

Team Member Full Name	NetID
Hannah Huston	hhuston
Elissa McDermott	emcderm3
Sophie Chou	schou2

Persistent Storage Design

Persistent Storage Design

We are using SQLite database, provided by the original code outline, to persist our data. Our database includes the tables shown below in Figures 1.1 and 1.2. Figure 1.1 represents the User database within our program. This database contains four different columns, each representing a user input during our registration process that we then save into the database. These columns represent the Username, Email Address, First and Last Name of each of the users. The last column represents the status of whether or not the user has admin/staff access. In this case, the only admin account is “group14” and the other users in the database were created using our website. This persistent storage design allows us to keep track of the users in our app as well as allow them to log in and log out easily. Building upon our original database schema, we added many different classes to help with each of our new features. For example, we added a Product class for Feature 2, a Message class for Feature 3, and UserExtraListings class for Feature 4. Each one of these classes has various attributes that help detail its features as well as many have `__str__` functions to help print out the names correctly. We also have certain definitions that we call throughout the program in order to get information related to the classes such as our `get_default_user()` and `upload_to`. Our `ProductImage` class is the only class that is connected to another as it makes it simpler to add photos to our products rather than trying to do it all within the Product class, though their information is connected.

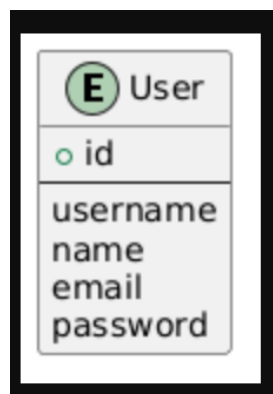


Figure 1.1 Original Database Schema

Classes - Class Diagram

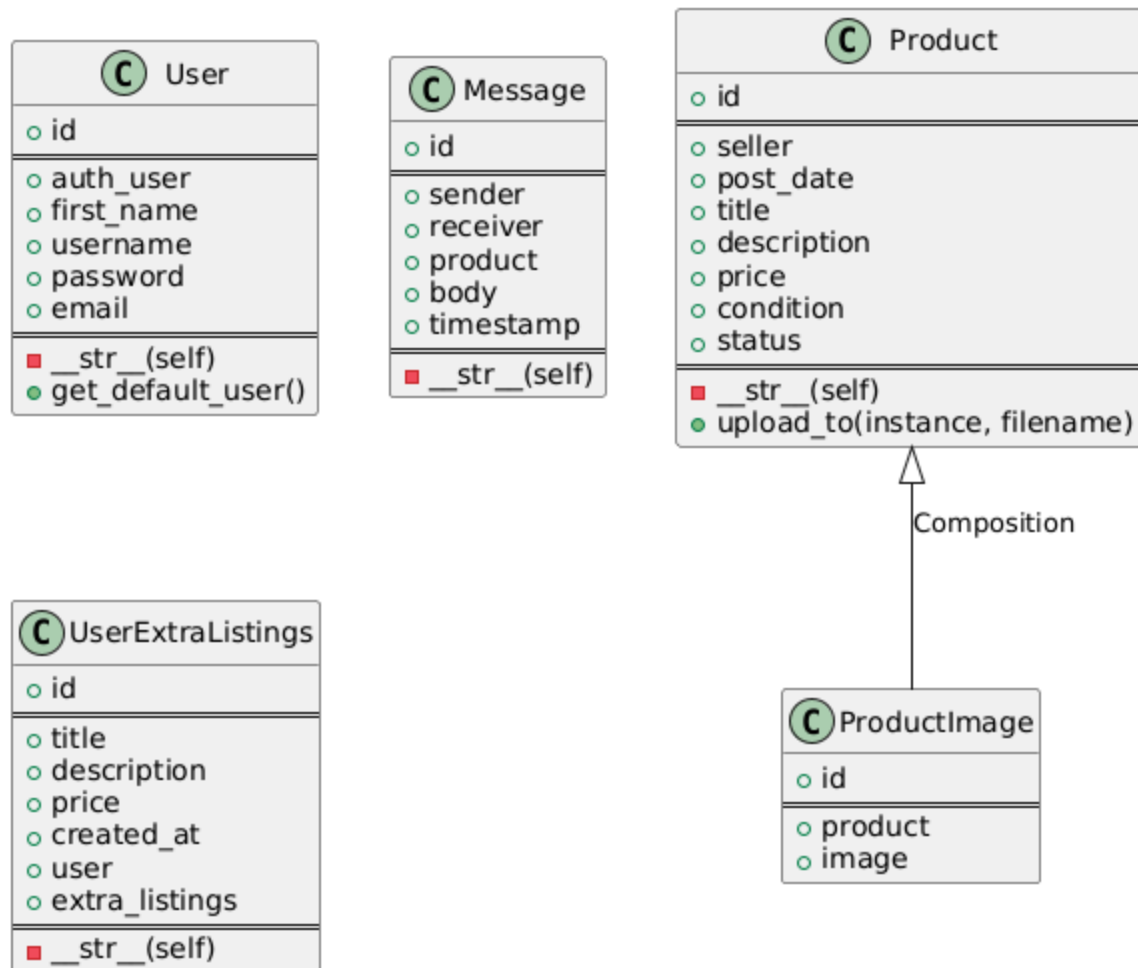



Figure 1.2 Overall Database Schema

Demonstration of the Features

Feature 1

Feature A: 1.1: Create new User Profile

