Broken-Link Inc.
Edwin Rodriguez
Computer Science 480
March 12, 2018

Open-Fridge.com Recipe Database

Hungry? Find your next meal here. Use Open-Fridge to search, create, favorite and comment on recipes. Also follow friends or family to see and interact with their recipes.

Have random ingredients lying around? Use our ingredient search from a database of 150 recipes to find the perfect meal suggestion for you.

Table of Contents

Intro

Client: Edwin Rodriguez

Company: Broken Link

Broken-Link is a 6 man company of computer science majors with no background in website design/management. The client, Edwin Rodriguez, in need of a recipe database, came to us with a website idea to create. The idea (pg5) was a Recipe Database that would help with finding items to cook. Our six member team and their main responsibilities were as follows:

Member	Main Responsibility
Matthew Silvestre	Scrum Master and Back-End
Pedro Ruelas	Back-End and Django Implementation
James Lee	DataBase Creation and Recipe Input
Jason Kaufman	Minor Front-End and Minor Recipe Collection
Chris Leal	Front-End and Recipe Creation/Formatting
Erik Gutierrez	Front-End and Django

The main goal of the project was to learn through first hand experience the conditions of real world development as well as the development technique SCRUM.

SCRUM is an agile framework most widely used to develop software. The members of each team work on weekly sprints to fulfill a major project deadline (called sprints). Ideally the members of the team would meet each day to discuss progress or any problems they encountered. However, due to scheduling conflicts of the 6 college students, daily meetings were not always possible, and random voice chats or message threads usually consisted of most of our

company's meetings. Most communication and updates on each persons individual contributions, in most cases, was left to the individual to report to the SCRUM Master.

In some cases following SCRUM was difficult and hard to do. It didn't always seem like the most time effective way to do things. In this development "framework" there aren't supposed to be any experts in a certain area. This makes it so that people are always rotating on what they are working on. If someone develops knowledge or a technique of doing something that allows them to be faster and more efficient at developing their particular area, under this technique it goes to waste as another person struggles to catch up or follow the other persons ideas. Since all of us had to learn everything before we could start to develop the site, the first two to three weeks of our 10 week project, were used to learning what was needed. If we were not forced to follow this guideline, it is possible there would have been less time wasted and more time allocated to actually designing and making our product. As you will see later, an extra week or two would have made a gigantic difference. Additionally, allowing for one person to be in charge of one specific area would allowed for a more even approach to development and most likely solved most of our development conflicts.

Problem

The problem given was to create a website that would simplify searching and creating recipes online. The website should be able to do a search for certain ingredients, and specific recipes. The point for looking for ingredients comes from the idea that sometimes you have certain ingredients in your fridge but you are not sure what to make. Instead of thinking and searching for recipes that contain the ingredients you have on your fridge, you should be able to just type the the ingredients and the site would give you a recipe that contains those ingredients. Our client wanted us to create a recipe website idea where a user can look up recipes as mentioned before, as well as a social media like interface allowing users to interact with each other. This would mean that the website should allow users to create a profile on which they could create recipes, the users would then be able to like other users recipes. The users should be able to follow others users in the site as well as comment on various recipes. All these interactions required an account on the site. Additionally, the site needed to be functional and aesthetically pleasing.

Design Approach

Stack Flow: HTML5/CSS > JS, JQuery and Ajax > Python w/ Django framework >mySQL

mySQL workbench: used to create and fill the database tables through forward engineering.

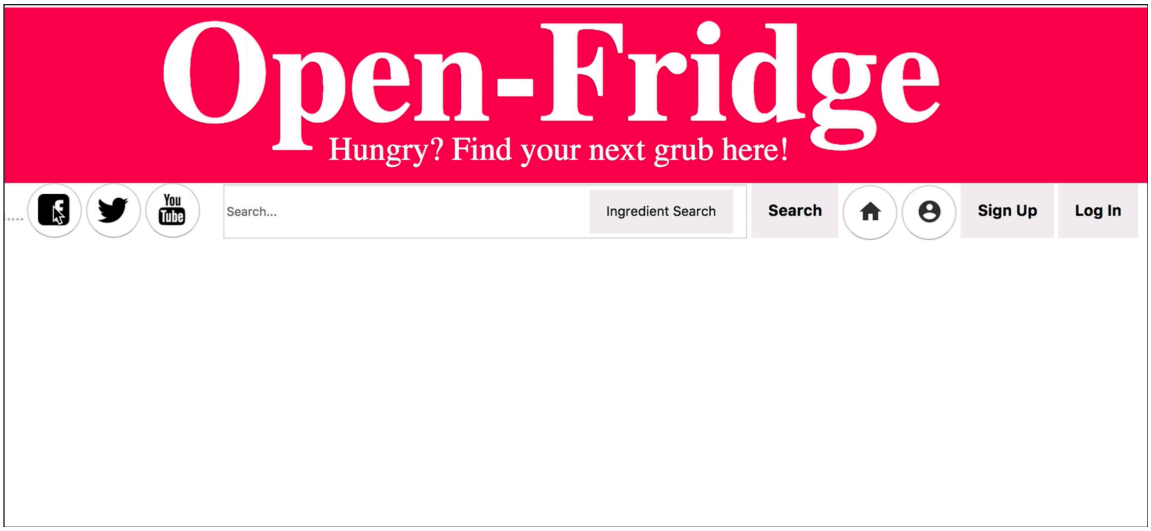
The design approach relied heavily on interaction with the client, we used the SCRUM method that we discussed in the beginning of the class. There was a meeting with the client every Friday to discussed what had been working on that week, as well as to talk about what new things needed to be implemented the next week. By introducing some of our own ideas and receiving feedback from the client we were allowed more opportunities to meet our implementations expectations with the client. We welcomed revamping the site and design to fit the needs of the client, this meant that the front the would end up being reconstructed multiple times over the development of the project as previously mentioned. The steps we took in development of the front end was based off of what the backend could accomplish to avoid making unused pages. Weekly we would set goals for the back end to accomplish and have our web designers create HTML pages to host the functionalities of the site. Initially we had everyone try taking a crack at designing the front end since we thought that we could all have a common idea as to what the site should look like. One the meeting we had all the users come up with their idea of what the site should look like, this showed that everyone wanted different things out of the site. After discussing the design for a coupe of the meetings, we decided on a very simple front page. We had a working search bar that would allow a user to access instances of information from the database and display them on a page, account pages and recipe creation

pages, commenting systems as well as favoriting recipes as well. Account interaction was crucial so we also allowed for users to follow each other for more access to recipes.

- # Front-End

For the front end the stack flow was HTML5/CSS > JS, JQuery and Ajax , doing so allowed for our front-end design to communicated with the database and with help of JS and Python dynamically create the needed html to populate the pages regardless of how much data was returned.

The first and earliest design approaches kept the site clean and easily navigable. The site would have been made to emulate an existing social media site since most users would already be familiar with its layout. This approach started off with a header, shown below, that would allow for users to navigate between each page on the site effectively.



Half way through the project the design idea changed, to what the site is today. This new design featured a more compartmentalized approach that would help with design on mobile and other small screened devices. This approach was more aesthetically pleasing than the pink and white site from before.

• Back-End

Initial tables were created to hold the basic Account and Recipe information. It was originally discussed that the passwords would be contained in the database as a hashed value, but the idea was scrapped as an account table that is provide by Django was used that automatically takes certain security measures . Each user has a username, password, and email space stored for them based off of user input when signing up for

the site. The table would also contain a space for containing the friend associated with the account, and a user id number would be automatically generated, and would serve as the primary key for the table. This account was eventually overtaken by a different user table called auth_user as it was a table that was provided by Django that already had security measures in place. This along with the table user_stats that was used as an extension to the auth_user table contained most of the information that was stored within the original account table with some changes to better implement changes that appeared further in development. The Friends attribute was replaced with Followers and Following attributes to assist with displays when the information was requested and counts for user information. Likes and an account image attribute were also added. The Likes attribute was made keep track of the amount of time a recipe was liked by users and to keep track of a user's liked recipes. The account image attribute was made to store the image path for a user's profile image should they need one but it couldn't be correctly implemented on time. Another table was made to contain the details of the recipe. A recipe id was generated for the primary key, the steps, title, and image path are created based on user input, and and the recipe id would be connected to individual ingredients to connect them to that specific recipe. The ingredients would have their own information (name, measurements, etc.) contained in a third table and compared to the recipe table to allow the display to know which ingredients to connect to which recipe.

