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:

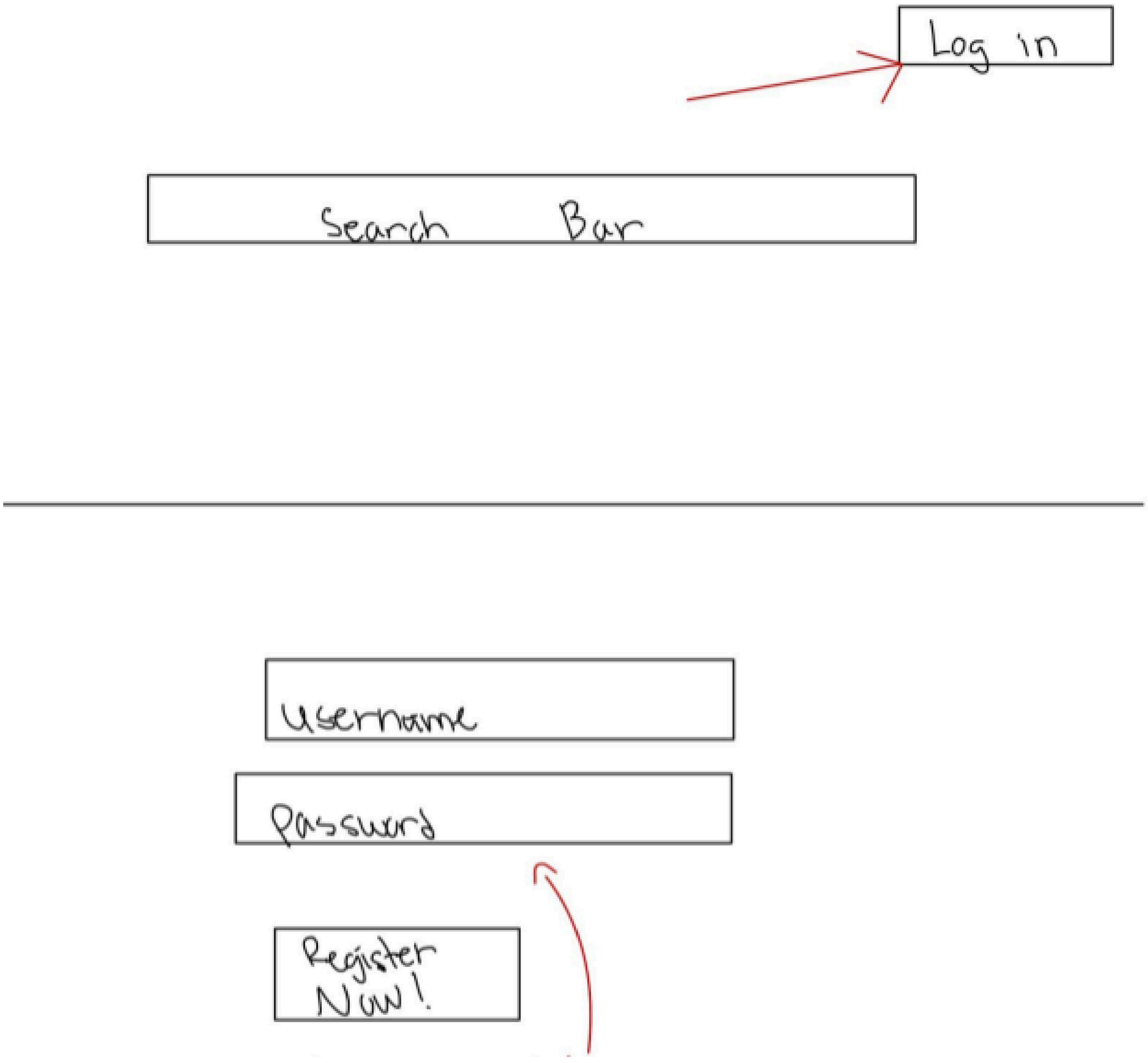
An **SFSU Student** logs into the tutoring platform and searches for a specific **Tutor** by name using the search bar. The system displays matching **Tutor** profiles and their availability from the **Calendar**. The student can view each **Profile** for more details such as ratings and **Review / Feedback**,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

