SW Engineering CSC648/848

UMA*Me*

Section 04 | Team 03

| **Member** | **Role** |
| --- | --- |
| Khalid Mehtab Khan | Team Lead |
| Dat Vo | Backend Lead |
| Anish Khadka | Frontend Lead |
| Renee Sewak | Scrum Master |
| Jacob Perez | GitHub Master |

Server Link: <http://3.14.254.41/>

Github Repository Link: <https://github.com/CSC-648-SFSU/csc648-04-fall23-csc648-04-fall23-team03/tree/master>

**Directory:**

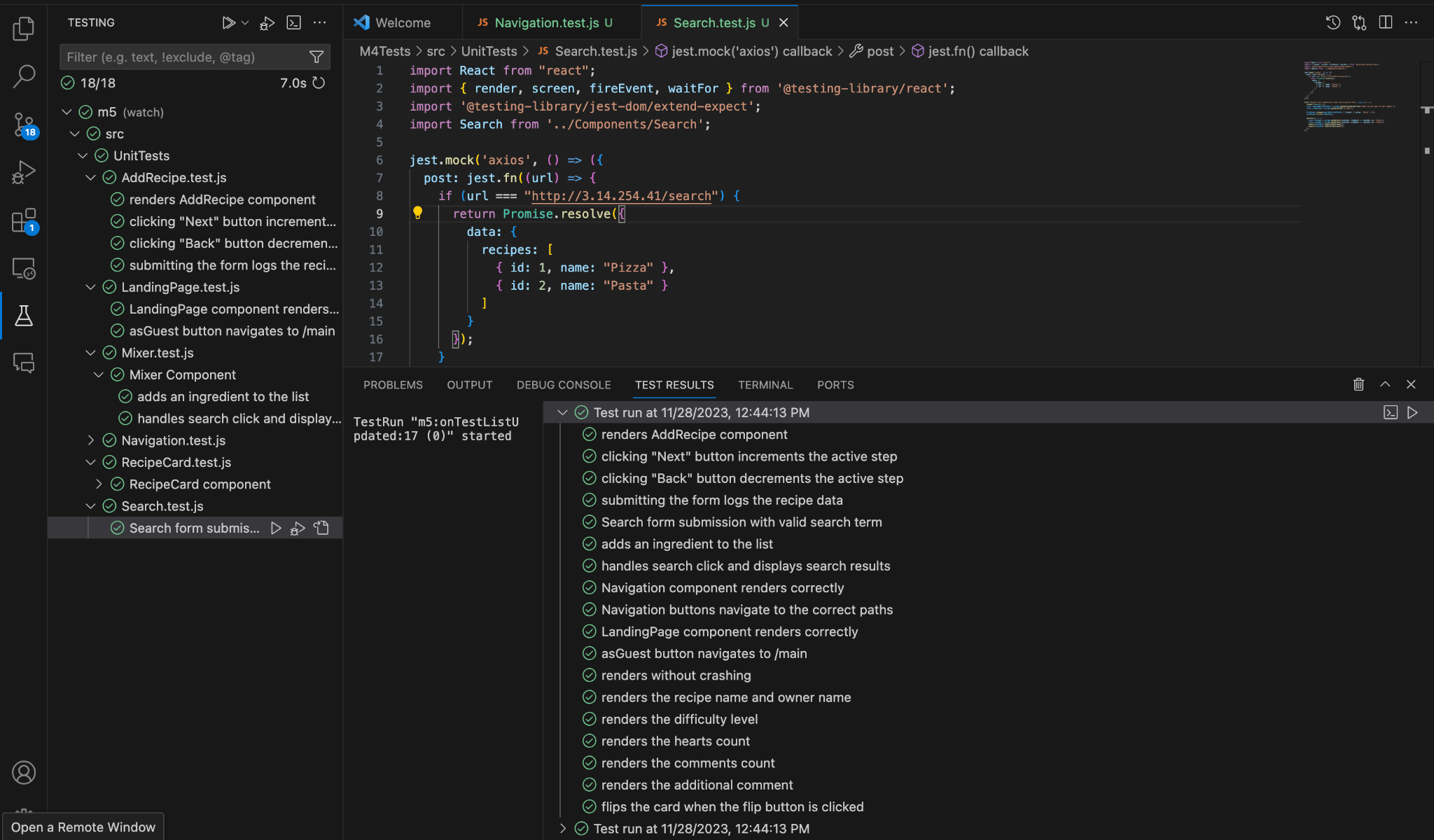
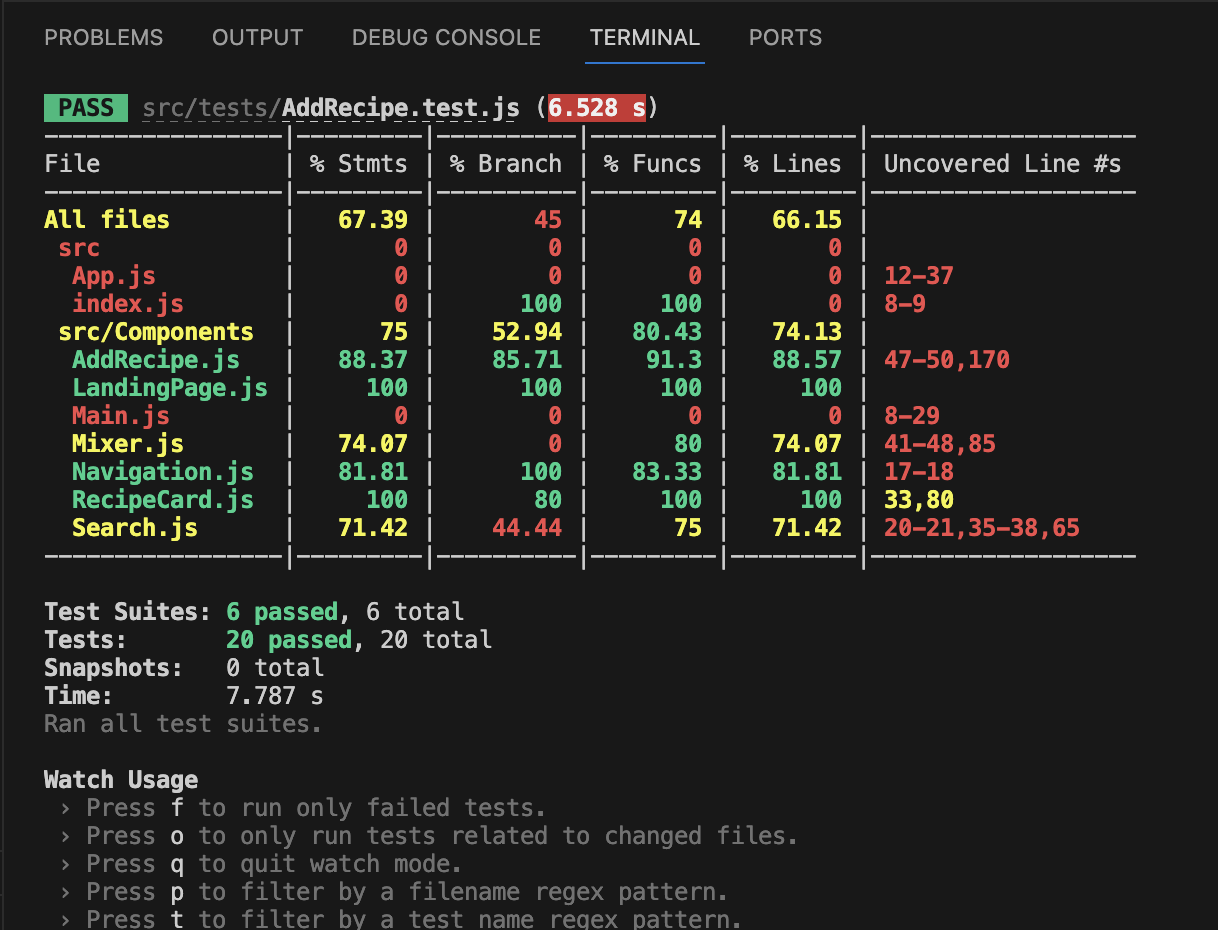
Front end Source Code: master > M4 > UmameFrontEnd

Back end Source Code: master > M4 > UmameBackEnd

Testing % Coverage: master > M4 > UmameTesting

JsDoc and Style Guides: master > M4 > JsDOC and Styles

November 29, 2023

Unit Tests:  
  
  
- All unit tests passed.  
  
