Palette Picker

Simon Stockton, Victoria Senn, Eric Brewer, Lily Thompson

Introduction

Our project is a web-based application that allows users to create color palettes of five colors from both provided and uploaded images. This app will also allow users to create profiles on the site, allowing the sharing of palettes between members with accounts. Having an account will also allow users to access features such as saving palettes, adding other members as friends, and viewing activity of friends.

The primary functions of the application include creating and saving palettes to a personal library, as well as searching palettes made by others and saving those. One of the application's main features automatically creates a palette of five colors from a user-selected image, utilizing the most common color within. The user can upload their own image or choose a pre-existing image built into the site. Users will also be able to click on different areas of an image if they want to get the colors from specific points.

On the social side of the site, users will be allowed to share their color palettes with friends and the community. The community page will allow users to find color palettes created by other users of the Palette Picker community. When users save a palette, it will automatically post to their profile, and be available through the search page.

Beginning with the idea of an application that would match hex codes to corresponding mason stains and underglazes, the Palette Picker evolved to solve a more general problem of being able to create sets of colors to reference for any purpose. We then added the social element in order to create a more robust application that we could all work on and have an interest in.

Use cases for our application include creating a palette from an image in order to use those colors in personal art, trying to approximate the colors in a digital image in order to use them in traditional art, or finding complementary colors for a color the user is already planning on using in their art. Outside of creating artwork a user can use this complementary color feature to decide how to decorate or style their home.

Technology

I. Coming up to Speed on New Technology

The technology being used was new to all of us. Much of the time spent on the application was spent learning new technology and applying what we learned dynamically. PHP, HTML, CSS, and JavaScript were foreign to all of us going into this project. Our team had a small amount of experience in MySQL/MariaDB, so this was the only thing that we had some sort of guidance in. Permission issues occurred in the later stages of development, which was a challenge to overcome. Setting up the web server was also a challenge because none of our team members had any prior experience in web development or server management. Setting up localhost and using XAMPP was also new to members of the team and came with installation and running complications that slowed development. Learning to use SSH in the console was a huge pace change considering our team is used to using the point and click interface that windows provides. The whole team had experience in GitHub, so this was the only thing that was not a challenge. Overall the learning curve was a huge challenge for the team and had major ramifications on the development speed and completion.

II. Using New Technology to Solve the Problem

This project was developed utilizing Windows operating systems and is hosted on a Linode server. The codebase consists of JavaScript, HTML, CSS, and PHP, and was developed on Visual Studio Code. HTML and CSS are used to create the general layout of the site, including the links and headers. Javascript is utilized primarily in the palette creation, for both manual creation and creation from an image. PHP is used throughout the application, as it allows the HTML to connect to our database. For version control, we connected GitHub to gitDesktop, which ensured that unsafe states could be reversed and corrected. The database is relational, developed in MySQL. A relational database suits our needs as we need to be able to relate or connect users to their palettes and their saved palettes, as well as their login information.

III. Switching Up Technology

Our team's original plan was to use JavaScript to run the entirety of the backend because we originally believed we had some experience in JavaScript, and it would be the most simple implementation. This theory proved to not be true. After doing some research we learned that PHP would be much easier to learn in a short amount of time, so we quickly switched our plan. Though some JavaScript was used in behind-the-scenes palette creation, the majority of our functional code was written in PHP.

Design

Files	
127_0_0_1.sql	System Data.php
Authentication.php	Color Picker.js
Manual Code.js	

