Team Name: ColourBlind

Proposed Level of Achievement: Gemini

Motivation:

Typical resistors have many different coloured bands as a form of identification of how much resistance the resistor has. Whilst the majority of the population has no problem identifying the colours, they have a hard time memorising the meanings of the colours of the band as well as the position of the band itself. But that is not the real problem. The real problem is that there is a sub-population which has a legitimate problem identifying the colours on the resistor. This population is the colour-deficient population where they are unable to differentiate between colours easily. This is a major inconvenience to them as they have to take extra steps to figure out the resistor. In the worst case scenario, they may select the wrong resistor and cause a malfunction in the circuit that they are designing. Thus, there is a need for a system which allows a simple yet effective identification of resistor values.

Aim:

We hope to create an app which is able to identify resistor values by observing the resistor through a phone's camera.

User Stories:

1. As a student/ designer, I want to be able to almost instantly identify the resistor value of my resistors. I also wish to be able to identify the resistor value in different places.
2. As a person with colour-deficiency, I want to be able to rely on this app to identify the correct resistor value.

Technology Stack

1. Python
2. OpenCV
3. Java
4. Android Studio

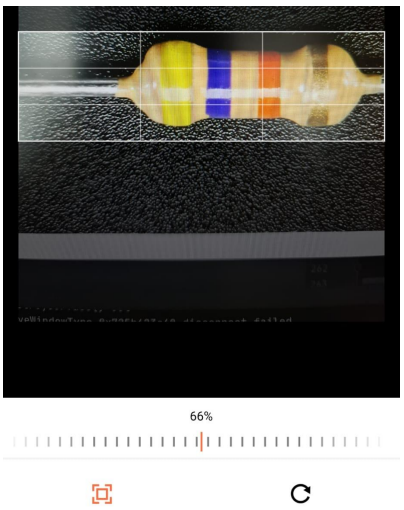
Features:

The **app** has an interface that resembles a phone camera app. Once the resistor is in view and the main push button is pressed, the image of the resistor is captured and analysed. The app also has a reference to the resistor value calculation system for referral. The app also includes a manual input option where the user can input the colours.

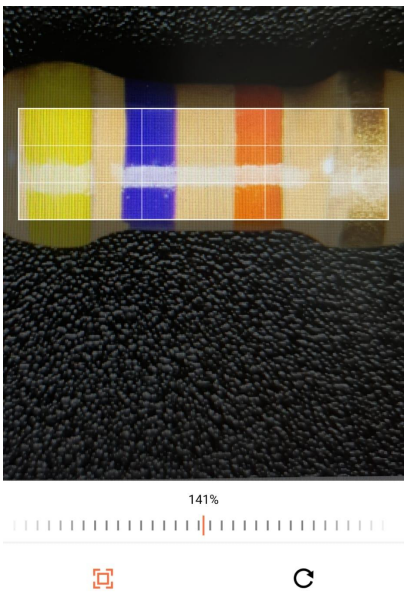
The **image analyser** will take in the image of the resistor and output the resistor value of the resistor at the bottom of the image.

The process of image analysis is broken down into 4 steps.
Step 1: Cropping Resistor (Manually)

The user has to take a photo of a resistor and manually crop the resistor such that the background is completely removed.



→



Step 2: Colour Quantization

There is an option to enable white balancing and quantization. Quantize the image to reduce the number of colours that are used to represent the colour bands. Results vary. For this, we are using opencv's k means algorithm.