Name	Steps	username	image	likes
Toast	Put In Toaster	fatcatmat	images/download.jpeg	2
Mango Smoothie	Blend a Mango	fatcatmat	images/Screenshot_from_2018-02-26_20-09-30…	0
Buttered Toast	Make Toast	fatcatmat	images/butteredToast.jpeg	0
Sexy Bread	Delicately heat bread up until golden brown.	Chef	images/o.jpg	0
Venison Bacon Burger	Cook bacon in a skillet over medium heat until b…	khxiii	images/Venison_Bacon_Burger.jpg	1
Vegan Pancakes	Preheat the oven to 250 degrees F. Whisk toget…	khxiii	images/VeganPancakes.jpeg	1
Ultimate Caesar Salad	Mince 3 cloves of garlic, and combine in a small…	khxiii	images/UltimateCaesarSalad.jpg	0
Tuna Salad	In a small mixing bowl break up the tuna with a f…	khxiii	images/TunaSalad.jpeg	0
Tequila Lime Pork Tenderloin	Whisk together the lime juice, tequila, orange jui…	khxiii	images/TequilaLimePorkTenderloin.jpg	0
Tender Italian Baked Chicken	Preheat oven to 425 degrees F (220 degrees C)…	khxiii	images/TenderItalianBakedChicken.jpg	0
Talapia with Mango Salsa	"Whisk together the extra-virgin olive oil, 1 table…	khxiii	images/TalapiawithMangoSalsa.jpg	0
Tailgate Chili	"Heat a large stock pot over medium-high heat.…	khxiii	images/TailgateChili.jpg	0
Szechwan Shrimp	"In a bowl, stir together water, ketchup, soy sau…	khxiii	images/SzechwanShrimp.jpg	0
Stuffed Mushrooms	Arrange mushroom caps on a medium baking s…	khxiii	images/StuffedMushrooms.jpg	0
Strawberry Spinach Salad	"In a medium bowl, whisk together the sesame…	khxiii	images/StrawberrySpinachSalad.jpg	0
Spinach Tomato Tortellini	"Bring a large pot of water to a boil. Add the tort…	khxiii	images/SpinachTomatoTortellini.jpg	0
Spam Musubi	"Soak uncooked rice for 4 hours; drain and rins…	khxiii	images/SpamMusubi.jpg	1

(a picture of our main database table “Recipe”)

Features

Features include:

- ## Recipe/User/Ingredient Search

A three option search bar that allowed for the user to select rather to search by recipe, user or ingredient. By searching a blank field, the user could browse through all recipes or users on the site. The ingredient search was unique in that it allowed for multiple ingredients to searched for at the same time (comma separated) which would return only the recipes that contained those ingredients.

- ## Account Page

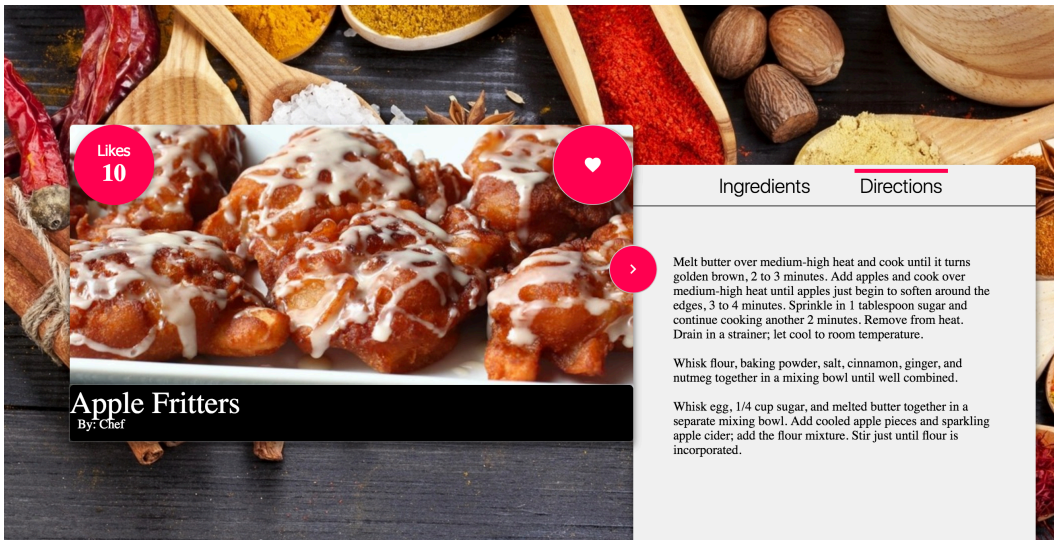
After signing up/logging in, an account page displayed the users info (including username and website stats), submitted recipes and friend list. Other users would navigate to this page to look at the users recipes and potentially follow them.

- ## Recipe Creation

Users would be able submit additional recipes. This page featured input fields for all relevant recipe information(title, steps, and ingredients). Upon submitting all info was sorted into tables in the database.

- **Recipe Display**

A slide out recipe display used for showcasing recipes as well its ingredients, steps and likes.



Product Architecture

Storing recipes was the primary concern, so the first task was to decide what would be stored for each recipe within the database, and how the database would interact with other components of the site. Eventually it was decided that each recipe would be composed of multiple ingredients that would be stored elsewhere, containing further information pertaining to how that specific ingredient interacts with the recipe itself through measurement of ingredients and preparation details. In addition to this, the database would store user account information. The initial information would be created from the user interacting with a sign-up page on the front end, and further changes would be made through its interactions with various other pages, such as liking recipes and following other users. The database would then need to be bridged to the front end, where the information stored within it would be retrieved as needed and displayed in a clean and easy to read manner, and the front end would need to be bridged back to the database through text fields and automatic processes that would update and store information within the database. In the planning of an actual site, more time would have been devoted to the security of user information on the site as well, but as that was not a concern at the time, security was left to its default level provided by the software used to create the site.

Implementation

• Front-End

It was in the early development stages that we decided to create the website from the ground up by using HTML and CSS. We believed that this would give us the best understanding of what designing a website would be like. The specific software we decided to use to create and test the front end of the website was Brackets, a relatively new web designing software that was compatible with Windows and Mac. The website needed to have the following pages designed in order to fulfill the client's requirements:

1. Front Page
2. Sign Up Page
3. Login Page
4. My Account Page
5. Other Accounts Page
6. Create Recipe Page
7. Recipe Display Page
8. Search Results Page

Once the pages were designed with HTML and stylised with CSS, the functionality of the pages would need to be implemented and that we used Django. Django allowed us to make certain items display when a user was logged in and it would also hide item when the user was not logged in. We were able to implement Django code to replicate some the templates in loops, that displayed the recipes result

page as well as the user search results page, so they would not need to be hard coded in HTML.

- # Back-End

The back end was designed to add the user functionality. Things like a search function, and create recipe page were not able to work without the addition from the back end. These functions also interact heavily with the database and connect that information with the front end. This allowed for pages to be filled dynamically based on the user credentials. This connectivity was done by using Django and Python to perform queries and loading data and importing the information to html templates.

The Search Bar worked by adding a string of text into the search bar, from here the bar would insert the string of text into a query and run a filter through the MySQL database to search for the selected term, the search bar also depends on a field in the HTML that also passes it information about the search, whether it pertains to a recipe, an ingredient or a user in the database. The search bar also run a multiple ingredient search that performs the same function multiple times, equivalent to the amount of ingredients separated by a comma.

The Recipe Result Page comes after the search bar. After filtering through the input string from the search bar the Results Page the page displays all recipes linked to that string that was input. This will display a quick briefing on the recipe along with that recipes name and an image.

The Recipe Creation Page holds a series of forms that each load data into the database. It requires several different items, name, steps, ingredients, and images. The way this information is stored is under different MySQL tables. The Recipe Table stores the name, recipe ID, steps, as well as an image path for extracting the image in the database. The Ingredients are stored in a separate table called Ingredients that stores an individual ingredient name and the Recipe ID that it is associated with.

The only way to access this page is by user authentication. Users can log in through a page that verifies the users account information using Django's own authentication methods, this is stored in a table predefined by Django. The Register Account information uses some predesigned Django functions to store user information on creation. The register account pages needs a username, email, and two instances of the same password.

After logging into the website, the user is sent to their own account page. The Account Page is filled with information given from the user and also displays different information about the recipes the User has created. The account page also displays the number of total likes and the number of followers that the User has. This is also affected by the amount of Likes inside the User table.Following Users was done by including a button on the account page this will then display the names of the other users that are followed on the Account Page.

