## Report

**Following is the source code of restaurant server python file, which creates the restaurant portal handler to maintain the restaurant database.**

**Explanation of RestaurantServer.py source code**

This code works on a basic HTTP server for a restaurant portal. Here are some observations and suggestions:

* Imports: Ensure that the imports are correct and that all necessary modules are available.
* Handler Initialization: In the \_\_init\_\_ method of RestaurantPortalHandler, you're initializing the RestaurantDatabase object. Make sure that the RestaurantDatabase class is correctly defined in the restaurantDatabase module.
* POST Method Handling: When handling a POST request to add a reservation (/addReservation), you're correctly parsing the form data and calling the addReservation method of the database. Ensure that the addReservation method is defined in your RestaurantDatabase class.
* GET Method Handling: When handling a GET request, you're correctly displaying all reservations (/) and handling requests to view reservations for a specific customer (/findReservations). However, you're trying to access form data (form.getvalue("customer\_id")) without parsing the form. You should parse the form data similar to how you did in the POST method.
* HTML Response: The HTML responses seem fine, but consider using string formatting or HTML templates for better readability and maintainability.
* Error Handling: Ensure robust error handling, especially for file not found errors (IOError). You may want to provide more informative error messages or log them for debugging purposes.
* Security: Validate user input, especially when directly querying the database. This helps prevent SQL injection attacks.
* Comments: Add comments to your code to explain complex logic or document important parts of your implementation.
* Testing: Thoroughly test your server with various types of requests to ensure it behaves as expected.
* Additional Functionality: Implement other methods mentioned in the comments like addSpecialRequest, deleteReservation, etc., as needed for your application.

Once these points are addressed, the server should be in good shape to handle basic reservation management for the restaurant portal.

Following two statements imports the library to connect to the http server, request handler, current directory library.

from http.server import HTTPServer, BaseHTTPRequestHandler

from os import curdir, sep

Following statement imports the restaurantdatabase class.

from restaurantDatabase import RestaurantDatabase

Following statement imports the cgi library.

import cgi

To declare the restaurant portal handler class.

class RestaurantPortalHandler(BaseHTTPRequestHandler):

Following function used to initialize the class

def \_\_init\_\_(self, \*args):

To create the instance of restaurant database class

self.database = RestaurantDatabase()

BaseHTTPRequestHandler.\_\_init\_\_(self, \*args)

Following function handles the POST action of http form

def do\_POST(self):

try:

Following block handles addReservation form submission

if self.path == '/addReservation':

self.send\_response(200)

self.send\_header('Content-type','text/html')

self.end\_headers()

form = cgi.FieldStorage(

fp=self.rfile,

headers=self.headers,

environ={'REQUEST\_METHOD': 'POST'}

)

Following statements gets the value from http form input boxes

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")

# Call the Database Method to add a new reservation

self.database.addReservation(customer\_id, reservation\_time, number\_of\_guests, special\_requests)

print("Reservation added for customer ID:", customer\_id)

To create the HTML page for restaurant portal

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>|\

<a href='/addCustomer'>Add Customer</a></div>\

")

self.wfile.write(b"<hr>")

self.wfile.write(b"<h3>Reservation has been added</h3>")

self.wfile.write(b"<div><a href='/addReservation'>Add Another Reservation</a></div>")

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

except IOError:

self.send\_error(404,'File Not Found: %s' % self.path)

return

Following function handles the GET action of http form

def do\_GET(self):

try:

if self.path == '/':

To fetch the records from the reservation tables using database getAllReservations stored procedure

data=[]

records = self.database.getAllReservations()

print(records)

data=records

To create the HTML page to display reservations table records for restaurant portal

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>|\

<a href='/findReservations'>Search 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 ID </th>\

<th> Reservation Time </th>\

<th> Number of Guests </th>\

<th> Special Requests </th></tr>")

for row in data:

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></tr>')

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

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

return

if self.path =='/addCustomer':

return

if self.path =='/findReservations':

data=[]

customer\_id = int(form.getvalue("customer\_id"))

records = self.database.findReservations(customer\_id)

print(records)

data=records

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>|\

<a href='/findReservations'>Search Reservations</a></div>") self.wfile.write(b"<hr><h2>All Reservations for specific customer</h2>")

self.wfile.write(b"<table border=2> \

<tr><th> Reservation ID </th>\

<th> Customer ID </th>\

<th> Reservation Time </th>\

<th> Number of Guests </th>\

<th> Special Requests </th></tr>")

for row in data:

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></tr>')

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

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

return

## Add any other methods like addSpecialRequests etc

if self.path =='/addSpecialRequest':

return

if self.path =='/deleteReservation':

return

except IOError:

self.send\_error(404,'File Not Found: %s' % self.path)