Figure 2 shows a screenshot for our create new user profile page which we call Register. This page is what will open if a user types in our http with the register attribute. A user will enter in their name, preferred username, email, and password in order to create an account. This will be saved into the admin database in order to officially register the user and keep a record of their login information.




Register

Figure 2 Screenshot for Feature A

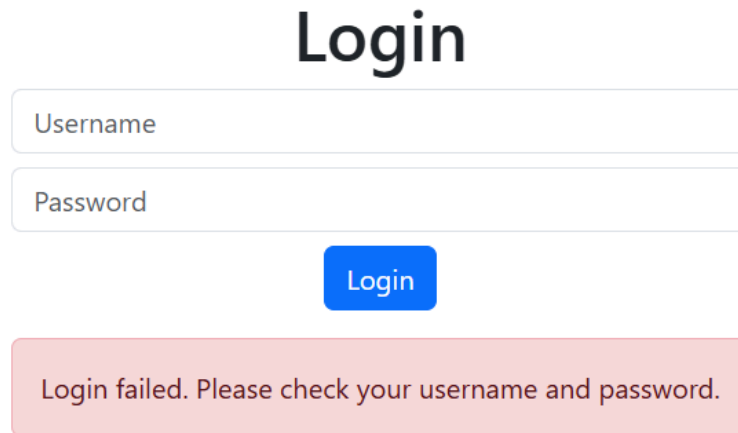
Feature B: 1.2: Log-in

Figure 3 shows a screenshot for our login page! This login page is what a user is brought to after they type in our http followed by the login attribute. This is where a user will enter their previously registered username and password in attempts to login to their CampusMart account. If successful, the user will be brought to the homepage, if unsuccessful, an error message will be displayed and they will stay on the login page. We also included an extra button that allows users to register a new account from the login page.



Login

Figure 3.1 Screenshot for Feature B

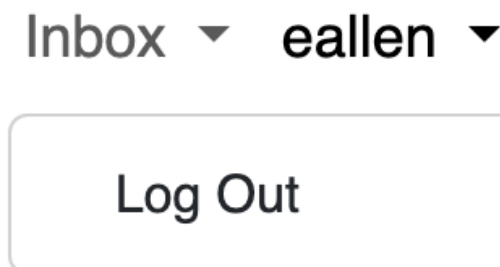


A login form with the title "Login" in a large, bold, black font. Below the title are two input fields: "Username" and "Password", both with light gray borders. A blue "Login" button is positioned below the password field. At the bottom, a light red rectangular box contains the text "Login failed. Please check your username and password." in a dark red font.

Figure 3.2 Screenshot for Feature B

Feature C: 1.3: Log-out

Figure 4 shows a screenshot for our homepage that contains our logout button. This button is placed in an easily accessible location for the user to find once they are logged in on the homepage. When clicked, this button will direct them back to the login page, which means they have successfully logged out.



A header section with the text "Inbox" and a downward arrow, followed by "eallen" and another downward arrow. Below this is a large, light gray rectangular button with rounded corners that contains the text "Log Out" in a bold, black font.

