SW Engineering CSC648/848 Fall 2019

“Gator Store”

Section 02

Milestone 4

Team 12

Beibei Jiang - Team Lead / Github Master / Back-end Developer
Email: bjiang1@mail.sfsu.edu
Jinying Ren - Front-end Lead
Jesse Smick - Back-end Lead
Angelo Solitario - Front-end Developer
Melissa Endang - Front-end Developer
Jinghan Cao - Back-end Developer

Submitted Date	12/12/2019
Revised and Frozen Date	12/17/2019

1. Product summary

Our product named “Gator Store” is a website that allows SFSU students to buy and sell items such as books, electronic products, furniture, and more. For those who have an economic burden and need to sell items in a short time, our platform is the best choice. What makes our product unique is in its ability to provide SFSU students with easy, fast, economic experience from one seamless shopping website.

All major committed functions:

Unregistered Users:

1. Users shall be able to create an account
2. Users shall be able to view listed items for sale
3. Users shall be able to browse through category
4. Users shall be able to see the remaining amount of items
5. Users shall be able to filter items by price

Registered Users:

7. +Unregistered users’ functions
8. Users shall be able to log into the website
9. Users shall be able to post items attached with prices for sale on this website
10. Users’ posted items shall be reviewed by an administrator before it is officially posted on website
11. Users shall be able to have their own dashboard

    a. View the in-site messages

    b. View past posted items

    c. Delete past posted items

12. Users shall be able to send in-site messages among each other
13. Users shall be able to determine a pick up spot on SFSU campus

Admins:

14. Administrators shall be able to log in the website

15. Administrators shall be able to have administrator dashboard
16. Administrators shall be able to review the items before they are officially posted
17. Administrators shall be required to approve items before they are officially posted
18. Administrators shall be able to delete items

URL: https://gatorstore.duckdns.org/

2. Usability test plan

Test Objective: This test is designed to test the effectiveness, subjectivity, and efficiency of the search functionality. We decided to test the search functionality since it is a buy and sell website. Users are most likely going to be looking for specific items therefore the search functionality is very important. The intended goal of our website is to be able to provide users the efficiency and effectiveness of searching items.

Test background and setup:

System Setup: Using the latest 3 updates of the browsers listed below

Chrome

Firefox

Safari

Intended Testers:

Any SFSU student or faculty member.

Usability Task Description

Tasks:

Starting point for the test: http://gatorstore.duckdns.org

Each user will be asked to find an item that they are interested in.

Each user will be asked to find a book.

Each user will be asked to find items by category

Each user will be asked to browse all the items

Each user will be asked to complete a satisfactory survey

Effectiveness: The effectiveness of the functionality is based on how many users are able to complete the task without any errors within the given time. For this test, we will be looking for if the user can complete the task in a minute. For example, if a user types in "book", the results should show all the books that are available for sale. When using categories, the user should be able to see items related to that category plus any keywords that a user enters in. In simple terms, we will measure the percentage of successful tasks completed.

Tasks	Success / Fail %	Errors
Task1		
Task4		

Success/Fail refers to whether a user is able to find the item

Tasks refer to the tasks listed above

Efficiency: To measure the efficiency of the function, we will be watching closely on how fast they can find the item. Therefore, we will be watching the number of clicks and the amount of time they spent looking for the item.

Tasks	Number of Clicks / Pages	Amount of time task took
Task1		
Task4		

Tasks refer to the tasks listed above

To ensure consistency, we plan that all users will be completing the same tasks as outlined above.

(Lickert) Satisfactory survey (circle one).

1. The pull down category was easy to use to find items in that category.

Strongly Agree Agree Neither agree or disagree Disagree Strongly Disagree

2. It was easy to find the book that I was looking for.

Strongly Agree Agree Neither agree or disagree Disagree Strongly Disagree

3. I am satisfied with the search function.

Strongly Agree Agree Neither agree or disagree Disagree Strongly Disagree

4. Comments (if any):

3. QA test plan - max 2 pages

Test Objectives: Test the search feature for correctness under multiple scenarios. Ensure it adheres to the specification.

