SW Engineering CSC648/848 Spring 2019

GatorRooms.com

Team 9 Milestone 4

QA and Usability Testing and
Final Commitment for Product Features

Inez Wibowo
Team Lead & Document Editor
iwibowo@mail.sfsu.edu

Aliaksei Siarheyeu
Front End & Back End Lead
Github Master

Marcus Wong
Front End & Back End Lead

Romeel Chaudhari
Back End Developer

Ismael San Juan
Front End Developer

Hang Li
Front End Developer

Jia Nan Mai
Front End Developer

Versions

History Table

Version	Date	Notes
ver 1.1	5/8/19	First submission Milestone 4
ver 1.2	5/19/19	Revised version after integrating prof feedback

1. Product Summary

The rental website GatorRooms.com is a website for San Francisco State University students. SFSU students will find it easier to find rentals on a website designed specifically with features important to students. GatorRooms therefore stands out by prioritizing sorting by price and by distance to campus. The website is also lightweight (support ~50 users at a time) and it's easy for students to find rentals on the go, i.e. on mobile. Registered users/landlords on GatorRooms.com can also post their available rentals easily and will be able to post only after the post has been verified by GatorRooms.com admin.

We hope students and registered users alike find GatorRooms a website indispensable and spread the availability of the site through word-of-mouth, through Slack channels, and posters across campus.

All major committed functions this team will deliver (P1):

User - Guest User

❏ User shall be able to browse the list.
❏ User shall be able to sign up and register.
❏ User shall be able to sort the list ascending, descending date of posting and monthly rent price.
❏ User shall be able to click display the property's larger version of the image.
❏ User shall be able to review the listings quickly whether they are on mobile or desktop.
❏ User shall be able to sort based on distance to the campus.
❏ User shall be able to review the About page and read team's profile on mobile and desktop easily.

User - Registered User

❏ Users shall be able to login to their accounts.
❏ Users shall be able to post after the post have been approved by admin.
❏ Users / Landlords shall be able to publish multimedia like images so that students can view it.
❏ Users/ Renters shall be able to send a message from their dashboard to the landlord/listing owner.
❏ Users / Landlord shall have a dashboard allowing them to review messages from user and contact the student.

User - Admin

❏ Admins shall be able to login as admin.
❏ Admins shall be able to review, remove, approve listings before they go live.
❏ Admins shall be able to reject but not modify the listings from customers.

URL is a temporary AWS instance, which will be provided on demo day, locally it is http://localhost:3000

2. Usability Test Plan

We are going to test the Posting function for usability (2.5 page max)

1. Test Objectives (about half a page)

We will test the Post function. When a landlord needs to start a new listing posting is one of the most important function for this rental web application. It offers the landlord a tool to share their property to students who are looking for a place to rent. On the posting page, there are 11 fields that allow landlords to enter detailed information about their property, such as Title, Price, Address, City, State, Zip, Description, and dropdown boxes to further categorize Housing Type, Bedrooms, Bathrooms. Upload buttons allow landlord to upload images of their property and at end of the page, there are two buttons, reset and submit. Reset button allows users to delete all fields that is inputted, submit button allows users to submit everything that is ready to post. We will check these tasks and record results of the test to see whether the software is easy to use. We will give the user a task and he/she will navigate the pages. We'll collect input and how user feels after doing this task by using a Likert scale survey.

2. Test Description (total up to 1 page)

a. **System setup :**

 A user needs to launch a web browser on a laptop. The evaluator will be close by. At home of the tester.

b. **Starting point:**

The starting point of post function is the homepage.

c. **Intended Users:**

 An older person, looking to post an apartment listing.

d. **URL of the system to be tested:**

 http://localhost:3000/home

e. **What is to be measured (User Satisfaction evaluation Likert Tests):**

Methodology:

- First: test that available functions allow completion of main tasks/use cases
- Measure task completion: % of people who completed the defined task in defined time
- Count errors per task

What will be measured:

- We will measure the effectiveness, specifically: We'll test Post function and ensure it can be succesfully completed by the user.
- We will measure the efficiency, we'll test that the posting task can be completed quickly and easily.

