CSC648-848 Software Engineering Spring 2021

HungryGators-19

Food Delivery and Review Service

Available on Mobile and Desktop

Team 01

Branden J. Nguyen - Team Lead - bnguye11@mail.sfsu.edu

Jasmine Jahan - Front End Lead

Yongjian Pan - Back End Lead

Robert Freeman - Github Master

Gurjot Singh - Team Member Front End

Tin Thu Zar Aye - Team Member Back End

Milestone 4

Date Submitted	Date Revised
05/14/21	

Table of Contents

I. Product summary

- Hungry Gators-19 is the name of our application.
- Homepage: This is the main page of the website which will guide you to other pages.
- Registration: This feature takes you to the Registration page where you can create an account if you don't have one with us.
- Login (by default SFSU students/staff/faculty): There are three different logins for Driver, Restaurant owner, and student/staff. You have to login to checkout and restaurant owners have to login to add items on to the website.
- Search Bar: This feature helps you to search anything you are looking for on the website.
- Drop down tab: This feature includes different restaurants you can visit.
- Restaurant Menus : This feature takes you to the menus based on the restaurant you pick.
- Cart: This feature helps you to check out all the food items you picked.
- Driver Pickup, map of the campus: This feature is for the drivers where drivers look up for the orders and the map is provided for them so they can drop off the order.
- Upload Restaurant: This feature is for the restaurant owner so they can upload the items they have available for their restaurant.
- **(Link to the webpage) http://18.223.160.96:5002/**

II. Usability test plan

- Test objectives:
  - Search: This feature helps search for restaurants and cuisines by entering names or menu.
  - DropDown: This feature displays all the cuisine options which you can use to search
  - Search Bar and Cuisine Dropdown: This feature can be used for both the dropdown & a keyword.

- Test background and setup:
  - System setup: we are using pycharm locally and then we push it to Amazon AWS to deploy our application.
  - Starting point: Home page
  - Intended Users: SFSU students/staff, drivers, and restaurant owners.

- Usability Task description:
  - Making sure that the search function is working correctly and that genuine results are returned.
  - Guarantee that the function is simple to utilize across a wide range of search scenarios.
  - Making sure that results from the search bar and/or the cuisine drop-down menu adjacent to the search bar are included in the feature.

- Evaluation of Effectiveness: The Search bar is working effectively. You should see all the results deployed. It should not show an error if you are using more than 40 alphanumeric characters and the dropdown should be working fine.

- Evaluation of efficiency: The search works good as far as I tested. It is showing all the results I am looking for. The dropdown cuisine works fine.

- Evaluation of user satisfaction: As a user it was simple for me to use this application for searching different cuisines and restaurants. I did not face any difficulty using it. All the steps were clear.

III. QA Test Plan

- Test Objectives

  The main purpose of the quality assurance test plan is to make sure that the search bar is working perfectly as expected. To ensure that the search bar has a persistent pull-down menu to notify users of what filters are active. Moreover, after clicking the search button, the number of search results found should be displayed. Not only can the user search the restaurant by typing the restaurant name in the search bar but the user can also search the restaurant by filtering the cuisines.

- HW and SW setup (including URL)

