Name: Dheeraj Vemula

Python Assignment :5 -Ticket Booking System

Creating Classes:

Customer class

```python
class Customer:
    def __init__(self, customer_name, email, phone):
        self.customer_name = customer_name
        self.email = email
        self.phone = phone

    def display_customer_details(self):
        print(f"Customer Name : {self.customer_name}")
        print(f"Email : {self.email}")
        print(f"Phone Number : {self.phone}")
```

Event class

```python
class Event(Venue):
    def __init__(self, event_name, event_date, event_time, venue, total_seats, available_seats, ticket_price, event_type):
        self.event_name = event_name
        self.event_date = datetime.strptime(event_date, __format: "%Y-%m-%d").date()
        self.event_time = datetime.strptime(event_time, __format: "%H:%M").time()
        self.venue_name = venue.venue_name
        self.total_seats = total_seats
        self.available_seats = available_seats
        self.ticket_price = ticket_price
        self.event_type = event_type

    def calculate_total_revenue(self):
        return self.ticket_price * (self.total_seats - self.available_seats)

    def getBookedNoOfTickets(self):
        return self.total_seats-self.available_seats

    def book_tickets(self, num_tickets):
        self.available_seats = self.available_seats - num_tickets

    def cancel_booking(self, num_tickets):
        self.available_seats = self.available_seats + num_tickets

    def display_event_details(self):
        print(f"Event name = {self.event_name}")
        print(f"Date of event = {self.event_date}")
        print(f"Time of event = {self.event_time}")
        print(f"Venue name = {self.venue_name}")
        print(f"Available Seats = {self.available_seats}")
```

Venue class

```python
class Venue:
    def __init__(self, venue_name, address):
        self.venue_name = venue_name
        self.address = address

    def display_venue_details(self):
        print(f"Venue Name = {self.venue_name}")
        print(f"Address of venue = {self.address}")
```

Booking class

```python
from bean.Events import Event
import random
from datetime import date


class Booking(Event):

    def __init__(self, event, customer):
        self.booking_id = random.randint( a: 10000,  b: 99999)
        self.customer = customer
        self.event = event
        self.num_tickets = len(customer)
        self.total_cost = 0
        self.booking_date = date.today()


    def calculate_booking_cost(self, num_tickets):
        pass

    def book_tickets(self, num_tickets):
        super().book_tickets(num_tickets)

    def cancel_booking(self, num_tickets):
        super().cancel_booking(num_tickets)

    def getAvailableNoOfTickets(self):
        return self.event.available_seats

    def getEventDetails(self):
        pass
```

Movie class

```python
from bean.Events import Event

class Movie(Event):
    def __init__(self, event, genre, actorName, actressName):
        self.event = event
        self.genre = genre
        self.actorName = actorName
        self.actressName = actressName

    @property
    def getname(self):
        return self.event.event_name

    @getname.setter
    def setname(self, name):
        self.event.event_name = name
```

Booking system service provider:

```python
class IBookingSystemServiceProvider:

    @abstractmethod
    def calculate_booking_cost(self, num_tickets):
        pass

    @abstractmethod
    def book_tickets(self, num_tickets):
        pass

    @abstractmethod
    def cancel_booking(self, booking_id):
        pass

    @abstractmethod
    def get_booking_details(self, booking_id):
        pass
```

Booking system service provider implementation:

```python
class BookingSystemServiceProviderImpl(IBookingSystemServiceProvider):
    def __init__(self, dbUtil):
        self.dbUtil = dbUtil

    def calculate_booking_cost(self, num_tickets):
        pass

    def book_tickets(self, num_tickets):
        print("Please enter customer details so we can assure booking : ")
        customer_name = input("First please enter your name : ")
        customer_email = input("Please enter you email : ")
        customer_phone = input("Please enter your phone number : ")
        cursor = self.dbUtil.getDBConnection()
        cursor.execute("insert into customer(customer_name,email,phone_number) values (%s,%s,%s)",
                       (customer_name, customer_email, customer_phone,))
        self.dbUtil.con.commit()
        cursor.execute("select customer_id from customer where customer_name=%s", (customer_name,))
        customer_row = cursor.fetchone()
        if customer_row:
            customer_id = customer_row[0]
        print("Please select one events from the events listed below : ")
        cursor.execute("select event_name from events")
        events = cursor.fetchall()
        for event in events:
            print(event[0])
        name_of_event = input("Enter your event here : ")
        cursor.execute("select event_id,ticket_price from events where event_name = %s", (name_of_event,))
        rows = cursor.fetchone()
        if rows:
            event_id, price = rows
```

```python
        total_cost = price * num_tickets
        today = date.today()
        query = "insert into bookings (customer_id, event_id, num_tickets, total_cost, booking_date) values (%s,%s,%s,%s,%s)"
        cursor.execute(query, (customer_id, event_id, num_tickets, total_cost, today))
        self.dbUtil.con.commit()
        cursor.execute("select booking_id from bookings where customer_id = %s", (customer_id,))
        booking_id = cursor.fetchone()
        if booking_id:
            b_id = booking_id[0]
        print("Congratulations. Your booking is confirmed. Your booking id is ", b_id)

    def cancel_booking(self, booking_id):
        cursor = self.dbUtil.getDBConnection()
        query = "delete from bookings where booking_id = %s"
        cursor.execute(query, (booking_id,))
        self.dbUtil.con.commit()
        print("Your booking is cancelled successfully.")
```

