SW Engineering CSC 648/848 Summer 2019

Milestone 4: BaySpace

08/2/2019

Team 5

Team Lead: Jonathan Kaldani

Github Master: Brian Lai

Front End Team:

- Anya Livshyts
- Tianchen Liu
- Habtom Asfaha

Back End Team:

- Anwar Halteh (Back End Team Lead)
- Justin Zhu
- Sandeep Dhakal

Submission History Table

Initial Submission: 08/06/2019
Revised: 08/09/2019

BaySpace

1) Product summary

Who is this product for? The target audience for this application are people from any age range, and all walks of life. Any person or group that wants to know what is going on in a nearby open-space, park or beach - this is for you.

What are the product's basic details? BaySpace is a neat, sleek, information packed home for all things environmental. Made for outdoorsy people, by outdoorsy people, we know exactly what you worry about when going outside. With BaySpace, planning your outdoor adventure is easy.

Where would someone use this product? At home, outside, in your car, at work; anywhere in the Bay Area - we got you.

When should someone use the product? Want to check on a previous oil spill? Make sure the park you are taking your kids to has no problems? That the beach near you has no recent reports?

Why is this product useful or better than its competitors? Sleek, fast, and info-packed. We bring you the environmental scoop as soon as it is reported, and update as soon as it is resolved. Everything you need in one place, all you need to do is check.

How does the product work? Let's say you walk past your favorite park, and spot a busted fire hydrant. Water everywhere, flooding the playground. You log onto BaySpace, and with a few quick clicks, the incident is reported. Now Lucy, who was about to take her kids to that park for lunch, knows that she needs to be careful. When she checks BaySpace a few hours later, she sees that the incident has been resolved by the DPW.

BaySpace- community driven information, presented on a sleek and easy to use platform.

- Find us here: http://ec2-54-215-173-150.us-west-1.compute.amazonaws.com/

List of Functions

1. Search problems in parks and open spaces

2. Browse problems fro park locations

3. Images of parks

4. Map of park locations

5. Login/register

6. Post problems

7. Admin change the status of the aforementioned problems

8. Captcha for registration and posting problems - for safety measure

9. List of current posts

10. Number of current posts for each park

2) Usability test plan – 2 pages

Test Objectives:

The objective of this test is to illustrate how the search function interfaces usable our current environmental issue in the search box and finding the result that is looking by the users. Our test on this function is that we want the users how fast they are able to find the search box on our home page and how easily they can search their environmental issues in the search box as soon as they start typing their concerns or using the issue category. Also how we want to test how understandable the error message is when the users try to find irrelevant search or incorrectly typing during their search.

Test Plan:

A. System Setup:

Computer: MacBook Pro

Browser: Google Chrome Browser V.75

Operating system: macOS Mojave 10.4

B. Starting point:

the Home Page of BaySpace

C. Intended User:

Any user could be registered or unregistered who wants to know the environmental issue.

D. Usability Task description:

Find parks with fires.

Completion Criteria:

The user typed fire in the search box and clicking the search button to find the result in the database and successfully found that fire issue was in Alcatraz Island.

The user has completed this task within 1 minute.

E. Url to Test System:

http://ec2-54-215-173-150.us-west-1.compute.amazonaws.com/

localhost:3000

- **Lickert subjective test:**

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I found the search bar is easy to use. (Check one)					
I found the search category match my needs. (Check one)					
I found the search result useful to me. (Check one)					

Please leave any additional comments here:

3) QA test plan - max 2 pages

3. QA Test Plan

3.1 Test Objectives

The objective of the QA test is to thoroughly test the search function of BaySpace in order to ensure that the search results display issues relevant to the search query. Searching by issue name, location name and percent like(name of the park or description) will be tested.

3.2 System Setup

Processor: Intel(R) Core(TM) i5-7300u cpu @ 2.60ghz

RAM: 16 GB

Operating System: Windows 10 Professional 64-bit Laptop

Browser 1: Google Chrome Version Version 75.0.3770.142

Browser 2: Microsoft Edge 42.17134.10

3.3 Feature to be Tested

This QA test will test the functionality of the search function from all of BaySpace major pages: home page, search result page, park detail page, collectively hereinafter “Testing Pages.” The test will test for the following Expected Functionality:

- The search result page shall show all issues that match the search query (the search query being either an issue name, location name, and description), as well as the number of found issues.

- All results should be readable and openable.

- The search query shall not disappear from the search box after searching.

- All issues shall include an Issue Type, Park Name, Rating, Description, Author and Status.

- All results shall be the same across both browsers.

