Mini-Amazon: CompSci 316 Final Project

Jenny Huang Miguel Garzon Allison Li Jesse Prakken Caleb Woo

November 17, 2020

1 Introduction

E-commerce has continued to become more and more prevalent in our daily lives, and with the pandemic, it has become an indispensable way for people to shop within the safety of their homes. As all of us are avid users of Amazon and other e-commerce sites, we were excited for the opportunity to create our own e-commerce site. Creating our own e-commerce site was an opportunity for us to work with and apply the backend database knowledge we developed in class while implementing an appealing frontend using some inspiration from our favorite e-commerce sites. Being familiar with mainstream e-commerce sites certainly helped us design and implement key site features including profile, cart, and search. However, as Amazon's level of production is impossibly ambitious for a one semester project, we also had to find simple, creative, and effective ways to implement features that are typically more complex on professional e-commerce sites such as reviews, balance, and images. With only 10,000 items and 2,660 users, our website is nowhere near the capabilities of standard e-commerce sites. Nonetheless, managing and working with all this data was an exciting challenge for all of us that taught us about some of the many challenges full scale e-commerce sites are tasked with.

2 Division of Work

The Mini-Amazon site provides five major functionalities/parts. Each group member was assigned one of the parts and covered the database, back-end, and front-end for their assigned part. The table below provides a detailed breakdown of the division of work.

Name	**Design (E/R)**	**Design and Load Data**	**Frontend and Backend**
Jenny Huang	Buyer + Profile	Buyer + Profile	Login, register, forget password, profile
Jesse Prakken	Seller + Sells	Seller + Sells, Load/clean data	Add/modify item, selling list
Miguel Garzon	Item + Category	Item + Category	Search, search result, item page, category list
Allison Li	Cart + History	Cart + History	Cart, check out, add balance, sales history, order history
Caleb Woo	Review + Recommendation	Review + Recommendation, Load/clean data	Add review (seller and item), edit review, show average, home page–Show the recommended items when the user logs in

Load/clean data refers to finding, cleaning, and importing a production dataset to the database.

3 Design

The design of the Mini-Amazon site is specified as an E/R Diagram in Figure 1. This diagram is roughly the same as the original version that we created. There were slight changes, such as the images being moved from the item entity set to the category entity set. Users have new attributes security question and answer to allow for our implementation of forget password.

The E/R Diagram is translated into a SQL database design. The design has 8 main entity sets. Seller is a subclass of User, since a Seller has additional actions and behaviors besides a basic User. When translating to the database schema, this Seller-User subclass relationship is defined with the Entity in all Superclasses approach. The entire database schema is written out below.

