SW Engineering CSC 648-848 Fall 2024

Tutoring Platform: GatorAid

Team 04:

André Shannon (Team Lead, ashannon1@sfsu.edu)

Fernando Malca (Back-End Lead)

Anh Tran (Front-End Lead)

Geetarth Meduri (GitHub Master)

Aditya Sharma (Full-Stack Member)

Milestone 4

December 7, 2024

Date Submitted	Date Revised
12/18/2024	N/A

Product summary:

Name of the Product
GatorAid

Description
GatorAid is a tutoring platform designed specifically for the San Francisco State University (SFSU) community. Its primary purpose is to connect students with SFSU-affiliated tutors who are knowledgeable about specific courses offered at the university. Unlike generic tutoring platforms, GatorAid is tailored to SFSU's curriculum, ensuring course-specific academic guidance. This streamlined and accessible platform facilitates internal communication between tutors and tutees while ensuring all users are affiliated with SFSU, creating a highly focused and relevant tutoring experience.

Major Committed Functions (P1 List)
UnregisteredUsers

1. Users shall be able to register
2. Users shall be able to browse the tutors
3. Users shall be able to view tutor info
4. Users shall be able to search the tutors by subject and tutor name
5. Users shall be able to search the tutors by SFSU class name/number
6. Users shall be able to sort tutors by price range
7. Users shall be able to view an about page of the team members

RegisteredUsers

8. RegisteredUsers shall inherit all the capabilities of unregistered users
9. RegisteredUsers shall be able to apply to become a tutor
10. RegisteredUsers shall be able to send messages to tutors
11. The RegisteredUsers' dashboard shall display messages received from tutees
12. The RegisteredUsers' dashboard shall display any of the user's tutor posts
13. RegisteredUsers shall be able to login
14. RegisteredUsers' can specify courses they are offering tutoring for.
15. All RegisteredUsers must be verified as affiliated with SFSU before account creation.

Admins

14. Admins shall inherit all the capabilities of registered users
15. Admins shall be required to approve or reject tutor postings before they go live
16. Admins shall be able to remove postings
17. Admins shall be able to remove users

URL: ec2-54-215-254-157.us-west-1.compute.amazonaws.com

Usability test plan for selected function:

1. Test objectives:
Want to test the usability of the search bar. The search bar is our main distinguishing function that differs from other tutor platforms and should be very intuitive to use. It will probably be one of the most used features of our platform.

2. Test background and setup

To set up the system and the starting point, users need a desktop with a network connection. They also need either the Chrome or Firefox browser.

The intended users are current SFSU college students who want extra help for courses they are taking. The age range can be 18+ and it is assumed that college students are familiar with using computers.

The URL of the system is http://ec2-54-215-254-157.us-west-1.compute.amazonaws.com/.

For the test environment, testers will be in the lab with cameras (can be a room and a smartphone). They do not need any training.

3. Usability Task description:
Navigate to the tutor website listed above. Find a math tutor named Emily Knight and then find a tutor that has taken CSC 415.

4. Plan for evaluation of Effectiveness:
If the user successfully locates the two tutors then they get full marks for effectiveness. If they only locate one, 50%, and none, 0.

5. Plan for Evaluation of efficiency:
We will use time and number of clicks to measure efficiency. If testers complete it in under 30 seconds and under 8 clicks, they get full marks, if it takes longer than 30 seconds or more than 5 clicks, then they can get marked down 1 point for every second and 5 for every click.

6. Plan for Evaluation of user satisfaction (Likert scale questionnaire):
The search bar was easy to locate.
__Strongly Agree __Agree __Nuetral __Disagree __Strongly Disagree

Selecting the subject on the side of the search bar was very intuitive.
__Strongly Agree __Agree __Nuetral __Disagree __Strongly Disagree

It was easy to identify which tutors had taken what classes.
__Strongly Agree __Agree __Nuetral __Disagree __Strongly Disagree

QA test plan and QA testing:

1. Test objectives: QA test on search bar

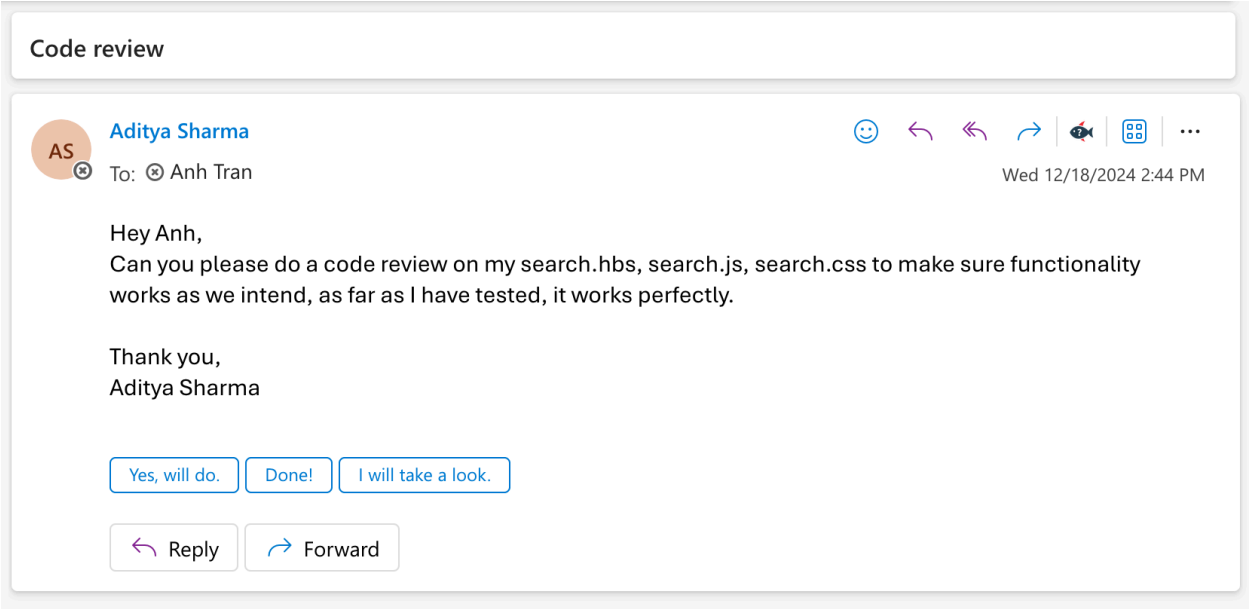
2. HW and SW setup (including URL): The URL of the system is http://ec2-54-215-254-157.us-west-1.compute.amazonaws.com/.

3. Feature to be tested:
Search shows all the tutors for all the subjects and shows specific tutors for specific subjects.

4. QA Test plan:

Test number	Test title	Test description	Test input	Expected output	PASS/ FAIL
1	Quality assurance test on search bar for all subjects	Test database as it will return all the tutor posts	Choose “All Subjects” in the search field	Get 9 results as all tutors in all subjects	PASS
2	Quality assurance test on search bar for specific subject	Testing filter by subject	Choose “CSC” in the search field	Get 6 results as all CSC tutors	PASS
3	Quality assurance test on search bar for specific name	Testing filter by specific name	Choose “Alice Smith” in the search field	Get 1 result back with exact tutor looking for	PASS

Peer Code Review:



Review comments for search.hbs:

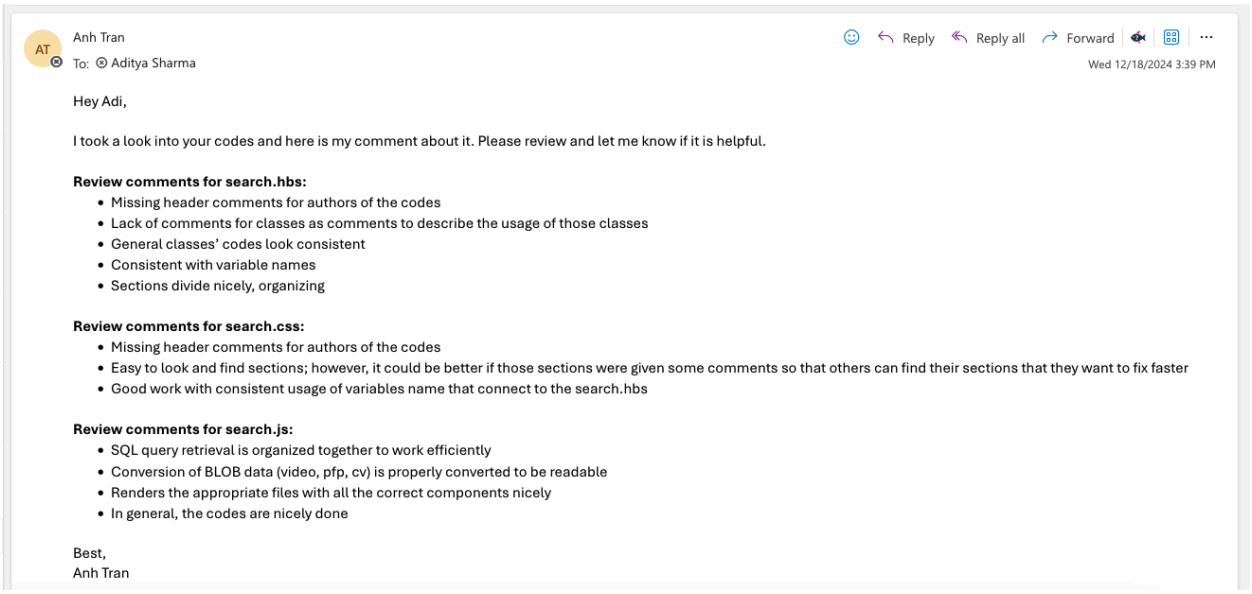
- Missing header comments for authors of the codes
- Lack of comments for classes as comments to describe the usage of those classes
- General classes' codes look consistent
- Consistent with variable names
- Sections divide nicely, organizing

