Spartan Marketplace

Project Team: Kanika Sun, Trinity Boler, Kennedy Page, Nehabahen Chauhan

Table of Contents

1. Project Definition

College students right now don’t have a dedicated platform to share their ideas and communicate with other students on campus easily. Some students may also have small businesses they want to promote, have items that they don't want anymore, or just want to help the community and give back. It is possible to do things like that on sites like Instagram or Snapchat, but that could be unsafe because they could be communicating with people they don’t know, because sites like that are prone to fake accounts. Therefore, students could be communicating with strangers with malicious intent.

So what we did is create a platform called Spartan Marketplace that allows students to do all those things: buy or sell, one-to-one messaging, and talk within their community, all in one platform. Students won’t have to go from site to site to accomplish different things.

To make transactions safer, we made it a platform where only people with an authenticated UNCG email address can enter and access the site. Therefore, if any problems were to occur with the people there, it would be easy to track them down and report the situation, and have it solved quickly because every student would be linked to their account.

---

2. Project Requirements Analysis

2.1. Functional Requirements

#	Functional Requirement	Category	Notes	Priority
1	System should allow users to create an account with their valid UNCG email address.	Data		High
2	System should allow users to view a listing of products that were created by other users on the site.	Content display		High
3	System should allow users to buy, sell, or trade items with other users on the site.	Data		High
4	System should allow users to search for items through the AI	Search		High

	Chat Assistant to find things that suit their best interest.			
5	System should allow users to log in and log out. It should also allow users to modify their profile.	Data		High
6	System should allow users to filter for their specific search and go through different categories.	Search		High
7	System should allow admins to manage user accounts and moderate posts made on the site.	Data		High
8	System should allow users to be able to delete their own listings.	Data		High
9	System should allow for admin accounts to be created.	Data		High
10	System should allow users to be able to flag/report products and chat messages.	Data		Medium
11	System should allow passwords to only be accessible to individual account users.	Data		High
12	System should allow users to receive a message alert.	Content Display, Data		High
13	System should allow a choice of payments.	Data		Medium
14	System should allow messaging between users.	Content Display, Data		High

15	System should allow users to save items to favorites.	Content Display, Data		Low
16	System should allow users to use the AI Chatbot.	Data		High
17	System should allow users to confirm, complete and reject purchases.	Content Display, Data		High
18	System should allow users to visit other users store's.	Content Display, Data		High
19	System should allow users to save their favorite items and posts.	Content Display, Data		Low

2.2. Usability Requirements

The site is easy to navigate, with all tabs being connected to one dropdown, so you can find everything you need. It is the most responsive on the desktop.

The design of the site is consistent throughout all the pages, keeping the theme of UNCG colors. The layout of the site is clean and minimal, so it is not too harsh to look at and is very readable.

Page loading occurs relatively quickly, each page always has a mini loading page if the page is taking a bit of time to load. Messages are sent and received in real time. Our backend is scalable. Since we used Supabase, it can handle a large number of users, including 1,000+, without any major changes. Supabase is built on top of PostgreSQL and offers real-time data handling, so it's designed to scale as usage grows.

2.3. System Requirements

Everyone in the group used different types of laptops to complete the project. Any laptop, desktop or PC with a sufficient amount of space and an internet connection will be able to host our site.

Frontend

- Framework: React.js, HTML
- Styling: CSS/Tailwind

- Browers: Any modern browser

Backend
- Runtime: Node.js
- Framework: Javascript
- Real-time Communication: Supabase
- Authentication: Supabase
- Local Development: VSCode
- Cloud Deployment: Vercel

Database
- Relational DB: PostgreSQL
- Cloud hosted DB: Supabase

2.4. Security Requirements

2.4.1. User Authentication and Authorization

- Secure Login System: UNCG authenticated email addresses only. Password requirements.
- Session Management: Unused sessions give warnings and logs out.
- Authorization checks: Regular users are only allowed to view regular content. Admins have access to admin features.

2.4.2. Input Validation

- Frontend & Backend Validation: Check all forms for empty fields, valid email, appropriate lengths, etc.

---

3. Project Specification

3.1. Focus / Domain / Area

Our focus was to create a student-to-student marketplace web platform designed exclusively for the University of North Carolina at Greensboro (UNCG) community.

Our domain is:
- E-commerce / Marketplace (Student-to-student item exchange)
- Social Networking (Messaging, open community board)
- AI Integration (Chat assistant for navigation/help)

3.2. Libraries / Frameworks / Development Environment

Frontend

- Framework: React.js, HTML
- Styling: CSS/Tailwind
- Browers: Any modern browser

Backend

- Runtime: Node.js
- Framework: Javascript
- Real-time Communication: Supabase
- Authentication: Supabase
- Local Development: VSCode
- Cloud Deployment: Vercel

Database

- Relational DB: PostgreSQL
- Cloud hosted DB: Supabase

AI Assistant:

- Custom logic / integration with OpenAI API

Development Environment:

- Code Editor: VS Code
- Version Control: Git + GitHub
- Local Development: Node.js, npm, React scripts

Platform

We created Spartan Marketplace to be compatible with desktop only, for now. It can be viewed on a mobile phone, but styling becomes messed up.

Genre

Spartan Marketplace is an E-commerce site as well as a Social Media site and a messaging platform.

3.3. Dataset / Data Resources

Real-time, user-generated content:

- Listings
- User profiles

- Messages
- Community Board messages

External Resources:

- Dummy images from Google images.
- University resources

---

4. System – Design Perspective

4.1. Subsystem

- **User Authentication**: Handles login, registration, and authentication.
- **Listing Management**: Handles product listings, editing, deleting, and viewing. It also includes **search and filtering** which enables users to search and filter through listings.
- **Messaging:** Enables communication between buyers and sellers.
- **AI Chat:** Handles user search request and creates new listings

User Interaction (Front-End to Controllers)

- The user interacts with **HTML Views** via a web browser.
- HTTP requests (REST API) are sent to the appropriate **Controllers** based on user actions (e.g., login, product creation, messaging).
- The **Auth Controller** handles authentication and user type determination (User/Admin).
- The selected view is returned as an HTTP response.
- User interacts with Open AI Chat route to create and search listings

Controller Communication (REST API)

- The **Auth Controller** authenticates users and determines their access levels.
- The **User Controller** manages user profile data (viewing and editing).
- The **Product Controller** handles CRUD operations on product listings (Create, Read, Update, Delete).
- The **Message Controller** manages private chats and open board messages.
- The **Security Controller** handles password authentication and email verification.
- These controllers communicate via internal REST API calls to fetch and update data.

Database Communication (CRUD Operations)

- Controllers interact with the **Model** to fetch or update data.

- The **Model** communicates with the **Rational Database** via SQL queries.
- The database processes these queries and returns data, which is then sent back to the controllers.

Search and Filtering

- Users submit search queries through the **Views**.
- The request is handled by the **Product Controller** using filters.
- The filtered results are fetched from the database and returned to the user.
- Open AI route search through the database to return relevant listings

Messaging System

- Users send messages via **private chat** or **open board** features.
- The **Message Controller** processes and stores messages in the database.
- Notifications/alerts are triggered for new messages.

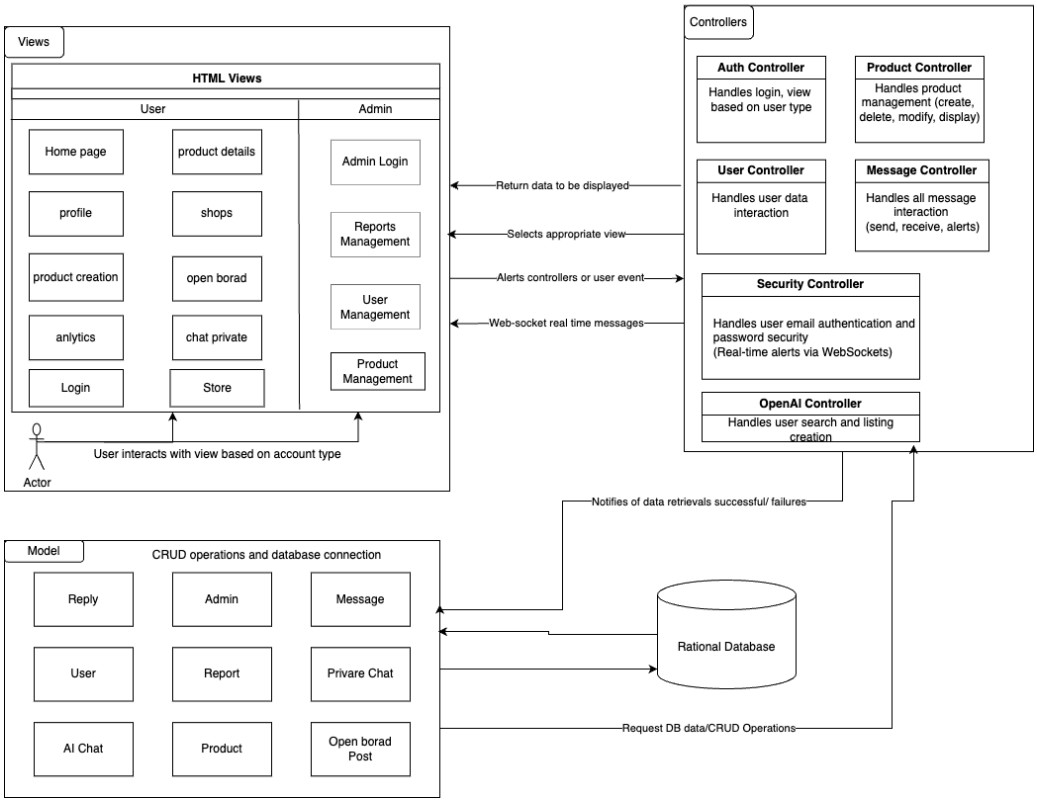
Technology Stack for Communication

- **REST API (HTTP/HTTPS)**: Used for communication between the front-end (HTML Views) and controllers.
- **SQL Queries**: Used by models to interact with the database.
- **Authentication Protocols (JWT, OAuth)**: Used by the **Security Controller** for login and authentication.
- **WebSockets**: Used for real-time messaging between users.

