Software Engineering CSC648/848 Spring 2019

BetterHome

Spring 2019

Team 43

Taso Grigoriou (team lead - agrigori@mail.sfsu.edu)
Henok Kassegn (frontend lead)
Sawyer Nixon (backend lead)
Cole (Michael) Tormey (github master)
Austin Sy-Velasco (backend / DB manager)
Liwang Gao (frontend)

Milestone 4

05/09/2019

History Table (revisions):

Revisions	**Date**
M4V1	05/12/2019
M3	04/26/2019
M2V2	04/19/2019
M2V1	04/02/2019
M1V2	03/20/2019
M1V1	03/17/2019

Table of Contents:

1. Product Summary:

Our product named "BetterHome" will be the leading property marketplace dedicated to providing consumers with services they need to buy, sell, or rent the place they call home. BetterHome will provide many services to users of all kinds, including landlord hosts, apartment / real-estate searchers and resellers. BetterHome will give property owners the ability to host their property on the website, and provide all general and customized information needed to make their property an attractive candidate for someone's next home. The website will also provide this same service to "house-flippers," or resellers, and will allow them to browse all listings with filters, all while being able to host a property up on the website and maintain each aspect simultaneously. What makes BetterHome a unique product is our incredibly detailed advanced search which allows users to browse the various property listings on the website with feature rich and detailed customization of search options. This includes accessibility filters such as laundry and wheelchair access, and hospital / BART proximity that other property marketplaces don't possess.

URL:

https://better-home-234220.appspot.com

Priority 1 Committed Functions:

For Admins:
(Admin username: **admin**
Admin password: **better-home-admin**)

1. Admins are able to update and maintain user information in the case where users forget their password or if there is inappropriate content on their profile.
2. Admins are able to delete user accounts who host properties (landlords), as well as users who search for property listings (clients).
3. Admins are able to lock accounts for both landlord and clients, as well as unlock their accounts.
4. Admins are able to find registered landlord and clients on the users page by their username, full name, email address, etc.
5. Admins are able to view a gallery collection of property listing images on the images page.

6. Admins are able to delete images on the images page.
7. Admins are able to access username and email for all users on the users page.

For Landlords:

1. Landlords are able to browse and experience all regular features of the website without having to login.
2. Soon to be registered landlords are able to register on the registration page.
3. Registered landlords are able to post their own property with all the information about their property.
4. Registered landlords are able to update the location of their hosted property.
5. Registered landlords are able to update the price of their hosted property.
6. Registered landlords are able to update the city and zipcode of their hosted property.
7. Registered landlords are able to select if their property is a house or apartment.
8. Registered landlords are able to select if their property is wheelchair accessible.
9. Registered landlords are able to login and logout at any and all times.
10. Registered landlords are able to add photos of their newly created listings, as well as adding new photos to their already hosted listing.
11. Registered landlords are able to delete photos of their already hosted property.

For Clients:

1. Soon to be clients are able to create new accounts.
2. Registered clients are able to login and logout whenever they see fit.
3. Registered clients are be able to modify all information on their profile, including email address, phone number, etc.
4. Registered clients are able to add properties they view to their list of favorite properties.
5. Registered clients are able to revisit any and all of their favorite properties.
6. All clients are able to browse the website and search for property listings without having to login.
7. All clients are able to view listings that are currently for sale.
8. All clients are able to see listings that have been recently bought or rented out.
9. A basic search and advanced search box is displayed on the home page to all clients.
10. All clients are able to search property listings by the city they want to view.
11. All clients are able to search listings by a minimum and maximum price range.
12. All clients are able to search listings by the type of property, including houses, apartments, and condos.
13. All clients are able to change the search options they last entered on the search results page.

2. Usability test plan:

Software usability testing is a key methodology that ensures applications are intuitive and easy to use for the target audience. Usability testing has direct benefits for companies as usability improvements often are fundamental to the success of a product. A standard usability test includes the following five steps: obtain suitable participants, design test scripts, conduct usability sessions, interpret test outcomes, and produce recommendations.The main objective of Usability Testing is to identifies usability errors in our BetterHome website in its early development cycle and to save our website from failure. For our BetterHome website, we will test the usability of the Advanced Search functionality. The objective of the Advanced Search usability testing include establishing a baseline of user performance in using the Advanced Search and identifying potential design concerns to be addressed in order to improve the efficiency, productivity, and end-user satisfaction in the Advanced Search functionality.

