Milestone 4

# Olympian Tracker

SW Engineering CSC 648/848 Spring 2024

Section 2

04/23/2024

**Team 3**

Lei Woods (Team Lead)

Eric Kunzel

Chunyin (Alex) Chan (Front End Lead)

Elliot Bullard (Github Master)

Mila Avagimova

Oscar Galvez

Riel Orque (Back End Lead)

History Table

| **Submission Date:** | **Revision Date:** | **Revisions Made:** |
|---|---|---|
| 04/23/2024 | | |
| | | |
| | | |

## Section I: Product Summary

Product Name: **Olympian Tracker**
Major Committed Functions:

1. Home Page
2. Search Function and Search Page
3. Search Filtering
4. Data Upload
5. Register and Login Function
6. UI Responsiveness

Project URL: http://ec2-54-219-139-205.us-west-1.compute.amazonaws.com:3000/

## Section II: Usability Test Plan

**Introduction**
Number of Participants: 3-5
The purpose of this usability test is to evaluate the effectiveness, efficiency, and user satisfaction of the search function on Olympian Tracker. This test will be an exploratory assessment that will explore the search function itself, how easy it is to use, user satisfaction with the display of results, and the effectiveness of the result map.

Users will navigate through the search interface, providing feedback on their initial impressions and overall experience with the search page layout and design. Users will then be asked to search for a school using a keyword, partial word, and school acronym and asked about their initial opinion of the display of the results.

Throughout the test, testers will be encouraged to provide feature suggestions aimed at enhancing the search function's usability and intuitiveness. By soliciting user input, Olympian Tracker aims to gather valuable insights that will inform future enhancements and optimize the search experience for users.

**Test Background and Setup**
Test URL: http://ec2-54-219-139-205.us-west-1.compute.amazonaws.com:3000/search
System Setup: This test can be performed on any home computer or laptop with an internet connection. The test should be performed from the Google Chrome browser.
Starting Point: The starting point for all users assisting with this test should be the test link.
Intended Users: Intended users are individuals with UX/UI experience, individuals without UX/UI experience, but with programming experience and individuals without UX/UI experience and without programming experience. At least one user should be high-school aged if possible.

**Usability Task Descriptions:**
**Task 1:** Click the test link and navigate to the search page by clicking on the "search" button on the header of the homepage. User will then be asked to fill out a short questionnaire.
- Effectiveness will be measured in terms of users satisfaction with the ease of finding the search page from the landing page, and user satisfaction with the overall look of the search page.
- Efficiency will be measured on how quickly users were able to locate and navigate to the search page. Goal is under 5 seconds after opening the webpage.

**Task 2:** Select the search bar and search for a keyword or keyphrase. Do this by clicking your cursor in the search bar, typing in a keyword or keyphrase and navigating your cursor to click on the 'search' button. (e.g "California", "San Francisco", currently the database only contains schools in the CSU system.) User will then be asked to fill out a short questionnaire.

- Effectiveness will be measured in terms of users first opinions on the look of the search page, user satisfaction with ease of use in terms of searching for a keyword or key phrase and user satisfaction with the method of results display.
- Efficiency will be measured on how quickly users are able to type into the search and then actually search for results (e.g. should there be an on ‘enter’ event listener), and how quickly the results display on the page.

**Task 3:** Search CSU in the search bar. Do this by navigating your cursor back to the search bar, deleting the key phrase and typing CSU. Then navigate the cursor back to the search button and click it. User will then be asked to fill out a short questionnaire.

- Effectiveness will be measured in terms of users first opinions on the look of the search page, user satisfaction with ease of use in terms of searching for a school acronym and user satisfaction with the method of results display.
- Efficiency will be measured on how quickly users are able to type into the search and then actually search for results (e.g. should there be an on ‘enter’ event listener), and how quickly the results display on the page.

**Lickert Subjective Test:**
Task 1 Questions:
Question 1: I found the homepage easy to understand at first glance. (e.g. You could understand what the different parts of the homepage were and where to look for the search.)
Question 2: I found the search button easy to find. (e.g. You were able to find the button for the search page easily.)
Question 3: I found the search page easy to understand. (e.g. At first glance, the different parts of the search page made sense.)

Task 2 Questions:
Question 1: I found the search bar easy to find. (e.g. You could spot the search bar easily on the page.)
Question 2: I found the search bar easy to use and understand. (e.g. It was easy to click into the bar, type and then search for a keyword.)
Question 3: I found the results easy to understand and read. (e.g. When the results were populated, you could tell what schools the different results belonged to.)

