SW Engineering CSC648/848 Section 01 Spring 2019 Milestone 4

SAHARA LLC.

Oasis

Team 07

Team Members:

Ratna Lama:	**Team Leader**	**rlama7@mail.sfsu.edu**
Ade Adetayo:	**Front-End Lead**	
Andrew Sarmiento:	**Back-End Lead**	
Adam Tremarche:	**GitHub Master/Document Editor**	
Hadia Andar:	**UI Designer**	
Shuyuan Deng:	**Database Designer**	

5/7/19

Version History:

5/7/19: document submitted for review

Table of Contents:

Product summary

Name of product: OASIS

Itemized list of functions:

All **unregistered** users shall be able to:

1. Register a new account provided they pass a captcha test.
2. View the application's terms of service.
3. Log into an existing account.
4. Visit the homepage to use the search engine.
5. Filter search results by price minimum and maximum.
6. Filter search results by distance to campus.
7. View search results featuring thumbnails of uploaded images.
8. View a property details page containing: details, an image, and a map of the available rental.
9. Visit the about page to learn more about our team.

All **registered** users shall be able to:

1. Retain all functions of unregistered users.
2. Log out from their account.
3. Message a landlord via the contact button on the property details page.
4. Post a rental property containing: the property type (room, apartment, house) one image, an address (street address, unit number, zip-code), the monthly rent, the numbers of rooms, the number of bathrooms, and a short description.
5. Access the user dashboard which contains: their inbox of messages and a list of properties offered for rent by the user.

Admins shall be able to:

1. Retain all functions of registered and unregistered users.
2. Access their admin dashboard.
3. View properties that need review, and then approve or deny those properties, view a list of all registered users, and ban users.
4. View all listings and remove any if necessary.
5. View all users and ban any if necessary.

What is unique to product:

Oasis features easy login and user registration. It's simple and easy to navigate. Using the search feature, a user can filter the results by varying distances to campus. The listing description page comes complete with a map next to the picture so the user can know the location of the listing. Messaging the poster is integrated into the "contact seller" button on the description page.

URL: http://52.53.190.148/

Usability test plan

Test Objectives:

For our usability test we will look at the robustness of our search and filter features. Search is key to using our product effectively, so it is imperative for us to know if our search feature is implemented effectively. Our search bar can filter between the three types of listings (Houses, Apartments, and Rooms) and then our home screen offers options to further refine these results by sorting from low to high or high to low and further limit the results based on distance to campus. Our test will have users accessing all of these features.

Test Description:

Our usability test will be performed over the internet with testers connecting to the OASIS website using both Google's Chrome and Mozilla's Firefox web browsers. The URL of our website is http://52.53.190.148/ which will lead our testers directly to the homepage of our website where our search feature is accessed.

The intended users of OASIS consist of internet users within the San Francisco Bay Area who are looking to lease rental properties and San Francisco State University students who are looking for properties to rent. Because of the large population targeted by our product, our users will most likely range in familiarity with website based rental platforms. We expect traffic from both first-time users and experienced and comfortable visitors, and we hope to capture a significant portion of this audience by making our website accessible, easy, and intuitive to use.

We will ask our testers to fill out a Lickert scale survey wherein we will determine the user satisfaction level after compiling scores from multiple testers.

Usability Task description:

Our testers will attempt 3 different searches utilizing different filter options for each one:

- First, they will attempt a search of all properties ranging in price from 0 to 800 dollars. They will then sort this list of results from low to high.

- Next, our testers will be asked to search for rooms ranging in price from 0 to 500 dollars. For this search, the testers will then filter the results to include only those results which are within 5 miles of campus.

- Finally, our testers will try a search for just houses ranging from 500 to 1000 dollars. The tester will then be asked to sort the results from high to low and filter out results that are more than a mile away from campus.

After attempting these three searches our testers will complete the following questionnaire:

Questionnaire:

Using the property type filter was simple and intuitive.

◉ Strongly Agree
◉ Agree
◉ No Opinion
◉ Disagree
◉ Strongly Disagree

Sorting the list from low-to-high and high-to-low is simple and intuitive.

