

Affamato: Phase Three Report

Team:

Canvas Group: **Falcon**

Phase Three Team Lead: Julia Rebello

Members:

- Cameron Clark: cameron.clark0821@utexas.edu \\ github.com/cameronclark0821
- Justin Henry: justinhenry@utexas.edu \\ github.com/justinhenry
- Alex Issa: alex.issa32@utexas.edu \\ github.com/alexissa32
- Julia Rebello: julialrebello@utexas.edu \\ github.com/JLRebello
- Samir Riad: sriad123@utexas.edu \\ github.com/sriad123
- Rooshi Patidar: rooshipatidar@utexas.edu \\ github.com/rooshimadethis

URL to Github/Gitlab repo and shared Google docs:

- Github: <https://github.com/alexissa32/Affamato>
- Google Drive: <https://drive.google.com/drive/folders/oANUp-cLx6lnZUk9PVA>
- Slack: <https://team-falcon-group.slack.com>

Website URL: <https://www.affamato.xyz/>

Phase III Report Contents:

1. Goals and Accomplishments
2. Design and Requirements
3. Scraping and Database (Description)
4. Scraping and Database (Screenshots)
5. Screens, Features and Functionality (Description)
6. Screens, Features and Functionality (Screenshots)
7. Tools, Software and Frameworks Used
8. Testing
9. Final Reflections

1. Goals and Accomplishments

Phase III - due April 22

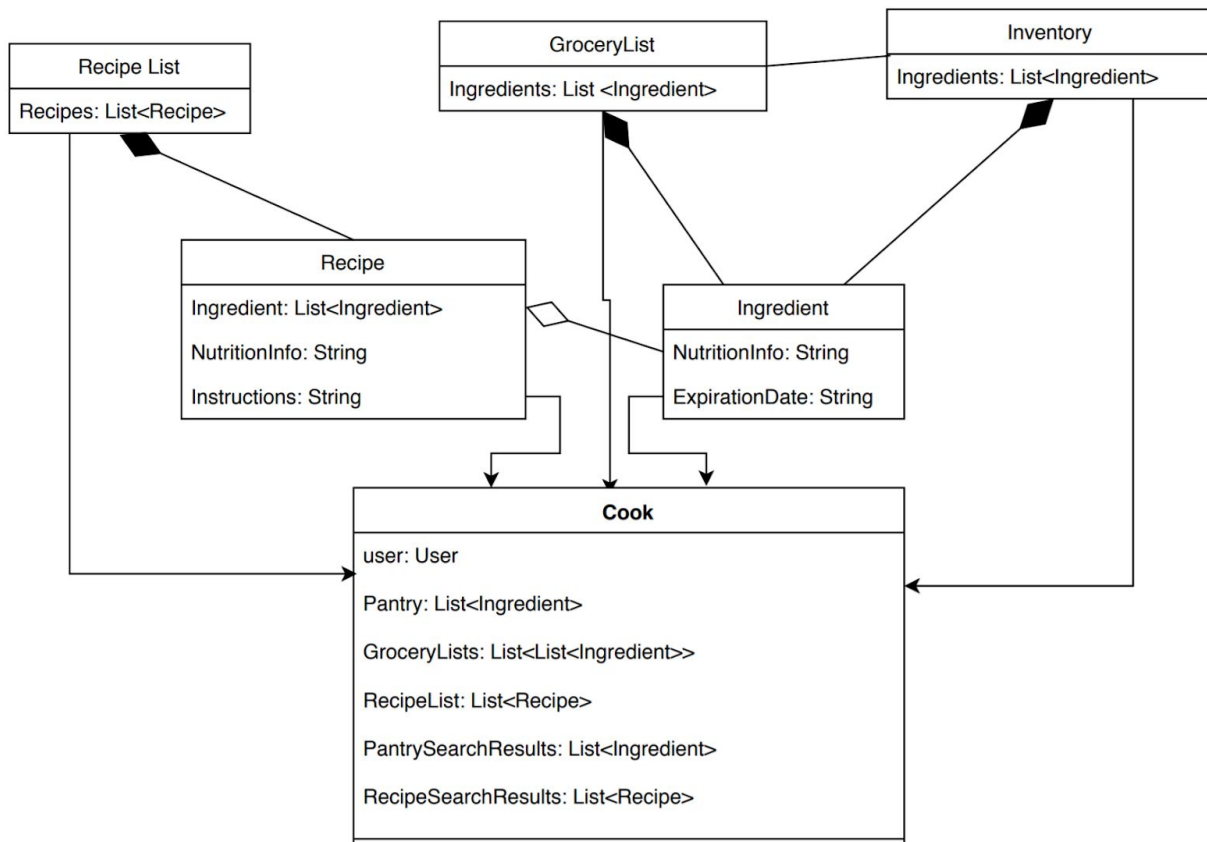
- At least 5 more user stories –put them on your issue boards (Done)
- Scraping/DB: (Done)
 - Finish data collection- all recipes, food, and relevant info is scraped and stored in the database (Done)
 - Database also holds all Cooks (Users of the site) and their relevant information (Done)
- Backend: (In progress)
 - Login page is fully functional but no account creation stored by Affamato (sign in with google -> new account creation would be like making a new google acct. Still holds their recipe list, grocery list, and inventory, etc.) (In progress)
 - Login functionality works, and user can fully use inventory, recipe list, grocery lists via front end (In progress)
 - All users have functional: grocery list, pantry, recipe list, etc. (In progress)
 - Fully functional search implementation (In progress)
- Frontend: (In progress)
 - Add front end pages for search results, discover page (In progress)
 - Finish functionality of recipes page, inventory page, and grocery list page
 - Further refinement of dynamic site with many pages hosted on GCP
 - Refined about page (In progress)
 - Refined landing page (In progress)
- Testing (Refine and expand): (Done)
 - Most Frontend/GUI tests using Selenium (Done)
 - Most Backend/Java+Servlet tests using JUnit (Done)
 - Visual inspection is also used in niche scenarios (Done)
- Add to and refine the technical report, updating as we go (Done)
 - Fix Phase 2 Errors in Class diagram+add sequence diagram (Done)

Comments:

- To see our user stories on issue boards, check our Github. To see them in the report, check the “Design and Requirements” section below.
- To see the other major bullet points and their progress, check the relevant sections titles below.
- Things we realized we didn’t need (That were initially listed in our Proposal):
 - Java Spring-Not necessary for a small-scale application
 - Elasticsearch-We built our own search functionality
 - SQL DB and querying-We used NoSQL/datastore and JSON to hold all information
 - Reactjs-Built our front end with JSPs/DHTML and bootstrap
 - Mocha-There’s no JS code to test
 - Postman-No APIs able to test (Most aren’t exposed/require a user to be logged in through Google)

