CS 387 Project - Restaurant Management System

Project Report

30 April, 2022

IIT-Bombay

Team YARA

190050107	Satwik M
190050124	Thivesh M
190050038	Akash G
190050026	Karthikeya B

Github Repository : https://github.com/Akash-116/CS387-Project

Table of contents :-

1. Requirements
    a. Problem Description
    b. User classes and use cases
    c. User interfaces for use cases
    d. Entities and ER diagram
2. Design
    a. Tables
    b. Integrity Constraints
    c. DDL file & Insert script
    d. Transctions & SQL query
    e. Bussiness Logic Controller
    f. User Forms
3. Testing Plan
    a. Functional Test Plan
    b. Load Test Plan
4. Testing Results
    a. Functionality Testing
    b. Load Testing Results
5. Possible Improvements

1. Problem Description :

A typical Restaurant Management System has to support a complete restaurant setting, from the owner's endpoint of managing a business to the customer's endpoint of having a good dining experience. It should be neat, fast, easy-to-use, resilient to heavy loads. Analyzing the myriad of data captured and providing insights into them is also a possibility. Such predictive analytics is an important feature for businesses, helping them in making key decisions.

YARA a.k.a. Yet-Another-Restaurant-App, is a one-stop solution for managing your restaurant business. It provides an interface to both the managers and customers. Owners and personnel at various levels - waiters, chefs, inventory managers, billing managers, delivery managers can easily monitor their responsibilities such as customer orders, inventory levels, revenue for the day, online deliveries among others. Customers can login into the website and place online orders on the website after going through the menu of the restaurant.

The application is supposed to capture and deliver large amounts of data of various types, in a quick and organized fashion. As such, databases are the ideal choice over file systems. Databases provide many advantages over file system storage such as accessibility, maintaining data integrity, concurrent access to the data, data security, and data resilience. These features are necessary for a practical, day-to-day business application.

Relational databases serve our purpose. In a restaurant setting, we typically have a lot of abstraction as entities and relationships between these entities. And as such, relational databases are a good choice for modeling this abstraction. GraphDBs don't serve much purpose here since we don't have a network of relationships among entities and rather have mostly single-level relationships. Time Series DBs also don't fit with our problem statement since we don't have extensive usage of time attributes.

For developing the frontend, we are planning to use ReactJS and for the backend, NodeJS. To implement and manage databases, we are using PostgreSQL API from NodeJS. To incorporate analytics into our databases, we plan to use Apache Spark engine. Leveraging this analytics, we plan to provide various observations and patterns to both customers and managers.

2. User classes and use cases :

The classes of users are - Restaurant Manager, Chefs, Head Waiter, Waiters, Inventory Manager, Billing Manager, Delivery Manager, Delivery Persons, Customers

The use cases with the corresponding primary actors are as follows -

#1

Title: Logging on to the system

Description:

It will allow the user to log onto the website to access it.

Trigger event:

It will be both automatically Triggered.

Primary actor:

Everyone.

Inputs:

Login details such as email/username and password.

Preconditions:

No user is logged in or the previous user logged out from this computer.

Main Success Path:

The correct email and password are entered and the login is successful.

Exceptions:

Wrong password:

We will redirect to the login page and notify the user that their password is Wrong.

Postconditions:

The system will take the user to the home page.

#2

Title: Changing/Adding Dishes and Menu

Description:

It allows the user to change the available dishes and change the menu accordingly.

Trigger event:

It is Human Triggered.

Primary Actor:

Restaurant Manager, Chef

Inputs:

Dish name, new ingredients, new menu

Preconditions:

The user is logged in and has appropriate privileges.

Main Success Path:

If a new dish is to be added, then we add a new row to the dishes table and add appropriate rows to the dishes_items table

If a dish is being updated then the corresponding row is updated in the dishes table and correspondingly rows are added or deleted in the dishes_items table

Exceptions:

If some constraint fails:

Update or Adding is unsuccessful

Postconditions:

The menu will be updated and the dish added/updated.

#3

Title: Manage Table Status

Description:

Allows the user to manage the status of tables (Occupied/Empty)

Trigger event:

Human Triggered

Primary Actor:

Head waiter, Restaurant Manager

Inputs:

The new status of the table

Preconditions:

The user is logged in and has appropriate privileges.

Main Success Path:

The status column in the row corresponding to the table gets updated

Exceptions:

The Table in the question may not be present, So we will give out an error message.

Postconditions:

Table status gets changed.

#4

Title: Manage Orders and Customers

Description:

It allows users to add/remove/edit orders by customers and manage customer details.

Trigger event:

It is Human Triggered.

Primary Actor:

Restaurant Manager, Billing Manager

Inputs:

Order by customers, Details of customers.

Preconditions:

The user should have the necessary privileges and the input data should be of appropriate types.

Main Success Path:

If a new order is to be added, then we add a new row to the Orders table and add appropriate rows to the orders_dishes table

If a new customer is to be added, then we add a new row to the Customers table.

Exceptions:

If the input data types are not proper or the user does not have the required permissions we will give an error message.

Postconditions:

Order is added/updated to Orders Table or customer is added/updated to Customers Table.

#5

Title: Manage delivery People

Description:

Helps manage the delivery personnel

Trigger event:

Human Triggered

Primary Actor:

Delivery Manager, Restaurant Manager

Inputs:

Name of the person, their details to be updated

Preconditions:

The user should have the necessary privileges and the input data should be of appropriate types.

Main Success Path:

If a new person is to be added, we check if the person already exists.

If a person's details are being changed, we check that the details are valid

Exceptions:

If the input is invalid/wrong we notify the user through an error message.

If the user doesn't have the necessary permissions we redirect them back to the home page.

Postconditions:

A new delivery person is added/ the details of that person are updated

#6

Title: Manage Items

Description:

It allows users to add/remove/edit items.

Trigger event:

It is Human Triggered.

Primary Actor:

Restaurant Manager, Head Waiter

Inputs:

Details of items to modify/add.

Preconditions:

The user should have the necessary privileges and the input data should be of appropriate types.

Main Success Path:

If a new item is to be added, then we add a new row to the Items table

If an item is being updated then the corresponding row is updated in the Items table.

Exceptions:

If the input data types are not proper or the user does not have the required permissions we will give an error message.