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()

**Following is the source code of restaurant database python file, which creates the class for restaurant database**

**Overall, the database module provides a solid foundation for managing restaurant reservations in a MySQL database. With a few tweaks and improvements, it will serve the application well.**

Following two statements imports the library to connect to the mysql database and error library.

import mysql.connector

from mysql.connector import Error

To declare the restaurant database class.

class RestaurantDatabase():

Following function initializes the class

def \_\_init\_\_(self,

host="localhost",

port="3306",

database="restaurant\_reservations",

user='root',

password=''):

self.host = host

self.port = port

self.database = database

self.user = user

self.password = password

self.connection = None

self.cursor = None

self.connect()

Following function is used to connect to the restaurant\_reservations mysql database and displays error if not able to connect by any reason.

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")

return

except Error as e:

print("Error while connecting to MySQL", e)

Following function adds record to the reservations table by using customer\_id, reservation\_time, number\_of\_guests, special\_requests parameters as attribute values. First it will open the cursor and then executes the declared query using parameters, then commits the transaction to make the changes permanent in the database.

def addReservation(self, customer\_id, reservation\_time, number\_of\_guests, special\_requests):

''' Method to insert a new reservation into the reservations table '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "INSERT INTO reservations (customer\_id, reservation\_time, number\_of\_guests, special\_requests) VALUES (%s, %s, %s, %s)"

self.cursor.execute(query, (customer\_id, reservation\_time, number\_of\_guests, special\_requests))

self.connection.commit()

print("Reservation added successfully")

return

Following function fetches records from the reservations table. First it will open the cursor and then executes the declared query to retrieve all the records.

def getAllReservations(self):

''' Method to get all reservations from the reservations table '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "SELECT \* FROM reservations"

self.cursor.execute(query)

records = self.cursor.fetchall()

return records

Following function adds record to the customers table by using customer\_name, contact\_info parameters as attribute values. First it will open the cursor and then executes the declared query using parameters, then commits the transaction to make the changes permanent in the database.

def addCustomer(self, customer\_name, contact\_info):

''' Method to add a new customer to the customers table '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "INSERT INTO customers (customer\_name, contact\_info) VALUES (%s, %s)"

self.cursor.execute(query, (customer\_name, contact\_info))

self.connection.commit()

print("Customer added successfully")

return

Following function fetches records from the diningpreferences table. First it will open the cursor and then executes the declared query to retrieve all the records for given customerId.

def getCustomerPreferences(self, customer\_id):

''' Method to retrieve dining preferences for a specific customer '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "SELECT \* FROM diningPreferences WHERE customerId = %s"

self.cursor.execute(query, (customer\_id,))

preferences = self.cursor.fetchall()

return preferences

# Add more methods as needed for restaurant operations

Following function fetches records from the reservations table for the given customer\_id parameter. First it will open the cursor and then executes the declared query to retrieve all the records for given customer\_Id.

def findReservations(self, customer\_id):

''' Method to get all reservations for specific customer\_id '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "SELECT \* FROM reservations WHERE customerId = %s"

self.cursor.execute(query, (customer\_id,))

records = self.cursor.fetchall()

return records

Following function updates the value of specialrequests attribute of reservation table for the given reservation\_id and requests parameter. First it will open the cursor and then executes the declared query to update the table with value of specialRequests for the given customer\_Id. Then finally it commits the transaction to make the changes permanent in the table.

def addSpecialRequest(self, reservation\_id, requests):

''' Method to add special request '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "UPDATE reservations SET specialRequests = %s WHERE reservationId = %s"

self.cursor.execute(query, (requests,reservation\_id,))

self.connection.commit()

print("Special request added successfully")

return

Following function deletes the record from the reservation table for the given reservation\_id parameter. First it will open the cursor and then executes the declared query to delete the record from the table with value of given reservation\_Id. Then finally it commits the transaction to make the changes permanent in the table.

def deleteReservation(self, reservation\_id):

''' Method to delete reservation '''

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")

return

Following function search records from the diningpreferences table for the given customer\_id. First it will open the cursor and then executes the declared query to retrieve all the records for given customer\_Id.

def searchPreferences(self, customer\_id):

''' Method to get all preferences for specific customer\_id '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "SELECT \* FROM DININGPREFERENCES WHERE customerId = %s"

self.cursor.execute(query, (customer\_id,))

records = self.cursor.fetchall()

return records

Following function fetches records from the reservations table. First it will open the cursor and then executes the declared query to retrieve all the records.

def viewAllReservations(self):

''' Method to get all reservations from the reservations table '''

if self.connection.is\_connected():

self.cursor = self.connection.cursor()

query = "SELECT \* FROM reservations"

self.cursor.execute(query)

records = self.cursor.fetchall()

return records