SW Engineering CSC648 Spring 2021

Gator Grub - Team02

Team Member	Role
Patricia Louise Sarno	Team Lead | psarno@mail.sfsu.edu
Erik Chacon	GitHub Master
Danny Collan	Front End Lead
Saloni Mahat	Front End Member
Affaan Ghazzali	Back End Lead
Edmund-Russel Manzano	Back End Member

Milestone 4

16 May 2021

History Table	
Date Submitted	**Date Revised After Instructor(s) Comments**
16 May 2021	

1. **Product Summary**

Name of Product: Gator Grub

Explicit Itemized List Of All Major Committed Functions (P1 Function List):

SFSU Customers

1. SFSU Customers shall have the ability to search food through different categories such as cuisines and restaurants.
2. SFSU Customers shall have the ability to register and log in.
3. SFSU Customers shall have the ability to choose food from the menu of specific restaurants.
4. SFSU Customers shall have the ability to specify the location of delivery in SFSU campus.
5. SFSU Customers shall have the ability to choose delivery or pick up.
6. SFSU Customers shall receive notifications on the update of their driver.
7. SFSU Customers shall be able to specify certain options through comments on an order (e.g. no onions on a burger). Restaurants will be held to their comments and these comments will be displayed as part of the order, rather than something extra.
8. Unregistered users shall be required to register before placing an order.

   Restaurant Owner
9. Restaurant owners shall have the ability to register their restaurants into the application.

   Delivery Driver
10. Drivers will be required to register their information for a specific restaurant before becoming a driver.
11. Drivers shall be able to access order to deliver.

URL to Product: http://ec2-54-215-253-212.us-west-1.compute.amazonaws.com

2. **Usability Test Plan: Search Function**

Test Objectives:

The objective of our Usability Test Plan for the search function is to evaluate if the feature is usable and point to problems that may arise when completing comprehensive test plans. Our team began developing the search function by applying user centered design (UCD) from the prototyping stages. This test plan will provide quantitative results in regards to how well our team adhered to USD by way of evaluating effectiveness, efficiency, and user satisfaction. Effectiveness will be evaluated by the completion rate of our usability task, efficiency will be evaluated by way of how long our usability task takes, and user satisfaction will be evaluated by way of three Lickert scale questions. These three metrics are being evaluated because the International Organization for Standardization (ISO) points to these as key usability metrics which provide insight into the complete usability of the feature. This usability test is a validation test, which is being performed late in our development cycle, to certify the search feature's product usability and provide 'disaster insurance' against launching and presenting a poor product.

Test Background and Set Up:

System Setup: The system setup for our User Test Plan is fairly straightforward, as the development team will simply provide the latest working version of the system. This is to ensure that the development team receives complete and uninfluenced feedback on all the work that has been done thus far. Our team has been active in following UCD throughout the development process, and as a result there will be no specific preparations made for the Usability Tests that could potentially change the results of our measured metrics.

Starting Point: The starting point for our Usability Test Plan will be the home page of our application. Here, the development and testing teams will only cover what the user will do and not how they do it. This will potentially uncover usability flaws in our system, given that the users will have to figure out how to execute the usability tasks set to them, rather than follow a help document informing them on the 'correct' way to execute the task. In our task description, we have selected the most frequently occurring scenarios that may occur in the usage of the search function as this will expose our system's potential to flaws and unusable features.

Intended Users: The intended users for our system, and more specifically, the search function are SFSU students, faculty, and other staff. Given that a majority of our users will not have a technical background, these tests serve to help the development team catch usability flaws that may be too 'low level' from our perspective.

Tested System URL:
http://ec2-54-215-253-212.us-west-1.compute.amazonaws.com

Measured Metrics: The metrics that we will be evaluating through the Usability Test Plan will be Effectiveness, Efficiency, and User Satisfaction. These are three usability metrics that the ISO recommends each Usability Test plan has as they will cover any and all potential usability pitfalls. From the results of these tests, the development team can analyze the data collected and proceed to take action to rectify any issues that may arise. This process will be iterative as the developers will revise the product, iterate on any design changes necessary, and then proceed to retest. This usability test will be a validation test which is being done at the end of our team's development cycle. We chose to go this route, as throughout development our team has taken measures to ensure that the system was developed with UCD in mind. And as a result, this test will provide adequate data on whether the system is usable for our intended users. And in conjunction with our QA test plan, will ensure that our system is 'disaster' safe and well designed for our users.

