

SW Engineering CSC648/848 Summer 2024

## ***ScholarEats***

### **Team 4:**

Angelo Arriaga (Team Lead - aarriaga1@sfsu.edu)

Donovan Taylor (Front-End Lead)

Hancun Guo (Front-End)

Tina Chou (Front-End)

Edward McDonald (Back-End Lead)

Karl Carsola (Back-End)

Sai Bavisetti (Database)

Maeve Fitzpatrick (Docs Editor)

Sabrina Diaz-Erazo (GitHub Master)

#### **Milestone 4**

08/01/24

History (Revisions)	
08/01/2024	Milestone 4 Version 2 Submission
07/30/2024	Milestone 4 Version 1 Submission
07/23/2024	Milestone 3 Part 2 feedback added
07/23/2024	Milestone 3 Version 1 Submission
07/22/2024	Milestone 2 Version 2 Submission
07/09/2024	Milestone 2 Version 1 Submission
06/26/2024	Milestone 1 Version 1 Re-Submission
06/20/2024	Milestone 1 Version 1 Submission

## **1. Product Summary**

*Name of the Product:* ScholarEats

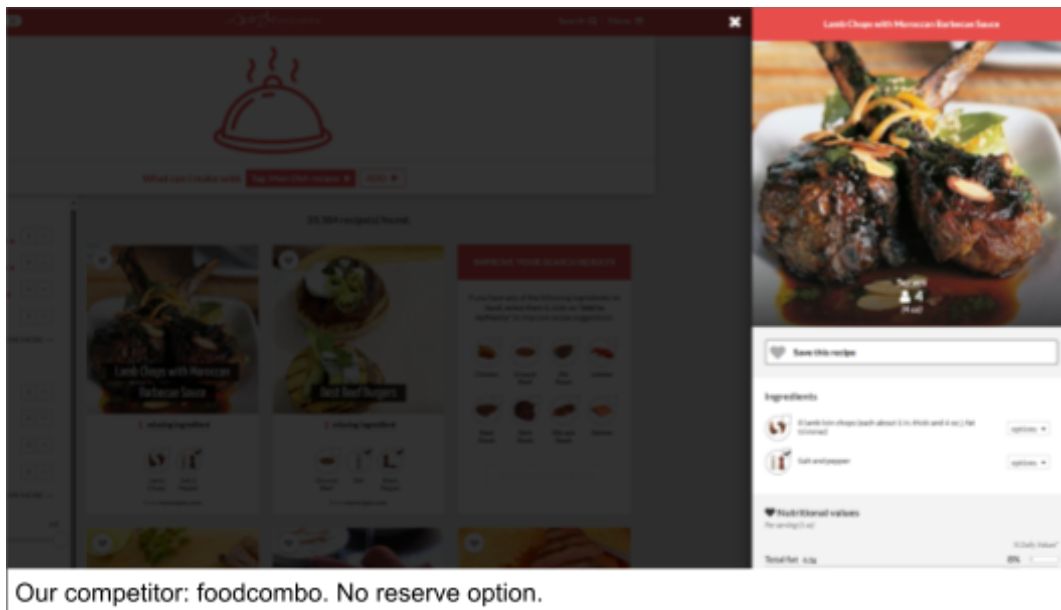
*ALL major committed functions:*

1. ADMINISTRATORS SHALL BE ABLE TO AUTHENTICATE THE USERS.
2. ADMINISTRATORS SHALL BE ABLE TO BLACKLIST/ WHITELIST A USER
3. ADMINISTRATORS SHALL BE ABLE TO EDIT THE INGREDIENT LIST
4. ADMINISTRATORS SHALL BE ABLE TO REMOVE USER ACCOUNTS
5. ADMINISTRATORS SHALL BE ABLE TO FOLLOW THE EXPIRATION DATE CLOSELY AND TAKE THE NECESSARY ACTIONS.
6. ADMINISTRATORS SHALL BE ABLE TO MAKE UNIVERSITY-WIDE ANNOUNCEMENTS
7. SYSTEM SHALL AUTOMATICALLY ENROLL USERS IN CORRESPONDING UNIVERSITY FOOD PROGRAMS
8. SYSTEM SHALL BE ABLE TO GENERATE RECOMMENDED RECIPES
9. SYSTEM SHALL TELL ADMINISTRATORS WHEN AN INGREDIENT HAS RUN OUT OF STOCK
10. SYSTEM SHALL TELL ADMINISTRATORS WHEN FOOD HAS SPOILED
11. USERS SHALL BE ABLE TO CHANGE THEIR PASSWORD
12. USERS SHALL BE ABLE TO EDIT THEIR USERNAME
13. USERS SHALL BE ABLE TO FILTER RECIPES BY COOKING AIDS REQUIRED
14. USERS SHALL BE ABLE TO FILTER RECIPES BY DIETARY RESTRICTIONS
15. USERS SHALL BE ABLE TO FILTER RECIPES BY DIFFICULTY
16. USERS SHALL BE ABLE TO FILTER RECIPES BY INGREDIENTS
17. USERS SHALL BE ABLE TO MAKE ACCOUNTS WITH VALID UNIVERSITY EMAIL.
18. USERS SHALL BE ABLE TO SET ALLERGIES
19. USERS SHALL BE ABLE TO SET DIETARY RESTRICTIONS
20. USERS SHALL BE ABLE TO SORT RECIPES BY CALORIES
21. USERS SHALL BE ABLE TO SORT RECIPES BY FAT
22. USERS SHALL BE ABLE TO SORT RECIPES BY FIBER