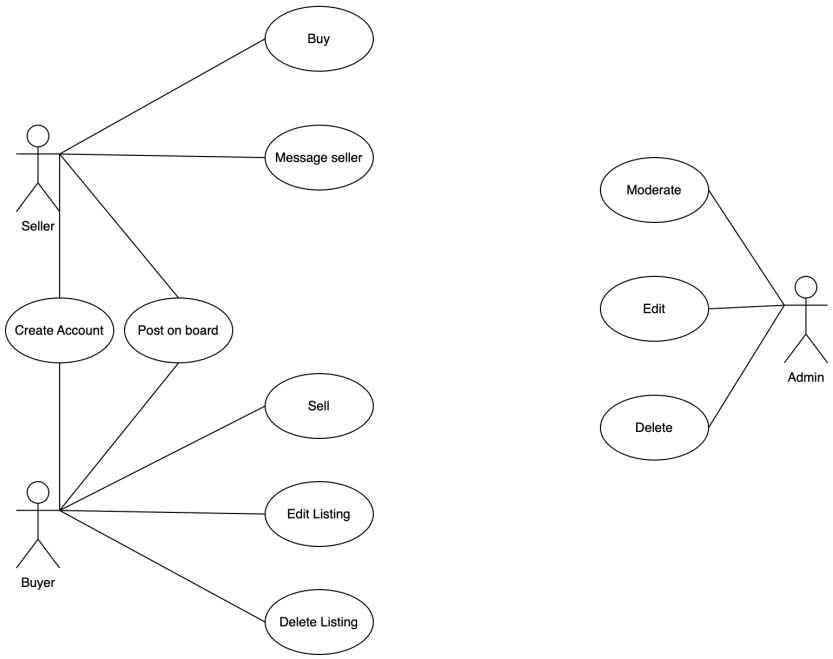
Open AI Chat

- User interacts with Open AI **route** to **create** and **search** listings
- Open AI route search through the database to return relevant listings

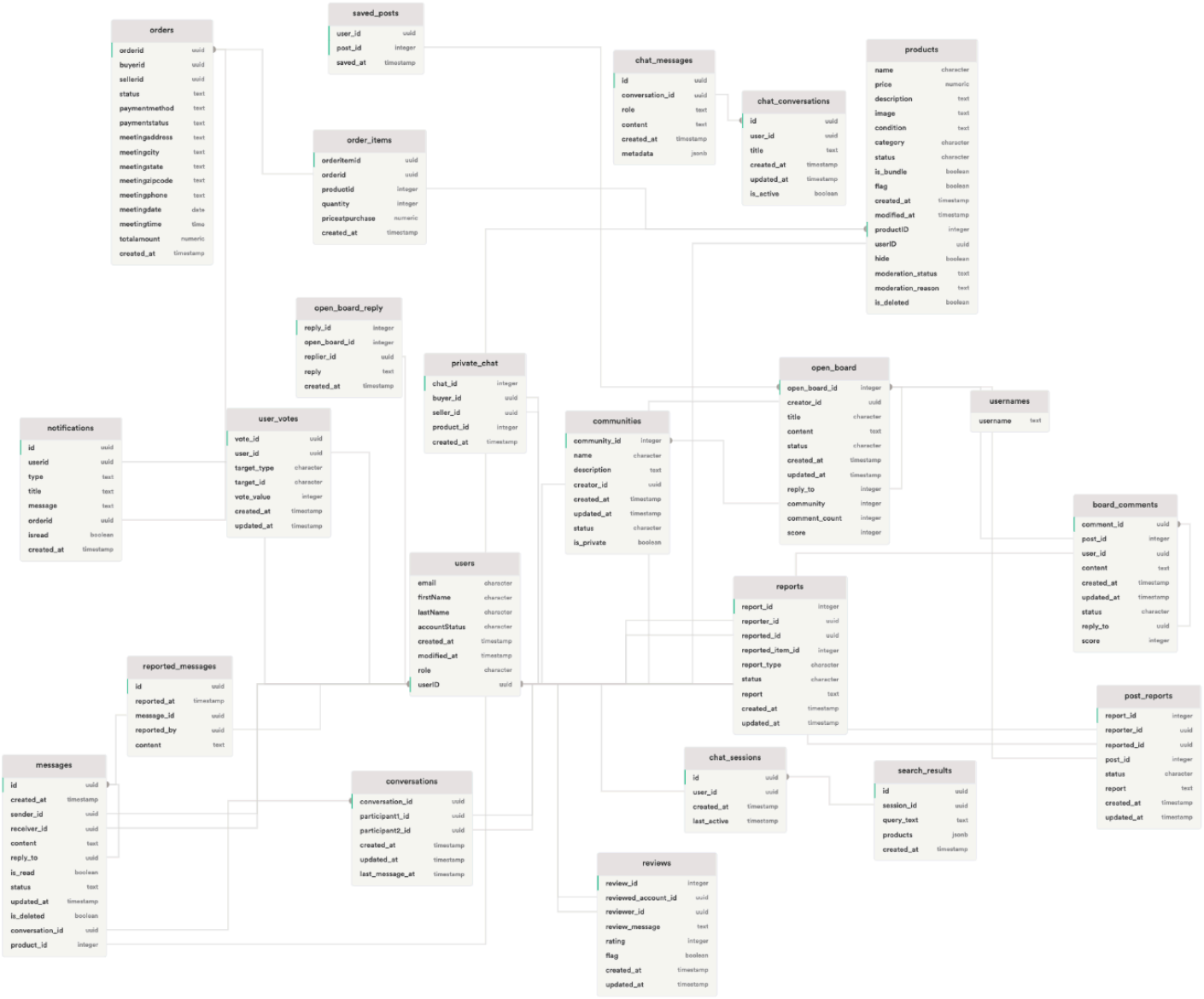
4.2. Overall System Architecture



4.3. Use-Case model



4.4. Relational schema



---

5. System – Analysis Perspective

5.1. Identify subsystems – analysis point of view

Definitions and Acronyms

Definitions	Descriptions
UNCG	University of North Carolina Greensboro
Buyer	User on the site that will be able to buy listings.
Seller	User on the site that will be able to sell listings.
Admin	User on the site that will be able to moderate.
HTML	Language used to make structured web pages.
Controller	Handles request flow.
Model	Handles data logic.
View	Handles data presentation.
Relational Database	A database that organizes data into tables of rows and columns.
Open Board	Community chat for UNCG students.
Rest API	A way for applications to communicate with one another.
SQL	Language used to manage and manipulate data in relational databases.
UUID	Universally Unique Identifier

5.2. Use-Case Descriptions

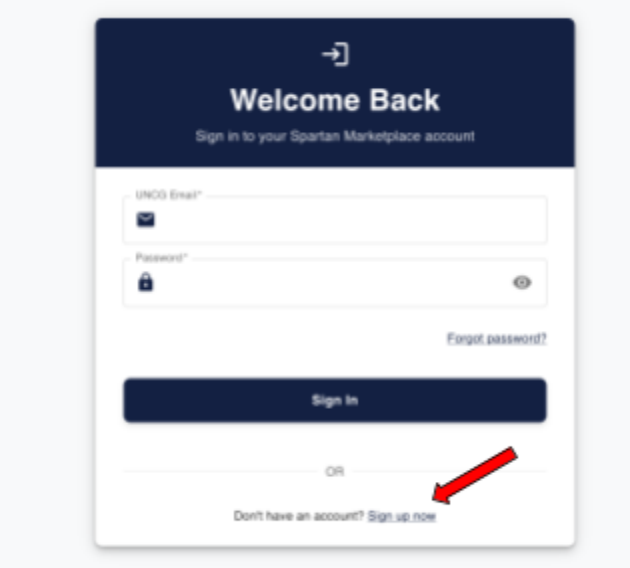
User: Account Creation

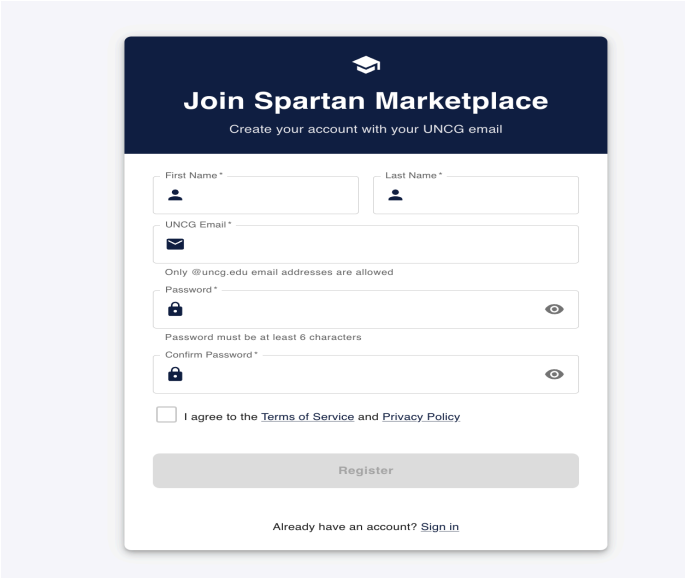
Use Case Description

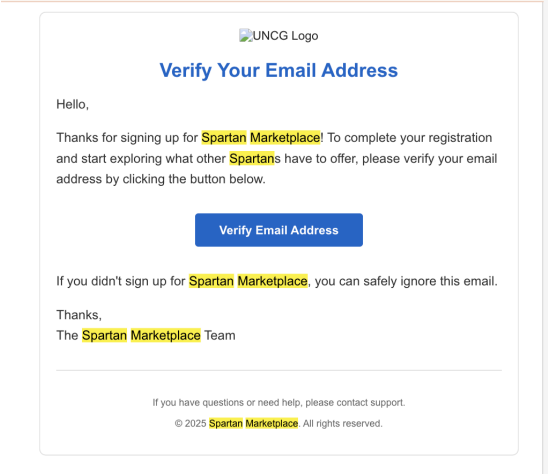
Use Case	Account Creation
Actors	Students, faculty
Preconditions	User has valid UNCG email User has clicked create account
Basic Sequence	1. click on “Sign up now” 2. Enter information asked 3. click on “Register” 4. account successfully created message displayed 5. Email is sent
Exceptions	● User doesn’t have a valid UNCG account ● User already has an existing account

Post Conditions	• Account is created • redirected to homepage

3.1.2. Layout







Data Objects

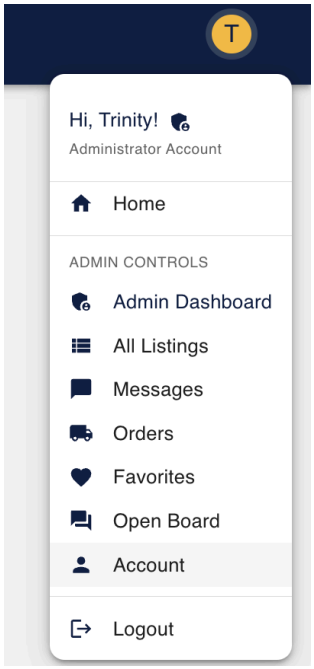
Use Case	Data	Description
Account Creation	email	User sign-in email
	password	User sign-in password
	first Name	User first name
	last Name	User last name

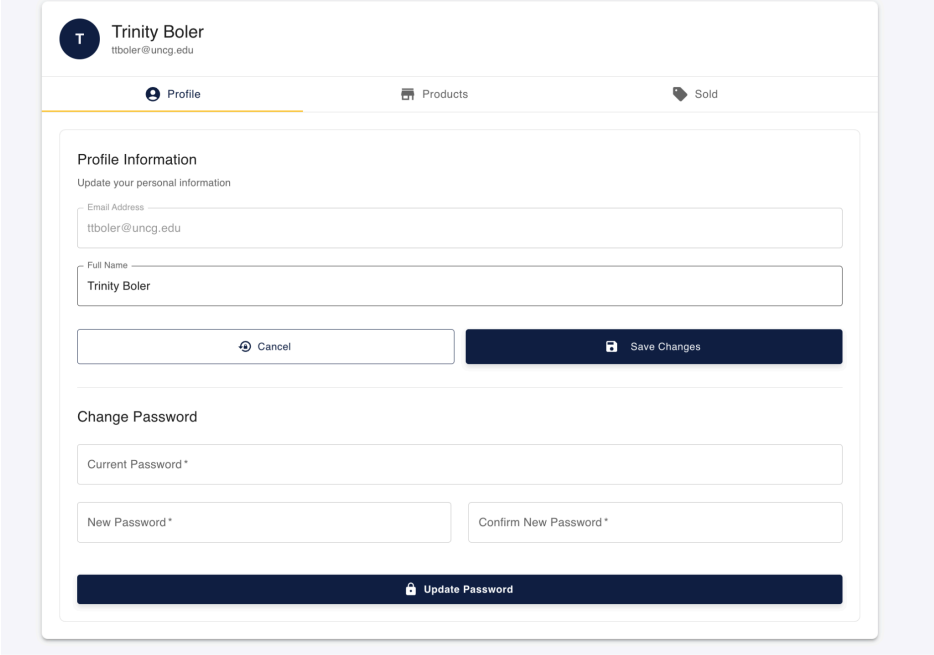
User: Edit and Modify Account information

