

Design Document

Team:

Ramineni Amith Varma - 190050099

Sunkari Pranay Varma - 190050120

Vishnu Vardhan A - 190050130

Yallanki Prathyusha - 190050132

Logical Schema

1. Item_id -> item_name, item_type, price, availability
2. Ing_id -> ing_name, availability, price
3. person_id -> person_name, person_type, type_from, type_to, address, phone_no, salary, email, password
4. purchase_id -> purchase_name, purchase_date, purchase_time, quantity
5. table_id -> table_type, capacity, price
6. dp_id -> dp_name, rating, primary_no, secondary_no, phone_no, salary
7. item_f_id -> item_id, person_id, feedback_txt, suggestions, rating
FOREIGN KEY(item_id) references items
FOREIGN KEY(person_id) references persons
8. dp_f_id -> dp_id, person_id, feedback_txt, suggestions, rating
FOREIGN KEY(person_id) references persons
FOREIGN KEY(dp_id) references delivery_persons
9. off_order_id -> quantity, order_price, order_date, order_time
10. on_order_id -> quantity, person_id, order_price, order_date, order_time, delivery_address, is_delivered, is_cancelled, estimated_time, dp_id, delivery_date, delivery_time
FOREIGN KEY(person_id) references persons
FOREIGN KEY(dp_id) references delivery persons
11. coupon_id -> coupon_txt, coupon_type, availability, start_date, end_date
12. c_id -> on_order_id, c_reason, date, time
FOREIGN KEY(on_order_id) references online_orders
13. item_id , ing_id -> quantity
FOREIGN KEY(item_id) references items
FOREIGN KEY(ing_id) references ingredients
14. purchase_id, ing_id -> quantity
FOREIGN KEY(purchase_id) references purchases
FOREIGN KEY(ing_id) references ingredients

15. on_order_id, item_id -> quantity
 FOREIGN KEY(on_order_id) references online_orders
 FOREIGN KEY(item_id) references items
16. off_order_id, item_id -> quantity
 FOREIGN KEY(item_id) references items
 FOREIGN KEY(off_order_id) references offline_orders
17. coupon_id, person_id -> use_date, use_time, is_used
 FOREIGN KEY(coupon_id) references coupons
 FOREIGN KEY(person_id) references persons
18. booking_id -> table_id, person_id, booking_date, booking_from, booking_to
 FOREIGN KEY(table_id) references tables
 FOREIGN KEY(person_id) references persons

Integrity Constraints

Items

1. item_id, int - Primary Key
2. Item_name NOT NULL
3. availability >=0
4. price in [1, 500]
5. item_type = { food , beverage }

Ingredients

1. ing_id, int - Primary Key
2. ing_name NOT NULL
3. availability >= 0
4. price in [1,100]

Persons

1. person_id, int - Primary Key
2. Person_name NOT NULL
3. person_type = { SuperUser, Base Customer, Premium Customer, General Manager, Kitchen Manager , Billing Manager , Delivery Manager , Food Server , Chef, Delivery Person }
4. type_from <= type_to where type_from, type_to = subscription from and to dates
5. phone_no should contains 10 digits
6. Email should be of email format
7. Password is stored in hash format
8. Salary in [10000, 50000] and NULL for base_customer and premium_customer

Purchases

1. purchase_id, int - Primary key
2. Purchase_name NOT NULL
3. quantity >= 1

Tables

1. table_id, int - Primary Key
2. table_type = { normal , family , booth , outdoor }
3. capacity lies in [1,10]
4. price lies in range [1 , 500]

Item_Feedback

1. item_f_id, int - Primary Key
2. Item_id, person_id NOT NULL
3. feedback_txt = {bad , average , good , worthy }
4. rating lies in range [0 , 5]

Dp_Feedback

1. dp_f_id, int - Primary Key
2. Dp_id, person_id NOT NULL
3. feedback_txt = { fast_delivery , slow_delivery , neutral }
4. rating lies in range [1 , 5]

Offline Orders

1. Primary Key - {off_order_id, item_id}
2. quantity >= 1
3. Order_date, order_time, order_price NOT NULL

Online Orders

1. Primary Key - {on_order_id, item_id}
2. quantity >= 1
3. { (order_date < delivery_date)
OR
((order_date == delivery_date) AND (order_time <= delivery_time)) }
4. Order_price, order_date, order_time NOT NULL

Coupons

1. coupon_id, int - Primary Key
2. start_date <= end_date
3. availability >= 0
4. Start_date, end_date, coupon_type NOT NULL

