SW Engineering CSC 648-848 Fall 2024

GatorTutor

Team 6

Team Lead, Alex Hoff - (ahoff2@mail.sfsu.edu)
Backend Lead 1, Dylan Faulder - (dfaulder@mail.sfsu.edu)
Backend Lead 2, Austin Ng - (Ang@sfsu.edu)
FrontEnd Lead, Dalan Choy - (dchoy3@mail.sfsu.edu)
Github Master, Jack Richards - (jrichards7@sfsu.edu)

Milestone 4

History

Date Submitted:	**December 12th, 2024**

Product Summary

Product Name and Description: "*GatorTutor*"

A site for San Francisco State University students to find academic help from other students who've been in their position. They are able to find and search for tutors that specialize in the courses they need assistance in.

GatorTutor is a tutoring service created by San Francisco State University students for SFSU students. We aim to provide a place that facilitates the matching between all SFSU students with their best fit SFSU tutor. We feel that the dividing factor between a successful student and a struggling student is gaps in their knowledge and our goal is to bridge that gap. Our service helps match tutors with experience in those subjects and the students seeking assistance. Since our application is institution specific, our tutors are able to provide a level of service unique to SFSU that foreign services can not compete with.

Itemized List of P1 Functions:

Browsing: Users shall be able to browse and easily navigate the website and most of what it has to offer.
Searching: Users shall be able to use the search bar in order to search for SFSU-specific classes and professors to find hyper-relevant material.
Posting: Users shall be able to create a tutor profile and advertise their skills on the website.
Dashboard: Users shall have a functioning dashboard that displays their postings and messages received.
Messaging: One way messaging system from student to tutor.
Approving Post: Site Admin shall be able to delete inappropriate items or users based on the criteria of the post.
Deployment URL:
https://gatortutor.net/

Usability Test

1. **Test Objectives**:
   The objective of this usability test is to evaluate how easily users can create a tutor post on GatorTutor. This function is critical to the platform's success, as it enables tutors to share their information and students to find suitable tutors. Ensuring a seamless posting experience is essential for maintaining the platform's functionality and user satisfaction.

2. **Test Background and Setup**:

System Setup
A standard personal computer with a modern operating system capable of running web browsers such as Google Chrome and Mozilla Firefox. No specialized hardware or software is required beyond these browsers and a stable internet connection to access the specified URL.

Intended Users
GatorTutor is intended for students and tutors with varying levels of experience in navigating online with a browser. This means that users with the minimal experience should still be able to navigate through the website without any difficulties. The navigation should feel intuitive to people of all levels of experience.

URL of the System
Testing of tutor post creation functions on GatorTutor will be conducted on https://gatortutor.net/

Test Environment
The test will take place in a home setting with no cameras present. Additionally, training will not be done because GatorTutor aims to have a simplistic navigation experience.

3. **Usability Task Description**:
    1. Navigate to Create Tutor Post section
    2. Fill out the Basic Information section with the following details
        a. Set the Name to "John Doe"
        b. Set the Subject to "Finance"
        c. Set the Hourly Rate to "$20"
        d. Set Contact Information to "test@email.com"
        e. Set Bio to "I successfully completed and filled out the bio section."
    3. Set the availability to Monday and Friday
    4. Upload a given sample MP4, PDF CV, and PNG profile picture in the Additional Documents section.
    5. Submit the post

4. **Plan for Evaluation of Effectiveness**:
Effectiveness will be determined by the number of people who were able to complete the task within a specific amount of time. This will be compared to the number of people who weren't able to successfully complete the task. The second observation that will be made is looking at each field to see if it was filled out correctly. The fields must be filled out exactly as the task description mentions and must not be missing. Lastly, viewing the

post to observe if it was submitted and uploaded successfully will be taken into consideration.

5. **Plan for Evaluation of Efficiency**:
   Efficiency will be determined by two main factors. The first factor will be how long it took to complete the task. The tester's skill level will be taken into consideration when observing the completion time. The second factor that will be observed is the number of clicks it took to create and submit the post.

