

# Software Engineering CSC 648/848 Section 02 Spring 2019



## Milestone 4

Team 08

May 8th, 2019

Cheryl Fong

[cfong8@mail.sfsu.edu](mailto:cfong8@mail.sfsu.edu)

Jerry Wong

Chirag Agarwal

Harold Pedroso

Johanna Hadgu

Sharan Lao

Ermias Haile

REVISION TABLE		
Date	Version Number	Description
05/08/19	1.0	Date submitted for review
05/19/19	1.1	Revised after feedback

# 1. PRODUCT SUMMARY

- 1.1** Our product GatorRent is an accommodation-search application that is intended for the use of the students of San Francisco State University (SFSU) to find housing and for landlords that are looking for renters alike. This application is important as on-campus housing at SFSU is extremely limited and expensive thus making it difficult for both domestic students and international students to find accommodation within their budget and preference.

The GatorRent application will allow a user (users of the GatorRent application includes landlords looking for renters and students looking for housing) to either post houses, apartments, rooms that they are renting out, or for the user to filter through the currently available listings posted on the application. Our application will allow users to find listings based on important factors to consider as a student of San Francisco State University such as distance from the university, rent price. Thus, making GatorRent uniquely tailored for students and landlords as compared to other applications currently on the market and we will market it through societies at San Francisco State University to local students and to incoming international students.

GatorRent is accessible through the following link: <http://54.183.44.255/>

## 1.2 Committed Function List

### 1.2.1 Users can:

- 1.2.1.1 Login or register themselves a user of the website.
- 1.2.1.2 Easily search and browse for listings.
- 1.2.1.3 Post a place for rent.
- 1.2.1.4 Filter their search results by price, rent-type, postcode, if pets are allowed and if the rent-type is shared.
- 1.2.1.5 View additional details about any listing, including a map displaying the location of the listing.
- 1.2.1.6 Message any landlord about a specific listing and view all received messages on their dashboard.
- 1.2.1.7 Contact the Admin regarding any concerns through the Contact Us page.
- 1.2.1.8 **Unique feature:** users can filter listings respective to their distance from SFSU

- 1.2.2 We will ensure the proper safety precautions when users are visiting and using this site. Passwords will be encrypted.**

## 2. USABILITY TEST PLAN

### 2.1 Test Objectives:

The main purpose of this usability test is to review the effectiveness, efficiency, and user satisfaction of our site. It is designed to observe and interpret the user's interactions with our product, assess its strengths and weaknesses, and allow us to take the proper steps to improve the user experience for future use. In this test plan, the main function being tested and analyzed for usability is the search function.

The effectiveness of the site will be measured by the user's ability to correctly retrieve the needed results. For this specific test plan, the user will be provided with a task description that consists of three simple tasks to complete. If the user is able to successfully find the correct listings, based on the parameters of the task, the site is then deemed effective.

The efficiency of the site will be quantitatively measured by the length of time it takes the user to complete the task, in relation to accuracy and completeness. It will be influenced by the user's ability to adjust to the layout and search function provided on the site.

And lastly, the satisfaction of the site will be studied by the user's comfort and acceptability while completing a series of given tasks. Once the user has completed a task, they will be required to complete a questionnaire provided at the end of this section. This allows us to properly assess if our product is satisfying our targeted users.

### 2.2 Test Description:

#### 2.2.1 System Startup & URL to be Tested

The system startup requires a computer, either a laptop or desktop, with access to the internet. The device must contain either a Windows 7 and up, Mac OS or Linux operating system and the type of browser to be utilized is either Chrome (version 74.0.3729.131) or Safari (version 12.1).

The URL of the system to be tested is found on <http://54.183.44.255/results> with the objective of searching for a listing. In the test plan, it is already assumed that the provided URL to be tested has already been loaded and that the site already consists of several posted listings.

#### 2.2.2 Starting Point

The starting point will be a Chrome (version 74.0.3729.131) or Safari (version 12.1) web browser, set to the provided URL of the system being tested, <http://54.183.44.255/results>.

### 2.2.3 Intended Users

While this site is open to anyone, the intended users for this usability test plan are current or prospective SFSU students that are looking to find or provide a place to rent. They are average consumers, with some to little familiarity with computers and smartphones. More specifically, they are first-time users, naturally unregistered, who are not familiar with the layout, implementation, or effectiveness of the site, therefore allowing us to retrieve unbiased and accurate feedback on the usability of our search function.

