SW Engineering CSC 648/848 Fall 2020
SFSU Trade Mart
Team 07

Team Members:

Team Lead:	Alicia Ramirez (Email: aramirez23@mail.sfsu.edu)
Front-End Lead:	Jonathan Pak
Back-End Lead:	Valeria Vallejo
Github Master:	Ricardo Carretero
Member:	Chandler Cruz
Member:	Chris Manaoat

Milestone 4
December 8, 2020

History Table

Date Submitted: December 8, 2020
Date Revised:

1. Product Summary:

Name of the product: SFSU TradeMart

All Major Committed Functions:
All Users (Unregistered/Registered):
1: All users shall be able to register accounts with the website.
2: All users who have not logged in shall be able to access the login/register page, and log in through any page of the website.
3: All users who are not logged in shall be suggested the option to log in or register upon making an offer or submitting a post for approval.
4: All users shall be able to view listings on the website.
5: All users shall be able to search for specific listings based on words in its title, description, or category.
6: All users shall be able to see descriptions of the items for sale on the search results and browsing pages of the website.
7: Users who are logged in shall be able to log out.
8: Registered users shall be able to create listings for their items and put them on sale through the website, thus known as posts. Postings contain the name of the seller, condition, and price.
9: Registered Users interested in specific listings shall be able to contact the seller to decide themselves the details of the transaction.
10: Registered Users shall be able to choose the time and meeting location from a set of predefined locations when contacting a seller.
11: Registered Users shall be able to access a dashboard either upon signing in or clicking the "Welcome [username]" on the navbar, where they can access three tabs, My Postings, Messages Sent, and Messages Received.
12: Admins shall be required to approve or disapprove listings according to the TOS using Workbench.
13: Admins shall be required to remove users for violating the TOS using Workbench.

URL: http://52.53.215.116/

2. Usability Test Plan:

Test Objective

The selected function we are testing for this usability plan is the search function. This test is to ensure the search function is easy and fast to use for users. The test objectives for this usability test plan is based on how the user can execute these actions:

- Search product listings by category
- Search product listings by title and description
- Sort product listings by price

  Through this testing we will be able to take feedback from users and use it to improve the ease of use for our search function.

Test Background and Setup

The test is to receive usability feedback on the search function for the SFSU TradeMart website. This includes searching by category, title, and description as well as sorting based on price. The web browsers it will be tested on are Google Chrome, Firefox, and Safari. The intended users of the site are SFSU students and faculty.

The test will begin on the website's homepage on Google Chrome (52.53.215.116). The user will be tasked with searching for certain items, as well as utilizing the sorting options. They will receive no assistance other than starting on the homepage and being given these tasks. Once the user has completed these tasks we will assess the results of the usability test through the criteria discussed in the Usability Task Description table and the responses of the Likert test.

The URL of the website is 52.53.215.116/ which is also the URL of the search page we are testing. The user can input the search term which can include the option of specifying a category from a dropdown menu. Once searched, they can then sort their results from a dropdown menu above the listings. The usability test is successful and completed if the user finds the appropriate result by searching the product listings available.

Usability Task Description

Task	**Description**
Task	Find a product listing
Machine State	Search query empty
Successful completion criteria	Search results properly loaded related to search term with chosen sorting option correctly applied

Benchmark	Completed in 1 minute

Effectiveness: Effectiveness would be measured by how many testers were able to properly complete the task in time and how many errors they made (clicks that were not necessary to complete the task).

Efficiency: Efficiency would be measured by the average time it took testers to complete the task, the average number of clicks it took to accomplish the task, and the average number of screens needed to accomplish the task.

Subjective Testing
("**X**" column that is your answer)

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I was able to find the search bar easily.					
I was able to find the desired listings easily.					
I was able to sort the search results to search listings easily.					

3. QA Test Plan:

Test Objectives

We are testing the search function and its sorting function to see if relevant search results are listed and sorted in the correct order.

HW and SW setup

Hardware: Windows 10 PC

Software setup: The setup will have the website's homepage (which is also the search page) loaded on two different browsers: Google Chrome and Firefox. We check to make sure the navbar at the top of the screen contains the search bar as well as the category dropdown list.

