

# SW Engineering CSC648/848 Spring 2022

## Gator Express

### Team 3

Amir Modan (Team Lead) - [amodan1@mail.sfsu.edu](mailto:amodan1@mail.sfsu.edu)

Dev Soni (Front-End Lead)

Kayvaun Khoshkhou

Vicente Pericone (Back-End Lead)

Brian Cheng

Majid Samir (Github Master)

### Milestone 4

05/16/2022

| Version # | Date Submitted | Changes                 |
|-----------|----------------|-------------------------|
| 1         | 05/16/2022     | First version submitted |

## 1. Product summary

### Gator Express

Life can be stressful for college students. This is especially true due to the challenges associated with being an independent adult who, oftentimes, must obtain necessary items such as books, supplies, and clothing. We aim to alleviate this burden by designing “Gator Express”, a web-based service on which college students, staff, and faculty can easily exchange items. Students will no longer have to pay high prices for many of their required items as most will already be sold by their predecessors on Gator Express at a fraction of the cost.

Our application allows users to:

- Search for items to buy.
- View all the categories of posts on the website and navigate to their preferred section.
- Sort search results.
- Click on any post and see its details, pricing, etc.
- View Gator Express’s about page.
- View the map of SFSU for reference.
- Register using name and email id (must be .edu).
- Login to create posts.

Registered users are allowed to:

- Post items to sell (Must be approved).
- View their own created posts.
- View messages received from prospective buyers.
- Send their contact information to a seller via one-way chat.

Admin are allowed to:

- Manage posts created by the users.
- View requests to add posts.
- Delete inappropriate posts.
- Approve posts before they go live.

We will ensure that all users can make informed decisions when selecting a meeting location by displaying a labeled map within our UI. Furthermore, the map will provide users with the locations of SFSU’s police department and “Emergency Blue Light Phones”, so that the users can make exchanges with maximum personal security incomparable to exchanges made on competing platforms.

<http://ec2-54-236-202-65.compute-1.amazonaws.com:3000/>

## 2. Usability Test Plan

### Test Objectives:

The objective of this test is to analyze the functionality of Creating an Ad post for the GatorExpress Marketplace application. The link is provided below to run the web application. This feature is being tested to make sure the user has a pleasant experience of using the web app and successfully create a post to sell the item they want and find a potential buyer for the item. It is also necessary to note the user has a smooth rundown and easy way to navigate through the application for achieving the expected results.

### Test Background and Setup:

Background : Will the user be able to create an Ad post to sell the item of his/her choice effectively and with minimum clicks. People can buy and sell items on campus with trust and safety and without having to reach out for external apps.

System Setup : The user can open the web browser of his/ her choice with internet connection from anywhere in the world. This application is specifically for members fo SFSU (Students & Staff). The user can then navigate the website for various options/ features. The test will take place using electronic observation room (Online Zoom Meetings)

Starting Point : URL of the application mentioned below is the starting point. Specifically the home page of the application is the first page the user sees once the website is loading.

Intended Users : Our application focuses on members of San Francisco State University that includes students and all the staff members. Anyone above age 17, with an authorized sfsu.edu email id can use GatorExpress to buy/ sell items. Our website is beneficial for safe exchange of items on campus using the provided campus map.

URL of the system : <http://ec2-54-236-202-65.compute-1.amazonaws.com:3000/>

### Subjective Feedback:

- Measure task satisfaction by creating a Likert questionnaire
- Ease of use
- Search result relevance
- How likely would you use the application in real life
- How streamlined is the buying & selling process

### Usability Task:

The task given to the user is to post an ad for used/ reusable items for a potential buyer on campus and perform safe exchange of items. Perform voluntary usability tests with people and keep the user's identity anonymous.