23. USERS SHALL BE ABLE TO SORT RECIPES BY PROTEIN
24. USERS SHALL BE ABLE TO VIEW AVAILABLE INGREDIENTS
25. USERS SHALL BE ABLE TO VIEW RECIPES
26. USERS SHALL BE ABLE TO EDIT PROFILE BIO
27. USERS SHALL BE ABLE TO ADD THEIR PREFERRED PRONOUNNS
28. USERS SHALL BE ABLE TO SHARE RECIPES
29. USERS SHALL BE ABLE TO SWITCH BETWEEN LIGHT AND DARK MODE
30. USERS WILL RECEIVE EMAILS WITH PUSH NOTIFICATIONS

*What is unique in our product:*

The features that make our product unique is that we offer automatic enrollment for eligible academic emails into their school's food pantry, and that we enable our users the option to reserve the ingredients they wish to pick up for the week (see the following screenshots for comparison to other companies).



SearchFilter

## Sarah's Homemade Applesauce



Prep Time: 10

Cook Time: 15


Servings: 4

RESERVE

Ingredients

Instructions

1. Combine apples, water, sugar, and cinnamon in a saucepan; cover and cook over medium heat until apples are soft, about 15 to 20 minutes.
2. Allow apple mixture to cool, then mash with a fork or potato masher until it is the consistency you like.
3. Photo by coconinana.
4. cookin nana



Contact Info

San Francisco

State University



(123) 456-7890

Contact Us

About

Privacy Policy

Terms of Service



Us: reserve option.

URL to Final Version of our Product: <http://3.148.145.110:3000/>

## **2. Usability Test Plan**

### *Test Objectives:*

#### **Function 1: Searching for Recipes**

The primary objective of this usability test is to ensure that users can quickly and accurately find the recipes they need using the search bar and filters. This involves evaluating the entire search process, from the initial input of search queries to the final selection of recipes. We aim to assess both the speed and accuracy with which users can navigate the search functionality and refine their results using the available filters.

#### **Function 2: Updating Allergy Information**

The primary objective of this usability test is to ensure that users can conveniently and efficiently update their allergy information on the account management page. We aim to evaluate the ease of use and effectiveness of the allergy information update process.

#### **Function 3: Change Dietary Restrictions**

The primary objective of this usability test is to ensure that users can easily and efficiently change their dietary restrictions on the account management page. This involves evaluating the entire update process, from locating the dietary restrictions section to successfully saving the updated information. We aim to assess both the ease of use and the effectiveness of the interface, ensuring that users can complete the task with minimal effort and without encountering any issues. Additionally, we seek to gather user feedback on their overall experience, including the intuitiveness of the design and any suggestions for improvement. This will help us identify potential areas for enhancement and ensure a smooth and satisfying user experience.

#### **Function 4: Managing User Profile**

The primary objective of this usability test is to ensure that users can conveniently and efficiently update and manage their profile information. This includes tasks such as editing personal details, changing passwords, updating contact information, and adjusting account

settings. We aim to assess both the ease of use and the effectiveness of the profile management interface, ensuring that users can complete these tasks with minimal effort and without encountering any issues. Additionally, we seek to gather user feedback on their overall experience, including the intuitiveness of the design and any suggestions for improvement. This will help us identify potential areas for enhancement and ensure a smooth and satisfying user experience.

#### Function 5: Reserving Available Ingredient

The primary objective of this usability test is to ensure that users can successfully and efficiently reserve an available item, such as a cooking ingredient, through our platform. We aim to evaluate the entire reservation process, from the initial search for the item to the final confirmation of the reservation. This involves assessing both the ease of use and the effectiveness of the reservation system.

#### *Test Description*

##### Function 1: Searching for Recipes

- System Setup:
  - We will be testing the latest version of our recipe search platform. The system will be connected to the internet and optimized to handle multiple user requests simultaneously, ensuring there is no lag during the test.
- Starting Point:
  - The test will commence with the user starting from the homepage of the website. This simulates a typical user journey, where they begin their search for recipes from the main landing page.
- Intended Users:
  - The primary users for this test are home cooks and cooking enthusiasts looking for specific recipes. This includes both new users exploring the platform for the first time and returning users who are familiar with the site but are looking for new recipes.

- URL of the System to be Tested:
  - <http://3.148.145.110:3000/recipes>
- What is to be Measured:
  - User Interaction Time: Measure the time taken from entering the homepage to finding the desired recipe. This will assess the efficiency of the search interface.
  - Accuracy of Search Results: Evaluate how accurately the search bar and filters help users find the recipes they are looking for. This involves examining the relevance of search results to the user's query.
  - Number of Steps to Complete a Search: Count the number of steps users take to find and select a recipe. This will help measure the complexity and usability of the search process.
  - User Satisfaction: Through post-interaction surveys or interviews, assess user satisfaction with the search and filtering process. Gather feedback on the ease of use, intuitiveness of the interface, and overall experience.
  - Error Rates: Track the frequency and types of errors encountered during the search process. This includes system errors, irrelevant search results, or user mistakes due to interface design issues.