Review comments for search.css:

- Missing header comments for authors of the codes
- Easy to look and find sections; however, it could be better if those sections were given some comments so that others can find their sections that they want to fix faster
- Good work with consistent usage of variables name that connect to the search.hbs

Review comments for search.js:

- SQL query retrieval is organized together to work efficiently
- Conversion of BLOB data (video, pfp, cv) is properly converted to be readable
- Renders the appropriate files with all the correct components nicely
- In general, the codes are nicely done



Anh Tran
To: Aditya Sharma

Reply | Reply all | Forward

Wed 12/18/2024 3:39 PM

Hey Adi,

I took a look into your codes and here is my comment about it. Please review and let me know if it is helpful.

Review comments for search.hbs:

- Missing header comments for authors of the codes
- Lack of comments for classes as comments to describe the usage of those classes
- General classes' codes look consistent
- Consistent with variable names
- Sections divide nicely, organizing

Review comments for search.css:

- Missing header comments for authors of the codes
- Easy to look and find sections; however, it could be better if those sections were given some comments so that others can find their sections that they want to fix faster
- Good work with consistent usage of variables name that connect to the search.hbs

Review comments for search.js:

- SQL query retrieval is organized together to work efficiently
- Conversion of BLOB data (video, pfp, cv) is properly converted to be readable
- Renders the appropriate files with all the correct components nicely
- In general, the codes are nicely done

Best,
Anh Tran

Self-check on best practices for security:

Asset to be protected	**Types of possible/expected attacks**	**Consequence of security breach**	**Your strategy to mitigate/protect the asset**
User Credentials (Name, Email, Password)	- Brute force attacks - Password cracking - Phishing - Database leaks	- Unauthorized access to user accounts - Exposure of personal information	- encrypt passwords
Tutor Post Information	- Unauthorized database access - Injection attacks (SQL) - Scraping	- Leakage of personal tutor data (e.g., bio, video, ratings) - Potential harassment	- Input validation to prevent SQL injection. - Only associated user and admin can make changes to the TutorPost
One-Way Messages	- Unauthorized access to messages - Eavesdropping on message traffic	- Privacy violation for users - Leakage of sensitive user messages	- Restrict message access to sender/receiver
File Uploads (Tutor Posts	- Malware injection - File upload abuse	- Platform compromise	- Restrict uploaded file types and size

Pictures/Videos)		- Resource exhaustion	

The password is encrypted in the database.
For input validation:

- Search bar input for up to 40 alphanumeric characters
- SFSU customer registration e-mail to include “sfsu.edu” at the end
- Acceptance of terms (terms is a dummy link) in registration form

List major assets you are protecting in your team application
- List major threats for each asset above
- For each asset above say how you are protecting it (1-2 lines of text per each)
- Confirm that you encrypt PW in the DB
- Confirm Input data validation (list what is being validated and what code you used) .

NOTE: We require that at minimum you validate
- Search bar input for up to 40 alphanumeric characters
- SFSU customer registration e-mail to include “sfsu.edu” at the end
- Acceptance of terms (terms is a dummy link) in registration form

Self-check of the 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

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

ON TRACK

3. All or selected application functions shall render well on mobile devices (no native app to be developed)

ON TRACK

4. Posting of tutor information and messaging to tutors shall be limited only to SFSU students

DONE

5. Critical data shall be stored in the database on the team’s deployment server.

DONE

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

DONE

7. Privacy of users shall be protected

ON TRACK

8. The language used shall be English (no localization needed)

DONE

9. Application shall be very easy to use and intuitive

ON TRACK

10. Application shall follow established architecture patterns

DONE

11. Application code and its repository shall be easy to inspect and maintain

ON TRACK

12. Google analytics shall be used

ISSUE - didn't get the tag set up on the server soon enough to collect data

13. No e-mail clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application

DONE

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

DONE

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

ON TRACK

16. Media formats shall be standard as used in the market today

ON TRACK

17. Modern SE processes and tools shall be used as specified in the class, including collaborative and continuous SW development and GenAI tools

ON TRACK

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 2024. For Demonstration Only" at the top of the WWW page Nav bar. (Important so as to not confuse this with a real application).

DONE

Use of genAI tools like ChatGPT and copilot:

- ChatGPT was great for finding assets, expected attacks, consequences, and mitigation strategies.
- We used ChatGPT as a reference tool to search up specific needs like how to change or move an element using CSS and make some edits to the JS files that query our DB.
- We used it for making routes

- Polish some of our english in our Documentation