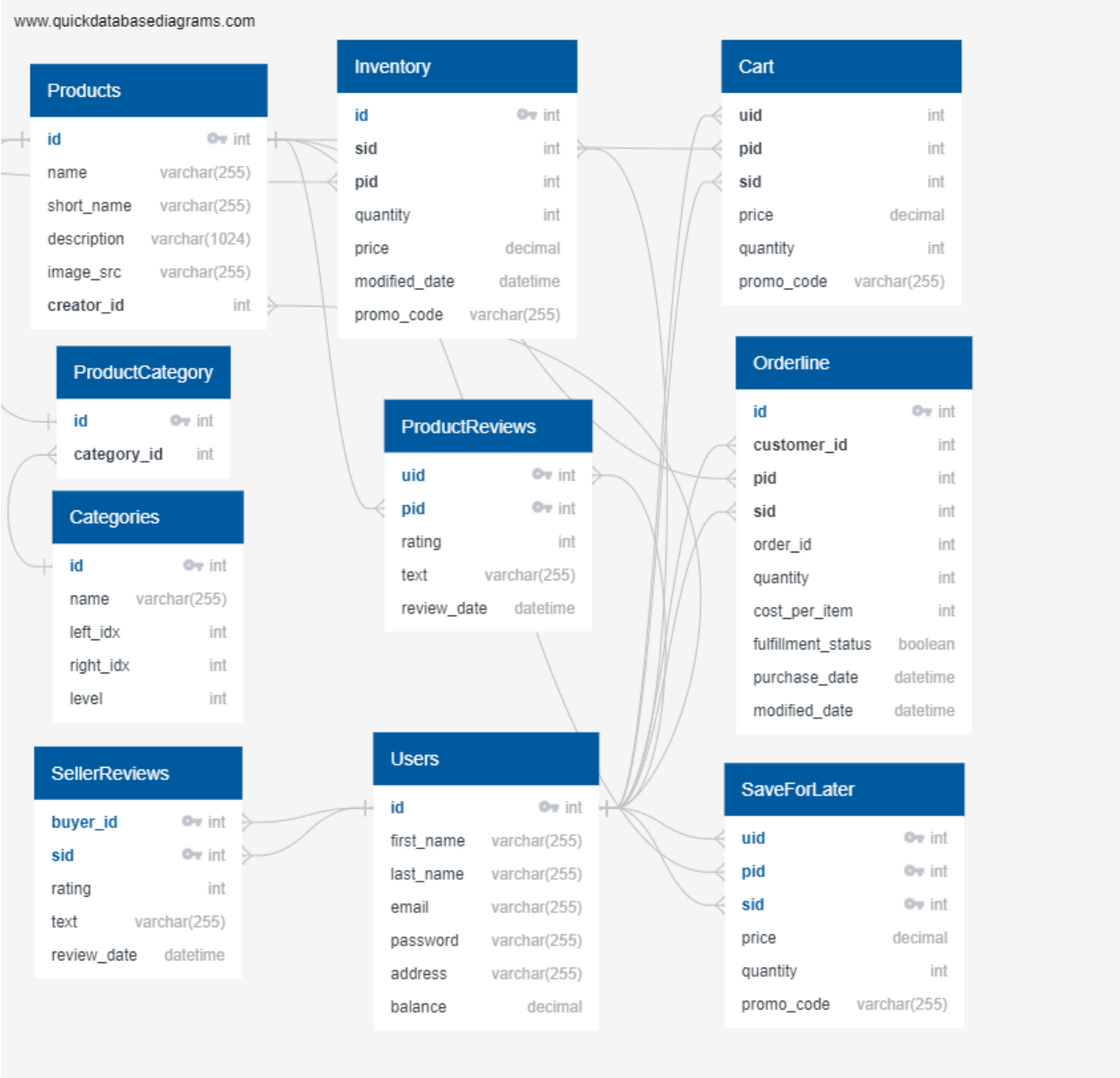
CS 516 Final Project Report



This is the new database schema

Changes to database schema:

- Orderline
  - Added purchase_date
- SellerReviews
  - Made (buyer_id, sid) the primary key
- ProductReviews
  - Made (uid, pid) the primary key

- Cart
    - Made (uid, pid, sid) the primary key
    - Added a price field
- Inventory
    - Made id the primary key
- Categories
    - Added left_idx, right_idx for nested set model
    - Added level which corresponds to the category's level within the category tree
- SaveForLater
    - Made (uid, pid, sid) the primary key
    - Added a price field
- Products
    - Removed the available field
    - Added a creator_id to specify which user created the product

Account / Purchases

Basic Requirements

1. A new user can register for a new account; an existing user can log in using email and password. **(fully implemented)**
2. Each user account has a system-assigned id. Other account information includes email, full name of user, address, and password. Users can update all information except the id. Ensure that email is unique among all users. **(fully implemented)**
3. A new user can register for a new account; an existing user can log in using email and password.Each user account has a system-assigned id. Other account information includes email, full name of user, address, and password. Users can update all information except the id. Ensure that email is unique among all users.**(fully implemented)**

- Each account is associated with a balance. It starts out as $0, but can be topped up by the user. The user can also withdraw up to the full balance. In practice these actions would require some real payment mechanism, but it is not required for this project. **(fully implemented)**
- Users can browse their history of purchases, sorted in reverse chronological order by default. For each purchase in this list, show a summary (e.g., total amount, number of items, and fulfillment status) and link to the detailed order page (see Cart / Order). **(fully implemented)**
- Provide a public view for a user. It will show the account number and name as well as any other summary information you deem necessary. If the user also acts

as a seller, show also email, address, and include a section with all reviews for this seller (see Feedback / Messaging). **(fully implemented)**