  Server host: Amazon AWS Linux CPU 5 GB RAM

  Operating System : macOS Big Sur 11.2.1

  Web Browser : Safari, Latest Version of Google Chrome

  Database: MySQL Workbench80

  Web Server: NGINX 1.19.6

  Server-Side Language: Python

  Additional Technologies: Web Framework: Flask IDE: PyCharm

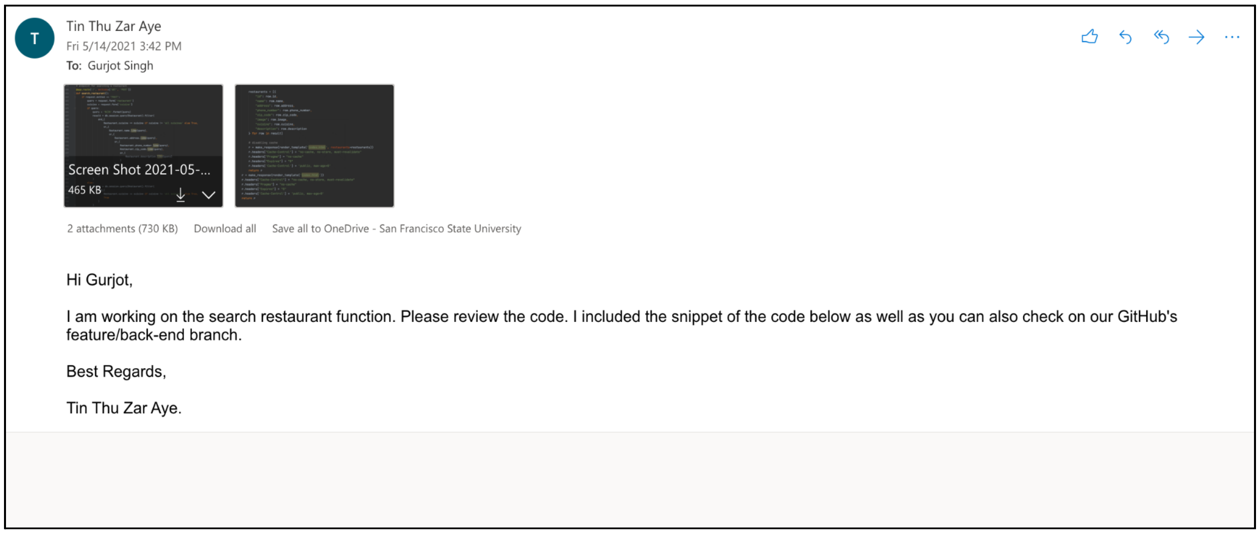
- Feature to be tested

In this quality assurance test plan, the feature to be tested is that the search field should only accept a maximum of 40 alphanumeric characters. If the user enters the over 40 alphanumeric characters, it should display the error message that is telling the users that they are entering the invalid query.

Test#	Title	Description	Input	Output	Result
1	Display all the restaurants' list	Click the search button which is in the right corner of the search bar to display all the restaurants.	User doesn't need to input anything to see all the lists of the restaurant. Users only need to click the search button from the search bar.	Display all the lists of the restaurant.	PASS (Safari) PASS (Google Chrome)
2	Filtering the Cuisines	There is a "all cuisines" drop-down search box left corner of the search bar. Users need to choose the specific cuisine to see the list of the specific cuisine's restaurant.	Users choose the specific cuisine from the "All Cuisines" drown down bar. Then the user needs to click the search button.	Display all the lists of the specific cuisine that the user searches for.	PASS (Safari) PASS (Google Chrome)
3	Search the restaurant with the restaurant name	In the search bar, user can type the restaurant to find the specific restaurant	User can type the restaurant name in the search then click the search button	Display the specific restaurant.	PASS (Safari) PASS (Google Chrome)

IV. Code Review

One team member sent the email to another team member to review the code.


