SW Engineering CSC648-848 Spring 2025

Project/application title and name: Gator Market

Section 04 Team 1 Milestone 5

Name	Role	Email
Dev Modi	Team Lead, Backend, Frontend, database	dmodi@sfsu.edu
Yash Pachori	Backend Lead with Database	ypachori@sfsu.edu
Kyle Yuen	Frontend Lead	kyuen4@sfsu.edu
Hsueh-Ta Lu	Scrum Master with Frontend	hlu@sfsu.edu
Daniel	Tech Lead and Github	domstead@sfsu.edu

Revision History

Date	**Notes**
12 May 2025	Initiation of Milestone 5

Topic Index

1) Product Summary

Product Name: Gator Market

Product Description:

Gator Market is a student-exclusive marketplace platform built specifically for San Francisco State University (SFSU). It allows students to buy and sell items safely on campus, including books, dorm supplies, electronics, and more. Unlike public platforms like Craigslist, it verifies all users via SFSU email and includes secure messaging, admin-moderated listings, and AI-powered price suggestions. Gator Market is designed and developed by students, for students, solving everyday campus exchange problems with a trustworthy, school-centered solution.

Final P1 Functionalities (Committed):

For unregistered users:

- Can browse all product listings
- Can search and filter by keyword, category, price, and condition

For registered users:

- Can register using a verified @sfsu.edu email address
- Can log in and manage their account
- Can create, edit, and delete their listings
- Can upload multiple images when creating a listing
- Can send and receive messages via secure in-app chat
- Can get alerts when new listings match saved interests
- Can report suspicious users or listings
- Can rate and leave reviews after a transaction

For admins:

- Admin can view, approve, or reject product listings
- Admin can view and handle reports on listings or users

Deployment URL:

http://csc648g1.me

2) Milestone Document

Milestone 1

SW Engineering CSC648-848 Spring 2025

Project/application title and name: Gator Market

Section 04 Group 1 Milestone 1

Name	Role	Email
Dev Modi	Team Lead, Backend, Frontend, database	dmodi@sfsu.edu
Yash Pachori	Backend Lead with Database	ypachori@sfsu.edu
Kyle Yuen	Frontend Lead	kyuen4@sfsu.edu
Hsueh-Ta Lu	Scrum Master	hlu@sfsu.edu
Daniel	Tech Lead and Github	domstead@sfsu.edu

Revision History

Date	**Notes**
10 March 2025	Initial publication of M1 document

Topics for the milestone

Executive Summary

In an era of digital connectivity and on-demand convenience, San Francisco State University students face a persistent challenge: finding a safe, efficient, and reliable way to buy and sell personal items. Whether it's textbooks, dorm furniture, electronics, or other essentials, students often rely on platforms like Craigslist or Facebook Marketplace. However, these platforms present significant risks, including scams, unverified buyers and sellers, and transactions that require meeting strangers in unsecured locations. To address these issues, we introduce SFSU BUY & SELL PLATFORM, an exclusive SFSU student marketplace designed to create a secure, seamless, and community-driven buying and selling experience tailored specifically for the university.

At the core of SFSU BUY & SELL PLATFORM lies an unwavering commitment to trust and security. By requiring "SFSU email" verification, we ensure that only verified students can participate, fostering a safe and student-only environment where users can confidently list and purchase items. Additionally, our built-in in-app messaging system allows buyers and sellers to communicate securely without sharing personal phone numbers or relying on third-party messaging services, reducing the likelihood of fraud and ensuring peace of mind for both parties.

The SFSU Buy & Sell Platform offers AI-powered pricing suggestions, helping students set competitive prices and ensuring fair market value. It's group buying feature allows students to save money on bulk purchases like textbooks and electronics. Unlike generic marketplaces, the platform focuses on SFSU, eliminating irrelevant listings and fostering a tight-knit community. With an intuitive interface, users can easily create listings, filter results, and receive notifications for items matching their interests, providing a seamless experience for all.

Behind this initiative is a team of passionate SFSU students, each bringing expertise in software development, UX design, and business strategy. Being students ourselves, we recognize the daily hurdles involved in campus transactions, and we are committed to creating a marketplace that's built by students, for students. Our varied backgrounds enable us to infuse innovative ideas into the project and continually refine the platform to adapt to the ever-changing needs of our peers. With the right backing, SFSU BUY & SELL PLATFORM has the potential to become the go-to hub for SFSU transactions, transforming how students buy and sell while promoting a safer, smarter, and more budget-friendly campus economy.

Personas

Persona 1: Emily Chen – The Budget-Conscious Buyer

- **Age:** 20
- **Major:** Business Administration
- **Attitude:** Practical, money-savvy, and always on the lookout for a good deal. She values sustainability and prefers second-hand items over new ones to save money and reduce waste.
- **Skills:** Comfortable navigating online marketplaces but cautious about scams and poor-quality items. She compares prices before making a decision.
- **Limitations:** Works part-time and has a tight budget, making her hesitant about purchasing non-essential items. Prefers cash transactions but is willing to use payment apps.
- **Pain Points:**
    - Difficulty verifying seller credibility and item condition.
    - Prefers not to meet strangers off-campus.
    - Struggles to find the right listings quickly.
- **Goals:**
    - Find affordable textbooks, dorm furniture, and electronics from fellow students.
    - Get notifications for new listings that match her needs.
    - Ensure safe and convenient transactions.

Persona 2: Jason Patel – The Frequent Seller

- **Age:** 22
- **Major:** Computer Science
- **Attitude:** Entrepreneurial, resourceful, and always looking for ways to make extra money. He enjoys decluttering and reselling items for profit.
- **Skills:** Experienced with online marketplaces, takes high-quality item photos, and understands how to price competitively.
- **Limitations:**
    - Limited storage space in his dorm, so he wants quick sales.
    - Gets frustrated with unreliable buyers who cancel last minute.
    - Prefers digital payments but sometimes deals with cash.
- **Pain Points:**
    - Dealing with lowball offers and flaky buyers.
    - Managing multiple listings across different platforms.
    - Ensuring safe, hassle-free meetups.
- **Goals:**
    - Sell textbooks, tech gadgets, and old gaming consoles before moving out.
    - Use a trusted platform where buyers are SFSU students only to avoid scams.
    - Automate messaging responses to common buyer questions.

Persona 3: Sarah Martinez – The Convenience Seeker

- **Age:** 21
- **Major:** Nursing
- **Attitude:** Busy, efficiency-driven, and values convenience over price. Prefers **ready-to-go** deals instead of prolonged negotiations.
- **Skills:** Familiar with online shopping, prefers mobile apps, and avoids time-consuming transactions.
- **Limitations:**
    - No time for in-person meetups due to her demanding coursework.
    - Prefers verified sellers to reduce risk.
    - Doesn't like haggling over prices.
- **Pain Points:**
    - Frustrated by delayed responses from sellers.
    - Annoyed by listings without enough details or photos.
    - Prefers secure payment methods but doesn't want to deal with cash.
- **Goals:**
    - Quickly find nursing books, scrubs, and study materials without wasting time.
    - Use a buy-now option with secure, pre-arranged pickup spots.
    - Receive notifications for items she frequently searches for.

Persona 4: Marcus Thompson – The First-Time User

- **Age:** 19
- **Major:** Psychology
- **Attitude:** Cautious, hesitant about online transactions, and new to buying/selling items. Wants a simple, guided experience.
- **Skills:** Basic familiarity with e-commerce, prefers step-by-step help when posting a listing.
- **Limitations:**
    - Doesn’t know how to price items competitively.
    - Worried about safety when meeting buyers in person.
    - Not sure what items will sell quickly.
- **Pain Points:**
    - Nervous about getting scammed.
    - Unsure how to describe and photograph items properly.
    - Afraid his listings won’t attract buyers.
- **Goals:**
    - Successfully sell his extra dorm furniture to upperclassmen before moving out.
    - Learn how to make safe transactions on campus.
    - Gain confidence using online marketplaces.

Persona 5: Tiffany Nguyen – The Social Seller & Buyer

- **Age:** 20
- **Major:** Marketing
- **Attitude:** Outgoing, community-driven, and enjoys interacting with students through buy-and-sell groups.
- **Skills:** Experienced in social media sales, uses creative descriptions to attract buyers.
- **Limitations:**
    - Prefers selling to students she already knows or has mutual friends with.
    - Avoids long-distance meetups due to her packed social schedule.
    - Wants a more interactive and fun marketplace experience.
- **Pain Points:**
    - Doesn't trust random sellers with no profile details.
    - Wishes the platform had more social features like reviews, community groups, or shared wishlists.
    - Finds text-heavy listings boring—prefers visual, Instagram-style listings.
- **Goals:**
    - Sell clothes, accessories, and dorm decor to students with similar interests.
    - Engage in trendy marketplace features like "trending now" or "best deals this week."
    - Connect with mutual friends for safer transactions.

High-Level Use Cases for SFSU Buy & Sell Platform

1. Finding and Purchasing Items from Verified SFSU Students

Summary: Ensuring safe and exclusive exchange by limiting interactions to verified SFSU students.

Emily, a sophomore at SFSU, is preparing for the upcoming semester and needs a used biology textbook to save money. She has tried other platforms like Facebook Marketplace and Craigslist, but she is wary of scammers and overpriced listings. She logs into the SFSU Buy & Sell platform, where only verified students with an active SFSU email address can list and purchase items. This feature gives her peace of mind, knowing she is dealing with her fellow students instead of unknown strangers.

Using the platform's search and filter functionality, Emily quickly narrows her options by selecting the "Textbooks" category and filtering results by "Condition: Used" and "Price: Under $50." She finds a listing that meets her criteria, posted by another student who has a verified SFSU badge next to their name. Before committing, she checks the seller’s rating and transaction history, which reassures her of their reliability.

Emily messages the seller using the in-app chat feature, asking if the book is still available and where they can meet. The seller responds quickly, and they agree to meet at an SFSU-designated safe exchange zone, such as the library or student union, which is monitored for security. Emily appreciates the ease of communication. The exchange was successful.

2. Selling Items Quickly and Efficiently with Smart Pricing Suggestions

Summary: Helping students price items fairly and complete sales efficiently.

Jason, a senior computer science student, is moving out of his apartment and needs to sell his gaming monitor, desk chair, and dorm furniture before graduation. He wants to make some quick cash but isn't sure how to price his items competitively.

Jason logs into the SFSU Buy & Sell platform and starts listing his gaming monitor. The platform's Smart Pricing Tool suggests a price range based on similar items sold by other SFSU students, considering factors like brand, condition, and demand. This helps Jason avoid overpricing, which could slow down his sale, or underpricing, which could make him lose money.

Within a few hours, Jason receives several messages from interested buyers. To reduce back-and-forth negotiations, he enables the Quick Sale option, allowing buyers to make instant offers. He also selects a designated campus meet-up point near his dorm, making it easy for buyers to pick up items at their convenience.

One buyer, Emily, expresses strong interest in the monitor and appreciates the convenience of the designated campus meet-up point. After discussing the details with Jason, she decides to explore her options further. Despite no transaction taking place, Jason's positive interaction and responsiveness help him build credibility for future sales on the platform.

3. Reserving Items and Group Buying for Student Discounts

Summary: Enabling students to buy in bulk together for better deals.

Sarah, a nursing major, needs to buy a stethoscope, medical scrubs, and anatomy flashcards for her upcoming clinical rotations. However, these items are expensive when purchased individually, and she is hoping to find a more affordable option.

While browsing the SFSU Buy & Sell platform, she notices that some listings have a "Group Buy" feature, allowing multiple students to purchase in bulk and receive a discount. She finds a listing for stethoscopes, where the seller offers 10% off for group purchases of five or more. Sarah joins the Group Buy listing, and within two days, four other nursing students also sign up.

Once the required number of buyers is reached, the seller confirms the discounted price and arranges a bulk delivery to campus, making it convenient for all participants. Sarah pays her share using the platform's secure payment integration, and the group picks up their items from the seller in a pre-arranged drop-off location at the student center.

This feature not only saves Sarah money but also helps sellers sell multiple items at once, benefiting both parties. It fosters a sense of community-driven commerce among students, making bulk buying an efficient and cost-effective option.

4. First-Time Sellers Using Guided Listing and Seller Ratings

Summary: Helping inexperienced users confidently list and sell their items.

Marcus, a freshman at SFSU, has never sold anything online before but wants to get rid of a mini fridge and dorm lamp that he no longer needs. However, he is unsure how to write a good listing or how much to charge.

When he opens the platform, Marcus is guided through an intuitive step-by-step listing process that helps him:

1) Take high-quality photos by suggesting good lighting and angles.
2) Write an effective item description using pre-filled templates.
3) Set a competitive price based on market trends at SFSU.
4) Choose a safe meetup location or enable on-campus drop-off lockers.

Since Marcus is a first-time seller with no transaction history, buyers may hesitate to purchase from him. However, the platform allows him to verify his account using his SFSU student ID, giving him a verified seller badge that builds trust. He also sees an option to offer a first-time seller discount, encouraging buyers to take a chance on him.

Shortly after listing his items, Marcus receives messages from interested buyers. He uses the in-app chat to coordinate a sale and successfully sells his mini fridge within a week. Thanks to the rating system, the buyer leaves Marcus a positive review, which boosts his credibility for future sales.

This feature makes it easy for new sellers to participate confidently, ensuring a thriving marketplace where even first-time users can successfully buy and sell items.

5. Finding Free or Discounted Essentials Through Student Giveaways

Summary: Encouraging sustainable and community-driven transactions.

Tiffany, a marketing major, is moving into her new apartment near campus and needs basic furniture and kitchen supplies. She is on a tight budget, so she hopes to find affordable or free items from other students.

She logs into the platform and explores the “Student Free & Discounted” section, which lists items that students are giving away or selling at a minimal cost. This feature encourages sustainability and waste reduction, allowing students to pass down used but functional items instead of throwing them away.

Tiffany finds a free microwave and a $10 desk chair, listed by graduating seniors who no longer need them. She messages the sellers and arranges a pickup at the dormitory donation station, a designated drop-off area where students can leave unwanted items for others to claim.

This feature not only helps students save money but also promotes a culture of sharing and sustainability at SFSU. Students can find essential items without financial strain, making college life more affordable and environmentally friendly.

List of Main Data Items and Entities

1. User

Description: Represents a person interacting with the platform, which can be classified into different user types:

- **Registered User:** A student who has signed up and can buy, sell, and interact with other users.
- **Admin User:** A platform moderator with additional privileges to manage content and enforce rules.
- **Anonymous User:** A visitor who can browse products but must register to purchase or interact with sellers.

Attributes:

- **User ID** (unique identifier)
- **Name**
- **Email** (used for login and notifications)
- **Role** (e.g., buyer, seller, admin, anonymous user)
- **Registration date** (only applicable to registered users)
- **Profile picture** (optional)
- **Contact details** (only visible to admin or during transactions)

2. Product

Description: An item listed for sale by a registered user.

Attributes:

- **Product ID** (unique identifier)
- **Seller ID** (reference to **Registered User**)
- **Name** (title of the product)
- **Description** (detailed information about the product)
- **Category** (e.g., electronics, furniture, clothing, etc.)
- **Price**
- **Condition** (new, used, etc.)
- **Image(s)**

- **Date listed**
- **Status** (available, sold, reserved)

3. Category

Description: Represents the classification of products.

Attributes:

- **Category ID** (unique identifier)
- **Name** (e.g., electronics, furniture, clothing)
- **Parent category** (for subcategories)

4. Review

Description: A rating or feedback given by buyers and sellers after an interaction.

Attributes:

- **Review ID** (unique identifier)
- **Reviewer ID** (reference to **Registered User**)
- **Reviewee ID** (reference to **Registered User**)
- **Product ID** (optional reference)
- **Rating** (1-5 stars)
- **Comment**
- **Date of review**

5. Wishlist

Description: A collection of products a user is interested in purchasing later.

Attributes:

- **Wishlist ID** (unique identifier)
- **User ID** (reference to **Registered User**)
- **Product ID(s)** (reference to **Product**)

- **Date added**

6. Message

Description: A message exchanged between users for product inquiries.

Attributes:

- **Message ID** (unique identifier)
- **Sender ID** (reference to **Registered User**)
- **Receiver ID** (reference to **Registered User**)
- **Message content**
- **Timestamp**
- **Status** (read, unread)

7. Admin Actions

Description: Actions taken by an **Admin User** to manage platform content.

Attributes:

- **Admin ID** (reference to **Admin User**)
- **Action type** (delete user, block product, etc.)
- **Target entity** (User, Product, etc.)
- **Action description**
- **Timestamp**

8. Listing Report

Description: A report submitted by a user regarding a suspicious, inappropriate, or fraudulent listing.

Attributes:

- **Report ID** (unique identifier)
- **Reporter ID** (reference to **Registered User**)
- **Product ID** (reference to **Product**)
- **Reason** (e.g., scam, misleading info, inappropriate content)
- **Additional comments**
- **Date reported**
- **Status** (pending, reviewed, resolved)

Data Glossary (Google Analytics data not included)

ACCOUNT:

Field Name	Data Type	Description
user_id	INT	Unique identifier for each user, auto-incremented, serves as the primary key.
username	VARCHAR(50)	A unique username chosen by the user for identification and login purposes.
password_hash	VARCHAR(255)	The user's password, stored as a hashed value for security (e.g., using bcrypt).
first_name	VARCHAR(50)	The user's first name, for personalization and identification.
last_name	VARCHAR(50)	The user's last name, for personalization and identification.
email	VARCHAR(100)	The user's university email (e.g., ending in @sfsu .edu), unique, used for login and verification.
verification_status	ENUM	Indicates if the user is a verified current SF State student (‘permanent’, 'annual', 'not verified').

Field Name	Data Type	Description
phone_number	VARCHAR(15)	An optional contact phone number for verification purposes. (can be NULL)
profile_picture_url	VARCHAR(255)	An optional URL to the user's profile picture (can be NULL).
date_joined	TIMESTAMP	Timestamp recording when the user created their account, defaults to current time.
last_login	TIMESTAMP	Timestamp of the user's most recent login, for tracking activity (can be NULL).
user_role	ENUM	The user's role on the platform ('user', 'moderator', 'admin'), defaults to 'user'.
account_status	ENUM	The current status of the user's account ('active', 'inactive/banned', 'deleted'), defaults to 'active'.

POST:

Field Name	Data Type	Description
post_id	INT	Unique identifier for each post, auto-incremented, serves as the primary key.
user_id	INT	Foreign key referencing the Users table, indicating the user who created the post.
title	VARCHAR(100)	The title of the post, limited to 100 characters for brevity and clarity.
description	TEXT	A detailed description of the item being sold or sought, allowing flexibility in length.
category_id	INT	Foreign key referencing the Categories table, specifying the item's category.

price	DECIMAL(10,2)	The price of the item, with two decimal places (e.g., supports up to 99999999.99).
condition	ENUM('new', 'used', 'refurbished')	The condition of the item, restricted to predefined options for consistency.
location	VARCHAR(100)	Uses geocode to work with google maps API.
created_at	TIMESTAMP	Timestamp of when the post was created, defaults to the current time for sorting purposes.
status	ENUM('active', 'sold', 'deleted')	The current status of the post, managing its lifecycle (defaults to 'active').
approval_status	ENUM('pending', 'approved', 'rejected')	Indicates if the post has been reviewed by moderators (defaults to 'pending').

List of High-Level Functional Requirements

1. **Browsing, Searching, and Reviewing Item Information:** User shall search and look for products
2. **Contacting Sellers:** User shall contact sellers regarding product issues
3. **Uploading Sales Item Information:** User shall input and update product information
4. **Dashboard for Sellers:** Provides a user-friendly interface for sellers to manage their products
5. **Site Administration:** Offers tools for user management and website maintenance
6. **Wishlist:** User shall save products they wish to purchase later
7. **Chat System:** Users shall chat among themselves easily.
8. **Item Reservation System:** Users shall be able to reserve an item to indicate interest before purchasing, allowing sellers to track potential buyers.
9. **Favorite & Watchlist Feature:** Users shall be able to save items to a favorites list or watchlist to easily revisit them later.
10. **Rating and Reviews:** User shall post reviews and rate products
11. **Report Errors:** Users shall report system or product-related issues
12. **Image and Video Upload for Listings:** Users shall be able to upload multiple images and short videos for their listings to better showcase product details and condition.
13. **Recommendations:** User shall have personalized product recommendations
14. **Group Buying Features:** User(s) shall collaborate on purchases through a shared cart system
15. **Responsive and Intuitive UI:** Users shall have an easy UI to use
16. **SFSU Verification:** User must be an SFSU student
17. **Search & Filter Enhancements:** Users shall be able to apply advanced search filters (e.g., category, condition, price range, location) to quickly find relevant listings.
18. **Seller Reputation System:** Users shall be able to leave feedback and ratings for sellers based on past transactions to enhance trust and credibility on the platform.
19. **Analytics and stats:** User shall be able to see product performance and activity
20. **Computer/Mobile Use:** Compatible across various devices, enhancing user accessibility

List of Non-Functional Requirements

1. Application shall be developed, tested and deployed using tools and cloud 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 render well on mobile devices (no native app to be developed)
4. Posting of sales information and messaging to sellers 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 or chat services shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application
14. Pay functionality, if any (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 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, Spring 2025. For Demonstration Only" at the top of the WWW page Nav bar. (Important so as to not confuse this with a real application). You have to use this exact text without any editing.

Competitive Analysis

Feature	Our Platform	FB Marketplace	Craigslist	OfferUp
SFSU Student Verification	✅	❌	❌	❌
Chat system for transactions	✅	✅	❌	✅
Category-Based Listings	✅	✅	✅	✅
Payment Integration	❌	❌	❌	✅
User ratings and reviews	✅	✅	❌	✅
Admin moderation	✅	❌	❌	❌
Wishlist Feature	✅	❌	❌	❌
Event-Based Listings	✅	❌	❌	❌

Our platform stands out by offering SFSU student verification which makes this platform **exclusive to SFSU students**. This ensures a safer and more trustworthy marketplace compared to platforms such as Facebook Marketplace and Craigslist, which are open to the public and thus carry a certain degree of risk. And unlike Craigslist, our platform includes a built-in chat system for seamless communication. While this does make our product similar to Facebook Marketplace and OfferUp, what gives us an edge over those competitors is our **admin moderation** to prevent scams and inappropriate listings. The user rating and review feature also aids in further moderation because users will be able to inform other users whether a certain seller or service provider is worth their money or not. We share a review system with Facebook Marketplace and OfferUp but Craigslist is lacking one. Our wishlist feature enhances the user experience by allowing students to track desired items and services. This is a feature that none of our competitors have. Another feature that sets us apart is our **event-based listings** feature which caters to student-specific needs. For example, an event for a textbook exchange and meet-and-greet could be posted at the start of the semester. A feature like this can only work on a platform that is built by and for students, which none of our competitors are. These unique functions provide a more secure and tailored experience to the SFSU community than existing alternatives.

High-level system architecture and technologies used

Software:
Flask ver 3.02: Python web framework for the backend server. Also very supportive of messaging and various auth-related features.
MySQL 8.036 (general availability): relational database as required by the CEO.
NGINX 1.26.0 (mainline): open source web server for handling requests and serving static content
Swagger UI 5.18.2: API documentation

Deployment:
AWS EC2 with Docker 27.2.0. Simplifies deployment and potential scalability and reliability. Granted, the whole idea of this site is directly geographically limited in scope so physical hosting might be better in the long run, but scalability might be a good idea too as this is the kind of idea that would receive way more traffic at specific times of year for a week or 2.
Note: Our free tier t2 micro only has 1GB RAM, 30GB SSD, 1 VPU.

Supported Browsers:
Google Chrome (Version 100 and above) – This is one of the most widely used browsers on both desktop and mobile.
Mozilla Firefox (Version 95 and above) – A popular open-source browser known for privacy and developer-friendly features.

Database:
mySQL 8.0.36: A relational SQL database required by the CEO. Stores all critical data on the Instance.

Front-End Frameworks:
Vite 5.4.14: Build took and deployment server for fast development/local server debug.
React 19: Javascript library for building user interfaces.
Axios 1.7.7: (could be changed to fetch or ky if we want mobile to look better more easily) http client for API requests

Major external APIs:

Google Analytics 4:will be used to track user website use and navigation as requested by the CEO. Has a free tier.

Google maps API 3.57: can be used for location navigation and help users locate each other. Has a free tier.

Possibly:

Cloudinary/imgix/skip it/urlbox/etc.: help with thumbnail images, some like cloudinary have a free tier

Other tools:

Namecheap: Free domain and SSL certificate via GitHub Student Developer Pack for secure deployment

Doxygen 1.12.0: assists in generating code documentation

Google maps API: can be used for location navigation and help users locate each other.

Trello: task management and planning

Google maps API: can be used for location navigation and help users locate each other

Visual Studio Code 1.87.0: The primary development environment.

Use of GenAI Tools like ChatGPT for Milestone 1

As part of our development process for Milestone 1, our team utilized **ChatGPT** to assist with brainstorming, drafting, and structuring the document. The objective was to explore how this tool could improve efficiency, creativity, and clarity in our software engineering workflow. Below, we outline our experience with ChatGPT for this milestone.

GenAI Tool Used

- **ChatGPT 4.0** – Used for generating content, refining technical descriptions, and structuring reports.

Tasks Where ChatGPT Was Used and Effectiveness Ratings

We used ChatGPT for several tasks and assessed its effectiveness based on the quality of results and the time saved.

Task	Rating	Description of Use
Executive Summary	MEDIUM	Helped structure the summary and refine the wording to make it concise and engaging.
Personas & Use Cases	LOW	Generated sample personas based on existing marketplace apps and helped describe common user scenarios. Required manual revision for relevance.
Functional Requirements	MEDIUM	Provided structured formats and suggested functional requirements, but required human validation for alignment with project goals.
Competitive Analysis	LOW	Helped summarize key differences between our platform and competitors, but manual research was still needed.
Document Formatting	HIGH	Helped with structuring sections, refining grammar, and ensuring consistency.
Brainstorming Unique SFSU Features	LOW	Provided ideas for custom SFSU-specific functions but needed human refinement.

Key Examples and Prompts Used

1. **Executive Summary Prompt**
    - **Prompt:** “Create an executive summary for an SFSU student buy-and-sell platform emphasizing its uniqueness.”
    - **AI Response (Edited):** “The SFSU Buy & Sell Platform is a secure, student-only marketplace with university authentication to prevent scams.”
2. **Functional Requirements Prompt**
    - **Prompt:** “List some functional requirements for a student-to-student marketplace web application, categorized by user roles.”
    - **AI Response (Edited):**
        - **Users:** Post listings, contact sellers, leave reviews, report items, View listings, search by category.
        - **Admin Features:** Moderate content, verify users, track flagged items.

Benefits and Limitations Observed

Benefits

- **Time-saving** – AI significantly reduced the time required for drafting and structuring documents.
- **Idea generation** – Helped us brainstorm and improve clarity in requirements.
- **Improved organization** – Ensured consistency in formatting and writing style across different document sections.

Limitations

- **Contextual accuracy** – AI sometimes generated responses that were too generic or not fully aligned with our project's scope.
- **Manual validation required** – While AI-generated content provided a strong starting point, human review was necessary to refine and validate the outputs.
- **Limited real-time collaboration** – AI did not replace brainstorming sessions among team members but rather complemented them.

Final Thoughts on AI Use

Overall, our team found **ChatGPT highly beneficial** for structuring content, generating ideas, and improving efficiency. While the tool was not perfect and required human oversight, it helped accelerate our workflow, particularly in **drafting personas, functional requirements, and competitive analysis**. For future milestones, we plan to refine our approach by leveraging ChatGPT for more technical aspects.

Team And Roles

Name	Role
Dev Modi	Team Lead, Backend, Frontend, database
Yash Pachori	Backend Lead with Database
Kyle Yuen	Frontend Lead
Hsueh-Ta Lu	Scrum Master
Daniel	Tech Lead and Github

Team Lead Checklist

So far, all team members are fully engaged and attending team sessions when required
Ans) Done

• Team found a time slot to meet outside of the class
Ans) 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
Ans) Done

• Team reviewed class slides on requirements and use cases before drafting Milestone 1
Ans) Done

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

• Team lead checked Milestone 1 document for quality, completeness, formatting and compliance with instructions before the submission
Ans) Done
• Team lead ensured that all team members read the final M1 and agree/understand it before submission
Ans) Done

• Team shared and discussed experience with GenAI tools among themselves
Ans) Done

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

Milestone 2

SW Engineering CSC648-848 Spring 2025

Project/application title and name: Gator Market

Section 04 Team 1 Milestone 2

Name	Role	Email
Dev Modi	Team Lead, Backend, Frontend, database	dmodi@sfsu.edu
Yash Pachori	Backend Lead with Database	ypachori@sfsu.edu
Kyle Yuen	Frontend Lead	kyuen4@sfsu.edu
Hsueh-Ta Lu	Scrum Master	hlu@sfsu.edu
Daniel	Tech Lead and Github	domstead@sfsu.edu

Revision History

Date	**Notes**
10 March 2025	Initial publication of M1 document
24 March 2025	Revised for Milestone 2 Part 1 Submission

Topics for the milestone

Executive Summary

In an era of digital connectivity and on-demand convenience, San Francisco State University students face a persistent challenge: finding a safe, efficient, and reliable way to buy and sell personal items. Whether it's textbooks, dorm furniture, electronics, or other essentials, students often rely on platforms like Craigslist or Facebook Marketplace. However, these platforms present significant risks, including scams, unverified buyers and sellers, and transactions that require meeting strangers in unsecured locations. To address these issues, we introduce "Gator Market", an exclusive SFSU student marketplace designed to create a secure, seamless, and community-driven buying and selling experience tailored specifically for the university.

At the core of "Gator Market" lies an unwavering commitment to trust and security. By requiring "SFSU email" verification, we ensure that only verified students can participate, fostering a safe and student-only environment where users can confidently list and purchase items. Additionally, our built-in in-app messaging system allows buyers and sellers to communicate securely without sharing personal phone numbers or relying on third-party messaging services, reducing the likelihood of fraud and ensuring peace of mind for both parties.

The "Gator Market" offers AI-powered pricing suggestions, helping students set competitive prices and ensuring fair market value. It's group buying feature allows students to save money on bulk purchases like textbooks and electronics. Unlike generic marketplaces, the platform focuses on SFSU, eliminating irrelevant listings and fostering a tight-knit community. With an intuitive interface, users can easily create listings, filter results, and receive notifications for items matching their interests, providing a seamless experience for all.

Behind this initiative is a team of passionate SFSU students, each bringing expertise in software development, UX design, and business strategy. Being students ourselves, we recognize the daily hurdles involved in campus transactions, and we are committed to creating a marketplace that's built by students, for students. Our varied backgrounds enable us to infuse innovative ideas into the project and continually refine the platform to adapt to the ever-changing needs of our peers. With the right backing, "Gator Market" has the potential to become the go-to hub for SFSU transactions, transforming how students buy and sell while promoting a safer, smarter, and more budget-friendly campus economy.

List of Main Data Items and Entities

1. User

Description: Represents a person interacting with the platform, which can be classified into different user types:

- **Registered User:** A student who has signed up and can buy, sell, and interact with other users.
- **Admin User:** A platform moderator with additional privileges to manage content and enforce rules.
- **Anonymous User:** A visitor who can browse products but must register to purchase or interact with sellers.

Attributes:

- **User ID** (unique identifier)
- **Name**
- **Email** (used for login and notifications)
- **Role** (e.g., buyer, seller, admin, anonymous user)
- **Registration date** (only applicable to registered users)
- **Profile picture** (optional)
- **Contact details** (only visible to admin or during transactions)

2. Product

Description: An item listed for sale by a registered user.

Attributes:

- **Product ID** (unique identifier)
- **Seller ID** (reference to **Registered User**)
- **Name** (title of the product)
- **Description** (detailed information about the product)
- **Category** (e.g., electronics, furniture, clothing, etc.)
- **Price**

- **Condition** (new, used, etc.)
- **Image(s)**
- **Date listed**
- **Status** (available, sold, reserved)

3. Category

Description: Represents the classification of products.

Attributes:

- **Category ID** (unique identifier)
- **Name** (e.g., electronics, furniture, clothing)
- **Parent category** (for subcategories)

4. Review

Description: A rating or feedback given by buyers and sellers after an interaction.

Attributes:

- **Review ID** (unique identifier)
- **Reviewer ID** (reference to **Registered User**)
- **Reviewee ID** (reference to **Registered User**)
- **Product ID** (optional reference)
- **Rating** (1-5 stars)
- **Comment**
- **Date of review**

5. Wishlist

Description: A collection of products a user is interested in purchasing later.

Attributes:

- **Wishlist ID** (unique identifier)

- **User ID** (reference to **Registered User**)
- **Product ID(s)** (reference to **Product**)
- **Date added**

6. Message

Description: A message exchanged between users for product inquiries.

Attributes:

- **Message ID** (unique identifier)
- **Sender ID** (reference to **Registered User**)
- **Receiver ID** (reference to **Registered User**)
- **Message content**
- **Timestamp**
- **Status** (read, unread)

7. Admin Actions

Description: Actions taken by an **Admin User** to manage platform content.

Attributes:

- **Admin ID** (reference to **Admin User**)
- **Action type** (delete user, block product, etc.)
- **Target entity** (User, Product, etc.)
- **Action description**
- **Timestamp**

8. Listing Report

Description: A report submitted by a user regarding a suspicious, inappropriate, or fraudulent listing.

Attributes:

- **Report ID** (unique identifier)
- **Reporter ID** (reference to **Registered User**)
- **Product ID** (reference to **Product**)
- **Reason** (e.g., scam, misleading info, inappropriate content)
- **Additional comments**
- **Date reported**
- **Status** (pending, reviewed, resolved)

Data Glossary (Google Analytics data not included)

ACCOUNT:

Field Name	Description
user_id	Unique identifier for each user, auto-incremented, serves as the primary key.
username	A unique username chosen by the user for identification and login purposes.
password_hash	The user's password, stored as a hashed value for security (e.g., using bcrypt).
first_name	The user's first name, for personalization and identification.
last_name	The user's last name, for personalization and identification.
email	The user's university email (e.g., ending in @sfsu .edu), unique, used for login and verification.