Delivery Persons

1. dp_id , int - Primary Key
2. dp_rating lies in [0,5]
3. phone_no contains 10 digits
4. Phone_no, salary, primary_no dp_name NOT NULL

Cancellations

1. c_id , int - Primary Key
2. On_order_id, c_reason, date, time NOT NULL

Item_ing

1. {item_id, ing_id} - Primary Key
2. Quantity NOT NULL

pur_ing

1. {purchase_id, ing_id} - Primary Key
2. Quantity NOT NULL

Online_items

1. {on_order_id, item_id} - Primary Key
2. Quantity NOT NULL

Offline_items

1. {off_order_id, item_id} - Primary Key
2. Quantity NOT NULL

coupons_users

1. {coupon_id, person_id} - Primary Key
2. is_used NOT NULL

book_tables

1. booking_id, int - Primary Key
2. Table_id, person_id, booking_date, booking_from, booking_to NOT NULL

Views

- **out_of_stock_ing**

Get all ingredients which are currently out of stock , we need to keep track of Ingredients. Based on these results , managers will purchase the required ingredients
It reduces manager effort to search the ingredients which are out of stock , also reduces while using USE-CASES.

QUERY :

Create view out_of_stock_ing as (select ing_id , price from ingredients where availability <=0)

- **Free_dps**

Get all delivery persons who are free

While assigning delivery_persons to online_orders we use delivery persons which are free.

It reduces the work load of managers

This view will be used for further statistical analysis like who are most free_delivery guys during that weekend / month.

QUERY :

Create view free_dps as (Select dp_id as ready_person_id from delivery_persons where dp_id NOT IN (select dp_id from online_orders where is_delivered = 'False'))

- **free_tables**

Get all tables which are free

While booking a table we need to check everytime which tables are free by Managers

We will be using it in further analysis.

Note: We have not used these views currently but we MAY use it if needed at the time of writing code. These are just attached as extra material

Transactions

Description and Queries

1. -----LOGIN-----

DESCRIPTION:

Get the email and password from the inputs
Check email and passwords in persons table
On successful login display dashboard
For invalid login be on login page

QUERY :

Select count(*) from persons where email = input_email and password = input_password

2. -----ADDING INGREDIENTS-----

DESCRIPTION:

Get the ing_name ,price, availability from the inputs
We directly add into ingredients with these fields and constraints on the fields will be checked using integrity constraints.
Check the type of person adding the ingredients.
Ing_id will be auto -incremented.

QUERY :

(Select role_type from persons where person_email = logged_user_email and
person_password = logged_user_password);
If (role type = Kitchen Manager or General Manager or SuperUser) {
Insert into ingredients values (ing_name,availability,price);
}

3. -----ADDING ITEMS-----

DESCRIPTION:

Get the item_name , item_type ,price, availability from the inputs
We directly add into items with these fields and constraints on the fields will be checked by using integrity constraints.
Check the type of person adding the items.
Item_id will be auto-incremented.

QUERY :

(Select role_type from persons where person_email = logged_user_email and
person_password = logged_user_password);
If (role type = Kitchen Manager or General Manager or SuperUser) {
Insert into items values (item_name,availability,price);
}

4. -----ADDING ADMINS(staff)-----

DESCRIPTION:

Get the details like person_name , person_email , person_phone_no,etc., from the Inputs.

Check the type of person adding the staff only General Manager, SuperUser can add the staff into the database.

Person_id will be auto-incremented.

QUERY :

```
(Select role_type from persons where person_email = logged_user_email and
    person_password = logged_user_password );
If (role type = General Manager OR SuperUser) {
    Insert into persons values
(person_name,...,person_type,...,person_email,...person_password,...);
}
```

5. -----ONLINE ORDERS-----

DESCRIPTION:

Get item_id_value's , required_quantity's , delivery_address from the inputs.

Get logged_user_id from persons table.

Check the role of person it should be base_customer or premium_customer.

Check all item_id's availability wrt to the customer required quantity.

Check the delivery_person who is free to deliver.

On_order_id is auto-incremented

QUERY :

```
(Select role_type,logged_user_id from persons where person_email =
logged_user_email and person_password = logged_user_password );
If (role type = Base Customer OR Premium Customer) {
    (Select quantity as stock_left from items where item_id = item_id_value)
    Check whether (required_quantity <= stock_left)
    On success, repeat the same procedure for the next item with required_quantity.
    (Select dp_id as ready_person_id from delivery_persons where dp_id NOT IN
        (select dp_id from online_orders where is_delivered = 'False') limit 1;)
    Finally, if there are at least one_delivery_person free then{
    Insert into online_order_items(item_id , required_quantity );
    Insert into online_orders
values(item_id_val1,quantity1,logged_user_id,...,ready_person_id,...);
    }
}
```

6. -----ADDING COUPONS-----

DESCRIPTION:

Get coupon_txt, coupon_type, availability, start_date, end_date from the inputs
We directly add into coupons with these fields and constraints on the fields will be checked using integrity constraints.

