SW Engineering CSC648/848 Spring 2019

GatorList

Team – 04

Karuna Nayak (knayak@mail.sfsu.edu)

Victor Muñoz

Huawei Gao

Dylan Shwan

Daniel Mossaband

Gabriel Alfaro

Aditya Sheoran

Milestone 4

Date Created: May 2nd, 2019

Date Submitted: May 7th, 2019

Date Revised: May 13th, 2019

1. Product Summary

Product Name: GatorList

Below are the features offered by GatorList.

- A search engine which allows users to make quick searches based on any part of the listing information such as street name, zip code, apartment type and area name.
- Filters to allow users to get listing filtered by type, price range, number of bedrooms and bathrooms.
- Sort feature which allows the sorting the listings by posting date, distance to campus, commute time to campus, price from low - high and high - low.
- The details of the listing with the map of area where it is located when the user clicks on any listing.
- Feature to contact the landlord if user wishes.
- Admin features to approve/disapprove the listings before posted on website.

GatorList, a web application that offers an easy and simple way to connect with a renter or rentee that is tailored specifically to San Francisco State University students. It is the single best tool for finding a suitable place to live while attending SFSU since it provides the features like distance to SFSU campus and commute time to SFSU.

Product URL: http://18.224.109.221/

2. Usability test plan

Test objectives: The function that is being tested for the site is the filter bar function. The filter bar consists of four different filter. Each of these filter bars are used to define a specific action that take place to have better outcomes. The main of objective of the tester will determine the effectiveness and efficiency of the filter bar. The first filter that will be tested is the “Type”. The “Type” filter allows users to narrow their search to a specific property. Depending on what the tester wants to rent, they can select from three different choices, which are rooms, house or apartments. The next filter is the price filter, this filter is one of the most important one because it will allow users to set a price range and the tester will need to set a positive value number into the given space to see the functionality of the filter. The last two filters are the bedrooms and bathrooms. These two filters will further narrow the search result and allow the tester to find the perfect choice of property rent.

Test description:

- System setup: In order to begin the test of the filter bar function. The tester will need a device first, which has internet access. The tester will then need to open a working browser either google chrome or safari. This browser will allow the tester to test the product and especially the filter bar function.
- Starting point: Once the system has been set up, the tester will have the opportunity to visit the site and test the function that is being tested, which is on the homepage of Gatorlist website.
- Who are the intended users: The intended users of this function are anyone who has a specific type of place they want to rent out. These are users who have made a choice a certain type of a property.
- URL of the system to be tested: The URL of the system that is being tested is http://18.224.109.221/
- What is to be measured:

The test evaluation will allow the effectiveness of site.

Test/Use Case	Completed	Errors	Average time spent/Clicks	Comments
Find filter bar				
Select type				
Set price range				
Select # of bedrooms				
Select # of bathrooms				

- UsabilityTask description:

Task	Description
Task	Narrow search results using filter bar on price and bedroom and bathroom
Machine state	Results are filtered
Successful completion criteria	Search results are narrowed
Benchmark	Completed in 1 sec

- Questionnaire:

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	Comment
The filter bar function was easy to navigate.						
Selecting the type was simple and easy.						
Adding price range was simple						

3. QA test plan

- Test objectives: The objective of filter bar function QA test is to check the functionality of the rolls given to each filter and meet the specs planned by the team members. The test will also confirm that the results given after selecting and inputting data are accurate and find any bugs that related to filter bar function.
- HW and SW setup:

Hardware setup: Computer used Windows Hp, MacBook Air
Software setup: Google chrome 74.0.3729.131, Safari 12.1

- Feature to be tested: The feature that is being tested is the functionality of the filter bar on the homepage of GatorList.

List of following things that needs to be tested.

1. The type filter shows the matching listing as the selected filter: for example, selecting a apartment will show only the apartment on the homepage.
2. The test of Price filter allows the tester to set min and max price range, allowing a better result.
3. The test of Bedrooms filter should increase the number of bedrooms a user is interested to rent. For example, selecting 2+ bedroom will show only show listing with 2 or more bedrooms.
4. The last test is for the filter bathrooms, the results will be similar to bedrooms. The user can select the number of bathrooms the property should have.

