SW Engineering CSC648/848 Fall 2018

UniTrade

Team 10

Xinhuan Qiu (xqiu1@mail.sfsu.edu), Mario Marcos, Pohung Wang, Jianhong Kuang, Zachary Martin, Toluwanimi Oyewumi, Rong Tian Huang

Milestone 4

December 2, 2018

Created on

November 27, 2018

Revised on

1. **Product Summary:**

Name of product:

Unitrade

Final Functions:

1. Unregistered users shall be able to browse by category.
2. Unregistered users shall be able to use text search.
3. Unregistered users shall be able to search within category.
4. Unregistered users shall be able to sort by date and price.
5. Unregistered users shall be prompted to register or sign in upon confirming to post or buy an item.
6. Unregistered users shall be able to post and buy an item.
7. Unregistered users shall be able to register.
    7.1. Registration shall require at least a username, password, and email.
8. Unregistered users shall be required to accept terms and conditions upon registering.
    8.1. Terms and conditions shall be defaulted as unchecked.
    8.2. Users shall be required to accept this in order to create their account.
9. Unregistered users shall be able to see number of items displayed out of the total results.
10. Registered users shall have all of the functions of unregistered users.
11. Registered users shall be able to login.
    11.1. Login shall require an valid username and password.
12. Registered users shall be able to access their User Dashboard.
    12.1. Users shall be required to be logged in to access this dashboard.
13. User Dashboard shall show all of the items an user has created.
    13.1. Items displayed shall be sorted by the most recent posted date.
14. User Dashboard shall display the title, status (pending or approved), and link to each item.
15. User Dashboard shall contain all messages sent or received.
    15.1. Messages shall be separated by a sent and received section.
    15.2. Messages sent or received shall be sorted by most recent timestamp.
16. Registered users shall be able to create an item that is for sale.
    16.1. Users shall be prompted to confirm to post their item.
    16.2. Items shall not be visible on the site until approval.
    16.3. Items shall be of pending status after confirming to post.
17. Registered users shall be able to choose their item category, add title, description, images, and price for their items.
    17.1. Item category shall be chosen by a drop-down menu.
    17.2. Description shall be an optional field.

17.3. At least one image shall be required.
17.4. Images shall be of jpg format and have a maximum size of 3 MB.
17.5. One of the images shall be used as the thumbnail.
18. Registered users shall be able to delete their own items.
18.1. Users shall be able to delete their items from their dashboard.
18.2. Users shall be prompted to confirm when deleting their item.
19. Registered users shall be able to contact sellers.
20. Registered users shall be able to host an sale event.
21. Admins shall have all the functions of registered and unregistered users.
22. Admins shall be able to access the Admin Dashboard.
23. Admin Dashboard shall contain all messages created on site.
23.1. Messages received shall be sorted by post date.
24. Admin Dashboards shall contain items waiting for approval.
25. Admins shall be able to approve and disapprove items.
25.1. Disapproved items shall be removed from the site.
26. Admins shall be able to remove a registered user from the site.

What is Unique in Our Product:

Our product is exclusively designed to give ease to SFSU students by allowing them to buy items for classes and selling off what is no longer needed. We give our users the function to search for items for a specific class. This will return to the user all items which have the class ID or name in their title or description. Students are often busy with school and with their personal life, so it is necessary that we can provide an quick and simple way for them to get their needs.

Product URL:

http://onlinestore-env.us-west-1.elasticbeanstalk.com/

2. **Usability Test Plan:**

Test Objective:

The purpose of this test is to see if it is easy for a new user to search for an item in UniTrade. The design of UniTrade must be simple and easy to use for our target users which are mainly college students of San Francisco State University. Since our target users are often busy with classes, we prioritized making the process of searching and buying an item fast and simple.

We will choose a select students in SFSU to be the testers since they are our target audience. The testers will be given a task to search for a laptop. These students will be monitored and timed from the start of the task (UniTrade Index Page) till the completion of the task (Laptop is surfaced in results)

After the task is completed, they'll be given a quick survey regarding the overall user experience of the website. We will use the likert scale template to take the user input for each question, while leaving a section for comments or suggestions.

