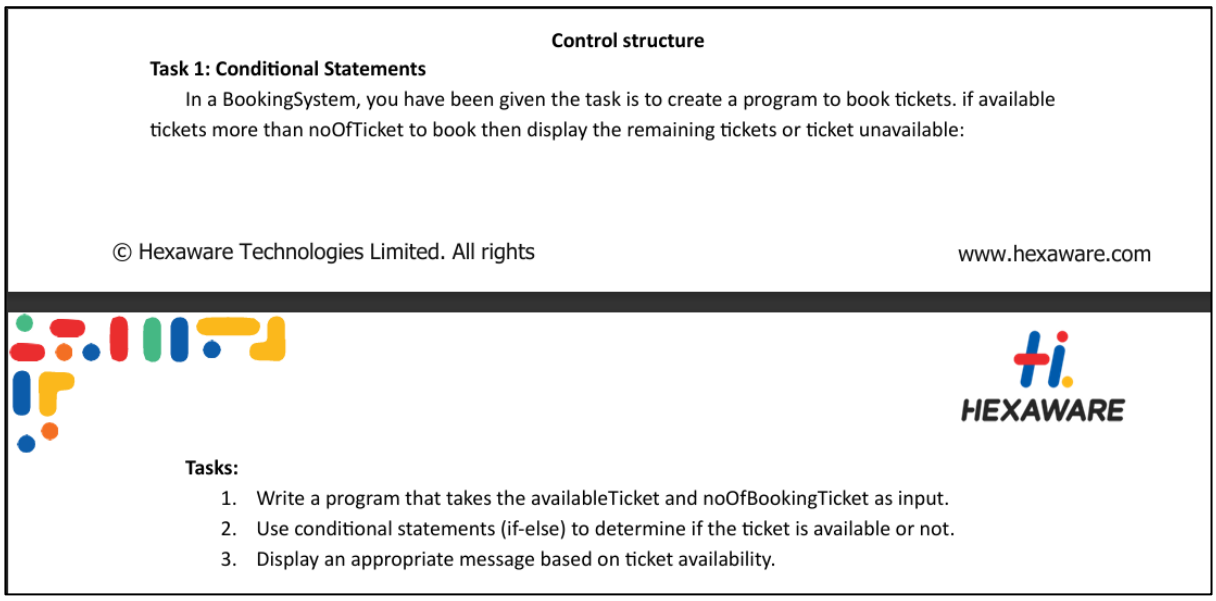
Assignment – Python

TASK 1:



Control structure

Task 1: Conditional Statements

In a BookingSystem, you have been given the task is to create a program to book tickets. if available tickets more than noOfTicket to book then display the remaining tickets or ticket unavailable:

HEXAWARE

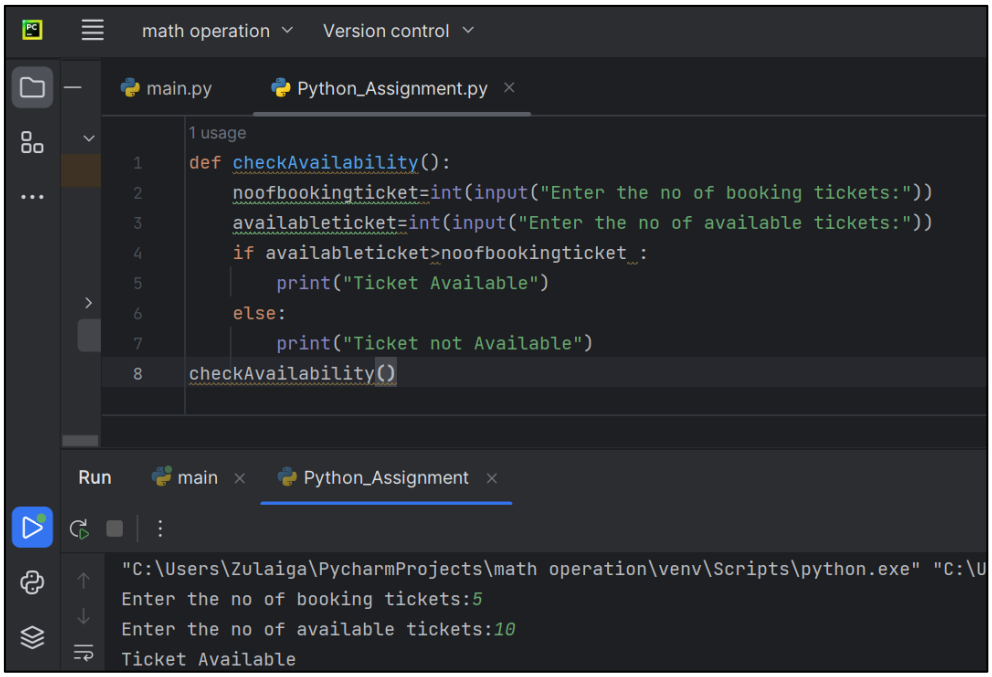
Tasks:

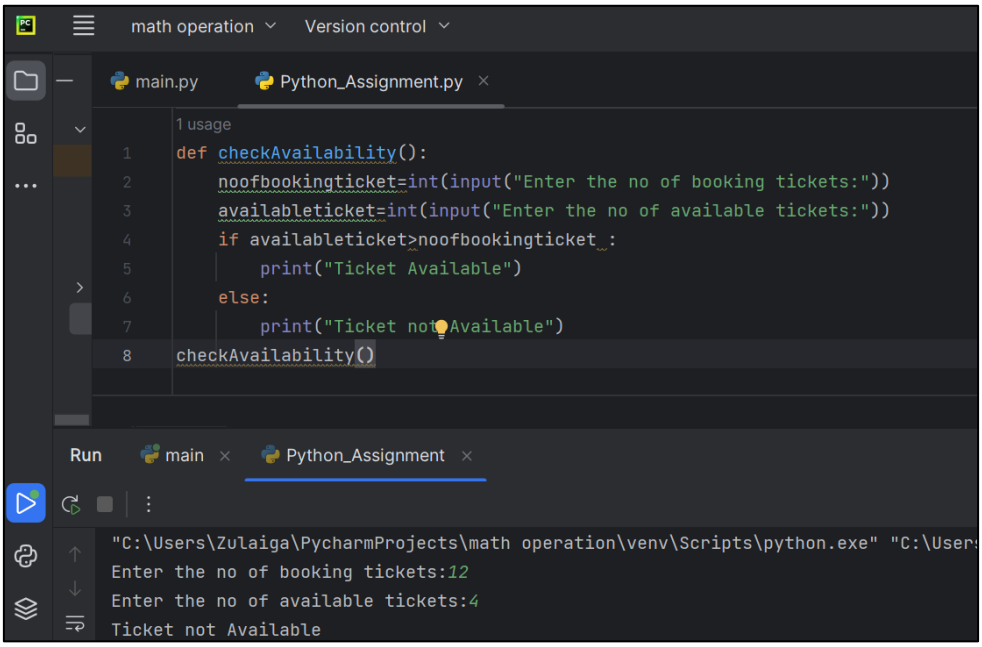
1. Write a program that takes the availableTicket and noOfBookingTicket as input.
2. Use conditional statements (if-else) to determine if the ticket is available or not.
3. Display an appropriate message based on ticket availability.

CODE :

```
def check_ticket_availability():
    availabletickets = int(input("Enter available tickets:"))
    bookingtickets = int(input("Enter no of booking ticket:"))
    if availabletickets > bookingtickets:
        print("Ticket is available")
    else:
        print("Ticket is not available")
check_ticket_availability()
if check_ticket_availability() == True :
    print("Checked")
else :
    print("Not Checked")
```

OUTPUT :


math operation
Version control
main.py
Python_Assignment.py
1 usage
def checkAvailability():
noofbookingticket=int(input("Enter the no of booking tickets:"))
availableticket=int(input("Enter the no of available tickets:"))
if availableticket>noofbookingticket :
print("Ticket Available")
else:
print("Ticket not Available")
checkAvailability()
Run
main
Python_Assignment
"C:\Users\Zulaiga\PycharmProjects\math operation\venv\Scripts\python.exe" "C:\U
Enter the no of booking tickets:5
Enter the no of available tickets:10
Ticket Available


math operation
Version control
main.py
Python_Assignment.py
1 usage
def checkAvailability():
noofbookingticket=int(input("Enter the no of booking tickets:"))
availableticket=int(input("Enter the no of available tickets:"))
if availableticket>noofbookingticket :
print("Ticket Available")
else:
print("Ticket not Available")
checkAvailability()
Run
main
Python_Assignment
"C:\Users\Zulaiga\PycharmProjects\math operation\venv\Scripts\python.exe" "C:\Users
Enter the no of booking tickets:12
Enter the no of available tickets:4
Ticket not Available

TASK 2 & 3:

Task 2: Nested Conditional Statements

Create a program that simulates a Ticket booking and calculating cost of tickets. Display tickets options such as "Silver", "Gold", "Dimond". Based on ticket category fix the base ticket price and get the user input for ticket type and no of tickets need and calculate the total cost of tickets booked.

Task 3: Looping

From the above task book the tickets for repeatedly until user type "Exit"

CODE :

```
def select_ticket_options():
   while True:
    print("1.Book Tickets")
    print("2.Exit")
    option=int(input("Enter an option 1/2:"))
    if option == 2:
       print("Thankyou,Visit us again!")
       break
    if option == 1:
     print("Displaying ticket options:")
     choice = int(input(("Enter 1 for Silver,Enter 2 for Gold,Enter 3 for Diamond:")))
     if(choice == 1):
       print("You've selected ticket under Silver category")
       print("Ticket Price:50")
       no_of_ticket=int(input("Enter no of ticket:"))
       Total_price=50*no_of_ticket;
       print("Your Total Price:",Total_price)
       print("Thank you for Booking!")
     elif(choice == 2):
       print("You've selected ticket under Gold category")
       print("Ticket Price:100")
       no_of_ticket = int(input("Enter no of ticket:"))
       Total_price = 100 * no_of_ticket;
       print("Your Total Price:", Total_price)
       print("Thank you for Booking!")
     elif(choice == 3):
       print("You've selected ticket under Diamond category")
       print("Ticket Price:200")
       no_of_ticket = int(input("Enter no of ticket:"))
       Total_price = 200 * no_of_ticket;
       print("Your Total Price:", Total_price)
       print("Thank you for Booking!")
select_ticket_options()
```

OUTPUT :

```
"C:\Users\Zulaiga\PycharmProjects\math operation\venv\Scripts\python
1.Book Tickets
2.Exit
Enter an option 1/2:1
Displaying ticket options:
Enter 1 for Silver,Enter 2 for Gold,Enter 3 for Diamond:2
You've selected ticket under Gold category
Ticket Price:100
Enter no of ticket:2
Your Total Price: 200
Thank you for Booking!
1.Book Tickets
2.Exit
Enter an option 1/2:2
Thankyou,Visit us again!
```

TASK 4 :

Task 4: Class & Object

Create a Following classes with the following attributes and methods:

1. **Event** Class:
   - **Attributes:**
     - event_name,
     - event_date DATE,
     - event_time TIME,
     - venue_name,
     - total_seats,
     - available_seats,
     - ticket_price DECIMAL,
     - event_type ENUM('Movie', 'Sports', 'Concert')
   - **Methods and Constuctors:**
     - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter, (print all information of attribute) methods for the attributes.
     - **calculate_total_revenue()**: Calculate and return the total revenue based on the number of tickets sold.
     - **getBookedNoOfTickets()**: return the total booked tickets
     - **book_tickets(num_tickets)**: Book a specified number of tickets for an event. Initially available seats are equal to the total seats when tickets are booked available seats number should be reduced.
     - **cancel_booking(num_tickets)**: Cancel the booking and update the available seats.
     - **display_event_details():** Display event details, including event name, date time seat availability.

2. **Venue** Class
   - **Attributes**:
     - venue_name,
     - address
   - **Methods and Constuctors:**
     - **display_venue_details():** Display venue details.

   - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.

3. **Customer** Class
   - **Attributes:**
     - customer_name,
     - email,
     - phone_number,
   - **Methods and Constuctors:**
     - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
     - **display_customer_details()**: Display customer details.

4. **Booking** Class to represent the Tiket booking system. Perform the following operation in main method. Note:- Use Event class object for the following operation.
   - **Methods and Constuctors:**
     - **calculate_booking_cost(num_tickets)**: Calculate and set the total cost of the booking.
     - **book_tickets(num_tickets)**: Book a specified number of tickets for an event.
     - **cancel_booking(num_tickets)**: Cancel the booking and update the available seats.
     - **getAvailableNoOfTickets()**: return the total available tickets
     - **getEventDetails()**: return event details from the event class

CODE :

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

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

    def get_booked_no_of_tickets(self):
        return self.total_seats - self.available_seats
```

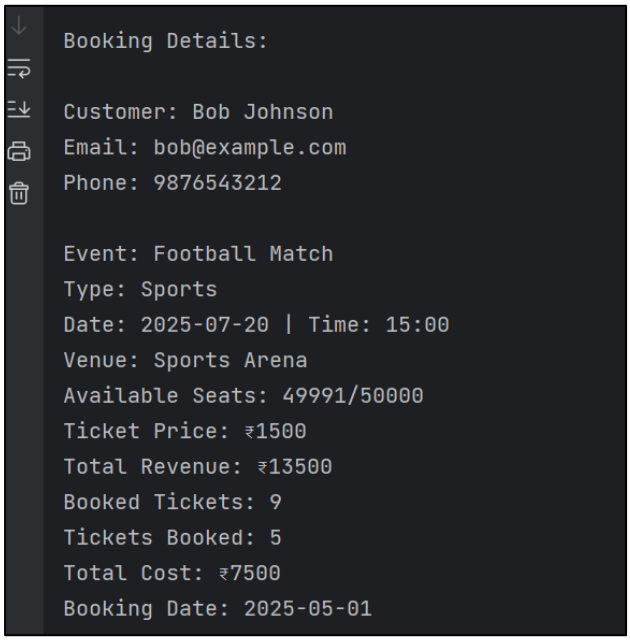
```
    def book_tickets(self, num_tickets):
        if self.available_seats >= num_tickets:
            self.available_seats -= num_tickets
            return True
        return False

    def display_event_details(self):
        print(f"\nEvent: {self.event_name}")
        print(f"Type: {self.event_type}")
        print(f"Date: {self.event_date} | Time: {self.event_time}")
        print(f"Venue: {self.venue_name}")
        print(f"Available Seats: {self.available_seats}/{self.total_seats}")
        print(f"Ticket Price: ₹{self.ticket_price}")
        print(f"Total Revenue: ₹{self.calculate_total_revenue()}")
        print(f"Booked Tickets: {self.get_booked_no_of_tickets()}")

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

    def display_venue_details(self):
        print(f"\nVenue: {self.venue_name}")
        print(f"Address: {self.address}")


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

    def display_customer_details(self):
        print(f"\nCustomer: {self.customer_name}")
        print(f"Email: {self.email}")
        print(f"Phone: {self.phone_number}")


class Booking:
    def __init__(self, event, customer, num_tickets):
        self.event = event
        self.customer = customer
        self.num_tickets = num_tickets
        self.total_cost = num_tickets * event.ticket_price
        self.booking_date = "2025-05-01"


        self.event.book_tickets(num_tickets)
```

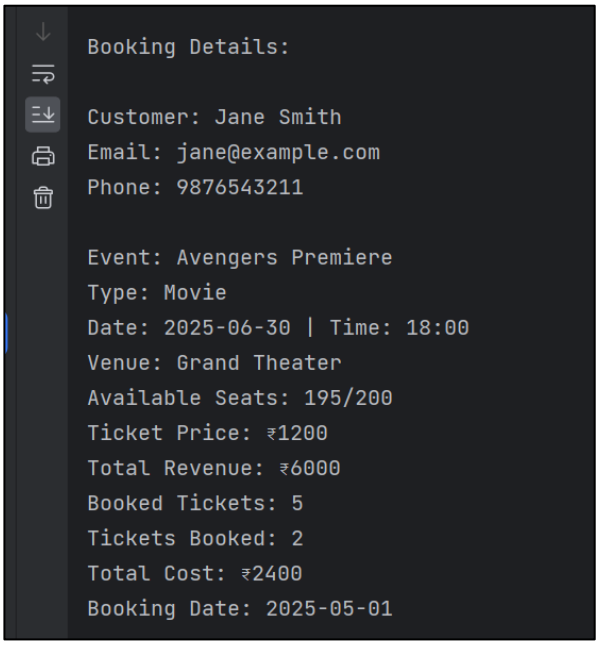
```
    def display_booking_details(self):
        print("\nBooking Details:")
        self.customer.display_customer_details()
        self.event.display_event_details()
        print(f"Tickets Booked: {self.num_tickets}")
        print(f"Total Cost: ₹{self.total_cost}")
        print(f"Booking Date: {self.booking_date}")


venue1 = Venue("Grand Theater", "123 Main St")
venue2 = Venue("Sports Arena", "456 Oak Ave")
venue3 = Venue("Sakthi Concert Hall","789 Main st")

event1 = Event("Avengers Premiere", "2025-06-30", "18:00", venue1.venue_name, 200, 1200,
"Movie")
event2 = Event("Football Match", "2025-07-20", "15:00", venue2.venue_name, 50000, 1500,
"Sports")
event3 = Event("AR Rahman Concert","2025-08-
01","20:00",venue3.venue_name,5000,800.00,'Concert')

customer1 = Customer("John Doe", "john@example.com", "9876543210")
customer2 = Customer("Jane Smith", "jane@example.com", "9876543211")
customer3 = Customer("Bob Johnson", "bob@example.com", "9876543212")

booking = Booking(event1, customer1, 3)
booking1 = Booking(event2, customer1, 4)
booking2 = Booking(event2, customer3, 5)
booking3 = Booking(event1, customer2, 2)
booking4 = Booking(event3, customer1 , 1)

booking.display_booking_details()
booking1.display_booking_details()
booking2.display_booking_details()
booking3.display_booking_details()
booking4.display_booking_details()
event1.display_event_details()
event2.display_event_details()
event3.display_event_details()
```

OUTPUT :

```
"C:\Users\Zulaiga\PycharmProjects\math operation\venv\Scripts\

Booking Details:

Customer: John Doe
Email: john@example.com
Phone: 9876543210

Event: Avengers Premiere
Type: Movie
Date: 2025-06-30 | Time: 18:00
Venue: Grand Theater
Available Seats: 195/200
Ticket Price: ₹1200
Total Revenue: ₹6000
Booked Tickets: 5
Tickets Booked: 3
Total Cost: ₹3600
Booking Date: 2025-05-01
```

```
Booking Details:

Customer: John Doe
Email: john@example.com
Phone: 9876543210

Event: Football Match
Type: Sports
Date: 2025-07-20 | Time: 15:00
Venue: Sports Arena
Available Seats: 49991/50000
Ticket Price: ₹1500
Total Revenue: ₹13500
Booked Tickets: 9
Tickets Booked: 4
Total Cost: ₹6000
Booking Date: 2025-05-01
```





