Restaurant Management System

REVIEW REPORT

Submitted by

Anchit Agarwal (19BCE2279)

Prithish Samanta (19BCE2261)

Tejas Jonnadula (19BCE2259)

Prepared For

DATABASE MANAGEMENT SYSTEM (CSE2004)

PROJECT COMPONENT

Submitted To

Dr. P.MohanKumar

Associate Professor

School of Computer Science and Engineering

ACKNOWLEDGMENT

We would like to express our special thanks to the VIT Chancellor, Dr. G. Viswanathan *and our teacher Dr.* Mohan Kumar for giving us the golden opportunity to do this wonderful project.

This project has allowed us to do a lot of research and has helped us to widen our knowledge in this subject. Throughout this journey we have learnt a lot of new things like connecting the website with the database, creating a good database etc. We hope to use these topics and create better and more advanced projects in the future.

Content

1. Introduction

1.1 Background

1.2 Objective

1.3 Motivation

1.4 Contributions of the Project

1.5 Organization of the Project

2. Project Resource Requirements

2.1 Software Requirements

2.2 Hardware Requirements

3. Literature Survey

4. Design of the Project

4.1 ER Diagram

4.2 ER to Relational Mapping (Schema Diagram)

5. Implementation

5.1 Introduction

5.2 Implementation

6. Snapshot

6.1 Normalization

6.2 Screenshots of connectivity code and website

6.3 Few Queries

7. Conclusion and Future Work

8. Future Work

Work break down structure template:

Team Member Registration Number	Name	Work Assigned

19BCE2261	PRITHISH SAMANTA	DESIGNED THE DATABASE, SOME WEB PAGES AND ALL THEORY RELATED WORK I.E. ER DIAGRAM, CONVERTING IT TO SCHEMA, NORMALIZATION AND PRESENTATION
19BCE2279	ANCHIT AGARWAL	DID MOST OF THE FRONTEND CODING, FULL BACKEND I.E. CREATION OF ROUTES, CONNECTING WITH DATABSE AND RUNNING ALL QUERIES
19BCE2259	TEJAS JONNADULA	IN CHARGE OF MAKING THE DOCUMENT

1. INTRODUCTION

1.1. BACKGROUND

This project is based on managing a database for restaurants. We will use the concept of relational database management system to collect and extract records of the data. The project is to ensure the smooth operation of a restaurant at peak rush hours. Not only waiting for waiters, but also staff of restaurants i.e. waiters and chefs do not have a proper display of which order to serve whom or which table.

1.2. OBJECTIVE

Our objective is to make an easy and aesthetic interface which will hide the complexity of database side programming so that anyone can use it without learning it from anyone or any prior training.

Our website’s main objective is to make sure that the customers or users are not made to wait in a restaurant for a long time just for ordering food. Using our website he or she can give their orders smoothly and also efficiently.

The customer’s responses will be prompted and updated quickly through this website and the chefs/ waiters will be able to serve our customers properly and create a wonderful experience for them.

1.3. MOTIVATION

Whenever we go to a restaurant, we need to always wait for the waiter to come to us with the menu card and then only we can give him our order. This process is very time consuming and can sometimes take more than 10 mins. This long and tedious process can anger the customers and also spoil the restaurant's reputation, which no restaurant owner wants to face. With our website, DBMS facilities this problem can be solved easily and can also help the customers too have a pleasant meal.

1.4. CONTRIBUTIONS OF THE PROJECT

Our project will ensure that everybody working at the restaurant plus the customers are happy with their meal and service at the end of the day.

We can also use the same piece of tech to provide contactless ordering in restaurants and bars in these difficult times. Since the customers only have to use the website for ordering the food they will not come in contact with the waiters while they are taking the order.

1.5. ORGANIZATION OF THE PROJECT

Our project is an essential tool for any restaurant . It is designed to keep the restaurant running smoothly and efficiently by keeping a track on the employees , inventory and sales . It is a comprehensive tool that allows the owner of the restaurant to examine the restaurant closely and its need in a glance , which can simplify the workload on a day-to-day basis .

2. PROJECTS RESOURCE REQUIREMENTS

2.1 SOFTWARE REQUIREMENTS

1. HTML - Frontend
2. CSS - Beautifying the frontend
3. BOOTSTRAP - For some predefined classes for frontend
4. TAILWIND CSS - For some predefined classes for frontend
5. TAILBLOCKS - For some predefined classes for frontend
6. JQUERY - For making web pages dynamic and adding search through text google API
7. FLASK - For making https requests, and routing i.e. defining web pages in the website
8. MYSQL_DB LIBRARY - It is imported inside Flask to connect with database (localhost in this case) and run SQL queries
9. MYSQL8.0 DATABASE COMMAND LINE CLIENT OR

   MYSQL 8.0 WORKBENCH - Setting up the Databaser

2.2 HARDWARE REQUIREMENTS

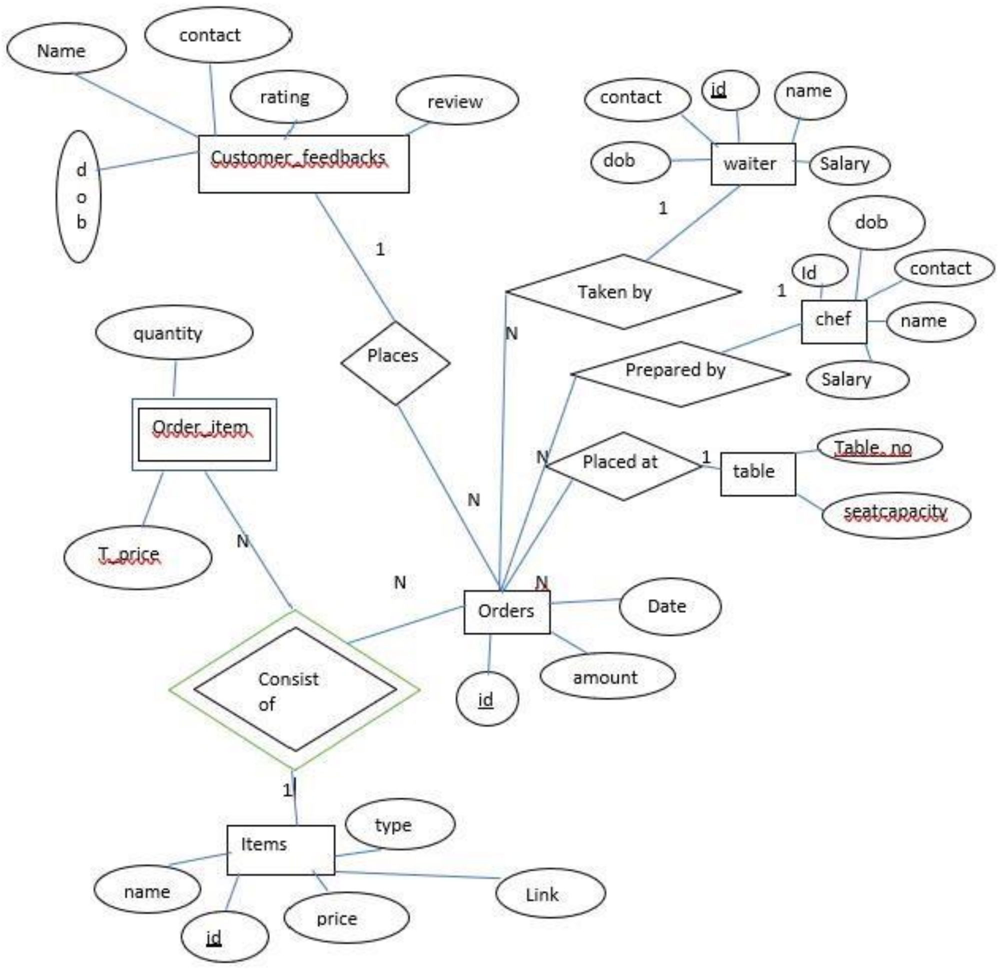
Our program will work in any computer or a laptop with MySQL DB.

3. LITERATURE SURVEY

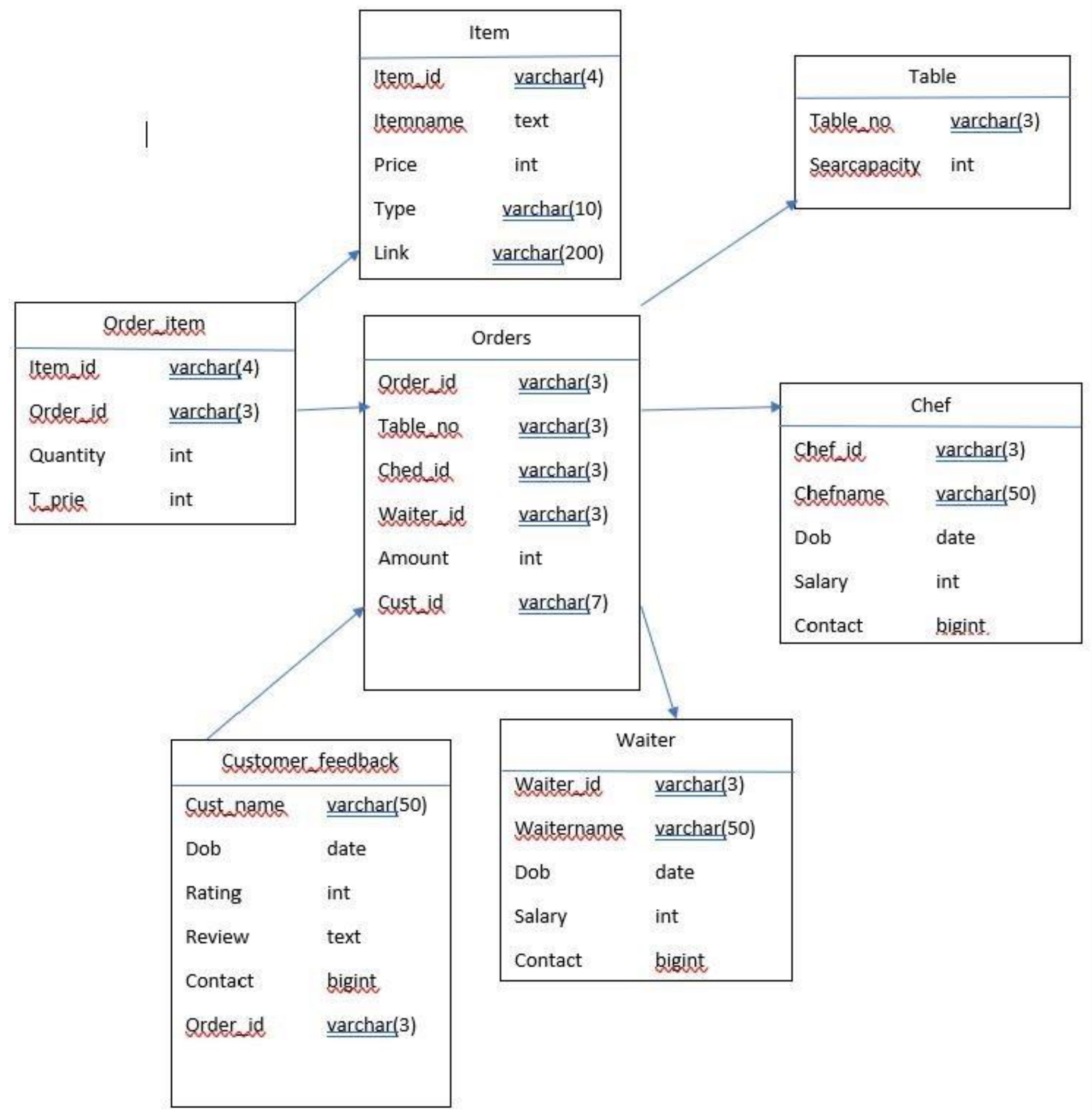
A digitalised menu completely revolutionizes the customer's dining experience . Existing methods are a bit slower and sometimes inefficient which sometimes results in the dissatisfaction of the customer which is bad for the business . Our project makes the process a lot easy and fast as compared to the pen paper method used . Instead of going through a long menu we can just swipe , flip and tap to order our favourite dishes . Our main aim is to increase the efficiency of the food ordering , reduce human errors and provide high quality service to the customers of the restaurants .

4. DESIGN OF THE DATABASE

4.1. ER DIAGRAM



4.2. ER TO RELATION SCHEMA



5. IMPLEMENTATION

5.1 INTRODUCTION

This part details the various technologies used in the development of the restaurant management system including the programming languages used for the database and connection . There are certainly a number of npm packages we have installed in order to make our website fully functional . We have connected our frontend and backend part, completed the setup of the server and have added a database to store all the information.

5.2 IMPLEMENTATION

We have implemented the frontend part of our website using html, css, bootstrap, tailwind css, and tailblocks.

In order to make the frontend dynamic, we have used jQuery and also some google apis for adding search blocks i.e. filters in forms.

For the database, we have used mysql RDBMS with mysql 8.0 workbench.

Creation of so many different routes i.e. different webpages, extraction of data (GET), uploading data (POST) we have used FLASK.

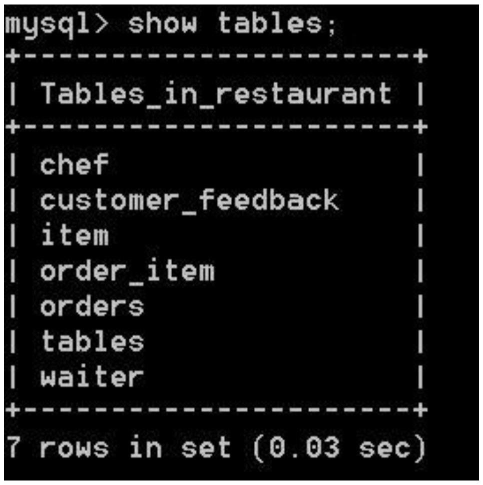
Last but not the least, we have used the mysql_db library in flask to connect to the database and running queries.

For now, we have kept localhost as our server. Phpmyadmin can be used as an alternative to this.

6. SNAPSHOTS AND NORMALIZATION

6.1. NORMALIZATION

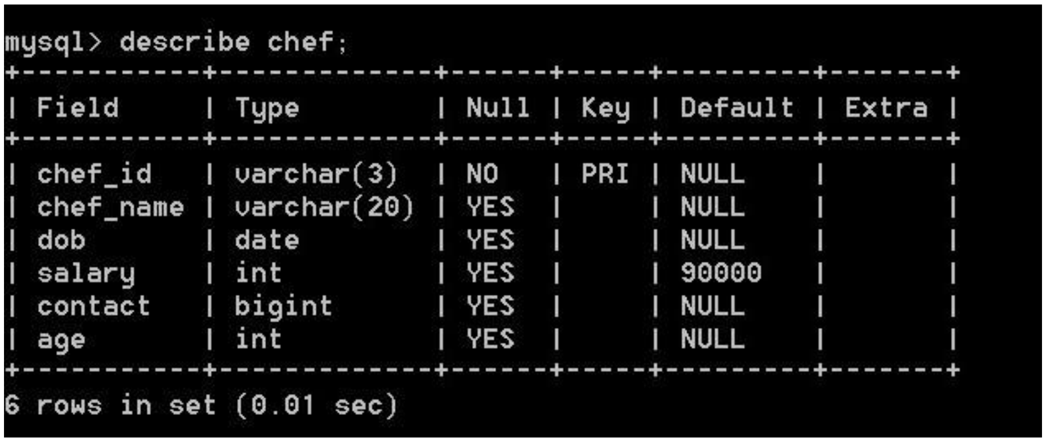
Our database contains these 7 tables



```
mysql> show tables;
+----------------------+
| Tables_in_restaurant |
+----------------------+
| chef                 |
| customer_feedback    |
| item                 |
| order_item           |
| orders               |
| tables               |
| waiter               |
+----------------------+
7 rows in set (0.03 sec)
```

All the above tables are in BCNF NORMAL FORM (i.e) the LHS of each relation in the database is either a Super Key or a Candidate Key.

chef table:-



```
mysql> describe chef;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| chef_id   | varchar(3)  | NO   | PRI | NULL    |       |
| chef_name | varchar(20) | YES  |     | NULL    |       |
| dob       | date        | YES  |     | NULL    |       |
| salary    | int         | YES  |     | 90000   |       |
| contact   | bigint      | YES  |     | NULL    |       |
| age       | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

R(chef_id, chef_name, dob, salary, contact, age)

chef_id is the candidate key chef_id ---->

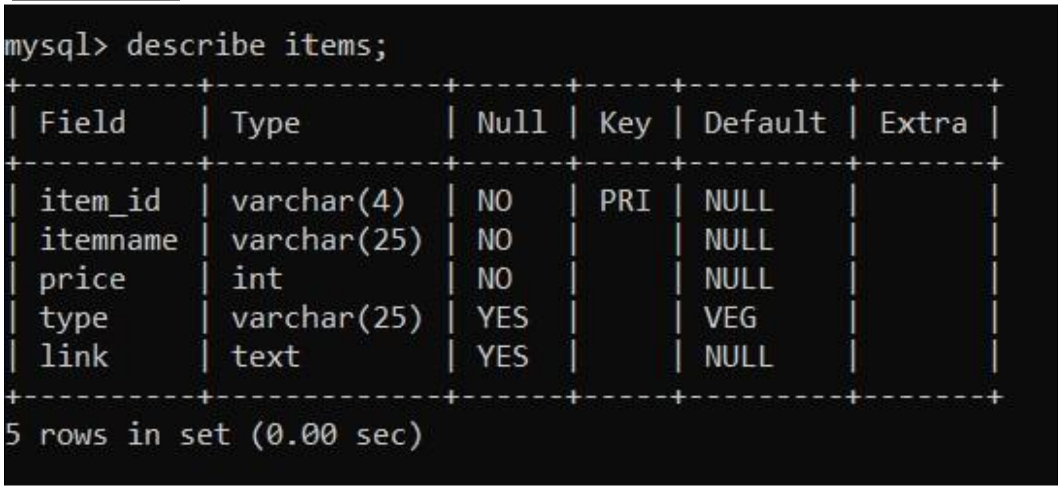
chef_name chef_id ----> dob chef_id ---->

salary chef_id ----> contact chef_id ---->

age

The Table is in BCNF as it is Third Normal Form and the LHS of each functional dependency is a Candidate key.

<u>item table</u>:-



```
mysql> describe items;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| item_id  | varchar(4)  | NO   | PRI | NULL    |       |
| itemname | varchar(25) | NO   |     | NULL    |       |
| price    | int         | NO   |     | NULL    |       |
| type     | varchar(25) | YES  |     | VEG     |       |
| link     | text        | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

R(item_id, price, link, type, item_name)

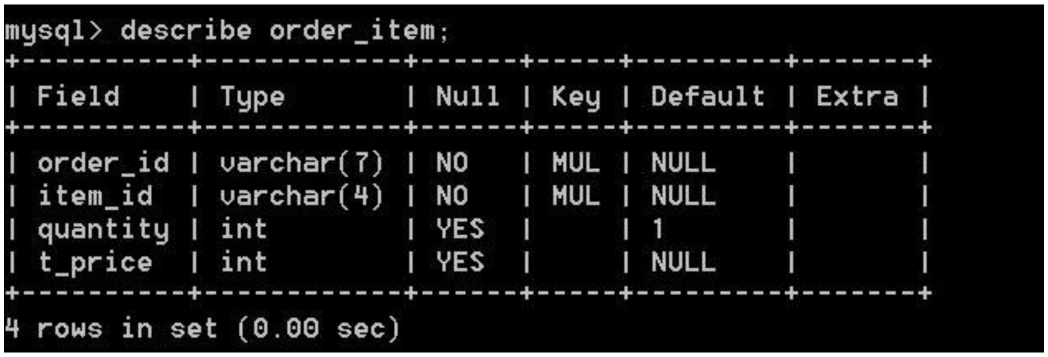
item_id is the candidate key item_id -----

> price item_id -----> type item_id

-----> link item_id -----> item_name

The Table is in BCNF as it is Third Normal Form and the LHS of each functional dependency is a Candidate key.

