

SW Engineering CSC648/848 Spring 2023

Milestone 4: Product description, Beta Launch, QA, and Usability

Testing

Food Feast

Team 01

Elahe Bashiri (Ebashiri@sfsu.edu): Team lead

Abbas Mahdavi: Back end lead

Megan Lew: Front end lead

Nathan Rennacker: Github master

Jed Graves: Developer

Alexander Diaz: Developer

Revision History

Date Submitted	Date Revised
5/19/2023	-

1) Product Summary:

Product Name: Food Feast

Food Feast is a dynamic web application designed to make finding meals and food delivery more efficient for the San Francisco State University (SFSU) community, comprising students, staff, and faculty. It targets the challenges of busy schedules and limited food options within the campus, by providing an accessible and user-friendly platform to search for nearby restaurants, order meals, and have them delivered right on campus or to the dorms. Food Feast stands as a strong supporter of local businesses, providing a unique channel to reach a specific customer base.

P1 Functions:

- Registration: Unregistered users shall be able to create an account using SFSU information.
- Browse Food Items: Unregistered users shall have the ability to view available food items.
- Map Accessibility: Unregistered users shall be able to access a visual map.
- Filter Application: Unregistered users shall be able to apply filters to restaurants.
- Restaurant Form Access: Potential restaurant owners shall have access to a registration form.
- Search Bar Usage: Users shall be able to use the search bar to find a specific restaurant.
- Login: Registered SFSU users shall have the capability to log in to their accounts.
- Order Placement: Registered SFSU users shall have the ability to select and order items.
- Restaurant Registration: Restaurant owners shall be able to register their establishments.
- Menu Upload: Restaurant owners shall have the ability to upload their menus.
- Delivery Interface Utilization: Delivery personnel shall have an interface to manage orders.
- Approval or Rejection of Restaurant Requests: The admin shall authority over approvals.
- Managing Approved Restaurants: The admin shall be able to revoke privileges of restaurant.
- User Management: The admin shall have the authority to manage the list of users.

Unique Proposition:

Food Feast's uniqueness lies in its tailored solution for the SFSU community, merging technology, convenience, and local business support. Unlike general food delivery services, Food Feast offers a campus-oriented service, providing fast delivery within the campus and the dorms. Furthermore, our student verification system offers exclusive discounts to verified students, making nutritious meals more affordable. We also accept payments through Gator Passes, providing a seamless and speedy payment process for the SFSU community.

URL: <http://35.160.127.228/>

2) Usability test plan for a selected function: (Search)

Test objectives: The search function is being tested for our application. It is being tested because it is a priority one function. Unregistered users shall be able to type in the name of a restaurant or cuisine.

Test background and setup:

System setup: Type the URL of the system to be tested.

Starting Point: Start on the homepage of the website.

Who are the intended users: The intended users of the search function would be anyone who is searching for food. This would be the faculty, staff, and students of SFSU.

URL of the system to be tested: Server URL: <http://35.160.127.228/>

What is to be measured: (user satisfaction evaluation Likert scale)

The efficiency of the search is being measured using a user satisfaction evaluation Likert scale. Users will use the same scale (strongly agree, agree, neutral, disagree, strongly disagree). We can measure the average and standard deviation of the responses and the percentage overall of the responses.

Usability Task Description:

search for restaurants with American cuisine.

Using the cuisine filter, search for American cuisine.

Task	Description
Search	Search for American Restaurants
Search using cuisine filter	Using filter search for American
Successful Completion Criteria	Search results show American Restaurant

Plan for evaluation of effectiveness: First, test the search function using the search bar and using the cuisine filter. The task completion is measured by the percentage of people who were able to complete the defined task in the defined time. Count the number of errors by the number of restaurants showing up in the search result that were not defined by the search term or cuisine filter.

Can implement using table:

Test / Use Case	% completed	errors	comments
search	X%	X	X
search using filter	X%	X	X

Plan for evaluation of efficiency: For search, efficiency can be measured with efficiency in time and efficiency in content. Efficiency in time is the average time it takes the user to complete the search. Efficiency in content is the number of results from the search.

