DS-5110-Restaurant Database Management System

Lasya Manthripragada
Student, MS in DS, NEU
manthripragada.l@northeastern.edu

Abstract

Restaurants are a rapidly growing business. It needs a strong and solid database structure. This application is a restaurant management system. It is aimed to decrease the manual work needed to maintain records. The application can take details from a customer, such as, their reservation date, menu etc., It also maintains all the records of the employees (non-cooking staff) as well as the chefs. Manager can perform CRUD operations on certain tables of the database, such as, Menu, inventory etc., This application has an element with data visualizations, for example, daily sales, frequent customers etc., Furthermore, a chef should be able to see the orders assigned to them, and their priorities. The entire application was made using Flask for the web framework and PHPMyadmin for the database.

Introduction

Restaurant application has many users and functionalities. For example, a customer should be able to register and login. It is the same with chefs, only a manager can register a chef or any other non-cooking staff. Manager can also insert/modify/delete reservation_details, which include the no.of tables available on each day at a particular time. Manager should be able to manage the inventory of the restaurant along with menu of the restaurant. That is, they can add any new menus and monitor the items in the restaurant, and their quantities. He can also manage prices in of the said attributes. A chef should be logged into the system to see the orders that are assigned to them, as well as the priorities. Once they are done with the order, they can update the status of the order from not done to done. To not have any confusion, only the orders placed on the present day are made visible to the chef. Flask, an open source microframework, is used to build the web framework.[1] The database used is PHPMyadmin [2] which is also an open-source software tool for database administration. Manager, chef, and a customer should have separate login tables. This is required because for every user, there different pages which are navigated. To give an example, only admin should be able to see the list of customers, employees etc., This is achieved by creating three view tables for the login page. Below are brief descriptions of different languages, software used to build this application.

Python: "Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems" [3]

PHPMyadmin: "PhpMyAdmin is a free and open-source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting services." [4]

Flask: "Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions."[5]

Database Design

Database consists of twelve tables, to accommodate all the necessary data to manage a restaurant. The database consists of seven strong entities, which are Customer, Manager, Inventory, Chef, Employees, Menu, Reservation. They are not affected by other tables i.e., their existence is not dependent on other tables. However, weak entities are dependent on other tables activity. We have order, reservation_wait, reservation_table, bill, food which are modified depending on the tables they are referenced to (Foreign keys).