 All coverage of unit tests  
The above ss displays the coverage of all the .js files.   
The functional and statement coverage of the p1 features tests that are written are upto the

requirement.  
  
**Add Recipe.js**

The AddRecipe component provides a step-by-step recipe creation workflow,

allowing users to input recipe details, including name, ingredients, instructions, and

additional comments. The user can navigate through each step using a stepper, and  
The data is logged upon submission.

| Functional Coverage: | * Ensures that the AddRecipe component renders successfully. * Validates that the stepper navigation functions correctly, allowing users to move forward and backward through the steps. * Verifies that submitting the form logs the correct recipe data to the console. * Tests the ability to dynamically add and delete ingredients during the recipe creation process. * Tests the ability to dynamically add and delete instructions during the recipe creation process. |
| --- | --- |
| Statement Coverage: | * Verifies that the component structure and stepper management code are executed. * Ensures that functions handling user input, such as handleChange and handleListChange, are executed correctly. * Validates that the functions responsible for adding and deleting items in dynamic lists are executed without errors. * Ensures that the renderStepContent function correctly renders content based on the active step. * Verifies that the handleSubmit function, logging the recipe data, is executed. |

**RecipeCard.js:**  
  
 This feature provides a visually appealing and interactive representation of a recipe card. It displays key information about the recipe, including the recipe name, owner name, image, difficulty level, hearts count, comments count, and additional comments. Users can interact with the card by flipping it to view additional details.

| Functional Coverage: | * - Ensures that the RecipeCard component renders successfully. * - Validates that the recipe card displays key information accurately. * - Verifies that the card flips when the user clicks the flip button, toggling between the front and back views. |
| --- | --- |
| Statement Coverage: | * Ensures that the component structure is correct, and the state (isFlipped) is managed appropriately. * Validates that the flip function is executed when the user clicks the flip button. * Ensures that the content on both the front and back views is rendered based on the provided recipe data. * Verifies that click events, such as flipping the card, are handled correctly. |

**LandingPage.js:**  
  
 The LandingPage serves as the initial interface for users accessing the application. It includes elements such as a welcome message, a name input field, a login button, and an "Continue as Guest'' button. It allows users to navigate in a guest access mode without requiring user authentication. This feature aims to provide a welcoming and user-friendly entry point for both registered users and guests.

| Functional Coverage: | * Ensures the "Welcome" message is correctly displayed, welcoming users to the application. * Verifies the presence and functionality of the name input field, allowing users to enter their names. * Checks the rendering of the login button, providing users with the option to log in. * Verifies the rendering and functionality of the "Continue as Guest" button, allowing users to access the application as guests. |
| --- | --- |
| Statement Coverage: | * confirms that clicking the "Continue as Guest" button triggers the correct navigation to the "/main" route. * Ensures that the code rendering the name input field is executed, making the field available for user input. * Verifies that the code rendering the login button is executed, providing users with a login option. * Checks the code handling the click event on the "asGuest" button, ensuring appropriate responses to user interactions. * Confirms that the code rendering the welcome message is executed, displaying a welcoming message to users. |

**Mixer.js:**  
  
 This component serves as a mixer for searching recipes based on ingredients. Users can input ingredients, add them to the mixer, and then perform a search to find recipes containing those ingredients.