```
Booking Details:

Customer: John Doe
Email: john@example.com
Phone: 9876543210

Event: AR Rahman Concert
Type: Concert
Date: 2025-08-01 | Time: 20:00
Venue: Sakthi Concert Hall
Available Seats: 4999/5000
Ticket Price: ₹800.0
Total Revenue: ₹800.0
Booked Tickets: 1
Tickets Booked: 1
Total Cost: ₹800.0
Booking Date: 2025-05-01
```

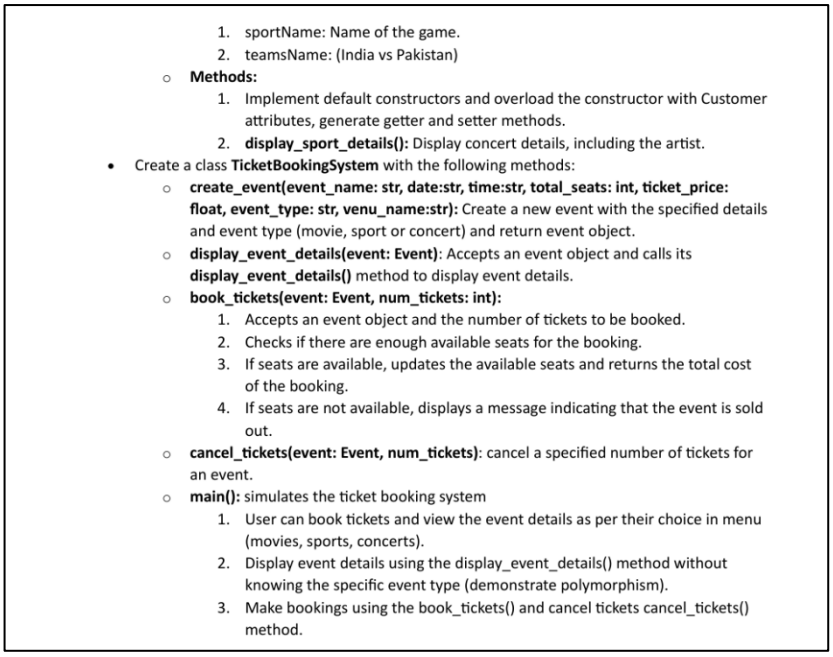
```
Event: Avengers Premiere
Type: Movie
Date: 2025-06-30 | Time: 18:00
Venue: Grand Theater
Available Seats: 195/200
Ticket Price: ₹1200
Total Revenue: ₹6000
Booked Tickets: 5

Event: Football Match
Type: Sports
Date: 2025-07-20 | Time: 15:00
Venue: Sports Arena
Available Seats: 49991/50000
Ticket Price: ₹1500
Total Revenue: ₹13500
Booked Tickets: 9
```

```
Event: AR Rahman Concert
Type: Concert
Date: 2025-08-01 | Time: 20:00
Venue: Sakthi Concert Hall
Available Seats: 4999/5000
Ticket Price: ₹800.0
Total Revenue: ₹800.0
Booked Tickets: 1
```

TASK 5 :

Task 5: Inheritance and polymorphism

1. **Inheritance**
   - Create a subclass **Movie** that inherits from **Event**. Add the following attributes and methods:
     - **Attributes:**
       1. genre: Genre of the movie (e.g., Action, Comedy, Horror).
       2. ActorName
       3. ActresName
     - **Methods:**
       1. Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
       2. **display_event_details():** Display movie details, including genre.
   - Create another subclass **Concert** that inherits from **Event**. Add the following attributes and methods:
     - **Attributes:**
       1. artist: Name of the performing artist or band.
       2. type: (Theatrical, Classical, Rock, Recital)
     - **Methods:**
       1. Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
       2. **display_concert_details():** Display concert details, including the artist.
   - Create another subclass **Sports** that inherits from **Event**. Add the following attributes and methods:
     - **Attributes:**



1. sportName: Name of the game.
2. teamsName: (India vs Pakistan)

- **Methods:**
  1. Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
  2. **display_sport_details():** Display concert details, including the artist.

- Create a class **TicketBookingSystem** with the following methods:
  - **create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu_name:str):** Create a new event with the specified details and event type (movie, sport or concert) and return event object.
  - **display_event_details(event: Event)**: Accepts an event object and calls its **display_event_details()** method to display event details.
  - **book_tickets(event: Event, num_tickets: int):**
    1. Accepts an event object and the number of tickets to be booked.
    2. Checks if there are enough available seats for the booking.
    3. If seats are available, updates the available seats and returns the total cost of the booking.
    4. If seats are not available, displays a message indicating that the event is sold out.
  - **cancel_tickets(event: Event, num_tickets)**: cancel a specified number of tickets for an event.
  - **main():** simulates the ticket booking system
    1. User can book tickets and view the event details as per their choice in menu (movies, sports, concerts).
    2. Display event details using the display_event_details() method without knowing the specific event type (demonstrate polymorphism).
    3. Make bookings using the book_tickets() and cancel tickets cancel_tickets() method.

CODE :

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


    def display_event_details(self):
        print(f"\nEvent: {self.event_name}")
        print(f"Event_type: {self.event_type}")
        print(f"Date: {self.date} | Time: {self.time}")
        print(f"Venue: {self.venue_name}")
        print(f"Available Seats: {self.available_seats}/{self.total_seats}")
        print(f"Ticket Price: ₹{self.ticket_price}")

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

class Movie(Event):
    def __init__(self, event_name, date, time, total_seats, ticket_price, venue_name, genre, actor,
```

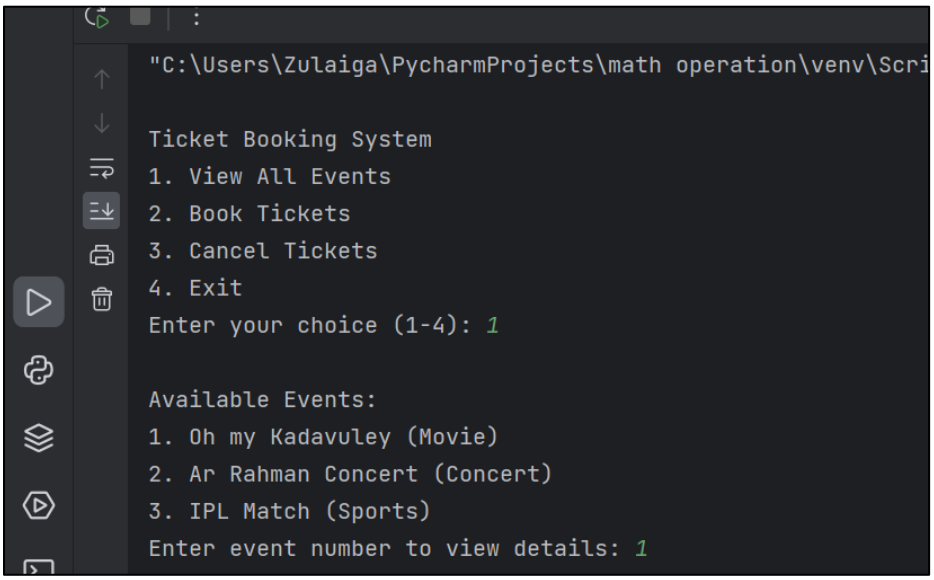
```
actress):
        super().__init__(event_name, date, time, total_seats, ticket_price, venue_name, "Movie")
        self.genre = genre
        self.actor = actor
        self.actress = actress

    def display_event_details(self):
        super().display_event_details()
        print(f"Genre: {self.genre}")
        print(f"Actor and Actress: {self.actor} and {self.actress}")

class Concert(Event):
    def __init__(self, event_name, date, time, total_seats, ticket_price, venue_name, artist,
concert_type):
        super().__init__(event_name, date, time, total_seats, ticket_price, venue_name, "Concert")
        self.artist = artist
        self.concert_type = concert_type

    def display_event_details(self):
        super().display_event_details()
        print(f"Artist: {self.artist}")
        print(f"Concert Type: {self.concert_type}")

class Sports(Event):
    def __init__(self, event_name, date, time, total_seats, ticket_price, venue_name, sport_name,
teams):
        super().__init__(event_name, date, time, total_seats, ticket_price, venue_name, "Sports")
        self.sport_name = sport_name
        self.teams = teams

    def display_event_details(self):
        super().display_event_details()
        print(f"Sport: {self.sport_name}")
        print(f"Teams: {self.teams}")

class TicketBookingSystem:
    def __init__(self):
        self.events = []

    def create_event(self, event_type, **kwargs):
        if event_type == "Movie":
            event = Movie(**kwargs)
        elif event_type == "Concert":
            event = Concert(**kwargs)
        elif event_type == "Sports":
            event = Sports(**kwargs)
        else:
            raise ValueError("Invalid event type")
```

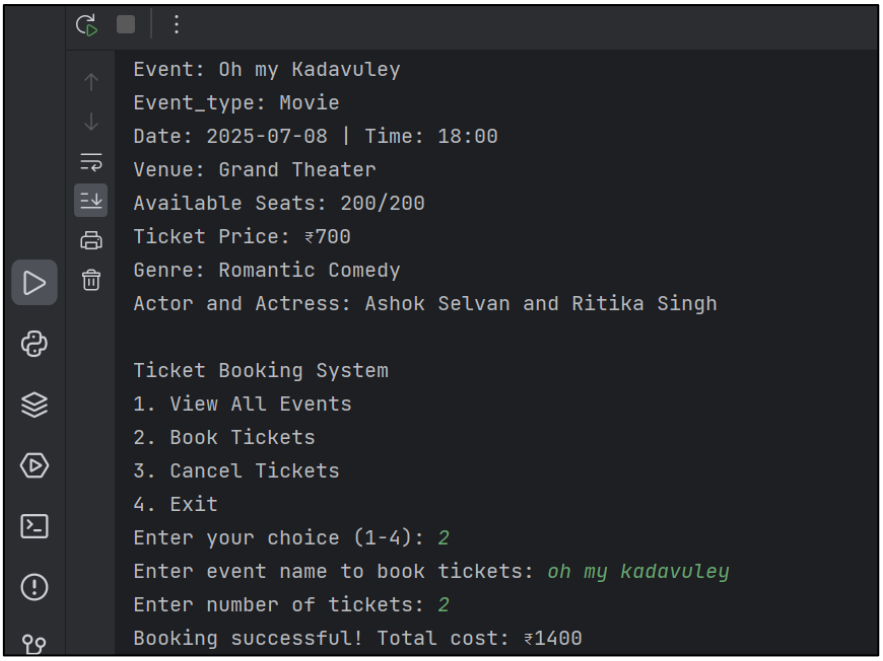
```
    self.events.append(event)
    return event

def book_tickets(self, event_name, num_tickets:int):
    for event in self.events:
        if event.event_name == event_name:
            if event.available_seats >= num_tickets:
                event.available_seats -= num_tickets
                return num_tickets * event.ticket_price
            else:
                print(f"Only {event.available_seats} seats available!")
                return None
    print("Event not found!")
    return None

def cancel_tickets(self, event: Event, num_tickets):
    event.cancel_booking(num_tickets)
    print(f"Cancelled {num_tickets} tickets for {event.event_name}")

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

def find_event_by_name(self, name: str) -> Event:
    for event in self.events:
        if event.event_name.lower() == name.lower():
            return event
    return None

def main(self):
    self.create_event("Movie",
event_name="Oh my Kadavuley",
date="2025-07-08",
time="18:00",
total_seats=200,
ticket_price=700,
venue_name="Grand Theater",
genre="Romantic Comedy",
actor="Ashok Selvan",
actress="Ritika Singh")

    self.create_event("Concert",
      event_name = "Ar Rahman Concert",
      date = "2025-08-01",
      time = "20:00",
      total_seats = 1000,
      ticket_price = 800.00,
      venue_name = "Shakthi concert hall",
      artist = "AR Rahman",
```

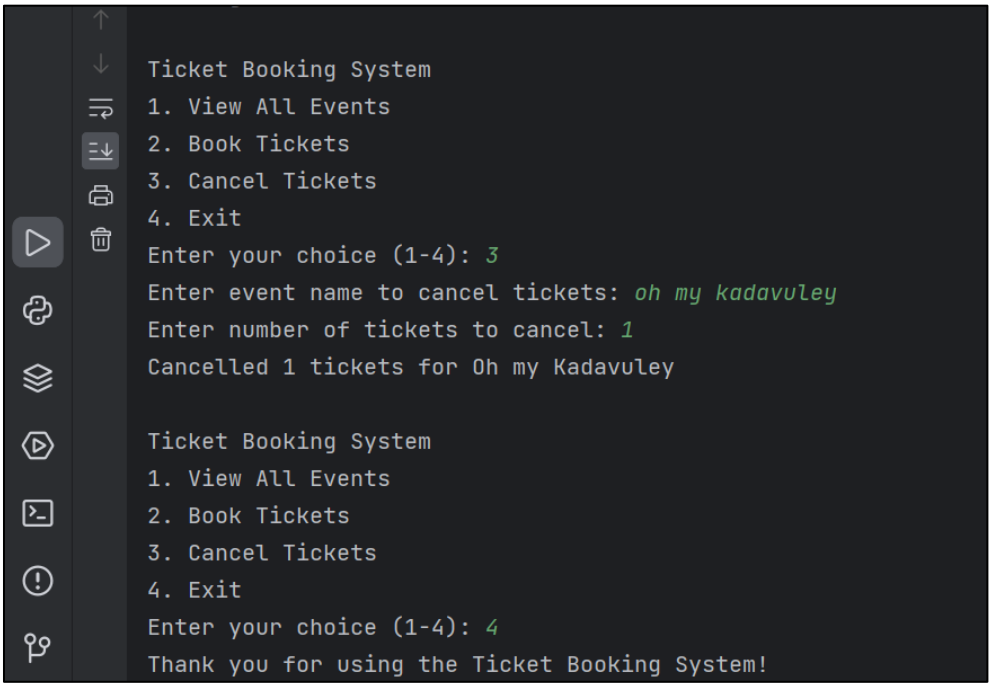
```
  concert_type = "Classical Concert")

self.create_event("Sports",
  event_name = "IPL Match",
  date = "2025-07-05",
  time = "19:30",
  total_seats = 12000,
  ticket_price = 850.00,
  venue_name = "Chinnaswamy stadium",
  sport_name = "Cricket",
  teams = "CSK VS RCB")

while True:
    print("\nTicket Booking System")
    print("1. View All Events")
    print("2. Book Tickets")
    print("3. Cancel Tickets")
    print("4. Exit")

    choice = input("Enter your choice (1-4): ")

    if choice == "1":
        print("\nAvailable Events:")
        for i, event in enumerate(self.events, 1):
            print(f"{i}. {event.event_name} ({event.__class__.__name__})")

        event_choice = input("Enter event number to view details: ")
        if event_choice.isdigit() and 0 < int(event_choice) <= len(self.events):
            self.display_event_details(self.events[int(event_choice) - 1])

    elif choice == "2":
        event_name = input("Enter event name to book tickets: ")
        event = self.find_event_by_name(event_name)
        if event:
            try:
                num_tickets = int(input("Enter number of tickets: "))
                total_cost = self.book_tickets(event.event_name, num_tickets)
                if total_cost > 0:
                    print(f"Booking successful! Total cost: ₹{total_cost}")
            except ValueError:
                print("Please enter a valid number")
        else:
            print("Event not found")

    elif choice == "3":
        event_name = input("Enter event name to cancel tickets: ")
        event = self.find_event_by_name(event_name)
        if event:
```

```
                try:
                    num_tickets = int(input("Enter number of tickets to cancel: "))
                    self.cancel_tickets(event, num_tickets)
                except ValueError:
                    print("Please enter a valid number")
            else:
                print("Event not found")

        elif choice == "4":
            print("Thank you for using the Ticket Booking System!")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    system = TicketBookingSystem()
    system.main()
```

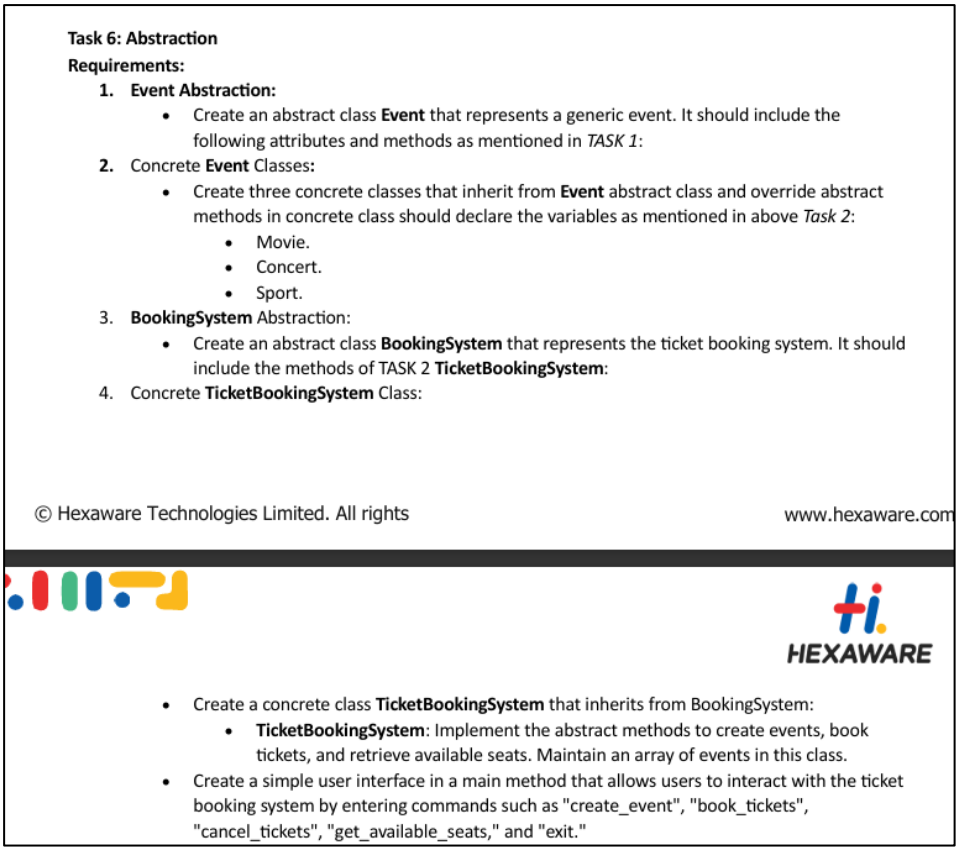
OUTPUT :




Event: Oh my Kadavuley
Event_type: Movie
Date: 2025-07-08 | Time: 18:00
Venue: Grand Theater
Available Seats: 200/200
Ticket Price: ₹700
Genre: Romantic Comedy
Actor and Actress: Ashok Selvan and Ritika Singh
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Exit
Enter your choice (1-4): 2
Enter event name to book tickets: oh my kadavuley
Enter number of tickets: 2
Booking successful! Total cost: ₹1400


Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Exit
Enter your choice (1-4): 3
Enter event name to cancel tickets: oh my kadavuley
Enter number of tickets to cancel: 1
Cancelled 1 tickets for Oh my Kadavuley
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Exit
Enter your choice (1-4): 4
Thank you for using the Ticket Booking System!

