SW Engineering CSC648/848 Spring 2022

Team 01

Gator Xchange

Team Lead
Drashti Pareshkumar Shah
dshah5@mail.sfsu.edu

Back End Lead
Thomas Duc Nguyen
tnguyen135@mail.sfsu.edu

Front End Lead
Mary Nicole Tangog
mtangog@mail.sfsu.edu

Github Master
Wilfredo Joel Aceytuno Jolon
waceytun@mail.sfsu.edu

Team Members
Micheas G Gebere
mgebere@mail.sfsu.edu

Javier Guintu Marquez
jmarquez21@mail.sfsu.edu

Milestone 4

Date Submitted	Date Revised
05/16/22	

Table of Contents

Product Summary

Product Name: Gator XChange

Product Description:

Our application's main purpose is to serve as a virtual marketplace solely for San Francisco State students, faculty, and staff to facilitate the buying and selling of products. We believe our application will be able to fill a void in the market that the gator community has been looking for. We offer a consistent user experience through a blend of sleek minimalistic design combined with fast and responsive performance. Our advantages over our competitors are our focus on affordable prices, community development, and personal safety. We seek to unite the growing gator community by establishing our application on a foundation of trust and camaraderie that will grow in tandem with current and future generations of gators.

List of All Major Committed Functions:

Our product allows users to:

- Register an account and login/logout (Only SFSU students, faculty, and staff)
- Search and browse products

Registered users can:

- Create a post to sell a product
- Send a message to product seller
- View a list of their uploaded posts
- View a list of messages received for their post

Admin:

- Admin approves posts before they're shown to users

Product Uniqueness:

Our registered users can set designated pickup locations on the SF State campus when creating a post to meet an interested buyer. This feature allows transactions to be safely conducted in well-lit, populated areas. Both buyers and sellers can view an image of the campus map highlighting the pick up location for reference in case they get lost.

Product URL:

http://54.67.78.9

Usability Test Plan

Test Objectives:

We will test our product search function for usability to gauge areas that need improvement in the user experience department. Product search is one of our core features that users will interact with so it is imperative that we test it extensively for usability.

Test Background and Setup:

We will set up our testing system to use a website URL that usability testers can access remotely over the internet in an environment of their choosing. This system will simulate real world conditions that our intended users will access our application. By simulating real world conditions as closely as possible, the quality of our usability metric data will be higher and give us a better idea of usability flaws.

Our starting point for our usability test will be the moment our website loads and the testers sees the user interface. It will be up to them to figure out how to conduct all of the usability tasks we describe below.

Our intended users will be San Francisco State students, faculty, and staff members. Their age, demographics, devices, and technical skill levels will be diverse, so we need to consider these factors in order to make the user experience seamless, inclusive, and accessible to all of them.

Our system URL used to test usability will be http://54.67.78.9. This URL will lead to a website that loads our application's home page user interface.

We will measure user satisfaction using a Likert scale with a range from highly disagree, disagree, neutral, agree, and strongly agree.

Usability Task Description:

Tasks to be tested will be:

- Search with only a category filter
- Search with only text input
- Search with a category filter and text input
- Search without a category filter and text input

Evaluation of Effectiveness:

If we were to evaluate effectiveness, we would test out our search function to evaluate whether it allows search's use cases to be completed such as searching products using only category filters, searching products using just text input, searching products with both a category filter and text input, and searching with no search parameters. We will also record the user task completion percentage and the number of errors encountered by users during task attempts. An optional comment section will be provided to users if they would like to express any specific feedback which serves as another data point during the effectiveness evaluation.

Evaluation of Efficiency:

To evaluate the efficiency of the search function, we shall consider efficiencies in time, effort, and design. One of the factors for time efficiencies we will record is the average completion time to search using only a category filter, searching with only text input, or a combination of both a category filter and text input. Another time efficiency factor that we'll record is the average time it takes for all users to search successfully. We will measure design efficiency by noting the number of clicks required to complete a search using our search's use cases and number of clicks it takes to correct a subsequent search after initially searching with incorrect parameters like wrong category or a typo in the input field.

Evaluation of User Satisfaction:

Statement	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Optional Comments
Submitting a search was intuitive.				X		The search button text describes what clicking on it does. The button's color makes it distinct
Changing category filters was easy.					X	Changing the category filters felt smooth
Search results were consistent with my search parameters.				X		If there are 0 search results, suggest the user to try a different search or show a few random results

QA Test Plan

Test Objectives: We will test the drop down categories and the search functionality.

