Octopus - Meal Recommendation system

Overview

About the system

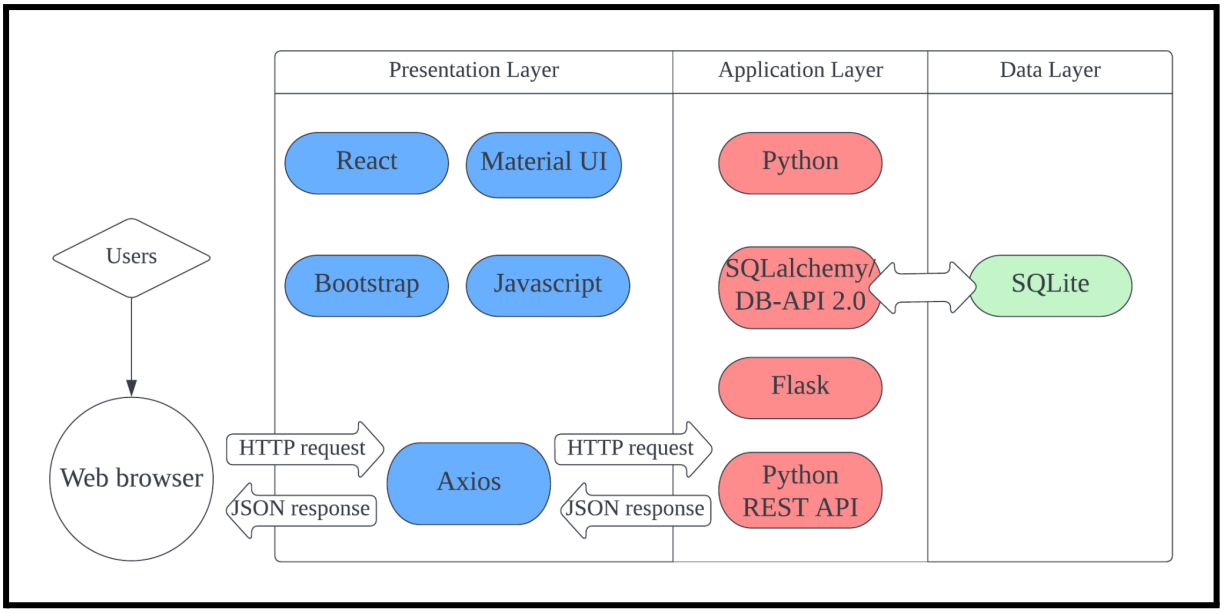
The goal of this project is to allow explorers to find others with similar interests and passions for cooking and sharing. We want users to be able to develop a community where they can share their recipes, they are proud of to allow others to enjoy their creations. In order to do that there must be a system in place to guide users toward each other. Within our site, we would allow users to post their own recipes as well as allow explorers to comment and encourage further discussion through their own recreations or by simply saying how they felt about the recipe. For each comment they can either provide their own attempt through an image and description or simply rate it if this isn't possible. This allows for a more community approach to the application rather than simply a recipe site.

To streamline this recipe is required to be organised with a title, ingredients, meal types, style, etc. This allows filtering later for searches to more accurately reflect an explorer desired dish. Our site would also have a customised feed to show users content we think they would be interested in. The primary difference between our system and currently existing ones is in how we handle user interaction as well as user retention. As mentioned previously, we want a community approach to a recipe app. This means that user feedback is the most important regarding ranking. So, we used these metrics to design our novel methods such as subscribing to independent contributors as well as allowing user rating on comments to quickly gauge the viability of a recipe and to provide feedback to contributors.

Our project can be somewhat summarised as a hybrid of instagram and a typical food recipe website. The reason we did this was ease of access and cleaner UI features. We also want our project to have friction and discussion within the community like what you would typically see in an instagram post, which is accomplished through the feedback and comment system.

System Architecture

For our Meal Recommendation system, we divide the system structure into three parts: view layer, Business logic layer, and Data layer, as shown in the following figure.



System Architecture diagram

The view layer (user interface, UI): mainly completes the UI interaction function of the recipe system. A good UI can improve the user experience and enhance the user's comfort when using the food forum system. UI interface design should also adapt to different versions of recipe systems and resolutions of different sizes to achieve good compatibility. The UI interaction function requirements are reasonable, and users must get the corresponding interaction results when conducting interactive operations, which requires the presentation layer to be well connected with the business logic layer.

Business logic layer: mainly completes the data processing function of the recipe system. The data transmitted from the presentation layer is processed by the business logic layer and delivered to the data layer. The data read by the system from the data layer is processed by the business logic layer and delivered to the presentation layer.

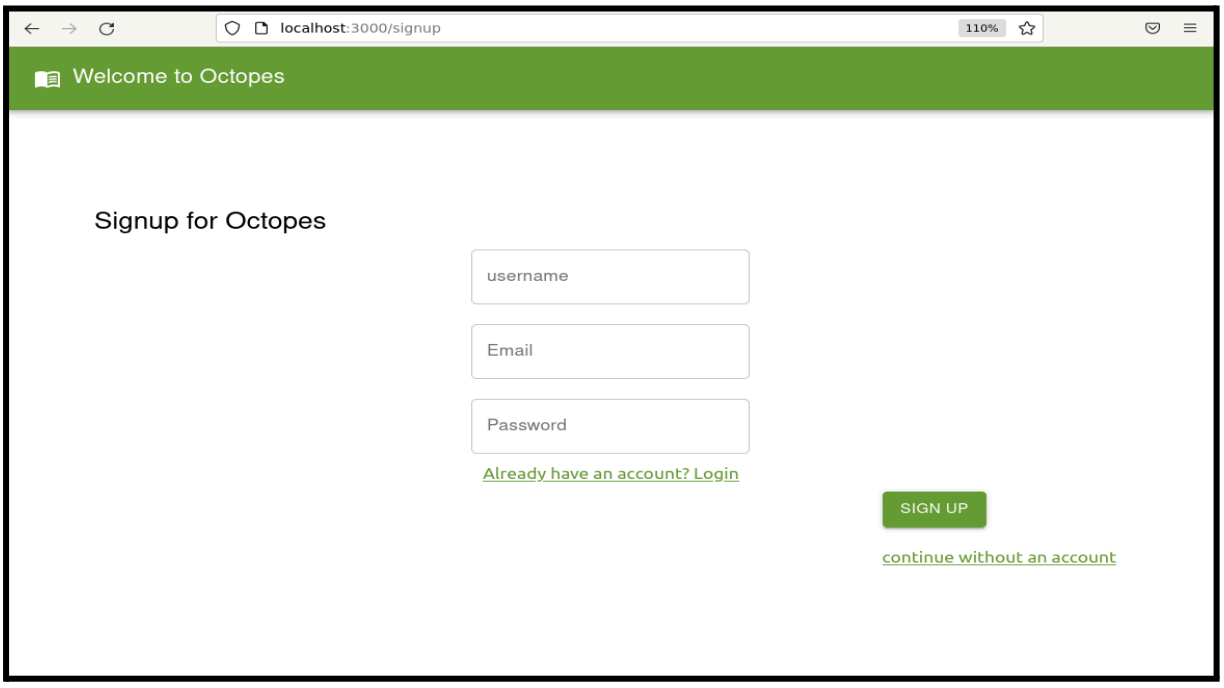
Data layer: Since the data of the recipe system is placed in the SQLite database on the server side, the part that belongs to the service layer can be directly integrated into the business logic layer, so there is only a database in the data layer, which mainly completes the data storage and management functions of the recipe system.

Project Objectives

Objective 1 - Authentication

According to the user's story, users need to have their own account to log in to the Meal Recommendation system, so that they can upload their own recipes or subscribe to others. Therefore, the main requirement for the authentication part is to register an account, log in and log out of the account.
If the user does not have an account in the system, he/she needs to register. Email, username, and password are required. If the user's name or email address provided already exists in the system, the system will prompt that the user's name or email address is not allowed.

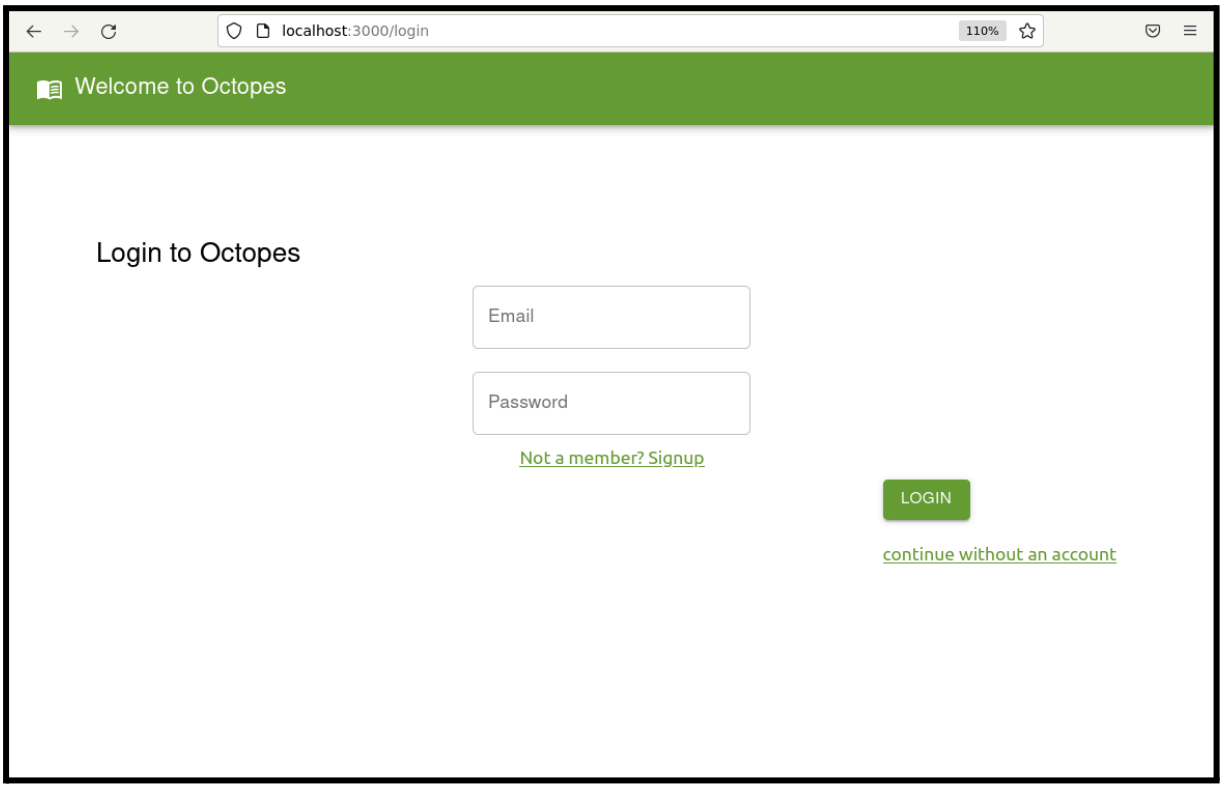
Repeated username or incorrect password



Create account page

If the user has already registered, he can log in with the account that has been successfully registered in the system. If a user enters an unmatched account or password, he will be prompted that the account error does not allow you to log in. And we choose not to hide the password in the black dot when the user enters the password in order to make the user experience better in the

process of entering the password. The password in the black dot is outdated. In the past, people shared computers, but now no one will sit next to you and stare at your screen. It is also more user friendly, reducing the chance of wrong input, and users can directly see their operating results. This effectively reduces unnecessary trouble and prevents our users from getting upset due to continuous mistakes in entering passwords.



Sign in page

Users can log out of their accounts at any time after logging in. Or it will automatically expire to protect account security.

After logging in, users can go to their own profile page. Your recipe username and email will be displayed on the user's profile page. Users can upload or edit their own information there, such as changing passwords, usernames, or email.

