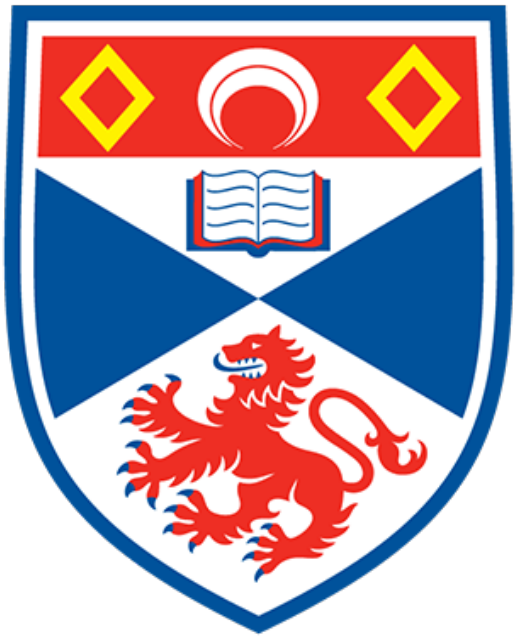
What is the Colour?

Jae Woo Chang

190027921



University of St Andrews

This thesis is submitted in partial fulfilment for the degree of Bachelors at the University of St Andrews

Supervised by Saleem Bhatti

School of Computer Science

29th March, 2024

Abstract

Colour is everywhere in our lives, an important and seemingly normal aspect that we often overlook until the need for precise replication or understanding arises. Recognizing colours, a task that seems inherently simple, has become a complex domain of interest with the development of technology. As we navigate through the digital age, the recognition and calibration of colours transcend mere identification. These processes have become professional and essential tasks, particularly valuable for individuals with specialized needs. Among these, people with colour vision deficiencies face unique challenges, as the world of colour they perceive differs significantly from the norm, making tasks like distinguishing between certain hues not just difficult but sometimes impossible without assistance. This digital journey into the spectrum of colours has spurred the development of an application that marries technology with accessibility. Aimed at not just recognizing but also accurately calibrating colours, this tool seeks to bridge the gap for those who experience colour differently, ensuring that colour recognition is not a privilege but a universal tool accessible to all. This study leads to the development of a colour recognition application, with a focus on ensuring accurate colour calibration and providing an interface that is easy for users to navigate.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 7945 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Acknowledgements

I am grateful to my supervisor, Saleem Bhatti, for his continuous support throughout this academic year, especially in introducing me to ideas that I was not previously familiar with, especially in the field of Colour Calibration and App Development. His insight and feedback have been greatly valued. Also, I would like to thank my family and friends for the support they have provided for my project and degree. Their unwavering belief in my abilities and their encouragement during challenging times have been a source of strength and motivation for me. Above all, I extend my deepest gratitude to my father, Jin Wook Chang, whose experience with colour blindness inspired me to develop a colour recognition application.

Contents

1 Table of Contents

2 Introduction

Since the advent of digital technology, the field of color recognition has undergone a transformative evolution, increasingly intertwining with the capabilities of modern mobile devices. This domain, rooted in the early days of digital imaging and computer graphics, has historically aimed at enabling machines to recognize and interpret colors in a manner similar to human perception. The contemporary significance of color recognition, however, extends far beyond its initial applications, driven by its utility in enhancing accessibility, fostering creativity in art and design, and facilitating educational advancements through interactive digital interfaces.

In this dissertation, I delve into the development of a color recognition application crafted using the Flutter framework, motivated by the desire to leverage its cross-platform efficiency to deliver an accessible, user-friendly tool for accurate color identification. This venture is particularly aimed at addressing the nuanced challenges present in the accurate identification of colours under varying lighting conditions and presenting colour information in an intuitive format accessible to all users, including those with colour vision deficiencies. My research encompasses an extensive analysis of existing colour recognition technologies, a deep dive into Flutter's development environment, and the integration of colour analysis methods.

3 Context Survey

Colour correction and recognition technology plays a pivotal role across various domains, including digital imaging, assistive technology, and industrial automation, by enabling precise identification and classification of colours. This literature review explores the foundational theories, methodologies, and recent developments in colour recognition, with a focus on their application to mobile app development. The review aims to provide a comprehensive overview of the field, identifying trends and gaps that could inform the development of a new colour recognition app.

3.1 Colour Correction & Colour Recognition

3.1.1 Colour Correction

Colour correction is a critical process across various digital imaging disciplines, such as photography, film production, and graphic design. Its main goal is to modify and

improve the colour balance and accuracy of images or videos. To ensure that the colours are a true representation of the original scene or are modified to achieve a specific aesthetic effect. This process involves the precise adjustment of an image's colour properties, including hue, saturation, brightness/contrast and white balance (Strickland, 2021). These properties such as hue, saturation, brightness/contrast, and white balance are crucial elements that define the visual characteristics of a photograph or digital image. Hue refers to the aspect of colour which is determined by the wavelength of light; it is what we commonly refer to when identifying colours like red, blue, or yellow. Saturation measures the intensity or purity of the hue, ranging from vibrant, rich colours at high saturation to more muted tones at lower saturation. Brightness/contrast represents the lightness or darkness of the image, with brightness affecting the overall exposure and contrast defining the difference between the lightest and darkest parts of the image, contributing to its depth and dimensionality. White balance is the process of removing unrealistic colour casts, so that objects which appear white in person are rendered white in your photo. It adjusts the colours of the image to look natural and accurate, compensating for the colour temperature of different light sources (Barnhart, 2023). Together, these properties play a pivotal role in the aesthetic and emotional appeal of images, influencing the viewer's perception and interpretation. By manipulating these attributes, either automatically through software algorithms or manually by artists or technicians, colour correction strives to imbalances and fulfil creative intentions, enhancing the overall visual appeal of the digital content (Adobe, 2021).

There are widespread applications of color correction such as in the domain of film and television. In the domain of film and television, it is employed to craft a visual mood or tone that supports the narrative. Similarly, photographers use it to ensure that their final prints or digital displays accurately reflect the scene they've captured. In graphic design, colour correction is essential for maintaining colour consistency across different media, such as between digital screens and printed materials. Moreover, recently it has been spreading across various online platforms and applications, enhancing user engagement by improving the visual quality of content shared on social media, web interfaces, and mobile apps (Nabeel, 2021). This broadening scope highlights the growing recognition of the importance of accurate colour representation in our increasingly digital world.

3.1.2 Colour Recognition

Colour recognition technology represents a pivotal area of research and development within the fields of computer vision and digital image processing. At its core, colour recognition involves the capability of computer systems to detect and identify colours within digital imagery accurately. This technology leverages the principles of optics,

algorithms, and machine learning to interpret the visual data in a manner akin to human vision, albeit within the confines of digital parameters (Samara, et al., 2017).

The origin of colour recognition can be traced back to early explorations in computer vision, where the primary challenge was to enable computers to "see" and interpret visual information. Initial approaches were heavily reliant on the RGB (Red, Green, Blue) colour model, a straightforward representation of colour in digital images. However, the RGB model's limitations in mimicking human colour perception led to the exploration of more complex colour spaces such as HSV (Hue, Saturation, Value) and LAB ((Lightness, A, B). These colour spaces offer a more nuanced understanding of colour by separating chromaticity from luminance, thereby aligning more closely with human colour differentiation capabilities (Samara, et al., 2017).

Advancements in digital imaging and machine learning have significantly impacted the development of colour recognition technologies. The introduction of high-definition cameras and sophisticated image sensors has improved the quality and accuracy of colour capture in digital images. Concurrently, the evolution of machine learning algorithms, especially in deep learning, has revolutionized colour recognition. Deep learning models that are trained on extensive datasets of images can now identify and classify colours with remarkable accuracy, even in complex images with varying lighting conditions and backgrounds (Samara, et al., 2017).

Colour recognition technology finds applications across a wide spectrum of industries and domains. In industrial manufacturing, it is used for quality control, ensuring that products meet specific colour standards. In retail and fashion, it aids in colour matching and trend analysis, enhancing customer experiences. Moreover, colour recognition plays a crucial role in autonomous vehicles and robotics, enabling machines to navigate and interact with their environment effectively. In the realm of healthcare and accessibility, colour recognition technologies offer novel solutions. For individuals with colour vision deficiencies, applications leveraging colour recognition can alter colour schemes in real-time to improve visibility and comprehension. Furthermore, in digital art and preservation, colour recognition helps in the restoration of artworks, ensuring that colours match the original hues as closely as possible (Samara, et al., 2017).

3.2 Colour Distance and Euclidean distance

Colour distance measures the perceived difference between two colours. This concept is crucial in digital image processing, colour recognition, and colour correction to determine how similar or different two colours are. In the context of colour spaces like RGB (Red, Green, Blue) or LAB (Lightness, A, B), colour distance is calculated using formulas that quantify the disparity between two colours based on their colour space coordinates (Lea, et al., n.d.). The goal is to represent the visual difference between

colours in numerical terms, which can be particularly useful in tasks such as colour matching, colour correction, and in algorithms that require colour classification or segmentation.

Euclidean Distance

Euclidean distance is a geometric concept that represents the shortest path between two points in a multi-dimensional space, commonly referred to as the "straight-line" distance. In a two-dimensional space, for instance, the Euclidean distance between two points (x_1, y_1) and (x_2, y_2) is calculated using the formula (Lea, et al., n.d.):

Distance = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

In the context of colour, when colours are represented in a three-dimensional space (such as the RGB colour space), the Euclidean distance can be used to calculate the colour distance between two colours. Each colour is considered a point in this space, with its coordinates given by the intensity values of Red, Green, and Blue. Therefore, the Euclidean distance between two colours C_1 and C_2 with RGB values (R_1,G_1,B_1) and (R_2 ,G_2,B_2) respectively, can be calculated as (Lea, et al., n.d.):

Colour Distance = $\sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$

3.3 SpyderCheckr 24

The SpyderCheckr24 is a colour calibration tool designed for photographers and videographers to ensure colour accuracy and consistency in their images and video footage. It is a compact version of the larger colour calibration targets, featuring 24 scientifically engineered colour patches that represent the colours of natural objects such as human skin, foliage, and blue sky, as well as primary and secondary colours. These patches are used to create a colour profile that can be applied to images or footage to adjust colours to match their true appearance (SpyderCheckr User's Guide, n.d.).

The primary function of the SpyderCheckr24 is to provide a standard reference for colour correction in the post-production process. By including a shot of the SpyderCheckr24 in the same lighting conditions as the main subject, photographers and videographers can use software to analyse the colour differences between the captured colours and the known colour values of the SpyderCheckr24 patches. This analysis allows the software to create a profile that corrects any colour deviations

caused by the camera sensor, lens, lighting conditions, or other factors. This profile can then be applied to all images or footage captured under the same conditions to ensure consistent and accurate colour reproduction (SpyderCheckr User's Guide, n.d.).

Feature	Datacolor SpyderCHECKR	Datacolor SpyderCHECKR 24
Patch Number	48	24
Patch Size	Large	Large
Color Coverage	Saturated & Low Saturation	Saturated Colors
Skintone Patches	8	2
Gray Ramp Patches	13	6
Near White Tints	3	0
Near Black Tones	3	0
Gray Face	Large	Medium
Large 18% Gray Patch	Yes	Yes
FadeCheckr	Yes	No
Tripod Mount	Yes, 1/4" x 20	No
Cube Mount	Yes, 1/4" x 20	No
Rigid Case	Yes	No
Thin, Flexible Case	No	Yes
Dimensions Open	31cm wide, 23 cm high, 20mm thick	14cm wide, 20cm high, 1mm thick
Dimensions Stored	15cm wide, 23 cm high, 15mm thick	15cm wide, 21 cm high, 4mm thick

Figure 1 – SpyderCheckr24 (SpyderCheckr User's Guide, n.d.).

