Final Report

RecipeSearch

Group 12

Jacob Wahib z5164984

Ryan Eves z5164560

Estella Arabi z5165786

Bailey Ivancic z5162498

Nabil Shaikh z5163480

Purpose and Problems

RecipeSearch is designed as a tool to reduce food wastage in a user's everyday life. It accomplishes this by suggesting recipes for the user based on available ingredients that they choose. The main issues that RecipeSearch attempts to solve are summarised in the following problem statements.

Problem Statements

1. Recipe planning that utilises ingredients that are already owned is difficult and time consuming, and lack of planning leads to unused ingredients being wasted.
2. It is tedious to keep a list of owned ingredients up to date, and the lack of such a list leads to confusion as to what food is currently available to use in recipes.

RecipeSearch solves these problems via its Ingredients Search mechanism. Planning meals is made easy as the search engine delivers recipes for meals that cater to your available ingredients, fast tracking the brainstorming process and allowing immediate commencement of cooking.

After making an account, a user can input their saved ingredients, items that they know they will always have like salt or olive oil (or ingredients that are readily available to them at the moment). It also provides a weekly meal planner that, based on chosen recipe selections for the week, will create a grocery list to optimise one's shopping choices and eradicate purchases of ingredients that will ultimately go to waste. Thus RecipeSearch will minimise food wastage and work to maximise efficient use of ingredients based on desired recipe choices.

Software Design

Features

- *US1 [User ingredient list]*
    - As a logged in user, I can input a list of ingredients that I own to store on my account, so that I know what kind of food I currently have.
- *US2 [Saving recipes as a user]*
    - As a logged in user, I can click a button to save recipes that I enjoy, so that I can make them at a later date.
- *US3 [Search recipes by ingredient]*
    - As a user, I can find a list of recipes based on my ingredients that I enter into the website, so that I know how to make recipes that are relevant to my tastes and restrictions.
- *US4 [Search recipe by name]*
    - As a user, I can search for recipes by name, so that I can find alternative ways of cooking a specific dish.
- *US5 [Planning meals for the week ahead]*
    - As a logged in user, I can plan meals ahead of time based on what I desire to eat, so that I can plan my grocery purchases and only purchase ingredients I intend to use.

User Stories

Key: US = User Story, Priority = Scale from 1 - 5 (Highest - Lowest).
Screenshots that visualise each user story are provided in the appendix (Appendix A), labelled by the appropriate user story code (e.g. US1) and the name of the feature.

Requirement: Problem Statement 2
US1 Feature: User ingredient list
As a: Student **So that:** I can easily recall what ingredients that I own **I want to:** store my ingredients that I own
Scenario: A user wants to update their saved ingredients **Given:** I am on the user page **When:** I click on the 'My Ingredients' button **Then:** I am given a way to input ingredients that I wish to save. **When:** I input an ingredient to 'My Ingredients' **And:** I click 'Add Ingredient' **Then:** The list is updated and the new ingredient is stored
Priority: 3

Requirement: Problem Statement 1
US2 Feature: Saving Recipes for later use
As a: Student **So that:** I can make recipes that I have made before **I want to:** see recipes that I have previously made
Scenario: A user likes a particular recipe and wishes to revisit it later by saving it **Given:** I am on the recipe page **When:** I click on the save recipe button **Then:** I am given feedback that the recipe is saved **Scenario:** A user wants to view recipes that they have saved **Given:** I am on my account page **When:** I click on a ‘saved recipes’ button **Then:** A list of my previously saved recipes is displayed.
Priority: 4

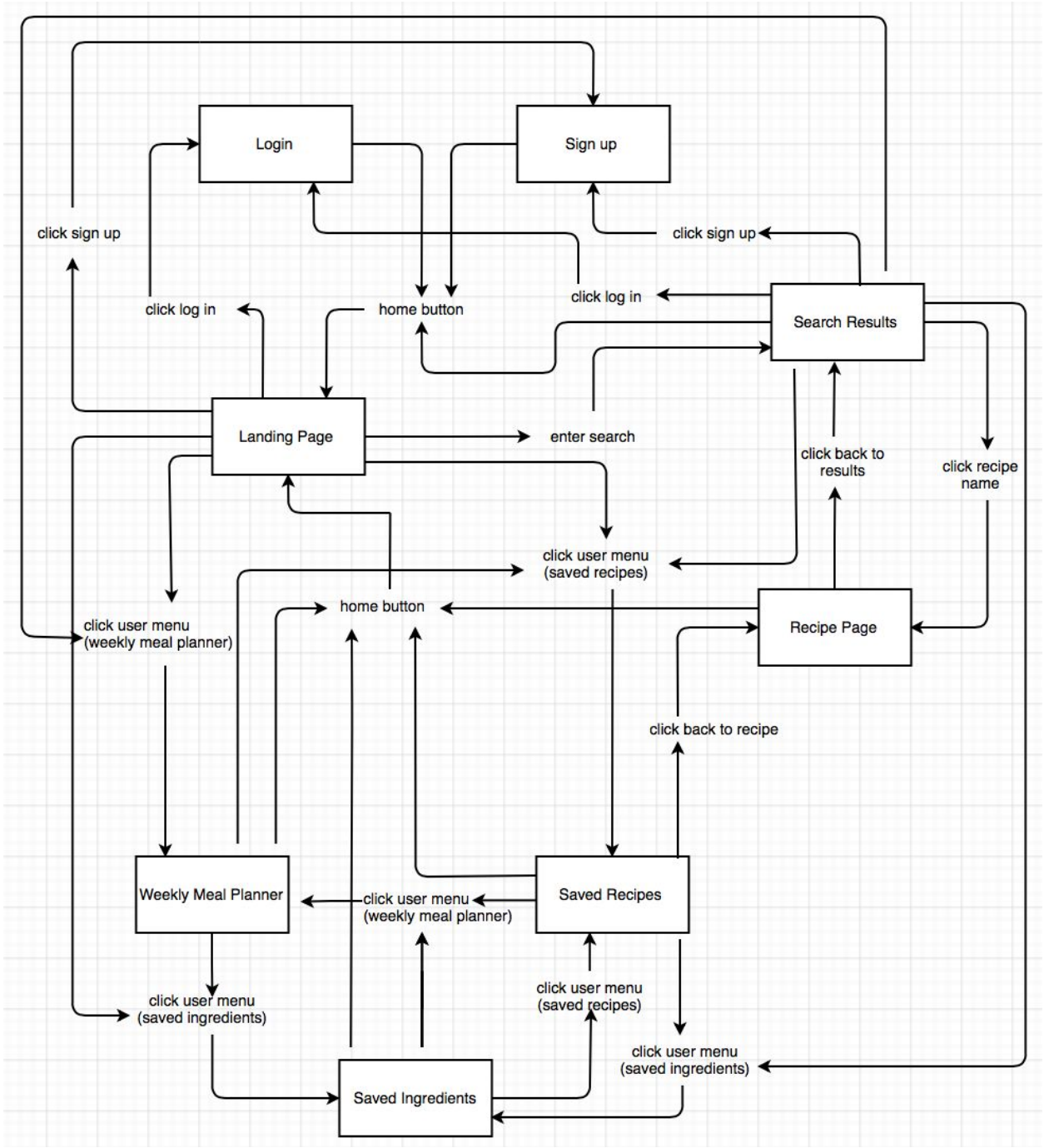
Requirement: Problem Statement 1
US3 Feature: Search for Recipe by Ingredient
As a: Student **So that:** I can search for recipes that include ingredients that I own **I want to:** find a list of recipes based on my ingredients that I enter into the website
Scenario: A user wants to cook but does not know what recipe they want to make. However they do know which ingredients they own. **Given:** I am logged in as a user, am on the homepage and have an ingredient list populated with ingredients **When:** I am on the homepage **And:** I click on a search bar titled 'Search for Recipes' **Then:** A dialog prompts me to enter ingredients that I own **And:** the list is already populated with ingredients that are associated with my account if logged in **When:** I click on the search button **Then:** A series of recipes are displayed that use the ingredients given
Priority: 1

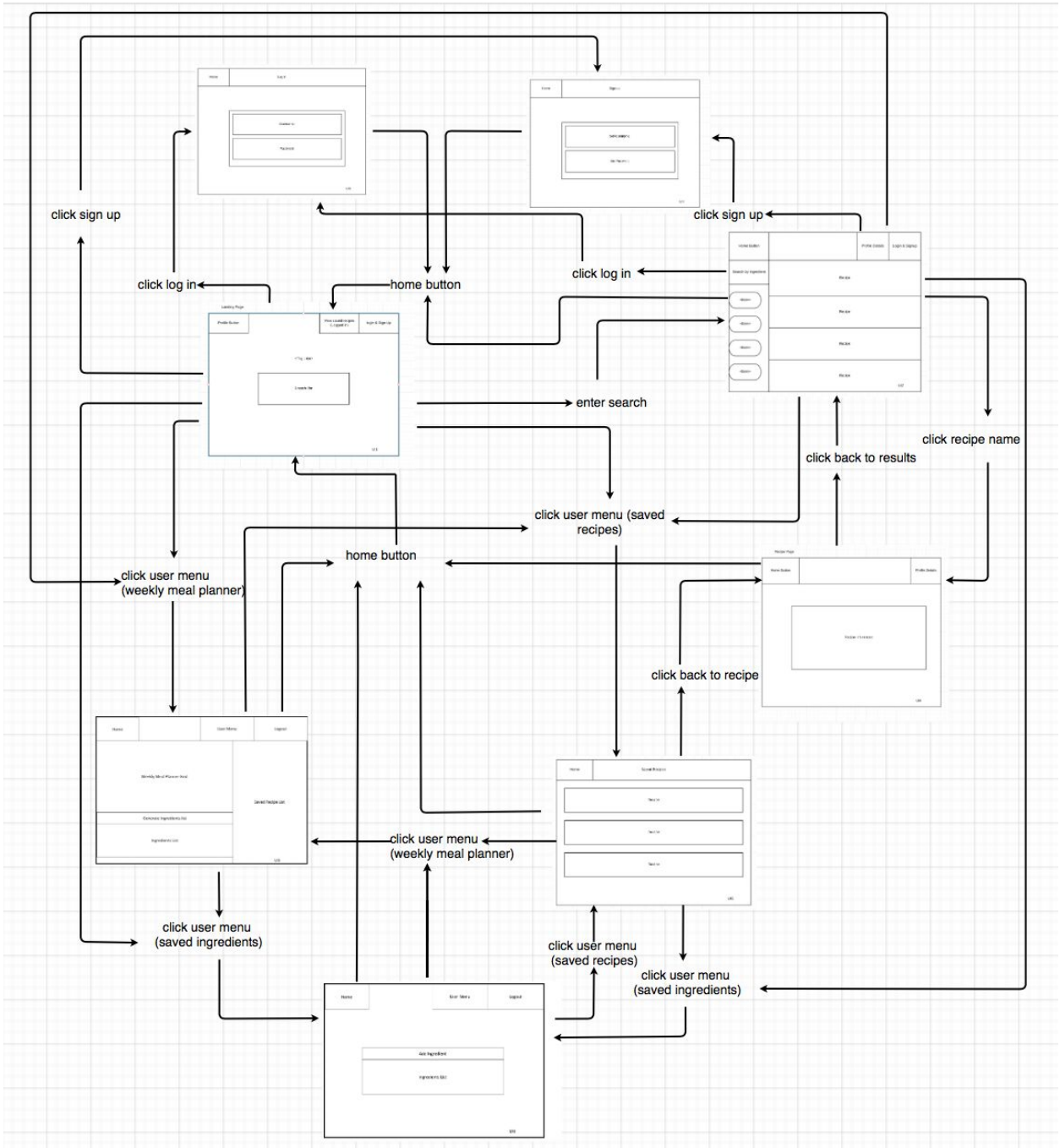
Requirement: Problem Statement 1
US4 Feature: Search recipe by name
As a: Cook **So that:** Find alternative ways of cooking a specific dish **I want to:** I search for recipes by name
Scenario: A user knows that they want to cook lasagna for dinner. **Given:** I am on the homepage **When:** I click on the 'Search by Recipe Name' tab **Then:** The search bar in the centre of the page will change so that I can search for recipes by name and not by ingredient **And:** I type in the name of the recipe I want to search for, lasagna **When:** I click on the search button **Then:** A series of lasagna recipes are displayed.
Priority: 4

