Lecture 26: Building Zepto (Inventory Management)

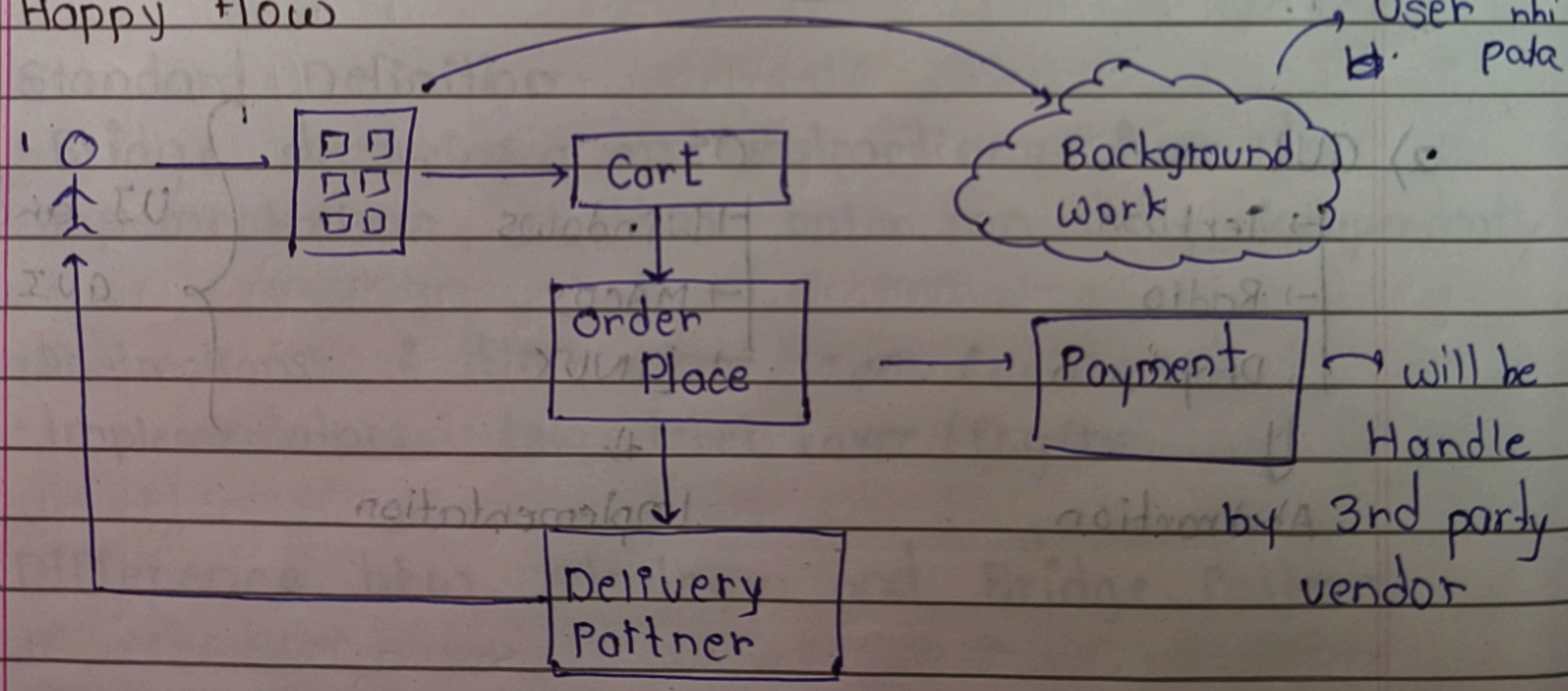
Page No.
Date

Functional Requirements

1. We should be able to manage inventory (Add/ Remove items)
2. We should have Replenish strategies (Threshold, Weekly), and ~~th~~ should be scalable.
3. We can have multiple Inventory store (like DB Inventory store, etc.) and can be further extended.
4. User should able to see items from all the Darkstores closer to him/her (5 km)
5. If one Darkstores cannot fulfil order, one order can be split into multiple DS, fulfilled by Multiple Delivery Agents.

uniquely identify ← product (it is productId)

Happy Flow



→ No
but
mo

→ Th
wh
an

UML Diagram for Zepto

1. Hum bottom-up approach use karege.
2. The very basic class to build in this application will be Zepto.

• Inv
ma

simple class having only variables and getter-setter

<<model>>

Product
int sku;
String name;
double price;

uniquely identify product (it is productId)

(1..*)

Product Factory
CreateProduct (int sku) {..}

<<abstract>>

Inventory Store
removeProd (sku, q);
addProduct (Product p, int q);
checkStock (int sku);
listAllProducts ();

- To create product we need product Factory
- We need to store multiple products :: creating Inventory store

⟶ Now, in Dark store there will be inventory store but we can't directly link them as it will make them tightly coupled.

