SW Engineering CSC648/848
Milestone 4

Fall 2018

Team: 12
Project: GatorList
Date: 11/28/18
Version: 1.0

1. Product Summary

Name of product: GatorList

Functions:

1. Unregistered users shall be able to register providing name, surname, username, password.
2. Unregistered users shall be able to see the list of items.
3. Unregistered users shall be able to filter postings by categories.
4. Unregistered users shall be able to search for items.
5. Unregistered users shall be able to fill up information for a new item, such as 1 to 4 images, price and description of no more than 500 characters.
6. Registered users shall be able to login to existing accounts using his/her username and password.
7. Registered users shall be able to see the list of his/her items.
8. Registered users shall be able to filter items by categories.
9. Registered users shall be able to search for items.
10. Registered users shall be able to fill up information for a new item, such as 1 to 4 images, price and description of no more than 500 characters.
11. Registered users shall be able to submit an item for reviewing to Admin.
12. Registered users shall not be able to revise live posts; must be deleted first then submit new post.
13. Registered users shall be able to send a message to another user.
14. Admin shall be able to approve or deny an item.
15. Admin shall be able to see a list of all the items on the site.
16. Admin shall be able to see details about individual items.
17. Admin shall be able to see a list of all the registered users.
18. Admin shall be able to delete an item.
19. Admin shall be able to delete a user’s account.

URL: https://csc648team12.herokuapp.com/

2. Usability Test Plan

The search function shall be tested for usability.

1. Test Objectives
    a. Have the user search various entries in the search bar; i.e. book, chair, a color, ect
    b. Have the user change the category for another search
    c. Have the user check whether the search entry is retained after clicking search
    d. Have the user use search while being on one of the different links/routes (Home, About Team, Sell, Register, Login)
    e. Have the user type specific search entries; i.e. Use non-alphabetic characters
2. Test Plan

    This test plan is intended to see whether any person may use the search function with ease and check whether the search function is working correctly. To highlight any abnormalities with the search function, we shall instruct the user to search various entries, change the categories, and confirm whether the entries are retained after using search or clicking the other routes of the website. Lastly, to prevent injection or security risks, we will need to confirm that non-alphabetic characters in search queries are removed. These are the core elements of the search function. The search function is expected to be simple and straightforward. By having the user test all of these scenarios, we will have a better grasp of its usability.

3. Questionnaire

    Please check the statements that correspond to your experience with the search function.

    You were able to search for an item

    ❏ While being on the Homepage
    ❏ While being on the About Team page
    ❏ While being on the Sell page
    ❏ While being on the Login page

    You were able to change search categories

    ❏ While being on the Home page
    ❏ While being on the About Team page
    ❏ While being on the Sell page
    ❏ While being on the Login page

The search entries and category you selected remained the same after searching
- ❏ While being on the Home page
- ❏ While being on the About Team page
- ❏ While being on the Sell page
- ❏ While being on the Login page

- ❏ A search attempt returned no items or was blank
- ❏ The search results were relevant to the search's category
- ❏ If no items relating to the search terms were found, the page displayed that "no items were found"
    - ❏ Also, popular items were displayed instead of empty, white space

  After doing a search with non-alphabetic characters within the entry
    - ❏ The non-alphabetic characters are removed after searching

3. QA Test Plan

QA Test Objectives

Make sure that search functionality works as expected, there is no security issues related to unexpected strings as user inputs and that items displayed are correct.

HW and SW setup

We are testing on MacBook Pro with macOS 10.14.1. Used browsers are Google Chrome 70.0.3538.110 and Mozilla Firefox Quantum 63.0.3.

URL: https://csc648team12.herokuapp.com/

Feature to be tested

Just as in the usability test plan, we are going to design the QA test plan for Search.

