SW Engineering CSC648/848 Fall 2019

SFSU Buy and Sell

Team 11

Milestone 4

December 10th 2019

1. **Daniel Janes** - Team Lead djanes@mail.sfsu.edu
2. **Gem Lagman** - Github master / Back-End
3. **Monique Martinez** - Front-End Lead
4. **Raya Farshad** - Back-End Lead
5. **Pramish Dhakal** - Front-End

Revision	Date Submitted
Milestone 4 v.1	Dec. 12 2019
Milestone 4 v.2	Dec. 16 2019

1) Product Summary

The name of our product is SFSU Buy and Sell. SFSU Buy and Sell will provide the following functions to the user:

1. Anyone can browse and search for items
2. Anyone can browse textbook listings by SFSU classes
3. Anyone can filter listings by price or most recent
4. Anyone can register with an SFSU email
5. Registered users can message sellers
6. Registered users can list items to sell
7. Registered users can see their listings
8. Registered users can mark their listings as sold or delete them
9. Registered users can see their messages and reply to them

SFSU Buy and Sell will provide the user to search by specific SFSU class in order to find the right textbook they need. The user will also be able to specify a specific SFSU class when listing a textbook for sale. Our welcome can be found at: http://sfsubuyandsell.xyz/

2) Usability Test Plan

The major feature that we will focus on for our usability testing is our search and filter features. We would like to test search and filter because they are essential for our users to find what they are looking for inorder to buy products. This feature goes hand in hand and allows the user to find tune exactly the kind of item they are interested in. We also want to know our user's thoughts on ease of use of our process. Making sure that it is easy and intuitive is key to having a

good user experience. Once the user has searched for the item they are interested in, we have to ensure they can filter by most recently listed or by the price.

System Setup: The website and database are both hosted on our VPS and is accessible through any internet connection. In order to be able to test the features with users, the user must have an up to date modern web browser. Chrome, Firefox, and Safari are all approved browsers.

Starting point: The user must start at the home page, http://sfsubuyandsell.xyz/ , and start the tests from there. The state of the user being logged in or out does not matter. The test can be done either way.

The intended users: The intended user are SFSU students with basic internet usage skills. Any non computer science/engineering major student is eligible.

What will be measured: We will measure the user's satisfaction with the ease of use and efficiency for searching an item and filter the results.

Usability Task Description:

We will measure effectiveness if the test results are the same as the successful completion criteria. We will also look at the users response to our Lickert scale questions to ensure user satisfaction is above average.

We will measure efficiency by measuring the time it took the user to complete the task and compare it to our expected benchmark time. We will also look at the number of clicks it takes the user to complete the task and ensure they are standard across all user tests. The lower number of clicks the more efficient the task is.

Task 1: Perform a search to ensure proper results are returned relating to the search query.

Task	Description
Task	Perform a search for an iphone
Machine State	User starts on the homepage http://sfsubuyandsell.xyz/
Successful completion criteria	User performed the search and listings for iphones are returned.
Benchmark	Successful completion in under 30 seconds.

Task 2: Perform a search and ensure that the correct listings for that item pop up in a modal.

Task	Description
Task	Search for another item the user is interested in and click on one of the listings
Machine State	User starts on the homepage http://sfsubuyandsell.xyz/
Successful completion criteria	The listing the user clicked on will pop up on the screen providing more details on that listing.
Benchmark	Successful completion in under 30 seconds.

Task 3: Perform a search and test the filter mechanise to ensure all searched items are filtered properly.

Task	Description
Task	Search for “book” and filter by price high to low
Machine State	User starts on the homepage http://sfsubuyandsell.xyz/
Successful completion criteria	Multiple book listings are shown to the user and sorted by the high price first and the lowest price last.
Benchmark	Successful completion in under 45 seconds.

Lickert Scale Questionnaire:

1) I found the search for the iphone to be easy and intuitive:

_Strongly Disagree _Disagree _Neutral _Agree _Strongly Agree

Comments:

2) Clicking on a listing was intuitive and easy:

_Strongly Disagree _Disagree _Neutral _Agree _Strongly Agree

Comments:

3) Filtering listings by price was intuitive and easy:

_Strongly Disagree _Disagree _Neutral _Agree _Strongly Agree

Comments:

3) **QA Test Plan**

Test objectives: We will be performing QA testing on our search and filtering features. It is important to make sure that both are working as expected because they are crucial to the user being able to find exactly what they are looking for. We will be testing different kinds of input to the search bar and will be using the filter feature to filter said inputs. Any reference to the main homepage is referring to the base URL of www.sfsubuyandsell.xyz

HW and SW: We will be performing these QA tests on our website which is hosted on our VPS. We will also be using the latest stable Google Chrome and Firefox builds as our browsers of choice.