| Functional Coverage: | * The handleAdd function is responsible for adding ingredients to the mixer list when the "Add" button is clicked. * The onSearchClick function performs a search based on the ingredients in the mixer list. * The search results are displayed using the RecipeCard component. * Users can interact with the search bar, "Add" button, and "Mix" button to add ingredients and initiate searches. |
| --- | --- |
| Statement Coverage: | * The first test case ( 'adds an ingredient to the list' ) covers the statement related to adding an ingredient to the mixer list. * The second test case (it('handles search click and displays search results', ...)) covers the statement related to triggering a search and displaying search results. |

**Search.js:**  
  
 The Search component serves as a search interface where users can input a search term for recipes. The search is performed based on the recipe name, and the results are displayed using the RecipeCard component.

| Functional Coverage: | * + The onTermChange function updates the state with the current search term.   + The handleSubmit function handles the form submission and triggers the search based on the entered search term.   + The component uses Axios to send a POST request to a specified API endpoint (http://3.14.254.41/search).   + If the search is successful and recipes are found, they are displayed using the RecipeCard component.   + Appropriate alerts are shown for invalid search terms or errors during the search.   + Users can interact with the search bar and the search button to input search terms and trigger searches. |
| --- | --- |
| Statement Coverage: | * The test case (Search form submission with valid search term) covers the statement related to form submission with a valid search term. It simulates user input in the search bar, triggers the search button click, and verifies that the recipes ("Pizza" and "Pasta") are displayed in the search results. |

**Navigation.js**  
  
 The navigation bar consists of four buttons: Home, Search, Mixer, and Add. Each

button corresponds to a different route within the application, and clicking on a button

triggers navigation to the associated route using the useNavigate hook from

'react-router-dom'.

| Functional Coverage: | * The first test (Navigation component renders correctly) checks whether the component renders correctly. It ensures that each navigation button ("Home," "Search," "Mixer," and "Add") is present in the rendered component. * The second test (Navigation buttons navigate to the correct paths) checks whether clicking on each navigation button correctly updates the URL path using the useNavigate hook from react-router-dom. It verifies that clicking on "Home" navigates to '/main,' "Search" navigates to '/search,' "Mixer" navigates to '/mixer,' and "Add" navigates to '/add.' |
| --- | --- |
| Statement Coverage: | * The Navigation component is a functional component that renders a set of buttons based on the provided icons and labels. * The handleChange function is responsible for updating the state (value) and triggering navigation when a button is clicked. * The component uses the useNavigate hook to update the URL path based on the selected button. * The tests ensure that each button is present in the rendered component and that clicking on each button triggers the expected navigation. |

