SW Engineering CSC 648/848 Fall 2020
SFSU Market

Team # 03

Team Lead: Nicholas Green ngreen2@mail.sfsu.edu
Back end Lead: Steven McHenry
Front end Lead: Ahmad Afghan
Github Master: Lauren Luke
Waqas Hassan

"Milestone 4"

Due Date: December 8, 2020

Product Summary

Name: SFSU Market

List of P1 Functions:

Unregistered users:

- Unregistered users shall be able to create accounts on the application.
- All users shall be able to view listings.
- All users shall be able to search for listings by title and other criteria such as location and price.
- On load, the site shall display the most recent listings made on the application.

Registered users:

- Registered users shall be able to log in to their preexisting accounts.
- Registered users who have logged in shall be able to log out through the application.
- Registered, logged-in users shall be able to list an item or service for sale on this application and attach images to that listing.
- Registered, logged-in users shall be able to message other users privately.
- Registered users shall have a dashboard page to view their listings and messages.

ADMIN:

- Site admins shall be required to verify all listings before they are made public.
- Site admins shall be able to delete or lock listings made by registered users.
- Site admins shall be able to ban accounts from creating listings or messaging users.

Link: https://648-group3.xyz/

Usability Test Plan

Test Objective

The test objective is to determine the efficiency and ease-of-use of the search feature. Expose items with the search function by asking the tester to perform the search task with no prior instruction.

The search bar is for general use, not just for account users. Typical online browsers may not want to sign up for SFSU Market but may be interested to view its features and possible listings.

Test Plan

System Setup

- Testers will log into a computer with access to the internet and use the site URL address: https://648-group3.xyz/ with accepted internet browsers. Google chrome, Safari, Firefox.

Starting point

- Testers will begin at the site homepage wherein a search bar will be present.

Task to be accomplished

- Testers will attempt a search for listings and products of their choice.
- Testers will attempt to search for the item by category using the drop down menu.

Intended Users

- Students looking for items and services near San Francisco State University, who desire to buy them as soon as possible.
- Users will be assumed to have no technical expertise.

Completing Criteria

- Tester is satisfied that they have accomplished the search query and found items they were looking for

Feedback Process

- When the student is interested in their desired items that fit their personal tastes, the student may contact the seller via messaging option. The two may communicate with one another through message and may exchange phone numbers, information, etc.

Usability Questionnaire

1. I found the Search function easy to use (check one)

1	2	3	4	5
Totally Disagree	Disagree	Neutral	Agree	Totally Agree

2. I am satisfied with the search results.

1	2	3	4	5
Totally Disagree	Disagree	Neutral	Agree	Totally Agree

3. I understand the individual search parameters in relation to the search category.

1	2	3	4	5

Totally Disagree Disagree Neutral Agree Totally Agree

Comments:

<u>QA Test Plan</u>

- **Test Objective**
    - The test objective is to allow registered and unregistered users to easily find items on the website using the search function.

HW and SW setup

- **Hardware:**
    - Windows 10

- **System Setup:**
    - The system setup starts off with testers using a computer to run the application on Google Chrome, Safari, or Firefox from which they will be prompted to the homepage. From the homepage, they have access to the search functions which they will use to locate the demo items listed within the application. Testers will

continue to use the search function to locate items within the application using a different web browser each time.

- **Intended users:**
  - San Francisco State University's faculty and students.

- **URL:**
  - https://648-group3.xyz/

- **Feature to be tested:**
  - Search function.

- **QA Test Plan:**
  - The test plan is to test the functionality of the search functions within the application and by using this table, we can test if the application is capable of working through different web browsers such as Google Chrome, Safari, and Firefox.

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Google Chrome	Safari	Firefox
1	Search functions exist (search bar/button)	Search bar and button should be easily accessible from the homepage.	N/A	Visible search bar.	PASS	PASS	PASS
2	Search functions work.	Enter the search item and click on the search button, the homepage should turn into a results page.	Type an item in the search bar and click the search button.	Homepage changes to results page.	PASS	PASS	PASS
3	Results are relevant to the search	The results page should display demo	Type an item in the search bar	Should show books, devices, etc	PASS	PASS	PASS

	item.	items relevant to the search.	and click the search button.	relevant to biology.			

