
# 5200

# INTODUCTION TO DATABASE MANAGEMENT

# PROJECT REPORT

TITLE:

"FOODIE PALACE"

BY-

KARTHIK CHINTAMANI DILEEP

SHRADA CHELLASAMI

# ACKNOWLEGDEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the Khoury College of Computer Science at Northeastern University for providing us with a great opportunity to pursue our Master's degree in this institution.

We would like to thank our Professor, **Dr Kathleen Durant** for sparing her valuable time to extend help in every step of our project work, which paved the way for smooth progress and the fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Project work.

# TABLE OF CONTENTS

# ABSTRACT

"Good food, Good Mood". We all like eating. It is a necessary component of everyone's existence. Having the same eating cycle as our habit gets boring for all of us. Eating out can help us break this pattern, cut down on cooking time, and experience a variety of cuisines. Hence, we have chosen to build a management system for the food palace restaurant. This management system helps the restaurant manage the customers, chefs, payment and the overall administration of the business.

# INTRODUCTION

The system for managing the restaurant industry is called a restaurant management system. This management system is designed for the Foodie Palace restaurant wherein it can be used by employees in a restaurant to handle customers, their orders, payments and can help them with table reservations. The restaurant can have multiple menus. Example: Food menu, Drinks menu. Dish has a name, a price, quantity, type of cuisine and if its vegan. Each dish has a recipe which contains instructions on how to prepare it. A recipe uses a list of ingredients. Every order contains orderId, status, subtotal, item discount if any, tax, promocode if available and total amount to be paid by customer.

When it comes to creating goals, there is always one thing that every restaurant manager may have in common: enhancing a restaurant's service and profitability. One must constantly assess and comprehend a restaurant's operational costs and how they relate to a restaurant's productivity and efficiency in providing great service to its clients in order to optimize a restaurant's profitability. Management takes a very important role in controlling and manipulating the balance of costs and profitability. Often times, a restaurant’s profitability either rises or falls depending on how well it is being managed. Managing a restaurant using a well-developed software minimizes the liabilities of mismanagement and productivity loopholes. The incorporation of a Restaurant Management Software in the managing of various business processes entails that your restaurant is competitive, innovative, well-managed, and up to date with the latest management and business trends.

# README

Software Prerequisites

- Python 3 or higher version must be installed on the computer.
- Python3 download - https://www.python.org/downloads/

Python 3 Libraries to be installed

- pip3 install pandas
- pip3 install matplotlib
- pip3 install Flask
- pip3 install pymysql

Install the above libraries by:

- Option 1 – Run setup.py file in Foodie Palace folder.
- Option 2 – Install above libraries manually by the running the given commands in the terminal or command prompt.

Note: If you find any difficulties downloading any of the dependencies, please run the below command and then try running the above commands. We have also used 0.5 sleep between multiple calling of procedures in order to avoid the crash between calling multiple procedures.

pip3 install --upgrade pip

MySql workbench

Steps to run the Application –

1. Open the Foodie Palace SQL dump file in MySql workbench and run the file to create the database and import all the data associated with it.
2. Open the Foodie Palace folder and navigate to the app.py file line 623 “connectDB” function, add your database username and password credentials inside it. name: ‘root’. password - ‘password123’ at line 626 and line 627 line respectively.
3. Open the Foodie Palace folder either through an IDE or through the terminal.
4. In the terminal, run the command python3 app.py
5. The server will run on http://127.0.0.1:5000
6. Open the above link.
7. Once the website is open, click on Login button on top right corner of the screen.
8. Enter username and password as “admin” and “admin” respectively.

9. Now a new Home is open for the admin with supported operations in the navigation bar.
10. Video Link to continue: FoodiePalaceVideo.mp4

# TECHNICAL SPECIFICATIONS

- In our project we have built the front end using JavaScript, HTML, CSS.