Test objectives:

- To verify that BetterHome website is intuitive to the average user that would be likely to search for Apartments.
- To check how the navigation back and forward in a search impacted the result.
- Establish baseline user performance and user-satisfaction levels of the user interface and search functionality for future usability evaluations.
- To check the total number of results that displayed on the search result page

Problem statement:

Is the Advanced Search function of BetterHome website easy to use?

Potential sources of error may include:

- Advanced Search result errors – failure to display the correct search result and if a keyword is typed incorrectly, then the relevant result message should be displayed

- Presentation errors – failure to locate and properly act upon desired information in the search result screens.

Test Description and System setup:

The participants' responsibilities will be to attempt to complete a set of representative task scenarios presented to them in as efficient and timely manner as possible, and to provide feedback regarding the usability and acceptability of the search functionality. The participants will be directed to provide honest opinions regarding the usability of the website search usability, and to participate in post-session subjective questionnaires and debriefing.

Participants will take part in the usability test at San Francisco State University, Business building Room number 217. A personal laptop with the BetterHome Website will be used in a typical school class environment to test the Advanced Search functionality. The participant's interaction with the Website will be monitored by the facilitator seated in the classroom. Note takers and data logger(s) will monitor the sessions.

The facilitator will brief the participants on the Web site and instruct the participant that they are evaluating the website. Participants will sign an informed consent that acknowledges: the participation is voluntary, that participation can cease at any time. The facilitator will ask the participant if they have any questions. After each task, the participant will complete the post-task questionnaire and elaborate on the task session with the facilitator. After all task scenarios are attempted, the participant will complete the post-test satisfaction questionnaire.

Hardware and software setup:

- Hardware Setup – Website on Google Cloud Platform running on Linux Machine
- Software Setup – BetterHome homepage (logged out) available on Chrome 72 browser of Ubuntu 16.04.5. The MySQL database currently has 7 property listings.

URL: https://better-home-234220.appspot.com/#/

Legal issues: This test is voluntary, no identifying information will be kept, only experience with website. Any personal information entered upon registration will be deleted at the completion of the project.

Report: Will contain information based on how easy the users found it to navigate and operate BetterHome

Usability Task Table:

Task	**% Completed**	**Error**	**Comments**
Search Apartments	100	None	None
Search Apartment by Listing Type	100	None	None
Search apartment by City	100	None	None
Search apartment by Bedrooms	100	None	None
Advanced Search box appear on mobile or Tablet	100	None	None
Advanced Search option exclude for not available Apartments	100	None	None

Task:

Find apartment for rent in San francisco that has two bedroom using the Advanced Search.

Task	Search apartment in San Francisco
Machine State	Home page of BetterHome https://better-home-234220.appspot.com
Success Criteria	Three Search results displayed

Questionnaire:

	Agree									Disagree
The Advanced Search feature was easy to be found	O	O	O	O	O	O	O	O	O	O
You find the Advanced Search interface intuitive.	O	O	O	O	O	O	O	O	O	O
The Advanced Search process was quick and simple.	O	O	O	O	O	O	O	O	O	O
The Advanced Search option didn't ask for many information.	O	O	O	O	O	O	O	O	O	O
It was clear how to select the search options.	O	O	O	O	O	O	O	O	O	O

Scenarios for testing Advanced Search functionalities:

1. Advanced Search results displayed should be relevant to search keyword
2. % sign in search keyword should not redirect to 404 ERROR
3. Application should not crash if user inserted % in search field
4. When user start typing word in text box it should suggest words that matches typed keyword
5. There should be pre-defined search criteria for auto complete e.g. after typing first 3 letter it should suggest matching keyword
6. When user clicks on any link from result and navigates back, then result should be maintained
7. After clicking Search field - search history should be displayed (latest search keyword)
8. Search results should be cleared on clicking clear search button
9. History displayed in search field should be relevant to logged in user only
10. Total number of search records/results should be displayed on page
11. Search keyword should suggest similar kind of properties
12. For Advanced Search - limited search filters should be provided
13. Validate search rules defined for "Exact Match" with the search key word
14. User should be able to search when he/she enters the keyword and hits 'Enter' button on keyboard

3. QA test plan:

Test objectives:

- Purpose: Quality Assurance testing is performed to validate that system functionalities are per the requirements. Testing is performed to validate that all functionalities work according the specifications mentioned in previous phases.

