Team Member Full Name	NetID
Hayden Slade	hslade
Ian Setia	isetia
Jack Blake	jblake4
Wesley Omerso	womerso

Persistent Storage Design

We are using SQLite database to persist our data. Our database includes the tables shown in Figure 1. It contains 4 tables. The `User` table tracks data related to a user account, such as their name, username, password, and email. The username and password are necessary to sign in. The `Listing` table tracks information about a user's listing, including the title, description, price, condition, and the primary image which will be shown in thumbnails. It also tracks data not entered by the user, such as whether the listing is still available, the date the item was posted, and the account of the user who created the listing. The `Image` table serves to track additional images attached to a listing. For each image entered beyond the required first image, a new Image entry will be created with a reference to the listing it belongs to. The `Message` table tracks all the messages between users. It is tied via two foreign keys to separate sender and receiver users. It also keeps track of the message contents that is sent as well as the date and time it was sent and if it has been read yet. The only part the user has control over is the receiver and the message contents. The sender is automatically set as the user, the date and time is set as the current time, and the read status is marked when the receiver views the message for the first time.

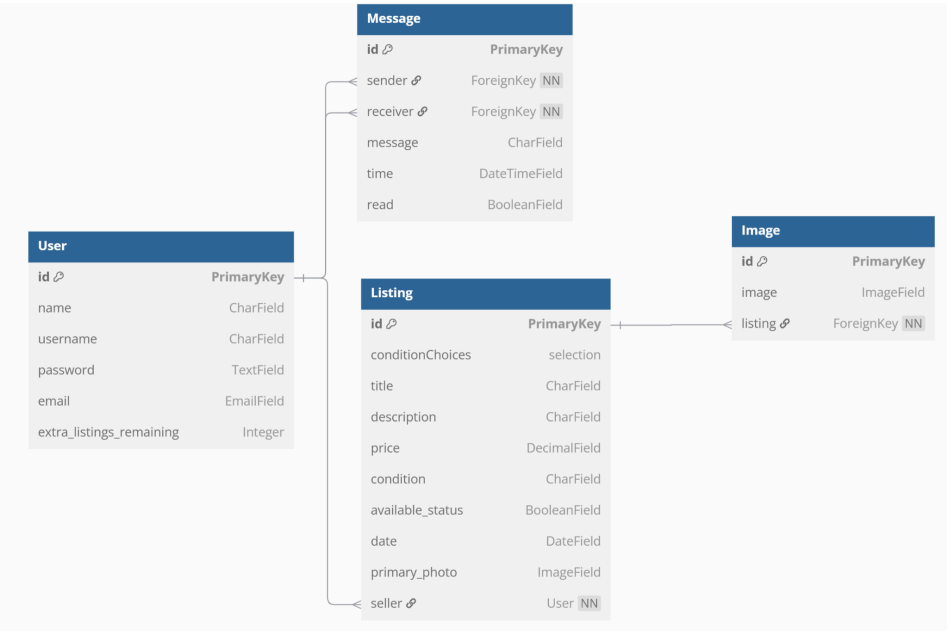


Figure 1 database schema

Demonstration of the Features

Feature 1.1

Figure 1.1.1 shows a screenshot of our user creation form, which consists of a name field, username field, email field, and 2 password fields. The error messages at the top are created after an attempt to recreate an existing user account. This shows our ability to display multiple error messages and to ensure the uniqueness of our usernames and emails. The password fields are also checked to ensure they are matched before an account may be created.

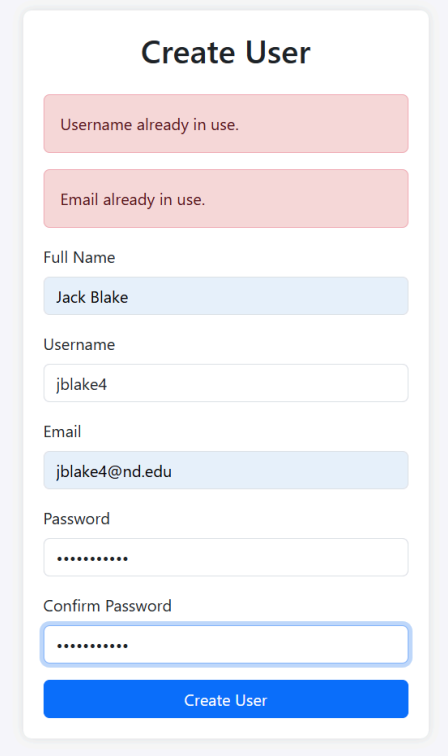


Figure 1.1.1 screenshot for feature 1.1 showing the create user field after the user attempted to put in an invalid account

Feature 1.2

Figure 1.2.1 shows a screenshot of our login page, which consists of a username field and a password field. The error message at the top is created after an unsuccessful login attempt, showing that attempts are properly authenticated.

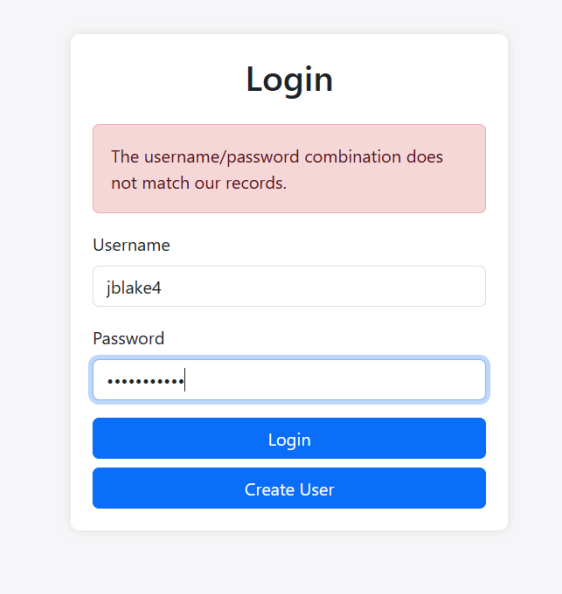


Figure 1.2.1 Screenshot for feature 1.2 showing the login screen after an unsuccessful login attempt

Feature 1.3

Figure 1.3.1 shows the location of the logout button when a user is signed in, and Figure 1.3.2 shows the resulting deletion of the user session after the logout button has been clicked.

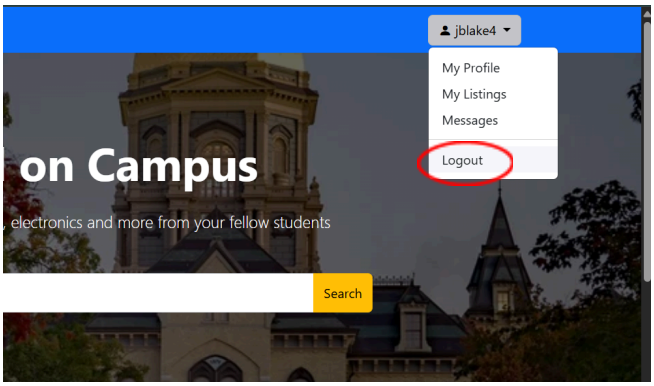


Figure 1.3.1 screenshot for feature 1.3, showing the logout button

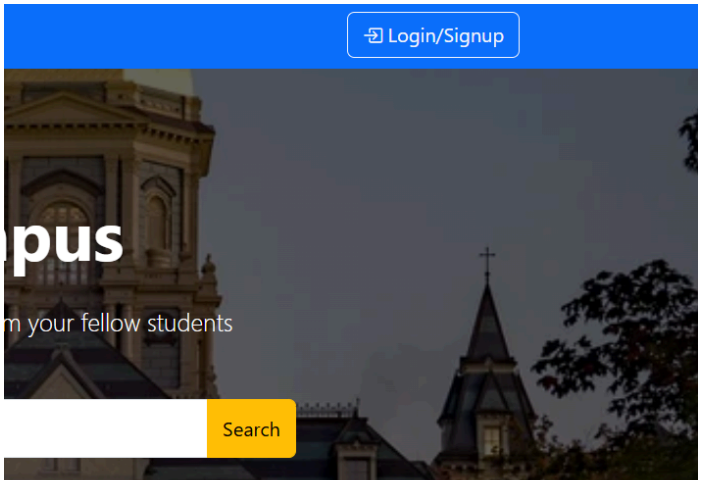


Figure 1.3.2 screenshot for feature 1.3, showing the site after clicking the logout button