TASK 6 :



Task 6: Abstraction

Requirements:

1. **Event Abstraction:**
   - Create an abstract class **Event** that represents a generic event. It should include the following attributes and methods as mentioned in *TASK 1*:
2. Concrete **Event** Classes:
   - Create three concrete classes that inherit from **Event** abstract class and override abstract methods in concrete class should declare the variables as mentioned in above *Task 2*:
     - Movie.
     - Concert.
     - Sport.
3. **BookingSystem** Abstraction:
   - Create an abstract class **BookingSystem** that represents the ticket booking system. It should include the methods of TASK 2 **TicketBookingSystem**:
4. Concrete **TicketBookingSystem** Class:

HEXAWARE

- Create a concrete class **TicketBookingSystem** that inherits from BookingSystem:
  - **TicketBookingSystem**: Implement the abstract methods to create events, book tickets, and retrieve available seats. Maintain an array of events in this class.
- Create a simple user interface in a main method that allows users to interact with the ticket booking system by entering commands such as "create_event", "book_tickets", "cancel_tickets", "get_available_seats," and "exit."

CODE :

```
from abc import ABC, abstractmethod


class Event(ABC):
    def __init__(self,event_name,date,time,total_seats,ticket_price,venue_name,event_type):
        self.event_name = event_name
        self.date = date
        self.time = time
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.venue_name = venue_name
        self.event_type = event_type

    @abstractmethod
    def display_event_details(self):
        pass

    def calculate_total_revenue(self):
```

```
        return (self.total_seats - self.available_seats) * self.ticket_price

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

class Movie(Event):

    def __init__(self,event_name,date,time,total_seats,ticket_price,venue_name,actor,actress,genre):
        super().__init__(event_name,date,time,total_seats,ticket_price,venue_name,"Movie")
        self.actor = actor
        self.actress = actress
        self.genre = genre

    def display_event_details(self):
        print("Displaying Movie Details")
        print(f"Event_name:{self.event_name}")
        print(f"Genre:{self.genre}")
        print(f"Actor:{self.actor} Actress:{self.actress}")
        print(f"Date:{self.date}")
        print(f"Time:{self.time}")
        print(f"Total_seats:{self.total_seats}")
        print(f"Ticket_price:₹{self.ticket_price}")
        print(f"Venue_name:{self.venue_name}")

class Concert(Event):

    def __init__(self,event_name,date,time,total_seats,ticket_price,venue_name,artist,concert_type):
        super().__init__(event_name,date,time,total_seats,ticket_price,venue_name,"Concert")

        self.artist = artist
        self.concert_type = concert_type

    def display_event_details(self):
        print("Displaying Concert Details")
        print(f"Event_name:{self.event_name}")
        print(f"Artist:{self.artist}")
        print(f"Concert_type:{self.concert_type}")
        print(f"Date:{self.date}")
        print(f"Time:{self.time}")
        print(f"Total_seats:{self.total_seats}")
        print(f"Ticket_price:₹{self.ticket_price}")
        print(f"Venue_name:{self.venue_name}")

class Sports(Event):

    def __init__(self, event_name, date, time, total_seats, ticket_price, venue_name, sports_name,
teams):
        super().__init__(event_name, date, time, total_seats, ticket_price, venue_name, "Sports")
```

```
        self.sports_name = sports_name
        self.teams = teams

    def display_event_details(self):
        print("Displaying Concert Details")
        print(f"Event_name:{self.event_name}")
        print(f"Sport_Name:{self.sports_name}")
        print(f"Teams:{self.teams}")
        print(f"Date:{self.date}")
        print(f"Time:{self.time}")
        print(f"Total_seats:{self.total_seats}")
        print(f"Ticket_price:₹{self.ticket_price}")
        print(f"Venue_name:{self.venue_name}")


class Bookingsystem(ABC):

    @abstractmethod
    def create_event(self,event_type,**kwargs):
        pass

    @abstractmethod
    def book_tickets(self,event,num_tickets):
        pass

    @abstractmethod
    def cancel_tickets(self,event,num_tickets):
        pass

    @abstractmethod
    def display_event_details(self,event):
        pass

    @abstractmethod
    def main(self):
        pass

class TicketBookingSystem(Bookingsystem):

    def __init__(self):
        self.events = []

    def create_event(self,event_type,**kwargs):
        if event_type == "Movie":
            event = Movie(**kwargs)
        elif event_type == "Concert":
            event = Concert(**kwargs)
```

```
        elif event_type == "Sports":
            event = Sports(**kwargs)
        else:
            raise ValueError("Invalid event type")

        self.events.append(event)
        return event

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

    def book_tickets(self, event, num_tickets):
        if event.available_seats >= num_tickets:
            event.available_seats -= num_tickets
            return num_tickets * event.ticket_price
        else:
            print(f"Event is sold out! Only {event.available_seats} seats available.")
            return 0.0

    def cancel_tickets(self, event, num_tickets):
        if (event.available_seats + num_tickets) <= event.total_seats:
            event.available_seats += num_tickets
            print(f"Cancelled {num_tickets} tickets for {event.event_name}")
        else:
            print("Cannot cancel more tickets than were booked!")

    def get_available_seats(self, event):
        return event.available_seats

    def main(self):

        movie = self.create_event("Movie",
            event_name="Oh my Kadavuley",
            date="2025-08-15",
            time="18:00",
            total_seats=200,
            ticket_price=700.00,
            venue_name="Grand Theater",
            actor="Ashok Selvan",
            actress="Ritika Singh",
            genre="Romantic comedy",
        )

        concert = self.create_event( "Concert",
            event_name="A.R. Rahman Live",
            date="2025-07-15",
            time="20:00",
            total_seats=10000,
```

```
    ticket_price=2500,
    venue_name="Arena",
    artist="A.R. Rahman",
    concert_type="World Music"
)

sports = self.create_event("Sports",
    event_name="IPL Final",
    date="2025-06-30",
    time="19:30",
    total_seats=50000,
    ticket_price=1500,
    venue_name="Stadium",
    sport_name="Cricket",
    teams="India vs Pakistan"
)

while True:
    print("\nTicket Booking System")
    print("1. View All Events")
    print("2. Book Tickets")
    print("3. Cancel Tickets")
    print("4. Check Available Seats")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == "1":
        print("\nAvailable Events:")
        for i, event in enumerate(self.events, 1):
            print(f"{i}. {event.event_name} ({event.event_type})")

        event_choice = input("Enter event number to view details: ")
        self.display_event_details(self.events[int(event_choice) - 1])

    elif choice == "2":
        print("\nAvailable Events:")
        for i, event in enumerate(self.events, 1):
            print(f"{i}. {event.event_name} ({event.event_type})")

        event_choice = input("Enter event number to book tickets: ")
        event = self.events[int(event_choice) - 1]
        try:
            num_tickets = int(input("Enter number of tickets: "))
            total_cost = self.book_tickets(event, num_tickets)
            if total_cost > 0:
                print(f"Booking successful! Total cost: ₹{total_cost}")
        except ValueError:
```

```
                print("Please enter a valid number")

        elif choice == "3":
            print("\nAvailable Events:")
            for i, event in enumerate(self.events, 1):
                print(f"{i}. {event.event_name} ({event.event_type})")

            event_choice = input("Enter event number to cancel tickets: ")
            event = self.events[int(event_choice) - 1]
            try:
                num_tickets = int(input("Enter number of tickets to cancel: "))
                self.cancel_tickets(event, num_tickets)
            except ValueError:
                 print("Please enter a valid number")

        elif choice == "4":
            print("\nAvailable Events:")
            for i, event in enumerate(self.events, 1):
                print(f"{i}. {event.event_name} ({event.event_type})")

            event_choice = input("Enter event number to check seats: ")
            event = self.events[int(event_choice) - 1]
            print(f"Available seats for {event.event_name}: {self.get_available_seats(event)}")

        elif choice == "5":
            print("Thank you for using the Ticket Booking System!")
            break

        else:
            print("Invalid choice. Please try again.")



if __name__ == "__main__":
    system = TicketBookingSystem()
    system.main()
```

OUTPUT :

```
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Check Available Seats
5. Exit
Enter your choice (1-5): 1

Available Events:
1. Oh my Kadavuley (Movie)
2. A.R. Rahman Live (Concert)
3. IPL Final (Sports)
Enter event number to view details: 1
Displaying Movie Details
Event_name:Oh my Kadavuley
Genre:Romantic comedy
Actor:Ashok Selvan Actress:Ritika Singh
Date:2025-08-15
Time:18:00
Total_seats:200
Ticket_price:₹700.0
Venue_name:Grand Theater
```

```
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Check Available Seats
5. Exit
Enter your choice (1-5): 2

Available Events:
1. Oh my Kadavuley (Movie)
2. A.R. Rahman Live (Concert)
3. IPL Final (Sports)
Enter event number to book tickets: 1
Enter number of tickets: 2
Booking successful! Total cost: ₹1400.0
```

```
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Check Available Seats
5. Exit
Enter your choice (1-5): 4

Available Events:
1. Oh my Kadavuley (Movie)
2. A.R. Rahman Live (Concert)
3. IPL Final (Sports)
Enter event number to check seats: 1
Available seats for Oh my Kadavuley: 198
```

```
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Check Available Seats
5. Exit
Enter your choice (1-5): 3

Available Events:
1. Oh my Kadavuley (Movie)
2. A.R. Rahman Live (Concert)
3. IPL Final (Sports)
Enter event number to cancel tickets: 1
Enter number of tickets to cancel: 1
Cancelled 1 tickets for Oh my Kadavuley
```

```
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
4. Check Available Seats
5. Exit
Enter your choice (1-5): 4

Available Events:
1. Oh my Kadavuley (Movie)
2. A.R. Rahman Live (Concert)
3. IPL Final (Sports)
Enter event number to check seats: 1
Available seats for Oh my Kadavuley: 199
```

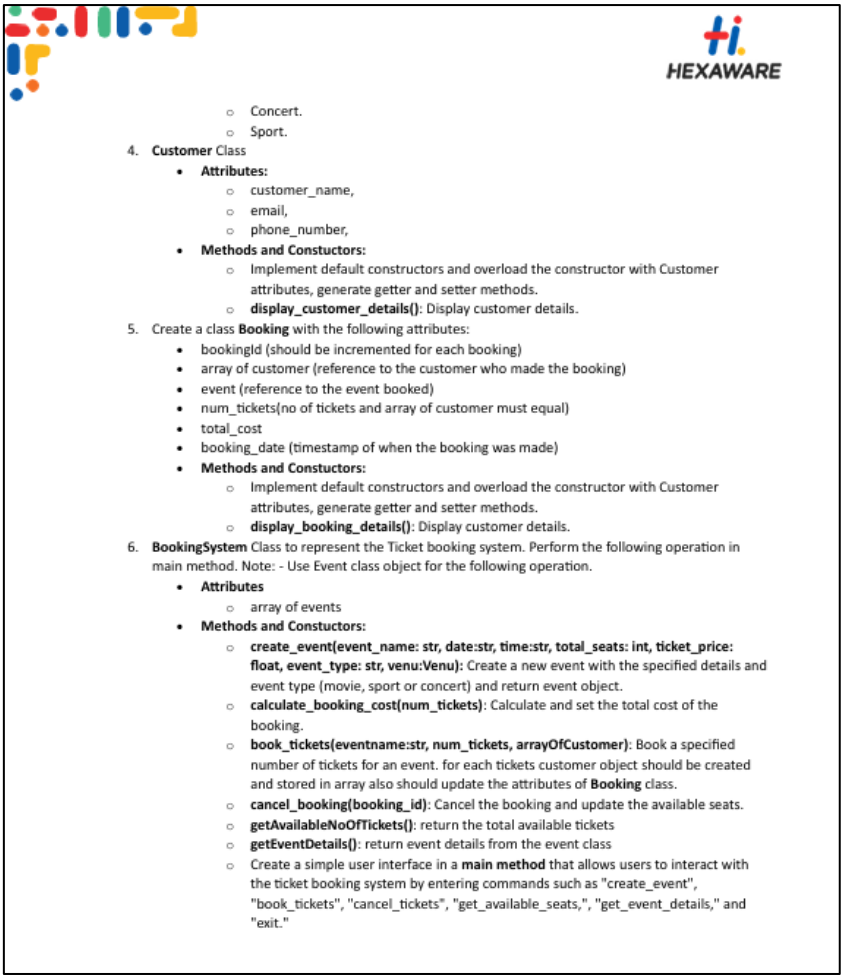
```
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Tickets
Services Alt+8 Available Seats
5. Exit
Enter your choice (1-5): 5
Thank you for using the Ticket Booking System!
```

TASK 7:

Task 7: Has A Relation / Association

Create a Following classes with the following attributes and methods:

1. **Venue** Class
   - **Attributes**:
     - venue_name,
     - address
   - **Methods and Constuctors:**
     - **display_venue_details():** Display venue details.
     - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
2. **Event** Class:
   - **Attributes:**
     - event_name,
     - event_date DATE,
     - event_time TIME,
     - venue (reference of class **Venu**),
     - total_seats,
     - available_seats,
     - ticket_price DECIMAL,
     - event_type ENUM('Movie', 'Sports', 'Concert')
   - **Methods and Constuctors:**
     - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter, (print all information of attribute) methods for the attributes.
     - **calculate_total_revenue()**: Calculate and return the total revenue based on the number of tickets sold.
     - **getBookedNoOfTickets()**: return the total booked tickets
     - **book_tickets(num_tickets)**: Book a specified number of tickets for an event. Initially available seats are equal to total seats when tickets are booked available seats number should be reduced.
     - **cancel_booking(num_tickets)**: Cancel the booking and update the available seats.
     - **display_event_details():** Display event details, including event name, date time seat availability.
3. **Event** sub classes:
   - Create three sub classes that inherit from **Event** abstract class and override abstract methods in concrete class should declare the variables as mentioned in above *Task 2*:
     - Movie.



- Concert.
- Sport.

4. **Customer** Class
   - **Attributes:**
     - customer_name,
     - email,
     - phone_number,
   - **Methods and Constuctors:**
     - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
     - **display_customer_details()**: Display customer details.
5. Create a class **Booking** with the following attributes:
   - bookingId (should be incremented for each booking)
   - array of customer (reference to the customer who made the booking)
   - event (reference to the event booked)
   - num_tickets(no of tickets and array of customer must equal)
   - total_cost
   - booking_date (timestamp of when the booking was made)
   - **Methods and Constuctors:**
     - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
     - **display_booking_details()**: Display customer details.
6. **BookingSystem** Class to represent the Ticket booking system. Perform the following operation in main method. Note: - Use Event class object for the following operation.
   - **Attributes**
     - array of events
   - **Methods and Constuctors:**
     - **create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu:Venu):** Create a new event with the specified details and event type (movie, sport or concert) and return event object.
     - **calculate_booking_cost(num_tickets)**: Calculate and set the total cost of the booking.
     - **book_tickets(eventname:str, num_tickets, arrayOfCustomer)**: Book a specified number of tickets for an event. for each tickets customer object should be created and stored in array also should update the attributes of **Booking** class.
     - **cancel_booking(booking_id)**: Cancel the booking and update the available seats.
     - **getAvailableNoOfTickets()**: return the total available tickets
     - **getEventDetails()**: return event details from the event class
     - Create a simple user interface in a **main method** that allows users to interact with the ticket booking system by entering commands such as "create_event", "book_tickets", "cancel_tickets", "get_available_seats,", "get_event_details," and "exit."

CODE:

```
from abc import ABC, abstractmethod
from datetime import datetime
from enum import Enum


class EventType(Enum):
    MOVIE = "Movie"
    SPORTS = "Sports"
    CONCERT = "Concert"


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: {self.address}")
```

```
class Event(ABC):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, event_type=None):
        self.event_name = event_name
        self.event_date = event_date
        self.event_time = event_time
        self.venue = venue
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.event_type = event_type

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

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

    def book_tickets(self, num_tickets):
        if self.available_seats >= num_tickets:
            self.available_seats -= num_tickets
            return True
        return False

    def cancel_booking(self, num_tickets):
        if (self.available_seats + num_tickets) <= self.total_seats:
            self.available_seats += num_tickets
            return True
        return False

    @abstractmethod
    def display_event_details(self):
        pass


class Movie(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, genre="", actor="", actress=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.MOVIE)
        self.genre = genre
        self.actor = actor
        self.actress = actress
```

```
    def display_event_details(self):
        print("\nMovie Details:")
        print(f"Event Name: {self.event_name}")
        print(f"Date: {self.event_date}")
        print(f"Time: {self.event_time}")
        print(f"Genre: {self.genre}")
        print(f"Actor: {self.actor} Actress: {self.actress}")
        print(f"Venue: {self.venue.venue_name}")
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: {self.ticket_price}")


class Concert(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, artist="", concert_type=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.CONCERT)
        self.artist = artist
        self.concert_type = concert_type

    def display_event_details(self):
        print("\nConcert Details:")
        print(f"Event Name: {self.event_name}")
        print(f"Date: {self.event_date}")
        print(f"Time: {self.event_time}")
        print(f"Artist: {self.artist}")
        print(f"Concert Type: {self.concert_type}")
        print(f"Venue: {self.venue.venue_name}")
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: {self.ticket_price}")


class Sports(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, sport_name="", teams=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.SPORTS)
        self.sport_name = sport_name
        self.teams = teams

    def display_event_details(self):
        print("\nSports Event Details:")
        print(f"Event Name: {self.event_name}")
        print(f"Date: {self.event_date}")
```

```
        print(f"Time: {self.event_time}")
        print(f"Sport: {self.sport_name}")
        print(f"Teams: {self.teams}")
        print(f"Venue: {self.venue.venue_name}")
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: {self.ticket_price:}")


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

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


class Booking:
    booking_count = 1

    def __init__(self, customers=None, event=None, num_tickets=0, total_cost=0.0):
        self.booking_id = Booking.booking_count
        Booking.booking_count += 1
        self.customers = customers if customers else []
        self.event = event
        self.num_tickets = num_tickets
        self.total_cost = total_cost
        self.booking_date = datetime.now()

    def display_booking_details(self):
        print(f"Booking ID: {self.booking_id}")
        print(f"Booking Date: {self.booking_date}")
        print(f"Event: {self.event.event_name}")
        print(f"Number of Tickets: {self.num_tickets}")
        print(f"Total Cost: {self.total_cost}")
        print("Customers:")
        for customer in self.customers:
            customer.display_customer_details()


class BookingSystem:
    def __init__(self):
```