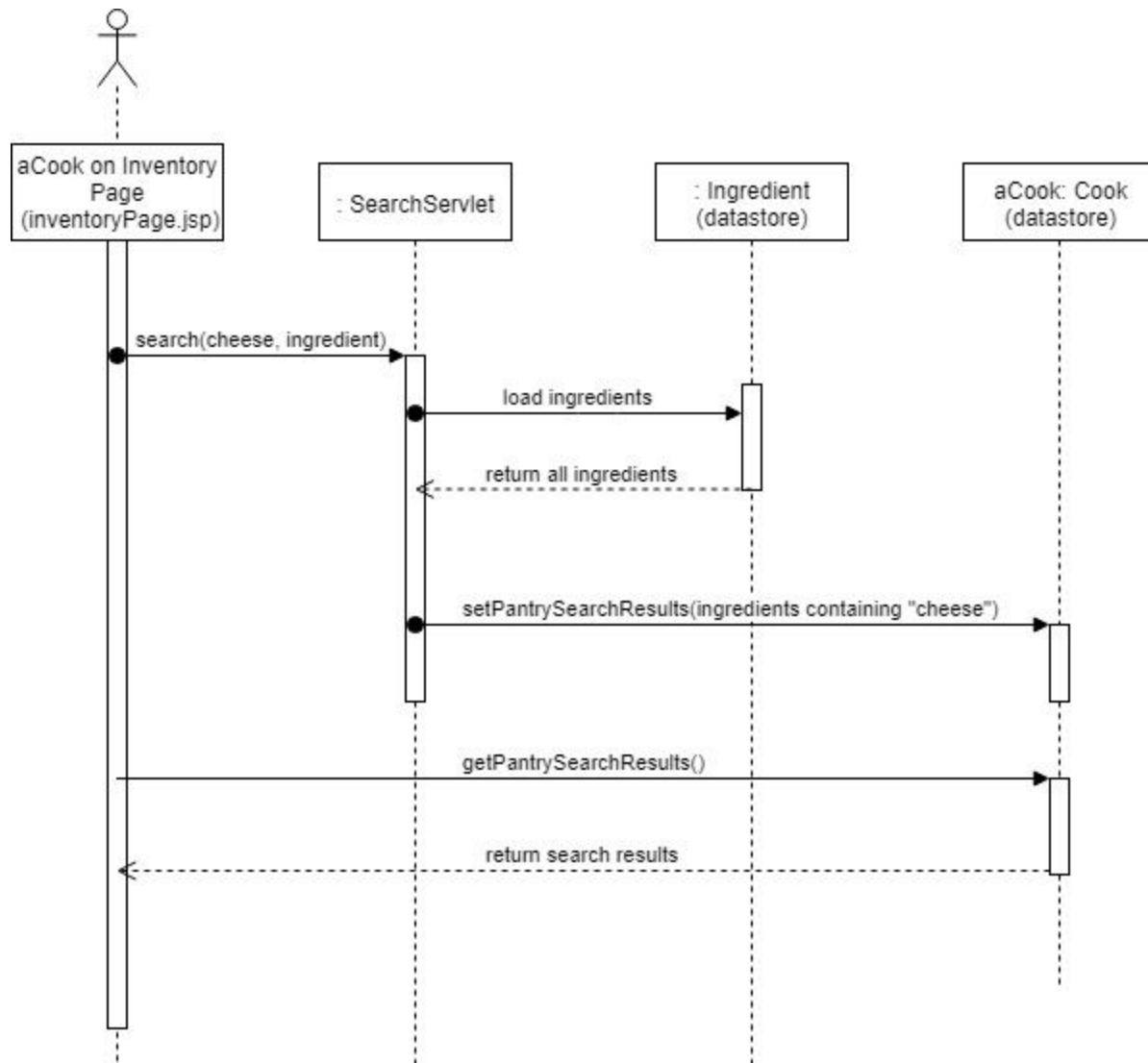
2. Design and Requirements

Class Diagram:



Note: The relationship between the other classes and the Cook class is a weak association because the Cook fields are made from some parts of ingredients and recipes but is not strictly made of those objects.

Sequence Diagram:



Phase I User Stories:

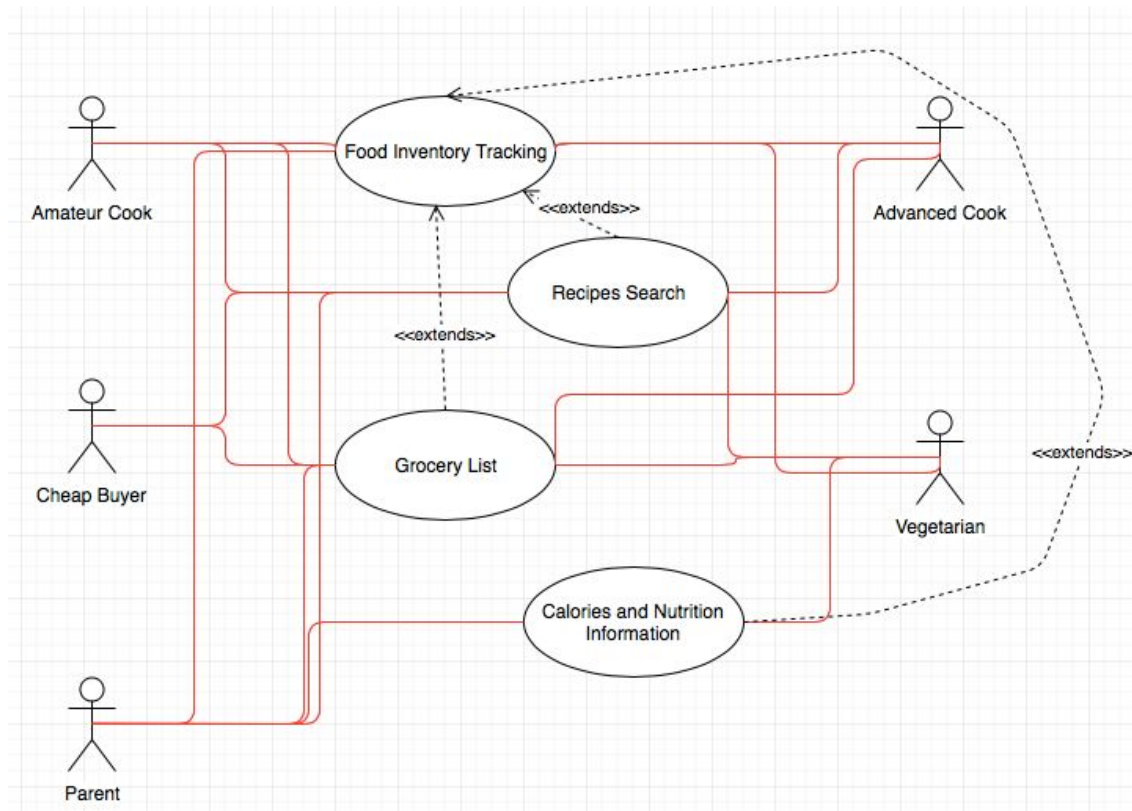
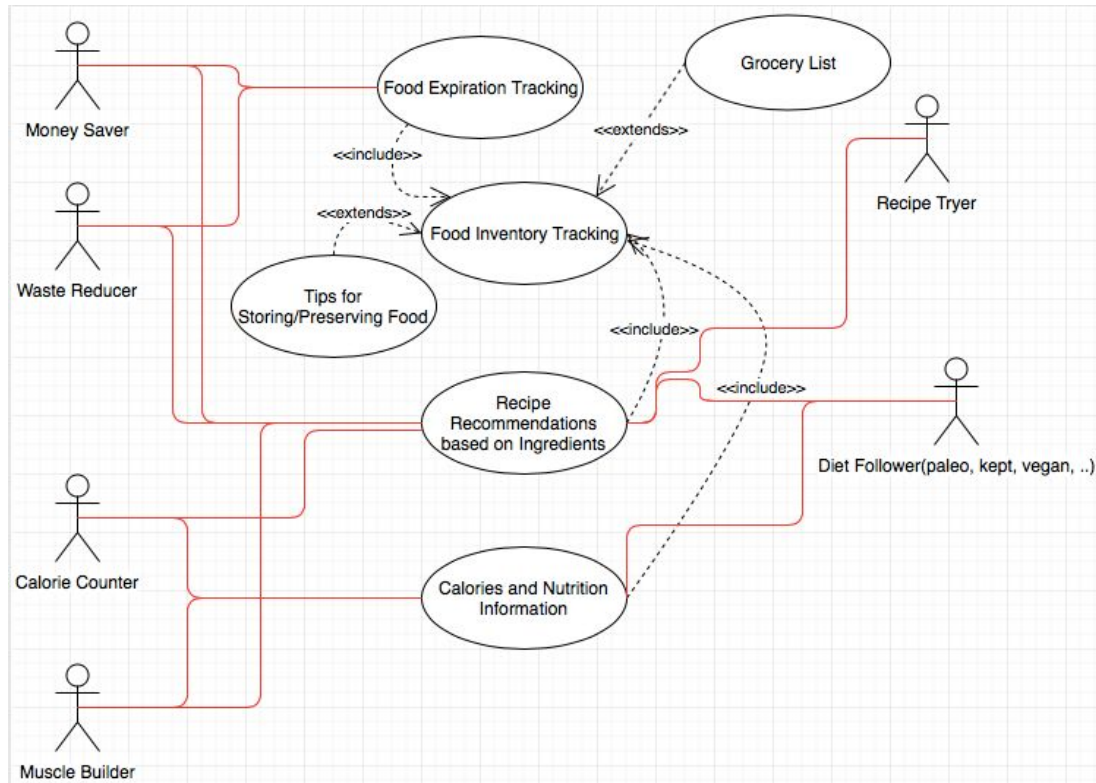
1. Save user's money
 - a. As a thrifty consumer, I would like to cut down on grocery shopping by efficiently using my groceries, so I am not re-purchasing items unnecessarily.
2. Reduce food waste
 - a. As a green consumer, I would like to reduce the amount of food I waste by tracking when items expire so that I may use all of the perishables that I purchase.
3. Calorie counter: lose weight, focusing on calories
 - a. As someone who would like to lead a healthier lifestyle, I would like to have quick access to the nutritional information of the food I buy.
4. Tries new recipes (For example, wanting cuisine type X)