Use Case Description

Use Case	Edit and Modify Account information
Actors	User
Preconditions	If the account is already created and logged in
Basic Sequence	1. User click on icon 2. User click “Account” 3. USer change name or click change password 4. User clicks “Save changes” or “update password” to save edited information
Exceptions	The user has not created an account
Post Conditions	The account is updated

Layout





Data Object

Use Case	Data	Description
Edit and Modify Account	email	User sign in email
	password	User sign in password

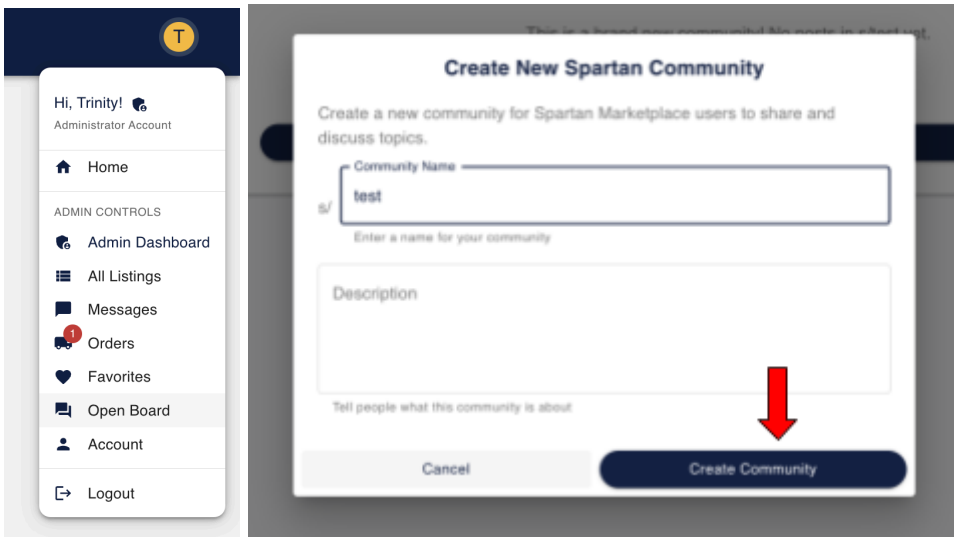
	userId	The unique ID for the user

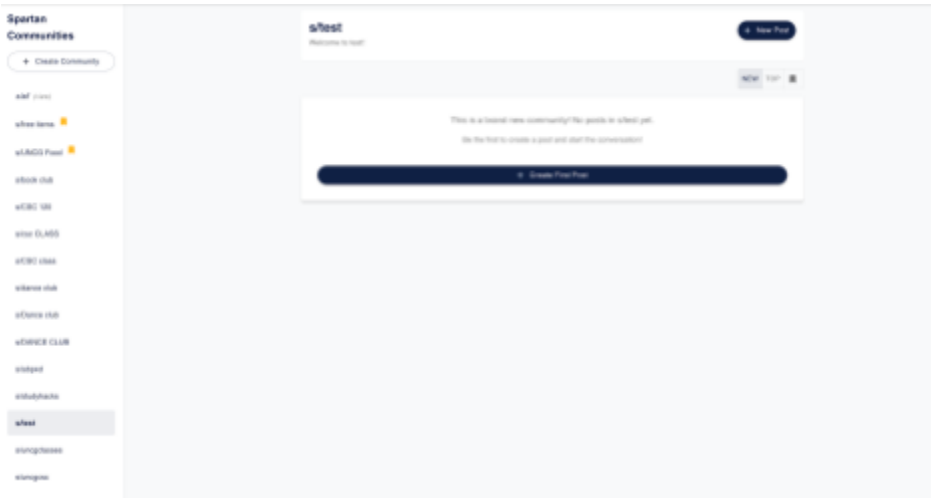
User: Creates a community board

Use Case Description

Use Case	Post to the message board
Actors	User
Preconditions	User is logged in
Basic Sequence	1. User clicks “Open Board” icon 2. User clicks on “Create post”. 3. Users fill out information 4. Users click to create posts
Exceptions	The post creation is canceled.
Post Conditions	Community posts are successfully posted and can be viewed by everyone.

Layout





Data Object

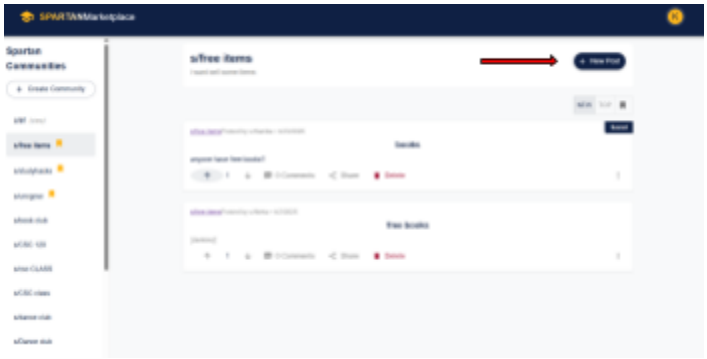
Use Case	Data	Description
Creates a community board	userId	The unique ID for the user
	communityID	The unique ID for the community post

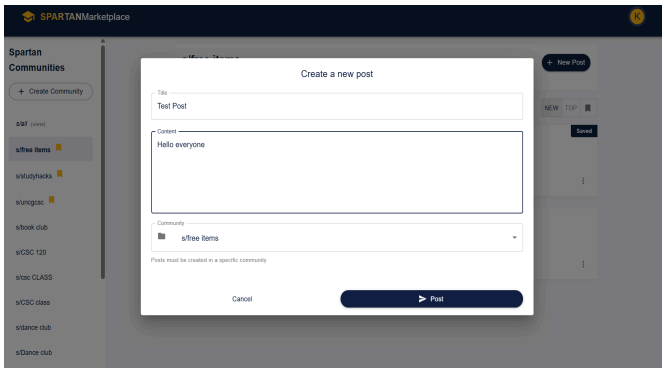
User: Post to the Open board

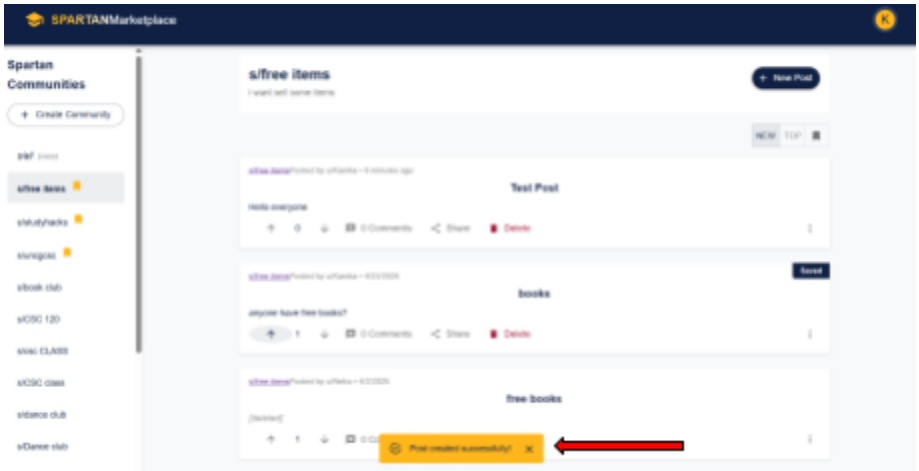
Use Case Description

Use Case	Post to the open board
Actors	User
Preconditions	User is logged in User is on a community
Basic Sequence	1. The user clicks “create first post” 2. User fills out information and click “post”
Exceptions	The post isn’t created.
Post Conditions	The Community post is successfully posted, notification shows and now post can be viewed by everyone.

Layout







Data Object

Use Case	Data	Description
Post to the message board	userId	The unique ID for the user
	List[post]	Message on the board
	post	Message user adds to the open board

User: Interact with Message Board Post (upvote, downvote, delete, save)

Use Case Description

Use Case	Interact with Message Board Post (upvote, downvote, delete, save)
Actors	User
Preconditions	User is logged in User is on the board tab
Basic Sequence	1. User clicks on the community board 2. The user is on the post and clicks one or multiple of the following: - upvote - downvote - delete - : - User clicks Save
Exceptions	post is not updated
Post Conditions	The post upvote increases by one The post upvote decreased The post is deleted and left as [deleted] The post is saved with a bookmark icon on the post

Layout

Data Object

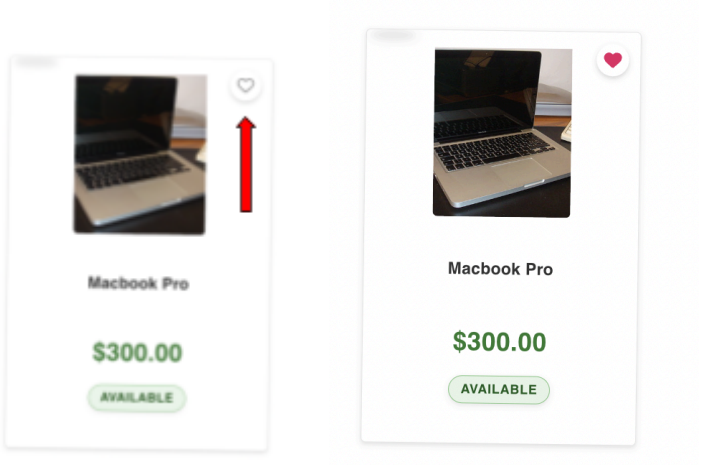
Use Case	Data	Description
Interact with Message Board Post (upvote, downvote, delete, save)	userID	The unique ID for the user
	CommunityID	The unique ID for the community
	PostID	The unique ID for the post

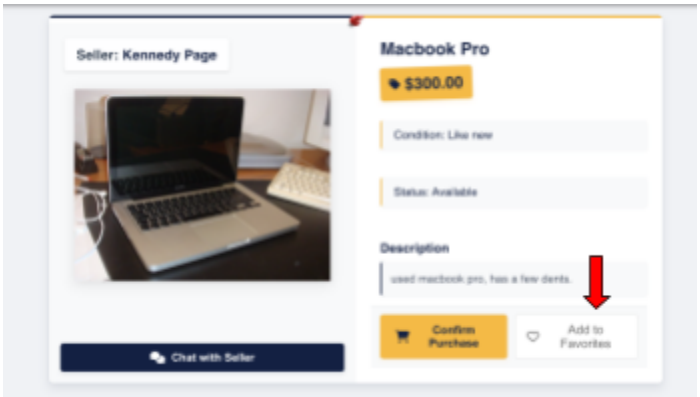
User: favorite items

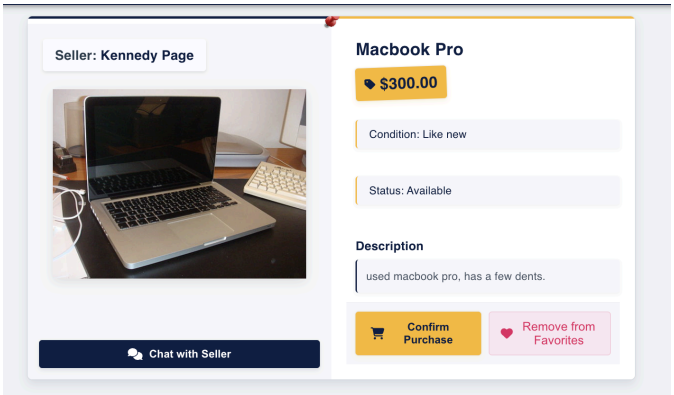
Use Case Description

Use Case	User favorite items
Actor	User
Preconditions	User is logged in User is viewing all listings
Basic Sequence 01	1. User clicks heart icon a. Click again to unfavorite
Basic Sequence 02	2. User clicks on listing 3. User clicks on “add to favorites” a. Click again to unfavorite
Exceptions	Post doesn't favorite
Post Conditions	Post is a favorite or an unfavorite