Feature 2.1

Figure 2.1.1 shows all the clickable parts of the home page that would redirect the user to create a listing. Of course, the user must be logged into an existing account before being redirected to the interface shown in Figure 2.1.2. Otherwise, the site will redirect the user to the login page. In order to create a listing the seller must input a title, description, price, condition, and image. The user can click “add another image” to reveal additional image fields, but only the first one is required. If any of these are missing, the user will be given a warning and will not be able to post the listing until the conflicts have been resolved. Figure 2.1.2 displays such an example of when price has not been defined. The condition of the item is defaulted to “New” but can be changed by selecting the appropriate condition from the dropdown. In the top-right of Figure 2.1.2, the number of listings allocated to the seller is shown (defaults to 3 per day, but more can be purchased). Figure 2.1.3 displays what is shown to the seller once the listing is posted successfully and shows the updated listing count for the day.

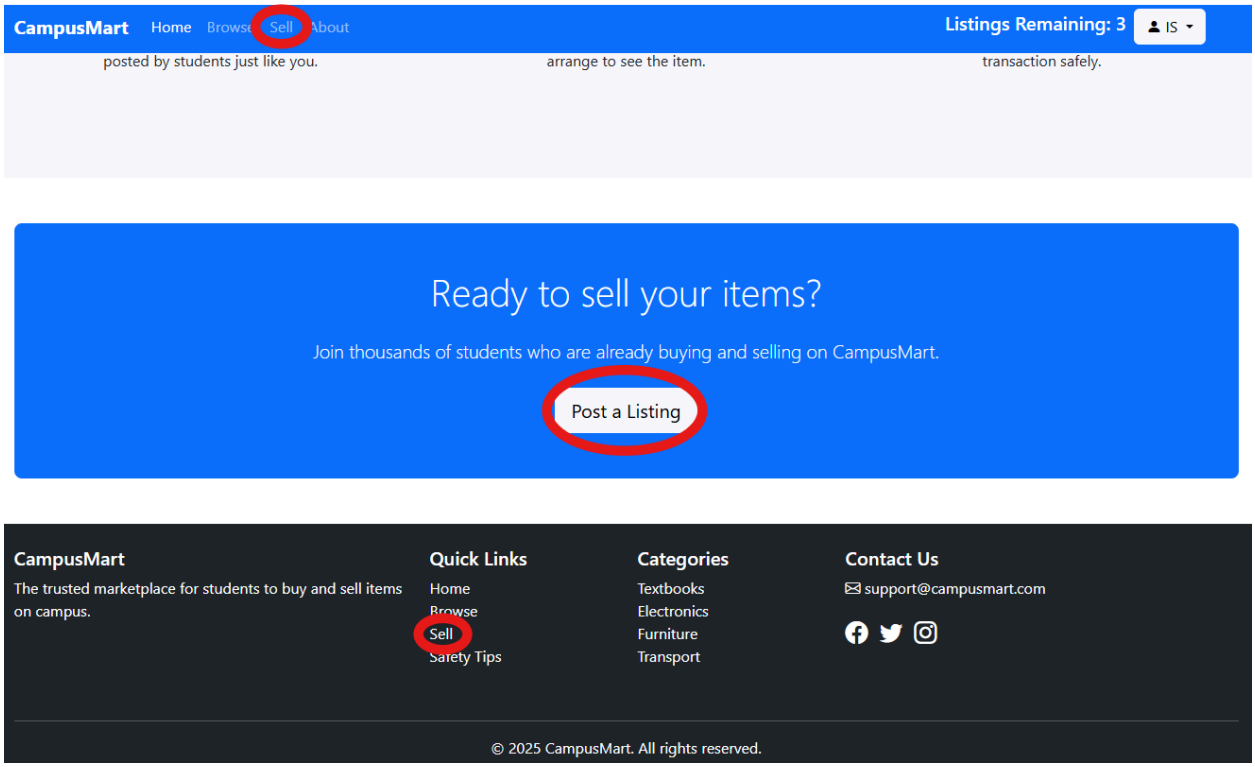


Figure 2.1.1. Bottom of homepage displaying clickables to create a listing

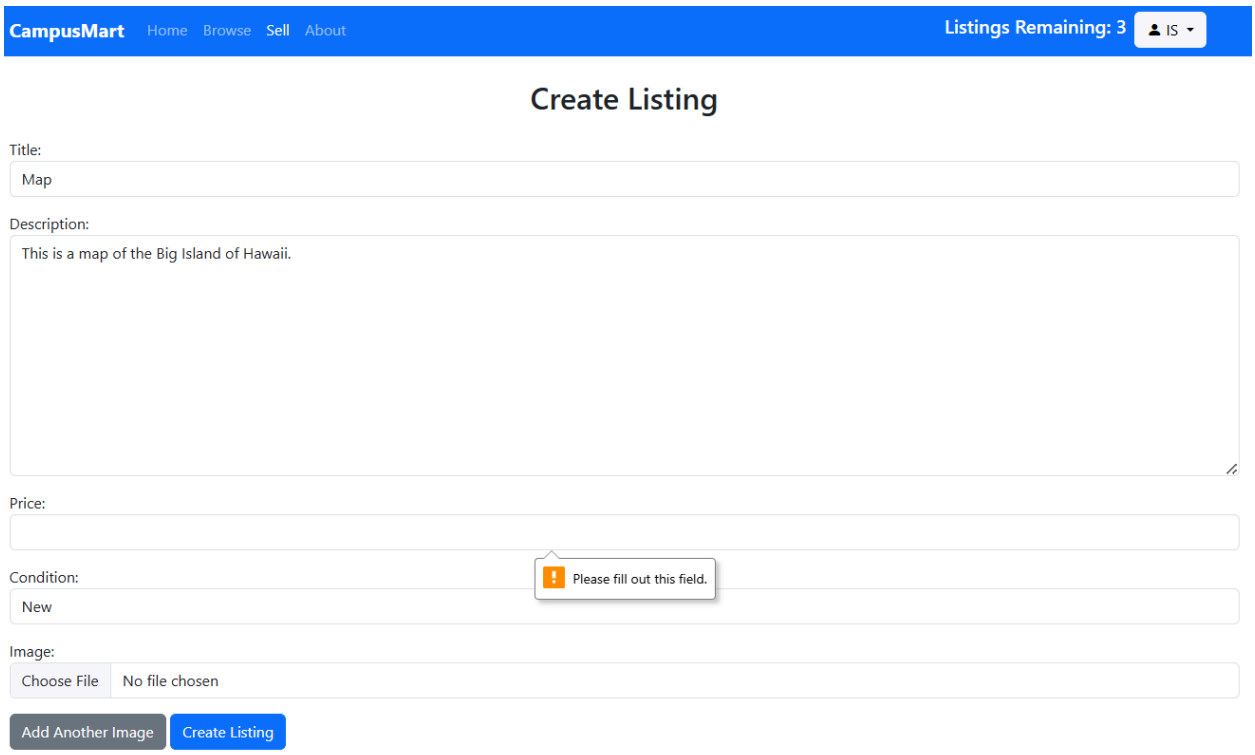


Figure 2.1.2. Create listing interface with error displayed

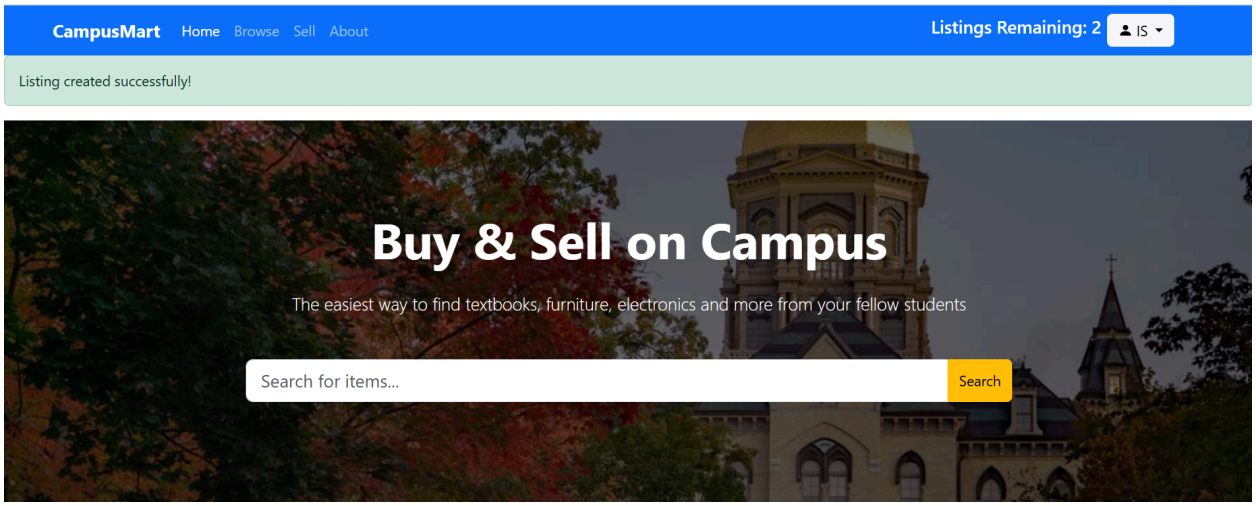


Figure 2.1.3. Successful listing created

Feature 2.2

Figure 2.2.1 shows the "My Listings" page, which displays all listings owned by the user currently signed in. Each card has options for deleting and editing the listing. When the "Edit Item" button is clicked, the user is redirected to the form in figure 2.2.2, with the details of their item auto-populating the fields. The availability field is also displayed in addition to the other fields shared with create listing. The form shares the same error checking as create listing, preventing the user from entering incomplete or incorrectly formatted data. When the user submits the form, a confirmation message appears, shown in figure 2.2.3. The changes made in the edit form persist, as shown in 2.2.4, where the chair item now has the new data that was entered in figure 2.2.2.