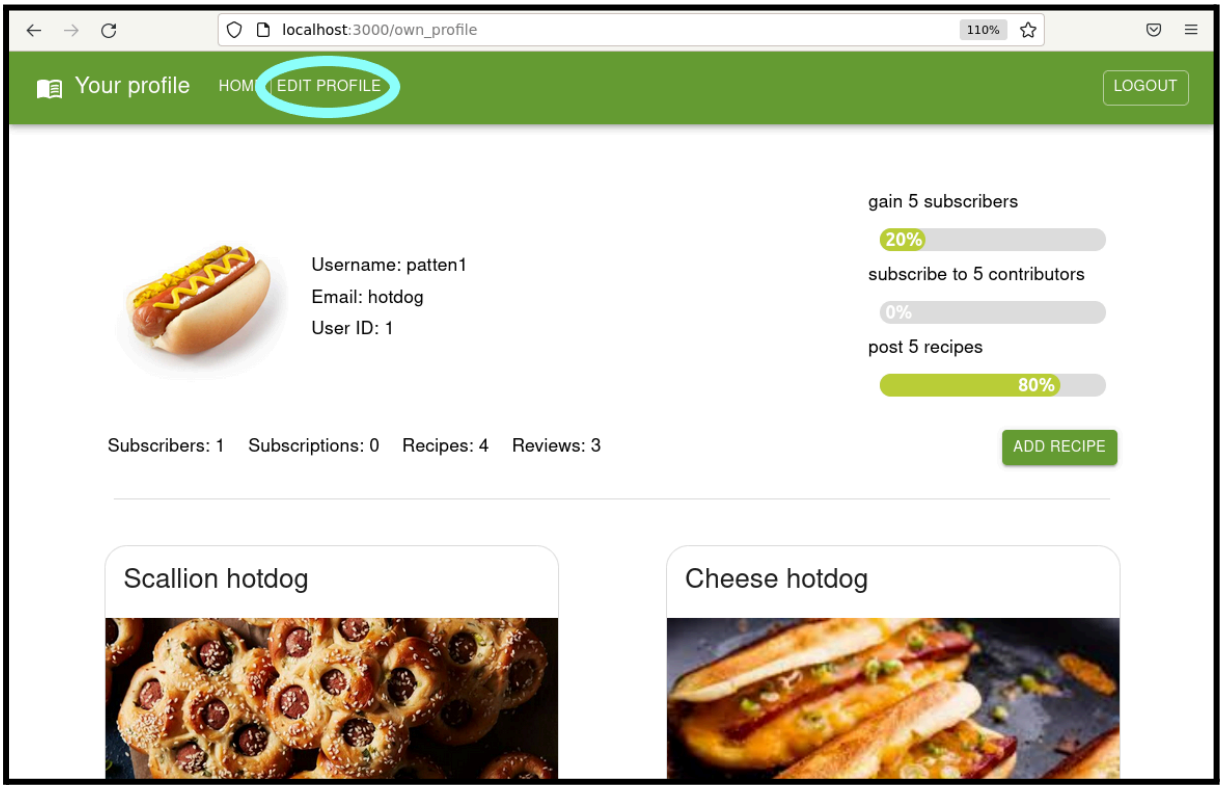
Of course, users can access recipes without logging in. However, if you want to subscribe to someone without logging in, the system will redirect the user to the login page.

Objective 2 - Maintain profile

After a user creates an account or login to an account, they now have unlocked the full functionalities provided by the application and have full access to the entire system. The system stored the users' data such as the username, email, users' recipes, user id and password which can be viewed and managed in their user profile.

Users are able to access their user profile from the homepage "profile" button where the "login" button was previously located.

Inside the profile page, they can view their profile details which contains not only the users' authentication details but also additional information displayed that helps users to keep track of their subscribers, subscriptions, recipes and reviews.

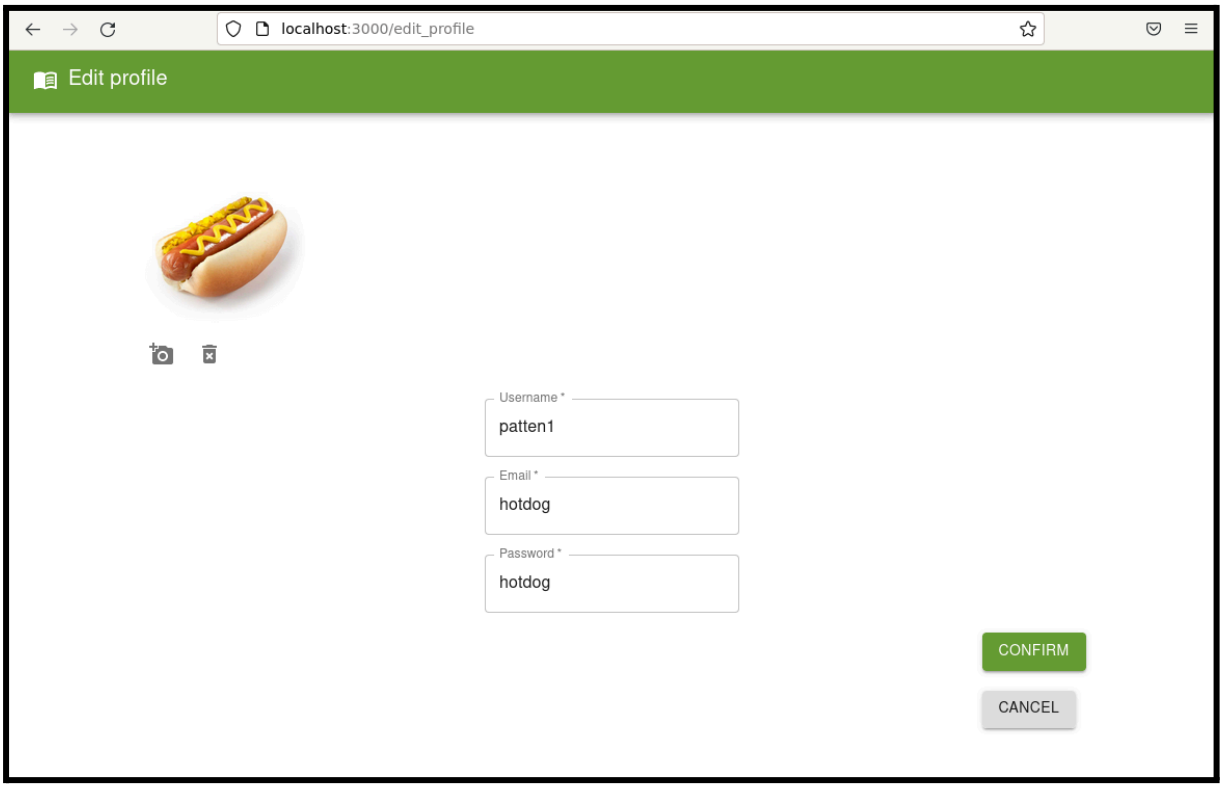


User's own profile page

Inside the profile page, users can navigate to the edit profile page by clicking on the "edit profile" button on the navigation bar. Inside the edit profile page there is a set of prefilled editable fields whereby the users can edit their username, email and password, the system then follows a procedure of

checking similar to the one that is used in the registration process; to ensure both the username and email are not already been taken by other users.

On the same page, there is an avatar component that either displays the default user icon or the users' profile picture. Users can optionally upload/ replace a profile picture by clicking on the add photo button or remove one by clicking on the garbage bin button if they have a profile picture.



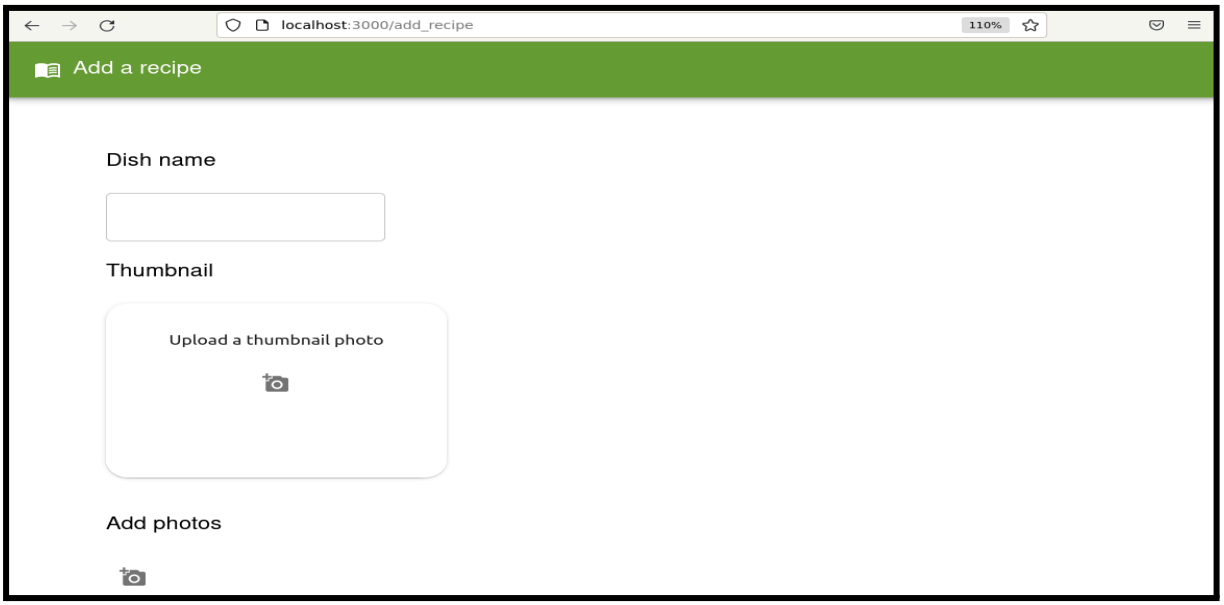
Edit profile page

Users are able to look at other people's profiles by clicking on their profile picture in the recipe page owner's details or comment section. Inside other people's profiles, they will not have the option to edit the profile or add a recipe, instead, you can subscribe to them by clicking on the subscribe button.

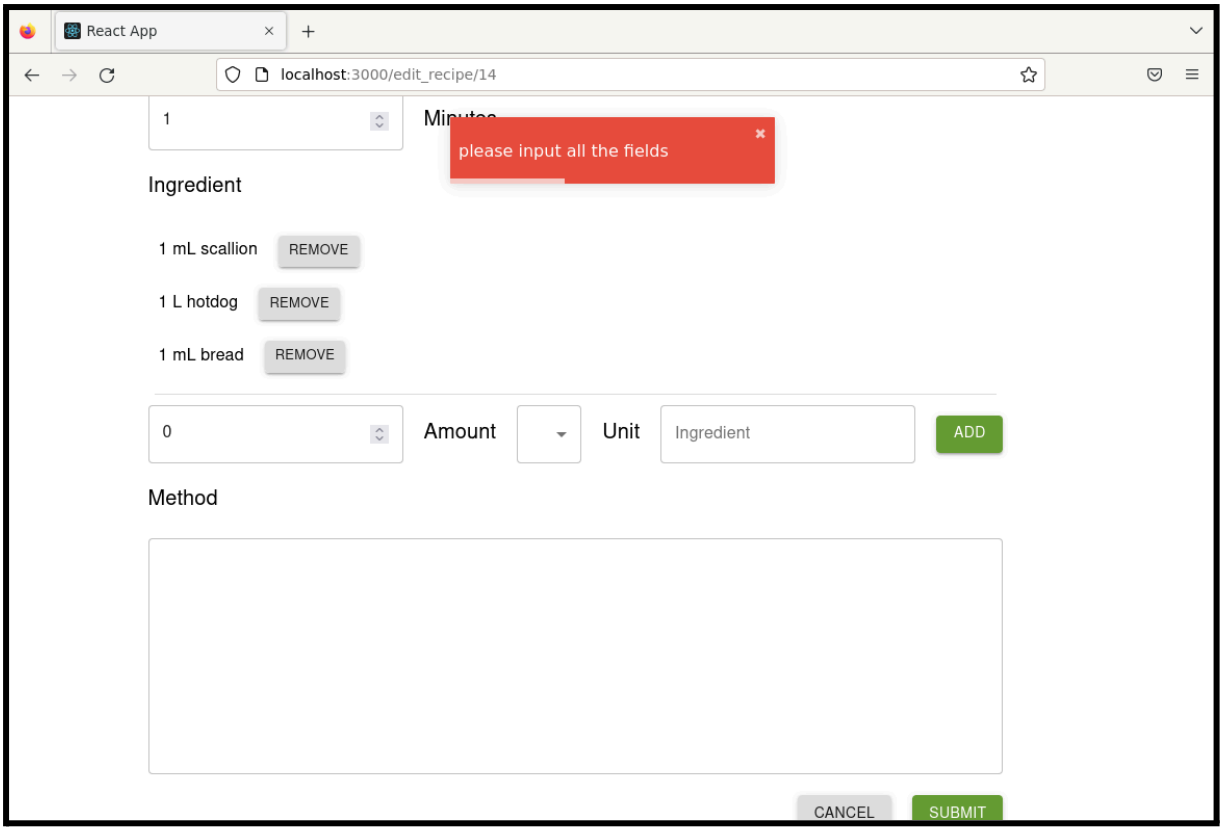
Objective 3 - Create and maintain recipe

Once logged in to the system, users can make contributions by posting recipes to the system. This type of user is called a “contributor”. Contributors can manage their recipes by editing and deleting them. To successfully post a recipe, users are required to provide compulsory information including recipe name, image, ingredients, meal types,cooking method, thumbnail and recipe photos. If users try to submit a recipe without filling in all the required information, they will be prompt to input all the fields.