Block Diagram

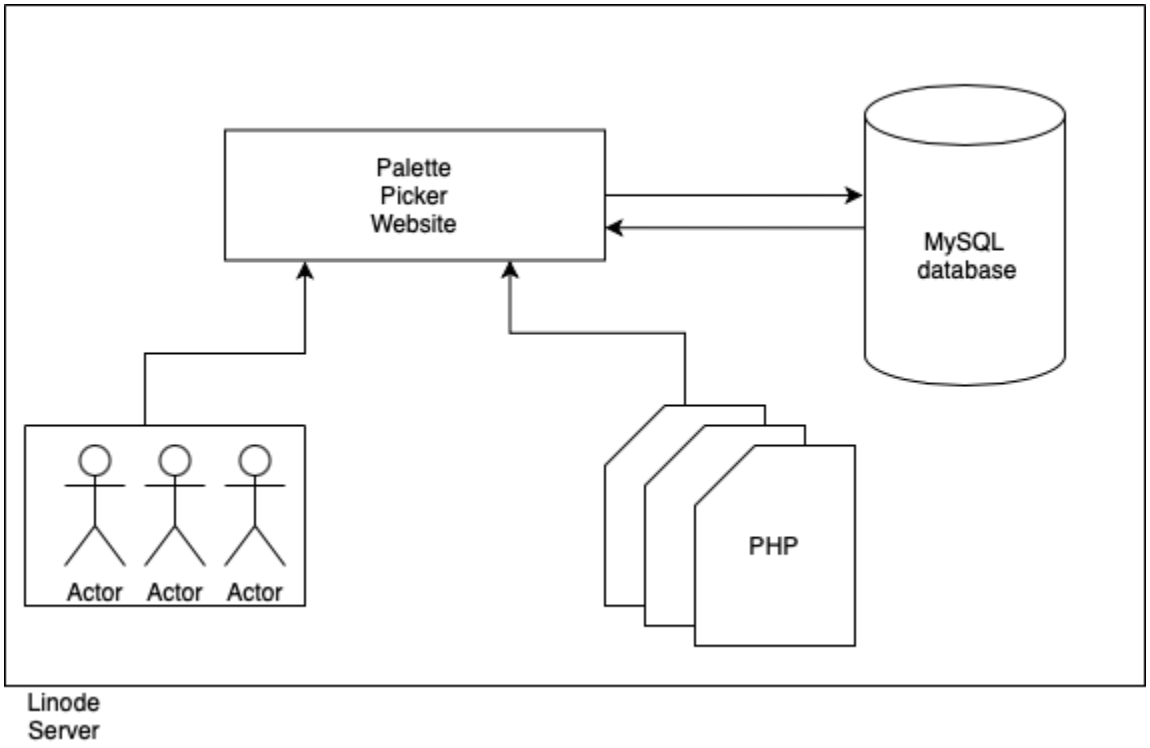

Figure 1.

The above figure demonstrates the design of how users will interact with the web server, and how the web server will interact with itself and the user.

Component Diagram

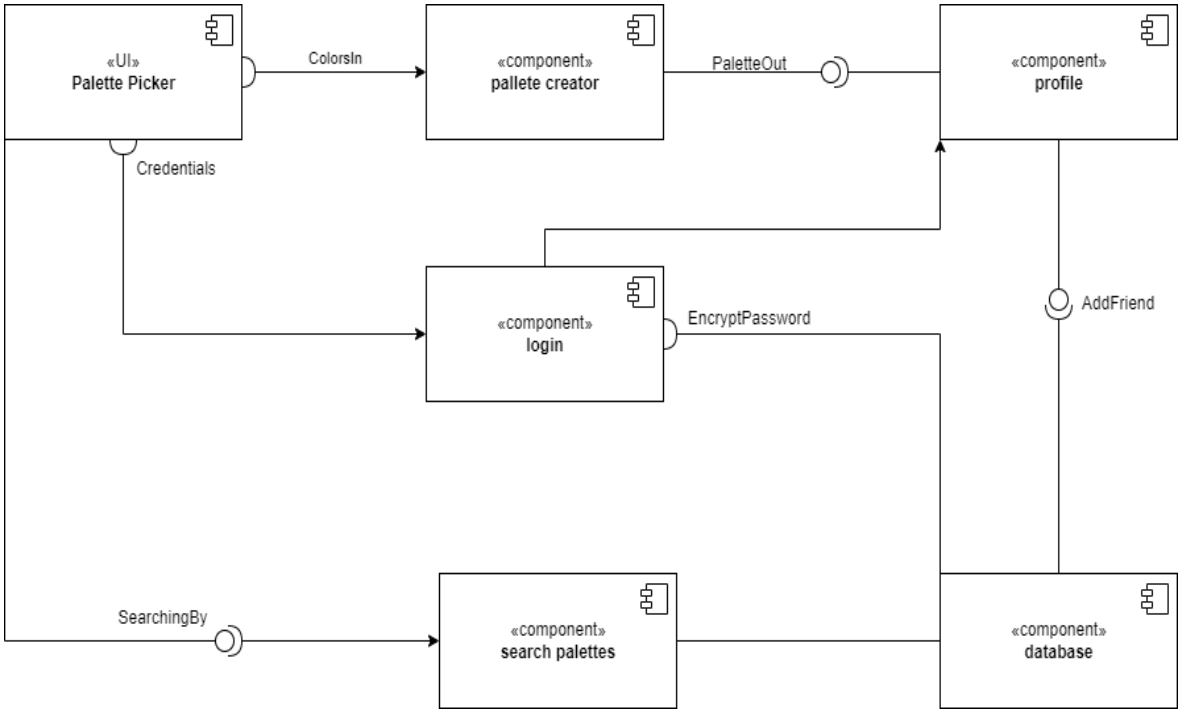

Figure 2. demonstrates how various components will interact with each other in the system.