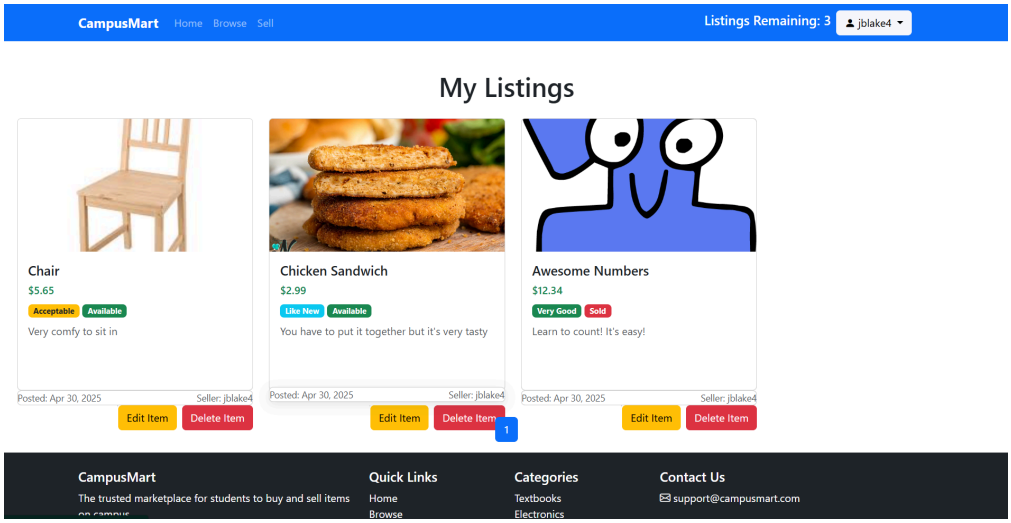


Figure 2.2.1. View user listings

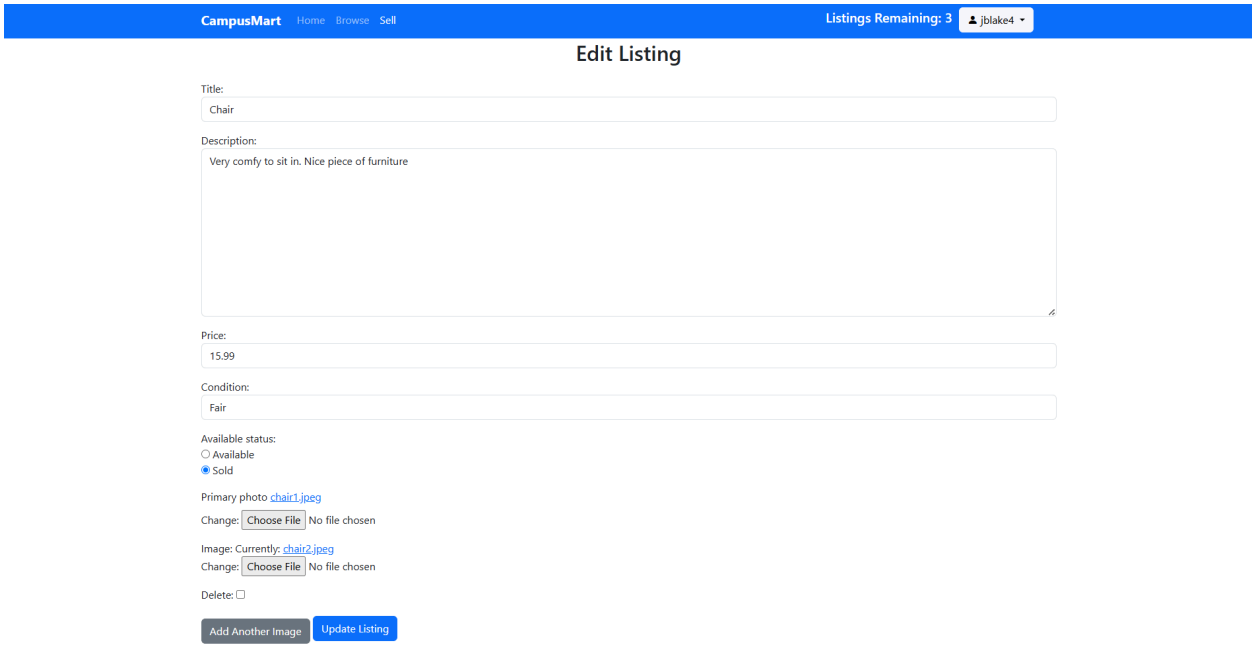


Figure 2.2.2. Edit listing form

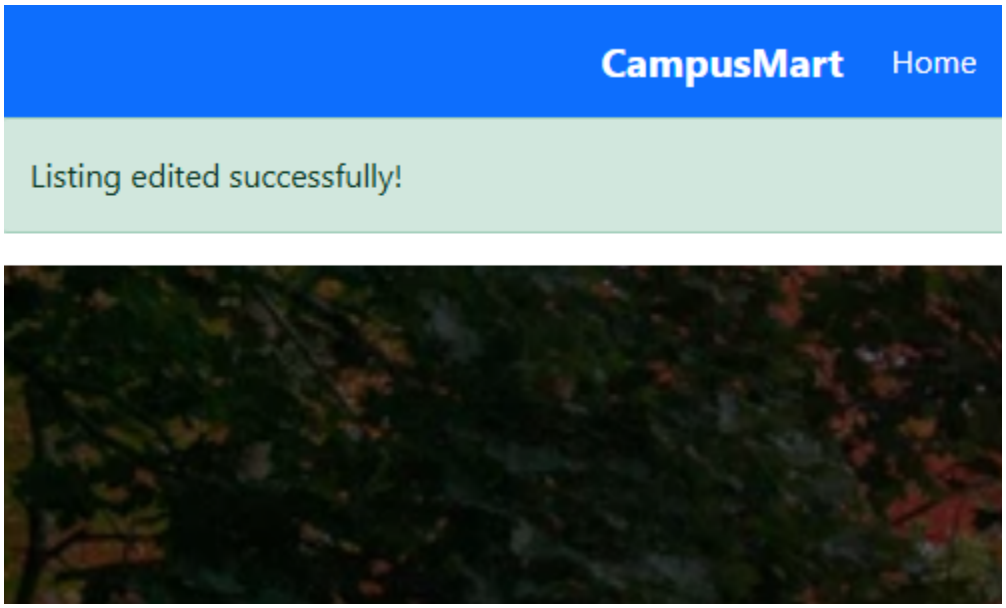


Figure 2.2.3. Edit success message

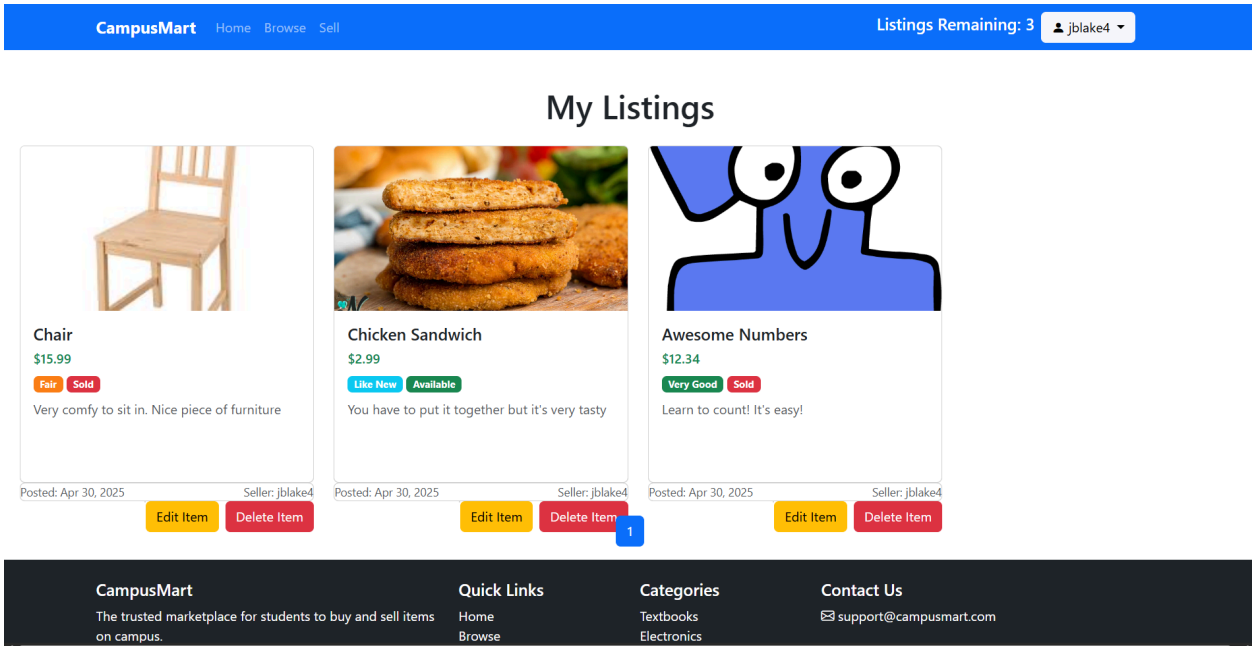


Figure 2.2.4. Results from edit

Feature 2.3

Figure 2.3.1 shows the "My Listings" page, which displays all listings owned by the user currently signed in. Each card has options for deleting and editing the listing. When the "Delete Item" button is clicked, the user is redirected to the home page and a confirmation message appears, as in figure 2.3.2. When the "My Listings" page is revisited in 2.3.3, the deleted listing has disappeared. This shows that the deletion persists after the button is clicked.

