SW Engineering CSC648-848 Fall 2023

SFSU TutorLink

Team 02

Jeremy Woodling - Team Lead - jwoodling@sfsu.edu

Abel Seyoum - Github Manager

Brandon Watanabe - Front End Lead

Ankita Mukherjee - Backend Lead

Guillermo Villar

Lars Severson

Milestone 2 Part 1

Submitted	Revised
10/13/23	**10/30/23**

<u>Table of Contents</u>

1. Executive Summary

In the SFSU community, students sometimes find challenges that extend beyond the classroom. Different timetables, working on the side or managing a school-life balance are problems many students struggle with, and no one knows how to manage time and courses better than those that have already done it. That is why SFSU Tutorlink appears as a platform to connect SFSU students with tutors who have gone through the same problems earlier, and are best equipped to help students learn and pass their courses swiftly.

We are a web-based platform bridging SFSU students in need with tutors with expertise, tailored to the students courses and needs. Students can schedule their tutoring sessions matching their own timetables and goals through our messaging system linking them with their tutors. This all can seem not very tied into SFSU, but we are tied to our community through various different unique offerings like the ability to search through your Courses and Professors, an adherence to all SFSU guidelines, and our focus on development of connections within the community, making our university a more supportive and collaborative learning environment.

Finally, we believe that investing into our student-run startup is an investment on academic success and the well-being of students, as well as an investment into the sense of community here at SFSU. SFSU TutorLink is not just a project; it's a promoter for academic excellence and a testament to your commitment to student success at SFSU.

2. Data Glossary

User Types	
User	An SFSU student who is only interested in using TutorLink to hire a tutor for classes.
Unregistered User	A user who is visiting the site, but has not created an account.
Registered User	A user of the website that created an account using their "@sfsu.edu" email
Tutor	An experienced SFSU student who uses TutorLink to offer their services to other users. They may also use TutorLink to hire tutors for other classes.
Admin	A member of our workforce that will work on maintaining the website in day to day operations. This includes tasks like reviewing and publishing tutor applications via SQL Workbench.

General Items	
Search	A Segment of the application that allows users to search for tutors by various aspects of their provided date (eg. class taught).
Dashboard	A platform where users can see messages and posts
Off-site	Not through TutorLink, eg. a tutor contacting a user via the phone number they included in their message.
Tutor Application	A form that a prospective tutor would fill out with their name, topic, class, description. They will also be able to upload videos, pictures, and a resume. This application will be verified by an Admin before their listing is posted

Registered User	
Registered User	A user of the website that has created an account through the "sign up" page.
Email	An "@sfsu.edu" email used in signup/login.
Password	A hashed text string used for logging in.

Tutor Post	
Tutor	A registered TutorLink user that gained approval for their application to be a tutor and now has an active Tutor Post.
Tutor Post	A profile listing created by a registered user that contains the below items.
Subject	A class subject selected from a drop-down menu during the application. E.g. "CSC"
Class	A tutor-entered number for a class associated with the topic. E.g. "648"
Description	A body of text for introducing themselves to users.
Photo	A tutor-uploaded image file for introducing themselves to users.
Video	A tutor-uploaded video file for introducing themselves to users.
Resume	A tutor-uploaded text document for introducing themselves to users.

Dashboard	
Dashboard	A platform where users can see messages and posts.
Postings	A tab available to registered users that contains a list of their tutor postings.
Received	The default tab for message viewing.

Sent	A secondary tab for viewing messages the user has sent.

Messages	
Message	A text message sent to tutors to connect them with students for outside (off-site) communications.
User	Displays the user that the message was sent to
Subject	A generated line of text pre-filled in the message with details from the tutor's posting. E.g. "CSC 648"
Date	The time and date the message was sent
Body	The main text of the message

Admin	
Admin	A member of our workforce that will work on maintaining the website in day to day operations. This includes tasks like reviewing and publishing tutor applications via SQL Workbench.
Tutor Application	An entry in the "tutor_request" table of our MySQL database.
Approved Tutor	An entry that was moved from the "tutor_request" table to the "tutors" table of our MySQL database.

