Project 2

Overview

In this practicum you will modify the database you built for Project 1 to adjust it to a document based database (Mongo). You can reuse the project description, adjusting the logical model and choosing the main collections to use. Finally you will modify the implementation from Project 1 to make it work with MongoDB and Node.

Task 1

(5 pts) Provide the problem requirements and the conceptual model in UML for your project. You can reuse the ones made in Project 1.

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.

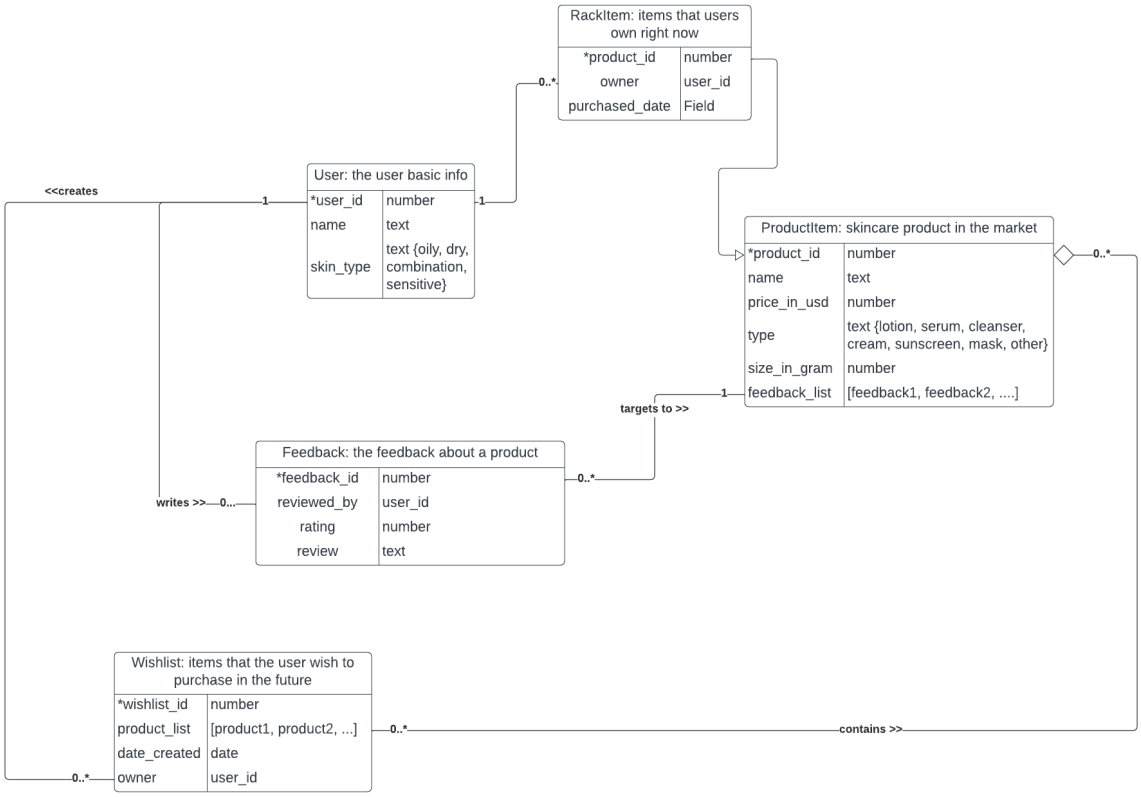
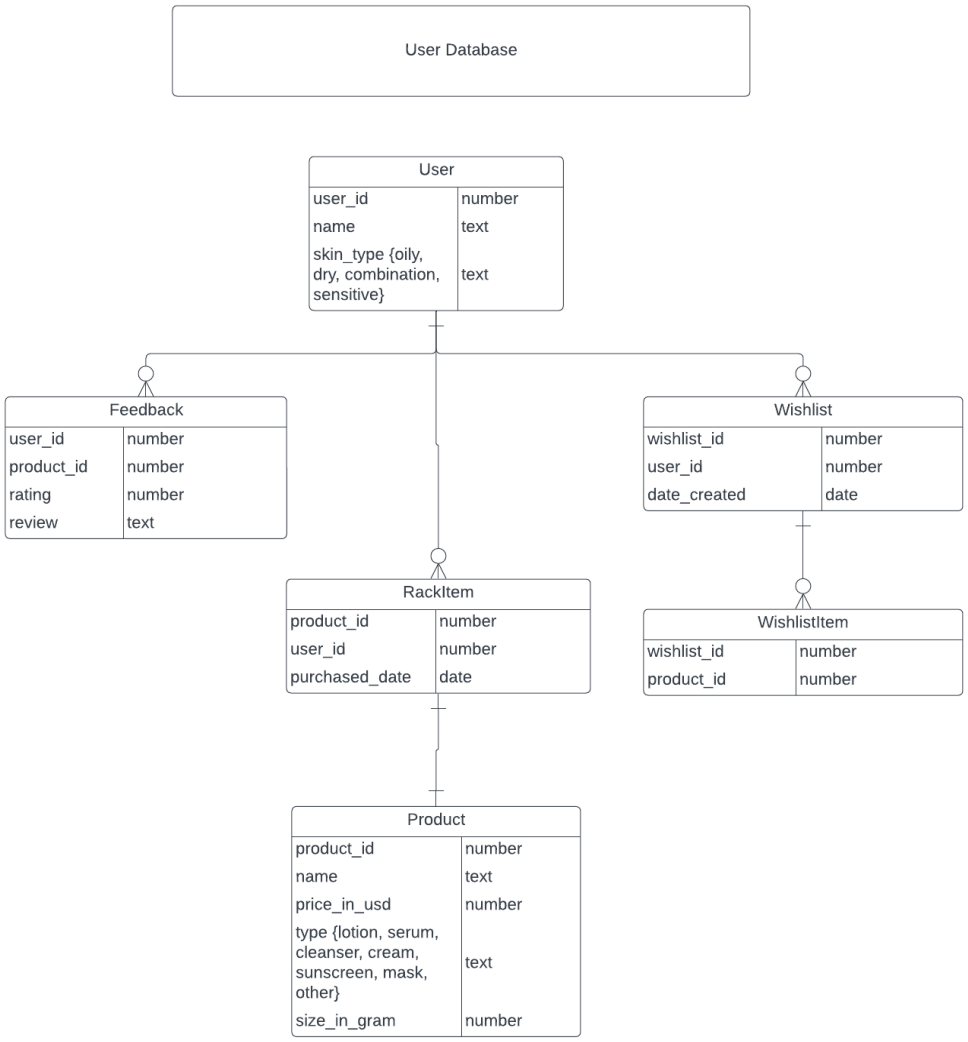


