reciPicker – A Recipe Sharing Web Application

Liam Olohan

N00193194

Supervisor: Catherine Noonan

Second Reader: John Montayne

Year 4 2022-23

DL836 BSc (Hons) in Creative Computing

Abstract

The aim of this project is to construct a web application that allows for finding recipes and learning for people of all levels, ranging from newly learning home cooks to experienced culinary professionals. The purpose of this application is to make the process of finding new recipes from various global cuisines easier by consolidating them into a single platform, rather than having to conduct multiple searches for a specific meal.

Throughout the application's development, all functionality was tested to ensure they worked correctly, and the results indicate that it meets the desired standard for its existing features and capabilities. However, the application could be enhanced with additional features and improved functionality if provided with additional development time. For instance, allowing administrators to edit and delete categories and ingredients and implementing advanced security measures that enable users to modify their personal information, such as email and password, with verification processes to confirm ownership of their account email could be potential areas for further improvement.

Acknowledgements

I would like to thank all the people who showed their support and offered me advice. In particular, I would like to thank my project supervisor, Catherine Noonan, and my second reader John Montayne, for their help, communication, and support throughout, during the development of the recipicker application.

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not to leave copies of your own files on a hard disk where they can be accessed by other. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student :

Signed

_____

Failure to complete and submit this form may lead to an investigation into your work.

Table of Contents

1 Introduction

The recipicker application is a recipe sharing platform that aims to provide users with a wide range of recipes, complete with detailed information on time, ingredients, and nutritional value. This application is designed for anyone who loves food and wants to discover new recipes to try. With a user-friendly interface, recipicker makes it easy for users to find the perfect meal to cook for themselves, their families, or anyone else.

To develop this application, we used several technologies, including ReactJS for the front-end, Express.js for the back-end, and MongoDB as the database. ReactJS is an open-source JavaScript library that allows developers to build dynamic user interfaces by breaking them down into smaller, reusable components. This approach simplifies the debugging process and makes it easier to manage complex applications. Express.js is a lightweight and flexible back-end framework for Node.js that provides developers with the tools they need to build scalable web applications. Finally, MongoDB is a NoSQL database program that uses JSON-like documents to store data, making it ideal for web applications that need to handle large amounts of unstructured data.

To manage this project, we used GitHub, a popular version control system that allows developers to collaborate on code and track changes over time. By using GitHub, we were able to ensure that our code was always up to date. This also made it easier to work on the project from different locations and devices, as all of the code was stored in a centralized location.

Overall, the recipicker application is a user-friendly recipe sharing platform that is designed to meet the needs of anyone who loves food.

2 Research

2.1 Introduction

In recent years, there has been a large shift by web developers to use frontend JavaScript frameworks rather than just basic HTML, CSS & [Vanilla] JavaScript due to the want and / or need for the ability to do more advanced techniques and with the use of these sophisticated frameworks, the techniques become a lot more accessible to developers of all levels in much less amount work, time and complexity.

The aim of the web application that I hoping to develop is to create a platform that creates a community of people that are going to share their food recipes, give feedback on other people's recipes, and so much more, but to do this I must decide which web framework I want to use, that will have the best feature set for my needs, and ensure it has the ability to scale as the application grows, the user base grows and will be able to support potential future feature implementations without compromising on optimisation and performance to ensure the best user experience across all devices.

2.2 What Are Web Application Frontend Frameworks

A front-end framework is a collection of code libraries, the language of which will depend on the type of framework you are using. Primarily JavaScript is the language used in front-end frameworks for web applications. Due to this, JavaScript Front-end frameworks will be focused on in this.

The most popular JavaScript front-end frameworks at the end of 2022, based on the number of questions asked on Stack Overflow, which is a platform for developers to go and ask questions and attempt to get the help they need, are as follows:

- Reactjs
- Angular
- Vue.js

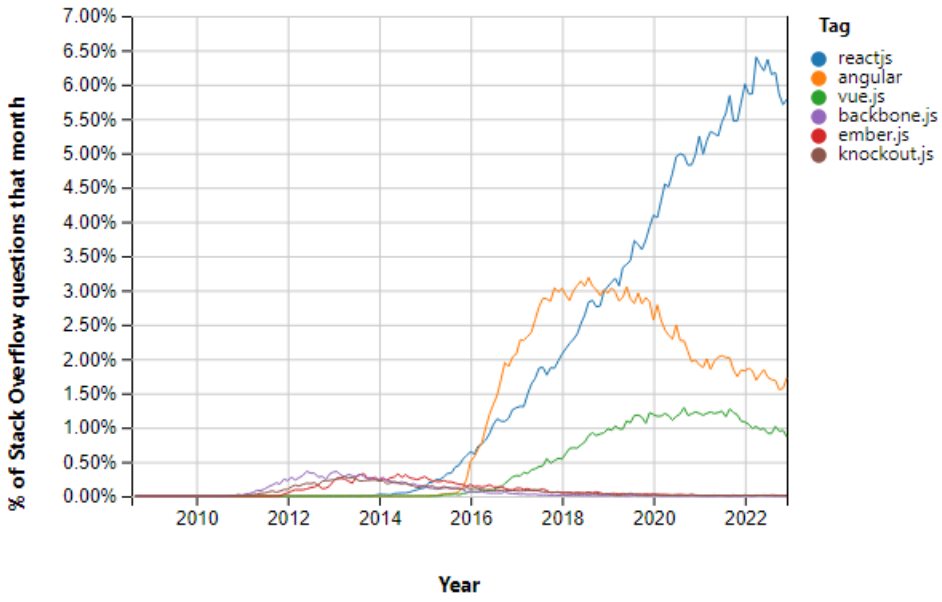


FIG 0.0 – SOURCE: Stackoverflow

2.2.1 What is Angular

"Angular is FW for creating client applications in HTML and JavaScript or in a language such as TypeScript compiled into JavaScript. It was developed by Google in 2010 as AngularJS, and in 2014 it was completely reworked and rewritten and since then it has been operating under the name Angular. FW consists of several libraries, some of which are basic and some optional"

- KALUŽA, M., TROSKOT, K., & VUKELIĆ, B. (2018).

Angular, which was formerly known as AngularJS is a front-end JavaScript framework was originally developed in 2010 by Google. It was then overhauled in 2014 when it was renamed to just Angular.

Some of the benefits to using Angular are linked to the fact it is developed by a large corporation in Google. With Google being such a large corporation, it means they built Angular to scale to a good standard, as Google were a growing company when Angular was originally developed and they needed to be able to scale with the growth of the company. This is also another reason for it being a well-supported framework as it has the Google developers behind it to keep it updated, a long with a large community of independent developers that use the framework developing different packages to improve it for their needs also.

Angular does have a large learning curve when you are first getting started with it, as it requires a certain knowledge of JavaScript in order to use.

2.2.1.1 Pros of Angular

- Maintained by a large corporation, Google.
- Good performance
- Very scalable framework
- Large Ecosystem

2.2.1.2 Cons of Angular

- Large learning curve
- Complicated

2.2.2 What is React

"React is designed to enhance interactive UI development by making it easier to update the view when the data changes. It is done through dividing the view into smaller components, that can be composed to create complex UIs. Components are built in JavaScript instead of templates, enabling easy flow of data"

- ELAR SAKS. (2019).

React, also known as ReactJS is a front-end JavaScript library that was created in 2013 by Jordan Walke at Meta, formerly known as Facebook Inc. React is maintained by Meta themselves, along with a large community of independent developers.

React is a front-end JavaScript library, it is not specifically a framework, although you will often hear it still referred to as a framework. A JavaScript library is a large collection of pre-existing JavaScript code that can be used any number of times, on a per use basis, it is not required. React specialty and the reason it was created by Facebook was to help in the creation of User Interfaces (UI), a user interfaces are the on-screen elements such as the buttons, forms, search boxes, etc. on a website.

React works by using a Virtual Document Object Model (DOM). A DOM is what displays the output of the website on the page, what React does is it updates the Virtual DOM when changes are made, be it user input, or by the developer during the creation of the web application. React checks constantly for changes to the Virtual DOM, then it will update the regular DOM, with what whatever is different in the Virtual DOM in order to reduce the amount of re-rendering. React uses this method in order to be more lightweight and keep the compute to a minimum when re-rendering, rather than re-rendering the entire DOM when making changes.

React does have a large learning curve, as it requires a certain knowledge of JavaScript in order to use, unlike a different framework that was developed for UI purposes such as bootstrap.

2.2.2.1 Pros of React

- Maintained by a large corporation, Meta & Independent developers / companies
- Great Performance due to virtual DOM
- Advanced set of useful dev tools
- Updated often
- Reusable components

2.2.2.2 Cons of React

- Large learning curve
- Documentation isn't the best due to consistent updates

2.2.3 What is Vue.js

"Vue.js is today one of the fastest growing JavaScript FWs when it comes to popularity. On its official pages it is described as accessible, comprehensive and high performing FW for development of interactive interfaces"

- KALUŽA, M., TROSKOT, K., & VUKELIĆ, B. (2018).

Vue.js is a front-end JavaScript framework, that follows the model-view-viewmodel architectural pattern. Vue.js was created in 2014 by a former Google employee, Evan You. Vue was considered the fastest growing front-end framework as of 2018, but this was surpassed by React.

Vue.js takes from different parts from both Angular and React in the way it works. Vue has a large number of different plugins and packages available to developers who are taking advantage of this front-end framework.

Vue.js unlike Angular & React is currently undergoing a major shift in the way it is used and works. It is upgrading to Vue 3 from Vue 2 over the last number of years, which fundamentally changes Vue as it was rebuilt from the ground up using their new Options API in place of the formerly used Compositions API, which is still usable but is being phased out. Making Vue work in a much more similar way to React than it did previously.

2.2.3.1 Pros of Vue.js

- Best performance of the three frameworks
- Simpler syntax if you have JavaScript knowledge
- Great documentation, making it perfect for beginners
- TypeScript support

2.2.3.2 Cons of Vue.js

- Small Community compared to Angular & React
- Documentation isn't the best due to consistent updates
- Lack of third party packages in comparison
- Not Maintained by a large corporation

2.3 Angular vs React vs Vue

"Angular is a fully-fledged front-end framework, React is a UI library, and Vue.js is a progressive framework. They can be used interchangeably to build front-end applications, but they're not 100 percent the same, so it makes sense to compare them and understand their differences. Each framework is component-based and allows the rapid creation of UI features. However, they all have different structures and architecture."

- VYAS, R. (2022).

Angular is the oldest of the three front-end frameworks in this comparison, with it being created in 2010 by Google. It was followed by the creation of React by Facebook in 2013, shortly followed by Vue a year later in 2014 by Evan You.

Although React was created by a large corporation in Facebook, it is the only option of the three frameworks that is not only maintained by a single entity. React is maintained by Meta, formerly Facebook as well as an incredibly large number of independent developers and companies, making it probably the most well maintained framework. Whereas Angular is also created by a mega company in Google, they are also the only ones to maintain the framework. Vue on the other hand was not created by a mega corporation, but it is still only maintained by Evan You and the Vue core team members.

React is not a very opinionated JavaScript library, but in the case of how the page is rendered, it is opinionated. But this is the only time. Unlike Angular & Vue.js although they have certain parts that are unopinionated, with Vue being a lot less opinionated than Angular, Angular is only not opinionated when it comes to State Management.

In terms of the framework being cross-platform, Angular is the only one of the frameworks that was built with flexibility in mind, meaning it was there from the offset. Whereas with both Vue and React, they require Vue Native and React Native respectively for building cross-platform applications.

The languages in which each of the frameworks is written is the same for both Angular and Vue, both of these frameworks were written in the Object Orientated Programming (OOP) languages known as TypeScript. TypeScript is known as JavaScript with syntax, as it builds on JavaScript. TypeScript was developed by Microsoft, who are also the ones that maintain the language. On the other hand, React was written in JavaScript, unlike TypeScript, JavaScript is not an Object Orientated Programming language, JavaScript is considered a prototype-based language.

It is with data binding that is one of them major differences between Angular and the other two, React and Vue. Angular uses a two way data binding system, and both of Vue and React use a one way data binding system.

A more concise view of the above, with some extra information.

Question	Angular	React	Vue.js
What type of DOM is used?	Real	Virtual	Virtual
Can it manage Data States?	Yes	Yes	Yes
Do you need external packages?	No	Yes	Yes*
Is there too much overhead?	Yes	No	No
Do independent developer help with maintaining?	No	Yes	No
Does it have a large ecosystem?	Yes	Yes	Yes
Is it opinionated	Yes*	No*	Yes*
Built by a major corporation?	Yes	Yes	No
Type of data-binding?	Two-way	One-way	One-way
Is it cross-platform?	Yes	Yes*	Yes*
Does it use server-side rendering?	Yes	Yes	Yes
What language is it written in?	TypeScript	JavaScript	TypeScript

*Sometimes

2.4 Conclusion

Every single one of these front-end JavaScript frameworks in Angular, Vue & React have their pros and cons to them. But are all a valid choice to use in early in 2023. When should we use each one of them, you may wonder. Well, from my research and review I have found out a number of different things. Starting with the popularity, React is by far the most popular “framework”, although it is a JavaScript library in 2023, followed by Angular and finally Vue, so in terms of finding jobs beyond my specific use case for what I am hoping to build as my web app, React would make the most sense to learn and master, as although it is not the easiest of the three to learn, that being Vue, and Angular being the hardest, it is far and away the most popular. If you are wanting to build a large scale application that will be able to continue to expand further and further into the future, Angular would be the choice to go with as Google has developed Angular in a way that it should easily out do both React and Vue in that use case.

But if you are looking for the most performant of the three frameworks, Vue is the best option. Vue has taking parts from both React and Angular in the way that it has been built and optimised them further than they already were in the other two, making it the most lightweight and fastest of them all. What Vue suffers from, specifically as moves into Vue 3 is the lack of support from the community’s third party packages. With how Vue was developed, and the massive under the hood changes going from Vue 2 to Vue 3, many third party packages have not been updated due to them still need to be able to support Vue 2 due to the much larger user base that has not been able to easily migrate to Vue 3.

In conclusion, all of the frameworks have their benefits and their ideal use cases. Angular for larger, scaling applications, React for general purpose and Vue for lightweight, best performance. But it will be up to the project lead to decide which is best for them to use, and the knowledge they have. But if it is for learning purposes and the future ability to gain work, React should be the focus as the popularity has soared in the last number of years, and will continue to do so with many jobs looking for React capable developers.

3 Requirements

3.1 Introduction

In this project, we will be building a web application that allows users to share food recipes with people around the world. The application will have a variety of features to enable users to fully engage with the content on the platform. The application will be built using ReactJS for the frontend, ExpressJS for the backend, and MongoDB for the database.

Users will have the ability to create, edit, and delete their own recipes. They will also be able to bookmark recipes for future reference and leave comments on other users' recipes. Additionally, users will be able to search for recipes by food category and view recently added recipes.

Pros of a recipe sharing application:

- Large collection of recipes: A vast collection of recipes, which makes it easy for users to find something they are interested in.
- User-generated content: It will allow users to submit their own recipes and give feedback which can add a sense of community and engagement to the platform.
- Search functionality: A search functionality will make it easy for users to find recipes that meet their specific criteria.
- Social features: Having features such as commenting and rating, which can allow users to engage with the content on the platform and get feedback on their own recipes.

Cons of a recipe sharing application:

- Quality control: The quality of recipes can vary depending on the source, and some of the recipes on the website might not be accurate or well written.
- Information overload: With so many recipes available, it can be overwhelming for users to find the one they want.

3.2 Requirements gathering

3.2.1 Similar applications

Epicurious - https://www.epicurious.com/

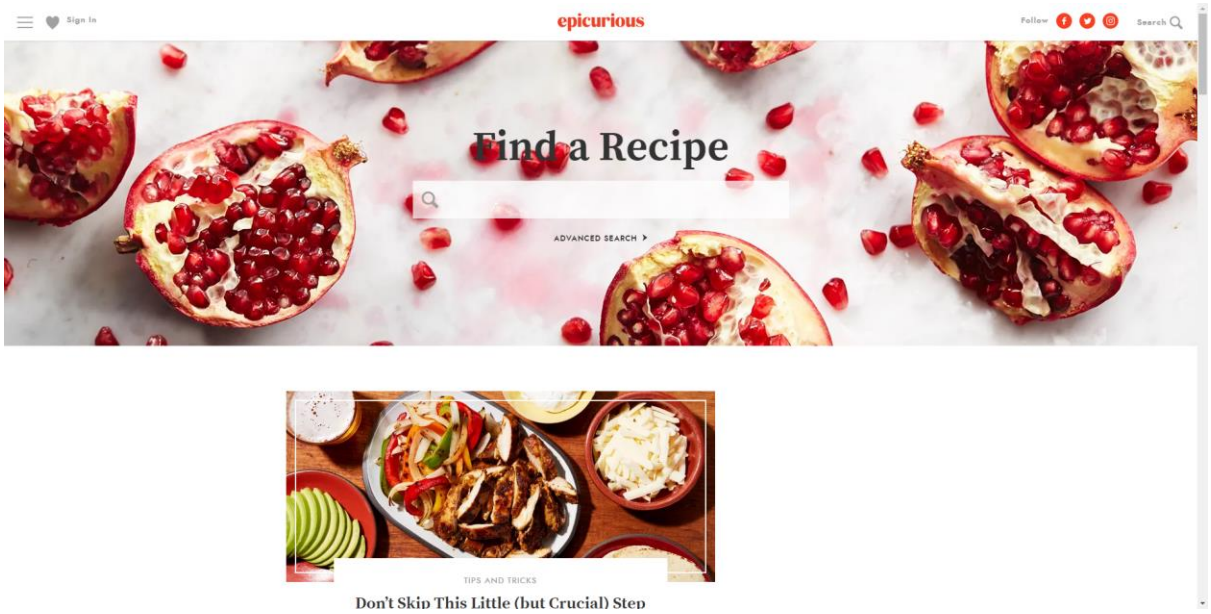


FIG 1.0 – EPICURIOUS HOMEPAGE

Epicurious is a web application that provides users with recipes & menus, expert advice through guides, tips and tricks and general advice from people with years of experience. They also provide many different articles about food related holidays and events taking place, along with different gift ideas for people that are interested in food.

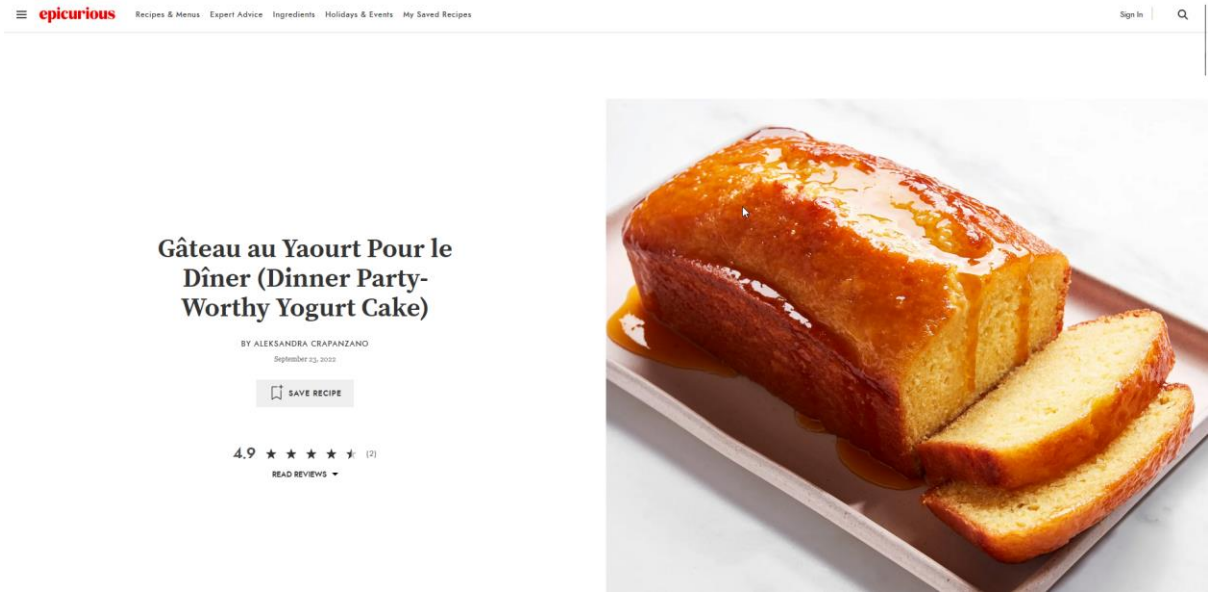


FIG 1.1 - EPICURIOUS RECIPE PAGE

Advantages of Epicurious:

- Ability to create a menu of all your recipes.

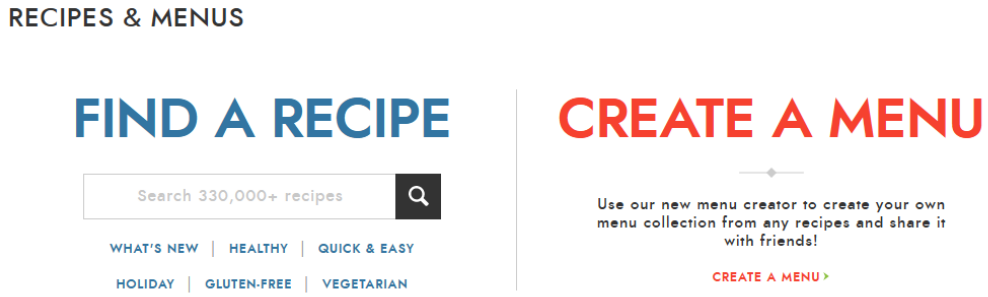


FIG 1.2 – EPICURIOUS CREATE/SEARCH

- Recipes for specific holidays and events

FIG 1.3 – EPICURIOUS HOLIDAY RECIPES

- Simple & Clean design for new recipes section

FIG 1.4 - EPICURIOUS NEWEST RECIPES

Disadvantages of Epicurious:

- Limited viewing experience as a free user.

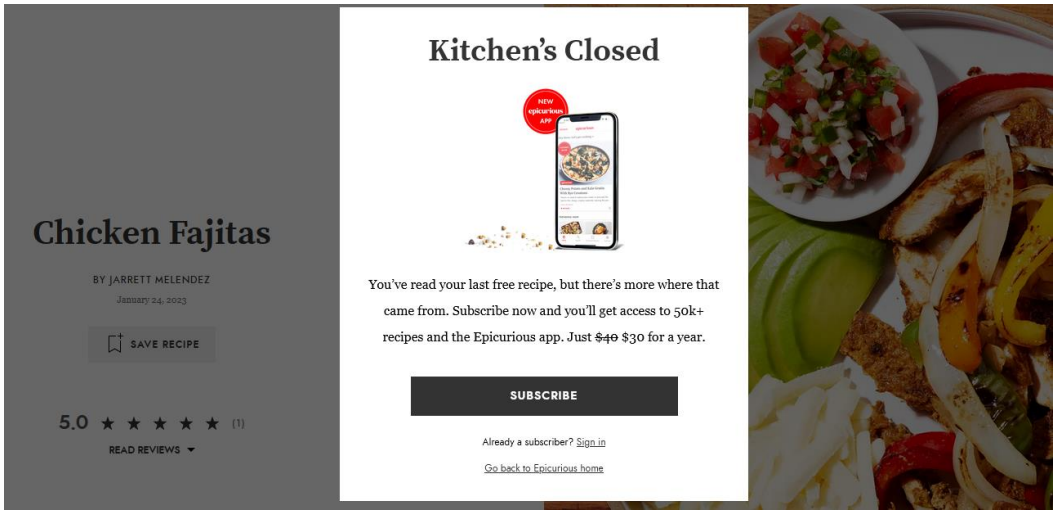


FIG 1.4 – EPICURIOUS SUBSCRIPTION

- Only supports a light mode design.