Own profile add recipe

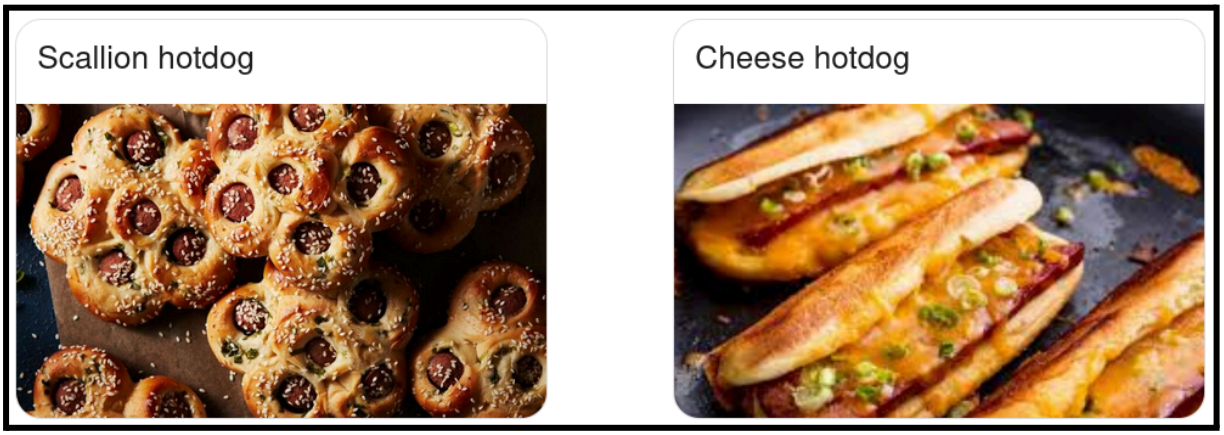


Add recipe



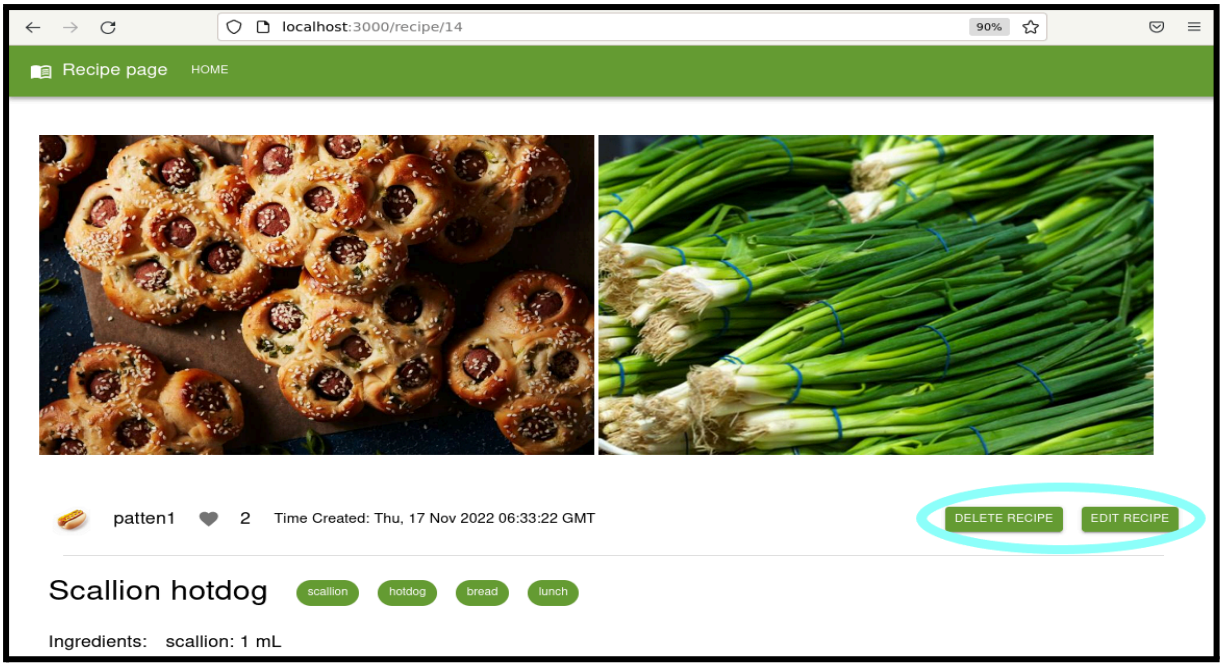
Error prompt of missing field

After the recipe is successfully posted, the recipe can be found in the homepage news feed/ search result and in the contributor's profile page in the form of recipe summaries, which is a simple card that contains the recipe thumbnail, and the recipe name to draw users' attention with loading them with too much information.



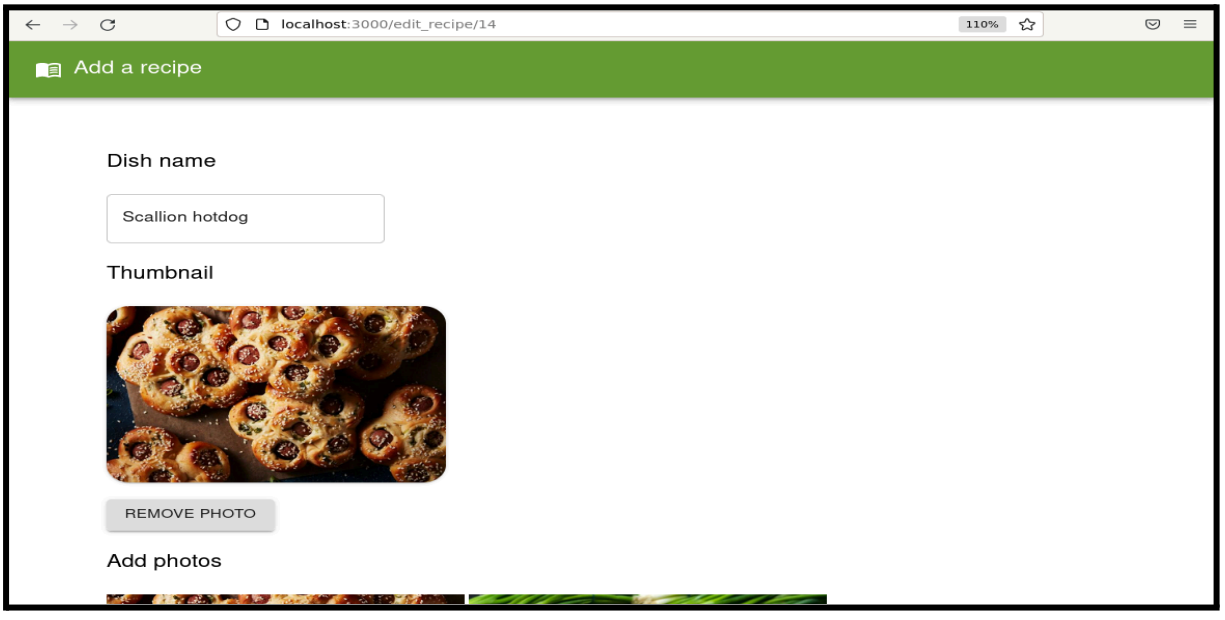
Recipe summaries

The recipe page contains all the information about the recipe, including recipe pictures, owner's username, recipe name, recipe tags, ingredients, meal type, tutorial, comment section and recommendation recipes.



Sample recipes for editing and deleting

Users can edit or delete their recipes through clicking on the edit / delete buttons in their recipe page. The UI of the edit recipe page is exactly the same as the add recipe page, except all the existing information of the recipe is prefilled, users can change all their recipe details except the recipe pictures as changing it would mean the existing likes and comments does not accurately reflect the edited recipe. It also prevents users from exploiting and taking advantage of this freedom to change a recipe to a completely different recipe while keeping the scores and rating preserved. The criteria to successfully edit a recipe would be exactly the same as adding a recipe. After editing, the likes, reviews and comments on the recipe are preserved.



Objective 4 - Search recipe

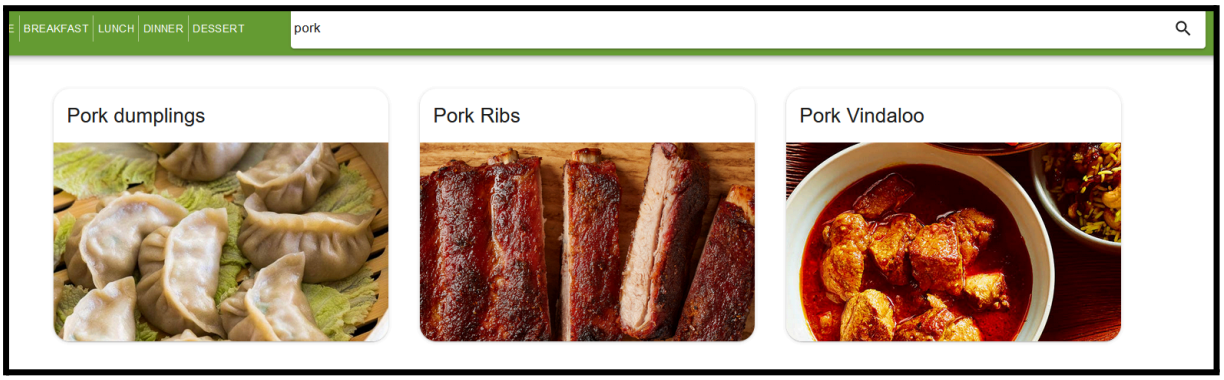
Default search bar on visiting the site

The search system allows users to accurately find the things they are looking for. It also accepts partial string matching for ingredients matching. For example, while looking for mush (short for mushroom), will return all recipes where either the title or ingredients contain the substring mush, this returns all mushroom recipes and any other ingredients that may have the substring mush.