Usability Task Description:

1. Test the search bar for invalid entries (Ex: “aiciezmsl”, “9ankl#^%”).
2. Test the search bar for entries that are too long. The maximum number of characters allowed is 40.
3. Search with the drop down select menu.
4. Search for restaurants or cuisines with the search bar.
5. Select a cuisine from the drop down select menu AND type a valid entry into the search bar. Make sure the entries are different.
6. Search for all restaurants.
7. Search for specific restaurants.

Evaluation of Effectiveness:

- We would measure it by testing the search function. We would collect the percentage of people who completed the task (success rate) and number of errors.

Evaluation of Efficiency:

- We would measure efficiency by calculating how long it takes to finish the task of searching (end time - start time) and count how many clicks the user does within that time of task

Evaluation of User Satisfaction:

1. The results from the search function were helpful (check one)

o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree

Comments/Suggestions

2. I found the search function easy to use (check one)

o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree

Comments/Suggestions

3. The search box is appropriately placed where I easily noticed it (check one)

o Strongly Disagree o Disagree o Neutral o Agree o Strongly Agree

Comments/Suggestions

3. QA Test Plan: Search Function

Test Objectives: The objective of the QA test plan is to test if our search function is performing correctly without any bugs. By using the search bar and the cuisines drop down, we will type in and choose a variety of search inputs and observe if the results prove to be reliable and produce the correct outputs.

HW and SW Set Up: The tests will be done on our deployment server which has the up to date version of our search function. The tester should go to this URL to perform the tests: http://ec2-54-215-253-212.us-west-1.compute.amazonaws.com

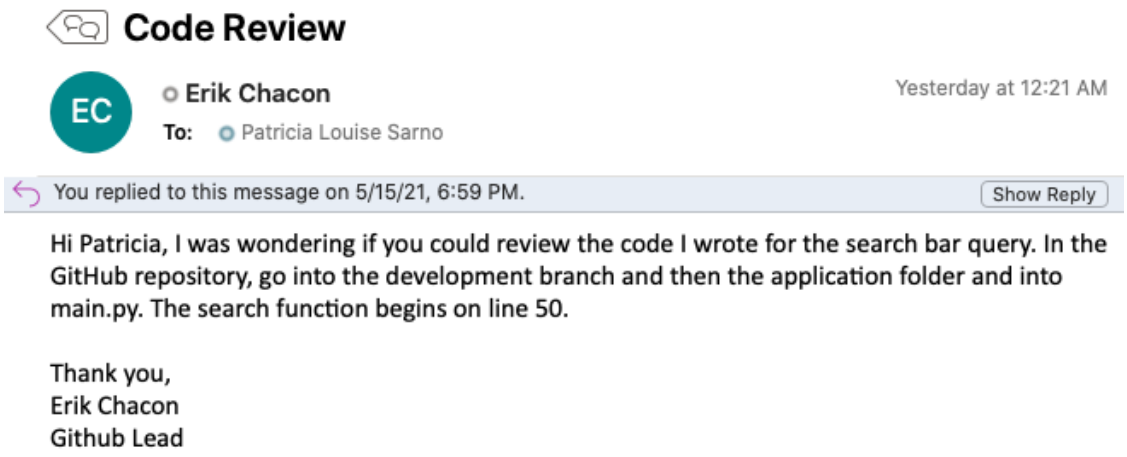
Feature to Be Tested: The features to be tested are the food type drop down and the search bar located at the nav bar.