Requirement: Problem Statement 2
US5 Feature: Weekly Meal Planning
As a: logged in user **So that:** I can easily organise my grocery shopping and reduce waste **I want to:** plan my meals for the week
Scenario: A user wants to plan their meals based on recipes of their choice **Given:** I am on my account page **When:** I click on the 'Weekly Planner' button **Then:** I am shown all of my of saved recipes to choose from **When:** I select a recipe to add **And:** I click and drag a recipe to the desired box in the planner **Then:** The new Recipe is added to the weekly plan, and its ingredients to a grocery list
Priority: 2

Low Fidelity Prototypes (Preliminary Designs)

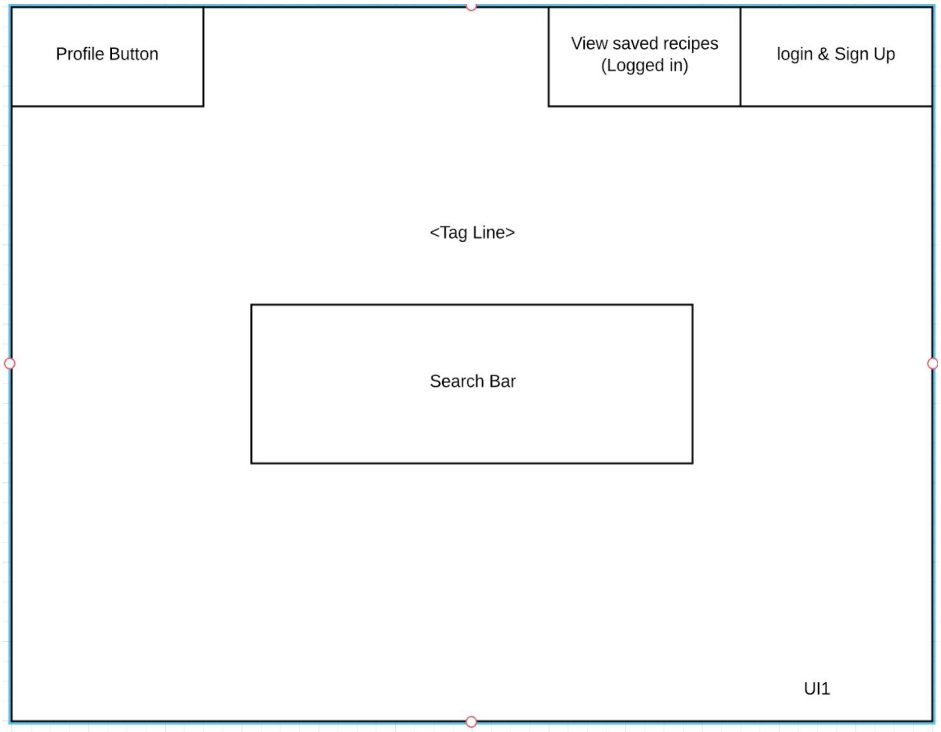
Storyboard Interactions



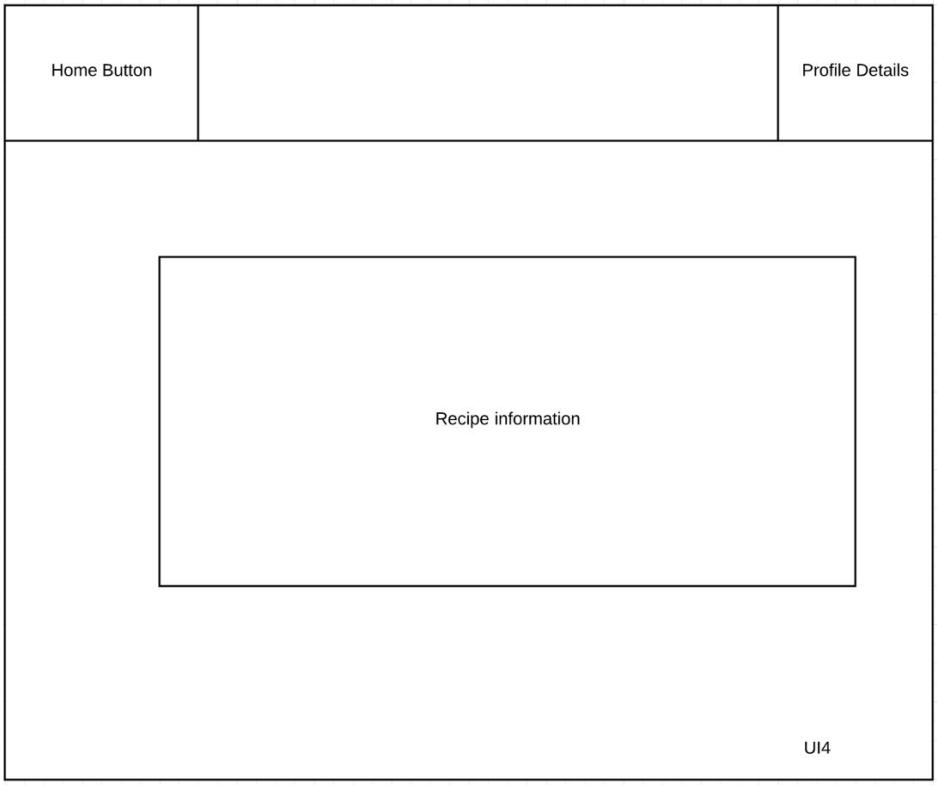

click sign up
click sign up
click log in
click log in
home button
enter search
click recipe name
click back to results
click user menu (saved recipes)
home button
click user menu (weekly meal planner)
click back to recipe
click user menu (weekly meal planner)
click user menu (saved ingredients)
click user menu (saved recipes)
click user menu (saved ingredients)