Layout







Data Object

Use Case	Data	Description
User favorite items	UserID	The unique ID for the user
	ProductID	The unique ID for the product

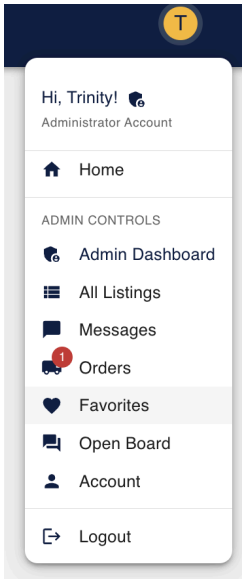
User views favorite items

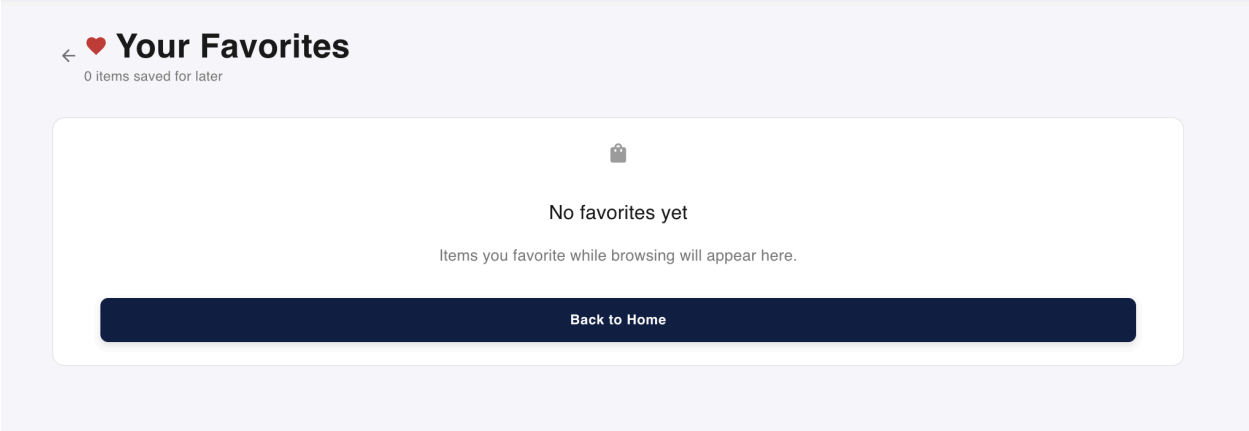
Use Case Description

Use Case	User views favorite items
Actor	User
Preconditions	User is logged in User has favorited item
Basic Sequence	1. User click on account icon 2. User clicks “favorts” tab 3. User views all favorites

	4. User click filter to filter through favorites 5. User clicks clear to clear all favorites a. User clicks trash icon to remove
Exceptions	User has no favorited items
Post Conditions	User viewed favorites User has removed item from favorites User has cleared favorites

Layout





Data Object

Use Case	Data	Description
User views favorite items	UserID	The unique ID for the user
	ProductID	The unique ID for the product

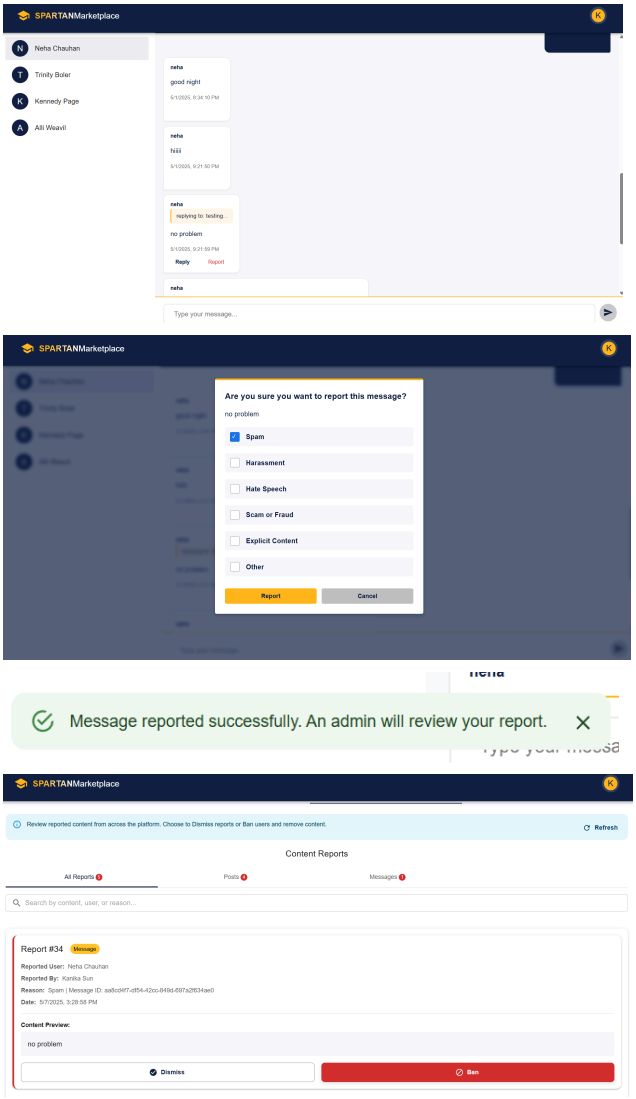
User: Report messages

Use Case Description

Use Case	Report messages or listings
Actors	User, Admin
Preconditions	The user is logged in to the e-commerce site. The user has found a message or listing that they wish to report. The system has mechanisms for reporting messages and listings (e.g., buttons or forms).
Basic Sequence	1. The user navigates to the message or listing they want to report. 2. The user clicks the “Report” button next to the message. 3. The system displays a report form where the user can select a reason for the report (e.g., inappropriate content, misleading description, etc.). 4. The user selects a reason and submits the report. 5. The system logs the report and sends a notification to the admin. 6. The admin reviews the report in the admin panel. 7. The admin takes action on the message/listing (e.g., removes it, sends a warning, or leaves it as is).
Exceptions	● If the user is not logged in, they are prompted to log in before they can report. ● If the user does not select a reason or provides incomplete information, the system asks for the necessary details to complete

	the report. • If the admin does not take action within a certain timeframe, the system may send a reminder notification to the admin.
Post Conditions	• The message/listing is either removed, flagged, or left unchanged based on the admin's decision. • The user is notified that their report has been submitted. • The admin's decision is logged for future reference.

Layout



Data Object

Use Case	Data	Description
Report messages or	email	User sign in email

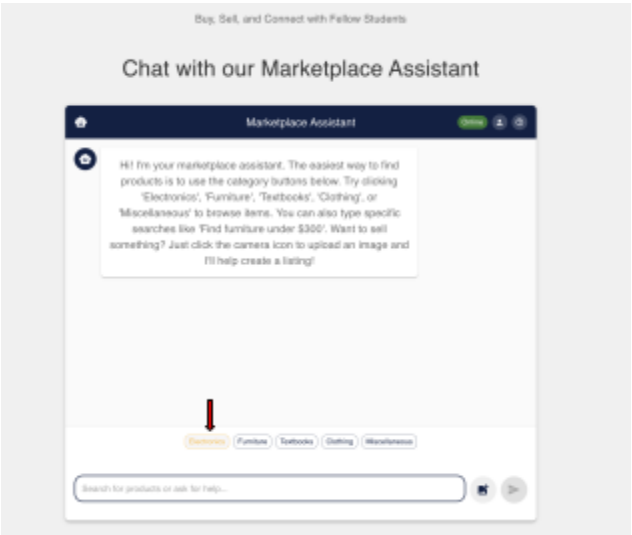
listings	password	User sign in password
	userId-buyer	The unique ID for the user account
	userId-seller	The unique ID for the user account
	chatId	The unique ID for the chat
	productId	The unique ID for the product
	reportId	The unique ID for the report made

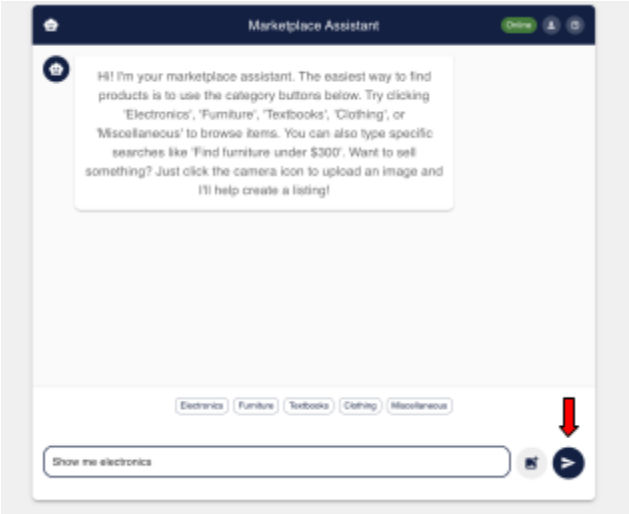
User: Search with AI Chat

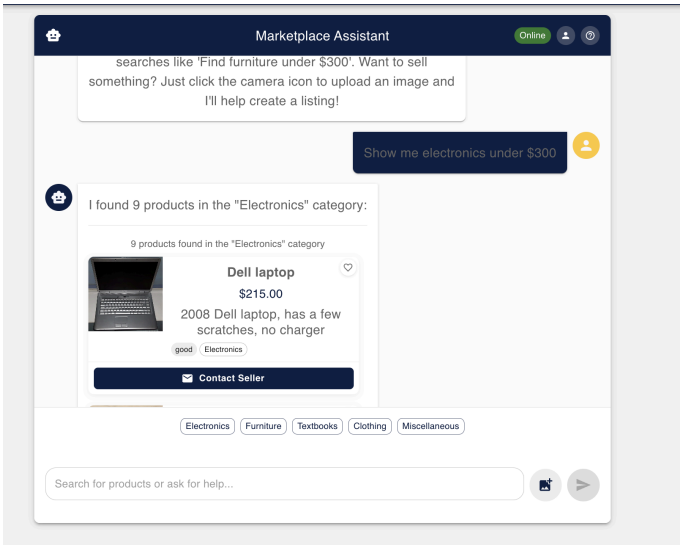
Use Case Description

Use Case	User Search with AI Chat
Actor	User
Preconditions	User is logged in User is on the homepage
Basic Sequence	1. User clicks on one of the category buttons 2. User adds to the text box more specific instructions 3. User clicks the send icon
Exceptions	User enters complex instructions that the AI cant respond to
Post Conditions	AI gives listings based on search conditions

Layout







Data Object

Use Case	Data	Description
User Search with AI Chat	UserID	The unique ID for the user account