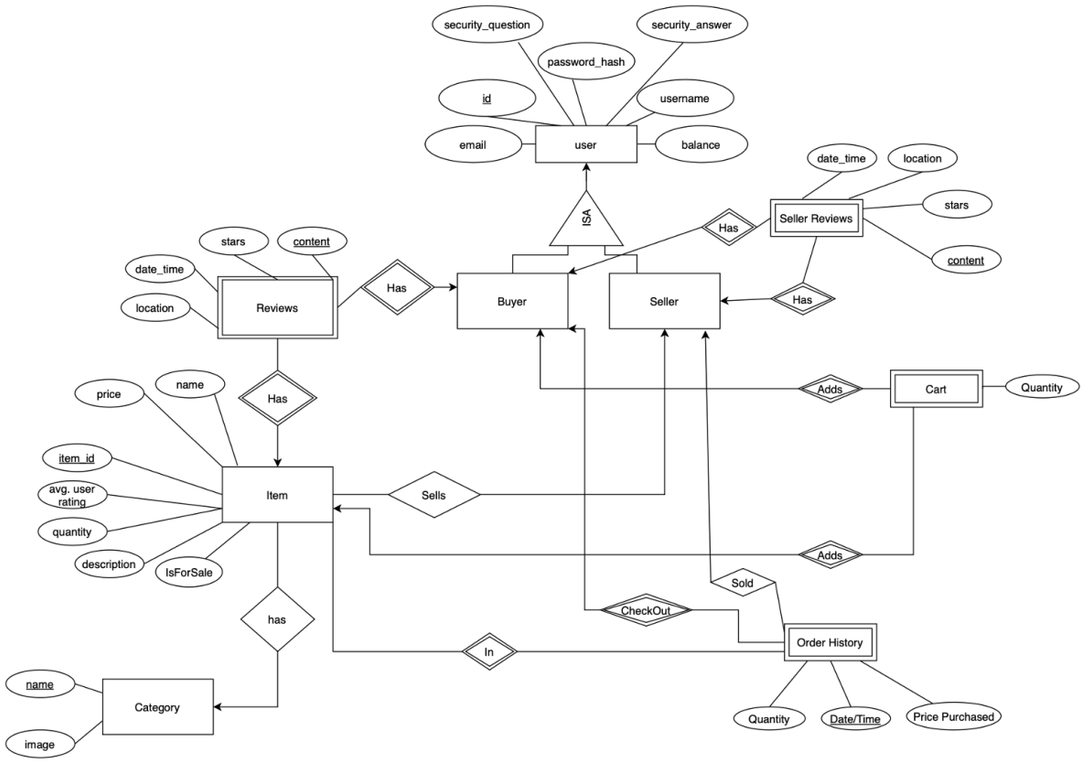

Figure 1: E/R Diagram

User (
    id INTEGER PRIMARY KEY,
    username VARCHAR(20) UNIQUE,
    password_hash VARCHAR(128) NOT NULL,
    email VARCHAR(120) UNIQUE NOT NULL,
    balance FLOAT NOT NULL,
    security_question VARCHAR(250) NOT NULL,
    security_answer_hash VARCHAR(250) NOT NULL)

Seller (
    id VARCHAR(15) PRIMARY KEY)

Item (
    id INTEGER PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    price FLOAT NOT NULL,
    quantity INTEGER NOT NULL,
    avg_user_rating FLOAT,
    description VARCHAR(300),
    category VARCHAR(45) REFERENCES Category(name),
    seller_id INTEGER NOT NULL REFERENCES Seller(id),
    if_for_sale BIT)

Category (
    name VARCHAR(45) PRIMARY KEY)

Cart (
    buyer_id INTEGER NOT NULL REFERENCES User(id),
    item_id INTEGER NOT NULL REFERENCES Item(id),
    quantity INTEGER NOT NULL,
    PRIMARY KEY (buyer_id, item_id))

OrderHistory (
    buyer_id INTEGER NOT NULL REFERENCES User(id),
    seller_id INTEGER NOT NULL REFERENCES Seller(id),
    item_id INTEGER NOT NULL REFERENCES Item(id),
    datetime DATETIME ,
    quantity INTEGER NOT NULL,
    price_sold FLOAT NOT NULL,
    PRIMARY KEY(buyer_id, item_id, datetime))

```
Reviews (
    user_id INTEGER NOT NULL REFERENCES User(id),
    item_id INTEGER NOT NULL REFERENCES Item(id),
    datetime VARCHAR(10) NOT NULL,
    location VARCHAR(120),
    stars INTEGER NOT NULL,
    content VARCHAR(300),
    PRIMARY KEY (user_id, item_id, content))

Seller Reviews (
    user_id INTEGER NOT NULL REFERENCES User(id),
    seller_id INTEGER NOT NULL REFERENCES Seller(id),
    datetime VARCHAR(10) NOT NULL,
    location VARCHAR(120),
    stars INTEGER NOT NULL,
    content VARCHAR(300),
    PRIMARY KEY (user_id, seller_id, content))
```

4 Implementation

4.1 Overview

We created our website using Flask, a micro web framework in Python. We created a Flask directory in a GitHub repository and worked off individual branches to avoid merge conflicts. The backend code is primarily written in Python using Flask's WTForms, SQLAlchemy, and other Flask extensions while the frontend code is primarily written in HTML with some CSS, Bootstrap and JavaScript for frontend styling. The database management system we utilized is SQLite.

4.2 Login + Registration

A user can register from the login page. When registering, a user must enter a username, email, password, repeat the password, a security question, and a security answer. If the username is already taken, there is a message "[Please use a different username.]". If the email is not valid or already registered, there is a message "[Invalid email address.]". If the repeat password field is not equal to the password, "[Field must be equal to password.]" will appear. Users are able to write their own security questions and answers. We allow users to write their own security questions because of personal preference. This is not how most websites work, but we find having a dropdown menu of select security questions to choose from to be restrictive. We want our users to have more freedom with this. There is also an option to register as a seller via a checkbox. If this box is not checked, then the user will be registered as strictly a buyer.

One thing to note about how we create users is that we hash the password and security answers so that they are not actually stored in the database. This way, we feel the website is more secure and protects the user's information.