◉ Strongly Agree
◉ Agree
◉ No Opinion
◉ Disagree
◉ Strongly Disagree

Narrowing the search further using the “distance to campus” feature is simple and intuitive.

◉ Strongly Agree
◉ Agree
◉ No Opinion
◉ Disagree
◉ Strongly Disagree

QA Test Plan

Test objective:

To ensure that search function works accordingly. Search function is tested to provide accurate result of house listings based on search filters.

Setup of HW & SW:

We would be testing using a Windows 10 and Mac OS 10.14 Mojave. Browsers used would be Google chrome and Mozilla Firefox

URL: http://52.53.190.148/

Feature to be tested:

The search function would be tested.

QA test plan:

Number	Description	Test input	Expected Output	Pass/Fail
1	Test Search for an item on a specific search field like room	“Room to share”	Shows 3 results	Chrome: Pass Firefox: Pass
2	Test Search for a number	“94132”	Shows 4 results	Chrome: Pass Firefox: Pass
3	Test Search with empty field	“ “	Shows all results	Chrome: Pass Firefox: Pass

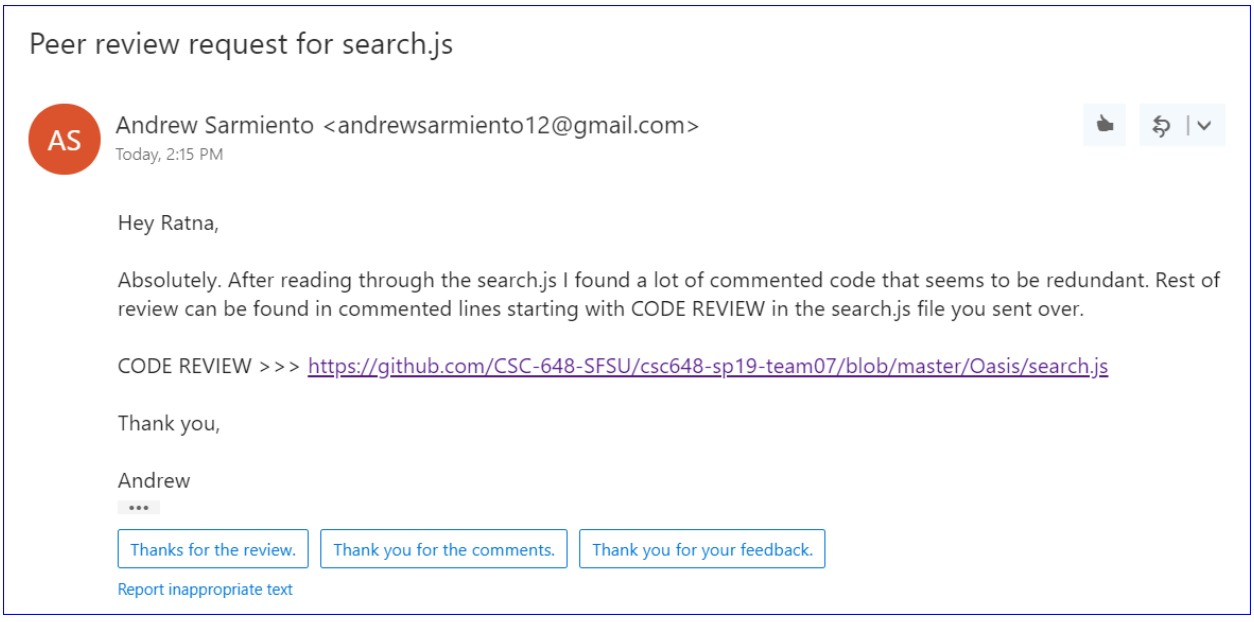
Code Review

Coding Style

Our coding style is simple and self-explanatory. We name our variables and functions such that it makes it easy for us to look back later and understand what we did. We push each part of the code we worked on to our own individual branches and we merge to the develop branch once it is tested and works. We push to Master the code that we know works perfectly before every milestone. Master always has a working version of our page.