- QA Test plan:

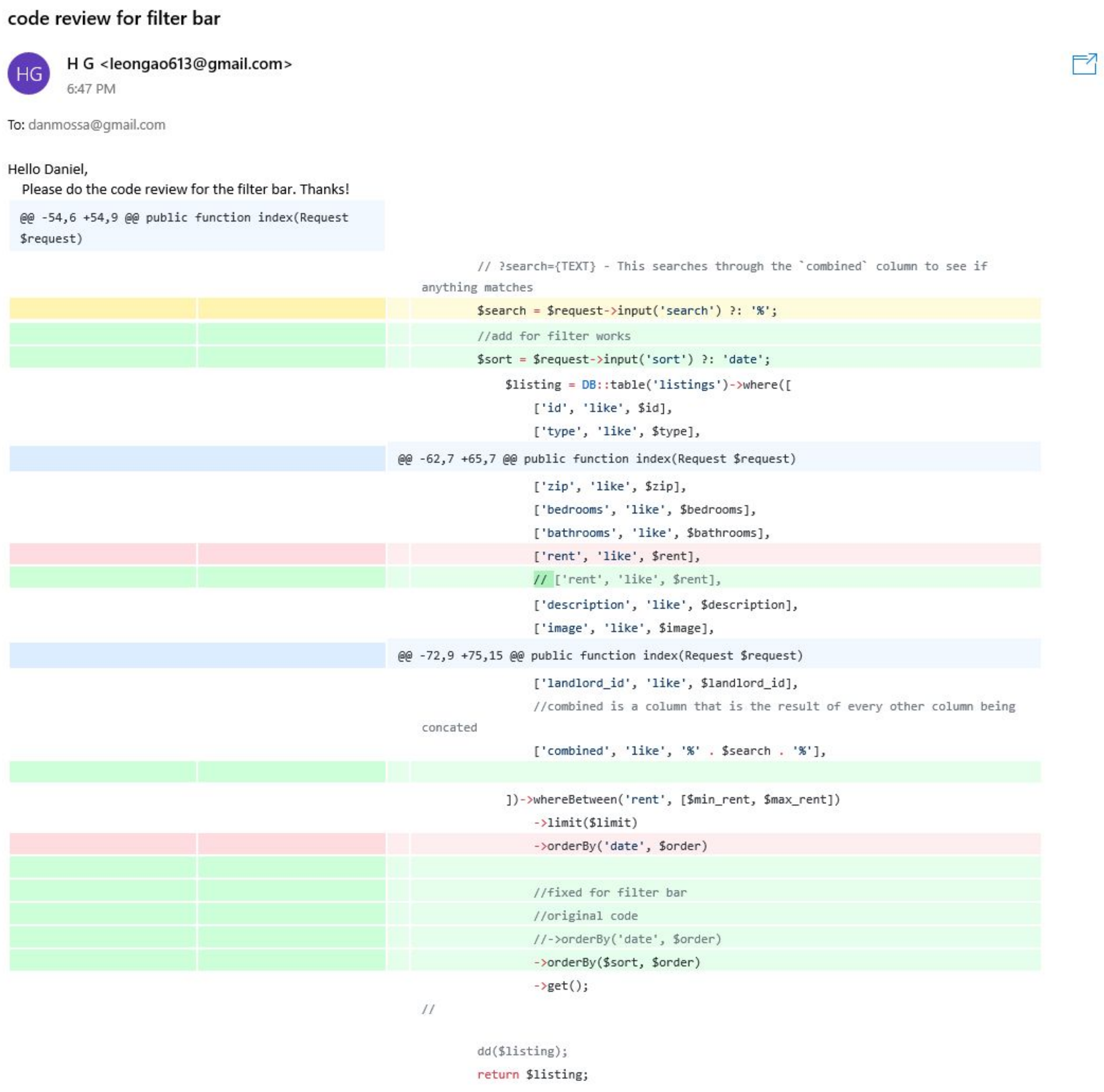
Number	Description	Test Input	Expected Output	Result (Chrome)	Result (Safari)
1	Filter Type of Listing	Select Filter Option “Room”	Display four results with the header “Room”	PASS	PASS
2	Filter Price Range	Min Option “500” Max Option “1500”	Display four results that are within the price range of $500 to $1500	PASS	PASS
3	Filter Number of Bedrooms	Select Filter Option “2+”	Displays seven results that have two or more bedrooms”	PASS	PASS
4	Filter Number of Bathrooms	Select Filter Option “3+”	Displays seven results that have three or more bathrooms	PASS	PASS

B) The test works on both Chrome and Safari browsers. Please refer above table.