Allrecipes – https://www.allrecipes.com/

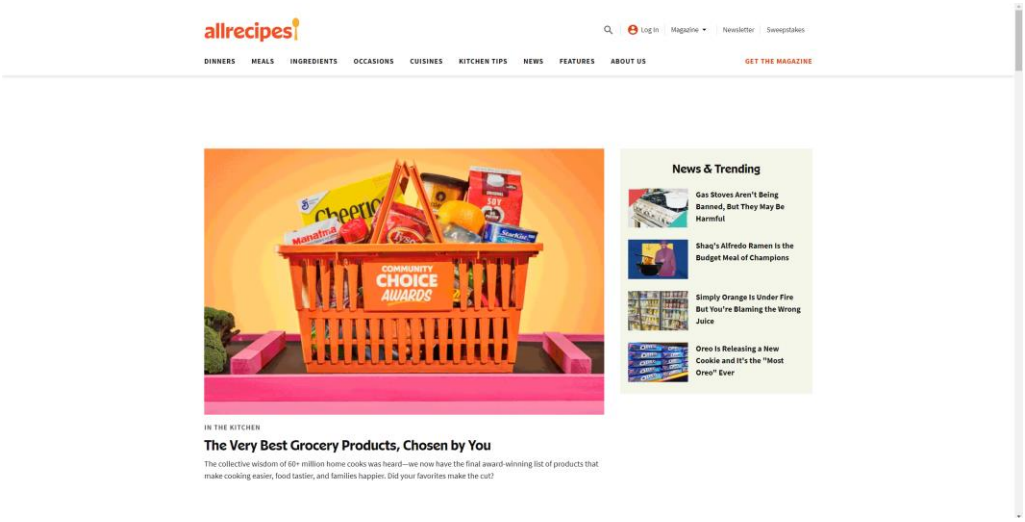


FIG 2.0 – ALLRECIPES HOMEPAGE

Allrecipes.com is a recipe-sharing website where users can browse, save, and share recipes. The website features a diverse selection of recipes, including options for different dietary restrictions. Users can leave ratings and comments on recipes and post their own recipes to the community. Allrecipes.com also offers cooking tips, tutorials, and videos to assist users in the kitchen.

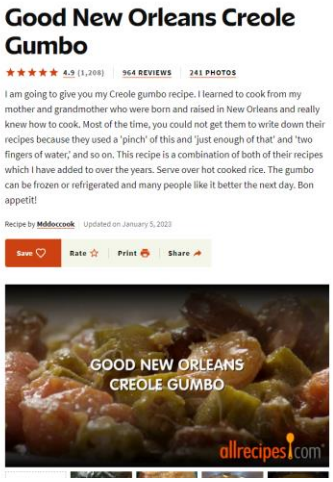


FIG 2.1 – ALLRECIPES RECIPE PAGE

Advantages of Allrecipes

- A vast collection of recipes that have been contributed by users all around the world.

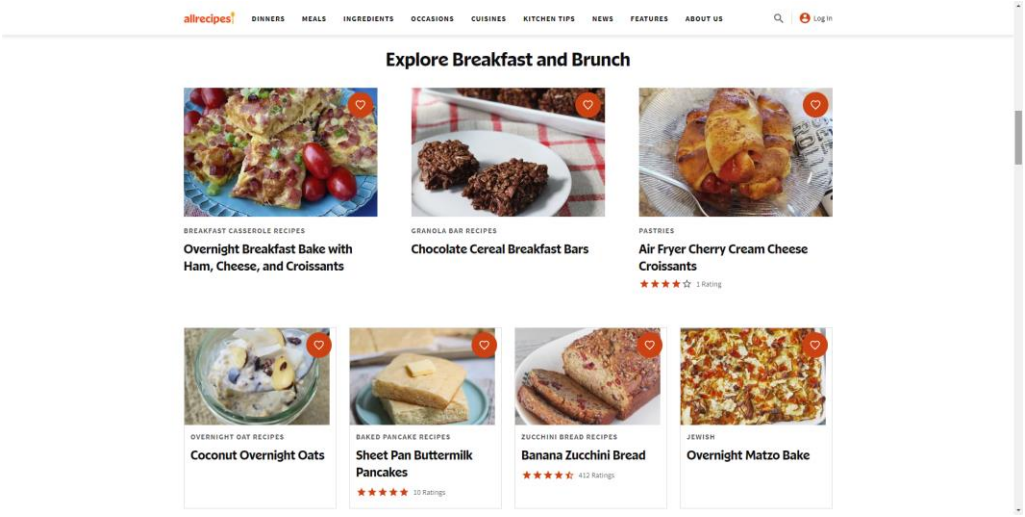


FIG 2.2 – ALLRECIPES ALL RECIPES PAGE

- All Recipes has a recipe rating system that allows for users to rate, and review based on their personal experiences, which helps to identify popular recipes.

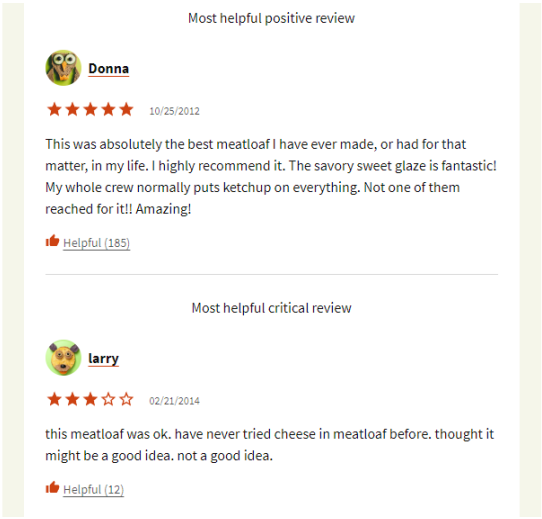


FIG 2.3 - ALLRECIPES REVIEW SECTION

- Easy to use design, with a straightforward navigation system.

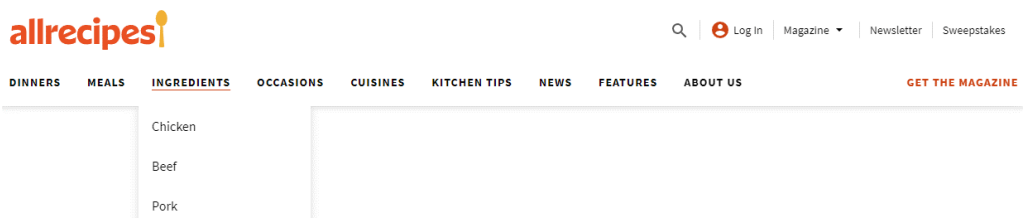


FIG 2.4 – ALLRECIPES NAVIGATION

Disadvantages of Allrecipes

- Some of the advertisements can be quite intrusive to the user experience.

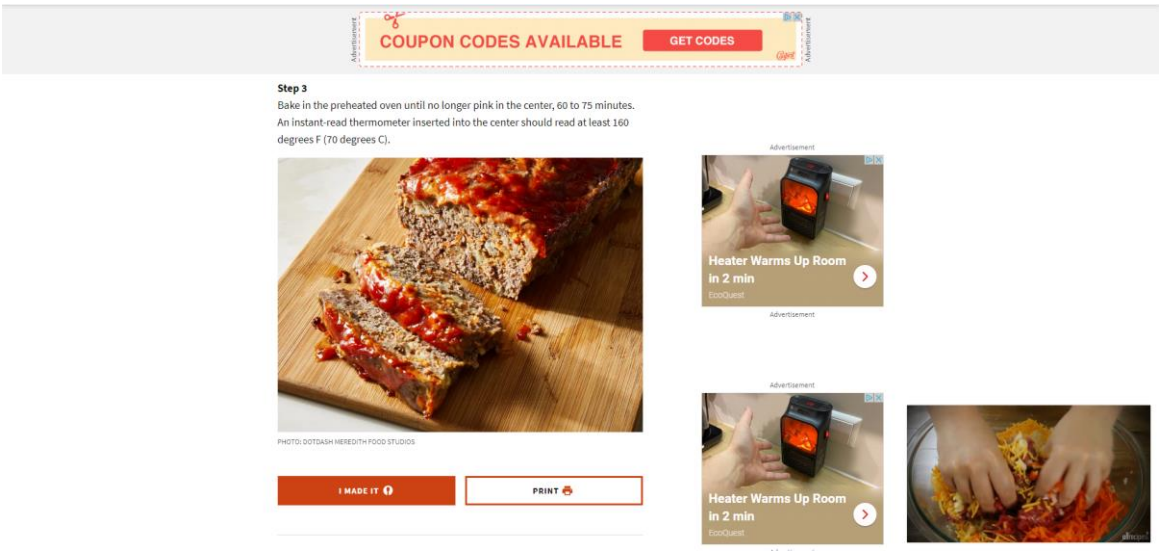


FIG 2.5 – ALLRECIPES ADVERTISEMENTS

- There can be a lot of variation in the recipes posted by users, leading to an inconsistent experience for users, if some users do not put the same amount of effort into their recipe.

Food – https://www.food.com/

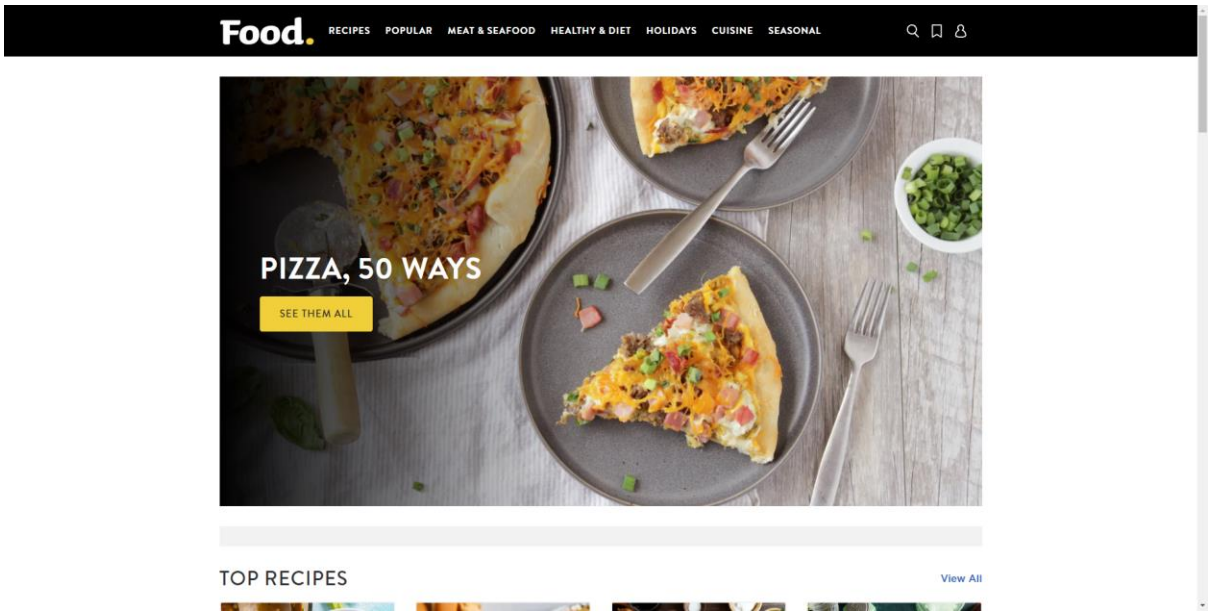


FIG 3.0 – FOOD.COM HOMEPAGE

Food.com is a recipe-sharing website that offers a variety of recipes for different meals and occasions. The site allows users to search and filter recipes by various criteria, including ingredients, cuisine, and dietary needs. Users can save recipes, create shopping lists, and leave reviews on recipes. Food.com also features cooking tips, tutorials, and a community forum where users can share their experiences and knowledge. Additionally, the website offers a meal planner tool to help users plan and organize their meals.

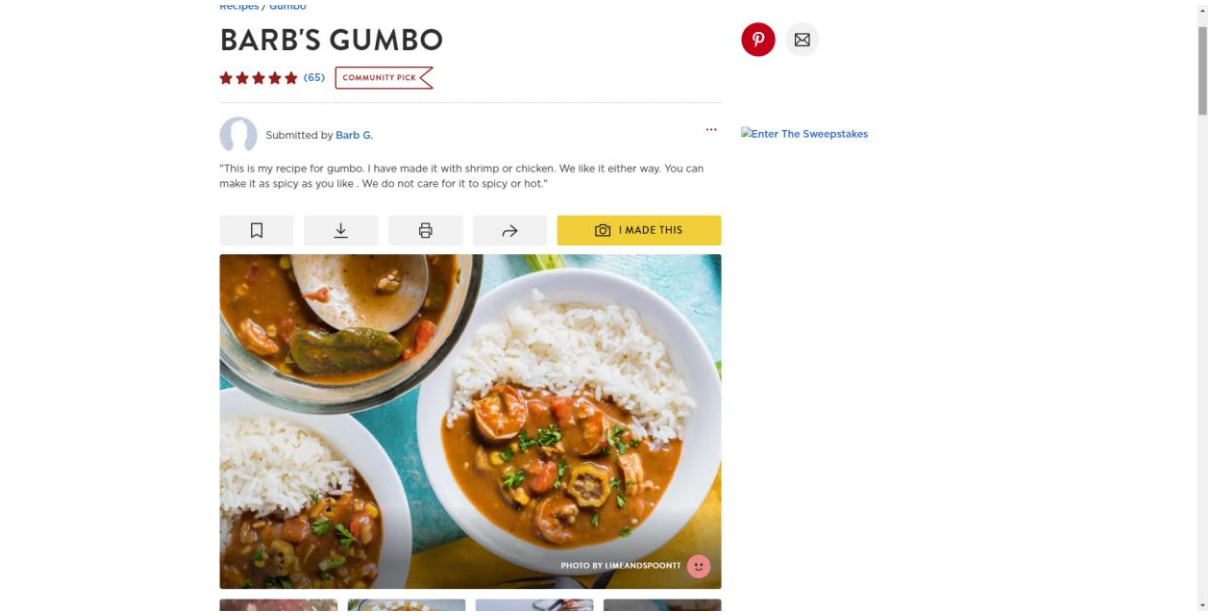


FIG 3.1 - FOOD.COM RECIPE PAGE

Advantages of Food.com

- Food.com has the ability to filter recipes.

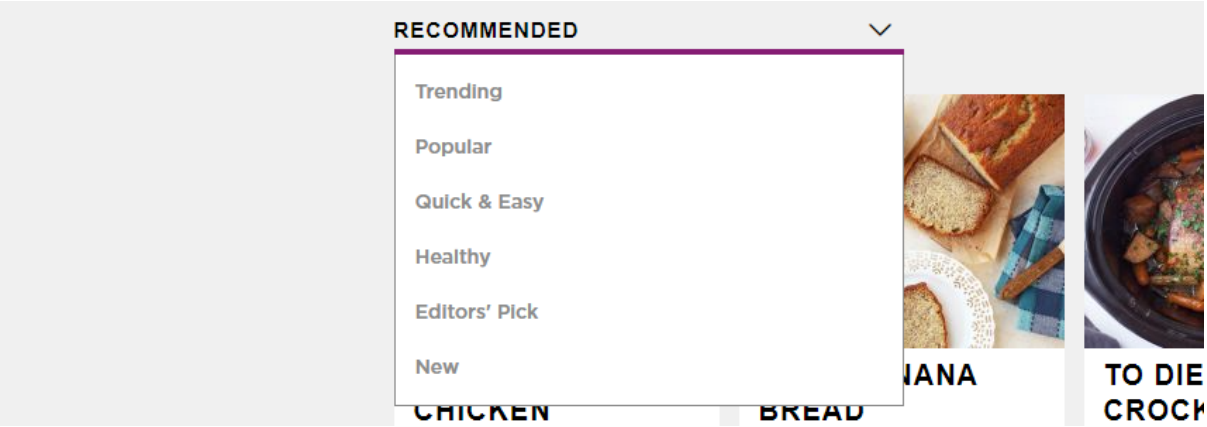


FIG 3.2 – FOOD.COM FILTERS

- Food.com offers a community forum and user reviews on recipes, which can provide valuable feedback and tips from other users.

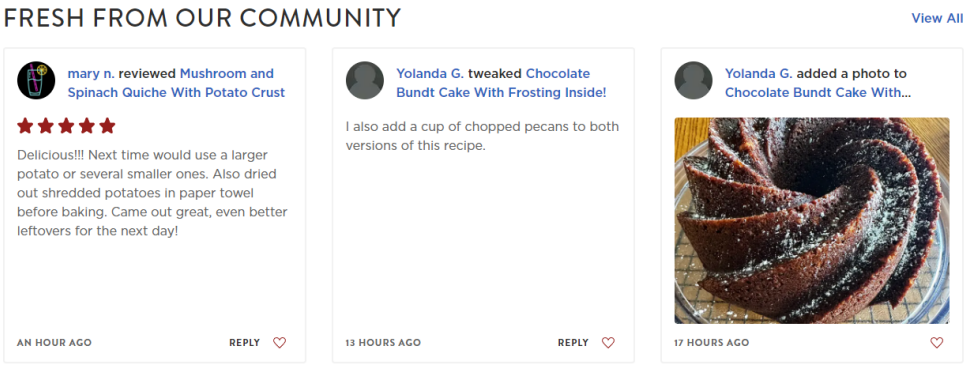


FIG 3.3- FOOD.COM REVIEW SECTION

Disadvantages of Food.com

- Food.com includes some nutritional information on its recipes, but often is not as detailed or as easy to notice where it is, as other recipe websites or apps.

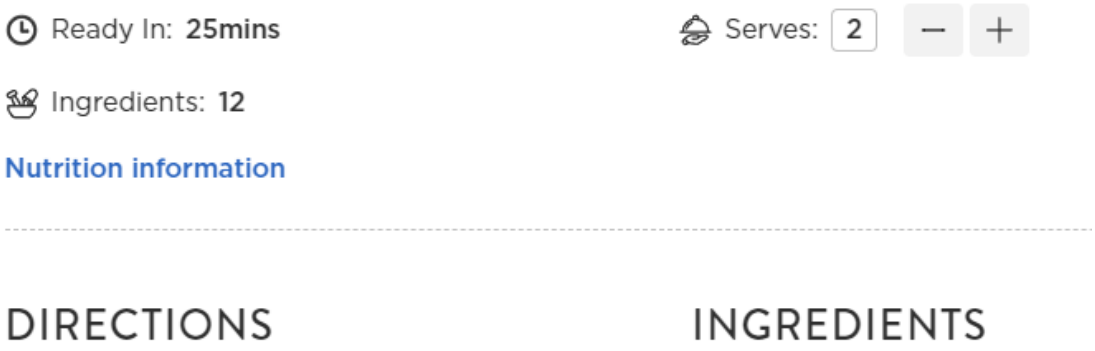


FIG 3.4 – FOOD.COM RECIPE INFORMATION

- While users can filter recipes, the website doesn't offer personalized recommendations based on a user's preferences or browsing history, which could be a drawback for some users.

3.2.2 Interviews

The questions asked in the interviews are as follows:

- Can you tell me about your experience with meal planning and cooking?
- How do you currently find new recipes?
- Are specific dietary requirements something you would be looking out for when deciding on a recipe to follow?
- Can you tell me about any specific features or functionalities that you would like to see in a recipe-sharing web application?
- How important is the ability to save and keep track of your favourite recipes to you?
- Do you like to get and give feedback on the meals you cook or find?

The four interviews that were conducted revealed several common themes and preferences for a recipe-sharing web application. All four users expressed a need for a search function that would allow them to filter recipes by different categories, such as ingredients, dietary restrictions, and cook time. They also valued the ability to save and organize their favourite recipes and leave comments or ratings on other users' recipes. Additionally, some users, highlighted the importance of having high-quality photos to accompany the recipes to help guide them in the cooking process.

The interviews also revealed some unique insights based on each interviewee's background and experiences. For example, one user expressed a preference for recipes that were adaptable to different skill levels, which would make the application more user-friendly for beginners. This insight highlights the need for user-cantered design and the importance of considering the diverse needs of the target audience when designing a recipe-sharing web application.

Overall, the interviews demonstrate the importance of creating a platform that provides a seamless and enjoyable experience for users. By understanding the needs and preferences of the target audience, developers can create a web application that is intuitive, user-friendly, and capable of meeting the diverse needs of its users. The insights gathered from these interviews can inform the feature and functionality decisions of the web application, and ultimately create a platform that is highly valued by its users.

3.3 Requirements modelling

3.3.1 Personas

1. "Samantha" is a 35-year-old stay-at-home mom with two young children aged 5 and 8. She is always looking for new and healthy recipes to cook for her family but doesn't have a lot of time to spend on meal planning. She wants a web application that is easy to use and allows her to quickly search for recipes that meet her dietary restrictions. She follows a gluten-free diet and tries to keep her meals low in sugar. She wants to filter and find recipes that meet her dietary needs and also has a time constraint feature that allows her to filter recipes that can be cooked under 30 minutes. She wants to be able to save her favourite recipes and also get feedback from other users on the meals she cooks for her family.

FIG 4.0 - PERSONA #1

2. "Mike" is a 22-year-old college student who is just learning how to cook. He is looking for a web application that has a wide variety of recipes, including beginner-friendly options. He wants to find recipes that are easy to follow with simple instructions and pictures. He wants the application to have a feature that allows him to save his favourite recipes and also get feedback from other users on the meals he cooks. He wants to be able to filter recipes by cuisine, dietary restrictions, and difficulty level. He also wants to be able to leave comments on other users' recipes and rate them.

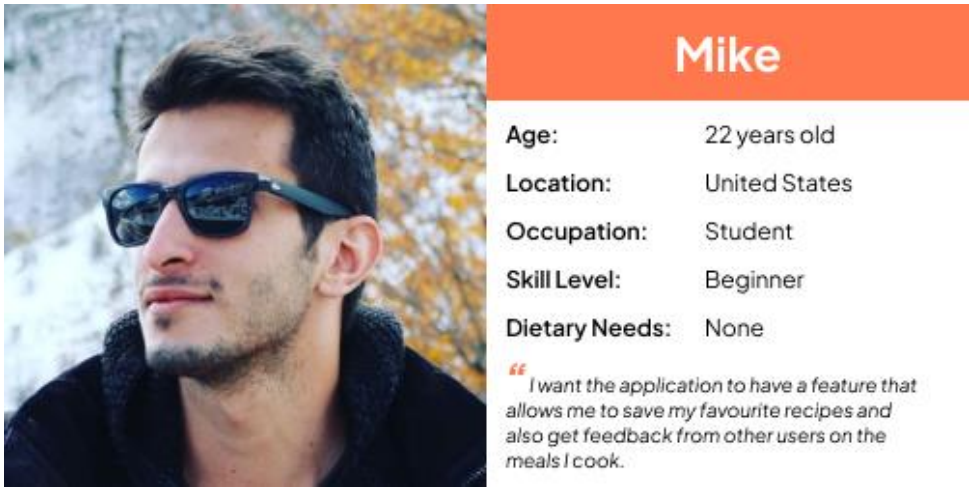


FIG 4.1 – PERSONA #2

3. s a 32-year-old professional chef who wants to share her recipes with a wider audience. She wants a web application that allows her to easily upload and organize her recipes, and that has a community of users who are passionate about food. She wants to be able to upload her recipes with detailed instructions, pictures, and a video of her cooking the recipe. She wants to be able to get feedback from other users on her recipes and also rate other users' recipes. She wants to be able to filter recipes by cuisine, dietary restrictions, and difficulty level.

FIG 4.2 - PERSONA #3

4. "David" is a 65-year-old retiree who is an experienced cook and enjoys experimenting with new recipes. He wants a web application that has a large collection of recipes from around the world and that allows him to easily search for recipes based on ingredients, cuisine, and dietary restrictions. He follows a low-carb diet and wants to be able to filter recipes that meet his dietary needs. He also wants to be able to save his favourite recipes and leave comments on other users' recipes. He wants to be able to rate recipes, and also filter recipes by difficulty level, cook time, and number of servings.

FIG 4.3 – PERSONA #4

3.3.2 Functional requirements

The functional requirements for this project are as follows:

#	Functional Requirement	Priority
1	As a user, I want to be able to view recipes on the application.	High
2	As a user, I want to be able to register and log into an account on the application.	High
3	As a member, I want to be able to create, edit, and delete my own recipes on the application.	High
4	As a member, I want to be able to bookmark recipes I like.	High
5	As a member, I want to be able to comment on other users' recipes.	High
6	As a user, I want to be able to search for recipes on the application by name, category, and ingredients.	Medium
7	As a user, I want to be able to view the most recent recipes added to the application.	Medium
8	As a user, I want to filter recipes by popularity, dietary needs, etc.	Medium
9	As a user, I want to be able to change the measurements used to the ones I would use, e.g., teaspoons instead of grams.	Medium
10	As a member, I want to get notifications when other users comment on my recipes.	Low

3.3.3 Non-functional requirements

The non-functional requirements for this project are as follows:

#	Non-Functional Requirement	Classification
1	The application should be responsive with multiple screen sizes.	Usability
2	The application should have a FAQ page.	Usability
3	You should be able to filter recipes by newest, oldest, and rating.	Usability
4	Save unfinished recipes as a draft in order to finish creating it later.	Usability
5	You must be logged in to bookmark recipes.	Security
6	You must be logged in to comment on recipes.	Security
7	You must be logged in to create a recipe and be the author to delete it.	Security
8	Members should be able to change their password.	Security

3.3.4 Use Case Diagrams

This recipe sharing application will consist of three different types of users, that include: User, Member and Administrator.

A User should be able to:

- Register for an account.
- View all recipes & categories on the application.

A Member should be able to:

- Log in & out of the application.
- View all recipes & categories on the application.
- Create their own recipes.
- Edit the recipes they have created.
- Delete recipes they have created.
- Comment on recipes submitted to the application.
- Bookmark recipes they like that were submitted to the application.
- Update their personal information, including their name, date of birth, email & password.