Figure 1. UML from Project 1

Task 2

(*15 pts*) Adapt the Logical Data model from your Project 2 to have hierarchical tables. This is, main (root) tables from which all the other tables relate to. This main tables will become later your Mongo Collections. From your main tables you can have aggregation/composition, one to many and many to many relationships.



Note that *Product* would be separated into another collection to improve the usability.

Task 3

(*10 pts*) From this logical model define the main Collections (Documents/Tables) you will be using in your Mongo Database. Provide a couple of JSON examples of these objects with comments when necessary. Think about a document that you will give to another database engineer that would take over your database.

User collection

```
{
user: {
    user_id<number>,
    name <text>,
    skin_type<text>,
  },
  wishlists:[
    {
      wishlist_id<number>,
      date_created<number/date>,
      items: [
        product_id<number>,
      ]
    }
  ],
  rack:[
    {
      purchase_date<number/date>,
       product_id<number>,
```

```
    },
    ...
  ],
  feedbacks:[
    {
      rating<number>,
      review<text>,
      product_id<number>,
    }
  ]
}

```

Product Collection

```
{
        product_id<number>,
        name<text>,
        price_in_usd<number>,
        type<text>,
        size_in_grams<number>
}

```

Task 4

Populate the tables with test data. Include in your repository a dump file that can be use to regenerate your database, and the instructions on how to initialize it

See the repo and readme.md

Task 5

(*30 pts*) Define and execute at least five queries that show your database. At least one query must contain an aggregation, one must contain a complex search criterion (more than one expression with logical connectors), one should be counting documents for an specific user, and one must be updating a document based on a query parameter (e.g. flipping on or off a boolean attribute for a document, such as enabling/disabling a song)

Aggregation

Find product that is a cream and has size less than 50.

```
5200_proj2.Product.aggregate([
  { $match: { type: 'cream', size_in_grams: { $lt: 50 } } },
  {
    $group: {
      _id: null,
      averagePrice: { $avg: '$price_in_usd' },
    },
  },
  {
    $project: {
      _id: 0,
      averagePrice: 1,
    },
  },
])
```

complex search criterion

Find the products that:

1) Dry skin user has on their rack
2) Price less than 50

```
5200_proj2.User.aggregate([
  {
    $match: {
      "user.skin_type": "dry",
    },
  },
  {
    $unwind: "$rack",
  },
  {
    $group: {
      _id: "$rack.product_id",
    },
  },
  {
    $lookup: {
      from: "Product",
      localField: "_id",
      foreignField: "product_id",
      as: "products",
    },
  },
  {
    $unwind: "$products",
  },
  {
    $match: {
      "products.price_in_usd": { $lt: 50 }
    },
  },
  {
    $replaceRoot: {
      newRoot: "$products",
    },
  },
]);
```

Counting documents for an specific user

Count the number of feedbacks that has rating higher than 3.

```
5200_proj2.User.aggregate([
   {
       $unwind: "$feedbacks"
   },
   {
       $match: {
           "feedbacks.rating": { $gt: 3 }
       }
   },
   {
       $group: {
           _id: null,
           feedbackCount: { $sum: 1 }
       }
   }
]);
```

Updating a document based on a query parameter

Update the purchased date for a given product by a given user.

```
5200_proj2.User.updateOne(
   {
       "user.user_id": 1,
       "rack": {
           $elemMatch: {
               "product_id": 74
           }
       }
   },
   {
       $set: {
           "rack.$.purchased_date": "7/13/2020"
       }
   }
);
```

Another query

Find all serum products

```
5200_proj2.Product.aggregate([
  {
    $match:
      /**
       * query: The query in MQL.
       */
      {
        type: "serum",
      },
  },
]);
```

Task 6

(25 *pts*) Create a basic Node + Express application that let's you create, display, modify and delete at least two of the tables. One of the tables can be the users table. 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 Project 1

Minimum Requirements & Grading Rubric

- Conceptual Data Model in UML (Class Diagram) Diagram must be drawn using a tool.

Submission

Submit a Github Repository containing:

A. The requirements document as a PDF.
B. UML Class Diagram as an embedded JPG/PNG.
C. ERD as an embedded JPG/PNG and URL to its LucidChart diagram.
D. Definition of Documents/Collections/Tables as example JSON objects.
E. Initialization files for the database containing the mockup data in CSV or Extended JSON format as well as instructions on how to initialize the database.
F. The code of your basic application with a proper README file