Final Report

Caroline Levenson, Clara Lyra, Nicole Malpeli, Julia Ziaee, Jennifer Schultz

Refined Database Design

Changes from the last milestone have been highlighted.

Relations

User(UID, Email, Password, FirstName, LastName, Email, Street1, Street2, City, State, Zip)
Funding(UID, transactionDT, amount)
Seller(SellerID)
Products(ProductID, Name, Description, Category, unitPrice, Inventory, SellerID, image)

- We assumed that only one seller can list a product with the same product ID; however, you can list a product with the same name as another seller

Purchases(UID, ProductID, SellerID, OrderDateTime, finalUnitPrice, Quantity, FulfillmentStatus, FulfillmentDateTime)
Cart(UID, ProductID, SellerID, Quantity)
SavedItems (UID, ProductID, SellerID)

- Added this table for saved for later feature

ProductReview(UID, ProductID, Rating, numVotes, Review, DateTime)

- Number of votes was combined into one overall variable rather than up and down

SellerReview(UID, SellerID, Rating, numVotes, Review, DateTime)

- Number of votes was combined into one overall variable rather than up and down

Messages(UID, SellerID, Sender, Message, MessageDateTime)

- We have a messages table and setup SQL for it, but do not have it implemented as a frontend feature yet as this was just an extra credit feature we did not get to

Triggers

- TF_MessagingSeller()
  - Checks to see if user can message seller (condition is that must have made a purchase from them)
- TF_SellerReview()
  - Ensures that customer actually bought product from seller before review
- TF_ProductReview()
  - Ensures that customer actually bought product before review
- TF_DoubleProductReview()
  - Ensures that customer has not already submitted a product review
- TF_DoubleSellerReview()
  - Ensures that customer has not already submitted a seller review
- TF_validCartQuantity()

        - Raises error if quantity of an item existing in someones cart is no longer valid when they go to purchase that quantity
- TF_updateInventory()
    - Trigger to update the product inventory after a purchase
- TF_balance()
    - Trigger to ensure user has enough balance for cart order to be purchased
- TF_funds()
    - Trigger to ensure user has enough balance to deduct funds

Views

- cartTotalPrice(uid, totalPrice)
    - List total cost of all items currently in each users cart
- sellerpage(ID)
    - Create view page for buyer profile
- userBalance(id, amount)
    - Create view page to get each user's current balance

List of Features Implemented *(with notes on where the code can be found for documentation purposes)*

Account/ Purchases

Required (all required basic features)

- Functioning user login with the database existing users email and password
    - Can be found in login.html, login functions in user.py and users.py
    - Accessible from login button on top right of screen
    - Fully functional
- Register a user with a new account (enforces email is unique), system assigned id, email, password (gets hashed for storage), address
    - Can be found in register.html, register functions in user.py and users.py
    - Accessible from login button on top right of screen, then press register
    - Fully functional
- Allow users to update their information (except user id). Includes password, email and info in separate forms
    - Can be found in updateemail.html, updatepassword.html and updateuserinfo.html with update functions in user.py and users.py
    - Accessible from navbar account tab, account details tab, update buttons on screen
    - Fully functional
- Allow users to see and update their balance. Balance starts out at 0. Has trigger error handling to prevent users from withdrawing more than they have in their account.
    - Can be found in balance.html with functions in user.py and users.py
    - Accessible from account details nav header, account balance tab
    - Fully functional
- Show history of purchases with a summary that links to a more detailed order page

    - Can be found in orderHistoryOverview.html
    - Accessible from account details nav header, order history tab
    - Fully functional
- Show a public view for a user. Also shows more information for sellers and if they are a seller with reviews shows their reviews
    - Can be found in userdetails.html with function in users.py and calling sql functions in user.py and reviews.py
    - Accessible on product detail page if you click on seller ID, also accessible from sellers view of order history by clicking on user id for someone who bought their product
    - Fully functional

Additional Features

- Display spending history by category
    - Can be found in spendinghistory.html with spendinghistory function in users.py and get_by_category sql function in Purchase.py
    - Accessible under accounts tab under spending history
    - Fully functional

Products

Required (all required basic features)

