# SW Engineering CSC648/848 Section 02 Fall 17

# Real Estate Website

# "Agent 007"

# Group 07

# Local


**Team Members**

Dillan Brosnan (Team Leader)

-San Francisco State University

(dillonbrosnan11@gmail.com)

Roy Anguiano (Front-end leader)

-San Francisco State University

Indra D Gairhe

-San Francisco State University

Vikram Grewal (Back-end Leader)

-San Francisco State University

Anirudh Mohan

-San Francisco State University

Darryl Raveche

-San Francisco State University


| Revision | Changes |
|---|---|
| 12/4 | Initial state of Milestone 4 Documentation |

1. **Product Summary:**

The website will allow perspective buyers and sellers to connect to buy and sell houses of their preferences. Our website will have following feature at the minimum.

**Browsers**

Browsers will be able to see the listing in the website, search listings with price, geographic distance in mileage, zip code, number of bed and bathrooms in house. The website will allow the browsers to see the search results' location in google map. Browsers will also be able to sign up for User account and be able to contact admin as well as agent.

**Users:**
User will have all capability of browsers
Users will be able to contact agent and admin, update account information, view message from agent and admin.

**Agents:**
Agents will be able to register with separate form as an agent with additional required information which includes professional license number and business address.
Agents will also be able to respond users and admin.
Admin will also have access to dashboard for his/her listings with option to update, create new and delete listings.

The website will facilitate purchasing and selling homes in user friendly website. Users will be able see houses listed for sale within the preferred location and feature with minimal clicks in an easy-to- understand search page and agents will be able to list houses efficiently and communicate with users professionally. The web address of our final product will be https://www.sfsuse.com/fa17g07/

2. **Usability Test Plan**

Objective:
In this test, the objective is to find any bugs and technical issues related to functionality, efficiency and comfortableness of users when using search option in our website. With our focus on user centric design in developing our website, the searching on our website should be self-guiding to users of various level of knowledge on how to use internet to search. Our website allows visitors to search listing without having to register. Users can search in two ways: simple search and advanced search. Advanced option will include various search criteria such as build year, number of bathroom, price range etc. With simple search option, users will be able to search within specified distance range from specified. The correct entry to

distance range will be integer ranging from 1-300 and to the location, it will be city, state and country. This test will take less than minute to complete the task. The successful result will be a display of search result with picture and available information of the data of each result.

TEST PLAN and SET UP ENVIRONMENT:
The following set up will be required to perform the test.

- Laptop or desktop with mouse and pad or touch screen/pad to assist entering data in the search area. Browser of latest two version must be ready to be used.
- Access to Internet.
- Data server at SFSUSE/fa1g07 running.
- A user will be handed the control of the computer to enter the data to search listing.
- A tester will be present to monitor the time and result to further analyze the result.
- The context can vary from public to private such as library, public place, residence or public vehicle as long as there is internet connection, and the time can be any time to conduct this test.
- The tester will NOT instruct on how to enter search criteria but will have the home page of the website ready in the computer browser and delegate the control of computer to the user.
- The user is volunteer basis and may discontinue any moment.
- Browsers: two latest version of each of the browsers listed:
    - Internet explorer
    - Chrome
    - Firefox Mozilla

Questionnaire:

Please mark the appropriate option in relation to your experience using the website.

1. The website is easy to use.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

2. It is easy to see where search box is.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

3. It is easy to know what to enter in the search box.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

4. The search result provided me enough information needed to make my buying decision.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

5. The website is easy to use.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

6. The search result was fast enough to keep me engaged.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

7. I will revisit the website.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

8. I will recommend this website to others.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

Additional input:

What feature did you like most about the website?

……
……
……
……

What feature did you like least about the website?

……
……
……
……
……
……
……

**3. <u>QA Test Plan</u>**

Objective: The objective of this test plan is to find, analyze and categorize bugs and technical issues related to search function of our product's home page. With our focus on user centric design in developing our website, the searching on our website should be self-guiding to users of various level of knowledge on how to use inter to search, it should process search input and display result in an easy-to-read format. Our website allows visitors to search listing without having to register. Users can search in two

ways: simple search and advanced search. With the two options for search, we hope that it will match the level of effort users want to put in searching homes listed in our database. Advanced option will include various search criteria such as build year, number of bathroom, price range etc. With simple search option, users will be able to search within specified distance range from a location specified using zip code or city name, state and address. The correct entry to distance range will be integer ranging from 1-300 and to the location will be city, state and country or simply 5-digit zip code. This test will focus on simple search option.

