

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

Team 5

Gator Savvy

Yu-Ming Chen (Team Lead) - ychen88@sfsu.edu

David Cabanela (GitHub Master) - dcabanel@mail.sfsu.edu

David (Jun) Chang (Backend Lead) - dchang6@mail.sfsu.edu

Alexis Perez (Frontend Lead) - aperez75@sfsu.edu

Athan Cheung (Frontend) - acheung4@mail.sfsu.edu

Demo URL: <http://35.94.172.233>

May 19, 2025

Table of Contents

1) Product summary	3
2) Milestone documents – M1-M4	4
Milestone 1	4
Milestone 2	19
Milestone 3	53
Milestone 4	54
3) Product Screenshots	71
4) Database Organization	77
5) GitHub organization	81
6) Google Analytics stats plot for our WWW site	82
7) Project management	83
8) Team member self-assessment and contributions	85
Alexis -> Frontend Lead	85
Athan -> Frontend Tech	85
David -> Github Master	86
Jun -> Backend Lead	87
Yu-Ming Chen (Jim) -> Team Lead	88

1) Product summary

Name of our product: Gator Savvy

Product Description: Gator Savvy is a marketplace web application for SFSU students and faculty to buy, sell, rent items, and get exclusive student deals. Whether it's textbooks, clothes, tech, or everyday essentials, students can find what they need at affordable prices from other students or SFSU-affiliated groups/services.

Itemized List of All Major Committed Functions

1. Guests can browse and search for items.
2. Guests can filter item listing search results by category, price, and more.
3. Guests can sign up for a Gator Savvy account (SFSU email required).
4. Registered users can post item listings for sale or rent.
5. Registered users can filter item listing search results by category, price, and more.
6. Registered users can build credibility by requesting a verification badge.
7. Registered users can set up a profile with a username and picture.
8. Registered users can message each other inside the app.
9. Admins review and approve item listings before they show up.

What's Unique/SFSU specific: Gator Savvy is specifically designed for the SFSU community. It requires users to register with an SFSU email to ensure that all transactions are within a trusted network of students and faculty. This secure environment allows members to buy, sell, and rent items tailored to the needs of university life, from textbooks to household items, while fostering a sense of community and supporting both the academic and social experience unique to SFSU.

URL to Gator Savvy: <http://35.94.172.233>

2) Milestone documents – M1-M4

“SW Engineering CSC648-848 Spring 2025”

Gator Savvy

Team 5

Yu-Ming Chen (Team Lead) - yuchen88@sfsu.edu

David Cabanel (GitHub Master) - dcabanel@mail.sfsu.edu

David (Jun) Chang (Backend Lead) - dchang6@mail.sfsu.edu

Alexis Perez (Frontend Lead) - aperez75@sfsu.edu

Athan Cheung (Frontend) - acheung4@mail.sfsu.edu

Milestone 1

History

Updates	Date Submitted	Date Revised
Milestone 1 first meetup		2/26/2025
Milestone 1 meetup		3/1/2025
Milestone 1 meetup		3/8/2025
Milestone 1 1st submission	3/12/2025	3/11/2025
Milestone 1 meetup		3/12/2025
Milestone 1 2nd submission	3/12/2025	3/12/2025
Milestone 1 meetup		3/16/2025
Milestone 1 Final Revision	3/19/2025	3/19/2025

1. Executive Summary

Project Name: Gator Savvy

Students often struggle to make ends meet in today's fast-paced world while balancing their academic responsibilities. The rising cost of living, textbooks, and everyday essentials can leave students searching for affordable options. Gator Savvy aims to solve this problem by creating a dedicated online marketplace that connects SFSU students with exclusive discounts and deals from campus services, student organizations, and university-affiliated vendors. By centralizing these offers, Gator Savvy ensures that students can access the discounts they need, all in one place.

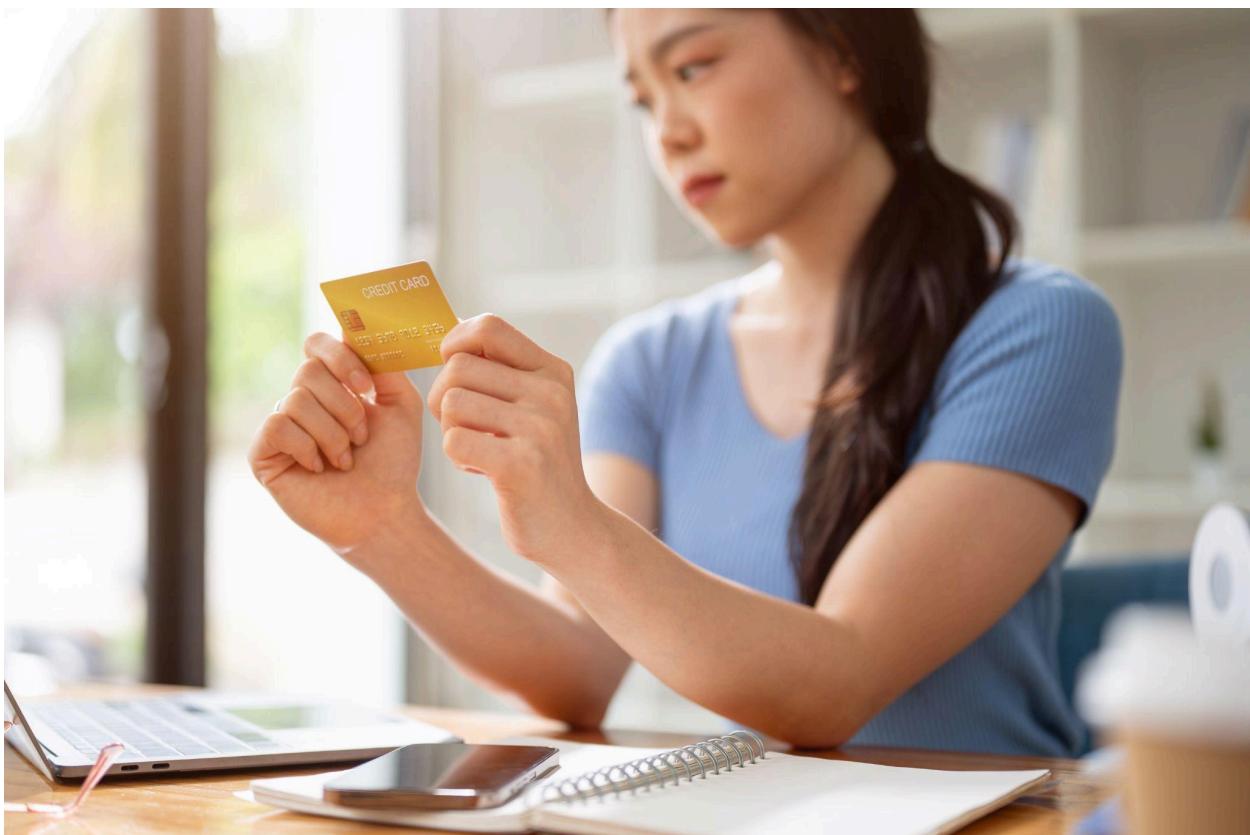
Gator Savvy will function as a one-stop shop for students looking for deals on everything from food, clothing, and tech gadgets to textbooks, transportation, and entertainment. Students can access a wide array of verified, time-sensitive discounts through an intuitive website. The platform will also include features like personalized recommendations based on location and student preferences, and a user-friendly interface that allows businesses to update and offer new promotions easily. Using Gator Savvy, students can save money on the products and services they need most, while businesses gain access to a loyal and targeted customer base.

What makes Gator Savvy unique is its specific focus on the student experience. Unlike other general coupon sites, Gator Savvy is designed exclusively for the student community, ensuring that all deals are relevant and beneficial to students' everyday needs. Additionally, the platform will partner with local San Francisco businesses and offer exclusive, campus-specific discounts that students can't find elsewhere. This personalized approach to discounts and its seamless integration with campus events and student life ensures that Gator Savvy stands out as a vital tool for SFSU students.

Our startup team comprises driven SFSU students who understand the challenges of managing finances during school. We are a group of passionate individuals with tech, marketing, and business development backgrounds. We are committed to creating a marketplace that saves students money and strengthens the relationship between SFSU and the local community. Funding this project will allow us to refine the platform, expand our reach, and ensure that Gator Savvy becomes a must-have resource for students looking to make the most of their time at SFSU.

2. Personae

Persona 1: The Budget-Conscious Student (Sophia)



About: Business Administration Major, 21 years old, Junior

- Very Busy
- Balancing her budget with school and social life.
- Familiar with shopping apps and comfortable with many technologies.
- Difficulty finding, trusting(third-party sellers), and keeping track of limited deals.
- Lack of time to search

Goals:

- Looking for deals and savings promptly.
- They want quick and easy access to deals on food and school-related items.

How Gator Savvy Helps:

Gator Savvy simplifies Sophia's shopping experience by offering a curated marketplace of student discounts in one place. It saves her time by providing tailored offers based on her preferences and location. The platform's student verification ensures all deals are trustworthy, and personalized notifications alert her to relevant deals she might have missed otherwise.

Persona 2: Faculty Member who supports Gator Savvy (Professor Jen)



About: Busy full-time Business Professor at SFSU, 38 years old

- Wants to help students save money
- Doesn't have time to search for discounts or suggest them to students
- Finds it hard to connect students with helpful campus resources outside of class

Goals:

- Needs an easy way to direct students to helpful and affordable deals
- Needs to get rid of old textbooks and some old furniture
- Needs to find a way to share cheaper lecture materials or second-hand textbooks for students

How Gator Savvy Helps:

Gator Savvy makes it easy for Professor Jen to share discounts with her students by offering all the best SFSU deals in one place. The platform saves her time by showing verified offers on things like textbooks, food, and transportation. She can quickly share these deals with students, making sure they get the savings they need. She also wants to get rid of old textbooks and some old furniture; Gator Savvy helps make it simple to list and sell these items for people who need them on the platform.

Persona 3: The Tech-Savvy Freshman (James)



About: Computer Science Major, 18 years old, freshman

- Very comfortable with technology.
- Frequently uses technologies to simplify tasks.
- Can be overwhelmed with too much information.
- Can get frustrated if a site requires too much effort to navigate.
- Wants to save money

Goals:

- Looking to explore any available resources as a new student.
- Prioritize convenience and quick access.

How Gator Savvy Helps:

Gator Savvy's user-friendly interface makes it easy for James to find discounts that match his needs quickly. The platform can show him the best options to buy, exchange, or rent that item quickly and simply. With Gator Savvy, he can access discounts with just a few taps, simplifying his search for savings.

3. High-level Use cases

1. Search for Affordable School Supplies

Sophia (Persona 1) is looking for cheaper ways to get some tools for class. She logs into Gator Savvy using her school email and searches for exclusive deals on school supplies and groceries.

2. Explore Alternative Options for School Supplies

Sophia clicks on SFSU Exclusive deals, but she doesn't need that item for too long. From the item page, she can look for other options, including renting, borrowing, exchanging, or purchasing second-hand items.

3. Shop for Textbooks and Supplies

Jen (Persona 2) is looking for some textbooks and school supplies for her students in her finance class. She's able to register her account via her SFSU email to shop for supplies and extra lecture materials with ease and reasonable values for her students.

4. List Items for Sale

Jen is not that good with technology. With the simple design of our website, she can put her items, for example, old textbooks and furniture that she wants to sell, with ease.

5. Search for Related Items

James (Persona 3) is trying to look for some related items to purchase for his robot project. He can look up related items or textbooks using our simple, categorized search, which refreshes in real time, along with the search conditions.

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

Users & Roles:

Registered User

Verified user who can interact with the entirety of the website and has access to student deals.

For a user to be verified, they must have an SFSU email. Items can also be traded or even donated between Registered Users.

- View, search, and purchase items.
- Like and review items.
- Update their profiles.
- Make posts and sell items.
- Give a listed item the option to rent.
- Can access special offers
- Message sellers.

Guest User

Common user with limited access. This provides a preview for our app.

- View and search events and items.
- Read reviews.
- Can navigate to any section that does not require signing up.

Key entities:

Student Profile

Stores the registered user's information and preferences.

- Username
- Full name
- Student Email
- Student id
- Major
- Preferences(categories)
- Liked/Wishlisted items
- Verification Badge (if any)
- Seller's Rating & Reviews/Comments
- Items list (if any)
 - For sale
 - Sold

Item Listing

Provides a full description of everything you need to know about an item. This listing is where registered users can also message the sellers.

- Item Name
- Description
- Pictures
- Videos(if any)
- Price
- Category(can be multiple)
- Seller Reviews
- Quality description(new, like new, etc.)
- Chat with seller
- Location
- Rental option(if provided by Seller)

List of Items

This would be a general view after you search for something. You get a list of items for quickly viewing the picture name and price of each item.

- Item name
- Picture
- Price
- Location/Distance

Verification Badge

A “verified” badge icon, visible on a Registered User's profile, indicating that they are a more reliable/trusted seller. Users with little to no sales will not have a badge.

Categories

Used to categorize items for our website. Example categories: textbooks, furniture, electronics, clothing, etc.

Search

Simple search functionality that allows users to specify categories or plain text to find items they are looking for.

5. List high-level functional requirements

1. **Student Verification:** To ensure the platform is restricted to students and faculty, users shall only be able to register and post items with an SFSU email address.
2. **Classified by Category:** Sales shall be organized into categories such as textbooks, furniture, electronics, clothing, and event tickets to help students quickly find what they are looking for.
3. **Price Comparison Tool:** Students shall be able to compare prices for similar items posted by different sellers (or even compare with online stores).
4. **Customizable Alerts:** Students shall be able to set up price drop or availability alerts for specific items they are looking for.
5. **Student-to-Student Reviews:** Buyers and sellers shall be able to leave reviews specifically about the transaction experience, product quality, and communication.
6. **Wishlist & Save for Later:** Users shall be able to create a wishlist and "save for later" items they're interested in but not ready to buy yet.
7. **Instant Messaging for Buyers & Sellers:** In-app messaging shall be available for direct communication between students, with the option to send quick replies or images.
8. **Event-based Deals:** The platform shall offer a space to share event-based promotions, such as discounts on textbooks during registration periods or specials during finals week.
9. **Item Rental Option:** Users shall be able to list items for rent (e.g., textbooks, cameras, or even bicycles) for a fixed period.
10. **Geo-location-based Search:** Deals or sellers who are geographically close to the user's campus or dorm shall be displayed to help with local transactions.
11. **Barter System:** Students shall be able to exchange items (e.g., "I'll trade my old bike for your old laptop") instead of just buying and selling.
12. **Limited-time Flash Sales:** Timed sales events shall be available, where certain items are discounted for a limited period or have a specific number of products available.
13. **Verification Badges:** Badges shall be introduced for top-rated sellers, verified sellers, or trusted students to build community trust.
14. **Student ID Integration:** The platform shall integrate with the SFSU student ID to get special offers or student-specific deals.
15. **Donation Feature:** Students shall be able to donate items directly to other students in need (for example, textbooks for students who can't afford them).
16. **Video Listings:** Sellers shall be able to upload short videos showcasing their items to give buyers a better idea of the product's condition.
17. **Transportation Deals:** The platform shall offer a space for students to sell or share transportation options (e.g., carpooling or bike-sharing).

18. **Cultural Event Tickets:** Access shall be provided to discounted tickets or merchandise for SFSU-related cultural events, sports, or student gatherings.
19. **Community Polls for Popular Items:** Users shall be able to vote on which items or types of deals they'd like to see more of (e.g., "Would you prefer discounted winter jackets or tech accessories this semester?").
20. **"Take My Stuff" Section:** Students shall be able to post about items they want to give away for free (e.g., old furniture, clothes, or kitchen appliances), contributing to sustainability efforts.

