SW Engineering CSC648 / 848 Spring 2022

BusyGator

Team 04		
Role	**Name**	**Email**
Team Lead	Samantha Saxton-Getty	ssaxtongetty@mail.sfsu.edu
Github Lead	Vishal Ramanand Sharma	vsharma5@mail.sfsu.edu
Front End Lead	Elyssa Mari Tapawan	etapawan1@mail.sfsu.edu
Back End Lead	Aaron Carlson	acarlson8@mail.sfsu.edu
Front End	Abdullah Sharaf	fabdullah1@mail.sfsu.edu
Front End	Siqi Guo	sguo4@mail.sfsu.edu
Back End	Janvi Patel	jpatel6@mail.sfsu.edu

May 16, 2022

Milestone 04

Date	Version
05.16.22	Finished Version 01 submitted

Table of Contents

1. Product Summary

Application Name: BusyGator

Our application, BusyGator, provides various materials to help SFSU students, staff, and faculty find everything they are looking for on campus all online. BusyGator is an e-commerce application that allows students, staff, and faculty to buy different items, from school materials to gym equipment, as well as list items for others to buy. The BusyGator has numerous functionalities and services. The main service the application will provide is allowing SFSU students, staff and faculty to post and / or buy textbooks, sports gear, research devices, and much more all online. The application has search options and categories to look for more specific items, and a sign up / sign in option to confirm users are SFSU students or faculty. BusyGator gives SFSU an opportunity to help their community by managing their resources within the application. The unique aspect of BusyGator is that all of the features and services in buying and selling material will be in one place and in one press of a button, where anyone can easily follow.

Priority 1 Feature List:

Unregistered, Registered, and Administrator Users:

1. View Listings and Marketplace
2. Search Listings and Marketplace
3. Filter Listings and Marketplace

Unregistered Users:

4. Register Account

Registered Users:

5. Login Account
6. Post Listing on Marketplace
7. Message Seller to receive Approval or Denial of interest

Administrator Users:

8. Remove User
9. Remove Listing
10. Approve or Deny Listings

URL: http://3.23.79.193/

2. Usability Test Plan

Test Objectives:

Our usability test plan's objective is to test the users satisfaction when posting data to our website, BusyGator. Since BusyGator is an e-commerce marketplace that allows users to post many items, we can expect this test to help determine which areas of posting need to be optimized. This test will ensure we can implement areas of improvement currently and in the future.

Test Background and Setup:

System Setup:

Any browser on a mobile device, laptop, or PC with access to the internet.

Starting Point:

Any user that is already registered within BusyGator, but has yet to login.

Intended Users:

Any San Francisco State University faculty, staff, and students who have basic knowledge and understanding of how to operate a browser on a mobile device, laptop, or PC.

URL: http://3.23.79.193/

Usability Task Description:

Please open a browser and navigate to BusyGator. Once there, log into your BusyGator account. After logging in, navigate to the post page and fill out the prompted information on the screen. Once this is complete, please submit your post.

Evaluation of Effectiveness:

In order to evaluate the effectiveness of posting an item, we need to measure the success rate of users that complete the following tasks successfully (see below). In addition to completing and recording these tasks, we need to record any errors that happened during the process.

Test (Use) Case	Percent Completed	Errors Encountered	Comments
Navigate to BusyGator Website	100	N/A	N/A
Login	50	Unable to Login.	This feature is not fully complete.
Navigate to the	100	N/A	N/A

Post Page			
Fill out the Form	100	N/A	N/A
Upload a Picture	100	N/A	N/A
Submit the Form	100	N/A	N/A

Evaluation of Efficiency:
In order to evaluate the efficiency of posting an item, we need to measure the time it takes for the user to complete each task, the number of errors encountered, as well as the number of clicks the user performs. These will be measured through the following tasks (see below).

Test (Use) Case	Completion Time (Seconds)	Number of Errors Encountered	Number of Clicks Performed
Navigate to BusyGator Website	3	N/A	3
Login	5	Unable to Login.	4
Navigate to the Post Page	1	N/A	1
Fill out the Form	15	N/A	8
Upload a Picture	4	N/A	3
Submit the Form	1	N/A	1

Evaluation of User Satisfaction:
In order to evaluate the user satisfaction of posting an item, we need to direct the focus group to this survey below.

