CS387

RESTAURANT MANAGEMENT SYSTEM

180050024 - BHOGI SURAJ KRISHNA
180050043 - KANIKE PUNEETH SAI KUMAR
180050083 - RAMANATHI MANJUNATH
180050113 - TELU BARGAV SAI

REQUIREMENTS

We have 4 roles in total -- Customer, Staff, Delivery Person and Manager. Each user will have a username and password to login to their respective webpage. Manager has control over all the other users.

Customer Requirements :

1. Customers can sign up or login if they are an existing user.
2. Once logged in, users can view trending items at our restaurant on that day and our top sellers (Based on total no of orders) on the home page.
3. Can search their favourite dish by typing it out or can filter by clicking pre defined tags (like starters, desserts, pizza etc)
4. Registered users can add food items to cart and order directly from the cart at any point of time in the future. Once they place the order cart will be emptied.
5. Registered users can reserve tables in a particular slot online.
6. Can see the order status once the order is placed and confirm the delivery once the order arrives. After delivery confirmation order moves to previous orders.
7. Will have the option to rate the delivery service and the food items ordered.

Manager and Staff Requirements :

1. Staff:
    a. Assigns area code for orders, then the delivery person with that area code as the primary address can see this order as an available order.
    b. Can view all the previous orders assigned by him.
    c. Maintains the kitchen by updating raw materials (consumed/bought)
    d. Takes up orders of customers via phone call or offline (People who are coming in person without any reservations).
2. Manager :
    a. Can manage all the staff
    b. Can update the menu by adding new dishes or changing prices of existing items.
    c. Can see projections of no of orders and top selling items between two given dates.
    d. Manage raw materials just like a kitchen manager.

Delivery Person Requirements :

1. Once he receives an order he can accept that order and start the ride.
2. Completes the delivery and confirms payment (just for namesake, we are not handling any payment gateways) after the customer pays.

USE CASES

1. User login/signup : Framework for user (manager, staff, customer) signup, login and logout interfaces.

2. Search menu : Customer searches for a menu item on the search bar with a name typed or can take the help of tags to view items in that category

3. Adding to Cart : Registered customers have a permanent cart and menu items along with count of that item can be added into it.

4. Order from registered customer : Registered Customers can place the order from the cart**.** Once the order is placed from the cart, the cart becomes empty.

5,6. Order from unregistered customer : Anonymous users can maintain the selected items in a cart (specific to session) . If a person wants to order without any hassle of account creation, he can order without logging in similar to registered users, now he doesn't have any old cart to get his pre-selected choices in the past

7. Assign the area code : Registered staff are given work to confirm the order from all types of modes (registered, unregistered-online, unregistered-phone) by assigning the area code the order.

8. Order via phone call : This is for the staff who places an order for an anonymous customer who interacts through a call.

9. Delivery person self assigning the order : All orders with primary region same as the delivery person’s appear top and secondary region’s appear below. He/She can self assign the order based on current status.

10. Delivery person completes the order : Delivery person completes his part of the order after the food is delivered. Customer has to still confirm the food is delivered.

11. Customer feedback : User can/skip the feedback of the menu item after the order is completed from his side too.

12,13. Ingredients Update (staff) : The system that implements the consumption and arrival of raw materials.

14. Table Booking : Registered customers can search for the availability of tables at a given time for certain hours and book accordingly