6. List of non-functional requirements

1. The application shall be developed, tested, and deployed using tools and cloud servers approved by Class CTO and as agreed in M0.
2. The application shall be optimized for standard desktop/laptop browsers, e.g., it must render correctly on the two latest versions of the 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.s
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.e
7. The privacy of users shall be protected
8. The language used shall be English (no localization needed)
9. The application shall be straightforward to use and intuitive
10. The application shall follow established architecture patterns
11. The application code and its repository shall be easy to inspect and maintain
12. Google Analytics shall be used
13. No email clients or chat services shall be allowed. Interested users can only message the 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 or simulated in UI.
15. Site security: basic best practices shall be applied (as covered in the class) for the main data items
16. Media formats shall be standard as used in the market today
17. Modern SE processes and tools shall be used as specified in the class, including collaborative and continuous SW development and GenAI tools
18. The application UI (WWW and mobile) shall prominently display the following exact text on all pages: "SFSU Software Engineering Project CSC 648-848, Spring 2025. For Demonstration Only" at the top of the WWW page Nav bar. (It is important not to confuse this with a real application.) You have to use this exact text without any editing.

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

Feature	Gator Savvy	Student Beans	Craigslist	UNiDAYS	Amazon
Text Search	+	+	+	+	++
SFSU Student Verification Login	+	-	-	-	-
Shopping Cart	+	+	+	+	+
Browse	++	+	+	+	++
Giveaway free items	+	-	-	-	-
Rental System	+	-	-	-	-

** + feature exists; ++ superior; - does not exist

Gator Savvy's focus on local, campus-specific, and personalized student deals in San Francisco sets it apart from the competition. While platforms like Student Beans, Craigslist, UNiDays, and Amazon offer student discounts, Gator Savvy differentiates itself by integrating deeply into the student experience at SFSU. Gator Savvy provides exclusive SFSU student verification, ensuring that all discounts are legitimate and tailored to the campus community. Additionally, the platform includes a Giveaway Free Items feature and a Rental System, which none of the competitors offer, allowing students to exchange or borrow items sustainably. The Text Search, Shopping Cart, and Browse features are common across platforms, but Gator Savvy's unique offering of community-based, verified discounts and the ability to rent and give away items are crucial advantages that directly address the needs of SFSU students.

8. High-level system architecture and technologies used

Server Host: Amazon AWS EC2

Operating System: Ubuntu Server 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

Database: MySQL Ver 8.0.41-Ubuntu0.24.04.1 for Linux on x86_64 ((Ubuntu))

Web Server: NGINX 1.26.3

Server-Side Language: Python 3.12.3

Web-Side Language: JavaScript

Backend Framework: Flask 3.0.2

Frontend Framework: Vue.js v3.5.13

IDE: PyCharm, vs code

Web Analytics: Google Analytics

SSL Cert: Let's Encrypt (Cert Bot)

Documentation: Doxygen (1.13.2)

Supported Browsers: Safari 18.3 & Chrome LTS-126

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

AI models used:

ChatGPT (GPT-4o) - HIGH

ChatGPT:

We used ChatGPT to give us ideas for our marketplace application, and Gator Savvy is one of them.

Task 1: Find the port and IP for the MySQL server and allow access from everywhere.

(<https://chatgpt.com/share/67bab863-1cc8-800d-abe7-d26b5873914d>)

Helpfulness: Very helpful

This is very clear and straight to the point. I got it going within 15 minutes.

Task 2: Setting up NGINX

(<https://chatgpt.com/share/67bab8b9-d25c-800d-964f-14100aff63e7>)

Helpfulness: Not so helpful

It was going well at first, but it provided too much extra/unnecessary information and kept looping answers that weren't working before.

Task 3: Suggesting front-end languages

Helpfulness: Very helpful

We asked ChatGPT for suggestions on an easy-to-learn and useful front-end language for our website, and we chose Vue.js.

Comment on anything else you found useful:

- Brainstorming ideas
- Resolving merge errors

10. Team and roles

Name	School Email	Role
Yu-Ming Chen	ychen88@sfsu.edu	Team Lead
David (Jun) Chang	dchang6@mail.sfsu.edu	Back-End Lead
Alexis Perez	aperez75@sfsu.edu	Front-End Lead
Athan Cheung	acheung4@mail.sfsu.edu	Front-End
David Cabanel	dcabanel@mail.sfsu.edu	Github Master

11. Team Lead Checklist to be completed by the team lead

For each item below, the team lead must answer with only one of the following: DONE/OK, or ON TRACK (meaning it will be done on time and no issues perceived), or ISSUE (you have some problems, and then define the problem with 1-3 lines)

- So far, all team members are fully engaged and attending team sessions when required. **DONE**
- The team found a time slot to meet outside of the class. **DONE**
- The team is ready and able to use the chosen back and front-end frameworks, and those who need to learn are working on learning and practicing **ON TRACK**
- The team reviewed class slides on requirements and use cases before drafting Milestone 1 **DONE**
- The team reviewed non-functional requirements from the “How to start...” document and developed Milestone 1. **DONE**
- The team lead checked the Milestone 1 document for quality, completeness, formatting, and compliance with instructions before the submission. **n DONE**
- The team lead ensured that all team members read the final M1 and agreed/understood it before submission. **DONE**
- The team shared and discussed the experience with GenAI tools among themselves **DONE**
- GitHub is organized as discussed in class (e.g., master branch, development branch, folder for milestone documents, etc). **DONE**

“SW Engineering CSC648-848 Spring 2025”

Gator Savvy

Team 5

Yu-Ming Chen (Team Lead) - ychen88@sfsu.edu

David Cabanela (GitHub Master) - dcabanel@mail.sfsu.edu

David (Jun) Chang (Backend Lead) - dchang6@mail.sfsu.edu

Alexis Perez (Frontend Lead) - aperez75@sfsu.edu

Athan Cheung (Frontend) - acheung4@mail.sfsu.edu

Milestone 2

History

	Date Submitted	Date Revised
Milestone 2 class time		3/17/2025
Milestone 2 meetup		3/19/2025
Milestone 2 meetup		3/22/2025
Milestone 2 part 1 meetup		3/24/2025
Milestone 2 part 1 meetup	3/25/2025	3/25/2025
Milestone 2 part 2 meetup		3/31/2025
Milestone 2 Final Submission	4/2/2025	4/2/2025

1. Executive summary

Project Name: Gator Savvy

Students often struggle to make ends meet in today's fast-paced world while balancing their academic responsibilities. The rising cost of living, textbooks, and everyday essentials can leave students searching for affordable options. Gator Savvy aims to solve this problem by creating a dedicated online marketplace that connects SFSU students with exclusive discounts and deals from campus services, student organizations, and university-affiliated vendors. By centralizing these offers, Gator Savvy ensures that students can access the discounts they need, all in one place.

Gator Savvy will function as a one-stop shop for students looking for deals on everything from food, clothing, and tech gadgets to textbooks, transportation, and entertainment. Students can access various verified, time-sensitive discounts through an intuitive website. The platform will also include features like personalized recommendations based on location and student preferences, and a user-friendly interface that allows businesses to update and offer new promotions easily. Using Gator Savvy, students can save money on the products and services they need most, while businesses gain access to a loyal and targeted customer base.

What makes Gator Savvy unique is its specific focus on the student experience. Unlike other general coupon sites, Gator Savvy is designed exclusively for the student community, ensuring that all deals are relevant and beneficial to students' everyday needs. Additionally, the platform will partner with local San Francisco businesses and offer exclusive, campus-specific discounts that students can't find elsewhere. This personalized approach to discounts and its seamless integration with campus events and student life ensures that Gator Savvy stands out as a vital tool for SFSU students.

Our startup team comprises driven SFSU students who understand the challenges of managing finances during school. We are passionate individuals with tech, marketing, and business development backgrounds. We are committed to creating a marketplace that saves students money and strengthens the relationship between SFSU and the local community. Funding this project will allow us to refine the platform, expand our reach, and ensure that Gator Savvy becomes a must-have resource for students looking to make the most of their time at SFSU.

2. List of main data items and entities

Users & Roles:

Registered User

The verified user shall interact with the entirety of the website and have access to student deals.

For a user to be verified, they must have an SFSU email. Registered users shall also trade or donate items.

- View, search, and purchase items.
- Like and review items.
- Update their profiles.
- Make posts and sell items.
- Give a listed item the option to rent.
- shall access special offers
- Message sellers.

Guest User

Common user with limited access. This provides a preview for our app.

- View and search events and items.
- Read reviews.
- Shall navigate to any section that does not require signing up.

Key entities:

Profile

Stores the registered user's information and preferences.

- Username
- Full name
- SFSU Email
- SFSU ID
- Preferences(categories)
- Liked/Wishlisted items
- Verification Badge (if any)
- Seller's Rating & Reviews/Comments
- Items list (if any)
 - For sale
 - Sold

Item Listing

This listing provides a full description of everything you need to know about an item. Registered users can also message the sellers from this listing.

- Item Name
- Description
- Pictures
- Videos (if any)
- Price
- Category (can be multiple)
- Reviews
- Quality description (new, near mint, etc.)
- Location
- Rental (if provided by Seller)

Verification Badge

A “verified” badge icon, visible on a Registered User's profile, indicates that they are a more reliable/trusted seller. Users with little to no sales will not have a badge.

- Badge Type

Category

Used to categorize items for our website. Example categories: textbooks, furniture, electronics, clothing, etc.

- Category Name

Shopping Cart

Used to store information about the items a user intends to purchase. It holds the user's selected items and their details until they are ready to proceed with the checkout process.

- Item Name
- Price
- Quantity

3. Functional Requirements - prioritized

- **Priority 1 (Must-have)**

- Guest user
 - Shall search and view items
 - Shall search and filter listings by category, price, etc.
- Registered user
 - Shall register with an SFSU email to be registered
 - Shall post items for sale or rent
 - Shall search and filter listings by category, price, etc.
 - Shall opt for a verification badge to build credibility
 - Shall create a profile with a username and profile picture
 - In-app messaging
- Admin
 - Shall approve item listings

- **Priority 2 (Desired)**

- Guest user
 - None
- Registered user
 - Shall search and filter items by location
 - Shall save items and save history
 - Shall remove item listings
- Admin
 - Secure access for deal management and analytics
 - Shall monitor reports on flagged content
 - Shall remove item listings
 - Shall manage and verify users

- **Priority 3 (Opportunistic)**

- Guest user
 - Shall read reviews of items and sellers
 - Receive promotional emails if subscribed
- Registered user
 - Implement Google Maps for geo-location-based searches
 - Profiles shall have preferences and reviews
- Admin
 - Advanced analytics on user engagement or items

4. UI Storyboards for each main use case (low-fidelity B&W wire diagrams only).

Storyboard - Use case 1

Sophia (Persona 1) is looking for cheaper ways to get some tools for class. She logs into Gator Savvy using her school email and searches for exclusive deals on school supplies and groceries.

The wireframe shows the Gator Savvy homepage. At the top, there is a header with a logo, the site name "Gator Savvy", a "All Categories" dropdown, a search bar with placeholder text "Search for items...", a "Sell" button, and a "Login/Sign Up" button. A red arrow points from the text below to the "Login/Sign Up" button. Below the header is a large rectangular area containing placeholder text: "[Big image] Left scroll image maybe Promotions? Special deals? Specific to SFSU affiliated stores". Underneath this is a section titled "Big Title! Foods or whatever" followed by five placeholder cards labeled "Item 1 - Price". Below these cards is a "(See more)" link. Further down is another section titled "Second hand stuff etc. or something else" with five placeholder cards labeled "Item 1". Below these is another "(See more)" link. At the very bottom is a footer bar with the text "FOOTER, contact us, FAQ, and more...".

[Big image] Left scroll image maybe
Promotions? Special deals?
Specific to SFSU affiliated stores

Big Title! Foods or whatever

Item 1 - Price Item 1 - Price Item 1 - Price Item 1 Item 1

(See more)

Second hand stuff etc. or something else

Item 1 Item 1 Item 1 Item 1 Item 1

(See more)

FOOTER, contact us, FAQ, and more...

Sophia is on the homepage and clicks the login/signup button, and is redirected to the login.

Sign in

SFSU email

Password

Login

Forgot password

Don't have an account



FOOTER, contact us, FAQ, and more...

Sophia fills in her account information and hits login, which then redirects her back home.

Logo Gator Savvy

All Categories ▾ | Search for items...



Sell

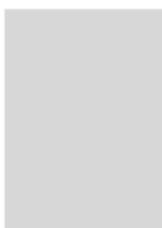
Username

[Big image] Left scroll image maybe
Promotions? Special deals?
Specific to SFSU affiliated stores

Big Title! Foods or whatever



Item 1 - Price



Item 1 - Price



Item 1 - Price



Item 1



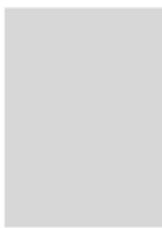
Item 1

(See more)

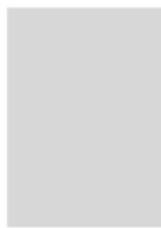
Second hand stuff etc. or something else



Item 1



Item 1



Item 1



Item 1

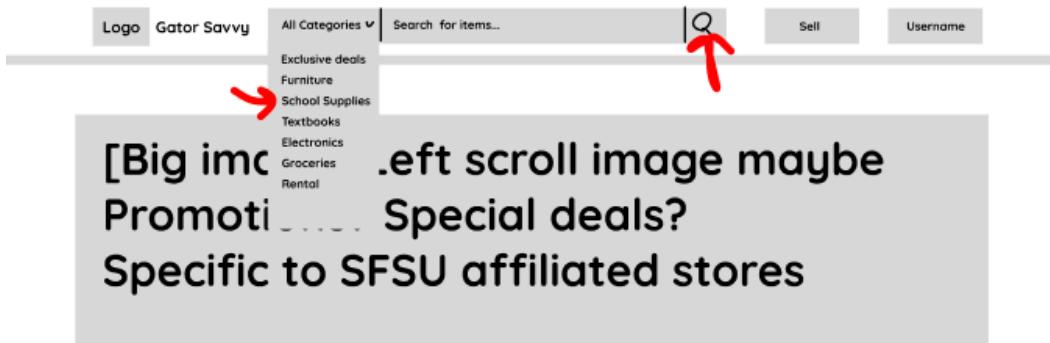


Item 1

(See more)

FOOTER, contact us, FAQ, and more...

*Sophia hits the drop-down menu where the search bar is to access any relevant filters.
Specifically, the exclusive deals (made for students)*



Big Title! Foods or whatever

Item 1 - Price Item 1 - Price Item 1 - Price Item 1 Item 1

(See more)

Second hand stuff etc. or something else

Item 1 Item 1 Item 1 Item 1 Item 1

(See more)



Once ready, she clicks the search button to bring up a list of items.

Logo Gator Savvy

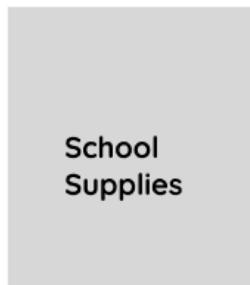
School Supplies

Search for items...



