SW Engineering CSC 648-05 Spring 2023

RecipeReel

T03 Milestone 4

Team lead: Yueling Liu yliu50@sfsu.edu

Backend lead: Duncan Herington, Marcel Azouri

GitHub lead: Nathan Le Howland

History table

M4V1	May 18, 2023
M3V2	May 18, 2023
M3V1	April 27, 2023
M2V2	April 27, 2023
M2V1	April 3, 2023
M1V2	April 3, 2023
M1V1	March 2, 2023

Table of Contents

1. Product Summary

- Product Name: RecipeReel
- Selling our product:

  Welcome to the future of recipe sharing !RecipeReel is not just an average recipe website; it's a dynamic and engaging social platform designed to connect food enthusiasts from all walks of life. At RecipeReel, we believe that cooking is more than just following instructions; it's an art form that should be shared, and enjoyed together. Our platform goes beyond the traditional recipe website and it creates a vibrant and interactive space where food lovers can connect, inspire, and be inspired with one another. It provides a user-friendly interface that feels like a social media platform, users can browse recipes, they can rate a recipe,leaving a comment to engage with the author who posted the recipes when they are signed up and logged in. We are aiming to make cooking a social experience, building connections with like-minded individuals who share their love for cooking.

- Final Priority 1 Functions
  - General User:
    - A general user shall be able to register for an account on RecipeReel using a username, password, and email
    - A general user shall be able to search using a keyword or terms that match a recipe's ingredients or title
    - A general user shall be able to view recipes, details, and comments.
  - Registered User:
    - A registered user shall be able to log in with an email and password.
    - A registered user shall be able to leave comments on recipes.
    - A registered user shall be able to pst recipes to RecipeReel
    - A registered user shall be able to log out.
    - A registered user shall be able to follow other registered users.
    - A registered user shall be able to unfollow other registered users.

        - A registered user shall be able to delete a recipe they posted to RecipeReel.

- What makes RecipeReel unique?

  RecipeReel believes that cooking is not just about the food, but about community. In order to reflect this, RecipeReel behaves more like a social media platform rather than a traditional recipe website. This could help others discover the best recipe and create a fun and engaging and interactive space where food enthusiasts can come together to share their passion for cooking.

- URL to RecipeReel: https://master.d3u7tlcu99u8gi.amplifyapp.com/

2. Usability Test Plan

Usability Test Plan

The purpose of this usability test is to evaluate the usability of the five main functions of RecipeReel. The five functions to be tested are: Post a recipe, following a user, searching for a recipe based on keywords,rating on a recipe, and deleting a post of the recipe. The test measures the effectiveness, efficiency, and user satisfaction of each function.

- Post a Recipe
    - Test Objective:

      This test checks each post contains a recipe name, description, cooking time, difficulty level, ingredients, instructions and category. The reason it checks these inputs is that the recipe name is essential as it helps other users quickly find the recipe they are interested in by enforcing having a recipe name. And the description provides a brief overview of the recipe, which helps users understand if the recipes are the one they are looking for before diving into it. Cooking time allows the user to plan time accordingly. Difficulty level is crucial for users to gauge whether a recipe aligns with their cooking skills and experience. The list of ingredients is vital as it is an essential and necessary component for the recipe. This also helps users to search recipes by entering ingredients. We test the instructions as it provides clear directions on how to prepare the recipe, step by step instruction ensures users can follow along easily and achieve cooking their desired meal. Testing category helps users explore recipes based on their preferences, or specific cuisine. Images of recipes are highly beneficial and it makes the recipe appealing, but it is an optional. By enforcing these inputs in each post, the website can provide users with comprehensive and consistent information across all recipes.
    - Test Description:

      System Setup:

The system being tested is our website that allows registered users to post their own recipes. Our website has a user registration and login process in place to ensure that only authorized users can access the posting functionality. The system is accessible via a specific URL provided for testing purposes.

Starting Point:

The starting point for this test is the recipe posting page. This page serves as the entry point for users to enter information about their recipes and publish them on the website. Users are expected to navigate to this page after logging into the system.

Intended Users:

The target users for this test are registered users of the recipe website who wish to post their recipes. These users have already completed the registration and login process.