An Administrator should be able to:

- Everything a member can do.
- View the admin dashboard.
- Manage all of the content on the application.

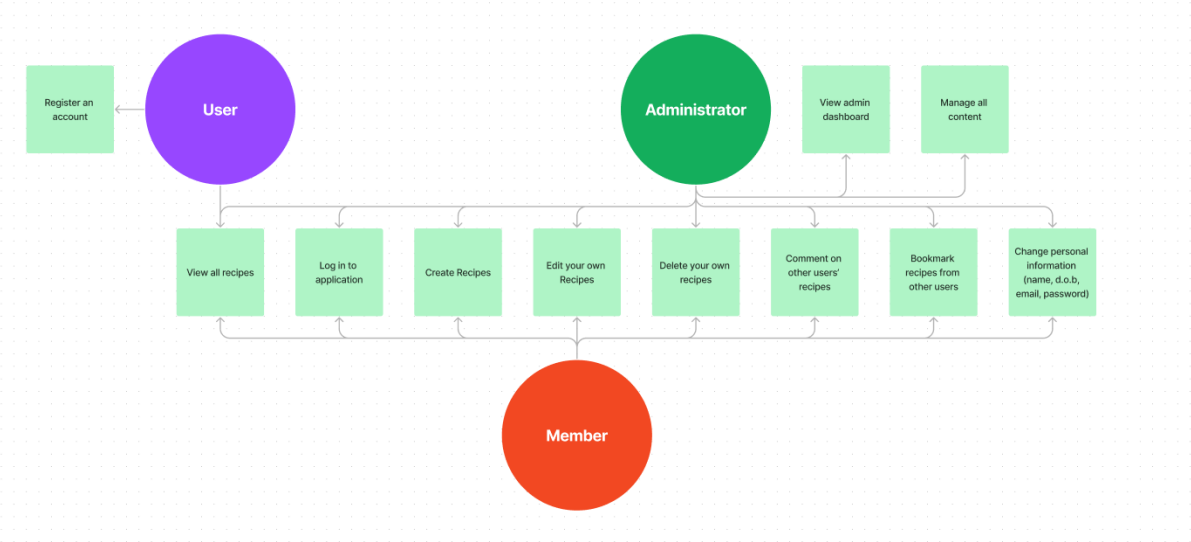


FIG 5.0 - USE CASE DIAGRAM

3.4 Feasibility

Listed below are the technologies that have been chosen for the development of the recipe sharing application:

- React / Svelte – Frontend
- Express - Backend
- MongoDB - Database

All of React, Svelte and Express, along with mongoose for connecting to MongoDB database have documentation on their websites, which will guide developers through the development of their applications.

3.5 Conclusion

The requirements chapter provides the developers a better understanding of the people they wish to become users of the application, which in turns helped to determine all of the requirements, both functional and non-function for the web application being created. It helps to get a better idea of the goals that the developer will want for the application.

Getting user input by creating personas and conducting interviews helps to give the developer an idea of where their focus should be when creating the application and the most important features that should be focused on, aside for the basic functionality that the application will need to function.

4 Design

4.1 Introduction

The purpose of the design phase of the project is to allow for developers to arrive at a design for the application so that the application meets the requirements for the application as set out in the Requirements chapter.

The application under development is a web-based platform that allows users to access an extensive collection of food recipes. The primary objective of the application is to enable users to search and find recipes that cater to their dietary requirements, food preferences, and more. The target audience for this application is individuals of all ages who possess an interest in cooking and baking and are looking for a convenient way to access a diverse range of recipe options.

To meet the diverse needs of its users, the application design incorporates several features, including a search function that allows users to filter recipes based on their preferences, an intuitive user interface that simplifies recipe browsing and selection. By leveraging these features, the application aims to provide a convenient, accessible, and comprehensive recipe-finding solution for users of all skill levels and backgrounds.

4.2 Program Design

Below is the Entity Relationship Diagram (ERD) for the web application that has been created. It shows each of the tables, Users, Recipes, Categories, and Bookmarks.

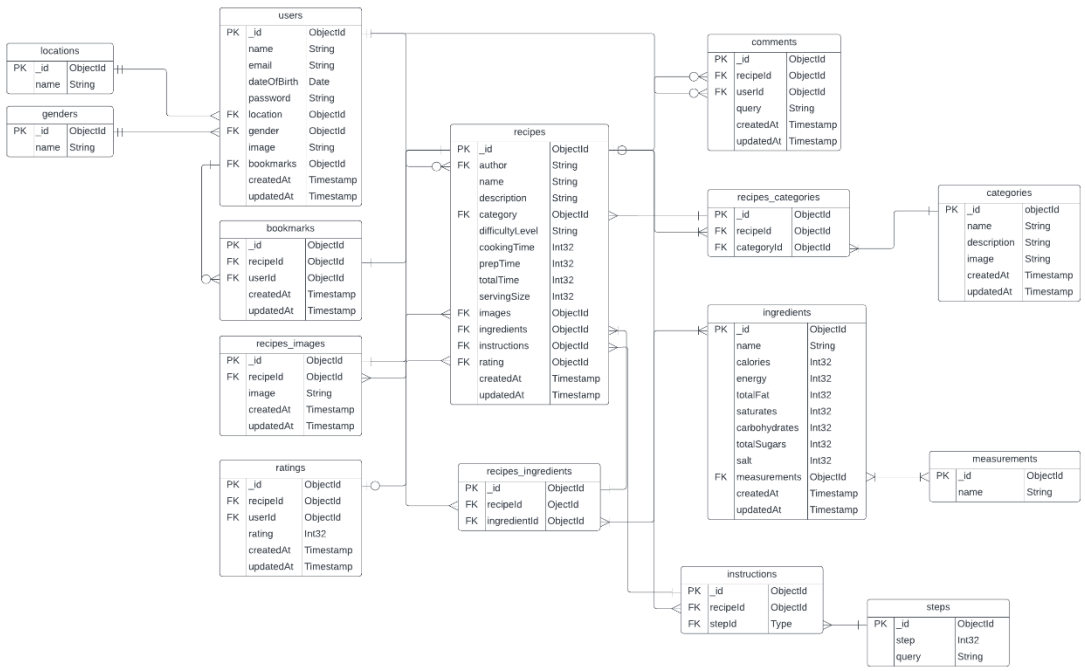


FIG 6.0 - ENTITY RELATIONSHIP DIAGRAM

4.2.1 Technologies

The technologies that will be employed in creating the application are the following:

- ReactJS
- Axios
- Express.js
- MongoDB

React.js, a front-end JavaScript library, was selected due to its ability to enhance the development process with features such as search and filter functions. The Axios library for HTTP communication will be used in conjunction with React to retrieve and post data between the front and back ends.

The Express.js back-end framework was chosen due to its reputation as a lightweight framework and its status as a widely adopted standard server for Node.js. The Express back end will communicate with the NoSQL database, MongoDB, and interface with React, the front end of the website via Axios. MongoDB was selected as the database for the project due to its use of JSON-like documents, which are well-suited to the project's needs.

While alternative technologies such as Laravel, Bootstrap, and MySQL were available, the decision was made to employ the MERN stack for this project, leveraging MongoDB, Express.js, React.js, and Node. The choice to forego Laravel and MySQL was made based on the perceived advantages of the MERN stack in the context of this particular use case.

4.2.2 Structure of React & Express (2 pages)

Describe the structure of whichever technology you are using, for instance the various folders inside of Laravel, the use of routes controllers and views. Include diagrams.

4.2.3 Design Patterns

This may apply to your project. For instance, Laravel is based on the Model View Controller (MVC) Design pattern.

4.2.4 Application architecture

The application architecture provides a framework for managing interactions between the frontend and the backend of the application. The frontend is responsible for rendering the application's content through HTML/CSS and JavaScript, while the backend utilizes the NodeJS framework, Express.js, to handle server-side functionality.

To understand how the application works, consider the example of a user attempting to view the dashboard. If the user is not logged in and tries to access the edit recipe form, the frontend will communicate with the backend through the appropriate route. The request will then be processed by the middleware, which will check if the user is logged in. Since the user is not logged in, the middleware will return an error message to the frontend. However, if the user is logged in and attempts to access the edit recipe form, the middleware will allow the request to pass through to the controller, which will then interact with the model to retrieve the appropriate data from the database. The resulting data will be passed back to the controller, which will then display the contents of the edit recipe form to the user through the frontend.

Overall, the application architecture serves as a crucial framework for ensuring efficient communication between the frontend and backend of the application. By effectively managing requests and responses between these two components, the application can provide users with a seamless and intuitive experience.

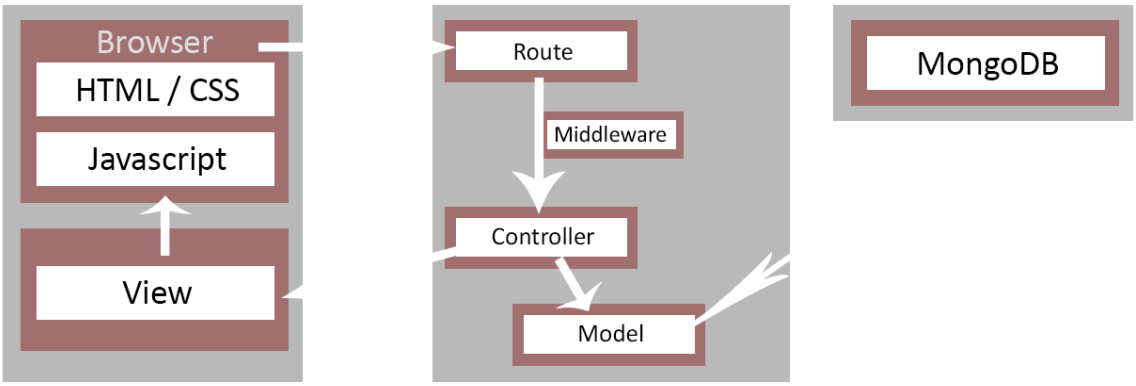


FIG 7.0 - APPLICATION ARCHITECTURE DIAGRAM

4.2.5 Database design

Entities	Attributes
Users	_id, name, email, password, location(fk), gender(fk), bookmarks(fk), role, createdAt, updatedAt
Recipes	_id, name, author(fk), description, image_path, category(fk), ingredients(fk), instructions, cookingTime, prepTime, totalTime, rating(fk), createdAt, updatedAt
Categories	_id, name, description, image_path, createdAt, updatedAt
Bookmarks	_id, recipeId(fk), userId(fk), createdAt, updatedAt
Comments	_id, recipeId(fk), userId(fk), comment, createdAt, updatedAt
Ratings	_id, recipeId(fk), userId(fk), rating, createdAt, updatedAt
Measurements	_id, name, abbreviatedName
Locations	_id, name
Genders	_id, name

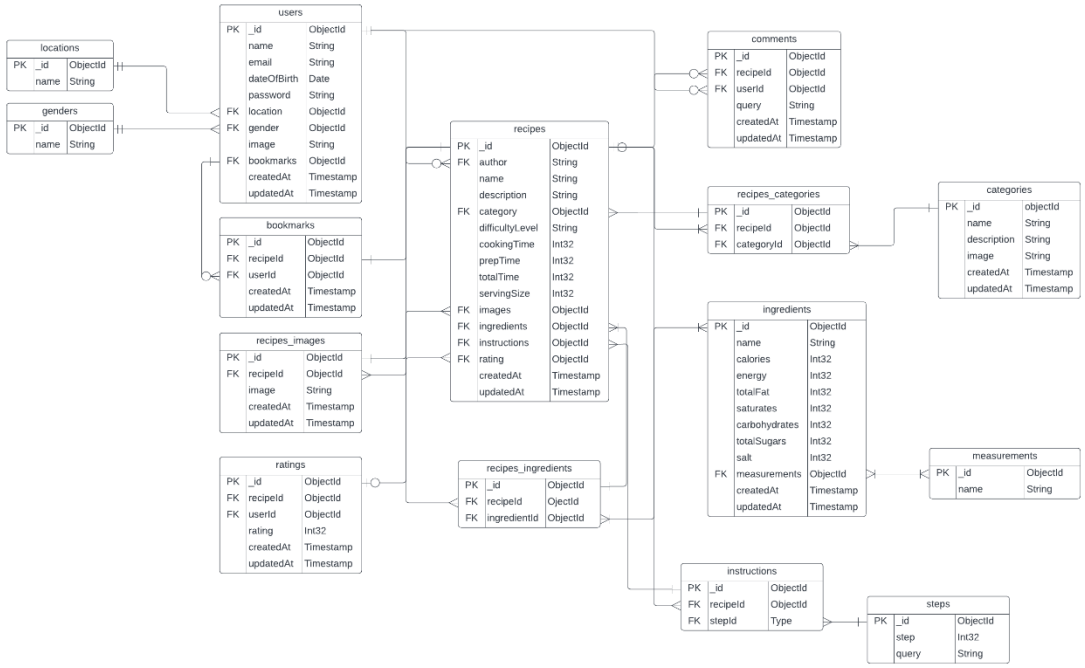


FIG 8.0 – ENTITY RELATIONSHIP DIAGRAM

4.3 User interface design

4.3.1 Wireframe

Below is the wireframe showing the layouts for the following pages: Home, Show recipe, Show all recipes, show all/specific categories.

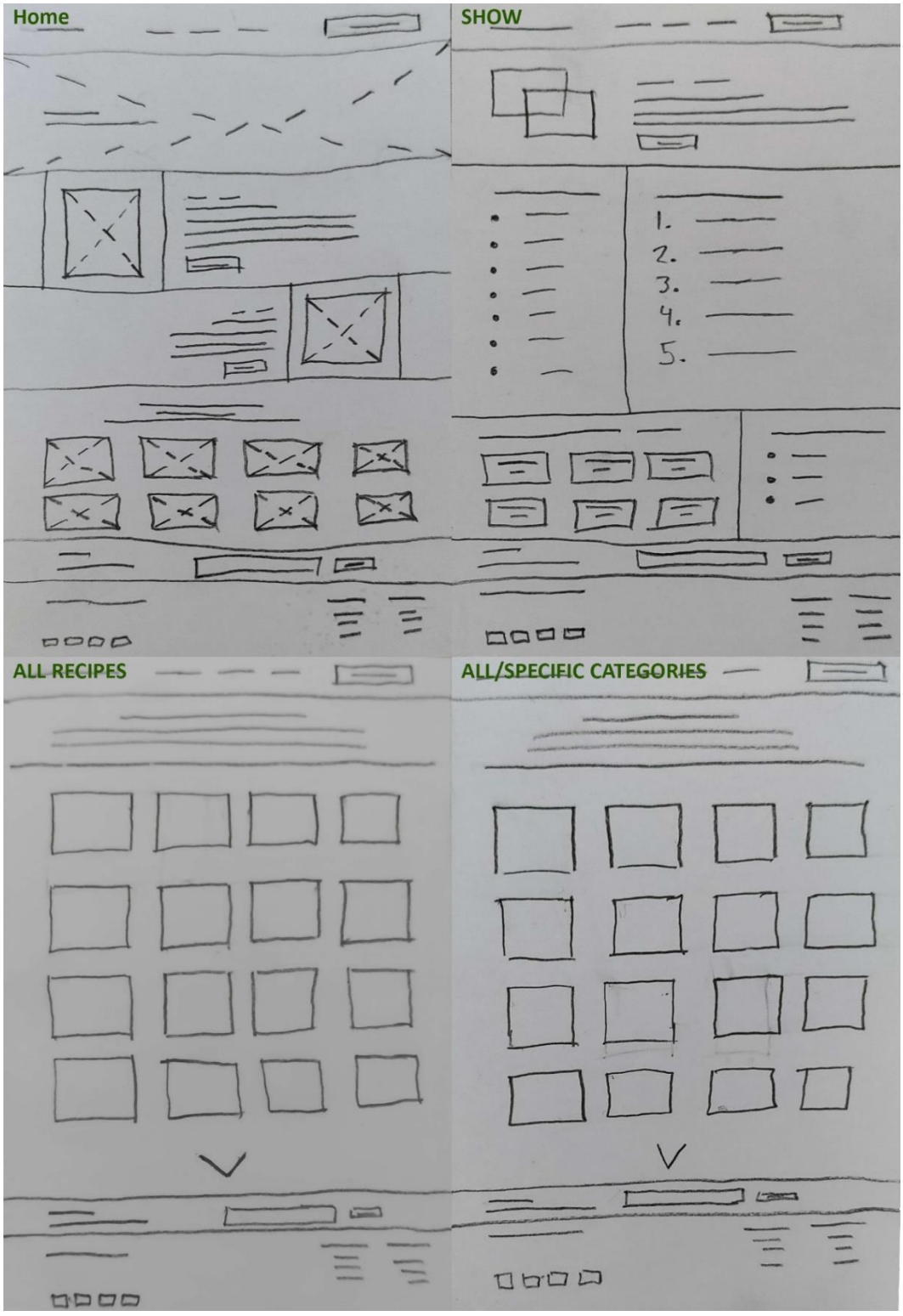


FIG 8.0 - WIREFRAME

4.3.2 User Flow Diagram

The user flow diagrams were designed using a FigJam file in figma. The diagram illustrates the navigation process of users within the application, whether or not they log in to the web application. The diagram provides an overview of how the user will transition from one page to another and serves as a useful tool to analyse the user's journey throughout the application.

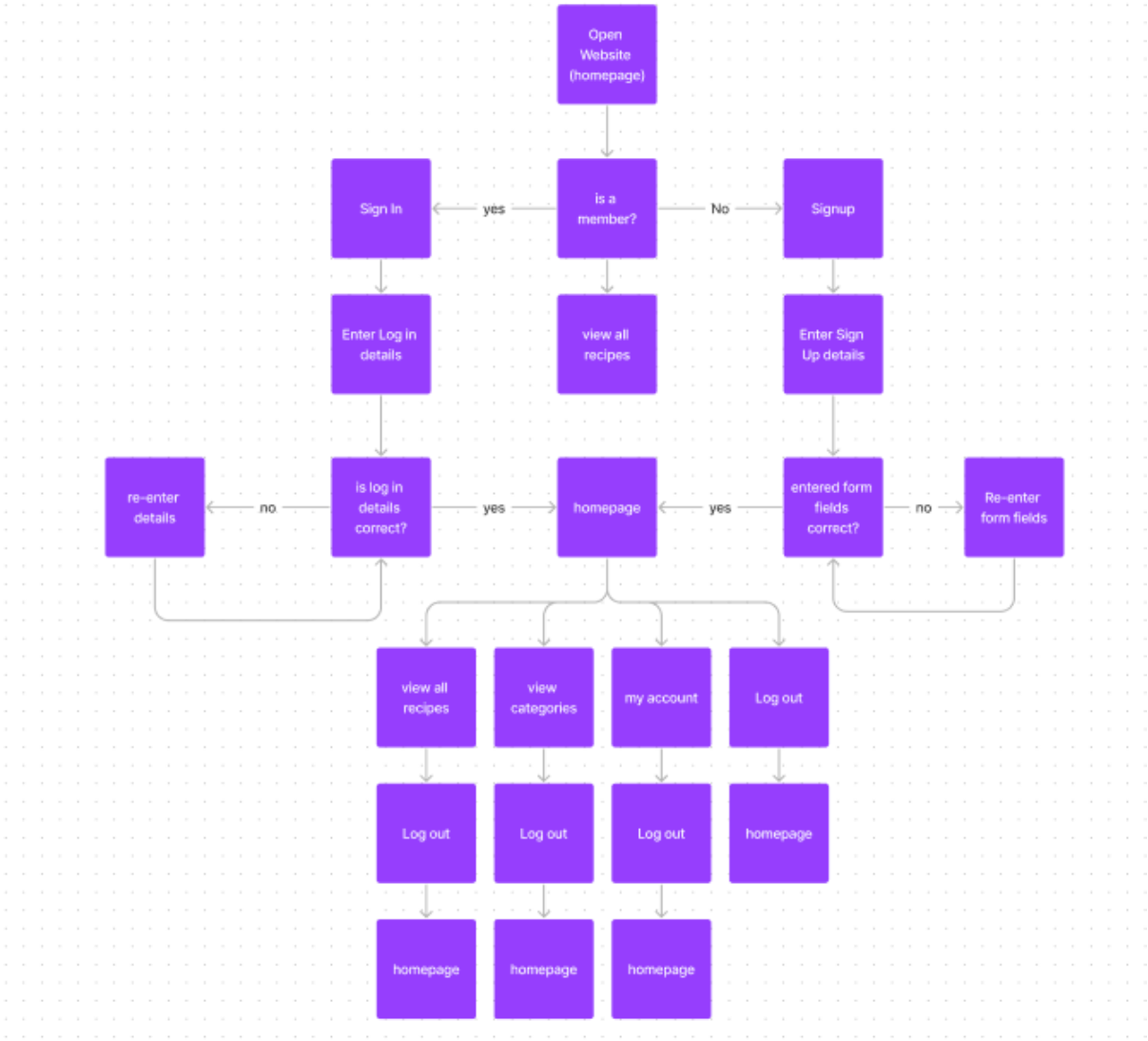


FIG 9.0 - USER FLOW DIAGRAM

4.3.3 Style guide

Colour Palette

To maintain the design's overall quality and avoid complexity, the colour palette for this application has been deliberately kept simple. Adding additional colours may hinder the understanding of the design from both a user and design perspective, hence it was decided to use a limited range of colours.

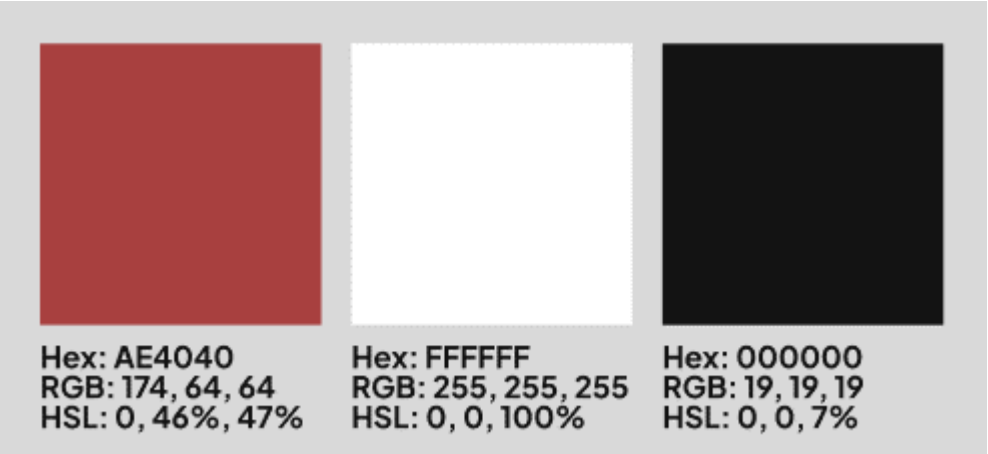


FIG 10.0 - COLOUR PALLET

Typography

The selected font family for this application is Plus Jakarta Sans, which is a free sans-serif font readily available on Google Fonts, featuring seven distinct font weights and their corresponding italic variations. This font was chosen for its suitability to the overall design aesthetic and its readability on various screen sizes and resolutions.

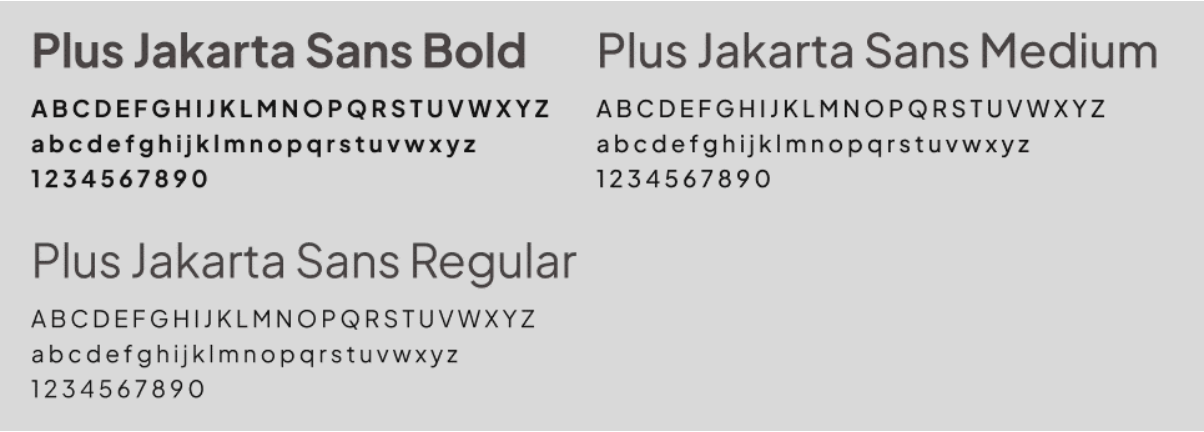


FIG 10.1 - EXAMPLES OF TYPOGRAPHY

FIG 10.2 - EXAMPLES OF TYPOGRAPHY

Buttons

The primary button in this application will have a medium font weight and white text colour, with the accent colour serving as the background colour. The button will feature a larger spacing number of 24px / 1.5rem for padding on the left and right, and a medium spacing number of 16px / 1rem for vertical padding.