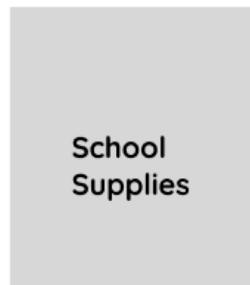
Sell

Username



School
Supplies

Item 1
Price



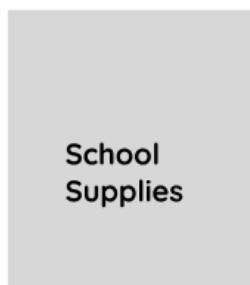
School
Supplies

Item 1
Price



School
Supplies

Item 1
Price



School
Supplies

Item 1
Price



School
Supplies

Item 1
Price



School
Supplies

Item 1
Price



School
Supplies

Item 1
Price



School
Supplies

Item 1
Price



School
Supplies

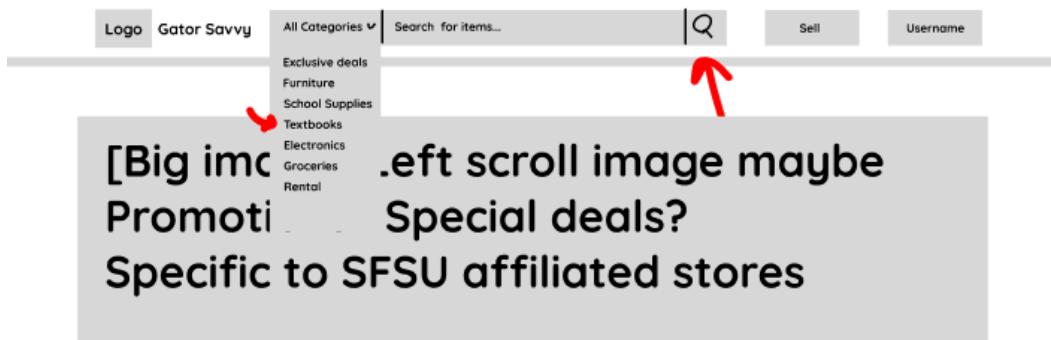
Item 1
Price

(See More)

FOOTER, contact us, FAQ, and more...

Storyboard 2 - Use case 2

Sophia clicks on SFSU Exclusive deals, but she doesn't need that item for too long. From the item page, she shall be able to look for other options, including renting, borrowing, exchanging, or purchasing second-hand items.



[Big image left scroll image maybe]

Promotional image

Special deals?

Specific to SFSU affiliated stores

Big Title! Foods or whatever

Item 1 - Price

Item 1 - Price

Item 1 - Price

Item 1

Item 1

(See more)

This section shows a grid of five placeholder cards for food items. Each card has a small grey square at the top left and the text "Item 1 - Price" below it. Below the grid is a link "(See more)".

Second hand stuff etc. or something else

Item 1

Item 1

Item 1

Item 1

Item 1

(See more)

This section shows a grid of five placeholder cards for second-hand items. Each card has a small grey square at the top left and the text "Item 1" below it. Below the grid is a link "(See more)".



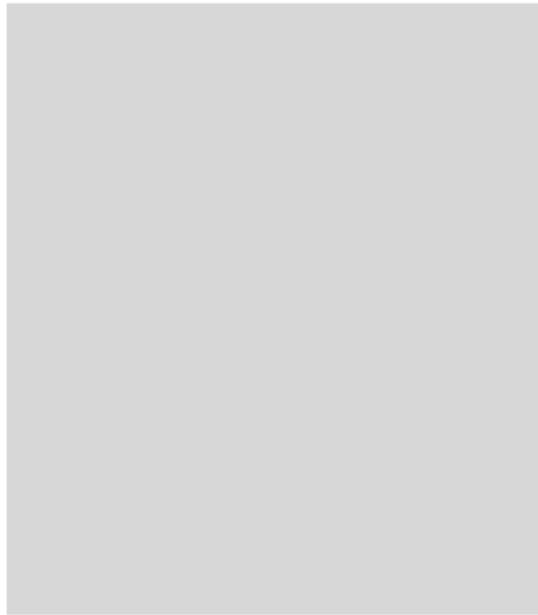
Sophia now uses the dropdown menu to specify textbooks, or exclusive deals(These are the deals available due to having the SFSU email), and items available to rent.

The screenshot shows a search results page for 'Textbooks' on the Gator Savvy website. At the top, there is a navigation bar with a logo, the site name 'Gator Savvy', a red circle highlighting the 'Textbooks' link, a search bar with placeholder text 'Search for items...', a magnifying glass icon, a 'Sell' button, and a 'Username' input field.

The main content area displays a grid of nine search results, arranged in three rows of three. Each result card has a gray background, the word 'Textbook' centered at the top, and a red arrow pointing to the bottom-left corner with the text 'Click!!'. Below each card, the text 'Item 1 Price' is visible. In the bottom right corner of the third row, there is a link '(See more)'.

At the very bottom of the page is a dark gray footer bar containing the text 'FOOTER, contact us, FAQ, and more...'.

This is just one of the lists (textbooks) that she gets back when completing a search. You will know what filter is applied in the search bar.



Short Description



Seller Badge Ratings/Reviews

^Can Click to go seller page^

Item Name

Price

Location

Condition

Categories



Rent

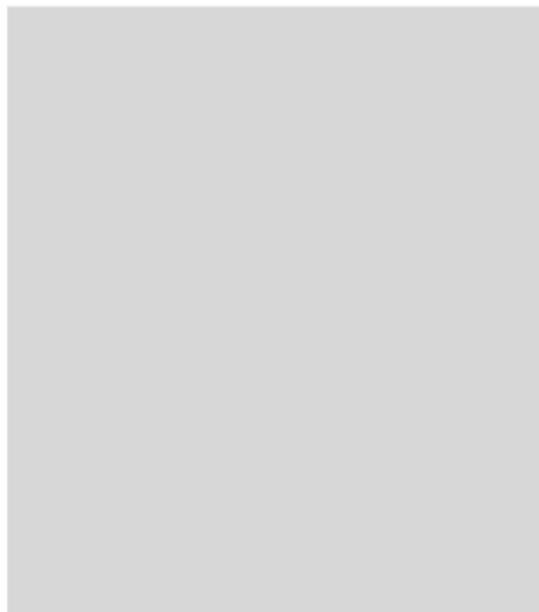
Buy



If looking for
exchanges then you
can chat with
seller

FOOTER, contact us, FAQ, and more...

Here she shall see all relevant details of said item, and is also provided with some seller information. If she wishes to make possible exchanges/trades, then she can contact the seller directly.

**Item Name****Price****Location****Condition****Categories****Rent****Buy****Seller****Short Description****Seller**  Ratings/Reviews

^Can Click to go seller page^

FOOTER, contact us, FAQ, and more...

Storyboard 3 - Use case 3

Jen (Persona 2) is looking for some textbooks and school supplies for her students in her finance class. She's able to register her account via her SFSU email to shop for supplies and extra lecture materials with ease and reasonable value for her students.

The storyboard illustrates a user interface for a website called "Gator Savvy". At the top, there is a navigation bar with a "Logo" icon, the site name "Gator Savvy", a "All Categories" dropdown, a search bar containing "Search for items...", a "Sell" button, and a "Login/Sign Up" button. A red arrow points from the text below to the "Login/Sign Up" button. Below the navigation bar is a large, semi-transparent callout box containing the text: "[Big image] Left scroll image maybe Promotions? Special deals? Specific to SFSU affiliated stores".

Big Title! Foods or whatever

Item 1 - Price Item 1 - Price Item 1 - Price Item 1 Item 1

(See more)

Second hand stuff etc. or something else

Item 1 Item 1 Item 1 Item 1 Item 1

(See more)

FOOTER, contact us, FAQ, and more...

Logo

Gator Savvy

Sign in

SFSU email

Password

Login

[Forgot password](#)

[Don't have an account?](#)



FOOTER, contact us, FAQ, and more...

Because Jen does not yet have an account, she clicks the don't have an account link.

Sign Up

First name

Last name

Username

SFSU email

Password

Repeat password

Create Account



FOOTER, contact us, FAQ, and more...

She fills in her appropriate information (SFSU EMAIL IS NEEDED) and creates an account. After clicking Create Account, she is redirected to the login.

Sign in

SFSU email



Password



 **Login**

[Forgot password](#)

[Don't have an account](#)

FOOTER, contact us, FAQ, and more...

Logo Gator Savvy

All Categories ▾

Search for items...



Sell

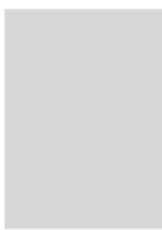
Username

[Big image] Left scroll image maybe
Promotions? Special deals?
Specific to SFSU affiliated stores

Big Title! Foods or whatever



Item 1 - Price



Item 1 - Price



Item 1 - Price



Item 1



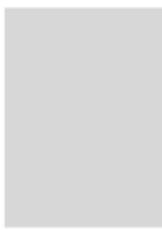
Item 1

(See more)

Second hand stuff etc. or something else



Item 1



Item 1



Item 1



Item 1

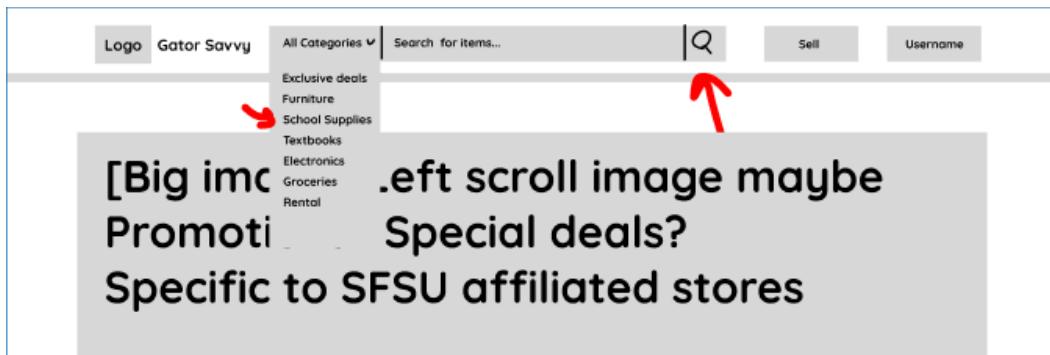


Item 1

(See more)

FOOTER, contact us, FAQ, and more...

After successfully logging in, she now searches using the drop-down with predefined filters to look for what she needs.



Big Title! Foods or whatever



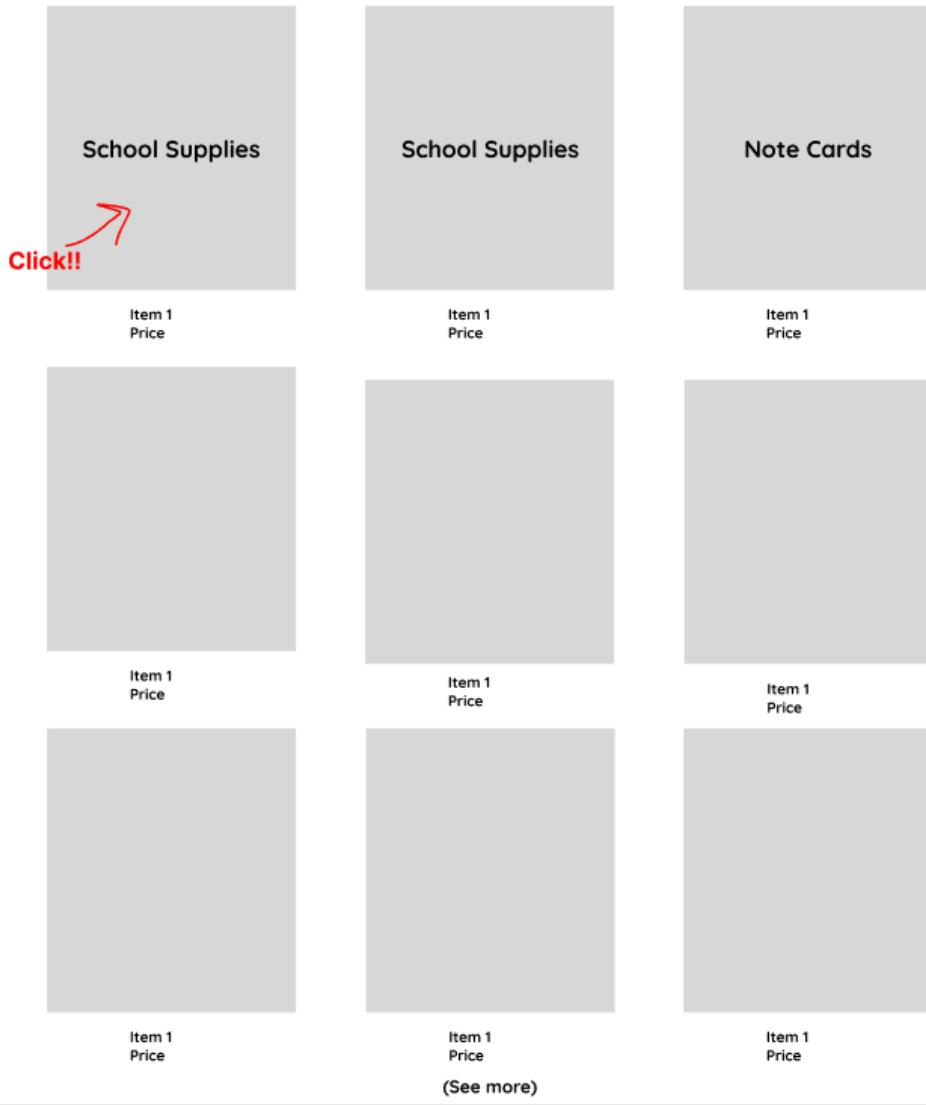
(See more)

Second hand stuff etc. or something else



(See more)

FOOTER, contact us, FAQ, and more...



FOOTER, contact us, FAQ, and more...

This is the list of items Jen was looking for.

Storyboard 4 - Use case 4

Jen is not that good with technology. With the simple design of our website, she can put her items, for example, old textbooks and furniture that she wants to sell, with ease.



[Big image] Left scroll image maybe
Promotions? Special deals?
Specific to SFSU affiliated stores

Big Title! Foods or whatever



(See more)

Second hand stuff etc. or something else



(See more)

FOOTER, contact us, FAQ, and more...

Jen clicks the Sell button at the top right of our homepage. (This button is only available if logged in.)

Sell

Item



Category



Price

\$ 

Available to rent?

Image

Condition

Upload 

Location



Description



Post



FOOTER, contact us, FAQ, and more...

She fills in all relevant information for said item she wants to sell, along with images, and if she wants to allow the item to be rentable.

After posting, she is redirected to the homepage.

Storyboard 5 - Use case 5

James (Persona 3) is trying to look for some related items to purchase for his robot project. He can look up related items or textbooks using our simple, categorized search, which refreshes in real-time, along with the search conditions.

The storyboard shows a user interface for a search engine. At the top, there is a navigation bar with a 'Logo' icon, the text 'Gator Savvy', a 'All Categories' dropdown menu (which has a red arrow pointing to it), a search input field with placeholder text 'Search for items...', a magnifying glass icon, a 'Sell' button, and a 'Username' button.