QA Test Plan: In Table Format:

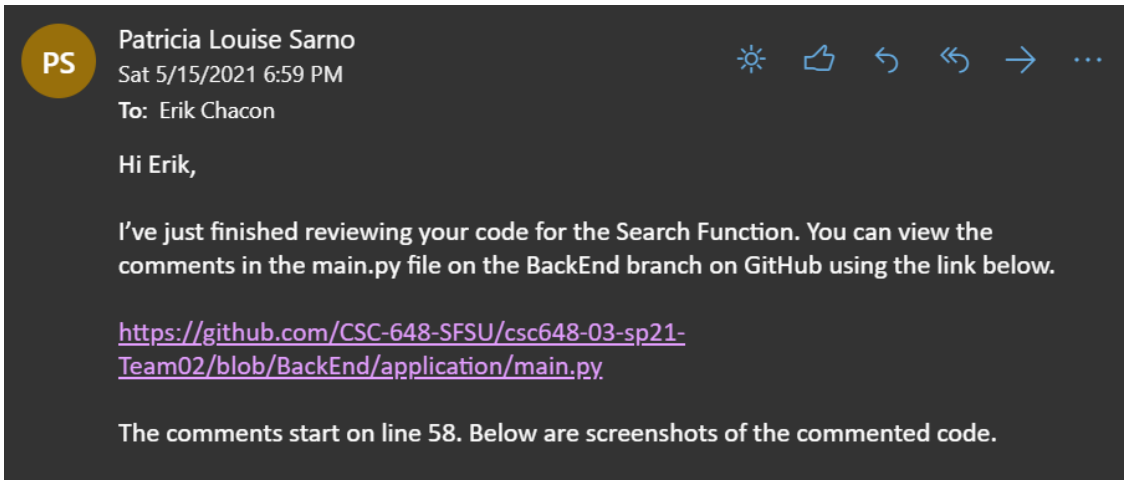
Test#	Test Title	Test Description	Test Input	Expected Correct Output	Test Results (Brower #1) Safari	Test Results (Brower #2) Google Chrome
1	Searching for a specific restaurant using both the drop down and the search bar.	Select a Mexican as a food type and then search for Tina's Tamales.	Drop Down: Mexican Search Bar: Tina's Tamales	All the Mexican food types because the drop down has highest priority.	PASS	PASS
2	Searching for a specific restaurant using only the search bar.	Search for McDee's restaurant.	Search Bar: McDee's	McDee's Restaurant to show up as a result.	FAIL	FAIL
3	Testing for insensitive cases.	Search for Woke Yolk in the search bar using different cases.	Search Bar: wOkE yOlK	Woke Yolk should pop up as a result.	PASS	PASS
4	Searching for part of a restaurants name.	Search for restaurants with 'la' in it in the search bar.	Search Bar: la	All the restaurants containing 'la' in them should pop up as a result.	PASS	PASS
5	Searching for a food type in the search bar.	Search for pizza in the search bar.	Search Bar: pizza	All the restaurants with category or 'pizza' in their title should pop up.	PASS	PASS

4. **Code Review**

Our GitHub Lead, Erik, sent his code to the Team Lead, Patricia, to review his code on the search function.



Patricia replied back with the updated commented/reviewed code linked on GitHub on the BackEnd branch as well as screenshots of the commented code.