Field Name	Description
verification_status	Indicates if the user is a verified current SF State student (‘permanent’, 'annual', 'not verified').
phone_number	An optional contact phone number for verification purposes. (can be NULL)
profile_picture_url	An optional URL to the user's profile picture (can be NULL).
date_joined	Timestamp recording when the user created their account, defaults to current time.
last_login	Timestamp of the user's most recent login, for tracking activity (can be NULL).
user_role	The user's role on the platform ('user', 'moderator', 'admin'), defaults to 'user'.
account_status	The current status of the user's account ('active', 'inactive/banned', 'deleted'), defaults to 'active'.

POST:

Field Name	Description
post_id	Unique identifier for each post, auto-incremented, serves as the primary key.
user_id	Foreign key referencing the Users table, indicating the user who created the post.
title	The title of the post, limited to 100 characters for brevity and clarity.
description	A detailed description of the item being sold or sought, allowing flexibility in length.

category_id	Foreign key referencing the Categories table, specifying the item's category.
price	The price of the item, with two decimal places (e.g., supports up to 99999999.99).
condition	The condition of the item, restricted to predefined options for consistency.
location	Uses geocode to work with google maps API.
created_at	Timestamp of when the post was created, defaults to the current time for sorting purposes.
status	The current status of the post, managing its lifecycle (defaults to 'active').
approval_status	Indicates if the post has been reviewed by moderators (defaults to 'pending').

Functional Requirements - Prioritized

Priority 1 - Must Do (P1)

Unregistered User

- Unregistered users shall be able to view item listings.
- Unregistered users shall be able to use search and filtering functionalities (category, keyword, price, condition).

Registered User

- Registered users shall be able to register using an SFSU email (must be verified).
- Registered users shall be able to log in, log out, and manage account settings.
- Registered users shall be able to create, edit, and delete product listings.
- Registered users shall be able to upload one or more product images.
- Registered users shall be able to message other users securely within the app.
- Registered users shall be able to view and respond to messages.
- Registered users shall receive email or in-app notifications for new listings based on their interests.
- Registered users shall be able to report listings or users for inappropriate content.
- Registered users shall be able to rate and review buyers/sellers after a transaction.

Priority 2 - Good to Have (P2)

Registered User

- Registered users shall be able to enable group buying features for similar items (e.g., textbooks).
- Registered users shall be able to add multiple images per product listing.
- Registered users shall be able to filter search results by campus building or nearby location.
- Registered users shall be able to archive old listings for personal sales history.

Admin

- Admins shall be able to approve or reject listings before they are published.
- Admins shall be able to review and act on user reports.
- Admins shall be able to manage user accounts (ban/unban/delete).

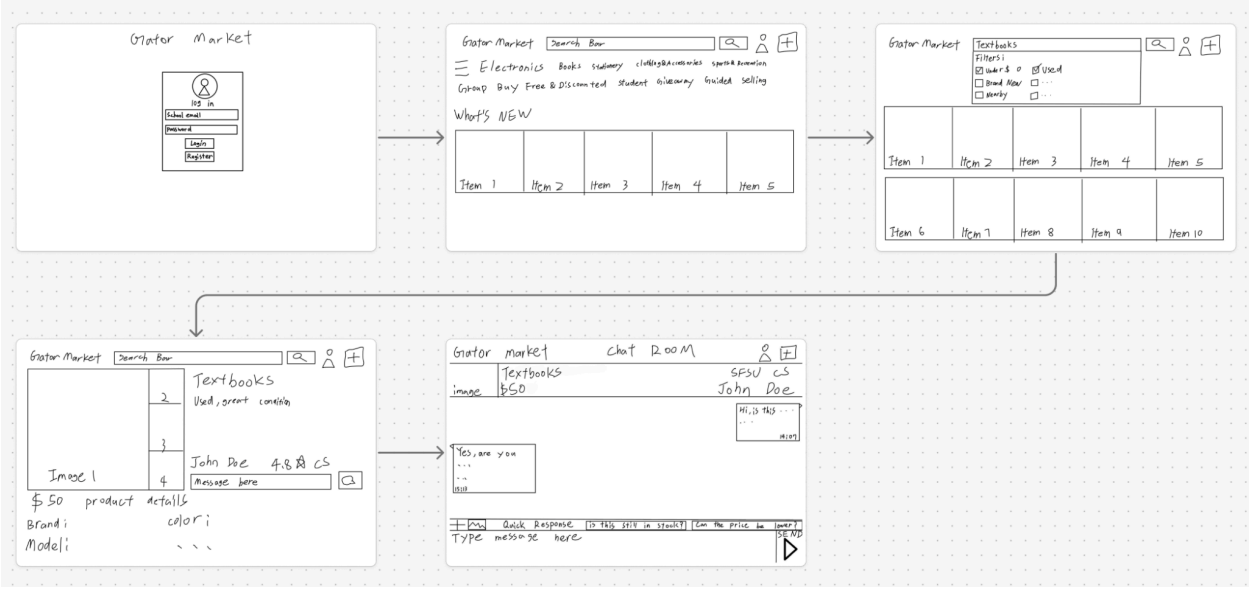
- Admins shall be able to access audit logs for moderation transparency.

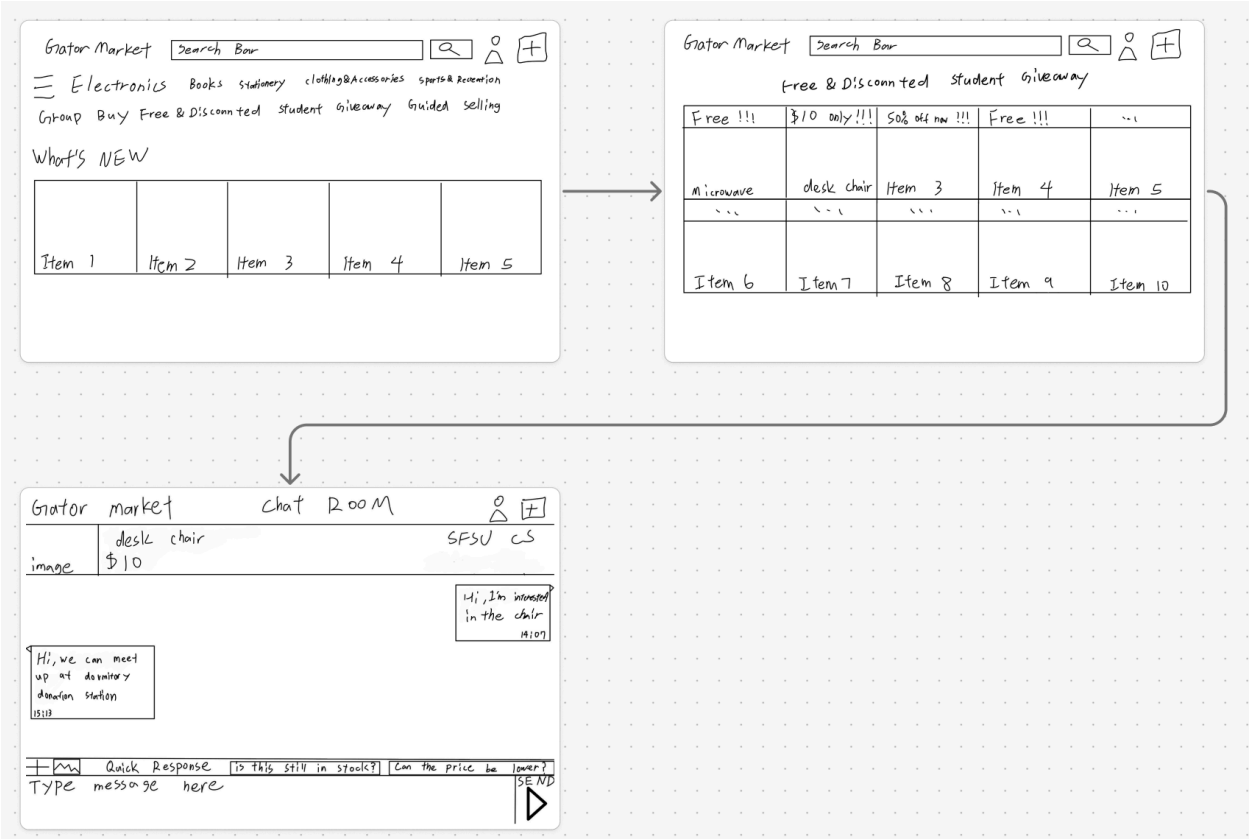
Priority 3 - Future Development (P3)

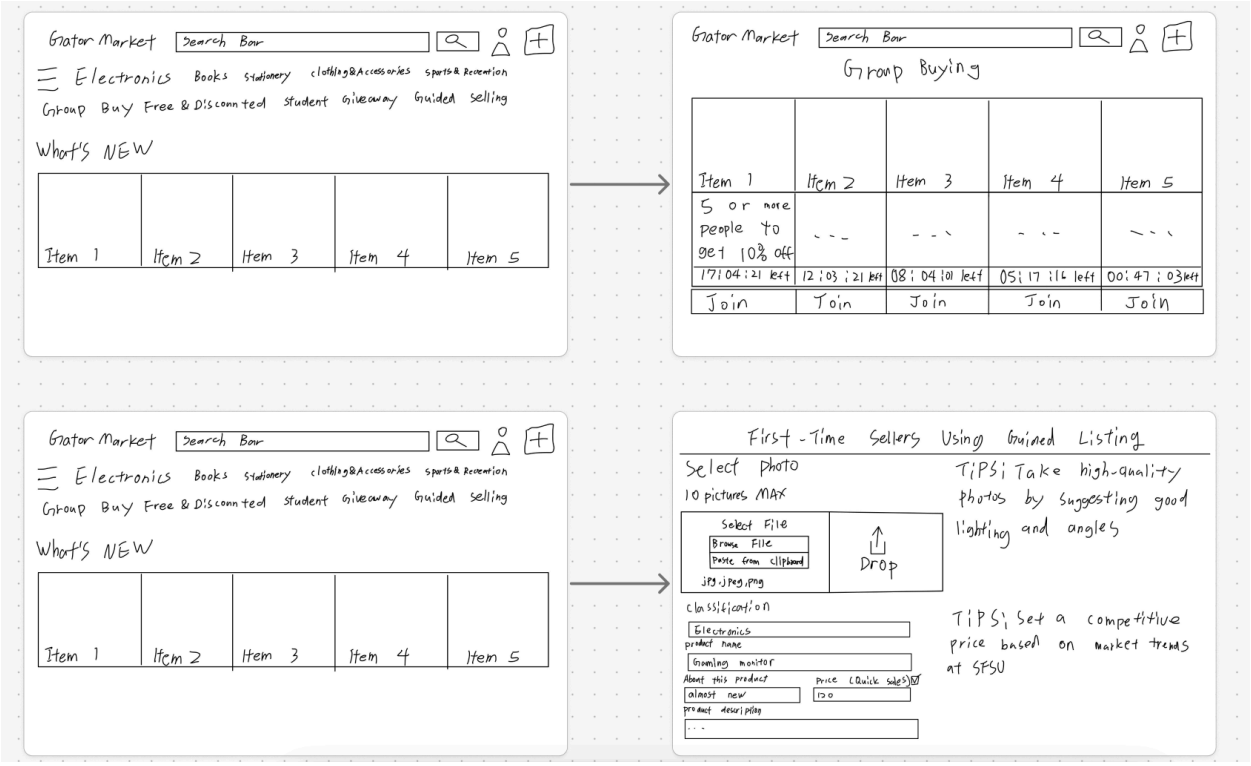
Registered User

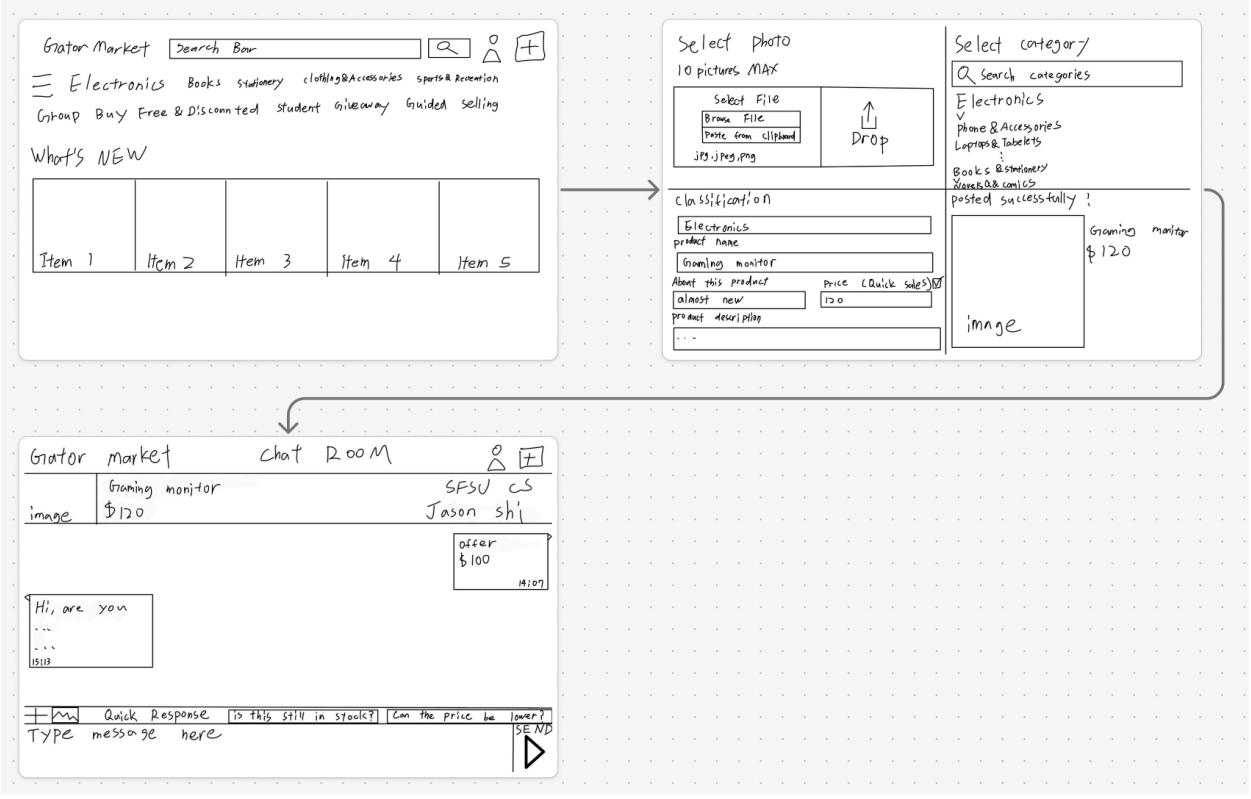
- Registered users shall be able to schedule meet-up time and location with sellers within the chat interface.
- Registered users shall be able to flag a product as "Urgent Sale" or "Free Giveaway."
- Registered users shall be able to bookmark sellers or listings for quick access.
- Registered users shall be able to enable a toggle for "Hide Sold Items" in the product feed.

UI Storyboards for Main Use Cases






Grator Market
Search Bar
Electronics Books Stationery clothing&Accessories sports& Recreation
Group Buy Free & Discounted Student Giveaway Guided selling
What's NEW
Item 1 Item 2 Item 3 Item 4 Item 5
Group Buying
5 or more people to get 10% off
17:04:21 left
12:03:21 left
08:04:01 left
05:17:16 left
00:47:03 left
Join
First-Time Sellers Using Guided Listing
Select photo
10 pictures MAX
Select File
Browse File
Paste from clipboard
jpg.jpeg.png
Drop
Tips: Take high-quality photos by suggesting good lighting and angles
Tips: Set a competitive price based on market trends at SFSU
classification
Electronics
product name
Gaming monitor
About this product
almost new
Price (Quick sales)
120
product description


Grator Market
Search Bar
Electronics Books Stationery clothing&Accessories sports& Recreation
Group Buy Free & Discounted Student Giveaway Guided selling
What's NEW
Item 1 Item 2 Item 3 Item 4 Item 5
Select photo
10 pictures MAX
Select File
Browse File
Paste from clipboard
jpg.jpeg.png
Drop
Select category
Search categories
Electronics
Phone & Accessories
Laptops & Tablets
Books & Stationery
Games & comics
classification
Electronics
product name
Gaming monitor
About this product
almost new
Price (Quick sales)
120
product description
posted successfully!
Gaming monitor
$120
image
Grator market
Chat ROOM
image
Gaming monitor
$120
SFSU CS
Jason shi
offer
$100
14:07
Hi, are you
15:13
Quick Response
Is this still in stock?
Can the price be lower?
Type message here
SEND

High-Level Architecture and Database Organization

The following provides an overview of the system components and how they interact with each other, including frontend, backend, database, and media storage.

Interaction Flow

1. The user shall interact with the frontend.
2. The frontend shall send API requests to the backend.
3. The backend shall process data and fetch/save data to the MySQL database.
4. If an image is uploaded, the backend shall generate an S3 pre-signed URL for the image upload.
5. The backend shall store the image URL in the MySQL database

Content for High-Level Architecture

1. Architectural Overview

The Product Catalog System shall follow a Model-View-Controller (MVC) architecture to ensure modularity and maintainability. The system shall consist of:

- A frontend UI for users to browse and search products.
- A backend server handling business logic and database operations.
- A relational database storing product, user, and message information.
- Media storage for product images.

This architecture shall allow seamless interaction between users and the product catalog while maintaining data integrity and security.

2. System Components

Frontend (Client-Side)

- Implemented using React and Tailwind CSS.
- Shall provide product listings, search functionality, and UI components for messaging and account management.

Backend (Server-Side)

- Built with Python and Flask.
- Shall handle API requests, process business logic, manage user accounts, and interact with the database.

Database (Data Storage)

- MySQL shall be used for structured data storage.
- Tables shall include:
    - **Users Table:** Stores user account information.
    - **Product Listings Table:** Stores product details.
    - **Categories Table:** Names of the categories of products sold.
    - **Messages Table:** Stores user-to-user messages.

Media Storage

- Product images shall be stored in a cloud-based storage like AWS S3.
- Backend shall generate pre-signed URLs to allow secure image uploads.
- Ensure no payments are required for S3 usage.

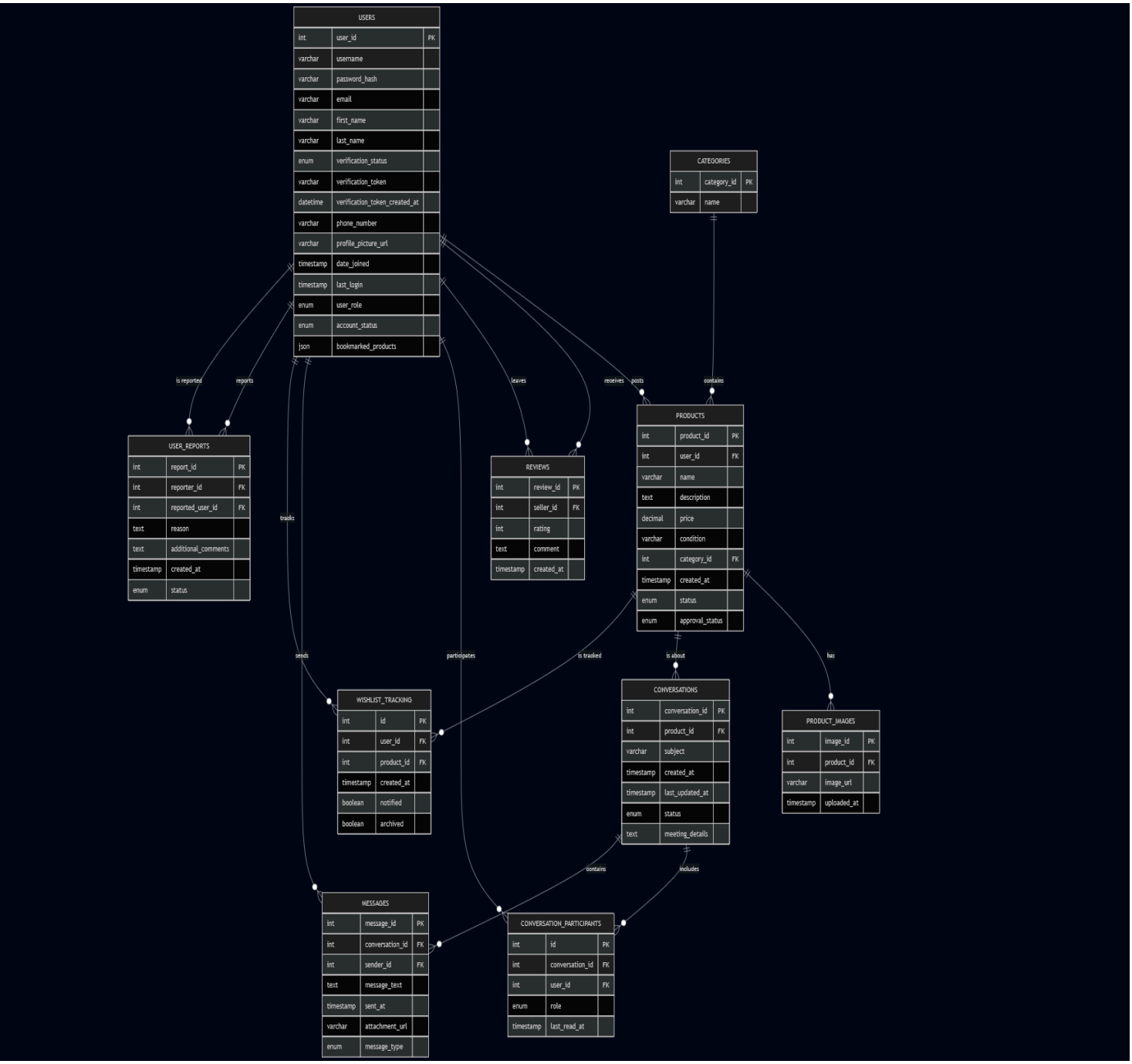
Search and Filtering

- SQL-based **%LIKE** queries shall be used for search.
- Filtering options shall be based on:
    - Categories
    - Price ranges
    - Product condition
    - Keywords

3. Technology Stack

Component	Technology
Frontend	React + Tailwind CSS
Backend	Python + Flask
Database	MySQL
Media Storage	AWS S3

Database Organization (Schema Overview)



The Gator Market database is designed around a normalized, scalable schema optimized for student-to-student transactions. The schema includes key tables such as users, products, categories, messages, and several supporting tables to enable features like image uploads, reviews, wishlist tracking, conversations, and reporting.

In the current design, all product listings are stored in a single unified products table, rather than using separate tables for each category (e.g., books, electronics, clothes). This avoids redundancy, simplifies cross-category querying, and allows the database to scale efficiently with new categories in the future. Each product includes general attributes (e.g., name, price, description, condition) and is linked to a category_id and a user_id for ownership and classification.

User messages are organized using a robust messaging system consisting of conversations, conversation_participants, and messages tables, supporting multi-user threads and message metadata.

Product images are stored in the product_images table, allowing one-to-many relationships between products and images, and supporting S3-based secure media storage via URLs.

The reviews table enables seller ratings and feedback after transactions, and wishlist_tracking allows users to bookmark items. Suspicious behavior can be flagged through the user_reports table, which stores reports made by users against others.

Products Table

Field	Type	Description
product_id	INT, PK, Auto Increment	Unique product ID

user_id	INT, FK to users	Seller (User)
name	VARCHAR(100)	Product name
description	TEXT	Product description
price	DECIMAL(10,2)	Product price
condition	ENUM('new', 'used', 'refurbished')	Item condition
image_url	VARCHAR(255)	Link to product image
category_id	INT, FK to categories	Product category
created_at	TIMESTAMP	Listing date
status	ENUM('active', 'sold', 'deleted')	Product listing status
approval_status	ENUM('pending', 'approved', 'rejected')	Moderation status

Categories Table

Field	Type	Description

category_id	INT, PK, Auto Increment	Unique category ID
name	VARCHAR(100)	Category name (e.g., electronics, books)

Users Table

Field	Type	Description
user_id	INT, PK, Auto Increment	Unique user ID
username	VARCHAR(50)	Unique username for login
password_hash	VARCHAR(255)	Hashed password
email	VARCHAR(100)	Unique SFSU email
first_name	VARCHAR(50)	First name
last_name	VARCHAR(50)	Last name
verification_status	ENUM('permanent', 'annual', 'not verified')	SFSU verification type
phone_number	VARCHAR(15)	Optional contact number

profile_picture_url	VARCHAR(255)	Optional profile picture URL
date_joined	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Registration date
last_login	TIMESTAMP NULL	Most recent login timestamp
user_role	ENUM('user', 'moderator', 'admin') DEFAULT 'user'	User privilege level
account_status	ENUM('active', 'inactive/banned', 'deleted') DEFAULT 'active'	Account state

Messages Table

Field	Type	Description
message_id	INT, PK, Auto Increment	Unique message ID
sender_id	INT, FK to users	Sender's user ID
receiver_id	INT, FK to users	Receiver's user ID
product_id	INT, FK to products	Associated product (optional)

message_text	TEXT	Message content
sent_at	TIMESTAMP	Timestamp when the message was sent

Conversations

Field	Type	Description
conversation_id	INT, PK, Auto Increment	Unique conversation ID
product_id	INT, FK to products	ID of product that the conversation is about
subject	VARCHAR	Conversation subect
created_at	TIMESTAMP	When conversation started
last_updated_at	TIMESTAMP	Last time the conversation was active
status	ENUM	Whether conversation is ongoing or not
meeting_details	TEXT	First few lines of most recent message

Conversation Participants

Field	Type	Description
id	INT, PK, Auto Increment	Unique ID
conversation_id	INT, FK to conversations	ID of conversation
role	ENUM	Role of the user in the conversation
last_read_at	TIMESTAMP	Last time the user’s message was read

Reviews

Field	Type	Description
review_id	INT, PK, Auto Increment	Unique review ID
seller_id	INT, FK to users	Seller's user ID
rating	INT	Rating out of 5
comment	TEXT	Comment written in review

created_at	TIMESTAMP	When review was left

Product_Images

Field	Type	Description
image_id	INT, PK, Auto Increment	Unique image ID
product_id	INT, FK to product	ID of the product
image_url	VARCHAR	URL of the uploaded image
uploaded_at	TIMESTAMP	When the image was uploaded

Wishlist_Tracking

Field	Type	Description
id	INT, PK, Auto Increment	Unique ID
user_id	INT, FK to users	ID of the user
product_id	INT, FK to products	Associated product

created_at	TIMESTAMP	When item was saved
notified	boolean	Whether a notification for this product was sent
archived	boolean	Whether the item has been archived

User_Reports

Field	Type	Description
report_id	INT, PK, Auto Increment	Unique report ID
reporter_id	INT, FK to users	ID of the user that is filing the report
reported_user_id	INT, FK to users	ID of the user that is being reported
reason	TEXT	Reason for report
additional_comments	TEXT	Any additional comments
created_at	TIMESTAMP	When the report was filed

status	ENUM	Status of report (pending, reviewed, resolved)

Relationships

One-to-One:

- Each user has a unique verification status and profile.

One-to-Many:

- One user can post multiple products
- One product can have multiple images
- One user can receive multiple reviews as a seller
- One category can contain multiple products
- One conversation can contain multiple messages
- One conversation can involve multiple participants

Many-to-Many:

- Many users participate in many conversations
- Many users can wishlist many products
- Many users can have many reports

5. Data Flow and Interaction

- User Request: User accesses the platform to browse or post products.
- API Call to Backend: The frontend sends API requests to fetch, post, or update product information.
- Backend Processes Request: The backend handles business logic and interacts with the database.
- Database Query Execution: The database returns product, user, or message data as requested.
- Response to Frontend: The backend returns the data as a JSON response.
- Frontend Displays Data: The frontend dynamically displays listings, messages, or user information.

6. Deployment Strategy

- Frontend: Hosted on AWS S3/CloudFront.

- Backend: Deployed on AWS EC2.
- Database: Hosted on MySQL service (AWS RDS or equivalent).
- Media Storage: Product images stored in localStorage
  → Magic and mimetype security ensure what type of file is being uploaded (JPEG or PNG) and nginx 404 returns to keep them secure and prevent harmful code injection

[User] → [React Frontend] → [Node.js Backend]

↓

[localStorage - Store Images]

↓

[MySQL - Save image URL in products table]

Product images uploaded through the frontend are now stored locally using the browser's localStorage API. When a user uploads an image, it is converted to a base64-encoded string and saved in the user's local browser storage. The encoded image string is then submitted along with the product form and stored in the backend's product_images table. This approach simplifies development by removing the need for external storage services and allows image data to be displayed immediately without requiring asynchronous fetching from a remote server.

Identification of Key Risks

1. Skills Risk

Issue: The team consists of members with different skill sets and levels. This may include the technical aspects, such as the knowledge of the technology stack used for the project, as well as soft skills such as communication and coordination.

Solution: Conducted weekly knowledge-sharing sessions and assigned small tasks for skill-building.

2. Schedule Risk

Issue: Overlapping deadlines with other coursework.

Solution: Set internal milestone deadlines 3-4 days ahead of official submission dates.

Issue: The team has scheduled team meetings during and between class times. It may still be insufficient to establish a robust understanding for all team members. Even then, it is not guaranteed that all members can attend all meetings, as everyone has their own life.

Solution: The team can reduce it by prioritizing necessary tasks, planning ahead for contingencies, creating checkpoints to assess progression, and dynamically assigning members based on the current state.

3. Technical Risk

Issue: Deployment issues on AWS due to limited experience with Docker.

Solution: Backend team will set up a staging environment for early testing.

4. Teamwork Risk

Issue: Coordination challenges between frontend and backend teams.

Solution: Implement weekly stand-ups with clear task assignments. The team can organize the individual tasks chronologically to reduce dependency and distribute them according to the team member's skill and workload level.

5. Legal/Content Risk

Issue: Ensuring product images are owned by users or use copyright free images.

Issue: The team must be aware of copyright infringement, licensing requirements, intellectual property, privacy, data protection, or any other regulations.

Issue: The team shall ensure that the project does not contain harmful or inappropriate content within its scope of usage.

Solution: The team can ensure the project complies with laws and regulations, implements necessary standards, and reduces liability.

Project Management Strategy

1. **Task Management:**

- Trello is used as our central project management board. Each task card includes:
    - Clear descriptions and subtasks
    - Assigned members
    - Due dates
    - Labels for task categories (frontend, backend, documentation, etc.)
- Tasks are tracked through "To Do," "In Progress," and "Completed" columns.
- The Scrum Master and Team Lead collaboratively update and review progress on Trello before every meeting.
- We have separate boards for Milestones, Bug Tracking, and Feature Requests to avoid clutter and improve focus.

2. Communication & Collaboration:

- Daily communication is maintained via Discord, where we maintain organized channels for specific workstreams like #frontend, #backend, #m1, #m2-part1 and #m2-part2.
- Google Docs is used for collaborative drafting and notes during team sessions.

3. Workflow and Meeting Cadence:

- Weekly sprints run from Wednesday to the following Tuesday. Sprint planning is conducted during Wednesday meetings.
- On Wednesdays, tasks are assigned or reprioritized based on sprint progress.
- By Friday, team members are expected to reach mid-sprint deliverables, verified via Trello status updates.
- Weekend work is self-paced and documented on Trello.
- Monday/Tuesday is reserved for internal testing, documentation, and refinement.

4. Documentation & Version Control:

- Google Docs is used for all milestone planning, ensuring centralized collaborative editing.
- GitHub is our version control hub. Team members create branches per feature/task, and Pull Requests are reviewed by Tech Lead or assigned reviewer.

5. Milestone Tracking and Quality Assurance:

- Each milestone is broken into mini-goals, with Trello labels for better tracking.

- The Scrum Master checks on the status of major goals twice per sprint.
- Before each milestone deadline, the team conducts a QA pass to ensure that documentation, database, and frontend/backend functionality are in sync.

6. Conflict Resolution & Syncing Teams:

- When conflicts or confusion arise, a dedicated zoom call room is used for screen-sharing and real-time collaboration.
- If needed, temporary working groups are formed to resolve integration or logic issues.

7. Learning & Upskilling Support:

- We regularly encourage team members to explore new tools (e.g., Docker, Tailwind) with small guided tasks.
- Pair programming is used for difficult or unfamiliar components.
- Team members share useful resources and tutorials on Discord.

To sum it up, our team used Trello to manage tasks. Everyone was previously split into front-end and back-end groups based on their familiarity with the respective technologies. This made task assignment easier to break up for Trello. Every week after every meeting respective tasks would be assigned on Trello and everyone would be given proper deadlines. Every meeting (Wednesday and Friday) we set a goal to get work done by the weekend. On Friday, we check with each other (previously on github) whether our tasks have been completed. Now we can track them in Trello. On Wednesdays, we talk about how our sprint is going and make sure we are all on track. We are in constant communication on Discord to resolve any issues between sections of work. If there are any conflicts, the concerned members will join a call room and talk about their issues as well as sharing screens to better visualize and explain the issues.

Use of GenAI Tools in Milestone 2

GenAI Tools Used

For Milestone 2, our team utilized various Generative AI tools to streamline different aspects of the project. The primary tool used was:

ChatGPT-4.0 – Used for brainstorming, refining Database Organisation, structuring database schema, Risk Identification planning, and code prototyping.

Tasks Where ChatGPT Was Used and Effectiveness Ratings

Task	Rating	Description of Use
Expanding Functional Priority Requirements	HIGH	AI provided structured requirements with user role breakdowns. We refined them for accuracy.
Database Schema Optimization	HIGH	Suggested efficient table structures and foreign key relationships, which we validated against project needs.
UI Ideation	MEDIUM	AI-generated suggestions, but made changes to customize and fit project requirements.
Risk Identification & Mitigation	MEDIUM	Provided general risk factors, which were adapted for our specific challenges.
Code Boilerplate for API Endpoints	HIGH	Generated Flask API templates, reducing development time.
Search & Filter Logic Recommendations	MEDIUM	Suggested SQL %LIKE% queries and indexing strategies, which we refined.

Key Examples and Prompts Used

Functional Priority Detailing Prompt

Prompt: "List detailed functional Priority requirements for a university marketplace, categorized by user roles. Include key actions for registered users, admins, and guest users."

AI Response(Edited):

- **Registered Users:** Can create/edit/delete listings, contact sellers via in-app messaging, verify SFSU email, receive notifications for matching listings.
- **Admins:** Can moderate content, approve/reject listings, manage user roles and permissions.
- **Guest Users:** Can browse listings but cannot contact sellers without registration.

Database Schema Prompt

Prompt: "Suggest a relational database schema for an online student marketplace with users, product listings, messaging, and reviews. Include foreign key relationships."

