DinDinn | Mini Assignment

Problem Description

Suppose we have 2 kinds of apps, merchant app and customer app. When the customer makes an order, the order will be received on the merchant app as a **new order**.
The api response is like this (**just for example, you can modify the API if you feel you have to. The most required fields are created_at, alerted_at, expired_time, and id**):

```
{
        "status": {
                "success": true,
                "statusCode": 200,
                "message": "success"
        },
        "data": [{
                "id": 10,
                "title": "Special extra large fried rice",
                "addon": [{
                        "id": 21,
                        "title": "Fried Egg",
                        "quantity": 3
                }],
                "quantity": 1,
                "created_at": "2021-06-10T15:00+00Z",
                "alerted_at": "2021-06-10T15:03+00Z",
                "expired_at": "2021-06-10T15:05+00Z"
        }, {
                "id": 11,
                "title": "Chicken Noodle",
                "addon": [{
                        "id": 26,
                        "title": "Extra chicken",
                        "quantity": 2
                }, {
                        "id": 27,
                        "title": "Sambal",
                        "quantity": 1
                }],
                "quantity": 1,
                "created_at": "2021-06-10T15:10+00Z",
                "alerted_at": "2021-06-10T15:13+00Z",
                "expired_at": "2021-06-10T15:15+00Z"
        }]
}
```

The data will contain an array of order objects, and every order has a list of addons. When we receive that response, we need to start the timer, the progress bar animation will be triggered for the given time for every single order item.

There are 3 major scenarios involved here which are: “**Loaded for the first time**” and “**accept the order**”.

First Scenario: Load Data

Let's say we received new orders from our api, we got two orders here. The first order expiration time is at “**2021-06-10T15:05+00Z**” (universal time), and we need to create a progress bar that will be reduced every minute (countdown progress bar). When the time has reached the given specific **alerted_at**, then we need to play a default system ringtone (you can use an external sound file also).

So if the response contains 2 order items, that means we are going to have 2 counter threads (to animate each item asynchronously).

You are asked to design the asynchronous rendering for every single item in our list.

Second Scenario : Accept the Order

We have received 2 orders, and we have them rendered on our screen. The timers are on now. When the timer is passing the expired time, then the view will be changed, and there is no “**accept**” button anymore (change the order item view to just show the “Expired” button).
When the expired button is pressed, the item will be removed from the list. But when the user presses the “**accept**” button before passing its expiration time, then the item will be removed from the list. The sound and the timer which related to this item has to be stopped.

You need to implement that feature with the most efficient thread management strategy.

Third Scenario : Go to the ingredient screen, and back to the order screen

We need to create the ingredient screen, which displays a list of ingredients. The data will be loaded from the API (**you can create your own JSON payload**). There is a search bar on the top of the ingredient screen. It takes text input and will call a search API. The response will be rendered on the ingredient list as the search result.

As you can see from the design, the order screen could have many tabs, based on how many menu categories we have. Every tab will call the same API (for example /ingredient_by_category?category_id=5)

When the user is back from ingredient to order screen, make sure the timer thread will keep running as long as the current time does not exceed the expiration time.

<u>The problems to be solved here are:</u>

- How you design the logic so that we could have multiple tabs based on the api response
- How you update the screen after searching for the ingredient.
- How you maintain the thread state when you back to the order screen