Task	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Navigating to the Post Page was easy	✔				
The mandatory sections were marked clearly	✔				

I was able to complete the task in a reasonable amount of time	✔				

3. QA Test Plan

Test Objectives:

The objective of the test is to verify the functionality of making posts works according to the specifications.

- When the user wants to make a new post, the user can fill out the required information needed to make a new post and press the post button to post their product. When submitting a post, the post will show up on the home page along with the information the seller filled out.
- The information the user filled out in the new post should be recorded in the database.
- All required information within the new post should be filled, if not then it should produce an error.

The final outcome of the test would be a new post of the product with the correct details and data of the new post being recorded into the database.

HW and SW Setup (including URL):

HW Setup: A mobile device, laptop, and/or computer that is connected to the internet.
SW Setup: Google Chrome or Firefox browser.
URL: http://3.23.79.193/

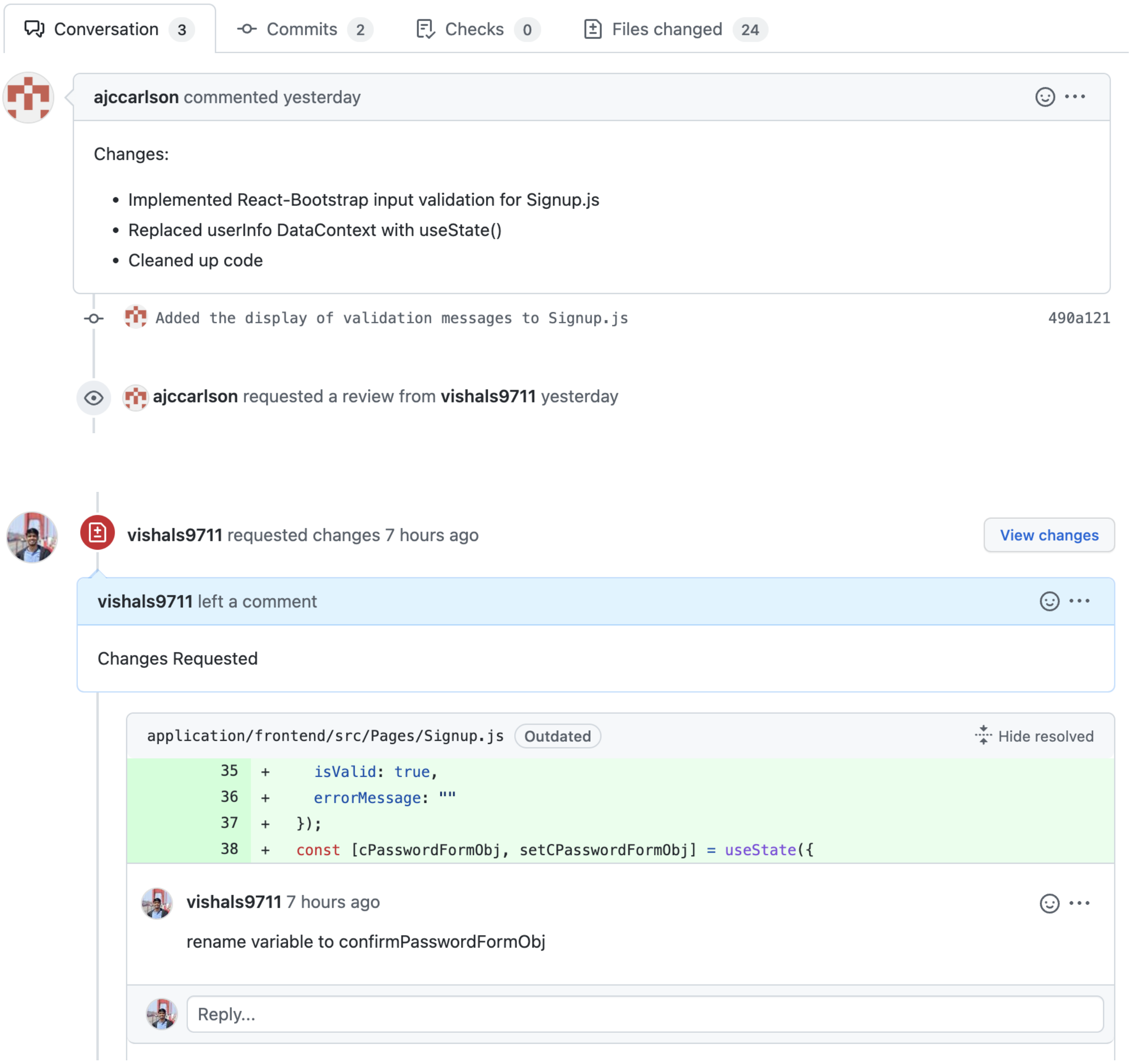
Feature to be Tested:

