SW Engineering CSC648/848 Spring 2023

Project Name: GatorEats - Online Meal-Delivery Service

By: Team 7

Milestone 4

Github: https://github.com/CSC-648-SFSU/csc648-03-sp23-team07.git

Team Lead: Andy Almeida - aalmeida1@sfsu.edu,

Team Members: Emily Huang, Melisa Sever, Eunice Borres, and Juan David Liang Liao

Date: 5/11/2023

Document History:

Date Submitted:	5/22/2023

Table of Contents

1) Product Summary

Name: **GatorEats**

Description: GatorEats is an online platform that connects food-selling establishments with the San Francisco State University community. Everyone with an SFSU email account, including students, employees, and faculty, can access GatorEats. Clients can have their food delivered to one of the numerous drop-off locations or dorm rooms on campus. Customers have the freedom to select from a variety of cuisines thanks to the app's extensive range of options from numerous eateries. With its unique delivery service to particular areas, GatorEats distinguishes itself further and gives clients a hassle-free delivery experience. By offering everyone a quick, dependable, and practical service, we hope to change the food delivery sector.

Itemized List of Committed Functions:

1. Guest users shall be able to browse the site and search using food categories or cuisines.
2. Guest users shall be able to register for an account if they choose to.
3. Guest users shall be able to add items to a cart.
4. Guest users shall be required to sign up for an account in order to check out an order.
5. SFSU registered users shall be able to log into their account.
6. SFSU registered users shall have a email address (sfsu.edu/mail.sfsu.edu) and password
7. SFSU registered users shall be able to order food.
8. SFSU registered users must choose a location for the food during the order check out.
9. SFSU registered user must complete the check out by using Gator Card as payment
10. A restaurant user shall be able to register their account.
11. A restaurant user shall be able create a menu with dishes and prices for those dishes.
12. An admin user shall approve the restaurant via MySQL Work Bench.
13. A driver user shall be able to register as a delivery driver.
14. A driver user shall be able to receive a delivery order.
15. A driver shall have access to a list of created orders to choose from and deliver.
16. A menu shall have one or more food options, descriptions, and prices.
17. Restaurants shall be linked to a category.
18. A pickup point shall have a building name and a room number.

URL: http://34.219.117.242:3000

2) Usability Test Plan for Selected Function

Test objectives: The major function that will be tested for usability will be the functionality of the Restaurant search and menu/menu item interactions. This usability testing is being conducted to ensure that the ease of use for this process is as seamless as possible. The goal is to understand the challenges and difficulties that a user might encounter when attempting to utilize a much needed function of our application.

Test background and setup:

The System setup for the usability test is as follows. A user must select a working computer or handheld device with a modern operating system such as Windows, IOS, or Linux distribution. They must also have a working internet connection available on this device. The device must also have a modern version of a working internet web browser. The one that will be used for this set of testing would be Mozilla Firefox. With these requirements fulfilled, the user can begin to conduct the usability testing on their device.

The starting point for this set of testing will begin at the home page of the application. The starting point can be accessed by this URL: http://34.219.117.242:3000

The intended users that would be utilizing the functions that are being tested for usability are actual users that are interested in using GatorEats as a product delivery service. The users that are intended to use this may have a varying level of knowledge with technology, however this should not affect the usability of the application. The users may or may not know exactly what they are looking for in terms of food categories/restaurants, but the application should be supportive in helping a user reach their goal of receiving food seamlessly.

The features that will be measured for usability will relate to the user experience of a regular user. A user should be able to easily navigate the application to search for a desired restaurant, fill out a cart for the order that they are interested in placing, and follow through with placing that order. This process will be the main focus of this round of usability testing and will be thoroughly measured using multiple metrics.

Usability Task description:

- Register for an account on the Application
- LogIn to the application
- Search for a Restaurant to select from
- Fill your cart up with desired food items
- Place your order for delivery

Milestone 4

Plan for evaluation of Effectiveness:

Effectiveness of the application for this use case will be measured through measurements of the user being able to fill out all of their desired information to the system and task completion rates. These fields will be monitored during the testing user's experience. Having a low percentage of users that were able to complete all tasks would imply that the application is difficult to maneuver and not supportive to intuition.