URL: http://52.53.215.116/

Feature to be tested

Search/Sorting Function

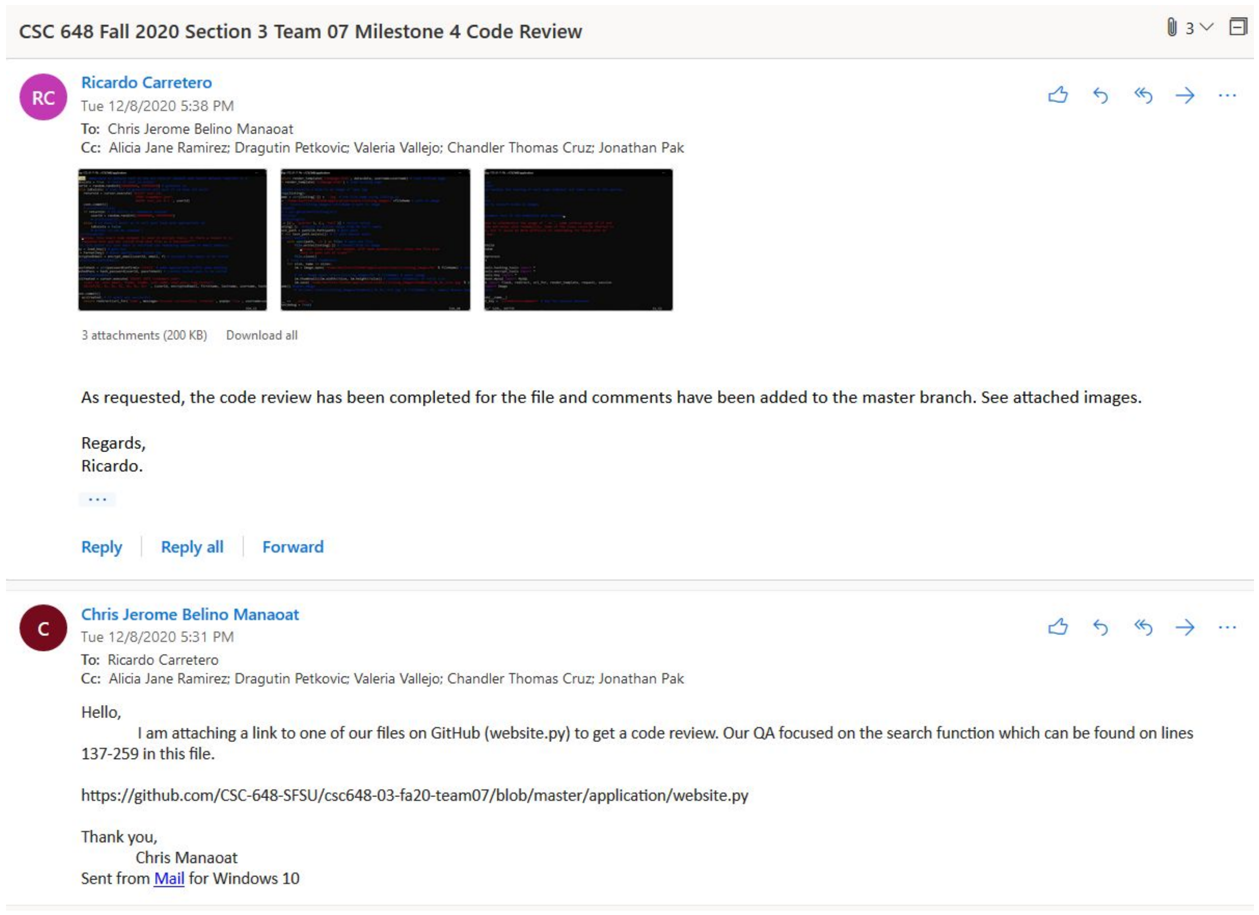
QA Test plan - table format

Number	Title	Description	Test Input	Expected Output	PASS/FAIL Chrome	PASS/FAIL Firefox
1	Test % like in search for name field	The search button should return the search results page with the correct results.	Type "iclicker" into the search bar with "All" as the category option and click the search button.	The search results page contains two listings: "take my iclicker" and "iclicker for any class".	**PASS**	**PASS**
2	Test category in search	The category search should return only listings that are of the selected category.	Select the "Tutoring" category with no search term and click the search button.	The search results page contains three listings: "Math Tutoring (1 hour)", "Art Mentor", and "Physics Tutoring (1 hour)".	**PASS**	**PASS**