Plan for evaluation of user satisfaction: (provide 3 Likert scale questions)

- The search was easy to use.
- The search results showed restaurants that matched the cuisine.

- The cuisine filter on the search showed search results matching the cuisine selected.

Scale is 1-5 with Strongly Disagree(1), Disagree(2), Neutral(3), Agree(4), Strongly Agree(5).

3) QA test plan and QA testing:

Unregistered Search Test

Setup:

Go to the home page of the application at <http://35.160.127.228/>, click on the yellow browse all restaurants button, no further setup is needed

Testing search functionality for unregistered users (in Chrome browser)

#	Title	Description	Input	Expected Result	Result
1	Empty Search	Testing empty search in search field	Press space and enter or simply press the search button on the right of the search bar	Eight restaurants should appear	PASS
2	% Like	Testing % like in search field for name	Type <i>indian</i> into the search bar and press enter	Two restaurants should show up in the results	PASS
3	Exact match	Testing exact match in the search field	Type <i>Roti Bistro</i> into the search bar and press enter	One restaurant called Roti Bistro should show up in the results	PASS
4	No matches	Testing no matches using the search field	Type <i>Ethiopian</i> into the search bar and press enter	No restaurants should show up in the results of the search	PASS
5	Long string	Testing excessively long search value	Type a string of 100+ characters or more into the search bar (use lorem ipsum for ease)	No restaurants should show up in the results of the search	PASS

Registered Test Search

Setup:

Go to the home page of the application at <http://35.160.127.228/>, and press the login button at the top right.

In the username field, enter **Test10**

In the password field, enter **Test10**

Press the Sign in button, and you will be redirected to the home page as a signed-in user

Press the yellow browse all restaurants button

No further setup is needed

Testing search functionality for registered users (in Chrome browser)

#	Title	Description	Input	Expected Result	Result
1	Empty Search	Testing empty search in the search field	Press space and enter or simply press the search button on the right of the search bar	Eight restaurants should appear	PASS
2	% Like	Testing % like in search field for name	Type <i>indian</i> into the search bar and press enter	Two restaurants should show up in the results	PASS
3	Exact match	Testing exact match in the search field	Type <i>Roti Bistro</i> into the search bar and press enter	One restaurant called Roti Bistro should show up in the results	PASS
4	No matches	Testing no matches using the search field	Type <i>Ethiopian</i> into the search bar and press enter	No restaurants should show up in the results of the search	PASS
5	Long string	Testing excessively long search value	Type a string of 100+ characters or more into the search bar (use lorem ipsum for ease)	No restaurants should show up in the results of the search	PASS

Testing in Microsoft Edge Browser

#	Title	Description	Input	Expected Result	Result
1	Empty Search	Testing empty search in the search field	Press space and enter or simply press the search button on the right of the search bar	Eight restaurants should appear	PASS
2	% Like	Testing % like in search field for name	Type <i>indian</i> into the search bar and press enter	Two restaurants should show up in the results	PASS
3	Exact match	Testing exact match in the search field	Type <i>Roti Bistro</i> into the search bar and press enter	One restaurant called Roti Bistro should show up in the results	PASS
4	No matches	Testing no matches using the search field	Type <i>Ethiopian</i> into the search bar and press enter	No restaurants should show up in the results of the search	PASS
5	Long string	Testing excessively long search value	Type a string of 100+ characters or more into the search bar (use lorem ipsum for ease)	No restaurants should show up in the results of the search	PASS


4) Peer Code Review:



AM

Abbas Mahdavi

To: Jed Graves

Sat 5/20/2023 6:58 PM



2 attachments (112 KB)  Save all to OneDrive - San Francisco State University  Download all

Hello Jed,

Hope you are doing well.

I am reaching out to you to request your assistance in reviewing the search functionality for our app.

Kindly review the attached code snippets attached, from the Browse.jsx file, and provide your valuable feedback.