- Problem statement: To verify that BetterHome's 'Advanced Search' feature returns proper listings according to the parameters specified.

Test plan:

The user will follow the test cases below to search for listings that satisfy certain characteristics of the property. They will complete this search without logging in or registering for an account.

- Hardware and software setup:

  Hardware Setup – Website on Google Cloud Platform running on Linux Machine
  Software Setup – BetterHome homepage (logged out) available on Chrome 72 browser of Ubuntu 16.04.5. The MySQL database currently has 7 property listings.

Features to be tested:

Listings can be returned by entering at least the city in which to search, and can be further specified to include available amenities in the options located in the Advanced Search bar. The search functionality is to be tested through the 3 test cases below:

1. Search for a listing in San Francisco with access to Bart using the search bar
2. Search for a listing in San Francisco with wheelchair access and onsite laundry access using the search bar
3. Search for a listing in Daly City using the search bar

Test Cases:

Test #	Description	Test Input	Expected Output	Pass/Fail
1	Search for a listing in San Francisco with access to Bart using the search bar	Select 'Advanced Search'. Select Accessibility - 'BART' Enter 'San Francisco' into search by city	Property Listings page appears and displays the one (1) property that matches these parameters.	PASS
2	Search for a listing in San Francisco with wheelchair access and onsite laundry access using the search bar	Select 'Advanced Search'. Select Accessibility - 'laundry' Enter 'San Francisco' into search by city	Property Listings page appears and displays the three (3) properties that match these parameters.	PASS
3	Search for a listing in Oakland using the search bar	Enter 'Daly City' into search by city	A popup is displayed that reads: "Unable to retrieve any listing based on your search and filter options. Please try again"	PASS

4. Code Review:

Code Style of HTML:

1. Use lowercase for element names
2. Use lowercase letters in attribute names
3. Use double quote ("") around attribute values
4. Close all elements
5. Do not add blank lines without a reason
6. Provide alternative contents for multimedia
7. Remove spaces around equal signs
8. Two spaces of indentation instead of using tab key
9. When line-wrapping, each continuation line should be indented at least 4 additional spaces from the original line.
10. Use a new line for every element

Code Style of CSS:

1. Use shorthand properties where possible
2. Do not use units after 0 values unless they are required
3. Alphabetize declarations
4. Use a semicolon after every declaration
5. Undent all block content
6. Separate rules by new lines and put a blank line between rules.

Code Style of Typescript:

1. Use camelCase for variable and function names
2. Use PascalCase for class names
3. Use camelCase of class members and methods
4. Use PascalCase for type name.
5. Use camelCase for type members
6. Use single quotes (') unless escaping
7. Use 2 spaces for indentation. Not tabs
8. Space before type
9. Use semicolons
10. Annotate arrays as foos: Foo[] instead of foos:Array<Foo>.
11. Name files with camelCase
12. Put else on a separate line from the closing curly brace
13. Put the opening curly braces on the same line.
14. Do suffix a service class name with Service.
15. Name test specification files with a suffix of .spec

Code Style of Javascript:

1. Declare variable with let or const (no vars)
2. Always use semicolons
3. For name: functionNamesLikeThis, variableNamesLikeThis,ClassNamesLikeThis, EnumNamesLikeThis, methodNamesLikeThis,CONSTANT_VALUES_LIKE_THIS, foo.namespaceNamesLikeThis.bar, andfilenameslikethis.js.
4. Use Array and Object literals instead of Array and Object constructors.
5. Follow C++ formatting rules (https://google.github.io/styleguide/cppguide.html#Formatting)
6. Use parentheses only where required
7. Strings should use ‘ over “ but either is acceptable
8. Encouraged, use JSDoc annotations @private and @protected

Code Review from Team 6 (Reviewer - Huy Nguyen):

Tuesday, May 7, 2019, 4:58 PM -0700 from Liwang Gao <lgao2@mail.sfsu.edu>:

Hi Huy,

The below is a piece of the front-end code for the search function which is written in typescript, please take a look and give us some feedback/code review. Thank you!