The following is a result of testing tasks for Effectiveness and Efficiency

Test/Use Case	% Completed <Effectiveness>	Errors <Effectiveness>	Average time spent/Clicks <Efficiency>	Comments
Find the post button	100	No	1s	submit button is located at bottom
Start a post	100	no	1min	works fine
Submit post	100	no	5s	works fine
Fill out the form	100	No	20s/1 click	works easily
Upload an image	100	No	3s	it is pretty easy

- Satisfaction:Testing satisfaction using Likert scale with simple questions from Qualtrics survey https://sfsu.co1.qualtrics.com/jfe/form/SV_bBOhvNlWzqqorsN

Below are qualitative results from Qualtrics Survey small sample size n=3, 73% satisfaction

Questions <Satisfaction>	Rating (1-5)
Overall, how satisfied were you with GatorRooms?	3
Based on your experience with GatorRooms, how likely are you to purchase their products or services again?	3
Based on your experience with GatorRooms, would you recommend their products or services to a friend or family member?	4
Average	3.6 (73%)

3. Usability Task description for Testers

These are the tasks the testers will conduct during testing:

a. Tester will provide user a list of actions as shown above, i.e. "Find the post button", "Start a post", etc.

b. Tester will record the Effectiveness, Efficiency, and Satisfaction metrics and record them in a table like the above
c. Tester will deliberately leave it to the user to complete the task to be able to measure the intuitiveness of the UI
d. Goal of the tester is to expose usability flaws and document them

4. **Questionnaire , provided here are 3 general Likert scale question format to gain insight into user satisfaction**

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Post function is easy to find.					
I have all the fields required to create a high quality listing.					
Posting is intuitive and easy to use.					
I would recommend GatorRooms to a friend.					
Comments:	<Free-form field>				

3. QA Test Plan

1. **Test Objectives**

   Check if the software is according to specs.

   We will give the user a task and he/she will navigate the pages.

   We'll collect input on whether or not the experience is as written in the specs, PASS or FAIL

   Make sure there aren't any bugs

   Perform these test on Chrome and Safari:

   Test Homepage search

   Test Login

   Test registration of new Listing Post
2. HW and SW setup (and the URLs)ng to post from home page

   https://localhost:3000/home

   https://localhost:3000/new

   https://localhost:3000/login

   Perform Tests on Safari and chrome installed in different systems, such as windows, mac, test data, test database, front end running environment

3. Features to be tested

   Posting flow:

   a. Navigation to posting a listing
   b. Creating a listing
   c. Filling out the form
   d. Submitting the form for approval
   e. Review the listing as a registered user
4. Risk of data loss due to improper process implementation, failed system or some external events risk
5. QA Test Plan (include 3 test cases and results of testing them on your app about 1 page, use table from class. Suggested format for QA Test Plan Table: table columns are: test #; test title; test description; test input; expected correct output; test results (PASS or FAIL for each tested browser).

Test Number	Test Title (Unit/Smoke/	Test Description	Test Input	Expected Correct	Test Results (PASS/FAIL)

	Regression)			Output	
1	Unit	Testing to find posting on Homepage	Enter in search field zipcode “10000”	GET 3 results all have “10000” in zip field	PASS(chrome) PASS(safari)
2	unit	Search for listing post in home page	Enter in search field “listing”	Get 1 result with “Listing1” in the description	PASS(chrome) PASS(safari)
3	Smoke	Login User	Try login in with invalid credentials like “...” for username and password entered in the inputs	Get error username and password are invalid	PASS(chrome) PASS(safari)
4	Regression	Adding new listing	Add a new posting “new apartment” to database	Data “new apartment” inserted	PASS(chrome) PASS(safari)

4. Code Review (Alex & Marcus)

4a. Coding style we have chosen:

Our project follows the basic folder structure for any nodeJS web app.

https://docs.npmjs.com/misc/coding-style.html

Coding style

https://github.com/CSC-648-SFSU/csc648-sp19-team09/blob/master/Coding_Style.txt

Additionally to coding style and documentation, we have two main folders which divide the server side code and the client side code

A. Client side

Folder Structure

All React components are divided into their own folders which contain related styles or other components related

We have one front end api folder that contains an api for all the frontend endpoint calls. For our simple, api we categorize requests by their action type for example all user requests are contained in a user.actions.js file.

