SW Engineering CSC648/848 Spring 2024
Milestone 2 Part I

Gator Garage

Team 6

Name	Role
Utku Tarhan (mtahan@mail.sfsu.edu)	Team Lead
Akram Alraeeini	Github Master
Jacob Gerales	Backend Lead
Mohammed Mohamed	Frontend Lead
Eliza Ouyang	Support Engineer
Cesar A. Herrera	Support Engineer

History Table

Date Submitted	Date Revised
2024-03-23	

Table of Contents

Executive Summary

The Gator Garage Application revolutionizes the trading of goods and services within the San Francisco State University (SFSU) community which includes students, staff, faculty, and alumni through an intuitive online platform. Our mission is to provide a seamless and secure environment for transactions while offering personalized features tailored to the diverse needs of SFSU students, faculty, and staff.

In addition to general marketplace features, the Gator Garage Application is finely tuned to the unique characteristics of the SFSU community. From facilitating safe pickup zones around campus to offering additional services like tutoring, our platform caters to the specific requirements of our users, ensuring a superior and customized experience.

Led by a passionate team of SFSU students, the Gator Garage Project is driven by a commitment to enhancing campus life through innovation and inclusivity. With a deep understanding of campus dynamics and a dedication to user-centric design, our team is poised to deliver tangible benefits that enhance safety, convenience, and accessibility for all members of the SFSU community.

List of Main Data Items and Entities

1. User Types

Unregistered User: A type of user that is not signed into an account using their SFSU email. Can browse and search approved posts.

Registered User: A type of user who registered with an affiliated SFSU email account. Can buy, sell, and message other users.

Admin: A user who has access to software tools such as mySQL. Shall review and approve every post before they get uploaded.

2. Entities

○

Entity	Definition	Attributes
users	A registered person who buys and sells goods on campus	• userID • username • password • firstName • lastName • email • bio • profilePicture
itemPost	An item offered for sale by a registered user	• postId • itemName • itemPrice • itemPicture • itemThumbnail • itemDescription

		• userId • location • course • postDate • categoryId • isSold
itemRequest	An item requested by a registered user	• requestId • itemName • desiredPrice • itemPicture • itemThumbnail • itemDescription • userId • course • postDate • categoryId • isBought
categories	Names of different category types for items being sold	• categoryId • name
favorites	Posts which were favorited by registered users	• favoriteId • userId • postId

messages	Messages between registered users in th context of specific posts	• messageId • content • recipient • sender • postId • requestId • sendTime

High-Level Functional Requirements - Prioritized

Priority 1:

Unregistered User:

1. Shall be able to search and browse content without an account.
2. Shall be able to search for items and view basic information about listings.
3. Shall be able to create an account using a valid SFSU email address and password.
4. Shall be able to login with a valid account.
5. Shall be prompted to create an account or login when trying to initiate contact with the seller (sending a message).
6. Shall be prompted to create an account or login when trying to view the dashboard.

Registered User:

7. Shall inherit all the requirements of an unregistered user.
8. Shall be able to post listings of items for sale, including item details, pricing, and images.
9. Shall be able to communicate with sellers through sending a message.
10. Shall be able to remove listings from their account.
11. Shall be able to log out.

Admin:

12. Shall inherit all the functional requirements of a registered user.
13. Shall be required to review all posts before going live.
14. Shall be able to unpublish a post.

Priority 2:

Unregistered User:

Registered User:

15. Shall be able to save favorite listings for later viewing.
16. Shall receive notifications for new messages, comments, or activity on their listings.
17. Shall have the ability to leave feedback and ratings for transactions with other users.

Admin:

Priority 3:

Unregistered User:

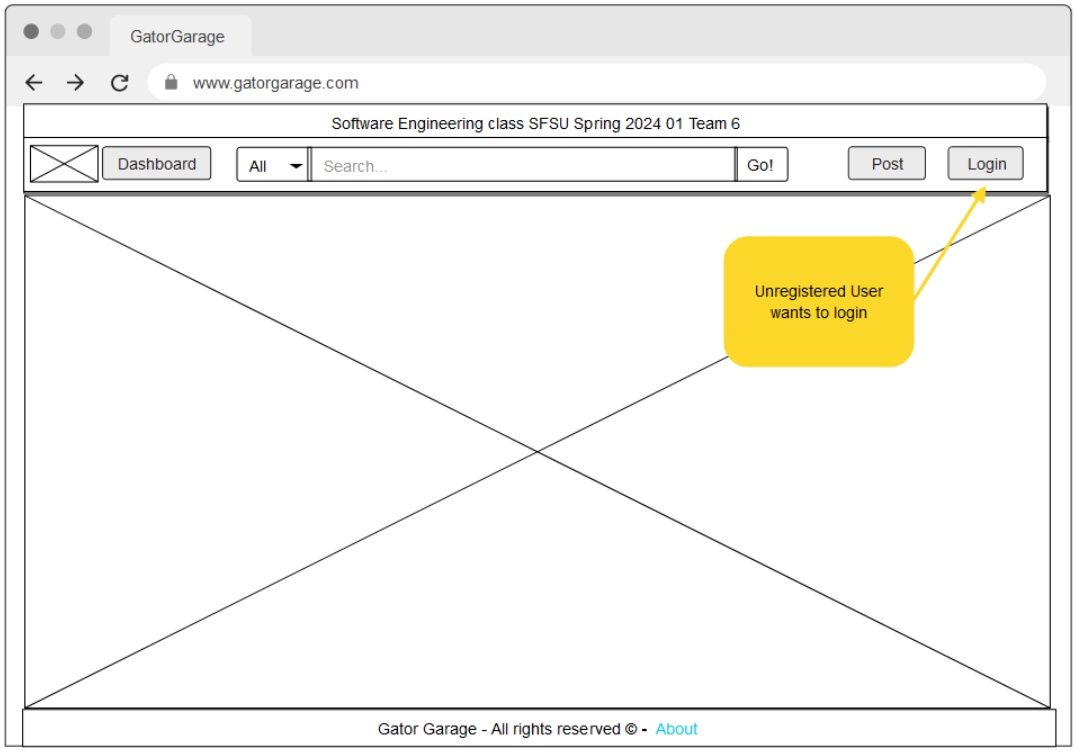
Registered User:

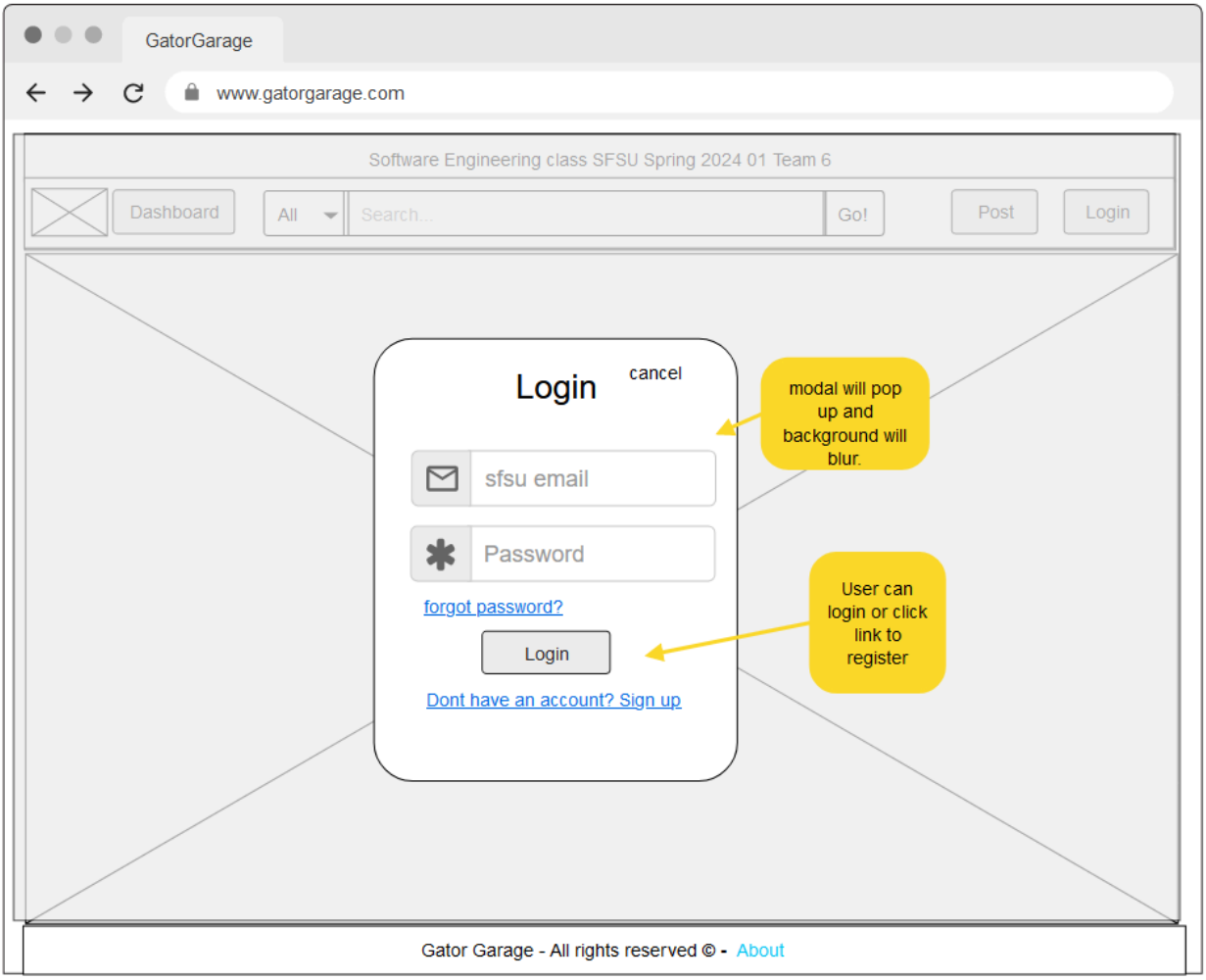
18. Shall be able to provide additional profile information, including a profile picture and a brief description.
19. Shall be able to rate other users.
20. Shall have access to additional features such as advanced search filters or sorting options.
21. User shall have access to educational resources or guides on safe trading practices

Admin:

Use Case Storyboards

Login/Signup

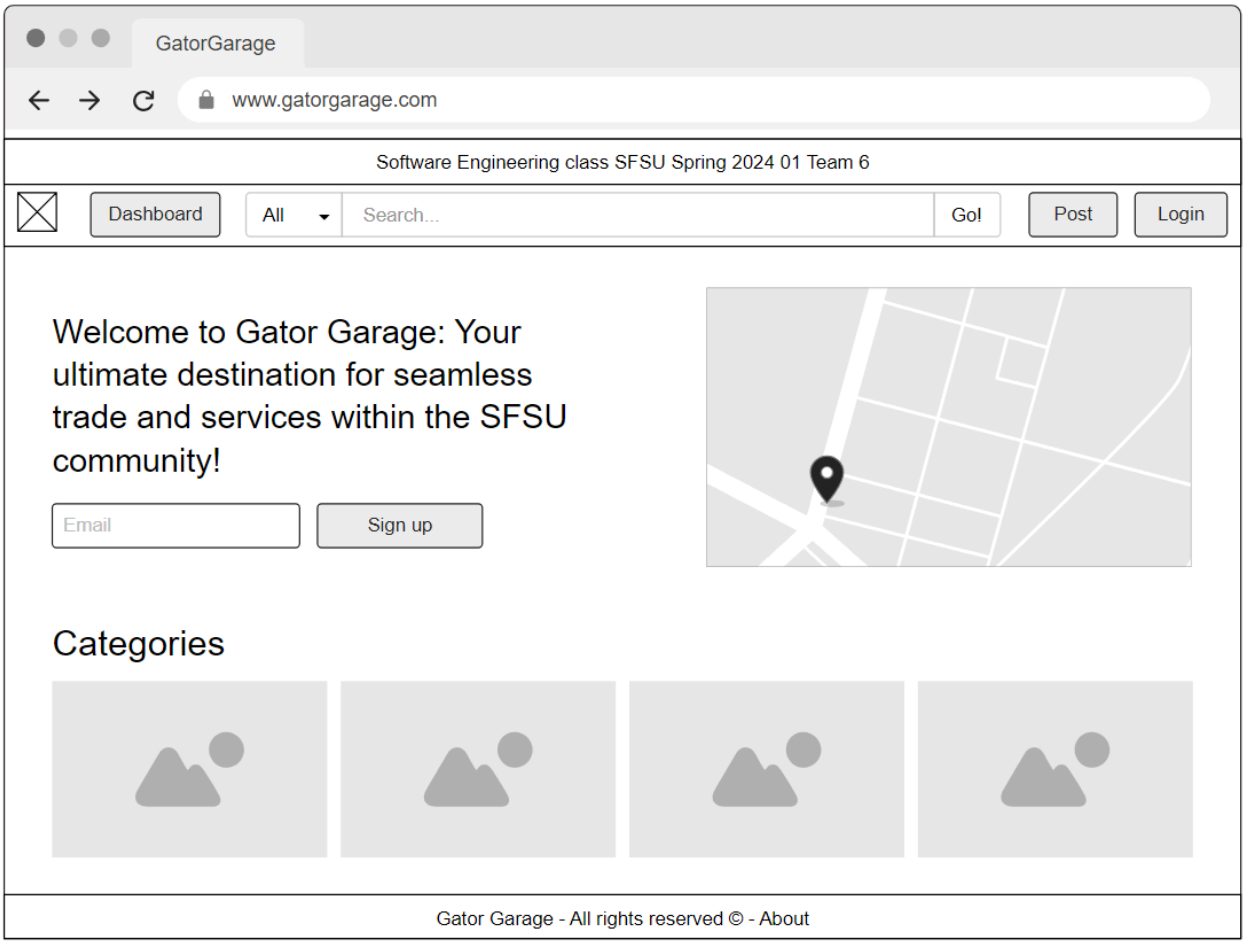



GatorGarage
www.gatorgarage.com
Software Engineering class SFSU Spring 2024 01 Team 6
Dashboard
All
Search...
Go!
Post
Login
Login
cancel
sfsu email
Password
forgot password?
Login
Dont have an account? Sign up
modal will pop up and background will blur.
User can login or click link to register
Gator Garage - All rights reserved © - About