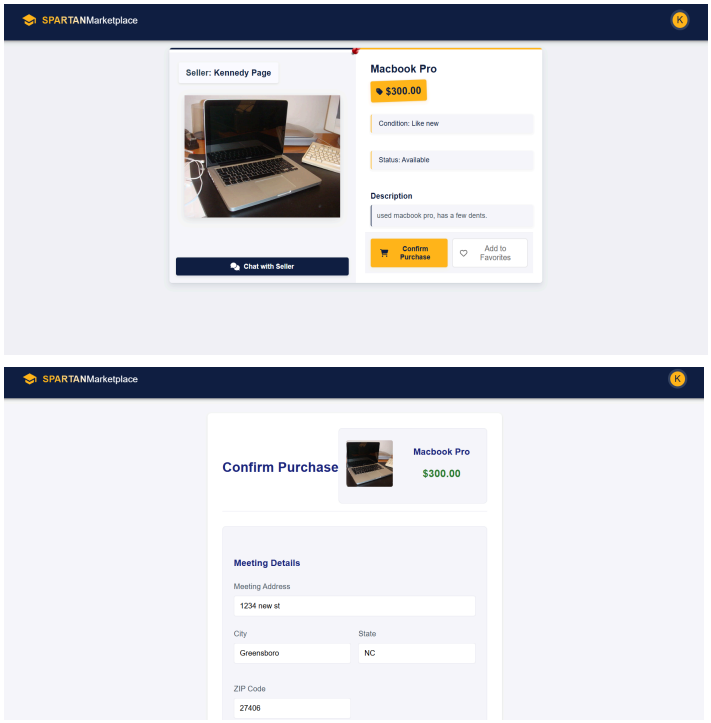
Buyer: Making a Purchase

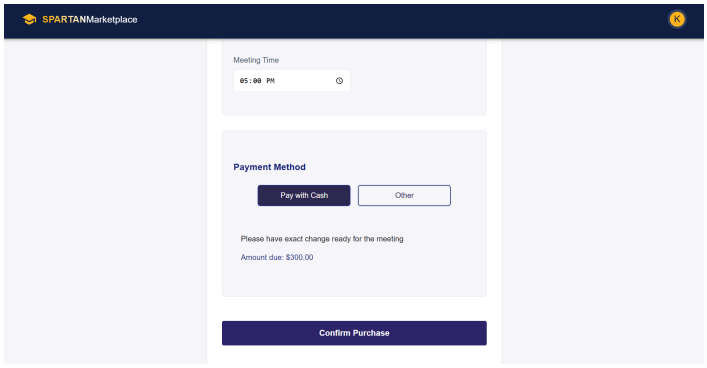
Use Case Description

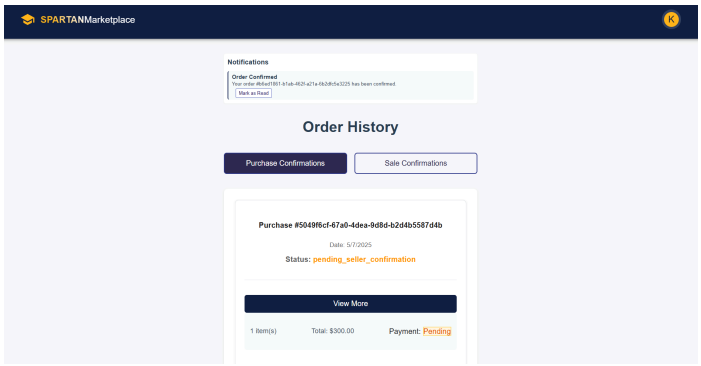
Use Case	Making a Purchase

Actors	Buyer, Seller
Preconditions	Buyer is logged in
Basic Sequence	1. Buyer clicks “Buy now” on a listing and adds to cart. 2. Buyer selects a payment method. 3. The system processes the action via payment gateway. 4. Seller is notified of the sale 5. Buyer receives confirmation.
Exceptions	Payment fails prompting the buyer to retry. Items are already sold and the system displays “Item Unavailable”.
Post Conditions	Purchase is successful.

Layout







Data Object

Use Case	Data	Description
Making a Purchase	email	User sign in email
	password	User sign in password
	userId-buyer	The unique ID for the user account
	userId-seller	The unique ID for the user account
	productId	The unique ID for the listing
	orderId	The unique ID for the order

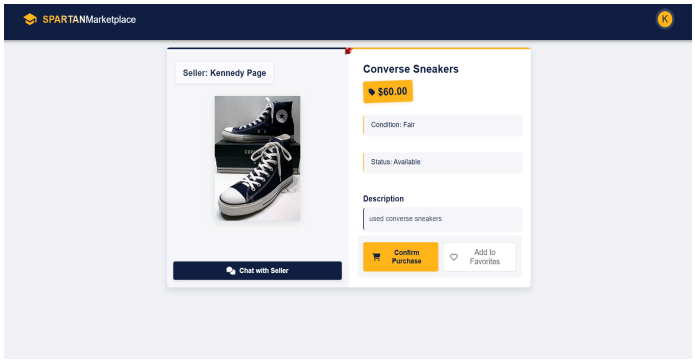
Buyer: Directly message seller

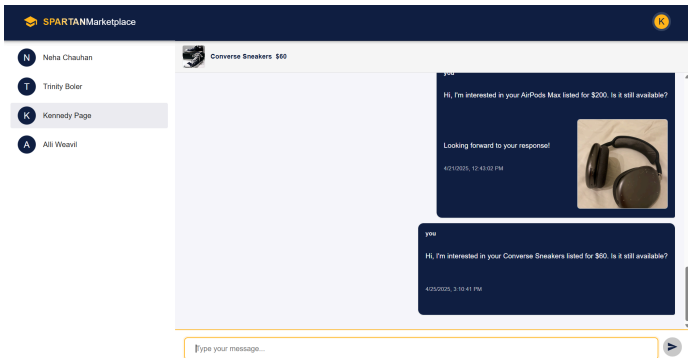
Use Case Description

Use Case	Directly message seller
Actors	Buyer, Seller
Preconditions	Buyer is logged in viewing a listing.
Basic Sequence	1. Buyer clicks “Message Seller” on a listing.

	2. System opens a chat window. 3. Buyer types and sends a message. 4. Seller receives a notification. 5. Buyer and seller communicate about the product.
Exceptions	Seller has blocked the buyer.
Post Conditions	Message sent.

Layout





Data Object

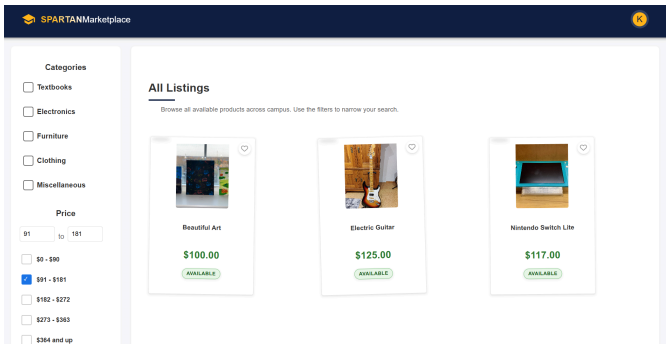
Use Case	Data	Description
Directly message seller	email	User sign in email
	password	User sign in password
	userId-buyer	The unique ID for the user account
	userId-seller	The unique ID for the user account
	chatId-private	The unique ID for the chat
	List[message]	List of messages

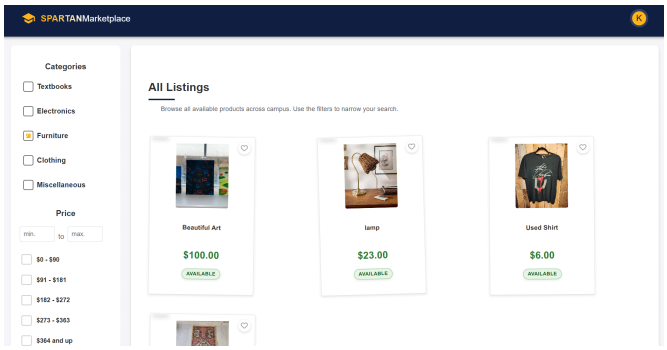
Buyer: Search and filter catalog

Use Case Description

Use Case	Search and filter catalog
Actors	Buyer
Preconditions	Buyer is logged in.
Basic Sequence	1. Buyer enters search keywords or selects filters. 2. The system retrieves relevant listings. 3. Buyer clicks on a listing to view details.
Exceptions	No matching result and system displays “No results found” message
Post Conditions	Matching results are shown.

Layout





Data Object

Use Case	Data	Description
Search and filter	List[product]	list of products

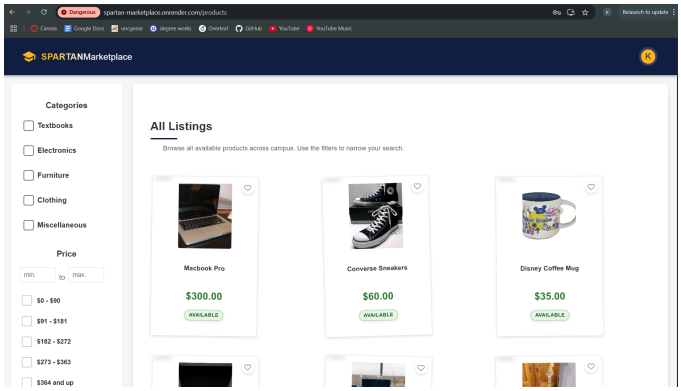
catalog		

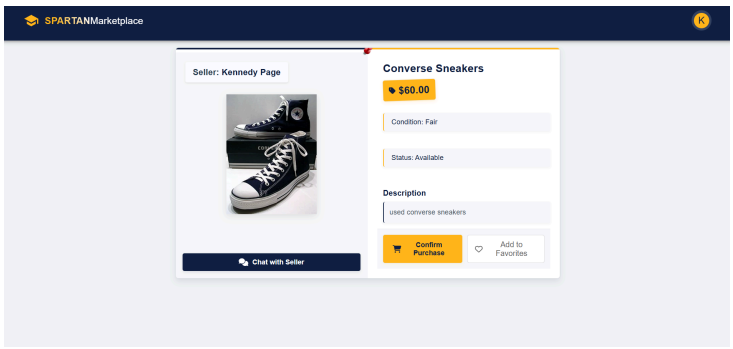
Buyer: View available listing

Use Case Description

Use Case	View available listing
Actors	Buyer
Preconditions	Buyer is logged in.
Basic Sequence	1. Buyer clicks on a product listing. 2. System displays item details (title, price, seller info, image, description)
Exceptions	Listing is deleted or unavailable before a buyer clicks. Listing doesn't exist.
Post Conditions	Buyer can take action to either message the seller, save the item, or buy the product.

Layout





Data Object

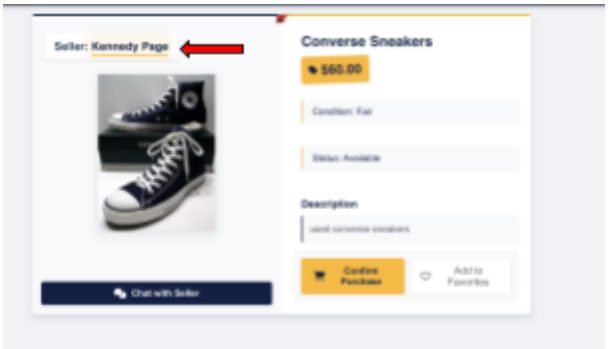
Use Case	Data	Description
View available listing	List[product]	list of products

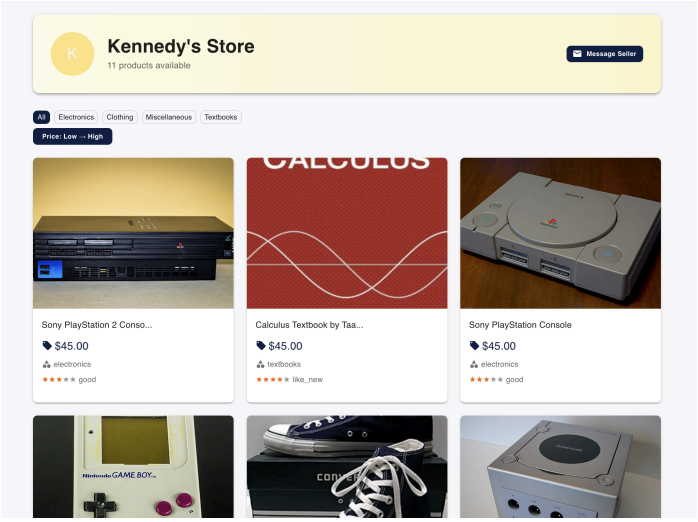
Buyer: View Seller Store

Use Case Description

Use Case	Buyer Views Seller Store
Actors	Buyer
Preconditions	Buyer is logged in and viewing a listing
Basic Sequence	1. Buyer clicks on listing details 2. Buyer clicks on the seller's name
Exceptions	The seller has no products
Post Conditions	Buyer views the seller's page

