Task 1

(15 pts) Provide the problem requirements and the conceptual model in UML for your project. You can reuse the one made on previous projects, but describe the functionalities that you selected to be used as an in-memory key-value storage, (e.g. most viewed products, a shopping cart, current logged-in users, etc).

In today's beauty and skincare industry, it's become a common trend for people to accumulate an overwhelming array of products in pursuit of the perfect skincare routine. With countless options available, from serums and creams to masks and cleansers, consumers often fall into the trap of over-purchasing without considering whether these products are truly suited to their unique skin needs. This impulsive buying not only strains the wallet but can also lead to disappointment as mismatched products may not deliver the desired results. A more informed and minimalist approach, guided by understanding one's skin type and concerns, can be a more effective and cost-efficient way to achieve healthy, glowing skin while reducing clutter in the bathroom cabinet.

Therefore, I would like to create an application that help users to record and manage what skincare products they already own, so that they would always be aware of what they already have before purchasing more. Moreover, I would love to build a platform for users to communicate on their feedback about these products. With the platform, users who are interested in purchasing the products would make a more informed decision.

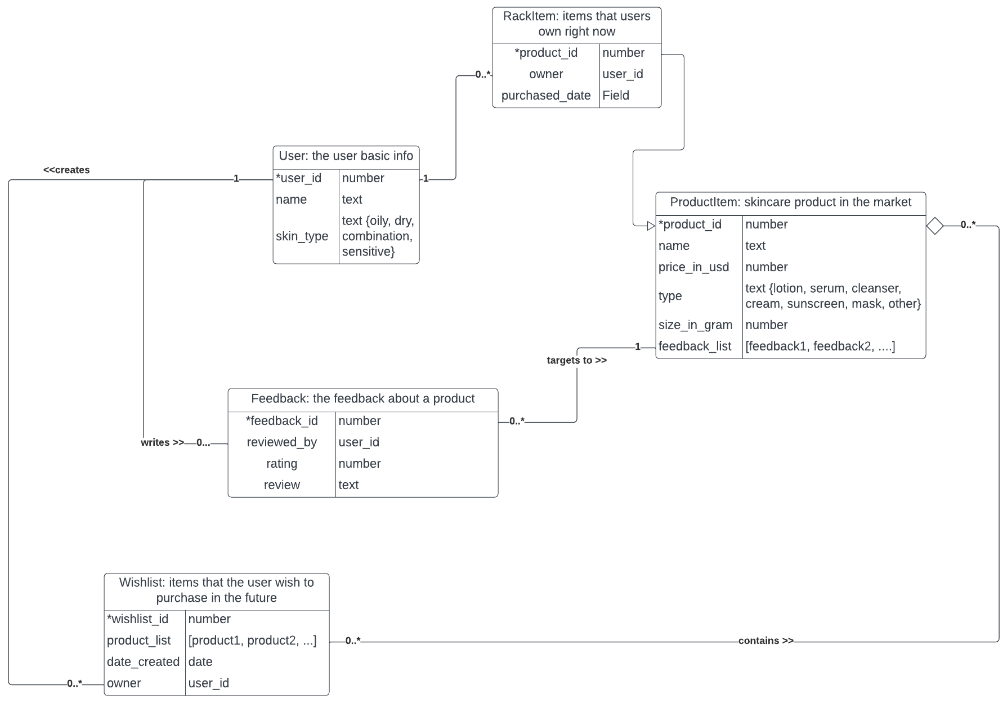
RackItem: items that users own right now
*product_id | number
owner | user_id
purchased_date | Field
User: the user basic info
*user_id | number
name | text
skin_type | text {oily, dry, combination, sensitive}
ProductItem: skincare product in the market
*product_id | number
name | text
price_in_usd | number
type | text {lotion, serum, cleanser, cream, sunscreen, mask, other}
size_in_gram | number
feedback_list | [feedback1, feedback2, ...]
Feedback: the feedback about a product
*feedback_id | number
reviewed_by | user_id
rating | number
review | text
Wishlist: items that the user wish to purchase in the future
*wishlist_id | number
product_list | [product1, product2, ...]
date_created | date
owner | user_id
<<creates
writes >>
targets to >>
contains >>
1
0..*
0...

Figure 1. UML from Project 1

As for the functionalities that I want to select to be used as an in-memory key-value storage:

- Currently logged in user
- The mostly reviewed product
- The mostly in-used (aka, in rack) product

Task 2

(25 pts) Describe the Redis data structures that you are going to use to implement the functionalities you described in the previous point. (e.g. To implement the most viewed products I will use a Redis sorted set with key "mostViewed:userId", product ids as the values and a score of the number of views of the product.). You can use/describe more than one data structure, you will need to implement at least one.

- **Currently logged in user**
  - Key: currentUserId

    - Value: a single string that store the _id field from of the user (in Mongo, it would be user._id)
- **The product that got the most feedbacks**
    - Key: mostFeedbackProduct
    - Value: a sorted set with value of the productId and score of the number of feedback that is related to the product
- **The number of users also uses this product**
    - Key: OnRackProduct:productId
    - Value: the count of user having it on their racks

Task 3

(60 pts) Create a basic Node + Express application that let's you create, display, modify and delete at least one Redis data structure from the ones describe in the previous point. No need to have a polished interface, and you can use the code created in class as a starting point, and/or the code you created for previous projects

Would implement the feature "**the product that is most common on user's rack"**

Plan:

- When user create a product, OnRackProduct:productId would be assigned with value 0. The productId is the product._id field from mongoDB
- When any user adds the product into their rack, OnRackProduct:productId would increase the count by 1
- When any user removes the product from their rack, OnRackProduct:productId would decrease the count by 1
- This value would be displayed on the product card in the rack page as "XX other users is also using this product"