ASSIGMENT 5

TICKET BOOKING SYSTEM

AKILESH K

Task 1

```
def book_tickets(available_tickets, no_of_booking_tickets):
    if available_tickets >= no_of_booking_tickets:
        remaining_tickets = available_tickets - no_of_booking_tickets
        print(f"Tickets booked successfully! Remaining tickets: {remaining_tickets}")
    else:
        print("Ticket unavailable. Not enough tickets to book.")


if __name__ == "__main__":
    try:
        available_tickets = int(input("Enter the number of available tickets: "))
        no_of_booking_tickets = int(input("Enter the number of tickets to book: "))
        if available_tickets < 0 or no_of_booking_tickets < 0:
            print("Invalid input. Please enter non-negative values.")
        else:
            book_tickets(available_tickets, no_of_booking_tickets)
    except ValueError:
        print("Invalid input. Please enter valid numbers.")
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../.
sign\courier_management\test.py'
Enter the number of available tickets: 50
Enter the number of tickets to book: 35
Tickets booked successfully! Remaining tickets: 15
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python cl
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\L
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/
sign\courier_management\test.py'
Enter the number of available tickets: 50
Enter the number of tickets to book: 60
Ticket unavailable. Not enough tickets to book.
```

Task 2

```
def calculate_ticket_cost(ticket_type, no_of_tickets):


    silver_price = 50

    gold_price = 100

    diamond_price = 150


    # Validate ticket type

    if ticket_type.lower() == "silver":

        base_price = silver_price

    elif ticket_type.lower() == "gold":

        base_price = gold_price

    elif ticket_type.lower() == "diamond":

        base_price = diamond_price

    else:

        print("Invalid ticket type. Please choose from Silver, Gold, or Diamond.")

        return None


    total_cost = base_price * no_of_tickets

    return total_cost


if __name__ == "__main__":

    while True:

        try:
```

```
        ticket_type = input("Enter the ticket type (Silver/Gold/Diamond) or type 'Exit' to stop: ")



        if ticket_type.lower() == "exit":
            break



        no_of_tickets = int(input("Enter the number of tickets needed: "))


        # Validate number of tickets
        if no_of_tickets <= 0:
            print("Invalid number of tickets. Please enter a positive number.")
        else:


            total_cost = calculate_ticket_cost(ticket_type, no_of_tickets)
            if total_cost is not None:
                print(f"Total cost of {no_of_tickets} {ticket_type} ticket(s): ${total_cost}")
    except ValueError:
        print("Invalid input. Please enter valid numbers.")
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Loca
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/..
sign\courier_management\test.py'
Enter the ticket type (Silver/Gold/Diamond): Gold
Enter the number of tickets needed: 5
Total cost of 5 Gold ticket(s): $500
```

Task 4

```
class Event:
    def __init__(self, event_name, event_date, event_time, venue_name, total_seats, available_seats, ticket_price, event_type):
        self.event_name = event_name
        self.event_date = event_date
```

```
    self.event_time = event_time
    self.venue_name = venue_name
    self.total_seats = total_seats
    self.available_seats = available_seats
    self.ticket_price = ticket_price
    self.event_type = event_type
    self.booked_tickets = 0

def calculate_total_revenue(self):
    return self.booked_tickets * self.ticket_price

def getBookedNoOfTickets(self):
    return self.booked_tickets

def book_tickets(self, num_tickets):
    if num_tickets > self.available_seats:
        print("Not enough available seats.")
    else:
        self.booked_tickets += num_tickets
        self.available_seats -= num_tickets
        print(f"{num_tickets} tickets booked successfully.")

def cancel_booking(self, num_tickets):
    if num_tickets > self.booked_tickets:
        print("Invalid number of tickets to cancel.")
    else:
        self.booked_tickets -= num_tickets
        self.available_seats += num_tickets
        print(f"{num_tickets} tickets canceled successfully.")
```

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




class Customer:
    def __init__(self, customer_name, email, phone_number):
        self.customer_name = customer_name
        self.email = email
        self.phone_number = phone_number

class Booking:
    def __init__(self, event):
        self.event = event

    def calculate_booking_cost(self, num_tickets):
        return num_tickets * self.event.ticket_price