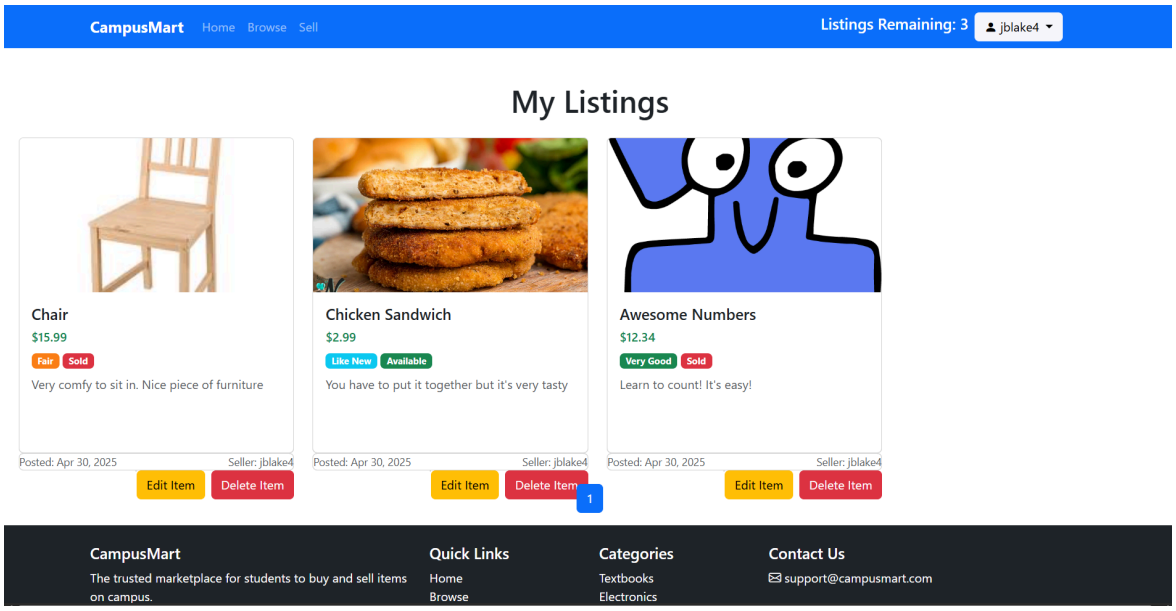


Figure 2.3.1. View user listings

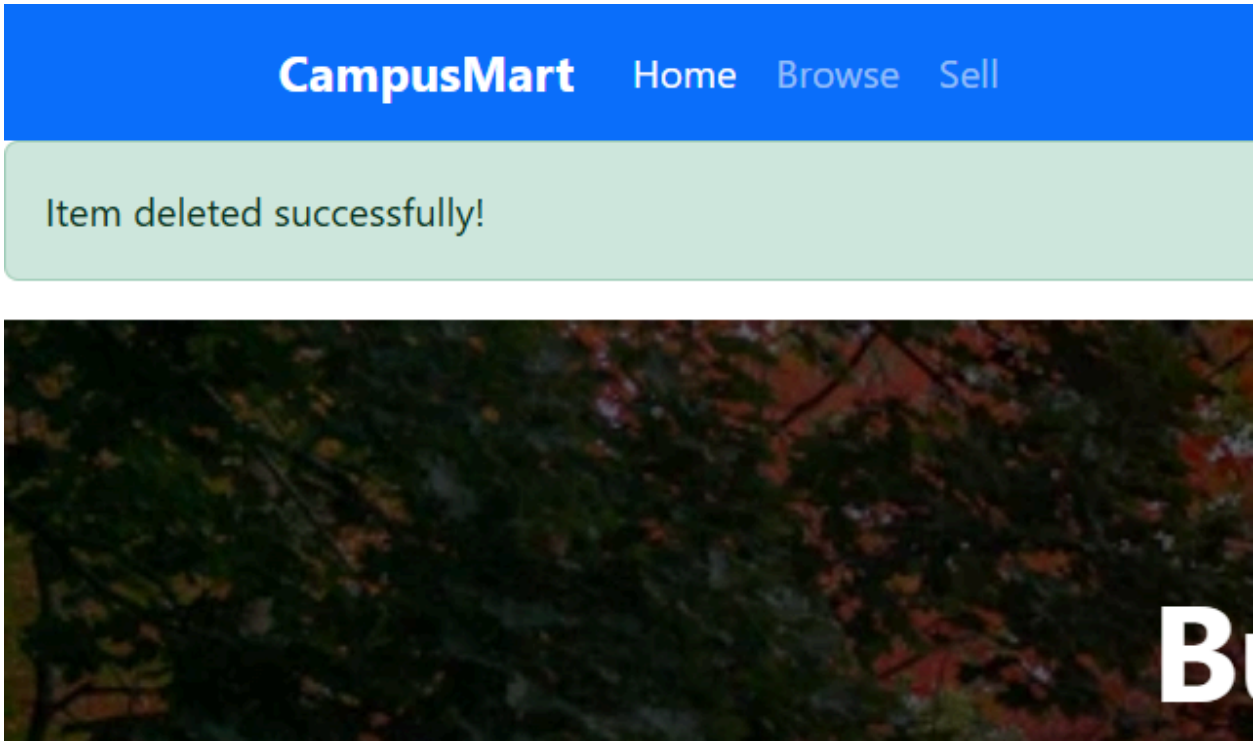
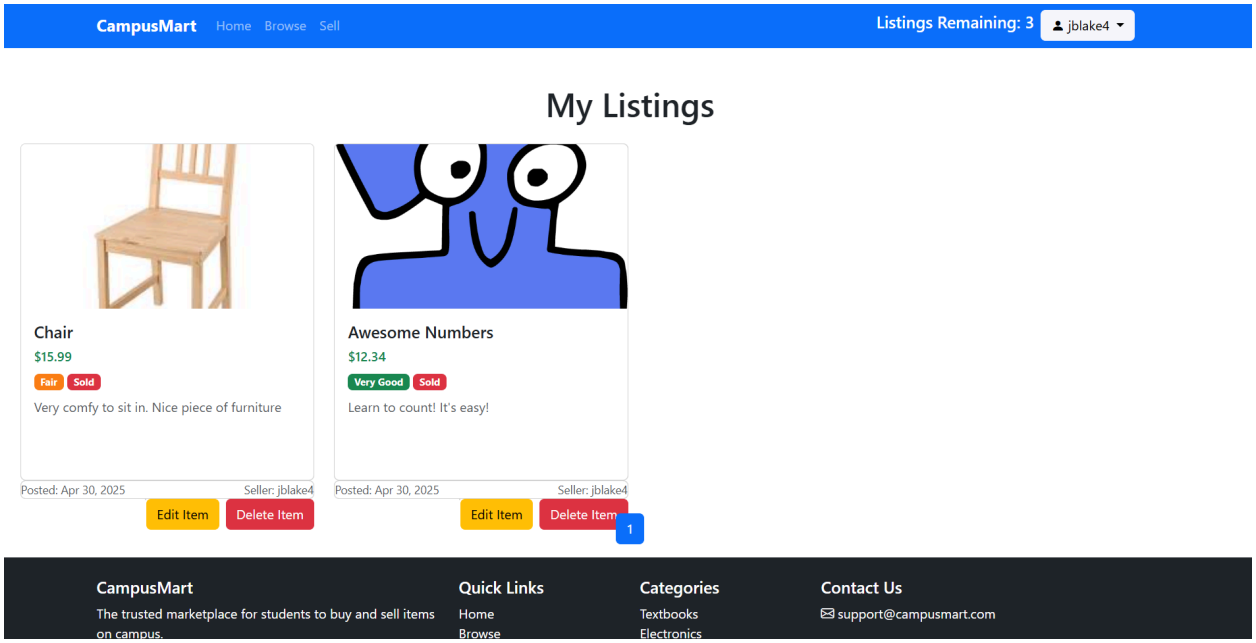


Figure 2.3.2. Delete success message



2.3.3. Results from delete

Feature 3.1

Figure 3.1.1 shows how listings are displayed to a user when they choose to browse through all listings available. There are a maximum of 20 listings per page. The current page is displayed at the bottom as shown in Figure 3.1.2. When the number of total listings exceeds 20, the page

