Final Project for SW Engineering Class

CSC 648-848 Section 01 Summer 2023

Team 4:

ScholarEats

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)

URL: http://3.148.145.110:3000/

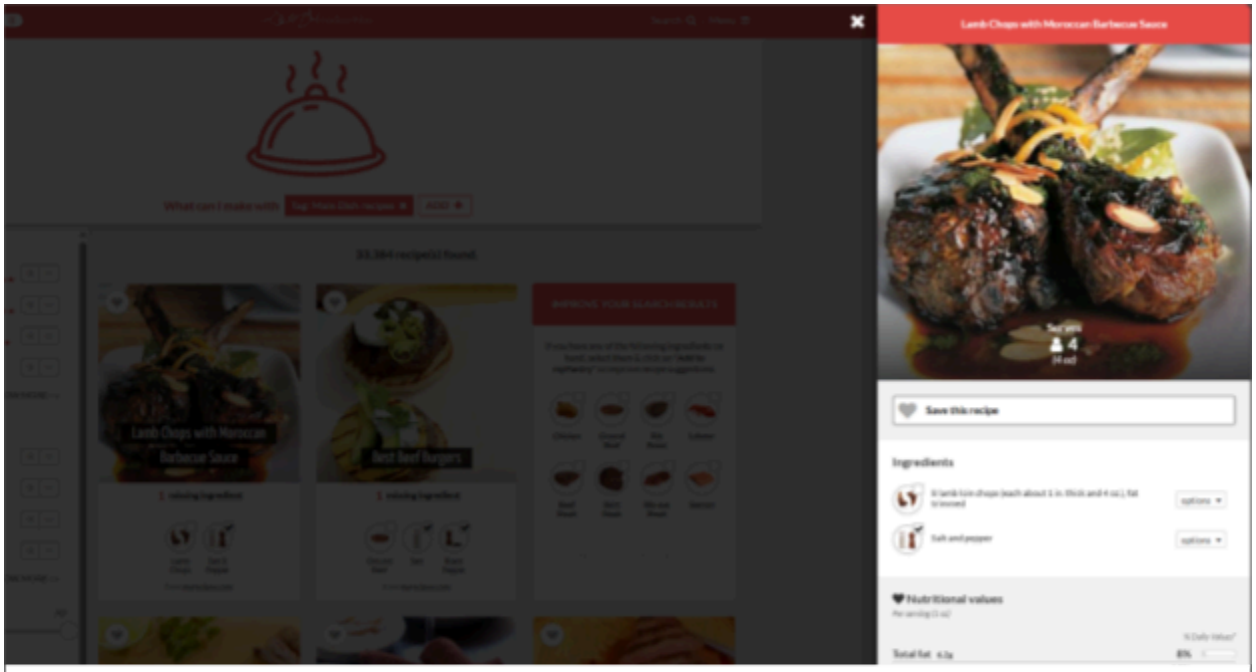
08/01/2024

History (Revisions)	
08/01/2024	Milestone 5 Version 1 Submission
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

2. 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 PRONOUNS
    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)



Our competitor: foodcombo. No reserve option.

Us: reserve option.

- *URL to Final Version of our Product*: http://3.148.145.110:3000/

3. Milestone documents - M1-M4 (only V2s)
M1 has no V2

Brainstorm Ideas for Project

1. *ScholarEats*
    - reverse cookbook, where student can find recipes to make based on ingredients being offered by university's grocery program
    - university makes admin account to enter what food is being offered
    - administration can auto-generate an email weekly that includes the ingredients and potential recipes
    - students can opt-in to getting the groceries that week if interested - allowing administration to have accurate number of prepared groceries (reduces waste)
    - useful for students specifically
    - using AWS, MySQL
2. *UpKeep*
    - is a social media site based around collecting and trading trading cards.
    - organized a little like reddit, where the emphasis is on following tags instead of other users. Like subreddits, people would post their cards to the pages, and other people following those pages will be able to see it, like it, comment, etc.
    - Cards can be posted "for sale" or "for trade" to link up with other players/collectors that are looking for that card and are willing to trade/sell. When a user posts cards to their page, they can also build a virtual collection.
    - This could be useful at conventions or swap meets, where traders could check each other's digital collection and see what they have up for sale, instead of flipping through those massive binders.
    - will implement ActivityPub (allows for connected communities - increases visibility)
    - using AWS, MySQL
3. *Crash*
    - AirBnB for Parking
    - users can rent out someone's parking space for a specific short-term time frame
    - could use Unfolding maps API (Java/Javascript)
    - using AWS, MySQL

SW Engineering CSC648-848-05 Summer 2024

ScholarEats

Milestone 1

Team 4:

Angelo Arriaga (Team Lead - aarriaga1@sfsu.edu)
Donovan Taylor (Front-End Lead)
Hancun Guo (Front-End)
Edward McDonald (Back-End Lead)
Karl Carsola (Back-End)
Sai Bavisetti (Database)
Maeve Fitzpatrick (Docs Editor)
Sabrina Diaz-Erazo (GitHub Master)

06/20/24

History (Revisions)	
06/20/24	V1.0 Milestone 1 finalized

Executive Summary

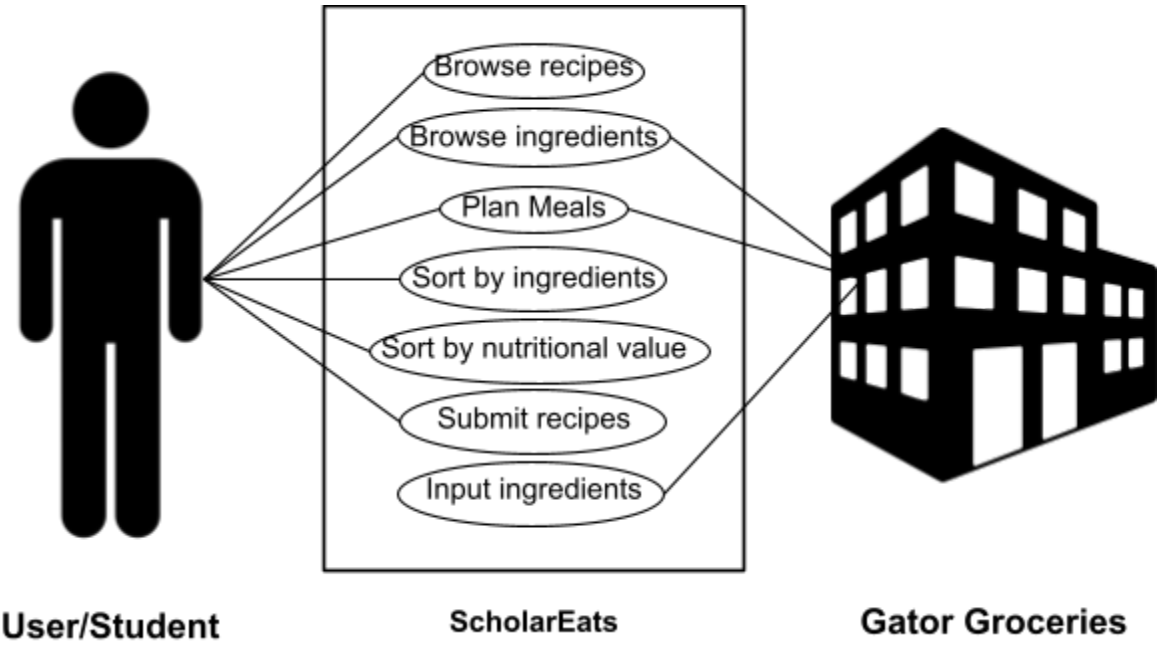
A crisis for many universities in America is that many students do not have the ability to get proper nutrition or even full meals–and if they do, they don't know how to cook them or do not have a full pantry of ingredients. For many students, eating out routinely is not a fiscally-responsible option, and it is often difficult to partake in affordable food programs or campus dining options. Additionally many student-aid programs are difficult to participate in and universities struggle with food waste resulting from un-distributed food.

This framework will allow universities to track their available groceries and have the current stock readily viewable by students. The program will examine the available ingredients that the university is offering, and compile recipes that utilize those ingredients. If intrigued by the week's ingredients or their associated recipes, students can opt-in to receive the selected ingredients and receive instructions for how to cook them. By linking students to the university's grocery offerings, the program supports students in learning how to cook by equipping them with the right tools to create nutritious and enjoyable meals for themselves without waste or financial strain .

Main Use Cases

Use Case 1: Meal Planning

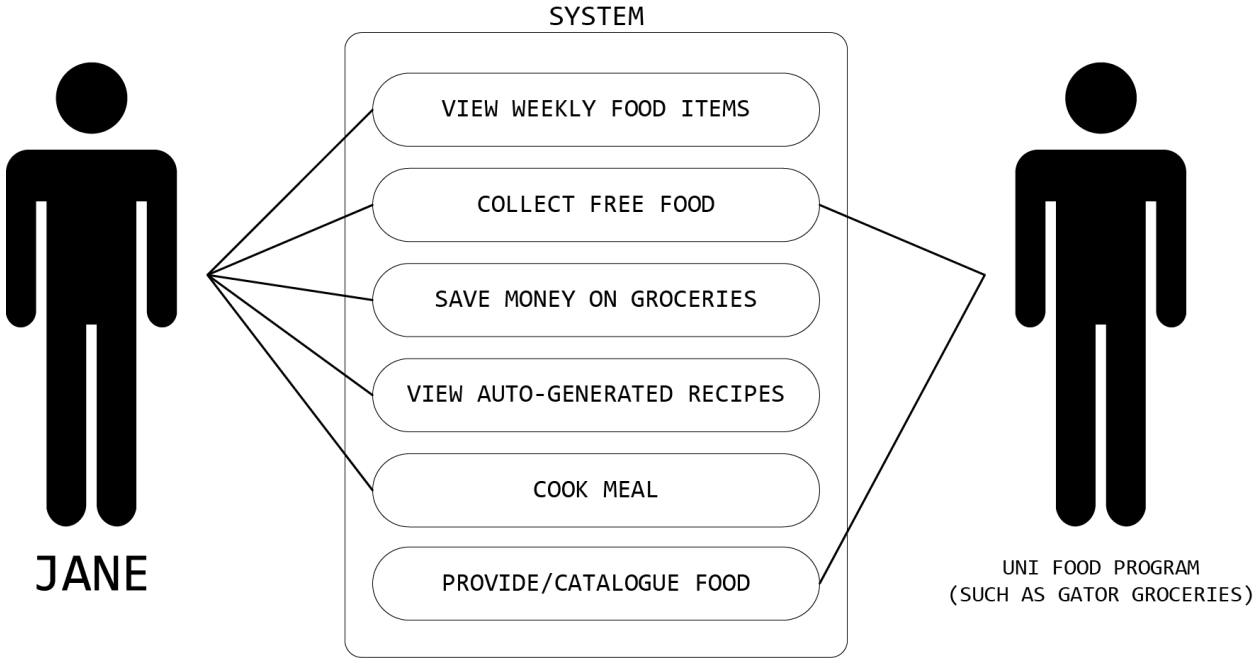
- **Actors**: Greg (student,customer), Gator Groceries(supplier), ScholarEats(system)ffff
- **Assumptions**:
    - Greg is too busy to cook most days
    - Greg is a beginner at cooking
    - Greg already takes advantage of his university's subsidized grocery program
    - Greg has an internet-enabled device
- **Use Case**: Greg is a student at SFSU, He's taking 15 units in the fall and works on the weekends, this leaves him little time to worry about food. He spends far too much of his free time thinking of what he could cook for the day and ends up eating out or boiling ramen noodles most days. He makes use of his university's subsidized grocery program, but the ingredients sit in his fridge slowly degrading. He overhears classmates talking about ScholarEats and how it makes meal planning quick and easy. Since he's already using his university's grocery program, all he needs to do is go online and look at the recipes that are provided for his school's inventory. Once he signs up, he can browse a myriad of recipes using ingredients supplied by Gator Groceries.
- **Benefits for customer**:
    - Stress–free meal planning
    - Beginner friendly recipes
    - Time saving
- **Use case modeling**:



Use Case 2: Money saving

- **Actors**: Jane (Student), ScholarEats (Program)

- **Assumptions**:
    - Jane is poor, and she needs some help affording food for the week
    - Jane has a computer, phone, or any other internet capable device
    - Jane is enrolled in a University with a free food distribution program
- **Use Case**:
    - Jane is a poor college student who needs some help affording her food for the week. Jane utilizes an online service (ScholarEats), which shows her the food available this week at her school's free food program. Along with this information, she also receives a few auto-generated recipes. This further encourages her to go to her university's free food program. After she collects her free food, she has saved a significant amount of money on groceries, and she is ready to cook one of the meals generated by ScholarEats.
- **Benefits for Jane**:
    - Having the available food displayed in advance lets Jane know that she will enjoy the food, saving her time and money
    - Having a recipe planned out encourages Jane to receive the free food
- **Use case modeling**:

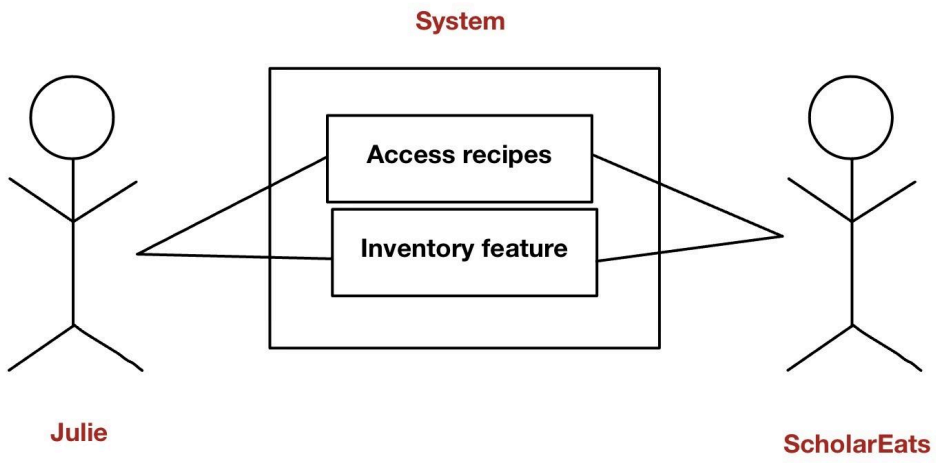


Use Case 3: <u>Waste reduction</u>

- **Actors**: Julie (Student), ScholarEats (Program)
- **Assumptions**:
    - Julie has internet access and devices to go online
- **Use Case**:
    - Julie utilizes the Gator Groceries program every week. However, she notices that she always happens to get more of an ingredient than she needs. She does not know what to do with the extra ingredient so it ends up going to waste. She is saddened by this

because she doesn't like to waste food, and it could have gone to someone who would have used it. In an effort to try to find recipes that utilize all of the ingredients she has from Gator Groceries, she gets stressed out from some recipes requiring more ingredients than what she has and either having too much or too little of an ingredient. Julie then finds out about ScholarEats and installs it on her phone. Julie is amazed with the variety of recipes that are available and how every ingredient from Gator Groceries is utilized in some way. Now, when she goes to Gator Groceries, she can get exactly the type and amount of ingredients she needs without having to worry about any of them going to waste. She also likes that there is an inventory feature so she does not have to worry about showing up to Gator Grocies and having to find out that everything is gone.

- **Benefits for Julie**:
    - Reduction of leftover ingredients since recipes are created based off all available ingredients
    - Inventory updates give Julie an ease of mind since she is able to see how much of an ingredient is left over
- **Use case modeling**:



Use Case 4: <u>Health Consciousness</u>

- **Actors**: Jack, ScholarEats
- **Assumptions**:

    ScholarEats has a system in place to track and manage grocery inventory.
    Jack has access to the inventory system and can view available ingredients.
    ScholarEats provides recipe suggestions based on available ingredients.
    The system supports dietary preferences and health-related needs.
- **Use Case**:

    Jack was always struggling with eating healthily since he is a student and he does not have much time to cook and he also does not know how to cook. He even had no idea about what ingredients he should get to cook a meal. Then Jack find out ScholarEats can help him. Jack logged in and checked available ingredients. He selected a low-carb recipe. The platform

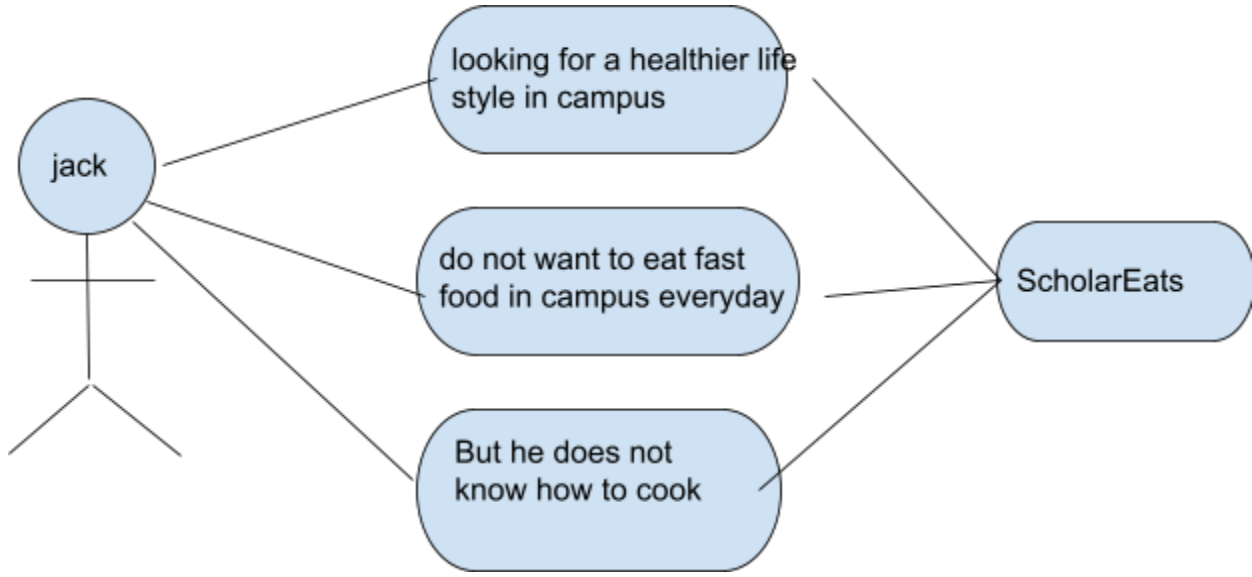
provided necessary ingredients and step-by-step cooking instructions. Jack picked up the ingredients from campus grocery hub and cooked his meal. This program helped him maintain a healthy diet.

- **Benefits for customer**:

  Improved Nutrition: Jack can easily access nutritious meal options that cater to his dietary preferences.

  Healthier Lifestyle: Consistent access to nutritious meals helps Jack maintain better overall health and energy levels.

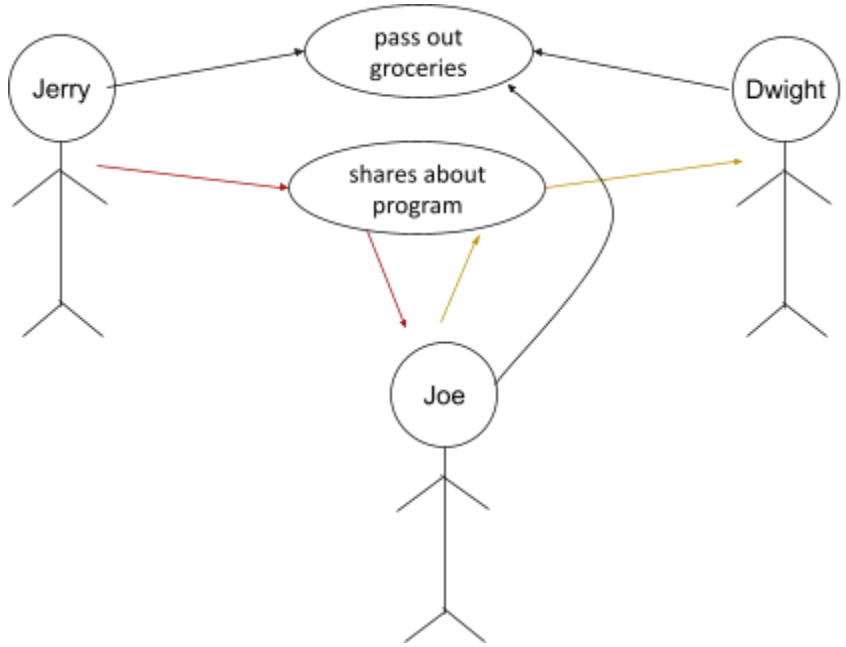
- **Use case modeling**:



Use Case 5: Student Engagement

- **Actors**: Joe (student), Dwight (student), Jerry (student/ScholarEats volunteer), ScholarEats(program)
- **Assumptions**:
  - Joe has a secure and fast wireless Internet connection
  - Dwight and Joe are connected on a social media platform.
- **Use Case**: Jerry thinks being engaged with his university is great, so he volunteers with the ScholarEats program by passing out the weekly groceries. His friends, Joe and Dwight, don't have the same engagement with the school, and Jerry wishes there was a way to get them more involved. After his shift, Jerry runs into his friend Joe. Jerry starts telling Joe about his volunteer shift; Joe isn't familiar with the program, so Jerry explains how it works. Joe, for the first time, is actually interested in something extracurricular that the school offers. He looks into his automatically-enrolled account, and opts in for the next week's groceries. After opting in himself, Joe posts about the program on social media, which catches their friend Dwight's attention; he is intrigued by the program, looks into it himself and also opts in for the groceries. After a few weeks of picking up groceries and enjoying the program, Joe and Dwight realize what Jerry was

talking about when he expressed the value in being involved at school - so much so that they want to help out, so they sign up to volunteer with Jerry.

