SW Engineering CSC 648
Spring 2023
Team 06
Milestone 04

Justin Shin (Team Lead)
Jack Lee (Back End Lead)
Ryan Scott (Github Lead)
Konnor Nishimura (Front End Support)
Xiao Deng (Back End Support)
Alexander Scott Griffin (Front End Lead)

Date submitted: 5/24/2023

Product Summary

With our Gateway app, ordering food to your door has never been easier. Are you an SFSU student, faculty, or staff and tired of leaving the comfort of your room to meet a driver at a meet up spot? Wouldn't it be nicer if the driver would meet you at your door? Our competition simply cannot deliver the food to your door, where as we can. Enjoy your meals faster and easier than ever with our app, we'll deliver the food right to your room. On top of that we have special discounts exclusive to SFSU, meaning you not only save time and energy, but also money ordering food from our app.

Major Committed Functions (final P1 list):

- Unregistered users shall be able to search and browse for restaurants and the restaurant's information
- Unregistered users shall be able to view a restaurant's menu and pricing, and estimated time of delivery
- Unregistered users shall be able to add dishes to their shopping cart
- Unregistered users shall be able to create an account by filling in a registration from
- Unregistered users shall be displayed a map of the surrounding restaurants of the SFSU campus
- Unregistered users shall be able to search for their desired restaurants by entering text into a search bar
- Unregistered users shall be able to view restaurants of specific categories
- Unregistered users shall be required to sign up or sign in as a customer after they order
- Customers shall be required to sign up using their SFSU email
- Customers shall be able to enter the location at campus they want their order to be delivered to
- Customers shall be able to order their requested food after checking out their shopping cart
- Drivers shall be required to sign up by showing proof they are able to drive
- Drivers shall be displayed a list of orders they can deliver with delivery address on map, recipient name and phone number.
- Drivers shall be able to see specific information about an order
- Drivers shall be displayed a campus map showing pick up and drop off locations for their deliveries
- Restaurants manager shall be required to sign up by registering the restaurant's information
- Restaurants manager shall be able to enter menu items
- Restaurants manager shall be able to edit menu items

- Admins shall be required to approve all restaurant applications before they go live
- Admins shall be able to remove users
- Admins shall be able to remove restaurants
- Admins shall have access to data of all users
- Admins shall be required to approve all changes to a restaurant's menu before they go live

Our app is unique in that we allow drivers to deliver to the door. Right now, if you want to order food online you have to meet at a designated meeting point. With our app we will allow drivers to drop food off right at the door, just like if you were at home. Our delivery destination will specify which building, and room number to deliver to, as well as a campus map to help drivers along the way.

URL: http://ec2-35-164-61-115.us-west-2.compute.amazonaws.com:3000/

Usability Test Plan for Selected function

Responsibility: Alex

Test objectives:

We are testing the search functionality, to see if customers can easily find restaurants. This is the main feature of the app, and most users will interact with it. If the search wasn't easy to use, no restaurants would benefit from hosting on our app.

Test background and setup:

Setting up the system for this test will involve filling the database with several restaurants. This test will use Estrella Taqueria, Korean Tacos, Kura Thai, McDonald's, and Joyful Kitchen. The more restaurants are in the system, the better.

The starting point for this test will be the homepage of the website. Attention should be directed to the navbar at the top of the page,with search bar and category dropdown.

The intended users for this test will be those with average or below average computer skills.

URL: http://ec2-35-164-61-115.us-west-2.compute.amazonaws.com:3000/

With this test, we will measure user satisfaction with the search using several Likert scale questions, which will be answered after the test. We want to make sure customers can easily find what they came for: food.

Usability Task description:

Open Google Chrome and enter the url http://ec2-35-164-61-115.us-west-2.compute.amazonaws.com:3000/ but do not navigate to the page yet. Start a timer and immediately navigate to the page. Find the search bar. Use the search bar to find a restaurant named McDonald's, then find a restaurant selling Chinese food. Find the restaurant Estrella Taqueria. Stop the timer and record the time used for the test.

Plan for evaluation of effectiveness:

Score the user based on the number of subtasks completed. The 4 sub tasks were 1) Finding the search bar 2) Find McDonald's 3) Find Chinese restaurant 4) Find Estrella Taqueria. Each sub-task completed is worth 1 point out of a possible 4 points.