number shown in Figure 3.1.2 will have directional arrows added to the left or right as shown in Figure 3.1.3. Hitting the button with “>>” will move the user to the subsequent page whereas the “>> >>” button will move the user to the last page. There are also “<<” and “<< <<” buttons as well, once you move past the first page as shown in Figure 3.1.4 which redirect the user to the previous page and the first page respectively.

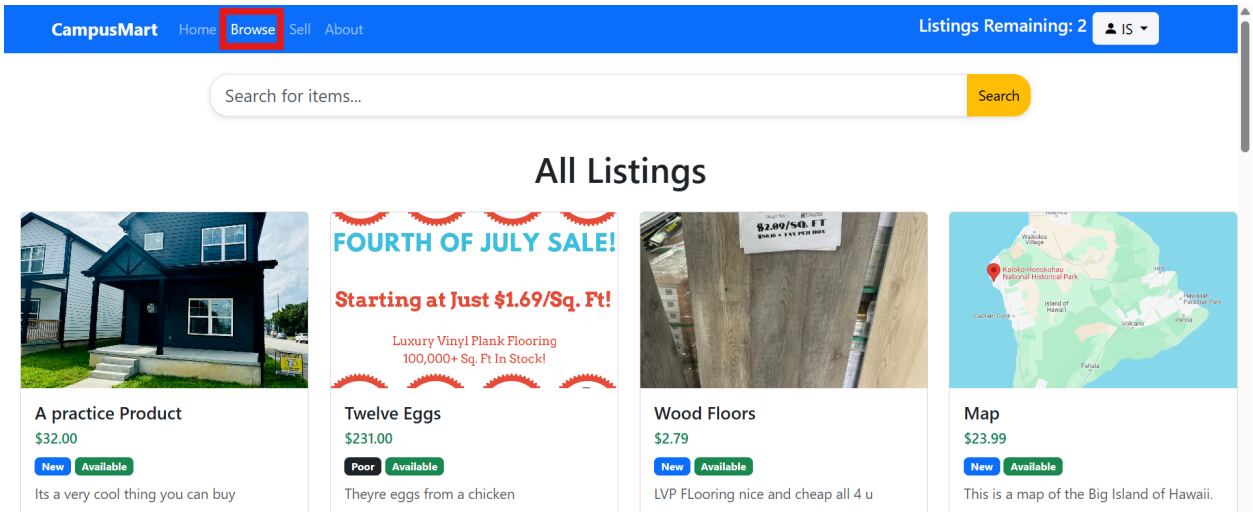


Figure 3.1.1. Grid view of all listings after selecting “Browse”

