Abraham Thomas
Final Project
CIS 344
https://github.com/AbrahamT1969/Final-Project-CIS-344.git

Objective: The objective of this project is to finalize a web platform for restaurant managers to manage reservation schedules and dining experiences. The provided starter code in Python runs on a Python HTTP Server. The task involves connecting to a MySQL server and completing the Database class so that the platform functions smoothly. The final system allows restaurant managers to add, modify, or cancel reservations, as well as update dining preferences for regular customers.

Tools: MySQL Workbench MySQL Server IDLE MySQL Python

Restaurant Portal

Home| Add Reservation| Delete Reservation| View Reservations

All Reservations

Reservation ID	Customer Name	Contact Info	Reservation Time	Number of Guests	Special Requests
1	John Doe	555-1234	2024-05-25 19:00:00	4	Window seat
2	John Doe	555-1234	2024-05-25 19:00:00	4	Window seat
3	John Doe	555-1234	2024-05-25 19:00:00	4	Window seat
4	John Doe	johndoe@example.com	2024-05-22 23:28:00	1	1
5	John Doe	johndoe@example.com	2024-05-22 23:59:00	1	1
6	John Doe	johndoe@example.com	2024-05-23 10:41:00	1	1

Add Reservation

CustomerId: 7
Reservation Time: 05/23/2024 11:16 AM
Number of Guests: 1
Special Requests: 1
Add Reservation

Home
Add Another Reservation
View ReservationsThank You, Reservation Added

MySQL Database Setup

1. Database and Table Creation
   The MySQL script creates a database named restaurants_reservations with three tables: customers, reservations, and diningPreferences.

```sql
CREATE DATABASE restaurants_reservations;
USE restaurants_reservations;

CREATE TABLE customers (
    customerId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    customerName VARCHAR(45) NOT NULL,
    contactInfo VARCHAR(200)
);

CREATE TABLE reservations (
    reservationId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    customerId INT,
    reservationTime DATETIME NOT NULL,
    numberOfGuests INT NOT NULL,
    specialRequests VARCHAR(200),
    FOREIGN KEY (customerId) REFERENCES customers(customerId)
);

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

Inserting Initial Values

Initial values are inserted into the customers, reservations, and diningPreferences tables.

```sql
-- Insert initial values
INSERT INTO customers(customerName, contactInfo) VALUES
('John Doe', 'johndoe@example.com'),
('Jane Smith', 'janesmith@example.com'),
('Alice Johnson', 'alicej@example.com'),
('Bob Brown', 'bobbrown@example.com'),
('Charlie Davis', 'charlied@example.com');

INSERT INTO reservations(reservationTime, numberOfGuests, specialRequests, customerId) VALUES
(1, '2024-05-20 19:00:00', 2, 'Window seat preferred'),
(2, '2024-05-21 20:00:00', 4, 'Allergic to peanuts'),
(3, '2024-05-22 18:30:00', 3, 'Celebrating anniversary'),
(4, '2024-05-23 19:45:00', 5, 'Need high chair for a toddler'),
(5, '2024-05-24 20:30:00', 2, 'Vegetarian meal options');

INSERT INTO diningPreferences(favoriteTable, dietaryRestrictions, customerId) VALUES
(1, 'Table 5', 'None'),
(2, 'Table 9', 'Gluten-free'),
(3, 'Table 20', 'Vegetarian'),
(4, 'Table 15', 'No dairy'),
(5, 'Table 1', 'Vegan');
```

Stored Procedures

Stored procedures are created to encapsulate commonly performed operations.

```sql
-- Procedure to update the specialRequests field in the reservations table
DELIMITER //
CREATE PROCEDURE addSpecialRequest(IN in_reservationId INT, IN in_requests VARCHAR(200))
BEGIN
    UPDATE reservations SET specialRequests = in_requests WHERE reservationId = in_reservationId;
END //
DELIMITER ;

-- Procedure to create a new reservation with customer details
DELIMITER //
CREATE PROCEDURE addReservation(
    IN in_customerName VARCHAR(45),
    IN in_contactInfo VARCHAR(200),
    IN in_reservationTime DATETIME,
    IN in_numberOfGuests INT,
    IN in_specialRequests VARCHAR(200)
)
BEGIN
    DECLARE customerId INT;

    -- Check if customer already exists
    SELECT customerId INTO customerId FROM customers
    WHERE customerName = in_customerName AND contactInfo = in_contactInfo;

    -- If customer does not exist, create a new one
    IF customerId IS NULL THEN
        INSERT INTO customers (customerName, contactInfo) VALUES (in_customerName, in_contactInfo);
        SET customerId = LAST_INSERT_ID();
    END IF;

    -- Add reservation
    INSERT INTO reservations (customerId, reservationTime, numberOfGuests, specialRequests) VALUES
    (customerId, in_reservationTime, in_numberOfGuests, in_specialRequests);
END //
DELIMITER ;
```

Python Database Interaction

The restaurantDatabase.py file is responsible for connecting to the MySQL database and performing CRUD operations.

1. Database Connection
   The RestaurantDatabase class initializes the connection to the MySQL database using the mysql.connector library.