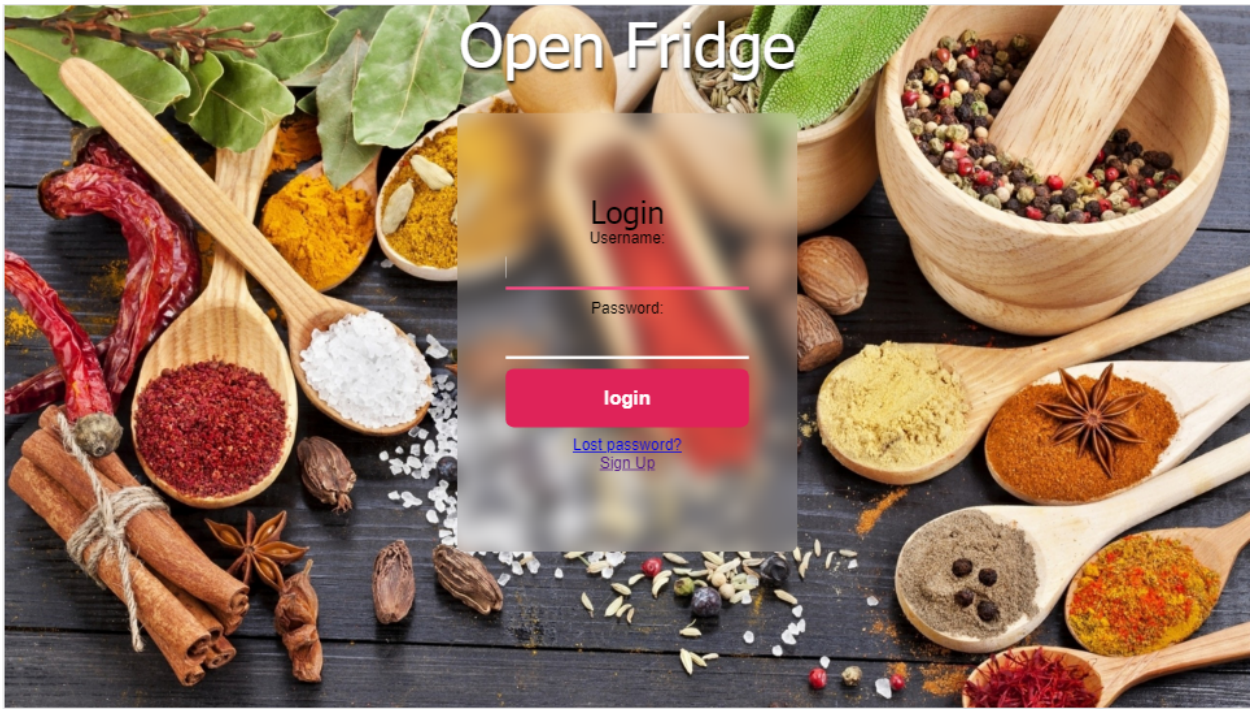
Testing Data

In order to test the effectiveness of our implementations, we had to run several brute force tests. Testing the search bar to return items manually entered in the database the first initial search bar that was made checked if the word was inside the name of the recipe. As we updates it stayed the same when searching for recipes, but search for ingredients using an exact string.

Testing once the functionalities were completed was done by making a dummy account and adding recipes and information to the database through the website. This was then confirmed after checking database if it had been updates. Once confirmed we tried searches and liking certain recipes, from the individual recipe pages. We tested to see if these pages also displayed the exact details the page required. The testing was just creating edits and additions to each element that we created and checking if the confirmed changes also affected other pages.

Description of Final Product

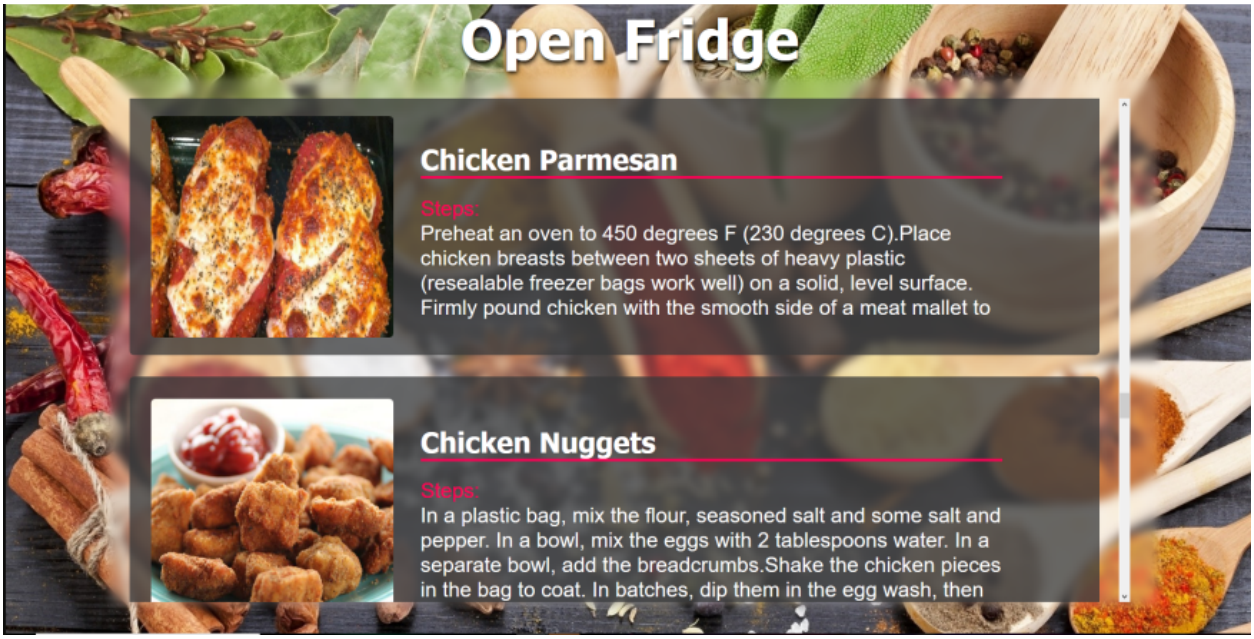
The final product is both a recipe database and a social platform. Beginning with the home page, the user is greeted with a search bar accompanied by a food theme background. From there, the user has a few courses of action. With the search bar, the user can search the recipe database using recipe names and ingredients. In addition, the user can use the search bar to find specific users that have registered accounts.



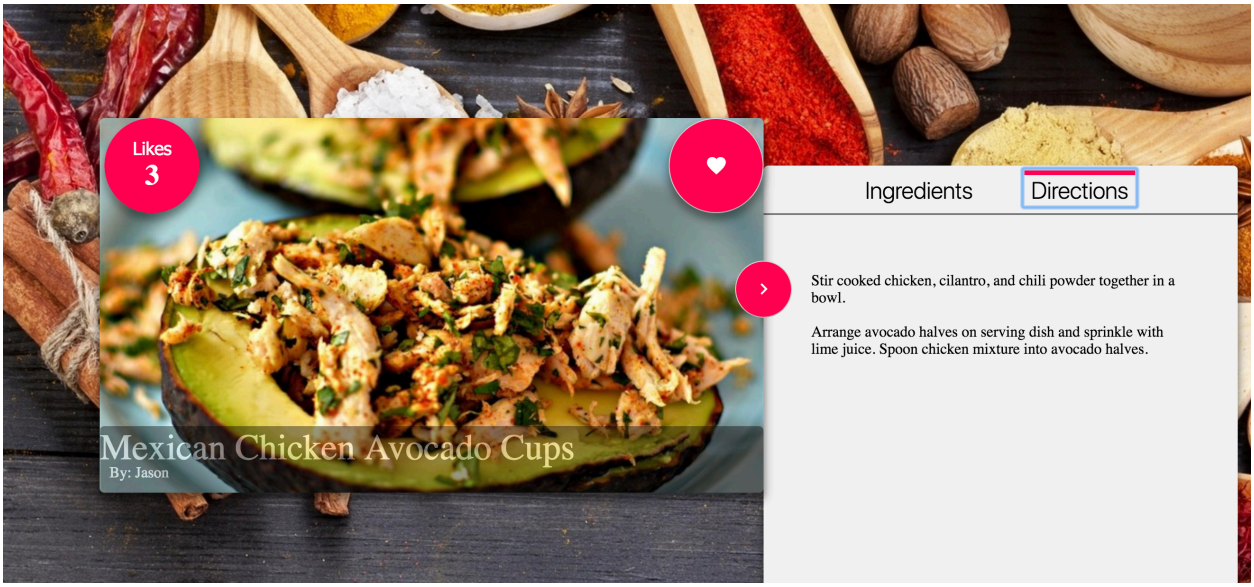
The initial design for the “Log In” page(pictured above), when presented to the client in week 7 is what eventually lead to the overhaul of the entire redesign of the site from the ground up.That’s because of the very cool glass panel effect that that was added through CSS and the simple and minimalistic aspect and feel it gave the site. Taking on this new style for the site would require a lot of new techniques applied to the HTML templates along with a lot of new CSS code that would make all the pages have the glass panel effect with all the contents housed

