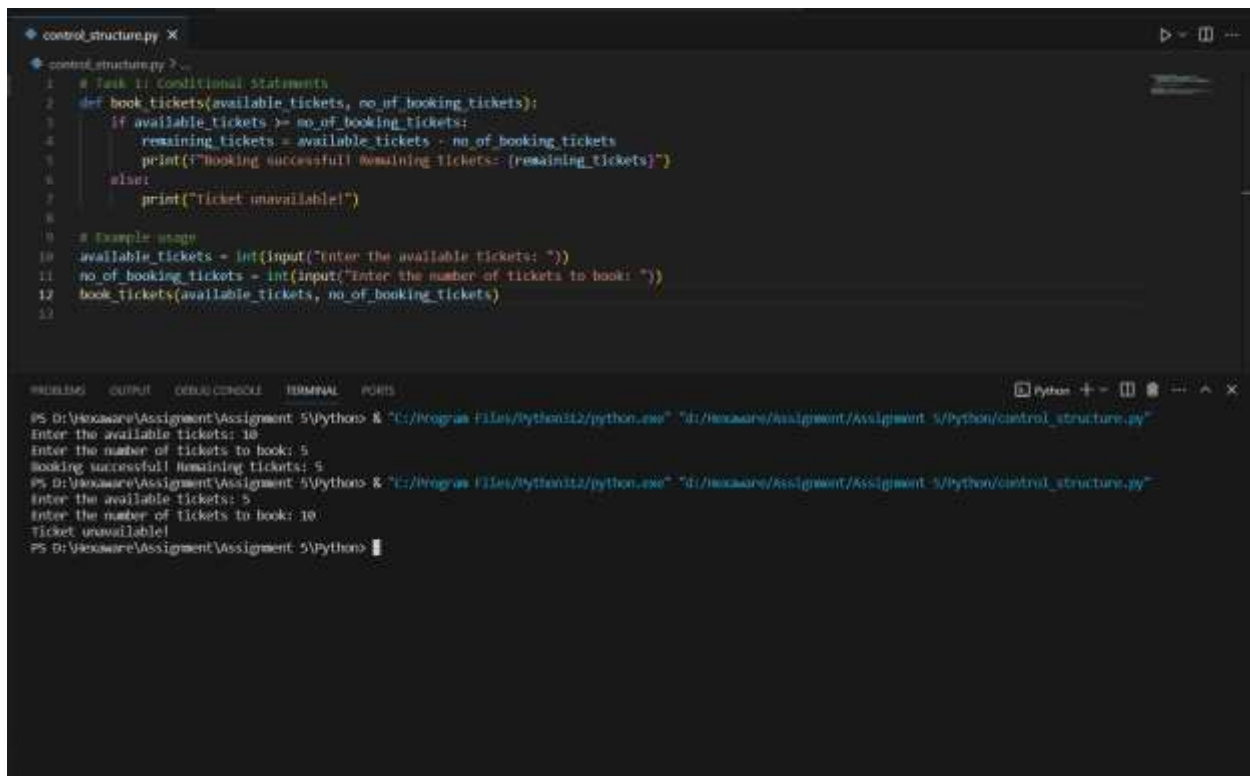


Assignment 5

Ticket Booking System

1.Control structure.

Task1:




The screenshot shows a Python IDE with a file named `control_structure.py`. The code implements a ticket booking system using conditional statements. It defines a function `book_tickets` that takes `available_tickets` and `no_of_booking_tickets` as arguments. If the available tickets are greater than or equal to the booking tickets, it prints the remaining tickets; otherwise, it prints "Ticket unavailable!". Below the function, there is an example usage section that prompts the user for the number of available tickets and the number of tickets to book, then calls the `book_tickets` function.

```
1 # Task 1: Conditional Statements
2 def book_tickets(available_tickets, no_of_booking_tickets):
3     if available_tickets >= no_of_booking_tickets:
4         remaining_tickets = available_tickets - no_of_booking_tickets
5         print(f"Booking successful! Remaining tickets: {remaining_tickets}")
6     else:
7         print("Ticket unavailable!")
8
9 # Example usage
10 available_tickets = int(input("Enter the available tickets: "))
11 no_of_booking_tickets = int(input("Enter the number of tickets to book: "))
12 book_tickets(available_tickets, no_of_booking_tickets)
13
```

The terminal output shows the program being executed twice. In the first run, 10 available tickets and 5 booking tickets are entered, resulting in "Booking successful! Remaining tickets: 5". In the second run, 5 available tickets and 10 booking tickets are entered, resulting in "Ticket unavailable!".

```
PS D:\vscode\Assignment\Assignment 5\Python> "C:/Program Files/Python312/python.exe" "d:/vscode/Assignment/Assignment 5/Python/control_structure.py"
Enter the available tickets: 10
Enter the number of tickets to book: 5
Booking successful! Remaining tickets: 5
PS D:\vscode\Assignment\Assignment 5\Python> "C:/Program Files/Python312/python.exe" "d:/vscode/Assignment/Assignment 5/Python/control_structure.py"
Enter the available tickets: 5
Enter the number of tickets to book: 10
Ticket unavailable!
PS D:\vscode\Assignment\Assignment 5\Python>
```

TASK2:



The screenshot shows a VS Code editor with a file named `control_structure.py` open. The code defines a function `calculate_ticket_cost` that takes a category and number of tickets as input and returns the total cost. The function uses a dictionary for base prices and includes error handling for invalid categories. Below the function, there is a sample usage section that prompts the user for category and number of tickets, and then calls the function.

```

1 # task 2: nested
15
16 def calculate_ticket_cost(category, num_tickets):
17     base_prices = {"silver": 50, "gold": 100, "diamond": 200}
18
19     if category in base_prices:
20         base_price = base_prices[category]
21         total_cost = base_price * num_tickets
22         print(f"Total cost for {num_tickets} {category} tickets: ${total_cost}")
23     else:
24         print("Invalid ticket category.")
25
26 # Example usage
27 ticket_category = input("Enter the ticket category (silver/gold/diamond): ")
28 num_tickets = int(input("Enter the number of tickets to book: "))
29 calculate_ticket_cost(ticket_category, num_tickets)
30