**Testing Environment:**

The following set up will be required to perform the test.

- Laptop or desktop with mouse and pad or touch screen to assist entering data in the search area.
- Internet access.
- Data server at SFSUSE/fa1g07 running
- Individuals: To enter the data to search listing, and to record result and monitor the computer and surrounding.
- The context can vary from public to private such as library, public place, residence or public vehicle as long as there is internet connection, and the time of the day can be any time.
- The tester will NOT instruct on how to enter search criteria but will have the home page of the website ready in the computer browser and delegate the control of computer to the user.
- Browsers: two latest version of each of the browsers listed:
    - Microsoft Edge
    - Chrome
    - Firefox Mozilla

**Features to be tested:**

Users will enter distance in miles in the left search box and address in the right search box before clicking search button. we will use equivalence partitioning of numbers for distance mileage, along with variation of characters as input for range. For address input, we will use various combination of identification and characters such as zip code, street name and city etc. If users successfully enter the required information, the search result will display with basic information along with picture and google maps along with marked areas of search in the map. If users are unsuccessful to enter criteria as expected, our website will alert with proper guide on what is needed in the specific field.

Test Case:

Google Chrome: Pass
Microsoft Edge: Pass
Mozilla Firefox: Pass

4. **CODE REVIEW**

**Code style:**

To suit the objective of the project and design pattern we have used following style to code in this project.

- Validation of input
- Encryption of sensitive data such as password for security
- Variable are declared in relation to its role
- Proper indentation for better visibility
- Comments for maintainability.
- Higher modularity for scalability and dependency and for easier testing for continuous software development.
- Concise and simple for less complicacy

The following code was fewviewed to Vikram Grewal by Dillon Brosnan.

**Dillon Brosnan <dillonbrosnan11@gmail.com>** 8:52 PM (8 hours ago)

to Vikram, Indra

post.js
-vikram greqal

```
var express = require('express');
var router = express.Router();
var bcrypt = require('bcrypt');
var uuidv4 = require('uuid/v4');
var moment = require('moment');
var fileUpload = require('express-fileupload');
var PostModel = require('../models/postModel');

// Route
router.get('/',function(req,res){
  console.log(req.session.role);
  console.log(req.session.agentId);
  if(req.session.role != "agent") {
    res.redirect('../../login/agent');
  } else  {
  res.render('post');
  }
});

router.post('/', function(req,res){

  var beds = req.body.beds;
  var baths = req.body.baths;
```

```
  var sqFt = req.body.sqFt;
  var lotSqFt = req.body.lotSqFt;
  var yearBuilt = req.body.yearBuilt;
  var hoa = req.body.hoa;
  var lotType = req.body.lotType;
  var price = req.body.price;
  var lat = req.body.lat;
  var lng = req.body.lng;
  var formattedAddress = req.body.formattedAddress;
  var description = req.body.description;
  var currentTimestamp = moment().unix();
  var datePosted = moment(currentTimestamp*1000).format("YYYY-
MM-DD HH:mm:ss");
  var saleId = uuidv4({msecs: new Date().getTime()});
  var imageId = uuidv4({msecs: new Date().getTime()});
  var agentId = req.session.sessionId;

  beds = Number(beds);
  baths = Number(baths);
  sqFt = Number(sqFt);
  lotSqFt = Number(lotSqFt);
  yearBuilt = Number(yearBuilt);
  hoa = Number(hoa);
  price = Number(price);
  lat = Number(lat);
  lng = Number(lng);


  // All form validation is include below using express-
validator
  req.checkBody('lat', 'Latitude must be between -90 and
90').notEmpty().isFloat({ min: -90, max: 90 });
  req.checkBody('lng', 'Longitude must be between -180 and
180').notEmpty().isFloat({ min: -180, max: 180 });
  req.checkBody('beds', 'Beds must be between 0 and
20').notEmpty().isInt({ min: 0, max: 20 });
  req.checkBody('baths', 'Baths must be between 0 and
20').notEmpty().isFloat({ min: 0, max: 20 });
  req.checkBody('sqFt', 'Square feet must be between 0 and
1000000').notEmpty().isInt({ min: 0, max: 1000000 });
  req.checkBody('lotSqFt', 'Lot square feet must be between 0
and 1000000').notEmpty().isInt({ min: 0, max: 1000000 });
  req.checkBody('yearBuilt', 'Year built must be between 0 and '
+ new Date().getFullYear())
      .notEmpty().isInt({ min: 0, max: new Date().getFullYear()
});
```

```
  req.checkBody('hoa', 'Hoa must be between 0 and
10000').notEmpty().isInt({ min: 0, max: 10000 });
  req.checkBody('description', 'Description must be between 0
and 255 characters').isLength({ min: 0, max: 255 });
  req.checkBody('price', 'Price must between 0 and
1000000000').notEmpty().isInt({ min: 0, max: 1000000000 });
  var errors = req.validationErrors();
//VERY WELL ORGAIIZED









  if(errors)  {
    var response = { errors: [] };
    errors.forEach(function(err) {
      response.errors.push(err.msg);
    });
    res.statusCode = 400;
    return res.json(response);
  }
  else if(req.files.saleImage.length > 12 || !req.files) {
      res.send("File wrong");
  }
  else  {
    req.files.saleImage.mv('public/images/' + imageId + '.jpg',
function(err)  {
      if(err) {
        console.log(err);
        res.send(err);
      }
    });
    PostModel.checkFormattedAddress(formattedAddress)
    .then(function() {
      return Promise.all([PostModel.insertImage(saleId,
imageId)]);
    })
    .then(function() {
      return Promise.all([PostModel.insertPosting(beds, baths,
sqFt, lotSqFt, yearBuilt,
      hoa, lotType, price, lat, lng, formattedAddress, saleId,
datePosted, description, agentId)]);
    })
    .then(function() {
      return res.redirect('../../forSale/' + saleId + '/')
```

```
    })
    .catch(function(err) {
      console.log(err);
      res.status(400);
      return res.send("Couldn't post posting");
    });
  }

});


module.exports = router;
```