AI Response (Edited):

- **Tables Suggested:** Users, Products, Messages, Reviews, Categories, Admin_Actions.
- **Relationships:** One-to-many (Users → Products), Many-to-Many (Users ↔ Messages).

Search & Filter Mechanism Prompt

Prompt:"Suggest an efficient search and filtering strategy for a student marketplace, considering SQL and indexing best practices."

AI Response(Edited):

- Use SQL %LIKE% queries for flexible keyword searching.
- Implement database indexing on frequently searched fields (product name, category).
- Allow multi-filtering (e.g., price range, condition, category).

Final Thoughts on AI Use

Overall, GenAI tools were highly beneficial in structuring content, brainstorming ideas, and speeding up certain development tasks. However, human oversight was necessary to:

- Refine functional requirements for precision.

- Adjust database schema based on actual project constraints.
- Customize UI layouts to align with our user experience goals.
- Validate search and filtering logic for efficiency.

For future milestones, we plan to leverage AI more for:

- Code debugging & optimization
- Further UI/UX enhancements

Team Lead Checklist

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 3 – Feedback Summary

Feedback Item	Instructor Feedback	Our Response / Action Taken
1. Executive Summary layout	**Too much white space at the top of the page; executive summary should fit on one page.**	**We removed unnecessary spacing above the title so the summary fits on a single page.**
2. Data Glossary	**Data type column is only needed for DB design section, not glossary.**	**We moved the data types into the DB schema section and simplified the glossary to only include field names and descriptions.**
3. Functional Requirements phrasing	**“Shall” should be used for formal requirement language.**	**We revised the P1 and P2 function lists to include “shall” for each required behavior (e.g., “Users shall be able to...”).**
4. Admin features	**Certain admin functions like banning users can be done via Workbench and may be de-prioritized.**	**We moved less critical admin actions (e.g., user bans) to P2 or removed them, based on feedback.**
5. Wishlist feature caution	**Wishlists may not be practical for low-quantity items.**	**We acknowledged this and kept wishlist functionality minimal; focus shifted to essential transaction features.**
6. UI Storyboards	**Hard to follow; unclear use cases. All storyboards should start from the homepage.**	**We reorganized our UI storyboards so each use case begins from the homepage, with clearly marked flow arrows and labels.**
7. Image hosting	**Ensure no payment is required for using S3.**	**We verified our usage of AWS S3 pre-signed URLs stays within the free tier and added this note to our documentation.**

8. Table and diagram mismatch	**Data table descriptions did not match schema diagram fields.**	**We updated our ER diagram and glossary table to match one another exactly.**
9. Messages schema missing	**Diagram missing table or explanation for buyer-seller messaging.**	**We added a messages table to the schema and updated documentation to reflect how messaging is stored.**

Milestone 4

SW Engineering CSC648-848 Spring 2025

Project/application title and name: Gator Market

Section 04 Team 1 Milestone 4

Name	Role	Email
Dev Modi	Team Lead, Backend, Frontend, database	dmodi@sfsu.edu
Yash Pachori	Backend Lead with Database	ypachori@sfsu.edu
Kyle Yuen	Frontend Lead	kyuen4@sfsu.edu
Hsueh-Ta Lu	Scrum Master with Frontend	hlu@sfsu.edu
Daniel	Tech Lead and Github	domstead@sfsu.edu

Revision History

Date	**Notes**
30 March 2025	Initial publication of M4 document

Topic Index

1) Product Summary

Product Name: **Gator Market**

Product Description:

Gator Market is a secure, student-exclusive online marketplace built specifically for San Francisco State University (SFSU). Its primary purpose is to make buying and selling items—like textbooks, dorm supplies, and electronics—easy, safe, and trustworthy within the SFSU community. By requiring users to sign up with verified @sfsu.edu emails and including built-in chat messaging, the platform eliminates risks found on platforms like Craigslist or Facebook Marketplace. Unique to Gator Market are AI-powered pricing suggestions and group-buying tools, making it smarter and more budget-friendly than traditional marketplaces. The platform is student-built and designed to solve real problems that students face daily, turning campus transactions into a safer, smarter, and community-driven experience.

Final List of Committed (P1) Functionalities:

These are the features our team commits to deliver and test fully. Each line is a complete, plain-English function that will be available and functional at launch.

For Unregistered Users:

- Can view all item listings.
- Can use search and filter by category, keyword, price, and condition.

For Registered Users:

- Can register using a verified SFSU email address.
- Can log in, log out, and manage account settings.
- Can create, edit, and delete their product listings.

- Can upload one or more images when posting products.
- Can send and receive secure messages to/from other users within the app.
- Can view messages and respond to them.
- Can receive email or in-app alerts when new listings match their interests.
- Can report suspicious or inappropriate listings or users.
- Can leave ratings and written reviews for other users after a transaction.

Deployment URL:

http://csc648g1.me

2) Usability test plan for selected function

1. Test Objectives

Goal: Ensure users can easily upload a product, fill in necessary details (like product name, price, description), upload an image, and use the safety feature (allowing photo/video recording during meetups).

- Test if users can complete the upload process without issues.
- Check if the safety option (photo/video recording for safety) is clear and understandable.

---

2. Test Background and Setup

System Setup:

- Testing will happen on desktop browsers, using our live production server.

Testers:

- Regular users with no technical background.
- They should be able to fill in a form and upload a product.

Testing Environment:

- Tests will be done from home or office with a stable internet connection and desktop or laptop devices.

3. Usability Task Description

Tasks:

1. Go to the product upload page.
2. Fill in these fields:
    - Product name
    - Description
    - Price
    - Category
    - Condition
3. Upload a product image (max 5MB).
4. Decide if you'd like to allow photo/video recording for safety during meetups (optional).
5. Submit the form and check for a success message.

---

4. Effectiveness and Efficiency Evaluation

Effectiveness:

- Can users complete the product upload without help?
- Do they understand the safety option (photo/video recording)?
- Is the image upload feature working smoothly?

Efficiency:

- How long does it take to complete the upload task?
- Are there any issues or confusion while filling out the form?
- Do users easily understand the safety feature?

---

5. User Satisfaction

After the task, ask users to rate the following on a Likert scale (1 = Very Unsatisfied, 5 = Very Satisfied):

1. How easy was the product upload process?
2. How satisfied are you with the form design and layout?
3. How did you feel about the image upload feature (e.g., preview)?
4. How clear were the error messages (if any)?
5. How comfortable are you with the option to allow photo/video recording during meetups for safety?

---

6. Use of GenAI

- Tool Used: We used ChatGPT to help generate and refine this test plan.
- How It Was Used: ChatGPT made sure the test objectives, tasks, and questions were clear, concise, and comprehensive.
- Feedback: It helped improve the flow of the tasks and clarified the safety feature's role in the test.

---

7. Conclusion

This plan ensures that the product upload feature is user-friendly, including the safety option for photo/video recording. We'll check:

- Whether users can upload a product with no issues.
- If they understand and use the safety feature.
- How satisfied they are with the process overall.

3) QA Test Plan and QA Testing Report: Gator Market - Homepage Search & Filter

1. Test Objectives

To verify that the search and category filter system on the homepage:

- Correctly fetches and displays products based on user input
- Maintains a history of recent searches
- Clears and updates product results dynamically
- Functions consistently across different browsers

2. Hardware and Software Setup

Hardware:

- PC/Laptop (Windows 11/macOS)

Software:

- Google Chrome

URL:

- VM Ware
- VS Code

3. Features to be tested:

- Product search bar functionality
- Category dropdown filter
- Search history display and clearing
- Fetching and rendering filtered product results

4. Q&A Test Plan

Test #	Test Title	Description	Input	Expected Output	Pass/Fail
1	Initial Product Display	The product should be listed	Page loads, no input needed	Products should be there	Pass
2	Keyword Search	Test product search using the keyword	Enter “laptop” or “computer ”	Takes you to your input	Pass
3	Category Filter	Filter products by category	Select “phones”	Should show all phones	Pass
4	Search History	Verifies local storage and saves the search history	Search phone, then reload the page	phone	Pass
5	Bookmark	Tests to see if we can bookmark an item	Click the bookmark emblem	The product appears in the bookmarks	Pass

5. Multi-Browser Testing Results

- Google Chrome (pass)
- Safari (pass)

6. GenAI use

GenAI tools, including ChatGPT, were actively used throughout the QA testing process to:

- Generate detailed and structured test case templates
- Refine test descriptions and expected outcomes for clarity and completeness
- Suggest additional edge cases for broader test coverage
- Assist in drafting automated test scripts for search and filter functionalities
- Provide real-time feedback on UI/UX elements based on usability principles

The use of GenAI significantly accelerated the documentation process and helped ensure consistency and thoroughness in testing.

4) Peer Code Review

1. **Peer Review #1: Yash (Backend Lead) Reviewed Dev's Frontend Code (AdminDashboard.jsx)**

Code Reviewed:

- File: AdminDashboard.jsx
- Feature: Admin interface for managing product approvals, users, and viewing reports

Review Request:

Dev shared the GitHub pull request feature/admin-dashboard-ui and requested a formal review from Yash for usability, API integration, and error handling feedback.

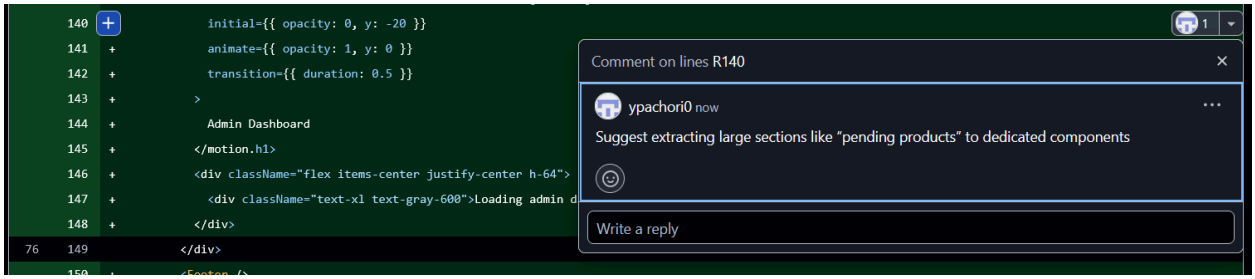
Review Comments Summary:

Area	Observation	Suggestion
Navigation & Access	Correctly restricts access to admins using localStorage role checks	Consider using context-based auth protection to centralize logic
API Integration	Endpoints for products, users, and stats are properly consumed using Axios	Ensure proper error boundaries in case any one of the three fails
UI/UX	Excellent use of Lucide icons and tab system for admin views	Suggest adding pagination for user list in "User Management" tab

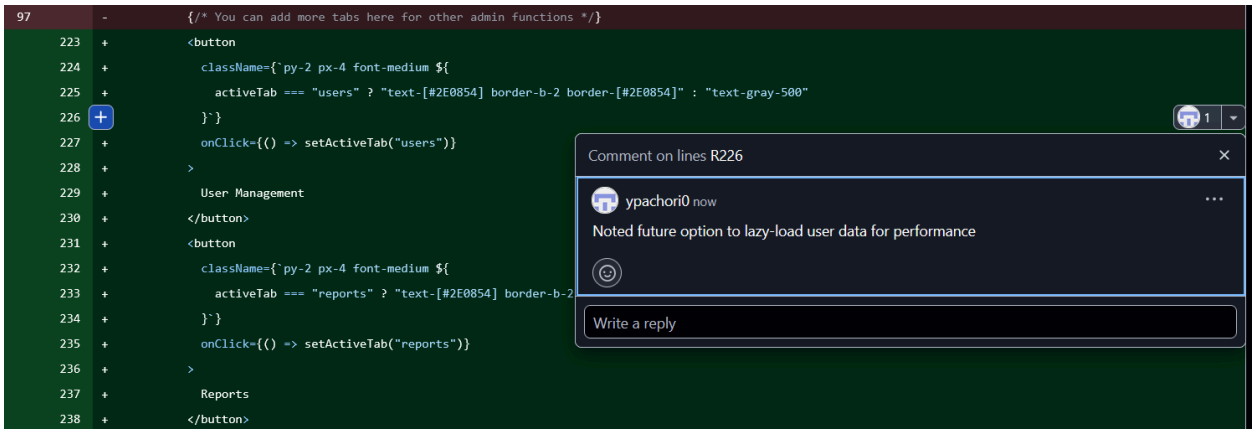
Code Structure	Large file with conditional render logic for each tab	Break down into subcomponents like <PendingProducts />, <UserTable /> etc.
State Management	useState and useEffect hooks used correctly for fetching and managing data	Good use of centralized fetchAdminData() function

GitHub Inline Review Highlights:

- Line 140: Suggest extracting large sections like “pending products” to dedicated components



- Line 226: Noted future option to lazy-load user data for performance



GenAI Review of Dev's Code

Prompt Used:

> "Review a React admin dashboard component that displays pending product approvals, user bans, and reports. It uses Axios, React hooks, and Tailwind. Suggest improvements to performance and structure."

Suggestions:

- Recommended splitting into smaller child components
- Suggested adding useCallback or debouncing for API refresh button
- Highlighted good use of loading indicators and toast notifications

Utility Rating: MEDIUM
Benefit: Reinforced modularity and performance optimizations

2. **Peer Review #2: Hsueh (Frontend) Reviewed Daniel's Backend Code (admin.py)**

Code Reviewed:

- File: admin.py
- Feature: Backend routes for admin functionality – moderation, reports, dashboard stats

Review Request:

Daniel shared the feature/admin-api-moderation branch and asked Hsueh to provide a frontend-consumption-focused review.

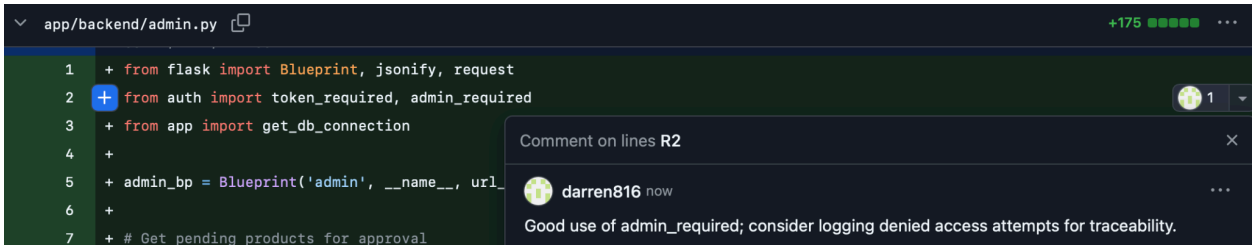
Review Comments Summary:

Area	Observation	Suggestion
Access Control	All routes properly protected with @admin_required	Very secure — just make sure decorators log denied access

API Design	Endpoint structure is RESTful and clearly named	Suggest clearer status codes in failure cases (e.g., 404 if report not found)
DB Usage	Queries are parameterized and avoid injection risks	Could use with conn.cursor() for better resource management
Reuse & Modularity	Repeated status validation in multiple functions	Suggested a helper function for status validation
Testability	Return data is JSON and frontend-compatible	Well-aligned with frontend needs for list display and error messages
API Usage	Limit is not capped	Add a reasonable upper limit (e.g., min(limit, 500))
Data Aggregation	Dashboard queries are clean and efficient	Good use of SQL aggregation to support frontend

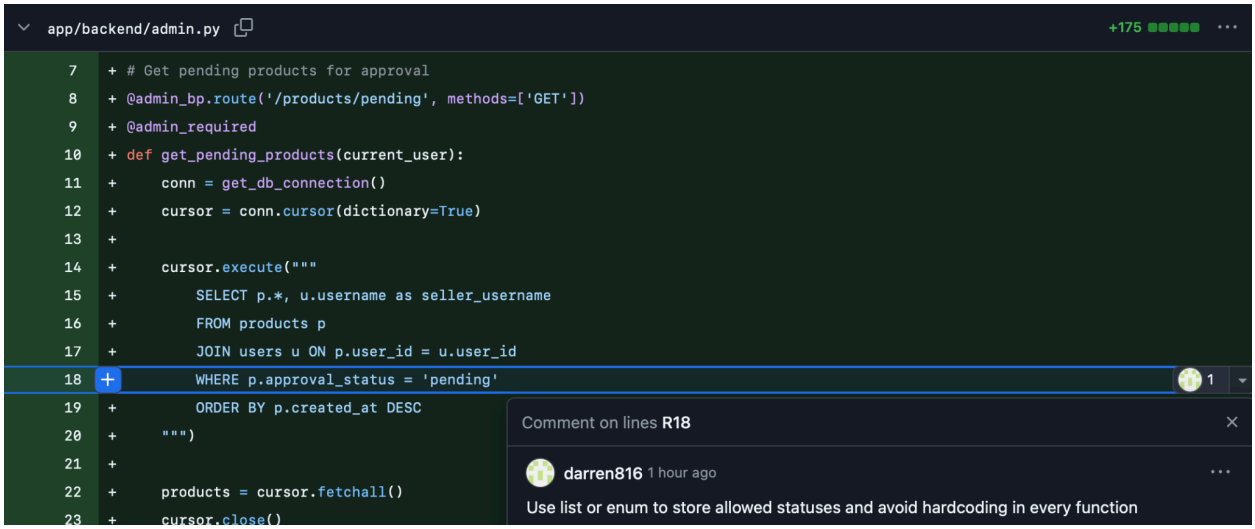
GitHub Inline Review Highlights:

- Line 2: Good use of admin_required; consider logging denied access attempts for traceability.

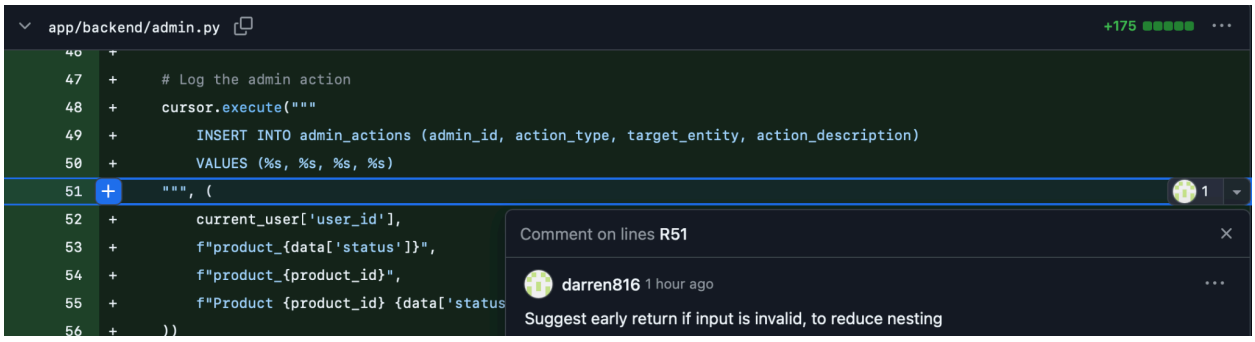


- Line 12: Consider using with conn.cursor() to ensure proper cleanup.

- Line 18: Use list or enum to store allowed statuses and avoid hardcoding in every function



- Line 51: Suggest early return if input is invalid, to reduce nesting



- Line 132: Consider adding a max cap (e.g., min(limit, 500)) to prevent abuse with very large query limits.

- Line 165: Nice breakdown of product stats here — it's clear and efficient. Good use of aggregation to support the dashboard.

GenAI Review of Daniel's Code

Prompt Used:

> "Review this Flask Blueprint backend code for an admin panel with endpoints to approve/reject products, view reports, and get dashboard stats. Suggest improvements in performance and structure."

Suggestions:

- Use context managers (with conn.cursor() as cursor) for DB operations
- Consolidate repeated status validation
- Consider separating out logging logic for admin actions

Utility Rating: HIGH
Benefit: Helped emphasize DRY principles and secure DB patterns

5) Security Self Check

This section summarizes our security strategy for protecting the key assets of **Gator Market**, covering both backend (Flask, MySQL) and frontend (React) implementations. We have followed OWASP top 10 recommendations, secure database design, and enforced strict input validation and role-based access control.

Asset to be Protected	Types of Possible/Expected Attacks	Consequence of Security Breach	Your Strategy to Mitigate/Protect the Asset
User credentials (email, password)	Brute-force login, credential stuffing, data breach	Account hijacking, identity theft	Passwords are hashed using bcrypt before storage. Login checks include password hash validation. No plaintext passwords are stored. JWT tokens used for session management.
SFSU email-only registration	Fake user creation using non-SFSU emails	Unauthorized access to student-only features	Regex-based validation (@sfsu.edu enforced) and domain matching on backend during registration (auth.py)

User input in forms	SQL injection, XSS (Cross-Site Scripting)	Data compromise, stored XSS attacks	Backend uses parameterized SQL queries with placeholders (e.g., %s). Client-side and backend validations restrict script inputs.
Product listings	Spam, inappropriate content	Loss of trust, abuse of platform	Admin approval is required before listings go public. Admins can review/reject content.
In-app messaging	Injection, phishing attempts	Abuse of communication channel, potential scams	Input sanitization in message content, limit on message size. No file/image sending permitted.
Bookmarked/wishlist products	JSON injection in MySQL JSON fields	Logic corruption or data overwrite	All JSON operations are server-controlled using JSON_APPEND, JSON_SEARCH. Inputs are strictly type-checked before database operations.
Listing reports / user reports	Report spam, bypass moderation	Admin overwhelmed with invalid reports	Only authenticated users can report. Cannot self-report. Backend enforces logic and logs all actions.

Admin endpoints	Unauthorized access, privilege escalation	Platform compromise	All admin routes protected by @admin_required decorator which checks role before access is granted.

Password Storage Strategy

- Passwords are hashed with bcrypt (12 rounds) before storing in the users table.
- We never store plaintext passwords.
- Password complexity is enforced via regex: minimum 8 characters, uppercase, lowercase, number, and symbol.

Input Validation (Implemented in auth.py and Frontend Forms)

Field	Validation	Regex/Check Used
Username	4–20 chars, alphanumeric and underscore	^[a-zA-Z0-9_]{4,20}$
Email	Must end in @sfsu.edu	^[a-zA-Z0-9._-]+@sfsu\.edu$

Password	Min 8 chars, uppercase, lowercase, digit, symbol	^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&]).{8,64}$
Search Input	Up to 40 alphanumeric characters	^[a-zA-Z0-9\s]{0,40}$
First/Last Name	2–30 alphabetic characters	^[a-zA-Z\s'-]{2,30}$
Terms Acceptance	Required checkbox on registration form	Checked before submission

Additional Protections

- **JWT-based Authentication**: Used for login and route protection. Tokens expire after 24 hours.
- **CORS Policy**: Strict CORS settings in app.py to allow only frontend domain (https://csc648g1.me).
- **Rate Limiting (Planned)**: Admin and login endpoints are planned to be protected using Flask-Limiter before final deployment.
- **S3 Upload Security**: All file uploads are routed through pre-signed URLs with limited access and expiration.
- **Role-Based Access**: Admin routes protected using admin_required() decorator, which wraps token_required() and checks user role.

6) Non-Functional Requirements Self-Check

#	Non-Functional Requirement	Status	Comments
1	Application shall be developed, tested and deployed using tools and cloud servers approved by Class CTO	DONE	AWS EC2, MySQL, and Flask used
2	Application shall be optimized for standard desktop/laptop browsers (latest 2 versions of 2 major browsers)	DONE	Chrome tested, Firefox tested
3	Application functions shall render well on mobile devices (no native app required)	DONE	Responsive layout implemented
4	Posting of sales info and messaging shall be limited only to SFSU students	DONE	Email must contain “@sfsu.edu”
5	Critical data shall be stored in the database on the team’s deployment server	DONE	Data stored in “gator_market” database
6	No more than 50 concurrent users shall be accessing the application at any time	DONE	To ensure compliance with the performance requirement of no more than 50 concurrent users (Requirement #6), we configured NGINX with a connection limit

			directive using the limit_conn module. This was added to the NGINX configuration file (nginx.conf) as follows: nginx CopyEdit limit_conn_zone $binary_remote_addr zone=addr:10m; limit_conn addr 50;
7	Privacy of users shall be protected	DONE	Passwords encrypted using bcrypt
8	The language used shall be English	DONE	Entire interface and content in English
9	Application shall be very easy to use and intuitive	DONE	Simple and clean UI with guided forms
10	Application shall follow established architecture patterns	DONE	MVC + RESTful API structure
11	Application code and its repository shall be easy to inspect and maintain	DONE	Codebase organized using clear folder

			structure, naming conventions, and documented routes; version control maintained via GitHub with meaningful commits.
12	Google Analytics shall be used	DONE	Google Analytics tracking code embedded in frontend for usage insights
13	No e-mail/chat services allowed; only internal messaging with single round allowed	DONE	In-app message system implemented
14	No pay functionality or payment simulation allowed in UI	DONE	No payment logic or UI placeholder present
15	Site security: basic best practices must be applied (password encryption, input validation, etc.)	DONE	Passwords encrypted using bcrypt, input validation implemented
16	Media formats must be current market standard (e.g., jpg, png)	DONE	jpg/png used for product images

17	Modern SE processes/tools shall be used, including collaborative and GenAI-assisted development	DONE	GitHub, Trello, ChatGPT used throughout
18	UI must display: "SFSU Software Engineering Project CSC 648-848, Spring 2025. For Demonstration Only" on every page header	ON TRACK	Planned in final UI build

7) Use of GenAI Tools Summary

Throughout the development of Milestone 4, our team extensively used **ChatGPT (GPT-4)** to enhance clarity, speed, and consistency across technical and design deliverables. This builds on our prior experiences in Milestone 1 and 2, where GenAI tools were used to support brainstorming, functional breakdowns, and database planning.

Tools Used

- ChatGPT GPT-4 (OpenAI)

Where We Used GenAI in Milestone 4

- **Usability Test Plan:** Helped draft clear test objectives, step-by-step task instructions, and Likert-scale questions.
- **QA Test Plan:** Improved test case structure and input/output descriptions; suggested broader test scenarios.
- **Code Review:** Used to analyze React component logic for clarity, naming conventions, and missing error handling.
- **Writing Assistance:** Ensured professional and consistent tone throughout the milestone document.

Key Example Prompts

- *"Help me write a usability test plan for a product upload feature that includes image upload and safety options."*
- *"Generate QA test cases for search and filter functions on an online marketplace."*
- *"Review this JavaScript code and suggest improvements in naming, security, and structure."*

Benefits Observed

- Accelerated writing and reduced rework cycles
- Improved clarity in test plans and documentation
- Provided second opinions on code structure and possible edge cases

Limitations

- Needed human validation for contextual accuracy
- Occasionally offered too generic suggestions for SFSU-specific constraints

GenAI Utility Ranking: HIGH

Overall, GenAI played a critical supporting role across planning, testing, and review tasks in M4. While it did not replace our work, it significantly enhanced the speed and polish of our output. We expect to continue using GenAI tools for Milestone 5 in areas such as user documentation, final QA refinement, and presentation design.

3) Product Screenshots

Figure 1: Homepage – Search and Category Filters

Displays the landing page with keyword search and category filter dropdowns, enabling users to begin item discovery.

