Final Project for SW Engineering Class CSC 648-848 Fall 2025

EduGator

Team 5:

Team lead:	Grady Walworth wwalworth@sfsu.edu
GitHub Master:	Michael
Frontend lead:	Tejas
Frontend developer:	Kameron
Backend lead:	Chris
Backend developer:	Hardy

Milestone 5

Date: 12/16/2025

Demo URL: http://18.217.207.30/

1). Product Summary:

EduGator

EduGator is a peer-to-peer tutoring platform designed exclusively for San Francisco State University students. Its purpose is to make finding academic support as simple and trustworthy as asking a knowledgeable classmate. Every user signs in with their SFSU credentials, ensuring a safe, authentic environment where students can confidently connect for help or offer guidance to others. EduGator stands apart from generic tutoring services by focusing on community-building, campus exclusivity, and ongoing mentorship rather than one-time, commercial transactions.

The platform tailors each student's experience by filtering subjects based on what they are studying, and tutoring opportunities are organized by broad course and subject areas. A color-coded calendar makes scheduling effortless, allowing students to browse tutor availability, filter by subject, and choose either Open group sessions or one-on-one appointments. An integrated search bar enables quick access to sessions by subject, tutor name, or day. Built *by SFSU students for SFSU students*, EduGator strengthens academic success, reduces learning barriers, and fosters a supportive campus culture rooted in mentorship, collaboration, and mutual growth.

Final P1 Functional Commitment (Major Committed Functions)

For Unregistered Users

- Can browse tutors and view public tutor profiles.
- Can register as a student or tutor using an SFSU email.
- Are redirected to log in or register when trying to access restricted features.
- Can view available tutoring subjects and general course areas.

Messaging

- Registered users can send and receive messages with tutors.

For Registered Users (Students and Tutors)

- Can sign in using their SFSU email.
- Can create their personal profile.
- Tutors can post their available tutoring times.
- Tutors receive notifications for requests and confirmations.
- Tutors can list the subjects and courses they offer.
- Students can view a tutor's expertise.
- Students can filter tutors by subject, course, keyword, or availability.

System / Admin

- The system creates a tutoring session record when a tutor posts availability, and this session appears on the student's calendar.
- Admins can approve or reject tutor applications.

URL to your product accessible to instructors, on deployment server:

http://18.217.207.30/

2) Milestone documents – M1-M4 (Pages 4 - 77 of this PDF)

Contains frozen revised reports for M1 and M2; M3 feedback summary report; and M4 (M4 is submitted with M5 for the first time). Simply concatenate the above reports starting from M1 report.

Milestone 1

SW Engineering CSC648-848 Fall 2025

EduGator

Team 5

Team lead: Grady Walworth - wwalworth@sfsu.edu
GitHub Master: Michael
Frontend lead: Tejas
Frontend developer: Kameron
Backend lead: Chris
Backend developer: Hardy

Milestone 1

History Table:

Version 1.0	Due: 10/16/2025
Version 2.0	Submitted: 10/28/2025

1. Executive Summary

Our project, EduGator, is a peer-to-peer tutoring platform designed exclusively for San Francisco State University students. It provides a trusted, intuitive space where members of the SFSU community can connect for academic support, share knowledge, and build ongoing learning relationships. The goal is to make finding or offering tutoring as simple and secure as connecting with a classmate.

Unlike generic tutoring apps or social media groups, EduGator verifies every user through their SFSU credentials, ensuring a safe and authentic academic community. The novelty of our approach lies in its focus on campus exclusivity, mutual trust, and recurring peer connections rather than one-time transactions. All tutoring is community-driven and non-commercial, emphasizing collaboration, mentorship, and academic growth over profit.

When students register, the platform automatically tailors their experience by pre-filtering content based on the subjects they are studying. Tutoring opportunities are organized by general subject areas (not specific course numbers) and distinguished by Lower Division and Upper Division levels to keep the system streamlined yet meaningful.A color-coded calendar makes scheduling simple. Students can view tutor availability at a glance, filter by subject, and choose between Drop-in sessions (open group tutoring) or Appointments (1-on-1 meetings). The integrated search bar allows users to find sessions by subject, tutor name, or day

By funding this project, the university will empower students to support one another academically, reduce learning barriers, and foster a stronger, more connected campus culture. EduGator not only enhances student success but also reinforces the values of accessibility, inclusivity, and community service that define SFSU.

Our team is made up of passionate San Francisco State University students who want to help our peers succeed. EduGator is built by students, for students. Our goal is to create something that truly reflects the spirit of SFSU. We see EduGator as more than an app, it’s a step toward building a stronger culture of mentorship and mutual growth across our university.

2. Personae

Persona 1: SFSU Student User - Alex (SFSU Student)

General Characteristics:

The undergraduate student is majoring in Computer Science at San Francisco State University. He frequently attends campus events, workshops, and tech meetups to build his professional skills and meet other students.

Goals:

The guest user hopes to use the platform to discover public events, browse photos and descriptions, and get a clear sense of the university's culture. By accessing this information easily, they can better evaluate whether SFSU aligns with their academic goals and lifestyle preferences.

Pain Points:

Within SFSU's Student Center, students often find it difficult to locate the specific functions or services they need. Important information is scattered across multiple pages and systems. For example, course registration, event announcements, and student services are managed separately, which makes the navigation process confusing and time-consuming. Students hope for a unified and well-organized platform where all campus-related information is centralized and easy to access. This would allow them to complete tasks more smoothly and efficiently.

Persona 2: SFSU Staff - Dr. Maria Lopez

General Characteristics:

A faculty or staff member at San Francisco State University, such as a professor or event coordinator. She frequently organizes campus events, workshops, and extracurricular activities to increase student engagement and foster a stronger sense of campus community.

Goals:

The staff member aims to simplify and centralize event management. They want to post new activities efficiently, edit event details when plans change, and track participation through an easy-to-use dashboard. They also wish to communicate updates and collect feedback from students within a single platform to improve event organization.

Pain Points:

Many instructors at SFSU struggle because there is no single, consistent place to post important information such as class announcements, assignments, or upcoming exams. Each course or event often uses a different system, which forces both faculty and students to switch between multiple platforms like Canvas, email, and department websites. This fragmented setup makes it difficult for teachers to ensure that all students receive timely updates, and it often results in confusion or missed information.

Persona 3:Guest User - Jordan Lim

General Characteristics:

This persona represents a high-school senior interested in applying to San Francisco State University. The guest user wants to explore what campus life is like and understand what kinds of student activities are available before making a college decision.

Goals:

The guest user hopes to use the platform to discover public events, browse photos and descriptions, and get a clear sense of the university's culture. By accessing this information easily, they can better evaluate whether SFSU aligns with their academic goals and lifestyle preferences.

Pain Points:

Many official university websites are not intuitive, and some event details require login credentials that prospective students do not have. It is difficult for them to find accurate, up-to-date information about campus life. They seek a simple, welcoming, and easy-to-navigate interface that allows non-students to explore the community freely.

3. **High-level Use cases**

Use Case 1: Searching for Tutors by Name

Actors: Alex (SFSU Student)
Description:
An **SFSU Student** logs into the tutoring platform and searches for a specific **Tutor** by name using the search bar. The system displays matching **Tutor** profiles and their availability from the **Calendar**. The student can view each **Profile** for more details such as ratings and **Review / Feedback**, then proceed to filtering or booking.
Frequency/Importance: Weekly; important for quickly locating preferred tutors.
Environment/Context: Accessed via desktop or mobile browser through the website.

Use Case 2: Filtering Tutors

Actors: Alex (SFSU Student)
Description:
After performing a search, the **Student** refines results using filters such as **Subject**, availability, and **Review / Feedback** rating. The system dynamically updates the list to show **Tutor**s matching the selected filters and displays their earliest available sessions from the **Calendar**. This enables students to efficiently find suitable academic support without browsing irrelevant results.
Frequency/Importance: Weekly; enhances the efficiency of tutor discovery and booking.
Environment/Context: Used from dorms, libraries, or off-campus via web browser or mobile device.

Use Case 3: Guest Browsing

Actors: Jordan Lim (Guest User)

Description:

A **Guest User** visits the tutoring website to explore available **Tutor**s and **Subject**s. The system allows the guest to search for **Tutor**s by name and apply filters such as **Subject**, availability, or rating to refine results. Guests can view public content, including **Tutor** summaries, general **Post**s, and open time slots on the **Calendar**. When a guest attempts to view a full **Profile** or book a **Session**, the system prompts: “Only registered SFSU members can continue. Please log in with your SFSU email.” Upon logging in, the user is returned to their original page to continue seamlessly. This encourages exploration while maintaining SFSU exclusivity.

Frequency/Importance: Daily; supports user onboarding and smooth transition to full access.

Environment/Context: Publicly accessible on desktop or mobile browsers.

Use Case 4: Booking a Tutoring Session

Actors: Alex (SFSU Student), Jordan Lim (Guest User), Tutor

Description:

An **SFSU Student** selects a preferred **Tutor** from the search results or **Profile** and clicks “Book Session.” The system displays the **Tutor**’s available slots from the **Calendar**, allowing the student to choose a convenient time. A confirmation panel shows **Session** details including **Subject**, session type (from **Post**), and duration. Once confirmed, the **Session** appears on both the **Tutor** and **Student Calendar**s. If a **Guest User** attempts to book, a prompt requests SFSU login before proceeding.

Frequency/Importance: Weekly; essential for connecting students to personalized academic support.

Environment/Context: Used via website on any browser.

Use Case 5: Staff Management of Tutoring Sessions

Actors: Dr. Maria Lopez (SFSU Staff), Tutor

Description:

A **Tutor** logs into the tutoring system to manage tutoring operations. They can create and edit **Post**s for upcoming sessions, adjust **Calendar** availability, view **Session** attendance, and review **Feedback** submitted by students. This centralized management ensures consistent quality and effective coordination of tutoring services across the platform.

Frequency/Importance: Weekly; critical for maintaining accurate schedules and supporting academic success.

Environment/Context: Accessed from office desktops or remotely through the website.

4. **List of main data items and entities – data glossary/description**

- **Admin**:
    - can access all data and content and modify the database. Needs to login/register
    - Allows for managing users, subjects/courses, sessions, disputes, payments, site content, and analytics.
    - Can impersonate users for support. Requires login/registration with elevated privileges.
- **Tutor**
    - Has their own profile, can set availability, offer session types/prices, accept/decline bookings, can deliver sessions, and record outcomes/notes.
    - Requires login/registration and verification.
- **Student**
    - Has their own profile, can search tutors, view profiles, book sessions or packages, attend sessions, message tutors, and manage cancellations.
    - Requires login/registration for booking.
- **Unregistered User**
    - Does not have an account
    - Can browse public content (tutor summaries, subjects, available), initiates signup.
    - Can see available times
    - No booking/messaging until registered.
- **Subject**
    - Represents a general academic area (e.g., “Mathematics,” “Computer Science”).
    - Used to categorize tutors and sessions for easier searching and filtering.
    - Ensures consistent organization across all tutoring Posts.
- **Post**
    - Created by tutors to advertise availability for tutoring sessions.
    - Includes details such as subject, date/time, description, and session type (Drop-in or Appointment).
    - Acts as the primary way tutors make themselves discoverable to students.

- **Session**
    - A confirmed tutoring meeting between a tutor and a student.
    - Contains information such as participants, subject, location, time, and status.
    - Can originate from a Post or direct booking request.
    - Generates confirmation and reminder notifications for both parties.
- **Calendar**
    - The central scheduling feature shows all available and booked sessions.
    - Allows users to filter by subject, tutor, or color-coded session type.
    - Tutors and students have personalized views to manage their schedules.
- **Review / Feedback**
    - Submitted by students after a tutoring session.
    - Contains a rating, written comments, and a timestamp.
    - Supports quality assurance and helps other students choose tutors.
- **Profile**
    - Displays personal and academic details of a user.
    - Tutor profiles include expertise, bio, and session history.
    - Student profiles may show booked sessions or preferred subjects.
    - Accessible only to authenticated users.
- **Analytics Record**
    - Aggregated data used by admins to analyze system usage.
    - Tracks metrics such as active users, popular subjects, and session frequency.
    - Does not contain personally identifiable information.

5. List high level functional requirements

Unregistered Users:

1. Unregistered users shall be able to browse tutors and view public tutor profiles.
2. Unregistered users shall be able to register as a student or tutor.
3. Unregistered users shall be able to register using their SFSU email.
4. System shall redirect unregistered users to the login/register page when attempting to access certain features.
5. Unregistered users shall be able to view available tutoring subjects and courses.

Registered Users (Students and Tutors):

6. Users shall be able to sign in using their SFSU email.
7. Users shall be able to create and edit their personal profile.
8. Users shall be able to edit their personal profile.
9. Tutors shall be able to post their available tutoring times.
10. Tutors shall be able to receive notifications for requests, and confirmations.
11. Tutors shall be able to state the courses and subjects they offer.
12. Students shall be able to view a tutor's expertise.
13. Students shall be able to filter tutors by course, subject, keyword, or availability.
14. Students shall be able to cancel their tutoring requests.
15. Students shall be able to reschedule their tutoring requests.
16. Students shall be able to rate and provide feedback about the tutors.
17. Tutors shall be able to update previously posted tutoring sessions.
18. Tutors shall be able to remove previously posted tutoring sessions.

System/Admin:

19. System shall create a record when a tutor creates a tutoring session; this session shall appear on the student’s calendar.
20. Admins shall be able to approve or reject tutor applications.
21. Admins shall be able to delete tutoring sessions.
22. Admins shall be able to remove user accounts that violate policies.
23. System shall notify tutors when a tutoring request is made.

6. List of non-functional requirements

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers
3. All or selected application functions shall be rendered well on mobile devices (no native app to be developed)
4. Posting of tutor information and messaging to tutors shall be limited only to SFSU students
5. Critical data shall be stored in the database on the team's deployment server.
6. No more than 50 concurrent users shall be accessing the application at any time
7. Privacy of users shall be protected
8. The language used shall be English (no localization needed)
9. Application shall be very easy to use and intuitive
10. Application shall follow established architecture patterns
11. Application code and its repository shall be easy to inspect and maintain
12. Google Analytics shall be used
13. No e-mail clients shall be allowed. Interested users (clients) can only message service providers via in-site messaging. One round of messaging (from client to service provider) is enough for this application. No chat functions shall be developed or integrated
14. Pay functionality (e.g., paying for goods and services) shall not be implemented nor simulated in UI.
15. Site security: basic best practices shall be applied (as covered in the class) for the main data items
16. Media formats shall be standard as used in the market today
17. Modern SE processes and tools shall be used as specified in the class, including collaborative and continuous SW development and GenAI tools
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 2025. For Demonstration Only" at the top of the WWW page Nav bar. (Important so as to not confuse this with a real application).

7. **Competitive analysis (functions/features only, not business or marketing analysis)**

Our advantages:

Comparison Table:

Feature	Tutoring SFSU	Superprof.com	Tutor.com	Bayareatutor .org	EduGator
Text Search	+	++	+	+	+
Browse	+	+	+	+	++
Calendar for scheduling	+	-	-	-	++
User verification via email	++	+	+	+	+

\+ Feature exists ; ++ superior feature; - does not exist

Summary:

Our planned tutoring platform builds on the strengths of existing services like Superprof, Tutor.com, and BayAreaTutor.org while focusing on unique SFSU-specific advantages. Unlike general tutoring sites, our system is designed exclusively for SFSU students and tutors, providing verified access through university email authentication to ensure trust and safety. It also offers a superior integrated scheduling calendar for easy coordination of sessions, a powerful browsing and search system to quickly match students with suitable tutors, and the convenience of meeting on campus in familiar spaces like the library or CS lab. These tailored features make our platform both more secure and more relevant to the SFSU community, giving it a competitive edge over broader, less personalized tutoring solutions.

8. High-level system architecture and technologies used

- Server Host: Amazon AWS EC2, Instance: t3.micro
- Operating System: Ubuntu 24.04 LTS
- Database: MySQL 8.0.43
- Web Server: NGINX 1.24.0
- Frontend Tools: HTML5, CSS3, JavaScript ECMAScript 2025
- Server-Side Language: JavaScript with Node.js v18.19.1
- Additional Technologies:
    - Web Framework: Express v5.1.0
    - IDE: Visual Studio 2022
    - Web Analytics: Google Analytics GA4
    - Browsers:
        - Google Chrome v141.0 - v144.0
        - FireFox v144.0 - v147

9. Use of GenAI tools like ChatGPT and copilot for Milestone 1

- Executive Summary: ChatGPT was helpful in combining our goals, rough outline, and list of features into a professional introduction for the platform that is easy to read and represents our ideas accurately and concisely.
- High-level Use cases: ChatGPT was used to organize, and make it more professional and structured. The content, ideas were originally developed by author.
- High-level Functional Requirements: ChatGPT helped word the requirements in a more professional way. At first they were somewhat informal, but it allowed us to make them sound more professional.
- Competitive Analysis: ChatGPT was used to update the summary paragraph for the Comparison Table for a more professional and thoughtful summarization of our competitive analysis.
- High Level System Architecture - Useful in conjunction with internet searches to verify latest LTS versions of software that we are using.

10. Team and roles

Team 5:

Student Name	School Email	Role
Michael Thompson	mthompson17@sfsu.edu	Github Master
Tejas Rajan	trajan@sfsu.edu	Frontend Lead
Grady Walworth	wwalworth@sfsu.edu	Team Lead
Christopher Chan	cchan39@sfsu.edu	Backend Lead
Pei Huan Chang	pchang@sfsu.edu	Backend Developer
Kameron Jacob	kjacob2@sfsu.edu	Frontend Developer

11. Team Lead Checklist to be completed by team lead

• So far all team members are fully engaged and attending team sessions when required
DONE

• Team found a time slot to meet outside of the class
DONE

• Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing
DONE

• Team reviewed class slides on requirements and use cases before drafting Milestone 1
DONE

• Team reviewed non-functional requirements from “How to start...” document and developed Milestone 1 consistently
DONE

• Team lead checked Milestone 1 document for quality, completeness, formatting and compliance with instructions before the submission
DONE

• Team lead ensured that all team members read the final M1 and agree/understand it before submission
DONE

• Team shared and discussed experience with GenAI tools among themselves
DONE

• Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.)
DONE

Milestone 2

SW Engineering CSC648-848 Fall 2025

EduGator

Team 5:

Team lead:	Grady Walworth wwalworth@sfsu.edu
GitHub Master:	Michael
Frontend lead:	Tejas
Frontend developer:	Kameron
Backend lead:	Chris
Backend developer:	Hardy

Milestone 2 Part I

History Table:

Version 1.0	Due: 11/02/2025
Version 2.0	11/21/2025

Table of Contents

1. Executive Summary

Our project, EduGator, is a peer-to-peer tutoring platform designed exclusively for San Francisco State University students. It provides a trusted, intuitive space where members of the SFSU community can connect for academic support, share knowledge, and build ongoing learning relationships. The goal is to make finding or offering tutoring as simple and secure as connecting with a classmate.

Unlike generic tutoring apps or social media groups, EduGator verifies every user through their SFSU credentials, ensuring a safe and authentic academic community. The novelty of our approach lies in its focus on campus exclusivity, mutual trust, and recurring peer connections rather than one-time transactions. All tutoring is community-driven and non-commercial, emphasizing collaboration, mentorship, and academic growth over profit.

When students register, the platform automatically tailors their experience by pre-filtering content based on the subjects they are studying. Tutoring opportunities are organized by general subject areas (not specific course numbers) and distinguished by Lower Division and Upper Division levels to keep the system streamlined yet meaningful.A color-coded calendar makes scheduling simple. Students can view tutor availability at a glance, filter by subject, and choose between Drop-in sessions (open group tutoring) or Appointments (1-on-1 meetings). The integrated search bar allows users to find sessions by subject, tutor name, or day

By funding this project, the university will empower students to support one another academically, reduce learning barriers, and foster a stronger, more connected campus culture. EduGator not only enhances student success but also reinforces the values of accessibility, inclusivity, and community service that define SFSU.

Our team is made up of passionate San Francisco State University students who want to help our peers succeed. EduGator is built by students, for students.Our goal is to create something that truly reflects the spirit of SFSU. We see EduGator as more than an app, it’s a step toward building a stronger culture of mentorship and mutual growth across our university.

2. List of main data items and entities

- **Admin**:
    - can access all data and content and modify the database. Needs to login/register
    - Allows for managing users, subjects/courses, sessions, disputes, payments, site content, and analytics.
    - Can impersonate users for support. Requires login/registration with elevated privileges.
- **Tutor**
    - Has their own profile, can set availability, offer session types/prices, accept/decline bookings, can deliver sessions, and record outcomes/notes.
    - Requires login/registration and verification.
- **Student**
    - Has their own profile, can search tutors, view profiles, book sessions or packages, attend sessions, message tutors, and manage cancellations.
    - Requires login/registration for booking.
- **Unregistered User**
    - Does not have an account
    - Can browse public content (tutor summaries, subjects, available), initiates signup.
    - Can see available times
    - No booking/messaging until registered.
- **Subject**
    - Represents a general academic area (e.g., “Mathematics,” “Computer Science”).
    - Used to categorize tutors and sessions for easier searching and filtering.
    - Ensures consistent organization across all tutoring Posts.
- **Post**
    - Created by tutors to advertise availability for tutoring sessions.
    - Includes details such as subject, date/time, description, and session type (Drop-in or Appointment).
    - Acts as the primary way tutors make themselves discoverable to students.
- **Session**
    - A confirmed tutoring meeting between a tutor and a student.

