SW Engineering CSC 648/848 Fall 2023
WiseGATOR
Team 6

Role	Team Members
Team Lead	Joaquin Warren jwarren3@mail.sfsu.edu
Github Master	Ronnie Huang
Front-end Lead	Karl Moreno
Front-end Developer	Darien Banuelos
Back-end Lead	Sean Ryan
Back-end Developer	Philip Karnatsevich

Milestone 4

12/13/23

History Table

Date of Submitted	12/13/23
Date Revised	

1) Product Summary

WiseGATOR (Guidance And Tutoring Outreach Resource)

Our application aims to quickly connect SFSU students to SFSU tutors and SFSU students interested in becoming tutors to paid mentorship roles. Students can get 1 on 1, specific help with homework assignments, coursework, study guides, and more at a reasonable, cost-efficient rate. This is also a great opportunity for SFSU students that excel in their courses to make money by becoming a mentor and offering tutoring services on our site to other SFSU students. It's a one stop shop for any SFSU student seeking or offering expert help. Our goal is to streamline the process of finding tutoring, by allowing students to filter by subject and by courses offered at SFSU. This way, students are learning from one another in the same courses, and they do not have to rely on outdated and irrelevant internet forums or paid services from people that have never even been in that class. What makes our application so unique is that it is exclusive to San Francisco State University, whether you want to find a tutor or tutor others in any of the courses offered at SFSU. Searching by courses and subjects will optimize the experience for students seeking tutoring services, and SFSU tutors will be able to cater their service directly to SFSU students.

1. Unregistered users shall be able to have access to view all tutors
2. Unregistered users shall be able to view all available tutors as default
3. Unregistered users shall be able to filter tutors based on topic/subject e.g. MATH, ENG
4. Unregistered users shall be able to narrow search results by typing in search bar
5. Unregistered users shall be able to register
6. Unregistered users shall be able to apply to become a tutor, and will be prompted to register or log in
7. Registered users shall be able to apply to be a tutor, submitting their information
8. Registered users shall be able to upload a video
9. Registered users shall be able to upload a photo
10. Registered users shall be able to upload their CV, Flyer, or Resume
11. Registered users shall be able to log in
12. Registered users shall be able to log out of their account
13. Registered users shall be able to message a tutor of their choice
14. Registered users shall be able to access their messages and tutor posts via dashboard
15. Registered users shall be prompted to choose what topic(s) they tutor when applying as tutor
16. Registered users shall be able to write information about themselves inside of the description field
17. Admin shall be able to view database
18. Admin shall be required to approve registered users to become tutors before they go live
19. Admin shall be able to reject user posts before they are published

Link to Website: http://ec2-3-14-250-152.us-east-2.compute.amazonaws.com:3000/

2) Usability test plan for selected function:

Test Objectives: We will be writing a usability test plan for the searching functionality of our application. We will test the ability to find relevant tutors because it will be one of two main services offered through our application. Finding tutors is a feature that both students and students that want to offer tutoring may both need, since tutors can also be tutored by other students in subjects or courses they need help with.

Test Background and Setup: To set up our test, each tester will need an end system with an interface and the ability to input. Some examples are a smartphone and its screen or a laptop with a screen and keyboard. Each tester will begin at our website's landing page, what we refer to as the homepage. This is where users are to perform the intended test of searching for a tutor.

Our intended users are students at San Francisco State University, not specifically in any area of study. We want to offer our services to as many students as we can, because it increases the chances of students applying for tutor jobs that have good experience to help other students. This means that tutors get more business, making more money from using our website.

Our users will start with this link, and open it on their preferred internet browser: http://ec2-3-14-250-152.us-east-2.compute.amazonaws.com:3000/

Users will conduct this usability test from any location they'd like that has a reliable connection to the internet. They can do it from home or on campus at San Francisco State University. There will not be any cameras or other ways to monitor students as they test our website's searching feature other than a team member to proctor. Their purpose is not to help but just to count clicks and keep track of time. There will not be any required training for the students since they are our primary intended users and are quite familiar with simple search bar interfaces. Our search bar is intentionally not intricate so that it is very friendly for all future users, whether tech savvy or not.

