

Recipe Ready

Design Document

Team 32

Michael Pike, Moshi Li, Austin Wentz, Vasudha Kashyap, Joseph Marshall

Table of Contents:

1. Purpose
2. Design Outline
3. Design Issues
4. Requirements
 - a. Functional
 - b. Non-Functional
5. Sample Views
6. Design Details

Purpose

Currently, most recipe websites require you to know what you want to eat and expect you to purchase the ingredients you don't have to make the desired recipe. However, more often than not, people are unsure about what to eat and have no interest in buying additional ingredients to make their meal. With our webapp, individuals will be able to search for recipes based on ingredients they already have, allowing them to find recipes based on those ingredients.

Functional Requirements:

I. Users can login and register with an email and password

As a user:

1. I want to be able to register with an email address and password.
2. I want to be able to log in with my email and password.
3. I want to be able to log out.

II. Dietary preferences:

As a user:

1. I want to be able to store a list of ingredients that represent allergies/dietary restrictions to my profile.
2. I want to be able to add dietary restrictions by ingredient.
3. I want to be able to remove dietary restrictions by ingredient.

III. Shopping Lists:

As a user:

1. I want to store separate lists of ingredients by amount as "shopping lists" on my profile
2. I want to be able to delete and rename shopping lists

IV. Users can track their available ingredients by storing ingredients in a "pantry"

As a user:

1. I want all of my ingredients to be stored to my profile in my "pantry".
2. I want to be able to add ingredients to my pantry from a shopping list.
3. I want ingredients in my pantry to be subdivided into "packages" representing separate packages or quantities of each ingredient added at different times.
4. I would like to be able to specify an expiration date for each package in my pantry.
5. I want to be able to specify a purchase date for each package in my pantry.

6. I want the purchase date to be set to the date on which the package was entered by default.
7. I would like the ingredients in my pantry to be shown in descending order of the expiration date of the package with the closest expiration date.
8. I want each ingredient entry in the pantry view to display the expiration date of the next package that will expire, along with the amount in that package

V. Users can interact with various features by entering ingredients into a form

As a user:

1. I want to be able to enter a list of ingredients by amount into an ingredients form.
2. I want to be able to add ingredients to my pantry from the ingredients form.
3. I want to be able to add ingredients to a new shopping list from the ingredients form.
4. I want ingredients that are in my dietary restrictions list to be omitted from the ingredients form

VI. Users can search for recipes by ingredient

As a user:

1. I want recipes found during searches to be ignored if they contain ingredients in my dietary restrictions.
2. I want to be able to automatically add ingredients I do not have for a recipe to my shopping list
3. As a user, I would like to be able to see what percentage of ingredients I have for a recipe

VII. Users can access a recipe view form

As a user:

1. I want to be able to “confirm” a recipe and have the required ingredients be subtracted from my available ingredients.
2. I would like the recipe view to display separate lists of ingredients that are in my pantry and not in my pantry.
3. I would like to be able to add ingredients from the recipe view to a pre-existing or new shopping list

VIII. If time permits

As a user:

1. I want to have minor ingredient (spices) replacements suggested to me.
2. I want to know where I can buy ingredients I need for a recipe.
3. I would like to be able to search for recipes by cuisine style.

4. I would like to be able to scan grocery store item's barcodes.
5. I would like to be able to submit a feedback form.
6. I want my ingredients to be stored as browser cookies.
7. I want displayed recipes to be sorted based on the proportion of required ingredients available in my pantry.
8. I would like to have recipes suggested to me based on the items I have in my saved list of ingredients

IX. **Developer Wishlist**

As a developer:

1. I want recipes to be obtainable from the internet via web crawling*
2. I would like to be able to view user feedback (if time permits).
3. I would like to gather statistics on my users (if time permits).

Non-Functional Requirements:

I. **View Requirements**

1. Front-end user interface should scale well on mobile and desktop browsers.
2. Front-end user interface should be user friendly and aesthetically pleasing.

II. **Performance Requirements:**

1. Front-end user interface should be very responsive.
2. Recipe search queries should return results promptly.

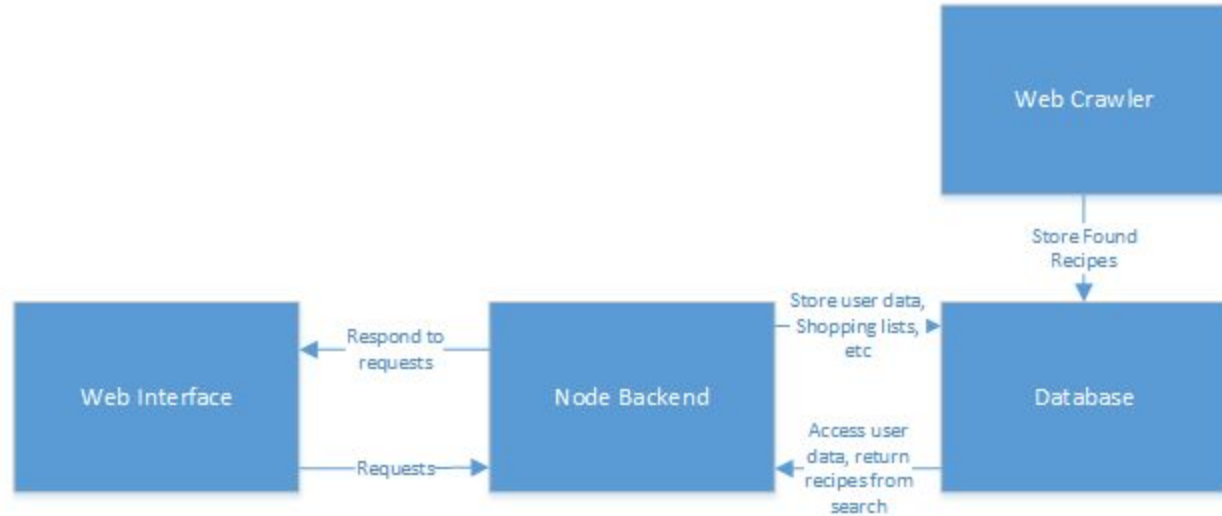
III. **Load Requirements:**

1. The server should be able to support 100 concurrent active users
2. Database should store login info, preferences, and pantry entries for up to 500 users.