Measurements:

Time: The time it takes for a user to post a recipe will be measured. This includes the time spent on filling out the necessary fields, uploading images, and submitting the recipe. This measurement helps assess the efficiency of the posting process.

Clicks: The number of clicks made by a user during the recipe posting process will be recorded. This measurement helps evaluate the user interface's intuitiveness and the ease of navigating through the required steps. And the clicks are varies and depend on the complexity and difficulty of the recipe.

Ease of Process: The overall ease of the posting process will be evaluated, taking into account factors such as clarity of instructions, intuitiveness of the user interface. This measurement helps determine the user-friendliness of the recipe posting feature.

By conducting this test, we aim to ensure that the recipe posting process is user-friendly, efficient, and meets the expectations of registered users. These collected measurements will be used to identify any potential issues, improve the user experience, and enhance the overall functionality of the recipe posting feature on the website.

URL: https://master.d3u7tlcu99u8gi.amplifyapp.com/post-recipe (It only for registered user)

- Comment on a posted recipe
    - Test Objective:

      The objective of this test is to ensure that registered users can leave comments on other users' posted recipes. We test the features below:

        1. Testing if a user enters a valid comment in the input field
        2. Testing the “Post Comment” button works properly
        3. Testing that the comment is successfully added to the recipe post
        4. Testing if the comment appears in the correct order

      This reason for testing commenting on other users' posted recipes focuses on fostering user engagement, as it plays a crucial role in creating interactions and building a sense of community among users. This aspect of the website aims to differentiate it from traditional recipe websites and provide a social media-like experience for users, and this is one of the unique features of our website. It also helps admins to evaluate if these comments are relevant to recipes.

    - Test Description:

      System Setup:

      The system being tested is our website that allows registered users to comment on other’s user’s posted recipes.

      Starting Point:

      The starting point of this test is a recipe post displayed on the home page or when a user searches for a recipe and that displays on the application's user interface.

Intended Users: The target users for this test are registered users of our recipe website who wish to comment on other users' posted recipes. These users have already completed the registration and login process.

Measurements:

Time: The time it takes for a user to comment on a recipe will be measured. This measurement helps assess the efficiency of the commenting process and evaluate their engagement with one another.

Comment Quality: Evaluating the quality of comments can be subjective but can provide insights into the level of user interaction and meaningful discussions.

User Engagement: Measuring user engagement who leave comments, the frequency of return visits by users to the recipe page, and the duration spent on the page. These help determine the overall user interest and involvement with the commenting feature.

- Searching for a recipe based on keywords
    - Test Objective: the objective of this test is to ensure that both registered and unregistered users can efficiently search for recipes using keywords, such as recipe names or ingredients.We test the features below:
        1. Testing the performance of the search
        2. Testing if the search query valid input returns correct results

      This reason for testing search is to make sure that all users have an efficient way of finding specific recipes that match their query. This will save the user time and enhance their experience.

    - Test Description:

      System Setup:

      The system being tested is the search feature, which is in the navigation bar. This search feature handles keywords to query recipes for the user.

      Starting Point:

The starting point of this test is any page on the website as the search bar is located in the nav bar which is on every page.

Target users:

The targeted users are all users on the website, that includes registered and unregistered users who want to search for specific recipes

Measurements:

Search Accurate:

Measure the accuracy of search results by comparing the relevance of the displayed recipes to the user's search query. This can be assessed through manual evaluation or automated methods, such as comparing search keywords with recipe titles, descriptions, and tags.

Search Completion Time:

Measure the time taken for the search feature to display relevant results after the user enters their search query. This helps assess the speed and efficiency of the search functionality.

URL: https://master.d3u7tlcu99u8gi.amplifyapp.com/ type ingredients in search bar field

- Rate a posted recipe
    - Test Objective: To test whether registered users can rate recipes and whether there are problems in the scoring process. We test:
        1. Regular users cannot rate recipes, since only registered users can rate them.
        2. Registered user can rate successfully

      The reason we test this is to gather feedback on the quality of the posted recipes. This function can provide valuable insights into what users find valuable, interesting. Rating systems can encourage user engagement and participation, and enhance user's experience.
    - Test Description:

      System Setup:

The system being tested is our website that allows registered users to rate on other user's posted recipes.