Should you have any questions or require additional clarification, please don't hesitate to reach out.


Your contribution is greatly appreciated. Thank you!


Best regards,
Abbas M.
Backend Lead

Completed.

Ok, I will take a look.

Please try it now.

 Reply

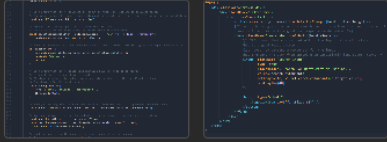
 Forward



JG

Jed Graves

To: Abbas Mahdavi

Sat 5/20/2023 7:55 PM



2 attachments (140 KB)  Save all to OneDrive - San Francisco State University  Download all

Abbas,

I have reviewed the code for the search function and made some comments. Overall, it looks great! Good choices for useability and very readable code.


I did make a couple minor suggestions regarding naming and clarity, but aside from that, it appears to be a well put together function.


I have provided screenshots here. You can also view my comments in Browse.jsx file the development branch of the repository:
<https://github.com/CSC-648-SFSU/csc648-03-sp23-team01/blob/development/application/client/src/components/Browse.jsx>

Your header comment looks good and contains the necessary proprietary information and explanation of functionality, as do your GitHub commit messages.

Please let me know if you have any further questions.

Best,
Jed G,
Backend Engineer

 Reply

 Forward


```

120     return (
121       <div className='browse-div'>
122         <div className='filter-div'>
123           <div className='filter'>
124             <Filter cuisines={cuisines} handleFilterChange={handleFilterChange} />
125             {/* Good job using a form for the search bar. This allows the user to submit the form by
126              pressing Enter, which is a good user experience decision. */}
127             <form className='search-bar' onSubmit={handleSearch}>
128               {/* It's great that you're controlling the input value with state.
129                This is a good React pattern.
130                Also, good job setting a maxLength for the input.
131                This prevents the user from entering an excessively long search term. */}
132               <input className='search-input'
133                 type="text"
134                 placeholder="Search for Restaurant or Cuisine..."
135                 value={searchRestaurants}
136                 onChange={(e) => setSearchRestaurants(e.target.value)}
137                 maxLength={40}
138               />
139             <button type="submit">
140               <PageviewIcon sx={{fontSize: 60}} />
141             </button>
142           </form>
143         </div>
144       </div>
145     </div>
146   </div>
147 );
148 };

```

```

76 const handleSearch = (event) => {
77   // Good job preventing the default form submission behavior.
78   event.preventDefault();
79
80
81   // It's great that you're trimming the input to remove any leading or trailing spaces.
82   // However, the variable name searchRestaurants is a bit confusing, since it's actually the search term.
83   const searchTerm = searchRestaurants.trim();
84
85   // Nice use of ternary operator to handle the 'All' case.
86   // This is a clean and concise way to create the filter function.
87   const selectedCuisineFilter = selectedCuisine === 'All' ? () => true : (restaurant) =>
88     restaurant.cuisine_name === selectedCuisine;
89
90   // Good job handling the case where the search term is empty. This is a good user experience decision.
91   if (!searchTerm) {
92     setRestaurants(allRestaurants.filter(selectedCuisineFilter));
93     navigate('/browse');
94     return;
95   }
96
97   // It's great that you're specifying multiple keys for Fuse.js to search.
98   // This will make the search more robust.
99   // It might be helpful to add a comment explaining what the 'threshold' option does
100   // for those unfamiliar with Fuse.js.
101   const fuseOptions = {
102     keys: ['name', 'cuisine', 'description'],
103     threshold: 0.25,
104   };
105
106   // Good job reusing the 'selectedCuisineFilter' from earlier, as it prevents redundant code.
107   const fuse = new Fuse(allRestaurants.filter(selectedCuisineFilter), fuseOptions);
108
109   // Nice use of map to extract the actual restaurant objects from the search results.
110   const searchResults = fuse.search(searchTerm);
111   const searchedRestaurants = searchResults.map((result) => result.item);
112   setRestaurants(searchedRestaurants);
113
114   // Good job navigating the user to the browse page after the search.
115   // This is a good user experience decision.
116   navigate('/browse');
117 };

```



Abbas Mahdavi

To: Jed Graves



Sat 5/20/2023 7:58 PM

Hello again Jed,

Thanks for the feedback and getting back to me in a timely manner.