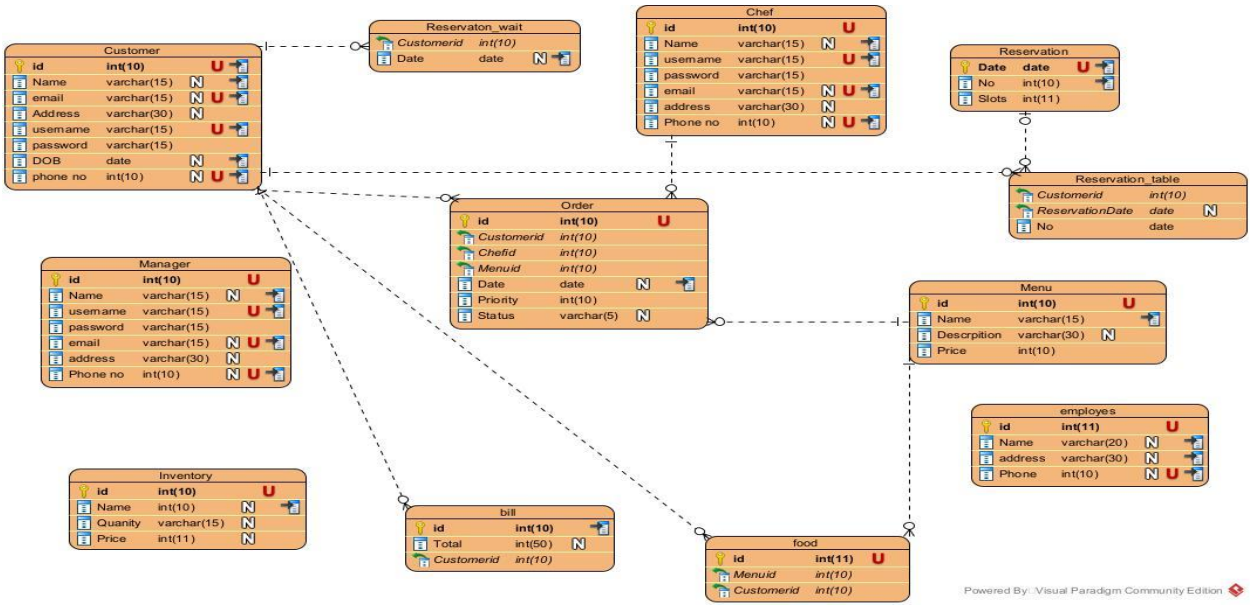

Figure-1 E-R-D for restaurant database management system

Figure-1 is the ERD for our application. Tables like customer, employees, manager, and chef contain basic information like name, address, email, phone no. It also contains, which is the most important data, usernames, and passwords. Customers have an addition column, which is the date of the birth customer. Having this information can be very useful when extracting some interesting patterns in the datasets. It helps in finding out which demographic is more attracted and/or are frequent customers to the restaurants. Manager, inventory, and employees tables don't have any relation with the other tables. Manager can insert/modify/delete data from both inventory and employees table but doesn't have any cardinal relations with each other. Inventory consists, essentially, the name and quantity of the item in the storage, like, "tomatoes-500". Menu consists of the menu item, its description, and the price. All the tables have unique ids, which are acting as primary keys. For order table, chef id, menu id, customer id that is, the customer id, the item they ordered, and the chef to whom it was assigned, are foreign keys referencing the primary keys for customer id, menu id, and chef id. Order table also consists of the date on which it was ordered, priority of the order, in this case, the id, which is auto incremented for every entry is the priority. After the order is ready to be served the chef can update the status column from not done to done.

Reservation_table consists of customer_id, who booked the table, and the date they booked the table for. Reservation table has date as the primary key, only unique dates are allowed with time slots, in this case, 4 pm to 10 pm every day, and the no. of tables available for each slot on that particular date. In reservation_table, date column of reservation table acts as the primary key to its foreign key, because date is always unique. It also contains the time during which they booked the table.

If there are no slots available on the date which customer selected, the customer will be updated in reservation_wait table. When a reservation is cancelled, the customer in the reservation_wait table will be pushed up, until there they are the first ones in the table.

Food table is useful while creating the cart feature in the application, which is elaborated in the Application description section. Bill/Check table has a unique key which is the primary key of the table. However, customer id acts as foreign key referring from the customer table which can grouped to see how much profit each customer is bringing to the restaurant.

- Functions: Applications consists of 4 functions which return the no.of unique values in each of the tables. This is useful to plot graphs.
    - Customers

    - Employers
    - Managers
    - Chef
- Views: There are login views containing only the username and password of the user.
- Trigger: Whenever, reservation_table gets updated, the date and time slot for that reserved date is decremented, on which trigger was created
- Transactions: Manage can perform CRUD operations, so for each time he modifies/inserts something, if there is an error, it should rollback, otherwise commit.
- There are 5 procedures:
    - The first procedure returns a table with the customers' reservation history. It takes the customer id as the input.
    - The second procedure returns a table with chef's order history for a particular day. It takes the chef id as the input along with the date
    - The third procedure takes the date and time and gives the no.of reservations for that day, which is one of the functionality in admin page
    - The fourth one takes the date and returns the total amount earned by the restaurant (calculated from bill/check table) on that day.
    - The final one takes date as an input and returns a table of customers who have their birthdays on the same day, this may help to give a discount or offer something special to the customer, it is a marketing strategy.

Query Optimization

While fetching data, when referencing a foreign key, or when it has index defined, the query takes less time for execution than for when there is no foreign key or index defined.

For example, when retrieving the checks for each time a customer ordered. There is index key defined on customer id. The query takes 0.0004 seconds to execute. Which is less when compared to the same query but without no index key defined.