UI Component Sketches

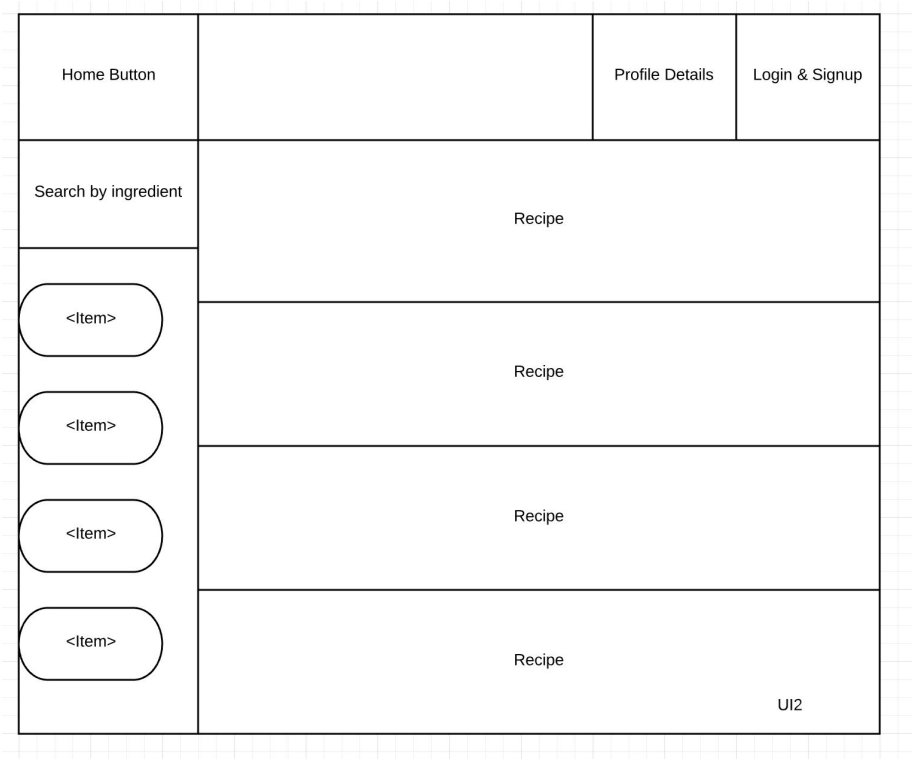
Landing Page



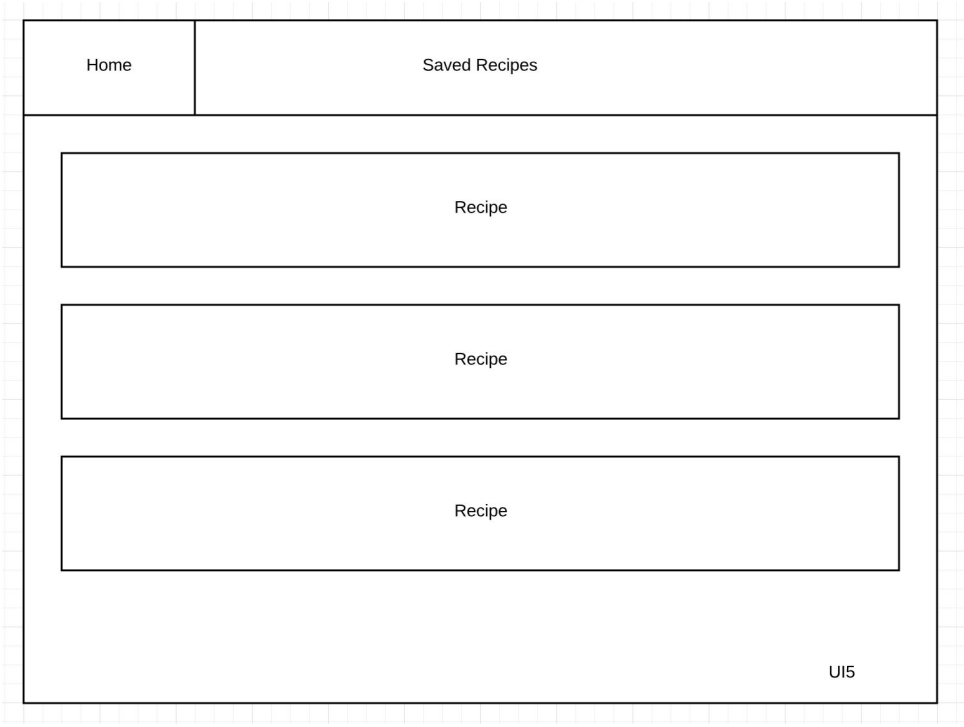
Recipe Page



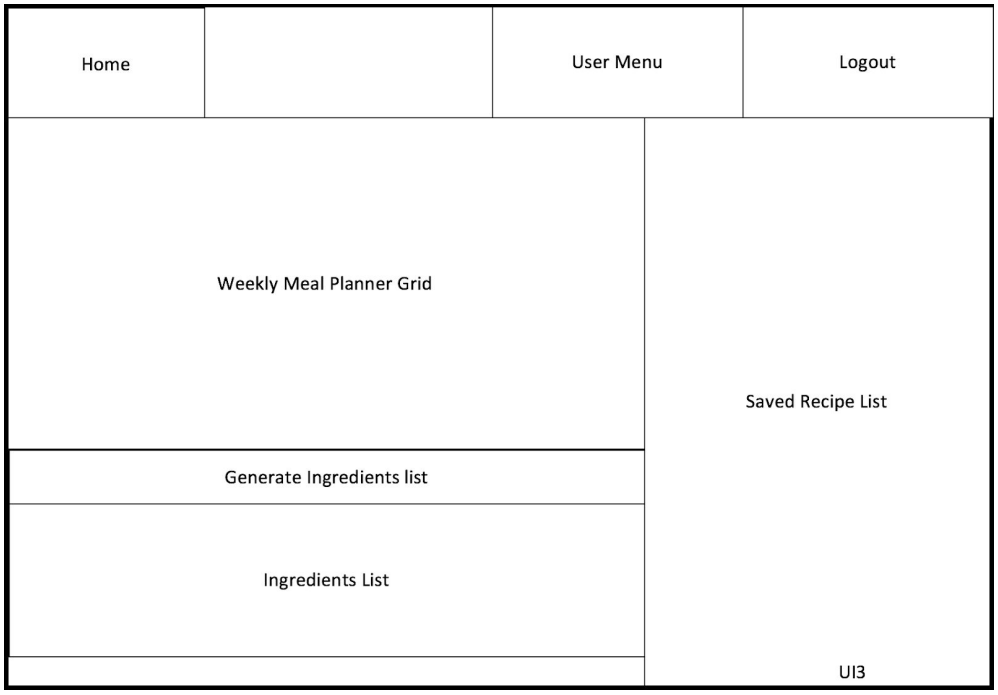
Results Page



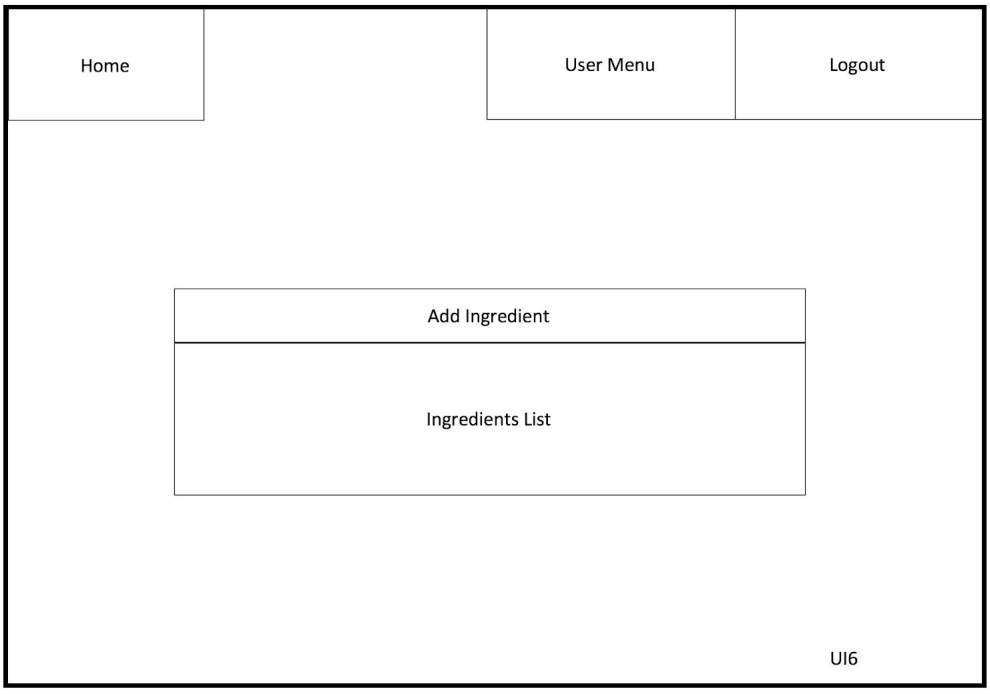
Saved Recipes Page



Weekly Meal Planner Page



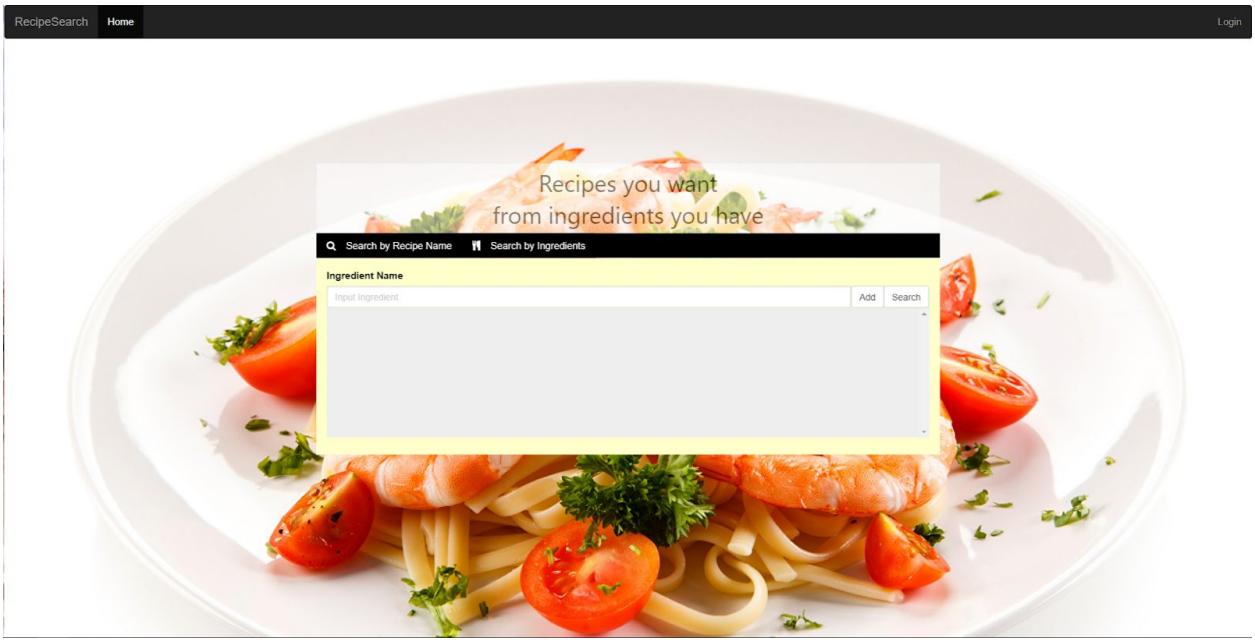
Saved Ingredients



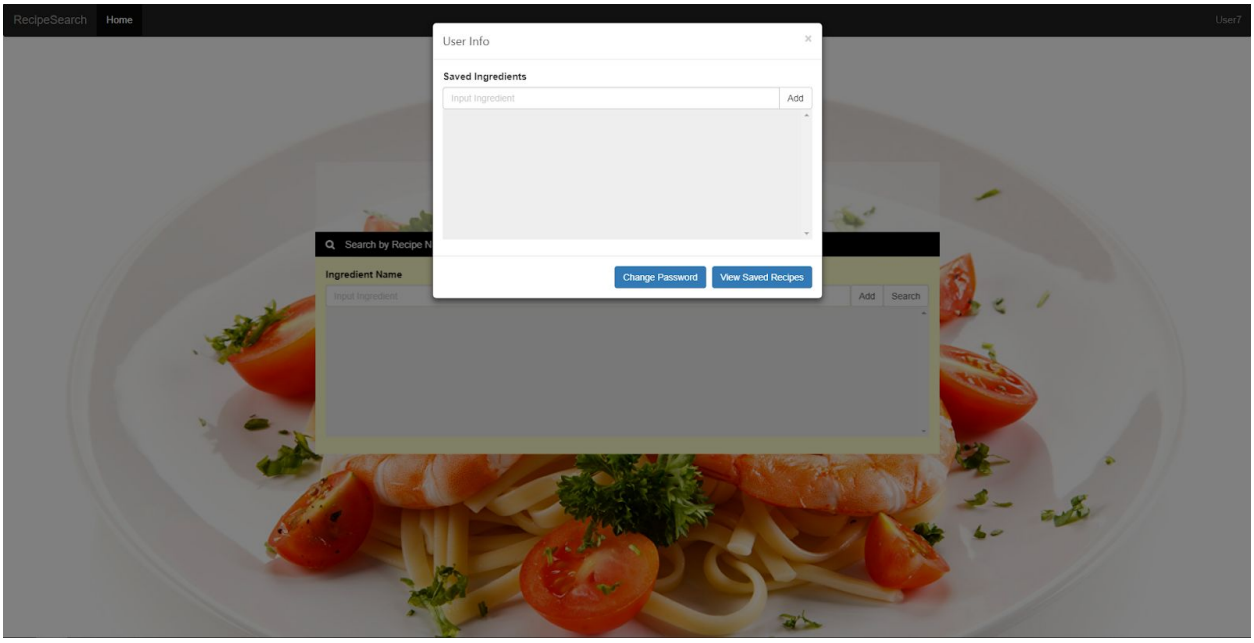
High Fidelity Prototypes (Preliminary Designs)

The following mockups were created using HTML and CSS. Evidence has been provided via screenshots of a sample of our code, which can be found in the appendix (Appendix B).

