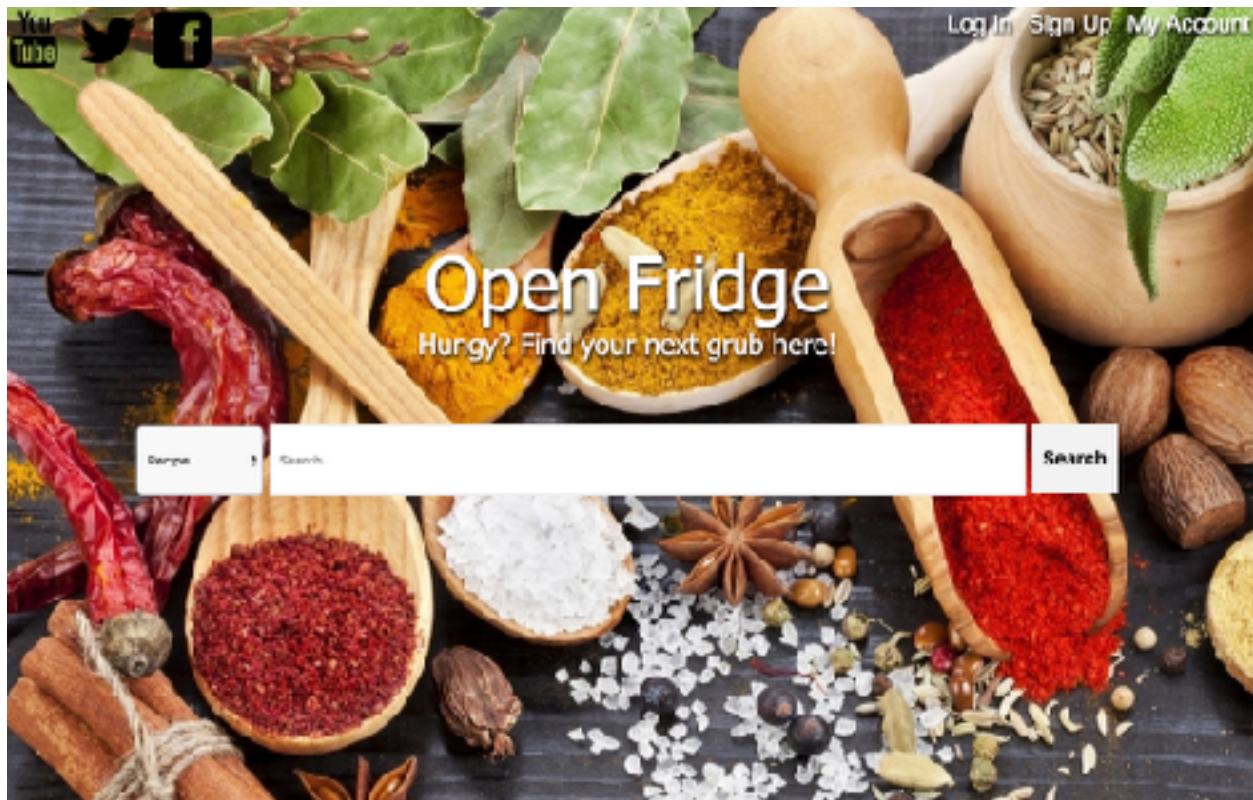


Open-Fridge.com Recipe Database



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 to find the perfect meal suggestion for you.

Table of Contents

Intro	page 3
Problem	page 5
Design Approach	page 6
• Front-End	
• Back-End	
Features	page 10
Product Architecture	page 2
Implementation	page 2
• Front-End	
• Back-End	
Testing Data	page 2
Description of Final Product	page 2
Project Timeline	page 2
Conclusions	page 2
Suggestions for Future Work	page 2

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 person's 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 person's 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 allow 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. 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.

