Online Cosmetic Retail Store

group 6 COMP421 - Project Part I
Bessie Luo, 260708568
Xintong Li, 260710002
Zhuoran Zhao, 260716696
Lancer Guo, 260728557

I.INTRODUCTION

I.I PURPOSE

As more and more people are purchasing products online, the popularity of online stores skyrocketed and together with the belief that wearing makeup will provide a fleeting confidence boost to some, we decided to model an online cosmetic retail store that is similar to the design concept of Sephora. Our store contains a variety of selected brands where customers could browse for products and add to their shopping carts. The aim is to provide our customers with an efficient and curated cosmetic shopping experience.

I.II SCOPE AND SPECIAL REQUIREMENTS

We limit our model within the strict scope of the online store itself. Our primary focus is on the elements that are integral to the store itself, therefore we did not model the end-to-end shopping experience which included the payment processing, the logistics of shipping & delivery and the physical management of inventory and the store's administrator information. We will assume that these supporting infrastructures act as black-boxes and thus are omitted from our design.

II. REQUIREMENT ANALYSIS

II.I DATA REQUIREMENTS

The following entities and their attributes will be stored in tables of a relational database:

Brand: A brand has a brand name(bname) and an address indicating where its official office located. Each product is made from a brand. A brand can be identified by its primary key, bname.

Customer: A customer has a customer ID(cid), an email address, birthday and full name. Customer is identified by their cid. A customer could add/delete items to cart items and they can do payments for their cart. A customer can be referenced by its unique primary key, cid.

Payment: A payment has a payment id(pid), date which the payment has been made, order details and tracking number. Every payment must have at least one order item, however we are unable to model this participation constraint in the relation model. The payment id(pid) is its primary key to identify one payment.

ForSale: A ForSale Product is a type of Product that has a name, volume, price and a overall rating. Different from sample, a forsale product has a price associated with it. A ForSale Product is indicated by its product name.

Sample: A Sample is a type of Product that our store sent to customers as gifts. It has a volume without a price because samples are free gifts. A sample is indicated by its product name.
Products: Product is a weak entity set because multiple products could have the same name so every product needs an additional brand name to uniquely identity itself. Each product is associated with several attributes, including its product name(pname), the category it belongs to (makeup, skincare, etc), its volume size and its stock.
CartItems: CartItems are the items in the shopping cart added by a customer. It could be either composed of Forsale or Sample. A CartItem has quantity of products added and an item_id. A CartItem is uniquely identified by its item_id. Customers could add/delete cart items.
OrderItems: OrderItems are the items that the customer have already paid for. Every order item has to belong to a payment. It has the product paid for (pname, bname), the total cost and quantity for the ordered item and an unique orderItem_id as its primary key. Customers could refund the items they have already ordered.

Notes

- shopping carts are shown on the customers screen but not modeled as an entity set because we could search through the CartItems table by the cid of the customer to output all of their current cart items.
- When customers check out, the cart items are deleted from the CartItems table and we add these paid items to the OrderItems Table. However, this ER model is not able to model the constraint that all order items come from the deleted entries of the CartItems table

II.II RELATIONSHIP REQUIREMENTS

Made from: a product is *made from* a brand. Since Product is a weak entity set, it is a one-to-many relationship because a product needs a brand to identify itself while a brand may not have any products.
IS A: products are either for sale or it is a sample. This IS A relationship is therefore covering and non-overlapping.
Rate: customers could choose to rate a ForSale product by giving it a comment and a rating. It is a many-to-many relationship because it is not mandatory for customers to rate products and for ForSale products to have a rating
Added: a product (ForSale or Sample) may be added as CartItems such that it could be purchased later. It is a many-to-one relationship because not all products need to be converted into cart items but every cart item contains a product.
AddedBy: cart items are added by customers. It is a one-to-many relationship because one cart item must be added by one customer, while a customer could add or not add any cart items.
Refund: customers could apply to *refund* their payments. It is a many-to-many relationship because it is not mandatory for customers to refund and a payment doesn't need to be refunded.
PaidFor: a payment is made for a set of order items. It is a one-to-many relationship because every order item must be paid for and every payment must have at least one order item included.
CheckOut: after customers checked out, cart items become order items. It is a one-to-one relationship because an order item must come from a cart item when a customer checks it out.