Layout





Data Object

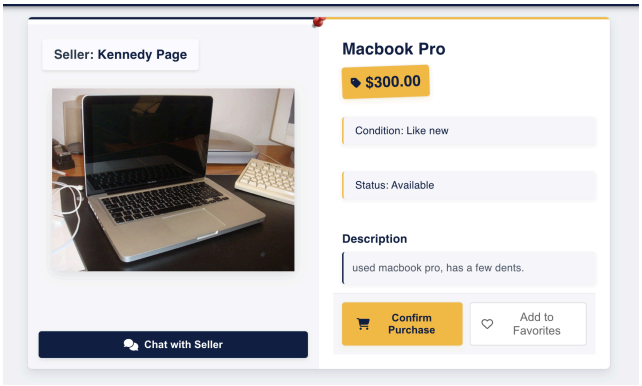
Use Case	Data	Description
Buyer Views Seller Store	UserID	The unique ID for the user account
	ProductID	The unique ID for the Product
	SellerID	The unique ID for the seller

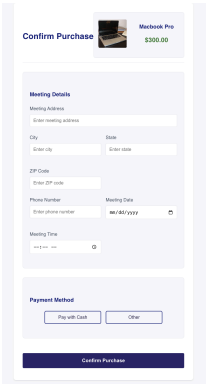
Buyer: Checkout

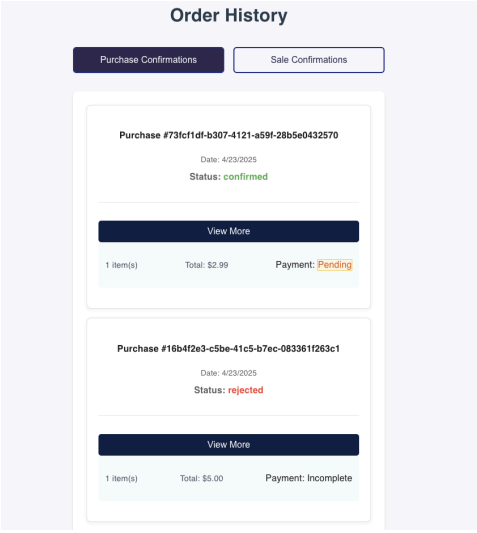
Use Case Description

Use Case	confirms order
Actors	Buyer Seller
Preconditions	Buyer is logged in. The seller has bought an item and checked out
Basic Sequence	1. The buyer finds an item and checks out 2. The buyer clicks confirm purchase 3. Buyer fills out information 4. Buyer clicks confirm purchase 5. The Buyer gets taken to the purchase history
Exceptions	Order Rejection confirmation is sent ot the buyer once the seller has rejected
Post Conditions	The order is sent out the seller

Layout







Data Object

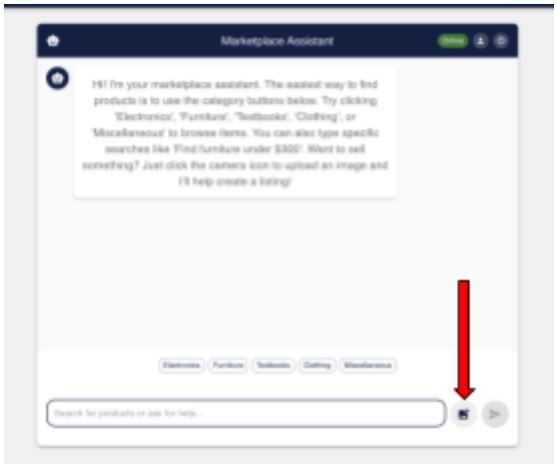
Use Case	Data	Description
Leave Review	email	User sign in email
	password	User sign in password
	userId-buyer	The unique ID for the user account
	userId-seller	The unique ID for the user account
	reviewId	The unique ID for the review
	review	review text
	rating	number rating of review

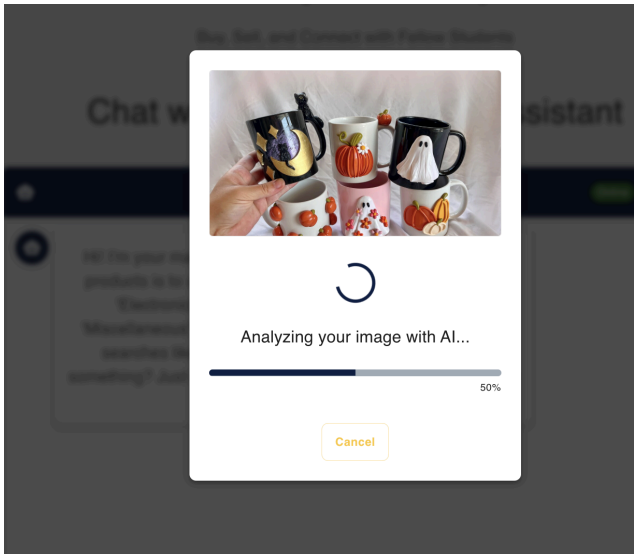
Seller: Post Listing with AI Chat

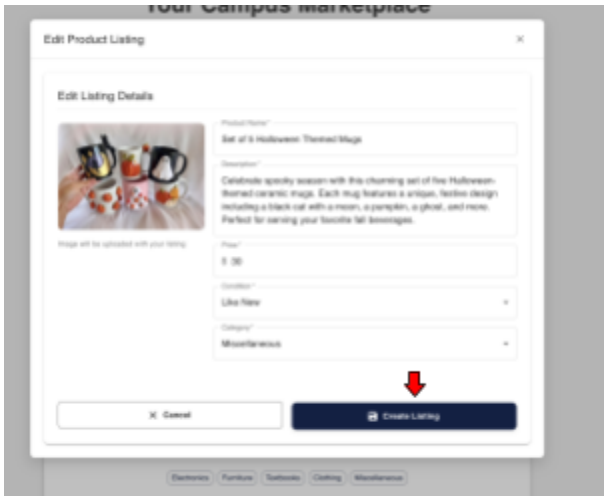
Use Case Description

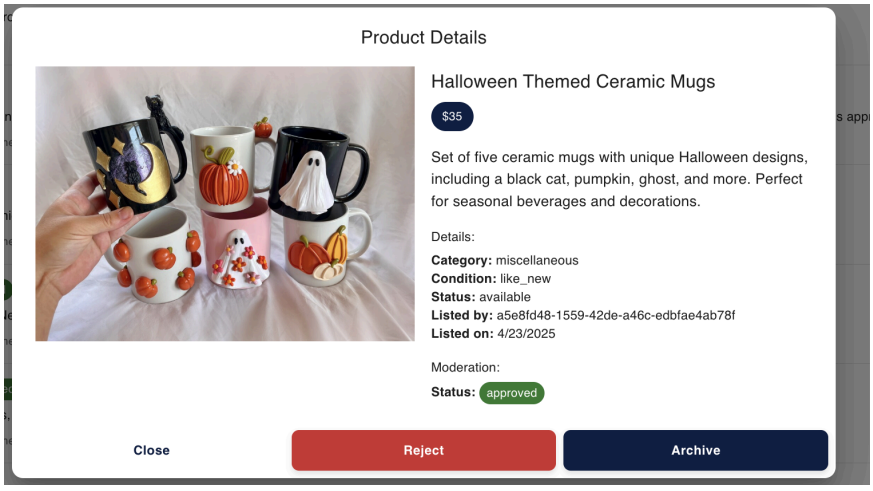
Use Case	Seller Post Listing with AI Chat
Actors	Seller, Admin
Preconditions	seller is signed in
Basic Sequence	1. Seller clicks on the image icon 2. Seller selects image to upload 3. Seller looks over the given listing from chat 4. Seller makes modifications and clicks “Create Listing” 5. Admin approves listing
Exceptions	● AI fails to create listing ● Seller failed to upload image ● User clicks cancel ● Admin Rejects listing
Post Conditions	The product is uploaded to the site.

Layout


Marketplace Assistant


Analyzing your image with AI...
50%
Cancel


Edit Product Listing
Edit Listing Details
Cancel


Product Details
Halloween Themed Ceramic Mugs
$35
Set of five ceramic mugs with unique Halloween designs, including a black cat, pumpkin, ghost, and more. Perfect for seasonal beverages and decorations.
Details:
Category: miscellaneous
Condition: like_new
Status: available
Listed by: a5e8fd48-1559-42de-a46c-edbfae4ab78f
Listed on: 4/23/2025
Moderation:
Status: approved
Close
Reject
Archive

Data Object

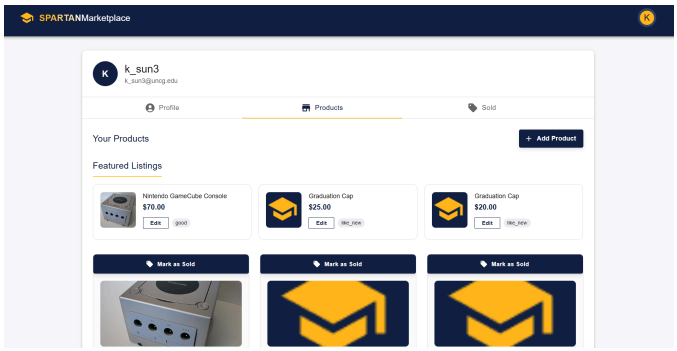
Use Case	Data	Description
Seller Post Listing with AI Chat	UserID	The unique ID for the user account
	ProductID	The unique ID for the user product

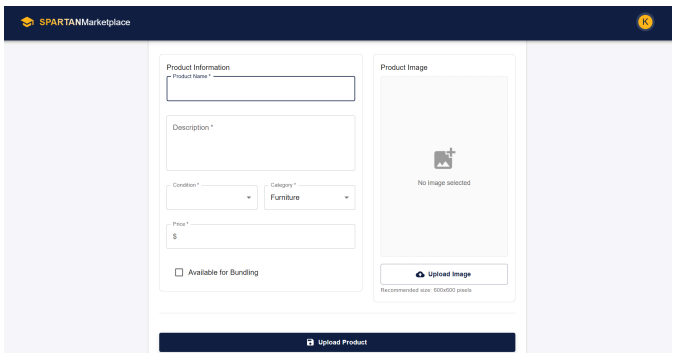
Seller: Post listing

Use Case Description

Use Case	Post listing
Actors	Seller
Preconditions	Seller has opened their listings and decided to create a new listing.
Basic Sequence	1. Seller clicks on “My Listings”. 2. Seller clicks on a button that prompts to create a new listing. 3. Seller adds their listing details. 4. Seller clicks “Publish” and the listing would be available to everybody.
Exceptions	Sellers listing doesn’t save and wouldn’t be published to everyone.
Post Conditions	The listing has been added and shown for everyone.