- Shows a summary (image, product name and id, price, description, category) for each product in the result list, with options to view details or add item to cart
    - Can be found in index.html with functions in product.py and products.py
    - Fully functional
- Users can create new products for sale and edit products
    - Can be found in create.html and edit.html with functions in product.py and products.py
    - Fully functional
- Can click on a product entry to show a detail view of product, seller information, and reviews
    - Can be found in detailview.html and edit.html with functions in product.py, reviews.py, purchase.py, and products.py
    - Fully functional
- Users can change the quantity of the product before adding to cart
    - Can be found in detailview.html and edit.html with functions in cart.py and products.py
    - Fully functional
- Added filtering by category and keyword and sorting by price
    - Can be found in index.html and edit.html with functions in product.py and products.py
    - Fully functional

Additional Features

- Added filtering by price (0-100, 100-200, 200-300, 300-400, 400-500, 500+)
    - Can be found in index.html with functions in product.py and products.py
    - Fully functional
- Added sorting by newest, oldest, lowest quantity, highest quantity
    - Can be found in index.html with functions in product.py and products.py
    - Fully functional

Cart/ Order

Required (all required basic features)

- Users can add to cart using the 'Add to cart' button from both the product detail page & the homepage listing all products. They will be redirected to the login page if they attempt to add to cart or access their cart without being logged in.
    - Will not let the user add more to cart than the current inventory amount available
    - **Files:** products.py, index.html, cart.py, login.html, users.py
    - **fully functional**
- Each user has their own cart, which lists all products line by line with: option to change the quantity of each item, unit price of each item, product image/name/id of each item, total price of the item at the quantity user wants (changes dynamically as user changes quantity), option to remove the item from their cart, and option to move the item from 'cart' to their 'saved for later' items
    - **Files:** cart.html, navbar.html, cart.css, products.py, saved.py
    - **fully functional**
- Cart lists subtotal for all items (changes dynamically as user adds/removes items and as item quantities change)
    - **Files:** cart.html, cart.css, products.py, cart.py
    - **fully functional**
- Cart has option to 'checkout' at the bottom of the page; clicking button will either result in an 'order success' message that gives users buttons with options to either view their order history or go back to the product listing home page (i.e. 'continue shopping')
    - Failure messages were coded with error handling (in cart.py & create.sql). If there is an error with just one item in the cart being purchase the entire order will not go through (i.e. if there are other items in the cart they will not be purchased either). The following cases were accounted for in our db design with triggers and display specific messages instead of a general error message:
        - 'User ___ has insufficient funds for this order' (i.e balance error)
        - User ___ can no longer purchase ___ units of ___ ' (i.e. inventory error)
    - **Files:** cart.html, index.html, orderHistoryOverview.html, orderFail.html, orderSuccess.html, cart.py, products.py, users.py, cart.css
    - **fully functional**
- Once a user checks out, the order will be added to their order history, which is displayed as a page under the 'Account' tab. Orders are grouped by date/time of order (so clicking 'checkout' in the cart groups all items currently in the cart under the date/time the button was clicked). User can see the date/time of an order, the address it will be/was shipped to, total price of order, number of items ordered, overall order fulfillment status, the date/time of when fulfillment status was last updated, and a button to click to view the details of their order.
    - When user clicks 'Details', they are sent to a page that lists each line item in their order with the following attributes for each item: Order date, product name, product id, seller id, the fulfillment status of that individual item, the unit price the item was purchased at, the quantity of an item purchased, the total price spent on that item at that quantity, and if the order has been fulfilled they have the option to leave a review for the product or, if they have already left a review, edit their existing review for the product.

    - **Files:** cart.html, orderHistoryOverview.html, orderhistory.html, newreview.html, updatereview.html, cart.py, products.py, users.py, reviews.py, cart.css
    - **fully functional**

Additional Features

- 'Save for later' functionality: for every item in a user's cart, they will see the option to 'Save' the item, which will move the item out of their cart into their saved for later list. On the 'Saved' page, for each line item, the user can see: product image, product name, product id, unit price of the product, the option to remove the item from their saved list, and the option to add the item to their cart (which will delete it from saved when moving to cart and will only add the item to their cart with a quantity of 1 and the user can change the quantity in the cart if they want)
    - attempting to access the saved page without being logged in will redirect user to login page
    - **Files:** saveditems.html, navbar.html, saved.py, products.py, cart.css
    - **fully functional**
- Aesthetic features: Used CSS to make buttons dynamically change color when the user hovers over it to indicate that they would be invoking the button's functionality if they click right now. Additionally, did overall UI cleanup & smoothed out user flow with the CSS.
    - **Files:** cart.css, cart.html
    - **fully functional**