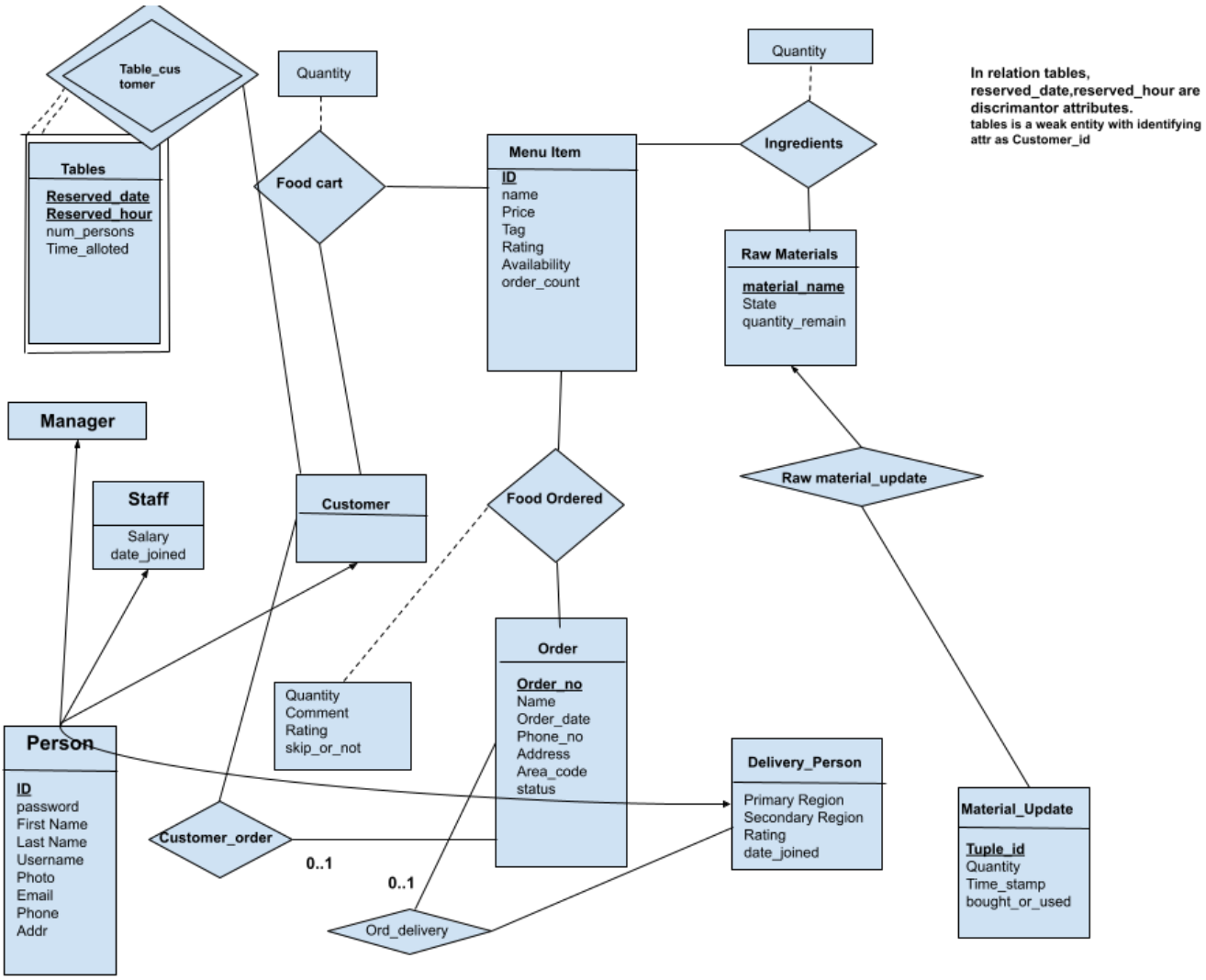
15. Analytics for manager : Managers can view no of orders per day/week/month within a time period in pictorial form. Trends will be made available as in which dish is most popular on that particular day or throughout a period.

16. User confirms the delivery : User can see the order status once the order is placed and confirm the delivery once the order arrives

17,18. Ingredients Update (manager) : Same as the case with 12,13 use cases with manager role.

DESIGN

DATABASE DESIGN



NORMALISATION

Person ID password First_Name Last_Name Username Email Phone Addr	**Customer** (Derived from person)	**Manager** (Derived from person)	**Staff** (Derived from person) Salary date_joined	**Delivery Person** (Derived from person) 1^0Region 2^0Region Rating date_joined	**Menu Item** ID name Price Tag (multi_valued) Rating Availability	**Table_slots** num_persons Reserved_date Reserved_hour cust_id time_allotted

Orders Order_No Staff_id Customer_ID (anonymous or known) Name Timestamp PhoneNo Address Area_Code Delivery Person Bill stat(status)	**Food Ordered** Order_No Item_ID Price Quantity Comment Rating Skip_or_not	**Food Cart** Customer_ID Item_ID Quantity	**Raw Materials** Name State (S or L)	**Material Update** ID material_name Quantity Timestamp Bought_or_Used	**Ingredients** Item_ID material_name Quantity

We find that the above schema is in normalized form as the only dependencies we found are of primary key on lhs for that respective table.