**INTEGRATION TESTS**

| **Test ID** | **Test Description** | **Prerequisite** | **Test Data** | **Test Scenario** | **Results** | **Date Tested** |
| --- | --- | --- | --- | --- | --- | --- |
| 01 | A test of the website’s login functionality | - User is not already logged into the website | - username: dcv  - password: password | - Navigate to validation page  - Submit credentials | Expected: Site opens  Actual: As expected  Passed: Yes  Expected: User is logged in  Actual: As expected  Passed: Yes | 11/28/2023 |
| 02 | A test of the website’s signup functionality | - User is not already logged into the website  - Provided email is not already in use by another account | - name: Dat Vo  - username: dcv  - email: [dat@mail.com](mailto:dat@mail.com)  - password: password | - Navigate to validation page  - Submit new user information | Expected: Site opens  Actual: As expected  Passed: Yes  Expected: New account is created  Actual: As expected  Passed: Yes | 11/28/2023 |
| 03 | Non-registered users can view recipes | - User is not logged in | N/A | - Navigate to homepage | Expected: Site opens and recipes appear  Actual: As expected  Passed: Yes | 11/28/2023 |
| 04 | Registered users can create and post a recipe | - User is logged in | - name:Chicken Noodle Soup  - ingredients: [  {  "name": "butter",  "amount": 0.5,  "unit": "tbsp"  }],  - tags: ["warm", "winter", "soup"]  - additionalComment: Add salt, pepper, more herbs and spices | - Fill form to create a new recipe  - Recipe appears on homepage | Expected: Form is submitted to backend  Actual: As expected  Passed: Yes  Expected: Newly created recipe appears first on homepage  Actual: As expected  Passed: Yes | 11/28/2023 |
| 05 | Difficulty bar appears on each recipe card | - Must be on a page that displays recipe cards | N/A | - Navigate to homepage | Expected: Recipes appear with complexity bar to indicate difficulty  Actual: As expected  Passed: Yes | 11/28/2023 |
| 06 | Any user is able to search for a recipe using a search bar. | - User can be not logged in | {  "searchType": "name", "searchTerm": "Chicken Noodle Soup"  } | - Search function | Expected: Form is submitted to back end, returns Chicken Noodle Soup  Actual: As expected  Passed: Yes | 11/28/2023 |
| 07 | Any user is able to filter for recipes using recipe tags  for example: ingredients, dishes, food type etc. | - User can be not logged in | {  "searchType": "tag", "searchTerm": "filipino" } | - Search function with specific requirements | Expected: Form is submitted to back end, returns Pork Menudo  Actual: As expected  Passed: Yes | 11/28/2023 |
| 08 | Any user is able to search for another user | - User can be not logged in | {  "searchType": "user", "searchTerm": "Jacob Perez"  } | - Search function with specific requirements | Expected: Form is submitted to back end, returns Jacob Perez user  Actual: As expected  Passed: Yes | 11/28/2023 |
| 09 | All users can input a list of ingredients they have on hand, and the app will point to recipes they can cook with just those ingredients. | - Must have list of ingredients | {  "ingredients":  [  "Chicken", "Ginger", "Water", "Taro Root"  ]  } | - Mixer search | Expected: Form is submitted to back end, returns  Actual: Returns multiple recipes, some of which don’t even have the ingredients  Passed: Yes? | 11/28/2023 |
| 10 | Registered users can leave comments on recipes. | - User is logged in | {  "commentOwner": "653e9a1fba8bd6face7a8568", "commentContent": "I hoped you like it.", "recipeID": "653e9dd6ba8bd6face7a8590"  } | - Leave comment on recipe | Expected: Form is submitted to back end, returns success message  Actual: As expected  Passed: Yes | 11/29/2023 |
| 11 | Registered users can leave hearts on recipes and comments. | - User is logged in | {  "recipeID": "653e9dd6ba8bd6face7a8590"  } | - User likes comment or recipe | Expected: Form is submitted to back end, returns success message  Actual: As expected  Passed: Yes | 11/29/2023 |

2.) Coding Practices

**Coding Style**

Google JavaScript Style Guide

The JavaScript Style Guide provides a set of best practices and conventions for

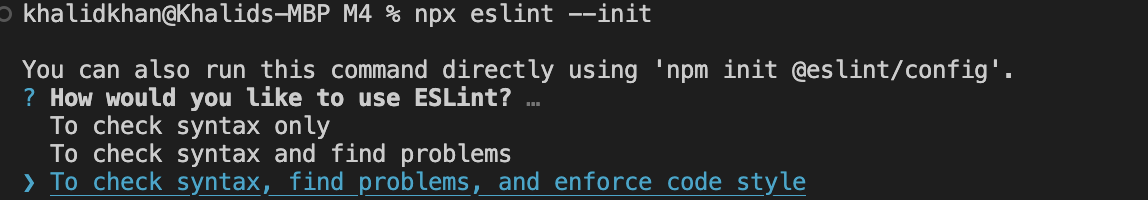
Writing JavaScript code. It includes formatting, naming conventions, comments,

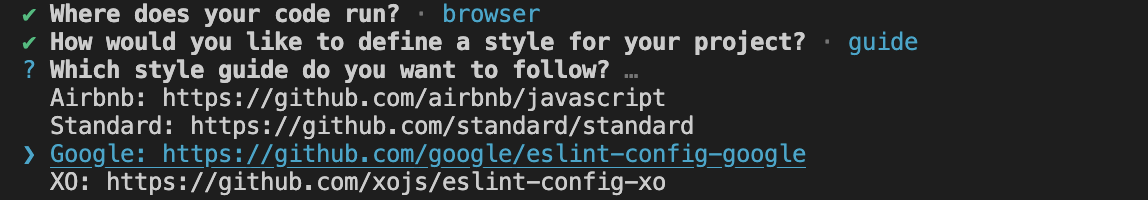
Programming practices and more.