The component diagram starts with the UI. From the UI, depending on what the user wants to do, it can connect to the palette creator component, the login component, or the search palettes component. The palette creator component can save palettes to the database via the profile component. From the login component, the user will be able to access the palette creator component via the profile component. The login component will also encrypt the password before checking with the database to see if the password is verified or not. The search palettes component will pull all of the relevant palettes defined by the user, and display them to the UI.

Design Diagram:

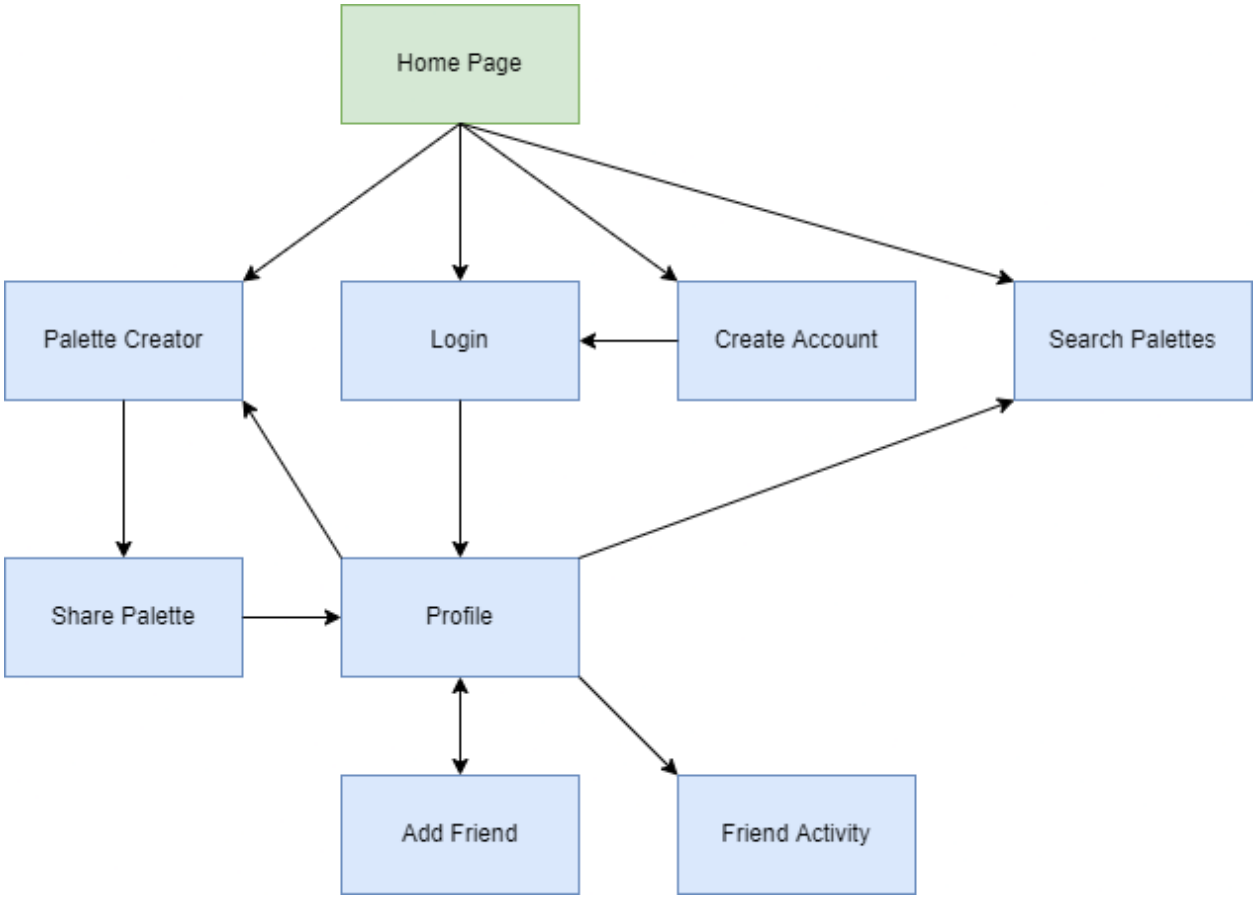

Figure 3.

The design diagram shown above illustrates the design pattern of how we implemented our components.

The home page or index.html is where the user will begin their Palette Picker experience. From the home page, the user will be able to access four unique pages, those being the palette creator itself, the login page, the create account page, and the search palettes page. The search palettes page will allow users to search the database for existing palettes. The create account page will allow users to register on the website. The login page will allow users to log in to their accounts. The palette creator will allow users to manually create a palette and save that palette to a database. The profile, add friend, and friend activity have yet to be implemented (see future work).

