CSC 648/848 Fall 2017

Milestone 4: Beta Launch

December 8th, 2017

HouseShop

TEAM 09

Franklin Henry Boswell, Team Lead (fboswell@mail.sfsu.edu)

Nehemya McCarter-Ribakoff, Backend Lead

Khanh Nguyen, Front End Lead

Evgeny Stukalov, Backend Dev

Kevin Lay, Backend Dev

Jesse Christiansen, Front End Dev

Revision History

Date	Revision
12/7/17	Rough draft submitted

Product Summary

- **Product Name**: HouseShop

- **List of Functional Requirements:**

Our product allows Guest to:

1. Navigate through all of our offered pages (excluding account information)
2. Create an account
3. Login
4. Manage their accounts
5. Find seller contact information
6. Sort listings based on bedrooms
7. Sort listings based on Price

Our product allows Registered Users to:

1. Find seller information
2. Sort listing based on Price
3. Sort listing based on Bedrooms
4. Manage their accounts
5. Contact listing agents

Our product allows Listing Agents to:

1. Access special “Agents” dashboard
    a. Dashboard will contain:
        i. Inbox Management
        ii. Activity on each listing
        iii. Contact for listings

Our product allows Admin users to:

1. See flagged listings that violate terms of service
2. Access terms and services
3. Delete user accounts
4. Remove user accounts from database

- Nothing is unique about our product.

- **URL**: https://sfsuse.com/fa17g09

Usability Test Plan for the Search Feature

Test Objectives:

The usability test objectives are:

- To determine design inconsistencies and usability problem areas within the user interface and content areas. Potential sources of error may include:
    - Navigation errors – failure to locate functions, excessive keystrokes to complete a function, failure to follow recommended screen flow.
    - Presentation errors – failure to locate information on web pages, or selection errors due to labeling ambiguities.
    - Control usage problems – improper entry field usage.
- Exercise the application or web site under controlled test conditions with representative users. Data will be used to assess whether usability goals regarding an effective, efficient, and well-received user interface have been achieved.
- Establish baseline user performance and user-satisfaction levels of the user interface for future usability evaluations.
- To highlight failures in the website's ability to provide the user with intuitive and easy accessibility

Our target audience are users interested in purchasing or selling homes. Groups that will participate in our usability tests will fall into multiple target audiences. The number of participants will meet satisfactory sample size requirements. These will include users that are interested in selling or purchasing homes, in addition to other types of users. The usability testing will be done remotely before beta launch.

Test Plan:

System Setup: Setup will consist of collection of users to test. The test will require a range of different types of users. Tasks to be completed will be given to all users at start of the test. After success or failure to complete the tasks given the user will be given a questionnaire form prepared prior to the testing. This information will be collected and used to improve user satisfaction and chance of success at launch.

Starting Point: After Collecting data on which type of users are willing to participate in our tests, we will have them begin by following our tasks to be completed. Test conductors will not interfere or cause any bias in the user experience. Testers will then complete our questionnaire form after completion of tasks.

Participants: The intended participates of this test will range between a couple of qualifications. Users that are causally browsing a couple hours a week for new homes. Users that are aggressively seeking a new home and browse multiple real estate

websites for multiple hours a week. As well as home owners that wish to sell their homes. Real estate agents will also be part of the participants.

Task	Description
Search for houses by zip code	Use a zip code to search for listings
Search for houses by city	Use a city to search for listings
Search for houses by state	Use a state to search for listings
Search via string of different attributes	Use search bar to find listings via multiple attributes, example full address entry

System to be tested: https://sfsuse.com/fa17g09

Questionnaire form:

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. Visually website is aesthetically and visually pleasing	1	2	3	4	5
2. Navigation to desired pages was intuitive and easy	1	2	3	4	5
3. Search results were useful and accurate	1	2	3	4	5
4. Website was quick and responsive	1	2	3	4	5
5. Process of searching was intuitive and easy	1	2	3	4	5
6. Filter system to narrow searches was useful and satisfactory	1	2	3	4	5
7. Ability to complete required tasks was easy	1	2	3	4	5
8. I would recommend HouseShop to others	1	2	3	4	5
9. All information and functionality is immediately obvious	1	2	3	4	5

QA Test Plan

Test Objectives

Test the functionality of HouseShop site's "Search" feature with all supported browsers.