The results of the questionnaire will give us an idea of the overall functionality and efficiency of the design.

Test plan:

The tester will be provided a laptop to use with the site opened and loaded to the index page.

TASK	**DESCRIPTION**
Task	Search for a laptop.
Default State/Starting Point	Index Page
Successful Completion Criteria	Laptop surfaces in results.
Benchmark	Completed in 15 seconds.
URL	http://onlinestore-env.us-west-1.elasticbeanstalk.com/items/search/?q=laptop

Questionnaire:

UniTrade Search	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
It is easy to spot the search field.					
Searching for an item in Unitrade is easy.					
Searching for an item in UniTrade surfaces accurate results.					
The load time for search is fast.					
The image quality of the searched item is fantastic.					
The font size is perfect.					
I would recommend UniTrade to my friends.					

Comments/Suggestions:

3. **QA Test Plan:**

Test Objective:

Test: Search input response for regular keyword, spaces input, and special characters.
Not Test: Search with category filter.

HW/SW setup:

A computer device with functional keyboard, mouse cursor, monitor, and power.
wifi: connected, and be able to access internet.
url: http://onlinestore-env.us-west-1.elasticbeanstalk.com/
Browser1: Chrome
Browser2: Firefox

Feature to be tested:
% like search, special characters input response, and spaces input response.

Test Plan:

Test Number	1
Test Field	Search Bar
Test Title	Test search function for regular input
Test Description	Test the % like search function return related output result with regular input.
Test Procedure	1. Verify the Category dropdown shows "All" 2. Enter "bo" in input text field 3. Click the "Search" button
Expected Output	Get 2 items, which the 2 items are related to book
Result on Chrome	Get 2 items: "GRE Book Bundle" and "Introduction to Algorithm" which are related.
Result on Firefox	Get 2 items: "GRE Book Bundle" and "Introduction to Algorithm" which are related.
PASS/FAIL	PASS
Test Number	2
Test Field	Search Bar
Test Title	Test search function for space input
Test Description	Test search function return all results as no input while input contains only spaces.

Test Procedure	**Setup:** 1. Verify the Category dropdown shows "All" 2. Enter "bo" in input text field 3. Click the "Search" button 4. Check the 2 related results appear. **Test:** 1. Enter 5 spaces in input text field 2. Click the "Search" button
Expected Output	Get 8 items, which are the total items.
Result on Chrome	Get 8 items.
Result on Firefox	Get 8 items.
PASS/FAIL	PASS
Test Number	3
Test Field	Search Bar
Test Title	Test search function for special characters input
Test Description	Test search function return input invalid warning message while input contains special characters.
Test Procedure	1. Verify the Category dropdown shows "All" 2. Enter "bo%" in input text field 3. Click the "Search" button
Expected Output	"Please don't include special characters." message appears.
Result on Chrome	See the message "Please don't include special characters." appears.
Result on Firefox	See the message "Please don't include special characters." appears.
PASS/FAIL	PASS

4. **Code Review:**

The coding style we chose is the reStructuredText docstring format. We use github to perform code reviews by adding comments to the pull requests. An issue is created and then assigned to a group member, once the group is done working on a task, they create a pull request and then request a review from either the team lead or front end/back end lead. The code reviewer then goes through the code and tests it on their local machine to make sure it's functioning, bug-free and formatted accordingly. If any issues are found within the code, the reviewer comments on the pull request asking the coder to make required changes. Once changes are made and the code reviewer is content with the changes the pull request is then approved, then the team member can merge the pull request.

The code below is for sorting searched items:

```
<div class = "row">
        <!--Sort-->
        <div class = "col-10">
            <div class = "float-left">
            {% if num_items > 0 %}
            <h5>{{num_items}} results</h5>
            {% endif %}
        </div>
        </div>
        <div class ="col">
            {% if num_items > 0 %}
            <form id ="sortform">
                 
            <div class ="float-right">
            Sort by:
          <select name = sort_value>
              {% if picked_sort == "price"%}
                <option value="price">Lowest Price</option>
                <option value = "-create_time">Most Recent</option>
              {% else %}
                <option value = "-create_time">Most Recent</option>
                <option value = "price">Lowest Price</option>
              {% endif %}
          </select>
                {% if search_keyword != None %}
                <input type = "hidden" name="q" value ="{{search_keyword}}">
                {% endif %}
                <button type="submit">Sort</button>
            </div>
            </form>
            {% endif %}

            </div>
        </div>
```

