Ella Hagen

Code Review

1. For both create_account and create_shop, I think just for the sake of clarity when testing, it might be better to return the account id that was created instead of just "ok." This way if you don't have access to the database you can still follow the flow of that specific account/shop id.
    a. We implemented this feedback
2. For create_shop, I would first check to make sure that the shop name isn't already in the database, so that way two sellers won't have the same shop name.
    a. We implemented this feedback
3. Similarly to the one above, for create_account, I would add a check to make sure that the email that's passed in isn't already in the users table, so that way it's not possible to have multiple accounts under the same email.
    a. We implemented this feedback
4. With the search function, it might be better to have it output a string representing the shoe instead of just the listing_id, otherwise it's kind of hard to see what shoe is actually for sale.
    a. We implemented this feedback and now return additional information in search.
    b. In addition to making search clearer, we added a compare endpoint so we can isolate the two listings we are looking at.
5. This is kind of trivial since it's not an actual shop, but right now carts_checkout doesn't have a way for customers to pay. With the potion shop there was a payment string, so maybe you could add that as well.
    a. We implemented this feedback
6. In create_listing, there isn't any error handling, so even if a transaction fails, it still returns "ok." This can be pretty easily fixed though with a try, except clause.
    a. We implemented this feedback.
7. Similarly, create shop and create account return "ok" no matter what, so maybe consider adding more error handling in case something fails.
    a. We implemented this feedback
8. I saw this when testing, but it doesn't look like the search list is updated when a shoe is bought. I bought a pair of shoes, but then when I ran the search function again, the shoe was still there. I think it's because the search function is looking at the original quantity that was entered when the seller put in the shoe, but then the quantity is continuously changed through the ledger (everything works during checkout).
    a. We implemented this feedback
9. Also, since the quantity of a certain shoe is maintained in a ledger based system, there should be a check in the search function that sums the quantity and if the quantity is zero then it shouldn't be added to the return list.
    a. We implemented this feedback.
10. For checkout, this also isn't super necessary, but instead of returning false if the inventory isn't enough, it might be better to return "insufficient inventory," or something like that, so that way it's more clear for the customer why their transaction didn't go through.
    a. We implemented this feedback.
11. In create_cart, if you put in an account_id that doesn't exist, it will return an Internal Server Error, so it might be good to check first to see if it exists, or use a try/except statement.

    a. We implemented this feedback

12. Similarly to the one above, in set_item_quantity, if you put a non-existent listing id, it'll cause an Internal Server Error. So it might be good to add more extensive error handling so instead of just saying Internal Server Error, it'll explain what went wrong.
    a. We implemented this feedback
13. I'm not quite sure how to test this since the search function doesn't update, but I think that if someone has multiple items in their cart it will only checkout the first item since .first() is called, instead of doing something like for row in res:
    a. We implemented this feedback

Schema/API

1. For style in your api spec, you say it should be constrained to options such as "high tops, low, mid, etc," but right now a user could type in anything, so it might help to add an enum to the database so they'll be forced to choose.
    a. We did not do this because there are many possible styles of shoes. Our database takes in any types of shoes so there are many possibilities for what could be implemented, therefore we determined an Enum would not be reasonable for this portion.
2. Similarly, I think some other fields like color could be changed to an enum, just so it's more restrictive.
    a. We implemented this feedback.
3. It could be good to implement a function that gets all listings for a specific store or add a store field to the search function so if a customer has a specific store that they know and trust, they can look to see what the store offers.
    a. We implemented this feedback as part of a search rather than its own function.
4. For simplicity's sake, this might be unnecessary, but to fit your design, there are a couple fields whose data types could change. The main one being size could be a float instead, so that way people can sell shoes that are, for example, size eight and a half.
    a. We implemented this feedback
5. For store verification, it might be good to implement the ratings/review system, so that way only stores that have good reviews and ratings can be verified, regardless of how many shoes they have sold.
    a. We implemented this feedback
6. Maybe you could add a "condition" parameter to the listing so customers could know whether the shoe is used, new, lightly worn, or etc.
    a. We implemented this feedback.
7. I think this is just a matter of preference, but the endpoint paths could contain more information about the transaction taking place. For example, in the set quantity function, instead of "set_item_quantity", it could be "/{cart_id}/listing/{listing_id}."
    a. We implemented this feedback.
8. I would consider maybe automating the verification of stores, so maybe after a customer has purchased a shoe/left a rating, it would check to see if it meets the qualifications for verification. That way the administrators wouldn't have to do it by hand.
    a. We did not implement this feedback, as it would waste resources and time as our shop scales to check if a shop meets a verification. We approached this from the

perspective of Instagram verification, where users can apply for the status rather than being automatically awarded.

