

# Introduction

This project involves finalizing a web platform for restaurant managers to manage reservations and dining experiences. The platform is built in Python and connects to a MySQL database. Restaurant managers can use the platform to add, modify, or cancel reservations and update dining preferences for regular customers.

**Link to repository: <https://github.com/sassaidi/CIS344-Final-Project.git>**

## Project Setup

### 1. Database Creation

#### Step 1: Create the Database

We start by creating a MySQL database named restaurant\_reservations.

```
CREATE DATABASE restaurant_reservations;  
USE restaurant_reservations;
```

#### Step 2: Create the Tables

Next, we create three tables: Customers, Reservations, and DiningPreferences.

```
CREATE TABLE Customers (  
    customerId INT NOT NULL UNIQUE AUTO_INCREMENT,  
    customerName VARCHAR(45) NOT NULL,  
    contactInfo VARCHAR(200),  
    PRIMARY KEY (customerId)  
);  
  
CREATE TABLE Reservations (  
    reservationId INT NOT NULL UNIQUE AUTO_INCREMENT,  
    customerId INT NOT NULL,  
    reservationTime DATETIME NOT NULL,
```

```
    numberOfGuests INT NOT NULL,  
    specialRequests VARCHAR(200),  
    PRIMARY KEY (reservationId),  
    FOREIGN KEY (customerId) REFERENCES Customers(customerId)  
);
```

```
CREATE TABLE DiningPreferences (  
    preferenceId INT NOT NULL UNIQUE AUTO_INCREMENT,  
    customerId INT NOT NULL,  
    favoriteTable VARCHAR(45),  
    dietaryRestrictions VARCHAR(200),  
    PRIMARY KEY (preferenceId),  
    FOREIGN KEY (customerId) REFERENCES Customers(customerId)  
);
```

### **3. Stored Procedures**

We create stored procedures for managing reservations and special requests.

```
DELIMITER //
```

```
CREATE PROCEDURE findReservations(IN customer_id INT)  
BEGIN  
    SELECT * FROM Reservations WHERE customerId = customer_id;  
END //
```

```
CREATE PROCEDURE addSpecialRequest(IN reservation_id INT, IN  
requests VARCHAR(200))  
BEGIN  
    UPDATE Reservations SET specialRequests = requests WHERE  
reservationId = reservation_id;  
END //
```

```
CREATE PROCEDURE addReservation(  
    IN customer_name VARCHAR(45),  
    IN contact_info VARCHAR(200),  
    IN reservation_time DATETIME,
```

```

        IN guests INT,
        IN special_requests VARCHAR(200)
    )
BEGIN
    DECLARE customer_id INT;

    -- Check if customer exists
    SELECT customerId INTO customer_id FROM Customers WHERE
customerName = customer_name AND contactInfo = contact_info;

    -- If customer does not exist, add them
    IF customer_id IS NULL THEN
        INSERT INTO Customers (customerName, contactInfo) VALUES
(customer_name, contact_info);
        SET customer_id = LAST_INSERT_ID();
    END IF;

    -- Add reservation
    INSERT INTO Reservations (customerId, reservationTime,
numberOfGuests, specialRequests)
VALUES (customer_id, reservation_time, guests, special_requests);
END //

DELIMITER ;

```

## Python Integration

### 1. Configuration

We configure the restaurantDatabase.py file to connect to the MySQL database.

```
import mysql.connector
```

```
class Database:
```

```

    def __init__(self, host='localhost', user='root',
password='your_password', database='restaurant_reservations'):
        self.connection = mysql.connector.connect(
            host=host,
```

```
        user=user,  
        password=password,  
        database=database  
    )  
    self.cursor = self.connection.cursor()
```

## 2. Running the Server

We run restaurantServer.py to start the platform.

```
``python restaurantServer.py``
```

## 3. Adding Reservations

We complete the addReservation method in restaurantDatabase.py to add new reservations.

```
def addReservation(self, customerName, contactInfo, reservationTime,  
    numberOfGuests, specialRequests):  
    self.cursor.callproc('addReservation', [customerName, contactInfo,  
        reservationTime, numberOfGuests, specialRequests])  
    self.connection.commit()
```