T

Tin Thu Zar Aye
Fri 5/14/2021 3:42 PM
To: Gurjot Singh

Screen Shot 2021-05-...
465 KB

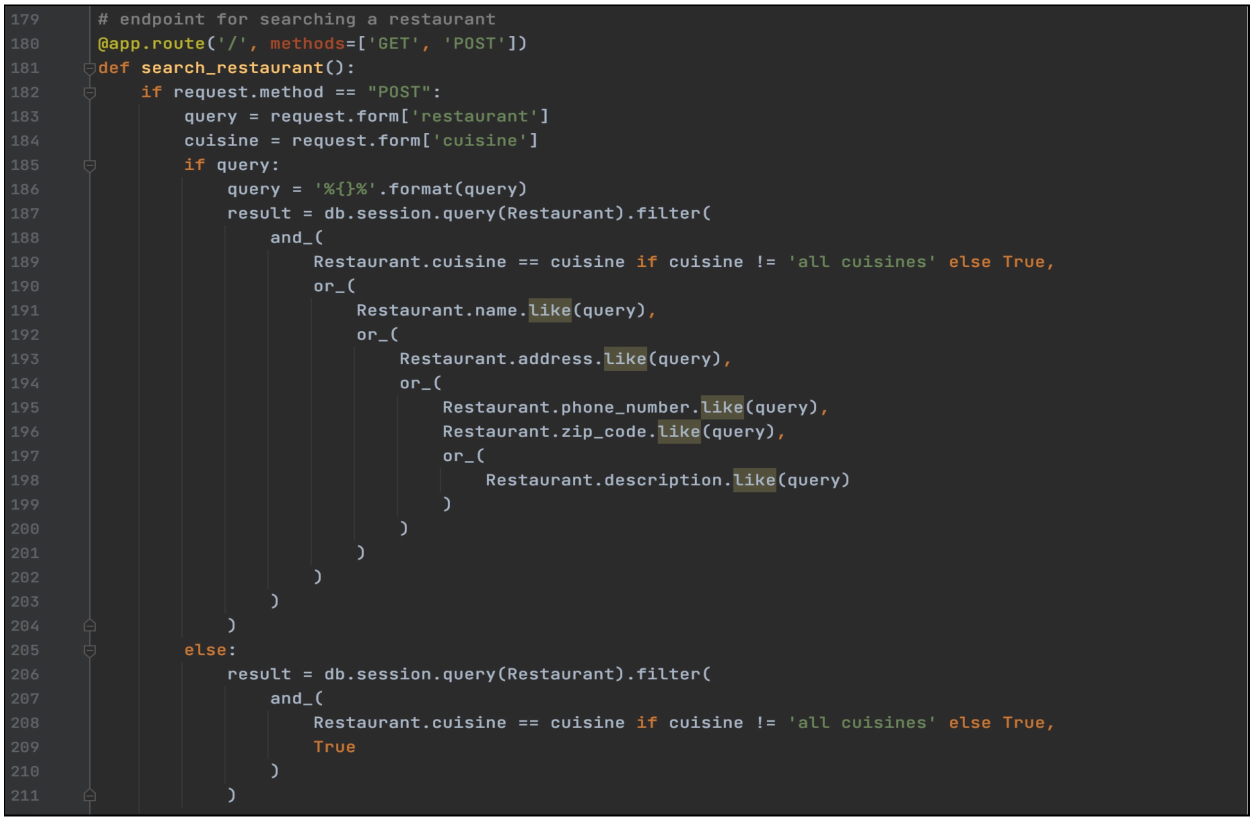
2 attachments (730 KB) Download all Save all to OneDrive - San Francisco State University

