SW Engineering CSC648/848 Fall 2020

GatorMart

Team 6

Lothar Narins	Team Lead
Aaron Colmenares	Backend Lead
Allyson Leung	Frontend Lead
Wilson Young	GitHub Master
Saloni Bhatia	Backend
Kevin Nunura	Frontend

lnarins@mail.sfsu.edu

Milestone 4

December 2, 2020

Date Submitted:	December 8, 2020
Date Revised:	December 10, 2020

Table of Contents

Product Summary

Product Name: GatorMart

Major Committed Functions: (In Progress)
- All users shall be able to browse items for sale.
- All users shall be able to register with their unique SFSU email address.
- All users shall be able to log in with their registered account to message other sellers.
- All users shall be able to select an item for sale to read more information on the product.
- All users shall be able to browse the website utilizing categorized filters within the search bar.
- Registered users inherit all functionality allowed for unregistered users.
- Registered users shall be able to see and read their messages through their user dashboard.
- Registered users shall be able to privately message a user for an item they are interested in.
- Registered users shall be able to post items for sale or request specific products, and such posts will be reviewed by the website administrator.
- Registered users shall be able to log out of their account.
- The website administrator shall inherit all functionality allowed for non-registered and registered users.
- Administrators shall be required to use MySQLWorkbench to manage the website.
- Administrators shall be able to approve or deny post submissions from registered users.

URL: http://3.134.106.28:3000

Usability Test Plan

Test Objective

The main objective of this usability test is to assess and evaluate the usability of our website GatorMart. The major function that is being tested for usability is our search function. We are choosing the search function because we feel it is a major function that is crucial for navigating our website. The usability test is used to determine if our website is user friendly and fast. The goal of this test is to evaluate the user's ability to search for items using the search function and to filter their results through our category filter. We are expecting feedback from testing to improve our website's user friendliness.

Test Background and Setup

The test is designed to evaluate GatorMart's ability to direct users to their desired product using the search function. The website must be able to run in the most commonly used browsers, such as Firefox, Internet Explorer, and Google Chrome. The intended users are primarily San Francisco State University students and staff or faculty.

To start the usability test, we first open up Google Chrome and navigate to the homepage of GatorMart. We then give the user a list of tasks to complete without giving them any information on how to accomplish the task. After the user completes the tasks, we will use the criterias in the Usability Task Description table and the Likert test to determine if the function passed the test.

The URL of the website is http://3.134.106.28:3000/ which will take you to the homepage of the website. The user can input the valid search query into the search bar which includes the class ID or the title of the post. Users can also filter through results by the category of the item utilizing the category drop down list or sort the items by price. The usability test is successful if the user is able to use the search function to find the specified item by searching the proper term in the search bar. When the correct items load onto the page according to the user's search criteria, we can consider the test a success.

Usability Test Description

Task	Description
Task	Search for items in MATH325
Machine State	Search listings loaded
Successful Completion Criteria	Shows related search results according to the search query
Benchmark	Complete within 12 seconds

Effectiveness : We would measure effectiveness by looking at the percentage of users who completed the task within the time limit. We would also take into account any errors that the user encountered while navigating to the search bar and entering their query, while also listening to the feedback given.

Efficiency : We would measure efficiency by the average time it took for users who completed the test and users who attempted the test while eliminating outliers. Efficiency could also be measured by the amount of clicks it took to complete the task.

("**X**" applicable column)

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
I was able to find the search bar easily.					
I was able to search for the specified item.					
I found a related search result.					

Comments and Feedback (Optional):

QA Test Plan

Test objectives:
Assessing if the search function performs per specs.
HW and SW setup:
URL: http://3.134.106.28:3000/
Feature to be tested:
Search function

QA Test Plan:
Browsers used in tests: Chrome and Firefox

Test #	Test Title	Description	Input	Expected output	Chrome	Firefox
1	searching by class id	Test whether search by class id works	Enter "MATH325" into search bar and press search	Should show 1 result: "Linear Algebra Textbook"	PASS	PASS
2	searching by category	using category dropdown menu	Select "Supplies" in dropdown menu next to search bar, leave search bar empty, press search	Should show 2 results: "50 index cards", "Slides for microscope"	PASS	PASS
3	searching by title/ item name	Test if text field search works for title/item name	Enter "book" into search bar and press search	Should show 2 results: "Howl on Trial Book", "Linear Algebra Textbook"	PASS	PASS

Code Review

For the code review, the marketplace.html page was sent to another team member. This page has the code for the feature used for the QA and usability test, which is the search function. The Html file includes not only the structure for the entire page but also the javascript code for the search function itself. Comments explaining the code are embedded and below are the comments made by the team member who reviewed the file.