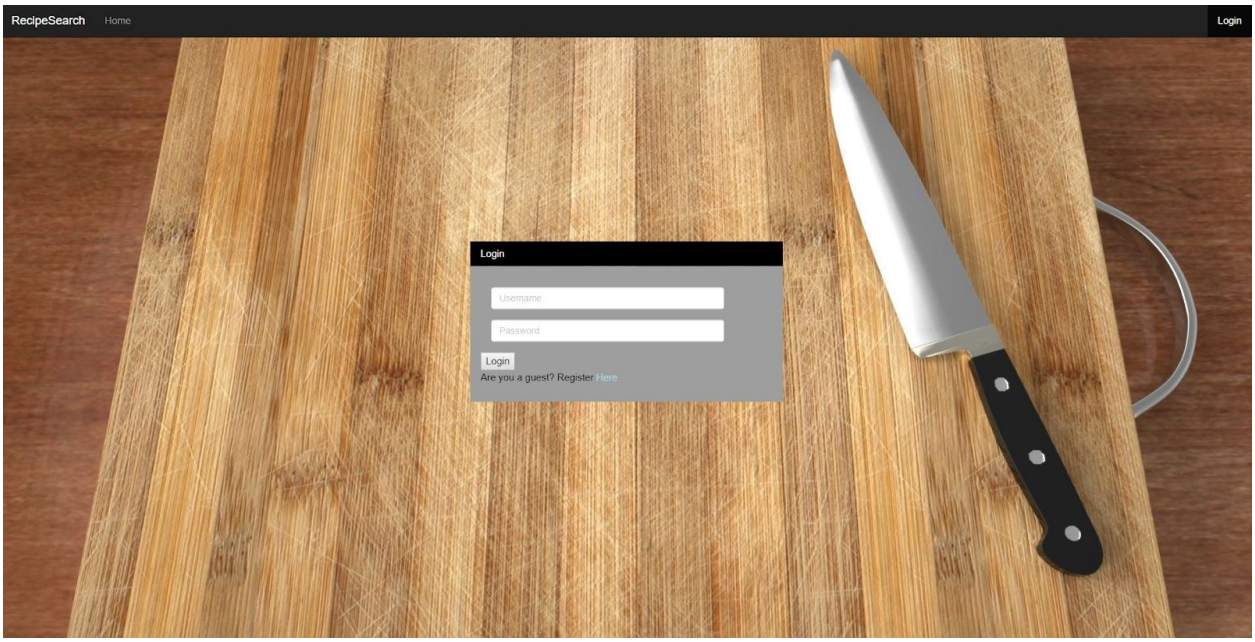
Home Page



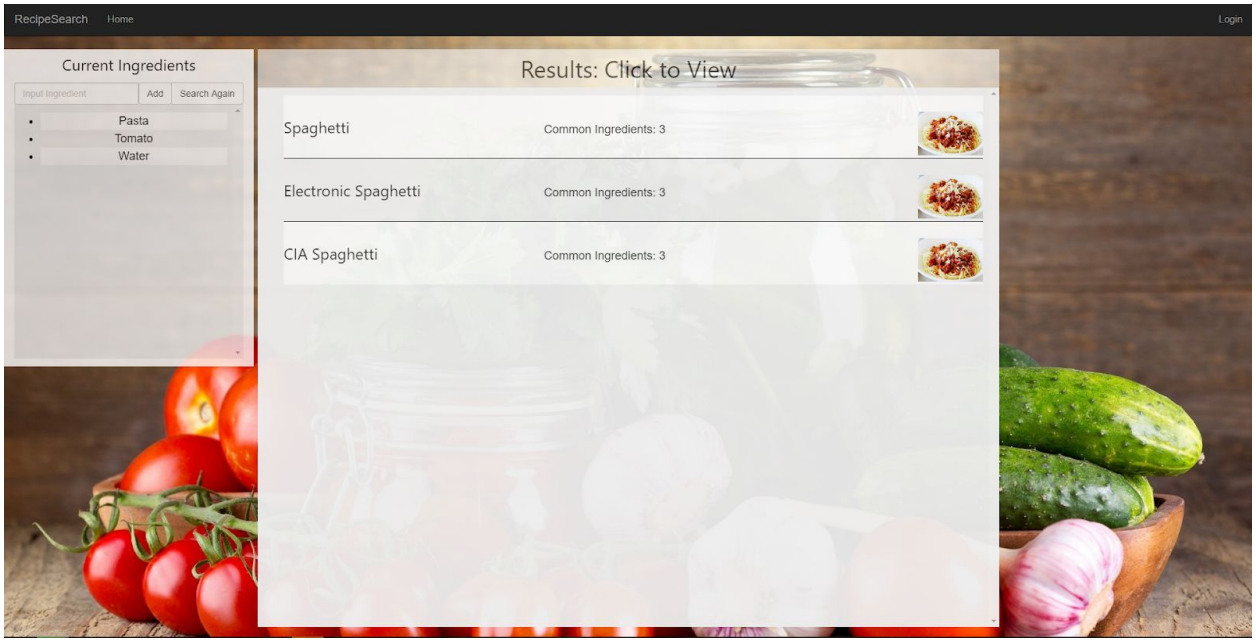
Saved Ingredients Modal



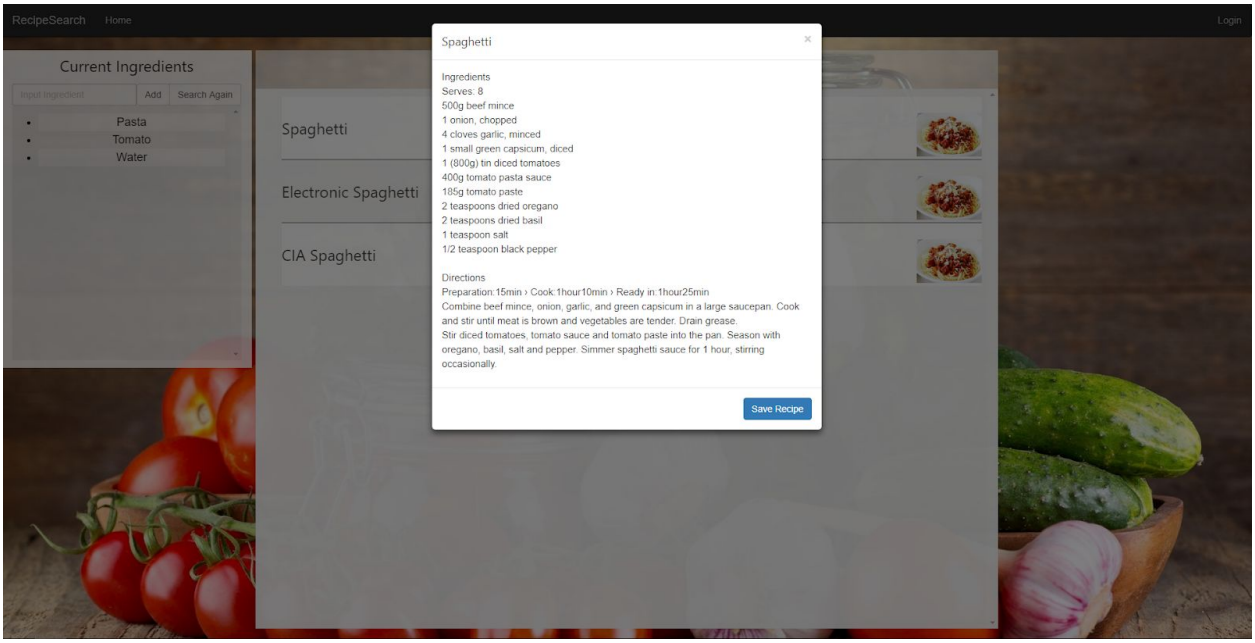
Login/Register Page



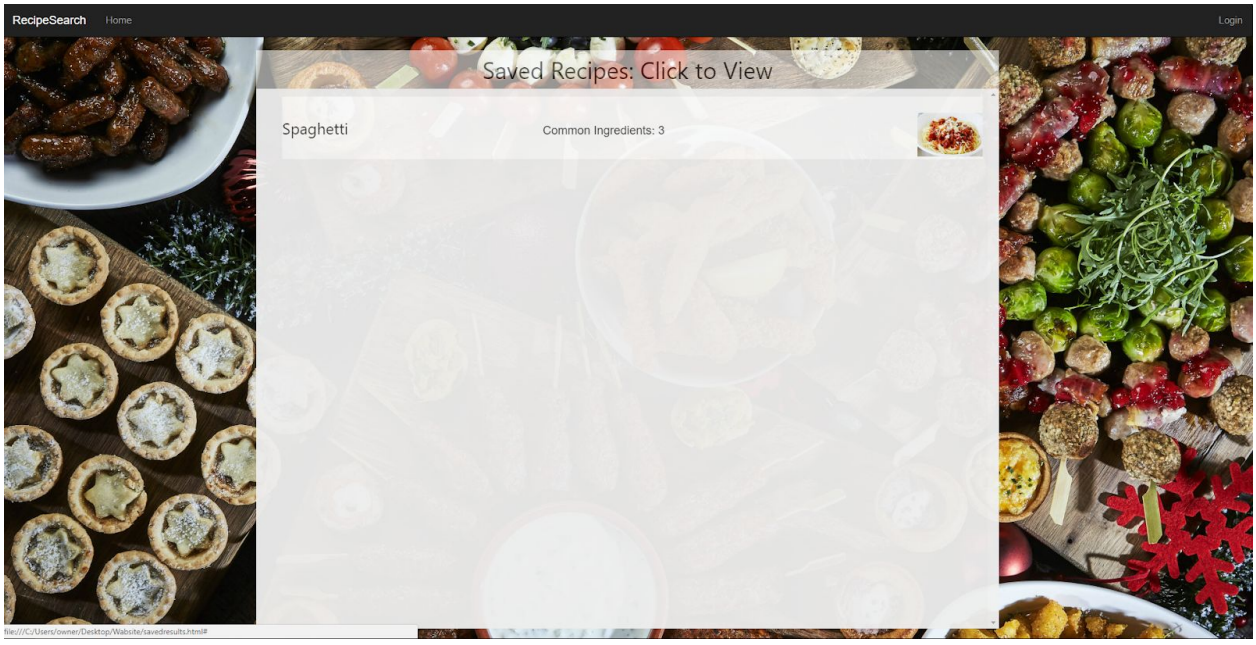
Results Page



Recipe Modal



Saved Recipes Page



Software Architecture

External Data Sources

Our main external data sources come from our recipe API, Edamam, found at https://developer.edamam.com/edamam-recipe-api

We chose Edamam API because it had the largest selection of recipes to choose from out of all the APIs that were considered. Edamam API supports both searching by ingredient keywords and by recipe, providing useful resources that we can use in RecipeSearch such as images for each recipe and a link to the website in which the recipe was originally found. While searching by ingredients, the API will display the results that feature a subset of the included ingredients.

Additionally, we used a Bootstrap template in our final product taken from freehtml5.co to assist in the visual design of RecipeSearch and ease the process of deciding on aesthetic components such as colour schemes and menu placement on each individual page.

Software Components

Software:

- Front-end (HTML, CSS, Bootstrap, JavaScript)
- Back-end (Python/Flask)
- Database (mySQL3)

Third Party:

- Web browser (Testing with Firefox and Chrome)
- Edamam API

Relating Choices to Components

- Front-end: HTML, CSS, Bootstrap, JS
- Back-end: Python/Flask
- Database: mySQL

Choice of Implementation/Framework

- HTML/CSS - Framework; suitable for website design
- Javascript - Functionality; suitable for making HTML-friendly features

- Python (Flask) - Back-end; ideal for website management
- mySQL - Database; Specialises in the management and creation of databases for the storing of user information

HTML/CSS is the standard choice that is used to make web pages, and is a powerful tool when used correctly. Almost all websites require a HTML component, and so this combined with CSS was a necessity. We chose JavaScript as our front-end language as it works in conjunction with HTML/CSS very effectively, and enabled us to use tools like Jquery to dynamically change the webpage. Since our application seeks to provide quick and easy recipe searching, our website needs to have a user interface that facilitates this, and JavaScript enabled us to do this.

The choice for Python was driven by its simplicity and its ubiquity in the web design world. We also chose it based on previous experience in using it - our team's familiarity with Python meant we could focus more on the implementation of our prototype rather than spending a large amount of time learning a new language. This in turn meant iterations were produced faster.

Flask was chosen as our back-end framework as our group has the most experience with it, and as previously mentioned, facilitated the quick development of a prototype, and hence the refinement of important things such as our weekly meal planner feature. Even though there are more current technologies available, Flask is reliable and tested, and allowed us to devote more time and effort to the UX/UI components of our application.

MySQL, another tried and tested database system, facilitated the usage of our application. The database was primarily be used for the storage of user credentials and data, and as such choosing a secure and reliable DBMS was a requirement, which mySQL fulfilled.

Choice of Platform

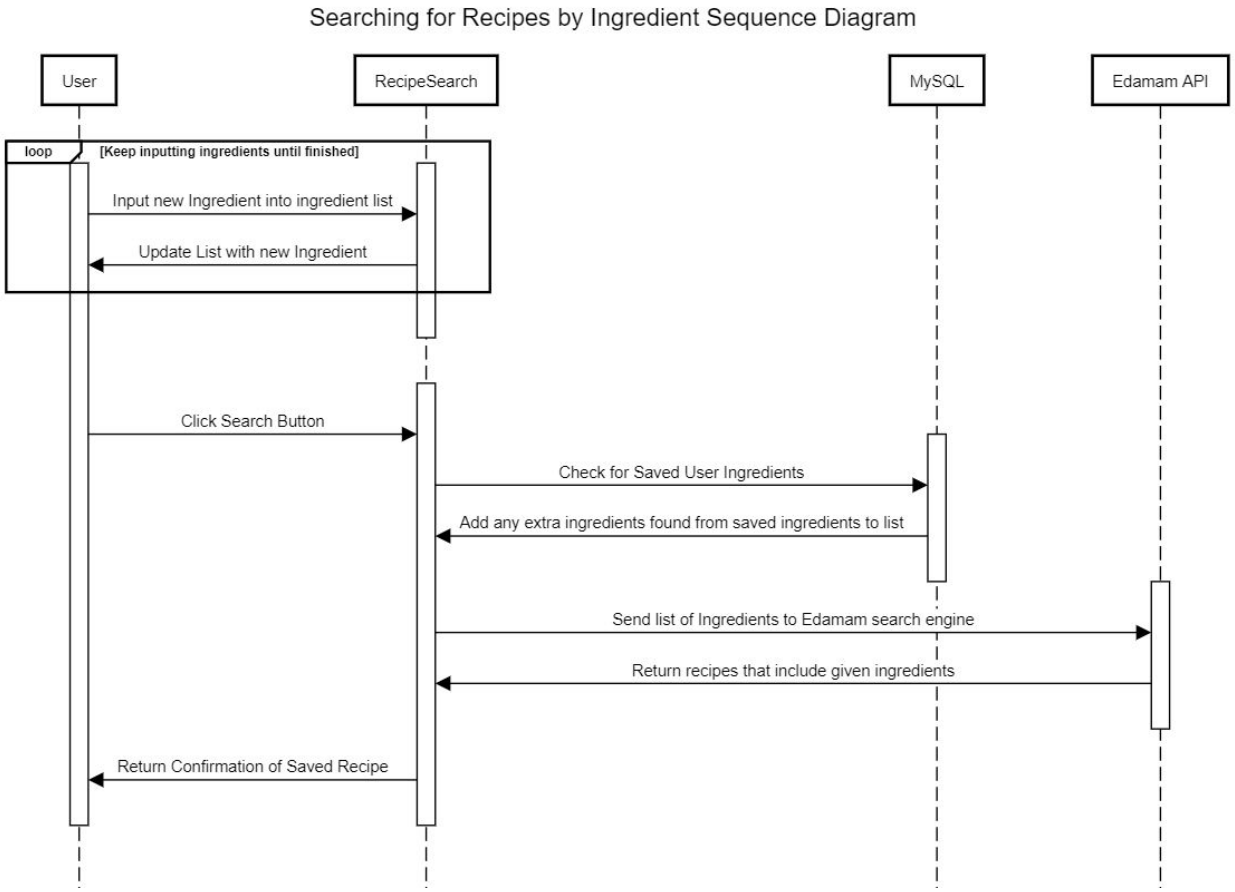
Our application was web-based, and as such any platform capable of running a web browser should be able to use it. Due to the short development time we decided to limit this to desktop machines, including Windows, Mac and Linux. RecipeSearch was tested with most modern web browsers that run on these platforms (Chrome, Firefox). Mobile testing/optimisation was considered but ultimately it was decided that traditional machines will be the main scope of the application. We came to this conclusion after considering the use cases for most users, and the likelihood of them using this application on their desktop or laptop was going to be highest, and as such decided to

cater for this platform. Although development time did not permit, we would have liked to integrate a few features that would have made interacting with the app while away from a computer much easier.