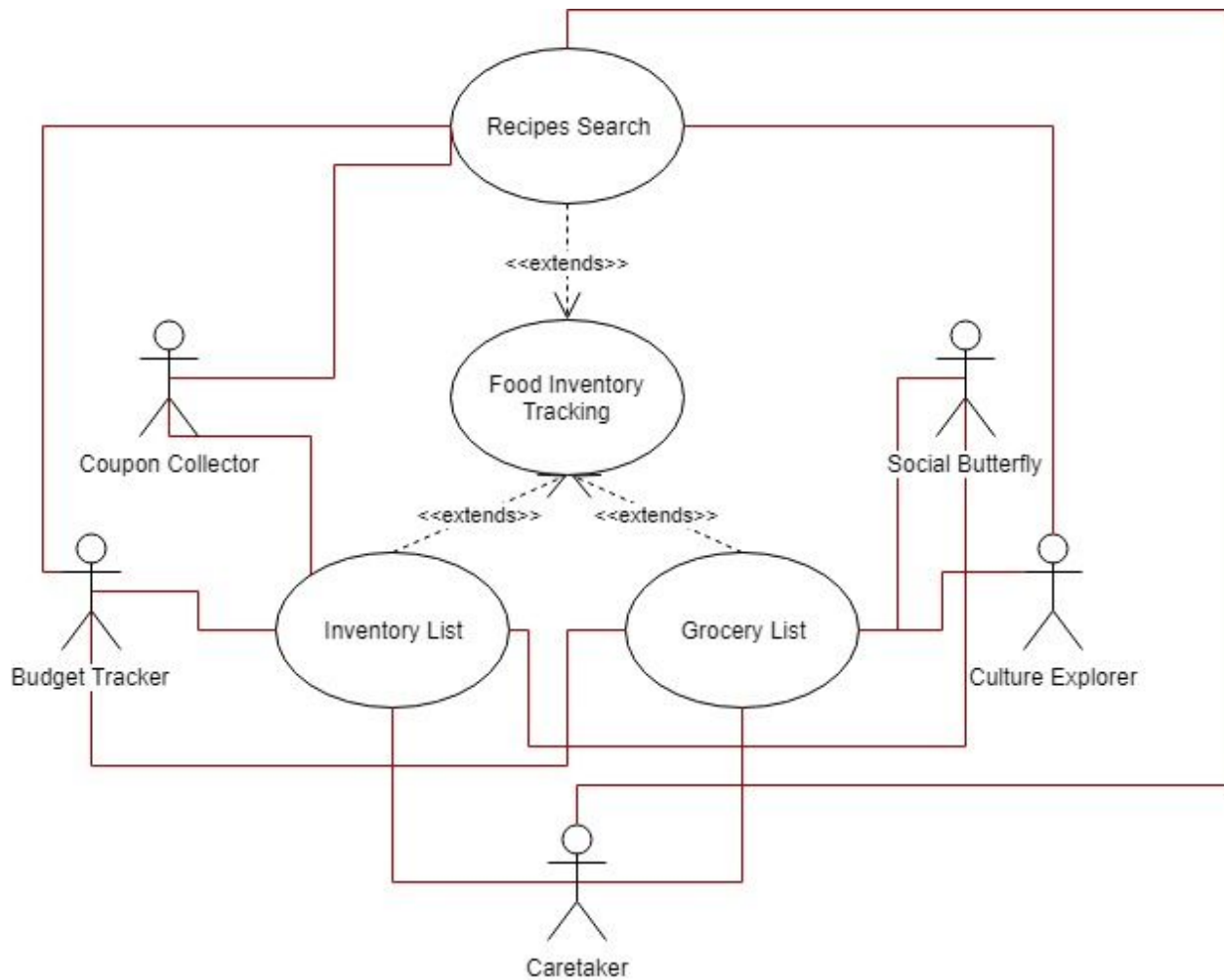
- a. As an adventurous foodie, I would like to try new recipes.
- 5. Bulking: gain weight, focusing on maximizing volume of protein and healthy fats, getting more with less so one can fit it in the fridge/only need to get groceries once a week.
 - a. As a bodybuilder, I would like to increase my muscle mass by consuming the right foods at a great enough volume.
- 6. Trying to follow a specific diet for health-related reasons
 - a. As a someone who has dietary and health restrictions, I would like to have easy access to what foods I can eat, and what recipes I can make with them.
- 7. User who wants to obtain cooking skills
 - a. As a someone who has just left home, I would like to become more independent by learning some easy recipes I can make for myself.
- 8. User who wants to purchase cheap items and find recipes for them
 - a. As a someone who likes to purchase on-sale items and use coupons to shop cheaply, I would like to find recipes that will accomodate my purchases.
- 9. Vegetarian User
 - a. As a someone who has chosen to be a vegetarian for personal reasons, I would like to find recipes I can make with ingredients that don't include any meats, poultry, or fish.
- 10. Advanced cooking-level/chef user who wants to improve skills
 - a. As a chef or someone who cooks very often and already have skills in the kitchen, i'd like to improve and advance my skills even further by exploring recipes that include unfamiliar ingredients.
- 11. Parent User
 - a. As a parent or guardian of young children, I would like to find recipes that use kid-friendly ingredients so that I can cook healthy meals my children will want to eat.
- 12. Coupon Collecting User
 - a. As someone who saves money by using coupons, I'd like to purchase my items based on what coupons are available, and then discover new recipes to use these items.
- 13. Budget Tracking User
 - a. As someone who wants to track my food budget, I'd like to track how much food I buy, and how much of it I use before it perishes, and be able to maximize my budget, and learn the best food purchasing habits to keep my budget lean.
- 14. Caretaker User
 - a. As a caretaker, I'd like to be able to easily buy food items that fit within various dietary restrictions, and learn various recipes I can make with these items to make healthy and varying dishes.
- 15. Social Butterfly User
 - a. As a user whose social life is very important to them, I'd like to be able to manage several different grocery lists so that if I plan to throw an event, I can quickly see what I have at home and what I need to buy, as opposed to a grocery list that I have for normal daily runs to the grocery store.