<u>order_item table</u>:-



```
mysql> describe order_item;
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| order_id | varchar(7) | NO   | MUL | NULL    |       |
| item_id  | varchar(4) | NO   | MUL | NULL    |       |
| quantity | int        | YES  |     | 1       |       |
| t_price  | int        | YES  |     | NULL    |       |
+----------+------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

R(order_id, item_id, quantity, t_price) order_id

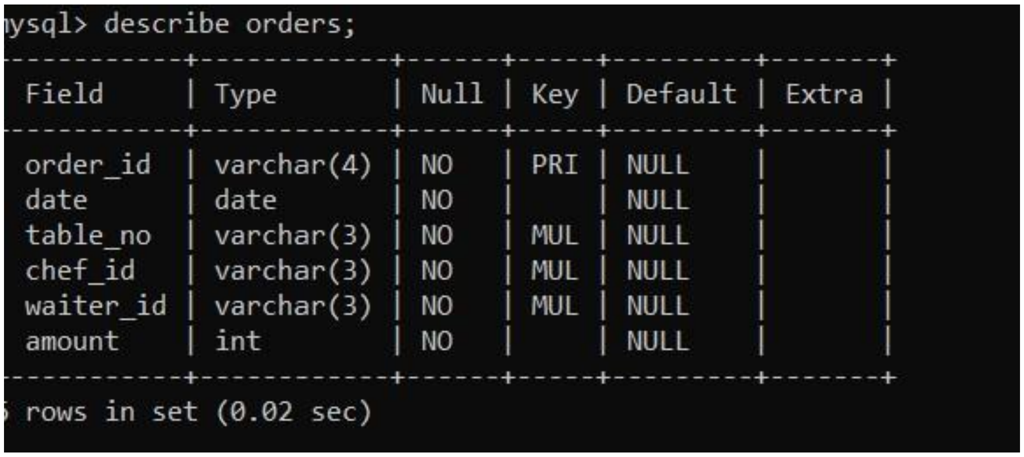
and item_id together form the candidate key.

(order_id, item_id) -----> quantity

(order_id, item_id) -----> t_price

The Table is in BCNF as it is Third Normal Form and the LHS of each functional dependency is a Candidate key.

orders table:-



```
ysql> describe orders;
-----------+-----------+------+-----+---------+-------+
 Field     | Type      | Null | Key | Default | Extra |
-----------+-----------+------+-----+---------+-------+
 order_id  | varchar(4) | NO  | PRI | NULL    |       |
 date      | date       | NO  |     | NULL    |       |
 table_no  | varchar(3) | NO  | MUL | NULL    |       |
 chef_id   | varchar(3) | NO  | MUL | NULL    |       |
 waiter_id | varchar(3) | NO  | MUL | NULL    |       |
 amount    | int        | NO  |     | NULL    |       |
-----------+-----------+------+-----+---------+-------+
 rows in set (0.02 sec)
```

R(order_id, date, table_no, chef_id, waiter_id, amount)

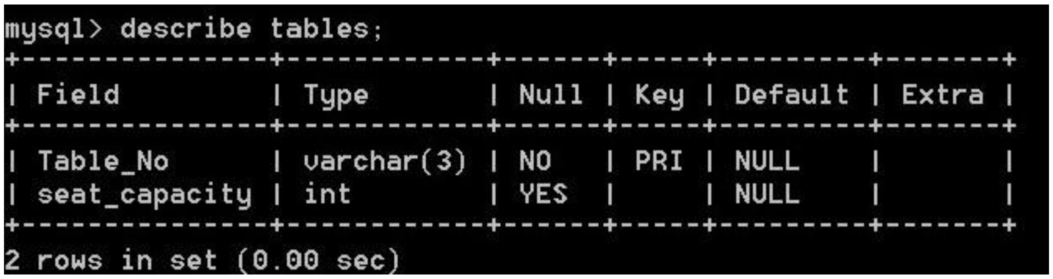
order_id is the candidate key order_id -----> table_no

order_id -----> date order_id -----> chef_id

order_id -----> waiter_id order_id -----> amount

The Table is in BCNF as it is Third Normal Form and the LHS of each functional dependency is a Candidate key.

tables table:-



```
mysql> describe tables;
+---------------+------------+------+-----+---------+-------+
| Field         | Type       | Null | Key | Default | Extra |
+---------------+------------+------+-----+---------+-------+
| Table_No      | varchar(3) | NO   | PRI | NULL    |       |
| seat_capacity | int        | YES  |     | NULL    |       |
+---------------+------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

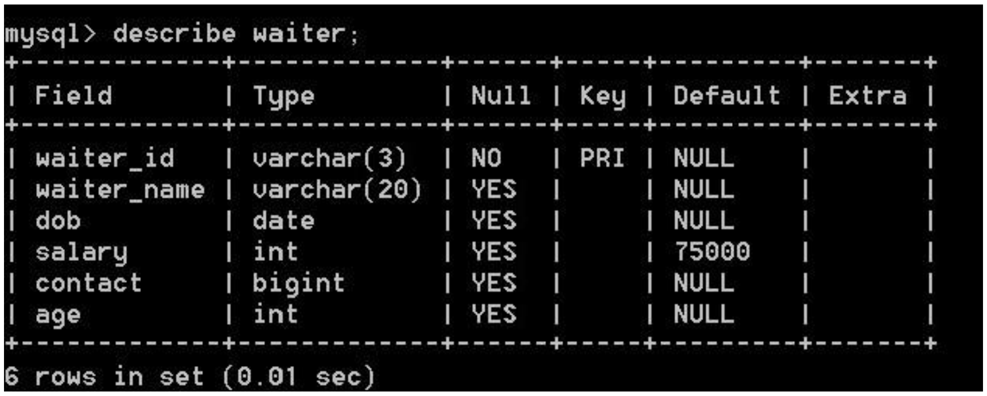
R(Table_No, seat_capacity)

Table_No is the candidate key

Table_No ----> seat_capacity

The Table is in BCNF as it is Third Normal Form and the LHS of each functional dependency is a Candidate key.

<u>waiter table</u>:-



```
mysql> describe waiter;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| waiter_id   | varchar(3)  | NO   | PRI | NULL    |       |
| waiter_name | varchar(20) | YES  |     | NULL    |       |
| dob         | date        | YES  |     | NULL    |       |
| salary      | int         | YES  |     | 75000   |       |
| contact     | bigint      | YES  |     | NULL    |       |
| age         | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

R(waiter_id, waiter_name, dob, salary, contact, age)

waiter_id is the candidate key

waiter_id -----> waiter_name

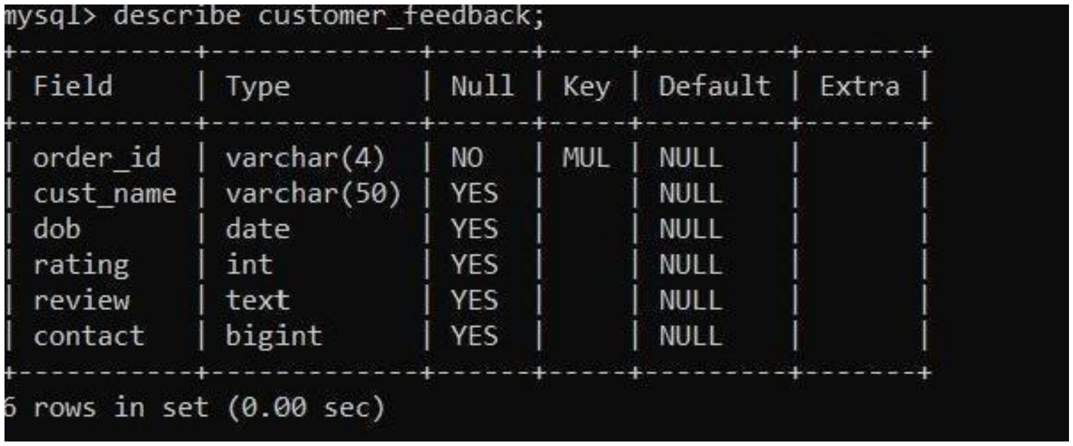
waiter_id -----> dob waiter_id

-----> salary waiter_id ----->

contact waiter_id -----> age

The Table is in BCNF as it is Third Normal Form and the LHS of each functional dependency is a Candidate key.

customer feedback table:-



```
mysql> describe customer_feedback;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| order_id  | varchar(4)  | NO   | MUL | NULL    |       |
| cust_name | varchar(50) | YES  |     | NULL    |       |
| dob       | date        | YES  |     | NULL    |       |
| rating    | int         | YES  |     | NULL    |       |
| review    | text        | YES  |     | NULL    |       |
| contact   | bigint      | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

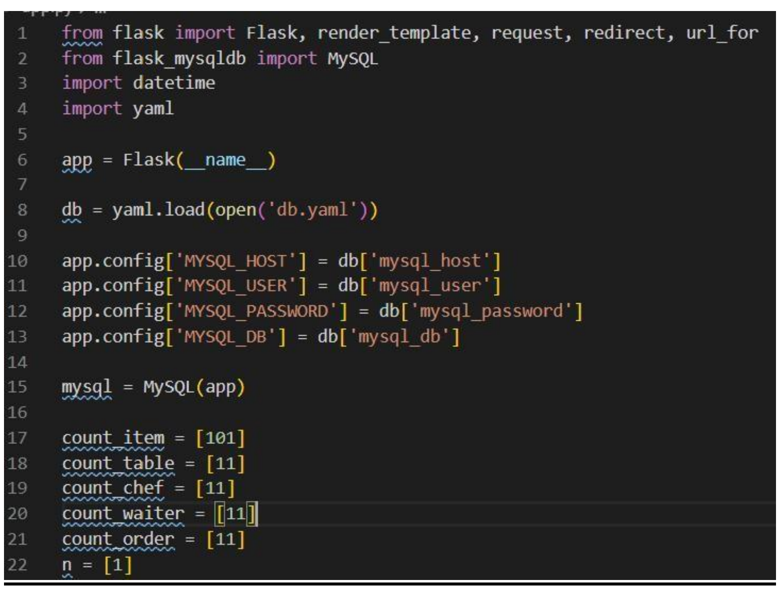
R(cust_name, contact_no, dob, rating, review, order_id)

order_id, contact_no is the candidate key order_id, contact_no-----> cust_name order_id, contact_no -----> dob order_id, contact_no -----> rating order_id, contact_no -----> review

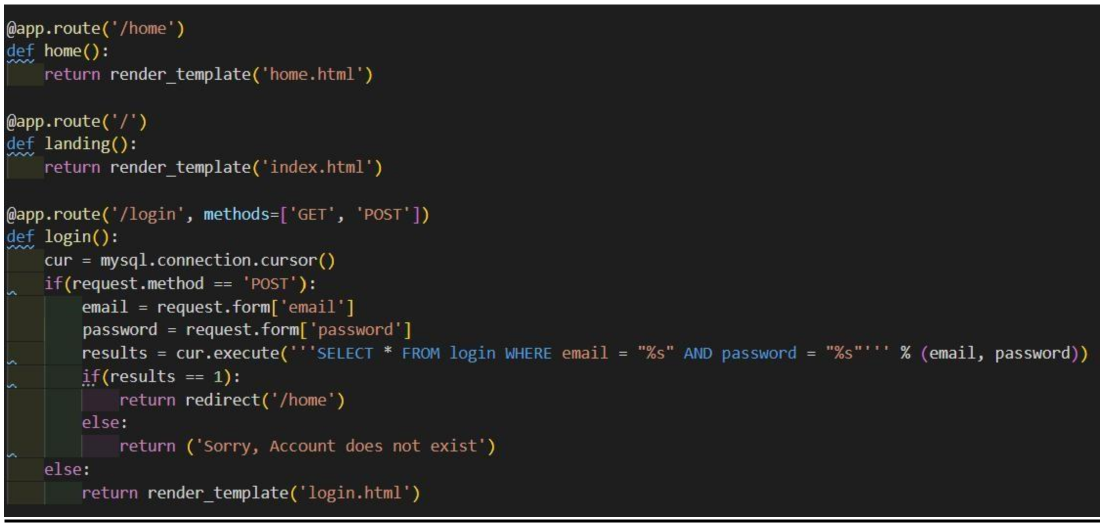
The Table is in BCNF as it is Third Normal Form and the LHS of each functional dependency is a Candidate key.

6.2. SNAPSOTS OF THE CODE

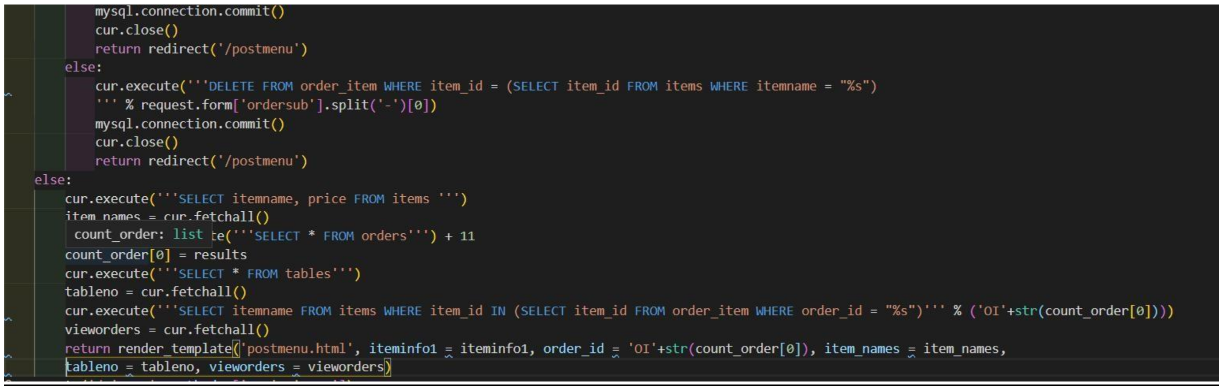
Connectivity Code



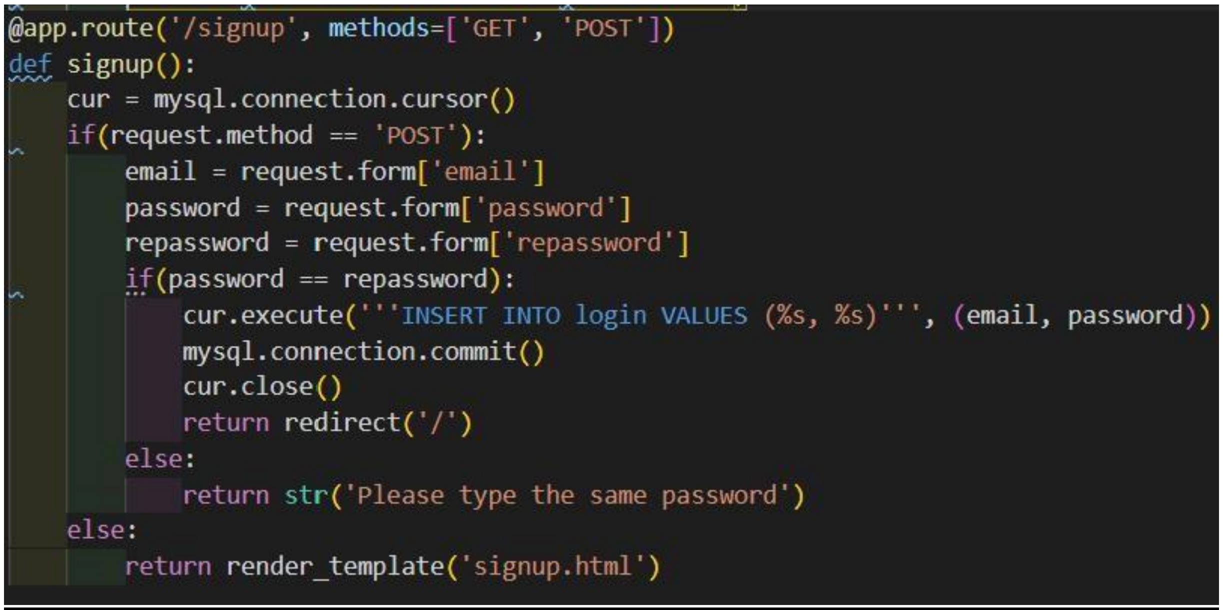
```python
from flask import Flask, render_template, request, redirect, url_for
from flask_mysqldb import MySQL
import datetime
import yaml

app = Flask(__name__)

db = yaml.load(open('db.yaml'))

app.config['MYSQL_HOST'] = db['mysql_host']
app.config['MYSQL_USER'] = db['mysql_user']
app.config['MYSQL_PASSWORD'] = db['mysql_password']
app.config['MYSQL_DB'] = db['mysql_db']

mysql = MySQL(app)

count_item = [101]
count_table = [11]
count_chef = [11]
count_waiter = [11]
count_order = [11]
n = [1]
```



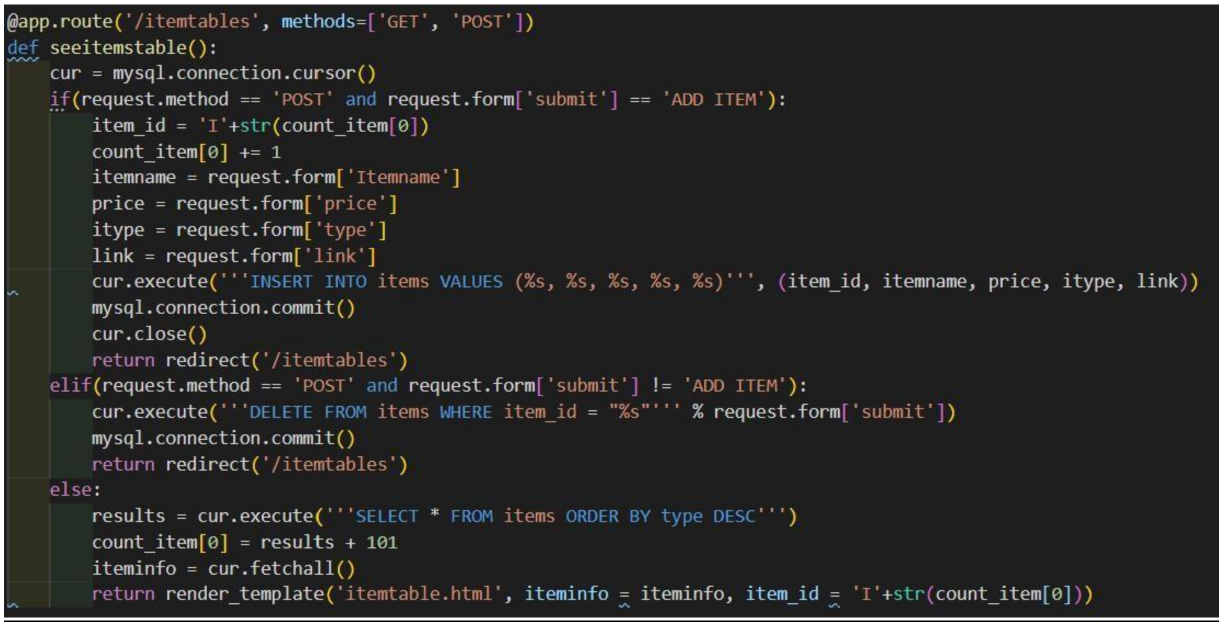
```python
@app.route('/home')
def home():
    return render_template('home.html')

@app.route('/')
def landing():
    return render_template('index.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    cur = mysql.connection.cursor()
    if(request.method == 'POST'):
        email = request.form['email']
        password = request.form['password']
        results = cur.execute('''SELECT * FROM login WHERE email = "%s" AND password = "%s"''' % (email, password))
        if(results == 1):
            return redirect('/home')
        else:
            return ('Sorry, Account does not exist')
    else:
        return render_template('login.html')
```