3.4 Test Cases

3.4.1 Test 1 – Search with Location drop down menu.

Select Alcatraz Island on the Location drop down menu and click search. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

3.4.2 Test 2 – Searching with Issue drop down

Select Ebola on the drop down of All Issue and click search. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

3.4.3 Test 3 – Searching with percent like

Enter Angel and click search. Validate that every Expected Functionality is met. If all of the Expected Functionality is met, the test result shall be PASS. Perform the search on both browsers.

1. Google Chrome

Test #	Test Title	Test description	Test input	Expected correct output	Test result
1	Search with Location	Test if the search work with Location drop down menu	Select Alcatraz Island on the drop down of All Location and enter search	Check if there are 3 results; Are they all related to Alcatraz	PASS +
2	Search with Issue	Test if the search work with Issue drop down menu	Select Ebola on the drop down of All Issue and click search	Check if there is 1 result, has ebola in ticket details	PASS +
3	Search with percent like	Test if the search work with input	Enter Angel and click search	Check if there is 1 result, and if it is related to Angel Island	PASS +

2. Microsoft Edge

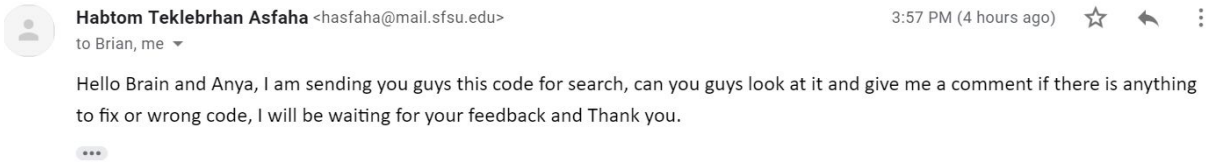
Test #	Test Title	Test description	Test input	Expected correct output	Test result
1	Search with Location + Issue	Test if the search work with Location drop down + Issue drop down menu	Select Alcatraz Island on the drop down of All Location and enter search	Check that if there are more than 1 result; does the result have Alcatraz details	PASS +
2	Search with Issue	Test if the search work with Issue drop down menu	Select Ebola on the drop down of All Issue and click search	Check that if there is 1 result; that is has Ebola related detail	PASS +
3	Search with percent like	Test if the search work with input	Enter Angel and click search	Check if there is 1 result, and if it is related to Angel Island	PASS +

4) Code Review:

Coding style

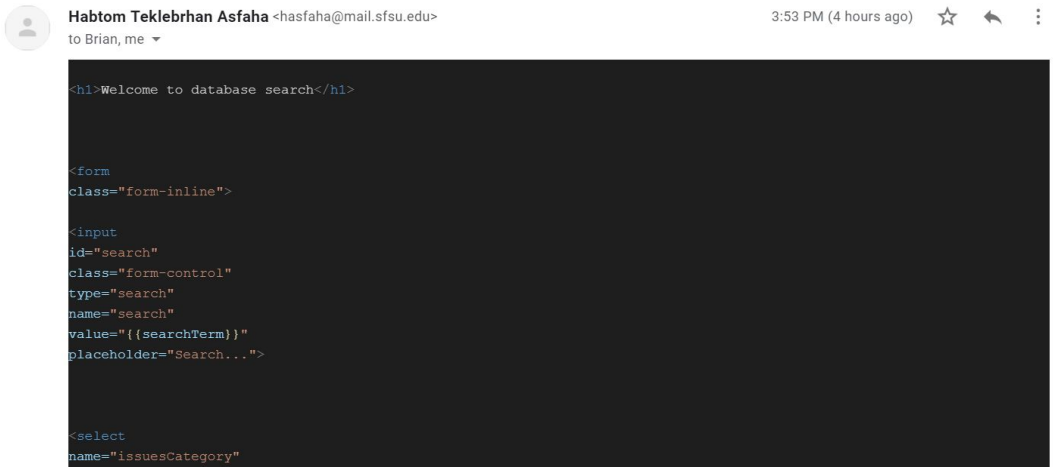
1. 2 spaces for indentation (HTML, CSS, JavaScript, Handlebars)
2. Opening brace on same line (JavaScript, CSS)
3. Closing brace on new line (JavaScript, CSS)
4. Space after keywords like “`if`”, “`else`”, etc (JavaScript)
5. Use arrow functions for callback functions (JavaScript)
6. Header comments use block comment (JavaScript)
7. Comments inside code use “`// `”, notice the space “`// `” (JavaScript)
8. Comments indented to the same level as context (All languages)
9. Comments have spaces before and after “`<!-- some comment -->`”, notice the spaces “`<!-- some comment -->`” (HTML, Handlebars)
10. Empty line between each class block (CSS)

Screenshots of code review process



Habtom Teklebrhan Asfaha <hasfaha@mail.sfsu.edu> 3:57 PM (4 hours ago)
to Brian, me

Hello Brain and Anya, I am sending you guys this code for search, can you guys look at it and give me a comment if there is anything to fix or wrong code, I will be waiting for your feedback and Thank you.