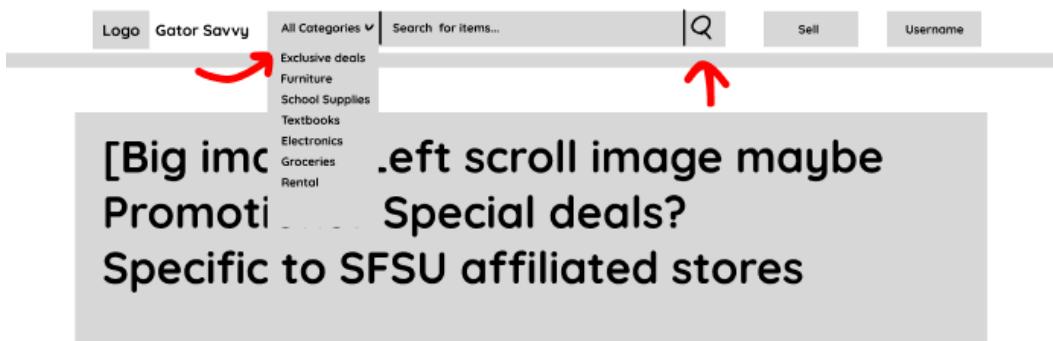
Below the navigation bar is a large gray box containing the text: '[Big image] Left scroll image maybe Promotions? Special deals? Specific to SFSU affiliated stores'.

Underneath this box, there is a section titled 'Big Title! Foods or whatever' followed by five placeholder image boxes labeled 'Item 1'. Below these boxes is a '(See more)' button.

Further down, there is another section titled 'Second hand stuff etc. or something else' followed by five more placeholder image boxes labeled 'Item 1'. Below these boxes is another '(See more)' button.

At the bottom of the page is a footer bar with the text 'FOOTER, contact us, FAQ, and more...'.

Assume James is logged in. He now uses the predefined filters to find what he needs.



Big Title! Foods or whatever

Item 1 - Price Item 1 - Price Item 1 - Price Item 1 Item 1

(See more)

Second hand stuff etc. or something else

Item 1 Item 1 Item 1 Item 1 Item 1

(See more)



Because he is logged in with an SFSU Email, he can access the exclusive deals.

Logo

Gator Savvy

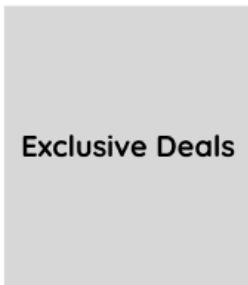
Exclusive Deals

Search for items...

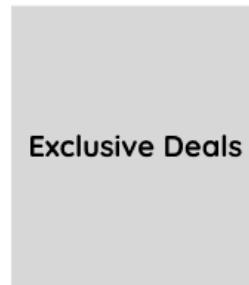


Sell

Username

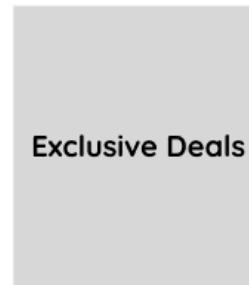


Item 1
Price



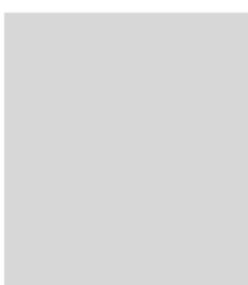
Exclusive Deals

Item 1
Price

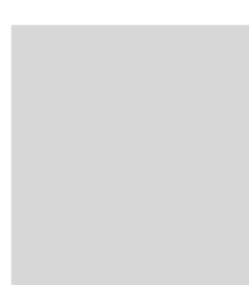


Exclusive Deals

Item 1
Price



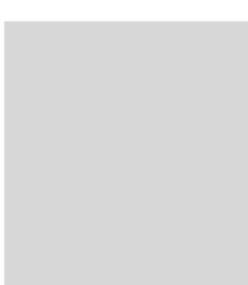
Item 1
Price



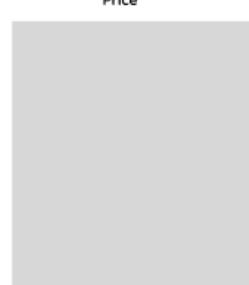
Item 1
Price



Item 1
Price



Item 1
Price



Item 1
Price



Item 1
Price

(See more)

FOOTER, contact us, FAQ, and more...

The filter and text would then be changed to find the rest of his items.

Description/Contents of pages

Homepage - Student View

- Header with search bar, search filters, login/signup, and sell button(if logged in)
- Featured deals carousel
- Categories (Food, Clothing, Tech, Textbooks, etc.)
- Recommended items/deals section

Item Browsing Page

- Dropdown with filter options (Category, Expiration Date, Location)
- List of items with thumbnails, locations, and prices
- Load More button

Item Details Page

- Large image of the deal
- Item name and seller info (name/badge/ratings)
- Price, Condition, Location
- Add to cart button, rent(if available)
- Chat with seller

Profile Page

- Profile photo, name, and school email
- Saved Deals section
- Deal History (Previously claimed deals)
- Overview of active and expired deals
- Edit or Delete Existing Deals

Login Page

- SFSU email
- Password
- Login button
- Forgot Password button
- Don't have an account button (Signup page link)

Signup Page

- First name
- Last name
- Username
- SFSU email
- Password
- Repeat password
- Create account button

Sell/Post item Page

- Item name
- Categories
- Price
- Availability to rent
- Image
- Condition/Expiry date
- Location
- Description

Error / No Deals Found Page

- Simple message: "No deals found. Check back later or explore other categories."
- Suggested categories or deals

NOTE: These wireframes will be used as the foundation for the visual design and development stages of Gator Savvy. Each interface will prioritize a clean, intuitive user experience tailored to SFSU students and vendors.

5. High-level Architecture, Database Organization summary only

Database Organization

Below is the list of the main database tables and the columns (attributes) each table contains, consistent with the naming conventions from Section 2.

User

- user_id (Primary Key)
- username
- full_name
- email (Must be a valid SFSU email for verification)
- Password_hash (Encrypted password)
- role (e.g., "Registered User", "Admin")
- is_verified (Boolean to check if the user is verified)
- created_at (Timestamp of user creation)
- updated_at (Timestamp of last update)

Profile

- profile_id (Primary Key)
- user_id (Foreign Key to Users table)
- student_id (Unique identifier for students)
- reviews_count (Number of reviews written by the user)
- rating (Average user rating from seller feedback)
- is_faculty
- preferences (e.g., categories the user is interested in) (optional)

Wishlist

- wishlist_id (Primary Key)
- item_id (Foreign Key to Item_Listing table)
- user_id (Foreign Key to User table)
- created_at (Timestamp when the wishlist items were added)

Item_Listing

- item_id (Primary Key)
- user_id (Foreign Key to User table)
- name
- description
- price
- category_id (Foreign Key to Category table)
- location
- quality (e.g., "new", "used", etc.)
- rental_option (Boolean: indicates if the item can be rented)
- created_at
- updated_at
- is_active (Boolean: indicates if the listing is active)

Review

- review_id (Primary Key)
- item_id (Foreign Key to Item_Listing table)
- user_id (Foreign Key to User table)
- created_at (Timestamp when the review was issued)
- updated_at (Timestamp for when the review was last updated)
- rating
- review_description

Verification_Badge

- badge_id (Primary Key)
- user_id (Foreign Key to User table)
- badge_type (Type of badge, e.g., "Verified Seller")
- created_at (Timestamp when the badge was issued)
- updated_at (Timestamp for when the badge status was last updated)

Category

- category_id (Primary Key)
- category_name

Media Storage

We'll use AWS S3 buckets for storing images, videos, and user profile photos. Plan to connect S3 buckets to an AWS RDS database and to the EC2 instance created.

Search/filter architecture and implementation

The system will search through specific columns using SQL queries with a %LIKE filter to ensure relevant results are returned quickly and accurately.

The following columns from the Item_Listing table will be used for the search:

- Category: Exact match using the category column in the Item_Listings table.
- Item Name: Substring search using %LIKE on the name column.
- Description: Substring search using %LIKE on the description column.

Changes to SW Tools/Frameworks

- Server Host: Amazon AWS EC2
- Operating System: Ubuntu Server 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)
- [[**NEW**]] Database: Amazon AWS RDS (MySQL Ver 8.0.41-0ubuntu0.24.04.1 for Linux on x86_64)
- [[**NEW**]] Storage: Amazon AWS S3 buckets
- Web Server: NGINX 1.26.3
- Server-Side Language: Python 3.12.3
- Web-Side Language: JavaScript
- Backend Framework: Flask 3.0.2
- [[**NEW**]] Backend Library: SQLAlchemy (handling MySQL in Python), bcrypt (encrypting user passwords)
- Frontend Framework: Vue.js v3.5.13, Vue Router v4.2.5
- IDE: PyCharm, VS Code
- Web Analytics: Google Analytics
- SSL Cert: Let's Encrypt (Cert Bot)
- Documentation: Doxygen (1.13.2)
- Supported Browsers: Safari 18.3 & Chrome LTS-126

6. Identify actual key risks for your project at this time

Skill risks

Our team has a slight gap in each field, and it would be hard to bring everyone up to the same skill level for this development. The way to close this gap is to have the experience to work with other members of the team, and finish it fast and well while practicing using our software stack.

Schedule risks

At this particular time, the spring break has set our team apart from school for a whole week. It will be hard to find everyone's availability for a specific meeting time. Our solution to this is to have a small 13-15-minute quick catch-up session for the progress done each day, and that ensures we keep up with the schedule and submission.

Technical risks

The biggest technical risk is our AWS EC2 instance at this moment. An incident happened where two EC2 instances were running at the same time, causing our available free tier hours to be limited, estimating less than 5 days of total hours available. Our solution to this is to use whatever we have, do development on localhost, and update the server on April 1st. Since we are

Teamwork risks

Our teamwork risks are several issues combined above. One missing component could slow our progress from meeting the deadline on time, and our members could get nervous. Our solution is to cross-check each other's changes and progress at the end of the day, making sure that everyone is on track and comfortable with discussing changes.

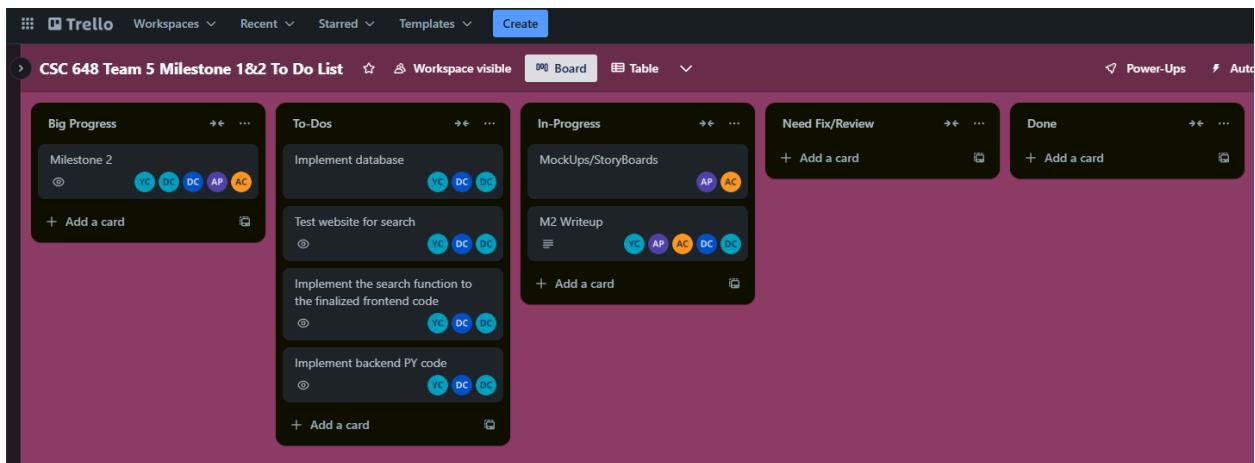
Legal/content risks

One risk is the liability of buyers/sellers who use the marketplace. Since we are not legal experts, one thing we can aim to do is take inspiration from the other marketplace platforms and mimic a version of legal statements ourselves.

7. Project management

We are using Trello to manage our project. Here is our plan:

- Frontend
 - Fine-up mockups and storyboards
 - Finalize use cases using
 - Finish write-up (before 3/24)
- Backend
 - Help M2 Pt1 (before 3/24)
 - Finalize DB contents and variable (in Pt1, before 3/24)
 - Test out code on localhost before April 1st (Google Colab for temporary collab coding)
 - After April 1st, EC2 refreshes and pushes updates to the server for submission
- GitHub Master
 - Change credential information after EC2 instance refresh & stable IP



8. Use of GenAI tools like ChatGPT and Copilot

AI models used:

ChatGPT (GPT-4o) - HIGH

ChatGPT:

We used ChatGPT for the following tasks

Task 1: Generating some example entities

Helpfulness: Somewhat helpful

Description:

It is nice to have some examples of what entities to include in our database. However, some of the items from GPT's response were too off-topic. We are still able to edit the listed items to meet our team's requirements.

Task 2: Generating a summary for #5's instruction

Helpfulness: Helpful

Description:

We throw in the instructions for #5, let GPT summarize the paragraph for us, and list the items we need to complete. It was great that GPT is good at summarizing paragraphs, saving us more time reading the instructions.

9. Team Lead Checklist

- So far, all team members are fully engaged and attending team sessions when required. **DONE**
- Team is 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**.
- The team reviewed suggested resources before drafting Milestone .2 **DONE**
- The team lead checked the Milestone 2 document for quality, completeness, formatting, and compliance with instructions before the submission. **DONE**
- The team lead ensured that all team members read the final Milestone 2 document and agreed/understood it before submission. **DONE**
- The team shared and discussed the experience with GenAI tools among themselves.s **DONE**

Milestone 3

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

Team number: 5

Meeting date: Monday, April 21st, 2025

- Summary of feedback on UI (record all pages that need revision)
 - → Need asterisks for required items in login/signup
- Summary of feedback on code and architecture
 - Don't make liked items into an array or a comma-separated list of liked IDs.
→ Make a "likes" table.
 - The search code can be simplified.
 - Server code cannot be in one file.
 - Add the images to the results and postings
- Summary of feedback on GitHub usage
 - Commit counts need some work.
- Summary of feedback on DB
 - Listings are missing an image column/table
 - Branch usage is ok, but can be better
- Summary of feedback on teamwork and risk management
 - Separate jobs & set checkpoints
- Confirm that you have done an architecture review to check that developers adhere to the MVC pattern, coding style, and minimal agreed-upon documentation → **DONE**.
- List below the agreed-upon P1 list of features for final delivery, which constitutes the product plan.
 - Guests shall browse and search for items.
 - Guests shall filter item listing search results by category, price, and more.
 - Guests shall sign up for a Gator Savvy account (SFSU email required).
 - Registered users shall post item listings for sale or rent.
 - Registered users shall filter item listing search results by category, price, etc.
 - Registered users shall be verified and receive a verification badge.
 - Registered users shall set up a profile with a username and picture.
 - Registered users shall message each other inside the app.
 - Admins shall review and remove item listings.

“SW Engineering CSC648-848 Spring 2025”

Gator Savvy

Team 5

Yu-Ming Chen (Team Lead) - ychen88@sfsu.edu

David Cabanella (GitHub Master) - dcabanel@mail.sfsu.edu

David (Jun) Chang (Backend Lead) - dchang6@mail.sfsu.edu

Alexis Perez (Frontend Lead) - aperez75@sfsu.edu

Athan Cheung (Frontend) - acheung4@mail.sfsu.edu

Milestone 4

History

	Date Submitted	Date Revised
Group meeting & started doc		5/1/2025
Group meeting		5/4/2025
Group meeting		5/11/2025
Group meeting + Demo prep		5/15/2025
Group meeting + Demo prep		5/17/2025
Group meeting + Final wrap up	5/18/2025	5/18/2025
Presentation day	5/19/2025	5/19/2025

1) Product Summary

Name of our product: Gator Savvy

Product Description: Gator Savvy is a marketplace web application for SFSU students and faculty to buy, sell, rent items, and get exclusive student deals. Whether it's textbooks, clothes, tech, or everyday essentials, students can find what they need at affordable prices from other students or SFSU-affiliated groups/services.

