[Shared Kitchens 2](#_Toc51001375)

[Assumptions 2](#_Toc51001376)

[Build and Run 2](#_Toc51001377)

[Order Age Calculation 2](#_Toc51001378)

[Some Math regarding the Order Value and Order Age: 2](#_Toc51001379)

[Max age of order based on the OrderValue: 3](#_Toc51001380)

[Architecture 5](#_Toc51001381)

[Design 6](#_Toc51001382)

[Models used by the Storage Service 7](#_Toc51001383)

[Order Life Cycle 8](#_Toc51001384)

[Kitchen 8](#_Toc51001385)

[Storage 8](#_Toc51001386)

[Normal Shelves (Hot/Cold/Frozen) 8](#_Toc51001387)

[Overflow Shelf: 9](#_Toc51001388)

[Check Expired Orders Worker – Normal Shelves (Hot/Cold/Frozen): 10](#_Toc51001389)

[On New shelf space available event received – Overflown Shelf: 11](#_Toc51001390)

[Overflown Shelf Expired Orders Checking Worker – Overflown Shelf: 12](#_Toc51001391)

[Dispatch (Courier) 13](#_Toc51001392)

[Supervisor 13](#_Toc51001393)

[Report 14](#_Toc51001394)

[Final Note 14](#_Toc51001395)

# Shared Kitchens

## Assumptions

* Orders input is valid and the temperatures allowed are Hot, Cold, Frozen (made extensible though)
* Courier would arrive only after an order is ready by the kitchen and stored in the shelf
* Orders are not duplicated
* Decay rate is valid number
* (1 + (factor \* decay rate) ) > 0 always holds good

## Build and Run

**Please refer ReadMe file**

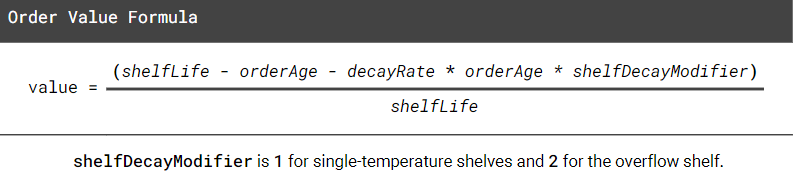
## Order Age Calculation

### Some Math regarding the Order Value and Order Age:

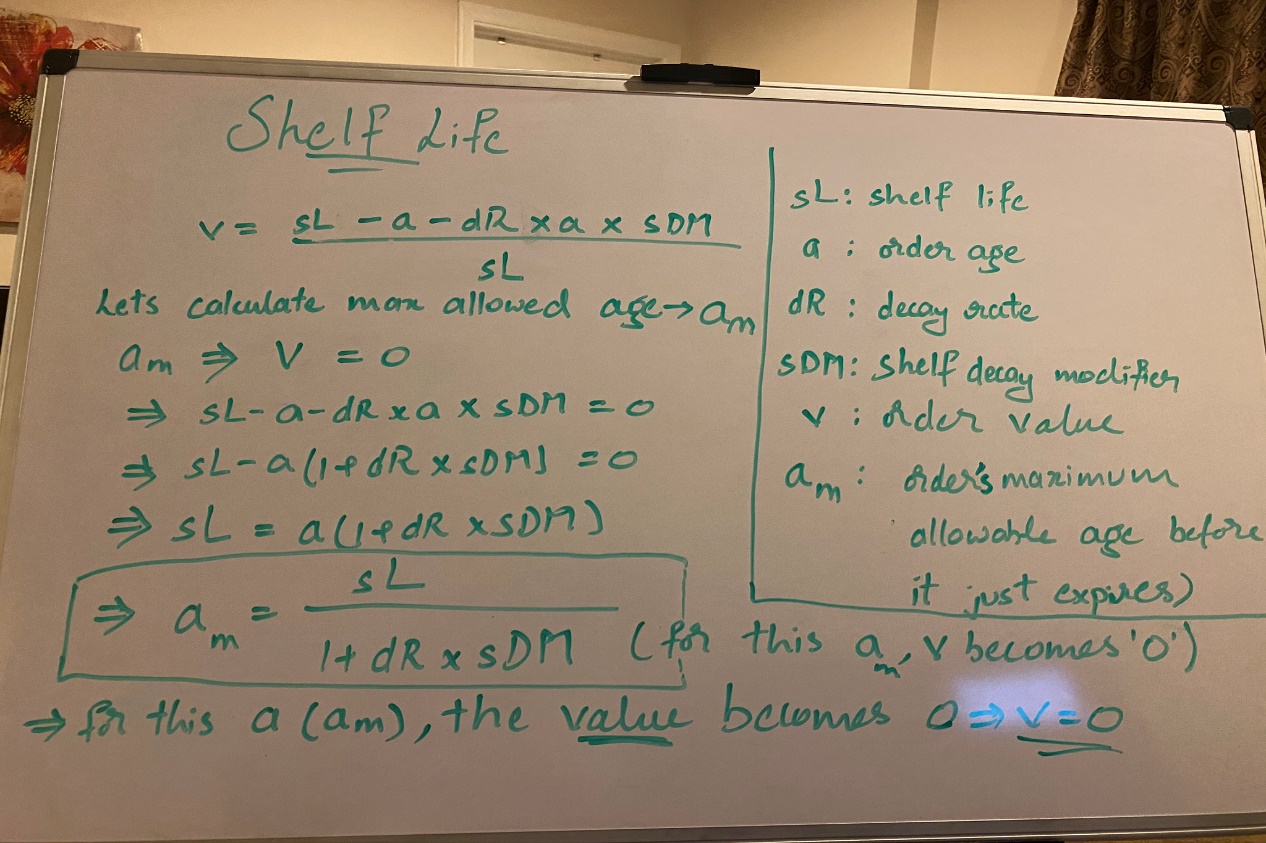
The maximum allowable age of an order is calculated based on the given params solved in the image below:

The given max age of an order for which the *value* becomes 0 is shown below:

Given equation:



#### Max age of order based on the OrderValue:



Based on this, an order is termed ***expired*** if:

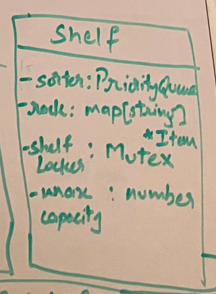
currAge: Current Age of an Order

maxAllowedAge: Maximum allowed age for an order based on above formula (am)

***currAge – maxAllowedAge >= 0***

***where currAge = Current Time - Order.CreatedTimeS*** *All in seconds units of time*

This maxAllowedAge could be used to set priority on prepared orders. Using a priority queue (using min heap) based on this priority (am), will help know which order is expiring soon or already expired in O(1) complexity. The push operation costs O(logN) and retriving an element costs O(1) with the help of a map (used as reverse index for lookup purposes) (Check **Shelf** **data structure** in the design diagram)

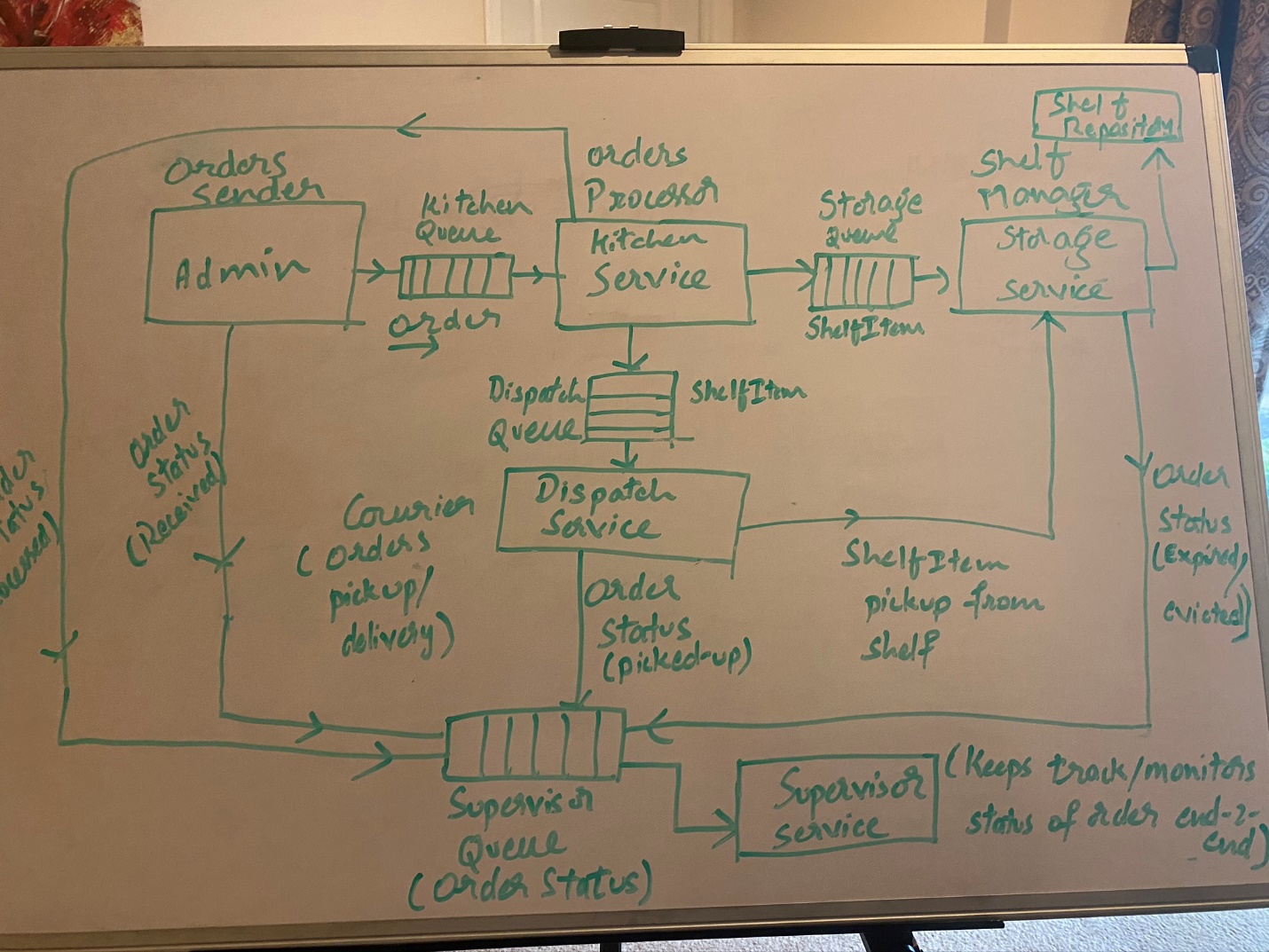


# Architecture

Divided into below components

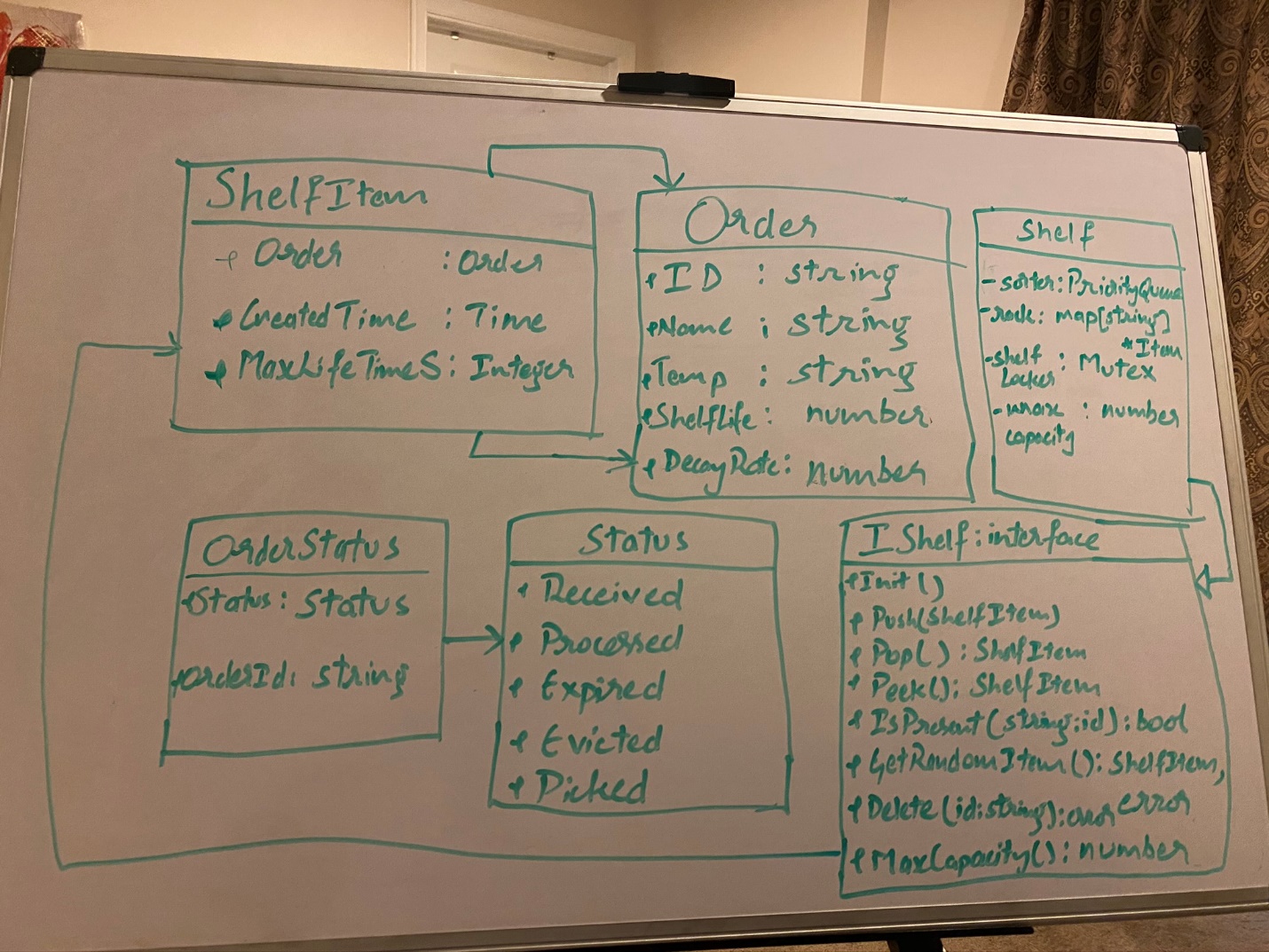
* Admin (Orders Receiver)
* Kitchen Service(Orders Processing)
* Storage Service(Shelves Manager)
* Dispatch Service (Courier)
* Supervisor Service(Supervision of order end-to-end)

The components, interaction, communication channels, messages being shared and status reporting shown in the below architecture diagram



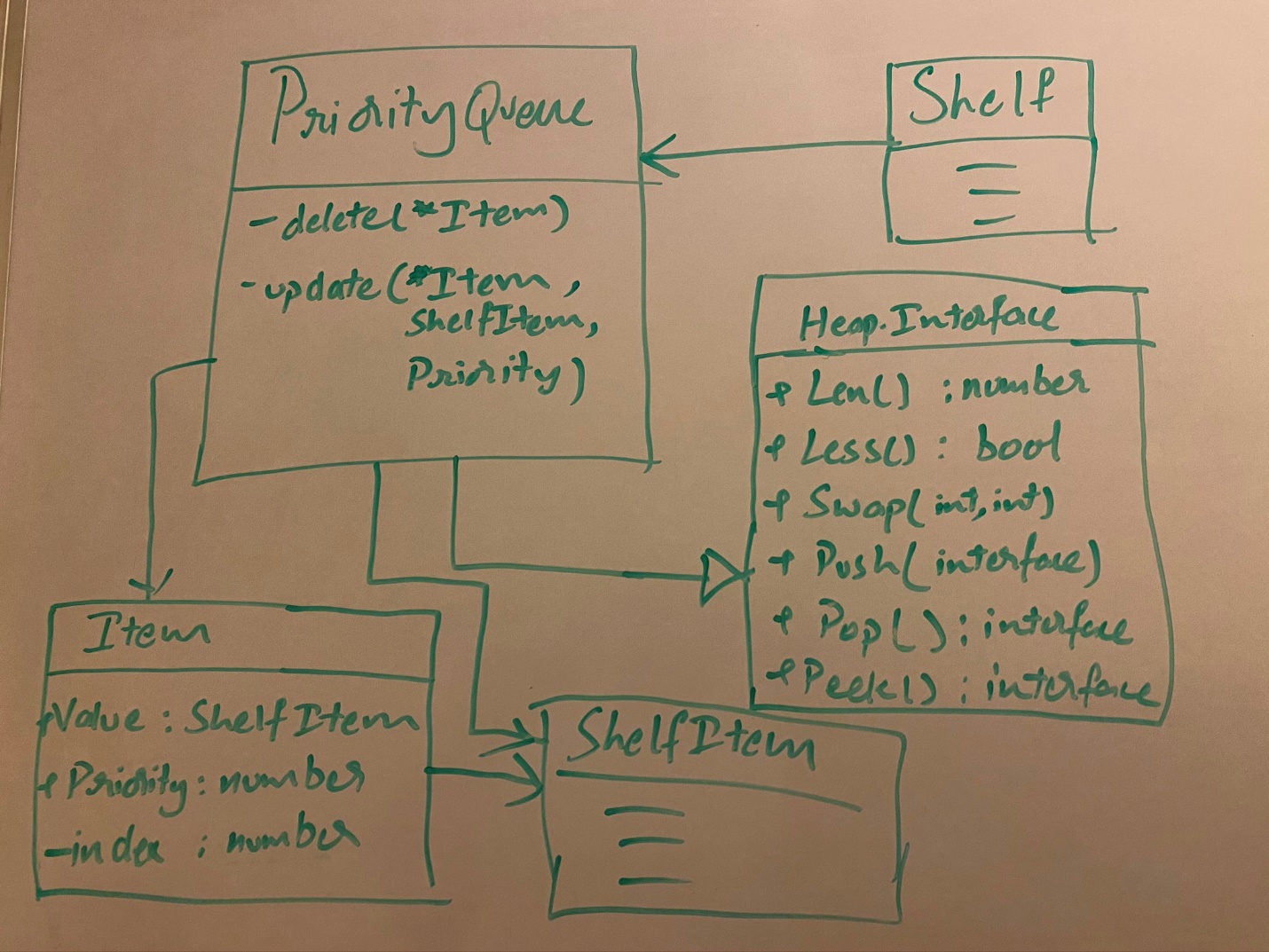
All the above components send/process interact through messages from/to buffered channels (queues) to achieve isolation between those services so they are not coupled. Differed the idea of centralized event aggregator (pub/sub service) due to time limitations. But, that’s left as a future scope of the solution