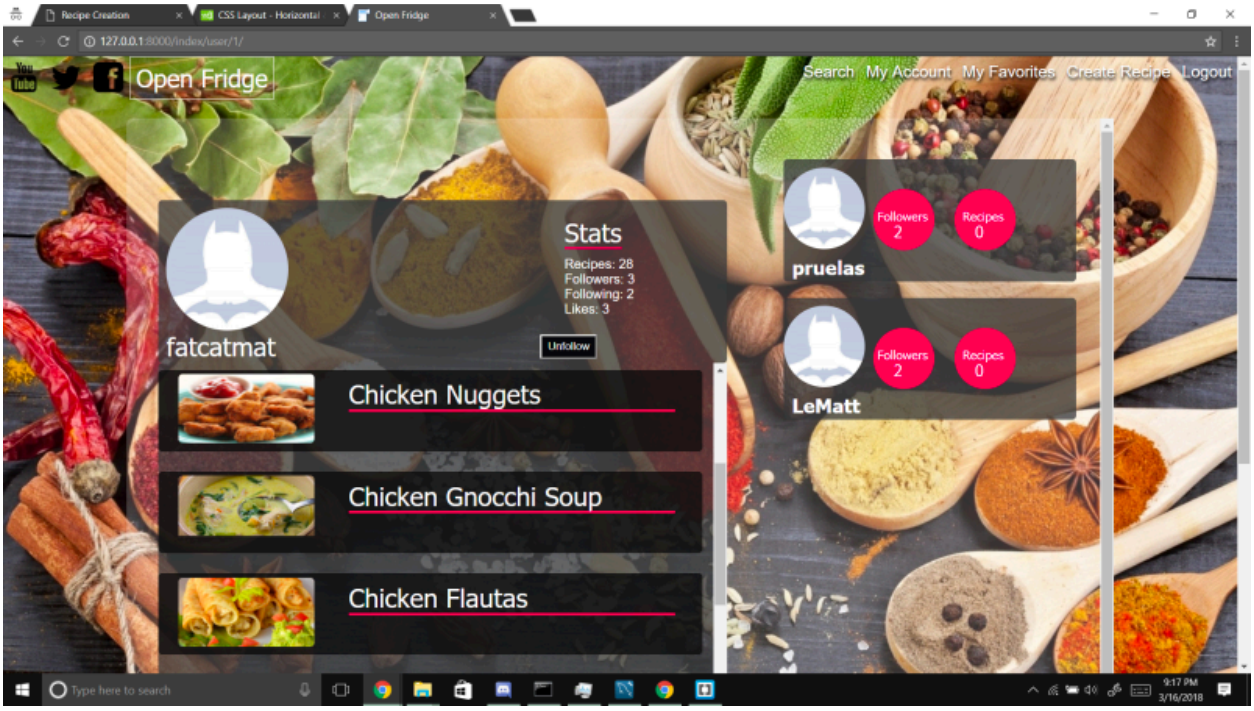
in the center of the glass panel. This made the website stand out and look like something that was made by someone that hose experience with making websites. The reality is that this was our first real break taking an aesthetic approach to the site, and the front end team was finally making strides with feel of the site.

Once a user created or logged into their account, they were immediately redirected to the main page. When in the main page, a user could press search on the search bar, they will be taken to a new page that will displays the search results. If the user searched for recipes, they will be able to click on any recipe name from the search results. For the recipe results page, the design was based of a interface where the user could scroll down the list of recipes he or she had just searched for. The user should be able to look at an image of the final result of the recipe along with the title of page and a brief description of how the recipe is made.



If the user does so, they will be taken to a new page with the title of the recipe, a picture of the recipe, the list of ingredients, and the instructions. The user is able to favorite the recipe by clicking a heart button. Below the recipe, the user can create and view comments pertaining to that recipe. This form of displaying the recipe was introduced by Chris, and it had been positively reinforced by the Product Owner at some point early on in the development of the site. Unfortunately when the page was resized to fit in smaller screens it was not displayed as it should have. Some elements were so far from their original location, that when it came down to the presentation, this was what ended up getting us in trouble with the client.





Like the recipes, if the user searches for other users, they can click on a username in the search results. By clicking on the username, the user will be brought to that user's profile page. There, they can follow that user by clicking a button. The profile displays that user's recipes, number of followers, and the number of other users they are following. This page took elements to the from the recipe search and it was implemented in the recipes created by the users panel. The friends were also similar but without being held in the container, that gave that part of the user display page a more open feel. It was once again all housed in a glass container. For this particular page the "Open Fridge" title page was moved right next to the Social Buttons, this allowed the user to get back to the main page without having much trouble.

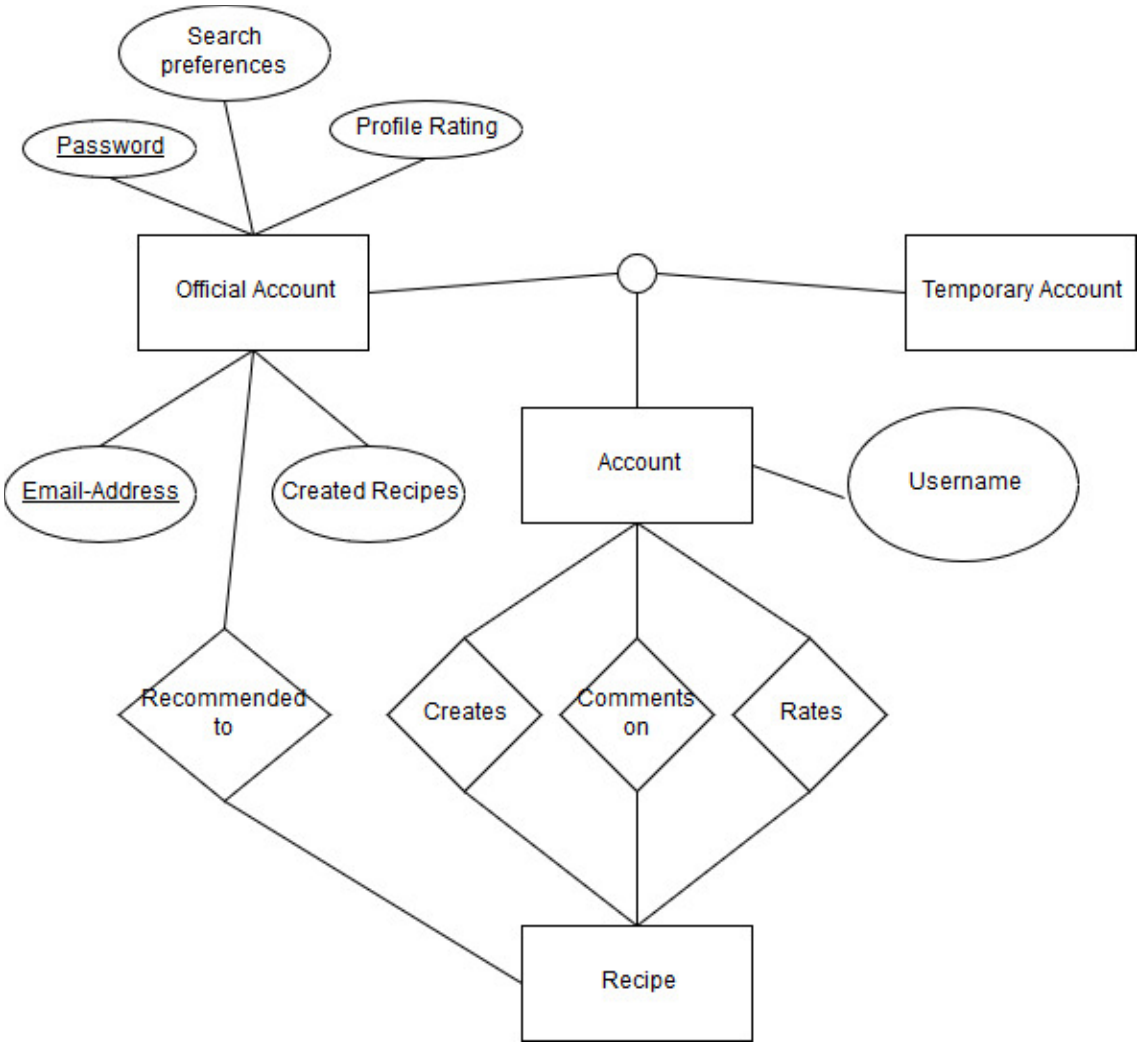
Project Time-Line

- ## Sprint 1 - Week of January 26

This week we set the initial base for the project. We were required to come up with a software stack that gave an overview of the software we planned to use from front end to back end. This included deciding on the type of database we were to use, how we were going to host the website, which software we were going to use to interact with the database and front end, and the software we were going to use to build the webpages. In order to be able to determine what software stack would be best for our project, we researched multiple sites and attempted to gain a brief but thorough understanding of what was required to create and host a working website. Through research and guidance from Professor Rodriguez, it was learned that MySQL could be used for the database and python could be used along with Django to serve as a server-side web framework that would fulfill the requests from the front end and communicate with the database. Additionally, we would utilize Html along with CSS to create webpage. Also Amazon Web Services (AWS) could be used to host the Django application. This led to the creation of the original stack that would be used and built upon through the development of the project.