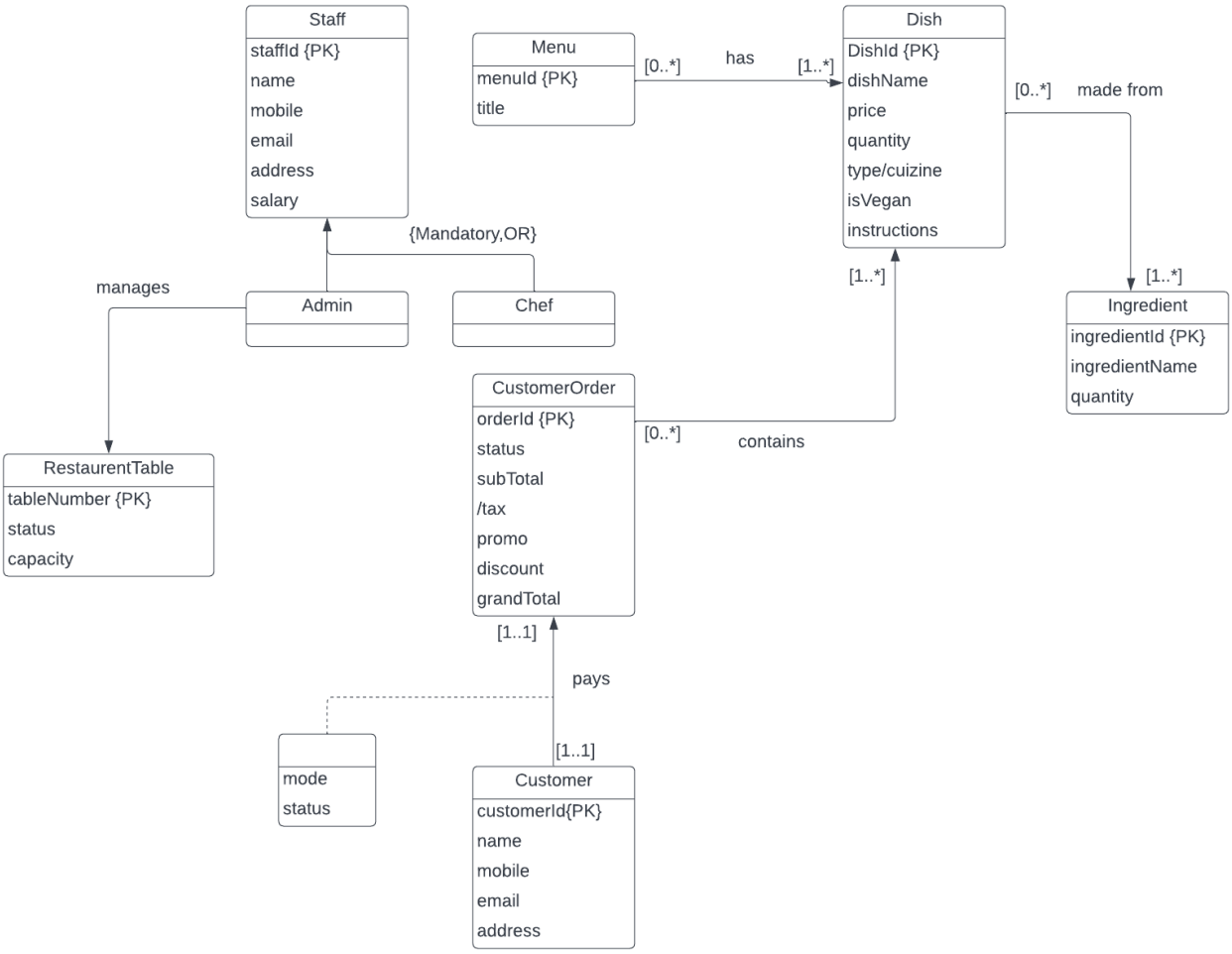
- For backend we have used MySQL where all the tables are created, procedures, functions, triggers, events are created and called.

- To connect the front-end and back-end, Python MySQL is used.

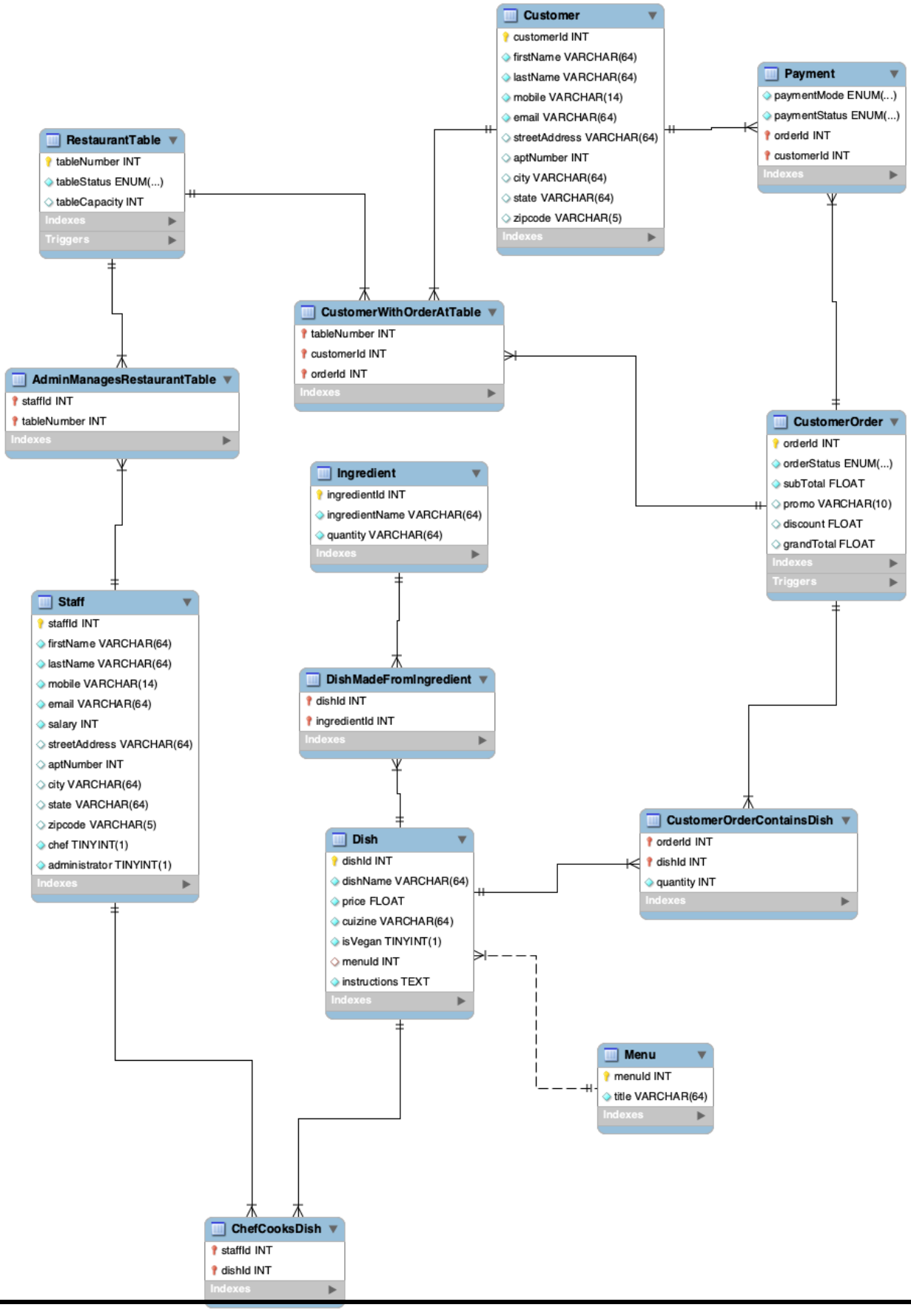
- Hardware Requirements: o Memory (RAM): 8.00 GB o System Type: 64-bit OS, x64-based processor Storage Capacity: 128 GB o CPU: 2.30GHz