    - Contains information such as participants, subject, location, time, and status.
    - Can originate from a Post or direct booking request.
    - Generates confirmation and reminder notifications for both parties.
- **Calendar**
    - The central scheduling feature shows all available and booked sessions.
    - Allows users to filter by subject, tutor, or color-coded session type.
    - Tutors and students have personalized views to manage their schedules.
- **Review / Feedback**
    - Submitted by students after a tutoring session.
    - Contains a rating, written comments, and a timestamp.
    - Supports quality assurance and helps other students choose tutors.
- **Profile**
    - Displays personal and academic details of a user.
    - Tutor profiles include expertise, bio, and session history.
    - Student profiles may show booked sessions or preferred subjects.
    - Accessible only to authenticated users.
- **Analytics Record**
    - Aggregated data used by admins to analyze system usage.
    - Tracks metrics such as active users, popular subjects, and session frequency.
    - Does not contain personally identifiable information.

3. Functional Requirements - Prioritized

Priority 1 (must have):

Unregistered Users:

1. Unregistered users shall be able to browse tutors and view public tutor profiles.
2. Unregistered users shall be able to register as a student or tutor using their SFSU email.
3. System shall redirect unregistered users to the login/register page when attempting to access certain features.

5. Unregistered users shall be able to view available tutoring subjects and courses.

24. Registered users shall be able to send and receive messages with tutors.

Registered Users (Students and Tutors):

6. Users shall be able to sign in using their SFSU email.
7. Users shall be able to create their personal profile.
9. Tutors shall be able to post their available tutoring times.
10. Tutors shall be able to receive notifications for requests, and confirmations.
11. Tutors shall be able to state the courses and subjects they offer.
12. Students shall be able to view a tutor's expertise.
13. Students shall be able to filter tutors by course, subject, keyword, or availability.

System/Admin:

19. System shall create a record when a tutor creates a tutoring session; this session shall appear on the student’s calendar.
20. Admins shall be able to approve or reject tutor applications.

Priority 2 (desired):

Unregistered Users:

Registered Users (Students and Tutors):

8. Users shall be able to edit their personal profile.
14. Students shall be able to cancel their tutoring requests.
15. Students shall be able to reschedule their tutoring requests.
17. Tutors shall be able to update previously posted tutoring sessions.
18. Tutors shall be able to remove previously posted tutoring sessions.

System/Admin:

21. Admins shall be able to delete tutoring sessions.
23. System shall notify tutors when a tutoring request is made.

Priority 3 (opportunistic):

Unregistered Users:
N/A

Registered Users (Students and Tutors):

16. Students shall be able to rate and provide feedback about the tutors.

System/Admin:

22. Admins shall be able to remove user accounts that violate policies.

4. UI Storyboards for each Main Use Case

Use Case 1: Searching for Tutors by Name

Actors: Alex (SFSU Student)

Description:

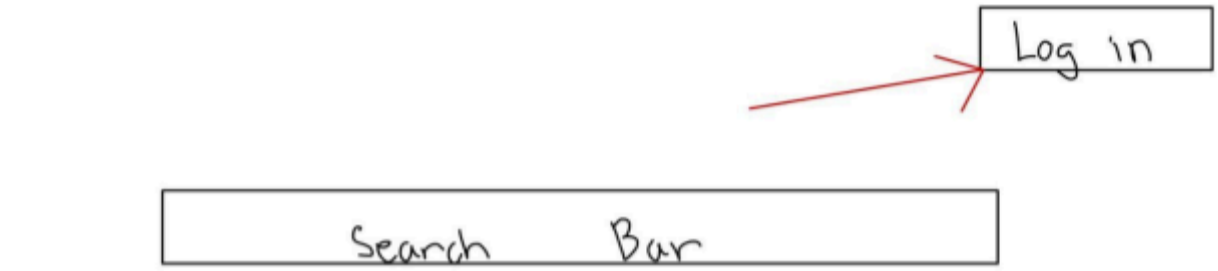
An **SFSU Student** logs into the tutoring platform and searches for a specific **Tutor** by name using the search bar. The system displays matching **Tutor** profiles and their availability from the **Calendar**. The student can view each **Profile** for more details such as ratings and **Review / Feedback**, then proceed to filtering or booking.

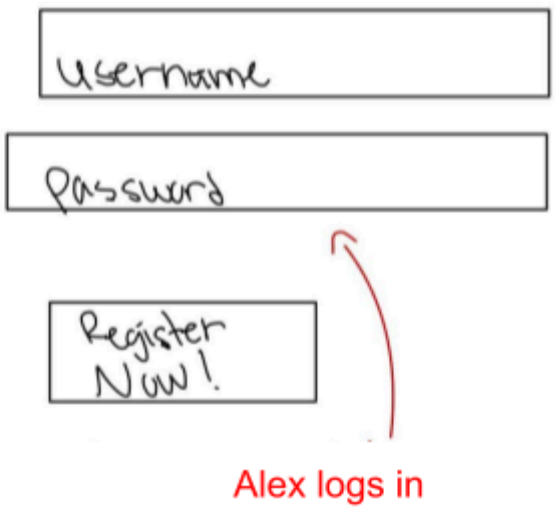
Frequency/Importance: Weekly; important for quickly locating preferred tutors.

Environment/Context: Accessed via desktop or mobile browser through the website.

Story Board:

Searching for Tutors by Name





Searching for Tutors by Name

Log in

Search Bar

search for tutor

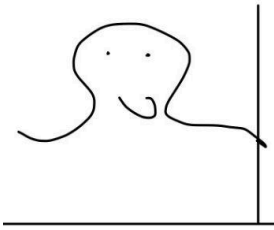
John Beige

John Beige ★Rating

Availability: Mon-Fri 8am-5pm

Results: 1

Click on profile



John Beige ★/5 rating

About me:

Booking filters:

Available Bookings:

Reviews/Feedback:

Use Case 2: Filtering Tutors

Actors: Alex (SFSU Student)

Description:

After performing a search, the **Student** refines results using filters such as **Subject**, availability, and **Review / Feedback** rating. The system dynamically updates the list to show **Tutor**s matching the selected filters and displays their earliest available sessions from the **Calendar**. This enables students to efficiently find suitable academic support without browsing irrelevant results.

Frequency/Importance: Weekly; enhances the efficiency of tutor discovery and booking.

Environment/Context: Used from dorms, libraries, or off-campus via web browser or mobile device.

Story Board:

Filtering Tutors

Log in

Search Bar

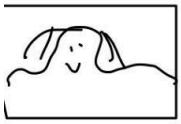
Filter Results
Search by subject/ Ratings, etc

search for tutor

John Beige

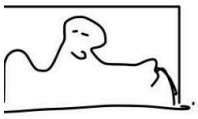
John Beige ★ Rating

Availability: Mon - Fri 8am - 5pm



Katy Barne ★ Rating

Availability: Sat - Sun 5pm - 8pm



Joey Svent ★ Rating

Availability: Thurs - Sat 9am - 5pm

Results: 3

Use Case 3: Guest Browsing

Actors: Jordan Lim (Guest User)

Description:

A **Guest User** visits the tutoring website to explore available **Tutor**s and **Subject**s. The system allows the guest to search for **Tutor**s by name and apply filters such as **Subject**, availability, or rating to refine results. Guests can view public content, including **Tutor** summaries, general **Post**s, and open time slots on the **Calendar**. When a guest attempts to view a full **Profile** or book a **Session**, the system prompts: “Only registered SFSU members can continue. Please log in with your SFSU email.” Upon logging in, the user is returned to their original page to continue seamlessly. This encourages exploration while maintaining SFSU exclusivity.

Frequency/Importance: Daily; supports user onboarding and smooth transition to full access.

Environment/Context: Publicly accessible on desktop or mobile browsers.

Story Board:

Guest Users

Log in

user has not signed in and is just looking

Search Bar

Filters:
subject
availability
Rating

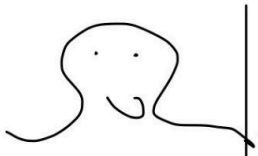
searches for tutor

Avery Hignot ★Rating

Availability: Sun - Thurs 8am-5pm

Results: 1

clicks on profile



Avery Hignot

logs in → Log in

★/5 rating

About me:

Booking filters:

Available Bookings:

Only Registered SFSU members can continue. Please log in with your SFSU email

Reviews/Feedback:

Searching for Tutors by Name

Log in

user is now logged in

Search Bar

Username

Password

Register Now!

Use Case 4: Booking a Tutoring Session

Actors: Alex (SFSU Student), Jordan Lim (Guest User), Tutor

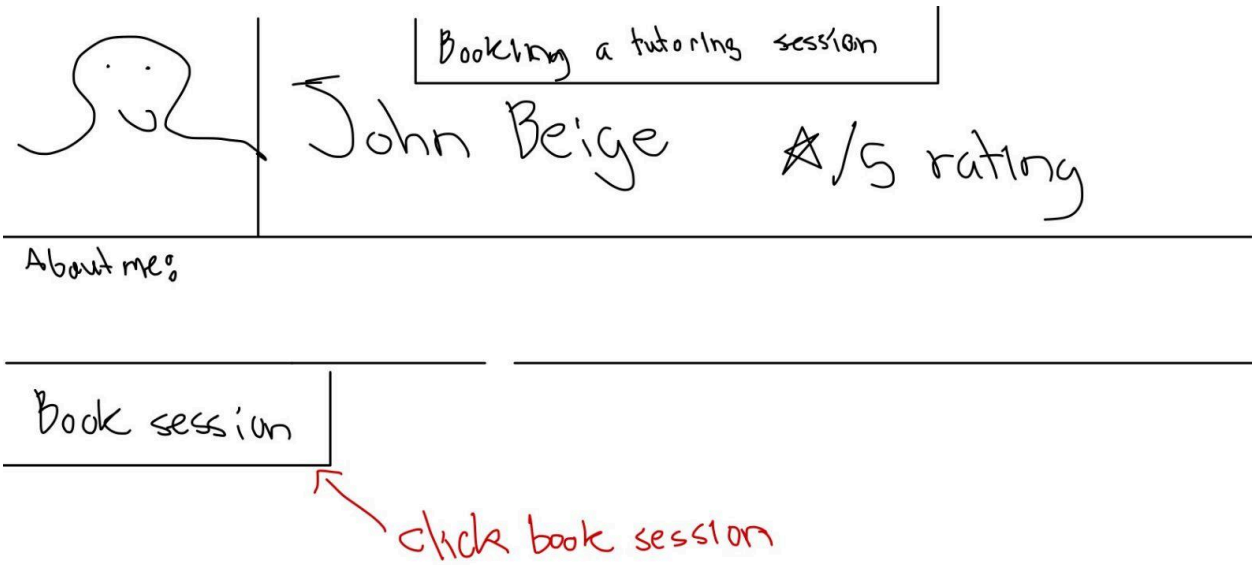
Description:

An **SFSU Student** selects a preferred **Tutor** from the search results or **Profile** and clicks “Book Session.” The system displays the **Tutor**’s available slots from the **Calendar**, allowing the student to choose a convenient time. A confirmation panel shows **Session** details including **Subject**, session type (from **Post**), and duration. Once confirmed, the **Session** appears on both the **Tutor** and **Student Calendar**s. If a **Guest User** attempts to book, a prompt requests SFSU login before proceeding.

Frequency/Importance: Weekly; essential for connecting students to personalized academic support.

Environment/Context: Used via website on any browser.

Story Board:


Booking a tutoring session
John Beige
★/5 rating
About me:
Book session
click book session
Reviews/Feedback:

Calendar

1	2	3	4	5	6	7
8	9	10	11 8am-5pm	12	13	14
15	16	17	18 clicks on this one	19	20	21
22	23	24	25	26	27	28
29	30	31				

9am 11am

Calculus II

Tutoring session

if user is a guest prompts login option

Confirm Booking

Use Case 5: Staff Management of Tutoring Sessions

Actors: Dr. Maria Lopez (SFSU Staff), Tutor

Description:

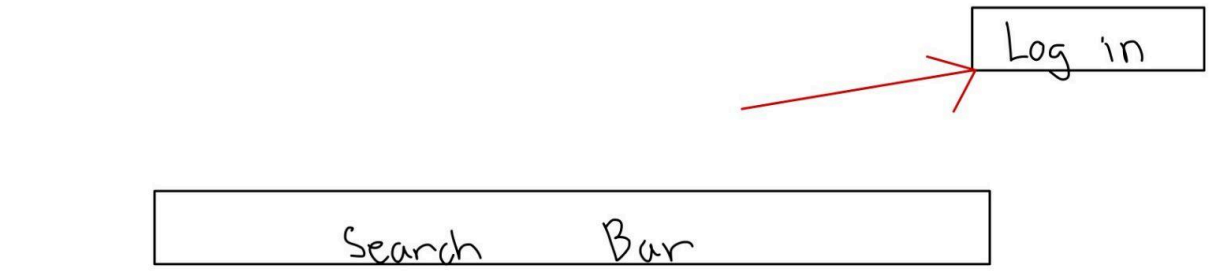
Dr. Lopez, a **Tutor** on the platform has a sudden dental appointment and can't make her Thursday afternoon tutoring session. She logs in to the tutoring platform, navigates to the **Calendar** page, and cancels her scheduled **Session.** This action automatically sends a message to the corresponding **Student** that was signed up for the **Session**.

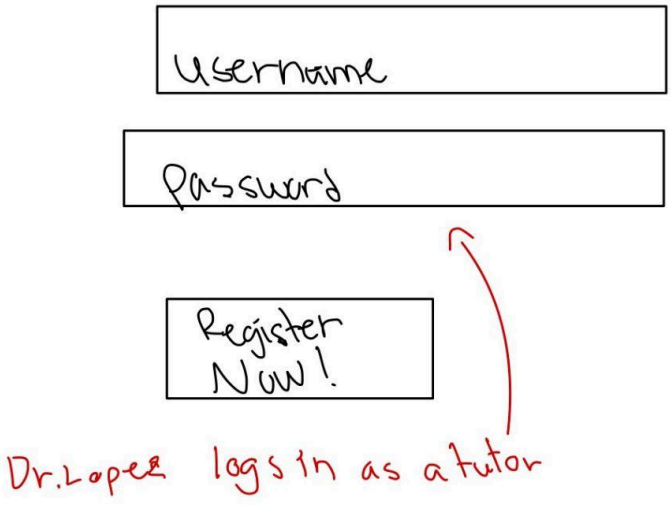
Frequency/Importance: Weekly; critical for maintaining accurate schedules and supporting academic success.

Environment/Context: Accessed from office desktops or remotely through the website.

Story Board:

Searching for Tutors by Name

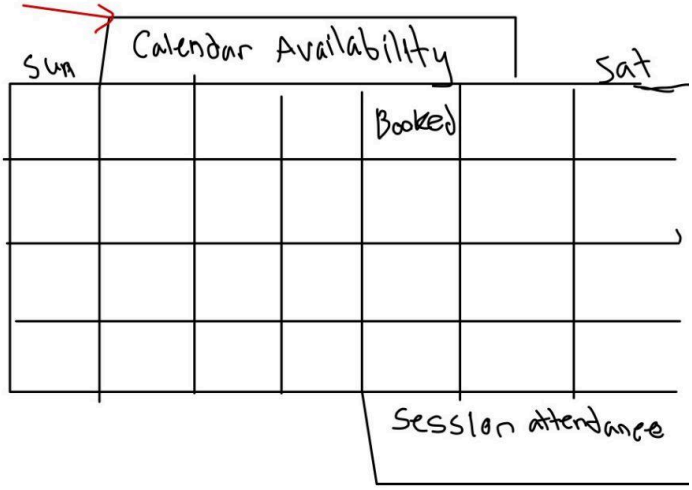




Staff Management of Tutoring Sessions

Dr. Lopez

Dr. Lopez clicks on the Calendar availability



- Create a post
- edit previous post
- Review Feedback

Calendar

Tutor clicks on scheduled session

Dr. Lopez

Sun	Mon	Tues	Weds	Thurs	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12 1pm-3pm session	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

1pm-3pm

Calculus II

Tutoring session

Sends cancelation email to student

Cancel session

5. High level Architecture, Database Organization summary

Database Organization Summary:

- User
    - User_id (PK)
    - Email (unique must end with @sfsu.edu)
    - Password_hash
    - First_name
    - Last_name
    - Profile_picture (FK to files)
    - Role (FK to role)
- Role
    - Role_id (PK)
    - Name (ENUM: Student, Tutor, Admin)
- Tutor_profiles
    - User_id (PK FK to user)
    - Years_experience
    - Verification_status
    - Resume (FK to files)
    - Hourly_rate
    - Description (used for tutor search)
    - bio
- Student_profiles
    - User_id (PK FK to user)
    - Major
    - Academic_level (ENUM: Freshman, Sophomore, Junior, Senior, Grad)
    - GPA
    - Expected_graduation_date
    - Bio
- Student_courses
    - User_id (PK FK to user)
    - Course_id (PK FK to courses)
    - Status (ENUM: Current, planned, completed)
    - Term_label
    - Added_at

    - Instructor_name
    - description
- Tutor_subjects
    - User_id (PK FK to user)
    - Subject_id (PK FK to subject)
- Tutor_courses
    - User_id (PK FK to user)
    - Course_id (PK FK to courses)
- Subject
    - subject_id (PK)
    - Name (unique)
    - Description
- Department
    - Department_id (PK)
    - Code (ENUM: CSC, MATH, etc.)
    - name
- Courses
    - course_id (PK)
    - Department_id (FK to department)
    - Course_title
    - description
- Sessions
    - Session_id (PK)
    - Tutor_id (FK to user)
    - Start_time
    - End_time
    - Session_type (ENUM: open, one_on_one)
    - Capacity
    - location_details
    - Status (ENUM: Scheduled, Active, Over)
    - Created_at
    - Updated_at
- Session_attendees
    - Session_id (FK to session)
    - Student_id (FK to user)
    - Join_status (ENUM: Approved, Pending, Declined, Waitlisted)
    - joined_at
- Requests
    - Request_id (PK)
    - Student_id (FK to user)

    - Tutor_id (FK to user)
    - Request_type (ENUM: Join_session, one_on_one_request)
    - Session_id (FK to session)
    - message
    - Created_at
    - Updated_at
    - Status (ENUM: Accepted, Declined, Pending)
- Notifications
    - Notification_id (PK)
    - User_id (FK to user)
    - Type (ENUM: request_created, request_accepted, session_canceled)
    - Payload (JSON)
    - Created_at
- Messages
    - Message_id (PK)
    - Send_id (FK to user)
    - Receiver_id (FK to user)
    - Message_content
    - time_sent
- Reviews
    - Review_id (PK)
    - Session_id (FK to session)
    - Student_id (FK to user)
    - Tutor_id (FK to user)
    - Rating (1-5)
    - Comment
    - Created_at
- Tutor_ratings
    - User_id (FK to user)
    - Rating_avg (calculated from reviews.rating based on the tutor_id)
    - rating_count
- Files
    - File_id (PK)
    - Owner_user_id (FK to user)
    - Url
    - created_at

Media storage:

- We will be storing media in the file system, keeping the URLs/path in MySQL instead of the BLOBs.
- The database itself will only store the path to the uploaded media. This will be better performance, and scalability wise. It makes it much faster to retrieve the media files when they're stored in this way rather than keeping them in the database.

Search/filter architecture and implementation:

We want to allow for efficient search of tutors and sessions using MySQL queries and %like.

- What users can search/filter
    - Text search:
        - Matches (with %like):
            - Users.first_name, users.last_name
            - subjects.name
            - Tutor_profiles.description
    - Tutors:
        - Verified only
        - Subject name (subjects.name)
        - Minimum rating (tutor_ratings.rating_avg)
    - Sessions
        - Verified only (through the tutor)
        - Subject name (subjects.name)
        - Minimum tutor rating
        - Date window (start_time >= startDate, end_time <= endDate)

Significant non-trivial processes:

- Conflict detection:
    - Before a Tutor creates an available tutoring session, ensure there are no overlaps with their previously created sessions.
    - Before a Student requests to join a Tutoring session, ensure they have not already joined another session during the same time.
    - If there is an overlap, give some sort of warning.

- Seat control: For open tutoring sessions, prevent approval of more students if there are no more available seats.
- Visibility rules: Only display approved tutors and scheduled sessions to users.

New SW tools:

- Bcrypt to hash stored passwords in the database

6. Identify key risks for your project at this time

Skills Risks:

Risk: Some backend and frontend team members are still becoming familiar with key technologies such as AWS, MVC architecture, the vertical prototype workflow, MySQL, and API integration. Differences in individual technical proficiency may slow down the early phase of development.

Solution: The backend and frontend leads will provide sample code, hold short internal walkthrough sessions, and maintain shared documentation to ensure all members understand their assigned technical tasks.

Schedule Risks:

Risk: All team members are balancing this project alongside other coursework and responsibilities. This may limit the amount of time available for long, complex development sessions, potentially delaying progress if tasks become too large or interdependent.

Solution: The team will break larger objectives into smaller, manageable tasks that can be completed incrementally. Internal deadlines will be set earlier than the official due date, and progress will be tracked with our task management page and weekly check-ins to ensure steady, sustainable progress throughout each milestone.

Technical Risks:

Risk: Integration issues may arise between frontend calls and backend API endpoints if parameter names, data formats, or routes differ from expected behavior. Additionally, designing a universal data access API for consistent interaction with the MySQL database may prove complex.

Solution: The team will use an API contract (shared JSON schema and endpoint documentation) to standardize data exchange. Incremental integration tests will be run after each new endpoint to confirm compatibility. A dedicated data access layer will be created and reviewed early to avoid redundant or inconsistent database queries.

Teamwork & Communication Risks:

Risk: The frontend and backend teams may complete their assigned tasks at different times. For example, the frontend may finish UI mockups or page layouts before the backend APIs and database are fully ready. This timing mismatch can lead to idle waiting.

Solution: To reduce timing mismatches, the team will establish a shared development timeline identifying when UI mockups, database schema, and APIs will be ready. The frontend will begin with static pages and placeholder data, while the backend provides early mock API responses for testing.

Content & Legal Risks:

Risk: Using unlicensed tutor photos, course material, or personal information in prototypes may violate copyright or privacy policies. Storing unnecessary personal data may also introduce security concerns.

Solution: The team will use only team-created or royalty-free content for Milestone 2. The database will store only minimal test data and avoid real personal information.