Figure 4.1 Screenshot for Feature C



Login

Log out was successful!

Figure 4.2 Screenshot for Feature C

Feature 2

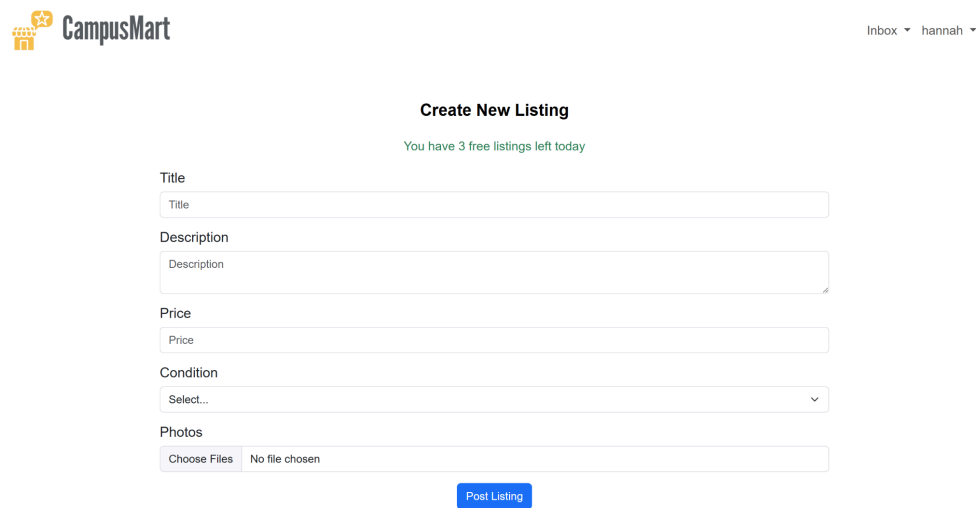
Feature A: 2.1 Create Listings

Figure 5 represents Feature A, create listings. We display a blue button for the user to click in order to make a listing on CampusMart. We also display how many free listings they still have in green at the top in order to give them an easy visual reminder! After a listing is made, we also give a pop-up to let members know that their listing has been successfully posted!

Welcome to CampusMart, Emma !

You have 3 listings left

Figure 5.1 Screenshot for Feature A

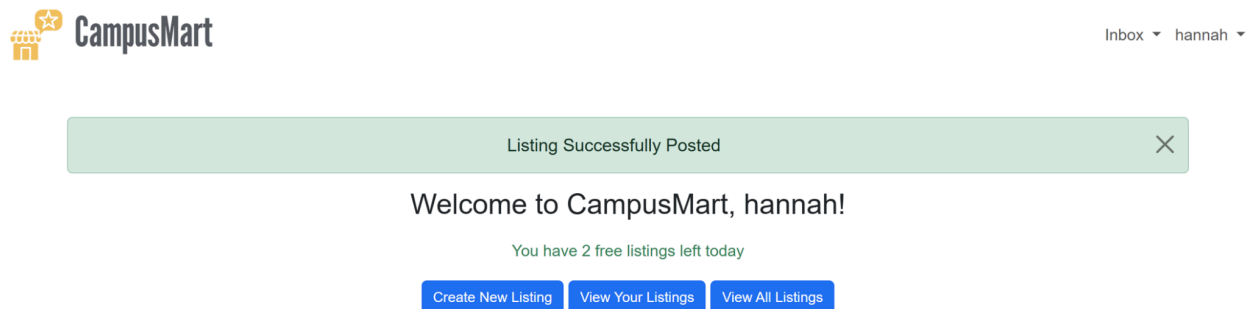


The screenshot shows the 'Create New Listing' form in the CampusMart application. The form is centered on the page and includes the following fields:

- Title:** A text input field with the placeholder 'Title'.
- Description:** A text area with the placeholder 'Description'.
- Price:** A text input field with the placeholder 'Price'.
- Condition:** A dropdown menu with the placeholder 'Select...'.
- Photos:** A section with a 'Choose Files' button and the text 'No file chosen'.

Below the form is a blue 'Post Listing' button. Above the form, the text 'You have 3 free listings left today' is displayed. The top of the page features the CampusMart logo and a user profile section with 'Inbox' and 'hannah'.