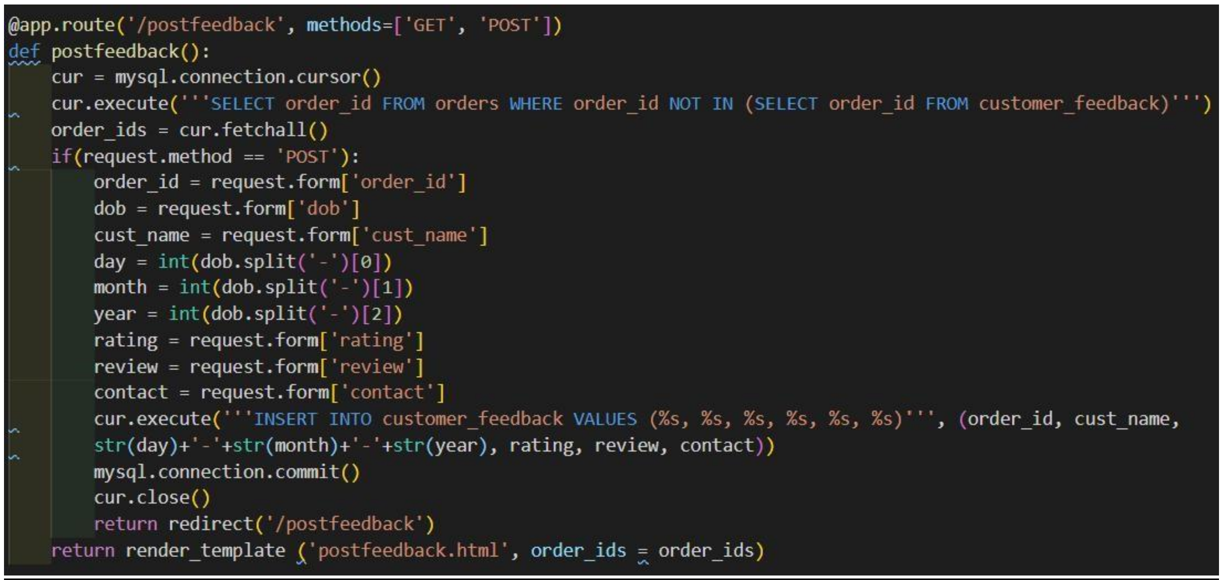
```
            mysql.connection.commit()
            cur.close()
            return redirect('/postmenu')
        else:
            cur.execute('''DELETE FROM order_item WHERE item_id = (SELECT item_id FROM items WHERE itemname = "%s")
            ''' % request.form['ordersub'].split('-')[0])
            mysql.connection.commit()
            cur.close()
            return redirect('/postmenu')
    else:
        cur.execute('''SELECT itemname, price FROM items ''')
        item_names = cur.fetchall()
         count_order: list te('''SELECT * FROM orders''') + 11
        count_order[0] = results
        cur.execute('''SELECT * FROM tables''')
        tableno = cur.fetchall()
        cur.execute('''SELECT itemname FROM items WHERE item_id IN (SELECT item_id FROM order_item WHERE order_id = "%s")''' % ('OI'+str(count_order[0])))
        vieworders = cur.fetchall()
        return render_template('postmenu.html', iteminfo1 = iteminfo1, order_id = 'OI'+str(count_order[0]), item_names = item_names,
        tableno = tableno, vieworders = vieworders)
```



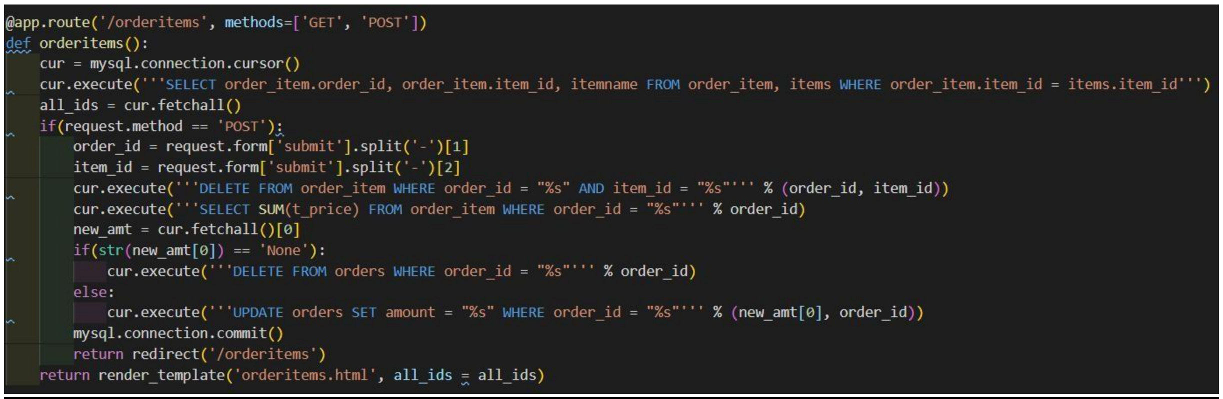
```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    cur = mysql.connection.cursor()
    if(request.method == 'POST'):
        email = request.form['email']
        password = request.form['password']
        repassword = request.form['repassword']
        if(password == repassword):
            cur.execute('''INSERT INTO login VALUES (%s, %s)''', (email, password))
            mysql.connection.commit()
            cur.close()
            return redirect('/')
        else:
            return str('Please type the same password')
    else:
        return render_template('signup.html')
```



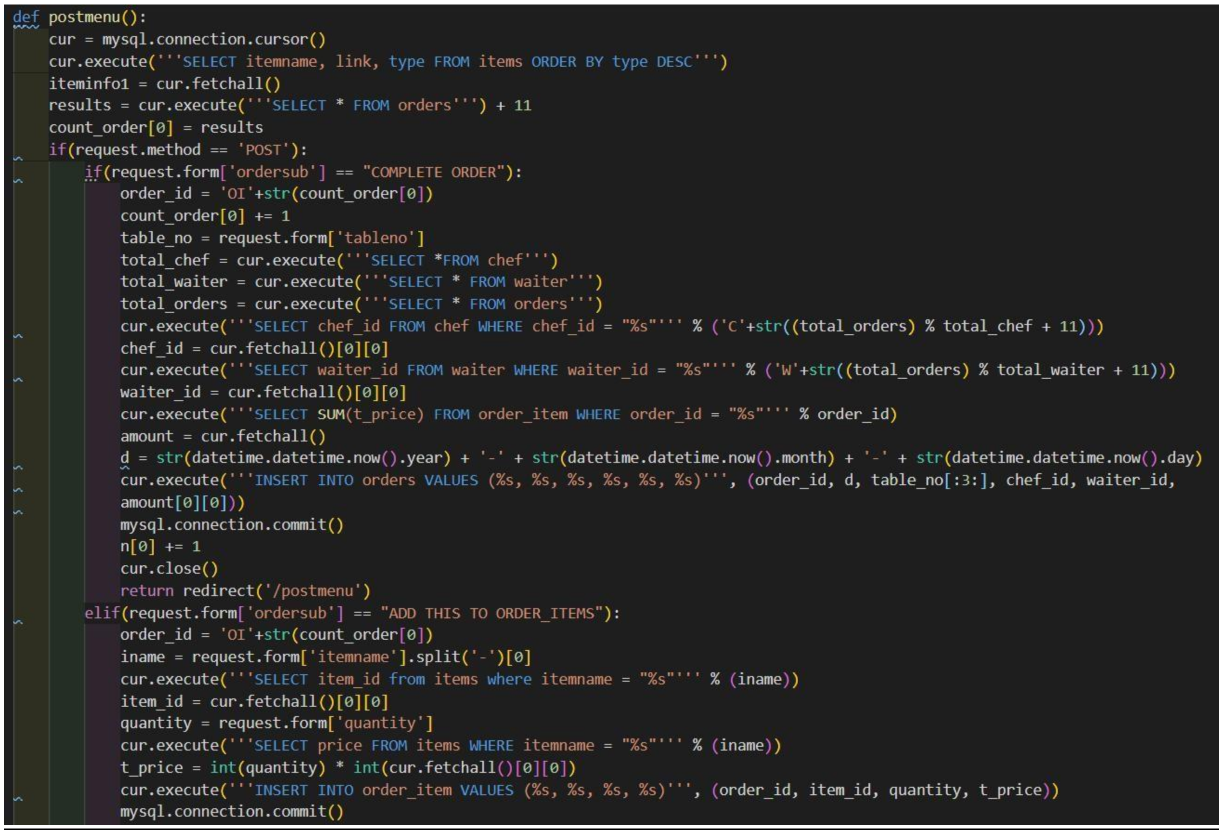
```
@app.route('/itemtables', methods=['GET', 'POST'])
def seeitemstable():
    cur = mysql.connection.cursor()
    if(request.method == 'POST' and request.form['submit'] == 'ADD ITEM'):
        item_id = 'I'+str(count_item[0])
        count_item[0] += 1
        itemname = request.form['Itemname']
        price = request.form['price']
        itype = request.form['type']
        link = request.form['link']
        cur.execute('''INSERT INTO items VALUES (%s, %s, %s, %s, %s)''', (item_id, itemname, price, itype, link))
        mysql.connection.commit()
        cur.close()
        return redirect('/itemtables')
    elif(request.method == 'POST' and request.form['submit'] != 'ADD ITEM'):
        cur.execute('''DELETE FROM items WHERE item_id = "%s"''' % request.form['submit'])
        mysql.connection.commit()
        return redirect('/itemtables')
    else:
        results = cur.execute('''SELECT * FROM items ORDER BY type DESC''')
        count_item[0] = results + 101
        iteminfo = cur.fetchall()
        return render_template('itemtable.html', iteminfo = iteminfo, item_id = 'I'+str(count_item[0]))
```



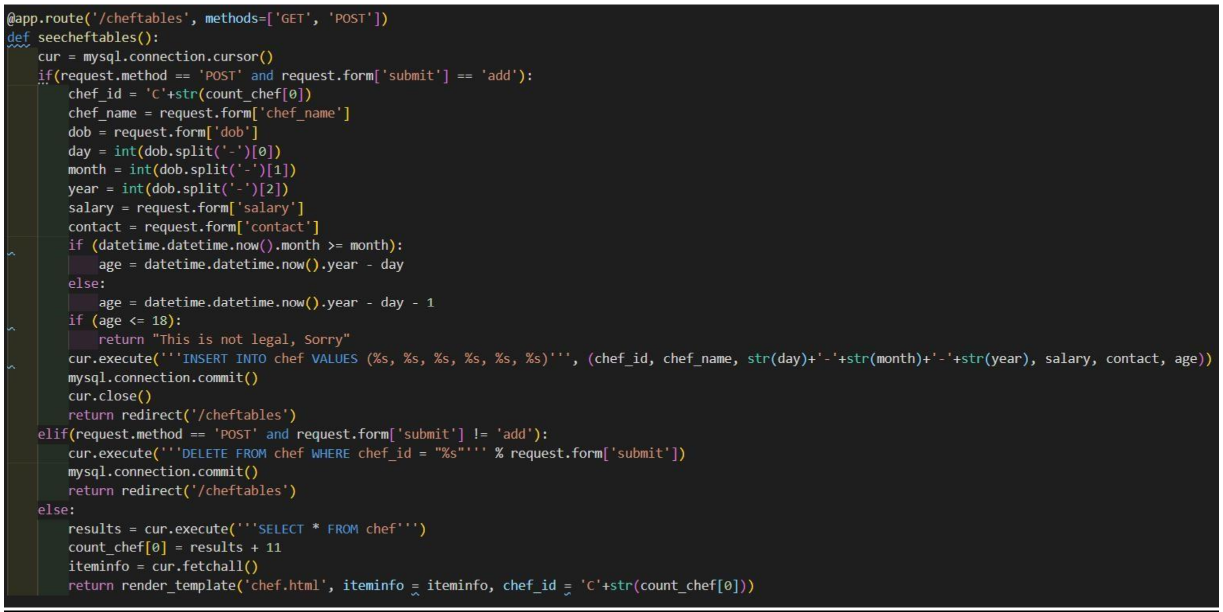
```
@app.route('/postfeedback', methods=['GET', 'POST'])
def postfeedback():
    cur = mysql.connection.cursor()
    cur.execute('''SELECT order_id FROM orders WHERE order_id NOT IN (SELECT order_id FROM customer_feedback)''')
    order_ids = cur.fetchall()
    if(request.method == 'POST'):
        order_id = request.form['order_id']
        dob = request.form['dob']
        cust_name = request.form['cust_name']
        day = int(dob.split('-')[0])
        month = int(dob.split('-')[1])
        year = int(dob.split('-')[2])
        rating = request.form['rating']
        review = request.form['review']
        contact = request.form['contact']
        cur.execute('''INSERT INTO customer_feedback VALUES (%s, %s, %s, %s, %s, %s)''', (order_id, cust_name,
        str(day)+'-'+str(month)+'-'+str(year), rating, review, contact))
        mysql.connection.commit()
        cur.close()
        return redirect('/postfeedback')
    return render_template ('postfeedback.html', order_ids = order_ids)
```



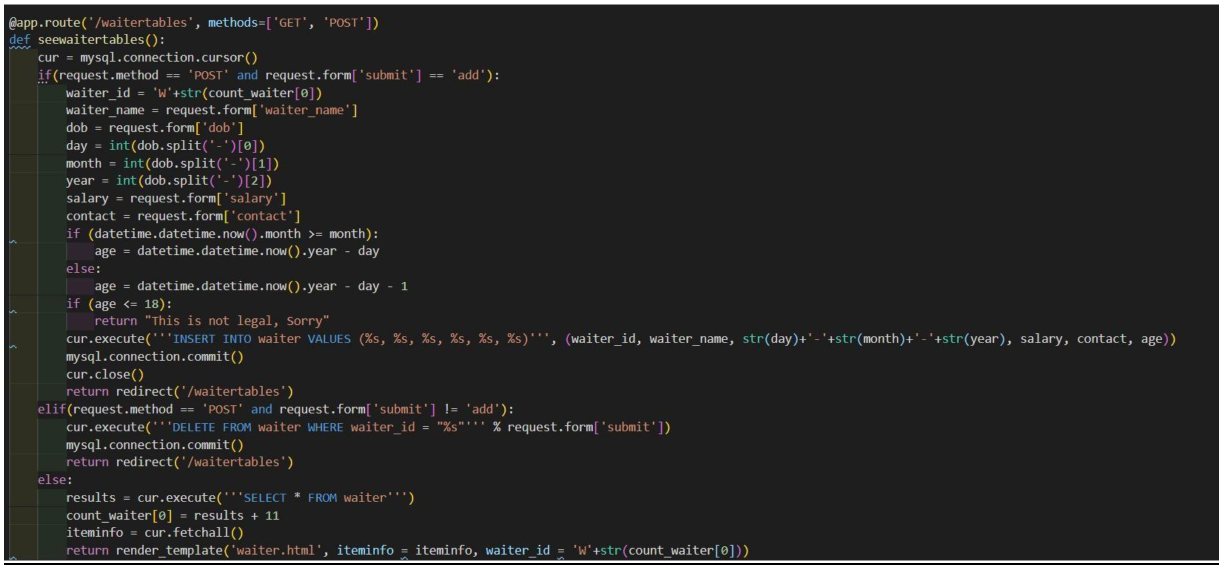
```python
@app.route('/orderitems', methods=['GET', 'POST'])
def orderitems():
    cur = mysql.connection.cursor()
    cur.execute('''SELECT order_item.order_id, order_item.item_id, itemname FROM order_item, items WHERE order_item.item_id = items.item_id''')
    all_ids = cur.fetchall()
    if(request.method == 'POST'):
        order_id = request.form['submit'].split('-')[1]
        item_id = request.form['submit'].split('-')[2]
        cur.execute('''DELETE FROM order_item WHERE order_id = "%s" AND item_id = "%s"''' % (order_id, item_id))
        cur.execute('''SELECT SUM(t_price) FROM order_item WHERE order_id = "%s"''' % order_id)
        new_amt = cur.fetchall()[0]
        if(str(new_amt[0]) == 'None'):
            cur.execute('''DELETE FROM orders WHERE order_id = "%s"''' % order_id)
        else:
            cur.execute('''UPDATE orders SET amount = "%s" WHERE order_id = "%s"''' % (new_amt[0], order_id))
        mysql.connection.commit()
        return redirect('/orderitems')
    return render_template('orderitems.html', all_ids = all_ids)
```



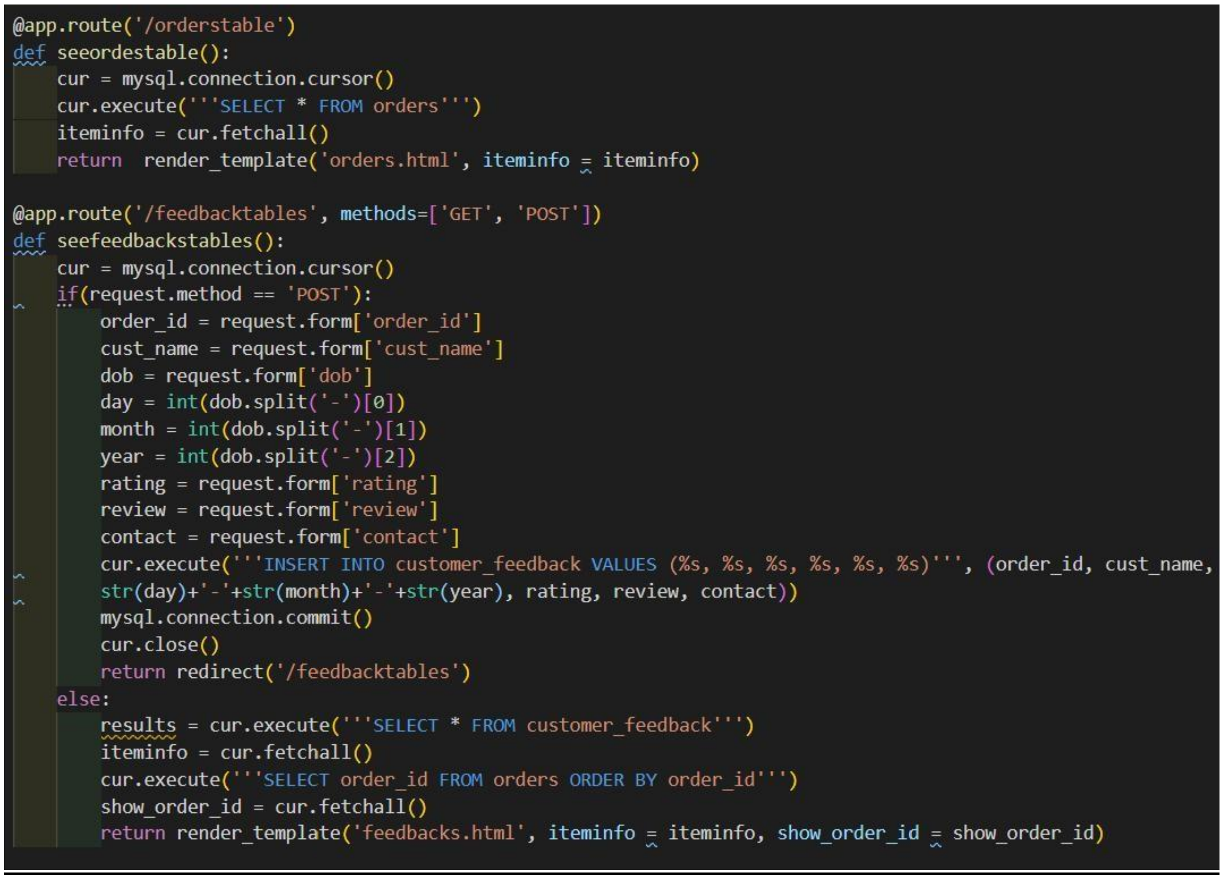
```python
def postmenu():
    cur = mysql.connection.cursor()
    cur.execute('''SELECT itemname, link, type FROM items ORDER BY type DESC''')
    iteminfo1 = cur.fetchall()
    results = cur.execute('''SELECT * FROM orders''') + 11
    count_order[0] = results
    if(request.method == 'POST'):
        if(request.form['ordersub'] == "COMPLETE ORDER"):
            order_id = 'OI'+str(count_order[0])
            count_order[0] += 1
            table_no = request.form['tableno']
            total_chef = cur.execute('''SELECT *FROM chef''')
            total_waiter = cur.execute('''SELECT * FROM waiter''')
            total_orders = cur.execute('''SELECT * FROM orders''')
            cur.execute('''SELECT chef_id FROM chef WHERE chef_id = "%s"''' % ('C'+str((total_orders) % total_chef + 11)))
            chef_id = cur.fetchall()[0][0]
            cur.execute('''SELECT waiter_id FROM waiter WHERE waiter_id = "%s"''' % ('W'+str((total_orders) % total_waiter + 11)))
            waiter_id = cur.fetchall()[0][0]
            cur.execute('''SELECT SUM(t_price) FROM order_item WHERE order_id = "%s"''' % order_id)
            amount = cur.fetchall()
            d = str(datetime.datetime.now().year) + '-' + str(datetime.datetime.now().month) + '-' + str(datetime.datetime.now().day)
            cur.execute('''INSERT INTO orders VALUES (%s, %s, %s, %s, %s, %s)''', (order_id, d, table_no[:3:], chef_id, waiter_id,
            amount[0][0]))
            mysql.connection.commit()
            n[0] += 1
            cur.close()
            return redirect('/postmenu')
        elif(request.form['ordersub'] == "ADD THIS TO ORDER_ITEMS"):
            order_id = 'OI'+str(count_order[0])
            iname = request.form['itemname'].split('-')[0]
            cur.execute('''SELECT item_id from items where itemname = "%s"''' % (iname))
            item_id = cur.fetchall()[0][0]
            quantity = request.form['quantity']
            cur.execute('''SELECT price FROM items WHERE itemname = "%s"''' % (iname))
            t_price = int(quantity) * int(cur.fetchall()[0][0])
            cur.execute('''INSERT INTO order_item VALUES (%s, %s, %s, %s)''', (order_id, item_id, quantity, t_price))
            mysql.connection.commit()
```