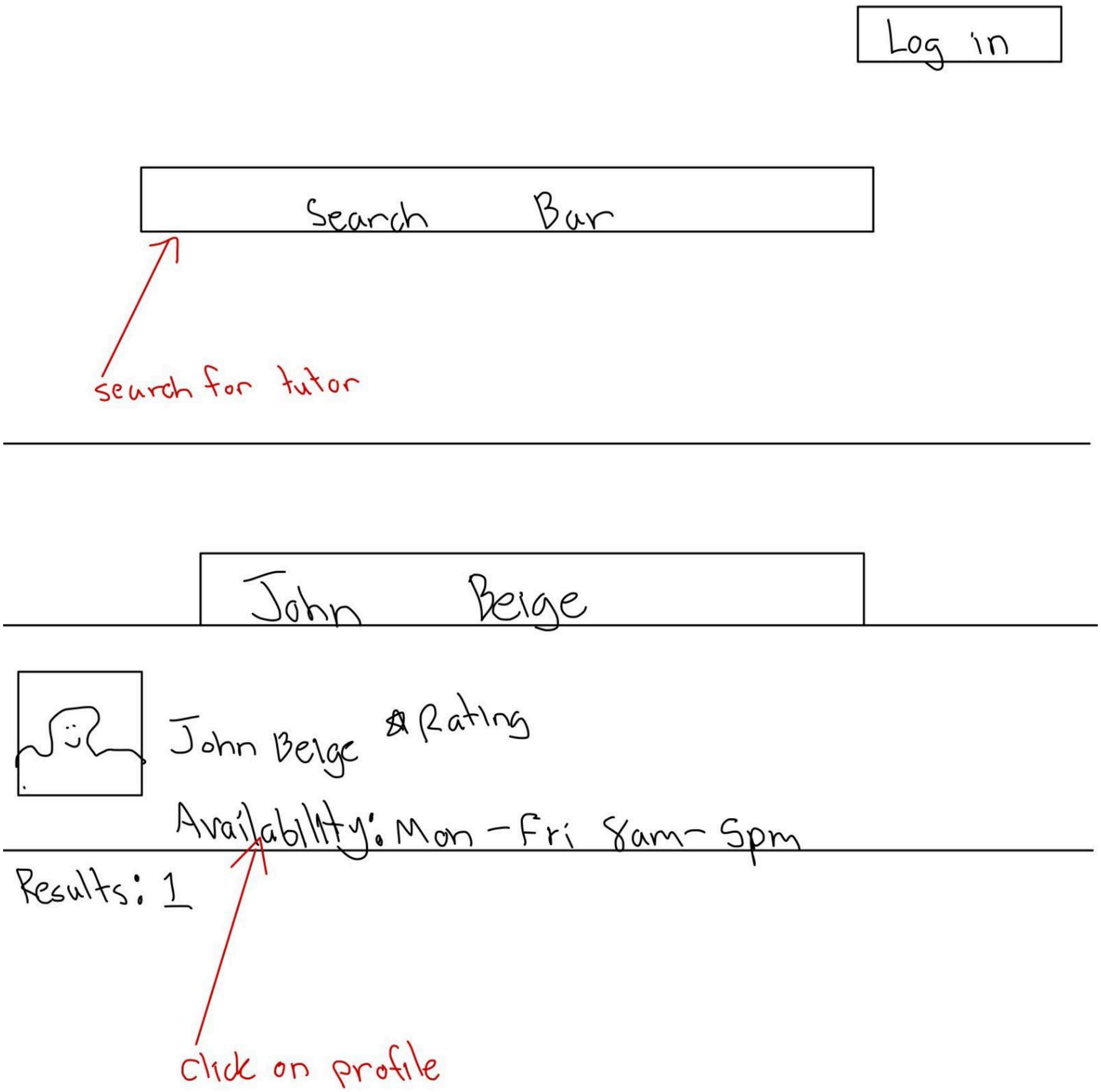
Story Board:

Searching for Tutors by Name



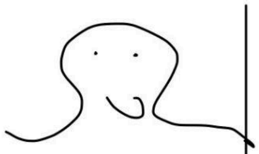
Alex logs in

Searching for Tutors by Name



search for tutor

click on profile



John Beige ☆/5 rating

About me:

Booking filters:

Available Bookings:

Reviews/Feedback:

Use Case 2: Filtering Tutors

Actors: Alex (SFSU Student)