Hardware and Software Configuration

Use the latest version of Google Chrome under Windows or Mac.
Repeat all test cases with the latest version of Mozilla Firefox under Windows or Mac.

Feature to be Tested

"Search" feature provides users with ability to find listings based on a search string, filter the search results based on filtering criteria, and sort the search results based on a selected sort order. Search results are displayed as a list, with a map containing locations of the listings displayed on one side.

Test cases

#	Description	Test steps	Expected output	PASS/FAIL
1	Searching for listings by state.	Pre-seeded test data: A single listing with the address "1600 Holloway Avenue San Francisco, CA" and an image exists in the site database. Another listing with the address "5740 County Road 125 Freedom, WY" and an image also exists in the site database. Steps: 1. Access https://sfsuse.com/fa17g09 with the test browser. 2. Enter the string "CA" into the field in the center of the page. 3. Click the "search" button (button with the "looking glass" image).	Single listing with the address "1600 Holloway Avenue San Francisco, CA" is displayed in the results list. Image for the listing is properly displayed.	PASS

2	Searching for listings by a substring in street address.	Preconditions: Pre-seeded test data as described in testcase #1 has to be present in the database. Steps: 1. Access https://sfsuse.com/fa17g09 with the test browser. 2. Enter the string “County” into the field in the center of the page. 3. Click the “search” button (button with the “looking glass” image).	Single listing with the address “5740 County Road 125 Freedom, WY” is displayed in the results list. Image for the listing is properly displayed.	PASS
3	“Search” text field length verification (40 characters).	Steps: 1. Access https://sfsuse.com/fa17g09 with the test browser. 2. Paste the string “Lorem ipsum dolor sit amet, consectetur!” into the field in the center of the page. 3. Try to type an extra character into the field at the end of the pasted string.	At the end of step 2: the pasted string is completely displayed in the text field. At the end of step 3: unable to type an extra character at the end of the pasted string due to field length restriction.	PASS

Code Review for the Search Feature

Gmail Franklin Henry Boswell <franklin.henry.boswell@gmail.com>

CSC 648 Fall 2017 Team 09 code review request for routes/search.js

3 messages

Evgeny Anatolyevich Stukalov <evgeny@mail.sfsu.edu> Mon, Dec 4, 2017 at 7:30 PM
To: Franklin Henry Louis Boswell <fboswell@mail.sfsu.edu>
Cc: Nehemya McCarter-Ribakoff <nmccarte@mail.sfsu.edu>

Hi Henry,

Please review the code that we wrote for the search feature.

```
//routes/search.js

var express = require('express');
var router = express.Router();
var models = require('../models');
var expressValidator = require('express-validator');

const Op = models.sequelize.Op;

router.use(expressValidator());

/* POST search page
   '/' is NOT Home page
*/
router.post('/', function(req, res, next) {

    if (req.body.city < 0) {
        req.checkBody('city', 'Error: You entered a negative number').isInt({min: 0});
    }
    req.checkBody('city', 'Search string too long').isLength({max: 40})
        .notEmpty(req.body.city).withMessage('Search field empty. Please enter an address, zip
code, city, or state')
    req.sanitize('city')
        .blacklist('!@#$%^*;+');

    var errors = req.validationErrors();
    console.log("Errors object: " + errors);
    if (errors) {
        res.cookie('errors', errors[0]);
        res.redirect('search');
        res.send(errors);
```

```
    }
    else {
        var queryBuilderArguments = {searchString : req.body.city};
        if(req.body.sortOption) {
            queryBuilderArguments.orderMode = req.body.sortOption;
        }

        models.Listing.findAll(buildListingsQuery(queryBuilderArguments)).then(function(listings) {
            res.render('search', { // render the Search/Browse page
                title: 'Search',
                listings: listings,
                previousSearchString: req.body.city,
                previousSortOption: req.body.sortOption,
                UserState: req.cookies.UserState,
                User: req.cookies.User,
                errors: req.cookies.errors
            });

            // START HOW TO GET AND USE ASSOCIATED MODELS
            console.log(models.Listing.prototype);

            listings.forEach(function(listing) {
                console.log("listing.address: " + listing.address);
                listing.getMedia().then(function(media){
                    media.forEach(function(medium) {
                        console.log("medium.id: " + medium.id + ", medium.imageFilePath: " +
medium.imageFilePath);
                    });
                });
            });
            // END HOW TO GET AND USE ASSOCIATED MODELS
        });
    }
});

router.get('/', function(req, res, next) {
    //res.sendFile(path.join(__dirname + '/index.html'));
    models.Listing.findAll()
    .then(function(listings) {
        res.render('search', { // render the Search/Browse page
            title: 'Search',
            listings: listings,
            previousSearchString: req.body.city,
            previousSortOption: req.body.sortOption,
            UserState: req.cookies.UserState,
            User: req.body.User,
            errors: req.cookies.errors
```

```
        });
    });
    res.cookie('errors', '');
});

module.exports = router;

function buildListingsQuery(queryBuilderArguments) {
    var sequelizeQuery = {
        include: [ models.Media ] // !! This line requests retrieval of the associated model.
    };

    if (queryBuilderArguments.searchString){
        sequelizeQuery.where = {
            [Op.or]: [
                {
                    address: {
                        [Op.like]: '%' + queryBuilderArguments.searchString + '%'
                    }
                },
                {
                    city: {
                        [Op.like]: '%' + queryBuilderArguments.searchString + '%'
                    }
                },
                {
                    state: {
                        [Op.like]: '%' + queryBuilderArguments.searchString + '%'
                    }
                },
                {
                    zipcode: {
                        [Op.like]: '%' + queryBuilderArguments.searchString + '%'
                    }
                }
            ]
        };
    }

    if (queryBuilderArguments.orderMode){
        sequelizeQuery.order = models.sequelize.literal(queryBuilderArguments.orderMode);
    }

    return sequelizeQuery;
}
```