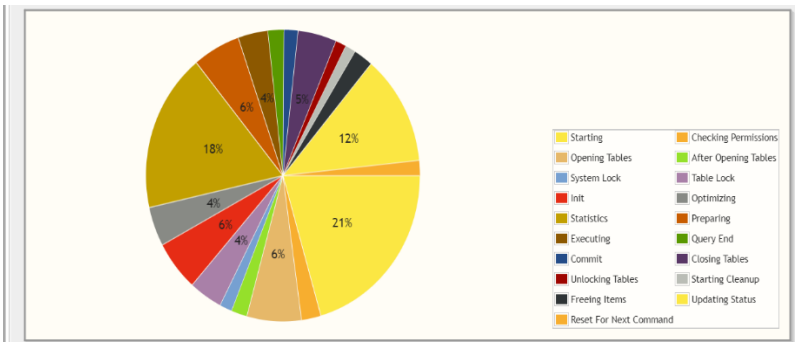

Figure-2 With index key

When the index key is removed, and the same query was executed it took 0.0008 seconds to execute, which is double the time when compared to the former.

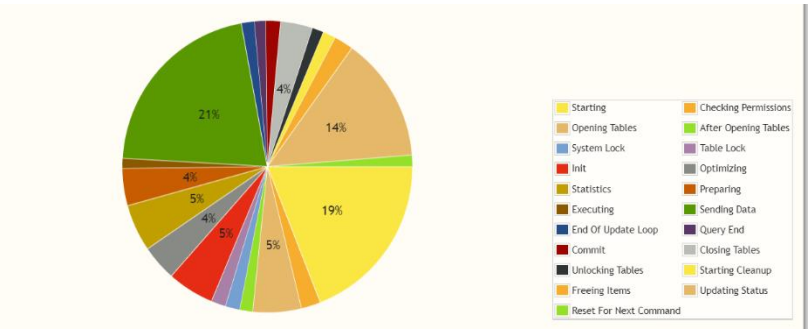

Figure-3 Without index key

In the same fashion, index keys are defined on all tables to decrease the execution time. For tables- customer, manager, chef- all the names, address, phone no., username have index keys defined on them. It drastically decreases the execution time which henceforth increases the efficiency of the application. The difference is not significant here because there is not much data in the database, but when there is big data, we can understand the difference more clearly. Foreign keys are defined the weak entities (tables which are dependent on other tables) so there is no confusion. Delete, update on cascade is defined on all the references tables so that once the entries are modified/ deleted in the original tables, it ill automatically will update the work in the referee tables. Figures 2 and 3 visualize how time is divided among many tasks with an index and without an index key.

Application Description

Application for restaurant database is developed using flask, an open source microframework, as mentioned in introduction section.[11]

```
from flask_mysqldb import MySQL
import MySQLdb.cursors
```

Mysql_host : "name of host to connect to. Default: use the local host via a UNIX socket (where applicable)"[6]

Mysql_user: "user to authenticate as. Default: current effective user."[6]

Mysql_password: "password to authenticate with. Default: no password."[6]

There are many other parameters such as the name of the databsse, database mode etc, which can set accordingly. The dataset in this application is called "Restaurant". The host is the local host, for this there is no password assigned. Application needed basic packages like "Datetime, re,". Datetime is used to get the present day's date which will be used in queries in displaying orders , reservations of that date. To build the UI- html [7], boostsrap [8], javascript [9], cdn [10] (For graphs) were used.

The application starts with a login page. If a customer doesn't have an account registered under their name, they can register by navigating to "Sign up" page. Sign up page has basic information like name, username, password, address etc., (customer table attributes) Once the customer is registered, they can, again, navigate to "sign in" page to login in.