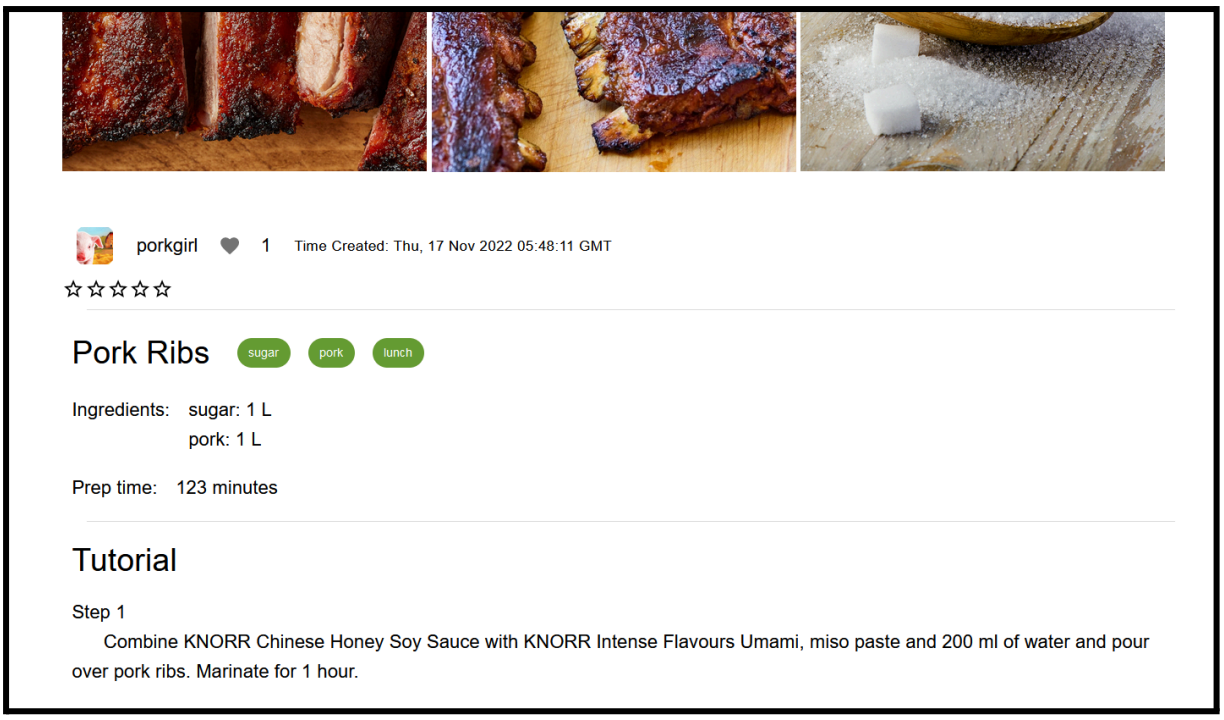
On the top left are the four main categories of food, breakfast, lunch, dinner, and dessert. These can be manually clicked on or typed into the search bar to filter strictly by mealtypes holding these tags. Every recipe is required to have one of these meal-type tags therefore every recipe can be located with the four buttons on the top left.

Within the backend we split all strings by spaces to create auto generated tags. This method is applied to the title and ingredients. This allows for us to find partial matched strings for either the title or ingredients.

One known limitation is double tags which we have addressed in the backend. When a recipe title includes a meal-type as part of its title say “breakfast sandwich”, it would end up with a double tag inflating its ranking for sorting in recommendations and personalised news feed. To combat this we implemented a check to ensure tags only get counted once. More on this in Objective 6 and 7.



Searching for pork yields the above recipes



Sample recipe page by porkgirl

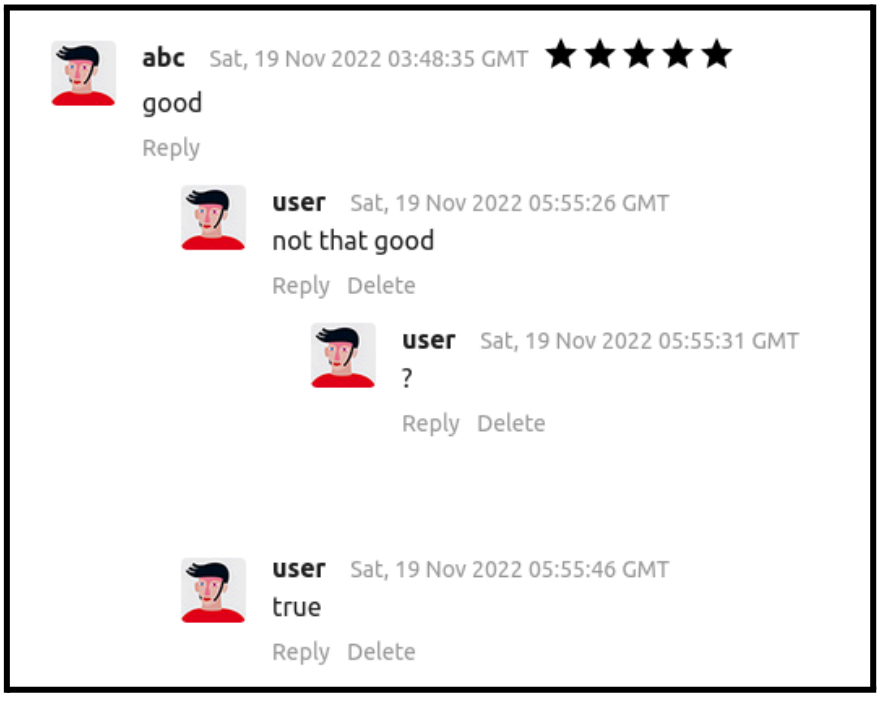
The relative user stories to this function have all been achieved. We wanted to be able to allow users to first find ingredients they may be interested in, which they can access by clicking the picture, which then shows more details about the recipe and the author. The primary function of this section is being able to sort by the front-end tags of meal types for organisation and ease of access which we have accomplished.

Objective 5 - Feedback

The feedback system allows users to express their opinion freely within the system. Explorers can like or comment on recipes under the comment section to share their thoughts on recipes. Likewise, contributors can comment on review posts to share their thoughts on the explorer's cooking.
Users can like a recipe by clicking on the heart icon, the heart icon then turns red to indicate the liked status. Similarly, users can unlike by clicking on the red heart icon and the heart icon will return to normal status.

Like a recipe

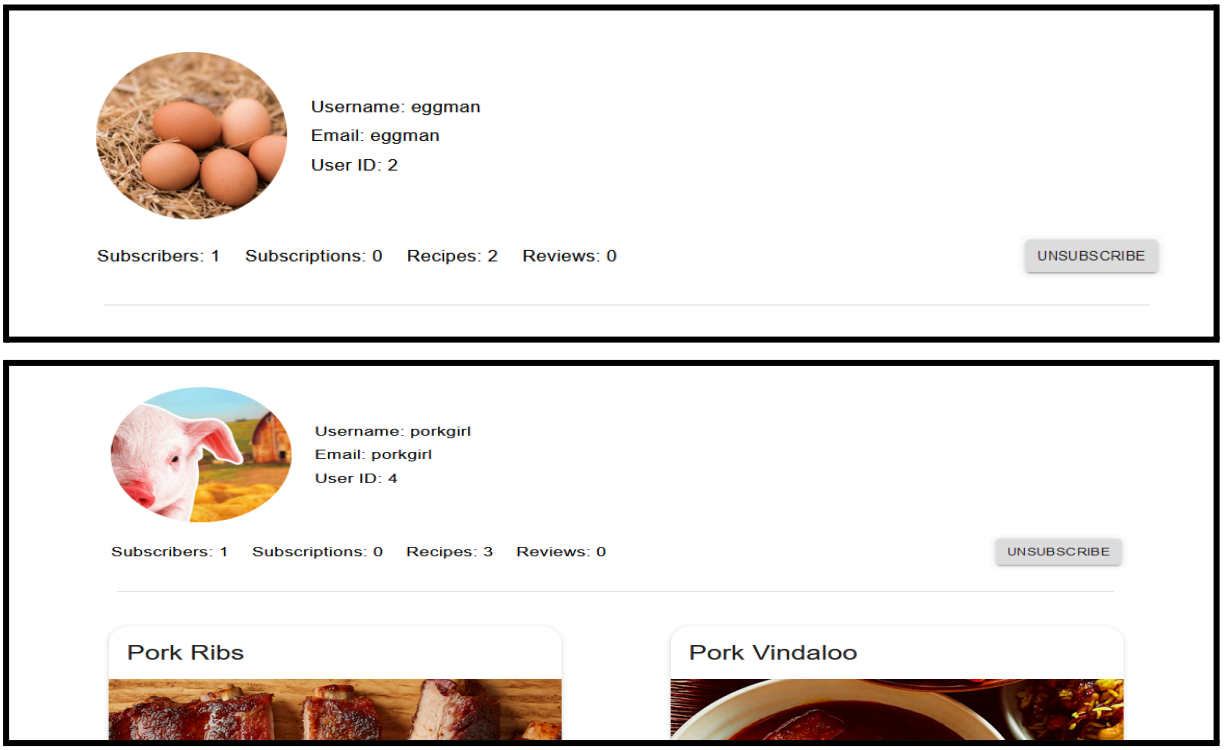
For the implementation of likes, the like relation is stored in a separate table in the database. This makes the like operations trackable and able to undo.
Users can comment on posts by inputting into the text box in the comment section, a comment composed of the message, the profile picture, the username of the user, and the time it is commented. Also, users can reply to any replies to continue discussion.



Example of comment structure

The comment part is implemented by a tree structure. A top comment (comments that directly replies to the recipe) is the root of a comment tree and all replies under this comment are nodes of the tree.

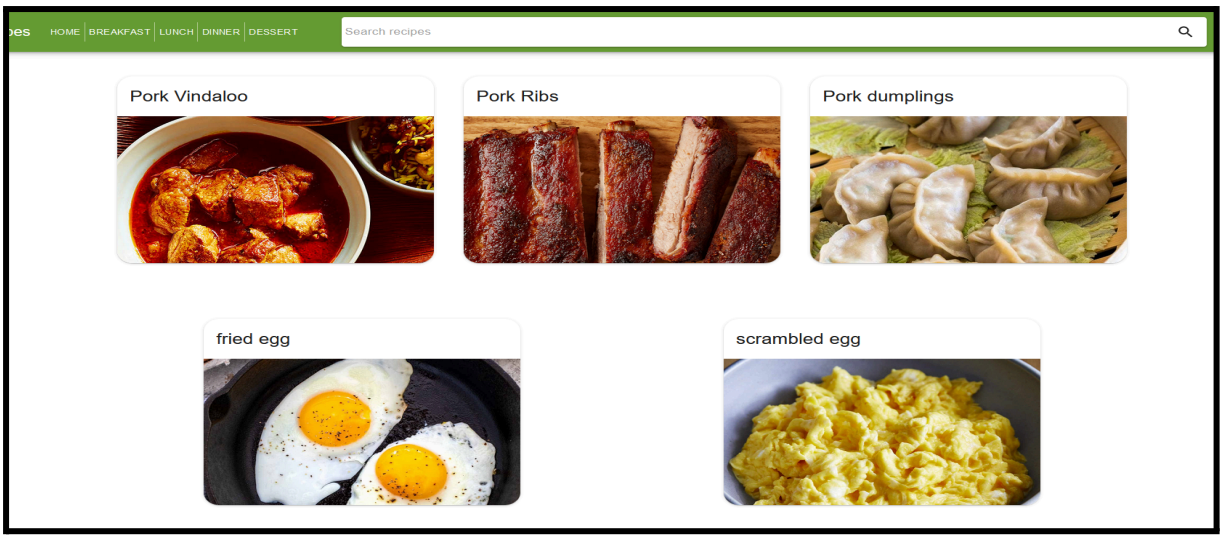
Objective 6 - Subscription/ newsfeed



Users are able to find subscribe to other contributors through clicking into any recipe and clicking on the contributors profile image or name. This redirects them to their contributor page where they can hit the subscribe button on the left. In the picture above the user is already subscribed to the contributor therefore if they click the button again they will be unsubscribed.

In the below example, the user is subscribed to two users, eggman and porkgirl. This means when they click on the home button on the top-left. They will only be shown their personalised news feed consisting of recipes only relating to subscribed contributors. The ranking of the recipes shown here is determined by the amount of times the user has liked recipes by the contributor. Then it is sorted by time-created to create the ranked list as mentioned in Objective 4.

Currently the user has liked the recipes pork ribs, pork dumplings, and fried egg. This puts all of the pork girl's recipes above eggman because the user has liked more of their recipes. Pork vindaloo is the most recently created recipe, followed by ribs and pork dumplings. It is worth noting that time created is a secondary criteria and that the number of times a user has liked a contributor's recipe is the primary factor. For time we treat each hour as the smallest denominator since minute differences and seconds are irrelevant.