3.4 Why Grey Reference Card?

The grey reference card is indispensable in the realm of digital photography and videography, especially during the post-processing phase. This phase is critical for refining captured images to achieve the desired visual outcomes. The key benefit of utilizing a grey reference card lies in its neutral colour profile, which acts as a consistent benchmark for colour balance calibration within post-production software. This ensures that any colour biases, whether from the camera sensor or lighting conditions,

are corrected. Consequently, the final image faithfully represents the original scene, mirroring what the human eye perceives (Pro Edu, 2024).

Beyond colour correction, the grey reference card is vital for adjusting exposure in post-processing. With its established reflectance value at approximately 18% grey, which occupies the midpoint between white and black with RGB values of (120,120,120), it becomes an essential tool for setting exposure levels across various images. This process helps preserve the dynamic range, facilitating nuanced adjustments in both highlights and shadows to prevent detail loss from over- or underexposure. Such adjustments are particularly beneficial under challenging lighting conditions, ensuring balanced exposure in the final imagery (Pro Edu, 2024).

Employing a grey reference card during post-processing not only secures accurate color balance and exposure but also maintains visual consistency across images. This practice not only enhances the shooting phase but lays the groundwork for a streamlined and precise editing workflow. By guaranteeing correct colour and exposure calibration, the grey reference card unlocks the true potential of digital imagery, elevating the quality and authenticity of the visual representation (Pro Edu, 2024).

3.5 Technological Advancements and Colour Recognition Apps

Advancements in smartphone hardware and colour calibration are significant drivers behind the improved performance and capabilities of colour recognition apps. Modern smartphones come equipped with high-quality cameras and powerful processors, enabling these apps to analyse colours with greater accuracy and speed. Furthermore, the application of machine learning and AI algorithms has made colour recognition more sophisticated, allowing for nuanced detection and identification that goes beyond basic colour matching (Nabeel, 2021). These technological improvements not only enhance the user experience but also open new possibilities for app functionality.

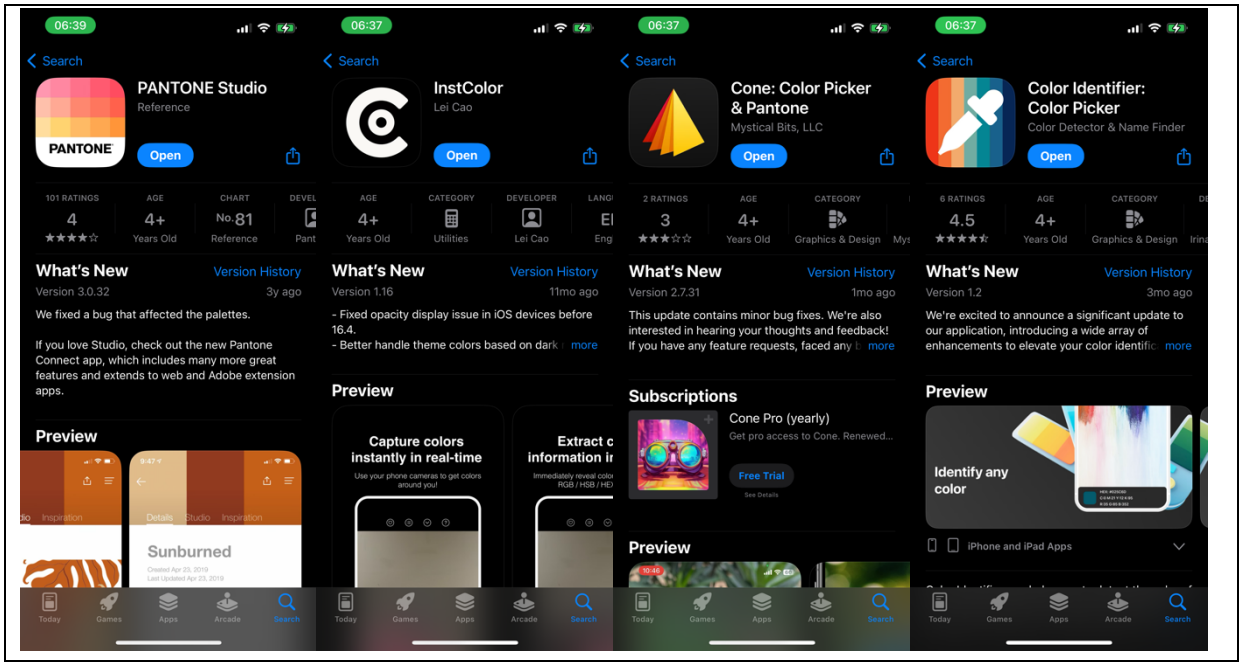


Figure 2 – Screenshots of App Store 'Colour Recognition' search results

4 Preliminary Project Objectives

Below is a list of original objectives of the project as documented in the Description, Objectives, Ethics & Resources (DOER) document from the first semester.

1. Development of colour recognition app: The primary objective is to design and develop a mobile application capable of recognizing and classifying colours within images or backgrounds.
2. Real-World colour recognition: Ensure the app's effectiveness in real-world scenarios, accounting for diverse lighting conditions, angles, and objects within images.
3. User-Friendly interface: Design the interface to make the app accessible to a broad audience, including those with limited technical expertise.
4. Evaluation and Accuracy improvement: conduct various testing and evaluation of the app's performance, followed by iterative improvements to enhance its accuracy and reliability of colour recognition.

5 Requirements Specification

5.1 Functional Requirements

Functional requirements describe the specific behaviors or functions of the application.

1. Colour Detection and Recognition
   - The app must be able to detect and recognize colours in images captured by the device's camera or uploaded from the device's storage.
   - It should provide the ability to identify primary colours along with shades and possibly suggest names for the colours detected.
2. Image Capture and Upload
   - Users should be able to capture images directly through the app using the device's camera.
   - The app must allow users to upload images from the device's gallery for colour recognition.
3. Colour Information Display
   - Upon recognizing colours in an image, the app should display relevant information about the detected colours, such as colour names, RGB values, and hex codes.
   - If applicable, the app could also provide additional information like colour psychology, complementary colours, and usage suggestions.
4. User Interface
   - The app should offer an intuitive and user-friendly interface, allowing users to easily capture or upload images, view detected colours, and access colour information.
   - It should support basic image editing functions, such as cropping and zooming, to enhance the colour recognition process.
5. Settings and Preferences
   - Users must be able to adjust settings related to colour recognition sensitivity, app themes, and notifications.

5.2 Non-Functional Requirements

Non-functional requirements define the system's operation and may include performance, security, and usability criteria.

1. Performance
   - The app should perform colour recognition tasks quickly and efficiently, with minimal latency, to ensure a smooth user experience.
   - It should be optimized for low memory usage and battery consumption on mobile devices.
2. Usability
   - The app must be designed with a focus on ease of use, with clear navigation and consistent design elements.
   - It should be accessible to users with varying levels of technical proficiency and support accessibility features for users with disabilities.

3. Security
    - The app must ensure user data privacy and security, particularly if users are able to create accounts or share images.
    - It should implement secure data storage and transmission practices.
4. Compatibility
    - The app should be compatible across a wide range of devices and screen sizes.
    - It should support the latest versions of iOS and Android, with consideration for backward compatibility.

6 Ethics

In the development and deployment of the Colour Recognition Application, consideration has been given to ethics, focusing specifically on user privacy and the handling of personal information. Ensuring the ethical use of technology, especially in applications that interact closely with personal data and devices, is crucial.

Upon downloading and initially launching the application, users are greeted with a clear and straightforward prompt: "The Colour Recognition App would like to find and connect to devices on your local network." This prompt allows users the autonomy to either "Allow" or "Don't Allow" the app's access to their local network. This level of upfront transparency ensures that users are fully aware of the app's functionalities and the extent of network access required, enabling them to make informed decisions.

Similarly, the application's functionality involving the user's photo library is designed with user consent at its core. Before the app accesses any images, it presents a prompt stating, "The Colour Recognition App would like to access your photo library," thereby giving users the option to "Allow", “Limit Access” or "Don’t Allow" access. This step is crucial for maintaining user control over their personal data and ensuring that access to their photo library is granted explicitly by the user.

For features that require image capturing using the device's camera, the app similarly asks for user permission through a prompt: "The Colour Recognition App would like to access the camera." Here again, users have the choice to either "Allow" or "Don’t Allow" access, reinforcing the principle of user consent and personal autonomy over device functionalities.

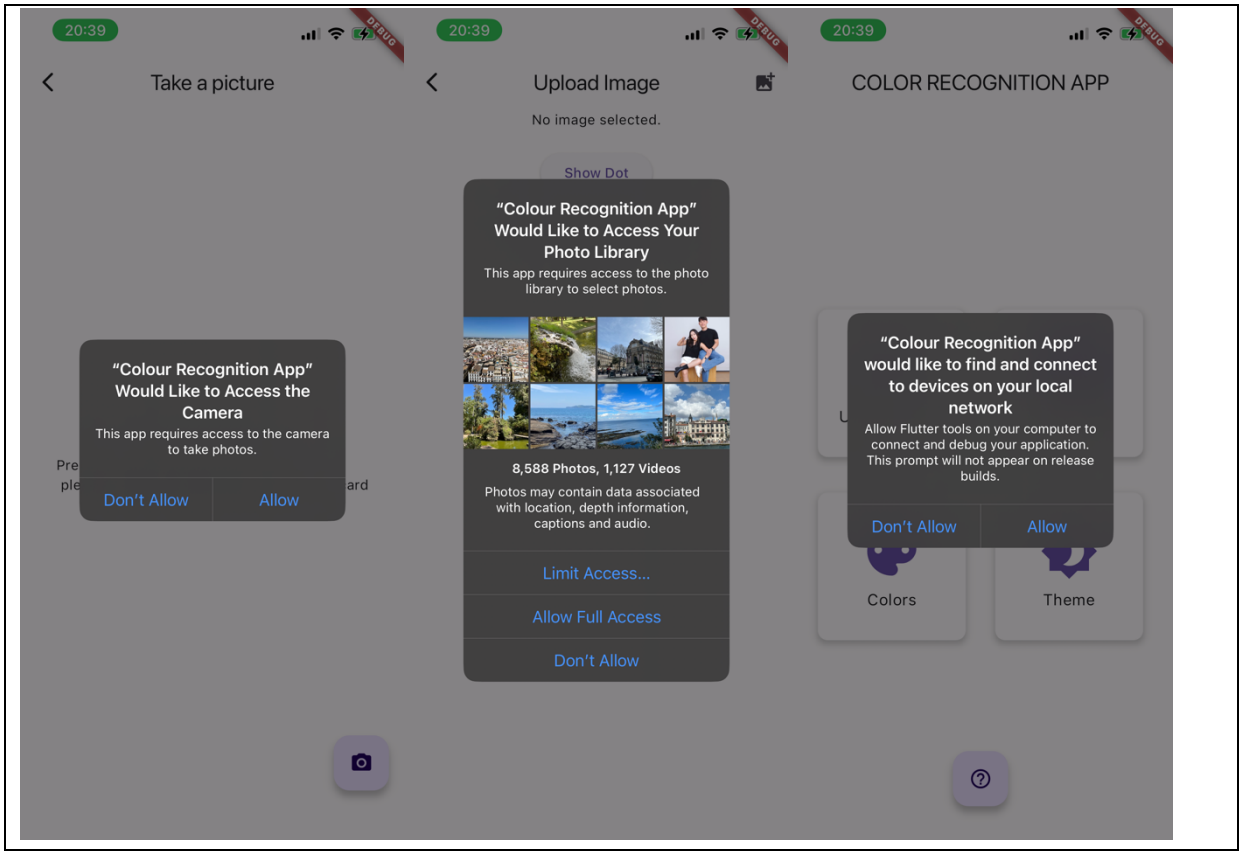


Figure 3 - Prompt messages for user to allow access to personal data.