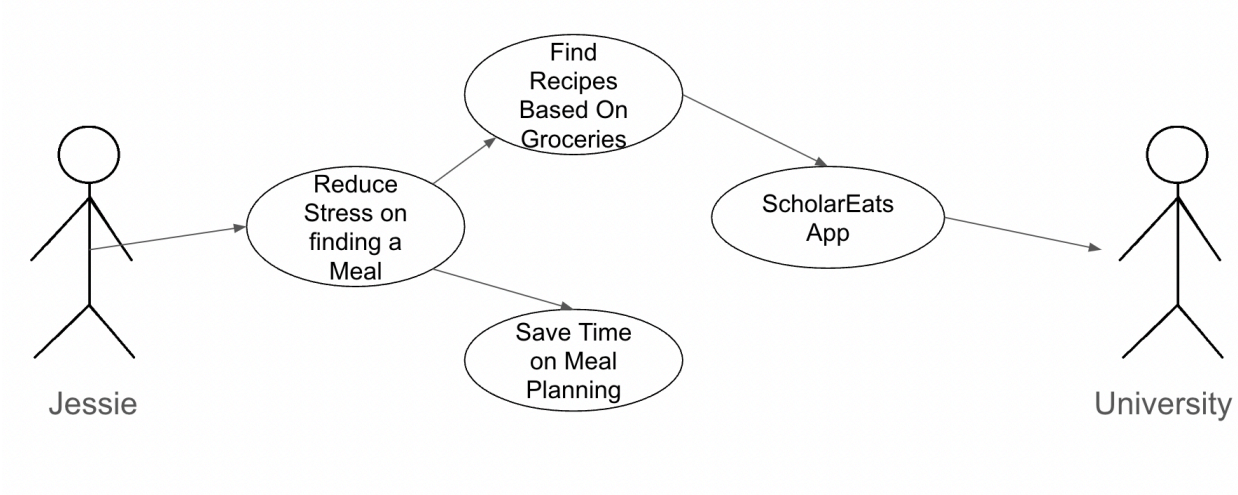
- **Benefits for Jerry**:
    - Gives him an opportunity to be involved in the school
    - Enables him to get more people involved in the school, which is important to him
    - Can simultaneously spend time with his friends, while helping them to get accessible groceries
- **Use case modeling**:
    - 

Use Case 6: <u>Time-saving</u>

- **Actors**: Jessie (Student), University
- **Assumptions**:
    - Jessie is currently a student at a university that offers groceries
    - The university supports the usage of ScholarEats
    - Jessie is capable of cooking
    - Jessie is a full time student and a part time worker
- **Use Case**:
    - Jessie, a student who is currently attending full-time at her university and a part-time worker at a convenience store, has a hard time balancing her school and work life. Despite the many efforts that she has made to manage her time properly she always falls short with the time she has for herself. Luckily, she found that she can save more time with the use of ScholarEats where Jessie can easily get recipes based on the groceries that her university has. With this, she is able to be stress free and able to spend less time on finding what to make.
- **Benefits for customer**:

- Reduces Stress for the customer. The customer no longer needs to worry what the customer needs to eat.
- Reduces Time for the customer. The customer no longer needs to waste time to find a recipe.
- The customer has accessibility for ScholarEats anywhere and anytime.

- **Use case modeling**:

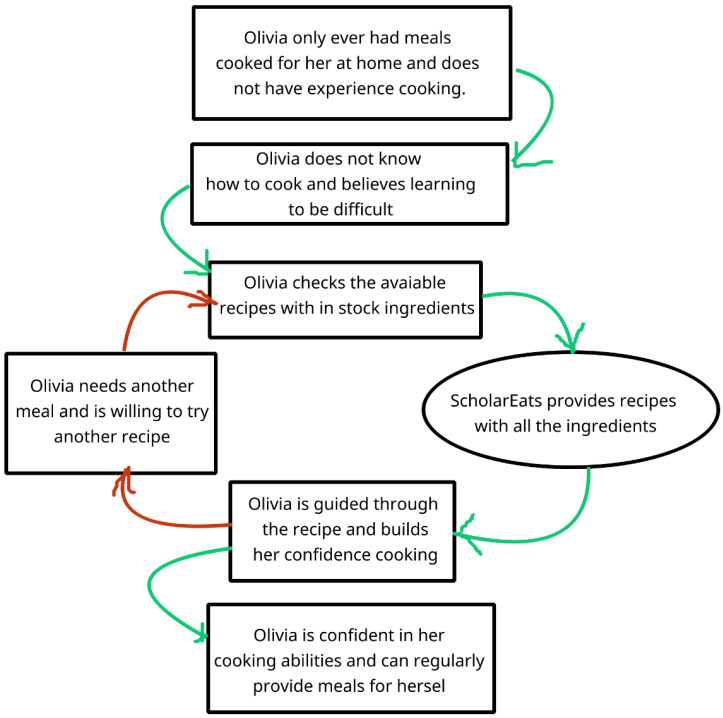


Use Case 7: <u>Cooking Education</u>

- **Actors**:
  - Olivia (student)
  - University Faculty
- **Assumptions**:
  - Olivia is a young woman away from home for the first time. Her family cooked dinner at home every single day and she is having to adapt to a life away from home where she doesn't have parents ensuring that she eats a full meal every day after studying.
  - Olivia does not have any experience cooking meals at home and does not know where to begin. She often has to resort to eating food from the vending machine in the lobby of her dorm because she doesn't always have a chance to go to food spot on campus while they are open with her schedule.
  - The University has a food pantry in which cheap, or even free, groceries are available and faculty or student-workers that can manage the program.
- **Use Case**:
  - Olivia can use this app that her school manages locally to see which recipes have ingredients readily available within the university's food pantry.
  - Olivia confirms her intention of picking up a set of ingredients earlier in the day and gets notified when it is ready to pick up.
  - Olivia picks up the ingredients.
  - Olivia goes to her dorm and the app gives her step by step instructions on how to cook.
- **Benefits for customer**:

- Motivation to learn how to cook which is a useful life skill that avoids excessive spending habits and enables her to feed herself.

- **Use case modeling**:



Use Case 8: Community Recipe Submission

- **Actors**:
    1) Emily wants to share personal recipes with the university community. [Primary Actor]
    2) Scholar Eats Platform [System]
- **Assumptions**:
    1) Emily has access to the internet and a device to access the ScholarEats Platform.
    2) Emily has a recipe with a complete list of ingredients and cooking instructions ready to submit.
- **Use Case**:

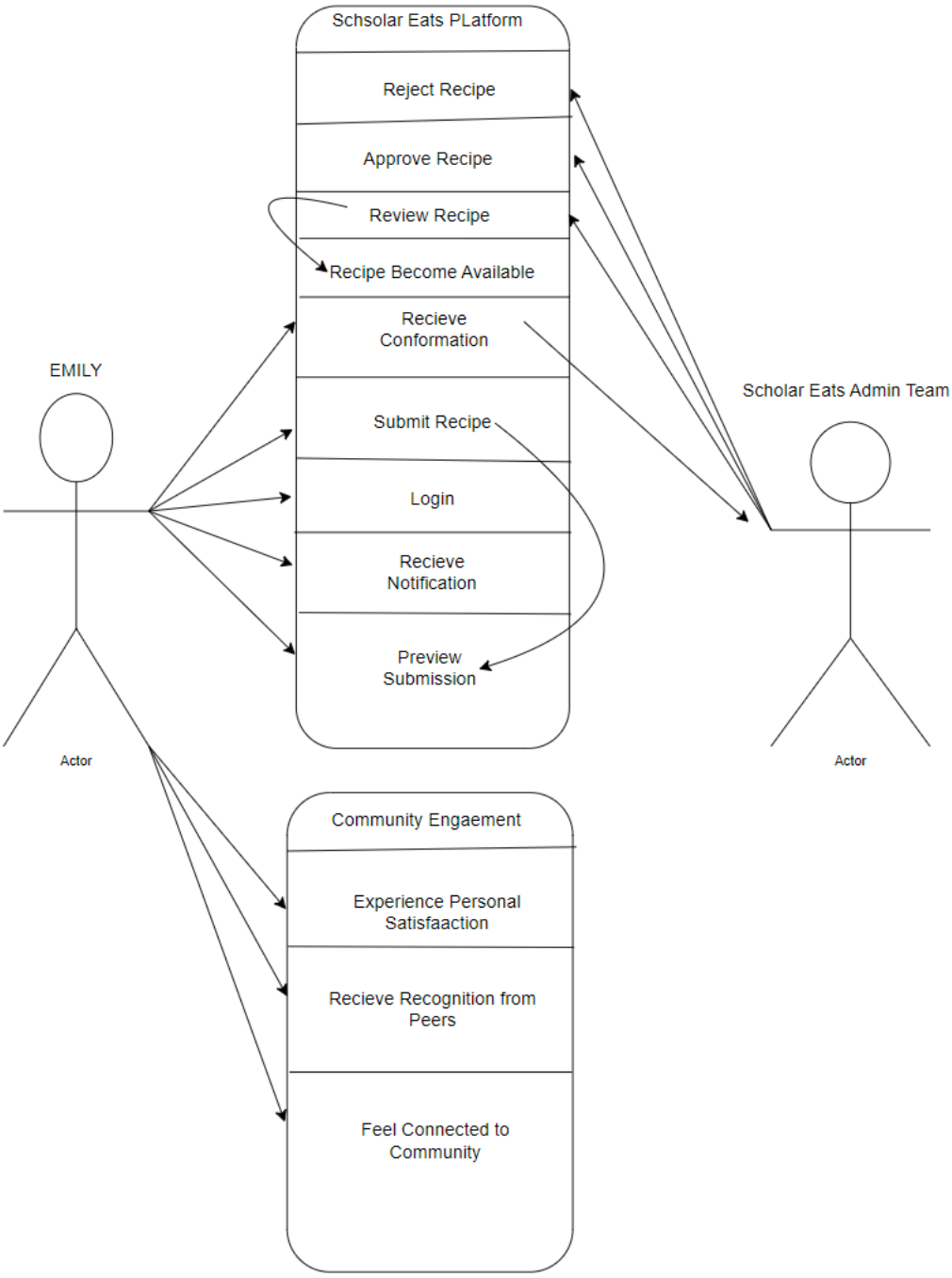
    Emily, thrilled to share a unique dish she recently perfected, decides to contribute her recipe to the ScholarEats community. Motivated by a desire to engage with her peers and enrich the campus dining experience, she logs into the ScholarEats platform, eager to introduce others to her culinary creation. The platform's user-friendly interface seamlessly guides Emily through the recipe submission process, ensuring she provides all necessary details, including the ingredient list, step-by-step instructions, and optional photos of the dish.

    As Emily fills out the submission form, she finds the fields well-organized and easy to navigate, significantly reducing the hassle typically associated with digital submissions. She appreciates the option to preview her entry before finalizing it, ensuring accuracy and completeness. Upon submission, the ScholarEats system quickly processes the recipe, displaying a confirmation message that reassures Emily that her contribution has been received and is under review.

    The review process, conducted by the ScholarEats administration team, ensures that the recipe meets health and safety standards as well as the platform's quality guidelines. Once approved, the recipe becomes available to the entire student body, enhancing the communal recipe database and encouraging culinary exploration among peers. Emily receives a notification when her recipe is live, adding a sense of accomplishment and fostering a deeper sense of community involvement.

- **Benefits for customer**:
    1) Community Engagement: Emily feels connected to the campus community by sharing her personal recipes.
    2) Recognition: Contributing to the platform can bring Emily recognition from peers who try and enjoy her recipes.
    3) Satisfaction: Emily experiences personal satisfaction in knowing her culinary creations can help others diversify their meals and cooking experiences.

- **Use case 8 modeling**:



List of Main Data Items and Entities

What do we need to store:

- User Account(Global)
    - Email
    - Password (ENCRYPTED with bcrypt? NOT PLAINTEXT)
    - Username
    - UserType
        - Student
        - Administrator
    - Phone Number (Notifications)
    - Allergies
    - CreationDate (Annual baskets?)
    - University
- User Preferences(Global)
    - Light Mode /Dark Mode
    - Notifications (Y/N)
    - Asdasdad
- Recipe Book(Global)
    - Recipes
        - Ingredients
            - Allergens
        - Instructions
        - Cookware
        - Time Required
        - Servings
        - Nutritional Values
        - Pictures
        - Reviews
        - Difficulty
        - Author
        - Dietary restrictions
            - VEGAN
            - VEGETARIAN
            - HIGH PROTEIN
            - LOW-FAT
            - KOSHER
            - DAIRY-FREE
            - HALAL
- Food Pantry(Local)
    - Ingredients
        - Size/Quantity

- Expiration Dates
- Allergens

Initial List of Functional Requirements

1. USERS SHALL BE ABLE TO MAKE ACCOUNTS WITH VALID UNIVERSITY EMAIL.
    - Users shall be able to make accounts utilizing a valid university email, such as those ending with *@mail.sfsu.edu*. This will automatically enroll them in the corresponding university.
2. USERS SHALL BE ABLE TO CHANGE PROFILE PHOTO
    - Users shall be able to upload an image from their device, which will be viewable by other users as their profile image.
3. USERS SHALL BE ABLE TO CHANGE THEIR PASSWORD
    - Users shall be able to change their password, with email verification. This will be done by inputting two passwords and ensuring that they match, while fulfilling the password requirements.
4. USERS SHALL BE ABLE TO EDIT PROFILE BIO
    - Users shall be able to edit a short blurb about themselves within their user bio of up to 150 characters,
5. USERS SHALL BE ABLE TO EDIT THEIR USERNAME
    - Users shall be able to change their username, ensuring that their username is not already in use by another user.
6. USERS SHALL BE ABLE TO ADD THEIR PREFERRED PRONOUNS
    - Users shall be able to edit their preferred pronouns through a dropdown list of pre-selected pronouns.
7. USERS SHALL BE ABLE TO SET DIETARY RESTRICTIONS
    - Users shall be able to select dietary restrictions from an extensive dropdown list, including restrictions such as vegan, vegetarian, halal, etc. Users will be able to select multiple dietary restrictions.
8. USERS SHALL BE ABLE TO SET ALLERGIES
    - Users shall be able to select allergy restrictions from an extensive dropdown list, including restrictions such as peanuts, shellfish, dairy, etc. Users will be able to select multiple allergy restrictions.
9. USERS SHALL BE ABLE TO FAVORITE INGREDIENTS
    - Users shall be able to favorite ingredients, which will locally sort that ingredient as “favorited.”
10. USERS SHALL BE ABLE TO VIEW RECIPES
    - Users shall be able to click on a suggested recipe based on the available ingredients and view the recipe.
11. USERS SHALL BE ABLE TO FAVORITE RECIPES
    - Users who view a recipe that the user likes will be able to favorite the recipes for later use.
12. USERS SHALL BE ABLE TO VIEW OWN FAVORITE RECIPES
    - Able to view the users own favorite recipes so that the user is able to cook it.
13. USERS SHALL BE ABLE TO VIEW HISTORY OF RECEIVED RECIPES

- Users who didn't favorite a recipe but changed their minds later are able to view their history to see the said recipe.

14. USERS SHALL BE ABLE TO REVIEW/RATE RECIPES
    - If the user liked or didn't like the recipe they can give their feedback on the recipe.
15. USERS SHALL BE ABLE TO SORT RECIPES BY RATING
    - If the user wants the best or worst rated recipe, the user is able to sort by rating through descending or ascending order.
16. USERS SHALL BE ABLE TO SORT RECIPES BY CALORIES
    - If the user is on a diet and wants a lower calorie recipe, the user can sort the recipes by calories through descending or ascending order.
17. USERS SHALL BE ABLE TO SORT RECIPES BY PROTEIN
    - For users who are interested in increasing their protein intake (fitness reasons, etc), they can find recipes that have a high level of protein
18. USERS SHALL BE ABLE TO SORT RECIPES BY FAT
    - User can find recipes with lower levels of fat
19. USERS SHALL BE ABLE TO SORT RECIPES BY FIBER
    - Users can find recipes that have high or low amounts of fiber depending on their dietary needs
20. USERS SHALL BE ABLE TO FILTER RECIPES BY INGREDIENTS
    - Users who don't want an ingredient in their recipe, the user can filter the recipe with what the user wants.
21. USERS SHALL BE ABLE TO FILTER RECIPES BY DIETARY RESTRICTIONS
    - Halal, Kosher
22. USERS SHALL BE ABLE TO FILTER RECIPES BY COOKING AIDS REQUIRED
    - Stovetop, oven, microwave
23. USERS SHALL BE ABLE TO FILTER RECIPES BY DIFFICULTY
    - Users who are new to cooking can find the simplest recipes easily, while experienced cooks can find more complex recipes, as well as recipes in between
24. USERS SHALL BE ABLE TO VIEW OTHER USER'S FAVORITE INGREDIENTS
    - If a student knows they have a similar taste in food as another user, they can see what ingredients they've enjoyed cooking with
25. USERS SHALL BE ABLE TO VIEW OTHER USER'S FAVORITE RECIPES
    - If a student knows they have a similar taste in food as another user, they can see what that other user has cooked and enjoyed before
26. USERS SHALL BE ABLE TO VIEW AVAILABLE INGREDIENTS
    - A user can look ahead of time at what is being offered to see if they are interested in those ingredients and/or the recipes associated with them
27. USERS SHALL BE ABLE TO SHARE RECIPES
    - A user who liked a recipe are able to share it to another user.
28. USERS SHALL BE ABLE TO SUBMIT RECIPES
    - Available to be viewed by fellow students enrolled in the food program
29. USERS SHALL BE ABLE TO TRANSFER ENROLLMENT IN UNIVERSITY FOOD PROGRAMS

- Migrate information/favorites/history to a new account at a different school(?)

30. USERS SHALL BE ABLE TO SWITCH BETWEEN LIGHT AND DARK MODE
    - Depending on the user preference, the user is able to switch the style of the user's page by their liking.
31. USERS SHALL BE ABLE TO ALLOW NOTIFICATIONS
    - MAILING LIST, TEXT NOTIFICATIONS
    - Users can enable notifications on their device to be alerted of the week's groceries and recipes
32. USERS SHALL BE ABLE TO UPLOAD PHOTOS OF THE RECIPES THEY COOK FOR REVIEWS
    - If users have cooked a recipe, they can share how it turned out visually
33. USERS SHALL BE ABLE TO REPORT PHOTOS OR REVIEWS
    - If other users are violating community guidelines, users can report them to administration
34. USERS SHALL BE ABLE TO BLOCK OTHER USERS
    - If a user does not wish to view the content or account of another user, they will be able to block that user. This will restrict that user from viewing the content of the other user, and vice versa.
35. USERS SHALL BE ABLE TO REPORT USER ACCOUNTS
    - If a user includes content that violates community guidelines, other users will be able to send an automated report to the university administrator.
    -
36. USERS AND ADMINISTRATORS SHALL BE ABLE TO FOLLOW THE EXPIRATION DATE CLOSELY AND TAKE THE NECESSARY ACTIONS.
    - To best utilize grocery inventory, the expiration dates of the items will be tracked to use first-in-first-out system
37. ADMINISTRATORS SHALL BE ABLE TO AUTHENTICATE THE USERS.
    - Administrators can prevent duplicate or spam accounts from joining
38. ADMINISTRATORS SHALL BE ABLE TO EDIT THE INGREDIENT LIST
    - Depending on the university's ingredients, if the ingredients change for that week, the administrator should be able to change the ingredient list.
39. ADMINISTRATORS SHALL BE ABLE TO BLACKLIST/ WHITELIST A USER
    - If a user violates community agreements, the administration has the ability to remove them
40. ADMINISTRATORS SHALL BE ABLE TO DELETE USER POSTED RECIPES
    - If a user posts a recipe that violate community agreements, the administration has the ability to remove it
41. ADMINISTRATORS SHALL BE ABLE TO MAKE UNIVERSITY-WIDE ANNOUNCEMENTS
    - If there is a special announcement that the university wants to make, administrators are able to make the announcements.
42. ADMINISTRATORS SHALL BE ABLE TO PROMOTE RECIPES
    - "Gator-fried rice!" "You're telling me a gator fried this rice?"
    - "Golden-gate fries"

43. ADMINISTRATORS SHALL BE ABLE TO VIEW ALL USER-REPORTS
    - Users who reported another user for any reason, the administrator should be able to review the report and take action if needed.
44. ADMINISTRATORS SHALL BE ABLE TO REMOVE RECIPES OR REVIEWS
    - If a user posted an inappropriate review or recipe, the administrator is able to remove it.
45. ADMINISTRATORS SHALL BE ABLE TO REMOVE USER ACCOUNTS
    - If a user doesn't follow community guidelines/ agreements or if there is something technically wrong, the administrator is able to remove the user account from the system.
46. SYSTEM SHALL BE ABLE TO GENERATE RECOMMENDED RECIPES
    - Based on the user's preferences and weekly ingredients, system will show recipes that correspond to them
47. SYSTEM SHALL AUTOMATICALLY ENROLL USERS IN CORRESPONDING UNIVERSITY FOOD PROGRAMS
    - As a student of the university, they get an account automatically made for them with the program.
48. SYSTEM SHALL ENSURE RECIPES ARE AVAILABLE FOR ALL (MAJOR) DIETARY RESTRICTIONS
    - I.e. VEGAN/VEGETARIAN/PESCATARIAN/LACTOSE-INTOLERANCE
    - Administrators shall be able to review the recipes submitted before posted on the system.
49. SYSTEM SHALL TELL ADMINISTRATORS WHEN FOOD HAS SPOILED
    - If food has expired/spoiled, the system will notify the administration as soon as possible so they can best respond/make sure they don't distribute the items
50. SYSTEM SHALL TELL ADMINISTRATORS WHEN AN INGREDIENT HAS RUN OUT OF STOCK
    - If a food item has run out, the system will notify the administration as soon as possible so they can best respond/possibly offer something else

List of Non-Functional Requirements

- Privacy & Security:
    - Users must check and confirm their preferences regarding personal information upon account creation.
    - Passwords MUST be encrypted and NOT PLAINTEXT for security reasons
    - Emails must be validated (be an email from the universities domain)
    - WE WILL NOT SELL YOUR DATA TO AI
- Expected Load:
    - Support 200,000 unique students
- Response Time:
    - System shall respond visually within 7 seconds
    - File Size of Photos shall not exceed 10MiB
    - File formats for photos shall include:
        - .png
        - .jpeg
        - .heic
        - .heif
- Stylistic:
    - Each page shall has the program's logo visible in the header
- Hardware:
    - Display dimensions supported(css):
        - Phone
        - Tablet
        - Monitor
- Software:
    - We will support:
        - Firefox
        - Chromium

Competitive Analysis

Feature	Company			
	HelloFresh	FoodCombo	Too Good To Go	Imperfect Foods
Strengths	• Lots of variety with different diets • Flexible with skipping meals • Several discounts available	• Robust meal planning • X missing ingredients to next best recipe • Lots of options to sort by	• Reduces food waste • Affordable meal options • Easy to use app interface	• Simple model - sustainable grocery delivery • Strong control over delivery cadence
Weaknesses	• Expensive • Does not have options for 1 person (only 2 or 4 people)	• No way of procuring ingredients • Long load times • hard to select ingredients	• Limited availability depending on location • Quality of food can be inconsistent	• Food is surplus - cosmetic imperfections • Weekly delivery, slot based • Website is cluttered
Pricing	• $8.99 per serving ($10.49 per serving if buying 2 meals per week for 2 people)	• Free online tool	• Prices vary; typically significantly lower than regular prices	• Signup is free, just pay for groceries and delivery fee
Social Media	• Instagram, Facebook, Television, Podcasts	• none	• Instagram, Facebook, Twitter	• Instagram, Facebook, Twitter, Tiktok, Pinterest
Onboarding Experience	• 5 relatively simple steps	• Easy - start from the homepage	• Simple app download and account creation	• Free sign up, start by entering zip code

Feature	HelloFresh	FoodCombo	Too Good To Go	Imperfect foods	ScholarEats
Ease of Obtaining Ingredients	++	-	++	++	+

Has Recipes Readily Available	+	+	+	-	++
Ease of sorting by dietary restrictions	+	+	-	-	++
Regular update of Inventory	+	-	-	-	+
Automatic Enrollment	-	-	-	-	+

Summary

ScholarEats can stand out by focusing on the specific needs of university students. It can integrate with existing college grocery distribution systems, making it easier for students to find ingredients they know and love. This strategy will help ScholarEats grow and become a valuable tool for students, helping them with meal planning, cooking, and healthy eating. ScholarEats will excel in presenting users with quick and easy recipes which can be sorted by various dietary restrictions, with a constantly updated list of groceries that students can pick up from their local campus grocery distribution system. This will make it easier for our users to take advantage of programs that will benefit them that they may not know about or may be too lazy to look into.