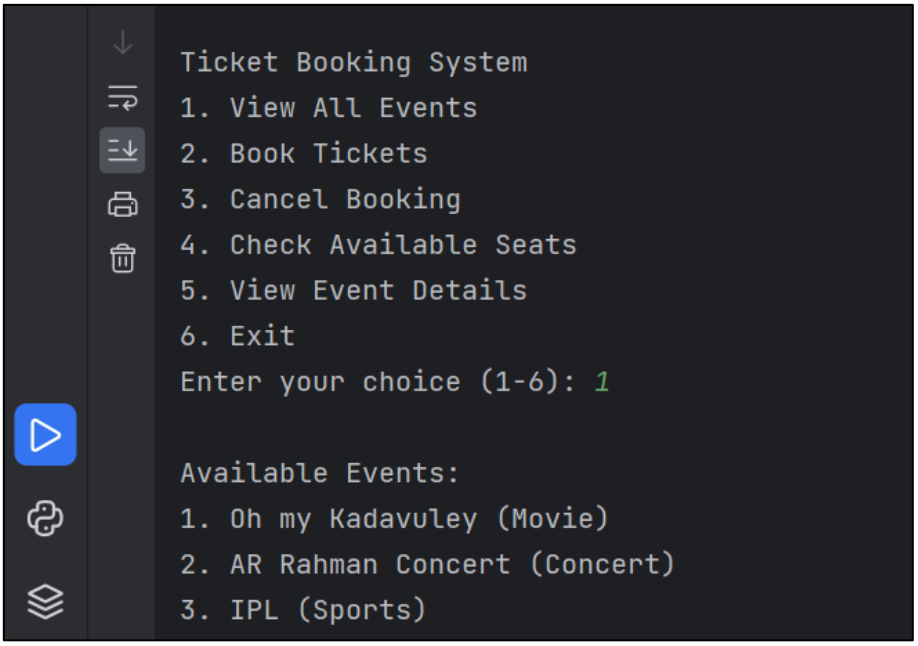
```
    self.events = []
    self.bookings = []

def create_event(self, event_name, date, time, total_seats, ticket_price,
                 event_type, venue, **kwargs):
    event = None
    if event_type == EventType.MOVIE:
        event = Movie(event_name, date, time, venue, total_seats, ticket_price,
                      kwargs.get('genre'), kwargs.get('actor'), kwargs.get('actress'))
    elif event_type == EventType.CONCERT:
        event = Concert(event_name, date, time, venue, total_seats, ticket_price,
                        kwargs.get('artist'), kwargs.get('concert_type'))
    elif event_type == EventType.SPORTS:
        event = Sports(event_name, date, time, venue, total_seats, ticket_price,
                       kwargs.get('sport_name'), kwargs.get('teams'))

    if event:
        self.events.append(event)
    return event

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

def book_tickets(self, event_name, num_tickets, customers):
    event = None
    for e in self.events:
        if e.event_name.lower() == event_name.lower():
            event = e
            break
    if not event:
        print("Event not found!")
        return None

    if len(customers) != num_tickets:
        print("Number of customers must match number of tickets!")
        return None

    if event.book_tickets(num_tickets):
        total_cost = self.calculate_booking_cost(num_tickets, event)
        booking = Booking(customers, event, num_tickets, total_cost)
        self.bookings.append(booking)
        return booking
    else:
        print("Not enough seats available!")
        return None
```

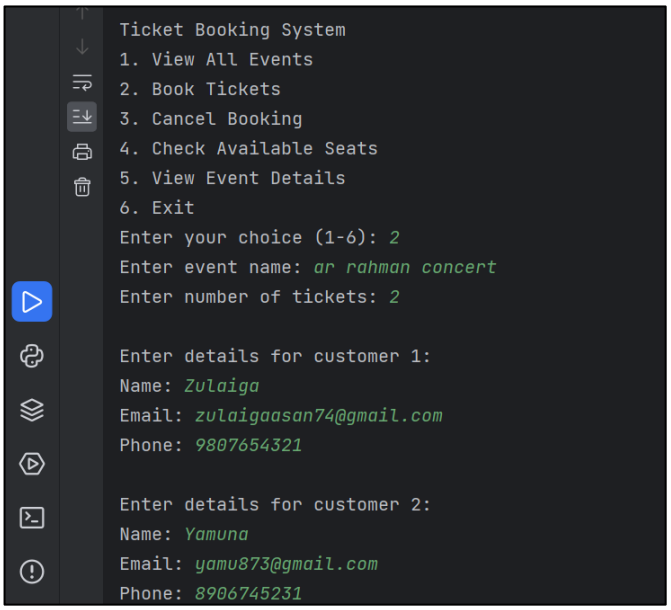
```
def cancel_booking(self, booking_id):
    booking = None
    for b in self.bookings:
        if b.booking_id == booking_id:
            booking = b
            break
    if not booking:
        print("Booking not found!")
        return False

    event = booking.event
    if event.cancel_booking(booking.num_tickets):
        self.bookings.remove(booking)
        print(f"Booking {booking_id} cancelled successfully.")
        return True
    else:
        print("Failed to cancel booking.")
        return False

def get_available_no_of_tickets(self, event_name):
    event = None
    for e in self.events:
        if e.event_name.lower() == event_name.lower():
            event = e
            break
    return event.available_seats if event else 0

def get_event_details(self, event_name):
    event = None
    for e in self.events:
        if e.event_name.lower() == event_name.lower():
            event = e
            break
    if event:
        event.display_event_details()
    else:
        print("Event not found!")

def main(self):

    venue1 = Venue("Dorais Theatre", "Opp to Pazhamudir")
    venue2 = Venue("Sakthi concert hall", "Market Road")
    venue3 = Venue("Sports Arena", "789 Stadium Rd")

    movie = self.create_event(
        "Oh my Kadavuley", "2025-06-25", "18:00", 200, 250.00,
```

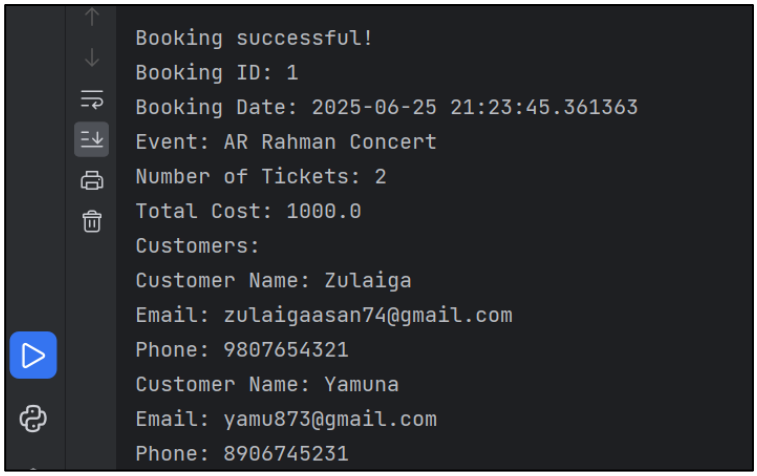
```
    EventType.MOVIE, venue1, genre="Romantic Comedy", actor="Ashok Selvan",
    actress="Ritika Singh"
)

concert = self.create_event(
    "AR Rahman Concert", "2025-07-01", "20:00", 5000, 500.00,
    EventType.CONCERT, venue2, artist="AR Rahman",
    concert_type="Classical"
)

sports = self.create_event(
    "IPL", "2025-06-18", "15:30", 30000, 750.00,
    EventType.SPORTS, venue3, sport_name="Cricket",
    teams="CSK vs RCB"
)

while True:
    print("\nTicket Booking System")
    print("1. View All Events")
    print("2. Book Tickets")
    print("3. Cancel Booking")
    print("4. Check Available Seats")
    print("5. View Event Details")
    print("6. Exit")

    choice = input("Enter your choice (1-6): ")

    if choice == "1":
        print("\nAvailable Events:")
        for i, event in enumerate(self.events, 1):
            print(f"{i}. {event.event_name} ({event.event_type.value})")

    elif choice == "2":
        event_name = input("Enter event name: ")
        try:
            num_tickets = int(input("Enter number of tickets: "))
            customers = []
            for i in range(num_tickets):
                print(f"\nEnter details for customer {i + 1}:")
                name = input("Name: ")
                email = input("Email: ")
                phone = input("Phone: ")
                customers.append(Customer(name, email, phone))

            booking = self.book_tickets(event_name, num_tickets, customers)
            if booking:
```

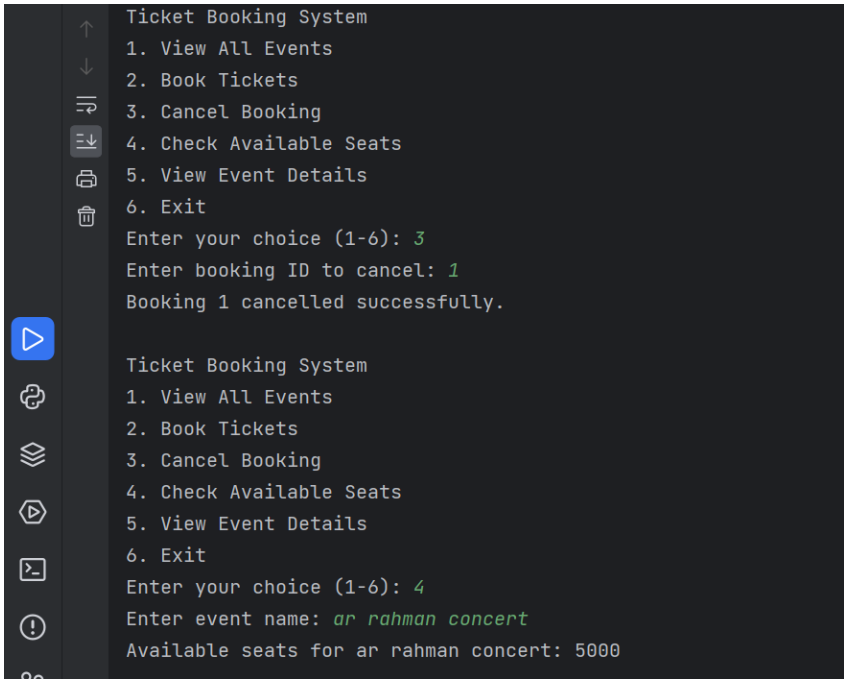
```
                print("\nBooking successful!")
                booking.display_booking_details()
            except ValueError:
                print("Please enter a valid number")

        elif choice == "3":
            try:
                booking_id = int(input("Enter booking ID to cancel: "))
                self.cancel_booking(booking_id)
            except ValueError:
                print("Please enter a valid booking ID")

        elif choice == "4":
            event_name = input("Enter event name: ")
            available = self.get_available_no_of_tickets(event_name)
            print(f"Available seats for {event_name}: {available}")

        elif choice == "5":
            event_name = input("Enter event name: ")
            self.get_event_details(event_name)

        elif choice == "6":
            print("Thank you for using the Ticket Booking System!")
            break

        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    system = BookingSystem()
    system.main()
```

OUTPUT:


Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 1
Available Events:
1. Oh my Kadavuley (Movie)
2. AR Rahman Concert (Concert)
3. IPL (Sports)


Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 2
Enter event name: ar rahman concert
Enter number of tickets: 2
Enter details for customer 1:
Name: Zulaiga
Email: zulaigaasan74@gmail.com
Phone: 9807654321
Enter details for customer 2:
Name: Yamuna
Email: yamu873@gmail.com
Phone: 8906745231


Booking successful!
Booking ID: 1
Booking Date: 2025-06-25 21:23:45.361363
Event: AR Rahman Concert
Number of Tickets: 2
Total Cost: 1000.0
Customers:
Customer Name: Zulaiga
Email: zulaigaasan74@gmail.com
Phone: 9807654321
Customer Name: Yamuna
Email: yamu873@gmail.com
Phone: 8906745231


Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 3
Enter booking ID to cancel: 1
Booking 1 cancelled successfully.
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 4
Enter event name: ar rahman concert
Available seats for ar rahman concert: 5000

TASK 8 :

Task 8: Interface/abstract class, and Single Inheritance, static variable

1. Create **Venue,** class as mentioned above Task 4.
2. **Event** Class**:**
   - **Attributes:**
     - event_name,
     - event_date DATE,
     - event_time TIME,
     - venue (reference of class Venu),
     - total_seats,
     - available_seats,
     - ticket_price DECIMAL,
     - event_type ENUM('Movie', 'Sports', 'Concert')
   - **Methods and Constuctors:**
     - Implement default constructors and overload the constructor with Customer attributes, generate getter and setter, (print all information of attribute) methods for the attributes.
3. Create **Event** sub classes as mentioned in above Task 4**.**
4. Create a class **Customer** and **Booking** as mentioned in above Task 4**.**
5. Create interface/abstract class **IEventServiceProvider** with following methods:
   - **create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu: Venu):** Create a new event with the specified details and event type (movie, sport or concert) and return event object.
   - **getEventDetails():** return array of event details from the event class.
   - **getAvailableNoOfTickets()**: return the total available tickets.
6. Create interface/abstract class **IBookingSystemServiceProvider** with following methods:
   - **calculate_booking_cost(num_tickets):** Calculate and set the total cost of the booking.
   - **book_tickets(eventname:str, num_tickets, arrayOfCustomer):** Book a specified number of tickets for an event. for each tickets customer object should be created and stored in array also should update the attributes of Booking class.
   - **cancel_booking(booking_id):** Cancel the booking and update the available seats.
   - **get_booking_details(booking_id):**get the booking details.
7. Create **EventServiceProviderImpl** class which implements **IEventServiceProvider** provide all implementation methods.
8. Create **BookingSystemServiceProviderImpl** class which implements **IBookingSystemServiceProvider** provide all implementation methods and inherits **EventServiceProviderImpl** class with following attributes.
   - **Attributes**
     - array of events
9. Create **TicketBookingSystem** class and perform following operations:
   - Create a simple user interface in a main method that allows users to interact with the ticket booking system by entering commands such as "create_event", "book_tickets", "cancel_tickets", "get_available_seats,", "get_event_details," and "exit."
10. Place the interface/abstract class in service package and interface/abstract class implementation class, all concrete class in bean package and **TicketBookingSystem** class in app package.

CODE :

```
from enum import Enum
from datetime import date, time
from abc import ABC, abstractmethod

class EventType(Enum):
    MOVIE = 'Movie'
    SPORTS = 'Sports'
    CONCERT = 'Concert'


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

```
        self.address = address

    def display_venue_details(self):
        print(f"Venue: {self.venue_name}")
        print(f"Address: {self.address}")


class Event:
    def __init__(self, event_name, date, time, venue, total_seats, ticket_price, event_type):
        self.event_name = event_name
        self.date = date
        self.time = time
        self.venue = venue
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.event_type = event_type

    def display_event_details(self):
        print(f"\nEvent: {self.event_name}")
        print(f"Type: {self.event_type.value}")
        print(f"Date: {self.date}")
        print(f"Time: {self.time}")
        self.venue.display_venue_details()
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: ${self.ticket_price}")


class MovieEvent(Event):
    def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price,
genre, actor, actress):
        super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price,
EventType.MOVIE)
        self.genre = genre
        self.actor = actor
        self.actress = actress

    def display_event_details(self):
        super().display_event_details()
        print(f"Genre: {self.genre}")
        print(f"Actor: {self.actor}")
        print(f"Actress: {self.actress}")


class SportsEvent(Event):
```

```
    def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price,
sport_name, teams):
        super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price,
EventType.SPORTS)
        self.sport_name = sport_name
        self.teams = teams

    def display_event_details(self):
        super().display_event_details()
        print(f"Sport: {self.sport_name}")
        print(f"Teams: {self.teams}")


class ConcertEvent(Event):
    def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price,
artist, concert_type):
        super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price,
EventType.CONCERT)
        self.artist = artist
        self.concert_type = concert_type

    def display_event_details(self):
        super().display_event_details()
        print(f"Artist: {self.artist}")
        print(f"Concert Type: {self.concert_type}")


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: {self.customer_name}")
        print(f"Email: {self.email}")
        print(f"Phone: {self.phone}")


class Booking:
    booking_counter = 0

    def __init__(self, event, num_tickets, customer):
        Booking.booking_counter += 1
        self.booking_id = Booking.booking_counter
        self.event = event
```

```
        self.num_tickets = num_tickets
        self.customer = customer
        self.total_cost = event.ticket_price * num_tickets
        self.booking_status = "Confirmed"

    def display_booking_details(self):
        print(f"\nBooking ID: {self.booking_id}")
        self.event.display_event_details()
        self.customer.display_customer_details()
        print(f'Number of Tickets: {self.num_tickets}")
        print(f'Total Cost: ${self.total_cost}")
        print(f'Status: {self.booking_status}")

class IEventServiceProvider(ABC):
    @abstractmethod
    def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue):
        pass

    @abstractmethod
    def get_event_details(self):
        pass

    @abstractmethod
    def get_available_no_of_tickets(self):
        pass


class IBookingSystemServiceProvider(ABC):
    @abstractmethod
    def calculate_booking_cost(self, num_tickets,event):
        pass

    @abstractmethod
    def book_tickets(self, event_name, num_tickets, array_of_customers):
        pass