HW and SW setup (including URL):

Hardware Setup:

- Server Host: AWS EC2 t2.micro, 1vCPU, 1 GiB RAM, 8 GiB storage
- Operating System: Ubuntu Server 20.04 LTS
- Database: MySQL v8.0.28
- Web Server: NGINX 1.20.1 (Stable)

Software Setup:

- Server Side Language: Javascript (Node.js v16.14.0 LTS)
- Web Frameworks: React, Express.js, Bootstrap, React Bootstrap,
- Node.js Packages: Nodemon, multer, sharp, pm2, bcrypt, crypto, morgan, helmet, socket.io, redis
- Tools: Trello
- IDE: Visual Studio Code

URL: http://54.67.78.9

Feature to be tested: Search Function

Chrome

Test #	Description	Input	Expected Correct Output	Results (Pass or Fail)
1	Test “electronics” category and input “charger” on the search bar.	Type “charger” in search field	Check that it shows the number of results and the chargers listing.	PASS
2	Test “All” category and input random values on the search bar	Type random values such as “ahdahsd” on the search field	Check that it shows the number of results. If it does not match the string, it should let the user know that there is no result found.	PASS

3	Test when a user inputs “pencil” on the search bar.	Type “pencil” on the search field	Check that if there is no match, it should show other listings instead of an empty result.	FAIL
4	Test “clothes” category and input “book” on the search bar	Change category to “Clothes” and input “book” on the search field.	Check that no results returned as the input “book” is not in the correct category.	PASS

FireFox

Test #	Description	Input	Expected Correct Output	Results (Pass or Fail)
1	Test “electronics” category and input “charger” on the search bar.	Type “charger” in search field	Check that it shows the number of results and the chargers listing.	PASS
2	Test “All” category and input random values on the search bar	Type random values such as “ahdahsd” on the search field	Check that it shows the number of results. If it does not match the string, it should let user know that there is no result found.	PASS
3	Test when a user inputs “pencil” on the search bar.	Type “pencil” on the search field	Check that if there is no match, it should show other listings instead of an empty result.	FAIL
4	Test “clothes” category and input “book” on the search bar	Change category to “Clothes” and input “book” on the search field.	Check that no results returned as the input “book” is not in the correct category.	PASS

Code Review

Emails asking for code review:

CSC 648-848 Spring 2022 Team 1 - M4 Code Review

From: Mary Nicole Tangog
Sent: Friday, May 13, 2022 10:16 PM
To: Drashti Pareshkumar Shah <dshah5@mail.sfsu.edu>
Subject: CSC 648-848 Spring 2022 Team 1 - M4 Code Review

Hello Drashti,

We are testing our search functionality.
Can you please review the search result source code?

Thank you,
Mary Nicole Tangog

From: Mary Nicole Tangog <mtangog@mail.sfsu.edu>
Sent: Friday, May 13, 2022 10:34:18 PM
To: Drashti Pareshkumar Shah <dshah5@mail.sfsu.edu>
Subject: Re: CSC 648-848 Spring 2022 Team 1 - M4 Code Review

Hello Drashti,

I attached more screenshots of codes that needs to be reviewed.
Can you also please review the Post, Login, and Sign-Up code?

Thank you,
Mary Nicole Tangog

Email of code review summary:

Login.css

From: Drashti Pareshkumar Shah <dshah5@mail.sfsu.edu>
Sent: Saturday, May 14, 2022, 3:24 PM
To: Mary Nicole Tangog <mtangog@mail.sfsu.edu>
Subject: RE: CSC 648-848 Spring 2022 Team 1 - M4 Code Review

Hello Mary,

I have reviewed and updated the header comments and inline comments in files Post.js, Login.js, SignUp.js. Here is the reviewed code.
Also I noticed that the application had common names for some classNames and so I changed them to specific file names and even added css file for login.