→
Quantizing

Step 3: Body Colour Filtering

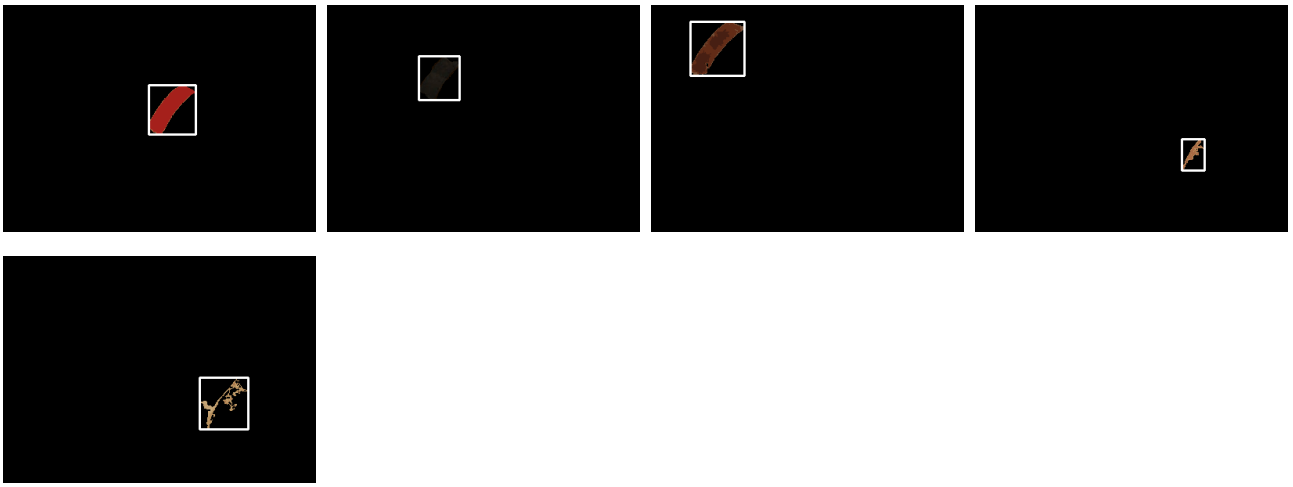
The body of the resistor can affect the colour detection process, so to improve accuracy, we use openCV's adaptiveThresholding to remove it.

→
adaptiveThreshold

Step 4: Colour Detection

We create masks for each hsv boundary (colour boundaries) and find contours with areas that exceed some threshold value and save the bounding box coordinates of that contour and its corresponding colour. To avoid double counting, if the bounding boxes overlap and are the same colour, keep only one of them.The results will be the colour bands. For

starters, we assume that we are only dealing with resistors that only have 5% or 10% tolerance which means that the last band will always be either gold or silver. From that we can get the order of colours by comparing the coordinates of the bounding boxes. To extend it into a general resistor with different tolerance levels, we will have to assume for now that the orientation of the resistor is correct. That is, the algorithm will read and calculate the colour bands in the same way as how one would read off a resistor in real life, from left to right.