Plan for Evaluation of Effectiveness: In order to measure effectiveness, we would need some figures to help us see if our usability is effective. The main way to do this is to see if a user was able to search for a tutor and find one relevant to their search. If the user was able to complete the task, or in other words, find a tutor, we take the quantity of successful searches and divide it by unsuccessful searches. This will give us a percent of users that found the search functionality effective. A good way to validate search effectiveness cases is to give them the name of a tutor to find, so we don't count cases where someone searched for a non-existent tutor, an impossible task. Given the tutor exists, we can conduct that test with multiple tutors to get a more accurate percentage of successful tests.

Plan for Evaluation of Efficiency: To measure efficiency, we can measure two different quantitative factors being number of clicks, and time. To be more specific, the number of clicks can be recorded for each user up until they find the target tutor. The time taken will be exactly

the minutes and seconds taken to reach the result, which is to see the tutor that we assigned them to search for. The time taken and the number of clicks can both be averaged to get a sense of how efficient our search feature is. This test is best conducted on users that are new to the website and haven't used it or seen it before. They can be the same test users as the evaluation for effectiveness, but a member of our team will count clicks and time the test.

Plan for Evaluation of User Satisfaction:

We will instruct users to do the following: Search for and identify a tutor on our website. After this we will have them fill out the following Likert scales to assess user's subjective thoughts on our website's effectiveness, efficiency, and overall satisfaction.

The search feature on this website is easy to use (Circle One)

Strongly Agree Agree Neutral Disagree Strongly Disagree

The search bar has an intuitive design (Circle One)

Strongly Agree Agree Neutral Disagree Strongly Disagree

I am satisfied with my experience using WiseGATOR (Circle One)

Strongly Agree Agree Neutral Disagree Strongly Disagree

3) QA test plan and QA testing:

a) Our usability test will be that our search functionality can find tutors from our database that are related to the text entered in our search bar. Testers will need the appropriate hardware and software in order to test searching. It will require an end-system that has a way to input text such as a laptop and keyboard. It will also require the following software; latest version of OS such as Windows or Mac, the most recent version of a browser of their choice (examples include Google Chrome, Safari, Firefox). Lastly, they will need the link to access our website:
http://ec2-3-14-250-152.us-east-2.compute.amazonaws.com:3000/
QA Tester will navigate to the link above on their preferred browser with their personal device. Each QA Tester will then conduct the following tests on our website by using the search bar we implemented.

Test # browser used	Test Title	Test Description	Test Input	Expected Output	Result

1	Search Tutor	The tester only inputs a tutor topic, while leaving the search bar to be nothing.	On the home page, other than the "About Me" pages, go to the top of the page, and locate the first gray box to the left in the top bar (on the right hand side of the text saying "WiseGATOR"). Make sure the gray box says "Select Topic". If not, click on said box, and a white box should show up below it. In the white box, press "Select Topic". To the right hand side of the gray box, there should be a longer white box. Press it, and then type “Sean”. After which, press “Search”, which is on the right hand side of the search bar.	The search result should display Sean Ryan, with the subject ID of “CSC”.	
2	Search Topic	The tester only inputs a name on the search bar, while leaving the tutor topic to be nothing.	On the home page, other than the "About Me" pages, go to the top of the page, and locate the first gray box to the left in the top bar (on the right hand side of the text saying “WiseGATOR”). Once you find the gray box, click on said box, and a white box should show up. In the white box, press "ARTH". To the right hand side of the gray box, there should be a longer white box. Make sure the white box is empty. If not, press it, and then delete all the text inside it. After which, press “Search”, which is on the right hand side of the search bar.	The search result should display Darien Banuelo, with the subject ID of “ARTH”.	
3	Search Tutor with Topic	The tester inputs both a name on the search bar and a tutor topic.	On the home page, other than the “About Me” pages, go to the top of the page, and locate the first gray box to the left in the top bar (on the right hand side of the text saying “WiseGATOR”). Once you find the gray box, click on said box, and a white box should show up. In the white box, press "DM". To the right hand side of the gray box, there should be a longer white box. Press it, and then type “Ka”. After which, press “Search”, which is on the right hand side of the search bar.	The search result should display only Karl Moreno, with the subject ID of “DM”.	
4	Empty Search	The tester leaves both the	On the home page, other than the “About Me” pages, go to the top of the page, and locate the first gray box to	The search result	