IV. **Security Requirements:**

1. Passwords should be stored as salted hashes.
2. Passwords (hashes) and user info should be stored securely.

Design Outline



Web Interface: The browser front-end will provide forms for login and registration. It will also have various views allowing users to create and manage shopping lists, manage the pantry, set dietary restrictions and preferences, and search for recipes by ingredients.

Node.js Backend: The backend will respond to client requests by validating users, sending user information, and running recipe searches on the database. It will also interface with the web crawler by initializing recipe searches and formatted search results for storage in the database. The server will be built in Node.js.

Web Crawler: The web crawler will be hosted on the server. It will crawl various sites for recipes during times of low server load and return search results.

Database: The database will store all user information including login info, settings, dietary restrictions, pantry contents, and shopping lists. It will also store recipes found via web crawl to allow search results to be cached, speeding up user searches.

Design Issues

- 1) Do users need to create an account to get on our website?
 - Option 1: User must have an account to use our website.
 - Option 2: User does not need an account to use our website
 - **Option 3: Users do not need an account to use our website, but users need an account for functions other than searching recipes.**

We decided to go with option 3, because option 1 is unlikely to attract users and option 2 is unlikely to maintain consistent users.

- 2) What should we use for our database?
 - Option 1: We use a relational database such as MySQL
 - **Option 2: We use a non relational database such as MongoDB**

We wanted to choose a non-relational database because the recipes we will be fetching come from a variety of websites and we cannot predefine our schema, our collection will include records that have varying attributes. We chose MongoDB, because it has fast queries, is horizontally scalable, easily manageable, dynamic, and lightweight. Moreover, it is mostly in JavaScript, which our team members are familiar with.

- 3) What front end framework should we use?
 - **Option 1: Angular.js**
 - Option 2: React.js
 - Option 3: Ember.js
 - Option 4: Backbone.js

We decided to use Angular as a front end for multiple reasons. It has client-side MVC and creates a single page application that is responsive. It has excellent data binding (only one with two way data binding which was the largest difference in our other options). It is built for easy testing as it comes bundled with Protractor.

- 4) What back end framework(s) should we use?
 - Option 1: PHP
 - **Option 2: Node.js with Express.js**
 - Decision: We are using Node.js because it is extremely lightweight and efficient

Express is linked tightly to Node as well, and we will be using that as a “fast, unopinionated, minimalist” web framework with Node. Express abstracts out some low level logic that can could cause unnecessary problems if we went without. Express also helps with file organization for a MVC structure and we will use it for routing. Using Node.js would also allow us to code in JavaScript on the client and server; we wouldn’t have to switch gears. Node.js, however, when compared to PHP, does not integrate with as many older technologies, which is not an issue for us because we don’t plan to be working with any obscure API’s.

5) Which method of authentication should we use

- Option 1: Cookie based authentication
- Option 2: Token based authentication
- **Option 3: Passport.js**

We chose to use passport.js, a node.js middleware, because it is easiest to implement, since we are already using Express. The other two options have a steeper learning curve and passport would allow us to easily add multiple user login options (Facebook, Twitter, Google, etc).

6) What type of architecture should we use?

- Option 1: Unified architecture
- **Option 2: Model View Control architecture**

We are using Model View Control architecture because that is how our other technologies in the MEAN stack will set up our webapp.

7) Where should the shopping list view be placed?

- **Option 1: Pull-out drawer on home screen and “My Pantry” page**
- Option 2: Only on “My Pantry” page
- Option 3: A completely separate page

We decided that we will implement the shopping list view as a pull-out sidebar that can pop-up both while searching for recipes and also while visiting the “My Pantry” page because users could become especially aware of what they don’t have while also thinking about the ingredients they do have. It would prove especially effective if the user could immediately write down or select foods they need to buy as they remember them.

Sample Views

Registration Page - Simplistic registration screen with logo and email/password/confirm password boxes

Login Page - Simplistic login screen with logo and email/password boxes

Landing/Search Page - Contains a search bar where ingredients can be entered to search for recipes. Contains links to the user's pantry, logout, and preferences.

Search Results Page - Contains list of recipes from search results and expand view for whole recipes. Also buttons to do frequent actions such as Cook (remove ingredients from pantry), Favorite (save recipe), and Add Needed Ingredients to current shopping list

Pantry Page - Displays list of ingredients, subdivided into "packages". List of Shopping Lists on the side

Preferences Page - Allows users to set and change dietary restrictions (other things?)

Shopping List View - Pop-up view to display and allow editing of the contents of multiple shopping lists

Recipe View - Pop-up view for a single recipe with a thumbnail and link to its host website. Can expand when prompted to show the ingredients from the user's pantry and available amounts that will be used in the recipe in a list in green. Show the ingredients required in a parallel list in red.