4. Code Review

a) Code style must follow PSR-1 and PSR-2. Everything string-like that's not public-facing should use camelCase.

b) Changed code by team member:



code review for filter bar

H G <leongao613@gmail.com>
6:47 PM

To: danmossa@gmail.com

Hello Daniel,
Please do the code review for the filter bar. Thanks!

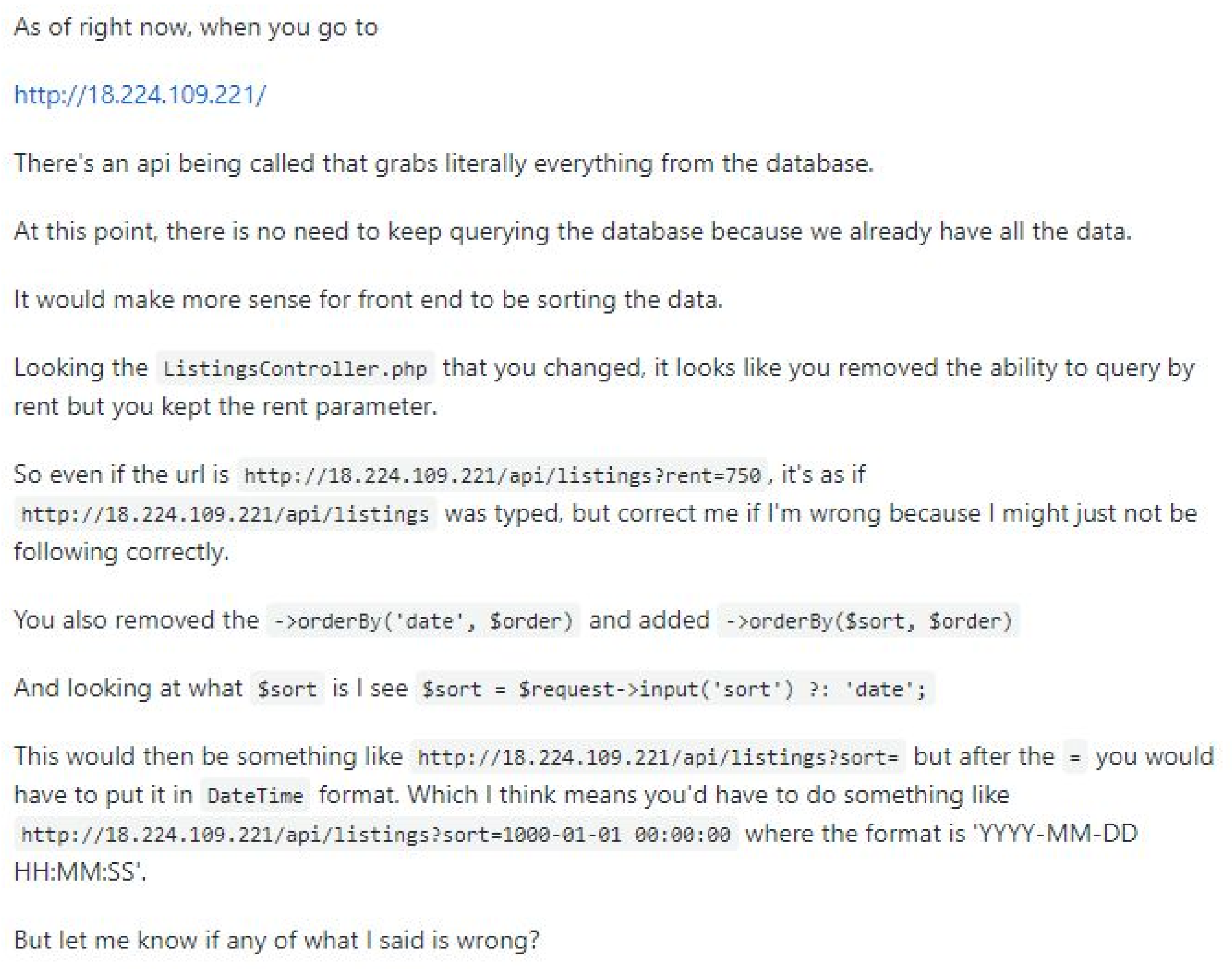
```
@@ -54,6 +54,9 @@ public function index(Request $request)
        // ?search={TEXT} - This searches through the `combined` column to see if
anything matches
        $search = $request->input('search') ?: '%';
        //add for filter works
        $sort = $request->input('sort') ?: 'date';
            $listing = DB::table('listings')->where([
                ['id', 'like', $id],
                ['type', 'like', $type],
@@ -62,7 +65,7 @@ public function index(Request $request)
                ['zip', 'like', $zip],
                ['bedrooms', 'like', $bedrooms],
                ['bathrooms', 'like', $bathrooms],
                ['rent', 'like', $rent],
                // ['rent', 'like', $rent],
                ['description', 'like', $description],
                ['image', 'like', $image],
@@ -72,9 +75,15 @@ public function index(Request $request)
                ['landlord_id', 'like', $landlord_id],
                //combined is a column that is the result of every other column being
concated
                ['combined', 'like', '%' . $search . '%'],

            ])->whereBetween('rent', [$min_rent, $max_rent])
                ->limit($limit)
                ->orderBy('date', $order)

                //fixed for filter bar
                //original code
                //->orderBy('date', $order)
                ->orderBy($sort, $order)
                ->get();
//

        dd($listing);
        return $listing;
```

