System Structure:

The system structure is kept pretty close to the test requirement in that each product has one category & each category has a parent until the chain ends. For clarity's sake, the invoice price is replaced with product price & attached to each product for easy manipulation & also to keep structure minimal to understand. Hence, we don’t need to pass invoice price nor user id because price is delivered along product & instead of user id, we pass jwt bearer token in Authorization header i.e. Postman.

The discount is attached either to the product or to category. The product or category may or may not have a discount that can also be seen in the fixture/seed files inside the database directory.

The app is dockerized & accompanied with *docker compose* file as requested by the test but due to the nature of docker network & node/typescript, the seeding files cannot be run automatically so to seed the data, we need to login to nodejs docker container & then execute npm command manually from there. Although, it requires only following two commands to achieve seeding but this flow could be automated given more time was available i.e.

```
docker exec -it aib_server /bin/bash
```

```
npm run seed
```

The flow of the system is that, we provide product code from client/postman & server finds that product along the category & then look for the discount in either product (first) or in category. If the discount is not available, then we get all parent categories with reference to the product’s category & start finding discount one-by-one no matter how deep the nesting goes; otherwise we return -1.

In order to easily manage testing operations i.e. request/response, price, discount changes, the postman collection & **pgadmin** app is also included to make interacting with the app smooth.

One little caveat is that, one category cannot belong to two products because of the 1-to-1 relationship b/w them but it can be easily improved without causing any change to the desired outcome.