The comments start on line 58. Below are screenshots of the commented code.

```
58  # Reviewed by Patricia Sarno
59  # on 05/15/2021
60
61  #COMMENT #1
62  #SUGGESTION TO PUT A MORE DETAILED DESCRIPTION OF THE SEARCH QUERY RIGHT HERE.
63
64  #COMMENT #2
65  #EXPLAIN WHAT THE DROP DOWN AND SEARCH BAR IS SUPPOSED TO DO LIKE WHAT PARAMETERS IT NEEDS, ETC.
66
67  #COMMENT #3
68  #CAN EXPLAIN WHAT DATA WE ARE GETTING/WHAT IS NEEDED AND WHAT IS BEING RETURNED WHEN
69  #CURSOR.EXECUTE IS EXECUTED.
70
71  def searchQuery():
72
73      # drop down has highest priority in filter
74      input = request.form.get('dropdown')
75      print(input)
76
77      # if using drop down to filter
78      if input != "Choose Cuisine" and input is not None:
79          cursor.execute(inputQuery(input))
80          data = cursor.fetchall()
81          outer = list(data)
82          convertPhoto(outer)
83          data = tuple(outer)
84          return data
85
86
```

```
92          if len(input) > 40:
93              return "error"
94
95          # get the data
96          cursor.execute(inputQuery(input))
97          data = cursor.fetchall()
98
99          # if search was empty or user wants everything
100         if len(data) == 0 or input == 'all':
101
102             # error in input
103             if input != 'all':
104                 return "error"
105             # return everything
106             return allData
107
108         # convert photos to base64
109         else:
110             # convert BLOB (image) to base64
111             outer = list(data)
112             convertPhoto(outer)
113             data = tuple(outer)
114         # return query results
115         return data
```

Let me know if you have any questions. Thank you.

Kind Regards,

Patricia

5. **Self-check: Best Practices For Security**

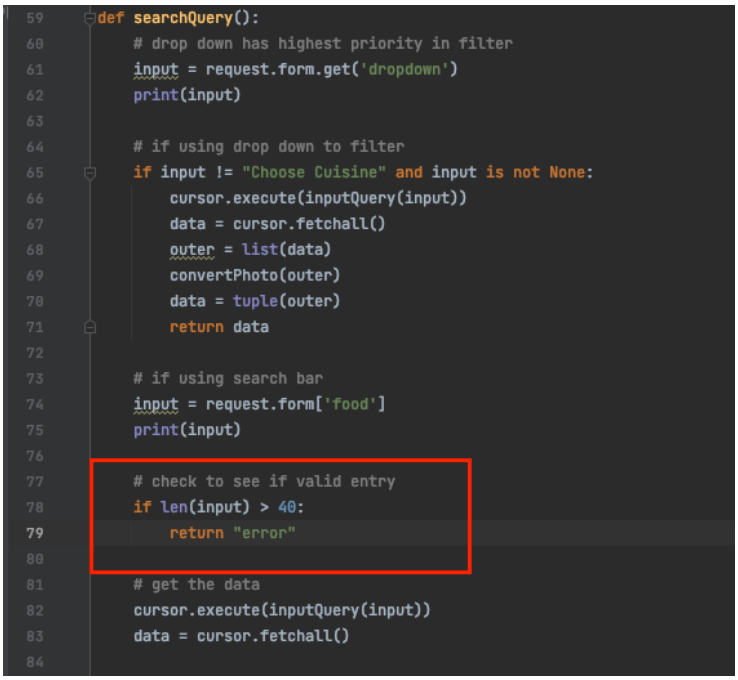
Asset to be protected	**Types of possible/expected attacks**	**Your strategy to mitigate/protect the asset**
Passwords	Has access to entire account with username	Encrypting passwords even from admins
User Addresses	Safety of user, access to important mails	Only users have access to their own address through account
Email Address	Prone to phishing emails, malicious attachments	Required username separate from email
Application Database	Has access to all users information	Database password stored in a secure file and only known by team lead

In the figure shown below, when users register their passwords are immediately encrypted and stored in the database using the sha1 function provided by MySQL. No one has any information on the password and no way of retrieving the password without brute force. The second figure shown below is what the password of each users looks like in the database, the admins do not have any information about their passwords for security purposes.

```
else:
    # If account doesn't exist, insert new account to RegisteredCustomer table
    cursor.execute('INSERT INTO RegisteredCustomer VALUES ("", %s, %s, "", "", "", "", %s , sha1(%s), "", "")',
                   (full_name, sfsu_email, sfsu_email, password))
    conn.commit()
    print("Successfully registered")
    # After registration, redirect to login page
    # CONTAINS BUG that after register, it won't let user log in because of search
    return render_template("customerlogin.html", food=foodTypes)
```