B. Server side

Folder Structure

Server side folder structure is very basic and simplified.

We are using an ORM to model our SQL database tables so we contain all our model definitions in the models folder.

We have one express router per model which handles all HTTP requests for the model. All these routes are contained in one routes folder.

Naming

Models files are named with lowercase and dashes substituting spaces

Model variables are named following Java's class naming convention, that is they are using UpperCamelCase. lowerCamelCase is used for all other variables.

C. Code Clean up

Code clean up is done periodically and typically before or during pull requests. We utilize the

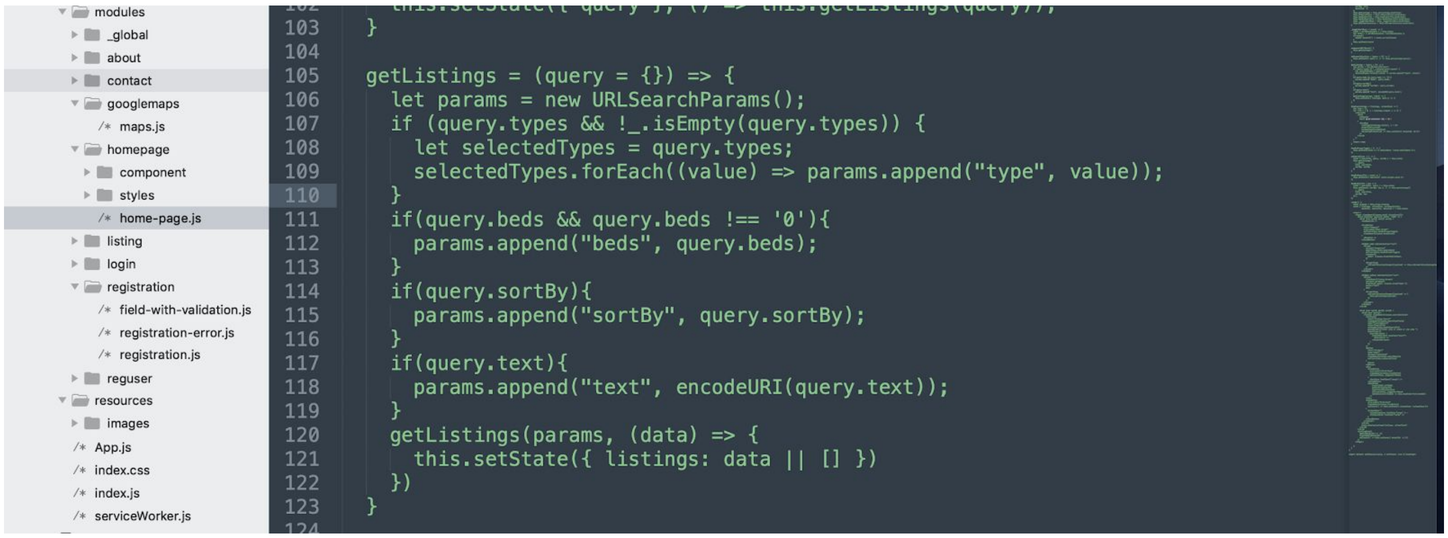
Javscript ES6 syntax, for example to reduce code amount.
https://www.w3schools.com/js/js_es6.asp

4b. Here is an example of the code under review : Github and Code Review

Overview

- Code review is managed through github pull requests.
- Team members are expected to submit working tested code before making a pull request.
- Since all new code is integrated with pull request, we have the ability to do code review on all code.
- Conflicting code will also be resolved during pull request.
- All reviews are reviewed by our main lead software engineer Alex.
- As part of review, the code reviewers Alex and Marcus also check for header and in-line comments to make sure they are descriptive.