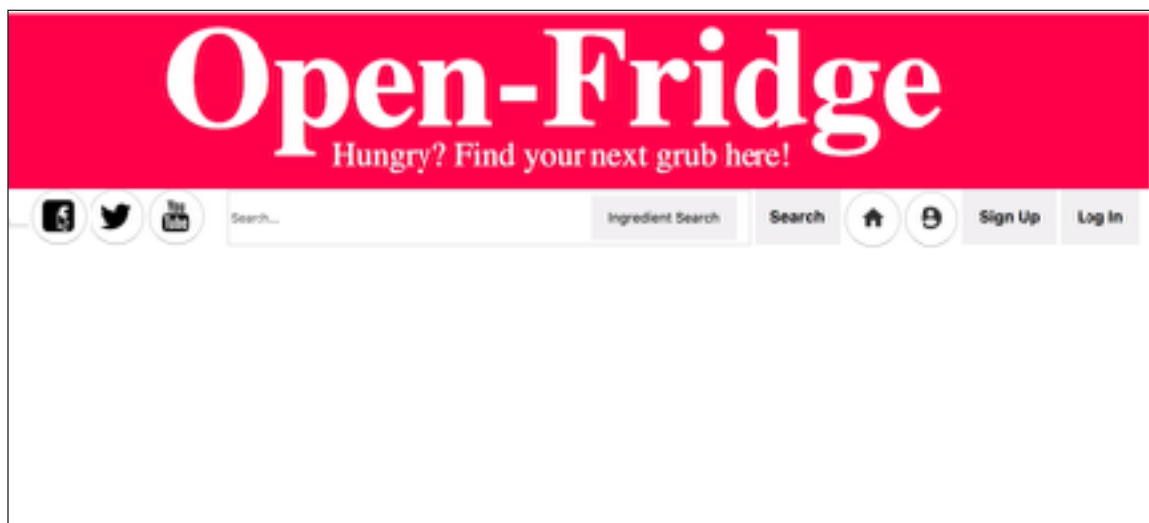
The design approach relied heavily on interaction with the client, we used the scrum method that we discussed in the beginning of the class. There was a meeting with the client every friday to discussed what had been working on that week, as well as to talk about what new things needed to be implemented 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**

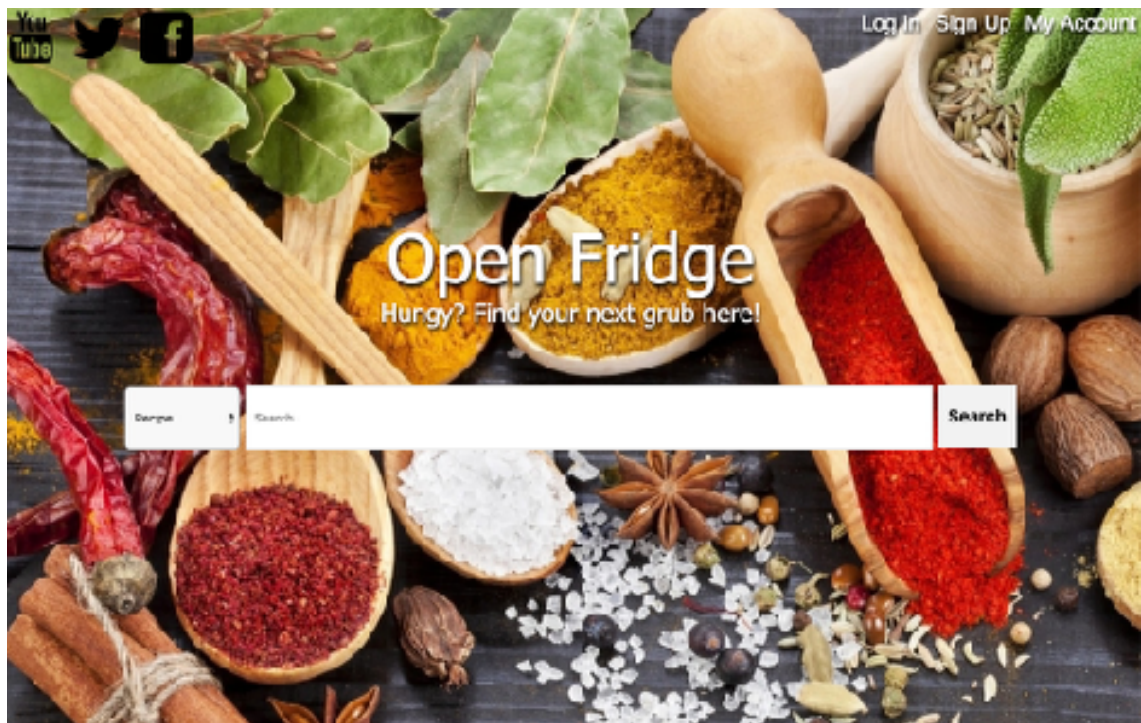
For the front end the stack flow was HTML5/CSS > JS, JQuery and Ajax , doing so allowed for our front-end design to communicate with the database and with help of JS and Python dynamically create the needed html to populate the pages regardless of how much data was returned.