During this sprint, we were also required to create a data model that clearly expressed the overall functionalities of our website. To be able to this, we had to first obtain a general overview of what the project would require and

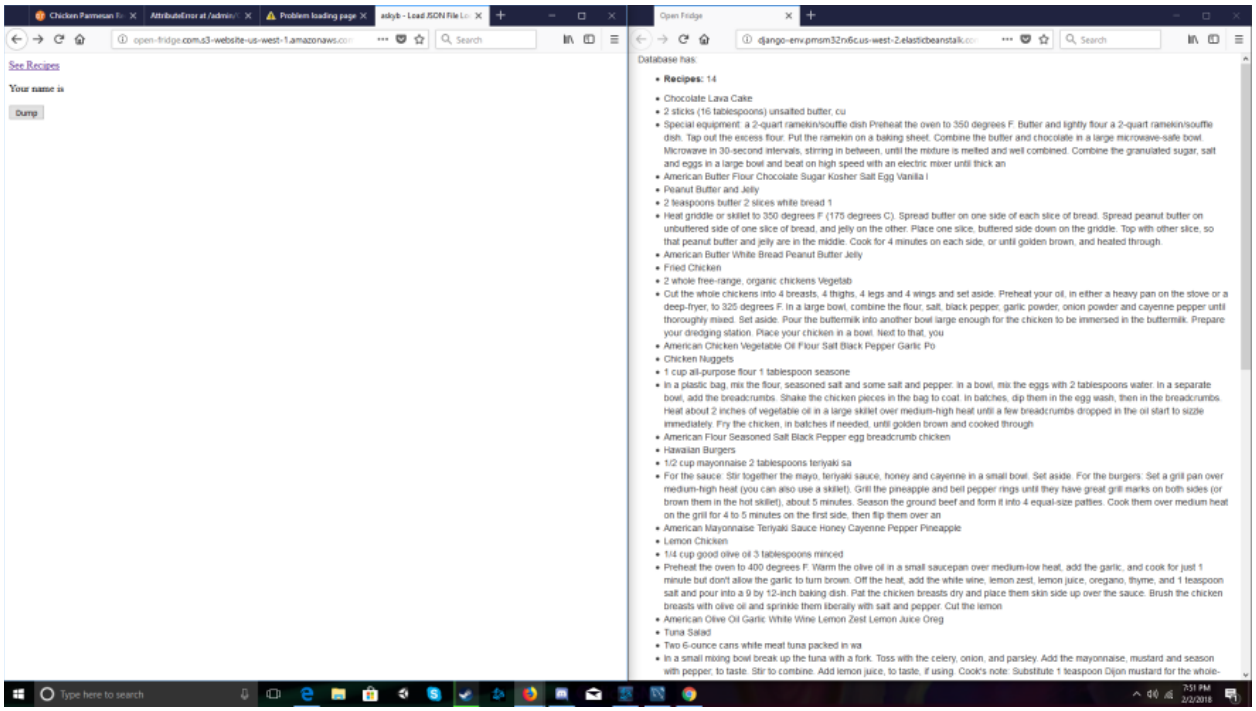
then start determining our project's entities and attributes. We then had to understand their relationships with one another to be able to create the data model. This led to the creation of the following data model that was used as a basis for our project's database.



The model would undergo future modifications as the project continued and new needs coupled with more efficient processes were discovered, though the original model outlines the most basic and concrete information that would remain fairly constant throughout the design.

- ## Sprint 2 - Week of February 2

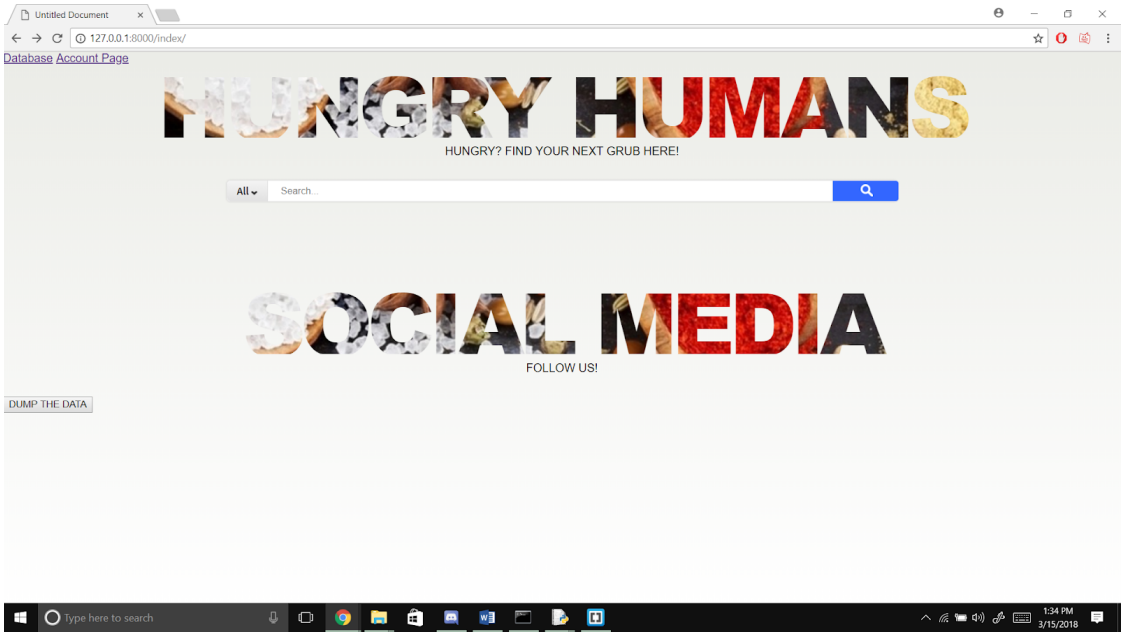
This week we began the necessary installations and setup for the tools we would be using during the project. This included setting up a server to store the database information and following multiple tutorials to work with things like MySQL and Django. Most of the group was fairly unfamiliar with these tools, and there was a lot of time spent simply watching online tutorials and learning how to best utilize these tools to suit our needs. There were also several issues allowing everyone to access the server, as only a few group members seemed to have access to it at any given time, which hampered progress in the development of the database and front end. The main goal for this sprint was to set up a server to host the Django application on as well as being able to show the current contents of our database. We unfortunately were unable to accomplish this before the meeting with the client as we were having a hard time learning the ins and outs of deploying a Django application using Amazon Web Services. Fortunately, we were able to figure it out before the day ended and by the end of the day, Friday, we had the first iteration of our website deployed and working. The following image show how our website first looked after its first deployment.



At this time the website only showed the recipes stored within our database, and the contents for the webpage of the database were actually retrieved from the database itself by using a form to submit a request to call a Django view that returned a query of all the present objects within our database. The html then had a python for loop that would iterate through each object displaying its information onto the webpage as text with not CSS. As a result of this, the webpage at this point wasn't static but dynamic and would be able to display any new entries that were added to the database from the Django admin page that was present at the time.