Note: The peer reviewer marked their comments with a "@Todo" prefix

```
25   <body class="body-color">
26       <div class="border alert-warning d-flex justify-content-center" role="alert" style="font-weight: bold; padding-bottom
27           SFSU Software Engineering Project CSC 648-848, Fall 2020. For Demonstration Only
28       </div>
29       @TODO(Lothar): Fix spelling on [background theme]          <----
30       <!-- [navbar] [theme] [backrgound theme]-->
31       <nav class="navbar navbar-expand-sm navbar-color">
32           <!-- Logo -->
33           <a class="navbar-brand" href="/">
34               @TODO(Lothar): This old code could probably go          <----
35               <!-- <i class="fa fa-cube wid" aria-hidden="true"></i>
36               GatorMart -->
37               <!--Below is for GatorMart actual logo when available-->
38               <!-- <img src="/docs/4.0/assets/brand/bootstrap-solid.svg" width="30" height="30"
39                   class="d-inline-block align-top" alt="">
40               GatorMart -->
41               <img src="../Images/Logo/gatormart_logo.svg" height="40" class="d-inline-block align-top" alt="GatorMart" />
42           </a>
```

```
50           <div class="collapse navbar-collapse" id="navbarMenu">
51               @TODO(Lothar): Make the categories come from the database as in the other pages          <----
52               <!-- Search bar code -->
53               <form class="mx-5 d-inline w-100" id="search-parts" method="GET" action="/HTML/marketplace.html">
54                   <div class="input-group">
55                       <select class="form-control search-slt col-sm-2" id="category" name="category" onchange="filteredSearch();">
56                           <option value="All">All</option>
57                           <option value="Books">Books</option>
58                           <option value="Furniture">Furniture</option>
59                           <option value="Supplies">Supplies</option>
60                           <option value="Textbooks">Textbooks</option>
61                           <option value="Tutoring">Tutoring</option>
62                           <option value="Item Request">Item Request</option>
63                       </select>
```

```
95                   @TODO(Lothar): This is fine to remove. It is unlikely to come back, and we can dig it out of old commits on GitHub if necessary for reference.
96                   <!--Filters COL div-->
97                   <!--<div class="col-sm-2">
98                       <div class="border filters-card shadow-sm form-color">
99                           <h6>Filters</h6>
100                          <hr class="featurette-divider my-0">
101                          <div class="form-horizontal my-2">
102                              <div class="form-group" id="search-type">
103                                  <label for="typeLabel">Category</label>
104                                  <select class="form-control" id="category" name="category" onchange="filteredSearch();">
105                                      <option value="All">All</option>
106                                      <option value="Books">Books</option>
107                                      <option value="Furniture">Furniture</option>
108                                      <option value="Supplies">Supplies</option>
109                                      <option value="Textbooks">Textbooks</option>
110                                      <option value="Tutoring">Tutoring</option>
111                                  </select>
112                              </div>
113                              <div class="form-group">
114                                  <label for="classLabel">Class ID</label>
115                                  <input type="text" class="form-control" id="classLabel" onchange="filteredSearch();" placeholder="e.g. CSC648">
116                              </div>
117                          </div>
118                      </div>
119                  </div>-->
120                  <!--Results COL div-->
```

```
@TODO(Lothar): Remove old code
<!--<button class="border btn-sm btn-light dropdown-toggle float-right" id="sortby" type="button"
    data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Sort By</button>-->
```

```
@TODO(Lothar): This function could have a more specific name, e.g. parseURLQuery
//This function processes the important information that is grabbed from the URL and then calls actual search function
function parse() {
    var url = document.location.href,
        params = url.split('?')[1].split('&'),
        data = {}, tmp;
    for (var i = 0, l = params.length; i < l; i++) {
        tmp = params[i].split('=');
        data[tmp[0]] = tmp[1];
        tmp[1] = tmp[1].replace(/\+/g, " ");
        //console.log(tmp[0]);
        setStuff(tmp[0], tmp[1]);
    }
    filteredSearch();
}
```

```
@TODO(Lothar): This function could have a better name, and why does it have an if statement if it does the same thing in both cases? Could be cleaned up.
//This function is used to set values to their correct elements, can be expanded if more filters are needed
function setStuff(name, val) {
    if (name == "category") {
        document.getElementById("category").value = val;
    }
    else {
        var holder = document.getElementById(name);
        holder.value = val;
    }
}
```

```
@TODO(Lothar): Feel free to remove the commented out declarations of type and classID. Perhaps display some error message to the user on error, instead of just console.log?
/*
    This function processes all of the information for the search
    it grabs info from the page and creates the result text that is displayed with the results
*/
function filteredSearch() {
    let searchTerm = document.getElementById('searchtext').value || '__NO_VALUE__';
    let category = document.getElementById('category').value;
    // let type = document.getElementById('typeSelect').value;
    // let classID = document.getElementById('classLabel').value || '__NO_VALUE__';
    let searchOrder = document.getElementById('sortBy').value;
    // console.log(category + ', ' + type + ', ' + classID + '.');
```

Self-check on Best Practices for Security

Major Assets	Threats	Protection
User records	Unauthorized user gains access to confidential data	1. Require users to authenticate themselves 2. Track system usage 3. Needs access to the database by connecting it to the server by SSH 4. Requires input validation
Posts	Unauthorized user makes feature unavailable, scrambles post information or posts inappropriate content	1. Require users to authenticate themselves 2. Track system usage 3. Needs access to the database by connecting it to the server by SSH 4. Requires input validation
Messages	Unauthorized user makes feature unavailable, distorts messages and can also gain access to personal information shared between buyer and seller	1. Require users to authenticate themselves 2. Track system usage 3. Needs access to the database by connecting it to the server by SSH 4. Requires input validation
Images	Unauthorized user makes feature unavailable, or can post incorrect or inappropriate images	1. Require users to authenticate themselves 2. Track system usage 3. Needs access to the database by connecting it to the server by SSH 4. Requires input validation 5. File system inaccessible without credentials

PW encryption in the DB: DONE

Confirm Input data validation; ON TRACK

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 (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   DONE

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   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   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).   DONE