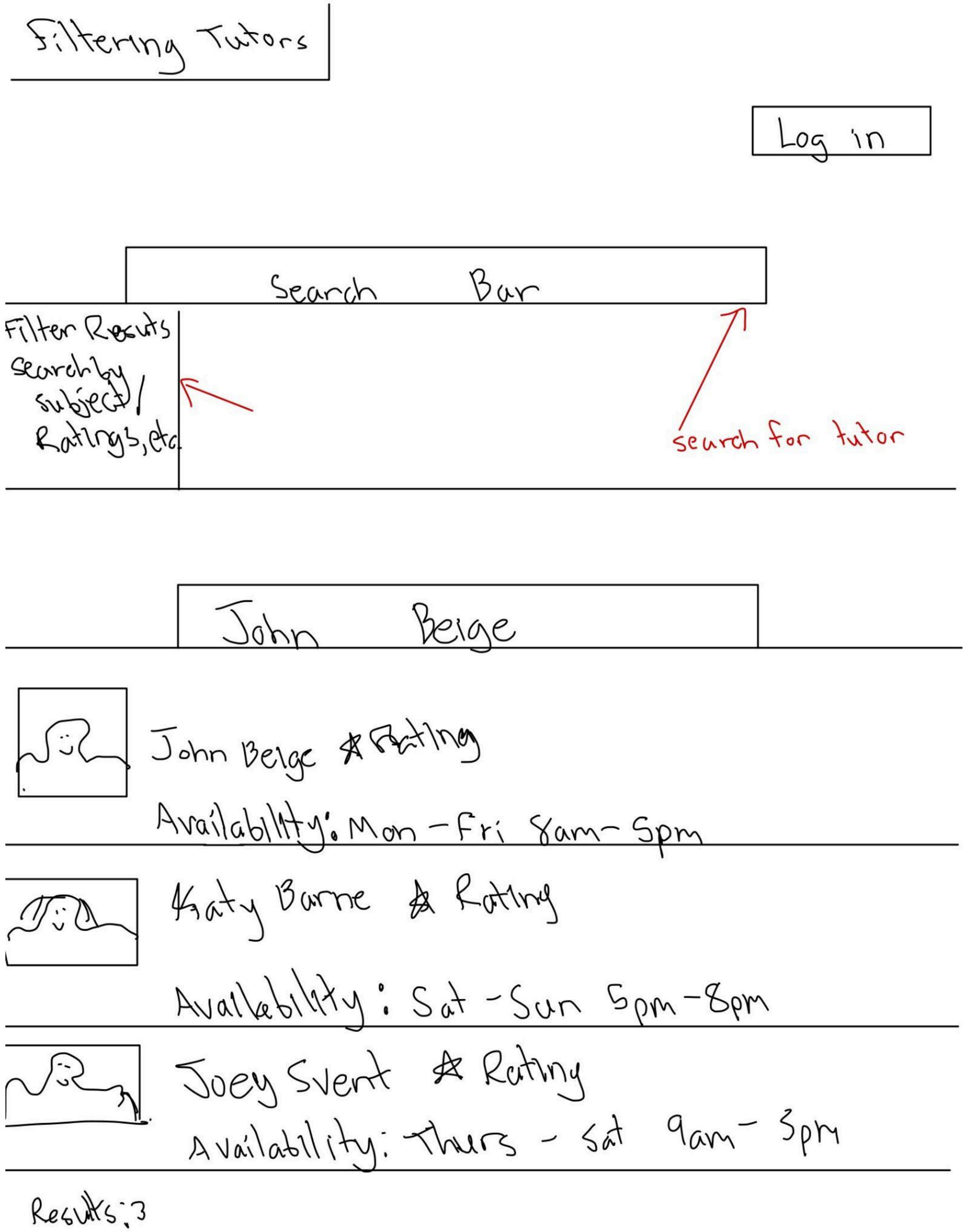
Description:

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 - 3pm
Results: 3

Use Case 3: Guest Browsing

Actors: Jordan Lim (Guest User)