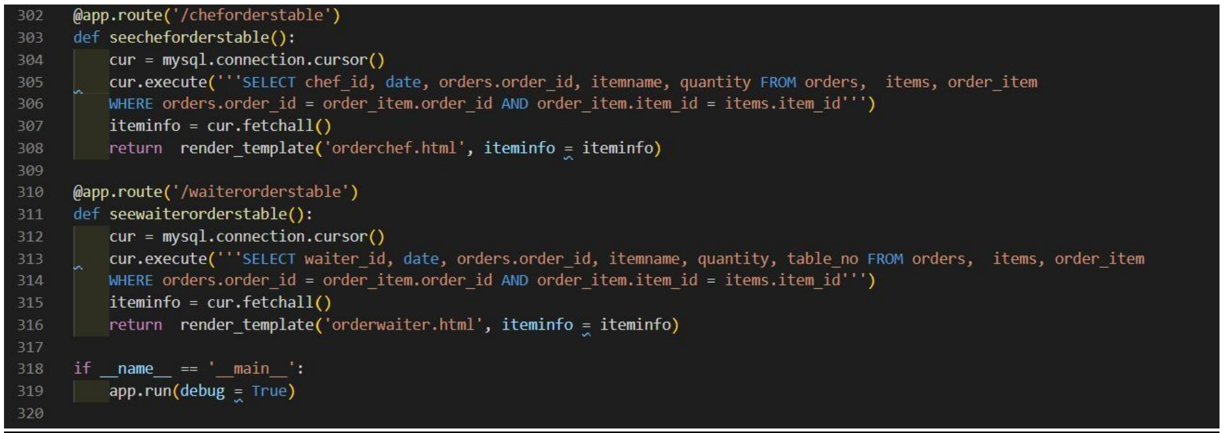
```python
@app.route('/cheftables', methods=['GET', 'POST'])
def seecheftables():
    cur = mysql.connection.cursor()
    if(request.method == 'POST' and request.form['submit'] == 'add'):
        chef_id = 'C'+str(count_chef[0])
        chef_name = request.form['chef_name']
        dob = request.form['dob']
        day = int(dob.split('-')[0])
        month = int(dob.split('-')[1])
        year = int(dob.split('-')[2])
        salary = request.form['salary']
        contact = request.form['contact']
        if (datetime.datetime.now().month >= month):
            age = datetime.datetime.now().year - day
        else:
            age = datetime.datetime.now().year - day - 1
        if (age <= 18):
            return "This is not legal, Sorry"
        cur.execute('''INSERT INTO chef VALUES (%s, %s, %s, %s, %s, %s)''', (chef_id, chef_name, str(day)+'-'+str(month)+'-'+str(year), salary, contact, age))
        mysql.connection.commit()
        cur.close()
        return redirect('/cheftables')
    elif(request.method == 'POST' and request.form['submit'] != 'add'):
        cur.execute('''DELETE FROM chef WHERE chef_id = "%s"''' % request.form['submit'])
        mysql.connection.commit()
        return redirect('/cheftables')
    else:
        results = cur.execute('''SELECT * FROM chef''')
        count_chef[0] = results + 11
        iteminfo = cur.fetchall()
        return render_template('chef.html', iteminfo = iteminfo, chef_id = 'C'+str(count_chef[0]))
```



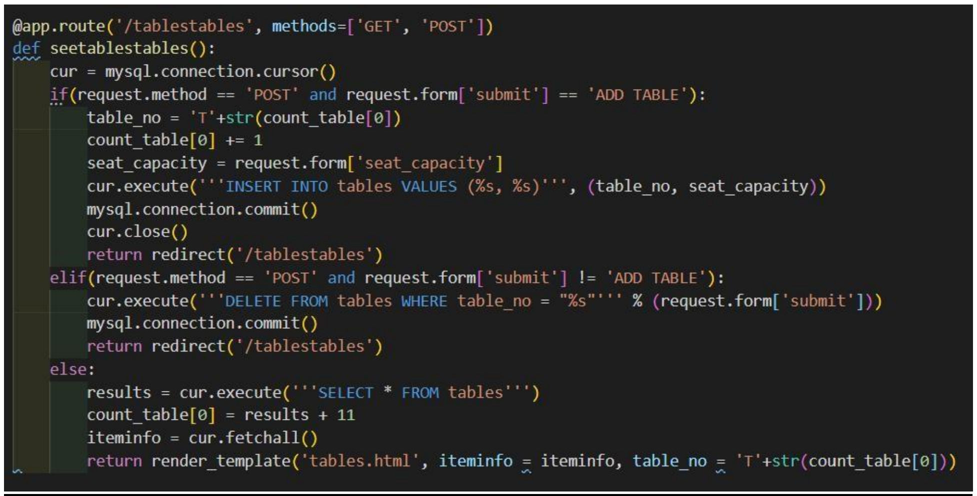
```python
@app.route('/waitertables', methods=['GET', 'POST'])
def seewaitertables():
    cur = mysql.connection.cursor()
    if(request.method == 'POST' and request.form['submit'] == 'add'):
        waiter_id = 'W'+str(count_waiter[0])
        waiter_name = request.form['waiter_name']
        dob = request.form['dob']
        day = int(dob.split('-')[0])
        month = int(dob.split('-')[1])
        year = int(dob.split('-')[2])
        salary = request.form['salary']
        contact = request.form['contact']
        if (datetime.datetime.now().month >= month):
            age = datetime.datetime.now().year - day
        else:
            age = datetime.datetime.now().year - day - 1
        if (age <= 18):
            return "This is not legal, Sorry"
        cur.execute('''INSERT INTO waiter VALUES (%s, %s, %s, %s, %s, %s)''', (waiter_id, waiter_name, str(day)+'-'+str(month)+'-'+str(year), salary, contact, age))
        mysql.connection.commit()
        cur.close()
        return redirect('/waitertables')
    elif(request.method == 'POST' and request.form['submit'] != 'add'):
        cur.execute('''DELETE FROM waiter WHERE waiter_id = "%s"''' % request.form['submit'])
        mysql.connection.commit()
        return redirect('/waitertables')
    else:
        results = cur.execute('''SELECT * FROM waiter''')
        count_waiter[0] = results + 11
        iteminfo = cur.fetchall()
        return render_template('waiter.html', iteminfo = iteminfo, waiter_id = 'W'+str(count_waiter[0]))
```



```python
@app.route('/orderstable')
def seeordestable():
    cur = mysql.connection.cursor()
    cur.execute('''SELECT * FROM orders''')
    iteminfo = cur.fetchall()
    return  render_template('orders.html', iteminfo = iteminfo)

@app.route('/feedbacktables', methods=['GET', 'POST'])
def seefeedbackstables():
    cur = mysql.connection.cursor()
    if(request.method == 'POST'):
        order_id = request.form['order_id']
        cust_name = request.form['cust_name']
        dob = request.form['dob']
        day = int(dob.split('-')[0])
        month = int(dob.split('-')[1])
        year = int(dob.split('-')[2])
        rating = request.form['rating']
        review = request.form['review']
        contact = request.form['contact']
        cur.execute('''INSERT INTO customer_feedback VALUES (%s, %s, %s, %s, %s, %s)''', (order_id, cust_name,
        str(day)+'-'+str(month)+'-'+str(year), rating, review, contact))
        mysql.connection.commit()
        cur.close()
        return redirect('/feedbacktables')
    else:
        results = cur.execute('''SELECT * FROM customer_feedback''')
        iteminfo = cur.fetchall()
        cur.execute('''SELECT order_id FROM orders ORDER BY order_id''')
        show_order_id = cur.fetchall()
        return render_template('feedbacks.html', iteminfo = iteminfo, show_order_id = show_order_id)
```

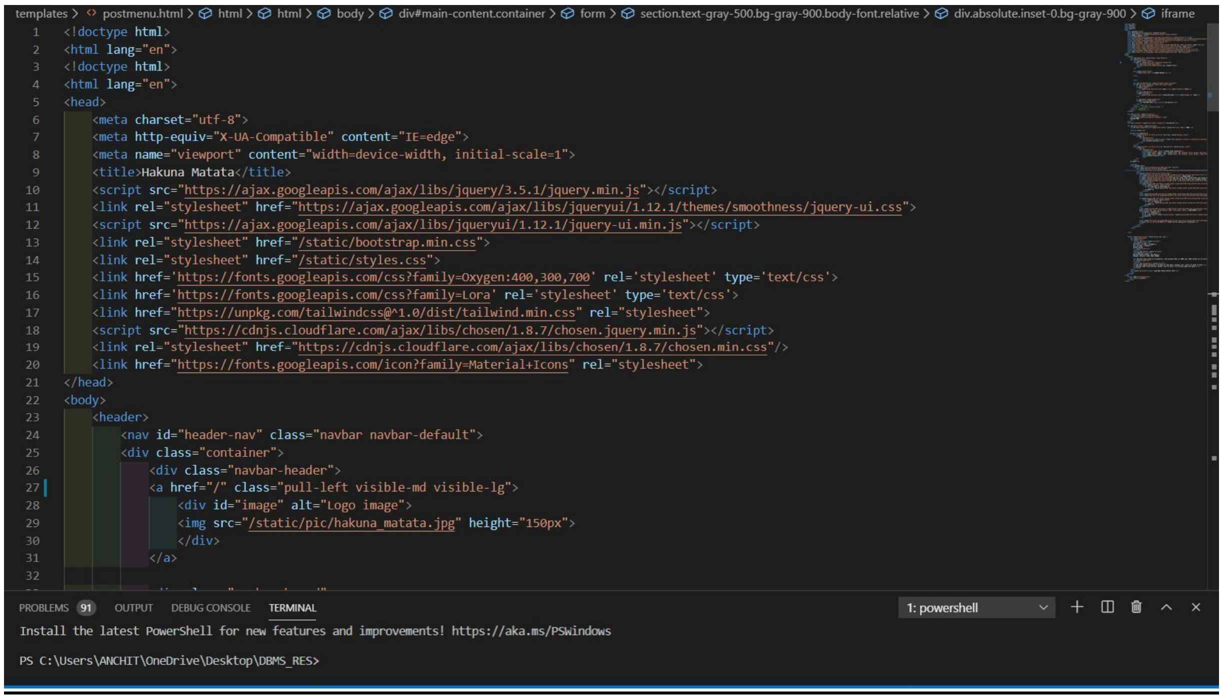


```python
@app.route('/cheforderstable')
def seecheforderstable():
    cur = mysql.connection.cursor()
    cur.execute('''SELECT chef_id, date, orders.order_id, itemname, quantity FROM orders,  items, order_item
    WHERE orders.order_id = order_item.order_id AND order_item.item_id = items.item_id''')
    iteminfo = cur.fetchall()
    return  render_template('orderchef.html', iteminfo = iteminfo)

@app.route('/waiterorderstable')
def seewaiterorderstable():
    cur = mysql.connection.cursor()
    cur.execute('''SELECT waiter_id, date, orders.order_id, itemname, quantity, table_no FROM orders,  items, order_item
    WHERE orders.order_id = order_item.order_id AND order_item.item_id = items.item_id''')
    iteminfo = cur.fetchall()
    return  render_template('orderwaiter.html', iteminfo = iteminfo)

if __name__ == '__main__':
    app.run(debug = True)
```



```python
@app.route('/tablestables', methods=['GET', 'POST'])
def seetablestables():
    cur = mysql.connection.cursor()
    if(request.method == 'POST' and request.form['submit'] == 'ADD TABLE'):
        table_no = 'T'+str(count_table[0])
        count_table[0] += 1
        seat_capacity = request.form['seat_capacity']
        cur.execute('''INSERT INTO tables VALUES (%s, %s)''', (table_no, seat_capacity))
        mysql.connection.commit()
        cur.close()
        return redirect('/tablestables')
    elif(request.method == 'POST' and request.form['submit'] != 'ADD TABLE'):
        cur.execute('''DELETE FROM tables WHERE table_no = "%s"''' % (request.form['submit']))
        mysql.connection.commit()
        return redirect('/tablestables')
    else:
        results = cur.execute('''SELECT * FROM tables''')
        count_table[0] = results + 11
        iteminfo = cur.fetchall()
        return render_template('tables.html', iteminfo = iteminfo, table_no = 'T'+str(count_table[0]))
```

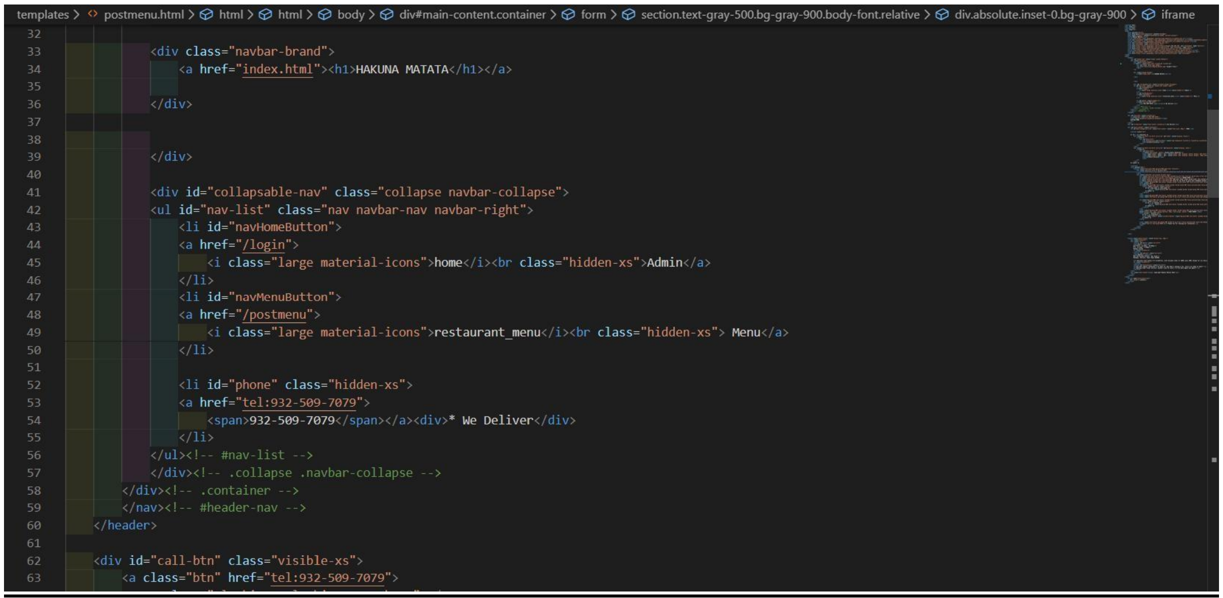
FEW SCREENSHOTS OF THE FRONTEND CODE



templates > postmenu.html > html > html > body > div#main-content.container > form > section.text-gray-500.bg-gray-900.body-font.relative > div.absolute.inset-0.bg-gray-900 > iframe

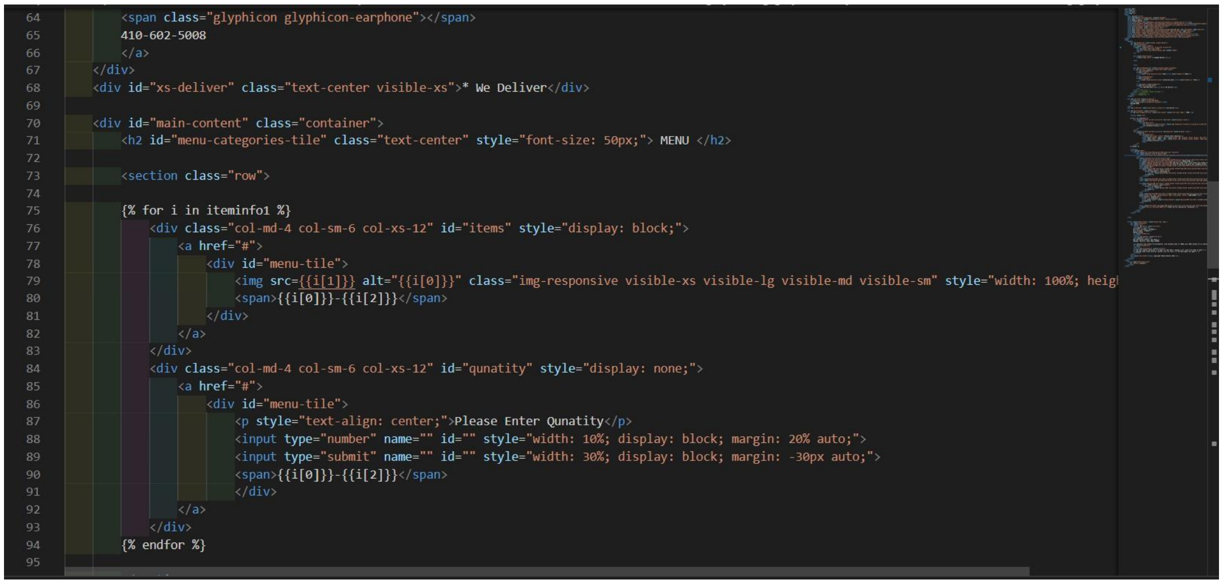
```html
<!doctype html>
<html lang="en">
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Hakuna Matata</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-ui.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"></script>
    <link rel="stylesheet" href="/static/bootstrap.min.css">
    <link rel="stylesheet" href="/static/styles.css">
    <link href='https://fonts.googleapis.com/css?family=Oxygen:400,300,700' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Lora' rel='stylesheet' type='text/css'>
    <link href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/chosen/1.8.7/chosen.jquery.min.js"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/chosen/1.8.7/chosen.min.css"/>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>
<body>
    <header>
        <nav id="header-nav" class="navbar navbar-default">
        <div class="container">
            <div class="navbar-header">
            <a href="/" class="pull-left visible-md visible-lg">
                <div id="image" alt="Logo image">
                <img src="/static/pic/hakuna_matata.jpg" height="150px">
                </div>
            </a>