- Code Review by Marcus
  Reviewing Front End HomePage component



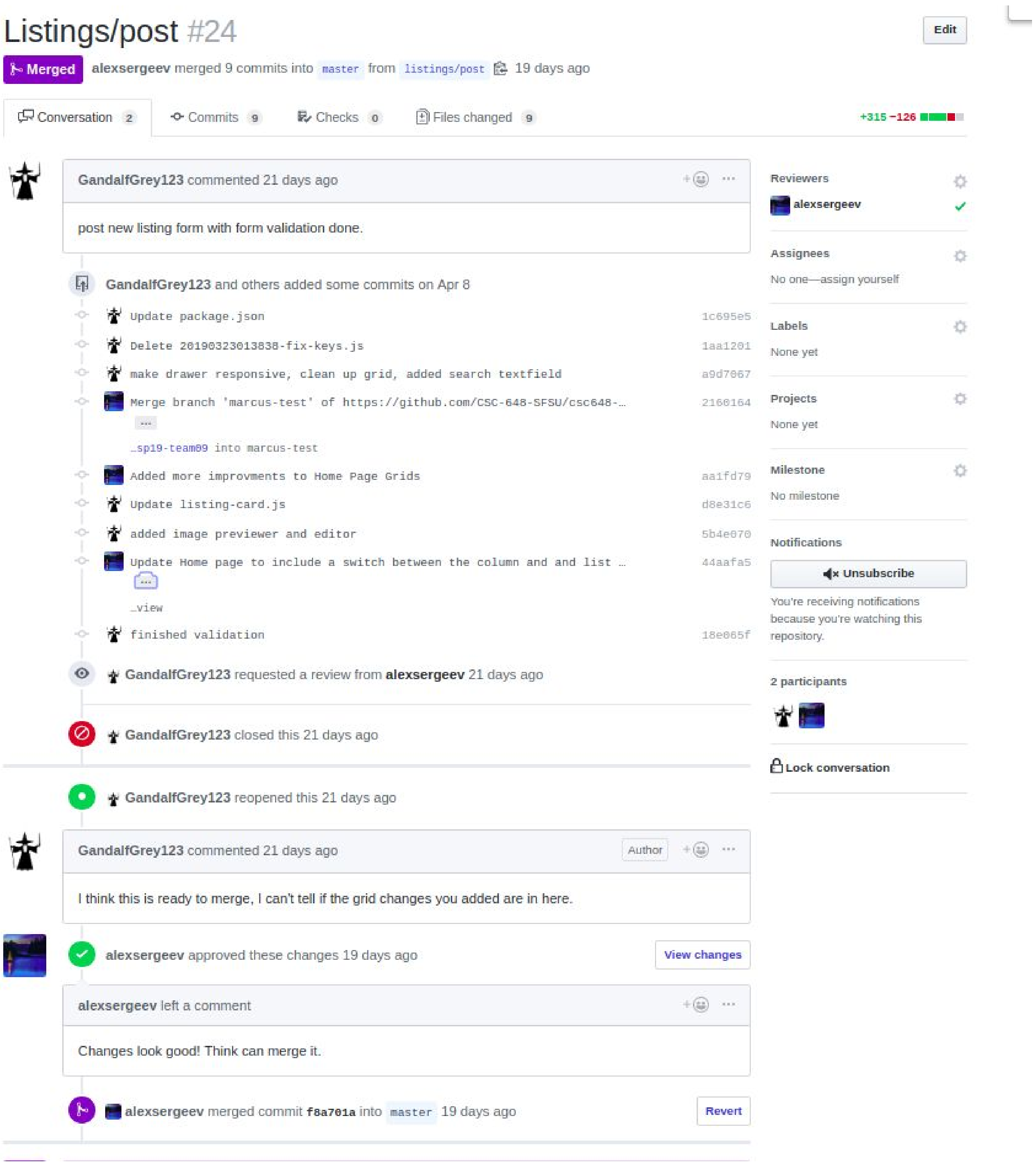
```
  }

  getListings = (query = {}) => {
    let params = new URLSearchParams();
    if (query.types && !_.isEmpty(query.types)) {
      let selectedTypes = query.types;
      selectedTypes.forEach((value) => params.append("type", value));
    }
    if(query.beds && query.beds !== '0'){
      params.append("beds", query.beds);
    }
    if(query.sortBy){
      params.append("sortBy", query.sortBy);
    }
    if(query.text){
      params.append("text", encodeURI(query.text));
    }
    getListings(params, (data) => {
      this.setState({ listings: data || [] })
    })
  }
```

The HomePage class is getting too large, and it will get larger if we have more searching mechanisms. But to clean up this code a bit and make our project more maintainable, we could implement an OOP design pattern to reduce some of the responsibility of the HomePage class.