3	Test search result sorting	The results should order themselves according to the chosen sorting option.	Select the “Books” category with no search term and click the search button. Click the “sorted by” button and select “Prices low to high”.	The search results contains four listings in this order: “cool math games textbook, math 100”, “engineering textbook, hardcover”, “calc 3 text book”, and “Math 100 Book for Sale”	**PASS**	**PASS**

4. Code Review:

Done over email as specified. Screenshots below:



CSC 648 Fall 2020 Section 3 Team 07 Milestone 4 Code Review

Ricardo Carretero
Tue 12/8/2020 5:38 PM
To: Chris Jerome Belino Manaoat
Cc: Alicia Jane Ramirez; Dragutin Petkovic; Valeria Vallejo; Chandler Thomas Cruz; Jonathan Pak

3 attachments (200 KB) Download all

As requested, the code review has been completed for the file and comments have been added to the master branch. See attached images.

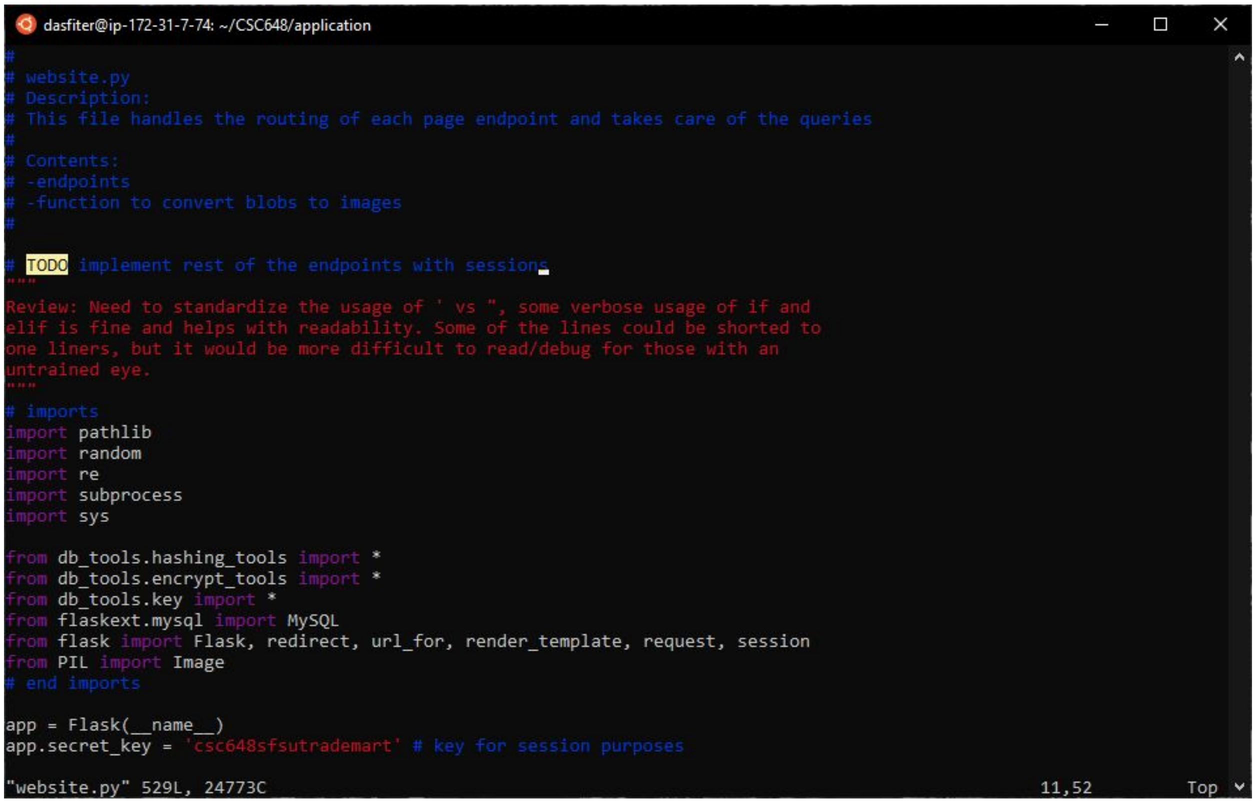
Regards,
Ricardo.