7 Design & Implementation

7.1 Design

7.1.1 UI Design

When the app opened by the user, the app directs the user to the main screen. From the main screen, there are buttons here that navigate to different screens. And this is a brief explanation of what each screen does.

1. Image upload Screen – This screen allows users to upload an image from their photo library. It displays the RGB values and hex code at the location of a selected dot, as well as the image's dominant colours.
2. Image Capture Screen – Users can capture images using the camera, including an external grey reference card within the scene. Post-capture, users have the option to resize and crop the image around the grey reference card. The app then adjusts the image based on the cropped reference card. In the adjusted image, users can view the RGB values and hex code at the dot's position, along with the image's dominant colours.
3. Colour Screen – This screen engages users by allowing them to select primary and accent colours and then displays a range of hues within the chosen colour's

spectrum. It serves as an enjoyable way for users to explore and familiarize themselves with different shades and variations of colours.

4. Theme Screen – Allows users to customize the app's appearance with several colour themes: Dark Mode, Blue-Yellow, Red, and Green, enabling a personalized user experience.
5. Help Screen – Provides users with guidance on how each screen functions, offering step-by-step assistance for navigating and utilizing the app's features effectively.

7.1.2 System Architecture Design

In the design component of my Flutter app project, for colour calibration, I employed the concept of Euclidean distance, a fundamental mathematical principle, to enhance the accuracy and precision of colour calibration in digital images. The rationale behind incorporating Euclidean distance into the app's functionality stems from its ability to quantitatively measure the disparity between two points in a multidimensional space, such as the RGB colour space used in digital imaging (Lea, et al., n.d.).

Understanding that accurate colour representation is essential in digital imaging, I devised a method wherein an external reference card with known RGB values serves as a benchmark for colour accuracy. By comparing the RGB values of each pixel in the image to those of the grey reference card using the Euclidean distance formula, I could assess the colour deviation present. This assessment is based on the premise that the Euclidean distance between the colour values of the image pixels and the reference card reflects the extent of colour inaccuracies.

If the RGB values of the image are adjusted such that the Euclidean distance to the reference card's RGB values is minimized, the result is a significantly reduced colour disparity. This minimization process effectively aligns the image's colour representation closer to the real-world colours as depicted by the reference card. The hypothesis underpinning this approach is that by reducing the colour distance, we can achieve a more precise colour rendition in the digital image, making it appear closer to its true appearance.

In the context of my Flutter app project, the integration of the Euclidean distance concept with the use of a grey reference card was a strategic decision aimed at refining the colour hue and brightness of digital images. The grey reference card, selected for its neutral and consistent colour in varied lighting conditions, served as an optimal external reference for ensuring colour accuracy. This card's known RGB values provide a standard against which the colours in digital images can be precisely calibrated, underpinning the rationale for its selection. By comparing the RGB values of each pixel in the image to those of the grey reference card using the Euclidean distance, it was

possible to accurately gauge the extent of colour inaccuracies present. This process hinges on the understanding that minimizing the Euclidean distance between the image's colour values and those of the reference card is key to reducing colour disparities. It leverages mathematical precision to tackle the challenge of colour accuracy in digital imaging.

Thus, by adjusting the RGB values of the image to minimize this distance, the colour calibration process effectively aligns the image's colour representation with the real-world colours denoted by the reference card. This methodology not only improves the colour hue of the image and the brightness but also ensures that the digital representation closely mirrors its true appearance. The hypothesis that underlies this approach posits that a reduced colour distance results in a more accurate colour rendition, a premise that has been significantly validated through the application of these combined techniques in my project.

For my project, I incorporated the SpyderCheckr24 as an external colour reference due to its primary function as a colour calibration tool. This decision was driven by the presence of 24 scientifically engineered colour patches on the SpyderCheckr24, offering a level of precision markedly superior to that of a colour chart printed on standard A4 paper. I specifically used the 18% grey colour on the SpyderCheckr patch. Although the SpyderCheckr24 comes with its own calibration software, I didn't use it. This was because SpyderCheckr already came with their own software of calibrating colours, that could not be integrated with the app.

Through this approach, the project contributes to enhancing the quality and authenticity of digital images, making it a valuable addition to the discourse on digital imaging technology in my dissertation.

7.2 Implementation

I chose to develop my application using Flutter, an app development framework powered by the Dart language, primarily for its capability of facilitating cross-platform app creation. This means that with just one set of code, I can efficiently produce an application that runs seamlessly on both iOS and Android platforms. Moreover, Flutter is renowned for its exceptional support for crafting visually appealing user interfaces, allowing for the design of engaging and intuitive app experiences. Additionally, Flutter's rich set of widgets and tools simplifies the development process, enabling faster coding and testing. This framework's adaptability and comprehensive documentation also ensure a smoother development journey, even for complex app functionalities, making

it an ideal choice for developers aiming to deliver high-quality, multi-platform applications efficiently (Amazon Web Services, n.d.).

7.2.1 Main Screen

The main screen serves as the central navigation hub, providing users with intuitive access to the app's core functionalities. It begins with defining a scaffold that structures the main screen, including an app bar titled "COLOR RECOGNITION APP" and a floating action button designed to navigate users to the help screen. The body of the scaffold is strategically organized to present a series of icon buttons, each labelled and dedicated to a specific function of the app—image upload, camera access, colour picking, and theme settings. This layout is achieved through a combination of rows and columns, ensuring that the options are displayed in an intuitive and accessible manner.

7.2.2 Image Upload Screen

With the Image Upload Screen, when a user selects an image, either through capturing a new photo with the camera or uploading one from their photo library, the application utilizes the *ImagePicker* package to handle this action (Flutter API Reference, n.d.). Once an image is chosen, it's stored as a File object, and its bytes are read and decoded into an *img.Image* object from the image library. This conversion is crucial as it transforms the image into a format that allows detailed pixel-level analysis.

The core interaction on this screen involves the user placing a dot on the uploaded image, which serves as a marker for colour extraction. This interaction is captured through a *GestureDetector* that wraps the image display. The *GestureDetector's onPanUpdate* callback function is triggered by user gestures, such as tapping or dragging across the screen (Flutter API Reference, n.d.). Within this call-back, the application calculates the precise location of the dot relative to the image's dimensions, ensuring that the dot's position accurately corresponds to the intended area on the original image.

Upon determining the dot's location, the application then proceeds to extract the colour information of the pixel under the dot. This is achieved by using the *getPixelSafe* method on the *img.Image* object, which returns the colour of the specified pixel in the image. The colour data retrieved is in the form of an integer, which the application then breaks down into its constituent red, green, and blue (RGB) components. These components are essential for both displaying the colour information to the user and converting it into a hex code representation.

This screen also identifies and displays the dominant colours present in the entire image. The dominant colour refers to the hue that appears most prominently in an image, as would be recognized by a human observer (Sightengine, n.d.). This is utilised by the *PaletteGenerator* package, which analyses the image and generates a palette of prominent colours (Flutter API Reference, n.d.). By presenting these colours, the

application offers a broader understanding to the user of the image's colour composition.

7.2.3 Image Capture Screen

The screen initiates the camera functionality using the *ImagePicker* package. This package provides an intuitive interface for accessing the device's camera and photo library (Flutter API Reference, n.d.). When a user tries to take a picture, the *_takePicture* method invokes *ImagePicker.pickImage*, which captures an image using the device's default camera app. The captured image is then stored as an *XFile*, marking the first step in the image processing workflow.

Upon capturing the image, the user is directed to the *DisplayPictureScreen*, where they can crop the image to focus on a specific area, which in this app is a grey reference card. This cropping functionality is achieved through the *ImageCropper* package, which offers customizable cropping tools that can resize the image and the cropping zone (Flutter API Reference, n.d.).

Following the cropping process, the application transitions to the *DisplayCroppedImageScreen*, where the core image analysis and adjustment activities are carried out. An initial step in this phase involves the utilization of the *PaletteGenerator* package to generate a palette of dominant colours from the cropped image (Flutter API Reference, n.d.). This analysis is instrumental in providing a comprehensive understanding of the image's colour composition, laying the groundwork for the critical adjustment phase that follows. The decision to employ the *PaletteGenerator* package for extracting the dominant colour of the cropped image was driven by the need to capture a more accurate overall hue of the image, rather than one colour pixel of the cropped reference.

This approach was particularly justified as testing revealed that, depending on the lighting conditions and various environmental settings, the grey reference card within the image could exhibit varying shades. These shades introduced discrepancies in the RGB values across different parts of the reference card. By focusing on the dominant colour, it aims to reduce these variances, offering a more consistent basis for colour adjustment. This method acknowledges the reality that lighting conditions can significantly influence the perception of colour and that a singular, averaged colour representation can provide a more stable reference point for subsequent image adjustments.

Once the dominant colour is determined, the application proceeds with the colour adjustment process. This involves calculating the difference between the RGB values of the cropped RGB values and the known RGB values of the grey reference card, which is expected to be a neutral grey with equal RGB values. This calculated difference

indicates the deviation caused by lighting conditions and is used to adjust the colours of the entire image.

The adjustment is applied to each pixel in the image. Using the image library for pixel-level manipulation, the application adjusts the RGB values of each pixel to counteract the identified deviation. This systematic approach ensures that the colour adjustment is uniformly applied across the entire image, effectively shifting its overall colour towards the reference hue.

The *image* library plays a pivotal role in this process, enabling pixel-level manipulation of the image (Flutter API Reference, n.d.). The application iterates over each pixel, adjusting its colour based on the previously calculated differences. This meticulous adjustment ensures that the colours of the final image closely match the real-world colours represented by the reference card.

Once the colour adjustment is complete, the application encodes the adjusted image into a Uint8List, facilitating its display within the Flutter UI. The *DisplayCroppedImageScreen* then presents the adjusted image alongside the original for comparison, highlighting the effectiveness of the colour correction process.

Subsequently, the app transitions to the *AdjustedImageScreen*, where it identifies the dot's location, displaying pixel values and dominant colours in a manner consistent with the Image Upload Screen.

In the *AdjustedImageScreen*, I added features to enhance user interaction with colour-adjusted images, incorporating functionality for both viewing and saving these images. This widget stands out by allowing users to toggle between the original and the adjusted versions of an image, providing a direct comparison to observe the colour calibration's effect. This toggle functionality is achieved through a state variable, *_showOriginalImage*, which determines which version of the image, original or adjusted, is displayed. Users can switch between the two states using a button, enabling them to assess the adjustments made instantly.