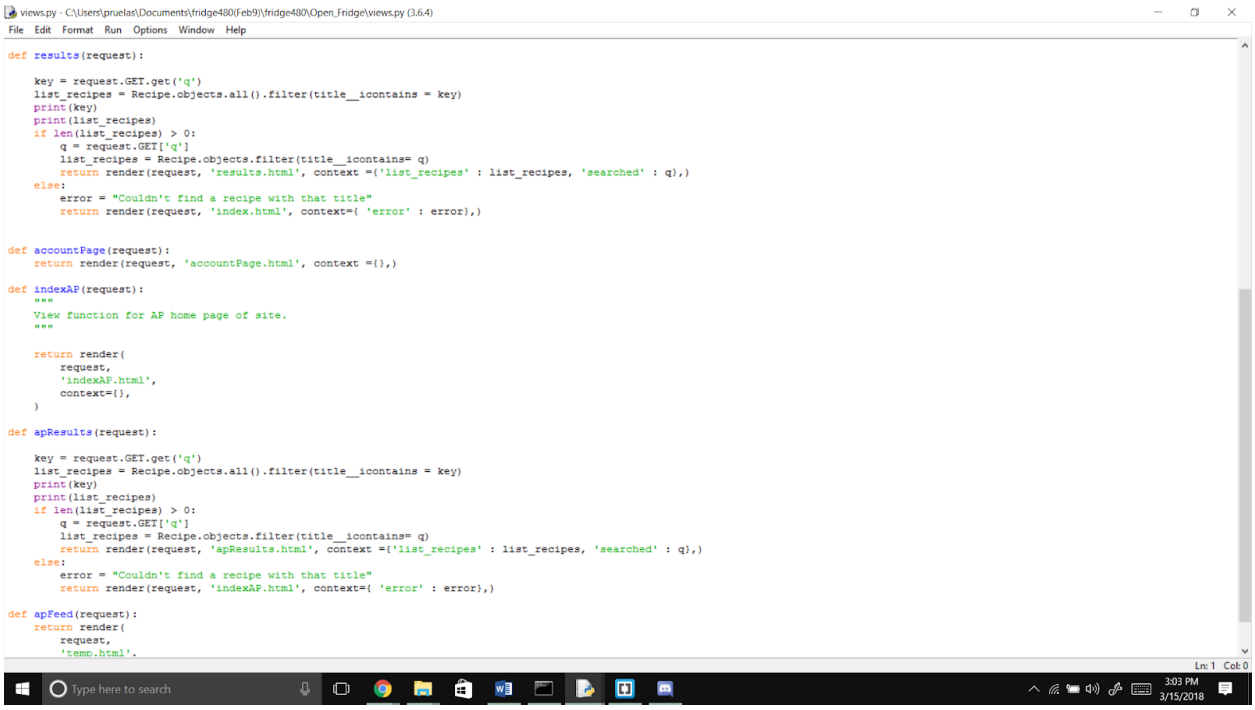
- **Sprint 3 - Week of February 9**

The original design to the website (below) was Chris's first attempt at Front-End development, without any prior knowledge, the site didn't look to good but, rather, was supposed to look cool, by his standards. The client initially rejected the website name, saying that it reminded him of zombies rather than of cooking a meal. The design was also rejected, noting that future designs should be made more practical and simplistic.

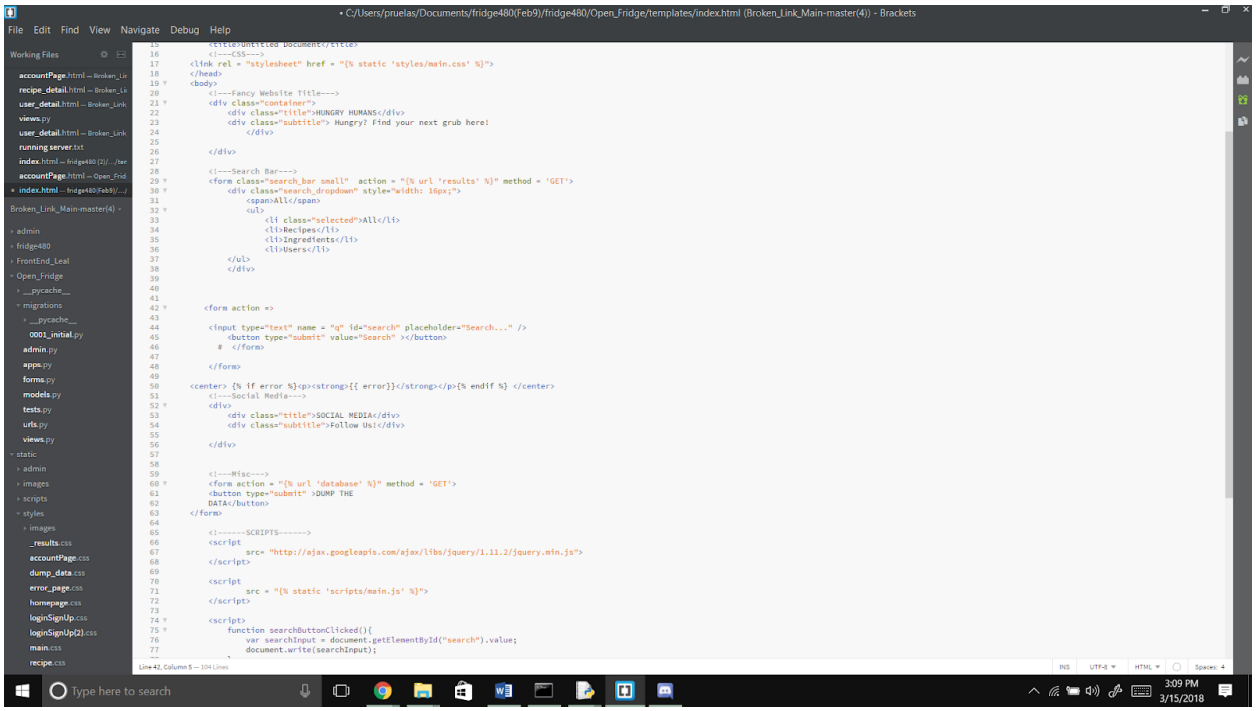


In order to get the search by title to work, we had to create a Django view that accepted the entry from the search bar and then queried through the entries using the string that was sent through the form. The view that was created to handle the 'GET' form was called results and can be seen in the screenshot of our then views.py file. The view as explained retrieved the string entered onto the search bar by the user and returned a list containing all the recipes within our database that contained the specified

string within its name. If no recipe contained the string specified, a string was returned to the index page and displayed stating that no recipe was found with such title.

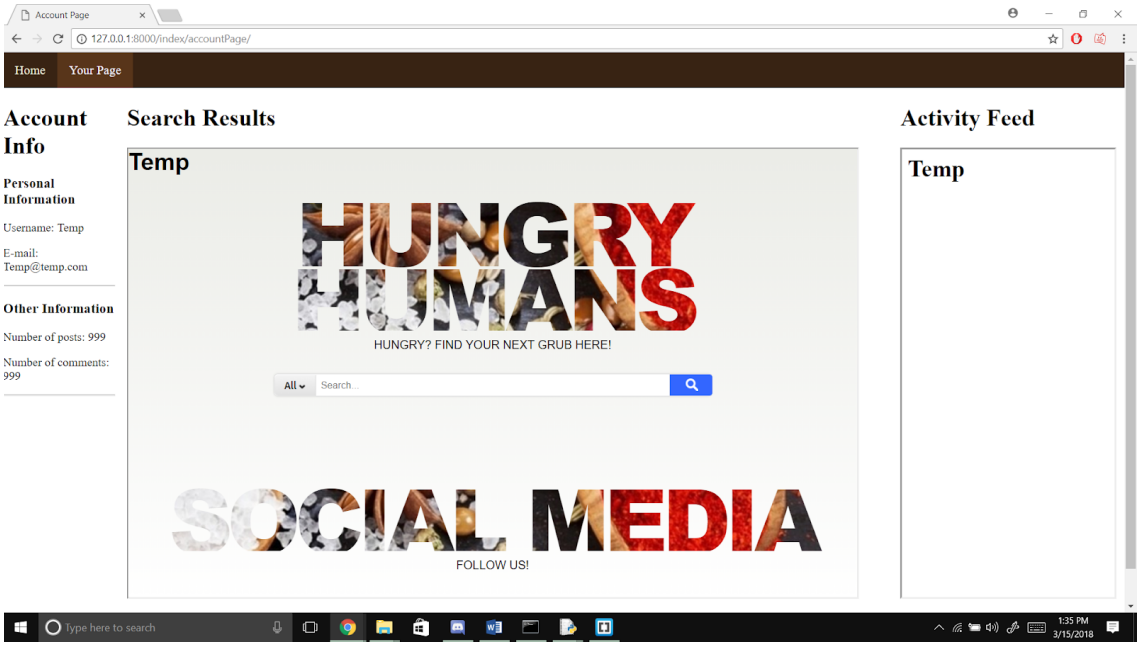


Unfortunately in this sprint, the search bar options were not working so only search by title functioned correctly. As seen in the html below, there was already a drop down menu that essentially allowed the user to select the appropriate option for searching denoted by the <li> tags, but at the time it was unknown to us how to store that option and send it along with the form to be able to determine the option that was selected. As a result, it can also be seen in the results function in the views.py file above that no if statements were present to return different queries for the different options available.



The initial account page (pictured below) had four parts to it. The first was the navigation bar located at the top. The purpose of this bar is to help the user navigate through different pages. In the image below, the present options are “Home” and “Your Page.” Since the current page is the account page, “Your Page” is highlighted. If the “Home” tab was clicked, the user would have been taken to the website’s main page. The second part is the “Account Info” located on the left of the page. This area was reserved for information pertaining to the user’s account. It was meant to display things like the username, e-mail, total number of recipes, and the total number of comments. The third part is the “Search Results” area on the center of the page. It was planned to have the search bar in the navigation bar at the top of the screen. The user could search for recipes without having to leave the account page. The search results would display under the “Search Results” area instead of going to another part of the website. The last

part of the account page was the “Activity Feed” on the right of the page. This is where a user would be notified of the various activities other users would perform. This would include the users following other users, users liking recipes, and users commenting on recipes.