3. Prioritized Functional Requirements

Priority 1 (Necessary):

1. Unregistered User:

    1.1. Unregistered users shall be able to register for one and only one account by using their "@sfsu.edu" email

    1.2. Unregistered users shall be able to browse tutors

    1.3. Unregistered users shall be able to search for a tutor by name

    1.4. Unregistered users shall be able to search for a tutor by subject

    1.5. Unregistered users shall be able to search for a tutor by class

    1.6. Unregistered users shall be able to view a tutor's photo

    1.7. Unregistered users shall be able to access a tutor´s video

    1.8. Unregistered users shall be able to access a tutor's resume

2. Registered User:

    2.1. A registered user shall be able to login

    2.2. A registered user shall be able to request a new password

    2.3. A registered user shall inherit all functions available to an unregistered user

    2.4. A registered user shall be able to send a message to a tutor

    2.5. A registered user shall have one and only one dashboard

    2.6. A registered user shall be able to submit an application for becoming a tutor

    2.7. Tutors shall be able to upload one and only one picture

    2.8. Tutors shall be able to upload one and only one resume

3. Messages:

    3.1. Messages shall be able to be created by a user

    3.2. Messages shall be able to be received by one and only one tutor

4. Dashboard:

    4.1. The dashboard shall display zero to many of a registered user's tutor postings

    4.2. The dashboard shall display the metadata of many messages

    4.3. The dashboard shall display a link to read the full message

5. Admin:

    5.1. Admin shall be required to approve tutor applications prior to going live

    5.2. Admin shall be required to deny inappropriate tutor applications

    5.3. Admin shall be able to block users

Priority 2 (Convenient) :

1. Unregistered User:

    1.1. Unregistered users shall be able to start the application for becoming a tutor without prior registration

2. Registered User:

    2.1. Tutors shall be able to upload one and only one video

Priority 3 (Possible addition):

1. Registered User:

    1.1. A registered user shall be able to leave one and only one review on a tutor's

    profile

    1.2. A tutor shall be able to change their posted photo

    1.3. A tutor shall be able to change their posted video

    1.4. A tutor shall be able to change their posted resume

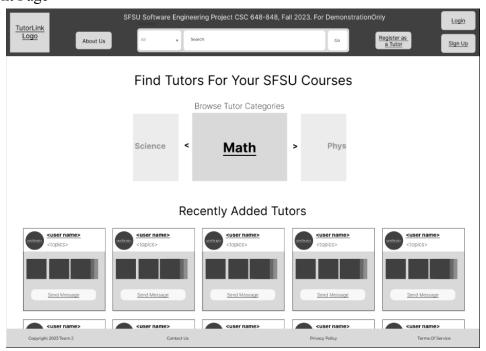   1.5. A tutor shall be able to change their posted description
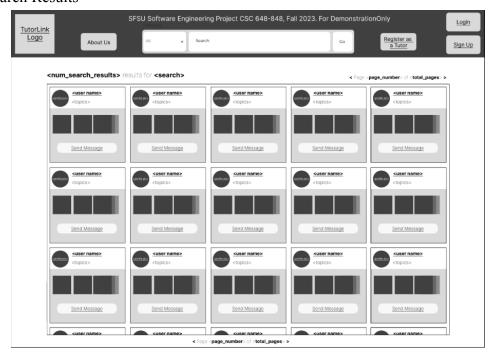
4. UI Storyboards

[Link to Figma Prototype](#)

Use Case 1: Student seeking assistance

Mia, a **student** at SFSU, needs a **tutor** for her heavy coursework. First, she browses through tutoring services to find tutors who are experts in the subjects she wants to learn. She does this **search** by either using the categories or by typing the class name in the search bar. Mia examines the tutor's information to ensure they are qualified and then proceeds to send a message to them. Next, she is prompted to either sign in or sign up on the tutoring website. Once Mia has signed in, she is able to contact the selected tutor through the in-site messaging system. She sends a message to the tutor to discuss her specific challenges and arranges the schedule for tutoring sessions by filling out the message form on the site. After this, Mia is excited and eagerly awaits the tutor contacting them **off-site**.
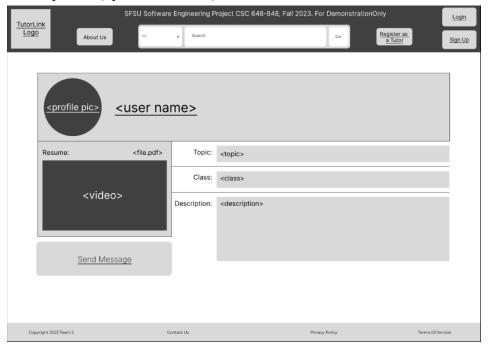
1. Front Page