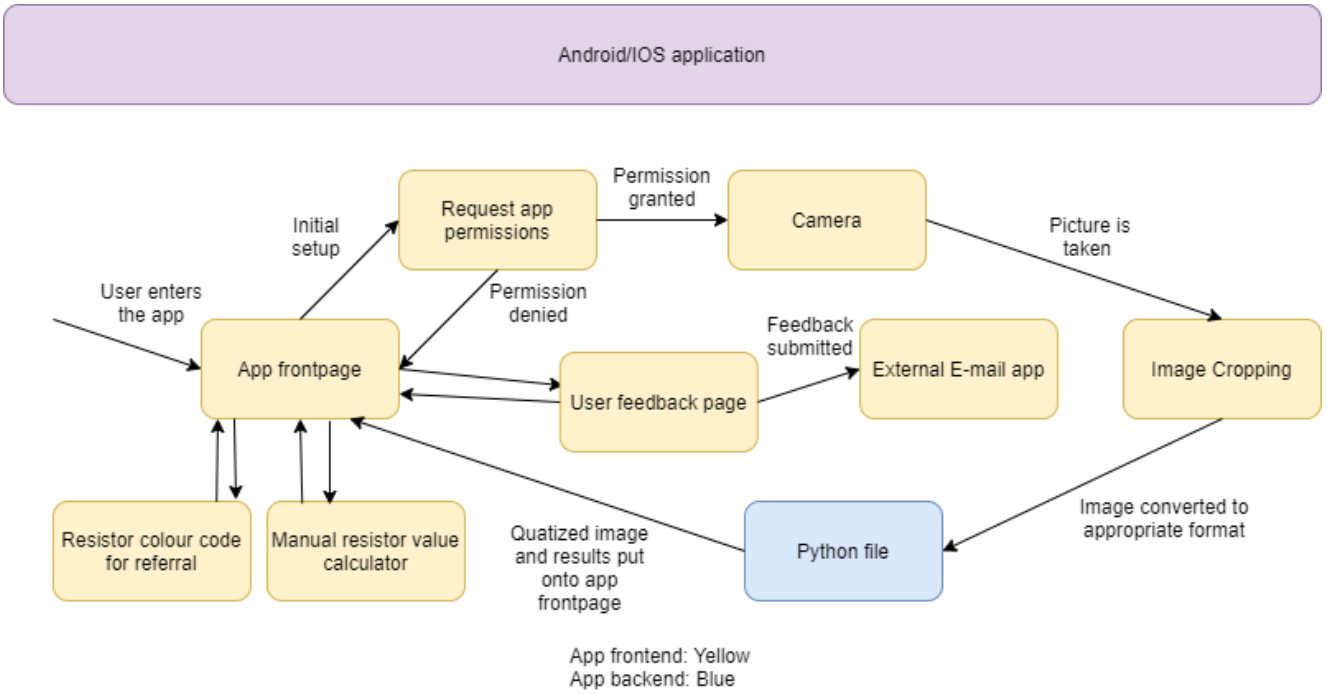
Result:
Number of colour bands detected: 4
Sorted Position and Colours: [((50, 21), 'brown', 4692), ((118, 65), 'black', 2856), ((186, 101), 'red', 3904), ((251, 155), 'gold', 4092)]

In this example, orange is misdetected at the gold's position. Our resolution to the problem of detecting multiple colours for the same band is comparing the areas of the bounding box of the contour that occupy similar positions and take the one with the larger area. In this case, the area of the bounding box for gold is greater than orange so orange is discarded.

Step 4: Resistance Calculation
From the order of colours, and the resistor colour code acting as a check, the resistor value can be calculated.
Resistance: 1.00 kOhms LB: 950.00 Ohms UB: 1.05 kOhms

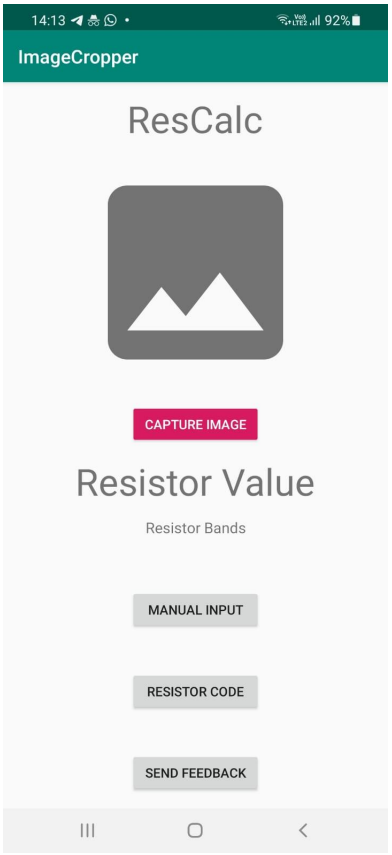
This entire process is done in the app's backend.