Moreover, the application includes an image-saving feature. Utilizing the *gallery_saver* package, this functionality allows users to save the adjusted images directly to their device's photo gallery. The process involves generating a temporary image file from the adjusted image's byte data, which is then stored using the device's file system—courtesy of the *path_provider* package. Upon successfully saving the image, the application confirms the action to the user with a notification, enhancing the user experience by providing immediate feedback on the save operation.

7.2.4 Colour Screen

For the Colour Screen, I integrated the *FlexColorPicker* package into the Flutter application, which allowed for a sophisticated yet user-friendly colour selection

interface. The primary objective was to create an interactive and educational experience around colour exploration.

The core of the implementation involved managing the state of the user-selected primary colour. I utilized a stateful widget, *ColorPickerScreen*, to maintain this state. The initial state was set to a default colour, which updates when the user selects a new colour from the colour picker dialog. This approach ensured that the application could reactively display the corresponding colour spectrum, enhancing user engagement.

7.2.5 Theme Screen

In the development of the Theme Screen for a Flutter application, the primary objective was to furnish users with an intuitive and engaging means to toggle between various theme modes, including Dark Mode, Yellow Theme, Red Theme and Green Theme. This functionality was encapsulated within a *StatelessWidget*, which, despite its stateless nature, facilitates dynamic theme changes across the application.

The structure of the Theme Screen is crafted to ensure ease of use and clarity. By employing *SwitchListTile* widgets for each theme option, the screen provides a straightforward interface for users to switch themes. Each switch is associated with a boolean flag indicating its state, active or inactive, and is tied to a specific theme through its *onChanged* call-back, which triggers the corresponding toggle function.

7.3 Testing

The testing with my colour recognition app was done by running the debug version of the app on my phone.

The testing was done divided by each screen, trying to find malfunctioning or situations that weren't expected, to ensure that the app works as designed and implemented.

Starting from the main screen, I have tried accessing other screens, pressing the buttons that lead to other screens, and coming back to the main screen.

7.3.1 Image Capture Screen

The core component of the project was the image capture adjustment screen. This feature was critical because variables such as lighting conditions, shade, light temperature, and other factors could significantly impact the accuracy of the captured image's colour. To address these challenges, I conducted extensive testing under a variety of settings to ensure the highest possible precision in colour representation.

To account for the variables that could affect colour accuracy, my testing strategy focused on various lighting conditions. I began with scenarios featuring normal lighting, progressively introducing additional lamp lights to simulate enhanced illumination.

Subsequently, I adjusted the environment to dimmer lighting conditions, gradually reducing the light until the room was almost completely dark. This methodical approach allowed me to comprehensively evaluate the system's performance across a spectrum of lighting scenarios.

I also conducted tests in various environment settings.

While testing the app with the external reference multiple times, I have realized that when the setting was dark, the RGB values of the reference was a bit higher than it is supposed to be, and when the setting is bright, the RGB values were a bit lower in a few testing cases. This was due to the AWB(auto white balance) of the iPhone Camera app (Depaolo, 2016).

Starting with bright lighting conditions, I anticipated the RGB values of the external reference sheet, specifically my grey reference card, to be at their peak due to the increased illumination. Upon examining the values of the external reference sheet under the brightest setting, the registering RGB values were high with bright lightings. Under dim light settings, the registering RGB values of the reference was low.

Light Setting	Bright	Moderate	Dim	Very Dim
Before Adjustment (RGB)	194,203,208	196,203,213	109,103,89	42,41,36
After Adjustment (RGB)	141,136,133	144,140,137	138,136,137	147,143,142
Actual Reference (RGB)	139,136,135	139,136,135	139,136,135	139,136,135
Cropped reference (RGB)	191,200,207	189,197,208	107,100,82	32,32,26