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {Router} from "@angular/router";
import {ListingSearch, SearchListingsService} from "../core/services/search.listings.service";
import {Listing} from "../core/services/listings.service";
import {MatDialog} from "@angular/material";
import {RegisterDialog} from "../register/register.dialog";
import {FormControl} from "@angular/forms";

@Component({
  selector: 'app-advanced-search',
  templateUrl: './advanced-search.component.html',
  styleUrls: ['./advanced-search.component.css']
})
export class AdvancedSearchComponent implements OnInit, OnDestroy {

  listingSearch: ListingSearch;
  listings: Listing[];

  accessibilities = new FormControl();
  accessibilityList: string[] = ['Laundry', 'Hospital', 'Wheelchair', 'BART'];

  isLoaded = true;
```

```
constructor(
  private router: Router,
  private searchService: SearchListingsService,
  public dialog: MatDialog
) {}

ngOnInit() {
  if (localStorage.getItem('listingSearch')) {
    this.listingSearch = JSON.parse(localStorage.getItem('listingSearch'));
  }
  else {
    this.listingSearch = {
      city: ''
    }
  }
}

ngOnDestroy() {
  if (this.listingSearch) {
    localStorage.setItem('listingSearch', JSON.stringify(this.listingSearch));
  }
}

onAccessibilityChange() {
  this.listingSearch.accessibilities = this.accessibilities.value;
}

onSearchClick() {
  if (!this.listingSearch.city.length) {
    this.openDialog('Please enter some text for the city field');
  }
  else {
    this.isLoaded = false;
    this.searchService.getSearchListings(this.listingSearch)
      .subscribe(listings => {
          this.isLoaded = true;
          this.searchService.saveSearchListings(listings);
          this.router.navigate(['/properties']);
        },
        err => {
          this.isLoaded = true;
          this.openDialog('Unable to retrieve any listing based on your search. Please try again');
```

```
        });
    }
  }

  numberOnly(event): boolean {
    const charCode = (event.which) ? event.which : event.keyCode;
    return !(charCode > 31 && (charCode < 48 || charCode > 57));
  }

  openDialog(message: string, subscribe: boolean = false) {
    const dialog = this.dialog.open(RegisterDialog, {
      width: '250px',
      data: {
        message: message
      }
    });
    if (subscribe) {
      dialog.afterClosed().subscribe(result => {
        this.router.navigate(['/properties']);
      });
    }
  }
}
```

Huy Nguyen <huy9997@gmail.com> Today, 2:55 PM Liwang Gao

Hi Liwang, I did the basic code review below. It was good for the most part.
I think their is nothing wrong with your code as far as I can see. I only only work on making this more moduler.
That means reach function such as openDialog should be its own file. I think that would make it cleaner.
Then you connect them all to a index.js
overall good code though
Example:
Folder Search:
file openDialog
file numberOnly
file onSearchClick
file onAccessibilityChange
file ngOnDestroy
file ngOnInit
file index.js

5. Self-check on best practices for security:

Major Assets to Protect:

1. Passwords

Confirmation of Password encryption:

Passwords are encrypted using the bcrypt library upon registration of the user. This password is validated using the bcrypt compare function upon logging in. Here is an example of the encoded passwords in the database.

userId	username	password	firstName	lastName
6	tasogrigoriou	$2b$10$3/vnZWw492PymyxMs..R0.jDS/sN3hrt...	Anastasios	Grigoriou
10	anotheruser	$2b$10$EIl6GtBhy6G6H1CGsNxeH./HwBe57u/y...	uuu	oooo
14	iamaustinsy	$2b$10$mWmhehhVhpxDUiB8Yq5JluP/IC.xnhu/l...	Austin	Sy-Velasco
22	jortizco	$2b$10$PvjlEZ5aIiprIk60xegwr.KaF.RuYv3wRm...	Jose	Ortiz
34	tasog	$2b$10$Sig9/dXa/KCkO9TZPpxHPenIYP55Hnr6...	Anastasios	Grigoriou
37	stephCurry	$2b$10$VQgt7egpMOvdL9xYqyhwj.5jMffgMvW...	Steph	Curry

Confirm Input Data Validation:

In order to validate all inputs we use escaping on all data that comes from the front end. By tokenizing all input we prevent malicious sql injection through our api. Below is a snippet as an example from our delete image function.