## Function 2: Updating Allergy Information

- System Setup:
  - We will be testing the latest version of our account management platform using the most recent version of a popular web browser, connected to the internet. The system will be optimized to handle multiple user requests simultaneously to ensure smooth performance during the test.
- Starting Point:
  - The test will commence with the user starting from the account management page of the website. This simulates a typical user journey, where they begin their update process directly from the main account management section.
- Intended Users:

- The primary users for this test are individuals who need to update their allergy information. This includes both new users exploring the platform for the first time and returning users who are familiar with the site.
- URL of the System to be Tested
  - <http://3.148.145.110:3000/accountManagement>
- What is to be Measured:
  - User Interaction Time: Measure the time taken from entering the account management page to completing the allergy information update. This will assess the efficiency of the update interface.
  - Accuracy of Information Update: Evaluate how accurately users can update their allergy information. This involves examining the ease of input and clarity of the interface.
  - Number of Steps to Complete an Update: Count the number of steps users take to find and update their allergy information. This will help measure the complexity and usability of the update process.
  - User Satisfaction: Through post-interaction surveys or interviews, assess user satisfaction with the allergy information update process. Gather feedback on the ease of use, intuitiveness of the interface, and overall experience.
  - Error Rates: Track the frequency and types of errors encountered during the update process. This includes system errors, incorrect information updates, or user mistakes due to interface design issues.

### Function 3: Change Dietary Restrictions

- System Setup:
  - We will be testing the latest version of our platform on an updated web browser connected to the internet. The system will be optimized to handle multiple user requests simultaneously, ensuring smooth performance during the test.
- Starting Point:



- The test will commence with the user starting from the account management page of the website. This simulates a typical user journey, where they begin the process of changing dietary restrictions directly from the main account management section.
- Intended Users:
  - The primary users for this test are individuals who need to update their dietary restrictions. This includes both new users exploring the platform for the first time and returning users who are familiar with the site.
- URL of the System to be Tested:
  - <http://3.148.145.110:3000/accountManagement>
- What is to be Measured:
  - User Interaction Time: Measure the time taken from entering the account management page to completing the dietary restriction update. This will assess the efficiency of the update interface.
  - Accuracy of Information Update: Evaluate how accurately users can update their dietary restrictions. This involves examining the ease of input and clarity of the interface.
  - Number of Steps to Complete an Update: Count the number of steps users take to find and update their dietary restrictions. This will help measure the complexity and usability of the update process.
  - User Satisfaction: Through post-interaction surveys or interviews, assess user satisfaction with the dietary restriction update process. Gather feedback on the ease of use, intuitiveness of the interface, and overall experience.
  - Error Rates: Track the frequency and types of errors encountered during the update process. This includes system errors, incorrect information updates, or user mistakes due to interface design issues.

#### Function 4: Managing User Profile

- System Setup:

- We will be testing the latest version of our user profile management platform using the most recent version of a popular web browser, connected to the internet. The system will be optimized to handle multiple user requests simultaneously to ensure smooth performance during the test.
- Starting Point:
  - The test will commence with the user starting from the account management page of the website. This simulates a typical user journey, where they begin the process of managing their profile directly from the main account management section.
- Intended Users:
  - The primary users for this test are individuals who need to update and manage their profile information and password. This includes both new users exploring the platform for the first time and returning users who are familiar with the site.
- URL of the System to be Tested:
  - <http://3.148.145.110:3000/accountManagement>
- What is to be Measured:
  - User Interaction Time: Measure the time taken from entering the account management page to completing various profile management tasks. This will assess the efficiency of the interface.
  - Accuracy of Information Update: Evaluate how accurately users can update their profile information. This involves examining the ease of input and clarity of the interface.
  - Number of Steps to Complete an Update: Count the number of steps users take to find and update their profile information. This will help measure the complexity and usability of the management process.
  - User Satisfaction: Through post-interaction surveys or interviews, assess user satisfaction with the profile management process. Gather feedback on the ease of use, intuitiveness of the interface, and overall experience.

- Error Rates: Track the frequency and types of errors encountered during the management process. This includes system errors, incorrect information updates, or user mistakes due to interface design issues.

#### Function 5: Reserving Available Ingredient

- System Setup:
  - We will be testing the latest version of our reservation platform using the most recent version of a popular web browser, connected to the internet. The system will be optimized to handle multiple user requests simultaneously to ensure smooth performance during the test.
- Starting Point:
  - The test will commence with the user starting from the ingredients page of the website. This simulates a typical user journey, where they begin their reservation process directly from the main ingredients listing.
- Intended Users:
  - The primary users for this test are individuals who want to reserve cooking ingredients. This includes both new users exploring the platform for the first time and returning users who are familiar with the site and are looking to reserve specific ingredients.
- URL of the System to be Tested:
  - <http://3.148.145.110:3000/ingredients>
- What is to be Measured:
  - User Interaction Time: Measure the time taken from entering the ingredients page to completing the reservation. This will assess the efficiency of the reservation interface.
  - Accuracy of Search Results: Evaluate how accurately the search bar and filters help users find the ingredients they are looking to reserve. This involves examining the relevance of search results to the user's query.

- Number of Steps to Complete a Reservation: Count the number of steps users take to find and reserve an ingredient. This will help measure the complexity and usability of the reservation process.
- User Satisfaction: Through post-interaction surveys or interviews, assess user satisfaction with the reservation process. Gather feedback on the ease of use, intuitiveness of the interface, and overall experience.
- Error Rates: Track the frequency and types of errors encountered during the reservation process. This includes system errors, irrelevant search results, or user mistakes due to interface design issues.