File Descriptions:

Inside Public_html

CreateAccount.php	HomePageLoggedIn.php
DatabaseConnection.php	Login.php
ProfilePageRoughDraft.php	Search.php
Index.html	LargePalettePage.php
Pallet_Screen.html	Style.css
FromImage.html	manual_code,js
Colorpicker.js	

Inside Images	
RainbowHeader.png	Flowers.jpg
Starry.jpg	Wheat.jpg

The public_html folder is home to all of our functional code, along with our CSS file Style.css, which is utilized throughout the folder for basic layouts. It also contains an images folder that holds the images that will be displayed on our UI pages. The public_html folder is also where all of our code gets put on our web server into a user directory that hides the locations of our files.

Stepping through the program from a user perspective the files are called as such: Firstly a user encounters the landing page, index.html. This page contains a link to the palette creator, which can be utilized without being logged in. There are also links in the top navigation bar that allow a user to sign in, create an account, or search.

If a user has not created an account and they click the button to do so they will be directed to the createaccount.php file. This page requests for users to enter a new username and password, the latter of which is entered twice for confirmation. The user will not be allowed to continue until all of these fields are completed, as a dialog displays under the empty field requesting the user fills it in. Once all fields are completed the system then uses the function verifyUsernameNotTaken to check that the username is not already being used by another user, and also checks to make sure the passwords entered match each other. After these are confirmed, the password is then encrypted using PHP's default b_crypt and sent to the database for future use via sendNewUserToDb.

Supposing the user had already made an account they would have instead gone from index.html to login.php, where they would enter their credentials. Once again these fields cannot be left empty or a prompt to fill them will display. The entered data is then checked against the database utilizing authentication.php and the dbConnection.php function getUserHashedPasswordFromDB, which displays text on both successful and unsuccessful logins.

Continuing as a logged-in user the next page is homePageLoggedIn.php which serves as the home base for logged-in users. Here is where users will be provided hyperlinks to get to other web pages, similar to index.php. Again a prompt to create a palette is displayed, as well as a link to search pre-existing palettes.

Assuming a user would like to search for inspiration, they can currently enter a title, though more implementations are discussed in Future Work. If no palette is found that matches the user's query a "no results" text displays. If there are palettes that correspond to the search, then they are displayed within inline blocks and can be clicked to show their corresponding LargePalettePage. They are retrieved using the findPaletteByTitle function from dbConnection.php.

The LargePalettePage is a larger view of a palette found from searching that uses the function getPaletteInfo from the file dbConnection.php to display information such as the palette's title, hex codes, and creator. From this page, a user can save a copy of the palette into their own library which is done using the savePaletteForUser function in dbConnection.php.

Currently, the plan for a user to view their saved palettes is to have them displayed on ProfilePageRoughDraft.php, but this page and its functions are still being worked on and will be discussed further in Future Work.

Returning from search to homePageLoggedIn.php a user can click the "Go to Palette Creator" button and be taken to Pallet_screen.html. This HTML file links to manual_code.js to allow a user to create a palette one color at a time. The hex codes are read by taking the X and Y values of the mouse pointer in relation to a canvas that lays out all the colors of a specific hue value. This hue value can be manipulated by adjusting a slider next to the canvas that has a variable gradient that represents any hue the user might want. This slider is split up into 6 gradients, Red->Yellow, Yellow->Green, Green->Cyan, Cyan ->Blue, Blue->Purple, Purple->Red. After the value of the slider is adjusted, the code takes the modulus of that value to see which gradient the slider is in. The code then takes the slider's value within that gradient and puts it into a color vector based on which gradient it's in. After the vector has been calculated, the value gets sent to the canvas, which creates a 2-dimensional gradient, showing every color between the chosen color hue and black, as well as the chosen color hue and white. There is also an option to reset all of the blocks for the palette to hex code #000000, or black.

If the user would rather use an image to create a palette, they can click on the "create from image" button, which will transfer them to a new page, where the user will have a button allowing them to choose an image. This will open their file explorer and select any .JPEG or .PNG file. After the user selects an image off their computer, the code will create a palette of the most common colors from the image. This is done by reading each of the red, green, and blue (RGB) values from each pixel into an array. Then the algorithm slices the picture into 5 partitions, top-left,top-right,bottom-left, and the whole picture. The algorithm then takes these partitions and finds the average value of each R, G, and, B value, then puts the color created with those values into the palette. When pushing the code onto Linode, an error came up which prevented us from taking pictures of the users desktop(See Known Bugs), due to this we provided sample pictures on the page to take pictures of. This uses the same functions, the only difference being the source of the images are from the database rather than off the users computer.