Major Assets & Their Protection

- User Credentials
  - Threats: Password theft, unauthorized access
  - Protection: Passwords encrypted using bcrypt with salt rounds of 10, secure HTTP-only cookies for sessions
- User Personal Data (SFSU emails, profiles)
  - Threats: Data breach, identity theft
  - Protection: Data access restricted through authentication middleware, sensitive data filtered from API responses
- Tutor-Student Messages
  - Threats: Unauthorized message access, impersonation
  - Protection: Message access verified against user session, sender/recipient validation on server

Passwords are encrypted in database using bcrypt

```
const hashedPassword = await bcrypt.hash(data.password, 10);
```

Search Bar Input (api also restricts > 40 char)

```
maxLength={40}
```

SFSU Email Validation

```
email: z.string()
  .email("Invalid email format")
  .endsWith("@sfsu.edu", "Must be an SFSU email address")
```

Terms Acceptance

```
acceptTerms: z.boolean().refine((val) => val === true, {
  message: "You must accept the terms and conditions"
})
```

6. **Plan for Evaluation of User Satisfaction**:
    1. Creating a tutor post was a straightforward process.
       1) Strongly Disagree 2) Disagree 3) Neutral 4) Agree 5) Strongly Agree

    2. The fields were easy to understand and fill out.
       1) Strongly Disagree 2) Disagree 3) Neutral 4) Agree 5) Strongly Agree

    3. I am satisfied with the experience of creating a tutor post.
       1) Strongly Disagree 2) Disagree 3) Neutral 4) Agree 5) Strongly Agree

QA Test Plan and QA Testing

1. Test Objective,

   The Goal of these tests are to verify we have a working create tutor post functions.

2. Specify all hardware and software requirements, including the URL

   Basic hardware capability, enough to give access to basic web browser abilities.

ADD TUTOR URL

3. Feature to be tested

   The create tutor post function, along with working Title, Name,Hourly Rate, Bio and Availability.

4. QA Test plan Table

Test #	1	2	3
Test Title	Test of successful tutor post creation	Test of successful tutor post creation	Test of successful tutor post creation
Test Description	Testing if when given input that the create tutor post functions successfully create a working tutor	Testing if when given input that the create tutor post functions successfully create a working tutor post	Testing if when given input that the create tutor post functions successfully create a working tutor post visible to users.

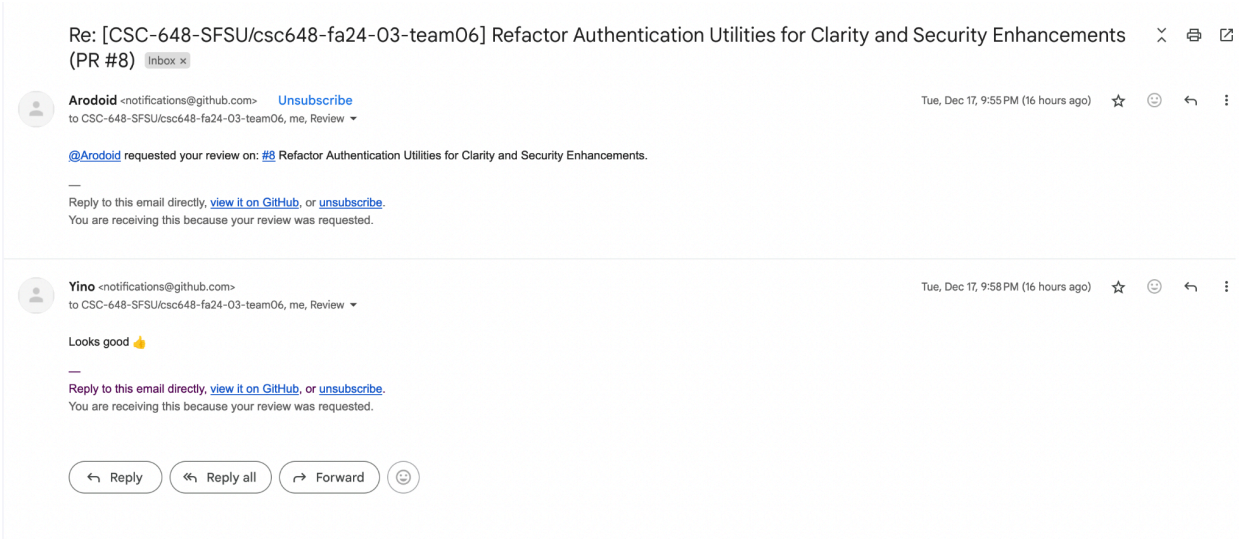
	post visible to users.	visible to users.	
Test Input	Title: Native Japanese tutor Hello I am a Native Japanese Speaker who specializes in teaching Japanese to students Name: Test Name #1 Hourly Rate: 21.00 Subject: Japanese Bio: Hello I am a Native Japanese Speaker who specializes in teaching Japanese to students Availability: Monday Tuesday Wednesday	Title: Highly Skilled Computer Scientist Name: Test Name #2 Hourly Rate: 100 Subject: Computer Science Bio: This is a test for checking if the bio will produce exactly what is given 18001222 Availability: Monday Tuesday Wednesday Thursday Friday Saturday Sunday	Title: 123456789 Name: Test Name #3 Hourly Rate: 1 Subject: Art History Bio: Art History is a interesting major that expands one's knowledge of art throughout history Availability: Saturday Sunday
Expected Output	Title: Native Japanese tutor Hello I am a Native Japanese Speaker who specializes in teaching Japanese to students Name: Test Name #1 Hourly Rate: 21.00 Subject: Japanese Bio: Hello I am a Native Japanese Speaker who specializes in teaching Japanese to students Availability: Monday Tuesday Wednesday	Title: Highly Skilled Computer Scientist Name: Test Name #2 Hourly Rate: 100 Subject: Computer Science Bio: This is a test for checking if the bio will produce exactly what is given 18001222 Availability: Monday Tuesday Wednesday Thursday Friday Saturday Sunday	Title: 123456789 Name: Test Name #3 Hourly Rate: 1 Subject: Art History Bio: Art History is a interesting major that expands one's knowledge of art throughout history Availability: Saturday Sunday
Test results:	Correct output given	Correct output given	Correct output given