Starting Point:

The starting point of this test is a recipe post displayed on the home page or when a user searches for a recipe and that displays on the application's user interface.

Target users:

The target users for this test are registered users of our recipe website who wish to rate on other users posted. These users have already completed the registration and login process.

Measurements:

Engagement: We measure if the posted recipe encourages other users' engagement

Satisfaction: By testing rating a posted recipe, we measure the satisfaction for users and quality of the recipe.

- Deleting a posted recipe
    - Test Objective: To test whether registered users can delete their own post or recipe content. This test verifies that the delete functionality is working properly and that users encounter any issues when attempting to delete a post.
        1. Test if the recipe actually exists, if the recipe does not exist we cannot delete it
        2. Test if the user that requests to delete the recipe is the owner of the recipe, if the user is not the owner of the recipe than they should not be allowed to delete it
        3. Test that the recipe has been deleted, we need to make sure that all data tied to this single recipe is deleted.

    - Test Description:

      System Setup:

The system being tested is our server. The front end sends data to the server in the form of URL params and request body. The server uses this data to delete data in the DB.

Starting Point:

The starting point for this test is the front-end client. The user must find a post that they own and delete it. The client then sends a DEL request to the server along with some data.

Target users:

The target users for this test are registered users of our recipe website who wish to delete their own posted recipes. These users have already completed the registration and login process.

Measurements:

Successful Deletion:

We need to make sure the post is deleted successfully. All the information such as recipe name, descriptions, comments, ratings, and images, are completely removed and no longer displayed on the user's interface after deleting a recipe.

Data Integrity:

We need to make sure the right recipe is deleted. We also need to delete all tables that have a FK of recipe_id. The measure of accuracy is if all this data is deleted from DB.

Usability Test Table - Effectiveness

Test Case	Completed %	Errors	Comments
Posting a Recipe	90%	Only 1 Image for each recipe is stored	We intended to allow user post a recipe with 1-6 maximum images, however we are able to store 1 image only for each recipe
Commenting on other user's posted recipes	80%	insert into "comments" for type integer is "undefined" for these posted recipes displayed on homepage	Commenting on recipes that have been searched from search bar worked and displayed on the comment filed
Searching for a recipe based on keywords	100%	None	It works when user search from home page and enter ingredient keywords
Rating a Post	0%	None	We haven't implemented this feature in this milestone
Deleting a Post	50%	It doesn't rendering in the front end	We have delete a post api set up and it worked, however we haven't completed this feature

Usability Test Table - Efficiency

Test Case	Time to Completion	Number of Clicks	Ease of Use	Comments
Post a Recipe	Easy recipe: 5-12 minutes; Intermediate recipe: 10-25 minutes: Difficult recipe: 15-25 minutes or even longer	Easy recipe: 30-45 clicks Intermediate recipe: 45-55 clicks Difficult recipe: up to 55 clicks	Satisfied	This test varies, and it depends on how difficult the recipe is. The more ingredients and instructions the longer it takes and the more clicks.
Commenting on other users posted recipes	Within 1 minute	2	Satisfied	It only worked for a searched recipe that displayed on the user's interface.
Searching for a recipe based on keywords	Within 1 minute	2	Very Satisfied	User can enter a ingredient keyword, such as rice, it will display all the recipes that contains rice
Rating a Post	Within 1 minute	2-5	Not Satisfied	Not working
Deleting a Post	Within 1 minute	2	Not Satisfied	Not working in the front end

User Satisfaction

User 1:

	Very Satisfied	Satisfied	Neutral	Unsatisfied	Very Unsatisfied
Posting a recipe was easy to understand		✓			
I felt confident that my recipe was posted successfully.		✓			
The restrictions on the number of images and characters were reasonable.			✓		
Finding and following other users was easy.					✓
The updates about the following users' content was clear and helpful.				✓	
Overall, I was satisfied with the process of following a user.					✓
The search results were relevant to the keywords entered.	✓				
I was able to filter and refine my search results as needed.				✓	
Overall, I was satisfied with the search functionality.	✓				
Rating a recipe was easy.					✓
The rating system was fair and accurate.					✓
I felt that my rating would be helpful to other users.				✓	
Deleting a post was easy to understand.		✓			
I felt confident that my post was successfully deleted.		✓			
Overall, I was satisfied with the ability to delete my own posts.			✓		
Commenting on a post was easy to understand	✓				
I felt that my comment was posted successfully		✓			