The algorithm for finding the most prominent colors has changed multiple times over the course of the project. When first implementing the function to read data from the picture, the function was just built as a placeholder to find the overall average color of the picture.That code

was imprecise and there needed to be five colors for a palette rather than the one given. The algorithm was then changed to increment through the image and increment a count for each original color in the image. This algorithm was quickly thrown out due to being very computationally intensive and not representing similar colors well. The algorithm was then changed back to taking the average of the image, this time taking four vertical slices of the image. Certain phenomena within pictures made it so that vertical slices very poorly represent images, giving near identical values for the palette, as well as gray values. This brought the algorithm to its current state of taking average from four quadrants. While this algorithm does fall into some traps, it is currently the best option. One of the problems this algorithm has is getting gray values if a quadrant has many vibrant colors. Another error is that if an image has a large focus in the middle, It'll slice the image on top of the focus and not represent it in the palette.

On either palette creation page, after the color is chosen, the palette takes in the saturation of the color as well as how dark/light it is and decides the best color to display the hex code in (Black or White). Each page also has a save button that will open up a prompt allowing the user to input a name for their palette then save it to the database using the userID, Name they input, And the values for each box of the palette. An example of the palette from an image is proved in Figure 4 below:

Example Palette from Image:

Figure 4. An example palette was created using the create from image feature. The picture is Van Gogh's *Sunflowers*.

If the user chooses to save their palette via "save" then a dialog will open, prompting the user to enter a title for their palette. This is then sent to the database via sendNewPaletteToDb in dbConnection.php and, in the future, should also be viewable on ProfilePageRoughDraft.php, as well as the search page.

User Interface Images:

Home Page (Not Logged In)

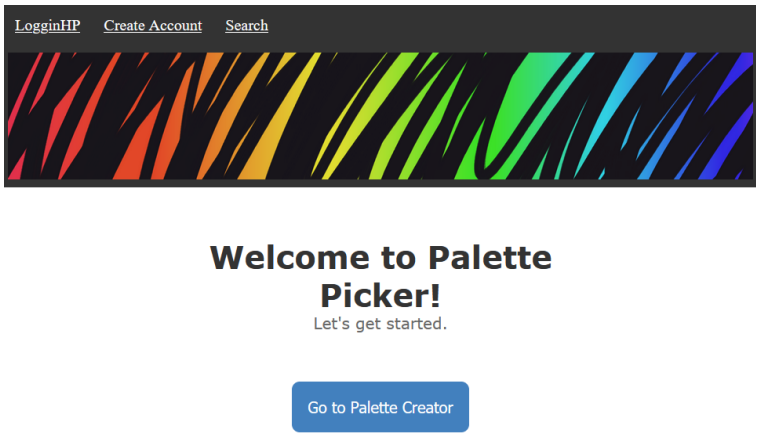

Figure 5. This is the home page for someone not logged in. From here, a user can go to the palette creator page, go to the login page, go to the create account page, or go to the search page.

Search Page with 2 Results

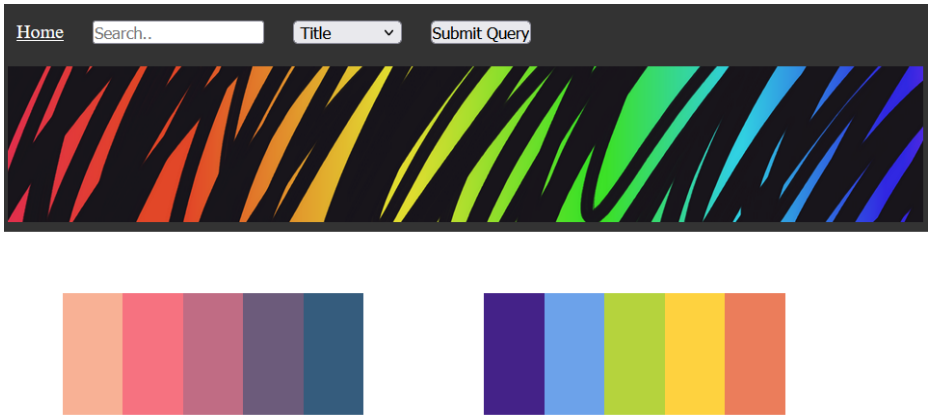

Figure 6. This is the search results page. Currently, users can only search by a palette's title. The user can click the palettes to go to their pages, enter a new search query, go to the home page, or click the dropdown menu, which gives the choices of searching by title, color, or creator, but has no functionality for those options at this point.

