Lecture26 :- Zepto LLD

(Inventory management)

functional Requirements:

① we should be able to manage inventory (Add / Remove items)

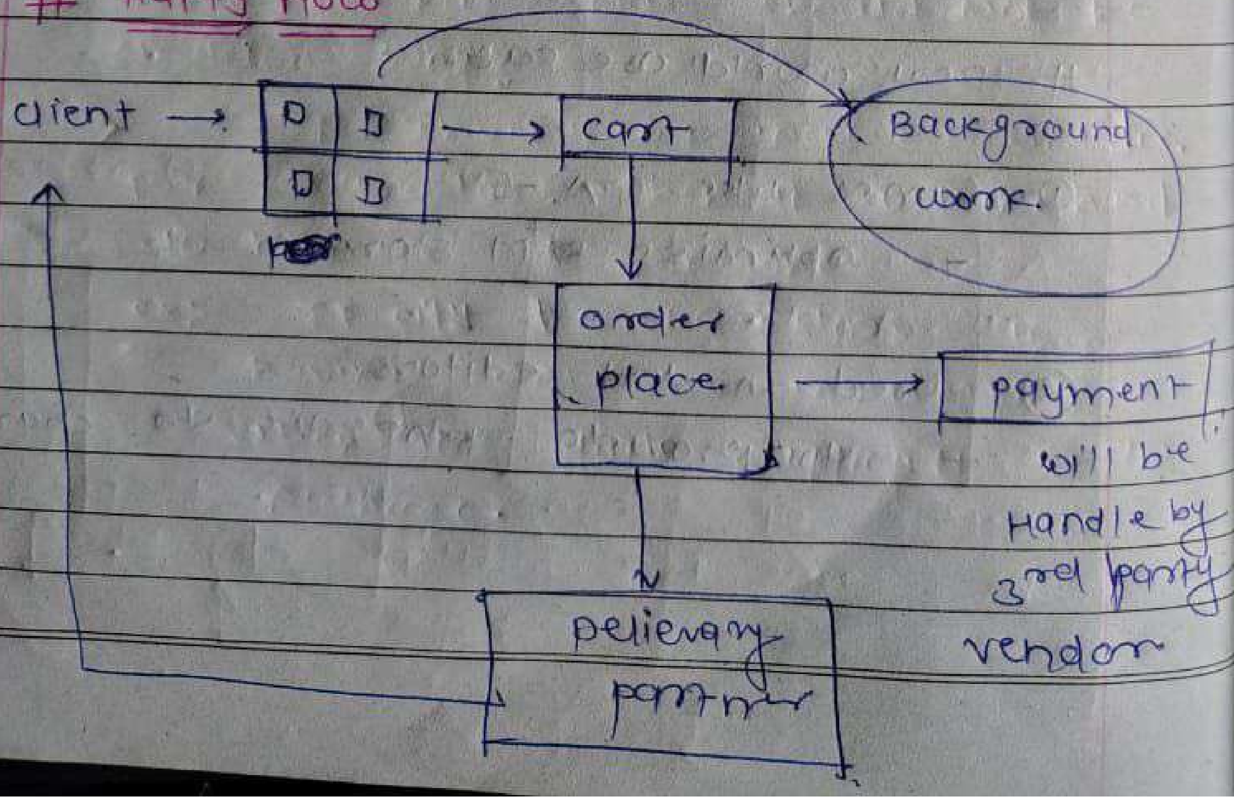
② we should have Replenish strategies.(Threshold weekely), and should be scalable.

③ we can have multiple inventory store (like DB inventory store etc). and can be further extended.

④ user should be able to see items from all the darkstores closer to him/her. (5km).