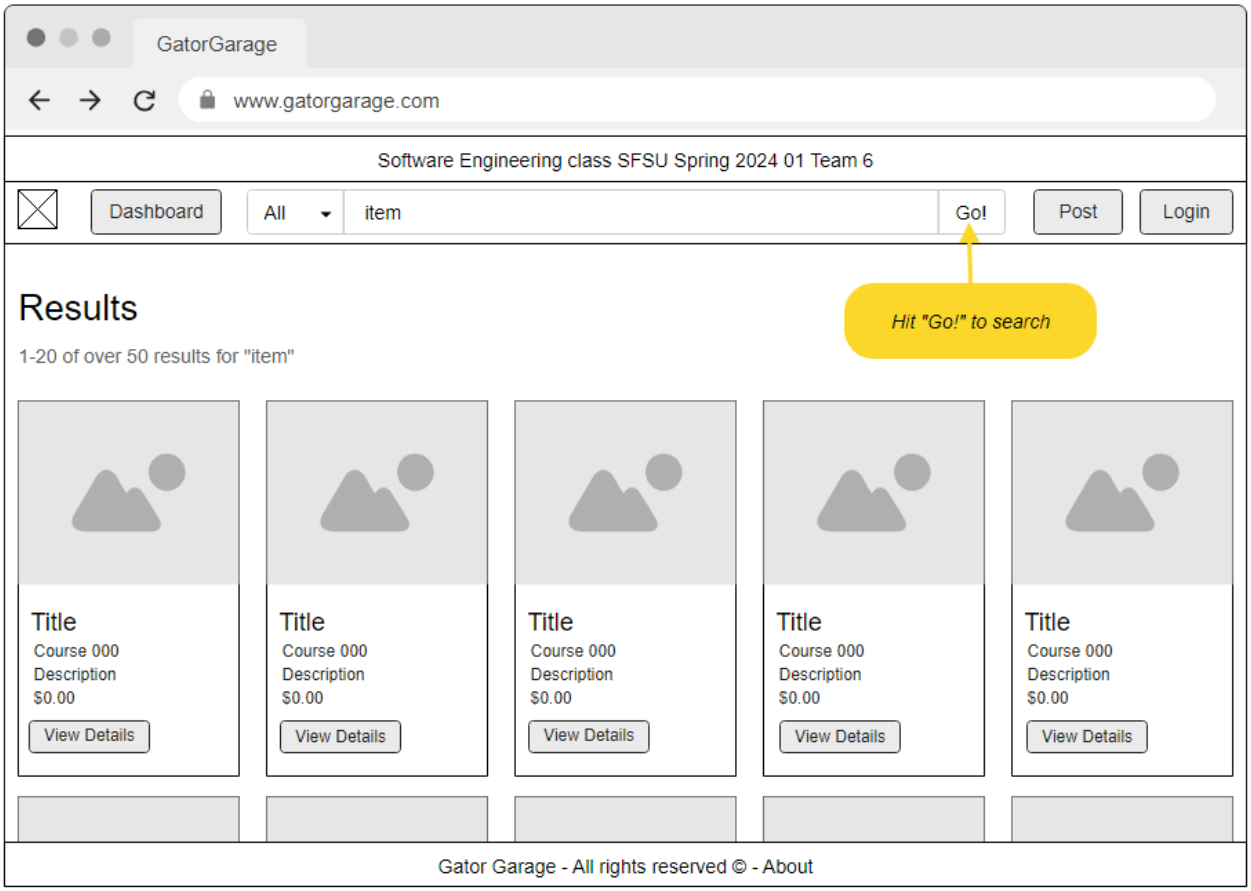
1. User will click login and the login modal will popup and blur the background.
2. If a user doesn't have an account they can click a link to direct to sign up modal.
3. Front-end validation will check if a user uses sfsu.edu email and matching passwords.
4. Users can exit at any modal by clicking cancel.
5. Once a user signs in or logs in they will have access to registered user features.
6. Login button will become Logout

Unregistered User - Browsing, Messaging, Registering

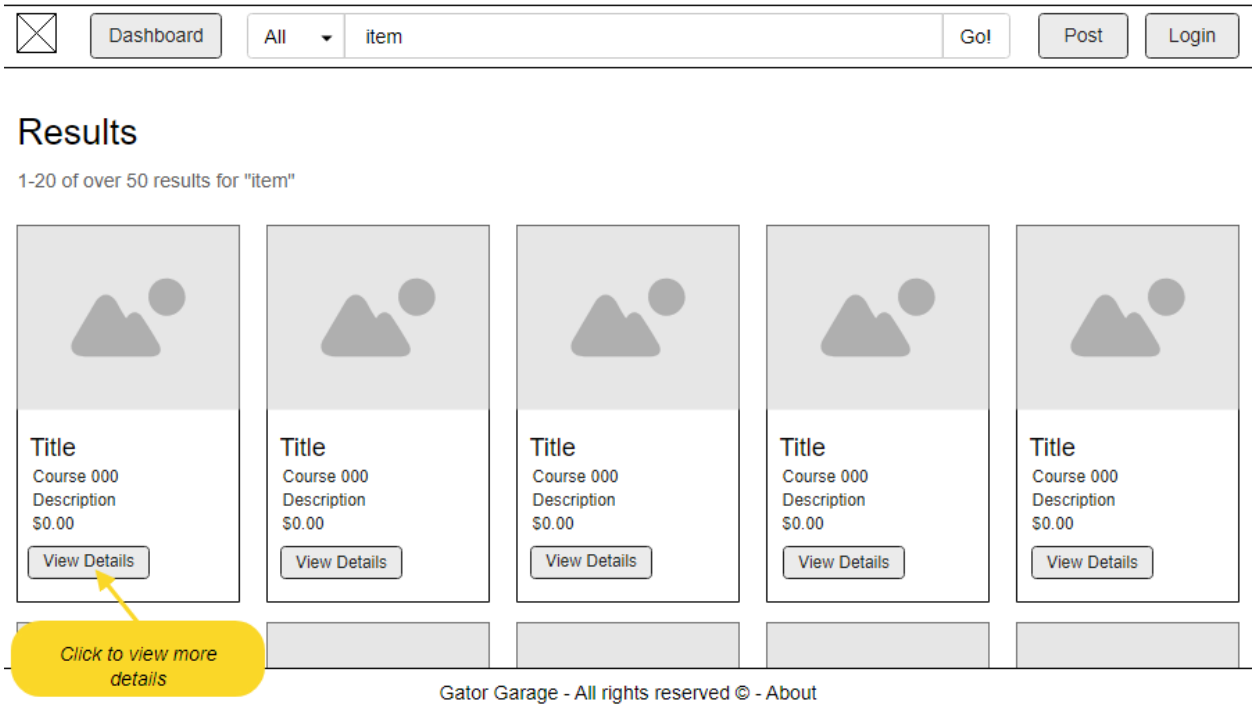
Jane wants to browse the site and use the search function to look for specific items. She begins at the homepage of GatorGarage.