## 4. Additional Methods (Bonus)

We implement additional methods such as addSpecialRequest, findReservations, and deleteReservation.

```
def addSpecialRequest(self, reservationId, requests):  
    self.cursor.callproc('addSpecialRequest', [reservationId, requests])  
    self.connection.commit()
```

```
def findReservations(self, customerId):  
    self.cursor.callproc('findReservations', [customerId])  
    return self.cursor.fetchall()
```

```
def deleteReservation(self, reservationId):  
    self.cursor.execute('DELETE FROM Reservations WHERE reservationId = %s',  
        (reservationId,))  
    self.connection.commit()
```

## **Conclusion:**

This project successfully finalizes the restaurant management platform, providing essential functionalities for managing reservations and customer dining preferences. The integration with MySQL ensures efficient data management and retrieval, enhancing the overall user experience for restaurant managers.

**Screenshots of restaurantDatabase.py below:**

```
1  import mysql.connector
2  from mysql.connector import Error
3
4  ✓ class RestaurantDatabase:
5  ✓      def __init__(self,
6              host="localhost",
7              port="3306",
8              database="restaurant",
9              user="root",
10             password="Yemen123"):
11
12         self.host = host
13         self.port = port
14         self.database = database
15         self.user = user
16         self.password = password
17         self.connection = None
18         self.cursor = None
19         self.connect()
20
21  ✓ def connect(self):
22      try:
23          self.connection = mysql.connector.connect(
24              host=self.host,
25              port=self.port,
26              database=self.database,
27              user=self.user,
28              password=self.password
29          )
30          if self.connection.is_connected():
31              print("Successfully connected to the database")
```

```

32         except Error as e:
33             print("Error while connecting to MySQL:", e)
34
35     def addCustomer(self, customer_name, contact_info):
36         try:
37             if self.connection.is_connected():
38                 self.cursor = self.connection.cursor()
39                 query = "INSERT INTO customers (customer_name, contact_info) VALUES (%s, %s)"
40                 self.cursor.execute(query, (customer_name, contact_info))
41                 self.connection.commit()
42                 print("Customer added successfully")
43         except Error as e:
44             print("Error adding customer:", e)
45
46     def findCustomer(self, customer_name):
47         try:
48             if self.connection.is_connected():
49                 self.cursor = self.connection.cursor()
50                 query = "SELECT customerId FROM customers WHERE customer_name = %s"
51                 self.cursor.execute(query, (customer_name,))
52                 result = self.cursor.fetchone()
53                 if result:
54                     return result[0] # Return customerId if customer exists
55                 else:
56                     return None # Customer not found
57         except Error as e:
58             print("Error finding customer:", e)
59             return None
60
61     def addReservation(self, customer_name, reservation_time, number_of_guests, special_requests):
62         try:
63             if self.connection.is_connected():
64
65                 # Check if customer exists or add new customer
66                 customer_id = self.findCustomer(customer_name)
67                 if not customer_id:
68                     self.addCustomer(customer_name, "") # Add customer with empty contact info
69                     customer_id = self.findCustomer(customer_name)
70
71                 # Insert reservation
72                 self.cursor = self.connection.cursor()
73                 query = "INSERT INTO reservations (customerId, reservation_time, number_of_guests, special_requests) VALUES (%s, %s, %s, %s)"
74                 self.cursor.execute(query, (customer_id, reservation_time, number_of_guests, special_requests))
75                 self.connection.commit()
76                 print("Reservation added successfully")
77         except Error as e:
78             print("Error adding reservation:", e)
79
80     def getAllReservations(self):
81         try:
82             if self.connection.is_connected():
83                 self.cursor = self.connection.cursor()
84                 query = "SELECT r.reservationId, c.customer_name, r.reservation_time, r.number_of_guests, r.special_requests FROM reservations r JOIN customers c ON r.customerId = c.customerId"
85                 self.cursor.execute(query)
86                 records = self.cursor.fetchall()
87                 return records
88         except Error as e:
89             print("Error fetching reservations:", e)
90             return []
91
92     def closeConnection(self):
93         try:
94             if self.connection.is_connected():
95                 self.connection.close()
96                 print("MySQL connection is closed")
97         except Error as e:
98             print("Error closing MySQL connection:", e)