Design Details

Database Design

We will be using three different collections in our database to store information:

- **User**

- Email:string
- Password:hashed
- userIngredients: object (list<userIngredient>)
- shoppingList: object (list<recipeIngredient>)
- recipeList: object (list<recipe>)
- dietaryList: object (list<recipeIngredient>)

The “User” collection will contain records that represent each user, including their email, password, a list of ingredients they own, a shopping list of ingredients they must buy, a list of favorited recipes, and a list of ingredients they cannot have (dietary preferences).

- **Ingredient**

- name: string

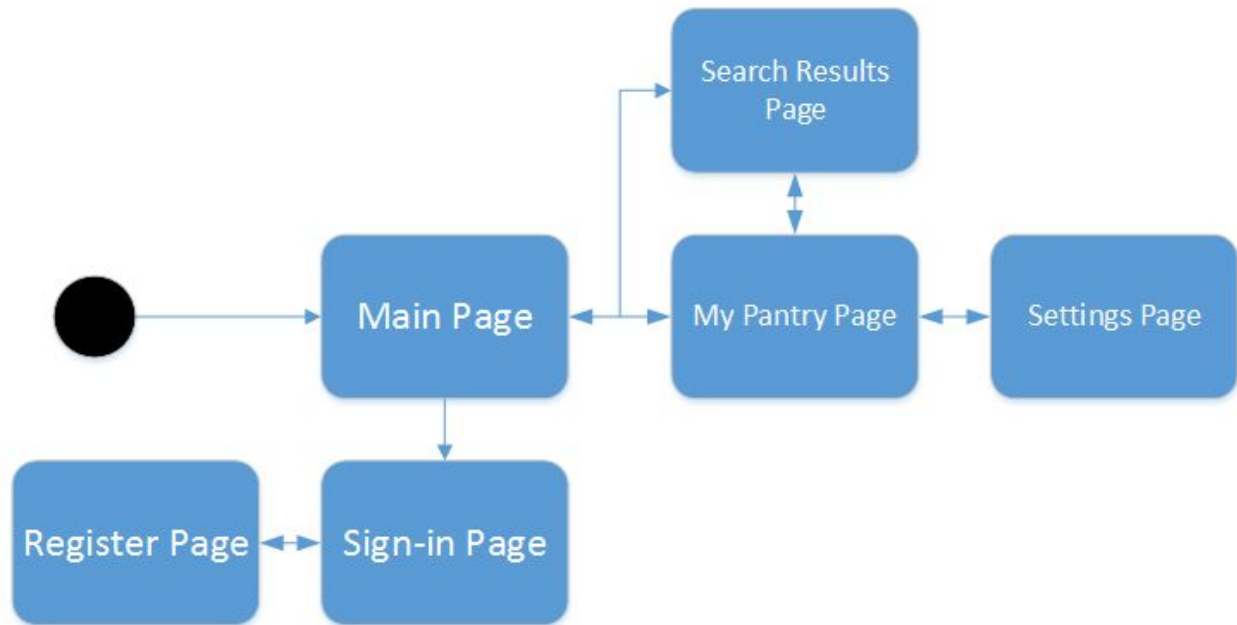
The “Ingredient” collection will contain records that represent each ingredient that the user is allowed to choose from when searching for recipes and adding foods to their pantry. It contains the following fields: the name of the ingredient.

- **Recipe**

- link: string
- name: string
- ingredientList: object (list<recipeIngredient>)
- thumbnail: string

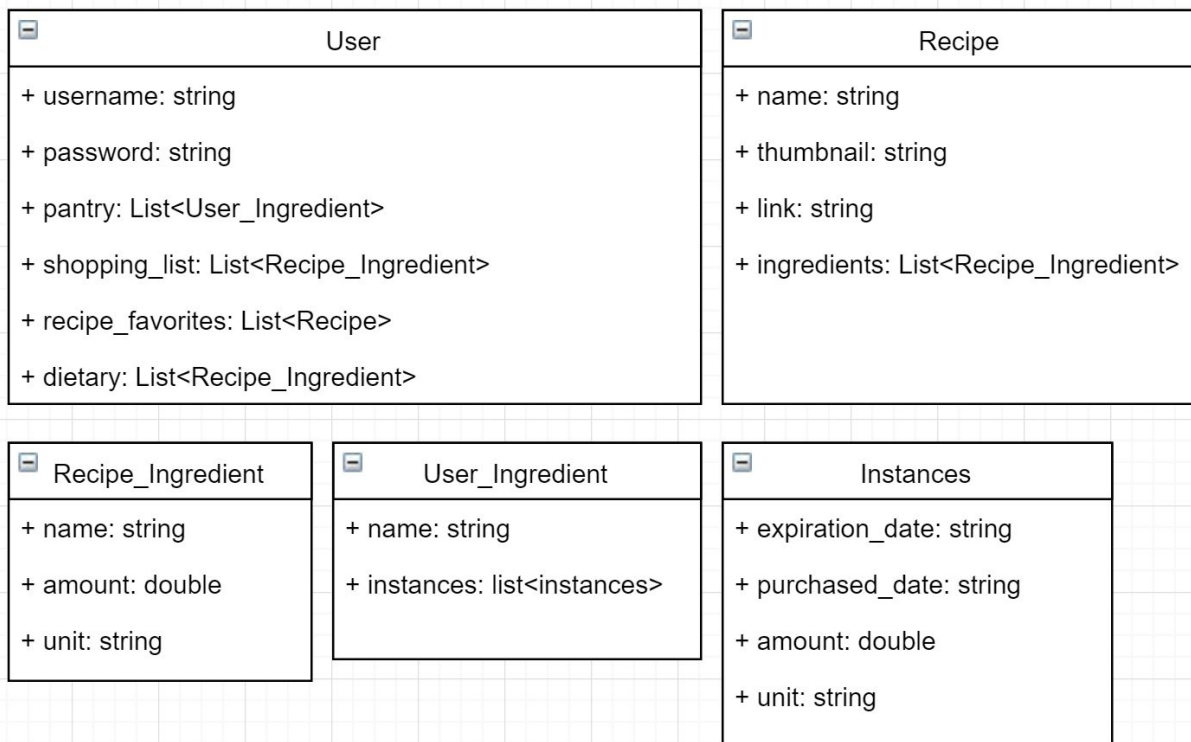
The “Recipe” collection will contain records with the following fields: a link to the website the recipe came from, the name of the recipe, a list of the Ingredients used in the recipe, and a link to a photo of the recipe to use as a thumbnail.