Habtom Teklebrhan Asfaha <hasfaha@mail.sfsu.edu> 3:53 PM (4 hours ago)
to Brian, me

```
<h1>Welcome to database search</h1>



<form
class="form-inline">

<input
id="search"
class="form-control"
type="search"
name="search"
value="{{searchTerm}}"
placeholder="Search...">



<select
name="issuesCategory"
```

Brian Hoang Lai <blai2@mail.sfsu.edu> 4:23 PM (4 hours ago)
to Habtom, me

Hello Habtom,

Thank you for sending in your HTML code of the search function for code review.

A few things I want to point out:

1. Hopefully your code is actually formatted correctly with the right amount of spaces for indentation. I assume sending the code through email has destroyed all formatting and everything is slammed to the left.
2. We may want the search functionality to be in the navbar so it will have to be moved to layout.hbs. The table of results will stay in its separate file: search.hbs.

Time for the code review:

1. The form tag is missing the method and action properties. It needs to have method="post" and action="/search".
2. For the table of search results, we need a better way of formatting the columns. Having a column for each field is a bit too excessive.

Here are my proposed changes:

search.hbs

1. Here is the new code for search.hbs. The form for the search functionality has been moved to layout.hbs as it will be living in the navbar. Submitting that form will route to this search page that shows results. You may also find new code from back-end. Please leave those be as back-end has also been making progress.
2.

```
<h1>Search results</h1>

<!-- Print any messages that is passed in from back-end such as "No results found" -->
<h4>
 {{#if msg}}
  {{msg}}
 {{else}}
  <!-- Print number of results or if no search query was submitted, prompt user to enter search -->
  {{#if numResults}}
   Showing <strong>{{numResults}}</strong> results
  {{else}}
   Please enter search term and/or search categories
  {{/if}}
 {{/if}}
</h4>
```

Brian Hoang Lai <blai2@mail.sfsu.edu> 4:26 PM (4 hours ago)

to Habtom, me

Additionally, please write header comments for search.hbs.

I have provided an example you may use. Feel free to add on to it

```
<!-- Front-end body for search results page -->
<!-- This page is accessed by submitting the search form at the top of the site in the navbar -->
```

5) Self-check on best practices for security

Major assets being protected:

1. User passwords - hash encrypted with salt
2. Database tables - images stored as relative path, not BLOBs
3. Registration and login - field validation and captcha
4. Image upload - captcha and image formats only, file size limit of 10 MB

Passwords:

Passwords are encrypted in the database when a user registers for an account. The passwords will appear in the database as a long hash string. This is done on the server side by using a NodeJS library called “bcrypt”. Bcrypt is a password hashing function that uses a salt to protect against rainbow table attacks. Bcrypt is an adaptive function so the iteration count can be increased to make it slower over time. Therefore, bcrypt remains resistant to brute-force search attacks even with increasing computing power.

Field Validation:

All input fields in the registration and sign-in forms are validated and required. Additionally, a captcha is required to be verified upon submitting the forms, there is no exception to this. This prevents spam bots from registering fake accounts on the website. When adding a ticket, the add ticket form includes issue name, location name, and rating fields which are all required and validated. There are two additional fields that are optional which are description and upload image. A captcha is also required to be verified when submitting a ticket to prevent upload spam. Lastly, the search bar is validated to be 0-50 alphanumeric characters. When left blank, search results will include every ticket. We used a NodeJS library called “express-validator” to validate fields. Express validator is a set of express.js middleware that wraps validator.js, which is a library that provides validator and sanitary functions. Express validator is incorporated into our app to perform server-sided field validation.

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

Copy all original non-functional specs as in high level application document published at the very beginning of the class. Then for each say either: DONE if it is done; ON TRACK if it is in the process of being done and you are sure it will be completed on time; or ISSUE meaning you have some problems and then explain it.

Note: you must adhere to all original non-functional specs as published in the original high level specification document. Failure to do so may cause reduced SE Product grade

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in MO (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 sewer.

   DONE

5. No more than 50 concurrent users shall be accessing the application at any time

- ON TRACK

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 (Clients with no email can log in, aka non-reg user)
- ON TRACK

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
- ON TRACK

14. Modem SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
- ON TRACK

15. The website shall 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