### 2.2.4 What is Measured

Overall, we will measure the effectiveness, efficiency, and user satisfaction. However, for this specific test plan, we will mainly focus on user satisfaction. After asking users to perform a series of tasks, we will require the user to complete a Lickert scale on their experience with completing the tasks. Although the measure of satisfaction is subjective, our Lickert scale questions will allow us to acquire measurable and productive feedback on our site

## 2.3 Usability Task Description

Before filling out the Likert questionnaire, the task testers will have to complete the following 3 tasks:

1. From the webpage, search and find a house.
2. From the webpage, search and find a room.
3. From the webpage, search and find an apartment for less than \$700 per month.

TASK	MACHINE STATE
Task	Perform a search on GatorRent
Machine State	GatorRent webpage successfully loaded
Successful Completion Criteria	Listings that satisfy the search are successfully found
Benchmark	1 minute & 30 seconds

## 2.4 Questionnaire

### GatorRent Usability Questionnaire

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. The user interface for searching on this website was straightforward and simple to use	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2. It was easy to filter search results by property type (i.e. room, apartment, house)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3. The process of filtering by price is easy and intuitive	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Optional: Your comments are greatly appreciated

SUBMIT

### 3. QA TEST PLAN

#### 3.1 Test Objectives

3.1.1 The objective of this test is to check if the search bar is providing the correctly desired results.

#### 3.2 HW and SW Setup

3.2.1 The hardware and software setup consist of a laptop with a Chrome browser (74.0.3729.131) with <http://54.183.44.255/results> opened.

#### 3.3 Feature to be Tested

3.3.1 The feature to be tested in this plan is the search bar.

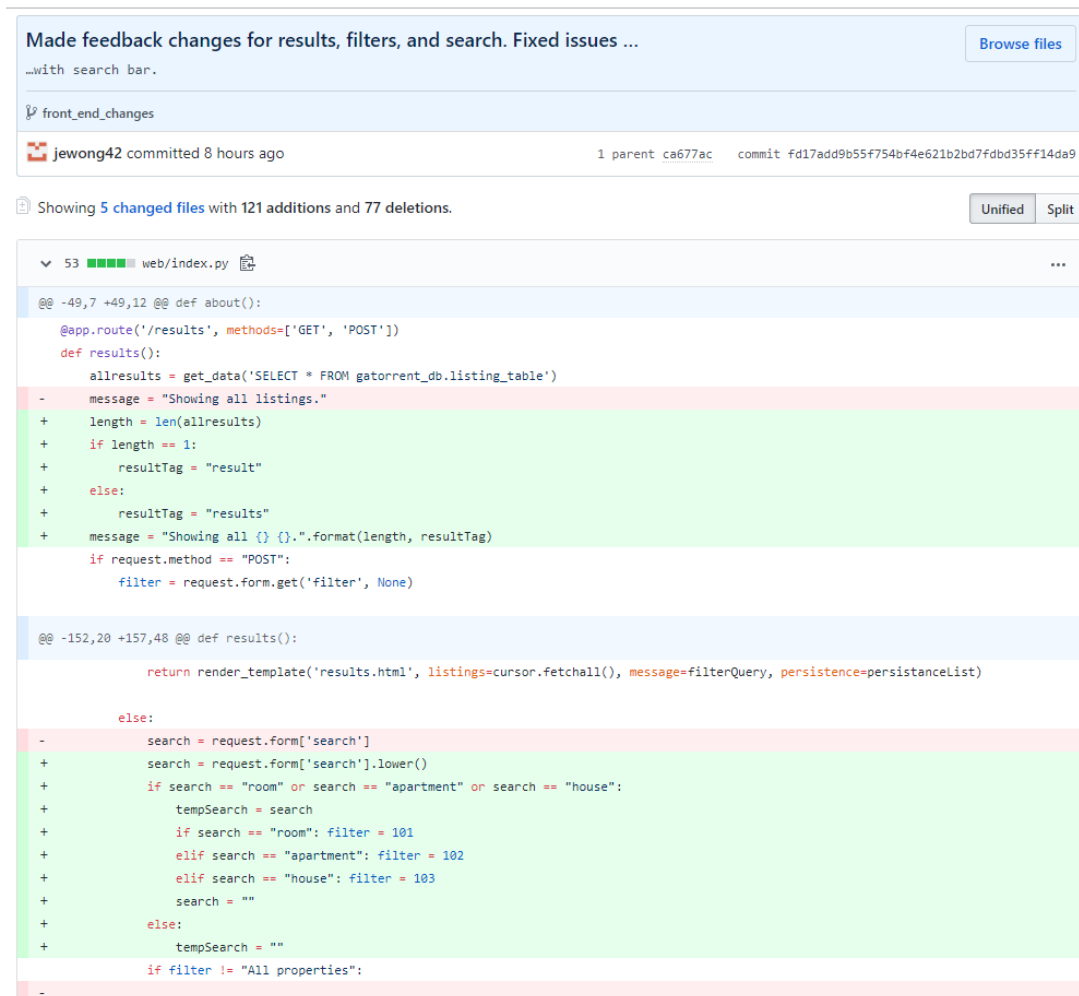
TEST #	TEST TITLE	TEST DESCRIPTION	TEST INPUT	EXPECTED CORRECT OUTPUT	TEST RESULTS	
					CHROME 74.0.3729.131	SAFARI 12.1
1	Search Bar Property	Open the webpage: <a href="http://54.183.44.255/results">http://54.183.44.255/results</a> Change the dropdown search bar located near the top of the page to "All properties". In the search bar type "apartment". Click the blue "Search" button.	Search bar drop down filter: All properties  Search bar: "apartment"	3 Listings that fall under the "apartment" category  <b>3 apartment results</b>	FAIL	FAIL
2	Search Bar Location	Open the webpage: <a href="http://54.183.44.255/results">http://54.183.44.255/results</a> Change the dropdown search bar located near the top of the page to "All properties". In the search bar type "San". Click the blue "Search" button.	Search bar drop down filter: All properties  Search: "San"	5 Listings that fall under the category "san"  <b>5 listings results [3 apartments, 1 house, 1 room]</b>	PASS	PASS
3	Search Bar Validation	Open the webpage: <a href="http://54.183.44.255/results">http://54.183.44.255/results</a> Change the dropdown search bar located near the top of the page to "All properties". In the search bar type "12344?".	Search bar drop down filter: All properties  Search: "12344?"	Alert: "Search must not contain special characters"	FAIL	FAIL