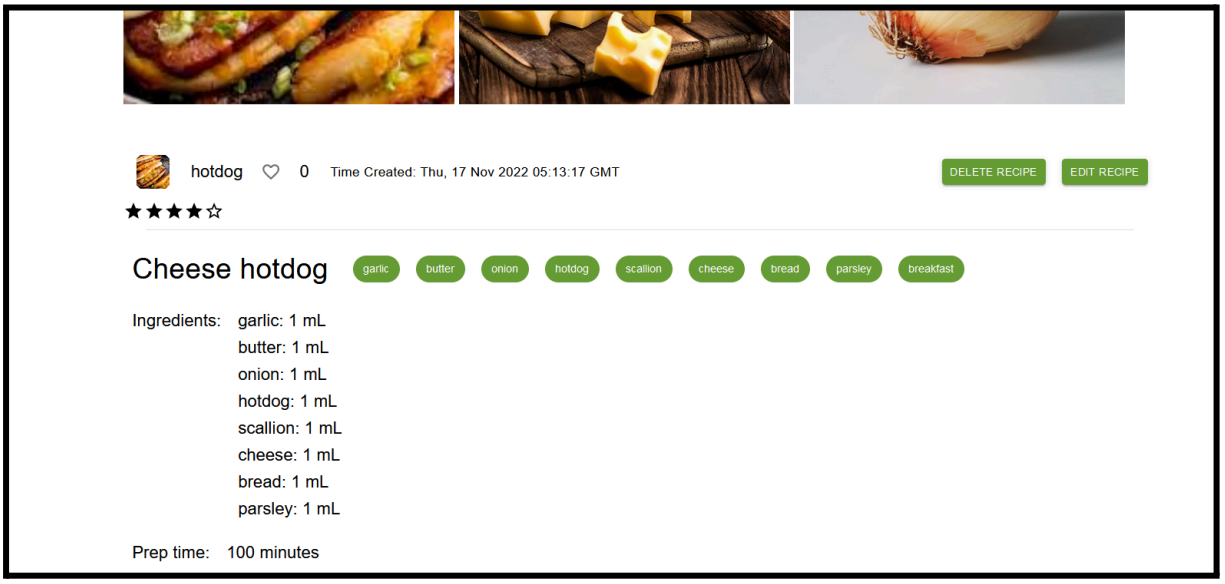


Sample personalised news feed

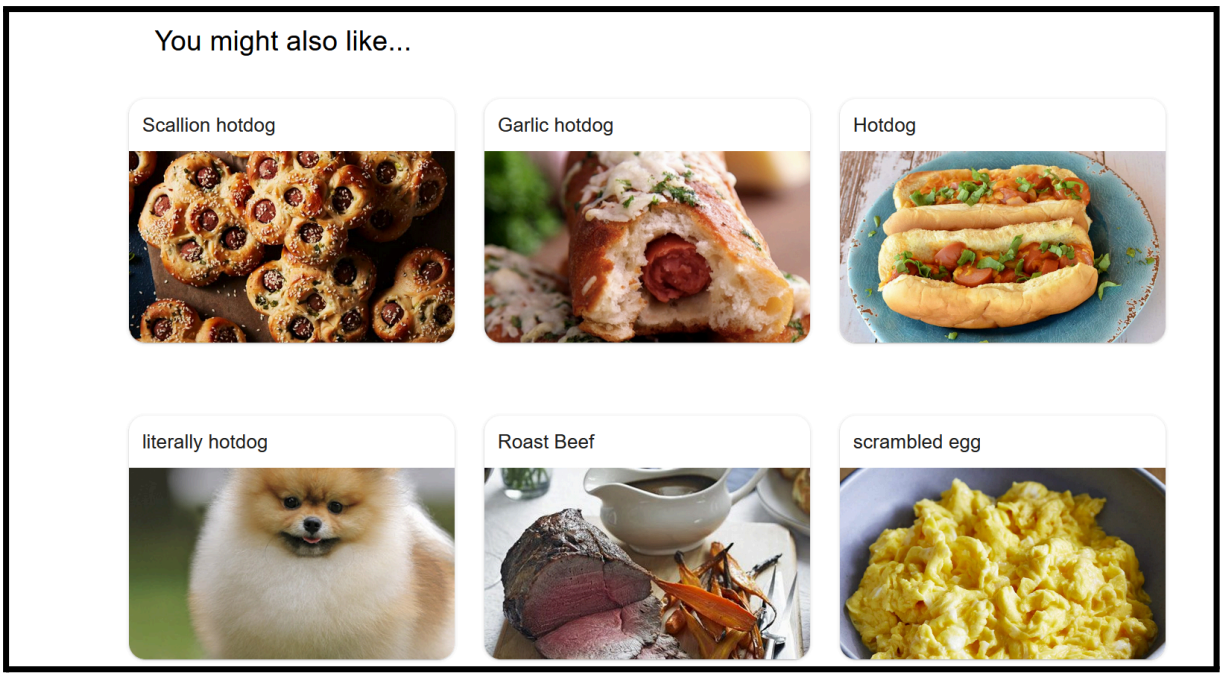
Our primary objective here was to create a relevant news feed for each user's interests which we have determined based on their subscribers and likes. We believe this system is currently good because it allows a user to always find things they are interested in. If a user did not like a particular recipe they would not have subscribed to the contributor. Therefore if they did subscribe it means they are interested in their content making it a good match for them.

One future implementation is being able to take ingredients from these recipes and include more to create a more varied user experience allowing them to find more creators and more recipes.

Objective 7 - Recommandation



Sample ingredients list



Sample recommendation of recipes near the bottom of recipes

The recommendation system is designed to recommend similar recipes below every single recipe once the user has clicked into it as shown in the second image. The ranking system here is based on two factors. First, how many ingredients in the parent recipes match the recommended recipes' ingredients. Second is how recently they were created, here the time factor is chosen to be a minute since there are more recipes compared to the newsfeed which is filtered by subscriptions only.

In the example above the parent recipe is the cheese dog. Since every hotdog has bread and hotdog in it we can see how they are ordered given their titles. Scallion hotdog has an additional scallion ingredient and garlic hotdog has an additional garlic ingredient.

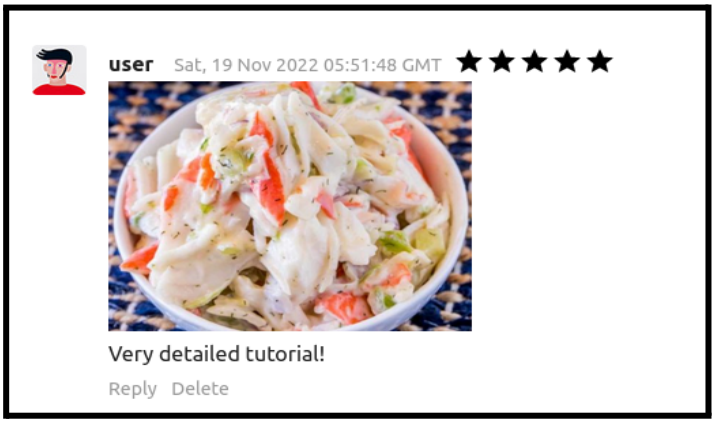
Therefore since parent has all of those ingredients the closest matches are scallion and garlic hotdog since they have an additional match. This is followed by hotdog and literally hotdog where the latter only has hotdog as an ingredient.

The reason roast beef and scrambled eggs are also recommended is because they each share one ingredient with the parent recipe, parsley. While “literally hotdog”, roast beef, and scrambled eggs each share one ingredient, literally hotdog was made the most recently which is why it is sorted first.

The objectives here were to recommend only related recipes, as well as ranking them in a logical order which we have accomplished through this dual ranking of ingredient partial matching and time sorting.

Objective 8 - Review

Review post is a novel functionality that allows explorers to post a review in the form of comments of recipes that they have used. A review post must include a rating out of five stars the user gives to the recipe, a comment paragraph allowing the user to share their thoughts on the recipe and option to include a picture of their cooking.



A review with picture

The system then generates an overall score (rating out of five stars) for the recipe based on these review posts and displays on the recipe page.

In other similar recipe recommendation systems such as Bestrecipes (Australia's Best Recipes, n.d.), they fail to provide a platform for explorers the opportunity to receive any feedback on their cooking since the only way for them to interact with the system is commenting on recipes; the system only taught its user on how to make food, but not how to make good food.

Through this functionality, explorers can finally demonstrate their cooking skills and can improve by taking constructive feedback from other users such as the owner of the recipe.

This novel function focuses on maximising the explorer’s benefit from the system, Furthermore, the objective benefits contributors by having a more sophisticated feedback from the review post, which are made mostly by the users who have tried to follow the recipe and make their own dish, therefore, unlike the conventional feedback system (liking), db it is not only based on how good the recipe photo looks. This makes the rating of the recipe more accurate, which allows the contributor of the recipe to make improvements from it, and produce more quality recipes.

Objective 9 - Reward

Reward system aims to encourage users to participate actively in the meal recommendation system. It provides incentives for explorers/contributors to post more review posts/ recipes through giving them rewards. Users will be awarded with a corresponding title when they complete a certain mission which can be found in their user profile.

Currently, there are three mission types including: posting 5 recipes, subscribing to 5 contributors, and gaining 5 subscribers. Each mission corresponds to different titles such as “contributors”, “explorer” and “influencer”. The reward system is coded to be extendable and can potentially store a large amount of missions and corresponding reward titles so that harder missions can rotate in when the old missions are finished.

The reward functionality can be used to excite users to do all sorts of activities that help the website grow. The reward system is specifically designed to stand out amongst the large number of competitors such as Bestrecipes (Australia's Best Recipes, n.d.) and thousands of meal recommendation systems on the internet, the reward system increases user adhesion by appreciating the users and give them a sense of accomplishment.

Reward system

Technologies

Presentation Layer

- **Javascript**
  Javascript is the program language that will be used in the frontend. JavaScript is a cross-platform and object-oriented scripting language used to create webpages. It also supports many libraries and frameworks to assist web development. One bonus is in user-created libraries where common features do not have to be reimplemented as we can just use the libraries given.

- **React**
  React is a declarative, efficient, and flexible JavaScript library for building user interfaces. A website consists of many components, which may have complicated structures. React helps to collaborate on these components to build a complete user interface. React is fast as the webpage is built with small components and when the page changes, the web browser simply retrieves the components that are already stored locally, instead of sending a request everytime the page changes. React is licensed under the MIT licensing which allows for commercial use, modification, distribution and private use, therefore, it has no impact on the project.

- **Material UI**
  Material UI is a react library which provides ready-to-use user interface elements including text fields, checkbox, buttons, radio buttons, drop down menus, etc. Material UI is licensed under the MIT licensing which allows for commercial use, modification, distribution and private use, therefore, it has no impact on the project. (“MUI Core,” 2022)

- **Semantic UI**
  Semantic UI is a frontend development library that provides modern complex website components such as forms, comments, buttons..etc to use without coding nested components. Semantic UI is licensed under the MIT licensing which allows for commercial use, modification, distribution and private use, therefore, it has no impact on the project.(“Semantic UI,” 2022)

- **React Toastify**
  React Toastify is a toast notification library, it allows users to add notifications with different themes and styles. React Toastify is licensed under the MIT licensing which allows for commercial use, modification, distribution and private use, therefore, it has no impact on the project. (Khadra, 2022)

- **Axios**
  Axiso is a promises-based HTTP client which passes on the HTTP request from the web browser to the REST API. Axios is licensed under the MIT licensing which allows for commercial use, modification, distribution and private use, therefore, it has no impact on the project. (“Axios Website,” 2022)

Business Layer

- **Python (Open source)**
  Python is a dynamic and interpreted language suitable for developing large projects. Compared to other languages, Python's syntax is simpler and more readable. Also, team members all have some Python experience in previous courses. Therefore, Python is chosen as the backend development language.

  Within python, string matching and list comprehension is also made easier and simpler, which makes it a good candidate when considering the design of our ranking metrics. Built into it are also JSON libraries which can handle JSON objects easily and allows us to pull specific sections of a JSON body passed from the frontend quickly and intuitively.

  Python is an OSI-approved open source licence, making it freely usable and distributable, therefore there are no concerns here.

- **Flask**
  Flask is a web micro-framework that contains the core features needed to build a web application. The core features within flask allow us as programmers to be able to abstract away the low-level details such as protocols and thread management while focusing on functionality of our code.

