Report

Milestone 3

1. Implementation features

I have used ORM Flask-MongoAlchemy library to interact with MongoDB database.

2. Summary of how data is structured in NoSQL

In MongoDB database I have 3 collections: MongoItem, MongoOrder and MongoUser.
Structure of collections:

MongoItem:

```
{
        “id”: int,
        “type”: string,
        “price”: float,
        “description”: string,
        “amount_available”: int,
        “build_url”: string
}
```

MongoOrder:

```
{
        “id”: int,
        “date”: date,
        “status”: string,
        “delivery_method”: string,
        “items”:
         [
                {
                        “amount”: int,
                        “item”: MongoItem
                }
        ]
}
```

MongoUser:

```
{
        “id”: int,
        “name”: string,
        “surname”: string,
        “email”: string,
        “password”: string,
        “cart”:
        [
                {
                        “amount”: int,
                        “item”: MongoItem
                }
        ],
        “order”:
        [
                MongoOrder.id
        ]
}
```

3. Reasons for designing collections / documents the way they are.

Instance of MongoUser collection has all cart items inside since user access to the cart happens quite often, so it is more convenient to have cart items inside user’s object. User also contains list of all performed orders, but these are only references, because if a user performs multiple purchases it would be inefficient to store all his/her orders inside the users’s object (the object would become huge and will take a lot of space).

Since for the main use case an overview of all orders, users that made these orders and items that the orders contain is needed, storing orders inside MongoUser collection is efficient, because it is easy access to corresponding MongoOrder object having its id.

4. How data should be indexed

I haven’t implement indexes in my app, because for my main use case they were not necessary. But for example if one would need to find out which items are more frequently bought, it would be useful to make an id field of MongoItem subdocument in MongoOrder collection as index to enable faster iteration over items in orders.

5. Aggregation pipeline for the reporting use case

```
db.users.aggregate([
        {
                $lookup:
                {
                        from: “MongoOrder”,
                        localField: “order”,
                        foreignField: “id”,
                        as: “orders”
                }
        }
])
```