Postconditions:

Item is added/updated to Items Table

#7

Title: Manage dishes in Cart

Description:

It allows a customer to remove/add dishes to the cart.

Trigger event:

It is Human Triggered.

Primary Actor:

Customer

Inputs:

Dish to be added/removed in cart.

Preconditions:

The customer has to be logged in.

Main Success Path:

If a new dish is to be added to the cart, then we add a new row to the Cart table

If a dish is being updated/removed then the corresponding row is updated/deleted in the Cart table.

Exceptions:

If the customer is not logged in we will redirect him/her to the login page.

Postconditions:

Dish is added/updated to Cart Table

#8

Title: View Employee details

Description:

It allows a manager to view the details of an employee

Trigger event:

It is Human Triggered.

Primary Actor:

Manager

Inputs:

Employee ID whose details he wants to view

Preconditions:

The manager has to be logged in, as only he can access these details

Main Success Path:

Get the row corresponding to the employee and show his details on the page

Exceptions:

If there is no employee with that ID then we print an error message

Postconditions:

Employee details are shown on the webpage

#9

Title: Manage Employees

Description:

It allows a manager to edit the details of an employee or delete and add employees

Trigger event:

It is Human Triggered.

Primary Actor:

Manager

Inputs:

Employee ID whose details he wants to edit or remove and the details to update

Details of the new employee he wants to add

Preconditions:

The manager has to be logged in, as only he can access these details

Main Success Path:

Update the row corresponding to the employee

Add row corresponding to the new employee

Exceptions:

If there is no employee with that ID then we print an error message

Postconditions:

Employee details are shown on the webpage

#10

Title: View menu

Description:

It allows users to view the dishes and their details and cost

Trigger event:

It is Human Triggered.

Primary Actor:

All users can view the menu

Inputs:

Dish ID if he wants to view details of one dish, otherwise look at the whole menu

Preconditions:

The user has to be logged in

Main Success Path:

Get the details of the row corresponding to all the dishes or a specific dish

Exceptions:

If there is no dish with that ID then we print an error message

Postconditions:

Menu or dish details will be given on the webpage

#11

Title: View and Edit Restaurant details

Description:

It allows the manager to view and edit restaurant details

Trigger event:

It is Human Triggered.

Primary Actor:

Manager

Inputs:

Various details of our Restaurant

Preconditions:

The user has to be logged in

Main Success Path:

Get the details of the row corresponding to all the dishes or a specific dish

Exceptions:

If there is no dish with that ID then we print an error message

Postconditions:

Menu or dish details will be given on the webpage

#12

Title: Best Waiter

Description:

It allows a manager to see the best waiter

Trigger event:

It is an Automatic Trigger

Primary Actor:

Manager

Inputs:

Nothing

Preconditions:

The manager has to be logged in, as only he can access these details

Main Success Path:

Getting data about waiter and displaying best waiter

Exceptions:

If the manager is not logged in, we redirect to the login page

Postconditions:

The best waiter is shown

#13

Title: Best Dishes according to rating

Description:

It allows everyone to see the best dish.

Trigger event:

It is an Automatic Trigger

Primary Actor:

All users

Inputs:

Nothing

Preconditions:

The user has to be logged in, as only he can access these details

Main Success Path:

Getting data about the dishes and displaying dishes with the best rating

Exceptions:

If the user is not logged in, we redirect to the login page

Postconditions:

Best rating dish is shown

#14

Title: Best Customer

Description:

It allows a manager to see the best customer

Trigger event:

It is an Automatic Trigger

Primary Actor:

Manager

Inputs:

Nothing

Preconditions:

The manager has to be logged in, as only he can access these details

Main Success Path:

Getting data about customers and displaying the best customer

Exceptions:

If the manager is not logged in, we redirect to the login page

Postconditions:

The best customer is shown

#15

Title: Most ordered dish

Description:

It allows a manager to see the most ordered dish

Trigger event:

It is an Automatic Trigger

Primary Actor:

Manager

Inputs:

Nothing

Preconditions:

The manager has to be logged in, as only he can access these details

Main Success Path:

Getting data about dishes and displaying the most ordered dish.

Exceptions:

If the manager is not logged in, we redirect to the login page

Postconditions:

The most ordered dish is shown

#16

Title: Best day of the week

Description:

It allows a manager to see the best day of the week

Trigger event:

It is an Automatic Trigger

Primary Actor:

Manager

Inputs:

Nothing

Preconditions:

The manager has to be logged in, as only he can access these details

Main Success Path:

Getting data about orders and displaying the best day of week

Exceptions:

If the manager is not logged in, we redirect to the login page

Postconditions:

The best day of the week is shown.

#17

Title: The best delivery person

Description:

It allows a manager to see the best delivery person

Trigger event:

It is an Automatic Trigger

Primary Actor:

Manager

Inputs:

Nothing

Preconditions:

The manager has to be logged in, as only he can access these details

Main Success Path:

Getting data about delivery persons and displaying the best delivery person

Exceptions:

If the manager is not logged in, we redirect to the login page

Postconditions:

The best delivery person is shown

3. User interfaces for use cases :

The user interfaces for different user types are as follows -

For Everyone:-

Use Case : Login
- form:
  - Username <text>
  - password <password>
  - New User?
  - Forgot Password?
  - Login with google
- Input:
  - Login credentials
- Output:
  - Entry into website

For Owners:-

Use Case: View Employee details
- Input:
  - Select an employee to view
- Output:
  - Employee details visible

Use Case: Delete Employee details
- Input:
  - Select the employee to Delete
- Output:
  - Employee record deleted

Use Case: Add Employee details
- Form:
  - Name
  - Address
  - Ph. No.
  - Status
  - Type
  - Salary
- Input:
  - Fill the form for adding an employee
- Output:
  - Success/Fail

Use Case: View Dish details
- Input:
  - Select dish to view details
- Output:
  - Dish details visible

Use Case: Delete dish
- Input:
  - Select dish to Delete
- Output:
  - dish record deleted

Use Case: Add Dish details
- Form:
  - Name
  - Recipe
  - Time taken
  - Cost
  - Type
  - Photo
- Input:
  - Fill the form for adding a dish
- Output:
  - Success/Fail