Layout





Data Object

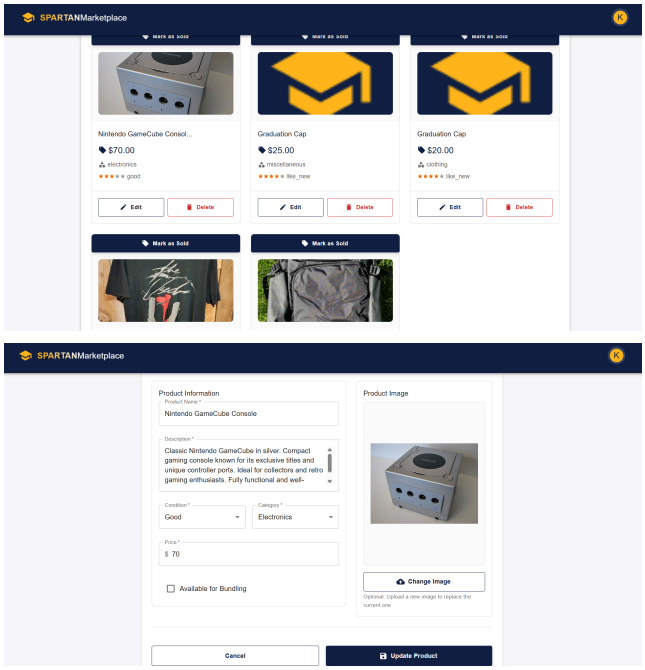
Use Case	Data	Description
Post Listing	email	User sign in email
	password	User sign in password
	userId	The unique ID for the user account
	productId	The unique ID for the product
	name	product name
	price	price of product
	description	details of product
	image	picture of product
	condition	current condition of product
	category	category the product belongs too (textbook, furniture, clothing…)
	status	current states of account
	iskit	the product is a kit (items sold as a bundle)
	modifiedAt	last time item was modified

Seller: Edit Listing

Use Case Description

Use Case	Edit Listing
Actors	Seller
Preconditions	Seller clicks on one of their listings.
Basic Sequence	1. Seller clicks on a button that says “Edit Listing”. 2. They’ll be prompted to a screen where they can change details on the listing. 3. Seller saves their changes. 4. Seller posts the changes of the listing to the public.
Exceptions	The listings changes aren’t saved.
Post Condition	The changes are successfully made.

Layout



Data Object

Use Case	Data	Description
Edit Listing	email	User sign in email
	password	User sign in password
	userId	The unique ID for the user account
	productId	The unique ID for the product
	name	product name
	price	price of product
	description	details of product
	image	picture of product
	condition	current condition of product
	category	category the product belongs too (textbook, furniture, clothing…)
	status	current states of account
	iskit	the product is a kit (items sold as a bundle)

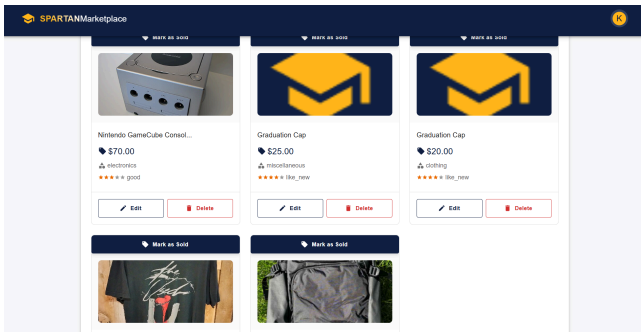
	modifiedAt	last time item was modified

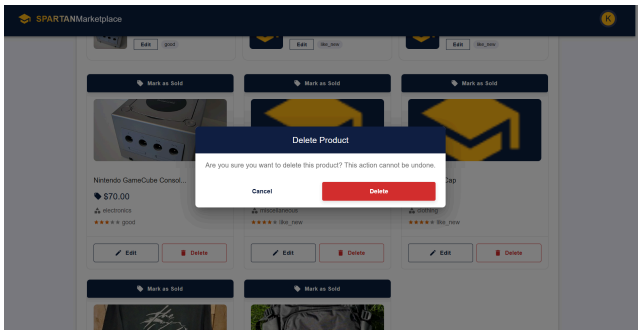
Seller: Delete Listing

Use Case Description

Use Case	Delete Listing
Actors	Seller
Preconditions	Seller clicks on one of their listings.
Basic Sequence	1. Seller clicks on a button to delete a listing. 2. A screen is prompted asking if they are sure about deleting the listing. 3. Seller clicks delete button again to ensure the listing is actually deleted.
Exceptions	Listing doesn't get removed.
Post Condition	The listing has been successfully deleted.

Layout





Data Object

Use Case	Data	Description

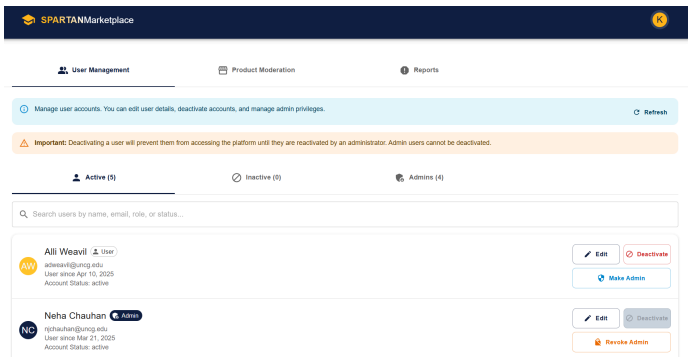
Delete Listing	email	User sign in email
	password	User sign in password
	userId	The unique ID for the user account
	productId	The unique ID for the product

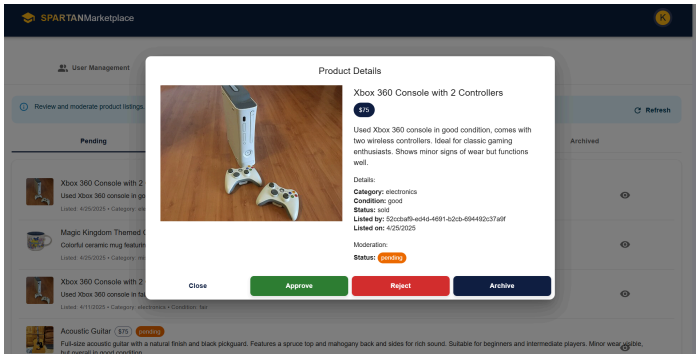
Admin: Delete accounts, messages, listings

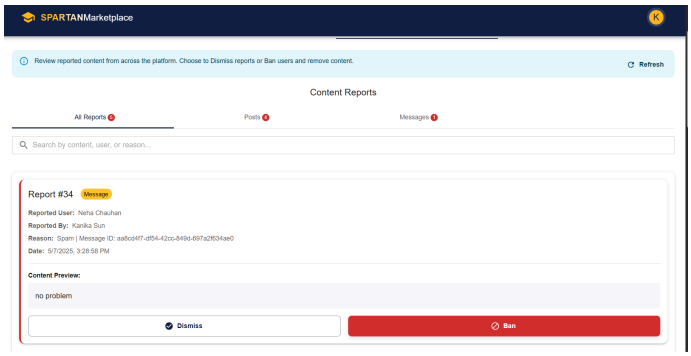
Use Case Description

Use Case	Delete accounts, messages, listing, reviews
Actors	Admin
Preconditions	Admin clicks on an account, message, or listing.
Basic Sequence	1. There is a button that will delete an account, message, or listing. 2. When deleting, there will be a prompt that asks if they are sure about deleting. 3. Admin will then be asked to give a reason for the deletion. 4. After giving a reason, a second delete button will be clicked.
Exceptions	Account, message, or listing doesn't get deleted.
Post Conditions	Account, message, or listing is successfully deleted.

Layout







Data Object

Use Case	Data	Description
Delete accounts, chat, listing	email	User sign in email
	password	User sign in password
	userId-Admin	The unique ID for the admin
	productId	The unique ID for the product
	chatId	The unique ID for the chat
	accountId	The unique ID for the user account

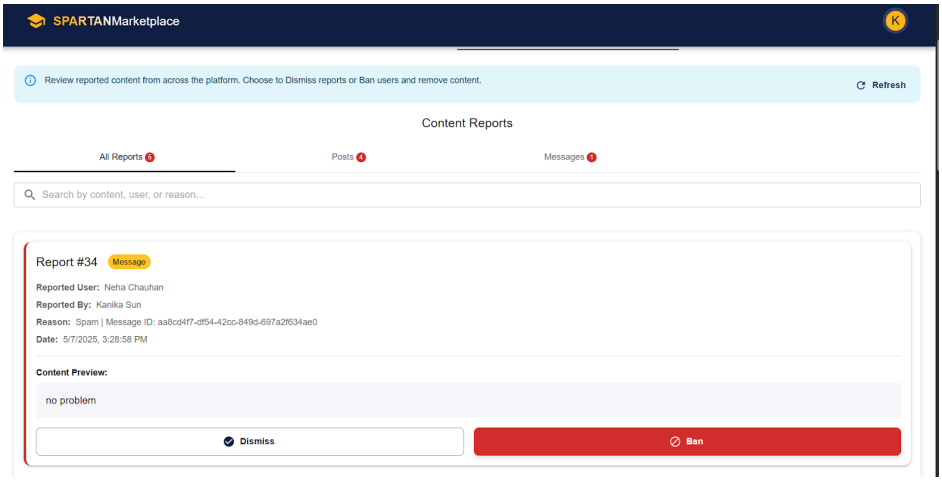
Admin: View Report

Use Case Description

Use Case	View Reports
Actors	Admin
Preconditions	Admin clicks on Reports.
Basic Sequence	1. Admin will click on a button that allows them to view all reports. 2. The admin can view all reports, activities, and account information.

	3. Admin can address the report 4. Admin will then click on a button to exit the report.
Exceptions	No report exists.
Post Conditions	Admin successfully viewed a report.

Layout



Data Object

Use Case	Data	Description
View Report	email	User sign in email
	password	User sign in password
	userId-Admin	The unique ID for the admin
	reportId	The unique ID for the report
	userId- reporter	The unique ID for the person reporting the problem
	report	The description of the report
	reportType	The type of report (message, review, product)
	statue	The status of the report
	reportedItemId	What is being reported with unique ID

5.3. Tables and Descriptions

board_comments Table

board_comments			
Field Name	**Data Type(formate)**	**Accept Null?**	**Attributes**
comment_id	uuid	no	PK
user_id	uuid	no	FK
post_id	integer	no	FK
content	text	no	
created_at	timestamp with time zone	no	
updated_at	timestamp with time zone	no	
status	char	no	
reply_to	uuid	no	FK
score	integer	no	

communities Table

communities			
Field Name	**Data Type(formate)**	**Accept Null?**	**Attributes**
community_id	inter(uuid)	no	PK
name	char	no	
description	text	no	
creator_id	string(uuid)	no	FK
created_at	string(timestamp)	no	
update_at	string (timestamp)	no	
status	char	no	
is_private	boolean	no	

conversation Table

conversation			
Field Name	**Data Type(format)**	**Accept Null?**	**Attributes**
conversation_id	string(uuid)	no	PK
participant1_id	string(uuid)	no	FK
participant2_id	string(uuid)	no	FK
created_at	string(timestamp)	no	
updated_at	string(timestamp)	no	
last_message_at	string(timestamp)	no	

messages Table

messages			
Field Name	**Data Type(fomate)**	**Accept Null?**	**Attributes**
id	string(uuid)	no	PK
created_at	timestamp with time zone	no	
sender_id	string(uuid)	no	FK
reciver_id	string(uuid)	no	FK
content	string(text)	no	
reply_to	string(uuid)	no	FK
is_read	boolean	no	
status	string(text)	no	
updated_at	string(timestamp)	no	
is_deleted	boolean	no	
conversation_id	string(uuid)	no	FK
product_id	integer	no	FK

notifications Table

notifications			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
id	string(uuid)	no	PK
userid	string(uuid)	no	FK
type	string(text)	no	
title	string(text)	no	
message	string(text)	no	
orderid	string(uuid)	no	FK
isread	boolean	no	
created_at	string(timestamp)	no	