- Making a new post.
  - The user must input all the required data within the new post, if left empty then errors will display under the mandatory fields.
  - When all the fields are filled correctly, the user can press the post button and it will be posted onto the home page where it displays the product and its details.
- The post data will be stored in the database.
  - All the data the user put in should be stored correctly into the database.
  - The product title, price, location, description, seller, and date listed should be recorded in the database.

QA Test plan:

Test #	**Test Title**	**Test Description**	**Test Input**	**Expected Correct Output**	**Test Results (Firefox)**	**Test Results (Google Chrome)**
1	Inputting no fields in the New Post form	When the user inputs no fields and attempts to post by clicking the post button.	1. Keep the fields empty. 2. Press the Post button.	Error messages should appear under the mandatory fields labeled by asterisks reminding the user to fill out the required fields.	Pass	Pass
2	Filling all inputs to the New Post form	When the user completes all the fields on the New Post form and post by clicking the post button.	1. Fill in all the fields on the New Post form correctly. 2. Press the Post button.	The user should be redirected to the home page while the fields that were filled are waiting to be approved by the admin to be posted.	Pass	Pass
3	Canceling the New Post fields	When the user fills out one or more fields on the New Post form, they will be able to cancel the post.	1. Fill in one more field on the New Post form correctly. 2. Press the cancel button.	The forms that were filled out should be cleared.	Fail	Fail
4	Data recorded in the database	When the user fills out the forms on the New Post correctly and presses the Post button, the data is stored within the database.	1. Fill in all the fields on the New Post form correctly. 2. Press the post button. 3. Check the database to see all the post data.	All the post data is stored correctly into the database.	Pass	Pass

4. Code Review



ajccarlson commented yesterday

Changes:

- Implemented React-Bootstrap input validation for Signup.js
- Replaced userInfo DataContext with useState()
- Cleaned up code

```
Added the display of validation messages to Signup.js                490a121
```