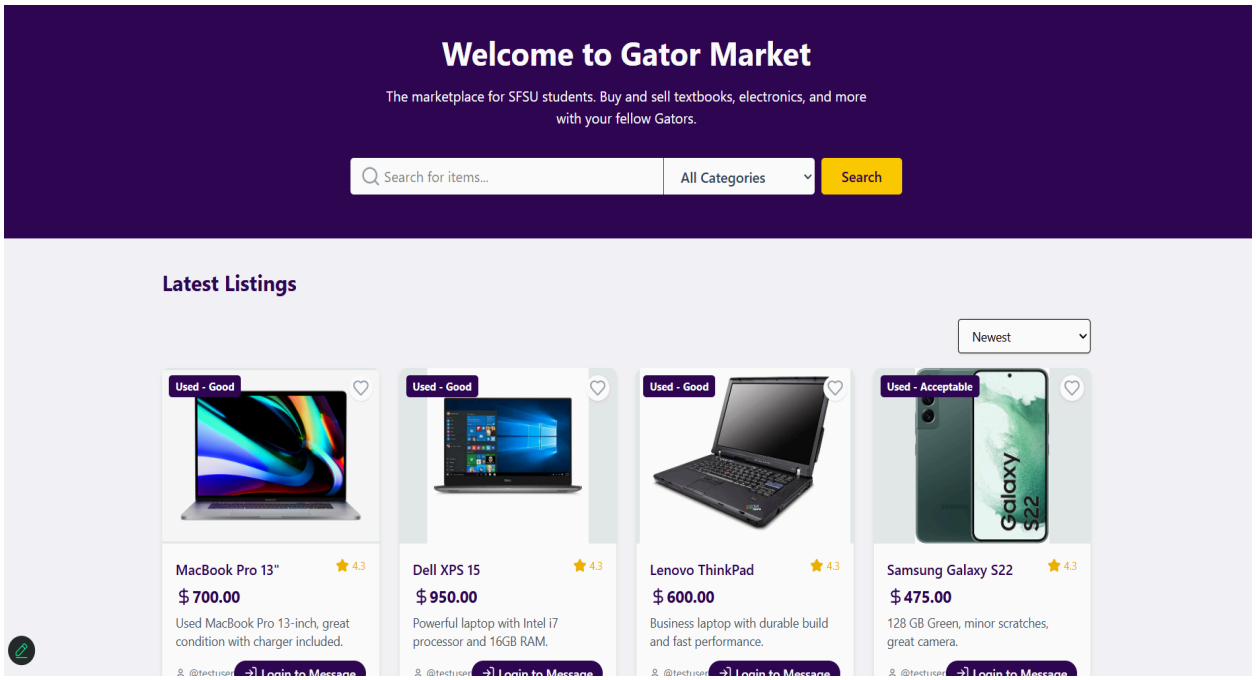


Figure 2: Login

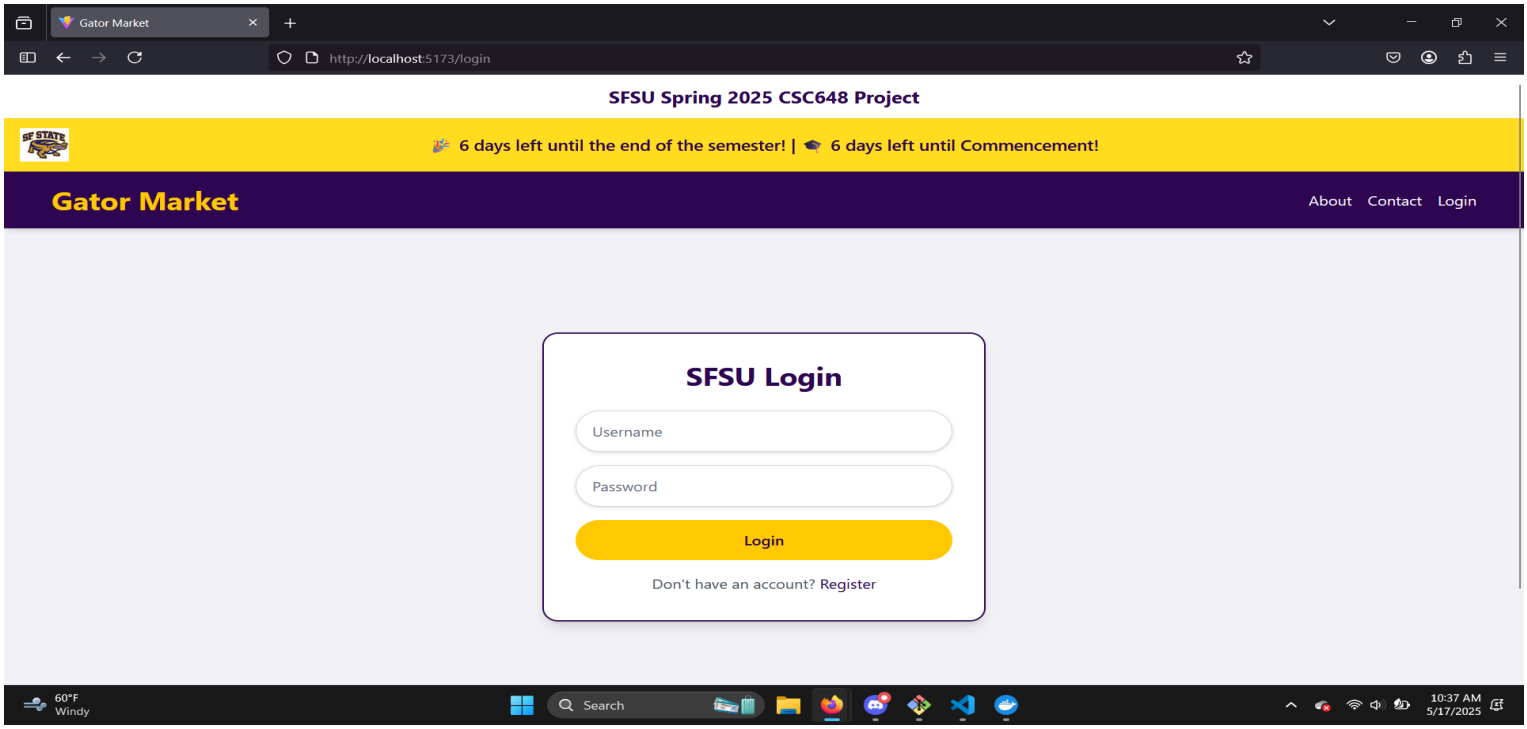


Figure 3: Registration

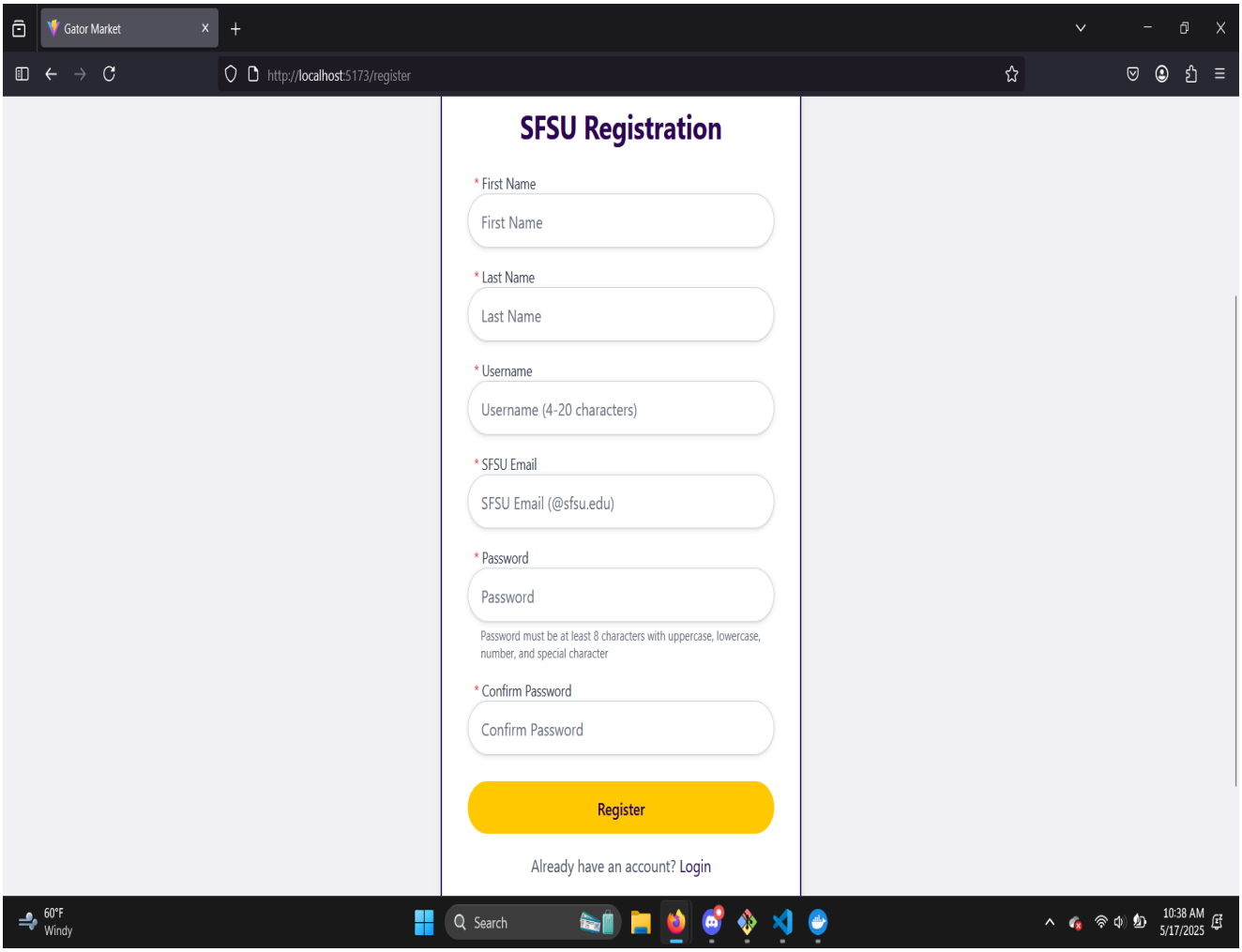
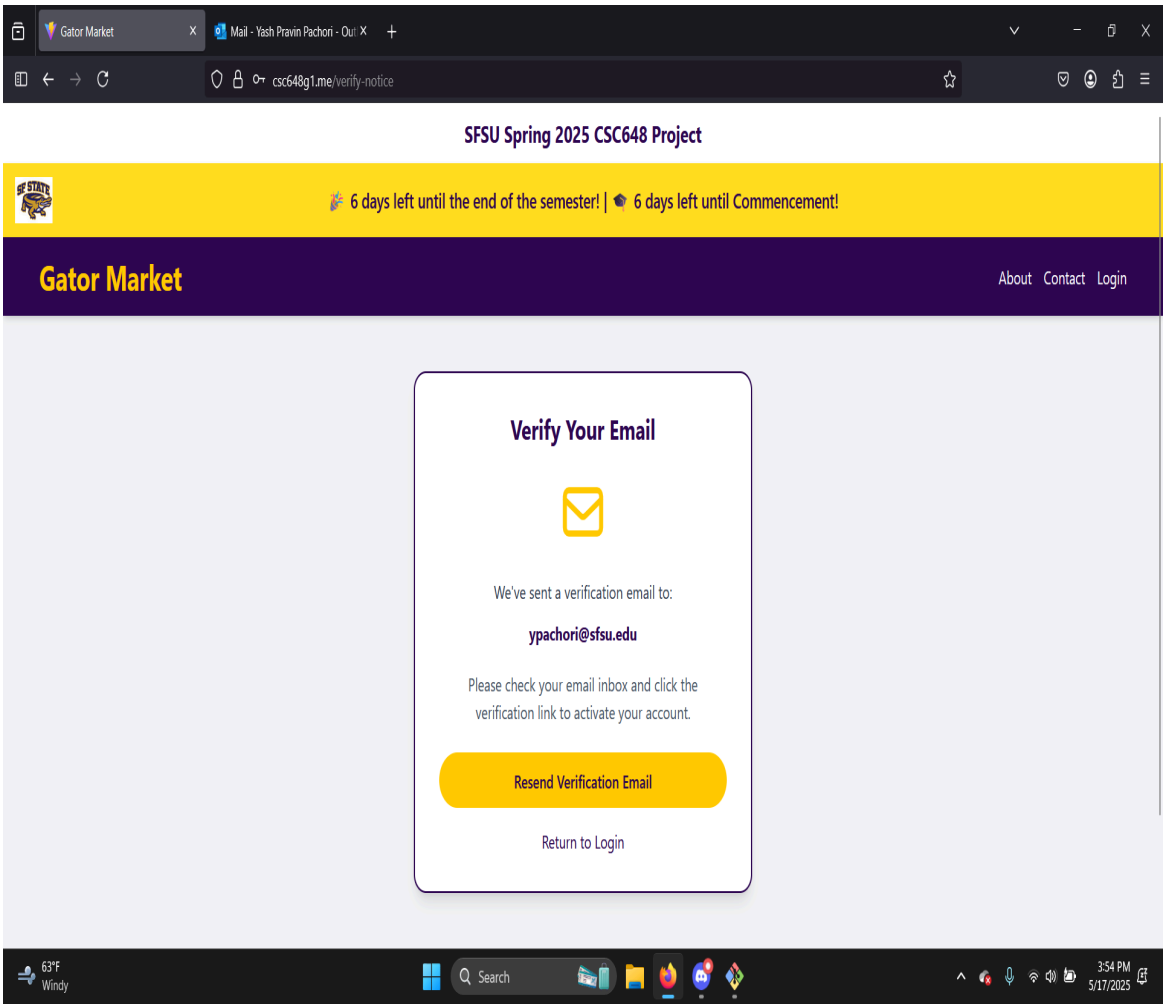


Figure 4: Email Verification



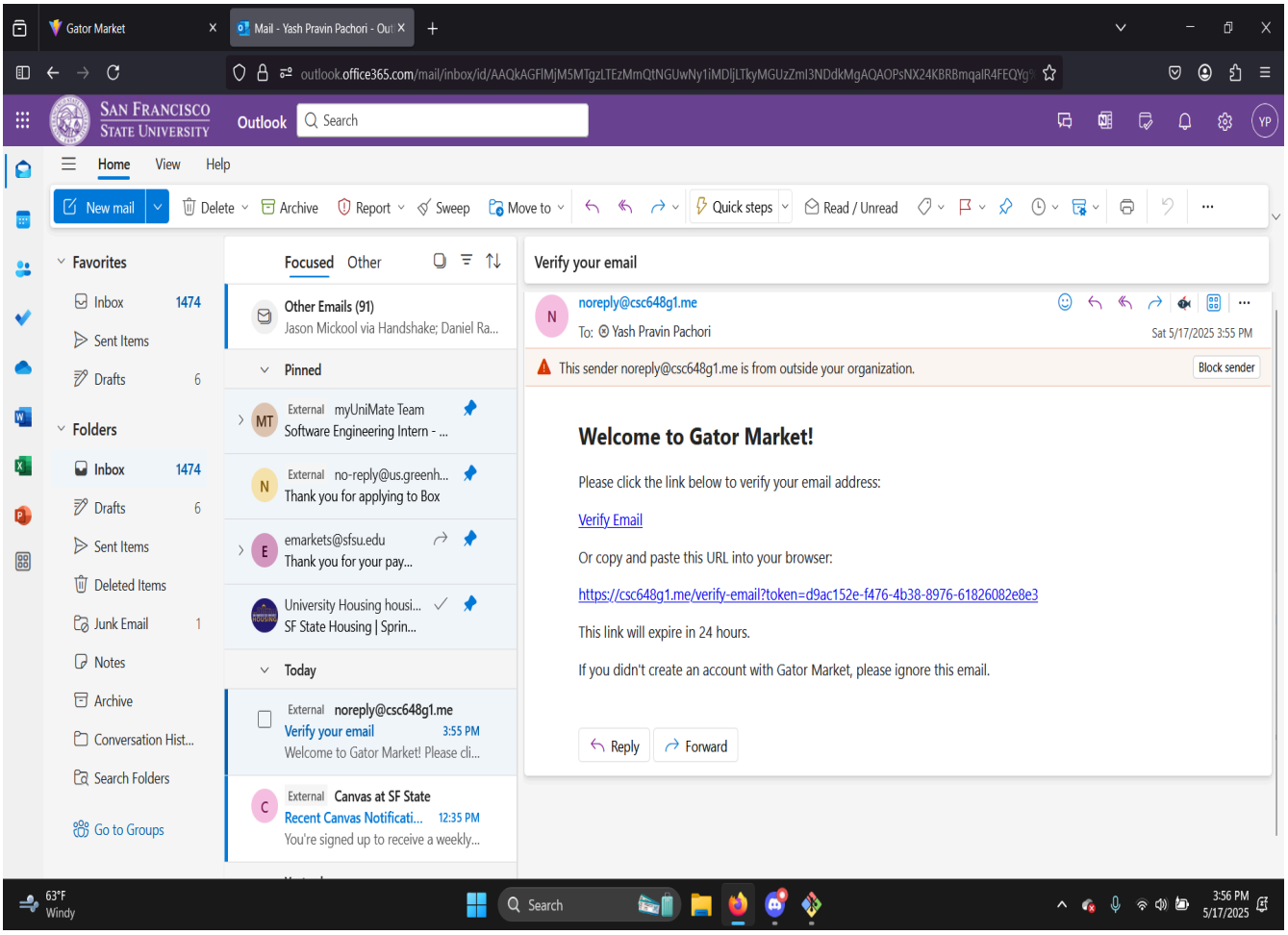
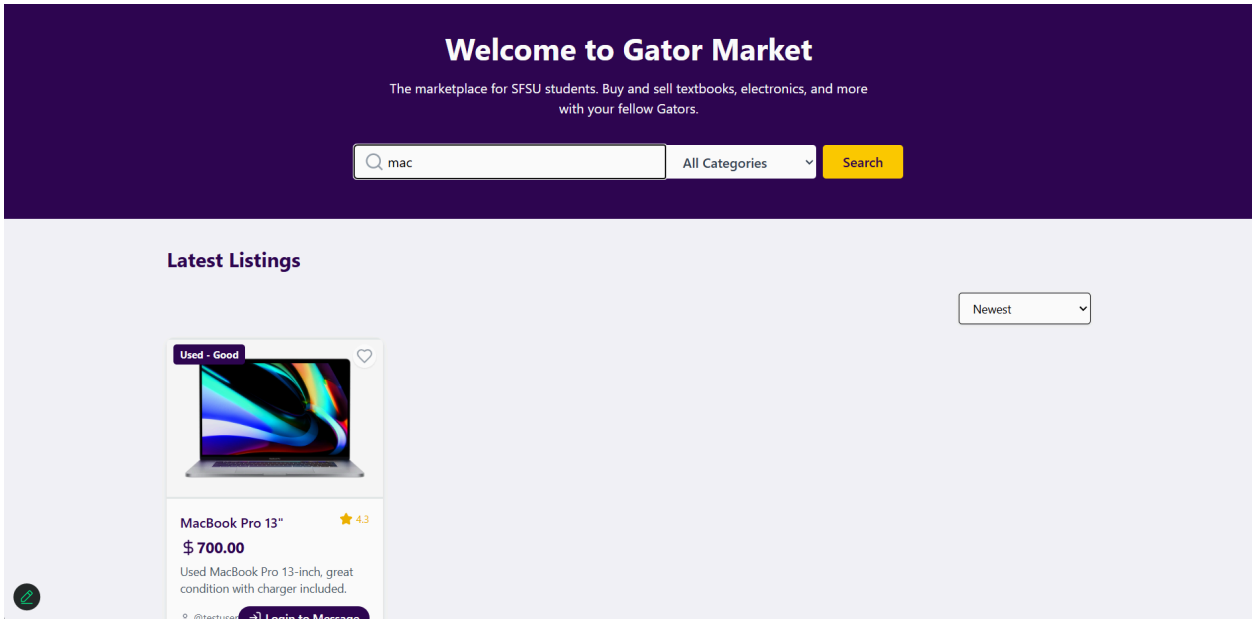

Gator Market
Mail - Yash Pravin Pachori - Out
outlook.office365.com/mail/inbox/id/AAQkAGFlMjM5MTgzLTEzMmQtNGUwNy1iMDljLTkyMGUzZmI3NDdkMgAQAOPsNX24KBRBmqaIR4FEQYg
San Francisco State University
Outlook
Search
Home
View
Help
New mail
Delete
Archive
Report
Sweep
Move to
Quick steps
Read / Unread
Favorites
Inbox 1474
Sent Items
Drafts 6
Folders
Inbox 1474
Drafts 6
Sent Items
Deleted Items
Junk Email 1
Notes
Archive
Conversation Hist...
Search Folders
Go to Groups
Focused
Other
Other Emails (91)
Jason Mickool via Handshake; Daniel Ra...
Pinned
External myUniMate Team
Software Engineering Intern - ...
External no-reply@us.greenh...
Thank you for applying to Box
emarkets@sfsu.edu
Thank you for your pay...
University Housing housi...
SF State Housing | Sprin...
Today
External noreply@csc648g1.me
Verify your email 3:55 PM
Welcome to Gator Market! Please cli...
External Canvas at SF State
Recent Canvas Notificati... 12:35 PM
You're signed up to receive a weekly...
Verify your email
noreply@csc648g1.me
To: Yash Pravin Pachori
Sat 5/17/2025 3:55 PM
This sender noreply@csc648g1.me is from outside your organization.
Block sender
Welcome to Gator Market!
Please click the link below to verify your email address:
Verify Email
Or copy and paste this URL into your browser:
https://csc648g1.me/verify-email?token=d9ac152e-f476-4b38-8976-61826082e8e3
This link will expire in 24 hours.
If you didn't create an account with Gator Market, please ignore this email.
Reply
Forward
63°F
Windy
Search
3:56 PM
5/17/2025

Figure 5: Search Results Page



Shows filtered product listings based on user-selected category and keyword, with results rendered dynamically.

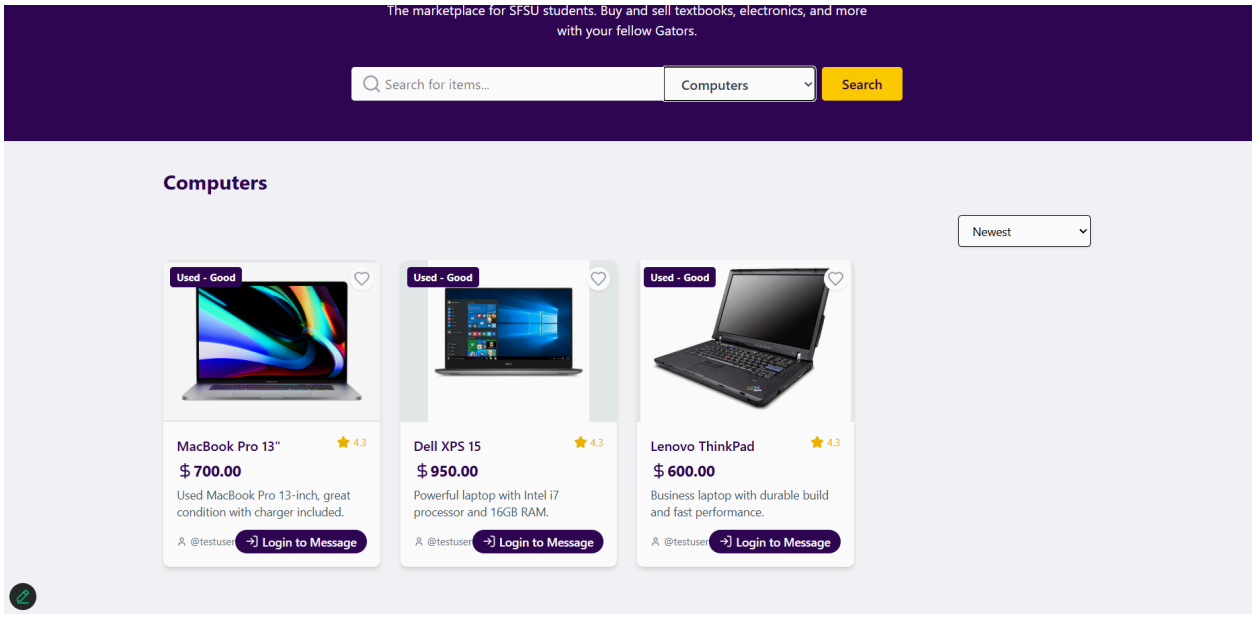


Figure 6: Product Upload Page (Create Listing)

Upload form interface where users can enter product details, upload images, and optionally enable meetup safety recording.

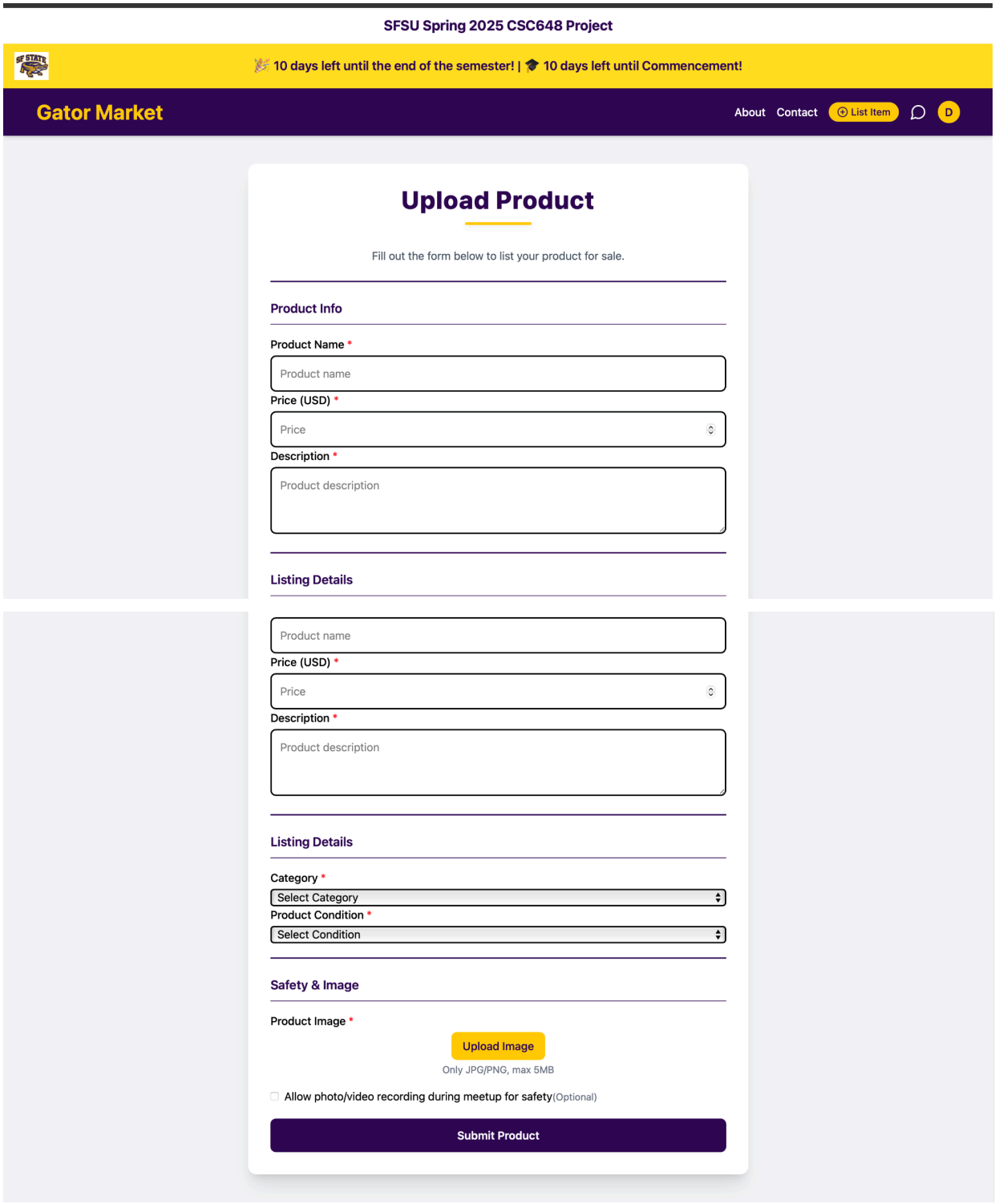


Figure 7: In-App Messaging Interface

A page view showing secure chat between buyer and seller without revealing personal contact information.

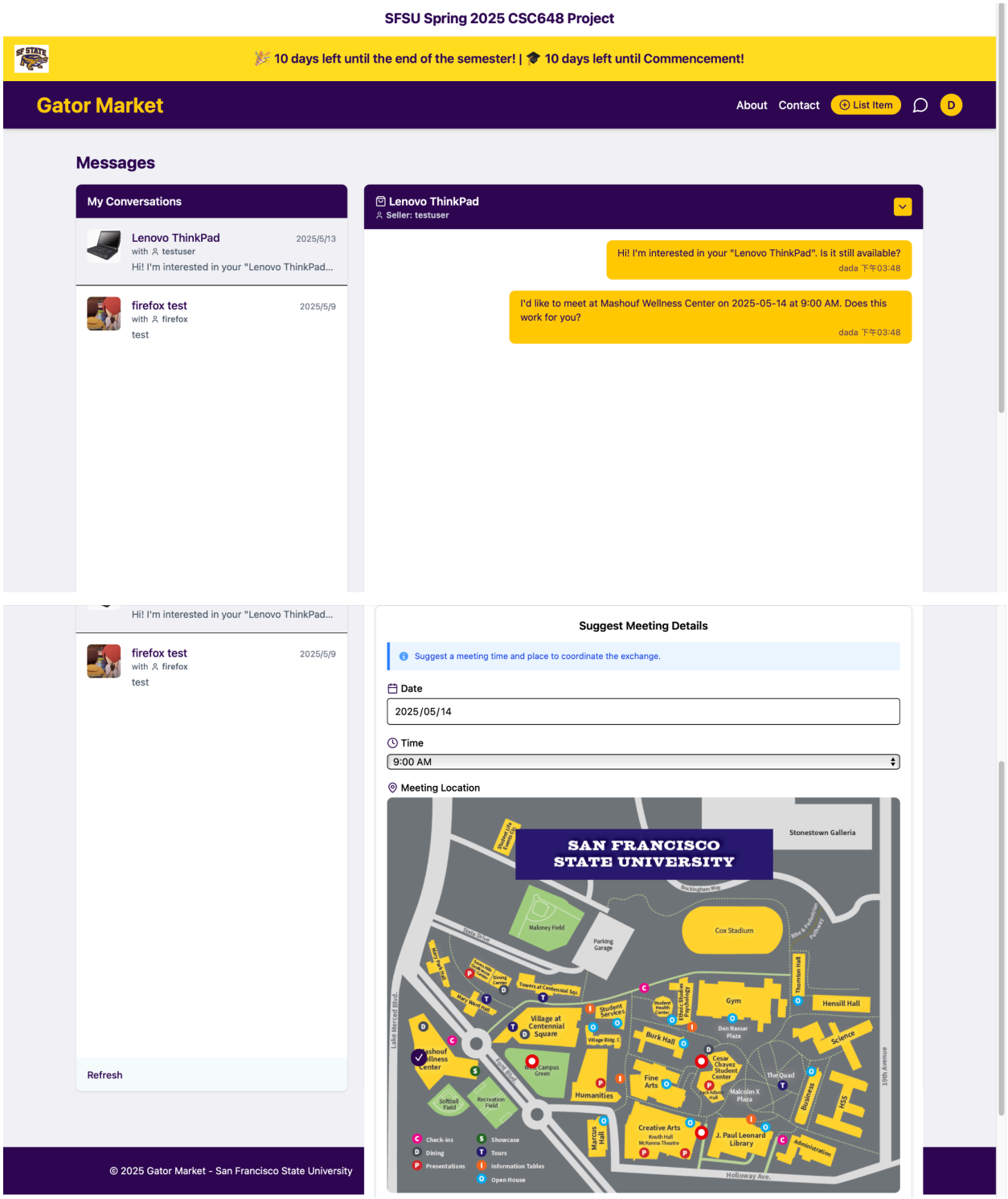


Figure 8: Product Details Page

Shows item information including image, name, price, condition, description, and contact options.

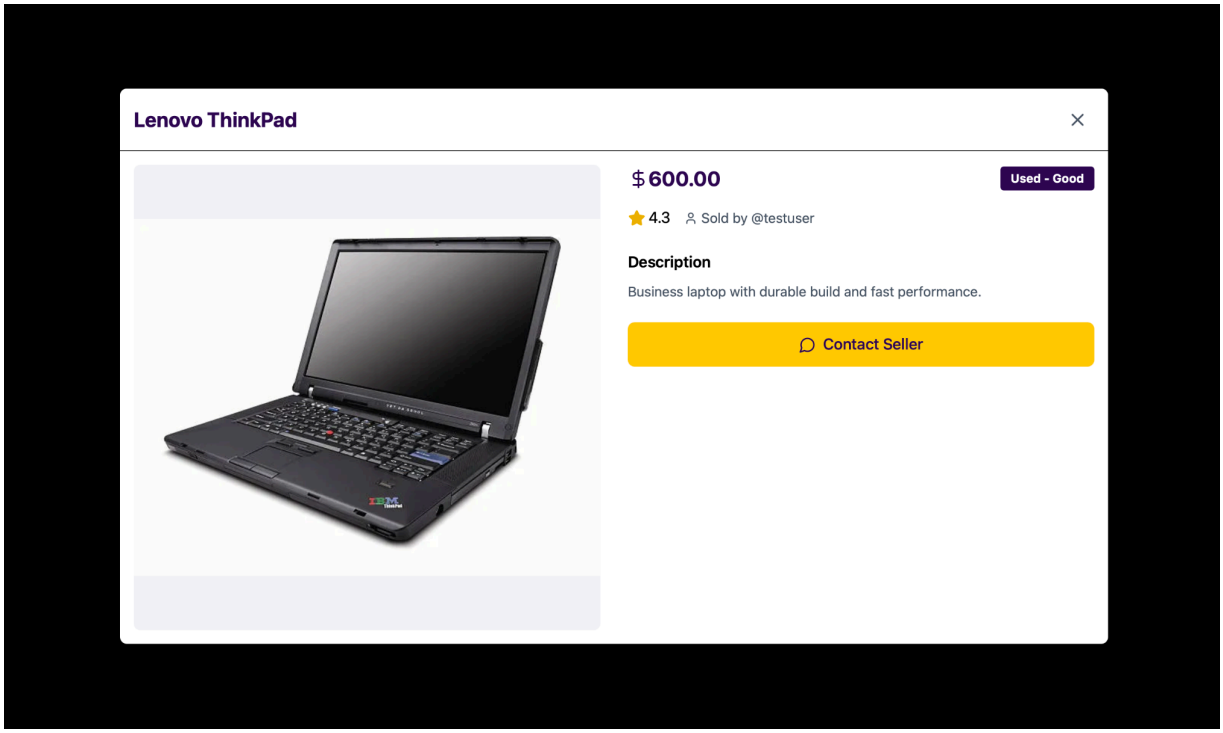


Figure 9: User Dashboard / My Listings View

Displays user's active listings, allowing them to edit, delete, or view engagement metrics.

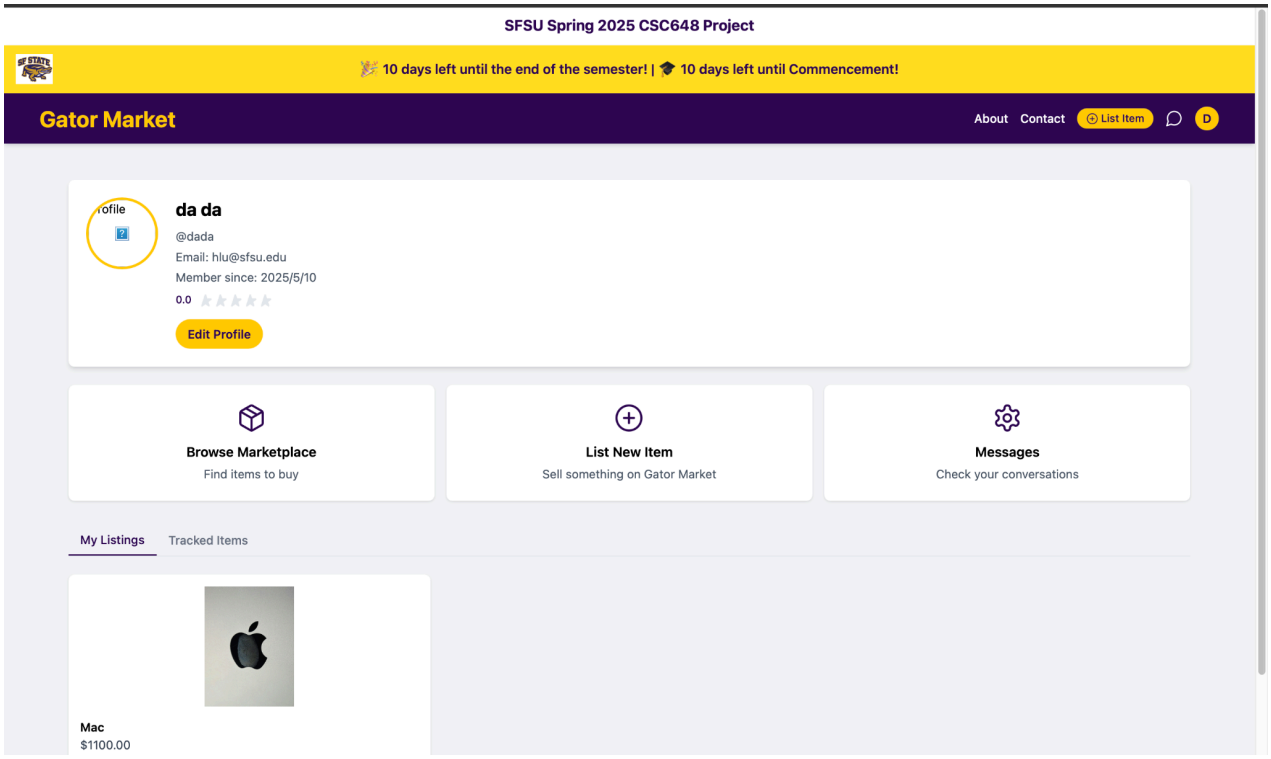
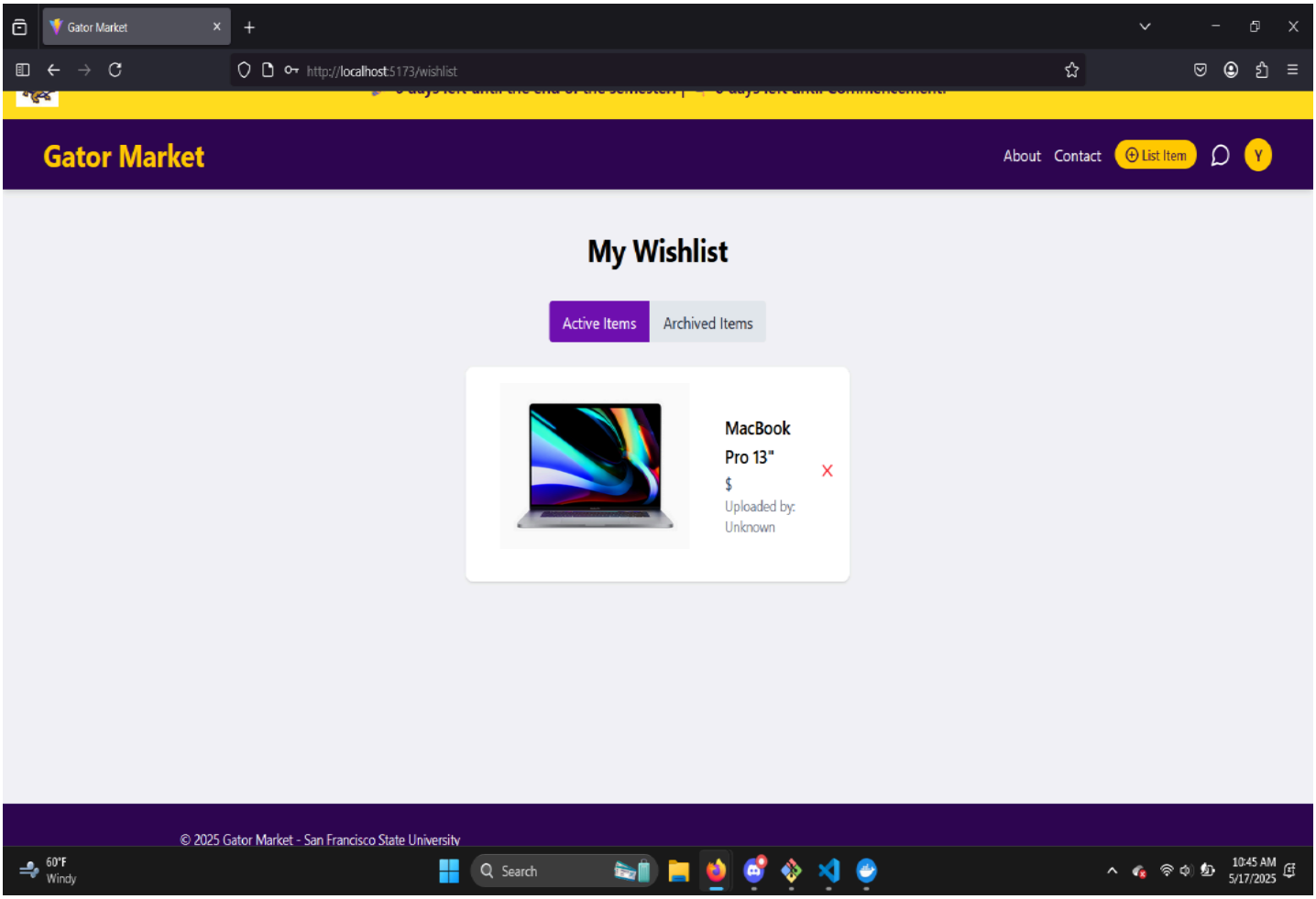


Figure 10: Wishlist and Archive Page

Page showing items the user has bookmarked for later viewing or potential purchase.



