Project Report

Group Members and Roles

- Audrey Kline: Account/Purchases
- Michaela Gates: Products
- Aayushi Patel: Cart/Orders
- Adeildo Vieira: Seller/Inventory

GitLab Repo: https://gitlab.oit.duke.edu/agk27/mini-amazon-skeleton-24-fall
Video Link (also in README.txt): **316 Final Demo.mp4**

Database Design:

- **User(id, email, firstname, lastname, is_seller, balance)**
  - **id**: Unique identifier for each user account.
  - **email**: Email address used for login and account verification.
  - **firstname/lastname**: Name of the user provided during registration.
  - **is_seller**: Boolean value indicating if the user is a seller (True) or a buyer (False).
  - **balance**: Account balance for the user, used for purchases.

- **Purchase(id, uid, pid, price, name, time_purchased, fulfillment_status, tracking_id, carrier)**
  - **id**: Unique identifier for each purchase transaction.
  - **uid**: Foreign key referencing Users(id), representing the user who made the purchase.
  - **pid**: Foreign key referencing Products(id), representing the purchased product.
  - **price:** Price of the item purchased.
  - **name:** name of the product purchased
  - **time_purchased**: Timestamp of when the purchase was made.
  - **fulfillment_status:** Fulfillment status of the order containing the purchase
  - **tracking_id:** Tracking ID for the order containing the purchase
  - **carrier:** Carrier of the order containing the purchase

- **Product(id, short_name, description, image_url, price, available, category)**
  - **id**: Unique identifier for each product.
  - **short_name**: Name of the product.
  - **description:** Description of the product
  - **image_url**: url of image product
  - **price**: Price of the product.

    - **available**: Boolean value indicating if the product is currently in stock and available for purchase.
    - **category:** Category of the product

- **Cart(<u>id</u>, uid, pid, name, price, quantity)**
    - **id**: Unique identifier for each cart entry.
    - **uid**: Foreign key referencing Users(id), representing the user associated with this cart entry.
    - **pid**: Foreign key referencing Products(id), representing the product in the cart.
    - **name**: Product name that is in the cart
    - **price:** Price of the product in the cart
    - **quantity**: Quantity of the product in the user's cart.

- **Inventory(<u>id</u>, seller_id, product_id, quantity, price, product_name)**
    - **id**: Unique identifier for each inventory record.
    - **seller_id**: Foreign key referencing Users(id), representing the seller associated with this inventory.
    - **product_id**: Foreign key referencing Products(id), representing the product in the inventory.
    - **quantity**: Quantity of the product available in the seller's inventory.
    - **price**: Price of the product the seller sets (if different from the base price in Products).
    - **product_name**: Name of the product

Changes to schema since milestone 4:

- Added price, fulfillment_status, tracking_id, carrier to Purchases table
- Added name, price to Cart table
- Added product_name to Inventory table
- Added category to Product table

Features Outline and Status:

Account / Purchases:

1. Login (Fully functional)
2. Register if a new user (Fully functional)
3. View purchase history (Fully functional)
4. View order details page for a past purchase (Fully functional)
5. View other users and email if they’re a seller (Fully functional)
6. View user balance (Fully functional)
7. Add/withdraw from balance (Fully functional)

8. User can change their password (Fully functional)
9. User can change their email (Fully functional)

Products:

1. Browse all available products (Fully functional)
2. Filter available products with k top/bottom-most expensive (Fully functional)
3. Search available products for a specific product (Fully functional)
4. View detailed product page with description, inventory, etc. (Fully functional)
5. Add product to the cart with specified quantity (Fully functional)

Cart/Order:

1. Browse products and add available products to the cart (Fully functional)
2. View cart with unique line items with varied quantities as per added to cart (Fully functional)
3. Edit line items (increase/decrease quantities or remove altogether) in cart (Fully functional)
4. Submit a cart as an order - and upon submission cart empties (Fully functional)
    a. Submission of an order increments the seller's balance and decreases the buyer's balance (Fully functional)
    b. You cannot order items that are not available in inventory (Fully functional)
5. Cart contents are persistent (Fully functional)
6. Each user's detailed order page with fulfillment/ shipping details is available within their purchase history (Fully functional)

Inventory/Order Fulfillment:

1. Register as a seller when registering as a new user (Fully functional)
2. View the seller's unique inventory when logged in as that seller (Fully functional)
3. Add/remove from inventory or change availability status (Fully functional)
4. The seller can fulfill the orders and change it's status (Fully functional)

Advanced Features Outline (for "plus points")

Account / Purchases:

- Purchase history filtering: A user can filter their purchase history by date of purchase using the calendar pop-up from the filter bar
- Dynamic Balance Updates: a user can view their balance, add to their balance, or withdraw from their balance, all of which update the balance view automatically (using CSS & HTML) so a user can immediately view changes to their balance

Products:

- N/A

Order / Cart:
- N/A

Inventory:
- A comprehensive feature of the order's shipping process when fulfilling that implements carrier selection and tracking ID to the order; it is integrated with the user's end, and it updates into purchase history.

Other:
- Aesthetic and realistic front-end creates exceptional user experience
- User flow is smooth and logical with buttons to return back to previous pages or advance to "View Details" (i.e. within the Purchase History page)
- Data is realistic and generated dataset is large
- All files are thoughtfully commented through and easily understood by reader outside of the team

Additional Notes

Dataset:

Data is generated by our gen.py script, which generates the quantities (adjustable) for each CSV (db/generated/):
- Users.csv: 500
- Purchases.csv: 700
- Products.csv: 700
- Inventory.csv: 700
- Cart.csv: 500

Hard-coding:
- No values have been hard-coded

SQL Injection Attacks:
- Our user input (emails, passwords) is not fed directly into our User database schema in a manner that allows the User to inject input that can run within our queries. Additionally, when Users first register, there are constraints imposed on their email formats; even if they are fake emails (like boo@boo.com), they need to be formatted as general emails, and we have implemented hashing for our passwords.