Plan for Evaluation of efficiency:

Efficiency of the application for this use case will be measured through the time it takes to reach the next step in the usability testing, and how long the overall experience took. These measurements will indicate how easily the tasks can be completed, and the overall effectiveness of the placements, sizing, and layouts of the buttons/application

Plan for Evaluation of user satisfaction:

1) "Registering and logging in on the application was easy to navigate"

Strongly Disagree / Disagree / Neutral / Agree / Strongly Agree

2) "Finding a Restaurant that was interesting to me was easy for me to do"

Strongly Disagree / Disagree / Neutral / Agree / Strongly Agree

3) "Filling out my cart and placing an order was straightforward and intuitive"

Strongly Disagree / Disagree / Neutral / Agree / Strongly Agree

3) QA Test Plan and QA Testing

Test objectives:

The major function that will be tested for Quality Assurance will be the functionality of the Restaurant search and database item interactions.

HW and SW setup:

The System setup for the Quality Assurance test is as follows. A user must select a working computer or handheld device with a modern operating system such as Windows, IOS, or Linux distribution. They must also have a working internet connection available on this device. The device must also have a modern version of a working internet web browser. With these requirements fulfilled, the user can begin to conduct the Quality Assurance testing on their device.

The starting point for this set of testing will begin at the home page of the application. Here is the URL that will take the user to the homepage to begin the Quality Assurance: http://34.219.117.242:3000

Feature to be tested:

The feature to be tested will be the search feature of the application. Testing the search feature will also test the integrity and cohesion of the database, and ensure that the database is producing proper results for the frontend/user.

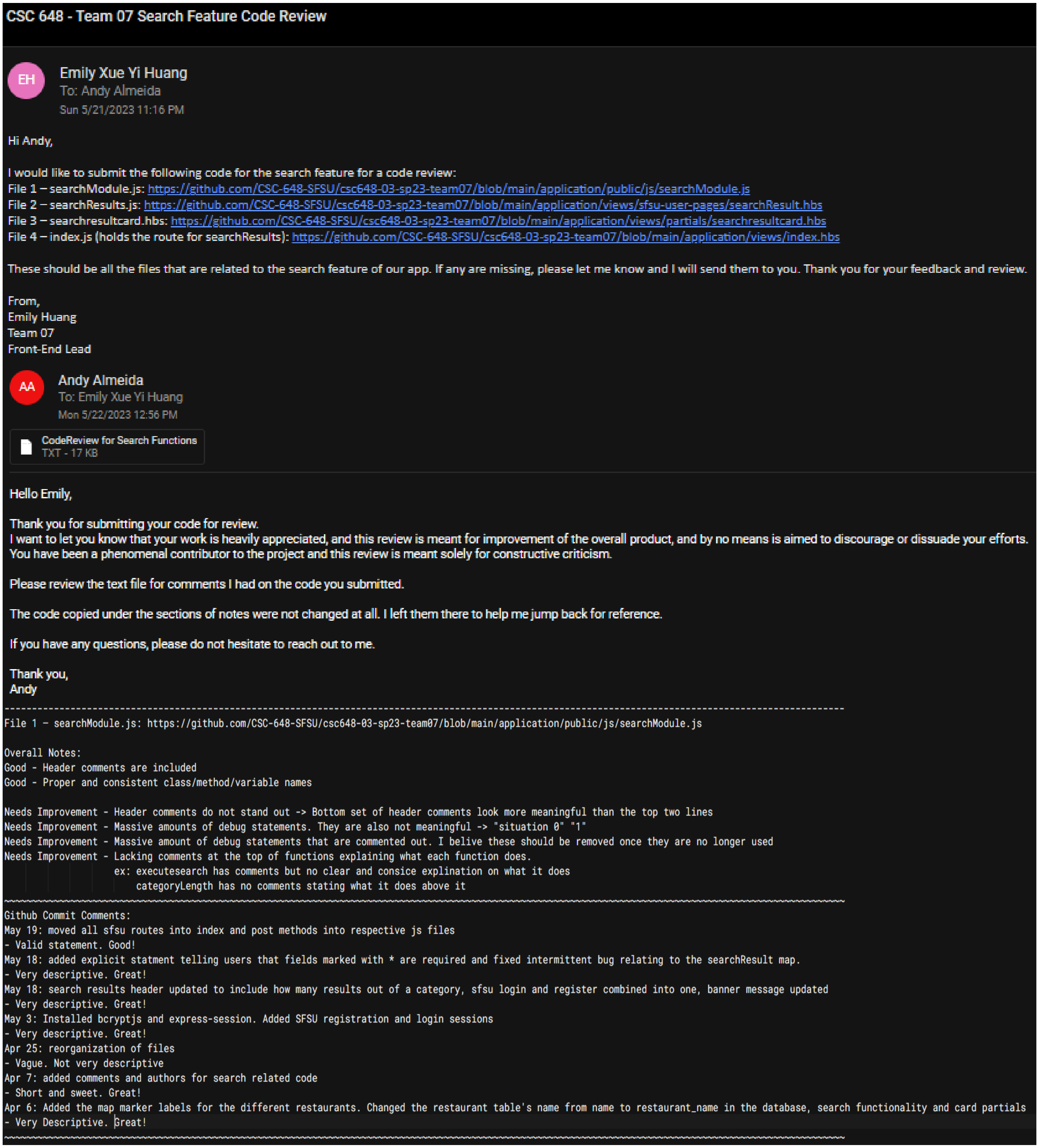
Milestone 4

QA Test plan: in table format:

Test #	Title	Description	Input	Expected Correct Output	Results (PASS/FAIL) Browser #1	Results (PASS/FAIL) Browser #2
1	General Search: Description Based	Testing the search feature that uses the description of the restaurants as the main guiding search query	In the top search bar, enter tempura, and click the search icon.	One entry shows up in the list of available restaurants: Marugame Udon	PASS	PASS
2	Category Search: Italian	Basic testing of the category based search function	In the top search bar, click the drop down labeled categories. Click the italian category. Click the search icon.	One entry shows up in the list of available restaurants: Olive Garden	PASS	PASS
3	Category Search with Input: ___ & ___	Specified testing of the category and input based search function	In the top search bar, click the drop down labeled categories. Click the pizza category. In the text box, enter “dominos”. Click the search icon.	One entry shows up in the list of available restaurants: Dominos Pizza	PASS	PASS

b) You also must <u>perform the testing</u> as per the plan above on 2 major WWW browsers of your choice and record the results in a form above (PASS or FAIL).

4) Peer Code Review



CSC 648 - Team 07 Search Feature Code Review

EH Emily Xue Yi Huang
To: Andy Almeida
Sun 5/21/2023 11:16 PM

Hi Andy,

I would like to submit the following code for the search feature for a code review:
File 1 – searchModule.js: https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/public/js/searchModule.js
File 2 – searchResults.js: https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/views/sfsu-user-pages/searchResult.hbs
File 3 – searchresultcard.hbs: https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/views/partials/searchresultcard.hbs
File 4 – index.js (holds the route for searchResults): https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/views/index.hbs

These should be all the files that are related to the search feature of our app. If any are missing, please let me know and I will send them to you. Thank you for your feedback and review.

From,
Emily Huang
Team 07
Front-End Lead

AA Andy Almeida
To: Emily Xue Yi Huang
Mon 5/22/2023 12:56 PM

CodeReview for Search Functions
TXT - 17 KB

Hello Emily,

Thank you for submitting your code for review.
I want to let you know that your work is heavily appreciated, and this review is meant for improvement of the overall product, and by no means is aimed to discourage or dissuade your efforts.
You have been a phenomenal contributor to the project and this review is meant solely for constructive criticism.

Please review the text file for comments I had on the code you submitted.

The code copied under the sections of notes were not changed at all. I left them there to help me jump back for reference.

If you have any questions, please do not hesitate to reach out to me.