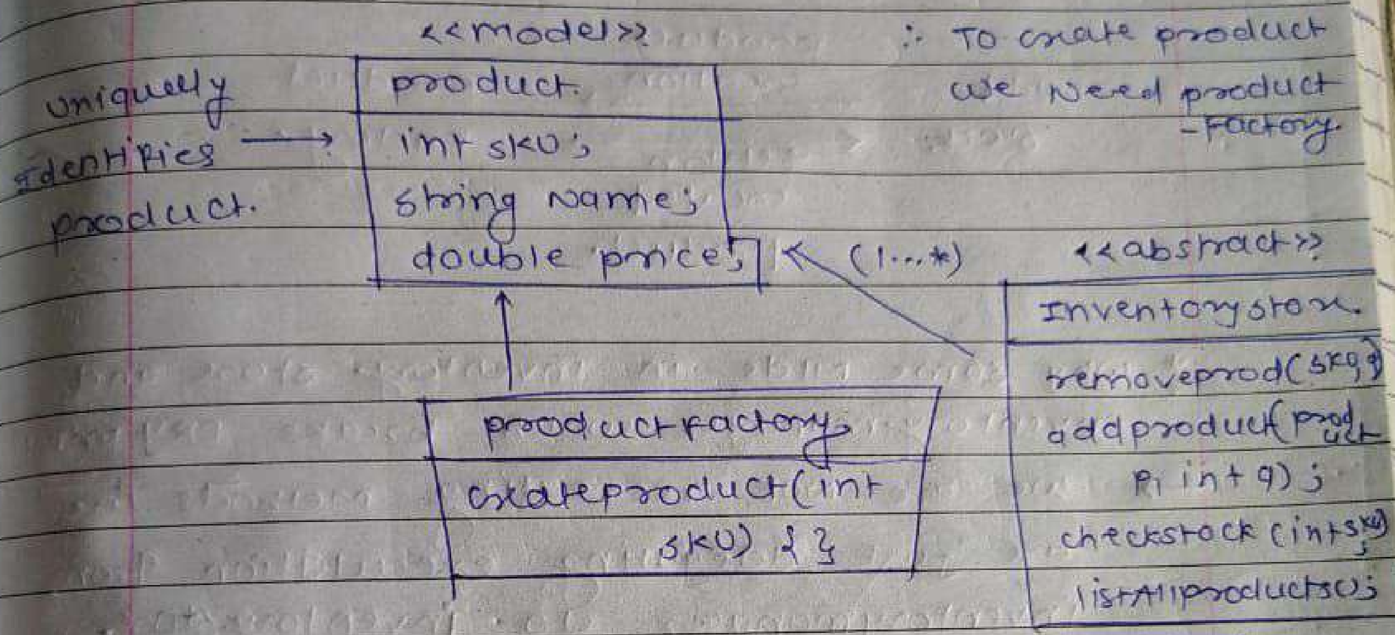
⑤ If one darkstore can't fulfill order oneorder can be split into multiple DS fulfilled by multiple Delievary agents.

Happy Flow



UML Diagram for Zepto.

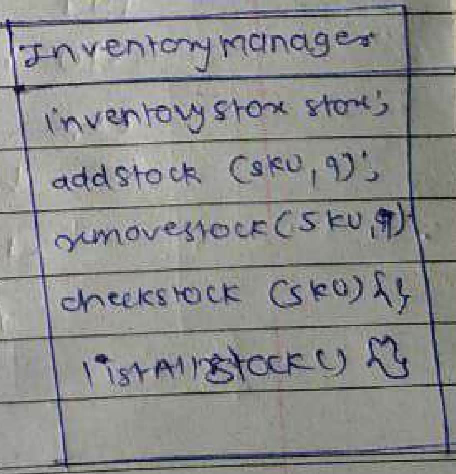
- we use bottom-up approach.
- Basic class to build in this application will be Zepto.



: we Need to store multiple products creating Inventorystore

: Now, in Darkstore, there will be inventory store but we don't directly link them as it will make them tightly coupled.

: Therefore we will include inventory manager which will be midman beth inventorystore and Darkstore.



:. Darkstore performs add or remove item actions. that could not be directly link to inventorystore.

Inventorymanager performs does not care about product It focus on stock.

$$\begin{tikzcd}
\text{add stock} \arrow[r, "\text{product creation}", shift left] & \text{product factory} \arrow[l, "\text{return created product.}", shift left]
\end{tikzcd}$$

* why some mtds in inventory store and inventorymanager which breaks DRY principle

→ we are not recreating methods we are just delegating executeition from inventorymanager to inventory store.

Inventorymanager class should not be singleton why?

→ If Inventorymanager is unique for darkstore then it will be hard to manage

ex:- If city has onlyone darkstore then having only one inventorymanager then it's quite difficult to manage.

pattern choose

- Inventory store, inventory manager and DB inventory store class Use Bridge Design pattern.
    - Inventory store (implementation) and inventorymanager (abstraction).
    - DB inventory will store these data in Database.
    - darkstore manager created to take user Nearby location & check Nearby store.