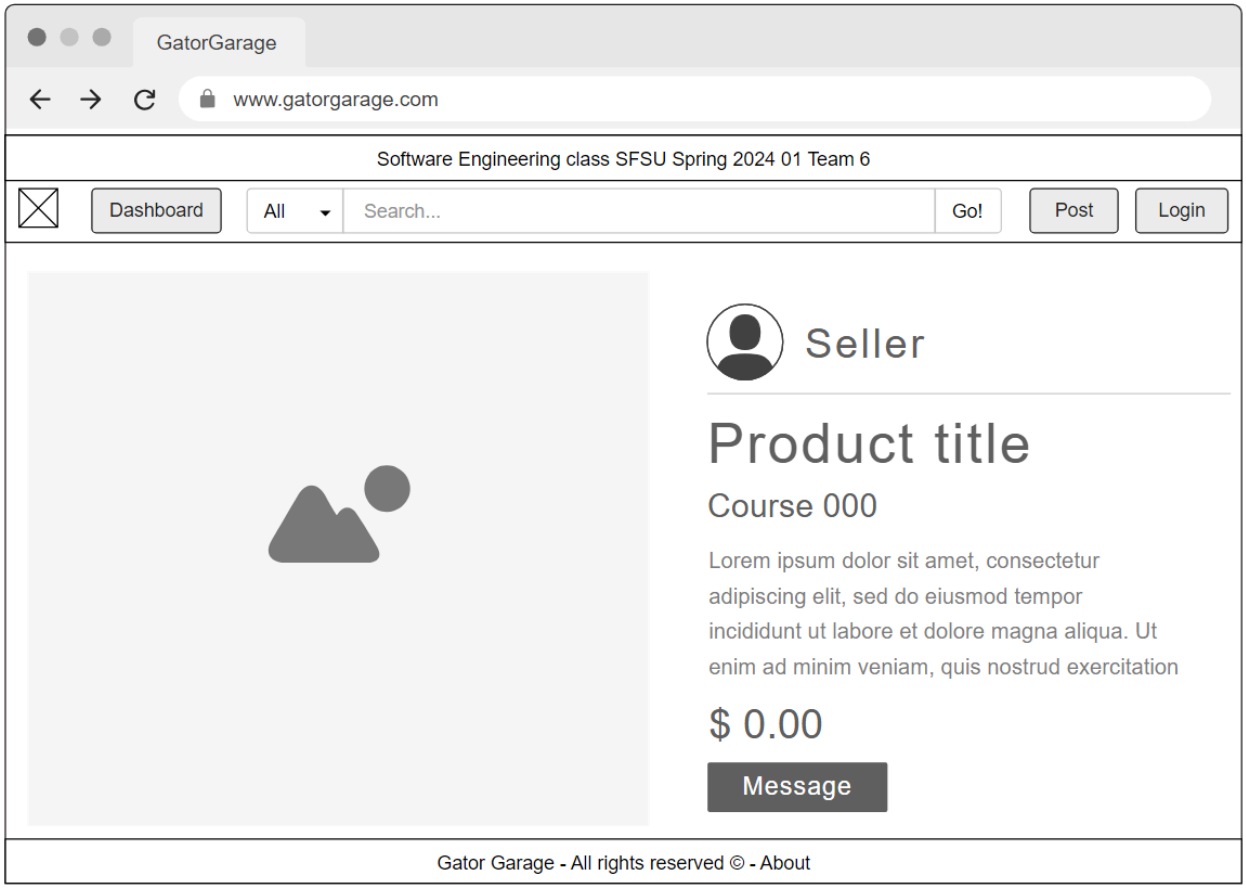


Jane enters items she wants to browse in the search box and hits “Go!,” which then displays all results for that specific item.



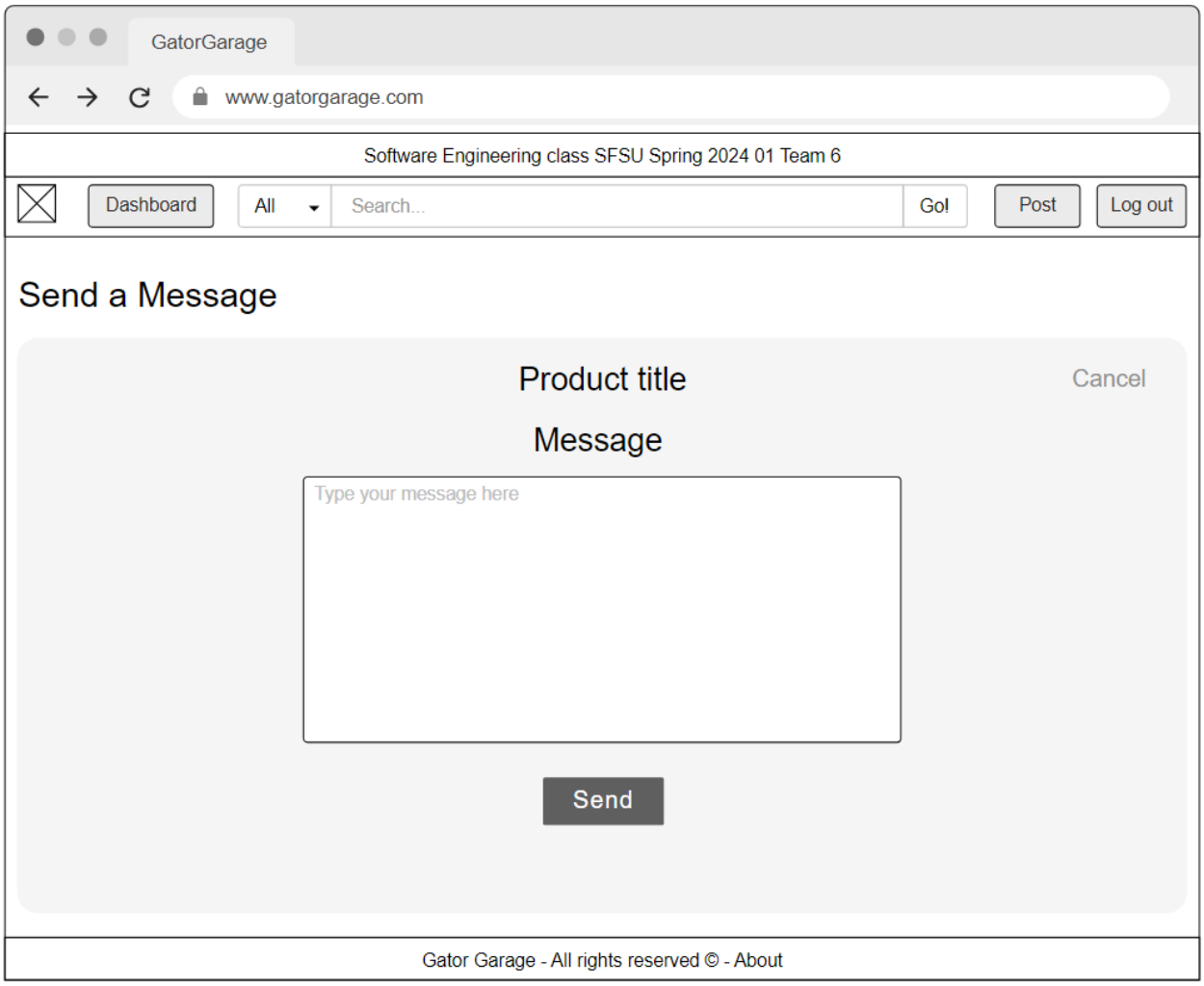
Jane then clicks on “View Details” on an item’s post to view more of its details.





