**SW Engineering CSC648/848**
**Section 01**
**Fall 2018**

# GatorTrade

**Team 04 | local**
Abigail Chin | chinabigail08@gmail.com
Alyssa Malunao
Dhruv Shah
Jarek Rettinghouse
Jed Ahmadia
Nikita Bajracharya
Robert Quiñones

**Milestone 4**

| Submission Date | History |
|---|---|
| **December 2, 2018** | First draft |
| | |

## 1. Product Summary

We are coming to market with our product GatorTrade. It is a buy and sell website designed for SFSU students for selling, listing, and buying items. Below are the features our users can look forward to on our platform.

- A robust search engine allowing for quick searches based on any part of a product. Whether it be household items, electronics, books, events, services, apparels .
- Advanced filters that allow the user to quickly refine search based off price, locations, and categories.
- Easy signup setup for a user who wants post their products or post their services, he/she will be able also access the user dashboard, which contains all of their posts and messages from interested buyers.
- For safety and convenience reasons designated safe place locations are provided on SFSU campus to exchange goods for money.
- Clear and concise layout of detailed information on the product description, price and locations.

What we think is unique about out product is its ease of use, through good and clean design. The website is free of clutter from niche tools and advertisements, allowing users to cut straight to finding products and services. We believe this fosters a more efficient buy and sell website for SFSU Students.

Website: https://gator-trade.herokuapp.com/

## 2. Usability Test Plan

### Test Objectives:

The objective of our usability test plan will be to identify specific user interactions with the website that could be improved for the sake of general usability. Since our website is in it's beta, it's important to find out the weaknesses in its usage flow now rather than when it is launched.

We will be conducting a test that will send a beta user through a series of usage flows touching on the main components of the website. These components will be usage of search, filters, and posting a item or service on the website. Success factors will be based on the likert scale, test users will be asked how they **felt** about the process(s). From this we will be able to judge the general usability of the website, and more importantly on the key areas of weaknesses within the website.

### System Setup:

A laptop with internet capabilities and Google chrome, Mozilla Firefox, Safari, or another internet browser.

### Test Plan:

- **Purpose:** The purpose of this plan is to identify key areas of weaknesses within our website when it comes to general usability.

- **System Setup:** There will be two versions of this test. Desktop browser and Mobile browser. For the sake of consistency we will be looking to use Chrome Browser (latest two versions only) on the desktop, and Safari browser (latest two versions only) for mobile. This allows us to test with the most popular configurations within the college students.

- **Starting Point:** The starting point will be the homepage of the website.
    - Located at: https://gator-trade.herokuapp.com/

- **Tasks Description:** There are three important tasks in order to successfully test out the search function.

| **Task** | **Description** |
|---|---|
| Task | Default Category Selected |
| Machine State | Category Selected, Search Query Typed In |
| Successful Completion Criteria | “Events” displayed on “Category” Tab |
| Benchmark | Completed in 5 Seconds |

| **Task** | **Description** |
|---|---|
| Task | Find Coffee Pot |
| Machine State | Empty Search bar |
| Successful Completion Criteria | “Coffee Pot” typed out in Search Bar |
| Benchmark | Completed in 5 Seconds |

| **Task** | **Description** |
|---|---|
| Task | Search for “Books” in “Background” Category |
| Machine State | Category Selected, Search Query Typed In |
| Successful Completion Criteria | Page Displaying all items with “Books” in the name |
| Benchmark | Completed in 5 Seconds |

**Completion Criteria:** User has successfully done the following:

- Found all posts in “Events” Category
- Found all posts related to “Coffee Pot”
- Has gotten a search result based off the users choices. Example: Books.

**User Satisfaction Questionnaires:**

- The GUI was simple to understand (select one):
    - __Strongly Disagree

    - __Disagree
    - __Neither agree nor disagree
    - __Agree
    - __Strongly Agree
    - **Comments:** _____

- It was easy to search for items posted (select one):
    - __Strongly Disagree
    - __Disagree
    - __Neither agree nor disagree
    - __Agree
    - __Strongly Agree
    - **Comments:** _____

- The results displayed were appropriate to the search query you entered (select one):
    - __Strongly Disagree
    - __Disagree
    - __Neither agree nor disagree
    - __Agree
    - __Strongly Agree
    - **Comments:** _____

