TICKET BOOKING SYSTEM

Final code:

🎯 Project Title: Ticket Booking System

📌 Objective:

To create a console-based ticket booking system for events (Movie, Sports, Concert) that supports event creation, ticket booking, booking cancellation, viewing available seats, and event details using OOP concepts, MySQL database, and layered architecture.

🏗 Architecture and Components:

1. Database Utility (getDBConn)

- Provides a connection to the MySQL database named ticketbookingsystem.

- Handles host, username, password, and database name configuration.

2. Bean Classes (Model Layer)

- Venue: Stores venue name and address.

- Event: Contains all details about an event (name, date, time, total/available seats, price, type, venue).

- Customer: Stores customer information (name, email, phone).

3. Interfaces (Service Contracts - Abstract Layer)

- IEventServiceProvider: Declares method for event creation.

- IBookingSystemServiceProvider: Inherits from IEventServiceProvider, adds booking, cancellation, seat availability, and event detail methods.

- IBookingSystemRepository: Declares low-level database operations required by the service layer.

4. Repository Implementation (BookingSystemRepositoryImpl)

Implements all database interactions including:

- Creating events with venue management (insert/select venue).
- Booking tickets for multiple customers and calculating cost.
- Cancelling bookings and updating seat count.
- Getting available seats.

Fetching full event details using a JOIN between event and venue tables.

5. Service Implementation (BookingSystemServiceImpl)

Implements IBookingSystemServiceProvider and acts as a middle layer between user inputs and database actions.Delegates all tasks to BookingSystemRepositoryImpl.

📋 **Main Menu (User Interface)**

Handles interaction via console:

- Create Event – Takes event and venue info and saves to the database.
- Book Tickets – Takes number of tickets and customer details, and books tickets if available.
- Cancel Booking – Cancels a booking using booking ID and updates seats.

- Get Available Seats – Displays number of available seats for an event.
- Get Event Details – Shows all event details including venue.
- Exit – Terminates the program.

💾 Database Tables Expected:

- venue(venue_id, venue_name, address)
- event(event_id, event_name, event_date, event_time, total_seats, available_seats, ticket_price, event_type, venue_id)
- customer(customer_id, customer_name, email, phone_number)
- booking(booking_id, num_tickets, total_cost, booking_date, customer_id, event_id)

⚙️ Technologies Used:

Language: Python

Database: MySQL

IDE: VS Code / PyCharm

Library: mysql.connector for DB access

Concepts Used:

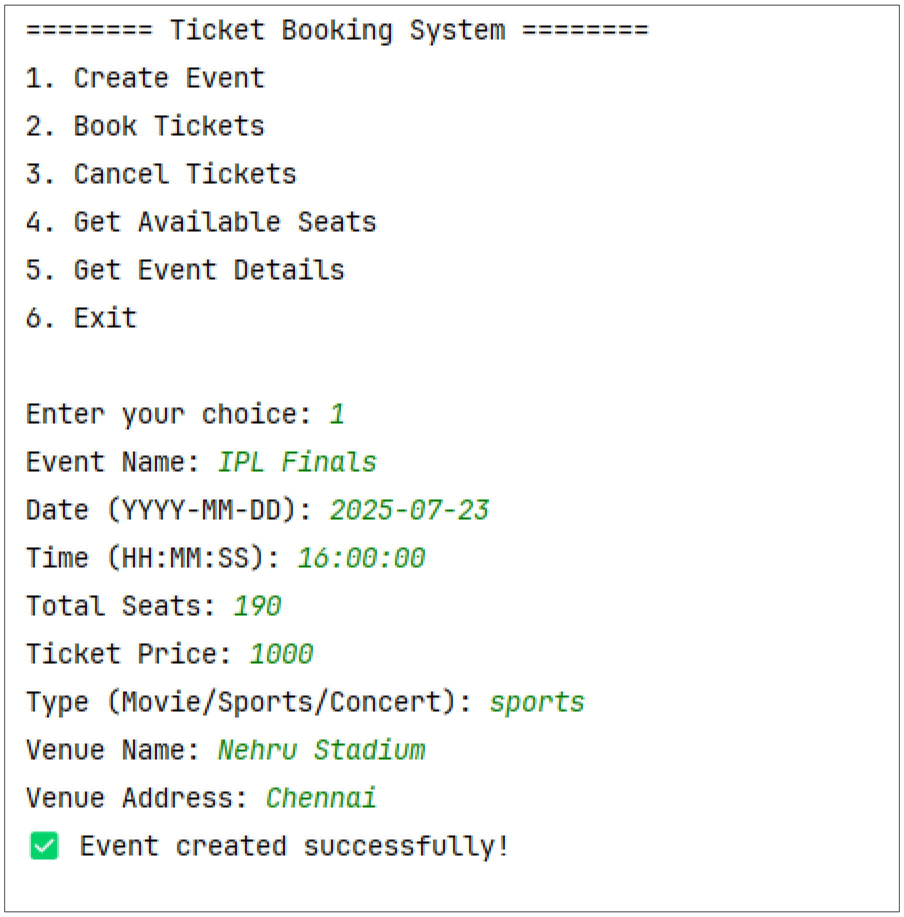
- Object-Oriented Programming (Classes, Inheritance, Abstraction)