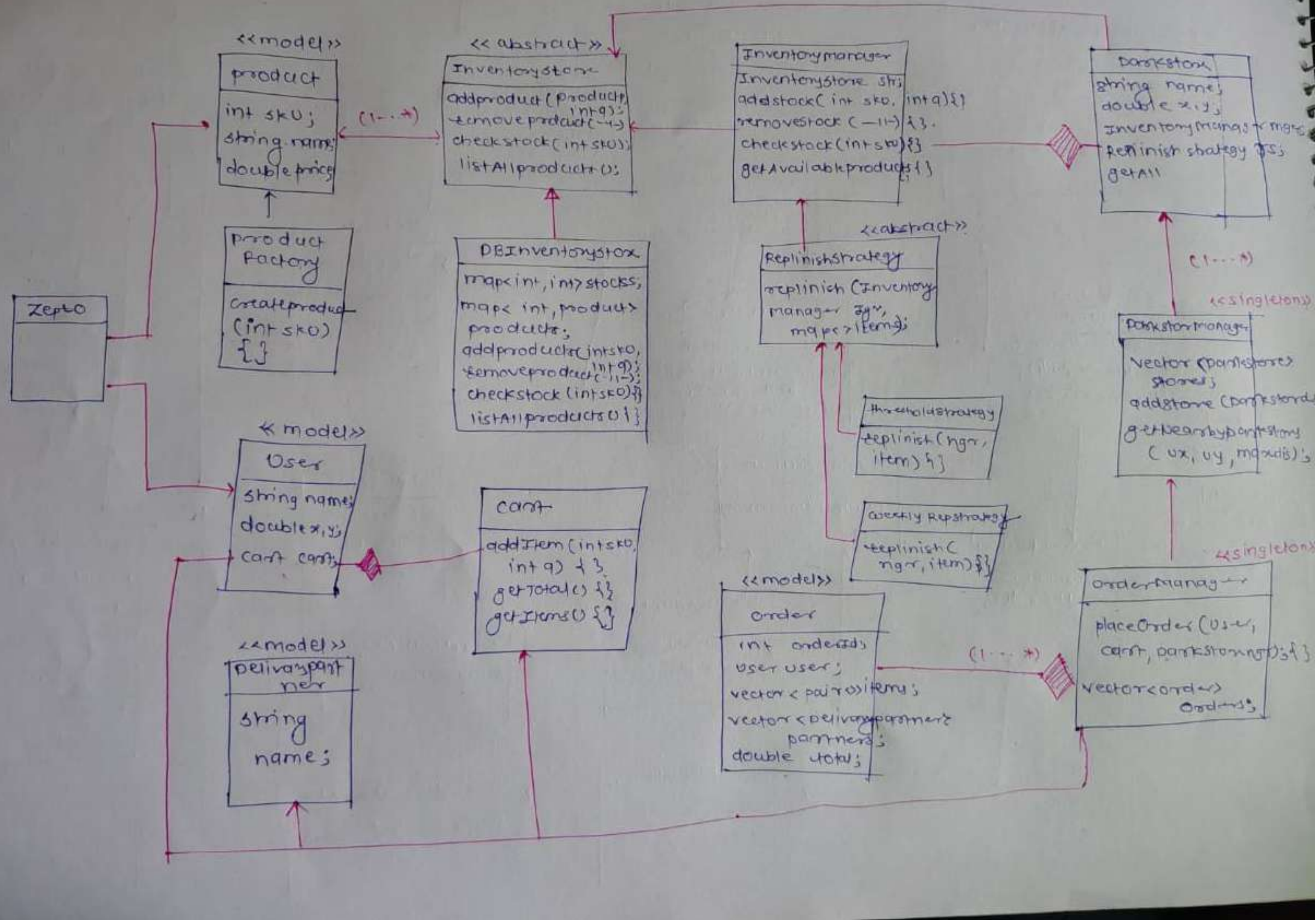
- placeorder (user, cart, darkstoremanager)
  {}

- placeorder.
  ① get Nearby store
  ② fetch cart items.
  ③ check if 1st store can fulfill.
  complete order → if true
  → assign delivary partner
  else
  ↳ multiple delivary partner

further Extensions

- Add Rider mapping Algorithm.
- Add payment / Coupon mechanism.
- make the code modular.

Zepto :- Inventory management (UML)



<<model>>
product
int sku;
string name;
double price;
(1-..*)
<< abstract >>
Inventorystore
addproduct(product, int q);
removeproduct(--,-);
checkstock(int sku);
listAllproducts();
Inventorymanager
Inventorystore st;
addstock(int sku, int q){}
removestock(—11—){}.
checkstock(int sku){}
getAvailableproducts(){}
Darkstore
string name;
double x,y;
Inventorymanager mgr;
Replinish strategy rs;
getAll
product Factory
createproduct (int sku) {}
DBInventorystore
map<int, int> stocks;
map< int, product> products;
addproducts(int sku, int q);
removeproduct(-,-);
checkstock (int sku){}
listAllproducts(){}
<<abstract>>
Replinishstrategy
replinish (Inventory manager mgr, map<>items);
(1...*)
<<singleton>>
Darkstoremanager
vector <Darkstore> stores;
addstore (Darkstore ds)
getNearbyDarkstores (ux, uy, maxdis);
Zepto
<< model>>
User
string name;
double x,y;
cart cart;
cart
addItem (int sku, int q) {}
getTotal() {}
getItems() {}
thresholdstrategy
replinish(mgr, item){}
weekly Repstrategy
replinish(mgr, item){}
<<model>>
Deliverypartner
string name;
<<model>>
order
int orderId;
User user;
vector < pairs> items;
vector <Deliverypartner> partners;
double total;
(1...*)
<<singleton>>
OrderManager
placeOrder (user, cart, darkstoremgr);{}
vector<order> Orders;