### 3. QA Test Plan

| Test Number | User Type | Test Setup Description | Test Input | Expected Results | Test Status |
|---|---|---|---|---|---|
| **1** | **Non-Registered User** | **Setup:** Open up the following link in the browser: **http://gator-trade.herokuapp.com/**<br><br>**Description:** User shall be able to search for items in search bar | **Search Bar:** Input: "Books" | Page showing all books for sale. | Pass |
| **2** | | **Setup:** Open up the following link in the browser: **http://gator-trade.herokuapp.com/**<br><br>**Description:** User shall be able to click on categories from home page | **Categories:** Clicked on: "Apparel" | Takes you to page showing all different apparel being sold. | Pass |
| **3** | | **Setup:** Open up the following link in the browser: **http://gator-trade.heroku** | **Recent Posts:** Clicked on: "Economics Tutor" | Takes you to the item's detail page where you are able to message the seller. | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | app.com/<br><br>**Description:** User shall be able to see and click on the four most recent posts | | | |
| **4** | | **Setup:** Open up the following link in the browser: http://gator-trade.herokuapp.com/<br><br>Then click on item taking you to item's detail page<br><br>**Description:** User shall be redirected to sign up page when trying to send a message | **Item's Detail Page:** Message: "I am interested in your item". User: Clicks on "Send Message" button. | Page is redirected to SignUp page | Pass |
| **5** | | **Setup:** Open up the following link in the browser: http://gator-trade.herokuapp.com/<br><br>Then click the sign up button on the top right | **Sign Up Page:** User fills out all information required and clicks on "Create Account" button | Successfully Registered | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | **Description:** User shall be able to register | | | |
| **6** | **Registered User** | **Setup**: Open the following link in the browser: **http://gator-trade.herokuapp.com/**<br><br>Then click the login button on the navbar<br><br>**Description:** Users shall be able to login into the system | **Login:** User requires SFSU email and password and clicks on "SIGN In" button | Successfully Logged In | Fail |
| **7** | | **Setup**: Open the following link in the browser: **http://gator-trade.herokuapp.com/**<br><br>**Description:** Users shall be able to message the seller of the interested item. | **Item's Detail Page:** Message: "I am interested in your item". User: Clicks on "Send Message" button. | Message sent | Fail |
| **8** | **Registered User(Seller)** | **Setup**: Open the following link in the browser: | **Post:** User fills out all the detailed | Item posted | Fail |

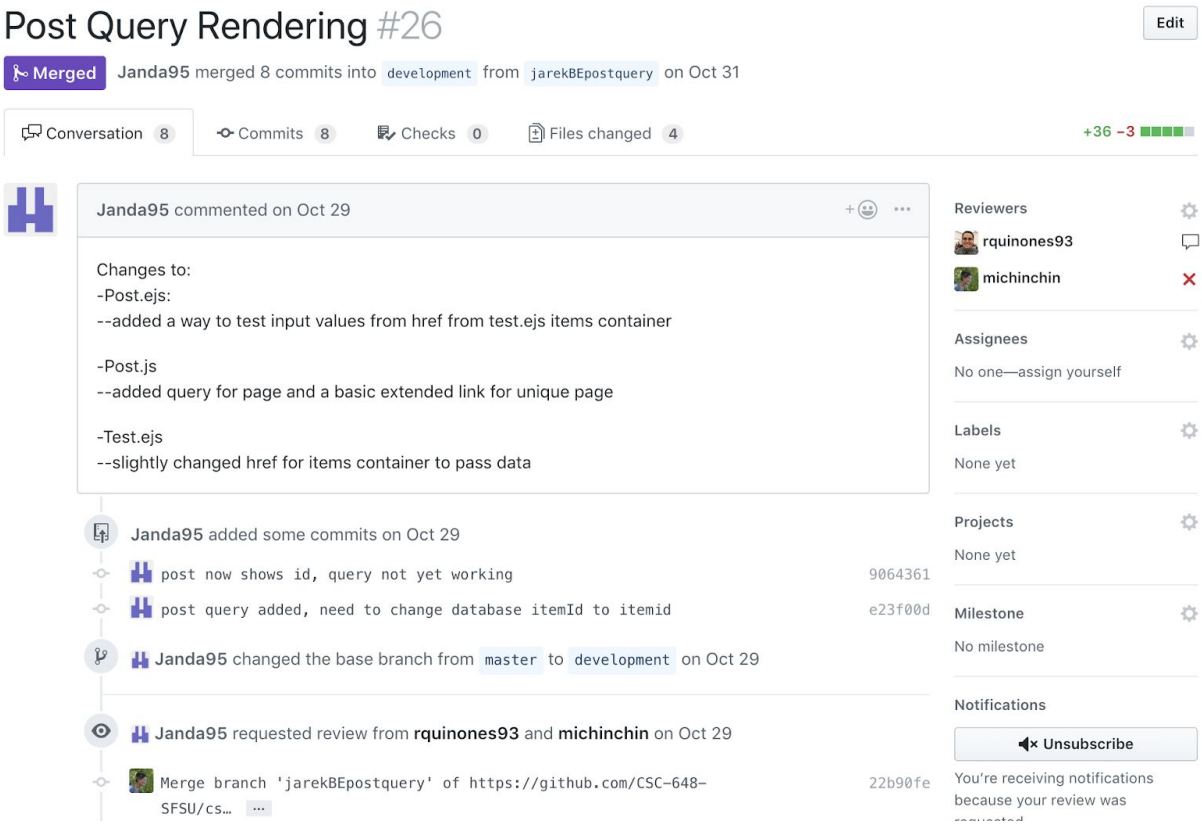
| | | | | | |
|---|---|---|---|---|---|
| | | **http://gator-trade.herokuapp.com/**<br><br>Then click the post button on the top left corner of navbar<br><br>**Description:** User shall be able to post items for sale. | information of the item/service and clicks on "Post Item For Sale" button | | |