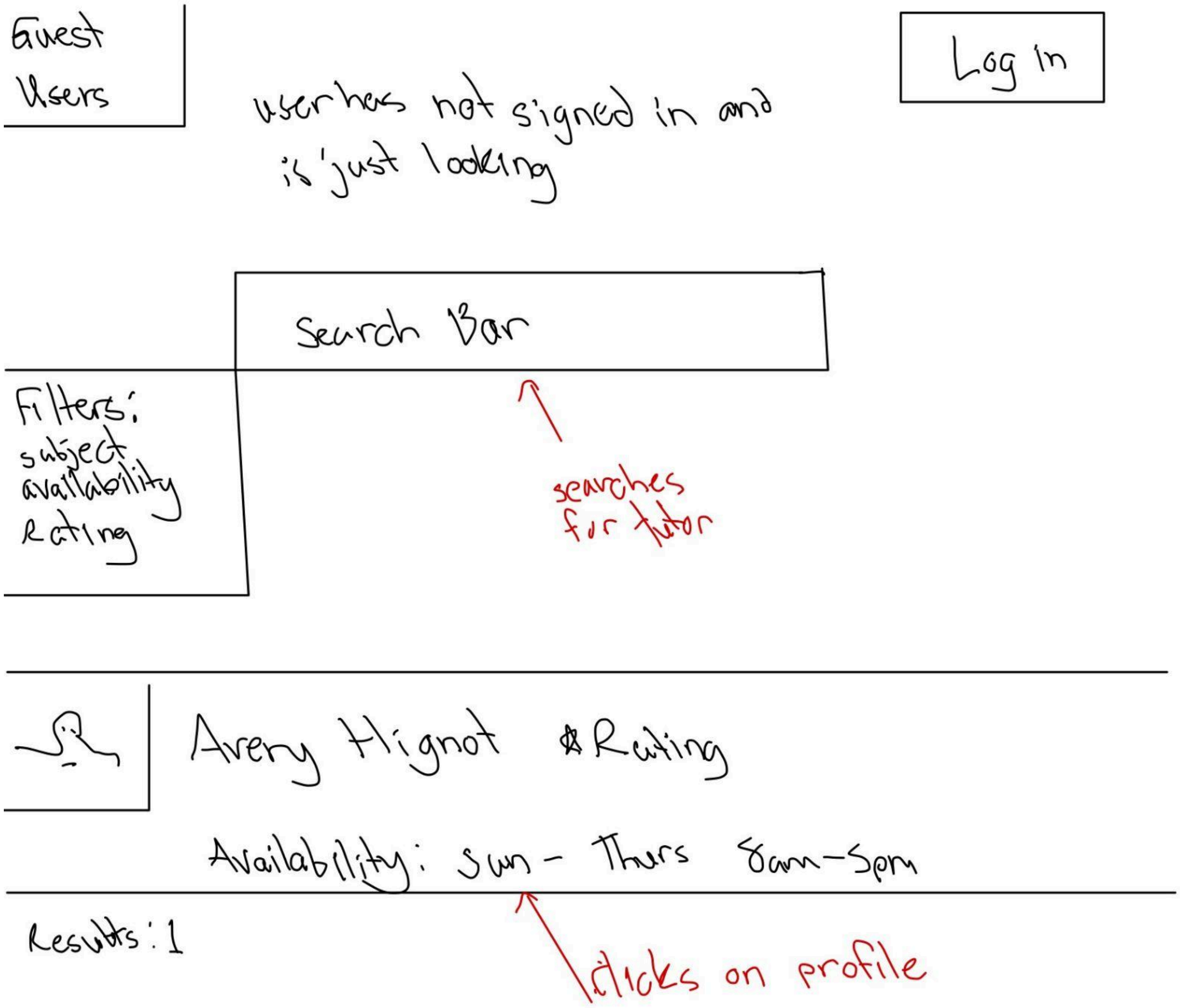
Description:

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.