7. Project Management

For Milestone 2, our team established a shared Google Doc specifically for task management and milestone planning. Each milestone was broken down into smaller, well-defined steps, and a task table was created listing every deliverable, the team member responsible, and the corresponding deadline. This structure ensured that all tasks were clearly assigned, progress could be tracked transparently, and accountability was maintained across the team. Each member regularly updates their task status, allowing everyone to stay informed on what has been completed and what still needs attention.

We communicate primarily through Discord, which serves as our central hub for collaboration. We maintain separate channels for frontend, backend, and general discussions, allowing quick clarification, technical troubleshooting, and reassignment of work when necessary. Team members also use Discord to share progress updates, screenshots, and development tips throughout the week, ensuring that blockers are identified early.

8. Use of GenAI tools like ChatGPT and copilot for Milestone 2

- Used GenAI to help speed up some of the frontend changes. Helped with adding the search bar to every page, and cleaning up the login page.
- GenAI helped with some of the database organization, and helped explain the search algorithm.
- ChatGPT used to polish the key risks and help separate the skill risks from the technical risks.

9. Team Lead Checklist to be completed by team lead

• So far all team members are fully engaged and attending team sessions when required
DONE

• Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing
DONE

• Team reviewed suggested resources before drafting Milestone 2
DONE

• Team lead checked Milestone 2 document for quality, completeness, formatting and compliance with instructions before the submission
DONE

• Team lead ensured that all team members read the final M1 and agree/understand it before submission
DONE

• Team shared and discussed experience with GenAI tools among themselves
DONE

Milestone 3

SW Engineering CSC648-848 Fall 2025

EduGator

Team 5:

Team lead:	Grady Walworth wwalworth@sfsu.edu
GitHub Master:	Michael
Frontend lead:	Tejas
Frontend developer:	Kameron
Backend lead:	Chris
Backend developer:	Hardy

Milestone 3

History Table:

Version 1.0	11/17/2025

Appendix II: M3 meeting review summary and development plans template – to be filled out by the team immediately after M3 review meeting

This summary must be done as soon as M3 meeting review is done to document the findings and conclusions (best do it after the meeting). Instructor does not need to see this document until it is submitted with final M5 folder for final grading but will check for its completion via e-mail checkpoint to team leads. This document is used to guide team plans for the rest of the class – team from then on will be focused only in agreed upon P1 features – e.g. is in "feature freeze" mode meaning all team members work ONLY on P1 features on their task list. In the parlance of SCRUM SE project management this list of tasks constitutes your "sprint backlog", see https://en.wikipedia.org/wiki/Scrum_(software_development)#/media/File:Scrum_Framework.png (we will cover it in SE Processes segment).

• IMPORTANT for selection of P1 features: analyze what needs to be done, prioritize based on two factors: a) importance for the product/user and b) cost/ability to deliver it in given schedule. Based on this come up with the plan (list of P1 features) then execute it. **After this the team is in "feature freeze" mode, focus is on P1 features only. All listed P1 features (no more no less) MUST be delivered in usable way, free of bugs and will be used for grading**

Team leads: ensure a) all tasks are covered; b) all team members have tasks assigned; c) strictly manage deliverables and have checkpoints along the way to ensure progress

Template for this document is below. Be concise (1-2 line per issue is OK) but ensure you record all issues that apply. You also must fill in section with P1 priority functions

Summary of Milestone 3 meeting review with instructor and plans for further development

Team number: Team 5

Meeting date: 11/14/2025

• **Summary of feedback on UI (record all pages that need revision)**

Search

- Filtering needs to go on the left side, need to include time option
    - Keep the subject filter, but put it next to the search bar
    - ability to filter by tag/specific course? (i.e. “CSC 317”), which would help with the number/length of tags on tutor cards

- remove the term “**scheduled**” from sessions search / profile, its confusing as it’s similar to, but not the same as, the term “**booked**”
    - maybe rework “Open” wording
- For Open sessions
    - Include Capacity, and Current students booked. Once its full, remove from search
- For One-on-One sessions, possibly remove capacity, as its redundant (low priority)
- remove fully **booked** (either one-on-one, or a fully booked group) sessions from the search, as they just get in the way

On a student profile

- remove Personal Identifying Information (student ID) from the profile page
    - keep email, name, and major
- Courses are current semester, set a date to dump courses at the end of each semester, so legacy courses aren’t left in
- Need to add “add courses” functionality on student profile
- Add button to “become a tutor”, see Registration below

On a tutor profile

- Put available sessions at the top, unavailable/booked sessions at the bottom (in red)
- Funnel all bookings into the sessions details page first, then it can be booked for real

- Current tags are too vague:
    - Make the tags into smaller, SFSU specific tags, either:
        - Course numbers and grouping tags, but put a limit on how many actually show in the search card
        - consider a cap of number of courses a tutor can tutor
        - upper division vs lower division courses

Registration

- make the registration the same for both, then on the students profile, make a button to “become a tutor”

Inbox

- remove drafts
- remove trash
- consider removing alerts, or make it modal, so the browser checkbox alert doesn't come up

Navbar

- Change the inbox button, the icon looks like an unrecognized symbol for a font

Calendar

- put the color key/legend at the top of the page, add another color for student schedule
    - make it clickable, so it can show/hide items on the calendar
- think of the worst case scenario, what happens if a lot of cells go on one day, (tutoring, classes, etc), how will that be handled?

• Summary of feedback on code and architecture
N/A

• Summary of feedback on GitHub usage
Need to work on distributing the commits more evenly.

• Summary of feedback on DB
N/A

• Summary of feedback on teamwork and risk management
Good teamwork and our technical and teamwork risks are negligible, just need to manage the timeline as we all approach finals.

• Confirm that you have done **architecture review** to check that developers adhere to MVC pattern, coding style, minimal agreed documentation etc. **Record if OK or list the issues found**. Request developers follow up on corrections and follow up later by doing code reviews.

Architecture Review is OK.

• List below agreed upon P1 list of features for final delivery which constitute product plan.
NOTE: after this meeting the team focuses solely on this P1 list of features, e.g. the development is in “feature freeze mode”. **All listed P1 features (no more no less) MUST be delivered in usable way, free of bugs and will eb used for grading**

Our agreed upon P1 list:

Unregistered Users:
Unregistered users shall be able to browse tutors and view public tutor profiles.
Unregistered users shall be able to register as a student or tutor using their SFSU email.
System shall redirect unregistered users to the login/register page when attempting to access certain features.

5. Unregistered users shall be able to view available tutoring subjects and courses.
24. Registered users shall be able to send and receive messages with tutors.

Registered Users (Students and Tutors):

6. Users shall be able to sign in using their SFSU email.
7. Users shall be able to create their personal profile.
9. Tutors shall be able to post their available tutoring times.
10. Tutors shall be able to receive notifications for requests, and confirmations.
11. Tutors shall be able to state the courses and subjects they offer.
12. Students shall be able to view a tutor's expertise.
13. Students shall be able to filter tutors by course, subject, keyword, or availability.

System/Admin:

19. System shall create a record when a tutor creates a tutoring session; this session shall appear on the student’s calendar.
20. Admins shall be able to approve or reject tutor applications.

• Any other comments and issues
Need to make sure unregistered user is directed to log in before they are able to book a session.

• Check Point (CP) if given, DUE
N/A

Milestone 4

SW Engineering CSC648-848 Fall 2025

EduGator

Team 5:

Team lead:	Grady Walworth wwalworth@sfsu.edu
GitHub Master:	Michael
Frontend lead:	Tejas
Frontend developer:	Kameron
Backend lead:	Chris
Backend developer:	Hardy

Milestone 4

History Table:

Version 1.0	Due: 12/15/2025

1). Product Summary:

EduGator

EduGator is a peer-to-peer tutoring platform designed exclusively for San Francisco State University students. Its purpose is to make finding academic support as simple and trustworthy as asking a knowledgeable classmate. Every user signs in with their SFSU credentials, ensuring a safe, authentic environment where students can confidently connect for help or offer guidance to others. EduGator stands apart from generic tutoring services by focusing on community-building, campus exclusivity, and ongoing mentorship rather than one-time, commercial transactions.

The platform automatically tailors each student's experience by filtering subjects based on what they are studying, and tutoring opportunities are organized by broad subject areas and divided into Lower and Upper Division levels. A color-coded calendar makes scheduling effortless, allowing students to browse tutor availability, filter by subject, and choose either Drop-in group sessions or one-on-one appointments. An integrated search bar enables quick access to sessions by subject, tutor name, or day. Built *by SFSU students for SFSU students*, EduGator strengthens academic success, reduces learning barriers, and fosters a supportive campus culture rooted in mentorship, collaboration, and mutual growth.

Final P1 Functional Commitment (Major Committed Functions)

For Unregistered Users

- Can browse tutors and view public tutor profiles.
- Can register as a student or tutor using an SFSU email.
- Are redirected to log in or register when trying to access restricted features.
- Can view available tutoring subjects and general course areas.

Messaging

- Registered users can send and receive messages with tutors.

For Registered Users (Students and Tutors)

- Can sign in using their SFSU email.
- Can create their personal profile.
- Tutors can post their available tutoring times.
- Tutors receive notifications for requests and confirmations.
- Tutors can list the subjects and courses they offer.
- Students can view a tutor's expertise.
- Students can filter tutors by subject, course, keyword, or availability.

System / Admin

- The system creates a tutoring session record when a tutor posts availability, and this session appears on the student's calendar.
- Admins can approve or reject tutor applications.

URL to your product accessible to instructors, on deployment server:

http://18.217.207.30/

2) Usability test plan for selected function – about 2 pages

Selected Function

Messaging feature (sending and reading messages between students and tutors)

1. Test Objectives

The objective of this usability test is to evaluate the messaging feature of the EduGator platform. This test focuses on whether users can easily read messages, understand message content, and send replies to tutors. The goal is to identify any usability issues that may affect communication between students and tutors.

2. Test Background and Setup

System setup and starting point

The system under test is the EduGator web application. The messaging feature is available after a user logs into the system. For this test, the tester will start from the main dashboard page and navigate to the Messages (Inbox) section. The tester must have access to a computer or laptop with an internet connection and a modern web browser.

Intended users

The intended users are college students who use EduGator to communicate with tutors. The users are assumed to have basic experience using web-based messaging systems such as email or chat platforms but do not need any technical background.

Test environment

The usability test will be conducted remotely in the tester's home environment. No cameras or screen recording tools are required. The tester will not be monitored during the test. No training or prior instructions will be given before the test to better simulate a first-time user experience.

3. Usability Task Description

Before starting the test, the tester will be informed that the goal is to complete the tasks naturally and provide honest feedback afterward. The tester may ask clarifying questions only after all tasks are completed.

Task 1:

Navigate to the Messages section and locate a message from a tutor regarding a confirmed tutoring session.

Task 2:

Open the message and read the session details, including the tutor name, date, and time.

Task 3:

Reply to the tutor with a short message asking a follow-up question about the session.

Task 4:

Confirm that the sent message appears in the Sent messages section.

After completing all tasks, the tester will proceed to fill out the usability questionnaire.

4. Plan for Evaluation of Effectiveness

Effectiveness will be measured by observing whether the tester can successfully complete each task without assistance. Task completion rates and the number of errors will be recorded. A task is considered successful if it is completed correctly within the system.

5. Plan for Evaluation of Efficiency

Efficiency will be evaluated by measuring the time taken to complete each task and the number of steps required. The tester’s navigation path will be analyzed to identify unnecessary actions or confusion. Faster task completion with fewer steps will indicate higher efficiency.

6. Plan for Evaluation of User Satisfaction (Likert Scale Questionnaire)

After completing the tasks, the tester will rate the following statements using a 5-point Likert scale

(1 = Strongly Disagree, 5 = Strongly Agree):

“It was easy to find and open messages in the messaging system.”

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

“I clearly understood the information presented in the messages.”

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

“Sending a reply message was simple and straightforward.”

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

7. GenAI Use

ChatGPT (GPT-5.2) was used to review and improve the usability test plan. The tool was used to check clarity, organization, and alignment with usability testing best practices discussed in class. GenAI helped refine task descriptions and ensure the Likert scale questions were properly formatted.

Example prompts used:

- Review this usability test plan for clarity and completeness.
- Help rewrite usability tasks in a user-instruction format.

Utility ranking of GenAI: MEDIUM

GenAI was helpful for reviewing structure and wording

3) QA test plan and QA testing - about 2 pages

1. Test Objectives

The objective of this QA test plan is to verify that the messaging feature in the EduGator system functions correctly and reliably. This includes ensuring that users can view messages, open message content, and send messages without system errors. The goal is to identify functional defects and confirm that the feature works as expected across different web browsers.

2. HW and SW Setup (including URL)

Hardware Setup

Laptop or desktop computer

Stable internet connection

Software Setup

Operating System: macOS or Windows

Web Browsers:

Google Chrome and Mozilla Firefox

3. Feature to be Tested

The feature under test is the Messages functionality within the EduGator web application. This feature allows students to view received messages, open message details, send new messages, and view sent messages. The QA testing focuses on verifying correct system behavior for these core messaging actions.

4. QA Test Plan

Test	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Open Inbox Messages	Verify that the Inbox loads correctly and displays received messages	User clicks on the Messages or Inbox link from the dashboard	Inbox page loads and a list of received messages is displayed	Chrome: PASS Firefox: PASS
2	View Message Content	Verify that a message can be opened and its content is displayed correctly	User clicks on a message from the Inbox list	Message opens and displays sender, subject, and message body	Chrome: PASS Firefox: PASS
3	Send Reply Message	Verify that a user can successfully send a reply message	User types a short message and clicks the Send button	Message is sent successfully and confirmation is shown	Chrome: PASS Firefox: PASS
4	View Sent Messages	Verify that sent messages appear in the Sent folder	User navigates to the Sent messages section	The previously sent message appears in the Sent list	Chrome: PASS Firefox: PASS

5. Cross-Browser Testing Results

All QA test cases were executed on two major web browsers: Google Chrome and Mozilla Firefox. The messaging feature performed consistently across both browsers. All test cases passed successfully, indicating that the feature functions correctly and does not exhibit browser-specific issues.

6. GenAI Use

ChatGPT (OpenAI, GPT-5.2) was used to assist in reviewing and refining the QA test plan. The tool was primarily used to improve the clarity of test case descriptions, ensure proper formatting of the QA table, and verify alignment with QA testing concepts discussed in class.

Example prompts used:

- Review this QA test plan for clarity and completeness.
- Help generate QA test cases for a messaging feature in a web application.

Utility ranking of GenAI: MEDIUM

GenAI was helpful for reviewing structure and wording, but all final decisions and testing results were based on actual system behavior and course requirements.

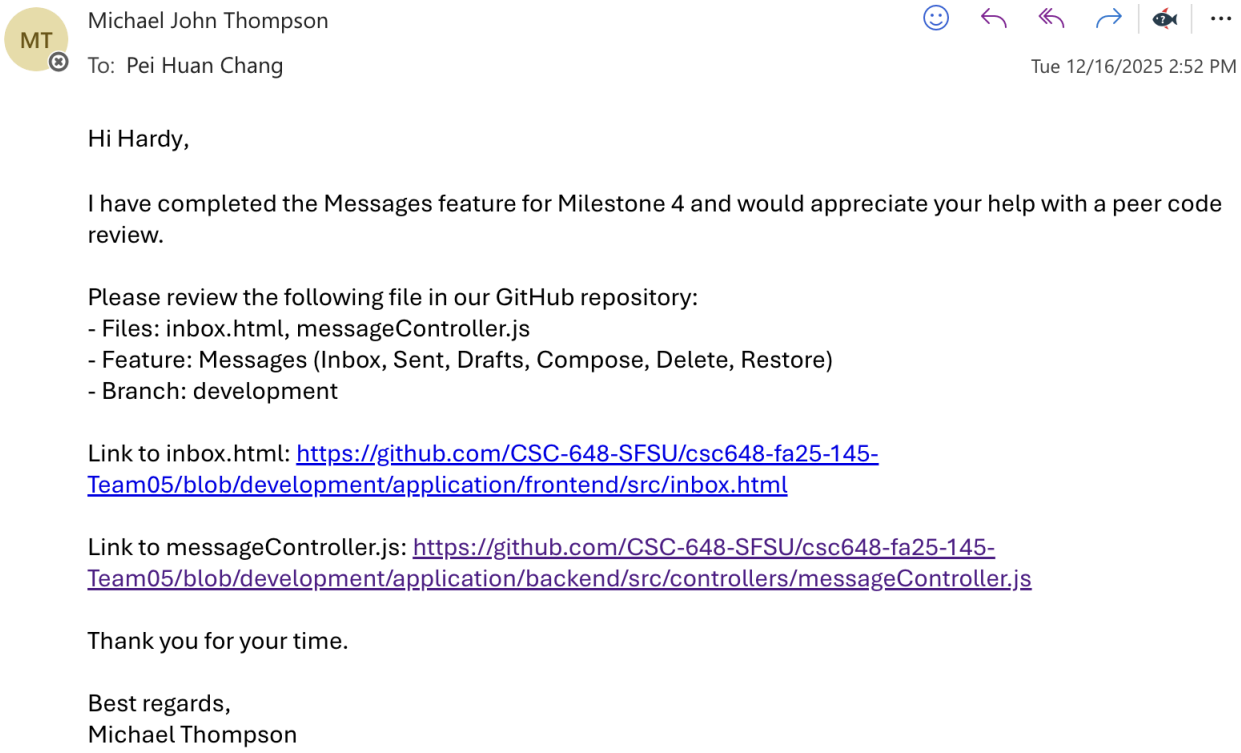
4) Peer Code Review:

1) Code under review: inbox.html

Feature: Messages (Inbox, Sent, Drafts, Trash, Compose, Reply, Delete, Restore)

Description: This file implements the front-end messaging functionality for the EduGator system, including inbox display, message detail view, composing messages, draft management, deletion, restoration, pagination, and session request handling.

2) Human Peer Review Process and Feedback



Michael John Thompson

To: Pei Huan Chang

Tue 12/16/2025 2:52 PM

Hi Hardy,

I have completed the Messages feature for Milestone 4 and would appreciate your help with a peer code review.

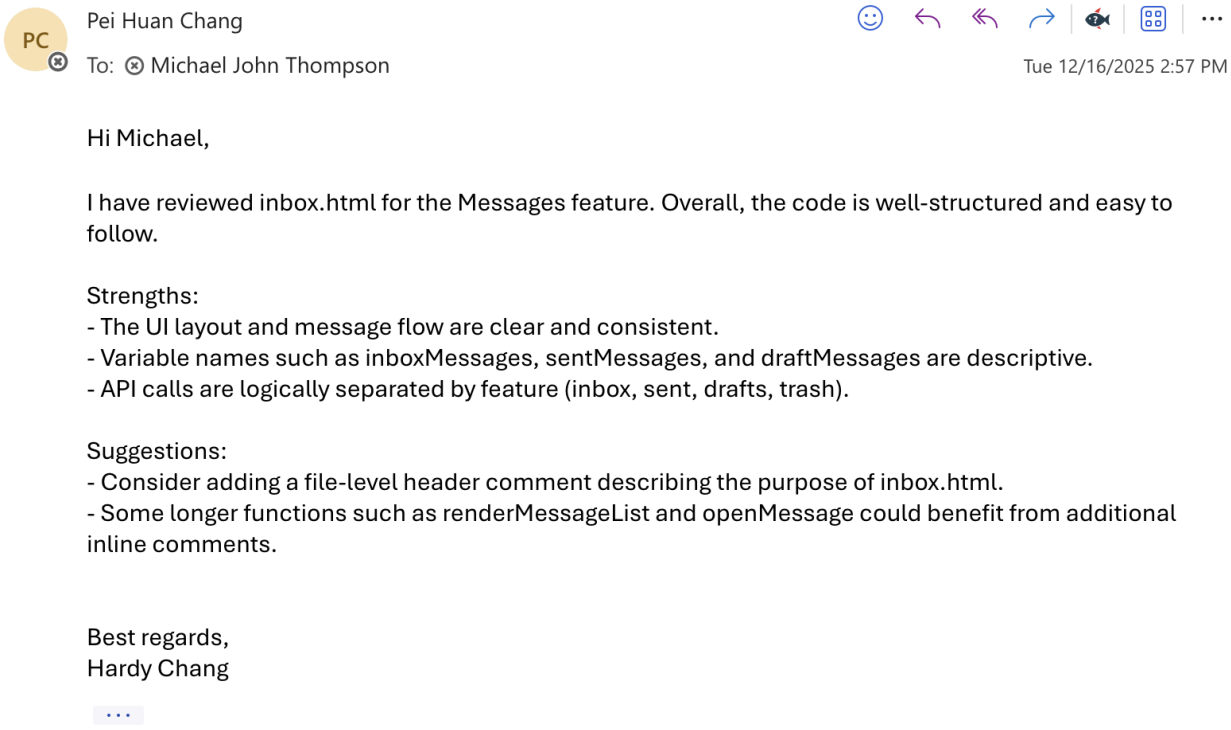
Please review the following file in our GitHub repository:
- Files: inbox.html, messageController.js
- Feature: Messages (Inbox, Sent, Drafts, Compose, Delete, Restore)
- Branch: development

Link to inbox.html: https://github.com/CSC-648-SFSU/csc648-fa25-145-Team05/blob/development/application/frontend/src/inbox.html

Link to messageController.js: https://github.com/CSC-648-SFSU/csc648-fa25-145-Team05/blob/development/application/backend/src/controllers/messageController.js

Thank you for your time.

Best regards,
Michael Thompson



3) Code Review Comment

Human Peer Review

A peer code review was conducted on the inbox.html file related to the Messages feature.

General Comments:

The overall code structure is clear and easy to follow.

The UI layout and message flow are consistent across Inbox, Sent, Drafts, and Trash views.

Variable names such as inboxMessages, sentMessages, and draftMessages are descriptive and consistent.

API calls are logically separated by feature, which improves readability and maintainability.