		Click the blue "Search" button.		and must be less than 40 characters."		
--	--	---------------------------------	--	---------------------------------------	--	--

## 4. CODE REVIEW

### 4.1 Coding Style

- 4.1.1 As part of using [VSCode](#) or [Atom](#) for development, [Pylint](#) is the suggested default code analysis and quality check. [Python Coding Style](#), written and published by Mozilla is a good summary of coding style best practices being followed right now. This is the complete Style Guide as specified by [Python.org](#), developers are expected to glanced through and try to enforce rules as stated in this guide.
- 4.1.2 Code related to the feature used in the QA and usability test:



Made feedback changes for results, filters, and search. Fixed issues ... [Browse files](#)

...with search bar.

front\_end\_changes

jewong42 committed 8 hours ago 1 parent ca677ac commit fd17add9b55f754bf4e621b2bd7fdbd35ff14da9

Showing 5 changed files with 121 additions and 77 deletions. [Unified](#) [Split](#)

```
53 web/index.py
@@ -49,7 +49,12 @@ def about():
    @app.route('/results', methods=['GET', 'POST'])
    def results():
        allresults = get_data('SELECT * FROM gatorrent_db.listing_table')
-       message = "Showing all listings."
+       length = len(allresults)
+       if length == 1:
+           resultTag = "result"
+       else:
+           resultTag = "results"
+       message = "Showing all {} {}.".format(length, resultTag)

        if request.method == "POST":
            filter = request.form.get('filter', None)

@@ -152,20 +157,48 @@ def results():

        return render_template('results.html', listings=cursor.fetchall(), message=filterQuery, persistence=persistencelist)

    else:
-       search = request.form['search']
+       search = request.form['search'].lower()
+       if search == "room" or search == "apartment" or search == "house":
+           tempSearch = search
+           if search == "room": filter = 101
+           elif search == "apartment": filter = 102
+           elif search == "house": filter = 103
+           search = ""
+       else:
+           tempSearch = ""
+       if filter != "All properties":
```



```

searchresults = get_data("SELECT * FROM gatorrent_db.listing_table WHERE CONCAT_WS(' ', street, city, state, country, postcode) L
-
- filterToString = get_data("SELECT DISTINCT type FROM gatorrent_db.rent_type_table WHERE rent_type_id = {}".format(filter))
- message = "Showing all results for '{}' in {}".format(search, filterToString)
+ filterName = get_data("SELECT DISTINCT type FROM gatorrent_db.rent_type_table WHERE rent_type_id = {}".format(filter))[0]
+ filterToString = filterName[0]
+ length = len(searchresults)
+ if length == 1:
+     resultTag = "result"
+ else:
+     resultTag = "results"
+ if search == "":
+     message = "Showing {} {} in {}".format(length, resultTag, filterToString)
+     search = tempSearch
+ elif search == "house" or search == "room" or search == "apartment":
+     message = "test"
+ else:
+     message = "Showing {} {} for '{}' in {}".format(length, resultTag, search, filterToString)
+ if tempSearch != "":
+     search = tempSearch
return render_template('results.html', listings=searchresults, filter=filter, search=search, message=message, persistence=None)
else:
-
searchresults = get_data("SELECT * FROM gatorrent_db.listing_table WHERE CONCAT_WS(' ', street, city, state, country, postcode) L
-
- message = "Showing all results for '{}'".format(search)
