SW Engineering CSC 648/848 Fall 2024

Team 08

StudyGator

Team Lead: Bryan Lee (blee37@sfsu.edu)
Front-end lead: Min Ye Thway Khaing
Back-end lead: Nishi Suratia
GitHub master: Kenneth Wen

Milestone 4

November 24, 2024

Date Submitted	Date Revised
12/18/2024	n/a

1. Product Summary

Name of product: StudyGator

Product Description:

StudyGator is a tutoring platform allowing SFSU students to tutor and receive tutoring from other SFSU students. While there are numerous other tutoring websites to choose from, StudyGator stands out by connecting students with tutors who are also SFSU students, offering firsthand knowledge of the same course materials and challenges. Users can easily find tutors by searching subjects and keywords or filtering by price to fit their budget. For those looking to offer tutoring services, **StudyGator** provides a seamless process to create and promote listings. **StudyGator** ensures that academic help is accessible and is always just a click away to empower SFSU students to focus on what matters most–their educational success.

The itemized list of ALL primary committed functions:

- Unregistered user
  1. Unregistered users shall be able to browse tutor listings.
  2. Unregistered users shall be able to view more details about the tutor listing.
  3. Unregistered users shall be able to search for a tutor.
  4. Unregistered users shall be able to filter tutor listings by price.
  5. Unregistered users shall be able to register an account using an SFSU email.
  6. Unregistered users shall be able to sign in to an existing account.
  7. Unregistered users shall be able to write a message but must sign in to send it.
  8. Unregistered users shall be able to fill out the tutor application form but must sign in to submit.
- Registered user
  1. Registered users shall inherit all previous privileges.
  2. Registered users shall be able to message tutors about their services.
  3. Registered users shall be able to submit a tutor application to create a new listing.
  4. Registered users shall be able to view their message(s) in the dashboard.
  5. Registered users shall be able to delete their message(s) in the dashboard.

6. Registered users shall be able to view their listing(s) in the dashboard.
7. Registered users shall be able to delete their listing(s) in the dashboard.

- Admin
    1. Admin users shall inherit all previous privileges.
    2. Admin users shall be required to verify all tutor listing applications before they are public.
    3. Admin users shall be able to remove users or listings from the website.
    4. Admin users shall be able to view and manage all user data.

URL to product: **https://eclipsesakura.online/**

2. Usability test plan for the selected function

1) **Test objectives:** This usability test evaluates our application's search functionality, focusing on its effectiveness in finding relevant tutor listings and measuring user satisfaction. The evaluation will gather feedback on the relevance of search results to user queries and chosen subjects, as the search feature is the backbone of our product's advertised services.
2) **Test background and setup:**
    a) **System setup and starting point:** The testers will need an electronic device with internet access, such as a computer, laptop, tablet, or mobile device. They will start on the website's landing page and access the search bar via the navigation bar at the top.
    b) **Hardware Requirements:** Testers will need a working electronic device with access to the internet.
    c) **Intended User:** Our product's intended users are SFSU students, whether they are seeking a tutor for a specific course or looking to offer their tutoring services.
    d) **URL:** **https://eclipsesakura.online/**
    e) **Test Environment:** The usability test can occur in any location with internet access. Ideally, no monitoring or training is necessary before testing. The usability

test simulates the role of a stressed student (who has some technological skills) with limited patience and who is a new user of the application.

3) **Usability Task description:** The specific instructions given to the tester prior to the testing and filling out our Likert survey will be as follows: Your task is to play the role of a stressed student with upcoming exams. Given a short list (about three items to not drain the tester's patience) of specific tutors and courses (related to some tutor, of course) to search for, you will open the website and then complete the following:
    - Search for a tutor by the course they teach.
    - Locate the desired tutor from the search results.
    - Modify the query if needed to find the desired tutor.
    - Repeat the process for the remaining items on the list.
4) **Plan for Evaluation of Effectiveness:** We plan to measure effectiveness by having the tester fill out a brief questionnaire after each search. The questionnaire is not part of the Likert scale and serves as a means to consolidate a more in-depth report or document the steps to recreate any issues. The questionnaire will ask the tester to provide the number of search attempts to find the tutor successfully, what queries they used, the time taken, and any frustrations, issues, or possible comments. A lower number of search queries indicates a more streamlined search process. Additionally, a success rate metric will then classify the percentage of successful searches within the predefined three queries and quantify effectiveness. Any patterns in failed searches or searches that require substantial effort could then easily be traced and identified for resolution.
5) **Plan for Evaluation of Efficiency:** We will measure efficiency by evaluating the time and effort testers spend completing each search task. As specified in the questionnaire above, testers will report how long it takes them to locate a specific tutor and the queries they use. We will document instances where testers need to modify their searches multiple times, as these may indicate inefficiencies or usability issues. To provide a comprehensive evaluation, we will analyze task completion times and calculate averages to determine the overall speed of navigating and using the search function. These data will help identify specific areas for improvement, such as query responsiveness and result relevance.