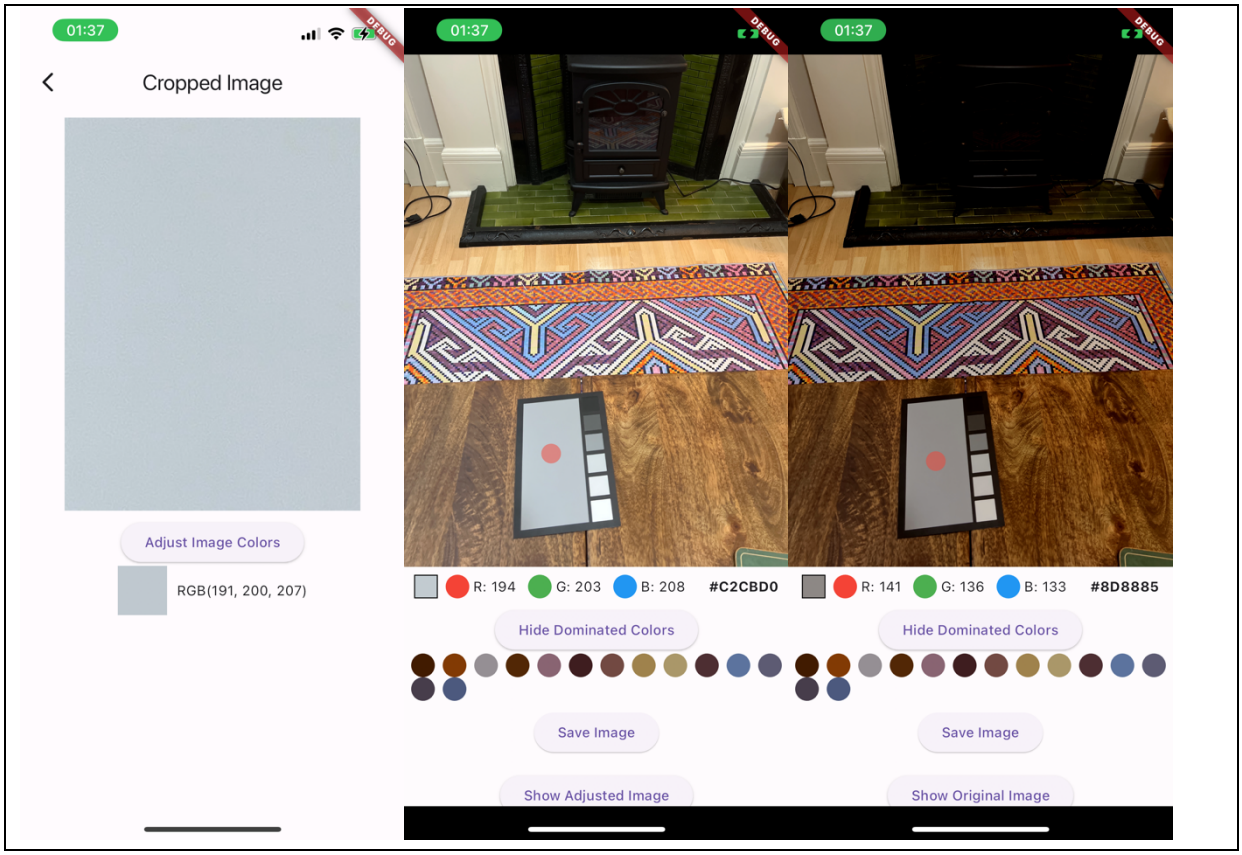


Figure 4 – Bright Lighting

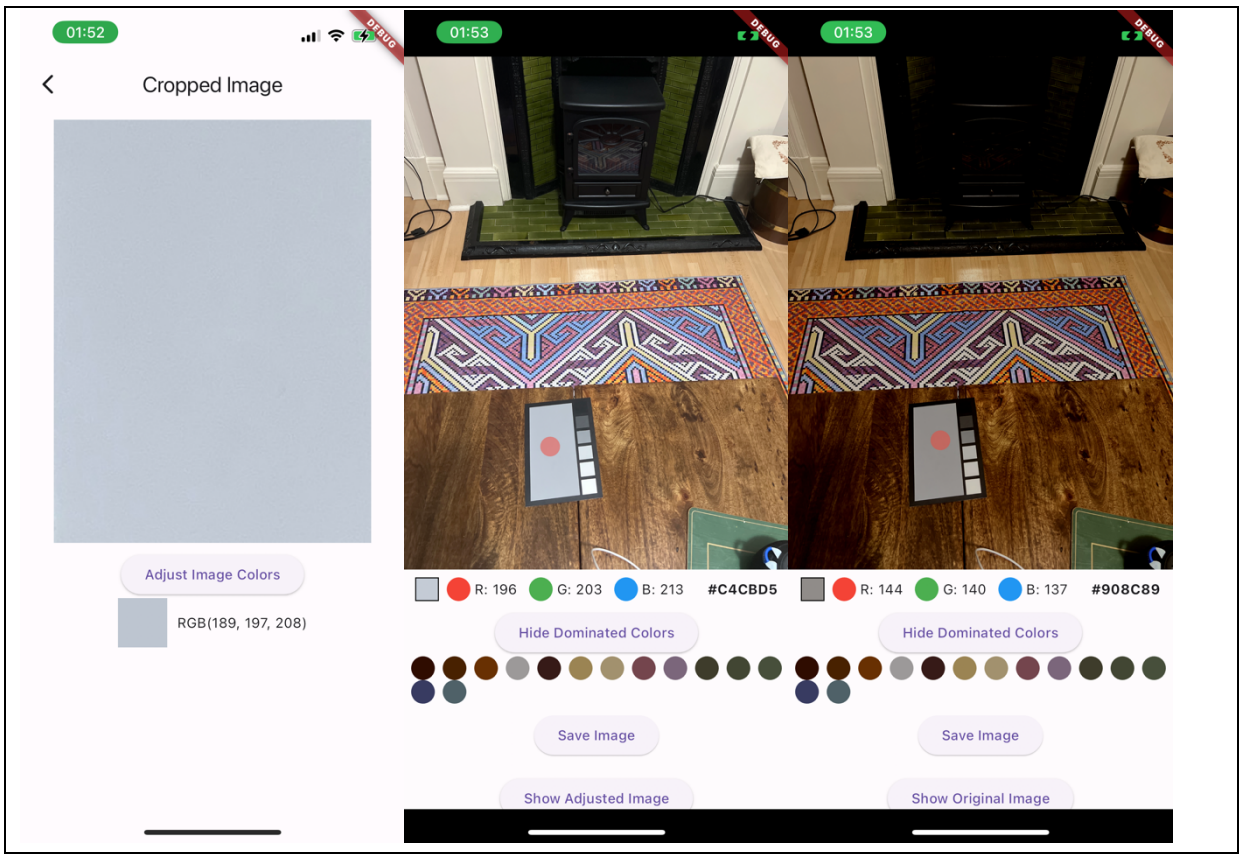


Figure 5 – Moderate Lighting

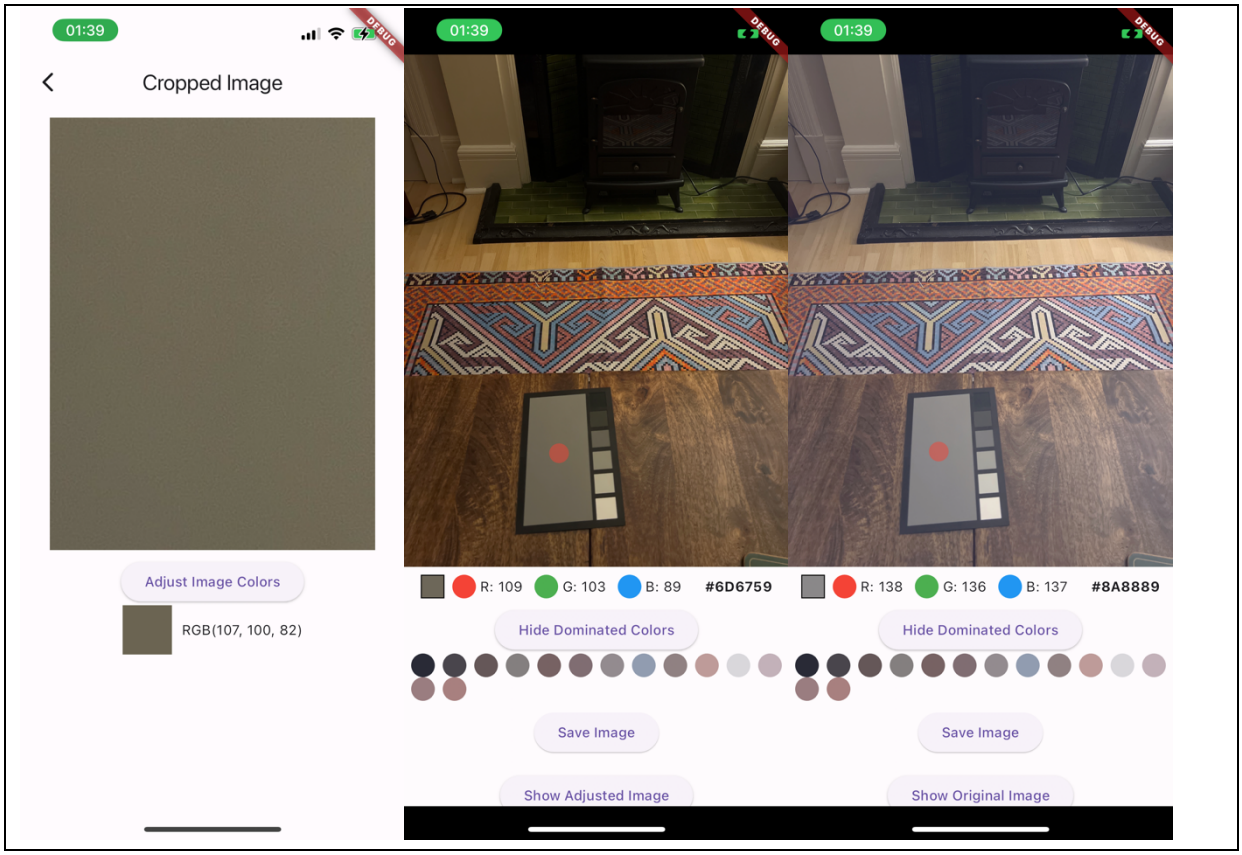


Figure 6 – Dim Lighting

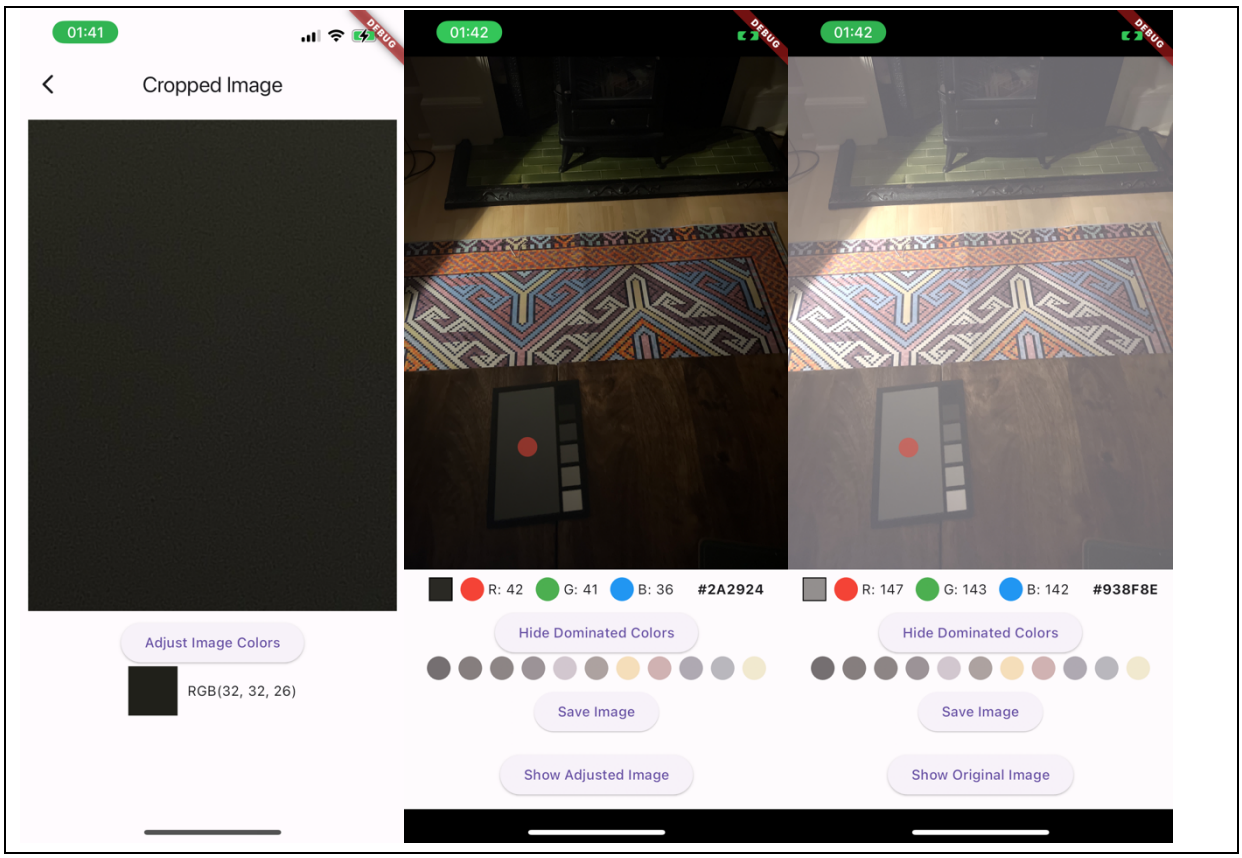


Figure 7 – Dark Lighting

Testing in different Moderate Lighting but different Environments

Light Setting	Library Indoor	Library stairs	Library Outdoor (Night)
Before Adjustment (RGB)	96,96,95	99,99,92	123,115,99
After Adjustment (RGB)	146,142,139	146,142,143	143,138,142
Actual Reference (RGB)	139,136,135	139,136,135	139,136,135

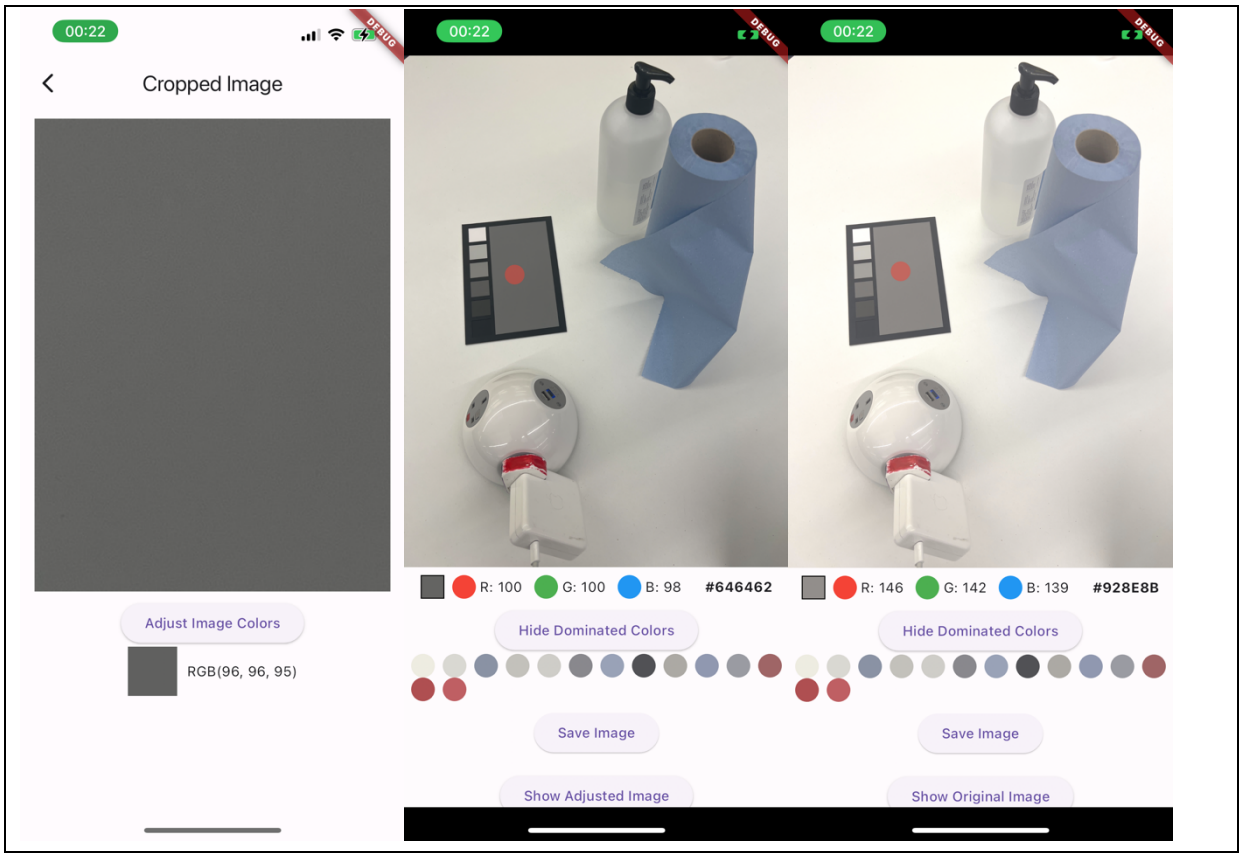


Figure 8 - University of St Andrews Main Library (Indoor)

