SW Engineering CSC648/848 Spring 2021

GatorDash

Team 05

Team lead: Bryan Caldera / bcaldera@mail.sfsu.edu

Frontend lead: John To
Backend lead: Denny Feng
Github lead: Marco Marino
Calvin Tan
Huan Nguyen

Milestone 4

5/14/21

History Table

- Version 1 of Milestone 4 - 5/14/21

Table of Contents

1. Product Summary

a. Gator Dash

b. Major committed functions

  i. **General User:**
    1. General users shall be able to create an account
    2. General users shall be able to use search (all features)

  ii. **Registered User:**
    1. Registered users shall be able to order food
    2. Registered users shall be able to choose pickup or delivery
    3. Registered users shall be able to login

  iii. **Restaurant:**
    1. Restaurants shall be able to upload restaurant info with our service
    2. Restaurants owners shall be able to manage (add or delete) their menu

  iv. **Deliverer:**
    1. Shall be able to view orders available to be picked up and delivered
    2. Shall see order details of order they are delivering
    3. Shall have access to an SFSU map
    4. Shall see map with pin on restaurant address

  v. **Admin:**
    1. Admins shall have an easy to access list of the restaurants that are applying to join the service.
    2. Admins shall be required to approve restaurant registrations before they go live.

c. URL:
http://ec2-3-135-197-193.us-east-2.compute.amazonaws.com/

2. Usability Test Plan

a. Test objectives

i. We will be testing our website's search function. The goal of this test will be to assess its accuracy and thoroughness in retrieving the requested information and presenting it to the user in an organized, easy to read manner. The search function is a vital component of any website and provides a method that allows users to find the information they want as quickly and easily as possible. A search function in a website like ours is especially critical as a malfunctioning search function is liable to cause immense frustration in users and almost immediately dissuade them from using our website any longer. We want to use user feedback to best understand how to streamline our search function and improve its effectiveness, efficiency, and ease of use.

b. Test background and setup

i. Each user in our test will have their

ii. Each user will start their usability test on the homepage of our website.

iii. The intended users are SFSU students and faculty

iv. The URL to be used will be http://ec2-3-135-197-193.us-east-2.compute.amazonaws.com/

v. We will be measuring the user satisfaction of our testers after the testing process using likert questions and fill-in questions. The questions will cover not just the holistic satisfaction of the user during the test but will cover their thoughts on the search function's ease-of-use, simplicity of design, aesthetics, and readability.

c. Usability Task description

i.

Task 1	Search for Italian food
Task 2	Search for a restaurant
Task 3	Search for American food

d. Evaluation of Effectiveness

   i. We would measure the effectiveness of the search based on the accuracy and number of search results returned and how quickly the user is able to achieve their goal.

e. Evaluation of efficiency

   i. We would measure the efficiency of our search based on the time spent by the user to complete the assigned task as well as the time spent searching and amount of interactions used, such as clicks.

f. Evaluation of user satisfaction

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The search bar was easy to use.					
I got the search results that I was looking for.					
The search bar was easy to find.					

3. QA Test Plan

a) Test Objectives

i. The goal of our QA testing is to assess the robustness and stability of code written for and used by the search function. The search function is being tested to ensure output is correct and to discover any bugs the search function may have. The search function must return the correct amount of relevant results with little to no issue

HW and SW setup

ii. HW

1. Our server host is running Ubuntu 20.04 LTS off of AWS t2.micro. It has 1vCPU at 2.5 GHz and 1GiB memory.

iii. SW

1. Our web server is running Express 4.17.1.
2. Our database is running MySQL 8.0.23
3. Our website is also using Node 14.15.5 LTS, React 17.0.1, and Redux 4.0.5
4. The URL for our website is http://ec2-3-135-197-193.us-east-2.compute.amazonaws.com/

Feature to be tested

iv. Search function

QA Test plan

Test #	Test Title	Test Description	Test Input	Expected correct output	Test Result
1	Search for italian cuisine	Tests backend for filtering restaurants by cuisine.	Press search button in navigation bar then use cuisine	The restaurant ItalianCusine is the only search result	

			selector from navbar to select Italian		
2	Search for restaurant	Tests backend for %like restaurant name.	Type “Dim” into search bar and press search button	Taken to search result page and restaurant DimSum is the only result.	
3	Search for All cuisines	Tests front end for providing all restaurants in service.	Press search button on navigation bar without typing anything into search bar	Taken to search result page and all available restaurants are listed	

b)

Test 1 Safari Browser:

Test #	Test Title	Test Description	Test Input	Expected correct output	Test Result
1	Search for italian cuisine	Tests backend for filtering restaurants by cuisine.	Press search button in navigation bar	The restaurant ItalianCusine is the only search result	PASS

			then use cuisine selector from navbar to select Italian		
2	Search for restaurant	Tests backend for %like restaurant name.	Type “Dim” into search bar and press search button	Taken to search result page and restaurant DimSum is the only result.	PASS
3	Search for All cuisines	Tests front end for providing all restaurants in service.	Press search button on navigation bar without typing anything into search bar	Taken to search result page and all available restaurants are listed	PASS

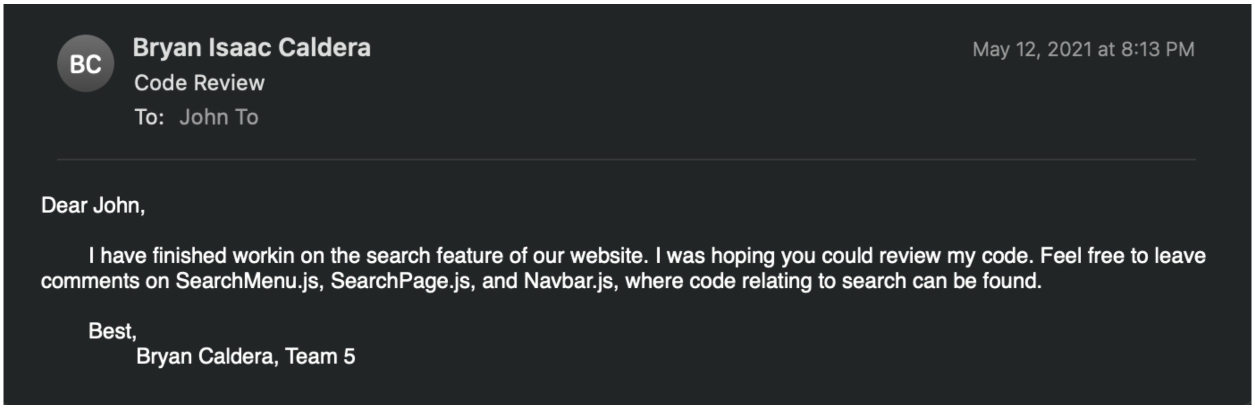
Test 2 Chrome Browser:

Test #	Test Title	Test Description	Test Input	Expected correct output	Test Result
1	Search for italian cuisine	Tests backend for filtering restaurants by cuisine.	Press search button in navigation bar then	The restaurant ItalianCusine is the only search result	PASS

			use cuisine selector from navbar to select Italian		
2	Search for restaurant	Tests backend for %like restaurant name.	Type “Dim” into search bar and press search button	Taken to search result page and restaurant DimSum is the only result.	PASS
3	Search for All cuisines	Tests front end for providing all restaurants in service.	Press search button on navigation bar without typing anything into search bar	Taken to search result page and all available restaurants are listed	PASS

4. Code Review

John reviewing Bryan:



BC **Bryan Isaac Caldera**

Code Review

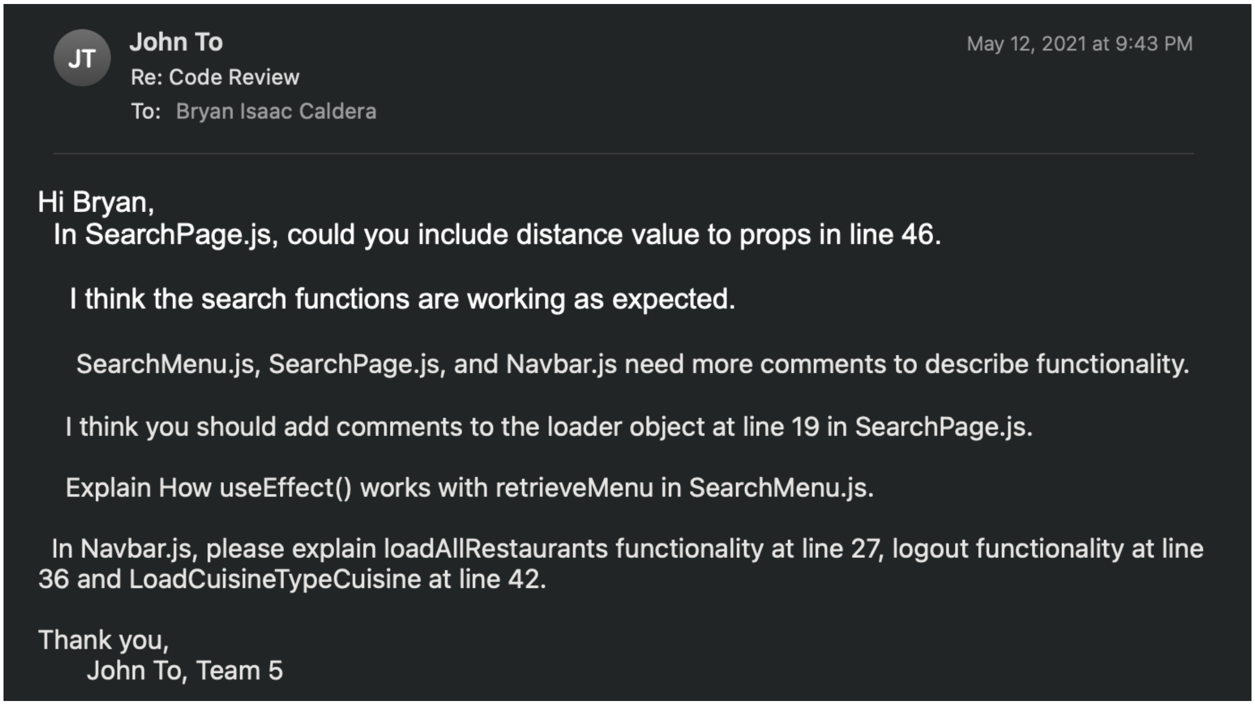
To: John To

May 12, 2021 at 8:13 PM

Dear John,

I have finished workin on the search feature of our website. I was hoping you could review my code. Feel free to leave comments on SearchMenu.js, SearchPage.js, and Navbar.js, where code relating to search can be found.

Best,
Bryan Caldera, Team 5



JT **John To**

Re: Code Review

To: Bryan Isaac Caldera

May 12, 2021 at 9:43 PM

Hi Bryan,
In SearchPage.js, could you include distance value to props in line 46.

I think the search functions are working as expected.

SearchMenu.js, SearchPage.js, and Navbar.js need more comments to describe functionality.

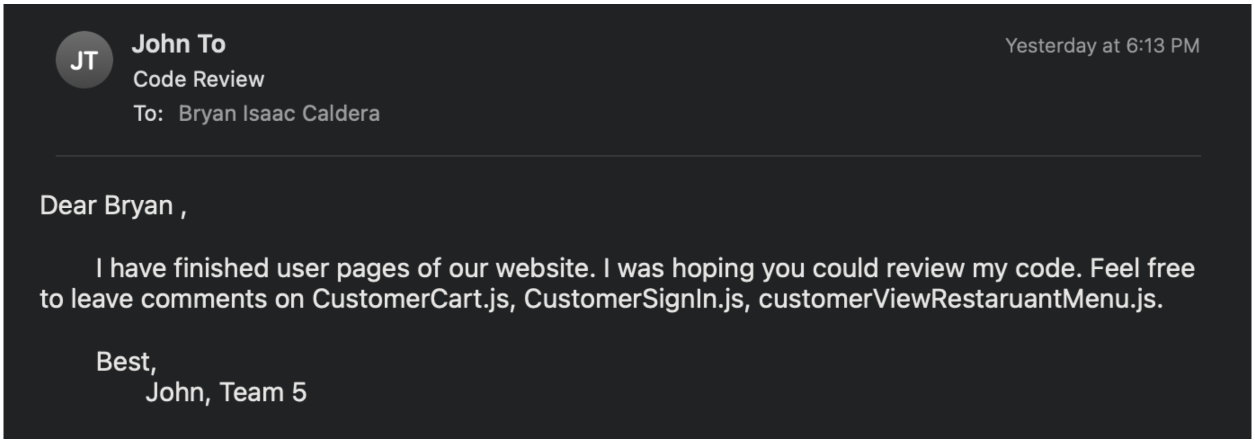
I think you should add comments to the loader object at line 19 in SearchPage.js.

Explain How useEffect() works with retrieveMenu in SearchMenu.js.

In Navbar.js, please explain loadAllRestaurants functionality at line 27, logout functionality at line 36 and LoadCuisineTypeCuisine at line 42.