    @abstractmethod
    def cancel_booking(self, booking_id):
        pass

    @abstractmethod
    def get_booking_details(self, booking_id):
        pass


class EventServiceProviderImpl(IEventServiceProvider):
```

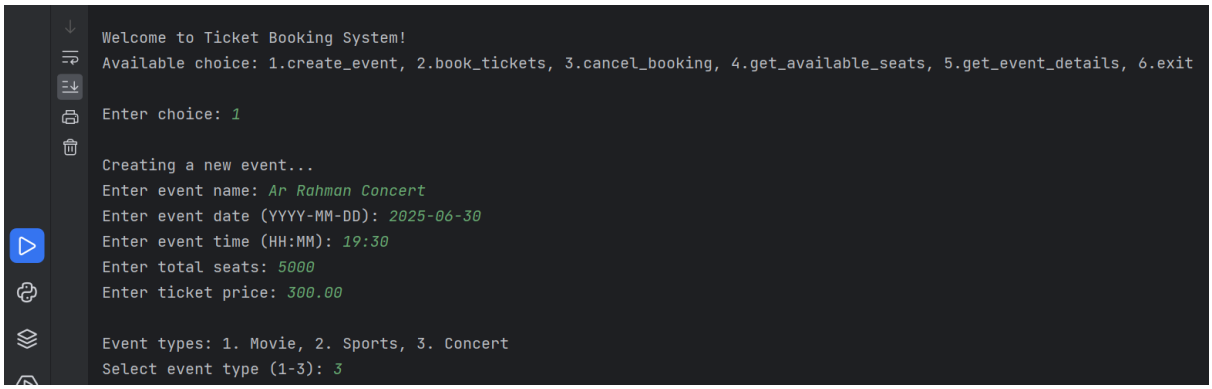
```
    def __init__(self):
        self.events = []

    def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue):
        if event_type == EventType.MOVIE:
            genre = input("Enter movie genre: ")
            actor = input("Enter actor: ")
            actress = input("Enter actress: ")
            event = MovieEvent(event_name, date, time, venue, total_seats, ticket_price, genre,
actor, actress)
        elif event_type == EventType.SPORTS:
            sport_name = input("Enter sport name: ")
            teams_name = input("Enter teams: ")
            event = SportsEvent(event_name, date, time, venue, total_seats, ticket_price,
sport_name, teams_name)
        elif event_type == EventType.CONCERT:
            artist = input("Enter artist name: ")
            concert_type = input("Enter concert type: ")
            event = ConcertEvent(event_name, date, time, venue, total_seats, ticket_price, artist,
concert_type)
        else:
            event = Event(event_name, date, time, venue, total_seats, ticket_price, event_type)

        self.events.append(event)
        return event

    def get_event_details(self):
        return [event.display_event_details() for event in self.events]

    def get_available_no_of_tickets(self):
        return {event.event_name: event.available_seats for event in self.events}

    def find_event(self, event_name):
        for event in self.events:
            if event.event_name.lower() == event_name.lower():
                return event
        return None


class BookingSystemServiceProviderImpl(EventServiceProviderImpl,
IBookingSystemServiceProvider):
    def __init__(self):
        super().__init__()
        self.bookings = []

    def calculate_booking_cost(self, num_tickets, event):
```

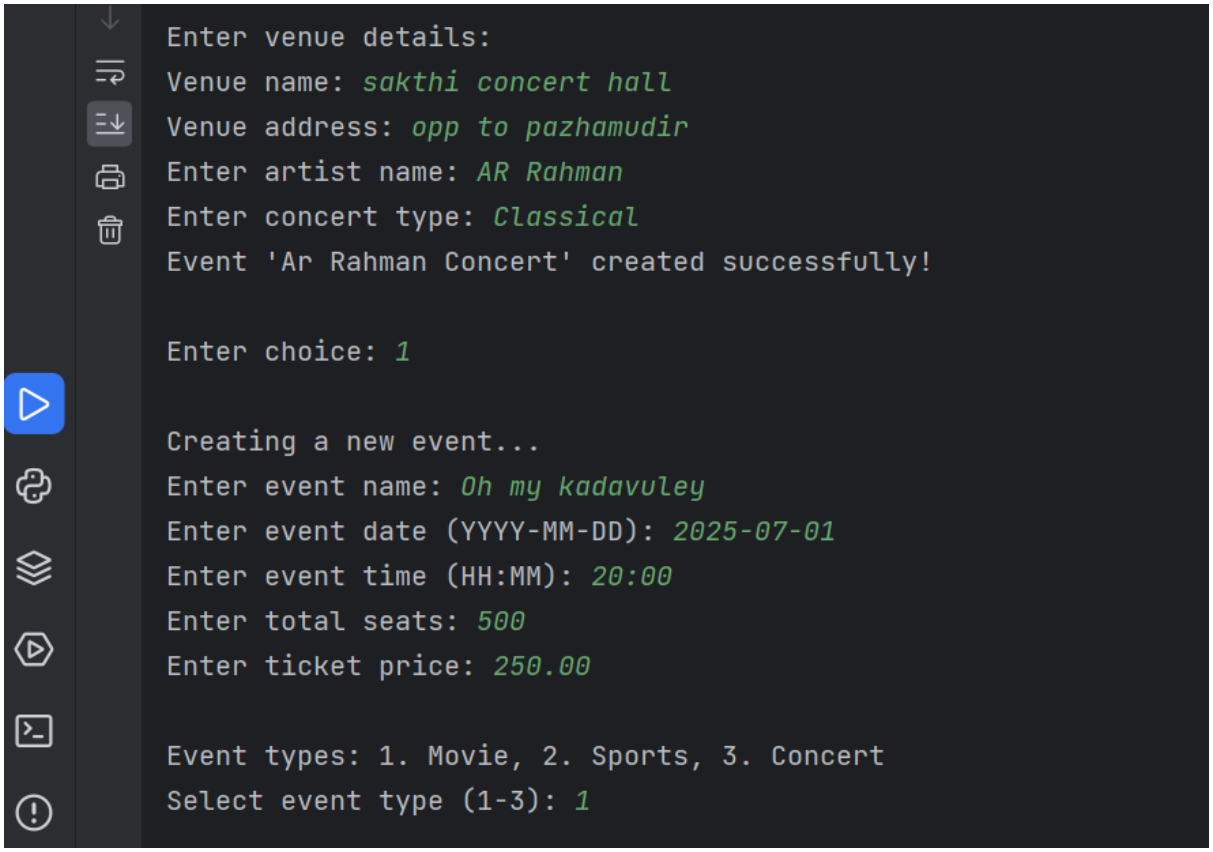
```
        return num_tickets * event.ticket_price

    def book_tickets(self, event_name, num_tickets, array_of_customers):
        event = self.find_event(event_name)
        if not event:
            print(f"Event '{event_name}' not found!")
            return None

        if event.available_seats < num_tickets:
            print(f"Not enough seats available! Only {event.available_seats} left.")
            return None

        bookings = []
        for customer in array_of_customers:
            booking = Booking(event, num_tickets, customer)
            event.available_seats -= num_tickets
            self.bookings.append(booking)
            bookings.append(booking)
            print(f"Booking successful! Booking ID: {booking.booking_id}")

        return bookings

    def cancel_booking(self, booking_id):
        booking = next((b for b in self.bookings if b.booking_id == booking_id), None)
        if not booking:
            print(f"Booking ID {booking_id} not found!")
            return False

        booking.event.available_seats += booking.num_tickets
        booking.booking_status = "Cancelled"
        print(f"Booking {booking_id} has been cancelled.")
        return True

    def get_booking_details(self, booking_id):
        booking = next((b for b in self.bookings if b.booking_id == booking_id), None)
        if booking:
            booking.display_booking_details()
        else:
            print(f"Booking ID {booking_id} not found!")
        return booking

    def find_booking(self, booking_id):
        return next((b for b in self.bookings if b.booking_id == booking_id), None)


class TicketBookingSystem:
```

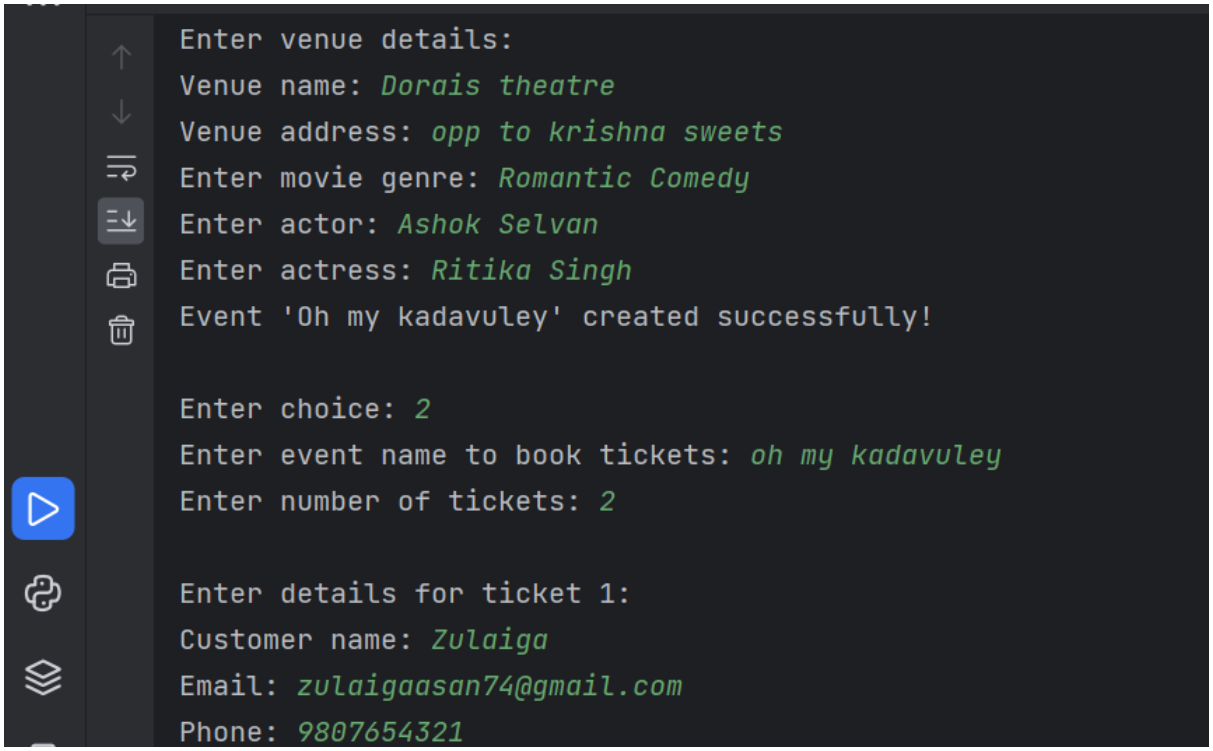
```
    def __init__(self):
        self.booking_system = BookingSystemServiceProviderImpl()

    def run(self):
        print("\nWelcome to Ticket Booking System!")
        print(
            "Available choice: 1.create_event, 2.book_tickets, 3.cancel_booking,
4.get_available_seats, 5.get_event_details, 6.exit")

        while True:
            choice = int(input("\nEnter choice: "))

            if choice == 6:
                print("Thank you for using Ticket Booking System!")
                break

            elif choice == 1:
                self.handle_create_event()

            elif choice == 2:
                self.handle_book_tickets()

            elif choice == 3:
                self.handle_cancel_booking()

            elif choice == 4:
                self.handle_get_available_seats()

            elif choice == 5:
                self.handle_get_event_details()

            else:
                print("Invalid choice! Please try again.")

    def handle_create_event(self):
        print("\nCreating a new event...")
        event_name = input("Enter event name: ")
        event_date = input("Enter event date (YYYY-MM-DD): ")
        event_time = input("Enter event time (HH:MM): ")
        total_seats = int(input("Enter total seats: "))
        ticket_price = float(input("Enter ticket price: "))

        print("\nEvent types: 1. Movie, 2. Sports, 3. Concert")
        event_type_choice = int(input("Select event type (1-3): "))
        event_type = list(EventType)[event_type_choice - 1]
```

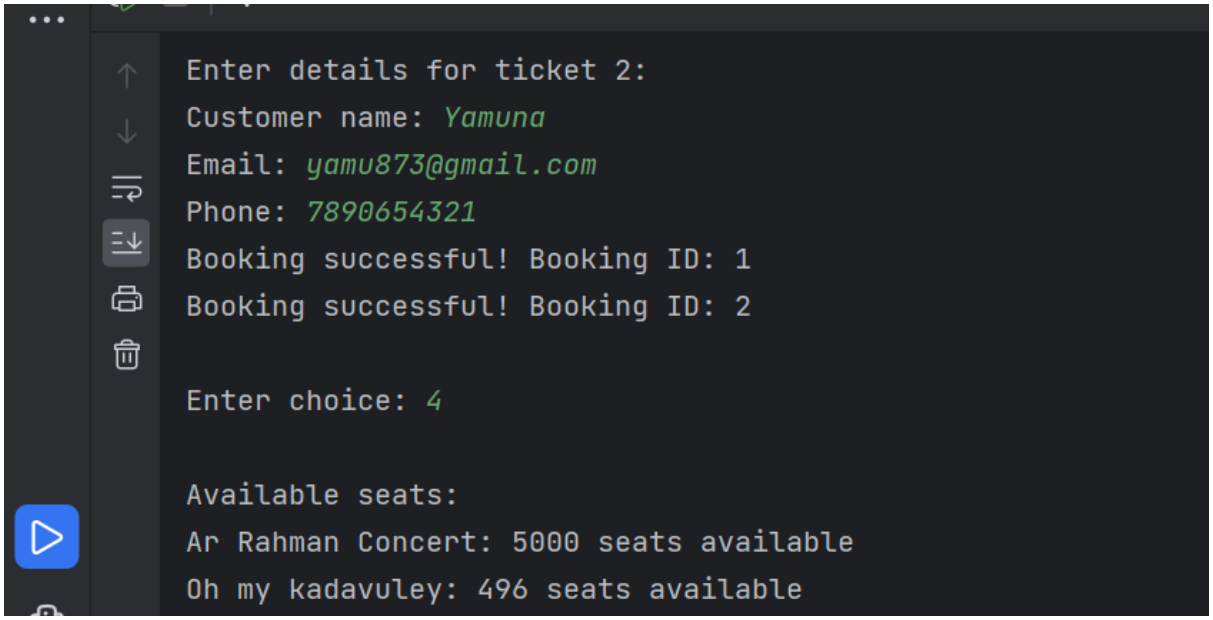
```
        print("\nEnter venue details:")
        venue_name = input("Venue name: ")
        address = input("Venue address: ")
        venue = Venue(venue_name, address)

        self.booking_system.create_event(
            event_name, event_date, event_time,
            total_seats, ticket_price, event_type, venue
        )
        print(f"Event '{event_name}' created successfully!")

    def handle_book_tickets(self):
        event_name = input("Enter event name to book tickets: ")
        num_tickets = int(input("Enter number of tickets: "))

        customers = []
        for i in range(num_tickets):
            print(f"\nEnter details for ticket {i + 1}:")
            name = input("Customer name: ")
            email = input("Email: ")
            phone = input("Phone: ")
            customers.append(Customer(name, email, phone))

        self.booking_system.book_tickets(event_name, num_tickets, customers)

    def handle_cancel_booking(self):
        booking_id = int(input("Enter booking ID to cancel: "))
        self.booking_system.cancel_booking(booking_id)

    def handle_get_available_seats(self):
        available_seats = self.booking_system.get_available_no_of_tickets()
        print("\nAvailable seats:")
        for event_name, seats in available_seats.items():
            print(f"{event_name}: {seats} seats available")

    def handle_get_event_details(self):
        print("\nEvent Details:")
        self.booking_system.get_event_details()