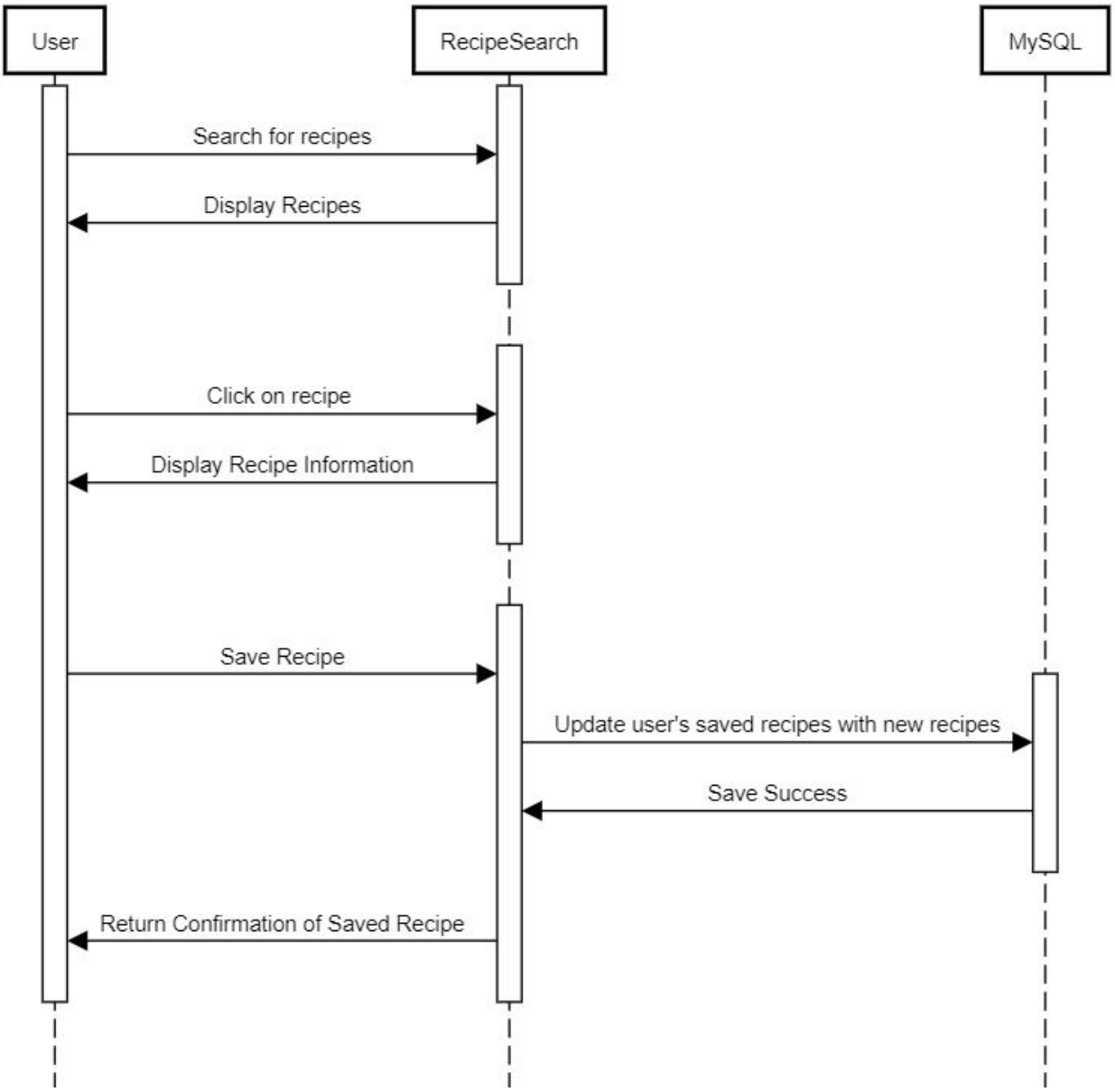
Benefits/Achievements of Architectural Choice

- Our choice of architecture allowed us to make use of a fully featured suite of web development tools that can expedite development time. We chose technologies we were familiar with in order to focus on functionality/user experience rather than implementation.
- Databases allowed to store data in an easy to access format. Our choice of DBMS technology was also secure and reliable, which was required when dealing with user information.
- HTML/CSS allowed us to quickly deploy changes to our webpages. Further to this, Bootstrap provided pre-developed code for different sections of a website that we were able to use and adjust for our specific needs.
- The Python/Flask back-end framework is a mature platform, allowing us to focus on implementation. Our team has also had the most experience with these two technologies, and as such development time will be shorter than using something we are unfamiliar with.
- Edamam API contains over 1 million recipes, and provides many extra details for each recipe such as kilojoule count, ingredients and difficulty. Using this API, we were able to do search and acquire the desired results. The API was also well documented, making it easy to integrate into our application.

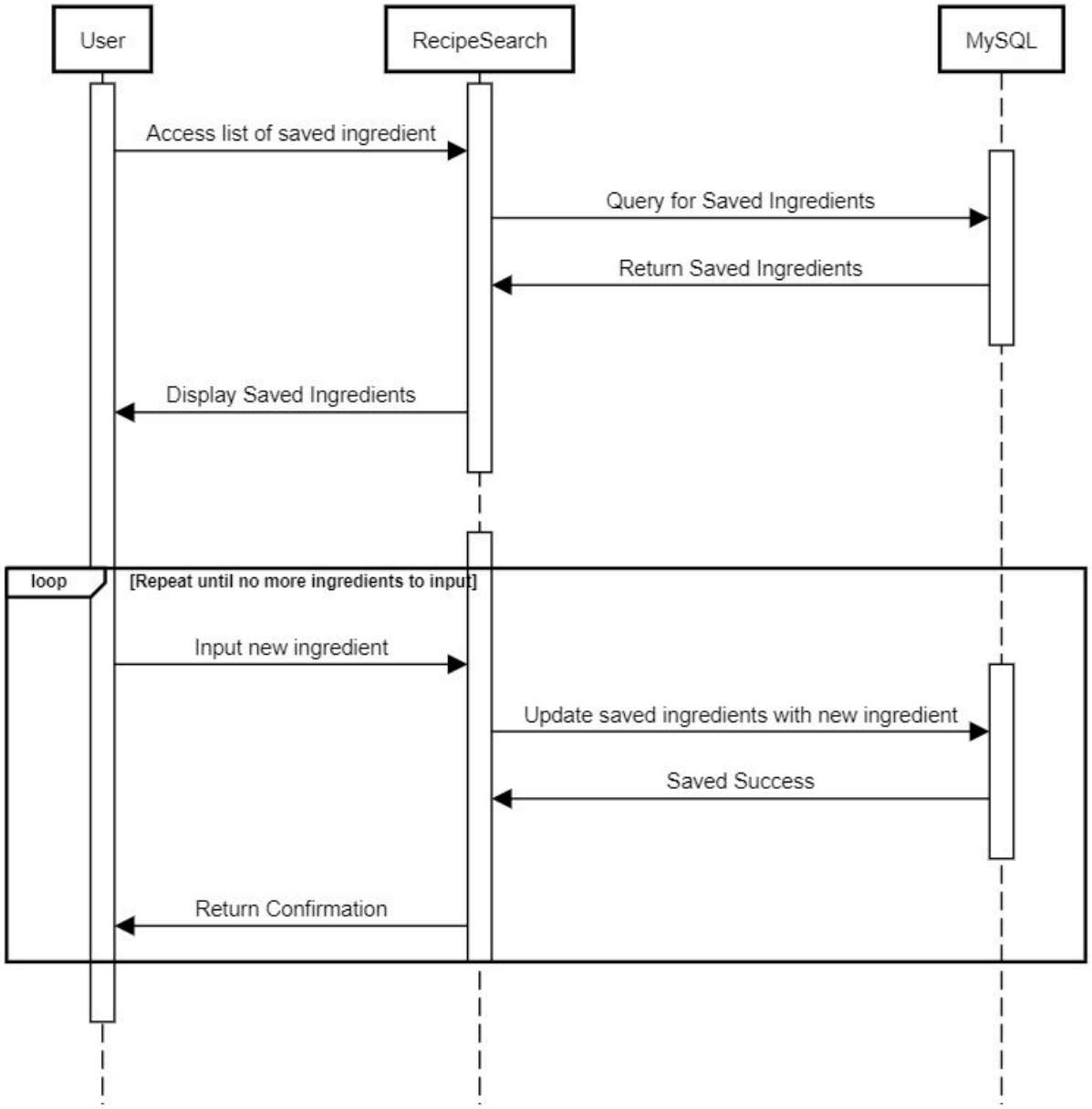
Sequence Diagrams

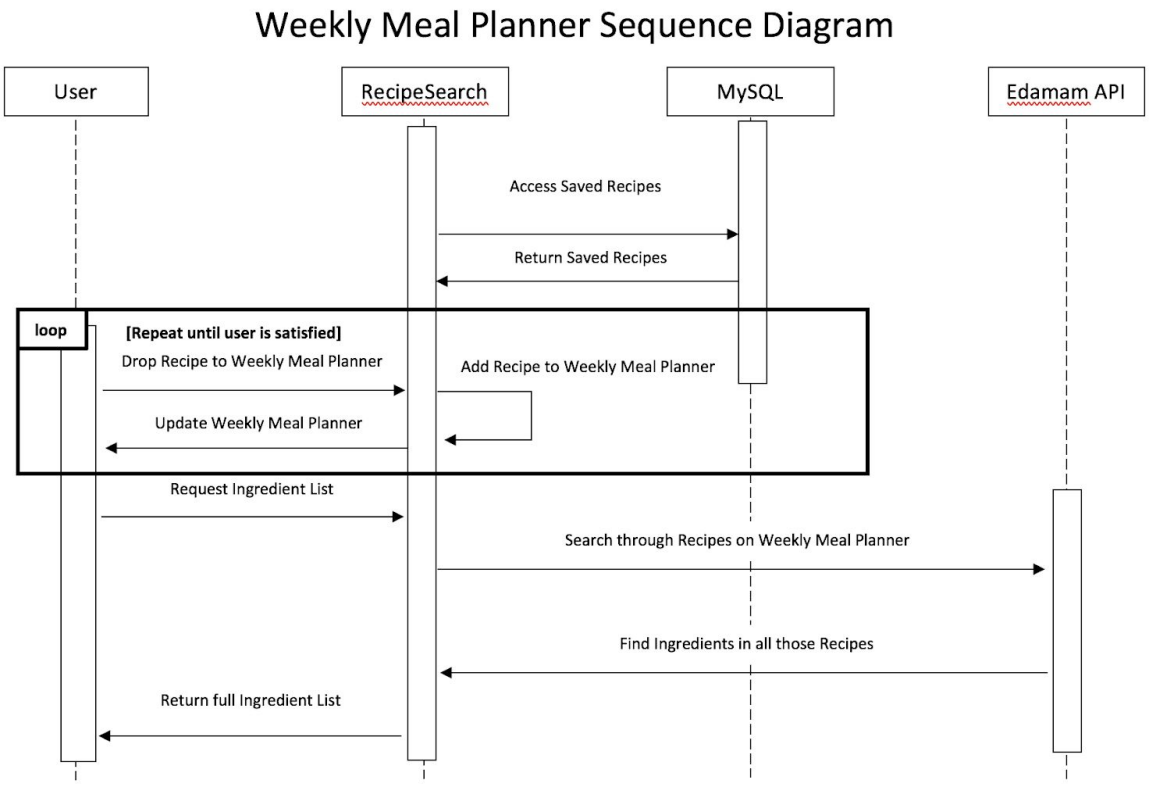


Saving Recipes for later use Sequence Diagram



Inventory Taking Sequence Diagram





Conclusion

Overall, we would consider the current state of our website prototype a success. We were able to provide a viable solution to our problem statements, and did so in a way that is functionally stable (and expandable) and aesthetically pleasing, something we hope the public would actually want to use if it were to be polished.

Our team worked well together as a whole. Although there were moments where opinions conflicted, we always managed to discuss our opinions in a civil manner, reaching a compromise that everyone was happy with. The responsibilities of our team members were as follows:

Ryan: Created the high fidelity prototypes, and developed the front-end HTML/CSS/JS for RecipeSearch, along with some of the Flask backend that interfaced with the SQL database. Ryan researched the initial API to be used for the project and was a proponent of Edamam Recipe API. Ryan was also a contributing member of the core RecipeSearch development team, whose primary duties were creation and testing of recipe searching functionality, ingredient filtering capabilities and implementation of user

settings. Furthermore, he implemented many of the routing functions found within the web framework and the interface it had with the SQL database.

Bailey: Was active within the design and the development of the project. Bailey helped design the UI of the system, completing low fidelity prototypes, as well as building on this design through being a member of the core RecipeSearch development team during the prototype phase onwards. Bailey helped integrate the prototype to a new, updated UI through the use of Bootstrap templates, in which he was a member of the refactoring team. While working on the development of the application, Bailey also conducted testing to ensure stability and functionality. Bailey also focused mainly on the user elements of the application, contributing to features like saving ingredients, saving recipes and the weekly planner.