Itemized List of All Major Committed Functions

1. Guests can browse and search for items.
2. Guests can filter item listing search results by category, price, and more.
3. Guests can sign up for a Gator Savvy account (SFSU email required).
4. Registered users can post item listings for sale or rent.
5. Registered users can filter item listing search results by category, price, and more.
6. Registered users can build credibility by requesting a verification badge.
7. Registered users can set up a profile with a username and picture.
8. Registered users can message each other inside the app.
9. Admins review and approve item listings before they show up.

URL to Gator Savvy: <http://35.94.172.233>

2) Usability test plan for a selected function

ONE major function that will be tested for usability is the **SEARCH** function.

1. Test objectives:

We will be testing the usability of the search function of our app. The goal is to observe whether users can effectively apply filters, navigate search results, and complete searches without confusion. We also want to measure user satisfaction and identify any usability issues that make the search difficult or frustrating (e.g., unclear filter options, complicated workflows).

2. Test the background and setup

System setup:

The app can be tested on our deployment server IP or locally on one of the developers' computers. Users/Testers can receive the IP address for testing. Our app runs on standard web browsers like Google Chrome or Firefox.

Starting point:

Users/Testers will start at our home page (/). From there, they can go to the search bar located on top of the home page and in the center of the navigation bar. Users do not have to be logged in to search for items.

Hardware requirements:

Testers will need access to internet connections and a web browser, either on a computer or a mobile device (if accessing the app via the deployment IP).

Intended users:

The intended users are students or faculty from San Francisco State University, with email addresses ending in (@sfsu.edu or @mail.sfsu.edu), who are interested in posting or browsing textbooks, clothing, tech items, or everyday essentials.

URL of the system to be tested:

<http://35.94.172.233/>

Test environment:

The tests can be conducted at home (a typical environment where Gator Savvy would be used). No cameras are necessary. There will be no monitoring since testers will provide feedback via a questionnaire after the test is complete. Training will not be provided since we want to test whether users can use the search function naturally without any guidance.

3. Usability Task Description:

Tasks (Specific & Measurable):

1. Search for an item you would normally look for (like a textbook or a calculator).
2. Apply a category filter (like "Electronics" or "Books") to your search.
3. Change the filter and try searching for a different item.

Feedback after performing the required task:

- Did the user find something? Something relevant?
- Did the filter work properly?
- Was the change in filter/search successful?

Task	Description
Task	Search for an item on the website with the search function
Machine State	On the main page, not logged in
Successful completion criteria	Results display relevant items matching the search query
Benchmark	Completed in under 1 minute

4. Plan for evaluation of Effectiveness

We will measure effectiveness by checking if users can complete the search tasks. A task is considered completed and effective if:

1. The user finds an item or items that match their search
2. The category filter returns the correctly filtered results
3. The user shall switch filters and search again without any issues

5. Plan for Evaluation of Efficiency:

We will measure efficiency by tracking how long it takes users to complete each task and how easily they navigate the process.

- Measure how long it takes users to finish tasks.
- Track the time spent searching, applying filters, and changing them.
- Observe if users make extra or unnecessary clicks or look confused.
- Expect tasks to be completed in under 1 minute with minimal confusion.

We expect users to complete each task quickly (less than 1 minute). If the user experiences any delays or issues, that would mean that the function is not very efficient.

6. Plan for Evaluation of user satisfaction (Likert scale questionnaire):

The search function was easy to use.

1. Strongly Disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly Agree

Using a filter made it easier to find what I was looking for.

1. Strongly Disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly Agree

It was easy to switch between different filters.

1. Strongly Disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly Agree

The search results were relevant to what I was looking for.

1. Strongly Disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly Agree

Overall, I am satisfied with my experience using the search function.

1. Strongly Disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly Agree

7. GenAI use:

ChatGPT - HIGH

Task: I asked ChatGPT about the usability task description and Likert Scale Questionnaire. It helped with wording, structuring/organizing the sections, and gave feedback on what to include and improve. Very helpful.

3) QA test plan and QA testing

1. Test objectives: - What is being tested

We will test whether the search function works properly. This includes checking whether users can search for items, apply filters like category, and get correct results. We also want to make sure it doesn't break or show errors.

2. HW and SW setup (including URL):

HW: Any laptop or desktop with internet

SW: Google Chrome or Firefox browser (Latest Version)

URL: <http://35.94.172.233/>

3. Feature to be tested

We will be testing the **search** feature. Search bar, filters, and results.

4. QA Test plan

The 2 major WWW browsers we chose are Google Chrome and Firefox. Both passed the test with flying colors. Everything worked as expected.

Test Title	Description	Input	Expected	Chrome	Firefox
Blank search	Searching with nothing	Query: ""	Show all items	PASS	PASS
Search w/o filter	Searching for something without a filter	Query: "Camera" Filter: n/a	Show related items to the search query	PASS	PASS
Search with filter	Searching for something with a filter	Query: "Camera" Filter: "Electronics"	Show filtered plus matching items to search	PASS	PASS
No match	Searching for something we don't have	"testingtesting"	Display "Results (0)"	PASS	PASS

4) Peer Code Review

Jim is requesting code review of the search functionality via the GitHub pull request link in the email:

[M4 Code Review Request] Search Feature

 **Yu-Ming Chen**          

To:  David B Cabanela Thu 5/15/2025 10:23 AM

Hi David,

I've pushed code changes for the search feature (search.py, Searchbar.js, and Results.js) to a new branch: `search_m4_code_review`.

Here's the GitHub pull request link:

<https://github.com/CSC-648-SFSU/csc648-fa25-0104-team05/pull/4>

Can you review it on GitHub and send me a quick summary email after? We'll use the screenshots for the M4 submission.

Thanks!

Jim

David, the reviewer, responded to the email with a summary of code review feedback:

[M4 Code Review Request] Search Feature

 **David B Cabanela**          

To:  Yu-Ming Chen Sat 5/17/2025 8:17 PM





Hi Jim,

I have completed the code review and reviewed your pull request on GitHub. I left a comment on GitHub for you to check out. Below is a summary of the code review feedback. I have also attached screenshots with more detailed comments for your reference.

Code Review Summary

Overall, great job on this search functionality feature. Here is a summary of feedback including kudos and improvements you can make to the code for this feature:

What's working well:

- Clear, readable code with intuitive variable names, proper indentation, and Python method docstrings.
- You properly follow the naming convention established in the Data section in Milestone 2.
- Good separation of concerns by separating search logic into dedicated .py and .js files.

Recommendations for improvement:

- Consider removing some of the in-line comments explaining obvious logic to reduce redundancy.
- Add a brief header comment to the top of each code file to briefly summarize its purpose.
- In the future, we can dynamically search for categories in the Category database table instead of hardcoding it.
- We can more intuitively group and organize import statements.

Hope this helps! Let me know if you have any questions.

Best regards,
David

...

David's code review comments within the GitHub pull request:

Prepared search functionality code for peer review #4

 JimboRumbleCad... wants to merge 1 commit into `dev` from `search_m4_code_review` 

Conversation 2 Commits 1 Checks 0 Files changed 3

JimboRumbleCadumbo commented 3 days ago

M4 Peer Code Review: Prepared search.py, Searchbar.js, and Results.js for review.

Prepared search functionality code for peer review 

davidcabanela reviewed 15 hours ago 

davidcabanela left a comment • edited

Overall, great job on this search functionality feature. Here is a summary of feedback including kudos and improvements you can make to the code for this feature:

What's working well:

- The code is very readable with intuitive variable names and proper indentation.
- The docstrings at the beginning of each Python method are also helpful.
- You properly follow the naming convention established in the Data section in Milestone 2 (e.g. Item_Listing for marketplace items).
- It's great that you separated out the search logic into dedicated.py and .js files instead of combining everything into a single, giant Flask app.py. This really improves maintainability and modularity.
- Git commit comments are concise and informative.

Recommendations for improvement:

- Consider removing some of the in-line comments sprinkled across the code that explain straightforward code. It would be better to only include comments to explain non-obvious or unconventional logic.
- Maybe we can add a header comment to the top of the file to briefly summarize the overall purpose of the code for that file at a glance.
- In the future, we can enhance flexibility by dynamically search for categories in the Category database table instead of hardcoding it at the top of search.py.
- Something minor, but we can more intuitively group the import statements at the top of each Python file more intentionally (e.g. by standard library, third-party packages, local modules).



Screenshots of more specific code review comments from David are included in the email reply to Jim, which is included in the next few pages.

Code review comments for search.py:

Prepared search functionality code for peer review #4

Changes from all commits ▾ File filter ▾ Conversations ▾ ▾ ▾

Filter changed files

application/main/search.py

```

1  ... @@ -1,3 +1,5 @@
2  1 +#Please code review this file
3  2 +
4  3   from flask import Flask, request, jsonify
5  4   from flask_mysqldb import MySQL
6  5   from dotenv import load_dotenv
7  6   import MySQLdb.cursors
8  7   import os
9  8   import base64
10 9   ...
11 10  # Category constants for better readability and maintainability
12 11  CATEGORY_IDS = {
13 12  'Electronics': 1,
14 13  'Books': 2,
15 14  'Clothing': 3,
16 15  'Furniture': 4,
17 16  'Sports Equipment': 5
18 17 }
19 18
20 19 def init_search_routes(app):
21 20     """
22 21     Initialize search routes and MySQL configuration for the Flask application.
23 22
24 23     Args:
25 24         app: Flask application instance
26 25
27 26     # Get the shared MySQL connection
28 27     mysql = app.config.get('MYSQL_CONNECTION')
29 28     if not mysql:
30 29         # Fallback if shared connection is not available
31 30         load_dotenv()
32 31
33 32         # Configure MySQL connection
34 33     app.config['MYSQL_HOST'] = os.getenv('MYSQL_HOST')
35 34     app.config['MYSQL_USER'] = os.getenv('MYSQL_USER')
36 35     app.config['MYSQL_PASSWORD'] = os.getenv('MYSQL_PASSWORD')
37 36     app.config['MYSQL_DB'] = os.getenv('MYSQL_DB')
38 37
39 38     mysql = MySQL(app)
40 39
41 40     @app.route('/api/search', methods=['POST'])
42 41     def api_search():
43 42     """

```

Consider using the Category table instead of hardcoding for flexibility

Prepared search functionality code for peer review #4

Changes from all commits ▾ File filter ▾ Conversations ▾ ▾ ▾

Filter changed files

application/main/search.py

```

39 41     def api_search():
40 42         """
41 43             API endpoint for searching items with optional filtering by category.
42 44
43 45             Returns:
44 46                 JSON response with search results
45 47
46 48             data = request.get_json()
47 49             query = data.get('query', '')
48 50             filter_by = data.get('filter', '')
49 51
50 52             cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
51 53
52 54             try:
53 55                 # Build the SQL query based on filter
54 56                 if filter_by == 'all':
55 57                     sql_query = "SELECT * FROM Item_Listing"
56 58                     params = None
57 59
58 60                     if query:
59 61                         sql_query += " WHERE name LIKE %s"
60 62                         params = ('%' + query + '%')
61 63
62 64                 # Get category ID from constants
63 65                 category_id = CATEGORY_IDS.get(filter_by)
64 66                 if category_id is None:
65 67                     return jsonify({"error": "Invalid category"}), 400
66 68
67 69                 sql_query = "SELECT * FROM Item_Listing WHERE category_id = %s"
68 70                 params = (category_id, '%' + query + '%')
69 71
70 72                 if query:
71 73                     sql_query += " AND name LIKE %s"
72 74                     params = (category_id, '%' + query + '%')
73 75
74 76                 # Execute query with or without parameters
75 77                 if params:
76 78                     cursor.execute(sql_query, params)
77 79                 else:
78 80                     cursor.execute(sql_query)
79 81
80 82                 results = cursor.fetchall()
81 83
82 84                 # Process the results to convert BLOB images to base64
83 85                 processed_results = []
84 86                 for item in results:
85 87                     # Create a new dictionary with the same keys but without the BLOB data
86 88                     processed_item = {}
87 89                     for key, value in item.items():
88 90                         if key == 'image' and value:
89 91                             try:
90 92                                 # Convert BLOB to base64 string
91 93                                     ...

```

This in-line comment explains straightforward code, can remove

Proper usage of Data Entity naming from M2. Nice!

Code review comments for Searchbar.js:

github.com/CSC-648-SFSU/csc648-fa25-0104-team05/pull/4/files

Prepared search functionality code for peer review #4

Changes from all commits ▾ File filter ▾ Conversations ▾

Filter changed files

application/main/search.py

```

1 + // Please code review this file.
2 +
3 export default {
4     template:
5         <!-- Search Bar and Filters Dropdown in Flexbox -->
6         <div class="search-bar-container">
7             <!-- Filters Dropdown -->
8             <div class="dropdown" :open="isDropdownVisible">
9                 <button @click="toggleDropdown" class="dropdownbtn">{{ selectedFilter || "All" }}</button>
10                <div class="dropdown-content">
11                    <div v-for="filter in filters" :key="filter" @click="selectFilter(filter)">
12                        {{ filter }}
13                    </div>
14                </div>
15            </div>
16
17            <!-- Search Bar -->
18            <div class="search-bar">
19                <input type="text" v-model="searchQuery" placeholder="Search..." @keyup.enter="submitSearch" />
20                <button @click="submitSearch">Search</button>
21            </div>
22        </div>
23    </div>
24
25    setup() {
26        const searchData = Vue.inject('searchData');
27        const searchQuery = Vue.ref('');
28        const selectedFilter = Vue.ref('');
29        const isDropdownVisible = Vue.ref(false);
30        const router = useRouter();
31
32        function submitSearch() {
33
34            fetch('/api/search', {
35                method: 'POST',
36                headers: { 'Content-Type': 'application/json' },
37                body: JSON.stringify({
38                    query: searchQuery.value,
39                    filter: selectedFilter.value || 'All'
40                })
41            })
42            .then(res => {
43                if (!res.ok) {
44                    return res.json().then(err => {
45                        throw new Error(`Server error: ${err.error || res.statusText}`);
46                    })
47                }
48            })
49            .then(res => {
50                if (!res.ok) {
51                    return res.json().then(err => {
52                        throw new Error(`Server error: ${err.error || res.statusText}`);
53                    })
54                }
55                return res.json();
56            })
57            .then(data => {
58                searchData.results = data;
59                console.log('Search results:', data); // Update global state
60                router.push('/results'); // Navigate to results page
61            })
62            .catch(error => {
63                console.error('Search error:', error);
64                alert(`Search failed: ${error.message}`);
65            });
66        }
67
68        return {
69            searchQuery,
70            selectedFilter,
71            isDropdownVisible,
72            filters: ['All', 'Electronics', 'Books', 'Clothing', 'Furniture', 'Sports Equipment'],
73            submitSearch,
74            toggleDropdown() {
75                isDropdownVisible.value = !isDropdownVisible.value;
76            },
77            selectFilter(filter) {
78                selectedFilter.value = filter;
79                isDropdownVisible.value = false;
80            }
81        };
82    }
83}

```

application/main/static/components/Searchbar.js

Add brief header summary comment to top of file

application/main/static/pages/Results.js

In future iterations, consider not hardcoding, fetch from DB

Code review comments for Results.js:

The screenshot shows a GitHub code review interface for a file named `Results.js`. The file is located in the `application/main/static/pages` directory. The code is a Vue.js component for displaying search results. A specific line of code, line 39, has a review comment: "Consider utilizing a reusable footer component for modularity". An orange callout bubble points to this comment. The code itself includes imports for `search.js`, `useRoute()`, and `Vue`, and defines a `Results` component with a template containing a header, a filter section with price range inputs, and a product grid. The grid iterates over `filteredResults` and links to individual item details.

```
... @@ -1,3 +1,5 @@
  + // Please code review this file.
  +
1   3   console.log("Dashboard.js Loaded!");
2   4
3   5   export default {
4   6     template: `
5   7       <Navbar></Navbar>
6   8       <div class="page-wrapper">
7   9         <div class="container">
8  10           <div class="content">
9  11             <h1> Results <span class="result-count">{{ filteredResults.length }}</span> </h1>
10 12
11 13           <div class="filter-section">
12 14             <label>Price Range:</label>
13 15             <input type="number" v-model="minPrice" placeholder="min" />
14 16             <span>-</span>
15 17             <input type="number" v-model="maxPrice" placeholder="max" />
16 18
17 19             <button @click="applyFilter">Apply Filter</button>
18 20             <button @click="clearFilter">Clear Filter</button>
19 21
20 22
21 23           <!-- Product Grid -->
22 24           <div class="result-product-grid" v-show="filteredResults.length > 0">
23 25
24 26             <router-link v-for="result in filteredResults"
25 27               :key="result.id"
26 28               :to="'/item?id=' + result.item_id"
27 29               class="card-link"
28 30             >
29 31               <div class="result-card">
30 32                 
31 33                 <h3>{{ result.name || result.title }}</h3>
32 34                 <p>${{ result.price || result.cost }}</p>
33 35               </div>
34 36             </router-link>
35 37
36 38           </div>
37 39         </div>
38 40       </div>
39 41       <footer class="footer">
40 42         <p>&copy; 2025 CSC 648 Team 05. All rights reserved.</p>
41 43         <router-link to="/about" class="footer-link">About</router-link>
42 44       </footer>
43 45     </div>
44 46   `,
45 47   setup() {
46 48     //Added to make sure searchresults are reset if you hit home page
47 49     const route = useRouter().useRoute();
48 50     const searchData = Vue.inject("searchData");
```

Consider utilizing a reusable footer component for modularity

GenAI Use for Code Review

- **Tool:** ChatGPT, **Version:** GPT-4-turbo
- **Task:** I used this tool to perform a code review of the same 3 files for the search functionality. I provided context of what our project is about and gave instructions on what to pay attention to in the code review, such as naming conventions and code comments.
- **Benefits:** Having ChatGPT review the code was a good way to double-check our work, like having a different set of eyes in case we missed something. It also structured the code review similarly to how David did it (e.g., what's working well section, improvements section), which was a good indicator that we are on the right track in terms of performing code reviews within the team.
- **Example prompt:**
 - *"I'm working on a team project for my software engineering capstone class. This project is a marketplace web app where students from my university can buy and rent items. One of the key features of this webapp is the search functionality to allow users to browse and filter item listings. Perform a code review of the following code for this feature. Code review must include checking for basic header and in-line comments, as well as proper and consistent class/methods/variable names, and consistency with naming established in the DATA section of Milestone 2."*

Here is the naming established in milestone 2: [included bulleted list]

There are 3 code files for you to review: search.py, Searchbar.js, and Results.js. [shared code files]"

- Example response from ChatGPT:

 search.py (Flask Backend)

What's Good

- Header comment and docstrings: Present and descriptive.
- Modular route definition: `init_search_routes()` is good practice.
- Secure parameterized queries: Avoids SQL injection.
- Image handling with `base64`: Nice touch for frontend use.

Suggestions for Improvement

- Naming Consistency: Use `item_name` instead of `name` when filtering (Milestone 2 defines “Item Name”).
 - Error Handling: Good use of `try/except`, but logging can be cleaner and centralized.
 - Inline Comments: Add inline comments, especially for SQL query construction.
 - Hardcoded Categories: Consider dynamically loading these from a Category table to stay DRY.
-
- **Rank of GenAI Utility: MEDIUM** – while it was a good exercise to have ChatGPT review the code, a lot of the review points were similar to what David had already commented (e.g., dynamically retrieving categories). The GenAI tool also didn’t have access to the full codebase, and having that fuller context might have yielded more insightful feedback. Also, some feedback was inconsistent with what we learned in class. For example, we learned only to include inline code comments for unconventional logic, but ChatGPT recommended adding inline comments for SQL query construction, which should be relatively obvious logic that doesn’t necessarily need comments.

5) Self-check on best practices for security

Major assets we are protecting in our team application:

- Password, item posting, server storage, SQL injection

The major threats to each asset above:

- Password: Easy to brute force without extra encryption
- Item posting: Bypass law enforcement, selling illegal items
- SQL injection: Having root access to the database or even the server kernel

Consequences of a security breach:

- Personal data breach; potential legal infringement; lose access to server

Methods of protection

- Password: We encrypted our password using bcrypt, so it will be impossible for hackers to brute force the password
- Item posting: Admin approval will ensure the item is legal and contains no potential harm to people
- SQL injection: Passed SQL injection detection tool - sqlmap's highest level & risk injection test on all pages.

Input data validation:

- The search bar can accept input of up to 40 alphanumeric characters.
- The customer's e-mail must include “@sfsu.edu” or “@mail.sfsu.edu” at the end.
- Acceptance of terms is provided in the registration form as a required item.

6) Self-check of the adherence to original Non-functional specs – performed by team leads

Our list of non-functional requirements:

1. The application shall be developed, tested, and deployed using tools and cloud servers approved by Class CTO and as agreed in M0 **DONE**.
2. The application shall be optimized for standard desktop/laptop browsers, e.g., it must render correctly on the two latest versions of the two major browsers. **DONE**.
3. All or selected application functions shall render well on mobile devices (no native app to be developed). **DONE**.
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. **DONE**
6. No more than 50 concurrent users shall be accessing the application at any time. **DONE**
7. The privacy of users shall be protected.
8. The language used shall be English (no localization needed). **DONE**
9. The application shall be straightforward to use and intuitive. **DONE**
10. The application shall follow established architecture patterns.
11. The application code and its repository shall be easy to inspect and maintain. **DONE**
12. Google Analytics shall be used.
13. No email clients or chat services shall be allowed. Interested users can only message the sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application. **DONE**.
14. Pay functionality, if any (e.g., paying for goods and services), shall not be implemented or simulated in UI. **DONE**
15. Site security: basic best practices shall be applied (as covered in the class) for the main data item. **DONE**
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. (It is important not to confuse this with a real application.) You have to use this exact text without any editing.
- **DONE**

7) Use of GenAI tools like ChatGPT and Copilot

AI models used:

ChatGPT (GPT-4o-mini-high)

Task 1: Used ChatGPT for debugging messaging. There was a query error, and the error is hard to spot.

Helpfulness: Very helpful. It found the error very quickly and fixed it within 5 minutes.

Task 2: I asked ChatGPT about the usability task description and Likert Scale Questionnaire. It helped with wording, structuring/organizing the sections, and gave feedback on what to include and improve.

Helpfulness: Helpful, saves everyone a lot of time. Helped with wording, gave good examples and feedback, and helped finalize and polish.

Task 3: I asked ChatGPT about adding a bold SOLD text next to the messaging page if the item is detected as sold

Helpfulness: Very helpful, saves tons of time. Even for backend people, it is easy to integrate your code into the frontend even if you don't know front-end code. Seems easier to become a full-stack developer now with the help of GenAI.

Task 4: I asked ChatGPT for assistance on getting the functionality for sign-up working.

Helpfulness: It helped me to implement the functionalities and saved a lot of time. It even helped me with many trials and errors, and how to resolve them.

Task 5: I asked ChatGPT how to connect Google Analytics to track our site's activities.

Helpfulness: It was straightforward and helpful as it told me how to implement it and get it working.

3) Product Screen Shots

Home page & category select

The screenshot shows the Gator Savvy marketplace homepage. At the top, there's a purple header bar with the text "Gator Savvy" and "SFSU Software Engineering Project CSC 648-848, Spring 2025. For Demonstration Only". Below the header is a search bar with "All" selected and a "Search..." placeholder. To the right is a "Login/Signup" button. A yellow sidebar on the left lists categories: All, Electronics, Books, Clothing, Furniture, and Sports Equipment. The main content area features a yellow banner with the text "Welcome to Our Marketplace!" and "Browse items from students at SFSU.". Below the banner, a section titled "Featured Items" displays four products in cards:

- Tennis Racket** - \$150.00: Wilson Pro Staff Tennis Racket
- Bookshelf** - \$120.00: Wooden Bookshelf, 5 feet
- Shoes** - \$89.99: Nike Running Shoes, Size 10
- Camera** - \$1299.99: Canon EOS 90D DSLR

Search result page

The screenshot shows the Gator Savvy search results page. The interface is similar to the home page, with a purple header, search bar, and login/signup button. The main content area is titled "Results (10)" and includes a "Price Range: 0 - max" filter with "Apply Filter" and "Clear Filter" buttons. Below the filter are ten product cards arranged in two rows of five:

Item	Price
Laptop	\$999.99
Book	\$15.99
Jacket	\$49.99
Couch	\$399.99
Football	\$25.99
Smartphones	
Camera	
Shoes	
Bookshelf	
Tennis Racket	

Item info page

Gator Savvy SFSU Software Engineering Project
CSCE 446/546, Spring 2025
For Demonstration Only

All Search... Search

Post Welcome, blue 

Laptop 

Price: \$999.99

Seller: asmith ★★★★ Verified

Categories: Electronics

Condition: NM

Description

Dell Laptop, 16GB RAM

Rent Buy



Gator Savvy SFSU Software Engineering Project
CSCE 446/546, Spring 2025
For Demonstration Only

All Search... Search

Post Welcome, blue 

Laptop 

Price: \$999.99

Seller: asmith ★★★★ Verified

Categories: Electronics

Condition: NM

Description

Dell Laptop, 16GB RAM

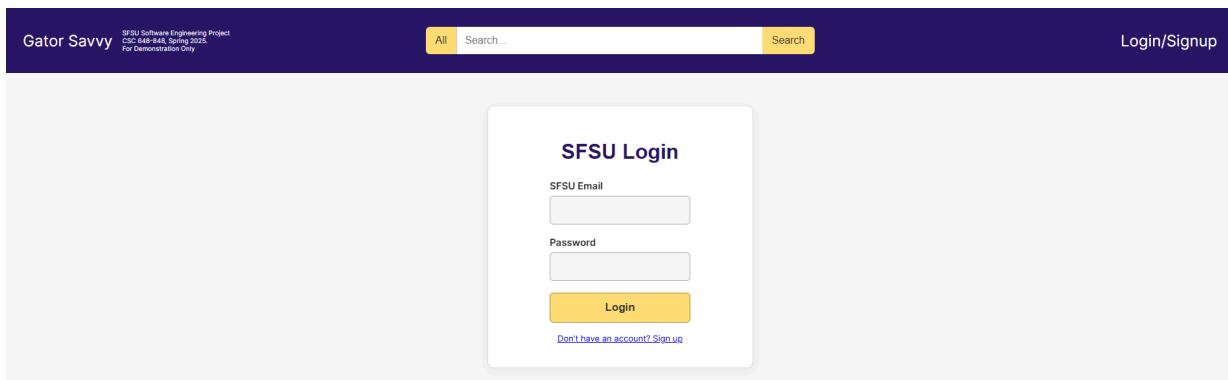
Rent Buy

Seller: asmith X

I want to buy Laptop (\$999.99)

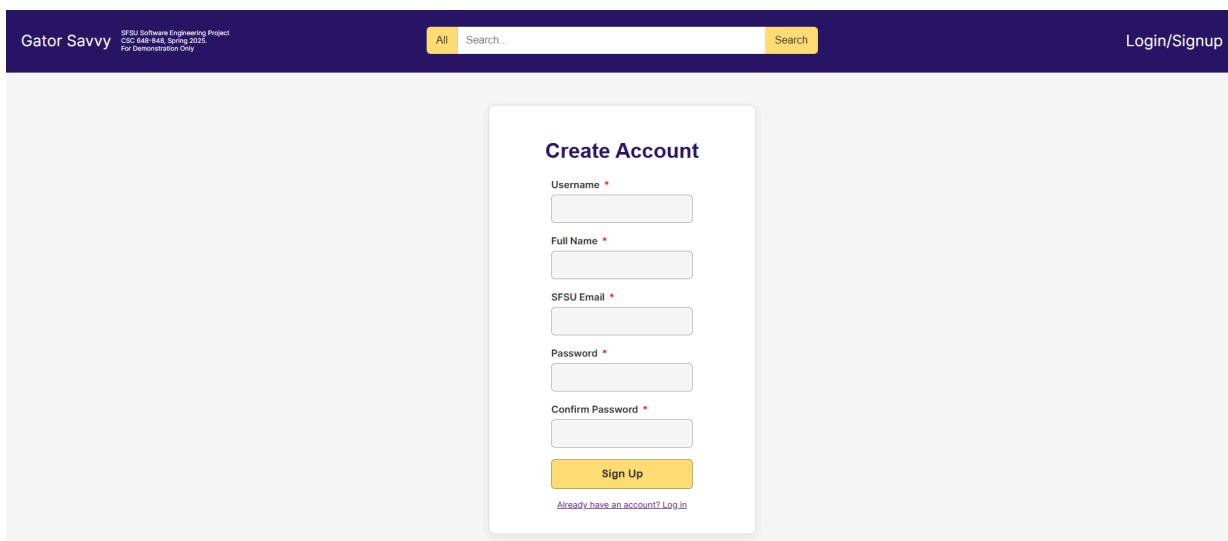


User login page



The screenshot shows the user login interface. At the top, there is a dark blue header bar with the text "Gator Savvy" on the left, "SFSU Software Engineering Project", "SFSE 448-848, Spring 2023", and "For Demonstration Only" in smaller text. In the center, there is a search bar with the placeholder "Search..." and a yellow "Search" button. On the right side of the header, there is a "Login/Signup" link. Below the header, the main content area has a light gray background. In the center, there is a white rectangular form titled "SFSU Login". It contains fields for "SFSU Email" and "Password", each with a corresponding input box. Below these fields is a yellow "Login" button. At the bottom of the form, there is a link "Don't have an account? Sign up".

User signup page



The screenshot shows the user signup interface. At the top, there is a dark blue header bar with the text "Gator Savvy" on the left, "SFSU Software Engineering Project", "SFSE 448-848, Spring 2023", and "For Demonstration Only" in smaller text. In the center, there is a search bar with the placeholder "Search..." and a yellow "Search" button. On the right side of the header, there is a "Login/Signup" link. Below the header, the main content area has a light gray background. In the center, there is a white rectangular form titled "Create Account". It contains fields for "Username *", "Full Name *", "SFSU Email *", "Password *", and "Confirm Password *", each with a corresponding input box. Below these fields is a yellow "Sign Up" button. At the bottom of the form, there is a link "Already have an account? Log in".