Use Case: View and edit Restaurant details
- Form:
  - Name
  - Address
  - Ph.No.
  - Description
  - Photo
- Input:
  - Various details of our Restaurant
- Output:
  - Success/Fail

Use Case: View Item details
- Input:
  - Select Item to view details
- Output:
  - Item details visible

Use Case: View Items purchased and used up on that day

Input:

None

Output:

Per day information on items consumed and bought

Use Case: View Table Status

Input:

Select table to view Status

Output:

Status of the table - location, status are displayed

Use Case: View order history details

Input:

Select order for details

Output:

Order details - cost, customer, Items, Mode - are displayed

Use Case: View customer details

Input:

Select Customer to view

Output:

Customer details - name, ph.no, address, no. of orders, no. of dishes ordered -are displayed

Use Case: View offer details

Input:

Select offer to display

Output:

Offer details - offerID, discount given - displayed

Use Case: View delivery person details

Input:

Select delivery person to display

Output:

Display delivery person details - Primary area, Secondary area, EmployeeID

For Customers:-

Use Case: View menu and select items

Form:

Various items are selected

Input:

User selects needed items and clicks "Next"

Output:

If successful, the user is shown the CART page

Use Case: View cart and confirm the order

Input:

Just confirms the order

Output:

Success/Failure

Use Case: View previous orders

Input:

Select order to view

Output:

Previous order details shown

Use Case: Change account details

Form:

Name

Address

Ph. No.

Input:

Input required details

Output:

Success/Failure

For Chef:-

The relevant subset of Owner use cases - (Manage Dishes,Menu,Recipe of dishes)

For Head Waiter:-

The relevant subset of Owner use cases - (Manage Table Status, Manage Waiters)

Use Case: Change table status

- Form:
  - tableID
  - location
  - Status
- Input:
  - Fill the form
- Output:
  - success/failure

For Inventory Manager :-

The relevant subset of Owner use cases - (Manages inventory of items)

Use case : Update about items

- form:
  - itemID
  - Name
  - Market Cost
  - Quantity of inventory
- Input :
  - fill form
- Output:
  - Success/failure

For Billing Manager:-

The relevant subset of Owner use cases - (Managers Orders and Customers)

Use Case: Create an order

form:

customer

dishes

Type

Offer

Input:

fill the form

Output:

Success/Failure

For Delivery Manager:-

The relevant subset of Owner use cases - (Manages Delivery Persons and Area Codes)

Use Case: Assign a delivery person

form:

Order

Customer

Delivery Person

Address

Input:

Fill the form

Output:

Success/Failure

4. Entities and ER diagram :

We have included the main entities and the information about them as their attributes in the following ER Diagram on the next page -

We represented the relations also as rectangular boxes similar to entities. For Entities, the header is darker, for relations the header is white.

5. TABLES:

1)item(item_id,item_name,cost,qua_inv,unit)
2)dish(dish_id,dish_name,recipe,time_taken,dish_type,cost,rating,photo)
3)dish_items(dish_id,item_id,quantity)
4)area(area_id,loc,city)
5)customer_type(c_type_id,c_type,min_num_dishes,max_num_dishes)
6)customer(c_id,name,username,pswd,ph_no,addr,num_orders,num_dish)
7)cart(c_id,dish_id,quantity)

8)employee(e_id,name,username,pswd,salary,ph_no,addr,e_type,join_date,status,left_dateprim_area_id,sec_area_id)
9)offer(offer_id,name,discount)
10)day(dat,day)
11)offer_valid(offer_id,dat,dish_id,c_type_id)
12)day_to_day_dishes(dat,dish_id,dish_count)
13)day_to_day_items(dat,item_id,used,bought,left_inv)

14)orders(order_id,c_id,area_id,table_id,dat,received_time,finished_time,delivered_time,deliver_person,status,order_type)
15)order_dishes(order_id,dish_id,quantity,offer_id)
16)table_status(table_id,loc,status)

There are no functional dependencies in our shema. Hence it is 3NF form.

6. INTEGRITY CONSTRAINTS:

1)item:

primary key(item_id)

2)dish:

primary key(dish_id)

dish_type in ('Veg Starter','Nog-Veg Starter','Veg Main','Non-Veg Main','Deserts')

3)dish_items:

Primary key(dish_id,item_id)

Foreign key(dish_id) references dish

Foreign key(item_id) references item

4)area:

Primary key(area_id)

5)table_status:

Primary key(table_id)

Status in (’O’,’E’) - ‘O’ is for Occupied and ‘E’ is for Empty

6)customer_type:

Primary key(c_type_id)

C_type in (‘Platinum’,’Gold’,’Silver’,’Normal’)

7)customer:

Primary key(c_id)

unique(username) - each customer should have unique username

Username not null

Pswd not null

8)cart:

Primary key(c_id,dish_id)

Foreign key(c_id) references customer

Foreign key(dish_id) references dish

9)employee:

Primary key(e_id)
unique(username) - each employee should have unique username
Username not null
Pswd not null
E_type in ('Chef','Waiter','Head Waiter','Delivery','Manager')
Status in ('Working','Leave','Left')
Foreign key(prim_area_id) references area
Foreign key(sec_area_id) references area

10)offer:

Primary key(offer_id)

11)day:

Primary key(dat)
Day in ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')

12)offer_valid:

Unique(offer_id,dat,dish_id,c_type_id)
Foreign key(offer_id) references offer
Foreign key(dat) references day - Can be null, null value means that this offer is valid on all days
Foreign key(dish_id) references dish - Can be null, that means offer is valid for all dishes
Foreign key(c_type_id) references customer_type - Can be null, that means offer is valid for all customer types
offer_id not null

13)day_to_day_dishes:

Primary key(dat,dish_id)
Foreign key(dat) references day
Foreign key(dish_id) references dish

14)day_to_day_items:

Primary key(dat,item_id)

Foreign key(dat) references day

Foreign key(item_id) references item

15)orders:

Primary key(order_id)

Foreign key(c_id) references customer

Foreign key(area_id) references area - null value means it's a Dine-in order

Foreign key(delivery_person) references employee - null value means it's a Dine-in order

Foreign key(table_id) references table_status - null value means it's a Dine-in order

Foreign key(dat) references day

Status in ('Preparing','Out for delivery','Delivered','Served')

Order_type in ('Online','Dine')

Dat not null

16)order_dishes:

Primary key(order_id,dish_id)

Foreign key(order_id) references orders

Foreign key(dish_id) references dish

Foreign key(offer_id) references offer - null value means no offer was applicable

c)Not using any views.