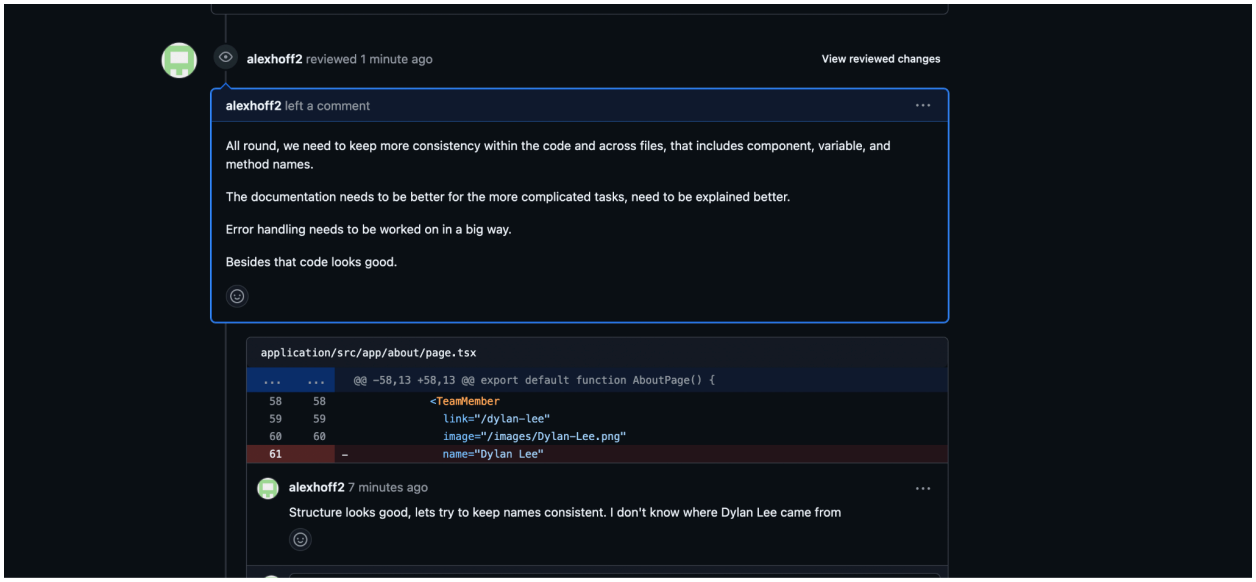
Browser Test	Chrome: Successful:PASS Firefox: Successful:PASS	Chrome: Successful:PASS Firefox: Successful:PASS	Chrome: Successful:PASS Firefox: Successful:PASS