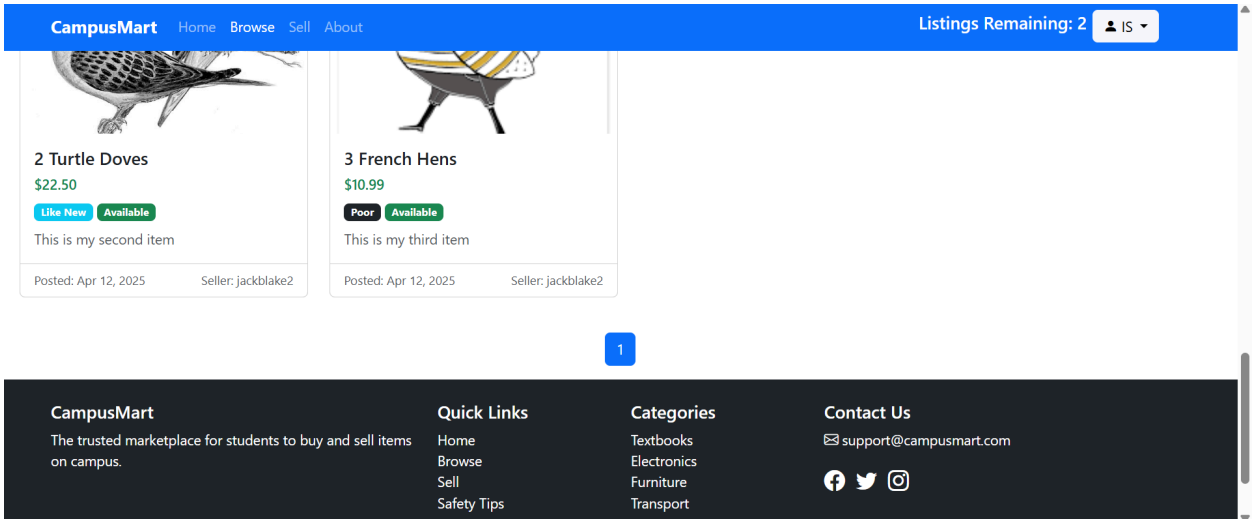


Figure 3.1.2. Bottom of page shown in Figure 3.1.1 with current page number

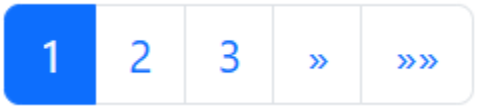


Figure 3.1.3. Pagination arrows on page 1

Figure 3.1.4. Pagination arrows on page 3

Feature 3.2

Figures 3.1.1 and 2.1.3 display where the search bars can be found on the "Browse" and "Home" pages respectively. The user can use this search bar to search for items based on their title or description. Figure 3.2.1 shows how the search results are displayed and how the user can continue to use the search bar after having searched for a keyword. The UI is very similar to the "Browse" page with the major difference being the displayed listings must have a matching keyword to the search term. (Searching a blank string "" would produce the same results as the "Browse" page)

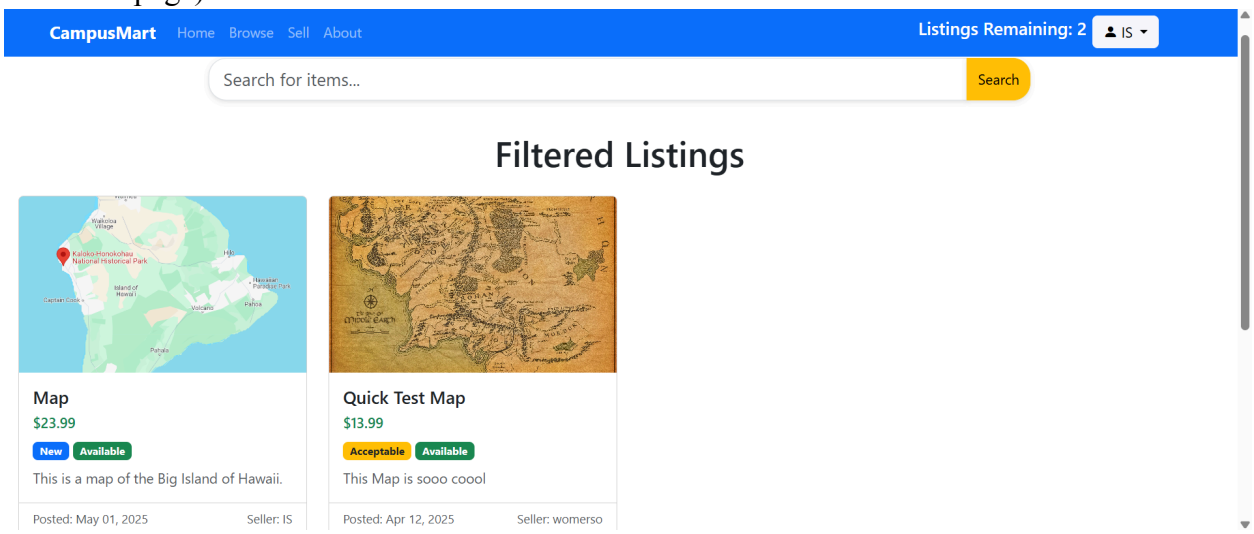


Figure 3.2.1. All listings after searching the keyword "map"

Feature 3.3

Figure 3.3.1 shows the "Chat with Seller" button that can be clicked to open the message style interface to communicate with the seller. This is the only method available to create a new chat with another user. Clicking the "Chat with Seller" button will redirect the user to a page with the chat interface shown in Figure 3.3.2 where the user is able to type a message and send it to the seller. The message history is shown directly above the textbox where messages can be written. Messages sent by the user are colored blue whereas messages sent by the other user (seller) are in grey. The button circled in Figure 3.3.3 shows where the user can navigate to in order to view all messages they have received from other users. Upon clicking "Messages" from the dropdown the user is redirected to the interface shown in Figure 3.3.3 where all chatrooms are displayed with the latest message. If a chat room has unread messages for the user, it will show a "New" badge next to the chat. Clicking into the chatroom will redirect the user to the interface shown in figure 3.3.2 to respond to the other user.