Palette Creation Page (from scratch)

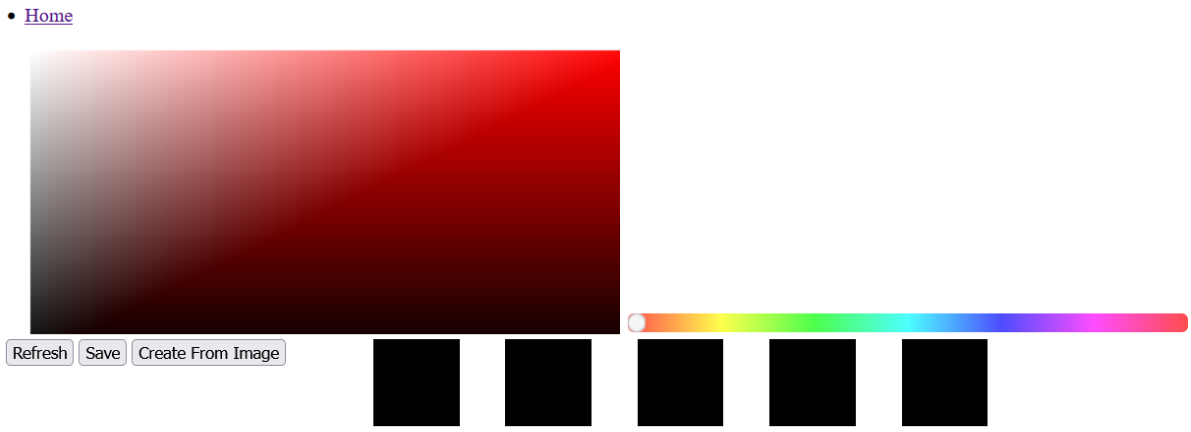

Figure 7. This is the page for creating a palette from scratch. Users can click one of the five boxes and then click somewhere on the gradient box to select a color or use the slider on the right to change the gradient box's hue. The user can also save their palette (implementation pending), refresh, go to the create from an image page, or return to the home page.

Create Account and Login Pages:

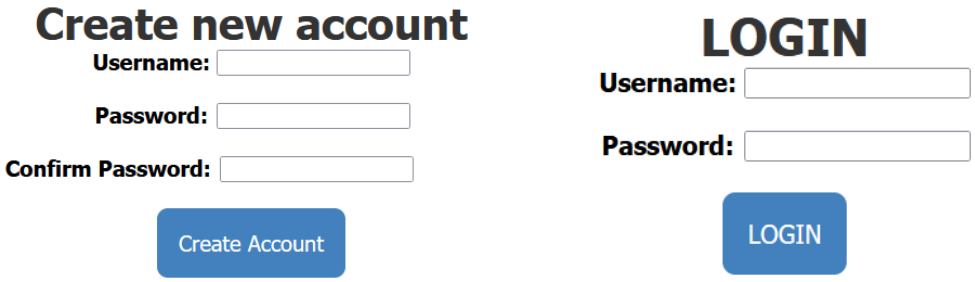

Figure 8. Here are the create an account and login menus. Using them, a user can create a new account, which adds a new user to the database, or login if a user already exists.

Database Tables:

The 127_0_0_1.sql file is our database file. This file is used to run a query to create a new database with all the tables required to give our application functionality.

The systemData.php file works to provide access to the database by supplying it with vital data. It contains a username, password, database name, and IP address to access the database. This file is not available on GitHub because of the sensitive information it contains.