Check the type of person adding the coupons.

coupon_id will be auto -incremented.

QUERY :

```
(Select role_type from persons where person_email = logged_user_email and
      person_password = logged_user_password );
If (role type = General Manager OR Billing Manager) {
    Insert into coupons values (coupon_txt,coupon_type,...,start_date,end_date);
}
```

7. -----ADDING DELIVERY PERSONS-----

DESCRIPTION:

Get dp_name, phone_no, primary, secondary,..so on... from the inputs
We directly add into delivery_persons with these fields and constraints on the fields will be checked using integrity constraints.

Check the type of person adding the delivery_persons.

dp_id will be auto -incremented.

QUERY :

```
(Select role_type from persons where person_email = logged_user_email and
      person_password = logged_user_password );
If (role type = General Manager OR Billing Manager) {
    Insert into delivery_persons values (dp_name,...,primary, secondary, phone_no);
}
```

8. -----VIEW STAFF-----

DESCRIPTION:

Check the role of logged user , only managers can view staff

QUERY :

```
(Select role_type from persons where person_email = logged_user_email and
      person_password = logged_user_password );
If (role type = General Manager OR SuperUser) {
    (Select * from persons where person_type NOT IN (SuperUser ,
      Base Customer , Premium Customer ); )
}
```

9. -----VIEW AVAILABLE ITEMS-----

DESCRIPTION:

Check the role of logged user , only customers can view available items

QUERY :

```
(Select role_type from persons where person_email = logged_user_email and
      person_password = logged_user_password );
```



```

    If (role type = Base customer OR Premium Customer) {
        ( Select * from items where availability > 0 ; )
    }

```

10. -----DELETING COUPONS AFTER EXPIRY-----

DESCRIPTION:

Get the current_date from input/frontend
 Checking the validity of coupon using end_date attribute from coupons
 Deleting the expired coupons by General Manager or SuperUser

QUERY :

```

(Select role_type from persons where person_email = logged_user_email and
    person_password = logged_user_password );

```

```

If (role type = General Manager OR SuperUser){
    DELETE FROM coupons where end_date > current_date;
}

```

11. -----UPDATING FREE DELIVERY PERSONS AFTER DELIVERY-----

DESCRIPTION:

Get the online_delivered_id from input/frontend.
 Checking the status of delivery_person using is_delivered variable from
 Online_orders
 Check the role of person who is updating the database

QUERY :

```

(Select role_type from persons where person_email = logged_user_email and
    person_password = logged_user_password );
If (role type = General Manager OR SuperUser){
    UPDATE online_order SET is_delivered = 'True' WHERE on_order_id =
        online_delivered_id ;
}

```

12. -----VIEW USER PENDING ONLINE ORDERS-----

DESCRIPTION:

Check the role of the person , only customers can view this.
 Check the variable is_delivered in online_orders for pending online orders

QUERY :

```

(Select role_type from persons where person_email = logged_user_email and
    person_password = logged_user_password );
If (role type = Base customer OR Premium Customer ) {
    Select * from online_orders where is_delivered = 'False'
}

```

13. -----VIEW HISTORY OF ONLINE ORDERS-----

DESCRIPTION:

Get the person_id from persons table using email and password
Check the role of person - only Base Customer or Premium Customer can view the history of their orders

QUERY :

p_id = Select person_id from persons where
(person_email = logged_user_email and person_password = logged_user_password
and ((person_type = Base Customer) or (person_type = Premium Customer))) limit 1;

Select * from online_orders where person_id =p_id;

14. -----VIEW PROFILE-----

DESCRIPTION:

Complete profile can be viewed in thai use -case

QUERY :

(Select * from persons where person_email = logged_user_email and
person_password = logged_user_password);

15. -----UPDATE PROFILE-----

DESCRIPTION:

Get the mail,password from frontend
Check the role of person - Every user can update the profile except delivery person
Person can update the mail, password, name, address, phone number

QUERY

p_id = Select person_id from persons where
(person_email = logged_user_email and person_password = logged_user_password
and person_type != delivery person);

UPDATE persons SET email=updated_mail, password=updated_password,
person_name = updated_name, address=updated_address,phone_no=updated_phone
where person_id=p_id; [person can update any of these columns]

16. -----SORT THE ITEMS BASED ON RATING , PRICE -----

DESCRIPTION:

Sorting items for user based on price or rating and order food accordingly.
Check the role of logged user , only customers can view the sorted list for ordering
items based on rating , prices.

