *think aloud* (see section 2.3.1.1) protocol, evaluated for usability using *SUS* (see section 2.3.1.3). The overall feedback was positive and the mean usability score (`83.2`) can be interpreted as `Great` (see fig. 2.5). Feedback was analysed and prioritised using *thematic analysis* (see section 2.3.2.1). All bugs identified were fixed and most feature suggestions were implemented. A new an improved version of the app was released.

1.4 Dissertation Structure

The remaining chapters of this dissertation outline the following:

Chapter 2 introduces the background related to *CVD*s, provides an overview of existing tools in the area, presents the Human-Computer Interaction methods used.

Chapter 3 provides a structured overview of the methodology used to design, implement and evaluate the application developed throughout the project duration.

Chapter 4 presents the process of the design stage and creation of interactive medium-fidelity prototypes of the mobile application proposed.

Chapter 5 describes the challenges and steps related to the implementation of the prototype application design as described in chapter 4.

Chapter 6 explains the process and results of an evaluation study run with the implementation described in chapter 5, details improvements implemented resulting from direct feedback.

Chapter 7 reviews the project outcome by discussing associated challenges and relating to the research questions as detailed in section 1.2, presents proposed future work.

Chapter 2

Background

This chapter introduces the background necessary to undertake further work outlined in section 1.2, including an overview on colourblindness, existing tools aiming to reduce the effects of the condition in everyday life and the *Human-Computer Interaction* research methods to be used in design and evaluation stages.

2.1 Colour Blindness

2.1.1 Overview

As described in section 1.1, colourblindness is a condition that is typically genetically inherited and is a quite frequent occurrence, especially amongst males [34].

Although there are many different types of colour vision deficiencies (see section 2.1.2), some are more prevalent than others. *Red-green* colourblindness, which covers *deuteranomaly* (5% male, 0.5% female), *deuteranopia* (1.5% male, 0.01% female), *protanomaly* (1% male, 0.01% female) and *protanopia* (1% male, 0.01% female), is the most common affecting around 8% of male and 0.5% of female population [60].

2.1.1.1 Issues faced

Colourblind people have many areas of their life affected with varying severity [35]. Although most common types of *CVD*s affect everyday life in smaller ways, for example, being unable to distinguish between a ripe and raw banana or between ketchup and mustard, there are many cases where the effect is considerably more drastic. In fact, colourblindness can affect access to education, some career paths, obtaining a driver's license and more. Additional consideration is needed to provide safety for those with colourblindness, e.g. traffic light LEDs are stacked in a standard order rather than changing in a single unit.

While some issues listed above are unsolvable, there are existing products and applications that help to alleviate some of the smaller issues affecting the life of an individual with colourblindness, see section 2.2.

2.1.2 Classification

Photoreceptor cells in the human retina utilised in colour recognition are called *cones*. There are three types of cone cells with varying sensitivity to different wavelengths of light (see fig. 2.1): *short (S)*, *medium (M)* or *large (L)*, also known as *blue*, *green* or *red* respectively [16].

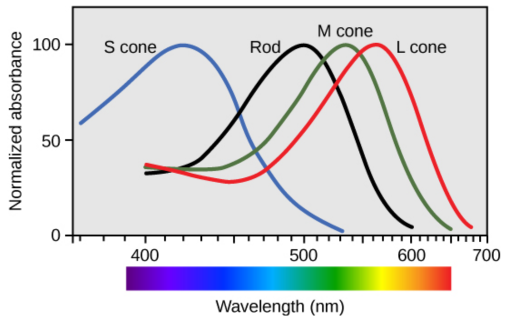


Figure 2.1: The three types of cone cells wavelengths visualised [49]

The different types of colour blindness are classified based on the way the three cone types in the human retina are affected [15]. People with normal (unaffected) vision are called *trichromats*, while those with deficiencies have conditions categorised as followed [45][36]:

1. **Anomalous trichromacy** - all three types of cones are used, but one is altered in sensitivity.
   (a) **Protanomaly** - red cone affected, difficulty discriminating red/green hues.
   (b) **Deuteranomaly** - green cone affected, as with protanomaly, difficulty discriminating red/green hues.
   (c) **Tritanomaly** - blue cone affected, difficulty discriminating blue–green and yellow–red/pink hues.
2. **Dichromacy** - one of the cone pigments is missing.
   (a) **Protanopia** - red cone missing, difficulty discriminating blue/green, red/green hues; reds appearing as black, orange-tinted reds as dimmed yellows.
   (b) **Deuteranopia** - green cone missing, as with *protanopia* - difficulty discriminating blue/green, red/green hues, but without the dimmed effect.
   (c) **Tritanopia** - blue cone missing, difficulty discriminating certain shades of blue/gray, purple/black, green/blue and orange/red.
3. **Monochromacy** - known as ”total colour blindness”.
   (a) **Cone monochromacy** - two of three cones are non-functional, which results in vision being reduced to only black/gray/white shades.
   (b) **Rod monochromacy**, also known as *achromatopsia* - all cone cells are non-functional and only *rod* (light) cells are used, resulting in severely affected visions only perceiving sources of light.

2.1.3 Deficiency Assessment

Individuals are typically diagnosed with colour vision deficiency using a test. Depending on the subject's background, such as age and occupation, one of several different types

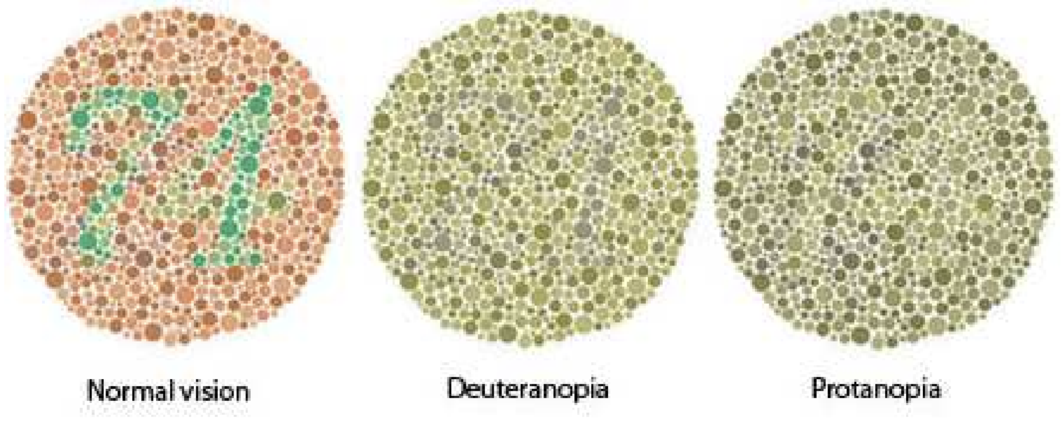


Figure 2.2: *Ishihara* plate number *74* as seen by individuals with *deuteranopia* or *protanopia* [58]

of assessment may be used, but the most widespread and commonly used is considered to be the *Ishihara* colourblindness test.

2.1.3.1 Ishihara test

The most well-known form of testing for colour blindness is using the pseudoisochromatic plates also known as the *Ishihara* plates [38]. This test can help to diagnose red/green deficiencies, meaning it is not suitable for *tritan-based* types.

The test relies on the fact that people with color vision deficiencies are unable to distinguish colours with similar hues [38]. The standard version consists of 38 plates of coloured dots. Plates contain Arabic numbers traced in colours, usually carefully chosen to be fall on the confusion spectrum for colour blind. See fig. 2.2 for a simulated visualisation of how a plate can be perceived by normal and affected vision.

There are different types of plates as well: only visible to those with normal vision (*vanishing*), with color vision deficiencies (*hidden digit*), those with a particular type of deficiency (*diagnostic*), plates appearing differently for normal and affected vision (*transformation*), and those seen by everyone (*demonstration*) [38].

The test has many variations (including smaller plate sets, versions for children, etc.) and is the *de facto* choice for diagnosis due to high accuracy and simplicity.

2.2 Existing Tools

There is a variety of different software tools in the scope of colour blindness. These includes both mobile applications and desktop software, mostly in four major categories: simulation, identification assessment and correction. The following section describes and compares some of these existing tools.

2.2.1 Simulation

Applications in this category are designed to help those with normal vision simulate what people with different colour vision deficiencies see like. Being the most popular category of the four, it has the broadest variety of applications available for all platforms.

One of the most popular tools, the *Chromatic Vision Simulator* [10] (see fig. 2.3), is available for iOS, Android and web and allows to upload video or images that can be recoloured to simulate select deficiency types. A similar experience can be achieved on desktop, using the *Color Oracle* [18] application available on *Windows*, *MacOS* and *Linux*. There are also innovative, gamified solutions, such as *Experience: Colorblindness* [28] that utilise the power of VR to enhance the experience.

Although very simple, not only do these apps help to support awareness of various color vision deficiencies, they are also a crucial tool for designers that utilise these to optimise their solutions for colour blind.

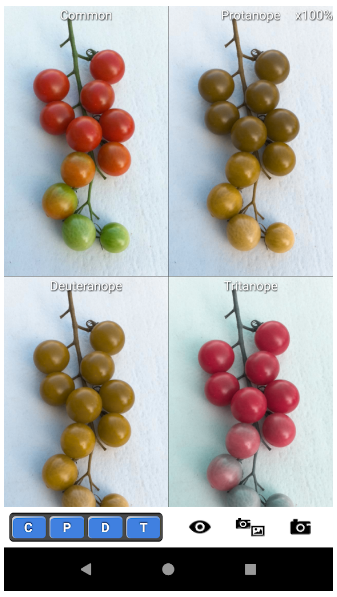


Figure 2.3: *Chromatic Vision Simulator* colour blindness simulation app on *Android* [10]

2.2.2 Assessment

This category of applications is meant to assist with determining if and what colour vision deficiencies the user might be suffering from.

Most of the applications in this category are based on the *Ishihara* test, such as the *EnChroma* [26] test available both as an app or on web. Some of the more interesting solutions in the field include the *Color Blind Test* [14] app on *Android* that has 3 types of colour vision deficiency tests, including *Ishihara*, built-in, and even games designed to entertain as well as test - *Kuku Kube* [40] being a great example on *Android*.

While they are not a replacement for a visit at the oculist, these applications can support early diagnosis and prevention of colour vision deficiencies, especially as some of them are acquired, rather than inherited, and in some cases may even be reversed.

2.2.3 Colour identification

Applications in this category can help those with colour vision deficiencies by identifying colours they might not otherwise correctly identify.

This category also sports many different solutions, however, most of them are not tailored for users who are colour blind, but rather designers who can also use these apps to point at an object and identify the colour they can later use in their work (e.g. *Cone* [22] on *iOS*).

The most novel and by far the most popular solution suited for affected-vision users is the *Color Blind Pal* [13] app available on *iOS*, *Android* and *MacOS*. It can retrieve selected colours and their names in multiple naming granularity levels both real-time and from an imported image. It also supports some features from other categories, such as correction and testing.

2.2.4 Correction

Correction applications can augment the user's view using specialised filters to attempt to modify the colour gamut for their specific colour vision deficiency type.

The selection of these types of applications is the smallest of the four categories. A nice example of such an app is *CB-Vision* [9] on *iOS* - it allows for the user to switch between preset filters for select types of colour blindness by saturating or replacing colours affected individuals are not able to distinguish otherwise (see fig. 2.4). Another solution is the *Visolve* [50] application, available on *Windows*, *MacOS* and *iOS* - it can also highlight a selected color (filtering) and apply stripes (hatching) that make nearby colours more distinguishable from one another.

These types of apps are extremely useful for the colour blind - they can help to support individual life with enabling simple tasks, such as identifying raw fruit, to be done when typically impossible with certain colour vision deficiencies.

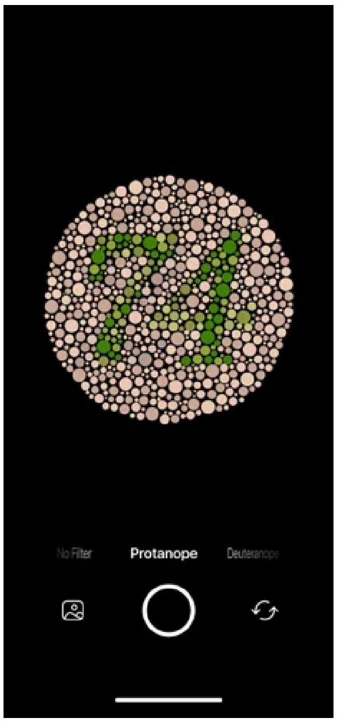


Figure 2.4: *CBVision* colour correction app on *iOS* [9]

2.3 Research Methods In Human-Computer Interaction

The following section introduces the *HCI* research methods used throughout the project and presents their main characteristics, advantages impacting method selection.

2.3.1 Data Collection Methods

2.3.1.1 Think aloud

Think aloud is a *concurrent* method of data collection that involves users completing pre-determined tasks related to the goals of the system in evaluation and verbalising their thoughts, actions and feelings at the same time [59]. There is also a less common version of *retrospective think aloud* protocol that asks the participants to complete the tasks in silence and reflect afterwards.

Since only the most necessary interaction and input from researcher is allowed, the resulting data can quite accurately depict the system usability aspects, instruct the researchers on how the system would actually be used and portray what user expectations might be, show clear frustration or difficulties in participant attempts with the tasks presented [59].