QA Test Plan

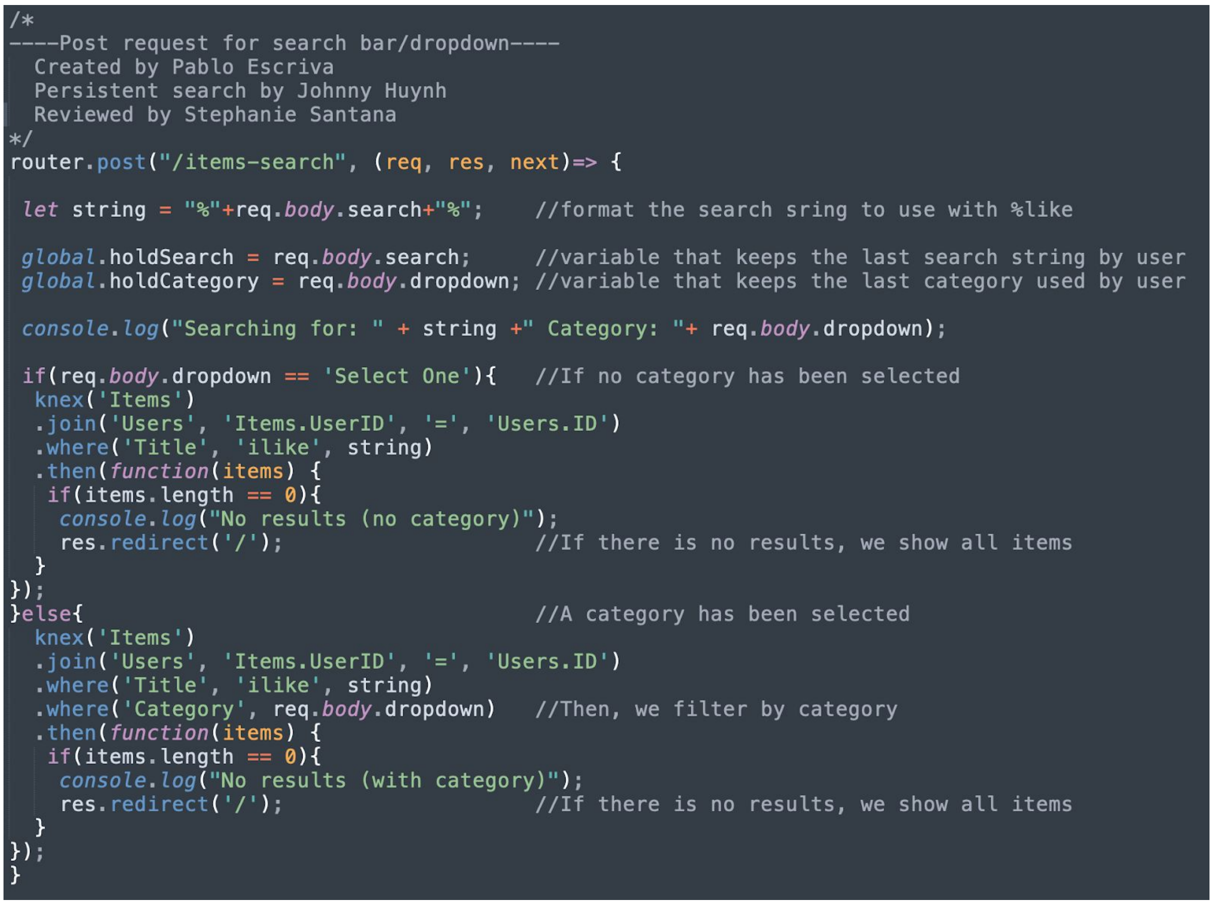
Test #	Title	Description	Input	Expected Correct Output	Results
1	Search with category	Search for an item and filter the results to only show “Electronics”	Search: “Laptop” Category: Electronics	Shows only 1 result (Laptop)	PASS
					PASS
2	Search for similar items	Search for an item with an incomplete string	Search: “Phy” Category: All	Shows only 1 result (Physics Book)	PASS
					PASS
3	Search for numerical string	Search for a number	Search: “1”	Shows all items	PASS
					PASS

4. Code Review

Coding Style

We try to write self-explanatory, we name the variables in a way that makes it easy to understand what we use them for. We use ES6 standard for javascript (let -> var, arrow functions...). We did not use header comments but we are working on this.
We only add code to Developer branch via pull request. Once there is a new feature that a team member has finished a new pull request needs to be created. Then, someone else pulls that changes and tests it locally. If everything works, the pull request is accepted and the changes are added to Developer. We only push to master before every milestone and we freeze the branch, master always has a working version of our page.

Code example



```
/*
----Post request for search bar/dropdown----
  Created by Pablo Escriva
  Persistent search by Johnny Huynh
  Reviewed by Stephanie Santana
*/
router.post("/items-search", (req, res, next)=> {

 let string = "%"+req.body.search+"%";    //format the search sring to use with %like

 global.holdSearch = req.body.search;     //variable that keeps the last search string by user
 global.holdCategory = req.body.dropdown; //variable that keeps the last category used by user

 console.log("Searching for: " + string +" Category: "+ req.body.dropdown);

 if(req.body.dropdown == 'Select One'){   //If no category has been selected
  knex('Items')
  .join('Users', 'Items.UserID', '=', 'Users.ID')
  .where('Title', 'ilike', string)
  .then(function(items) {
   if(items.length == 0){
    console.log("No results (no category)");
    res.redirect('/');                    //If there is no results, we show all items
  }
});
}else{                                    //A category has been selected
  knex('Items')
  .join('Users', 'Items.UserID', '=', 'Users.ID')
  .where('Title', 'ilike', string)
  .where('Category', req.body.dropdown)   //Then, we filter by category
  .then(function(items) {
   if(items.length == 0){
    console.log("No results (with category)");
    res.redirect('/');                    //If there is no results, we show all items
  }
});
}
```

Peer review

a) Pablo creates a new function to add functionality to the search bar and dropdown menu
b) Pablo creates a new Pull request named “Post request for search Bar/dropdown menu” and chooses Stephanie Santana to review his code. He sends a Slack message to her explaining the purpose of the code and how to test it
c) Stephanie pulls the code and tries it on her local machine, she confirms the code is working but asks pablo to add header comments and line comments for important parts of the code
d) Pablo adds comments, updates the pull request and Stephanie accepts it, the Developer branch now has the new changes

5. Self-check on best practices for security

Major protected assets:	username, first name, last name, password, item information (images, description, price)
PW encryption:	on track
Input data validation:	on track

6. 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 all major browsers: Mozilla, Safari, Chrome. **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. **DONE**
12. Site security: basic best practices shall be applied (as covered in the class) **ON TRACK**
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-Fulda Software Engineering Project CSC 648-848, Fall 2018. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). **DONE**