6) **Plan for Evaluation of User Satisfaction (Likert scale):** We will provide this at the very end after all the tests. Example:

Statement	**Strongly Disagree**	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
I was able to quickly find the tutor I was looking for using the search function.	☐	☐	☐	☐	☐
I was satisfied with the amount of information available about the tutor.	☐	☐	☐	☐	☐
I found the search experience user-friendly.	☐	☐	☐	☐	☐

There will also be a section for any additional comments that the tester has.

3. QA test plan and QA testing

1) **Test objectives:** The objective is to ensure our application's search functionality functions as expected, focusing on its effectiveness in finding the relevant tutor listings.

2) **Hardware and Software setup:**
    a) **Hardware:** An electronic device with internet access, like a laptop or desktop.
    b) **Software:** Google Chrome and Microsoft Edge.
    c) **URL:** https://eclipsesakura.online/
3) **Feature to test:** The primary feature under test is the **search bar functionality,** which allows users to find tutors based on input queries.

4) **QA Test plan:**

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test results (Chrome/Microsoft Edge)
1	Search with Valid Query	Verify the search returns relevant results for a valid query.	Query: "Math 100"	Relevant tutor profiles for "Math 100 " are displayed within 2-3 seconds.	Chrome: Pass Edge: Pass
2	Search for Non-Existent Subject	Verify the fallback message when no results are found.	Query: "Astrology"	Message: "We couldn't find your desired tutor listing, but here are some tutors that might interest you."	Chrome: Pass Edge: Pass
3	Search empty query	Verify the ability to see all listings when searching with an empty query.	Query: N/A	Displays 15 tutor listings (All tutor listings are currently in the database).	Chrome: Pass Edge: Pass

Google Chrome and Microsoft Edge test results:

Test #	Test Title	Google chrome result	Microsoft Edge result	Notes
1	Search with a valid query	PASS	PASS	Relevant tutor profiles (e.g., for "Math 100 Tutor") displayed within 2-3 seconds.
2	Search for nonexistent query	PASS	PASS	Correctly displayed the error message when the search results showed no listings found. This message appeared when searching for a subject not listed (e.g., "Astrology").
3	Search empty query	PASS	PASS	Searching with an empty query displays all tutor listings.

Test 1: Search with a valid query

Description: Verify the search returns relevant results based on the input in the search bar.

Input: Enter “Math 100”

Expected Output: 1 result containing “Math 100” in their tutor listing details.

Results (both Chrome & Edge): 1 result containing “Math 100” in their tutor listing details.

Test 2: Search with a query with no results

Description: Verify the search returns relevant results based on the input in the search bar.

Input: Enter “Astrology”.

Expected Output: We couldn't find your desired tutor listing, but here are some tutors that might interest you. Please refine your search or select a subject for a more precise result.

Results (both Chrome & Edge): Displayed “We couldn't find your desired tutor listing, but here are some tutors that might interest you. Please refine your search or select a subject for a more precise result.” and also displayed other tutors the user might be interested in.

Test 3: Search with an empty query

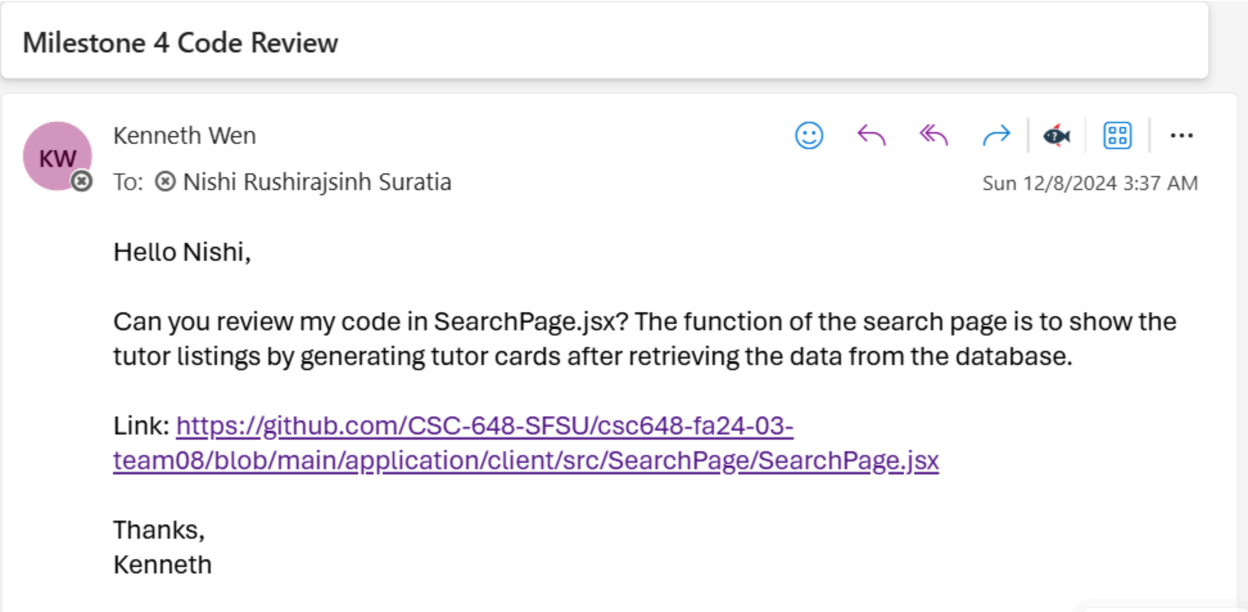
Description: Pressing the search button without inputting anything in the search bar.