For example in the code snippet above, we could use the Strategy Design Pattern to build Query classes. Therefore building the queries would not be the HomePage's responsibility.

- Code Review by Alex Reviewing Listing/Posting functionality
  Review below includes pull request/code review requested by Marcus to Alex, through Github.



Listings/post #24

Edit

Merged alexsergeev merged 9 commits into `master` from `listings/post` 19 days ago

Conversation 2 · Commits 9 · Checks 0 · Files changed 9 · +315 −126

GandalfGrey123 commented 21 days ago

post new listing form with form validation done.

GandalfGrey123 and others added some commits on Apr 8

```
Update package.json                                                        1c695e5
Delete 20190323013838-fix-keys.js                                          1aa1201
make drawer responsive, clean up grid, added search textfield              a9d7067
Merge branch 'marcus-test' of https://github.com/CSC-648-SFSU/csc648-…     2160164
…sp19-team09 into marcus-test
Added more improvments to Home Page Grids                                  aa1fd79
Update listing-card.js                                                     d8e31c6
added image previewer and editor                                           5b4e070
Update Home page to include a switch between the column and and list …    44aafa5
…view
finished validation                                                        18e065f
```

GandalfGrey123 requested a review from **alexsergeev** 21 days ago

GandalfGrey123 closed this 21 days ago

GandalfGrey123 reopened this 21 days ago

GandalfGrey123 commented 21 days ago — Author

I think this is ready to merge, I can't tell if the grid changes you added are in here.

alexsergeev approved these changes 19 days ago — View changes

alexsergeev left a comment

Changes look good! Think can merge it.

alexsergeev merged commit `f8a701a` into `master` 19 days ago — Revert

Reviewers
alexsergeev ✓

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe
You're receiving notifications because you're watching this repository.

2 participants

Lock conversation

5. Self-check on best practices for Security (½ page Romeel)

Some user data assets we are protecting are:-

1. User Information

   The user information includes email-id and the password used during the login and registration is encrypted in the database. In this way, the user information cannot be obtained in a malicious manner.

2. Passwords

   To protect users identity, we are encrypting masking the password entry. We further encrypt the password in the database as described below.

3. Image validation

   Whether the image of a listing is declined/approved/pending. The approval of the image will depend on the admin. It is same as admin approval of the listing. If the image is approved it can be posted. If declined or pending it will be held back with the listing.

4. Encryption of Passwords (PW) in the Database

   Passwords in the database are not clear text. We are hashing the combination of the username and password to encapsulate the password.

5. Data validation

   Data validation – In our application, we are validating the search bar input based on the city, zip, state. We are also encrypting user Login and registration information. The textfield data validation is also done during the login and registration process.

6. Self-check Adherence to original Non-functional specs (Jia nan)

6.1. 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
6.2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	DONE
6.3. Selected application functions must render well on mobile devices	DONE
6.4. Data shall be stored in the team's chosen database technology on the team's deployment server.	DONE
6.5. No more than 50 concurrent users shall be accessing the application at any time	DONE
6.6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	dONE
6.7 The language used shall be English.	DONE
6.8. Application shall be very easy to use and intuitive.	DONE
6.9. Google analytics shall be added.	DONE
6.10. No e-mail clients shall be allowed.	DONE
6.11. Pay functionality, if any (e.g. paying for goods and services) shall not be	Not Applicable

implemented nor simulated.	
6.12. Site security: basic best practices shall be applied (as covered in the class).	DONE
6.13. Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator	DONE
6.14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.	DONE
6.15. The website shall <u>prominently</u> display the following <u>exact</u> text on all pages *"SFSU Software Engineering Project CSC 648-848, Spring 2019. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application).	DONE