ajccarlson requested a review from vishals9711 yesterday

vishals9711 requested changes 7 hours ago

vishals9711 left a comment

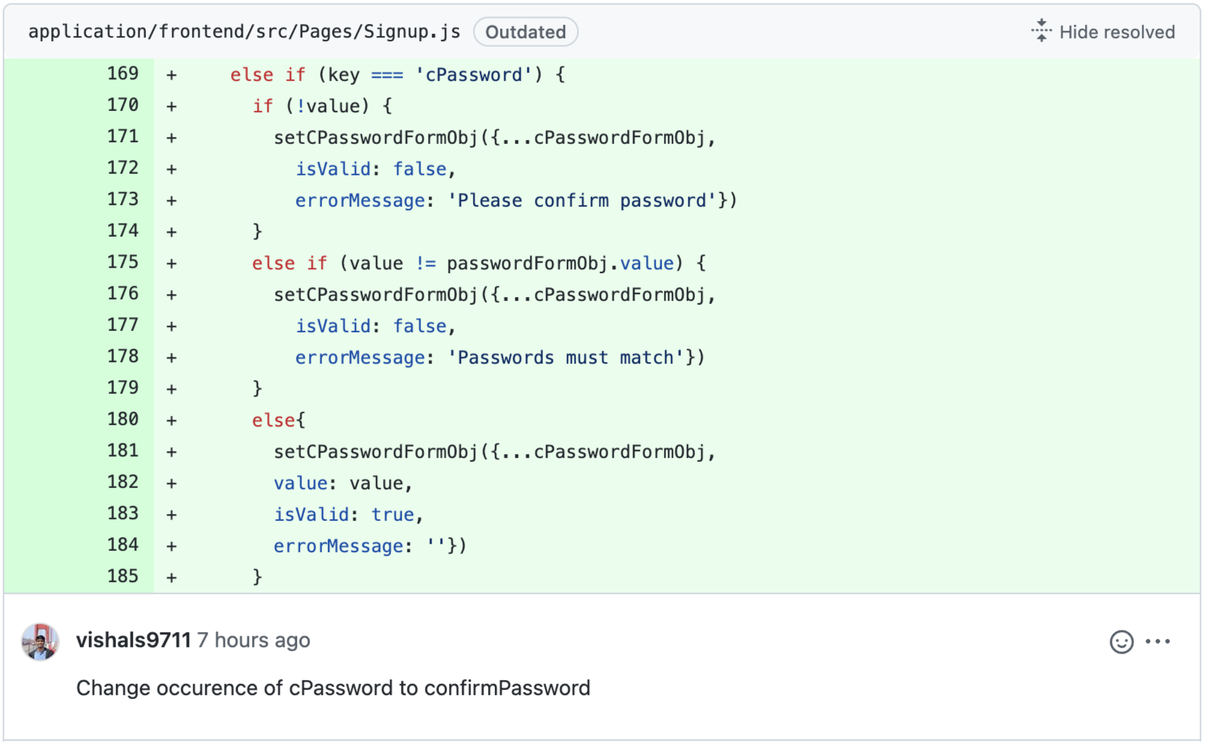
Changes Requested

application/frontend/src/Pages/Signup.js Outdated

```
+    isValid: true,
+    errorMessage: ""
+  });
+  const [cPasswordFormObj, setCPasswordFormObj] = useState({
```

vishals9711 7 hours ago

rename variable to confirmPasswordFormObj

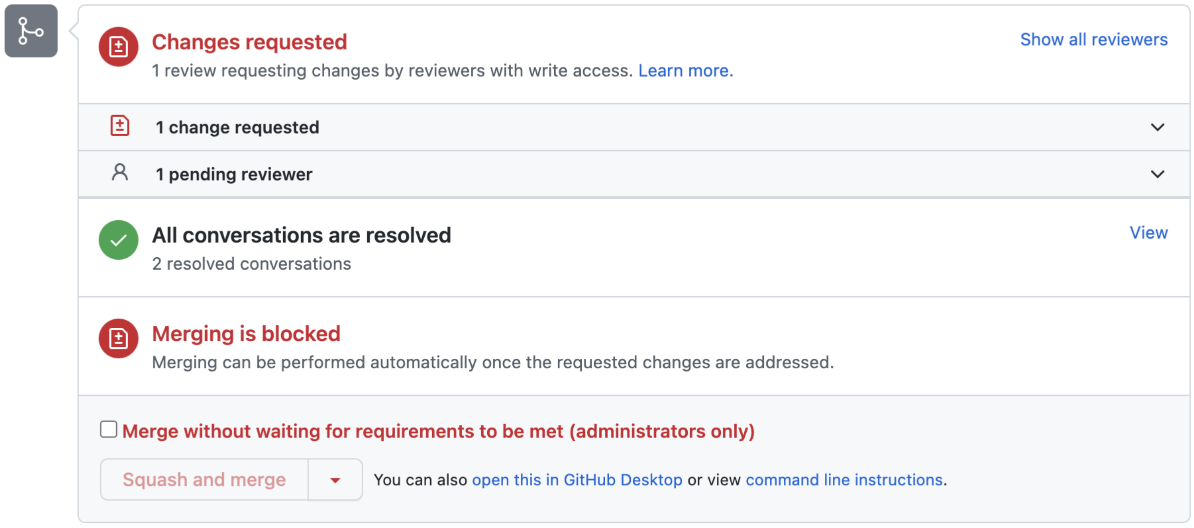


application/frontend/src/Pages/Signup.js Outdated Hide resolved

```
+     else if (key === 'cPassword') {
+       if (!value) {
+         setCPasswordFormObj({...cPasswordFormObj,
+           isValid: false,
+           errorMessage: 'Please confirm password'})
+       }
+       else if (value != passwordFormObj.value) {
+         setCPasswordFormObj({...cPasswordFormObj,
+           isValid: false,
+           errorMessage: 'Passwords must match'})
+       }
+       else{
+         setCPasswordFormObj({...cPasswordFormObj,
+         value: value,
+         isValid: true,
+         errorMessage: ''})
+       }
```

vishals9711 7 hours ago

Change occurence of cPassword to confirmPassword



Changes requested

1 review requesting changes by reviewers with write access. Learn more.