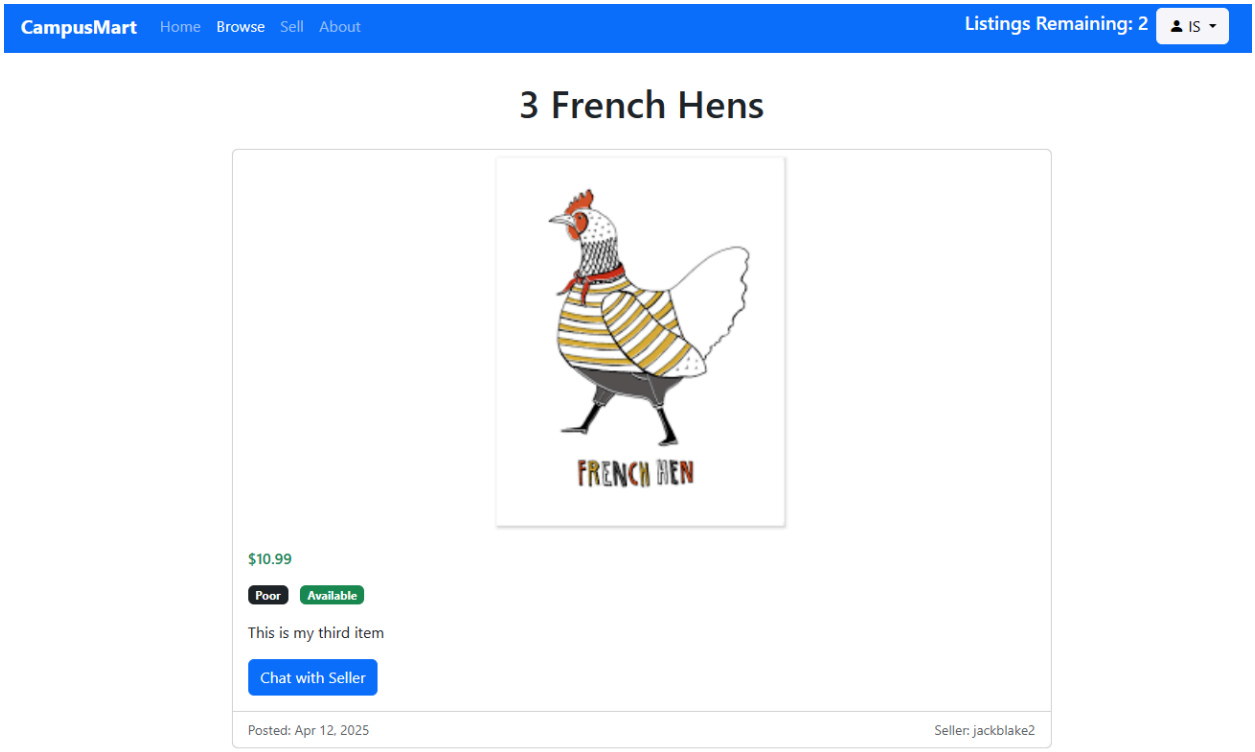


Figure 3.3.1. Detailed view of listing showing "Chat with Seller" button

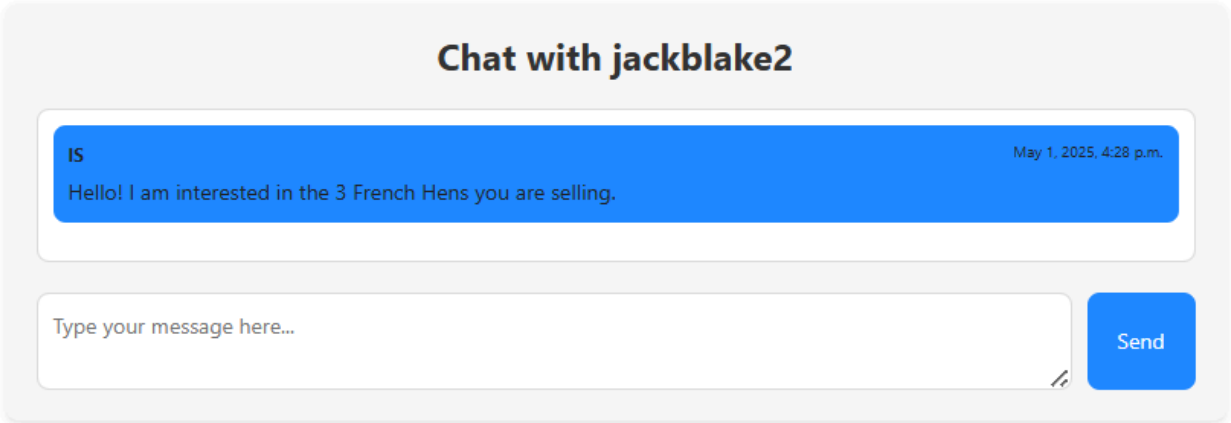


Figure 3.3.2 Chat interface

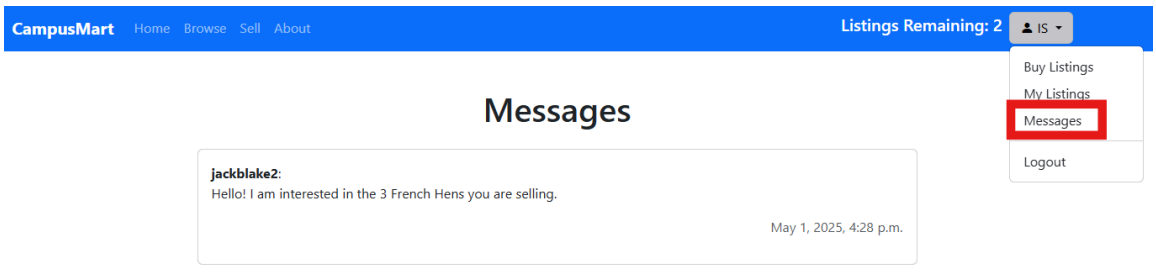


Figure 3.3.3. Viewing all messages page

Feature 4.1

Figure 4.1.1 demonstrates the landing page for buying more listings with Krato$Coin. This includes a form to purchase any amount of listings for Krato$Coin - at an exchange of 1:1. The form validates input, limited to number input and the transaction only goes through if there is enough Krato$Coin in the user's account to complete the purchase (Figure 4.1.2), otherwise the user is alerted. (See Figure 4.1.3) The user can access the page to buy more listings from the user drop down menu (Figure 4.1.4) or from the paywall that activates when the user tries to post a listing with none remaining (Figure 4.1.5). Upon the purchase of an additional listing, the count is updated (Figure 4.1.6) and does not expire/refresh unlike the free three listings each day.