- return render_template('results.html', listings=searchresults, message=message, persistence=None)
+ length = len(searchresults)
+ if length == 1:
+     resultTag = "result"
+ else:
+     resultTag = "results"
+ if search == "":
+     message = "Showing all {} {}".format(length, resultTag)
+ else:
+     message = "Showing {} {} for '{}'".format(length, resultTag, search)
+ if tempSearch != "":
+     search = tempSearch
+ return render_template('results.html', listings=searchresults, message=message, search=search, persistence=None)
return render_template('results.html', listings=allresults, message=message, persistence=None)

# Auth pages

```

CSC-648-SFSU / csc648-sp19-team08
Watch 2
Star 0
Fork 1

Code
Issues 15
Pull requests 4
Projects 1
Wiki
Insights

## Search Bar Bug Fix #92

Open
johannah10 wants to merge 9 commits into master from front\_end\_changes

Conversation 1
Commits 9
Checks 0
Files changed 16
+396 -158

johannah10 commented 3 days ago
Great work @jewong42 and to the rest of the front end team! After testing it locally, it resolved the following issues mentioned in #82

- should not take symbols
- min should not be greater than max in Prices (perhaps use a toggle)
- add persistence of value in the search bar

jewong42 added some commits 3 days ago

- Made feedback changes and added header comments to register and login. c10a24a
- Provided a temporary fix to missing dotenv issue. To be reverted later. 5329aee
- Made feedback changes, added header comments, and implemented variabl. e83228f

New changes since you last viewed
View changes

jewong42 added some commits 3 days ago

- Made feedback changes and implemented variables for dashboard. Fixed ca677ac
- Made feedback changes for results, filters, and search. Fixed issues fd17add
- Added multi-image upload to post form. 8db5265

johannah10 commented 3 days ago
Note: The search bar still allowed me to enter nonalphanumeric symbols, such as @, #, +. This needs to be resolved before the demo

Reviewers
cherylfong
At least 1 approving review is required to merge this pull request.

Assignees
No one—assign yourself

Labels
bug
front-end

Projects
None yet

Milestone
Milestone 5 & Demo

Notifications
Unsubscribe
You're receiving notifications because you authored the thread.

5 participants

cherylfong changed the title Front-end-changes Search Bar Bug Fix 2 days ago
Lock conversation

Sharan144 added this to the Milestone 5 milestone 2 days ago

Sharan144 added bug front-end labels 2 days ago

haropedro and others added some commits 2 days ago

- configuration files 700de4e
- fix teh settings files db 7c1e75e
- form\_post.html's all bug has been fixed 890bb3f

cherylfong self-requested a review 6 hours ago

johannah10 commented 2 minutes ago
Also, just to maintain consistency, make sure to have the proper header comments. Otherwise, it can get to be a bit confusing when we are going through the code, especially for the back end.

Add more commits by pushing to the front\_end\_changes branch on CSC-648-SFSU/csc648-sp19-team08.

Review required
At least 1 approving review is required by reviewers with write access. Learn more.
Show all reviewers

This branch has conflicts that must be resolved
Use the web editor or the command line to resolve conflicts.
Resolve conflicts

Conflicting files
web/templates/includes/form\_register.html

Merge pull request
You can also open this in GitHub Desktop or view command line instructions.