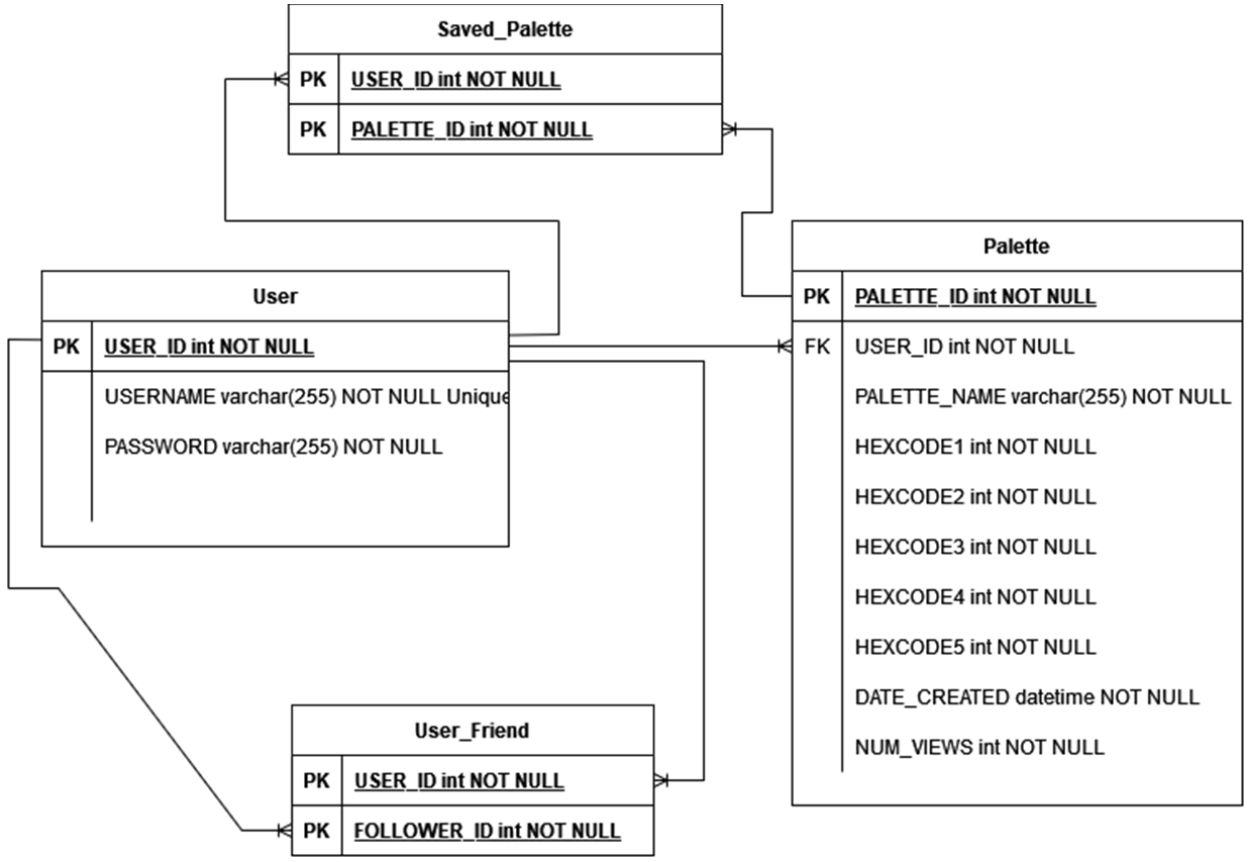

Figure 9. The diagram of all the database tables.

As shown above, in Figure 9 the database includes four tables: a User table, a Palette table, a User_Friend table, and a Saved_Palette table. The User table has 3 classes: an integer ID as the primary key, a 255 varchar username, and a 255 varchar password. The Palette table has ten classes: an integer ID as a primary key, an integer user id that is a foreign key, a 255 varchar title, 5 integer hex codes representing each color in the palette, a date representing the day the palette was created, and the number of times a person has clicked the palette's page. The User_friend table has two classes that make its primary key: an integer user id and another integer user_id representing the person's friend. The Saved_palette table also has two fields as a primary key: an integer user id and an integer palette id. All classes must not be null. All primary keys for every table must be unique. A User's username must also be unique.

These relations are utilized in dbConnection.php, as previously mentioned. The functions verifyUsernameNotTaken, sendNewUserToDb, getPaletteInfo, findPaletteByTitle, savePaletteForUser, and sendNewPaletteToDb have been explained in their contexts in the section above. Beginning the remaining functions are connectToDB and closeDB which are used in the public.html files above to connect to the database when necessary. Then there are a series of gets for user ids, user friends, and user-saved palettes. Next, there are the two not yet implemented search functions, findPaletteByColor, and findPaletteByCreatorUsername. More functions are outlined in future work.

Here are more in depth descriptions for each function in the dbConnection.php file (and mentioned above). connectToDB establishes a connection to the database. It needs the hidden systemData.php file to work. closeDB closes the database connection. getFromDB and

sendToDB are the functions every other function in the file relies upon. Both use an sql string to get some query from the database.

Next are the verification functions. verifyUser takes a username and password and determines if a user with both those arguments exists, returning true or false. verifyUsernameNotTaken takes a username as a parameter and returns true if no user with said username exists and false otherwise.

Then are the data selection functions. getUserID takes a username and returns the user's user ID. getUserFriends takes a user ID and returns the User IDs of all users the passed user ID has listed as friends. getUserSavedPaletteIDs takes a user ID and returns the IDs of every palette the user has saved. getPaletteInfo takes a palette ID and returns the palette's title, creator's name, and hex colors (as decimal numbers).

Fourth are the functions implemented specifically for the search feature. findPaletteByColor takes a color (should be a decimal number) and finds every palette ID that has said color as part of its palette. findPaletteByCreatorUsername takes a username and returns all palette IDs of palettes that were created by the searched for user. findPaletteByTitle takes any string and returns palette IDs for every palette whose title contains that string.

Then are the functions for inserting data. sendNewUserToDb takes a username and password and inserts a new user into the database using those parameters. sendNewPaletteToDb takes a user ID, a palette title, and five colors (in decimal format) and inserts a new palette into the database. getUserHashedPasswordFromDM takes a username and returns said user's password.

Finally there is one function to save a palette for a user. savePaletteForUser takes a user ID and a palette ID and inserts a new record in the saved_palette table for the user to be able to find a palette again later.