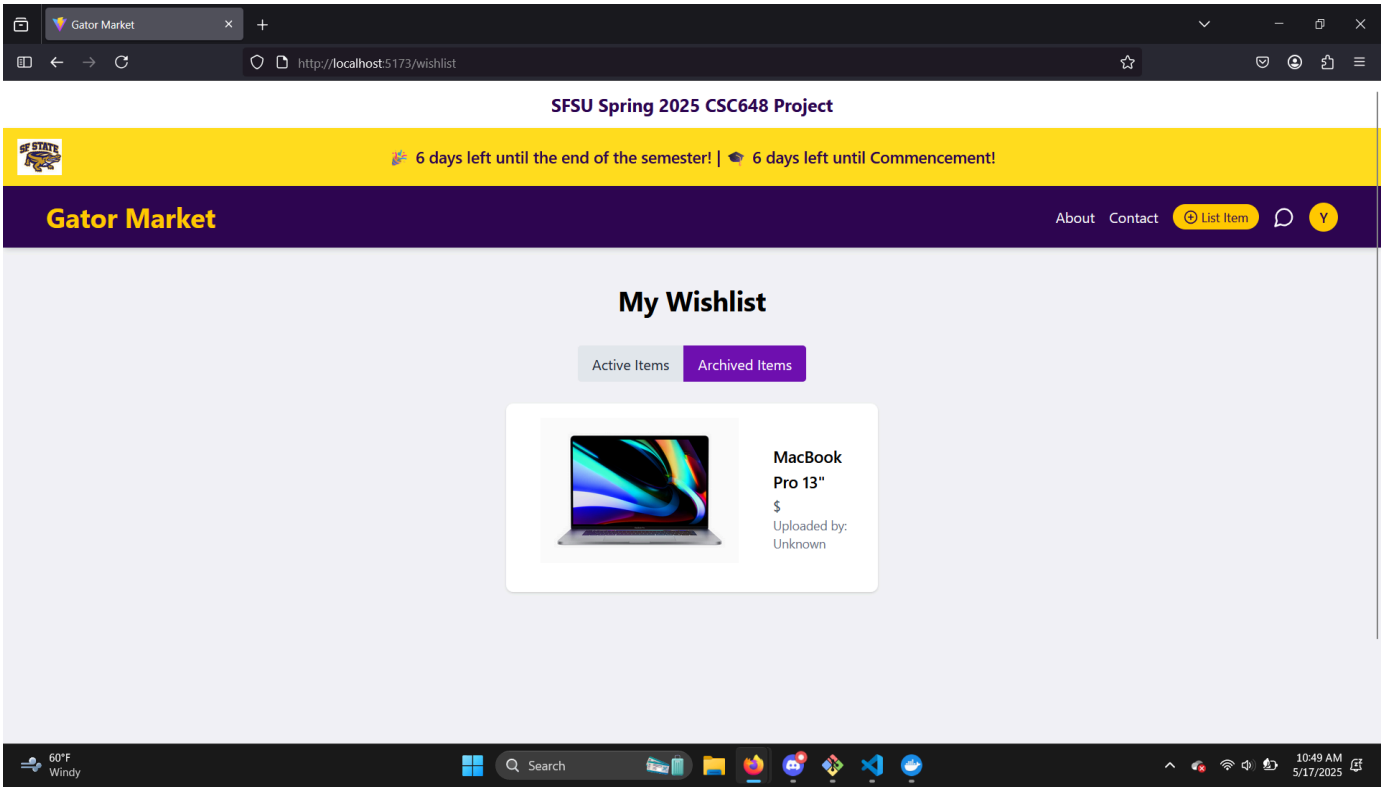

Gator Market
http://localhost:5173/wishlist
SFSU Spring 2025 CSC648 Project
6 days left until the end of the semester! | 6 days left until Commencement!
Gator Market
About
Contact
List Item
Y
My Wishlist
Active Items
Archived Items
MacBook Pro 13"
$
Uploaded by: Unknown
60°F
Windy
Search
10:49 AM
5/17/2025

Figure 11: Report User

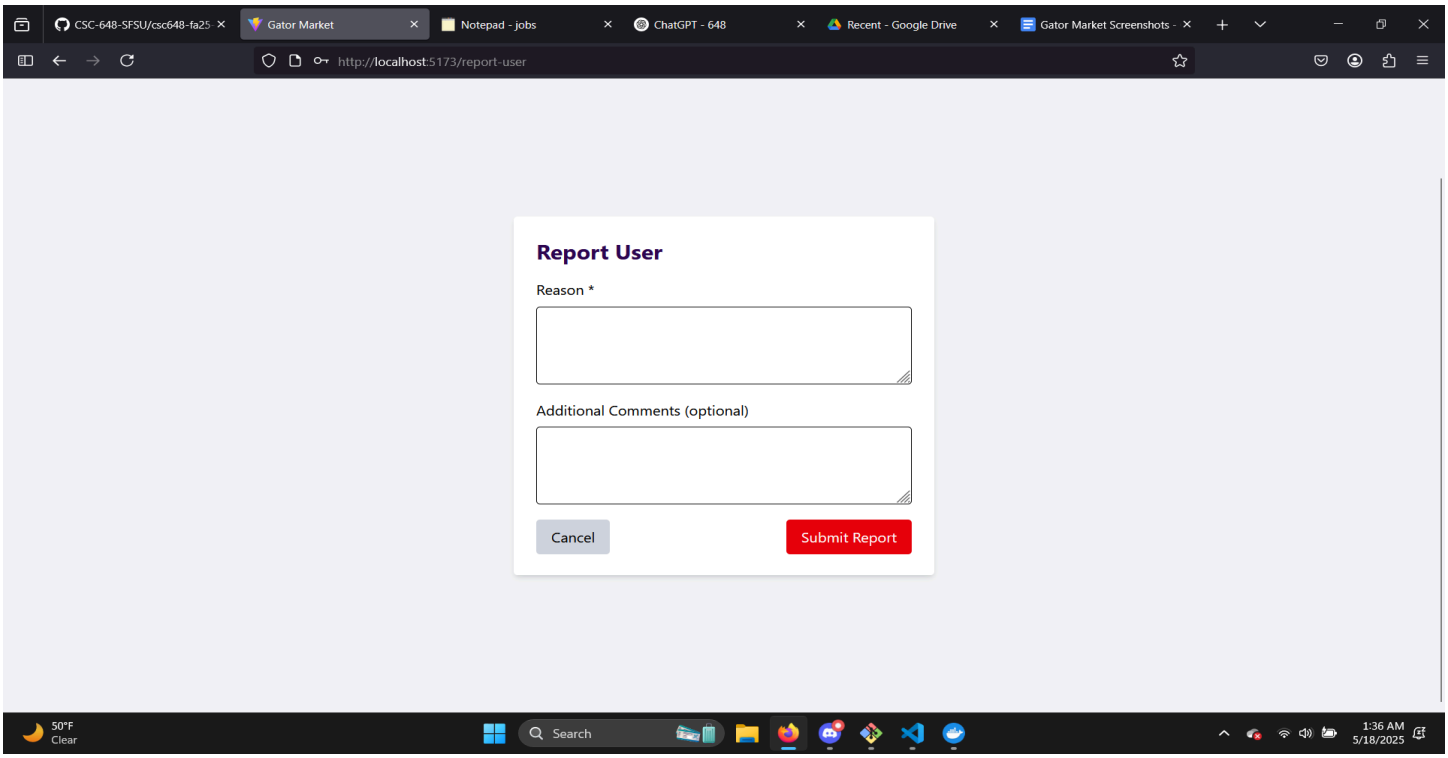


Figure 12: Leave a Review for the User

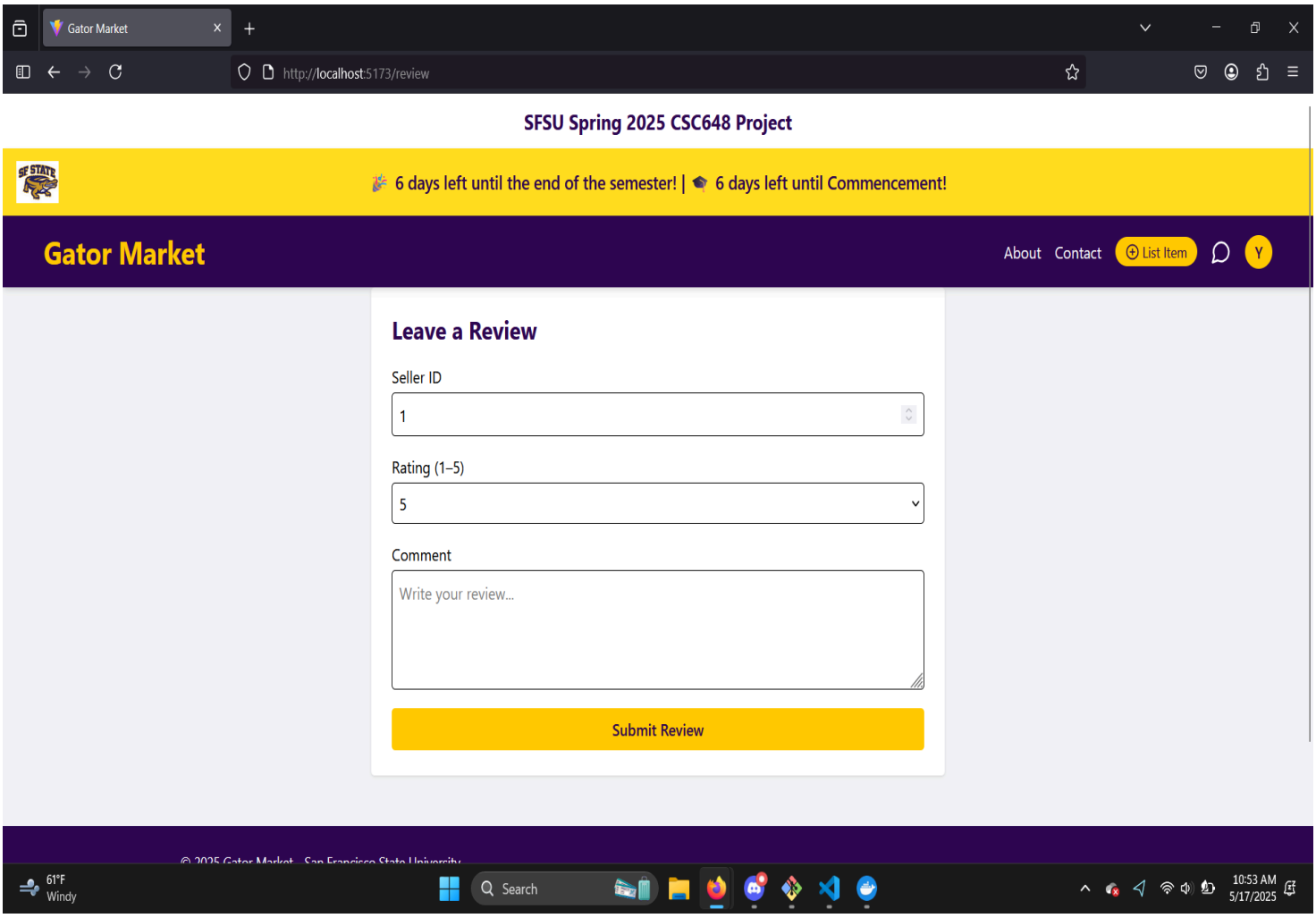


Figure 13: Edit Profile

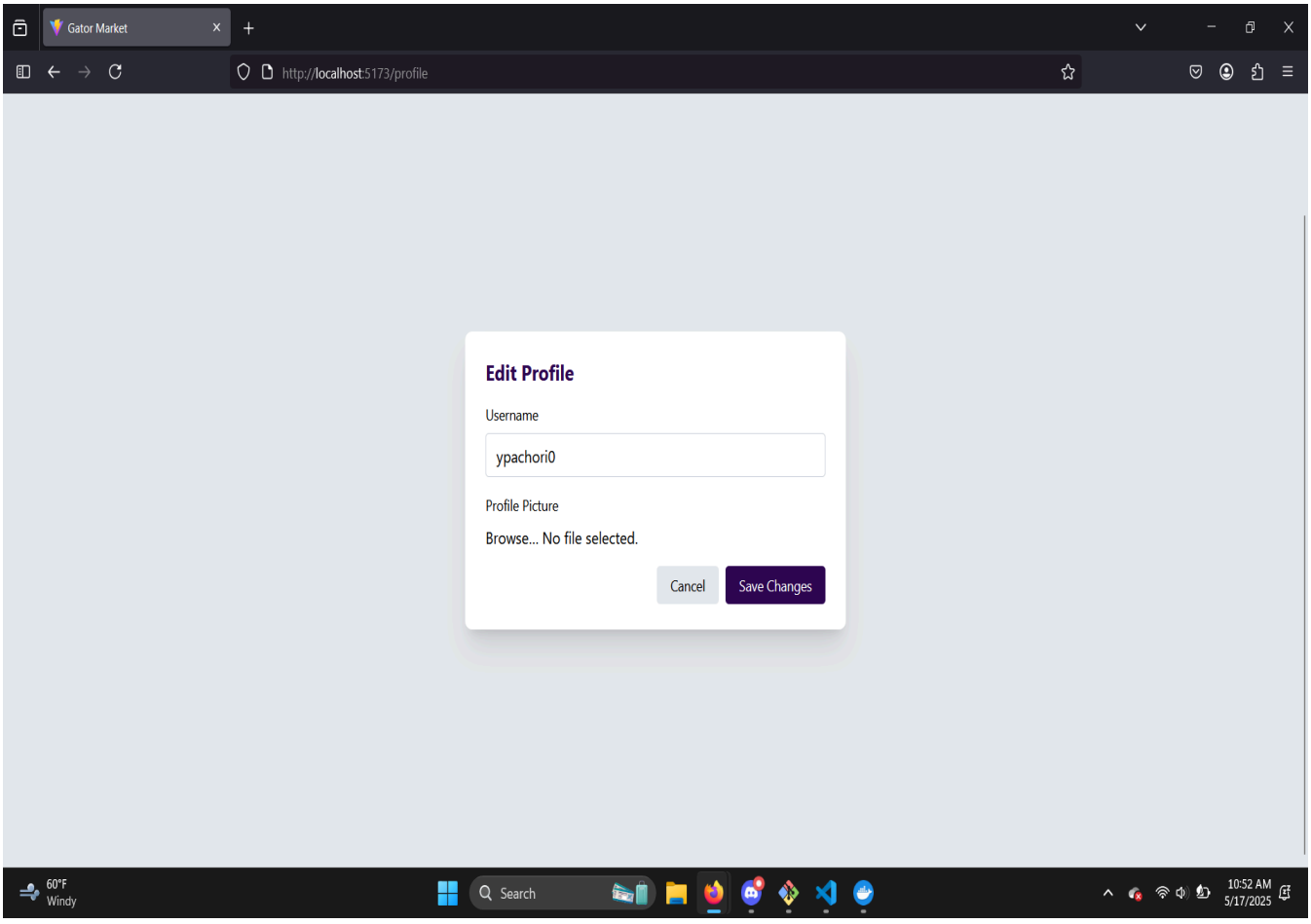


Figure 14: Public Profile

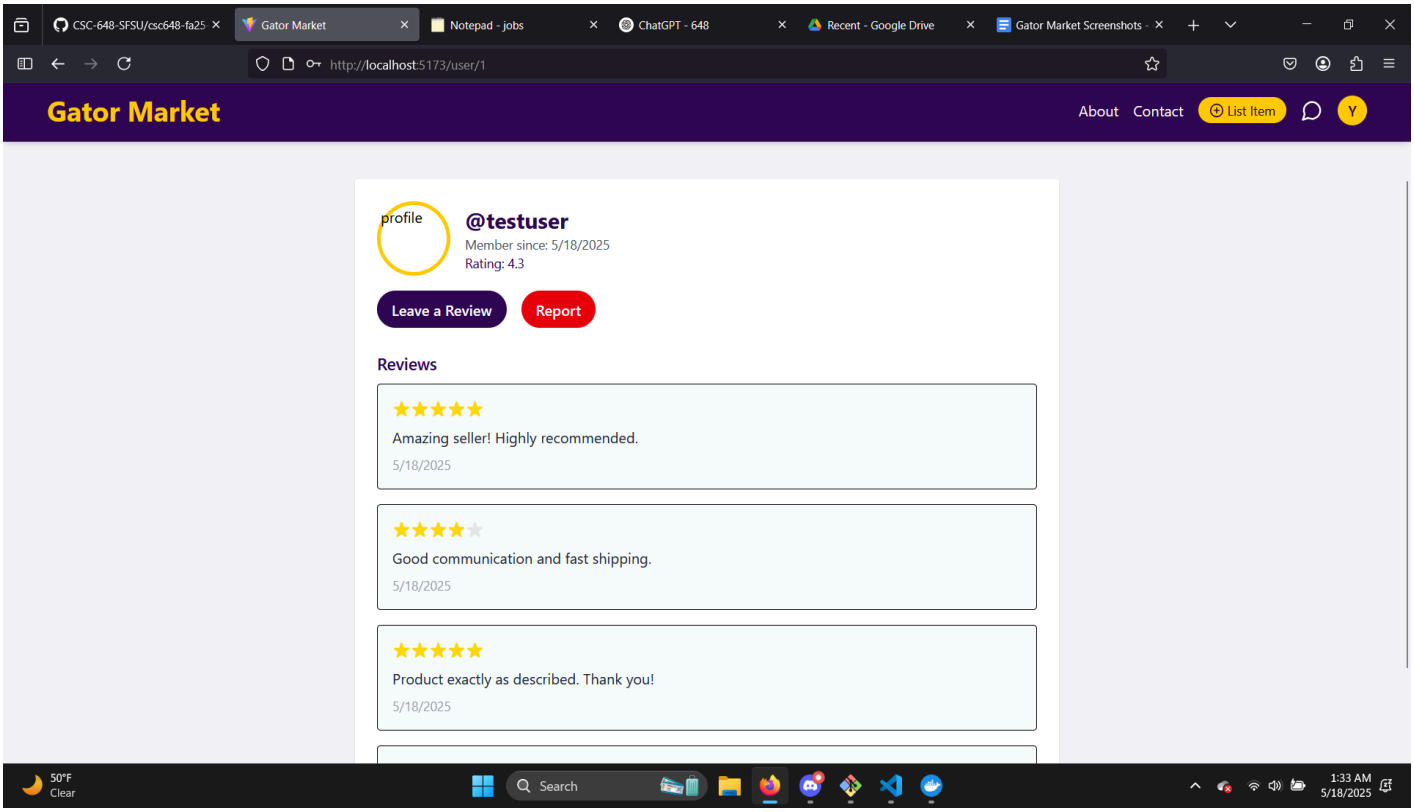
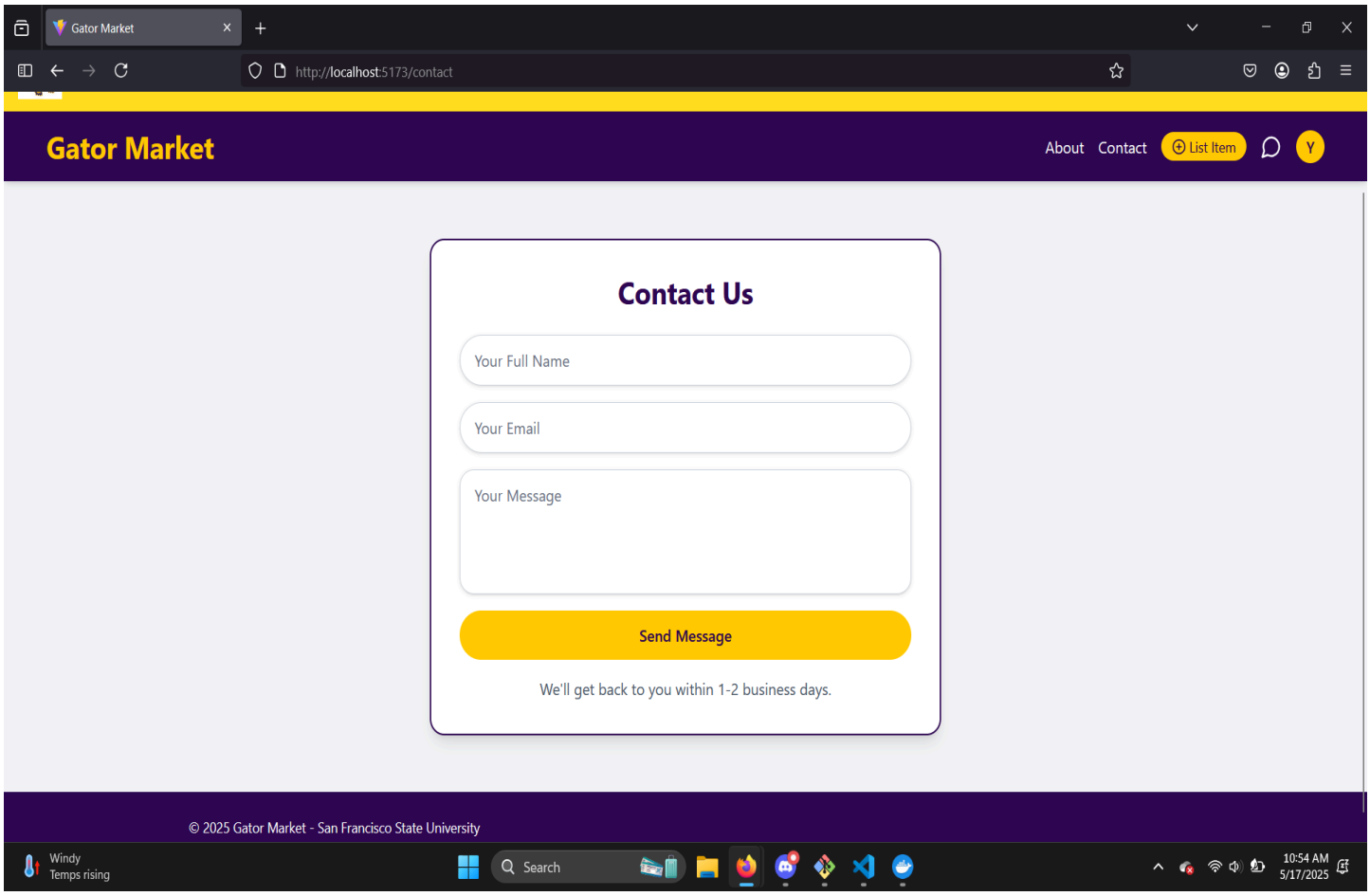
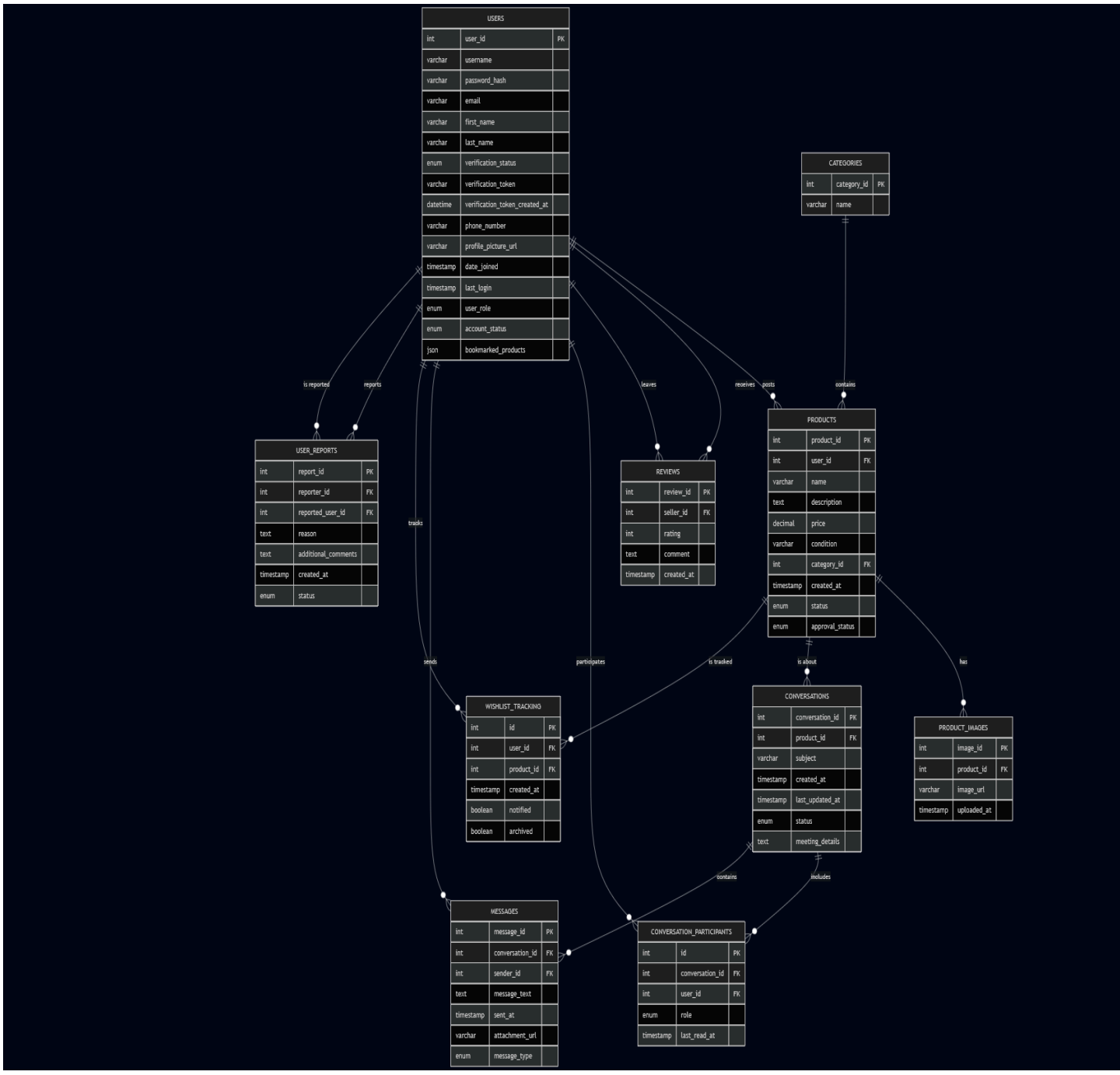


Figure 15: Contact Us



4) Database Organization



The Gator Market database is designed around a normalized, scalable schema optimized for student-to-student transactions. The schema includes key tables such as users, products, categories, messages, and several supporting tables to enable features like image uploads, reviews, wishlist tracking, conversations, and reporting.

In the current design, all product listings are stored in a single unified products table, rather than using separate tables for each category (e.g., books, electronics, clothes). This avoids redundancy, simplifies cross-category querying, and allows the database to scale efficiently with new categories in the future. Each product includes general attributes (e.g., name, price, description, condition) and is linked to a category_id and a user_id for ownership and classification.

User messages are organized using a robust messaging system consisting of conversations, conversation_participants, and messages tables, supporting multi-user threads and message metadata.

Product images are stored in the product_images table, allowing one-to-many relationships between products and images, and supporting S3-based secure media storage via URLs.

The reviews table enables seller ratings and feedback after transactions, and wishlist_tracking allows users to bookmark items. Suspicious behavior can be flagged through the user_reports table, which stores reports made by users against others.

Products Table

Field	Type	Description
product_id	INT, PK, Auto Increment	Unique product ID
user_id	INT, FK to users	Seller (User)
name	VARCHAR(100)	Product name
description	TEXT	Product description

price	DECIMAL(10,2)	Product price
condition	ENUM('new', 'used', 'refurbished')	Item condition
image_url	VARCHAR(255)	Link to product image
category_id	INT, FK to categories	Product category
created_at	TIMESTAMP	Listing date
status	ENUM('active', 'sold', 'deleted')	Product listing status
approval_status	ENUM('pending', 'approved', 'rejected')	Moderation status

Categories Table

Field	Type	Description
category_id	INT, PK, Auto Increment	Unique category ID
name	VARCHAR(100)	Category name (e.g., electronics, books)

Users Table

Field	Type	Description
user_id	INT, PK, Auto Increment	Unique user ID
username	VARCHAR(50)	Unique username for login
password_hash	VARCHAR(255)	Hashed password
email	VARCHAR(100)	Unique SFSU email
first_name	VARCHAR(50)	First name
last_name	VARCHAR(50)	Last name
verification_status	ENUM('permanent', 'annual', 'not verified')	SFSU verification type
phone_number	VARCHAR(15)	Optional contact number
profile_picture_url	VARCHAR(255)	Optional profile picture URL
date_joined	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Registration date

last_login	TIMESTAMP NULL	Most recent login timestamp
user_role	ENUM('user', 'moderator', 'admin') DEFAULT 'user'	User privilege level
account_status	ENUM('active', 'inactive/banned', 'deleted') DEFAULT 'active'	Account state

Messages Table

Field	Type	Description
message_id	INT, PK, Auto Increment	Unique message ID
sender_id	INT, FK to users	Sender's user ID
receiver_id	INT, FK to users	Receiver's user ID
product_id	INT, FK to products	Associated product (optional)
message_text	TEXT	Message content
sent_at	TIMESTAMP	Timestamp when the message was sent

Conversations

Field	Type	Description
conversation_id	INT, PK, Auto Increment	Unique conversation ID
product_id	INT, FK to products	ID of product that the conversation is about
subject	VARCHAR	Conversation subect
created_at	TIMESTAMP	When conversation started
last_updated_at	TIMESTAMP	Last time the conversation was active
status	ENUM	Whether conversation is ongoing or not
meeting_details	TEXT	First few lines of most recent message

Conversation Participants

Field	Type	Description
id	INT, PK, Auto Increment	Unique ID

conversation_id	INT, FK to conversations	ID of conversation
role	ENUM	Role of the user in the conversation
last_read_at	TIMESTAMP	Last time the user’s message was read

Reviews

Field	Type	Description
review_id	INT, PK, Auto Increment	Unique review ID
seller_id	INT, FK to users	Seller's user ID
rating	INT	Rating out of 5
comment	TEXT	Comment written in review
created_at	TIMESTAMP	When review was left

Product_Images

Field	Type	Description

image_id	INT, PK, Auto Increment	Unique image ID
product_id	INT, FK to product	ID of the product
image_url	VARCHAR	URL of the uploaded image
uploaded_at	TIMESTAMP	When the image was uploaded

Wishlist_Tracking

Field	Type	Description
id	INT, PK, Auto Increment	Unique ID
user_id	INT, FK to users	ID of the user
product_id	INT, FK to products	Associated product
created_at	TIMESTAMP	When item was saved
notified	boolean	Whether a notification for this product was sent
archived	boolean	Whether the item has been archived

User_Reports

Field	Type	Description
report_id	INT, PK, Auto Increment	Unique report ID
reporter_id	INT, FK to users	ID of the user that is filing the report
reported_user_id	INT, FK to users	ID of the user that is being reported
reason	TEXT	Reason for report
additional_comments	TEXT	Any additional comments
created_at	TIMESTAMP	When the report was filed
status	ENUM	Status of report (pending, reviewed, resolved)

Relationships

One-to-One:

- Each user has a unique verification status and profile.

One-to-Many:

- One user can post multiple products
- One product can have multiple images
- One user can receive multiple reviews as a seller
- One category can contain multiple products
- One conversation can contain multiple messages
- One conversation can involve multiple participants

Many-to-Many:

- Many users participate in many conversations
- Many users can wishlist many products
- Many users can have many reports

5. Data Flow and Interaction

- User Request: User accesses the platform to browse or post products.
- API Call to Backend: The frontend sends API requests to fetch, post, or update product information.
- Backend Processes Request: The backend handles business logic and interacts with the database.
- Database Query Execution: The database returns product, user, or message data as requested.
- Response to Frontend: The backend returns the data as a JSON response.
- Frontend Displays Data: The frontend dynamically displays listings, messages, or user information.

6. Deployment Strategy

- Frontend: Hosted on AWS S3/CloudFront.

- Backend: Deployed on AWS EC2.
- Database: Hosted on MySQL service (AWS RDS or equivalent).
- Media Storage: Product images stored in localStorage
  → Magic and mimetype security ensure what type of file is being uploaded (JPEG or PNG) and nginx 404 returns to keep them secure and prevent harmful code injection

[User] → [React Frontend] → [Node.js Backend]

↓

[localStorage - Store Images]

↓

[MySQL - Save image URL in products table]

Product images uploaded through the frontend are now stored locally using the browser's localStorage API. When a user uploads an image, it is converted to a base64-encoded string and saved in the user's local browser storage. The encoded image string is then submitted along with the product form and stored in the backend's product_images table. This approach simplifies development by removing the need for external storage services and allows image data to be displayed immediately without requiring asynchronous fetching from a remote server.