```

The terminal output shows the execution of the script. It prompts the user for the ticket category and the number of tickets, and then displays the total cost for each category.

```

PS D:\Hexaware\Assignment\Assignment 5\Python> "C:\Program Files\Python12\python.exe" "d:\Hexaware\Assignment\Assignment 5\Python\control_structure.py"
Enter the ticket category (silver/gold/diamond): Silver
Enter the number of tickets to book: 10
Total cost for 10 Silver tickets: $500
PS D:\Hexaware\Assignment\Assignment 5\Python> "C:\Program Files\Python12\python.exe" "d:\Hexaware\Assignment\Assignment 5\Python\control_structure.py"
Enter the ticket category (silver/gold/diamond): Gold
Enter the number of tickets to book: 10
Total cost for 10 Gold tickets: $1000
PS D:\Hexaware\Assignment\Assignment 5\Python> "C:\Program Files\Python12\python.exe" "d:\Hexaware\Assignment\Assignment 5\Python\control_structure.py"
Enter the ticket category (silver/gold/diamond): Diamond
Enter the number of tickets to book: 10
Total cost for 10 Diamond tickets: $2000
PS D:\Hexaware\Assignment\Assignment 5\Python>

```

Task3:

```

control_structure.py ?
25 > """
26 # Task 1
27 while True:
28     ticket_category = input("Enter the ticket category (Silver/Gold/Diamond): ")
29     if ticket_category == "exit":
30         break
31     num_tickets = int(input("Enter the number of tickets to book: "))
32     calculate_ticket_cost(ticket_category, num_tickets)
33
34
35

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\VivaWare\Assignment\Assignment_5\Python> "C:\Program Files\Python12\python.exe" "D:\VivaWare\Assignment\Assignment_5\Python\control_structure.py"
Enter the ticket category (Silver/Gold/Diamond): Silver
Enter the number of tickets to book: 10
Total cost for 10 Silver tickets: \$800
Enter the ticket category (Silver/Gold/Diamond): exit
PS D:\VivaWare\Assignment\Assignment_5\Python>

Class & OOPS:

Task-4,5&6:

Code:

```
class.py X
class.py: X
1 from enum import Enum
2
3 # Enum for event type:
4 class EventType(Enum):
5     MOVIE = 'movie'
6     SPORTS = 'sports'
7     CONCERT = 'concert'
8
9 # Event class
10 class Event:
11     def __init__(self, event_name, event_date, event_time, venue_name, total_seats, ticket_price, event_type):
12         self.event_name = event_name
13         self.event_date = event_date
14         self.event_time = event_time
15         self.venue_name = venue_name
16         self.total_seats = total_seats
17         self.available_seats = total_seats
18         self.ticket_price = ticket_price
19         self.event_type = EventType(event_type)
20
21     def calculate_total_revenue(self):
22         return (self.total_seats - self.available_seats) * self.ticket_price
23
24     def get_booked_no_of_tickets(self):
25         return self.total_seats - self.available_seats
26
27     def book_tickets(self, num_tickets):
28         if self.available_seats >= num_tickets:
29             self.available_seats -= num_tickets
30             print("Booking successful! Remaining seats: (self.available_seats)")
31         else:
32             print("Ticket unavailable!")
33
```

```
class.py X
class.py: --
34
35     def cancel_booking(self, num_tickets):
36         self.available_seats += num_tickets
37         print("Booking canceled! Remaining seats: (self.available_seats)")
38
39     def display_event_details(self):
40         print(f"Event Name: {self.event_name}")
41         print(f>Date and Time: {self.event_date} {self.event_time}")
42         print(f>Venue: {self.venue_name}")
43         print(f>Total Seats: {self.total_seats}")
44         print(f>Available Seats: {self.available_seats}")
45         print(f>Ticket Price: ${self.ticket_price}")
46         print(f>Event Type: {self.event_type.value}")
47
48 # Venue class
49 class Venue:
50     def __init__(self, venue_name, address):
51         self.venue_name = venue_name
52         self.address = address
53
54     def display_venue_details(self):
55         print(f"Venue Name: {self.venue_name}")
56         print(f>Address: {self.address}")
57
58 # Customer class
59 class Customer:
60     def __init__(self, customer_name, email, phone_number):
61         self.customer_name = customer_name
62         self.email = email
63         self.phone_number = phone_number
64
65     def display_customer_details(self):
66         print(f"Customer Name: {self.customer_name}")
67         print(f>Email: {self.email}")
68         print(f>Phone Number: {self.phone_number}")
69
```

```

classroom X
classroom > % Event
# Booking class
class Booking()
22     def __init__(self, event, num_tickets):
23         self.event = event
24         self.num_tickets = num_tickets
25         self.calculate_booking_cost()
26
27     def calculate_booking_cost(self):
28         self.total_cost = self.num_tickets * self.event.ticket_price
29
30     def display_booking_details(self):
31         print(f"Booking details for event '{self.event.event_name}':")
32         print(f"Number of tickets: {self.num_tickets}")
33         print(f"Total cost: ${self.total_cost}")
34
35 # Movie class (subclass of Event)
class Movie(Event)
36     def __init__(self, event_name, event_date, event_time, venue_name, total_seats, ticket_price, genre, actor_name, actress_name):
37         super().__init__(event_name, event_date, event_time, venue_name, total_seats, ticket_price, EventType.MOVIE)
38         self.genre = genre
39         self.actor_name = actor_name
40         self.actress_name = actress_name
41
42     def display_event_details(self):
43         super().display_event_details()
44         print(f"Genre: {self.genre}")
45         print(f"Actor: {self.actor_name}")
46         print(f"Actress: {self.actress_name}")
47
48 # Concert class (subclass of Event)
class Concert(Event)
49     def __init__(self, event_name, event_date, event_time, venue_name, total_seats, ticket_price, artist, concert_type):
50         super().__init__(event_name, event_date, event_time, venue_name, total_seats, ticket_price, EventType.CONCERT)
51         self.artist = artist
52         self.concert_type = concert_type
53
54

```