III. Functional Requirements

Search: Customers could choose to search by product name, brand name, category or any combination of these parameters. If the input was product name, then the algorithm will output the item(s) in the Product table that have a matching name. If the input was a combination of product name and brand name, then the algorithm will look at these columns in the table and output the matching item(s). If there are multiple items that fit the criteria, then the algorithm will order the items according to their overall rating and output from highest to lowest rating.

search(pname)
search(pname,bname)
search(pname,bname,category)
search(bname)
etc …

Rate: Customers have the freedom to rate a particular product (ForSale type) by giving it a rating (0-5 stars) and a detailed comment. A given customer could rate a product multiple times or delete their previous rating and comment. Once a new rating is added or deleted from a particular product, the algorithm will calculate the overall rating of this product by summing up all the ratings for that product divided by the number of ratings and modify the overall rating attribute in ForSale table.

rate(rating, comment, pname, bname)
deleteRating(cid, dateForRating, pname, bname)

Add To Cart: Products could be added to a customer's shopping cart with an indicated quantity. The algorithm will add an new entry in the CartItems table (if there were no same product added by the same customer to the CartItems table) that records the product added, its cost, the quantity and which customer added this item and generate a unique ID for this cart item. If the product is already added to the cart by this customer, then the algorithm will simply update the quantity in the record.

addCartItem(pname, bname, qty, cid)

Delete Cart Item: Products could also be deleted from a customer's shopping cart. The algorithm will delete an entry in the CarItems table that corresponds to the ID of the cart item if the given quantity equals the recorded quantity for that item. If the indicated quantity is smaller and the recorded quantity, then the algorithm will update the recorded quantity instead of deleting the whole record.

deleteCartItem(cartItem_id, cid, qty)

Check Out: Once a customer is satisfied with the product selection, he/she can check out the items that are currently in the cart and proceed to the payment stage. The total cost of the products will be calculated based on the price and the quantity of the product. Once the items are paid, an unique payment ID and order details are generated and are associated with the paid product. All the cart items will be added to the OrderItems table with the cid of the customer who paid for this product. The cart that is displayed on the screen will be cleared as the corresponding paid cart items being deleted from the CartItems table.

checkOut(item_id, cid)
calculateCost(pname, bname, qty)

deleteCartItem(cartItem_id)
addOrderItem(pname, bname, qty, cost, cid)
generateOrderDetail(cid, date)

Refund: Customers could choose to refund the products they have purchased within 1 month. Sample products could not be refunded since they are already free gifts. Once a customer applied for a payment refund, the algorithm will search the product in the order items. If the returned quantity is equal to the ordered quantity, then the record will be deleted from the OrderItems table and thus the customers could not refund this item anymore. If the returned quantity is smaller than the ordered quantity, the the ordered quantity attribute is updated. We could not enforce customers to add a refund reason since our ER diagram do not store an entity set as administrator, who is responsible for checking the reason.

refund(cid, pid, pname,bname, qty)

III.RELATIONS

III.I ENTITIES:

1. Customers(<u>cid</u>, email, birthday, fullName)
2. Brands(<u>bname</u>, address)
3. Products(<u>pname,bname,</u> stock,volume,category) (bname ref Brands)
4. Sample(<u>pname,bname)</u> (pname&bname ref Products)
5. ForSale(<u>pname,bname</u>, price, overall_rating) (pname&bname ref Products)
6. CartItems(<u>item_id</u>, quantity, bname, pname, cid) (pname&bname ref Products, cid ref Customers)
7. OrderItems(<u>orderItem_id</u>, pid, bname, pname, cid, total_cost, quantity) (pname&bname ref Products, pid ref Payments, cid ref Customers)
8. Payments(<u>pid</u>, trackingNumber, date, orderDetail)

III.II WEAK ENTITIES:

1. Products(<u>pname,bname</u>, stock, volume, type) (pname is the partial key, bname ref Brands)

III.III RELATIONSHIPS:

1. Rate(<u>pname, bname, cid, rate_date</u>, comment, rating) (pname & bname ref products, cid ref customers)
2. Refund(<u>cid, pid, date</u>) (cid ref Customers, pid ref Payments)