- Software Requirements: Development Tools: SQL Workbench, VS Code or Anaconda o Languages: MySQL, Python, HTML, CSS, Bootstrap o Libraries: Flask, mysql-connector.

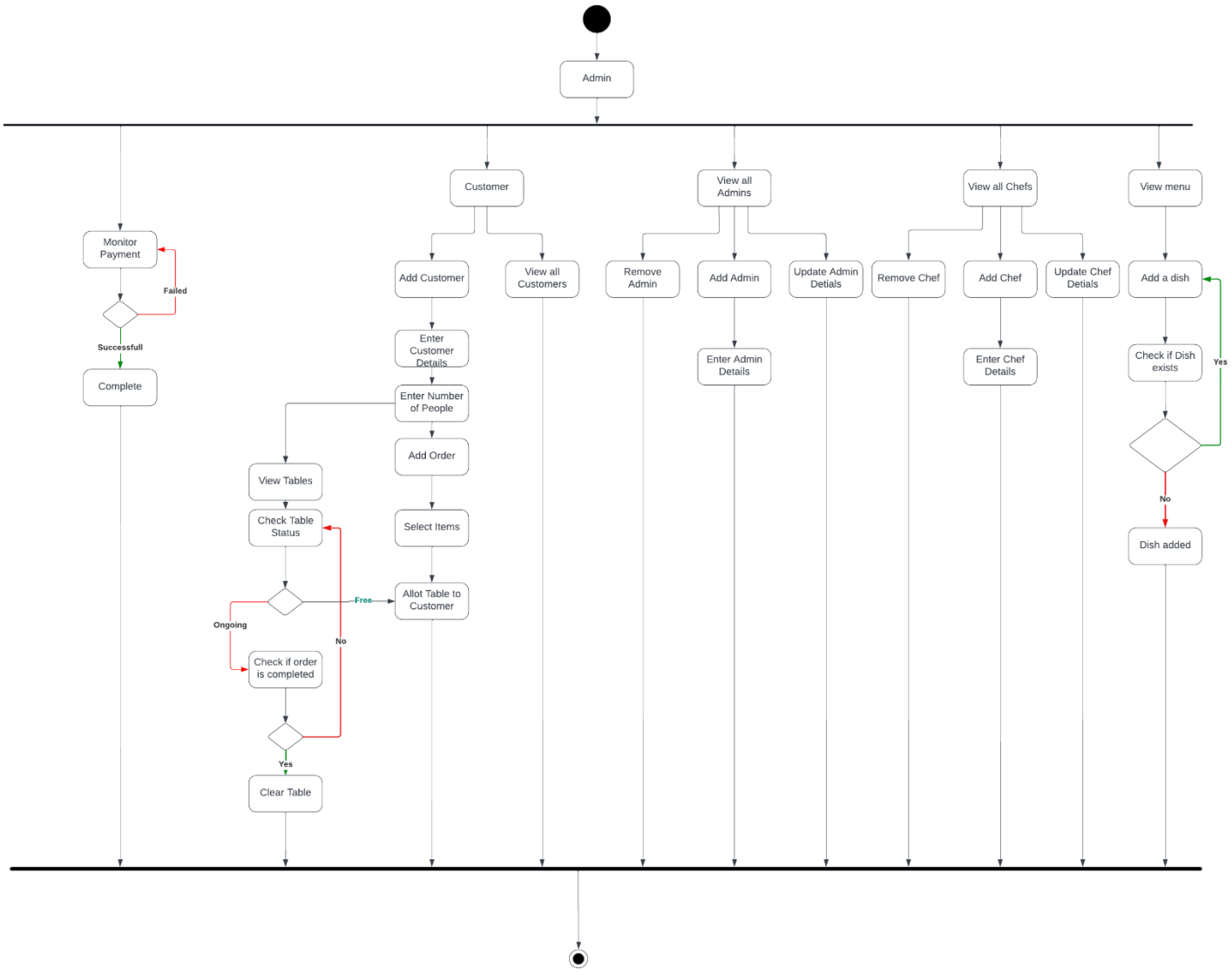
# UML DIAGRAM


Staff
staffId {PK}
name
mobile
email
address
salary
Menu
menuId {PK}
title
[0..*]
has
[1..*]
Dish
DishId {PK}
dishName
price
quantity
type/cuizine
isVegan
instructions
[0..*]
made from
[1..*]
Ingredient
ingredientId {PK}
ingredientName
quantity
{Mandatory,OR}
manages
Admin
Chef
[1..*]
CustomerOrder
orderId {PK}
status
subTotal
/tax
promo
discount
grandTotal
[0..*]
contains
RestaurentTable
tableNumber {PK}
status
capacity
[1..1]
pays
[1..1]
mode
status
Customer
customerId{PK}
name
mobile
email
address

# SCHEMA


Customer
customerId INT
firstName VARCHAR(64)
lastName VARCHAR(64)
mobile VARCHAR(14)
email VARCHAR(64)
streetAddress VARCHAR(64)
aptNumber INT
city VARCHAR(64)
state VARCHAR(64)
zipcode VARCHAR(5)
Indexes
Payment
paymentMode ENUM(...)
paymentStatus ENUM(...)
orderId INT
customerId INT
Indexes
RestaurantTable
tableNumber INT
tableStatus ENUM(...)
tableCapacity INT
Indexes
Triggers
CustomerWithOrderAtTable
tableNumber INT
customerId INT
orderId INT
Indexes
AdminManagesRestaurantTable
staffId INT
tableNumber INT
Indexes
CustomerOrder
orderId INT
orderStatus ENUM(...)
subTotal FLOAT
promo VARCHAR(10)
discount FLOAT
grandTotal FLOAT
Indexes
Triggers
Ingredient
ingredientId INT
ingredientName VARCHAR(64)
quantity VARCHAR(64)
Indexes
Staff
staffId INT
firstName VARCHAR(64)
lastName VARCHAR(64)
mobile VARCHAR(14)
email VARCHAR(64)
salary INT
streetAddress VARCHAR(64)
aptNumber INT
city VARCHAR(64)
state VARCHAR(64)
zipcode VARCHAR(5)
chef TINYINT(1)
administrator TINYINT(1)
Indexes
DishMadeFromIngredient
dishId INT
ingredientId INT
Indexes
CustomerOrderContainsDish
orderId INT
dishId INT
quantity INT
Indexes
Dish
dishId INT
dishName VARCHAR(64)
price FLOAT
cuizine VARCHAR(64)
isVegan TINYINT(1)
menuId INT
instructions TEXT
Indexes
Menu
menuId INT
title VARCHAR(64)
Indexes
ChefCooksDish