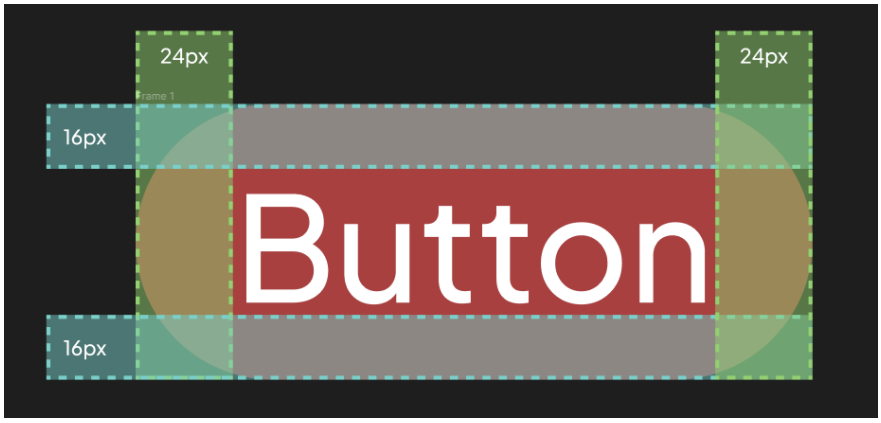


FIG 10.3 - BUTTON DESIGN

Card

The recipe card component will be designed with a consistent padding of 24px surrounding the border, while maintaining the visual appeal and readability through the use of a smaller 16px padding between each component in the content of the card, except for the tags and recipe information which will be separated by a smaller gap of 8px. Additionally, an image showcasing the fully prepared meal on which the recipe is based will not only add to the aesthetic appeal but also contribute to a more engaging user experience.

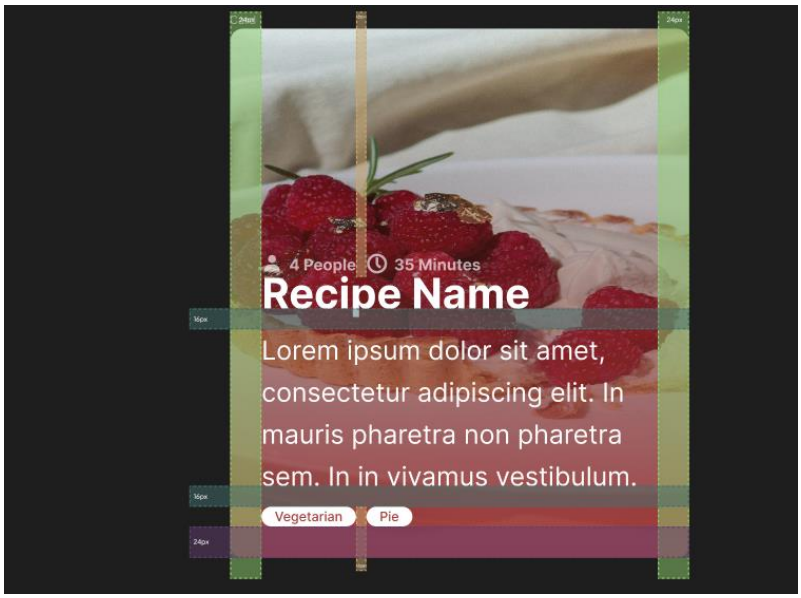


FIG 10.4 - CARD DESIGN

Forms Elements

In the form design, there is a deliberate separation of different sections by a 24px gap. However, for the elements that are interrelated such as the label and the input, they are only separated by a smaller gap of 16px. This decision was made to visually indicate the correlation between these elements.

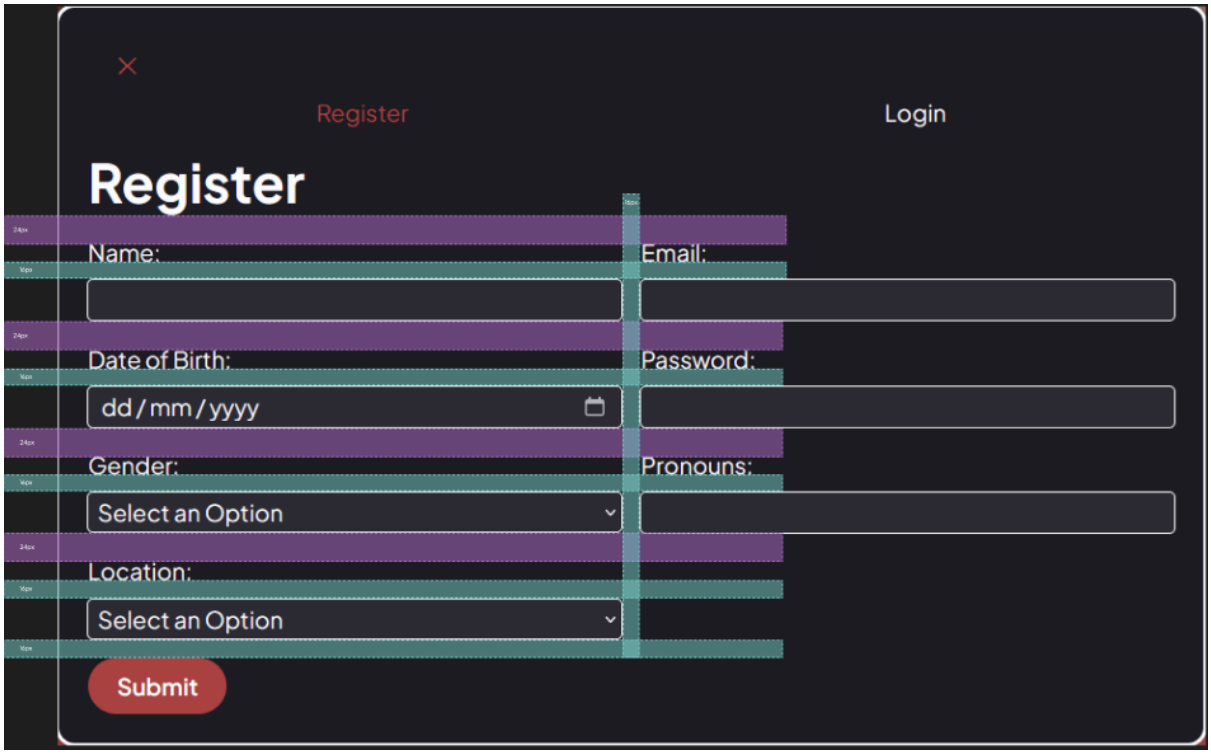


FIG 10.5 - MODAL DESIGN

4.4 Conclusion

In the Design chapter, various aspects of the design are elaborated upon. The program design mainly encompasses the Entity Relationship Diagram, which showcases the relationships between different tables in the database. The application architecture shows how the web application functions and the communication between its frontend and backend components. The user flow diagram highlights how the user can navigate the site, whether logged in or out. Finally, the report discusses the style guide, which outlines the visual aspects of the application such as typography and the colour palette.

5 Implementation

5.1 Introduction

In the implementation chapter, the focus will be on how the application will be built, the methodology that will be used, and the development environment that will be used. As a sole developer of this project, it is important to understand the plan that will be followed, the tools and technologies that will be utilized, and the timeline for project. By providing detailed information on the implementation process, this will ensure that the project is completed efficiently and effectively. The application will be developed using the following technologies:

ReactJS – Frontend framework

ReactJS is a JavaScript library used for building user interfaces. It allows for the creation of reusable UI components and uses a virtual DOM for efficient rendering. It was developed by Facebook and is currently maintained by them and a community of developers.

ExpressJS – Backend framework

Express.js is an open-source back-end web application framework for Node.js. It is a fast and minimalistic framework, that provides a robust set of features.

MongoDB - Database

MongoDB is a cross-platform document-orientated database that is designed to be scalable and flexible. MongoDB uses JSON-like documents and is known as a NoSQL database program.

The application for this project is a recipe sharing website, that users can create and edit their recipes, find other recipes that they can try. As well they can bookmark recipes they enjoy and give feedback via commenting on the recipes themselves.

5.2 Implementation Roles

As the sole developer, all roles in the implementation process will be undertaken by the same person. These roles include the project manager, web designer and developer and finally the deployment engineer. The project manager will be responsible for setting project goals and timelines, while the web designer and developer will be responsible for the design and development of the web application and the deployment engineer will be responsible for deploying the application to the hosting environment.

5.3 SCRUM Methodology

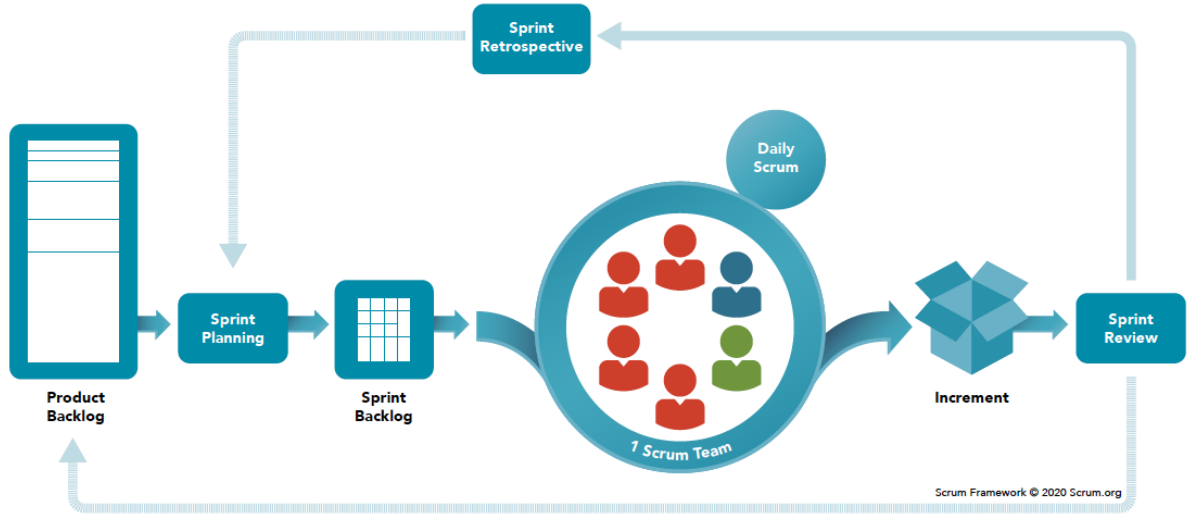


FIG 11.0 - SCRUM METHODOLOGY

The Scrum methodology will be used for the development process of this project. This methodology follows an iterative and incremental approach to software development and is particularly well suited to solo developers. The Scrum framework involves the use of sprints, which are short periods of time in which specific development goals are set. This approach helps to ensure that the project is completed efficiently, with regular feedback and evaluation.

5.4 Development environment

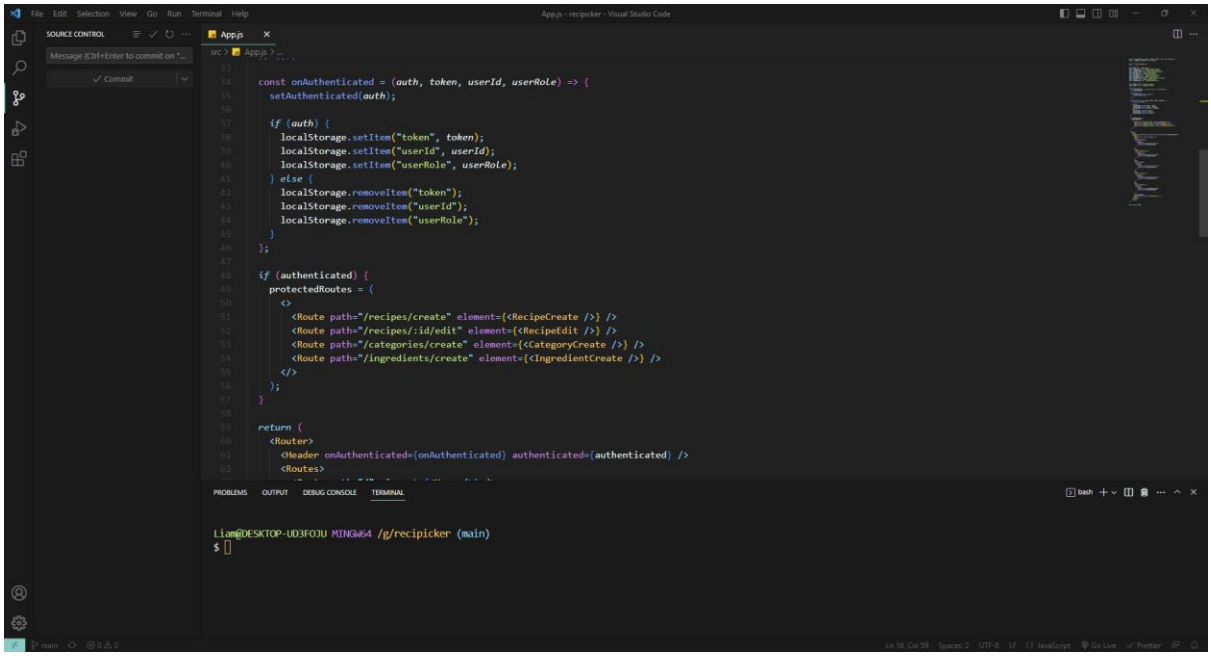


FIG 12.0 - DEVELOPMENT ENVIRONMENT

The development environment used for the project was Microsoft's Visual Studio Code, which was chosen due to its versatility as a lightweight and powerful code editor available across various platforms. This decision ensured that the project could be accessed on any operating system without compromising on functionality.

To streamline project updates, a GitHub repository was used as it is the industry standard version-control platform. This allowed for easy access to the latest version of the project in different locations, such as at home, or on the college campus. The use of version control also helped to track and manage changes made to the project over time, facilitating troubleshooting and bug fixes.

The combination of Visual Studio Code and GitHub provided a powerful development environment that allowed for efficient and organized project management, enabling the project to be easily maintained and updated throughout its development cycle.

5.5 Sprint 1

5.5.1 Goal

The goal for Sprint 1 will be to research similar applications to the recipe sharing application that have been created, and getting an understanding of the types of features and information users look for. As well as this, the design of the application will be finalised in a hi-fi prototype of the application, that will be created in Figma.

5.5.2 Item 1

- Research similar applications.
- Decide on the features that will be implemented:
    - Search for recipes / categories
    - Create, Read, Edit, Delete Recipes / Categories / Ingredients.
    - Upload Recipe / Category images
    - Bookmark recipes
    - Comment on recipes
    - Measurement conversion
    - Nutritional Information

5.5.3 Item 2

- Finalise the design of the recipe sharing application.
- Create a hi fidelity prototype in figma.

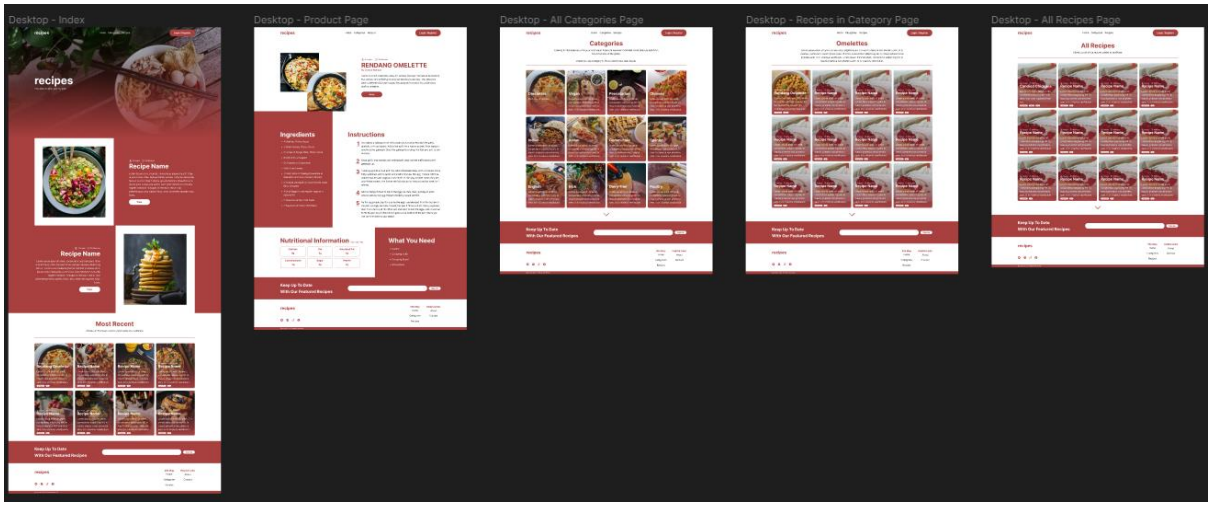


FIG 13.0 - HI FIDELITY PROTOTYPE IN FIGMA

5.6 Sprint 2

5.6.1 Goal

The goal of Sprint 2 will be to create and style the user registration and login forms and implement the functionality. The output will be a working registration and login system.

5.6.2 Item 1

- Create and style the registration and login forms, then implement functionality.
- Create and style the modal, then implement functionality.

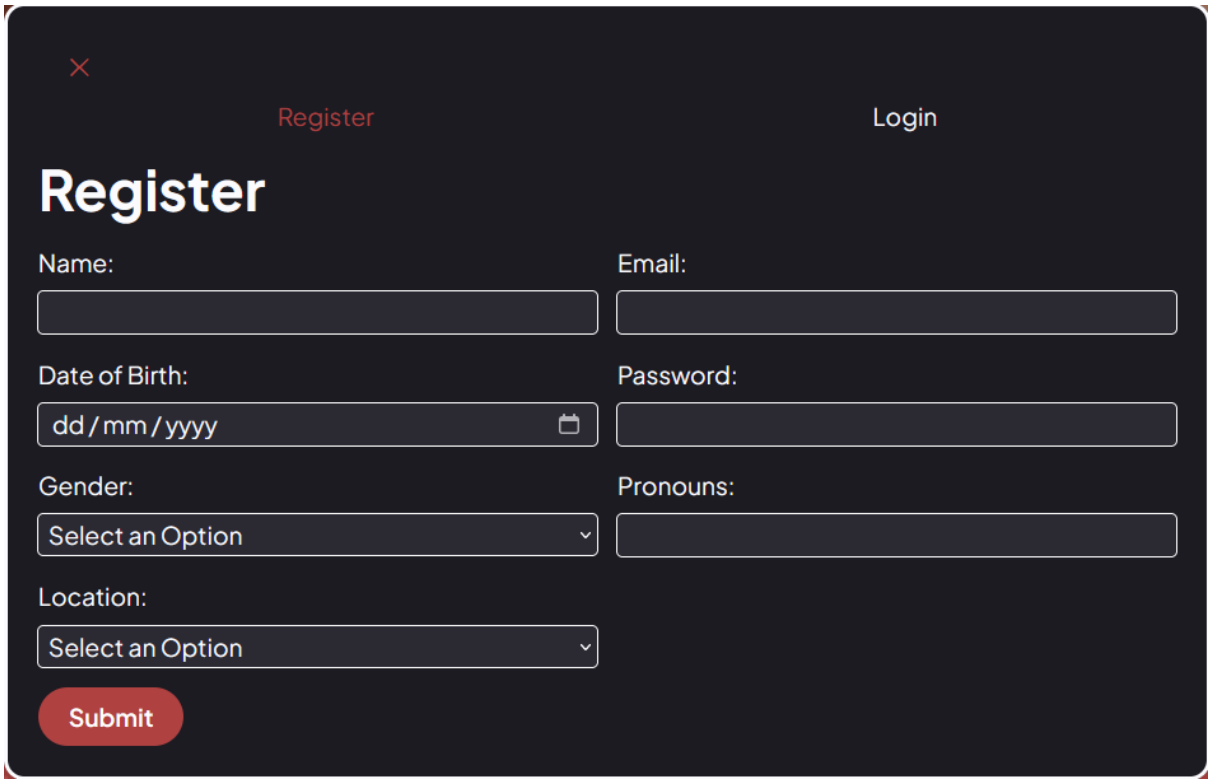
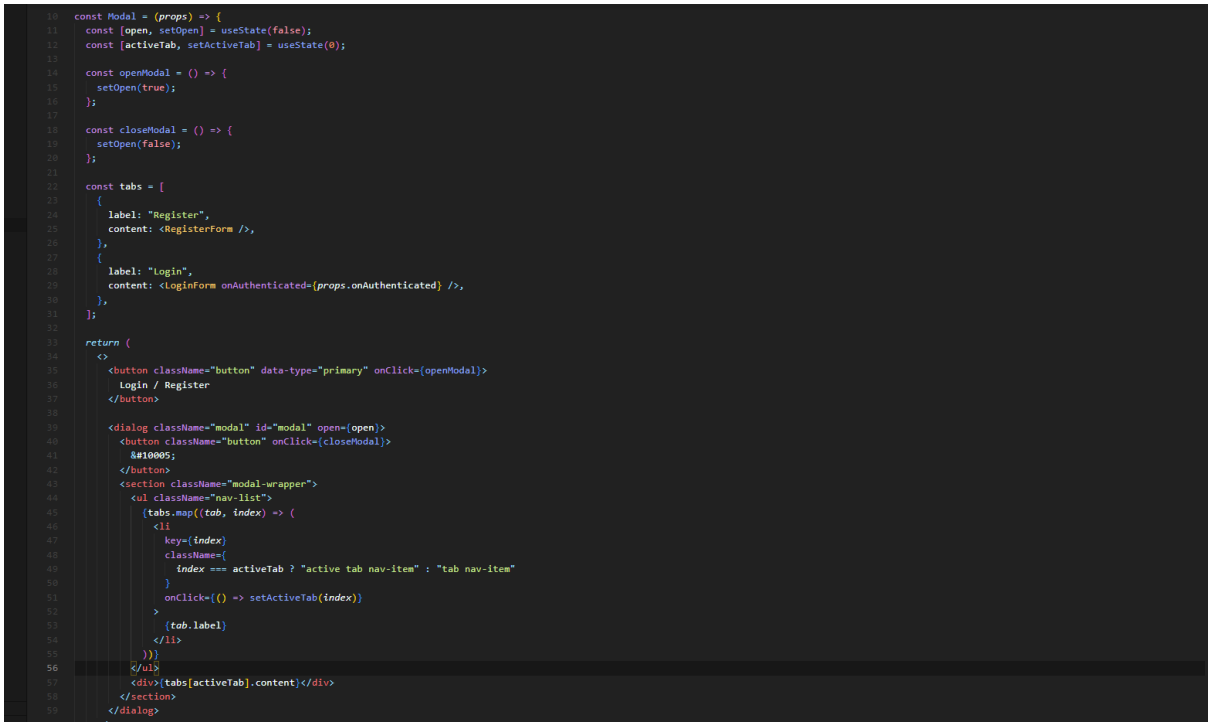


FIG 14.0 - REGISTRATION / LOGIN MODAL

5.6.3 Item 2

- Implement modal functionality.
    - Open/Close Modal.
    - Tabs.



```jsx
const Modal = (props) => {
  const [open, setOpen] = useState(false);
  const [activeTab, setActiveTab] = useState(0);

  const openModal = () => {
    setOpen(true);
  };

  const closeModal = () => {
    setOpen(false);
  };

  const tabs = [
    {
      label: "Register",
      content: <RegisterForm />,
    },
    {
      label: "Login",
      content: <LoginForm onAuthenticated={props.onAuthenticated} />,
    },
  ];

  return (
    <>
      <button className="button" data-type="primary" onClick={openModal}>
        Login / Register
      </button>

      <dialog className="modal" id="modal" open={open}>
        <button className="button" onClick={closeModal}>
          &#10005;
        </button>
        <section className="modal-wrapper">
          <ul className="nav-list">
            {tabs.map((tab, index) => (
              <li
                key={index}
                className={
                  index === activeTab ? "active tab nav-item" : "tab nav-item"
                }
                onClick={() => setActiveTab(index)}
              >
                {tab.label}
              </li>
            ))}
          </ul>
          <div>{tabs[activeTab].content}</div>
        </section>
      </dialog>
```

FIG 15.0 - MODAL.JS

- Implement registration form functionality.
    - Submit registration form.
    - Redirect to home after submission.

```
const submitForm = () => {
  axios
    .post("/users/register", {
      name: form.name,
      email: form.email,
      dateOfBirth: form.dateOfBirth,
      password: form.password,
      gender: form.gender,
      pronouns: form.pronouns,
      location: form.location,
      role: form.role,
    })
    .then((res) => {
      console.log(res.data);
      navigate("/");
      window.location.reload(true);
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 15.1 - REGISTER FORM FUNCTIONALITY

```
const login = (req, res) => {
  User.findOne({
    email: req.body.email,
  })
    .then((user) => {
      if (!user || !user.comparePassword(req.body.password)) {
        res.status(401).json({
          msg: "Authentication failed. Invalid email or password",
        });
      } else {
        let token = jwt.sign(
          {
            name: user.name,
            email: user.email,
            dateOfBirth: user.dateOfBirth,
            location: user.location,
            gender: user.gender,
            _id: user._id,
          },
          process.env.APP_KEY
        );

        res.status(200).json({
          msg: "Successfully Logged In!",
          token: token,
          userId: user._id,
          userRole: user.role,
        });
      }
    })
    .catch((err) => {
      throw err;
    });
};
```

FIG 15.2 - LOGIN FUNCTION IN THE RECIPICKER BACKEND

- Implement login form functionality.
    - Set onAuthenticated to true.
    - Pass token, userId and userRole to onAuthenticated to set in local storage.