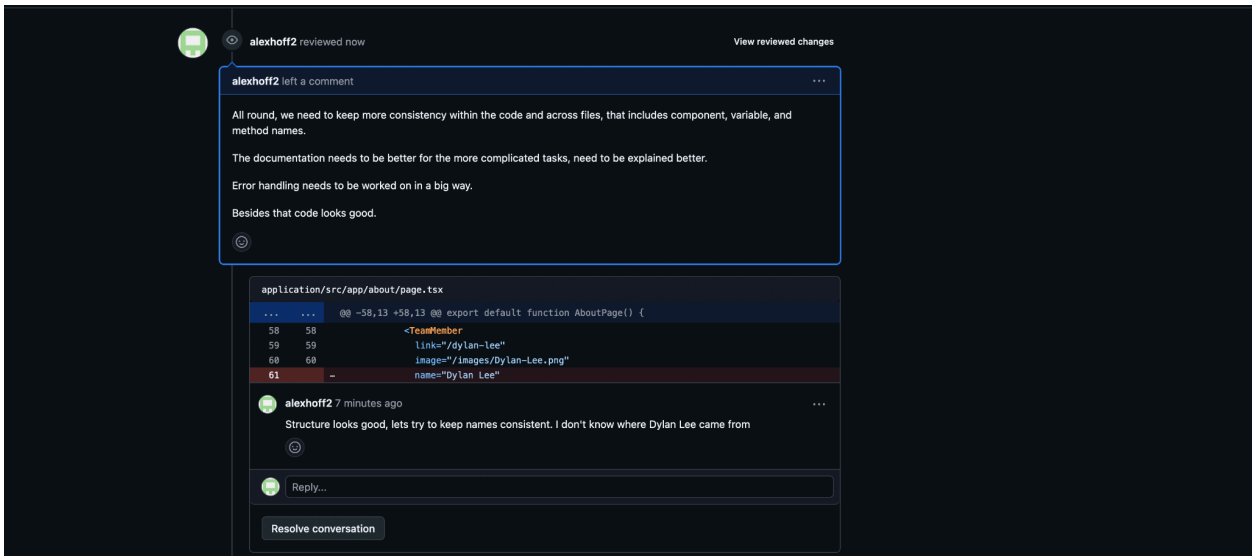
5. Perform tests on two different browsers
Brave (Chromium) and Firefox