Overall, I was satisfied with the ability to comment on a posted recipe .			✓		

User 2:

	Very Satisfied	Satisfied	Neutral	Unsatisfied	Very Unsatisfied
Posting a recipe was easy to understand	✓				
I felt confident that my recipe was posted successfully.	✓				
The restrictions on the number of images and characters were reasonable.				✓	
Finding and following other users was easy.					✓
The updates about the following users' content was clear and helpful.				✓	
Overall, I was satisfied with the process of following a user.					✓
The search results were relevant to the keywords entered.	✓				
I was able to filter and refine my search results as needed.					✓
Overall, I was satisfied with the search functionality.	✓				
Rating a recipe was easy.					✓
The rating system was fair and accurate.					✓
I felt that my rating would be helpful to other users.			✓		
Deleting a post was easy to understand.		✓			
I felt confident that my post was successfully deleted.			✓		
Overall, I was satisfied with the ability to delete my own posts.			✓		
Commenting on a post was easy to understand	✓				

I felt that my comment was posted successfully	✓				
Overall, I was satisfied with the ability to comment on a posted recipe .		✓			

3. QA Test Plan

3.1. QA Test 1

3.1.1. Test Objectives: “Create Post” required fields.

3.1.2. HW & SW Setup: Windows computer on Google Chrome

3.1.3. Feature to be tested: Checking to see if all fields are filled out before a post can be made.

Test #	Description	Input	Expected Output	Results
1	Test to see if post can be created with only recipe name, ingredients and instructions	Filled fields: Recipe name Ingredients Instruction	A warning error: “Oops! It looks like you forgot to select a cooking time for your recipe”	Pass
2	Test to see if the post can be created with recipe name, ingredients, instructions, cooking time, and difficulty filled out.	Filled fields: Recipe name Cooking time Difficulty Ingredients Instruction	A warning error: “Oops! It looks like you forgot to enter a name for your recipe”	Pass
3	Test to see if posts can be created if all fields are filled out.	Filled fields: Recipe name Description Cooking time Difficulty Ingredients Instruction Category	A warning error: “~ops! It looks like you forgot to select a cooking time for your recipe”	Pass

3.2. QA Test 2

3.2.1. Test Objectives: Creating a password

3.2.2. HW & SW Setup: Windows computer on Google Chrome

3.2.3. Feature to be tested: Checking to see if a password is greater than or equal to 6, less than or equal to 20, and contains a special character.

Test #	Description	Input	Expected Output	Results
1	Test password less than 6 characters	"hi!"	"Password must be 6-20 characters long and contain a special character."	Pass
2	Test password greater than 20 characters	"PrettyPleaseGiveUsGoodGradeProfessor!!!"	"Passwords must be 6-20 characters long and contains a special character."	Pass
3	Test password greater than 6 characters, less than 20 characters, and containing a special character	"Pass123!"	"created account successfully"	Pass

3.3. QA Test 3

3.3.1. Test Objectives: Load times

3.3.2. HW & SW Setup: Windows computer on Google Chrome

3.3.3. Feature to be tested: Check to see how fast loading parts of the website.

Test #	Description	Input	Expected Output	Results
1	Test load time logging in	Login	Less than 10 seconds	Pass
2	Test load time	Top Rated	Less than 10	Pass

	going to “Top Rated”		seconds	
3	Test load time of search	Chicken	Less than 10 seconds	Pass

3.4. QA Test 4

3.4.1. Test Objectives: Works on other browsers

3.4.2. HW & SW Setup: Windows computer on Google Chrome

3.4.3. Feature to be tested: Make sure the website is functional on a variety of browsers.

Test #	Description	Input	Expected Output	Results
1	Test if RecipeReel works on Google Chrome	Google Chrome	Working	Pass
2	Test if RecipeReel works on Microsoft Edge	Microsoft Edge	Working	Pass
3	Test if RecipeReel works Firefox	Firefox	Working	Pass