*Usability Test Table (Effectiveness)*

Test/Use Case	Percentage Completed	Errors	Comments	Efficiency (User Time/Average Time)
Search for Recipes	100%	None	<ul style="list-style-type: none"> <li>- Some users suggested adding more advanced search options</li> <li>- Users appreciated ability to narrow down search results effectively</li> <li>- Navigation from search results to recipe page from smooth and straightforward</li> </ul>	20 sec/ 20 sec
Update Allergy Information	100%	None	<ul style="list-style-type: none"> <li>- Users easily located the allergy information section.</li> <li>- The update process was straightforward and clear.</li> </ul>	20 sec / 20 sec

			<ul style="list-style-type: none"> <li>- Users experienced a smooth saving process with clear confirmation.</li> </ul>	
Change Dietary Restrictions	100%	None	<ul style="list-style-type: none"> <li>- Users easily located the dietary restrictions section.</li> <li>- The update process was straightforward and clear.</li> <li>- Users experienced a smooth saving process with clear confirmation.</li> </ul>	20 sec / 20 sec
Manage User Profile	100%	None	<ul style="list-style-type: none"> <li>- Users easily located the profile information section.</li> <li>- The process was straightforward and clear.</li> <li>- Users experienced a smooth and clear password change process, in addition to verify their new password again.</li> </ul>	55 sec / 45 sec
Reserve Available Ingredient	100%	None	<ul style="list-style-type: none"> <li>- The process of adding items to the reservation list was straightforward.</li> </ul>	25 sec / 25 sec

*Usability Test Table (Efficiency)*

Test/Use Case	Average Time to Complete	Av. Time of User /Av. Time of All Users	Effort	Content/Design
---------------	--------------------------	---	--------	----------------

Search for Recipes	20 sec	20 sec/ 20 sec	<ul style="list-style-type: none"> <li>- Exactly as to be expected</li> </ul>	<ul style="list-style-type: none"> <li>- Uniform - without confusion</li> </ul>
Update Allergy Information	20 sec	20 sec / 20 sec	<ul style="list-style-type: none"> <li>- Easily located section and smooth saving process</li> </ul>	<ul style="list-style-type: none"> <li>- Straightforward and clear single screen</li> </ul>
Change Dietary Restrictions	20 sec	20 sec / 20 sec	<ul style="list-style-type: none"> <li>- Found section without confusion</li> <li>- Saved dietary info without delays</li> </ul>	<ul style="list-style-type: none"> <li>- Single page to complete change</li> </ul>
Manage User Profile	45 sec	45 sec / 55 sec	<ul style="list-style-type: none"> <li>- Easy to find account button when logged in</li> <li>- Easy to change and divide to submit each part of profile</li> </ul>	<ul style="list-style-type: none"> <li>- Looks uniform</li> </ul>

Reserve Available Ingredient	25 sec	25 sec / 25 sec	- Works as you would expect	- Clear, uniform screen
------------------------------------	--------	-----------------	-----------------------------------	-------------------------------

*User Satisfaction*

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Search for Recipes					
The search function was easy to use					X
I am satisfied with the overall experience of using the search and filters					X
The design of the search and filtering features is user-friendly and intuitive					X
Update Allergy Information					
The allergy information update process was easy to use					X
I am satisfied with the overall experience of updating my allergy information					X
The design of the allergy information update features is user-friendly and intuitive				X	

Change Dietary Restrictions					
The dietary restriction update process was easy to use					X
I am satisfied with the overall experience of updating my dietary restrictions				X	
The design of the dietary restriction update features is user-friendly and intuitive					X
Manage User Profile					
The profile management process was easy to use					X
I am satisfied with the overall experience of managing my profile					X
The design of the profile management features is user-friendly and intuitive			X		
Reserve Available Ingredient					
The reservation process was easy to use			X		
I am satisfied with the overall experience of reserving items			X		



The design of the reservation features is user-friendly and intuitive			X		
---	--	--	---	--	--

### **3. QA Test Plan**

#### **1. Privacy and Security**

- Test objectives: Test if user's email is valid.
- HW setup: Desktop computer.
- SW setup: Windows system, Chrome browser, <http://3.148.145.110:3000/>
- Feature to be tested: Email validation.
- QA Test Plan:

Test #	Test Description	Input	Expected Output	Pass/Fail	Efficiency
1	Test if an apparently incorrect email can register an account.	12345	The page loads a window to mention user that the email entry is invalid.	Pass	<1s
2	Test if an email not from university domain can register an account.	abcde@gmail.com	The page loads a window to mention user that the email entry is invalid.	Pass	<1s
3	Test if an email from university domain and is truly exist can register an account	hguo4@sfsu.edu	The page tells user that is successfully registered.	Pass	2s

#### **2. Response time**

- Test objectives: Test if the page always respond in 7 seconds.
- HW setup: Desktop computer.
- SW setup: Windows system, Chrome browser, <http://3.148.145.110:3000/>

- Feature to be tested: response time.

