PALETTE

Team 18: Palette Project Documentation

Jin Young Bang, Jacob Gruver, Jin Lou, Chris Balboni,
Nonso Chukwujama, and Justin Chan

CS 411 | Perry Donham
May 5, 2021

Table of Contents

1. Project Brainstorming

Project Brainstorming: **(Web App)**

1. A tinder-and-ebay-inspired app, designed for students to explore products from the student community.
    - Products can be artwork, fashion, online tutor, textbook, PC parts.... ANYTHING
    - Swipe to show interests (or nah) to the goods, if you swipe right, then you build a connection with the seller
    - Make friends in the student community, connect with popular sellers or talented freelancers, Sell goods in the local
2. Food delivery aggregation between Uber Eats, Grubhub, Postmates, Doordash etc.
    - Most of these apps have overlapping restaurants but some apps have more drivers → their delivery prices will be lower
    - Will allow users to use just 1 app to find restaurant instead of having to open each one and seeing if it is available on that particular platform
    - Will have a central payment hub (apple pay, android pay, etc)
3. Club/bar party locator:
    - uses google maps to allow users to mark where they are and the experience they're having (good, meh, bad); the app would then show the average score within the last hours so that bar hoppers and the like can see which spots are "bumping" at the moment without having to think much
    - maybe it automatically orders an uber to bring them there if they click a button?
4. Terrier Scavenger Hunt
    - A game app designed to test how well BU students know the campus. The app will provide clues for candidates, such as pictures of BU architecture, and candidates earn points based on how many locations they find. When a location is found, candidates use the app to scan the architecture and the app should recognize it and give points based on correctness.
    - Display team rankings based on points
    - Each player has a profile with their names and bu ID, and several players form a team.
    - When candidates spend too long on one quest, the app will provide hints.
5. Social media sentiment analysis to give relevant movie quotes
    - Run sentiment analysis on social media posts and assign an appropriate movie quote based on the posts for the past couple of weeks
6. **Spotify Playlist Color Scheme Generator**
    - **APIs: Spotify + Google Vision API**
    - **Get all the album covers, run it through Google Vision API, get hex codes of colors, rank it based off of most seen and showcase it on the frontend**
    - **“album cover inspired outfits”**

2. Proposal

Our first idea is to create a web application that allows us to get the top 5 color schemes based on a user's Spotify Playlist. Firstly, a user would sign on to their Spotify account, and the web app would allow the user to choose a playlist. Then, we will make an API call to Spotify's API to fetch all the images of the album cover and send that to Google Vision API to get the color data. After aggregating all these data, we plan on showcasing the Top 5 colors of your playlist. We plan on using Spotify API and Google Vision API for this potential project.

3. Technology Stack

Stack

- Flask
- React
- MongoDB

Process:

- We chose **React** as our frontend JavaScript library because:
    - React is useful for building web applications with data. Because NPM is one of the largest package managers available, it can also be combined with other javascript frameworks or libraries for more features.
    - React is easy to test through view. We can test our work by manipulating the states passed into the view and check the output to see if they are functioning.
    - As a popular frontend library, we can easily find resources for React so as to build different functionalities for our project.

- We chose **Flask** as our backend framework due these main reasons:
    - Flask is a framework written in Python and uses Python to allow us to create APIs – as all members are familiar with Python (as we have taken CS 111), we thought that using Flask would be appropriate for us.
    - Flask is a micro-framework and has an easy learning curve. Majority of us do not have the experience of using a backend framework. Using Flask is very intuitive and will allow us to understand how backend frameworks work.
    - Python and pip also has an array of open-source libraries available for us to use. In the case that we need to work with additional libraries and functionalities, using Flask would be an appropriate case for us,

- We chose **MongoDB** to host our database for the following reasons:
    - Firstly, some of our members are familiar with the PyMongo library – this makes it easier to work with data using Python and MongoDB.
    - We preferred using a NoSQL database due to its simplicity and flexibility in storing data as a JSON.

4. User Stories

User Story #1

We want to launch the web application and be greeted. After we are greeted, we want to be able to login to our Spotify account to connect the web app to our Spotify playlists. If we don't have a valid Spotify account, we should be told that we cannot login to the app. If our Spotify account is valid, we should be redirected to the home page, where we can see previously generated color palettes from previous uses of the application and all of our other Spotify playlists as well.

User Story #2

After we login to our Spotify account, we should be directed to the homepage of the application. We can then choose to generate a new color palette where we can select one of our existing Spotify playlists to generate color palettes. If we do not have any playlists, a message should pop up prompting us to create a new playlist. If we have one or more playlists, we can select one playlist. If the playlist of our choice contains no songs, then we cannot generate a color palette until we add at least one song into the playlist. Otherwise, we can click on the "start generating color palettes" button, and the application will display a menu with the color palette as well as other data about the playlist. This information will also be accessible from the "Color Palettes Generated" button.

User Story #3

When we log in again, we want to be able to view our previously generated color palettes by clicking on a "View Generated Playlists" button. If we do not have any color palettes generated, then we should not have a blank screen and there should be a popup informing us of this and navigating us back to the main page. When we click on a previously generated color palette, we should have the ability to "refresh" the palette, where the application will regenerate the color palette to adjust for songs being added and/or removed from the playlist. If the playlist is refreshed and the playlist now has no songs in it, then an error message will appear saying that the playlist that we refreshed is empty, and the app will delete the playlist from the generated list. We can also manually delete generated playlists, which would remove the palette's playlist from the generated list and place it back with the playlists that don't have generated palettes.

User Story #4

When a color palette is generated, it will display colors with their hex codes and have a background pattern of the color that the mouse is currently over. This pattern should be randomized each time a palette is generated. The user should also be able to click on any of the colors to receive more information about them. The user should also be able to view the colors associated with each playlist from the list of generated palettes.

5. Final Proposal

This is our final proposal for our semester long project that we have chosen to name Palette. Using the user stories as a reference point, we were able to meet most of our goals. In this final proposal, for sake of non-repetition, we will only reference features that have either required subtle changes or a complete pivot.

Once a user has successfully connected their Spotify account to Palette, they can access all the playlists by either listening to the music or generating the color palettes. All the generated palettes will be stored in the database. Users can access them through the top right corner button each time they log in.

The methods being used meet all the requirements for the project: 1) We use MongoDB as the database to store user ID and the generated color palettes so that users can access previously generated palettes. 2) We use Spotify and Google Vision as the two APIs to fetch data for our project. The Spotify API can get us the user's playlists after log in, and the Google Vision API lets us fetch the hex color code from the playlist cover image. 3) We use Spotify for the third-party authentication requirement. 4) The project implements the decoupled architecture as we chose React for frontend and Flask for backend.