16. Culture Exploring User

- a. As a user who loves to travel, I would like to explore different cuisines by learning about new ingredients and recipes so that I may familiarize myself ahead of time before an international trip.

Phase III User Diagrams:





3. Scraping and Database (Description)

Recipes and Ingredients exist in our datastore to serve as a database of ingredients and recipes to be searched. Both include fields with relevant information to use. For the Recipes, these include: strings for the recipe title and instructions, integers for the minutes to prepare and cook the recipe, and booleans for if the recipe is vegetarian, vegan, ketogenic, gluten free, and dairy free. For the Ingredients these include: a string for the ingredient name, along for the ingredient ID (provided by spoonacular), a float for the amount of the ingredient, and string for the units of that amount. We also included a string that includes information about the nutrients in the ingredient given a quantity of the ingredient. These fields are summarized in the following screenshots.

In phase three we implemented the Cook object in the datastore. Each cook represents one user who has logged in via Google. This object stores all of the relevant information for the user. The

datastore does not allow storage of arrays or JSON objects, so the fields were implemented as string representations of JSON arrays. The fields in the cook class include all of the relevant information for the user. This includes: **PantryList**, a list of the ingredients the user has in their inventory; **GroceryList**, a list of the ingredients the user intends to buy at the store; and **RecipeList**, a list of Recipes the user has saved. We also have fields for search results for searches the user just made: **RecipeSearchResults**, **GrocerySearchResults**, and **RecipeSearchResults**. All of these fields are also summarized in the following screenshots. We update these fields whenever necessary, and we clear the search results data whenever a person leaves the page or searches for something new.

All of our code can be seen on the [Github](#).

4. Scraping and Database (Screenshots)

-Extensive List of Ingredients in GCP Datastore:

[FILTER ENTITIES](#)

name/ID ↑	amount	ingredient	jsonString	nutrientString	spoonid	unit	unitShort
=4504321248985088	10	curry seasoning	{\"id\": 2015, \"original\": null, \"originalName\": nul...	[{\"amount\":921.36,\"unit\":\"cal\",\"percentOfDaily...	2015	ounce	oz
=4504436542013440	10	frozen spinach	{\"id\": 11463, \"original\": null, \"originalName\": n...	[{\"amount\":82.21,\"unit\":\"cal\",\"percentOfDailyN...	11463	ounce	oz
=4504956434382848	10	vanilla cake mix	{\"id\": 18137, \"original\": null, \"originalName\": n...	[{\"amount\":1054.6,\"unit\":\"cal\",\"percentOfDaily...	18137	ounce	oz
=4505283858530304	10	pistachio	{\"id\": 12151, \"original\": null, \"originalName\": n...	[{\"amount\":1593.24,\"unit\":\"cal\",\"percentOfDaily...	12151	ounce	oz
=4507511872815104	10	shredded unsweetened coconut	{\"id\": 10012108, \"original\": null, \"originalName...	[{\"amount\":1871.07,\"unit\":\"cal\",\"percentOfDail...	10012108	ounce	oz
=4507550057758720	10	blue cheese	{\"id\": 1004, \"original\": null, \"originalName\": nul...	[{\"amount\":1000.74,\"unit\":\"cal\",\"percentOfDail...	1004	ounce	oz
=4508533940813824	10	oregano leaves	{\"id\": 2027, \"original\": null, \"originalName\": nul...	[{\"amount\":751.26,\"unit\":\"cal\",\"percentOfDaily...	2027	ounce	oz
=4509303947919360	10	green chili	{\"id\": 31015, \"original\": null, \"originalName\": n...	[{\"amount\":76.54,\"unit\":\"cal\",\"percentOfDailyN...	31015	ounce	oz
=4510643977715712	10	dried fruit	{\"id\": 1009094, \"original\": null, \"originalName\":...	[{\"amount\":705.9,\"unit\":\"cal\",\"percentOfDailyN...	1009094	ounce	oz
=4510780410036224	10	Semi-Sweet Chocolate Baking Chips	{\"id\": 10019903, \"original\": null, \"originalName...	[{\"amount\":1641.44,\"unit\":\"cal\",\"percentOfDail...	10019903	ounce	oz
=4511144005861376	10	toffee	{\"id\": 10019383, \"original\": null, \"originalName...	[{\"amount\":1587.57,\"unit\":\"cal\",\"percentOfDail...	10019383	ounce	oz
=4512270562361344	10	oyster crackers	{\"id\": 10018228, \"original\": null, \"originalName...	[{\"amount\":1193.51,\"unit\":\"cal\",\"percentOfDail...	10018228	ounce	oz
=4514024788066304	10	condensed milk	{\"id\": 1095, \"original\": null, \"originalName\": nul...	[{\"amount\":910.02,\"unit\":\"cal\",\"percentOfDaily...	1095	ounce	oz
=4517010662752256	10	red jalapeno chile	{\"id\": 10111819, \"original\": null, \"originalName...	[{\"amount\":113.4,\"unit\":\"cal\",\"percentOfDailyN...	10111819	ounce	oz
=4518560273530880	10	poppy seed	{\"id\": 2033, \"original\": null, \"originalName\": nul...	[{\"amount\":1488.35,\"unit\":\"cal\",\"percentOfDail...	2033	ounce	oz
=4519035001634816	10	green cabbage	{\"id\": 11109, \"original\": null, \"originalName\": n...	[{\"amount\":70.87,\"unit\":\"cal\",\"percentOfDailyN...	11109	ounce	oz
=4521986516582400	10	fig jelly	{\"id\": 11519297, \"original\": null, \"originalName...	[{\"amount\":788.12,\"unit\":\"cal\",\"percentOfDaily...	11519297	ounce	oz