Checklist

- Team found a time slot to meet outside of the class [**DONE**]
- Github master chosen [**DONE**]
- Team decided and agreed together on using the listed SW tools and deployment server [**DONE**]
- Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing [**ON TRACK**]
- Team lead ensured that all team members read the final M1 and agree/understand it before submission [**DONE**]
- Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.) [**ISSUE**]
    - Team lead and GitHub Master have not discussed how we are going to handle testing our team's code before pushing to main yet, but we have a meeting scheduled June 28th to do so.

High-Level System Architecture and Technologies Used

- Host: Amazon Web Service
    - EC2
    - RCS
- Package Manager: NodeJS
- Node Packages being Utilized:
    - Bcrypt
    - Express
    - Express-Handlebars
    - Express-Session
    - Handlebars
    - Path
    - mysql-server
- Database: MySQL
- Primary Back-End Language: JavaScript
- Additional Supportive Tools:
    - Docker

List of team contributions

Team Member	Role	Score	Contributions
Donovan Taylor	Frontend Lead	10/10	● Drafted the HTML templates for the Team's About Pages ● Drafted CSS for the Team's About Pages ● Took the initiative to lay the groundwork for utilizing Node.js in the tech stack ● Led meetings for relevant tasks
Hancun Guo	Frontend	7/10	● Assisted Front-end Lead ● Assisted Documentation Editor
Edward Mcdonald	Backend Lead	9.5/10	● Installed and configured Tech Stack ● Identified Issue with deployment and incorporated pm2 into the tech stack to make deployment consistent and easier. ● Assisted the Git-Hub Master with included procedures within the Git Repository ● Led meetings for relevant tasks
Karl Carsola	Backend	6/10	● Assisted Back-end Lead ● Assisted Documentation Editor
Sai Bavisetti	Database	5/10	● Assisted Documentation Editor
Maeve Fitzpatrick	Docs Editor	10/10	● Coordinated with many members of the team simultaneously to complete large sections of the Milestone 1 documentation. ● Led during relevant tasks
Sabrina Diaz-Erazo	Github Master	10/10	● Took the initiative to start the RCS Database Instance while the Team Lead was unavailable ● Ensured all procedures and required documentation was present within the Github Repository ● Coordinated with other team members to start/stop testing as required.

Milestone 2 Version 2

SW Engineering CSC648/848 Summer 2024

ScholarEats

Team 4:

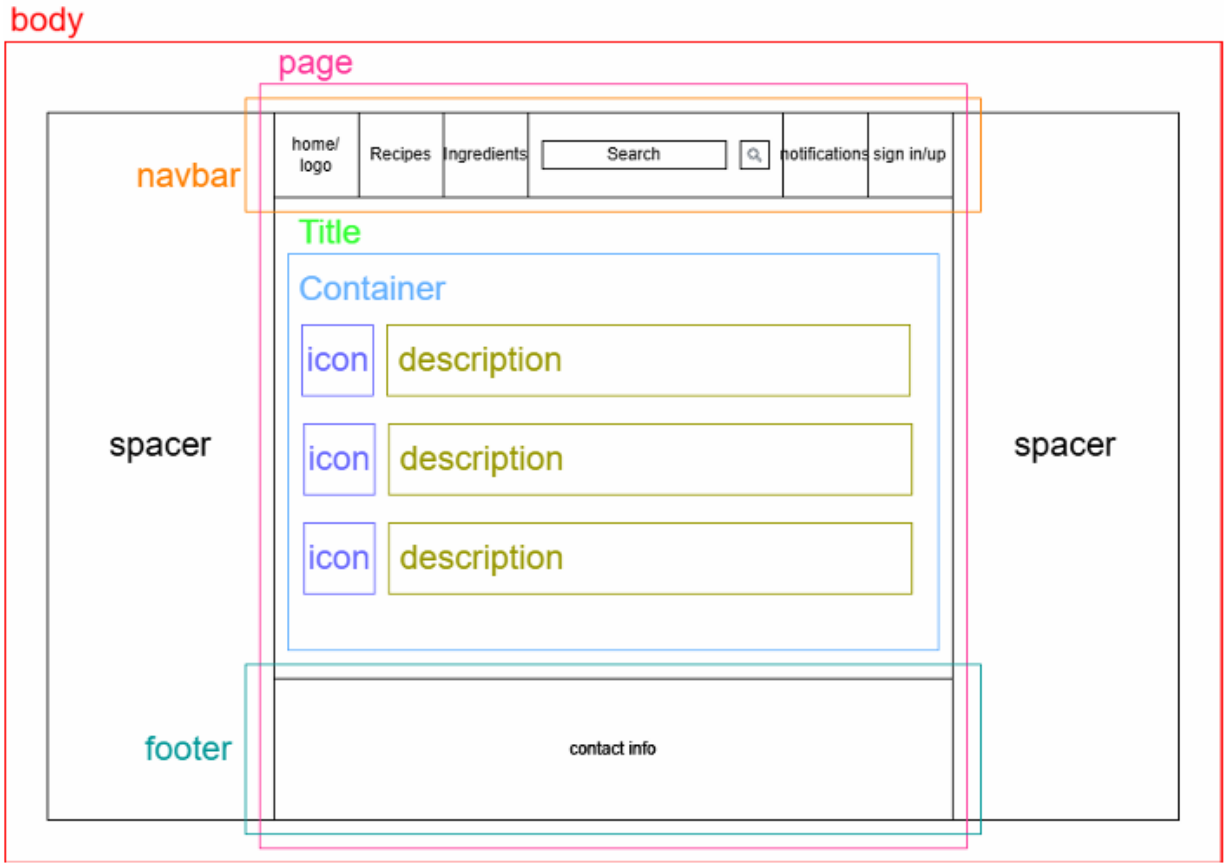
Angelo Arriaga	Team Lead
Donovan Taylor	Front-End Lead
Hancun Guo	Front-End
Tina Cho	Front-End
Edward McDonald	Back-End Lead
Karl Carsola	Back-End
Sai Bavisetti	Database
Maeve Fitzpatrick	Docs Editor
Sabrina Diaz-Erazo	GitHub Master

Milestone 2 (V2) 2024/07/22

History & Revisions

2024-06-20	Milestone 1 v1 Submission
2024-06-26	Milestone 1 v1 Re-submission
2024-07-09	Milestone 2 v1 Submission
2024-07-22	Milestone 2 v2 Submission

1. Data Definitions



Database:

activity_log	log_id
	user_id
	action
	timestamp

admin	admin_id
	uuid
	username

	password_hash
	verification_status
	accessibility

email_log	email_log_id
	user_id
	email
	timestamp

expired_products	ingredient_id
	Name
	expiration_date
	quantity

has	user_id
	university_id

ingredients	ingredient_id
	name

manages	admin_id
	user_id

notifications	notification_id
	user_id
	message
	is_read

recipe_ingredient	recipe_id
	ingredient_id

recipes	Unnamed: 0
	recipe_name

	prep_time
	cook_time
	total_time
	servings
	yield
	ingredients
	directions
	rating
	cuisine_path
	nutrition
	timing
	img_src
	dietary restrictions
	calories
	protein
	fat
	fiber
	difficulty
	cooking tip
	user_id
	MyUnknownColumn

roles	role_id
	role_name

sessions	session_id
	user_id
	session_start
	session_end
	ip_address

	user_agent

store	store_id
	Name
	ingredient_id
	user_id
	quantity
	expiration_date

university	university_id
	name
	email_suffix

user_info	user_info_id
	user_id
	profile_photo
	bio
	bio
	pronouns
	allergies
	dietary_restrictions
	favorited_recipes
	favorited_ingredients
	university

user_recipes	user_recipe_id
	user_id
	recipe_name
	prep_time
	cook_time
	total_time

	yield
	directions
	img_source
	nutrition
	servings
	dietary_restrictions
	calories
	fat
	protein
	fiber
	difficulty
	cooking_tip
	accessibility
	is_approved
	admin_id

Users	user_id
	uuid
	email
	username
	password_hash
	verification_status
	accessibility
	blacklist
	role_id
	university

2. Initial List of Functional Requirements

PRIORITY 1 - Critical

- Administrators
    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.
- System
    1. SYSTEM SHALL AUTOMATICALLY ENROLL USERS IN CORRESPONDING UNIVERSITY FOOD PROGRAMS
    2. SYSTEM SHALL BE ABLE TO GENERATE RECOMMENDED RECIPES
    3. SYSTEM SHALL TELL ADMINISTRATORS WHEN AN INGREDIENT HAS RUN OUT OF STOCK
    4. SYSTEM SHALL TELL ADMINISTRATORS WHEN FOOD HAS SPOILED
- Users
    1. USERS SHALL BE ABLE TO CHANGE THEIR PASSWORD
    2. USERS SHALL BE ABLE TO EDIT THEIR USERNAME
    3. USERS SHALL BE ABLE TO FILTER RECIPES BY COOKING AIDS REQUIRED
    4. USERS SHALL BE ABLE TO FILTER RECIPES BY DIETARY RESTRICTIONS
    5. USERS SHALL BE ABLE TO FILTER RECIPES BY DIFFICULTY
    6. USERS SHALL BE ABLE TO FILTER RECIPES BY INGREDIENTS
    7. USERS SHALL BE ABLE TO MAKE ACCOUNTS WITH VALID UNIVERSITY EMAIL.
    8. USERS SHALL BE ABLE TO SET ALLERGIES

9. USERS SHALL BE ABLE TO SET DIETARY RESTRICTIONS
10. USERS SHALL BE ABLE TO SORT RECIPES BY CALORIES
11. USERS SHALL BE ABLE TO SORT RECIPES BY FAT
12. USERS SHALL BE ABLE TO SORT RECIPES BY FIBER
13. USERS SHALL BE ABLE TO SORT RECIPES BY PROTEIN
14. USERS SHALL BE ABLE TO VIEW AVAILABLE INGREDIENTS
15. USERS SHALL BE ABLE TO VIEW RECIPES

PRIORITY 2 - Desired

- Administrators
    1. ADMINISTRATORS SHALL BE ABLE TO REMOVE RECIPES OR REVIEWS
- Systems
    1. SYSTEM SHALL ENSURE RECIPES ARE AVAILABLE FOR ALL (MAJOR) DIETARY RESTRICTIONS
- Users
    1. USERS SHALL BE ABLE TO FAVORITE INGREDIENTS
    2. USERS SHALL BE ABLE TO FAVORITE RECIPES
    3. USERS SHALL BE ABLE TO REVIEW/RATE RECIPES
    4. USERS SHALL BE ABLE TO SORT RECIPES BY RATING
    5. USERS SHALL BE ABLE TO VIEW OWN FAVORITE RECIPES

PRIORITY 3 - Opportunistic

- Administrators
    1. ADMINISTRATORS SHALL BE ABLE TO DELETE USER POSTED RECIPES
    2. ADMINISTRATORS SHALL BE ABLE TO MAKE UNIVERSITY-WIDE ANNOUNCEMENTS
    3. ADMINISTRATORS SHALL BE ABLE TO PROMOTE RECIPES

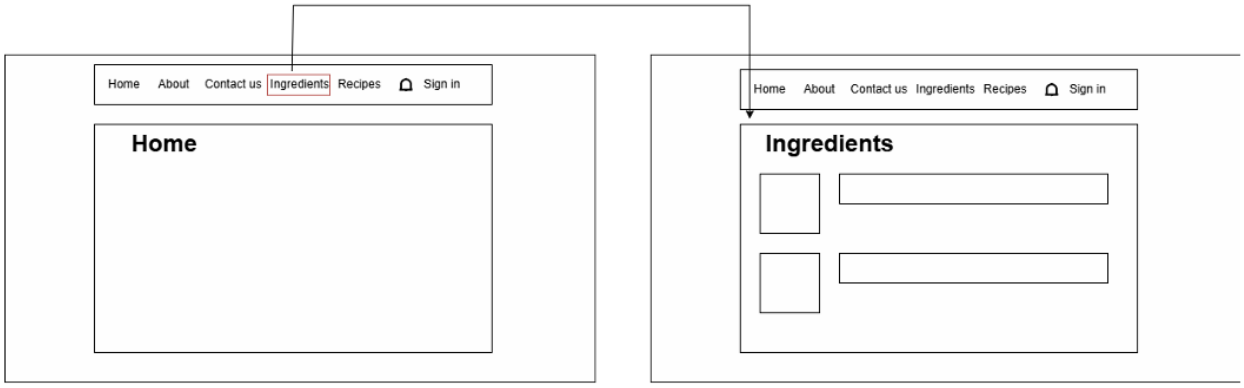
4. ADMINISTRATORS SHALL BE ABLE TO VIEW ALL USER-REPORTS

- Users
    1. USERS SHALL BE ABLE TO ADD THEIR PREFERRED PRONOUNS
    2. USERS SHALL BE ABLE TO ALLOW NOTIFICATIONS
    3. USERS SHALL BE ABLE TO BLOCK OTHER USERS
    4. USERS SHALL BE ABLE TO CHANGE PROFILE PHOTO
    5. USERS SHALL BE ABLE TO EDIT PROFILE BIO
    6. USERS SHALL BE ABLE TO REPORT PHOTOS OR REVIEWS
    7. USERS SHALL BE ABLE TO REPORT USER ACCOUNTS
    8. USERS SHALL BE ABLE TO SHARE RECIPES
    9. USERS SHALL BE ABLE TO SUBMIT RECIPES
    10. USERS SHALL BE ABLE TO SWITCH BETWEEN LIGHT AND DARK MODE
    11. USERS SHALL BE ABLE TO TRANSFER ENROLLMENT IN UNIVERSITY FOOD PROGRAMS
    12. USERS SHALL BE ABLE TO UPLOAD PHOTOS OF THE RECIPES THEY COOK FOR REVIEWS
    13. USERS SHALL BE ABLE TO VIEW HISTORY OF RECEIVED RECIPES
    14. USERS SHALL BE ABLE TO VIEW OTHER USER'S FAVORITE INGREDIENTS
    15. USERS SHALL BE ABLE TO VIEW OTHER USER'S FAVORITE RECIPES

3. UI Mockups and Storyboards

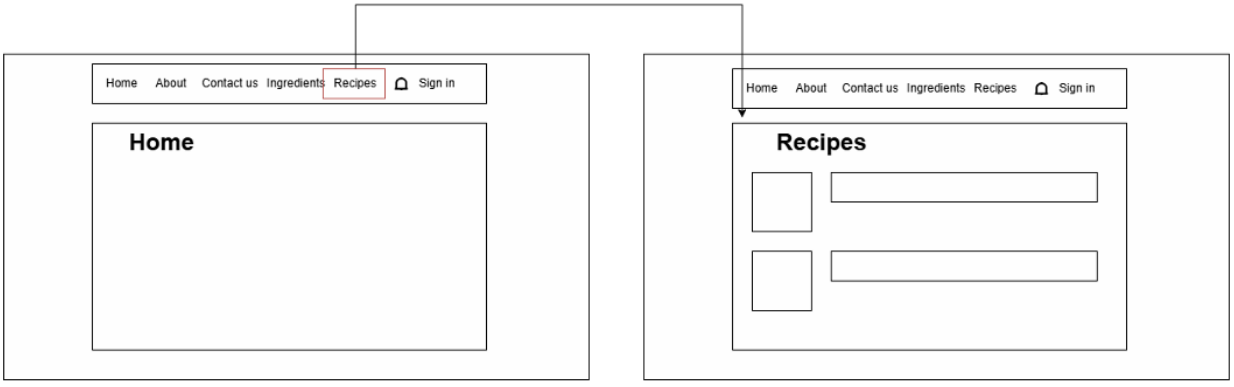
Use Case 1: Meal Planning:

Since the student is already using his university's grocery program, all he needs to do is go online and look at the recipes that are provided for his school's inventory. Once he signs up, he can browse a myriad of recipes using ingredients supplied by Gator Groceries.



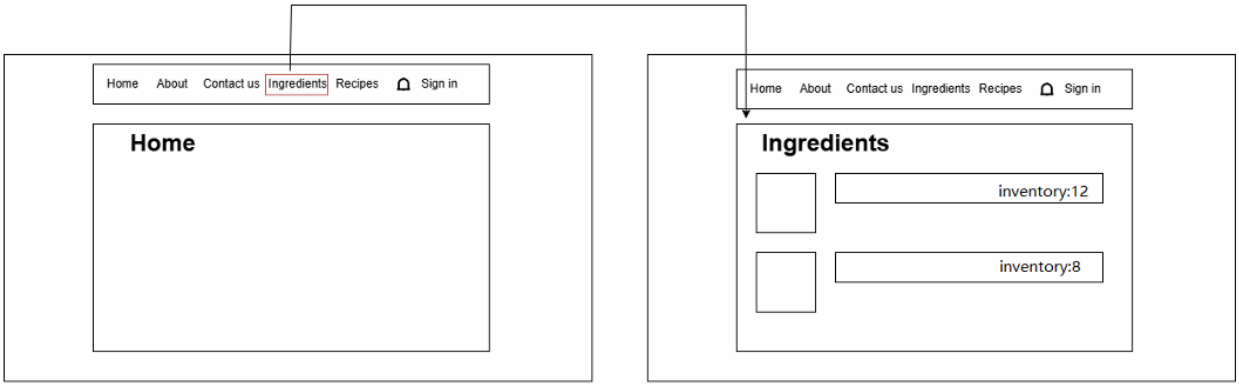
Use Case 2: Money Saving:

Jane utilizes an online service (ScholarEats), which shows her the food available this week at her school's free food program. Along with this information, she also receives a few auto-generated recipes. This further encourages her to go to her university's free food program.



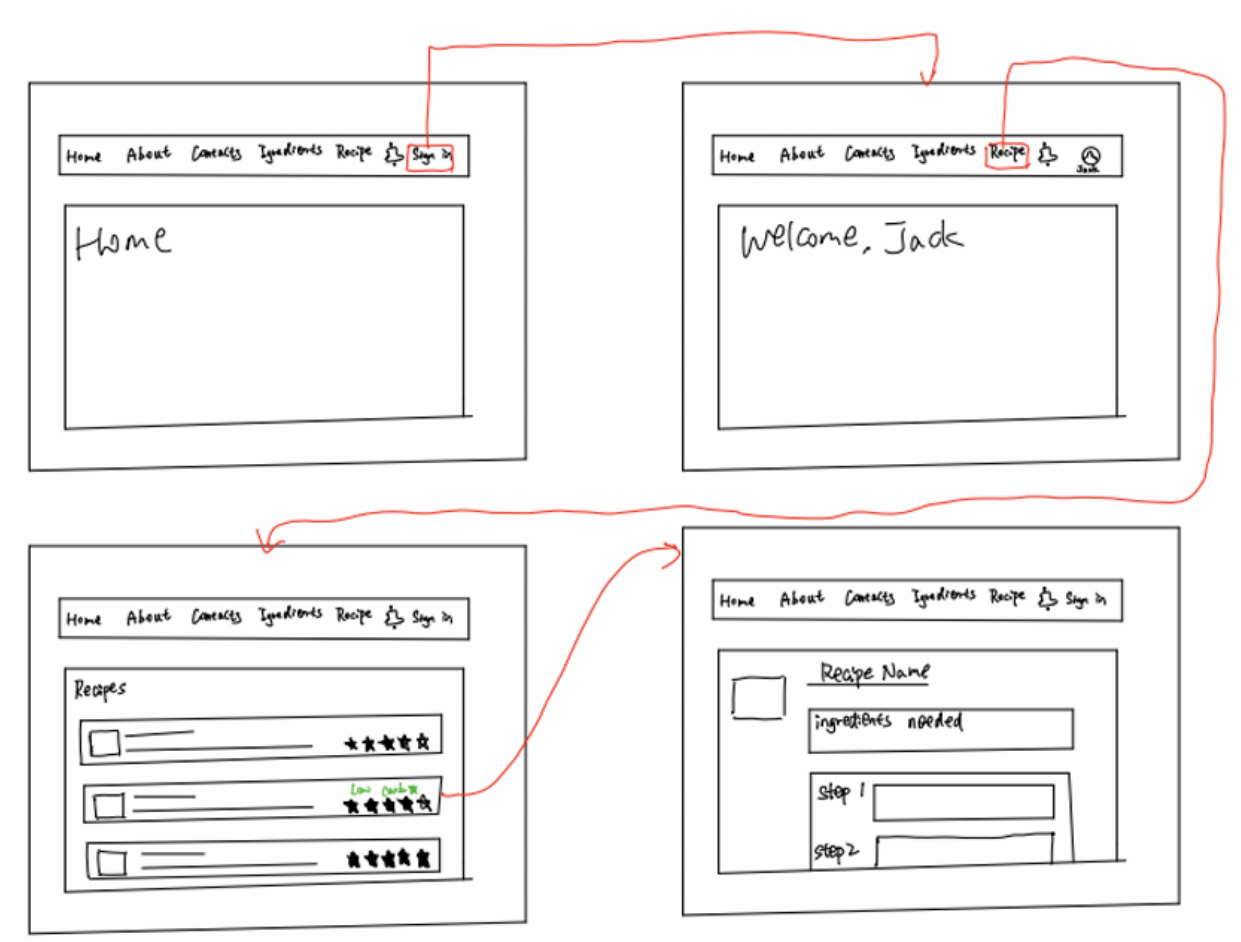
Use Case 3: Waste Reduction:

Julie then finds out about ScholarEats and accesses it on her phone. Julie is amazed with the variety of recipes that are available and how every ingredient from Gator Groceries is utilized in some way. Now, when she goes to Gator Groceries, she can get exactly the type and amount of ingredients she needs without having to worry about any of them going to waste. She also likes that there is an inventory feature so she does not have to worry about showing up to Gator Groceries and having to find out that everything is gone.



Use Case 4: Health Consciousness:

Jack logged in and checked available ingredients. He selected a low-carb recipe. The platform provided necessary ingredients and step-by-step cooking instructions. Jack picked up the ingredients from campus grocery hub and cooked his meal. This program helped him maintain a healthy diet.

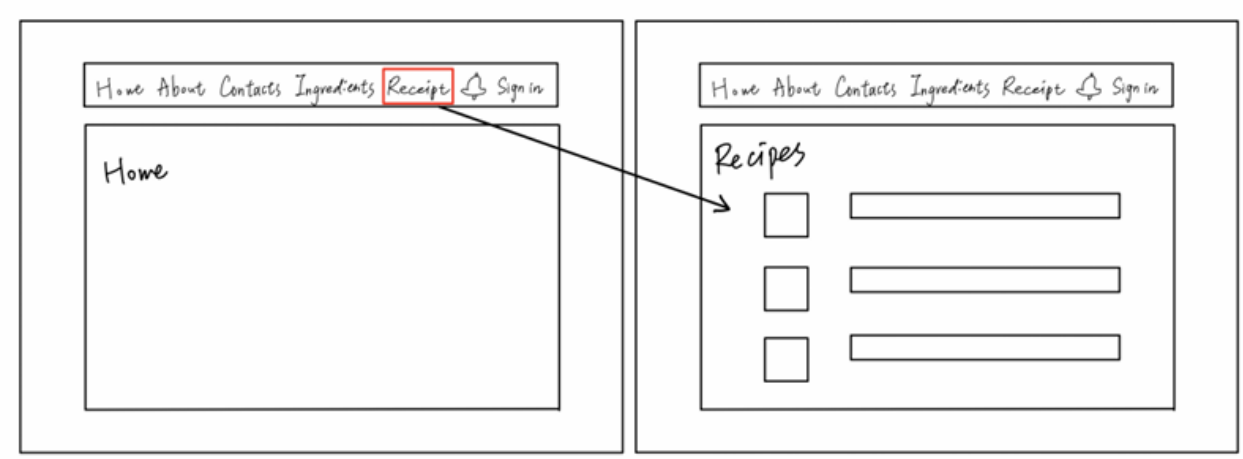


Use Case 5: Student Engagement:

Joe looks into his automatically-enrolled account, and opts in for the next week's groceries. After opting in himself, Joe posts about the program on social media, which catches their friend Dwight's attention; he is intrigued by the program, looks into it himself and also opts in for the groceries.