Figure 5.2 Screenshot for Feature A




The screenshot shows the 'Listing Successfully Posted' confirmation screen in the CampusMart application. The page features a green success message at the top: 'Listing Successfully Posted' with a close button (X). Below this, the text 'Welcome to CampusMart, hannah!' is displayed. Underneath, it says 'You have 2 free listings left today'. At the bottom, there are three blue buttons: 'Create New Listing', 'View Your Listings', and 'View All Listings'. The top of the page features the CampusMart logo and a user profile section with 'Inbox' and 'hannah'.

Figure 5.3 Screenshot for Feature A

Feature B: 2.2 Update Listings

Figure 6 demonstrates our update listing functionality. In the bottom left corner of a user's listing, there is a blue button that when clicked, brings up the original listing and allows the user to update any part of the listing they deem necessary. It allows the user to change the price, photo, or even availability to allow for seamless interactions.

Your Listings



Item: Waterbottle

Description: This is a water bottle I am selling!

Price: 12.00 Krato\$Coin

Status: AVAILABLE

UpdateDelete

Figure 6.1 Screenshot for Feature B

Update Listing


Title

Description

Price

Condition

Status

Current Images


Upload New Images (optional — replaces old ones)
 No file chosen

Update ListingCancel

Figure 6.2 Screenshot for Feature B

Condition

Status

Figure 6.3 Screenshot for Feature B

Feature C: 2.3 Deleting Listings

Figure 7 represents our delete listing functionality. On the user's listings, there is a red button found at the bottom right, that when clicked, gives the user a pop-up message asking to make sure they want to delete their listing. This allows the user to go back in case they have second thoughts or accidentally clicked delete. The listings page also updates and shows the user that their listing has been successfully deleted.

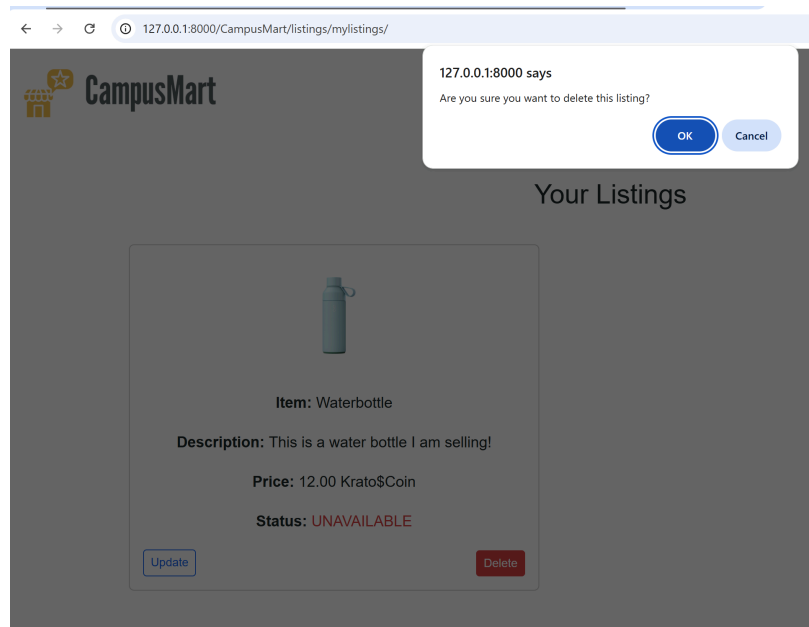


Figure 7.1 Screenshot for Feature C



Inbox ▾ hannah ▾

Your Listings

You haven't posted any listings yet.

Figure 7.2 Screenshot for Feature C

Feature 3

Feature A: 3.1 View All Listings

Figure 8 shows our All Listings page! The all listings feature can be accessed via the rightmost blue button on the home page, shown in Figure 5.3, and when clicked, brings up a list of all

postings in the CampusMart. A user can scroll through all of the listings until they hit the bottom where we included a page number so they know what page of listings they are on.