Bonus Feature

- Search/filter purchase history by item, by seller, by date etc. **(fully implemented)**
- Visualize history of balances, spending amounts and purchases by category, etc. **(not implemented)**

Products

Basic requirements:

- There is a list of predefined product categories, and each product belongs to one category. At the minimum, a product should have a short name, a longer description, an image, and a price. **(fully implemented)**
- Users can browse and search/filter all products. For each product in the result list, show a summary (e.g., image, name, average review rating, etc.) and link to the detailed product page. At the minimum, support browsing by category, searching by keywords in name/description, and sorting by price. **(fully implemented)**
- A detailed product page will show all details for the product, together with a list of sellers and their current quantities in stock. For each seller, provide an interface for adding a specific quantity of the product from this seller to the user's cart (see Cart / Order). The page should also include a section showing all reviews for this product (see Feedback / Messaging). **(fully implemented)**
- Users can create new products for sale. The user who created a product will be able to edit the product information. **(fully implemented)**

Bonus features:

- There are options to filter the list of products shown in the landing page according to various criteria (seller rating, min price, max price, and category)
- Hierarchical product categories implemented using a nested set model. Products in the database only have categories at the leaves of the tree structure. Products can be filtered according to these subcategories on the landing page at different levels of granularity. When a user adds a new product, it is given a default, hard-coded leaf category.
- For the same product, different users in our system may create their own versions of this product in our system. Different sellers can charge different prices - in this way each product page is its own competitive marketplace.

Cart / Orders

1. A user who is not logged in will not be able to see the Cart - they will be instead directed to the login page. **(fully implemented)**
2. A logged in user can select any product from the main page. In the product page, they can choose the specific seller they want to buy the product from and add the desired quantity to either cart or a wishlist. **(fully implemented)**
3. Once the user is finished shopping, they can proceed to checkout by clicking on Cart.
    a. They have the option to modify the quantity of any item they want to buy or remove items from Cart. **(fully implemented)**
    b. The page also shows their current wishlist. Items from the wishlist can be added to Cart or removed entirely. **(fully implemented bonus feature)**
    c. The user can then click on Place Order, which takes them to the final order page. This page contains the same details as the Cart page, but also shows the fulfillment status of each order. This page can be accessed anytime by seeing the user's purchase history (see Purchases). **(fully implemented)**
4. There are a few checks in place to enable a seamless operation for the users.
    a. If the user does not have sufficient balance in their account, they will not be able to place an order for items in their cart - they will instead be directed to an account which says 'insufficient balance'. **(fully implemented)**
    b. There are checks in the product and carts page to ensure that the user cannot add to cart a quantity greater than in the inventory of the corresponding seller. **(fully implemented)**
    c. Once the order is placed, the user's account balance is decremented and the seller's account balance and inventory is incremented. **(fully implemented)**
5. Addition of promo codes **(partially implemented in database, but not shown in front end)**

Inventory / Order Fulfillment

Basic requirements:

- A user who wishes to act as a seller will have an inventory page that lists all products for sale by this user. **(fully implemented)**
- A product sold by this seller can be added to their inventory. **(fully implemented)**

- For each product in the user's inventory, the user can view and change the available quantity for sale by this user, or simply remove it altogether from the inventory. **(fully implemented)**
- A seller can browse/search the history of orders fulfilled or to be fulfilled, sorted by in reverse chronological order by default. **(fully implemented)**
- For each order in this list, a summary of buyer information including address, date order placed, total amount/number of items sold by this seller, and overall fulfillment status. **(fully implemented)**
- There exists a mechanism for marking a line item as fulfilled. This interfaces with the Cart / Order functionality, which will show the final order marked as fulfilled if the buyer searches for it. **(fully implemented)**

Bonus features:

- Seller analytics page displays tables ranking products and customers by the amount of revenue each generated since your inception as a seller. **(fully implemented)**
- Top products table has a button that lets you increase the quantity available for your top product. **(fully implemented)**
- Top buyers table displays each customer's email, allowing you to get in touch and target customer retention efforts. **(fully implemented)**
- Add analytics about buyers who have worked with this seller, e.g., ratings, number of messages, etc. **(not implemented)**

Seller analytics page displays tables ranking products and customers by the amount of revenue each generated since your inception as a seller. Top products table has a button that lets you increase the quantity available for your top product. And top buyers table displays each customer's email, allowing you to get in touch and target customer retention efforts

Feedback / Messaging

Basic requirements:

- A user can submit a single rating/review for a product. The submission link will be incorporated in the detailed product page (see Products). The user cannot submit multiple ratings/reviews for the same product, but can edit/remove any existing ratings/reviews by this user. **(fully implemented)**
- A user can submit a single rating/review for a seller, provided that the user has ordered something from the seller. The submission link for this rating is in the detailed order page. The user cannot submit multiple ratings/reviews for the same seller, but an existing rating/review can be edited or removed. **(fully implemented)**

- Each user should be able to list all ratings/reviews authored by this user, sorted in reverse chronological order by default. From this interface the user should be able then select ratings/reviews to update. Incorporate the link to this interface in user account view (see Account / Purchases). **(fully implemented)**
- Produce summary ratings for products and sellers; pages or sections showing lists of reviews for products and sellers. At the very least, the summary needs include the average and number of ratings; the reviews lists should be sorted by rating or date. **(fully implemented)**

Bonus features:

- Public lookup of reviews made by any user, just like Amazon. **(fully implemented)**

Generative AI Acknowledgement

Generative AI was used extensively across the repository both through code completions and code/assistance via ChatGPT.