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:

```python
import mysql.connector
from mysql.connector import Error

class RestaurantDatabase:
    def __init__(self,
                 host="localhost",
                 port="3306",
                 database="restaurant",
                 user="root",
                 password="Yemen123"):

        self.host = host
        self.port = port
        self.database = database
        self.user = user
        self.password = password
        self.connection = None
        self.cursor = None
        self.connect()

    def connect(self):
        try:
            self.connection = mysql.connector.connect(
                host=self.host,
                port=self.port,
                database=self.database,
                user=self.user,
                password=self.password
            )
            if self.connection.is_connected():
                print("Successfully connected to the database")
```

```
            except Error as e:
                print("Error while connecting to MySQL:", e)

    def addCustomer(self, customer_name, contact_info):
        try:
            if self.connection.is_connected():
                self.cursor = self.connection.cursor()
                query = "INSERT INTO customers (customer_name, contact_info) VALUES (%
                self.cursor.execute(query, (customer_name, contact_info))
                self.connection.commit()
                print("Customer added successfully")
        except Error as e:
            print("Error adding customer:", e)

    def findCustomer(self, customer_name):
        try:
            if self.connection.is_connected():
                self.cursor = self.connection.cursor()
                query = "SELECT customerId FROM customers WHERE customer_name = %s"
                self.cursor.execute(query, (customer_name,))
                result = self.cursor.fetchone()
                if result:
                    return result[0]  # Return customerId if customer exists
                else:
                    return None  # Customer not found
        except Error as e:
            print("Error finding customer:", e)
            return None

    def addReservation(self, customer_name, reservation_time, number_of_guests, specia
        try:
            if self.connection.is_connected():

                # Check if customer exists or add new customer
                customer_id = self.findCustomer(customer_name)
                if not customer_id:
                    self.addCustomer(customer_name, "")  # Add customer with empty cont
                    customer_id = self.findCustomer(customer_name)

                # Insert reservation
                self.cursor = self.connection.cursor()
                query = "INSERT INTO reservations (customerId, reservation_time, numbe
                self.cursor.execute(query, (customer_id, reservation_time, number_of_gu
                self.connection.commit()
                print("Reservation added successfully")
        except Error as e:
            print("Error adding reservation:", e)

    def getAllReservations(self):
        try:
            if self.connection.is_connected():
                self.cursor = self.connection.cursor()
                query = "SELECT r.reservationId, c.customer_name, r.reservation_time, r
                self.cursor.execute(query)
                records = self.cursor.fetchall()
                return records
        except Error as e:
            print("Error fetching reservations:", e)
            return []

    def closeConnection(self):
```

```
    def closeConnection(self):
        try:
            if self.connection.is_connected():
                self.connection.close()
                print("MySQL connection is closed")
        except Error as e:
            print("Error closing MySQL connection:", e)
```

Screenshots of restaurantServer.py below:

```python
from http.server import HTTPServer, BaseHTTPRequestHandler
import cgi
import mysql.connector
from mysql.connector import Error

# Database connection parameters
db_name = "restaurant"
db_user = "root"
db_password = "Yemen123"
db_host = "localhost"
db_port = "3306"

class RestaurantDatabase:
    def __init__(self):
        # Initialize database connection
        try:
            self.connection = mysql.connector.connect(
                database=db_name,
                user=db_user,
                password=db_password,
                host=db_host,
                port=db_port
            )
            print("Connected to database successfully")
        except Error as error:
            print("Error connecting to MySQL:", error)

    def __del__(self):
        if hasattr(self, 'connection') and self.connection.is_connected():
            self.connection.close()
            print("MySQL connection is closed")

    def addReservation(self, customer_id, reservation_time, number_of_guests, special_requests):
```

```python
        try:
            cursor = self.connection.cursor()
            insert_query = '''
                INSERT INTO reservations (customerId, reservationTime, numberOfGuests, specialRequests)
                VALUES (%s, %s, %s, %s)
            '''
            cursor.execute(insert_query, (customer_id, reservation_time, number_of_guests, special_reque
            self.connection.commit()
            print("Reservation added successfully")
        except Error as e:
            print("Error adding reservation:", e)

    def getAllReservations(self):
        try:
            cursor = self.connection.cursor()
            select_query = "SELECT * FROM reservations"
            cursor.execute(select_query)
            records = cursor.fetchall()
            return records
        except Error as e:
            print("Error fetching reservations:", e)
            return []

class RestaurantPortalHandler(BaseHTTPRequestHandler):
    def __init__(self, *args, **kwargs):
        self.database = RestaurantDatabase()
        super().__init__(*args, **kwargs)

    def do_POST(self):
        try:
            if self.path == '/addReservation':
```

```
                form = cgi.FieldStorage(
                    fp=self.rfile,
                    headers=self.headers,
                    environ={'REQUEST_METHOD': 'POST'}
                )
                customer_id = int(form.getvalue("customer_id"))