		search bar and tutor topic to be nothing.	the left in the top bar (on the right hand side of the text saying “WiseGATOR”). Make sure the gray box says "Select Topic". If not, click on said box, and a white box should show up below it. In the white box, press "Select Topic". To the right hand side of the gray box, there should be a longer white box. Make sure the white box is empty. If not, press it, and then delete all the text inside it. After which, press “Search”, which is on the right hand side of the search bar.	should display every tutor on the site.	

b) Tested on the following browsers: Chrome, Firefox, Microsoft Edge

Test # With Google Chrome	Test Title	Test Description	Test Input	Expected Output	Result
1	Search Tutor	The tester only inputs a tutor topic, while leaving the search bar to be nothing.	On the home page, other than the "About Me" pages, go to the top of the page, and locate the first gray box to the left in the top bar (on the right hand side of the text saying "WiseGATOR"). Make sure the gray box says "Select Topic". If not, click on said box, and a white box should show up below it. In the white box, press "Select Topic". To the right hand side of the gray box, there should be a longer white box. Press it, and then type “Sean”. After which, press “Search”, which is on the right hand side of the search bar.	The search result should display Sean Ryan, with the subject ID of “CSC”.	Pass
2	Search Topic	The tester only inputs a name on the search bar, while leaving the tutor topic to be nothing.	On the home page, other than the "About Me" pages, go to the top of the page, and locate the first gray box to the left in the top bar (on the right hand side of the text saying “WiseGATOR”). Once you find the gray box, click on said box, and a white box should show up. In the white box, press "ARTH". To the right hand side of the gray box, there should be a longer white box. Make sure the white box is empty. If not, press it, and then	The search result should display Darien Banuelo, with the subject ID of “ARTH”.	Pass

			delete all the text inside it. After which, press “Search”, which is on the right hand side of the search bar.		
3	Search Tutor with Topic	The tester inputs both a name on the search bar and a tutor topic.	On the home page, other than the “About Me” pages, go to the top of the page, and locate the first gray box to the left in the top bar (on the right hand side of the text saying “WiseGATOR”). Once you find the gray box, click on said box, and a white box should show up. In the white box, press "DM". To the right hand side of the gray box, there should be a longer white box. Press it, and then type “Ka”. After which, press “Search”, which is on the right hand side of the search bar.	The search result should display only Karl Moreno, with the subject ID of “DM”.	Pass
4	Empty Search	The tester leaves both the search bar and tutor topic to be nothing.	On the home page, other than the “About Me” pages, go to the top of the page, and locate the first gray box to the left in the top bar (on the right hand side of the text saying “WiseGATOR”). Make sure the gray box says "Select Topic". If not, click on said box, and a white box should show up below it. In the white box, press "Select Topic". To the right hand side of the gray box, there should be a longer white box. Make sure the white box is empty. If not, press it, and then delete all the text inside it. After which, press “Search”, which is on the right hand side of the search bar.	The search result should display every tutor on the site.	Pass

4) Peer Code Review:

Code Review Request for Search-Tutors

ⓘ You replied on Sun 12/10/2023 7:02 PM

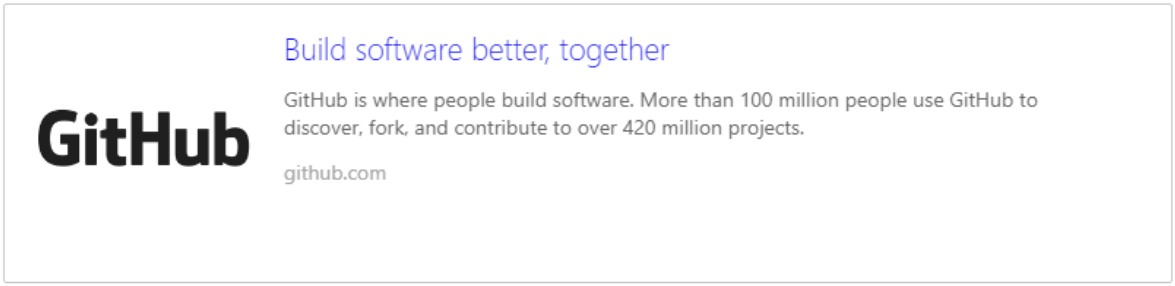
SR Sean Michael Ryan
To: Joaquin Nguyen Warren

Sun 12/10/2023 5:32 PM

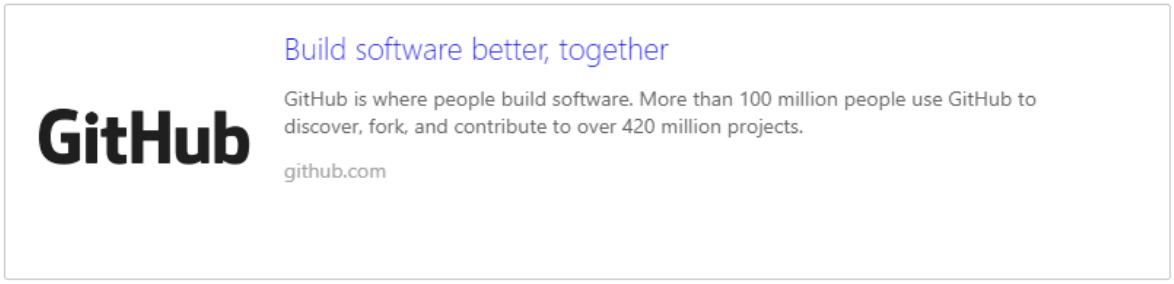
Good Evening Team Leader!

I am sending you links to my code for the search-tutors page and functionality for review. I used embedded java script to display the search results to the page and connected it to the database with a java script file. All advice and criticism is greatly appreciated!

https://github.com/CSC-648-SFSU/csc648-03-fa23-team06/blob/development/application/views/search-tutors.ejs



https://github.com/CSC-648-SFSU/csc648-03-fa23-team06/blob/development/application/routes/index.js



On the last link the implementation starts on line 241.

Thank you,
Sean Ryan
-Back End Team Lead

Reply Forward

Re: Code Review Request for Search-Tutors

JW

Joaquin Nguyen Warren

To: Sean Michael Ryan

Sun 12/10/2023 7:02 PM

Hey Sean!

Thanks for reaching out to me, I spent some time reviewing your code to give you something to work with in terms of feedback.