```

## Screenshots of restaurantServer.py below:

```
1  from http.server import HTTPServer, BaseHTTPRequestHandler
2  import cgi
3  import mysql.connector
4  from mysql.connector import Error
5
6  # Database connection parameters
7  db_name = "restaurant"
8  db_user = "root"
9  db_password = "Yemen123"
10 db_host = "localhost"
11 db_port = "3306"
12
13 class RestaurantDatabase:
14     def __init__(self):
15         # Initialize database connection
16         try:
17             self.connection = mysql.connector.connect(
18                 database=db_name,
19                 user=db_user,
20                 password=db_password,
21                 host=db_host,
22                 port=db_port
23             )
24             print("Connected to database successfully")
25         except Error as error:
26             print("Error connecting to MySQL:", error)
27
28     def __del__(self):
29         if hasattr(self, 'connection') and self.connection.is_connected():
30             self.connection.close()
31             print("MySQL connection is closed")
32
33     def addReservation(self, customer_id, reservation_time, number_of_guests, special_requests):
34         try:
35             cursor = self.connection.cursor()
36             insert_query = '''
37                 INSERT INTO reservations (customerId, reservationTime, numberOfGuests, specialRequests)
38                 VALUES (%s, %s, %s, %s)
39             '''
40             cursor.execute(insert_query, (customer_id, reservation_time, number_of_guests, special_requests))
41             self.connection.commit()
42             print("Reservation added successfully")
43         except Error as e:
44             print("Error adding reservation:", e)
45
46     def getAllReservations(self):
47         try:
48             cursor = self.connection.cursor()
49             select_query = "SELECT * FROM reservations"
50             cursor.execute(select_query)
51             records = cursor.fetchall()
52             return records
53         except Error as e:
54             print("Error fetching reservations:", e)
55             return []
56
57 class RestaurantPortalHandler(BaseHTTPRequestHandler):
58     def __init__(self, *args, **kwargs):
59         self.database = RestaurantDatabase()
60         super().__init__(*args, **kwargs)
61
62     def do_POST(self):
63         try:
64             if self.path == '/addReservation':
```



```

65         form = cgi.FieldStorage(
66             fp=self.rfile,
67             headers=self.headers,
68             environ={'REQUEST_METHOD': 'POST'})
69     )
70     customer_id = int(form.getvalue("customer_id"))
71     reservation_time = form.getvalue("reservation_time")
72     number_of_guests = int(form.getvalue("number_of_guests"))
73     special_requests = form.getvalue("special_requests")
74
75     self.database.addReservation(customer_id, reservation_time, number_of_guests, special_re
76
77     self.send_response(200)
78     self.send_header('Content-type', 'text/html')
79     self.end_headers()
80     self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
81     self.wfile.write(b"<body>")
82     self.wfile.write(b"<center><h1>Reservation Added</h1>")
83     self.wfile.write(b"<hr>")
84     self.wfile.write(b"<div><a href='/addReservation'>Add Another Reservation</a></div>")
85     self.wfile.write(b"<div><a href='/'>Home</a></div>")
86     self.wfile.write(b"</center></body></html>")
87
88     except Exception as e:
89         self.send_error(500, f'Internal Server Error: {e}')
90
91     def do_GET(self):
92         try:
93             if self.path == '/':
94                 self.handle_root_request()
95
96                 self.handle_root_request()
97             elif self.path == '/addReservation':
98                 self.render_add_reservation_form()
99             elif self.path == '/viewReservations':
100                 self.view_all_reservations()
101             else:
102                 self.send_error(404, f'File Not Found: {self.path}')
103
104         except Exception as e:
105             self.send_error(500, f'Internal Server Error: {e}')
106
107     def handle_root_request(self):
108         try:
109             # Fetch all reservations from the database
110             records = self.database.getAllReservations()
111
112             # Start building the HTML response
113             self.send_response(200)
114             self.send_header('Content-type', 'text/html')
115             self.end_headers()
116
117             # Begin HTML content
118             self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
119             self.wfile.write(b"<body>")
120             self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
121             self.wfile.write(b"<hr>")
122             self.wfile.write(b"<div> <a href='/'>Home</a>| \
123                             <a href='/addReservation'>Add Reservation</a>|\