```

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ANCHIT\OneDrive\Desktop\DBMS_RES>
```



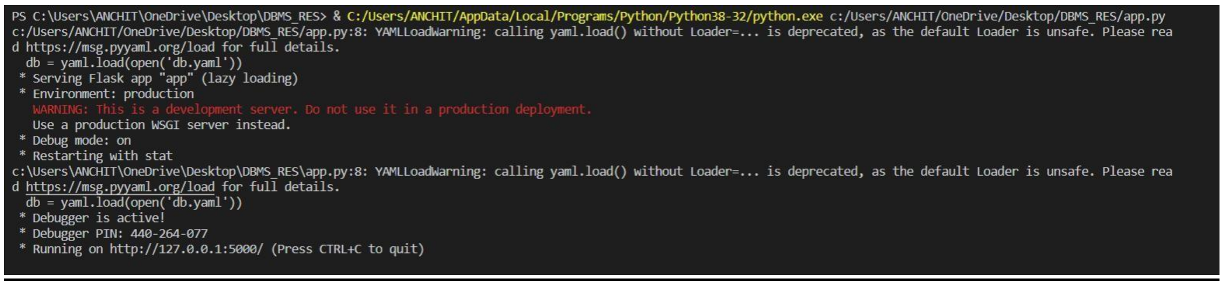
templates > postmenu.html > html > html > body > div#main-content.container > form > section.text-gray-500.bg-gray-900.body-font.relative > div.absolute.inset-0.bg-gray-900 > iframe

```

            <div class="navbar-brand">
                <a href="index.html"><h1>HAKUNA MATATA</h1></a>

            </div>


            </div>

            <div id="collapsable-nav" class="collapse navbar-collapse">
            <ul id="nav-list" class="nav navbar-nav navbar-right">
                <li id="navHomeButton">
                <a href="/login">
                    <i class="large material-icons">home</i><br class="hidden-xs">Admin</a>
                </li>
                <li id="navMenuButton">
                <a href="/postmenu">
                    <i class="large material-icons">restaurant_menu</i><br class="hidden-xs"> Menu</a>
                </li>

                <li id="phone" class="hidden-xs">
                <a href="tel:932-509-7079">
                    <span>932-509-7079</span></a><div>* We Deliver</div>
                </li>
            </ul><!-- #nav-list -->
            </div><!-- .collapse .navbar-collapse -->
        </div><!-- .container -->
        </nav><!-- #header-nav -->
    </header>

    <div id="call-btn" class="visible-xs">
        <a class="btn" href="tel:932-509-7079">
```



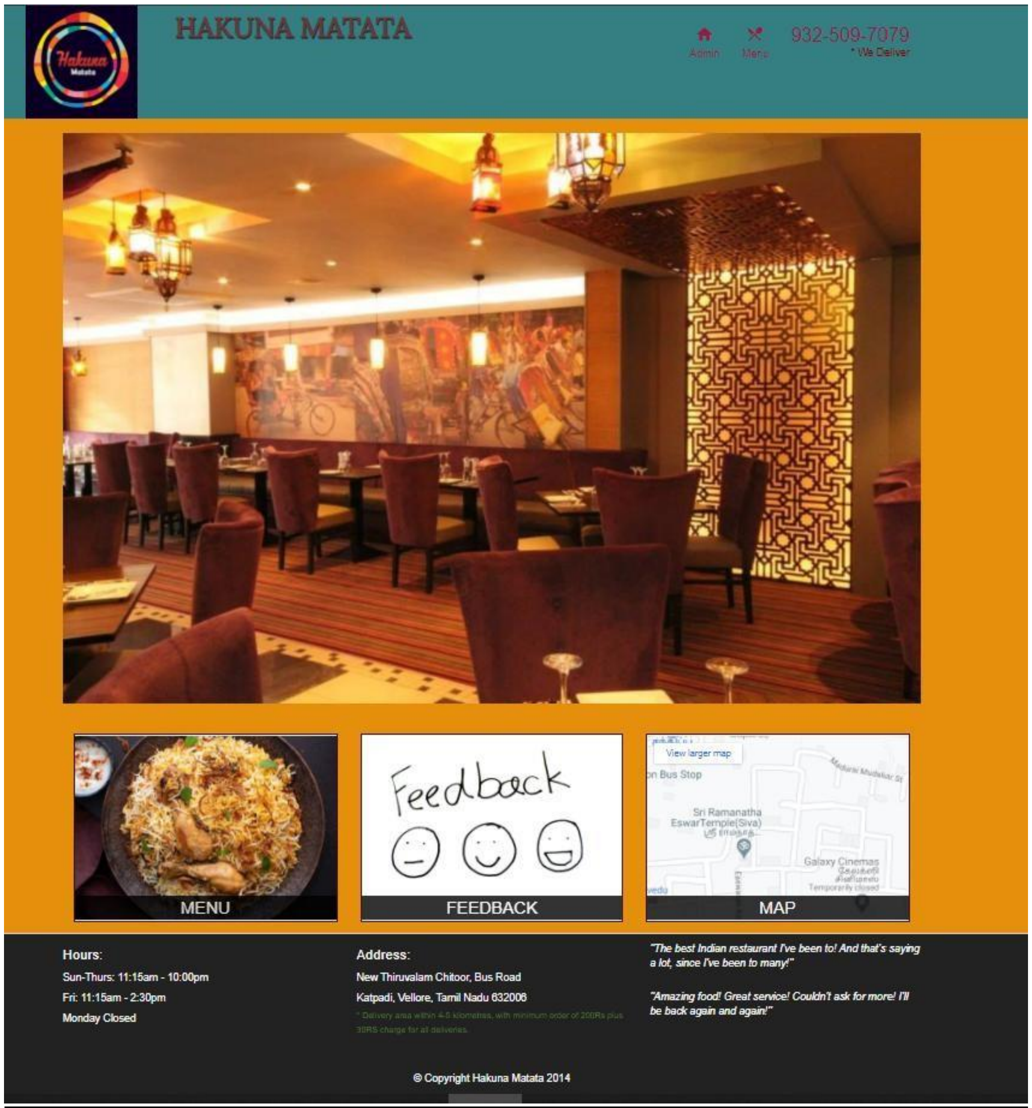
```
        <span class="glyphicon glyphicon-earphone"></span>
        410-602-5008
        </a>
    </div>
    <div id="xs-deliver" class="text-center visible-xs">* We Deliver</div>

    <div id="main-content" class="container">
        <h2 id="menu-categories-tile" class="text-center" style="font-size: 50px;"> MENU </h2>

        <section class="row">

        {% for i in iteminfo1 %}
            <div class="col-md-4 col-sm-6 col-xs-12" id="items" style="display: block;">
                <a href="#">
                    <div id="menu-tile">
                        <img src={{i[1]}} alt="{{i[0]}}" class="img-responsive visible-xs visible-lg visible-md visible-sm" style="width: 100%; heig
                        <span>{{i[0]}}-{{i[2]}}</span>
                    </div>
                </a>
            </div>
            <div class="col-md-4 col-sm-6 col-xs-12" id="qunatity" style="display: none;">
                <a href="#">
                    <div id="menu-tile">
                        <p style="text-align: center;">Please Enter Qunatity</p>
                        <input type="number" name="" id="" style="width: 10%; display: block; margin: 20% auto;">
                        <input type="submit" name="" id="" style="width: 30%; display: block; margin: -30px auto;">
                        <span>{{i[0]}}-{{i[2]}}</span>
                        </div>
                </a>
            </div>
        {% endfor %}

```

UPON STARTING THE SERVER I.E. ON RUNNING THE FLASK CODE



```
PS C:\Users\ANCHIT\OneDrive\Desktop\DBMS_RES> & C:/Users/ANCHIT/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/ANCHIT/OneDrive/Desktop/DBMS_RES/app.py
c:/Users/ANCHIT/OneDrive/Desktop/DBMS_RES/app.py:8: YAMLLoadWarning: calling yaml.load() without Loader=... is deprecated, as the default Loader is unsafe. Please rea
d https://msg.pyyaml.org/load for full details.
  db = yaml.load(open('db.yaml'))
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
c:\Users\ANCHIT\OneDrive\Desktop\DBMS_RES\app.py:8: YAMLLoadWarning: calling yaml.load() without Loader=... is deprecated, as the default Loader is unsafe. Please rea
d https://msg.pyyaml.org/load for full details.
  db = yaml.load(open('db.yaml'))
 * Debugger is active!
 * Debugger PIN: 440-264-077
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Website Screenshots



Hours:

Sun-Thurs: 11:15am - 10:00pm

Fri: 11:15am - 2:30pm

Monday Closed

Address:

New Thiruvalam Chitoor, Bus Road

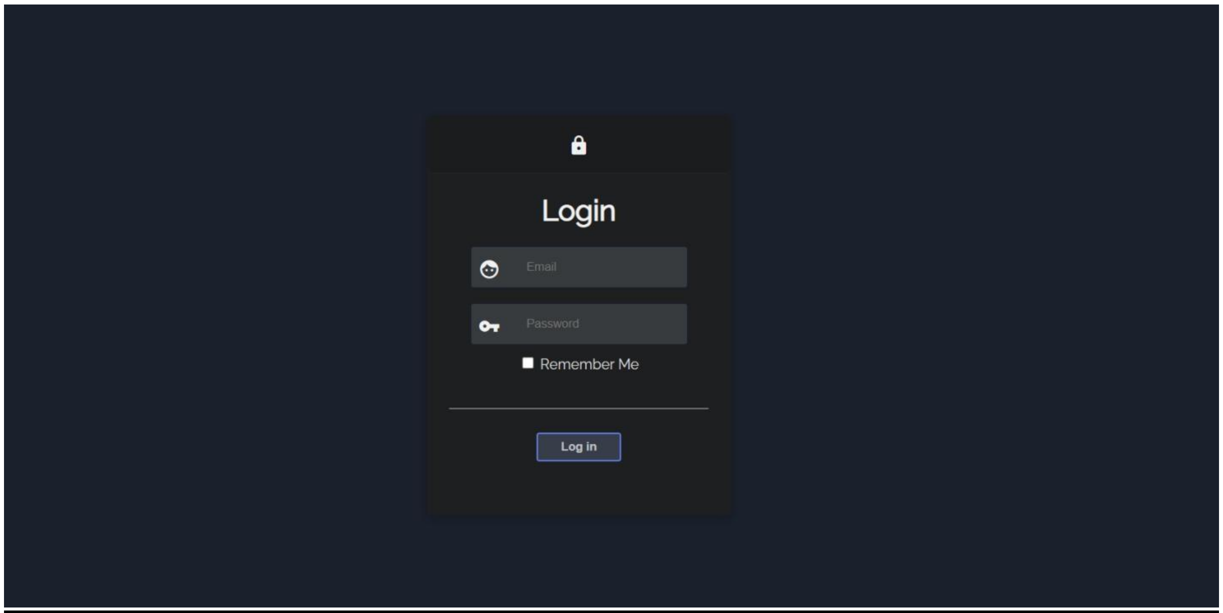
Katpadi, Vellore, Tamil Nadu 632006

* Delivery area within 4-5 kilometres, with minimum order of 200Rs plus 30RS charge for all deliveries.

"The best Indian restaurant I've been to! And that's saying a lot, since I've been to many!"

"Amazing food! Great service! Couldn't ask for more! I'll be back again and again!"

© Copyright Hakuna Matata 2014



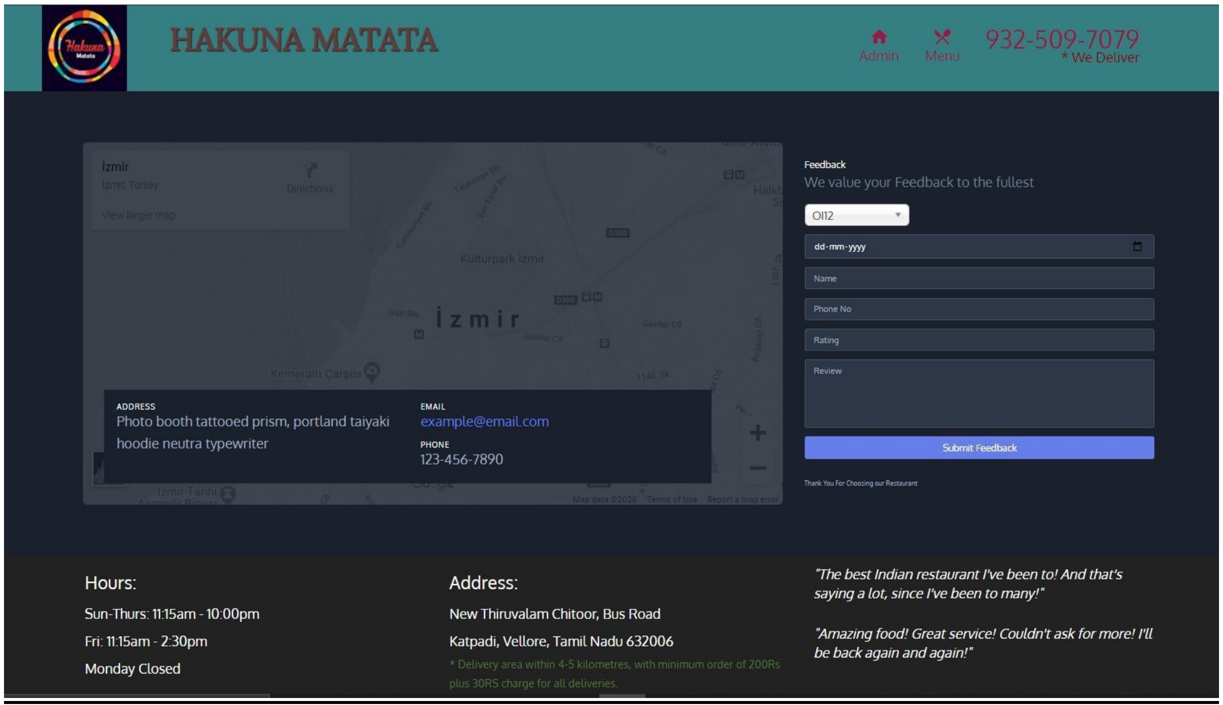
Login

Email

Password

Remember Me

Log in



HAKUNA MATATA

Admin Menu 932-509-7079
* We Deliver

ADDRESS
Photo booth tattooed prism, portland taiyaki hoodie neutra typewriter

EMAIL
example@email.com

PHONE
123-456-7890

Feedback
We value your Feedback to the fullest

O112

dd-mm-yyyy

Name

Phone No

Rating

Review

Submit Feedback

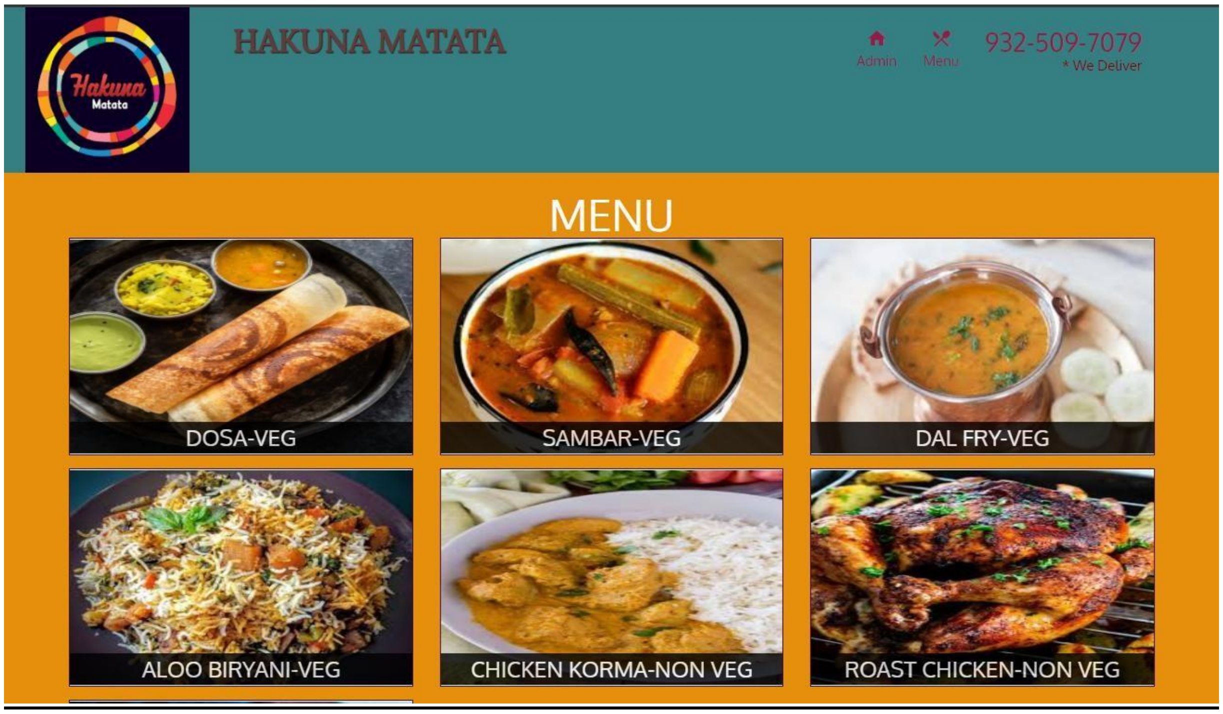
Thank You For Choosing our Restaurant

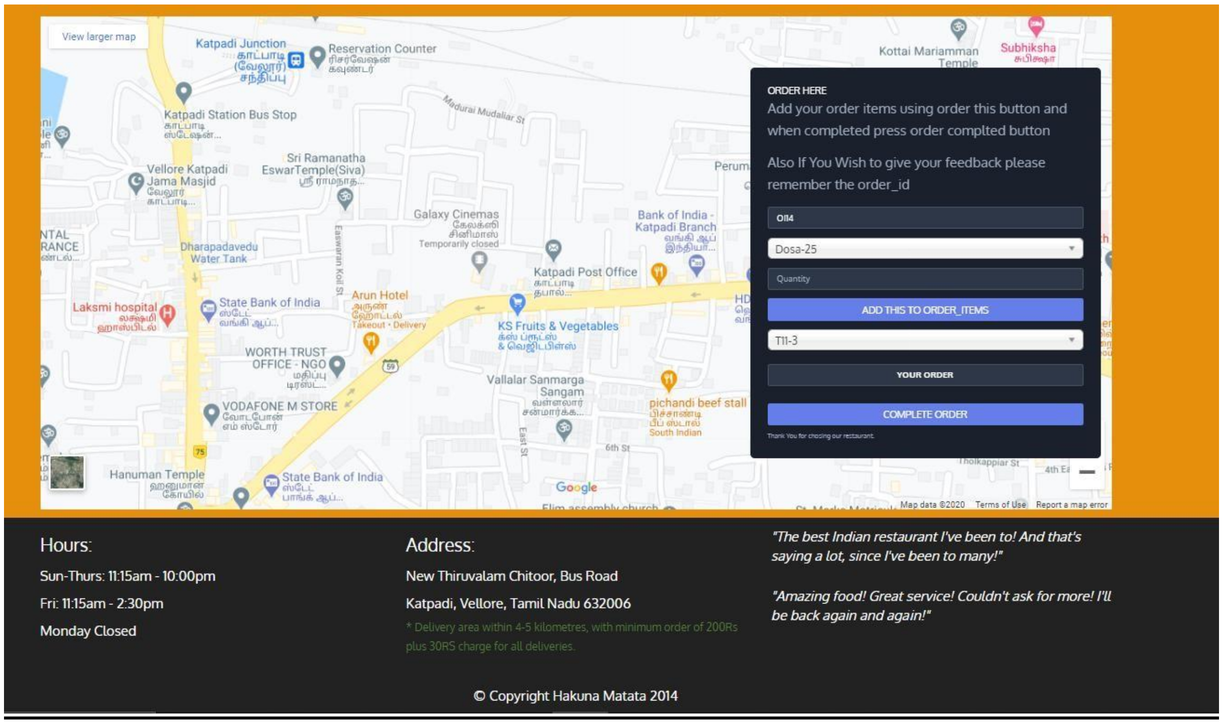
Hours:
Sun-Thurs: 11:15am - 10:00pm
Fri: 11:15am - 2:30pm
Monday Closed

Address:
New Thiruvalam Chitoor, Bus Road
Katpadi, Vellore, Tamil Nadu 632006
* Delivery area within 4-5 kilometres, with minimum order of 200Rs plus 30RS charge for all deliveries.

"The best Indian restaurant I've been to! And that's saying a lot, since I've been to many!"