QUERY :

```

        (Select role_type from persons where person_email = logged_user_email and
          person_password = logged_user_password );
    If (role type = Base customer OR Premium Customer) {
        ( Select * from items where availability > 0 ORDER BY price desc);
        ( Select * from items where availability > 0 ORDER BY rating desc);
    }

```

17. -----SORT DELIVERY PERSONS BASED ON RATINGS-----

DESCRIPTION:

Checking the best delivery_persons based on ratings for managers and giving rewards.
Check the role of logged user , only managers , superUSER can view the sorted list of delivery_persons.

QUERY :

```

        (Select role_type from persons where person_email = logged_user_email and
          person_password = logged_user_password );
    If (role type = General Manager OR SuperUser)) {
        ( Select * from delivery_persons ORDER BY rating desc);
    }

```

18. -----UPDATE AVAILABILITY OF ITEMS -----

DESCRIPTION:

Get input_online_id from input/frontend on placing the order.
Update the items quantity after ordering.
Check the role of person , only managers , superUser can update the data.

QUERY:

```

        (Select role_type from persons where person_email = logged_user_email and
          person_password = logged_user_password );
    If (role type = General Manager OR SuperUser)) {
        With store(id,quantity) as (Select item_id,quantity from online_order_items where
          on_order_id =input_online_id)
        UPDATE items SET availability = availability - store.quantity from items,store
          WHERE items.item_id = store.id ;
    }

```

19. -----UPDATE AVAILABILITY OF INGREDIENTS-----

DESCRIPTION:

Get input_online_id from input/frontend on placing the order.
Update the ingredients quantity after ordering.
Check the role of person , only managers , superUser can update the data.

QUERY:

```

(Select role_type from persons where person_email = logged_user_email and
  person_password = logged_user_password );
If (role type = General Manager OR SuperUser)) {
  With store(id,quantity) as (Select item_id,quantity from online_order_items where
    on_order_id =input_online_id),
    sto(id,quan) as (select ing_id,quantity*store.quantity from item_ing where
      item_id=store.id)
  UPDATE ingredients SET availability = availability - sto.quan from ingredients ,sto
  WHERE  ingredients.ing_id = sto.id ;

}

```

20. -----UPDATE ON CANCELING ORDERS-----**DESCRIPTION:**

Get online_id from input/frontend on canceling the order.

Update the item quantity after canceling.

Check the role of the person , only managers , superUser can update the data.

c_id is auto incremented

QUERY:

```

(Select role_type from persons where person_email = logged_user_email and
  person_password = logged_user_password );
If (role type = General Manager OR SuperUser)) {
  With store(id,quantity) as (Select item_id,quantity from online_order_items where
    on_order_id =input_online_id),
    UPDATE items SET quantity = quantity + store.quantity from items,store WHERE
      items.item_id = store.id ;
  UPDATE online_orders SET is_delivered = 'True', is_cancelled = 'True' where
on_order_id = online_id;
  INSERT INTO cancellations VALUES (online_order_id, c_reason, c_date, c_time);
}

```

21. ----VIEW ALL ITEMS IN A PARTICULAR ONLINE ORDER FOR CUSTOMER-----**DESCRIPTION:**

Get online_id from input/frontend of order

Check the role of a person - only customer can see this

QUERY:

```

(Select role_type from persons where person_email = logged_user_email and
  person_password = logged_user_password );
If (role type = Base Customer OR Premium Customer)) {
  Select item_id from online_order_items where on_order_id = input_online_id;
}

```

22. -----OFFLINE BOOKING-----

DESCRIPTION:

Check the availability of tables.

Get the item_name's, req_quantity, current date & current_time from inputs

Offline customers can book the table for 2hrs.

Check the availability of items in the order.

Insert into book_table with person_id = 0

booking_date = current_date

Booking_from = current_time

Booking_to = current_time + 2hrs

booking_id is auto-incremented

off_order_id is auto-incremented

QUERY :

```
free_table_id = (select (select table_id as a from tables) - (select table_id as b from
book_table where (booking_date=order_date and ((order_time > booking_from) or (order_time +
2hrs < booking_to)) or booking_date != order_date) limit 1;)
```

```
if (free_table_id > 0) {
```

```
  (Select quantity as stock_left from items where item_id = item_id_value)
```

```
  Check whether (required_quantity <= stock_left)
```

```
  On success, repeat the same procedure for the next item with required_quantity.
```

```
  Insert into offline_orders values (req_quantity , ..., current_date ,current_time);
```

```
  Insert into book_table values(free_table_id , 0 ,
booking_date,booking_from,booking_to );
}
```