```

classroom X
classroom > % TicketBookingSystem
55     def display_event_details(self):
56         super().display_event_details()
57         print(f"Artist: {self.artist}")
58         print(f"Concert type: {self.concert_type}")
59
60 # Sports class (subclass of Event)
class Sports(Event)
61     def __init__(self, event_name, event_date, event_time, venue_name, total_seats, ticket_price, sport_name, team_name):
62         super().__init__(event_name, event_date, event_time, venue_name, total_seats, ticket_price, EventType.SPORTS)
63         self.sport_name = sport_name
64         self.team_name = team_name
65
66     def display_event_details(self):
67         super().display_event_details()
68         print(f"Sport name: {self.sport_name}")
69         print(f"Team: {self.team_name}")
70
71 # TicketBookingSystem class
class TicketBookingSystem
72     @staticmethod
73     def create_event(event_name, event_date, event_time, total_seats, ticket_price, event_type, venue_name):
74         if event_type == EventType.MOVIE:
75             return Movie(event_name, event_date, event_time, venue_name, total_seats, ticket_price, "", "")
76         elif event_type == EventType.CONCERT:
77             return Concert(event_name, event_date, event_time, venue_name, total_seats, ticket_price, "", "")
78         elif event_type == EventType.SPORTS:
79             return Sports(event_name, event_date, event_time, venue_name, total_seats, ticket_price, "", "")
80         else:
81             print("Invalid event type.")
82             return None
83
84     @staticmethod
85     def display_event_details(event):
86         event.display_event_details()
87
88

```

```

89
90     @staticmethod
91     def book_tickets(event, num_tickets):
92         event.book_tickets(num_tickets)
93
94     @staticmethod
95     def cancel_tickets(event, num_tickets):
96         event.cancel_booking(num_tickets)
97
98

```

Inputs for movie:

```
147
148 @staticmethod
149 def main():
150     # Example usage
151     venue = Venue("PMU", "123 ,Jaipur")
152     event = TicketBookingSystem.create_event("Movie Night", "2023-01-01", "20:00", 100, 10.0, EventType.MOVIE, venue)
153     customer = Customer("Rohan Chaudhari", "rohan@example.com", "123-456-7890")
154
155     print("\nEvent Details:")
156     TicketBookingSystem.display_event_details(event)
157
158     print("\nBooking Tickets:")
159     TicketBookingSystem.book_tickets(event, 5)
160
161     print("\nCancel booking:")
162     TicketBookingSystem.cancel_tickets(event, 2)
163
164     print("\nCustomer Details:")
165     customer.display_customer_details()
166
167     booking = Booking(event, 3)
168     print("\nBooking Details:")
169     booking.display_booking_details()
170
171 if __name__ == "__main__":
172     TicketBookingSystem.main()
173
```

Output for movie:

```
PS D:\workspace\Assignment\Assignment 5\Python & "C:\Program Files\Python12\python.exe" "D:\workspace\Assignment\Assignment 5\python\cinema.py"
Event Details:
Event Name: Movie Night
Date and Time: 2023-01-01 20:00
Venue: C:\main_\Venue object at 0x0000014000000000
Total Seats: 100
Available Seats: 100
Ticket Price: $10.0
Event Type: Movie
Genre:
Actor:
Actress:

Booking Tickets:
Booking successful! Remaining seats: 95

Cancel Booking:
Booking canceled! Remaining seats: 97

Customer Details:
Customer Name: Rohan Chaudhari
Email: rohan@example.com
Phone Number: 123-456-7890

Booking Details:
Booking Details for Event 'Movie Night':
Number of Tickets: 3
Total Cost: $30.0
PS D:\workspace\Assignment\Assignment 5\Python
```

Input for sports:

```
147
148 @staticmethod
149 def main():
150     # Example usage
151     venue = Venue("Shreeide", "123 ,Rajkot")
152     event = TicketBookingSystem.create_event("Cricket", "2023-01-01", "20:00", 100, 10.0, EventType.SPORTS, venue)
153     customer = Customer("Rohan Chaudhari", "rohan@example.com", "123-456-7890")
154
155     print("\nEvent Details:")
156     TicketBookingSystem.display_event_details(event)
157
158     print("\nBooking Tickets:")
159     TicketBookingSystem.book_tickets(event, 4)
160
161     print("\nCancel booking:")
162     TicketBookingSystem.cancel_tickets(event, 3)
163
164     print("\nCustomer Details:")
165     customer.display_customer_details()
166
167     booking = Booking(event, 4)
168     print("\nBooking Details:")
169     booking.display_booking_details()
170
171 if __name__ == "__main__":
172     TicketBookingSystem.main()
173
```

Output for sports:

```
PS D:\oscar\Assignment\Assignment 5\Python & "C:/Program Files/Python12/python.exe" "D:/oscar/Assignment/Assignment 5/Python/sports.py"

Event Details:
Event Name: Cricket
Date and Time: 2023-01-01 20:00
Venue: <main.Venue object at 0x00000123456789AB>
Total Seats: 100
Available Seats: 100
Ticket Price: $10.0
Event Type: Sports
Sport Name:
Team:

Booking Tickets:
Booking successful! Remaining seats: 96

Cancel Booking:
Booking cancelled! Remaining seats: 97

Customer Details:
Customer Name: Rohan Chaudhari
Email: rohan@example.com
Phone Number: 123-456-7890

Booking Details:
Booking Details for Event 'Cricket':
Number of Tickets: 4
Total Cost: $40.0
PS D:\oscar\Assignment\Assignment 5\Python
```

Input for concert:

```
140     @staticmethod
141     def main():
142         # Example usage
143         venue = Venue("Concert in Mumbai", "1234, Mumbai")
144         event = TicketBookingSystem.create_event("Arjit Singh", "2023-01-01", "20:00", 100, 10.0, EventType.CONCERT, venue)
145         customer = Customer("Rohan Chaudhari", "rohan@example.com", "123-456-7890")
146
147         print("\nEvent Details:")
148         TicketBookingSystem.display_event_details(event)
149
150         print("\nBooking Tickets:")
151         TicketBookingSystem.book_tickets(event, 4)
152
153         print("\nCancel Booking:")
154         TicketBookingSystem.cancel_tickets(event, 2)
155
156         print("\nCustomer Details:")
157         customer.display_customer_details()
158
159         booking = Booking(event, 2)
160         print("\nBooking Details:")
161         booking.display_booking_details()
162
163     if __name__ == "__main__":
164         TicketBookingSystem.main()
```

Output for concert:

```
PS D:\oscar\Assignment\Assignment 5\Python & "C:/Program Files/Python12/python.exe" "D:/oscar/Assignment/Assignment 5/Python/classroom.py"

Event Details:
Event Name: Arjit Singh
Date and Time: 2023-01-01 20:00
Venue: <main.Venue object at 0x00000123456789AB>
Total Seats: 100
Available Seats: 100
Ticket Price: $10.0
Event Type: Concert
Artist:

Booking Tickets:
Booking successful! Remaining seats: 98

Cancel Booking:
Booking cancelled! Remaining seats: 96

Customer Details:
Customer Name: Rohan Chaudhari
Email: rohan@example.com
Phone Number: 123-456-7890

Booking Details:
Booking Details for Event 'Arjit Singh':
Number of Tickets: 2
Total Cost: $20.0
PS D:\oscar\Assignment\Assignment 5\Python
```

TASK 7:

Code:

```
task7.py •
task7.py > Event
1 from datetime import datetime
2
3 class Venue:
4     def __init__(self, venue_name, address):
5         self.venue_name = venue_name
6         self.address = address
7
8     def display_venue_details(self):
9         print(f"Venue name: {self.venue_name}")
10        print(f"Address: {self.address}")
11
12 class Event:
13     def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, event_type):
14         self.event_name = event_name
15         self.event_date = event_date
16         self.event_time = event_time
17         self.venue = venue
18         self.total_seats = total_seats
19         self.available_seats = total_seats
20         self.ticket_price = ticket_price
21         self.event_type = event_type
22
23     def calculate_total_revenue(self):
24         return (self.total_seats - self.available_seats) * self.ticket_price
25
26     def get_booked_no_of_tickets(self):
27         return self.total_seats - self.available_seats
28
29     def book_tickets(self, num_tickets):
30         if self.available_seats >= num_tickets:
31             self.available_seats -= num_tickets
32             print(f"Booking successful! Remaining seats: {self.available_seats}")
33         else:
34             print("Ticket unavailable!")
35
36
```

```
task7.py •
task7.py > Event
36
37     def cancel_booking(self, num_tickets):
38         self.available_seats += num_tickets
39         print(f"Booking cancelled! Remaining seats: {self.available_seats}")
40
41     def display_event_details(self):
42         print(f"Event Name: {self.event_name}")
43         print(f"Date and Time: {self.event_date} {self.event_time}")
44         print(f"Venue Details:")
45         self.venue.display_venue_details()
46         print(f"Total Seats: {self.total_seats}")
47         print(f"Available Seats: {self.available_seats}")
48         print(f"Ticket Price: {self.ticket_price}")
49         print(f"Event Type: {self.event_type}")
50
51 class Movie(Event):
52     def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, genre, actor_name, actress_name):
53         super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, "Movie")
54         self.genre = genre
55         self.actor_name = actor_name
56         self.actress_name = actress_name
57
58     def display_event_details(self):
59         super().display_event_details()
60         print(f"Genre: {self.genre}")
61         print(f"Actor: {self.actor_name}")
62         print(f"Actress: {self.actress_name}")
63
64 class Concert(Event):
65     def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, artist, concert_type):
66         super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, "Concert")
67         self.artist = artist
68         self.concert_type = concert_type
69
70
```



```

task7.py > # booking
def display_event_details(self):
    super().display_event_details()
    print(f"Artist: {self.artist}")
    print(f"concert type: {self.concert_type}")

class Sport(Event):
    def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, sport_name, team_name):
        super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, "sport")
        self.sport_name = sport_name
        self.team_name = team_name

    def display_event_details(self):
        super().display_event_details()
        print(f"sport name: {self.sport_name}")
        print(f"team: {self.team_name}")

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 Number: {self.phone_number}")

class Booking:
    booking_id_counter = 1

```

```

task7.py > # booking.py
def __init__(self, event, num_tickets, customers):
    self.booking_id = Booking.booking_id_counter
    Booking.booking_id_counter += 1
    self.event = event
    self.num_tickets = num_tickets
    self.customers = customers
    self.total_cost = self.calculate_booking_cost()
    self.booking_date = datetime.now()

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

    def display_booking_details(self):
        print(f"Booking ID: {self.booking_id}")
        print(f"Event Details:")
        self.event.display_event_details()
        print(f"Number of tickets: {self.num_tickets}")
        print(f"Total cost: {self.total_cost}")
        print(f"Booking Date: {self.booking_date}")
        print(f"Customer Details:")
        for customer in self.customers:
            customer.display_customer_details()

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

    def create_event(self, event_name, event_date, event_time, total_seats, ticket_price, event_type, venue):
        event = None
        if event_type == "Music":
            event = Music(event_name, event_date, event_time, venue, total_seats, ticket_price, "Music", "John Doe", "Live Show")
        elif event_type == "Concert":
            event = Concert(event_name, event_date, event_time, venue, total_seats, ticket_price, "Artist", "Rock")
        elif event_type == "Sport":
            event = Sport(event_name, event_date, event_time, venue, total_seats, ticket_price, "Football", "Team A vs Team B")
        if event:
            self.events.append(event)

```

```

self.events.append(event)
return event
class:
    print(f"Invalid event type.")
    return None