Suggestions:

Add a file-level header comment at the top of inbox.html describing the purpose of the file.

Add more inline comments to longer functions such as renderMessageList and openMessage to improve readability for future developers.

Consider standardizing error-handling messages across API calls for consistency.

The reviewer provided feedback via email and GitHub comments. Screenshots of the email exchange and code comments are included in the appendix.

GenAI Code Review

Tool used: ChatGPT (OpenAI, GPT-5.2)

GenAI helped identify areas where additional documentation could improve code readability, especially for longer functions and complex UI logic. It also confirmed that most variable names and function responsibilities were clearly defined.

Summary of GenAI Feedback

The GenAI review indicated that the overall structure of inbox.html is well-organized and suitable for a messaging feature. It suggested adding a file-level header comment and more inline comments in longer functions. GenAI also noted that the separation of concerns between message folders was clear and logical.

5) Self-check on best practices for security – ½ page

Asset to be protected	**Types of possible/expected attacks**	**Consequence of security breach**	**Your strategy to mitigate/protect the asset**
User Passwords	Brute force attacks, database leaks	data exposure, identity theft	passwords are encrypted with bcrypt, with 10 salt rounds before storage
Database	SQL injection, unauthorized access	Data corruption, loss, or unauthorized modification	Database credentials stored in environment variables, not in code
Search Function	SQL injection, resource exhaustion	Database compromise, system overload	Search inputs limited to 40 alphanumeric characters
Email Addresses	Unauthorized registration	Spam accounts, Database pollution	email validation enforces sfsu.edu domain, regex pattern ensures email address is alphanumeric with limited special symbols
User Input Fields	XSS attacks, code injection, buffer overflow	System compromise, malicious script execution	input length limits on all fields. Frontend AND backend validation for field length, datatypes and ranges

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 be rendered 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 (clients) can only message service providers via in-site messaging. One round of messaging (from client to service provider) is enough for this application. No chat functions shall be developed or integrated**
    DONE

14. **Pay functionality (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

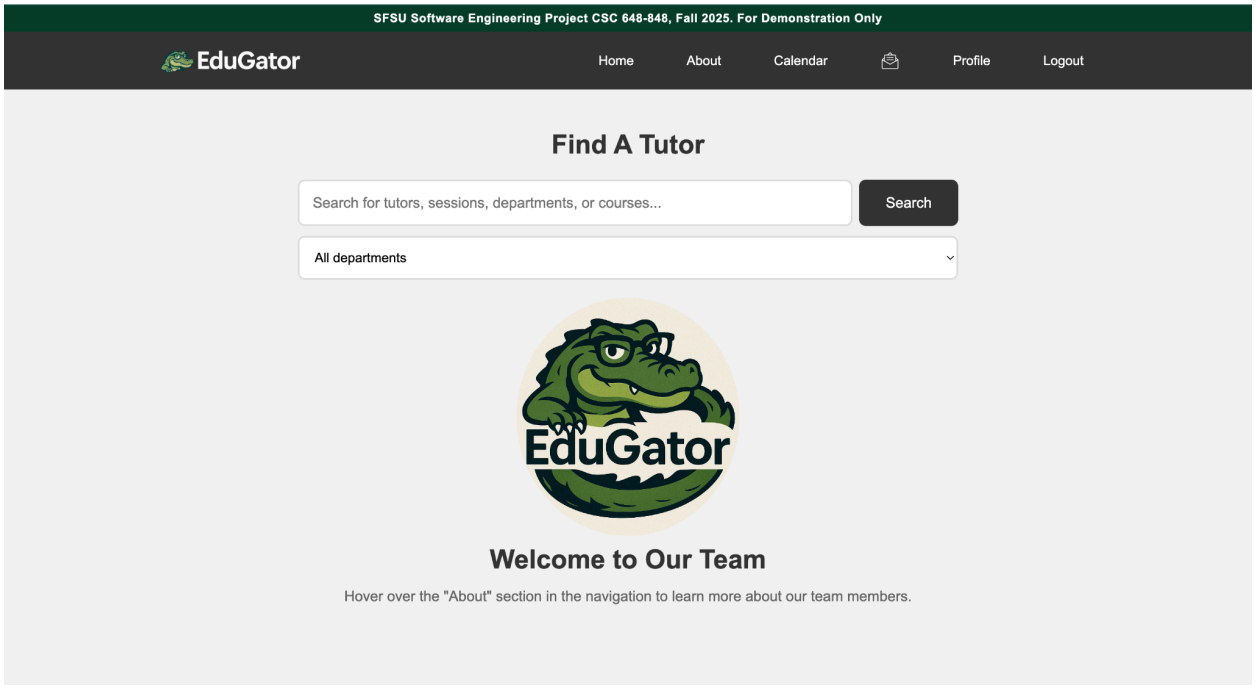
17. **Modern SE processes and tools shall be used as specified in the class, including collaborative and continuous SW development and GenAI tools**
    DONE

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 2025. For Demonstration Only" at the top of the WWW page Nav bar. (Important so as to not confuse this with a real application).**
    DONE

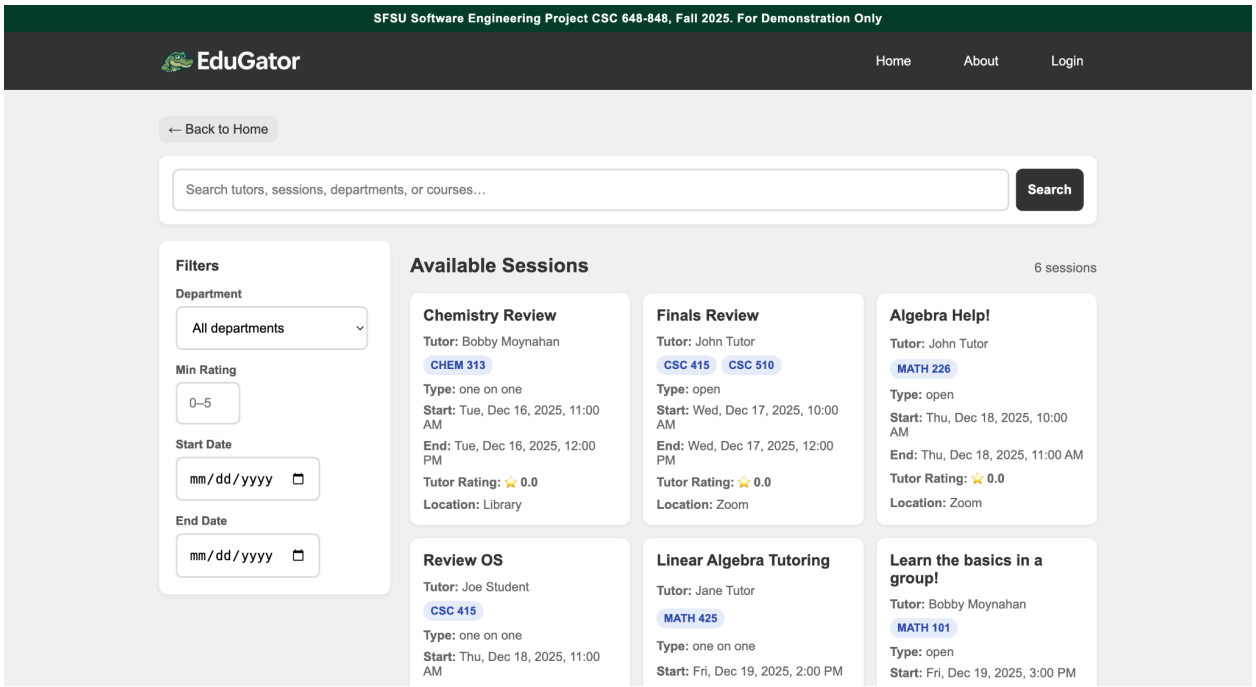
3) Product Screen Shots:

contain copies of all final screen shots of your team product (max 2 screen shots per page). B&W copies are OK.

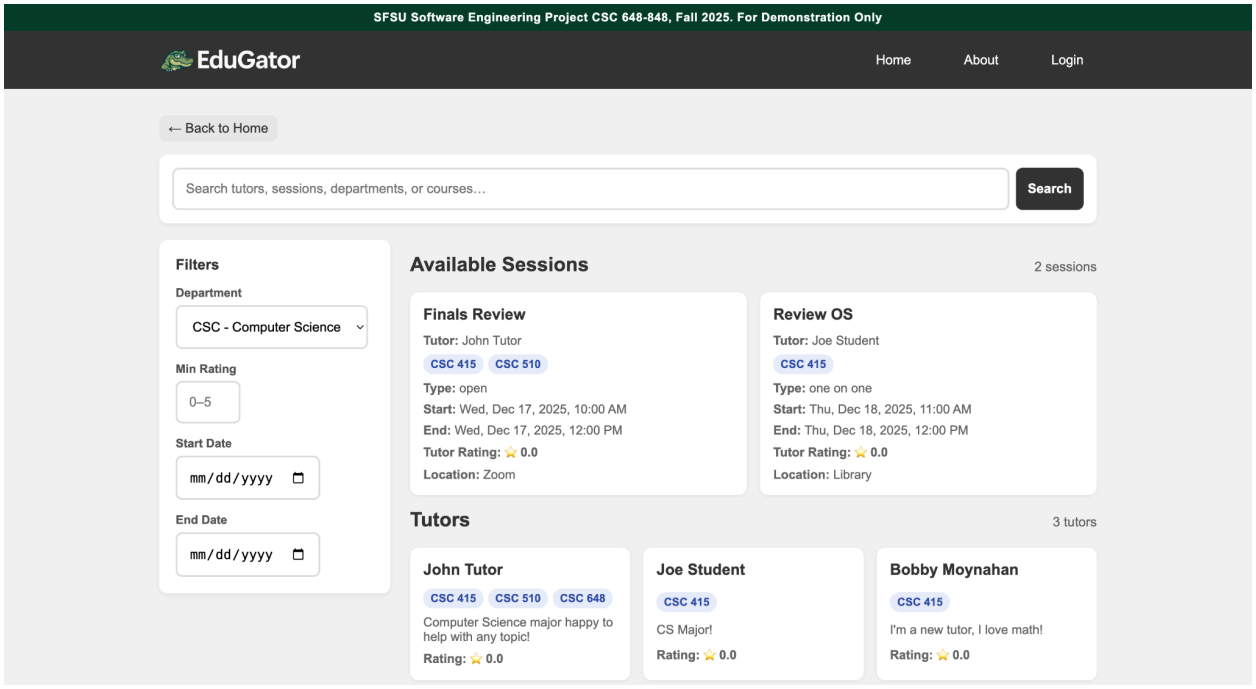
Home Page:



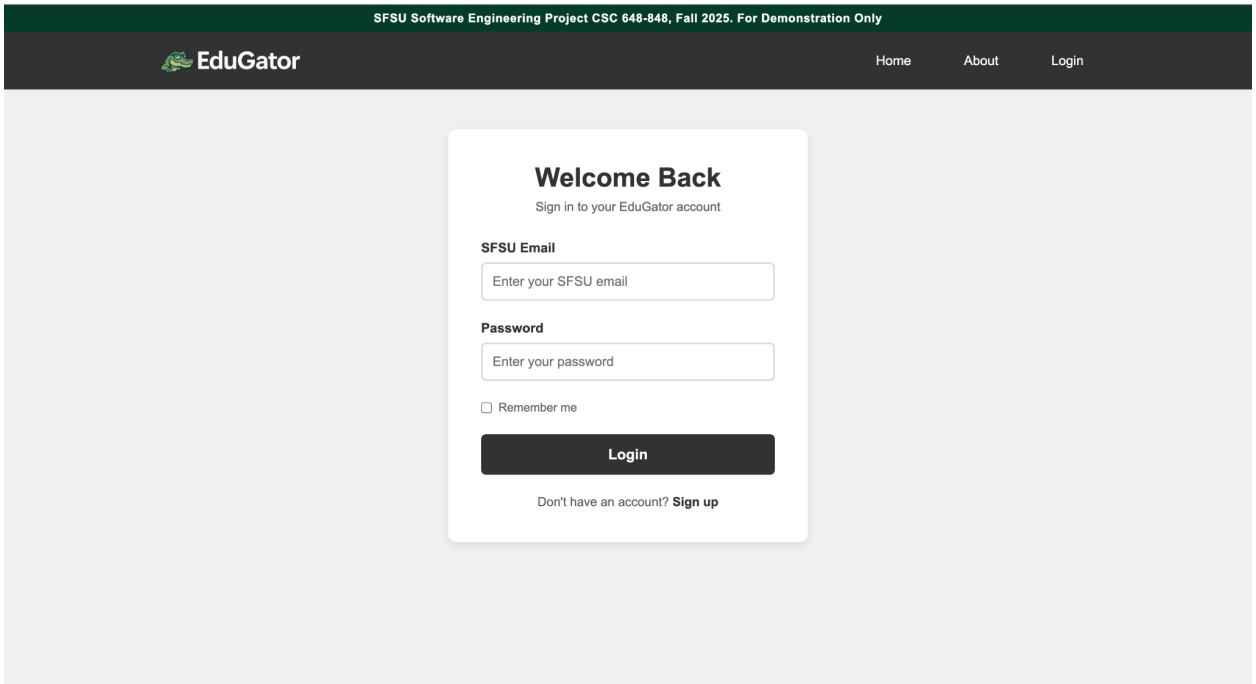
Search:



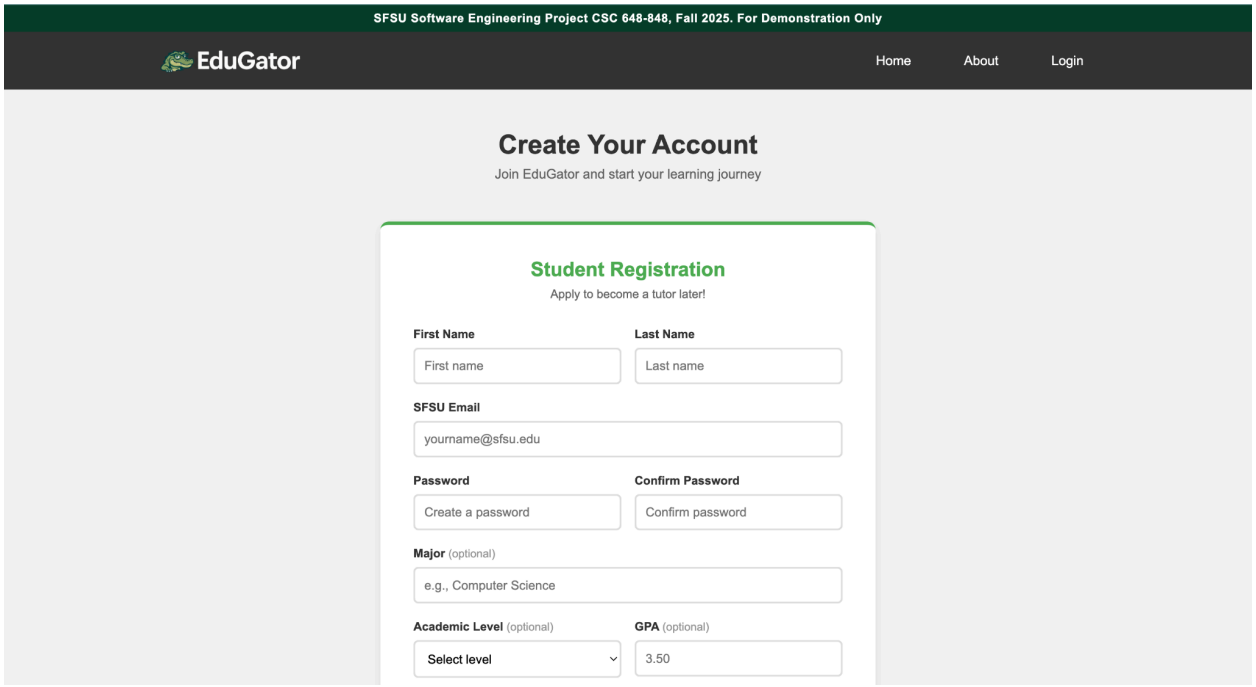
Filter:



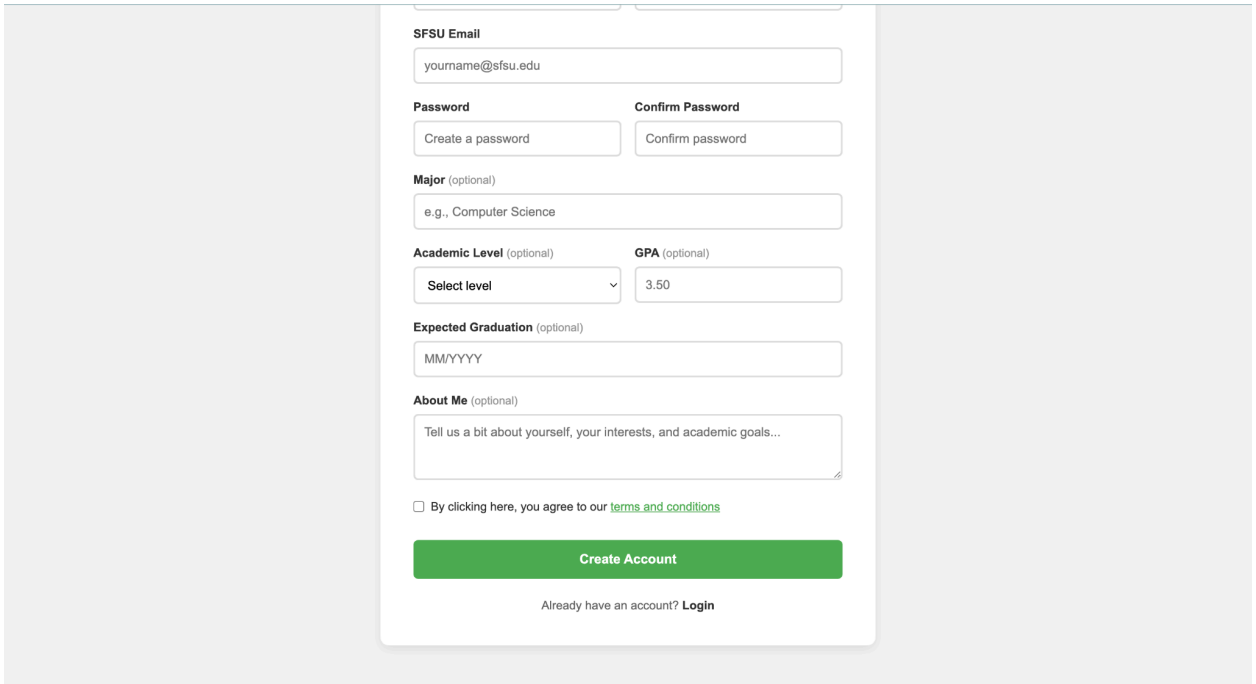
Login:



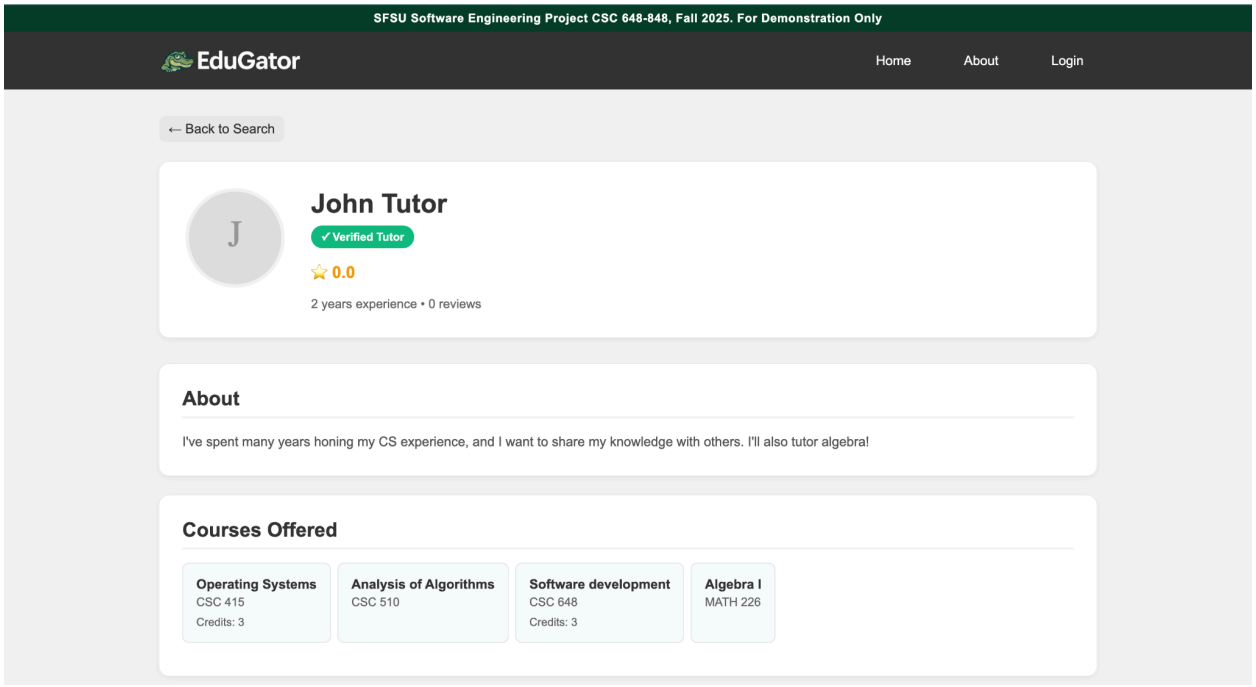
Registration (top half):



