Property Finder Go Bootcamp - Final Project

Through this project, we expect you to develop a basket service. The service will be a REST API developed in Go.
Customers will be able to purchase existing products. Use cases such as creating, updating, deleting products or creating users etc. for admin won't be developed. You can just insert necessary data into the database using SQL for customer use cases to run.
No UI will be developed. You can make your tests using REST test tools such as Postman.

The functions of this service are as follows;

1. **List Products**
   - Users should be able to list all products.
2. **Add To Cart**
   - Users can add their products to the basket and the total of the basket changes accordingly.
3. **Show Cart**
   - Users can list the products they have added to their cart and total price and VAT of the cart.
4. **Delete Cart Item**
   - Users can remove the products added from the cart. Notice removing an item may change discount.
5. **Complete Order**
   - Users can create an order with the products they add to their cart. There is no need to implement any payment mechanism. You can assume that when an order is completed it is fully paid.

Some business rules

1. Products always have price and VAT (Value Added Tax, or KDV). VAT might be different for different products. Typical VAT percentage is %1, %8 and %18. So use these values for your products.
2. There might be discount in following situations:
   a. Every fourth order whose total is more than given amount may have discount depending on products. Products whose VAT is %1 don't have any discount but products whose VAT is %8 and %18 have discount of %10 and %15 respectively.
   b. If there are more than 3 items of the same product, then fourth and subsequent ones would have %8 off.

c. If the customer made purchase which is more than given amount in a month then all subsequent purchases should have %10 off.
d. Only one discount can be applied at a time. Only the highest discount should be applied.

Expectations

1. Organize your code around packages.
2. Observe cohesion and coupling.
3. Apply clean code and clean architecture principles.
4. Apply Go idioms.
5. Create your own errors if needed.
6. “Given amount” in requirements can change so make it changeable.
7. Write unit tests for business logic only.
8. Apply well-known REST API patterns.
9. Use a well-known relational database such as MySQL or PostgreSQL.