SW Engineering CSC648/848 Fall2020
Gator Exchange

MileStone 4

Team 5
Team member

Naweeda Qeyam(nqeyam@mail.sfsu.edu)

Joshua Hansen

Dang Anh Tu Nguyen

Manuel Hernandez

Marvin Thai

Austin Bernard

Date Submitted:	**12/08/2020**
Date Revised (after instructor comments):	**12/09/2020**

Table of Contents

1) Product summary

Name of the product: Gator Exchange

Functions:

Unregistered Users:

1. Unregistered users shall be able to browse the products/services on the website.
2. Unregistered users shall be able to search and filter items.
3. Unregistered users shall be able to register.

Registered Users:

1. Registered users inherit all features of unregistered users.
2. Registered users shall be able to post items with all the data such as products/services with their username tied to the product, while being able to set the price of their product or service.
3. Registered users shall be allowed to remove their sales/services posts.
4. Registered users shall be able to message others users directly through the product post based on the item of interest.
5. Registered users shall be able to message back the exact buyers who initiated the message.
6. Registered users shall be able to view their post and messages.

Admin:(Manually using MySQL database)

1. Admin shall be required to approve all postings before they go live.
2. Admin shall be able to remove listings
3. Admin shall be able to remove users.

URL: http://34.209.214.191

2) Usability Test Plan

Test Objectives:

The objective of this is to test usability of the post function. What we will be looking for specifically is how effectively a user is able to navigate the website, how efficiently a user is able to post an item and how satisfied they are with their overall experience. The post feature is critical to the site's success so we believe that by making the user's experience during the process as simplified as we can, students will be more likely to return to our site to sell their products. It will be essential to identify any unnecessary hurdles users might encounter during their experience with our site so our success will depend on how easy the user's find creating a post was for them.

Test background and setup:

The test is to be conducted in a single room with only the user present. The intended user is a college student looking to sell an item on the site. It will be expected the users have some experience using an internet browser. The user will be provided instructions to follow on the site and they will be given an item with the appropriate information to make a successful post. The user will also be provided with a computer that will have the homepage of the site shown so the user will only have to follow the provided instructions. Once the user is settled in they will be told when to begin by an observer.

Once the user is told to begin the total amount of time will be recorded until all the tasks are completed. The observer remotely monitoring will record the user's navigation through the site. Every page the user spends time on will be recorded along with any information correctly or incorrectly inputted.

Usability Task Description:

TASK	DESCRIPTION	DIRECTIONS
Create a new post for your product.	The user should be able to navigate to the post page. When clicked the user will be asked to sign into their account or create one if needed. The user should use the information given about the product they are selling to fill in the description page.	Create a post
Sign in/Register	The user should be able to sign into their existing account or create a new account.	Create an account
View posting	The user should be able to find their posting on their account on the user dashboard.	View your post from user dashboard
Delete a posting	The user should be able to delete any post they recently made.	Remove the post you created from dashboard

Effectiveness:

To evaluate effectiveness we would document the percentage of users that were successful in creating a post. We would also track any errors the users might have made during the testing such as incorrect/incomplete descriptions.

Efficiency:

To evaluate efficiency we would record the total time it took each user to complete a task and the total number of screens a user viewed.

(Put X in box corresponding to rating given)

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
It was simple to create/upload a post					
I was able to login/register upon uploading a post					
My post was successfully uploaded and linked to my account properly					
Comments *(optional)*:					

3) QA Test Plan

Objectives

The objective of this plan is to test the functionality of a user's upload and posting functionality.

Registered users should be able to:

- Create a new post
    - Upload images for the new post
    - Create a description for the post
    - Categorize the post
    - Set a price for the post

Hardware and Software Setup

Users should use any reliable computer: Mac, Windows, or Linux, and any popular web browser: Google Chrome, Firefox, Safari, Opera, Microsoft Edge.

The user will start the test by following this link to the website:

http://34.209.214.191

Test Plan

Test Number	Description	Input	Expected Output	Pass/Fail
1	Test lazy login for posting	On the home page, click “Create Post” at the top left.	Login Page should prompt the user to login	
2	Test Login functionality	Enter “email” and “password” into the login prompt. Click “Submit”	If valid login credentials, should redirect user to home page. If invalid, redirect to login with empty form.	
3	Test “Create Post” proper redirection for logged-in user.	On the home page, click “Create Post” at the top left.	The user should be redirected to the create post page.	
4	Test Form Field Validation for Creating a post.	Enter “Textbook” in the “Title” field. Click “Submit”	The user should be prompted to fill out all fields.	
5	Test post submission validation	Enter: -”My book” in “Title” field -”This is a test description. 1234” in “Description field. -”1000.45” in “Price” field -Do not upload an image. Click “Submit”	The posting should be submitted to the admin for approval, the user should be redirected to the dashboard.	
6	Test User Dashboard update	No input - user should be on dashboard already	Verify that the new posting is listed on the user dashboard under “Postings”	

4) Code Review:

Review Notes:

Review 1 on routing code for the print statement.

