# Tookacolour - Project 1

CSC 130 - Spring 2018

Dominique Charlebois

V00832004

# Tookacolour

An application that allows anyone to visually view the dominant colours in a tookapic picture.

## Problem and App Specification

### What problem does your app solve?

Currently tookapic lacks in visual representative analytics. The platform informs users of dominant colours within a picture, but does nothing with that information. This app aims to let users analyze individual pictures.

### How does your app solve that problem?

The app allows users to paste in the ID from a tookapic picture, click a button, and view a visual graph. The graph displays colours to the users so they can see patterns in their photography.

### What other apps attempt to solve the problem, or a similar problem, and why yours an improvement over theirs?

Other apps that have attempted to solve a similar problem:

- http://lokeshdhakar.com/projects/color-thief/
- https://www.imgonline.com.ua/eng/get-dominant-colors.php

These application are great if you have the image downloaded on your local device, but it does not analyze web images. Tookacolour is an app specifically made for tookapic users.

# Design and Storyboard

## Libraries / API

- Bootstrap V4 (https://getbootstrap.com/)
- Font Awesome (https://fontawesome.com/)
- React.js (https://reactjs.org/)
- React Chart.js 2 (https://github.com/jerairrest/react-chartjs-2)
- React Router Dom (https://www.npmjs.com/package/react-router-dom)
- Tookapic API (https://developer.tookapic.com/)

## Server Requirements

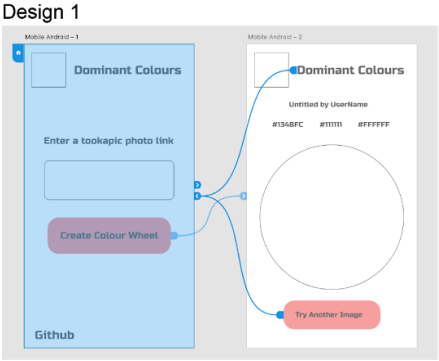
Backend that saves information from already searched images, and then creates a unique url with the photo ID, so users can go back to the page. Currently the app is being hosted on Heroku (https://www.heroku.com/home), so it can be easily scaled.

## Input, Output, and Processing Involved

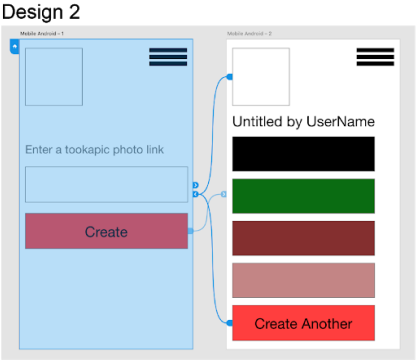
Users input a ID, that is unique to each photo uploaded to tookapic. They click the “create” button, and are sent to the colours page. On this page the information is outputted to users via a graph. The graph displays information gathered from the tookapic API. If the users wishes to view another graph they can go back to the home page by clicking the logo in the navigation bar, or by clicking the “create another” button. The users have an option of viewing the documentation and Github link in external links. They can access these by clicking the appropriate icons in the navigation bar. Lastly, the users can open an external link to the image that they were analyzing by clicking the title on the colours page.

## Designs

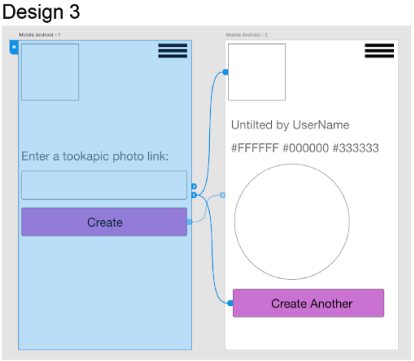
Design 1




Design 2




Design 3



### Compare and Contrast

| Design | Pros | Cons |
|---|---|---|
| **Design 1** | Simple design, everything is centered. | The additional background. |
| **Design 2** | The inputs are left aligned. No additional background. | The left aligned statistics. |
| **Design 3** | The inputs are left aligned. | The lined graph. |

## Implementation

Github: https://github.com/FlyteWizard/tookacolour
Website: http://tookacolour.dominiquecharlebois.com/

## Testing

### How did you establish your prototype is correct / complete?

We established correctness and completeness by performing quality insurance. There are no errors in the console when the app is running. And each page loads with a default. The requirements was to implement an application that allows user to insert an ID and get a graph back with the dominant colours in the image.

### What did you do to test its appearance and functionality?

I tested my code in an online Javascript Validator, to make sure that I didn't make any syntax mistakes. React.js has a built in error check, so my app runs, therefore it is syntax error free. I tested the appearance on multiple devices via the Chrome Inspection Tool. The application was also tested on different operating systems.

### What were the results of your testing?

The result in my testing showed that my colours page, loads before I send out an API Get Request. This means that the graph already loads the template component, before I am able to get the required information. The solution for the meantime, is hardcoded data for the graph. This means that the input field on the first page, serves no purpose for now. The testing also found an error with the React Chart.js 2 library. The pie chart does not respect set width and height, so in the mobile version the chart is very small.

### Test Your App on Multiple Devices

- Tested in Chrome Inspect Tools
    - Galaxy S5, Nexus 5X, Nexus 6P,  iPhone 8, iPhone X
- Tested in Browsers on macOS
    - Opera, Chrome, Firefox, Safari
- Tested in Browsers on Google Pixel 2
    - Chrome

## Documentation

### What work is left to do on the app to make it complete?

The call to the API is a functionality that needs to be improved. The call itself works. The user inputted ID is grabbed and sent with the API GET Request, but currently the graph loads before the Request is made and not updated after the data is retrieved. The last things to implement is to disable the button if nothing is in the input field, return an error if the ID isn't correct, and update the pie graph with the appropriate data. The biggest challenge that remains is getting the graph to update after the API call is made. The app was going to include the author of the image on the colours page, but the API call did not return the username. I would like to see if I can do another call with information from the first JSON file and gather the username.

### How could the app be generalized and extended to work in different domains?

The name can be changed to generalized and extend the app to work in different domains. If I were to create this app for other domains, I could accept URLs to images on the web, and use another library or API, to gather the dominant colours inside the image and return it in the graph.