Milestone 4

SW Engineering CSC648/848 Summer 2020

Housing State

8/3/2020

Team 5

- Garrett Johnson (lead)
- Buu Phan (github)
- Sarah Nafees (frontend)
- Mantasha Khan (backend)
- Nick Brown
- Kamelia Shaharova

Product Summary:

Name: Housing State

Committed functions

- All users shall have the capability of searching and viewing listings
- All users are required to register to contact seller or create a new listing
- Users shall register or login using their SFSU email and a password
- All users shall be able to access search from every page
- All users shall be able to reach out to renters and sellers through a message dashboard
- Users shall provide information of the listing like the cost, and picture of the house
- Registered users shall have a dashboard page where they can see their messages along with the listings they have posted.
- Users shall be contacted/be able to contabe by one in-site message
- System shall not support any type of payment system.
- System shall show all descriptions of listings to all users

URL: Housing State (http://3.17.193.189/)

Usability Test Plan

Test Objectives

The main motive of this evaluation is to assess and evaluate the Housing State website. The major function that we have selected for this usability testing is the search function. The usability test is done to evaluate if the product is user friendly, easy, usable, and fast. The test objectives for the usability study is to evaluate the search function on Housing State website which should be relative to the user's ability to:

- Search available housing either by address, city, or ZIP code
- Search desired listing using a filter

Through this testing, we are also expecting some room for improvements on usability and functionality which we can do by getting feedback from the users and evaluating the user's perception.

Test Background and Setup

The test is designed to evaluate the Housing State website in terms of user ability to easily access and search the desired listings. The website must be able to run in multiple browsers which are Google Chrome, Mozilla Firefox, and Safari. The intended users of this website are SFSU students and faculty.

To begin the usability test we will use the Google Chrome browser and start the user off at the home page of the website. We will then give the required task for the user to complete, but will not provide any assistance in completing those tasks. Once the user completes the tasks we will use the criterias in Usability Task Description table and the Likert test to determine whether the function passes the usability test.

The URL of the website is Housing State which is also the URL of the search page we are testing. The user can input the valid search query which can include the exact address, city, or ZIP code, and can also filter out searches by using the dropdown menu lists. The usability test is successful and completed if the user finds the appropriate result by searching the desired house listings available. For example, when a user opens the Housing State website and searches for all available listings in San Francisco, the user will be able to view the options related to their search query. After loading the appropriate search result according to their criteria, we can say that our usability test has been successfully completed.

Usability Task Description

TASK	DESCRIPTION
Task	Search listings in San Francisco

Machine state	Search listings loaded
Successful completion criteria	Shows related search results according to the search query.
Benchmark	Completed in 10 seconds

Effectiveness: I would measure effectiveness by looking at the percentage of people who completed the defined task in the time frame, the errors counted per task, and would allow comments or suggestions on the effectiveness of the test.

Efficiency: I would measure efficiency by looking at the average time it took to complete the task and find out the average time of those who completed the task/ average time of all users. Efficiency in effort could be known by the number of clicks it took to search a listing.

("**X**" applicable column)

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I was able to find the search bar easily.					
I was able to search for the desired listings.					
I found a related search result.					

QA test plan:

- Test objectives

  ○ Assessing the website's content through its search feature for both registered and unregistered users while validating the accuracy of the search results.

- HW and SW setup

  ○ Hardware:

    ■ MacOS running Docker

  ○ System setup:

    ■ The setup consists of first using the Google Chrome browser to run the website where we begin on the home page which is also the search page. After the application runs correctly we check if the search bar and search button exists and if some of the demo content exists. We then use the Firefox browser to run the website where we begin on the home page which is also the search page. After the application runs correctly we check if the search bar and search button exists and if some of the demo content exists.

  ○ Intended users are the students and faculty of San Francisco State University

  ○ URL: Housing State

- Feature to be tested:

  ○ Search function

- QA test plan (suggested format for QA test plan table)

  When we performed the test, we followed the table and tested the search feature on two different types of browsers (Chrome and Firefox)

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Chrome	Firefox
1	Search Bar Exists	The search bar should be visible in the main page of the application at all time	N/A	Visible Search Bar	PASS	PASS
2	Search Button Works	The search button should update the main page content when pressed	Type “San Francisco” in the search bar and press search	Update in the main page	PASS	PASS
3	Search Results are Relevant	The main page results should be relevant to the	Type “San Francisco” in the search bar and press search.	Should show listings that contain “San Francisco” in address (3 listings)	PASS	PASS

Code Review:

Done over email as specified. Screenshots below

Code review done over email as specified. Original code in black, review comments in **red**

```
import { Form, Formik } from 'formik';
import axios from 'axios';
import { TextInput, Select } from '../utils/inputs';

// good use of functional components
// perhaps we should be consistent with our backend code and use async/await instead of these promises
const searchbar = ({ setListings }) => {
  const searchListings = (query, { setSubmitting }) => {
    axios
      .get('/api/listing', { params: query })
      .then((res) => {
        setListings(res.data);
      })
      .catch((error) => {
        console.log(error);
      })
      // good use of finally!
      .finally(() => setSubmitting(false));
  };

  // insert comment about these being default values
  const formData = {
    search_term: '',
    unit_type: '',
    offer_type: '',
    bedrooms: 0,
    cost: 0,
  };
```

```
return (
  // good use of dynamic react sizing!
  <div className='row container-fluid align-item-center justify-content-center'>
    <div className='col-md-6'>
      <div className='card mb-4'>
        <div className='card-body'>
          <Formik
            initialValues={{ ...formData }}
            onSubmit={(values, { setSubmitting }) => {
              setSubmitting(true);
              searchListings(values, { setSubmitting });
            }}
          >
            {({ isSubmitting }) => (
              <Form>
                <div className='form-group'>
                  <TextInput
                    name='search_term'
                    className='form-control'
                    placeholder='Enter an address, city, or ZIP code'
                    maxLength='40'
                  />
                </div>
                <div className='mt-5 mb-1'>
                  <p className='text-left text-muted'>Advanced Search</p>
                </div>
                // maybe the "advanced search" features should be hidden by default?
                <div className='form-row'>
                  <div className='form-group col-sm-3 text-left'>
                    <label className='label-text' htmlFor='unit_type'>
                      Unit Type
                    </label>
                    <Select
                      id='unit_type'
                      name='unit_type'
                      className='form-control border drop-text'
                    >
                      <option value=''>All</option>
                      <option value='house'>House</option>
                      // Should be spelled "apartment"
                      <option value='appartment'>Apartment</option>
                      <option value='townhouse'>Townhouse</option>
                    </Select>
                  </div>
                  <div className='form-group col-sm-3 text-left'>
                    // maybe call this "Buy or Rent"
                    <label className='label-text' htmlFor='offer_type'>
                      Offer Type
                    </label>
                    <Select
                      id='offer_type'
                      name='offer_type'
                      className='form-control border drop-text'
                    >
                      <option value=''>All</option>
                      <option value='buy'>Buy</option>
                      <option value='rent'>Rent</option>
                    </Select>
                  </div>
```

```
                    <div className='form-group col-sm-3 text-left'>
                      <label className='label-text' htmlFor='bedrooms'>
                        Bedrooms
                      </label>
                      <Select
                        id='bedrooms'
                        name='bedrooms'
                        className='form-control border drop-text'
                      >
                        <option value=''>All</option>
                        <option value='1'>1+</option>
                        <option value='2'>2+</option>
                        <option value='3'>3+</option>
                        <option value='4'>4+</option>
                        <option value='5'>5+</option>
                      </Select>
                    </div>
                    <div className='form-group col-sm-3 text-left'>
                      <label className='label-text' htmlFor='cost'>
                        Cost
                      </label>
                      <Select
                        id='cost'
                        name='cost'
                        className='form-control border drop-text'
                      >
                        <option value=''>Any</option>
                        <option value='750'>$750+</option>
                        <option value='1000'>$1000+</option>
                        <option value='1500'>$1500+</option>
                        <option value='3000'>$3000+</option>
                      </Select>
                    </div>
                  </div>
                  <button
                    // Really cool that search button is disabled until the response is received. Is this
                       to prevent spamming the submit button?
                    type='submit'
                    disabled={isSubmitting}
                    className='btn btn-dark float-right mt-3'
                  >
                    Search
                  </button>
                </Form>
              )}
            </Formik>
          </div>
        </div>
      </div>
    </div>
  );
};

export default searchbar;
```

```
const Listing = require('../models/listing');
const Sequelize = require('sequelize');
const Op = Sequelize.Op;

// Nice job specifying the route of the endpoint
// route: GET /api/listing
// It looks like "next" is not used, maybe we should remove it?
exports.searchListings = async (req, res, next) => {
  try {
    let where = {
      // is the "and" condition here applying to each item below? could you be more specific here?
      [Op.and]: [
        // match and address and city or zip and unit_type and offer_type and bedrooms and cost
        // non-numeric fields use like so that we can match on a blank value '%%'
        { full_address: { [Op.like]: `%${req.query.search_term}%` } },
        { unit_type: { [Op.like]: `%${req.query.unit_type}%` } },
        { offer_type: { [Op.like]: `%${req.query.offer_type}%` } },
        { bedrooms: { [Op.gte]: req.query.bedrooms } },
        { cost: { [Op.gte]: req.query.cost } },
      ],
    };
    // good job defining the "where" above, it makes it much cleaner just reference it here
    const listings = await Listing.findAll({ where: where });
    res.send(listings);
  // thanks for catching any potential errors
  } catch (error) {
    console.log(error);
    res.sendStatus(500);
  }
};
```

Self-check - security:

Major assets

- User information – user information is a top priority. With many threats such as confidentiality of a system, and integrity, we took this seriously. We used a hook that takes the users password and encrypts it into hash.
- Listing information – this was also another top priority. Just like user information, threats of confidentiality and integrity of our system could be prevalent without the correct precautions. Since User id is a foreign key for our system and as stated above, we are using hash encryption to protect them. Our Databases are also password protected, so without the correct password one cannot access our MySQL database.

Security is a primary thing and needs to be taken seriously. We also were coming up with checks and balances for admin approval on all listings, but the professor told us to drop this and only work with it if we got everything else done.

Within our backend we ensured that when a new user signs up their password is converted into hash code so that it will be private only to that user. Since the data is encrypted in the backend, the real password never touches the database, only the secure, hashed password is entered.

Adherence to original Non-functional specs

Specs from Milestone 1

Spec	Status	Comment
The application shall be in English.	DONE	
The application shall be simple and user friendly.	ON TRACK	
The application shall display the exact text "SFSU Software Engineering Project CSC 648-848, Spring 2020" on top of the page.	ON TRACK	
The application shall be deployed in AWS	DONE	
The application shall render well in both desktop and mobile browsers	DONE	
The application shall only be for SFSU students and teachers.	DONE	
All users' data shall be stored in MySQL	DONE	
All users' data shall be encrypted and protected equally. Only the user themselves can review and edit their data.	DONE	
All posts shall be reviewed by admins before posting.	ISSUE	Since we were slightly behind during our M3 review, you suggested we skip admin functionality. This is the first thing we will do if we finish our P1 requirements
Google API shall be used.	ISSUE	We decided that integrating the google API is a P2, so we will do this only if we have time
No payment function shall be added.	DONE	
Best practices for security and privacy shall be applied.	ON TRACK	