/*HELLO VIKRAM. JUST WANTED TO TAKE SOME TIME TO REVIEW YOUR CODE. YOU DID A GOOD JOB AT ORGANIZING ALL OF THE VARIABLES INTO RELATED GROUPS, AS WELL AS USING CAMEL CASING. THERE IS A NICE FLOW TO THE WORK, AND IS EASILY READABLE. THERE IS NOT TOO MUCH OF A WORKLOAD ON POST.JS AS SOME OF THE WORK IS HANDLED BY POSTMODEL.JS. I WAS WONDERING HOWEVER IF IT WAS NECESSARY TO SEND THE ENTIRE MESSAGE OR SHOULD WE HAVE A CUSTOMIZED MESSAGE FOR SPESIFIC ERROR. THANK YOU VERY MUCH. OVERALL EXCELLENT WORK*/

-db

5. **Self-check on best practice**

Major assets protected are passwords, user-Ids and agent-Ids. We can confirm that passwords are encrypted using bcrypt, an express module. Input data below is validated using express validator:

Registering as user/agent:

- -First Name
- -Last Name
- -Email
- -Username
- -Password
- -Address

Posting

- -Address
- -Beds
- -Baths
- -Square Feet
- -Lot Square Feet
- -Price

-HOA fees

## 6. Self-check on non-functional specs

1. Application shall be developed and deployed using class provided deployment stack. DONE
2. 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
3. Application shall be hosted and deployed on Amazon Web Services as specified in the class. DONE
4. 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
5. Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed. ON TRACK
6. Data shall be stored in the MySQL database on the class server in the team's account. DONE
7. Application shall provide real-estate images and optionally video. DONE
8. Maps showing real-estate location shall be required. DONE
9. Application shall be deployed from the team's account on AWS. DONE
10. No more than 50 concurrent users shall be accessing the application at any time DONE
11. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. DONE
12. The language used shall be English. DONE
13. Application shall be very easy to use and intuitive. No prior training shall be required to use the website. ON TRACK
14. Google analytics shall be added. ON TRACK
15. Messaging between users shall be done only by class approved methods and not via e-mail clients in order to avoid issues of security with e-mail services. ON TRACK
16. Pay functionality (how to pay for goods and services) shall not be implemented. DONE
17. Site security: basic best practices shall be applied (as covered in the class). DONE
18. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. DONE

**19.** 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). ON TRACK.