As mentioned earlier, a manager, a customer, and a chef should be able to login, so while authenticating the application will first run the query on customer login view, then the chef's login view, and finally, the manager login. Usernames should be unique, not only in the table but among the three tables there should be unique values i.e, there shouldn't be any duplicates. If there are any, it would be a confusion as they are logging in from the same page, it would also result in compromising the security of the application. Hence, customer cannot register with a username that already exists in the database, if they try to do that, a prompt will be appeared saying it already exists, and that they should try another username. Manager can register themselves, only the database administrator can enter manager's details into the database. However, only a manager can register a chef, they themselves cannot register. For every user, there is a log out option as well.

After a successful registration, a customer can view the menu. However, they can not order unless and until they made a successful table reservation. Customers can click on "make a reservation" button, to make a reservation.

a. They cannot register in the past dates.

b. They cannot register once all the slots are filled on a particular date.

c. One the registration is done, they cannot cancel the reservation (feature is not included in this application) Hence, there is no reservation_wait feature as well.

d. After a successful reservation of a table, they can place the order by adding items into the cart. (Note: they can only order on the day of their reservation)

e. Customer can add, remove items from the cart, and they can view the total of the order.

Food table, as mentioned in database design section, is utilized during the functionality of 'cart' in the application. Note that food table is truncated once the session of that user is ended. Also, when customer finalizes the order, when they click order, the food table is truncated, the details are updated in order table during which, by using function rand (), a random chef is assigned to the order. So that chef can view, the customer id, and the orders priority. The higher the number, the lesser the ordered item as priority. After the order is placed, total is along with the customer id and the date of order are updated in bill/check table. Figures 4, 5, and 6 are a demo of how customer page look like.