Item Posting

Gator Savvy SFSU Software Engineering Project
CSC 646-648, Spring 2025
For Demonstration Only

All Search... Search

Post Welcome, blue 

Create New Item Listing

Item Name *

Price (\$) *

Condition *

Category *

This item is available for rental

Item Image *

Description

User Profile

Gator Savvy SFSU Software Engineering Project
CSC 646-648, Spring 2025
For Demonstration Only

All Search... Search

Post Welcome, blue 

About My Items Liked Items Sold Items Rented Items Messages Settings


blue
Joined: May 06 2025
★★★★★ (0/5)

Description:
My new pic

Users Posted Items

The screenshot shows the 'My Items' section of the Gator Savvy application. At the top, there is a navigation bar with tabs: 'About', 'My Items' (which is highlighted in blue), 'Liked Items', 'Sold Items', 'Rented Items', 'Messages', and 'Settings'. Below the navigation bar, there is a search bar with the placeholder 'Search...' and a 'Post' button. A user profile picture of a green cartoon gator is displayed, along with the text 'Welcome, blue'.

In the main content area, there is a single item listed:

- Random item**: A green cartoon gator icon.
- \$123.00**: Price of the item.
- Delete**: A red button.
- Sold**: A green button.

Users liked Items

The screenshot shows the 'Liked Items' section of the Gator Savvy application. The layout is identical to the 'My Items' section, with a navigation bar at the top and a search bar below it. A user profile picture of a green cartoon gator is displayed, along with the text 'Welcome, blue'.

In the main content area, there are four items listed:

- Jacket**: An image of a black zip-up hoodie. Price: \$49.99.
- Smartphone**: An image of five smartphones in different colors (red, blue, white, pink, green). Price: \$799.99.
- Couch**: An image of a grey sectional sofa in a living room setting. Price: \$399.99.
- Tennis Racket**: An image of a Wilson tennis racket. Price: \$150.00.

Users Sold Items

The screenshot shows the 'Sold Items' section of the Gator Savvy app. At the top, there is a navigation bar with tabs: 'About', 'My Items', 'Liked Items', 'Sold Items' (which is highlighted in blue), 'Rented Items', 'Messages', and 'Settings'. Below the navigation bar, the title 'Sold Items' is centered. A single item card is displayed, featuring a green cartoon gator head icon with a wide smile and the word 'SOLD' in a ribbon banner. The card is labeled 'Random item' and '\$123.00'.

Users Rented Items

The screenshot shows the 'Rented Items' section of the Gator Savvy app. At the top, there is a navigation bar with tabs: 'About', 'My Items', 'Liked Items', 'Sold Items', 'Rented Items' (which is highlighted in blue), 'Messages', and 'Settings'. Below the navigation bar, the title 'Rented Items' is centered. Three items are listed in a grid:

Portable Projector	Electric Scooter	Camping Tent
Return by: 2025-05-25	Return by: 2025-05-20	Return by: 2025-05-22
Portable Projector \$25/day	Electric Scooter \$15/day	Camping Tent \$10/day

Users Messages

The screenshot shows the 'Messages' section of the Gator Savvy application. At the top, there are navigation links: About, My Items, Liked Items, Sold Items, Rented Items, Messages (which is highlighted in blue), and Settings. The main area displays a message thread between two users:

asmith - Laptop dwhite - Couch

dwhite - Couch

I want to buy Couch (\$399.99).
Would you consider a cheaper price?

Type a message...

Users Settings

The screenshot shows the 'Settings' section of the Gator Savvy application. At the top, there are navigation links: About, My Items, Liked Items, Sold Items, Rented Items, Messages, and Settings (which is highlighted in blue). The main area displays the user's profile settings:

Display Name: blue

Profile Icon: No file chosen

Description: My new pic

New Password: Enter new password

4) Database Organization

Category

Name:	Category						
Comment:							
Columns:	+ Add	× Remove	▲ Up	▼ Down			
#	Name	Datatype	Length/Set	Unsign...	Allow NULL	Zerofill	Default
1	category_id	INT		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	category_name	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Item_Listing

Name:	Item_Listing						
Comment:							
Columns:	+ Add	× Remove	▲ Up	▼ Down			
#	Name	Datatype	Length/Set	Unsign...	Allow NULL	Zerofill	Default
1	item_id	INT		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT Primary Key
2	name	TINYTEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
3	description	TINYTEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
4	price	DECIMAL	10,2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0.00'
5	quality	TINYTEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
6	rental_option	TINYINT		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
7	created_at	DATETIME		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	now()
8	updated_at	DATETIME		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
9	status	ENUM	'active','inac...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'inactive'
10	user_id	INT		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
11	category_id	INT		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
12	image	MEDIUMBLOB		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Message

Name:	Message						
Comment:							
<hr/>							
Columns: + Add x Remove ▲ Up ▼ Down							
#	Name	Datatype	Length/Set	Unsign...	Allow NULL	Zerofill	Default
1	sender_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
2	receiver_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
3	text	TINYTEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
4	PK id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
5	created_at	DATETIME		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	now()
6	item_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default

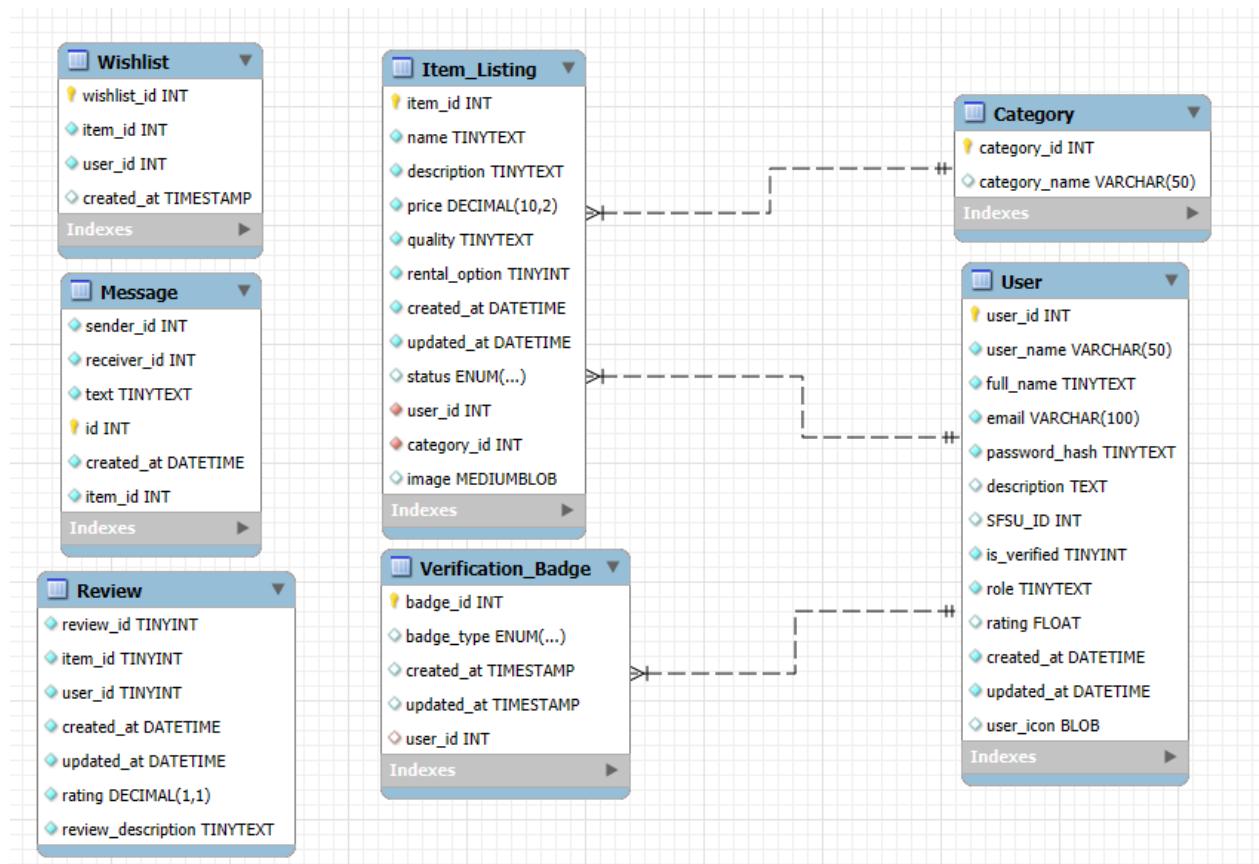
User:

Name:	User						
Comment:							
<hr/>							
Columns: + Add x Remove ▲ Up ▼ Down							
#	Name	Datatype	Length/Set	Unsign...	Allow NULL	Zerofill	Default
1	PK FK user_id	INT		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	PK user_name	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
3	full_name	TINYTEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
4	PK email	VARCHAR	100	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
5	password_hash	TINYTEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
6	description	TEXT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
7	SFSU_ID	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
8	is_verified	TINYINT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'0'
9	role	TINYTEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
10	rating	FLOAT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'0'
11	created_at	DATETIME		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
12	updated_at	DATETIME		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
13	user_icon	BLOB		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Wishlist:

Name:	Wishlist							
Comment:								
Columns:								
+ Add × Remove ▲ Up ▼ Down								
#	Name	Datatype	Length/Set	Unsign...	Allow NULL	Zerofill	Default	C
1	wishlist_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT	
2	item_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default	
3	user_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default	
4	created_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP	

ER diagram for our DB:



5) GitHub organization

a) Main branches in our team's GitHub:

- i) frontend
- ii) backend
- iii) dev
- iv) main

b) Who had access to the main GitHub branch?

- i) Jim – Team Lead
- ii) David – GitHub Master

c) Screenshot of the Team GitHub Home page

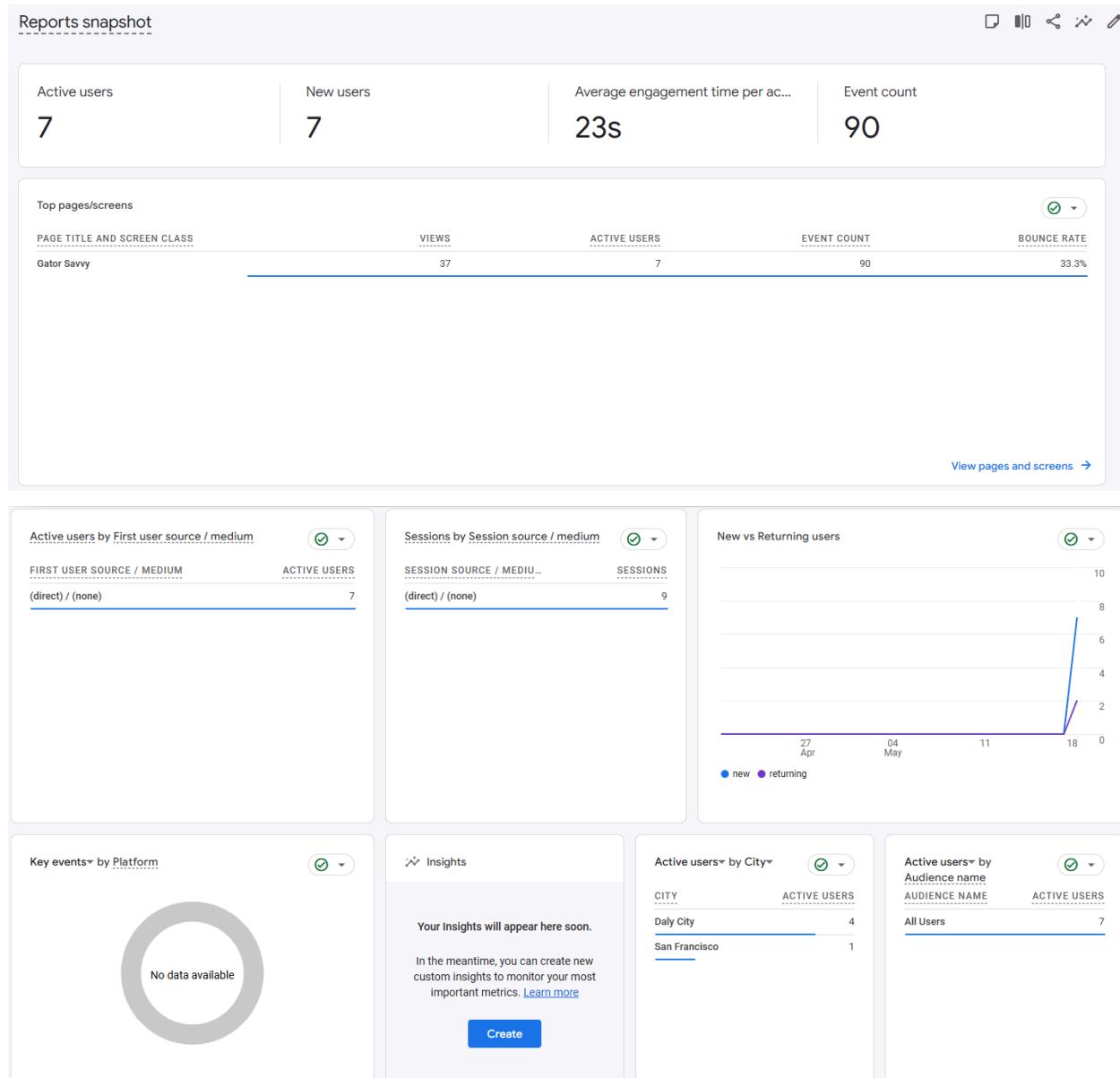
The screenshot shows the GitHub repository 'csc648-fa25-0104-team05'. The repository is private and was forked from 'CSC-648-SFSU/csc-648-sfsu-2025-fall-souza-01-csc648-base-repo'. The main branch has 275 commits ahead of the main branch. The commit list includes:

Commit	Message	Date
Milestones	Added M4 doc	yesterday
application	Doxygen file modification for misses	yesterday
credentials	Credentials updated	last month
.gitignore	add node modules to gitignore	last month
LICENSE	Initial commit	3 months ago
README.md	Update README.md	3 months ago
dump.sql	New dump, added comments for Doxygen	yesterday
package-lock.json	Testing Single page render format with Vue router	2 months ago

About
csc-648-sfsu-2025-fall-souza-01-csc648-base-repo created by GitHub Classroom
35.94.172.233
Readme
MIT license
Activity
Custom properties
1 star
0 watching
0 forks

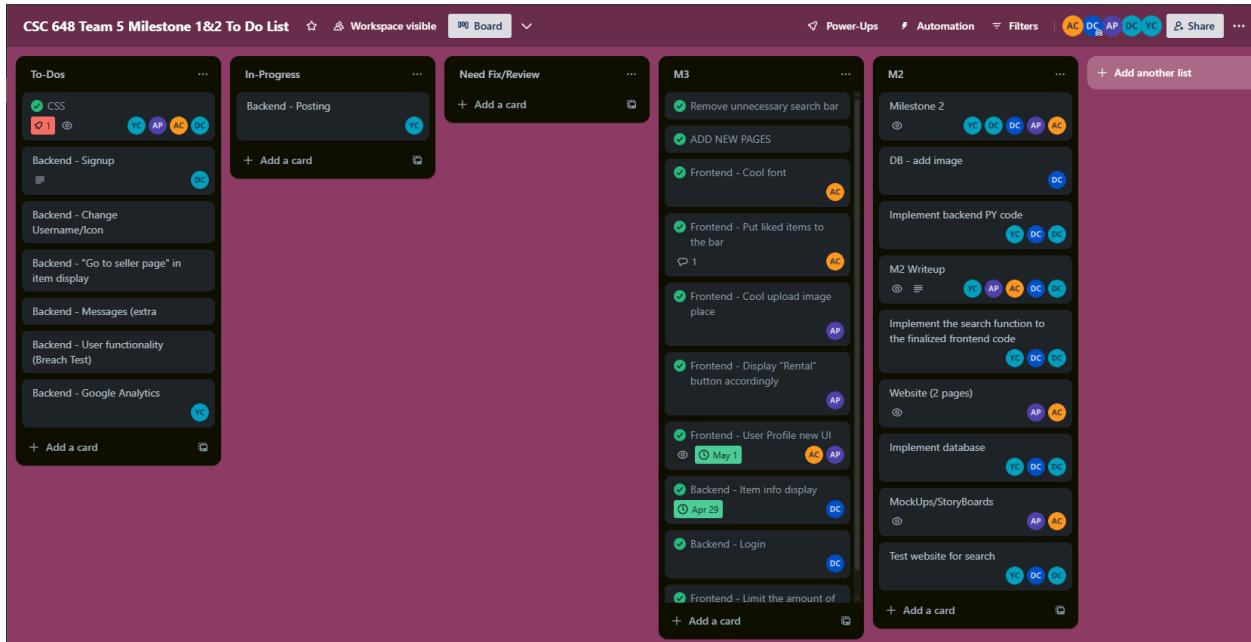
Releases
No releases published
Create a new release