To login, a user must enter their username and password. If the username does not belong to an existing account or the password is incorrect, "Invalid username or password" will appear. There is also a "Remember me" box that keeps the user logged in. On the bottom of the login page, there is a link for registering as a new user (elaborated above) and also a link for forget password. The forget password link leads you to a page where you enter your username. It says "Input username to get security question". If the username inputted does not belong to an existing account, "User does not exist" will appear. When a valid username is submitted, the user is brought to another page, titled "Answer security question". On this page, the security question of the user is displayed and they are able to submit their answer. If the answer is incorrect, "Wrong answer, try again." is displayed. If the right answer is submitted, the user is brought to a page where they can reset their password. They can simply enter a new password and then repeat the password. If the two inputs are equal, then the submission redirects back to the login page. We decided to allow users to set their password to be the same as their old password. In our implementation, the user is out of luck if they do not remember their security answer, but this is certainly something we could improve on in our site, where we could implement some kind of email validation as a part of the "forget password" option.

4.3 Profile

The profile page simply displays a user's username, email, and balance. This is useful if they wish to check such information. There are also hyperlinks to the add balance page, and the order history page. If the user is a seller, there is also a hyperlink to their seller reviews page.

4.4 Home

The home page has a message "Hi, current_user.username!" at the top of the page. Below is the Recommended Item List, which is made up of the 5 categories in our site with the most amount of items. Then below each category are 3 items. The 3 items are the 3 items in that category with the highest average stars rating from their item reviews. Each item is displayed as an item link that is the item's name, the item's picture, and the item's average stars rating. Clicking on any item link will take the user to that item's

specific page. If a user visits our home page and is not logged in, the site will automatically redirect them to the login page.

4.5 Items + Search

An item page displays all relevant attributes of the item, like price, quantity, and seller. The page also contains item reviews, the option to add a review and add to cart. There is also a link to the reviews page of the seller, to allow the user a convenient way to leave a review for the seller of the item they are viewing.

In our implementation, we assigned a one-to-one relationship between category and image (instead of one-to-one item to image) due to API key restraints. This is further explained in the Challenges section, but in essence, this means that items of the same category will have the same image.

Item pages can be reached through search. Our barebones search page has a text field that allows a user to input a string, and returns a table containing all items that have a name containing that substring. This table has a little search bar to its upper right that allows a user to use a substring to refine their search even more. This search bar is implemented through bootstrap.

We also added the ability to view all categories on our website. There is categories page, which has a table of hyperlinks of each category that exists in our site. Each hyperlinks will direct the user to a page of items that belong to the specified category. Both the table of category names and the table of items in a given category have search bars in the top right to allow a user to search within the tables. These search bars are implemented through bootstrap.

Unlike Amazon, in which a category may contain categories to form a "category tree", we have opted to have only one level of categories and to have each item only belong to one category.

4.6 Seller

All sellers have a seller summary page. The "seller summary" page displays a table with each item sold by the seller, along with the item's related information: its category, price, the quantity for sale, and whether or not the particular item is currently "for sale" or not. The "for sale" functionality allows for a seller to remove the item from active listing on the market, without deleting it from the database. This means the seller can easily re-list the item later, and any user who had bought the item previously can still view its details via the item's item page. Next to each item is a link to an "Edit Item" page specific to that item. This page auto-fills with the item's current data, and allows the seller to edit the item's category, price, quantity for sale, item description, and "for sale" or not.

At the bottom of the seller summary page is a link to "add item", that allows sellers to add new items. The seller must input all relevant information for the item before selling. There are input checks in both the "add item" and "edit item" pages that ensure the seller cannot enter invalid quantities or prices, such as negative numbers, or non-numerics.

4.7 Cart + History

The shopping cart is designed such that a user can add any item to their cart, as long as the quantity they add is less than the total quantity in stock. This means that any number of users can have the same item in their cart. On the cart page, the user can delete the item from their cart. JavaScript code was included to allow the user to edit the cart quantity using a set of '+' and '-' buttons for each item. The JavaScript function writes a POST request to the server to change the quantity incrementally. Editing the quantity is limited such that the user cannot decrement the quantity to less than zero, and the user cannot increment the quantity to be greater than the total quantity in stock for the item.

If the quantity in stock of an item changes so that it is less than the cart quantity, the cart quantity is updated to 0 and the user is notified. Cart items are links to the original page, so the user can navigate to the item page in case the item is still in stock.