Input: Nothing. Just press the search button.

Expected Output: Displays 15 tutor listings.

Results (both Chrome & Edge): Displays 15 tutor listings (All tutor listings currently in the database).

4. Peer Code Review:



Milestone 4 Code Review

KW

Kenneth Wen

To: Nishi Rushirajsinh Suratia

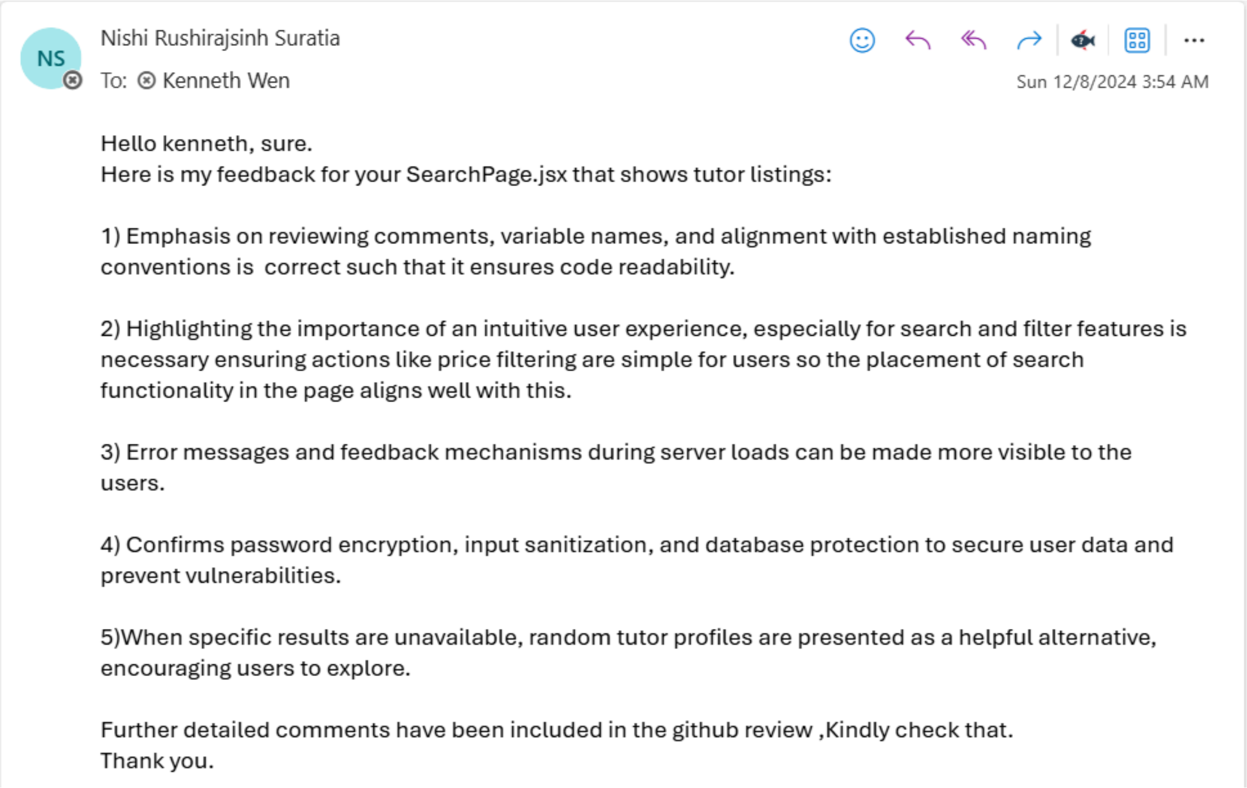
Sun 12/8/2024 3:37 AM

Hello Nishi,

Can you review my code in SearchPage.jsx? The function of the search page is to show the tutor listings by generating tutor cards after retrieving the data from the database.