Chris Jerome Belino Manaoat
Tue 12/8/2020 5:31 PM
To: Ricardo Carretero
Cc: Alicia Jane Ramirez; Dragutin Petkovic; Valeria Vallejo; Chandler Thomas Cruz; Jonathan Pak

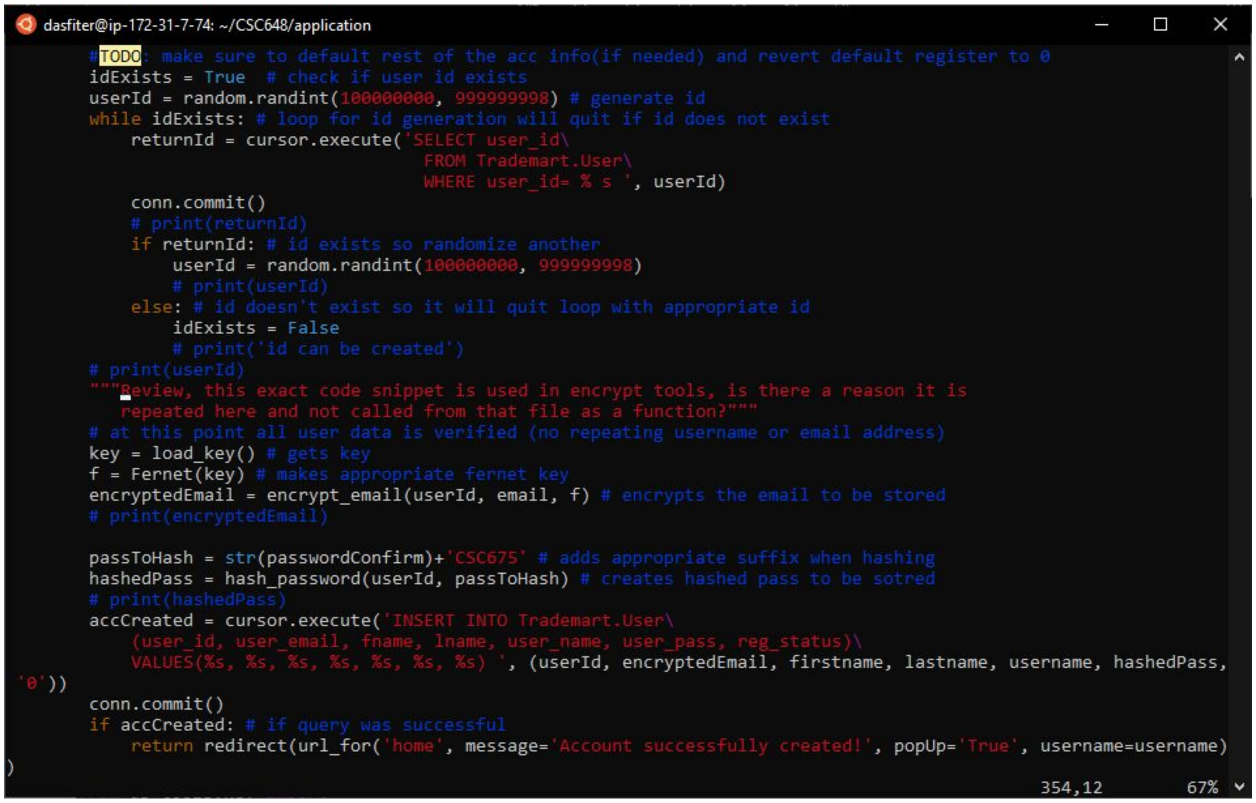
Hello,
I am attaching a link to one of our files on GitHub (website.py) to get a code review. Our QA focused on the search function which can be found on lines 137-259 in this file.

https://github.com/CSC-648-SFSU/csc648-03-fa20-team07/blob/master/application/website.py