Inventory/ Order Fulfillment

Required (all required basic features)

- Users are able to list items so that they are available for purchase and a user who has listed an item becomes a seller. The form checks that the fields are filled and that numeric fields have the correct values.
    - Can be found in create.html with functions in products.py and product.py
    - List tab
    - Fully functional
- Sellers have an inventory page that lists all product for sale by this user. The user can view and change the available quantity for sale by this user or remove it from the inventory altogether. Products are not shown in inventory or products if they have an inventory of 0.
    - Can be found in inventory.html with functions in inventory.py and products.py
    - Sellers tab → Inventory
    - Fully functional
- Sellers can browse/search the history of ordered fulfilled or to be fulfilled, sorted in reverse chronological order
    - Can be found in order.html with functions in orders.py and products.py
    - Sellers tab → Orders
    - Fully functional
- Orders are clickable so that all items in an order can be viewed. Sellers are able to mark each item in an order as fulfilled. This changes the status of the full order and the current date/time is captured as the fulfillment date/time.
    - Can be found in individualorder.html with functions in orders.py and products.py
    - Sellers tab → Orders → Edit/View Details

    - Fully functional

Additional Features

- Add analytics about buyers who have worked with this seller (e.g. amount spent, units bought)
    - Can be found in selleranalytics.html, orders.py, and products.py
    - Sellers tab → Analytics
    - Fully functional
- Able to search order analytics by buyer id and sort by amount spent and units bought (i.e. most to least or least to most)
    - Can be found in products.py, selleranalytics.html, and orders.py
    - Sellers tab → Analytics
    - Fully functional
- Able to search order history by buyer id and sort by date placed (i.e. newest to oldest, oldest to newest)
    - Can be found in products.py, orders.html, and orders.py
    - Sellers tab → Orders
    - Fully functional
- In Orders, buyer ids are clickable and take you to the user detail page of that user
    - Can be found in orders.html and links to userdetails.html
    - Sellers tab → Orders → Click blue id
    - Fully functional
- In seller analytics, buyer ids are clickable and take you to the user detail page of that user
    - Can be found in selleranalytics.html and links to userdetails.html
    - Sellers tab → Analytics → Click blue id
    - Fully functional

Feedback/ Messaging

Required (all required basic features)

- A User can submit a single review for products they have purchased
    - Can be found in reviews.py, detailview.html, products.py, newReview.html, updateReview.html
    - They cannot access the create new review screen if they have not purchased this product or if they have already left a review for it
    - Fully functional
- A User can submit a single review (no duplicates) for seller they have purchased from
    - Can be found in reviews.py, accountdetail.html, users.py, newSellerReview.html, editSellerReview.html
    - They cannot access the create new review screen if they have not purchased this seller or if they have already left a review for them
    - Fully functional
- Users can edit or remove any reviews they authored
    - Can be found in reviews.py, detailview.html, products.py, users,py, editSellerReview.html, updateReview.html
    - They can do this on the seller public page, detailed order history page, the product detail page, or on their account details page

    - Fully functional
- Users can see off the review they authored on their account details page
    - Can be found in reviews.py, detailview.html, products.py
    - Fully functional
- There are summary ratings for both products and sellers
    - Can be found in reviews.py, detailview.html, products.py
    - You can view both the total number of reviews and the average rating
    - Fully functional

Additional Features

- Buttons to create and modify reviews only appear modularly when the user is signed in and is the author of the review
    - Can be found in reviews.py, detailview.html, products.py, purchase.py
    - Fully functional
- Upvote and downvote functionality for all users in all types of reviews
    - Can be found in reviews.py, detailview.html, products.py
    - Fully functional
- Messages table and SQL setup for it, but do not have it implemented as a frontend feature yet as this was just an extra credit feature we did not get to (partially implemented)
    - Can be found in create.sql

Other Additional Features for Extra Credit

- Web scraping for some real product data (fully functional)
    - Data is realistic otherwise in terms of dates being in the right order, only having purchases from real sellers, etc.
    - Web scraping is under db/data in productScrape.py
    - Creating the data/overwriting with real scraped data in db/data in dataGenerator.py
- Improved UI including creating a logo for our website (fully functional)
- Added pagination (fully functional)
- Commented code (functions and sql queries) throughout our database (fully functional)