Link: https://github.com/CSC-648-SFSU/csc648-fa24-03-team08/blob/main/application/client/src/SearchPage/SearchPage.jsx

Thanks,
Kenneth



NS

Nishi Rushirajsinh Suratia

To: Kenneth Wen

Sun 12/8/2024 3:54 AM

Hello kenneth, sure.
Here is my feedback for your SearchPage.jsx that shows tutor listings:

1) Emphasis on reviewing comments, variable names, and alignment with established naming conventions is correct such that it ensures code readability.

2) Highlighting the importance of an intuitive user experience, especially for search and filter features is necessary ensuring actions like price filtering are simple for users so the placement of search functionality in the page aligns well with this.

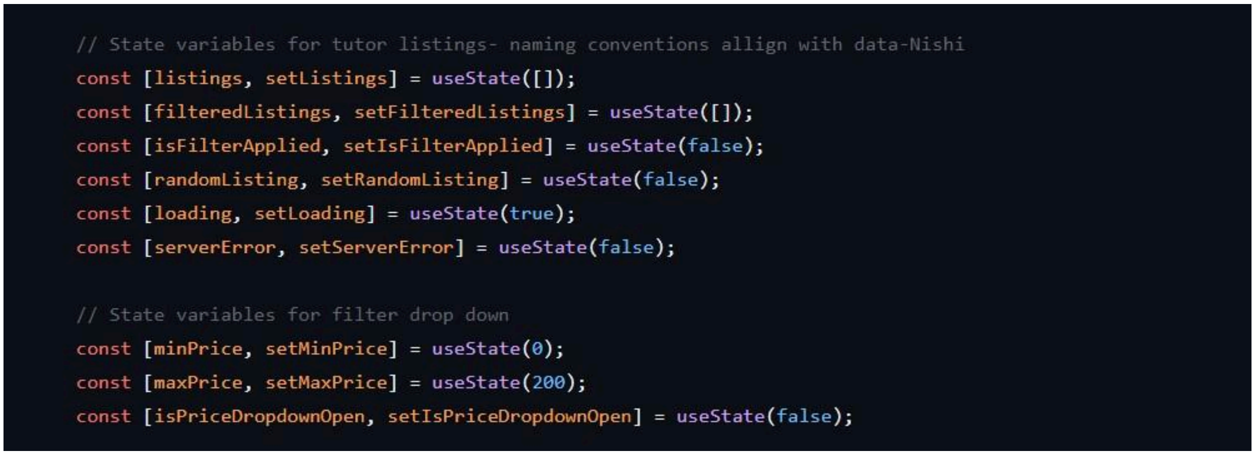
3) Error messages and feedback mechanisms during server loads can be made more visible to the users.

4) Confirms password encryption, input sanitization, and database protection to secure user data and prevent vulnerabilities.

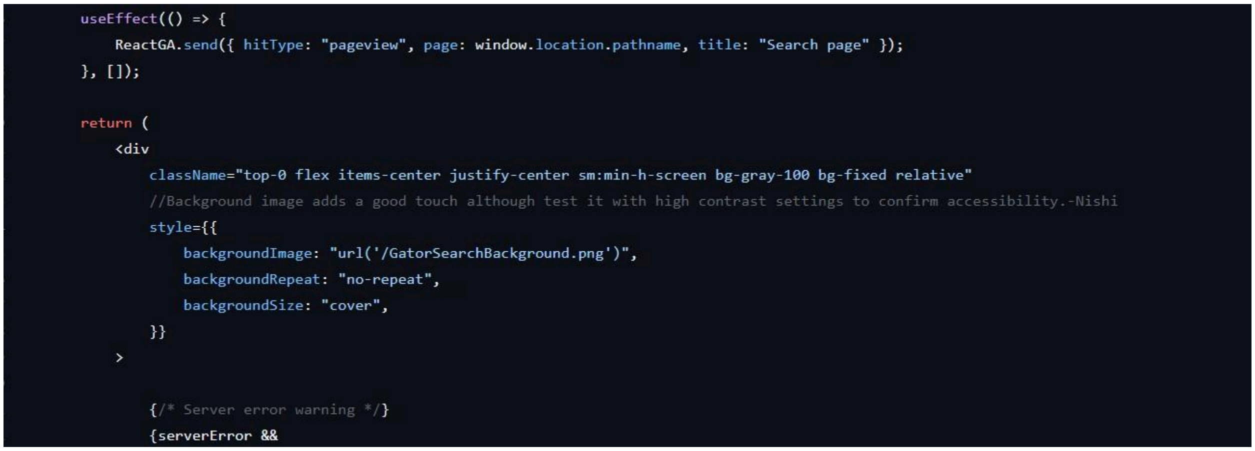
5)When specific results are unavailable, random tutor profiles are presented as a helpful alternative, encouraging users to explore.