Use Case 6: Time Saving:

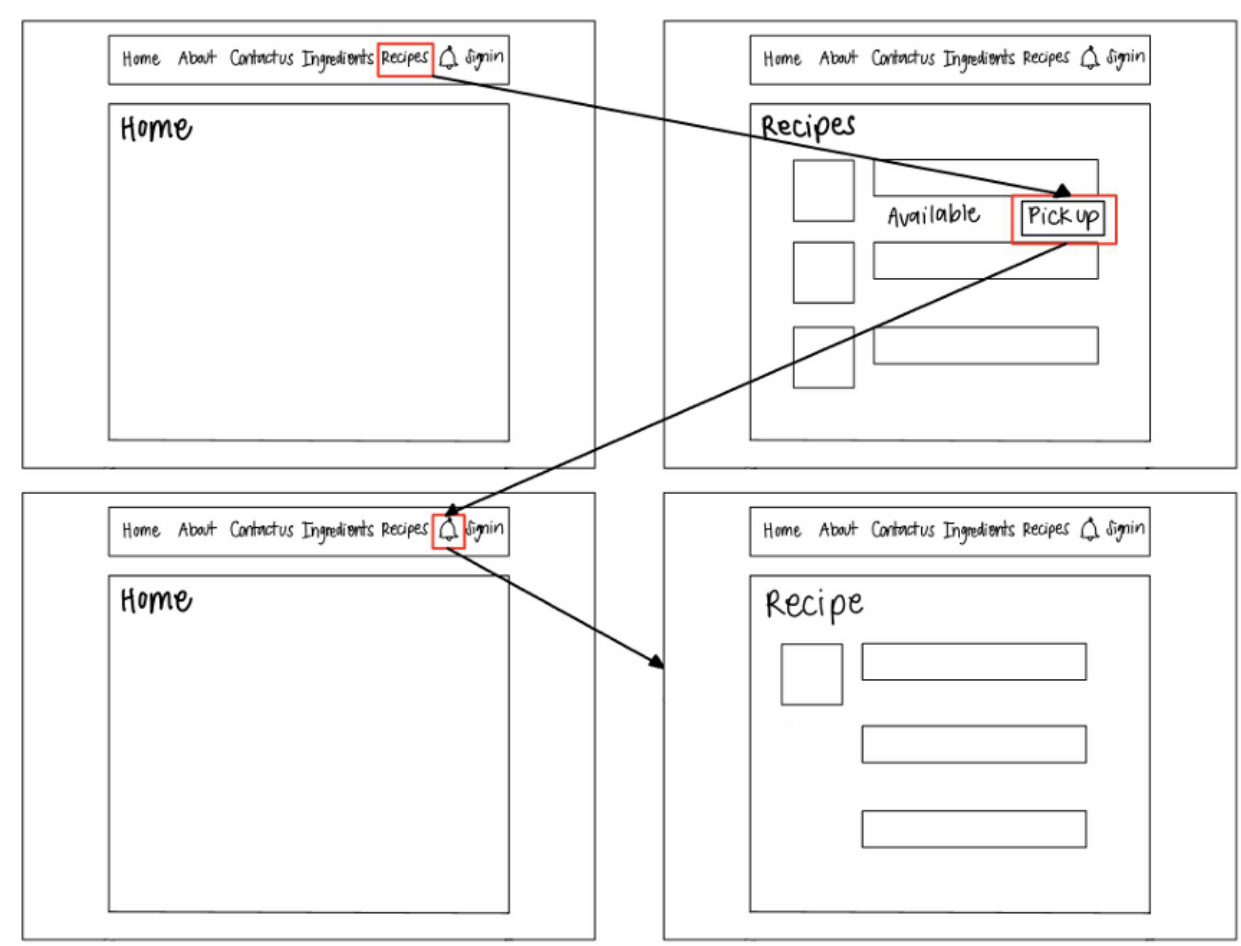
Jessie, a student who is currently attending full-time at her university and a part-time worker at a convenience store, has a hard time balancing her school and work life. Despite the many efforts that she has made to manage her time properly she always falls short with the time she has for herself. Luckily, she found that she can save more time with the use of ScholarEats where Jessie can easily get recipes based on the groceries that her university has. With this, she is able to be stress free and able to spend less time on finding what to make.



Use Case 7: Cooking Education:

Olivia can use this app that her school manages locally to see which recipes have ingredients readily available within the university's food pantry. Olivia confirms her intention of picking up a set of ingredients earlier in the day and gets notified

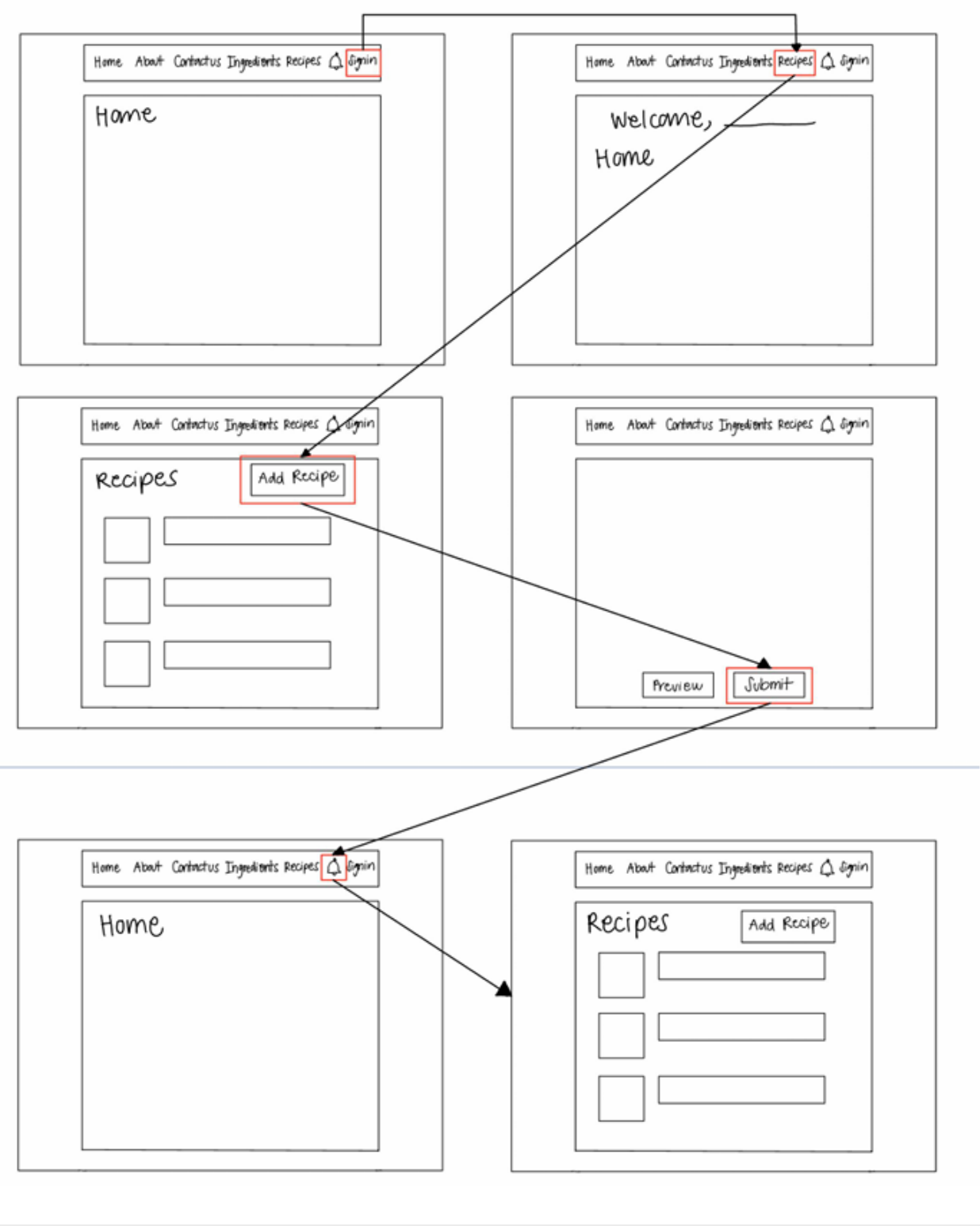
when it is ready to pick up. Olivia picks up the ingredients. Olivia goes to her dorm and the app gives her step by step instructions on how to cook.



Use Case 8: Community Recipe Submission:

Motivated by a desire to engage with her peers and enrich the campus dining experience, she logs into the ScholarEats platform, eager to introduce others to her culinary creation. The platform's user-friendly interface for the recipe submission process includes boxes for the ingredient list, step-by-step instructions, and a place to submit optional photos of the dish. It is well organized and easy to navigate. She appreciates the option to preview her entry before finalizing it. Upon submission, the ScholarEats system quickly processes the recipe, displaying a confirmation message that reassures Emily that her contribution has been received and is under review. Emily receives a notification

when her recipe is live, adding a sense of accomplishment and fostering a deeper sense of community involvement.



4. High level database architecture and organization

Database-Related Functional Requirements

1. ADMINISTRATORS SHALL BE ABLE TO EDIT THE INGREDIENT LIST
2. ADMINISTRATORS SHALL BE ABLE TO REMOVE USER ACCOUNTS
3. ADMINISTRATORS SHALL BE ABLE TO FOLLOW THE EXPIRATION DATE CLOSELY AND TAKE THE NECESSARY ACTIONS.
4. SYSTEM SHALL BE ABLE TO GENERATE RECOMMENDED RECIPES
5. SYSTEM SHALL TELL ADMINISTRATORS WHEN AN INGREDIENT HAS RUN OUT OF STOCK
6. SYSTEM SHALL TELL ADMINISTRATORS WHEN FOOD HAS SPOILED
7. USERS SHALL BE ABLE TO FILTER RECIPES BY DIETARY RESTRICTIONS
8. USERS SHALL BE ABLE TO SET ALLERGIES
9. USERS SHALL BE ABLE TO SET DIETARY RESTRICTIONS
10. USERS SHALL BE ABLE TO VIEW RECIPES

DBMS Definition:

The Database Management System used for this project is MySQL. MySQL is an open-source relational database management system known for its reliability, ease to use and performance. It is widely used for web type applications and supports SQL for managing the data.

Entity Descriptions:

1. **Users**: Contains user information such as email, username, password hash, and other personal details.
2. Recipes: Contains recipe details such as recipe name, prep time, cook time, ingredients and nutritional information.
3. Ingredients: Stores information about ingredients used in recipes.
4. Recipe_Ingredient: A join table that links recipes and ingredients.

5. Store: Manages the ingredients a user has in their store, including quantity and expiration date.
6. User_Recipes: Links users to their favorite or created recipes.
7. Activity_Log: Records user activities such as logins and specific actions within the application.
8. Notifications: Stores notifications sent to users.
9. Roles: Manages user roles and their permissions.
10. University: Contains information about universities, which users can be associated with.
11. Email_Log: Logs emails sent to users for activities like verification and notifications.
12. Manages: Links admin users to the users they manage.
13. Admin: Contains information about admin users.
14. Sessions: Tracks user sessions for login and activity tracking.
15. Expired_Products: Tracks products that have expired in the user's store.

Database Media Storage

For storing the media that the site will use we we use a simple system of having an “images” folder and having references to specific images for specific recipes point to a specific image within the “recipes” sub-folder. The same shall be done for ingredients.

This path will be returned from the database with the other recipe / ingredient information when they are queried to display to the page.

We chose this method because it allowed us to use images that could be unique to each item in the database while adding effectively no overhead as we were querying from the database already.

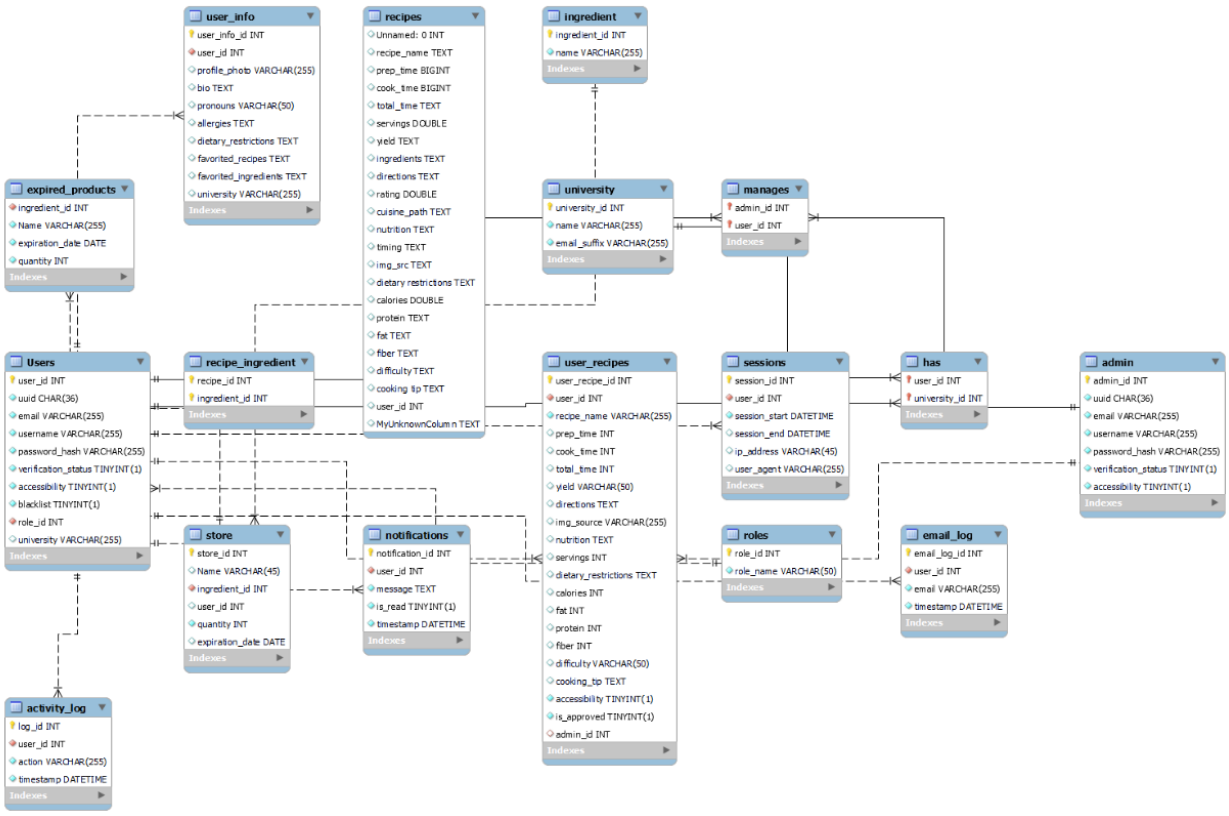
Additionally this lets us make an elegant solution where recipes could default to a placeholder image without the need for javascript to check if a value was null and then go fetch the image, therefore reducing load times.

Note that a file-size limitation will be imposed to avoid the possibility of malicious attacks and to prevent load-times of large sets of results from being too large.

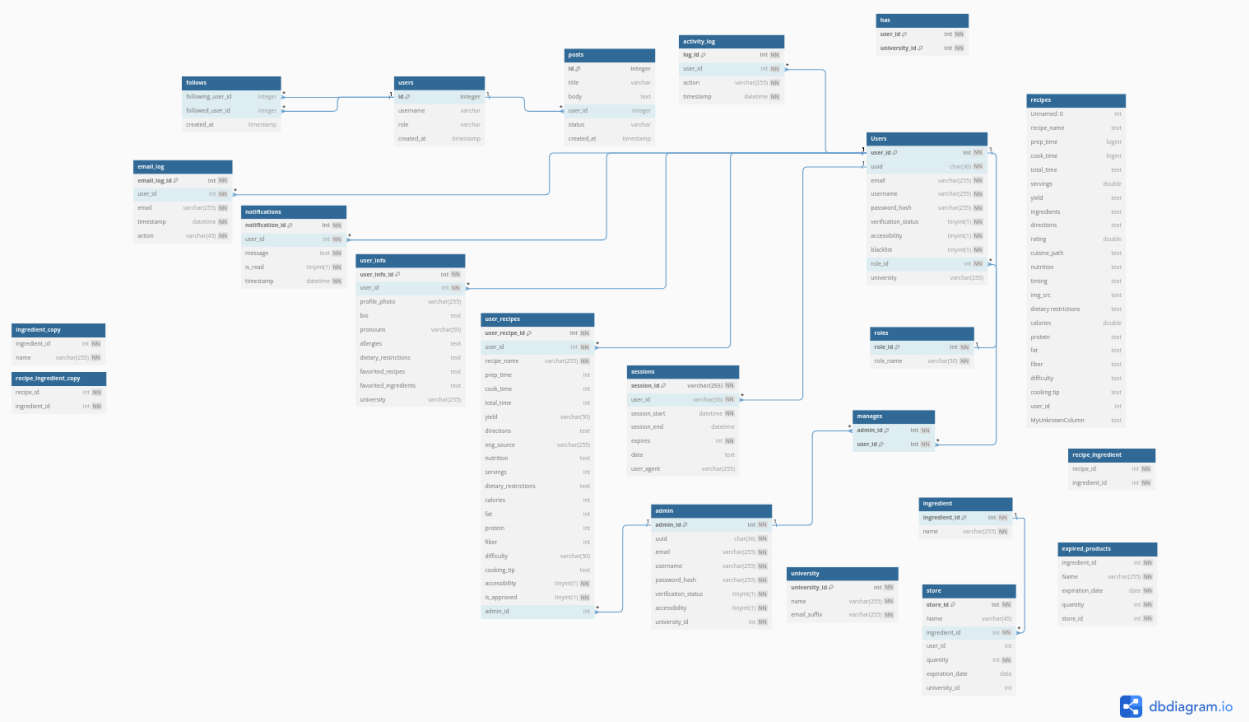
Folder Structure (within “public” directory):

- *Images Folder*
  - *Site Assets*
  - *User Profile Images*
  - *Recipe Images*
  - *Ingredient Images*

EER Diagram:



ERD Diagram:


follows
following_user_id integer
followed_user_id integer
created_at timestamp
users
id integer
username varchar
role varchar
created_at timestamp
posts
id integer
title varchar
body text
user_id integer
status varchar
created_at timestamp
activity_log
log_id int NN
user_id int NN
action varchar(255) NN
timestamp datetime NN
has
user_id int NN
university_id int NN
recipes
Unnamed: 0 int
recipe_name text
prep_time bigint
cook_time bigint
total_time text
servings double
yield text
ingredients text
directions text
rating double
cuisine_path text
nutrition text
timing text
img_src text
dietary_restrictions text
calories double
protein text
fat text
fiber text
difficulty text
cooking_tip text
user_id int
MyUnknownColumn text
email_log
email_log_id int NN
user_id int NN
email varchar(255) NN
timestamp datetime NN
action varchar(45) NN
notifications
notification_id int NN
user_id int NN
message text NN
is_read tinyint(1) NN
timestamp datetime NN
Users
user_id int NN
uuid char(36) NN
email varchar(255) NN
username varchar(255) NN
password_hash varchar(255) NN
verification_status tinyint(1) NN
accessibility tinyint(1) NN
blacklist tinyint(1) NN
role_id int NN
university varchar(255)
user_info
user_info_id int NN
user_id int NN
profile_photo varchar(255)
bio text
pronouns varchar(50)
allergies text
dietary_restrictions text
favorited_recipes text
favorited_ingredients text
university varchar(255)
ingredient_copy
ingredient_id int NN
name varchar(255) NN
recipe_ingredient_copy
recipe_id int NN
ingredient_id int NN
user_recipes
user_recipe_id int NN
user_id int NN
recipe_name varchar(255) NN
prep_time int
cook_time int
total_time int
yield varchar(50)
directions text
img_source varchar(255)
nutrition text
servings int
dietary_restrictions text
calories int
fat int
protein int
fiber int
difficulty varchar(50)
cooking_tip text
accessibility tinyint(1) NN
is_approved tinyint(1) NN
admin_id int
sessions
session_id varchar(255) NN
user_id varchar(30) NN
session_start datetime NN
session_end datetime
expires int NN
data text
user_agent varchar(255)
roles
role_id int NN
role_name varchar(30) NN
manages
admin_id int NN
user_id int NN
recipe_ingredient
recipe_id int NN
ingredient_id int NN
admin
admin_id int NN
uuid char(36) NN
email varchar(255) NN
username varchar(255) NN
password_hash varchar(255) NN
verification_status tinyint(1) NN
accessibility tinyint(1) NN
university_id int NN
ingredient
ingredient_id int NN
name varchar(255) NN
university
university_id int NN
name varchar(255) NN
email_suffix varchar(255) NN
store
store_id int NN
Name varchar(45)
ingredient_id int NN
user_id int
quantity int NN
expiration_date date
university_id int
expired_products
ingredient_id int NN
Name varchar(255) NN
expiration_date date NN
quantity int NN
store_id int NN
dbdiagram.io

5. High Level APIs and Main Algorithms

Sorting APIs:

- Endpoint: router.get('/sortByCaloriesAsc')
  - Returns recipes sorted in ascending order based on their calorie content.
- Endpoint: router.get('/sortByCaloriesDesc')
  - Returns recipes sorted in descending order based on their calorie content.
- Endpoint: router.get('/sortByProteinAsc')
  - Returns recipes sorted in ascending order based on their protein content.
- Endpoint: router.get('/sortByProteinDesc')
  - Returns recipes sorted in descending order based on their protein content.
- Endpoint: router.get('/sortByFatAsc')
  - Returns recipes sorted in ascending order based on their fat content.
- Endpoint: router.get('/sortByFatDesc')
  - Returns recipes sorted in descending order based on their fat content.
- Endpoint: router.get('/sortByFiberAsc')
  - Returns recipes sorted in ascending order based on their fiber content.
- Endpoint: router.get('/sortByFiberDesc')
  - Returns recipes sorted in descending order based on their fiber content.

Filtering APIs:

- Endpoint: router.get('/filterByDiet/:restriction')
  - Filters recipes based on a specific dietary restriction indicated by :restriction.
  - Returns recipes that match the specified dietary restriction.
- Endpoint: router.get('/filterByCookingAids/:aid')
  - Filters recipes based on specific cooking aids required indicated by :aid.
  - Returns recipes that require the specified cooking aid.

- Endpoint: router.get('/filterByDifficulty/:level')
    - Filters recipes based on difficulty level indicated by :level.
    - Returns recipes that match the specified difficulty level.

