**The Overview**

This case study discusses the process I went through when creating Kazi’s Kitchen. I was introduced to PHP (Hypertext Preprocessor) and SQL and had to apply what I already knew about coding to create a website that connected to a database to auto populate information. I will talk about the design choices I made and the steps I had to take to successfully code what I had planned to do for Kazi’s Kitchen. For the final project, I was able to code my website using HTML, CSS, JavaScript, PHP and SQL.

## Context and Challenges

### Background

This was a project I worked on for my IDM 232 Scripting II class during the Fall 2022 term. In this class, I learned about what PHP is, the important of it, the logic of how it works front- and back-end, as well as its structure. We learned how to create and use a database, and write code to add functions like create, edit, delete to the admin side of the website. As well as, functions like the search and auto-populating information (from the database), which is seen on the user’s end of the website.

I started off by researching some websites, trying to figure out what my website should be about. I decided to do a recipe site using recipes that my mom uses, which is why I branded the site as Kazi’s Kitchen. To ensure I was not over budgeting my time, I planned out the design and information structure using Figma. After wireframing and getting approval, I began to code my homepage, header, and footer in HTML and PHP.

### Problem

This project was assigned to challenge my new knowledge of PHP and SQL, as well as help me make the connects on how PHP works by working through it. This project required me to apply and create connections between the content I already knew and what I just learned in class. From the class, I learned that PHP and SQL are an important piece to incorporate into website with large amount of content that may need to be updated frequently.

### Goals

* Understanding PHP and SQL
* Maintaining organization both in my files and directory
* Creating components in PHP for any repetitive content
* Creating forms for the admin to Create, Edit, and Delete content.
* Creating a website using my new knowledge of PHP and SQl

## Process and Insight

### Alpha Description

Planning a simple microinteraction was the first step part of my project. The hardest part for me during this assignment was coming up with an interaction that is simple enough to build on. I spent my time during this process searching for inspiration and brainstorming ideas for this microinteraction. After brainstorming, I decided that my next would be to explain these interactions and create wireframes to visually understand and navigate through developing it.

View my [**Alpha Description**](https://samihashoshi.com/idm241/alpha/) to see my wireframes and microinteraction structure.

### Alpha Build

After planning and wireframing my microinteraction, it was finally time to develop it with code. I searched for working examples and tried to understand the existing code, in order to implement the example into my microinteraction. Through this phase, I was heavily reliant on Visual Studio Code, Codepen, and W3School. I had used CSS to create my animation for when the user hovers over the image. During this stage in my project, I struggled most to make sure that the text and image on the overlay were responsive. Also, I was worried that my plan to create a slideshow as my final product would become an issue with this overlay with content based on the image. For this reason, I coded a slideshow using some JavaScript to ensure everything I built moving forward would function properly with this new element.

View my [**Alpha Build**](https://samihashoshi.com/idm241/alpha/build).

### Beta Description

After coding my Alpha, I already knew what I wanted to do next. I need to learn how to create and implement a pre-loading animation. While brainstorming, I realized that the best way to implement this was to have the pre-loader animation spin while the learn more page is loading. Using an animation like this was very new to me. So before finalizing this idea, I spent more time brainstorming and came to the conclusion that this would be the best method for my project and I would get the opportunity to learn something new. In class, we were taught the different types of loading animation, but were not taught how to code it. At this stage, I feared that I would not be able to create and code a pre-loader that made sense my design and overall layout.

View my [**Beta Description**](https://samihashoshi.com/idm241/beta) to see my wireframes and microinteraction structure.

### Beta Build

After planning out my Beta, I started off by looking for working examples of a pre-loader what looked and acted similarly to my planned preloaded. After attempting to understand the working examples, I quickly realized that I would need a bit more time that I had originally thought to develop a pre-loader. Once, I found an animation that I thought would work with my design, I started off by playing with the example code to understand why the developer wrote his/her code in this method. Then, I worked hard to make my own version of the pre-loader in Codepen. After getting this microinteraction working, the next step was to actually implement it into my Beta Build. After doing so, there were a lot of challenges as at this point. In order for this loader to slide in on the overlay and also connect to the different link based on the image, I had to reformat and code a lot more in JavaScript that I initially thought. After adding this element to my build, there were some stylistic and responsiveness issue due to the added element. With the help of music and a lot of research, I was able to resolve these issues before the deadline.

View my [**Beta Build**](https://samihashoshi.com/idm241/beta/build).

### Final Description

After the Beta Build, it was time to make the microinteraction more complex by making sure that the overlay’s content changed with the images in the slide. I was glad I built out my slideshow earlier, that way at this stage I could focus more on the overall functionality instead of attempting to apply my microinteractions into a slideshow layout. Due to my proactiveness, I was all to focus more on the finer details in my project. My plan for my final build was to focus on the appearance, timing and the transition of the slideshow itself.

View my [**Final Description**](https://samihashoshi.com/idm241/final) to see my wireframes and microinteraction structure.