Show all reviewers

1 change requested

1 pending reviewer

All conversations are resolved

2 resolved conversations

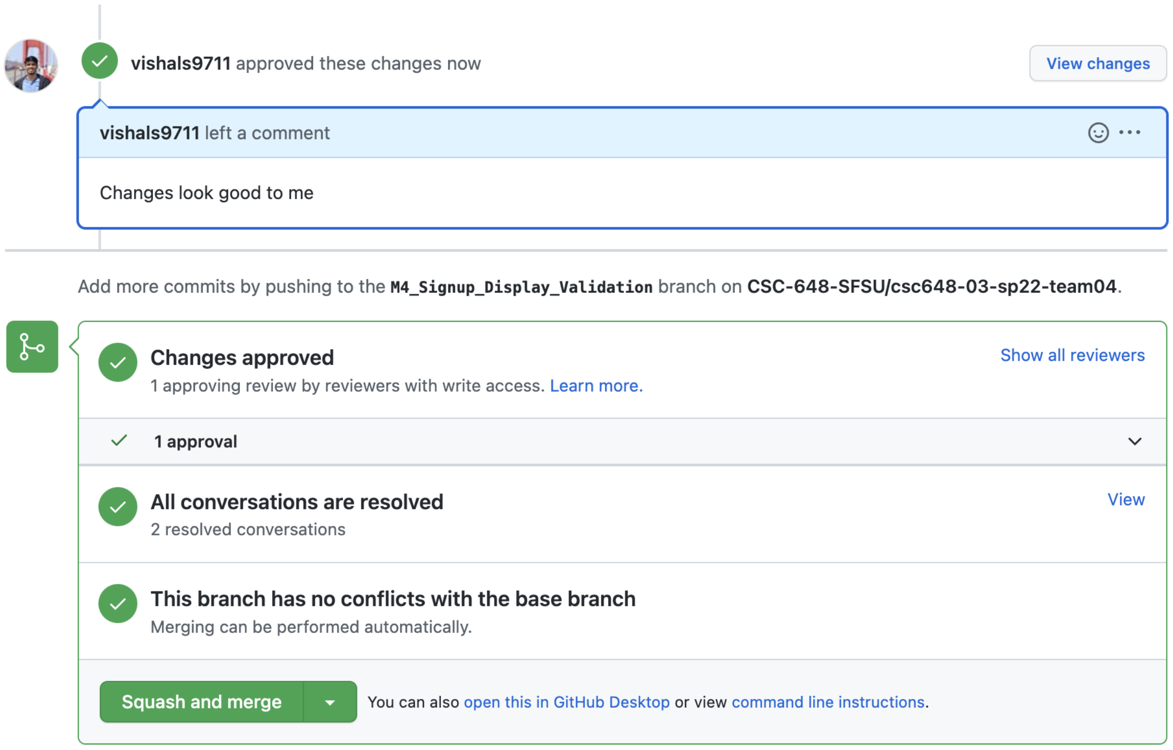
View

Merging is blocked

Merging can be performed automatically once the requested changes are addressed.

☐ **Merge without waiting for requirements to be met (administrators only)**

Squash and merge ▾ You can also open this in GitHub Desktop or view command line instructions.


vishals9711 approved these changes now

View changes

vishals9711 left a comment

Changes look good to me

Add more commits by pushing to the `M4_Signup_Display_Validation` branch on **CSC-648-SFSU/csc648-03-sp22-team04**.

Changes approved

1 approving review by reviewers with write access. Learn more.