Fetch Available Ingredients API

- router.get('/')
    - Retrieves a list of available ingredients from the database.
    - It queries the store table to fetch ingredient_id, quantity, and name of each ingredient joined with the ingredient table.
    - It then renders an ingredients page with the retrieved data, displaying each ingredient with an associated image, name, and quantity.

Check Expired Items API

- Endpoint: router.get('/checkExpired')
    - Updates the expired status of food items in the store table based on their expiration_date.
    - It sets expired = TRUE for items where expiration_date is less than or equal to the current date (YYYY-MM-DD format).

Remove Out of Stock Items API

- Endpoint: router.get('/checkOutOfStock')
    - Deletes items from the store table where quantity_available is less than or equal to zero, indicating that the item is out of stock.

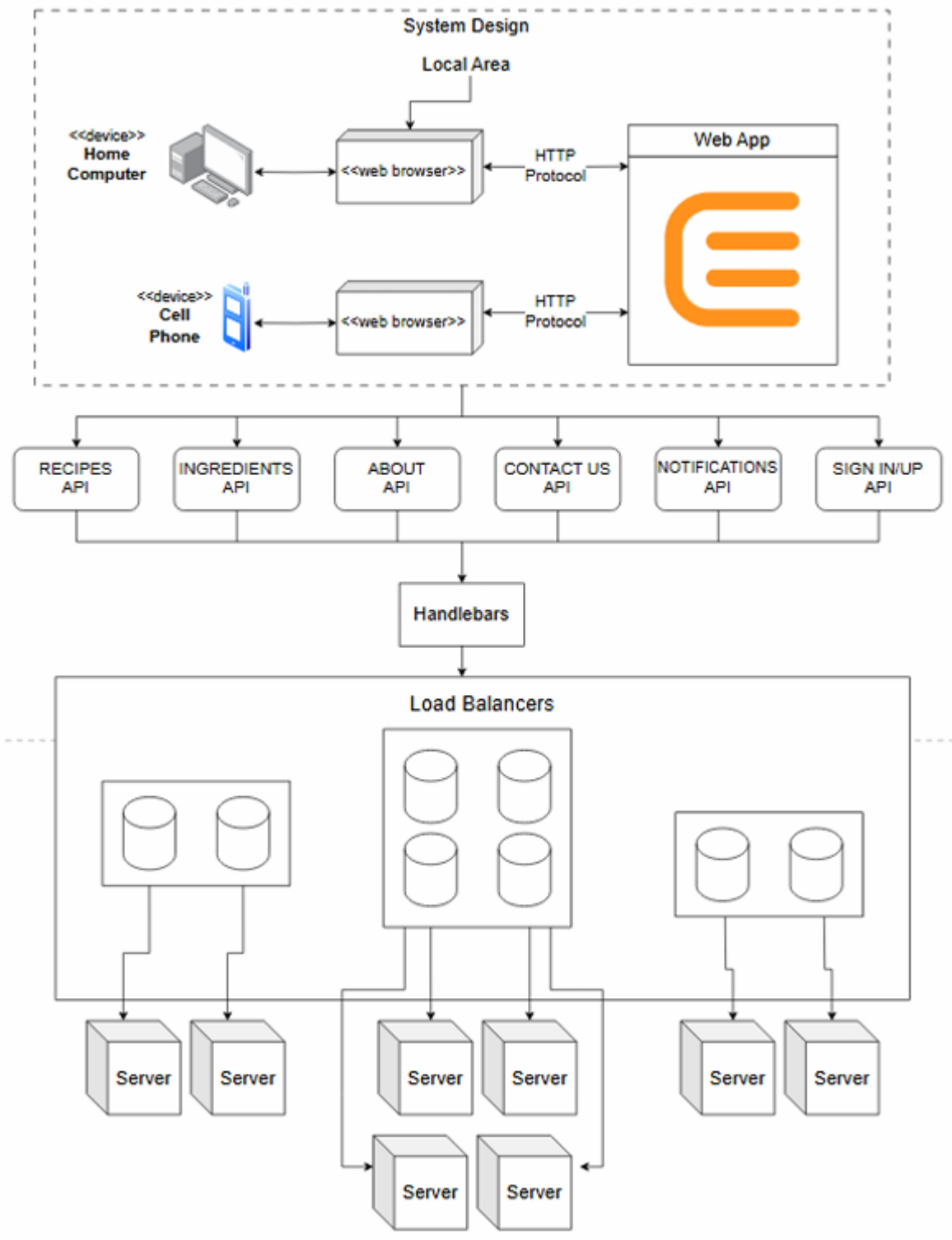
User APIs

- Endpoint: router.post(‘/register’)
    - Registers a new user their email and password and stores their credentials into the database.
- Endpoint: router.post(‘/login’)
    - Allows a user who created an account to log in with their email and password.
- Endpoint: router.post(‘/logout’)

    - Logs out a user that is currently logged in by terminating their session token.
- Endpoint: router.post(‘/change-password’)
    - Allows the user to change their password by giving the current password and then giving the new password.
- Endpoint: router.post(‘/change-username’)
    - Allows the user to change their username.
- Endpoint: router.post(‘/set-allergies)
    - Allows the user to set their allergies in their profile.
- Endpoint: router.post(‘/set-dietary-restrictions)
    - Allows the user to set their dietary restrictions in their profile.

6. System Design

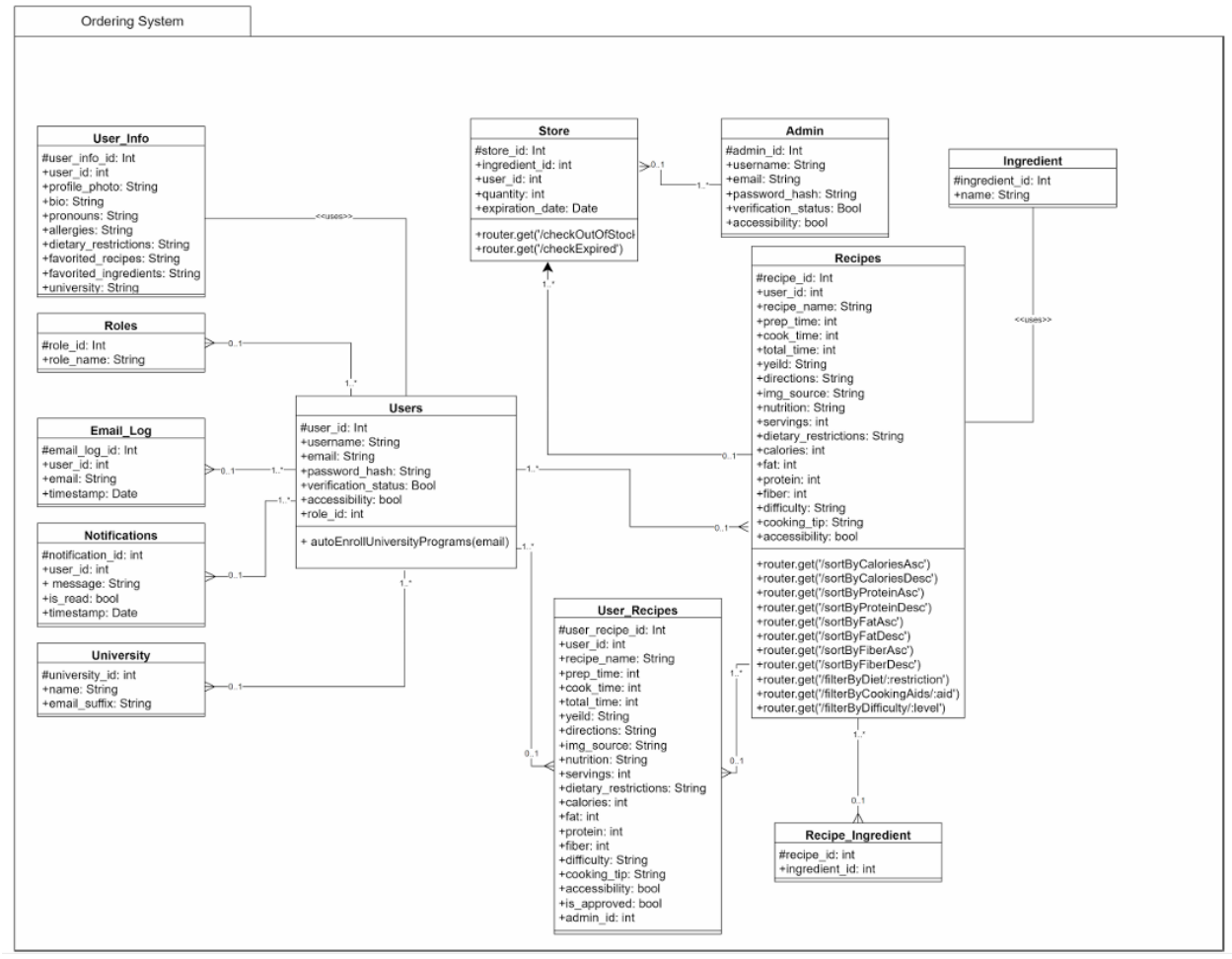
Scalability Illustration Diagram:



Summary of System Architecture Design:

Our system architecture involves every user utilizing either a computer or mobile device that has access to a web browser. That web browser accesses our web app, ScholarEats, over HTTP protocol. The web app interacts with several APIs, including the following tabs: Recipes, Ingredients, About, Contact Us, Notifications and Sign In/Up. The API requests are processed by Handlebars, which then sends the requests to the different load balancers. The load balancers distribute to whatever server is available.

UML Diagrams:



Summary of UML Diagram:

The main entities are ‘Users’, ‘User_Info’, ‘Email_Log’, ‘University’, ‘Store’, ‘Admin’, ‘Recipes’, and ‘Ingredient’.

The ‘User_Info’ class stores detailed user information, such as ‘profile photo’, ‘bio’, ‘pronouns’, ‘allergies’, ‘dietary restrictions’, ‘favorited ingredients’, ‘favorited recipes’, and the ‘university’ the user belongs to. It has a one-to-one association with the ‘Users’ class.

The ‘Users’ class contains basic user information, including ‘username’, ‘email’, ‘password hash’, ‘verification status’, and ‘role ID’. It also has a method to automatically enroll users in university programs based on their email domain. The decision to make the User and User_Info entities separate classes was made in order to simplify maintenance.

The Roles class assigns a role to each user, which is linked through the ‘role_id’. Having Roles as its own class will help with scalability if future roles are ever created.

The ‘University’ class contains details about different universities, including their ‘names’ and ‘email suffixes’, which are used for auto-enrollment. The ‘Admin’ class stores admin details such as ‘username’, ‘email’, ‘password hash’, ‘verification status’, and ‘accessibility’. By keeping these entities in separate classes, this design helps to maintain data integrity. Also, making the Admin entity separate from a regular User entity was chosen for security reasons.

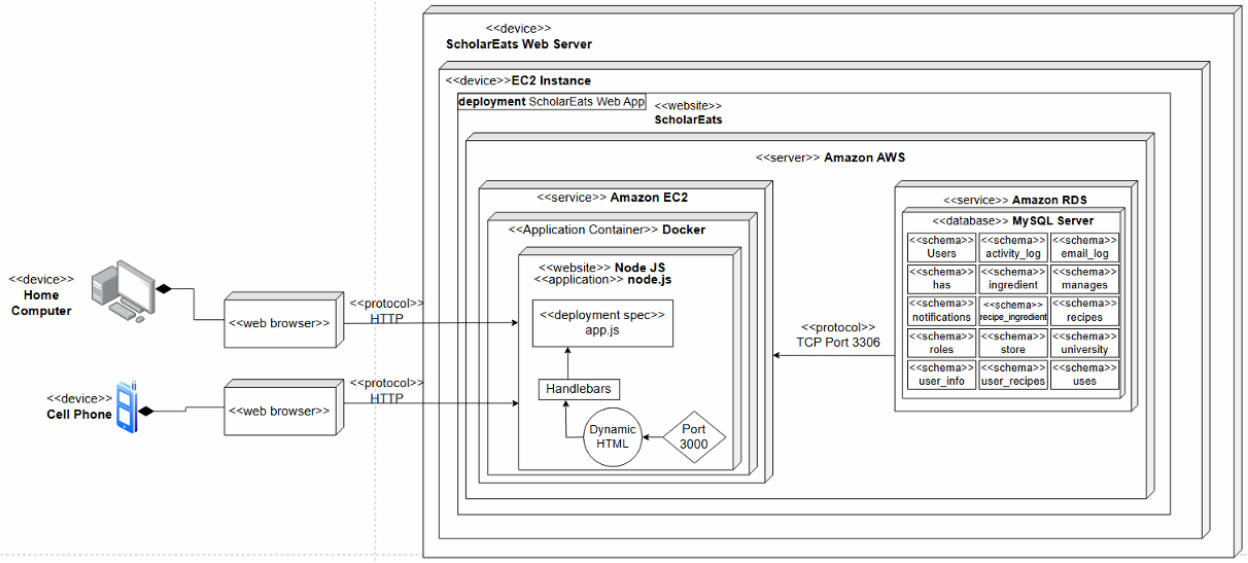
The ‘Store’ class manages inventory, including ‘ingredient IDs’, ‘quantities’, and ‘expiration dates’. It contains methods to check for out-of-stock items and expired ingredients and has a one-to-many relationship with the ‘Users’ class. This class is intended to help maintain the inventory in a centralized and consistent way.

The ‘Recipes’ class captures recipe details like ‘recipe name’, ‘preparation time’, ‘cook time’, ‘total time’, ‘yield’, ‘directions’, ‘image source’, ‘nutrition’, ‘servings’, ‘dietary restrictions’, ‘calories’, ‘fat’, ‘protein’, ‘fiber’, ‘difficulty’, ‘cooking tips’, and ‘accessibility’. It includes several methods for sorting and filtering recipes based on these attributes. The ‘User_Recipes’ class links users to their recipes, capturing details such as ‘servings’, ‘directions’, ‘ingredients’, and ‘admin approval status’. Having these various filters and sort options helps to optimize data retrieval.

The ‘Ingredient’ class lists possible ingredients, while the ‘Recipe_Ingredient’ class maps ingredients to recipes, forming a many-to-many relationship between ‘Recipes’ and ‘Ingredient’. By separating these classes, we can include information, like preparation instructions or quantity for a particular ingredient in a particular recipe without affecting the definition/function of the Recipe or

Ingredient as a whole.

7. High Level Application Network and Deployment Design



8. Identify actual key risks for your project at this time

- Skills Risks
  - Inexperience making website applications
    - For our website application it will be the first time that many of us will be building a website application from scratch. Previous experiences often had us building on top of a pre-existing project that had some critical dependencies pre-configured.
    - To remedy this, individuals who had in-depth knowledge of specific portions of the project will take brief sections of meetings in order to inform and train other members of the team in critical components, such as the usage of Handlebars and the functionality we could incorporate when it comes to designing our page rendering.
- Schedule Risks
  - Fast-Tempo of Summer
    - This class is a compressed class. It takes place over about two months as opposed to the usual three. This means individual milestones will be much closer together and thus features will have to be quickly implemented.
    - Moving forward tasks will be distributed and supervised through the use of Notion. By listing all requirements ahead of time and assigning them out with individuals in charge of them we can achieve the end product in future milestones.
  - Team Leader very heavy schedule
    - The team lead is taking a total of 15 units over the summer. This severely limits his ability to be present to all tasks being completed.
    - The team lead will have a lighter schedule following the completion of Milestone 2 as 6 of the 15 units will have concluded. This paired with the delegation of tasks through Notion and sitting with team leads to discuss requirements will lead to smoother workflows.
- Technical Risks:

    - Testing procedures have vague checklist requirements
        - Our designed workflow incorporates a final step of testing procedures before a Pull Request (PR) in GitHub could be created. Due to the nature of the project being in early development, these testing checklists are vague and subject to change.
        - To remedy this we introduced a new Git branch called “TestingGuidance” which will be utilized to pass out updated checklists to the master branch as the prototype becomes more developed. This will enable everyone to be able to pull the most recent guidance directly within their branches to minimize confusion.
- Teamwork Risks:
    - Code Review must be tested before-hand individually and then reviewed by 1 of 2 people
        - Our designed workflow’s finalization process starts with the maintainer going through the current guidance and ensuring that nothing is immediately broken by their changes. The maintainer then makes a Pull Request (PR) which is approved by either the Team-Lead of the GitHub Master. This process is simultaneously with several branches.
        - To ensure that features are developed and incorporated into the master branch Notion will be used to track the status of each feature. Once the maintainer has finished the branch and believes it to be ready the Notion task will be updated and the Discord team chat will be notified. This will allow all maintainers to be able to see where teammates are at and when they should pull from main to account for changes.
- Legal/Content Risks:
    - Copyrighted content: We have to ensure we have the legal right to use any images on the site
        - This application will be used potentially commercially or as an enterprise product. This means that we have to ensure that all images, resources, and materials used in the creation or operation of ScholarEats either belong to us or we have the legal ability to utilize

them. Failure to do so opens up the opportunity for our organization to face legal consequences.

- We are preemptively avoiding this issue by instituting a policy where we make our own materials or only introduce outside materials with the copyright information alongside it. For an example Institution we are avoiding the usage of the SFSU banner in the footer and will be incorporating a made-up university as an example.

9. Project management

Scholar Eats' development will be managed through the workflow-tracking site "Notion". Tasks will be identified by the Team-Lead and after being defined will be added to the Tasks Section of the Notion Teamspace. At this time the maintainer responsible for that task will be defined and that maintainer will be in charge of implementing the said task. Notion tasks will have statuses updated by the maintainer with where progress stops so that if another maintainer is assigned, they know where the task currently stands. The maintainer will fully and completely implement the task and once the task is believed to have been completed they will examine the checklist proved in the testing guidance. Once that checklist has been examined and verified, they are able to submit a GitHub Pull Request (PR). Once the PR has been made, either the GitHub Master or the Team-Lead will review the PR and either approve or close it. Once approved the PR can be merged into the master branch. At no points will the master branch ever be committed to or altered in any way outside of the procedure above. Once the PR has been successfully merged the "task" is considered complete and will be updated accordingly on Notion.

10. Detailed list of contributions (this section must be done by the team lead)

- Donovan: (Score 9.5/10)
  - Led session to progress Section 3 of Milestone 2
  - Contributed to UI Mockups
  - Incorporated the notification message notifying the user of unimplemented features in the UI
  - Created the Layout to utilize Handlebars for dynamic HTML generation
  - Led a training session to inform others of the functionality and usage of Handlebars
  - Oversaw the creation of all UI pages
- Hancun: (Score 7/10)
  - Contributed to UI Mockups
  - Created the Login Page
  - Added the Forgot Password functionality
  - Added Input Validation
  - Added Remember Me Option
- Tina: (Score 7/10)
  - Contributed to UI Mockups
  - Created the .hbs for the Ingredients Page
  - Created the .hbs for the Recipes Page
  - Created the .hbs for the User Account Page
  - Furthered Account Management
- Edward: (Score 10/10)
  - Led session to tackle Section 4 of Milestone 2
  - Contributed to UI Mockups

    - Created some helper queries to access data from the database
    - Utilized the database queries and APIs to properly pass and render the pages through the defined .hbs files
    - Worked with the defined search parameters to properly build the SQL Query to populate the database.
- Karl: (Score 7/10)
    - Contributed to the High-Level Database Design
    - Implemented the creation of users
    - Implemented the ability to sign in as a user
    - Implemented the ability to change a user’s username
    - Implemented the ability to change a user’s passwords
    - Implemented the ability to set a user’s dietary restrictions
    - Implemented the ability to set a user’s allergies
- Sai: (Score 7/10)
    - Contributed to the High-Level Database Design
    - Wrote a script that handled a majority of the auto-complete requirements for the Minimum Viable Prototype
    - Created some helper queries to access data from the database
    - Helped configure the UI to contain the autocomplete functionality.
    - Assisted with Documentation by completing EER and ERD Diagrams
- Maeve: (Score 9.5/10)
    - Led session to tackle Section 6 of Milestone 2
    - Contributed to UI Mockups
    - Finalized Section 6
    - Finalized Section 7
    - Assisted with the development of frontend.