Test #	Test Description	Input	Expected Output	Pass/Fail	Efficiency
1	Test the loading of recipe page.	Click on recipe in navbar.	The page loads with recipes and the response time is not exceeding 7s.	Pass	4s
2	Test the loading of single recipe page.	Go into recipe page and click on "Apple Pie by Grandma Ople".	The page loads photo and instructions of "Apple Pie by Grandma Ople" and the response time is not exceeding 7s.	Pass	1s
3	Test the loading of ingredient page.	Click ingredient in navbar.	The page loads with ingredients and the response time is not exceeding 7s.	Pass	3s

### 3. Expected Load

- Test objectives: Test if the system can support 200,000 unique students.
- HW setup: Desktop computer.

- SW setup: Windows system, MySQL database, Chrome browser,  
<http://3.148.145.110:3000/>
- Feature to be tested: load capacity.

Test#	Test Description	Input	Expected Output	Pass/Fail	Efficiency
1	Test the capacity of database.	Automatically make up 200,000 users in database.	The database works fine with this amount of users.	Pass	4s
2	Test the loading of system with 200,000 users in database.	Stay the 200,000 users in the database, login with username "test1", password "12345"	User successfully login.	Pass	1s
3	Test admin tool with 200,000 users in database.	Stay the 200,000 users in the database, admin login with username "admin1", password "12345". Add 1 user with username "CapacityTest" and password	User added and is able to login.	Pass	3s

		"12345"			
--	--	---------	--	--	--

#### 4. Hardware:

- Test objectives: Test if the system's display is supported on different devices.
- HW setup: Desktop computer, mobile phone, tablet.
- SW setup: Windows system, Chrome browser for desktop,
  - IOS system, safari browser for mobile phone,
  - IOS system, safari browser for tablet.
  - <http://3.148.145.110:3000/>
- Feature to be tested: Display dimensions support.

Test #	Test Description	Input	Expected Output	Pass/Fail	Efficiency
1	Test the display on desktop is user friendly.	Open chrome browser and enter" <a href="http://3.148.145.110:3000/">http://3.148.145.110:3000/</a> "	Landing page loads and every button is easily clickable.	Pass	2s
2	Test the display on mobile phone is user friendly.	Open safari browser on mobile phone and enter" <a href="http://3.148.145.110:3000/">http://3.148.145.110:3000/</a> "	Landing page loads and every button is easily clickable.	Pass	2s
3	Test the display on tablet is user friendly.	Open safari browser on tablet and enter"	Landing page loads and every button is easily clickable.	Pass	3s

		http://3.148.145 .110:3000/”			
--	--	---------------------------------	--	--	--

#### 5. Software support

- Test objectives: Test if the system is supported on different software.
- HW setup: Desktop computer, apple notebook.
- SW setup: Windows system, Chrome browser, Firefox browser.
  - OS system, Safari browser.
  - http://3.148.145.110:3000/
- Feature to be tested: Browser support.

Test#	Test Description	Input	Expected Output	Pass/Fail	Efficiency
1	Test the display on Chrome browser is user friendly and the system is functional working.	Open chrome browser and enter” <a href="http://3.148.145.110:3000/">http://3.148.145.110:3000/</a> ,” click each button on navbar.	Every page loads properly, every visible item is user friendly.	Pass	2s
2	Test the display on Firefox browser is user friendly and the system is functional working.	Open Firefox browser and enter” <a href="http://3.148.145.110:3000/">http://3.148.145.110:3000/</a> ,” click each	Every page loads properly, every visible item is user friendly.	Pass	2s

		button on navbar.			
3	Test the display on Safari browser is user friendly and the system is functional working.	Open safari browser on tablet and enter” http://3.148.145.110:3000/”, click each button on navbar.	Every page loads properly, every visible item is user friendly.	Pass	2s

## 4. Code Review

### **a) Our Coding Conventions:**

#### *Naming of Variables:*

- Ideally variables will be named in a single word that can capture their identity where possible.
- However in cases where a variable must be named with multiple words in order to properly convey its purpose, variables are to be named in the “camelCase” manner.
- For example, the case in which the sum or count of a “thing” is needed to be stored you would write it as sumThings or countThings (note we omit “Of” in this)
- GLOBAL VARIABLES ARE ALLCAPS

#### *Method Names:*

- Similar to above, we will name methods in “camelCase”.
- Note that for many things specifically in the “front-end” domain they exist using arrow-functions with things like app.use(...) and call existing things like res.render(...).
- This will largely apply to helper functions like the ones defined in the “back-end”, if any.

#### *Script Names:*

- Related to the above, a lot of things, like EventListeners, will be implemented by simply creating a series of statements in a .js file and including the script within the .hbs through a res.render() call.
- In this case the script should follow that naming convention.

#### *Database Specifics:*

- We should probably use db.execute(...) over db.query based on security concerns as it helps prevent SQL injections as execute does the database operation on the back-end whereas query does the operation on the front-end.



- Note that usage is similar however if we do use this then we need to use “await” and “promises”. I can give examples if needed.

Names will be snake\_case for column names (i.e. snake\_case)

#### *Naming Convention of Branches:*

- For branches we utilize names consisting of only lowercase letters and underscores “\_”.
- Any branches currently not following this convention will be changed to do so, keep in mind that this will add a step to your next push as you will have to update what remote repository branch you are pushing to
- Guidance can be found at [this link here](#).