staffId INT
dishId INT
Indexes

# USER FLOW


Admin
Monitor Payment
Failed
Successfull
Complete
Customer
Add Customer
View all Customers
Enter Customer Details
Enter Number of People
Add Order
Select Items
Allot Table to Customer
View Tables
Check Table Status
Free
Ongoing
No
Check if order is completed
Yes
Clear Table
View all Admins
Remove Admin
Add Admin
Update Admin Detials
Enter Admin Details
View all Chefs
Remove Chef
Add Chef
Update Chef Detials
Enter Chef Details
View menu
Add a dish
Check if Dish exists
Yes
No
Dish added

# CRUD OPERATIONS

1. **Admin:**
    a. **Add an admin – insertStaffAdmin()**
    b. **Update details of an Admin – updateStaff()**
    c. **Remove an admin – deleteStaff()**
    d. **View all admins – getAllAdmin()**
2. **Chef:**
    a. **Add a chef - insertStaffChef()**
    b. **Update details of a chef – updateStaff()**
    c. **Remove a chef - deleteStaff()**
    d. **View all chefs – getAllChef()**
3. **Dish:**
    a. **Add a dish to any menu – insertDish()**
    b. **View all dishes – getDishOfMenu()**
4. **Customer:**
    a. **View all customers – insertCustomer()**
    b. **Add a customer – getAllCustomers()**
    c. **View a specific customer details – getCustomerDetails()**
5. **Order:**
    a. **Create an order – insertCustomerOrder()**
    b. **Add items into an order –insertCustomerOrderContainsDish**
    c. **View all items ordered at the table – getTableDetails()**
    d. **Add more items to the order - insertCustomerOrderContainsDish**
    e. **Complete the order by clearing the table - update_table_status()**
6. **Table:**
    a. **View all tables – getAllTables()**
    b. **Get table status – getRestaurantTableStatus()**
    c. **Allot a free table to customer – getFreeRestaurantTable()**
    d. **Clear a table – updateRestaurantTableStatus()**
7. **Analysis**
    a. **View few visualization graphs based on the data in the database**
8. **We have also written several triggers, functions, events. Please look into the database dump for the details.**