# Main execution
if __name__ == "__main__":
    system = TicketBookingSystem()
    system.run()
```

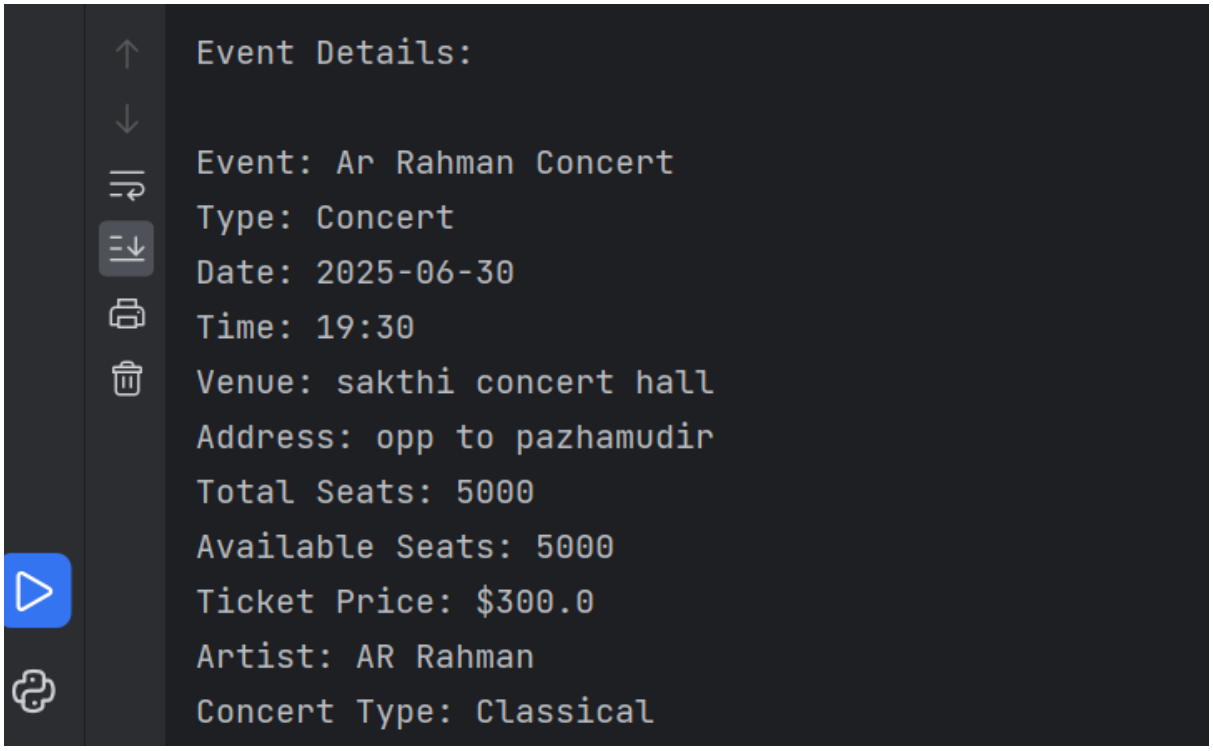
OUTPUT :

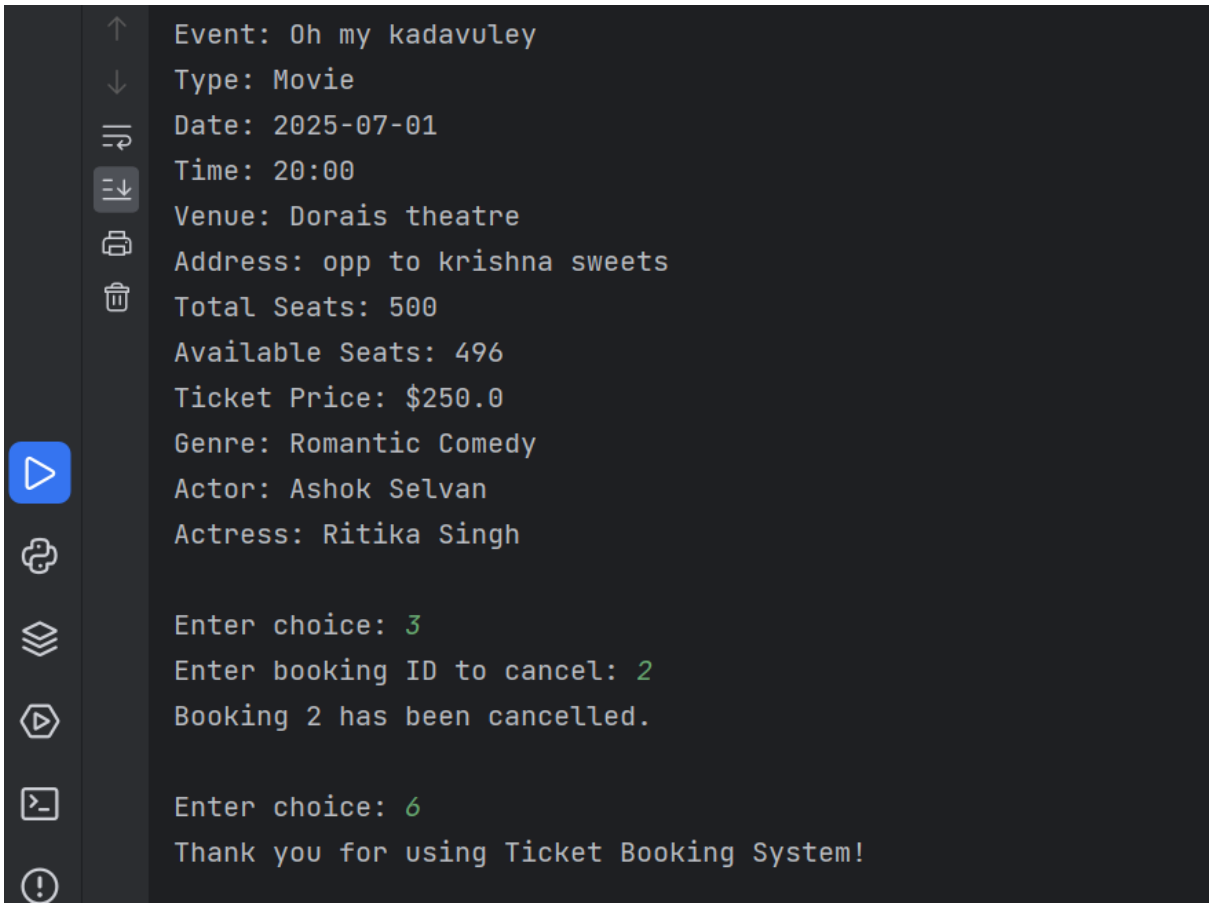

Welcome to Ticket Booking System!
Available choice: 1.create_event, 2.book_tickets, 3.cancel_booking, 4.get_available_seats, 5.get_event_details, 6.exit
Enter choice: 1
Creating a new event...
Enter event name: Ar Rahman Concert
Enter event date (YYYY-MM-DD): 2025-06-30
Enter event time (HH:MM): 19:30
Enter total seats: 5000
Enter ticket price: 300.00
Event types: 1. Movie, 2. Sports, 3. Concert
Select event type (1-3): 3


Enter venue details:
Venue name: sakthi concert hall
Venue address: opp to pazhamudir
Enter artist name: AR Rahman
Enter concert type: Classical
Event 'Ar Rahman Concert' created successfully!
Enter choice: 1
Creating a new event...
Enter event name: Oh my kadavuley
Enter event date (YYYY-MM-DD): 2025-07-01
Enter event time (HH:MM): 20:00
Enter total seats: 500
Enter ticket price: 250.00
Event types: 1. Movie, 2. Sports, 3. Concert
Select event type (1-3): 1


Enter venue details:
Venue name: Dorais theatre
Venue address: opp to krishna sweets
Enter movie genre: Romantic Comedy
Enter actor: Ashok Selvan
Enter actress: Ritika Singh
Event 'Oh my kadavuley' created successfully!
Enter choice: 2
Enter event name to book tickets: oh my kadavuley
Enter number of tickets: 2
Enter details for ticket 1:
Customer name: Zulaiga
Email: zulaigaasan74@gmail.com
Phone: 9807654321


Enter details for ticket 2:
Customer name: Yamuna
Email: yamu873@gmail.com
Phone: 7890654321
Booking successful! Booking ID: 1
Booking successful! Booking ID: 2
Enter choice: 4
Available seats:
Ar Rahman Concert: 5000 seats available
Oh my kadavuley: 496 seats available


Event Details:
Event: Ar Rahman Concert
Type: Concert
Date: 2025-06-30
Time: 19:30
Venue: sakthi concert hall
Address: opp to pazhamudir
Total Seats: 5000
Available Seats: 5000
Ticket Price: $300.0
Artist: AR Rahman
Concert Type: Classical


Event: Oh my kadavuley
Type: Movie
Date: 2025-07-01
Time: 20:00
Venue: Dorais theatre
Address: opp to krishna sweets
Total Seats: 500
Available Seats: 496
Ticket Price: $250.0
Genre: Romantic Comedy
Actor: Ashok Selvan
Actress: Ritika Singh
Enter choice: 3
Enter booking ID to cancel: 2
Booking 2 has been cancelled.
Enter choice: 6
Thank you for using Ticket Booking System!

TASK 9 :

Task 9: Exception Handling
throw the exception whenever needed and Handle in main method,

1. **EventNotFoundException** throw this exception when user try to book the tickets for Event not listed in the menu.
2. **InvalidBookingIDException** throw this exception when user entered the invalid bookingId when he tries to view the booking or cancel the booking.
3. **NullPointerException** handle in main method**.**

Throw these exceptions from the methods in **TicketBookingSystem** class. Make necessary changes to accommodate exception in the source code. Handle all these exceptions from the main program.

CODE :

```
from abc import ABC, abstractmethod
from exception import (EventNotFoundException,InvalidBookingIDException)


class Event(ABC):
    def __init__(self,event_name,date,time,total_seats,ticket_price,venue_name,event_type):
        self.event_name = event_name
        self.date = date
        self.time = time
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.venue_name = venue_name
        self.event_type = event_type

    @abstractmethod
    def display_event_details(self):
        pass

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

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

class Movie(Event):

    def
__init__(self,event_name,date,time,total_seats,ticket_price,venue_name,actor,actress,genre):
```

```
        super().__init__(event_name,date,time,total_seats,ticket_price,venue_name,"Movie")
        self.actor = actor
        self.actress = actress
        self.genre = genre

    def display_event_details(self):
        print("Displaying Movie Details")
        print(f'Event_name:{self.event_name}")
        print(f'Genre:{self.genre}")
        print(f'Actor:{self.actor} Actress:{self.actress}")
        print(f'Date:{self.date}")
        print(f'Time:{self.time}")
        print(f'Total_seats:{self.total_seats}")
        print(f'Ticket_price:₹{self.ticket_price}")
        print(f'Venue_name:{self.venue_name}")

class Concert(Event):

    def
__init__(self,event_name,date,time,total_seats,ticket_price,venue_name,artist,concert_type):
        super().__init__(event_name,date,time,total_seats,ticket_price,venue_name,"Concert")

        self.artist = artist
        self.concert_type = concert_type

    def display_event_details(self):
        print("Displaying Concert Details")
        print(f'Event_name:{self.event_name}")
        print(f'Artist:{self.artist}")
        print(f'Concert_type:{self.concert_type}")
        print(f'Date:{self.date}")
        print(f'Time:{self.time}")
        print(f'Total_seats:{self.total_seats}")
        print(f'Ticket_price:₹{self.ticket_price}")
        print(f'Venue_name:{self.venue_name}")

class Sports(Event):

    def __init__(self, event_name, date, time, total_seats, ticket_price, venue_name,
sport_name, teams):
        super().__init__(event_name, date, time, total_seats, ticket_price, venue_name,
"Sports")

        self.sport_name = sport_name
        self.teams = teams
```

```
    def display_event_details(self):
        print("Displaying Concert Details")
        print(f"Event_name:{self.event_name}")
        print(f"Sport_Name:{self.sport_name}")
        print(f"Teams:{self.teams}")
        print(f"Date:{self.date}")
        print(f"Time:{self.time}")
        print(f"Total_seats:{self.total_seats}")
        print(f"Ticket_price:₹{self.ticket_price}")
        print(f"Venue_name:{self.venue_name}")


class Bookingsystem(ABC):

    @abstractmethod
    def create_event(self,event_type,**kwargs):
        pass

    @abstractmethod
    def book_tickets(self,event,num_tickets):
        pass

    @abstractmethod
    def cancel_tickets(self,event,num_tickets):
        pass

    @abstractmethod
    def display_event_details(self,event):
        pass

    @abstractmethod
    def main(self):
        pass

class TicketBookingSystem(Bookingsystem):

    def __init__(self):
        self.events = []
        self.bookings = {}
        self.booking_count = 1


    def create_event(self, event_type, **kwargs):
        if event_type == "Movie":
            event = Movie(**kwargs)
        elif event_type == "Concert":
```

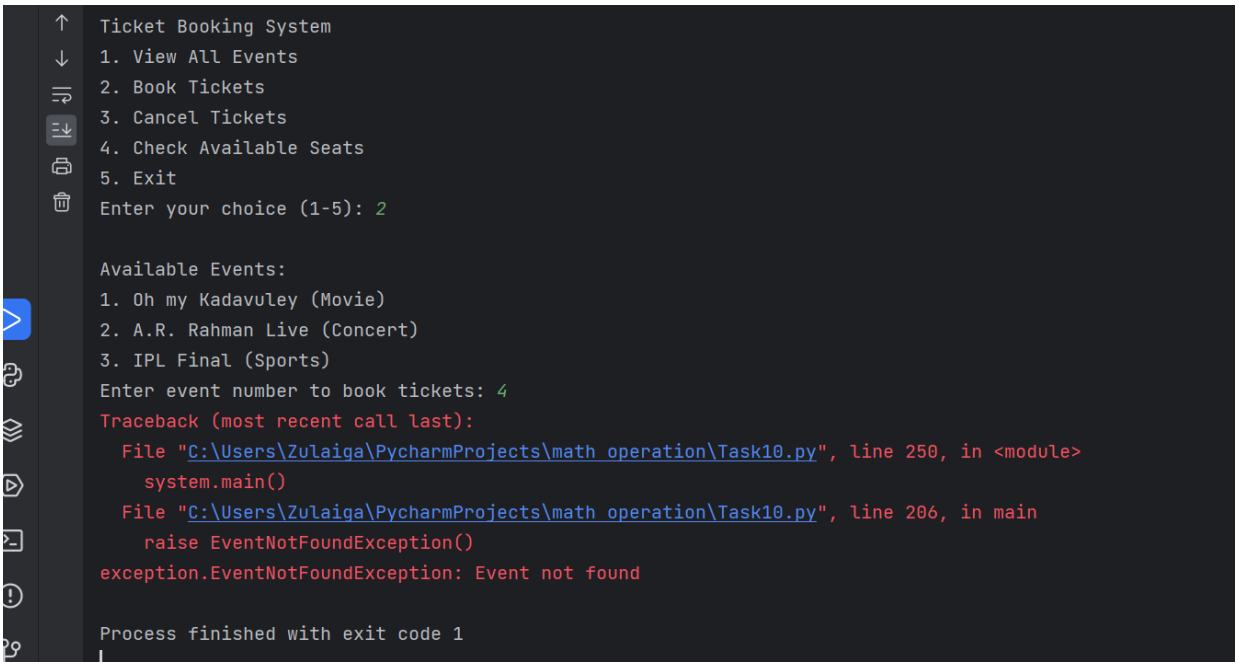
```
            event = Concert(**kwargs)
        elif event_type == "Sports":
            event = Sports(**kwargs)
        else:
            raise ValueError("Invalid event type")

        self.events.append(event)
        return event

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

    def book_tickets(self, event, num_tickets):
        if event.available_seats >= num_tickets:
            event.available_seats -= num_tickets
            booking_id = self.booking_count
            self.bookings[booking_id]={
            'event': event,
            'num_tickets': num_tickets,
            'total_cost': num_tickets * event.ticket_price
        }
            self.booking_count+=1
            print(f"Booking successful,Booking ID:{booking_id}")
            return booking_id, num_tickets*event.ticket_price
        else:
            print(f"Event is sold out! Only {event.available_seats} seats available.")
            return 0

    def cancel_tickets(self, booking_id, num_tickets=None):
        if booking_id not in self.bookings:
            raise InvalidBookingIDException(booking_id)
        booking = self.bookings[booking_id]
        event = booking['event']

        if num_tickets is None:
            num_tickets = booking['num_tickets']

        event.available_seats += num_tickets

        if num_tickets == booking['num_tickets']:
            del self.bookings[booking_id]
        else:
            booking['num_tickets'] -= num_tickets
            booking['total_cost'] = booking['num_tickets'] * event.ticket_price

        print(f"Cancelled {num_tickets} tickets for {event.event_name} (Booking ID:
```

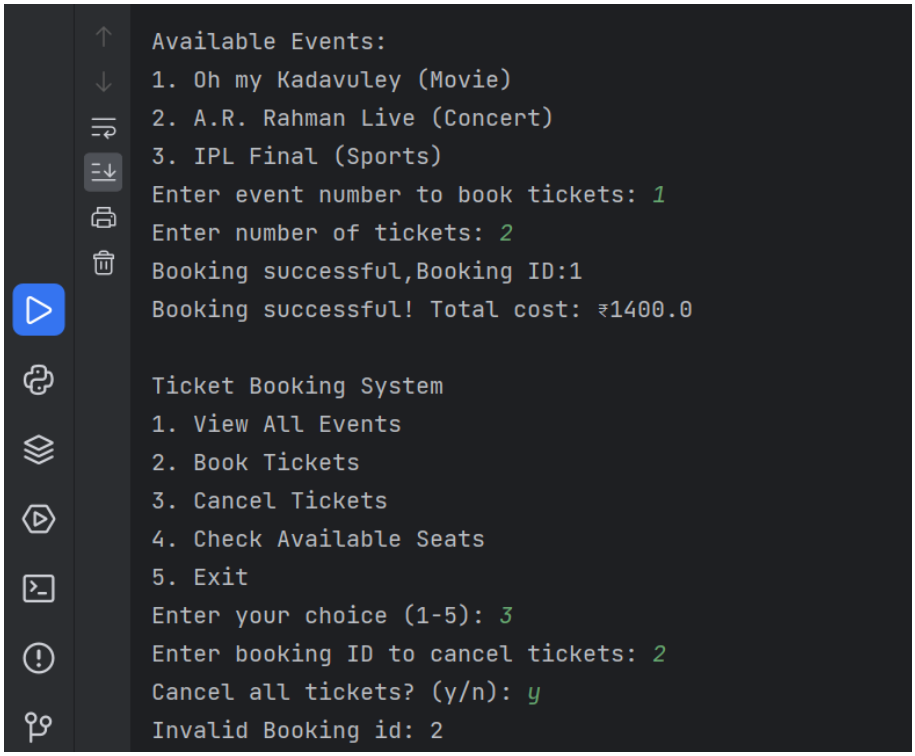
```
{booking_id})")

    def get_available_seats(self, event):
        return event.available_seats

    def main(self):

        movie = self.create_event("Movie",
            event_name="Oh my Kadavuley",
            date="2025-08-15",
            time="18:00",
            total_seats=200,
            ticket_price=700.00,
            venue_name="Grand Theater",
            actor="Ashok Selvan",
            actress="Ritika Singh",
            genre="Romantic comedy",
        )

        concert = self.create_event( "Concert",
            event_name="A.R. Rahman Live",
            date="2025-07-15",
            time="20:00",
            total_seats=10000,
            ticket_price=2500,
            venue_name="Arena",
            artist="A.R. Rahman",
            concert_type="World Music"
        )

        sports = self.create_event("Sports",
            event_name="IPL Final",
            date="2025-06-30",
            time="19:30",
            total_seats=50000,
            ticket_price=1500,
            venue_name="Stadium",
            sport_name="Cricket",
            teams="India vs Pakistan"
        )

        while True:
            print("\nTicket Booking System")
            print("1. View All Events")
            print("2. Book Tickets")
            print("3. Cancel Tickets")
```

```
print("4. Check Available Seats")
print("5. Exit")

choice = input("Enter your choice (1-5): ")

if choice == "1":
    print("\nAvailable Events:")
    for i, event in enumerate(self.events, 1):
        print(f"{i}. {event.event_name} ({event.event_type})")

    event_choice = input("Enter event number to view details: ")
    self.display_event_details(self.events[int(event_choice)-1])

elif choice == "2":
    print("\nAvailable Events:")
    for i, event in enumerate(self.events, 1):
        print(f"{i}. {event.event_name} ({event.event_type})")

    event_choice = int(input("Enter event number to book tickets: "))
    if event_choice >= len(self.events):
        raise EventNotFoundException()
    event = self.events[int(event_choice)-1]

    try:
            num_tickets = int(input("Enter number of tickets: "))
            booking_id,total_cost = self.book_tickets(event, num_tickets)
            if total_cost > 0:
                print(f"Booking successful! Total cost: ₹{total_cost}")
    except ValueError:
            print("Please enter a valid number")

elif choice == "3":
    try:
        booking_id = int(input("Enter booking ID to cancel tickets: "))
        cancel_option = input("Cancel all tickets? (y/n): ").lower()
        if cancel_option == 'y':
            self.cancel_tickets(booking_id)
        else:
            try:
                num_tickets = int(input("Enter number of tickets to cancel: "))
                self.cancel_tickets(booking_id, num_tickets)
            except ValueError:
                print("Please enter a valid number")
    except InvalidBookingIDException as e:
        print(f"Invalid Booking id: {e}")
    except ValueError:
```

```
            print("Please enter a valid booking ID")

        elif choice == "4":
            print("\nAvailable Events:")
            for i, event in enumerate(self.events, 1):
                print(f"{i}. {event.event_name} ({event.event_type})")

            event_choice = input("Enter event number to check seats: ")
            event = self.events[int(event_choice)-1]
            print(f"Available seats for {event.event_name}: {self.get_available_seats(event)}")

        elif choice == "5":
            print("Thank you for using the Ticket Booking System!")
            break

        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    system = TicketBookingSystem()
    system.main()
```

OUTPUT :





TASK 10:

Task 10: Collection

1. From the previous task change the **Booking** class attribute customers to List of customers and **BookingSystem** class attribute events to List of events and perform the same operation.
2. From the previous task change all list type of attribute to type Set in **Booking** and **BookingSystem** class and perform the same operation.
   - Avoid adding duplicate Account object to the set.
   - Create Comparator<Event> object to sort the event based on event name and location in alphabetical order.
3. From the previous task change all list type of attribute to type Map object in **Booking** and **BookingSystem** class and perform the same operation.

CODE :

List implementation:

```
from abc import ABC, abstractmethod
from datetime import datetime
from enum import Enum


class EventType(Enum):
```

```
    MOVIE = "Movie"
    SPORTS = "Sports"
    CONCERT = "Concert"


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: {self.address}")


class Event(ABC):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, event_type=None):
        self.event_name = event_name
        self.event_date = event_date
        self.event_time = event_time
        self.venue = venue
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.event_type = event_type

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

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

    def book_tickets(self, num_tickets):
        if self.available_seats >= num_tickets:
            self.available_seats -= num_tickets
            return True
        return False

    def cancel_booking(self, num_tickets):
        if (self.available_seats + num_tickets) <= self.total_seats:
            self.available_seats += num_tickets
            return True
        return False