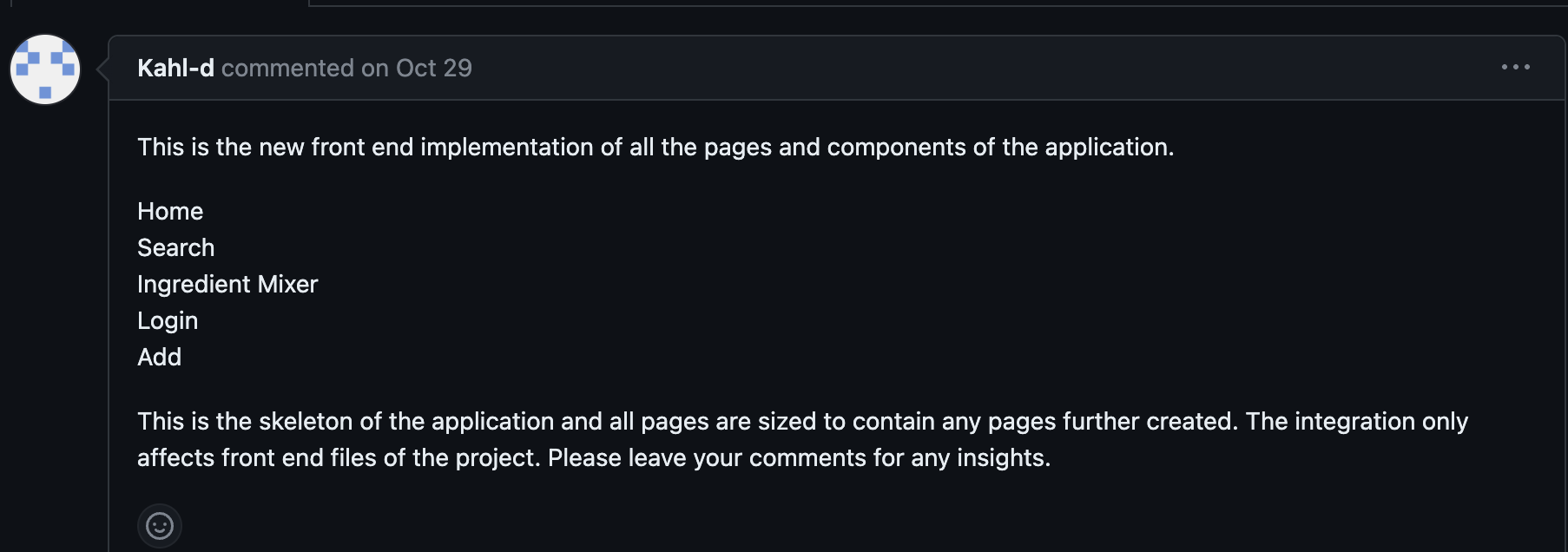
**Enforcement**

Tool: ESLint with Google Style Plugin

Npm provides easy installation and personalized setup for ESLint

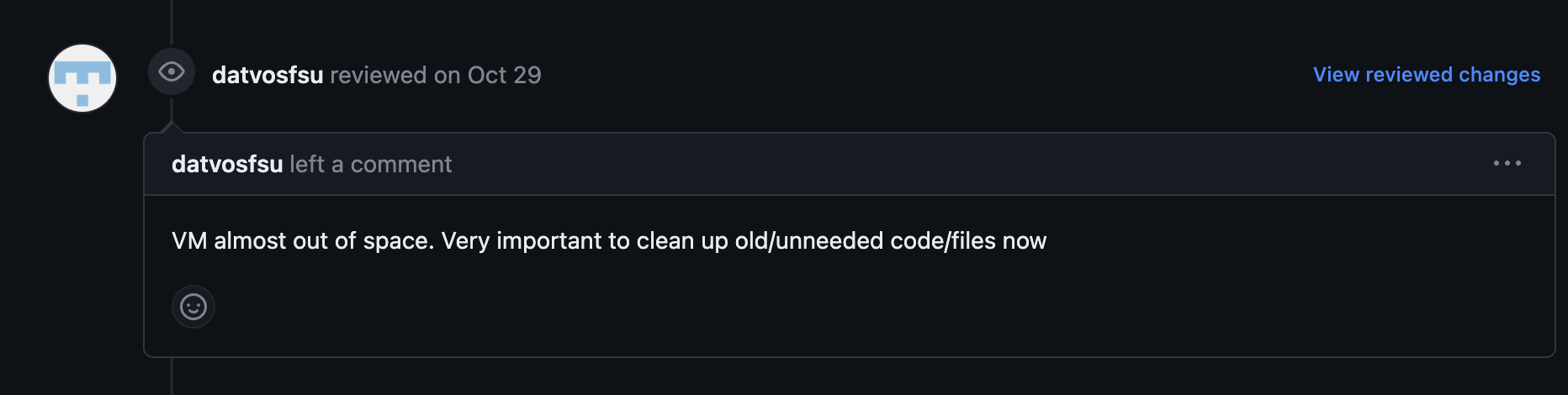


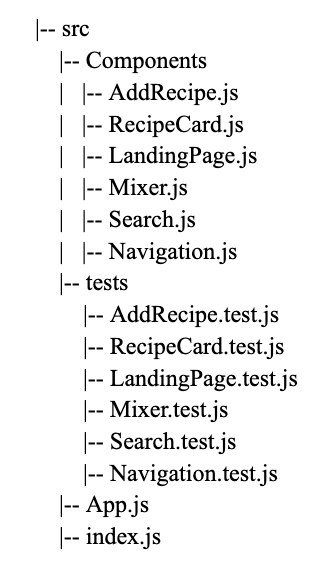


Comments & Documentation (Github Comments)

For best practices, our team utilizes the comments feature on Github to communicate any new updates/changes/recommendations regarding 

UMAMe. It allows the backend and frontend to communicate in a centralized location. It also keeps us on track in terms of progress since the comments and repo are located adjacent to one another.





Naming Conventions

By using CamelCase, a style of code that creates meaningful and descriptive names for our variables, classes, and functions, we are able to keep a more clean and concise project. By making sure our code is as concise as can be, it allows for a better flow through our code

By following a proper hierarchy, we can be stay organized & create an understandable flow

**Demonstration of Coding Style in Source Files**

| **Main Component (Main.js)** | Purpose: Renders the main page of the application  Proper use of naming conventions, function structure, and modular design as per the Google Style Guide |
| --- | --- |
| **Search Component (Search,js)** | Purpose: Handles the search functionality  Code readability, use of comments, and error handling following the style guide |
| **Navigation Component (Navigation.js)** | Purpose: Provides navigation between different parts of the app  Consistent formatting, clean code structure, and effective use of React hooks. |
| **Recipe Card Component (RecipeCard.js)** | Purpose: Displays individuals recipes in card format  Adherence to object and array formatting guidelines, as well as modular and reusable code structure |
| **Cooking Mode Component (CookingMode.js)** | Purpose: Offers a detailed view for cooking instructions  Demonstrates the style guide’s best practices in organizing long JSX and JavaScript code effectively |



jsDoc (Main.js) jsDoc (RecipCard.js)

(Proper naming conventions & error handling) (Proper commenting )

|-- src

|-- Components

| |-- AddRecipe.js

| |-- RecipeCard.js

| |-- LandingPage.js

| |-- Mixer.js

| |-- Search.js  
 | |-- Navigation.js

|-- tests

|-- AddRecipe.test.js

|-- RecipeCard.test.js

|-- LandingPage.test.js  
 |-- Mixer.test.js

|-- Search.test.js  
 |-- Navigation.test.js  
 |-- App.js  
 |-- index.js