"Amazing food! Great service! Couldn't ask for more! I'll be back again and again!"





Add your order items using order this button and when completed press order complted button

Also If You Wish to give your feedback please remember the order_id

OI14

Dosa-25

Quantity

T11-3

Thank You for chosing our restaurant.

Hours:

Sun-Thurs: 11:15am - 10:00pm

Fri: 11:15am - 2:30pm

Monday Closed

Address:

New Thiruvalam Chitoor, Bus Road

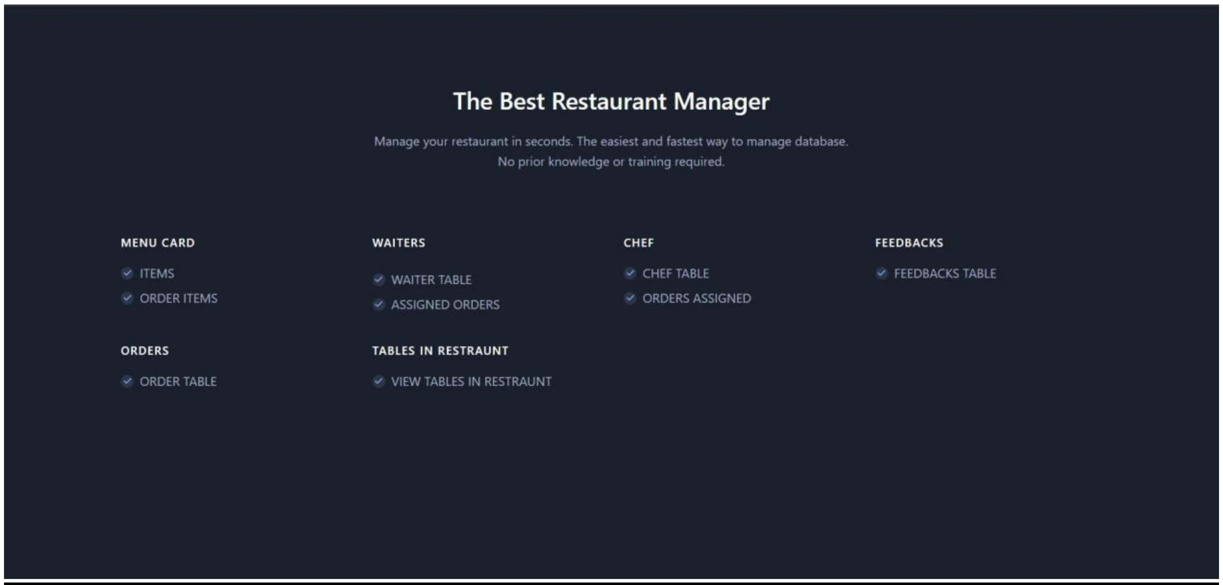
Katpadi, Vellore, Tamil Nadu 632006

* Delivery area within 4-5 kilometres, with minimum order of 200Rs plus 30RS charge for all deliveries.

"The best Indian restaurant I've been to! And that's saying a lot, since I've been to many!"

"Amazing food! Great service! Couldn't ask for more! I'll be back again and again!"

© Copyright Hakuna Matata 2014



The Best Restaurant Manager

Manage your restaurant in seconds. The easiest and fastest way to manage database.
No prior knowledge or training required.

MENU CARD
- ITEMS
- ORDER ITEMS

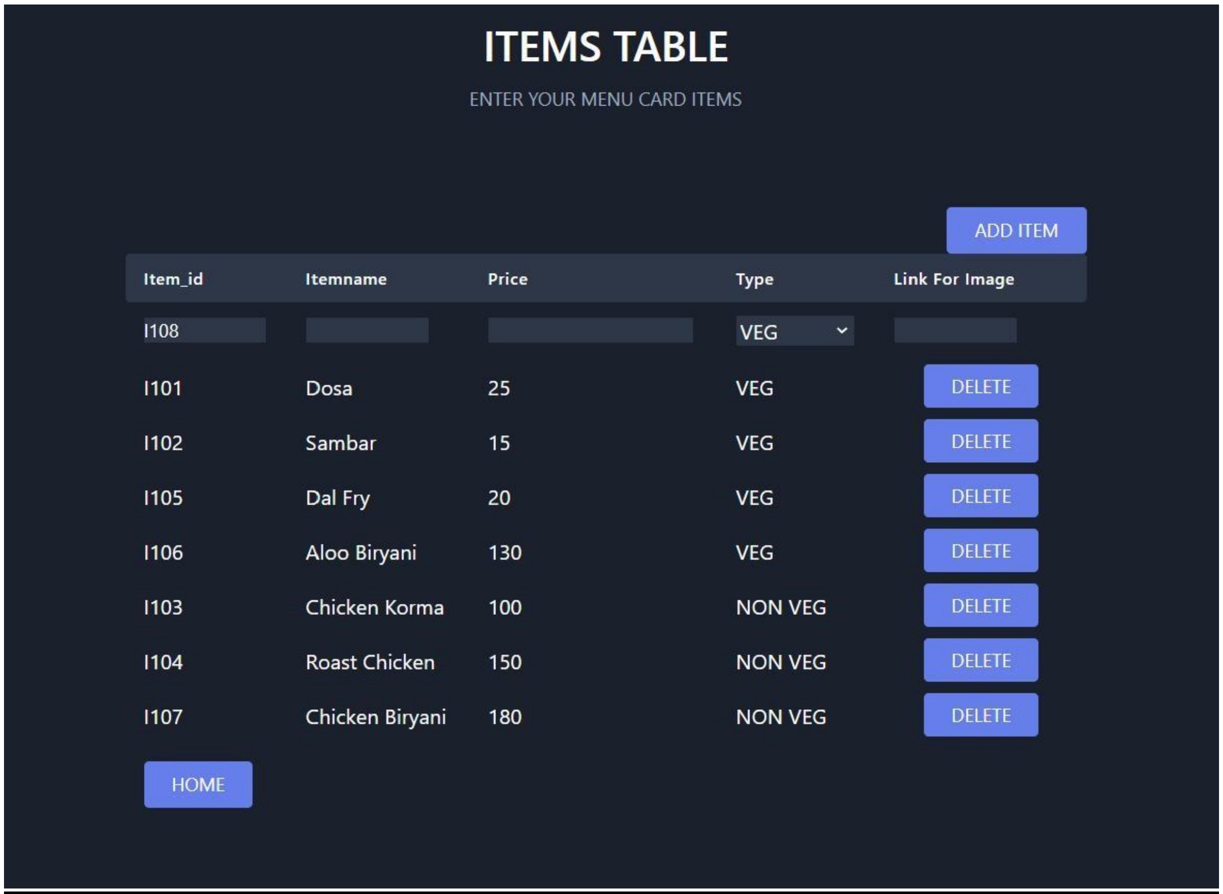
WAITERS
- WAITER TABLE
- ASSIGNED ORDERS

CHEF
- CHEF TABLE
- ORDERS ASSIGNED

FEEDBACKS
- FEEDBACKS TABLE

ORDERS
- ORDER TABLE

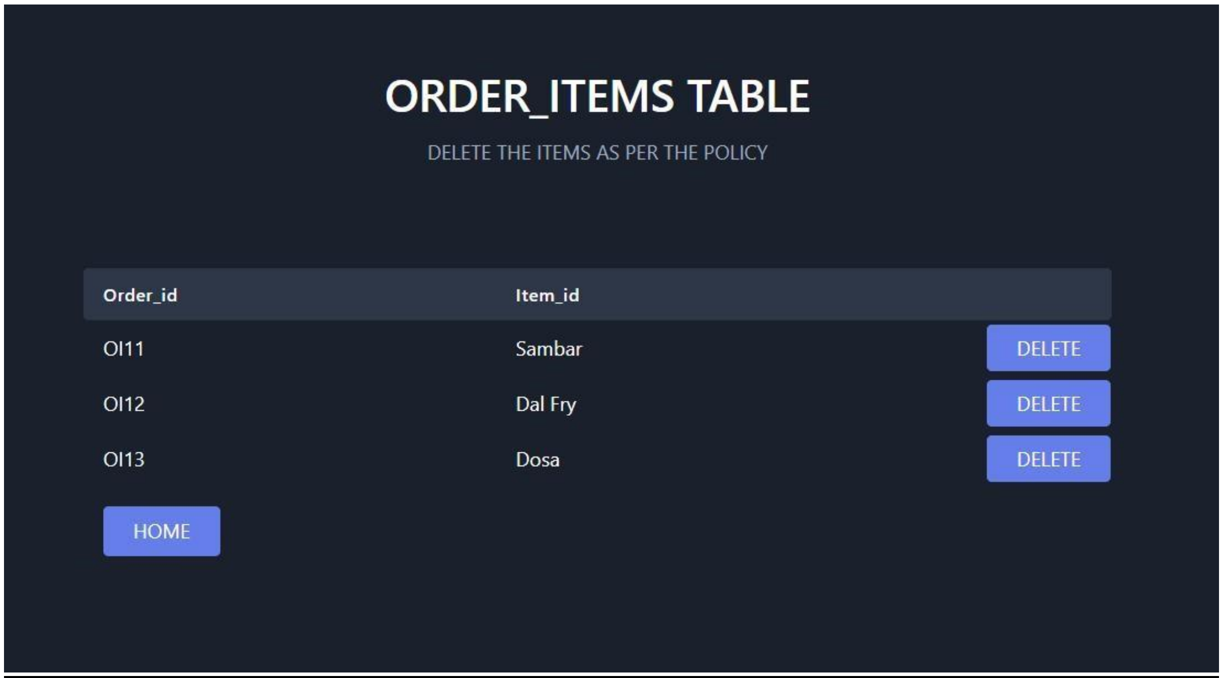
TABLES IN RESTRAUNT
- VIEW TABLES IN RESTRAUNT



ITEMS TABLE

ENTER YOUR MENU CARD ITEMS

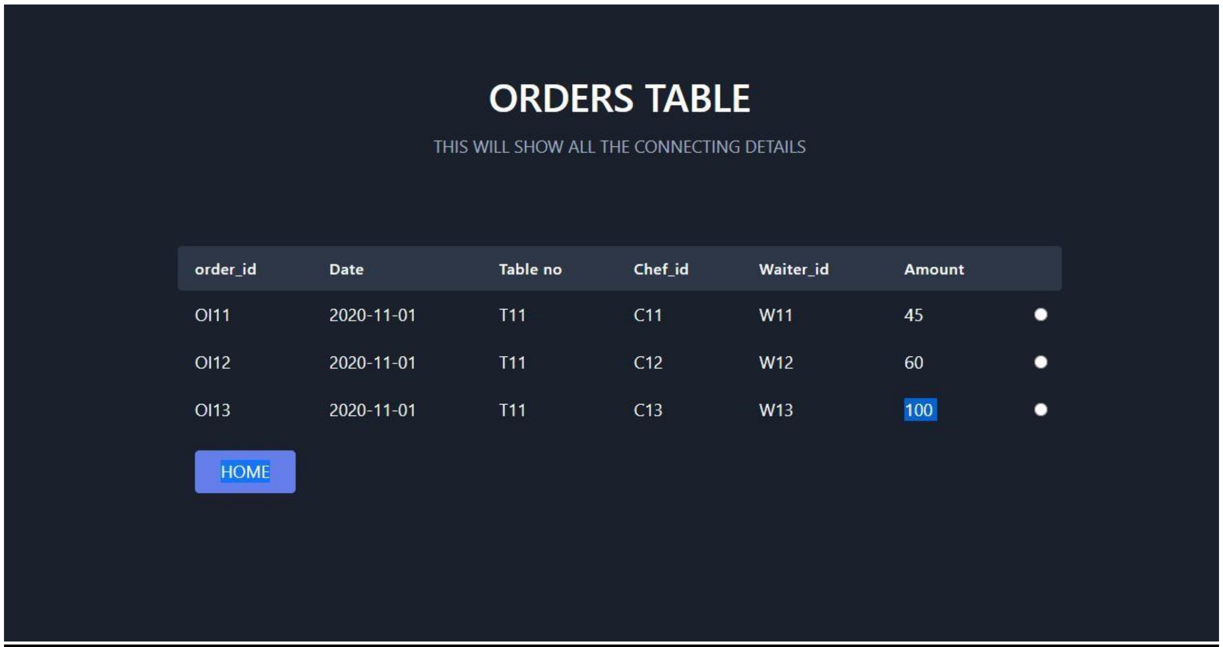
Item_id	Itemname	Price	Type	Link For Image
I108			VEG	
I101	Dosa	25	VEG	DELETE
I102	Sambar	15	VEG	DELETE
I105	Dal Fry	20	VEG	DELETE
I106	Aloo Biryani	130	VEG	DELETE
I103	Chicken Korma	100	NON VEG	DELETE
I104	Roast Chicken	150	NON VEG	DELETE
I107	Chicken Biryani	180	NON VEG	DELETE



ORDER_ITEMS TABLE

DELETE THE ITEMS AS PER THE POLICY

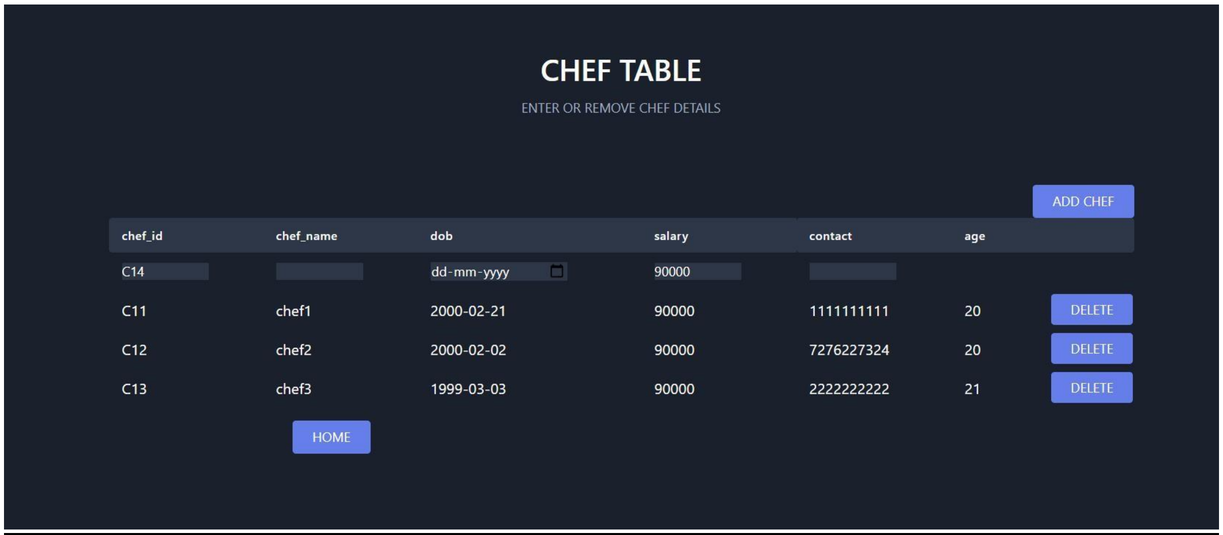
Order_id	Item_id	
OI11	Sambar	DELETE
OI12	Dal Fry	DELETE
OI13	Dosa	DELETE



ORDERS TABLE

THIS WILL SHOW ALL THE CONNECTING DETAILS

order_id	Date	Table no	Chef_id	Waiter_id	Amount	
OI11	2020-11-01	T11	C11	W11	45	●
OI12	2020-11-01	T11	C12	W12	60	●
OI13	2020-11-01	T11	C13	W13	100	●

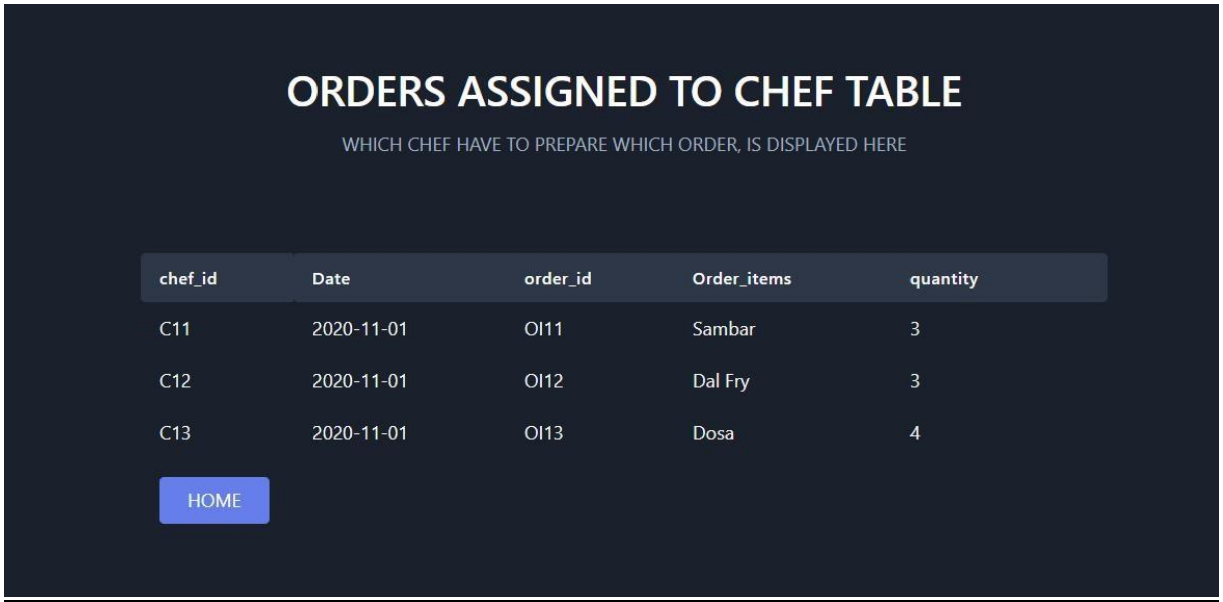

CHEF TABLE

ENTER OR REMOVE CHEF DETAILS

ADD CHEF

chef_id	chef_name	dob	salary	contact	age	
C14		dd-mm-yyyy	90000			
C11	chef1	2000-02-21	90000	1111111111	20	DELETE
C12	chef2	2000-02-02	90000	7276227324	20	DELETE
C13	chef3	1999-03-03	90000	2222222222	21	DELETE