Show all reviewers

✓ 1 approval

All conversations are resolved

2 resolved conversations

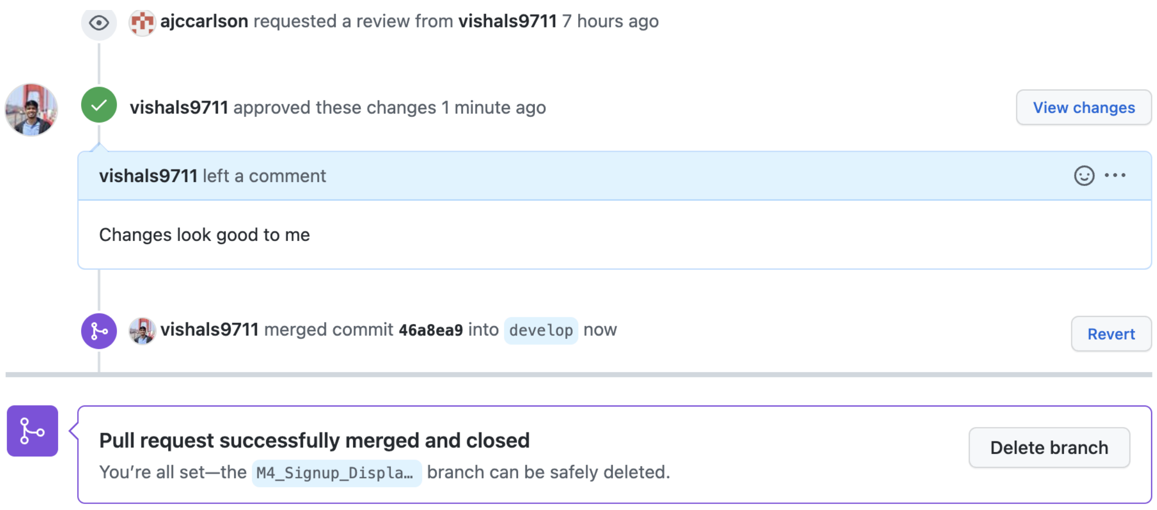
View

This branch has no conflicts with the base branch

Merging can be performed automatically.

Squash and merge

You can also open this in GitHub Desktop or view command line instructions.


ajccarlson requested a review from vishals9711 7 hours ago

vishals9711 approved these changes 1 minute ago

View changes

vishals9711 left a comment

Changes look good to me

vishals9711 merged commit **46a8ea9** into `develop` now

Revert

Pull request successfully merged and closed

You're all set—the `M4_Signup_Displa…` branch can be safely deleted.

Delete branch

5. Self-Check on Best Practices for Security

Asset to be Protected	Types of possible / expected Attacks	Your Strategy to mitigate / protect the Asset
Database	SQL Injection	All inputs have validation, queries are structured carefully prepared statements, and the database passwords are encrypted
Search	SQL Injection or XSS	The search bar input is limited to 40 alphanumeric characters
Password	Vulnerable account information	Database password encryption
Account	Robot, Scam, or Spam Users	User accounts are required to have an SFSU email to successfully create an account
Post	Inappropriate or dangerous posts, along with threats, or potential scammers	Administrators must approve and validate each new post within a 24 hour window

6. Self-Check of the Adherence to Original Non-Functional Specs

Non-Functional Specification	Status
Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0.	**ON TRACK**
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers.	**DONE**
All or selected application functions must render well on mobile devices.	**DONE**
Data shall be stored in the database on the team's deployment server.	**DONE**
No more than 50 concurrent users shall be accessing the application at any time.	**DONE**
Privacy of users shall be protected.	**DONE**
The language used shall be English (no localization needed).	**DONE**
Application shall be very easy to use and intuitive.	**DONE**
Application should follow established architecture patterns.	**DONE**
Application code and its repository shall be easy to inspect and maintain.	**DONE**
Google analytics shall be used.	**DONE**
No email 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**
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	**DONE**

Site security: basic best practices shall be applied (as covered in the class) for main data items.	**DONE**
Media formats shall be standard as used in the market today.	**DONE**
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.	**DONE**
The application UI (WWW and mobile) shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2022. For Demonstration Only" at the top of the WWW page nav bar. (Important so as to not confuse this with a real application).	**DONE**