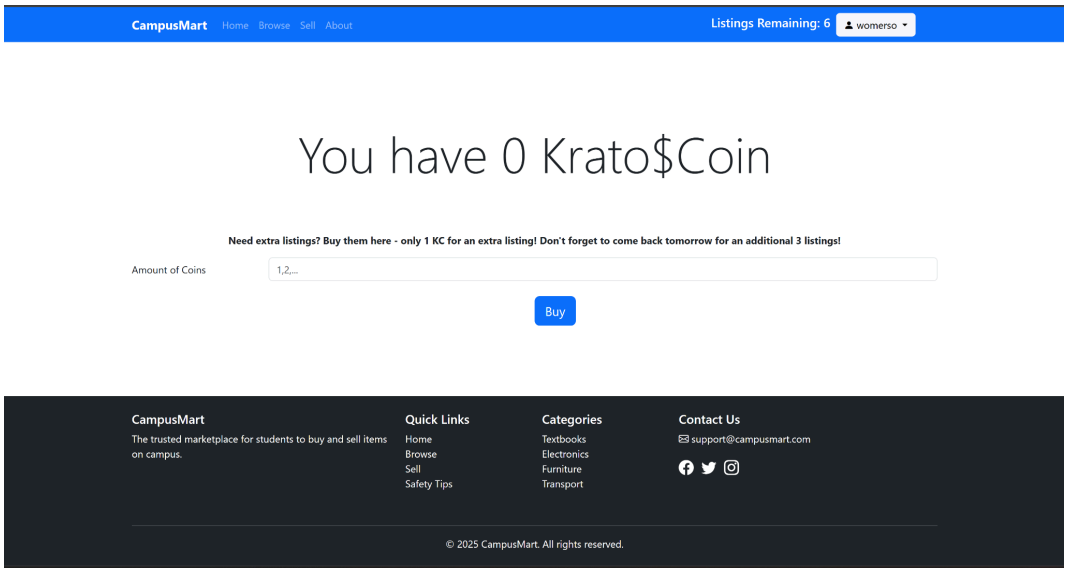


Figure 4.1.1. Microtransaction landing Page

Figure 4.1.2 - Successful purchase

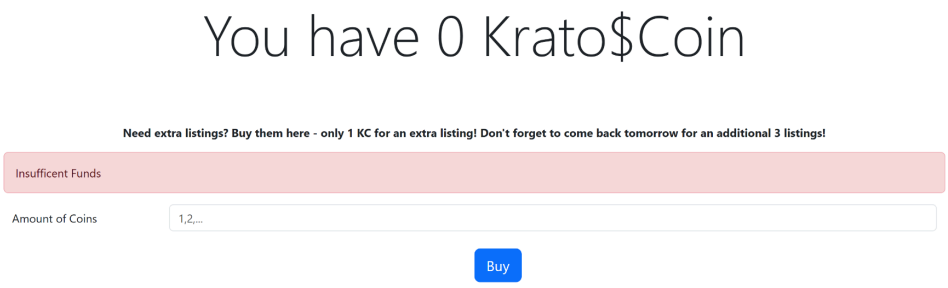


Figure 4.1.3. Alert for incomplete purchase

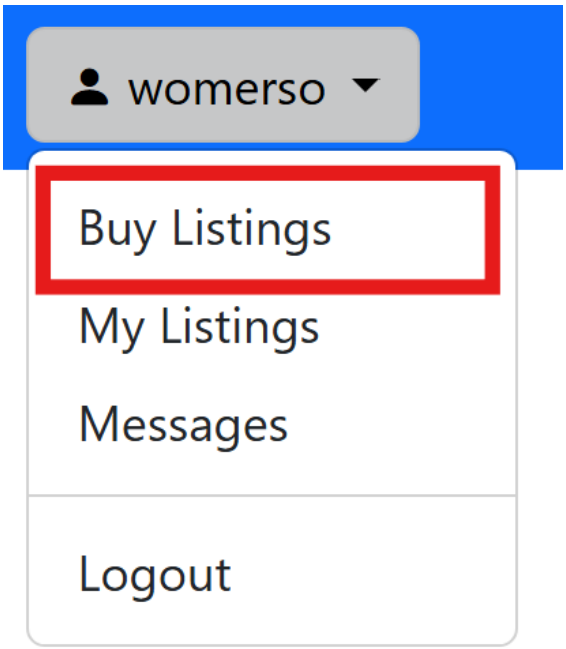


Figure 4.4 - Access Extra Listings

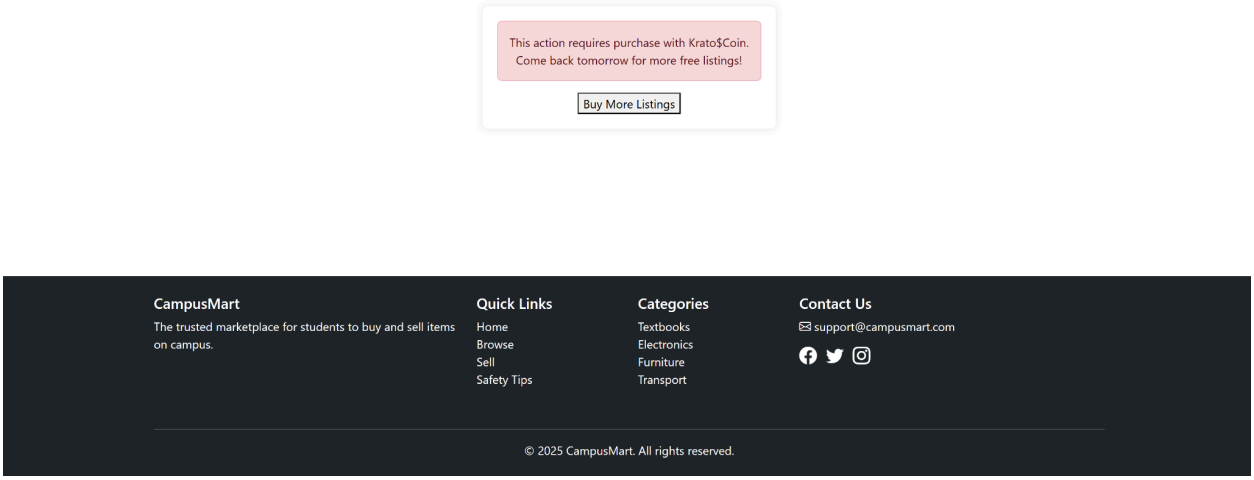


Figure 4.1.5. Paywall Page

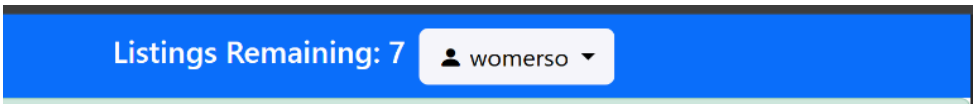


Figure 4.1.6. Updated Count

Project's Learned Lessons

In this section, you will reflect on the group's activities and performance through the entire project, and capture your reflections, observations and thoughts. It should address the following items, but feel free to expand on these in light of your group's unique experiences.

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

In this project we primarily used the Object-Oriented Programming paradigm, which aligns naturally with Django's architecture. Django's use of models, views, and forms encourages an OOP approach, which allows us to keep the code relatively modular, reusable, and easy to test and maintain. In addition to OOP, we have used some declarative style programming, which helped streamline development as we were able to describe the behavior that we wanted our site to have rather than imperatively implementing every aspect. There was also some procedural code in utility scripts and view functions where appropriate, but the core of the application benefited from Django's object-oriented structure. If we were to start the project from scratch, we might lean even more on class-based views and explore more functional programming patterns in views and data pipelines in order to create greater flexibility in the program. Overall, the OOP paradigm and Django's framework definitely helped in organizing and developing the project efficiently. The abstraction provided by Django's ORM makes it easier to focus on the main logic without getting bogged down by the small details.

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

A difficult aspect of the project was properly handling images. This required setting up a media root in the settings.py folder and directing our forms to save images there, since the image files are too large to store directly in the sqlite database. It was also difficult to allow our forms to accept a variable number of images. This was accomplished by adding an additional table in our database called Images, which we used to save all optional images and associate them with the parent listing. To allow the user flexibility, a Javascript function was created that adds a new form to a formset of optional image fields each time it is clicked, allowing the user to create as many image fields as they want.

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

One of the key factors that positively impacted our project's outcome was our communication through Slack. We maintained open and consistent communication at all times, which allowed us to quickly address questions, share updates, and coordinate next steps. This constant connection helped prevent misunderstandings and ensured everyone was on the same page throughout the project. Additionally, our effective distribution of project parts played a major role in our success. By clearly dividing responsibilities across the features and sub-features, we were able to work more efficiently and make steady progress. This organization allowed us to tackle different components simultaneously, meet deadlines, and easily integrate our work at each stage.