When the user wishes to checkout, there is a checkout button below the cart. A JavaScript function was written to prompt a confirm dialog to pop-up. If the user agrees to checkout, the checkout is processed. If the user hits cancel, the items remain in the cart. If the user does not have enough balance to complete the purchase, a message will appear, and checkout will not occur.

There is a separate add balance page where the user can view their balance and add money. The user can enter any amount they wish. There are input checks done so that the user cannot enter a negative amount, nor can they enter non-numerics. The balance is also displayed in the user drop-down menu for convenient viewing purposes. For our implementation, we did not go into, adding a credit cart option, or something similar that would be more representative of real transactions online for the purposes of simplicity, but this could certainly be improved upon in a future implementation.

Checkout is executed item by item. For each item in the cart, the cart quantity is compared to the in stock quantity. If the cart quantity is greater than the in stock quantity, it will not be included in this purchase,

and after checkout, the user will still see it in their cart. If the in stock quantity is sufficient, then the user's balance is deducted, the in stock quantity is deducted, the seller's balance is updated, and the cart item is deleted. All these changes are committed before the next cart item is considered. After the changes are committed, the item purchased is also added to the Order History table. We performed testing where multiple users were checking out with the same item and same quantity, and found that the site only allowed one user to successfully purchase the item, while all other users received a message indicating that purchase of that item was unsuccessful, so the database and the site can handle concurrent users and transactions.

The database contains an Order History table. Every time an item is purchased successfully, this table is updated. A user can view their order history, where items are grouped by a purchase date, and orders are displayed beginning with the most recent. Users can link out to the item pages of the items they have purchased in the past, and can also link out to a page of seller reviews. The trade history, or seller history, is similar and shows the items the seller has sold.

Throughout the site, there are prices and monetary values displayed. By nature, HTML will not display a float value with two decimal places unless the value has two non-zero values after the decimal. A JavaScript function was written and applied to all monetary values displayed on the site, so all monetary values are displayed with two decimal places.

4.8 Item Reviews

An item's reviews are on its item page. There is a reviews table with all reviews for that item and a write review form. The write review form has 3 fields: "Location", "Stars", and "Write your review:". "Location" is the country where the reviewer is from and is a string field. "Stars" is the star review rating from 1-5 and is an integer field. If a non-integer or a number less than 1 or greater than 5 is the input for "Stars", there will be an error message and it will reload the item page with the reviews table and the write review form. "Write your review:" is a text field for writing out the content of the review. When the "Add review" button is clicked, it will successfully add a review to the database and display the new review on the item page if the new review is unique in terms of the review's user id, date, and content. If the new review already exists or there is an invalid input for Stars, there will be an error message and the item page will be reloaded. The review will not be added.

If there are no reviews for an item, instead of an empty table, there will be a subtitle underneath the "Reviews" title that states "There are no reviews for this item". If there is at least one review for that item, there will be a table with 6 columns: Reviewer, Date, Location, Stars, Content, and a blank title column. Reviewer is the username of the user who left that review, Date is the month/date/year form of the date the review was posted, Location is the country where the reviewer is from, Stars is the star review rating from 1-5 of the item, and Content is the written review. The blank title column has nothing if the review was not written by the logged-in user and has an edit link if the review was written by the logged-in user. The edit link takes the user to an edit review page with 3 fields (Location, Stars, Write your review), pre-populated with the original review, and 2 submit buttons (Finished, Delete). The 3 fields correspond to the 3 fields in the write review form and allow the user to modify location, stars, and review content. By clicking Finished, the review will be updated in the database based on the form and will redirect to the item page to show the modified review with all the other reviews. By clicking Delete, the review will be removed from the database, and the site will redirect to the item page. The deleted review will no longer appear in the reviews table.

4.9 Seller Reviews

Reviews for a seller can be accessed through a hyperlink next to the Seller attribute on every item page. If a user is a seller, the user may access their own seller reviews by using the hyperlink provided on the profile page. By clicking on the link, the site will redirect the user to the corresponding seller's reviews page. This page will have the seller's username at the top of the page, a reviews table with all reviews for that seller, and a write review form. The write review form has the same fields as the Item reviews form, mentioned in the previous section: "Location", which is the country where the reviewer is from, "Stars", which is the star review rating from 1-5, and "Write your review", which is a text field for writing out the actual content of the review. The same input checks are executed when the form is submitted, and error messages are returned if the input is invalid. The new review must also be unique in terms of the review's user id, date, and content. As long as the review is valid and the "Add review" button is clicked, the seller review will be added to the database and displayed on the seller's reviews page. If the new review already exists or there is an invalid input for Stars, there will be an error message and the seller's reviews page will be reloaded.