Navigation flow map



From the main page, the user can click on a sign in button, that will lead them to a sign in page. On the sign in page, there will be a link titled “Don’t have an account?”, which will lead to the registration page. After signing in, the user is redirected to the main page; the main page remains the same as before, except the sign in button is replaced with a link to the user’s pantry page. You can search for recipes on the main page, which redirects you to the search results page. The MyPantry page shows you the foods you already have and allows you to search for recipes at the top, which also redirects to the search result page. Finally, you can access the settings from the pantry page.

Class Diagram



Class Diagram Description

User:

- Represents a user
- Contains identifying data(Username, Password)
- Contains user's lists such as dietary preference, shopping list, "Pantry", and recipes.

Recipes:

- Represents a recipe
- Contains identifying data(Name, Icon, and a link)
- Contains a list of Recipe_ingredient.

Recipe_Ingredient

- Represents an ingredient in a recipe.
- Contains identifying information (Name, Amount, Unit)
- This class will be used to create a library of ingredients that the user can choose from. The amount and unit type is included for a possible implementation in which the website subtracts the amount of ingredients

used in a recipe from the amount of that ingredient that the user has in the pantry, if the user indicates that he or she has made the recipe.

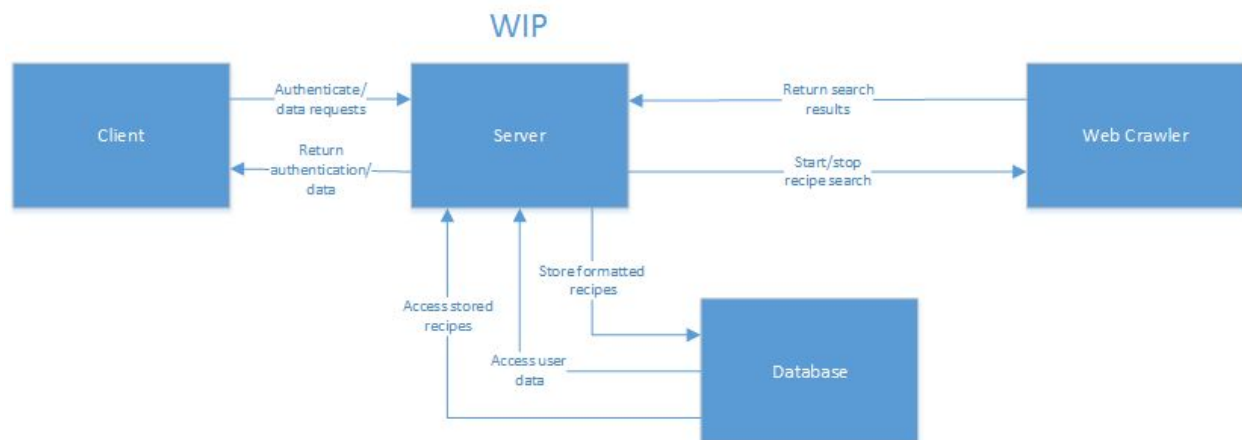
User_Ingredient:

- Represents an ingredient in user's "Pantry".
- Contains identifying information(Name)
- Contains a list of same items

Instance:

- Represents an item of one kind of User_ingredient.
- Contains Identifying information(Purchase date, Expiration date, Amount, Unit)
- Meant to keep track of different instances of food. For example, if a user buys 1 gallon of milk, and then another gallon a week later, the expirations will be different, but when the milk is considered for recipes, it is considered as single entry of 2 gallons of milk.