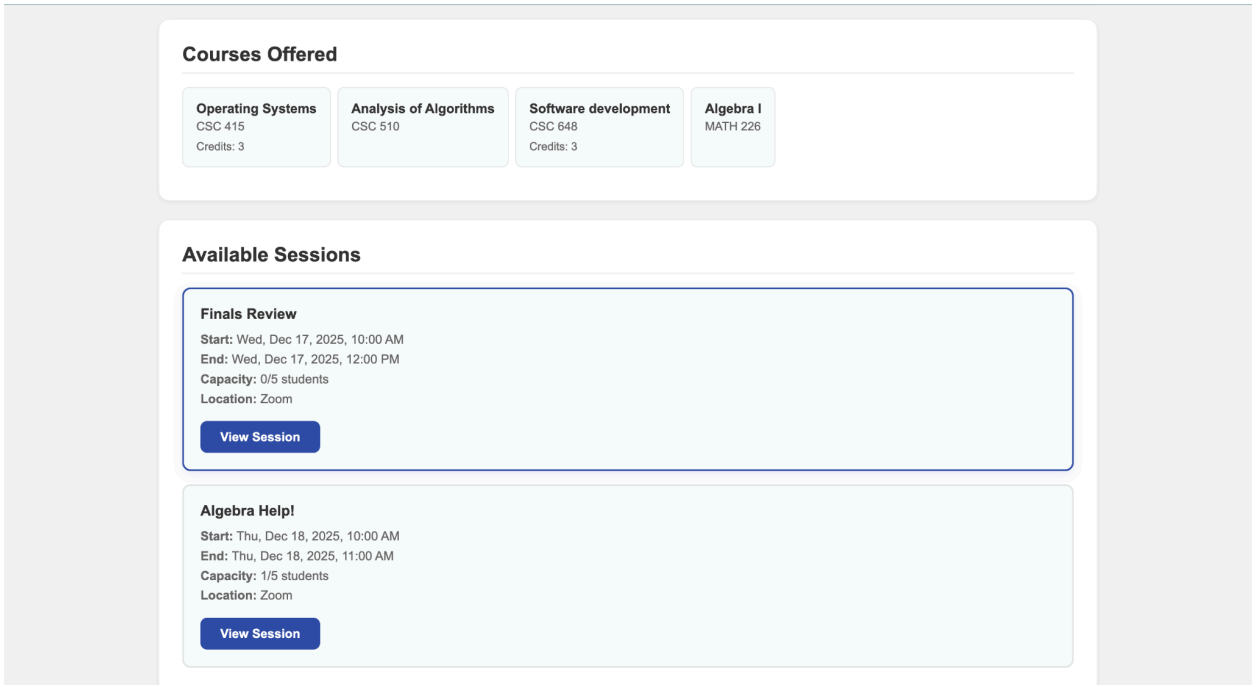
Registration (bottom half):



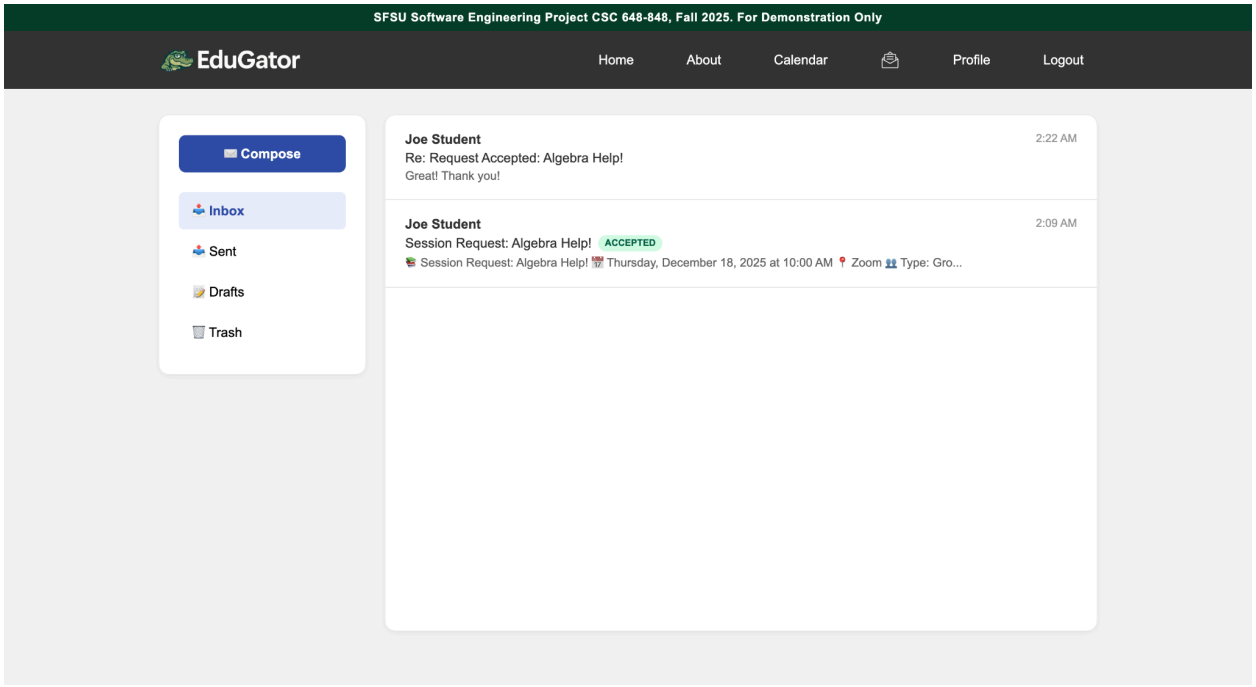
Tutor Profile (top half):



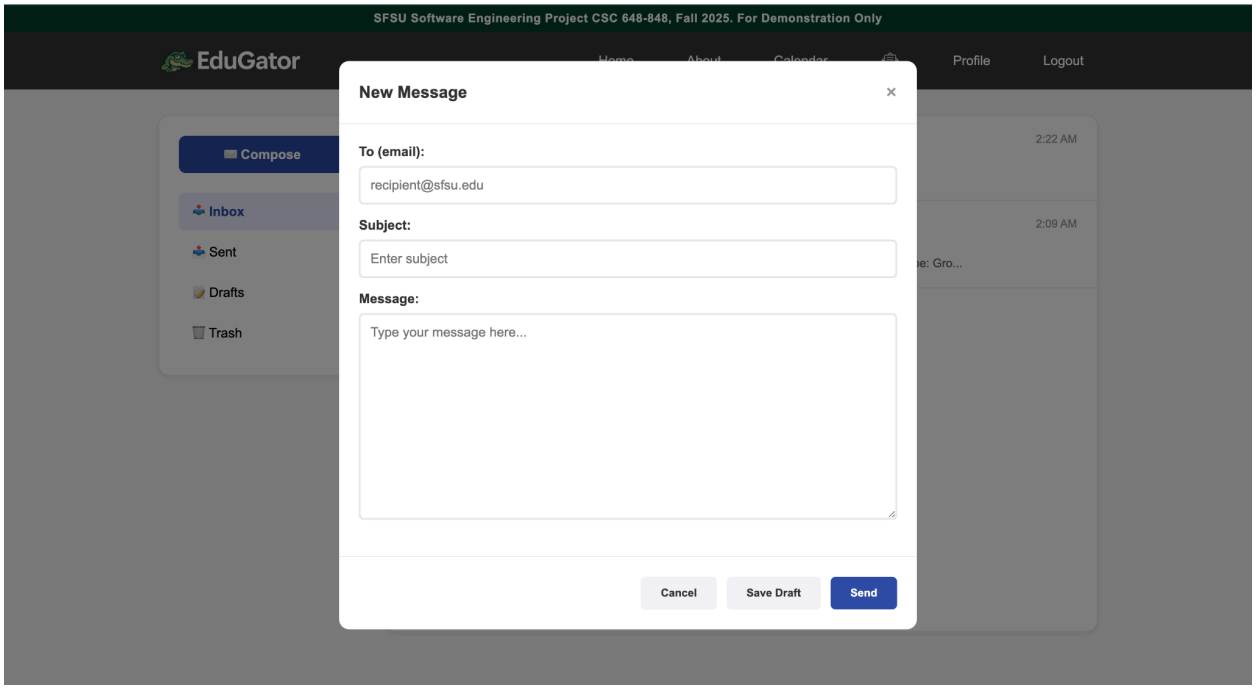
Tutor Profile (bottom half):



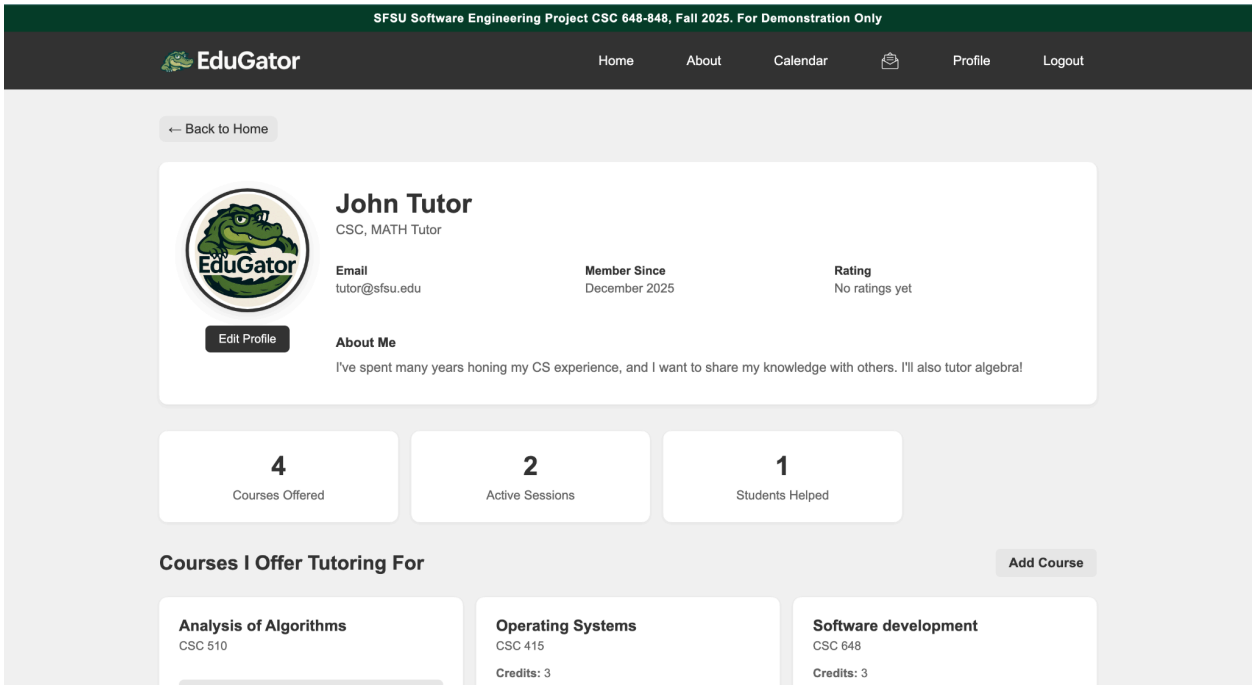
Inbox:



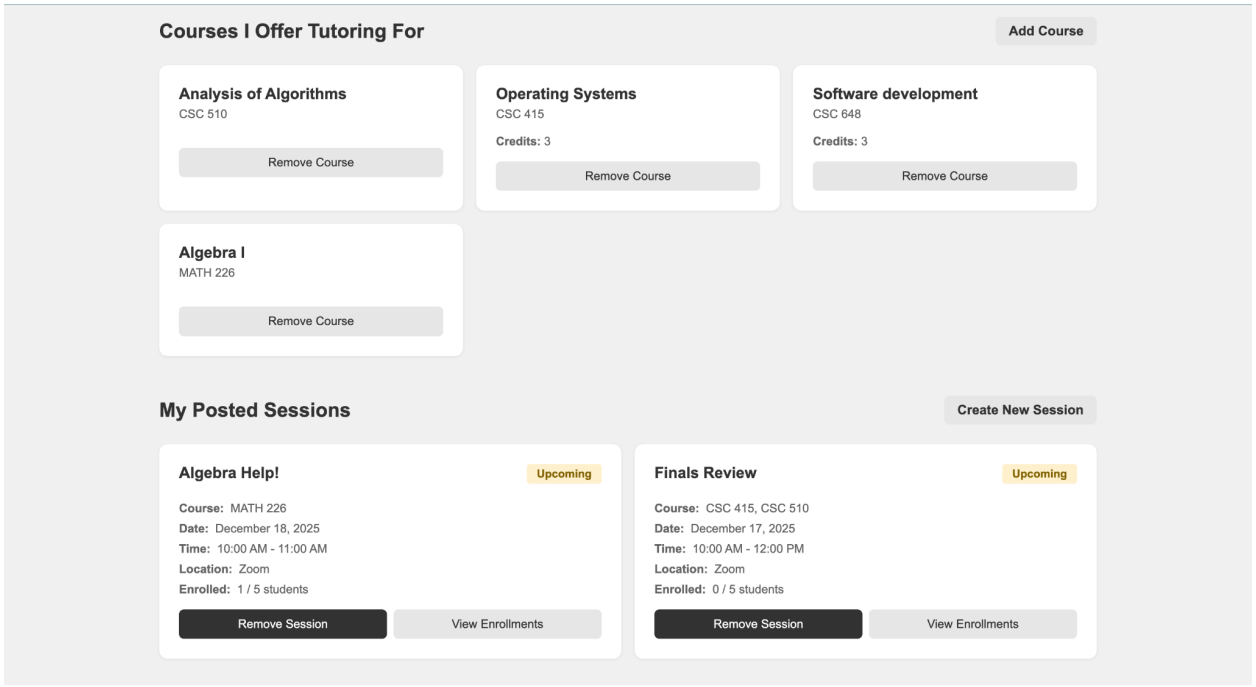
Inbox (compose):



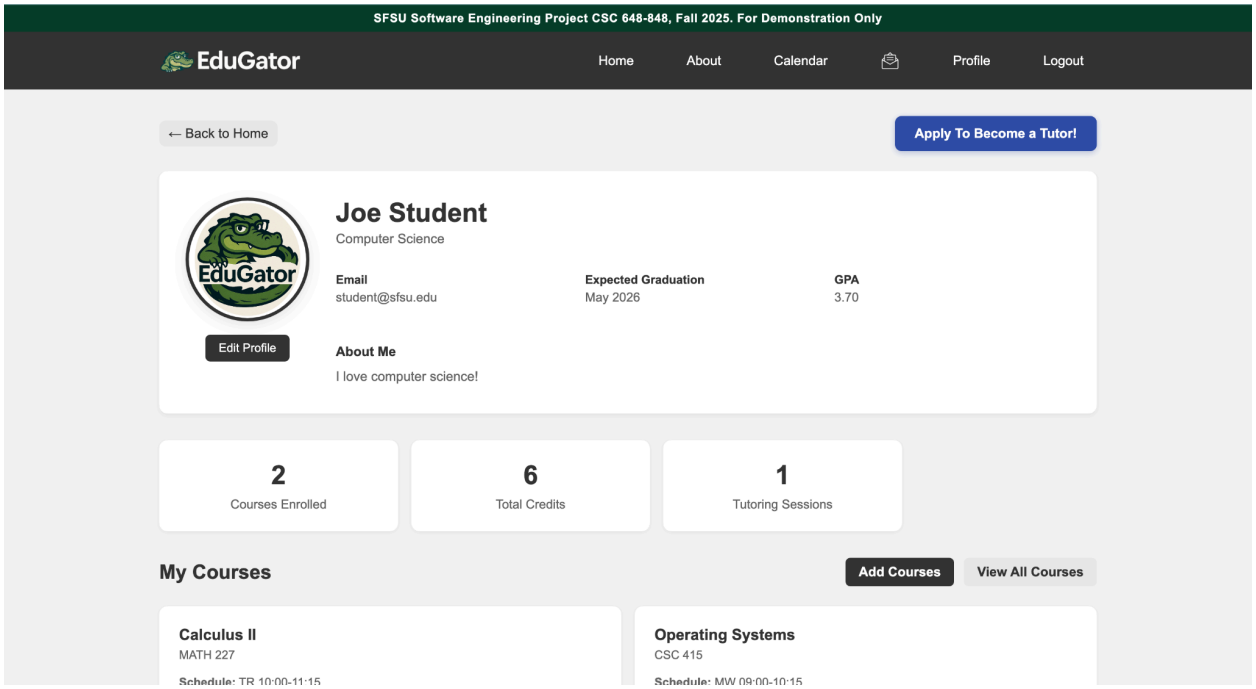
tutorDashboard (top-half):



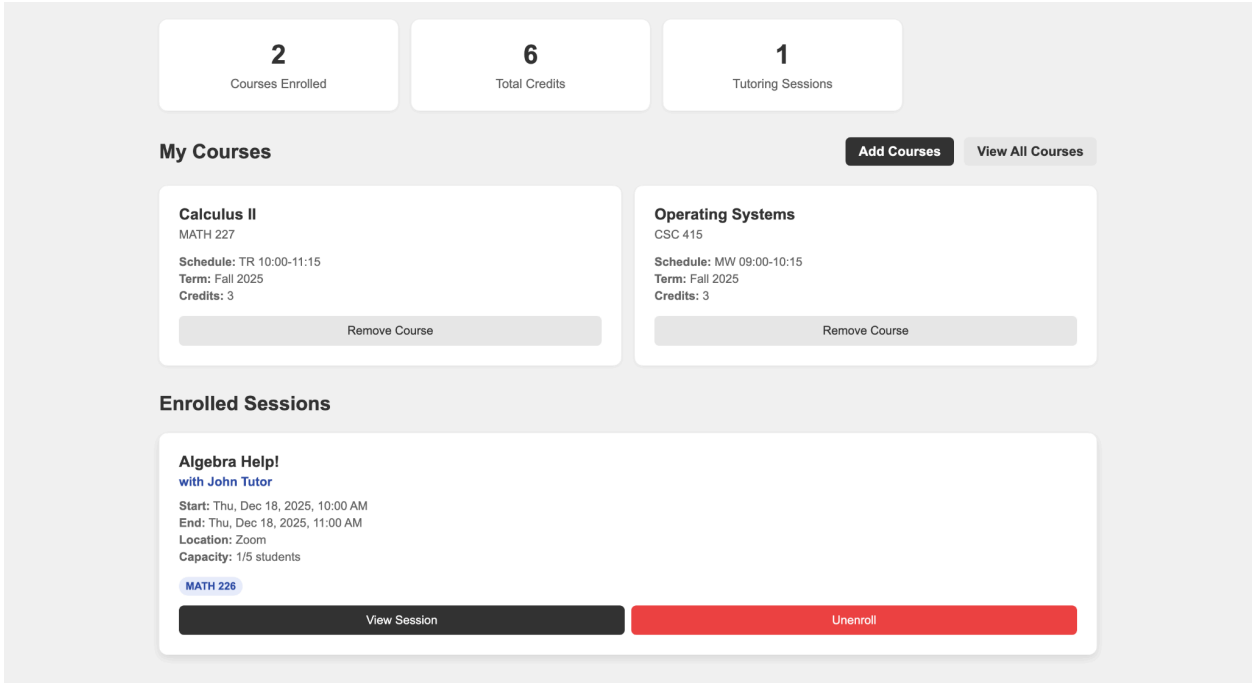
tutorDashboard (bottom-half):



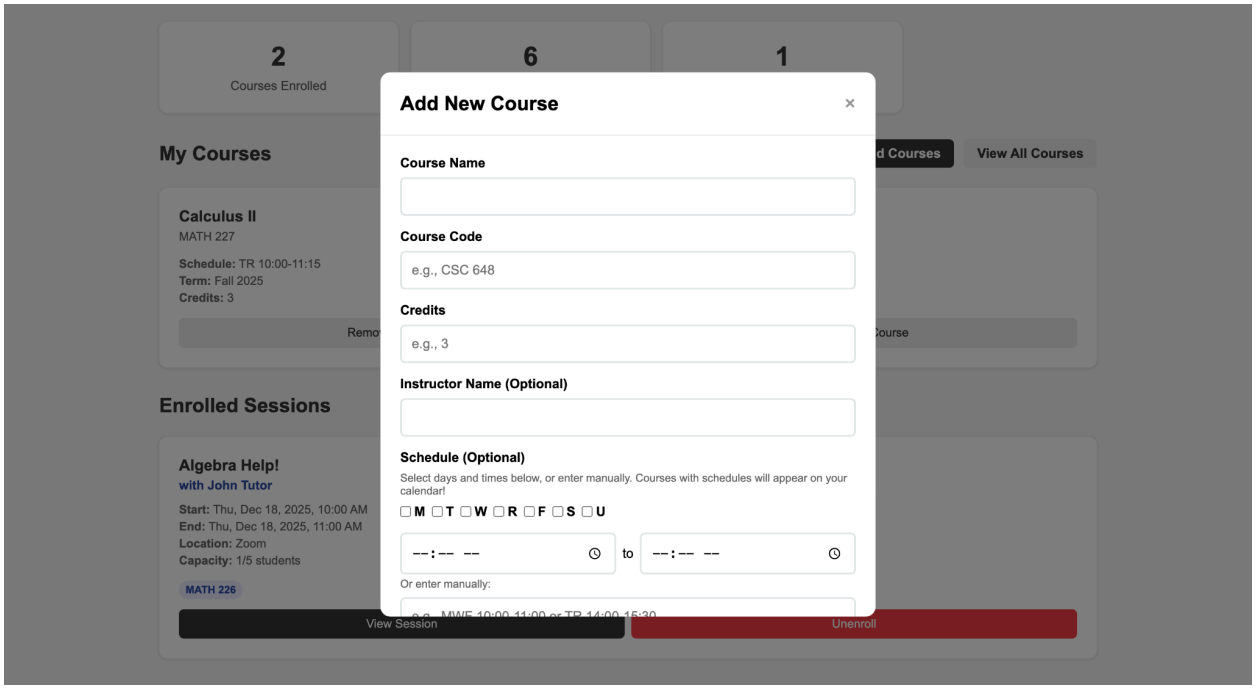
studentDashboard (top-half):