#### Steps:

1. Go to GatorExpress's Home Page (website url)
2. Explore some categories of items one can sell
3. Register and create an account for login to begin
4. Post an Ad for the item
5. Include category and images
6. Read the terms and disclaimer while posting your ad

#### Optional Steps:

7. Wait for a potential buyer to message you
8. Check the messages tab for any received messages
9. Visit the campus map & other posted items to buy

| TASK                           | DESCRIPTION   |
|--------------------------------|---|
| Task                           | Post an Ad for item                                   |
| Machine State                  | No items posted                                       |
| Successful completion criteria | Posted item is visible in marketplace within 24 hours |
| Benchmark                      | Completed in 3 minutes                                |

### Evaluation of Effectiveness:

We will use our real time website to show the testers the different pages of the application. We will start by using a demo account of GatorExpress to ask the tester to post an ad for a used item. We will also provide test images of used items. This process would be then judged for its effectiveness, efficiency, and feedback.

#### Measuring Effectiveness:

- If tester can complete the task of posting an ad
- Number of errors in each stage of the application

- Successful posting of ad and receiving messages from potential buyers

### Evaluation of Efficiency:

- Minimal amounts of clicks necessary to post an ad.
- Minimal effort required to complete tasks (low average time to complete)
- Minimal number of pages of the application to register, login and use the application to post an ad

| Test/Use Case                  | % Completed | Errors | Comments |
|--------------------------------|-------------|--------|----------|
| Search Engine for Posted Items |             |        |          |
| Register/ Login                |             |        |          |
| Post an Ad                     |             |        |          |
| Message a Seller               |             |        |          |

### Evaluation of User Satisfaction:

We will collect the feedback based on the above table and will evaluate it based on the Likert questionnaire below to measure efficiency.

Name: \_\_\_\_\_

Feedbacks:

It was easy to navigate through the application with few clicks.

☐ Strongly Agree   ☐ Agree   ☐ Neutral   ☐ Disagree   ☐ Strongly Disagree

The user experience makes me want to post used items and sell them on campus.

☐ Strongly Agree   ☐ Agree   ☐ Neutral   ☐ Disagree   ☐ Strongly Disagree

I will use this application multiple times in real life.

☐ Strongly Agree   ☐ Agree   ☐ Neutral   ☐ Disagree   ☐ Strongly Disagree

Very few clicks are required to post an item/ buy an item.

☐ Strongly Agree   ☐ Agree   ☐ Neutral   ☐ Disagree   ☐ Strongly Disagree

Any comments or suggestions on what we can do to provide you with a better service?

---

**Legal Issues:**

Tests are voluntary, anonymous and can be abandoned at any time.

**Report:**

The final report will contain efficiency results, errors in the application, suggestions from users, upcoming improvements in the application, and some ways to navigate through our application.

### 3. QA Test Plan

#### Test Objectives:

Once again, we will test the application's ability to post items. However this time, we will assess the *functionality* of posting rather than its *usability*. This means that the application must perform as per the functional requirements defined and must not produce any bugs. We will **not** consider the application's ease of use as this falls under Usability Testing rather than QA.

#### HW and SW setup:

While the QA process can be automated using testing software such as *Selenium*, we will test use cases manually for simplicity. The software we will be testing is located on our application's Development branch and once it has passed all tests, will be merged onto our main branch. The reason for this is to ensure that the main branch only contains code that is guaranteed to work, so that we will always have access to a working application. The application being tested can be found at

<http://ec2-54-236-202-65.compute-1.amazonaws.com:3000/>.

#### Feature to be tested:

We will test the **post** function which allows a registered user to post items to be sold. This means that the system should be able to save the item's information exactly as specified by the user. Although we will need to login to use this feature, we will not place much emphasis on the login use cases since that is a separate feature that would need to be tested on its own.