# LESSONS LEARNED

1. Technical expertise gained

- This project has given us an opportunity to work on several concepts of Database Management Systems.
- For this project we have used several technologies such as MySQL, Flask, HyperText Markup Language, CSS, Javascript.
- After completion of the project, we have learnt how to write procedures for inserting, updating and deleting tuples to the tables. We have learnt on how to write triggers for before insert and after insert.
- Gained experience on how to connect frontend and backend using Python SQL and Flask.
- Execution skills. Gained knowledge on how to execute the queries, solve errors, manipulate data from the database within the application.
- Gained hand on experience on writing complex SQL queries, learnt how to create conceptual and logical design for any given problem statement.
- Learnt to model represent real world entities and their relationships in form databases.

2. Insights, time management insights, data domain insights etc.

- Gained experience on how to reduce the amount of time you spend managing data, analyse data in a variety of ways.
- Learnt how to promote a disciplined approach to data management, turn disparate information into a valuable resource, improve the quality and consistency of information. Better decision making.

3. Realized or contemplated alternative design / approaches to the project

- Given the feedback on our project proposal we are now tracking the number of dishes in an order as it would be easier to track and maintain the Customer order.
- We have added "quantity" field in the CustomerOrderContainsDish table to keep track of the number of dishes in a unique order.
- Another modification to the schema taking the feedback on our project proposal was trying to keep the customer entity separate from the admin and chef entities. Previously Admin, Staff and Chef were subclasses of the User class.
- Now the Admin and Chef are subclasses of the class Staff and the Customer is maintained as a separate entity.

4. Document any code not working in this section

- Any code not working. The admin is the potential user of this restaurant management system. The login for a chef is yet to be implemented, where the chef can add instructions and ingredients of a Dish.

# FUTURE WORK

1. Planned uses of the database

- The "Foodie Palace" restaurant management system can be used for commercial purposes by restaurants with added functionalities. A commercial restaurant can manage their Menu, Chefs, Customer data, Restaurant tables using our restaurant management system. Managing the tables at a restaurant especially during weekends or holidays can be a tough task for the Admin.
- The Admin has to track which table is free and its capacity. Through this application the Admin can automatically assign tables to Customers as, once a Customer order is completed, then the admin can know which table is "Free" and the capacity of the table, then assign that table to a waiting Customer.
- Through our application the restaurant currently the Admin can track the Customer data. Customer data is very important to any business. Our management system provides a functionality where the Admin can keep track of the Customer data.

2. Potential areas for added functionality

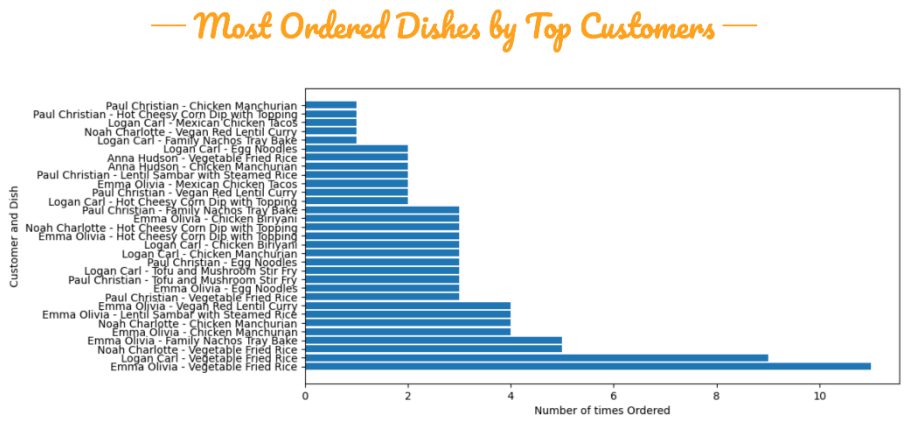
- The Chef could be a potential user. The functionality of a Chef modifying a dish or changing instructions can be implemented as part of Chef being a user.
- Customer feedback is very essential for any commercial business hence there could be a separate field for customer feedback.
- The chef can modify a dish or the admin can modify the price if many feel like a dish is overpriced, based of customer feedback.

# ADDITIONAL WORK

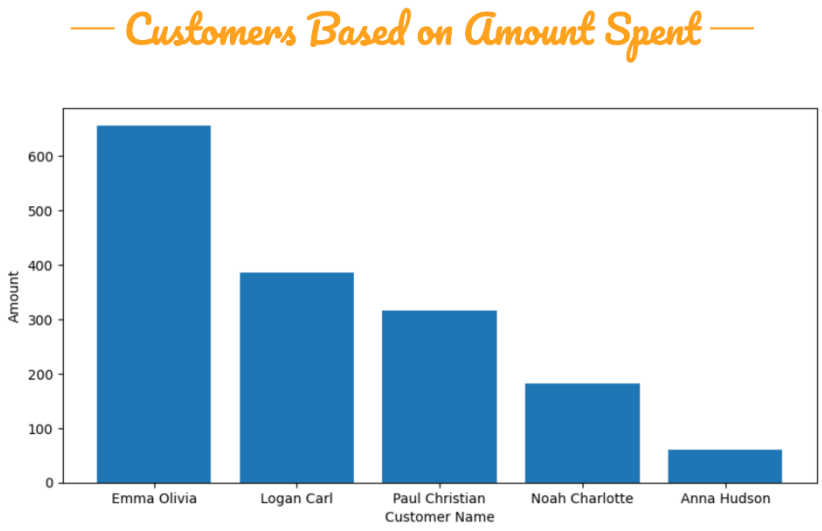
Apart from the necessary project implementation we have implemented a few bonus points questions.