# Design



Rest of the models continued below:

### Models used by the Storage Service



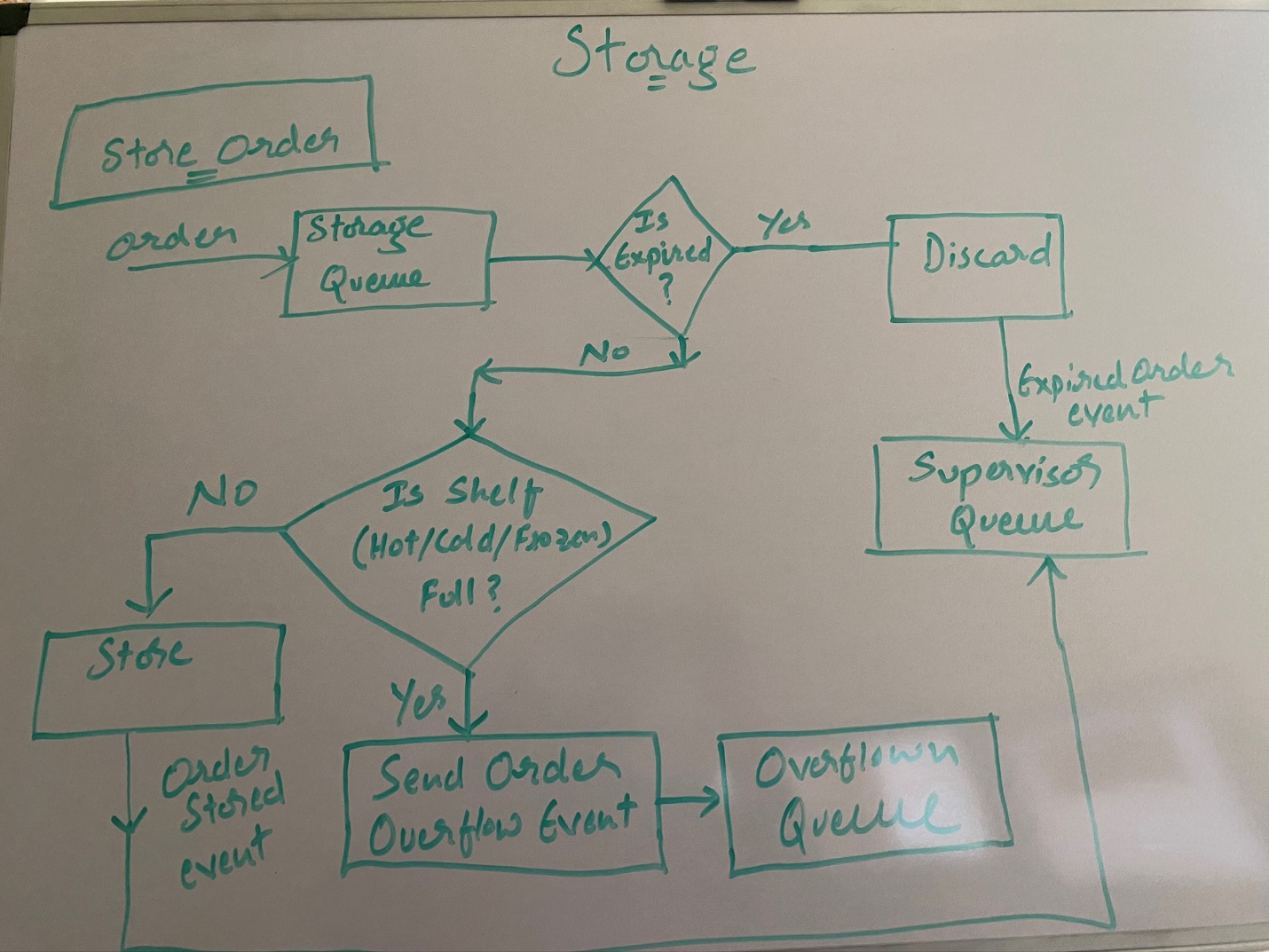
## Order Life Cycle

### Kitchen

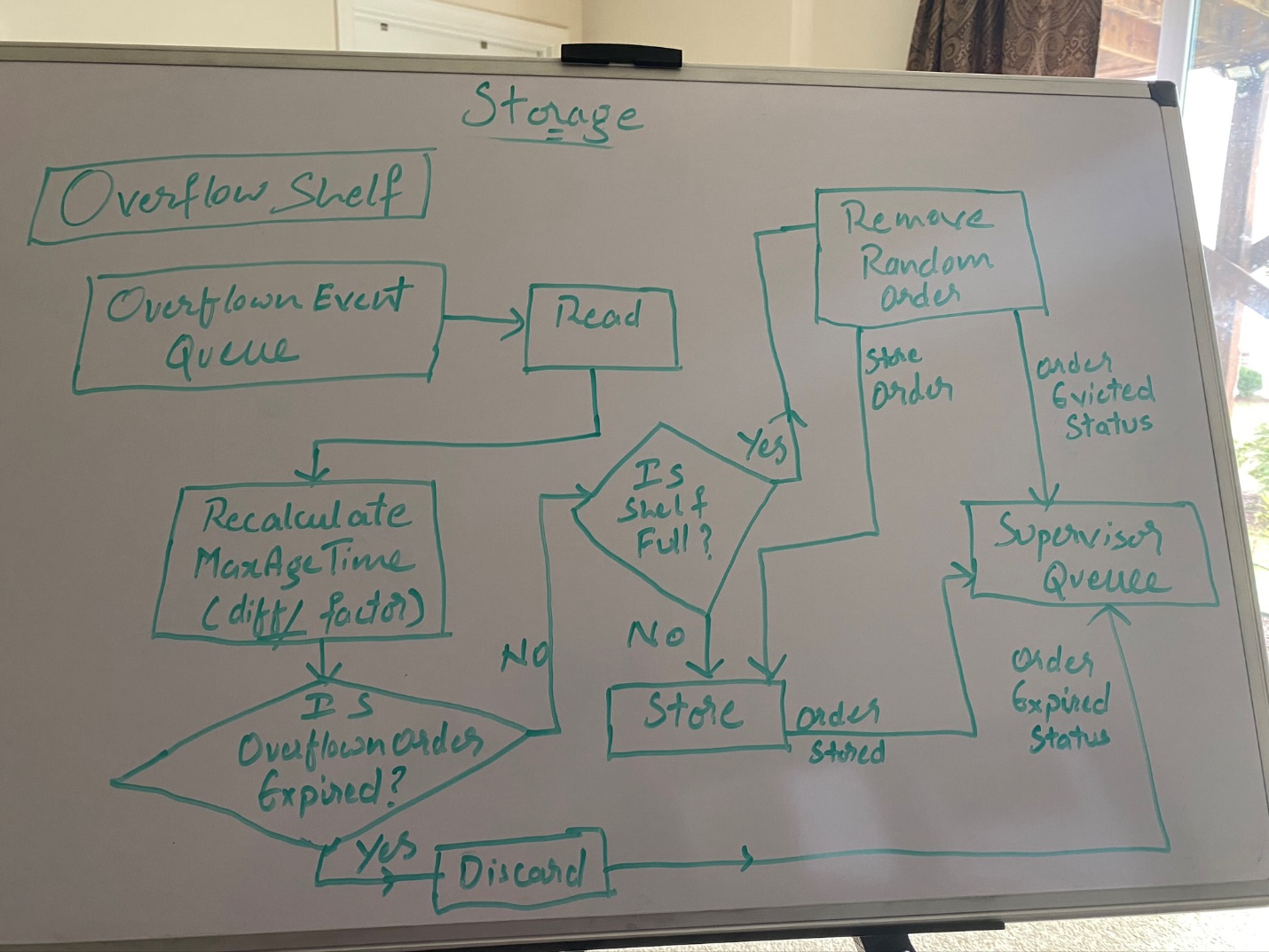
The Kitchen is straightforward. It receives order from Admin, processes it and Sends “Order Processed” event and “OrderReady” events to Storage and Dispatch Channels respectively