Flask also has prebuilt library functions to allow for quick and easy connection to our frontend display through routes. The setup object flask provides is also intuitive and simple to use allowing teardown and creation of the connection to be streamlined.

This is the primary method of communicating with the frontend. Werkzeug is one of the toolkits which flask is based on, it allows FLASK to be able to deal with requests as well as response objects. One handy functionality of flask is jsonify, which allows us to easily transmit objects to the frontend through python style lists and dictionaries

Flask has a BSD-3-Clause licence, which means we can modify and use flask in our code as long as we do not use flask as a means of promotion and retain all the copyright information within flask. Since this is a demonstration project and we are not promoting anything this does not affect us.

- **Flask-SQLAlchemy**
  SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. SQLAlchemy itself has had years of development with very well documented code. Allowing new users to learn quickly and easily which is an advantage during sprints.

  Flask-SQLAlchemy is a flask specific extension which simplifies using SQLAlchemy with Flask by setting up common objects and patterns for using those objects, such as a session tied to each web request, models, and engines. One of the biggest benefits of Flask-SQLAlchemy is the removal of metadata, this allows database management to be significantly simpler as it abstracts away the manual management of this.

  Flask-SQLAlchemy follows the flask licensing which means no problems once again with us using this library.

- **REST API**
  REST (Representational state transfer) is an API used by python flask to communicate with the presentation layer module. It exposes a set of public URLs, also known as endpoints, to develop a uniform interface for accessing resources.

Data Layer

- **SQLite**
  SQLite is a small and fast SQL embedded database engine. The main difference between this and other RDBMS is the headless server functionality. Therefore SQLite is more self-contained and we can access the information without the need of setting up an intermediate local host server.

  The reason we did chose to go with this RDBMS over a more fleshed out RDBMS is because we have SQLAlchemy to handle functions for us therefore the database itself can be very barebones and it would still be suitable for our purposes

  SQLite3 is an open source, public domain RDBMS, all code is original and although patches and merges are not allowed to be added to the source without a vetting process, this means that all our code and anything we do with SQLite3 is available for use however we see fit.

Testing Layer

- **Pytest**
  Pytest is an open source testing framework based on Python. It means pytest is a command-line tool that automatically finds tests you've written, runs the tests, and reports the results. This feature is useful in running multiple test files.

  With pytest, we can use the client method in Flask. The test client makes requests to the application without running a live server, which is the reason why we chose pytest for the API tests.

  Also, with pytest fixtures, you can give them as a parameter to the test, but you don't have to execute it. This helps us to reset the database and login session for each test.

  Pytest is distributed under the terms of the MIT licences , making it a free and open source software.

Implementation Challenges

Create requests for tests

A challenge we faced in making tests is to send requests to the backend server without using the frontend interface. Firstly we were using the unittest and requests module to send requests in code lines. Things were going well until the attempt to interact with the database. The database behaved strangely and can not be modified.

Then we were guessing if Flask-SQLalchemy has some bug with requests module, since the database can be queried or modified directly. We also asked the tutor for some advice. However, there was nothing changed with any effort on modifying the test function. Therefore, we decided to rewrite the whole test class with pytest, since it can use Flask's client module to send requests directly to the backend server. While building the new test environment and comparing it with the old test class, the vital issue is located. We defined the server in backend/__init__.py, and every time the backend module is imported, the Flask and database instance is conflicted with the instances created in the test file.

Therefore, to solve this issue, we used the same instance (app and db) in the test file and practical backend server. The side effect of doing so is that the database will be formatted once the test is executed. It can be prevented by manually changing the database URI before executing the test and changing it back afterwards. It may be improved by writing a method to create a Flask instance at the beginning of the project. This issue reminds us of the importance of having good programming practices.

Ranking Metrics & Partial Matching & Comments

The biggest difficulty in determining the ranking metric was mainly due to partial matching tags. We first had to determine a method to create tags. Since usually titles are created based on their ingredients, we decided to use ingredients primarily as the tag source.

For example, most recipes with beef as an ingredient would also mention that as the recipe title such as beef lasagne, beef casserole, or beef sandwich. The next challenge was sorting through these tags and making sure duplicates weren't counted twice in the ranking system.

To approach partial matching we took advantage of sqlalchemy's partial match function, where the DBMS is doing the heavy lifting in sorting as they already have efficient algorithms. Tags can also be indexed within the database.py file to decrease search times if the database gets larger. This allows us to find eggs and egg when we search for egg or eg. This partial matching would also be beneficial to users who have typos as similarly typed ingredients or titles would still have a match.

Once we created the tag system, matching is done through querying the database, filtering the results, ranking the results, then doing some list comprehension to create a viable JSON output for the frontend.

The comment system also had some difficulty. We had to determine a way to retrieve comments based on their status within the reply hierarchy. We decided to implement this in the database through having a parent. Every child knows their parents but a parent doesn't know their child. Within the database we left comments where they were direct replies to the recipes blank for the parent field. Then for child comments of these comments we had a comment ID that let us rebuild it for displaying on the frontend. This leaves us with a small recursion loop that is very clean to read.

We also connect the comment ID and relevant data to the users therefore the frontend doesn't have to process the information further other than to display and format. Here we pass the frontend a list of dictionaries with the relevant information, so the frontend can just loop through and print it out and the order of the comments will be correct as the backend processes this for them.

Connect database with backend and comment

This is the first time I developed software from 0 in the form of a team. Before that, I had some code and needed to complete a function. So, at the beginning of this project, I was always in a muddled state. For the backend, the biggest challenge is that I am not familiar with the database. I have not learned about databases, so I often do not know how to start when connecting to the backend database. I don't understand the method to use or composition of the database, which takes me a long time to learn and understand, and then I can continue to write functions connected with it.

Then there is the understanding of comments. In the past, I always wrote my own code without considering whether others could understand it if I could complete my function. However, through this project, I understand that annotations are critical. It is necessary to write comments about each program and the use of the parameters in it. Otherwise, only you can read your code, which is difficult for others to understand. This is also one of the challenges faced by the project.

Frontend - Web design

For a first time front-end programmer, having to learn the syntax of React JSX and javascript while having to make sure the frontend component is intuitive and user friendly is quite challenging.

When designing the user interfaces, I chose a light green colour as the webpage's theme colour, hoping to promote the healthy and natural culture of our webpage. After seeking suggestions from our tutor, I learnt that white text on top of a light green background is not user friendly as the colours do not contrast and it is hard to see the text.

While developing the frontend, I often find myself navigating around the webpage to test on the user experience. I realised that the frequent page refresh annoys me as a user. To solve this problem, I removed all the links and navigation methods associated with buttons provided by the react library and replaced all with the link and use the navigate method provided by react router dom. This makes navigation between pages way faster and smoother which increases user satisfaction rapidly. Furthermore, instead of clicking on the search icon on the search bar to search for recipes, i disabled the click action of the icon and instead triggers a query request to backend everytime there is a change in the search input base, this saves user from clicking every time they want to search a recipe and made the searching action more responsive through seeing the changing of the recipe summaries as the user type.

Another challenge regarding visual design is ensuring there is enough white space between UI elements. For this, I used the material UI grid component to line up the elements and apply paddings to all the grid items. I also used the marginTop and marginBottom prop to the container style to make sure there is enough space between the content and the navbar/ bottom of the page.

User Manual

For all instructions, we assume you are on the Lubuntu 20.4.1 LTS virtual machine image as described here: https://webcms3.cse.unsw.edu.au/COMP9900/22T3/resources/80157

Download the source code

- Download project.zip
- Extract all files from project.zip

Run the backend server

Software required: python 3.7/ python 3.8/ python3.9

1. Once python is installed create your virtual env with

   python3 -m venv venv

2. Then run your virtual env with

   source venv/bin/activate

3. In the capstone directory, run below to install all the dependencies packages listed in the requirement.txt.

   pip install -r requirments.txt

4. Once this is done, create the database from fresh if you want a clean database otherwise ignore the following Currently the database is premade with several example recipes

   python3 make.py

5. Finally, run the backend using the following Command.

   python3 run.py

6. Go to http://localhost:5000/check in your browser. It should display raw json data that includes the message “server is up”and status code “200”.

7. For tests. Do NOT use command pytest to prevent import error. In the capstone directory, input the following command to run the test.

```
python3 -m pytest tests
```

Run the frontend application

Software required: npm 6.16.17, node v14.20.1

1. Open a separate terminal. From the capstone directory, cd into the “client” directory and run the following command to install all the dependencies packages listed in the package.json file.

```
npm install
```

2. If errors occur use the following instead

```
npm install --force
```

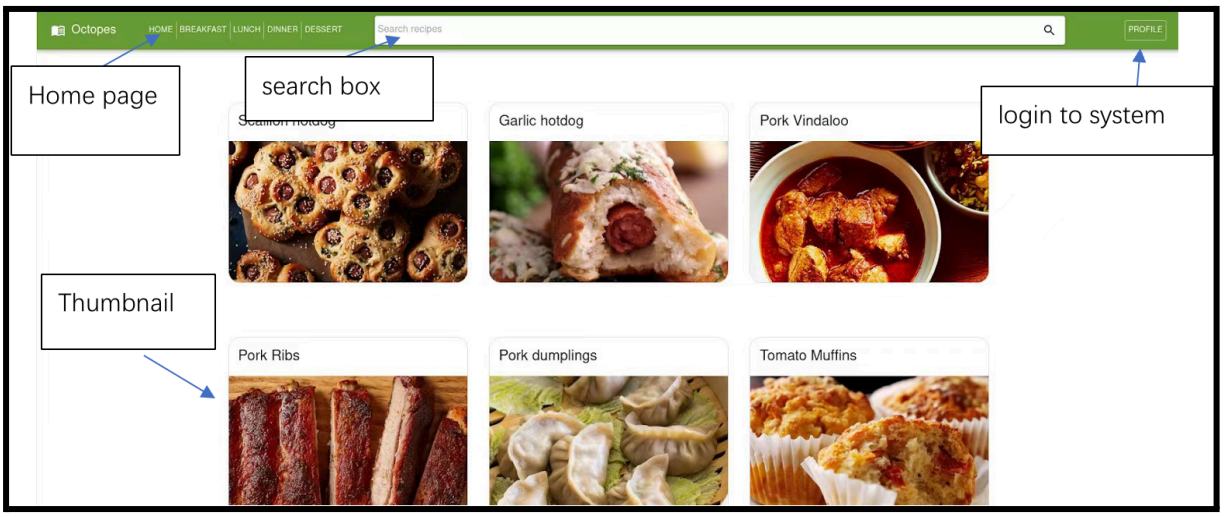
3. Once this is done, you can run the following command to start the application:

```
npm start
```

4. If the browser does not automatically pop up, open http://localhost:3000 to access the webpage.

How to use the user interface

Home page



Home page is where all users begin after entering the web application. Users can perform the follow actions:

• Clicking on the “login” button at the top right corner - to navigate to the login page, after login, the same button changes to become the profile button which navigates the user to the profile page.