⟶ Therefore we will include Inventory Manager which will be mid man between Inventory store and Dark store.

Inventory Manager
InventoryStore store;
addstock (sku, qK..};
removestock (sku, qK..}
checkstock (sku) <>
listAllstock<() <..}

- Inventory Manager ka kaam hai inventory ko manage karna.

- Dark store ka kaam hoga ki store mein kuch add karna hai yaa kuch remove karna ho so ~~mat~~ that Dark store could not directly talk to Inventory store.

→ Inventory manager ko product se mtlb nahi hai stock se matlab hai.

→ add stock —Product Creation→ Product Factory
←return created Product

Doubt :
Why some methods in inventory store and inventory manager which breaks DRY PRINCIPLE

→ Humne methods ko recreate nahi kiya hai yeh methods sirf deligate karne ka kaam kar rahe hai inventory store ko.

→ Inventory Manager ka kaam hai Inventory store se kaam karwana.

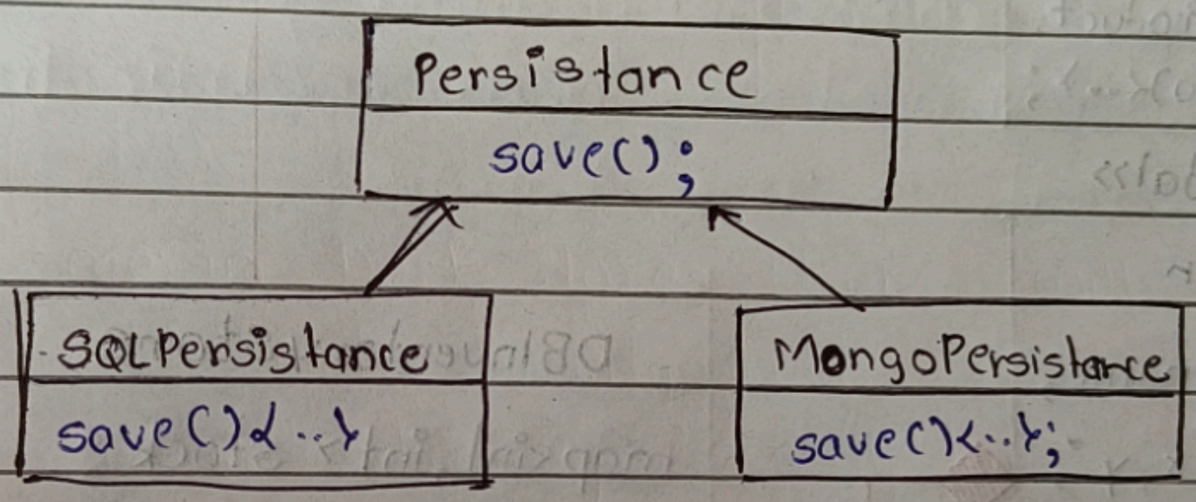
Que. Inventory Manager class should not be Singleton Why?
If Inventory Manager is unique for Dark store then it will be hard to manage. Ex: agar ek city mein 50 Dark store ho and ek hi inventory manager ho toh it's quite difficult to manage.

Pattern choose

- Inventory store, Inventory Manager and DB Inventory store class use Bridge Design Pattern.

- Inventory store wali hierarchy low level hierarchy hai implementation and inventory manager high level ~~big~~ hierarchy matlab abstraction

- DB Inventory store matlab ye chizze Database mein store karega but abhi toh ~~oo~~ discuss kiya ki maps mein store karege to aap DB mein kaise integrate kar sakte ho using persistence class