    - Helped ensure that every team member submitted their submission for Milestone 2
- Sabrina: (Score 9/10)
    - Tested and Reviewed over a dozen individual Pull Requests on GitHub.
    - Contributed to UI Mockups
    - Contributed to the documentation involving the System Design
    - Created and completed the requirements for the page footer of the Minimum Viable Prototype
    - Maintained and updated all relevant credentials

SW Engineering CSC648/848 Summer 2024

ScholarEats

Team 4:

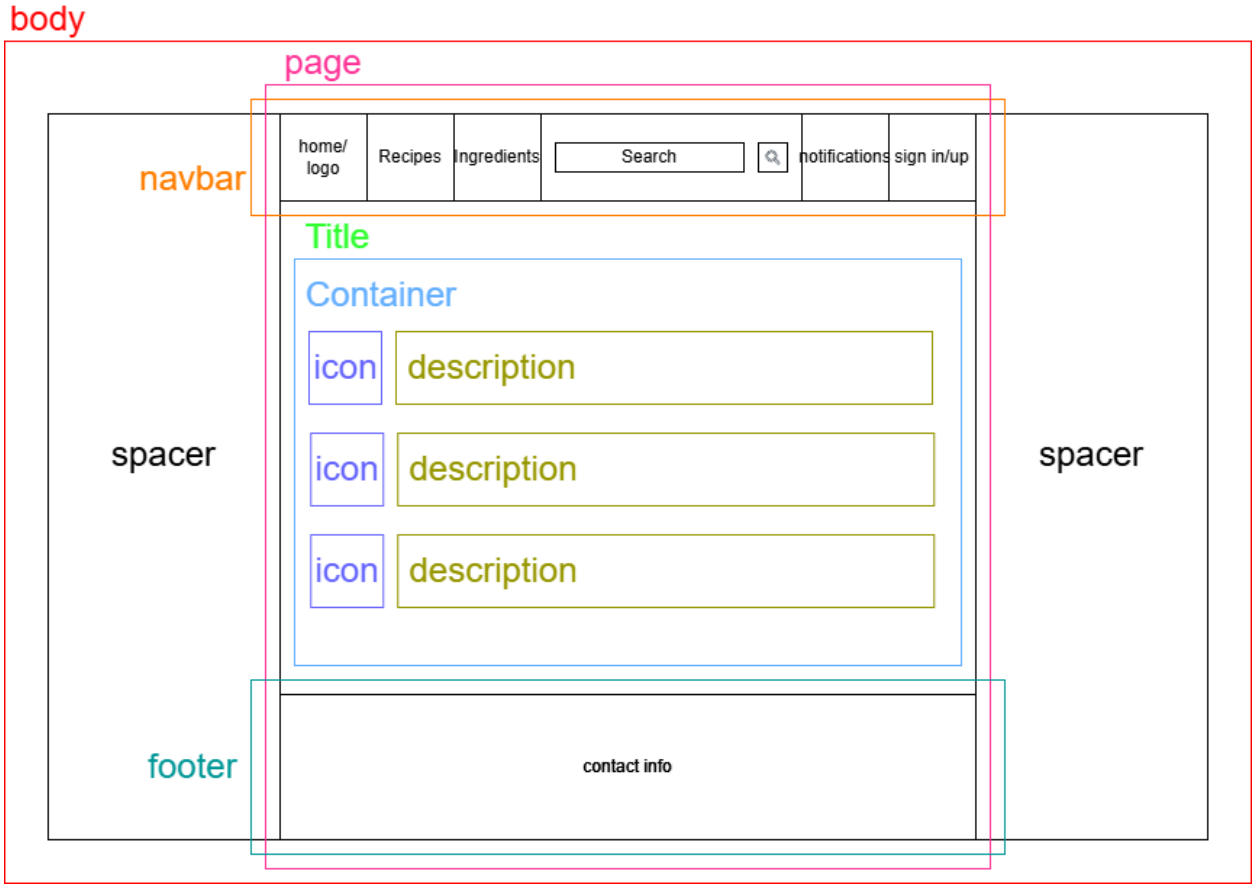
Angelo Arriaga (Team Lead - aarriaga1@sfsu.edu)
Donovan Taylor (Front-End Lead)
Hancun Guo (Front-End)
Tina Chou (Front-End)
Edward McDonald (Back-End Lead)
Karl Carsola (Back-End)
Sai Bavisetti (Database)
Maeve Fitzpatrick (Docs Editor)
Sabrina Diaz-Erazo (GitHub Master)

Milestone 3

07/23/24

History (Revisions)	
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. Data Definitions



Database:

activity_log	log_id
	user_id
	action
	timestamp

admin	admin_id
	uuid
	email
	username
	password_hash
	verification_status

	accessibility

email_log	email_log_id
	user_id
	email
	timestamp
	action

expired_products	ingredient_id
	Name
	expiration_date
	quantity
	store_id

has	user_id
	university_id

ingredient	ingredient_id
	name
	img_src

ingredient_copy	ingredient_id
	name

manages	admin_id
	user_id

notifications	notification_id

	user_id
	message
	is_read
	timestamp

recipe_ingredient	recipe_id
	ingredient_id

recipe_ingredient_copy	recipe_id
	ingredient_id

recipes	Unnamed: 0
	recipe_name
	prep_time
	cook_time
	total_time
	servings
	yield
	ingredients
	directions
	rating
	cuisine_path
	nutrition
	timing
	img_src
	dietary restrictions
	calories

	protein
	fat
	fiber
	difficulty
	cooking tip
	user_id
	MyUnknownColumn

roles	role_id
	role_name

sessions	session_id
	user_id
	session_start
	session_end
	expires
	data
	user_agent

store	store_id
	Name
	ingredient_id
	user_id
	quantity
	expiration_date
	university_id

university	university_id
	name
	email_suffix

user_info	user_info_id
	user_id
	profile_photo
	bio
	pronouns
	allergies
	dietary_restrictions
	favorited_recipes
	favorited_ingredients
	university
	modes

user_recipes	user_recipe_id
	user_id
	recipe_name
	prep_time
	cook_time
	total_time
	yield
	directions
	img_source
	nutrition

	servings
	dietary_restrictions
	calories
	fat
	protein
	fiber
	difficulty
	cooking_tip
	accessibility
	is_approved
	admin_id

Users	user_id
	uuid
	email
	username
	password_hash
	verification_status
	accessibility
	blacklist
	role_id
	university

2. Functional Requirements

PRIORITY 1 - Critical

- Administrators
    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.
- System
    1. SYSTEM SHALL AUTOMATICALLY ENROLL USERS IN CORRESPONDING UNIVERSITY FOOD PROGRAMS
    2. SYSTEM SHALL BE ABLE TO GENERATE RECOMMENDED RECIPES
    3. SYSTEM SHALL TELL ADMINISTRATORS WHEN AN INGREDIENT HAS RUN OUT OF STOCK
    4. SYSTEM SHALL TELL ADMINISTRATORS WHEN FOOD HAS SPOILED
- Users
    1. USERS SHALL BE ABLE TO CHANGE THEIR PASSWORD
    2. USERS SHALL BE ABLE TO EDIT THEIR USERNAME
    3. USERS SHALL BE ABLE TO FILTER RECIPES BY COOKING AIDS REQUIRED
    4. USERS SHALL BE ABLE TO FILTER RECIPES BY DIETARY RESTRICTIONS
    5. USERS SHALL BE ABLE TO FILTER RECIPES BY DIFFICULTY
    6. USERS SHALL BE ABLE TO FILTER RECIPES BY INGREDIENTS
    7. USERS SHALL BE ABLE TO MAKE ACCOUNTS WITH VALID UNIVERSITY EMAIL.
    8. USERS SHALL BE ABLE TO SET ALLERGIES
    9. USERS SHALL BE ABLE TO SET DIETARY RESTRICTIONS
    10. USERS SHALL BE ABLE TO SORT RECIPES BY CALORIES
    11. USERS SHALL BE ABLE TO SORT RECIPES BY FAT
    12. USERS SHALL BE ABLE TO SORT RECIPES BY FIBER
    13. USERS SHALL BE ABLE TO SORT RECIPES BY PROTEIN
    14. USERS SHALL BE ABLE TO VIEW AVAILABLE INGREDIENTS
    15. USERS SHALL BE ABLE TO VIEW RECIPES

PRIORITY 2 - Desired

- Administrators
    1. ADMINISTRATORS SHALL BE ABLE TO REMOVE RECIPES OR REVIEWS
    2. ADMINISTRATORS SHALL BE ABLE TO MAKE UNIVERSITY-WIDE ANNOUNCEMENTS
- Systems
    1. SYSTEM SHALL ENSURE RECIPES ARE AVAILABLE FOR ALL (MAJOR) DIETARY RESTRICTIONS

- Users
    1. USERS SHALL BE ABLE TO FAVORITE INGREDIENTS
    2. USERS SHALL BE ABLE TO FAVORITE RECIPES
    3. USERS SHALL BE ABLE TO REVIEW/RATE RECIPES
    4. USERS SHALL BE ABLE TO SORT RECIPES BY RATING
    5. USERS SHALL BE ABLE TO VIEW OWN FAVORITE RECIPES
    6. USERS SHALL BE ABLE TO ADD THEIR PREFERRED PRONOUNS
    7. USERS SHALL BE ABLE TO ALLOW NOTIFICATIONS
    8. USERS SHALL BE ABLE TO EDIT PROFILE BIO
    9. USERS SHALL BE ABLE TO SHARE RECIPES
    10. USERS SHALL BE ABLE TO SWITCH BETWEEN LIGHT AND DARK MODE
    11. USERS SHALL BE ABLE TO VIEW HISTORY OF RECEIVED RECIPES

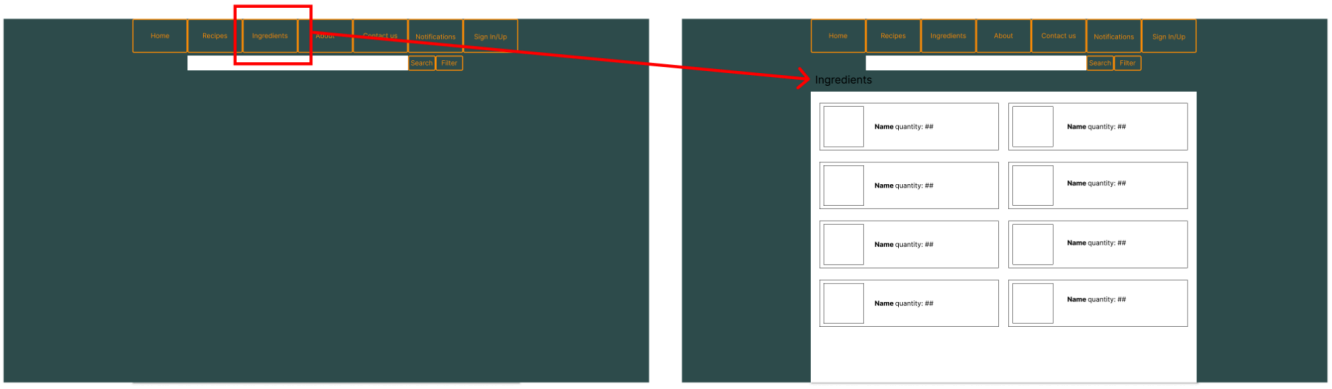
PRIORITY 3 - Opportunistic

- Administrators
    1. ADMINISTRATORS SHALL BE ABLE TO DELETE USER POSTED RECIPES
    2. ADMINISTRATORS SHALL BE ABLE TO PROMOTE RECIPES
    3. ADMINISTRATORS SHALL BE ABLE TO VIEW ALL USER-REPORTS
- Users
    1. USERS SHALL BE ABLE TO BLOCK OTHER USERS
    2. USERS SHALL BE ABLE TO CHANGE PROFILE PHOTO
    3. USERS SHALL BE ABLE TO REPORT PHOTOS OR REVIEWS
    4. USERS SHALL BE ABLE TO REPORT USER ACCOUNTS
    5. USERS SHALL BE ABLE TO SUBMIT RECIPES
    6. USERS SHALL BE ABLE TO TRANSFER ENROLLMENT IN UNIVERSITY FOOD PROGRAMS
    7. USERS SHALL BE ABLE TO UPLOAD PHOTOS OF THE RECIPES THEY COOK FOR REVIEWS
    8. USERS SHALL BE ABLE TO VIEW OTHER USER’S FAVORITE INGREDIENTS
    9. USERS SHALL BE ABLE TO VIEW OTHER USER’S FAVORITE RECIPES

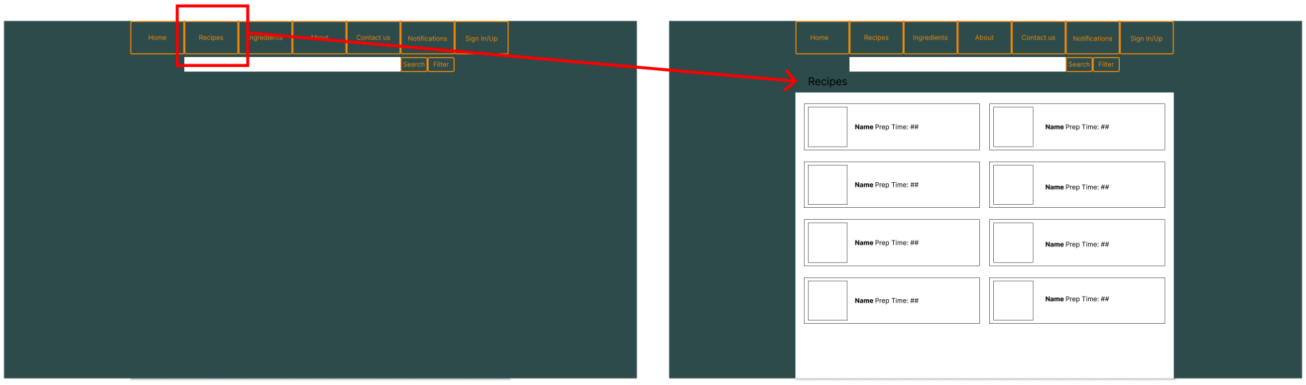
3. Wireframes Based on your Mockups/Storyboards (detailed)

[Figma Link]

1. Meal Planning
    a. Since the student is already using his university's grocery program, all he needs to do is go online and look at the recipes that are provided for his school's inventory. Once he signs up, he can browse a myriad of recipes using ingredients supplied by Gator Groceries.

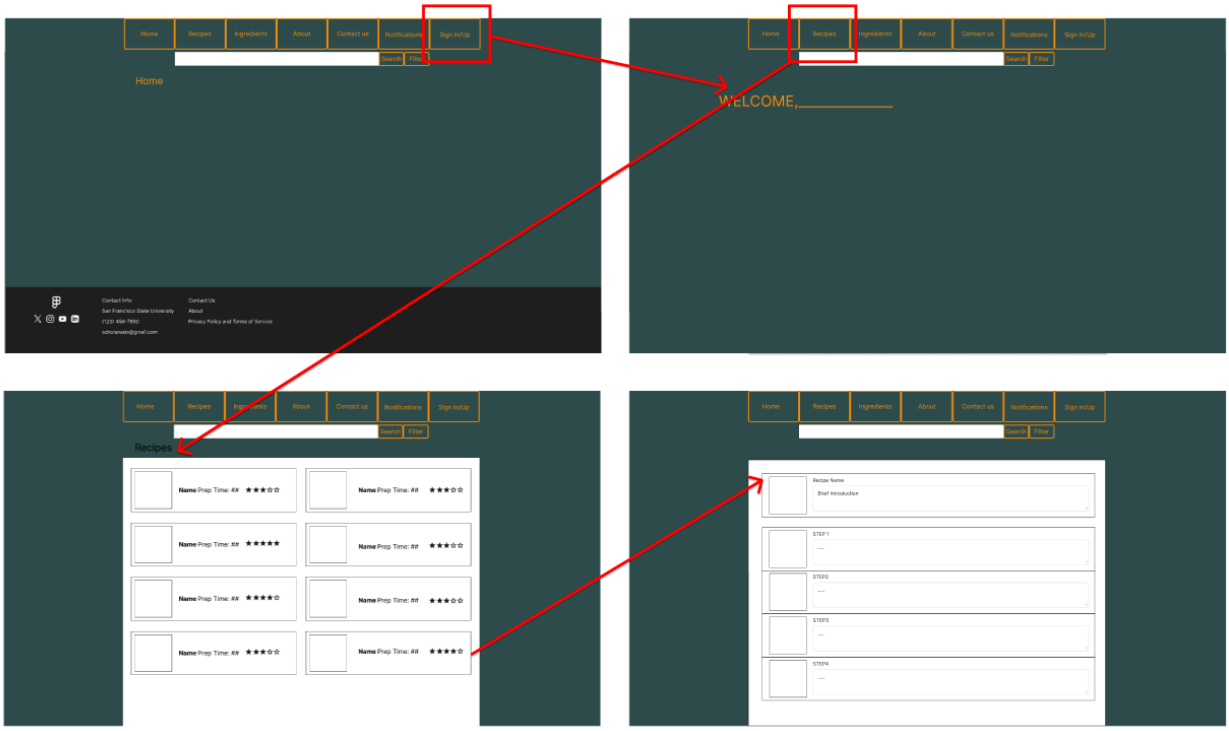


2. Money Saving
    a. Jane utilizes an online service (ScholarEats), which shows her the food available this week at her school's free food program. Along with this information, she also receives a few auto-generated recipes. This further encourages her to go to her university's free food program.



3. Waste Reduction
    a. Julie then finds out about ScholarEats and accesses it on her phone. Julie is amazed with the variety of recipes that are available and how every ingredient from Gator Groceries is utilized in some way. Now, when she goes to Gator Groceries, she can get exactly the type and amount of ingredients she needs without having to worry about any of them going to waste. She also likes that there is an inventory feature so she does not have to worry about showing up to Gator Groceries and having to find out that everything is gone.

4. Health Consciousness
    a. Jack logged in and checked available ingredients. He selected a low-carb recipe. The platform provided necessary ingredients and step-by-step cooking instructions. Jack picked up the ingredients from campus grocery hub and cooked his meal. This program helped him maintain a healthy diet.



5. Student Engagement
    a. Joe looks into his automatically-enrolled account, and opts in for the next week's groceries. After opting in himself, Joe posts about the program on social media, which catches their friend Dwight's attention; he is intrigued by the program, looks into it himself and also opts in for the groceries.

6. Time Saving
    a. Jessie, a student who is currently attending full-time at her university and a part-time worker at a convenience store, has a hard time balancing her school and work life. Despite the many efforts that she has made to manage her time properly she always falls short with the time she has for herself. Luckily, she found that she can save more time with the use of ScholarEats where Jessie can easily get recipes based on the groceries that her university has. With this, she is able to be stress free and able to spend less time on finding what to make.

7. Cooking Education
    a. Olivia can use this app that her school manages locally to see which recipes have ingredients readily available within the university's food pantry. Olivia confirms her intention of picking up a set of ingredients earlier in the day and gets notified when it is ready to pick up. Olivia picks up the ingredients. Olivia goes to her dorm and the app gives her step by step instructions on how to cook.

8. Community Recipe Submission
    a. Motivated by a desire to engage with her peers and enrich the campus dining experience, she logs into the ScholarEats platform, eager to introduce others to her culinary creation. The platform's user-friendly interface for the recipe submission process includes boxes for the ingredient list, step-by-step instructions, and a place to submit optional photos of the dish. It is well organized and easy to navigate. She appreciates the option to preview her entry before finalizing it. Upon submission, the ScholarEats system quickly processes the recipe, displaying a confirmation message that reassures Emily that her contribution has been received and is under review. Emily receives a notification when her recipe is live, adding a sense of accomplishment and fostering a deeper sense of community involvement.

We are not utilizing Use Case 8 anymore because it takes away from the main purpose of the application, which is compiling recipes that utilize the available ingredients that the university is offering. Allowing recipe submissions from students would introduce recipes that may require a whole lot more ingredients than what the user has and/or can afford.

4. High level database Architecture and organization (detailed)

Database-Related Functional Requirements:

1. ADMINISTRATORS SHALL BE ABLE TO EDIT THE INGREDIENT LIST
2. ADMINISTRATORS SHALL BE ABLE TO REMOVE USER ACCOUNTS
3. ADMINISTRATORS SHALL BE ABLE TO FOLLOW THE EXPIRATION DATE CLOSELY AND TAKE THE NECESSARY ACTIONS.
4. SYSTEM SHALL BE ABLE TO GENERATE RECOMMENDED RECIPES
5. SYSTEM SHALL TELL ADMINISTRATORS WHEN AN INGREDIENT HAS RUN OUT OF STOCK
6. SYSTEM SHALL TELL ADMINISTRATORS WHEN FOOD HAS SPOILED
7. USERS SHALL BE ABLE TO FILTER RECIPES BY DIETARY RESTRICTIONS
8. USERS SHALL BE ABLE TO SET ALLERGIES
9. USERS SHALL BE ABLE TO SET DIETARY RESTRICTIONS
10. USERS SHALL BE ABLE TO VIEW RECIPES

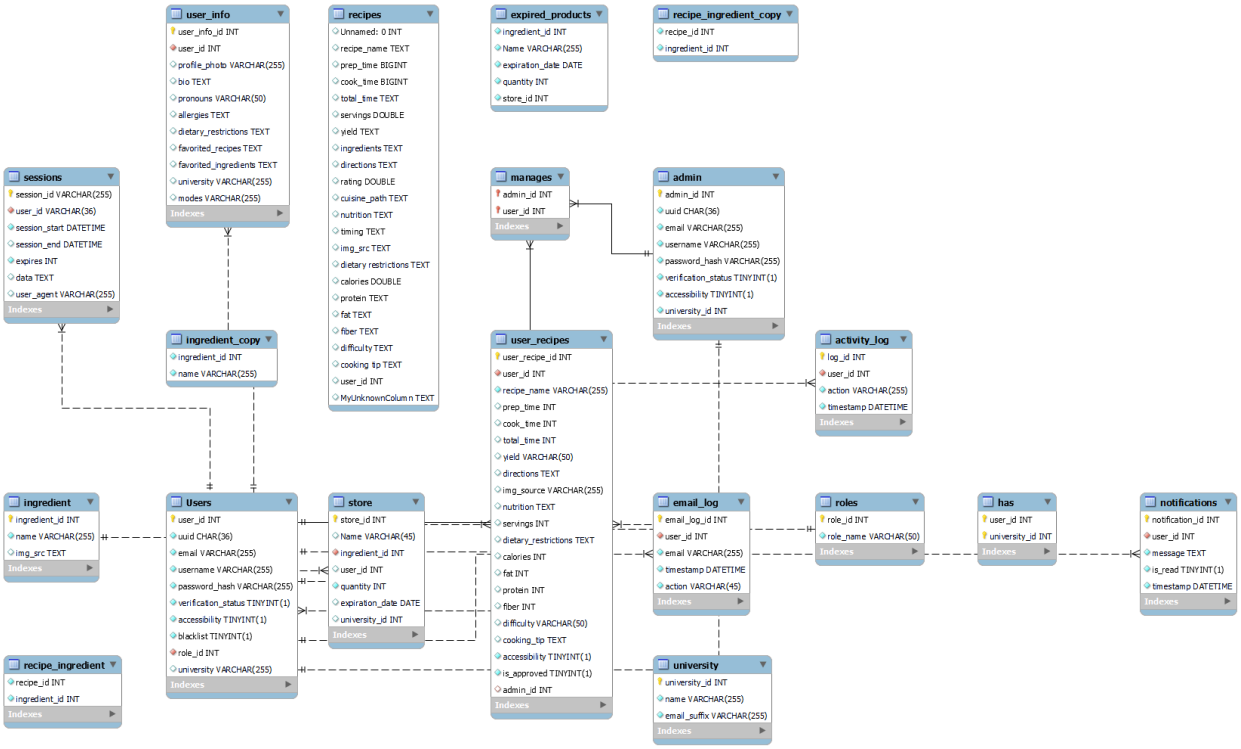
DBMS Definition:

The Database Management System used for this project is MySQL. MySQL is an open-source relational database management system known for its reliability, ease to use and performance. It is widely used for web type applications and supports SQL for managing the data.

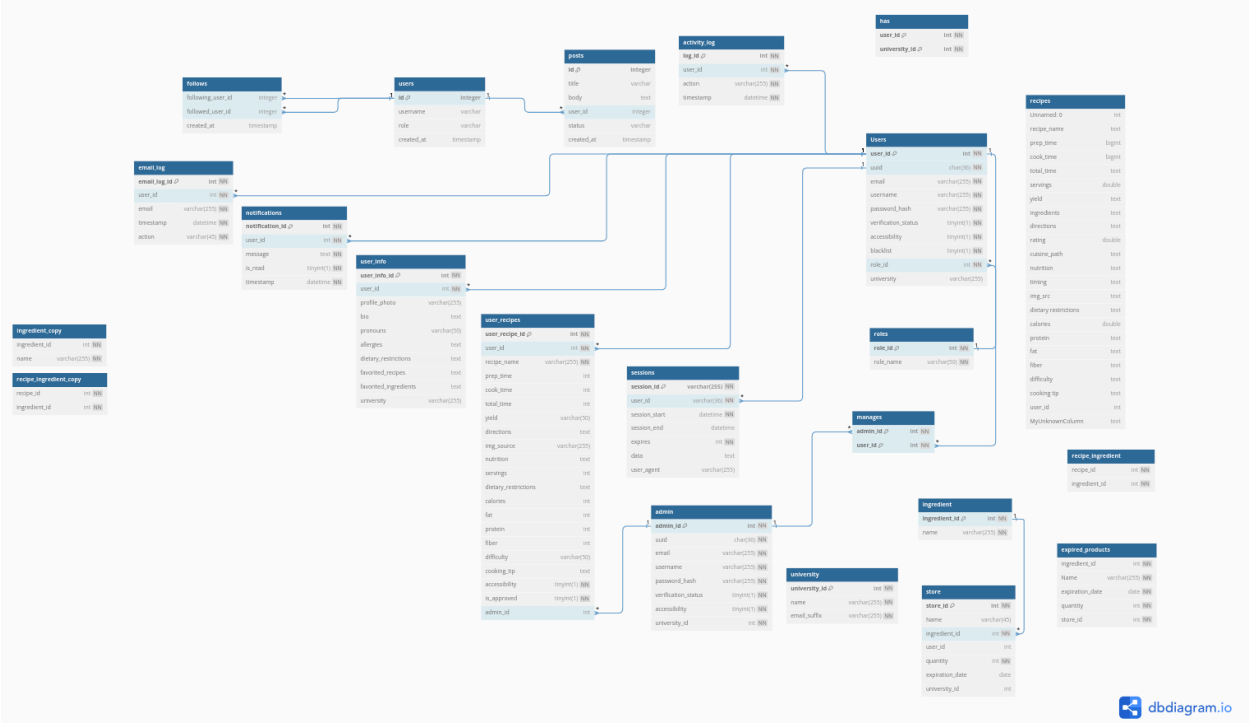
Entity Descriptions:

1. Users: Contains user information such as email, username, password hash, and other personal details.
2. Recipes: Contains recipe details such as recipe name, prep time, cook time, ingredients and nutritional information.
3. Ingredients: Stores information about ingredients used in recipes.
4. Recipe_Ingredient: A join table that links recipes and ingredients.
5. Store: Manages the ingredients a user has in their store, including quantity and expiration date.
6. User_Recipes: Links users to their favorite or created recipes.
7. Activity_Log: Records user activities such as logins and specific actions within the application.
8. Notifications: Stores notifications sent to users.
9. Roles: Manages user roles and their permissions.
10. University: Contains information about universities, which users can be associated with.
11. Email_Log: Logs emails sent to users for activities like verification and notifications.
12. Manages: Links admin users to the users they manage.
13. Admin: Contains information about admin users.
14. Sessions: Tracks user sessions for login and activity tracking.
15. Expired_Products: Tracks products that have expired in the user's store.