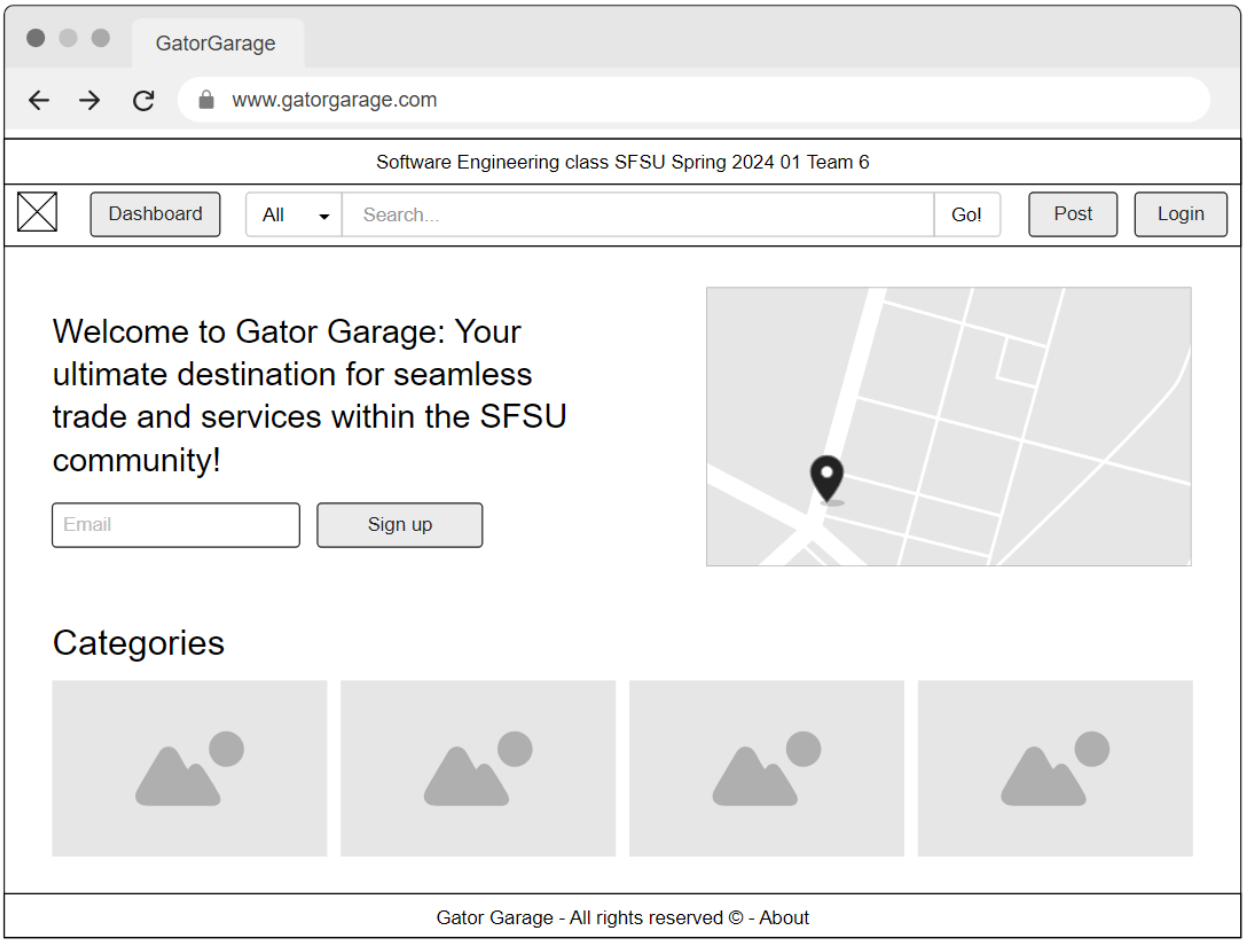
Jane wants to send a message to the seller, but after hitting the “Message” button she is asked to login or register. She registers by filling out her information such as her SFSU email, password, and name into the sign up modal.

Incorporating lazy registration into our website: after filling out the registration form, Jane can write and send her message to the seller.