```
##

import mysql.connector
from mysql.connector import Error

class RestaurantDatabase:
    def __init__(self,
                 host="localhost",
                 port="3306",
                 database="restaurants_reservations",
                 user='root',
                 password='monkey1234'):

        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)
```

Add Reservation Method

This method inserts a new reservation into the reservations table.

```
def addReservation(self, customerId, reservationTime, numberOfGuests, specialRequests):
    ''' Method to insert a new reservation into the reservations table '''
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        # Add reservation
        query = "INSERT INTO reservations (customerId, reservationTime, numberOfGuests, specialRequests) VALUES (%s, %s, %s, %s)"
        self.cursor.execute(query, (customerId, reservationTime, numberOfGuests, specialRequests))
        self.connection.commit()
        print("Reservation added successfully")
        return
```

Get All Reservations Method

This method retrieves all reservations from the reservations table.

```
def getAllReservations(self):
    ''' Method to get all reservations from the reservations table '''
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = """
        SELECT r.reservationId, c.customerName, c.contactInfo, r.reservationTime, r.numberOfGuests, r.specialRequests
        FROM reservations r
        JOIN customers c ON r.customerId = c.customerId
        """
        self.cursor.execute(query)
        records = self.cursor.fetchall()
        return records
```

Delete Reservation Method

This method deletes a reservation from the reservations table based on the reservation ID.

```
def deleteReservation(self, reservation_id):
    ''' Method to delete a reservation from the reservations table '''
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "DELETE FROM reservations WHERE reservationId = %s"
        self.cursor.execute(query, (reservation_id,))
        self.connection.commit()
        print("Reservation deleted successfully")
```

HTTP Server Handling

The restaurantServer.py file manages HTTP requests and serves HTML pages.

1. HTTP Server Setup
   The RestaurantPortalHandler class handles GET and POST requests.

```python
from http.server import HTTPServer, BaseHTTPRequestHandler
from restaurantDatabase import RestaurantDatabase
from urllib.parse import parse_qs

class RestaurantPortalHandler(BaseHTTPRequestHandler):

    def __init__(self, *args, **kwargs):
        self.database = RestaurantDatabase()
        super().__init__(*args, **kwargs)
```

Handling POST Requests

The do_POST method handles form submissions for adding reservations.

```python
File  Edit  Format  Run  Options  Window  Help
    def do_POST(self):
        try:
            if self.path == '/addReservation':
                content_length = int(self.headers['Content-Length'])
                post_data = self.rfile.read(content_length)
                form = parse_qs(post_data.decode('utf-8'))


                try:

                    customerId = int(form["customerId"][0])
                    reservationTime = form["reservationTime"][0]
                    numberOfGuests = int(form["numberOfGuests"][0])
                    specialRequests = form.get("specialRequests", [""])[0]



                    self.database.addReservation(customerId, reservationTime, numberOfGuests, specialRequests)
                    print("Reservation added for customer:", customerId)

                    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"<a href='/'>Home</a><br>")
                    self.wfile.write(b"<a href='/addReservation'>Add Another Reservation</a><br>")
                    self.wfile.write(b"<a href='/viewReservations'>View Reservations</a></center>")
                    self.wfile.write(b"Thank You, Reservation Added")
                    self.wfile.write(b"</body></html>")

                    return

                except KeyError as e:

                    print(f"Missing data: {e}")

                    self.send_error(400, 'Missing value')

        except Exception as e:

            print(f"Error: {e}")

            self.send_error(500, 'Server error: {}'.format(str(e)))
```

Handling GET Requests

The do_GET method serves various HTML pages.