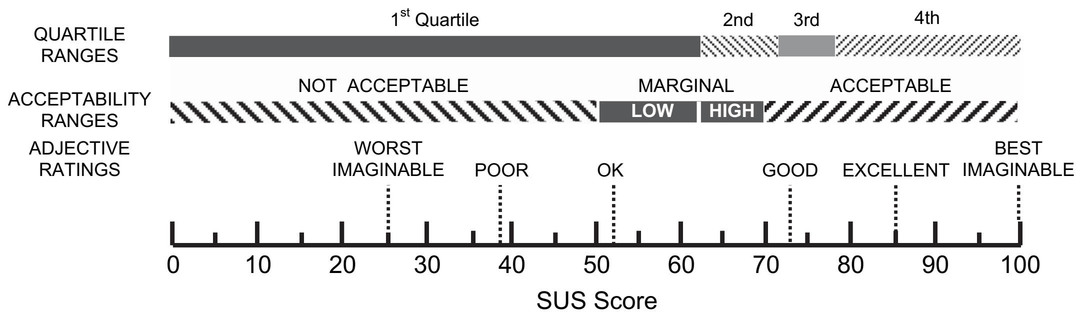


Figure 2.5: A visual guide for interpretation of *SUS* score ranges [54]

Coupled with audio and/or video recordings of the sessions, this method generates a lot of qualitative data for analysis, both as verbal and physical interaction with the system [55]. However, the method is quite expensive and time-consuming to run, so it typically is used with smaller participant counts than other data collections methods, such as questionnaires.

2.3.1.2 Interview

Interviews, one of *structured*, *semi-structured* or *unstructured*, are one of the most common *HCI* research methods typically used to collect participant opinions, perceptions and attitudes around a certain topic [59].

Being extremely versatile, *interviews* are often used in conjunction with other data collection methods as a supplement to gather additional data [59]. Depending on the type and the original research intent, they can vary from being very scripted to only focusing on a broader themes. The format of the interview largely impacts the effort of analysis later required - *structured interview* data is typically much easier to analyse, group or quantify.

2.3.1.3 System Usability Scale

The *System Usability Scale*, or *SUS* for short, is a questionnaire designed to evaluate and compare system usability using standardised methodology [54].

Participants are present a series of 10 standard statements relating to usability about the system under evaluation, and are asked to give a score from 1 (Strongly Disagree) to 5 (Strongly Agree) [54, 42]. The score is then computed as following: SUS = $(X+Y)\times 2.5$, where X is sum of all odd-numbered question answers-5 and Y is $25-$sum of all even-numbered question answers [42].

Once a score is computed, there are ways to present the obtained numerical value in other formats, such as adjectives representing the rating of usability, as well as ranges of acceptability of the system developed (see fig. 2.5) [54].

SUS is an effective tool to evaluate the system since it is cheap and easy to run, is very quick and widely accepted across the industry as the *de facto* choice for usability metrics [42].

2.3.2 Data Analysis Methods

2.3.2.1 Thematic analysis

Thematic analysis provides an effective and formulaic-like method of analysing qualitative data obtained through research [59]. It helps to categorise the main topics according to themes by *coding* the raw data according to multiple levels of specificity while updating, modifying and removing nodes obtained in the process.

Once the *coding* process is complete, the researcher is left with systematic data broken down into patterns which help to fully understand the underlying relationships and ideas involved [59].

Chapter 3

Methodology

This chapter provides an overview of the methodology applied throughout the duration of the project in section 3.1 and presents the stages of development involved in greater detail in the following sections.

3.1 Overview

Upon background review, project work was split into three major stages, each focusing on answering a subset of the research questions proposed in section 1.2, as outlined:

– **Stage 1** (Design): Stage focusing on requirements gathering and developing the design for a prototype solution, answering the following research questions:

  RQ1. How to design and implement a user-friendly, accessible and useful mobile application related to colourblindness? What features should it include? How could it improve over the existing solutions?

– **Stage 2** (Implementation): Stage focusing on implementing the proposed design obtained in Stage 1, answering the following research question:

  RQ1. How to design and implement a user-friendly, accessible and useful mobile application related to colourblindness? What features should it include? How could it improve over the existing solutions?

– **Stage 3** (Evaluation): Stage focusing on evaluating the prototype version of the application implemented in Stage 2 and improving based on feedback received, answering the following research questions:

  RQ2. What is the extent of impact, usefulness and usability of the implemented prototype application?

  RQ3. How to improve the proposed solution by utilising the feedback from user evaluation studies?

3.2 Stage 1 - Design

After completing the review of the background that included the overview of existing applications supporting colourblindness, a list of potential functionality was discussed with the dissertation supervisor. Final features (colour detection, simulation, recolouring, testing) were chosen by me and work moved to starting the design process.

Figma [29] was chosen as the platform for developing screens of the prototype application as I was familiar it from a previous project. Screen designs were developed iteratively and discussed with my supervisor during our weekly meetings. Screens and functionality were adjusted as result of some of these discussions and the final medium-fidelity interactive design prototype was derived.

Please refer to chapter 4 for the full details of the work described.

3.3 Stage 2 - Implementation

Once design was completed in Stage 1, work moved on to the implementation of the design. I decided I would be developing for the *Android* [3] operating system due to previous familiarity and cheaper and easier entry to the application distribution store.

Development process was done using *Android Studio* [24], with the application being built in *Kotlin* [39] utilising the *OpenCV* [2] vision library to assist working with the camera and frame processing necessary for the recolouring and simulation functionality of the application.

Once development process for the first iteration was complete, the application was submitted as an *open beta* to an app marketplace for evaluation.

Please refer to chapter 5 for the full details of the work described.

3.4 Stage 3 - Evaluation

Upon completion of the initial prototype version, I designed a usability evaluation study that was a modified *think aloud* version adapted for remote setting. Approval was seeked from the *Informatics Ethics Panel* [27].

After the study was given a green light, recruitment for participants was underway and in total 11 participants, 3 of which have colour vision deficiencies, agreed to participate. Subjects were asked to perform a series of tasks to discover the main features of the implemented prototype obtained in Stage 2.

The study outcomes were comments about functionality and design, usability evaluations, suggestions for future improvement. Resulting feedback was organised, analysed and prioritised based on the suggestion numbers. Select feedback was taken into account and implemented in a new iteration of the prototype application, which was then made public in *Google Play Store* [12].

Please refer to chapter 6 for the full details of the work described.

Chapter 4

Design

This chapter presents the methodology and the outcome of the design process of creating an interactive prototype[1] for the colourblindness themed mobile application.

4.1 Aims

The primary aims of process described in this chapter focus on tackling the following research question referring to a user-friendly and accessible prototype mobile application design creation:

RQ1. How to design and implement a user-friendly, accessible and useful mobile application related to colourblindness? What features should it include? How could it improve over the existing solutions?

Please refer to section 1.2 for the full list of research questions of the project.

4.2 Process overview

Design process was handled iteratively with consultations between myself and my supervisor between iterations.

The work has been done using the free *Figma* [29] software which allows to create interactive design mock-ups. *Unsplash* [6] and *Iconify* [37] plugins for *Figma* [29] were used to populate the screens with appropriately-licensed for non-commercial use images and icons, later to be used in the actual implementation as well.

An overview screenshot of the *Figma* [29] project screens, along with all interactive prototype connections, is shown in fig. 4.1.

After reviewing the background and existing solutions in the are, 4 potential main features have been chosen for the design and implementation stages: colour detection

[1]Available at `https://www.figma.com/proto/iK0r99QIofkdopHyu1eRoj/Colbi?node-id=562%3A0&scaling=scale-down`

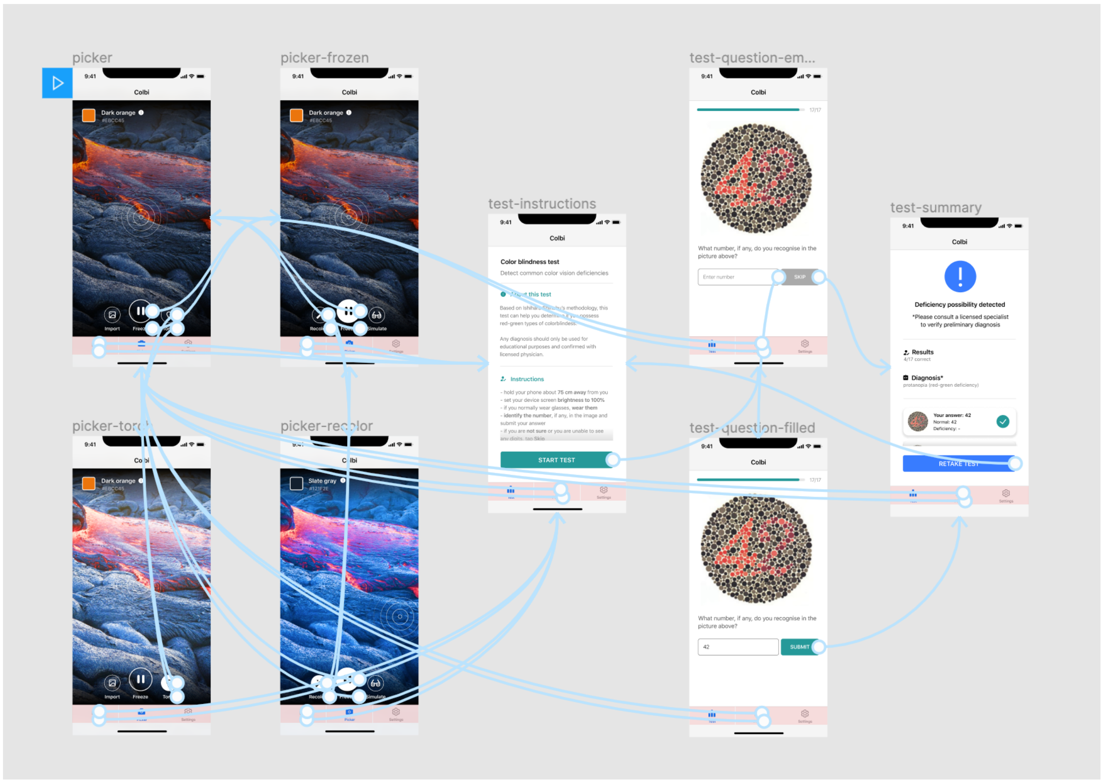


Figure 4.1: Screenshot of the *Figma* [29] project screen

(naming), colourblindness simulation, smart recolouring to increase discrimination of confusing colours for colourblind, and *CVD* testing.

The design choices for these features, along with appropriate justifications, are described in the sections following.

4.3 Design decisions

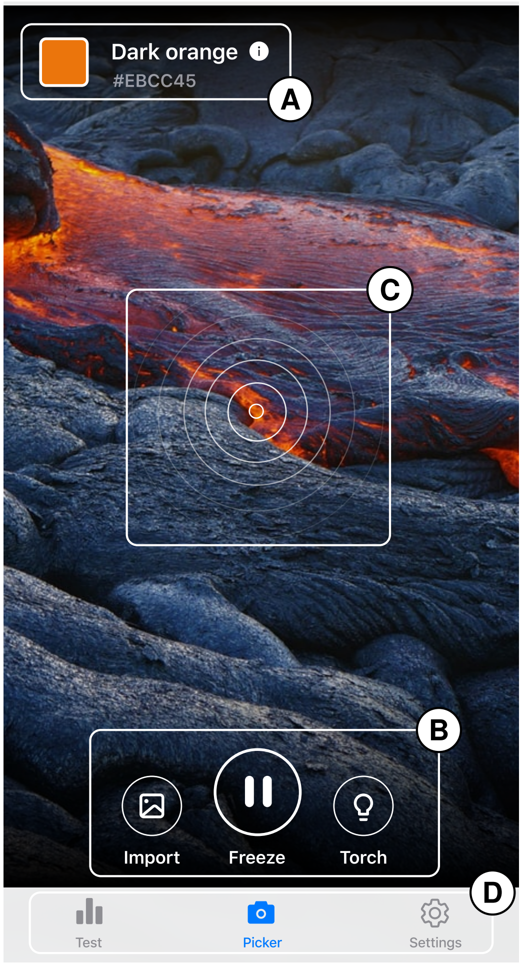
4.3.1 Core navigation

Core navigation is split into 3 main sections: *picker*, *test* and *settings*. Each of these can be easily switched by using the standard navigation bar visible on the bottom of the app at all times (see C in fig. 4.2).

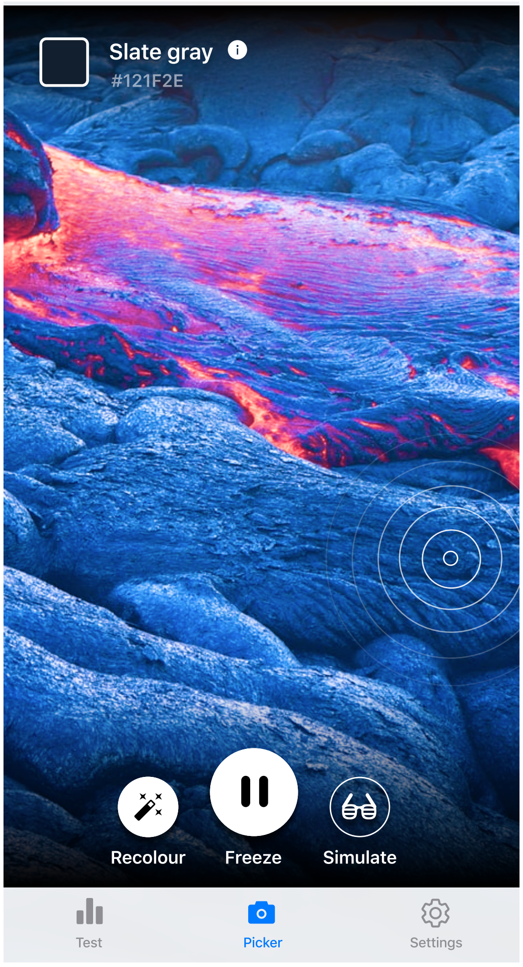
Picker, chosen as the entry point of the app as it would be the most frequently used section, contains the core features, in particular colour detection, simulation and recolouring, all in a single activity (more details in section 4.3.2). *Test* contains the *CVD* testing screens (see more in section 4.3.3), while *Settings* contains additional customisation options (more in section 4.3.4).