6) Google Analytics stats plot for our WWW site

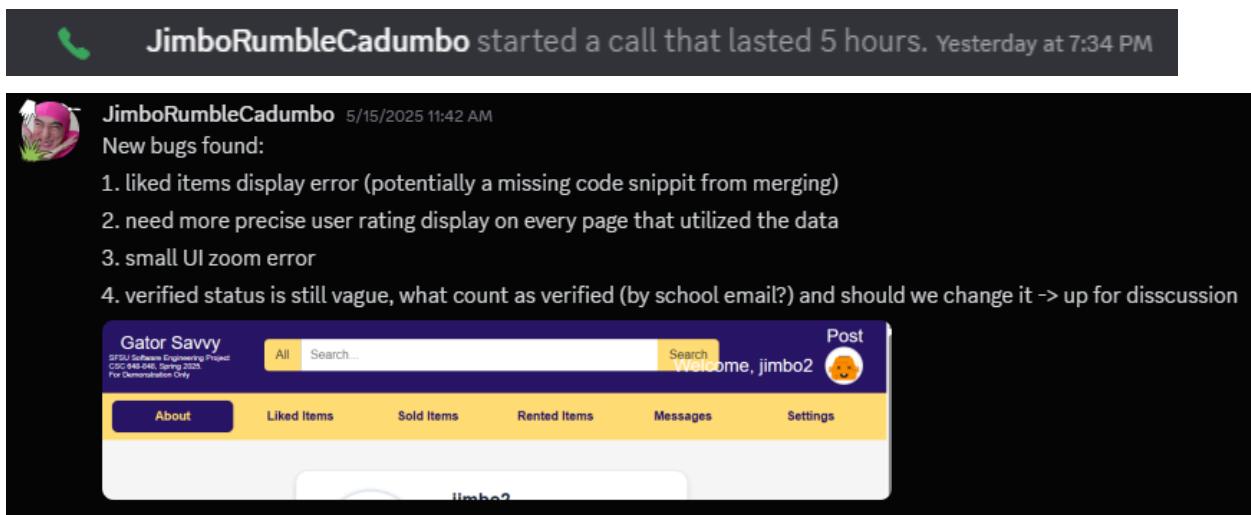


7) Project management

We used Trello to keep track of the status of our tasks. Trello To-Do List:



We used Discord for messaging, communication, and calls for working sessions. Example screenshot:



 diminishedmarkii 5/11/2025 11:24 PM
@everyone
 Important Changes to Our GitHub Process

Just to confirm we're all on the same page, FYI we recently updated our GitHub workflow to streamline our development process and keep things organized. From now on, here's how we'll be using our branches:

- `main`: This will be our *production* branch, and this is the branch that will be deployed on our AWS EC2 server for demo day.  **DO NOT push directly to `main`**. So far, only `@JimboRumbleCadumbo` will be merging from `dev` into `main`. The `main` branch should *always* be stable and working—this is crucial since it will be used for the live demo.
- `dev` [NEW]: This is where all of our active development will take place. Feel free to push directly to `dev` as we work on features and fixes. This will be the primary working branch for day-to-day tasks.
- `frontend`: This branch is for frontend-specific work. If you're working on the UI, components, or anything frontend-related, push your changes to this branch first. Once everything is working and tested, merge `frontend` into `dev`.
- `backend`: This branch is for backend-specific work. If you're working on APIs, database changes, or server-related tasks, push your changes here. Once everything is ready, merge `backend` into `dev`.

(edited)

 3  1 

 Riot 4/11/2025 7:27 PM

- Messaging - Alexis
- Item Page - Alexis
- Dashboard/Home - Athan
- Search Results - technically done - Athan
- Posting - Jim
- Login - Jun
- Signup - Jun

Mention specific SFSU Requirements i.e(SFSU Email) (edited)

8) Team member self-assessment and contributions

Alexis -> Frontend Lead

 Alexis M Perez
To: @Yu-Ming Chen; @Athan Cheung; @David B Cabanella; @David Junwon Chang

Hello Team 5,

1. Contributions to Team Project and Teamwork (Role: Frontend Lead)

- o Established and maintained the foundation for the frontend (Single page render using Vue).
- o Implemented most of the the frontend and backend for the User Profile page.
- o Helped maintain and apply changes to the DB.
- o Assisted team members when completing M3 Frontend pages.
- o Implemented sold and liked items functionality, logged-in to purchase restriction, price filter for search, featured items on front page, mobile responsiveness, and item status.
- o Assisted with any corrections, debugging and clean up.

2. Number of Submission made to GitHub team Dev Branch (73)

```
git shortlog -n --since=2023-05-19 --until=2023-05-20 | sort -nr | head -n 100
```

Author	Count
alexis perez	95
JinboRumblecadumbo	35
JunNotJune	25
davidcabanella	14
Athan Cheung	10
Alexis	9
David Cabanella	2
Ubuntu	2
github-classroom[bot]	1

3. Challenges

I think my main challenge for this project was when I shifted to also helping with backend implementation. It was difficult as I had never used Flask before and I wasn't too familiar with Python. Vue was also a brand new framework that I had never worked with before either but at the beginning of the semester I focused a lot of my time familiarizing myself with it.

4. Overall Experience with GenAI

I'd say my experience with GenAI was fairly good. It was really helpful with helping me learn the vue framework as I had a short amount of time to learn it. It was also a really good assistant for breaking down some of the Python code so that I could understand it better as I have never worked with Python or Flask before.

5. What I would do Better next time

Start much earlier on my tasks and try to avoid and possible delays. This is my first large scale team project which brought on a handful of challenges such as time. I think communication could also be a challenge at times since not everyone is always going to be available. In that case I would want to try and push out more updates to my team more often so that I can keep them up to speed with what I have worked on.

6. Anything else important

I don't really have anything else to say other than this project was a great learning experience.

...

Athan -> Frontend Tech

 Athan Cheung
To: @Anthony John Souza
Cc: @Alexis M Perez; @David B Cabanella; @David Junwon Chang; @Yu-Ming Chen

Hello team,

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

- Designed and developed the prototypes/wireframe on Figma with the frontend lead
- Implemented Doxygen
- Made Dashboard/Home page and Search Results
- Helped with making/redesigning the User Profile
- Made the heart items symbol and some other features in the individual items page
- Did a bunch of CSS
- Worked a little bit on the nav bar and search bar as well

b) Number of submissions they made to GitHub team Dev. Branch

```
git rev-list --count HEAD
```

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

One of the main challenges I encountered during the project was probably picking up new tools quickly, like learning vue and Doxygen and some implementations. Time management was also a bit hard to manage.

d) One brief paragraph of their own overall experience with GenAI

GenAI definitely helped for me, helping me pickup vue and Doxygen very quickly and debugging some js, and helping brainstorming for HTML and CSS.

e) One brief paragraph on what would do better next time

Something I can do better next time is setting and committing to stricter deadlines.

f) Anything else you deem important for instructors to know

I think this was a good introduction to real-world coding projects I think

David -> Github Master

David B Cabanela
To: Alexis M Perez; Yu-Ming Chen; Athan Cheung; David Junwon Chang
Mon 5/19/2025 11:39 PM

Hello Team 5,

a) Contributions to Team Project and Teamwork (Role: GitHub Master)

- Implemented login/authentication and user settings page (Flask and Vue.js)
- Implemented user icon upload image feature (BLOB encoding/decoding and DB storage)
- Developed item listing dynamic rendering from the DB
- Built Navbar component with Vue.js
- Set up team GitHub repository and branches (frontend, backend, dev, main)
- Established GitHub workflow and communicated to team (i.e. dev for testing, main for production/demo, restrict only team lead or GitHub master to merge from dev into main)
- Set up/designed initial database schema and populated tables with sample data (SQL scripts, MySQL Workbench EERD tool); started with AWS RDS, but moved over to local DB due to AWS free tier pricing issues
- Assisted team members with GitHub related questions and merge issues
- Contributed and proofread Milestone documents, gave feedback and edits
- Participated in team demo day (showed Gator Savvy website navigation)
- Attended and was engaged in all team meetings (in-class, in-person outside of class, Discord calls)
- Performed code review via email and practiced with GitHub pull request workflow

b) Number of submissions made to GitHub team Dev. Branch: 27

```
(myenv) dcyberking_kronos@DESKTOP-TDJ6CE7:~/csc648-fa25-0104-team05$ git shortlog -s -n
 95 JimboRumbleCadumbo
 73 alexis perez
 35 JunN0tJune
 25 davidcabanela
 14 Athan Cheung
 10 athanc2
   2 Alexis
   2 David Cabanela
   2 Ubuntu
    1 github-classroom[bot]
(myenv) dcyberking_kronos@DESKTOP-TDJ6CE7:~/csc648-fa25-0104-team05$
```

c) Main Challenges

Working across the full tech stack (i.e. frontend design, backend functionality, DB integration, cloud computing) was challenging. This was my first time integrating all of these technologies at once especially in a team setting, so I had to do a lot of research on tech I was not familiar with like AWS and discuss regularly with teammates. Another challenge was organizing GitHub usage within the team. We had to often remind each other to run `git pull` first to ensure we didn't overwrite anyone's changes. Also, there was initial confusion with coordinating the branches, so I researched and established a GitHub workflow to push to a dev branch and restrict access to the main branch for production. This made our GitHub process smoother and more robust.

d) Overall Experience with GenAI

GenAI was very helpful overall in the project for coding and documentation. It helped recommend GitHub workflow process improvement for our team. It helped me debug image rendering and how base64 encoding works for storing BLOB images in the DB. Also, while I was new to Vue.js, it guided me through piecing together frontend components to dynamically render data. I tried using it for code review, but it wasn't that helpful since it could only comment on snippets of code.

e) What I'd Do Better Next Time

Next time, I would have more frequent meetings with my teammates to communicate questions earlier in the process. I would also break up tasks into smaller, more manageable pieces. Lastly, I would integrate QA testing earlier and more regularly in the project so that we could iterate on improvements to further enhance our website.

f) Anything else you deem important for instructors to know

I contributed heavily to the database setup in terms of designing the schema (i.e. tables, columns, relationships, data entity naming convention), creating the tables, and populating sample data both manually and through SQL script. I also spent a lot of time working on the Milestone documents, including writing, proofreading, and ensuring they met the provided requirements. These were essential to the project but may not be fully reflected in the GitHub commit count. Lastly, I'd like to say that it was really awesome working with my teammates, who were respectful and fun to work with. This was also a great class experience overall. Thank you, Professor Souza for your support, and thank you to my team.

Best,
David Cabanela

Jun -> Backend Lead

[Team 5] -DAVID (JUN) CHANG- Project Feedback

David Junwon Chang
To: Anthony John Souza
Cc: Yu-Ming Chen; Athan Cheung; Alexis M Perez; David B Cabanela

Mon 5/19/2025 1:54 AM

Hello team,

a) Contributions to Team Project and Teamwork (Role: Backend Lead)

- Implemented the login and sign-up functionality for our Gator Savvy demo site using Flask and Vue.js
- Integrated frontend and backend components for account creation and user authentication
- Debugged and resolved backend errors, including MySQL integrity issues and form validation
- Assisted team members with backend routes and session handling using Flask
- Participated in group meetings and consistently communicated progress and blockers
- Contributed to testing user flows and improving server-side error handling

b) GitHub Submissions

- 259 commits made to the GitHub team repository
 - The commit count reflects regular work on feature implementation, bug fixing, and configuration updates

c) Main Challenges Encountered

The main challenges I faced were time management and overcoming my limited technical experience. Learning how to implement secure authentication and understand Flask-MySQL integration took significant time and required frequent troubleshooting. Balancing coursework and technical ramp-up was difficult but ultimately rewarding.

d) Experience with GenAI

GenAI was incredibly helpful throughout the project. I used it to debug backend errors, understand stack traces, and generate example code snippets for login/signup logic and routing. It saved me hours of searching and allowed me to focus on implementation. As someone without a strong technical background, GenAI made backend development more accessible.

e) What I Would Do Better Next Time

Next time, I would focus on managing my time more efficiently and allocating time earlier for trial and error. I also learned the importance of breaking tasks into smaller units, aligning better with software engineering management principles like agile planning and issue tracking.

f) Additional Notes

N/A

Best regards,
David (Jun) Chang

```
junnnotjune@LAPTOP-CDSCC4AH:~$ cd myproject/marketplace/
junnnotjune@LAPTOP-CDSCC4AH:~/myproject/marketplace$ git rev-list --count HEAD
259
junnnotjune@LAPTOP-CDSCC4AH:~/myproject/marketplace$
```

Yu-Ming Chen (Jim) -> Team Lead

[Team 5] Yu-Ming Chen Project Feedback

To: Alexis M Perez; Athan Cheung; David B Cabanela; David Junwon Chang
Cc: Yu-Ming Chen

Hello team 5,

1. Contributions to Team Project and Teamwork (role: Team Lead)
 - Implemented in-app messaging, search items, and item posting functionality
 - Ensuring code/functionality quality via user feedback
 - DB debugging and maintenance
 - Site security & protection check via executing breach tests.

Team Lead contributions:

- Organize weekly meetings & checkpoints w/ progressive tasks
- Enable team engagement

2. Number of submissions they made to GitHub team Dev. Branch:

```
ming9298@LAPTOP-481SSHIL:~/csc_648/marketplace$ git status
On branch dev
Your branch is up to date with 'origin/dev'.

nothing to commit, working tree clean
ming9298@LAPTOP-481SSHIL:~/csc_648/marketplace$ git shortlog -sn
 95 JimboRumbleCadumbo
 73 alexis perez
 35 JunN0tJune
 25 davidcabanela
 14 Athan Cheung
 10 athanc2
   2 Alexis
   2 David Cabanela
   2 Ubuntu
    1 github-classroom[bot]
ming9298@LAPTOP-481SSHIL:~/csc_648/marketplace$
```

3. Overall feedback from the team lead:

Main Challenges:

The main challenge during our process of development is the time & task management.
It is hard for time insufficient team members to devote to the project fully, where task management shows its importance.
There are more functionalities that can be done better with more future planning.

Overall experience with GenAI:

GenAI has done a huge impact to my overall experience of functionality testing and debugging.
Despite the lack of experience in front-end framework knowledge, GenAI made it easier for me to understand and integrate my code with currently existing front-end code.

What would do better next time:

- More detailed task description for team members and tighter checkpoint to reach milestones on time
- Give jobs that matches the skill level of team members
- Better communication during meetings and focus issues

Anything else you deem important for instructors to know

Great class great experience.

Best,
Jim