Task 3 Questions:
Question 1: I found it easy to search for a different term after searching previously. (e.g. It was easy to shift from a previous search to a new search.)
Question 2: I found it easy to read the results when more than 2-4 results are displayed.
Question 3: I am satisfied with how the search functions and found the page easy to navigate.

## Section III: QA Test Plan

**QA Test Objectives:** The objectives of this test are to test the functionalities of the search and sort functions. Whether the search can find a single specific result, whether the search can search by state, whether the search can display results based on school acronym, and whether the search results can be sorted.
**HW and SW Setup:** Testing should be done on the production website via link on a home computer or laptop, using Google Chrome browser and Firefox Browser.
**Feature to be Tested:** Search and Sort functions
**QA Test Plan:**

| Test # | 1 |
|---|---|
| Title: | Display All Search |
| Description: | This will test the search functions ability to display all results by searching an empty search bar. |
| Input: | “ “ |
| Expected Correct Output: | 'CSU Bakersfield', 'CSU Channel Islands', 'Chico State', 'CSU Dominguez Hills', 'CSU East Bay', 'Fresno State', 'CSU Fullerton', 'Cal Poly Humboldt', 'CSU Long Beach', 'CSU Los Angeles', 'CSU Maritime Academy', 'CSU Monterey Bay', 'CSU Northridge', 'Cal Poly Pomona', 'CSU Sacramento', 'CSU San Bernardino', 'San Diego State University', 'San Francisco State University', 'San José State University ', 'Cal Poly San Luis Obispo', 'CSU San Marcos', 'CSU Sonoma', 'CSU Stanislaus' |
| Results: | Google Chrome: PASS Firefox: PASS |

| Test # | 2 |
|---|---|
| Title: | Search Keyword Return |
| Description | This test will be to test the overall functionality of the search function by searching keywords. |
| Input: | “San” |
| Expected Correct Output: | CSU San Marcos, San Francisco State University, San Jose State University, CSU San Bernardino, San Diego State University, Cal Poly San Luis Obispo |
| Results: | Google Chrome: PASS Firefox: PASS |

| | |
|---|---|
| Test # | 3 |
| Title: | Search Keyphrase Return |
| Description: | This test will be to test the overall functionality of the search function by searching keyphrases. |
| Input: | “San Francisco” |
| Expected Correct Output: | San Francisco State University |
| Results: | Google Chrome: PASS Firefox: PASS |

| | |
|---|---|
| Test # | 4 |
| Title: | Partial Word Search Return |
| Description: | This test will be to test the overall functionality of the search function by searching a partial word. |
| Input: | “Sa” |
| Expected Correct Output: | CSU Sacramento, San Jose State University, CSU San Marcos, San Francisco State University, CSU San Bernardino, San Diego State University, Cal Poly San Luis Obispo |
| Results: | Google Chrome: PASS Firefox: PASS |

| | |
|---|---|
| Test # | 5 |
| Title: | Display All in California Search |
| Description: | This will test the search functions ability to display all schools located in California. |
| Input: | “CA” |
| Expected Correct Output: | 'CSU Bakersfield', 'CSU Channel Islands', 'Chico State', 'CSU Dominguez Hills', 'CSU East Bay', 'Fresno State', 'CSU Fullerton', 'Cal Poly Humboldt', 'CSU Long Beach', 'CSU Los Angeles', 'CSU Maritime Academy', 'CSU Monterey Bay', 'CSU Northridge', 'Cal Poly Pomona', 'CSU Sacramento', 'CSU San Bernardino', 'San Diego State University', 'San Francisco State University', 'San José State University ', 'Cal Poly San Luis Obispo', 'CSU San Marcos', 'CSU Sonoma', 'CSU Stanislaus' |
| Results: | Google Chrome: PASS Firefox: PASS |

| | |
|---|---|
| Test # | 6 |
| Title: | Search School Acronym Return |
| Description: | This test will be to test the overall functionality of the search function by searching school acronyms. |
| Input: | “SFSU” |
| Expected Correct Output: | San Francisco State University |
| Results: | Google Chrome: FAIL Firefox: FAIL |

| | |
|---|---|
| Test # | 7 |
| Title: | Search School Sort By Rating |
| Description: | This test will be to test the overall sort functionality of the search function by trying to sort by rating. |
| Input: | “ “ |
| Expected Correct Output: | Should output universities based on a 1-5 scale rating. |
| Results: | Google Chrome: FAIL Firefox: FAIL |