```
const submitForm = () => {
  axios
    .post("/users/login", {
      email: form.email,
      password: form.password,
    })
    .then((res) => {
      // console.log(res.data);
      props.onAuthenticated(
        true,
        res.data.token,
        res.data.userId,
        res.data.userRole
      );
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 15.3 - LOGIN FORM FUNCTIONALITY

- Add the registration and login form components to the correct tabs in the modal.

5.7 Sprint 3

5.7.1 goal

The goal of Sprint 3 is to create the forms for creating the recipes, categories, and ingredients. To create the forms for editing the recipes. Then to implement the functionality for each of the forms.

5.7.2 item 1

- Create and Style the recipe create and edit form.

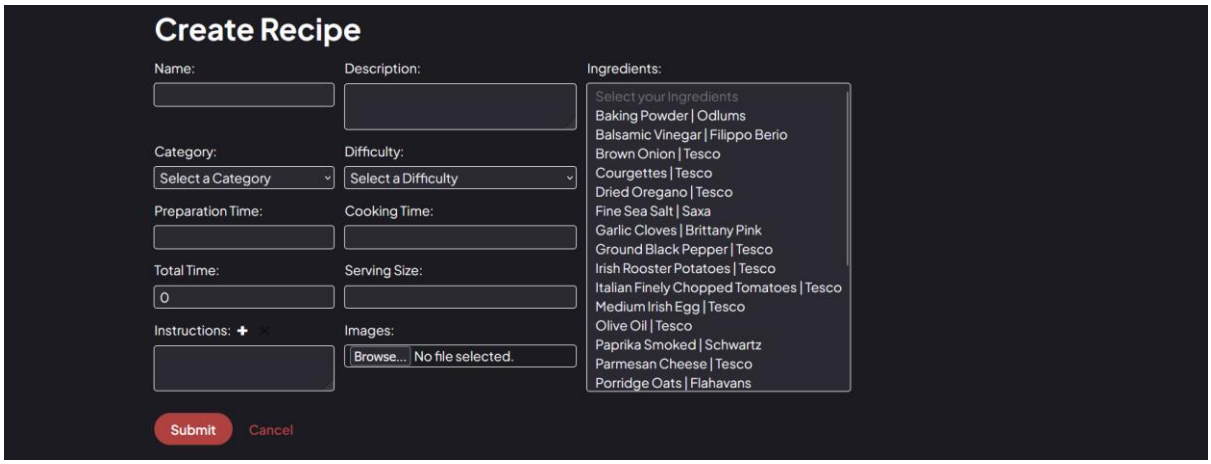


FIG 16.0 - CREATE FORM DESIGN

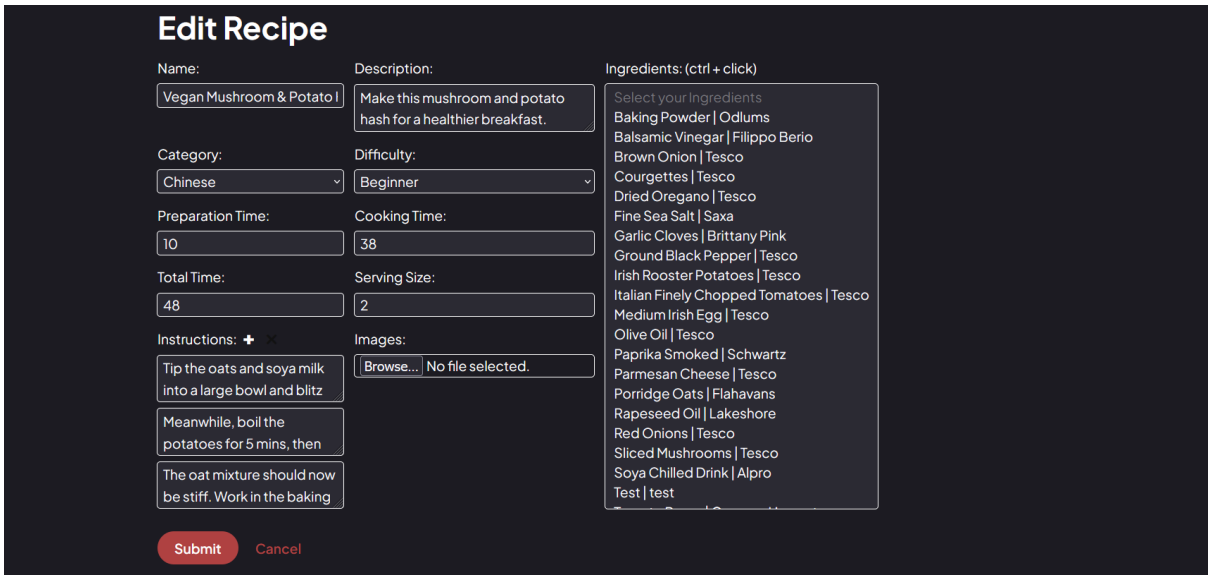


FIG 16.1 - EDIT FORM DESIGN

- Implement the recipe create and edit form functionality.

```
const submitForm = (e) => {
  e.preventDefault();
  const formData = new FormData();
  formData.append("file", form.file);

  let token = localStorage.getItem("token");

  axios
    .post(
      "/recipes",
      {
        author: form.author,
        name: form.name,
        description: form.description,
        category: form.category,
        difficulty: form.difficulty,
        cookingTime: form.cookingTime,
        prepTime: form.prepTime,
        totalTime: form.totalTime,
        servingSize: form.servingSize,
        file: form.file,
        image_path: form.image_path,
        ingredients: form.ingredients,
        instructions: form.instructions,
        // rating: form.rating,
        isFeatured: form.isFeatured,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
          "Content-Type": "multipart/form-data",
        },
      }
    )
    .then((res) => {
      console.log(res.data);
      navigate("/recipes");
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 16.2 - RECIPE CREATE FORM FRONTEND FUNCTIONALITY

```
const createData = (req, res) => {
  let recipeData = req.body;

  console.log(req.file);

  if (req.file) {
    recipeData.image_path =
      process.env.STORAGE_ENGINE === "S3" ? req.file.key : req.file.filename;
  }
  // include this else, if image required
  else {
    return res.status(422).json({
      message: req.imageError || "Image not uploaded!",
    });
  }

  Recipe.create(recipeData)
    .then((data) => {
      console.log("New Recipe Created!", data);
      res.status(201).json(data);
    })
    .catch((err) => {
      if (err.name === "ValidationError") {
        console.error("Validation Error!!", err);
        res.status(422).json({
          msg: "Validation Error",
          error: err.message,
        });
      } else {
        console.error(err);
        res.status(500).json(err);
      }
    });
};
```

FIG 16.3 - RECIPE CREATE BACKEND FUNCTIONALITY

```
const handleForm = (e) => {
  let name = e.target.name;
  let value = e.target.value;

  setForm((prevState) => ({
    ...prevState,
    [name]: value,
  }));
};

const addInstruction = (e) => {
  e.preventDefault();
  setInstructionCount(instructionCount + 1);
  setInstructionValues([...instructionValues, ""]);
};

const removeInstruction = (e) => {
  e.preventDefault();
  setInstructionCount(instructionCount - 1);
  setInstructionValues(instructionValues.slice(0, -1));
};

const submitForm = (e) => {
  e.preventDefault();
  const formData = new FormData();
  formData.append("file", form.file);

  axios
    .put(`/recipes/${id}`, form, {
      headers: {
        Authorization: `Bearer ${token}`,
        "Content-Type": "multipart/form-data",
      },
    })
    .then((res) => {
      // console.log(res.data);
      navigate("/recipes");
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 16.4 - RECIPE EDIT FORM FUNCTIONALITY

5.7.3 item 2

- Create the category create form.

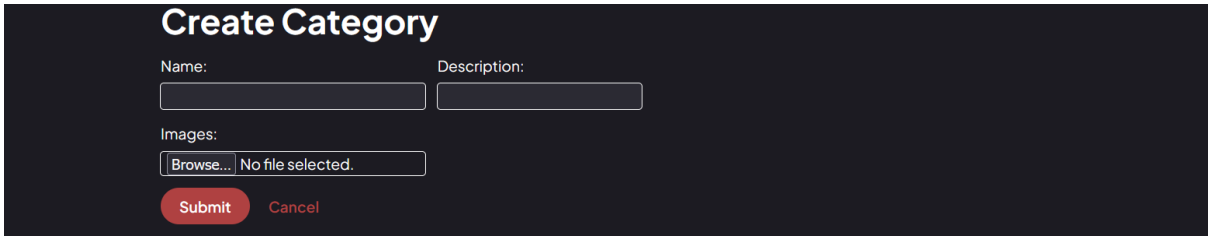


FIG 17.0 - CATEGORY CREATE FORM DESIGN

- Implement the functionality for the category create form, using previously declared styles.

```
const handleForm = (e) => {
  let name = e.target.name;
  let value = e.target.value;

  setForm((prevState) => ({
    ...prevState,
    [name]: value,
  }));
};

const submitForm = (e) => {
  e.preventDefault();
  const formData = new FormData();
  formData.append("file", form.file);

  let token = localStorage.getItem("token");

  axios
    .post(
      "/categories",
      {
        name: form.name,
        description: form.description,
        file: form.file,
        image_path: form.image_path,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
          "Content-Type": "multipart/form-data",
        },
      }
    )
    .then((res) => {
      console.log(res.data);
      navigate("/categories");
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 17.1 - CATEGORY CREATE FORM FRONTEND FUNCTIONALITY

```
const createData = (req, res) => {
  let recipeData = req.body;

  console.log(req.file);

  if (req.file) {
    recipeData.image_path =
      process.env.STORAGE_ENGINE === "S3" ? req.file.key : req.file.filename;
  }
  // include this else, if image required
  else {
    return res.status(422).json({
      message: req.imageError || "Image not uploaded!",
    });
  }

  Recipe.create(recipeData)
    .then((data) => {
      console.log("New Recipe Created!", data);
      res.status(201).json(data);
    })
    .catch((err) => {
      if (err.name === "ValidationError") {
        console.error("Validation Error!!", err);
        res.status(422).json({
          msg: "Validation Error",
          error: err.message,
        });
      } else {
        console.error(err);
        res.status(500).json(err);
      }
    });
};
```

FIG 17.2 - CATEGORY_CONTROLLER.JS CREATE FUNCTIONALITY

5.7.4 item 3

- Create the ingredient create form, using previously declared styles.

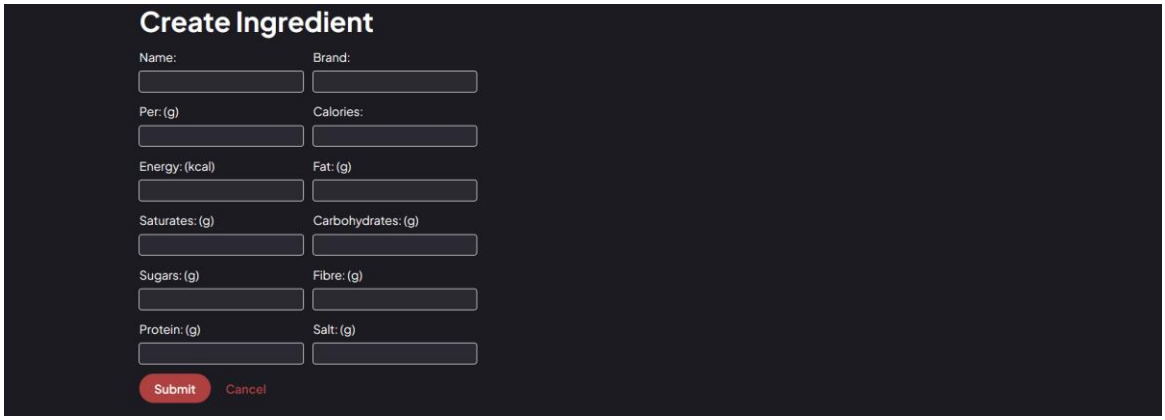


FIG 18.0 - INGREDIENT CREATE FORM

- Implement the functionality for the ingredient create form.

```
const handleForm = (e) => {
  let name = e.target.name;
  let value = e.target.value;

  setForm((prevState) => ({
    ...prevState,
    [name]: value,
  }));
};

const submitForm = (e) => {
  e.preventDefault();

  let token = localStorage.getItem("token");

  axios
    .post(
      "/ingredients",
      {
        name: form.name,
        brand: form.brand,
        per: form.per,
        calories: form.calories,
        energy: form.energy,
        fat: form.fat,
        saturates: form.saturates,
        carbohydrates: form.carbohydrates,
        sugars: form.sugars,
        salt: form.salt,
        fibre: form.fibre,
        protein: form.protein,
        measurements: form.measurements,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    )
    .then((res) => {
      console.log(res.data);
      navigate("/recipes");
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 18.1 - INGREDIENT CREATE FORM FRONTEND FUNCTIONALITY

```
const createData = (req, res) => {
  let ingredientData = req.body;

  Ingredient.create(ingredientData)
    .then((data) => {
      console.log("New Ingredient Created!", data);
      res.status(201).json(data);
    })
    .catch((err) => {
      if (err.name === "ValidationError") {
        console.error("Validation Error!!", err);
        res.status(422).json({
          msg: "Validation Error",
          error: err.message,
        });
      } else {
        console.error(err);
        res.status(500).json(err);
      }
    });
};
```

FIG 18.2 - INGREDIENT CREATE FORM BACKEND FUNCTIONALITY

5.8 Sprint 4

5.8.1 goal

The goal of Sprint 4 is to focus on creating the listing page for recipes and categories. Then to implement the search functionality to these pages.

5.8.2 item 1

- Create the recipe listing page.

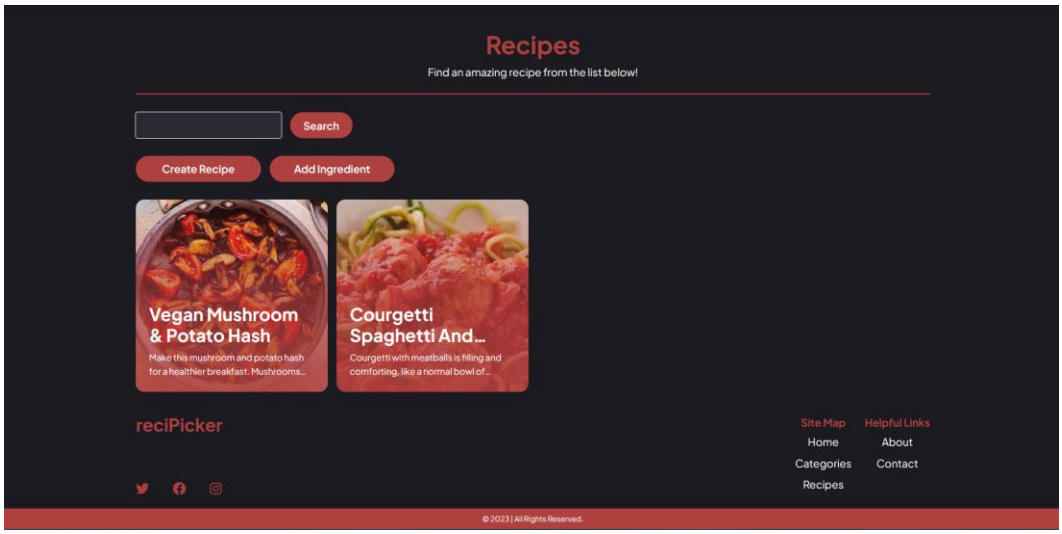


FIG 19.0 - RECIPE LISTING PAGE

5.8.3 item 2

- Create the category listing page.

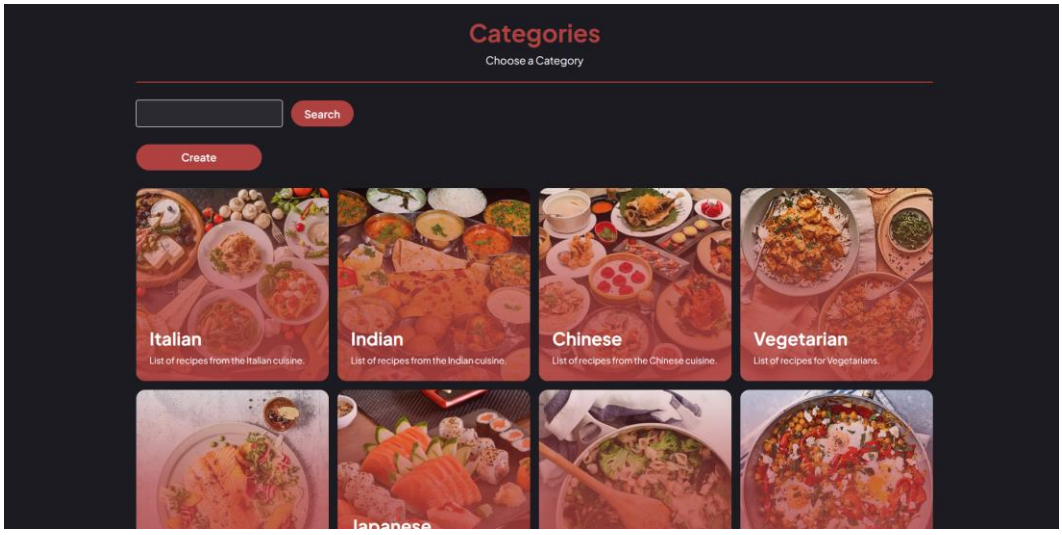


FIG 20.0 - CATEGORY LISTING PAGE

5.8.4 item 3

- Implement the search on the listing pages.

```
<section className="wrapper pd-block-md">
  <input
    className="form-input"
    type="text"
    value={query}
    onChange={handleInputChange}
  />
  <button className="button" data-type="primary" onClick={handleSearch}>
    Search
  </button>
</section>
```

FIG 21.0 - SEARCH INPUT FOR LISTING PAGE

```
const handleInputChange = (e) => {
  setQuery(e.target.value);
};

const handleSearch = () => {
  axios
    .get(`/recipes/search?q=${query}`)
    .then((res) => {
      setRecipes(res.data);
    })
    .catch((error) => {
      console.error(error);
    });
};
```

FIG 21.1 - SEARCH FUNCTIONALITY FOR LISTING PAGE

5.9 Sprint 5

5.9.1 goal

The goal of Sprint 5 is to focus on the creation of the page that displays the information of the recipes that have been created on the application.

5.9.2 item 1

- Create and style recipe show page.

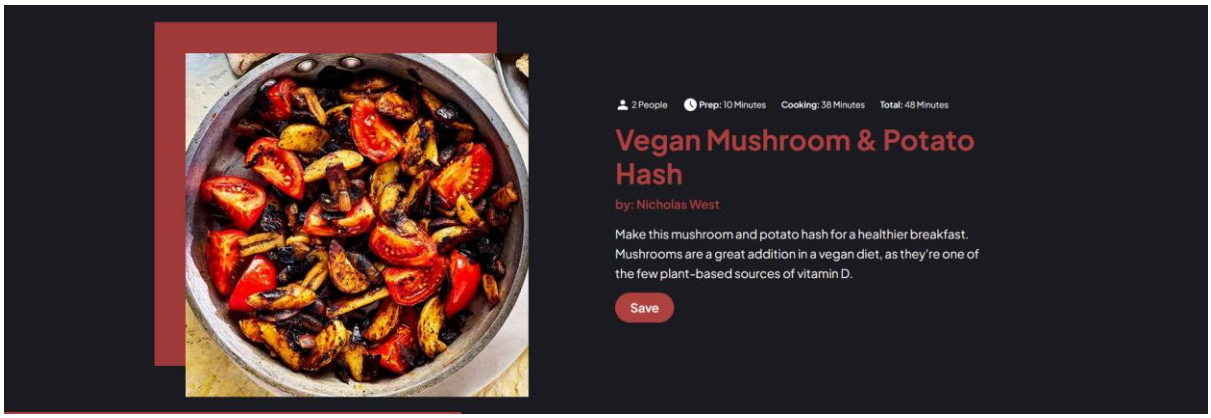


FIG 22.0 - RECIPE SHOW PAGE

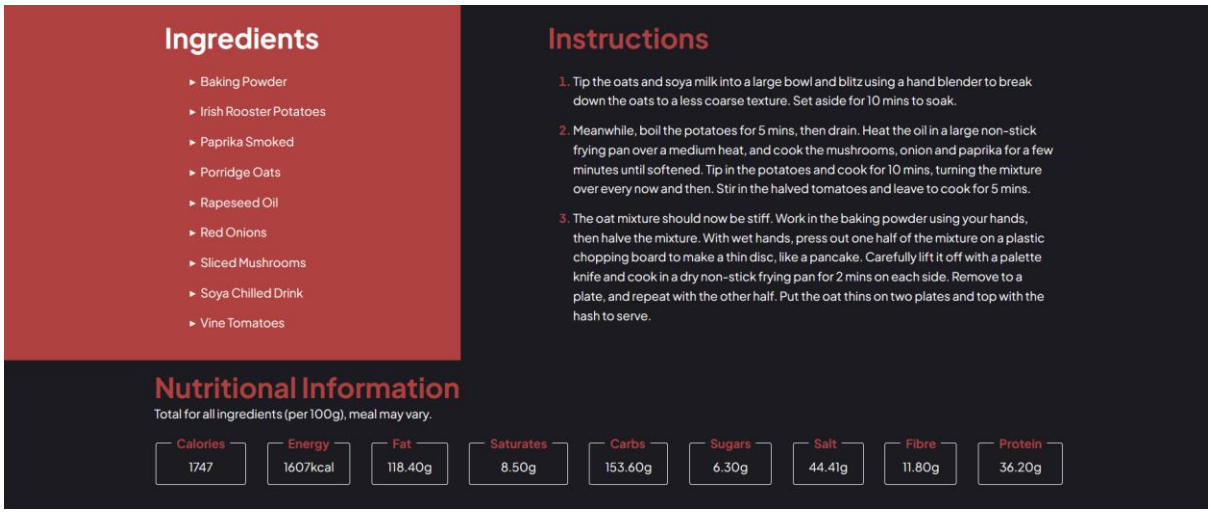


FIG 22.1 - RECIPE SHOW PAGE

5.10 Sprint 6

5.10.1 goal

The goal of Sprint 6 is to focus on creating the comments form and component. Then implementing the functionality and importing the comment component to the recipe show page.

5.10.2 item 1

- Create and style the comment component.

```
const Comment = (props) => {
  let userId = localStorage.getItem("userId");
  let userRole = localStorage.getItem("userRole");

  return (
    <fieldset className="comment">
      <legend className="comment-author">
        {props.data.author.name}
        {(props.authenticated && props.data.author._id === userId) ||
        (props.authenticated && userRole === "640d002ab551cf397c59abab") ? (
          <DeleteButton
            id={props.id}
            query="comments"
            callback={props.callback}
          />
        ) : (
          ""
        )}
      </legend>
      <p className="comment-text">{props.data.comment}</p>
    </fieldset>
  );
};
```

FIG 23.0 - COMMENT COMPONENT

5.10.3 item 2

- Implemented comment functionality.

```
  const handleCommentForm = (e) => {
    let name = e.target.name;
    let value = e.target.value;

    setCommentForm((prevState) => ({
      ...prevState,
      [name]: value,
    }));
  };
```

FIG 24.0 - COMMENT FUNCTIONALITY

```
const submitCommentForm = (e) => {
  e.preventDefault();

  axios
    .post(
      "/comments",
      {
        author: commentForm.author,
        recipeId: commentForm.recipeId,
        comment: commentForm.comment,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    )
    .then((res) => {
      console.log(res.data);
      navigate(0);
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 24.1 - COMMENT FUNCTIONALITY

5.10.4 item 3

- Imported comment component into the recipe show page.

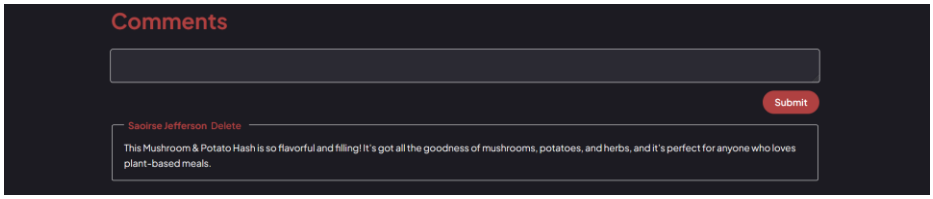


FIG 25.0 - COMMENT COMPONENT IMPORTED INTO RECIPE SHOW PAGE

5.11 Sprint 7

5.11.1 goal

The goal of Sprint 7 is to focus on creating the account page to store saved bookmarks.

5.11.2 item 1

- Create account to store saved bookmarks.