3.5. QA Test 5

3.5.1. Test Objectives: Works on other operating systems

3.5.2. HW & SW Setup: Google Chrome

3.5.3. Feature to be tested: Make sure the website is functional on a variety of operating systems

Test #	Description	Input	Expected Output	Results
1	Test if RecipeReel work on Mac	None	Working	Pass
2	Test if RecipeReel work on Windows	None	Working	Pass

3	Test if RecipeReel work on Linux	None	Working	Pass

4. Code Review

4.1. Coding Style:

Our application is using the standard React framework, and we have followed:

- Indentation and formatting :
    - We used used “Prettier”extension on Visual Studio Code that make sure a consistent indentation style of two spaces
    - Used proper line breaks and indentation to improve code readability

- Naming conventions:
    - We followed the convention using camelcase for handler functions and variables(e.g, handlePostRecipe, recipeImage)
    - We used Reserve PascalCase for React component names (e.g., CategoryCard and PopularDIshesCards in our application)

- Component Lifecycle and Hooks:
    - Follow the recommended guidelines for using useEffect, useState, useContext, and other React Hooks

- Styling:
    - Used SCSS to styling pages that support nesting, allowing nest CSS selectors within one another. This helps improve readability and maintainability by grouping related styles together. We used used media query to ensure website is responsive with different size of devices
    - Used Bootstrap that helped us quickly build responsive pages,it simplified the process of creating a consistent user interface

4.2. Chose the code

```
import React, { useEffect, useState } from 'react';
import DashboardCard from '../components/Cards/DashboardCard';
import Filterbar from '../components/filterbar/Filterbar';
import { useHistory } from "react-router-dom";
```

```
// MUI
import MenuItem from '@mui/material/MenuItem';
import FormControl from '@mui/material/FormControl';
import Select from '@mui/material/Select';

// Bootstrap
import Col from 'react-bootstrap/esm/Col';
import Row from 'react-bootstrap/esm/Row';
import Container from 'react-bootstrap/esm/Container';

const Search = ({ location }) => {
 const [results, setResults] = useState([]);
 const [title, setTitle] = useState('');

 const [filter, setfilter] = React.useState('');

 const handleChange = (event) => {
   setfilter(event.target.value);
 };

 const history = useHistory();

 useEffect(() => {
   const fetchResults = async () => {
     const query = new
URLSearchParams(location.search).get('query');
     try {
       const response = await fetch(
         `${process.env.REACT_APP_REQ_URL}/search?query=${query}`
       );
       const data = await response.json();

       if (response.ok) {
         setResults(data.results);
         console.log("Results:", data.results[0].recipe_title);

         setTitle(data.results[0].recipe_title);
         console.log("Recipe title:", title);

       } else {
         throw new Error(data.error);
       }
     } catch (err) {
       console.error(err);
     }
   };

   fetchResults();
 }, [location.search]);

  const handleDashboardCardClick = (recipe_id) => {
   history.push(`/post/${recipe_id}`);
 };

 return (
   <>
     <Container>
       <Filterbar title={title}/>

       {results.length > 0 ? (
         results.map((result) =>
          <DashboardCard
             key={result.recipe_id}
             result={result}
             onClick={() =>
handleDashboardCardClick(result.recipe_id)}
           />)
       ) : (
         <p style={{textAlign: 'center'}}>No results found.</p>
       )}
     </Container>
   </>
 );
};

export default Search;
```

Why did we choose this code?

This is one of the main functions of our website. We used React Components that handles search functionality and renders a list of searches using a card.

Code review feedback from a peer in our team:
Grouped the MUI imports and Bootstrap imports separately to improve code readability which is a good practice.
It seems the MenuItem, FormControl, and Select imports from MUI are not used in the code. It should be removed to avoid unnecessary imports.
Error Handling seems to only catch any errors that occur during the fetch request, and only log the error to the console. Maybe adding error handling to displaying an error message to the user would be a better user experience.
Overall, the code looks well-structured and readable. The useEffect hook is used correctly to fetch data based on the search query.

Code review feedback from Team02 :
Code looks well formatted and looks fairly self-explanatory. However, the overall purpose of the code is a little unclear since there are no function headers or inline comments. And just a minor thing, but you use both "useState" and "React.useState". Is there a reason for using one and then the other?