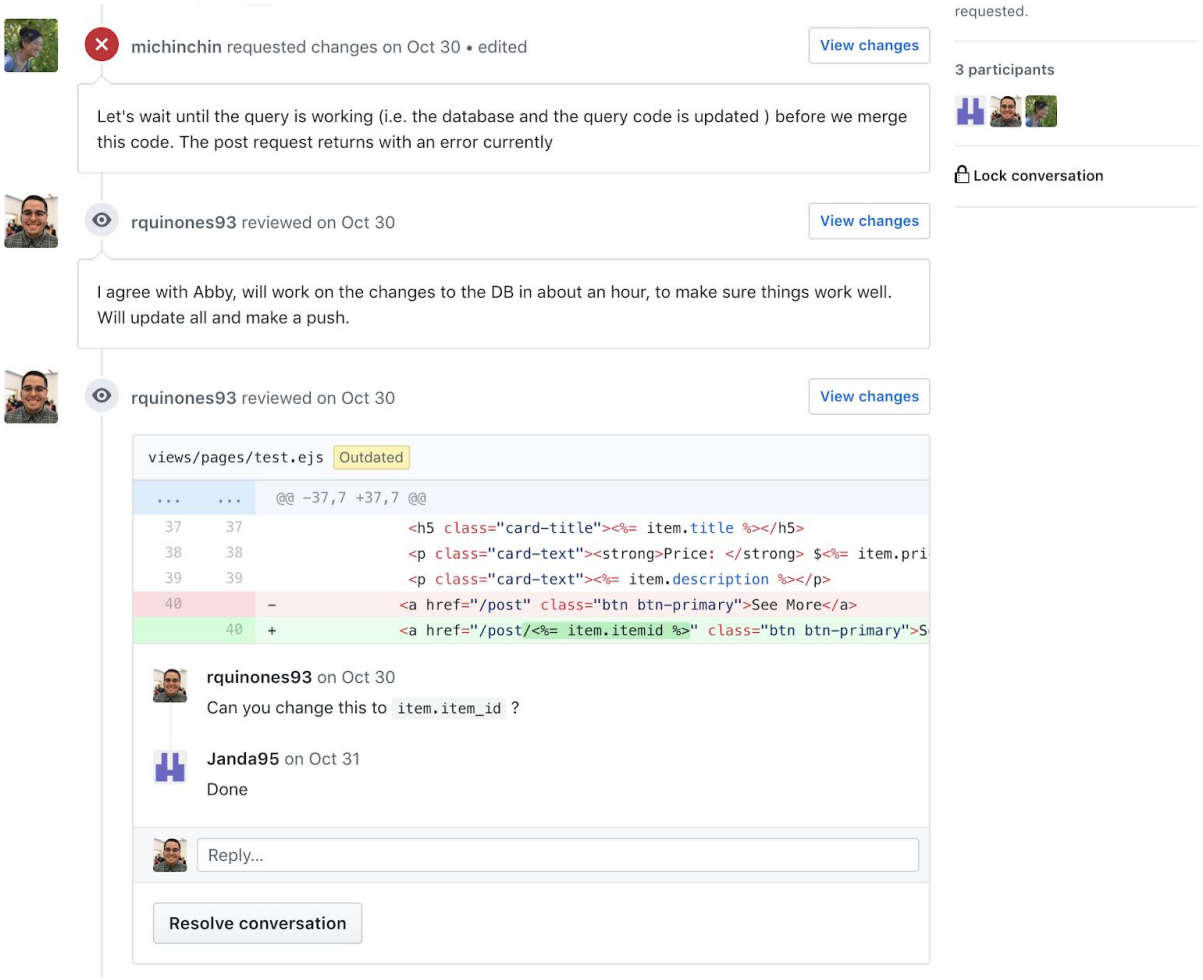
## 4. Code Review

### a) Coding Style

- Variables are defined in snake_case, defined with `let`
- Constants are defined in UPPER_CASE_SNAKE_CASE, defined with `const`
- All variables and constants are given relevant names to the data they contain
- New code blocks start { braces on the same line as the conditional statement / function declaration, and end } brace after the last line of relevant code
- ES6 Arrow Functions ( => ) are used for all Back End JS functions
- Comments are encouraged, and are mainly on HTML code, but Back End code is written with relevant function and variable names, so that code is self documenting
- **Code Organization**
    - **/auth** - Code dealing with user login & authentication, proper authentication on functions needing a user to be logged in or an admin.
    - **/database** - Code dealing with queries to the Database. Functions are exported, where the Back End team can import exactly which functions they need and nothing more.
    - **/migrations** - Database migration files that properly set up Database tables.
    - **/models** - Database model files that properly set up relations between tables.
    - **/routes** - Node.js code that determines which page is rendered and what DB queries are made. Business logic.
    - **/seeders** - Database files that seed Database with dummy data.
    - **/validation** - Custom Back End form validation files.
    - **/views** - Front End Code
    - **/views/partials** - Reused pieces of code that exist on each page