23. -----POSTING ITEM FEEDBACK-----

DESCRIPTION:

Check whether the person is signed up

```
if(success){
```

Check the role of a person - only Base Customer or Premium Customer can give feedback

Get item_id for the particular item

Check whether person has purchased the item

```
  if(success){
```

```
    Insert rating and suggestions in item_feedback table
```

```
  }}
```

Item_f_id is auto incremented

QUERY:

```
role_type = Select role_type from persons where person_email = logged_user_email and
              person_password = logged_user_password;
If (role_type = Base Customer or Premium Customer) {
    item_id = select item_id from items where item_name=input_item_name
    p_id     = select person_id from persons where person_email =logged_user_email
              and person_password = logged_user_password;
    online_o_id = select on_order_id from online_orders where person_id = p_id;
    i_id       = select item_id from online_order_items where on_order_id = online_o_id;
    if(item_id == i_id){
        INSERT INTO item_feedback
            VALUES ( item_id, person_id, feedback_txt, suggestions, rating);
    }
}
```

24. -----TABLE RESERVATION-----

DESCRIPTION:

check whether the person is signed up

if(success){

check the role of person - only Base Customer or Premium Customer can book

check if tables are available

Insert entries into book_table

}

QUERY:

```
role_type = Select role_type from persons where person_email = logged_user_email and
              person_password = logged_user_password;
If (role_type = Base Customer or Premium Customer) {
    table_id = select (select table_id as a from tables) - (select table_id as b from
                                                            book_table) limit 1;
    if(table_id > 0){
        INSERT INTO book_table VALUES
            (booking_id,table_id,person_id,booking_date,booking_from,booking_to);
    }
}
```

25. -----UPDATING TABLES BOOKED THROUGH OFFLINE CUSTOMERS-----

DESCRIPTION:

Check the tables which are booked for offline customers using person_id = 0.

Delete the entry

Check the role of person deleting the entry , it should be General managers
and SuperUser.

QUERY:

```
If (role type = General Manager OR SuperUser) {  
  
    DELETE from book_table where table_id IN (Select table_id from  
        book_table where person_id = 0);  
  
}
```

DDL and SQL Files

DDL.sql and InsertData.sql files can be found in this [link](#)

Indexes

Most of our relations rely on id as primary key and queries almost use id's as predicates. So, additional indices are not needed other than id

Forms and Technologies**Technologies**

Frontend - AngularJS

Backend - NodeJS

Database - PostgreSQL

Note:

Forms and Business Logic Controller are attached in subsequent pages

1) sign up for the website

Input

person name	<input type="text"/>	*
email	<input type="text"/>	*
Password	<input type="text"/>	*
confirm password	<input type="text"/>	*
person type	<input type="text" value="select type"/>	*
type from	<input type="text"/>	*
type to	<input type="text"/>	
Address	<input type="text"/>	*
phone no	<input type="text"/>	*
salary	<input type="text"/>	*

output:-

sign in page

2) Log in page

Input:

Email	<input type="text"/>	*
Password	<input type="text"/>	*
<input type="button" value="Login"/>		
Forget password?		

Output:

Homepage

3) Adding Ingredients

Input:

Ing-name	<input type="text"/>	*
Price	<input type="text"/>	*
Availability	<input type="text"/>	*
<input type="button" value="Add Ingredient"/>		

output:

Adds ingredient to database,
redirects to same page

4) Adding Items

Input:-

item_name	<input type="text"/>	*
item_type	<input type="text" value="select type"/>	*
availability	<input type="text"/>	*
Price	<input type="text"/>	*
<input type="button" value="Add Item"/>		

Output:

Add item to database,
Redirects to same page

5) Adding Admins

Input:-

Person-name	<input type="text"/>	*
Person-type	<input type="text" value="select type"/>	*
type-from	<input type="text"/>	*
type-to	<input type="text"/>	*
address	<input type="text"/>	*
phone-no	<input type="text"/>	*
<input type="button" value="Add Admin"/>		

Output:

Add person to database,
redirects to same page

6) Online orders

Input:

item_name	<input type="text"/>	*
quantity	<input type="text"/>	*
delivery address	<input type="text"/>	*
<input type="button" value="place order"/>		

Output:

Order will be placed & delivery person
details will be displayed.

7) Adding Coupons

Input:

coupon.txt	<input type="text"/>	*
coupon type	<input type="text" value="select type"/>	*
availability	<input type="text"/>	*
start date	<input type="text"/>	*
end-date	<input type="text"/>	*
<input type="button" value="Add Coupon"/>		

Output:

coupon is added & a success message is displayed.