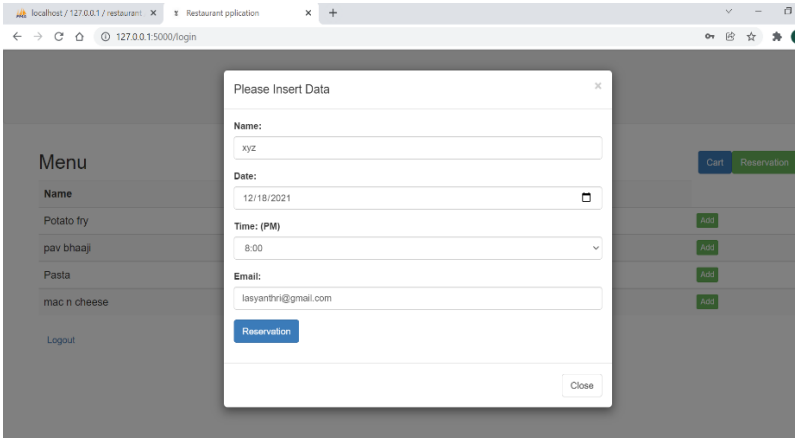

Figure 4- Reservation

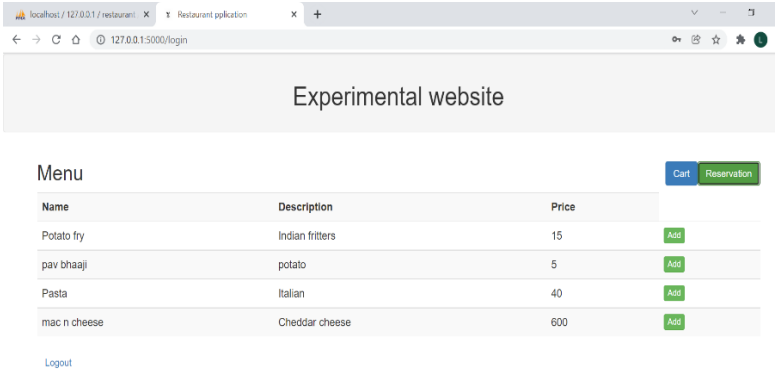

Figure 5- Menu

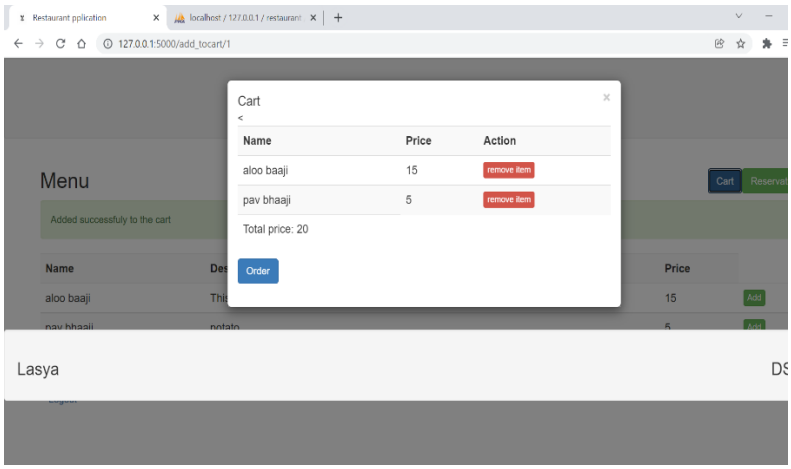

Figure 6- Cart

Chef, after a successful login, can view only present days orders. Datetime is used in finding out the date on a particular day. Chef can also view the customer id, the menu item they ordered, and the priority of each item. Once the item is done, then the chef can update the status from not done to done. Figure-7 is a demo of the chef page.

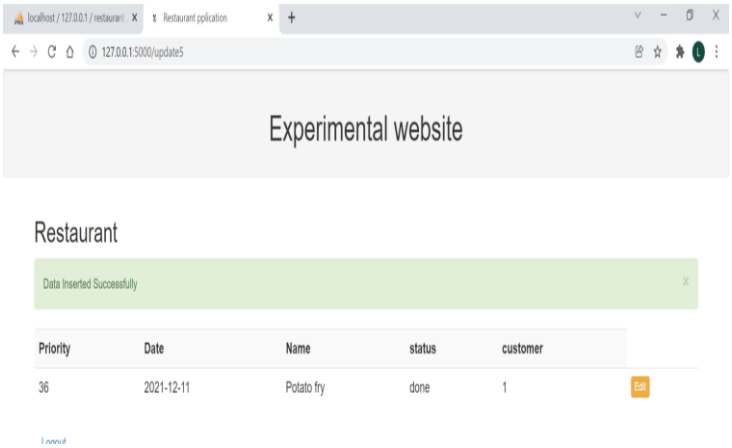

Figure-7 Chef

Manager, after a successful login, can perform CRUD operations on tables- menu, inventory, and employees. Mangere can view the list of customers but cannot perform any kind of operations. Manager can register emolpyees but employees don't have any login page. Manager can register a chef, but they cannot change the password or username once registered, CRUD operations on other columns in the chef table. Manager can view the reservations on that day but cannot perform any operations. Manager can insert reservations i.e slots (time) and their capacity for each day. Once inserted, manager cannot perform any operations on the table. Date is the primary key here, so each day has only one entry. Manager can also visualize data; all the graphs are dynamic. Figure-8 is a demo of manager page. There are eight pages, manager can navigate through all of them.

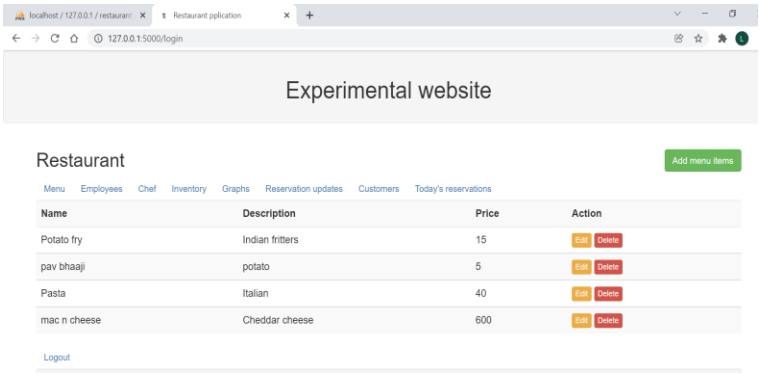

Figure 8 Manager

Data Analysis

Restaurant application contains a variety of data and can extract interesting statistics from the data. A customer can order multiple times from a restaurant. First statistics performed was top 5 frequent customers to the restaurant and the no.of times they ordered (figure-9)