9. In the search function, in the array of listings, it might be nice to add whether or not the store that is selling it is verified or not.
    a. We implemented this feedback
10. This is just a suggestion, but the functions could be broken up into different files, perhaps the search function could go into it's own file, since I'm not sure if it really matches the purpose of the other functions in carts.py.
    a. We implemented this feedback
11. Similarly to the one above, I think the filter endpoint should probably be something like "/filter" instead of "/carts/filter" since it doesn't really have anything to do with carts.
    a. We implemented this feedback
12. Perhaps consider having it so the filter endpoint doesn't need the API key since it's a function that doesn't really need proper authorization and should be available for all customers to access.
    a.

Suggestions

##Suggestion 1: Store Discounts
It would be cool if sellers could decide to offer a store wide discount, where all of the listings that they offer are marked down by however much they decide and for however long they want the sale to go on for. This wouldn't require any database additions, just a new endpoint that would update the listings table. This could also work well with the filter endpoint, if there was an option so that users could only see shoes that are currently on sale.

We implemented this feedback with a flash sale feature

##Suggestion 2: Promoted Shoes
If sellers want to promote their shoes, they could pay a small fee, and then when a customer uses the search function, that shoe will be displayed first, as long as it matches the criteria. This would require a small database change, which would probably just be a boolean in the listing table that says whether or not it has been promoted. Additionally this would require a new endpoint that would take in payment and the listing_id of the promoted shoe.

We implemented this feedback with a promotional tier a shop can purchase

Jin Wu

Code Review

1. in carts.py, filter(min_price, max_price) should be floats not integers to allow for prices that are not whole numbers.
    a. No, was told to use ints and change currency to cents
2. in carts.py, max_price is declared as an integer but is constantly being referenced to as a string.
    a. We implemented this feedback

3. /create_cart does not check if the user already has an active cart, so in that case a user may create two unique carts in the carts table which may cause issues in checkout such as doubling their order when it is not intended.
    a. We implemented this feedback
4. in /checkout, it may be useful to have more data in the description such as quantity, listing_id, and money paid.
    a. We implemented this feedback
5. /create_account does not account for if the user already exists in the table. This may create duplicate users and may interfere with checkouts and other functions which expect only a single unique account.
    a. We implemented this feedback
6. I think the Listing(BaseModel) should have price as a float to allow for decimal values of dollars.
    a. We were told to use int and convert dollars to cents
7. in /create_listing, it may be helpful to also contain the price of the shoe in the description
    a. We implemented this feedback
8. a shop may bypass the post_application due to create_shop not checking if a shop already exists in the table so there may be duplicate shops in the shops table and it will count each shoe sold per shop extra and will get them verified when you don't want them to.
    a. We implemented this feedback
9. /verification_status will error out on shops with multiple entries in the table, and that is possible because create_shop does not check if a shop already exists in the table before inserting a new shop into the shops table.
    a. We implemented this feedback
10. I would return more descriptive results when a user sets item quantity or checkout, maybe returning OK with the description of the item i added to my cart.
    a. We implemented this feedback
11. I would re-route filtering and getting the catalog of items away from the carts endpoint.
    a. We implemented this feedback
12. Maybe add an endpoint where no filters are passed and all the items in every store are returned just so users can see items without having to hunt for them.
    a. We implemented this feedback with the search endpoint, not a separate one.