4.3.2 Picker

The picker screen (see fig. 4.2) hosts a near-full screen camera view experience with all interface options available in an overlay layer allowing to maximise the area of the camera. The following subsections discuss the user interface options available.



(a) Interface elements - colour information (A), target picker (B), action menu (C) and navigation bar (D).



(b) Interface after switching to the *Frozen Frame* mode, applying recolouring and moving the target picker.

Figure 4.2: *Picker* screen designs

4.3.2.1 Action menu

The action menu (see B in fig. 4.2) helps to control the main functionality of the *picker* view.

At any time, there are three buttons available, although these differ according to the state the *Picker* section is in. The idea behind this was to reduce the interface clutter by hiding unavailable/irrelevant options the user is not able utilise at a time. There are 3 different states: *Live Camera*, *Frozen Frame* and *Imported Image*.

Live Camera: default mode, has the *Import*, *Freeze* and *Torch* options (see fig. 4.3). The *Import* option, if pressed, opens the camera roll picker and would replace the live camera background with the imported image on success, as well as change the state of the section to *Imported Image*. The *Freeze* button allows to toggle between the *Live Camera* and *Frozen Frame* states that resemble the back camera view, and a captured frame of the camera at the time of the toggle until switched back. Finally, the *Torch* button allows to toggle the state of the on-device flashlight, if one is available (option greyed-out otherwise).

Frozen Frame: *Recolour*, *Freeze* and *Simulate* options available (see fig. 4.3). Tapping *Recolour* toggles between the frozen frame and the recoloured version of it that enables easier discrimination of confusing colours for colourblind. Similarly, *Simulate* toggles between the original frame and a recoloured version that simulates how it would be perceived by a colourblind person. As in the *Live Camera* mode, *Freeze* toggles between the two *Picker* states.

Imported Image: options similar to those in *Frozen Frame* (see fig. 4.3) mode enabling recolouring and simulation on an imported image instead, as well as having the *Close* button (in place of the *Freeze* toggle) that would return the user to the *Live Camera* mode by closing the imported image.

The buttons that allow to toggle between states indicate this by inverting the colours (see *Freeze* in fig. 4.3 for an example). All buttons in the action menu have labels underneath for clarity, although an option to disable these to allow more unobstructed camera/frame/image space will be available in *Settings*.

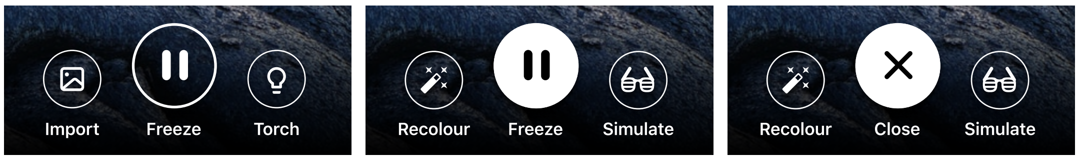


Figure 4.3: Action menu available actions in the *Live Camera*, *Frozen Frame* and *Imported Image* modes (left-to-right respectively)

4.3.2.2 Target picker

The *target picker* (see C in fig. 4.2) is a movable object, by default placed at the center of the screen, that helps to update location of the pixel chosen for colour detection (naming) feature, used for the colour information panel (see section 4.3.2.3). Although competing solutions already exist, some do not allow the flexibility of selecting any pixel on the screen so I wanted to have this in the designed product.

Although the pixel at the center of the *target picker* is chosen (the smallest semi-circle), more semi-circles surround the chosen pixel to increase the area and, consequently, visibility of the *picker*.

When dragging starts, the *picker* should slightly increase in size and invoke a small vibration on the device to provide feedback of the action started.

4.3.2.3 Colour information

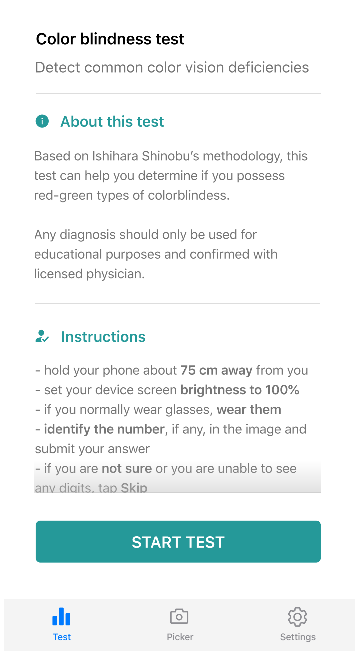
The *colour information* panel, highlighted as A in fig. 4.2, displays the real-time information about the colour currently targeted by the *target picker* (see section 4.3.2.2).

The colour detected is placed in the square, next to which is the colour name identified from a pre-defined set and the exact hexadecimal code of the colour for reference. There will be an option in the *Settings* to change the colour naming set used to customise to your liking, e.g. younger users might wish to use a set with less colour names which they might not be familiar with.

Since this feature is meant to support people with colourblindness, when recolouring is applied to the image, you are still presented the original colour name, prior to any recolouring. A pop-up notice about this is provided when a user taps the information `(i)` icon next to the colour name.

4.3.3 Test

The *test* section (see fig. 4.4) hosts a set of *Ishihara* [38] plates for diagnosing the most common *CVD* types. The following subsections describe each screen in the section in detail.



Color blindness test

Detect common color vision deficiencies

About this test

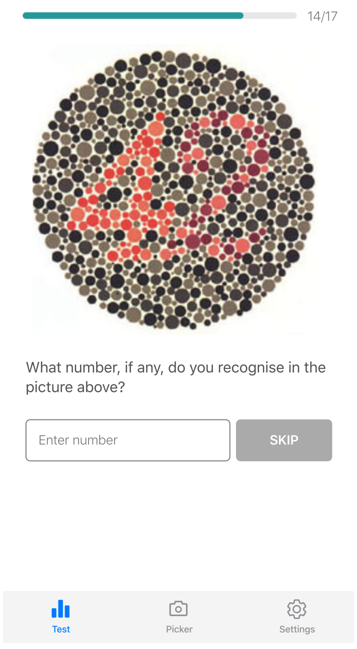
Based on Ishihara Shinobu's methodology, this test can help you determine if you possess red-green types of colorblindness.

Any diagnosis should only be used for educational purposes and confirmed with licensed physician.

Instructions

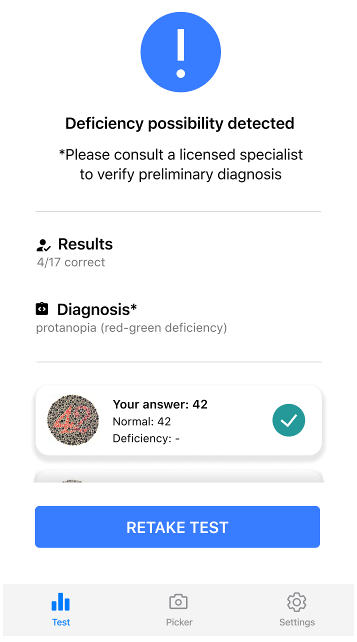
- hold your phone about **75 cm away** from you
- set your device screen **brightness to 100%**
- if you normally wear glasses, **wear them**
- **identify the number**, if any, in the image and submit your answer
- if you are **not sure** or you are unable to see any digits, tap **Skip**

(a) Instructions



What number, if any, do you recognise in the picture above?

(b) Question input



Deficiency possibility detected

*Please consult a licensed specialist to verify preliminary diagnosis

Results
4/17 correct

Diagnosis*
protanopia (red-green deficiency)

Your answer: 42
Normal: 42
Deficiency: -

(c) Result summary

Figure 4.4: *Test* section screen designs

4.3.3.1 Introduction & instructions

This screen (visualised in fig. 4.4a) hosts the necessary information to perform the tests, i.e. instructions, as well as some basic information about the nature of the test and a notice that any diagnosis should be confirmed with a licensed physician as there may be inaccuracies due to lightning, screen brightness and other factors.

The information area is scrollable and the only available button, *Start Test*, is always visible on-screen and leads to the first question in a *question screen*.

4.3.3.2 Question screen

The *question screen* (see fig. 4.4b) is generated and shown to the user multiple times until all test questions are completed.

At the top of the screen the user can see a progress bar indicating how many questions in the test they have left. Just below, the user finds the question plate, alongside with a quick instructional message and the input field.

By default, the only button visible is *Skip*, but if any input is provided to the field on the left, the button changes colour to green and the text then reads *Submit*. In either case, the button leads to a new question image with all input cleared, unless this was the last question, in which case it would open the *summary* screen.

4.3.3.3 Summary

The *summary* screen provides a quick overview of the results of a user when every test question is completed.

The screen shows the number of questions that was answered correctly, test result interpretation as possible diagnosis, and all the test questions along with, for each plate, the answer given, expected answers from people with and without colour vision deficiencies.

As with the *instructions* screen (see fig. 4.4a), the whole area is scrollable except for the button on the bottom, *Retake Test*, that is always on-screen and returns the user to the *Instructions* screen for another attempt.

4.3.4 Settings

No concrete design was created for the *Settings* screen in the interactive *Figma* [29] prototype as the *Android X Preference library* [4] was planned to use in advance to align the application with standard preferences design across the entire *Android* [3] ecosystem.

Planned customisation options in this screen included ability to change the filter for the recolouring and simulation features (e.g. switch between types of colourblindness simulated), the ability to select a different colour naming set for the colour detection feature, option to toggle hiding labels under the action menu in the *Picker* screen (see section 4.3.2) for users acquainted with the interface.

4.4 Summary

As part of the iterative design process, an interactive *Figma* [29] prototype application design[2] was created and made available.

[2]Available at `https://www.figma.com/proto/iK0r99QIofkdopHyu1eRoj/Colbi?node-id=562%3A0&scaling=scale-down`

The resulting design is minimal and clean, enables functionality of the 4 core main features planned. The result will be implemented as an *Android* [3] application and published on the *Google Play Store* [12]. Please refer to chapter 5 for details on the implementation process to follow.

Chapter 5

Implementation

This chapter outlines the process of implementing the interactive prototype colourblindness app design outlined in chapter 4.

5.1 Aims

The aims of the implementation process described in this chapter focus on tackling the following research question relating to an implementation of a previously described application design:

RQ1. How to design and implement a user-friendly, accessible and useful mobile application related to colourblindness? What features should it include? How could it improve over the existing solutions?

Please refer to section section 1.2 for the full list of research questions of the project.

5.2 Process overview

5.2.1 Development decisions

After completing my design prototype outlined in chapter 4, I made the decision of implementing the application for *Android* [3] using *Kotlin* [39] as the chosen programming language, coded in *Android Studio* [24].

The factors affecting this decision included prior familiarity with *Android* [3] development, specifically in *Kotlin* [39], having pre-existing access to an *Android* test device, the OS having broader device support and an active open-source community, and potential entry to the *Google Play Store* [12] being easier and more affordable than that of *iOS App Store* [5].

All code was version-controlled on a private repository using *Github* [1].

5.2.2 Libraries used

Besides the default material design libraries for *Android* [3], two additional libraries were used to aid the first iteration of development:

- *OpenCV* [2] library used for accessing the camera and manipulating frames (more details in sections 5.3.1 and 5.3.4),
- *Google's GSON* [32] library to read colour naming information saved in `JSON` files (see section 5.3.3).

Please note that more libraries were added as part of the second iteration after evaluation, see section 6.8 for more details.

5.2.3 User interface

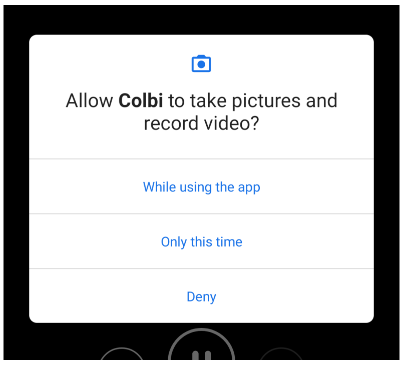
User design implemented closely follows that introduced in chapter 4 with only minor modifications if considered more user-friendly at the time of the implementation. Design is defined in *XML* files using standard *Android* [3] design components (`View`, `Button`, `RecyclerView`, etc.), unless specified otherwise in the following sections.

5.3 Core functionality

5.3.1 Camera access & configuration

As mentioned in section 5.2.2, I chose *OpenCV* [2] as the vision library for working with the camera and manipulating images. Unfortunately, the set up to include it in a blank *Android* [3] project proved to be more challenging than anticipated. Because the library is a cross-platform solution, the *Android* [3] version was very poorly documented and the guides were outdated. After days of trying to get the library set up, I found a project on Github [51] that provided a template with the library plugged in to a blank *Android* [3] project and used that to start the development process.

The next challenge was to set up camera permissions, since starting with *Android 6.0* [3] these must be requested during run-time. When the app is launched, since it immediately goes into the full-screen camera mode, it simply checks whether access is already granted (e.g. from prior use) and prompts a native request dialog if that is not the case (see fig. 5.1). The user is allowed to use the app without granting this as well, if they, e.g., simply want to use imported images or test for colour vision deficiencies only.