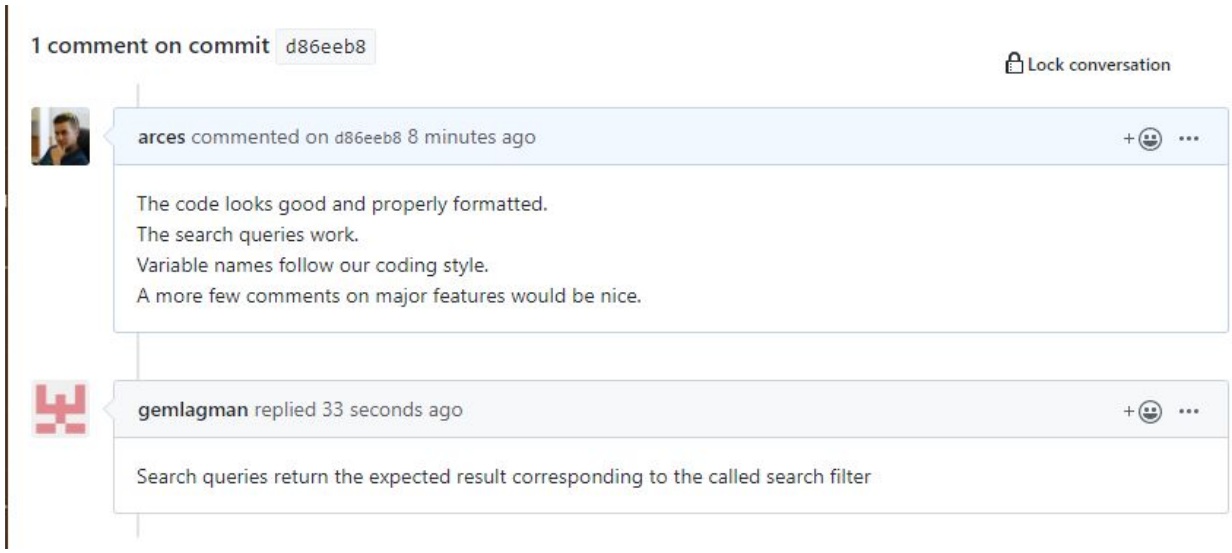
Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Listing Search	A test to confirm the correct number of items being returned and listed.	Enter “book” into the search bar. Click the search button to the right of the search bar	13 Listings will appear all with the text “book“ somewhere in their listing title or product description.	Chrome: Pass FireFox: Pass
2	Listing filter	A test to confirm that the filter feature is correctly	Enter “book” into the search bar. Click the	13 Listings will appear. The first listing will have	Chrome: Pass FireFox: Pass

		displaying the most expensive listings first.	search button. Select “Price: High to Low” from the filter drop down. Click the “filter” button.	a price of $1800 and the last listing will have a price of $1.	
3	Malformed Listing Input	A test to ensure that the search bar accepts any input and can properly handle it without the website returning an error.	Enter “dog123!*%” into the search bar. Click the search button.	0 Listings will appear and the website will not return any kind of error because it will filter it out and not show any results.	Chrome: Pass FireFox: Pass

4) Code Review

a) We are using Google’s JavaScript Style guide as our preferred style when coding our website.

b)




```
async function sortListingByPriceHighToLow(req, res, next) {
  let query =
```

```
      "SELECT listing.id, listing.title, listing.price, listing.description, listing.image,
listing.is_sold, listing.date, category.name, listing.user_id FROM listing INNER JOIN
category ON listing.category_id = category.id WHERE is_sold = 0 ORDER BY price
DESC LIMIT 9;";

   await db.execute(query, (err, results) => {
      if (err) {
         req.searchResult = "";
         next();
      }
      req.searchResult = results;
      next();
   });
}
async function sortListingByPriceLowToHigh(req, res, next) {
   let query =
      "SELECT listing.id, listing.title, listing.price, listing.description, listing.image,
listing.is_sold, listing.date, category.name, listing.user_id FROM listing INNER JOIN
category ON listing.category_id = category.id WHERE is_sold = 0 ORDER BY price Asc
LIMIT 9;";

   await db.execute(query, (err, results) => {
      if (err) {
         req.searchResult = "";
         next();
      }
      req.searchResult = results;
      next();
   });
}

//search
async function getClasses(req, res, next) {  async function getClasses(req, res, next) {
  await db.execute("SELECT * FROM class", (err, classes) => {          await
db.execute("SELECT * FROM class", (err, classes) => {
   if (err) throw err;        if (err) throw err;
@@ -94,6 +121,45 @@ async function search(req, res, next) {
}
}
```

5) Self-check on best practices for security

1) List of major assets to protect
    a) Listings
    b) Messages
    c) User information

    d) Images

2) How we protect each asset
    a) Database access is locked through a password. Search bar is limited to 40 alphanumeric characters in order to prevent SQL injection.
    b) Messages are private and are only able to be seen by the two users involved in the message. No public commenting system for listings.
    c) Plain text passwords given by users during signup are hashed using Bcrypt and hashed passwords are stored in the database. Users are not able to see the email of other users even if they are messaged by that user. The routes that provide access to posting, going on the dashboard, and messaging are protected by userId so users are only able to access their own information if they are logged in.
    d) Images uploaded by users are only stored on the server and not uploaded to github.
3) Password encryption?
    a) Yes, bcrypt used to hash passwords given by user during signup.
4) Input data validation?
    a) Yes, search limited to 40 alphanumeric characters through the use of bootstrap's built in functionality.

6) 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 (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. Selected application functions must render well on mobile devices **DONE**
4. Data shall be stored in the team's chosen database technology 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 and all privacy policies will be appropriately communicated to the users. **DONE**
7. The language used shall be English. **DONE**
8. Application shall be very easy to use and intuitive. **DONE**
9. Google analytics shall be added. **DONE**
10. No e-mail clients shall be allowed. **DONE**
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI. **DONE**
12. Site security: basic best practices shall be applied (as covered in the class). **DONE**
13. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. **DONE**
14. The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2019. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). **DONE**