7. Our DDL file and files for inserting Data are in the following drive link:

https://drive.google.com/drive/folders/1HdnO0YS0Dhby2dRFf5YNTOXBc3hZKFU-?usp=sharing

8. Transactions in DB for each use case and the corresponding SQL query

For Everyone:-

Use Case : Login

1. If the role is Customer then we search for the corresponding row in the customer table -

   SELECT * FROM customer where username = u_name and pswd = pwd

2. If the role is any other then we search for the corresponding row in the employee table -

   SELECT * FROM employee where username = u_name and pswd = pwd

For Owners:-

<u>Managing Employees -</u>

Use Case: View Employee details

1. Query all existing employee records's name

   SELECT * from employee

2. Return details of a single selected employee

   SELECT * from employee where e_id= employee_id

Use Case: Delete Employee details

1. Delete employee record details from the table

   DELETE FROM employee WHERE e_id =employee_id

Use Case: Edit Employee details

2. Edit employee record details from the table

UPDATE employee SET status=new_state where e_id = employee_id

Use Case: Add Employee details

1. Add a new entry into the employee table

   INSERT INTO employee
   VALUES (name, salary, ph_no, addr, username, pswd, e_type, join_date, status, left_date, prim_area_id, sec_area_id)

Managing Dishes -

Use Case: View Dish details or Menu

1. Query all records' names from the dish table
   SELECT * from dish

2. Return details of a selected dish
   SELECT * from dish where dish_id = dish_id

Use Case: Delete dish

1. Remove the particular dish record from database
   DELETE FROM dish WHERE dish_id = dish_id;
   DELETE FROM dish_items WHERE dish_id = dish_id;

Use Case: Add Dish details

1. Insert a new entry to the dishes table
   INSERT INTO dish VALUES (dish_id, dish_name, recipe, time_taken, dish_type, cost, rating, photo);
   INSERT INTO dish_items VALUES (dish_id, item_id, quantity);

Restaurant Details

Use Case: View and edit Restaurant details

1. These details will be stored as a JSON file.

Managing Items

Use Case: View Item details

1. Query all item names from the item table.

SELECT * from item;

2. Query a particular item record from the item table.
   SELECT * from item WHERE item_id = item_id;

Use Case: View Items purchased and used up on that day

1. Fetch all items from day-to-day-items relation.
   SELECT * from day_to_day_items where day = d;

Use Case: Update item count in the inventory

1. Update the qua_inv attribute for that item in the item table
   UPDATE item SET qua_inv=new_quantity WHERE item_id = item_id;

Managing Tables

Use Case: View Table Status

1. Fetch all tables from the table relation
   SELECT * from table_status;

2. Fetch further details about a table from the table relation
   SELECT * from table_status where table_id = t_id

Use Case: Edit Table Status

1. Edit the status attribute of a table
   UPDATE table_status SET status=new_state WHERE table_id = table_id;

Managing Orders

Use Case: View order history details

1. Fetch all order's details from the orders relation

SELECT * from orders

2. Fetch complete details of a particular order from the orders relation
   SELECT * from orders where order_id = order_id

Use Case: Add an order

1. Adds a row to the orders table, adds dishes in the orders to the order_dishes table

   INSERT INTO orders
   VALUES (c_id, area_id, table_id, dat, received_time, finished_time, delivered_time, delivery_person, status, order_type)

   INSERT INTO order_dishes
   VALUES (order_id, dish_id, quantity, offer_id)

Manage Customers

Use Case: View customer details

1. Fetch details of all customers from the customers relation
   SELECT * from customer

2. Fetch complete details of a particular customer.
   SELECT * from customer WHERE c_id = c_id

Use Case: Add a new customer

1. Adding a new row to the customer table
   INSERT INTO customer
   VALUES (c_id,name,username,pswd,ph_no,addr,0,0)

Use Case: Edit customer details

1. The corresponding row in the customer table is updated
   UPDATE customer
   SET name=name AND ph_no = ph_no AND addr=addr
   WHERE c_id = customer_id

Manage Offers

Use Case: View offer details

1. Fetch all offers from the offers relation
   SELECT * from offer

2. Fetch details about a particular offer

SELECT * from offer WHERE offer_id = offer_id

Use Case: Add an offer to the offer table and offer_valid tables

1. A new row is added to the order table and rows are added to the offer_valid tables
   INSERT INTO offer
   VALUES (offer_id,name,discount)

   INSERT INTO offer_valid
   VALUES (offer_id,dat,dish_id,c_type_id)

Manage Delivery People

Use Case: View delivery person details

1. Fetch all delivery person from the employees table / delivery person table
   SELECT * from employee where e_type = 'Delivery'

Use Case: Edit delivery person details

1. Change the area codes for a particular Delivery person
   UPDATE employee SET prim_area_id=p_id and sec_area_id = s_id WHERE e_id = e_id

For Customers:- (View Menu,

Use Case: View cart and confirm the order

1. Viewing rows corresponding to the customer in the cart table
   SELECT * from cart where c_id = c_id;

2. On confirmation, insert a new order into the orders relation. - Already written

Use Case: View previous orders

1. Fetch orders from orders table, filter on a particular customer.
   SELECT * from orders where c_id = c_id;

2. Fetch a particular order detail, on selection. - Already written

Use Case: Change account details - Already written

1. Fetch existing account details from Customers table and login table.

2. Modify customer details record in the customers table.

For Chef:-

The relevant subset of Owner use cases - (Manage Dishes,Menu,Recipe of dishes)

For Head Waiter:-

The relevant subset of Owner use cases - (Manage Table Status)

For Inventory Manager :-

The relevant subset of Owner use cases - (Manages inventory of items)

For Billing Manager:-

The relevant subset of Owner use cases - (Managers Orders and Customers)

For Delivery Manager:-

The relevant subset of Owner use cases - (Manages Delivery Persons and Area Codes)

Analytics Use cases

---

1. best dishes according to user rating
   SELECT * from dish ORDER BY rating limit 10;