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

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

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

    def getEventDetails(self):
        return self.event.display_event_details()
```

Task 5

```
class Movie(Event):

    def __init__(self, event_name, event_date, event_time, venue_name, total_seats, available_seats, ticket_price, event_type, genre, actor_name, actress_name):

        super().__init__(event_name, event_date, event_time, venue_name, total_seats, available_seats, ticket_price, event_type)

        self.genre = genre

        self.actor_name = actor_name

        self.actress_name = actress_name


class Concert(Event):

    def __init__(self, event_name, event_date, event_time, venue_name, total_seats, available_seats, ticket_price, event_type, artist, concert_type):

        super().__init__(event_name, event_date, event_time, venue_name, total_seats, available_seats, ticket_price, event_type)

        self.artist = artist

        self.concert_type = concert_type


class Sports(Event):

    def __init__(self, event_name, event_date, event_time, venue_name, total_seats, available_seats, ticket_price, event_type, sport_name, teams_name):

        super().__init__(event_name, event_date, event_time, venue_name, total_seats, available_seats, ticket_price, event_type)

        self.sport_name = sport_name

        self.teams_name = teams_name


class TicketBookingSystem:

    def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue_name):

        event_date = datetime.strptime(date, "%Y-%m-%d")

        return Event(event_name, event_date, time, venue_name, total_seats, total_seats, ticket_price, event_type)
```

```
    def display_event_details(self, event):
        event.display_event_details()

    def book_tickets(self, event, num_tickets):
        return event.book_tickets(num_tickets)

    def cancel_tickets(self, event, num_tickets):
        event.cancel_booking(num_tickets)
```

Task 6

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

    @abstractmethod
    def getBookedNoOfTickets(self):
        pass

    @abstractmethod
    def book_tickets(self, num_tickets):
        pass

    @abstractmethod
    def cancel_booking(self, num_tickets):
        pass

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

Task 9

```
class BookingSystemException(Exception):
    pass

class EventNotFoundException(BookingSystemException):
    def __init__(self, event_name):
        self.event_name = event_name
        super().__init__(f"Event '{event_name}' not found.")

class InvalidBookingIDException(BookingSystemException):
    def __init__(self, booking_id):
        self.booking_id = booking_id
        super().__init__(f"Invalid Booking ID: {booking_id}")

class NullPointerException(BookingSystemException):
    def __init__(self, message="Null pointer exception."):
        super().__init__(message)
```

Task 11

```
def main_menu():
    while True:
        print("\nMain Menu:")
        print("1. Display Event Listings")
        print("2. Get Available Seats")
        print("3. Get Booking Details")
        print("4. Calculate Total Amount")
        print("5. Book Tickets")
        print("6. Cancel Booking")
        print("7. Add New Event")
        print("8. Exit")

        choice = input("Enter your choice (1-8): ")
```

```
        if choice == '1':
            display_event_listings()
        elif choice == '2':
            get_available_seats()
        elif choice == '3':
            get_booking_details()
        elif choice == '4':
            cal_total_amount()
        elif choice == '5':
            book_tickets()
        elif choice == '6':
            cancel_booking()
        elif choice == '7':
            add_new_event()
        elif choice == '8':
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 8.")

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

DATABASE CONNECT

```
from datetime import datetime
import mysql.connector
from mysql.connector import Error
# Database connection
```

```
def create_connection():
    try:
        connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="root",
            port='3306',
            database="ticket_booking_system"
        )
        return connection
    except Error as e:
        print(f"Error connecting to the database: {e}")
        return None
```