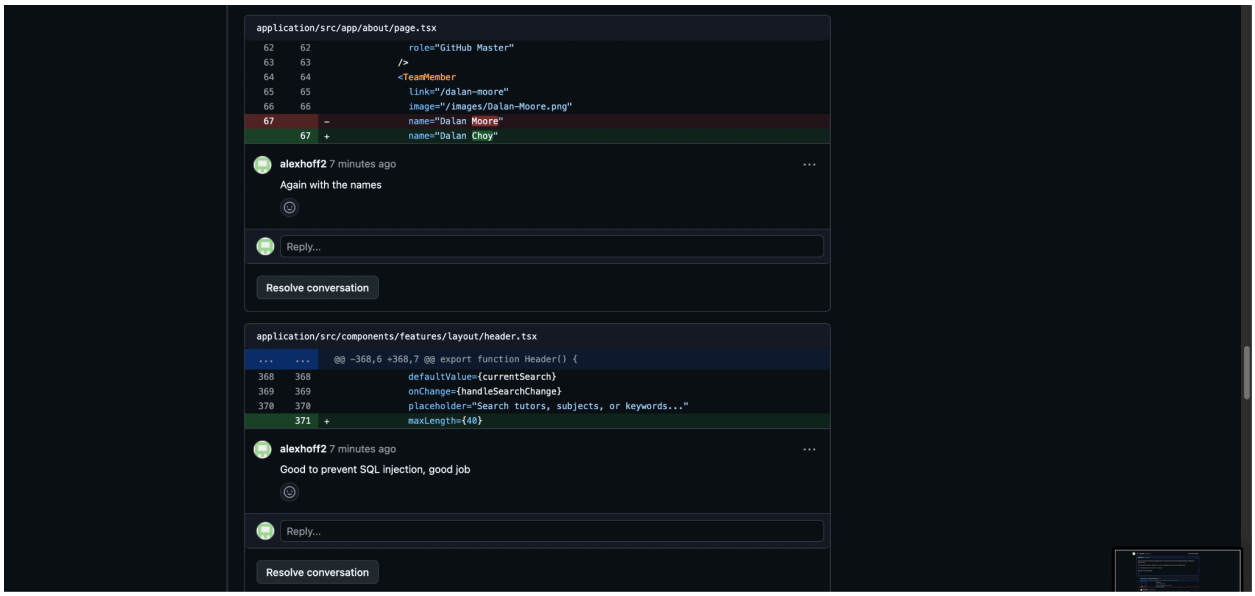
Peer Review Code

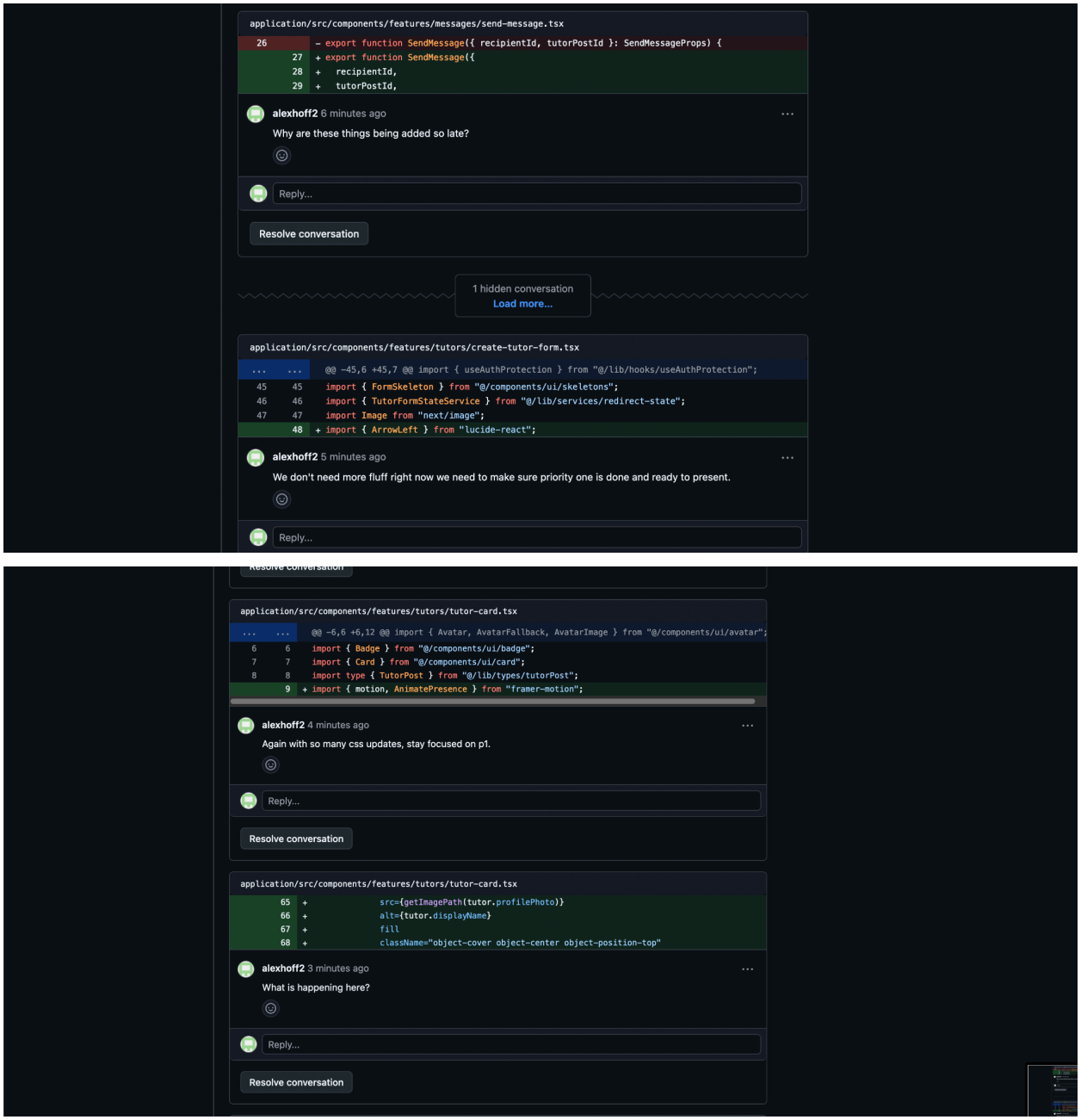
This peer review was done with Dylan's and Jack's code reviewed by Alex.