If there are no reviews for the seller, instead of an empty table on the seller reviews page, there will be a subtitle underneath the "Reviews" title that says "There are no reviews for this seller". If there is at least one review for that seller, there will be a table with 6 columns: Reviewer, Date, Location, Stars, Content, and a blank title column. This table follows the same format as the item reviews table, mentioned above. As before, the blank title column has nothing if the review was not written by the logged-in user and has an edit link if the review was written by the logged-in user. The edit link takes the user to an edit seller review page with 3 fields (Location, Stars, Write your review), again pre-populated with the original review the user wishes to edit, and 2 submit buttons (Finished, Delete). The user again also has the option to delete the review completely. Either way, changes made on the edit seller review are committed to the database, and the user will be redirected to the seller reviews page they were originally on, so they can see the changes they made.

4.10 Handling SQL Injection Attacks

We have handled SQL Injection Attacks primarily by following best practices regarding writing statements to the database, as well as utilizing SQLAlchemy. To begin with, we do not pass string literals to any queries or statements written to the database. We do not write raw SQL statements. We work with the ORM (Object Relational Mapper), so we interact with Python objects, which are then converted by SQLAlchemy into SQL statements [3]. SQLAlchemy also automatically escapes/quotes any special characters, like semicolons and apostrophes. [1]. These factors allow us to prevent SQL injection attacks to our site.

4.11 Prepared Statements

We did not incorporate prepared statements in the final version of our Mini-Amazon site.

5 Database/Dataset Generation

The data for the items, sellers, and categories is scraped from a .csv file containing information from the Amazon Toy dataset, which contains 10,000 items [2]. Every row in the .csv is read in, with the corresponding information from each labeled column, such as price, quantity, description, seller, and category. If a seller is read that is not yet in the database, they are added with an email address and username based off of their seller name. New categories are initialized in the same way. Fake reviews are added as part of the database generation when running the seed_db.py file, which was created to import data to our database. For every item, we add 5 random reviews. Each review is written by a randomly selected user among all website users, and is assigned today's date, a random location among "USA", "Canada", "United Kingdom", "China", and "Japan", a random star rating between 1 and 5, and a randomly generated 10-character string for its content. After all 5 reviews are added for that item, the average user rating is calculated as the average stars rating of all 5 reviews and is assigned to the item.

The original Amazon Toy dataset has a Categories variable. Each category observation for an item is a series of categories and subcategories, following a category "tree". Our site only has one category level, and one category item. To extract one category for each item, we parsed the highest level node from the dataset (e.g. if the original data was "A ¿ B ¿ C", our database only pulls category "A" into our "Category" table). Some items did not have this cell filled out in the original dataset. For these items, we assigned the item to the category "Other". We end up with 43 categories in our site. These categories were initialized as they were read in from the .csv file, and items were created.

Images were generated using the Google Image Search API, which allowed us to search by keyword, in this case the name of the category, and then resize and store the first result for that keyword in a static directory with a filename matching the category name. Any pages displaying items can then simply pull the image from the file with the name matching that item's category.

6 Challenges

We initially faced some challenges understanding how SQL queries and statements translate to SQLAlchemy, as none of us were extremely familiar with SQLAlchemy. We utilized many internet tutorials. We also met frequently and helped each other, since many of the queries and statements we needed to write were similar.

As we progressed further into the project and started working with our production dataset, we found that the initial database generation takes a long time and there was not much we could do to speed it up. Most of the time was likely spent on item generation and the 5 random reviews for each item. Since there are 10,000 items in the Toy Dataset and each item needs to generate 5 random reviews, items and their corresponding reviews require many iterations to generate.

In trying to get photos for each item, we tried a number of solutions, and ultimately decided to use a Google API to pull images. We found during our project that the particular in house Google API we were using to pull images required $5 for every 1,000 calls, which would have cost around $50 for our current database with 10,000 items. Due to this API key restraint, we were unable to upload a unique picture for each individual item in our database. We decided instead to upload one image per category, so items that belong to the same category have the same image. This means that every item is displayed with an image, and even newly added items by sellers will have an image as well based on the category selected for the item. This was our compromise to having images on our site.