- Layered Architecture (Model, Repository, Service)
- Exception Handling
- User Input & Menu-driven Console Interface

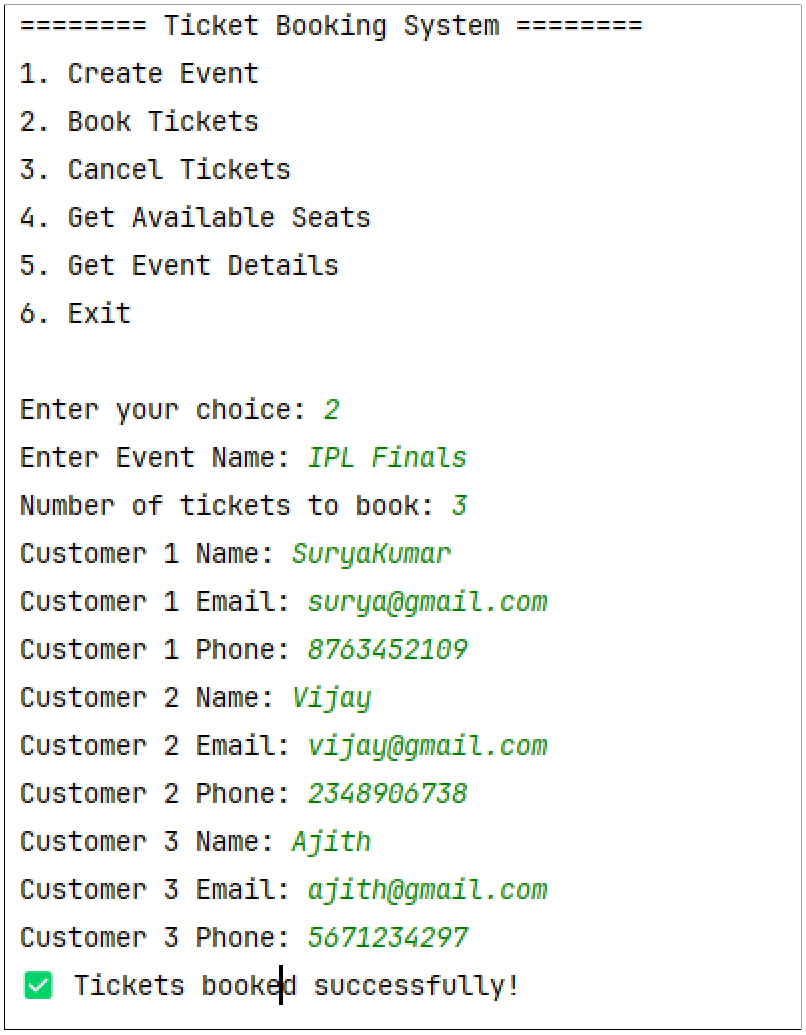
```
import mysql.connector
from datetime import datetime
from abc import ABC, abstractmethod

# DBUtil.py
def getDBConn():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="root",
        database="ticketbookingsystem"
    )

# bean/Venue.py
class Venue:
    def __init__(self, name, address):
        self.name = name
        self.address = address

# bean/Event.py
class Event:
    def __init__(self, name, date, time, total_seats, ticket_price, event_type, venue):
        self.name = name
        self.date = date
        self.time = time
        self.total_seats = total_seats
        self.available_seats = total_seats
        self.ticket_price = ticket_price
        self.event_type = event_type
        self.venue = venue

# bean/Customer.py
class Customer:
    def __init__(self, name, email=None, phone=None):
        self.name = name
        self.email = email
        self.phone = phone

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

# service/IBookingSystemServiceProvider.py
class IBookingSystemServiceProvider(IEventServiceProvider, ABC):
    @abstractmethod
    def book_tickets(self, event_name, num_tickets, customers):
        pass
    @abstractmethod
    def cancel_booking(self, booking_id):
        pass

    @abstractmethod
    def get_available_seats(self, event_name):
        pass

    @abstractmethod
    def get_event_details(self, event_name):
        pass

# service/IBookingSystemRepository.py
class IBookingSystemRepository(ABC):
    @abstractmethod
    def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue):
        pass

    @abstractmethod
    def get_event_details(self, event_name):
        pass

    @abstractmethod
    def get_available_seats(self, event_name):
        pass

    @abstractmethod
    def calculate_booking_cost(self, num_tickets, ticket_price):
        pass

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

    @abstractmethod
    def cancel_booking(self, booking_id):
        pass

# bean/BookingSystemRepositoryImpl.py
class BookingSystemRepositoryImpl(IBookingSystemRepository):
    def __init__(self):
        self.connection = getDBConn()
        self.cursor = self.connection.cursor()

    def clear_unread_results(self):
        while self.cursor.nextset():
            pass

    def create_event(self, event_name, date, time, total_seats, ticket_price, event_type, venue):
        self.cursor.execute("SELECT venue_id FROM venue WHERE venue_name = %s AND
address = %s", (venue.name, venue.address))
        venue_result = self.cursor.fetchone()

        if venue_result:
            venue_id = venue_result[0]
        else:
            self.cursor.execute("INSERT INTO venue (venue_name, address) VALUES (%s, %s)",
(venue.name, venue.address))
            self.connection.commit()
            venue_id = self.cursor.lastrowid

        self.cursor.execute("""
            INSERT INTO event (event_name, event_date, event_time, total_seats, available_seats,