Plan for evaluation of efficiency:

We will evaluate efficiency by looking at how many searches a user had to make before finding what they were looking for. We will also watch for how far they had to scroll down search results to find what they were looking for.

Plan for Evaluation of user satisfaction:

3 Likert test questions shall be used to evaluate the user satisfaction. Users shall mark one of "Strongly agree, agree, neutral, dis-agree" or "Strongly-disagree" for all 3 of the following statements

1. The search bar was easy to use.
2. It did not take a long time to find what I was looking for.
3. The result I was looking for was near the top of the search results.

	Strongly disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**

The search bar was easy to use					
It did not take a long time to find what I was looking for					
The result I was looking for was near the top of the search results					

We will look at the averages and standard deviations of these results to determine next steps in improving the website. If the search bar has a low satisfaction metric, we need to improve the search bar.

QA Test Plan and QA Testing

Responsibility: Xiao & Alex

Test objectives

The objective of QA testing is to ensure that our product meets specified requirements. More specifically, this test ensures that one of the core functionalities of our software, searching for restaurants, performs to specs.

HW and SW setup

1. Firefox 113.0.1 on MacBook Pro 2019 running macOS Ventura 13.3.1
2. Google Chrome 113.0.5672.126 on Lenovo ThinkPad T570 running Ubuntu 22.04.2
3. Test URL: http://ec2-35-164-61-115.us-west-2.compute.amazonaws.com:3000/