7 Future Work

Our working Mini-Amazon has a basic implementation for a lot of the features on the real Amazon site, but there is certainly room for improvement.

7.1 Replying to Reviews

In order to improve the UX (User eXperience), many online retailers have included additional features in their reviews. For example, Amazon allows other users to reply with a simple "Yes" or "No" if a review was

helpful in determining to purchase this item. Also, Amazon does not allow a user to review an item unless a user has purchased the item. The App Store also allows app developers to reply to reviews and it is used as a means for responding to feedback given in reviews and/or providing customer service or interaction. These features would be great additions to our review section, but would require an overhaul of the UI.

7.2 Adding More Images

Due to API key restraints, we were unable to upload a unique picture for each individual item in our database. Obviously, images are a very important factor in selling items as on Amazon, and each item usually has many pictures so a potential buyer can look at an item from many angles (as if they were in-store). This would not be an issue for if sellers would be uploading individual images themselves, but in initializing the somewhat random images for this "Test" Database, it made more sense to go with one image per category. As such, adding the feature of uploading an image to the add item page would be a feature to work on in the future. We could modify our database design to include image attributes for an item. We may need to research how to convert images into binary files, or those suitable for storage in a database. Along the same lines, it would be nice for users to be able to upload their own avatars.

7.3 Refining Search

Online retailers often allow users to add more restrictions in searches. For example, Amazon allows users to determine the minimum average user rating (e.g. only shows items with 4 or more stars) or even allow a user to only search items in one category. Ideally, an improved version of our search page would allow a user to input more restrictions on their search.

7.4 Creating Category Trees

A "category tree" with multiple levels would scale much better than our one-category approach and improve the User Experience, allowing for more refined search. For example, "Household supplies" is a very broad category, as perhaps everything from laundry detergent to vacuum cleaners to shampoo may fall into this category. Having sub-categories like "laundry supplies" and "appliances" would help a user narrow down their search. Additionally, an item may be able to fall into two categories, or be linked to another item or category (e.g. computer bag). Having a network that makes such connections would be a fine addition to the item page.

7.5 Personalizing Recommended Items List

The recommended item list currently lists the top 3 items by average review rating for the 5 categories that contain the most items. While this is a simple but effective way to display highly sought after items to a user, it would be even better to generate a personalized recommended item list. Sites such as Amazon recommend items based on a user's activity including search history, order history, and advertisements that are clicked on. We could take a similar approach for our recommended item list and generate the recommended items based off of a user's order history. For example, the 5 categories in the recommended item list can be the 5 categories that the user has bought most from based on the user's order history. The 3 items per category can still be the items within the category with the highest average review rating or they can be items that most closely match the names of previously purchased items in the user's order history.

7.6 Editing User Information

Currently, users are only able to change their passwords. The ability for a user to change their email, username, and security questions/answers would be very useful. Many websites give you the ability to change such attributes.

7.7 Checking Out

There is one important scenario to note that exists with Checkout in our implementation. If a user is viewing their cart, and before they checkout, the seller updates the price of an item in the cart, the checkout price will be the newly listed price, but the user might not realize this until after checkout because the cart page does not automatically refresh, and the user only saw the former price. This is an edge case, so it should not occur extremely often, but we could certainly address it better in the future. If we had a separate checkout page, as sites like Amazon do, where the price queried from the database is frozen once the checkout page is reached, we could better address this edge case. In our current implementation, since checkout happens directly from the cart, it is difficult to implement this price freeze. We cannot necessarily implement this price freeze from the cart, since items in the cart are not set for purchase and may stay in the cart for long periods of time before final checkout. Overall, this is an edge case that we could definitely address in the future.

8 Conclusion

Building all aspects of this Mini-Amazon site gave us a unique opportunity to be creative and completely create a database, backend and frontend on our own. We were able to develop many features and demonstrate our understanding of how a database is set-up, how it is maintained, and how users may interact, indirectly,

with it. We reached a better understanding of challenges that a site like Amazon may face, considering how many users, including ourselves, interact with the site and how much data is hosted on the site. As mentioned before, our Mini-Amazon is incomparable to the actual Amazon site, but it was very interesting to think through how to implement simpler versions of very complicated processes and present features in user-friendly ways.

References

[1] *A step-by-step SQLAlchemy tutorial.*

[2] PromptCloud. *Toy Products on Amazon*, 2017.

[3] SQLAlchemy. *Key Features of SQLAlchemy.*