2. Most ordered dish

   Select dish_name from dish, (SELECT dish_id FROM order_dishes GROUP BY dish_id ORDER BY count(*) DESC
   limit 1) as a where dish.dish_id=a.dish_id;

3. Best day with most orders

   SELECT dat from orders
   GROUP BY dat

```
ORDER BY count(*) DESC
limit 1;
```

4. Most frequent customers

```
SELECT *
FROM customer,(SELECT c_id,count(*) as freq from orders group by c_id ORDER BY freq desc
limit 1) as a1
WHERE a1.c_id = customer.c_id;
```

5. Best delivery person by no.of deliveries

```
select delivery_person,name
from (SELECT delivery_person,count(*) as num_deliveries from orders where delivery_person is not null group
by delivery_person order by num_deliveries desc limit 1) as A,employee where
A.delivery_person=employee.e_id;
```

9. Business Logic Controller :-

For Everyone:-

Use Case : Login

Users need to sign up at first, before logging into the web app. From next time onwards, they can directly sign in.

For Owners:-

Use Case: View Employee details

All employees previously added via the "Add Employee" option and those already existing in the database will be visible. A particular employee's details can be brought into focus by clicking on it.

Use Case: Delete Employee details

Employee records will be deleted. There is no way to recover these details.

Use Case: Edit Employee details

An existing employee's details such as name, ph.no , job can be edited.

Use Case: Add Employee details

A new employee has to be added to the database using this option. The new employee can have the same name as another employee, since we will have a unique employee ID.

Use Case: View Dish details

All dishes added previously via the "add dish" option and those already present in the database will be displayed. A particular dish can be viewed in focus by clicking on it.

Use Case: Delete dish

This particular dish will be removed from the database. This operation is irreversible.

Use Case: Add Dish details

A new dish is added into the menu using this option. The dish name needs to be unique, to avoid confusion for customers.

Use Case: View and edit Restaurant details

Restaurant details can be edited and saved. We need a photo, address, ph.no and a neat description.

Use Case: View Item details

All items added previously via the "add item" option and those already present in the database will be displayed. A particular item can be viewed in focus by clicking on it.

Use Case: View Items purchased and used up on that day

Items used up will be reflected in the dishes-orders served. Items purchased will be managed by the Inventory Manager.

Use Case: Update item count in the inventory

Item count will be updated in the database. This has to reflect the orders placed and inventory bought.

Use Case: View Table Status

The status will be free/occupied/reserved.

Use Case: Edit table status

Can update the status of tables. Free tables can be made busy/reserved. Busy table can be freed.

Use Case: View order history details

All previous orders are displayed. Click on a particular order to view it in focus.

Use Case: Add an order

Create an order manually after taking orders from dine-in customers. Online customers will place orders under their own name.

Use Case: View customer details

All current customers are displayed. Click on a particular customer to view them in focus.

Use Case: Add a new customer

This option is used to register new customers offline.

Use Case: Edit customer details

This option is used to edit details offline.

Use Case: View offer details

All current offers are displayed. Offers typically allot discounts on total bill.

Use Case: View delivery person details

Delivery persons details are displayed. Emphasis is kept on whether they are free or currently doing a delivery.

For Customers:-

Use Case: View menu and select items

All items are displayed to the user. They can select a particular item for ordering.

Use Case: View cart and confirm the order

After viewing their items again in the cart, they can apply for an offer and confirm their order.

Use Case: View previous orders

All previous orders of that user are visible. Clicking on a particular order will open it in focus.

Use Case: Change account details

Users can change their details like name, phone no., Address etc.

For Chef:-

The relevant subset of Owner use cases - (Manage Dishes,Menu,Recipe of dishes)

For Head Waiter:-

The relevant subset of Owner use cases - (Manage Table Status, Manage Waiters)

For Inventory Manager :-

The relevant subset of Owner use cases - (Manages inventory of items)

Use case : Update about items

Details about how much of which item was bought will be updated here. This will also update the attributes of that item in the Items relation.

For Billing Manager:-

The relevant subset of Owner use cases - (Managers Orders and Customers)

For Delivery Manager:-

The relevant subset of Owner use cases - (Manages Delivery Persons and Area Codes)

Use Case: Assign a delivery person

Delivery manager will assign a delivery person to make a delivery to an online customer. Manager needs to take care of assigning the right delivery location to that delivery person.

10. User forms :

11. Testing Plan

a. Functional Testing:

Use Case	Positive Case	Negative Case
Login	If username and password are correct and a row exists in the corresponding table for that role then login is successful.	If the pair (username, password) doesn't exist in the corresponding table for that role then login is unsuccessful.
View Employee details	If User login as a manager, then opens SingleEmployeePage to view details.	If not logged in as a manager, then go to LoginPage and says that, you don't have the permission.
Delete Employee details	If the employee details entered match with a row in the database then that row is deleted.	If the employee details entered do not match with a row in the database then a error message saying "the given details don't match with any employee" is shown.
Add Employee Details	Giving all the required inputs of the employee(username,pswd,name,ph_no,addr) and the User logged in as a manager.	Not giving some field of employee(like username or pswd) or the User not logged in as a Manager.
View Dish details	If logged in as any user, then opens SingleDishPage to view Dish details.	If not logged in, then goes to LoginPage and says to Login.
Delete dish	If the dish details entered match with a row in the dish table in the database then that row is deleted.	If the dish details entered do not match with a row in the database then a error message saying "the given details donot match with any dish" is shown.
Add Dish details	Giving all the required inputs of the Dish(dish_name,dish_type,cost) and the User logged in as a manager or a Chef.	Not giving some required field of Dish(like dish_name or dish_type or cost) or the User not logged in as a Manager and Chef.
View Item details	If logged in as any employee, then opens ItemsPage to view Items details.	If not logged in as a employee, then goes to LoginPage and says that, you don't have the permission.