Further detailed comments have been included in the github review ,Kindly check that.
Thank you.

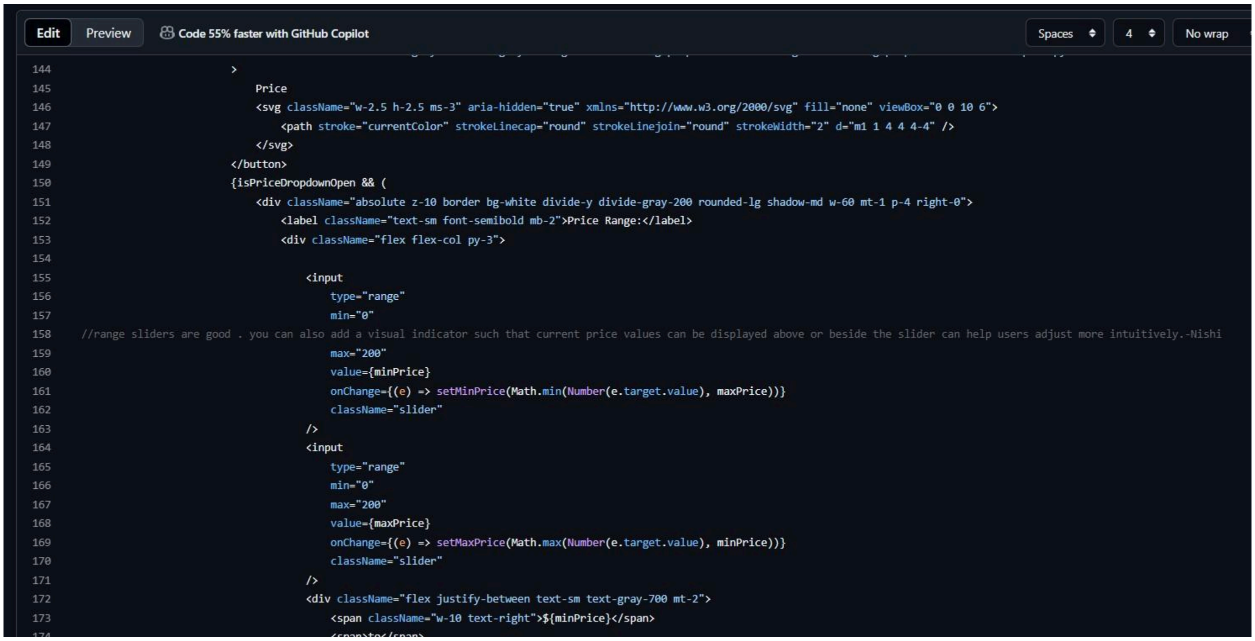
Github code review for SearchPage.jsx:



```
// State variables for tutor listings- naming conventions allign with data-Nishi
const [listings, setListings] = useState([]);
const [filteredListings, setFilteredListings] = useState([]);
const [isFilterApplied, setIsFilterApplied] = useState(false);
const [randomListing, setRandomListing] = useState(false);
const [loading, setLoading] = useState(true);
const [serverError, setServerError] = useState(false);

// State variables for filter drop down
const [minPrice, setMinPrice] = useState(0);
const [maxPrice, setMaxPrice] = useState(200);
const [isPriceDropdownOpen, setIsPriceDropdownOpen] = useState(false);
```



```
useEffect(() => {
    ReactGA.send({ hitType: "pageview", page: window.location.pathname, title: "Search page" });
}, []);

return (
    <div
        className="top-0 flex items-center justify-center sm:min-h-screen bg-gray-100 bg-fixed relative"
        //Background image adds a good touch although test it with high contrast settings to confirm accessibility.-Nishi
        style={{
            backgroundImage: "url('/GatorSearchBackground.png')",
            backgroundRepeat: "no-repeat",
            backgroundSize: "cover",
        }}
    >

        {/* Server error warning */}
        {serverError &&
```



```
                        >
                            Price
                            <svg className="w-2.5 h-2.5 ms-3" aria-hidden="true" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 10 6">
                                <path stroke="currentColor" strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="m1 1 4 4 4-4" />
                            </svg>
                        </button>
                        {isPriceDropdownOpen && (
                            <div className="absolute z-10 border bg-white divide-y divide-gray-200 rounded-lg shadow-md w-60 mt-1 p-4 right-0">
                                <label className="text-sm font-semibold mb-2">Price Range:</label>
                                <div className="flex flex-col py-3">

                                    <input
                                        type="range"
                                        min="0"
//range sliders are good . you can also add a visual indicator such that current price values can be displayed above or beside the slider can help users adjust more intuitively.-Nishi
                                        max="200"
                                        value={minPrice}
                                        onChange={(e) => setMinPrice(Math.min(Number(e.target.value), maxPrice))}
                                        className="slider"
                                    />
                                    <input
                                        type="range"
                                        min="0"
                                        max="200"
                                        value={maxPrice}
                                        onChange={(e) => setMaxPrice(Math.max(Number(e.target.value), minPrice))}
                                        className="slider"
                                    />
                                    <div className="flex justify-between text-sm text-gray-700 mt-2">
                                        <span className="w-10 text-right">${minPrice}</span>
```