Write
Preview
AA B i

## 5. SELF-CHECK ON BEST PRACTICES FOR SECURITY

### 5.1 Major Assets We're Protecting

- 5.1.1 The major assets we are protecting are relative to the user's personal information. User Confidentiality is a vital element that is being taken into account during the implementation of this site, therefore passwords are protected and encrypted via the [Bcrypt library](#). Passwords are hashed before being stored on the database.
- 5.1.2 To protect our site from spam, automated accesses and extractions of data from websites, we are using [reCAPTCHA](#). This form of captcha provides protection while also letting valid users navigate through our website with ease.
- 5.1.3 To secure our images, we used the `secure_filename()` function, as recommended by the **Flask Documentation Release 0.9 Dev**, to secure a filename before we stored it.
- 5.1.4 While we currently haven't done so, we plan to validate our search bar input. To provide protection against potential SQL injection, form inputs are validated via [Bootstrap's Form Validation](#). Functions that handle SQL and user input have taken into account the vulnerabilities, as stated in [W3Schools](#).

```
form_filters.html
6  -->
7
8  <div class="container py-3">
9    <h3 class="text-center">Filters</h3>
10   <form class="form-group" method="post" action="{{ url_for('results') }}">
11     <ul class="list-group py-3">
12       <li class="list-group-item justify-content-between">
13         <!-- Price range -->
14         <!-- Required data: 'minPrice', 'maxPrice' -->
15         <div class="container">
16           <h6>Price range</h6>
17           <div class="form-inline py-2 mx-2 justify-content-between">
18             <input type="number" class="form-control" name="minprice" placeholder="min" style="width:40%"
19               {% if minPrice %} placeholder={{minPrice}} {% endif %}>
20             <small class="mx-2"></small>
21             <input type="number" class="form-control" name="maxprice" placeholder="max" style="width:40%"
22               {% if maxPrice %} placeholder={{maxPrice}} {% endif %}>
23           </div>
24         </div>
25       </li>
26       <li class="list-group-item justify-content-between">
27         <!-- Location -->
28         <!-- Required data: 'city', 'zipcode' -->
```

This is an example of how filter inputs are validated. In line 21, `type="number"` specifies that only numbers can be typed into the field.

```

1  <!-- Login Form for the Website -->
2  <form class="w-25" data-toggle="validator" role="form" method="post">
3    <h3>Register for GatorRent</h3>
4    <div class="form-group my-4">
5      <div class="form-group">
6        <label>Username <small class="text-muted">(Maximum 30 characters)</small></label>
7        <input type="text" id="username" maxlength="30" class="form-control" placeholder="Enter username" required autofocus>
8      </div>
9      <div class="form-group">
10       <label>Password <small class="text-muted">(Maximum 15 characters)</small></label>
11       <input type="text" id="inputPassword" maxlength="15" class="form-control" placeholder="Enter password" required>
12     </div>
13     <div class="form-group">
14       <label>Confirm Password</label>
15       <input type="text" id="inputPasswordConf" maxlength="15" class="form-control" data-match="#inputPassword" placeholder="Enter password" required>
16     </div>
17     <div class="form-group">
18       <label>Email</label>
19       <input type="email" id="email" class="form-control" placeholder="Enter email" required >
20     </div>
21     <div class="form-check">
22       <input type="checkbox" class="form-check-input" id="agree">

```

In the image above, line 12 restricts the user from having a password that is more than 15 characters. This requirement is set so that an SQL injection of more than 15 characters will not work.

## 6. SELF-CHECK: ADHERENCE TO ORIGINAL NON-FUNCTIONAL SPECS

#	REQUIREMENT	STATUS
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).	ON TRACK
2.	Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers.	ON TRACK
3.	Selected application functions must render well on mobile devices.	ON TRACK
4.	Data shall be stored in the team's chosen database technology on the team's deployment server.	ON TRACK
5.	No more than 50 concurrent users shall be accessing the application at any time	DONE
6.	Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	ON TRACK
7.	The language used shall be English.	DONE
8.	Application shall be very easy to use and intuitive.	ON TRACK
9.	Google analytics shall be added.	ON TRACK
10.	No email clients shall be allowed.	DONE
11.	Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated.	DONE
12.	Site security: basic best practices shall be applied (as covered in the class)	ON TRACK
13.	Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator.	ON TRACK

14.	Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.	ON TRACK
15.	<b>The website shall <u>prominently</u> display the following <u>exact</u> text on all pages <i>"SFSU Software Engineering Project CSC 648-848, Spring 2019. For Demonstration Only"</i> at the top of the WWW page. (Important so as to not confuse this with a real application).</b>	ON TRACK