View Items purchased and used up on that day	If logged in as a Manager, then opens PerDayPage to view DailyUsage Details	If not logged in as a manager, then goes to LoginPage and says that, you don't have the permission.
View Table Status	If logged in as any employee, then opens TablesPage to view Tables status.	If not logged in as a employee, then goes to LoginPage and says that, you don't have the permission.
View order history details	If logged in as a Manager or Billing Manager, then opens PerDayPage to view Order History.	If not logged in as a Manager or a Billing Manager, then goes to LoginPage and says that, you don't have the permission.
View customer details	If logged in as a Manager or Customer, then opens CustomerDetailsPage.	If not logged in as a Manager or a Customer, then goes to LoginPage and says that, you don't have the permission.
View offer details	If logged in as a Manager or Customer, then opens SingleOfferPage.	If not logged in as a Manager or a Customer, then goes to LoginPage and says that, you don't have the permission.
View delivery person details	If logged in as a Manager, then opens DeliveryPersonPage	If not logged in as a Manager, then goes to LoginPage and says that, you don't have the permission.
View menu and select items	If logged in as a customer, then goes to DishesPage and selecting an item goes to SingleDishPage.	If not logged in as a Customer, then goes to LoginPage and says that, you have to Login.
View cart and confirm the order	If logged in as a customer, then goes to CartPage.	If not logged in as a Customer, then goes to LoginPage and says that, you have to Login.
View previous orders	If logged in as a customer, then shows their previous orders.	If not logged in as a customer then this page will not be accessible.
Change account details	If logged in as a customer then we will allow them to change their account details if the changes done are valid.	If not logged in as a customer then we will show an error message.

Change table status	If logged in as manager or head waiter and If the given table exists then it will change the table's status accordingly.	If not logged in as manager or if the specified table does not exist then it will give an appropriate error message.
Update about items	If logged in as manager or chef ,or automatically when a order is delivered/served. if the given items exist then they get updated.	If the login is not as a manager or chef or if the given item does not exist then we show an appropriate error message.
Create an order	If logged in as manager or head waiter or customer and the dishes are all available then the order will be accepted	If logged in as a delivery person or waiter or if any of the dishes is not available.
Assign a delivery person (This happens automatically or can be done by manager/ head waiter so if it is done manually we will also check if the person assigning the delivery person has corresponding authority to do so.)	If the given delivery person exists in the employee table and checks if they are out for delivery or on leave or they left , if not then it changes the status of the delivery person to out for delivery.	If the given delivery person doesn't exist or if he has left or on leave or out for delivery then it gives an appropriate error message.

b. Load Testing:

We plan to use Apache JMeter for load testing.
We plan to measure the time elapsed to get the results for each query for different sizes of data, we also plan to measure other statistics like number of requests that can be served per second and the throughput using JMeter.
We plan to use thread groups functionality in Jmeter to simulate load using multiple clients. We will vary the number of clients in between 1 and 100 keeping the amount of data the same and measure all the previously mentioned statistics.

We plan to measure all the statistics for different sizes of data. We will vary the data as follows -
First we will have 100s of rows in the orders, order_dishes and dish_items and 10s of rows for the remaining tables and we plan to increase the number of rows in all the tables in the order of 10 till 10,000 rows for orders table.

12. Testing Results

a. Functionality Testing

These use cases are tested in our application, which is available at the github link provided on the cover page.

The results are mainly - Pass, Fail, Doesn’t Apply.

We have decided not to implement a separate login page, but rather implement it as a popup, and so, wherever we intend to redirect to the login page, we just give an error alert and stop any submit functionality.

Use Case	Positive Case	Result	Negative Case	Result
Login	If username and password are correct and a row exists in the corresponding table for that role then login is successful.	Pass	If the pair (username, password) doesn't exist in the corresponding table for that role then login is unsuccessful.	Pass
View Employee details	If User login as a manager, then opens SingleEmployeePage to view details.	Pass- Note that we don’t have a singleEmployeePage, and rather show all employees at once with their details	If not logged in as a manager, then go to LoginPage and says that, you don't have the permission.	Pass- The unprivileged user will not have a natural browsing route to view employee details. But if visited via forced browsing, he is

				given an error message.
Delete Employee details	If the employee details entered match with a row in the database then that row is deleted.	Pass	If the employee details entered do not match with a row in the database then an error message saying "the given details don't match with any employee" is shown.	Doesn't Apply - We always show a employee from the database
Add Employee Details	Giving all the required inputs of the employee(username,pswd,name,ph_no,addr) and the User logged in as a manager.	Pass	Not giving some field of employee(like username or password) or the User not logged in as a Manager.	Pass
View Dish details	If logged in as any user, then opens SingleDishPage to view Dish details.	Pass- Note that we show details of all dishes at once, rather than alone	If not logged in, then goes to LoginPage and says to Login.	Pass
Delete dish	If the dish details entered match with a row in the dish table in the database then that row is deleted.	Pass	If the dish details entered do not match with a row in the database then an error message saying "the given details don't match with any dish" is shown.	Doesn't Apply - We always show a dish from the database
Add Dish details	Giving all the required inputs of the Dish(dish_name,dish_type,cost) and the User logged in as a manager or a Chef.	Pass	Not giving some required field of Dish(like dish_name or dish_type or cost) or the User not logged in as a Manager and Chef.	Fail- Did Not check for empty inputs

View Item details	If logged in as any employee, then opens ItemsPage to view Items details.	Pass	If not logged in as an employee, then goes to LoginPage and says that, you don't have the permission.	Pass
View Items purchased and used up on that day	If logged in as a Manager, then opens PerDayPage to view DailyUsage Details	Doesn't Apply- Didn't implement the frontend page. Implemented the backend API partially.	If not logged in as a manager, then goes to LoginPage and says that, you don't have the permission.	Doesn't Apply- Didn't implement the frontend page. Implemented the backend API partially.
View Table Status	If logged in as any employee, then opens TablesPage to view Tables status.	Pass	If not logged in as an employee, then goes to LoginPage and says that, you don't have the permission.	Doesn't Apply - No routing to Login page intended
View order history details	If logged in as a Manager or Billing Manager, then opens PerDayPage to view Order History.	Pass Implemented all orders page and not in PerDayPage	If not logged in as a Manager or a Billing Manager, then goes to LoginPage and says that, you don't have the permission.	Pass We have decided to include Head Waiter also to view orders
View customer details	If logged in as a Manager or Customer, then opens CustomerDetailsPage.	Pass	If not logged in as a Manager or a Customer, then goes to LoginPage and says that, you don't have the permission.	Pass
View offer details	If logged in as a Manager or Customer, then opens SingleOfferPage.	Pass	If not logged in as a Manager or a Customer, then goes to LoginPage and says that, you don't have the permission.	We have decided to let all users to