5. Self-check: Best practices for safety (½ page) -

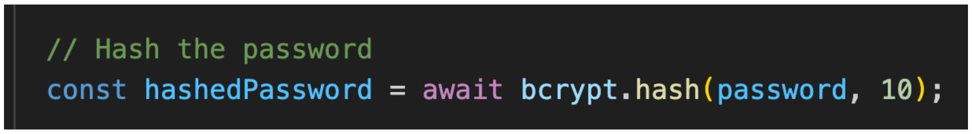
5.1. Major protected assets

5.1.1. User password:

To encrypt the password in the DB we use bcrypt to encrypt the password. (see screenshot 1)

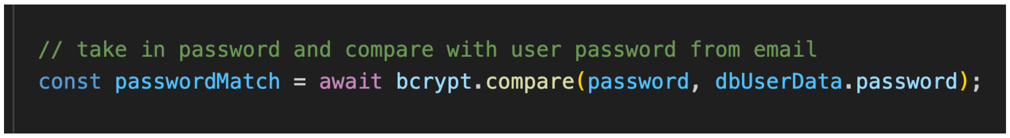
We store this hashed password to the database. When a user logged in we use bcrypt to compare our hashed password to the user supplied password. If there is a match, the user is validated.(see screenshot 2)

For example, when a user registers an account with a username : nate, password: pass123, it will be hashed and stored an encrypted string in the password column in our database. (see screenshot 3)



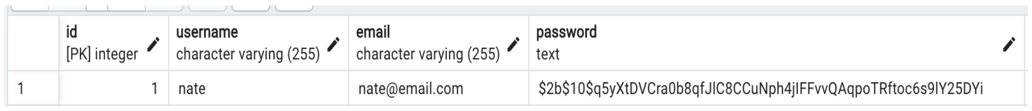
```
// Hash the password
const hashedPassword = await bcrypt.hash(password, 10);
```

Screenshot1 : using bcrypt



```
// take in password and compare with user password from email
const passwordMatch = await bcrypt.compare(password, dbUserData.password);
```

Screenshot 2: comparison with user password



	id [PK] integer	username character varying (255)	email character varying (255)	password text
1	1	nate	nate@email.com	$2b$10$q5yXtDVCra0b8qfJlC8CCuNph4jIFFvvQAqpoTRftoc6s9lY25DYi

Screenshot 3: encrypted password in database

5.2. Input data validation list

5.2.1. Username:

We have a validation for the username input, which occurs in both the frontend and backend to ensure data integrity.

In the front end, the username must be a minimum of 6 characters long and a maximum of 20 characters long. This is to enforce a specific length requirement for usernames.
We used a if statement to check the length of password:

```
   if (userName.length < 6 || userName.length > 20) {
     errors.username = 'Username must be between 6 and 20
characters';
     setValidationErrors(errors); // set validation errors state
     console.log(setValidationErrors(errors));
     toast.error('Username must be between 6 and 20 characters', {
       position: toast.POSITION.TOP_CENTER,
     });
     return;
   }
```

In the back end, our system uses a regular expression pattern to validate the username. It checks if the username consists of any alphanumeric characters or underscore and hyphen. And it has to be at least 6 characters long but not more than 20 characters long.

```
const usernameRegex = /^[a-zA-Z0-9_-]{6,20}$/;
if (!usernameRegex.test(username)) {
     console.log('Invalid username from Backend');
     return res.status(400).json({
       message:
         'Username must be 6-20 characters and may only contain
letters, numbers, underscores, and hyphens.',
     });
   }
```

5.2.2. Password:

We have a validation for the password input, which occurs in both the frontend and backend to ensure data integrity. The password must be a minimum of 6 characters long and a maximum of 20 characters long with a special character.

We used regular expressions to validate passwords in both frontend and backend.

In the front end we check if the password matches this regular expression pattern.

```
  if (!password.match(/^(?=.*[\W_])[a-zA-Z0-9\W_]{6,20}$/)) {
    errors.password =
      'Password must be 6-20 characters long and contains a
special character.';
    toast.error(errors.password, {
      position: toast.POSITION.TOP_CENTER,
    });
    return;
  }
```

In the backend, we have a const passwordRegex variable and we check if the input matches this pattern:

```
  const passwordRegex = /^(?=.*[\W_])[a-zA-Z0-9\W_]{6,20}$/;
  if (!passwordRegex.test(password)) {
    console.log('Invalid password from Backend');
    return res.status(400).json({
      message:
        'Password must be a combination of letters, numbers, and
special characters, with a maximum length of 20 characters.',
    });
  }
```

5.2.3. Email:

We have a validation for the email input, which occurs in both the frontend and backend to ensure email integrity.
In both front end and back end we used a regular expression used to validate email addresses.

