Calvin Barker

G. Bishop

COMP 580

8 March 2017

Color Helper Detailed Proposal

Key Idea

Some apps already exist that help people who are colorblind identify colors, but most of these aren't free, and the ones that are don't provide support for easily comparing colors from two different images. The main idea is that I will develop an Android application that will enable people to see color names from one picture in one half of a phone screen while the other half of the screen can hold a different picture and allow for quick and simple classifications and comparisons of colors between the two images.

Target Users

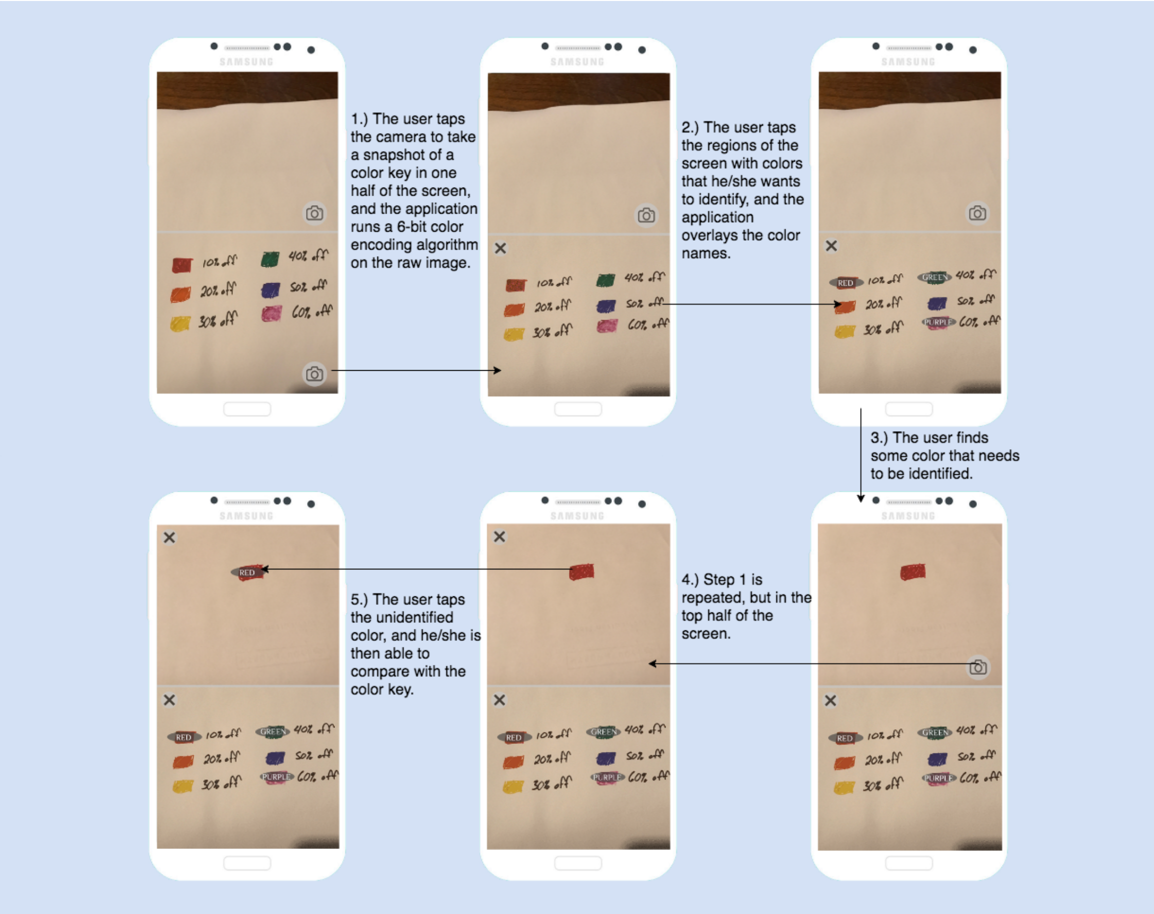
The initial targeted users of this application will be people who are colorblind or have some form of color vision deficiency, and according to the National Institutes of Health, this is approximately 8 percent of men and 0.5 percent of women with Northern European ancestry.[1] Given that this is pretty substantial portion of the population, potentially hundreds of thousands, if not the lower end of millions, of people could benefit from this application.

Originally, though, this idea was inspired by my father who is red-green colorblind. One of his hobbies is home improvement, and he enjoys shopping at Habitat for Humanity ReStore centers. However, one of his pains is that they have different sales for different

[1] Facts About Color Blindness. (n.d.). Retrieved March 07, 2017, from https://nei.nih.gov/health/color_blindness/facts_about

items on different days, and these sales are advertised using colored tags with their values on a poster board at the front of the store. Being colorblind, he's told me it can be challenging and rather frustrating to try to figure out how much an item is actually going to cost without getting help from staff.

What my dad wants is an app that will let him take a picture of the sign at the front of the store with the sale values, tap the colors to show their names on the screen, and then take pictures of item tags to see where they lie on the color key (see Figure 1.) Ultimately, my primary target user is going to be my dad, and I hope to use him extensively during trial runs of my app.



1.) The user taps the camera to take a snapshot of a color key in one half of the screen, and the application runs a 6-bit color encoding algorithm on the raw image.

2.) The user taps the regions of the screen with colors that he/she wants to identify, and the application overlays the color names.