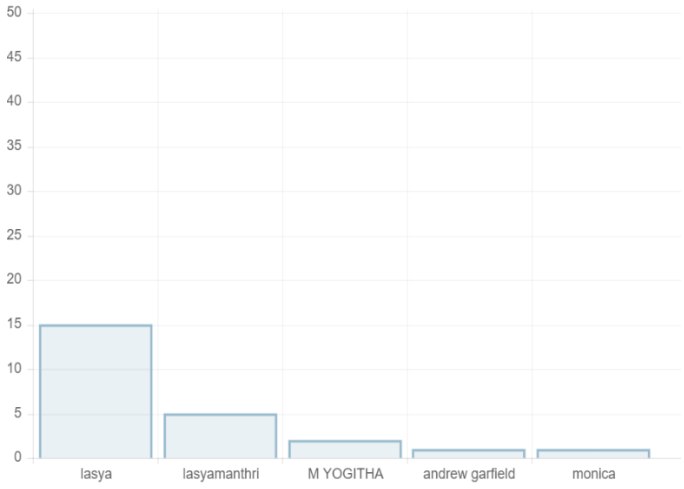

Figure-9 Top 5 frequent customers to the restaurant

In inventory, there are many items, a pie chart can be developed representing each item and their quantity. (Figure 10) If we hover over the chart, we can see the item, and their available quantity in the storage.

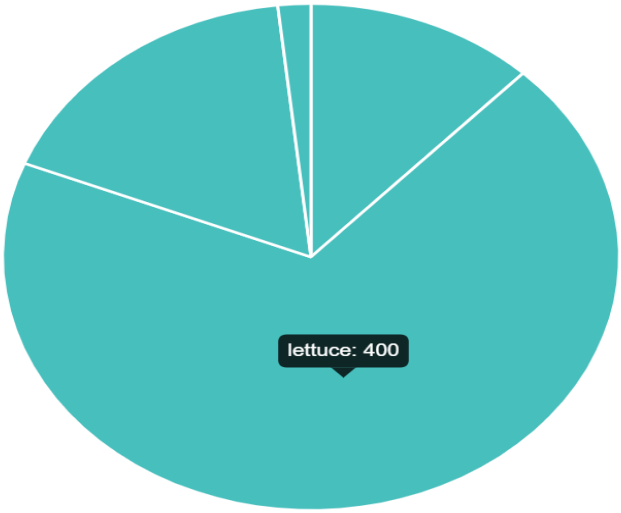

Figure-10 No.of items and their quantity

Daily sales/Weekly sales statistics are very important for a restaurant, to see how sales are going on weekdays, weekends, and to estimate sales and profit for coming days. Figure 11 depicts the weekly sales for the restaurant. The dates from bill/check table are grouped, and the total is calculated for each date. Then, the date is ordered in descending order, and top 7 dates are plotted in the bar graph. Database has only 4 dates, hence only 4 are being shown.

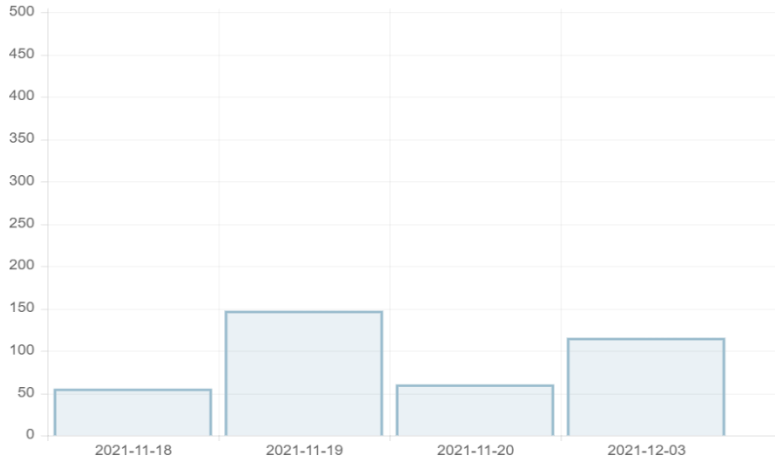

Figure-11 Daily sales

It is also crucial to know for which item the restaurant is famous for. So, a bar graph representing no.of times each menu item is ordered is also plotted.(Figure-12)