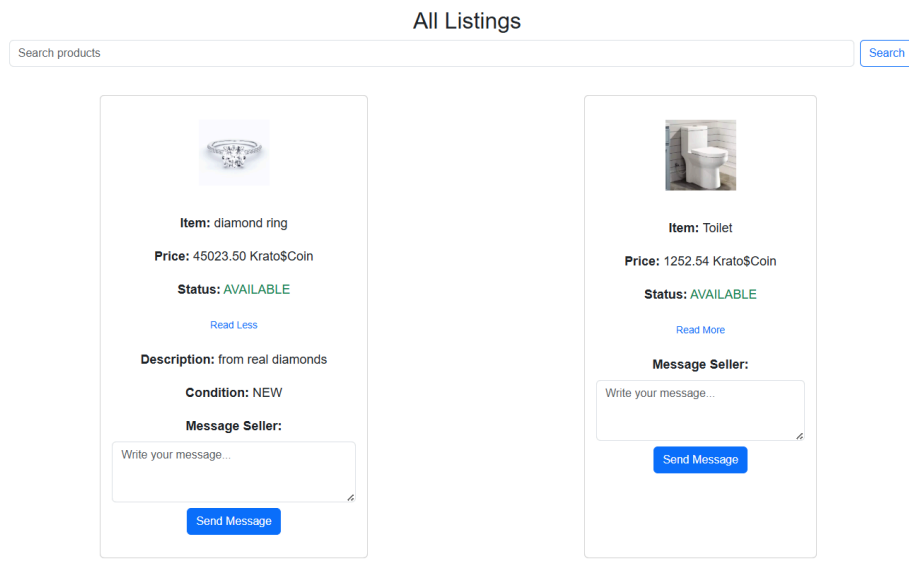


Figure 8.1 Screenshot for Feature A

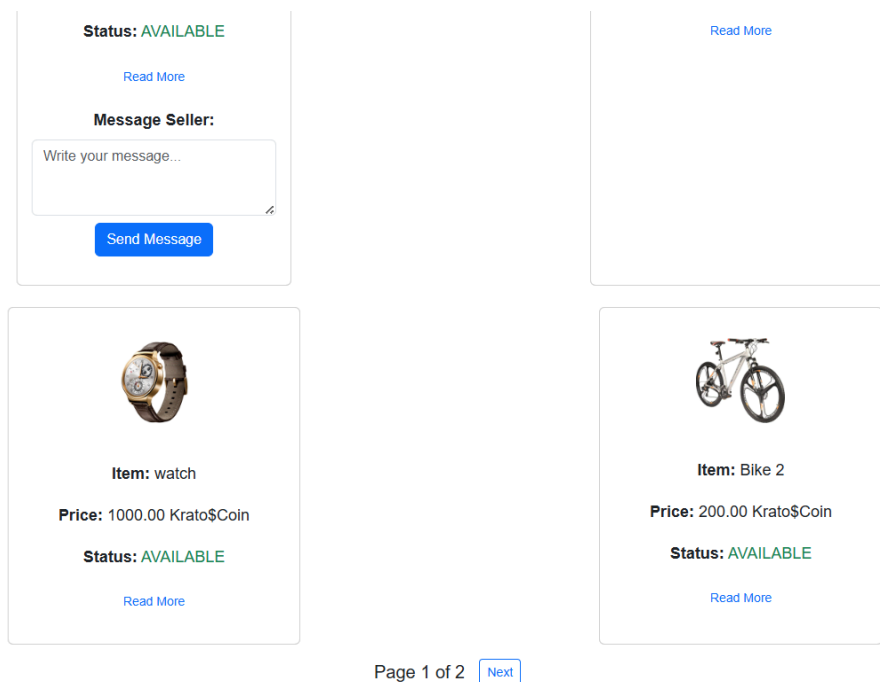


Figure 8.2 Screenshot for Feature A

Feature B: 3.2 Searching for Products

Figure 9 represents our search feature. This feature allows a user to search for a product listing via keyword on the title or description of the product. For example, Figure 9 represents a user looking for something to do with water, maybe a water bottle, and after clicking search, they are given all listings that have to do with water!