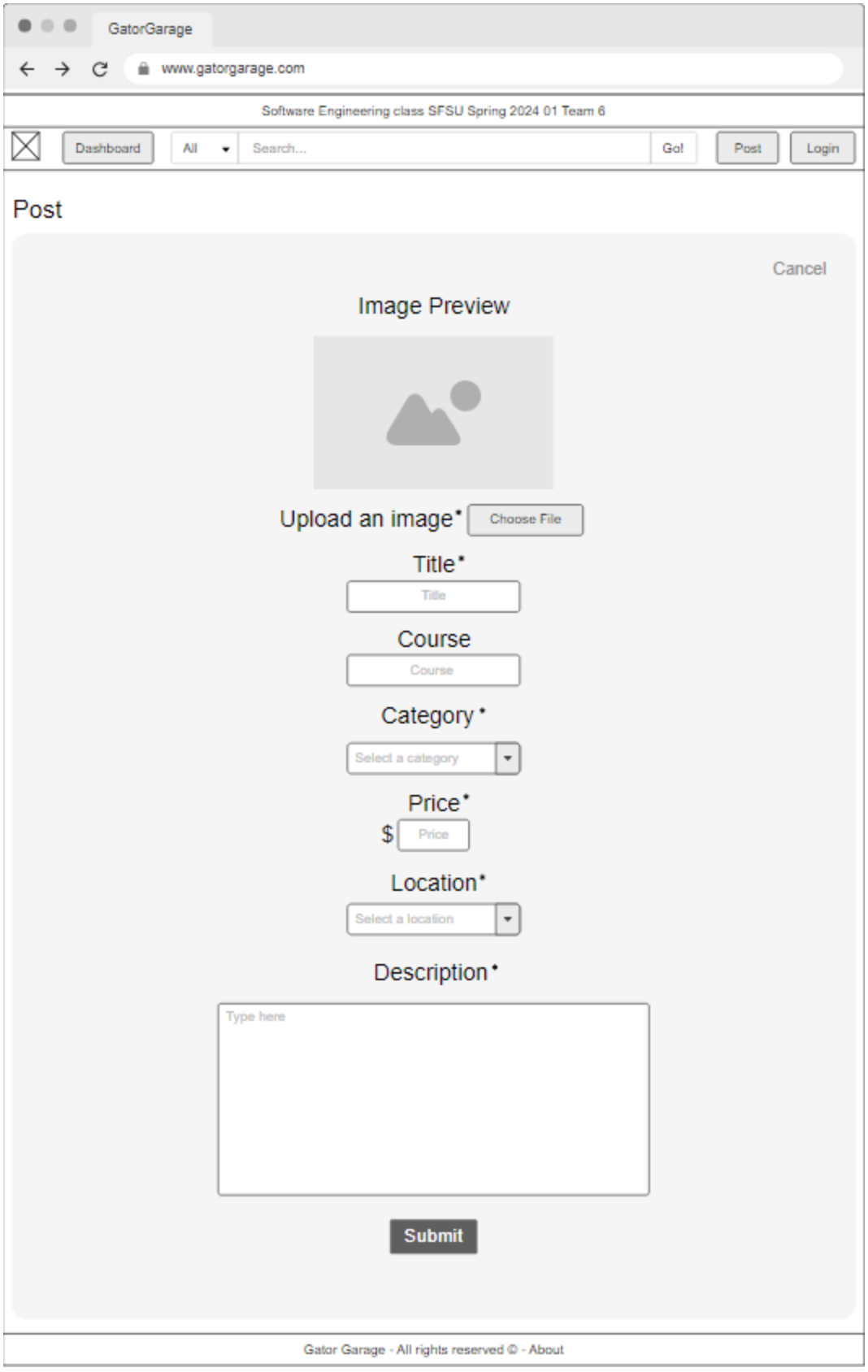


Unregistered User - Posting content to sell

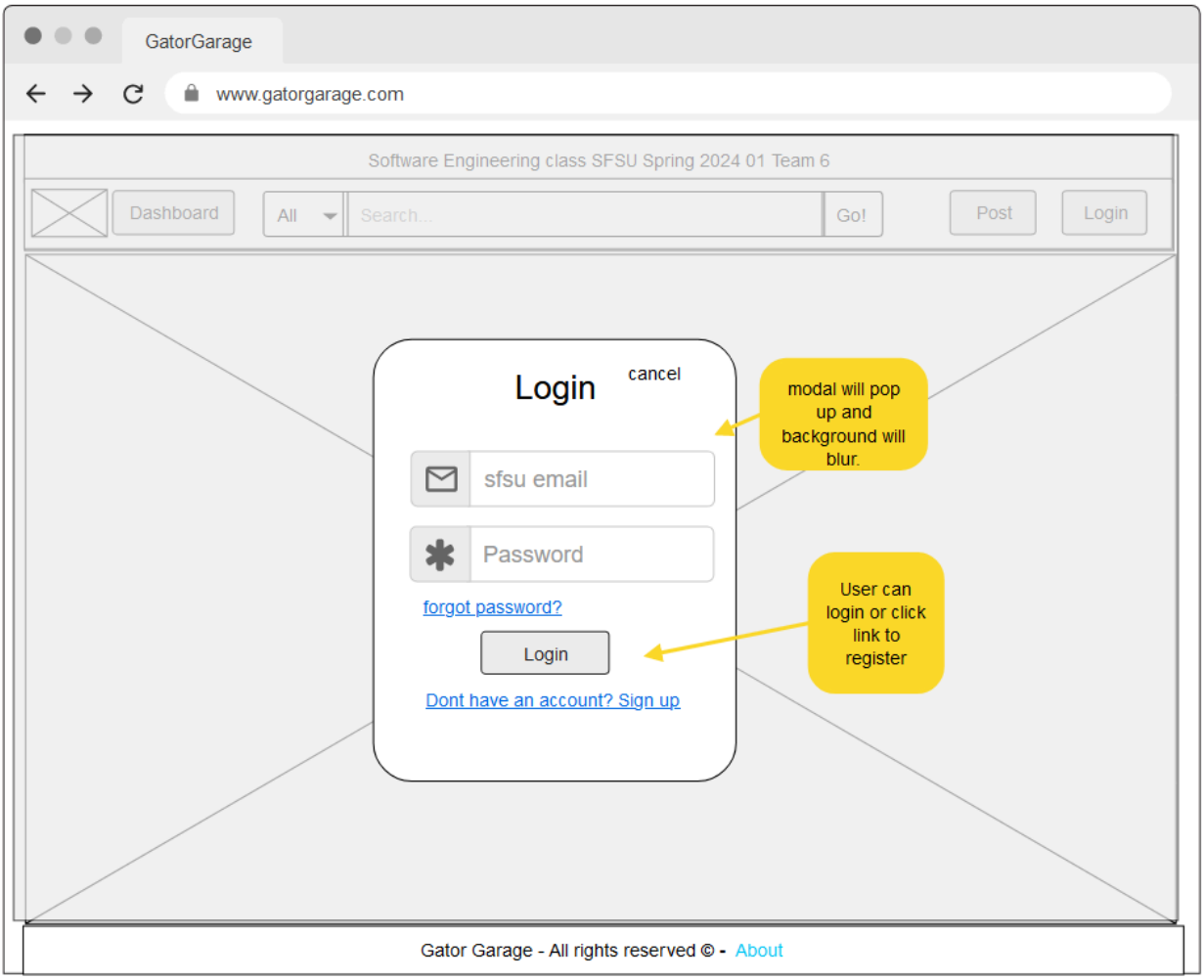
Jim has just created a new piece of artwork and wants to sell it using the website. He begins at the homepage of GatorGarage.



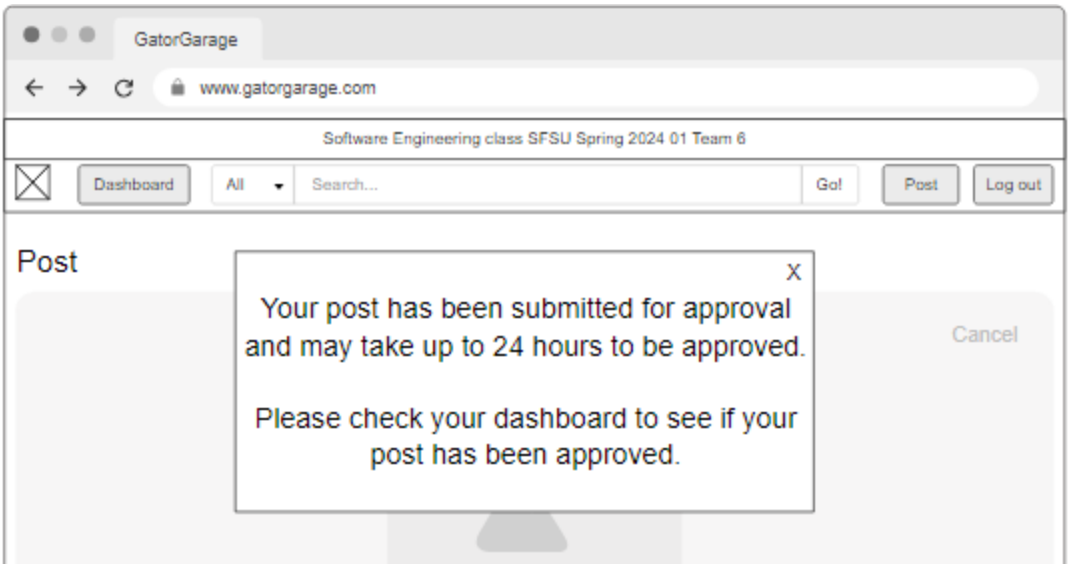
Jim wants to make a post and navigates to the post page by clicking on “Post,” located on the top right of the nav bar next to login. He will upload the image of the item and give it a name, price, description, category, and select an SFSU secure-pick-up location.


GatorGarage
www.gatorgarage.com
Software Engineering class SFSU Spring 2024 01 Team 6
Dashboard
All
Search...
Go!
Post
Login
Post
Cancel
Image Preview
Upload an image*
Choose File
Title*
Title
Course
Course
Category *
Select a category
Price*
$
Price
Location*
Select a location
Description*
Type here
Submit
Gator Garage - All rights reserved © - About

When Jim attempts to submit his post, he is prompted to login or register as the login modal pops up.



After entering his login credentials and logging into the site, he receives a message that his post has been submitted for approval.