**b) Code Peer Review - Code written by Jarek, Reviewed by Abby & Robert**




requested.
3 participants
Lock conversation
michinchin requested changes on Oct 30 • edited
View changes
Let's wait until the query is working (i.e. the database and the query code is updated ) before we merge this code. The post request returns with an error currently
rquinones93 reviewed on Oct 30
View changes
I agree with Abby, will work on the changes to the DB in about an hour, to make sure things work well. Will update all and make a push.
rquinones93 reviewed on Oct 30
View changes
views/pages/test.ejs
Outdated
@@ -37,7 +37,7 @@
<h5 class="card-title"><%= item.title %></h5>
Price:  $<%= item.pri
<%= item.description %>
- <a href="/post" class="btn btn-primary">See More</a>
+ <a href="/post/<%= item.itemid %>" class="btn btn-primary">S
rquinones93 on Oct 30
Can you change this to item.item_id ?
Janda95 on Oct 31
Done
Reply...
Resolve conversation

views/pages/post.ejs Outdated

```
...  ...  @@ -1,6 +1,6 @@
1    1     <div class="jumbotron">
2    2       <div class="container">
3    3         <h1 class="display-3">Under Construction!</h1>
4         -     <p>Placeholder for our Post Page</p>
     4     +     <p> Placeholder <%= postID%></p>
```

**rquinones93** on Oct 30

Can you change to `post_id` to stick with naming convention

**Janda95** on Oct 31

Done

Reply...

Resolve conversation

routes/post.js Outdated

```
24  +            title: "Post " + localpostId,
25  +            items: items,
26  +            currentCategory: currentCategory,
27  +            postID: localpostId
```