8) View staff

~~Input:~~

staff list					
<input type="text" value="Person-name"/>	<input type="text" value="Person.type"/>	<input type="text" value="type-from"/>	<input type="text" value="type-to"/>	<input type="text" value="address"/>	<input type="text" value="phone-no"/>
:	:	:	:	:	:
:	:	:	:	:	:

↑

output: Persons in staff are fetched and displayed.

9) Adding delivery persons

Input:

dp name	<input type="text"/>	*
phone no	<input type="text"/>	*
primary	<input type="text"/>	*
secondary	<input type="text"/>	*
dp rating	<input type="text" value="select"/>	*

Output:

Adds the delivery persons to the delivery person table, redirects to homepage.

10) view available items

Input: None

Items available			
item name	item type	availability	price
:	:	:	:

Output:

Items will be fetched and displayed

11.) View user pending online orders

Input: None

pending online orders list

order x	:	-----
order y	:	=====
!		

output:

Pending online orders of the user will be fetched and displayed.

12.) View history of online orders

Input: _____

History of online orders

on_order_id	quantity	delivery_date	delivery_time
:	:	:	:

Output: History of online orders of the user will be displayed.

13.) View profile

User profile

output:

Profile of the person will be displayed.

14.) Update profile

user profile

email

person name

phone_no

address

output:

user profile will get updated
& redirects to same page

15) Updating free delivery persons after delivery

online-delivered-id * -from front end

is-delivered *

output:

Free delivery persons list
will get updated.

16) Deleting Coupons after expiry.

current.date * -from front end

Coupon.end.date *

output:

Expired Coupons will
get deleted

17) Sort the items based on rating ; price

Items list after sorting based on
Price & rating,

Item-name	price	rating	availability
⋮	⋮	⋮	⋮

output:

rearrange the items based
on rating & price

18) view all items in a particular online order

Item in the online order.

⋮

output:

fetches & displays the all items
in the corresponding online
order

1) How to book an online order by user

CUSTOMER'S DASHBOARD

LOGIN

Email _____

Password _____

Submit

Successfull login

DASHBOARD

Profile

Place an order

History of orders

Pending orders

view profile

Online order

Online order page

Rating	Price
Item	Quantity
_____	_____
_____	_____
_____	_____

Submit

sort items based on price descending

Welcome! name

email

phone no

address

sort items based on rating descending

Items - sorted

Item 1 Rating 1

Item 2 Rating 2

Item 3 Rating 3

...

Items - sorted page

Item 1 price 1

Item 2 price 2

Item 3 price 3

...

Order - id

Viewing all items in that order - id

Item page

Item name

Item price

Availability

on successful submission

DO USE CASE

update availability of items

DO USE CASE

update availability of ingredients

DO USE CASE

update on canceling orders

view all pending orders

order - id 1

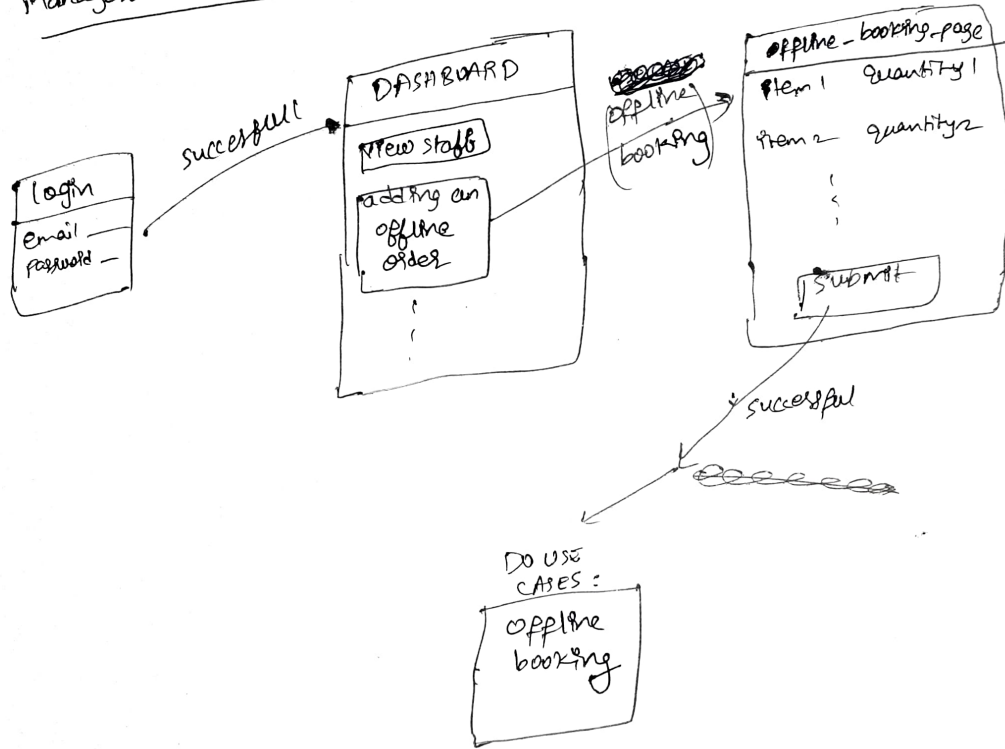
cancel ?