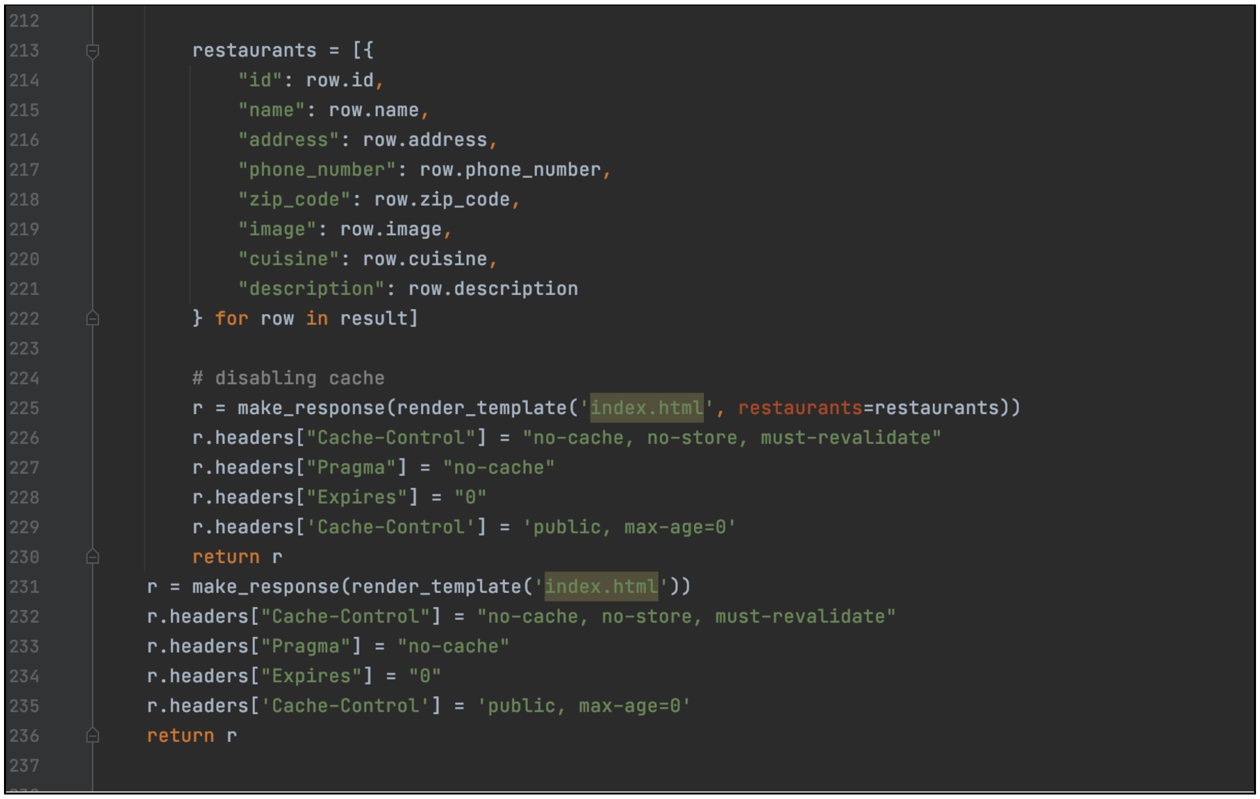
Hi Gurjot,

I am working on the search restaurant function. Please review the code. I included the snippet of the code below as well as you can also check on our GitHub's feature/back-end branch.

Best Regards,

Tin Thu Zar Aye.

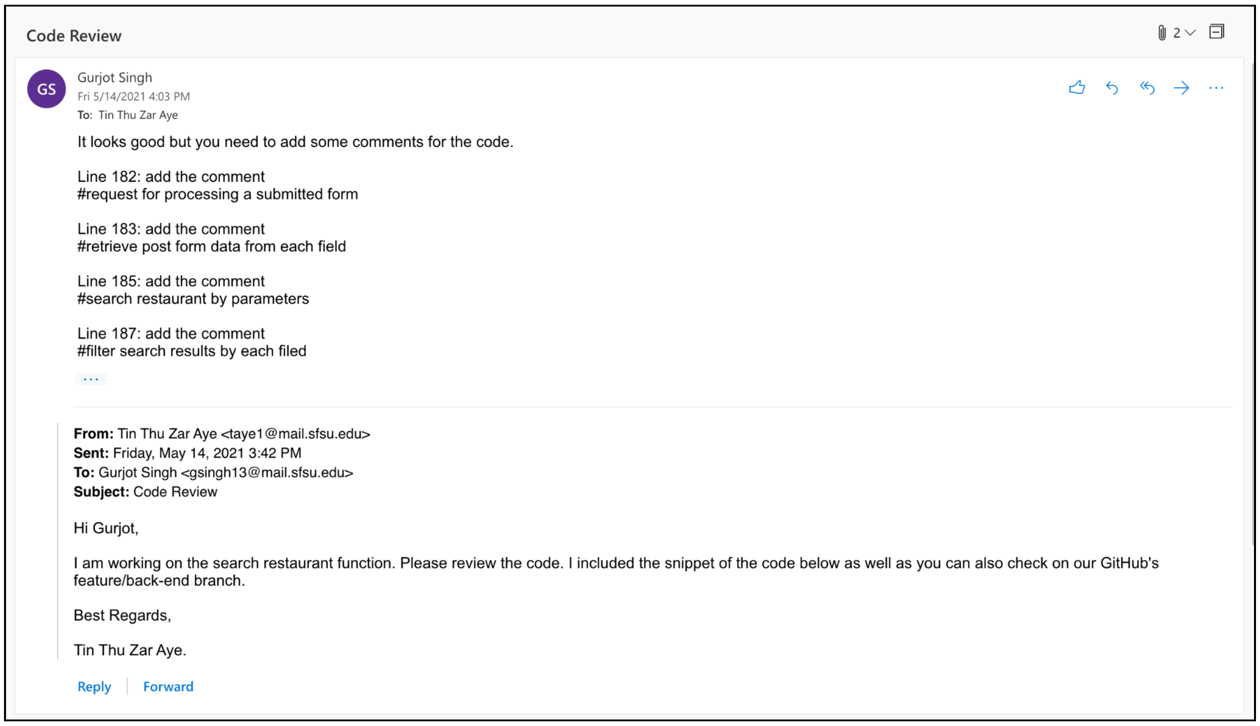

```
# endpoint for searching a restaurant
@app.route('/', methods=['GET', 'POST'])
def search_restaurant():
    if request.method == "POST":
        query = request.form['restaurant']
        cuisine = request.form['cuisine']
        if query:
            query = '%{}%'.format(query)
            result = db.session.query(Restaurant).filter(
                and_(
                    Restaurant.cuisine == cuisine if cuisine != 'all cuisines' else True,
                    or_(
                        Restaurant.name.like(query),
                        or_(
                            Restaurant.address.like(query),
                            or_(
                                Restaurant.phone_number.like(query),
                                Restaurant.zip_code.like(query),
                                or_(
                                    Restaurant.description.like(query)
                                )
                            )
                        )
                    )
                )
            )
        else:
            result = db.session.query(Restaurant).filter(
                and_(
                    Restaurant.cuisine == cuisine if cuisine != 'all cuisines' else True,
                    True
                )
            )
```



```
        restaurants = [{
            "id": row.id,
            "name": row.name,
            "address": row.address,
            "phone_number": row.phone_number,
            "zip_code": row.zip_code,
            "image": row.image,
            "cuisine": row.cuisine,
            "description": row.description
        } for row in result]

        # disabling cache
        r = make_response(render_template('index.html', restaurants=restaurants))
        r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
        r.headers["Pragma"] = "no-cache"
        r.headers["Expires"] = "0"
        r.headers['Cache-Control'] = 'public, max-age=0'
        return r
    r = make_response(render_template('index.html'))
    r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    r.headers["Pragma"] = "no-cache"
    r.headers["Expires"] = "0"
    r.headers['Cache-Control'] = 'public, max-age=0'
    return r
```