Nabil: Produced high-fidelity screenshots of the website however was mostly active during the technical development of the project. During the first iteration of the website, Nabil helped with the development and maintenance of basic features such as the recipe and ingredient search. His responsibilities during the final development process were the implementation of UI features and the database system. More specifically, he worked on the login system, refinement of the recipe and ingredient search functionality, textbox bubbles, flask queries to the database and finally the weekly planner.

Estella: Assisted in the system design and analysis of user interaction and experience of RecipeSearch throughout the development process. This involved testing the website and providing feedback to the team about bugs or other issues that needed to be fixed. Estella also contributed to the front end development of the prototype, and was largely involved in the writing and compilation of the final report.

Jacob: Was a database collaborator, creating a schema for a user's saved recipes and ingredients. Additionally, Jacob was a core developer of the final report having edited and written a large portion of it, and the primary language and grammar editor of all submitted documents.

This project gave us a good insight of the process required when building an application from scratch. We have previously only provided documentation or worked on code up to a certain point (e.g. only user stories and UMLs or the bare-bones scaffolding of a website). The creation of our website forced us to think more deeply about the value a user could derive from our service, and in what way we would be able to guide them step by step through our application in order to fully realise its potential.

Not only did this lead to more emphasis on user interactions and experience (and, later on, the overall aesthetic of our website design) but also made us realise the importance of scope - we were warned early on by our mentor to be mindful of keeping our scope small, lest we become too ambitious and are eventually incapable of delivering what we had proposed.

We also learnt, however, that creating a website was not just about programming and design. It was also about communicating and pitching our ideas, making them sound convincing enough to potential clients and investors (or in our case, judges). This, then, was a situation where we were able to practice a valuable real life skill.

Time was the main obstacle in the development of RecipeSearch. Certain features and exploration of alternatives were often beyond what we could manage in the given timeframe, resulting in forgone features. This was mainly due to certain sections of the website taking longer to implement, such as linking our database and front end features, particularly when it came to the weekly meal planner, as well as assignments from other subjects and outside commitments.

Additionally, our Edamam API, though useful, kept us largely restricted to its own capacity. When we attempted to add features beyond the scope of Edamam, we found that incorporating these features (such as sorting by dietary and difficulty information of recipes) was extremely difficult. This was partially due to the language in which the API is written (Javascript), and using different APIs meant that ideally they would all use Javascript as well. They instead used languages such as NodeJS which we were not familiar with and would have to learn about before we started using it. By the time we realised Edamam API did not fully cater to all that we wish it did, it was too late to switch to any alternative APIs we had found.

In the future, we would focus more on our desired target audience and make development choices around how we conceive their user experience. In our development, most of the focus was placed on functionality and getting things working without proper consideration of how and why a user would use features or find it helpful - We expected them to have a thought process we considered ‘intuitive’ when coming to our website. This factor permeated many areas of our development process, resulting often in confusion and somewhat awkward development decisions.

As a functional improvement, we would like RecipeSearch to include the filters that we did not manage to implement based on dietary requirements. It would also have been greatly beneficial if we managed to devise a way of adding the same ingredients

together when generating an ingredients list from the weekly planner - Edamam works by gathering recipes from different websites, which means the measurement system for ingredients aren't the same. Some may use grams and kilos, others use ounces and pounds. Making the effort to equate these measurements and present it in a cohesive format to the users would have no doubt granted our website extra value.

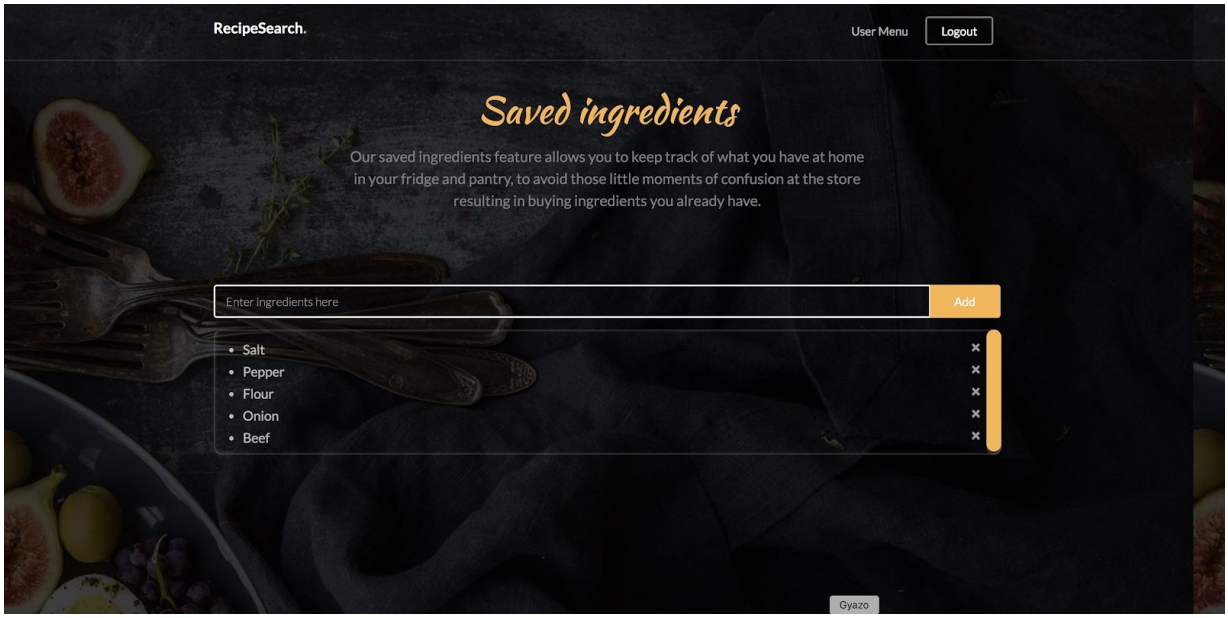
Lastly, for improvement in development time, we would consider not using HTML, CSS and Javascript for our frontend. Had we invested time into researching other frameworks such as AJAX, ReactJS or Angular, perhaps we would have spent less time on trivial design implementations and more time on the back end.

Nevertheless, we have learnt from our (many) mistakes. SENG2021 has provided us with valuable experience in terms of project management and design, and we have attained practical hard and soft skills we can apply in the future.