In the front end we check if the email matches this regular expression pattern.

```
  if (!email.match(/^[^\s@]+@[^\s@]+\.[^\s@]+$/)) {
    errors.email = 'Please enter a valid email address';
    toast.error(errors.email, {
      position: toast.POSITION.TOP_CENTER,
    });
    return;
  }
```

In the back end, we have a const emailRegex variable and we check if the input matches this pattern:

```

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailRegex.test(email)) {
    console.log('Invalid email from Backend');
    return res.status(400).json({ message: 'Invalid email
address.' });
  }

```

5.2.4. Policy and agreement

The policy and agreement checkbox on a register page when a user registers an account on our website, this checkbox must be selected to indicate their acceptance and agreement to the website's terms of service, privacy policy, or any other legal agreements or policies that govern the use of our website.

For policy and agreement input, we only need to validate this input in the front end.

```

const [isChecked, setIsChecked] = useState(false);

 if (!isChecked) {
    toast.error(
      'Please agree to the Privacy Policy before submitting the
form.',
      {
        position: toast.POSITION.TOP_CENTER,
        className: 'toast-message',
      }
    );
    return;
  }
```

5.2.5. Search bar

Search bar validation involves functions that work correctly and meets specific criteria which are ingredients for our website. Our system checks if it is an empty query and displays a prompt No results found message.

We are validating the search accuracy which we first need to validate the search functionality to retrieve recipes that match the provided query correctly.

Code we used:

```
 static async search(query) {
   // return await knex('recipes')
   // .join('categoriesToRecipe','categoriesToRecipe.recipeID',
'recipes.ID')
   // .join('categories', 'categories.ID',
'categoriesToRecipe.categoryID')
   // .where('category', 'like', `%${query}%`)
   // .orWhere('title', 'like', `%${query}%`);
   const searchQuery = `
   WITH search_result AS (
     SELECT DISTINCT r."id" AS recipe_id, r."title" AS
recipe_title, r."description" AS recipe_description,
r."created_at" AS recipe_CreateAt, u."username" AS userName
     FROM public.recipes r
     JOIN public.users u ON r."user_id" = u."id"
     LEFT JOIN public.categories_to_recipe ctr ON r."id" =
ctr."recipe_id"
     LEFT JOIN public.categories c ON ctr."category_id" = c."id"
     LEFT JOIN public.comments cm ON r."id" = cm."recipe_id"
     LEFT JOIN public.ingredients i ON r."id" = i."recipe_id"
     LEFT JOIN public.instructions ins ON r."id" = ins."recipe_id"
     LEFT JOIN public.ratings rt ON r."id" = rt."recipe_id"
     WHERE (
         r.title ILIKE '%${query}%'
```

```
        OR r.description ILIKE '%${query}%'
        OR c.category ILIKE '%${query}%'
        OR u.username ILIKE '%${query}%'
        OR cm.comment ILIKE '%${query}%'
        OR i.ingredient ILIKE '%${query}%'
        OR ins.instruction ILIKE '%${query}%'
    )
  )
  SELECT * FROM search_result
  ORDER BY recipe_id ASC;
`;
  const { rows } = await client.query(searchQuery);
  return rows;
}
```

6. Self-check: Adherence to original non-functional specs

6.1. System requirements

6.1.1. The website shall be able to use client – server. DONE

6.1.2. The website shall be able to handle when visitor scale is high. DONE, AWS handlings load balancing

6.1.3. The website shall be able to support both cellular and Wi-Fi networks - DONE

6.1.4. The website shall be able to support any user using any web browser. DONE

6.1.5. The database host shall be able to have a web server to host. DONE RDS is hosting the DB

6.1.6. The database host network shall be able to have good security. DONE

6.2. Performance requirements

6.2.1. The website reload shall be able to provide fast service. DONE

6.2.2. The website shall be able to have minified code to reduce file size.

6.2.3. The website shall be able to optimize image quick. DONE

6.2.4. The website shall be able to use a performance monitoring tool to catch bottlenecks. DONE, AWS amplify and EC2 provide monitoring tools