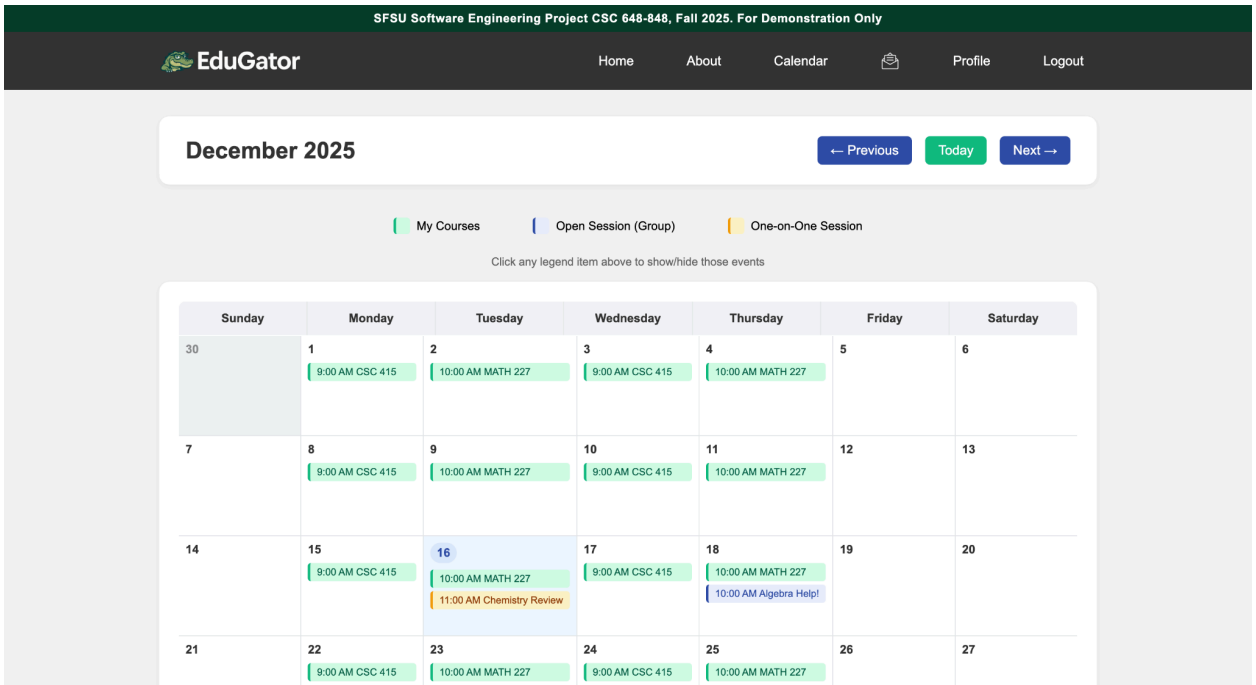
studentDashboard (bottom-half):



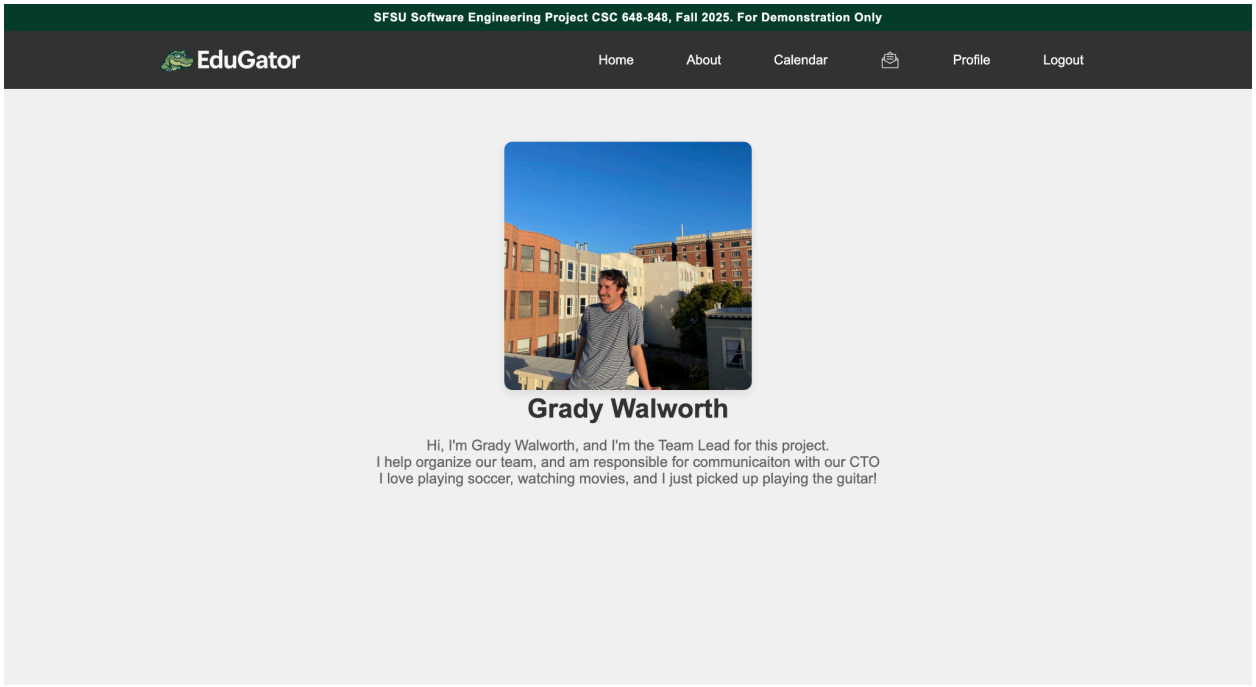
studentDashboard (Add Course):



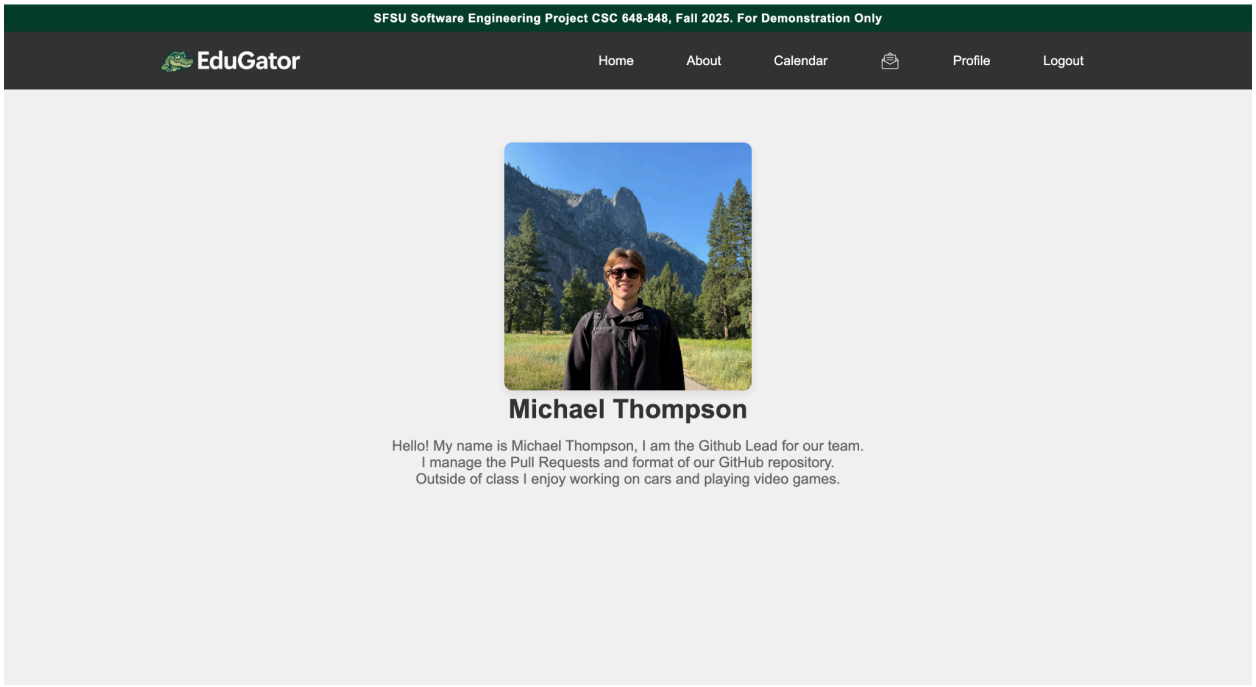
calendar:



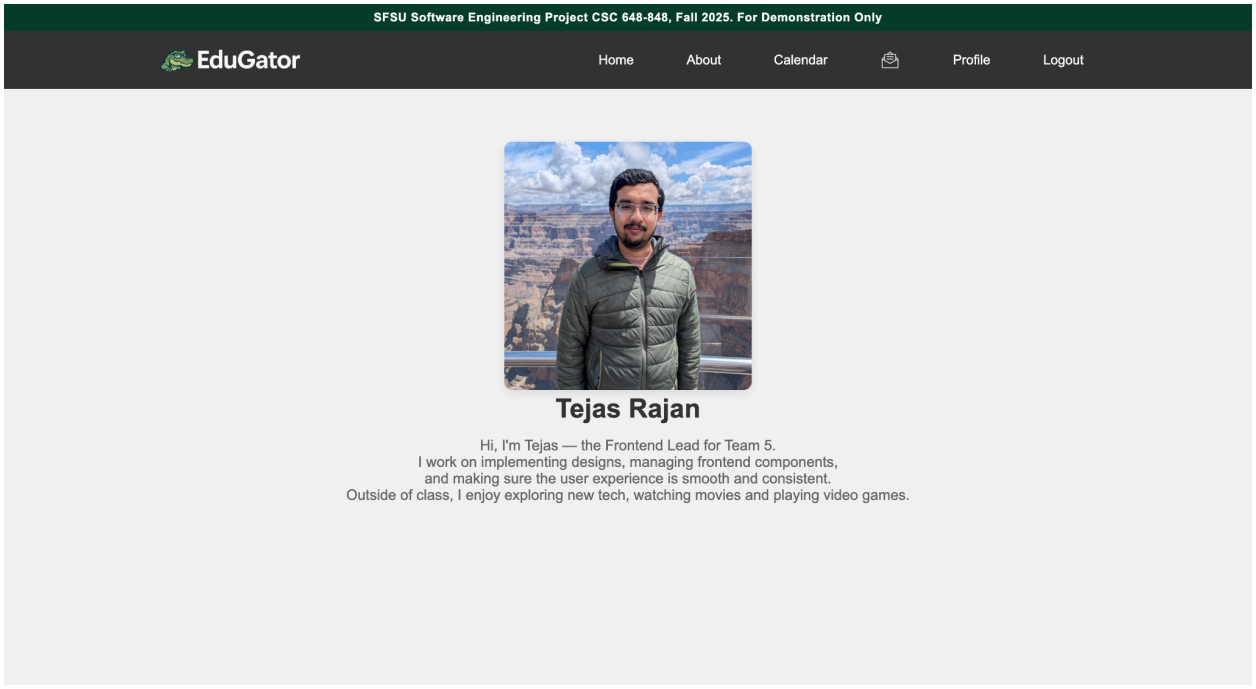
About (Grady):



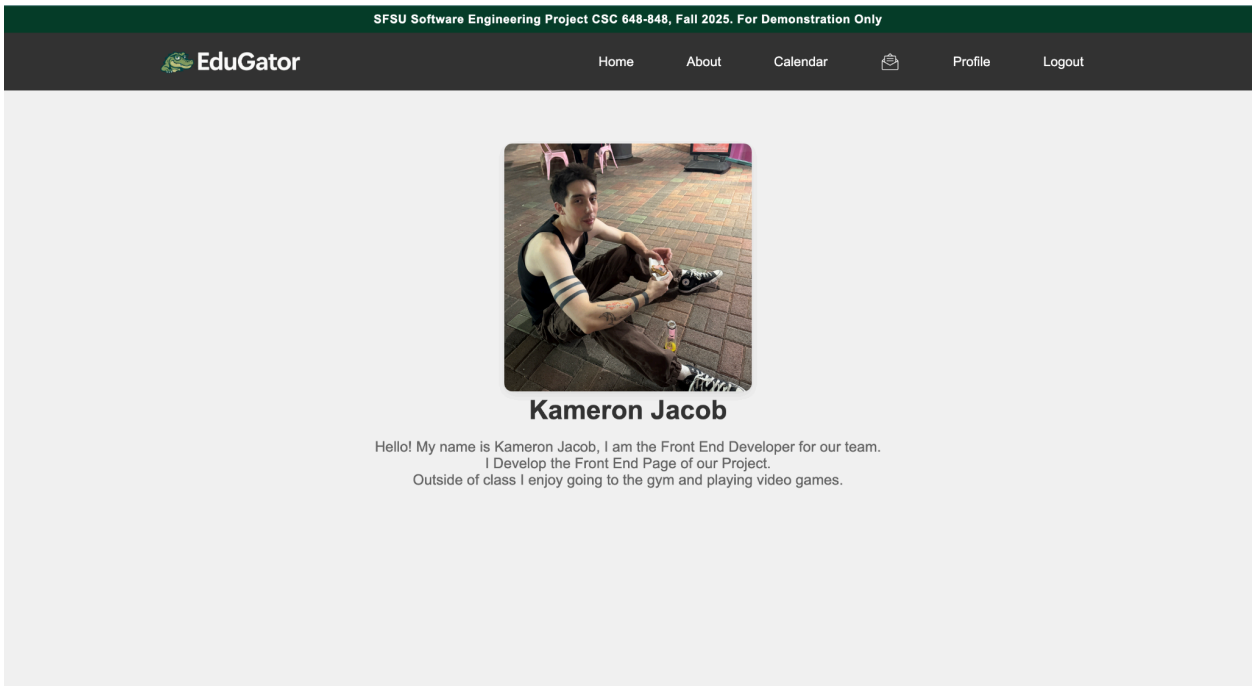
About (Michael):



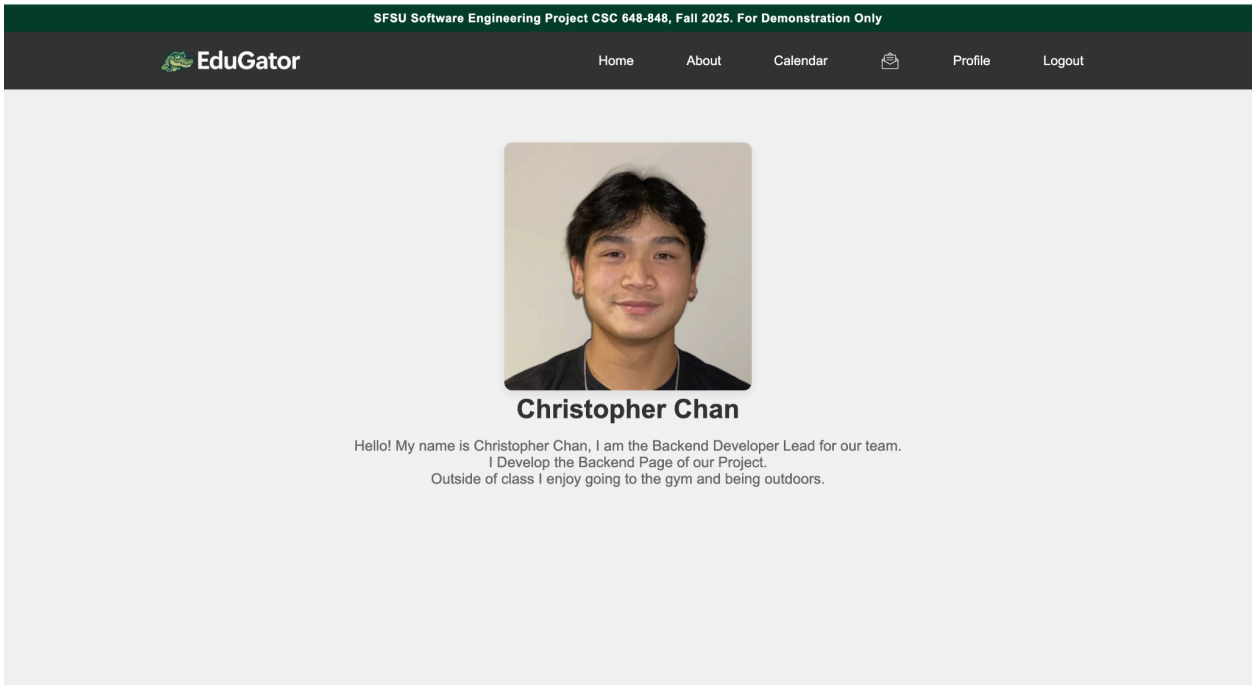
About (Tejas):



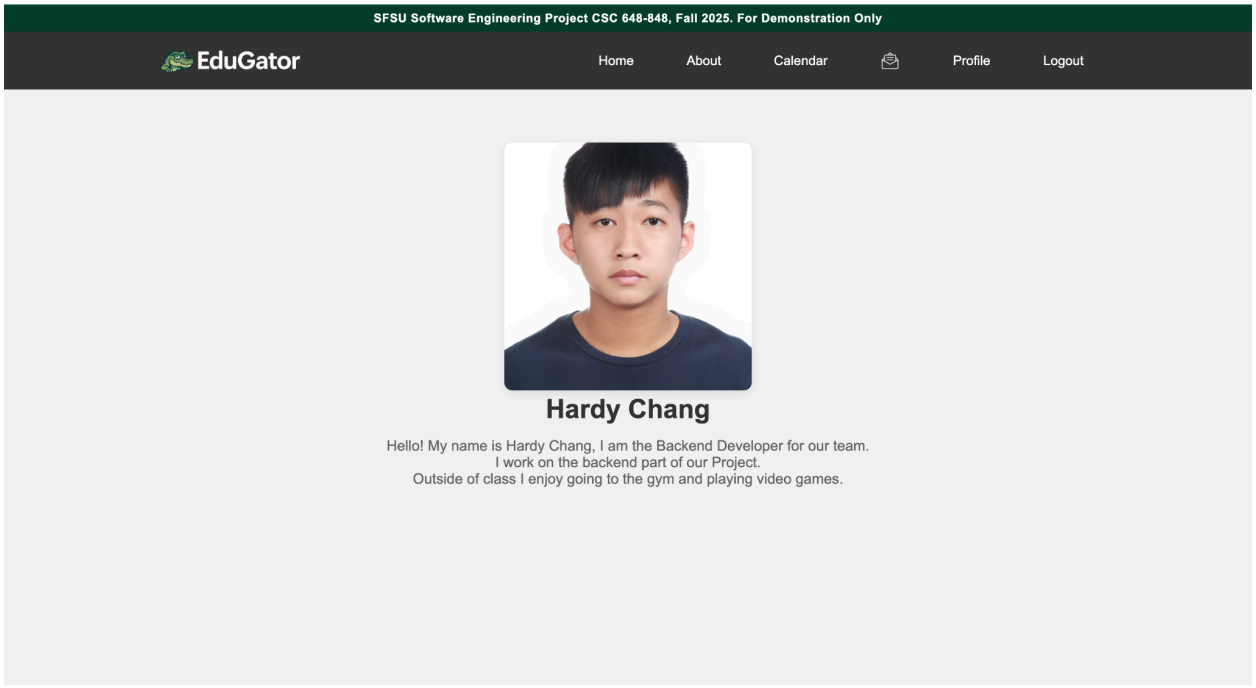
About (Kameron):



About (Chris):

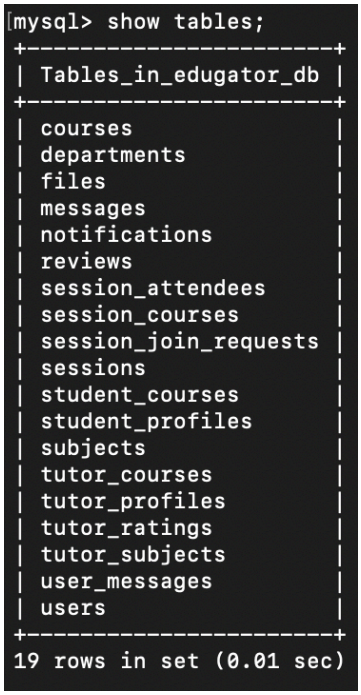


About (Hardy):



4) Database Organization:

(Screen shots of key DB tables (2-3 pages) Show snapshots of all important DB tables (e.g. users, items, category, messages etc.). Make it easy to read and review. OK to show screen shots of Workbench. Ensure screen shots are easy to read.



```
mysql> show tables;
+------------------------+
| Tables_in_edugator_db |
+------------------------+
| courses                |
| departments            |
| files                  |
| messages               |
| notifications          |
| reviews                |
| session_attendees      |
| session_courses        |
| session_join_requests  |
| sessions               |
| student_courses        |
| student_profiles       |
| subjects               |
| tutor_courses          |
| tutor_profiles         |
| tutor_ratings          |
| tutor_subjects         |
| user_messages          |
| users                  |
+------------------------+
19 rows in set (0.01 sec)
```

```
mysql> describe courses;
+---------------+------------------+------+-----+---------+----------------+
| Field         | Type             | Null | Key | Default | Extra          |
+---------------+------------------+------+-----+---------+----------------+
| course_id     | bigint           | NO   | PRI | NULL    | auto_increment |
| department_id | bigint           | NO   | MUL | NULL    |                |
| course_number | varchar(16)      | NO   |     | NULL    |                |
| course_title  | varchar(150)     | NO   |     | NULL    |                |
| description   | text             | YES  |     | NULL    |                |
| credits       | tinyint unsigned | YES  |     | NULL    |                |
+---------------+------------------+------+-----+---------+----------------+
6 rows in set (0.02 sec)

mysql> describe departments;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| department_id | bigint       | NO   | PRI | NULL    | auto_increment |
| code          | varchar(16)  | NO   | UNI | NULL    |                |
| name          | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> describe files;
+---------------+-----------+------+-----+-------------------+-------------------+
| Field         | Type      | Null | Key | Default           | Extra             |
+---------------+-----------+------+-----+-------------------+-------------------+
| file_id       | bigint    | NO   | PRI | NULL              | auto_increment    |
| owner_user_id | bigint    | NO   | MUL | NULL              |                   |
| url           | text      | NO   |     | NULL              |                   |
| created_at    | timestamp | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+---------------+-----------+------+-----+-------------------+-------------------+
4 rows in set (0.00 sec)

mysql> describe messages;
+-----------------+-------------------------------------+------+-----+-------------------+-------------------+
| Field           | Type                                | Null | Key | Default           | Extra             |
+-----------------+-------------------------------------+------+-----+-------------------+-------------------+
| message_id      | bigint                              | NO   | PRI | NULL              | auto_increment    |
| sender_id       | bigint                              | NO   | MUL | NULL              |                   |
| receiver_id     | bigint                              | NO   | MUL | NULL              |                   |
| subject         | varchar(255)                        | YES  |     | NULL              |                   |
| message_type    | enum('normal','session_join_request') | NO   |     | normal            |                   |
| message_content | text                                | NO   |     | NULL              |                   |
| time_sent       | timestamp                           | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------------+-------------------------------------+------+-----+-------------------+-------------------+
7 rows in set (0.00 sec)
```

```
mysql> describe notifications;
+-----------------+--------------+------+-----+-------------------+-------------------+
| Field           | Type         | Null | Key | Default           | Extra             |
+-----------------+--------------+------+-----+-------------------+-------------------+
| notification_id | bigint       | NO   | PRI | NULL              | auto_increment    |
| user_id         | bigint       | NO   | MUL | NULL              |                   |
| type            | varchar(50)  | NO   |     | NULL              |                   |
| title           | varchar(150) | YES  |     | NULL              |                   |
| body            | text         | YES  |     | NULL              |                   |
| data            | json         | YES  |     | NULL              |                   |
| is_read         | tinyint(1)   | NO   |     | 0                 |                   |
| created_at      | timestamp    | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------------+--------------+------+-----+-------------------+-------------------+
8 rows in set (0.03 sec)

mysql> describe reviews;
+------------+-----------+------+-----+-------------------+-------------------+
| Field      | Type      | Null | Key | Default           | Extra             |
+------------+-----------+------+-----+-------------------+-------------------+
| review_id  | bigint    | NO   | PRI | NULL              | auto_increment    |
| session_id | bigint    | NO   | MUL | NULL              |                   |
| student_id | bigint    | NO   | MUL | NULL              |                   |
| tutor_id   | bigint    | NO   | MUL | NULL              |                   |
| rating     | tinyint   | NO   |     | NULL              |                   |
| comment    | text      | YES  |     | NULL              |                   |
| created_at | timestamp | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+-----------+------+-----+-------------------+-------------------+
7 rows in set (0.00 sec)

mysql> describe session_attendees;
+-------------+-----------+------+-----+-------------------+-------------------+
| Field       | Type      | Null | Key | Default           | Extra             |
+-------------+-----------+------+-----+-------------------+-------------------+
| session_id  | bigint    | NO   | PRI | NULL              |                   |
| user_id     | bigint    | NO   | PRI | NULL              |                   |
| enrolled_at | timestamp | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------+-----------+------+-----+-------------------+-------------------+
3 rows in set (0.00 sec)

mysql> describe session_courses;
+------------+--------+------+-----+---------+-------+
| Field      | Type   | Null | Key | Default | Extra |
+------------+--------+------+-----+---------+-------+
| session_id | bigint | NO   | PRI | NULL    |       |
| course_id  | bigint | NO   | PRI | NULL    |       |
+------------+--------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> describe session_join_requests;
+------------------+-----------------------------------+------+-----+-------------------+-------------------+
| Field            | Type                              | Null | Key | Default           | Extra             |
+------------------+-----------------------------------+------+-----+-------------------+-------------------+
| request_id       | bigint                            | NO   | PRI | NULL              | auto_increment    |
| session_id       | bigint                            | NO   | MUL | NULL              |                   |
| requester_user_id | bigint                           | NO   | MUL | NULL              |                   |
| tutor_user_id    | bigint                            | NO   | MUL | NULL              |                   |
| message_id       | bigint                            | NO   | MUL | NULL              |                   |
| status           | enum('pending','accepted','denied') | NO   |     | pending           |                   |
| created_at       | timestamp                         | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| responded_at     | timestamp                         | YES  |     | NULL              |                   |
+------------------+-----------------------------------+------+-----+-------------------+-------------------+
8 rows in set (0.00 sec)

mysql> describe sessions;
+-----------------+-----------------------------------+------+-----+-------------------+-------------------+
| Field           | Type                              | Null | Key | Default           | Extra             |
+-----------------+-----------------------------------+------+-----+-------------------+-------------------+
| session_id      | bigint                            | NO   | PRI | NULL              | auto_increment    |
| tutor_id        | bigint                            | NO   | MUL | NULL              |                   |
| title           | varchar(150)                      | NO   |     | NULL              |                   |
| start_time      | datetime                          | NO   |     | NULL              |                   |
| end_time        | datetime                          | NO   |     | NULL              |                   |
| session_type    | enum('open','one_on_one')         | NO   |     | NULL              |                   |
| capacity        | int                               | NO   |     | 1                 |                   |
| location_details | varchar(255)                     | YES  |     | NULL              |                   |
| status          | enum('scheduled','active','over') | NO   |     | scheduled         |                   |
| created_at      | timestamp                         | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| updated_at      | timestamp                         | YES  |     | NULL              |                   |
+-----------------+-----------------------------------+------+-----+-------------------+-------------------+
11 rows in set (0.00 sec)

mysql> describe student_courses;
+----------------+-----------------------------------+------+-----+-------------------+-------------------+
| Field          | Type                              | Null | Key | Default           | Extra             |
+----------------+-----------------------------------+------+-----+-------------------+-------------------+
| student_user_id | bigint                           | NO   | PRI | NULL              |                   |
| course_id      | bigint                            | NO   | PRI | NULL              |                   |
| status         | enum('current','planned','completed') | NO   |     | current           |                   |
| term_label     | varchar(32)                       | YES  |     | NULL              |                   |
| instructor_name | varchar(100)                     | YES  |     | NULL              |                   |
| schedule_text  | varchar(120)                      | YES  |     | NULL              |                   |
| added_at       | timestamp                         | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+----------------+-----------------------------------+------+-----+-------------------+-------------------+
7 rows in set (0.01 sec)

mysql> describe student_profiles;
+--------------------------+-----------------------------------------------------------------------+------+-----+---------+-------+
| Field                    | Type                                                                  | Null | Key | Default | Extra |
+--------------------------+-----------------------------------------------------------------------+------+-----+---------+-------+
| user_id                  | bigint                                                                | NO   | PRI | NULL    |       |
| student_number           | varchar(32)                                                           | YES  |     | NULL    |       |
| major                    | varchar(100)                                                          | YES  |     | NULL    |       |
| academic_level           | enum('freshman','sophomore','junior','senior','graduate','other')     | YES  |     | NULL    |       |
| gpa                      | decimal(3,2)                                                          | YES  |     | NULL    |       |
| expected_graduation_date | date                                                                  | YES  |     | NULL    |       |
| bio                      | text                                                                  | YES  |     | NULL    |       |
+--------------------------+-----------------------------------------------------------------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
mysql> describe subjects;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| subject_id  | bigint       | NO   | PRI | NULL    | auto_increment |
| name        | varchar(100) | NO   | UNI | NULL    |                |
| description | text         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)

mysql> describe tutor_courses;
+---------------+--------+------+-----+---------+-------+
| Field         | Type   | Null | Key | Default | Extra |
+---------------+--------+------+-----+---------+-------+
| tutor_user_id | bigint | NO   | PRI | NULL    |       |
| course_id     | bigint | NO   | PRI | NULL    |       |
+---------------+--------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> describe tutor_profiles;
+---------------------+-----------------------------------------+------+-----+---------+-------+
| Field               | Type                                    | Null | Key | Default | Extra |
+---------------------+-----------------------------------------+------+-----+---------+-------+
| user_id             | bigint                                  | NO   | PRI | NULL    |       |
| years_experience    | int                                     | NO   |     | 0       |       |
| verification_status | enum('pending','verified','rejected')   | NO   |     | pending |       |
| hourly_rate         | decimal(8,2)                            | NO   |     | NULL    |       |
| description         | varchar(200)                            | NO   |     |         |       |
| bio                 | text                                    | YES  |     | NULL    |       |
| resume_file_id      | bigint                                  | YES  | MUL | NULL    |       |
+---------------------+-----------------------------------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> describe tutor_ratings;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| tutor_user_id | bigint       | NO   |     | NULL    |       |
| rating_avg    | decimal(7,4) | YES  |     | NULL    |       |
| rating_count  | bigint       | NO   |     | 0       |       |
+---------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> describe tutor_subjects;
+---------------+--------+------+-----+---------+-------+
| Field         | Type   | Null | Key | Default | Extra |
+---------------+--------+------+-----+---------+-------+
| tutor_user_id | bigint | NO   | PRI | NULL    |       |
| subject_id    | bigint | NO   | PRI | NULL    |       |
+---------------+--------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> describe user_messages;
+------------+-----------------------------------+------+-----+-------------------+-------------------+
| Field      | Type                              | Null | Key | Default           | Extra             |
+------------+-----------------------------------+------+-----+-------------------+-------------------+
| user_id    | bigint                            | NO   | PRI | NULL              |                   |
| message_id | bigint                            | NO   | PRI | NULL              |                   |
| folder     | enum('inbox','sent','drafts','trash') | NO   |     | NULL              |                   |
| is_read    | tinyint(1)                        | NO   |     | 0                 |                   |
| added_at   | timestamp                         | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+-----------------------------------+------+-----+-------------------+-------------------+
5 rows in set (0.01 sec)

mysql> describe users;
+-----------------+---------------------------------+------+-----+-------------------+-------------------+
| Field           | Type                            | Null | Key | Default           | Extra             |
+-----------------+---------------------------------+------+-----+-------------------+-------------------+
| user_id         | bigint                          | NO   | PRI | NULL              | auto_increment    |
| email           | varchar(255)                    | NO   | UNI | NULL              |                   |
| password_hash   | varchar(255)                    | NO   |     | NULL              |                   |
| first_name      | varchar(100)                    | NO   |     | NULL              |                   |
| last_name       | varchar(100)                    | NO   |     | NULL              |                   |
| role            | enum('Student','Tutor','Admin') | NO   |     | NULL              |                   |
| profile_file_id | bigint                          | YES  | MUL | NULL              |                   |
| created_at      | timestamp                       | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------------+---------------------------------+------+-----+-------------------+-------------------+
8 rows in set (0.00 sec)
```

5) GitHub organization:

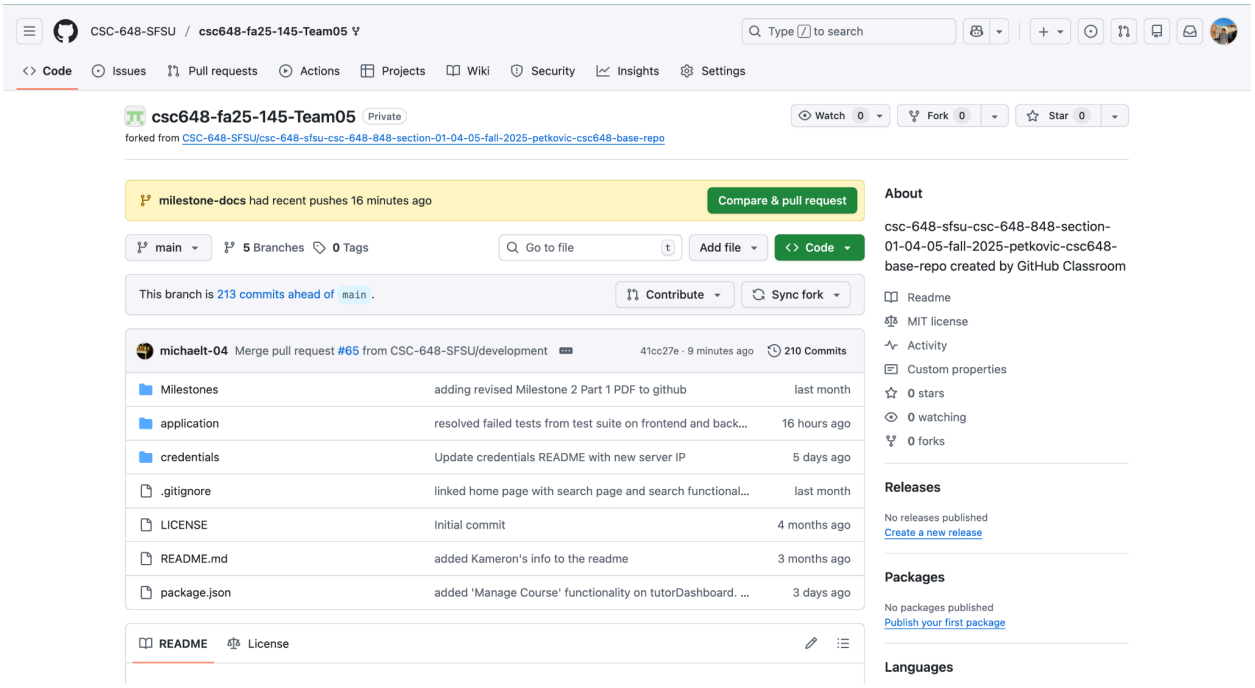
a) list main branches in your team GitHub;

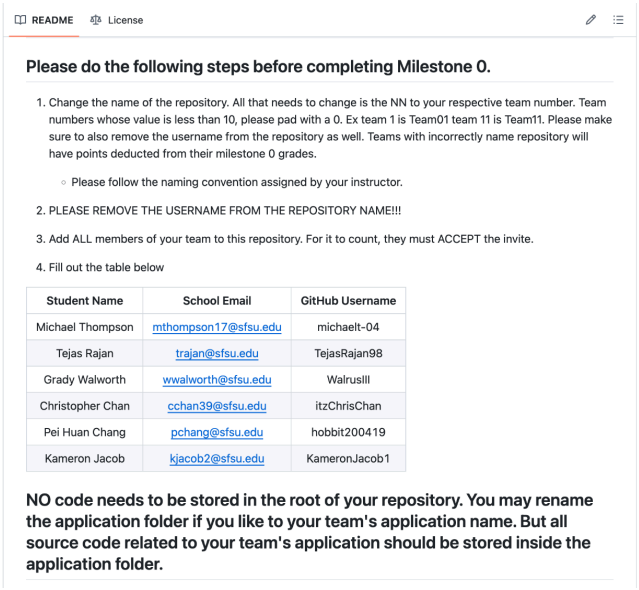
Main and Development were our primary branches, then additional branches were created to work on specific functions/pages as needed per person.

b) clarify who had access to master branch

Michael, our Github Master.

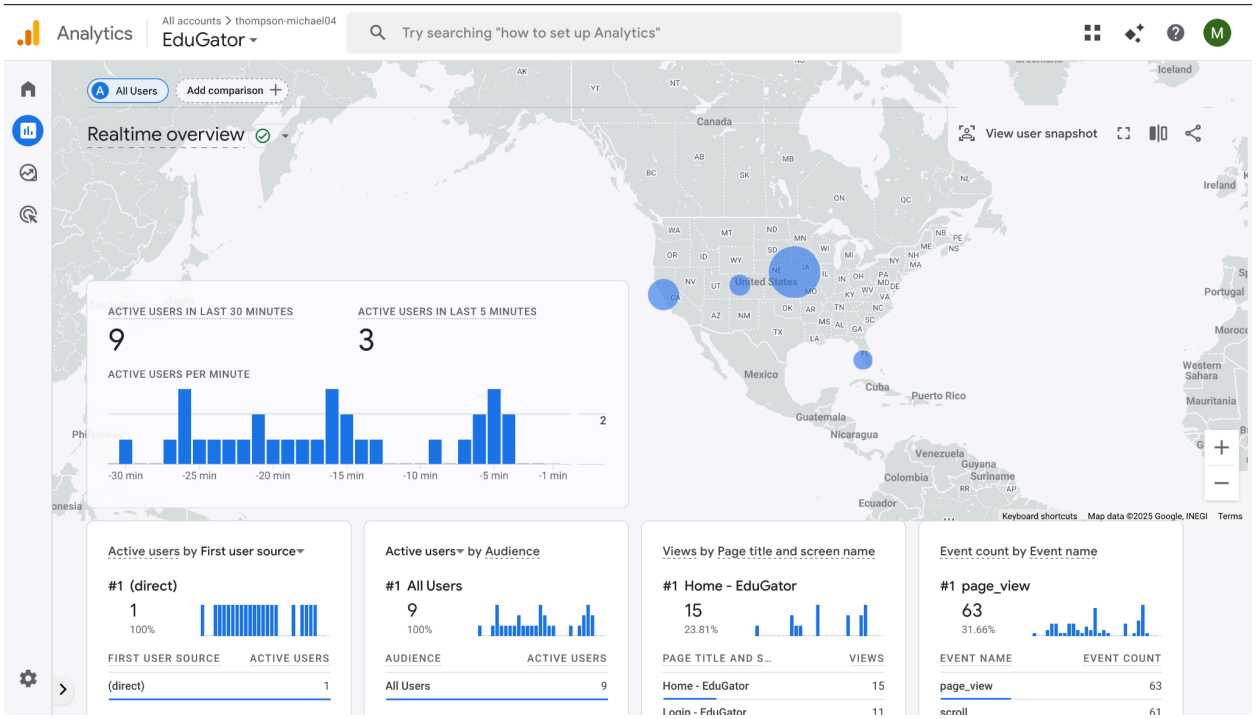
c) provide screen shot of your team GitHub home page

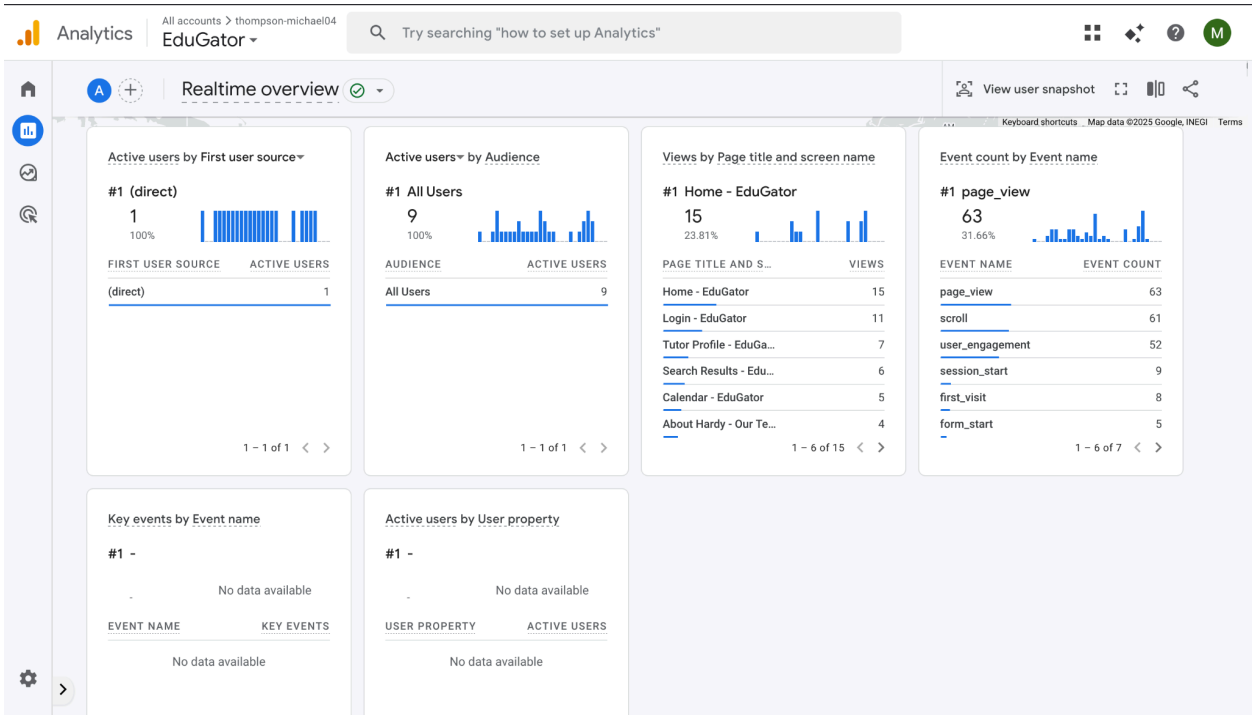




6) Google analytics stats plot for your WWW site

Install Google analytics a few days before the final demo. Show 1–2-page summary report provided by Google Analytics. OK to show stats just for a few days.