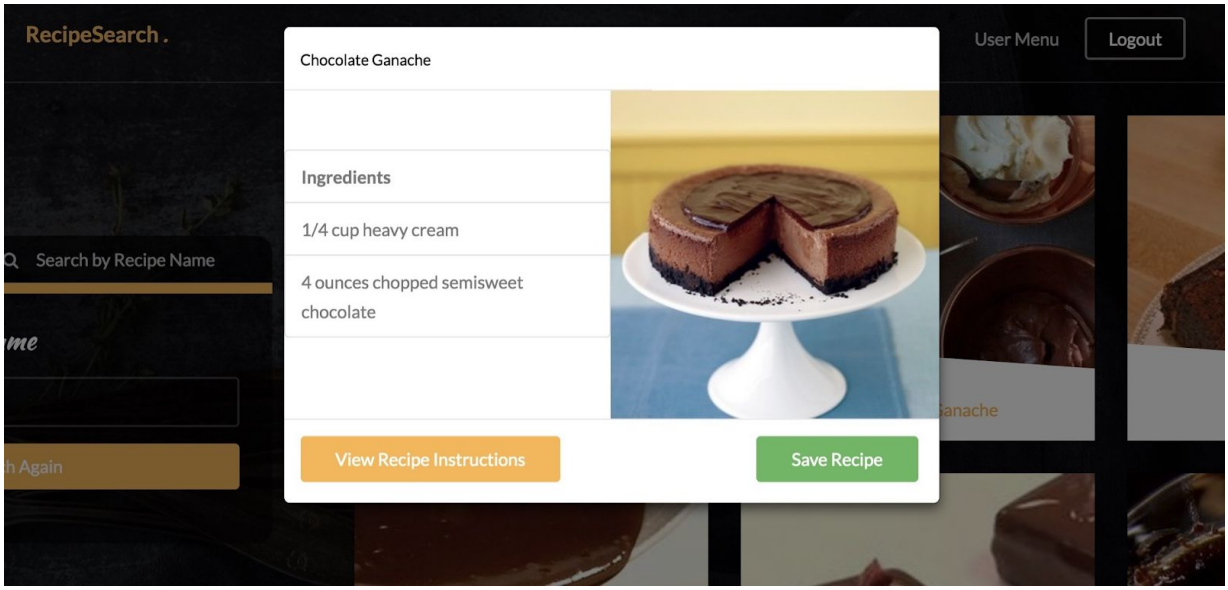
Appendix

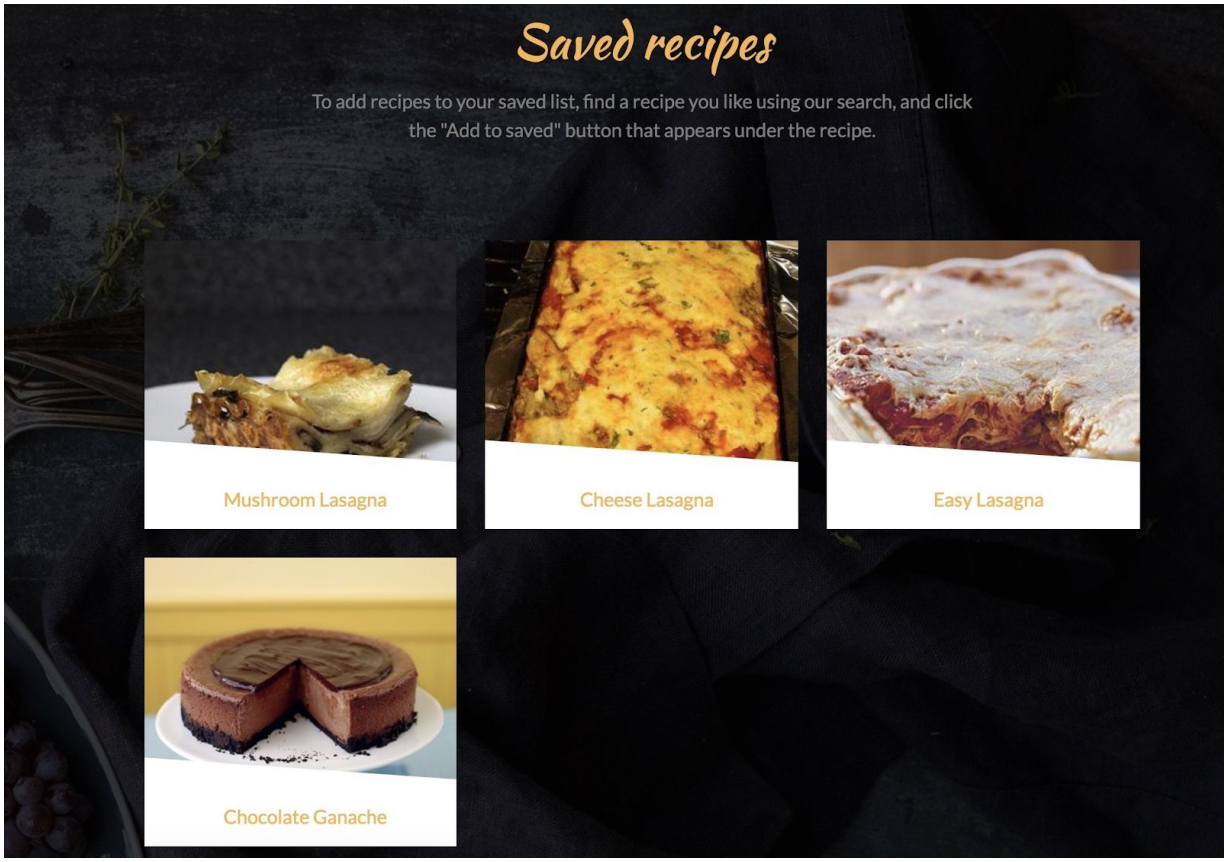
Appendix A: User Story Screenshots

US1: Saving Ingredients Screenshot

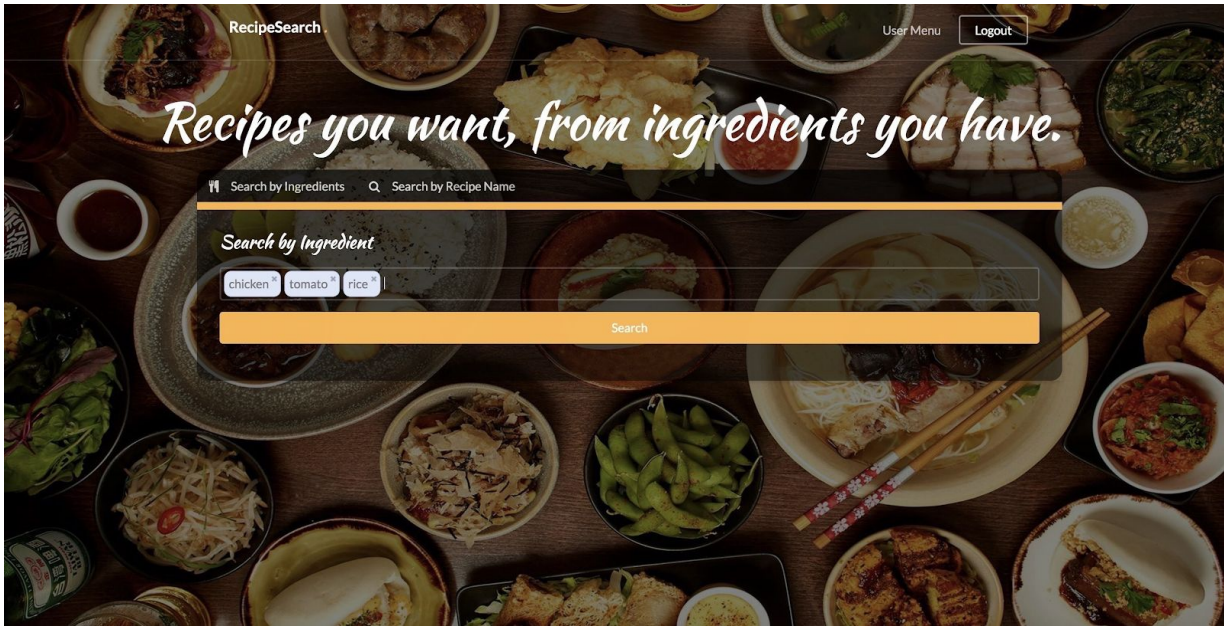


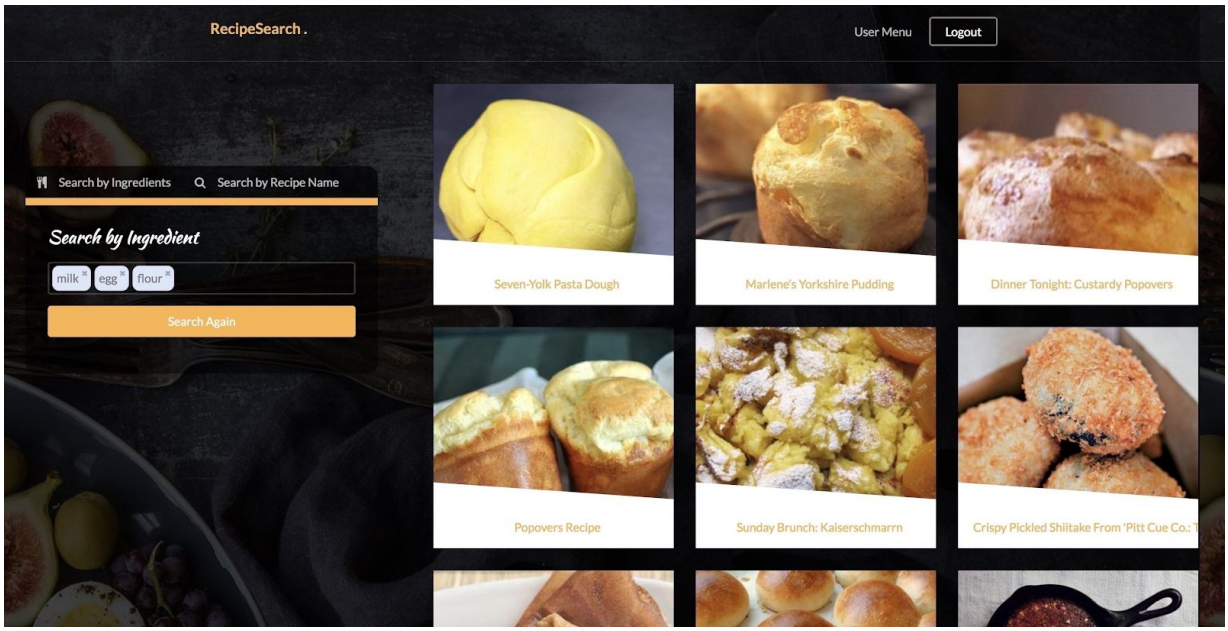
US2: Saving recipes for later use screenshots



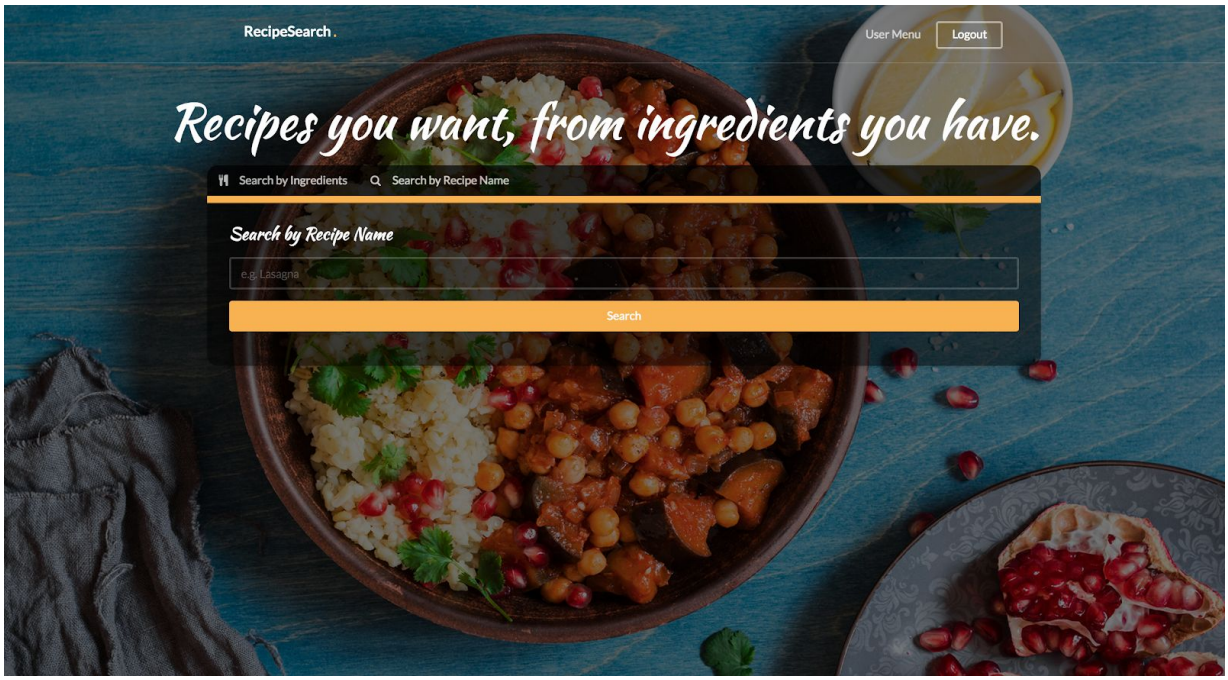


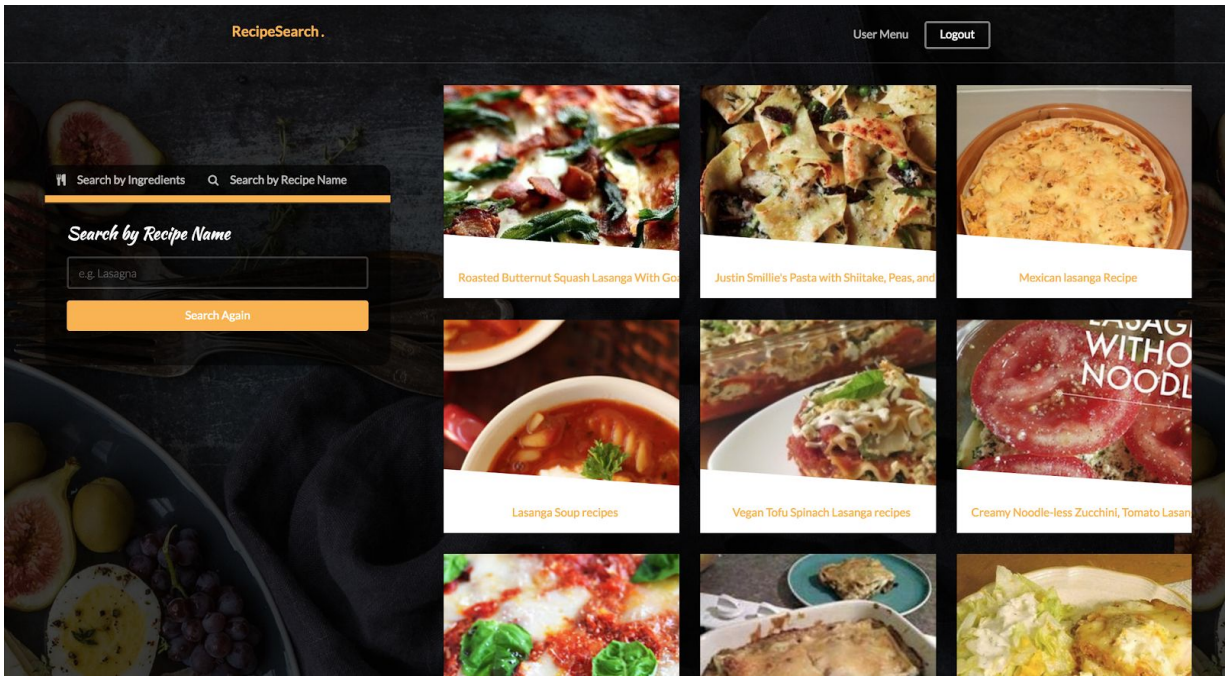
US3: Search recipes by ingredient screenshots



