Database Design

*All tables here have already been created within our project repo

Account / Purchases (Albert)

Tables:

- Users: contains all information regarding a user
  - user_id: system-assigned id
    - PRIMARY KEY
    - NOT NULL
  - email
    - UNIQUE
    - NOT NULL
  - first_name
    - NOT NULL
  - last_name
    - NOT NULL
  - address:
    - NOT NULL
  - password
    - NOT NULL
  - balance:
    - DEFAULT 0,
    - CHECK(balance >= 0),
    - NOT NULL
- Purchases: a way to track specific product purchases made by a user
  - order_id:
    - References Order(order_id)
    - NOT NULL
  - product_id
    - References Products(product_id)
    - NOT NULL
  - quantity
    - NOT NULL
    - check(quantity >= 0)
  - fulfillment_status
    - Not null, FOREIGN KEY
    - check(Fulfillment_status in (‘ordered’, ‘shipped’, ‘delivered’)
- Orders: has information on orders which makes it possible to group products that were ordered together
  - order_id
    - PRIMARY KEY
    - NOT NULL
  - user_id
    - NOT NULL

        - REFERENCES Users(user_id)
    - total_price
        - NOT NULL
        - check(total_price >= 0)
    - total_items
        - NOT NULL
        - check(total_items > 0)
    - time_stamp
        - NOT NULL
        - Current_timestamp AT TIME ZONE ‘UTC’
- Sellers: a separate table to list sellers
    - user_id
        - NOT NULL
        - REFERENCES Users(user_id)

Constraints / Assumptions:

- A user that tries to sign in with an invalid email/password will be denied
- Users can update all info in user table besides user_id
- A new user has a balance that starts out as $0 and can have the ability to deposit/withdraw
- Purchase history can be viewed by user sorted in reverse chronological order by default
- A public view for a user will be handled in the frontend and take info from the user table
- There will be a separate seller table that just references the user_id of the user table to keep a list of sellers in the table

Products (Advaita)

Tables:

- Categories: possible categories that products can belong to
    - cat
        - PRIMARY KEY
        - NOT NULL
- Products: all items in our online store
    - product_id
        - PRIMARY KEY
        - NOT NULL
    - user_id: seller of the product
        - NOT NULL
        - REFERENCES Users(user_id)
    - category
        - NOT NULL
        - REFERENCES Categories(name)
    - name
        - UNIQUE
        - NOT NULL

    - description
        - UNIQUE
    - price
        - NOT NULL
    - imageurl
    - quantity
        - NOT NULL
    - available
        - BOOLEAN DEFAULT TRUE
    - avg_rating

Constraints / Assumptions:

- A product will belong to exactly one product category
- Search/filter functions will be added to sort by category, price, and name/description
- A detailed product page will utilize the products table
- Users can create new products: once they do, they become listed as a seller
- Sellers of their products will be able to edit their products' info

Cart / Order (Alex)

Tables: *Order was handled in Account / Purchases

- Cart: shows a user's cart
    - user_id
        - NOT NULL
        - REFERENCES Users (user_id)
    - product_id
        - NOT NULL
        - REFERENCES Products (product_id)
    - seller_id
        - NOT NULL
        - REFERENCES Users (user_id)
    - quantity
        - NOT NULL

Constraints / Assumptions:

- The Cart table's primary use is to save a user's cart even after they exit the site
- Each tuple in the Cart table represents one item with information on whos cart it belongs to and the seller of the item with the quantity
- Quantity should be editable, items should be addable and removable, and carts should be able to be submitted as an order
- Each user has exactly one cart
- Must check inventories before confirming the order submit
- Balances must be updated and cart must be empty after order is placed

Inventory / Order Fulfillment (Parker)

Tables:

- Inventory: marks how many of each product is available
    - user_id
        - REFERENCES Users(user_id)
    - product_id
        - REFERENCES Products(product_id)
    - quantity
        - NOT NULL
- Fulfillment: marks if an order is fulfilled
    - user_id: seller id
        - REFERENCES Users(user_id)
    - order_id: has info on buyer user_id
        - REFERENCES Orders(user_id)
    - fulfillment_status
        - NOT NULL
        - check(fulfillment_status in ('ordered', 'shipped', 'delivered'))

Constraints / Assumptions:

- A seller will have an inventory page where they can add product to inventory and edit available quantity of a product
- A seller can browse through a history of orders to be fulfilled/already fulfilled
- Fulfillment shouldn't show info of other sellers' products
- Fulfillment shouldn't affect quantity as this is handled in Cart / Order submissions

Feedback / Messaging (Aaric)

Tables:

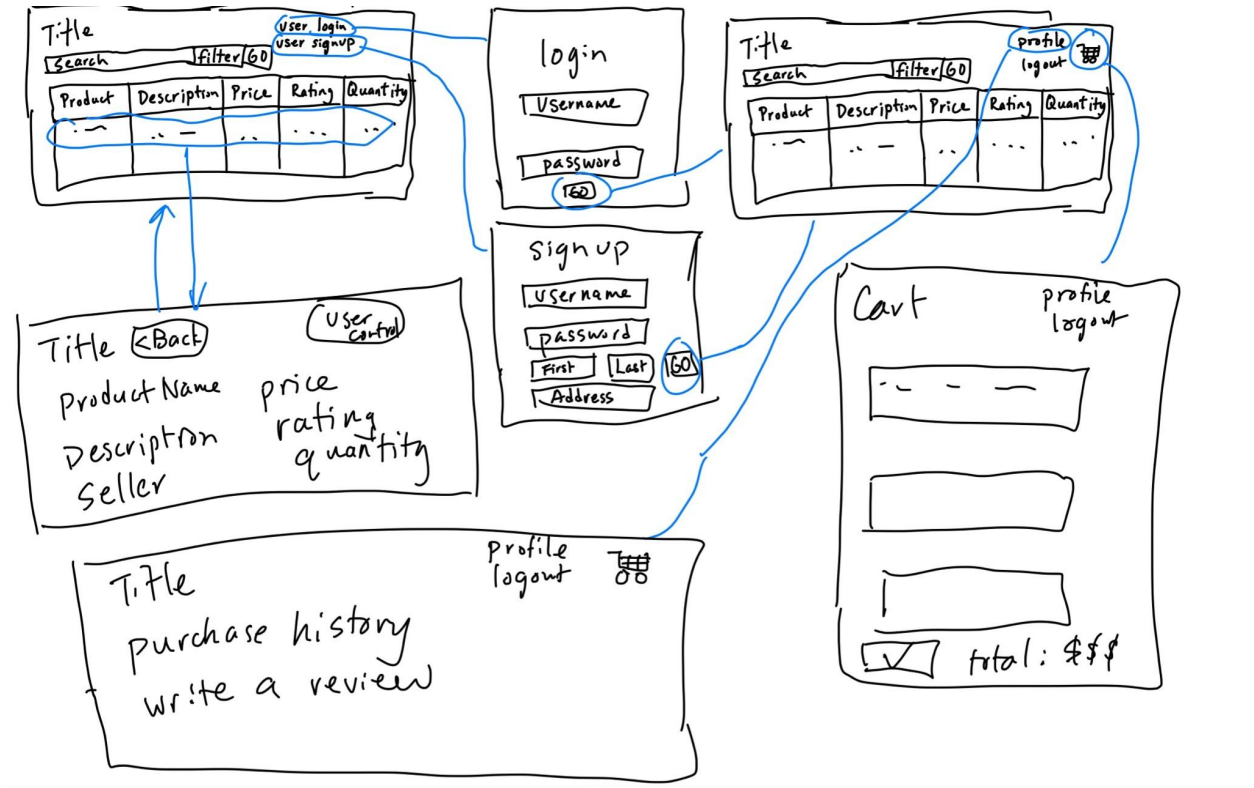
- ProductReviews: a way for users to review a product with a description and a rating
    - user_id
        - NOT NULL
        - REFERENCES Users (user_id)
    - product_id
        - NOT NULL
        - REFERENCES Products (product_id)
    - rating
        - NOT NULL
        - CONSTRAINT rating_limit CHECK (rating BETWEEN 1 AND 5)
    - review
        - NOT NULL
- SellerReviews: a way for users to review sellers with a description and a rating
    - user_id
        - NOT NULL
        - REFERENCES Users (user_id)
    - seller_id
        - NOT NULL

        - REFERENCES Users (user_id)
    - rating
        - NOT NULL
        - CONSTRAINT rating_limit CHECK (rating BETWEEN 1 AND 5)
    - review
        - NOT NULL

Constraints / Assumptions:

- One user can't submit multiple ratings for same product/seller
- A user can submit a rating on the detailed product page
- A user can edit/remove any of the reviews they made
- A user can be able to see the reviews they've made sorted in reverse chronological order by default
- Product reviews and seller reviews should all be able to be seen in their respective tables

Page by Page Design



Misc. notes about working on Milestone 2:

user_id: between 1 and 3 let's say for now (from users)
product_id: from 1 to 20 (from products)
order_id: 0 to 2 (from orders)

purchased quantity = quantity (from purchases)

Sample data I am going to add:
Inventory

user_id	product_id	quantity	price
0 (the only seller so far)	1 (macbook pro)	100 (at least 4 from purchases)	1599.99
0	2 (butterfly knife)	200 (at least 3 from purchases)	99.99

fulfillment

user_id	order_id	fulfillment status
0	0	delivered
0	1	delivered