6.2.5. The website database server shall be able to expect multiple users without any delay. DONE

6.2.6. The website database server shall be able to expect multiple requests. DONE, built into AWS RDS

6.3. Privacy

6.3.1. The website shall be able to provide clear and concise policies. DONE

6.3.2. Users shall be able to agree on how their openly information may be shared. DONE

6.3.3. The website shall be able to collect data like name, date of birth, email, and more personal information. DONE

6.3.4. The website shall be able to avoid collecting unnecessary data. DONE

6.3.5. The collected data shall be able to be used for improvement of the website and user experiences. DONE

6.3.6. All confidential user data shall be able store encrypted data. DONE

6.3.7. For more protection firewalls and protection systems shall be able be used. DONE, AWS handles this

6.3.8. Users shall be able to be trained not to share sensitive information. DONE

6.4. Storage

6.4.1. The website shall be able to have a good amount of storage that can handle contests like food images, food recipes, food description and more. DONE, AWS EDS provides sufficient storage for this app at FREE TIER

6.4.2. The website shall be able to support all file formats and sizes for better and effective performance. NONE.

6.4.3. The website shall be able to have a limit on the maximum file size that can be uploaded to prevent excessive resource usage. NONE

6.4.4. The website shall be able to have another storage for emergency backups. NONE

6.5. Security

6.5.1. The website shall be able to use HTTPS. (Secure Transfer protocol). DONE, SSL generated with certbot

6.5.2. The website shall be able to implement security measures to protect user data. DONE, passwords are encrypted

6.5.3. The website shall be able to have a strong password policy. DONE

6.5.4. The website shall be able to use a password transcription method. NONE

6.5.5. The website shall be able to offer multi-factor authentication to prevent unauthorized access. NONE.

6.6. Marketing and Legal requirements

6.6.1. Popular social media sites shall be able to be used to promote websites. NONE

6.6.2. The website shall be able to be user friendly so any user can browse comfortably. DONE

6.6.3. The information provided on the website shall be able to be clear for anyone to understand. DONE

6.6.4. The website shall be able to select a specific target audience for marketing. DONE.

6.6.5. Links of the website shall be able to be pushed through emails and other forms. DONE

6.6.6. The website shall be able to be branded. DONE

6.6.7. The website shall be able to have a name and logo. 80% DONE, this website has a name but not a logo

6.6.8. The website shall be able to accept feedback. NONE

6.6.9. The website shall be able to protect itself-from any legal claims. DONE

6.7. Content

6.7.1. The website will be using a clean and simple sans-serif font as it is easy to read and doesn't detract from the content. DONE

6.7.2. The website will be using a clear hierarchy for headings, subheadings, and other important information. Additionally, using bullet points or numbered lists for ingredients and instructions can make the recipe easier to follow. DONE

7. Team Member Contributions

Team member	Contributions/Activities	Score
Team lead Yueling Liu	• Completed section 1 Product Summary • Completed section 2 Usability Test plan for Post a recipe and comment on a posted recipe • Completed section 3 Usability effectiveness test • Completed section 3 Usability efficiency test • Completed section 3 User Satisfaction Survey with 2 users who are not in CS field • Completed section 4 Code review • Completed section 5.2 Input data validation list • Completed section 6 Non-functional specs checkup • Front end feature for posting a recipe • Added delete post feature in the FE • Delete a post API call in the FE	
Backend lead Duncan Herington	• completed section 2 Usability Test plan for Search recipe • Created comment API • Connected comment API to front end • Assisted with connecting post recipe to API	9
Backend lead Marcel Azouri	• Completed section 1 • Tested Non-functional requirements	7
Github lead Nathan Le Howland	• Completed section 2 Usability Test plan for delete a posted recipe • Completed section 5.1 protected assets • Completed section 6 non-functional specs checkup • Created Post recipe API	9

	● Created Save a recipe API ● Created a like a comment API ● Created a dislike comments API ● Created a get all recipe API ● Maintained database	