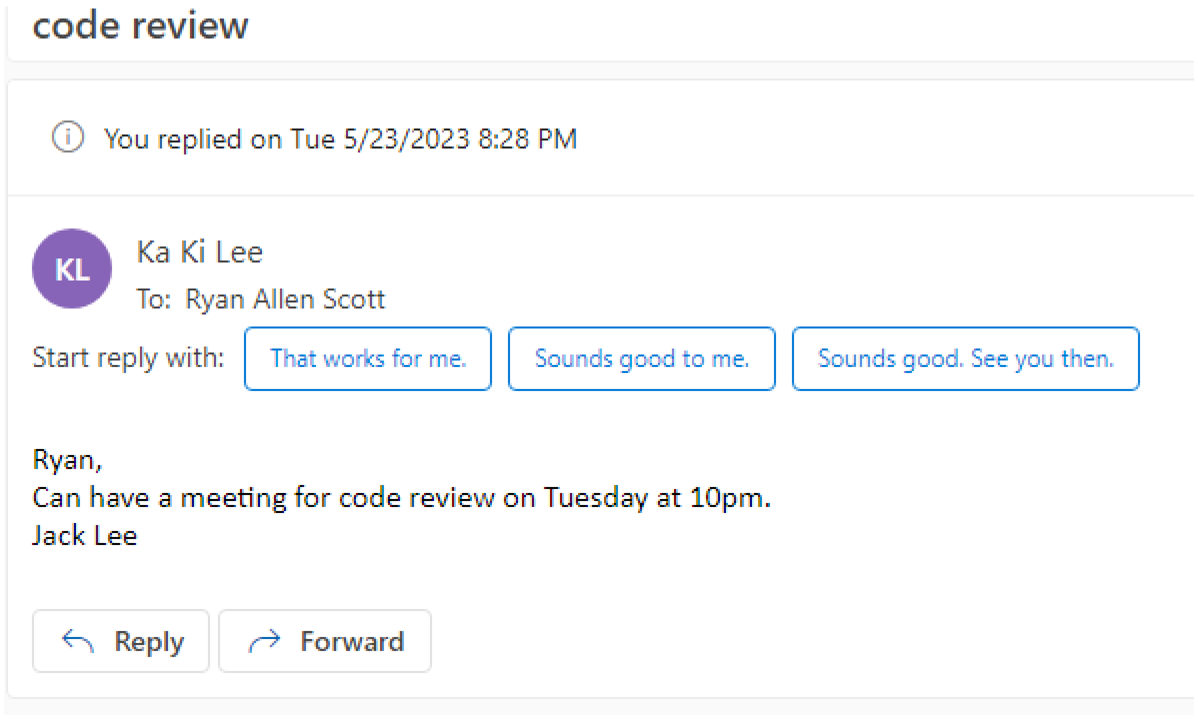
Feature to be tested

Searching for restaurants using category dropdown menu and text

QA Test plan

Test title	Test description	Test #	Test input	Expected correct output	Test results (Firefox)	Test results (Chrome)
search	Test %like search with category dropdown and text	1	Category: fast food Text field: McDonald's	Restaurant named "McDonald's" displayed, along with its logo and dishes	Pass	Pass
		2	Category: Chinese Text field: Joyful Kitchen	Restaurant named "Joyful Kitchen" displayed, along with its logo and dishes	Pass	Pass
		3	Category: All Text field: empty	All restaurants are displayed	Pass	Pass
		4	Category: Chinese Text field: Panda Express	No restaurants are displayed	Pass	Pass
		5	Category: Mexican Text field: Empty	Restaurant "Estrella Taqueria" is displayed	Pass	Pass