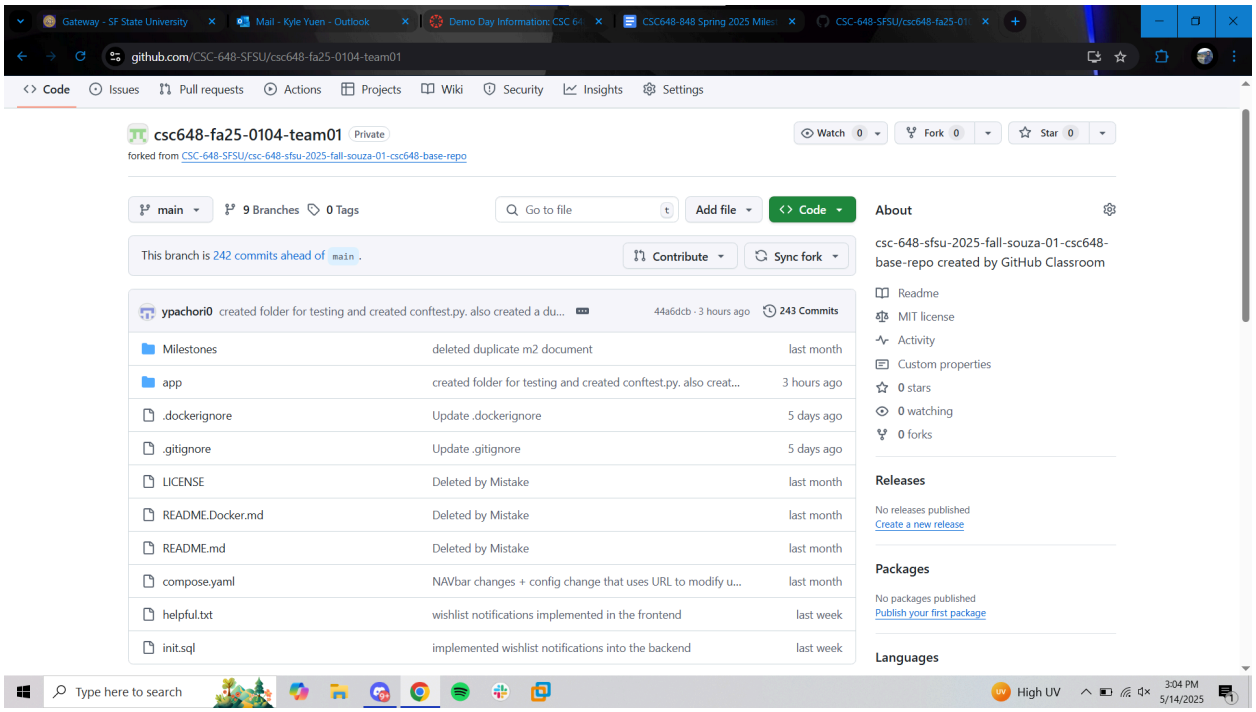
5) GitHub Organization

https://github.com/CSC-648-SFSU/csc648-fa25-0104-team01

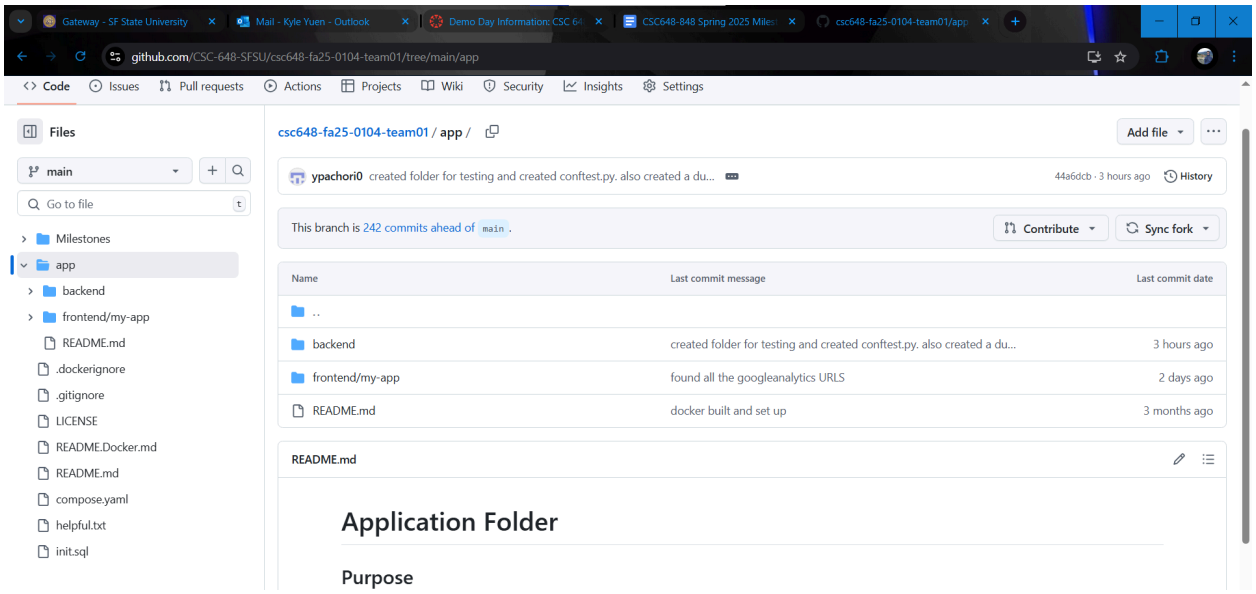
Our github master is Daniel O.

Everyone had access to our main branch due to our smaller scale team and somewhat different priorities. In retrospect, a more code review focused setup might have been better but we were basically just rolling with 'if there's a bug, fix it if you find it.'

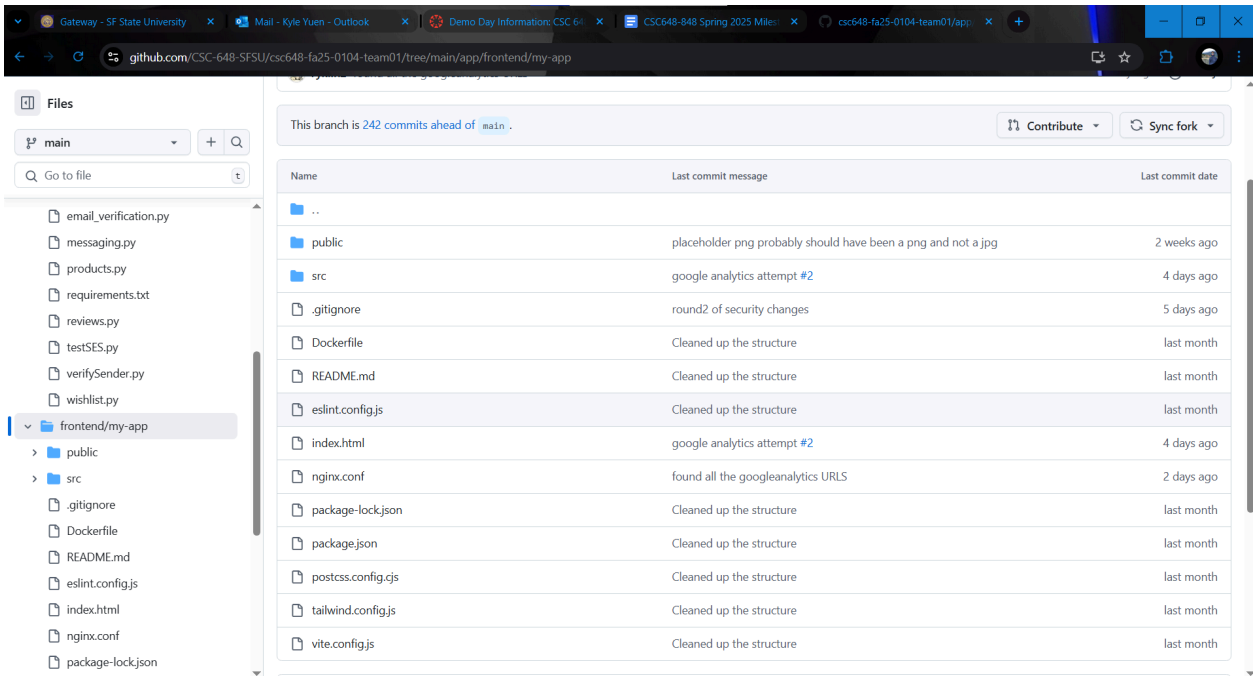
Our Github home page



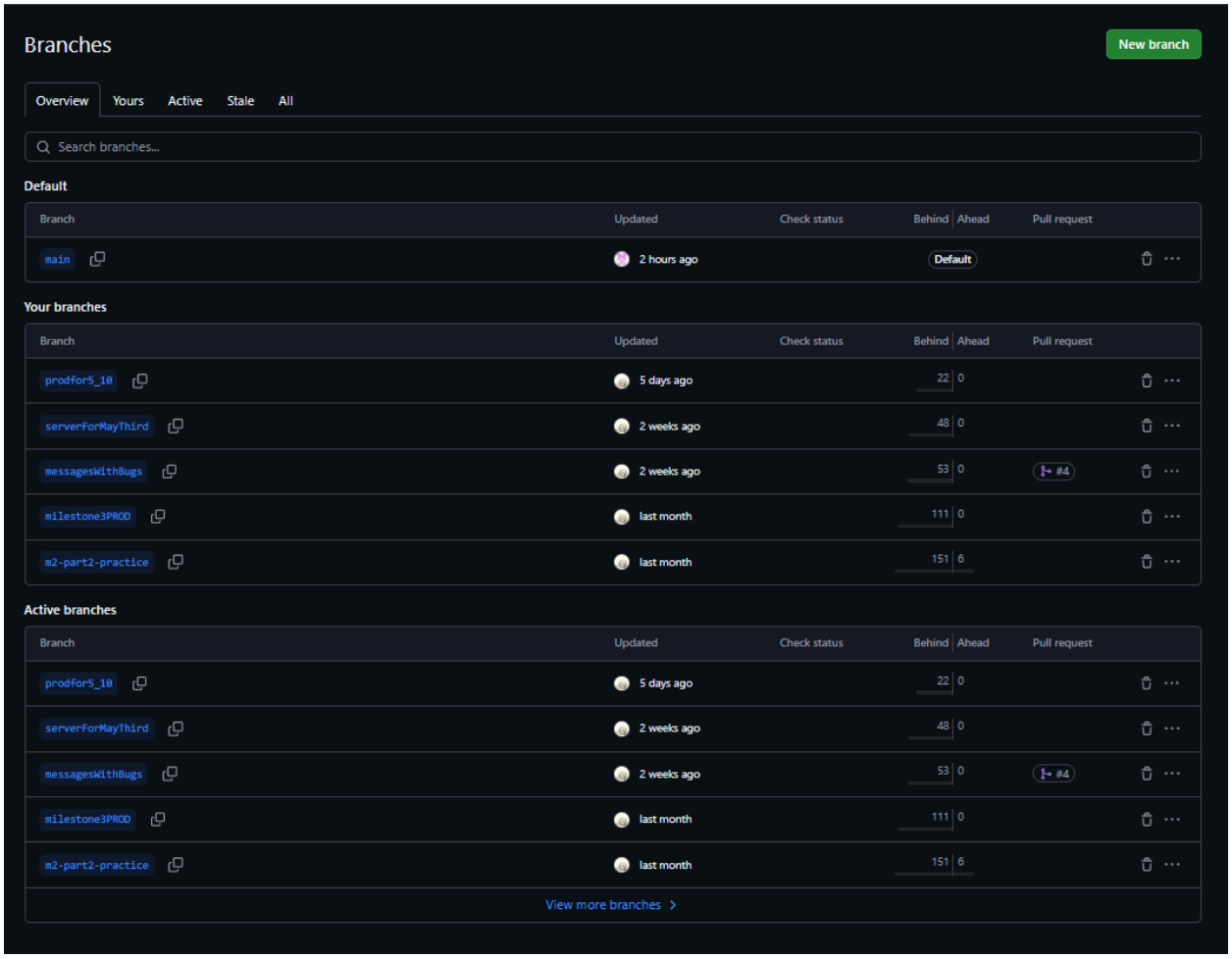
Our github frontend and backend separated



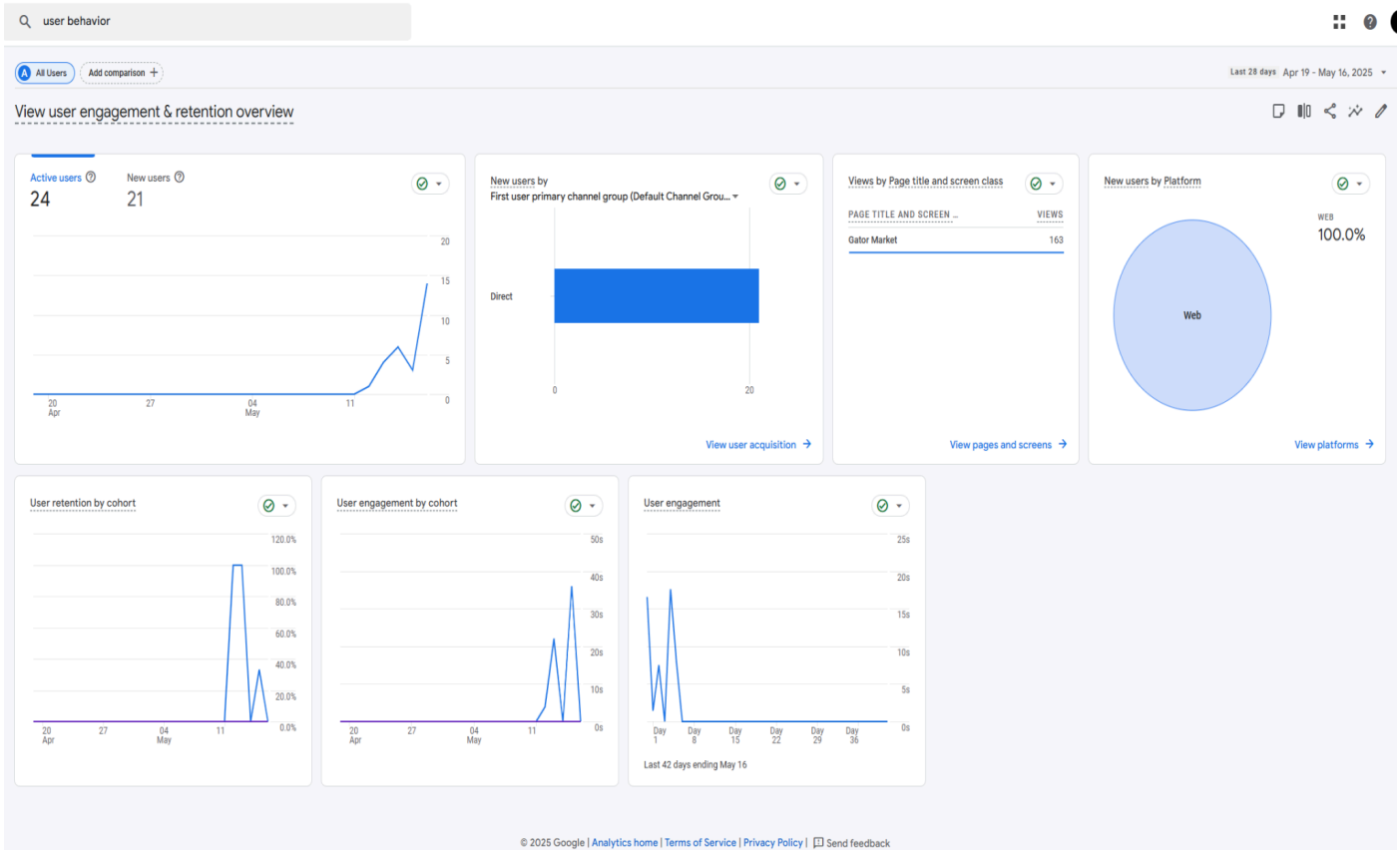
All of our front end pages



Our Branches prioritized feature freezes and production servers while essentially using our main branch as a development server. This works for a small team that can also afford to shift slowly with the pace of our class, but may not work as well in real world scenarios with needs for quick development or with larger teams.



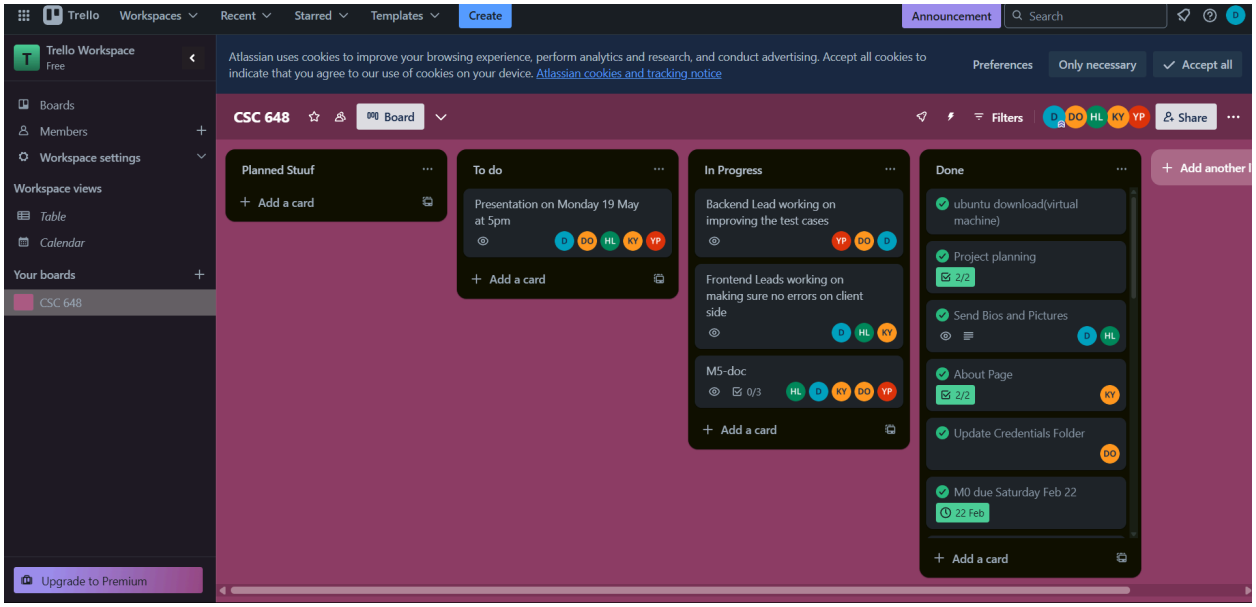
6) Google Analytics Stats

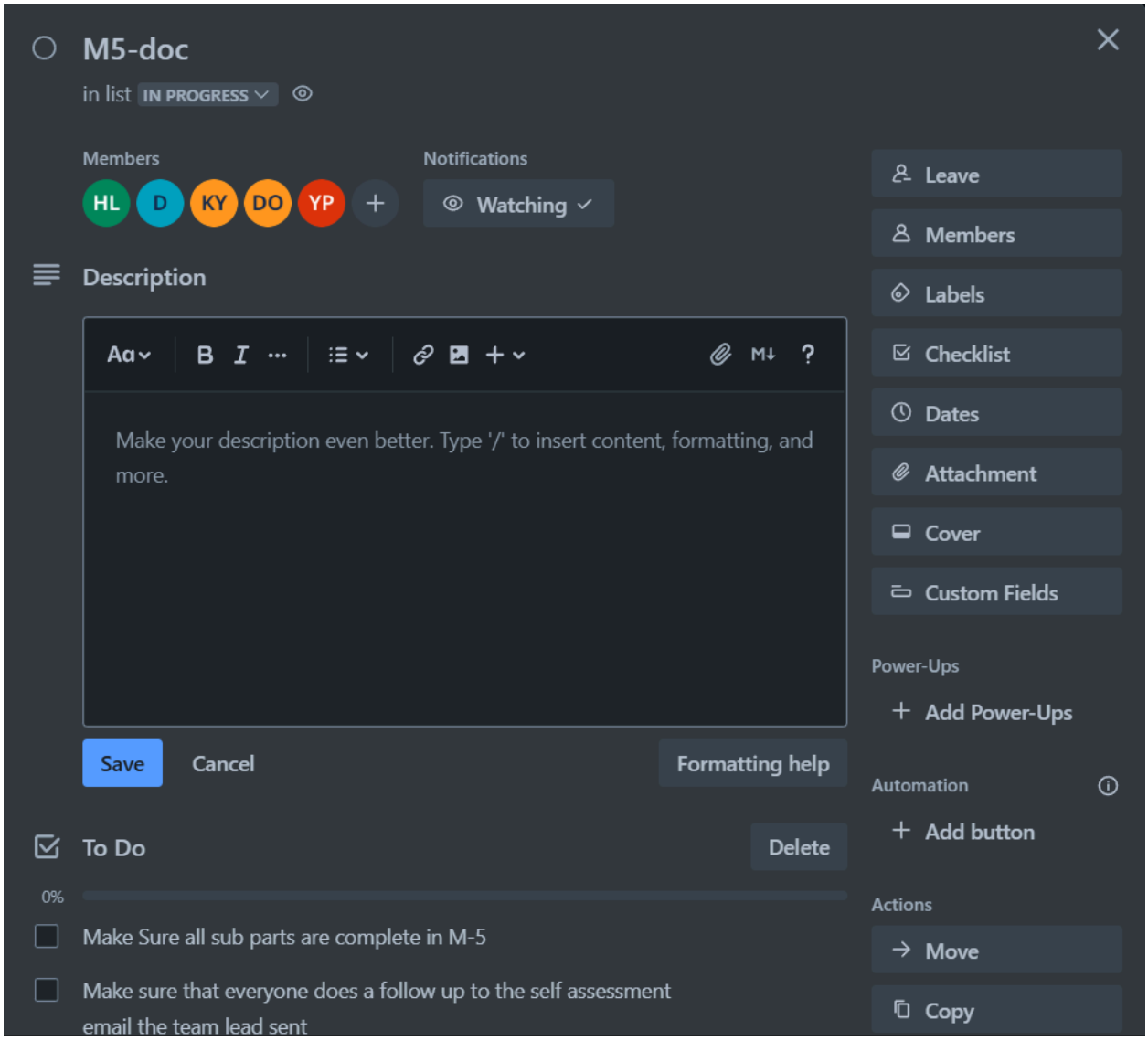


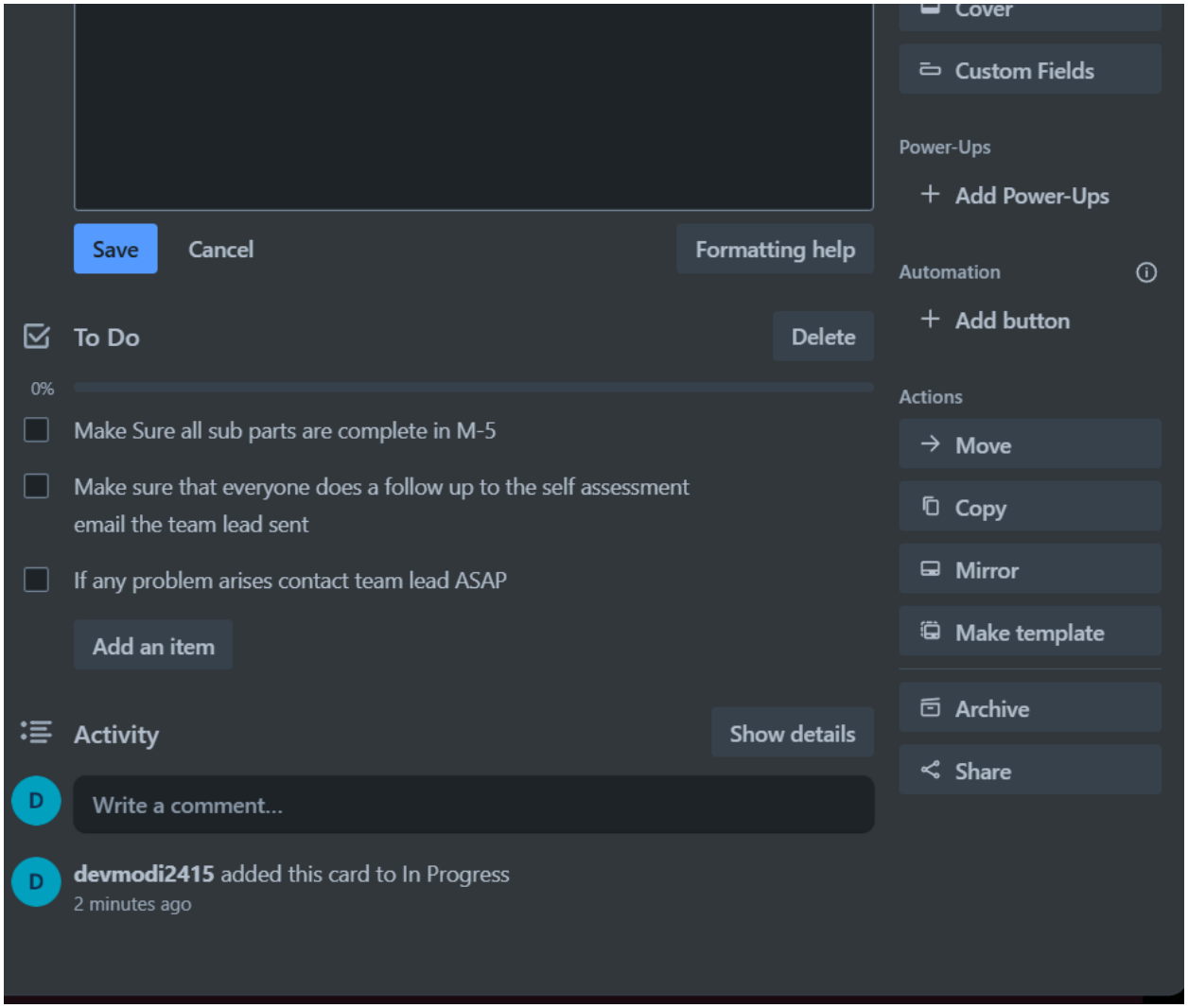
7) Project Management

For our project management, we primarily used Trello and Discord. Trello helped us organize tasks, assign responsibilities, and track progress on various aspects of the project. We had different boards set up for frontend and backend tasks, along with an overall project board to ensure everything was moving forward smoothly.

Trello:-

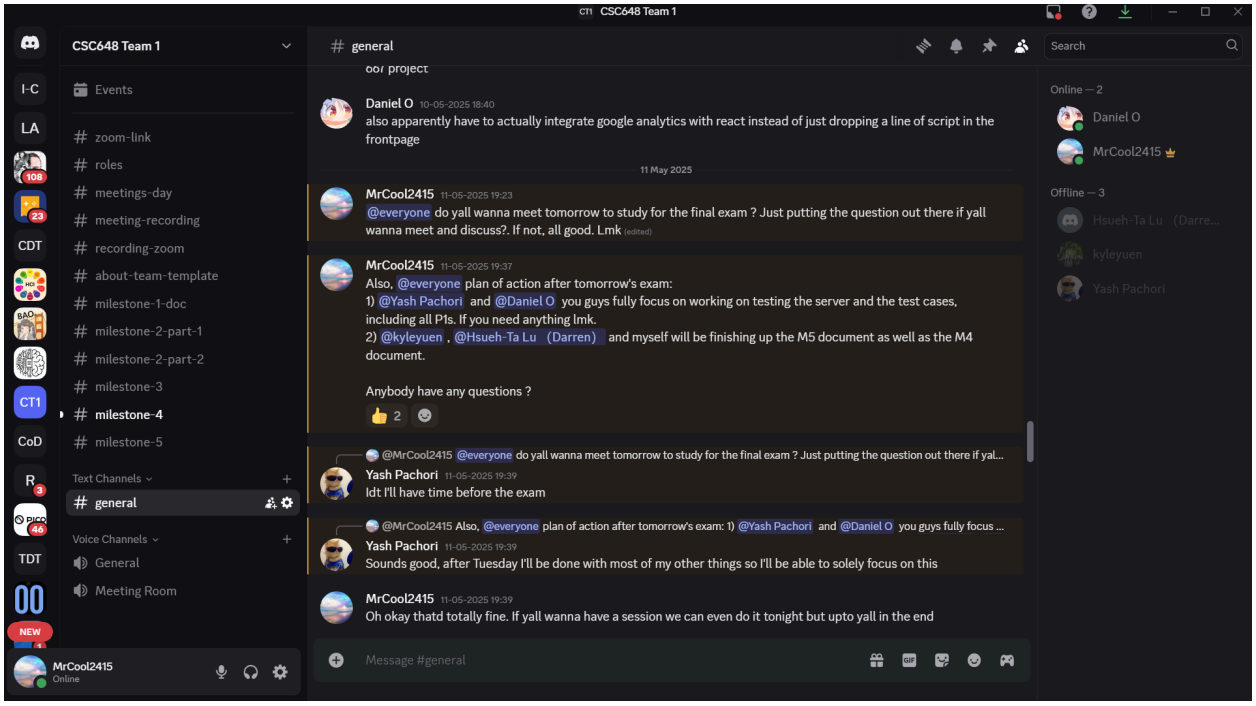


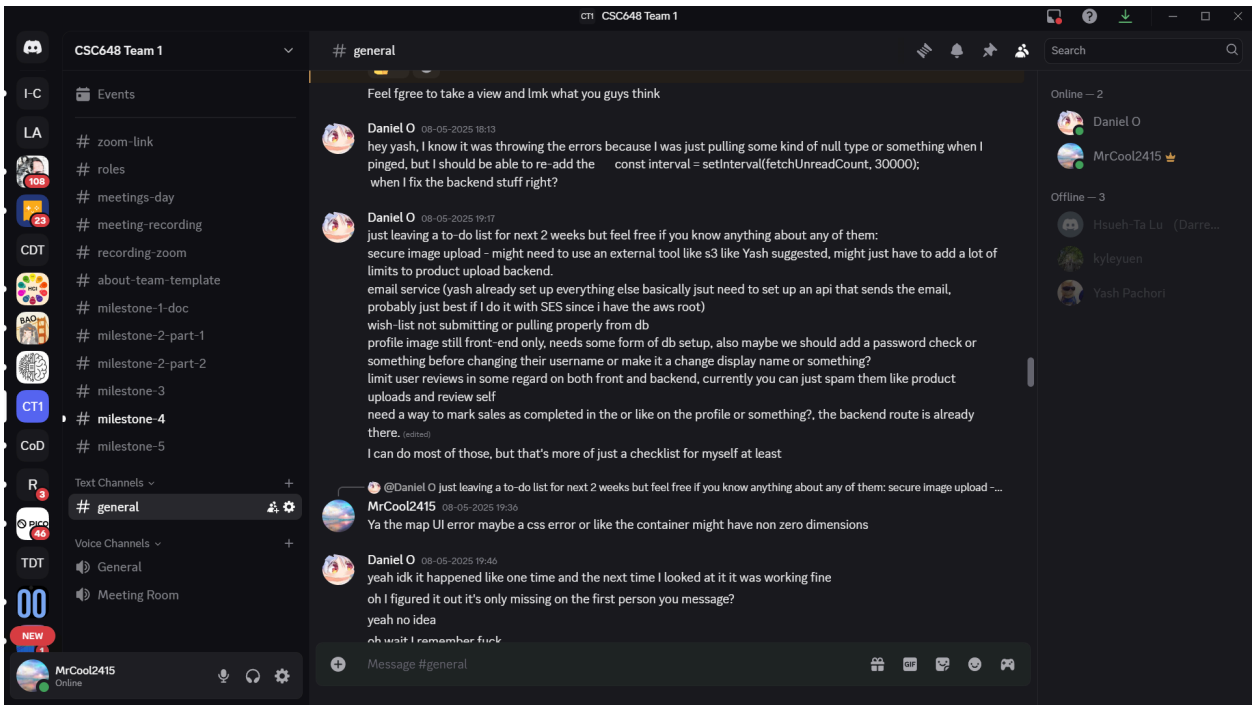

M5-doc
in list IN PROGRESS
Members
HL D KY DO YP
Notifications
Watching
Description
Aa
Make your description even better. Type '/' to insert content, formatting, and more.
Save
Cancel
Formatting help
To Do
Delete
0%
Make Sure all sub parts are complete in M-5
Make sure that everyone does a follow up to the self assessment email the team lead sent
Leave
Members
Labels
Checklist
Dates
Attachment
Cover
Custom Fields
Power-Ups
Add Power-Ups
Automation
Add button
Actions
Move
Copy

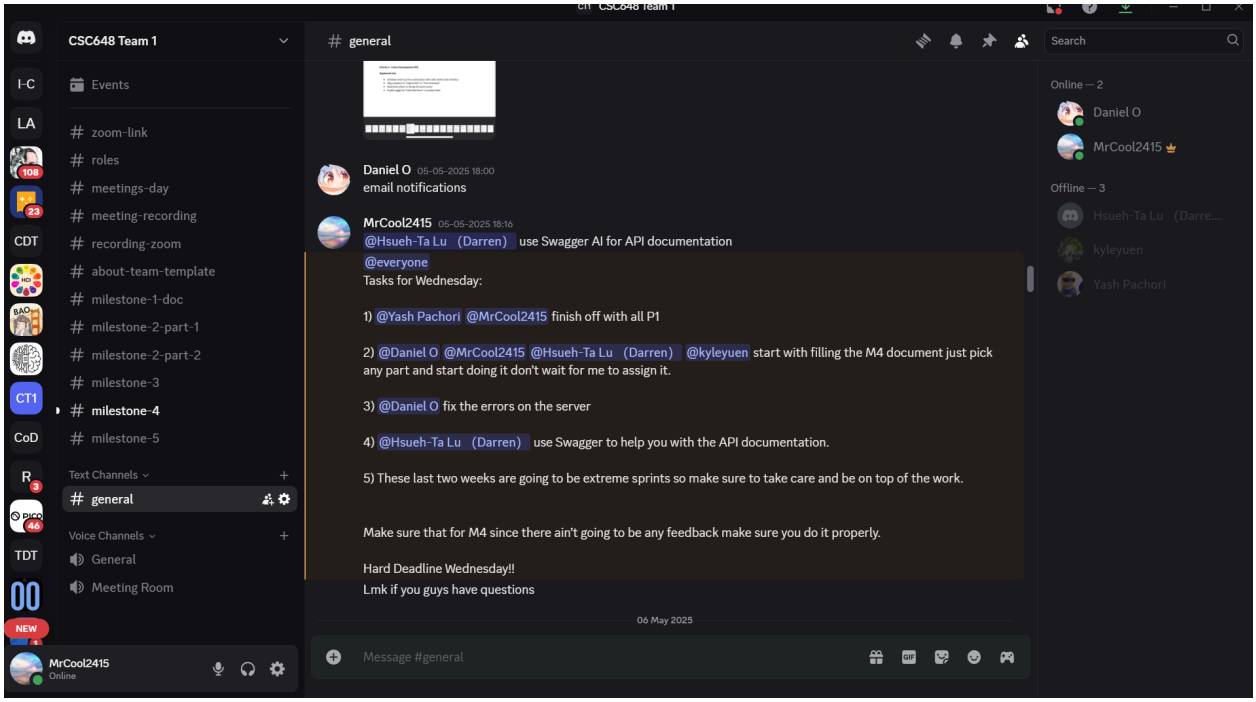


We also relied on Discord for communication and collaboration. It was our go-to tool for real-time chat, file sharing, and quick discussions. The channels in Discord allowed us to keep everything organized, with separate channels for different aspects of the project.