Code review:



As of right now, when you go to

http://18.224.109.221/

There's an api being called that grabs literally everything from the database.

At this point, there is no need to keep querying the database because we already have all the data.

It would make more sense for front end to be sorting the data.

Looking the `ListingsController.php` that you changed, it looks like you removed the ability to query by rent but you kept the rent parameter.

So even if the url is `http://18.224.109.221/api/listings?rent=750`, it's as if `http://18.224.109.221/api/listings` was typed, but correct me if I'm wrong because I might just not be following correctly.

You also removed the `->orderBy('date', $order)` and added `->orderBy($sort, $order)`

And looking at what `$sort` is I see `$sort = $request->input('sort') ?: 'date';`

This would then be something like `http://18.224.109.221/api/listings?sort=` but after the `=` you would have to put it in `DateTime` format. Which I think means you'd have to do something like `http://18.224.109.221/api/listings?sort=1000-01-01 00:00:00` where the format is 'YYYY-MM-DD HH:MM:SS'.

But let me know if any of what I said is wrong?

5. Self-check on best practices for security

- ☐ List major assets you are protecting : Passwords and images
- ☐ Confirm that you encrypt PW in the DB : Yes, while registering user, password will be hashed and salted.
- ☐ Confirm Input data validation (list what is being validated and what code you used) – we request you validate search bar input) : We have two main forms of input validation, when creating a listing and when retrieving rows from our database. When the user creates a listing, there is client-side validation to make sure that the user is unable to create a listing without filling out the correct information.
  search field is protected with up to 40 alphanumeric characters as a valid input.

When retrieving a row from the database via the url, we are using Laravel's Eloquent ORM which prepares the statement and then parameterizes the queries.

We can also protect database so only specific IP addresses can connect to mysql.

6. Self-check: Adherence to original Non-functional specs

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). - **DONE**
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers - **DONE**
3. Selected application functions must render well on mobile devices - **DONE**
4. Data shall be stored in the team's chosen database technology on the team's deployment server.- **DONE**
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. - **DONE**
7. The language used shall be English. - **DONE**
8. Application shall be very easy to use and intuitive. - **DONE**
9. Google analytics shall be added – **IN PROGRESS**
10. No e-mail 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) - **DONE**
13. Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator – **IN PROGRESS**
14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development - **DONE**
15. The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2019. For Demonstration Only" at the top of the WWW page. (Important so as to not confuse this with a real application). - **DONE**