```
180  // Delete imageURL for a given Listing
181  router.delete( path: '/image/:imageUrl', handlers: function (req, res) {
182      let sql = `DELETE FROM ListingImage WHERE imageUrl = ` + database.escape(req.params.imageUrl);
183      database.query(sql, function (err, result) {
184          if (err) {
185              res.status( code: err.status || 500).send(err.message);
186          } else {
187              res.send(result);
188          }
189      });
190  });
```

6. Self-Check: Adherence to original Non-Functional Specs:

Security:

1. Login shall be required for Clients and Admins. **DONE**
2. Username shall be the Client's email. **DONE**
3. Password shall be encrypted when stored. **DONE**
4. Client's session shall end upon leaving the site. **ISSUE - WE CACHE USER DATA**
5. Client's session shall only end by code design. **ISSUE - WE CACHE USER DATA**

Performance:

1. Loading time for site shall be less than 3 seconds for any screen. **DONE**

Capacity:

1. The total data storage allowed by the web site shall not exceed of 80 % of the server capacity for this site. **DONE**
2. The web site shall be prepared to support scalability for adding future new features. **DONE**
3. The web site shall be capable to handle at least 50 Clients simultaneously. **DONE**

Reliability:

1. Downtime for maintenance shall be less than 3 hours per month. **DONE**
2. Downtime for maintenance shall not affect the main functionality of the site. **DONE**
3. In all cases, downtime for maintenance shall be informed to the Client through email. **DONE**

Recovery:

1. In a total failure case, the whole site should be put down to revision. **DONE**
2. If broken, the mean time to recovery shall not exceed one day. **DONE**

Data Integrity:

1. Database tables shall be backed up every day. **DONE**
2. Administrator shall be able to execute a recovery when needed. **DONE**
3. Image Sizes shall be limited up to 1 megabyte. **DONE**
4. Images shall be uploaded in correct format (jpg, jpeg, or pdf) to the server. **DONE**
5. Links to images on the server shall be uploaded to the database. **DONE**

Compatibility:

1. The site shall be compatible with the last version of Microsoft Edge browser (44.17763). **DONE**
2. The site shall be compatible with the last version of Safari browser (12). **DONE**
3. The site shall be compatible with the last version of Firefox browser (65.0.2). **DONE**

4. The site shall be compatible with the last version of Chrome browser (7.3). **DONE**
5. Third party applications shall not be able to modify any content that may affect the site Compatibility. **DONE**
6. The site shall be ready to support with any or minimal changes any other compatibility that may be added in future versions. **DONE**
7. The site should be compatible to escalate to new relational databases. **DONE**

Conformance with Coding Standards:

1. Architecture and design standards shall meet all the requirements listed under the High Level Architecture section of M1 document. **DONE**
2. Only working code that meets all the code standards shall be submitted to the project Repository. **DONE**
3. Any working code shall be tested and debugged before being considered working code. **DONE**
4. Any internal errors or exceptions returned by the code shall be stored in a log. **DONE**
5. Any error that may affect the functionality of the site shall be reported to the Client. **DONE**
6. Any error shall be handled in a way that does not affect the functionality of the site. **DONE**
7. The whole production cycle of this site shall be finished 2 weeks before the delivery date. **ON TRACK**
8. This site shall not be launched without all the priority one features completed and tested. **ON TRACK**

Look and Feel Standards:

1. The application and its layouts shall look professional. **DONE**
2. The site shall be simple enough to handle by all the parties involved. **DONE**
3. Elements on screen shall have the correct density to meet the compatibility standard of the browsers. **DONE**
4. Elements on screen shall have rich and beautiful colors for Client delight. **DONE**
5. The site shall be able to work correctly without mouse interaction. **ON TRACK**
6. The site shall be able to work correctively without keyboard interaction. **ON TRACK**
7. Elements in screen shall be resized automatically without Client interaction when being loaded in all the different platforms supported by the site. **DONE**

Internationalization / Localization Requirements:

1. Default language shall be English. **DONE**
2. The site shall support scalability to add other languages. **ON TRACK**
4. Any copyrighted material shall be immediately be taken down upon reception of an official DMCA takedown request. **DONE**

Website Policies:

1. A link to the policies of this site shall be always visible in all its pages to be accessible by all the parties. **ON TRACK**
2. Clients' data shall not be sold to third parties. **DONE**
3. Clients and Landlords data that do not add any functionality to the system shall not be Collected. **DONE**
4. Clients that post inappropriate listings(false listings/ copyright images/ sexual images) shall have their postings taken down. **DONE**