```
def do_GET(self):
    try:
        if self.path == '/':
            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='/deleteReservation'>Delete Reservation</a>|\
                               <a href='/viewReservations'>View Reservations</a></div>")
            self.wfile.write(b"<hr><h2>All Reservations</h2>")
            self.wfile.write(b"<table border=2> \
                                <tr><th> Reservation ID </th>\
                                    <th> Customer Name </th>\
                                    <th> Contact Info </th>\
                                    <th> Reservation Time </th>\
                                    <th> Number of Guests </th>\
                                    <th> Special Requests </th></tr>")
            records = self.database.getAllReservations()
            for row in records:
                self.wfile.write(b' <tr> <td>')
                self.wfile.write(str(row[0]).encode())
                self.wfile.write(b'</td><td>')
                self.wfile.write(str(row[1]).encode())
                self.wfile.write(b'</td><td>')
                self.wfile.write(str(row[2]).encode())
                self.wfile.write(b'</td><td>')
                self.wfile.write(str(row[3]).encode())
                self.wfile.write(b'</td><td>')
                self.wfile.write(str(row[4]).encode())
                self.wfile.write(b'</td><td>')
                self.wfile.write(str(row[5]).encode())
                self.wfile.write(b'</td></tr>')

            self.wfile.write(b"</table></center>")
            self.wfile.write(b"</body></html>")
            return

        elif self.path == '/addReservation':
            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><h2>Add Reservation</h2>")
```

```
            self.wfile.write(b"<html><head><title>Add Reservation</title></head>")
            self.wfile.write(b"<body>")
            self.wfile.write(b"<center><h2>Add Reservation</h2>")
            self.wfile.write(b"<form method='post' action='/addReservation'>")
            self.wfile.write(b"<label>CustomerId: </label>")
            self.wfile.write(b"<input type='text' name='customerId' required><br>")
            self.wfile.write(b"<label>Reservation Time: </label>")
            self.wfile.write(b"<input type='datetime-local' name='reservationTime' required><br>")
            self.wfile.write(b"<label>Number of Guests: </label>")
            self.wfile.write(b"<input type='text' name='numberOfGuests' required><br>")
            self.wfile.write(b"<label>Special Requests: </label>")
            self.wfile.write(b"<input type='text' name='specialRequests'><br>")
            self.wfile.write(b"<input type='submit' value='Add Reservation'>")
            self.wfile.write(b"</form>")
            self.wfile.write(b"</center></body></html>")
            return

        elif self.path == '/deleteReservation':
            self.send_response(200)
            self.send_header('Content-type', 'text/html')
            self.end_headers()

            self.wfile.write(b"<html><head><title>Delete Reservation</title></head>")
            self.wfile.write(b"<body>")
            self.wfile.write(b"<center><h2>Delete Reservation</h2>")
            self.wfile.write(b"<form method='post' action='/deleteReservation'>")
            self.wfile.write(b"<label>Reservation ID to delete: </label>")
            self.wfile.write(b"<input type='text' name='reservation_id' required><br>")
            self.wfile.write(b"<input type='submit' value='Delete'>")
            self.wfile.write(b"</form>")
            self.wfile.write(b"</center></body></html>")
            return

        elif self.path == '/viewReservations':
            reservations = self.database.getAllReservations()

            self.send_response(200)
            self.send_header('Content-type', 'text/html')
            self.end_headers()

            self.wfile.write(b"<html><head><title>View Reservations</title></head>")
            self.wfile.write(b"<body>")
            self.wfile.write(b"<center><h1>View Reservations</h1>")
            self.wfile.write(b"<hr>")
            self.wfile.write(b"<div> <a href='/'>Home</a>| \
                             <a href='/addReservation'>Add Reservation</a>|\
                              <a href='/deleteReservation'>Delete 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 Name</th><th>Contact Info</th><th>Reservation Time</th><

            for row in reservations:
                self.wfile.write(b"<tr>")
                for item in row:
                    self.wfile.write(b"<td>")
                    self.wfile.write(str(item).encode())
                    self.wfile.write(b"</td>")
                self.wfile.write(b"</tr>")

            self.wfile.write(b"</table>")
            self.wfile.write(b"</center></body></html>")
            return

    except IOError:
        self.send_error(404, 'File Not Found: %s' % self.path)
```

Running the Server

The run function initializes and starts the HTTP server.

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

run()
```

Steps to Run the Project

Set Up MySQL Database

Execute the provided MySQL script to create the database and tables, and insert initial data. mysql -u root -p < database_setup.sql

Configure Python Environment

Ensure mysql-connector-python is installed. pip install mysql-connector-python

Run the Python Server

Execute the Python HTTP server script. python restaurantServer.py

Access the Web Portal

Open a web browser and navigate to http://localhost:8000 to interact with the Restaurant Portal.

Conclusion

This project integrates MySQL database management with a Python-based HTTP server to create a functional restaurant reservation system. The system allows restaurant managers to manage reservations and customer preferences through a web interface, providing a streamlined approach to handling customer interactions and reservations.