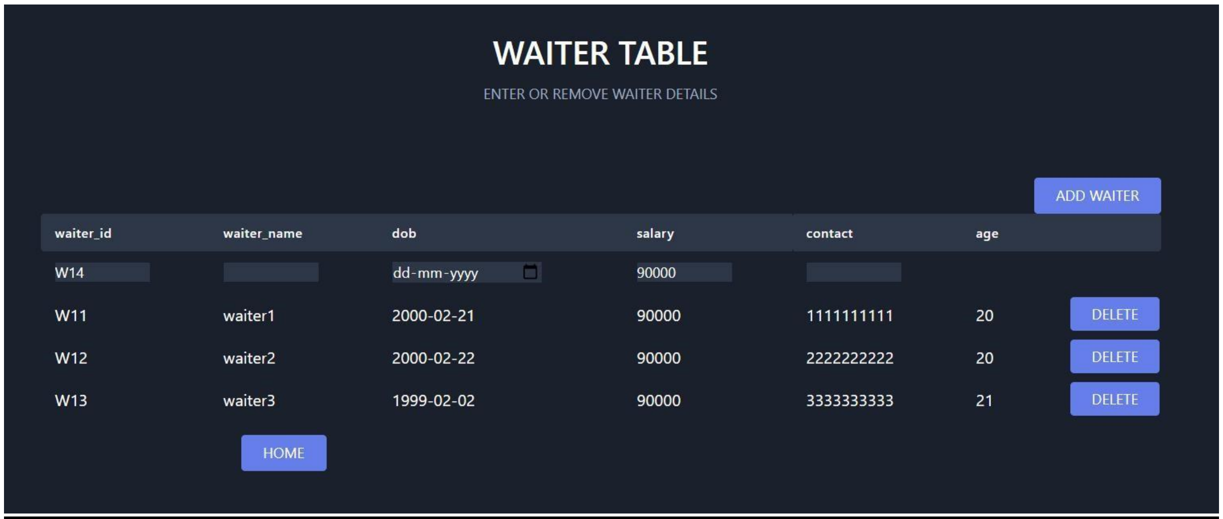
HOME


ORDERS ASSIGNED TO CHEF TABLE

WHICH CHEF HAVE TO PREPARE WHICH ORDER, IS DISPLAYED HERE

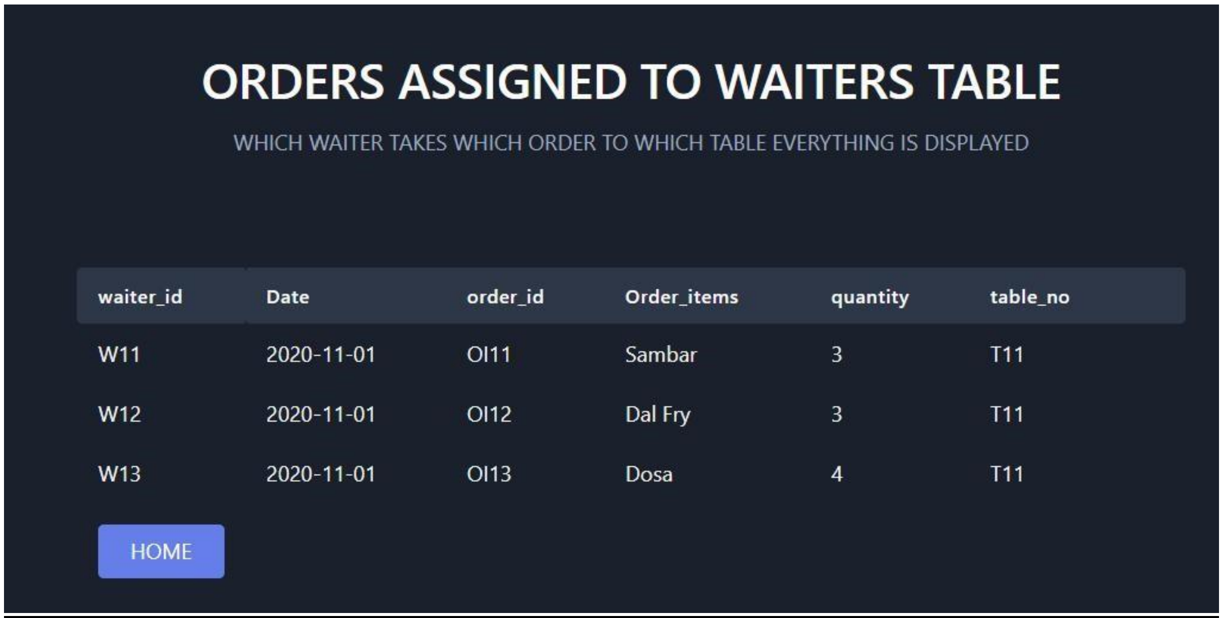
chef_id	Date	order_id	Order_items	quantity
C11	2020-11-01	OI11	Sambar	3
C12	2020-11-01	OI12	Dal Fry	3
C13	2020-11-01	OI13	Dosa	4

HOME


WAITER TABLE

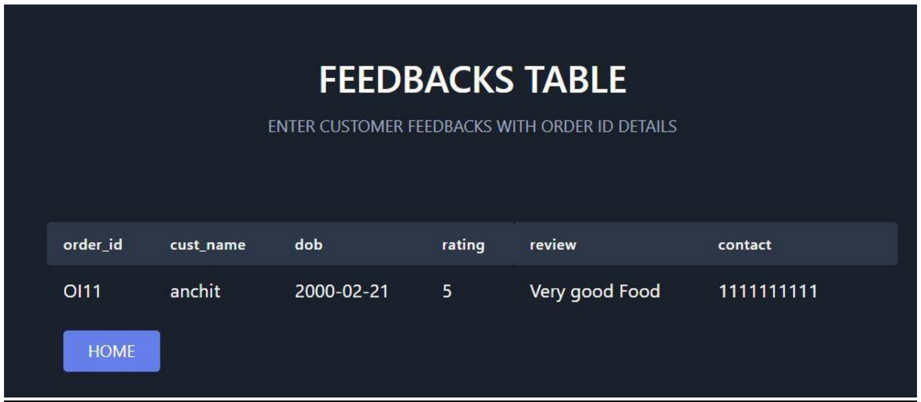
ENTER OR REMOVE WAITER DETAILS

waiter_id	waiter_name	dob	salary	contact	age	
W14		dd-mm-yyyy	90000			
W11	waiter1	2000-02-21	90000	1111111111	20	DELETE
W12	waiter2	2000-02-22	90000	2222222222	20	DELETE
W13	waiter3	1999-02-02	90000	3333333333	21	DELETE


ORDERS ASSIGNED TO WAITERS TABLE

WHICH WAITER TAKES WHICH ORDER TO WHICH TABLE EVERYTHING IS DISPLAYED

waiter_id	Date	order_id	Order_items	quantity	table_no
W11	2020-11-01	OI11	Sambar	3	T11
W12	2020-11-01	OI12	Dal Fry	3	T11
W13	2020-11-01	OI13	Dosa	4	T11


FEEDBACKS TABLE

ENTER CUSTOMER FEEDBACKS WITH ORDER ID DETAILS

order_id	cust_name	dob	rating	review	contact
OI11	anchit	2000-02-21	5	Very good Food	1111111111

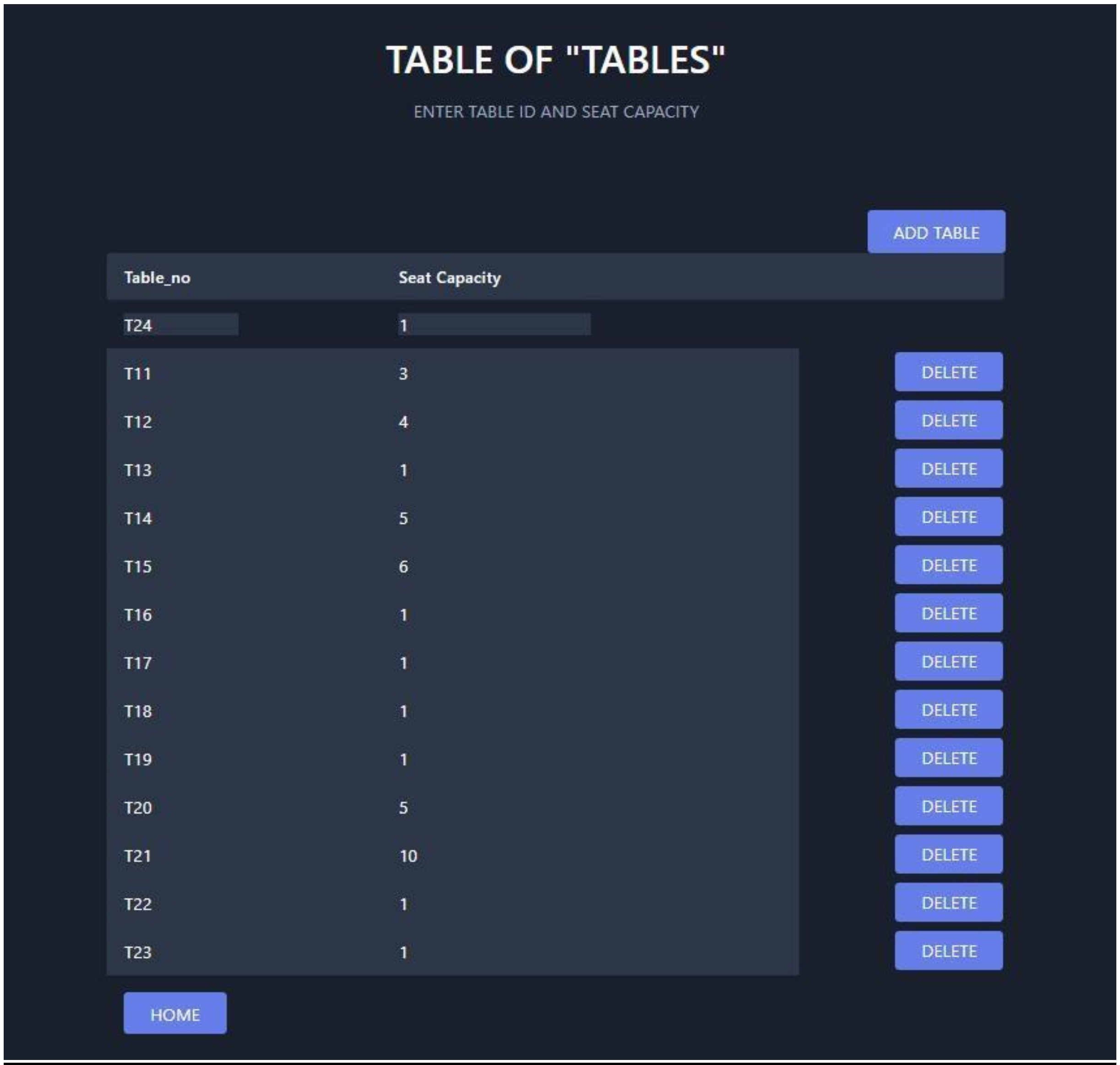


TABLE OF "TABLES"

ENTER TABLE ID AND SEAT CAPACITY

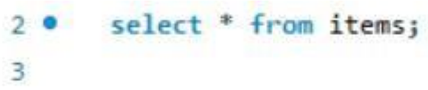
Table_no	Seat Capacity
T24	1
T11	3
T12	4
T13	1
T14	5
T15	6
T16	1
T17	1
T18	1
T19	1
T20	5
T21	10
T22	1
T23	1

6.3. BRIEF SUMMARY OF THE CODE WITH FEW QUERIES

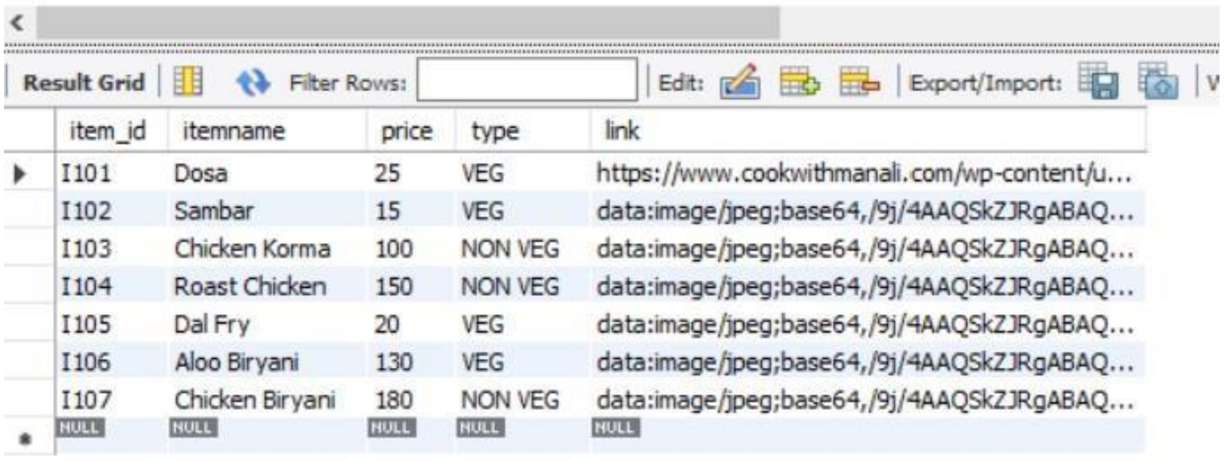
- All queries are written inside flask.
- In flask we have made routes (web pages) inside which we have defined https post and get requests.

- We have used insert, nested, join, delete and update queries in our project.
- We have also developed such an algorithm in query that chefs and waiters will be assigned in round robin fashion.

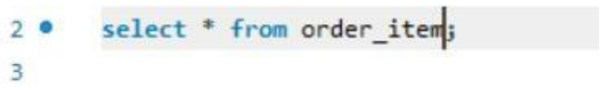
<u>SOME EXAMPLES</u>



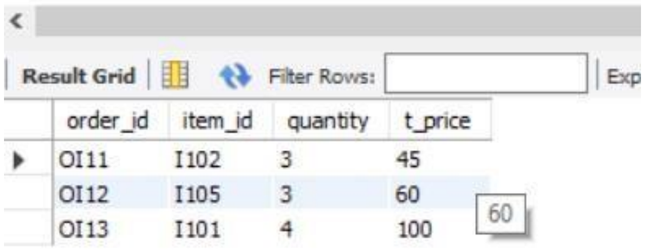
```
select * from items;
```



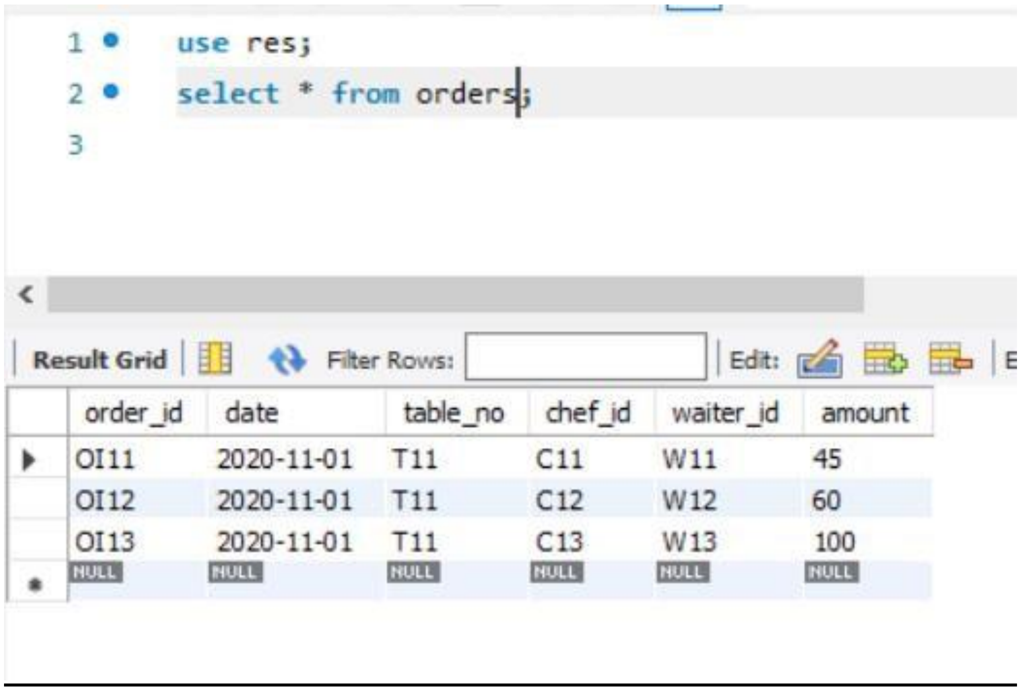
	item_id	itemname	price	type	link
▶	I101	Dosa	25	VEG	https://www.cookwithmanali.com/wp-content/u...
	I102	Sambar	15	VEG	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...
	I103	Chicken Korma	100	NON VEG	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...
	I104	Roast Chicken	150	NON VEG	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...
	I105	Dal Fry	20	VEG	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...
	I106	Aloo Biryani	130	VEG	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...
	I107	Chicken Biryani	180	NON VEG	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...
*	NULL	NULL	NULL	NULL	NULL



```
select * from order_item;
```

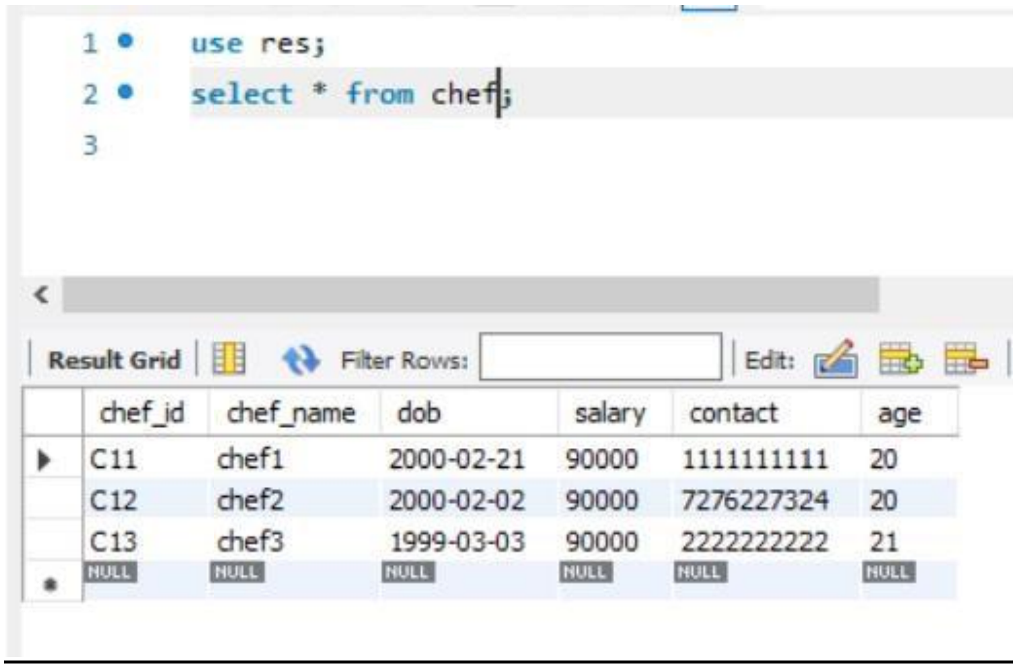


	order_id	item_id	quantity	t_price
▶	OI11	I102	3	45
	OI12	I105	3	60
	OI13	I101	4	100



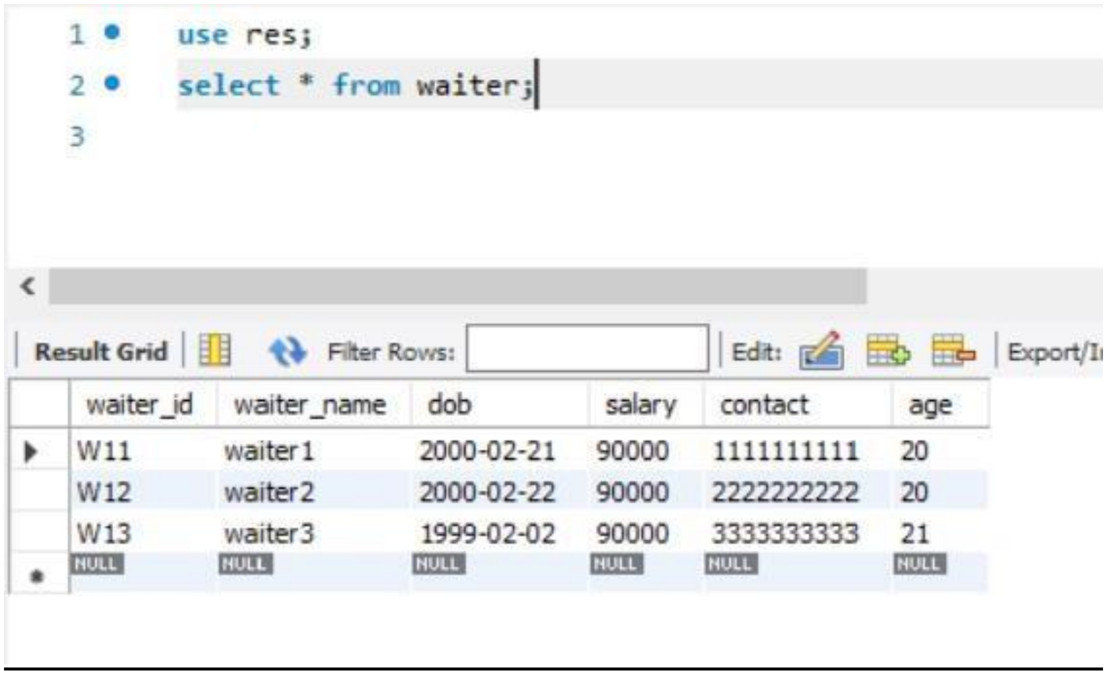
```
use res;
select * from orders;
```

order_id	date	table_no	chef_id	waiter_id	amount
OI11	2020-11-01	T11	C11	W11	45
OI12	2020-11-01	T11	C12	W12	60
OI13	2020-11-01	T11	C13	W13	100
NULL	NULL	NULL	NULL	NULL	NULL



```
use res;
select * from chef;
```