Login.css →

```
/*****************************************************
 * Purpose: To use customised css for login form
 * Output: A specific color, size background etc all
 *         used in registration form and login form
 *         the user's search parameters
 * Error Messages: None
 * Author:Drashti Shah
 *****************************************************/

.App {
  text-align: center;
}

/* Three columns side by side */

.container-login {
  width: 100%;
  display: flex;
  justify-content: center;
  flex-direction: column;
  align-items: center;
}

/* Display the columns below each other instead of side by side on small screens
@media screen and (max-width: 780px) {
  .column {
    width: 80%;
```

Post.js

Post.js

174 lines (164 sloc) | 5.97 KB

```
/*****************************************************
 * Purpose: Allows the user to create a post from their
 * account to buy or sell. Ensures if the user is logged in
 * if not the users are given the link to register for
 * an account at the bottom.
 * Error Messages: None
 * Author: Mary Tangog, Drashti Shah
 *****************************************************/

import React, { useState } from "react";
import Form from "react-bootstrap/Form";
import { Link } from "react-router-dom";
import "../css/post.css";
import axios from "axios";

const Post = () => {
  const [title, setTitle] = useState("");
  const [category, setCategory] = useState("");
  const [price, setPrice] = useState("");
  const [description, setDescription] = useState("");
  const [val, setVal] = useState();
  const [check, setcheckboxvalue] = useState(false);
  const uploadFileEle = document.getElementById("fileInput");

  // Event handler for submitting the post information to send to backend
  const handleSubmit = (e) => {
    // Don't refresh the page upon submitting post queries
    e.preventDefault();
    // Prevents form from being submitted if form is not valid

    // Create post parameters that will be used for SQL queries into the database
    const postData = {
      title: title,
```

Signup.js

Signup.js

227 lines (217 sloc) | 7.83 KB

```
/*****************************************************
 * Purpose: Allows the user to create an account on the
 * GatorXchange website by filling out the registration
 * form. The users enter sfsu id, SFSU email along with
 * username and password and so they are verified to
 * create and use the website, hence ensuring safety for SFSU
 * staff, students and faculty.
 * Error Messages: None
 * Author: Drashti Shah, Wilfredo Aceytuno
 *****************************************************/
import React from "react";
import { useState } from "react";
import "../css/registration.css";
import { Link } from "react-router-dom";
import { Form } from "react-bootstrap";
import axios from "axios";

const Signup = () => {
  const [username, setUsername] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfPassword] = useState("");
  const [sfsu_id, setID] = useState();
  const [setcheckboxvalue] = useState(false);

  // Prevents form from being submitted if form is not valid
  (function () {
    "use strict";
    var forms = document.querySelectorAll(".needs-validation");
```

SearchResults.js

```
25
26    // Prevents form from being submitted if form is not valid
27    (function () {
28      "use strict";
29      var forms = document.querySelectorAll(".needs-validation");
```

The search file had proper headers and inline-comments.

3 contributors

99 lines (89 sloc) | 2.94 KB

Raw | Blame

```
1  /*****************************************************
2   * Purpose: Displays a list of search results
3   * based on the user's search parameters that is
4   * returned from the database
5   * Input: User's search parameters in search bar
6   * Output: A list of search result cards relating to
7   * the user's search parameters
8   * Error Messages: None
9   * Author: Thomas Nguyen
10  *****************************************************/
11 import { useState, useEffect } from 'react';
12 import '../css/App.css';
13 import axios from 'axios';
14 import {Button} from 'react-bootstrap'
15
16 const SearchResults = () => {
17   let [results, updateResults] = useState([]);
18
19   // This function displays the list of search results received from database
20   const displayResults = results.map((result, i) => {
21     return (
22       <div className="card" key={i}>
23         <div className="card-content">
24           <p>Category: {result.category}</p>
25         </div>
26         {/* <div className="card-content">
```

SearchBar.js

```
24           <p>Category: {result.category}</p>
25         </div>
26         {/* <div class="card-content">
```

4 contributors

247 lines (228 sloc) | 8.16 KB

```
1  /*****************************************************
2   * Purpose: Allows the user to search for a post by
3   * selecting a category and entering a search term
4   * Input: Category, Search Term
5   * Output: Renders the search results page
6   * Error Messages: None
7   * Author: Thomas Nguyen
8   *****************************************************/
9  import React, { useState, useEffect } from "react";
10 // import { useNavigate, Routes, Route, Navigate, Link } from 'react-router-dom';
11 import {
12   Container,
13   Row,
14   InputGroup,
15   Dropdown,
16   DropdownButton,
17   Form,
18   FormControl,
19   Button,
20   Modal,
21 } from "react-bootstrap";
22 import axios from "axios";
23 import SearchResults from './SearchResults';
```

Now every important file for the M4 has comments, header and is properly formatted.