Below are the comments made by the reviewer regarding bugs and suggested changes

I found two issues:

- After you select "Lowest Price" and click Sort, the selection changes back to "Most recent" (Can you capitalize the "r" in "recent") even though the items have been sorted by price. Can you please persist the selection so that the user knows what the current sorting is?
- When you search by text and then sort, the sorting is implemented on all items instead of the search result items. Can you sort only the search results?

Something to improve if you had time (not necessary):

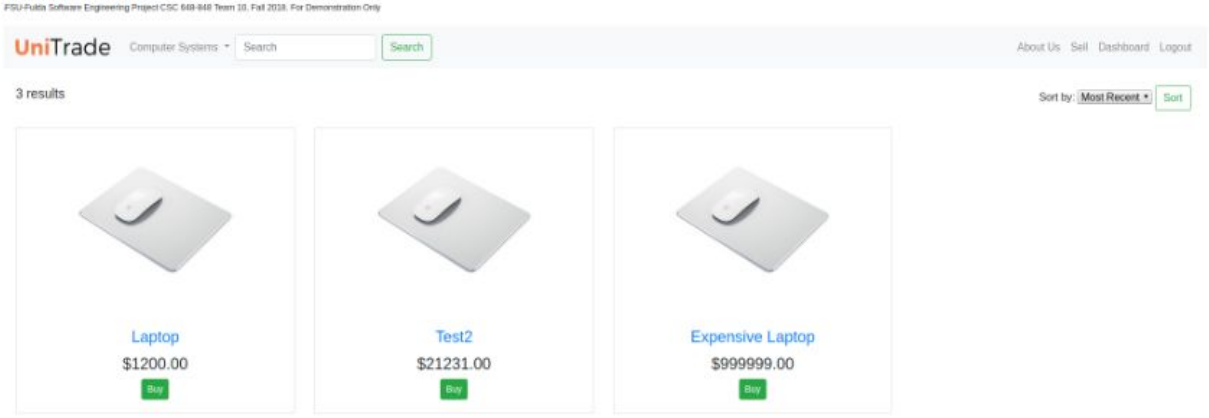
- Can you remove the sort button so that the results will be sorted just by choosing an option, like how category works?

Once the team member makes changes, they push their code and comment on the pull request with a screenshot to corroborate the changes.

---

fixed issue

see reference:



5. **Self-check on best practices for security:**

Major assets being protected:

- User information - passwords, emails
- Posts of Items

Password encryption:

- Passwords are encrypted via Django built-in function

Input data validation:

- Search bar input is validated. Any input with special characters other than alphabets and numbers are considered invalid. Also, the search bar input cannot exceed 40 characters.
- {% if search_keyword|length < 40 and has_special_characters == None%} is used in template to check if the number of characters is less than 40.
- The search_string() method in veiws.py of our onlinestore app is used to check if there's any special characters in the search input.

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, which include PostgreSQL 9.4.17, Apache 2.4.33 and Python 3.6.4. **DONE**
2. Application shall be optimized for standard desktop/laptop browsers and shall 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 Amazon Web Services. **DONE**
5. File size in no time shall exceed 3 MB. **ON TRACK**
6. No more than 50 concurrent users shall be accessing the application at any time. **DONE**
7. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **DONE**
8. The language used shall be English. **DONE**
9. Application shall be very easy to use and intuitive. **DONE**
10. The Systems visual response time shall be within 5 seconds. **DONE**
11. Google analytics shall be added to analyze visitor data. These include visitor clicks, name and email of registered users which shall ONLY be used for google analytics. **DONE**
12. No email clients shall be allowed. **DONE**
13. Pay functionality shall not be implemented nor simulated. **DONE**
14. Site security: basic best practices shall be applied (as covered in the class). **DONE**
15. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. **DONE**
16. 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 corner and the website logo in the navbar. (Important so as to not confuse this with a real application). **DONE**