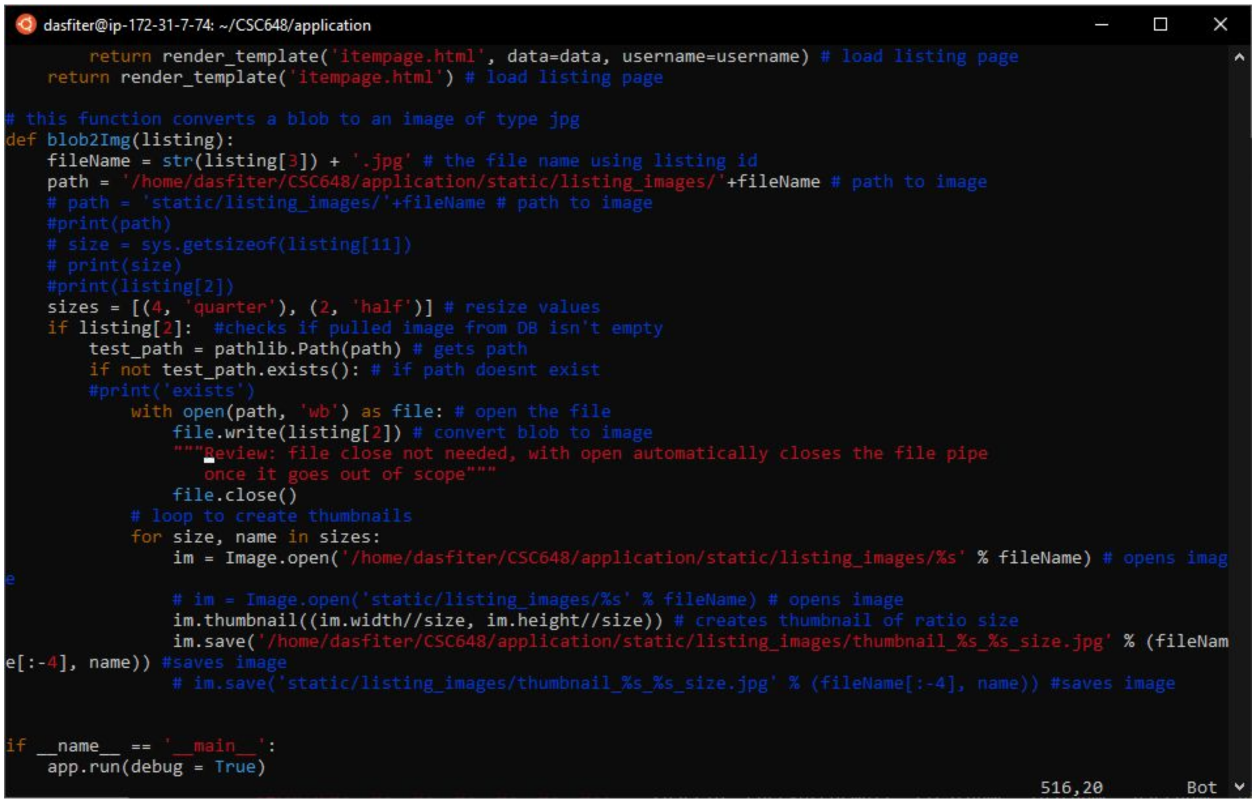
Thank you,
Chris Manaoat
Sent from Mail for Windows 10



```
dasfiter@ip-172-31-7-74: ~/CSC648/application
#
# website.py
# Description:
# This file handles the routing of each page endpoint and takes care of the queries
#
# Contents:
# -endpoints
# -function to convert blobs to images
#

# TODO implement rest of the endpoints with sessions
"""
Review: Need to standardize the usage of ' vs ", some verbose usage of if and
elif is fine and helps with readability. Some of the lines could be shorted to
one liners, but it would be more difficult to read/debug for those with an
untrained eye.
"""
# imports
import pathlib
import random
import re
import subprocess
import sys

from db_tools.hashing_tools import *
from db_tools.encrypt_tools import *
from db_tools.key import *
from flaskext.mysql import MySQL
from flask import Flask, redirect, url_for, render_template, request, session
from PIL import Image
# end imports

app = Flask(__name__)
app.secret_key = 'csc648sfsutrademart' # key for session purposes

"website.py" 529L, 24773C                                   11,52         Top
```



```
dasfiter@ip-172-31-7-74: ~/CSC648/application
        #TODO: make sure to default rest of the acc info(if needed) and revert default register to 0
        idExists = True  # check if user id exists
        userId = random.randint(100000000, 999999998) # generate id
        while idExists: # loop for id generation will quit if id does not exist
            returnId = cursor.execute('SELECT user_id\
                                        FROM Trademart.User\
                                        WHERE user_id= % s ', userId)
            conn.commit()
            # print(returnId)
            if returnId: # id exists so randomize another
                userId = random.randint(100000000, 999999998)
                # print(userId)
            else: # id doesn't exist so it will quit loop with appropriate id
                idExists = False
                # print('id can be created')
        # print(userId)
        """Review, this exact code snippet is used in encrypt tools, is there a reason it is
           repeated here and not called from that file as a function?"""
        # at this point all user data is verified (no repeating username or email address)
        key = load_key() # gets key
        f = Fernet(key) # makes appropriate fernet key
        encryptedEmail = encrypt_email(userId, email, f) # encrypts the email to be stored
        # print(encryptedEmail)

        passToHash = str(passwordConfirm)+'CSC675' # adds appropriate suffix when hashing
        hashedPass = hash_password(userId, passToHash) # creates hashed pass to be sotred
        # print(hashedPass)
        accCreated = cursor.execute('INSERT INTO Trademart.User\
            (user_id, user_email, fname, lname, user_name, user_pass, reg_status)\
            VALUES(%s, %s, %s, %s, %s, %s, %s) ', (userId, encryptedEmail, firstname, lastname, username, hashedPass,
 '0'))
        conn.commit()
        if accCreated: # if query was successful
            return redirect(url_for('home', message='Account successfully created!', popUp='True', username=username)
)
                                                            354,12        67%
```