#### QA Test plan:


| Post Test: Google Chrome |                |  |                                     |                                 |  |
|--------------------------|----------------|--|-------------------------------------|---------------------------------|--|
| Test #                   | Title          | Description  | Input                               | Expected Output                 | Results<br>(Google Chrome /<br>Microsoft Edge) |
| 1                        | Post Item Data | Post complete item information including name, description, image, price, and category. User is NOT logged in initially. | Click Log In / Register button      | Navigates to ./Login            | PASS / PASS                                    |
|                          |                |  | Type "QA_User" in username field    | "QA_User" is displayed in field | PASS / PASS                                    |
|                          |                |  | Type "Password1!" in password field | "....." is displayed in field   | PASS / PASS                                    |

|   |                        |   |  |  |             |
|---|------------------------|---|--|--|-------------|
|   |                        |   | Click Log In button                        | Navigates to home page   | PASS / PASS |
|   |                        |   | Click Post button                          | Navigates to ./Post  | PASS / PASS |
|   |                        |   | Click on "Choose File"                     | Opens file directory   | PASS / PASS |
|   |                        |   | Upload "Item.png"                          | "Item.png" is displayed in field   | PASS / PASS |
|   |                        |   | Type "QA Test" in title field              | "QA Test" is displayed in field  | PASS / PASS |
|   |                        |   | Type "This is a test" in description field | "This is a test" is displayed in field   | PASS / PASS |
|   |                        |   | Type "\$5" in price field                  | "\$5" is displayed in field  | PASS / PASS |
|   |                        |   | Click Category Drop-down                   | Drop-down displays: Books, Class Equipment, Clothes, Kitchen, School Supplies, Sports, Technology              | PASS / PASS |
|   |                        |   | Select "Books"                             | "Books" is selected in drop-down   | PASS / PASS |
|   |                        |   | Click Post Item button                     | Navigate to Home Page  | PASS / PASS |
| 2 | Check Database         | Check all attributes for the target item match expected.      | Enter query for "QA_Test" item             | Title = "QA_Test"<br>Image = "Item.png"<br>Price = 5.0<br>Category = "Books"<br>Description = "This is a test" | PASS / PASS |
| 3 | Check Item Is Not Live | Check that the target item is not live before being approved. | Type "QA_Test" in search field             | "QA_Test" is displayed in field  | PASS / PASS |
|   |                        |   | Click Magnifying Glass Icon                | "QA_Test" item is NOT displayed among results  | FAIL / FAIL |



## 4. Code Review







For the code review section, our team discussed what we should do a code review on and we decided that I (Vicente) should review Brian's code for posting products. After discussing it during the meeting, I decided to do the code review on github because there is a feature where you can review commits on a specific branch which makes it easy to see what code is being changed. I go through the documents changed during the merge of feature/post into the main branch. After reviewing all the code, I wrote a summary explaining the important aspects that I noticed. I then send the email to Brian with the summary and a link to my review on github.



Vicente Yazmil Pericone

Tue 5/10/2022 6:22 PM

To: Brian Christopher Gorospe Cheng



Hello Brian,

I forgot to include the link to the review. The review can be found here <https://github.com/CSC-648-SFSU/csc648-03-sp22-team03/pull/14>. Thank you for your hard work.


Thanks,  
Vicente

...

Reply

 | 






Forward



Vicente Yazmil Pericone

Tue 5/10/2022 6:20 PM

To: Brian Christopher Gorospe Cheng



Hello Brian,

I reviewed the code we discussed that we would go over during our last meeting. This is an old merge from the feature/post branch. The structure of the code was formed well, and it was very easy to read from an outside perspective. The main thing that I would say to work on is using comments and headers to explain the logic of the files to someone who might not be a programmer. And I believe we added a lot of headers after this merge, but it is important to just keep it in mind. I thought all the names for the class/methods/variable were consistent and proper. One last thing to remember is to write the commit, even if it's a merge, to be descriptive so that others understand the gist of what's being changed. Otherwise, you are doing a great job, and keep up the good work!

Best Regards,  
Vicente Pericone



Brian Christopher Gorospe Cheng

To: Vicente Yazmil Pericone



Wed 5/11/2022 9:23 AM

Hello Vicente,

Thanks for the feedback and for doing the code review!

Best,  
Brian

...

No problem!

Always happy to help!

Happy to do it!