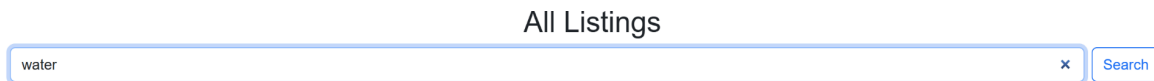


Figure 9.1 Screenshot for Feature B

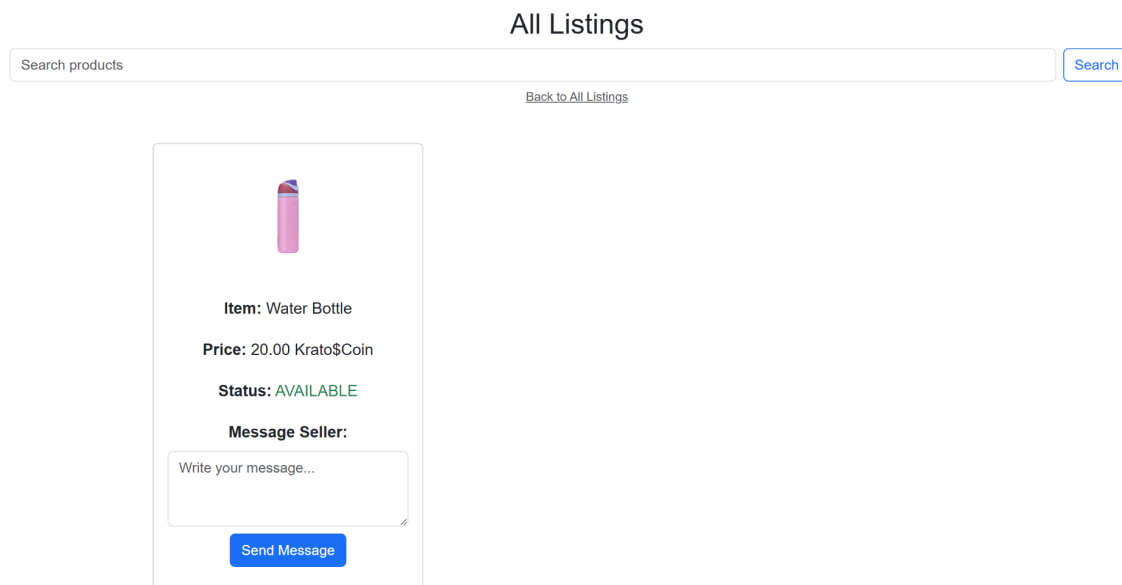


Figure 9.2 Screenshot for Feature B

Feature C: 3.3 Seller-Buyer Messaging

Figure 10 demonstrates our messaging functionality! This function allows the user and buyer to communicate with each other about a listing on CampusMart. As shown, there is a message box on each listing that is made by another user that allows the buyer to ask questions or communicate in general with the seller. We also designed a message pop-up to let the user know their message was successfully sent. We also designed an inbox button, found in the top right of the website, which when clicked, brings the user to all available messages. If a message is clicked, it opens up a chat function that allows the user to communicate with the seller, or buyer, about the listing.