### Storage

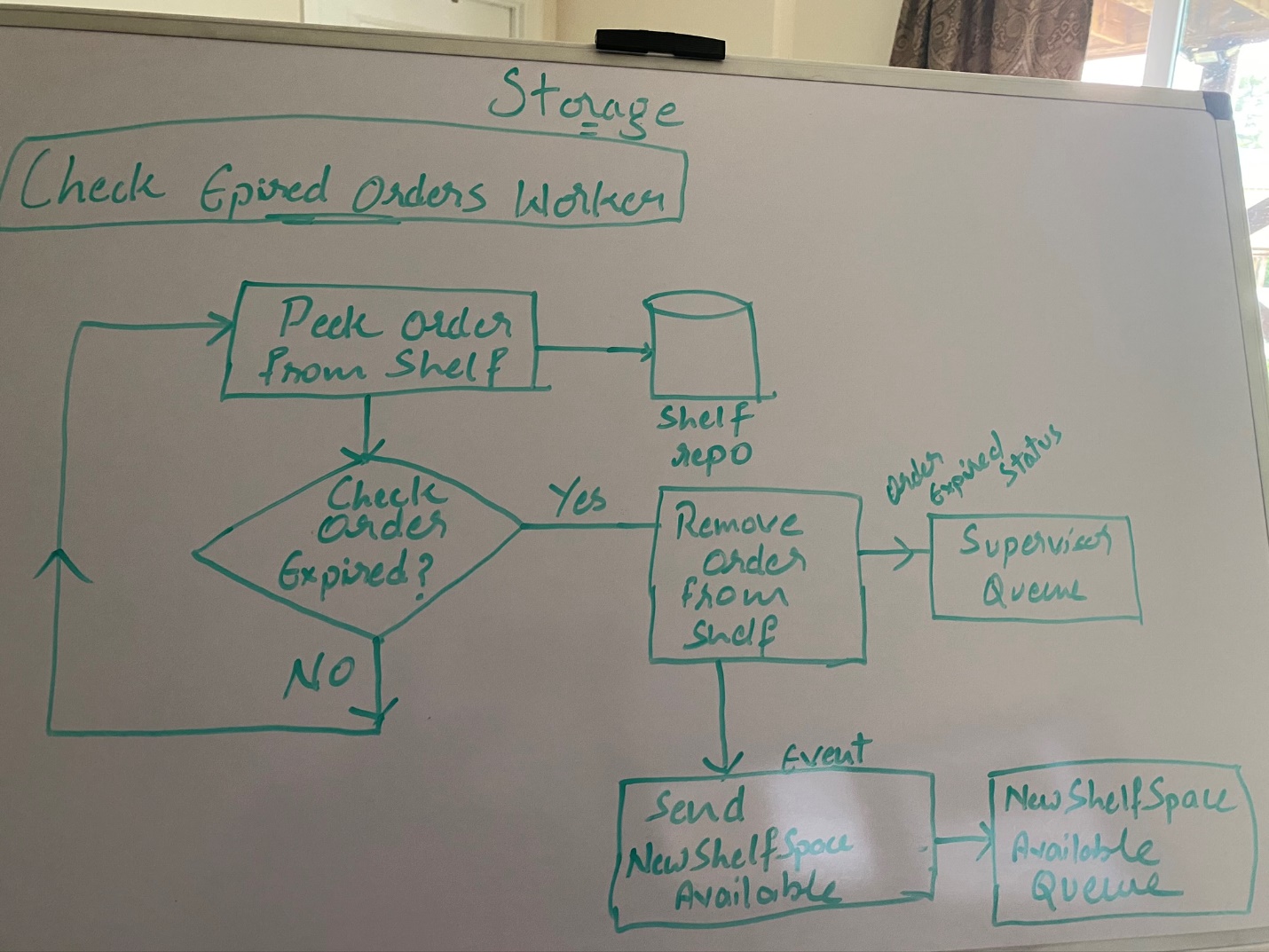
#### Normal Shelves (Hot/Cold/Frozen)



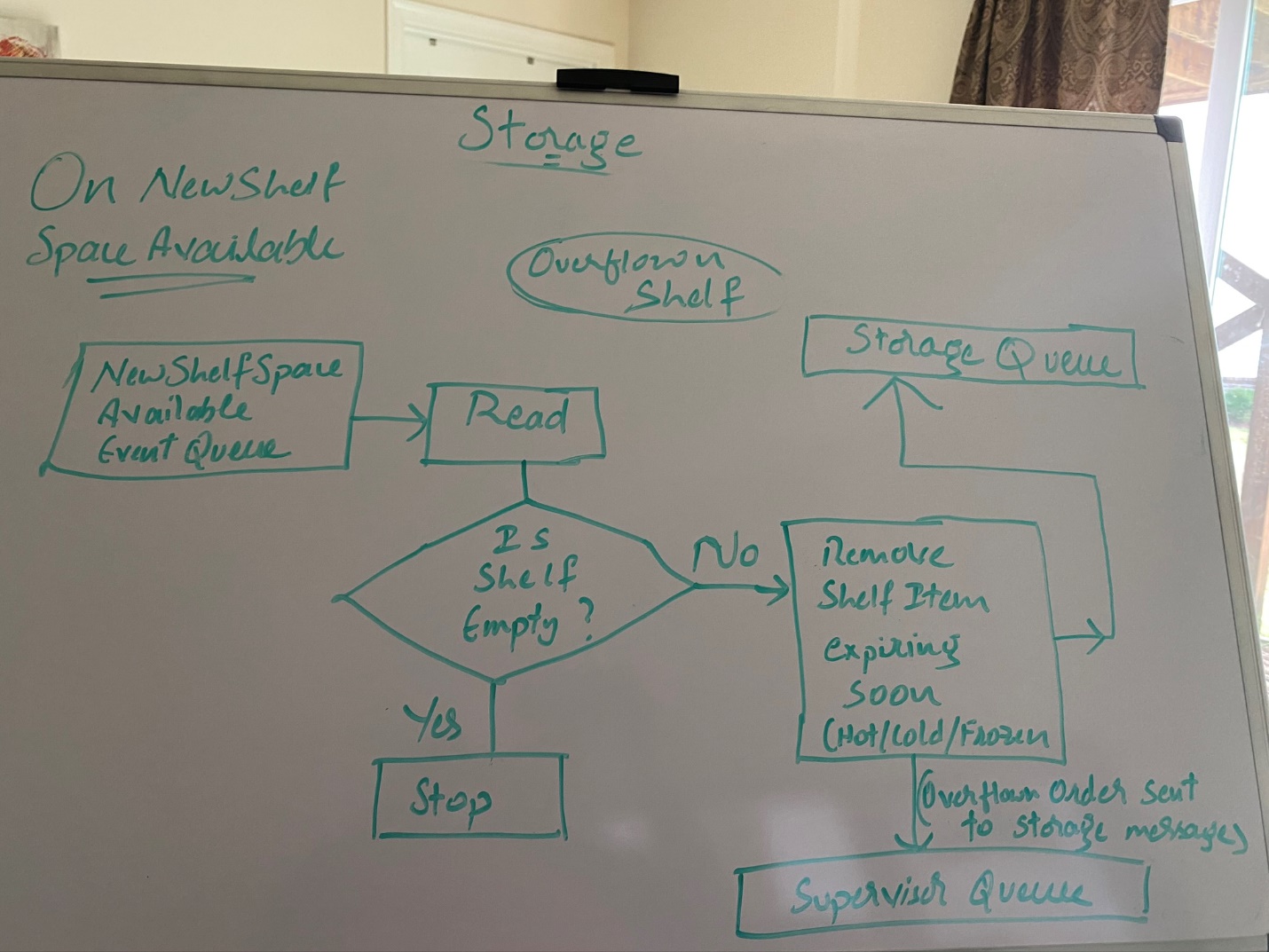
#### Overflow Shelf:



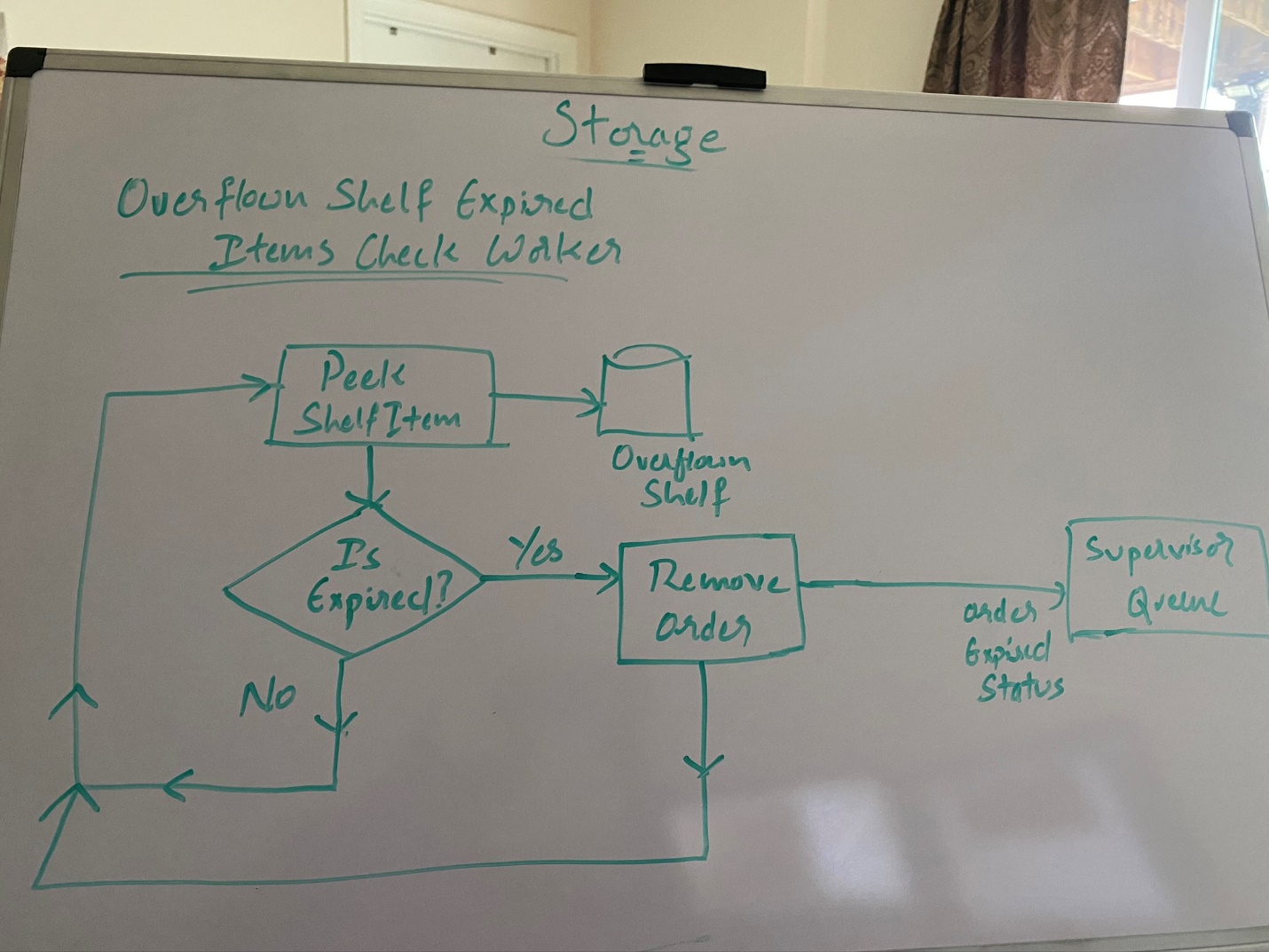
#### Check Expired Orders Worker – Normal Shelves (Hot/Cold/Frozen):