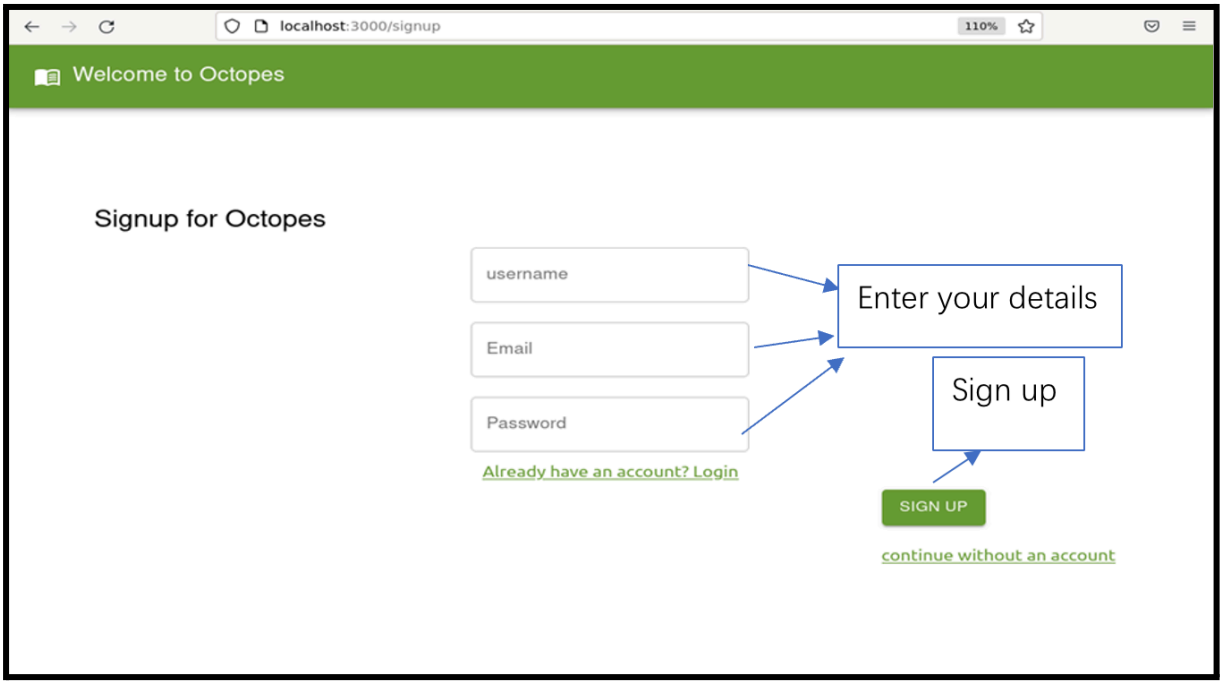
• Clicking on the recipe thumbnail - to navigate to the recipe page.

• Search using the search box - to replace the newsfeed/ previous search result by the searching result.

• Clicking on meal type buttons on the top bar - to search for recipes with the corresponding meal type filter.

• Clicking on home on the top bar - to clear the search result and replace it with news feed.

Signup page

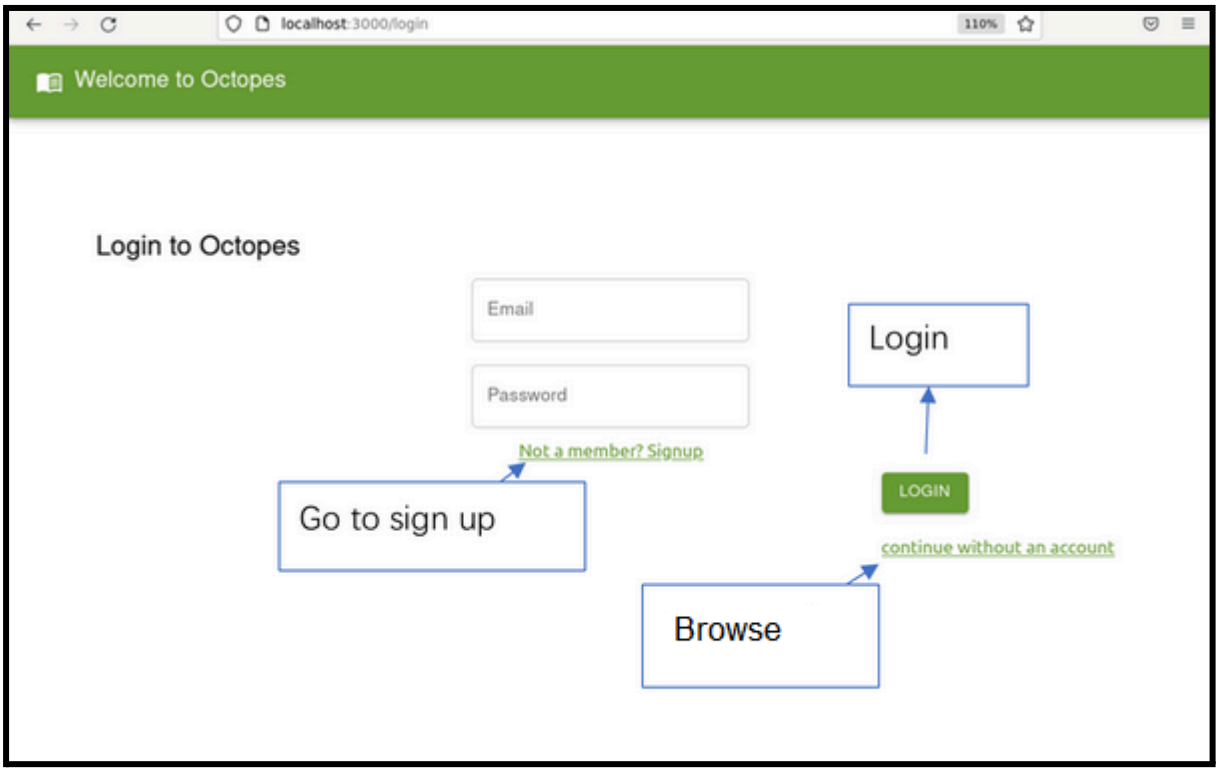


Sign-up page is where users create an account. Users can perform the follow actions:

• Clicking on the “Sign in here” hyperlink - to navigate to the sign in page.

• Clicking on the “create account” button - to create an account and navigate to the home page if all valid information is filled in the text boxes. If one or more fields are not valid, a corresponding error message is displayed, and the user will be prompted to enter again.

Signing In page



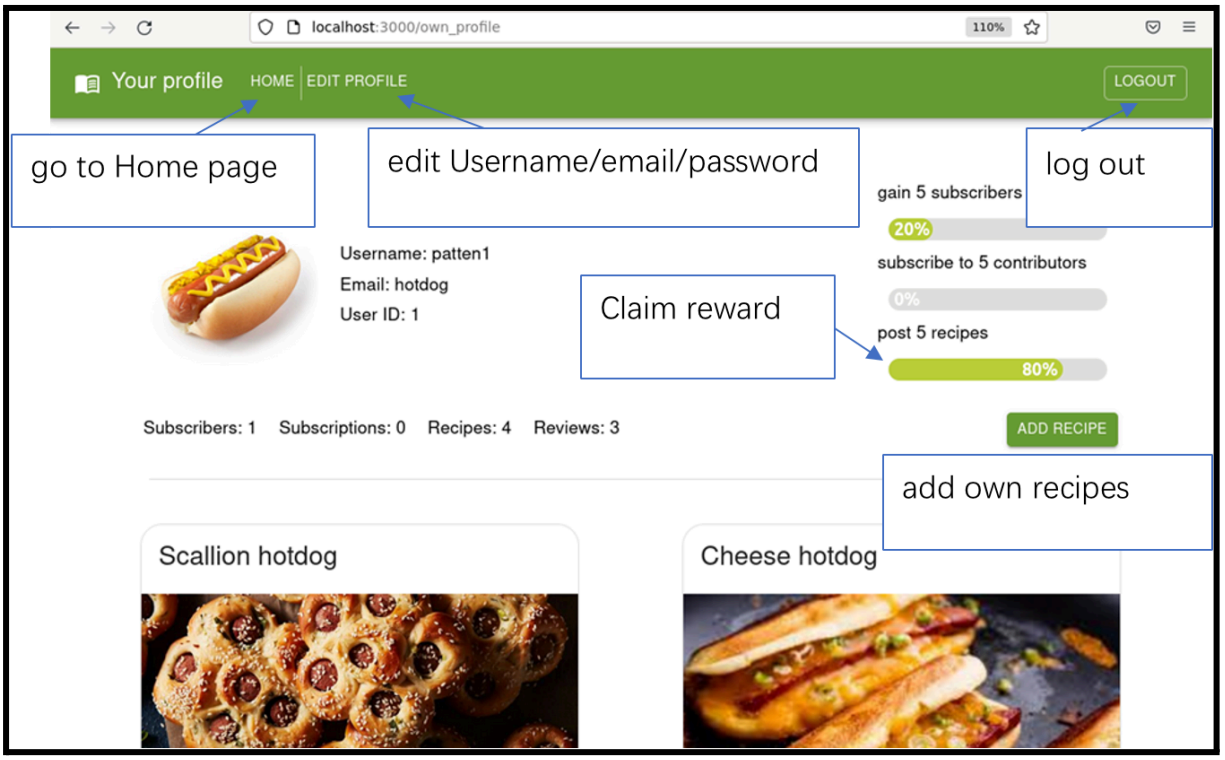
Sign in page is where user's login to the system using their login Confidentials. Users can perform the follow actions:

• Clicking on the “Not a member with us? Sign up” hyperlink - to navigate to the sign-up page.

• Clicking on the “login” button - to login and navigate to the home page if all valid information is filled in the text boxes. If one or more fields are not valid, a corresponding error message is displayed, and the user will be prompted to enter again.

• Clicking on the “continue without an account” - View without account

View your own profile page



Profile page is where users display their information and recipes. Users can perform the follow actions:

- Clicking on “home” on the top bar - to navigate to the home page.

- Clicking on “logout” on the top bar - to sign out and navigate to the home page.

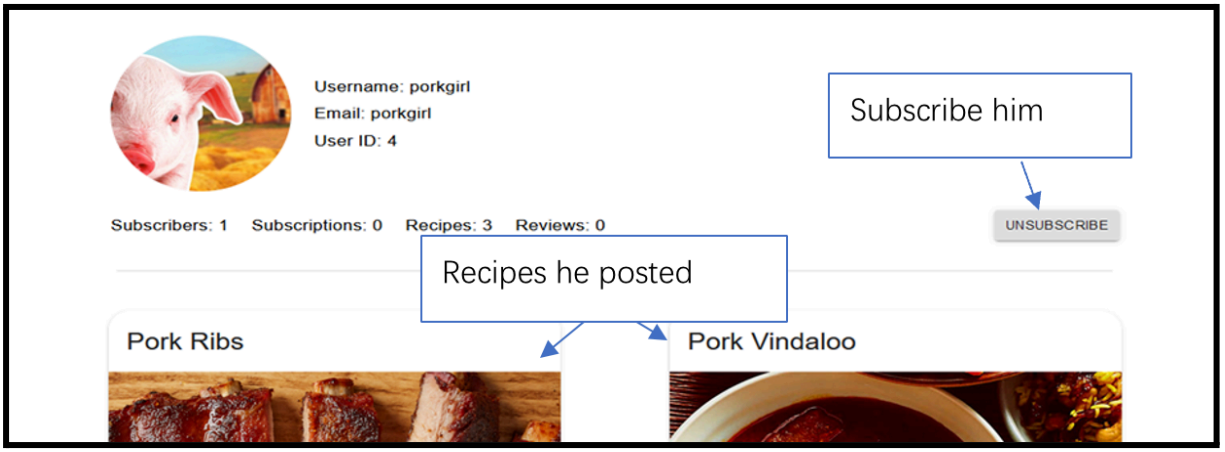
- Clicking on the “claim” button - to claim a reward if the button is activated. If the button is not activated, nothing happens.

- Clicking on the edit profile button on the top- to navigate to edit profile page.

- Clicking on the recipe thumbnail - to navigate to the recipe page.

- Clicking on the “add a new recipe” - to navigate to add recipe page.