Allow **Colbi** to take pictures and record video?

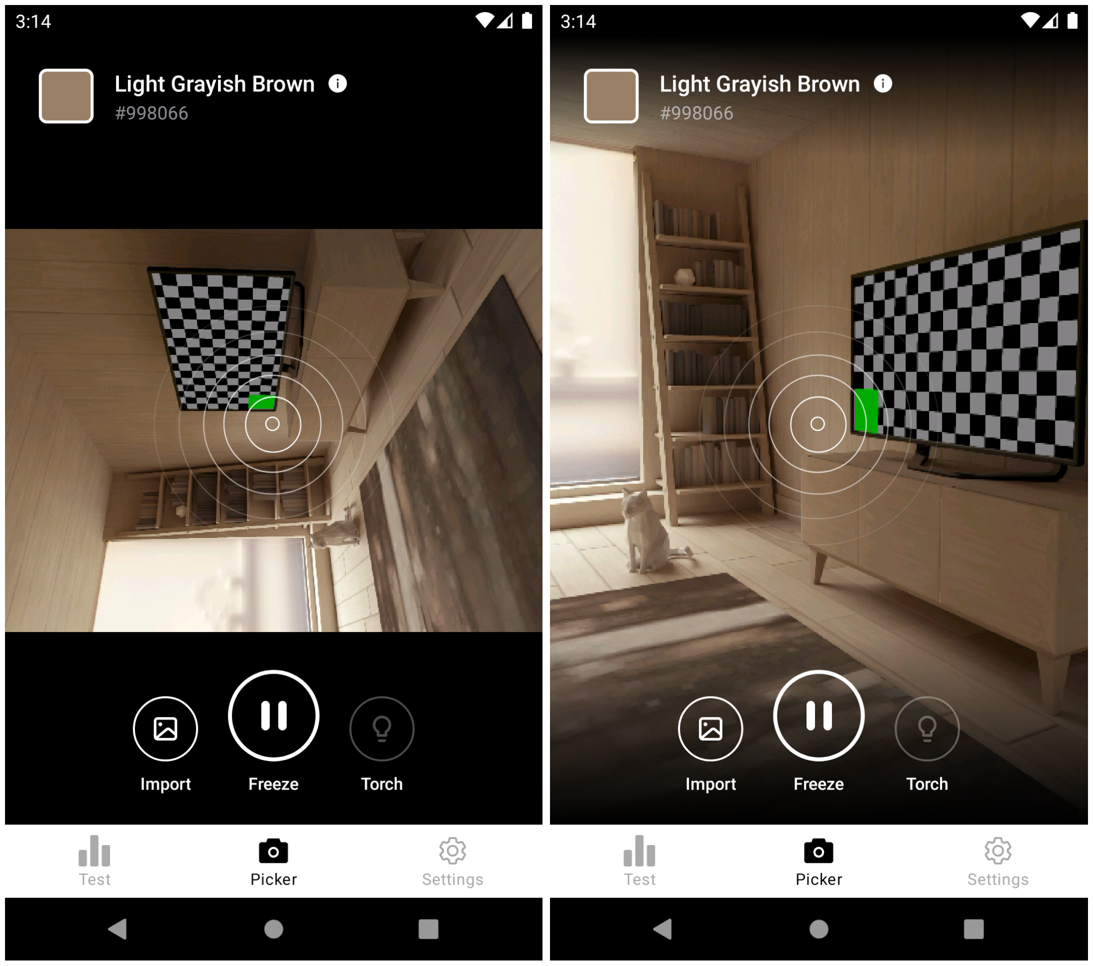
While using the app

Only this time

Deny

Figure 5.1: Camera permission request on application launch

Finally, by default, the camera module provided by *OpenCV* [2] displays the feed rotated by $-90°$ and fit into the width of the screen. Surprisingly, the library itself does not provide an easy way of orienting the camera view and scaling it to be fullscreen on the *Android* [3] fork (the camera feed might not be the same ratio and/or resolution as the screen so a simple rotation is not enough). This meant I had to extend and replace the `JavaCameraView` class providing the view, update (override) the library code in `CameraBridgeViewBase`, following a guide from *Mike Heavers* [52], to update the matrix drawn on the screen until the image was rotated, scaled to fill the width/height of the screen (whichever is larger) and centered to cover the entire application view (see fig. 5.2 for a comparison).



(a) Before (b) After

Figure 5.2: Default and modified *OpenCV* [2] camera view comparison

5.3.2 Image import

To retrieve an imported image, the native system gallery is invoked when the *Import* option is tapped. The image chosen comes back to the activity using the activity's `onRequestPermissionsResult` method as raw data which I process, turn off the camera view and replace it with an `ImageView` with the image selected.

Obviously, working with existing pictures might not be as convenient as having the live camera access to the subject when, e.g., you can simply get closer to it to get the fine-grained control of a pixel to you want to use for colour detection. Hence

I wanted to implement zoom for imported images. To achieve this, I extended the built-in `ImageView` class as `ZoomableImageView` using code adapted from code in *StackOverflow* from the user *Komal12* [33]. This required similar matrix manipulation to that described in section 5.3.1 to enable handy zooming, panning and dragging using familiar finger gestures like pinching.

5.3.3 Colour detection

5.3.3.1 Naming dictionaries

To implement the colour detection feature, I was first concerned with sourcing the colour name dictionaries. I found a useful page [17] hosted on MIT's servers with a list of different dictionaries and their comparisons. I selected 4 of them, namely *Crayola*, *CSS*, *HTML4* and *NBS-ISCC* and wrote a small *Python* script to convert the given files of names and associated colour hexadecimal codes to `JSON` object files for use in the app. The first 3 lists were initially chosen for the implementation but I soon realised some of the colour names were quite niche (*Cerise*, *Inchworm*, etc.) and would not be that helpful for some users, especially younger ones. Hence I looked for another set of colour names that would be more appropriate for the domain and came across *NBS-ISCC* which contains colour names that are based on several primary colours with very descriptive names (*Vivid Pink*, *Strong Pink*, *Deep Pink*, etc.) and this was chosen as the default option for the feature.

5.3.3.2 Distance measure

I also needed to consider how we could match which colour name should be shown as the dictionaries do not cover every possible value, hence we needed a definition of *closeness* for colour. The initial implementation simply calculated the *Euclidian* distance between the 3 `RGB` components. This works well, but, upon testing, I found some of the colour names assigned to be quite unexpected, at least to my eye. I looked more into this and found that the problem itself is extremely complex and there are many different colour distance measures. In fact, some measures take into account the perception of human eye not being linear [20]. One of such examples, a *"low-cost approximation"* based on a weighted *Euclidian* distance and dependent the saturation of the red colour component, is known as the *redmean* distance [20]. It was chosen as the distance measure for my implementation and is calculated as follows:

$$\Delta C = \sqrt{\left(2 + \frac{\frac{R_1+R_2}{2}}{256}\right) \times \Delta R^2 + 4 \times \Delta G^2 + \left(2 + \frac{255 - \frac{R_1+R_2}{2}}{256}\right) \times \Delta B^2},$$

where R, G and B reflect the values of the `RGB` components of the two colours to be compared (e.g. denoted as R_1 for red value of first colour).

5.3.3.3 Target picker

To implement the target picker, a `MotionEvent` listener was added to detect and apply moves of the target. Whenever the target is moved, a global variable holding the current chosen pixel coordinates are updated to the center of the target picker position, and the value is used to determine the colour of the pixel at the precise location. This update process itself proved to be quite challenging for the live camera view because all image matrix manipulations described in section 5.3.1 had to be reversed to map the touch location on the screen to the location of the pixel in the camera capture feed, accounting for the raw touch input coordinates being misplaced due to status, navigation bars and other interface elements provided by the *OS*. Similar calculations, as well as an additional listener for the `ZoomableImageView` class, were required for imported images, especially if any zooming or panning was applied as this way the location of the pixel chosen can change without the target picker itself being moved.

5.3.3.4 Implementation overview

Once dictionaries sourced as `JSON` files were placed along with other assets of the app and target pixel was comparable, it was time to implement the feature itself. The implementation is as follows:

1. On app launch, the selected `JSON` file is opened and the colour mapping (name to hexadecimal value) is read into memory.
2. When the colour detection is invoked (which happens every frame for the live camera feed, when an imported image is zoomed or panned, or when the target picker is moved), the `RGB` values of pixel targeted are converted into hexadecimal and the distance is compared (see section 5.3.3.2) between it and all the colour dictionary values to find the closest matching named colour.
3. The colour is shown on the colour information panel with the matching colour name and hexadecimal value of the original colour for reference.

5.3.4 Recolouring & simulation

As per design introduced in chapter 4, the app should support recolouring and simulation features on both frozen frames in camera view as well as imported images from the camera roll. The implementation of this involved getting a frame returned by the camera module of *OpenCV* [2] via the `onCameraFrame` method, or an image imported, and applying either the *daltonization* or *simulation* algorithm for recolouring or simulation respectively. The following subsections introduce the two algorithm implementations.

5.3.4.1 Daltonization

To enable easier discrimination of confusing colours for colourblind users, *LMS daltonization* [57, 23, 56] algorithm, one of the most popular and efficient recolouring options, was adapted and implemented.

The modifications to the original algorithm aim to utilise the matrix operations offered by the `Core` module of *OpenCV* [2] in effort to avoid the expensive looping through each pixel individually, and to take into account the presence of the alpha (`A`) channel in frames returned by *OpenCV* [2].

The summarised algorithm to *daltonize* an input frame or an image is as follows:

1. Get the `RGBA` $M \times N$ matrix of a frame or an image.
2. Reshape (flattening) the matrix to a 1D vector-like shape ($(M * N) \times 1$).
3. Transform the `RGBA` value matrix into *LMS* colour space by multiplying it with the *RGB*2*LMS* (eq. (B.1)) matrix [57].
4. Obtain the *LMS* simulation related to the *LMS* cone affected by multiplying (transforming) the result with one of eq. (B.5), eq. (B.4) or eq. (B.6) for *protanopia*, *deuteranopia* or *tritanopia* respectively [57].
5. Transform the result back to the `RGB` colour space using the inverse (eq. (B.2)) of the *RGB2LMS* matrix [57].
6. Calculate the difference between the original `RGBA` vector (step 1) and result in step 5 and multiply it with the *ERR2MOD* matrix (eq. (B.3)) - the result ”represents the information lost” when simulation was applied [57].
7. Add the result obtained step 6 to the original `RGBA` vector (step 1) to obtain the recoloured vector [57].
8. Reshape and return the recoloured vector as an `RGBA` $M \times N$ matrix.

5.3.4.2 Simulation

In comparison to the *daltonization* algorithm, the *simulation* algorithm was relatively simple and, similarly, adapted from [8] as follows:

1. Get the `RGBA` $M \times N$ matrix of a frame or an image.
2. Transform the vector to the simulated colour space by multiplying it with one of eq. (B.7), eq. (B.8), eq. (B.9), eq. (B.10), eq. (B.11), eq. (B.12), eq. (B.13) or eq. (B.14) for *protanopia*, *protanomaly*, *deuteranopia*, *deuteranomaly*, *tritanopia*, *tritanomaly*, *achromatopsia* or *achromatomaly* respectively [8].
3. Reshape and return the colourblindness-simulated vector as an `RGBA` $M \times N$ matrix.

5.3.5 Testing

As per design described in chapter 4, there are 3 main types of screens implemented for the testing capabilities: the instructions, the test question and the summary screens.

The instruction screen was simple to implement as it hosts fixed information and a single button that takes the user to the question fragment.

On launch, the test fragment generates a random order of 17 (skipped optional coloured-lines plates to shorten the time required) test *Ishihara* [38] plates from the *24 Plate Edition* [62], 2 of which are *diagnostic* plates allowing to determine the type of deficiency instead of presence (observed differently by *deutan*- and *protan*-related deficiencies). These are iterated in the order aforementioned in the test question screen before all the questions are answered.

When the last test plate is answered, the answers given are bundled and passed to the summary screen. There, using the source material [62] of the plates, an interpretation of the result (preliminary diagnosis) is computed based on the number of plates correctly answered by the user and shown on the screen, along with a list of all questions and answers using a scrollable `RecyclerView` [3]. The logic of the diagnosis shown is as follows [62]:

1. If 13 or more plates are correctly answered - *normal*.
2. Between 10 and 13 correctly answered correctly is extremely rare and requires additional testing - *indeterminate*.
3. If 1 or 0 plates are answered correctly - *achromatopsia*.
4. Otherwise, a *deficiency* type, based on the 2 *diagnostic* plate answers, is determined as follows:
   (a) If both plates are answered as observed by *protan* or *deutan*, *protanopia* or *deuteranopia* is shown respectively.
   (b) If one plate was answered correctly and the other as observed by *protan* or *deutan*, *protanomaly* or *deuteranomaly* is shown respectively.
   (c) Otherwise, user is only notified of an indeterminate *anomaly* as the result.