alexhoff2 reviewed 1 minute ago
View reviewed changes
alexhoff2 left a comment
All round, we need to keep more consistency within the code and across files, that includes component, variable, and method names.
The documentation needs to be better for the more complicated tasks, need to be explained better.
Error handling needs to be worked on in a big way.
Besides that code looks good.
application/src/app/about/page.tsx
@@ -58,13 +58,13 @@ export default function AboutPage() {
58 58 <TeamMember
59 59 link="/dylan-lee"
60 60 image="/images/Dylan-Lee.png"
61 - name="Dylan Lee"
alexhoff2 7 minutes ago
Structure looks good, lets try to keep names consistent. I don't know where Dylan Lee came from


alexhoff2 reviewed now
View reviewed changes
alexhoff2 left a comment
All round, we need to keep more consistency within the code and across files, that includes component, variable, and method names.
The documentation needs to be better for the more complicated tasks, need to be explained better.
Error handling needs to be worked on in a big way.
Besides that code looks good.
application/src/app/about/page.tsx
@@ -58,13 +58,13 @@ export default function AboutPage() {
58 58 <TeamMember
59 59 link="/dylan-lee"
60 60 image="/images/Dylan-Lee.png"
61 - name="Dylan Lee"
alexhoff2 7 minutes ago
Structure looks good, lets try to keep names consistent. I don't know where Dylan Lee came from
Reply...
Resolve conversation


application/src/app/about/page.tsx
role="GitHub Master"
/>
<TeamMember
link="/dalan-moore"
image="/images/Dalan-Moore.png"
- name="Dalan Moore"
+ name="Dalan Choy"
alexhoff2 7 minutes ago
Again with the names
Reply...
Resolve conversation
application/src/components/features/layout/header.tsx
@@ -368,6 +368,7 @@ export function Header() {
defaultValue={currentSearch}
onChange={handleSearchChange}
placeholder="Search tutors, subjects, or keywords..."
+ maxLength={40}
alexhoff2 7 minutes ago
Good to prevent SQL injection, good job
Reply...
Resolve conversation



Non-Functional Specs Self-Check

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 - **IN PROGRESS**

13. No email 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 - **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. (Important so as to not confuse this with a real application). - **IN PROGRESS**

GenAI Usage

***CHATGPT-4o mini* WAS USED FOR ALL TASKS LISTED BELOW**

Code Review Assistance
Usefulness: High
How We Used It: ChatGPT was used to assist in conducting code reviews by generating suggestions for improving variable naming consistency, function comments, and error-handling checks. For example, when reviewing our messaging feature, ChatGPT suggested adding input validation and improving feedback messages for failed message sends.

Prompt Example:"Review this JavaScript function for potential bugs and suggest improvements for better readability and performance."

Bug Identification & Debugging
Usefulness: Medium
How We Used It: GitHub Copilot was used during the debugging phase to detect logical issues. It helped catch edge cases in our availability filtering system and suggested fixes for incomplete SQL queries in our Prisma models.

Prompt Example:"Fix SQL query that fetches tutor availability from the database based on matching criteria."

UI/UX Recommendations
Usefulness: Medium
How We Used It: ChatGPT provided recommendations for improving usability in our Create Tutor Post form. It suggested adding placeholder text and form validation messages to enhance user experience.

Prompt Example:"Suggest improvements for a React form collecting tutor registration data."

Documentation Assistance
Usefulness: High

How We Used It: ChatGPT helped draft clear, concise comments for complex functions, such as the tutor card rendering logic. This saved time in documenting intricate logic in our codebase.

Prompt Example:"Explain how this React component renders tutor profiles with interactive effects."

Usability Testing Support
Usefulness: Low
How We Used It: ChatGPT was consulted for creating usability test scenarios and evaluation metrics. While some suggestions were relevant, others were too generic for our specific needs.

Prompt Example:"Generate usability testing scenarios for an online tutor-matching platform."

Additional Insights:

Best Use Cases: We found ChatGPT particularly useful for code reviews and generating documentation, which helped maintain high development standards.

Challenges: Some prompts required rephrasing to receive accurate responses, and specific technical queries occasionally yielded incomplete answers.

Conclusion:

Using GenAI tools in Milestone 4 proved beneficial, particularly in code reviews, debugging, and documentation. These tools supported our development process, improved our codebase, and saved significant time while allowing us to focus on core development tasks.