#### *Design Conventions:*

##### *Routing Structures:*

- We should have ALL routes within the main routes folder which is /application/routes within the repository
- We can however separate “front-end” and “back-end” for each route like “userRoutes\_FE” and “userRoutes\_BE”

##### *Testing Features:*

- Do NOT make separate app.js’s and commit them to branches that you are trying to merge into main/master.
- Master branch should be production ready code, free of testing aids. If you feel the need to include something to demonstrate functionality and it doesn’t exist, implement a view and use routing to make it possible to access in the same application as a separate page.
- Additionally, doing so will better demonstrate how the code would actually be implemented in existing pages via the existing architecture we developed, (i.e. dynamically feeding in scripts via res.render())

- Again, do not make separate applications with their own launching conditions or packages / package.json / package-lock.json.
- Failure to do so will result in your PR being denied until the branch is in compliance

#### *Handlebars and CSS:*

- We should be moving to utilize the “grid-area” form of setting up the layouts. This will enable us to better support displays of all dimensions
- As always, we should have a singular “layout” that maintains consistency throughout all pages.
- The unique and different pages will be done via “views” which will be inserted within the body of the html.
- Note that each page also has a “title” that we alter via our res.render() calls by passing in a value. For the sake of consistency this field should be a “header” tag within the “body” tag but it should be defined within the layout and not within each page’s independent views. This is something that we are currently not complying with and hurts the consistency of our pages
- Cascading Style Sheet (.css) will be passed in via the res.render() as we have decided upon before, this will allow easier implementation of “themes” or additional/conditional formatting. (i.e. dark mode)

## b) Screenshots of Code Review Process

sfsu-joseo / csc648-848-05-sw-engineering-su24-T4

<> Code

Issues

Pull requests

1

Actions

Projects

Wiki

Security

updated SQL to match updated DB terminology

#46

Merged

sderazo merged 3 commits into master from SQL\_Update 2 days ago

Conversation 2

Commits 3

Checks 0

Files changed 21

+80 -1,026

EdwardMcD commented 3 days ago

IN TABLE expired\_products

- renamed Name to name

IN TABLE notifications

- added column university

IN TABLE recipes

- renamed Unnamed: 0 to recipe\_id
- renamed dietary restrictions to dietary\_restrictions
- renamed cooking tip to cooking\_tip
- deleted MyUnknownColumn

IN TABLE store

- renamed Name to name

TABLE Users

- changed Users to users

Reviewers

sderazo

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

Notifications

Customize

Unsubscribe

You're receiving notifications because you modified the open/close state.

2 participants

Lock conversation

updated SQL to match updated DB terminology

04a53e3

sderazo requested changes 2 days ago

View reviewed changes

sderazo left a comment

- changes made in each file are necessary for variables to have the same naming convention
- in-line comments are good and descriptive
- in recipeRoutes\_BE.js: is the commented out block of code still needed?
- please include at least a minimal code header in each file

EdwardMcD added 2 commits 2 days ago

Added headers to files the backend team has worked on

f6f3dc9

Deleted unnecessary folder "Backend"

a0e1291

sderazo approved these changes 2 days ago

View reviewed changes

sderazo left a comment

- the files look good (each one has a code header and the commented out code was deleted)

sderazo merged commit f2d9aa0 into master

2 days ago

Revert

## Backend development #47

Edit <> Code

**Merged** sderazo merged 9 commits into master from backend\_development 17 minutes ago

Conversation 2 Commits 9 Checks 0 Files changed 12 +1,045 -498



EdwardMcD commented yesterday

- Added support for multiple universities
- You only see the recipes available for the ingredients belonging to your own university, and you only see ingredients available for your own university
- You can now click "reserve" when viewing a recipe, which will notify the admins associated with your university that you have reserved a recipe
- You can now only see notifications that pertain to you



Reviewers



sderazo



Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

Notifications

Customize

Unsubscribe

You're receiving notifications because you modified the open/close state.

2 participants



Lock conversation



EdwardMcD commented 9 hours ago

Author

PR Update:

- If you are logged out, the notification page does not crash
- Notifications now show the correct recipe
- If your university does not have an admin, you cannot reserve that recipe.
- After reserving a recipe, it now directs you back to that recipes page
- Image now properly shows up on the individual ingredients page (NOTE: this will need to change (and will most likely break) when sai finishes updating the database, and I do not have the ability to edit the table(s) that this is effected by)



sderazo approved these changes 18 minutes ago

[View reviewed changes](#)

sderazo left a comment

- in-line comments are detailed
- formatting is consistent and neat
- name of variables follow agreed-upon convention

files that need code header:

- authentication.js
- blacklist.js
- users.js
- individual\_recipes\_view.hbs

Edward's PR update (in conversation tab) addresses the problems I ran into when I first reviewed backend\_development's changes on the instance. Note: I listed out the problems I ran into in Discord instead of having them inside of this PR review. This is bad practice, so next time when I review a PR, I will comment on the instance in here.

issues from app on instance that will be addressed in a separate PR:

- admin tools shows up as unfinished button
- buttons in account management page need to be smaller (too long) (frontend will handle this)



sderazo merged commit 46fbc14 into master 17 minutes ago

[Revert](#)











**Pull request successfully merged and closed**


You're all set—the backend\_developm... branch can be safely deleted.