1. Visualization of the data.

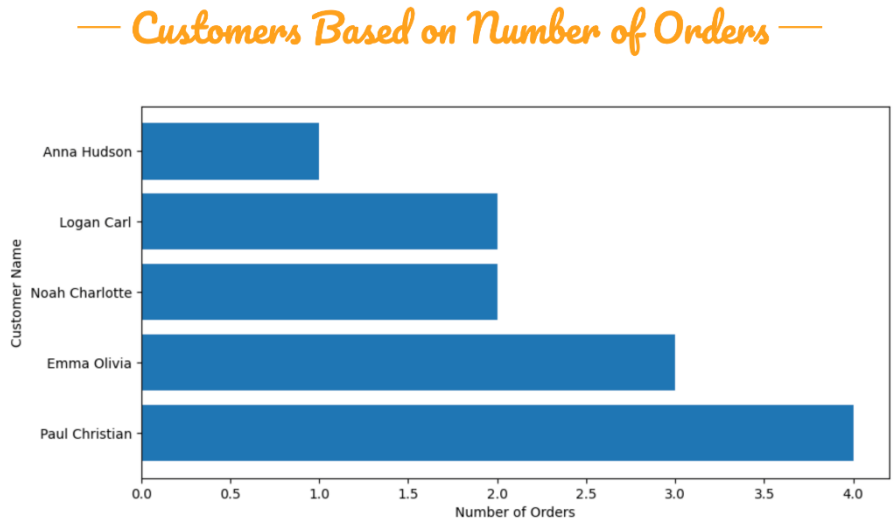
- One visualisation chart has been done for viewing the customers who have ordered the most number of dishes and the dish they have ordered.



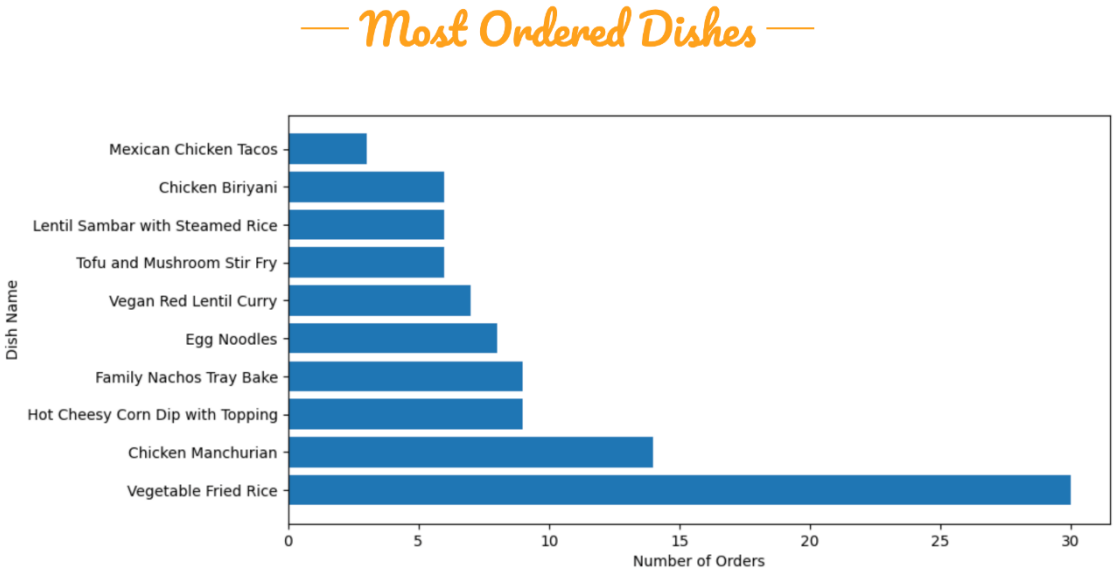

- The second visualization chart is for viewing the customers with the highest order amount.