    @abstractmethod
```

```
    def display_event_details(self):
        pass


class Movie(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, genre="", actor="", actress=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.MOVIE)
        self.genre = genre
        self.actor = actor
        self.actress = actress

    def display_event_details(self):
        print("\nMovie Details:")
        print(f'Event Name: {self.event_name}")
        print(f'Date: {self.event_date}")
        print(f'Time: {self.event_time}")
        print(f'Genre: {self.genre}")
        print(f'Actor: {self.actor} Actress: {self.actress}")
        print(f'Venue: {self.venue.venue_name}")
        print(f'Total Seats: {self.total_seats}")
        print(f'Available Seats: {self.available_seats}")
        print(f'Ticket Price: {self.ticket_price}")


class Concert(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, artist="", concert_type=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.CONCERT)
        self.artist = artist
        self.concert_type = concert_type

    def display_event_details(self):
        print("\nConcert Details:")
        print(f'Event Name: {self.event_name}")
        print(f'Date: {self.event_date}")
        print(f'Time: {self.event_time}")
        print(f'Artist: {self.artist}")
        print(f'Concert Type: {self.concert_type}")
        print(f'Venue: {self.venue.venue_name}")
        print(f'Total Seats: {self.total_seats}")
        print(f'Available Seats: {self.available_seats}")
        print(f'Ticket Price: {self.ticket_price}")
```

```
class Sports(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, sport_name="", teams=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.SPORTS)
        self.sport_name = sport_name
        self.teams = teams

    def display_event_details(self):
        print("\nSports Event Details:")
        print(f"Event Name: {self.event_name}")
        print(f"Date: {self.event_date}")
        print(f"Time: {self.event_time}")
        print(f"Sport: {self.sport_name}")
        print(f"Teams: {self.teams}")
        print(f"Venue: {self.venue.venue_name}")
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: {self.ticket_price:}")


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

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


class Booking:
    booking_count = 1

    def __init__(self, customers=None, event=None, num_tickets=0, total_cost=0.0):
        self.booking_id = Booking.booking_count
        Booking.booking_count += 1
        self.customers = customers if customers else []
        self.event = event
        self.num_tickets = num_tickets
        self.total_cost = total_cost
        self.booking_date = datetime.now()
```

```
    def display_booking_details(self):
        print(f"Booking ID: {self.booking_id}")
        print(f"Booking Date: {self.booking_date}")
        print(f"Event: {self.event.event_name}")
        print(f"Number of Tickets: {self.num_tickets}")
        print(f"Total Cost: {self.total_cost}")
        print("Customers:")
        for customer in self.customers:
            customer.display_customer_details()


class BookingSystem:
    def __init__(self):
        self.events = [] ---------- List Implementation
        self.bookings = []

    def create_event(self, event_name, date, time, total_seats, ticket_price,
                     event_type, venue, **kwargs):
        event = None
        if event_type == EventType.MOVIE:
            event = Movie(event_name, date, time, venue, total_seats, ticket_price,
                          kwargs.get('genre'), kwargs.get('actor'), kwargs.get('actress'))
        elif event_type == EventType.CONCERT:
            event = Concert(event_name, date, time, venue, total_seats, ticket_price,
                            kwargs.get('artist'), kwargs.get('concert_type'))
        elif event_type == EventType.SPORTS:
            event = Sports(event_name, date, time, venue, total_seats, ticket_price,
                           kwargs.get('sport_name'), kwargs.get('teams'))

        if event:
            self.events.append(event)
        return event

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

    def book_tickets(self, event_name, num_tickets, customers):
        event = None
        for e in self.events:
            if e.event_name.lower() == event_name.lower():
                event = e
                break
        if not event:
            print("Event not found!")
            return None
```

```
    if len(customers) != num_tickets:
        print("Number of customers must match number of tickets!")
        return None

    if event.book_tickets(num_tickets):
        total_cost = self.calculate_booking_cost(num_tickets, event)
        booking = Booking(customers, event, num_tickets, total_cost)
        self.bookings.append(booking)
        return booking
    else:
        print("Not enough seats available!")
        return None

def cancel_booking(self, booking_id):
    booking = None
    for b in self.bookings:
        if b.booking_id == booking_id:
            booking = b
            break
    if not booking:
        print("Booking not found!")
        return False

    event = booking.event
    if event.cancel_booking(booking.num_tickets):
        self.bookings.remove(booking)
        print(f"Booking {booking_id} cancelled successfully.")
        return True
    else:
        print("Failed to cancel booking.")
        return False

def get_available_no_of_tickets(self, event_name):
    event = None
    for e in self.events:
        if e.event_name.lower() == event_name.lower():
            event = e
            break
    return event.available_seats if event else 0

def get_event_details(self, event_name):
    event = None
    for e in self.events:
        if e.event_name.lower() == event_name.lower():
            event = e
            break
```

```
    if event:
        event.display_event_details()
    else:
        print("Event not found!")

def main(self):

    venue1 = Venue("Dorais Theatre", "Opp to Pazhamudir")
    venue2 = Venue("Sakthi concert hall", "Market Road")
    venue3 = Venue("Sports Arena", "789 Stadium Rd")

    movie = self.create_event(
        "Oh my Kadavuley", "2025-06-25", "18:00", 200, 250.00,
        EventType.MOVIE, venue1, genre="Romantic Comedy", actor="Ashok Selvan",
        actress="Ritika Singh"
    )

    concert = self.create_event(
        "AR Rahman Concert", "2025-07-01", "20:00", 5000, 500.00,
        EventType.CONCERT, venue2, artist="AR Rahman",
        concert_type="Classical"
    )

    sports = self.create_event(
        "IPL", "2025-06-18", "15:30", 30000, 750.00,
        EventType.SPORTS, venue3, sport_name="Cricket",
        teams="CSK vs RCB"
    )

    while True:
        print("\nTicket Booking System")
        print("1. View All Events")
        print("2. Book Tickets")
        print("3. Cancel Booking")
        print("4. Check Available Seats")
        print("5. View Event Details")
        print("6. Exit")

        choice = input("Enter your choice (1-6): ")

        if choice == "1":
            print("\nAvailable Events:")
            for i, event in enumerate(self.events, 1):
                print(f"{i}. {event.event_name} ({event.event_type.value})")

        elif choice == "2":
```

```
            event_name = input("Enter event name: ")
            try:
                num_tickets = int(input("Enter number of tickets: "))
                customers = []
                for i in range(num_tickets):
                    print(f"\nEnter details for customer {i + 1}:")
                    name = input("Name: ")
                    email = input("Email: ")
                    phone = input("Phone: ")
                    customers.append(Customer(name, email, phone))

                booking = self.book_tickets(event_name, num_tickets, customers)
                if booking:
                    print("\nBooking successful!")
                    booking.display_booking_details()
            except ValueError:
                print("Please enter a valid number")

        elif choice == "3":
            try:
                booking_id = int(input("Enter booking ID to cancel: "))
                self.cancel_booking(booking_id)
            except ValueError:
                print("Please enter a valid booking ID")

        elif choice == "4":
            event_name = input("Enter event name: ")
            available = self.get_available_no_of_tickets(event_name)
            print(f"Available seats for {event_name}: {available}")

        elif choice == "5":
            event_name = input("Enter event name: ")
            self.get_event_details(event_name)

        elif choice == "6":
            print("Thank you for using the Ticket Booking System!")
            break

        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    system = BookingSystem()
    system.main()
```

Dictionary Implementation:

```
from abc import ABC, abstractmethod
from datetime import datetime
from enum import Enum


class EventType(Enum):
    MOVIE = "Movie"
    SPORTS = "Sports"
    CONCERT = "Concert"


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: {self.address}")


class Event(ABC):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, event_type=None):
        self.event_name = event_name
        self.event_date = event_date
        self.event_time = event_time
        self.venue = venue
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.event_type = event_type

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

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

    def book_tickets(self, num_tickets):
        if self.available_seats >= num_tickets:
            self.available_seats -= num_tickets
            return True
```

```
        return False

    def cancel_booking(self, num_tickets):
        if (self.available_seats + num_tickets) <= self.total_seats:
            self.available_seats += num_tickets
            return True
        return False

    @abstractmethod
    def display_event_details(self):
        pass


class Movie(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, genre="", actor="", actress=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.MOVIE)
        self.genre = genre
        self.actor = actor
        self.actress = actress

    def display_event_details(self):
        print("\nMovie Details:")
        print(f'Event Name: {self.event_name}")
        print(f'Date: {self.event_date}")
        print(f'Time: {self.event_time}")
        print(f'Genre: {self.genre}")
        print(f'Actor: {self.actor} Actress: {self.actress}")
        print(f'Venue: {self.venue.venue_name}")
        print(f'Total Seats: {self.total_seats}")
        print(f'Available Seats: {self.available_seats}")
        print(f'Ticket Price: {self.ticket_price}")


class Concert(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, artist="", concert_type=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.CONCERT)
        self.artist = artist
        self.concert_type = concert_type

    def display_event_details(self):
        print("\nConcert Details:")
        print(f'Event Name: {self.event_name}")
```

```
        print(f"Date: {self.event_date}")
        print(f"Time: {self.event_time}")
        print(f"Artist: {self.artist}")
        print(f"Concert Type: {self.concert_type}")
        print(f"Venue: {self.venue.venue_name}")
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: {self.ticket_price}")


class Sports(Event):
    def __init__(self, event_name="", event_date="", event_time="", venue=None,
                 total_seats=0, ticket_price=0.0, sport_name="", teams=""):
        super().__init__(event_name, event_date, event_time, venue, total_seats,
                         ticket_price, EventType.SPORTS)
        self.sport_name = sport_name
        self.teams = teams

    def display_event_details(self):
        print("\nSports Event Details:")
        print(f"Event Name: {self.event_name}")
        print(f"Date: {self.event_date}")
        print(f"Time: {self.event_time}")
        print(f"Sport: {self.sport_name}")
        print(f"Teams: {self.teams}")
        print(f"Venue: {self.venue.venue_name}")
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: {self.ticket_price:}")


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

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


class Booking:
    booking_count = 1
```

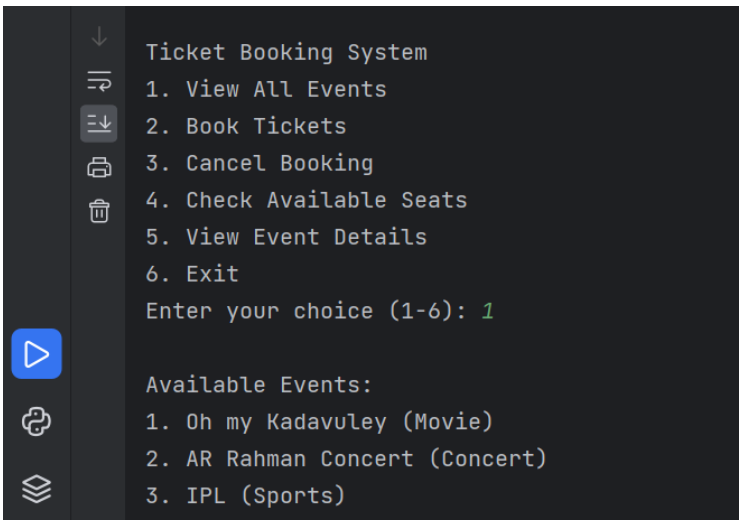
```
    def __init__(self, customers=None, event=None, num_tickets=0, total_cost=0.0):
        self.booking_id = Booking.booking_count
        Booking.booking_count += 1
        self.customers = customers if customers else {}
        self.event = event
        self.num_tickets = num_tickets
        self.total_cost = total_cost
        self.booking_date = datetime.now()

    def display_booking_details(self):
        print(f"Booking ID: {self.booking_id}")
        print(f"Booking Date: {self.booking_date}")
        print(f"Event: {self.event.event_name}")
        print(f"Number of Tickets: {self.num_tickets}")
        print(f"Total Cost: {self.total_cost}")
        print("Customers:")
        for customer in self.customers:
            customer.display_customer_details()


class BookingSystem:
    def __init__(self):
        self.events = {}---------- Dictionary Implementation
        self.bookings = []

    def create_event(self, event_name, date, time, total_seats, ticket_price,
                     event_type, venue, **kwargs):
        event = None
        if event_type == EventType.MOVIE:
            event = Movie(event_name, date, time, venue, total_seats, ticket_price,
                          kwargs.get('genre'), kwargs.get('actor'), kwargs.get('actress'))
        elif event_type == EventType.CONCERT:
            event = Concert(event_name, date, time, venue, total_seats, ticket_price,
                            kwargs.get('artist'), kwargs.get('concert_type'))
        elif event_type == EventType.SPORTS:
            event = Sports(event_name, date, time, venue, total_seats, ticket_price,
                           kwargs.get('sport_name'), kwargs.get('teams'))

        if event:
            self.events[event_name] = event
        return event

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

    def book_tickets(self, event_name, num_tickets, customers):
```

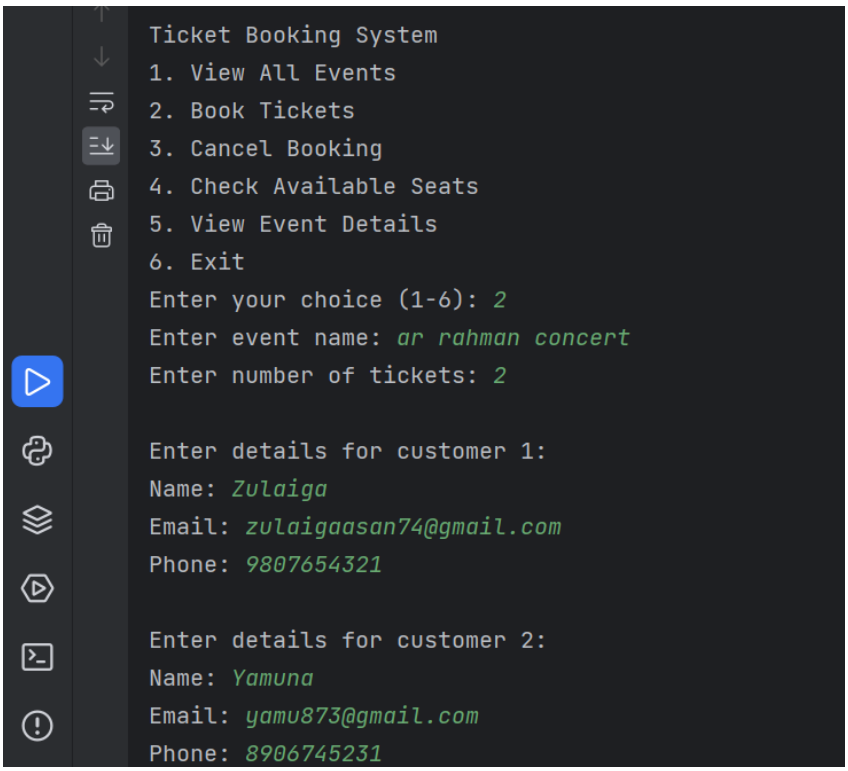
```
        event = self.events.get(event_name)
        if not event:
            print("Event not found!")
            return None

        if len(customers) != num_tickets:
            print("Number of customers must match number of tickets!")
            return None

        if event.book_tickets(num_tickets):
            total_cost = self.calculate_booking_cost(num_tickets, event)
            booking = Booking(customers, event, num_tickets, total_cost)
            self.bookings.append(booking)
            return booking
        else:
            print("Not enough seats available!")
            return None

    def cancel_booking(self, booking_id):
        booking = None
        for b in self.bookings:
            if b.booking_id == booking_id:
                booking = b
                break
        if not booking:
            print("Booking not found!")
            return False

        event = booking.event
        if event.cancel_booking(booking.num_tickets):
            self.bookings.remove(booking)
            print(f"Booking {booking_id} cancelled successfully.")
            return True
        else:
            print("Failed to cancel booking.")
            return False

    def get_available_no_of_tickets(self, event_name):
        event = self.events.get(event_name)
        return event.available_seats if event else 0

    def get_event_details(self, event_name):
        event = self.events.get(event_name)
        if event:
            event.display_event_details()
        else:
```

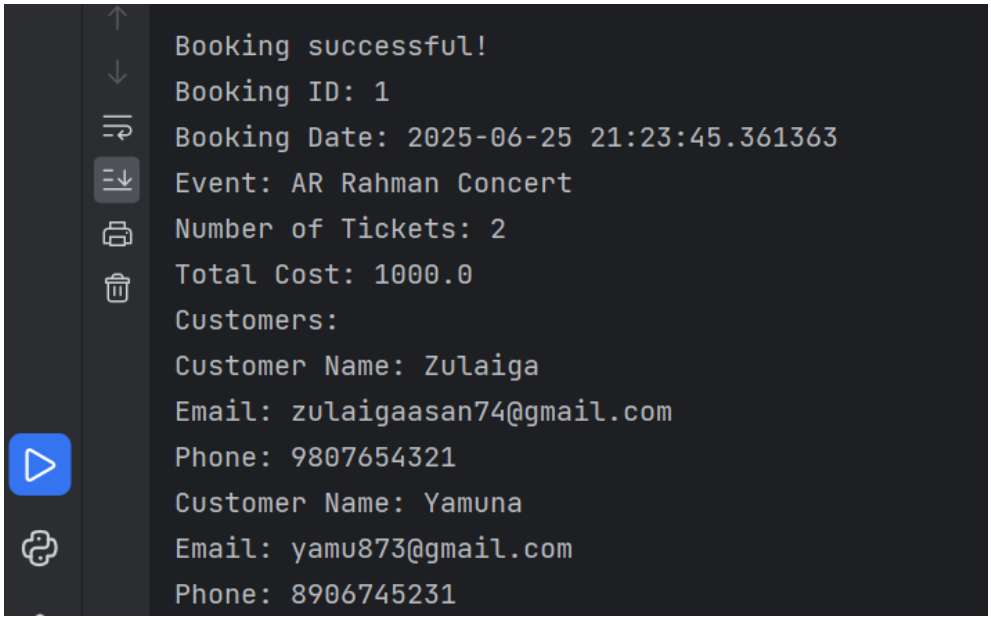
```
        print("Event not found!")

def main(self):
    venue1 = Venue("Dorais Theatre", "Opp to Pazhamudir")
    venue2 = Venue("Sakthi concert hall", "Market Road")
    venue3 = Venue("Sports Arena", "789 Stadium Rd")