Peer Code Review
Responsibility: Ryan, Jack



Ryan,
Can have a meeting for code review on Tuesday at 10pm.
Jack Lee

Re: code review

RS

Ryan Allen Scott
To: Ka Ki Lee

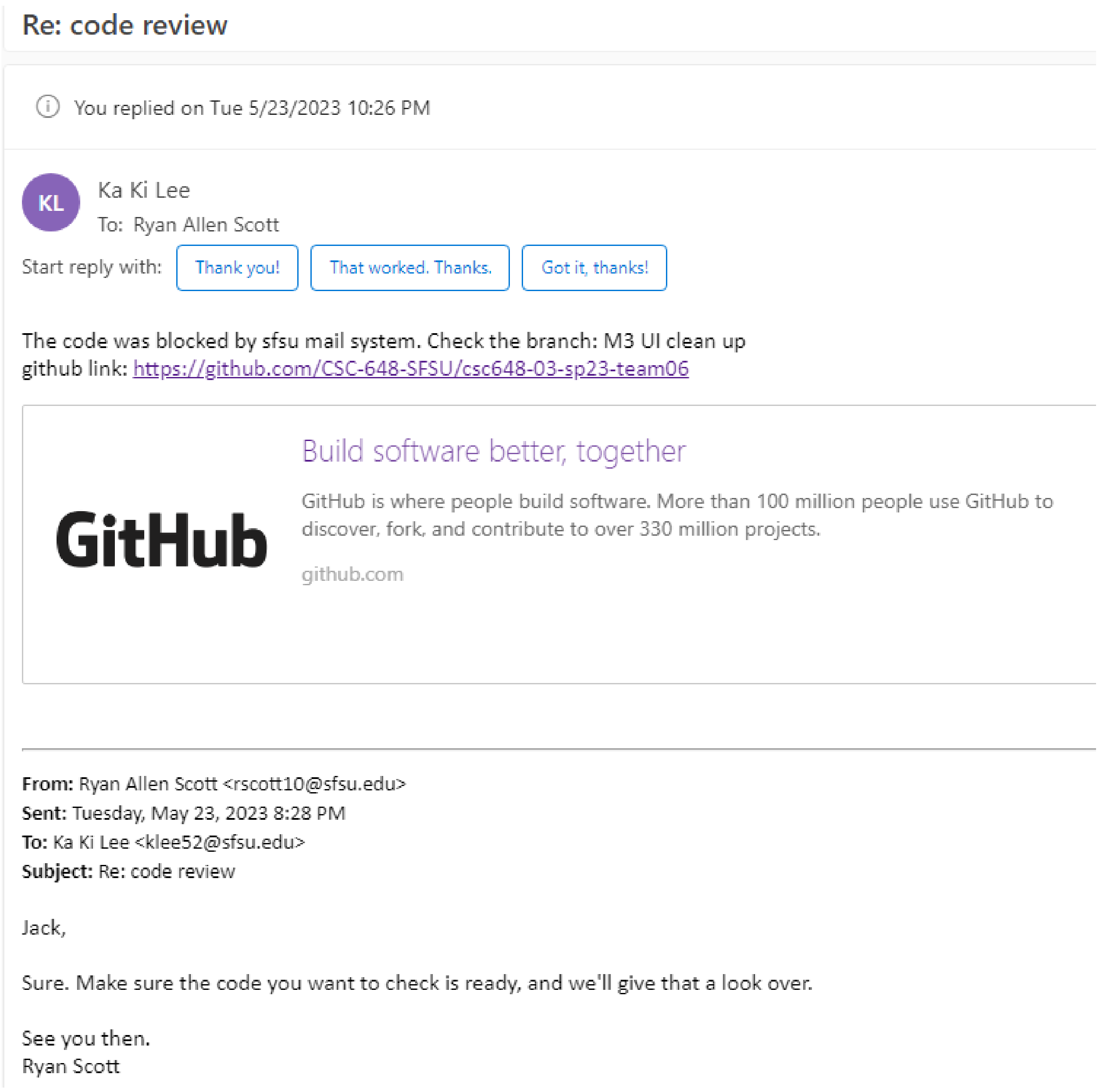
Jack,

Sure. Make sure the code you want to check is ready, and we'll give that a look over.

See you then.
Ryan Scott

From: Ka Ki Lee <klee52@sfsu.edu>
Sent: Tuesday, May 23, 2023 8:26 PM
To: Ryan Allen Scott <rscott10@sfsu.edu>
Subject: code review

Ryan,
Can have a meeting for code review on Tuesday at 10pm.
Jack Lee



Re: code review

You replied on Tue 5/23/2023 10:26 PM

KL Ka Ki Lee

To: Ryan Allen Scott

Start reply with:

The code was blocked by sfsu mail system. Check the branch: M3 UI clean up
github link: https://github.com/CSC-648-SFSU/csc648-03-sp23-team06

GitHub

Build software better, together

GitHub is where people build software. More than 100 million people use GitHub to discover, fork, and contribute to over 330 million projects.

github.com

From: Ryan Allen Scott <rscott10@sfsu.edu>
Sent: Tuesday, May 23, 2023 8:28 PM
To: Ka Ki Lee <klee52@sfsu.edu>
Subject: Re: code review

Jack,

Sure. Make sure the code you want to check is ready, and we'll give that a look over.