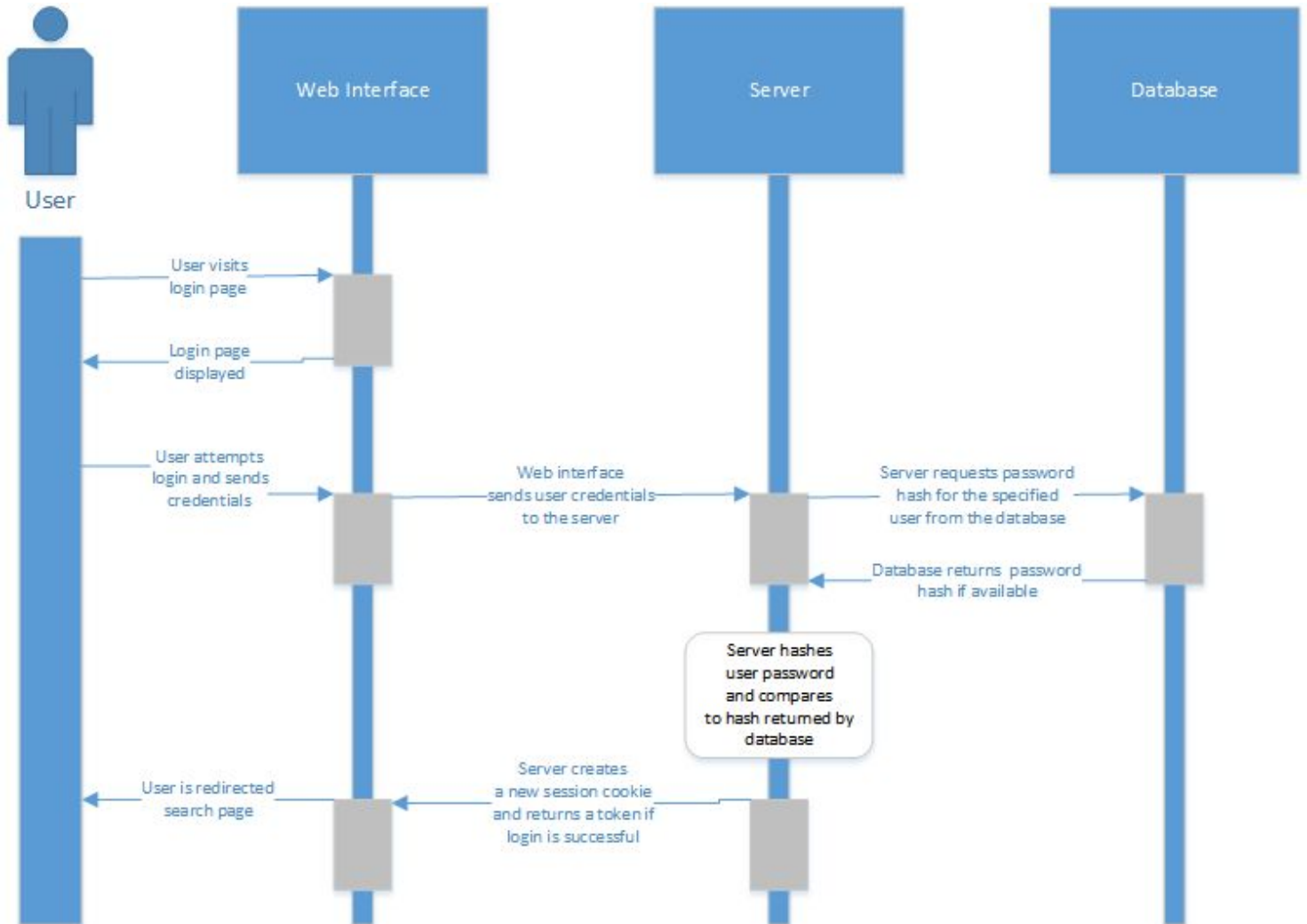
Interface/Server/Database/Crawler Interactions



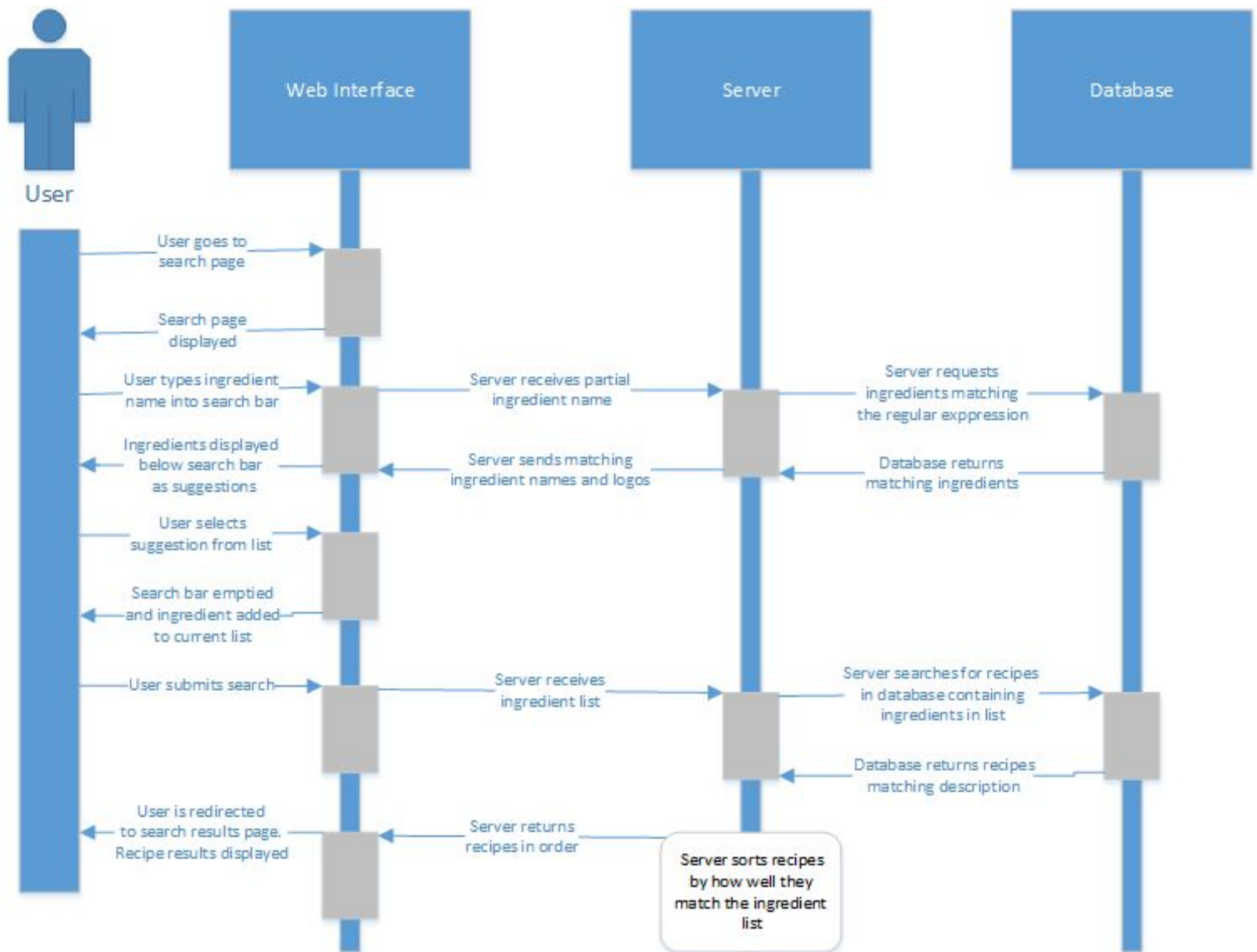
Our webcrawler will pull from select websites where the formatting is deemed acceptable for parsing. The recipes will be striped into appropriate strings according to our database and class diagram layouts. The information will then be sent to the server where these recipes are relayed to our database for storage. When a search request is made by a client, our server authenticate the request, and send the request to the database, which will return a list of recipes depending on the client input.

