

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



Pei Huan Chang

To: Michael John Thompson



Tue 12/16/2025 2:57 PM

Hi Michael,

I have reviewed inbox.html for the Messages feature. Overall, the code is well-structured and easy to follow.

Strengths:

- The UI layout and message flow are clear and consistent.
- Variable names such as inboxMessages, sentMessages, and draftMessages are descriptive.
- API calls are logically separated by feature (inbox, sent, drafts, trash).

Suggestions:

- Consider adding a file-level header comment describing the purpose of inbox.html.
- Some longer functions such as renderMessageList and openMessage could benefit from additional inline comments.

Best regards,
Hardy Chang



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