Color Buddy
Senior Design Project Fall 2019

Christina Bateman
Lukas Saul

Abstract

Imagine you're shopping at your favorite clothing store. You find an amazing shirt, but you don't know if it goes with anything you own. If it doesn't match anything, you will have wasted money on a shirt you'll never wear. What if there was a way to know without having to make any more wasteful purchases? Color Buddy is an Android app developed with Kotlin and Firebase which helps users curate the perfect wardrobe, interior design scheme, and more. It holds the color data for each item in the user's collection. When a user wants to ada new item to their collection, the user takes a picture of the object. Color Buddy forms a palette of the object's most prominent colors and, using color theory concepts, checks to see if a new item matches anything in the collection. Users are alerted in the new item matches or not, so they can make the most informed decisions.

1. Introduction

The goal of this project was an Android application that helps users determine if a new item would match items they already own (i.e. clothes, furniture, etc.). Our expected audience for this application is people who want help unifying their color schemes and potentially people who are color blind. We expect that people would develop a greater understanding of color matching, gain a more unified color scheme for whichever set of items they used the app for, and saved money by not buying things that don't match what they own.

1.1. Background

Color Theory:
The theory behind color matching is based on the color wheel. It typically describes four schemes by which colors are matched.
Complementary: These are colors that lie across from each other on the wheel.
Analagous: These are colors next to each other on the wheel
Triadic: These are three colors that are seperated by three other colors on the wheel. This scheme is always three primary colors, three secondary colors, or three tertiary colors.
Monochromatic: These are colors that are the same base color but are a shade(darker) or tint(lighter) of that color.
All of this information was translated to the unit circle to make it possible to do calculations over the pixel data obtained from the pictures. Initially, it was RGB (Red, Green, Blue) but this was converted to HSV (Hue, Saturation, Value) to allow for easier calculations. A color is then checked to see if its monochromatic with a color in the group, we check to see if its with 45 degrees of the color. This will be the basis for the rest of the schemes. When we look for another color that fits the scheme we add the appropriate amount for the scheme (180 degrees for Complementary, 30 or 60 degrees for Analagous, and 120 degrees for Triadic), then check if that new color is monochromatic with the one we're looking to match.

1.2. Challenges

Image size:
We cut off the outer edges of the photo (10% off each side) and took a sampling of the pixels from what was left.
Image and color quality:
We loosened up the definition of the monochromatic color scheme to account of variances in image quality, lighting, etc.

2. Scope

This section is a bit tricksy. You are going to do your best to set up ground rules: How will you know when your project is done?

If you were doing this under contract for a company, this would be your checklist to make sure you get paid. We will be going into this in more detail over time, but you should start planning your major goals of the project as soon as possible.

For every sub(sub)section below, make sure to mark which items are basic goals (project won't be done without it) and which ones are stretch goals (it would be really cool to do...). We will be meeting one-on-one to help identify which goals go where.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1. Requirements

2.1.1. Functional.

- User needs to have private groups, these will be the items the user wants to potentially match and no one else should be able to see them.
- Users should have the ability
- You'll need more than 2 of these...

2.1.2. Non-Functional.

- Security – user credentials must be verified and stored within firebase.
- Usability – the application must be fairly simple so that most people could be expected to operate it without much trouble.

2.2. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table 1.

Use Case Number: 1
Use Case Name: Add item to cart
Description: A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:
Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2
Use Case Name: Checkout
Description: A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

Figure 1. First picture, this is a kitten, not a use case diagram

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

5.1. Future Work

Machine learning - this would allow the application to differentiate the item from the background in the image.
Improve calculation time - This could be done in a couple ways. It could be off loaded to a server or the phone's GPU could be utilized.
Outfit suggestion - Outfit suggestion could be implemented for wardrobes as the clothing type is stored along with the colors. This would also take the current weather into account.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.