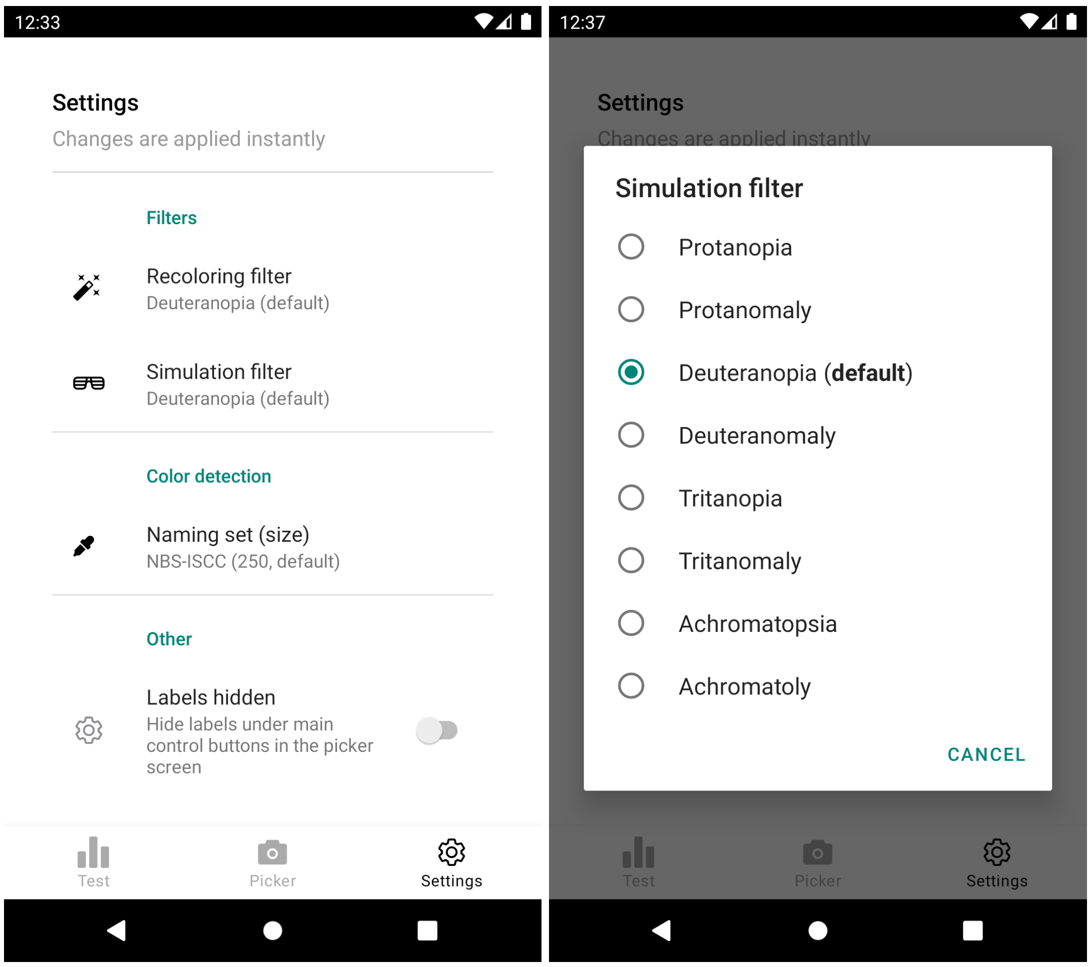
5.3.6 Settings

As mentioned in chapter 4, the *Settings* screen (see fig. 5.3) would be implemented using the *Android X Preference library* [4]. The preference included are those as described in the design, such as switching between the simulation filters (see fig. 5.3b). Settings are saved as `SharedPreferences` under app data in the device so user changes are preserved across app launches and updates.

5.4 Other

Along with the core functionality, some additional features were implemented to improve the usability of the application. These included small vibration and animation feedback when dragging the target picker, subtle animations when interacting with buttons in the interface, change of layout colours in the test summary screen depending on the resulting diagnosis.

Since I planned to continue the application development further, I added a plugin that checks for updates to the application on the *Google Play Store* [12] upon launching the


12:33

Settings
Changes are applied instantly

Filters

Recoloring filter
Deuteranopia (default)

Simulation filter
Deuteranopia (default)

Color detection

Naming set (size)
NBS-ISCC (250, default)

Other

Labels hidden
Hide labels under main control buttons in the picker screen

12:37

Settings
Changes are applied instantly

Simulation filter

- ( ) Protanopia
- ( ) Protanomaly
- [✓ Deuteranopia (**default**)]
- ( ) Deuteranomaly
- ( ) Tritanopia
- ( ) Tritanomaly
- ( ) Achromatopsia
- ( ) Achromatoly

(a) All settings (b) Simulation filter picker

Figure 5.3: *Settings* implementation screens

app. However, this, as any app that accesses the internet, introduced a notice on the store listing that the app has ”*full network access*” which was negatively received by some test users trying out the app who expected it to be completely offline and thus this feature was removed from the final implementation for the time being.

5.5 Summary

A fully functional app, based on the design introduced in chapter 4, was implemented in *Kotlin* [39] for the *Android* [3] mobile operating system. It contains all the core functionality planned, namely colour detection, *CVD* testing, recolouring and simulations, as well as some small tweaks to increase user-friendliness.

The application was published on the *Google Play Store* [12] as an *open beta* in preparation for the evaluation studies, as presented in the following chapter.

Chapter 6

Evaluation

This chapter presents the methodology and the results of the study run to evaluate the design, usability and usefulness of the implemented app, as described in chapter 5.

The study described in the chapter was certified according to the Informatics Research Ethics Process, RT number 2019/89367.

6.1 Aims

The aim of the study outlined in this chapter was to tackle the following research questions related to evaluation and improvement of the prototype described in chapter 5:

RQ2. What is the extent of impact, usefulness and usability of the implemented prototype application?

RQ3. How to improve the proposed solution by utilising the feedback from user evaluation studies?

Please refer to section 1.2 for the full list of research questions of the project.

6.2 Data Collection Methods

Data for the study was collected using a remote adaptation of the *think aloud* (see appendix A.5) protocol run via *Microsoft Teams* [44], followed-up by a short interview (see section 2.3.1.2). The *think aloud* protocol is often used to discover usability issues and to provide a better understanding of the system usage, provides a lot of meaningful qualitative data even when used with smaller participant numbers.

As well as that, to aid the upcoming data analysis, meetings were audio and video recorded. Note that for video, only the on-screen device interactions (shared device screen) of the participant were recorded.

Finally, to provide a summative evaluation component, the *System Usability Scale* (see section 2.3.1.3) was additionally used.

Participant	**Colour-vision deficiency**	**Device used**	**Recruitment method**
P1	Suspected *(red-green)*	Samsung Galaxy S10+	Mailing list
P2	None	Redmi K30 Pro	Mailing list
P3	None	Google Pixel 2	Direct
P4	None	Redmi Note 7	Direct
P5	None	Huawei Y7 2018	Direct
P6	None	Redmi Note 6	Direct
P7	Suspected *(red-green)*	Xiaomi MIX 2	Mailing list
P8	Diagnosed *(red-green)*	Pixel 4 *(emulated)*	Direct
P9	None	Samsung J6+	Direct
P10	None	Motorola G8 Power	Mailing list
P11	None	Huawei Mate 9	Mailing list

Table 6.1: Breakdown of participants involved in the study

6.3 Participants

Although the target group of potential users of the app is colourblind people, this particular group would be challenging to recruit for the purposes of this project. Because of this, no specific criteria has been chosen for recruitment, although it was preferred that participants had an *Android* [3] device of their own they could use for the study.

Advertisement methods included direct contact and the use of a *School of Informatics* Undergraduate mailing list (`ug-students`) [41] to reach a wider audience with hopes of recruiting at least some participants possessing colour vision deficiencies.

In total, 11 participants were involved, 3 of which had claimed to have some degree of colourblindness. The breakdown of different participants, their associated recruitment method, presence of colour-vision deficiencies and devices used in the study is presented in table 6.1.

6.4 Materials

Based on the examples given in the *School of Informatics* intranet's ethics procedure pages [27], *Participant Information Sheet* (*PIS* - see appendix A.2) and *Participant Consent Form* (*PCF* - see appendix A.1) documents have been prepared and submitted for approval to the *Informatics Ethics Panel*.

As well as this, a list of tasks (see appendix A.3) were designed for participants to try out the different functionality (colour naming, image import, simulation, recolouring, *CVD* testing) of the system while not giving guidance on how to use or access these features to discover potential usability issues.

Finally, to test the app itself, it was submitted as an *open beta* to the *Google Play Store* [12]. This was done purposefully before the study commenced to allow quick and easy installation using an official source rather than *side-loading* an `APK` file which would require to enable loading apps from unknown sources of the target device and would create additional steps for participants.

6.5 Protocol

Participants were thanked for agreeing to partake in the study and sent a copy of the *PIS* (appendix A.2) and *PCF* (appendix A.1) documents to get acquainted with the study and have an opportunity to ask any questions. If there were no further questions, they were asked to select a time-slot for a study using a *Doodle* [30] poll.

Once a time-slot was picked, participants were sent a *Microsoft Teams* [44] invite for the appropriate time and given some instructions to prepare for the study: to sign an electronic version of the *PCF* on *Microsoft Forms* [43], to pre-download the *Colbi* app from *Google Play Store* [12] and install the *Microsoft Teams* [44] client on their *Android* [3] device.

Participants without an *Android* [3] device were allowed to use an emulator hosted on the researcher's computer with the help on a *Team Viewer* [47] client.

At the agreed time, individual participants were asked to join the meeting on their mobile device that would be used in the study. They were briefed on how the *think aloud* (see section 2.3.1.1) protocol works using a script (see appendix A.5) adapted from the *Human-Computer Interaction* [48] course at the *School of Informatics*.

Participants were asked to share their device screen during the call and were notified the recording of the meeting is about to start. Before starting the tasks, they were also asked about presence of colour vision deficiencies and what is the model of the device they will be doing the study with (for debugging purposes).

Participants then carried out the tasks with close-to-none interaction with the researcher, unless they were stuck and unable to progress further. After they were done, the researcher answered any questions that arose during the session and asked for clarifications, as well as to answer some additional, more general follow-up questions (see appendix A.4) about the experience with the app, including rating the usability of the system using *SUS* (see section 2.3.1.3).

Afterwards, the recording would be stopped, the participant was asked whether they had any further comments without being recorded and were thanked for their time getting involved.

6.6 Data Analysis

As mentioned in section 6.2, recordings of the session were produced for each participant. These were carefully analysed (inspecting both the video of the on-screen interactions with the app, as well as noting down audible cues) and the problems discovered were *coded* using *thematic-analysis* (see section 2.3.2.1).

A list of issues compiled, sub-categorised into 5 themes (*testing*, *simulation/recolouring*, *colour detection*, *settings* and *bugs*), along with which participants experienced the issue and the numbers associated.

A similar approach of collating and quantifying similarly-themed responses was taken for the follow-up qualitative question responses. These were categorised according to each of the 3 questions asked (see appendix A.4) respectively (*liked*, *disliked*, *suggestions*).

Individual *SUS* (see section 2.3.1.3) scores were calculated and provided a meaningful interpretation for each participant using the steps outlined in the background chapter.

6.7 Results

As outlined in section 6.6, the issues faced and comments mentioned during the study were collated and categorised. The following subsections present summarised results (common to > 1 participant) of these categories separately. The full result table can be found in appendix A.6.

6.7.1 Issues raised

6.7.1.1 Testing

Most notable issues included lack of clear guidance whether there is an option to skip if a number in the test plate cannot be recognised (4/11), as well as UI issues on devices with smaller screens causing some parts of the interface not to be fully visible when the numpad for input is displayed on-screen (5/11).

> *"It's a bit annoying to have to like scroll down and scroll up to be able to see the number, but I feel like that might be unavoidable..."* - P3

6.7.1.2 Simulation / recolouring

Probably the most common issue that affected a very high number of participants (9/11) was the task on how to enable the recolouring / simulation feature for the camera view. Judging by some of the comments, this could have been partly caused due to the wording of the task, perhaps implying that users could apply this to the live camera view, rather then a captured (*frozen*) frame of the camera view. Several users went into the *Settings* activity to try and enable this.

> *"Maybe in the settings if there is some kind of a note that says that these settings [can only be] applied when you have frozen the picture?"* - P11

6.7.1.3 Colour detection

Colour detection feature was mostly successfully utilised, although there were participants (4/11) that mentioned the discovery of the target picker was limited, especially on lighter backgrounds. 3 participants skipped the task because they misunderstood the wording, while 2 participants also mentioned they expected the pixel to be detected to be tapped, rather than to drag the picker to.

> *"I think in the top-left it's [showing] greenish black because the top-right corner is black right now? I'm not sure... [taps the target picker] Oh, okay! I just noticed the circle thing in the middle."* - P1

6.7.1.4 Bugs

The usability study helped to uncover a lot of bugs with the app, some of which were very specific to the devices used in the study. These included the initial colour before the target picker is moved being recognised incorrectly (11/11), the camera view crashing after a large (in resolution) image is imported (2/11), images being imported sideways (5/11) and the colour vision deficiency showing an inconclusive result when it was not supposed to (5/11).

6.7.2 What did you like about the app?

There were several features of the application that participants mentioned they particularly liked. 3 participants highlighted that is is useful to be able to import pre-existing images from your camera roll instead of using the camera view; 3 participants said they liked having the target picker and the flexibility to precisely pick any pixel using the colour detection feature; 2 participants also appreciated the consistency in the UI icons in the *picker* and *settings* activities; majority of participants (7/11) liked the overall interface, including ease of use, simplicity; finally, 2 users mentioned liked having the recolouring functionality and 2 said they appreciate the built-in colour vision deficiency testing capabilities.

> *"I liked how clean the user interface was, <...> I liked the icons that clearly map to the features that we're trying to change. <...> I keep saying this word - it's just intuitive."* - P5

> *"I liked the fact that there is a built-in test and like a way to sort of import photos and apply a certain filter to them."* - P8

6.7.3 What did you dislike about the app?

Most participants (7/11) mentioned there was no particular area of the app they that disliked. Some participants (2/11) were annoyed about the crashes within the app, while others (2/11) also mentioned some performance-related subtleties (phone getting hot, loss of frames).