Schema/API

1. cart_items should always point to a specific cart so I think cart_id should never be null, that would mean there is an item in an unknown cart.
    a. We implemented this feedback
2. cart_items should not exist if there is no quantity associated with it, so the quantity should be not null and always be 1 or greater, as a cart_item with quantity zero should just not exist.
    a. We implemented this feedback
3. I would consider more descriptors to put under shoes. With how specialized this product is, the majority of users probably really care about exactly what shoe they are buying. Maybe add date_released, condition, and if it has all original accessories.
    a. We implemented this feedback with the condition.
4. I think adding a rating column to shops would be good for users to easily see how trustworthy a shop is at a quick glance.

    a. We implemented this feedback
5. I would add a reviews table with foreign key references to shops and the user who left the review as well as a rating(int) and a description(text) so users can leave ratings for particular shops and other users can judge a shop off of more descriptive ratings.
    a. We implemented this feedback but without description
6. Maybe add a way for users to display themselves without using either their email or real name as that can help with privacy.
    a. Not sure what this really meant. We do not display any information besides name and email now when one uses the get_account endpoint.
7. It seems that there is no way to take a listing off of the site after it has been sold other than to delete the listing all together. Maybe add a boolean column to it (active) to only display listings that are not sold, and keep all sold listings in the database.
    a. We have a filter to check the quantity. If it’s zero, we don’t show in the search results.
8. I think shop_balance_ledger (balance) should not be an integer as people can pay with cents as well, so a float would fit it better, same with all other variables representing dollar amounts.
    a. No, we used ints for converting to cents
9. I assume that dollar amounts are in USD but it might help to keep track of that in case you need to do conversions with other countries.
    a. No, we just wanted to focus on USD
10. Maybe add an endpoint in /catalog/ to display all listings without needing any filters to apply.
    a. Search endpoint handles the cases.
11. the endpoint /shop/create_shop takes in a store name and store owner both as strings. This may cause problems when there are two users on the platform with the same name. I think that creating a shop based off of a unique identifier (user id) would be much better.
    a. We implemented this feedback
12. the endpoint /carts/{cart_id}/checkout passes in gold_paid, which seems a bit odd for a real life shop.
    a. We implemented this feedback

Suggestions

Ratings

It would be nice if there was an endpoint like /shop/rating which took in an account_id and a rating (integer from 1-5) as well as a description (text). This endpoint would update a ratings table (which has a foreign key reference to a shop) and we could get a shops rating with the endpoint /shop/rating/{shop_id}

- We implemented this feedback

Threads / Discussion Board

For a particular shoe it would be cool if users could talk about the shoe and add comments/reply to others about it. There could be a /shoes/discussion/{shoe_id} endpoint which takes in an optional comment_id if this was a reply to another persons comment or would just take in text about the particular shoe.

- We did not implement this, but liked the idea. We wanted to focus on ratings.

Nick Patrick

Code Review

1. For create_account, you may want to check if a user is already in the database so there are no duplicate rows with the exact same information
    a. We implemented this feedback
2. Similarly, you may not want people to have the same shop name as another shop that already exists. I recommend checking that the shop name is unique in the create shop endpoint.
    a. We implemented this feedback
3. For your set_item_quantity function, I recommend adding the cart id, as well as the listing id into the post/curl statement. You can check the schema for the potion lab and basically copy gthe post statement there. "@router.post("/{cart_id}/items/{item_sku}")" replacing item_sku with listing_id
    a. We implemented this feedback
4. You can also do something similar in checkout. the curl statement should include the cart_id
    a. We implemented this feedback
5. Also, I recommend looking over shop.py and checking each of the curls. You could add "shop_id" or "account_id" where need be.
    a. We implemented this feedback
6. For checkout, you may want to return something other than True/false. for example, a message saying "Cart successfully checked out"/ "Checkout failed"
    a. We implemented this feedback
7. As of now, your post_application function just counts the sum of the quantity, meaning the total amount of shoes they have in their inventory at the moment. Instead, you should do something like count* where quantity <0 from the ledger. This would at least count how many times customers have purchased shoes from them, rather than their current stock of shoes.
    a. We implemented this feedback
8. I would recommend combining the post_application function and the uodate_verification. i do not see the purpose of having them in two separate functions.
    a. No because helps establish idempotency and thus concurrency control by preventing unnecessary posts if the planning logic fails, and also the intention is that an administrator would check the results of the application and manually update the shop's verification status instead of it being an automatic process. that way customers know that a verified shop has been reviewed by staff instead of a shop managing to game some algorithm
9. Pretty minor issue but you should change the server.py code to include members of your group as the contact
    a. We implemented this feedback
10. In filter, max and min price should be floats, not integers.