FINAL SCHEMA

```
drop table if exists ingredients;
drop table if exists material_update;
drop table if exists raw_materials;
drop table if exists cart;
drop table if exists food_ordered;
drop table if exists customer_order;
drop table if exists menu_item;
drop table if exists table_slots;
drop table if exists staff;
drop table if exists delivery_person;
```

```
drop table if exists customer;
drop table if exists manager;
drop table if exists person;


create table person(
    ID serial PRIMARY KEY,
    username    varchar(200) UNIQUE NOT NULL,
    pswd        varchar NOT NULL,
    first_name  text,
    last_name   text,
    email       text UNIQUE NOT NULL,
    phone       bigint NOT NULL,
    addr        text NOT NULL
);

create table customer(
) INHERITS (person);

create table manager(
)  INHERITS (person);

create table delivery_person(
    prim_region int NOT NULL,
    sec_region  int NOT NULL,
    rating      float
    check(rating >= 0 and rating <= 5),
    date_joined timestamp NOT NULL
)  INHERITS (person);

create table staff(
    salary      int NOT NULL,
    date_joined timestamp NOT NULL,
    accepted  boolean NOT NULL
)  INHERITS (person);

create table menu_item(
    ID serial PRIMARY KEY,
    name text UNIQUE NOT NULL,
    price       int NOT NULL,
    tag         text[],
    discount    int,
    dis_prem    int,
    rating      float
    check(rating >= 0 and rating <= 5),
    available   boolean NOT NULL,
    num_orders      int NOT NULL
);

create table customer_order(
    order_no serial PRIMARY KEY,
    staff_ID    int,
```

```
    customer_ID int,
    cust_name   text NOT NULL,
    time_stamp  timestamp NOT NULL,
    phone       bigint NOT NULL,
    addr        text NOT NULL,
    area_code   int,
    DP          int,
    bill        int NOT NULL,
    stat        text
    check(stat in ('yet_to_be', 'ready_to_dispatch', 'out_for_delivery', 'delivered','completed')) NOT
NULL,
    foreign key (customer_ID) references customer(ID),
    foreign key (staff_ID) references staff(ID),
    foreign key (DP) references delivery_person(ID)
);

create table food_ordered(
    order_no    int NOT NULL,
    item_ID     int NOT NULL,
    price       int NOT NULL,
    order_date  date NOT NULL,
    quantity    int NOT NULL,
    comment     text,
    rating      int
    check(rating in (0, 1, 2, 3, 4, 5)),
    skip_or_not boolean NOT NULL,
    primary key (order_no, item_ID),
    foreign key (order_no) references customer_order(order_no),
    foreign key (item_ID) references menu_item(ID)
);

create table cart(
    customer_ID int NOT NULL,
    item_ID     int NOT NULL,
    quantity    int NOT NULL,
    primary key (customer_ID, item_ID),
    foreign key (customer_ID) references customer(ID),
    foreign key (item_ID) references menu_item(ID)
);

create table raw_materials(
    mat_name    text PRIMARY KEY,
    chemistry   char(1)
    check(chemistry in ('S', 'L')) NOT NULL,
    remain      float NOT NULL
);

create table material_update(
    ID serial PRIMARY KEY,
    mat_name    text NOT NULL,
    quantity    float NOT NULL,
    time_stamp  timestamp NOT NULL,
```

```
    pos_or_neg  text
    check(pos_or_neg in ('in', 'out')) NOT NULL,
    foreign key (mat_name) references raw_materials(mat_name)
);

create table ingredients(
    item_ID     int NOT NULL,
    mat_name    text NOT NULL,
    quantity    float NOT NULL,
    primary key (item_ID, mat_name),
    foreign key (mat_name) references raw_materials(mat_name),
    foreign key (item_ID) references menu_item(ID)
);

create table table_slots(
    ID serial PRIMARY KEY,
    num_persons     int NOT NULL,
    reserved_date   date NOT NULL,
    reserved_hour   int NOT NULL,
    customer_ID     int NOT NULL,
    time_allotted   int
    check(time_allotted in (1,2,3)) NOT NULL,
    foreign key (customer_ID) references customer(ID)
);
```

DESIGN NOTES

TECHNOLOGIES

We have used node.js framework as backend and ejs, express, etc node modules to serve the front end from backend. We have used Postgres for database management.