Use Cases

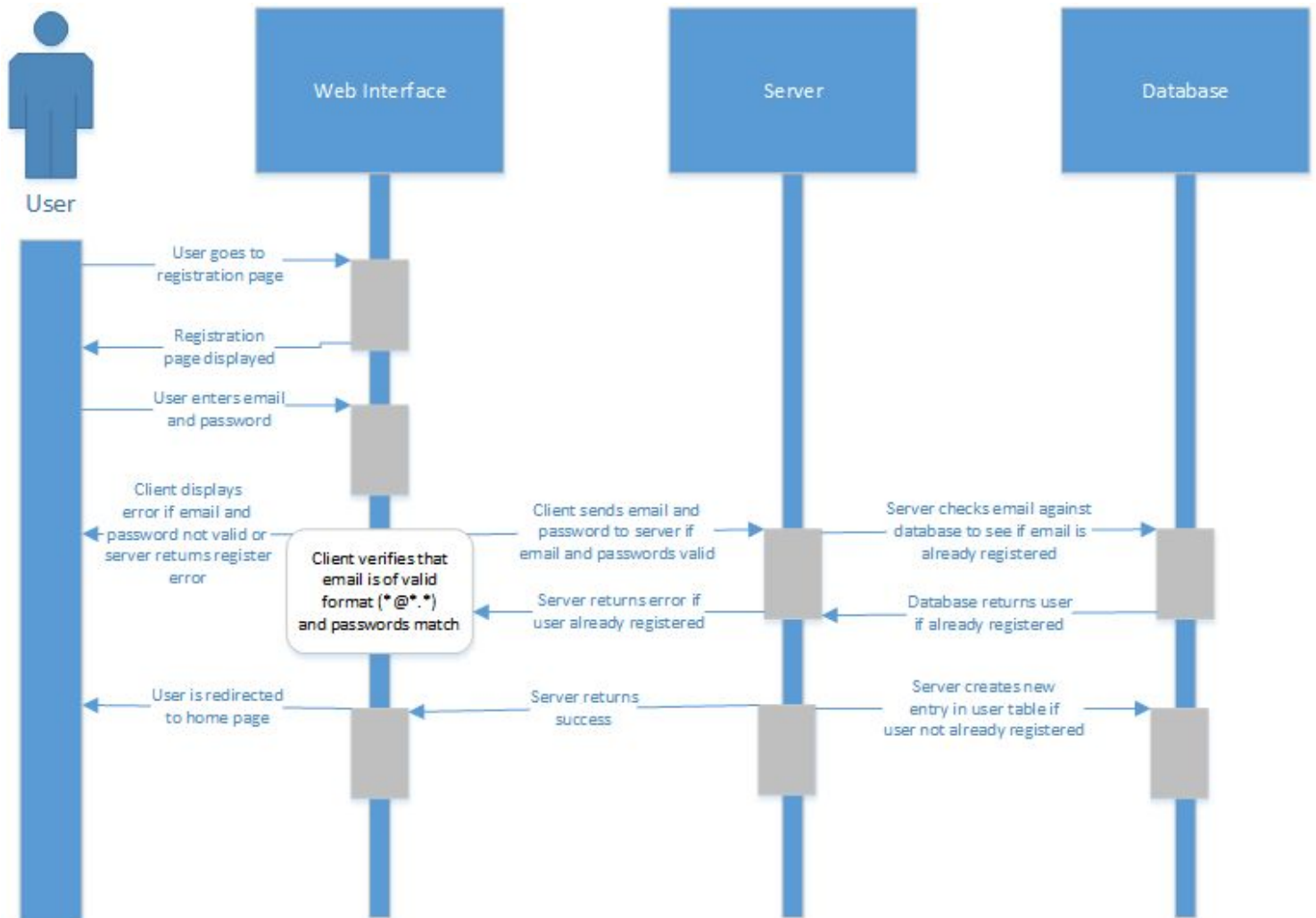
Login



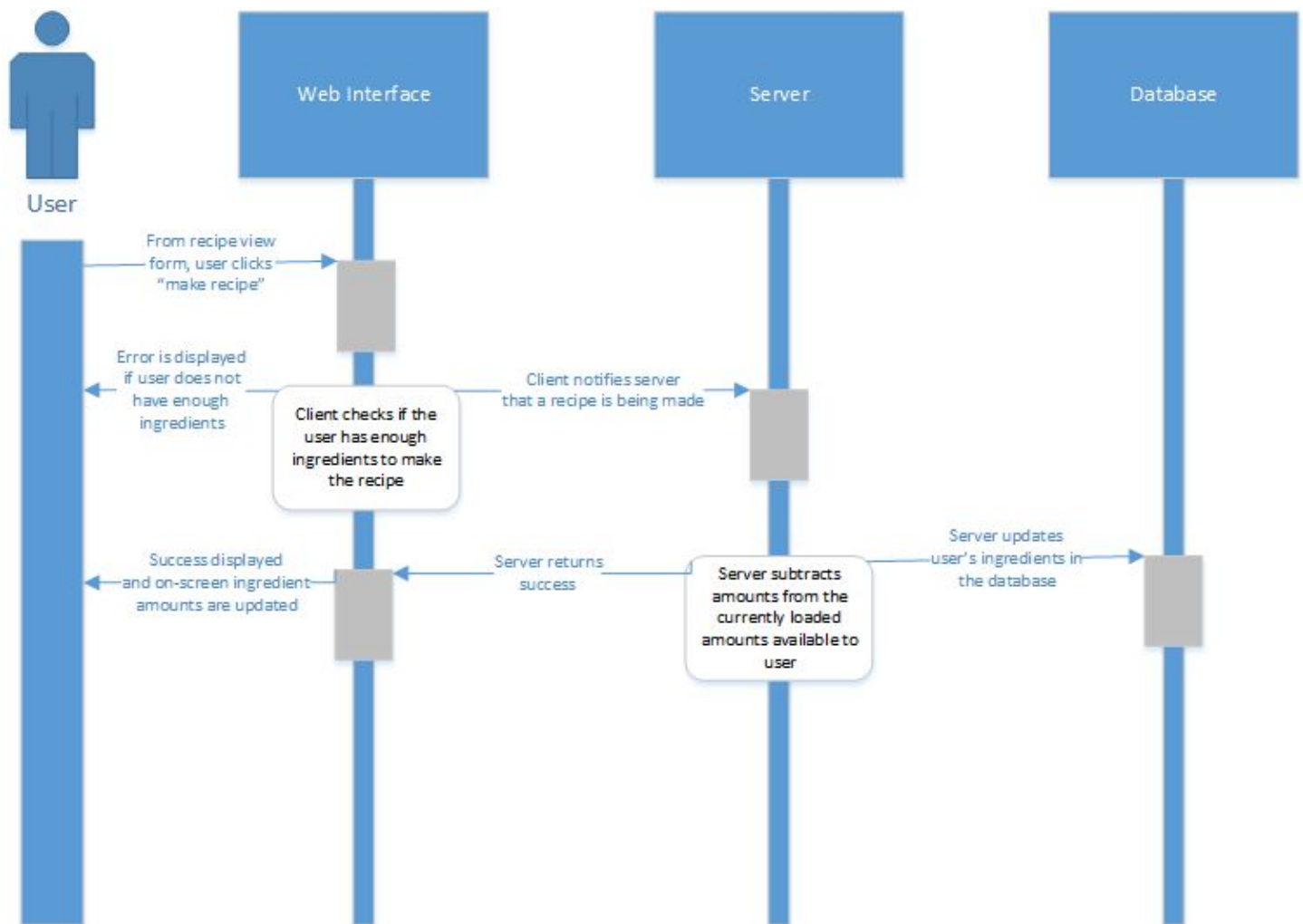
Recipe Search



Registration



Making a Recipe



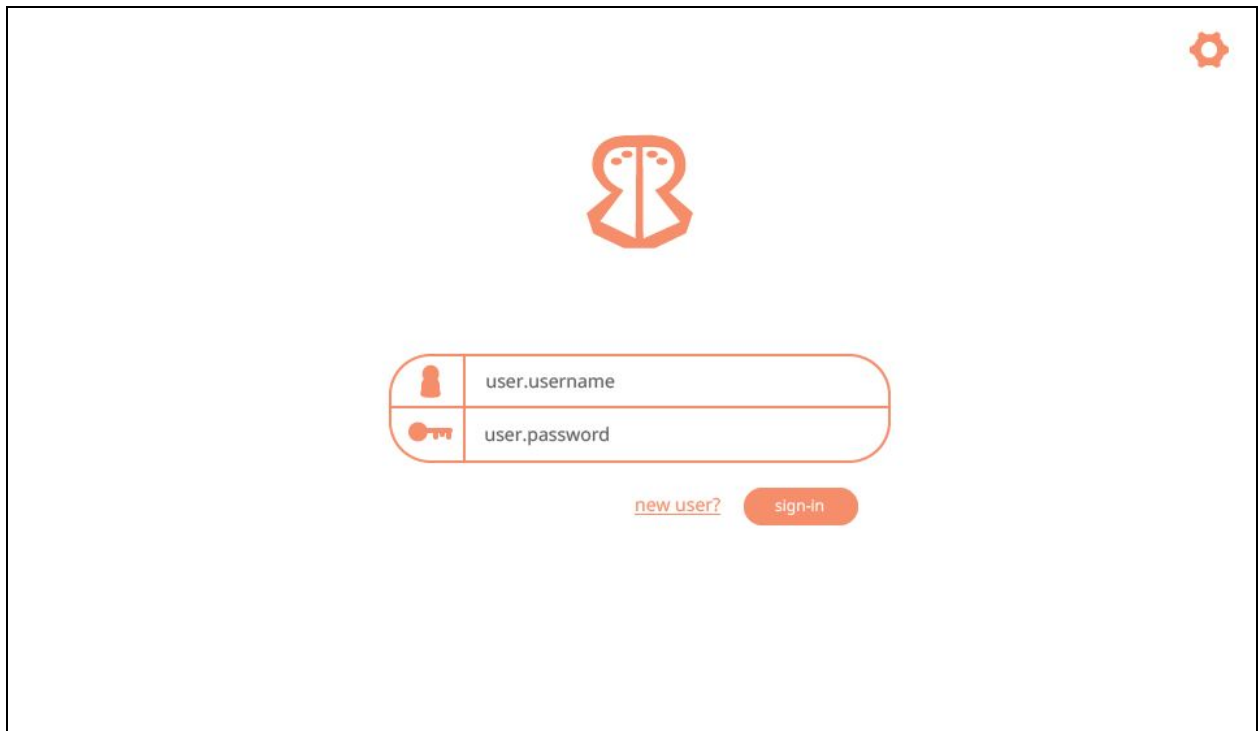
UI Mockups

Landing/Search Page - Top right will change if already logged in. Can be used with or without an account.



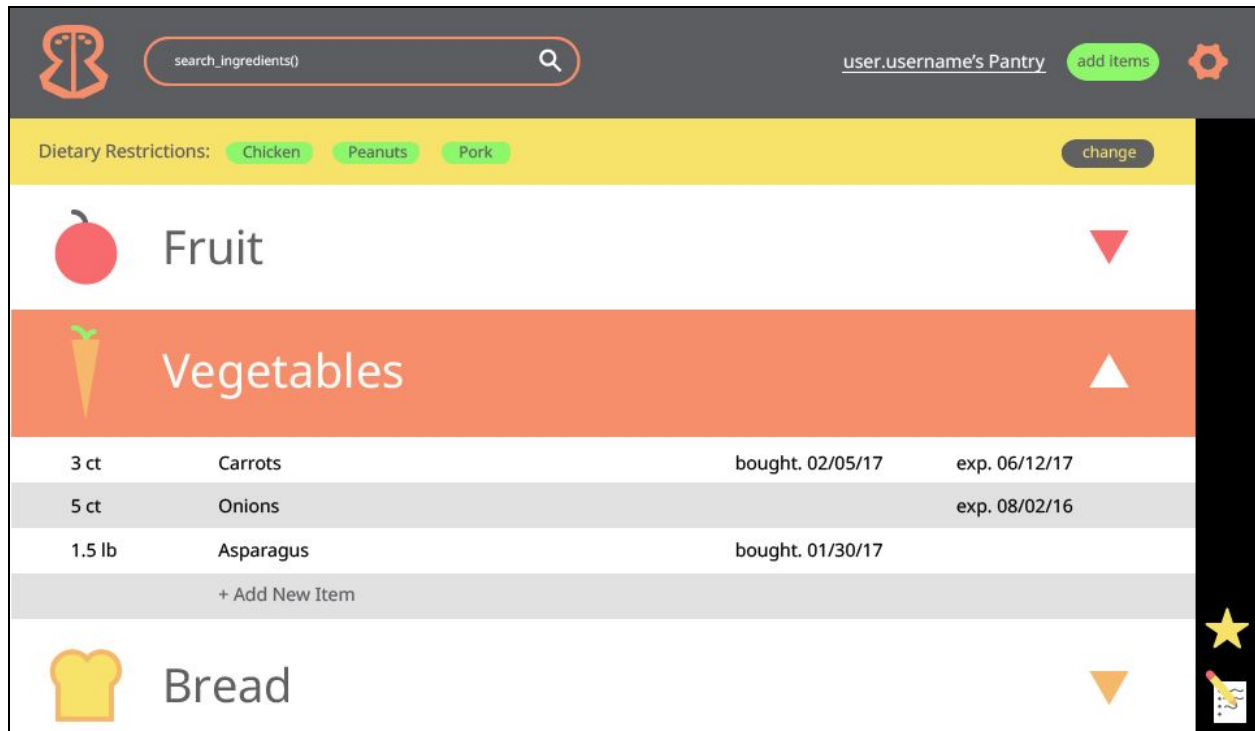
UI Mockup of the Landing/Search Page. The page features a white background with a light gray border. In the top right corner, there is a small orange button labeled "sign-in / register". Centered on the page is a large orange logo consisting of a stylized 'R' with a face. Below the logo, the text "Recipe Ready" is displayed in a large, orange, sans-serif font. Underneath the text is a search bar with a rounded orange border. Inside the search bar, the placeholder text "search_ingredients()" is visible, and a magnifying glass icon is positioned on the right side of the bar.

Login Page - Login and register will be same page to centralize all authentication tasks.



UI Mockup of the Login Page. The page features a white background with a light gray border. In the top right corner, there is a small orange gear icon. Centered on the page is a large orange logo consisting of a stylized 'R' with a face. Below the logo is a login form with a rounded orange border. The form is divided into two sections: the top section contains a user icon and the label "user.username", and the bottom section contains a key icon and the label "user.password". Below the form, there is a link labeled "new user?" and an orange button labeled "sign-in".

Pantry Page - Groups pre-defined, collapsible categories. Grey nav-bar always present. Can access saved recipes and shopping lists from sidebar.



Search Results Page - Card color is quick indication of % of ingredients owned. Clicking on card will expand card in view (as in google images). Buttons will have hover popup text describing each button for ease of user interaction.