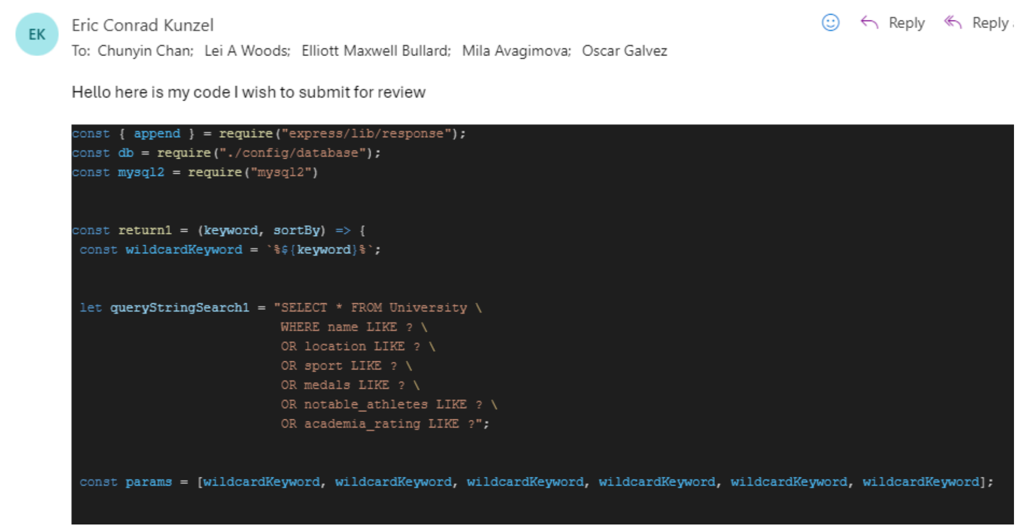
| | |
|---|---|
| Test # | 8 |
| Title: | Search School Sort Alphabetically |
| Description: | This test will be to test the overall sort functionality of the search function by trying to sort alphabetically. |
| Input: | “ “ |
| Expected Correct Output: | Should output universities in a sorted/alphabetized list.. |
| Results: | Google Chrome: PASS Firefox: PASS |

## Section IV: Code Review

Code To Be Reviewed:

```
const { append } = require("express/lib/response");
const db = require("./config/database");
const mysql2 = require("mysql2")

const return1 = (keyword, sortBy) => {
 const wildcardKeyword = `%${keyword}%`;

 let queryStringSearch1 = "SELECT * FROM University \
                           WHERE name LIKE ? \
                           OR location LIKE ? \
                           OR sport LIKE ? \
                           OR medals LIKE ? \
                           OR notable_athletes LIKE ? \
                           OR academia_rating LIKE ?";

 const params = [wildcardKeyword, wildcardKeyword, wildcardKeyword, wildcardKeyword,
wildcardKeyword, wildcardKeyword];

 //if a sort option is picked, an order clase is added to the sql statement
dynamically
 if (sortBy) {
   if (sortBy == "academia_rating"){
     //will be in descedning order if sorting based on rating
     queryStringSearch1 += " ORDER BY " + sortBy + " DESC"; // dynamically add the
ORDER BY clause
   } else {
     queryStringSearch1 += " ORDER BY " + sortBy; // dynamically add the ORDER BY
clause
   }
 }

 //db.query is called in a promise
 return new Promise(function(resolve, reject) {
   db.query(queryStringSearch1, params, function (error, results, fields) {
     if (error) {
       return reject(error);
     }
     resolve(results);
   });
 });
};

// endpoint uni-specific search: uni_name, keyword, triggered when user supplies
dropdown selection
const return2 = (keyword, dropDownSel, sortBy) => {
 const wildcardKeyword = `%${keyword}%`;
 const wildcardKeyword2 = `%${dropDownSel}%`;
 let queryStringSearch2 = "SELECT * FROM University \
                           WHERE sport LIKE ? \
                           AND (location LIKE ? \
                           OR name LIKE ?\
                           OR medals LIKE ? \
                           OR notable_athletes LIKE ? \
                           OR academia_rating LIKE ?)";

 const params = [wildcardKeyword2, wildcardKeyword, wildcardKeyword, wildcardKeyword,
wildcardKeyword, wildcardKeyword];

 if (sortBy) {
   queryStringSearch2 += " ORDER BY " + sortBy; // dynamically add the ORDER BY clause
 }

 return new Promise(function(resolve, reject) {
   db.query(queryStringSearch2, params, function (error, results, fields) {
     if (error) {
       return reject(error);
     }
     resolve(results);
   });
 });
};

module.exports = {return1, return2}
```