HW and SW setup: Server running on Google Cloud. Testing done with Firefox 71 and Chrome 78 on Fedora Linux. Reset database to a consistent and known set of posts. Website URL is https://gatorstore.duckdns.org/.

Feature to be tested: Searching for posts.

QA Test plan:

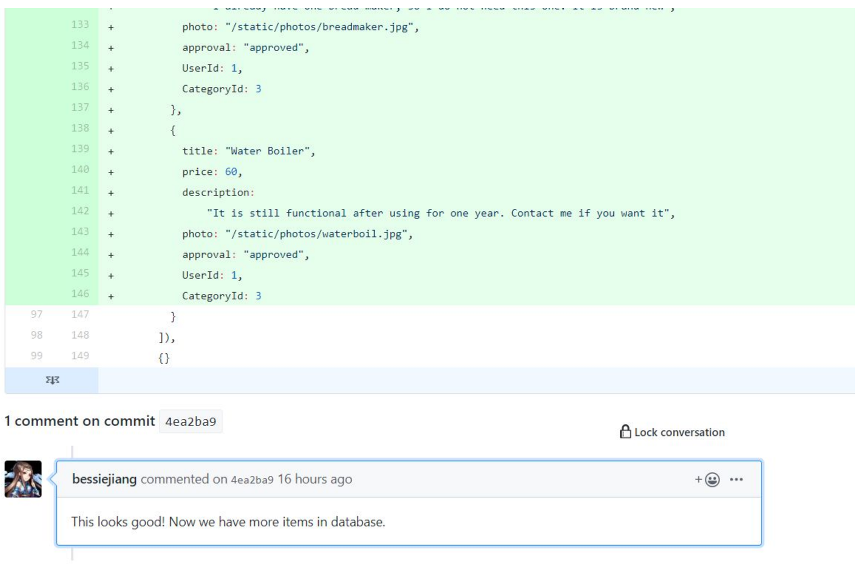
#	Title	Description	Input	Expected Output	Results
1	Search by Category	Perform search with empty search text and the "Furniture" Category selected	"Furniture" Category. Empty search text	Output contains one item titled "IKEA Sofa"	Firefox: ✓ Chrome: ✓
2	Search text ignores non alpha-numeric	Perform test with only alpha-numeric and then add non alphanumeric to make sure they are the same	"All" Category. First search text: "textbook" Second search text: "~>textbook@"	Both outputs contain one item titled "MATH 324 textbook".	Firefox: ✓ Chrome: ✓

3	Search text validation	Ensure search text is limited to 40 characters	“All” Category. Search text: 1. 39 characters 2. 40 characters 3. 41 characters	Search text contains: 1. 39 characters 2. 40 characters 3. 40 (NOT 41) characters	Firefox: ✓ Chrome: ✓
4	Search for item	Perform search for specific items	“All” Category. Search text “longboard”	Output contains one result titled “Sector 9 longboard”	Firefox: ✓ Chrome: ✓

4. Code Review

1. **Coding Style:**
   We focus on two major aspects in terms of the coding style. Firstly, code indentions should be controlled. For example, we applied the default indentation provided by the IDE (Visual Code Studio). When individual contributor pulls request to the official development branch, one or two other team members will check the codes to ensure no unnecessary empty lines and redundant comments. Secondly, we utilize the third-party npm tool Prettier to control the code formats of all the .js files. This tool helps us to break long one-line function down to several lines to make the code easier to maintain. To be more specific, we have the following rules to manage the uniformity of every member's coding style:
   - Ensure the indention is two-space long
   - Functions with multiple input parameters should break down to multiple lines
   - Between two functions, there should be an empty line to seperate
   - Object as input should break into multiple lines.
2. **Code Review:**
   All the committed code should pass code review before they are officially merged into the main repo. Before everyone starts to work, team leader assigns part of the project to each member through github so that everyone knows their own job. Each member is encouraged to set up their own branch and develop on the local. Once the codes are implemented, pull requests will be initiated to the official development branch. Every pull request should provide detailed description about new updates and functions added to the project. Then other team members will review and provide suggestions to ensure the code quality.