Event Service provider:

```python
class IEventServiceProvider(ABC):

    @abstractmethod
    def create_event(self):
        pass

    @abstractmethod
    def getEventDetails(self):
        pass

    @abstractmethod
    def getAvailableNoOfTickets(self):
        pass
```

Event Service provider implementation:

```python
def create_event(self):
    event_name = input("Enter event name : ")
    date = input("Enter event Date : ")
    event_date = datetime.datetime.strptime(date, _format: "%Y-%m-%d").date()
    time = input("Enter event time in format HH:MM:SS : ")
    event_time = datetime.datetime.strptime(time, _format: "%H:%M:%S").time()
    venue = input("Enter venue name : ")
    total_seats = int(input("Enter total seats : "))
    available_seats = int(input("Enter available seats : "))
    ticket_price = float(input("Enter ticket price : "))
    event_type = input("Enter event type : ")
    cursor = self.dbUtil.getDBConnection()
    query = "insert into venue (venue_name) values (%s)"
    cursor.execute(query, (venue, ))
    self.dbUtil.con.commit()
    cursor.execute("select venue_id from venue where venue_name=%s", (venue, ))
    venue_id = cursor.fetchone()
    createEvent = ("insert into events (event_name,event_date,event_time,venue_id,total_seats,available_seats,ticket_price,event_type) values
    cursor.execute(createEvent, (event_name, event_date, event_time, venue_id[0], total_seats, available_seats, ticket_price, event_type, ))
    self.dbUtil.con.commit()
    cursor.execute("select * from events")
    rows = cursor.fetchall()
    for row in rows:
        print(row[-1])
    print(f"Event id for this event is {rows[-1][0]}")
```

```
        print("Event created successfully you can see the details above.")

    def getEventDetails(self):
        cursor = self.dbUtil.getDBConnection()
        cursor.execute("select * from events")
        rows = cursor.fetchall()
        for row in rows:
            print(row)

    def getAvailableNoOfTickets(self):
        cursor = self.dbUtil.getDBConnection()
        query = "select event_name from events"
        cursor.execute(query)
        event_names = cursor.fetchall()
        print("Please select one events from below : ")
        for event in event_names:
            print(event)
        take_event = input("Please type your event name here correctly : ")
        query = "select available_seats from events where event_name=%s"
        cursor.execute(query, (take_event, ))
        seats = cursor.fetchall()
        print(seats)
```

Establishing the connection to database:

```
from mysql import connector
import mysql


class DBUtil:
    def __init__(self):
        self.con = mysql.connector.connect(
            host="localhost",
            port="3306",
            user="root",
            password="Kevink25*",
            database="ticketbookingsystem"
        )

    def getDBConnection(self):
        return self.con.cursor()
```