See you then.
Ryan Scott

Re: code review

Ryan Allen Scott
To: Ka Ki Lee

Tue 5/23/2023 10:26 PM

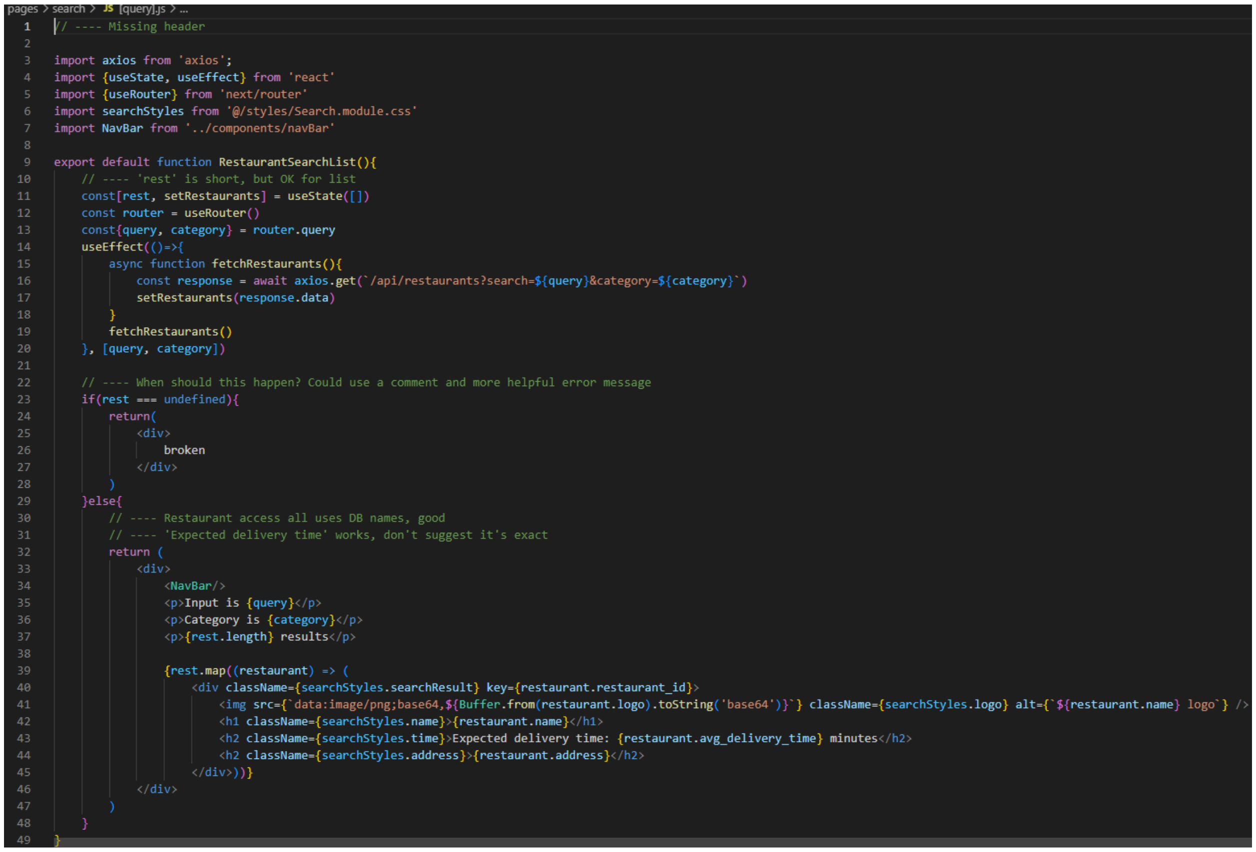
Jack,

Looks mostly good. I was checking [query].js, but also gave a look to the API route it was using, restaurants.js. Overall readable, and fairly intuitive flow with variable names. [query].js is missing a header, though, and I think some comments can help clear up what some edge cases are checking for. Functions to spec, so those comments are about all it might need.

Ryan Scott

From: Ka Ki Lee <klee52@sfsu.edu>
Sent: Tuesday, May 23, 2023 8:44 PM
To: Ryan Allen Scott <rscott10@sfsu.edu>
Subject: Re: code review

The code was blocked by sfsu mail system. Check the branch: M3 UI clean up
github link: https://github.com/CSC-648-SFSU/csc648-03-sp23-team06


pages > search > JS [query].js > ...