Deployment

To deploy the Palette Picker you must have an apache server either on your local machine or through a third-party hosting service. Our team developed our code running an apache server on our local machine, along with a MySQL server, using XAMPP. In order to upload pictures from a local machine to a website, the website must have a SSL certificate. A SSL certificate, or a Secure Sockets Layer, is an encryption layer that certifies a website's authenticity and prevents reading or modifying information sent between systems. All browsers The final product was deployed using a third-party hosting service called Linode, which runs an apache server off an Ubuntu machine. PHP and MySQL will also need to be downloaded on the server. This will more than likely need to be done manually on a Linux machine but will come with XAMPP if you choose to run Palette Picker off your local machine.

All of our code is available via GitHub through this link, https://github.com/CodeBrewer99/Palette-Picker.git. If you choose to create your own Palette Picker from scratch you will need an IDE that can be used to develop HTML, CSS, PHP, and JavaScript. You will also need something to create MySQL/MariaDB databases and tables (we used phpMyAdmin).

Known Bugs

We have a lot of features that work on their own, but not when connected to the whole system.

On the manual palette creation page there is a bug where if you select a color on the edge of the canvas, it will set that color to white, and the text to white. Also on the manual palette screen, there is a bug where when the Banner is displayed it offsets the x and y values of the canvas. In the demo there is a work around by temporarily removing the banner.

Saving a palette from the Palette_screen.html is not yet available (probably also on FromImage but there is not a way to check due to another bug). We cannot figure out which part of the system is working incorrectly for that.

On the LargePaletteScreen page, the button for adding a palette to the user's library does nothing at the moment. Also, clicking the palette creator's username does nothing as well, but should link to the creator's profile.

Getting a palette from an image is not possible without being able to upload an image. An image cannot be uploaded to the web server without having an SSL license, as mentioned in the section on deployment. This feature has been implemented in the GitHub project but is considered bugged on the webserver, as Linode has no free option for SSL licensing.

Going from the Home Page Logged In screen to the Search page back to home, bring the user to the Home Page for users that are not logged in. This is also the case for going from the home page (logged in) to the palette page back to home page.

Future work - Version 2 of Palette Picker

Search

Due to time constraints, some functionality for the project has yet to be implemented. For the search feature, we currently have a link on multiple pages that directs to the search page, where someone can type in a search. In the future, this should be a text box that allows a user to enter a search and sends the search information to the search page, and automatically finish getting results for said query.

Also relating to the search feature, the current search page only searches by a palette's title for results. There is a dropdown list to search for hex colors and creator and there are functions in the database connection file for searching by color or creator, but the two are not linked at this time.

In the future, there should also be ways to sort results by how recent a palette was uploaded and the number of times it has been viewed (all of this is data that is stored in the database but not currently used).

Palette Pages

On LargePalettePage, there is a button for saving the displayed palette to the user's library, but it has no functionality at the moment. It should connect to the database and save the palette via the savePaletteForUser function. Also, the creator username on the page should link

to the user's profile, but does not at the moment. The palette's number of views in the database should also increment every time this page is visited.

Account Creation

After adding a new account, the user must go to the login screen and reenter the information that was just created to log in. In the future, the user should immediately be logged in after creating an account.

Social Networking

The social networking part of the site is also still under construction, as friend-adding, friend-removing, friend-activity, and the alert system that goes along with these capabilities is also not fully implemented. For adding and removing friends new database functions will need to be added to dbConnection.php. Friend activity has a partial implementation in that there is a function for getting user friends, but there is currently no function to get user activity, and there is also no page to display the results of that function. On implementation, the function for getting activity would likely show the most recently saved palette for each friend. Alerts would require the most work out of these features as it has no foundation currently.

The user profile page is currently in a draft phase. In the future, it should display user information such as saved palettes, notifications, friend activity, and allow users to add other users as friends.

UI

When hovering over a search result, having the border change colors or the palette image expand would be a useful user interface(UI) feature. In general, the UI elements could be made neater or more aesthetic, which could be done by including new technology such as Bootstrap, which is an open-source front-end toolkit.

Palette Creation

Creating a palette from a photo has been developed already. This can be demonstrated locally, but to do this on linode we will need to obtain and install a SSL certificate. Uploading a photo to the browser requires a SSL certificate and can only be done when you are connected to a HTTPS website, not a HTTP.

We'd also like to improve on the algorithm and keep trying to find an algorithm to 100% accurately show the most prominent colors from an image. Also the bug that inhibits the feature of sending palettes to a database needs to be fixed in future versions.

Another feature within the palette creator, we would like to implement a feature that creates a second palette of colors that complement the first. For example, by uploading an image of a room, a user could learn what color items or furniture would work well with the room. Another feature would allow users to download png of any saved palette, including the hex codes of the colors. This would be useful for users that intend to create digital art or work with their palette in programs such as Illustrator or Photoshop.