SW Engineering CSC 648/848 Spring 2019

Gator Housing

Team 12

Andrew St. Germain (Team Lead)
Sagar Pandya (Backend Lead)
Steven Apicella (Frontend Lead)
David Adams (Document Master)
Sunny Wong (Github Master)
Ganzolboo Ayurzana (Frontend member)
Peter Lin (Frontend member)

Milestone 4

May 8, 2019

Version history can be seen on Google Drive here:
https://docs.google.com/document/d/15WlpiG261QIKrSNfnxkzKW34Oa0kJhicmv0PsLs-GDo/edit?usp=sharing

1. Product Summary

As a group of students currently studying Computer Science at San Francisco State University, we all know how difficult it can be to look for the perfect place to live. Many factors come into consideration when deciding where to live, which is why we have decided to build and launch a website named “Gator Housing” to help students look for housing. Gator Housing is a website specifically tailored to the needs of students at San Francisco State University. The features that will be included at the launch of Gator Housing are as follows:

1. Text Search: Users will be able to use text search to search for postings with the same key words.
2. Sort by price: Users will be able to sort by the price of the unit.
3. Sort by number of bedrooms: Users will be able to sort by the number of rooms in the unit.
4. Registration: New users will be able to register to become a user.
5. Login: Registered users will be able to login to contact landlords.
6. Administrator: An administrator will be required to approve/remove posts.
7. Messages: Registered users will be able to send a message to the landlord.
8. Post: Registered users will be able to post their own listings and remove their listings.

On top of these included features, what makes Gator Housing unique is the fact that this website is designed primarily for SFSU students. The website is populated by users from within the community, which will make it a safer option to use. A safety feature is that users can contact each other through an internal messaging system, which only registered users can use. Therefore, users feel safer by not having to post their personal contact information on the web or worry about coming in contact with risky phone numbers, emails, etc. The website can be accessed using the following URL: http://ec2-54-153-63-128.us-west-1.compute.amazonaws.com:3000/

2. Usability Test Plan

Test Objectives:

The feature that we will be testing for usability is the search function. This feature will be tested for its effectiveness, efficiency and satisfaction.

Test Description:

The user will be given a laptop with the homepage of Gator Housing loaded up on Google Chrome.

The starting point of the test will be the homepage of Gator Housing.

Usability Task Description:

The user will be asked to pick a key term from a list to search. After looking at the result, user should use the search bar from the results landing page to search for another key term. The user should click on a result and then from that page, they then will be allowed to search for their own key terms in the search bar.

Questionnaire:

1. I found this feature easy to use (circle one):
   Strongly Disagree Disagree Neutral Agree Strongly Agree

2. I found this feature to be fast enough (circle one):
   Strongly Disagree Disagree Neutral Agree Strongly Agree

3. Overall I am satisfied with this feature (circle one):
   Strongly Disagree Disagree Neutral Agree Strongly Agree

Comments:

3. QA Testing

3.1 Test Objective:

Stress test the search function with a basic search, search with a category selected and search with category selected plus price filter.

3.2 Hardware and Software Setup:

Software Setup: The server should already be running before beginning any search tests.
Hardware Setup: A laptop or desktop that can access the internet.

3.3 Features to be Tested:

Search bar, category selection and price filter.

3.4 QA Test Table:

Number	Test Title	Test Description	Input	Expected Output	PASS/FAIL
1	Basic Search	Test % like search in the search bar	“parkmerced”	Show 6 results, all with “parkmerced” under bolded post title	PASS
2	Search a keyword in a specific category	Test %like search with category selection	Keyword “park” and category “house”	Show only 2 houses having “park” under bolded title	PASS
3	Search a room under $1000	Test %like search with category selection and price filter	Keyword “park”, category “room” and price filter	Show 3 apartments having “park”under bolded title	PASS

			“under $1000”	and price less than $1000	

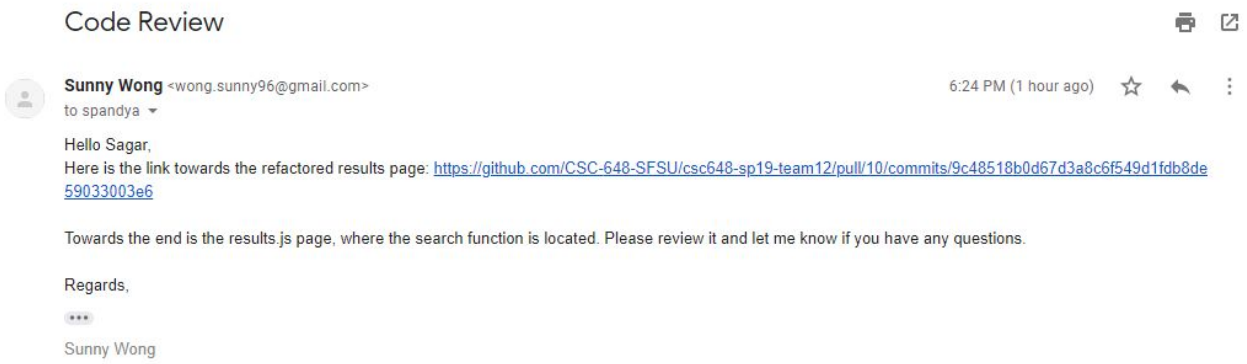
4. Code Review

Snippet of the code:

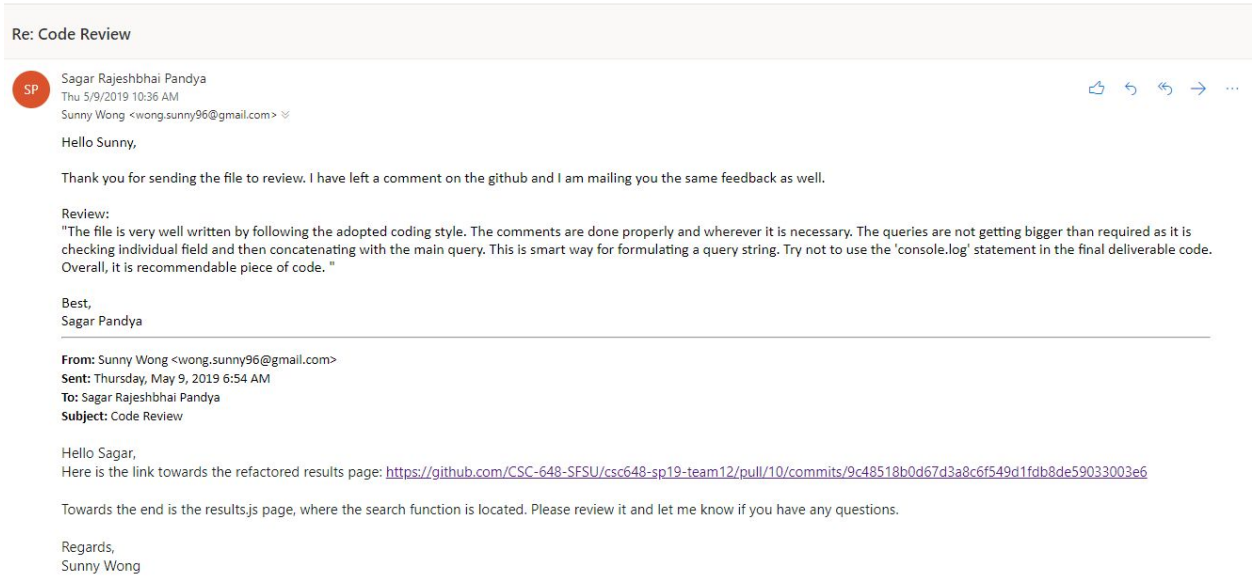
```
// Results page
var express = require('express');
var router = express.Router();
var db = require('./db');

router.get('/', search, (req, res) => {
    var searchResult = req.searchResult;
    console.log(searchResult);
    // Tells node to render this ejs file named results
    res.render('results', {
        // Ejs variables being passed into results.ejs
        results: searchResult.length,
        searchTerm: req.searchTerm,
        searchResult: searchResult,
        searchCategory: req.query.category,
        sortType: req.query.sortType,
        priceFilter: req.query.priceFilter,
        distanceFilter: req.query.distanceFilter
    });
});
```

Before review:



Post review:



Re: Code Review

SP Sagar Rajeshbhai Pandya
Thu 5/9/2019 10:36 AM
Sunny Wong <wong.sunny96@gmail.com>

Hello Sunny,

Thank you for sending the file to review. I have left a comment on the github and I am mailing you the same feedback as well.

Review:
"The file is very well written by following the adopted coding style. The comments are done properly and wherever it is necessary. The queries are not getting bigger than required as it is checking individual field and then concatenating with the main query. This is smart way for formulating a query string. Try not to use the 'console.log' statement in the final deliverable code. Overall, it is recommendable piece of code. "

Best,
Sagar Pandya

From: Sunny Wong <wong.sunny96@gmail.com>
Sent: Thursday, May 9, 2019 6:54 AM
To: Sagar Rajeshbhai Pandya
Subject: Code Review

Hello Sagar,
Here is the link towards the refactored results page: https://github.com/CSC-648-SFSU/csc648-sp19-team12/pull/10/commits/9c48518b0d67d3a8c6f549d1fdb8de59033003e6

Towards the end is the results.js page, where the search function is located. Please review it and let me know if you have any questions.

Regards,
Sunny Wong

5. Self-Check on best practice for security

- User table, Post table are being protected
- Passwords are encrypted
    - Using bycryptjs node module to encrypt all the passwords upon successful registration

```
bcrypt.genSalt(10, (err, salt) => {
bcrypt.hash(password, salt, (err, hash) => {
    // Storing hash password in DB
    let query = ` INSERT INTO users (name, email, password)
                  VALUES ('${userName}', '${email}', '${hash}')`;
```

- Posts are validated
- Search is validated
- Registration is validated
- Login is validated
    - Validation of any input fields while using the express-validator node module

```
req.checkBody('password')
    .not().isEmpty().withMessage("Password cannot be empty")
    .isAlphanumeric().withMessage('Alphanumeric characters only')
```

- Login authenticated through passport node module using Local Strategy and this function

```
// Need to be authenticated to view user dashboard
module.exports ={
    ensureAuthenticated : (req, res, next) => {
        if(req.isAuthenticated()){
            return next();
        }
        req.flash('danger', 'Please log in to access the dashboard');
        res.redirect('/login');
    }
}
```

- Admin authenticated through LocalStrategy, with function above
    - Checking for admin access after authentication

```
function checkAdmin(req, res, next){
   if(req.user!=undefined) {
       // req.user is an object, can call data using th
       // Checks if a user has admin access
       if (req.user[0].isAdmin != 1) {
           req.flash('danger', "Unauthorized access");
           res.redirect('/');
       } else {
           // If they do, then continue to the next
           next();
       }
   }
}
```

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. 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. 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. ONE 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. DONE.
9. Google analytics shall be added. ON TRACK.

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. DONE.
12. Site security: basic best practices shall be applied. DONE.
13. Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator. DONE.
14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. DONE.
15. The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Spring 2019. For Demonstration Only"* at the top of the WWW page. ON TRACK.