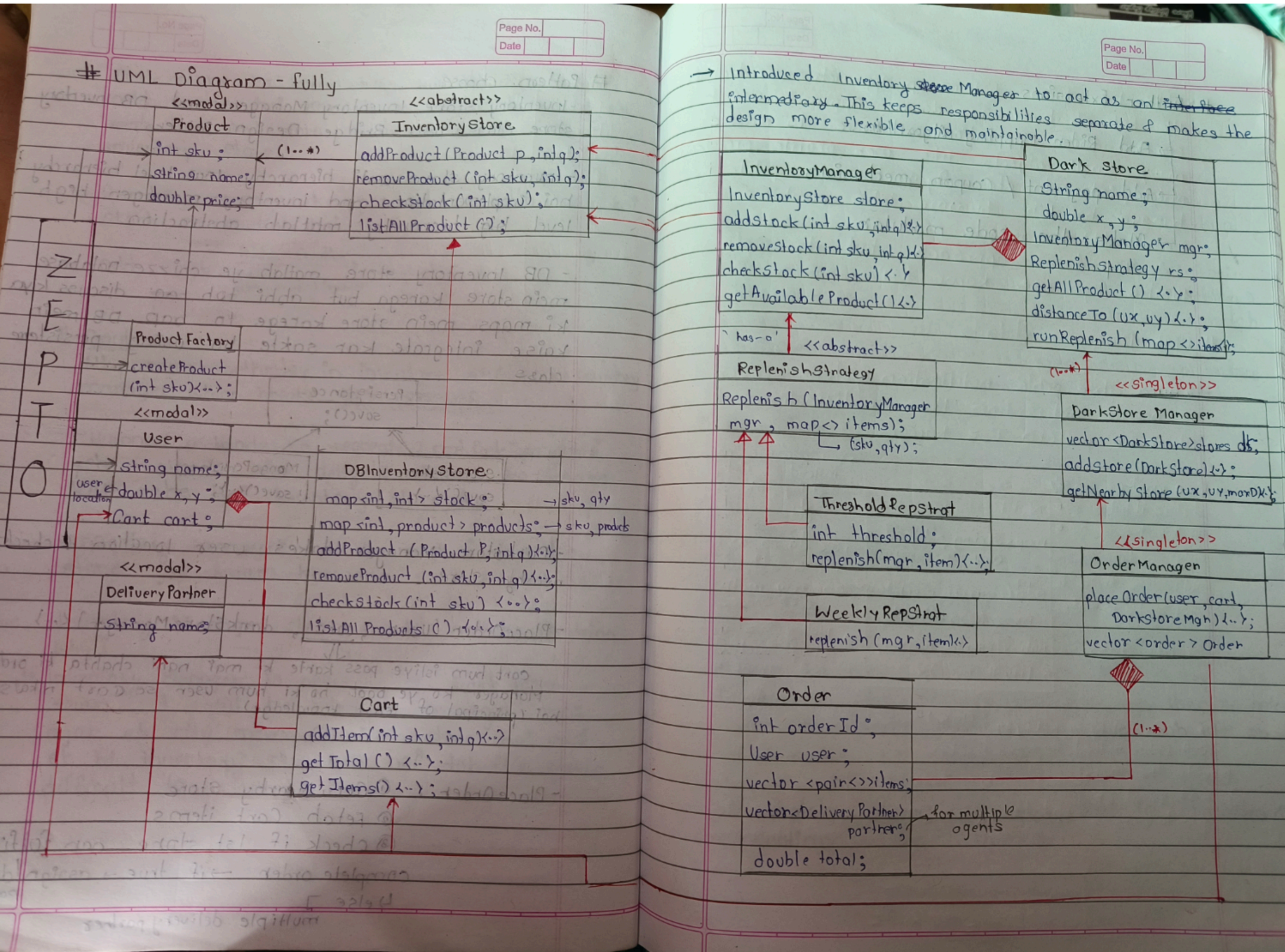
- Darkstore Manager takes user location & check nearby store.

- PlaceOrder (user, cart, darkstoreManager){..}

cart hum isliye pass karte ki mai nahi chahta ki order Manager ko ye baat ho ki hum user se cart nikal skte hai (principal of least knowledge)

- PlaceOrder : ① Get Nearby store
② Fetch Cart items
③ Check if 1st store can fulfill complete order → if true → assign 1 deliver partner
↳ else] multiple delivery partner

UML Diagram - fully



→ Introduced InventoryManager to act as an ~~interface~~ intermediary. This keeps responsibilities separate & makes the design more flexible and maintainable.

Further Extensions

- Add Rider Mapping Algorithm
- Add payment / Coupon mechanism
- Make the code modular.