SW Engineering CSC648/848 Spring 2019

Gatorbnb

Team 5

Peter Le - team lead (ple2@mail.sfsu.edu) local
Jay Sunga - front end lead
Wesley Goldfisher - back end lead
Franky
Mehi Ledwon
Tanya Wong
Anthony Owyeong

5/7/19

Milestone 4

Date	Description
5/4/19	Draft Created
5/7/19	Submitted Version
5/15	Frozen Version

1. Project Summary

- Name of product: Gatorbnb
- Url: http://13.57.19.213/
- Committed Functions
    - User registration - an unregistered user shall be able to sign up and create an account
    - User Login - a registered user shall be able to log into their registered account
    - Browsing - all users shall be able to browse through the current listings
    - Filtering - all users shall be able to filter the search results by House Type, number of Bedrooms, number of Bathrooms, Price and Distance from SFSU with a drop list
    - Search- all users shall be able to use free text search for zipcode and city
    - Post - a registered user shall be able to Post a new listing, while an unregistered user may not
    - Message - a registered user shall be able to Message the user who created a listing they're interested in
    - Map - from the selected listing, all users shall be able to view a map of the housing will be available
    - Search - all users shall be able to search for a listing in the search bar
    - Admin- administrators shall be able to deny or approve listing and can delete users from the database

2. Usability Test Plan

Test Objectives:

The selected function being tested is the search function. Gatorbnb relies heavily on the search function because it allows one of our many intended audiences, SFSU students, to be able to search for a place to live. By utilizing the search function, every user whether they're registered or unregistered are able to find listings based on the information they've entered within the search bar. The search bar searches the parameters of a city or zip code and sends them to a listings page. Since the search function is being relied upon, it is important to ensure it is functioning correctly to create a greater user experience.

Users who have difficulties using the search function on Gatorbnb will be unsatisfied, equating to a lost in profit in a real-world situation. So in order to create a well functioning website that satisfies every user, the main function being used (search) should be the function being tested to prevent any difficulties or problems.

Test Description:

Url: http://13.57.19.213/

System Setup:

The system is setup where testers are asked to use our search function. Testers will be instructed to use a computer, regardless if it's Windows or Linux, that has access to the internet. They may choose to use either Google Chrome or Safari as the browser.

Starting Point:

The starting point would be the home page of the website. After their initial search, they will be brought to another page which lists the listings based on their search results. After that, one more search will be given, to not only test the search function of the landing page, but also the listing page's search function.

Intended Users:

The intended users for the test are people who use a computer for an adequate amount of time, between the ages of 18-45. This will give us feedback from many different perspectives on whether the search function is easy to use or if it was difficult.

What's Measured:

For this test, the level of difficulty of using the search function is being tested. Testers should have a clear understanding of what searching is and how to search. What is also tested is if testers input the correct searchable arguments and whether invalid searches are handled correctly. Lastly responsiveness is also being tested. Testers will be able to observe whether they got listings based on their arguments and how long it took for them to receive their results.

Usability Task Description:

Testers are given the task to search for listings within San Francisco. They may either search for all listings within San Francisco by typing “San Francisco” in the search bar, or by a zip code within San Francisco for a more specific result.

Questionnaire:

Please, check one

	Strongly Disagree	Disagree	Neither agree or disagree	Agree	Strongly Agree
1) The parameters of search was clearly stated					
2) Search was easy to use and understand					
3) My search results were correct					

Comments:

3. QA Test Plan

Test Objectives:

- To ensure the search function operates correctly where the searched for listings are being displayed
- Results are obtained whether the tester were able to accomplish the given instructions by marking pass or fail.

Hardware and Software Setup

1. Testers will access a computer connected to the internet and use either Google Chrome or Safari
2. Testers will go to http://13.57.19.213/
3. Url: http://13.57.19.213/

Feature to be tested:

- The Search function will be the feature to be tested since it's one of the most utilized features. By typing a city or a zip code, listings will be displayed based on their input.

QA Test Plan

QA Test Number	QA Test Title (search by:)	QA Test Description	QA Test Input	QA Expected Output	QA Results Chrome	QA Results Safari
1	City	Type San Francisco on the search bar	San Francisco	Get 15 results of listings that are located in San Francisco	Pass	Pass
2	Zip Code	Type 94122 on the search bar	94122	Get 2 listings with the zip code of 94122	Pass	Pass
3	Case sensitivity	Type SaN fRanCiSco on the search bar	SaN fRanCiSco	Get all 15 results of listings in San Francisco	Pass	Pass
4	Whitespace	Type sanfrancisco in the search bar	sanfrancisco	Return the message “No matching results. Please check your spelling or enter a valid ZIP code”	Pass	Pass

4. Code Review

a.