**rquinones93** on Oct 30

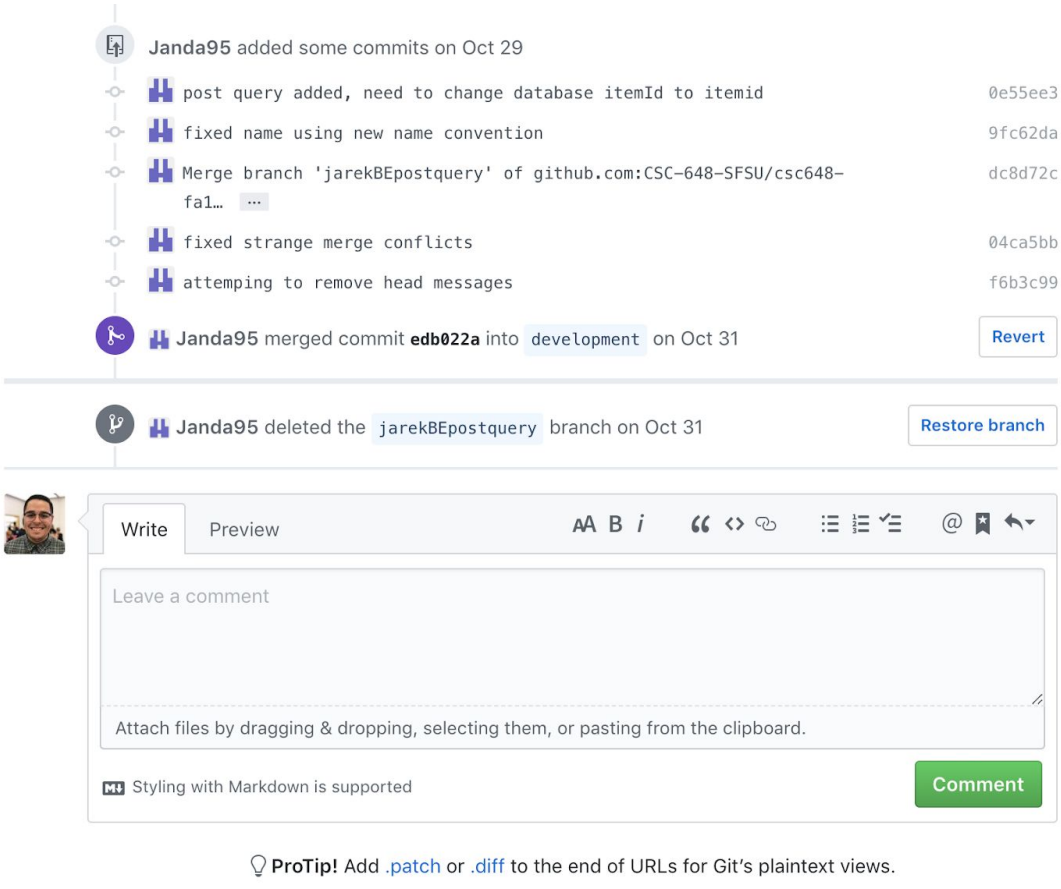
Can you change post_id, I will update to current_category

**Janda95** on Oct 31

Done

Reply...

Resolve conversation


Janda95 added some commits on Oct 29
post query added, need to change database itemId to itemid
0e55ee3
fixed name using new name convention
9fc62da
Merge branch 'jarekBEpostquery' of github.com:CSC-648-SFSU/csc648-fa1…
dc8d72c
fixed strange merge conflicts
04ca5bb
attemping to remove head messages
f6b3c99
Janda95 merged commit edb022a into development on Oct 31
Revert
Janda95 deleted the jarekBEpostquery branch on Oct 31
Restore branch
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Comment
ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.

**5. Self-Check on Best Practices for Security**

- Passwords are encrypted and saved using the BCrypt Node module.

  - User profile pictures & item pictures are uploaded to Cloudinary and are given a hard to assume, public URL in order to view on the site. The URL does not allow any users to access the Cloudinary storage website, just view the image. User assets are secured.

  - SQL code injection prevented through use of literal executions that are validated on pages when passed.

- Organization techniques for readability and functionality implemented where possible when excess code or repeated code deemed necessary.
  - For Example: Database connection and Queries organized into groups and indexed to give clear description of their purpose and displaces code away from main JavaScript routing files. Easy to see what is happening and where to look for database queries.

- Routes (JS), EJS, Database Connection, Configuration, and Dummy Database Variable files are separated and used for organizational depth and minimizing code needed to be displayed to user.

- Passport for personal user access has been implemented so users only can see information regarding them and not another users. Makes use of user sessions and stores them in a way that users are unable to change values to access other user's accounts

- Passport authentication methods also implemented on various functions that a user cannot complete until logged in: Creating a new post, Messaging a Seller, Accessing Admin Console (Logged in & Admin), Accessing User Dash. If user is not logged in, they are redirected to the Login Page and an error message is Flashed.

- Front End form validation for login and user creation forms, Back End Validation in progress.

## 6. Self-Check: Adherence to Original Non-Functional Specs

***High-level non-functional specifications***

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 all major browsers: Mozilla, Safari, Chrome. DONE
3. Selected application functions must render well on mobile devices ON TRACK
4. Data shall be stored in the team's chosen database technology 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. DONE
8. Application shall be very easy to use and intuitive. ON TRACK
9. Google analytics shall be added ON TRACK
10. No e-mail clients shall be allowed DONE
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. DONE
12. Site security: basic best practices shall be applied (as covered in the class) DONE
13. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development DONE
14. 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 of the WWW page. (Important so as to not confuse this with a real application). ON TRACK