-Extensive List of Recipes in GCP Datastore

Kind
Recipe

FILTER ENTITIES

cookMinutes	dairyFree	glutenFree	instructions	jsonString	ketogenic	prepMinutes	title	vegan	vegetarian
0	false	false	1. Preheat oven to 350F. Spray 10-inch Bundt...	({"vegetarian": true, "vegan": false, "glutenFree": ...	false	0	Commmeal Bundt Cake with Pecan Pie Swirl	false	true
0	false	false	Add half of the cookies into a food processor...	({"vegetarian": false, "vegan": false, "glutenFre...	false	60	Pumpkin Patch Oreo Pops	false	false
15	true	true	Watch how to make this recipe. For the Chick...	({"vegetarian": false, "vegan": false, "glutenFre...	false	10	Chicken with a Lemon Herb Sauce	false	false
20	false	false	Preheat a grill to medium-high heat. Toss the...	({"vegetarian": false, "vegan": false, "glutenFre...	false	20	Low Country Boil Packets	false	false
0	false	false	Preheat oven to 350 degrees. Line baking sh...	({"vegetarian": false, "vegan": false, "glutenFre...	false	0	Brown Butter Dark Chocolate Chunk Cookies	false	false
140	false	false	Watch how to make this recipe. Divide and s...	({"vegetarian": false, "vegan": false, "glutenFre...	false	15	Sunny's Easy Freeze and Bake 24/7 Mini Pizz...	false	false
0	false	false	Preheat oven to 375°F. Arrange pita wedges ...	({"vegetarian": true, "vegan": false, "glutenFree"...	false	0	Green Pea Dip with Toasted Pita Chips	false	true
205	true	true	Watch how to make this recipe. Combine the...	({"vegetarian": false, "vegan": false, "glutenFre...	false	10	Herb-Marinated Pork Tenderloins	false	false
360	true	true	Spray slow cooker with non-stick cooking spr...	({"vegetarian": false, "vegan": false, "glutenFre...	false	10	Slow Cooker Maple Dijon pork chops	false	false
0	false	false	Pound CakeCaramel Drizzle	({"vegetarian": true, "vegan": false, "glutenFree"...	false	0	Five-Flavor Pound Cake	false	true
0	true	true	Freeze glasses until frosted; remove from fre...	({"vegetarian": true, "vegan": true, "glutenFree"...	false	5	SOBE Blackberry Mojito Cocktail	true	true
0	false	true	1. To make Roasted Peppers and Onions: Pr...	({"vegetarian": false, "vegan": false, "glutenFre...	false	0	Black Bean Tacos with Roasted Peppers and...	false	false
20	false	true	Line a rimmed baking sheet with parchment ...	({"vegetarian": true, "vegan": false, "glutenFree"...	false	15	Almond Toffee	false	true
20	false	false	Preheat oven to 400 degrees F. To make the ...	({"vegetarian": false, "vegan": false, "glutenFre...	false	30	Stuffed Mushrooms with Sausage	false	false
0	false	false	To prepare the cookie dough, stir together th...	({"vegetarian": false, "vegan": false, "glutenFre...	false	0	Chocolate Chip Cookie Dough Cheesecake	false	false
0	false	false	Cook the spaghetti according to the package...	({"vegetarian": false, "vegan": false, "glutenFre...	false	0	Easy Italian Spaghetti Pasta Salad	false	false
0	true	true	Place first three ingredients in a medium bo...	({"vegetarian": false, "vegan": false, "glutenFre...	false	0	Fresh Southwest Salsa	false	false
12	false	true	Preheat oven to 350in a mixing bowl combin...	({"vegetarian": false, "vegan": false, "glutenFre...	false	7	Mini Deep Dish Paleo Pizzas	false	false
2	false	false	Heat the oven to 300F. In a microwave safe b...	({"vegetarian": false, "vegan": false, "glutenFre...	false	2	Candy Cane Smores	false	false
0	false	true	In a trifle bowl or large glass bowl, layer the L...	({"vegetarian": false, "vegan": false, "glutenFre...	false	15	Seven-Layer Salad	false	false
15	false	true	1. Combine first 6 ingredients in a large zip-t...	({"vegetarian": false, "vegan": false, "glutenFre...	false	15	Buttermilk- Brined Pork Chops	false	false
15	false	true	Heat the oil in a medium sized skillet over m...	({"vegetarian": true, "vegan": false, "glutenFree"...	false	5	Creamy Corn and Zucchini	false	true
15	false	true	For the bananas Foster: Add the butter and t...	({"vegetarian": false, "vegan": false, "glutenFre...	false	5	Bananas Foster Milkshake	false	false
0	false	false	Spray a 9 x 13" baking dish with cooking spra...	({"vegetarian": false, "vegan": false, "glutenFre...	false	10	Chocolate Peanut Butter Rice Krispie Treats	false	false

-Cook objects in the Datastore for users

QUERY BY KIND

QUERY BY GQL

Kind
Cook

FILTER ENTITIES

	Name/ID ↑	Parent	GrocerySearchResults	PantrySearchResults	RecipeSearchResults	user
<input type="checkbox"/>	id=5747712368574464	Key(Cook, 'affamatoapp@gmail.com')	({"dairyFree": false, "instructions": "Cut the stra...	({"unit": "ounce", "amount": 10, "unitShort": "oz", "n...	({"dairyFree": false, "instructions": "Beat softene...	affamatoapp
<input type="checkbox"/>	id=5659795562430464	Key(Cook, 'affamatotesting@gmail.com')		({"unit": "ounce", "amount": 10, "unitShort": "oz", "n...	({"dairyFree": false, "instructions": "Beat softene...	affamatotest
<input type="checkbox"/>	id=5681205034876928	Key(Cook, 'alexissa122@gmail.com')			({"dairyFree": false, "instructions": "In a large du...	alexissa122@gmail.com
<input type="checkbox"/>	id=5639686995312640	Key(Cook, 'julialrebello@utexas.edu')			({"dairyFree": false, "instructions": "Make the vin...	julialrebello@gmail.com
<input type="checkbox"/>	id=5747862927310848	Key(Cook, 'justinhenryy@gmail.com')		({"unit": "ounce", "amount": 10, "unitShort": "oz", "n...		justinhenryy@gmail.com
<input type="checkbox"/>	id=5671274802053120	Key(Cook, 'rooshipatidar@gmail.com')	({"dairyFree": false, "instructions": "Heat a nons...		({"dairyFree": false, "instructions": "Combine pic...	rooshipatidar@gmail.com