Program flow:

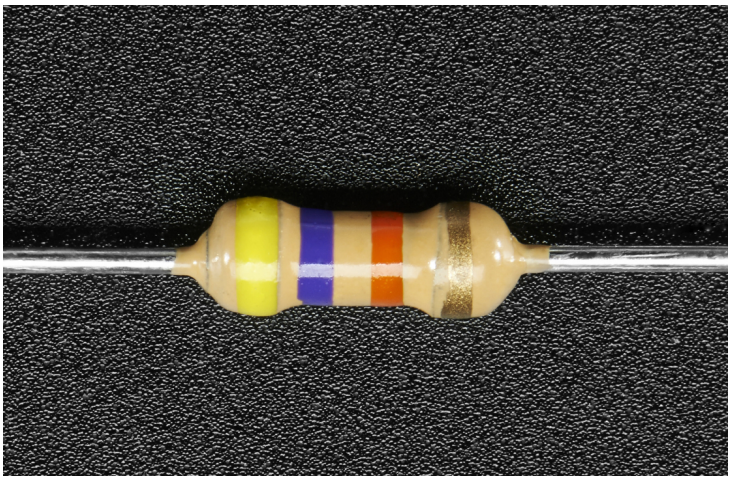


App layout:

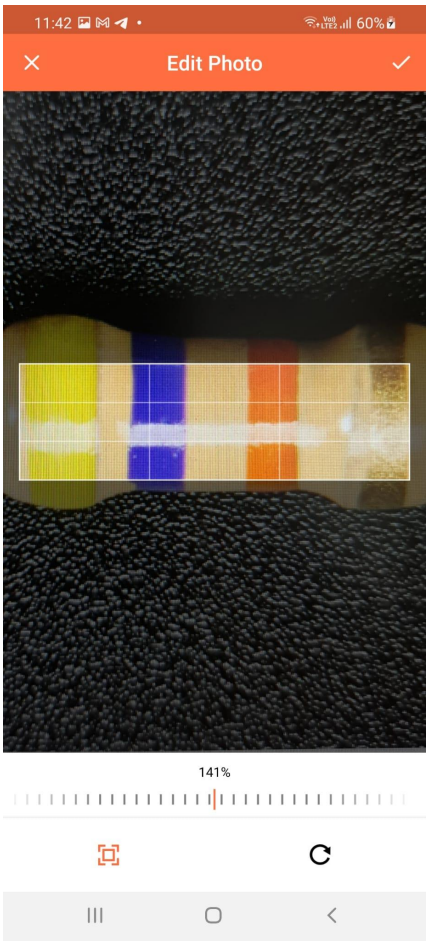
- **Default page:**



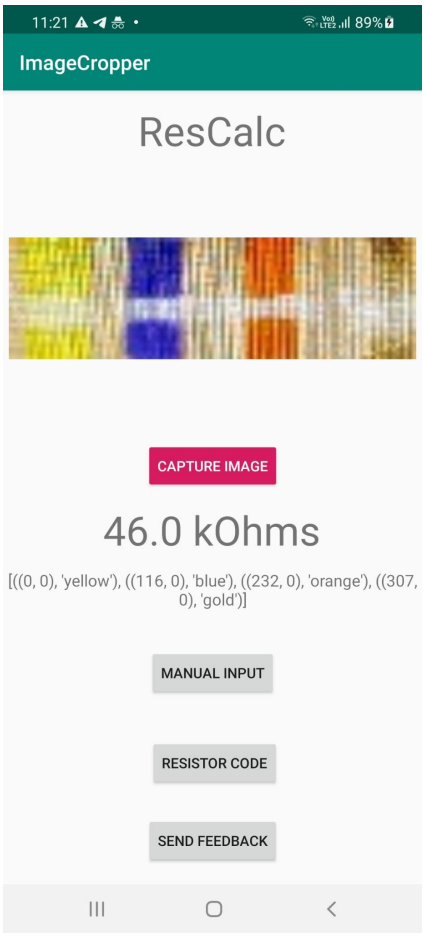
- **Sample image:**



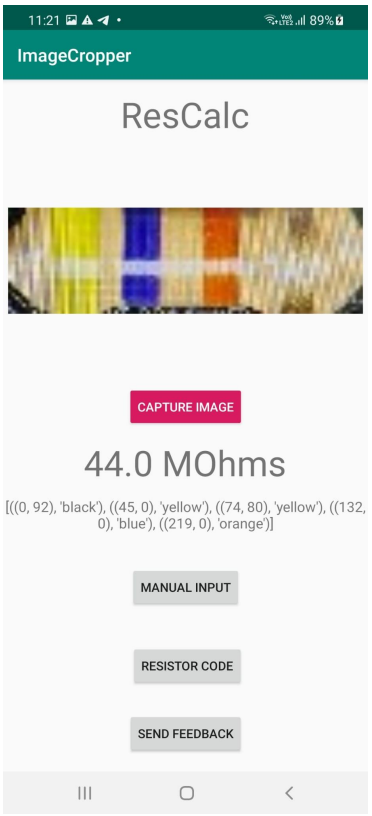
