Specification

A website where there are 6 entities:

1. A single seller’s inventory (named Head Office)
2. Many customers (named Restaurants). Each of them has its own inventory and get products from the seller.
3. Employees of the restaurants who make orders from the single seller (Head Office) and manages their own restaurant’s inventory.
4. Employees of the single seller who can accept/reject all the orders from restaurants and manage orders.
5. Orders
6. Products

A diagram of a restaurant

Description automatically generated

(The term managing inventory means view/add/edit/remove items)

[Database Model](https://lucid.app/lucidchart/6c8296d3-b7d4-42cb-92b0-0fbc1c9b0bd0/edit?viewport_loc=-980%2C-189%2C2313%2C1217%2C0_0&invitationId=inv_e3b5a2c9-9df4-4018-813e-f388d048cbc3) on LucidChart

[UI on Figma](https://www.figma.com/file/mZZnP92MDxxiVa8qzxSbC9/GroupProject?type=design&node-id=0%3A1&mode=design&t=JVhgPiMsGYV3FbHD-1)

Business Logic

When a HO employee log in the website, he can:

1. manage the HO inventory
2. view every restaurant’s inventory.
3. View / approve / disapprove orders from all restaurants

When a restaurant’s employee log in the website, he can:

1. View HO inventory as “browsing the store.”
2. And making orders from there (one product each time )
3. View its orders that has been successfully made.

When a restaurant’s employee browses the HO inventory and then make an order of a product of some quantity,

1. The frontend JS check the quantities (quantity in store > quantity asked) before sending the post request of creating order.
2. Server checks the quantities again with fetched data from Database.
3. Server reply with signal of success or fail to create the order.

Details about ‘Orders’

Apart from other self-explanatory fields such as OrderID, CreationDate, and other descriptive text fileds , there are two properties that need to be clear.

1. one Product in one order & its Quantity

ProductID is clearly a foreign key referencing an item in table Products. There’s only one ProductID field meaning an order can only hold one product of some quantity.

(the reason for not implementing a cart where customers can buy various products in one order is that ………//TODO…….)

1. Order\_State

Pending:

Initial state when an order is made, waiting for a HO employee’s approval or disapproval.

Approved:

this is the final state of a successful order. There’s no states like ‘in shipmen, completed’ as there’s no related business logic yet.

Disapproved:

The other final state.

Stage 2 spec

Marking faulty items in a restaurant’s inventory

Messaging Module

MailBox / Issue Tracking: an 'conversations' table in database where every conversation between HO and a restaurant is stored. When an employee log in, the website loads all the related unfinished conversations then he can add more messages / mark Complete a conversation. It's the same thing as Issue Tracking on Github.