2. Search Results

3. View tutor profile (opens in new tab)
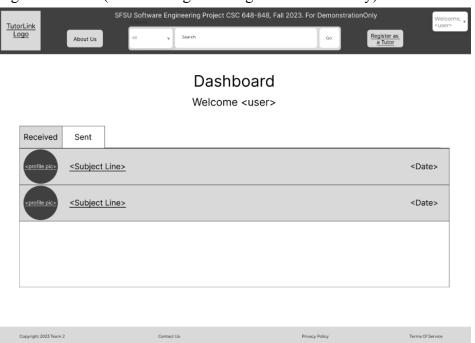
4. Tutor message prompt
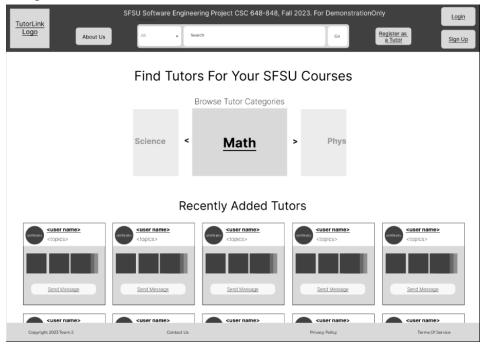

5. Login prompt

6. Signup prompt

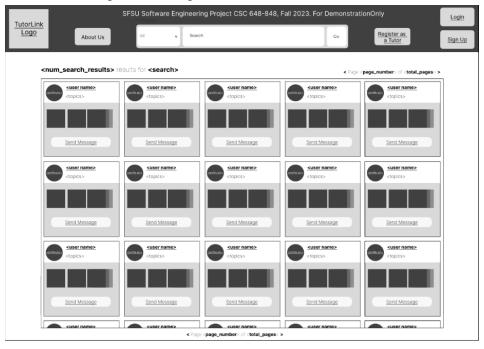7. Message Dashboard (flash message: message sent successfully)

Use Case 2: Tutor signs-up to help

Smith, a **tutor** seeking **students** to teach, follows these steps to use our tutoring service effectively. First, he explores the various courses he can offer by selecting from a category list or entering specific keywords in a search bar. Then, he creates a comprehensive tutor profile by completing a form with his name, topic, class, description, **resume** and a professional photo on the site. After submitting this information, he is prompted to either sign in or sign up. Following that, he regularly updates his profile with his skills and qualifications. Upon signing up in the **dashboard**, the tutor can check **messages** through the site's message system, sort them by date, and also post on the site.
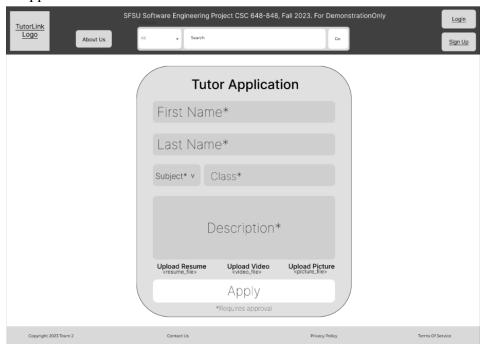
1. Front Page
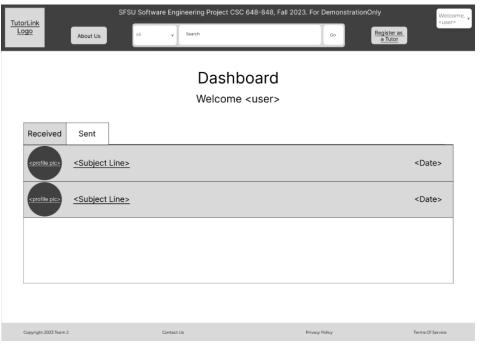
2. Search Results/Competitor tutor profiles