				view offers
View delivery person details	If logged in as a Manager, then opens DeliveryPersonPage	Pass	If not logged in as a Manager, then goes to LoginPage and says that, you don't have the permission.	Pass We have also allowed a Delivery Person to view details of other delivery people
View menu and select items	If logged in as a customer, then goes to DishesPage and selecting an item goes to SingleDishPage.	Pass	If not logged in as a Customer, then goes to LoginPage and says that, you have to Login.	Pass
View cart and confirm the order	If logged in as a customer, then goes to CartPage.	Pass	If not logged in as a Customer, then goes to LoginPage and says that, you have to Login.	Pass
View previous orders	If logged in as a customer, then shows their previous orders.	Pass	If not logged in as a customer then this page will not be accessible.	Pass
Change account details	If logged in as a customer then we will allow them to change their account details if the changes done are valid.	Pass	If not logged in as a customer then we will show an error message.	Pass- We will ask the user to login
Change table status	If logged in as manager or head waiter and If the given table exists then it will change the table's status accordingly.	Pass	If not logged in as manager or if the specified table does not exist then it will give an appropriate error message.	Pass
Update about items	If logged in as manager or chef ,or automatically when a order is delivered/served. if the given items exist then they get updated.	Pass	If the login is not as a manager or chef or if the given item does not exist then we show an appropriate error message.	Pass

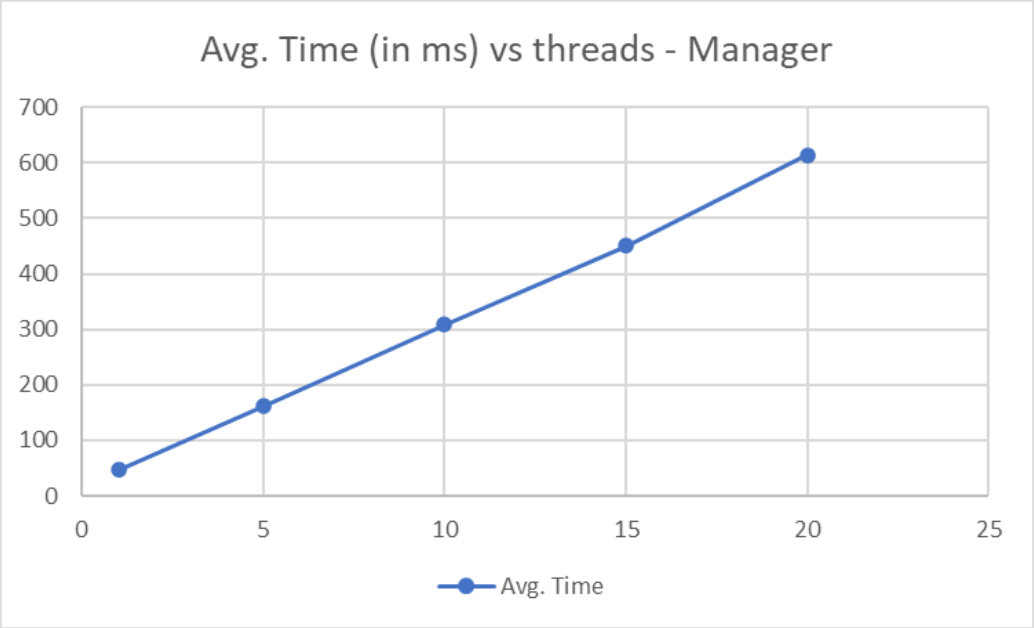
Create an order	If logged in as manager or head waiter or customer and the dishes are all available then the order will be accepted	Pass	If logged in as a delivery person or waiter or if any of the dishes is not available.	Pass
Assign a delivery person (This happens automatically or can be done by manager/ head waiter so if it is done manually we will also check if the person assigning the delivery person has correspondin g authority to do so.)	If the given delivery person exists in the employee table and checks if they are out for delivery or on leave or they left , if not then it changes the status of the delivery person to out for delivery.	Pass	If the given delivery person doesn't exist or if he has left or on leave or out for delivery then it gives an appropriate error message.	Pass

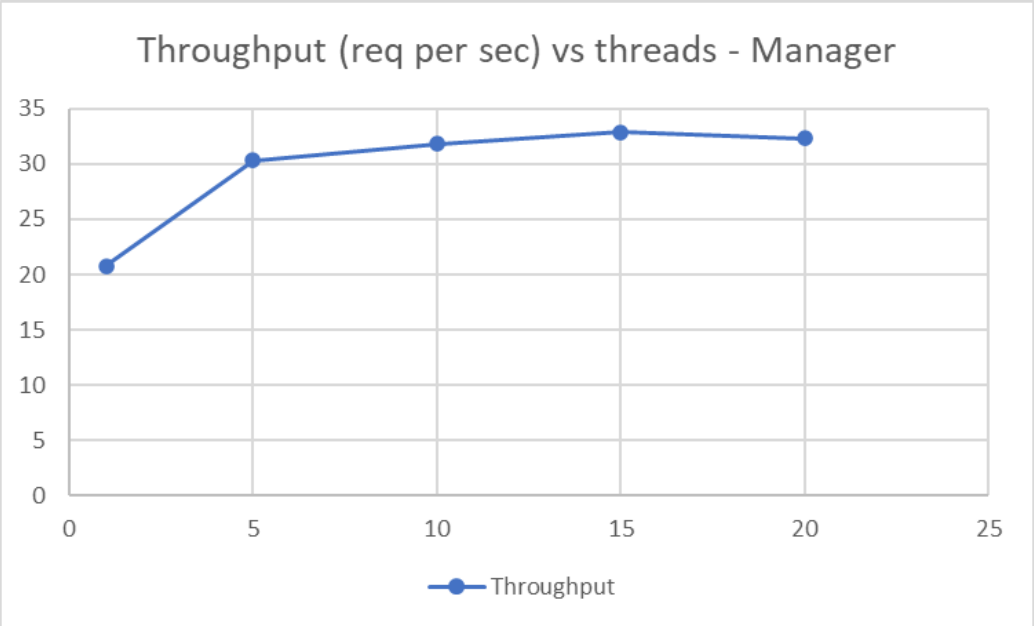
b. Load Testing

These are the results obtained for Manager, when we used 1 thread and it sends 20 loops of all the below API calls to the backend