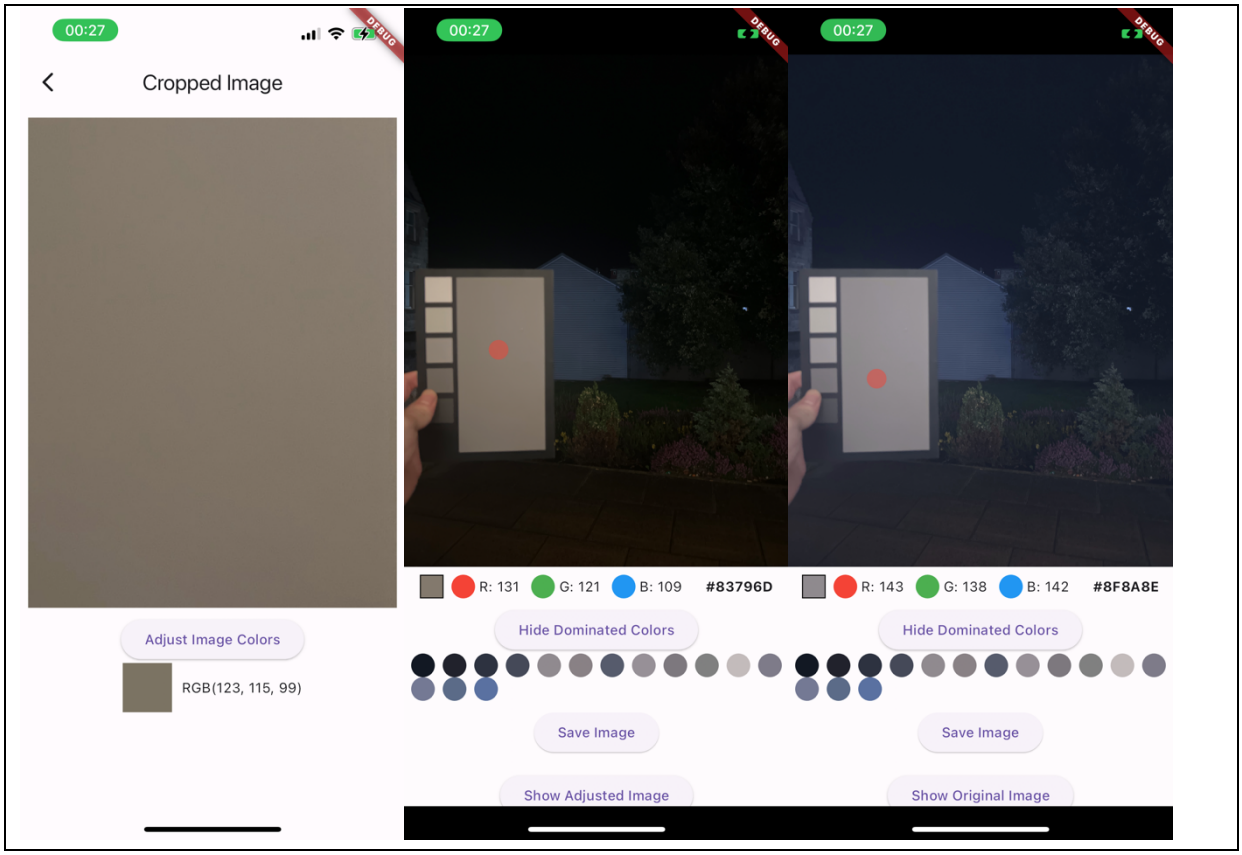


Figure 9 - University of St Andrews Library (Outdoor/Night)

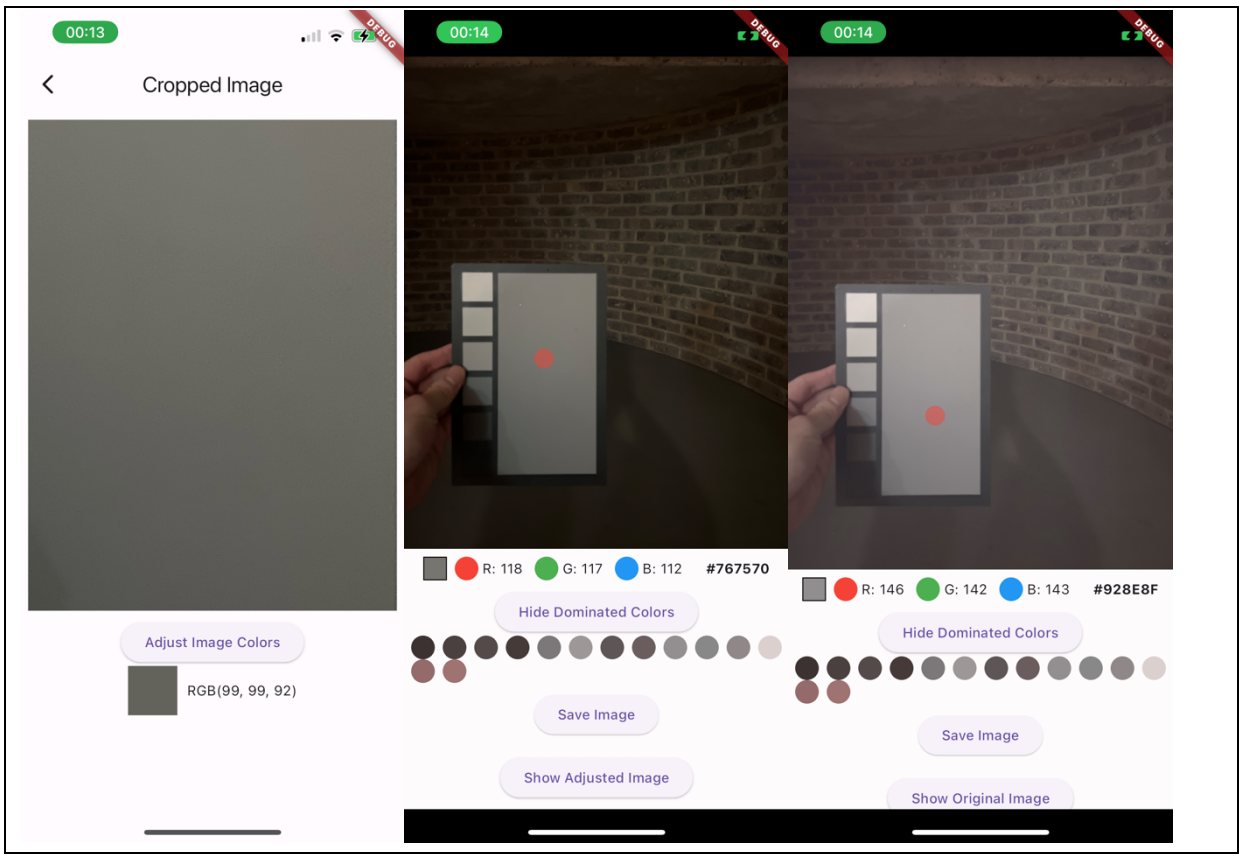
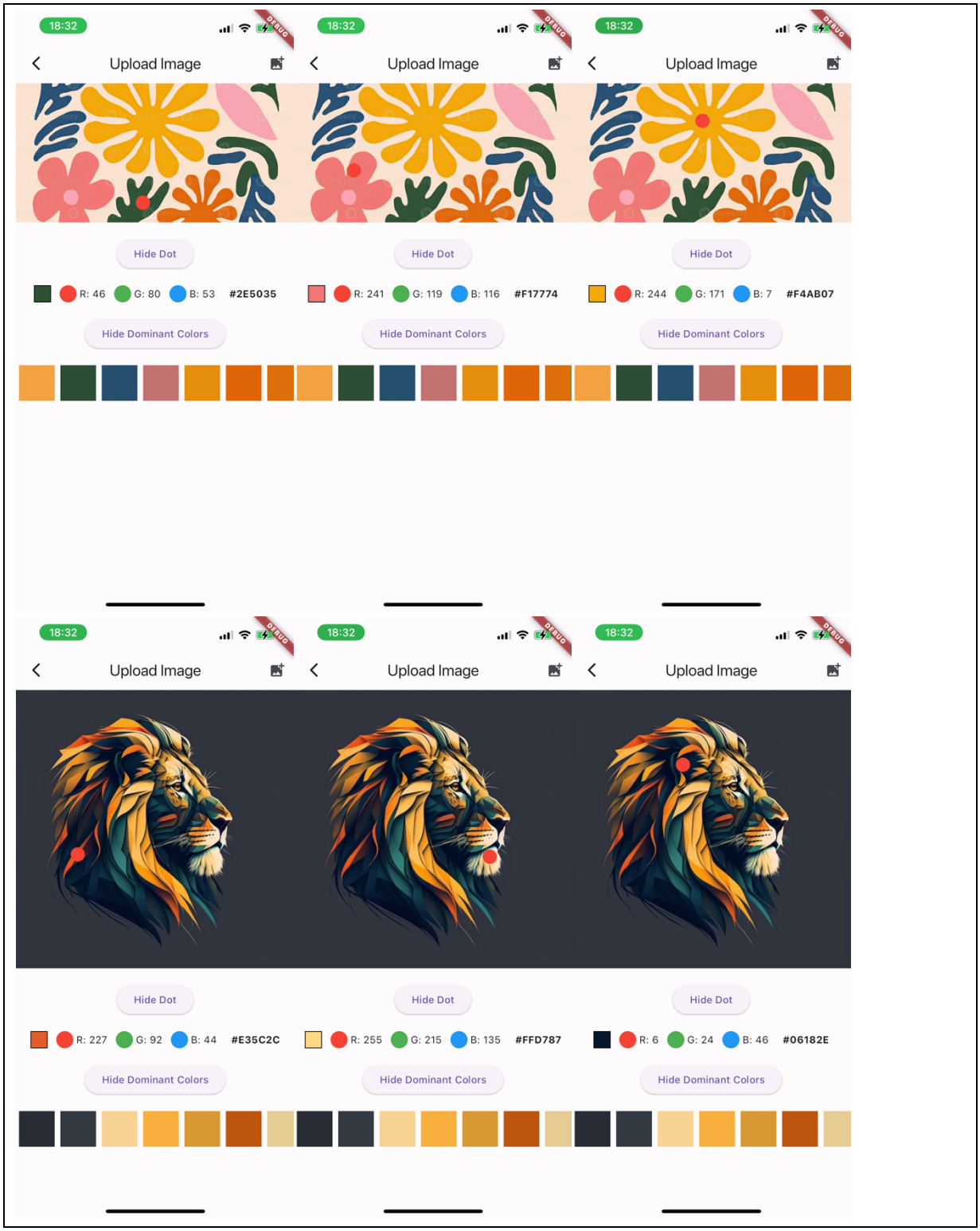


Figure 10 - University of St Andrews Library stairs (very dim lighting)

7.3.2 Image Upload Screen

The image upload screen did not feature a colour calibration function, so testing was conducted using a variety of images. These included photographs taken with a camera and various computer-generated graphics.