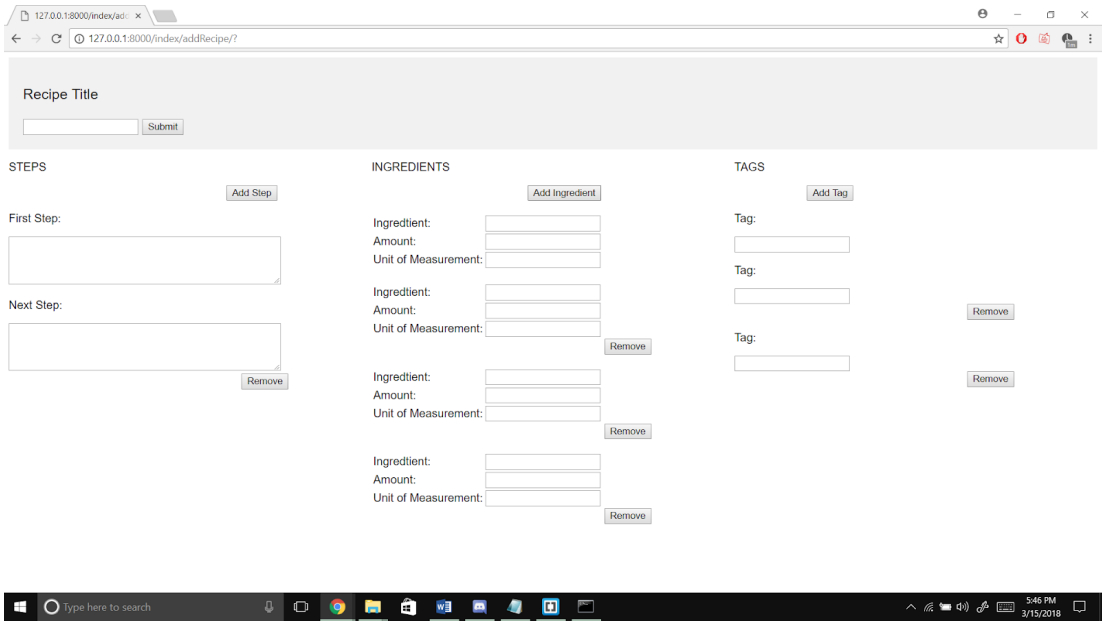
This design was ultimately rejected and revamped to later fit in with the rest of the sites theme. The main features of this page, except, the activity feed were implemented in the final site. The activity feed was removed because, as our site developed further, strayed away from the initial “twitter-like” site we tried to create.

- ## Sprint 4 - Week of February 16

Pre-loading the database with some stock recipes was bound to be a somewhat time-consuming task, and the easiest method that we found of doing it was through MySQL Workbench system. It allowed us to view the current contents of the database and make additions, removals, and alterations by attribute column for each individual tuple. The changes were then converted into SQL script by the workbench and applied to the current script that made up the tables. This proved to be much faster than creating all of the insertion script by hand, especially for quick editing.

- ## Sprint 5 - Week of February 23

During this sprint, an attempt was made to create a recipe creation page. In order to be able to make this page, research was conducted in order to learn how to dynamically append new unique forms to an already existing form using javascript that could be individually deleted. After research, it was learned that by appending divisions with unique IDs to an already visible division by using javascript functions which appended a child to a determined parent division. It was also learned that by using javascript functions one could delete divisions by using their unique IDs to ensure that only the desired division was removed. This was applied to the recipe creation page and the image below shows of what was created for the recipe creation during that week.

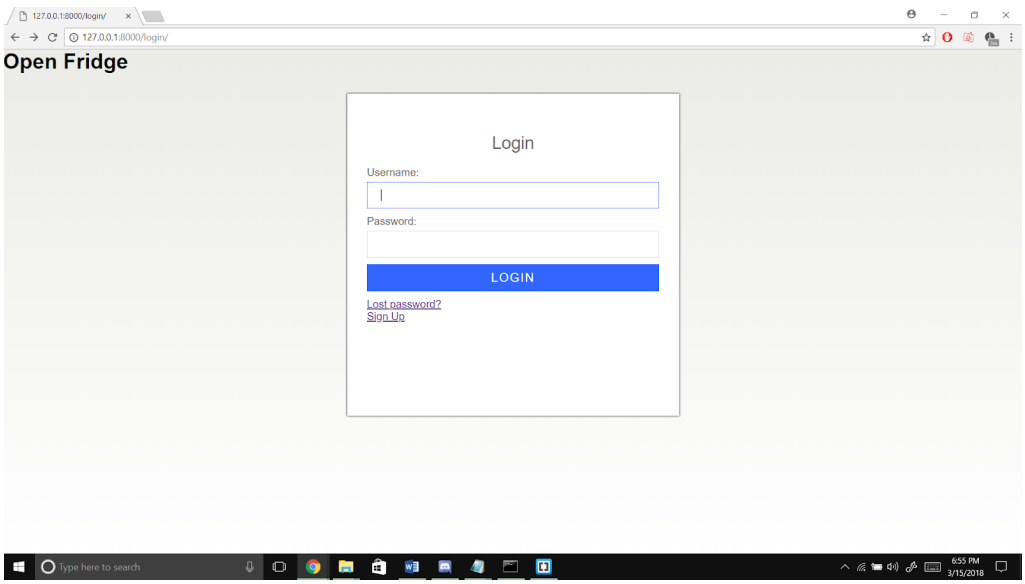


Unfortunately while this html was being created, the Django views required to 'POST' the data hadn't been created and as result was unable to be implemented in time. Also during its construction, its compatibility with Django wasn't heavily considered and as a result ended storing the inputs of the new forms in lists. It was never determined if this would actually work with Django as a better way was figured to implement a creation page using forms was discovered and as a result this page was never used again beyond this sprint. It appears that failing to properly implement this onto the Django application was greatly due to a lack of consideration of how it would work with Django, and lack of knowledge of better ways to implement the forms using Django.

- **Sprint 6 - Week of March 2**

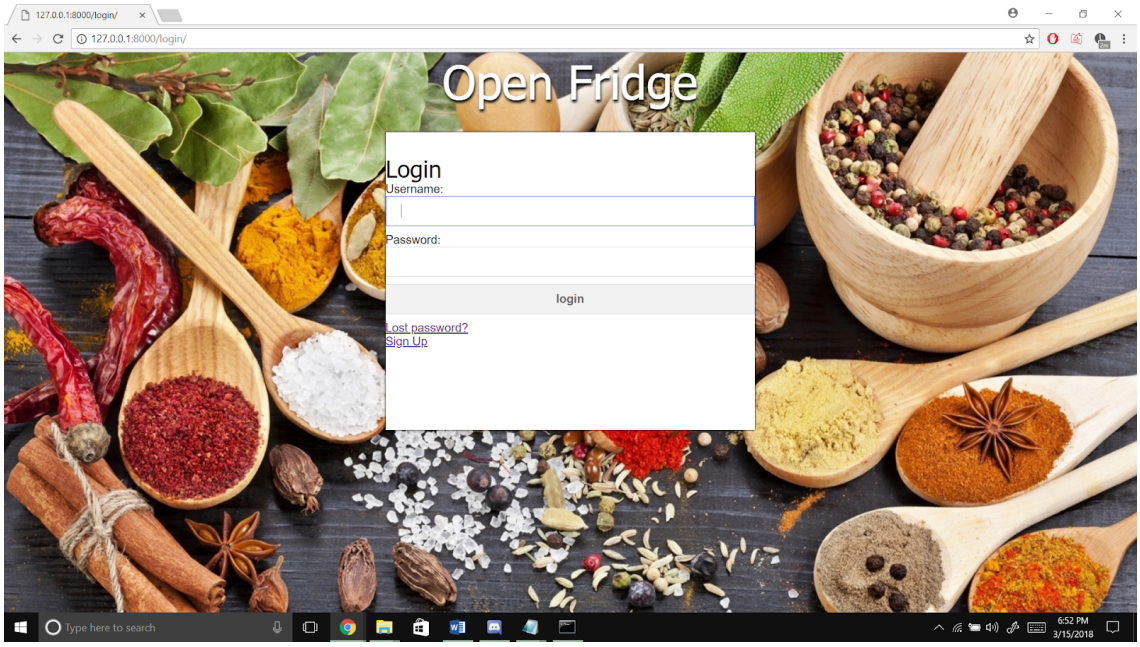
The Recipe Creation Page was finalized here after adding the ability to add images to the recipe. Several tests were made we were able to finally confirm this to work. This page still needed a style template at this stage. Our Comments table was added into the database to associate with the Recipe and allow a form to handle this information.

This sprint an attempt was made to improve the CSS for the login and sign up page that previously had little to no CSS attached to it. The login page was suppose to resemble the same login box presented below.



Unfortunately, inefficient communication, along with a lack of proper time management, and the ineffective use of Github prevented us from being able to quickly and effectively apply changes that were being worked on. This was an issue that was

prominent throughout the development of the project and was a huge detriment to the effective adoption of the scrum evolving sprints. As a result the login page (pictured below) did not work when meeting the client, it was eventually fixed later and was present in the final product.