### Final Build

Finally, it was time to code what I had planned in my final description. Somethings I focused on were the appearance of the dot indicators, the transition of a new slide coming in, and the timing of the slides themselves. Overall, I think my decision to code the slide itself out earlier was very wise. It really helped me focus on improving the smaller details, as its users allows pay attention to the fine details. After I coded the transitions and improved the appearance of dot indicators, I has some time to have peer test it to make sure nothing in the interaction malfunctioned. I wanted to make sure that after all the time and effort I poured into this project, there weren’t any minor issues I overlooked. Some people expressed that the time of the overlay was slow and others said it was too fast, so I reached out to my professors for advice and did some research on my own. After this, I made any other minor adjusts that I found from my testing.

View my [**Final Build**](https://samihashoshi.com/idm241/final/build).

Process and Insight

**1. Style Tile**

Started with putting together a style tile for what I wanted the site to look like before I started on building templates. I looked at other recipe websites for inspiration and wanted to go with a lighter interface for readability purposes.

**2. Alpha Design**

Next I planned out what pages I would be including on my website and then began wireframing. Once I was finished with my low-fidelity

wireframes, I began to stylize the content on each page and figured out what worked best with the color palette that I chose.

3. Coding Out the Pages

After designing each of my pages, I then began coding each page in HTML and tried my best to replicate what I had planned for. A lot of this part

was making sure I stayed organized with my class names so that I could easily customize them using CSS. I enjoyed this part because I was able

to build off of my knowledge of HTML and CSS from previous classes, and the amount of time it took me to complete the alpha was far less then

term long projects I completed in the past when I was first introduced to these languages. This is a screenshot of the home page that I coded.

4. Implementing PHP

In this stage, I had to apply what I learned so far throughout the class into this beta assignment. I had to convert all of the HTML pages I made into

PHP files and preview my website using MAMP, a local server. This was the most intimidating part, but I found that if I looked back and reviewed

what we learned each week of the class, I was able to follow along and make sure I was doing the assignment correctly. I used PHP to make

creating, reading, updating, and deleting recipes and users possible as an admin on my pages. At this point of the project, everything was working

perfectly locally and the database I connected to my website was working.

5. Connecting to Bluehost

Since this would be my first time dealing with a database on my server, I was nervous that even though my site worked locally, it wouldn't on my

domain for my beta assignment. I imported my local database onto my Bluehost and uploaded the files into Cyberduck. I loaded the page, but it

was broken. If I could go back, I would've done this step first so that I wouldn't have run into errors late into the project. After screensharing with my

professor, we found that errors weren't showing up on the page so that's why we couldn't figure out what was broken. Once we figured that out, we

spent a lot of time debugging. Even though the code for my file paths and global files were working locally, they weren't working on my server and

that was the cause of the problem. After a few tweaks, my site was up and running and the database was creating, updating, editing, and deleting

users and recipes.

**6. Final Project**

The beta assignment helped me to code a lot of what I already needed for the final project, so I was happy with the progress I had made. At this

point, I needed to figure out how to merge two tables (files and recipes table) so that I could upload images of my recipes when I create recipes on

the site and have them display. I got it to work locally so when I uploaded an image it would save to my uploads folder and then display with the

recipe on both my user and and admin recipes page. I tweaked the tables for those pages a little bit so you could tell the difference between the

user and admin page after that. When I uploaded it onto my domain, everything worked except that the images weren't loading. I realized that one

of my functions I used in my code was making the image path incorrect, so once I fixed that, my images loaded correctly. I ended up with users

pages and admin pages, and in the admin pages, the user is allowed to create, read, update, and delete recipes and users. View my final project

here.

**The Solution**

What really helped me succeed in coding the site was by referring back to my own code and keeping things uniform while only making small

adjustments. For example, the code I built for CRUD for my users was the same for my recipes, and all I did was tweak what I was pulling from the

database and the variables I was using. I made sure the code I built for users worked completely before trying to go on to the next thing. I also

went back to my lessons and built my code by following the steps that we did week-by-week, so that I could make sure I fully understood what I

was doing without getting ahead of myself. Sticking to these methods made my process a lot easier, and you can see by the design on my site that

a lot of the structure was uniform. For styling, I also went back to my original HTML files and kept the same class names so that I didn't have to

worry about trying to tweak elements or forgetting how I styled them. These steps made my final project what it is and I can confidently go back

into my code and understand what I did.

**The Results**

By the end of the term, I was able to build pages using HTML, CSS, and PHP/MySQL and connect information on the site to databases that allow me to create, read, update, and delete content. I learned how to stay organized after making 30+ files of code by grouping my files into folders and

being very careful to remember the changes I made when switching from file to file. Transferring my local files to my domain was a complicated

part of the project that I worked through and overcame. If I had more time, I would actually make my login system work and create sessions for

users that login. I learned that there is a lot of time and planning that goes into using databases and even more when building a large scale

website. I also saw how using databases can organize a lot of information and are a powerful tool that I hope to keep implementing into future

projects