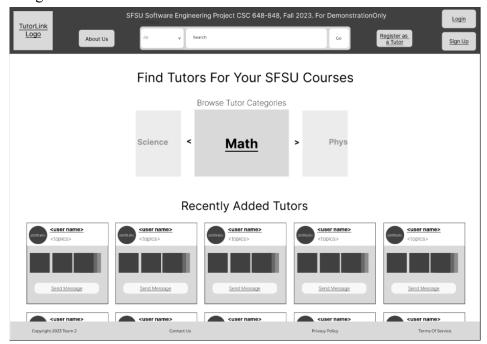3. Tutor Application Form

4. Login/Signup prompt

5. Message Dashboard

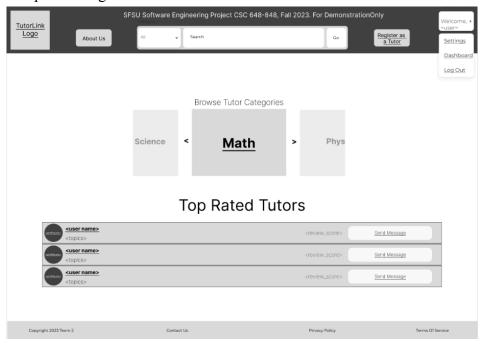## Use Case 3: User accessing their Dashboard

Smith, a tutor on our site, wants to check for recent activity. He navigates to his **dashboard**

where he sees a list of **messages** received and sent along with a list of his **tutor postings**.
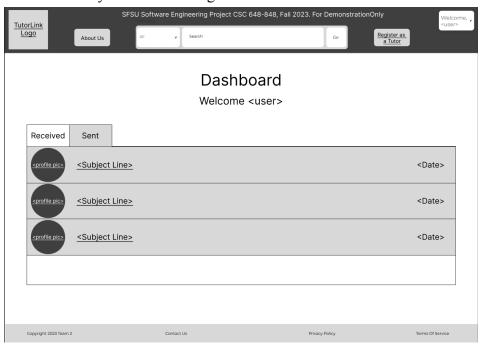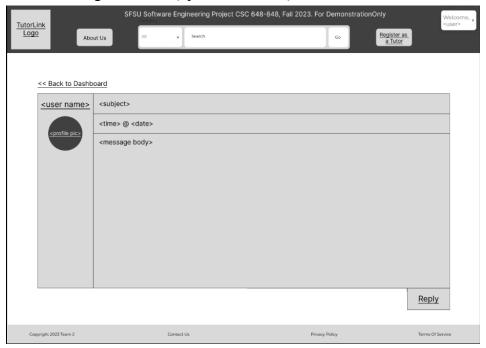
1. Front Page

2. Uses dropdown to go to Dashboard

3. Views their recently received messages

4. Clicks on a message to read it (opens in new tab)

Use Case 4: Admin moderating the tutors

Olivia is an **admin** responsible for ensuring the seamless operation of our web-based platform, which connects SFSU students with dedicated tutors. Her primary duty is to review and approve tutor profiles, sorting them by new postings in the SQL workbench, and granting approval if they adhere to the platform's rules, allowing them to go live. Additionally, Olivia addresses user issues and maintains the platform's safety and fairness by promptly removing inappropriate items or users when necessary.

1. Opens SQL workbench
2. Does "SELECT * FROM db.tutor_request" to view pending applications

3. Approves or denies application by moving it to the table of approved tutors

5. Architecture and DB organization

DB organization:

Table	User			
Key	user_id	user_name	user_pw	user_email
Value Type	int (pk)	string [32]	string[162]	string [64]
Details	Primary Key and index in DB	Users chosen username	SHA-512 hash of users password represented as hex string	Users full email, must be sfsu email.

Table	Subject		
Key	subj_id	subj_short	subj_long
Value Type	int (pk)	string [8]	string [32]
Details	Primary Key and index in DB	Short form of subject name (eg. CSC or PHYS)	Long form of subject name (eg. Computer Science or Physics)

Table	Tutor (Part 1)				
Key	tutor_id	tutor_user	tutor_name	tutor_bio	tutor_cv
Value Type	int (pk)	int (fk)	string [64]	string [512]	Static Link (string[64])
Details	Primary Key and index in DB	Reference to user account of tutor from 'User' table	Actual name of tutor (or real life name to be presented)	Short text bio (or description) for tutors. 512 char limit.	Static link to tutors CV/Flyer PDF that they uploaded.

Table	Tutor (Part 2)			
Key	tutor_photo	tutor_vid	tutor_subj	tutor_subj_num
Value Type	Static Link (string[64])	Static Link (string[64])	int (fk)	string[32]
Details	Static link to uploaded photo for tutor	Static Link to uploaded video for tutor	Reference to tutor subject from 'subject' table	String with course number(s) tutor is offering

Table	Message				
Key	msg_id	msg_tutor	msg_student	msg_listing	msg_text
Value Type	int (pk)	int (fk)	int (fk)	int (fk)	string [512]
Details	Primary Key and index in DB	Reference to tutor's user account from 'Users' table. Note: "To" field.	Reference to student's user account from 'Users' table. Note: "From" field.	Reference to tutor's listing from 'Tutor' table. (Used for subject info)	String holding the text of the message.