order - id 2

cancel ?

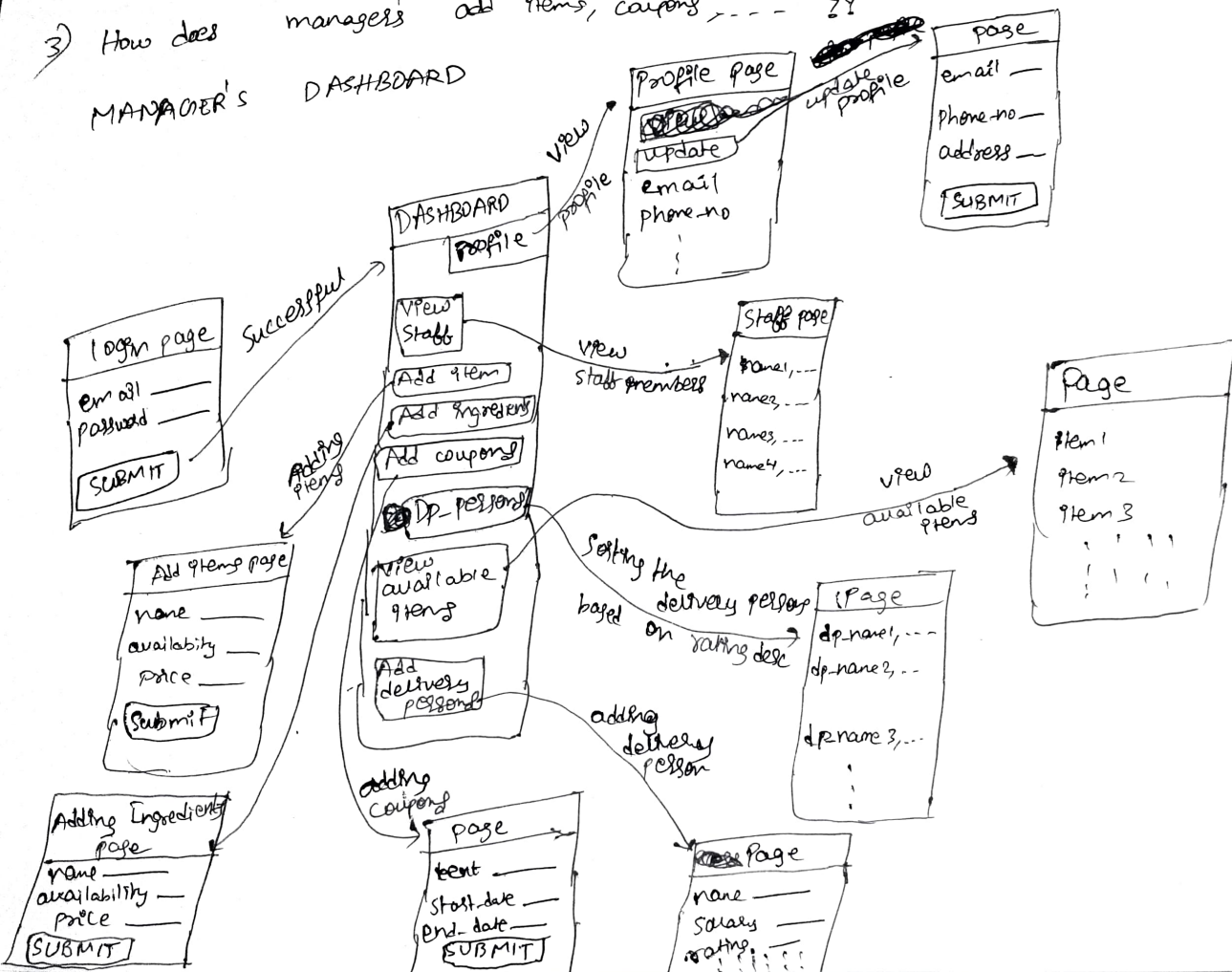
...