- **Image cropping page:**



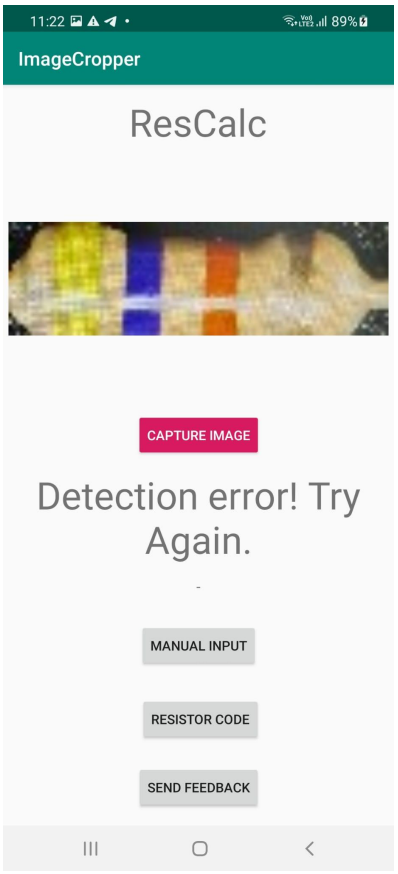
- **After image processing, upon correct identification:**



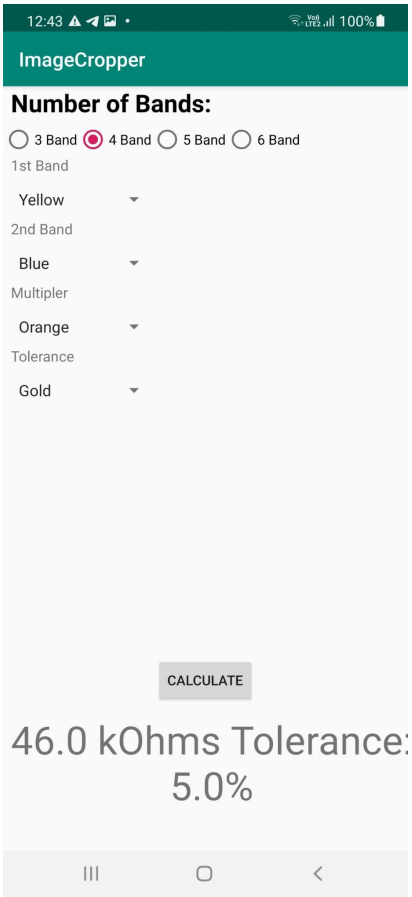
- **After image processing, upon incorrect identification:**