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 above, 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

For the back end

Name	Steps	username	image	likes
Toast	Put in Toaster	fatcatmat	Images/Download.jpeg	2
Mango Smoothie	Blend a Mango	fatcatmat	Images/Screenshot from 2016-06-25 20-09-30 ...	0
Buttered Toast	Make Toast	fatcatmat	Images/butteredToast.jpeg	0
Sexy Bread	Delicately heat bread up until golden brown.	Chef	Images/sj.jpg	0
Venison Bacon Burger	Cook bacon in a skillet over medium heat until b...	khadi	Images/Venison_Bacon_Burger.jpg	1
Vegan Pancakes	Preheat the oven to 250 degrees F. Whisk toget...	khadi	Images/VeganPancakes.jpeg	1
Ultimate Caesar Salad	Mince 3 cloves of garlic, and combine in a small...	khadi	Images/UltimateCaesarSalad.jpg	0
Tuna Salad	In a small mixing bowl break up the tuna with a l...	khadi	Images/TunaSalad.jpg	0
Tequila Lime Pork Tenderloin	Whisk together the lime juice, tequila, orange ju...	khadi	Images/TequilaLimePorkTenderloin.jpg	0
Tender Italian Baked Chicken	Preheat oven to 425 degrees F (220 degrees C)...	khadi	Images/TenderItalianBakedChicken.jpg	0
Talapia with Mango Salsa	*Whisk together the extra-virgin olive oil, 1 table...	khadi	Images/TalapiaWithMangoSalsa.jpg	0
Tailgate Chili	*Heat a large stock pot over medium-high heat...	khadi	Images/TailgateChili.jpg	0
Szechwan Shrimp	*In a bowl, stir together water, ketchup, soy sau...	khadi	Images/SzechwanShrimp.jpg	0
Stuffed Mushrooms	Arrange mushroom caps on a medium baking sh...	khadi	Images/StuffedMushrooms.jpg	0
Strawberry Spinach Salad	*In a medium bowl, whisk together the sesame...	khadi	Images/StrawberrySpinachSalad.jpg	0
Spinach Tomato Tortellini	*Bring a large pot of water to a boil. Add the tort...	khadi	Images/SpinachTomatoTortellini.jpg	0
Spam Musubi	*Soak uncooked rice for 4 hours; drain and rin...	khadi	Images/SpamMusubi.jpg	1

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**

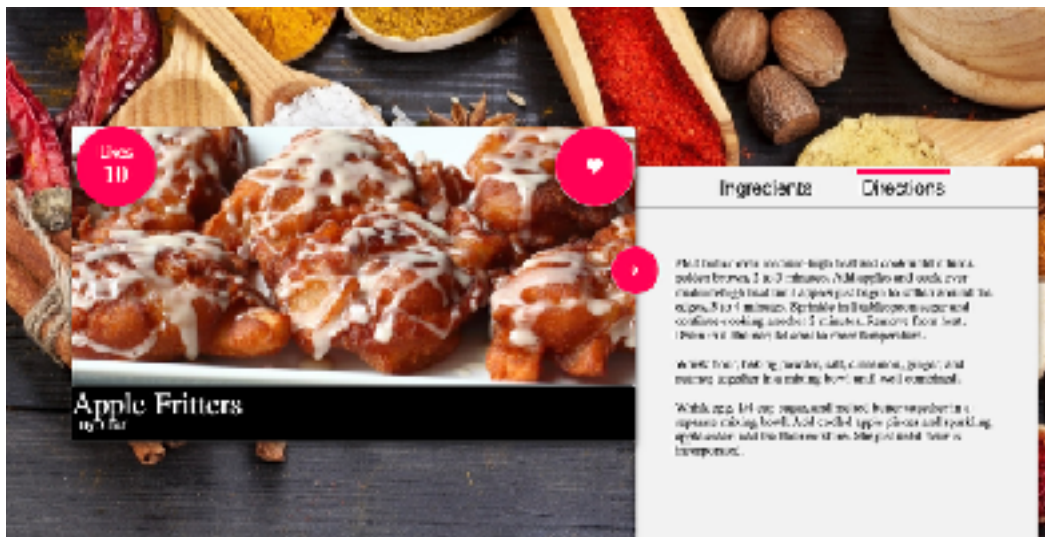
After signing up/logging in, an account page displayed the users info (including username and website stats), submitted recipes and friend list. Other users would navigate to this page to look at the users recipes and potentially follow them.