2) How we perform offline orders
Managers Dashboard (or) Super User Dashboard



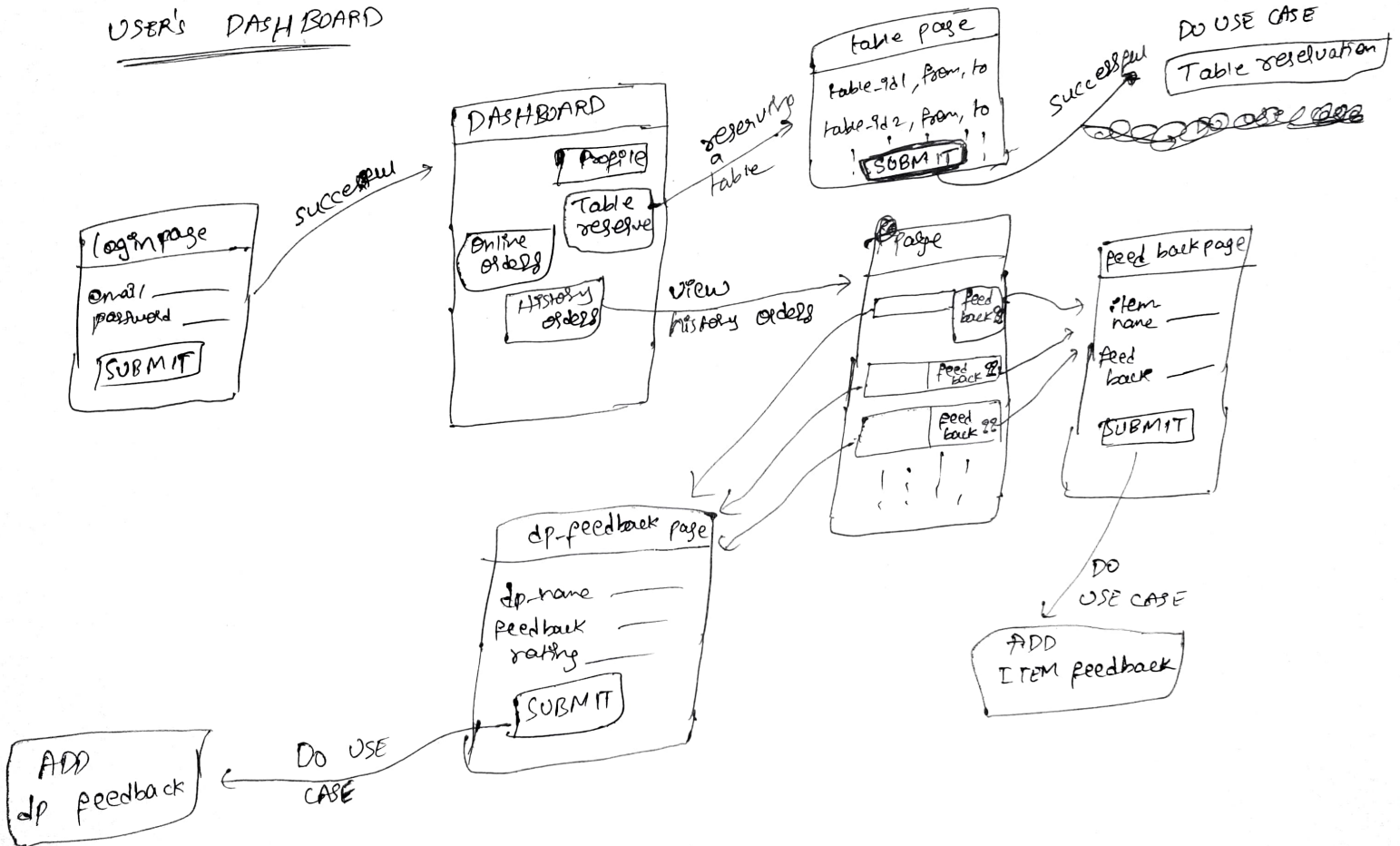
3) How does managers add items, coupons, --- ??

MANAGER'S DASHBOARD



4) Giving feedback by USERS to items / dp-persons

USER'S DASHBOARD



5) logging as Super User

SUPER USER DASHBOARD
It's above all

Login Page

email _____

password _____

SUBMIT

Successful
login

DASHBOARD

profile

add an item

remove empty coupon

updating tables booked through offline customers

!!!

DO USE
CASE

deleting coupon
after expiry

DO USE
CASE

updating tables
booked through
Offline customers

* Super User is ULTIMATE ONE, It has more powers than Manager's