dasfiter@ip-172-31-7-74: ~/CSC648/application

```
        return render_template('itempage.html', data=data, username=username) # load listing page
    return render_template('itempage.html') # load listing page

# this function converts a blob to an image of type jpg
def blob2Img(listing):
    fileName = str(listing[3]) + '.jpg' # the file name using listing id
    path = '/home/dasfiter/CSC648/application/static/listing_images/'+fileName # path to image
    # path = 'static/listing_images/'+fileName # path to image
    #print(path)
    # size = sys.getsizeof(listing[11])
    # print(size)
    #print(listing[2])
    sizes = [(4, 'quarter'), (2, 'half')] # resize values
    if listing[2]:  #checks if pulled image from DB isn't empty
        test_path = pathlib.Path(path) # gets path
        if not test_path.exists(): # if path doesnt exist
        #print('exists')
            with open(path, 'wb') as file: # open the file
                file.write(listing[2]) # convert blob to image
                """Review: file close not needed, with open automatically closes the file pipe
                   once it goes out of scope"""
                file.close()
            # loop to create thumbnails
            for size, name in sizes:
                im = Image.open('/home/dasfiter/CSC648/application/static/listing_images/%s' % fileName) # opens imag
e
                # im = Image.open('static/listing_images/%s' % fileName) # opens image
                im.thumbnail((im.width//size, im.height//size)) # creates thumbnail of ratio size
                im.save('/home/dasfiter/CSC648/application/static/listing_images/thumbnail_%s_%s_size.jpg' % (fileNam
e[:-4], name)) #saves image
                # im.save('static/listing_images/thumbnail_%s_%s_size.jpg' % (fileName[:-4], name)) #saves image


if __name__ == '__main__':
    app.run(debug = True)
                                                                                         516,20        Bot
```

5. Self-Check on Best Practices for Security:

Major assets being protected:

- User’s password
- User’s email
- User’s address

Major threats to assets:

- Admin getting access to private information from Users through MySQL Workbench.
- Access to our server via SSH from anywhere.
- Insertion of SQL queries in the search bar that could access our database.

How we’re protecting each asset:

- We’re protecting our users’ passwords by taking each user’s password, inserting the string ‘CSC675’ as a suffix to their password, and hashing that altered password before inserting it into the MySQL database. For the user’s emails, we hash them before inserting them into the database. This will prevent any site admins from having access to user sensitive information, even with direct access to the database. Admins are also given limited privileges with the database account they’re given, so they aren’t able to make drastic changes to our database.
- To prevent unwanted access via SSH to our server, we added inbound rules that only allow verified connections. This is in addition to the default security group through AWS.
- To prevent users from making SQL queries directly from the search bar, we implemented data validation directly into the search bar. We limited the user’s search bar input to 40 alphanumeric characters maximum. In addition to the search bar, we also implemented data validation into the register and sign-in pages to ensure users were inputting SFSU email addresses.

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 two major browsers - **ON TRACK**
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 - **ON TRACK**
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 - **ON TRACK**
9. Application should follow established architecture patterns - **ON TRACK**
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 - **ON TRACK**
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development - **ON TRACK**
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). - **ON TRACK**