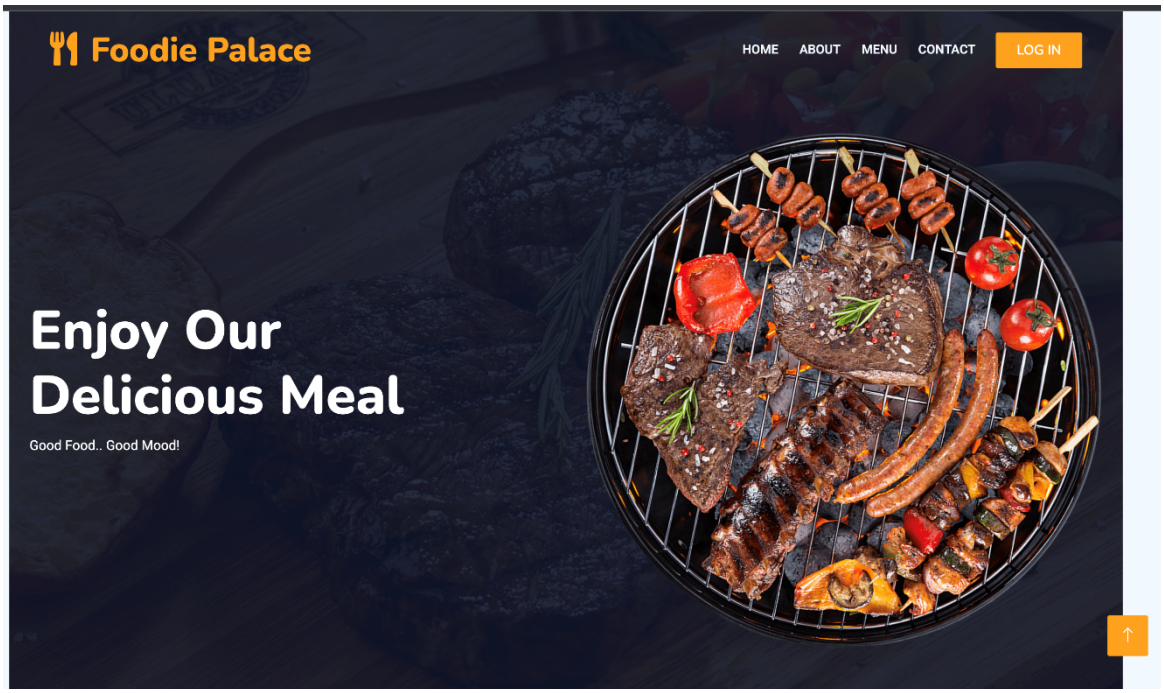
- The third visualization is for viewing the customers based on the number of orders.

