SW Engineering CSC648/848 Spring 2024 Section 2

Olympic Quest College Search

Team 02

Team Lead - Shriya Dandin (sdandin@sfsu.edu)
Front End Lead / System Administrator - Victoria Bialorucki (vbialorucki@mail.sfsu.edu)
Back End Lead - Sam Zandiasadabadi (szandiasadabadi@sfsu.edu)
Front End Team - Erika Chan (echan10@mail.sfsu.edu)
Github Master - Jake Klopukh (jklopukh@sfsu.edu)
Back End Team - Omar Ahmed (oahmed1@sfsu.edu)
Database Engineer - Subodh Khadka (skhadka@sfsu.edu)

Milestone 4 - Beta Launch, QA and Usability Testing and Final Commitment for Product Features (P1 list)

4/23/2024

Date Submitted	Date Revised
4/23/2023	

1. Product Summary

- **Name of Product**:
  - Olympic Quest College Search
- **All Major Functionality**:
  - Search and filter by name, state, and sport.
  - Rate a college and see a college’s average rating.
  - Contact via websockets so users can converse.
  - Register an account as either a student or staff member.
  - Search results uses Google Maps api.
  - Search results displays additional information about a college.
- **Unique Features**:
  - In addition to the functionality above, we are using websockets for real-time messaging rather than webmail.
  - Search automatically sorts rated colleges from highest to lowest.
  - A user can find the average rating for all the colleges; they can also see various details about a particular college.
  - A student or admin user can display their rating for a college.
  - A student or admin user can also favorite a college with the favorite button. This stores their favorited college in the database.
- **URL to Website**:
  - **http://52.2.194.93/html/website_html/webpage.html**

2. Usability Test Plan

Test Objective

The major function picked to be tested for usability is the search function. This function is responsible for searching through our database and showcase colleges that match an entered keyword inside the search bar. The purpose behind conducting this test plan is to ensure that the search function is implemented properly and behaves the way that it is supposed to. If the usability test plan is applied and the search function passes it, then the function is implemented properly and is ready for the next stage of production. If the function fails the test, then we must go back and re-implement the function to ensure that it passes the test next time.

Test Background and Setup

System Setup:
Providing a device connected to the internet to access the website. Additionally, having a secondary device that consistently takes pictures. An alternative would be to use the first device and screen record the test results to capture how the website behaves to different entries and commands in the search bar. Also having access to the database would be essential to ensure that there won't be any unexpected change in the database based on the input entered in the search bar.

Starting Point:
Defining the components of the search function to be tested, using different examples to test out the said components. Additionally, hiring professionals or assigning team members to form a focus group and conduct this test to ensure everything is up to standard. An example of this would be to assign the team lead to input different commands in the search bar and see the difference and the quality of the output. If we searched for colleges located in Texas, and a college such as San Francisco State University shows up, the search function is flawed and needs to be re-implemented.

Intended Users:
The focus groups would conduct various tests such as ensuring output validity so that the student-athletes who are the intended users of our websites can effortlessly find a suitable college and begin interacting and conversing with representatives of colleges to see which program would be the best available option for them.

URL of the System:
The search function and its filters are implemented in:
https://github.com/CSC-648-SFSU/csc-648-02-spring24-team02/blob/main/application/frontend/scripts/website_scripts/webpage_script.js

Usability Task Description

Visit Olympic Quest College Search and examine the search function provided in this webpage. To view all the offered and available colleges on the website, click on the search button, depicted as a white magnifying glass with a blue background. After examining all available colleges, click on “Search Colleges” and begin entering different inputs either based on the name of the college, the state they reside in, or the sport that they offer. To perform each search, make sure to use the appropriate ‘Name’, ‘State’, and ‘Sport’ filter (located on the left side of the search bar) respectively. For example, searching for San Francisco State University must be done under the ‘Name’ filter. Searching for all colleges located in California must be done under the ‘State’ filter, and searching for all colleges that offer a sport such as soccer must be done using the ‘Sport’ filter.

Measure Effectiveness:
Measure the effectiveness of the search function by determining the accuracy and completeness with which users achieve specified goals. Start this by testing all available features of the search function and determine if they successfully return the expected results within a certain period of time. Spend about 10 minutes completing this task. Then measure the percentage of people who completed the task amongst the focus group during the defined time. Lastly, count all the errors/incorrect outputs within each search result and write them down. We will use these data to determine the effectiveness of the search function.

Measure Efficiency:
Measure the efficiency of the search function by determining the resources expended in relation to the accuracy and completeness with which users achieve goals. Once again, start by testing the available features of the search function. However, this time take note of the average time it takes to properly search for a specific result. Additionally, keep track of the number of clicks, and number of screens, and page instructions that come up while conducting these searches. By doing so, we will measure the time efficiency, efficiency in effort, as well as efficiency in content. Having grasped all this data, we can determine the efficiency of the search function.