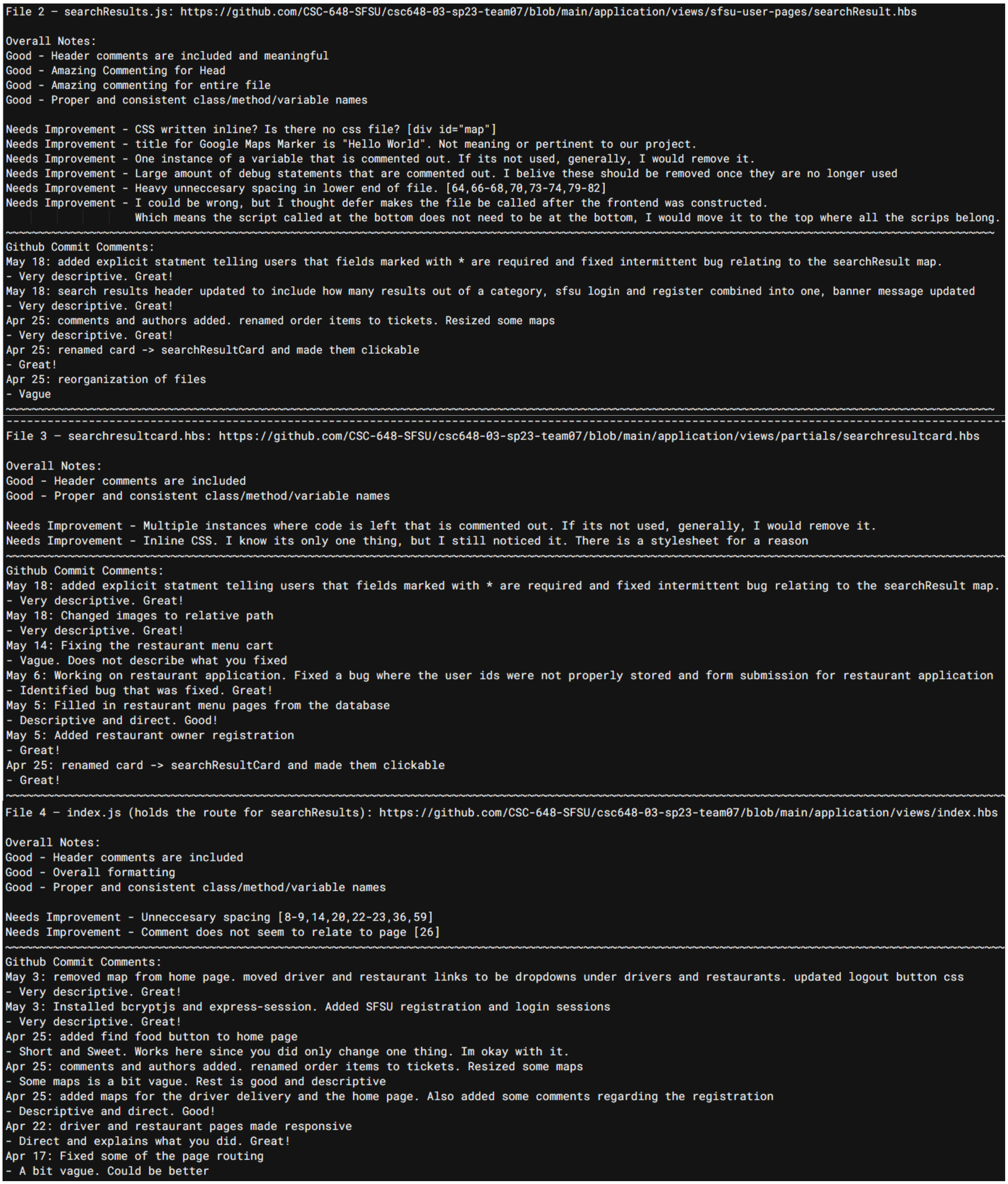
Thank you,
Andy

```
------------------------------------------------------------------------------------------------------------------------------------------------
File 1 - searchModule.js: https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/public/js/searchModule.js

Overall Notes:
Good - Header comments are included
Good - Proper and consistent class/method/variable names

Needs Improvement - Header comments do not stand out -> Bottom set of header comments look more meaningful than the top two lines
Needs Improvement - Massive amounts of debug statements. They are also not meaningful -> "situation 0" "1"
Needs Improvement - Massive amount of debug statements that are commented out. I belive these should be removed once they are no longer used
Needs Improvement - Lacking comments at the top of functions explaining what each function does.
                    ex: executesearch has comments but no clear and consice explination on what it does
                        categoryLength has no comments stating what it does above it
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Github Commit Comments:
May 19: moved all sfsu routes into index and post methods into respective js files
- Valid statement. Good!
May 18: added explicit statment telling users that fields marked with * are required and fixed intermittent bug relating to the searchResult map.
- Very descriptive. Great!
May 18: search results header updated to include how many results out of a category, sfsu login and register combined into one, banner message updated
- Very descriptive. Great!
May 3: Installed bcryptjs and express-session. Added SFSU registration and login sessions
- Very descriptive. Great!
Apr 25: reorganization of files
- Vague. Not very descriptive
Apr 7: added comments and authors for search related code
- Short and sweet. Great!
Apr 6: Added the map marker labels for the different restaurants. Changed the restaurant table's name from name to restaurant_name in the database, search functionality and card partials
- Very Descriptive. Great!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Milestone 4



```
File 2 - searchResults.js: https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/views/sfsu-user-pages/searchResult.hbs

Overall Notes:
Good - Header comments are included and meaningful
Good - Amazing Commenting for Head
Good - Amazing commenting for entire file
Good - Proper and consistent class/method/variable names