ticket_price, event_type, venue_id)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
        """, (event_name, date, time, total_seats, total_seats, ticket_price, event_type, venue_id))
        self.connection.commit()
        print("☑ Event created successfully!")

    def book_tickets(self, event_name, num_tickets, customers):
        self.clear_unread_results()
        self.cursor.execute("SELECT event_id, ticket_price, available_seats FROM event WHERE
event_name = %s", (event_name,))
        event = self.cursor.fetchone()

        if not event:
            print("❌ Event not found.")
            return

        event_id, ticket_price, available_seats = event

        if num_tickets > available_seats:
            print("❌ Not enough available seats.")
            return
        for customer in customers:
            self.cursor.execute("INSERT INTO customer (customer_name, email, phone_number) VALUES (%s, %s, %s)",
                                (customer.name, customer.email, customer.phone))
            customer_id = self.cursor.lastrowid

            total_cost = self.calculate_booking_cost(1, ticket_price)
            booking_date = datetime.today().strftime('%Y-%m-%d')

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

        self.cursor.execute("UPDATE event SET available_seats = available_seats - %s WHERE event_id = %s",
                            (num_tickets, event_id))
        self.connection.commit()
        print("✅ Tickets booked successfully!")

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

    def cancel_booking(self, booking_id):
        self.cursor.execute("SELECT event_id, num_tickets FROM booking WHERE booking_id = %s", (booking_id,))
        booking = self.cursor.fetchone()

        if not booking:
            print("❌ Booking not found.")
            return

        event_id, num_tickets = booking
        self.cursor.execute("DELETE FROM booking WHERE booking_id = %s", (booking_id,))
        self.cursor.execute("UPDATE event SET available_seats = available_seats + %s WHERE event_id = %s", (num_tickets, event_id))
        self.connection.commit()
        print("✅ Booking cancelled successfully!")

    def get_available_seats(self, event_name):
        self.cursor.execute("SELECT available_seats FROM event WHERE event_name = %s", (event_name,))
        result = self.cursor.fetchone()
        if result:
            print(f"🎟️ Available seats for '{event_name}': {result[0]}")
        else:
            print("❌ Event not found.")

    def get_event_details(self, event_name):
        self.cursor.execute("""
            SELECT e.event_name, e.event_date, e.event_time, e.total_seats, e.available_seats,
                   e.ticket_price, e.event_type, v.venue_name, v.address
            FROM event e
            JOIN venue v ON e.venue_id = v.venue_id
            WHERE e.event_name = %s
        """, (event_name,))
        event = self.cursor.fetchone()
        if event:
            print("\n📌 Event Details:")
            print(f"Name: {event[0]}")
            print(f"Date: {event[1]}")
            print(f"Time: {event[2]}")