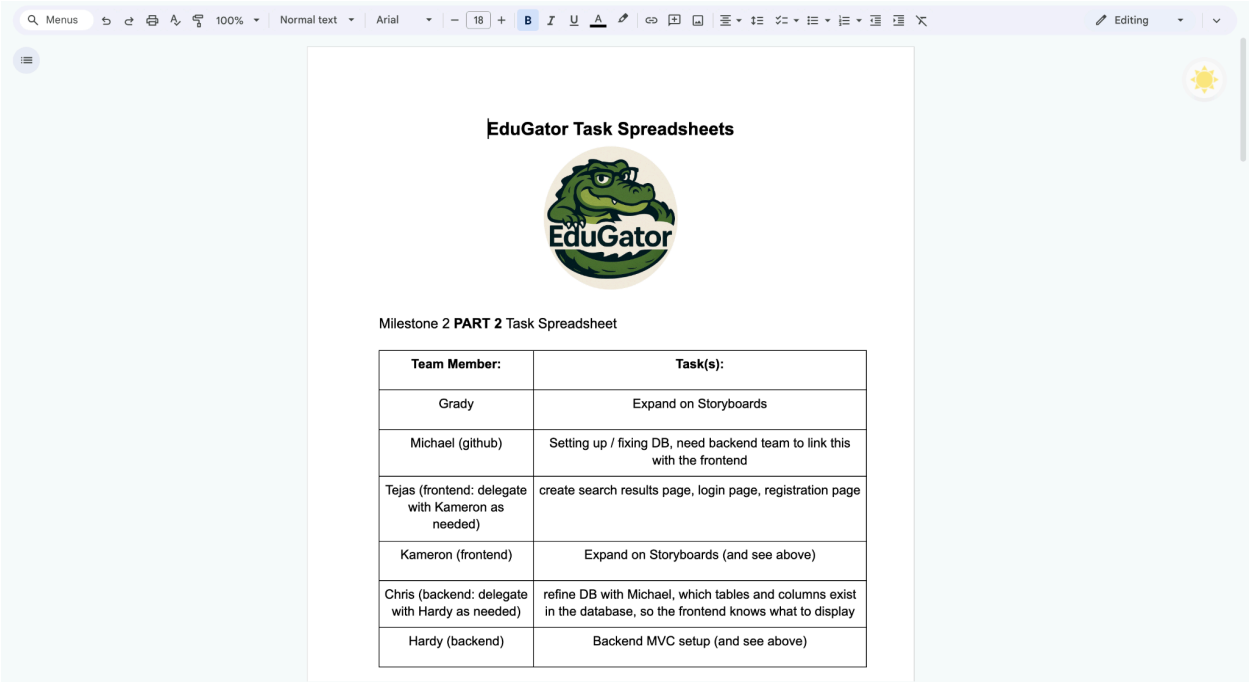




7) Project management:

Say what tools you used and show screen shots of your project management system (like Trello) showing a snapshot(s) of your project management (show 1-2 screens).

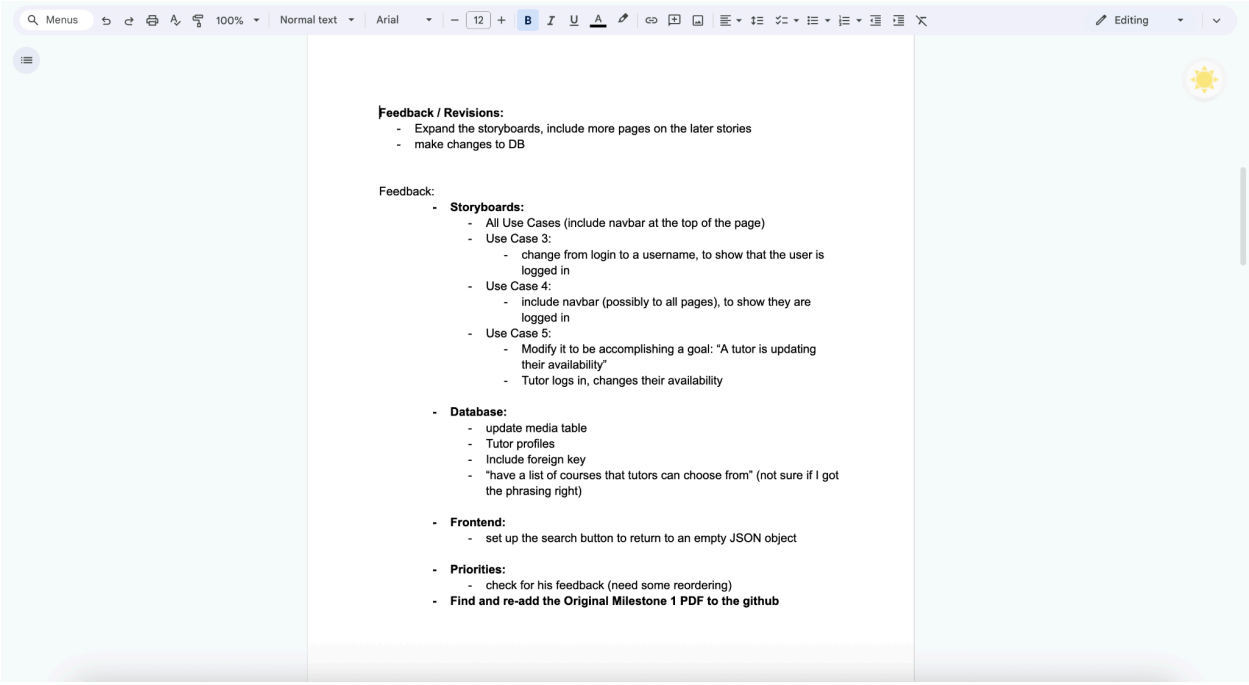
We used task spreadsheets on google docs, where we would list all the tasks needed for a Milestone, and then delegate those out to each member of the team.



EduGator Task Spreadsheets

Milestone 2 **PART 2** Task Spreadsheet

Team Member:	Task(s):
Grady	Expand on Storyboards
Michael (github)	Setting up / fixing DB, need backend team to link this with the frontend
Tejas (frontend: delegate with Kameron as needed)	create search results page, login page, registration page
Kameron (frontend)	Expand on Storyboards (and see above)
Chris (backend: delegate with Hardy as needed)	refine DB with Michael, which tables and columns exist in the database, so the frontend knows what to display
Hardy (backend)	Backend MVC setup (and see above)



Feedback / Revisions:

- Expand the storyboards, include more pages on the later stories
- make changes to DB

Feedback:

- **Storyboards:**
  - All Use Cases (include navbar at the top of the page)
  - Use Case 3:
    - change from login to a username, to show that the user is logged in
  - Use Case 4:
    - include navbar (possibly to all pages), to show they are logged in
  - Use Case 5:
    - Modify it to be accomplishing a goal: "A tutor is updating their availability"
    - Tutor logs in, changes their availability
- **Database:**
  - update media table
  - Tutor profiles
  - Include foreign key
  - "have a list of courses that tutors can choose from" (not sure if I got the phrasing right)
- **Frontend:**
  - set up the search button to return to an empty JSON object
- **Priorities:**
  - check for his feedback (need some reordering)
- **Find and re-add the Original Milestone 1 PDF to the github**

Tasks:

Backend:

- Search functionality
  - Search Bar
    - Search arguments (selection of menu and typed text) must remain persistent after search is executed
    - In upper left corner, below search UI and above listed results, show number of items found (check how Amazon is doing this)
    - If user does not enter any search parameters search results must show ALL items in the database (this is good for tester t know what is the content of the database).
  - Pull down menu with Categories
    - This component of search is exercised using simple SQL filter – see architecture slides from the class (Part II)
    - ANDed with above is one free TEXT ENTRY field (this text is then used in SQL %LIKE search on text string obtained by concatenating item title and description a values from the DB – check architecture slides Part II)
- Frontend:
  -
- design JSON object for interacting between the front and back end

Milestone 2 Part 2 Requirements:

Overall architecture of VP must follow MVC design pattern – check class slides on SW architectures Part I and also our VP SW tutorials, see below.

- UI for the VP test home page is a throw-away test one, and these below are the requirements and design instructions you must follow:
  - Title including Class, semester, year, team number
  - Search UI shall consist of two main components below, modeled after Amazon search:
    - One pull down menu referring to CATEGORIES. Have say 3 entries here like electronics, books, furniture, and default entry. This component of search is exercised using simple SQL filter – see architecture slides from the class (Part II)

8) Team member self-assessment and contributions

Grady Walworth

Michael John Thompson;Tejas Rajan;Christopher Chan;Kameron Allen Jacob;**+2 others**

Team member self-assessment and contributions

a). Their contributions to the team project and teamwork (technical and any other) in no more than half a page – list item format is OK. List also the role (e.g. front-end lead).

Grady Walworth

Role - Team Lead

- Student Dashboard:
    - Add / Remove course, Edit Profile functionality, schedule validation
- Tutor Dashboard:
    - Add / Remove course, Edit Profile functionality
- Calendar - hide/unhide feature
- Registration - contributed to authentication and frontend/backend validation of data
- AI generated Test Suite
- Navbar - contributed to integration of logged in vs logged out navbar, showing calendar and inbox when only logged in
- Session details page
- In general
    - bug fixes, contributions to the writeups for M1 through M5, general UX/UI, team delegation
- Frontend:
    - for UX/UI only: created Session Details page, Calendar page, Inbox page, Tutor Profile page,
        - setup fake demo data for display of our UX/UI for Milestone 3

b). Number of submissions they made to GitHub team Dev. Branch (explain if this number is very low)

28 Commits total.

c). One brief paragraph on the main challenges they encountered in team project

The main challenges I encountered individually were in the backend hookup and testing phases, where I was learning on the fly how much testing is needed to ensure that only valid user data that we want can be entered, and our database, user data, and server are safe from malicious attacks. Both our testing, and the testing suite we created, were great learning experiences for me about data validation and security.

d). One brief paragraph of their own overall experience with GenAI focusing on tasks it was applied to and how useful it was.

GenAI was extremely useful for me in this project. I mostly used Claude, and it was great for coming up with Demo Data for our UX/UI mockups, as well as debugging when I ran into issues, on both the front and back ends. I also was the one who used it to generate our testing suite, which was a good exercise in prompting, as it took a few tries to get it to understand our registration backend flow, and even then, it made an error about testing email length. It was testing to make sure that emails of 255 characters or over would be rejected, but in the suite it only provided an email of 245 characters, leading to an inaccurate result.

e). One brief paragraph on what would do better next time based on what was learned in the class about SE management and processes.

Next time, individually I would try to take better advantage of our long timeline, and start working on pages, even if only just the front end, much earlier in the semester. Even if they weren't due yet, it was obvious we were going to need a registration and login page, and a student and tutor dashboard page. This could've taken the stress off later in the term, when we had to create those pages on a deadline.

f). Anything else you deem important for instructors to know (e.g. why your GitHub count is low)

N/A, I'm proud of the team and our final product.

g). Team leads only: please address c), d) and e) from above speaking from the standpoint of the team lead.

Overall Feedback from Team Lead:

> **c). One brief paragraph on the main challenges they encountered in team project**

The main challenges I encountered as a team lead were in the transitions. When transitioning from the planning phase to the building out the frontend phase, and then from the frontend phase to backend/functionality phase, there were gaps where I should've encouraged myself and the team to chip away at the tasks we needed to accomplish. I could have delegated tasks more specifically and assertively, where instead many tasks fell to those who stepped up, so we were always *right on schedule*, rather than being ahead of schedule.

d). One brief paragraph of their own overall experience with GenAI focusing on tasks it was applied to and how useful it was.

As a team lead, it was a nice change of pace to be encouraged to use GenAI for this course, as it is inevitably integrated into the future of software development, so getting hands-on experience in a controlled and positive environment was extremely valuable. Our team was able to leverage AI together, helping each other with prompting, how to provide relevant context for accurate responses, and how to use it for debugging efficiently.

e). One brief paragraph on what would do better next time based on what was learned in the class about SE management and processes.

Next time, after our initial planning for our platform architecture, I would focus on having the team (myself included) build out the non-functional frontend as soon as possible. This would get a lot of the UX/UI decisions and delays that came up out of the way early on, and would allow us to then ONLY focus on hooking up the backend, and making everything functional, rather than get caught in situations where both UX/UI and backend were being implemented at the same time, because we hadn't thought of a use-case that came up later.

Kameron Allen Jacob

Grady Walworth;Michael John Thompson;Tejas Rajan;Christopher Chan;Pei Huan Chang

Team member self-assessment and contributions

a). Their contributions to the team project and teamwork (technical and any other) in no more than half a page – list item format is OK. List also the role (e.g. front-end lead).

Kameron Jacob

Role - Front End Developer

Student Dashboard:

- Designed and implemented front-end layout and components

Calendar:

- Connected calendar to backend to display student sessions and classes

In general:

- Bug fixes related to UI rendering and component state management

- Contributions to UX/UI consistency across the platform

- Collaborated on design decisions and user flow optimization

- Assisted with front-end testing and debugging aswell as backend

- Also did the storyboard and worked on milestone documents

b). Number of submissions they made to GitHub team Dev. Branch (explain if this number is very low)

4 Commits total. Did large commits of files instead of spread out commits often. Also first time working in a group setting where we couldn't just commit to main branch and it was hard to figure out the github side of things that way.

c). One brief paragraph on the main challenges they encountered in team project

The main challenges I encountered were integrating the front-end components with the backend API, particularly with the calendar functionality. There were a ton of small bugs and errors during testing but they all got worked out. Additionally, maintaining consistent styling and responsive design across different pages while meeting the specific requirements for both student and tutor dashboards presented layout challenges.

d). One brief paragraph of their own overall experience with GenAI focusing on tasks it was applied to and how useful it was.

GenAI was extremely useful for me throughout this project. I primarily used Claude for debugging CSS layout issues, and troubleshooting API integration problems. It was particularly helpful when working on the calendar backend connection, as it helped me understand how to properly structure API calls and handle asynchronous data fetching. Claude also helped with creating responsive design solutions.

e). One brief paragraph on what would do better next time based on what was learned in the class about SE management and processes.

Next time, I would have ensured better consistency across pages and reduced the need for rework when design decisions changed. I would also implement more thorough front end testing earlier in the development cycle, rather than catching UI bugs later during integration. I also would have spread out more of my commits and code so that I wouldnt lose progress when I didnt commit a chunk of code. This was my first time working with a group where we couldn't just push to main so it was difficult at first to figure it out.

f). Anything else you deem important for instructors to know (e.g. why your GitHub count is low)

N/A, I'm proud of the work I contributed and the final tutoring platform we developed as a team.

Pei Huan Chang

Grady Walworth;Michael John Thompson;Tejas Rajan;Christopher Chan;**+1 other**

Team member self-assessment and contributions

a). Their contributions to the team project and teamwork (technical and any other) in no more than half a page – list item format is OK. List also the role (e.g. front-end lead).

Hardy(Pei Huan Chang)

Role - Backend developer

- **Session Page:**
  - Modified the session page to allow students to select courses.
  - Implemented functionality that allows students to request to join a session, with requests sent directly to the tutor's inbox.
- **Course Selection Interface:**
  - Designed the initial course selection interface and user flow for students.
- **Authentication / Login:**
  - Added input validation during login to restrict users to required input formats.
- **In General:**
  - Assisted with bug fixes and feature integration related to backend functionality.
  - Collaborated with team members to ensure backend features aligned with frontend UI behavior.

b). Number of submissions they made to GitHub team Dev. Branch (explain if this number is very low)

9 Commits total.

c). One brief paragraph on the main challenges they encountered in team project

One of the main challenges I encountered in the team project was designing the session interface at the early stage. At first, it was difficult to decide the layout and user flow because the requirements were still changing and we had to consider different user roles.

d). One brief paragraph of their own overall experience with GenAI focusing on tasks it was applied to and how useful it was.

Overall, I used GenAI mainly to review wording and structure for documents and to check code readability. It was helpful for catching small issues and improving clarity, but the final decisions were still made by the team.

e). One brief paragraph on what would do better next time based on what was learned in the class about SE management and processes.

Based on what I learned in this class about software engineering management and processes, I would improve communication and planning in future projects. Setting clearer milestones earlier and having more frequent check-ins would help the team stay aligned. I would also put more emphasis on documenting design decisions and reviewing requirements regularly, which could reduce rework and confusion later in the project.

f). Anything else you deem important for instructors to know (e.g. why your GitHub count is low)

N/A, I'm proud of the work I contributed and the final tutoring platform we developed as a team.

Michael John Thompson

Pei Huan Chang;Kameron Allen Jacob;Grady Walworth;Tejas Rajan;Christopher Chan

Team member self-assessment and contributions

a). Their contributions to the team project and teamwork (technical and any other) in no more than half a page – list item format is OK. List also the role (e.g. front-end lead).

Michael Thompson

Role - GitHub Lead

- Inbox/Messaging
    - Implemented the messaging feature, connected it all to the backend
- Session Requests
    - Implemented how session requests work
- Tutor Applications
    - Implemented how students apply to be tutors
- Database design
    - Designed the structure for the database as well as what the columns for each table was
- AWS EC2 Organization and Setup
    - Managed the EC2 instance, made sure everything on the backend was updated and functioning
- Search Filtering
- Session Functionality
    - Implemented how tutors accept session requests as well as how students unenroll from sessions
- GitHub Management
    - Made sure all merge conflicts were resolved
    - Oversaw the GitHub repository structure as well as making sure all branches were properly used
- UX/UI Design
    - Contributed to how the website should look overall, and what the flow of the website should be

b). Number of submissions they made to GitHub team Dev. Branch (explain if this

number is very low)

79 commits

c). One brief paragraph on the main challenges they encountered in team project

One of the main challenges I encountered during this team project was ensuring that everything was functioning together and nothing was broken when pulled on the server. Prior to this, I had never been the manager of a GitHub repository, so I didn't know the difficulty of making sure everything would work together when merging branches together. Making sure everything was designed properly to work together without friction was something I hadn't dealt with in the past. Overall, this project taught me a lot about software engineering and what it is like to work in a team setting for a long-term project.

d). One brief paragraph of their own overall experience with GenAI focusing on tasks it was applied to and how useful it was.

GenAI was very helpful during this project and helped a lot with the way that the website should be structured. It gave a lot of insight into design choices on the backend that I hadn't considered since I hadn't built something like this before. I think using it as a tool to help with design is the best aspect of something like ChatGPT or Claude. When designing something like the session requests it helped a lot, I hadn't originally planned to create another table for session requests, but it made it much easier to manage the requests and see them live in the backend. GenAI allows you to build a good architecture for a big project like this even when you don't have a ton of experience building something like this, it does make mistakes, but if you understand what you are looking at and what you want to you are able to use it very effectively.

e). One brief paragraph on what would do better next time based on what was learned in the class about SE management and processes.

One thing I would do better next time based on what I learned in the class is think full processes through more before starting to work on them. Having a clear focused goal for how you want something to work saves a lot of time, since you won't have to go back if you don't like the way it works once you have it somewhat done. I really enjoyed the feedback that we received along the way. It was a good check to make sure we were on the right track instead of building a bad foundation from the start and having to work around it the whole time.

f). Anything else you deem important for instructors to know (e.g. why your GitHub count is low)

N/A

Tejas Rajan

Michael John Thompson;Pei Huan Chang;Kameron Allen Jacob;**+2 others**

Team member self-assessment and contributions

a). Their contributions to the team project and teamwork (technical and any other) in no more than half a page – list item format is OK. List also the role (e.g. front-end lead).

Tejas Rajan

Role - Front-end Lead (Edugator – Tutoring Website)

- Led the front-end development for Edugator, ensuring a clean, responsive, and user-friendly interface.
- Built key UX/UI pages including the Home page and Search page, and developed the full search UI, including components, layout, and display of results.
- Created fake demo data to showcase the Search page's UI/UX during development.
- Developed the responsive navbar and contributed to maintaining a uniform front-end design across the site.
- Provided ongoing feedback on design and functionality, helping shape features and improve overall user experience.
- Collaborated closely with teammates, provided code reviews, and supported others with front-end implementation.
- Ensured quality through testing, UI refinements, and responsiveness across devices.
- In general
    - Participated in team documentation efforts, contributing to write-ups for Milestones M1 through M5, general UX/UI.
- Frontend:
    - for UX/UI only: created Home page, Search page
        - setup fake demo data for display of our UX/UI for search

b). Number of submissions they made to GitHub team Dev. Branch (explain if this number is very low)

8 Commits total. The commit count is low because a substantial portion of my contributions centered on UI/UX feedback, design refinement, and milestone documentation, which were essential to the project's progress but did not directly generate individual GitHub commits.

c). One brief paragraph on the main challenges they encountered in team project

One of the main challenges I faced during the project was managing GitHub workflow and resolving merge conflicts. As multiple team members were contributing simultaneously, keeping my local branch in sync with the latest changes and ensuring smooth merges often required extra time and coordination. Handling conflicting files, understanding the source of discrepancies, and preventing repeated conflicts became a learning experience and a consistent challenge throughout development.

d). One brief paragraph of their own overall experience with GenAI focusing on tasks it was applied to and how useful it was.

My experience with GenAI tools such as ChatGPT and Gemini was notably beneficial throughout the project. I primarily used them to review code and was really useful for debugging code. These tools also assisted in generating sample data for the Search page and for the database workbench, which streamlined testing and UI demonstrations. Overall, GenAI provided effective supplementary support, enhancing both development efficiency and workflow quality.

e). One brief paragraph on what would do better next time based on what was learned in the class about SE management and processes.

Next time, I would begin working on the front end much earlier in the semester, even if only the UI could be completed at that stage. I would also aim to use my time more effectively so I can contribute more consistently throughout the project. Starting earlier and managing my time better would allow for steadier progress, more opportunities for iteration, and stronger overall contributions to the team.

f). Anything else you deem important for instructors to know (e.g. why your GitHub count is low)

N/A, I'm proud of the team and our final product.

Christopher Chan

Tejas Rajan;Michael John Thompson;Pei Huan Chang;Kameron Allen Jacob;**+1 other**

Team member self-assessment and contributions

a). Their contributions to the team project and teamwork (technical and any other) in no more than half a page – list item format is OK. List also the role (e.g. front-end lead).

Christopher Chan

Role - Backend Lead

- Search Functionality
  - Courses
  - Departments
- Connected the tutor profile to the backend
- Connected the tutor dashboard to the backend

b). Number of submissions they made to the GitHub team Dev. Branch (explain if this

number is very low)

8 Commits total.

c). One brief paragraph on the main challenges they encountered in team project

One of the biggest challenges when working on this project, as a backend lead, was deciding how to split up the work with our other backend developer. Another challenge was having to deal with merge conflicts, since this was the first big group project that I had to do.

d). One brief paragraph of their own overall experience with GenAI focusing on tasks it was applied to and how useful it was.

Overall, I used GenAI mainly to review the structure of documents and to check code readability. It was very helpful for dealing with debugging issues and understanding how to structure different API calls.

e). One brief paragraph on what would do better next time based on what was learned in the class about SE management and processes.

I learned a lot in this class about software engineering and some of the management processes. Next time, I would want to improve communication and planning for future projects. Setting clearer checkpoints earlier and having would make it a lot easier to know what to get done and keep on track.

f). Anything else you deem important for instructors to know (e.g. why your GitHub count is low)

N/A