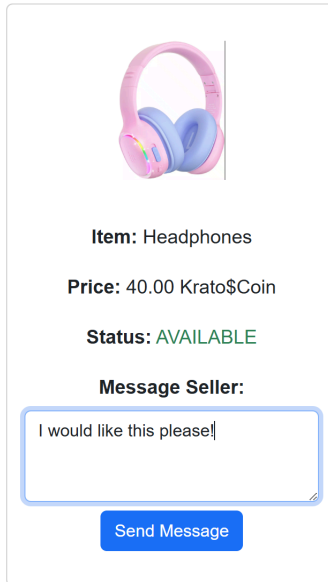


Figure 10.1 Screenshot for Feature C

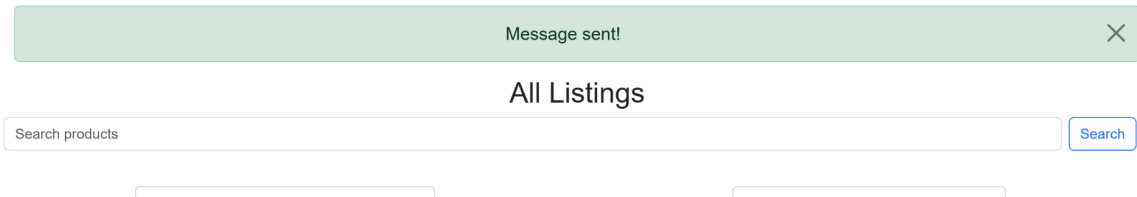


Figure 10.2 Screenshot for Feature C

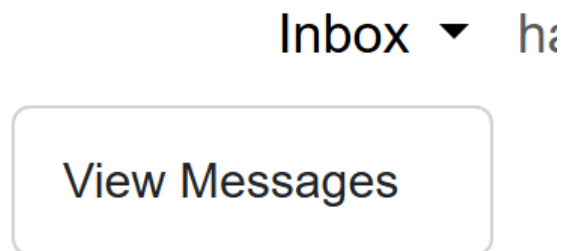


Figure 10.3 Screenshot for Feature C

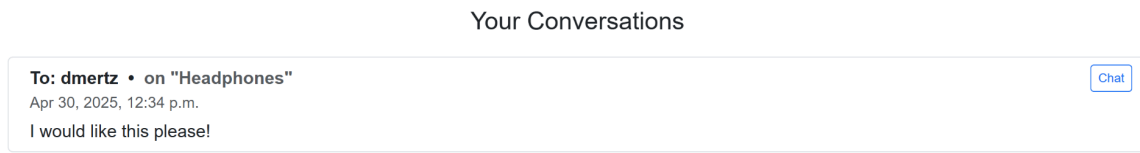


Figure 10.4 Screenshot for Feature C



Figure 10.5 Screenshot for Feature C

Feature 4

Feature A: 4.1 Buying Daily Listings

Figure 11 demonstrates our buying daily listings function. This function allows users to buy more listings, if they have the money. Once all free listings are used and a user clicks the “Create New Listing” button, they are brought to a page that lets them know they have two choices, to return home, or to buy more listings. If they choose to buy more listings, they are brought to a page where they are given the amount of coins they have, a box to enter how many listings they want to purchase, and two buttons, one allowing them to purchase listings and the other allowing them to return home.

Purchase Extra Listings

Your current balance is 25 Krato\$Coins

Number of Listings to Purchase:

[Return to Home](#)

[Buy Listings](#)

Figure 11.1 Screenshot for Feature C

Purchase Extra Listings

Your current balance is 23 Krato\$Coins

Number of Listings to Purchase:

[Return to Home](#)

[Buy Listings](#)

You successfully purchased 2 extra listings!

Figure 11.2 Screenshot for Feature C

Project's Learned Lessons

1. What programming paradigm(s) have you chosen to use and why?

We have chosen to use Object-Oriented Programming by implementing the Django models, views, and classed based views. Each model represents a distinct object with different and unique attributes and behaviors. We also implement procedural programming that is used in function-based views, where handling POST and GET requests is written imperatively.

2. **If you were to start the project from scratch now, would you make different choices? Do you think the paradigm(s) chosen helped (or not) in developing the project?**

If we were to start over, we would keep the primary paradigm choice as OOP, however, maybe structure parts of the project differently. Defining our core models like **Product**, **Message**, and **UserExtraListings** as classes made it easy to encapsulate related data and behavior, and allowed us to work with Django's ORM intuitively through model instances and method chaining. For example, we used our own custom User model instead of Django's pre-made user model which made requests to get the user information a little different and we had to work around that. We also did not implement a base.html file which would definitely have made things easier. We had to copy the navigation bar into each new template instead. Overall, while OOP clearly helped in structuring our project, some of our implementation choices like in user management led to avoidable complexity. Restructuring those aspects would make the codebase cleaner, reduce bugs, and improve maintainability while keeping the benefits of the OOP paradigm.

3. **What were the most intellectually challenging aspects of the project?**

The most intellectually challenging aspects of this project included many things. One of the most challenging aspects was integrating the REST API in Feature 4 correctly. Being able to connect our program to the REST API interface was a challenge as it required many moving parts that took a while to learn how to do. Another challenging aspect was connecting each of our interfaces together. For example, when creating a page, we would have to remember not only to create an HTML file representing the actual page, but include a new url for it, a new view function for it, and sometimes, even a new model for it. While each of these features on our own we had a lot of good practice with, making sure each part connected correctly together, and no step was missed, proved quite difficult.

4. **What aspects of your process or your group's organization had the largest positive effect on the project's outcome?**

The part of our group's process that had the largest positive impact was definitely the use of live coding as it allowed us to all sit together and code simultaneously without the issues of git pulling and pushing at the same time. Another part was working with our strengths. While we all worked together on large parts of the project, for smaller individual features, we worked on what we were individually good at.