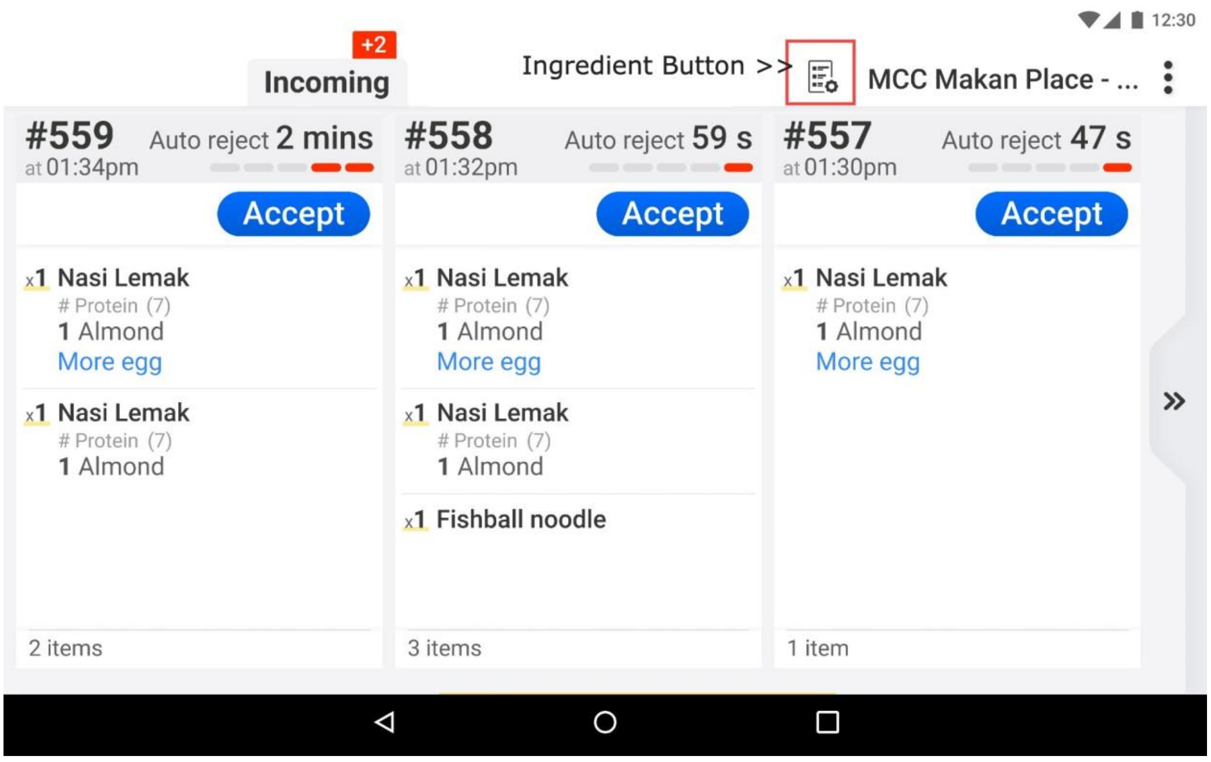
NOTE : Create unit test for all cases

Caveat

- The scenario is not limited just to what is written in this document. You can explore and apply some handlers for specific scenarios if you think that’s necessary.
- You can use any design pattern you think suitable for this problem (MVP, MVVM, MVI, etc).
- Pay attention to well written code structure and readability.
- Apply the concept of modularity.
- You can create mock api by yourself
- Stack : Kotlin, RxJava, Retrofit

The User Interface Design

Order Screen Design

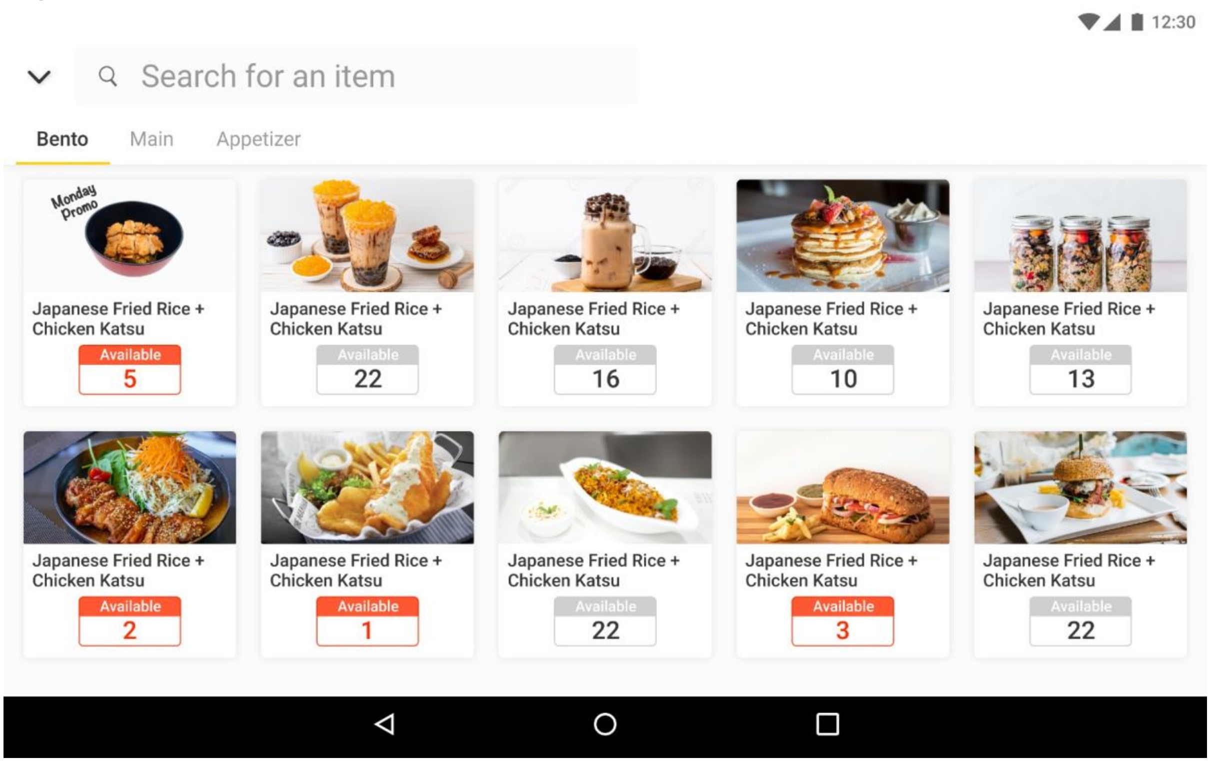


#559 Auto reject 2 mins
at 01:34pm
Accept
x1 Nasi Lemak
Protein (7)
1 Almond
More egg
x1 Nasi Lemak
Protein (7)
1 Almond
2 items

#558 Auto reject 59 s
at 01:32pm
Accept
x1 Nasi Lemak
Protein (7)
1 Almond
More egg
x1 Nasi Lemak
Protein (7)
1 Almond
x1 Fishball noodle
3 items

#557 Auto reject 47 s
at 01:30pm
Accept
x1 Nasi Lemak
Protein (7)
1 Almond
More egg
1 item

Ingredient Screen



Japanese Fried Rice + Chicken Katsu Available 5	Japanese Fried Rice + Chicken Katsu Available 22	Japanese Fried Rice + Chicken Katsu Available 16	Japanese Fried Rice + Chicken Katsu Available 10	Japanese Fried Rice + Chicken Katsu Available 13
Japanese Fried Rice + Chicken Katsu Available 2	Japanese Fried Rice + Chicken Katsu Available 1	Japanese Fried Rice + Chicken Katsu Available 22	Japanese Fried Rice + Chicken Katsu Available 3	Japanese Fried Rice + Chicken Katsu Available 22