```
<>
  <div className="wrapper">
    {props.authenticated && userId === id ? (
      <section className="grid-span-full">
        <h2>Welcome, {user.name}</h2>
        <h3>Your Bookmarks</h3>
        <ol>{bookmarksList}</ol>
      </section>
    ) : (
      <p className="grid-span-full">Unauthorized User</p>
    )}
  </div>
</>
```

FIG 26.0 - USER PROFILE

- Add functionality to list bookmarks.

```
const bookmarksList = bookmarks.map((data) => {
  return (
    <li
      key={data._id}
      data={data}
      onAuthenticated={props.onAuthenticated}
      authenticated={props.authenticated}
    >
      <Link to={`/recipes/${data.recipeId._id}`}>{data.recipeId.name}</Link>
      <p>{data.recipeId.description}</p>
    </li>
  );
});
```

FIG 26.1 - DISPLAY BOOKMARKS IN THE ARRAY

5.12 Sprint 8

5.12.1 Goal

The goal of Sprint 8 is to focus on the testing and bug fixes of the application, to ensure the application is working as intended and making any final adjustments to the user interface and functionality.

5.12.2 item 1

- Fix a bug where the userRole was not being returning undefined.

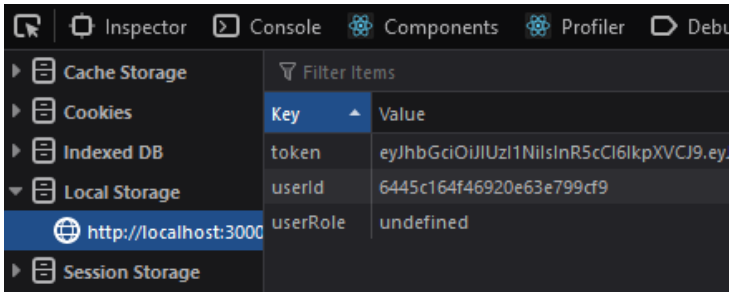


FIG 27.0 - LOCAL STORAGE, USERROLE UNDEFINED

The fix for this was to add “res.data.userRole” to the onAuthenticated passed through to the login form, when logging in.

```
const submitForm = () => {
  axios
    .post("/users/login", {
      email: form.email,
      password: form.password,
    })
    .then((res) => {
      // console.log(res.data);
      props.onAuthenticated(
        true,
        res.data.token,
        res.data.userId,
        res.data.userRole
      );
    })
    .catch((err) => {
      console.error(err);
      console.log(err.res.data);
    });
};
```

FIG 27.1 - LOGIN FORM FUNCTIONALITY

5.12.3 item 2

- Fix a bug where the image was not updating correctly when a recipe was edited. It would not update properly, it would try and return the same image, that was just deleted in the update.

```
if (file) {
  image_path =
    process.env.STORAGE_ENGINE === "S3" ? req.file.key : req.file.filename;
}
```

FIG 28.0 - RECIPE UPDATE FUNCTIONALITY IN BACKEND

To fix this issue, a change was made to the recipicker backend. By added “body.” Before the image_path.

```
if (file) {
  body.image_path =
    process.env.STORAGE_ENGINE === "S3" ? req.file.key : req.file.filename;
}
```

FIG 28.1 - RECIPE UPDATE FUNCTIONALITY WORKING IN BACKEND

5.12.4 item 3

- Fixed a blank screen returning if there is no content being returned from the database.

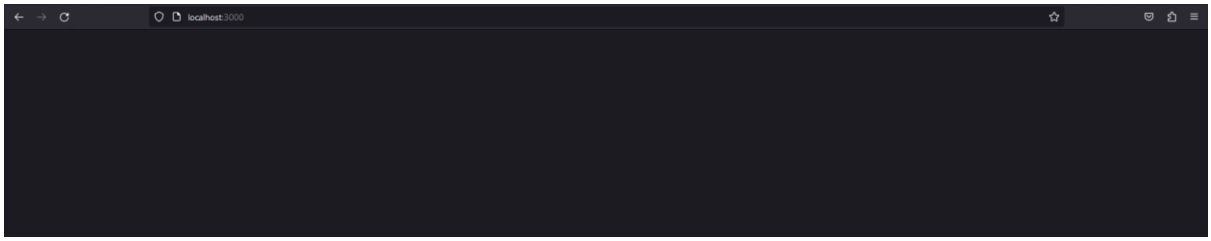


FIG 29.0 – ERROR CAUSING BLANK PAGE

The fix was to add some text saying loading until the data has been received from the database.

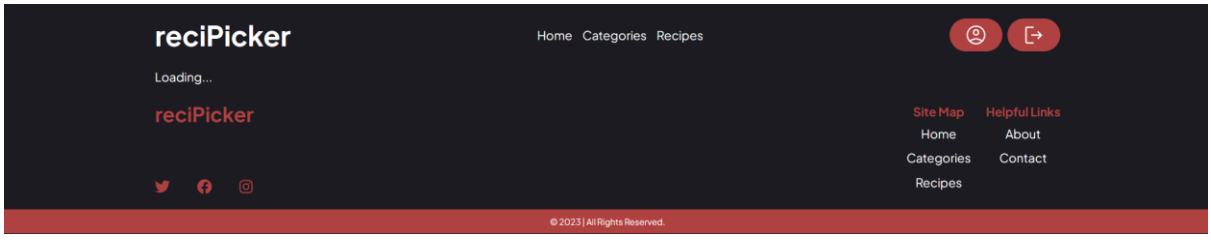


FIG 29.1 – ERROR FIX

5.12.5 item 4

- Fixed a bug where your account page would not load if a recipe that you have bookmarked was deleted.

This issue was fixed, by adding also adding to remove any bookmarks that have the same “recipeId”, as the recipe has “_id” in the database.

```
const deleteData = (req, res) => {
  let id = req.params.id;
  let imagePath = "";

  Recipe.findById(id)
    .then((data) => {
      if (data) {
        imagePath = data.image_path;
        return data.remove();
      } else {
        res.status(404).json({
          msg: `Recipe with id: ${id} not found`,
        });
      }
    })
    .then((data) => {
      console.log("Recipe removed!");