Open board Table

open_board			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
open_board_id	integer	no	PK
creator_id	string(uuid)	no	FK
title	string(char)	no	
content	string(text)	yes	
status	string(chat)	no	
created_at	string(timestamp)	no	
updated_at	string(timestamp)	no	
reply_to	integer	no	FK
community	integer	no	
comment_count	integer	no	
score	integer	yes	

order items Table

order_items			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
orderitemid	string(uuid)	No	PK
orderid	string(uuid)	No	FK
productid	integer	No	FK
quantity	integer	no	
priceatpurchase	number(numeric)	no	
created_at	string(timestamp)	no	

orders Table

orders			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
orderid	string(uuid)	no	PK
buyerid	string(uuid)	no	FK
sellerid	string(uuid)	no	FK
status	string(text)	no	
paymentmethod	string(text)	no	
paymentstatus	string(text)	no	
meetingadress	string(text)	no	
meetingcity	string(text)	no	
meetingstate	string(text)	no	
meetingstate	string(text)	no	
meetingzipcode	string(text)	no	
meetingphone	string(text)	no	
meetingdate	string(date)	no	
meetingtime	string(timestamp)	noi	

orders			
totalamount	interger	ni	
created_at	string(timestamp)	no	

post_reports table

post_report			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
report_id	Integer	No	PK
reporter_id	string(uuid)	No	FK
reported_id	string(uuid)	No	FK
report	string(text)	No	
created_id	string(timestamp)	No	
updated_id	string(timestamp)	no	

products

Product Table			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
productID	Integer(10)	No	PK
userID	string(uuid)	No	FK
name	string(char)	No	
price	integer	No	
description	string(text)	No	
image	string(text)	No	
condition	string(text)	No	
category	String(char)	No	
status	String(char)	No	

Product Table			
isbundle	Boolean	No	
modified_at	string(timestamp)	No	
created_at	string(timestamp)	no	
flag	boolean	No	
hide	boolean	no	
moderation_staus	string(text)	no	
moderation_reason	string(text)	no	
is_deleted	boolean	no	

reports

reports			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
report_id	integer	no	PK
repoter_is	string(uuid)	no	FK
reported_id	string(uuid)	no	FK
reported_item_id	interger	no	FK
report_type	string(char)	no	
status	string(char)	no	
report	string(text)	no	
created_at	string(timestamp)	no	
updated_at	string(timestamp)	no	

save_post

save_post			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**

save_post			
user_id	string(uuid)	no	FK
post_id	inerger	no	FK
saved_at	string(timestamp)	no	

user_votes

user_votes			
Field Name	**Data Type (format)**	**Accept Null?**	**Attributes**
vote_id	string(uuid)	no	PK
user_id	string(uuid)	no	FK
target_type	string(char)	no	
target_id	string(char)	no	FK
vote_value	integer	no	
created_at	string(timestamp)	no	
update_at	string(timestamp)	no	

users

User Table			
Field Name	**Data Type (length)**	**Accept Null?**	**Attributes**
userId	Integer(10)	No	PK
email	String(50)	No	Unique
password	String(20)	No	
firstName	String(50)	No	
lastName	String(50)	No	
accountStatus	String(10)	No	
created_at	String()	No	

User Table			
modified_at	String()	No	
role	char(10)	no	

5.4 Algorithm Analysis

- Big - O analysis of overall System and Subsystems

Subsystem	Typical Complexity	Worst Case
Auth & User Mgmt	O(log N)	O(N)
Listings	O(log N) to O(N)	O(N log N)
Messaging	O(1) / O(k)	O(N)
Community Board	O(k)	O(N)
AI Assistant	O(1)	O(k)
Admin Tools	O(log N)	O(N)
React Frontend	O(k)	O(n)

---

6. Project Scrum Report

6.1. Implementation Plan – Sprint Timeline

Week	**Kanika S.**
Week 1 02/20 - 02/27	● Set up the backend framework (Node.js/Express) ● Develop functionality for viewing product listings (all users) ● Implement a singular product listing view
Week 2 02/28 – 03/06	● Implement controller logic for messaging functionality ● Work on frontend integration for messaging
Week 3 03/07 – 03/20	● Continue working on messaging controllers (spillover from Week 2)
Week 4 03/21 – 03/27	● Tackle low/minor priority issues

Week 5 03/28 – 04/03	● Develop backend functionalities for the admin panel
Week 6 04/04 – 04/10	● Conduct testing and finalize backend changes ● UI enhancements

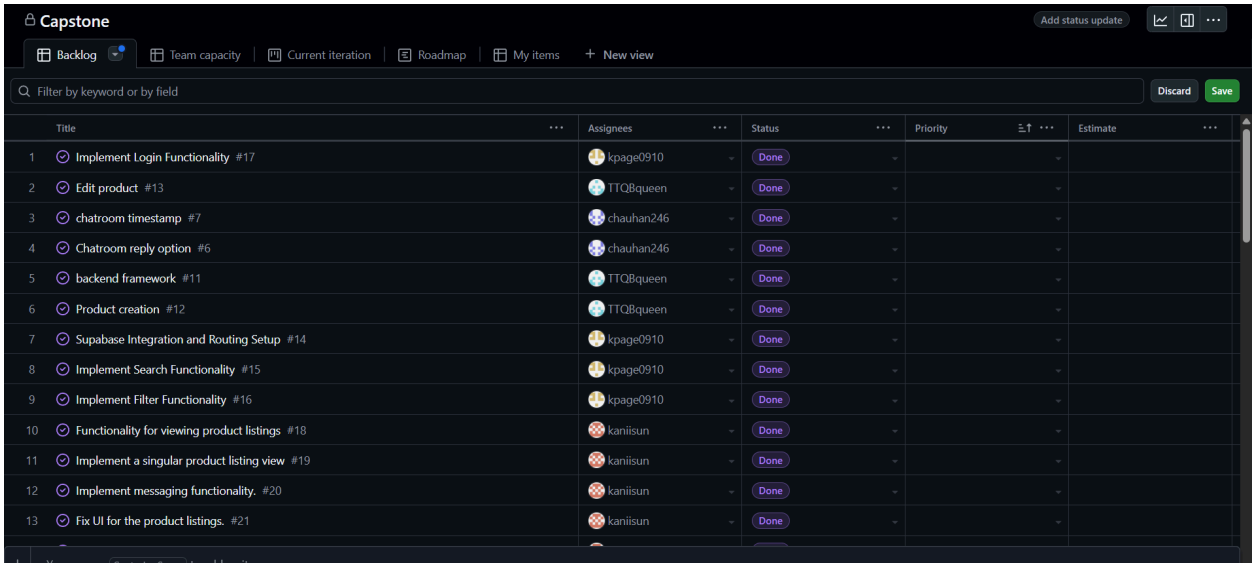
Week	**Trinity B.**
Week 1 02/20 - 02/27	● Set up backend framework ● Implement controller logic for editing listings (seller) ● Implemented controller logic for deleting listings (seller)
Week 2 02/28 – 03/06	● Design it and set up database tables for messaging (chat, open board, replies)
Week 3 03/07 – 03/20	● Continue working on database tables for messaging (spillover from Week 2)
Week 4 03/21 – 03/27	● Address and complete low-priority items
Week 5 03/28 – 04/03	● Address and complete medium/low-priority items styling and notifications ● User Store
Week 6 04/04 – 04/10	● Conduct testing and finalize database modifications ● UI enhancements

Week	**Kennedy P.**
Week 1 02/20 - 02/27	● Set up the database and models for listing data (e.g., seller info, product details)
Week 2 02/28 – 03/06	● Begin implementation of search and filter functionality
Week 3 03/07 – 03/20	● Implement user registration and login API endpoints
Week 4 03/21 – 03/27	● Implement email verification for user registration ● Chatbot fixes
Week 5 03/28 – 04/03	● Develop admin-related functionalities (dashboard, user management)
Week 6 04/04 – 04/10	● Conduct testing and finalize authentication, search, and admin functionalities ● AI image creation for a new product

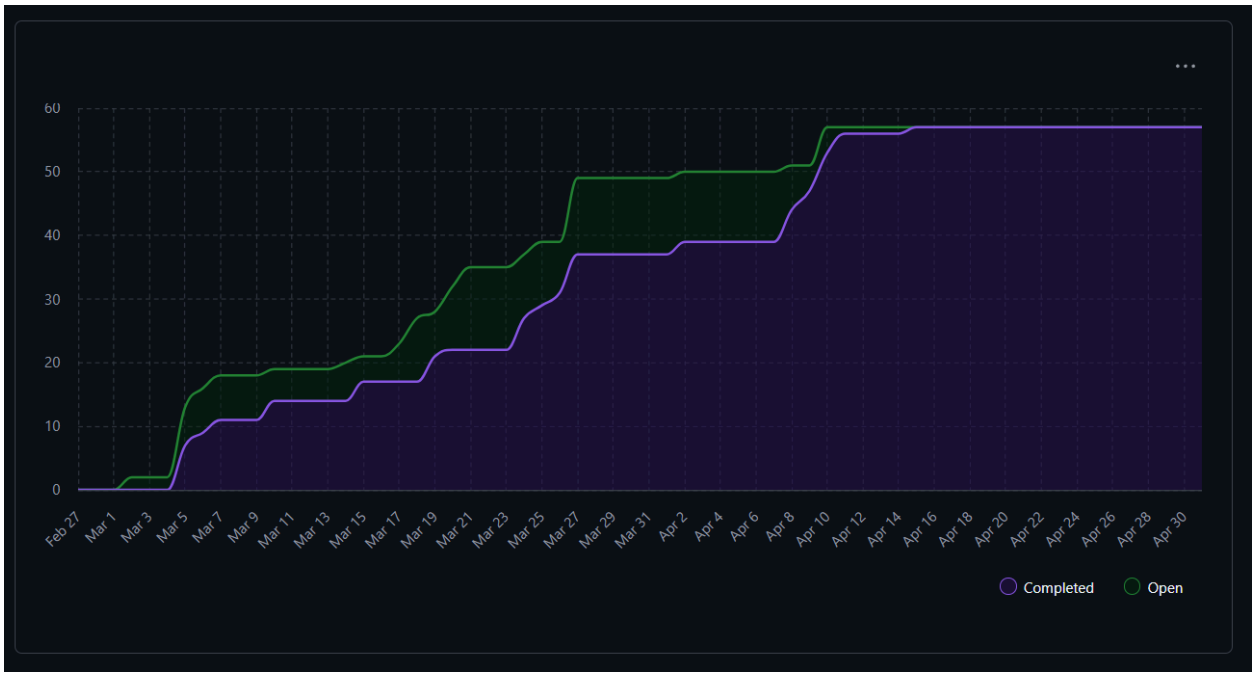
Week	**Nehabahen C.**
Week 1 02/20 - 02/27	● Set up the frontend framework ● Implement frontend UI for posting/creating listings (seller)
Week 2 02/28 – 03/06	● Develop frontend for messaging (chat UI)
Week 3 03/07 – 03/20	● Implement login/logout functionality
Week 4	● Implement account creation and security (Google API, Firebase authentication)

03/21 – 03/27	• Implement an open board chat function
Week 5 03/28 – 04/03	• Develop admin-related UI components and features • Create notification functionality for open board • Create confirmation order proof
Week 6 04/04 – 04/10	• Conduct testing and finalize frontend features • UI enhancements

6.2. GitHub Product Backlog



6.3. Burn Down/Up chart



7. Discussion

7.1. Limitations and future work

We definitely faced some limitations in terms of what we could and couldn't implement on the site, mainly due to our current skill set. We made an effort to stay within the scope of what we were familiar with while gradually learning new technologies like React and Supabase. Looking ahead, it would be exciting to expand the site's usability beyond just UNCG and make it accessible to students at other college campuses as well.

7.2. Other thoughts

---

8. Complete System

- https://github.com/kaniisun/Capstone490.git
- https://www.youtube.com/watch?v=5FPOsDh9m3A