5. Self-check on best practices for security

Yes, we have more than half a page of security self-check, but all is considered and implemented.

Asset to be protected	**Types of possible/expected attacks**	**Consequence of security breach**	**Our Strategy to Mitigate/Protect the Asset**
User Passwords	Brute force attacks, data breaches, unauthorized access	Exposure of sensitive user accounts, unauthorized logins	Encrypt passwords in the database using a secure hashing algorithm like bcrypt. Server-sided verifications.
User Personal Data	Data breaches, phishing attacks	Leakage of private user data, identity theft	Ensure HTTPS is used for data transfer. Implement access control for sensitive user data at both the backend and database layers. Encrypt cookies using JWT.
Tutor Images/Resumes	Unauthorized downloads, tampering with files	Misuse of personal branding material, spreading false data	Validate uploaded files for type and size before saving them. Use signed URLs for accessing sensitive documents.
Search Bar Input	SQL Injection, XSS (Cross-Site Scripting)	Database Corruption, malicious script execution	Validate search bar input to accept only alphanumeric characters of up to 40 characters. Use parameterized queries to prevent SQL injection. Escape special characters to avoid XSS.
Registration Email	Spam accounts, invalid email submissions	System clutter, reduced credibility, inability to contact	Using a regex pattern, validate email addresses by including “sfsu.edu” at the end of the registration. Reject invalid emails before processing.
Acceptance of Terms	Users bypassing terms, lack of accountability	Legal vulnerabilities, user disputes	Ensure the acceptance of the terms checkbox is mandatory before registration.

6. Self-check of the adherence to original Non-functional specs – performed by team leads.

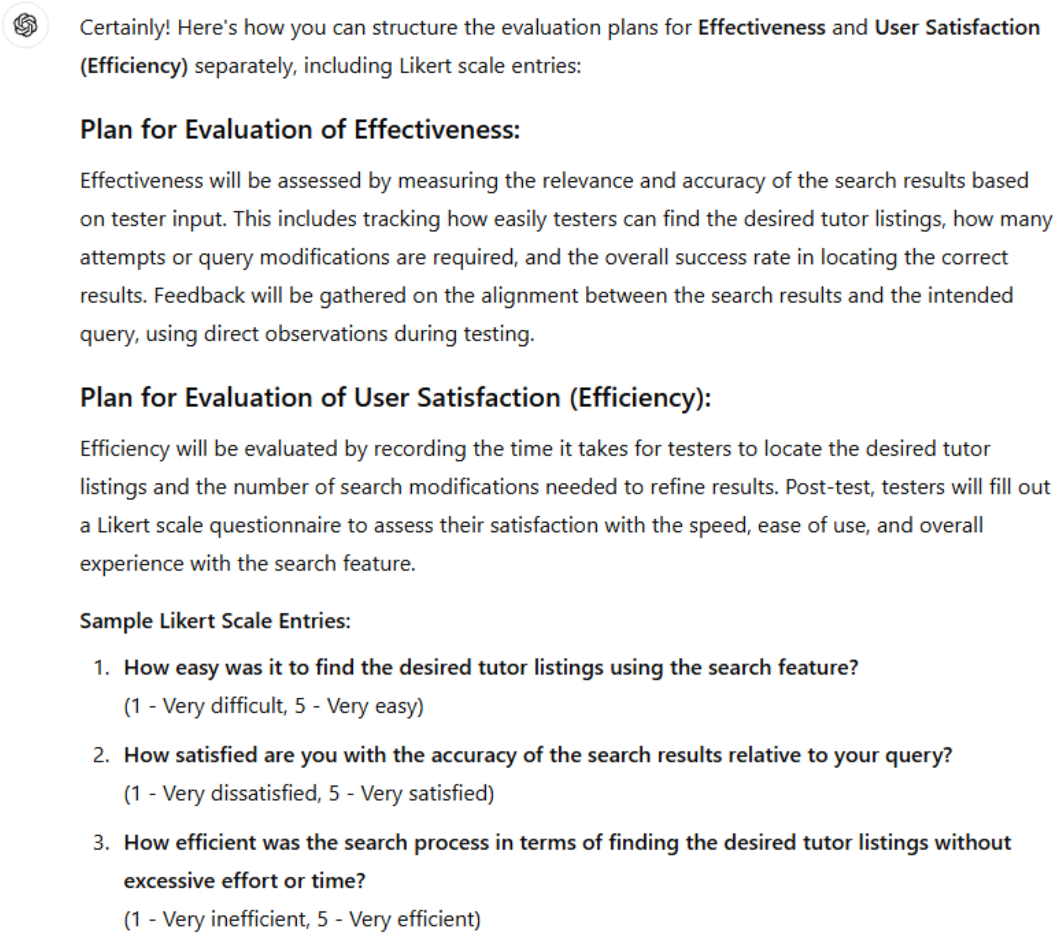
1. Application shall be developed, tested, and deployed using tools and servers approved by Class CTO and as agreed in M0 **[DONE]**
2. Application shall be optimized for standard desktop/laptop browsers, e.g., must render correctly on the two latest versions of two major browsers **[DONE]**
3. All or selected application functions shall render well on mobile devices (no native app to be developed) **[DONE]**
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 **[DONE]**
8. The language used shall be English (no localization needed) **[DONE]**
9. Application shall be very easy to use and intuitive **[DONE]**
10. Application shall follow established architecture patterns **[DONE]**
11. Application code and its repository shall be easy to inspect and maintain **[DONE]**
12. Google Analytics shall be used **[DONE]**
13. No e-mail clients shall be allowed. Interested users can only message 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 **[DONE]**
16. Media formats shall be standard as used in the market today **[DONE]**
17. Modern SE processes and tools shall be used as specified in the class, including collaborative and continuous SW development and GenAI tools **[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 2024. For