Code Review

SM

Steven Scott Mchenry
Tue 12/8/2020 5:09 PM
To: Lauren Jade Luke

Hi Lauren,

Can you please review this code for me? It is the code for searching through listings. Let me know if you have any questions!

Thanks

```
import { Listing } from '../database/entities/Listing'
import { Category } from '../database/entities/Category'
import { Request, Response } from 'express'
import { Like } from 'typeorm'
```

```
import { Listing } from '../database/entities/Listing'
import { Category } from '../database/entities/Category'
import { Request, Response } from 'express'
import { Like } from 'typeorm'
//Code Review By:Lauren Luke
//Make sure to include a code header

//Nice clarification with type of search being created
export const getListings = async (req: Request, res: Response) => {
  // search with a query
  if (req.query.searchQuery) {
    const search = req.query.searchQuery
    // search with JUST a query.
    if (req.query.category === 'all') {
      const listings = await Listing.find({ where: { title: Like(`%${search}%`) } })
      res.send({
        data: listings,
        errors: [],
      })
      return
    }
//Clear that we are searching with query and category
    // search with query and category selected
    const category = await Category.findOne({ where: { name: req.query.category } })
    if (category) {
      const listings = await Listing.find({
        where: {
          title: Like(`%${search}%`),
          category,
        },
      })
      res.send({
        data: listings,
        errors: [],
      })
      return
    }
  }
```

```
//Nice in-line comments
  // search with just a category
  if (req.query.category === 'all') {
    res.send({
      data: await Listing.find(),
      errors: [],
    })
  }

  // actually category selected
  const category = await Category.findOne({ where: { name: req.query.category } })
  if (category) {
    const listings = await Listing.find({
      where: {
        category,
      },
    })
    res.send({
      data: listings,
      errors: [],
    })
    return
  }
}
//Overall, code looks good and is easy to follow
```

Self-check on best practices for security

Major assets of protection:

1. User data
2. Api keys
3. Database access

Major threats:

1. User data
    a. Unauthorized user gains access to personal user information
    b. Hacker uses user data to gain access to other websites with financial information
    c. User data gathered and sold on the dark web for scamming purposes
2. Api keys
    a. Hacker gathers api keys to rack up usage costs for their own gain
    b. Hacker uses your api keys and gets admins blocked from platforms
    c. Api keys are stored on github and can be easily viewed
3. Database access
    a. All data can be accessed and analyzed for proprietary gain
    b. Malicious user can delete all of your stored data
    c. Malicious user can overload and shut down your system

How we are protecting:

1. User Data
    a. Gated api routes for authenticated users
    b. Multiple levels of authentication for each step of accessing user information
    c. Encrypted passwords stored on database
2. Api keys
    a. They will be stored as environment variables and those environment variables will be added to the .gitignore
3. Database access
    a. Shielded via ssh and admin password protection
    b. Few people will have direct access to the database to minimize potential security holes

Input Validation:

1. Search bar
    a. Maximum of 40 characters
2. Email
    a. Maximum of 40 characters
    b. Must have sfsu.edu
    c. Cannot be used already
3. Password

    a. Minimum of 6 characters
4. Listing title
    a. Maximum of 40 characters
5. Listing image
    a. Must be .png, .jpg, .jpeg, .heic, of .gif
    b. Maximum file size of 5 mb

Non functional spec check

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO). **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 **DONE**
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 **ON TRACK**
6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **ON TRACK**
7. The language used shall be English (no localization needed) **DONE**
8. Application shall be very easy to use and intuitive **ON TRACK**
9. Application should follow established architecture patterns **DONE**
10. Application code and its repository shall be easy to inspect and maintain **ISSUE**

The group as a whole needs to, and will, get better about code comments and labeling commits. A lack of communication in this way could make the code hard to maintain.

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 **DONE**
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 **ON TRACK**
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, Fall 2020. For Demonstration

Only" at the top of the WWW page. (Important so as to not confuse this with a real application).
ON TRACK