Franklin Henry Boswell <franklin.henry.boswell@gmail.com>

to Evgeny

...

Code Review For The Search Feature

Coding Style: MVC - Followed

1.Best Practice:
- Good everything from good variable names to proper indenting exists. Also proper use of the modules is clearly implemented
2. Error Detection:
- No Bugs or unintended side effects detected
- Code works locally
3. Vulnerability Exposure:
- All discussed risks are mitigated
- Input is sanitized
4. Malware Discovery:
-No suspicious or unexplained code

Only comment moving forward:
- More comments would be better

Overall:
-Fantastic implementation move to production

Self-check on best practices for security

Our two major fields of vulnerability are: (1) user data, and (2) the site's server. Below we discuss the measures we take to protect our users and our site from attacks:

1. User data

   To protect users' account information, it is imperative a user's account is secure. To achieve this, we encrypt all passwords with AES-256 encryption before storing them in our database. No sensitive user data is passed via GET requests to avoid such information from becoming exposed to outside entities.

2. Server control

   To protect our server, we validate input on the following forms:

   a. Search bar
   b. Contact form
   c. Signup form
   d. Listing creation form

   To avoid cross-site scripting, inputs of length greater than 40 characters are rejected. To avoid SQL Injections, special characters are sanitized out where unneeded. We use the nodeJS Validator module to accomplish all input validation.

Self-check: Adherence to original Non-functional specs

- Application shall be developed and deployed using class provided deployment stack
    - DONE
- Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks must be explicitly approved by Anthony Souza on a case by case basis.
    - DONE
- Application shall be hosted and deployed on Google Web Services as specified in the class
    - DONE
- Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome.
    - DONE
- Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed
    - DONE
- Data shall be stored in the MySQL database on the class server in the team's account
    - DONE
- Application shall provide real-estate images and optionally video
    - DONE
- Maps showing real-estate location shall be required
    - DONE
- Application shall be deployed from the team's account on Google
    - DONE
- No more than 50 concurrent users shall be accessing the application at any time
    - DONE
- Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
    - DONE
- The language used shall be English.
    - DONE
- Application shall be very easy to use and intuitive. No prior training shall be required to use the website.
    - DONE
- Google analytics shall be added
    - DONE
- Messaging between users shall be done only by class approved methods and not via email clients in order to avoid issues of security with e-mail services.

    - DONE
- Pay functionality (how to pay for goods and services) shall not be implemented.
    - DONE
- Site security: basic best practices shall be applied (as covered in the class)
    - DONE
- Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
    - DONE
- The website shall prominently display the following text on all pages *"SFSU Software Engineering Project, Fall 2017. For Demonstration Only"*. (Important so as to not confuse this with a real application).
    - DONE