Code Review: search.js

Email Exchange:



Code excerpts from search.js

```
12  const express = require("express");
13  const app = express();
14
15  const bodyParser = require("body-parser");
16  const port = 80;
17
18  // CODE REVIEW
19  // There are a lot of variables here that only I know its purpose
20  // My try to add short comments here and there to better explain
21  // the prupose to others who haven't worked on this code
22  var img_url = [];
23  var prop_type = [];
24  var prop_add = [];
25  var prop_city = [];
```

```
84      data: count,
85      resultCount: totalCount,
86      dropCount: drop_count,    // CODE REVIEW
87      listImg: img_url,         // This is a lot of variables to pass
88      type: prop_type,          // Consider only the ones that are
89      address: prop_add,        // absolutely necessary
90      city: prop_city,
91      state: prop_state,
92      zipcode: prop_zipcode,
93      price: prop_price,
94      size: prop_size,
95      room: prop_room,
96      bathroom: prop_bathroom,
97      min: min_price,
98      max: max_price,
99      all: select_all,
100     room: select_room,
101     apartment: select_apartment,
102     house: select_house,
103     distance: prop_distance,
104     none: select_none,
105     first: select_first,
106     second: select_second,
107     third: select_third
108   });
```

```
256
257   /**
258    * Search database that match user inpur parameters of min and max price
259    * @param {*} min_price
260    * @param {*} max_price
261    */
262   // CODE REVIEW
263   // Before turning in final project, consider cleaning up commented
264   // code that you don't need anymore
265   function search(min_price, max_price) {
266     // let totalCount = countResult(min_price, max_price);
267     // query if the result count is greater than 1
268     // console.log('total count is: ' + totalCount);
269     // console.log("TOTAL COUNT: " + totalCount);
270
271     // try {
272     //   //console.log("Again TOTALCOUNT_3 : " + totalCount);
273     //   if (totalCount < 1)
274     //     throw "Sorry found no matching result. Please try again with different price range.";
275     //   if (totalCount > 0) {
276     //     let sql = "SELECT FROM property where price >= ? AND price <= ?";
277     //     db.query(sql, [min_price, max_price], function(err, result, field) {
278     //       if (err) throw err;
279     //       // let item = JSON.stringify(result);
280     //       // console.log("Item result" + item);

434           prop_state.push(result[i].state);
435           prop_zipcode.push(result[i].zipcode);
436           prop_price.push(result[i].price);
437           prop_size.push(result[i].size);
438           prop_room.push(result[i].room);
439           prop_bathroom.push(result[i].bathroom);
440           prop_distance.push(result[i].distance);
441         }
442         console.log(result);
443       });
444   }
445
446   // CODE REVIEW
447   // I see the purpose of every function but a single .js file
448   // with over 400 lines of code. I'd consider making a folder
449   // where you can store and call the functions when needed
450   // as to avoid cluttering all the backend code onto one page
451
```

Self-Check: Security Best Practices

Major protected assets:

Registered user information: username, first name, last name, and password
Item information: image, description, price

PW encryption: ON TRACK

Input data validation: ON TRACK

Self-Check: Adherence to Non-Functional Specs

List of Non-Functional Requirements:

5.1 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**

5.2 Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers. **DONE**

5.3 Selected application functions must render well on mobile devices. **ON TRACK**

5.4 Data shall be stored in the team's chosen database technology on the team's deployment server. **ON TRACK**

5.5 No more than 50 concurrent users shall be accessing the application at any time. **DONE**

5.6 Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **ON TRACK**

5.7 The language used shall be English. **DONE**

5.8 Application shall be very easy to use and intuitive. **DONE**

5.9 Google analytics shall be added. **DONE**

5.10 No email clients shall be allowed. **DONE**

5.11 Pay functionality if any (e.g. paying for goods and services) shall not be implemented nor simulated. **DONE**

5.12 Site security: basic best practices shall be applied (as covered in the class). **ON TRACK**

5.13 Before posted live, all content (e.g. apartment listings and images) must be approved by the site administrator. **ON TRACK**

5.14 Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. **DONE**

5.15 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. (Important so as to not confuse this with areal application). **ON TRACK**