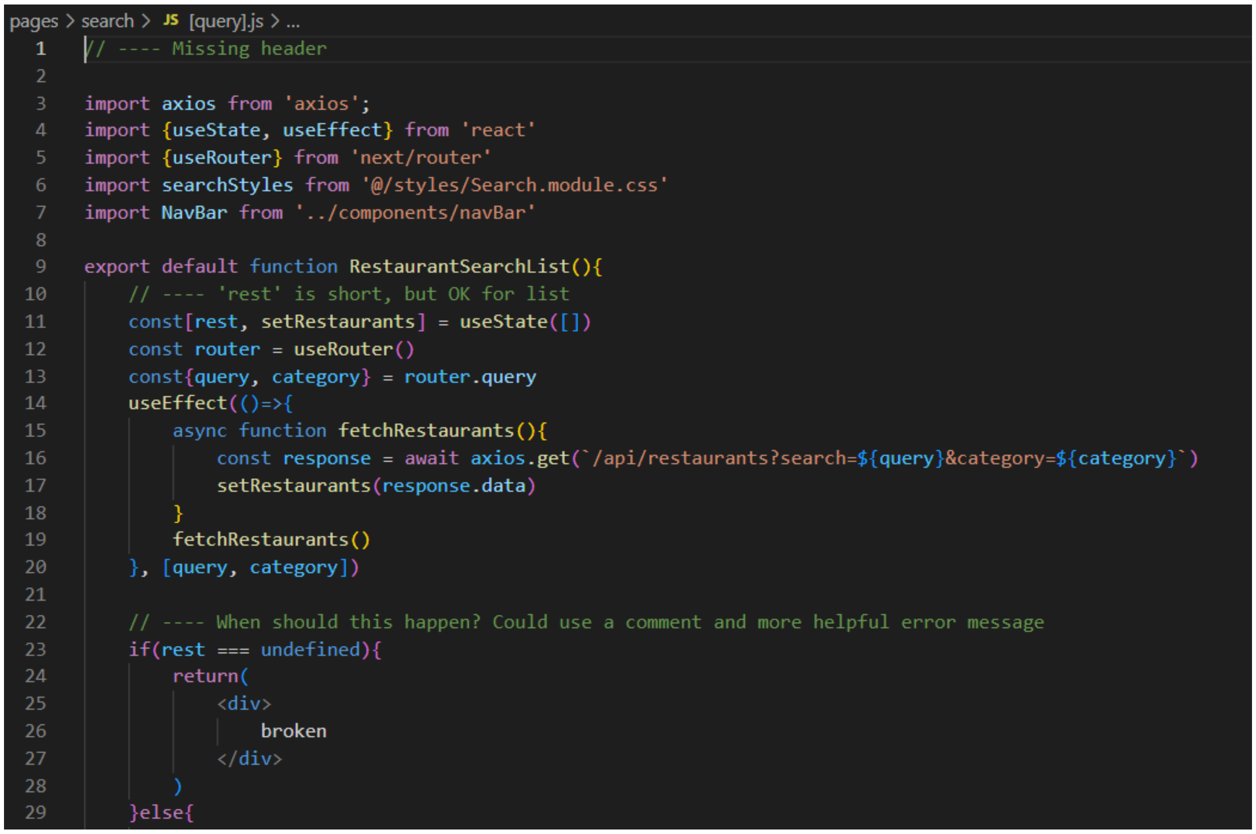
```
// ---- Missing header

import axios from 'axios';
import {useState, useEffect} from 'react'
import {useRouter} from 'next/router'
import searchStyles from '@/styles/Search.module.css'
import NavBar from '../components/navBar'

export default function RestaurantSearchList(){
    // ---- 'rest' is short, but OK for list
    const[rest, setRestaurants] = useState([])
    const router = useRouter()
    const{query, category} = router.query
    useEffect(()=>{
        async function fetchRestaurants(){
            const response = await axios.get(`/api/restaurants?search=${query}&category=${category}`)
            setRestaurants(response.data)
        }
        fetchRestaurants()
    }, [query, category])

    // ---- When should this happen? Could use a comment and more helpful error message
    if(rest === undefined){
        return(
            <div>
                broken
            </div>
        )
    }else{
        // ---- Restaurant access all uses DB names, good
        // ---- 'Expected delivery time' works, don't suggest it's exact
        return (
            <div>
                <NavBar/>
                <p>Input is {query}</p>
                <p>Category is {category}</p>
                <p>{rest.length} results</p>

                {rest.map((restaurant) => (
                    <div className={searchStyles.searchResult} key={restaurant.restaurant_id}>
                        <img src={`data:image/png;base64,${Buffer.from(restaurant.logo).toString('base64')}`} className={searchStyles.logo} alt={`${restaurant.name} logo`} />
                        <h1 className={searchStyles.name}>{restaurant.name}</h1>
                        <h2 className={searchStyles.time}>Expected delivery time: {restaurant.avg_delivery_time} minutes</h2>
                        <h2 className={searchStyles.address}>{restaurant.address}</h2>
                    </div>))}
            </div>
        )
    }
}
```