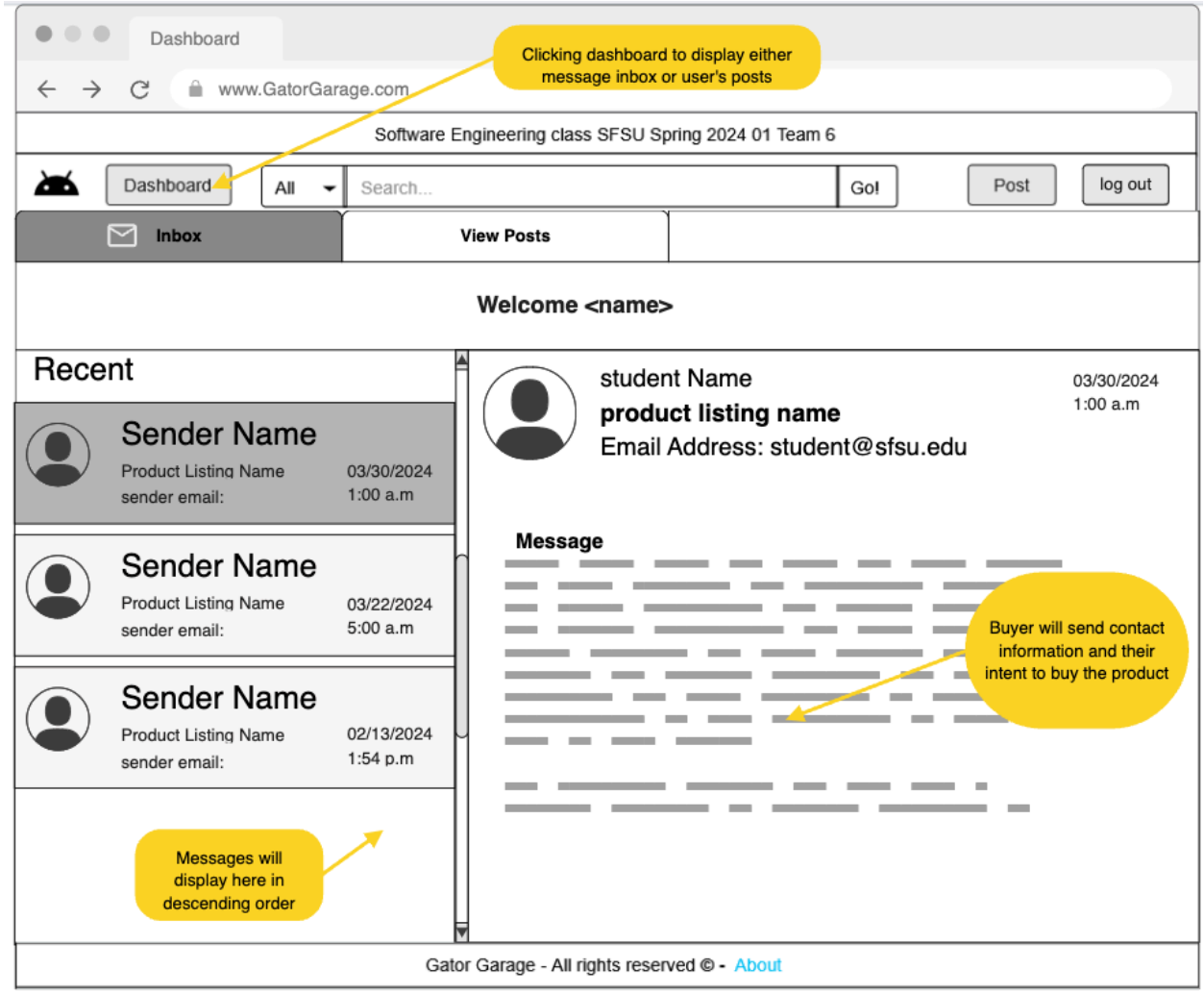


Dashboard

Jim wants to view all of his posts currently listed, so he navigates to the dashboard by clicking on “Dashboard,” located on the top left of the nav bar. He will be able to view information related to his posts and access his message inbox here.

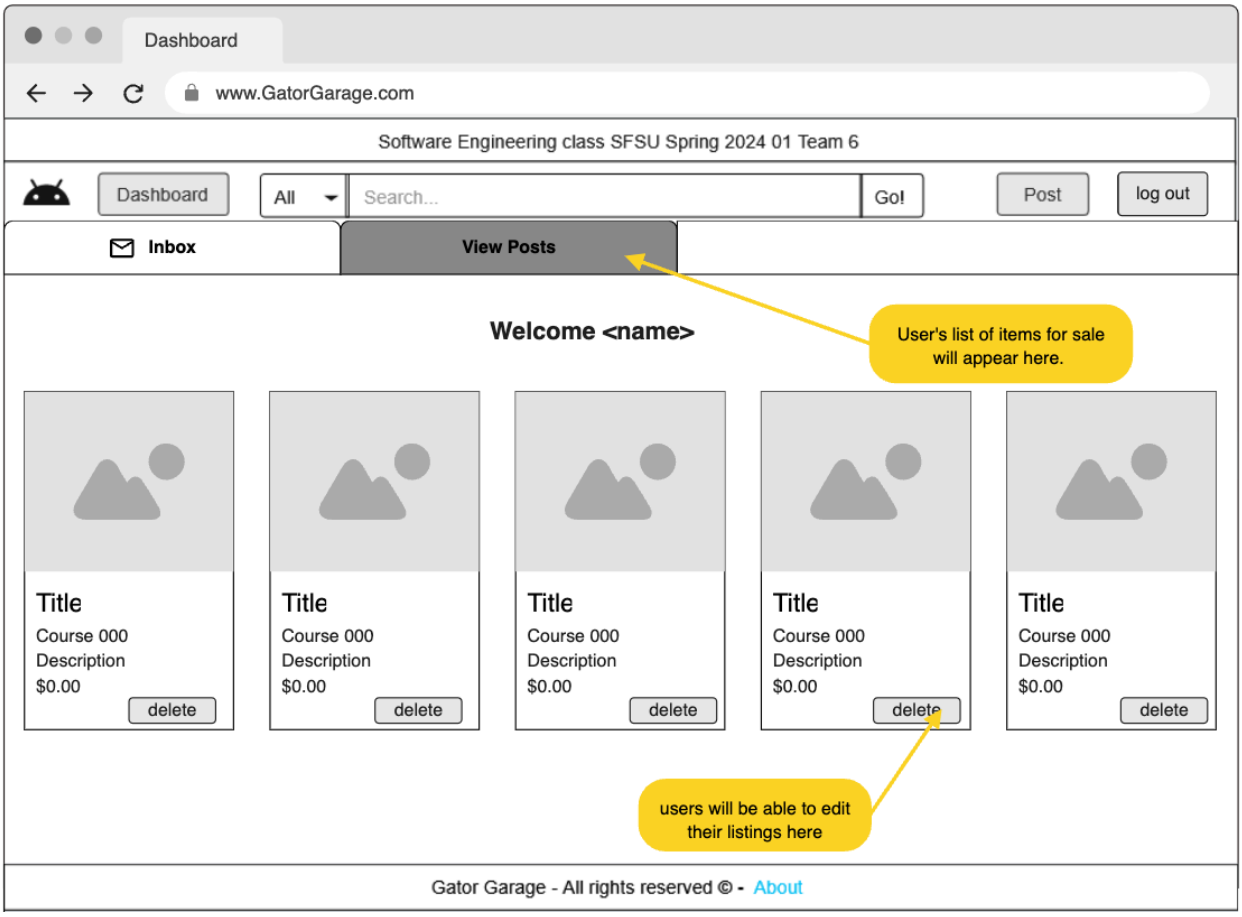
Inbox is where the email users are displayed regarding each product.

- Display info about message details like date.
- It displays the info about each buyer and what item they’re interested in.
- It contains the buyers' contact info in the message body.



View Posts will contain all the items Jim put up for sale.

- Users will be able to view their listings and delete the items they want to get rid of.