I have read your comments, they are very concise and practical. I will be sure to make the needed adjustments, based on your suggestions.

Thank you again!

Best,
Abbas M.
Backend Lead

Get [Outlook for iOS](#)



You are very welcome!

Sounds great, thank you!

Happy to help!

↩ Reply

➦ Forward

5) Self-check on best practices for security:

Assets you are protecting:

- **User Information:**

- Major Threats: Leaked user email, and password, Users can modify other user's accounts, such as deletion
 - Regular users: Last order location, current order's delivery location
 - For restaurant Owners: Leaked Restaurant details, List of menu Items
 - For Drivers: Accessible known location, last/current accepted order
- Process of protection: Keep the info in a private and secure database, encrypt the password, pass API requests only through protected routes, make sure important data is only available to the currently logged-in user, using sessions to keep track of the user's activity. Killing their session after a certain time has passed. Making sure the data stored in local storage is cleared when the user is logged out. Overall, keep every valuable information, such as location, email, and other info, in a database that is only accessible to the site admin, and only to the user.

- **System Infrastructure**

- Major Threats: Accessible database information, not protected routes via middleware
- Process of protecting: Keeping critical information/values in a .env file, that is kept only in Admin's local device. Ensuring the data is private, protected, and not posted anywhere, such as a GitHub repository. Additionally, making sure important routes, such as all admin routes, which have the ability to make major changes to the database, such as deleting users, and restaurants, and viewing the database tables and content. By having multiple layers of security, to routes, APIs, and call requests, both in the front end that denies access directly by browsing to users, but also backend modern and industry approved middleware security, for advanced hackers.

- **Ensuring Passwords are protected/encrypted:**
 - Passwords for ALL users are encrypted and are kept in the database
 - The encryption happens via crypt, which is a one-way encryption system, meaning it cannot be reversed.

id	username	email	password
43	Alex1	Alex1@gmail.com	\$2a\$10\$6oO.cPfDvzuUd6OQBj...

- **Confirming input Data Validation:**
 - Search input has a maxLenght of 40, allowing it a max and up to 40 characters in the search input:

Code: `<input className='search-input'`

`type="text"`

`placeholder="Search for Restaurant or Cuisine..."`

`value={searchRestaurants}`

`onChange={(e) =>`

`setSearchRestaurants(e.target.value)}`

`maxLength={40}`

`/>`

- Ensuring only students with a valid sfsu email can signup as a regular user (This does not apply to drivers or restaurant owners)

Code:

`// check if email ends with @sfsu.edu`

`const emailRegex = /^[w-.]++@sfsu.edu$/i;`

`if (!emailRegex.test(formData.email)) {`

`setErr("Please enter a valid SFSU email address");`

`return;`

`}`

Other validations:

Validating password is min 5 char, lower and upper case, with a number

```
const passwordRegex = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{5,}$/;
```

The rest are done in a similar fashion:

Confirming passwords match, all users agree to the terms of condition,

Restaurant enrollment validations, such as only \$ chars for price range,
the number for a phone number, valid URL for pictures, etc.

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

- Refer to system properties and constraints such as:
 - Reliability: **Done**
 - Response time: **Done**
 - HW and networking requirements: **Done**
 - Usability requirements: **Done**
 - Marketing, legal, licensing: **Done**
 - Media content (formats, size...): **Done**
 - Privacy: what is the data collected, how is the data used: **Done**
 - Compatibility (e.g. which browsers...): **Done**
 - Can refer to:
 - Product (product behavior like speed, reliability): **Done**
 - Organization (e.g. process, standards used): **Done**
 - External factors (e.g. branding, legal disclaimers displayed): **Done**