- **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.

Testing Data

Hello

Description of Final Product

Hello

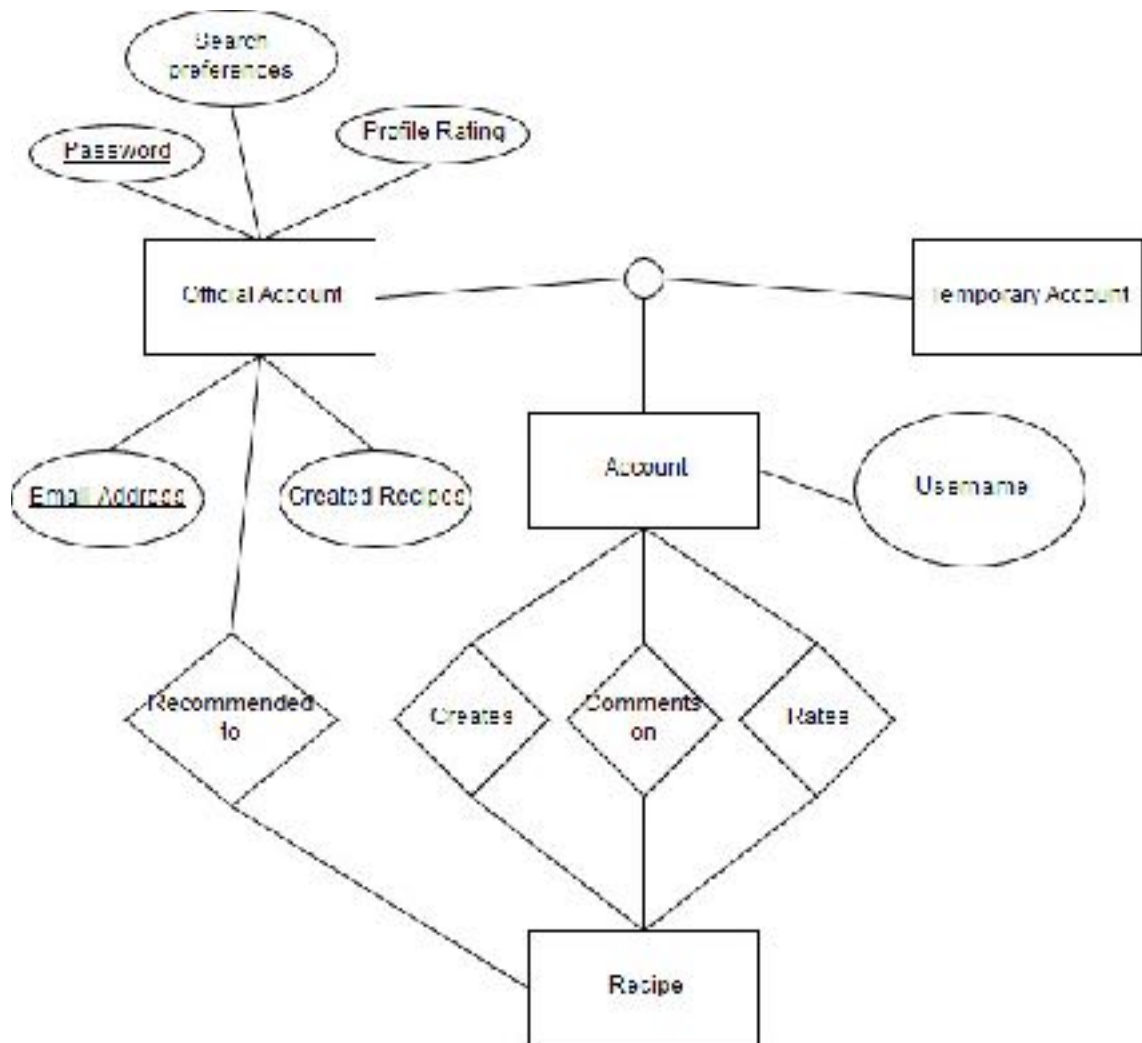
Project Time-Line

- **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

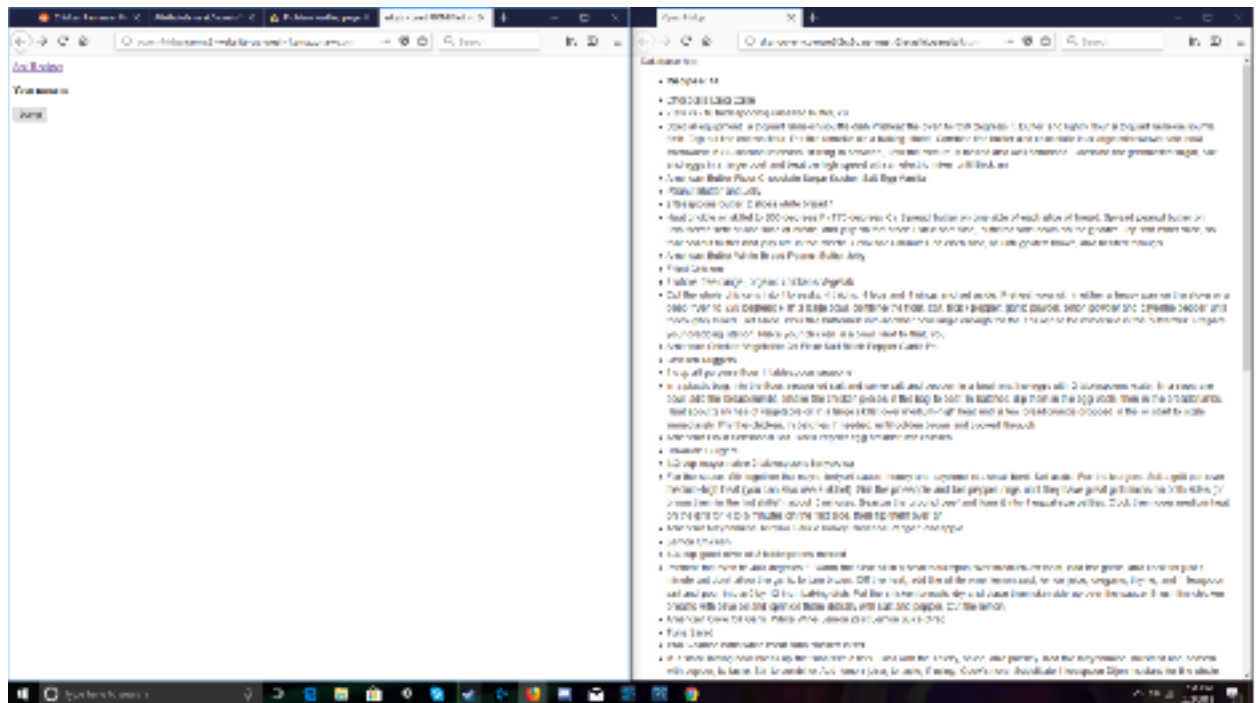
then start determining our project's entities and attributes. We then had to understand their relationships with one another to be able to create the data model. This led to the creation of the following data model that was used as a basis for our project's database.



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.