> *"When I touch Freeze and I touch Recolour it takes some time to generate the recoloured picture. But I guess that is reasonable because it's a live algorithm."* - P2

6.7.4 Suggested improvements

The most popular suggestion (9/11) included having some sort of guide or tutorial upon launching the app for the first time as some of the features needed some additional guidance. Other suggestions (2/11) included changing the *Skip* button colour from gray as it implies not being clickable, improving the visibility of the target picker (1/11), reducing the amount of text in the introduction screen (1/11) for the test and many more (see full list in appendix A.6).

Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	**Mean**
SUS score	95	75	82.5	82.5	90	82.5	82.5	82.5	85	80	77.5	**83.2**

Table 6.2: Breakdown of SUS scores of each participant

> *"Like the first time you load the app you can have some instructions that say how you can recolour, or that you can simulate how [people with colour vision deficiencies] see by freezing the live camera view."* - P6

Not all suggestions from participants were considered for implementation. For example, there was a request to confirm with the user (perhaps using a dialog) that they indeed wanted to *Skip* a test question, but colourblind users would only be able to identify couple of plates, meaning this would create a lot of additional confirmations and could be annoying and possibly deterring.

6.7.5 Usability scores

As mentioned in section 6.2, *SUS* (see section 2.3.1.3) scores were computed based on the answers from each participant. The full breakdown of individual participant results is displayed in table 6.2, while statistical analysis (`min`, `max`, `mode`, `mean`, `standard deviation`) of each survey question is summarised in table 6.3.

The lowest mean score (2.8) per question was given to question 1 with most participants immediately adding that their score is indicative of the fact that they were not the intended target audience for the app. The questions with the best mean scores (4.5) were 3 and 7.

Overall, the average usability score for all participants was `83.2`, which can be interpreted as `Great` (see section 2.3.1.3). Participants in the study indicated they were mostly confident using the system, with many indicating only slight improvements are needed to help ease the initial learning curve of using the application (see section 6.7.4).

6.8 Improvements implemented

After the evaluation study was completed, additional effort was put into fixing the bugs discovered and implement some of the suggestions identified by participants involved. If there were device-specific bugs that were impossible to replicate on an emulator, participants who faced these were asked to confirm later the bugs were fixed.

The following list contains the changes made in result of the study is sorted by the relevant number of participants the feature was suggested/affected by:

1. Bug of initial colour target being detected incorrectly, as well as colour detection bugs related to dark-mode, fixed (11/11).
2. First-use on-boarding screens added explaining main functionality usage using the *Ahoy! Onboarding* library [11], see fig. 6.3 (11/11).

No.	**Question**	**Min**	**Max**	**Std**	**Mode**	**Mean**
1.	I think that I would like to use this system frequently.	1	5	1.1	3	**2.8**
2.	I found the system unnecessarily complex.	1	2	0.5	1	**1.5**
3.	I thought the system was easy to use.	4	5	0.5	5	**4.5**
4.	I think that I would need the support of a technical person to be able to use this system.	1	2	0.3	1	**1.1**
5.	I found the various functions in this system were well integrated.	3	5	0.8	5	**4.1**
6.	I thought there was too much inconsistency in this system.	1	3	0.6	1	**1.2**
7.	I would imagine that most people would learn to use this system very quickly.	3	5	0.7	5	**4.5**
8.	I found the system very cumbersome to use.	1	2	0.5	1	**1.3**
9.	I felt very confident using the system.	2	5	0.7	4	**3.9**
10.	I needed to learn a lot of things before I could get going with this system.	1	3	0.7	1	**1.6**

Table 6.3: Breakdown of statistical analysis of scores for each SUS question

3. Test question image and answer input now fully visible on any screen size without scrolling due to applied image scaling and removal of the navigation bar above the numpad during input using the *KeyboardVisibilityEvent* library [53], see fig. 6.2 (5/11).
4. Bug of images imported sideways on some devices fixed (5/11).
5. Bug of unexpected inconclusive test result fixed (5/11).
6. Testing instructions updated with clearer guidance of action to take if a number cannot be recognised, see fig. 6.2 (4/11).
7. Target picker visibility increased significantly for lighter backgrounds by adding dark shadows and increasing the stroke of semi-circles, see fig. 6.1 (4/11).
8. Crashes due to context-switching and importing large images fixed (3/11).
9. Test summary screen now clearly scrollable with added scroll bars (3/11).
10. Formatting of test results broken on smaller-width screens fixed and clarified when a question was skipped (3/11).
11. *Skip* button no longer gray to avoid confusion regarding interaction, see fig. 6.2 (2/11).
12. *About this test* section now collapsed by default to increase visibility of the testing instructions section using the *ExpandableLayout* library [7] (2/11).

13. Additional instructions, as well as an option to invoke the on-boarding screens, added on the *Settings* screen (2/11).
14. The demonstration plate is now shown as the first question, all other questions are still in random order (1/11).

After implementation was complete, the app was made public on *Google Play Store*[1].

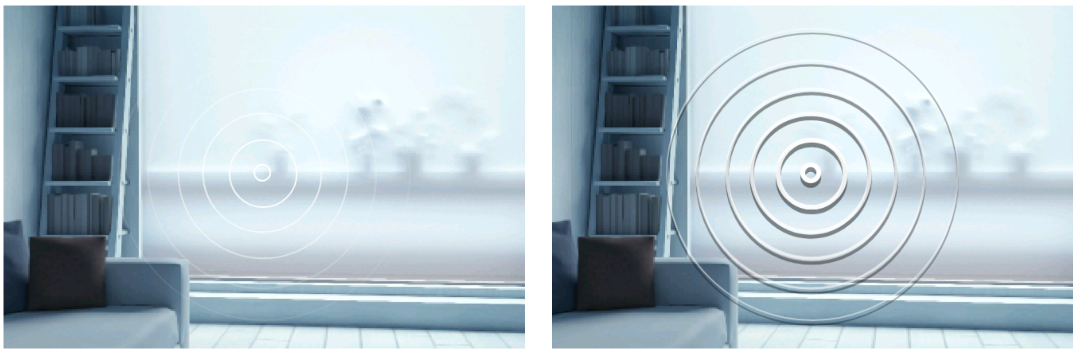
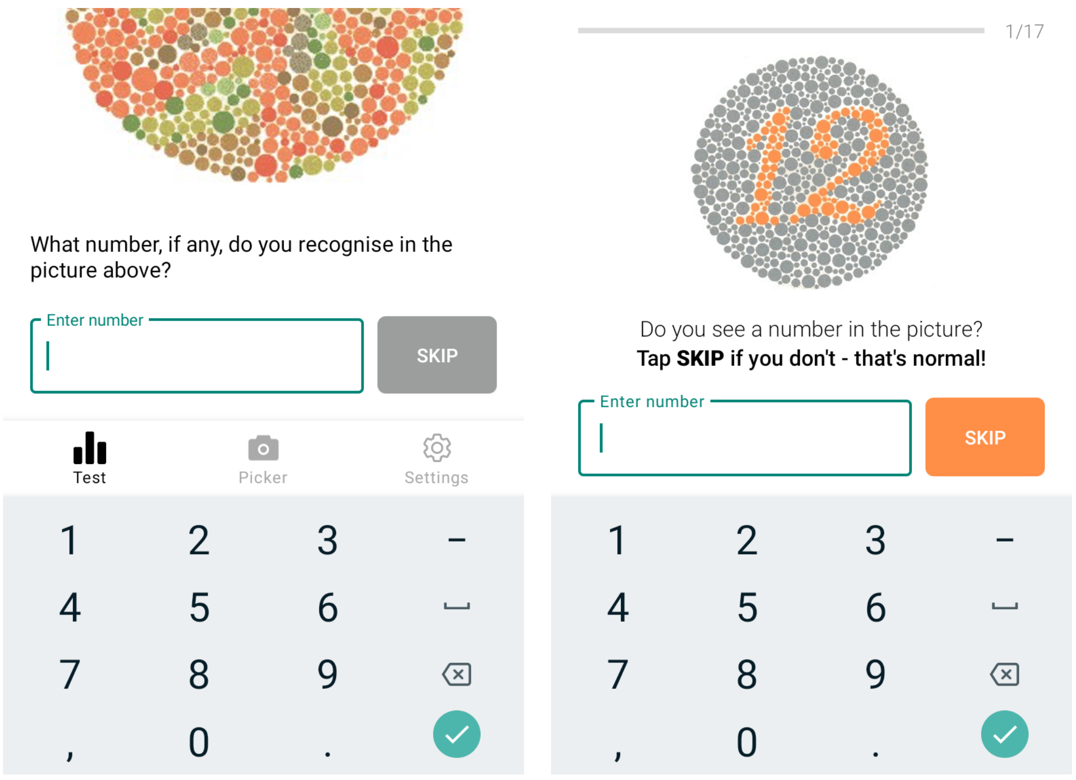


Figure 6.1: Target picker against a light background: before (left) and after (right)


What number, if any, do you recognise in the picture above?

Enter number

SKIP

Test Picker Settings

1/17

Do you see a number in the picture?
Tap **SKIP** if you don't - that's normal!

Enter number

SKIP

Figure 6.2: Test question screen comparison: before (left) and after (right)

[1]`https://play.google.com/store/apps/details?id=com.andriusgirdzius.colbi`

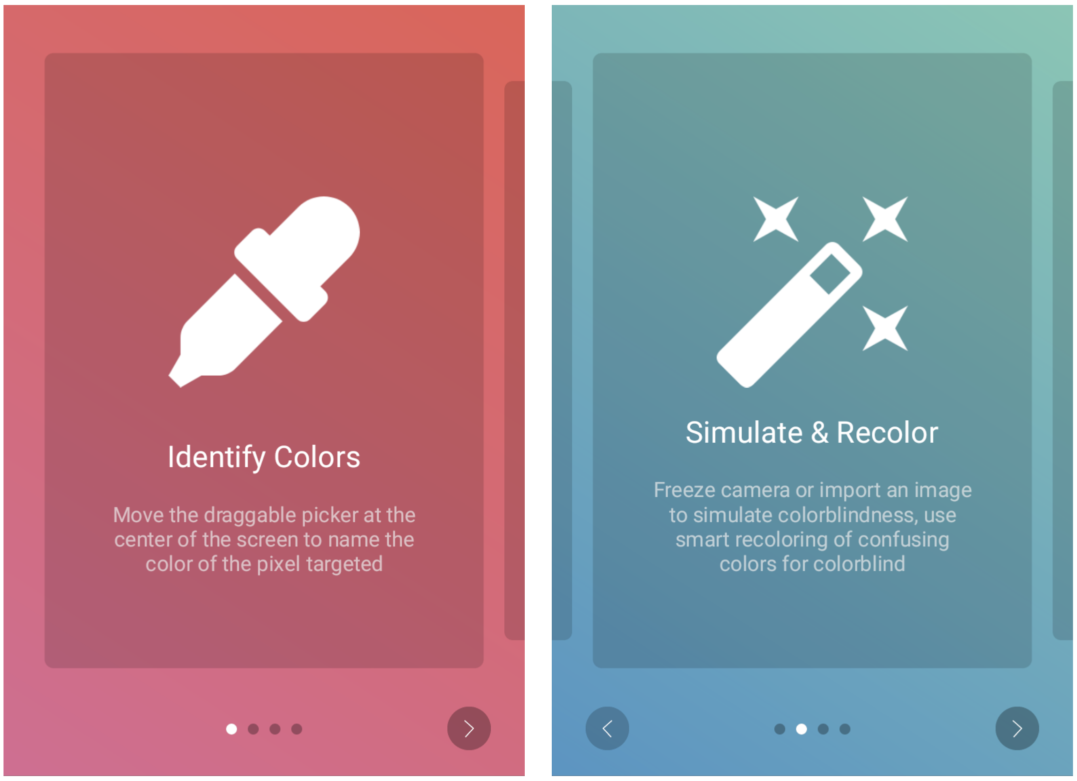

Identify Colors
Move the draggable picker at the center of the screen to name the color of the pixel targeted
Simulate & Recolor
Freeze camera or import an image to simulate colorblindness, use smart recoloring of confusing colors for colorblind

Figure 6.3: First-time use on-boarding carousel: pages 1 and 2 (of 4)

6.9 Summary

The evaluation chapter outlined the user study run with 11 participants that aimed to discover the underlying usability issues with the application prototype, as well as user opinions on the functionality it provides.

The overall feedback collected was mostly positive, with usability of the prototype being evaluated as `Great` using *SUS* (see section 2.3.1.3), but additional evaluation would be required to conclude the opinions about the impact and usefulness of the application, specifically with colour-blind participants.

Suggestions and comments from the participant study were collected and summarised to identify key points for improvement. Most proposals affecting multiple participants, as outlined in section 6.8, were implemented and released with the publicly-available version of the app on the *Google Play Store* [12].

Chapter 7

Conclusion

This final chapter of the dissertation outlines the overall project achievements and conclusions, discusses some of the challenges faced, skills acquired, limitations of the prototype and the study ran, presents future work for the application developed.

7.1 Discussion

7.1.1 Challenges faced

Throughout the development process, the most challenging part was setting up the *OpenCV* [2] library and configuring it for my needs. As explained in chapter 5, the set up process was particularly laborious - I was surprised how little proper documentation and support existed for such a known library (specifically the *Android* [3] fork). Even after the library was successfully plugged into my project, I needed to edit internal library files to make it properly scaled to fill the screen area using transformations, which in turn required a few weeks for me to later reverse engineer the touch location in the original frame returned by the camera to make the colour detection feature work.