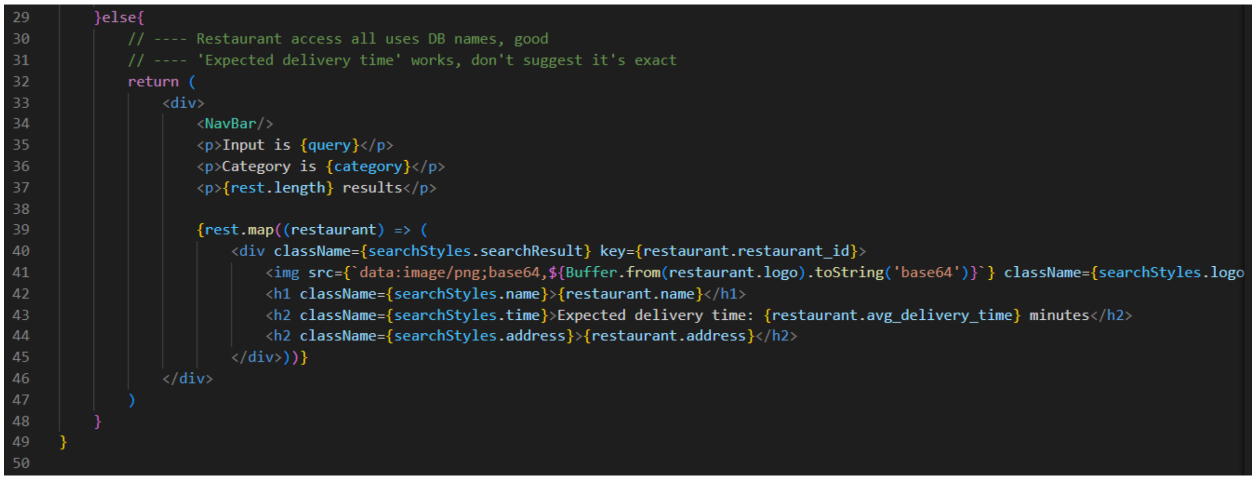

pages > search > JS [query].js > ...

```
// ---- Missing header

import axios from 'axios';
import {useState, useEffect} from 'react'
import {useRouter} from 'next/router'
import searchStyles from '@/styles/Search.module.css'
import NavBar from '../components/navBar'

export default function RestaurantSearchList(){
    // ---- 'rest' is short, but OK for list
    const[rest, setRestaurants] = useState([])
    const router = useRouter()
    const{query, category} = router.query
    useEffect(()=>{
        async function fetchRestaurants(){
            const response = await axios.get(`/api/restaurants?search=${query}&category=${category}`)
            setRestaurants(response.data)
        }
        fetchRestaurants()
    }, [query, category])

    // ---- When should this happen? Could use a comment and more helpful error message
    if(rest === undefined){
        return(
            <div>
                broken
            </div>
        )
    }else{
```



```
    }else{
        // ---- Restaurant access all uses DB names, good
        // ---- 'Expected delivery time' works, don't suggest it's exact
        return (
            <div>
                <NavBar/>
                <p>Input is {query}</p>
                <p>Category is {category}</p>
                <p>{rest.length} results</p>

                {rest.map((restaurant) => (
                    <div className={searchStyles.searchResult} key={restaurant.restaurant_id}>
                        <img src={`data:image/png;base64,${Buffer.from(restaurant.logo).toString('base64')}`} className={searchStyles.logo
                        <h1 className={searchStyles.name}>{restaurant.name}</h1>
                        <h2 className={searchStyles.time}>Expected delivery time: {restaurant.avg_delivery_time} minutes</h2>
                        <h2 className={searchStyles.address}>{restaurant.address}</h2>
                    </div>))}
            </div>
        )
    }
}
```

Self-Check on Best Practices for Security

Asset to be protected	**Types of possible/expected attacks**	**Your strategy to mitigate/protect the asset**
Account passwords	Passwords are compromised	Passwords will be encrypted using the crypto library

Database information	1. SQL injection 2. XSS attack	1. Queries will use SQL parameters which prevent SQL injection 2. Inputs for sign up, login, search bar, and adding dishes have a maximum length
Restaurant pages	Inappropriate content is posted	Restaurants listed must be approved by an admin
User cookies	Attacker accesses a user's cookies	Cookies will be secure and have a password to encrypt the cookie
Service access	Non-SFSU customer tries to make an account	Emails in customer registration must end in sfsu.edu

Self-Check of the adherence to original Non-Functional Specs

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0
   - DONE
2. Application shall be optimized for standard desktop/laptop browser e.g. must render correctly on the two latest versions of two major browsers
   - DONE
3. All or selected application functions shall render well on mobile devices
   - DONE
4. Data shall be stored in the database on the team's deployment server.
   - DONE
5. No more than 50 concurrent users shall be accessing the application at any time
   - Issue: unsure of how to run 50 concurrent users
6. Privacy of users shall be protected
   - DONE
7. The language shall be English
   - DONE
8. Application shall be very easy to use and intuitive
   - DONE

9. Application shall follow established architecture patterns
   - DONE
10. Application code and its repository shall easy to inspect and maintain
   - DONE
11. Google analytics shall be used
   - Issue: incomplete
12. No email clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging is enough for this application
   - DONE
13. Pay functionality, if any, shall not be implemented nor simulated in UI.
   - DONE
14. Site security: basic best practices shall be applied for main data items.
   - DONE
15. Media formats shall be standard as used in the market today
   - DONE
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
   - DONE
17. The application UI 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.
   - DONE