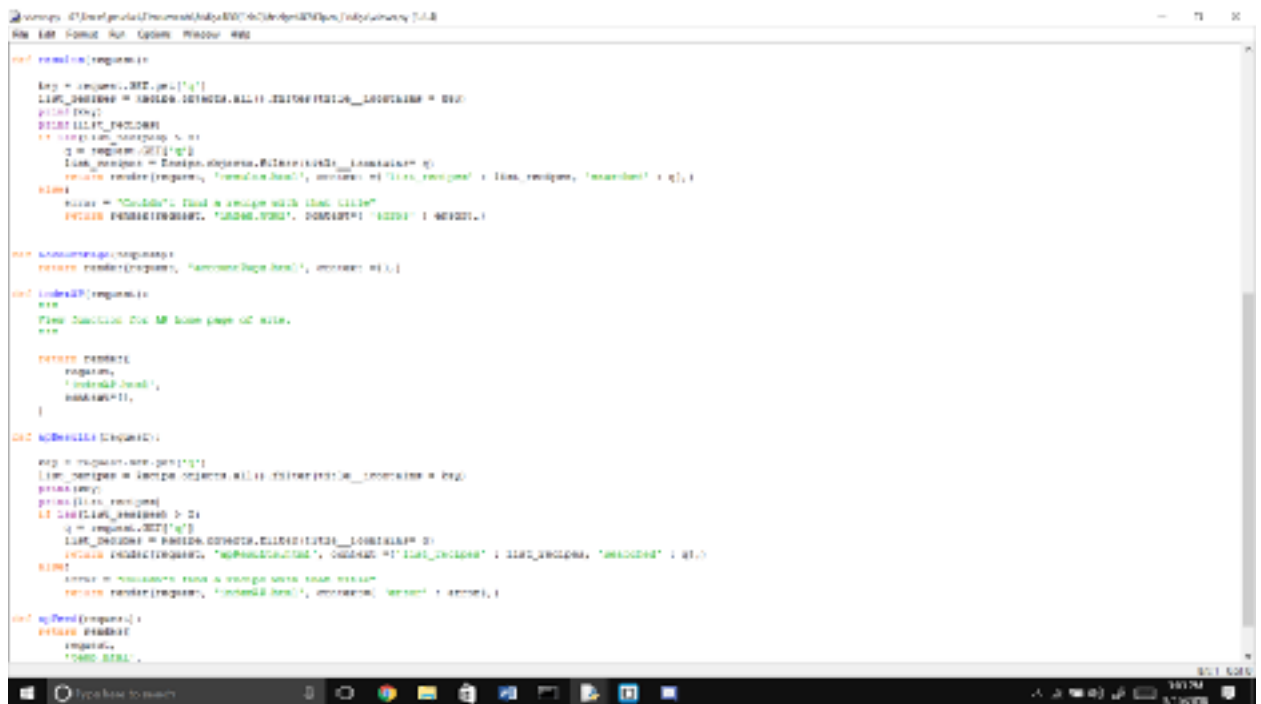


At this time the website only showed the recipes stored within our database, and the contents for the webpage of the database were actually retrieved from the database itself by using a form to submit a request to call a Django view that returned a query of all the present objects within our database. 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

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.

A screenshot of a code editor window showing a Python file named views.py. The code defines three functions: results, index, and add_recipe. The results function is the primary focus, showing a GET request handler that takes a search query 'q' and returns a list of recipes containing that string. It includes database queries and a response rendering. The index function is a simple GET handler for the home page. The add_recipe function is a GET handler for adding a new recipe. The code is written in Python and uses Django's ORM and rendering utilities.

```
def results(request):
    q = request.GET.get('q')
    list_recipes = Recipe.objects.filter(title__icontains=q)
    print(q)
    print(list_recipes)
    if request.method == 'GET':
        q = request.GET.get('q')
        list_recipes = Recipe.objects.filter(title__icontains=q)
        return render(request, 'results.html', context={'list_recipes': list_recipes, 'searched': q})
    else:
        error = "Could not find a recipe with that title"
        return render(request, 'results.html', context={'error': error})

def index(request):
    return render(request, 'homePage.html', context={'q': q})

def add_recipe(request):
    q = request.GET.get('q')
    list_recipes = Recipe.objects.filter(title__icontains=q)
    print(q)
    if list_recipes.count() > 0:
        q = request.GET.get('q')
        list_recipes = Recipe.objects.filter(title__icontains=q)
        return render(request, 'results.html', context={'list_recipes': list_recipes, 'searched': q})
    else:
        error = "Could not find a recipe with that title"
        return render(request, 'results.html', context={'error': error})

def add_recipe(request):
    return render(request, 'add_recipe.html', context={'q': q})
```

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 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

[illegible]

- 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.