The design part was also more challenging than I anticipated. I had a clear intention of making the app as clean, simple and user-friendly as possible as I myself have been let down by the design of many existing *Android* [3] applications out there, especially free ones. And although I had ideas of how to make the core features easily accessible independently, it proved to become increasingly more difficult as I added more of them in the same app - interface choices became counter-intuitive and contradictory. The design had to go through quite a few iterations before I settled on something that I considered to be usable, although there is still way to go about improving this (see section 7.1.2).

Finally, the implementation of the recolouring feature which I am particularly proud of was also quite challenging. The original *daltonization* [57] algorithm I was attempting to use was computing too many image values separately by looping through the pixels individually. This proved to be unusable in the app, due to other processes happening in the OS, as it could take up to 10 seconds for a high-resolution image to see the result. I got a chance to utilise my linear algebra knowledge and the built-in *OpenCV* [2] methods to vectorise the computations and in turn was able to produce a recoloured image in less than a second.

7.1.2 Limitations

As a direct result of the method of recruitment, it is possible that some participants may have been biased if they were approached directly for the evaluation study. This was inevitable as the recruitment process was particularly challenging during the pandemic, but I have included *SUS* (see section 2.3.1.3) scores of individual participants so these can be cross-referenced with recruitment information in table 6.1 to identify issues.

As well as this, due to the computationally-heavy nature of the *daltonization* algorithm [57] and the *OpenCV* [2] camera library being quite resource intensive, the current implementation does not allow *live* recolouring. I had a development version of the app that attempted this but a quite powerful device is required to keep up with the frame-rate of the native camera input to apply recolouring in real-time. This could potentially be achieved by optimising the algorithm, perhaps by using some pre-computed value dictionaries, removing other process overhead within the app or reducing the camera input frame-rate.

7.1.3 Other insights

I am really happy I had the chance to publish the app in *Google Play Store* [12] as this introduced me to the mass-deployment process of mobile applications, it also meant I learn some marketing by producing assets (banners, screenshots, descriptions) for the listing. Since I planned to pursue mobile development in the future, these skills will definitely be useful.

As well as this, I gained a lot of knowledge about colour-blindness. I was aware of the condition previously but I never fully understood the difference between different types of deficiencies, the impact it makes on design and usability. This will help me be more considerate and aware if and when designing software in the future.

7.2 Future work

All initially planned functionality, along with some extensions, was implemented. However, to my eye, the project could be developed further. The following future work is proposed:

- Algorithm optimisation to enable recolouring and simulations in real-time for the live camera feed,
- Implementation and ability to switch between alternative recolouring algorithms, e.g. *CBFS* [61],
- UI improvements to allow switching between recolouring / simulation filters without leaving the *Picker* screen,

- Further evaluation studies to assess the impact of the app, as well as the effect of recolouring for daily activities, with colourblind users,
- Additional features, such as colour pattern overlays, for easing discrimination of confusing colours for colourblind, particularly for fully-colourblind users,
- Addition of more *CVD* test options to diagnose less frequent colour deficiency types (e.g. *tritanopia*),
- Creation of an *iOS* or a platform-independent (e.g. by using *React Native* [46]) version,
- Implementation of zoom and panning capabilities for the live camera view in *OpenCV* [2],
- Additional marketing efforts to bring more awareness to the app in the *Google Play Store* [12].

7.3 Conclusions

As part of the project work, a mobile app for the *Android* [3] operating system related to colourblindness was developed and published on the *Google Play Store* [12].

The process started with requirements gathering and design work on the *Figma* [29] platform by developing an interactive design prototype. Following that, the design was implemented in *Kotlin* [39] with the aid of the *OpenCV* [2] library and the first iteration of the app was produced. This was evaluated in a remote *think aloud* (see section 2.3.1.1) study with 11 participants to gather formative input on design, usefulness and summative feedback on usability. Finally, the formative feedback was analysed and prioritised, and a new version of the app with major improvements was made public for download.

Although all major functionality planned in the design stage, namely colour naming, recolouring, simulation and testing for most frequent *CVD* types, was implemented, there are many possible improvements and extensions to the project, such as further evaluations with more colourblind participants, as outlined in section 7.2.

Bibliography

[1] agirdzius/colbi. https://github.com/agirdzius/colbi. (Accessed on 04/01/2021).

[2] Android - opencv. https://opencv.org/android/. (Accessed on 03/23/2021).

[3] Android (operating system) - wikipedia. https://en.wikipedia.org/wiki/Android_(operating_system). (Accessed on 03/18/2021).

[4] androidx.preference — android developers. https://developer.android.com/reference/androidx/preference/package-summary. (Accessed on 03/25/2021).

[5] App store - apple (uk). https://www.apple.com/uk/ios/app-store/. (Accessed on 03/25/2021).

[6] Beautiful free images & pictures — unsplash. https://unsplash.com/. (Accessed on 03/24/2021).

[7] cachapa/expandablelayout: An expandable layout container for android. https://github.com/cachapa/ExpandableLayout. (Accessed on 03/25/2021).

[8] ¡canvas¿ + colormatrix = color blindness. http://web.archive.org/web/20081014161121/http://www.colorjack.com/labs/colormatrix/. (Accessed on 03/31/2021).

[9] Cbvision - colorblind assist on the app store. https://apps.apple.com/us/app/cbvision-colorblind-assist/id1500023802?ign-mpt=uo%3D4. (Accessed on 10/25/2020).

[10] Chromatic vision simulator – ”google play”. https://play.google.com/store/apps/details?id=asada0.android.cvsimulator. (Accessed on 10/25/2020).

[11] codemybrainsout/ahoy-onboarding: Android onboarding library. https://github.com/codemybrainsout/ahoy-onboarding. (Accessed on 03/25/2021).

[12] Colbi: An assistive colorblindness app – apps on google play. https://play.google.com/store/apps/details?id=com.andriusgirdzius.colbi. (Accessed on 03/18/2021).

[13] Color blind pal - apps on google play. https://play.google.com/store/apps/details?id=com.colorblindpal.colorblindpal&hl=en_US&gl=US. (Accessed on 10/25/2020).

[14] Color blind test – apps on google play. https://play.google.com/store/apps/details?id=com.eyeexamtest&hl=en_IN. (Accessed on 10/25/2020).

[15] Color blindness - wikipedia. https://en.wikipedia.org/wiki/Color_blindness. (Accessed on 10/25/2020).

[16] Color blindness simulation research — ixora.io. https://ixora.io/projects/colorblindness/color-blindness-simulation-research/. (Accessed on 10/25/2020).

[17] Color-name dictionaries. http://people.csail.mit.edu/jaffer/Color/Dictionaries. (Accessed on 03/26/2021).

[18] Color oracle — color oracle. https://colororacle.org/. (Accessed on 10/25/2020).

[19] Color vision deficiency — aoa. https://www.aoa.org/healthy-eyes/eye-and-vision-conditions/color-vision-deficiency?sso=y. (Accessed on 03/23/2021).

[20] Colour metric. https://www.compuphase.com/cmetric.htm. (Accessed on 03/27/2021).

[21] Colour vision deficiency (colour blindness) - nhs. https://www.nhs.uk/conditions/colour-vision-deficiency/. (Accessed on 03/23/2021).

[22] Cone - live color picker on the app store. https://apps.apple.com/us/app/cone-live-color-picker/id1221305627. (Accessed on 10/25/2020).

[23] Daltonize.org: Lms daltonization algorithm. http://www.daltonize.org/2010/05/lms-daltonization-algorithm.html. (Accessed on 03/27/2021).

[24] Download android studio and sdk tools — android developers. https://developer.android.com/studio. (Accessed on 03/23/2021).

[25] Enchroma® color blind glasses — cutting-edge lens technology – enchroma uk. https://enchroma.co.uk/. (Accessed on 03/23/2021).

[26] Enchroma® color blind test - start now – enchroma. https://enchroma.com/pages/color-blind-test. (Accessed on 10/25/2020).

[27] Ethics procedure — infweb. https://web.inf.ed.ac.uk/infweb/research/ethics-and-integrity/ethics-procedure. (Accessed on 03/18/2021).

[28] Experience: Colorblindness on steam. https://store.steampowered.com/app/979100/Experience_Colorblindness/. (Accessed on 10/25/2020).

[29] Figma: the collaborative interface design tool. https://www.figma.com/. (Accessed on 03/23/2021).

[30] Free online meeting scheduling tool — doodle. `https://doodle.com/en/`. (Accessed on 03/18/2021).

[31] Global mobile consumer trends — deloitte — technology, media, and telecommunications. `https://www2.deloitte.com/global/en/pages/technology-media-and-telecommunications/articles/gx-global-mobile-consumer-trends.html`. (Accessed on 03/23/2021).

[32] google/gson: A java serialization/deserialization library to convert java objects into json and back. `https://github.com/google/gson`. (Accessed on 03/25/2021).

[33] How to zoom in an imageview in android - stack overflow. `https://stackoverflow.com/questions/43647754/how-to-zoom-in-an-imageview-in-android/43647820#43647820`. (Accessed on 03/26/2021).

[34] https://www.colourblindawareness.org/colour-blindness/. `https://www.colourblindawareness.org/colour-blindness/`. (Accessed on 03/23/2021).

[35] https://www.colourblindawareness.org/colour-blindness/living-with-colour-vision-deficiency. `https://www.colourblindawareness.org/colour-blindness/living-with-colour-vision-deficiency`. (Accessed on 04/03/2021).

[36] https://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/. `https://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/`. (Accessed on 10/25/2020).

[37] Icon sets • iconify. `https://iconify.design/icon-sets/`. (Accessed on 03/24/2021).

[38] Ishihara's test for colour deficiency: 38 plates edition – colblindor. `https://www.color-blindness.com/ishiharas-test-for-colour-deficiency-38-plates-edition/`. (Accessed on 10/25/2020).

[39] Kotlin programming language. `https://kotlinlang.org/`. (Accessed on 03/23/2021).

[40] Kuku kube: color blindness test game – ”google play”. `https://play.google.com/store/apps/details?id=appinventor.ai_information_SPLASHapps.Color_Vision`. (Accessed on 10/25/2020).

[41] Mailing lists for people in informatics — documentation. `http://computing.help.inf.ed.ac.uk/mailing-lists-guide`. (Accessed on 03/18/2021).

[42] Measuring and interpreting system usability scale (sus) - uiux trend. `https://uiuxtrend.com/measuring-system-usability-scale-sus/#calculation`. (Accessed on 04/03/2021).

[43] Microsoft forms - easily create surveys, quizzes, and polls. `https://forms.office.com/`. (Accessed on 03/18/2021).

[44] Microsoft teams – apps on google play. https://play.google.com/store/apps/details?id=com.microsoft.teams. (Accessed on 03/18/2021).

[45] Monochromacy - wikipedia. https://en.wikipedia.org/wiki/Monochromacy. (Accessed on 10/25/2020).

[46] React native · learn once, write anywhere. https://reactnative.dev/. (Accessed on 04/02/2021).

[47] Teamviewer – the remote connectivity software. https://www.teamviewer.com/en/. (Accessed on 03/18/2021).

[48] think_aloud.pdf. http://www.inf.ed.ac.uk/teaching/courses/hci/1718/tuts/think_aloud.pdf. (Accessed on 03/18/2021).

[49] Vision — biology ii. https://courses.lumenlearning.com/suny-biology2xmaster/chapter/vision/. (Accessed on 10/25/2020).

[50] Visolve - the assistive software for people with color blindness. https://www.ryobi-sol.co.jp/visolve/en/. (Accessed on 10/25/2020).

[51] Vlsomers/native-opencv-android-template: A tutorial for setting up opencv 4.5.0 (and other 4.x.y version) for android in android studio with native development kit (ndk) support. https://github.com/VlSomers/native-opencv-android-template. (Accessed on 03/26/2021).

[52] Working with the opencv camera for android: Rotating, orienting, and scaling — by mike heavers — heartbeat. https://heartbeat.fritz.ai/working-with-the-opencv-camera-for-android-rotating-orienting-\and-scaling-c7006c3e1916. (Accessed on 03/26/2021).

[53] yshrsmz/keyboardvisibilityevent: Android library to handle software keyboard visibility change event. https://github.com/yshrsmz/KeyboardVisibilityEvent. (Accessed on 03/25/2021).

[54] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *International journal of human-computer interaction*, 24(6):574–594, 2008.

[55] Elizabeth Charters. The use of think-aloud methods in qualitative research an introduction to think-aloud methods. *Brock Education : a Journal of Educational Research and Practice*, 12:68–82, 05 2010.

[56] Paul Doliotis, George Tsekouras, Christos-Nikolaos Anagnostopoulos, and Vassilis Athitsos. Intelligent modification of colors in digitized paintings for enhancing the visual perception of color-blind viewers. volume 296, pages 293–301, 04 2009.

[57] Onur Fidaner, Poliang Lin, and Nevran Ozguven. Analysis of color blindness. https://web.archive.org/web/20090725202236/http://scien.stanford.edu/class/psych221/projects/05/ofidaner/colorblindness_project.htm, 2005. (Accessed on 03/31/2021).

[58] O. Gambino, E. Minafo, R. Pirrone, and E. Ardizzone. A tunable digital ishihara plate for pre-school aged children. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5628–5631, 2016.

[59] Bruce Hanington and Bella Martin. *Universal Methods of Design : 100 Ways to Explore Complex Problems, Develop Innovative Strategies, and Deliver Effective Design So*. Quarto Publishing Group USA, Osceola, United States, 2012.