In-code review comments



```
+             "I already have one bread maker, so I do not need this one. It is brand new",
+          photo: "/static/photos/breadmaker.jpg",
+          approval: "approved",
+          UserId: 1,
+          CategoryId: 3
+        },
+        {
+          title: "Water Boiler",
+          price: 60,
+          description:
+              "It is still functional after using for one year. Contact me if you want it",
+          photo: "/static/photos/waterboil.jpg",
+          approval: "approved",
+          UserId: 1,
+          CategoryId: 3
         }
       ]),
       {}
```

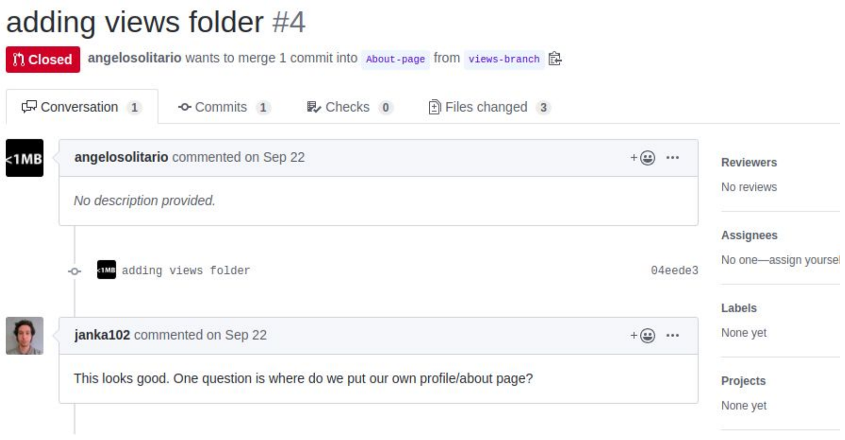
1 comment on commit 4ea2ba9

Lock conversation

bessiejiang commented on 4ea2ba9 16 hours ago

This looks good! Now we have more items in database.

Advice on project structure



adding views folder #4

Closed **angelosolitario** wants to merge 1 commit into `About-page` from `views-branch`

Conversation 1 | Commits 1 | Checks 0 | Files changed 3

angelosolitario commented on Sep 22

No description provided.

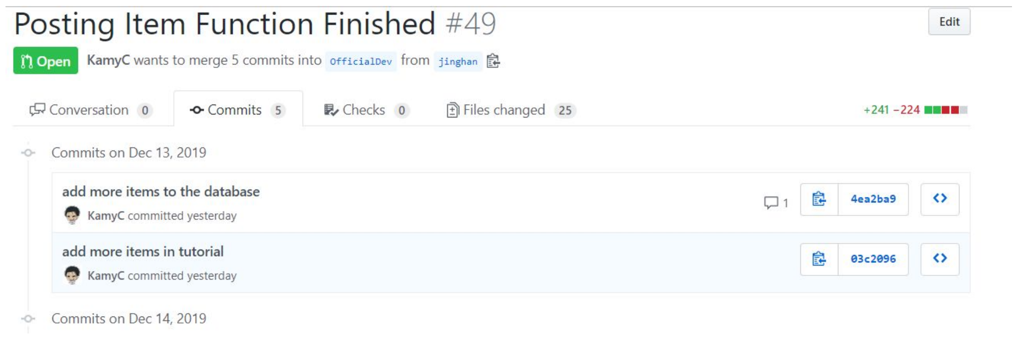
`adding views folder` 04eede3

janka102 commented on Sep 22

This looks good. One question is where do we put our own profile/about page?