```

```

122         <a href='/viewReservations'>View Reservations</a></div>")
123     self.wfile.write(b"<hr><h2>All Reservations</h2>")
124     self.wfile.write(b"<table border=1>")
125     self.wfile.write(b"<tr><th>Reservation ID</th><th>Customer ID</th><th>Reservation")
126
127     # Iterate through records and build table rows
128     for row in records:
129         self.wfile.write(b"<tr>")
130         for item in row:
131             self.wfile.write(f"<td>{item}</td>".encode())
132         self.wfile.write(b"</tr>")
133
134     # End HTML content
135     self.wfile.write(b"</table></center>")
136     self.wfile.write(b"</body></html>")
137
138     except Error as e:
139         self.send_error(500, f'Internal Server Error: {e}')
140
141     def render_add_reservation_form(self):
142         # This method remains the same as before
143         self.send_response(200)
144         self.send_header('Content-type', 'text/html')
145         self.end_headers()
146         self.wfile.write(b"<html><head><title>Add Reservation</title></head>")
147         self.wfile.write(b"<body>")
148         self.wfile.write(b"<center><h1>Add Reservation</h1>")
149         self.wfile.write(b"<form method='post' action='/addReservation'>")
150
151         self.wfile.write(b"<center><h1>Add Reservation</h1>")
152         self.wfile.write(b"<form method='post' action='/addReservation'>")
153         self.wfile.write(b"Customer ID: <input type='text' name='customer_id'><br>")
154         self.wfile.write(b"Reservation Time: <input type='text' name='reservation_time'><br>")
155         self.wfile.write(b"Number of Guests: <input type='text' name='number_of_guests'><br>")
156         self.wfile.write(b"Special Requests: <input type='text' name='special_requests'><br>")
157         self.wfile.write(b"<input type='submit' value='Add Reservation'>")
158         self.wfile.write(b"</form>")
159         self.wfile.write(b"</center></body></html>")
160
161     def view_all_reservations(self):
162         # This method remains the same as before
163         records = self.database.getAllReservations()
164         self.send_response(200)
165         self.send_header('Content-type', 'text/html')
166         self.end_headers()
167         self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
168         self.wfile.write(b"<body>")
169         self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
170         self.wfile.write(b"<hr>")
171         self.wfile.write(b"<div> <a href='/'>Home</a> | \
172             <a href='/addReservation'>Add Reservation</a> | \
173             <a href='/viewReservations'>View Reservations</a></div>")
174         self.wfile.write(b"<hr><h2>All Reservations</h2>")
175         self.wfile.write(b"<table border=1>")
176         self.wfile.write(b"<tr><th>Reservation ID</th><th>Customer ID</th><th>Reservation Time")
177         for row in records:
178             self.wfile.write(b"<tr>")

```

```

162         self.send_header('Content-type', 'text/html')
163         self.end_headers()
164         self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
165         self.wfile.write(b"<body>")
166         self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
167         self.wfile.write(b"<hr>")
168         self.wfile.write(b"<div> <a href='/'>Home</a>| \
169                             <a href='/addReservation'>Add Reservation</a>|\
170                             <a href='/viewReservations'>View Reservations</a></div>")
171         self.wfile.write(b"<hr><h2>All Reservations</h2>")
172         self.wfile.write(b"<table border=1>")
173         self.wfile.write(b"<tr><th>Reservation ID</th><th>Customer ID</th><th>Reservation Time</th><th>N")
174         for row in records:
175             self.wfile.write(b"<tr>")
176             for item in row:
177                 self.wfile.write(f"<td>{item}</td>".encode())
178             self.wfile.write(b"</tr>")
179         self.wfile.write(b"</table></center>")
180         self.wfile.write(b"</body></html>")
181
182     def run(server_class=HTTPServer, handler_class=RestaurantPortalHandler, port=8000):
183         server_address = ('localhost', port)
184         httpd = server_class(server_address, handler_class)
185         print(f'Starting httpd on port {port}')
186         httpd.serve_forever()
187
188     if __name__ == "__main__":
189         run()

```