                reservation_time = form.getvalue("reservation_time")
                number_of_guests = int(form.getvalue("number_of_guests"))
                special_requests = form.getvalue("special_requests")

                self.database.addReservation(customer_id, reservation_time, number_of_guests, special_re

                self.send_response(200)
                self.send_header('Content-type', 'text/html')
                self.end_headers()
                self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
                self.wfile.write(b"<body>")
                self.wfile.write(b"<center><h1>Reservation Added</h1>")
                self.wfile.write(b"<hr>")
                self.wfile.write(b"<div><a href='/addReservation'>Add Another Reservation</a></div>")
                self.wfile.write(b"<div><a href='/'>Home</a></div>")
                self.wfile.write(b"</center></body></html>")

            except Exception as e:
                self.send_error(500, f'Internal Server Error: {e}')

    def do_GET(self):
        try:
            if self.path == '/':
                self.handle_root_request()
```

```
                self.handle_root_request()
            elif self.path == '/addReservation':
                self.render_add_reservation_form()
            elif self.path == '/viewReservations':
                self.view_all_reservations()
            else:
                self.send_error(404, f'File Not Found: {self.path}')

        except Exception as e:
            self.send_error(500, f'Internal Server Error: {e}')

    def handle_root_request(self):
        try:
            # Fetch all reservations from the database
            records = self.database.getAllReservations()

            # Start building the HTML response
            self.send_response(200)
            self.send_header('Content-type', 'text/html')
            self.end_headers()

            # Begin HTML content
            self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
            self.wfile.write(b"<body>")
            self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
            self.wfile.write(b"<hr>")
            self.wfile.write(b"<div> <a href='/'>Home</a>| \
                               <a href='/addReservation'>Add Reservation</a>|\
```

```
                            <a href='/viewReservations'>View Reservations</a></div>")
            self.wfile.write(b"<hr><h2>All Reservations</h2>")
            self.wfile.write(b"<table border=1>")
            self.wfile.write(b"<tr><th>Reservation ID</th><th>Customer ID</th><th>Reservation

            # Iterate through records and build table rows
            for row in records:
                self.wfile.write(b"<tr>")
                for item in row:
                    self.wfile.write(f"<td>{item}</td>".encode())
                self.wfile.write(b"</tr>")

            # End HTML content
            self.wfile.write(b"</table></center>")
            self.wfile.write(b"</body></html>")

        except Error as e:
            self.send_error(500, f'Internal Server Error: {e}')

    def render_add_reservation_form(self):
        # This method remains the same as before
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(b"<html><head><title>Add Reservation</title></head>")
        self.wfile.write(b"<body>")
        self.wfile.write(b"<center><h1>Add Reservation</h1>")
        self.wfile.write(b"<form method='post' action='/addReservation'>")

        self.wfile.write(b"<center><h1>Add Reservation</h1>")
        self.wfile.write(b"<form method='post' action='/addReservation'>")
        self.wfile.write(b"Customer ID: <input type='text' name='customer_id'><br>")
        self.wfile.write(b"Reservation Time: <input type='text' name='reservation_time'><br>")
        self.wfile.write(b"Number of Guests: <input type='text' name='number_of_guests'><br>")
        self.wfile.write(b"Special Requests: <input type='text' name='special_requests'><br>")
        self.wfile.write(b"<input type='submit' value='Add Reservation'>")
        self.wfile.write(b"</form>")
        self.wfile.write(b"</center></body></html>")

    def view_all_reservations(self):
        # This method remains the same as before
        records = self.database.getAllReservations()
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
        self.wfile.write(b"<body>")
        self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
        self.wfile.write(b"<hr>")
        self.wfile.write(b"<div> <a href='/'>Home</a>| \
                        <a href='/addReservation'>Add Reservation</a>|\
                        <a href='/viewReservations'>View Reservations</a></div>")
        self.wfile.write(b"<hr><h2>All Reservations</h2>")
        self.wfile.write(b"<table border=1>")
        self.wfile.write(b"<tr><th>Reservation ID</th><th>Customer ID</th><th>Reservation Time
        for row in records:
            self.wfile.write(b"<tr>")
```

```
            self.send_header('Content-type', 'text/html')
            self.end_headers()
            self.wfile.write(b"<html><head><title>Restaurant Portal</title></head>")
            self.wfile.write(b"<body>")
            self.wfile.write(b"<center><h1>Restaurant Portal</h1>")
            self.wfile.write(b"<hr>")
            self.wfile.write(b"<div> <a href='/'>Home</a>| \
                               <a href='/addReservation'>Add Reservation</a>|\
                               <a href='/viewReservations'>View Reservations</a></div>")
            self.wfile.write(b"<hr><h2>All Reservations</h2>")
            self.wfile.write(b"<table border=1>")
            self.wfile.write(b"<tr><th>Reservation ID</th><th>Customer ID</th><th>Reservation Time</th><th>N
            for row in records:
                self.wfile.write(b"<tr>")
                for item in row:
                    self.wfile.write(f"<td>{item}</td>".encode())
                self.wfile.write(b"</tr>")
            self.wfile.write(b"</table></center>")
            self.wfile.write(b"</body></html>")

def run(server_class=HTTPServer, handler_class=RestaurantPortalHandler, port=8000):
    server_address = ('localhost', port)
    httpd = server_class(server_address, handler_class)
    print(f'Starting httpd on port {port}')
    httpd.serve_forever()

if __name__ == "__main__":
    run()
```