Passwords
▶ 66cfe7cbbc57bfc9fd3e1927ec5ba36aceecdc59
148af7c6ec6abaf35ead6b8ad51647e41ff2ff50
ba8830164aff0f6fb8cf811e31b41c87c5987e26
69991dd8cc40f9e3ac71bc8d1b4c4d4560406ac1
fd99bc932e13185505fe53802e17d19b1017b621
5d29c58124e53db567dafc817939f8452b5cfb85
9fc7292b15bb4291032e217fbce6dadb2ca0a5be
1079bb25718a12940a3c99b377320647a3b8eda0
506fc9e4da8534b89824dbfe18fc2011d6455a05
b342bd45ff3cf17d305c96b0b914bf4ee7dbf222
267a9c9ea00216746544a742140ee230cb172499
eb0123caef7a3a8c330bdd0aaad5612295863cef
d0b480a774a44088ed4f1f34427a92b016e5482d
b3b3c32f1edef1127ac6dc764e37abf9a78bcbf2
b1a8c5da5f053369ccbe420306b54049eb65afb5
2b987e8c876a3a02b3c48c36501446ab3fbeb7d2
8dfc8cb04d40228b0dbca316951d31d8ceed3b22
933bdef97364894e6827a232eaca2b0c0a3bed7c
dcb19d49727852fdaa0bca7f30a7aa53a22f50ec
8af040feef92ad1eca214101281fa3a3cc0a785f
044d304491be1ee58ca19243e7ba501332861c96

When searching through the search bar, we are validating the search through our back end. We send an error if the user searches with more than 40 characters.



```
59  def searchQuery():
60      # drop down has highest priority in filter
61      input = request.form.get('dropdown')
62      print(input)
63
64      # if using drop down to filter
65      if input != "Choose Cuisine" and input is not None:
66          cursor.execute(inputQuery(input))
67          data = cursor.fetchall()
68          outer = list(data)
69          convertPhoto(outer)
70          data = tuple(outer)
71          return data
72
73      # if using search bar
74      input = request.form['food']
75      print(input)
76
77      # check to see if valid entry
78      if len(input) > 40:
79          return "error"
80
81      # get the data
82      cursor.execute(inputQuery(input))
83      data = cursor.fetchall()
84
```

We are validating users' emails by using the **re (regular expression) Library** in Python. We are checking the string that is being sent through the registration text field if it contains a valid sfsu.email (ie. @mail.sfsu.edu, @faculty.sfsu.edu or @sfsu.edu).

```
elif not re.match(r'[^@]+@([a-z]{3,15}\.)?sfsu\.edu$', sfsu_email):
    # Checking for valid sfsu.edu email
    message = 'Invalid email address'
    print("invalid email")
```

6. Self-check: 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. Application delivery shall be from chosen cloud server.
   Progress: ***DONE***
2. Application shall be optimized for standard desktop/laptop browsers e.g must render correctly on the two latest versions of two major browsers.
   Progress: ***DONE***
3. All or selected application functions must render well on mobile devices (specifics to be developed in consultation with users e.g. Petkovic)
   Progress: ***ON TRACK***
4. Ordering and delivery of food shall be allowed only for SFSU students, staff and faculty.
   Progress: ***DONE***
5. Data shall be stored in the database on the team's deployment cloud server.
   Progress: ***DONE***
6. No more than 50 concurrent users shall be accessing the application at any time.
   Progress: ***ON TRACK***
7. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
   Progress: ***DONE***
8. The language used shall be English (no localization needed).
   Progress: ***DONE***
9. Application shall be very easy to use and intuitive.
   Progress: ***DONE***
10. Application should follow established architecture patterns.
    Progress: ***DONE***
11. Application code and its repository shall be easy to inspect and maintain.
    Progress: ***DONE***
12. Google analytics shall be used.
    Progress: ***ON TRACK***
13. No email clients shall be used.
    Progress: ***DONE***
14. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in the UI.

Progress: **DONE**

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

    Progress: **DONE**

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

    Progress: **DONE**

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

    Progress: **DONE**

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).

    Progress: **ON DONE**