      Bookmark.deleteMany({ recipeId: id })
        .then((data) => {
          console.log("Bookmark removed!");
        })
        .catch((err) => {
          console.error(err);
          if (err.name === "CastError") {
            res.status(400).json({
              msg: `Bad request, ${id} is not a valid id`,
            });
          } else {
            res.status(500).json(err);
          }
        });
```

FIG 30.0 – RECIPICKER BACKEND, BOOKMARKS REMOVED ON RECIPE DELETE

5.13 Conclusion

In the Implementation chapter, It was discussed what the technologies used were and how they were being used in the application. The development environment was also discussed, alongside the SCRUM Methodology, and the Sprints.

6 Testing

6.1 Introduction

This chapter describes the testing that has been undertaken for the application. This chapter is presented in two sections:

1. Functional Testing
2. User Testing

Functional testing is a type of software testing whereby the system is tested against the functional requirements. The app is tested by looking to see if the actual output for a given input corresponds with the expected output. The tests should be based on the requirements for the app. The results of functional testing can indicate if a piece of software is functional and working, but not if the software is easy to use.

User testing looks to see if a piece of software is easy and intuitive for the user.

6.2 Functional Testing

This section describes the functional tests which were carried out on the app. These functional tests can be categorised as: (whatever is relevant to your app)

- Navigation
- CRUD

Functional testing generally uses a Black Box Testing technique which means that the internal logic of the system being tested is not of interest to the tester. The tester is only interested in whether the actual output agrees with the expected output.

6.2.1 Navigation

6.2.1.1 Test 1

Description:

Navigate to the view all recipes page.

Input:

Click the recipes button in the navigation.

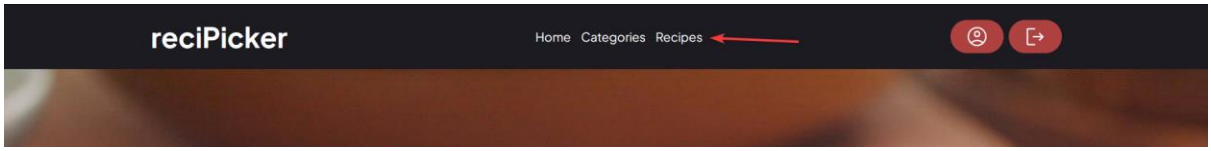


FIG 31.0 – NAVIGATE TO RECIPES

Expected Output:

Open the page that displays all recipes.

Actual Output:

The page that displays all recipes opened.

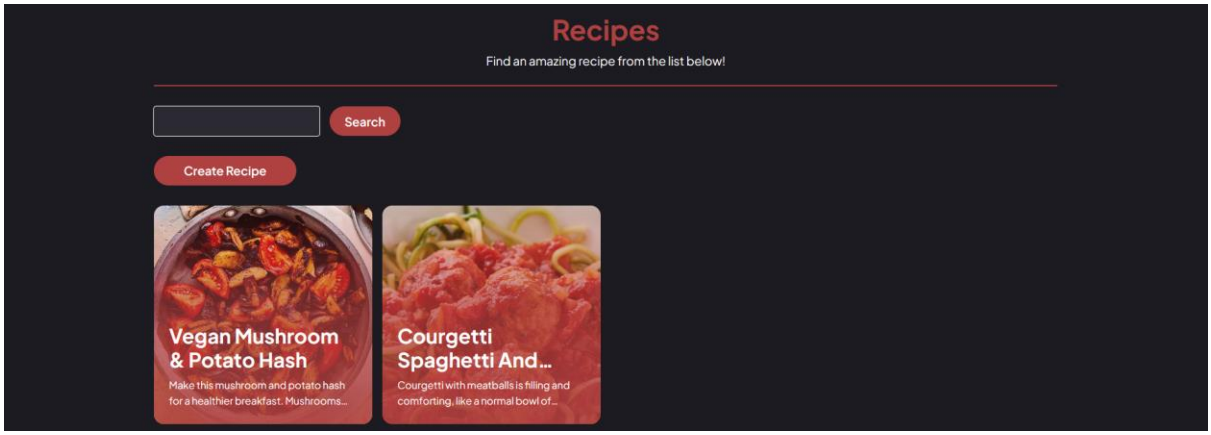


FIG 31.1 - VIEW ALL RECIPES PAGE

Comment:

This worked correctly.

6.2.1.2 Test 2

Description:

Navigate to the create a recipe form.

Input:

Click the create recipe button.

FIG 32.0 - NAVIGATE TO CREATE RECIPE FORM

Expected Output:

Open the create recipe form.

Actual Output:

The create a recipe form opened.

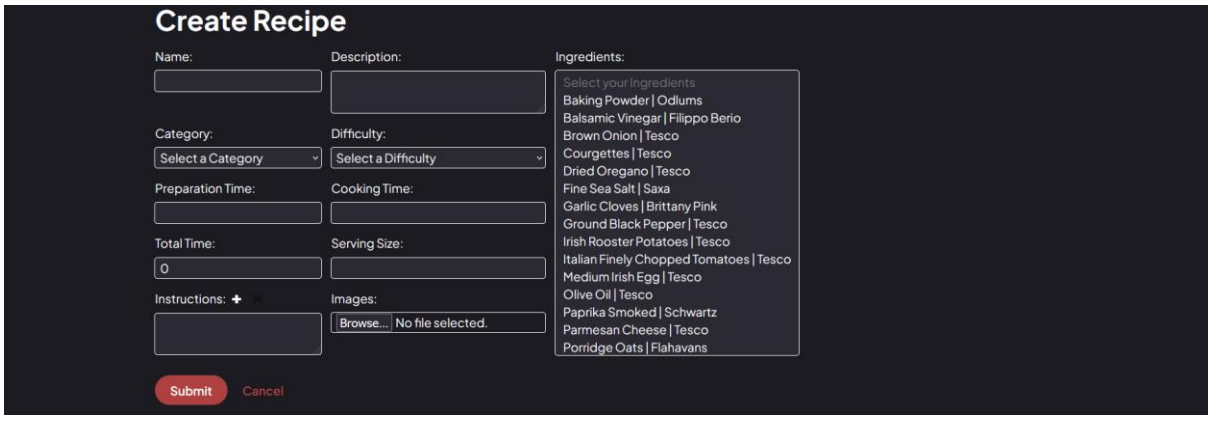


FIG 32.1 – CREATE RECIPE FORM

Comment:

This worked correctly.

6.2.1.3 Test 3

Description:

Navigate back to the view all recipes page.

Input:

Click the cancel button at the bottom of the form.

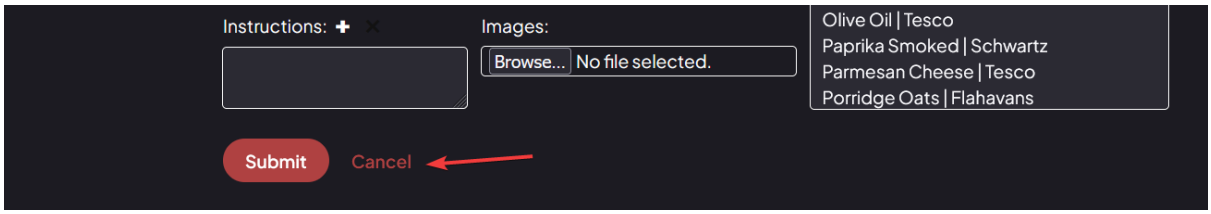


FIG 33.0 – NAVIGATE TO ALL RECIPES PAGE BY CLICKING CANCEL

Expected Output:

Navigate to the page to view all recipes page.

Actual Output:

Navigated successfully to the view all recipes page.

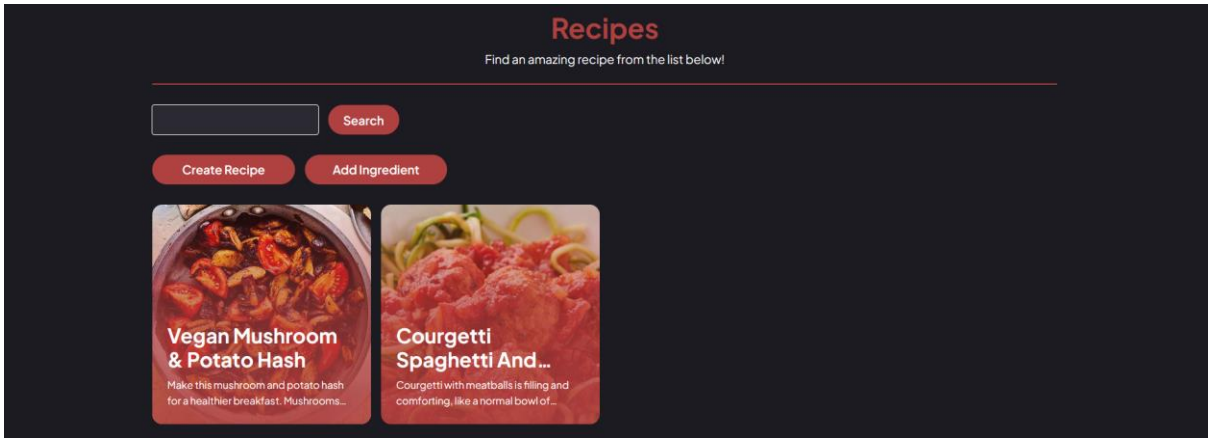


FIG 33.1 - THE VIEW ALL RECIPES PAGE

Comment:

This worked correctly.

6.2.1.4 Test 4

Description:

Navigate to the create a recipe form.

Input:

Click the add ingredient button.

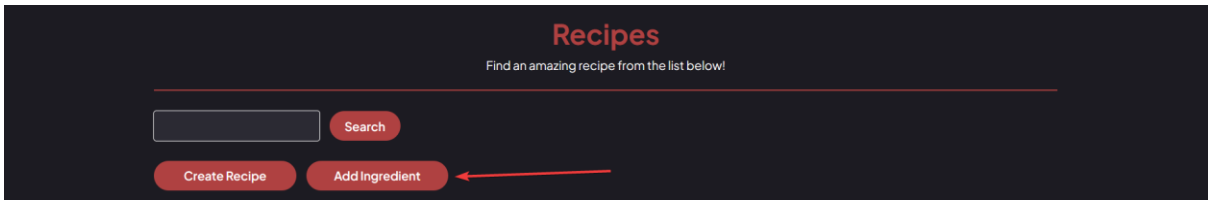


FIG 34.0 – NAVIGATE TO CREATE INGREDIENT FORM

Expected Output:

Open the create ingredient form.

Actual Output:

The create an ingredient form opened.

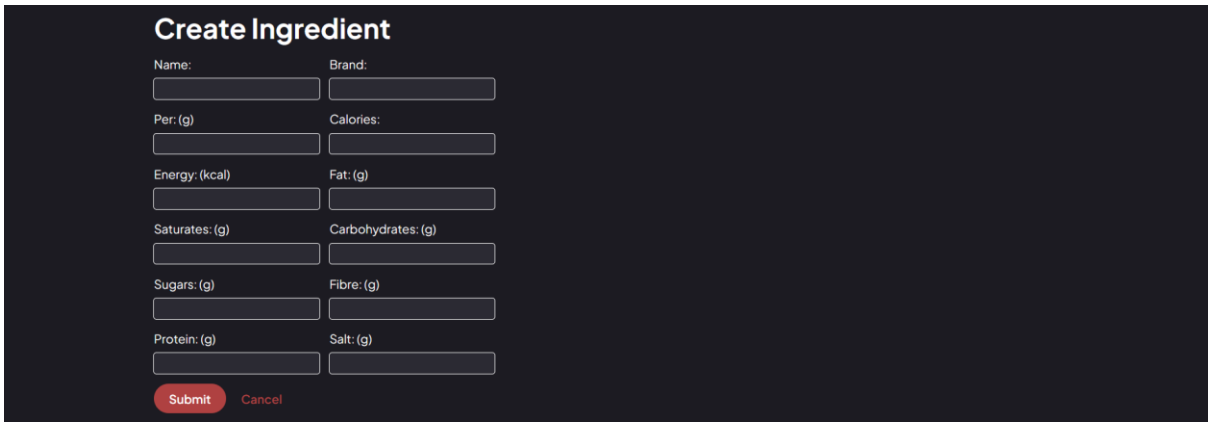


FIG 34.1 – CREATE INGREDIENT FORM

Comment:

This worked correctly.

6.2.1.5 Test 5

Description:

Navigate back to the view all recipes page.

Input:

Click the cancel button at the bottom of the form.

FIG 35.0 – NAVIGATE TO VIEW ALL RECIPES PAGE BY CLICKING CENCEL

Expected Output:

Navigate to the page to view all recipes page.

Actual Output:

Navigated successfully to the view all recipes page.

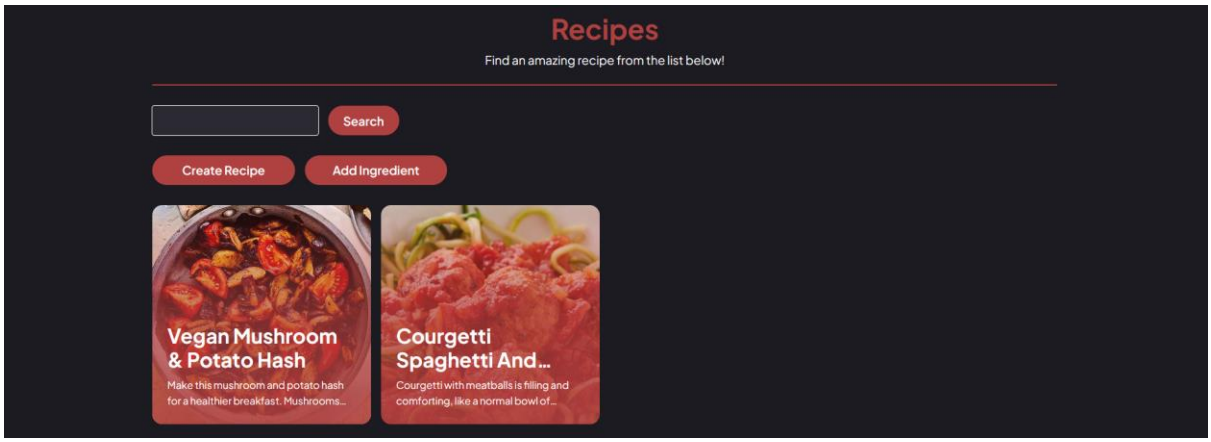


FIG 35.1 - VIEW ALL RECIPES PAGE

Comment:

This worked correctly.

6.2.1.6 Test 6

Description:

Navigate to the view a recipe page.

Input:

Click on the title in the recipe card.

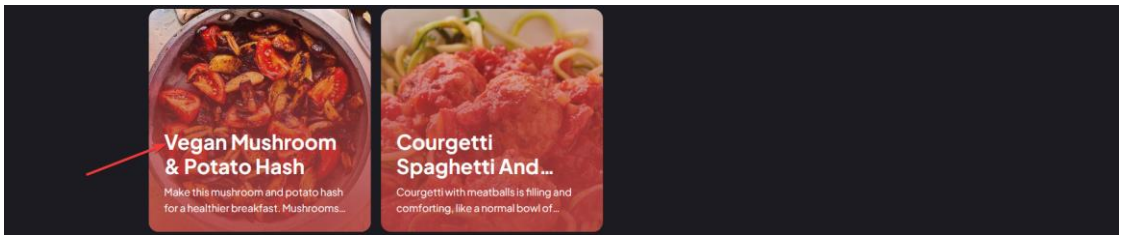


FIG 36.0 – NAVIGATE TO SHOW RECIPE PAGE

Expected Output:

Open the recipe show page, to display all the information about a recipe.

Actual Output:

The recipe show did not page open correctly, it returned a blank screen.

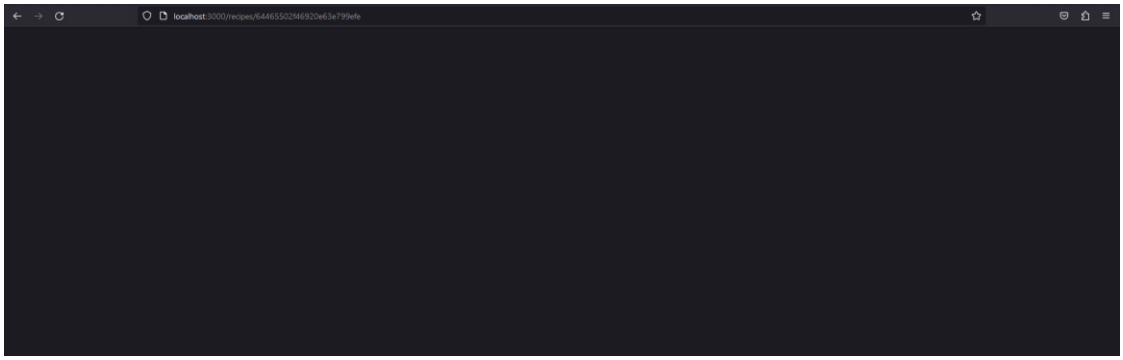


FIG 36.1 - BLANK PAGE ERROR

Comment:

This was due to there being no content / not having long enough to load. This was fixed by adding a placeholder text of “loading” until the content has been pulled from the database to be displayed.

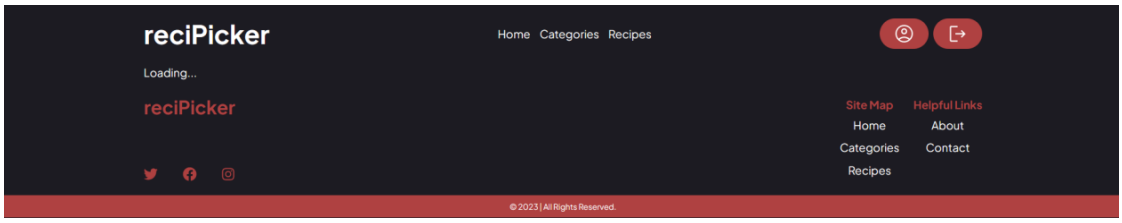


FIG 36.2 – BLANK PAGE ERROR FIX

6.2.1.7 Test 7

Description:

Navigate to the view a recipe page.

Input:

Click on the title in the recipe card.

Expected Output:

Navigate to the recipe show page, which displays all the information about a recipe.

Actual Output:

Navigated successfully to the recipe show page.

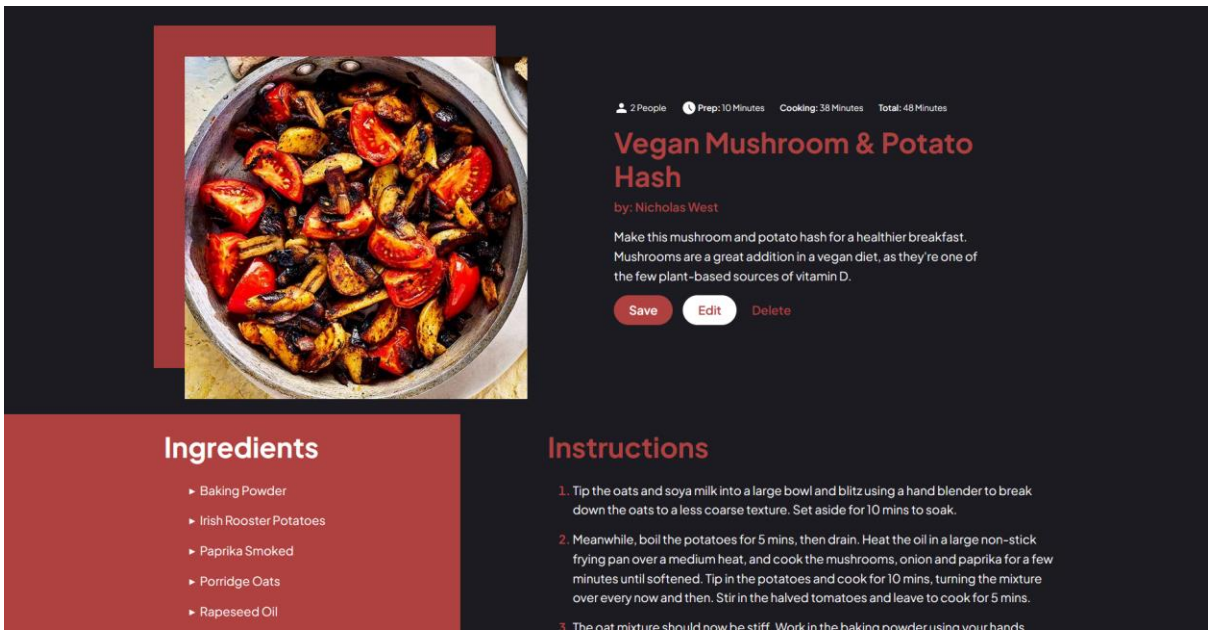


FIG 37.0 - SHOW RECIPE PAGE

Comment:

This worked correctly.

6.2.1.8 Test 8

Description:

Navigate to the edit recipe form.

Input:

Click on the edit button on the show recipe page.

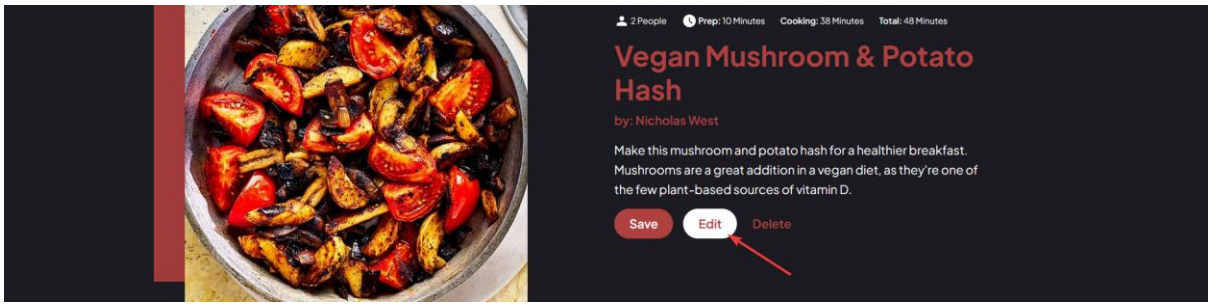


FIG 38.0 – NAVIGATE TO EDIT RECIPE FORM

Expected Output:

Open the edit recipe form.

Actual Output:

Opened the edit recipe form.

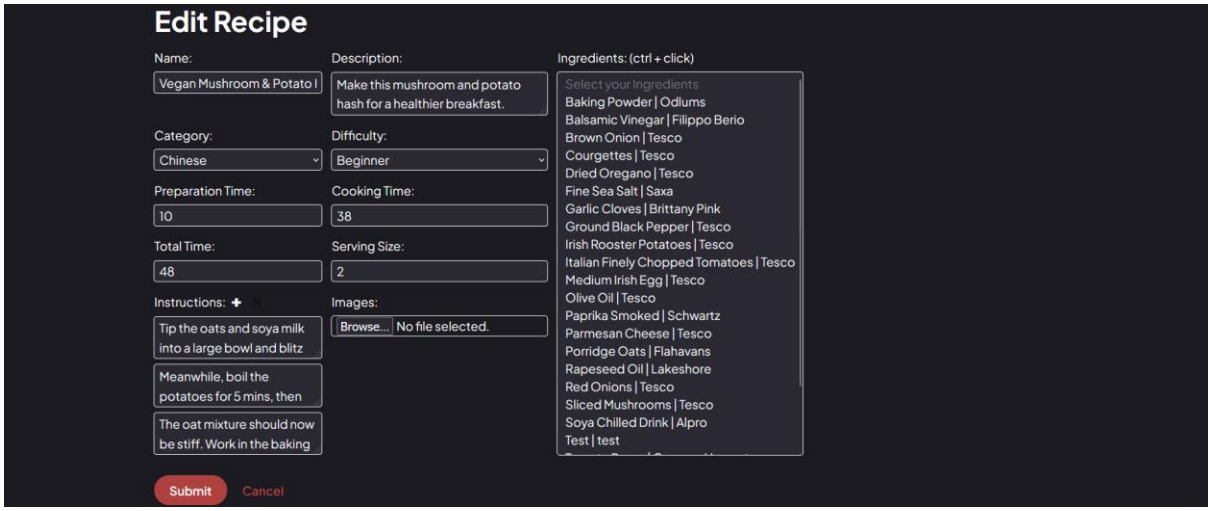


FIG 38.1 – EDIT RECIPE FORM

Comment:

This worked correctly.

6.2.1.9 Test 9

Description:

Navigate back to the view all recipes page.

Input:

Click the cancel button at the bottom of the edit recipe form.

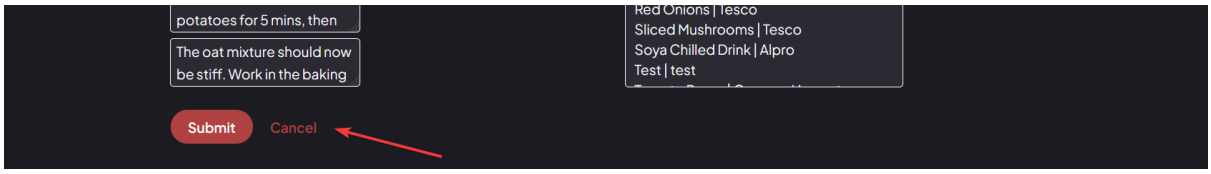


FIG 39.0 – NAVIGATE TO THE SHOW RECIPE PAGE

Expected Output:

Navigate back to the show recipe page.

Actual Output:

Navigated back to the show recipe page.

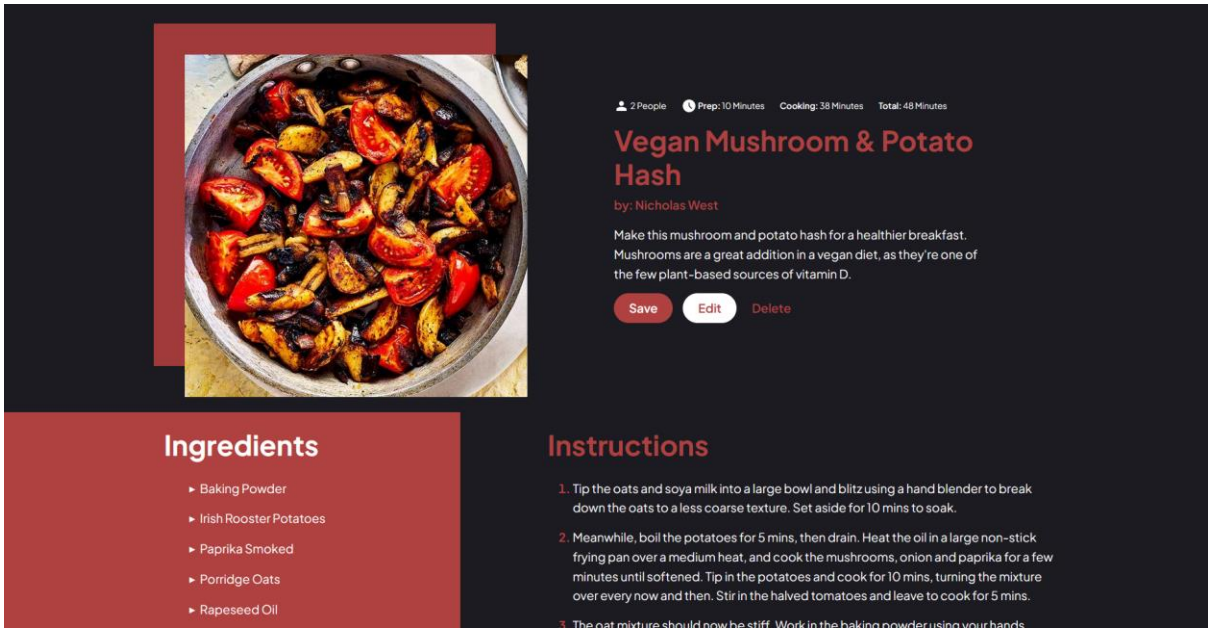


FIG 39.1 – RECIPE SHOW PAGE

Comment:

This worked correctly.

6.2.1.10 Test 10

Description:

Navigate to the view all categories page.

Input:

Click the recipes button in the navigation.

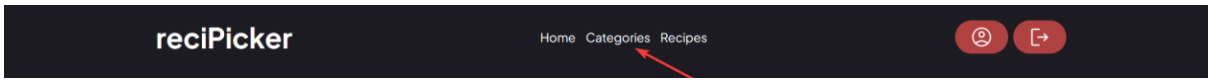


FIG 40.0 – NAVIGATE TO ALL CATEGORIES PAGE

Expected Output:

Open the page that displays all recipes.

Actual Output:

The page that displays all recipes opened.

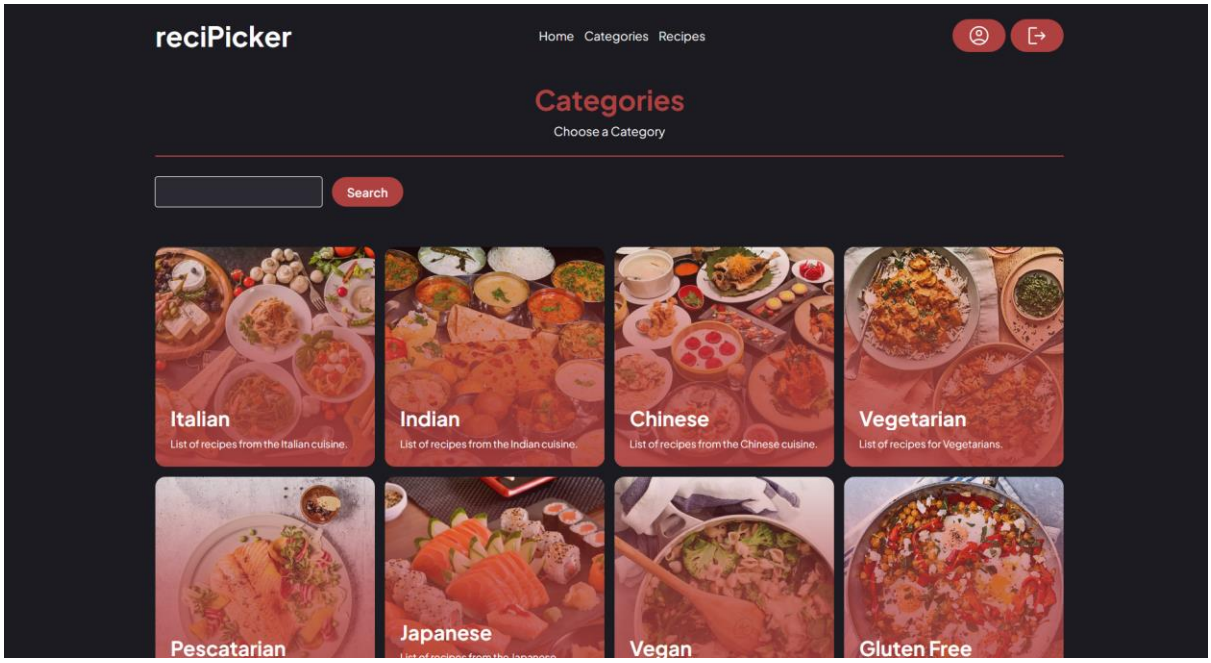


FIG 40.1 – ALL CATEGORIES PAGE

Comment:

This worked correctly.

6.2.1.11 Test 11

Description:

Navigate to the view all categories page.

Input:

Click the recipes button in the navigation.

FIG 41.0 - NAVIGATE TO SHOW CATEGORY PAGE

Expected Output:

Open the page that displays all recipes for the selected category.

Actual Output:

The page that displays all recipes for a specific category opened.

FIG 41.1 – SHOW CATEGORY PAGE

Comment:

This worked correctly.

6.2.1.12 Test 12

Description:

Navigate to the create a recipe form.

Input:

Click the create category button.

FIG 42.0 – NAVIGATE TO CREATE CATEGORY

Expected Output:

Open the create category form.

Actual Output:

The create a category form opened.

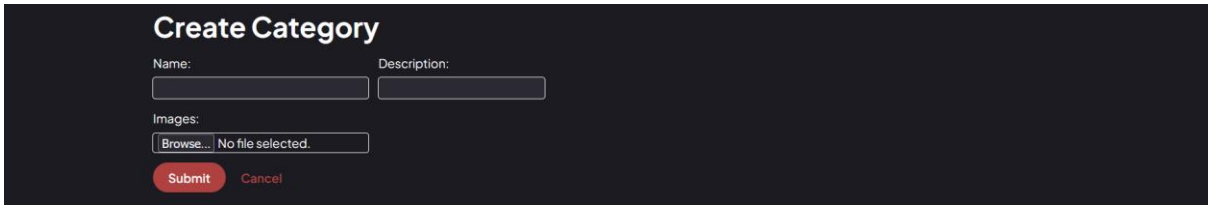


FIG 42.1 – CREATE CATEGORY FORM

Comment:

This worked correctly.

6.2.1.13 Test 13

Description:

Navigate back to the view all categories page.

Input:

Click the cancel button at the bottom of the create category form.

FIG 43.0 – NAVIGATE BACK TO ALL CATEGORIES PAGE

Expected Output:

Navigate back to the show all categories page.

Actual Output:

Navigated back to the show all categories page.

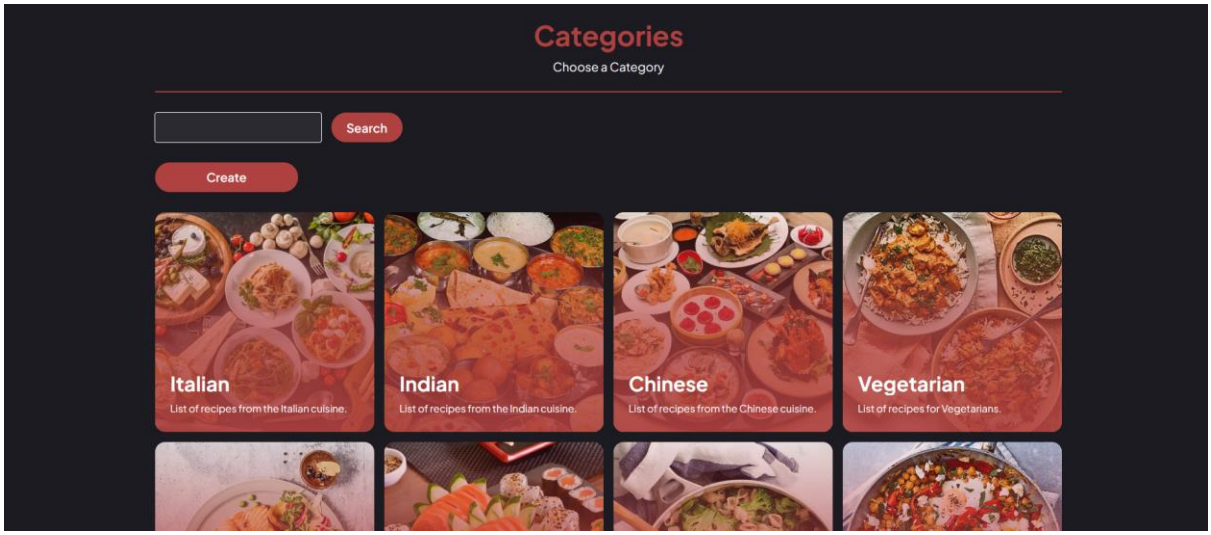


FIG 43.1 – ALL CATEGORIES PAGE

Comment:

This worked correctly.

6.2.1.14 Test 14

Description:

Navigate to the account page to view bookmarks.

Input:

Click the account button at the top right of the page.

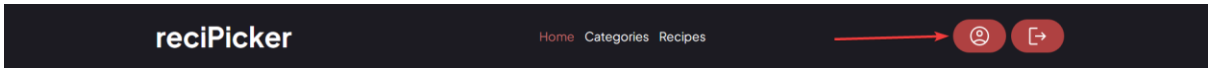


FIG 44.0 - NAVIGATE TO ACCOUNT PAGE

Expected Output:

Navigate to the account page.

Actual Output:

Navigating to the account page returned a blank screen.

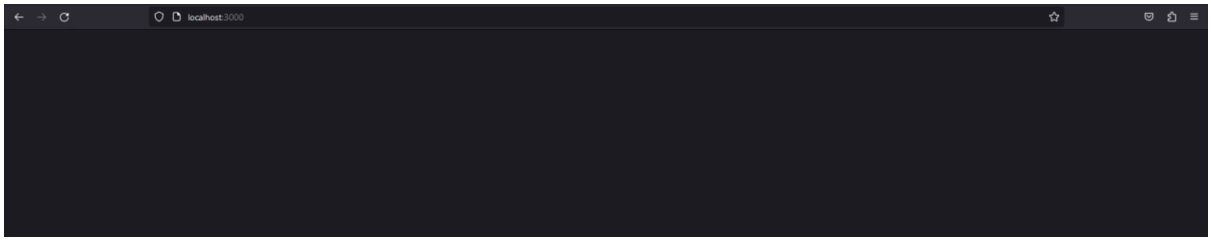


FIG 44.1 – BLANK PAGE ERROR

Comment:

This was caused by a bookmarked recipe being deleted. This was resolved by ensuring any bookmarks with the same recipe id were removed simultaneously.

6.2.1.15 Test 15

Description:

Navigate to the account page to view bookmarks.

Input:

Click the account button at the top right of the page.

Expected Output:

Navigate to the account page.

Actual Output:

Navigated to the account page.

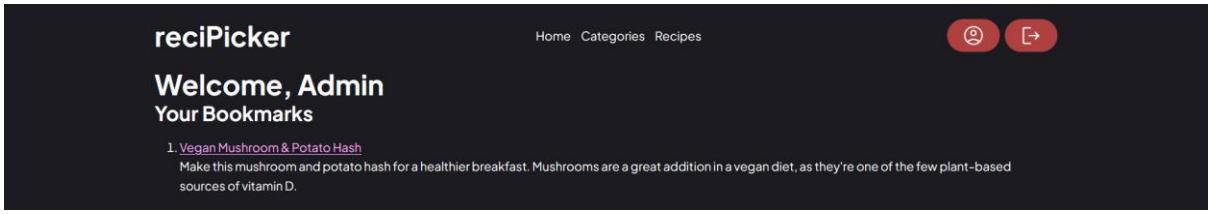


FIG 45.0 - ACCOUNT PAGE

Comment:

This worked correctly.

6.2.1.16 Test 16

Description:

Navigate to the home page.

Input:

Click the home link in the navigation.

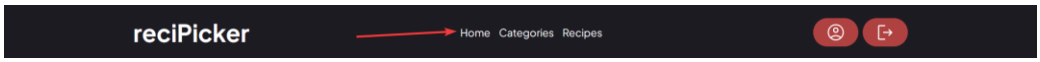


FIG 46.0 - NAVIGATE TO HOME

Expected Output:

Navigate to the home page.

Actual Output:

Navigated to the home page.

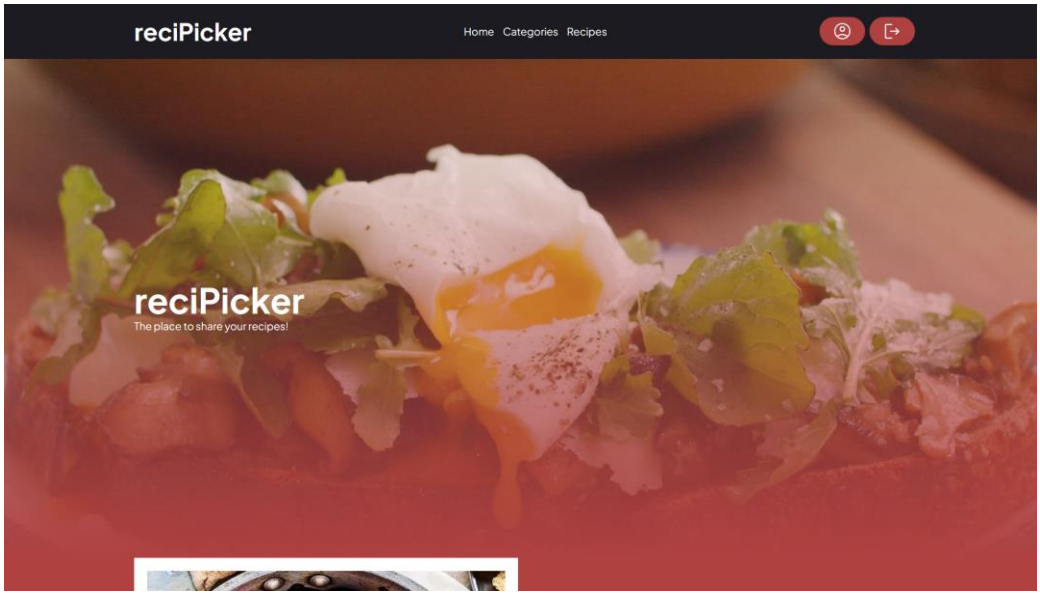


FIG 46.1 - HOMEPAGE

Comment:

This worked correctly.

6.2.2 CRUD

6.2.2.1 Test 1

Description:

Create an account.

Input:

Open the registration form.

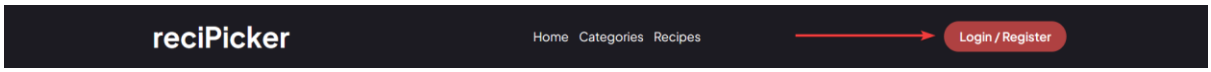


FIG 47.0 - OPEN THE REGISTRATION/LOGIN MODAL

fill the in the form and submit the form.

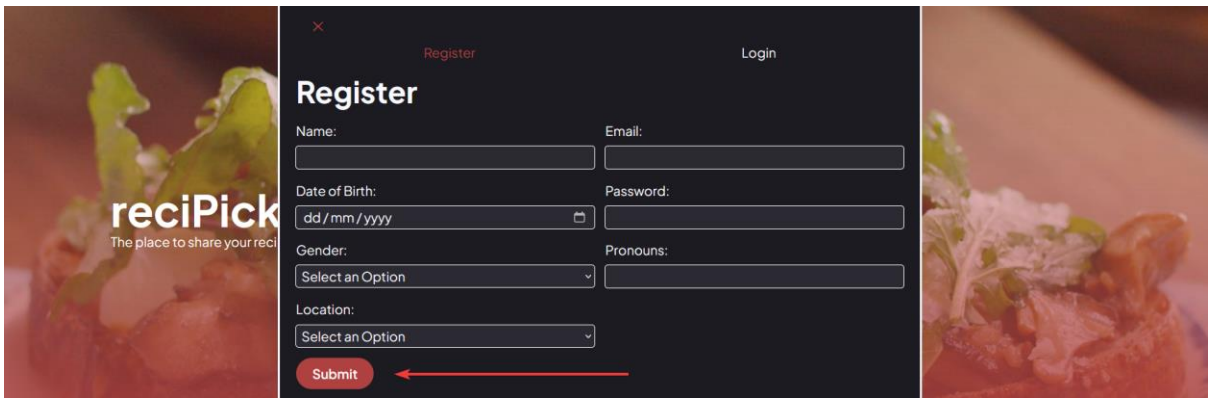


FIG 47.1 - OPENED MODAL

Expected Output:

Register successfully.

Actual Output:

Registration failed.

Comment:

This is due to changing the location to a third-party API, and not changing the “type” for location in the location_schema.js, in the backend.

6.2.2.2 Test 2

Description:

Log in to the application.

Input:

Open the login form.

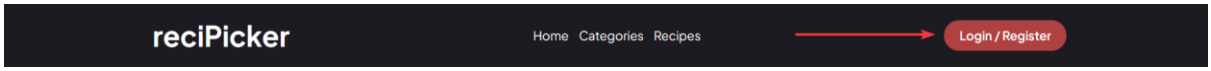


FIG 48.0 - TEXT

Fill in the form, submit the form.

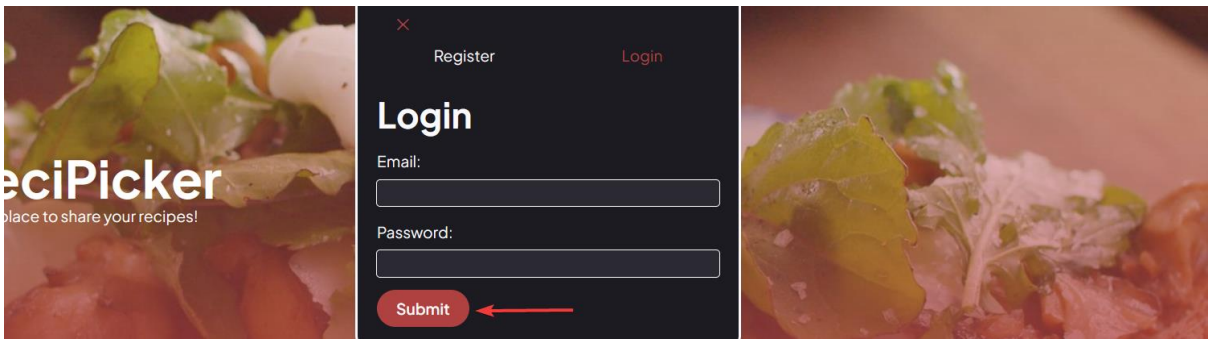


FIG 48.1 - TEXT

Expected Output:

Log in successfully.

Actual Output:

Logged in successfully.

Comment:

I logged in successfully.

6.2.2.3 Test 3

Description:

Create a recipe.

Input:

Navigate to the create a recipe form.

FIG 49.0 – NAVIGATE CREATE RECIPE FORM

Fill in the form and submit it.

FIG 49.1 – CREATE RECIPE FORM

Expected Output:

Create the recipe and redirect to the view all recipes page.

Actual Output:

The recipe was created successfully.

Comment:

This worked correctly.

6.2.2.4 Test 4

Description:

Edit a recipe.

Input:

Navigate to the edit recipe form.

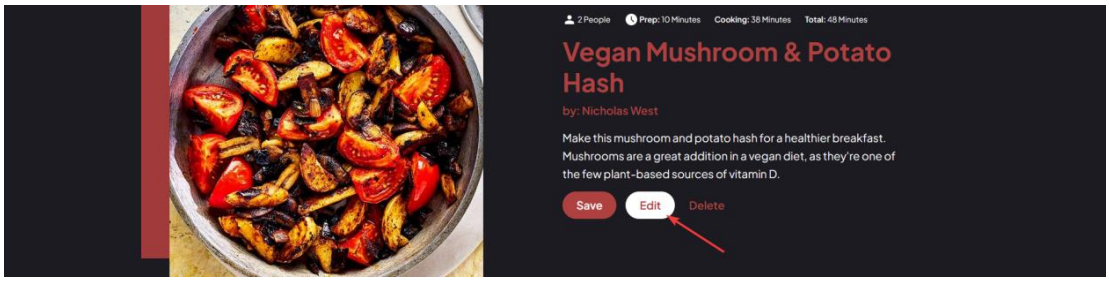


FIG 50.0 – NAVIGATE TO EDIT FORM

Make your changes and submit the form.

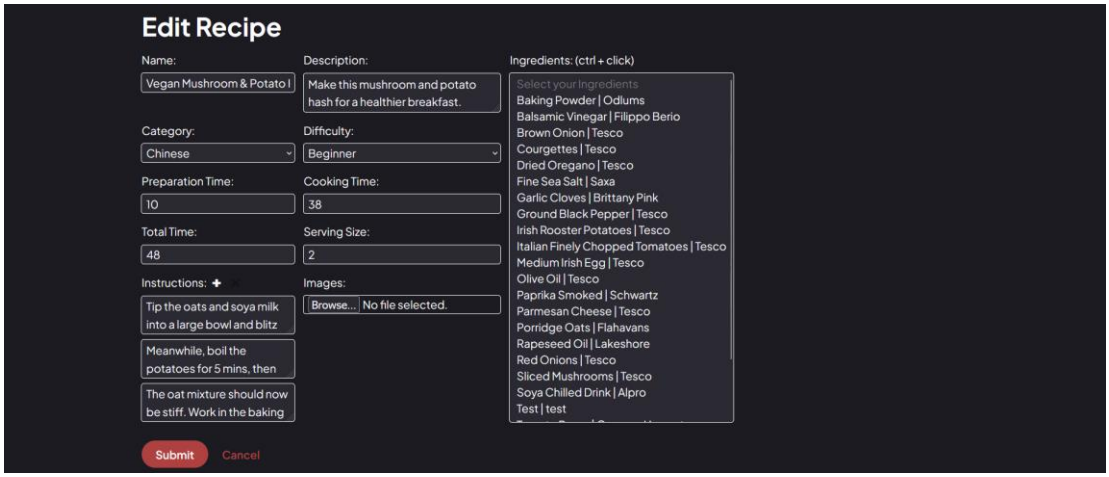


FIG 50.1 – EDIT FORM

Expected Output:

The edit form submits successfully.

Actual Output:

The edit form was submitted, and the recipe was updated.

Comment:

This worked correctly.

6.2.2.5 Test 5

Description:

Delete a recipe.

Input:

Click the delete button on the recipe.

FIG 51.0 – CLICK THE DELETE BUTTON

Expected Output:

The recipe deletes successfully.

Actual Output:

The recipe was deleted successfully.

Comment:

This worked correctly.

6.2.2.6 Test 5

Description:

Create an Ingredient.

Input:

Navigate to the create an Ingredient form.

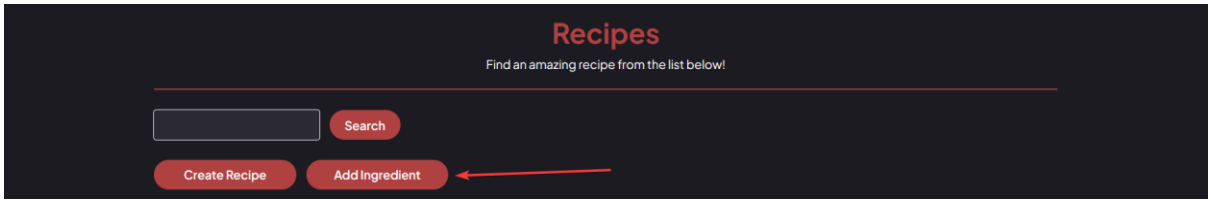


FIG 52.0 - NAVIGATE TO CREATE INGREDIENT FORM

Fill in and submit the form.

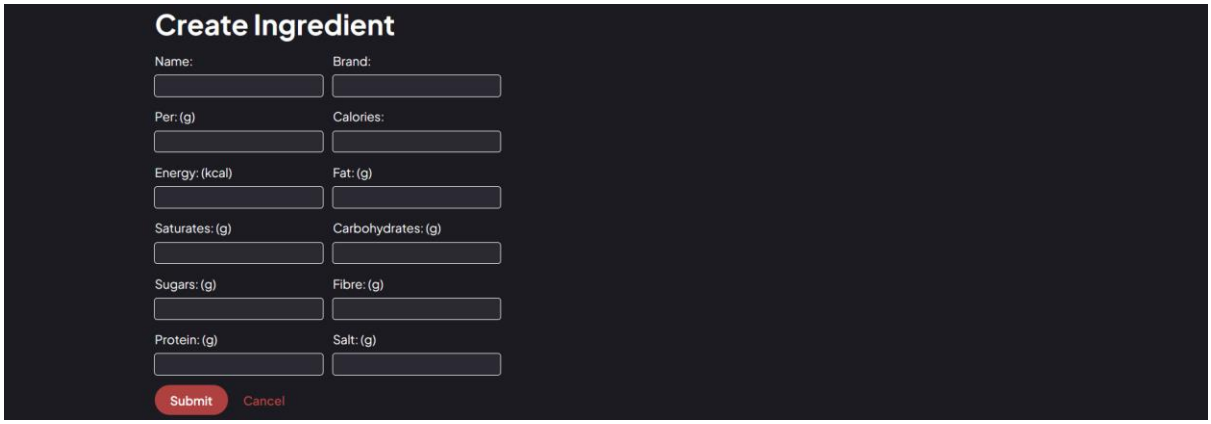


FIG 52.1 – CREATE INGREDIENT FORM

Expected Output:

Create ingredient form submits successfully.

Actual Output:

The create ingredient form was submitted, and the ingredient was added to the database.

Comment:

This worked correctly.

6.2.2.7 Test 6

Description:

Create a category.

Input:

Navigate to the create a category form.

FIG 53.0 – NAVIGATE TO CREATE CATEGORY FORM

Fill in and submit the form.

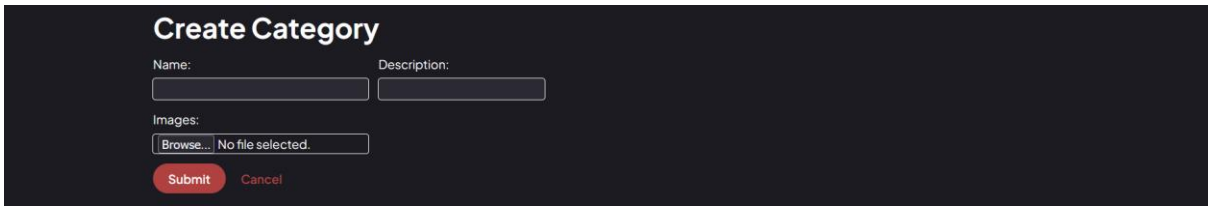


FIG 53.1 - CREATE RECIPE FORM

Expected Output:

Create a category that submits successfully.

Actual Output:

The create category form was submitted.

Comment:

This worked correctly.

6.2.2.8 Test 7

Description:

Save a bookmark.

Input:

Open a recipe and click the save button.

FIG 54.0 - CLICK THE SAVE BOOKMARK BUTTON

Expected Output:

Bookmark saves successfully to your account.

Actual Output:

The bookmark was saved successfully to your account.

FIG 54.1 – ACCOUNT PAGE

Comment:

This worked correctly.

6.2.2.9 Test 8

Description:

Comment on a recipe.

Input:

Fill in the form and post your comment.

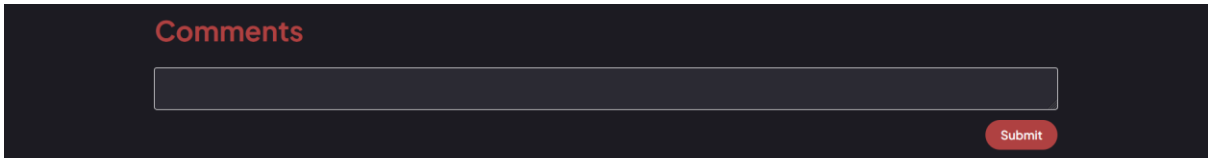


FIG 55.0 - POST A COMMENT

Expected Output:

Comment posts on the recipe.

Actual Output:

The comment was posted successfully onto the recipe.

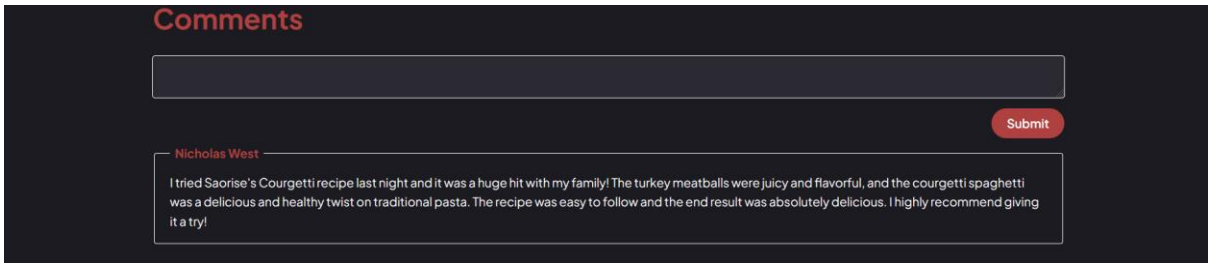


FIG 55.1 – COMMENT POSTED

Comment:

This worked correctly.

6.2.2.10 Test 9

Description:

Delete a comment on a recipe.

Input:

Click the delete button on your comment.

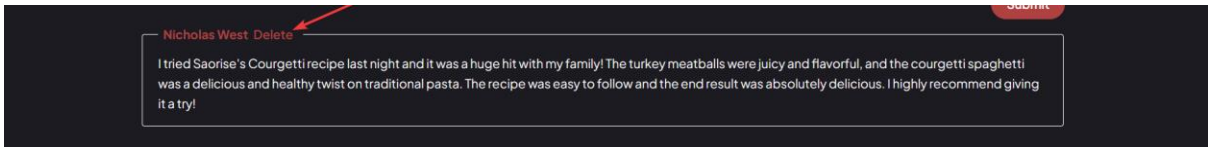


FIG 56.0 – DELETE COMMENT

Expected Output:

The comment deleted from the recipe.

Actual Output:

The comment was deleted successfully from the recipe.

Comment:

This worked correctly.

6.2.3 Discussion of Functional Testing Results

The navigation functionality of the application has been successfully implemented, ensuring that all routes redirect the user to the appropriate pages while displaying the correct information based on whether the user is logged in or logged out. Additionally, the registration/login modal is functioning correctly, allowing users to open and close it without any issues.

As for the CRUD aspect of the application, all implemented CRUD methods are working as intended. However, the forms currently lack client-side validation, which means that users are not provided with any visual feedback if they input incorrect information. To minimize the potential for errors, most of the inputs are in the form of text or select menus. While it is possible to add more CRUD methods, such as those for ratings, ingredient and category editing and deletion, and comment editing, these features have not yet been implemented.

Overall, the application has been developed with the necessary functionalities to ensure smooth navigation and proper data management, and while there is potential for further improvements, it currently meets the established requirements for a general user to use the application.

6.3 User Testing