            print(f"Total Seats: {event[3]}")
            print(f"Available Seats: {event[4]}")
            print(f"Ticket Price: ₹{event[5]}")
            print(f"Type: {event[6]}")
            print(f"Venue: {event[7]}, {event[8]}\n")
        else:
            print("❌ Event not found.")

# service/BookingSystemServiceImpl.py
class BookingSystemServiceImpl(IBookingSystemServiceProvider):
    def __init__(self):
        self.repo = BookingSystemRepositoryImpl()

    def create_event(self, *args, **kwargs):
        self.repo.create_event(*args, **kwargs)

    def book_tickets(self, *args, **kwargs):
        self.repo.book_tickets(*args, **kwargs)

    def cancel_booking(self, booking_id):
        self.repo.cancel_booking(booking_id)

    def get_available_seats(self, event_name):
        self.repo.get_available_seats(event_name)

    def get_event_details(self, event_name):
        self.repo.get_event_details(event_name)

# TicketBookingSystem.py (Main Menu)
def main():
    service = BookingSystemServiceImpl()

    while True:
        print("""
======== Ticket Booking System ========
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Exit
        """)
        try:
            choice = int(input("Enter your choice: "))
        except ValueError:
            print("❌ Invalid input. Please enter a number.")
            continue

        if choice == 1:
            event_name = input("Event Name: ")
            date = input("Date (YYYY-MM-DD): ")
            time = input("Time (HH:MM:SS): ")
            total_seats = int(input("Total Seats: "))
            ticket_price = float(input("Ticket Price: "))
            event_type = input("Type (Movie/Sports/Concert): ")
            venue_name = input("Venue Name: ")
            venue_address = input("Venue Address: ")
            venue = Venue(venue_name, venue_address)
            service.create_event(event_name, date, time, total_seats, ticket_price, event_type, venue)

        elif choice == 2:
            ename = input("Enter Event Name: ")
            n = int(input("Number of tickets to book: "))
            customers = []
            for i in range(n):
                cname = input(f"Customer {i+1} Name: ")
                email = input(f"Customer {i+1} Email: ")
                phone = input(f"Customer {i+1} Phone: ")
                customers.append(Customer(cname, email, phone))
            service.book_tickets(ename, n, customers)

        elif choice == 3:
            booking_id = int(input("Enter Booking ID to cancel: "))
            service.cancel_booking(booking_id)

        elif choice == 4:
            ename = input("Enter Event Name: ")
            service.get_available_seats(ename)
```

```
        elif choice == 5:
            ename = input("Enter Event Name: ")
            service.get_event_details(ename)

        elif choice == 6:
            print("👋 Exiting the system...")
            break

        else:
            print("❌ Invalid choice. Please select from the menu options.")

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

STEP 1:CREATING THE EVENT:

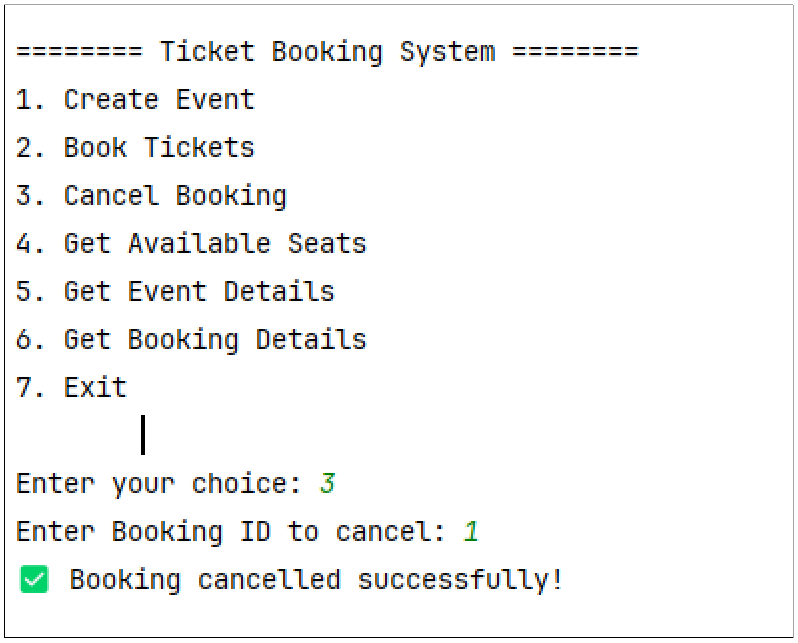