Admin UI

No mockup included for admin UI for reviewing a pending item post because this functionality is handled in MySQL workbench.

High-Level Architecture and Database Organizational Summary

Tables:

user:

- userId (PRIMARY KEY, UNIQUE, INT): Unique identifier for a user
- username(VARCHAR(255)): Display name chosen by user
- password(VARCHAR(255)): Encrypted password for user
- firstName(VARCHAR(255)): First name of user
- lastName(VARCHAR(255)): Last name of user
- email(UNIQUE, VARCHAR(255))l: Email address that must end in “@sfsu.edu”
- bio(OPTIONAL, TEXT): Description of user
- profilePicture(OPTIONAL, VARCHAR(255)): Path to user’s display avatar

itemPost:

- postId (PRIMARY KEY, UNIQUE, INT): Unique identifier for a post
- itemName (VARCHAR(255)): Name of item
- itemPrice (DECIMAL(4,2)): Item price in USD ($)
- itemPicture (VARCHAR(255)): Path to item picture
- itemThumbnail (VARCHAR(255)): Path to item thumbnail
- itemDescription (TEXT): Description of post
- userId (FOREIGN KEY, INT): Author of post
- location (ENUM): Secure pick - up location
- course (OPTIONAL, VARCHAR(255)): Course related to item
- postDate (TIMEDATE): Timestamp of item post
- categoryId (FOREIGN KEY, INT): Category ID of item
- isSold (TINYINT(0)): Flag that indicates if post is sold

itemRequest:

- requestId (PRIMARY KEY, UNIQUE, INT): Unique identifier for a request
- itemName (VARCHAR(255)): Name of item
- desiredPrice(DECIMAL(4,2)): Requested price in USD ($)
- itemPicture(VARCHAR(255)): Path to item picture
- itemThumbnail(VARCHAR(255)): Path to item thumbnail
- itemDescription (VARCHAR(255)): Description of request
- userId (FOREIGN KEY, INT): Author of request
- course (OPTIONAL, VARCHAR(255)): Course related to item
- postDate (DATETIME): Timestamp of item request
- categoryId (FOREIGN KEY, INT): Category ID of item
- isBought (TINYINT(0)): Flag that indicates if post is bought

messages:

- messageId (PRIMARY KEY, UNIQUE, INT): Unique identifier for a message
- content (TEXT): Written message from one user to another
- recipientId (FOREIGN KEY, INT): Recipient of message
- senderId (FOREIGN KEY, INT): Sender of message
- postId (FOREIGN KEY, OPTIONAL, INT): Post ID to provide context
- requestId (FOREIGN KEY, OPTIONAL, INT): Request ID to provide context
- sendTime (DATETIME): Timestamp of sent message

favorites:

- favoriteId (PRIMARY KEY, UNIQUE, INT): Unique identifier for a favorite post
- userId (FOREIGN KEY, INT): User which favorites a post
- postId (FOREIGN KEY, INT): Post which a User favorites

categories:

- categoryId (PRIMARY KEY, UNIQUE, INT): Unique identifier for a category
- categoryName (ENUM): Name of category

Media Storage:

Images will be accessed via file system. The file system was chosen because this approach is simpler and more scalable compared to the BLOB system. To access an image, queries for images shall use relative path names.

Search implementation:

Search will be implemented by concatenating every key word in a search query and using %like SQL statements.

Current Key Risks

Skills:

Skills Gap

Description: We may need additional support on the front or back-end to complete deadlines in a timely manner; however, we might lack adequate experience in the technologies (eg., bootstrap, express).

Plan: Utilize spring break as an opportunity to refresh and strengthen the fundamentals of our technology stack. Also, discuss with the front-end and back-end team leaders to identify the areas where help is needed the most.

Schedule:

Maintaining Pace towards M2 P.2 while Team Member Recovering from Surgery

Description: Our GitHub Master has scheduled surgery over spring break. Without appropriate delegation of duties, this could impact the team's capacity to meet the M2 Part 2 deadline.

Plan: Tentatively shift responsibilities of GitHub Master to Utku Tarhan, ensuring continuity of project execution and progress towards milestone M2 P.2.

Technical:

Insecure Handling of User’s Personal Information

Description: We have not identified a secure method to handle a user's personal information, specifically passwords.

Plan: Research encryption techniques for our technology stack, such as bcrypt from npm, and use AI to assist in areas of difficulty.

Insufficient Expertise in User Image Handling and Storage Processes

Description: We lack a clear understanding of how to implement the process of user image upload, generating the thumbnail, storage of both the original and thumbnail image in our application directory, and then recording of the image paths in our database.

Plan: First, we'll define our file upload requirements, including file naming conventions and storage directories. Then, we'll research Multer, a middleware for Node.js and Express.js, to verify its capabilities for handling the file uploads, storage, and naming based on our requirements. Also, review Sharp module (node) tutorial by Anthony Souza (CTO).

Absence of Structured Approach for Front-End Unit Testing

Description: We have not identified how we are going to approach unit testing for our front-end to mitigate the risks of releasing bug-afflicted features and improve the overall reliability of the application.

Plan: Identify which testing framework to use for javascript (eg., Jest, jsdom) before starting M3.

Teamwork:

Undefined Steps for Handling Team Member Disputes

Description: While we haven't previously faced unconstructive disagreements, the inherent dynamics of teamwork, coupled with tight deadlines and creative differences, hold potential for disputes that could unintentionally jeopardize our launch deadline.

Plan: Formulate a chain of command or action sequence for timely and effective dispute resolution

Project Management

Project management involves having clearly defined tasks, team member responsibilities, and deadlines to efficiently and effectively handle the stages of a project from start to finish. For us, each stage is an assigned milestone, with unique objectives that propel us towards our ultimate goal—launching our SFSU-focused e-commerce website by May 15th.

For efficient coordination, we utilize Jira software as a consolidated platform where everyone can easily view our team's tasks and understand their own personal responsibilities. This approach simplifies the project management process.

In Jira, we've established four stages for tasks: TO DO, IN PROGRESS, DONE, and VERIFIED.

Initially, we place a newly generated task in the TO DO stage. At this stage, we assign the task to a team member and set targeted deadlines. When a team member starts working on a task, it moves to the IN PROGRESS stage, signaling to the team that the task is being actively worked on. Upon completion, the task enters the DONE stage. Here, another team member verifies the task's completion and adds comments for revisions where necessary. If the task is truly completed, it is marked as VERIFIED. If there's still work to be done, it's returned to the TO DO stage, and we repeat the process until all revisions are addressed.

Use of GenAI Tools for M2

Articulation; Rating: Medium

There are times when we find our written words fall short of conveying the original intentions we had conceived in our minds. In such instances, Chat GPT 4.0 has been helpful, offering sentence variations that more accurately capture the essence of what we were initially striving to convey. GenAI also assisted in giving ideas and suggestions for executive summary which was used as a reference when creating the final iteration.

Team-Lead Checklist

- ☑ So far all team members are fully engaged and attending team sessions when required - DONE
- ☑ Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing - DONE
- ☑ Team reviewed suggested resources before drafting Milestone 2 - DONE
- ☑ Team lead checked Milestone 2 document for quality, completeness, formatting and compliance with instructions before the submission - DONE
- ☑ Team lead ensured that all team members read the final Milestone 2 document and agree/understand it before submission - DONE
- ☑ Team shared and discussed experience with genAI tools among themselves - DONE