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

```



```

task7.py •
# BookingSystem
144 def book_tickets(self, event_name, num_tickets, customers):
145     event = next((e for e in self.events if e.event_name == event_name), None)
146     if event:
147         event.book_tickets(num_tickets)
148         booking = Booking(event, num_tickets, customers)
149         print("Booking successful!")
150         return booking
151     else:
152         print("Event not found.")
153         return None
154
155 def cancel_booking(self, booking_id):
156     booking = next((b for b in self.bookings if b.booking_id == booking_id), None)
157     if booking:
158         booking.event.cancel_booking(booking.num_tickets)
159         self.bookings.remove(booking)
160         print("Booking cancelled!")
161     else:
162         print("Booking not found.")
163
164 def get_available_seats(self):
165     return sum(event.available_seats for event in self.events)
166
167 def get_event_details(self):
168     for event in self.events:
169         event.display_event_details()
170
171 def display_menu(self):
172     print("1. Create Event")
173     print("2. Book Tickets")
174     print("3. Cancel Booking")
175     print("4. Get Available Seats")
176     print("5. Get Event Details")
177     print("6. Exit")
178

```

Code to take input (While loop):

```

task7.py •
# BookingSystem
179 def main(self):
180     while True:
181         self.display_menu()
182         choice = input("Enter your choice: ")
183
184         if choice == "1":
185             event_name = input("Enter event name: ")
186             event_date = input("Enter event date (YYYY-MM-DD): ")
187             event_time = input("Enter event time (HH:MM): ")
188             total_seats = int(input("Enter total seats: "))
189             ticket_price = float(input("Enter ticket price: "))
190             event_type = input("Enter event type (Music/Concert/Sport): ")
191             venue_name = input("Enter venue name: ")
192             venue_address = input("Enter venue address: ")
193             venue = Venue(venue_name, venue_address)
194             self.create_event(event_name, event_date, event_time, total_seats, ticket_price, event_type, venue)
195
196         elif choice == "2":
197             event_name = input("Enter event name: ")
198             num_tickets = int(input("Enter number of tickets: "))
199             customers = []
200             for _ in range(num_tickets):
201                 customer_name = input("Enter customer name: ")
202                 email = input("Enter email: ")
203                 phone_number = input("Enter phone number: ")
204                 customer = Customer(customer_name, email, phone_number)
205                 customers.append(customer)
206             self.book_tickets(event_name, num_tickets, customers)
207
208         elif choice == "3":
209             booking_id = int(input("Enter booking ID: "))
210             self.cancel_booking(booking_id)
211
212         elif choice == "4":
213             available_seats = self.get_available_seats()
214             print(f"Total available seats: {available_seats}")
215
216

```

```

task7.py •
task7.py
217
218         elif choice == "5":
219             self.get_event_details()
220
221         elif choice == "6":
222             print("Exiting the Ticket Booking System. Goodbye!")
223             break
224
225         else:
226             print("Invalid choice. Please enter a valid option.")
227
228 if __name__ == "__main__":
229     booking_system = BookingSystem()
230     booking_system.main()
231

```

Inputs & Output In terminal:

1) for event sports(Similarly we can do for concert & Movie):

```

PYTHON  OUTPUT  TERMINAL  PORTS
Python  +  -  -  -  X
C:\Users\user\Assignment\Assignment 5\Python & "C:\Program Files\Python312\python.exe" "C:\Users\user\Assignment\Assignment 5\Python\Task1.py"
1. Create Seat
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
Enter your choice: 1
Enter event name: Cricket
Enter event date (YYYY-MM-DD): 2023-12-25
Enter event time (HH:MM): 17:00
Enter total seats: 300
Enter ticket price: 5000
Enter event type (movie/concert/sport): sport
Enter venue name: Lalbaan
Enter venue address: 123, Lalbaan

```

2) Created 2 tickets

```

PS D:\Yousang\Assignment\Assignment 3\Python & C++\Program Files\Python12\python.exe* "D:\Yousang\Assignment\Assignment 3\PythonTask3.py"
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
Enter your choice: 2
Enter event name: cricket
Enter number of tickets: 2
Enter customer name: Rohan
Enter email: rohan@example.com
Enter phone number: 1234567890
Enter customer name: Rohit
Enter email: rohit@example.com
Enter phone number: 1234567890
Event not found.
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
Enter your choice: 6
Exiting the Ticket Booking System. Goodbye!
PS D:\Yousang\Assignment\Assignment 3\Python

```

3)

```
PS C:\Users\user\Assignment\Assignment 5\Python & "C:\Program Files\Python312\python.exe" "D:\Users\user\Assignment\Assignment 5\Python\aw7.py"
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
Enter your choice: 3
Enter booking ID: 1
Booking deleted
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
Enter your choice: 6
Exiting the Ticket Booking System. Goodbye!
PS C:\Users\user\Assignment\Assignment 5\Python
```

4)

```
PROGRAM OUTPUT
D:\Jyoti\Assignment\Assignment 5\Python & "C:\Program Files\Python32\python.exe" "D:\Jyoti\Assignment\Assignment 5\Python\task0.py"
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
Enter your choice: 1
Enter event name: Cricket
Enter event date (YYYY-MM-DD): 2023-12-25
Enter event time (HH:MM): 07:00
Enter total seats: 100
Enter ticket price: 1000
Enter event type (Movie/Concert/Sport): Sport
Enter venue name: JaiHaram
Enter venue address: 123 Jalgam
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
Enter your choice: 4
Total Available Seats: 100
```

5)

```
Enter your choice: 3
Event Name: Cricket
Date and Time: 2023-12-25 07:00
Venue Details:
Venue Name: JaiHaram
Address: 123 Jalgam
Total Seats: 100
Available Seats: 100
Ticket Price: $1000.0
Event type: Sport
Sport Name: Football
Teams: Team A vs Team B
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available seats
5. Get Event Details
6. Exit
Enter your choice: 6
Exiting the Ticket Booking System. Goodbye!
D:\Jyoti\Assignment\Assignment 5\Python
```

Task8:

```
task8-10.py X
task8-10.py ? ..
1 from datetime import datetime
2 from os import import Enum
3 from abc import ABC, abstractmethod
4
5 # Base package
6 class Venue:
7     def __init__(self, venue_name, address):
8         self.venue_name = venue_name
9         self.address = address
10
11     def display_venue_details(self):
12         print(f"Venue Name: {self.venue_name}")
13         print(f"Address: {self.address}")
14
15 class Event(ABC):
16     event_id_counter = 1
17
18     def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, event_type):
19         self.event_id = Event.event_id_counter
20         Event.event_id_counter += 1
21         self.event_name = event_name
22         self.event_date = event_date
23         self.event_time = event_time
24         self.venue = venue
25         self.total_seats = total_seats
26         self.available_seats = total_seats
27         self.ticket_price = ticket_price
28         self.event_type = event_type
29
30     @abstractmethod
31     def display_event_details(self):
32         pass
```

```
task8-10.py X
task8-10.py ? ..
33
34 class Movie(Event):
35     def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, genre, actor_name, actress_name):
36         super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, "Movie")
37         self.genre = genre
38         self.actor_name = actor_name
39         self.actress_name = actress_name
40
41     def display_event_details(self):
42         super().display_event_details()
43         print(f"Genre: {self.genre}")
44         print(f"Actor: {self.actor_name}")
45         print(f"Actress: {self.actress_name}")
46
47 class Concert(Event):
48     def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, artist, concert_type):
49         super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, "Concert")
50         self.artist = artist
51         self.concert_type = concert_type
52
53     def display_event_details(self):
54         super().display_event_details()
55         print(f"Artist: {self.artist}")
56         print(f"Concert type: {self.concert_type}")
57
58 class Sport(Event):
59     def __init__(self, event_name, event_date, event_time, venue, total_seats, ticket_price, sport_name, teams_name):
60         super().__init__(event_name, event_date, event_time, venue, total_seats, ticket_price, "Sport")
61         self.sport_name = sport_name
62         self.teams_name = teams_name
63
64     def display_event_details(self):
65         super().display_event_details()
66         print(f"Sport Name: {self.sport_name}")
67         print(f"Teams: {self.teams_name}")
68
```

```

taskB-10.py X
taskB-10.py > ...
60 class Customer:
61     def __init__(self, customer_name, email, phone_number):
62         self.customer_name = customer_name
63         self.email = email
64         self.phone_number = phone_number
65
66     def display_customer_details(self):
67         print(f"Customer Name: {self.customer_name}")
68         print(f"Email: {self.email}")
69         print(f"Phone Number: {self.phone_number}")
70
71 class Booking:
72     booking_id_counter = 1
73
74     def __init__(self, event, num_tickets, customers):
75         self.booking_id = Booking.booking_id_counter
76         Booking.booking_id_counter += 1
77         self.event = event
78         self.num_tickets = num_tickets
79         self.customers = customers
80         self.total_cost = self.calculate_booking_cost()
81         self.booking_date = datetime.now()
82
83     def calculate_booking_cost(self):
84         return self.num_tickets * self.event.ticket_price
85
86     def display_booking_details(self):
87         print(f"Booking ID: {self.booking_id}")
88         print("Event Details:")
89         self.event.display_event_details()
90         print(f"Number of Tickets: {self.num_tickets}")
91         print(f"Total Cost: ${self.total_cost}")
92         print(f"Booking Date: {self.booking_date}")
93         print("Customer Details:")
94         for customer in self.customers:
95             customer.display_customer_details()
96
97

```

```

taskB-10.py X
taskB-10.py > ... EventServiceProviderImpl
107 class IEventServiceProvider(ABC):
108     @abstractmethod
109     def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue):
110         pass
111
112     @abstractmethod
113     def get_event_details(self):
114         pass
115
116     @abstractmethod
117     def get_available_no_of_tickets(self):
118         pass
119
120 class IBookingSystemServiceProvider(ABC):
121     @abstractmethod
122     def calculate_booking_cost(self, num_tickets):
123         pass
124
125     @abstractmethod
126     def book_tickets(self, eventname, num_tickets, array_of_customer):
127         pass
128
129     @abstractmethod
130     def cancel_booking(self, booking_id):
131         pass
132
133     @abstractmethod
134     def get_booking_details(self, booking_id):
135         pass
136
137 class EventServiceProviderImpl(IEventServiceProvider):
138     events = []
139

```

```

task8-10.py X
task8-10.py TicketBookingSystem
139
140 def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue):
141     event = None
142     if event_type == "Movie":
143         event = Movie(event_name, date, time, venue, total_seats, ticket_price, "Action", "John Doe", "Jane Doe")
144     elif event_type == "Concert":
145         event = Concert(event_name, date, time, venue, total_seats, ticket_price, "Artist", "Rock")
146     elif event_type == "Sport":
147         event = Sport(event_name, date, time, venue, total_seats, ticket_price, "Football", "TeamA vs TeamB")
148     if event:
149         self.events.append(event)
150         return event
151     else:
152         print("Invalid event type.")
153         return None
154
155 def get_event_details(self):
156     return self.events
157
158 def get_available_no_of_tickets(self):
159     return sum(event.available_seats for event in self.events)
160
161 class BookingSystemServiceProviderImpl(BookingSystemServiceProvider, EventServiceProvidImpl):
162     pass
163
164 # App package
165 class TicketBookingSystem:
166     booking_system = BookingSystemServiceProviderImpl()
167

```

Input method in code:

```

task8-10.py X
task8-10.py TicketBookingSystem main
168
169 @staticmethod
170 def display_menu():
171     print("1. Create Event")
172     print("2. Book Tickets")
173     print("3. Cancel Booking")
174     print("4. Get Available Seats")
175     print("5. Get Event Details")
176     print("6. Exit")
177
178 @staticmethod
179 def main():
180     while True:
181         TicketBookingSystem.display_menu()
182         choice = input("Enter your choice: ")
183
184         if choice == "1":
185             event_name = input("Enter event name: ")
186             date = input("Enter event date (YYYY-MM-DD): ")
187             time = input("Enter event time (HH:MM): ")
188             total_seats = int(input("Enter total seats: "))
189             ticket_price = float(input("Enter ticket price: "))
190             event_type = input("Enter event type (Movie/Concert/Sport): ")
191             venue_name = input("Enter venue name: ")
192             venue_address = input("Enter venue address: ")
193             venue = Venue(venue_name, venue_address)
194             TicketBookingSystem.booking_system.create_event(event_name, date, time, total_seats, ticket_price, event_type, venue)
195
196         elif choice == "2":
197             event_name = input("Enter event name: ")
198             num_tickets = int(input("Enter number of tickets: "))
199             customers = []
200             for _ in range(num_tickets):
201                 customer_name = input("Enter customer name: ")
202                 email = input("Enter email: ")
203                 phone_number = input("Enter phone number: ")
204                 customer = Customer(customer_name, email, phone_number)
205                 customers.append(customer)

```