Needs Improvement - CSS written inline? Is there no css file? [div id="map"]
Needs Improvement - title for Google Maps Marker is "Hello World". Not meaning or pertinent to our project.
Needs Improvement - One instance of a variable that is commented out. If its not used, generally, I would remove it.
Needs Improvement - Large amount of debug statements that are commented out. I belive these should be removed once they are no longer used
Needs Improvement - Heavy unneccesary spacing in lower end of file. [64,66-68,70,73-74,79-82]
Needs Improvement - I could be wrong, but I thought defer makes the file be called after the frontend was constructed.
                    Which means the script called at the bottom does not need to be at the bottom, I would move it to the top where all the scrips belong.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Github Commit Comments:
May 18: added explicit statment telling users that fields marked with * are required and fixed intermittent bug relating to the searchResult map.
- Very descriptive. Great!
May 18: search results header updated to include how many results out of a category, sfsu login and register combined into one, banner message updated
- Very descriptive. Great!
Apr 25: comments and authors added. renamed order items to tickets. Resized some maps
- Very descriptive. Great!
Apr 25: renamed card -> searchResultCard and made them clickable
- Great!
Apr 25: reorganization of files
- Vague
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
File 3 - searchresultcard.hbs: https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/views/partials/searchresultcard.hbs

Overall Notes:
Good - Header comments are included
Good - Proper and consistent class/method/variable names

Needs Improvement - Multiple instances where code is left that is commented out. If its not used, generally, I would remove it.
Needs Improvement - Inline CSS. I know its only one thing, but I still noticed it. There is a stylesheet for a reason
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Github Commit Comments:
May 18: added explicit statment telling users that fields marked with * are required and fixed intermittent bug relating to the searchResult map.
- Very descriptive. Great!
May 18: Changed images to relative path
- Very descriptive. Great!
May 14: Fixing the restaurant menu cart
- Vague. Does not describe what you fixed
May 6: Working on restaurant application. Fixed a bug where the user ids were not properly stored and form submission for restaurant application
- Identified bug that was fixed. Great!
May 5: Filled in restaurant menu pages from the database
- Descriptive and direct. Good!
May 5: Added restaurant owner registration
- Great!
Apr 25: renamed card -> searchResultCard and made them clickable
- Great!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
File 4 - index.js (holds the route for searchResults): https://github.com/CSC-648-SFSU/csc648-03-sp23-team07/blob/main/application/views/index.hbs

Overall Notes:
Good - Header comments are included
Good - Overall formatting
Good - Proper and consistent class/method/variable names

Needs Improvement - Unneccesary spacing [8-9,14,20,22-23,36,59]
Needs Improvement - Comment does not seem to relate to page [26]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Github Commit Comments:
May 3: removed map from home page. moved driver and restaurant links to be dropdowns under drivers and restaurants. updated logout button css
- Very descriptive. Great!
May 3: Installed bcryptjs and express-session. Added SFSU registration and login sessions
- Very descriptive. Great!
Apr 25: added find food button to home page
- Short and Sweet. Works here since you did only change one thing. Im okay with it.
Apr 25: comments and authors added. renamed order items to tickets. Resized some maps
- Some maps is a bit vague. Rest is good and descriptive
Apr 25: added maps for the driver delivery and the home page. Also added some comments regarding the registration
- Descriptive and direct. Good!
Apr 22: driver and restaurant pages made responsive
- Direct and explains what you did. Great!
Apr 17: Fixed some of the page routing
- A bit vague. Could be better
```

5) Self-check on best practices for security

Assets to be protected	Types of possible/expected attacks	Your strategy to mitigate/protect the asset
Personal Information entered by users (Emails/Passwords)	- SQL Injection - Unwarranted Database Access	- Data Validation - Password Hashing
Personal accounts on our service	- SQL Injection - Unwarranted Database Access	- Username/Password Validation - Data Validation - Password Hashing
Business accounts on our service	- SQL Injection - Unwarranted Database Access	- Username/Password Validation - Data Validation - Password Hashing
Integrity of our publicly displayed data	- Unwarranted Database Access	- Password Security to our backend - Backend monitoring
Integrity of business' registered to our service	- Unwarranted Database Access	- Password Security to our backend - Backend Approval before it reaches Live server
Integrity of the code used to construct the service.	- Unwarranted Modification of Raw Code	- Restricted access to the instance - Private code - Github

Status	Phrase
DONE	Password encryption in database
DONE	Validation: Search bar input for up to 40 alphanumeric characters
DONE	Validation: SFSU customer registration e-mail to include "sfsu.edu" at the end

6) Self-check of the adherence to original Non-functional specs

Status	Non-Functional Requirements
DONE	Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0
DONE	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 shall render well on mobile devices
DONE	Data shall be stored in the database on the team's deployment server.
DONE	No more than 50 concurrent users shall be accessing the application at any time
DONE	Privacy of users shall be protected
DONE	The language used shall be English (no localization needed)
DONE	Application shall be very easy to use and intuitive
DONE	Application shall 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 email clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application
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	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) shall prominently display the following exact text on all pages "*SFSU Software Engineering Project CSC 648-848, Spring 2023. For Demonstration Only*" at the top of the WWW page nav bar. (Important so as to not confuse this with a real application).