The White space in the initial website was very bland, and any solid color was also not too kind to look at. These were replaced with the above to make a more pleasing and lively effect to the website. Adding shading to the fonts and the objects in each page produced a more pop out feel to the site as well.

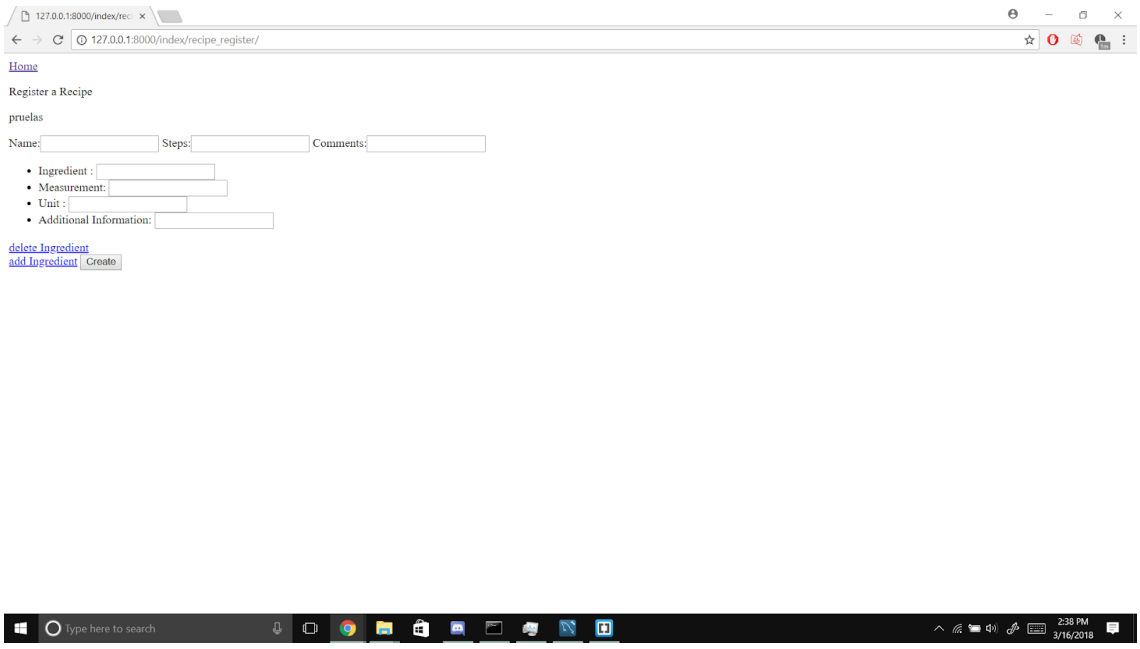
Through out the rest of the week various changes were made in an attempt to refine and cement the sites design. The first image below, an early form of the search bar that would include a smaller search bar for specified ingredients, this however caused some issues with our knowledge of Django and implementing the search in this bit and was also scrapped. The overall layout of the page followed the same structure

just with a background and the removal of the pink header. Additionally, we considered adding a recipe carousel (pictured below) that would display different recipes and suggest them to users on the main page.

Recipe creation as displayed in the image below was done by utilizing two different forms. A formset to handle multiple individual ingredients and a form to handle the names and steps of each recipe. The problems with this page was that they lacked css styles to remove the bland look and that the steps portion was very small. This made the steps section seem like it would only take one step at a time. This had to be worked on. Our assumptions were based off the size that this field would be storing in the database as a string with a max of 300 characters which even then proved to be too small. Our approach to fix this in the next week would involve increasing the set size of the table field and changing the form for recipes altogether.

Adding and removing ingredients had involved introducing a new addition to our webstack. Initially we attempted prompting an an amount of ingredients to display a

specified set of fields for ingredients, running a for loop in the html did in fact display the fields but we ran into the django actually storing only the last ingredient field in the page. We then resorted one extensive javascript work would be needed to produce a multiple set of forms to be stored into the formset, the easier way resolved to using jQuery to duplicate these forms dynamically with a plugin that allows it to access and mutate django functions. After this was completed as in the image below the add ingredient and remove ingredient buttons are found that will duplicate that field and allow for the recipe to store the ingredients portion successfully using the formset. The Recipe Form which is from Name to comments needed to be changed to a model form.



- ## Sprint 7 - Week of March 9

During this sprint, the main goals were to allow users to follow other users, allow users to comment on recipes, allow users to search by user and ingredients and to fix the CSS display issues that were present in certain webpages. In order to allow the follow button, a table was added to the database that would keep track of the follow relationships between users. Also, in order to allow for the follow button to work, functions were added to the views.py file that removed or added entries when a user unfollowed or followed another. It was also necessary to modify the user detail html and add JavaScript functions with AJAX and jQuery to be able to dynamically display and update the corresponding follow and unfollow buttons. The next main goal was to get the user and ingredient search working. In order to make this work, research was conducted and it was learned as seen below that using the select html tag and the option html tag, the string value of the selected option could be stored and passed to the corresponding function in views.py to correctly return the desired search results.

After the meeting with the client additional changes were made to polish the site before the presentation. The changes made were implementing user stats, implementing recipe images, implementing liking recipes, finishing the implementation of the comment system, and adding html and CSS to webpages to attempt to improve their visuals. The stats that initially displayed with default values on the webpage of each individual user were made to display the information stored using requests that called on defined functions in views.py that queried the information that was stored in the

User_Stats table within the database. The webpage displaying the information of the logged in user, such as the user's stats, the recipes created by the user and the information of the user's being followed by the logged in user, were also made to update dynamically by using functions define within the views.py file and retrieved the information by querying the User_Stats, the Following, and the Recipe tables. To allow the comment system to work properly, JavaScript functions along with modifications to an already present comment table were made. The comment table was changed to include a column that provided each comment entry with a unique primary key. This was necessary as unique primary keys allowed for the JavaScript function that deleted the comment to use the unique ID the comment entry was initially given in the database an as argument to allow the corresponding view to identify the correct comment entry that needed to be removed.

The variance in size of some images, or text fields (usually the recipe step process) caused some issues for the site. The CSS and HTML was reorganized and restructured to avoid unseen changes on the web client because of image or text field size. Scroll bars were also added in various areas as a "safety net" just in case the data still was too big for the size of the containers. Additionally, when we were creating the main site, resolution and differing window size did not occur to us. More CSS changes were made to make sure that website would be able to function at different screen sizes and resolutions.

Conclusions

Considering that our group was entirely inexperienced in website development, based on the amount of new languages and techniques we learned while making this website, the outcome of the project, at least personally, was not bad. Yes, the website did not turn out as any of our members had hoped. The decisions that we made during the project helped us learn from our mistakes. As mentioned earlier there were a few issues included; implementation, git storing and poor communication prohibited us from making easy progress. Our main front-end developer, Chris, had a unique issue that kept him from easily connecting to the mySQL database which made him unable to see his changes until implemented into the full site, which made the development process slow and tedious. Coming from various other school and group projects, each member of the team knew how to use git, but for some reason lacked any structure. For the first half of the project this was a key issue. For the majority of the project complete communication between all members was really bad. Because of this some members, predominately front-end, accidentally worked on the same thing during a sprint. As a result one or two members work on multiple occasions was wasted. As a result, some members became frustrated and hesitant with working on tasks that weren't exclusively theirs. Despite our shortcomings, given the opportunity to start this project again with the knowledge that we have now, it is certain that our finished product would greatly outshine our previous one. As a result of this project, most of the team plans to continue to learn front/back end development, in hopes that a future project will look a lot better and have all the functionality that it originally was meant to have.

Suggestions for Future Work

• Instructional Videos

One idea for future implementation would be to give users the ability to upload instructional videos. The videos would complement the instructions for the recipes. Anyone who wishes to not read the instructions or would like something supplemental would then be able to watch appropriate demonstrations to make the recipe easier to follow.

• Improved Social Media Functionality

Although the initial groundwork was laid to have "social media"-like functionality, those interactions were not as polished as they could've been:

1. Users should be able to interact more easily, (for example, through "liking" recipes or commenting on recipes). Due to front-end design flaws, the features, though present, were hard to actually see. By improving the functionality as well as appearance users could achieve a more social media type of experience.

2. Expanding on this idea, users should be able to send messages to one another. These would be similar to e-mails in that the messages are sent to an inbox (similar to facebook). To allow the ease of use, the inbox can be sorted

and searched through, as well as cleaned up according to the user's discretion. To reduce traffic to a users inbox, the chat function should be limited to users who are considered friends on the website.

3. The friending functionality in our final product was present but nothing was done with it. Perhaps by adding a twitter-like feed system to the account page the interaction and activities of friends could finish that concept. Seeing that will be necessary to be friends to message a completed friend system is crucial.