[Delete branch](#)



### c) External Code Review for Different Team



 Sabrina Diaz-Erazo



  Reply  Reply all  Forward   

To:  Michelle Nguyen

Sun 7/28/2024 3:56 PM

 ae0421c6-97c4-42ff-aa39-8c... 57 KB 


 recipeRoutes\_BE.txt 15 KB 








2 attachments (72 KB)  Save all to OneDrive - San Francisco State University  Download all


Hello Michelle,

My name is Sabrina (from Team 4) and I will be conducting your team's external code review. I have attached the file that our team would like for your team to review to this email (I had to change the extension from .js to .txt since outlook wouldn't let me send it). I have also included a document outlining the coding and design conventions we followed.



Best,  
Sabrina Diaz-Erazo  
GitHub Master  
Team 4



 Michelle Nguyen



  Reply  Reply all  Forward   

To:  Sabrina Diaz-Erazo

Mon 7/29/2024 8:50 PM

 Progress.txt 15 KB 

 UsersController.java 2 KB 

2 attachments (17 KB)  Save all to OneDrive - San Francisco State University  Download all

Hello Sabrina,

Hi Team 4,

I will be conducting your team's external code review.

Attached, you will find the file that our team would like your team to review. Please review our code for the back and front end. The files should be called: "progress.js" and "UserController.java".

We appreciate your time and effort in reviewing our code. Your feedback is valuable to us, and we look forward to your constructive comments.

Best,  
Michelle  
Team03

Sabrina Diaz-Erazo  
To: @ Michelle Nguyen



Mon 7/29/2024 9:02 PM

Hello Michelle,

According to the Milestone 4 Document, it states the following: "Team's code review can be performed to only one file or script from your project." Please let me know which single file you and your team would like for me to review. Thank you for taking the time to review our code. We look forward to see how we can improve it.

Best,  
Sabrina Diaz-Erazo  
Team 4  
GitHub Master

...

Michelle Nguyen  
To: @ Sabrina Diaz-Erazo



Mon 7/29/2024 9:47 PM

Please review any. I recommend the backend UserController.

...

Michelle Nguyen  
To: @ Sabrina Diaz-Erazo  
Cc: @ Uzair Hamed Mohammed



Mon 7/29/2024 11:20 PM

Hello Sabrina,

Our team (Uzair and I) reviewed your code that you gave us and here are some feedback we saw in your code. You all need a better layout/design of your application. For example we chose the MVC design pattern for the backend, and thanks to it, our files are very organized and easy to find. We are not sure how javascript projects work, but it looks like you all dumped everything in one file?

Additionally...

1. Avoid hardcoding credentials in the code.
2. Use environment variables or a secure vault to manage sensitive information
3. The error message in the res.status(400).json response should be more user-friendly and avoid exposing internal error codes directly to the user. Consider logging the detailed error internally and providing a generic message to the user
4. The comments are helpful, but ensure they are up-to-date and accurate. The comment about IS\_LOGGED\_IN is unclear without additional context.

I hope we were able to help you review your code. Let us know (Uzair and I) if you have any concerns or questions.

Best,  
Michelle & Uzair  
Team03

...

## 5. Self-Check on Best Practices for Security - ½ Page

We are currently protecting database login credentials, passwords, and SQL queries.

Database login credentials are not hard coded into each file, and are instead stored in the environment, separate from the code, and loaded through `dotenv`. The file is named `process.env`, and it is stored at the root of our application.

An example of loading variables from the .env file

```
const connection = mysql.createPool({
  host:      process.env.DB_HOST,
  user:      process.env.DB_USER,
  password:  process.env.DB_PASS,
  database:  process.env.DB_NAME
});
```

We are encrypting passwords using 'bcrypt', a node.js library to hash (and salt) passwords before putting them in the database. Currently, we are hashing the password with 10 distinct salts, like so:

```
const hash = await bcrypt.hash(password, 10);
```

For example, in practice, if a user creates an account with the password "password," it appears in the database like so:

username	password_hash
tester1234	\$2a\$10\$SCaFII/V.J.uRDHIhNV55eJ52Y0MFycF...

To protect SQL queries, we are using parameterized queries and await execute() rather than query(), even when we are just returning values. This is how we validate input data in the search bar. In the example below, we directly inject the search input into the query.

```
// Search input from the search bar
if (searchInput) {
  query += ` AND recipe_name LIKE ${searchInput}`;
```

This is bad practice, as it allows for SQL injections. Instead, we should push the search input to the array of query parameters, and execute them individually utilizing SQL's ? placeholder variable as shown below.

```
// Search input from the search bar
if (searchInput) {
    query += ' AND `recipe_name` LIKE ?';
    queryParams.push(`%${searchInput}%`);
}
```

Along with hashing and salting, we also verify that user information meets the criteria during the registration process. Usernames must be at least 5 characters long, passwords must be at least 8 characters long, and the two password fields must match during registration.



## 6. Self-Check: Adherence to Original Non-Functional Specs

### List of Non-Functional Requirements

- Privacy:
  - WE WILL NOT SELL YOUR DATA TO AI
    - **DONE**
  - Users must check and confirm their preferences regarding personal information upon account creation.
    - **DONE**
- Security:
  - Passwords MUST be encrypted and NOT PLAINTEXT for security reasons
    - **DONE**
  - Emails must be validated (be an email from the university's domain).
    - **DONE**
- Coding Standards:
  - We shall use the “grid-area” property in CSS for setting up the layouts in order to better support displays of all dimensions.
    - **DONE**
  - We shall include a code header in all JavaScript files.
    - **ON TRACK**
  - We shall include in-line comments in our code when it is necessary to do so.
    - **DONE**
  - We shall only have production ready code in the master branch.
    - **DONE**
- Look and Feel:
  - Each page shall be divided up by a header, body, and footer section. The header shall contain a navigation bar and a search bar.
    - **DONE**
  - Each page shall have the program’s logo visible in the header.