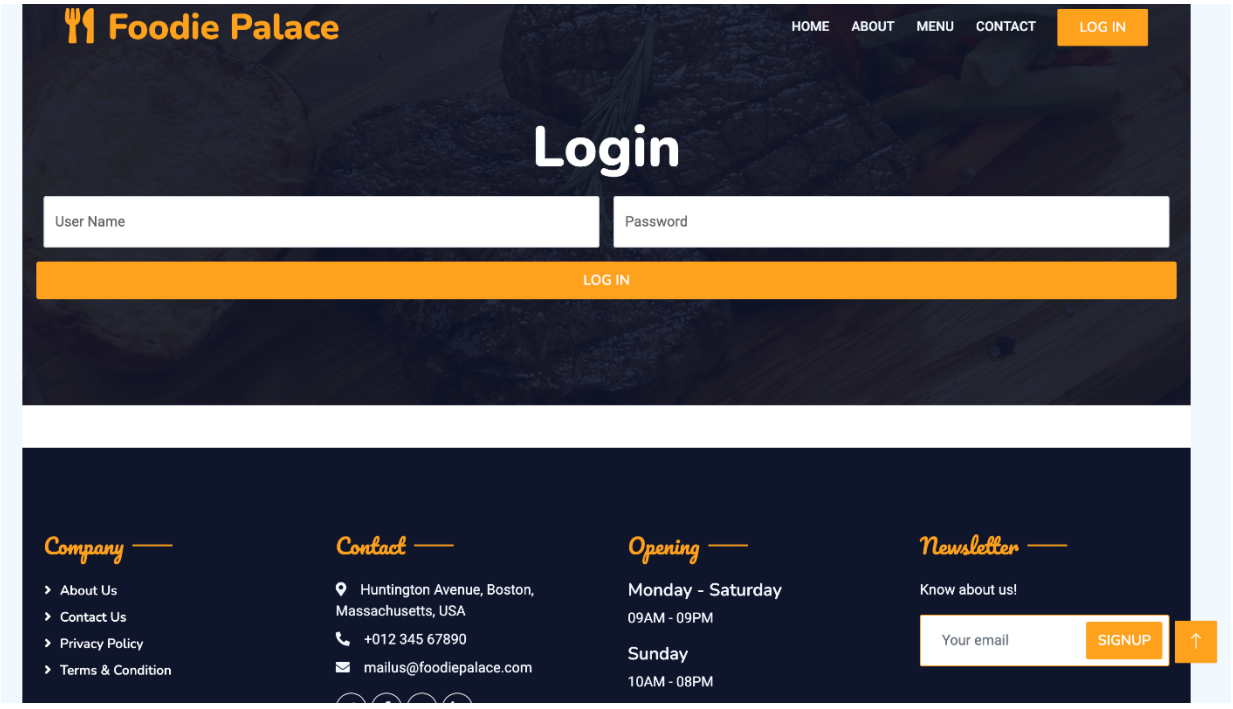


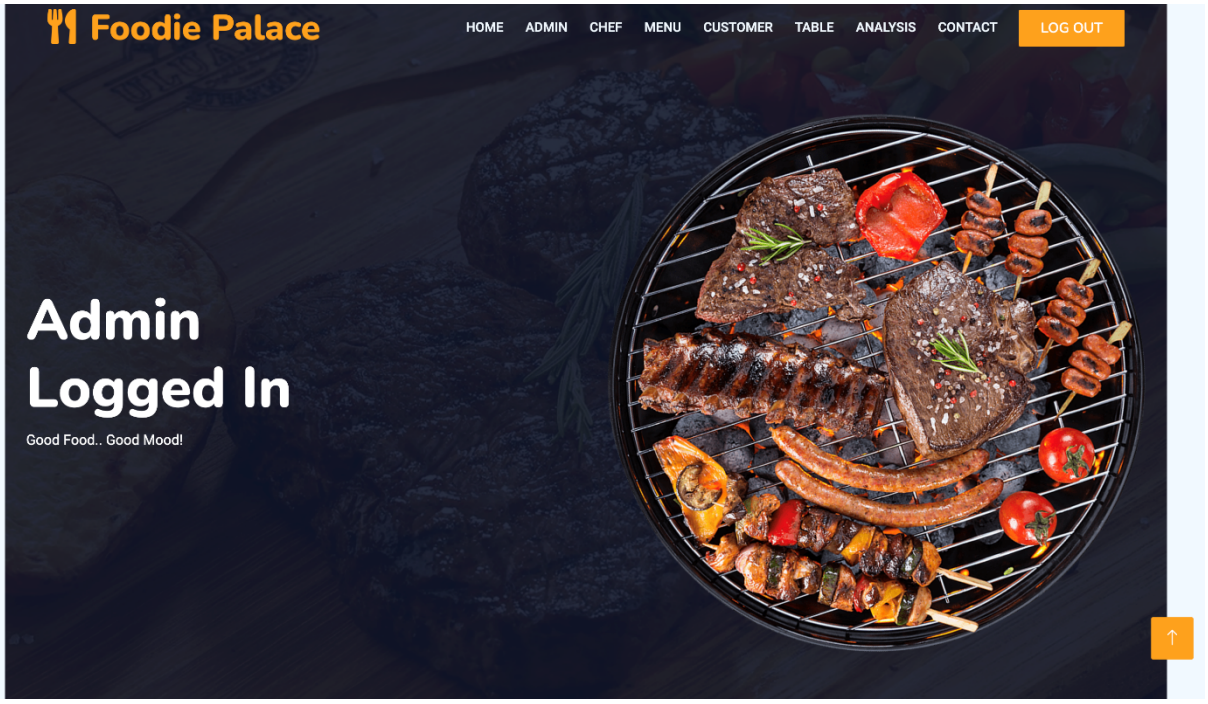

- The Fourth visualization chart is to view the most ordered dishes in the restaurant.



2. additional front-end functionality such as website or. a GUI (1-5 points)






3. 3 to 5 interesting queries that can be used for analysis or visualization of the data (at the detailed level or summary data) (1-5 points)

- sort the dishes based on number of orders

```
drop procedure if exists sortDishesBasedOnNumberOfOrders;
delimiter $$
create procedure sortDishesBasedOnNumberOfOrders()
begin
select d.dishName as DishName, COALESCE(sum(c.quantity),0) as numberOfOrders
from CustomerOrdercontainsdish c
left join dish d on d.dishId = c.dishId
group by c.dishId order by numberOfOrders desc;
end $$
```

```
delimiter ;
```

- sort customers based on number of orders

```
drop procedure if exists sortCustomersBasedOnNumberOfOrders;
delimiter $$
create procedure sortCustomersBasedOnNumberOfOrders()
begin
select concat(c.firstname," ",c.lastname) as CustomerName,
COALESCE(count(p.orderId),0) as numberOfOrders
from customer c
left join payment p on p.customerId = c.customerId
group by c.customerId
order by numberOfOrders desc;
end $$
delimiter ;
```

- sort customers based on amount paid

```
drop procedure if exists sortCustomersBasedOnAmount;
delimiter $$
create procedure sortCustomersBasedOnAmount()
begin
select concat(c.firstname," ",c.lastname) as CustomerName,
COALESCE(round(sum(o.grandTotal),2),0) as Amount
from customer c
left join payment p on p.customerId = c.customerId
left join CustomerOrder o on o.orderId = p.orderId
group by c.customerId
order by Amount desc;
```

```
end $$
delimiter ;
```

- most ordered dish by top customers

```
drop procedure if exists getMostOrderedDishByCustomers;
delimiter $$
create procedure getMostOrderedDishByCustomers()
begin
select concat(c.firstname," ",c.lastname) as CustomerName, d.dishName as
DishName, COALESCE(sum(cd.quantity),0) as MostOrderedDish
from customer c
left join payment p on p.customerId = c.customerId
left join CustomerOrder o on o.orderId = p.orderId
left join CustomerOrderContainsDish cd on cd.orderId = o.orderId
right join Dish d on d.dishId=cd.dishId
group by c.customerId, d.dishId
order by MostOrderedDish desc ;
end $$
delimiter ;
```

# Thank You