🗨 Are the suggestions above helpful? [Yes](#) [No](#)

↩ Reply

➦ Forward

ViP-Cente left a comment



Hello Brian. I reviewed the code we discussed that we would go over during our last meeting. This is an old merge from feature/post branch. The structure of the code was formed well in my opinion and it was very easy to read from an outside perspective. The main thing that I would say to work on is using comments and headers to explain the logic of the files to someone who might not be a programmer. And I believe we added a lot of headers after this merge but it is important to just keep it in mind. I thought all the names for the class/methods/variable were consistent and proper. One last thing to remember to label the commit, even if it's a merge, to be descriptive so that others understand the gist of what's being changed. Otherwise, you are doing a great job and keep up the good work!

application/server/routes/index.js

```
...    ...    @@ -33,20 +44,14 @@ router.get('/map', (req, res, next) => {  
33    44    res.render('map', {title: "Campus Map", onMap: true});  
34    45    })  
35    46
```



ViP-Cente 20 minutes ago



There can be comments for explaining how the myPage works with the options of either viewing the user's posts or messages.



Reply...

Resolve conversation

application/server/routes/users.js

```
...    ...    @@ -8,8 +8,128 @@  
8    8    *****/  
9    9
```



ViP-Cente 20 minutes ago



This file is pretty good. Since flash messages aren't currently working, we can comment it out or delete to be more concise. There can also be more comments to make the logic more clear. Missing header. Good variable names.



Reply...

application/server/models/Users.js

```
...    @@ -0,0 +1,61 @@  
1 + var db = require('../config/database');
```



ViP-Cente 21 minutes ago



Missing header. Needs more in-line comments. Good and consistent naming. Overall good structure



Reply...

Resolve conversation

application/server/middleware/validation.js

```
...    @@ -0,0 +1,58 @@  
1 + const checkUsername = (username) => {
```



ViP-Cente 21 minutes ago



Missing header. Needs more in-line comments. Good and consistent naming. Good structure but needs comments to explain it.



Reply...

Resolve conversation

application/server/middleware/myPageMiddleware.js

```
...    @@ -5,4 +5,31 @@  
5      5      *
```



ViP-Cente 21 minutes ago



Missing header. Needs more in-line comments. Good and consistent naming. The sql query can be refractured to a model file

application/server/helpers/debug/debugprinters.js

... @@ -0,0 +1,23 @@

1 + const colors = require('colors');



ViP-Cente 22 minutes ago



Missing header. Good and consistent naming. Good class structure



Reply...

Resolve conversation

application/server/middleware/myPageMiddleware.js

```
11 +  
12 + myPageMiddleware.loadMyPosts = (req, res, next) => {  
13 +   let userId = 1;
```



ViP-Cente 22 minutes ago



userId is a hardcoded value and needs a commented explaining that we need to change it later



Reply...

Resolve conversation

application/public/js/frontendjs.js

... @@ -11,4 +11,35 @@

```
11 11   function submitExternal() {  
12 12       let formElement = document.getElementById('search-bar');  
13 13       formElement.submit();  
14 + }  
15 +
```



ViP-Cente 22 minutes ago



This feature is currently not being used and could have a comment explaining so or commented out.

## 5. Self-check on best practices for security

| <u>Asset to be protected</u> | <u>Types of possible/expected attacks</u>                                  | <u>Your strategy to mitigate/protect the asset</u>   |
|------------------------------|--|--|
| The user database            | 1. Unauthorized user gains access to confidential data<br>2. SQL injection | 1. Require users to authenticate themselves.<br>2. Only a select few people can manage the database directly                 |
| User Passwords               | 1. Unauthorized user gains access to confidential data<br>2. SQL injection | 1. Passwords are encrypted in the database.<br>2. Passwords are validated by the front and backend                           |
| User Email                   | 1. Unauthorized user gains access to confidential data<br>2. SQL injection | 1. Emails are validated by the front and backend<br>2. Emails are required to have sfsu.edu at the end                       |
| Search Input                 | 1. Unauthorized user gains access to confidential data<br>2. SQL injection | 1. Search input is limited to 40 alphanumeric characters<br>2. Search is validated on the backend before executing the query |
|                              |  |  |

## **6. Self-check of the 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. **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. **DONE**
6. Privacy of users shall be protected. **ON TRACK**
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. **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. **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, 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**