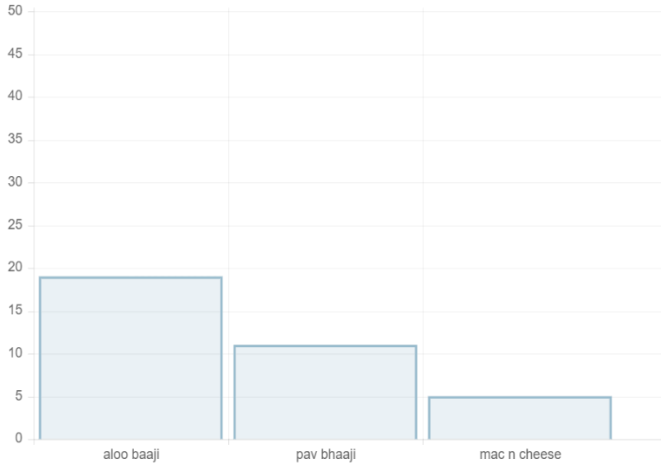

Figure-12 No.of times each menu item is ordered.

The staff ratio of the restaurant could help in understanding the investment of the restaurant. Figure 13 represents a pie chart with chefs v/s managers v/s employees ratio.

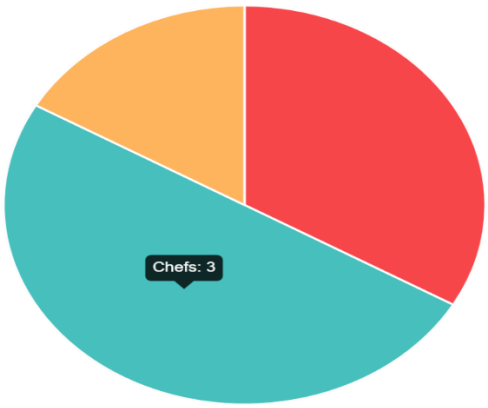

Figure 13- Ratio Chefs v/s managers v/s employees.

Conclusion and Future Scope

The restaurant database management application can register a customer. A customer can login, view the menu, reserve a table, and place an order. A chef can view the orders along with their priorities. A manager can perform CRUD operations on certain tables and can view the statistics of the data. There is no reservation_wait functionality to this application. In future, we can add a functionality where a customer can cancel their table, and customers in wait table can get a push up. We are not storing the rental prices, salaries of the staff, we can discover interesting patterns if we could do that. Manager or chef cannot register themselves, but another person should do it, for example, a chef should be registered by a manager, and a manager should be registered by the database admin, we could create a functionality where they can register themselves. There is no machine learning part to it because I couldn't collect enough real data, but if we could collect enough data, we can predict future sales by building a model on the previous week's sales data.

My advice to future DS 5110 students is that always get a team, you will understand how to coordinate with other people and always be hands on during lecture. This course has a lot of new software applications that we need to learn, it is always good to practice.

References

1. https://flask.palletsprojects.com/en/2.0.x/
2. https://www.phpmyadmin.net/
3. https://en.wikipedia.org/wiki/Python_(programming_language)
4. https://en.wikipedia.org/wiki/PhpMyAdmin
5. https://en.wikipedia.org/wiki/Flask
6. https://www.codementor.io/@adityamalviya/python-flask-mysql-connection-rxblpje73
7. https://developer.mozilla.org/en-US/docs/Web/HTML
8. https://getbootstrap.com/
9. https://www.javascript.com/
10. https://blog.ruanbekker.com/blog/2017/12/14/graphing-pretty-charts-with-python-flask-and-chartjs/
11. https://github.com/gurkanakdeniz/example-flask-crud