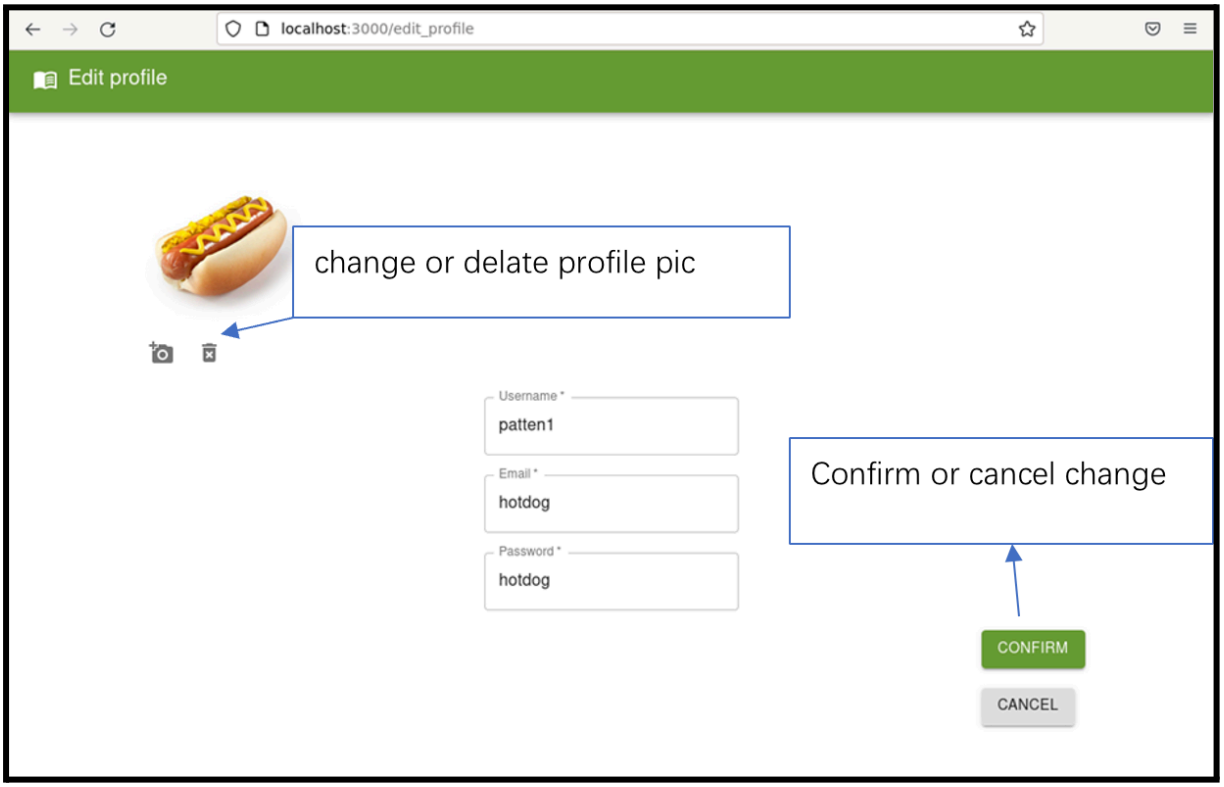
View Other Profiles



Other's profile page is where users view other user's profiles and subscribe to other users. Users can perform the follow actions:

- Clicking on "home" on the top bar - to navigate to the home page.
- Clicking on the "subscribe" button - to subscribe to the user.
- Clicking on the recipe thumbnail - to navigate to the recipe page.

Edit - Profile



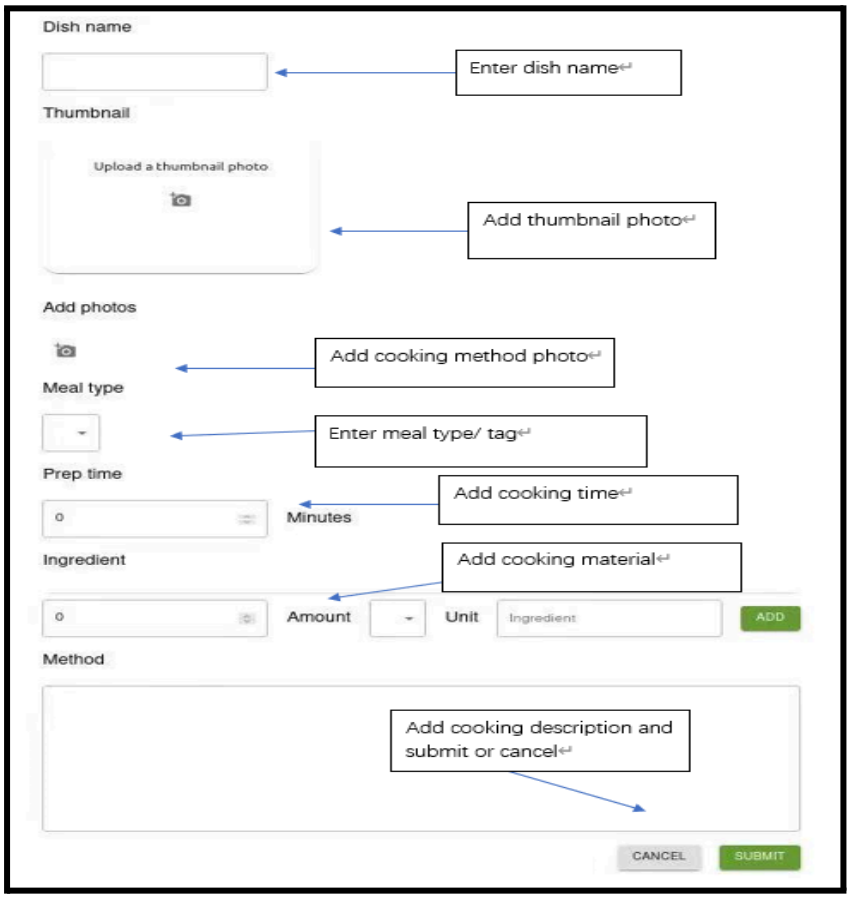
Edit profile page is where users maintain their profile information and maintain their recipes. Users can perform the follow actions:

- Clicking on the “confirm” button - to successfully edit information if all text boxes are validly filled, otherwise a corresponding error message is displayed, and the user will be prompted to edit the fields.

- Clicking on the “cancel” button - to cancel any changes made and navigate back to the profile page.

- Clicking on the camera icon on the profile picture - to change the profile picture by uploading an image.

Add-Recipe Page



Add recipe page is where users add a recipe. Users can perform the follow actions:

- Clicking on each input box to add recipe details and photos. For ingredients, enter the amount first, then select the unit and input ingredient name, finally click the “ADD” button to add a type of ingredient.

- Clicking on upload thumbnail photos to add cover for thumbnail.

- Clicking on “Submit” on the bottom - to successfully add a recipe if all necessary attributes are validly filled, otherwise a corresponding error message is displayed, and the user will be prompted to edit the fields.

- Clicking on “Cancel” on the bottom - to go back to the user profile page.

Recipe Page

A recipe page is where users learn to cook a dish. Users can perform the follow actions:

- Clicking on the “login” button at the top right corner - to navigate to the login page, after login, the same button changes to become the profile button which navigates the user to the profile page.

- Clicking on the users’ profile picture - to navigate to the users’ profile page.

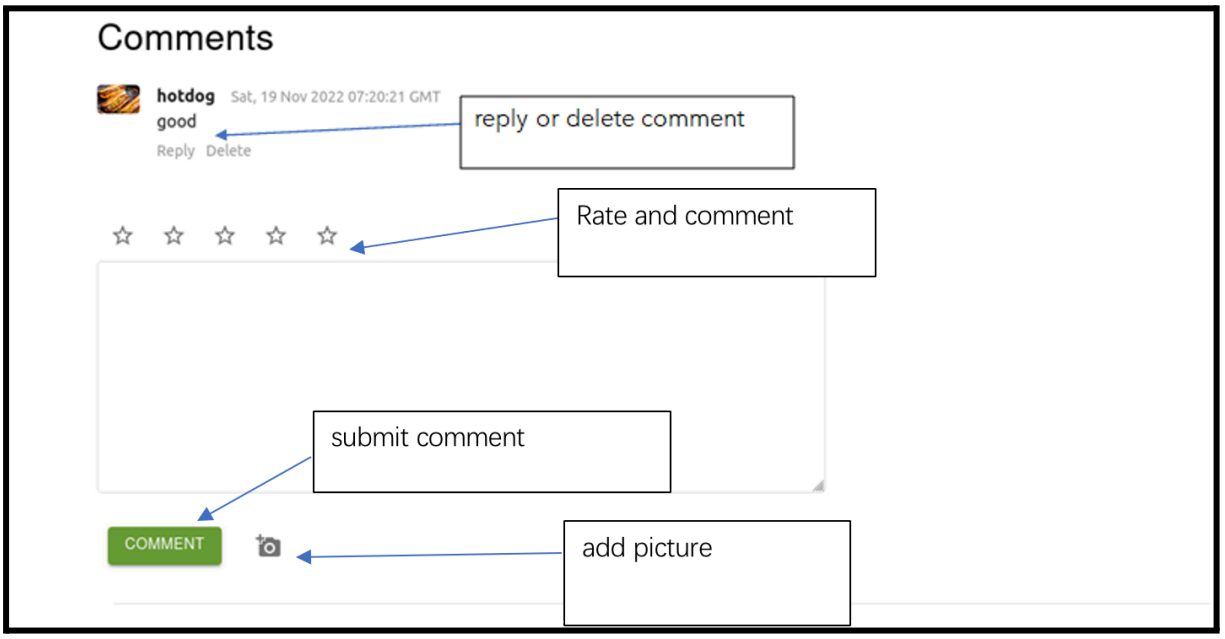
- Clicking on the heart icon - to like/ unlike the recipe.

- Clicking on the “Delete recipe” button - to delete the recipe. Only available when the recipe is created by the currently logged in user.

- Clicking on the “Edit recipe” button - to navigate to edit the recipe with all existing information filled in. Only available when the recipe is created by the currently logged in user

- Clicking on the recommendation recipe thumbnail on the bottom of the page - to navigate to other recipe pages.

Comment Page



Comments are below the recipe detail page. Users can perform the follow actions:

• Clicking on the “Reply” button - to reply to other users’ reviews or replies.

• Clicking on the “Delete” button - to delete the user’s reply. Only appear when the comment is posted by the currently logged in user. All replies to the deleted comment will also be deleted.

• Clicking on the “Comment” button - to successfully add a review post if all text boxes are validly filled with optional picture attached, otherwise a corresponding error message is displayed, and the user will be prompted to edit the fields. The rate bar should be filled by 1 to 5 stars.

• Clicking on the camera icon - to add a photo for the review.

References

json — JSON encoder and decoder — Python 3.8.3rc1 documentation. (n.d.). Retrieved from docs.python.org website: https://docs.python.org/3/library/json.html

License — Flask Documentation (2.2.x). (n.d.). Retrieved from flask.palletsprojects.com website: https://flask.palletsprojects.com/en/2.2.x/license/

License — Flask-SQLAlchemy Documentation (2.x). (n.d.). Retrieved November 19, 2022, from flask-sqlalchemy.palletsprojects.com website: https://flask-sqlalchemy.palletsprojects.com/en/2.x/license/

Object Relational Tutorial (1.x API) — SQLAlchemy 1.4 Documentation. (n.d.). Retrieved November 19, 2022, from docs.sqlalchemy.org website: https://docs.sqlalchemy.org/en/14/orm/tutorial.html

Quickstart — Flask-SQLAlchemy Documentation (2.x). (n.d.). Retrieved from flask-sqlalchemy.palletsprojects.com website: https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/

SQLite Copyright. (n.d.). Retrieved from www.sqlite.org website: https://www.sqlite.org/copyright.html

SQLite Frequently Asked Questions. (n.d.). Retrieved from www.sqlite.org website: https://www.sqlite.org/faq.html

Welcome to Python.org. (2019, August 30). Retrieved from Python.org website: https://www.python.org/about/

Werkzeug — Werkzeug Documentation (2.2.x). (n.d.). Retrieved from werkzeug.palletsprojects.com website: https://werkzeug.palletsprojects.com/en/2.2.x/

pytest: helps you write better programs — pytest documentation. (n.d.). Retrieved from docs.pytest.org website: https://docs.pytest.org/en/7.2.x/

License — pytest documentation. (n.d.). Retrieved from docs.pytest.org website: https://docs.pytest.org/en/7.2.x/license.html

Semantic UI. (2022, November 19). Retrieved November 19, 2022, from GitHub website: https://github.com/Semantic-Org/Semantic-UI/blob/master/LICENSE.md

Axios Website. (2022, November 17). Retrieved November 19, 2022, from GitHub website: https://github.com/axios/axios-docs/blob/master/LICENSE

MUI Core. (2022, April 22). Retrieved from GitHub website: https://github.com/mui/material-ui/blob/master/LICENSE

Khadra, F. (2022, November 18). React-Toastify. Retrieved from GitHub website: https://github.com/fkhadra/react-toastify/blob/main/LICENSE