getEventDetails()

```
def display_event_listings():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM event")
            cus = cursor.fetchall()

            print("events:")
            for clist in cus:
                print(clist)

        except Error as e:
            print(f"Error retrieving  listings: {e}")
        finally:
```

```
        connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals>  c:; cd 'c:\Users\Akilesh K\OneDrive\Documents
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Akilesh K\.vscode\exter
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '50418' '--' 'c:\Users\Akilesh K\OneDrive\Documents\
sign\ticketbooking\ticket.py'
events:
(11, 'Event1', datetime.date(2023, 1, 1), datetime.timedelta(seconds=43200), 1, 200, 150, Decimal('50.00'), 'Movie', 10)
(21, 'Event2', datetime.date(2023, 2, 15), datetime.timedelta(seconds=66600), 2, 300, 250, Decimal('75.00'), 'Sports', 22)
(31, 'Event3', datetime.date(2023, 3, 20), datetime.timedelta(seconds=72000), 3, 150, 100, Decimal('100.00'), 'Concert', 33)
(41, 'Event4', datetime.date(2023, 4, 10), datetime.timedelta(seconds=56700), 4, 250, 200, Decimal('60.00'), 'Movie', 44)
(51, 'Event5', datetime.date(2023, 5, 5), datetime.timedelta(seconds=50400), 5, 180, 120, Decimal('90.00'), 'Concert', 55)
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals>
```

getAvailableNoOfTickets()

```
def get_available_seats():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            eventid=int(input("enter event ID:"))
            select_query=("SELECT available_seats FROM event where event_id= %s")
            cursor.execute(select_query,(eventid,))
            cus = cursor.fetchall()

            print("available seats:")

            print(cus)

        except Error as e:
            print(f"Error retrieving listings: {e}")
        finally:
```

```
        connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\py
hallenge\petpals'; & 'C:\Users\Akilesh K\Ap
-2023.22.1\pythonFiles\lib\python\debugpy\a
sign\ticketbooking\ticket.py'
enter event ID:11
available seats:
[(150,)]
```

get_booking_details()

```
def get_booking_details():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            bookingid=int(input("enter bookingID:"))
            select_query=("SELECT * FROM booking where booking_id= %s")
            cursor.execute(select_query,(bookingid,))
            cus = cursor.fetchall()

            print("booking details:")

            print(cus)

        except Error as e:
            print(f"Error retrieving listings: {e}")
        finally:
            connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding chal
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\P
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\l
sign\ticketbooking\ticket.py'
enter bookingID:10
booking details:
[(10, 111, 11, 2, Decimal('100.00'), datetime.date(2023, 1, 1))]
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding chal
```

calculate_booking_cost

```
def cal_total_amount():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            eventid=int(input("enter event ID:"))
            nop=int(input("enter the number of tickets:"))
            select_query=("SELECT ticket_price FROM event where event_id= %s")
            cursor.execute(select_query,(eventid,))
            cus = cursor.fetchone()
            total=reduce(operator.__mul__, cus,nop)
            #total=cus*nop
            print("total amount:")
            print(total)

        except Error as e:
            print(f"Error retrieving listings: {e}")
        finally:
            connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../.
sign\ticketbooking\ticket.py'
enter event ID:11
enter the number of tickets:3
total amount:
150.00
```

book_tickets

```
w = 64


def generate_o_number():
    global w
    w += 1
    return w


def book_tickets():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            customerid=input("enter customerid:")
            eventid=int(input("enter event ID:"))
            bookingid=generate_o_number()
            nop=int(input("enter the number of tickets:"))
            date = datetime.now().strftime("%Y-%m-%d")
            select_query=("SELECT ticket_price FROM event where event_id= %s")
            cursor.execute(select_query,(eventid,))
            cus = cursor.fetchone()
            total=reduce(operator.__mul__, cus,nop)
            #total=cus*nop
```

```
        cursor.execute("INSERT INTO
booking(booking_id,customer_id,event_id,num_tickets,total_cost,booking_date) VALUES (%s, %s,
%s, %s,%s)",

                (bookingid,customerid,eventid,nop,total,date))

        connection.commit()



    except Error as e:

        print(f"Error retrieving listings: {e}")

    finally:

        connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding ch
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy
sign\ticketbooking\ticket.py'
enter customerid:555
enter event ID:11
enter the number of tickets:4
```

```
mysql> select * from booking;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|         10 |         111 |       11 |           2 |     100.00 | 2023-01-01   |
|         22 |         222 |       21 |           3 |     225.00 | 2023-02-15   |
|         33 |         333 |       31 |           1 |     100.00 | 2023-03-20   |
|         44 |         444 |       41 |           4 |     240.00 | 2023-04-10   |
|         55 |         555 |       51 |           2 |     180.00 | 2023-05-05   |
|         65 |         555 |       11 |           4 |     200.00 | 2023-12-20   |
+------------+-------------+----------+-------------+------------+--------------+
6 rows in set (0.00 sec)
```