[60] Ralph Nelson Helga Kolb, Eduardo Fernandez. *Webvision: The Organization of the Retina and Visual System*. University of Utah Health Sciences Center, Salt Lake City (UT), 1995.

[61] Gennaro Iaccarino, Delfina Malandrino, Marco Percio, and Vittorio Scarano. Efficient edge-services for colorblind users. pages 919–920, 01 2006.

[62] S. Ishihara. *Ishihara's Tests for Colour-blindness*. Kanehara Shuppan, 1972.

Appendix A

Usability study

A.1 Participant Consent Form

Participant number:________________________

Participant Consent Form

Project title:	Colbi: An Android app for people with colour vision deficiencies
Principal investigator (PI):	Boris Grot
Researcher:	Andrius Girdzius
PI contact details:	Boris.Grot@ed.ac.uk

By participating in the study you agree 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.
- My participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.
- I consent to my anonymised data being used in academic publications and presentations.
- I understand that my anonymised data will be stored for the duration outlined in the Participant Information Sheet.

Please tick yes or no for each of these statements.

1.	I agree to being audio recorded.	**Yes**	**No**
2.	I agree to have my on-screen device interactions recorded.	**Yes**	**No**
3.	I allow my data to be used in future ethically approved research.	**Yes**	**No**
4.	I agree to take part in this study.	**Yes**	**No**

Name of person giving consent	Date dd/mm/yy	Signature
Name of person taking consent	Date dd/mm/yy	Signature
Andrius Girdzius		

THE UNIVERSITY *of* EDINBURGH
informatics

A.2 Participant Information Sheet

Page **1** of **3**

Participant Information Sheet

Project title:	Colbi: An Android app for people with colour vision deficiencies
Principal investigator:	Boris Grot
Researcher collecting data:	Andrius Girdzius
Funder (if applicable):	N/A

This study was certified according to the Informatics Research Ethics Process, RT number 2019/89367. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

Andrius Girdzius (s1642301@ed.ac.uk), Boris Grot (Boris.Grot@ed.ac.uk)

What is the purpose of the study?

The goal of the study is to test the usability and usefulness of a mobile application regarding colour vision deficiencies developed as part of an undergraduate project.

Why have I been asked to take part?

Your feedback would help to support the next iteration of the development process and further improve the application.

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, up until the publication of the project research paper, without giving a reason. After this point, personal data will be deleted, and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw, contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

THE UNIVERSITY *of* EDINBURGH informatics

You will be asked to interact with a prototype mobile application related to colour vision deficiencies. We will be collecting data using the Think Aloud protocol so we will expect you to narrate the process of attempting the tasks performed, i.e. you may say what you do, feel, think, see, etc.

After the tasks are complete, follow up questions regarding your experience may be asked in the form a short interview or an online (GDPR compliant) survey. You may also be asked some information about your background: your gender, presence of colour vision deficiencies (to be only used to exhibit participant variety in the study).

The full session will take up to 30 minutes. This is a one-time session and no follow up will be required. The session location and time will be agreed individually.

If permission is given, audio and on-screen device interaction (via screen share) recording will also be taken. The recording will be transcribed and destroyed afterwards. Any personal or identifiable information will be anonymised by replacing with a unique participant number in the data.

Compensation.

You will be not be compensated for the participation.

Are there any risks associated with taking part?

There are no significant risks associated with participation.

Are there any benefits associated with taking part?

There are no direct benefits associated.

What will happen to the results of this study?

The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymized: We will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information can also be used for future research. Your data may be archived for a maximum of 4 years. All potentially identifiable data will be deleted within this timeframe if it has not already been deleted as part of anonymization.

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 (Andrius Girdzius, Boris Grot).

All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?
The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

Who can I contact?
If you have any further questions about the study, please contact the lead researcher, Andrius Girdzius (s1642301@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.
Updated information.
If the research project changes in any way, an updated Participant Information Sheet will be made available on http://web.inf.ed.ac.uk/infweb/research/study-updates.
Alternative formats.
To request this document in an alternative format, such as large print or on coloured paper, please contact Andrius Girdzius (s1642301@ed.ac.uk).
General information.
For general information about how we use your data, go to: edin.ac/privacy-research

THE UNIVERSITY *of* EDINBURGH
informatics

A.3 Tasks

1. Download the app from the Google Play Store: tinyurl.com/colbi-app. After installing, launch it and grant any permissions as requested.
2. Perform a color vision deficiency test. Identify your diagnosis and name a test question that would be perceived differently in normal and deficient vision.
3. Import an image from your camera roll. Find the color name of the top right of the image.
4. Return to the live camera view.
5. The app also allows smart recolouring of the view to enable easier discrimination of confusing colours for people with colour vision deficiencies. Enable this in the camera view.
6. Using the app, you may simulate various types of color blindness. Set the simulation mode to 'Tritanopia' and then enable the simulation for the camera view.

A.4 Follow-up questions

1. Did you encounter any issues while attempting the tasks?
2. What do you think could be improved?
3. Was there anything you disliked?
4. Anything that you particularly liked?
5. Please score the following 10 statements on usability from Strongly disagree (1) to Strongly agree (5):
   (a) I think that I would like to use this system frequently.
   (b) I found the system unnecessarily complex.
   (c) I thought the system was easy to use.
   (d) I think that I would need the support of a technical person to be able to use this system.
   (e) I found the various functions in this system were well integrated.
   (f) I thought there was too much inconsistency in this system.
   (g) I would imagine that most people would learn to use this system very quickly.
   (h) I found the system very cumbersome to use.
   (i) I felt very confident using the system.
   (j) I needed to learn a lot of things before I could get going with this system.

6. Do you have any other comments or questions?

A.5 Think aloud script

Hello, my name is Andrius.

Today I will ask you to perform a series of simple tasks on a prototype of an Android application related to color vision deficiencies. Your participation today is purely voluntary and you may stop at any time. The purpose of this study is identify potential issues with the app and evaluate design and usability. Remember - we are testing the app, not you.

In this observation, we are interested in what you think about as you perform the tasks we are asking you to do. In order to do this, I am going to ask you to talk aloud as you work on the task. What I mean by “talk aloud” is that I want you to tell me everything you are thinking from the first time you see the statement of the task till you finish the task. I do not want you to try and plan out what you say or try to explain to me what you are saying. Just act as if you were alone, speaking to yourself. It is most important that you keep talking. If you are silent for a long period of time, I will ask you to talk. Do you understand what I want you to do?

[Wait for confirmation]

Good. Now we will begin with some practice problems. First, I will demonstrate by thinking aloud while I solve a simple problem: “How many windows are there in my mother’s house?”

[Demonstrate thinking aloud]

Now it is your turn. Please think aloud as you multiply 120 * 8.

[Let them finish]

Good. Now, those problems were solved all in our heads. However, when you are interacting with the app you will also be looking for things, and seeing things that catch your attention. These things that you are searching for and things that you see are as important for our observation as thoughts you are thinking from memory. So please verbalize these too.

As you are doing the tasks, I won’t be able to answer any questions. But if you do have questions, go ahead and ask them anyway so I can learn more about what kinds of questions the app brings up. I will answer any questions after the session. Also, if you forget to think aloud, I’ll say, “please keep talking.” Do you have any questions about the think aloud protocol?

[Wait for confirmation]

Now I have some tasks prepared for you. I am going to go over them with you and see if you have any questions before we start.

[Display the task list on screen]

Here are the tasks you will be working on. Why don't you read them aloud just so you can get comfortable with speaking your thoughts?

A.6 Categorised study results

Category	**Comment**	**Occurrences**
Test	Unsure if should skip during test if can't see a number	**4**
	Expected test as the first thing to be opened	**1**
	Did not notice / read instructions	**2**
	Scroll needed to see image / input / progress	**4**
	Did not notice scrolling was possible in test result screen	**3**
	Test plate answer formatting wrong (narrow screen)	**2**
	Unsure how many questions left	**2**
Simulation / recolouring	Not sure where to enable recolouring / simulation	**9**
	Unclear task / skipped	**1**
	Not sure what filter is currently selected in camera view	**1**
	Frozen picture gone after settings screen	**1**
	Boolean buttons one another if both enabled	**1**
	Expected tap and hold to change filter mode	**1**
Color detection	Unsure how to pick the color	**2**
	Did not notice target picker	**4**
	Unclear task / skipped	**3**
	Thought you need to tap on target area, not drag picker	**2**
Settings	Confused between simulation / recolouring settings	**1**
Bugs	Context switching reset app fragment unintentionally	**1**
	Image imported sideways	**5**
	Camera view crash	**2**
	Initial colour detected incorrectly	**11**
	Forced dark mode ->incorrect detected color	**1**
	Forced dark mode ->test plate background is not clear	**1**
	Incorrect pixel selection (color detection)	**1**
	Crash after image import	**2**
	Can't turn off torch while frozen / imported	**1**
	Inconclusive test	**5**
	Filters - many different types and levels	**1**
	Image import	**3**

Liked

Table A.1 continued from previous page

Category	**Comment**	**Occurrences**
	Target picker convenient	**2**
	Target picker vibration	**1**
	Same icons in camera / settings	**2**
	Buttons and fonts nice / big / not easy to miss	**4**
	Interface in general / ease of use	**6**
	Color detection functionality	**3**
	Recolouring	**2**
	Testing within the app	**2**
Disliked	Loss of frames (i.e. when tapping import, recolouring, etc.)	**1**
	Phone getting hot in camera view (could be Teams)	**1**
	Too much text in instructions page (testing)	**1**
	Crashes	**2**
	Camera permissions required to work	**1**
	Camera/settings move resets import/frozen image	**1**
Suggestions	Target picker alternating colours (black/white/black…)	**1**
	Guide/tutorial on first use / settings page	**9**
	There should be feedback on skipping a test plate	**1**
	Skip button (testing) is gray ->implies not clickable	**2**
	Show ’Skipped’ instead of ’-’ in test results	**1**
	Take out navigation bar above keypad in testing mode	**1**
	About this test’ should be collapsed by default	**1**
	Stabilise camera if possible	**1**
	Test questions sorted in terms of difficulty	**1**
	Add instructions in settings about filter enable steps	**1**
	Tool-tips about recolouring/simulation above ’Freeze’	**1**
	Past deficiency test results saved	**1**
	Information / help feature	**1**
	Live recolouring	**1**
	Simulate/recolour buttons present always (e.g. disabled)	**1**
		131

Table A.1: Summary of comments per each participant from the usability study

Appendix B

Transformation matrices

B.1 Daltonization

$$RGB2LMS[57] = \begin{bmatrix} 17.8824 & 43.5161 & 4.11935 & 0.0 \\ 3.45565 & 27.1554 & 3.86714 & 0.0 \\ 0.0299566 & 0.184309 & 1.46709 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.1}$$

$$LMS2RGB[57] = \begin{bmatrix} 0.08094 & -0.13050 & 0.11672 & 0.0 \\ -0.01025 & 0.05402 & -0.11361 & 0.0 \\ -0.00037 & -0.00412 & 0.69351 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.2}$$

$$ERR2MOD[57] = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.7 & 1.0 & 0.0 & 0.0 \\ 0.7 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.3}$$

$$LMS\_SIM\_DEUTAN[57] = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.494207 & 0.0 & 1.24827 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.4}$$

$$LMS\_SIM\_PROTAN[57] = \begin{bmatrix} 0.0 & 2.02344 & -2.52581 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.5}$$

$$LMS\_SIM\_TRITAN[57] = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -0.395913 & 0.801109 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.6}$$

B.2 Simulation

$$SIM\_PROTANOPIA[8] = \begin{bmatrix} 0.567 & 0.433 & 0.0 & 0.0 \\ 0.558 & 0.442 & 0.0 & 0.0 \\ 0.0 & 0.242 & 0.758 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.7}$$

$$SIM\_PROTANOMALY[8] = \begin{bmatrix} 0.817 & 0.183 & 0.0 & 0.0 \\ 0.333 & 0.667 & 0.0 & 0.0 \\ 0.0 & 0.125 & 0.875 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.8}$$

$$SIM\_DEUTERANOPIA[8] = \begin{bmatrix} 0.625 & 0.375 & 0.0 & 0.0 \\ 0.7 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.7 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.9}$$

$$SIM\_DEUTERANOMALY[8] = \begin{bmatrix} 0.8 & 0.2 & 0.0 & 0.0 \\ 0.258 & 0.742 & 0.0 & 0.0 \\ 0.0 & 0.142 & 0.858 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.10}$$

$$SIM\_TRITANOPIA[8] = \begin{bmatrix} 0.95 & 0.05 & 0.0 & 0.0 \\ 0.0 & 0.433 & 0.567 & 0.0 \\ 0.0 & 0.475 & 0.525 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.11}$$

$$SIM\_TRITANOMALY[8] = \begin{bmatrix} 0.967 & 0.033 & 0.0 & 0.0 \\ 0.0 & 0.733 & 0.267 & 0.0 \\ 0.0 & 0.183 & 0.817 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.12}$$

$$SIM\_ACHROMATOPSIA[8] = \begin{bmatrix} 0.299 & 0.587 & 0.114 & 0.0 \\ 0.299 & 0.587 & 0.114 & 0.0 \\ 0.299 & 0.587 & 0.114 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.13}$$

$$SIM\_ACHROMATOMALY[8] = \begin{bmatrix} 0.618 & 0.320 & 0.062 & 0.0 \\ 0.163 & 0.775 & 0.062 & 0.0 \\ 0.163 & 0.320 & 0.516 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.14}$$