Lickert Subjective Test

After completing the above steps, take some time to reflect and answer the following questions on the scale from 1 to 5 from strongly disagree to strongly agree:

★ It is easy to search for a specific college given a name.
1.) Strongly Disagree 2.) Disagree 3.) Neutral 4.) Agree 5.) Strongly Agree

★ Please rate the clarity of instructions for performing different searches.
1.) Strongly Disagree 2.) Disagree 3.) Neutral 4.) Agree 5.) Strongly Agree

★ You are likely to recommend this website to others based on your experience today.
1.) Strongly Disagree 2.) Disagree 3.) Neutral 4.) Agree 5.) Strongly Agree

3. QA Test Plan

a. **Test Objectives:**
   i. Check whether searching accurately produces results from the database for the name, state, and sport filters.

b. **HW and SW setup:**
   i. Webpage to test: http://52.2.194.93/html/website_html/webpage.html
   ii. Computer HW (AWS Server):
      i. Processor:
      ii. 64-bit (x86)
      iii. Virtual CPU: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz
      iv. GiB Memory: 1 GB
      v. 30 GB storage
   i. Computer SW (AWS Server):
      i. Ubuntu 22.04.4 LTS
      ii. MySql Server v8.0.36-0ubuntu0.22.04.1 (Ubuntu)
      iii. Node v20.11.1

c. **Feature to be tested:**
   i. College search and filters

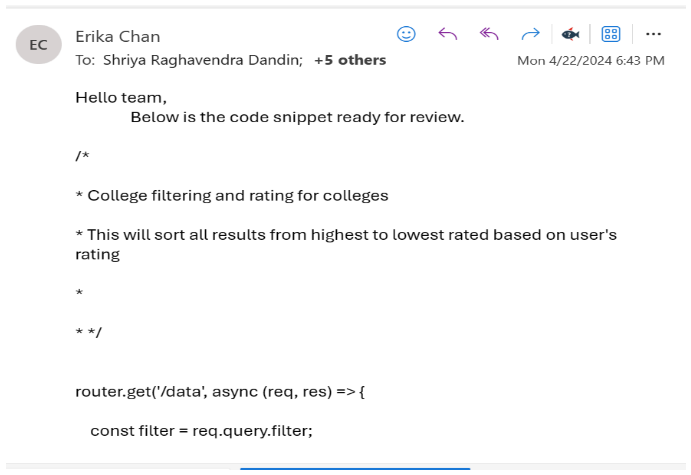
d. **QA Test plan:**

Test Number	Test Title	Description	Test Input	Expected output	Test Results
1	Name	Test % like in search for college_name field	Filter By set to “name” and enter “stanford”	Get 1 result exactly with college_name field of “Stanford University”	Chrome: PASS Firefox: PASS
2	State	Test % like in search for state field	Filter By set to “state” and enter “Ohio”	Get 2 results, each containing “Ohio” in the state field	Chrome: PASS Firefox: PASS
3	Sport	Test % like in search for olympic_sport field	Filter By set to “sport” and enter “rowing”	Get 5 results, each containing “rowing” in the olympic_sport field	Chrome: PASS Firefox: PASS

4. Code Review

a) Our coding style:
   i) Tabs for indentation
   ii) Separate files and routes for clarity
   iii) Comments for readability and maintainability
   iv) Different directories for organization

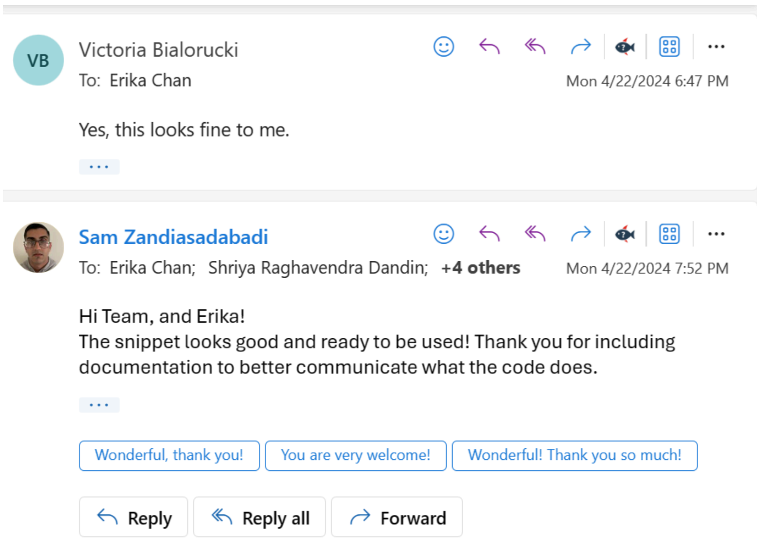
b) Peer Review & Code Snippet:



Erika Chan
To: Shriya Raghavendra Dandin; **+5 others**
Mon 4/22/2024 6:43 PM

Hello team,
Below is the code snippet ready for review.

```
/*

* College filtering and rating for colleges

* This will sort all results from highest to lowest rated based on user's
rating

*

* */


router.get('/data', async (req, res) => {

   const filter = req.query.filter;
```



Victoria Bialorucki
To: Erika Chan
Mon 4/22/2024 6:47 PM

Yes, this looks fine to me.

Sam Zandiasadabadi
To: Erika Chan; Shriya Raghavendra Dandin; **+4 others**
Mon 4/22/2024 7:52 PM

Hi Team, and Erika!
The snippet looks good and ready to be used! Thank you for including documentation to better communicate what the code does.

```
/*
* College filtering and rating for colleges
* This will sort all results from highest to lowest rated based on user's rating
*
* */

router.get('/data', async (req, res) => {
   const filter = req.query.filter;
   const input = req.query.input;
   const sortOrder = req.query.sortOrder || 'desc'; // Default to 'desc' for highest rated first
   const filterToCondition = {

      name: "college_name",
      state: "location",
      sport: "olympic_sport"

   };
...
// Filter the colleges
   const condition = filterToCondition[filter];
   if (condition) {
      const whereCondition =  WHERE ${condition} LIKE '%${input}%';
      queryStringAllColumns += whereCondition;
      queryStringAverageRating += whereCondition;
   }
...
```

5. Self-Check on Best Practices for Security

- **We are protecting**:
  - User accounts and their credentials.
  - The database and its tables.
  - The Ubuntu server, users in the Ubuntu server, and AWS.
  - The website by preventing hacking.

- **How we are protecting our major assets**:
  - Encryption for accounts and their login information
  - Password protection and not giving anyone access to the database outside of the team.
  - Using MFA to keep people from hacking into AWS; having preventions against brute-force script hacking like blocking ip addresses that attempt to login with our usernames and passwords. Keeping the server updated and installing security updates.
  - Allowing only our teammates access to the server and GitHub.

- **Encrypted passwords in the database**:
  - We have encrypted passwords in the database with the use of bcrypt on the back-end:

```
router.post('/register',...)...{
...
   const hashedPassword = await bcrypt.hash(password, 2);
...
```

```
router.post('/login',...)...{
...
if (userRows.length > 0) {
   // User is a student
   foundUser = userRows[0];
   passwordMatched = await bcrypt.compare(password, foundUser.password);
}
...
```

- **Input data validation**:
    - We have done input validation on the search bar with the following on the front-end:

```
function isValidInput(input) {
   const pattern = /^[a-zA-Z0-9\s]*$/;
   if (input.length > 40) {
       alert("Please enter no more than 40 characters.");
       clearSearchInput();
       return false;
   } else if (!pattern.test(input)) {
       alert("Please enter only letters and numbers.");
       clearSearchInput();
       return false;
   }
   return true;
}
```

        - isValidInput() forces the length of the string to be no longer than 40 characters; it also only allows valid input with either letters or numbers.

```
...
async function searchColleges() {
   const searchFilter =
document.getElementById("search-filter").value.toLowerCase();
   const searchInput =
document.getElementById("search-input").value.trim().toLowerCase();
...
```

        - Using value.trim() on the search-input stops whitespace and padding exploits from being used.

    - We have done input validation on the search bar with the following on the back-end:

```
router.get('/colleges',...)...{
...
if (req.query.name) {
   // Sanitize and validate the search query
   const searchQuery = req.query.name.trim();
   if (searchQuery.length < 2) {
       return res.status(400).json({error: 'Search query must be at least 2
characters long'});
   } ...
```

        - Searching must be at least 2 characters long. This isn't being used for now, since we want to be able to press the search button without input to display all the colleges.

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 (this is a plus) **DONE**

4. Data shall be stored in the team's chosen database technology on the team's deployment server. **DONE**

5. Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users. **DONE**

6. The language used shall be English. **DONE**

7. Application shall be very easy to use and intuitive. **DONE**

8. Google maps and analytics shall be added. **DONE**

9. No e-mail clients shall be allowed. You shall use webmail. **DONE**

10. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI. **DONE**

11. Site security: basic best practices shall be applied (as covered in the class) **DONE**

12. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development **DONE**

13. The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Spring 2024. For Demonstration Only"* at the top of the WWW page. (Important so not to confuse this with a real application). **DONE**