Another Team member reviewed the code and made the comments.



Code Review

GS
Gurjot Singh
Fri 5/14/2021 4:03 PM
To: Tin Thu Zar Aye

It looks good but you need to add some comments for the code.

Line 182: add the comment
#request for processing a submitted form

Line 183: add the comment
#retrieve post form data from each field

Line 185: add the comment
#search restaurant by parameters

Line 187: add the comment
#filter search results by each filed

...

From: Tin Thu Zar Aye <taye1@mail.sfsu.edu>
Sent: Friday, May 14, 2021 3:42 PM
To: Gurjot Singh <gsingh13@mail.sfsu.edu>
Subject: Code Review

Hi Gurjot,

I am working on the search restaurant function. Please review the code. I included the snippet of the code below as well as you can also check on our GitHub's feature/back-end branch.

Best Regards,

Tin Thu Zar Aye.

Reply | Forward

V. Self-check on best practices for security

Asset to be protected	Type of possible/expected attacks	Strategy to mitigate/protect the assets
Registration for user and delivery driver’s email to include “sfsu.edu”	User can’t register if they don’t have “sfsu.edu” email address	If the user/delivery driver tries to register with the email which is not ending with sfsu.edu,it is going to get the error.
Password confirmation for the registration	User can’t register if they enter the different password for the “confirm password” field when they register	Password will be saved as encrypted string
User Login	User can’t login if they don’t have “sfsu.edu” email	To order the food from the restaurant, user require to login with their “sfsu-edu” email address
Delivery Driver Login	Delivery Driver can’t login if they don’t have “sfsu.edu” email	To see the order detail and where to deliver the order, delivery driver need to login with the “sfsu.edu” email address
Search bar input validation	Over 40 alphanumeric characters will display the error message	User needs to input up to 40 valid alphanumeric character which does not include special characters

VI. Self-check: Adherence to original Non-functional specs

<u>High-level non-functional specifications (how the app is delivered and other constraints) that MUST be adhered to (not negotiable)</u>

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0. Application delivery shall be from chosen cloud server
ON TRACK

2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers
ON TRACK

3. All or selected application functions must render well on mobile devices (specifics to be developed in consultation with users e.g. Petkovic)
DONE

4. Ordering and delivery of food shall be allowed only for SFSU students, staff and faculty
ON TRACK

5. Data shall be stored in the database on the team's deployment cloud server.
ON TRACK

6. No more than 50 concurrent users shall be accessing the application at any time
DONE

7. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
ON TRACK

8. The language used shall be English (no localization needed)
DONE

9. Application shall be very easy to use and intuitive
ON TRACK

10. Application should follow established architecture patterns
ON TRACK

11. Application code and its repository shall be easy to inspect and maintain
ON TRACK

12. Google analytics shall be used
ON TRACK

13. No e-mail clients shall be allowed.
DONE

14. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.
DONE

15. Site security: basic best practices shall be applied (as covered in the class) for main data items
ON TRACK

16. Application shall be media rich (images, maps etc.). Media formats shall be standard as used in the market today
ON TRACK

17. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
ON TRACK

18. The application UI (WWW and mobile) 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).
ON TRACK