Overall, great work on the search function for our application. Other than it being very functional, I can tell right away that this code is very refined and organized. I have no trouble at all understand the context of each file. My only feedback is to make sure to include in-line comments in a couple parts of the code to further explain variable names and elaborate on the way the code between the files interacts with one another. I've included some screenshots of these opportunities below:

```
240  // Code Review: Joaquin Date: 12/10
241  // Overall this is very organized and easy to read, it would benefit from some inline comments.
242  // I can understand what this code does with context, my only suggestion is explain some variable names
243  // Route to display tutors and handle search
244  router.get('/search-tutors/', (req, res) => { // Inline comments to explain logic flow -Joaquin
245      let sql = 'SELECT * FROM tutors';
246      let queryData = [];
247
248      if ((req.query.search || req.query.subject)) {
249          sql += ' WHERE';
250
251          if (req.query.search) {
252              sql += ' name LIKE ?';
253              queryData.push('%' + req.query.search + '%');
254          }
255
256          if (req.query.subject) { // Good consistent naming with table in SQL -Joaquin
257              if (req.query.search) {
258                  sql += ' AND';
259              }
260              sql += ' subject_id = ?';
261              queryData.push(req.query.subject);
262          }
263      }
264      db.query('SELECT * FROM topics', (err, subjects) => {
265          if (err) throw err;
266          // Explain the query and what table is called -Joaquin
267          db.query(sql, queryData, (err, results) => {
268              if (err) throw err;
269              res.render('search-tutors', { loggedIn: req.session.userId ? true : false, tutors: results, topics: subjects, name: req.query.search,
270                  subject: req.query.subject });
271          });
272      });
273  });
274
```

```
<!-- Code Review: Joaquin  Date: 12/10 -->
<!-- In summary, this is a good way to use our stack to its best capabilities for the search functionality -->
<!-- It could benefit from some in-line comments to help a reader understand where the logic flows and how files interact-->
<!DOCTYPE html> <!-- Missing a file header comment -Joaquin -->
<html lang="en">
<head>
    <!-- Google tag (gtag.js) -->
    <script async src="https://www.googletagmanager.com/gtag/js?id=G-N3SV9BXBFJ"></script>
    <script>
      window.dataLayer = window.dataLayer || [];
      function gtag(){dataLayer.push(arguments);}
      gtag('js', new Date());

      gtag('config', 'G-N3SV9BXBFJ');
    </script>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="../webpage/CSS/navbar.css"/> <!-- For universal navbar -->
    <link rel="stylesheet" type="text/css" href="../webpage/CSS/dashboard.css"/> <!-- Tutor divs need dashboard.css for proper display -->
    <script src="../webpage/JS/search-tutors.js" defer></script>
    <title>Tutor Search</title>
</head>
<body>
    <header class="navbar">
        <div class="left">
            <a href="/" class="home-link">
                <img src="../webpage/tutor-pictures/GATORlogo.png" width="48" alt="Logo" class="Logo">
                <span class="title" style="color: #efebef; padding-right: 10px; padding-left: 10px;">WiseGATOR</span>
            </a>
        </div>
        <div class="search-container">
            <form class="search-form" action="/search-tutors" method="GET">
                <select name="subject">
                    <option value="">Select Topic</option>
                    <% topics.forEach(topic => { %> <!-- Good naming consistency with M2 -Joaquin -->
                        <option value="<%= topic.subject %>" <%= (typeof subject !== 'undefined' && subject === topic.subject) ? 'selected' : '' %>><%= topic.subject %></option>
                    <% }); %> <!-- In-line comments to describe the logic -Joaquin -->
                </select>
                <!-- Good validation of input length -Joaquin -->
                <input type="text" name="search" value="<%= typeof name !== 'undefined' ? name : '' %>" placeholder="Search by name" maxlength="40">
                <!-- Clever way to integrate js -Joaquin -->

                <button type="submit">Search</button>
            </form>
```

Thank you for taking the time to ensure the code's quality. Please keep up the hard work and feel free to message me if you have any questions.

Best,
Joaquin Warren
-Team Lead

---

5) Self-check on best practices for security:

Asset to Protect	Type of Possible Attack	Strategy to Mitigate or Protect Asset
User Passwords	Cyber Attack/Hacking	Password encryption
Database (containing student emails, passwords,	SQL Injection	Search bar authentication prevent form submission if non-alphanumeric characters are present, only allows up to 40 characters

AWS Console: Credit Card info, Billing info (address)	Account Hacking	Root User/Admin Login Credentials are not shared with anyone, including team members
Source Code	Plagiarizing Code, Using our source code without giving credit to earn profit	Keep all ideas within our company, don't share with competitors. Get a patent for ideas we originally create
Trademarked/Copyrighted Content: Logo, Name of Application	Stolen content	Add our own copyright or trademark to all materials we originally created
Customer Safety and Security	Spam bots, malicious tutor bots that are not real people and want to fish on our website with links and obtain payment information	Registration authentication that checks for SFSU emails, and tutors must be approved by admin before published

6) 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
DONE

3. All or selected application functions shall render well on mobile devices
DONE

4. Data shall be stored in the database on the team's deployment 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
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 shall 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. 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

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

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

17. The application UI (WWW and mobile) shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Fall 2023. For Demonstration Only” at the top of the WWW page nav bar. (Important so as to not confuse this with a real application).
DONE