Discord:-

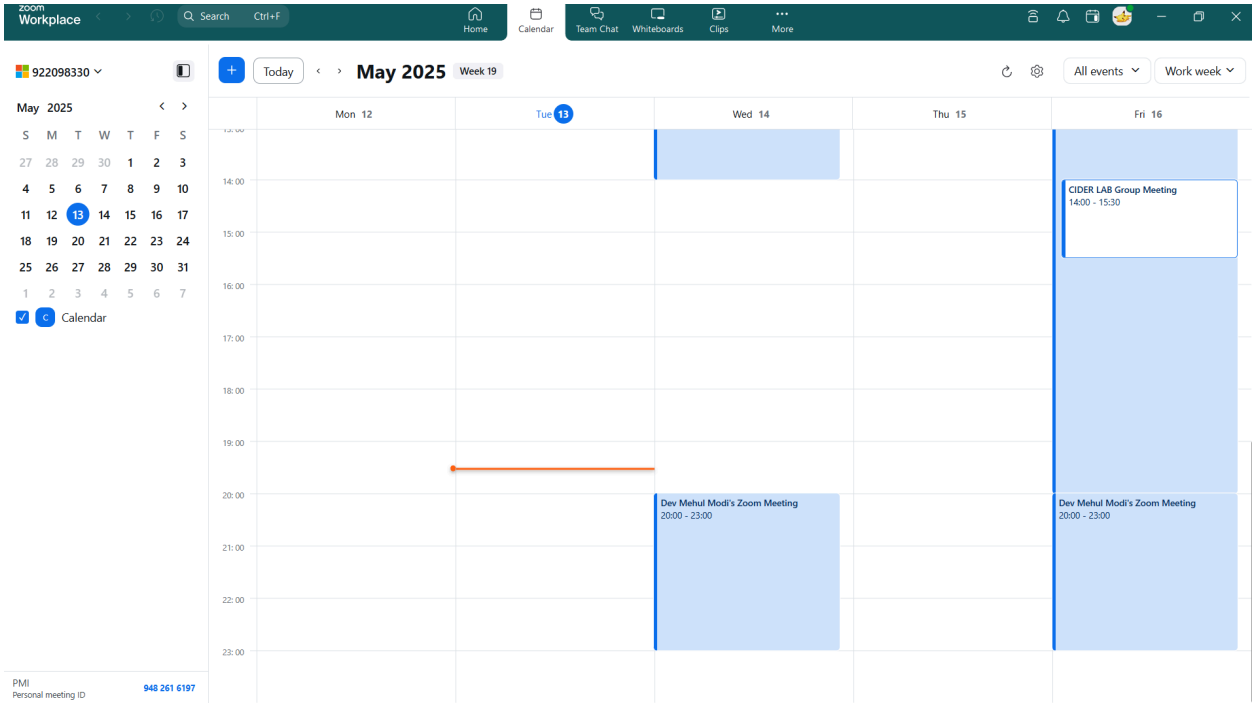

CSC648 Team 1
general
Search
Events
zoom-link
roles
meetings-day
meeting-recording
recording-zoom
about-team-template
milestone-1-doc
milestone-2-part-1
milestone-2-part-2
milestone-3
milestone-4
milestone-5
Text Channels
general
Voice Channels
General
Meeting Room
MrCool2415
Online
oor project
Daniel O 10-05-2025 18:40
also apparently have to actually integrate google analytics with react instead of just dropping a line of script in the frontpage
11 May 2025
MrCool2415 11-05-2025 19:23
@everyone do yall wanna meet tomorrow to study for the final exam ? Just putting the question out there if yall wanna meet and discuss?. If not, all good. Lmk (edited)
MrCool2415 11-05-2025 19:37
Also, @everyone plan of action after tomorrow's exam:
1) @Yash Pachori and @Daniel O you guys fully focus on working on testing the server and the test cases, including all P1s. If you need anything lmk.
2) @kyleyuen , @Hsueh-Ta Lu (Darren) and myself will be finishing up the M5 document as well as the M4 document.
Anybody have any questions ?
2
@MrCool2415 @everyone do yall wanna meet tomorrow to study for the final exam ? Just putting the question out there if yal...
Yash Pachori 11-05-2025 19:39
Idt I'll have time before the exam
@MrCool2415 Also, @everyone plan of action after tomorrow's exam: 1) @Yash Pachori and @Daniel O you guys fully focus ...
Yash Pachori 11-05-2025 19:39
Sounds good, after Tuesday I'll be done with most of my other things so I'll be able to solely focus on this
MrCool2415 11-05-2025 19:39
Oh okay thatd totally fine. If yall wanna have a session we can even do it tonight but upto yall in the end
Message #general
Online — 2
Daniel O
MrCool2415
Offline — 3
Hsueh-Ta Lu (Darre...
kyleyuen
Yash Pachori

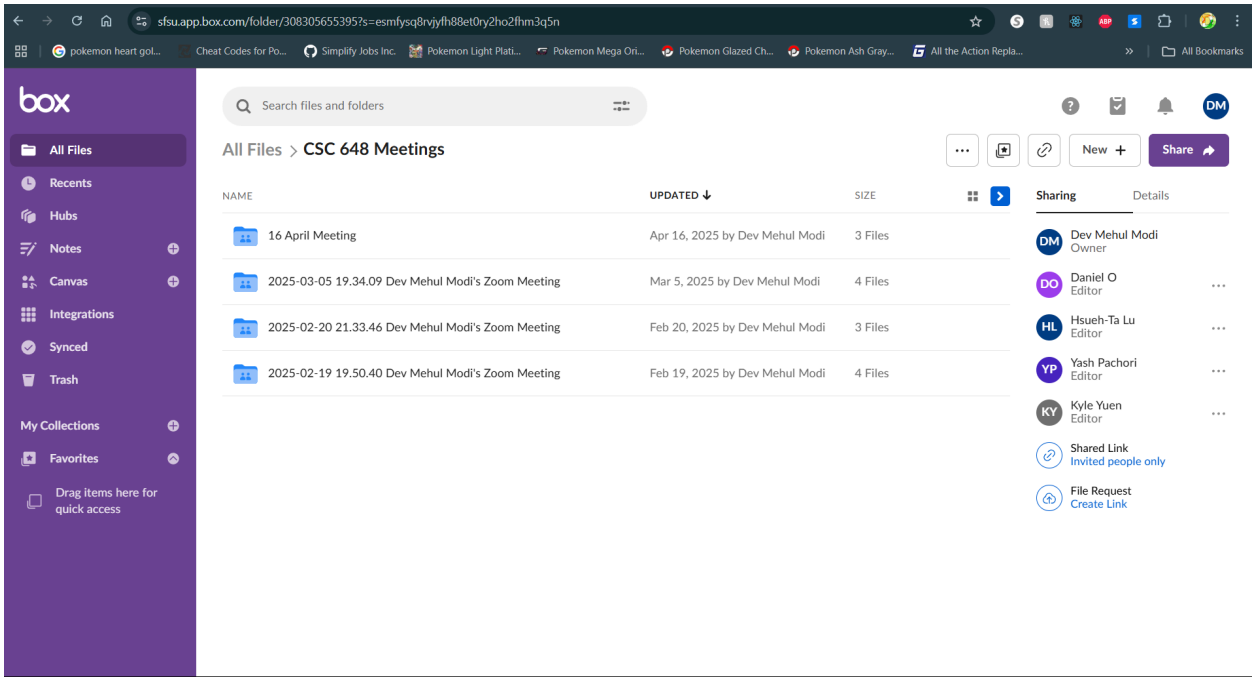

CSC648 Team 1
general
Search
Events
zoom-link
roles
meetings-day
meeting-recording
recording-zoom
about-team-template
milestone-1-doc
milestone-2-part-1
milestone-2-part-2
milestone-3
milestone-4
milestone-5
Text Channels
general
Voice Channels
General
Meeting Room
MrCool2415
Online
Feel fgree to take a view and lmk what you guys think
Daniel O 08-05-2025 18:13
hey yash, I know it was throwing the errors because I was just pulling some kind of null type or something when I pinged, but I should be able to re-add the const interval = setInterval(fetchUnreadCount, 30000); when I fix the backend stuff right?
Daniel O 08-05-2025 19:17
just leaving a to-do list for next 2 weeks but feel free if you know anything about any of them:
secure image upload - might need to use an external tool like s3 like Yash suggested, might just have to add a lot of limits to product upload backend.
email service (yash already set up everything else basically jsut need to set up an api that sends the email, probably just best if I do it with SES since i have the aws root)
wish-list not submitting or pulling properly from db
profile image still front-end only, needs some form of db setup, also maybe we should add a password check or something before changing their username or make it a change display name or something?
limit user reviews in some regard on both front and backend, currently you can just spam them like product uploads and review self
need a way to mark sales as completed in the or like on the profile or something?, the backend route is already there. (edited)
I can do most of those, but that's more of just a checklist for myself at least
@Daniel O just leaving a to-do list for next 2 weeks but feel free if you know anything about any of them: secure image upload -...
MrCool2415 08-05-2025 19:36
Ya the map UI error maybe a css error or like the container might have non zero dimensions
Daniel O 08-05-2025 19:46
yeah idk it happened like one time and the next time I looked at it it was working fine
oh I figured it out it's only missing on the first person you message?
yeah no idea
Message #general
Online — 2
Daniel O
MrCool2415
Offline — 3
Hsueh-Ta Lu (Darre...
kyleyuen
Yash Pachori



Additionally, we held regular Zoom meetings every Wednesday and Friday to discuss progress, address blockers, and plan the next steps. Additionally, all important meeting would be recorded and uploaded at Box SFSU. These meetings were crucial for maintaining alignment and ensuring the project stayed on track.

Zoom:-


zoom Workplace
Search Ctrl+F
Home
Calendar
Team Chat
Whiteboards
Clips
More
922098330
Today
May 2025
Week 19
All events
Work week
May 2025
Mon 12
Tue 13
Wed 14
Thu 15
Fri 16
CIDER LAB Group Meeting
14:00 - 15:30
Dev Mehul Modi's Zoom Meeting
20:00 - 23:00
Dev Mehul Modi's Zoom Meeting
20:00 - 23:00
Calendar
PMI
Personal meeting ID
948 261 6197


sfsu.app.box.com/folder/308305655395?s=esmfysq8rvjyfh88et0ry2ho2fhm3q5n
box
Search files and folders
All Files
Recents
Hubs
Notes
Canvas
Integrations
Synced
Trash
My Collections
Favorites
Drag items here for quick access
All Files > CSC 648 Meetings
New
Share
NAME
UPDATED
SIZE
16 April Meeting
Apr 16, 2025 by Dev Mehul Modi
3 Files
2025-03-05 19.34.09 Dev Mehul Modi's Zoom Meeting
Mar 5, 2025 by Dev Mehul Modi
4 Files
2025-02-20 21.33.46 Dev Mehul Modi's Zoom Meeting
Feb 20, 2025 by Dev Mehul Modi
3 Files
2025-02-19 19.50.40 Dev Mehul Modi's Zoom Meeting
Feb 19, 2025 by Dev Mehul Modi
4 Files
Sharing
Details
Dev Mehul Modi
Owner
Daniel O
Editor
Hsueh-Ta Lu
Editor
Yash Pachori
Editor
Kyle Yuen
Editor
Shared Link
Invited people only
File Request
Create Link

8) Team Member Self-Assessment & Contributions

Dev Modi (Team Lead)

Hi Team,

As part of the M5 final submission, here is my self-assessment and contribution summary for our CSC648 team project – Gator Market.

a) Contributions to the team project and teamwork:

- Acted as Team Lead for Team 1 – responsible for coordinating team efforts, managing task assignments, and ensuring milestone deadlines were met.
- Contributed significantly to both frontend and backend development:
    - Frontend: Implemented several core React components including UserProfile.jsx, ReviewPage.jsx, just to name a few and assisted in modernizing styles across pages like HomePage, UserProfile.
    - Backend: Helped debug and extend Flask-based routes in auth.py for features like adding an item to the wishlist, integrated review and wishlist system, and worked with Docker and Gunicorn setup for production deployment.
    - Database: Helped design and structure the Gator Market Database, outlining tables and their relationships to support application logic.
- Collaborated closely with all members to integrate features such as modals for product displays, messaging ideas, and resolving server-side errors.
- Acted as a communication bridge between frontend and backend teams to coordinate development efforts and minimize merge conflicts.
- Managed Trello tasks, organized Discord communications, led team meetings, and supported code reviews, GitHub workflow management, and conflict resolution.

b) GitHub submissions:

- Made frequent commits to the main branch throughout the semester across both frontend and backend directories.
- If my GitHub commit count appears lower than expected, it is due to prioritizing larger batch commits after completing and validating multi-feature work locally before pushing.

c) Main challenges encountered:

- Balancing team availability and syncing schedules was often difficult, especially during the middle of the semester.
- Managing full-stack coordination between React and Flask required careful debugging and constant communication.
- Integrating features while avoiding merge conflicts and maintaining a clean Git history was a recurring challenge.

d) Experience with GenAI:

- Used GenAI (ChatGPT) extensively to assist with:
    - Drafting frontend component logic and React state management patterns.
    - Debugging Flask routes and form validation logic.
    - Resolving Cors Errors in server configurations.
    - Writing SQL queries for MySQL database operations.
- GenAI was particularly helpful for quick syntax checks and exploring new approaches during development.

e) What I would do better next time:

- Introduce sprint-based task planning earlier in the semester with clear deadlines.
- Hold more structured check-ins to proactively resolve blockers.
- Add more unit and integration tests to verify feature robustness.
- Look for User Review (Provide the demo to as many people as possible to test and provide feedback).
- Push for more early testing of key features to reduce integration pressure in later phases.

f) Additional notes:

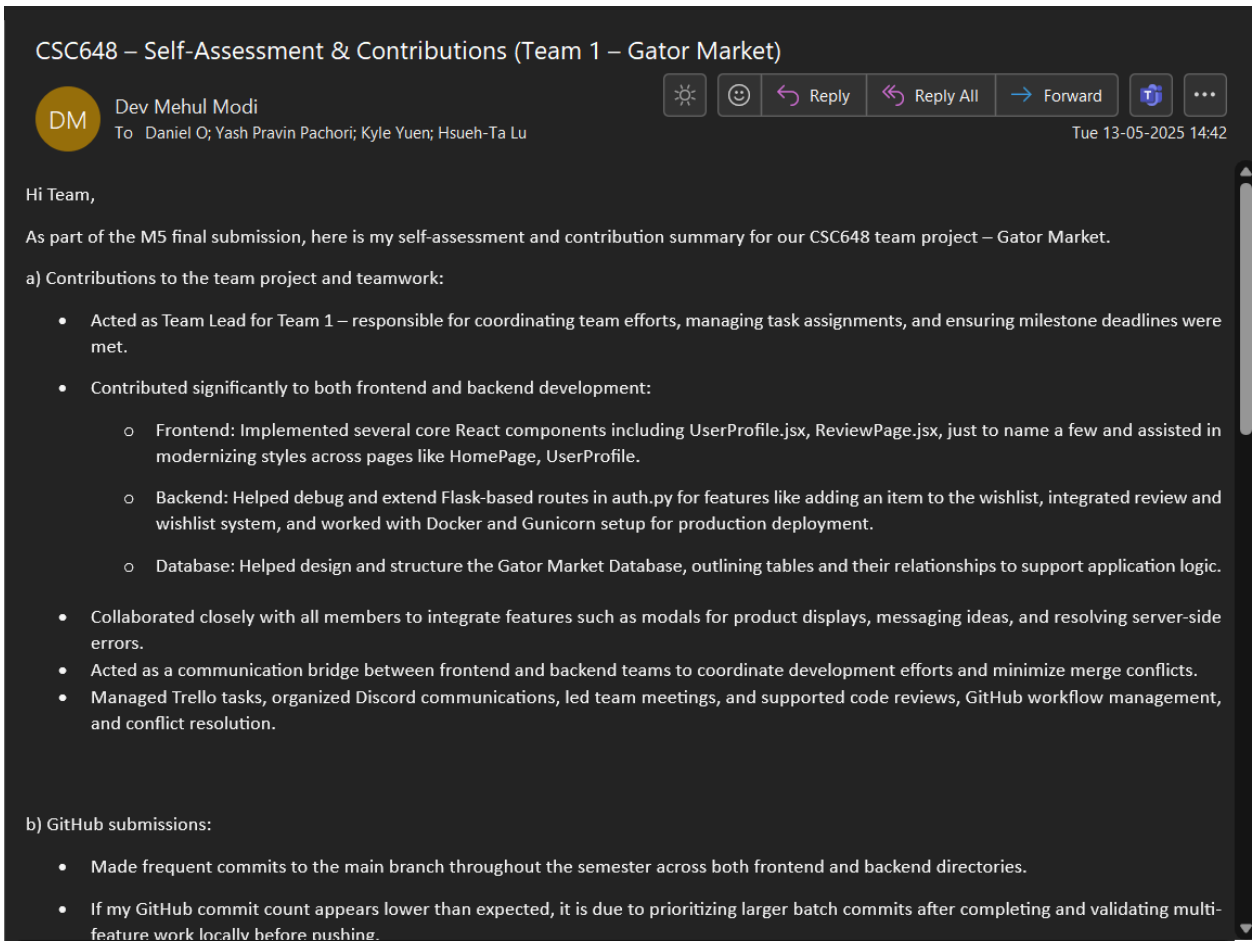
- I deeply appreciate everyone’s collaboration. My focus was not just technical contribution, but also ensuring we worked as a cohesive team. Everyone was very respectful and helpful to each other and the working environment was kept very professional which I greatly appreciate.
- This experience has strengthened my leadership, communication, and full-stack development skills.

g) Team Lead Section – Overall Feedback:

- As the team lead, I observed strong participation and engagement from each member, and I will summarize specific team member feedback in the team lead section of the final submission as required.
- Items c), d), and e) above were discussed from my viewpoint as both contributor and team lead.

Best,
Dev Modi

Team Lead – Team 1 (Gator Market)



CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

DM

Dev Mehul Modi
To Daniel O; Yash Pravin Pachori; Kyle Yuen; Hsueh-Ta Lu

Reply | Reply All | Forward

Tue 13-05-2025 14:42

Hi Team,

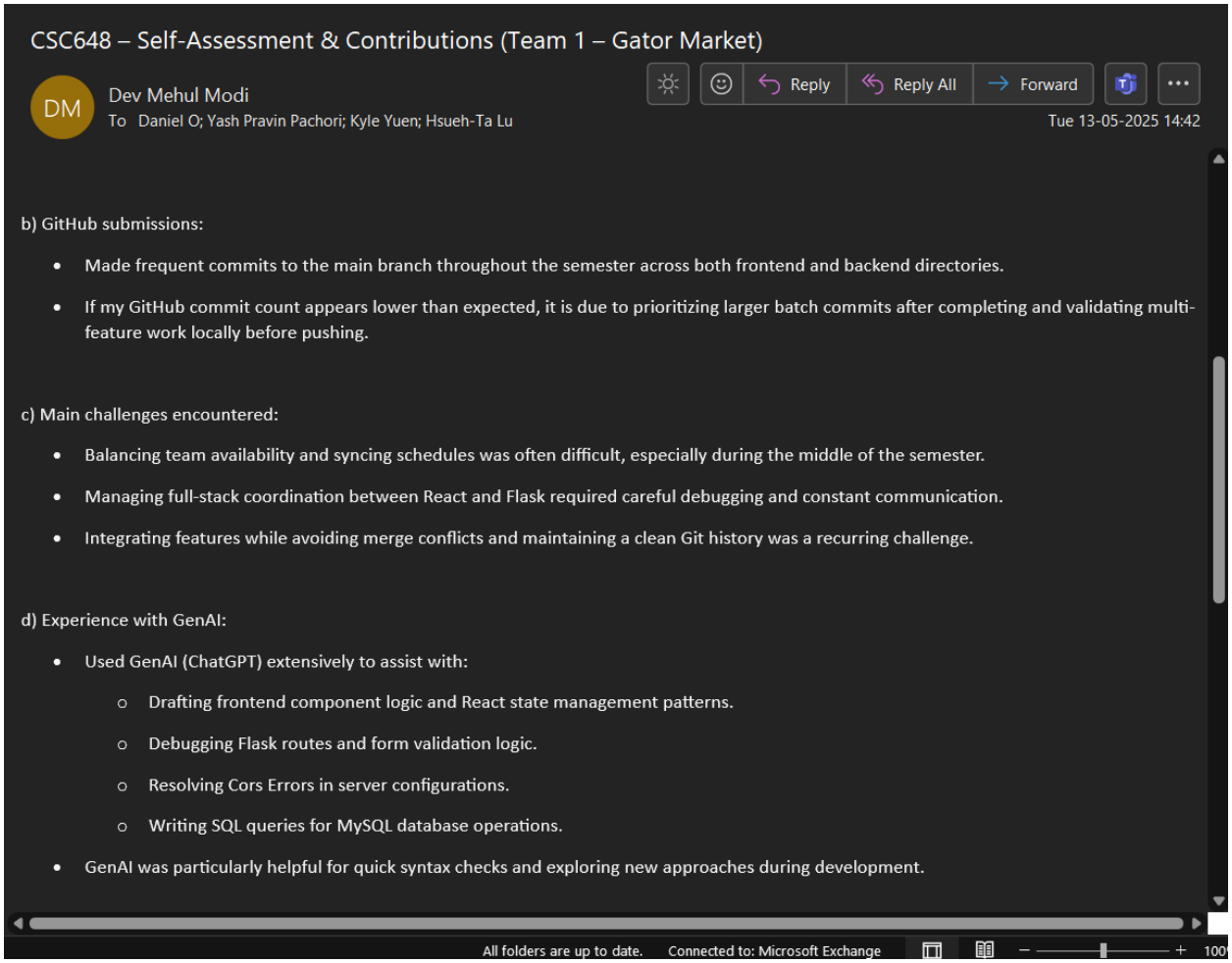
As part of the M5 final submission, here is my self-assessment and contribution summary for our CSC648 team project – Gator Market.

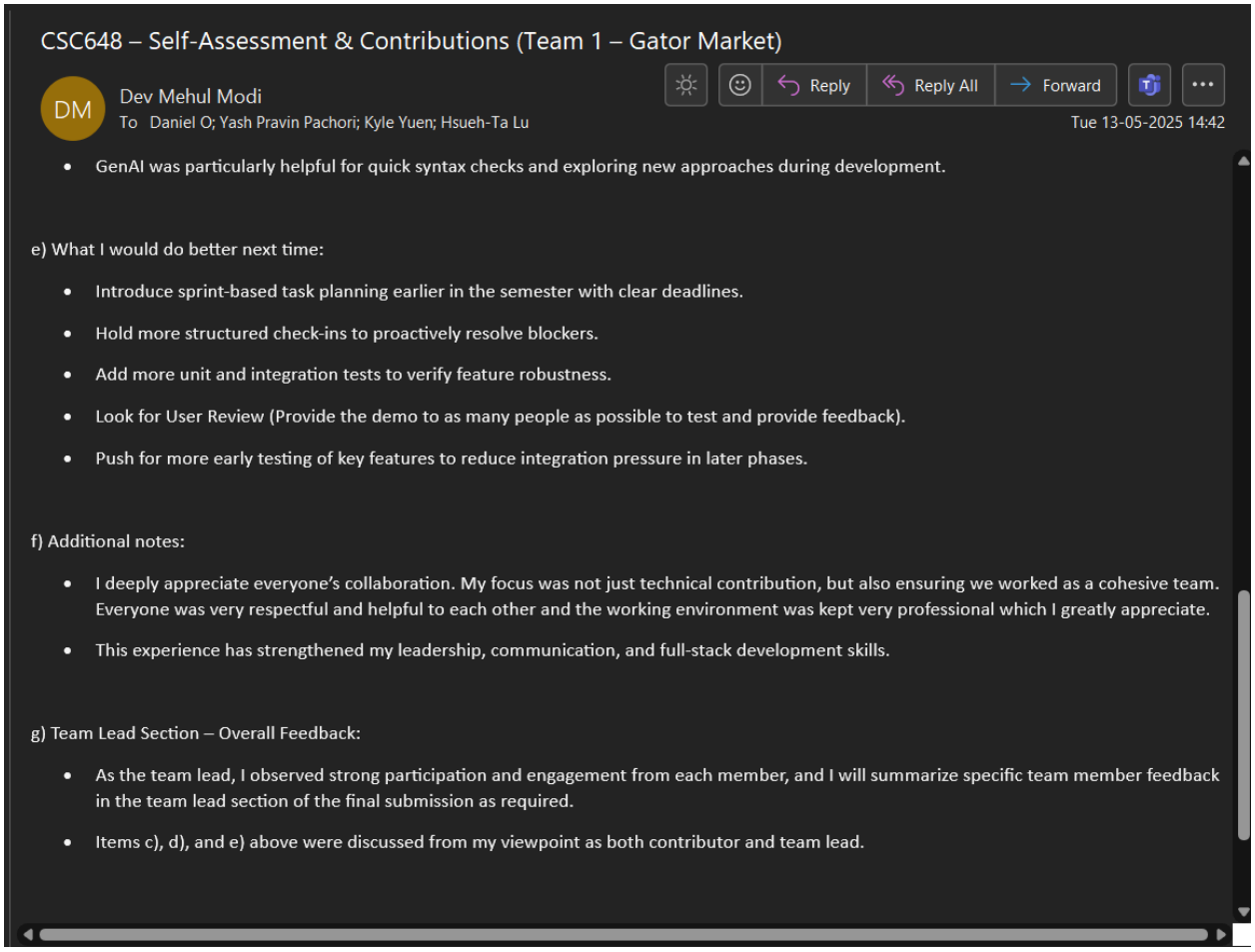
a) Contributions to the team project and teamwork:

- Acted as Team Lead for Team 1 – responsible for coordinating team efforts, managing task assignments, and ensuring milestone deadlines were met.
- Contributed significantly to both frontend and backend development:
  - Frontend: Implemented several core React components including UserProfile.jsx, ReviewPage.jsx, just to name a few and assisted in modernizing styles across pages like HomePage, UserProfile.
  - Backend: Helped debug and extend Flask-based routes in auth.py for features like adding an item to the wishlist, integrated review and wishlist system, and worked with Docker and Gunicorn setup for production deployment.
  - Database: Helped design and structure the Gator Market Database, outlining tables and their relationships to support application logic.
- Collaborated closely with all members to integrate features such as modals for product displays, messaging ideas, and resolving server-side errors.
- Acted as a communication bridge between frontend and backend teams to coordinate development efforts and minimize merge conflicts.
- Managed Trello tasks, organized Discord communications, led team meetings, and supported code reviews, GitHub workflow management, and conflict resolution.

b) GitHub submissions:

- Made frequent commits to the main branch throughout the semester across both frontend and backend directories.
- If my GitHub commit count appears lower than expected, it is due to prioritizing larger batch commits after completing and validating multi-feature work locally before pushing.


CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)
DM
Dev Mehul Modi
To Daniel O; Yash Pravin Pachori; Kyle Yuen; Hsueh-Ta Lu
Reply
Reply All
Forward
Tue 13-05-2025 14:42
b) GitHub submissions:
Made frequent commits to the main branch throughout the semester across both frontend and backend directories.
If my GitHub commit count appears lower than expected, it is due to prioritizing larger batch commits after completing and validating multi-feature work locally before pushing.
c) Main challenges encountered:
Balancing team availability and syncing schedules was often difficult, especially during the middle of the semester.
Managing full-stack coordination between React and Flask required careful debugging and constant communication.
Integrating features while avoiding merge conflicts and maintaining a clean Git history was a recurring challenge.
d) Experience with GenAI:
Used GenAI (ChatGPT) extensively to assist with:
Drafting frontend component logic and React state management patterns.
Debugging Flask routes and form validation logic.
Resolving Cors Errors in server configurations.
Writing SQL queries for MySQL database operations.
GenAI was particularly helpful for quick syntax checks and exploring new approaches during development.
All folders are up to date.
Connected to: Microsoft Exchange



CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

DM

Dev Mehul Modi

To Daniel O; Yash Pravin Pachori; Kyle Yuen; Hsueh-Ta Lu

Reply | Reply All | Forward

Tue 13-05-2025 14:42

- GenAI was particularly helpful for quick syntax checks and exploring new approaches during development.

e) What I would do better next time:

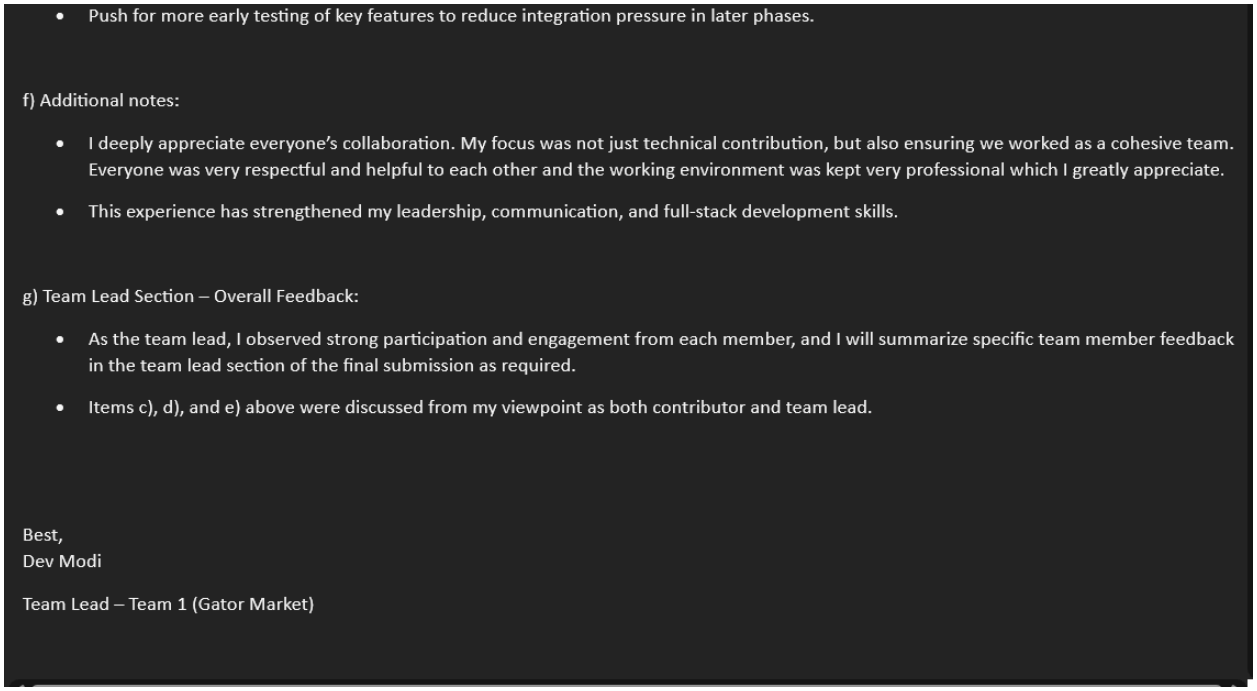
- Introduce sprint-based task planning earlier in the semester with clear deadlines.
- Hold more structured check-ins to proactively resolve blockers.
- Add more unit and integration tests to verify feature robustness.
- Look for User Review (Provide the demo to as many people as possible to test and provide feedback).
- Push for more early testing of key features to reduce integration pressure in later phases.

f) Additional notes:

- I deeply appreciate everyone's collaboration. My focus was not just technical contribution, but also ensuring we worked as a cohesive team. Everyone was very respectful and helpful to each other and the working environment was kept very professional which I greatly appreciate.
- This experience has strengthened my leadership, communication, and full-stack development skills.

g) Team Lead Section – Overall Feedback:

- As the team lead, I observed strong participation and engagement from each member, and I will summarize specific team member feedback in the team lead section of the final submission as required.
- Items c), d), and e) above were discussed from my viewpoint as both contributor and team lead.



- Push for more early testing of key features to reduce integration pressure in later phases.

f) Additional notes:

- I deeply appreciate everyone's collaboration. My focus was not just technical contribution, but also ensuring we worked as a cohesive team. Everyone was very respectful and helpful to each other and the working environment was kept very professional which I greatly appreciate.
- This experience has strengthened my leadership, communication, and full-stack development skills.

g) Team Lead Section – Overall Feedback:

- As the team lead, I observed strong participation and engagement from each member, and I will summarize specific team member feedback in the team lead section of the final submission as required.
- Items c), d), and e) above were discussed from my viewpoint as both contributor and team lead.

Best,
Dev Modi

Team Lead – Team 1 (Gator Market)

Hsueh-Ta Lu (Scrum Master and Frontend)

Hi team,
Here's my self-assessment and contribution summary for our Gator Market project (Milestone 5).

a) My contributions (technical and non-technical):

- Development of the product upload feature, including image preview, field validation, and success/error feedback
- Built and refined the wishlist (previously "cart") functionality, including UI display and interaction logic
- Contributed UI updates for the homepage buttons and About Us page
- Integrated map and meetup time components into the messaging modal
- Made small styling and usability enhancements throughout the site
- Helped write milestone documents with teammates and tested the site on Safari (Mac) to make sure it worked smoothly.

b) GitHub submissions:

I made approximately 24 commits, mostly focused on front-end components related to uploading, wishlist, styling improvements, and shared UI layouts.

c) Challenges faced:

One of the main challenges I encountered was integrating the map and meetup time selection into the chat interface without making the UI too cluttered or confusing. It required careful coordination between layout design and data handling. Another major challenge was implementing the product upload flow, especially getting image previews, field validation, and error handling to work reliably across edge cases. These components needed to be intuitive and resilient, which took several iterations and testing rounds to refine.

d) GenAI experience:

I used GenAI tools like ChatGPT regularly throughout the project to support planning, writing, debugging, and improving UI structure and communication. It helped streamline the process, reduce writer's block during documentation, and speed up the development of ideas that we later refined manually.