    # Create events (these will be added to the dictionary)
    movie = self.create_event(
        "Oh my Kadavuley", "2025-06-25", "18:00", 200, 250.00,
        EventType.MOVIE, venue1, genre="Romantic Comedy", actor="Ashok Selvan",
        actress="Ritika Singh"
    )

    concert = self.create_event(
        "AR Rahman Concert", "2025-07-01", "20:00", 5000, 500.00,
        EventType.CONCERT, venue2, artist="AR Rahman",
        concert_type="Classical"
    )

    sports = self.create_event(
        "IPL", "2025-06-18", "15:30", 30000, 750.00,
        EventType.SPORTS, venue3, sport_name="Cricket",
        teams="CSK vs RCB"
    )

    while True:
        print("\nTicket Booking System")
        print("1. View All Events")
        print("2. Book Tickets")
        print("3. Cancel Booking")
        print("4. Check Available Seats")
        print("5. View Event Details")
        print("6. Exit")

        choice = input("Enter your choice (1-6): ")

        if choice == "1":
            print("\nAvailable Events:")
            # Iterate over dictionary values (Event objects)
            for i, event_obj in enumerate(self.events.values(), 1):
                print(f"{i}. {event_obj.event_name} ({event_obj.event_type.value})")

        elif choice == "2":
            event_name = input("Enter event name: ")
            try:
```

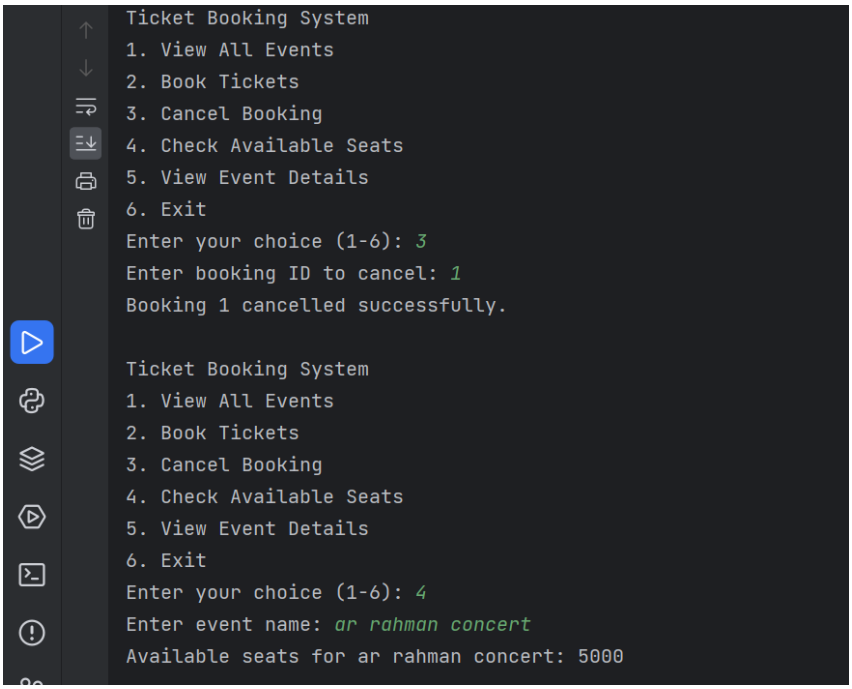
```
                num_tickets = int(input("Enter number of tickets: "))
                customers = []
                for i in range(num_tickets):
                    print(f"\nEnter details for customer {i + 1}:")
                    name = input("Name: ")
                    email = input("Email: ")
                    phone = input("Phone: ")
                    customers.append(Customer(name, email, phone))

                booking = self.book_tickets(event_name, num_tickets, customers)
                if booking:
                    print("\nBooking successful!")
                    booking.display_booking_details()
            except ValueError:
                print("Please enter a valid number")

        elif choice == "3":
            try:
                booking_id = int(input("Enter booking ID to cancel: "))
                self.cancel_booking(booking_id)
            except ValueError:
                print("Please enter a valid booking ID")

        elif choice == "4":
            event_name = input("Enter event name: ")
            available = self.get_available_no_of_tickets(event_name)
            print(f"Available seats for {event_name}: {available}")

        elif choice == "5":
            event_name = input("Enter event name: ")
            self.get_event_details(event_name)

        elif choice == "6":
            print("Thank you for using the Ticket Booking System!")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    system = BookingSystem()
    system.main()
```

OUTPUT :

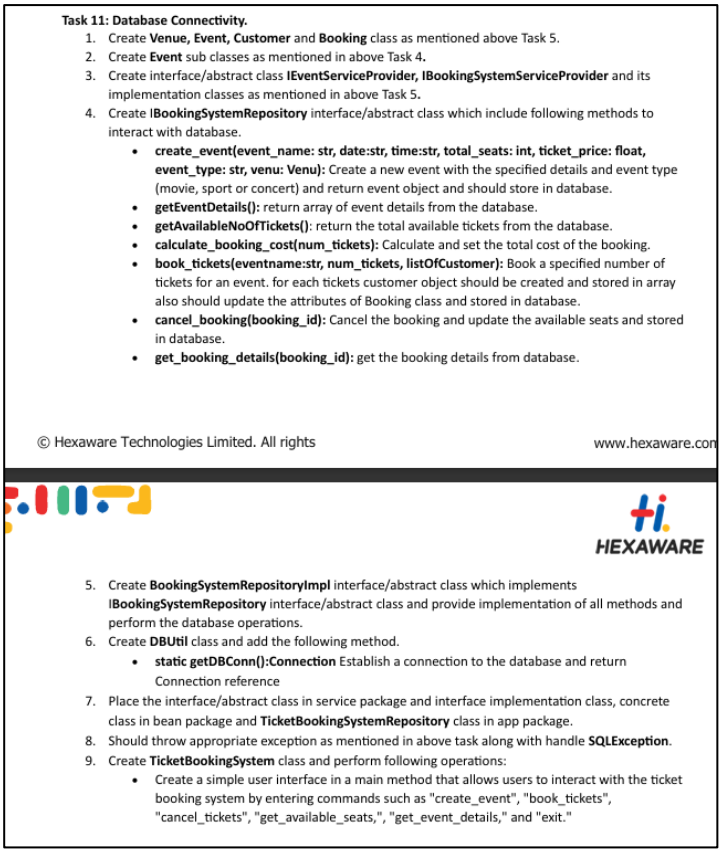

Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 1
Available Events:
1. Oh my Kadavuley (Movie)
2. AR Rahman Concert (Concert)
3. IPL (Sports)


Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 2
Enter event name: ar rahman concert
Enter number of tickets: 2
Enter details for customer 1:
Name: Zulaiga
Email: zulaigaasan74@gmail.com
Phone: 9807654321
Enter details for customer 2:
Name: Yamuna
Email: yamu873@gmail.com
Phone: 8906745231


Booking successful!
Booking ID: 1
Booking Date: 2025-06-25 21:23:45.361363
Event: AR Rahman Concert
Number of Tickets: 2
Total Cost: 1000.0
Customers:
Customer Name: Zulaiga
Email: zulaigaasan74@gmail.com
Phone: 9807654321
Customer Name: Yamuna
Email: yamu873@gmail.com
Phone: 8906745231


Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 3
Enter booking ID to cancel: 1
Booking 1 cancelled successfully.
Ticket Booking System
1. View All Events
2. Book Tickets
3. Cancel Booking
4. Check Available Seats
5. View Event Details
6. Exit
Enter your choice (1-6): 4
Enter event name: ar rahman concert
Available seats for ar rahman concert: 5000

TASK 11:



Task 11: Database Connectivity.

1. Create **Venue, Event, Customer** and **Booking** class as mentioned above Task 5.
2. Create **Event** sub classes as mentioned in above Task 4.
3. Create interface/abstract class **IEventServiceProvider, IBookingSystemServiceProvider** and its implementation classes as mentioned in above Task 5.
4. Create I**BookingSystemRepository** interface/abstract class which include following methods to interact with database.
   - **create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu: Venu):** Create a new event with the specified details and event type (movie, sport or concert) and return event object and should store in database.
   - **getEventDetails():** return array of event details from the database.
   - **getAvailableNoOfTickets()**: return the total available tickets from the database.
   - **calculate_booking_cost(num_tickets):** Calculate and set the total cost of the booking.
   - **book_tickets(eventname:str, num_tickets, listOfCustomer):** Book a specified number of tickets for an event. for each tickets customer object should be created and stored in array also should update the attributes of Booking class and stored in database.
   - **cancel_booking(booking_id):** Cancel the booking and update the available seats and stored in database.
   - **get_booking_details(booking_id):** get the booking details from database.

HEXAWARE

5. Create **BookingSystemRepositoryImpl** interface/abstract class which implements I**BookingSystemRepository** interface/abstract class and provide implementation of all methods and perform the database operations.
6. Create **DBUtil** class and add the following method.
   - **static getDBConn():Connection** Establish a connection to the database and return Connection reference
7. Place the interface/abstract class in service package and interface implementation class, concrete class in bean package and **TicketBookingSystemRepository** class in app package.
8. Should throw appropriate exception as mentioned in above task along with handle **SQLException**.
9. Create **TicketBookingSystem** class and perform following operations:
   - Create a simple user interface in a main method that allows users to interact with the ticket booking system by entering commands such as "create_event", "book_tickets", "cancel_tickets", "get_available_seats,", "get_event_details," and "exit."

CODE :

```
import mysql.connector
from mysql.connector import Error
from enum import Enum
from datetime import datetime
import sys

DB_CONFIG = {
    'host': 'localhost',
    'database': 'ticketbookingsystem',
    'user': 'root',
    'password': 'Zuhi743#'
```

```
}

class EventType(Enum):
    MOVIE = "Movie"
    CONCERT = "Concert"
    SPORTS = "Sports"

class DBUtil:
    @staticmethod
    def get_db_conn():
        try:
            conn = mysql.connector.connect(**DB_CONFIG)
            return conn
        except Error as e:
            print(f'Error connecting to MySQL: {e}")
            return None

class Venue:
    def __init__(self, venue_name, address, venue_id=None):
        self.venue_id = venue_id
        self.venue_name = venue_name
        self.address = address

class Customer:
    def __init__(self, name, email, phone, customer_id=None):
        self.customer_id = customer_id
        self.name = name
        self.email = email
        self.phone = phone
```

```
class Event:
    def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, event_type, event_id=None):
        self.event_id = event_id
        self.event_name = event_name
        self.event_date = event_date
        self.event_time = event_time
        self.venue = venue
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.event_type = event_type

class Movie(Event):
    def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, genre, actor_name, actress_name, event_id=None):
        super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, EventType.MOVIE, event_id)
        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, total_seats, ticket_price, artist, concert_type, event_id=None):
        super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, EventType.CONCERT, event_id)
        self.artist = artist
        self.concert_type = concert_type

class Sports(Event):
```

```
    def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, sport_name, teams, event_id=None):
        super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, EventType.SPORTS, event_id)
        self.sport_name = sport_name
        self.teams = teams


class Booking:
    def __init__(self, booking_id, num_tickets, total_cost, booking_date, event_id, customer_id):
        self.booking_id = booking_id
        self.num_tickets = num_tickets
        self.total_cost = total_cost
        self.booking_date = booking_date
        self.event_id = event_id
        self.customer_id = customer_id


class BookingSystemRepository:



    def get_available_seats(self, event_id):
        conn = None
        cursor = None
        try:
            conn = DBUtil.get_db_conn()
            cursor = conn.cursor()
            cursor.execute("SELECT available_seats FROM event WHERE event_id = %s", (event_id,))
            result = cursor.fetchone()
            return result[0] if result else 0
        except Error as e:
            print(f"Error getting available seats: {e}")
```

```
            return 0
        finally:
            if cursor: cursor.close()
            if conn: conn.close()




    def cancel_booking(self, booking_id):
        conn = None
        cursor = None
        try:
            conn = DBUtil.get_db_conn()
            cursor = conn.cursor()

            cursor.execute("SELECT num_tickets, event_id FROM booking WHERE booking_id
= %s", (booking_id,))
            result = cursor.fetchone()
            if not result:
                return False

            num_tickets, event_id = result
            cursor.execute(
                "UPDATE event SET available_seats = available_seats + %s WHERE event_id =
%s",
                (num_tickets, event_id)
            )
            cursor.execute("DELETE FROM booking WHERE booking_id = %s", (booking_id,))

            conn.commit()
            return True
        except Error as e:
```

```
            print(f"Error canceling booking: {e}")
            if conn: conn.rollback()
            return False
        finally:
            if cursor: cursor.close()
            if conn: conn.close()

    def get_booking_details(self, booking_id):
        conn = None
        cursor = None
        try:
            conn = DBUtil.get_db_conn()
            cursor = conn.cursor(dictionary=True)

            cursor.execute("""
                SELECT b.*, e.event_name, v.venue_name, c.customer_name, c.email,
c.phone_number
                FROM booking b
                JOIN event e ON b.event_id = e.event_id
                JOIN venue v ON e.venue_id = v.venue_id
                JOIN customer c ON b.customer_id = c.customer_id
                WHERE b.booking_id = %s
            """, (booking_id,))
            booking = cursor.fetchone()

            if not booking:
                return None

            return Booking(
                booking['booking_id'], booking['num_tickets'], booking['total_cost'],
                booking['booking_date'], booking['event_id'], booking['customer_id']
```

```
            )
        except Error as e:
            print(f"Error getting booking details: {e}")
            return None
        finally:
            if cursor: cursor.close()
            if conn: conn.close()

class TicketBookingSystem:
    def __init__(self):
        self.repository = BookingSystemRepository()

    def display_menu(self):
        print("\nTicket Booking System")
        print("1. Cancel Booking")
        print("2. Check Available Seats")
        print("3. View Booking Details")
        print("4. Exit")

    def run(self):
        while True:
            self.display_menu()
            choice = input("Enter your choice (1-4): ")



            if choice == "1":
                self.cancel_booking()
            elif choice == "2":
                self.check_available_seats()
            elif choice == "3":
```

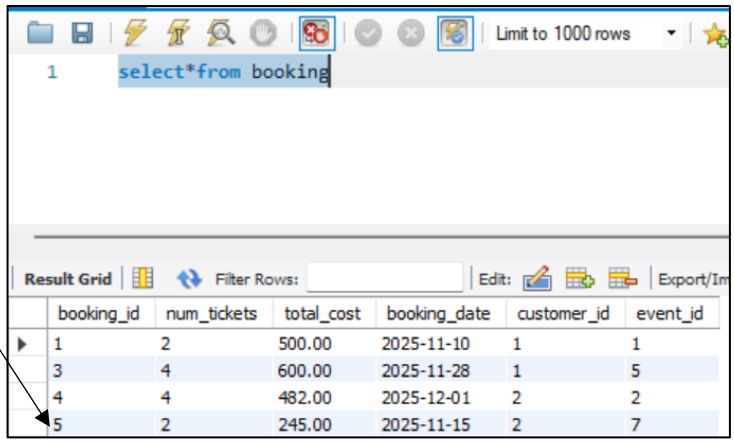
```
            self.view_booking_details()
        elif choice == "4":
            print("Thank you for using the Ticket Booking System!")
            sys.exit()
        else:
            print("Invalid choice. Please try again.")

    def cancel_booking(self):
        booking_id = input("Enter Booking ID to cancel: ")
        try:
            booking_id = int(booking_id)
        except ValueError:
            print("Invalid Booking ID. Please enter a number.")
            return

        if self.repository.cancel_booking(booking_id):
            print("Booking cancelled successfully.")
        else:
            print("Failed to cancel booking. Booking ID may not exist.")

    def check_available_seats(self):
        event_id = input("Enter Event ID: ")
        try:
            event_id = int(event_id)
        except ValueError:
            print("Invalid Event ID. Please enter a number.")
            return

        available = self.repository.get_available_seats(event_id)
        print(f"Available seats: {available}")
```

```
    def view_booking_details(self):
        booking_id = input("Enter Booking ID: ")
        try:
            booking_id = int(booking_id)
        except ValueError:
            print("Invalid Booking ID. Please enter a number.")
            return

        booking = self.repository.get_booking_details(booking_id)
        if not booking:
            print("Booking not found.")
            return

        print("\nBooking Details:")
        print("-" * 50)
        print(f'Booking ID: {booking.booking_id}")
        print(f'Number of Tickets: {booking.num_tickets}")
        print(f'Total Cost: {booking.total_cost:.2f}")
        print(f'Booking Date: {booking.booking_date}")
        print("-" * 50)

if __name__ == "__main__":
    system = TicketBookingSystem()
    system.run()
```

OUTPUT:

```
Ticket Booking System
1. Cancel Booking
2. Check Available Seats
3. View Booking Details
4. Exit
Enter your choice (1-4): 2
Enter Event ID: 1
Available seats: 150

Ticket Booking System
1. Cancel Booking
2. Check Available Seats
3. View Booking Details
4. Exit
Enter your choice (1-4): 3
Enter Booking ID: 5

Booking Details:
------------------------------------------------
Booking ID: 5
Number of Tickets: 2
Total Cost: 245.00
Booking Date: 2025-11-15
------------------------------------------------

Ticket Booking System
1. Cancel Booking
2. Check Available Seats
3. View Booking Details
4. Exit
Enter your choice (1-4): 1
Enter Booking ID to cancel: 10
```

```
Booking cancelled successfully.

Ticket Booking System
1. Cancel Booking
2. Check Available Seats
3. View Booking Details
4. Exit
Enter your choice (1-4): 4
Thank you for using the Ticket Booking System!
PS C:\Users\Zulaiga\OneDrive\Desktop\Python-Assignment>
```



booking_id	num_tickets	total_cost	booking_date	customer_id	event_id
1	2	500.00	2025-11-10	1	1
3	4	600.00	2025-11-28	1	5
4	4	482.00	2025-12-01	2	2
5	2	245.00	2025-11-15	2	7

Available seats for event_id : 150

```
select*from event
```

event_id	event_name	event_date	event_time	total_seats	available_seats	ticket_price	event_type	venue_id
1	Court teaser release	2025-11-15	19:00:00	500	150	250.00	Movie	1
2	Football Final	2025-12-10	20:30:00	20000	5000	120.50	Sports	2

```
select*from booking
```

booking_id	num_tickets	total_cost	booking_date	customer_id	event_id
1	2	500.00	2025-11-10	1	1
3	4	600.00	2025-11-28	1	5
4	4	482.00	2025-12-01	2	2
5	2	245.00	2025-11-15	2	7
6	1	489.99	2025-11-20	3	3
7	3	900.00	2025-12-05	3	8
8	3	359.97	2025-11-15	4	4
9	2	950.00	2025-12-10	4	6
10	2	300.00	2025-11-30	5	5
11	1	200.00	2025-12-08	5	9
12	2	950.00	2025-12-10	6	6

```
select*from booking
```

booking_id	num_tickets	total_cost	booking_date	customer_id	event_id
1	2	500.00	2025-11-10	1	1
3	4	600.00	2025-11-28	1	5
4	4	482.00	2025-12-01	2	2
5	2	245.00	2025-11-15	2	7
6	1	489.99	2025-11-20	3	3
7	3	900.00	2025-12-05	3	8
8	3	359.97	2025-11-15	4	4
9	2	950.00	2025-12-10	4	6
11	1	200.00	2025-12-08	5	9
12	2	950.00	2025-12-10	6	6
13	4	600.00	2025-11-18	6	10