Thank you,
John To, Team 5

Bryan reviewing John:



JT **John To**

Code Review

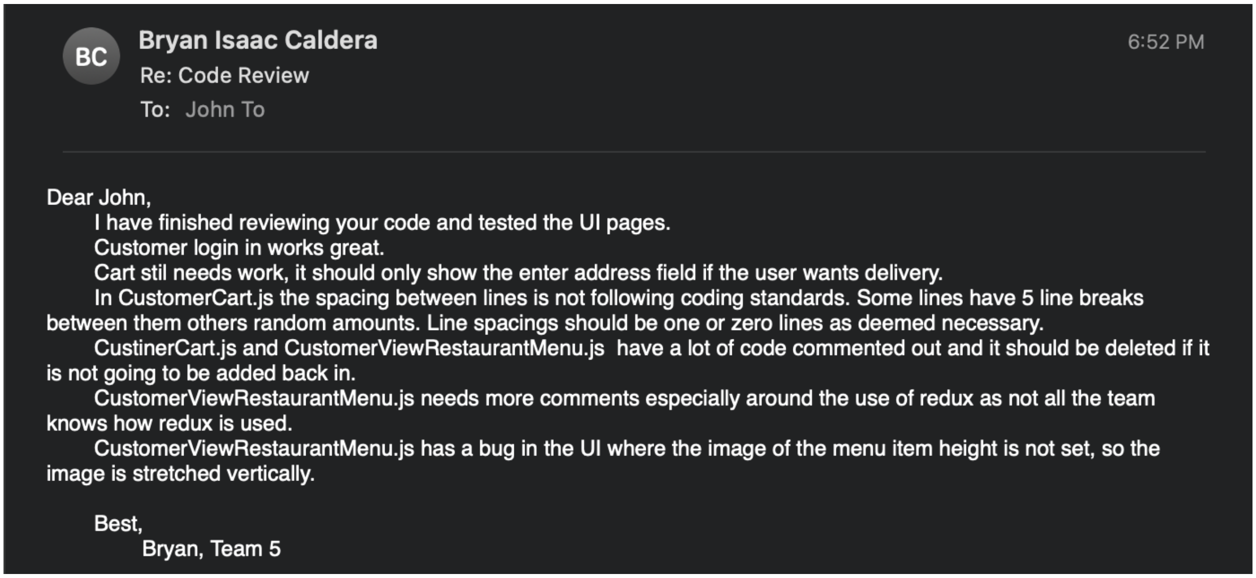
To: Bryan Isaac Caldera

Yesterday at 6:13 PM

Dear Bryan ,

I have finished user pages of our website. I was hoping you could review my code. Feel free to leave comments on CustomerCart.js, CustomerSignIn.js, customerViewRestaruantMenu.js.

Best,
John, Team 5



BC **Bryan Isaac Caldera** 6:52 PM

Re: Code Review

To: John To

Dear John,

I have finished reviewing your code and tested the UI pages.

Customer login in works great.

Cart stil needs work, it should only show the enter address field if the user wants delivery.

In CustomerCart.js the spacing between lines is not following coding standards. Some lines have 5 line breaks between them others random amounts. Line spacings should be one or zero lines as deemed necessary.

CustinerCart.js and CustomerViewRestaurantMenu.js have a lot of code commented out and it should be deleted if it is not going to be added back in.

CustomerViewRestaurantMenu.js needs more comments especially around the use of redux as not all the team knows how redux is used.

CustomerViewRestaurantMenu.js has a bug in the UI where the image of the menu item height is not set, so the image is stretched vertically.

Best,
Bryan, Team 5

5. Self-check on Best Practices for Security

a.

Asset to be Protected	Types of possible/expected attacks	Strategy to mitigate/protect the asset
User passwords	Database leak	Passwords are hashed in the database, at no point will passwords be visible as plain text
SFSU only	Users not from SFSU try to register	We have a check on user registration so only people with an sfsu.edu email can sign up.
User input into database	SQL code injection	Backend SQL calls use escaping variables
Search bar	Code injection	We limit search to only 40 characters

6. Self-Check:Adherence to original Non-functional specs

Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in	ON TRACK

M0. Application delivery shall be from chosen cloud server	
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	DONE
All or selected application functions must render well on mobile devices (specifics to be developed in consultation with users e.g. Petkovic)	DONE
Ordering and delivery of food shall be allowed only for SFSU students, staff and faculty	DONE
Data shall be stored in the database on the team's deployment cloud server.	DONE
No more than 50 concurrent users shall be accessing the application at any time	DONE
Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	DONE
The language used shall be English (no localization needed)	DONE
Application shall be very easy to use and intuitive	DONE

Application should follow established architecture patterns	DONE
Application code and its repository shall be easy to inspect and maintain	DONE
Google analytics shall be used	DONE
No e-mail clients shall be allowed.	DONE
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	DONE
Site security: basic best practices shall be applied (as covered in the class) for main data items	DONE
Application shall be media rich (images, maps etc.). Media formats shall be standard as used in the market today	DONE
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development	DONE
The application UI (WWW and mobile)	ON TRACK

shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Spring 2021 For Demonstration Only”* at the top of the WWW page. (Important so as to not confuse this with a real application).	