```
task8-10.py X
task8-10.py > ...
202         phone_number = input("Enter phone number: ")
203         customer = Customer(customer_name, email, phone_number)
204         customers.append(customer)
205         TicketBookingSystem.booking_system.book_tickets(event_name, num_tickets, customers)
206
207     elif choice == "3":
208         booking_id = int(input("Enter booking ID: "))
209         TicketBookingSystem.booking_system.cancel_booking(booking_id)
210
211     elif choice == "4":
212         available_seats = TicketBookingSystem.booking_system.get_available_no_of_tickets()
213         print(f"Total Available Seats: {available_seats}")
214
215     elif choice == "5":
216         events = TicketBookingSystem.booking_system.get_event_details()
217         for event in events:
218             event.display_event_details()
219
220     elif choice == "6":
221         print("Exiting the Ticket Booking System. Goodbye!")
222         break
223
224     else:
225         print("Invalid choice. Please enter a valid option.")
226
227 if __name__ == "__main__":
228     TicketBookingSystem.main()
229
```

Exceptions:

```
task8-10.py •
task8-10.py > TicketBookingSystem
161 class BookingSystemServiceProviderImpl(BookingSystemServiceProvider, EventServiceProviderImpl):
162     pass
163
164
165 class EventNotFoundException(Exception):
166     pass
167
168 class InvalidBookingIDException(Exception):
169     pass
170
171 class TicketBookingSystem:
172     booking_system = BookingSystemServiceProviderImpl()
173
174     @staticmethod
175     def display_menu():
176         print("1. Create Event")
177         print("2. Book Tickets")
178         print("3. Cancel Booking")
179         print("4. Get Available Seats")
180         print("5. Get Event Details")
181         print("6. Exit")
182
```

```
task5-10.py •
task5-10.py > TicketBookingSystem > @main
183 @staticmethod
184 def main():
185     while True:
186         ticketbookingSystem.display_menu()
187         choice = input("Enter your choice: ")
188
189         if choice == "1":
190             event_name = input("Enter event name: ")
191             date = input("Enter event date (YYYY-MM-DD): ")
192             time = input("Enter event time (HH:MM): ")
193             total_seats = int(input("Enter total seats: "))
194             ticket_price = float(input("Enter ticket price: "))
195             event_type = input("Enter event type (Movie/Concert/Sport): ")
196             venue_name = input("Enter venue name: ")
197             venue_address = input("Enter venue address: ")
198             venue = Venue(venue_name, venue_address)
199             ticketbookingSystem.booking_system.create_event(event_name, date, time, total_seats, ticket_price, event_type, venue)
200
201         elif choice == "2":
202             event_name = input("Enter event name: ")
203             num_tickets = int(input("Enter number of tickets: "))
204             customers = []
205             for _ in range(num_tickets):
206                 customer_name = input("Enter customer name: ")
207                 email = input("Enter email: ")
208                 phone_number = input("Enter phone number: ")
209                 customer = Customer(customer_name, email, phone_number)
210                 customers.append(customer)
211             try:
212                 ticketbookingSystem.booking_system.book_tickets(event_name, num_tickets, customers)
213             except EventNotFoundException as e:
214                 print(f"Error: {e}")
215
```

```
216
217         elif choice == "3":
218             booking_id = int(input("Enter booking ID: "))
219             try:
220                 ticketbookingSystem.booking_system.cancel_booking(booking_id)
221             except InvalidBookingIDException as e:
222                 print(f"Error: {e}")
223
224         elif choice == "4":
225             available_seats = ticketbookingSystem.booking_system.get_available_no_of_tickets()
226             print(f"Total Available Seats: {available_seats}")
227
228         elif choice == "5":
229             events = ticketbookingSystem.booking_system.get_event_details()
230             for event in events:
231                 event.display_event_details()
232
233         elif choice == "6":
234             print("Exiting the Ticket Booking System. Goodbye!")
235             break
236
237         else:
238             print("Invalid choice. Please enter a valid option.")
239
240 if __name__ == "__main__":
241     ticketbookingSystem.main()
242
```

Collections:

```
80 # booking class
81 class Booking:
82     booking_id_counter = 1
83
84     def __init__(self, event, num_tickets, customers):
85         self.booking_id = Booking.booking_id_counter
86         Booking.booking_id_counter += 1
87         self.event = event
88         self.num_tickets = num_tickets
89         self.customers = set(customers)
90         self.total_cost = self.calculate_booking_cost()
91         self.booking_date = datetime.now()
92
93     def calculate_booking_cost(self):
94         return self.num_tickets * self.event.ticket_price
95
96     def display_booking_details(self):
97         print(f"Booking ID: {self.booking_id}")
98         print("Event Details:")
99         self.event.display_event_details()
100         print(f"Number of Tickets: {self.num_tickets}")
101         print(f"Total Cost: ${self.total_cost}")
102         print(f"Booking Date: {self.booking_date}")
103         print("Customer Details:")
104         for customer in self.customers:
105             customer.display_customer_details()
```

```
162 # bookingsystem class
163 class BookingSystemServiceImpl(BookingSystemServiceProvider, EventServiceProvider):
164     bookings = set()
165
166     def book_tickets(self, eventname, num_tickets, array_of_customer):
167         event = next((e for e in self.events if e.event_name == eventname), None)
168         if event:
169             if event.available_seats >= num_tickets:
170                 event.available_seats -= num_tickets
171                 booking = Booking(event, num_tickets, array_of_customer)
172                 self.bookings.add(booking)
173                 print("Booking successful!")
174                 return booking
175             else:
176                 print("Ticket unavailable!")
177                 raise EventNotFoundException("Event not found in menu.")
178         else:
179             raise EventNotFoundException("Event not found in menu.")
180
181     def cancel_booking(self, booking_id):
182         booking = next((b for b in self.bookings if b.booking_id == booking_id), None)
183         if booking:
184             booking.event.available_seats += booking.num_tickets
185             self.bookings.remove(booking)
186             print("Booking canceled!")
187         else:
188             raise InvalidBookingIDException("Invalid booking ID.")
189
```

Task11:

```
4 # Service package
5 class DBUtil:
6     @staticmethod
7     def get_db_conn():
8         try:
9             connection = mysql.connector.connect(
10                 host="localhost",
11                 user="root",
12                 password="root",
13                 database="ticketbookingsystem"
14             )
15             return connection
16
17         except Error as e:
18             print(f"Error: {e}")
19             raise
```

```
# Service package
class EventServiceProviderImpl(EventServiceProvider):
    def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue):
        cursor = None
        connection = None
        try:
            connection = DBUtil.get_db_conn()
            cursor = connection.cursor()

            # Insert into the Event table
            query = "INSERT INTO event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
            cursor.execute(query, (event_name, date, time, venue.venue_id, total_seats, total_seats, ticket_price, event_type))

            connection.commit()
            return Event(event_name, date, time, venue, total_seats, ticket_price, event_type)

        except Error as e:
            print(f"Error: {e}")
            raise

        finally:
            if cursor is not None:
                cursor.close()
            if connection is not None and connection.is_connected():
                connection.close()
```

```

database.py •
database.py > EventServiceProviderImpl > get_available_no_of_tickets
87 def get_event_details(self):
88     try:
89         connection = DBUtil.get_db_conn()
90         cursor = connection.cursor(dictionary=True)
91
92         # Select all events from the event table
93         query = "SELECT * FROM event"
94         cursor.execute(query)
95
96         events = []
97         for row in cursor.fetchall():
98             venue = Venue(row['venue_name'], row['address'])
99             event = Event(row['event_name'], row['event_date'], row['event_time'],
100                          venue, row['total_seats'], row['ticket_price'], row['event_type'])
101             events.append(event)
102
103         return events
104
105     except Error as e:
106         print(f"Error: {e}")
107         raise
108
109     finally:
110         if cursor:
111             cursor.close()
112         if connection.is_connected():
113             connection.close()
114
115 def get_available_no_of_tickets(self):
116     try:
117         connection = DBUtil.get_db_conn()
118         cursor = connection.cursor()
119
120         # Select available seats from all events in the event table
121         query = "SELECT SUM(available_seats) FROM event"
122         cursor.execute(query)

```

```

database.py •
database.py > ...
126     except Error as e:
127         print(f"Error: {e}")
128         raise
129
130     finally:
131         cursor.close()
132         connection.close()
133
134
135
136 # App package
137 class TicketBookingSystem:
138     event_service_provider = EventServiceProviderImpl()
139
140     @staticmethod
141     def display_menu():
142         print("1. Create Event")
143         print("2. Get Event Details")
144         print("3. Get Available Seats")
145         print("4. Exit")
146
147     @staticmethod
148     def main():
149         while True:
150             TicketBookingSystem.display_menu()
151             choice = input("Enter your choice: ")
152
153             if choice == "1":
154                 event_name = input("Enter event name: ")
155                 date = input("Enter event date (YYYY-MM-DD): ")
156                 time = input("Enter event time (HH:MM): ")
157                 total_seats = int(input("Enter total seats: "))
158                 ticket_price = float(input("Enter ticket price: "))
159                 event_type = input("Enter event type (Movie/Concert/Sport): ")
160                 venue_name = input("Enter venue name: ")
161                 venue_address = input("Enter venue address: ")
162                 venue = Venue(venue_name, venue_address)

```

```

database.py
database.py > ...
153     if choice == "1":
154         event_name = input("Enter event name: ")
155         date = input("Enter event date (YYYY-MM-DD): ")
156         time = input("Enter event time (HH:MM): ")
157         total_seats = int(input("Enter total seats: "))
158         ticket_price = float(input("Enter ticket price: "))
159         event_type = input("Enter event type (Movie/Concert/Sport): ")
160         venue_name = input("Enter venue name: ")
161         venue_address = input("Enter venue address: ")
162         venue = Venue(venue_name, venue_address)
163         TicketBookingSystem.event_service_provider.create_event(event_name, date, time, total_seats,
164                                                                 ticket_price, event_type, venue)
165
166     elif choice == "2":
167         events = TicketBookingSystem.event_service_provider.get_event_details()
168         for event in events:
169             event.display_event_details()
170
171     elif choice == "3":
172         available_seats = TicketBookingSystem.event_service_provider.get_available_no_of_tickets()
173         print(f"Total Available Seats: {available_seats}")
174
175     elif choice == "4":
176         print("Exiting the Ticket Booking System. Goodbye!")
177         break
178
179     else:
180         print("Invalid choice. Please enter a valid option.")
181
182 if __name__ == "__main__":
183     TicketBookingSystem.main()

```

INPUT:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - - - - -
PS D:\Hexaware\Assignment\Assignment 5\python & "C:\Program Files\Python312\python.exe" "D:\Hexaware\Assignment\Assignment 5\python\database.py"
1. Create Event
2. Get Event Details
3. Get Available Seats
4. Exit
Enter your choice: 1
Enter event name: football match
Enter event date (YYYY-MM-DD): 2023-12-25
Enter event time (HH:MM): 12:00
Enter total seats: 100
Enter ticket price: 100
Enter event type (Movie/Concert/Sport): Sports
Enter venue name: Jalgaon
Enter venue address: jalgaon
1. Create Event
2. Get Event Details
3. Get Available Seats
4. Exit

```

DATABASE CHANGES:

```

SQL File 2
1 use ticketbookingsystem;
2 show tables;
3 select * from event;
4

```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	cricket	2023-12-25	07:00:00	1	100	10	1000.00	Sports	1
2	football match	2023-12-25	12:00:00	1	100	100	100.00	Sports	1