EER Diagram:

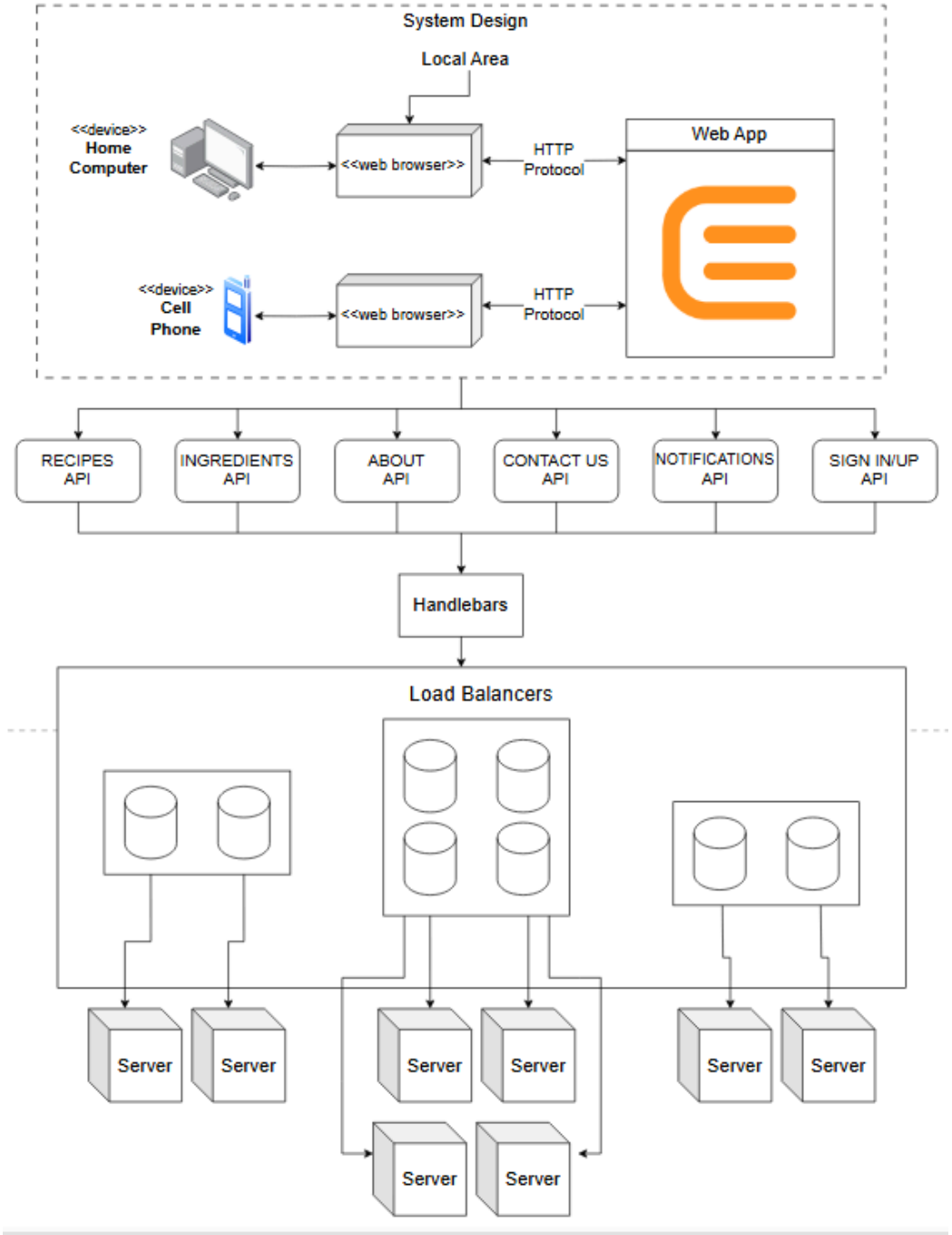

user_info
user_info_id INT
user_id INT
profile_photo VARCHAR(255)
bio TEXT
pronouns VARCHAR(50)
allergies TEXT
dietary_restrictions TEXT
favorited_recipes TEXT
favorited_ingredients TEXT
university VARCHAR(255)
modes VARCHAR(255)
Indexes
recipes
Unnamed: 0 INT
recipe_name TEXT
prep_time BIGINT
cook_time BIGINT
total_time TEXT
servings DOUBLE
yield TEXT
ingredients TEXT
directions TEXT
rating DOUBLE
cuisine_path TEXT
nutrition TEXT
timing TEXT
img_src TEXT
dietary restrictions TEXT
calories DOUBLE
protein TEXT
fat TEXT
fiber TEXT
difficulty TEXT
cooking tip TEXT
user_id INT
MyUnknownColumn TEXT
expired_products
ingredient_id INT
Name VARCHAR(255)
expiration_date DATE
quantity INT
store_id INT
recipe_ingredient_copy
recipe_id INT
ingredient_id INT
sessions
session_id VARCHAR(255)
user_id VARCHAR(36)
session_start DATETIME
session_end DATETIME
expires INT
data TEXT
user_agent VARCHAR(255)
Indexes
ingredient_copy
ingredient_id INT
name VARCHAR(255)
manages
admin_id INT
user_id INT
Indexes
admin
admin_id INT
uuid CHAR(36)
email VARCHAR(255)
username VARCHAR(255)
password_hash VARCHAR(255)
verification_status TINYINT(1)
accessibility TINYINT(1)
university_id INT
Indexes
user_recipes
user_recipe_id INT
user_id INT
recipe_name VARCHAR(255)
prep_time INT
cook_time INT
total_time INT
yield VARCHAR(50)
directions TEXT
img_source VARCHAR(255)
nutrition TEXT
servings INT
dietary_restrictions TEXT
calories INT
fat INT
protein INT
fiber INT
difficulty VARCHAR(50)
cooking_tip TEXT
accessibility TINYINT(1)
is_approved TINYINT(1)
admin_id INT
Indexes
activity_log
log_id INT
user_id INT
action VARCHAR(255)
timestamp DATETIME
Indexes
ingredient
ingredient_id INT
name VARCHAR(255)
img_src TEXT
Indexes
Users
user_id INT
uuid CHAR(36)
email VARCHAR(255)
username VARCHAR(255)
password_hash VARCHAR(255)
verification_status TINYINT(1)
accessibility TINYINT(1)
blacklist TINYINT(1)
role_id INT
university VARCHAR(255)
Indexes
store
store_id INT
Name VARCHAR(45)
ingredient_id INT
user_id INT
quantity INT
expiration_date DATE
university_id INT
Indexes
email_log
email_log_id INT
user_id INT
email VARCHAR(255)
timestamp DATETIME
action VARCHAR(45)
Indexes
roles
role_id INT
role_name VARCHAR(50)
Indexes
has
user_id INT
university_id INT
Indexes
notifications
notification_id INT
user_id INT
message TEXT
is_read TINYINT(1)
timestamp DATETIME
Indexes
recipe_ingredient
recipe_id INT
ingredient_id INT
Indexes
university
university_id INT
name VARCHAR(255)
email_suffix VARCHAR(255)
Indexes

ERD Diagram:

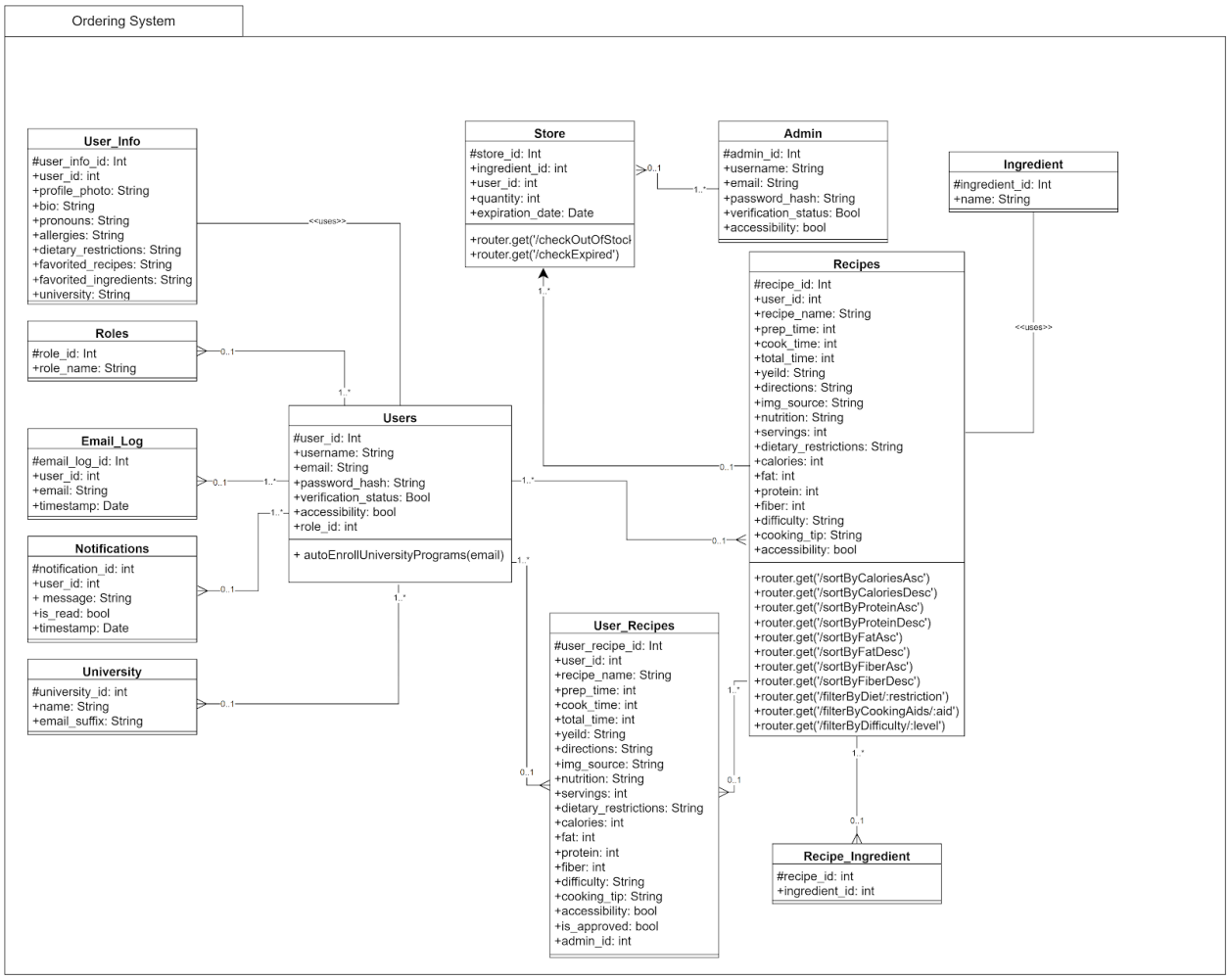

follows
following_user_id integer
followed_user_id integer
created_at timestamp
users
id integer
username varchar
role varchar
created_at timestamp
posts
id integer
title varchar
body text
user_id integer
status varchar
created_at timestamp
activity_log
log_id int NN
user_id int NN
action varchar(255) NN
timestamp datetime NN
has
user_id int NN
university_id int NN
email_log
email_log_id int NN
user_id int NN
email varchar(255) NN
timestamp datetime NN
action varchar(45) NN
notifications
notification_id int NN
user_id int NN
message text NN
is_read tinyint(1) NN
timestamp datetime NN
user_info
user_info_id int NN
user_id int NN
profile_photo varchar(255)
bio text
pronouns varchar(50)
allergies text
dietary_restrictions text
favorited_recipes text
favorited_ingredients text
university varchar(255)
Users
user_id int NN
uuid char(36) NN
email varchar(255) NN
username varchar(255) NN
password_hash varchar(255) NN
verification_status tinyint(1) NN
accessibility tinyint(1) NN
blacklist tinyint(1) NN
role_id int NN
university varchar(255)
recipes
Unnamed: 0 int
recipe_name text
prep_time bigint
cook_time bigint
total_time text
servings double
yield text
ingredients text
directions text
rating double
cuisine_path text
nutrition text
timing text
img_src text
dietary restrictions text
calories double
protein text
fat text
fiber text
difficulty text
cooking tip text
user_id int
MyUnknownColumn text
ingredient_copy
ingredient_id int NN
name varchar(255) NN
recipe_ingredient_copy
recipe_id int NN
ingredient_id int NN
user_recipes
user_recipe_id int NN
user_id int NN
recipe_name varchar(255) NN
prep_time int
cook_time int
total_time int
yield varchar(50)
directions text
img_source varchar(255)
nutrition text
servings int
dietary_restrictions text
calories int
fat int
protein int
fiber int
difficulty varchar(50)
cooking_tip text
accessibility tinyint(1) NN
is_approved tinyint(1) NN
admin_id int
sessions
session_id varchar(255) NN
user_id varchar(36) NN
session_start datetime NN
session_end datetime
expires int NN
data text
user_agent varchar(255)
roles
role_id int NN
role_name varchar(50) NN
manages
admin_id int NN
user_id int NN
recipe_ingredient
recipe_id int NN
ingredient_id int NN
admin
admin_id int NN
uuid char(36) NN
email varchar(255) NN
username varchar(255) NN
password_hash varchar(255) NN
verification_status tinyint(1) NN
accessibility tinyint(1) NN
university_id int NN
ingredient
ingredient_id int NN
name varchar(255) NN
university
university_id int NN
name varchar(255) NN
email_suffix varchar(255) NN
store
store_id int NN
Name varchar(45)
ingredient_id int NN
user_id int
quantity int NN
expiration_date date
university_id int
expired_products
ingredient_id int NN
Name varchar(255) NN
expiration_date date NN
quantity int NN
store_id int NN
dbdiagram.io

5. High Level Diagrams (detailed)

Our system architecture involves every user utilizing either a computer or mobile device that has access to a web browser. That web browser accesses our web app, ScholarEats, over HTTP protocol. The web app interacts with several APIs, including the following tabs: Recipes, Ingredients, About, Contact Us, Notifications and Sign In/Up. The API requests are processed by Handlebars, which then sends the requests to the different load balancers. The load balancers distribute to whatever server is available.



UML Diagram:



Summary of UML Diagram:

The main entities are ‘Users’, ‘User_Info’, ‘Email_Log’, ‘University’, ‘Store’, ‘Admin’, ‘Recipes’, and ‘Ingredient’.

The ‘User_Info’ class stores detailed user information, such as ‘profile photo’, ‘bio’, ‘pronouns’, ‘allergies’, ‘dietary restrictions’, ‘favorited ingredients’, ‘favorited recipes’, and the ‘university’ the user belongs to. It has a one-to-one association with the ‘Users’ class.

The ‘Users’ class contains basic user information, including ‘username’, ‘email’, ‘password hash’, ‘verification status’, and ‘role ID’. It also has a method to automatically enroll users in university programs based on their email domain. The decision to make the User and User_Info entities separate classes was made in order to simplify maintenance.

The Roles class assigns a role to each user, which is linked through the ‘role_id’. Having Roles as its own class will help with scalability if future roles are ever created.

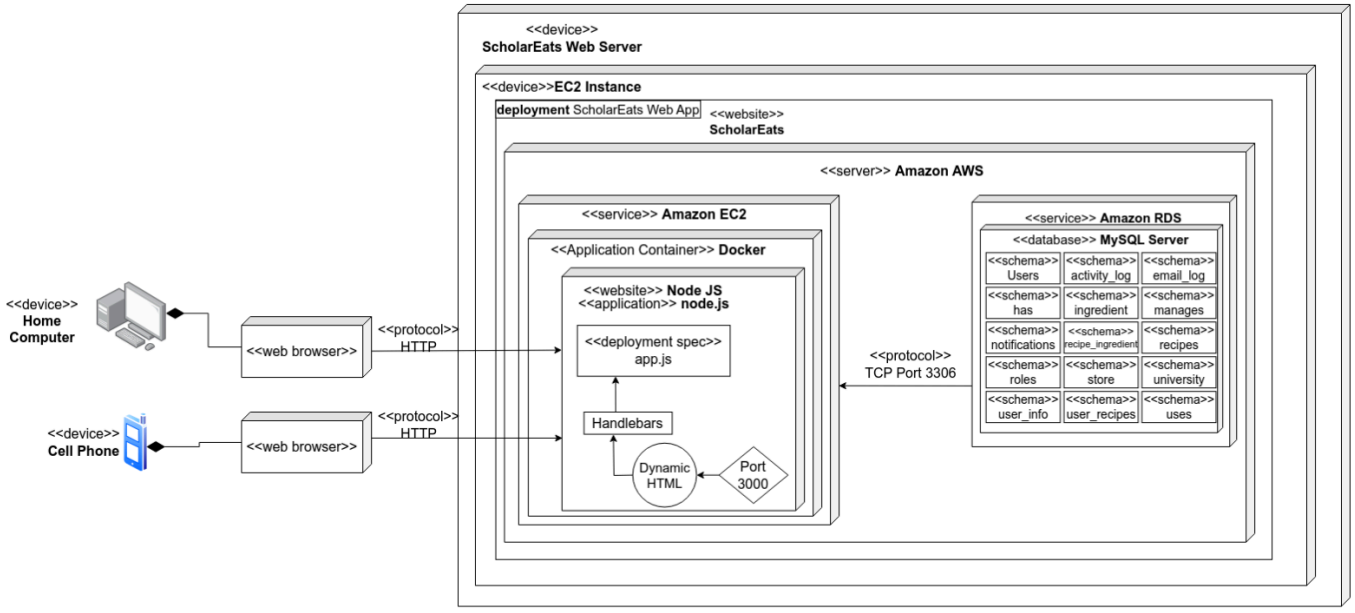
The ‘University’ class contains details about different universities, including their ‘names’ and ‘email suffixes’, which are used for auto-enrollment. The ‘Admin’ class stores admin details such as ‘username’, ‘email’, ‘password hash’, ‘verification status’, and ‘accessibility’. By keeping these entities in separate classes, this design helps to maintain data integrity. Also, making the Admin entity separate from a regular User entity was chosen for security reasons.

The ‘Store’ class manages inventory, including ‘ingredient IDs’, ‘quantities’, and ‘expiration dates’. It contains methods to check for out-of-stock items and expired ingredients and has a one-to-many relationship with the ‘Users’ class. This class is intended to help maintain the inventory in a centralized and consistent way.

The ‘Recipes’ class captures recipe details like ‘recipe name’, ‘preparation time’, ‘cook time’, ‘total time’, ‘yield’, ‘directions’, ‘image source’, ‘nutrition’, ‘servings’, ‘dietary restrictions’, ‘calories’, ‘fat’, ‘protein’, ‘fiber’, ‘difficulty’, ‘cooking tips’, and ‘accessibility’. It includes several methods for sorting and filtering recipes based on these attributes. The ‘User_Recipes’ class links users to their recipes, capturing details such as ‘servings’, ‘directions’, ‘ingredients’, and ‘admin approval status’. Having these various filters and sort options helps to optimize data retrieval.

The ‘Ingredient’ class lists possible ingredients, while the ‘Recipe_Ingredient’ class maps ingredients to recipes, forming a many-to-many relationship between ‘Recipes’ and ‘Ingredient’. By separating these classes, we can include information, like preparation instructions or quantity for a particular ingredient in a particular recipe without affecting the definition/function of the Recipe or Ingredient as a whole.

Application Network and Deployment Design:



6. List of Contributions in this milestone (detailed including contributions to the horizontal prototype)

- Donovan: (10 /10)
    - Adjusted Routing for the individual recipes in the database to have their own pages
    - Adjusted UI to incorporate filtering within search results
    - Notifications Page
    - Site-wide .css adjustments
    - Restructured handlebar's structure to improve ease of development
    - Iterated on Navigation bar to follow coding convention
    - Led a meeting regarding the Wireframes for the Use-cases
- Hancun: (8 /10)
    - Admin Tools Page
    - Created default image asset for recipes
    - Created default image asset for ingredients
    - Contributed to .css adjustments
    - Assisted in site-wide improvements
- Tina: (7.5 /10)
    - Added fields to Account Management to account for allergies, website theme and more
    - Drafted the handlebars view for the Individual Recipe page
    - Drafted the handlebars view for the site's Landing page
    - Contributed to the updating of the Use-Cases' Wireframes
- Edward: (10 /10)
    - Configured possibility of filtering within a given search result.
    - Created ways to check if user was logged in between pages.
    - Led the adjustments to Section 4 and 5 following the return of feedback from Milestone 2 Version 2
    - Led meetings to iterate on the back-end and set the groundwork for upcoming features
- Karl: (8.5 /10)
    - Connected Allergies to Database
    - Connected Dietary Restrictions to Database
    - Created ways to check if a user was logged in between pages.
    - Adjusted various validations
    - Restructured session management
- Sai: (7.5 /10)
    - Implemented methods for User Authentication
    - Implemented methods for Inventory Management
    - Populated various database tables
    - Admin Authorization Table
    - Updated the default values of the recipes and ingredients to point to Hancun's new assets for their image source (img_src column)

- Maeve: (8 /10)
    - Contributed to the updating of the Use-Cases' Wireframes
    - Managed and oversaw the move to documenting on Notion
    - Managed and oversaw the move back FROM Notion after our storage requirements changed
- Sabrina: (8 /10)
    - Contributed to the updating of the Use-Cases' Wireframes
    - Assisted in the UI iterations
    - Assisted in implementation of user preferences for the site
    - Drafted CSS for Dark Mode
    - Adjusted Account Management options

Prototype Feedback

- Simple but good mockups/use cases
- Should make a separation/space between the header components - it's too easy to accidentally click on the wrong button
- put a placeholder like "Search for recipes" in search bar - currently unclear what user should search for
- Must clarify that the results are recommendations when there are no results - ex. "No matches - here are some recommendations"
- Tagline itself is really good -says exactly what the website is about - but font size should be increased and should be formatted to not be in a single sentence
- Distinguish coloring of header and footer from the main content - currently confusing
- Registration is not good/doesn't work – make it look like Login
- WE USED ARROWS IN OUR WEB FRAMES YAY
- Make sure site matches the use cases - specifically header buttons
- User doesn't know if logged in – "Welcome" isn't good enough - add other signifier (ex. Hi _____ or Default Profile Picture)

Following the feedback given during class the team immediately met to discuss how to apply the feedback and correct any deficiencies that we saw. We immediately tasked members on site-wide .css adjustments involving the colors chosen on the site and adjust the UI for users to have a better experience traversing the site.