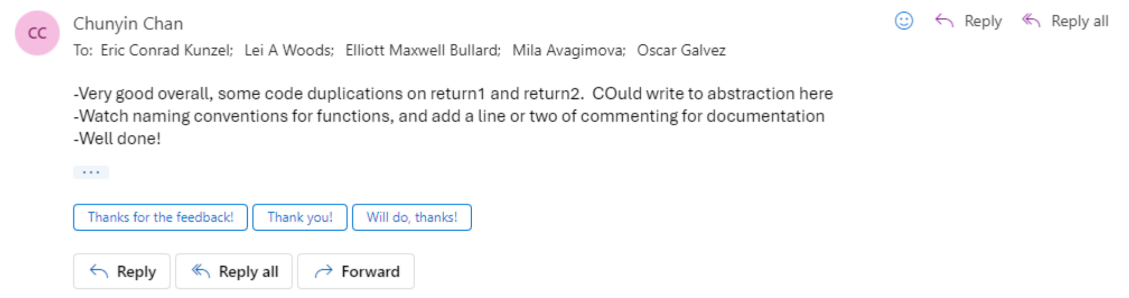
## Code Reviews:



Eric Conrad Kunzel
To: Chunyin Chan; Lei A Woods; Elliott Maxwell Bullard; Mila Avagimova; Oscar Galvez

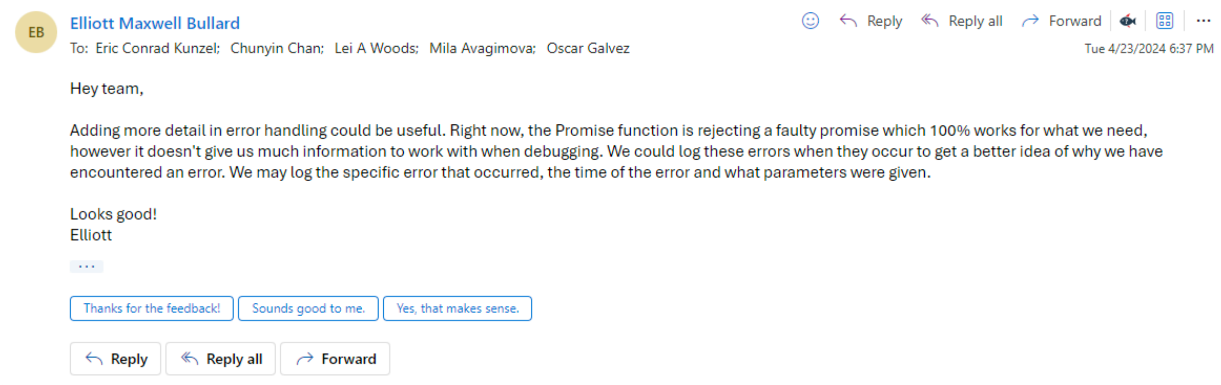
Hello here is my code I wish to submit for review

```
const { append } = require("express/lib/response");
const db = require("./config/database");
const mysql2 = require("mysql2")


const return1 = (keyword, sortBy) => {
 const wildcardKeyword = `%${keyword}%`;


 let queryStringSearch1 = "SELECT * FROM University \
                           WHERE name LIKE ? \
                           OR location LIKE ? \
                           OR sport LIKE ? \
                           OR medals LIKE ? \
                           OR notable_athletes LIKE ? \
                           OR academia_rating LIKE ?";


 const params = [wildcardKeyword, wildcardKeyword, wildcardKeyword, wildcardKeyword, wildcardKeyword, wildcardKeyword];
```



Chunyin Chan
To: Eric Conrad Kunzel; Lei A Woods; Elliott Maxwell Bullard; Mila Avagimova; Oscar Galvez

-Very good overall, some code duplications on return1 and return2. COuld write to abstraction here
-Watch naming conventions for functions, and add a line or two of commenting for documentation
-Well done!