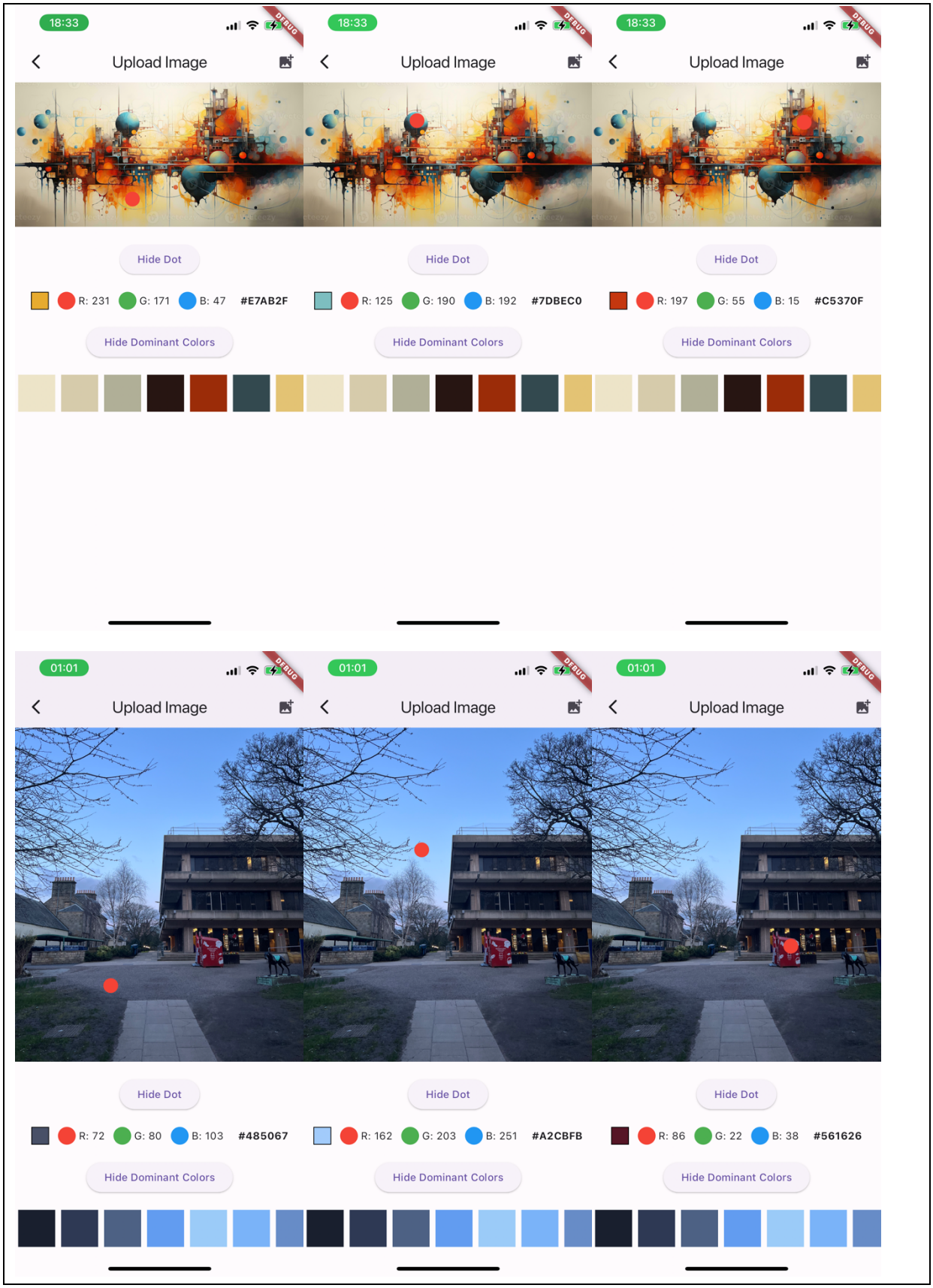


Figure 11,12 – Testing on Image Upload Screen

7.3.3 Theme Screen

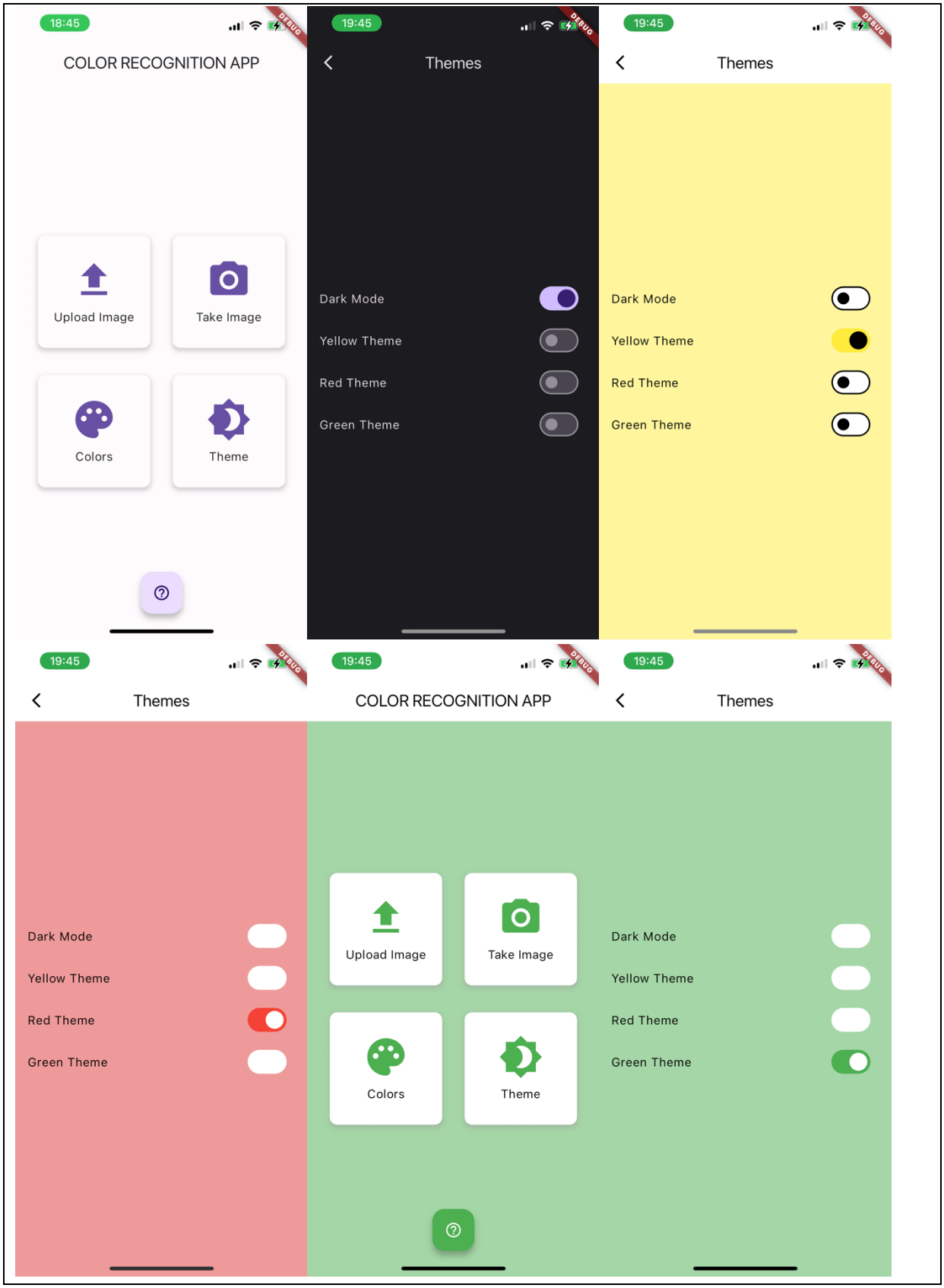


Figure 13 - Testing for theme applied on App

7.3.4 Colour Picker Screen

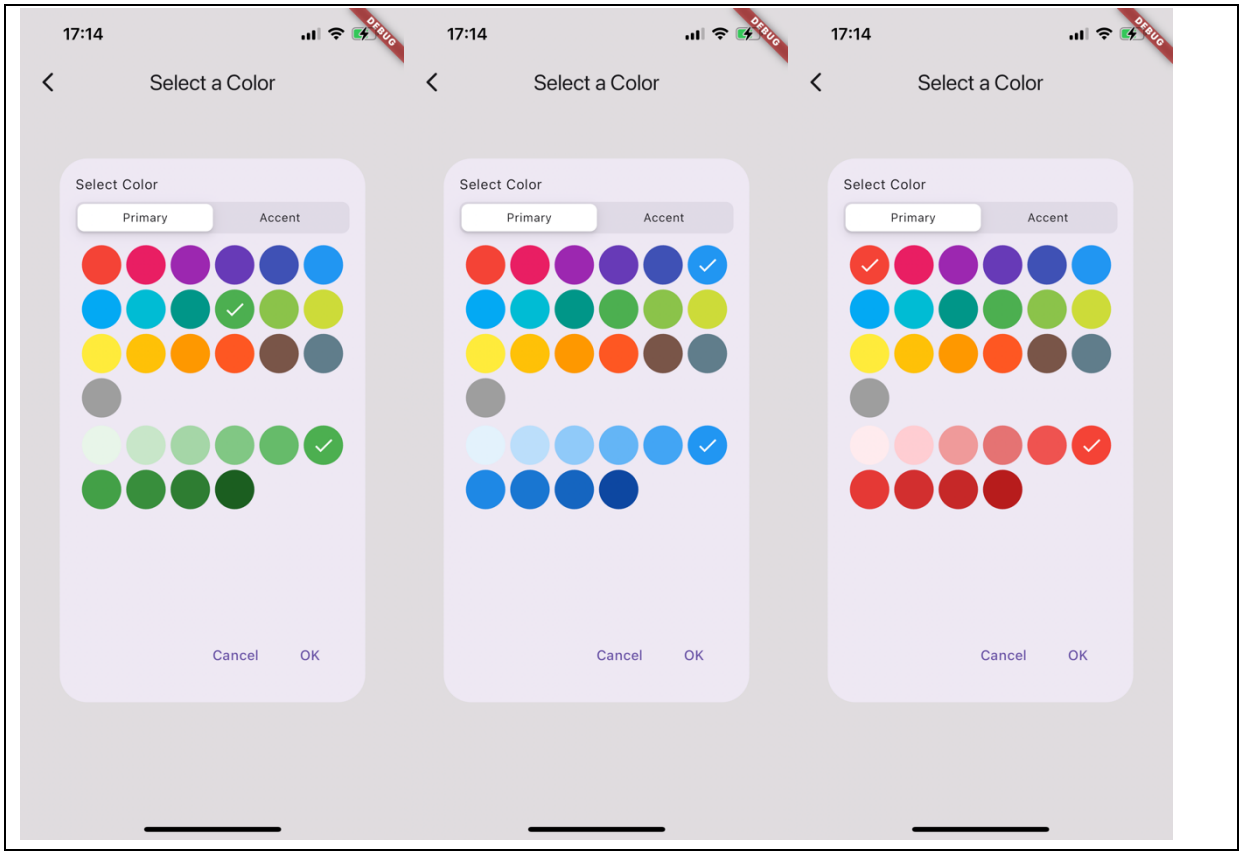


Figure 14 – Testing for Colour picker Screen

8 Evaluation

8.1 Objectives Met & the Drawbacks and Limitations

Based on the preliminary objectives in the course of developing the application, a pivotal aspect I targeted was the accuracy of colour calibration. This process is fundamental to ensure that the colours displayed by the app closely mirror real-world colours. To achieve this, I utilized the Euclidean distance concept to adjust the RGB values of images, comparing them with those of an external grey reference card. This method is recognized for its effectiveness in establishing a baseline for colour accuracy.

However, upon reflection and thorough evaluation, it has become apparent that my calibration process overlooked several essential factors that are critical to comprehensive colour calibration. Initially, while the use of a grey reference card benefits baseline adjustments, it fails to account for the full spectrum of colours and their various intensities.

Furthermore, my method did not incorporate adjustments for different lighting temperatures, which can significantly impact colour perception. The appearance of colours can be altered by both natural and artificial light sources, suggesting that the calibration might not be uniform across different lighting environments. This oversight indicates that the application's colour accuracy might vary in response to ambient light, potentially affecting user experience.

Despite a few adjustments the app itself couldn't accommodate, the built-in camera functionalities of my local device, an iPhone 13 Pro, for adjusting various lighting settings, including white balance, as well as accounting for images of objects from different angles, have proven to be effective. The iPhone Cameras have AWB(Auto White Balance) (Depaolo, 2016). Despite this success, it's important to note that the app could benefit from testing on a wider range of devices, especially those with fewer functionalities or lower camera quality, a factor that has not been considered thus far. Also, despite the auto white balancing of iPhone cameras, as shown in my testing, because of this function, depending on the lighting and the expected RGB values of the grey reference card wasn't exactly as expected.

Although, the image itself calibration wasn't exactly as expected, extracting the RGB values from the image was precise based on the testing.

In conclusion, while my initial approach to colour calibration using the Euclidean distance concept and a grey reference card laid down a fundamental basis for colour accuracy, it missed key elements necessary for a comprehensive colour calibration strategy. Going forward, it will be imperative to address these gaps by incorporating adjustments for colour intensity, lighting conditions, and device-specific colour gamuts.