From: ple2@mail.sfsu.edu
To: aownyeong@mail.sfsu.edu

Hello Anthony,

Thanks for letting me review your code, here are my thoughts.

For the variable naming convention, it is camel case just like how you did. The style of the coding is to separate the subfunctions if it is possible to pull out the make it as modular as possible. Separates the files by the functionality of the functions within in and by the routes / object that it is related to.

- Peter

b. Copy and paste email:

To: ple@mail.sfsu.edu
From :aownyeong@mail.sfsu.edu

Hello Peter, please Review my code.

```
/**
 * get function for search in listings route
 * @params queue | the search queue to search db
*/
router.get('/search/', (req, res) => {
 let { queue } = req.query
 const { type, beds, baths, priceMax, distanceMax } = req.query


 console.log( queue, type, beds, baths, priceMax, distanceMax )
```

```
            queue = queue.replace("+", " ")
           // create queue for database call
            let dbQueue = null
            if( !isNaN(queue) ) {
              dbQueue = `SELECT * FROM listings WHERE zipcode=$1`
            } else {
              dbQueue = `SELECT * FROM listings WHERE
LOWER(city)=LOWER($1)`
            }
           // database call for the search results
            db.any(dbQueue, [queue])
            .then(data => {
              console.log(data)
           // filter
              // @TODO move this function to module
              const filteredListings = data.filter(listing => {
                console.log(beds + " " + listing.bedroom)
                if( ( type && listing.housing_type !== type) )
                  console.log('invalid type')
                else if( ( beds && listing.bedroom < beds) )
                  console.log('invalid bedroom')
                else if( ( baths && listing.bathroom < baths) )
                  console.log('invalid bathroom')
                else if( ( priceMax && listing.price > priceMax) )
                  console.log('invalid price')
                else if( ( distanceMax && listing.distance > distanceMax) )
                  console.log('invalid distance')
                else
                  return listing
```

```
      })
      console.log('fitered stuffs here')
      console.log(filteredListings)

      console.log( "LENGTH: ", filteredListings.length)

      res.send(filteredListings)
     })
     .catch(error => {
      console.log(error)
      res.sendStatus(204)
     })
    })
```
Comments: In line comments

function is robust.

A bit long, so need to pull out functions to make it more readable.

5. Self-check on best practices for security

- For our application, the major assets we are protecting is the name, email, and the user's password. Personal information like user data is stored solely in the users table in the database and the password specifically will be encrypted to protect this personal information from being stolen and then exploited for malicious use. Images are stored in a file system with hashed path. Also the database cannot be controlled without the routes we provide.
- All the passwords in the database are hashed into a 64-bit char using the bcrypt password hashing function. We are encrypting the password in the database with 8 rounds of salt. This ensures the protection of the passwords in the users database against rainbow tables and someone trying to access password data maliciously.
- For input validation, we require that nothing over 40 alphanumeric only characters will be accepted in the search input validator. The code we used for the search bar input is as follows:

```
<Input
   style={inputStyle}
   size='large'
   action={{ icon: 'search' }}
   name="search"
   placeholder='Enter a city or ZIP code'
   value={this.props.queue}
   onChange={this.props.changeQueue}
   maxLength="40"
/>

getListings = (value) => {
    Var regex = /^[a-z\d-_\s]+$\
    if(regex.test(value)) {
        axios.get(`/api/listing/search/${value}`)
    }
}
```

6. Self-check on Non-functional specs

a. 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**
b. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers **DONE**
c. Selected application functions must render well on mobile devices **DONE**
d. Data shall be stored in the team's chosen database technology on the team's deployment server. **DONE**
e. No more than 50 concurrent users shall be accessing the application at any time **ON TRACK**
f. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **DONE**
g. The language used shall be English. **DONE**
h. Application shall be very easy to use and intuitive. **DONE**
i. Google analytics shall be added. **ON TRACK**
j. No e-mail clients shall be allowed. **DONE**

k. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. **DONE**
l. Site security: basic best practices shall be applied (as covered in the class) **DONE**
m. Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator **ON TRACK**
n. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. **DONE**
o. The website shall <u>prominently</u> display the following <u>exact</u> 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). **ON TRACK**