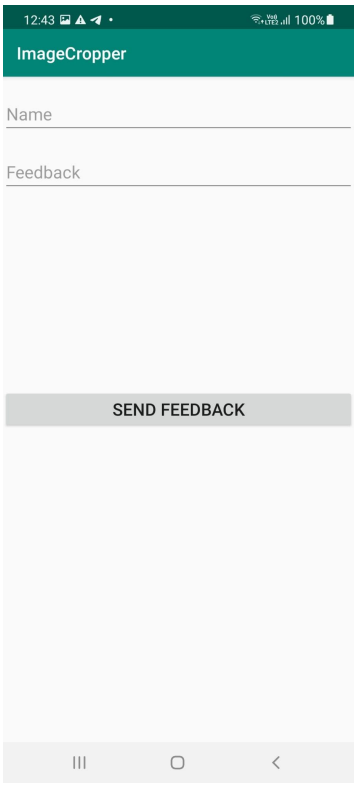
- **After image processing, upon no identification:**



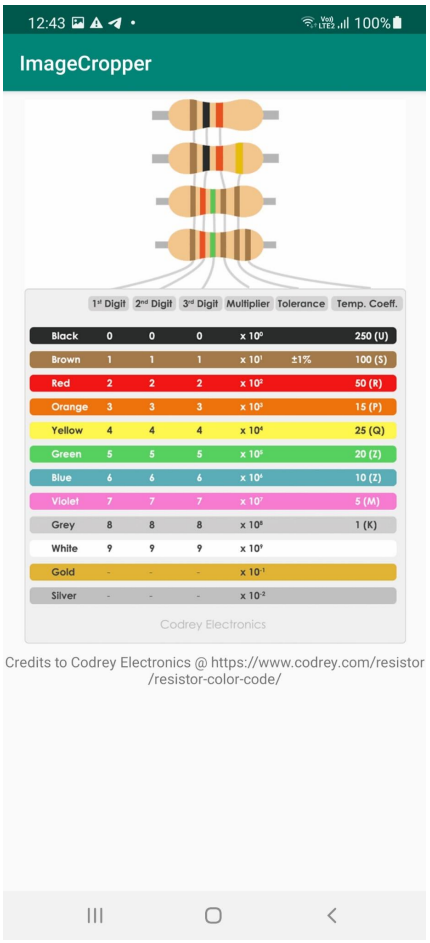
- **Manual resistor calculator:**



- **Feedback form:**



- **Resistor code reference:**



Credits to Codrey Electronics @ https://www.codrey.com/resistor/resistor-color-code/

Progress:

The current iteration of the app is able to identify the colour of the bands of the resistor relatively well. The app has been downsized considerably (current size is 120mb from 0.99Gb) and the processing time has also decreased (from 2 minutes to 2 seconds).

Instead of using the tensorflow model to isolate the resistor from the input image, we integrated an image cropping library instead so that users can identify the resistor for the app.

We also added extra features that might prove useful for users, such as a manual resistor value calculator, a feedback form and a reference to the resistor value code.

The downside to this approach is that a correct result is more dependent on how the user captures and crop the image.

Furthermore, the cropping library has a limited zoom capability, thus the user may need to zoom in on the camera preview page, which is just before the image cropping page.

Software Testing:

We let a few friends try our app and they identified several accidental bugs. For example, we initially had a button to bring the user back to the homepage from the other pages. After the app identifies a resistor and the results are displayed on the homepage, if the user goes into a new page (for example the feedback form), upon the return to the homepage, the results are gone. To remedy this, we just deleted the button since users could go back to the homepage already using the android default back button.

Multiple images of resistors were used as trial tests to see if the app could detect the colours successfully.

Moving forward:

We will attempt to convert the post and pre image processing processes into the native java language. We will also attempt to replace the many buttons into one.

Development Plan:

Dates	Goal	Remarks
17/05 - 31/05	1. Implement App Interface 2. Testing/Training different image recognition methods and deciding the best option	For 2., looking at different RCNNs to work with.
01/06 - 31/06	1. Optimising the image recognition process so that it can be done on a phone 2. Colour Detection of the resistor bands 3. Integration of frontend and backend	1. Includes tuning hyperparameters, maybe converting to tflite
01/07 - 31/07	1. Implement resistor colour code reference page and feedback page 2. Improve image processing duration and accuracy	