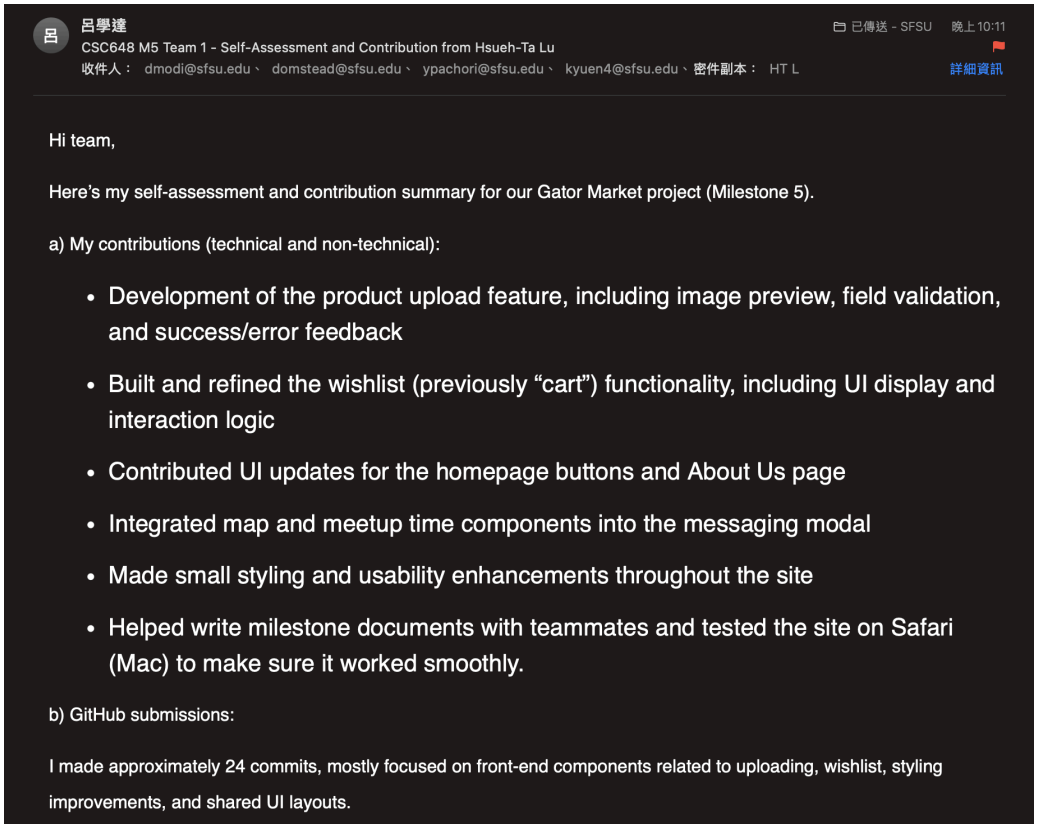
e) What I would do differently next time:

Next time, I would focus on clearer feature breakdowns across sprints and proactively align component dependencies earlier in the process. I also think documenting UI changes more consistently in GitHub commits would improve team communication.

Thanks again, everyone, for the collaboration throughout the semester!

Best,

Hsueh-Ta Lu



呂學達
已傳送 - SFSU 晚上10:11
CSC648 M5 Team 1 - Self-Assessment and Contribution from Hsueh-Ta Lu
收件人： dmodi@sfsu.edu、 domstead@sfsu.edu、 ypachori@sfsu.edu、 kyuen4@sfsu.edu、密件副本： HT L
詳細資訊

Hi team,

Here's my self-assessment and contribution summary for our Gator Market project (Milestone 5).

a) My contributions (technical and non-technical):

- Development of the product upload feature, including image preview, field validation, and success/error feedback
- Built and refined the wishlist (previously "cart") functionality, including UI display and interaction logic
- Contributed UI updates for the homepage buttons and About Us page
- Integrated map and meetup time components into the messaging modal
- Made small styling and usability enhancements throughout the site
- Helped write milestone documents with teammates and tested the site on Safari (Mac) to make sure it worked smoothly.

b) GitHub submissions:

I made approximately 24 commits, mostly focused on front-end components related to uploading, wishlist, styling improvements, and shared UI layouts.

c) Challenges faced:

One of the main challenges I encountered was integrating the map and meetup time selection into the chat interface without making the UI too cluttered or confusing. It required careful coordination between layout design and data handling. Another major challenge was implementing the product upload flow, especially getting image previews, field validation, and error handling to work reliably across edge cases. These components needed to be intuitive and resilient, which took several iterations and testing rounds to refine.

d) GenAI experience:

I used GenAI tools like ChatGPT regularly throughout the project to support planning, writing, debugging, and improving UI structure and communication. It helped streamline the process, reduce writer's block during documentation, and speed up the development of ideas that we later refined manually.

e) What I would do differently next time:

Next time, I would focus on clearer feature breakdowns across sprints and proactively align component dependencies earlier in the process. I also think documenting UI changes more consistently in GitHub commits would improve team communication.

Thanks again, everyone, for the collaboration throughout the semester!

Best,

Hsueh-Ta Lu

Kyle Yuen (Frontend lead)

Hi team,

Here's my self-assessment and contribution summary for our Gator Market project (Milestone 5).

a) My contributions (technical and non-technical):

I contributed across various parts of the project, especially on the front-end. Some specific contributions include:

- Worked on front-end components alongside Hsueh-Ta Lu
- Contributed to the layout and styling of the About Us and homepage sections
- Assisted in designing UI components and improving overall usability
- Helped plan the map and pickup process feature
- Provided support on the login and registration page functionality
- Acted as Scrum Master with Hsueh-Ta Lu during 2 few meetings and took meeting notes
- Participated in writing milestone documentation and collaborated on editing

b) GitHub submissions:

I made roughly 30 commits throughout the development cycle. These commits covered:

- Styling adjustments
- UI layout improvements
- Edits to shared pages like About Us
- Small logic updates for login/registration components

c) Challenges faced:

The biggest challenge for me was learning while building. A lot of the work required skills I was still developing, so balancing learning new concepts and applying them in a real project was tough but rewarding. Debugging issues also took time, and when I didn't understand something, I reached out to our team lead for clarification and support, which helped a lot.

d) GenAI experience:

I used ChatGPT to help with:

- Styling suggestions
- UI layout ideas
- Brainstorming features
  I often started with my concepts and then asked ChatGPT to build on them. I'd then merge the results with my original ideas and bring them back to the team for feedback or integration. It really helped spark creativity and accelerate design planning.

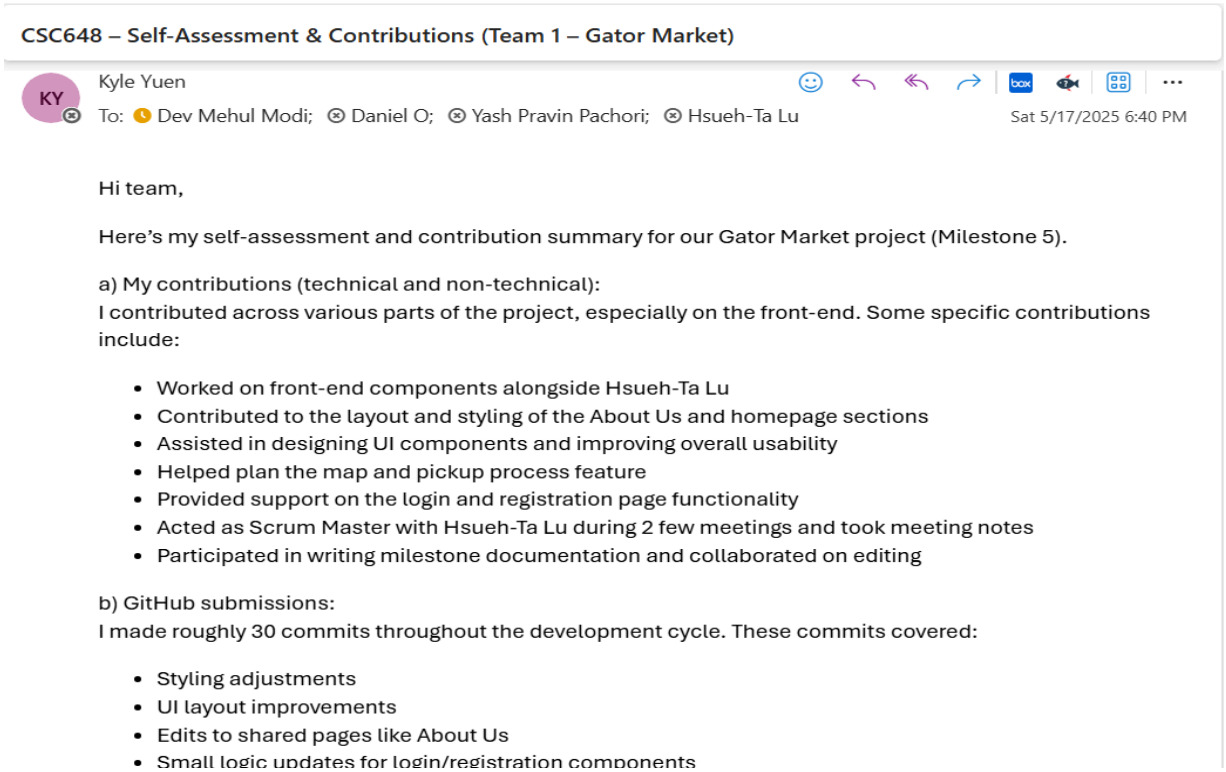
e) What I would do differently next time:

Next time, I would focus more on early planning to help myself learn and absorb new concepts more effectively. Having a clearer structure would allow me to better connect what I'm learning to the work we're doing and boost both understanding and productivity.

Thanks to everyone on the team for a nice semester and a fun group to work with!

Best,

Kyle Yuen



CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

KY Kyle Yuen

To: Dev Mehul Modi; Daniel O; Yash Pravin Pachori; Hsueh-Ta Lu Sat 5/17/2025 6:40 PM

Hi team,

Here's my self-assessment and contribution summary for our Gator Market project (Milestone 5).

a) My contributions (technical and non-technical):
I contributed across various parts of the project, especially on the front-end. Some specific contributions include:

- Worked on front-end components alongside Hsueh-Ta Lu
- Contributed to the layout and styling of the About Us and homepage sections
- Assisted in designing UI components and improving overall usability
- Helped plan the map and pickup process feature
- Provided support on the login and registration page functionality
- Acted as Scrum Master with Hsueh-Ta Lu during 2 few meetings and took meeting notes
- Participated in writing milestone documentation and collaborated on editing

b) GitHub submissions:
I made roughly 30 commits throughout the development cycle. These commits covered:

- Styling adjustments
- UI layout improvements
- Edits to shared pages like About Us
- Small logic updates for login/registration components

CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

b) GitHub submissions:
I made roughly 30 commits throughout the development cycle. These commits covered:

- Styling adjustments
- UI layout improvements
- Edits to shared pages like About Us
- Small logic updates for login/registration components

c) Challenges faced:
The biggest challenge for me was learning while building. A lot of the work required skills I was still developing, so balancing learning new concepts and applying them in a real project was tough but rewarding. Debugging issues also took time, and when I didn't understand something, I reached out to our team lead for clarification and support, which helped a lot.

d) GenAI experience:
I used ChatGPT to help with:

- Styling suggestions
- UI layout ideas
- Brainstorming features
  I often started with my concepts and then asked ChatGPT to build on them. I'd then merge the results with my original ideas and bring them back to the team for feedback or integration. It really helped spark creativity and accelerate design planning.

CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

d) GenAI experience:
I used ChatGPT to help with:

- Styling suggestions
- UI layout ideas
- Brainstorming features
  I often started with my concepts and then asked ChatGPT to build on them. I'd then merge the results with my original ideas and bring them back to the team for feedback or integration. It really helped spark creativity and accelerate design planning.

e) What I would do differently next time:
Next time, I would focus more on early planning to help myself learn and absorb new concepts more effectively. Having a clearer structure would allow me to better connect what I'm learning to the work we're doing and boost both understanding and productivity.

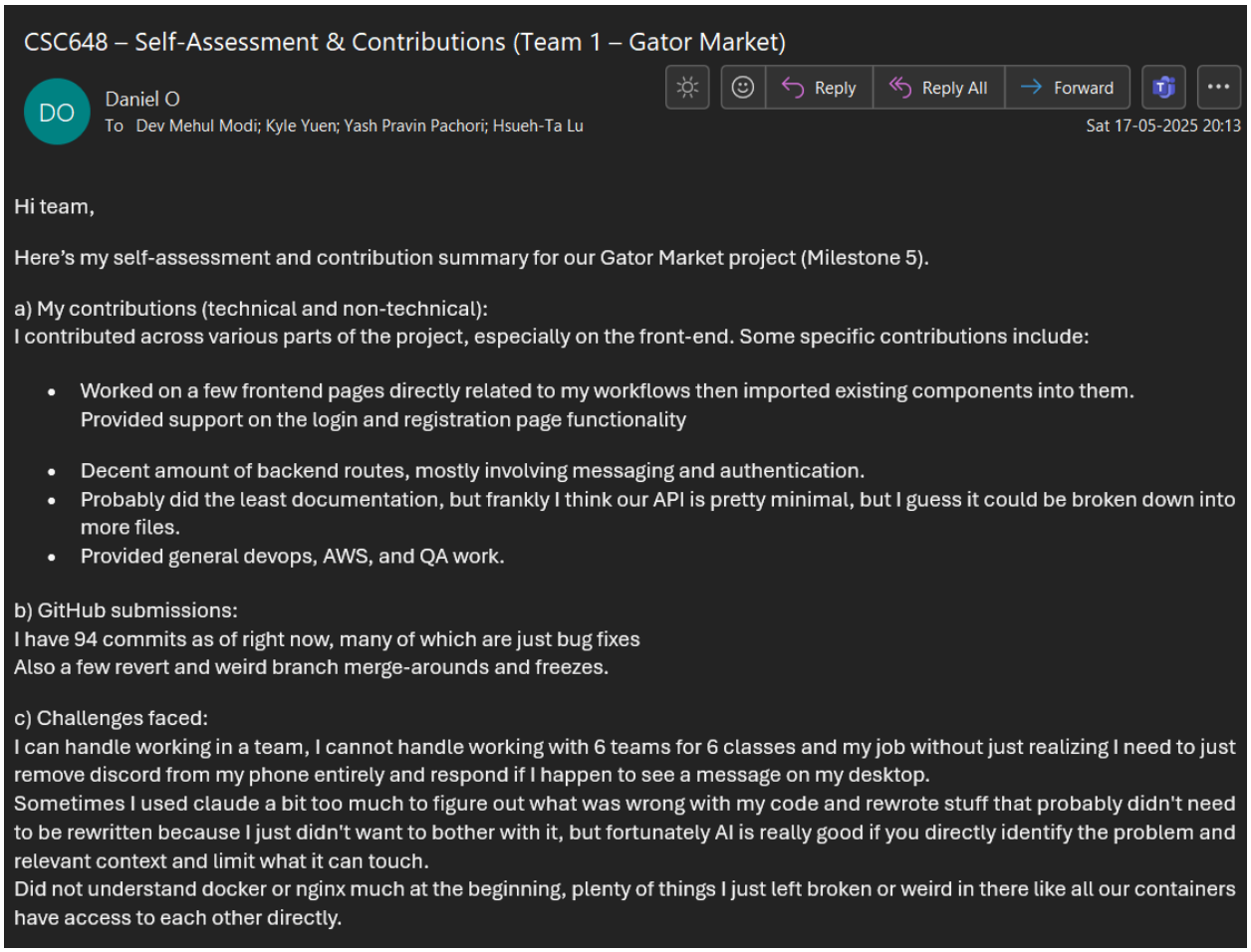
Thanks to everyone on the team for a nice semester and a fun group to work with!

Best,
Kyle Yuen

...

Reply | Reply all | Forward

Daniel Omstead (Backend and Github Master)



CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

DO

Daniel O

To Dev Mehul Modi; Kyle Yuen; Yash Pravin Pachori; Hsueh-Ta Lu

Reply | Reply All | Forward

Sat 17-05-2025 20:13

Hi team,

Here's my self-assessment and contribution summary for our Gator Market project (Milestone 5).

a) My contributions (technical and non-technical):
I contributed across various parts of the project, especially on the front-end. Some specific contributions include:

- Worked on a few frontend pages directly related to my workflows then imported existing components into them. Provided support on the login and registration page functionality
- Decent amount of backend routes, mostly involving messaging and authentication.
- Probably did the least documentation, but frankly I think our API is pretty minimal, but I guess it could be broken down into more files.
- Provided general devops, AWS, and QA work.

b) GitHub submissions:
I have 94 commits as of right now, many of which are just bug fixes
Also a few revert and weird branch merge-arounds and freezes.

c) Challenges faced:
I can handle working in a team, I cannot handle working with 6 teams for 6 classes and my job without just realizing I need to just remove discord from my phone entirely and respond if I happen to see a message on my desktop.
Sometimes I used claude a bit too much to figure out what was wrong with my code and rewrote stuff that probably didn't need to be rewritten because I just didn't want to bother with it, but fortunately AI is really good if you directly identify the problem and relevant context and limit what it can touch.
Did not understand docker or nginx much at the beginning, plenty of things I just left broken or weird in there like all our containers have access to each other directly.

CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

DO

Daniel O

To Dev Mehul Modi; Kyle Yuen; Yash Pravin Pachori; Hsueh-Ta Lu

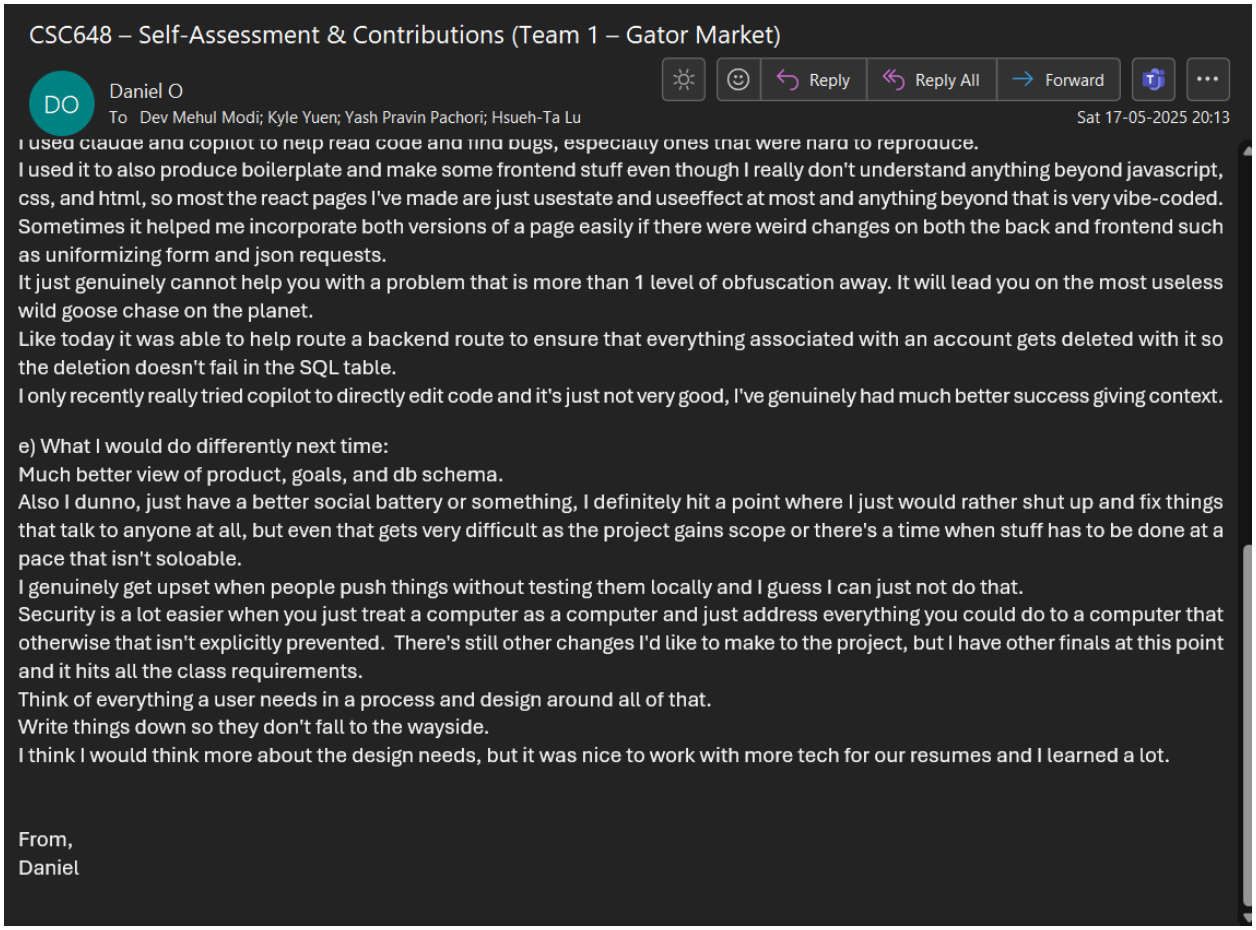
Reply | Reply All | Forward

Sat 17-05-2025 20:13

Did not understand docker or nginx much at the beginning, plenty of things I just left broken or weird in there like all our containers have access to each other directly.

d) GenAI experience:
I used claude and copilot to help read code and find bugs, especially ones that were hard to reproduce.
I used it to also produce boilerplate and make some frontend stuff even though I really don't understand anything beyond javascript, css, and html, so most the react pages I've made are just usestate and useeffect at most and anything beyond that is very vibe-coded.
Sometimes it helped me incorporate both versions of a page easily if there were weird changes on both the back and frontend such as uniformizing form and json requests.
It just genuinely cannot help you with a problem that is more than 1 level of obfuscation away. It will lead you on the most useless wild goose chase on the planet.
Like today it was able to help route a backend route to ensure that everything associated with an account gets deleted with it so the deletion doesn't fail in the SQL table.
I only recently really tried copilot to directly edit code and it's just not very good, I've genuinely had much better success giving context.

e) What I would do differently next time:
Much better view of product, goals, and db schema.
Also I dunno, just have a better social battery or something, I definitely hit a point where I just would rather shut up and fix things that talk to anyone at all, but even that gets very difficult as the project gains scope or there's a time when stuff has to be done at a pace that isn't soloable.
I genuinely get upset when people push things without testing them locally and I guess I can just not do that.
Security is a lot easier when you just treat a computer as a computer and just address everything you could do to a computer that otherwise that isn't explicitly prevented. There's still other changes I'd like to make to the project, but I have other finals at this point and it hits all the class requirements.
Think of everything a user needs in a process and design around all of that.
Write things down so they don't fall to the wayside.
I think I would think more about the design needs, but it was nice to work with more tech for our resumes and I learned a lot.



CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

DO

Daniel O

To Dev Mehul Modi; Kyle Yuen; Yash Pravin Pachori; Hsueh-Ta Lu

Reply Reply All Forward

Sat 17-05-2025 20:13

I used claude and copilot to help read code and find bugs, especially ones that were hard to reproduce.
I used it to also produce boilerplate and make some frontend stuff even though I really don't understand anything beyond javascript, css, and html, so most the react pages I've made are just usestate and useeffect at most and anything beyond that is very vibe-coded.
Sometimes it helped me incorporate both versions of a page easily if there were weird changes on both the back and frontend such as uniformizing form and json requests.
It just genuinely cannot help you with a problem that is more than 1 level of obfuscation away. It will lead you on the most useless wild goose chase on the planet.
Like today it was able to help route a backend route to ensure that everything associated with an account gets deleted with it so the deletion doesn't fail in the SQL table.
I only recently really tried copilot to directly edit code and it's just not very good, I've genuinely had much better success giving context.

e) What I would do differently next time:
Much better view of product, goals, and db schema.
Also I dunno, just have a better social battery or something, I definitely hit a point where I just would rather shut up and fix things that talk to anyone at all, but even that gets very difficult as the project gains scope or there's a time when stuff has to be done at a pace that isn't soloable.
I genuinely get upset when people push things without testing them locally and I guess I can just not do that.
Security is a lot easier when you just treat a computer as a computer and just address everything you could do to a computer that otherwise that isn't explicitly prevented. There's still other changes I'd like to make to the project, but I have other finals at this point and it hits all the class requirements.
Think of everything a user needs in a process and design around all of that.
Write things down so they don't fall to the wayside.
I think I would think more about the design needs, but it was nice to work with more tech for our resumes and I learned a lot.

From,
Daniel

Yash Pachori (Backend Lead and Database Organizer)

Hi Team,

Here is my self-assessment and contribution summary for our CSC648 project — Gator Market:

1.) Contributions to the team project and teamwork

- Acted as the lead backend developer — wrote API's and routes for the project, ensured that all the functions and features were implemented and worked in the backend.
- Created and maintained the database throughout the project
- Created and maintained several frontend components and made sure all features work throughout all levels of the code
- Set up docker for the project so that project could be run with containers
- Helped write various sections of the milestone documents

2.) GitHub Submissions

- I made 52 commits. My commits were low at first because I would focus on doing larger scale commits where lots of work was done in each commit. But later on in the project I decided to do smaller more frequent commits so that I could revert back to a working version of the code if I ever needed to

3.) Main challenges encountered

- The database caused a lot of issues for a while. At first the plan was to do development on a Ubuntu virtual machine, but the VM stopped working so I had to switch to developing on my host laptop. I created the database on the host laptop using my personal MySQL account which caused issues later with login on teammate computers because it had saved my login, so I had to delete the database and recreate it and this time let docker handle it. I created a user for the database so that everyone will be able to access the database if needed
- There were challenges with the backend and with exposing the ports. Originally, we had the backend on port 8000 but we realized that whenever we needed to update the AWS server, we would have to change the port number. So later we decided to have the backend on port 8001 but we were pretty deep into development at that point and so we would face a lot of issues where the backend wouldn't connect successfully because we forgot to change the port number somewhere in the code. We also faced a lot of issues with axios and nginx, but Daniel was able to figure those out and handle them

- One issue I specifically faced was with product uploads. Whenever a user uploaded a product, it wouldn't show up in the products table of the database. I found out that it was because the backend route was never being hit which meant that the database would never be updated, so to fix this I had to do a lot of troubleshooting with the backend and rewrite many sections of the code. I also had to rewrite a few sections of the database but in the end I was able to get it working properly

4.) Use of GenAI

- I used GenAI extensively throughout this project. I mainly used ChatGPT but sometimes I used Claude because ChatGPT would make a lot of errors. GenAI was really helpful because I could ask it to help me with debugging code. Sometimes I wouldn't be able to figure out why something wasn't working and ChatGPT would tell me it's just because of a simple syntax error
- Another way I used it is to help me brainstorm how I can implement a certain thing or brainstorm different ways I can approach a problem and the pros and cons of each path
- I also asked it to help me with SQL queries because I wasn't able to remember them sometimes
- ChatGPT helped me setup Docker because this was my first time ever using it and it was really helpful. I would ask it for different commands I can use and how I would be able to make the database inside Docker

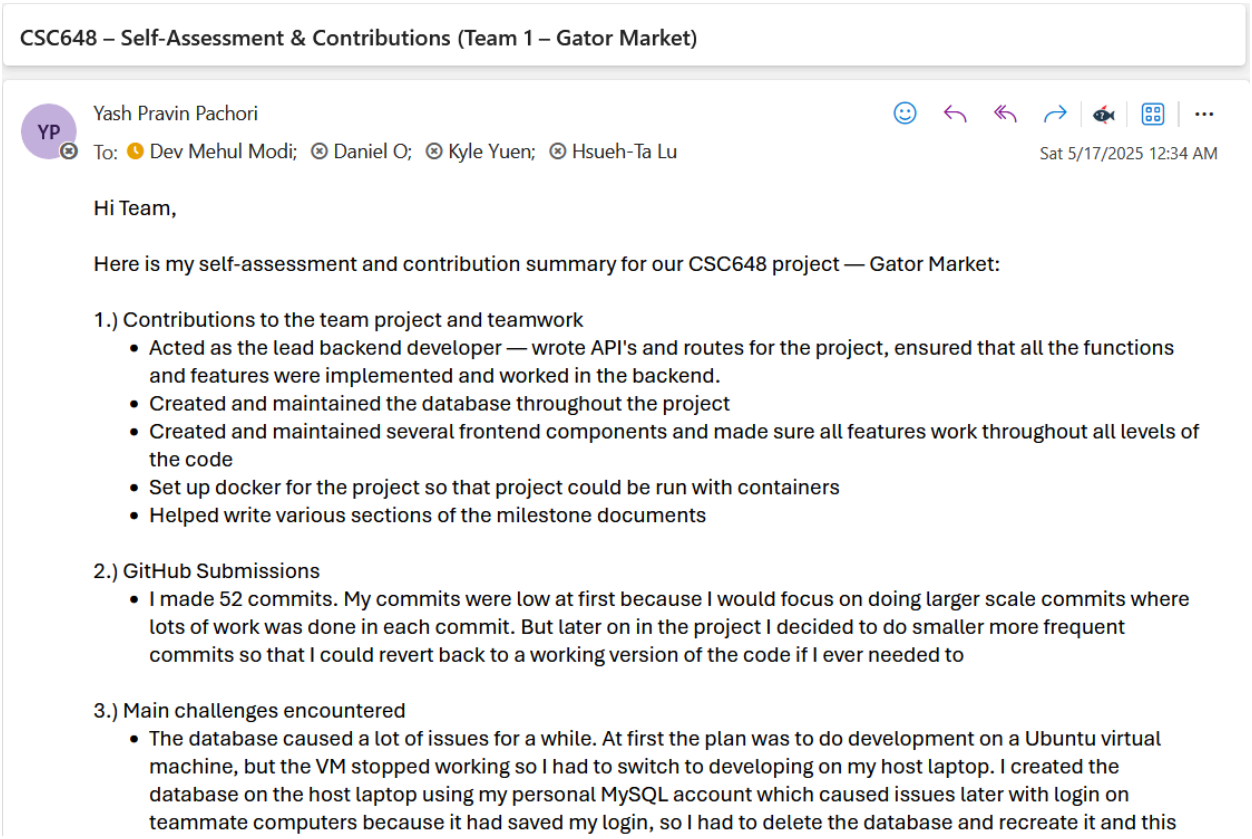
5.) What I would do better next time

- I think I would definitely have implemented the core features in the db, backend, and frontend a lot earlier and tested them more often. Features such as login, product upload, searching for products, etc. This is because these types of features ended up causing the most issues and bugs and a lot of time was spent in fixing them, and I think if I had started working on them a lot earlier then a lot of time and effort would be saved
- Another thing I would have done is ask for more help or at least split up tasks more. Dev and Daniel had some experience with Python Flask and were willing to help me, but I was thinking that I could do it all myself because I also have experience not only with Python Flask but with backend development, so I ended up putting too much on my plate. Later on, I worked collaboratively with Dev to implement features and coordinated with Daniel as well to troubleshoot backend issues
- I would communicate more clearly with frontend team about what specifically the backend components are doing and how they can use it to integrate the features into the frontend. I did explain what was going on but I feel like if I had gone into some more detail then it would have just made everything a little more easier

- I would have spent more time trying to figure out why environment variables weren't working because it would've saved Daniel a lot of trouble whenever he needed to update the server

Overall, this was a very educative and fun experience and I am glad I got to do it with this team!

Sincerely,
Yash Pachori



CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

YP

Yash Pravin Pachori

To: Dev Mehul Modi; Daniel O; Kyle Yuen; Hsueh-Ta Lu

Sat 5/17/2025 12:34 AM

Hi Team,

Here is my self-assessment and contribution summary for our CSC648 project — Gator Market:

1.) Contributions to the team project and teamwork

- Acted as the lead backend developer — wrote API's and routes for the project, ensured that all the functions and features were implemented and worked in the backend.
- Created and maintained the database throughout the project
- Created and maintained several frontend components and made sure all features work throughout all levels of the code
- Set up docker for the project so that project could be run with containers
- Helped write various sections of the milestone documents

2.) GitHub Submissions

- I made 52 commits. My commits were low at first because I would focus on doing larger scale commits where lots of work was done in each commit. But later on in the project I decided to do smaller more frequent commits so that I could revert back to a working version of the code if I ever needed to

3.) Main challenges encountered

- The database caused a lot of issues for a while. At first the plan was to do development on a Ubuntu virtual machine, but the VM stopped working so I had to switch to developing on my host laptop. I created the database on the host laptop using my personal MySQL account which caused issues later with login on teammate computers because it had saved my login, so I had to delete the database and recreate it and this

CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

time let docker handle it. I created a user for the database so that everyone will be able to access the database if needed
- There were challenges with the backend and with exposing the ports. Originally, we had the backend on port 8000 but we realized that whenever we needed to update the AWS server, we would have to change the port number. So later we decided to have the backend on port 8001 but we were pretty deep into development at that point and so we would face a lot of issues where the backend wouldn't connect successfully because we forgot to change the port number somewhere in the code. We also faced a lot of issues with axios and nginx, but Daniel was able to figure those out and handle them
- One issue I specifically faced was with product uploads. Whenever a user uploaded a product, it wouldn't show up in the products table of the database. I found out that it was because the backend route was never being hit which meant that the database would never be updated, so to fix this I had to do a lot of troubleshooting with the backend and rewrite many sections of the code. I also had to rewrite a few sections of the database but in the end I was able to get it working properly

4.) Use of GenAI
- I used GenAI extensively throughout this project. I mainly used ChatGPT but sometimes I used Claude because ChatGPT would make a lot of errors. GenAI was really helpful because I could ask it to help me with debugging code. Sometimes I wouldn't be able to figure out why something wasn't working and ChatGPT would tell me it's just because of a simple syntax error
- Another way I used it is to help me brainstorm how I can implement a certain thing or brainstorm different ways I can approach a problem and the pros and cons of each path
- I also asked it to help me with SQL queries because I wasn't able to remember them sometimes
- ChatGPT helped me setup Docker because this was my first time ever using it and it was really helpful. I would ask it for different commands I can use and how I would be able to make the database inside Docker

5.) What I would do better next time
- I think I would definitely have implemented the core features in the db, backend, and frontend a lot earlier and

CSC648 – Self-Assessment & Contributions (Team 1 – Gator Market)

5.) What I would do better next time
- I think I would definitely have implemented the core features in the db, backend, and frontend a lot earlier and tested them more often. Features such as login, product upload, searching for products, etc. This is because these types of features ended up causing the most issues and bugs and a lot of time was spent in fixing them, and I think if I had started working on them a lot earlier then a lot of time and effort would be saved
- Another thing I would have done is ask for more help or at least split up tasks more. Dev and Daniel had some experience with Python Flask and were willing to help me, but I was thinking that I could do it all myself because I also have experience not only with Python Flask but with backend development, so I ended up putting too much on my plate. Later on, I worked collaboratively with Dev to implement features and coordinated with Daniel as well to troubleshoot backend issues
- I would communicate more clearly with frontend team about what specifically the backend components are doing and how they can use it to integrate the features into the frontend. I did explain what was going on but I feel like if I had gone into some more detail then it would have just made everything a little more easier
- I would have spent more time trying to figure out why environment variables weren't working because it would've saved Daniel a lot of trouble whenever he needed to update the server

Overall, this was a very educative and fun experience and I am glad I got to do it with this team!

Sincerely,
Yash Pachori