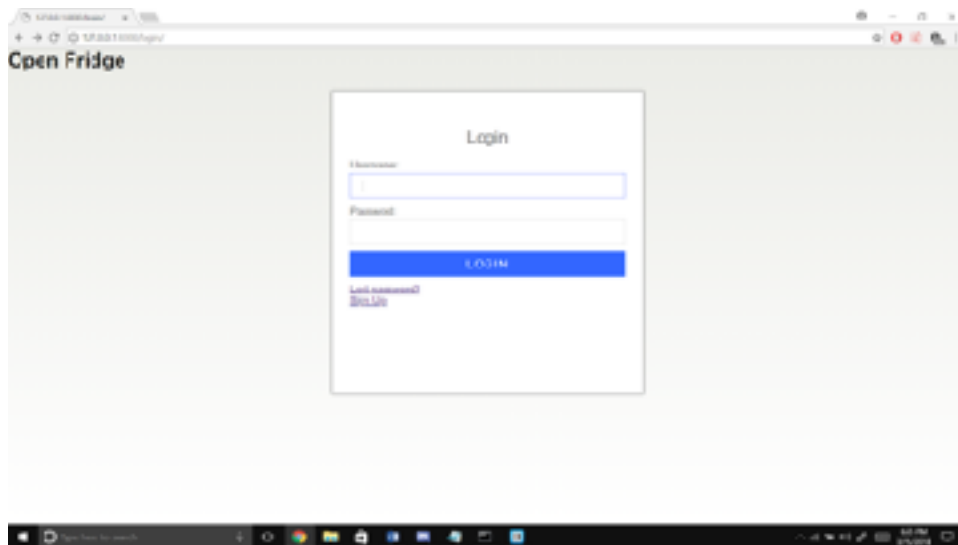
The screenshot displays a web browser window with the address bar showing 'http://127.0.0.1:8000/recipe/'. The page content includes a form titled 'Recipe Title' with a 'Submit' button. Below this, there are three main sections: 'First Step', 'Ingredients', and 'Tags'. The 'First Step' section has a text input field and a 'Submit' button. The 'Ingredients' section has a table with columns for 'Ingredient', 'Amount', and 'Unit of Measurement'. There are four rows of input fields for ingredients, each with a 'Submit' button. The 'Tags' section has a text input field and a 'Submit' button. The Windows taskbar is visible at the bottom.

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

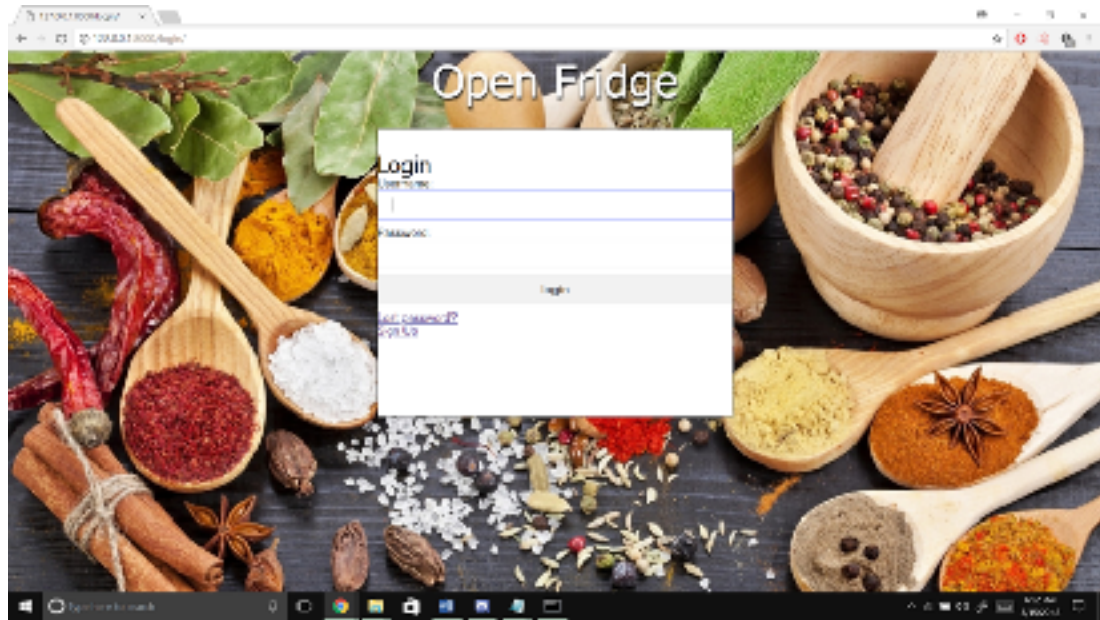
was greatly due to a lack of consideration of how it would work with Django, and lack of knowledge of better ways to implement the forms using Django.

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

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 image below contains the login box that ended up being displayed on the week's sprint.



Conclusions

Hello

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.