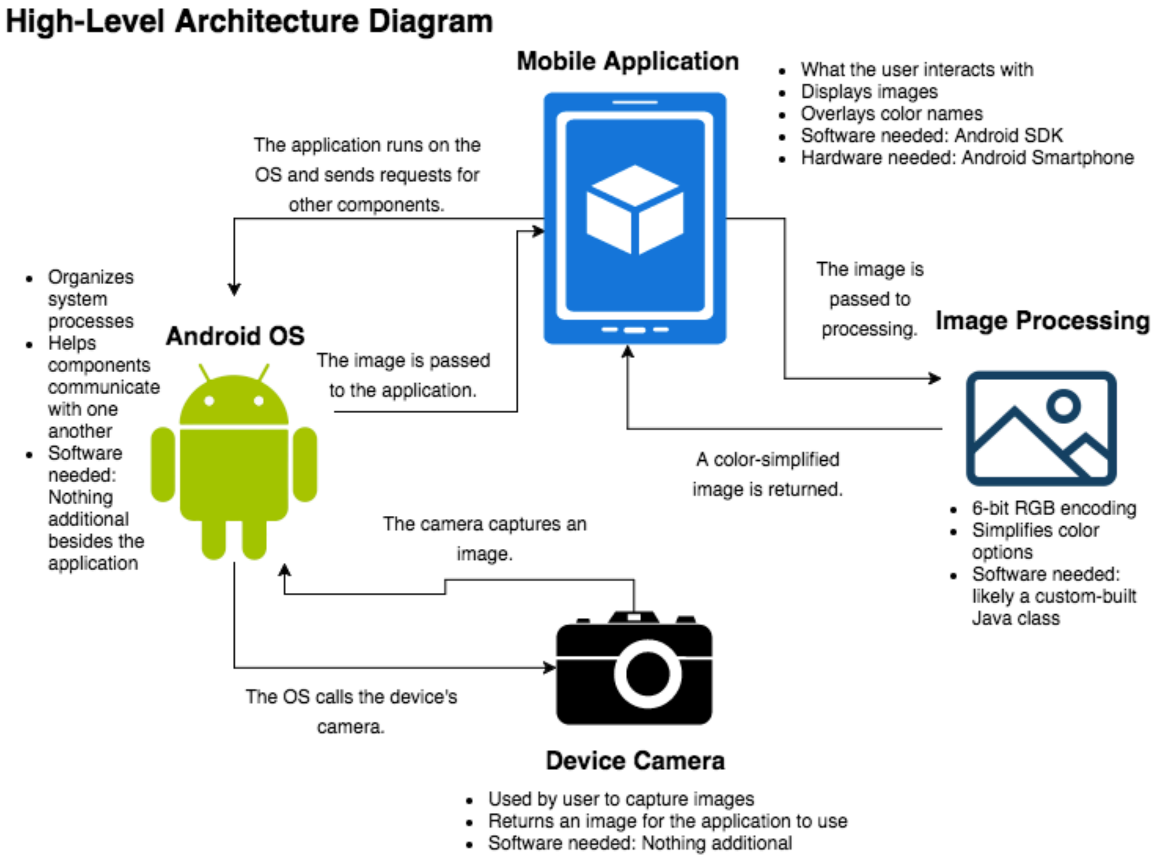
3.) The user finds some color that needs to be identified.

4.) Step 1 is repeated, but in the top half of the screen.

5.) The user taps the unidentified color, and he/she is then able to compare with the color key.

Figure 1: Example workflow for the target user usage case.

Architecture Diagram


High-Level Architecture Diagram
Mobile Application
• What the user interacts with
• Displays images
• Overlays color names
• Software needed: Android SDK
• Hardware needed: Android Smartphone
The application runs on the OS and sends requests for other components.
• Organizes system processes
• Helps components communicate with one another
• Software needed: Nothing additional besides the application
Android OS
The image is passed to the application.
The image is passed to processing.
Image Processing
A color-simplified image is returned.
• 6-bit RGB encoding
• Simplifies color options
• Software needed: likely a custom-built Java class
The camera captures an image.
The OS calls the device's camera.
Device Camera
• Used by user to capture images
• Returns an image for the application to use
• Software needed: Nothing additional

Technologies and Hardware

For this project's hardware, I've chosen to use an Android-based BLU H1 HD smartphone because it's relatively cheap, and I'm initially starting with Android development because I have some experience designing Android applications. Additionally, I'm choosing to make this a mobile application rather than a standalone device because I want this to be free and easily accessible.

Regarding libraries, the app will rely heavily on the basic Android SDK with a very low minimally required API level such as 15 (with 98.6 percent coverage) that will be determined more firmly during development in order to provide the greatest accessibility possible. The Android SDK will provide the abstractions needed to use the device's camera to take pictures, post the pictures on the screen, register the users' touch events, and get pixel colors to be used for identification. While the Android SDK will provide the bulk of the app's interaction with the device, one of the features that will be less cut-and-dry is encoding the image down to a finite set of possible colors to make comparison much easier. To handle this, I've come up with two options: I can either write an algorithm that will reduce a picture to a 6-bit RGB indexing scheme with 64 possible colors, or I can use the Java Image I/O API (javax.imageio) to compress the image which will simplify the colors as well. My concern with ImageIO's compression method is that the colors it provides may exceed the finite color palette, and it may reduce the image quality so much that it could make necessary details illegible. Currently, though, my plan is to give ImageIO a chance, but I'll also be ready to start writing my own class for 6-bit color for compression, encoding, or indexing.

Once the app is completed, users will be able to access the final Android application for free on the Google Play store, and if time allows for iOS development, it will also be made available on the Apple App store, but this decision will have to be made as development commences.

Team Logistics

Since I'm the only person working on this project, team logistics will be relatively streamlined. I will use Github for version control (using https://github.com/calvinsb1023/colorhelper as my repository,) and I'll be responsible for all code development. My name is Calvin Barker, my ONYEN is barkerc1, and my email for all points of contact is barkerc1@live.unc.edu.