```
======== Ticket Booking System ========
1. Create Event
2. Book Tickets
3. Cancel Tickets
4. Get Available Seats
5. Get Event Details
6. Exit

Enter your choice: 1
Event Name: IPL Finals
Date (YYYY-MM-DD): 2025-07-23
Time (HH:MM:SS): 16:00:00
Total Seats: 190
Ticket Price: 1000
Type (Movie/Sports/Concert): sports
Venue Name: Nehru Stadium
Venue Address: Chennai
✅ Event created successfully!
```

STEP 2:BOOKING TICKETS

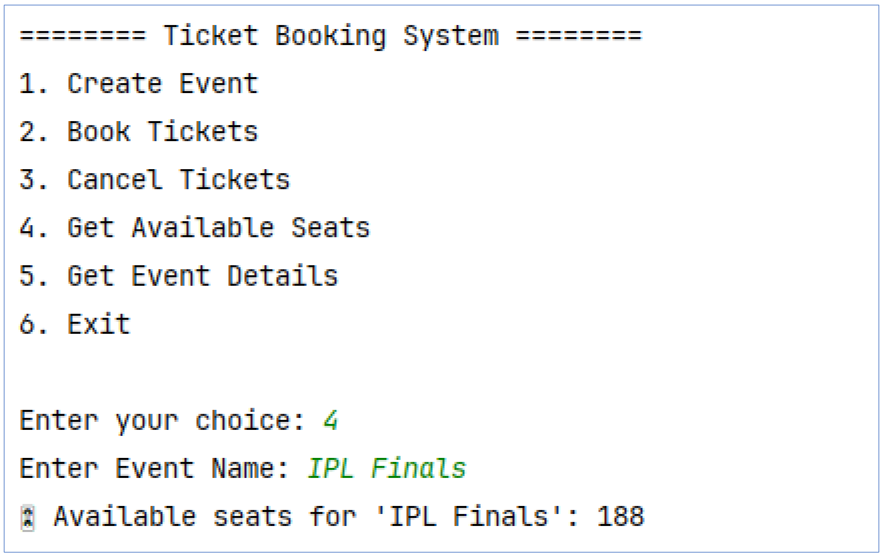


```
======== Ticket Booking System ========
1. Create Event
2. Book Tickets
3. Cancel Tickets
4. Get Available Seats
5. Get Event Details
6. Exit

Enter your choice: 2
Enter Event Name: IPL Finals
Number of tickets to book: 3
Customer 1 Name: SuryaKumar
Customer 1 Email: surya@gmail.com
Customer 1 Phone: 8763452109
Customer 2 Name: Vijay
Customer 2 Email: vijay@gmail.com
Customer 2 Phone: 2348906738
Customer 3 Name: Ajith
Customer 3 Email: ajith@gmail.com
Customer 3 Phone: 5671234297
✅ Tickets booked successfully!
```

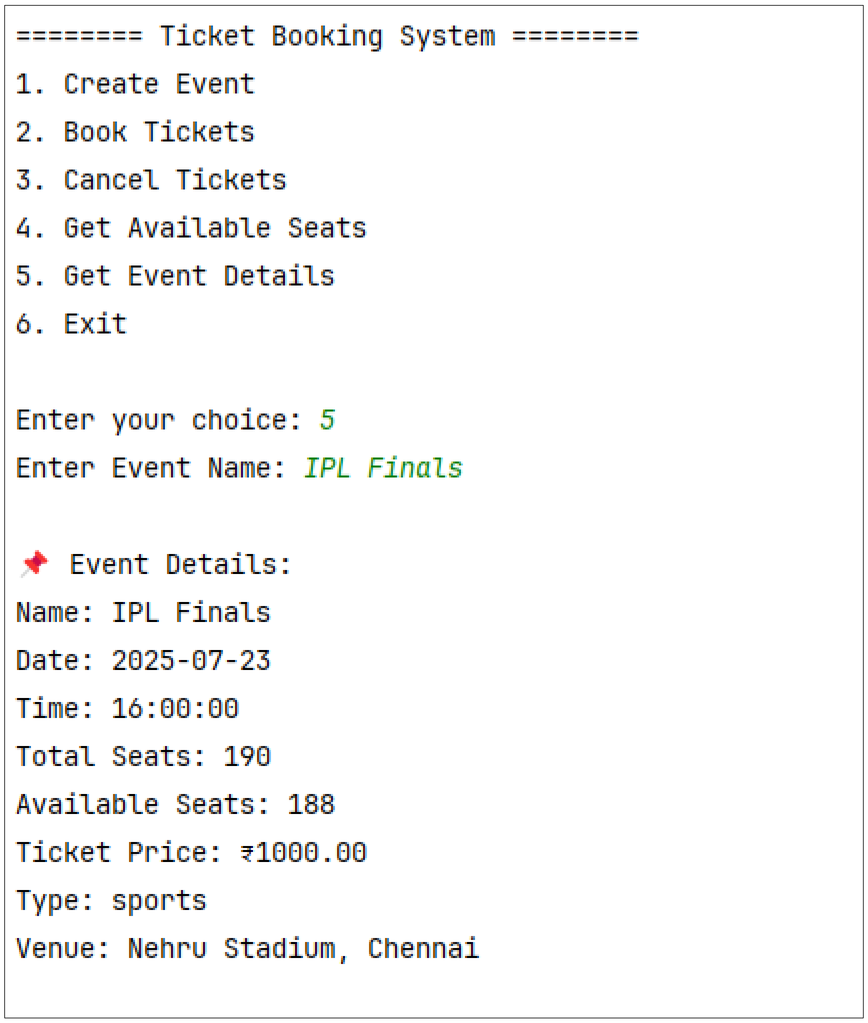
STEP 3: TICKET CANCELLING