In developing my colour recognition app with Flutter, I aimed to enhance its performance by adding advanced image processing features because Flutter itself lacked quite a lot of libraries and packages that directly dealt with advanced image processing. I planned to use OpenCV for its comprehensive image processing tools and GPUImage for its real-time filters and effects. The main goal was to improve the app's white balancing, adjust image temperatures, and change tints to ensure accurate colour recognition under different lighting conditions. However, I encountered unexpected challenges with setup and compatibility, leading me to leave out these features in the final version.

As mentioned, the way this app has calibrated the image was not the most precise way as various photoshop softwares such as Adobe, Lightroom and more. However, without using external libraries or packages that included the direct colour correction of the images, I have managed to find a way of my best capabilities to calibrate the colours using the platform of my choice, and an external reference of use.

For the user-friendly interface objective, to ensure the app was accessible to a broad audience, including those with limited technical expertise, I focused on designing a user-friendly interface. This was achieved by developing a simple and intuitive user interface (UI) and simplifying the calibration process as much as possible. By doing so, I made it easy for users to navigate the app and utilize its features without facing any complexities.

To enhance the accuracy and reliability of the app's colour recognition capabilities, I adopted a methodical approach to testing and evaluation during the development process. This involved building up the app's functionalities incrementally. I started by implementing one function at a time, conducting thorough testing to ensure its performance met the desired standards. Once a function was successfully tested and confirmed to work as intended, I proceeded to add another function. This iterative process of building, testing, and refining allowed me to systematically improve the app's performance, ensuring each added feature contributed positively to its overall accuracy and reliability in colour recognition.

In the process of enhancing our Flutter application with advanced image processing capabilities, I ventured into integrating the GPUImage library, a task that required bridging native iOS functionality with Flutter's cross-platform environment. Despite successfully understanding the integration process and establishing the necessary bridge, I encountered configuration issues that prevented GPUImage's operational implementation within our application. Nevertheless, I have included the relevant code within the submitted project to demonstrate the effort and groundwork laid towards achieving this enhancement (files AppDelegate.swift, OpenCV.swift, ImageProcesseor.dart, ImageProcessorChannel.dart).

9 Conclusion

In this dissertation, I explored the development and evaluation of a colour recognition application designed using the Flutter framework. The goal was to leverage Flutter's cross-platform capabilities to create a responsive and user-friendly app capable of accurately identifying colours in images captured by or uploaded to mobile devices. This endeavour was motivated by various practical needs, such as aiding colour-blind individuals and assisting in colour identification.

Throughout the project, I employed comprehensive research methodologies that included a thorough analysis of colour recognition technologies, an exploration of Flutter's development environment, and the integration of colour analysis methods. The

app's development was characterized by the incorporation of camera and photo library accessibilities, allowing users to easily select images for colour identification. The display of hexadecimal codes, RGB values and dominant colours upon recognition proved to be invaluable for a wide range of applications.

The evaluation phase of our project utilized both quantitative and qualitative methods, including accuracy testing against a predefined colour dataset and user experience. The results demonstrated accuracy in colour calibration based on the hue and brightness but lacked a few important factors. Despite encountering challenges such as the integration of advanced image processing features and compatibility issues across different platforms, I managed to navigate these obstacles and deliver a functional app.

One of the most significant learnings from this project is the importance of considering various factors in colour calibration, such as lighting conditions and device-specific colour spectrums. Although our initial approach provided a solid foundation for colour accuracy, it highlighted the need for a more comprehensive strategy that encompasses these elements.

This dissertation has not only contributed to a functional colour recognition application but also laid the groundwork for future enhancements. Our journey through the development and evaluation processes has illuminated the challenges and possibilities inherent in creating such tools, offering valuable insights for both current and future projects in this domain.

10 Bibliography

White balance. (n.d.). Cambridge in Colour. Available at: https://www.cambridgeincolour.com/tutorials/white-balance.htm (Accessed: [7 March, 2024]).

Sorathiya, N. (2024). Optimizing user Experience: Integrating Flutter Camera Package in Apps. Available at: https://www.dhiwise.com/post/optimizing-user-experience-integrating-flutter-camera-package (Accessed: [7 March, 2024]).

Flickinger, L. (2013). 6 Easy Methods to Achieve Perfect White Balance Every Time. Click it Up a Notch. Available at: https://clickitupanotch.com/white-balance-comparing-different-methods/ (Accessed: [22 Feb, 2024]).

Mastin Labs. (n.d.). How to shoot in Kelvin. Available at: https://mastinlabs.com/blogs/photoism/how-to-shoot-in-kelvin (Accessed: [5 March, 2024]).

Perez, F. Photography White Balance: A beginners Guide. Available at: https://www.picturecorrect.com/photography-white-balance-a-beginners-guide/ (Accessed: [Date]).

Rasrogi, K. All about White Balance. Colour Temperature & Tint in Photography. The Photo Teacher. Available at: https://thephototeacher.wordpress.com/2017/01/13/all-about-white-balance-colour-temperature-tint-in-photography/ (Accessed: [6 March, 2024]).

Grace, F., Lewis, M. What is the Kelvin Scale? The ultimate guide to every Colour Temperature. Available at: https://www.shutterstock.com/blog/kelvin-scale-breakdown-colour-temperature (Accessed: [13 March, 2024]).

GeeksforGeeks (2023). How to create image cropper app in Flutter. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/how-to-create-image-cropper-app-in-flutter (Accessed: [20 Feb, 2024]).

Chathuranjana, P. (2023) Integrating OpenCV to your Swift IOS project in Xcode and working with UIImages. Medium. Available at: https://medium.com/@hdpoorna/integrating-opencv-to-your-swift-ios-project-in-xcode-and-working-with-uiimages-4c614e62ac88 (Accessed: [20 Feb, 2024]).

Flutter API Reference Documentation. *Flutter*. Available at: https://api.flutter.dev/index.html (Accessed: [20 Feb, 2024]).

SpyderCheckr User's Guide. (n.d.). SpyderCheckr24. Available at: https://www.bhphotovideo.com/lit_files/381135.pdf (Accessed:[5 March, 2024]).

Amazon Web Services (n.d.). What is Flutter?. Available at: https://aws.amazon.com/what-is/flutter/ (Accessed: [15 March, 2024]).

Wolfram Research. (2014). ColourDistance, Wolfram Language function. Available at: https://reference.wolfram.com/language/ref/ColourDistance.html (Accessed: [10 March, 2024]).

Strickland, JR. (2021). What is Colour Correction and Why is it Important?. VideoMaker. Available at: https://www.videomaker.com/how-to/editing/colour-correction/everything-you-need-to-know-about-colour-correction/ (Accessed: [10 March, 2024]).

Lea,V., Chris, L. (n.d.). Euclidean Distance. Color Differences. Available at: https://colorjs.io/docs/color-difference (Accessed: [10 March, 2024]).

Depaolo, R. (2016). 6 Advanced iPhone Camera Controls for Jaw-Dropping Photography. iPhone Photography School. Available at: https://iphonephotographyschool.com/iphone-camera-controls/ (Accessed: [10 March, 2024])

M. Samara, J. AlSadah, M. Driche and Y. Osais. (2017) "A color recognition system for the visually impaired people," *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Salmabad, Bahrain, 2017, pp. 1-5 (Accessed: [10 March, 2024])

Barnhart, B. (2023). Color tone terminology handbook: tint, tone, shade, and more. Available at: https://www.linearity.io/blog/color-tone-terminology/ (Accessed: [15 March, 2024]).

Kirvan, P. (2022). Hue, saturation and brightness. WhatIS?. Available at: https://www.techtarget.com/whatis/definition/hue-saturation-and-brightness (Accessed: [15 March, 2024]).

Adobe. (2022). Adjust color, saturation, and hue. Available at: https://helpx.adobe.com/ie/photoshop-elements/using/adjusting-color-saturation-hue-vibrance.html (Accessed: [15 March, 2024]).

Pro Edu. (2024). Using a Gray Card for Perfect White Balance. Available at: https://proedu.com/blogs/photography-fundamentals/using-a-gray-card-for-perfect-white-balance (Accessed:[20 March, 2024]).

Nabeel. (2021). The Best Image Recognition apps that you should try in 2024. TrekRevol. Available at: https://www.tekrevol.com/blogs/best-image-recognition-apps/ (Accessed:[20 March, 2024]).

Sightengine. (n.d.). Color Detection. Sightengine. Available at: https://sightengine.com/docs/color-detection (Accessed: [22 March, 2024])

11 Appendicies

11.1 Ethics Form

UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
PRELIMINARY ETHICS SELF-ASSESSMENT FORM

This Preliminary Ethics Self-Assessment Form is to be conducted by the researcher, and completed in conjunction with the Guidelines for Ethical Research Practice. All staff and students of the School of Computer Science must complete it prior to commencing research.

This Form will act as a formal record of your ethical considerations.
Tick one box

Staff Project

Postgraduate Project

✓ **Undergraduate Project**

Title of project

What is the colour?

Name of researcher(s)

Jae Woo Chang

Name of supervisor (for student research)

Saleem Bhatti

OVERALL ASSESSMENT (to be signed after questions, overleaf, have been completed)

Self audit has been conducted **YES** **NO**

There are no ethical issues raised by this project

Signature Student or Researcher

Print Name Jae Woo Chang

Date 29/09/2023

Signature Lead Researcher or Supervisor

Print Name Saleem Bhatti

Date 02 Oct 2023

This form must be date stamped and held in the files of the Lead Researcher or Supervisor. If fieldwork is required, a copy must also be lodged with appropriate Risk Assessment forms. The School Ethics Committee will be responsible for monitoring assessments.

Computer Science Preliminary Ethics Self-Assessment Form

Research with secondary datasets

Please check UTREC guidance on secondary datasets (https://www.st-andrews.ac.uk/research/integrity-ethics/humans/ethical-guidance/secondary-data/ and https://www.st-andrews.ac.uk/research/integrity-ethics/humans/ethical-guidance/confidentiality-data-protection/). Based on the guidance, does your project need ethics approval?

YES **NO**

** If your research involves secondary datasets, please list them with links in DOER.*

Research with human subjects

Does your research involve collecting personal data on human subjects?

YES **NO**

If YES, full ethics review required

Does your research involve human subjects or have potential adverse consequences for human welfare and wellbeing?

YES **NO**

If YES, full ethics review required
For example:
Will you be surveying, observing or interviewing human subjects?
Does your research have the potential to have a significant negative effect on people in the study area?

Potential physical or psychological harm, discomfort or stress

Are there any foreseeable risks to the researcher, or to any participants in this research?

YES **NO**

If YES, full ethics review required
For example:
Is there any potential that there could be physical harm for anyone involved in the research?
Is there any potential for psychological harm, discomfort or stress for anyone involved in the research?

Conflicts of interest

Do any conflicts of interest arise?

YES **NO**

If YES, full ethics review required
For example:
Might research objectivity be compromised by sponsorship?
Might any issues of intellectual property or roles in research be raised?

Funding

Is your research funded externally?

YES NO

If YES, does the funder appear on the 'currently automatically approved' list on the UTREC website?

YES NO

If NO, you will need to submit a Funding Approval Application as per instructions on the UTREC website.

Research with animals

Does your research involve the use of living animals?

YES NO

If YES, your proposal must be referred to the University's Animal Welfare and Ethics Committee (AWEC)

University Teaching and Research Ethics Committee (UTREC) pages
http://www.st-andrews.ac.uk/utrec/