6.3.1 Application Uses

Users using the web application can:

- Register for an account.
- Log in to their account.
- View all recipes.
- View a single recipe.
- View comments on a recipe.
- Search all recipes by name.
- View all categories.
- View a single category.
- Search all categories.

Members using the web application can:

- Everything users can do.
- Create a recipe.
- Edit a recipe you have created.
- Delete a recipe you have created.
- Comment on any recipe.
- Delete a comment you have made on a recipe.
- Bookmark a recipe.

Members using the web application can:

- Everything users and members can do.
- Create a category.
- Create an ingredient.
- Delete any comment. (For moderation purposes only).

5.3.2 Tasks

The objectives set for the user are:

- Create an account.
- Log into the newly created account.
- Browse all recipes.
- Browse a single recipe.
- Bookmark a recipe.
- Comment on a recipe.
- Delete the comment, and comment again.
- Create a recipe.
- View the newly created recipe.
- Edit the recipe.
- Comment on the recipe.
- Bookmark the recipe.
- View all your bookmarks saved to your account.
- Go back to recipe you just created.
- Delete the recipe.
- View all your bookmarks again.
- Log out.

6.4 Conclusion

The testing chapter encompasses both user testing and functional testing of the application. Functional testing was conducted to find out that the application meets the required standards and has the functionalities outlined in the earlier section of this document, namely, the Requirements Chapter. The results of the functional testing indicate that all implemented functionalities are working as intended, with no deviations from expected standards. The focus, therefore, is on enhancing the application's feature set by implementing more features.

7 Project Management

7.1 Introduction

This chapter describes how the project was managed and how well the group worked together as a team. It shows the phases of the project, going from the project idea through the requirements gathering, the specification for the project, the design, implementation, and testing phases for the project. It also discusses, GitHub and google drive as tools which assist in project management.

7.2 Project Phases

7.2.1 Proposal

During the project's initial stage, a decision was required regarding the application area and type that would be developed. Initially, the plan was to create a recipe sharing application to compare different front-end frameworks, particularly React and Svelte, in an attempt to highlight the advantages and disadvantages of each framework. However, due to time constraints, the plan was changed, and creating a recipe sharing platform with in-depth features was chosen instead.

7.2.2 Requirements

In the requirements chapter of the project, I initially aimed to develop a comparative study of two frontend frameworks, React and Svelte, but later decided to shift my focus onto React and develop a more functional web application as the end product. In order to gather inspiration for my application, I analysed similar applications such as epicurious.com and food.com, taking note of their various features and functionality, both positive and negative, to help guide the development of my own application.

7.2.3 Design

During the design phase of the project, a lo-fidelity wireframe was created as a preliminary outline for the application's design. A high-fidelity prototype was then developed in Figma to finalize the main design components. However, during the development stage, certain design elements changed from the original plan. For instance, the application's theme was initially conceived as a "light mode" design, which it still maintains but was ultimately developed using a "dark mode" default, depending on the option selected in the user's operating system settings, the default theme will swap for accessibility reasons.

Other design changes included modifications to the homepage and the single recipe show page, such as removing the second featured recipe from the former and removing the "what you need" section while adjusting the nutritional information section found in the Figma prototype on the latter. Furthermore, the comment section was added to the bottom of the single recipe show page, which was inadvertently overlooked during the original prototyping phase. Lastly, the login and registration modal were changed so that they are now displayed in separate tabs instead of being presented simultaneously.

7.2.4 Implementation

During the implementation phase of the project, ReactJS was employed as the frontend JavaScript library, while the backend was built with Express.js. Since no external API was integrated for the recipes and ingredients, they had to be manually added to the database. Although it is possible to add recipes and ingredients via the application when logged in, adding ingredients requires admin privileges. While an external API could have been used, it was decided against due to the unavailability of a suitable API that met the application's needs. Additionally, this approach could delay the development of the application due to modifications needing to be made. However, a rest countries API was utilized for user registration, providing easy access to essentially all countries worldwide. A downside to this approach is that the API is subject to occasional maintenance downtime, during which users may be unable to choose their location. To avoid complete user registration failure during these periods, the application itself provides a "Not Specified" option.

7.2.5 Testing

Testing was conducted thoroughly throughout the development process of the application to ensure the correct functioning of various features as they were added. However, due to the interdependency of various components, adding new features sometimes required adjustments to other components, which could create a ripple effect and necessitate modifications to unrelated components.

7.3 Teamwork

7.3.1 Difficulties

As the development of my web application was centred around a specific content genre, which is recipes, I found that only a few external APIs were necessary. However, I also recognized the potential benefits of integrating additional APIs such as those related to ingredients, recipes, and other related resources, which would have reduced the need for manual input.

7.4 SCRUM Methodology

The implementation of the SCRUM methodology for my project was satisfactory. However, it did not prove to be as advantageous for a solo project as it would have for a team project, where it could have facilitated keeping team members up to date and on track with their assigned tasks. Nonetheless, the sprint cycles served as a useful tool in maintaining focus on the objectives that needed to be accomplished within the given timeframe, ultimately contributing to the successful delivery of a functional application.

7.5 Project Management Tools

7.5.1 GitHub

GitHub is a repository hosting service that provides version control and project management functionalities through git. The process involves creating a repository for the project and committing changes to the repository, which can then be accessed from any device. From a practical standpoint, GitHub proved to be an effective tool for maintaining multiple versions of the project and facilitating access to the latest version of the project. Although not fully utilized, the platform's version control feature served as an added benefit to the project management process.

7.5.2 Google Drive

Google Drive is a file storage in the cloud. It was used to store the results of my interviews, which was conducted through Google Forms. Due to Google Forms being a part of Google Drive made it easy to collect the data.

7.6 Reflection

7.6.1 Your views on the project

In my opinion, I feel the development process has gone quite well. The application works as I had intended it, although there are a number of design choices and functionalities I would have preferred to add, such as being able to swap between measurement units, converting grams to teaspoons and so on, as well as some design modifications to give it a more consistent style across the entire application.

7.6.2 Completing a large software development project

Throughout the development of this large software project, I have gained an appreciation and understanding for the different aspects of the process and the wide range of tasks that must be completed in the development. One area for personal improvement is the need to enhance my time management skills and reduce my procrastination, as it may interfere with the timely completion of tasks and finally, although I only did this in a minor aspect for the application, this project has also taught me the importance of integrating the frontend and backend components of a project, along with database and external API functionality, which is a more complex task than utilizing a database or API in isolation.

7.6.3 Working with a supervisor

This project has taught me the value of having a supervisor's input, as they provide an outside perspective on the assignment and offer suggestions that I may not have thought of for the development of the application. There have also been times after discussing certain tasks with my supervisor, I realized that what I initially thought felt like a lot more to do than they turned out to be.

7.6.4 Technical skills

Throughout this project, I feel that I have improved my skill in developing web applications using ReactJS and ExpressJS. By using Axios to access the backend, I have gained the knowledge and experience necessary to combine both frameworks to create a functional web application. This project has not only allowed me to further my skill with regards to frontend and backend web development but has also served as a launching pad for my continued improvement in these areas, providing me with the platform to further improve my skills in the hopes of creating more exciting and efficient web applications in the future.

7.6.5 Further competencies and skills

I believe that gaining a deeper understanding of JavaScript and the advanced features of ReactJS, such as JSX, will greatly enhance my frontend development skills. There is a large number of techniques and possibilities that I did not fully exploit during the development of the recipicker web application, and I am eager to further improve in this area.

7.7 Conclusion

During the project management chapter, I provided an overview of my personal evaluation of the development of the application. I talked about the areas where I could improve my skills to enhance the web application's performance in case of a future redevelopment or work on future projects. Also, I mentioned the usage of tools such as GitHub and Google Drive in the development process and the implementation of the SCRUM methodology. I also highlighted the challenges I encountered during the project. Lastly, I discussed my experience working under the guidance of a supervisor, emphasizing their significance in providing direction and recommendations.

8 Conclusion

The overall aim of this web application is to provide an easy to use, user-friendly platform for individuals of all levels of culinary expertise, from beginners to professionals, to easily search for and discover new recipes from various cuisines around the world. The key technologies used in the development of this application include ReactJS, ExpressJS, and MongoDB. ReactJS is a JavaScript library that is open-source and used to develop user interface components. ExpressJS is a web application framework that is minimalist, unopinionated open-source framework, that is used for building backend applications. MongoDB is a NoSQL database program that is cross-platform and uses JSON-like documents, this is known as a NoSQL database.

During the design phase of the project, the majority of aspects of the application were planned, ranging from the style guide to the user flow and the application architecture. Although some aspects, such as comments the bookmarks page, were overlooked during the design stage, and had to be addressed subsequently.

In the implementation chapter, the chosen technologies were discussed, and the folder structures of the frontend and backend were shown and discussed.

In the testing phase, both functional and non-functional testing took place. This was done both throughout the development process, as issues were identified and after the development of the application was complete, to ensure no new or unidentified issues had come to light as the development progressed.

For project management, version control was handled through GitHub, while Google Drive was used to store interview forms used for the requirements gathering process.

If the application was to be developed further, additional features could be added, such as error handling on the frontend for forms, improved permissions implementation, recipe filters, and more.

References

- Kaluža, M. (2018, May 3). *COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVELOPMENT*.
- Vyas, R. (2022). Comparative Analysis on Front-End Frameworks for Web Applications. International Journal for Research in Applied Science and Engineering Technology.
- Kumar, A., & Singh, R. K. (2016). Comparative analysis of angularjs and reactjs. International Journal of Latest Trends in Engineering and Technology, 7(4), 225-227.
- *Comparison with Other Frameworks - Vue.js*. (n.d.).
- React vs Angular: Which One is Best for Your Next Front-end Project? (2022, October 19). Radixweb.
- Wanyoike, M. (2019, September 26). History of front-end frameworks. LogRocket Blog.
- Saks, E. (2019). JavaScript Frameworks: Angular vs React vs Vue.
- Kevin Powell. (2022, March 17). *dialog = the easiest way to make a popup modal* [Video]. YouTube. https://www.youtube.com/watch?v=TAB_v6yBXIE
- *HTML details Tag*. (n.d.). https://www.w3schools.com/tags/tag_details.asp
- Usba, N. U. (2022, March 13). Create Dynamic Form Fields in React - Bits and Pieces. *Medium*. https://blog.bitsrc.io/how-to-create-dynamic-form-fields-in-react-45cc2cc7b1b0