    a. We are using ints to convert dollars to cents
11. Also, size should be a float so that people can search for half shoe sizes.
    a. We implemented this feedback
12. You should also have a condition "if min_price != 1" so that you can filter by the minimum price
    a. We implemented this feedback

Schema/API

1. You should take a look at what columns in your tables can be null. For example I think something like "quantity" in your cart_items table should not be nullable and maybe have a default value of 0 or 1.
    a. We implemented this feedback
2. Your "shoes" table could have a few more columns to be more specific on the actual shoe. For example there are multiple Nike Dunk Lows that are blue. I recommend adding something like 'factory color' column to differentiate between say "University Blue" and "Blue Raspberry" dunks. You may want both to come up when someone searches for blue dunks, but you would want to differentiate between the blue color-ways.
    a. We implemented part of the feedback. Color is constrained now by an enum and we are only containing basic colors at this time, based on what we saw on Nordstrom.com
3. Additionally, You may want a column for 'release year' to differentiate between shoes of the same color-way which were released in different years. If someone was trying to sell their 2009 Jordan Flu Game 12's, they would be under the same shoe id as if someone were to sell their 2016 Flu Games. While the shoes have the same color-way and style, they have minor differences in details and a significant difference in price.
    a. We decided not to implement this feature as it would further muddle results for users rather than providing concise feedback about each listing. In addition some sellers may not have this info.
4. I think you may want to differentiate between 'users' and 'customers'. Some users might only want to host a shop, and others may only want to purchase shoes. A customer would need info such as 'Shipping Address' and 'payment info' so that they can purchase the shoes they want and they will be delivered to them.
    a. We discussed this as a group when we were initially planning and decided there were too many similarities to separate the two
5. You will want to make user_id a foreign key to the users table in your shops table.
    a. We implemented this feedback
6. You may want users to have a 'username' column, unless you plan on using emails as the username.
    a. Emails are used as the username
7. You may want to change the 'size' column in your listings table to a float to include half sizes. Perhaps you could also add a constraint so it must be either a full size or a half size.
    a. We implemented this feedback
8. You may also want to add a 'sold' boolean column to the listings table to differentiate between listings that are active and listings that have been sold already.
    a. We do not display any listings with 0 quantity

9. You will want the 'balance' column in the shop balance ledger to be a float in case people list shoes for an amount that is not an integer value
    a. We convert dollars to cents
10. Similarly, You might want the 'price' column in your listings to be a float in case someone wants to list a shoe for a price such as $199.99
    a. We convert dollars to cents
11. You probably should not store user's passwords as plain text in the users table. Like we went over in class, you should find a way to use a hashmap so their info cannot be accessed
    a. Not implemented, but we do know that this is vital in a real application.
12. You might want some type of constraint on the email column in the users table. For example it should end in '@gmail.com' or something similar
    a. There are too many emails for us to constrain as valid inputs. I do agree with proper verification of valid emails.

Suggestions

Suggestion 1

For my first endpoint suggestion, I think it would be very cool if users were able to trade their shoes. It might be difficult to implement but you could have a separate catalog for shoes listed for trade. Then, you could maybe have some things that that user is looking for in return. For example, what size they wear, what brands they are looking for, etc. It would also be cool to have something like a size-swap, where users could list their shoes up while looking for the same shoe in a different size.

We don't want to have a trading feature as this can happen through third party ways. And hard for us as the developers to guarantee safe trading with real shoes.

Suggestion 2

I think a great endpoint you could implement would be customer ratings on the shops. You could include the rating (1-5) and even perhaps allow customers to leave comments/reviews on their experience. I think this could also work well with your shop verification. Perhaps the shop would need a minimum rating (i.e. 3 stars) and/or a certain amount of reviews to become verified.

We implemented this feedback