Table	Tutor_Request
Note	This table is a copy of the 'Tutor' table with the 'tutor_' segment of names replaced with 'req_'. When a tutor listing is posted, it will be posted here, then an admin will review it, and move the data from this table to the 'Tutor' table "authorizing" the tutor post.

Media Storage:

In file systems media formats are limited to only one for simplicity. JPEG for images, MPEG-4 for videos, PDF for CV. That being said this may be subject to change to allow for more file format support depending on if they fit into the scope of the project and deadlines. We're not using BLOBs.

Search/Filter Architecture and Implementation:

Search will primarily focus on class name and tutor names. These details are stored in our database within the "Tutor" table. For classes, we will search the "tutor_subj" column, and for the tutors, we will search the "tutor_name" column. We will use SQL to construct queries that utilize the SQL %LIKE operator to find matching class and tutor names. After the user enters class name or tutor name in the search bar and clicks the 'Go' button, it will use the value entered in the search field to perform a select query in the tutor table. Finally, the data retrieved from the database is ordered, then rendered using a POST request. If the user inputs null it retrieves all records from the table.

Non-Trivial Algorithms:

We should only need one non-trivial algorithm in our code which is the rating system. We will be using rolling averages to calculate a tutors rating. This will be fairly simple given that the database stores the current average and the total number of reviews left for a tutor. This then requires weighting of the new review and the current rating and then adding them together.

Beyond that, ranking will be very simple, since the reviews are saved as a rolling average, it will just be sorted by the rating field when making the SQL request.

New Frameworks:

We have added two new frameworks to our backend code. Firstly we added flask login, which allows easy state management for flask backends. Secondly we added Werkzueg, specifically the security package for generating secure hashes for secure password storage.

6. Identify Risks

Risk	Risk Level	Notes
Skills	Moderate	Although not everyone is fairly skilled with the exact frameworks we are using (Flask) everyone on the team has some level of experience with python, and a willingness to learn. I think the only setbacks we would get here is some time lost to code review and fixing.
Schedule	High	We are doing an okay job at staying on schedule, but we are all very busy college students. Most of us have other high intensity courses we are taking outside of this class so when it comes to development we may fall behind due to workload.
Technical	Low-Moderate	For the technical side of things we should have most of it planned out in a good bit of detail. The biggest concern right now is handling file uploads, mainly video file uploads.
Teamwork	Moderate	The FE and BE team leads are still getting into their roles leading their individual side which leaves some concern, but this will likely improve with time. Other than that, we are running into some organization and communication issues, but those have been actively improving with time.
Legal/Content	Low	Legal content in a realistic sense is fine because we are using it for an educational project which is protected under fair use. That being said, in an actual company, we would need to carefully review the licenses of everything we are using to see if it fits within the scope and requirements of our project, and avoid licenses like GNU GPL.

7. Project Management

For the project we have a few guidelines in place to help with the project go smoothly. Firstly, we have been moving towards the FE and BE lead taking a more active role in leading their sections of the team. Beyond this, we have started using Trello and are in the process of adding tasks to the trello page. This will help with identifying tasks and their deadlines and getting the project done in a reasonable amount of time. We also have a layout backend layout we will be using to make working between the front end and back end easier. For this we will have classes in the code that hold all the data for the front end. This will allow backend devs to insure their code works without the frontend being completed, and allow front end devs to test the front end with dummy back end code. Beyond that, we also have our regular Friday meetings where we spend some time getting work done.

8. Use of ChatGPT

New Update:

For Milestone 2, we used ChatGPT to generate example tutors via INSERT statements to our SQL table for tutor applications.

Old Update:

We have used the September 25th version.

We did review the policy and found it useful to go through some requirements text and as a spell checking tool for our executive summary. In addition to this, the team lead will be using ChatGPT for email fluff as it is effective with some minor editing.

We would say it definitely is helpful (Does a bit more in depth grammar checking than google docs word directory matching). It would be classified towards medium usefulness.

We did draft our assignment first, as mentioned it was used for spell checking and more general grammar adjustments, but our contents and ideas were already laid out.