	Avg time	Throughput(per second)	Avg. Bytes
Login	132	1.8	809
All dishes	31	1.8	3740
All offers	35	1.8	704
All customers	52	1.8	3518
All Employees	37	1.8	3609
All items	34	1.8	1048
Analytics-1	47	1.8	617
Analytics-2	33	1.8	502
Analytics-3	34	1.8	659
Delivery People	46	1.8	1044
All Tables	38	1.8	808
All Orders	50	1.8	31182
Total	47	20.8	4020

These are the plots obtained when we changed the number of threads and recorded Average Time and Throughput

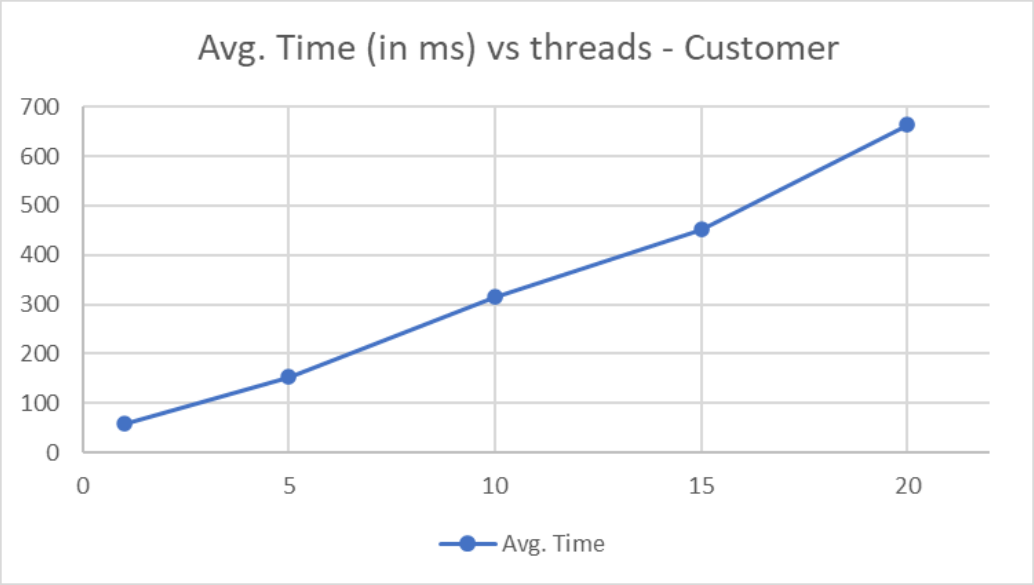

Avg. Time (in ms) vs threads - Manager
700
600
500
400
300
200
100
0
0
5
10
15
20
25
Avg. Time

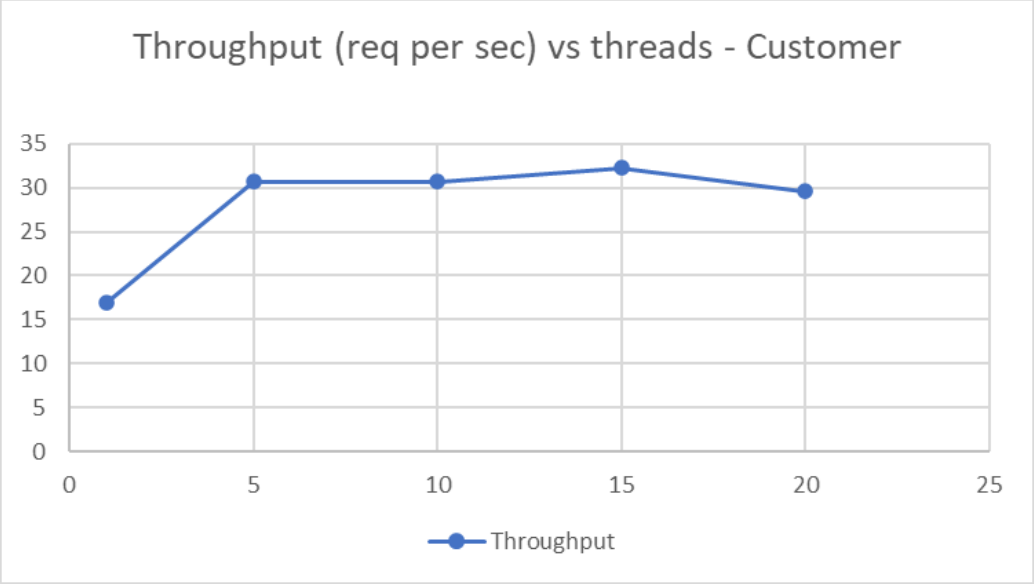

Throughput (req per sec) vs threads - Manager
35
30
25
20
15
10
5
0
0
5
10
15
20
25
Throughput

These are the results obtained for Customer, when we used 1 thread and it sends 20 loops of all the below API calls to the backend

Customers	1-Thread		
	Avg time	Throughput(per second)	Avg. Bytes
Login	127	3.5	669
Prev. Orders	38	3.5	1299
All dishes	46	3.5	3740
All offers	40	3.5	704
Cart	41	3.5	636
Total	58	16.9	1409

These are the plots obtained when we changed the number of threads and recorded Average Time and Throughput


Avg. Time (in ms) vs threads - Customer
700
600
500
400
300
200
100
0
0
5
10
15
20
Avg. Time


Throughput (req per sec) vs threads - Customer
35
30
25
20
15
10
5
0
0
5
10
15
20
25
Throughput

c. Possible Improvements and future ideas

- We have planned to but couldn't implement a separate page for each delivery person to view the current order assigned to him for delivery. We couldn't do this because the data we generated wasn't very accurate and might have more than 1 order for the same delivery person. We can try to clean our data and then we can create this page.
- We also wanted to implement a page for the chef where he can see the orders he has to prepare, but we couldn't do this because of time constraints.
- We also wanted to implement a separate page for each day to show the dishes made, orders done, items bought and used for that particular day. With the data available we couldn't implement this because it was taking a lot of time and also our data didn't have much information for a particular day.