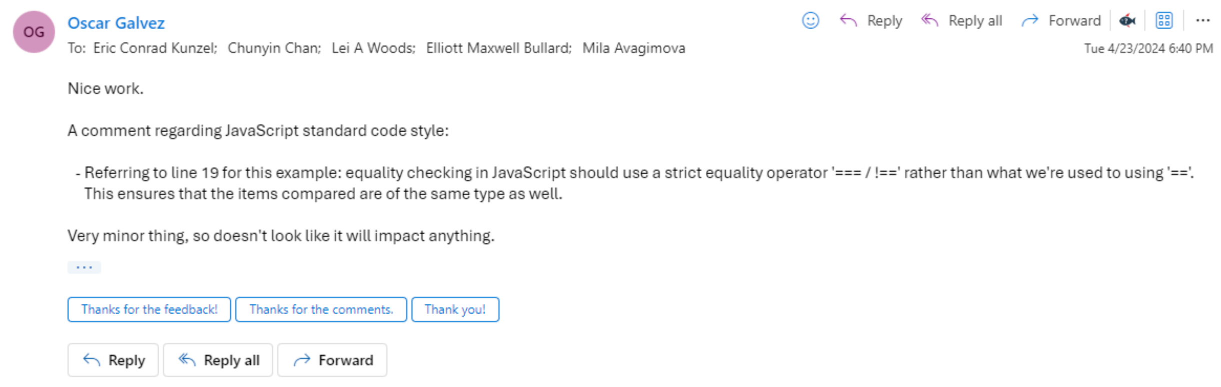


Elliott Maxwell Bullard
To: Eric Conrad Kunzel; Chunyin Chan; Lei A Woods; Mila Avagimova; Oscar Galvez
Tue 4/23/2024 6:37 PM

Hey team,

Adding more detail in error handling could be useful. Right now, the Promise function is rejecting a faulty promise which 100% works for what we need, however it doesn't give us much information to work with when debugging. We could log these errors when they occur to get a better idea of why we have encountered an error. We may log the specific error that occurred, the time of the error and what parameters were given.

Looks good!
Elliott



Oscar Galvez
To: Eric Conrad Kunzel; Chunyin Chan; Lei A Woods; Elliott Maxwell Bullard; Mila Avagimova
Tue 4/23/2024 6:40 PM

Nice work.

A comment regarding JavaScript standard code style:

- Referring to line 19 for this example: equality checking in JavaScript should use a strict equality operator '=== / !==' rather than what we're used to using '=='. This ensures that the items compared are of the same type as well.

Very minor thing, so doesn't look like it will impact anything.

RO **Riel Orque**

Reply Reply all Forward

To: Lei A Woods; Eric Conrad Kunzel

Tue 4/23/2024 6:44 PM

Hey Eric,

Great job so far but I would rename functions return1 and return2 to something meaningful and descriptive as it pertains to searching Universities. Another recommendation would be to also use SQL Injection Protection for the sortBy query. There might be a risk from the sortBy although there are no direct inputs from the user. The use of parametrized queries is good practice but directly appending user input to the query could cause injections.

Riel

Thanks for the feedback! Thank you so much for your feedback. Thank you for the suggestions.

Reply Reply all Forward

## Section V: Self-check on Best Practices for Security

Major Assets Protected:

- Users Password
    - User passwords will be protected by being stored as hashes in the database and when a user is asked to log in, the password entered upon login will be hashed and compared to the accompanying hash stored in the database. Un-hashed passwords will NEVER be stored.
    - When Users enter their passwords on account creation, they will be asked to confirm the entered password. Both the original password and the comparison password will be hashed and the hashes will be compared against each other, not the actual typed passwords.

Information Being Validated:

- User Email
- User Password
- Input Data:
    - Search bar input is validated for up to 40 alphanumeric characters.

## Section VI: 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. Only Desktop and laptop browsers will be used to test the performance and accessibility of this application. DONE
3. The application should be able to render on a mobile device, while not mandatory. ON TRACK
4. Data shall be stored in the team's chosen database technology on the team's AWS EC2 server. DONE
5. All user privacy shall be secured and protected to adhere to real-world legal standards. ISSUE - The team member who was supposed to be working on log-in and register has not been present because of family emergencies, so this is behind schedule.
6. The language shall be English. DONE
7. Application shall be very easy for users of all technological backgrounds to easily navigate. DONE
8. Google maps and analytics shall be used in this application. ON TRACK
9. No email clients, only webmail. ON TRACK
10. We will not implement pay functionality in this application. DONE
11. We shall use the best security practices learned in class for the application and the users. ISSUE - The team member who was supposed to be working on log-in and register has not been present because of family emergencies, so this is behind schedule.
12. Modern SE processes and practices shall be practiced 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