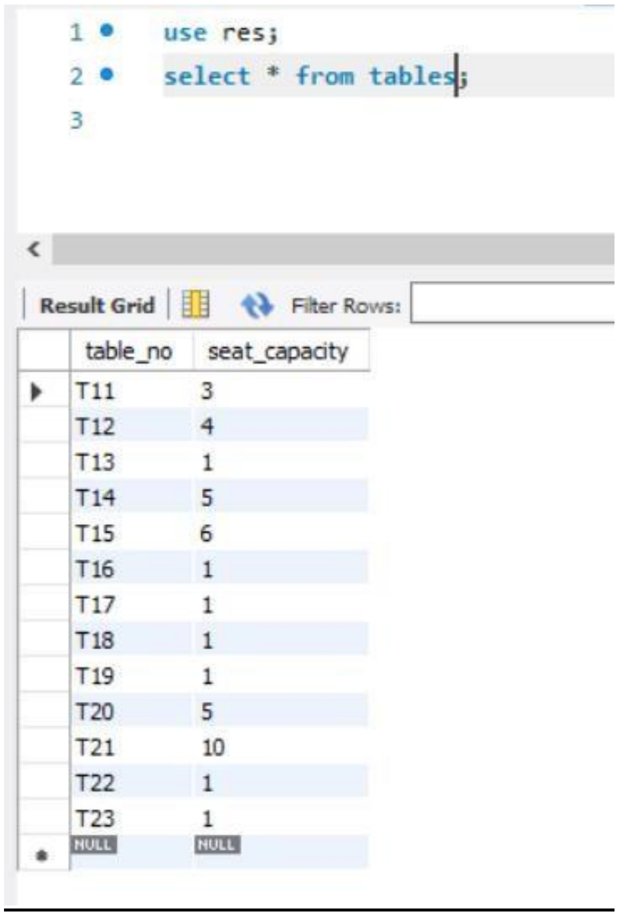
chef_id	chef_name	dob	salary	contact	age
C11	chef1	2000-02-21	90000	1111111111	20
C12	chef2	2000-02-02	90000	7276227324	20
C13	chef3	1999-03-03	90000	2222222222	21
NULL	NULL	NULL	NULL	NULL	NULL



```
use res;
select * from waiter;
```

Result Grid | Filter Rows: | Edit: | Export/I

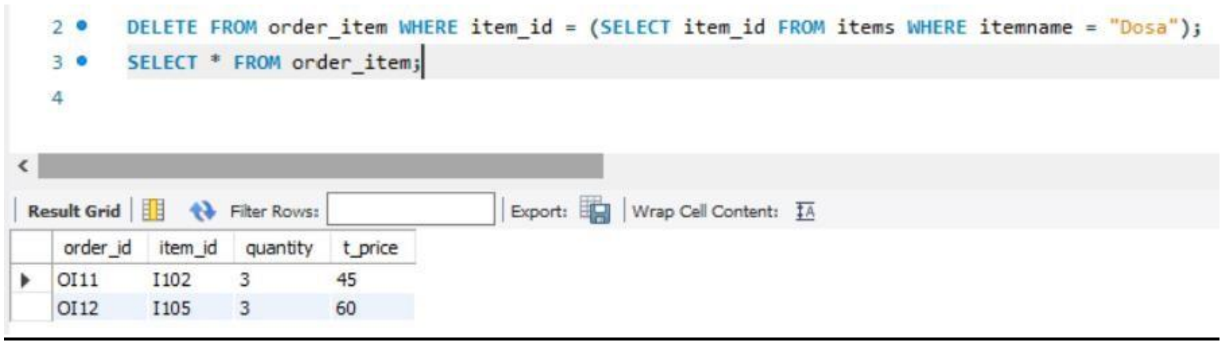
waiter_id	waiter_name	dob	salary	contact	age
W11	waiter1	2000-02-21	90000	1111111111	20
W12	waiter2	2000-02-22	90000	2222222222	20
W13	waiter3	1999-02-02	90000	3333333333	21
NULL	NULL	NULL	NULL	NULL	NULL



```
use res;
select * from tables;
```

Result Grid | Filter Rows:

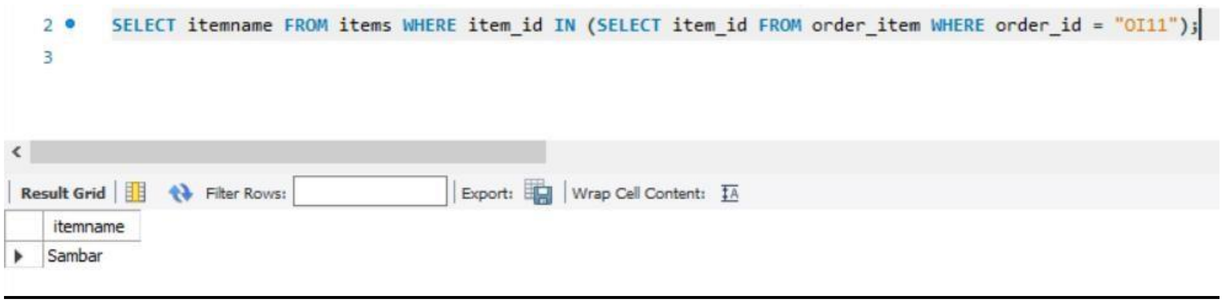
table_no	seat_capacity
T11	3
T12	4
T13	1
T14	5
T15	6
T16	1
T17	1
T18	1
T19	1
T20	5
T21	10
T22	1
T23	1
NULL	NULL



```sql
DELETE FROM order_item WHERE item_id = (SELECT item_id FROM items WHERE itemname = "Dosa");
SELECT * FROM order_item;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

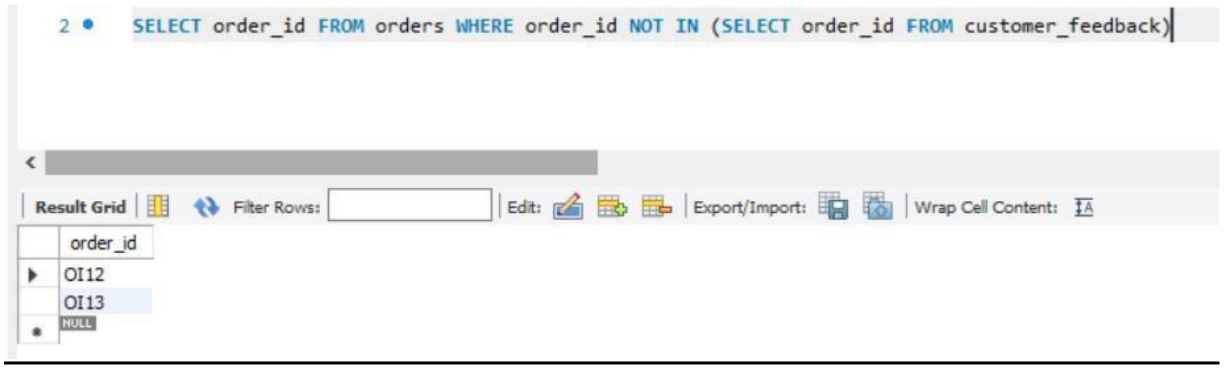
order_id	item_id	quantity	t_price
OI11	I102	3	45
OI12	I105	3	60



```sql
SELECT itemname FROM items WHERE item_id IN (SELECT item_id FROM order_item WHERE order_id = "OI11");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

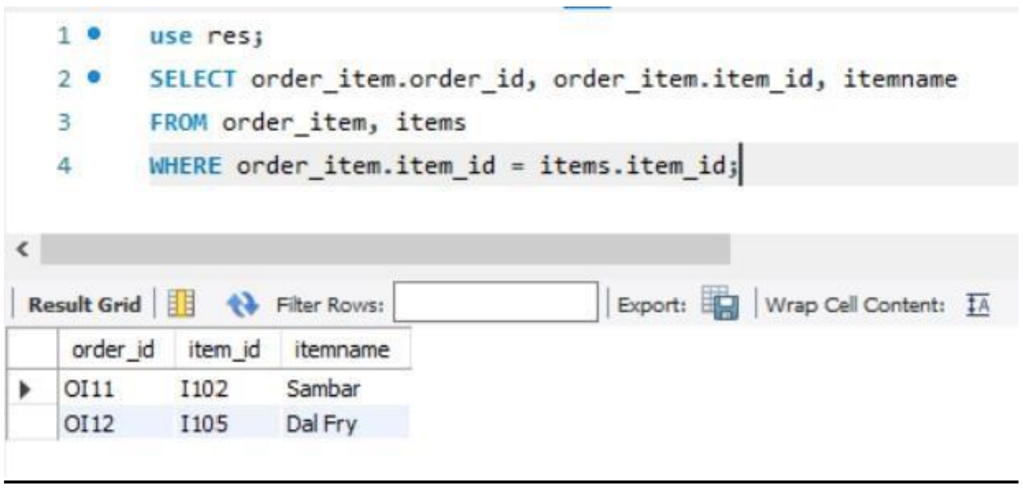
itemname
Sambar



```sql
SELECT order_id FROM orders WHERE order_id NOT IN (SELECT order_id FROM customer_feedback)
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

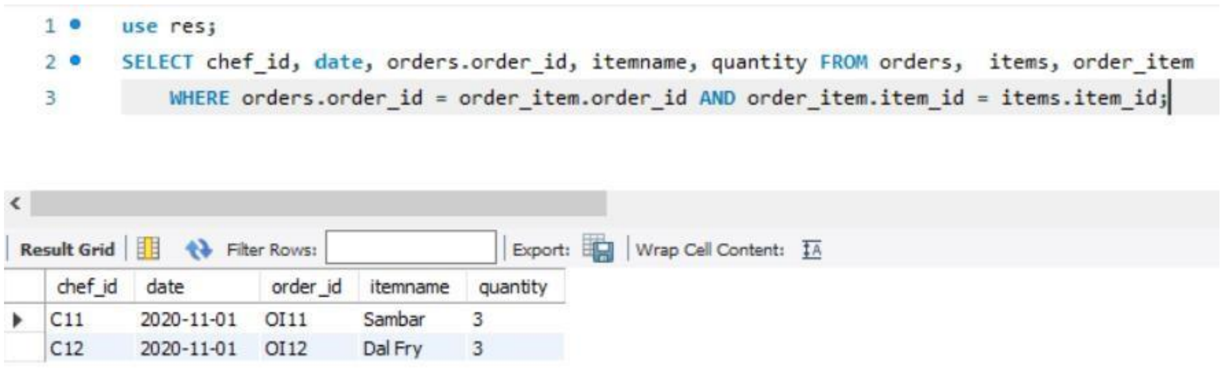
order_id
OI12
OI13
NULL



```sql
use res;
SELECT order_item.order_id, order_item.item_id, itemname
FROM order_item, items
WHERE order_item.item_id = items.item_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

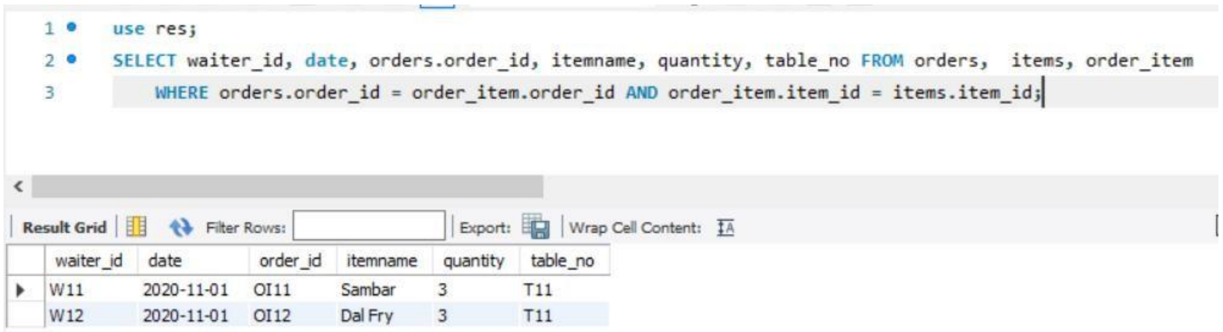
order_id	item_id	itemname
OI11	I102	Sambar
OI12	I105	Dal Fry



```sql
use res;
SELECT chef_id, date, orders.order_id, itemname, quantity FROM orders,  items, order_item
   WHERE orders.order_id = order_item.order_id AND order_item.item_id = items.item_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

chef_id	date	order_id	itemname	quantity
C11	2020-11-01	OI11	Sambar	3
C12	2020-11-01	OI12	Dal Fry	3

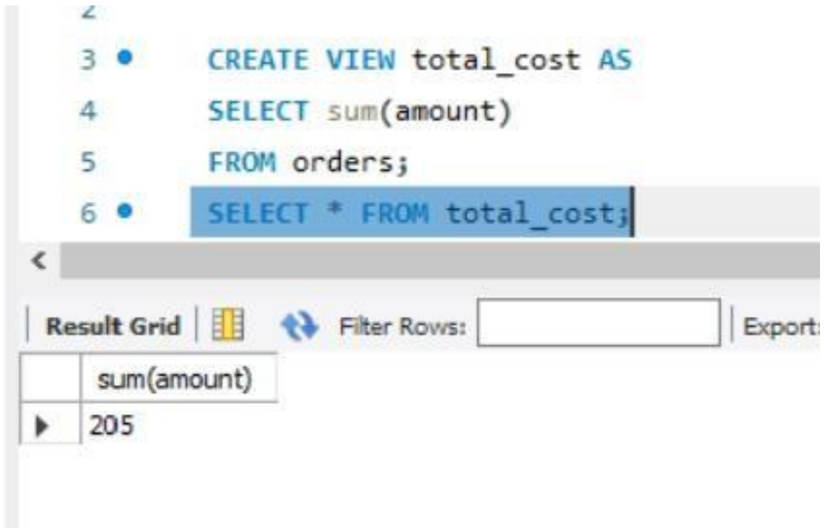


```sql
use res;
SELECT waiter_id, date, orders.order_id, itemname, quantity, table_no FROM orders,  items, order_item
   WHERE orders.order_id = order_item.order_id AND order_item.item_id = items.item_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

waiter_id	date	order_id	itemname	quantity	table_no
W11	2020-11-01	OI11	Sambar	3	T11
W12	2020-11-01	OI12	Dal Fry	3	T11

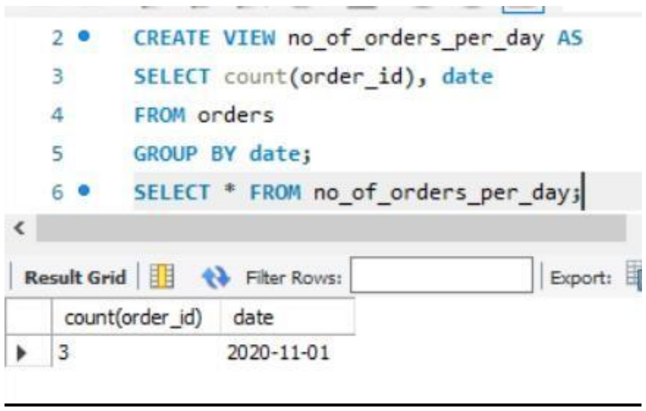
VIEWS



```sql
CREATE VIEW total_cost AS
SELECT sum(amount)
FROM orders;
SELECT * FROM total_cost;
```

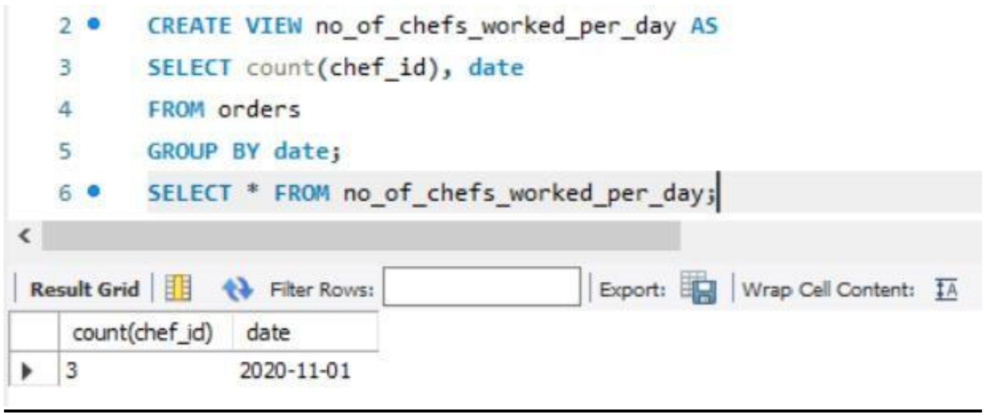
Result Grid | Filter Rows: | Export:

sum(amount)
205


```
CREATE VIEW no_of_orders_per_day AS
SELECT count(order_id), date
FROM orders
GROUP BY date;
SELECT * FROM no_of_orders_per_day;
```

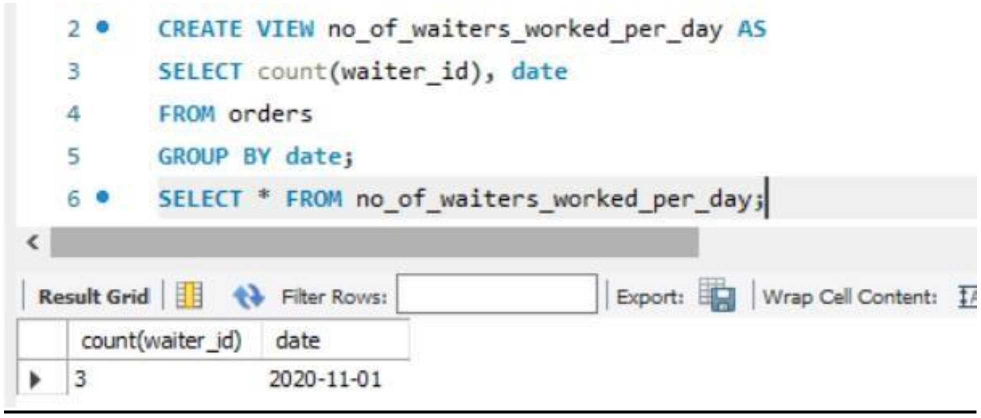
Result Grid | Filter Rows: | Export:

count(order_id)	date
3	2020-11-01


```
CREATE VIEW no_of_chefs_worked_per_day AS
SELECT count(chef_id), date
FROM orders
GROUP BY date;
SELECT * FROM no_of_chefs_worked_per_day;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(chef_id)	date
3	2020-11-01


```
CREATE VIEW no_of_waiters_worked_per_day AS
SELECT count(waiter_id), date
FROM orders
GROUP BY date;
SELECT * FROM no_of_waiters_worked_per_day;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(waiter_id)	date
3	2020-11-01

7. CONCLUSION

As the restaurant industry is growing bigger day by day , the fastness in the process of ordering and delivering the food has become a major issue in the success of the restaurant . The faster the service is, the more satisfied the customer is . When we use the old conventional methods , the process becomes slow , even if the difference is small, it affects largely on the business . So to compete with the growing industry and to survive we have to make the process as fast as it can be . Hence our project is a few steps closer to make the restaurant fast and successful . Our project makes the process a lot easier and less confusing as compared to sophisticated softwares used, hence making the process faster and more efficient.

8. FUTURE WORK

Innovation never stops, we can add some features like adding complete authorization to the website, making it available for multiple restaurants, including machine learning algorithms like sentimental analysis to feedbacks to yield more honest reviews, add options to download the reports in pdf form and mailing it to whosoever reviews the restaurant, and lastly, add payment portal which will make the payment faster and way more convenient.

But with respect to DBMS, I think we have done a very good job.