Ticket Booking System:

```python
from bean.BookingSystemServiceProviderImpl import BookingSystemServiceProviderImpl
from bean.EventServiceProviderImpl import EventServiceProviderImpl
from DBUtil import DBUtil


class TicketBookingSystem(EventServiceProviderImpl, BookingSystemServiceProviderImpl):
    def __init__(self, dbUtil):
        super().__init__(dbUtil)

    def main(self):
        while True:
            print("Select one options from the options given below : ")
            print("1. Type create_event to Create a new event.")
            print("2. Type book_tickets to book tickets.")
            print("3. Type cancel_tickets Cancel Tickets.")
            print("4. Type get_available_seats to Know how many seats are Available.")
            print("5. Type get_event_details to see every event and it's details.")
            print("6. Type exit to Exit from the application.")
            choice = input("Enter your choice here : ")
            match choice:
                case "create_event":
                    self.create_event()
                    print()
                case "get_event_details":
                    self.getEventDetails()
                    print()
                case "get_available_seats":
                    self.getAvailableNoOfTickets()
                    print()
                case "book_tickets":
                    num_tickets = int(input("Please enter the number of tickets you want to book : "))
                    self.book_tickets(num_tickets)
                    print()
                case "cancel_tickets":
                    booking_id = int(input("Please enter your booking id here : "))
                    self.cancel_booking(booking_id)
                    print()
                case "exit":
                    break
                case _:
                    print("Invalid input! Please Try Again.")
                    print("We're heading you to main menu.")
            print("Thanks for visiting our platform to book tickets. Hope to see you soon.")


dbutil = DBUtil()
events = TicketBookingSystem(dbutil)
events.main()
```

Ticket Booking System operations:

Create Event

```
def create_event(self):
    event_name = input("Enter event name : ")
    date = input("Enter event Date : ")
    event_date = datetime.datetime.strptime(date, __format: "%Y-%m-%d").date()
    time = input("Enter event time in format HH:MM:SS : ")
    event_time = datetime.datetime.strptime(time, __format: "%H:%M:%S").time()
    venue = input("Enter venue name : ")
    total_seats = int(input("Enter total seats : "))
    available_seats = int(input("Enter available seats : "))
    ticket_price = float(input("Enter ticket price : "))
    event_type = input("Enter event type : ")
    cursor = self.dbUtil.getDBConnection()
    query = "insert into venue (venue_name) values (%s)"
    cursor.execute(query, (venue, ))
    self.dbUtil.con.commit()
    cursor.execute("select venue_id from venue where venue_name=%s", (venue, ))
    venue_id = cursor.fetchone()
    createEvent = ("insert into events (event_name,event_date,event_time,venue_id,total_seats,available_seats,ticket_price,event_type) values
    cursor.execute(createEvent, (event_name, event_date, event_time, venue_id[0], total_seats, available_seats, ticket_price, event_type, ))
    self.dbUtil.con.commit()
    cursor.execute("select * from events")
    rows = cursor.fetchall()
    for row in rows:
        print(row[-1])
    print(f"Event id for this event is {rows[-1][0]}")
    print("Event created successfully you can see the details above.")
```

Book tickets

```
def book_tickets(self, num_tickets):
    print("Please enter customer details so we can assure booking : ")
    customer_name = input("First please enter your name : ")
    customer_email = input("Please enter you email : ")
    customer_phone = input("Please enter your phone number : ")
    cursor = self.dbUtil.getDBConnection()
    cursor.execute("insert into customer(customer_name,email,phone_number) values (%s,%s,%s)",
                   (customer_name, customer_email, customer_phone,))
    self.dbUtil.con.commit()
    cursor.execute("select customer_id from customer where customer_name=%s", (customer_name,))
    customer_row = cursor.fetchone()
    if customer_row:
        customer_id = customer_row[0]
    print("Please select one events from the events listed below : ")
    cursor.execute("select event_name from events")
    events = cursor.fetchall()
    for event in events:
        print(event[0])
    name_of_event = input("Enter your event here : ")
    cursor.execute("select event_id,ticket_price from events where event_name = %s", (name_of_event,))
    rows = cursor.fetchone()
    if rows:
        event_id, price = rows
    total_cost = price * num_tickets
```

Cancel Tickets

```
def cancel_booking(self, booking_id):
    cursor = self.dbUtil.getDBConnection()
    query = "delete from bookings where booking_id = %s"
    cursor.execute(query, (booking_id,))
    self.dbUtil.con.commit()
    print("Your booking is cancelled successfully.")
```

Get Available seats

```
def getAvailableNoOfTickets(self):
    cursor = self.dbUtil.getDBConnection()
    query = "select event_name from events"
    cursor.execute(query)
    event_names = cursor.fetchall()
    print("Please select one events from below : ")
    for event in event_names:
        print(event)
    take_event = input("Please type your event name here correctly : ")
    query = "select available_seats from events where event_name=%s"
    cursor.execute(query, (take_event, ))
    seats = cursor.fetchall()
    print(seats)
```

Get event Details

```
def getEventDetails(self):
    cursor = self.dbUtil.getDBConnection()
    cursor.execute("select * from events")
    rows = cursor.fetchall()
    for row in rows:
        print(row)
```