Reviewers: No reviews
Assignees: No one—assign yoursel
Labels: None yet
Projects: None yet

Past Commits are available



Posting Item Function Finished #49

Edit

Open **KamyC** wants to merge 5 commits into `officialDev` from `jinghan`

Conversation 0 | Commits 5 | Checks 0 | Files changed 25 | +241 −224

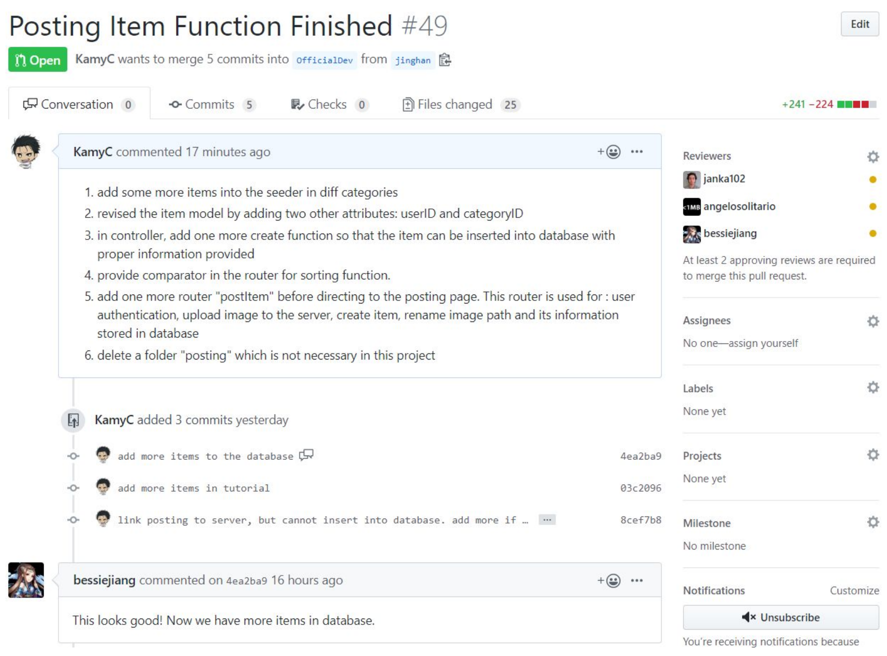
Commits on Dec 13, 2019

add more items to the database
KamyC committed yesterday — 1 — 4ea2ba9

add more items in tutorial
KamyC committed yesterday — 03c2096

Commits on Dec 14, 2019

New pull request waiting for code review


Posting Item Function Finished #49

Open KamyC wants to merge 5 commits into `officialDev` from `jinghan`

Conversation 0 · Commits 5 · Checks 0 · Files changed 25 · +241 −224

KamyC commented 17 minutes ago

1. add some more items into the seeder in diff categories
2. revised the item model by adding two other attributes: userID and categoryID
3. in controller, add one more create function so that the item can be inserted into database with proper information provided
4. provide comparator in the router for sorting function.
5. add one more router "postItem" before directing to the posting page. This router is used for : user authentication, upload image to the server, create item, rename image path and its information stored in database
6. delete a folder "posting" which is not necessary in this project

KamyC added 3 commits yesterday

```
add more items to the database                                        4ea2ba9
add more items in tutorial                                            03c2096
link posting to server, but cannot insert into database. add more if …   8cef7b8
```

bessiejiang commented on 4ea2ba9 16 hours ago

This looks good! Now we have more items in database.

Reviewers: janka102, angelosolitario, bessiejiang

At least 2 approving reviews are required to merge this pull request.

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Notifications: Customize · Unsubscribe

You're receiving notifications because

5. Self-check on best practices for security – ½ page

<u>Major assets we are protecting:</u>

1. Information system
2. User database
3. Individual user record

We protect the information system by using security procedures to prevent unauthorized users from accessing any information and protect sales item data and images. User database and individual user record will be protected by requiring users to authenticate themselves when logging into their account. Once users log into their account we can track system usage to increase security and prevent users from gaining access to confidential data.

<u>Password encryption in database:</u>

Confirmed password is encrypted in database.

<u>Input data validation:</u>

The search bar input validation has been confirmed to have up to 40 alphanumeric characters.

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
DONE
3. Selected application functions must render well on mobile devices
DONE
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
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.
DONE
8. Application shall be very easy to use and intuitive.
DONE
9. Google analytics shall be added
DONE
10. No email clients shall be allowed
DONE
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.
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 Software Engineering Project CSC 648-848, Fall 2019. For Demonstration Only" at the top of the WWW page. (Important so as to not confuse this with a real application).
DONE