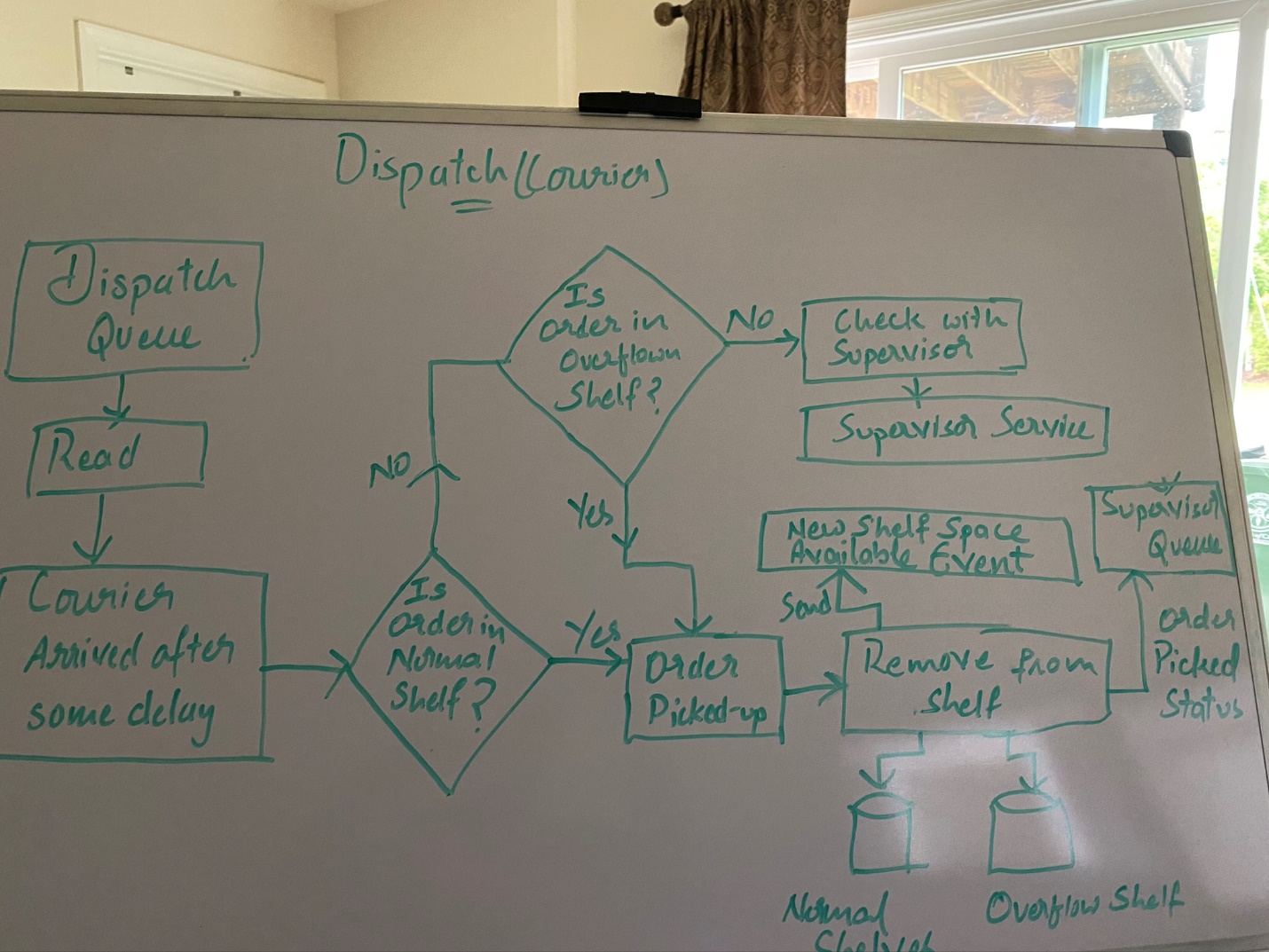
#### On New shelf space available event received – Overflown Shelf:



#### Overflown Shelf Expired Orders Checking Worker – Overflown Shelf:



### Dispatch (Courier)



If an order was not available while a courier picking up, the status is checked with Supervisor service. Supervisor service informs if that order is either evicted or expired or never even present (because it does book keeping of processed records).

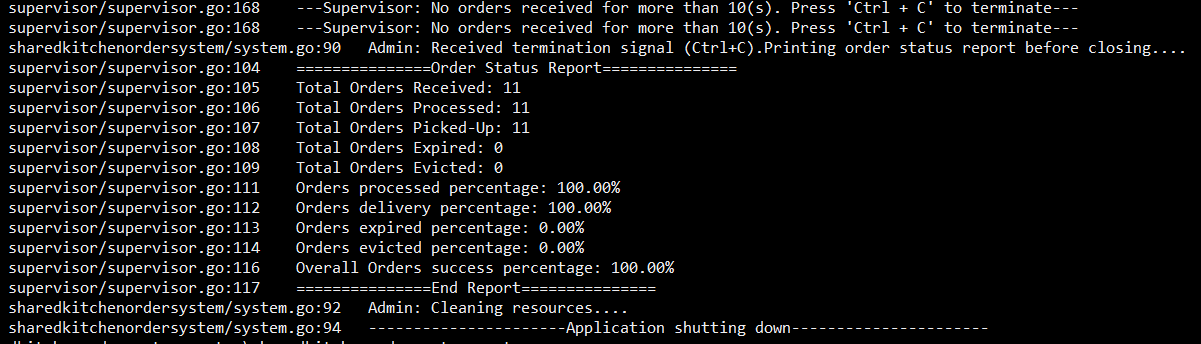
The supervisor finally would be able to print the overall report of all orders processed with how many of them were expired, evicted, processed and picked-up

### Supervisor

Supervisor continuously monitors the whole processes in different components. All the components are responsible to send events to supervisor. The pickup courier if could not find an order in the shelf to pick could consult the supervisor to seek the status.

A report is generated regarding the order status like percentage of orders processed, evicted, expired, picked-up (Press `Ctrl + C` at any point of time during the run). I just ran few sample sizes of orders and a report from the supervisor looks like this below:

#### Report



I ran several sample sizes of orders and generated reports over each process cross check the consistency of algorithms. This supervisor report helps to get feedback of the whole application and enables the engineer to improve the accuracy of the overall system and helps deep dive which algorithm and component needs to be fine tuned. Not only for accuracy but also helps to instrument and generate metrics so we know how the application could be scaled and tuned for performance

## Final Note

There are some short cuts employed only to expedite the submission due to some time constraints. Some limitations are:

* The overflow shelf could have had been extended IShelf interface and override few methods since its jobs and responsibilities are almost the same as other normal shelves (Hot/Cold/Frozen)
* There could more unit tests written to validate the whole system. But only the core components like Storage, Shelf Repo, Decay Calculator were tested thoroughly since these are the key ones that could change the accuracy of the application
* In production environment, the channels could be actually distributed AMQP services. But still internal queues/channels are necessary to help throttle the application and build resilience retrying failed requests due to some application/network exceptions. The supervisor component is used to do some analytics on the data it keeps. This could be any key-value store NoSql solution where some analytics could be run broke down by components, operations, order status or what not. Here I assumed only a simple scheme that is enough to validate the current design decisions made to work on this application.