SW Engineering CSC648/848 Fall 2021

Project Title: Gator Learn

Team Number: 04

Names of Students:

Manali Seth (Team Lead) (Email: mseth@mail.sfsu.edu)
Htet Soe (Front-end Lead)
Aarshil Patel (Back-end Lead)
Aditya Mohan (Github Master)
Seela Pant
Christian Samatra
William Yu

Milestone: 4

Date: 12/18/2021

History Table:

Date	Revision
12/18/2021	Created as the first version

Table of Contents

1. Product Summary

Name of the Product: Gator Learn

Major Functions:

1. Browsing
   One can browse through different pages from navigation bar. One can view recently approved tutor postings from the home page.

2. Searching:
   One can search list of tutors based on chosen major & courses available at SFSU.

3. Login
   One can login in Gator Learn Website to be able to message tutor or apply for tutor posting.

4. Registration
   One can register to become a Gator Learn member to have more priviledges than an unregistered user.

5. View SFSU Course Catalog
   Displays full list of different Majors and Courses available at SFSU

6. Tutor Posting
   One can apply for Tutor Posting to be able to let users know which course one will be tutoring for.

7. Admin Approval
   Tutor Postings are live only when admin approves them from the backend (SQL Workbench)

8. Send Message to Tutors
   Message can be sent to tutors only when one is registered.

9. Contact Us
   One can contact Gator Learn creators via email by clicking on Contact Us.

10. About Us
    One can browse individual profile of Gator Learn creators by clicking About Us.

Uniqueness:

- The Gator Learn website is designed exclusively for SFSU users.
- One can message tutors or apply for tutor posting only once logged in else there is lazy registration.
- One can view recently approved tutor postings on home page so to have the latest updates.

URL to Product: http://3.144.136.131:8000/

Usability Test Plan

Usability Test: Search implementation

- Test Objectives

The main purpose of the usability test plan is to provide a simple user experience for users. This allows users around the globe to enjoy and take advantage of online platforms like tutoring services. We have a simple online tutoring website where users can easily search for their class tutor and get help from them. Search function is one of the important functions that helps users to narrow down their results.

The main objective is to test the search function and see if it is convenient to users and if it correctly responds to the user input and provides the information that best matches the user's search text. Another objective is to get the search results even if there is typo or less information from users input.

With all these objectives mentioned, the development team will also get the feedback from users and can bring better user experience in future.

- Test background and setup

System setup

The system can be set up easily for the users. They just need to use our online website (Gator Learn) URL and launch the website on his/her computer / laptop and make sure the internet is connected. As of now, our website is supported by two browsers i.e. Chrome and Firefox.

Starting point

To effectively test our search function, we planned on having some simple user search for various course listings we have in our home page. If the character(s) or the length of user input is valid, relatable search results will be shown. Otherwise, the message will prompt as 'Search input must be alphanumeric' or 'Invalid Input' or 'Search input must be less than 40 characters' depending upon the case. In this case, we can check how quickly and correctly users get the results based on their input.

Who are the intended users?

The intended users for our website are registered, unregistered, and admin users. If the users are just there to browse and see what we have on our website, they do not have to login or register. Unregistered users can use our search function several times and check all the listings we have in our website so that they can decide if they want to contact our tutors and get help from them.

URL of the system to be tested: http://3.144.136.131:8000/

- Usability Task description

Users will be asked to search for a specific major or course. The results should only show listings of tutors according to the major and course searched.

1) User can just click on search button directly to list all the tutors available.

2) User can just select the major from the dropdown and click on search button to list tutors based on major tutored.

3) User can just type in the course name by typing in some characters and then choosing one from the list available using autocomplete feature and have list of tutors based on course tutored.

4) User can select both the major (from dropdown) and type in any course name to see list of tutors available based on major and course tutored.

- Evaluation of Effectiveness

It deals with how many users are able to complete the task effectively.

Search: Users even with average computing skills can implement the above 4 test cases with 0 errors and see the list of tutors as required.

- Evaluation of Efficiency

The users can view the search results within few milliseconds by choosing major and typing in the course name and by hitting on search button in max 3 clicks which shows good efficiency.

- Evaluation of User satisfaction

	STRONGLY AGREE	AGREE	NEUTRAL	DISAGREE	STRONGLY DISAGREE
THE SEARCH RESULTS WERE DISPLAYED CORRECTLY					
THE PROCESS OF SEARCHING A TUTOR LISTING WAS EASY					
THE SEARCH BAR WAS CONVENIENT TO USE					

General Comments/Feedback:

3. QA Test Plan

- Test Objectives

Use case: Searching as a non-registered/registered user.

Outline of Test plan
We are testing the search function for when users search for tutors that are available exclusively for SFSU users.

- HW and SW Setup

HW Setup: You just need a computer/ laptop for testing

SW Setup:

Server Host: AWS 1 vCPU 1 GB RAM
Operating System: Ubuntu 20.04
Server Database: MySQL Workbench
Server-Side Language: Python 3.7/+
Browser: Google/ Firefox
Additional Technologies:
Web Framework: Flask
IDE: PyCharm
Web Analytics: Google Analytics

- Feature to be tested

Test Sequence
Go to the app URL, open up our application.