PERFORMANCE IMPROVEMENT MEASURES

DENORMALISATION

1. We added a **date attribute in food_ordered table** although we have a timestamp in Orders, this is to help the analytics part where we filter highly sold food items in the past, so instead of going to orders and then going to food_ordered, we can directly query on food_ordered.

2. We added **num_orders in the menu_item table** although we have a food_ordered table with us, this is also for managers to know highly sold items upto now.

3. We have **quantity_remaining attribute in raw_materials table** although we material_update table, this is to have a quick info on each material remaining instead of running a query on the whole material_update table which may be huge. Quantity_remaining may be checked frequently by staff to know what they have to buy, so it's better to maintain an aggregate also.

INDEXING

1. We need to maintain index on **date** of **food_ordered** as in every user home page, he will have top selling items of today for which query is run conditioned on date.It is also useful for analytics part where we use date of food_ordered as a filtering attribute
2. index on **customer_id** of **orders** as every user who opens his site will frequently visit his orders tab to know about his latest order, so this may be helpful
3. Index on **tag** of **menu_item** since search is the most used functionality and user may see the available items with his favourite tag so we are having this index
4. Index on **stat(status)** of **orders** since delivery_persons regularly check for available orders,current_orders they picked; staff also regularly checks yet_to_be confirmed orders; all these are partitioned based on status attribute

TRIGGERS

Included triggers so that we can reduce load on the server with numerous requests.

1. Changing order count for menu item when order is placed consisting of that item
2. Deleting cart items from cart of registered customer when the order is successful
3. Updating quantity remaining of raw materials when they are used up or imported from the store

TEST RESULTS

Below test cases are from deliverable-4

Use case ID	Input	Expected Output	Result	Failure Cases
1	**Signup** : Email, username, password, name, addr, dob, mobile. **Login** : email, password	A message showing Failed or succeeded Home page	Pass	Handled
2	**Item search** : a string	A list of menu items matching that string	Pass	Handled
3	**Add to cart** : menu item	Updated cart list	Pass	Handled

4	**Place order (registered) :** cart items, personal details (editable).	New order will be placed using this ID		
5,6	**Place order (unregistered - online):** Item list with quantity, personal details	New order will be placed as anonymous	Fail	Not Handled
8	**Place order (unregistered - phone):** Item list with quantity, personal details	New order will be placed anonymous	Fail	Not Handled
7	**Assign Area Code** : Order details (Addr).	Delivery persons gets updated if one of the $1^0 or\ 2^0$ regions is same as this area code	Pass	Handled
9	**Accept the Order** : Order/addr details	The order will be removed from other Delivery person's interface	Pass	Handled
10	**Delivery** : Order ID	Delivery person should confirm 'order_delivered' first and then the user should confirm 'order_received'.	Pass	Not Handled If the user confirms 'order received' before even receiving the order then order moves to previous orders in the delivery person.
11	**Feedback :** user can(skip) feedback on any menu item ordered	NA	Pass	Not Handled User feedback takes in the delivery person and menu item rating but we didn't handle triggers for them i.e, rating for delivery person and menu item doesn't change.

12,13	**Raw Material Update** : ingredient name, quantity, state i.e. liquid/solid (for new item not in the store)	Message notifying that update is successful	Pass	Handled
14	**Table Booking** : date, time, number of persons, number of hours	A message showing confirmation or failure	Pass	Handled
15	**Projections :** All orders, from date, to date, mode	Graphs representing the data	Pass	Handled
16	**User role in Delivery** : order ID	Once user also confirms the order is delivered it becomes the previous order for both customer and Delivery person	Pass	Handled
17,18	**Raw Material Update** : ingredient name, quantity, state i.e. liquid/solid (for new item not in the store)	Message notifying that update is successful	Pass	Handled

CONCLUSION

Goals at beginning of project that we could complete :

All the above Use cases mentioned above are completed

Goals that we could not complete :

1. Un-registered customer's order (phone / online)
    a. **Desc :** use cases - 5,6,8. Un-registered customers can place the order via phone call to the staff at the restaurant or on portal without signing up.
    b. **Reason :** We have made enough queries but could not complete the front end in time.
2. Premium Customers - discount system
    a. **Desc :** System that decides discounts on menu items. Premium customers have more discounts than normal customers.

b. **Reason :** We had to skip this because we decided to work on more important parts of the project like table booking and projections.

LINK

https://github.com/bargavsaitelu/dbproj