Thanks.
Drashti

Sent from Mail for Windows

Self-Check on Security Best Practices

We confirm that passwords in the database have been encrypted:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

user_id	sfsu_id	username	email	password	registered	is_admin
1	900000000	admin	admin@mail.sfsu.edu	$2b$10$CcLwXdARaOarpwiJbwsHVe68ZgptnD/5U.Z...	1	1
2	900000001	user1	user1@mail.sfsu.edu	$2b$10$smxeJGHmKkNkPjehqYX0weSUJprGV4/TXT...	1	0
3	900000002	user2	user2@mail.sfsu.edu	$2b$10$DCr7LIrX5TnqnCtxM5G6tOeQempi22L.uKq...	1	0
4	900000003	user3	user3@mail.sfsu.edu	$2b$10$czQF./36ll5mibrXbusb.uwHKcp4B/cjr2Xz7ch...	1	0
5	900000004	user4	user4@mail.sfsu.edu	$2b$10$uFwE3/f4VXGCsPDD3up5LeTJZ2VsnkY8ICfI...	1	0
6	900000005	user5	user5@mail.sfsu.edu	$2b$10$VR6krdTKzagXMK65ZCPJ2ONP2N4OH/rg/bV...	1	0
7	900000006	user6	user6@mail.sfsu.edu	$2b$10$5y41MpNZP0n0rQHE3.bp1OYAGO0xOAeK6...	1	0
8	900000007	user7	user7@mail.sfsu.edu	$2b$10$V26wg54i7bPK8gJ.GsihjurrGMsbxePTZwA9...	1	0
9	900000008	user8	user8@mail.sfsu.edu	$2b$10$AgUQNsE0bhLoX/wP1.DrPeGsGQj8N3JgpeP...	1	0
10	900000009	user9	user9@mail.sfsu.edu	$2b$10$Qxm.FYhWwon4EHMPZ1LqKeh3QB2.8jWt6...	1	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

users 1

Output

Action Output

#	Time	Action	Message
1	17:04:30	SELECT * FROM users LIMIT 0, 1000	10 row(s) returned

We confirm following input data validation:

Search bar input only takes up to 40 alphanumeric characters

```
const handleSearchTerm = (event) => {
  if (event.target.value.length >= 40) {
    window.alert("Search term shouldn't exceed 40 characters!");
  }
  console.log(event.target.value);
  setSearchTerm(event.target.value);
};
```

Email must include “sfsu.edu” at the ends

```
<input
  class="form-control"
  id="validEmail"
  type="email"
  placeholder="@sfsu.edu"
  required
  pattern=".+@sfsu\.edu"
  value={email}
  onChange={(e) => setEmail(e.target.value)}
/>
```

Passwords must contain at least one number, one uppercase, one lowercase , and at least 8 or more characters.

```
<input
  class="form-control"
  id= "validPassword"
  type="password"
  placeholder="Password"
  required
  pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"
  value={password}
  onChange={(e) => setPassword(e.target.value)}
/>
```

SFSU ID must be 9 digits.

```
<input
  class="form-control"
  id="validID"
  required
  pattern="\d{8}[0-9]"
  placeholder="*********"
  value={sfsu_id}
  onChange={(e) => setID(e.target.value)}
/>
```

Security Assets Summary Table:

Asset to be protected	Types of possible/expected attacks	Your strategy to mitigate/protect asset
Password	1. Someone who is not in our team gains access to our database. 2. Someone guesses a user's simple password.	1. Passwords in our database are encrypted. 2. Passwords have to be at least 8 characters, must contain one number, one uppercase, and one lowercase.
Registration	1. Someone that is not part of SFSU tries to register for an account.	1. A SFSU ID and email is required to register.
Database	1. SQL injection in search bar, registration, and login input fields.	1. We limit the number of characters allowed in the search bar. 2. We use prepared statements to create SQL queries instead of user input string concatenation.

Self-Check on Adherence to Original Non-Functional Specifications

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 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.
    - ON TRACK

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.
    - DONE

6. Privacy of users shall be protected.
    - DONE

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

8. Application shall be very easy to use and intuitive.
    - DONE

9. Application should follow established architecture patterns.
    - DONE

10. Application code and its repository shall be easy to inspect and maintain.
    - DONE

11. Google analytics shall be used.
    - ON TRACK

12. No e-mail 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.
    - ON TRACK

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

14. Site security: basic best practices shall be applied (as covered in the class) 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 (WWW and mobile) shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2022. For Demonstration Only" at the top of the WWW page nav bar. (Important so as to not confuse this with a real application).
    - DONE