5. Screens, Features and Functionality (Description)

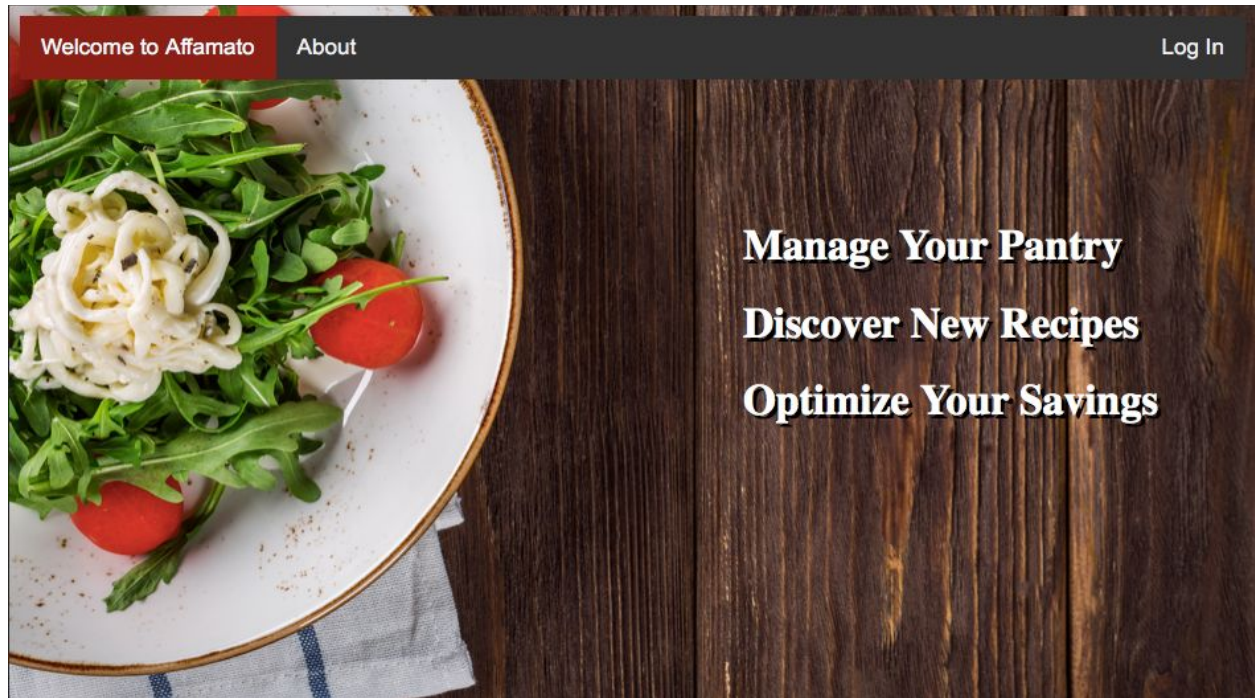
We have implemented a dynamic website (In DHTML). When the website is opened the welcome and about pages will be visible. When you log in with your gmail account, the dashboard page becomes available, and selections from the dashboard page which will display grocery list, food inventory, recipes list. Logging out will also become an option. The grocery list, inventory, and recipe list pages also have search bars. The pages are somewhat functional; accounts are now implemented and can search for recipes. However, users are unable to

save recipes and ingredients, in addition to searching for ingredients. Primarily we failed to implement this because we tried to build out the backend and the frontend separately, and because it took too long to make the backend functionality before we could attempt to connect the two. We spent a significant amount of time attempting to solve problems with the backend. These problems mainly involved updating the cook class when searching. We attempted to use JSONArrays in the datastore for a while before realizing that it isn't possible with Objectify - it wouldn't update. Because of this, we also spent a significant amount of switching the cook class from using strings to using JSONArrays and back to strings. This took a significant amount of time as we had to update all of the methods in addition to the fields. Please go to the link to see our website.

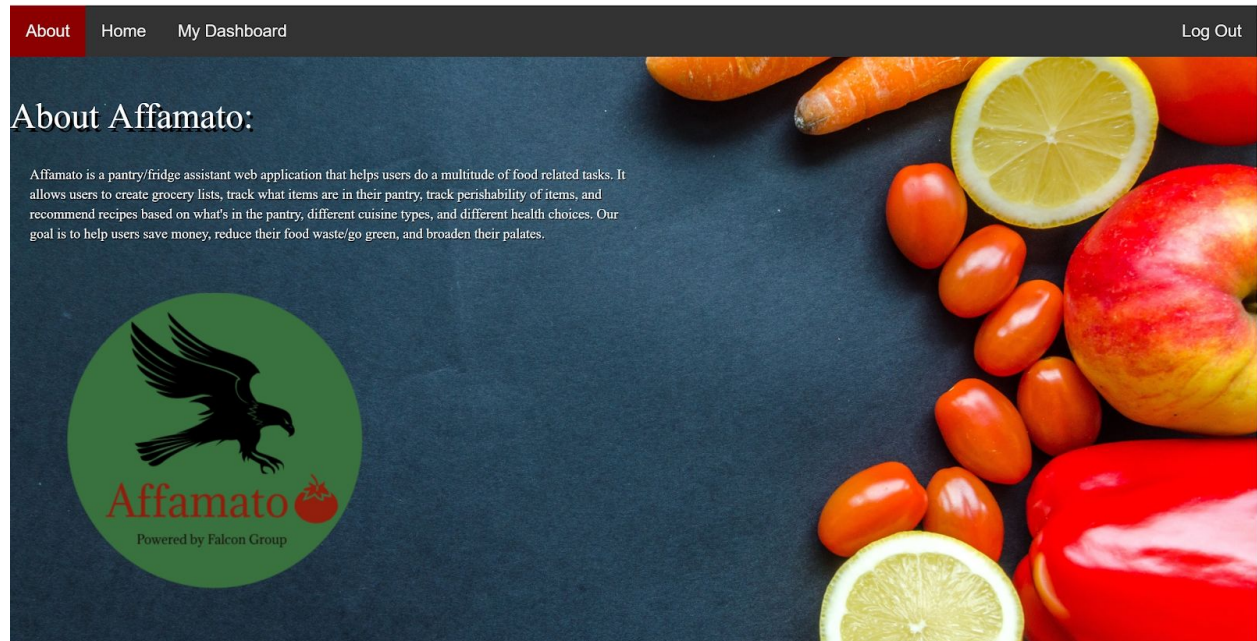
All of our code can be seen on the Github.

6. Screens, Features and Functionality (Screenshots)

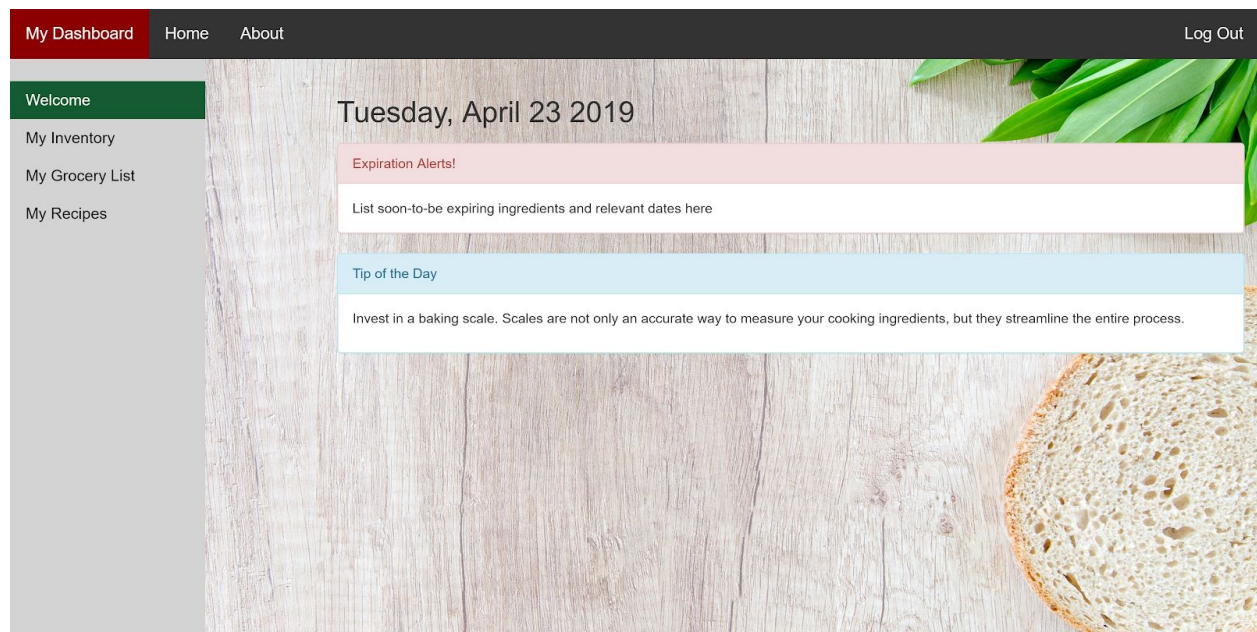
-Affamato Welcome Page:



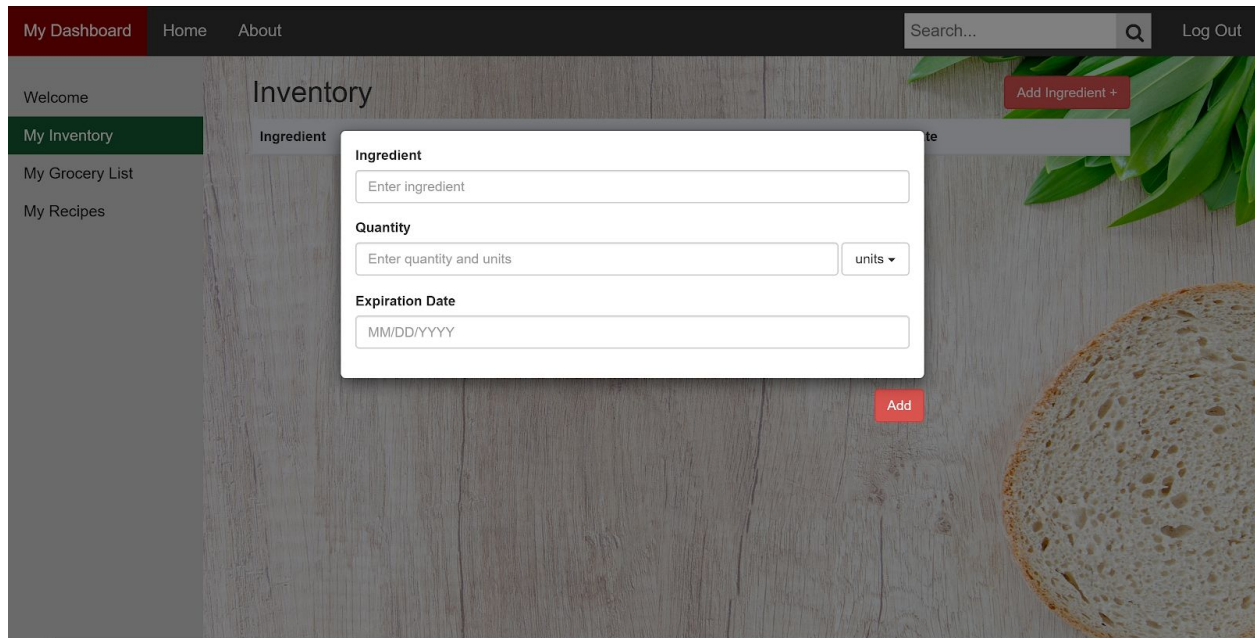
-Affamato About Page



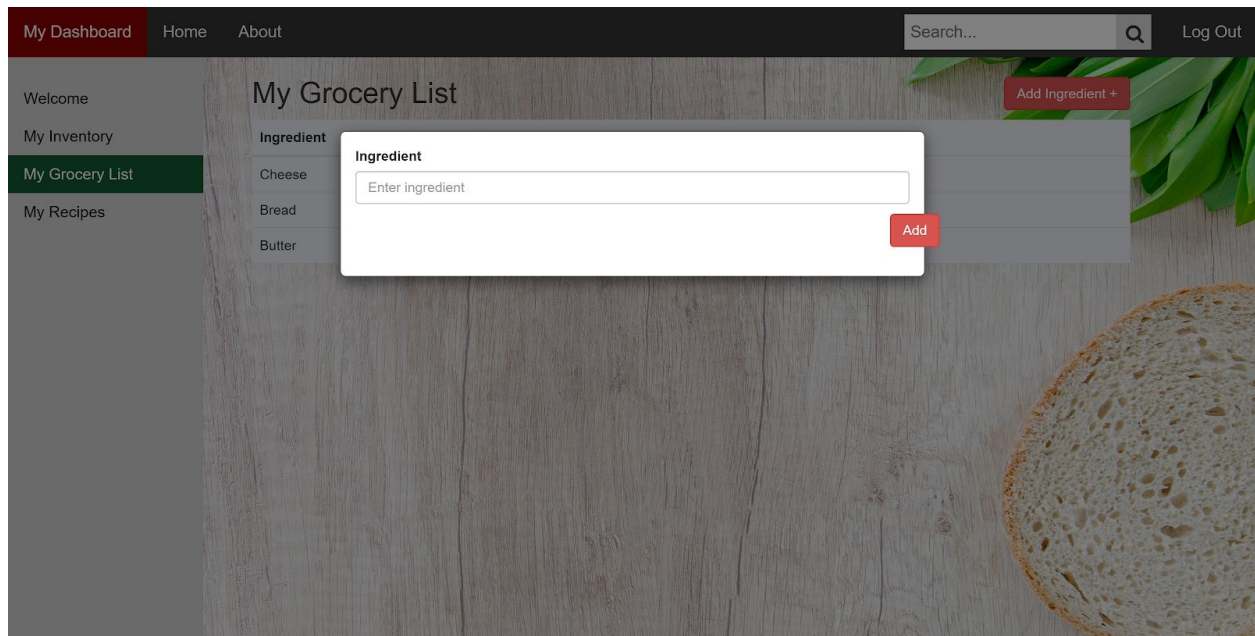
-Affamato Welcome Page



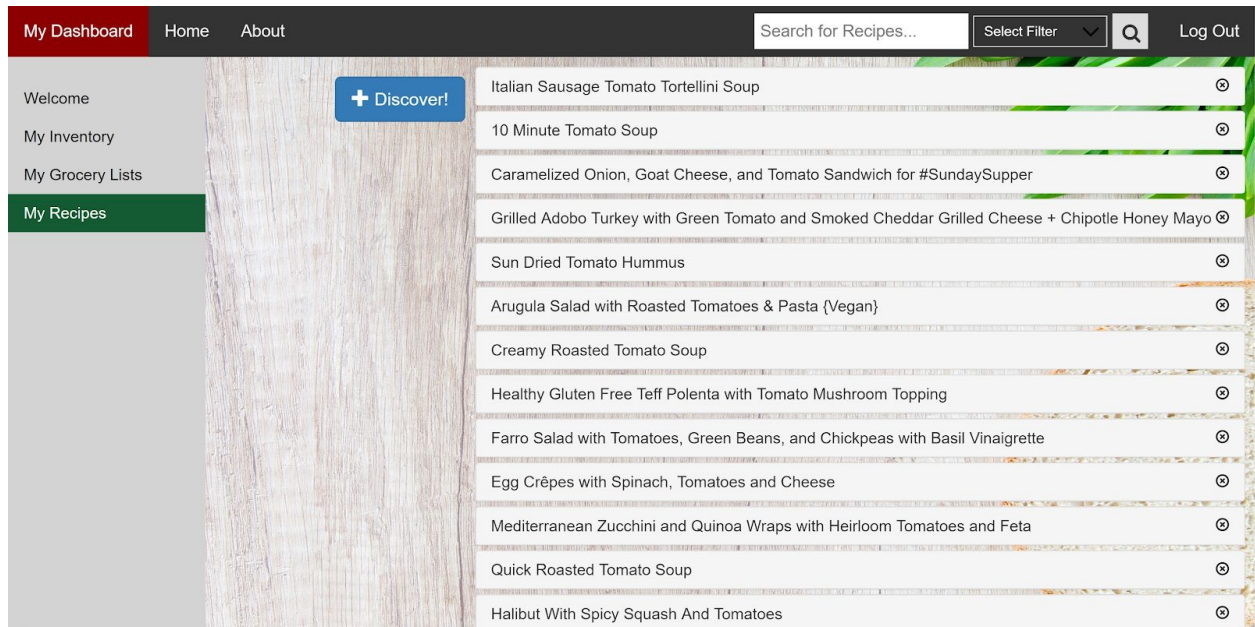
-My Inventory



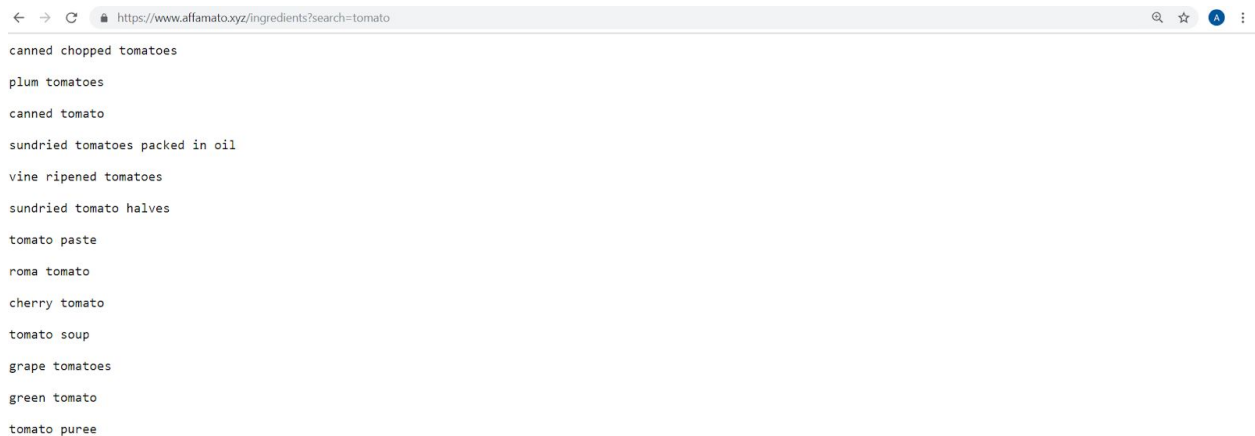
-My Grocery List



-My Recipes



-search results from recipes page



-search results if you use the search bar from the inventory/grocery list pages. Unable to implement in the front end so it only prints plain text

7. Tools, Software, Frameworks Used

In this phase we continued developing with DHTML in the frontend and Java in the backend, which used the Google objectify library and Datastore to store data. In the frontend we decided to not use ReactJS as the frontend was already built to a decent degree with JSPs. We implemented a new objectify object, the Cook class. This was also connected to the frontend using additional Java Servlets. We used a cronjob to pull all of the JSONs we scraped to our database, and organized them using the Ingredient and Recipe Java files, which essentially defined our object structures in addition to being used for initial searches. In terms of testing we

used JUnit for unit testing of the cook class, which is the only class that is interacted with in depth. For front end/UI testing we used Selenium. We also used the logging features of the App Engine to find more errors.

We refrained from using many tools and frameworks that we initially thought would be useful. We didn't use Java Spring as it is not necessary for a relatively small scale application. We used our own simple search function instead of using Elasticsearch. We used the Google NoSQL datastore based from scraped JSONs instead of using an SQL database and querying. We didn't use Mocha for testing as there was no pure JavaScript code to test. Lastly, we didn't use Postman for API testing, as all of our data is stored in our datastore and our application doesn't make any API calls.

8. Testing

In phase 3 our unit tests were implemented with JUnit. As the Ingredient and Recipe classes are only used when making the initial search and getting a string with the relevant information, all of our unit tests were focused on the Cook class. Additionally, the Ingredient and Recipe classes do not change; they only serve as a framework for the database to draw from. Consequently, the Cook class was the most essential class to test, as the fields in the Cook class are changing often as the user updates their lists and search results.

For the RecipeList, tests were implemented focusing on the functionality of adding recipes to the list and removing recipes from the list. Very similar tests were implemented for the Pantry, testing the functionality of adding and removing ingredients. Similar tests for GroceryList were also implemented testing the functionality of adding and removing ingredients. These tests were created to ensure that the lists are functional in multiple scenarios as these lists change often but are seldom cleared.