Locate dropdown menu next to search bar (if displaying search results separated based on major), choose "Accounting", click search, and make sure results contain "5" results from "Accounting"

Go to course input field
Type <course name>, hit search button, make sure response is <list of tutors based on course typed in> (check for correctness of operation)

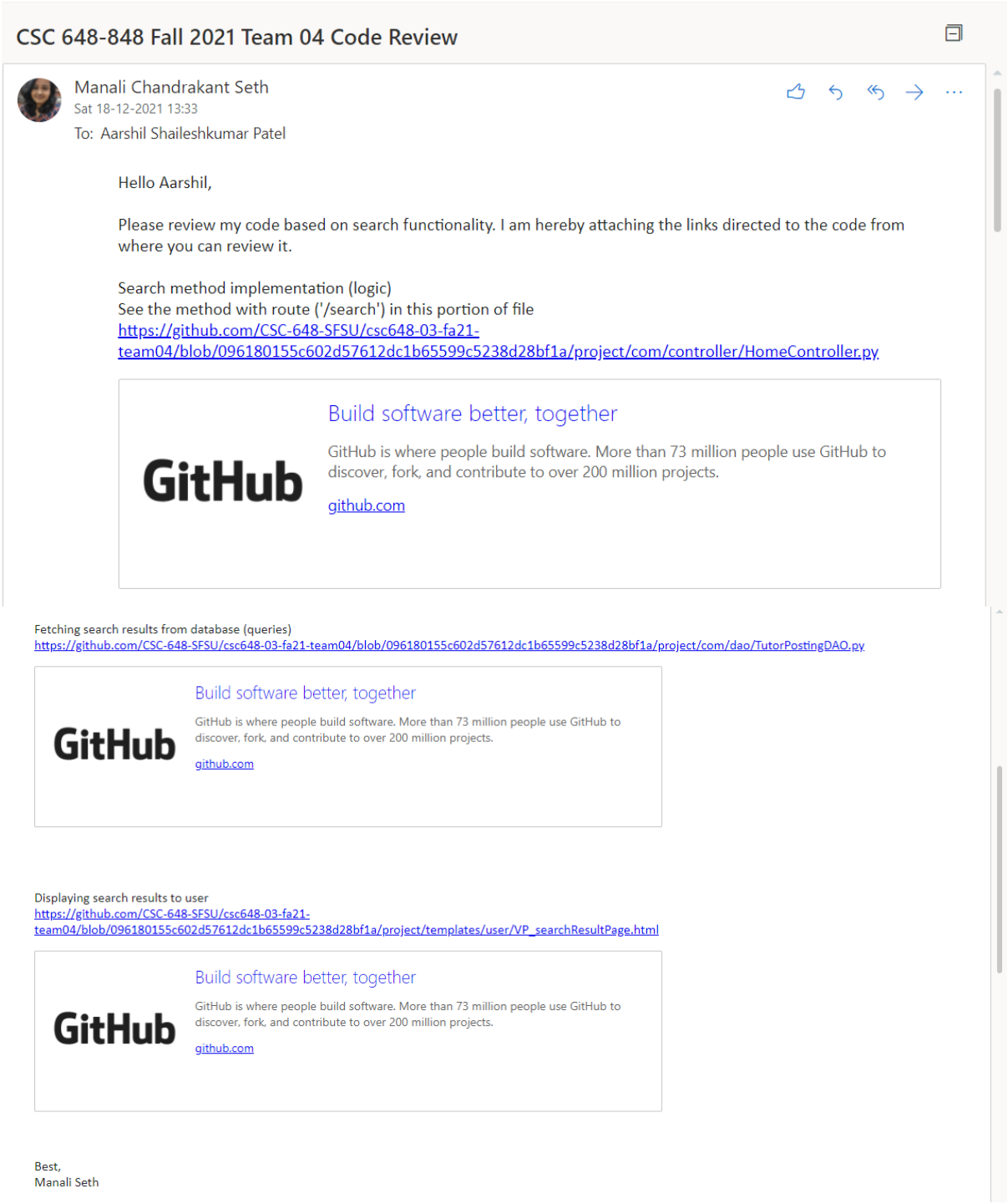
We are testing for search function accuracy - making sure text in search fields gives correct autocomplete results while typing in course name input, e.g. typing "fin" might display results such as "Introduction to **Fin**Tech".

We also want to test for search input validation, making sure special characters aren't allowed or inputs that are too long (over 40 characters).