Review 2 on html for logic on flask template code

Review 3 on making the 3 containers wrapped in a loop

Review 4 on removing unnecessary comments

General Comments:

Well documented and organized but lack of header info. clean spacing and uncluttered. Can make use of the flask Jinja template a bit more to avoid repetitive html

Comment 1

```
def createPost():
    if request.method == "POST":
        # Getting info from post and send to the database
        file = request.files['img']
        posting = {}
        data = request.form.to_dict()

        #############REVIEW 1: Comment out print statements after you use them for debugging so it does not clutter the terminal ################
        print(data)



        posting['title'] = data['title']
        posting['description'] = data['discription']
        posting['price'] = int(data['price'] )
        posting['category'] = data['filter']
```

Comment 2

```
 <select class="nav-link dropdown-toggle" name="filter"   data-toggle="dropdown">

            <!--------REVIEW 2: can add comments to make logic more clear in this statement, since navlst is not passed in from createPost route
            it is difficult to tell what data structure is being iterated over ---------->
```

```
            {%if navlst%}
            {% for n in navlst %}
            <option class="dropdown-item" value ="{{n[0]}}" {%if category == n[0] %}selected{%endif%} >{{n[0]}} </option>

            {% endfor %}
            {% endif %}
          </select>
```

Comment 3

```
<!-------- REVIEW 3 These three containers can be done using the Jinja Flask templating since id and names math label --------->
        <div class="container">
          <div class="form-group row justify-content-center">
           <label for="name" class="col-sm-2 col-form-label">Title</label>
            <div class="col-sm-4">
              <div class="form-check">
                <input type="text" class="form-control" id="title" name="title">

              </div>
            </div>
          </div>
        </div>



        <div class="container">
          <div class="form-group row justify-content-center">
            <label for="discription" class="col-sm-2 col-form-label">Description</label>
            <div class="col-sm-4 mx auto">
              <div class="form-check">
                <textarea class = "form-control mb-2"class="form-control" id="discription" name="discription" ></textarea>

              </div>
            </div>
          </div>
        </div>




        <div class="container">
          <div class="form-group row justify-content-center">
           <label for="price" class="col-sm-2 col-form-label">Price</label>
            <div class="col-sm-4 mx auto">
              <div class="form-check">
                <input type="text" class="form-control" id="price" name="price">

              </div>
            </div>
          </div>
```

```
        </div>
```

Comment 4

```
<!------ REVIEW 4 can remove this comment --------->

        <!-- <div class="container">
            <div class="form-group row justify-content-center">
            <lable for="courseId" class="col-sm-2 col-form-label">Course ID</lable>
                <div class="col-sm-4">
                    <div class="form-check">
                        <input type="text" class="form-control" id="courseId" ID="courseId">
                    </div>
                </div>
            </div>
        </div> -->
```

5) Self-check on best practices for security

Protecting-List major threats :

- SQL Database: Corrupting data can be sent to the database through web application. We need injection and input validation in the frontend and backend to prevent any possibility to trigger any malfunction in the application. For protecting our database on the server we have sanitized the data to make sure it's not a query when inserting to the database, and we have created a strong password in order to access it. For protecting images they are stored physically in the server and the route is stored in the database, and you need credentials to access either of it.
- User information : User information is top priority for security. These data need to be encrypted . For example, the password needs to be encrypted in the database.
- Account privilege: Only accounts with sfsu.edu can be able to register for an account. These registered accounts will allow users to use some features like message and posting . We will make sure in the front end that we have a check to only allow sfsu.edu to register an account.
- Search Input : Search input is only allowed up to 40 characters. This validation will be handled in the front end.

PW in the DB-Confirm Input data validation:

- PW in the database already been hashed and we used bcrypt to hash it before storing in the database

```
-           # encrypt password
-               password_byte = str.encode(password)
-               hashed = bcrypt.hashpw(password_byte, bcrypt.gensalt())
-               user['password']= hashed.decode("utf-8")
```

- Input data validation is in the application layer.

- Check if email is sfsu.edu in register is existed
- Check if password is secured by require password to have number letter, and be at least 7 character
- Check if username is valid by having no number in their name

```
        #checking whether email is a sfsu email or a email in database
        if "sfsu.edu" in email:
            user['email'] = email
        else:
            message = "Not SFSU Email"
        #checking whether the email has been rgistered already
        if db.user.getAUserbyEmail(email):
            message = "The email has been already registered"
        #checking if password contains at least 7 characters, 1 number, and 1 letter
        if len(password) < 7:
            message = "Password must have at least 7 characters"
        elif re.search('[0-9]',password) is None:
            message ="Password must include a number"
        elif re.search('[a-z]',password) is None:
            message ="Password must include a letter"
        else:
            user['password']=password
        #checking if name is only letters
        if fname.isalpha() & lname.isalpha():
            user['fname'] = fname
            user['lname'] = lname
        else:
            message = "First name and Last Name can only contain letters"
```

– Validate search bar input upto 40 character by setting maxlength to 40 :

```
            <input class="form-control mr-sm-2"  maxlength="40" type="search" style="width: 400px;"name="searchedData"
value="{{searchedData}}" aria-label="Search">
```

6) **Self-check:**

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. **DONE**
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. **DONE**
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. **DONE**
12. 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**
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. **DONE**
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"* **DONE**