Demonstration Only” at the top of the WWW page Nav bar. (This is important so as not to confuse this with a real application). **[DONE]**

7. Use of genAI tools like ChatGPT and Copilot (mandatory)

genAI tool	ChatGPT-4o
Task	Structure/idea on usability evaluation plan for effectiveness and efficiency.
Useful rating	Medium

- Prompt: Can you provide a structure/idea on how I can create a usability evaluation plan for effectiveness and efficiency for my tutoring website?



Certainly! Here's how you can structure the evaluation plans for **Effectiveness** and **User Satisfaction (Efficiency)** separately, including Likert scale entries:

Plan for Evaluation of Effectiveness:

Effectiveness will be assessed by measuring the relevance and accuracy of the search results based on tester input. This includes tracking how easily testers can find the desired tutor listings, how many attempts or query modifications are required, and the overall success rate in locating the correct results. Feedback will be gathered on the alignment between the search results and the intended query, using direct observations during testing.

Plan for Evaluation of User Satisfaction (Efficiency):

Efficiency will be evaluated by recording the time it takes for testers to locate the desired tutor listings and the number of search modifications needed to refine results. Post-test, testers will fill out a Likert scale questionnaire to assess their satisfaction with the speed, ease of use, and overall experience with the search feature.

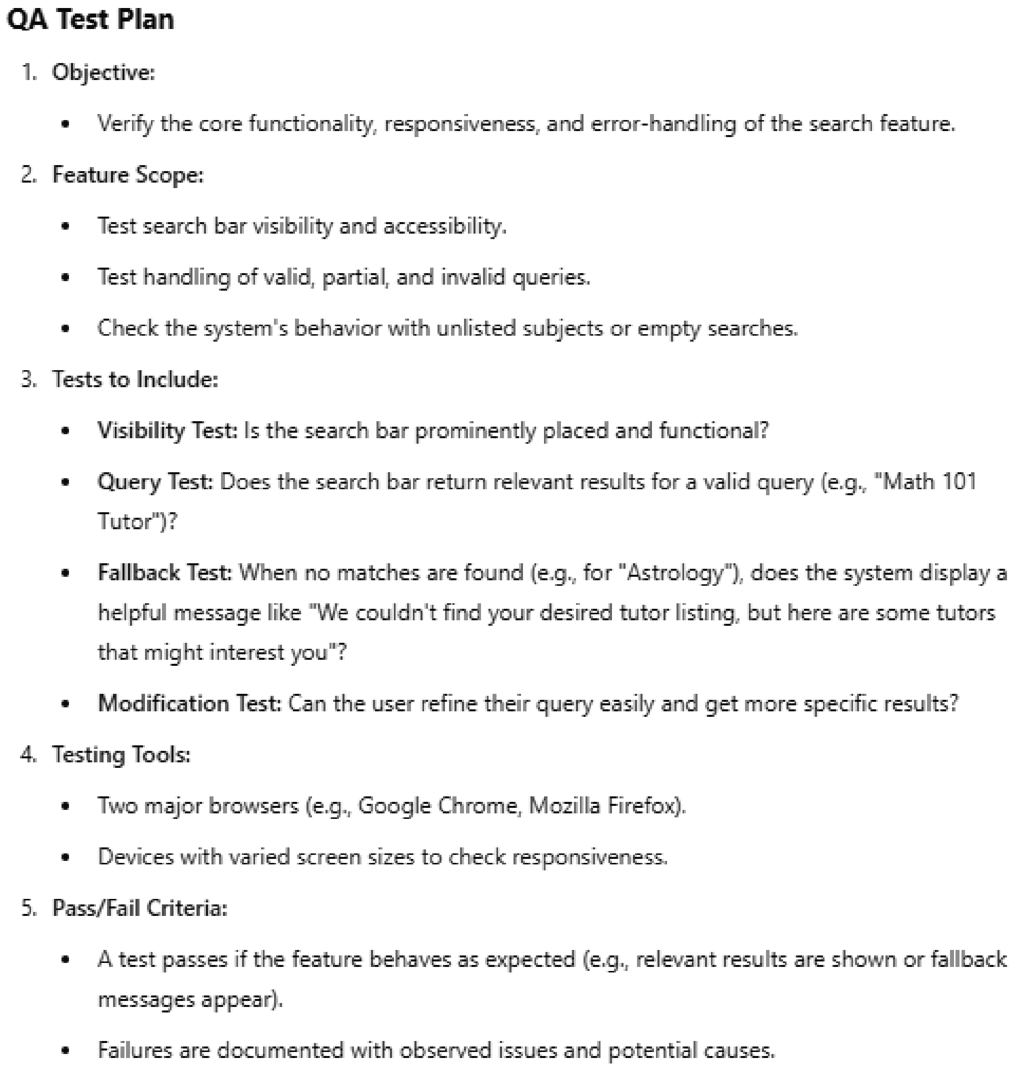
Sample Likert Scale Entries:

1. **How easy was it to find the desired tutor listings using the search feature?**
   (1 - Very difficult, 5 - Very easy)
2. **How satisfied are you with the accuracy of the search results relative to your query?**
   (1 - Very dissatisfied, 5 - Very satisfied)
3. **How efficient was the search process in terms of finding the desired tutor listings without excessive effort or time?**
   (1 - Very inefficient, 5 - Very efficient)