- **QA Test Plan**

TEST NUMBER	TEST TITLE	TEST DESCRIPTION	TEST INPUT	EXPECTED OUTPUT	TEST RESULTS (PASS/FAIL)
1	Search results test	Test % like in search for "All majors" field	Type "Hatha Yoga" in course name input field	Get 2 results, of all 2 results have "Hatha Yoga" in course tutored column	Pass in Chrome/ Pass in Firefox
2	Attempting to message tutor while not logged in	Test sending of message to tutor inbox	Typed the entire message "Hi, I am interested in…" while only browsing	User will be redirected to registration page (lazy-registration)	Pass in Chrome/ Pass in Firefox
3	Search input validation	Test the input validation in search field, no special characters	Select a category and type a special character (*, @, $) in search field	An alert message pops "Search input must be alphanumeric"	Pass in Chrome/ Pass in Firefox
4	Search input validation	Test the input validation in search field, character limit	Type "This search request contains too many characters" in search field	Site should limit and prevent searches with a 40 character limit	Pass in Chrome/ Pass in Firefox

4. Code Review



CSC 648-848 Fall 2021 Team 04 Code Review

Manali Chandrakant Seth
Sat 18-12-2021 13:33
To: Aarshil Shaileshkumar Patel

Hello Aarshil,

Please review my code based on search functionality. I am hereby attaching the links directed to the code from where you can review it.

Search method implementation (logic)
See the method with route ('/search') in this portion of file
https://github.com/CSC-648-SFSU/csc648-03-fa21-team04/blob/096180155c602d57612dc1b65599c5238d28bf1a/project/com/controller/HomeController.py

GitHub
Build software better, together
GitHub is where people build software. More than 73 million people use GitHub to discover, fork, and contribute to over 200 million projects.
github.com

Fetching search results from database (queries)
https://github.com/CSC-648-SFSU/csc648-03-fa21-team04/blob/096180155c602d57612dc1b65599c5238d28bf1a/project/com/dao/TutorPostingDAO.py

GitHub
Build software better, together
GitHub is where people build software. More than 73 million people use GitHub to discover, fork, and contribute to over 200 million projects.
github.com

Displaying search results to user
https://github.com/CSC-648-SFSU/csc648-03-fa21-team04/blob/096180155c602d57612dc1b65599c5238d28bf1a/project/templates/user/VP_searchResultPage.html

GitHub
Build software better, together
GitHub is where people build software. More than 73 million people use GitHub to discover, fork, and contribute to over 200 million projects.
github.com

Best,
Manali Seth

Search functionality related links to have Peer's code review on it

CSC 648-848 Fall 2021 Team 04 Code Review

AP

Aarshil Shaileshkumar Patel
Sat 18-12-2021 14:47
To: Manali Chandrakant Seth

Hello Manali,

I had an overview of the code you sent. And I would like to summarize it here.

Overall, I think the code is very well organized.

The comments in the code helped me understand clearly.

The code is properly indented as well.

The class/methods/variable names are consistent too.

They also meet the naming established before.

The commit messages in the github are also to the point and gives clear idea about that particular commit.

MVC architecture is well maintained.

Proper usage of flask routes to navigate between pages.

Well written header and inline comments.

In general, I believe the code meets the standard requirements and it is efficient.

Thank you

Aarshil Patel

Code Review

5. Self-check on best practices for Security

Asset to be protected	Types of possible/expected attacks	Your strategy to mitigate/protect the asset
The user's login/password database (High value)	Unauthorized access to the database (High probability)	Any new registered users will now have their passwords encrypted in the database. Track and manage database access and restrict usage to only admins.
The user's login/password database (High value)	SQL Injection through login/registration forms (Medium probability)	Input validation on login and registration forms that requires input to contain alphanumeric characters, other than e-mail, and have under 40 characters. Further validation on the e-mail where the suffix of the input must be "sfsu.edu".
The tutor/course database (Low value)	SQL Injection through the search bar (Medium probability)	Input validation on the search bar that requires input to only contain alphanumeric characters and have under 40 characters. The tutor and course database does not contain sensitive or confidential data even if it were to be accessed without authorization.
Application's hosting service credentials (High value)	Unauthorized user makes application unavailable (Low probability)	Strict management to credentials and keys to web hosting server.
An individual user's account (Low value, with exceptions)	Unauthorized access to the user's account (Medium probability)	A contacted admin can use the database-stored e-mail for verification in order to restore or reset a user's login credentials.

6. Self-check on adherence to non-functional specs

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in Milestone 0. Application delivery shall be from chosen cloud server. **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. All or selected application functions must render well on mobile devices. **ON TRACK**

4. Data shall be stored in the database on the team's deployment cloud 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 (no localization needed). **DONE**

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

9. Application should follow established architecture patterns. **DONE**

10. Application code and its repository shall be easy to inspect and maintain. **DONE**

11. Google analytics shall be used **DONE**

12. No e-mail clients shall be allowed. **DONE**

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

14. Site security: basic best practices shall be applied (as covered in the class) for main data items. **DONE**

15. Application shall be media rich (images, video etc.). Media formats shall be standard as used in the market today. **DONE**

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

17. For code development and management, as well as documentation like formal milestones required in the class, each team shall use their own github to be set-up by class instructors and started by each team during Milestone 0. **DONE**

18. The application UI (WWW and mobile) shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Fall 2021 For Demonstration Only" at the top of the WWW page. (Important so as to not confuse this with a real application). **DONE**