Tests were also implemented for GrocerySearchResults, RecipeSearchResults, and PantrySearchResults. Tests for each set of search results test adding to the list of search results and, getting the results back from the cook class successfully. These tests were more simple, as these fields are essentially used for passing information from backend to frontend.

For the frontend, we used Selenium to build integration and interaction tests that helped verify our front end was working as needed. We were able to check the ability to log in or out from different pages and verify that it worked correctly. We also checked that button redirects to different pages worked correctly, and that data was updating when we submitted information into the different pages.

Lastly, we used the Logging functionality of the App Engine to find further issues with how our code was running. This, combined with visual inspection, allowed us to catch bigger issues and then quickly pinpoint why there was an error with our code.

9. Final Reflections

There was a lot that was hard about this project. There were two main issues that we think caused a lot of blockers for us. The first was a general lack of familiarity with a majority of the things we had to use. No one on our team knew DHTML or Javascript, no one had experience with developing websites, there was minimal experience with the Cloud (And no experience with GCP specifically), etc. While we covered some of this material in class, the lack of depth or practice time that we got before being asked to essentially build a startup company wasn't enough to really feel comfortable with what we were doing and to do it efficiently. We ran into lots of blockers where we didn't know what to do, and while the answers were often trivial once we found them, our project would be completely derailed until we could find the answer. We don't think these issues are particularly the class's fault, we think there's just a lot of material to cover in the class, and most of the things we needed to do the project weren't what we went into depth on in class. There was a lot of self learning.

The second big issue was effectively a mistake on how we went about designing our project. We decided to do the backend and frontend completely independent of each other, and then connect everything together at the final stage. This was a huge error, as the backend team built a lot of stuff that wasn't quite what the frontend needed/anticipated having to pull the information and update it. Then from the frontend, when they didn't know how to interact with something from the backend, it was often because they didn't know the in and outs of the backend code. This lack of understanding from each side led to lots of delays in our ability to finish the project.

The main functionality we didn't accomplish is that users are unable to save recipes and ingredients, in addition to searching for ingredients displaying well.

If we could do something differently, the main thing we would change would've been to implement small pieces of the code completely on both the frontend and backend, and go from there. This would've let us make mistakes and learn from our issues at the beginning, and then be better off at the end.