For this milestone, one way we use genAI, particularly ChatGPT, is to generate ideas for the usability test plan. It was helpful in providing insightful ideas but could have been more accurate. However, ChatGpt inspired us to create a separate questionnaire besides the Likert scale to acquire more in-depth data about the tester's number of search attempts to find the tutor successfully, what queries they used, the time taken, and any frustrations, issues, or possible comments. Overall, ChatGpt provided valuable insights but required a brute-force approach to prompting. It would repeatedly give the same idea and offer no new insight besides empty fluff.

genAI tool	ChatGPT-4o
Task	Ideas for QA test planning
Useful rating	Medium

- Prompt:
  "Can you create a usability and QA test plan for a website's search functionality that caters to stressed students searching for tutors?"



QA Test Plan

1. **Objective:**
   - Verify the core functionality, responsiveness, and error-handling of the search feature.
2. **Feature Scope:**
   - Test search bar visibility and accessibility.
   - Test handling of valid, partial, and invalid queries.
   - Check the system's behavior with unlisted subjects or empty searches.
3. **Tests to Include:**
   - **Visibility Test:** Is the search bar prominently placed and functional?
   - **Query Test:** Does the search bar return relevant results for a valid query (e.g., "Math 101 Tutor")?
   - **Fallback Test:** When no matches are found (e.g., for "Astrology"), does the system display a helpful message like "We couldn't find your desired tutor listing, but here are some tutors that might interest you"?
   - **Modification Test:** Can the user refine their query easily and get more specific results?
4. **Testing Tools:**
   - Two major browsers (e.g., Google Chrome, Mozilla Firefox).
   - Devices with varied screen sizes to check responsiveness.
5. **Pass/Fail Criteria:**
   - A test passes if the feature behaves as expected (e.g., relevant results are shown or fallback messages appear).
   - Failures are documented with observed issues and potential causes.

We also use generative AI, particularly ChatGPT, to brainstorm ideas for our QA test plan. This tool provided valuable insights that enriched our plan, including suggestions for query testing, fallback testing, modification testing, cross-browser testing, and defining clear pass/fail criteria. While some recommendations, like visibility testing, were redundant, the overall contribution was highly beneficial.