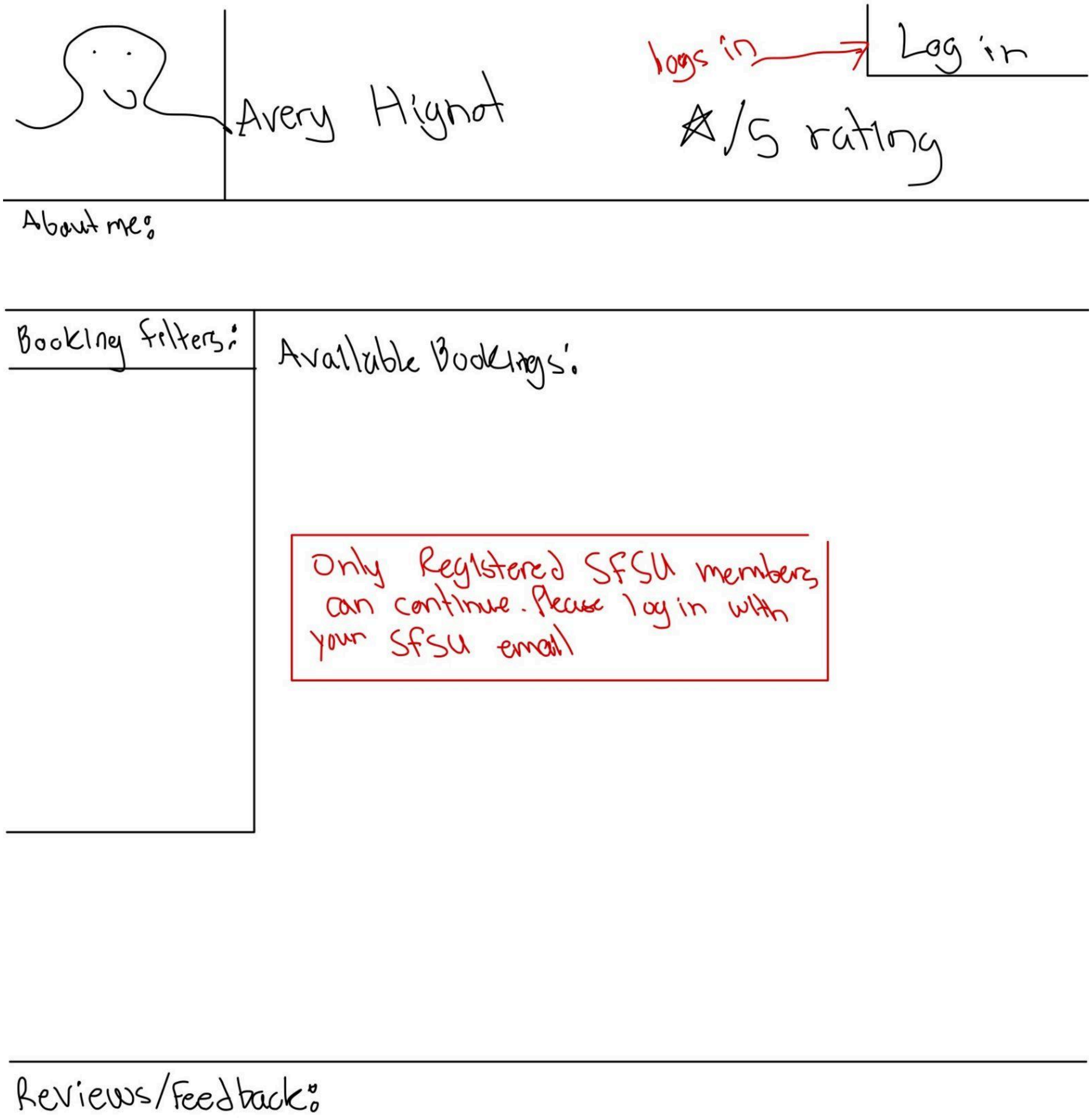
Story Board:



user has not signed in and is 'just looking

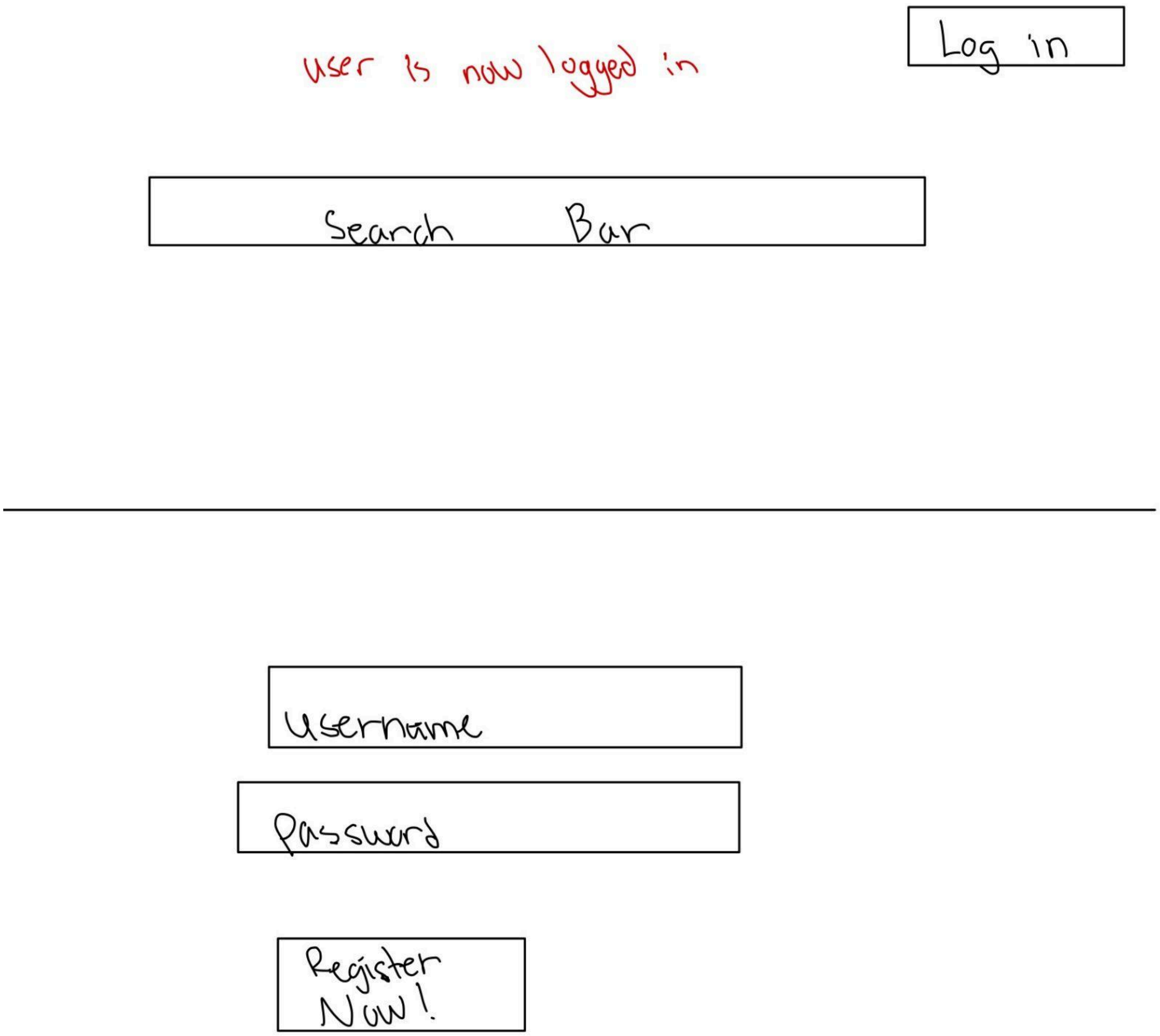
searches for tutor

clicks on profile



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

Searching for Tutors by Name



user is now logged in

Use Case 4: Booking a Tutoring Session

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

Description:

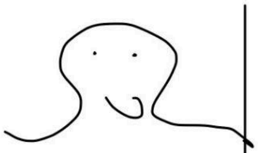
An **SFSU Student** selects a preferred **Tutor** from the search results or **Profile** and clicks “Book Session.” The system displays the **Tutor**’s available slots from the **Calendar**, allowing the student to choose a convenient time. A confirmation panel shows **Session** details including **Subject**, 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

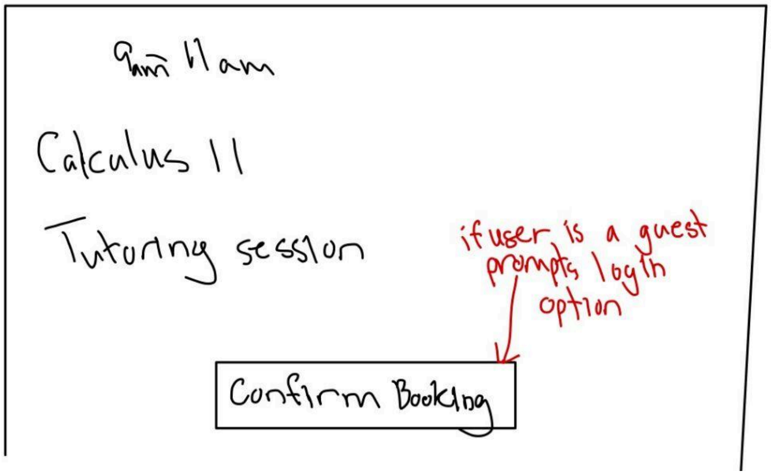
click book session

Reviews/Feedback:

Calendar

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

clicks on this one



if user is a guest prompts login option

Use Case 5: Staff Management of Tutoring Sessions

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

Description:

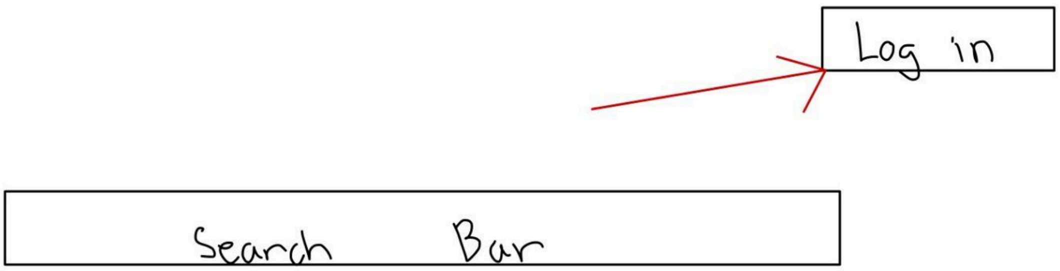
Dr. Lopez, a **Tutor** on the platform has a sudden dental appointment and can't make her Thursday afternoon tutoring session. She logs in to the tutoring platform, navigates to the **Calendar** page, and cancels her scheduled **Session.** This action automatically sends a message to the corresponding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

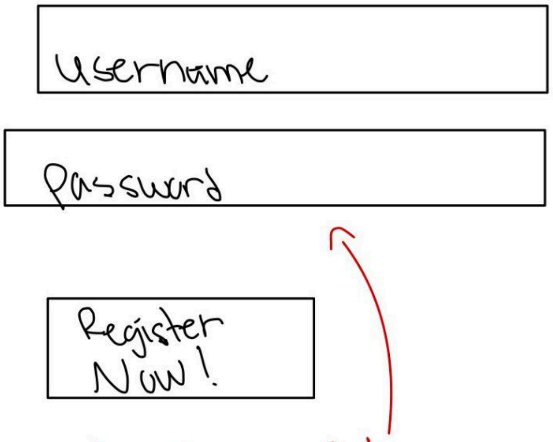
Frequency/Importance: Weekly; critical for maintaining accurate schedules and supporting academic success.

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

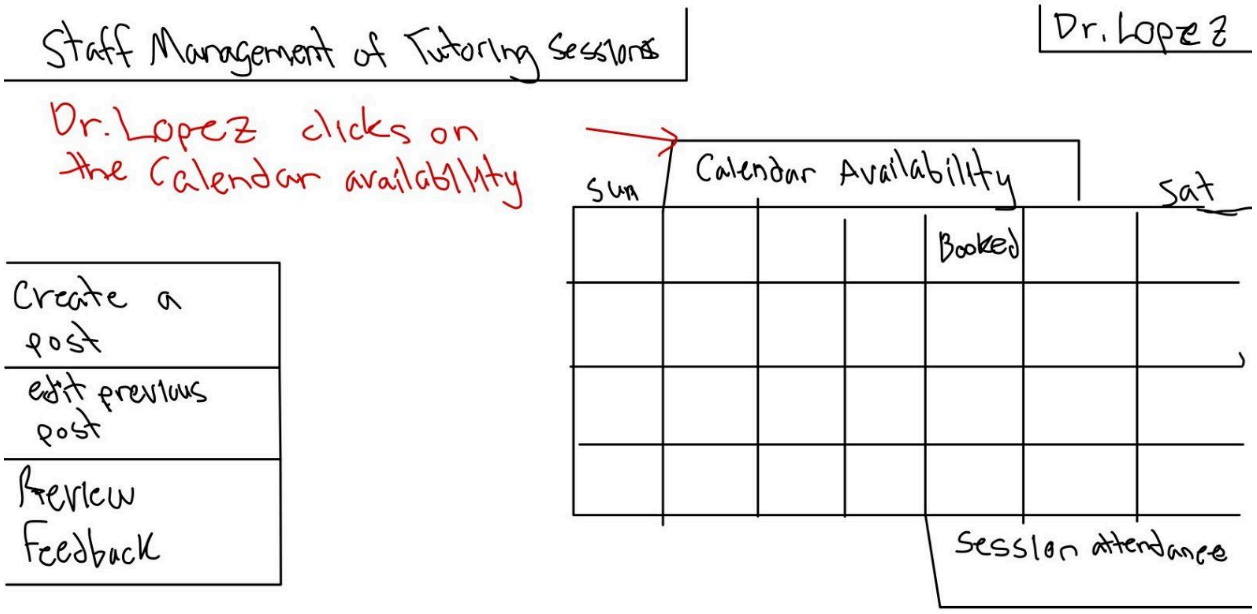
Story Board:

Searching for Tutors by Name





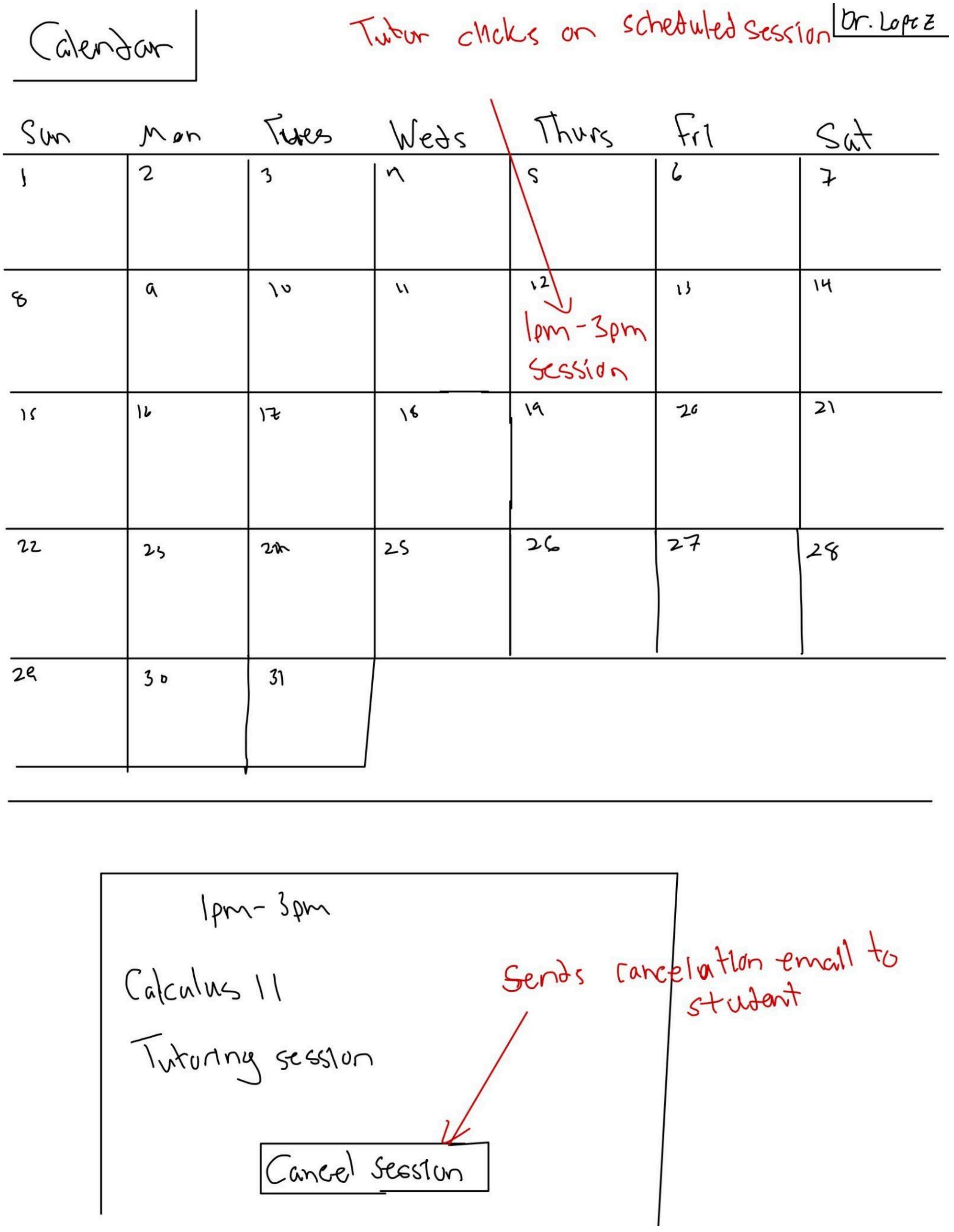
Dr.Lopez logs in as a tutor



Staff Management of Tutoring Sessions

Dr. Lopez

Dr. Lopez clicks on the Calendar availability



Calendar

Dr. Lopez

Tutor clicks on scheduled session

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

Cancel Session

Sends cancelation email to student

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