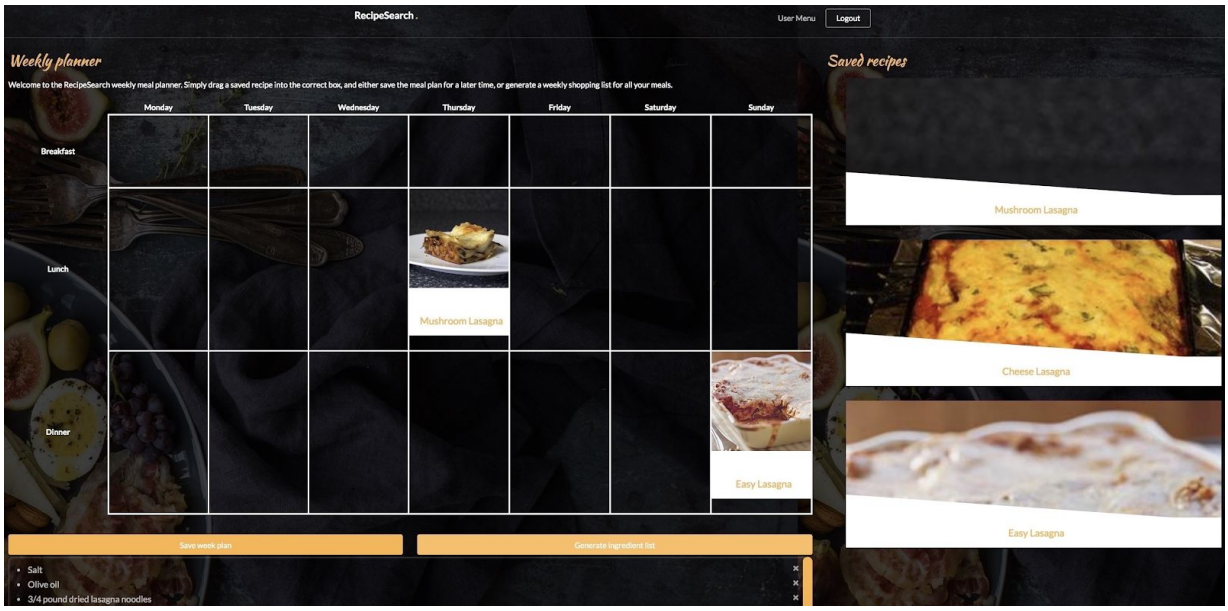


US4: Search recipe by name screenshot





US5: Planning meals for the week ahead

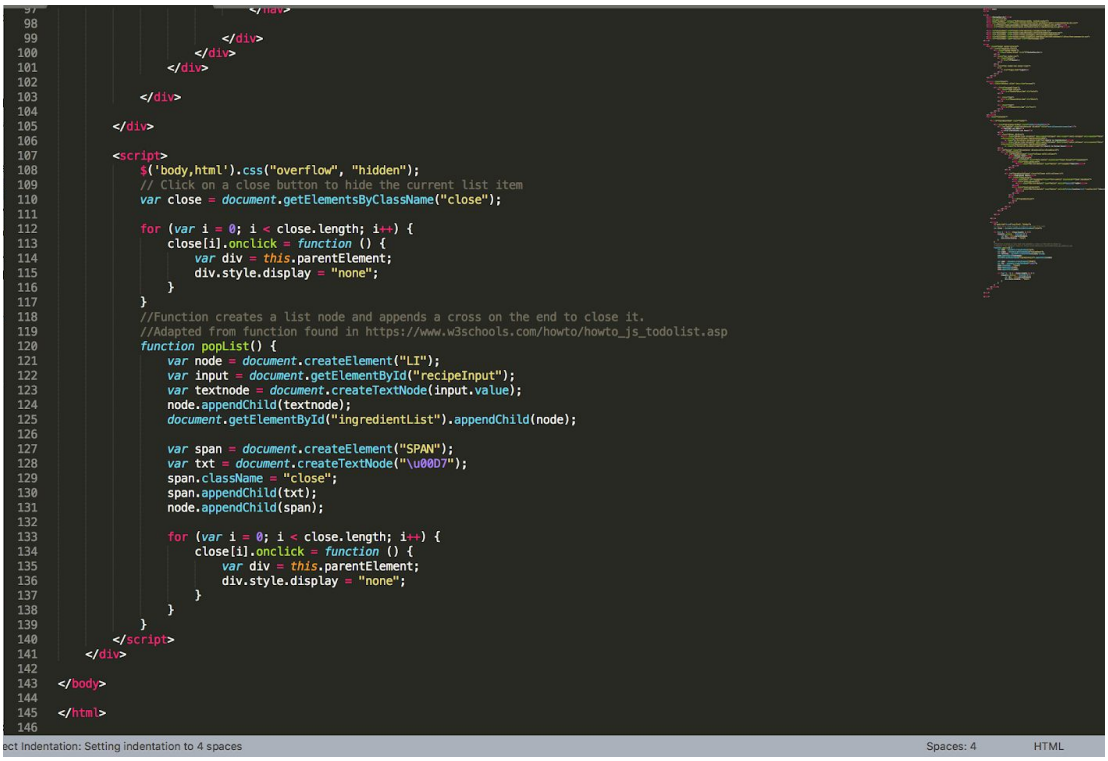


Appendix B: High Fidelity Code Fragments

HTML

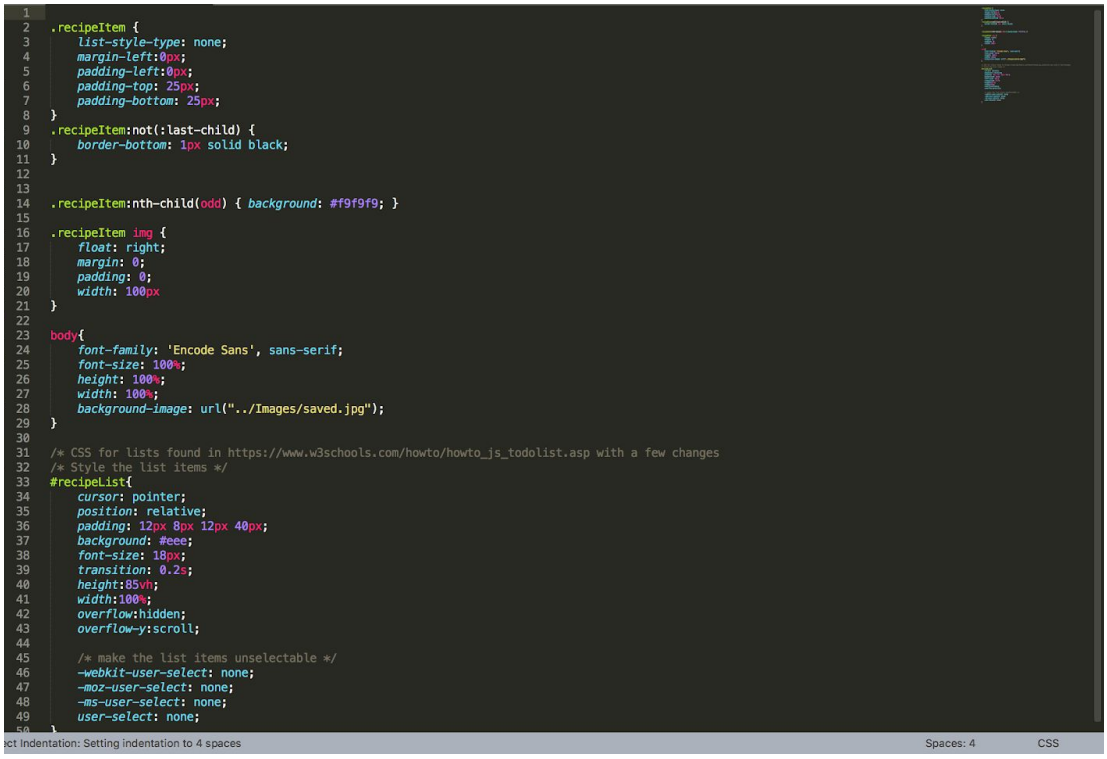
```
<!DOCTYPE html>
<html>

<head>
    <title>RecipeSearch</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
    <link rel="stylesheet" href="https://www.w3schools.com/lib/w3-theme-blue-grey.css">
    <link rel='stylesheet' href='https://fonts.googleapis.com/css?family=Open+Sans'>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" type="text/css" href="CSS/homepage.css">
</head>

<body>
    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="#">RecipeSearch</a>
            </div>
            <ul class="nav navbar-nav">
                <li class="active">
                    <a href="#">Home</a>
                </li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
                <li>
                    <a href="savedresults.html">View Saved Recipes</a>
                </li>
                <li>
                    <a data-toggle="modal" data-target="#userModal">User7</a>
                </li>

            </ul>
        </div>
    </nav>

    <section class="block">
        <div class="carousel slide" data-ride="carousel">

            <div class="carousel-inner">
                <div class="item active">
                    <img src="Images/Salad.jpg" alt="salad">
                </div>

                <div class="item">
```

Spaces: 4 HTML

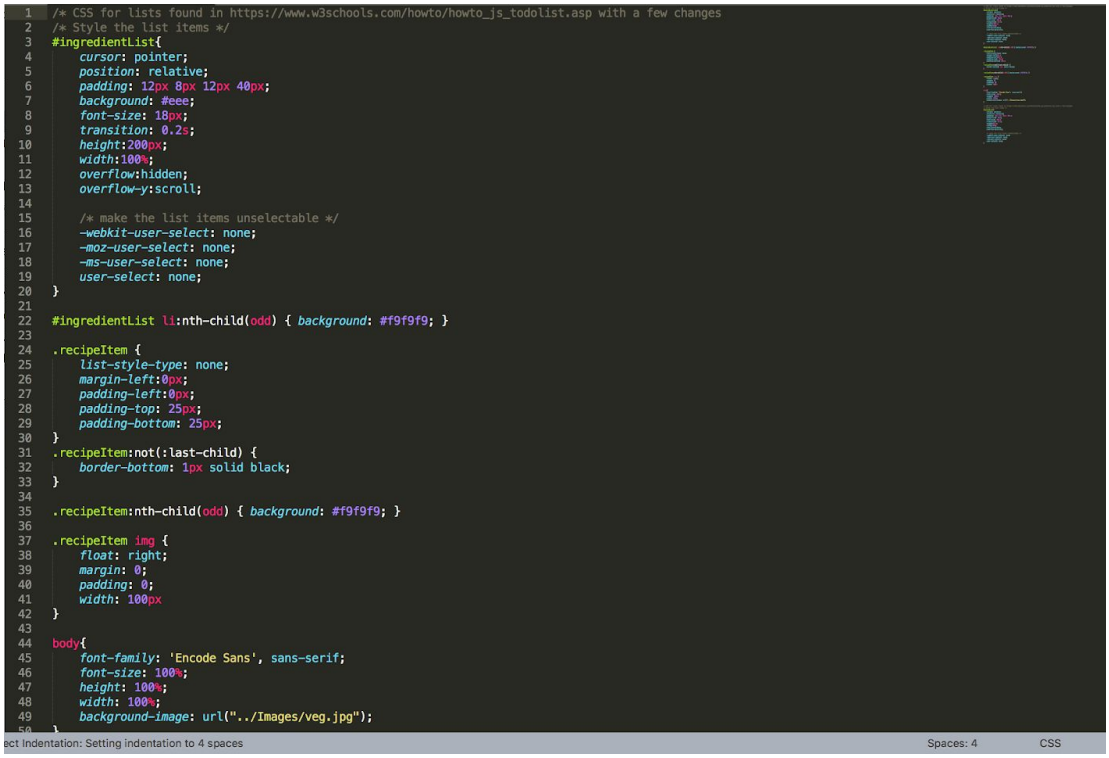


```
                    </nav>
                </div>
            </div>
        </div>

    </div>

    <script>
        $('body,html').css("overflow", "hidden");
        // Click on a close button to hide the current list item
        var close = document.getElementsByClassName("close");

        for (var i = 0; i < close.length; i++) {
            close[i].onclick = function () {
                var div = this.parentElement;
                div.style.display = "none";
            }
        }
        //Function creates a list node and appends a cross on the end to close it.
        //Adapted from function found in https://www.w3schools.com/howto/howto_js_todolist.asp
        function popList() {
            var node = document.createElement("LI");
            var input = document.getElementById("recipeInput");
            var textnode = document.createTextNode(input.value);
            node.appendChild(textnode);
            document.getElementById("ingredientList").appendChild(node);

            var span = document.createElement("SPAN");
            var txt = document.createTextNode("\u00D7");
            span.className = "close";
            span.appendChild(txt);
            node.appendChild(span);

            for (var i = 0; i < close.length; i++) {
                close[i].onclick = function () {
                    var div = this.parentElement;
                    div.style.display = "none";
                }
            }
        }
    </script>
  </div>

</body>

</html>

ect Indentation: Setting indentation to 4 spaces                    Spaces: 4    HTML
```

CSS



```
.recipeItem {
    list-style-type: none;
    margin-left:0px;
    padding-left:0px;
    padding-top: 25px;
    padding-bottom: 25px;
}
.recipeItem:not(:last-child) {
    border-bottom: 1px solid black;
}


.recipeItem:nth-child(odd) { background: #f9f9f9; }

.recipeItem img {
    float: right;
    margin: 0;
    padding: 0;
    width: 100px
}

body{
    font-family: 'Encode Sans', sans-serif;
    font-size: 100%;
    height: 100%;
    width: 100%;
    background-image: url("../Images/saved.jpg");
}

/* CSS for lists found in https://www.w3schools.com/howto/howto_js_todolist.asp with a few changes
/* Style the list items */
#recipeList{
    cursor: pointer;
    position: relative;
    padding: 12px 8px 12px 40px;
    background: #eee;
    font-size: 18px;
    transition: 0.2s;
    height:85vh;
    width:100%;
    overflow:hidden;
    overflow-y:scroll;

    /* make the list items unselectable */
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}

ect Indentation: Setting indentation to 4 spaces                    Spaces: 4    CSS
```



```css
/* CSS for lists found in https://www.w3schools.com/howto/howto_js_todolist.asp with a few changes
/* Style the list items */
#ingredientList{
    cursor: pointer;
    position: relative;
    padding: 12px 8px 12px 40px;
    background: #eee;
    font-size: 18px;
    transition: 0.2s;
    height:200px;
    width:100%;
    overflow:hidden;
    overflow-y:scroll;

    /* make the list items unselectable */
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}

#ingredientList li:nth-child(odd) { background: #f9f9f9; }

.recipeItem {
    list-style-type: none;
    margin-left:0px;
    padding-left:0px;
    padding-top: 25px;
    padding-bottom: 25px;
}
.recipeItem:not(:last-child) {
    border-bottom: 1px solid black;
}

.recipeItem:nth-child(odd) { background: #f9f9f9; }

.recipeItem img {
    float: right;
    margin: 0;
    padding: 0;
    width: 100px
}

body{
    font-family: 'Encode Sans', sans-serif;
    font-size: 100%;
    height: 100%;
    width: 100%;
    background-image: url("../Images/veg.jpg");
}
```