cancel_booking

```
def cancel_booking():

    connection = create_connection()

    if connection:

        try:

            cursor = connection.cursor()

            bookingid=int(input("enter booking id:"))
```

```
        delete_query = ("DELETE FROM booking WHERE booking_id = %s")
        cursor.execute(delete_query, ( bookingid,))

        connection.commit()
        print(f"deleted value in the database.")

    except Error as e:
        print(f"Error deleting value in the table: {e}")

    finally:
        connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Loca
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../
sign\ticketbooking\ticket.py'
enter booking id:65
deleted value in the database.
```

```
mysql> select * from booking;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|         10 |         111 |       11 |           2 |     100.00 | 2023-01-01   |
|         22 |         222 |       21 |           3 |     225.00 | 2023-02-15   |
|         33 |         333 |       31 |           1 |     100.00 | 2023-03-20   |
|         44 |         444 |       41 |           4 |     240.00 | 2023-04-10   |
|         55 |         555 |       51 |           2 |     180.00 | 2023-05-05   |
+------------+-------------+----------+-------------+------------+--------------+
5 rows in set (0.00 sec)
```

create_event

```
ev = 60


def generate_eve():
    global ev
    ev += 1
```

```
    return ev


def add_new_event():
    connection = create_connection()
    if connection:
        try:
            eventid=generate_eve()
            event_name = input("Enter event name: ")
            date = (input("Enter date: "))
            venueid = input("Enter venue id: ")
            totalseats = input("Enter total seat: ")
            available_seats = input("Enter available seats: ")
            ticket_price = input("Enter ticket price: ")
            event_time = input("Enter event time: ")
            seat_type = input("Enter event type: ")
            bookingid=65


            cursor = connection.cursor()
            cursor.execute("INSERT INTO event(event_id, event_name, event_date, event_time, venue_id,
total_seats, available_seats, ticket_price, event_type, booking_id) VALUES (%s, %s, %s, %s,%s, %s,
%s, %s,%s,%s)",
                           (eventid,event_name,date,event_time,venueid,totalseats,available_seats,ticket_price
,seat_type,bookingid))
            connection.commit()


            print("event details recorded successfully!")


        except (Error, ValueError) as e:
            print(f"Error recording courier: {e}")
        finally:
            connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launche
sign\ticketbooking\ticket.py'
Enter event name: event5
Enter date: 2024-02-22
Enter venue id: 5
Enter total seat: 200
Enter available seats: 150
Enter ticket price: 300
Enter event time: 16:20
Enter event type: movie
event details recorded successfully!
```

```
mysql> select * from event;
+---------+-----------+-----------+-----------+---------+------------+----------------+-------------+-----------+-----------+
| event_id | event_name | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+---------+-----------+-----------+-----------+---------+------------+----------------+-------------+-----------+-----------+
|      11 | Event1    | 2023-01-01 | 12:00:00  |       1 |        200 |            150 |       50.00 | Movie     |        10 |
|      21 | Event2    | 2023-02-15 | 18:30:00  |       2 |        300 |            250 |       75.00 | Sports    |        22 |
|      31 | Event3    | 2023-03-20 | 20:00:00  |       3 |        150 |            100 |      100.00 | Concert   |        33 |
|      41 | Event4    | 2023-04-10 | 15:45:00  |       4 |        250 |            200 |       60.00 | Movie     |        44 |
|      51 | Event5    | 2023-05-05 | 14:00:00  |       5 |        180 |            120 |       90.00 | Concert   |        55 |
|      61 | event5    | 2024-02-22 | 16:20:00  |       5 |        200 |            150 |      300.00 | Movie     |        65 |
+---------+-----------+-----------+-----------+---------+------------+----------------+-------------+-----------+-----------+
6 rows in set (0.01 sec)
```