- **DONE**
  - Layout of the website shall have an ease of use feel.
  - **DONE**
  - The user shall be able to switch between light and dark themes.
  - **ON TRACK**
- Compatibility:
  - The system shall have support for other browsers such as Safari.
  - **DONE**
  - The system shall be able to run on Windows, macOS, and Linux operating systems.
  - **DONE**
- Fault Tolerance:
  - We shall use try...catch in our JavaScript code to test our code for errors when it is being executed.
  - **ON TRACK**
- Availability:
  - The system shall be up and running during days that the university is open as stated on the academic calendar.
  - **DONE**
  - Maintenance shall be performed outside of university and class hours.
  - **ON TRACK**
- Expected Load:
  - Support 200,000 unique students **DONE**
- Performance:
  - System shall respond visually within 7 seconds
  - **DONE**
- Storage:
  - File Size of Photos shall not exceed 10MiB

- **ISSUE:** Our team has decided that the users do not have the ability to upload their own photos on our website.
- Support:
  - File formats for photos shall include:
    - .png
    - .jpeg
    - .heic
    - .heif
    - **ISSUE:** Our team has decided that the users do not have the ability to upload their own photos on our website.
- Hardware:
  - Display dimensions supported(css):
    - Phone **ON TRACK**
    - Tablet **ON TRACK**
    - Monitor **DONE**
- Software:
  - We shall support:
    - Firefox **DONE**
    - Chromium **DONE**
- Scalability:
  - We shall be able to add more users from universities other than San Francisco State University.
    - **DONE**
- Database High Level Specs:
  - We shall be able to prevent sql injections by using db.execute over db.query.
    - **ON TRACK**
- Capability:
  - The database shall be able to have 'no limit' on how many users it takes.
    - **ON TRACK**

- Environmental:
  - The system shall be able to be deployed on a cloud platform such as AWS.
    - **DONE**
- Licensing:
  - The system shall use and display images that are free use and/or have a Creative Commons license.
    - **DONE**

**7. List of contributions to the document and contributions score for all the team members. Also email with feedback/complaints. If you do not have complaints, then at least provide some feedback about the dynamics of the team for this milestone.**

1. Donovan Taylor (Score 10/10)
  - a. Adjusted landing page to have a more distinct blurb describing the site
  - b. Fixed the admin tools routes
  - c. The true Index page"/" loads the Landing Page instead of the Contact Us Page
  - d. Debugged and completed implementation of "Dark Mode"
2. Hancun Gao (Score 7/10)
  - a. Registration .css repaired
  - b. "Reserve" Button on Recipe page added
  - c. Addressed and completed QA Test Plan in the M4 Document
  - d. Ensured we updated the statuses of all Non-Functional Requirements in Section 6 of M4 Document
3. Edward McDonald (Score 10/10)
  - a. Hooked up functionality to the Recipes page to update the database
  - b. Added the usage of notifications to notify user of available ingredients to pick up
  - c. Examined the repository to ensure coding practices were followed
  - d. Retool-ed queries and routes to support different universities to demonstrate the "per-university" functionality of the product
  - e. Made examples for how to show separate universities's data and ingredients within the site to her per-university functionality
  - f. Hooked up Sabrina's dark-mode implementation to the database
  - g. Documented how we validated user input in forms and in SQL queries and the steps we took to avoid malicious efforts
  - h. Went through repository to ensure we utilized db.execute over db.query
  - i. Documented how passwords were encrypted
  - j. Worked to better Auto-Complete so it only worked on what was available

4. Karl Carsola (Score 8/10)
  - a. Implemented “Forgot Password” Functionality (check)
  - b. Moved the project away from using the alert function as some browsers prevent its usage
  - c. Incorporated Captcha
  - d. Addressed QA Test Plan in the M4 Document
  - e. Ensured we updated the statuses of all Non-Functional Requirements in Section 6 of M4 Document
5. Sai Bavisetti (Score 6/10)
  - a. Deleted Erroneous Entries
  - b. Assisted in the development of the Admin Tools used to modify the university’s possessed food store.
6. Maeve Fitzpatrick (Score 7/10)
  - a. Reworked coloring of .css sitewide to maximize readability and usability
  - b. Reworked .css sitewide to dynamically change with screen width between standard, tablet, and mobile views
  - c. Reorganized format of entire M4 Document for maximum readability/comprehension
7. Sabrina Diaz-Erazo (Score 8/10)
  - a. Privacy Policy when registering is accessible
  - b. Assisted Edward in hooking up “Dark Mode” to the user-preferences table through explaining code
  - c. Conducted the Code-Review process with other teams and attached the screenshots to the M4 Document
  - d. Adjusted Privacy Policy to accommodate non-functional requirement stating we will not sell user-data for AI purposes
  - e. Contributed to and finalized Section 6 of M4 Document
8. Tina Chou (Score 7/10)

- a. Reworked .css to have navbar buttons be less rounded and have margins between options
- b. Made adjustments to have user set dietary restrictions on registration
- c. Contributed to the Usability Test Plan