Additionally the team has moved to incorporate changes that allow for clearer error messaging for the user to understand why they are seeing certain results or why a search reveals recommendations instead of what they searched for. This will also expand to better showing that the user has logged in instead of going to another page without seemingly any response.

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

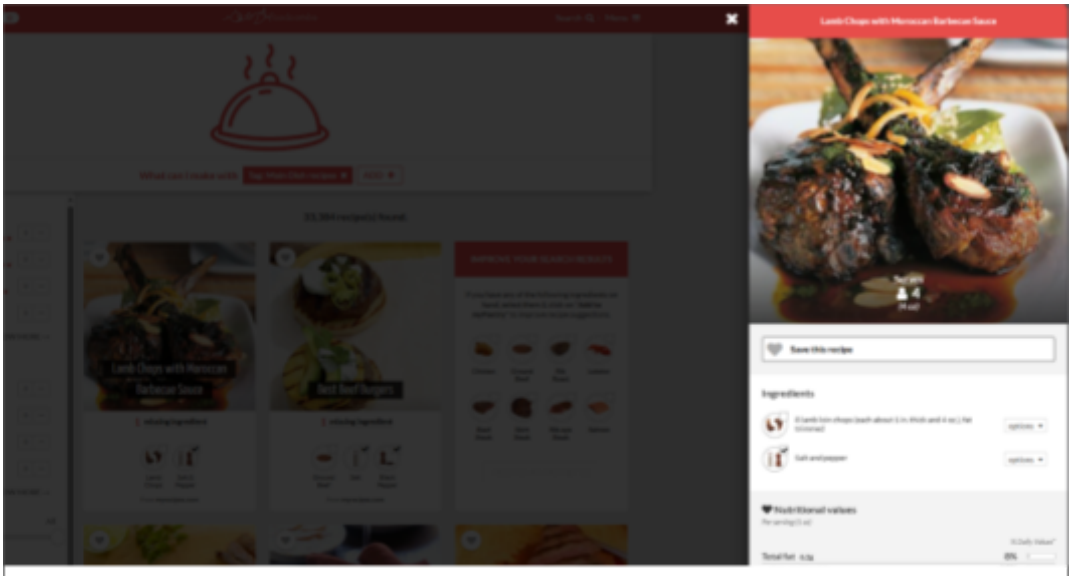
ALL major committed functions:

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 PRONOUNS
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).



Our competitor: foodcombo. No reserve option.

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/accountManagemen
- 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	- Some users suggested adding more advanced search options - Users appreciated ability to narrow down search results effectively - Navigation from search results to recipe page from smooth and straightforward	20 sec/ 20 sec
Update Allergy Information	100%	None	- Users easily located the allergy information section. - The update process was straightforward and clear.	20 sec / 20 sec

			- Users experienced a smooth saving process with clear confirmation.	
Change Dietary Restrictions	100%	None	- Users easily located the dietary restrictions section. - The update process was straightforward and clear. - Users experienced a smooth saving process with clear confirmation.	20 sec / 20 sec
Manage User Profile	100%	None	- Users easily located the profile information section. - The process was straightforward and clear. - Users experienced a smooth and clear password change process, in addition to verify their new password again.	55 sec / 45 sec
Reserve Available Ingredient	100%	None	- The process of adding items to the reservation list was straightforward.	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	- Exactly as to be expected	- Uniform - without confusion
Update Allergy Information	20 sec	20 sec / 20 sec	- Easily located section and smooth saving process	- Straightforward and clear single screen
Change Dietary Restrictions	20 sec	20 sec / 20 sec	- Found section without confusion - Saved dietary info without delays	- Single page to complete change
Manage User Profile	45 sec	45 sec / 55 sec	- Easy to find account button when logged in - Easy to change and divide to submit each part of profile	- Looks uniform

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” http://3.148.145.110:3000/”	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” http://3.148.145.110:3000/”	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” http://3.148.145.110:3000/”, 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” http://3.148.145.110:3000/”, 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:

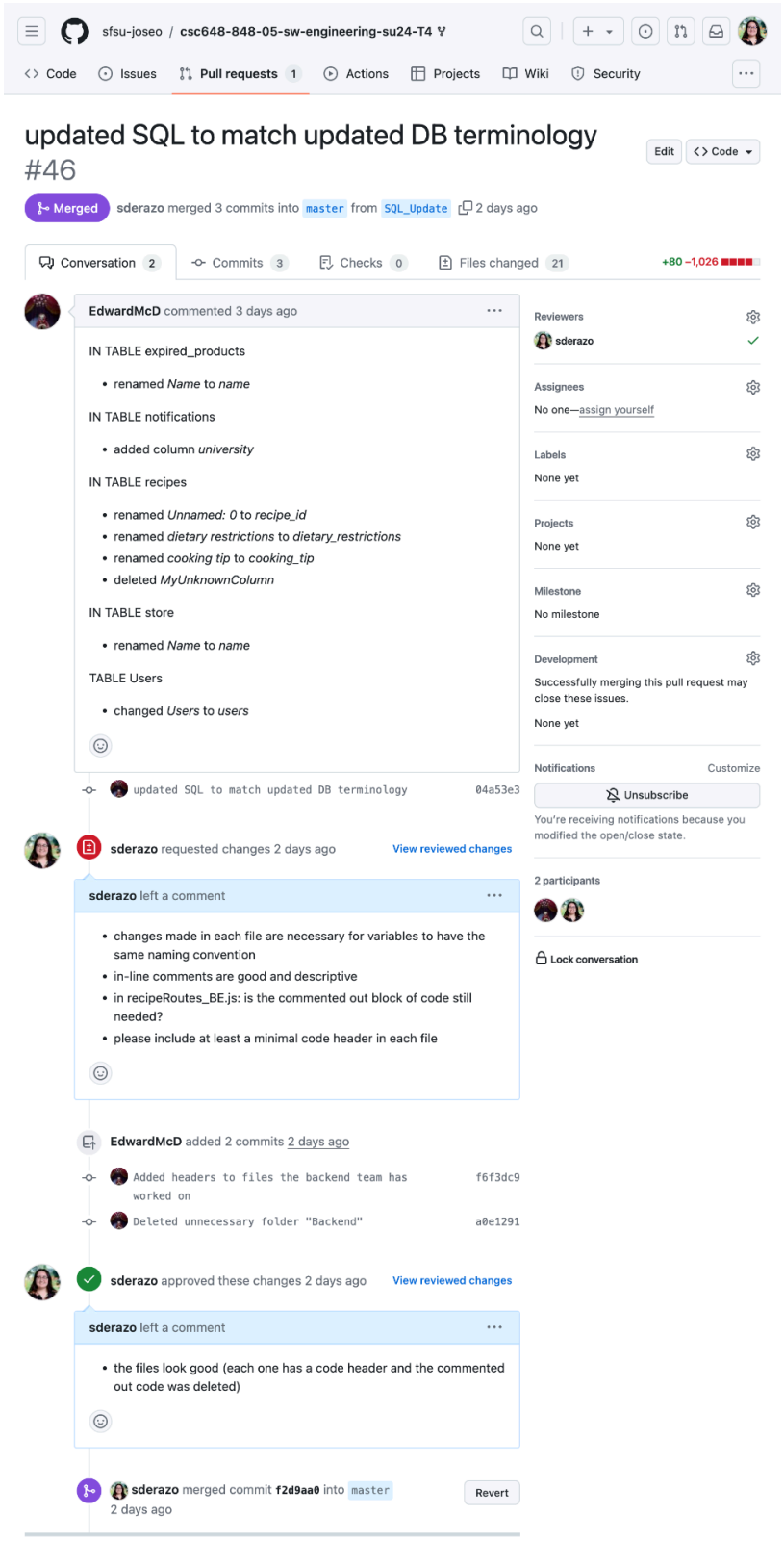
- Do NOT make separate app.js’s and commit them to branches that you are trying to merge into main/master.
- Master branch should be production ready code, free of testing aids. If you feel the need to include something to demonstrate functionality and it doesn’t exist, implement a view and use routing to make it possible to access in the same application as a seperate 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())

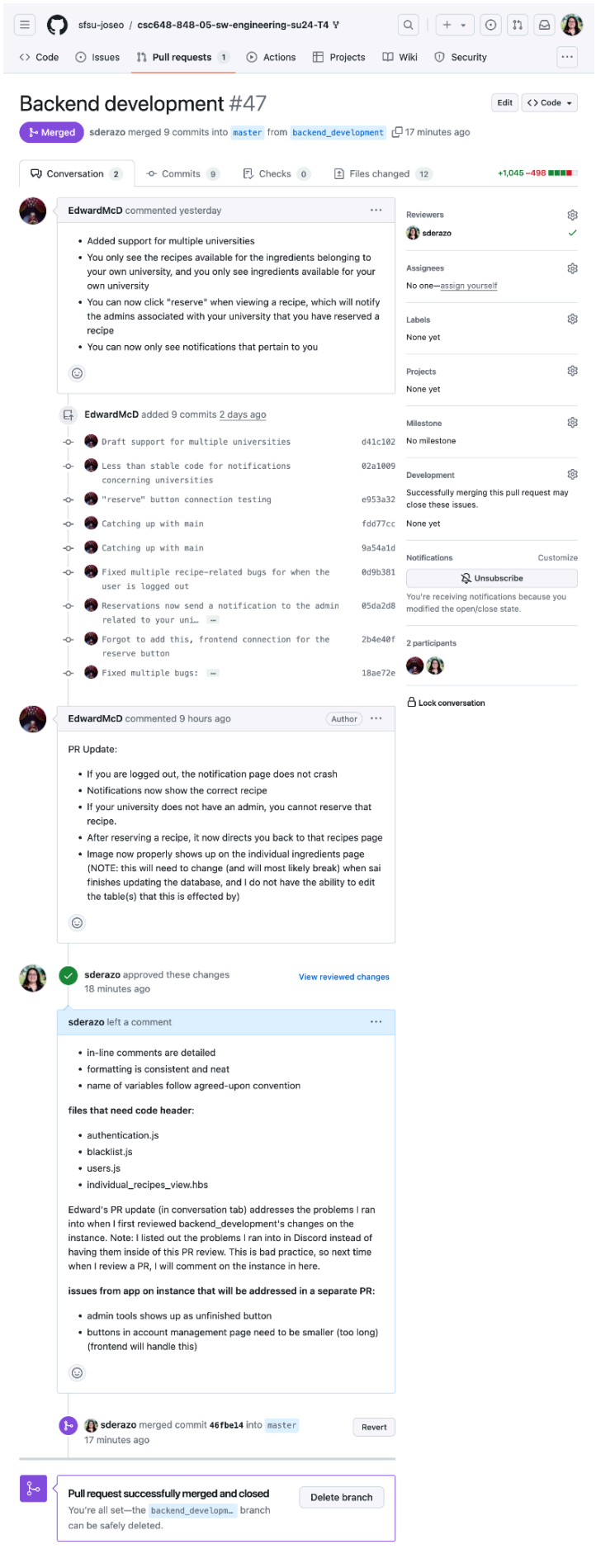
- Again, do not make separate applications with their own launching conditions or packages / package.json / package-lock.json.
- Failure to do so will result in your PR being denied until the branch is in compliance

Handlebars and CSS:

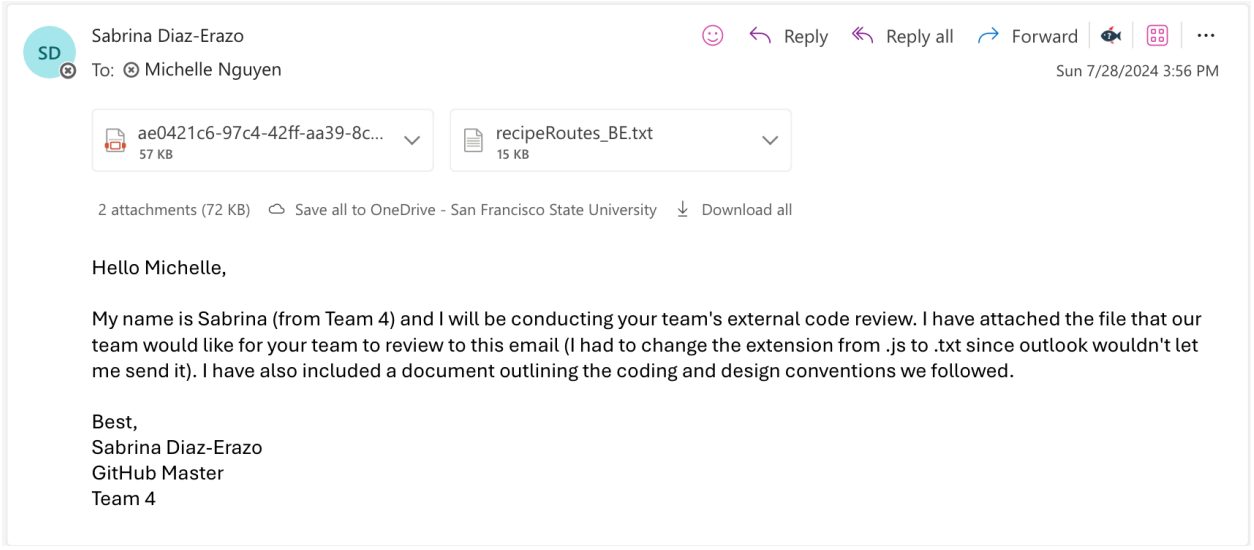
- We should be moving to utilize the “grid-area” form of setting up the layouts. This will enable us to better support displays of all dimensions
- As always, we should have a singular “layout” that maintains consistency throughout all pages.
- The unique and different pages will be done via “views” 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
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
EdwardMcD added 9 commits 2 days ago
Draft support for multiple universities d41c102
Less than stable code for notifications concerning universities 02a1009
"reserve" button connection testing e953a32
Catching up with main fdd77cc
Catching up with main 9a54a1d
Fixed multiple recipe-related bugs for when the user is logged out 0d9b381
Reservations now send a notification to the admin related to your uni… 05da2d8
Forgot to add this, frontend connection for the reserve button 2b4e40f
Fixed multiple bugs: 18ae72e
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 46fbe14 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
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

c) External Code Review for Different Team



Sabrina Diaz-Erazo
To: Michelle Nguyen

Reply | Reply all | Forward

Sun 7/28/2024 3:56 PM

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

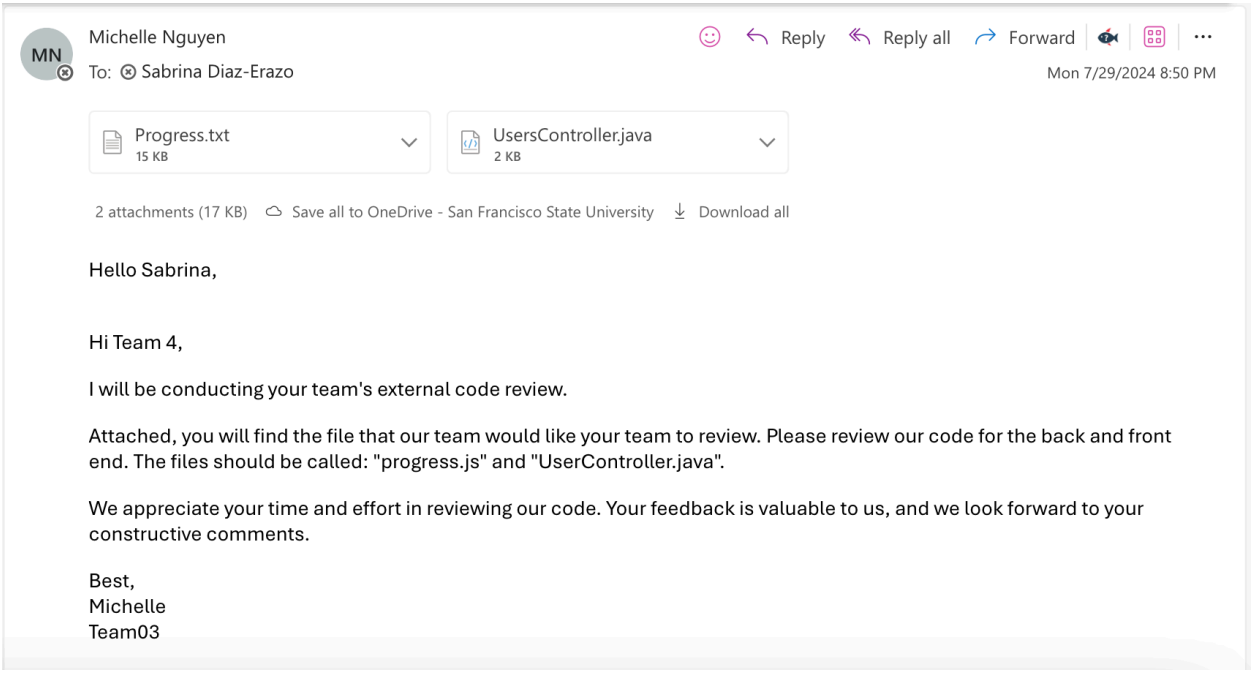
recipeRoutes_BE.txt 15 KB

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
To: Sabrina Diaz-Erazo

Reply | Reply all | Forward

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

SD

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

MN

Michelle Nguyen

To: Sabrina Diaz-Erazo

Mon 7/29/2024 9:47 PM

Please review any. I recommend the backend UserController.

MN

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. 1. Avoid hardcoding credentials in the code.
2. 2. Use environment variables or a secure vault to manage sensitive information
3. 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. 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

4. Post analysis – lessons learned (one page or so done by team lead)

Part A)

In the development of ScholarEats there were challenges in getting to a final product:

The first of which stems from the inexperience of virtually all team members in developing website applications. Prior to this, the only web experience many members had was in just CSC 317 and not all members utilized a tech stack similar to the one that ScholarEats utilizes.

Additionally many members have never utilized git and remote repositories in a team environment before so the usage of branches and pull requests to alter the main branch as opposed to simply pushing and pulling directly to and from the main branch was an area the team had to work on.

In regards to deadlines and work-tempo, the accelerated nature of summer has condensed many milestones to being completed much closer together which shortens both time for development and time for testing. Additionally the team lead had enrolled in a class load of 15 units for the summer session, including this class, which led to scheduling conflicts and quality assurance of a major section in one of the milestones not being completed.

Part B)

If we were to revisit the development of this app from the very beginning there are many things that would be addressed differently:

Donovan had taken the time to conduct training during some of our meetings to show the team as a whole how we were implementing things like generating the html for the pages dynamically and most importantly how to do so. This was critical for team members to be able to be tasked out and be able to complete work individually. There were definitely some other things that needed a similar effort, primarily the usage of git which Angelo had demonstrated once, however needed more given how prevalent it was to development. Examples of this include PR's, SQL database executions, and minor things like decisions behind our attempts at cascading style sheets and html architecture of pages.

The condensed time-frame called for things needing to have been done much earlier so proper testing and design iteration could occur. The team lead in future efforts should ensure a soft deadline ahead of the actual deadline to catch issues with testing and documentation at a point where there will still be time to fix it properly to avoid a rushed milestone or a rushed project.

5) Team member contributions

1. Donovan Taylor - Score (10 / 10)
    a. Front-End Lead
    b. Donovan performed remarkably as the lead for the front-end section. He often led meetings to tackle individual tasks and his contributions have been critical to the success of this product.
    c. Additionally he has gone out of his way to specifically show team-members how certain things have been implemented and how to use core dependencies that the project has decided to lean into (such as the usage of Handlebars). This helped others better understand our tech stack.
    d. Implemented the debugging tools showing warning messages for errors and un-implemented features through development.
    e. Configured the UI to properly be able to set and use Sabrina's "dark mode" .css implementation
2. Hancun Guo - Score (7 / 10)
    a. Front-End Developer
    b. Hancun consistently has been able to handle tasks given to him and has done many of the pages present on the site.
    c. Additionally Hancun has gone through and debugged many issues with .css and the passing of .css and .js files through Handlebars.
    d. Hancun has also gone through files to ensure compliance with the team's agreed upon Coding Conventions
3. Tina Chou - Score (7 / 10)
    a. Front-End Developer
    b. Since Tina has become a member of this team she has been eager to work and has been able to implement whole pages such as the individual recipe's page by herself after being designed. Overall an effective worker.
    c. Had a major role in implementing UI for user's preferences
4. Edward McDonald - Score (10 / 10)
    a. Back-End Lead
    b. Edward has gone above and beyond in his duties. Like Donovan he has held meetings to tackle tasks and most importantly been avid in his participation for all meetings to ensure that every element of the site properly interfaces with the backend. This alone has been the most important factor in maintaining smooth development and the roll-out of features
5. Karl Carsola - Score (7.5 / 10)
    a. Back-End Developer

    b. Karl has implemented and debugged many features that the site greatly benefitted from. He has implemented features such as Session Management and consistency in behavior using that framework.
    c. Additionally he worked with Sai in implementing features involving changing or setting users' credentials.
    d. Worked closely with the usage of bcrypt encrypting and storing passwords.
6. Sai Bavisetti - Score (5.5 / 10)
    a. Database Admin
    b. Participated in discussions to ensure that relevant sections were able to get the data or information that they needed as well as getting that information in a way that they needed.
    c. Provided all relevant and up-to-date database information to Maeve and others for documentation recording.
    d. Provided tools and API's for usage by other team members such as Auto-complete and administrative tools that manage user's or the ingredients available.
7. Maeve Fitzpatrick - Score (7.5 / 10)
    a. Document Editor
    b. Maeve has been versatile and very adamant about being involved in many different corners of the project. She has also led meetings and shown the ability to delegate effectively to reduce the turn-around time on large portions of requirements.
    c. Maeve has also been able to be assigned to assist others when needed, such as retooling the site-wide .css to better distinguish between elements on the page following feedback from the CTO.
    d. Maeve additionally made some .css adjustments to better display the site on tablet and mobile views
8. Sabrina Diaz-Erazo - Score (8 / 10)
    a. Github Master
    b. Sabrina has consistently been able to keep the logistical aspect of our code development running smoothly, minimizing conflicts involving any potential errors from the usage of Git.
    c. She has reviewed and approved Pull Requests quickly, sometimes several at a time, and ensured that our testing policies and quality requirements have been met throughout our development cycle.
    d. Additionally she has worked to implement features that deal directly with the user experience and look of the site such as the .css for Dark Mode.
    e. Created the site footer with the social media pages and email

f. Made the Contact Us page's view in Handlebars
g. Made the terms of service and privacy policy page
h. Added search suggestion text inside of search bar input