```
======== Ticket Booking System ========
1. Create Event
2. Book Tickets
3. Cancel Booking
4. Get Available Seats
5. Get Event Details
6. Get Booking Details
7. Exit

Enter your choice: 3
Enter Booking ID to cancel: 1
✅ Booking cancelled successfully!
```

STEP 4:GETTING AVAILABLE SEATS

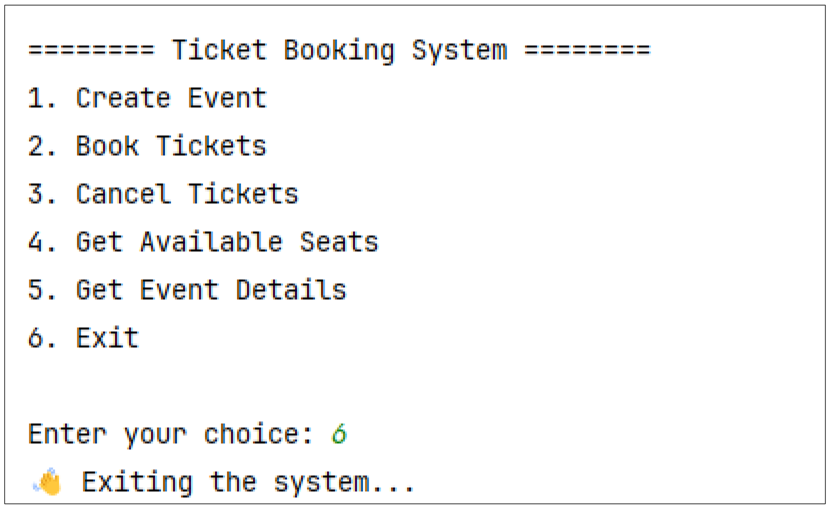

```
======== Ticket Booking System ========
1. Create Event
2. Book Tickets
3. Cancel Tickets
4. Get Available Seats
5. Get Event Details
6. Exit

Enter your choice: 4
Enter Event Name: IPL Finals
Available seats for 'IPL Finals': 188
```

STEP 5: GETTING EVENT DETAILS



```
======== Ticket Booking System ========
1. Create Event
2. Book Tickets
3. Cancel Tickets
4. Get Available Seats
5. Get Event Details
6. Exit

Enter your choice: 5
Enter Event Name: IPL Finals

📌 Event Details:
Name: IPL Finals
Date: 2025-07-23
Time: 16:00:00
Total Seats: 190
Available Seats: 188
Ticket Price: ₹1000.00
Type: sports
Venue: Nehru Stadium, Chennai
```

STEP 6:EXIT



```
======== Ticket Booking System ========
1. Create Event
2. Book Tickets
3. Cancel Tickets
4. Get Available Seats
5. Get Event Details
6. Exit

Enter your choice: 6
👋 Exiting the system...
```