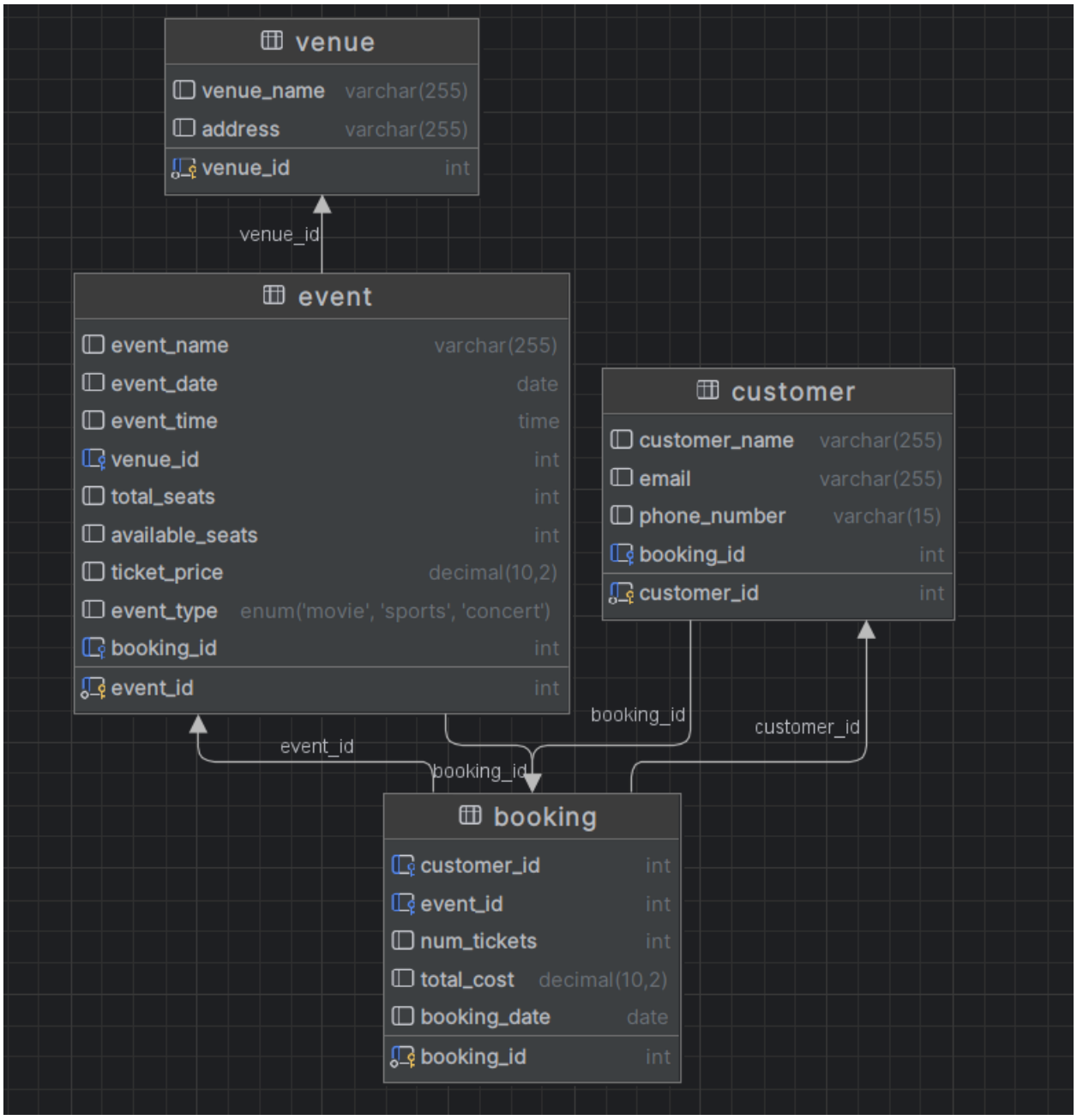
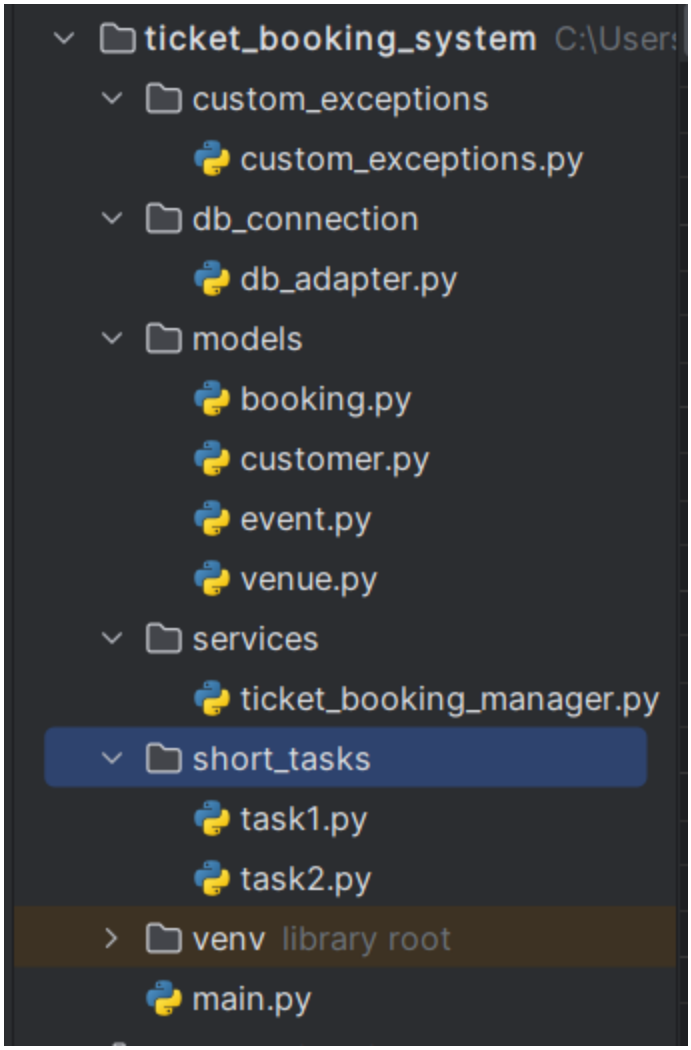
Ticket Booking System

Database Schema



File Structure



Custom_exceptions.py

```
class EventNotFoundException(Exception):
    def __init__(self, event_id):
        self.event_id = event_id
        self.message = f"Event with ID '{event_id}' not found in the menu."

class InvalidBookingIDException(Exception):
    def __init__(self, booking_id):
        self.booking_id = booking_id
        self.message = f"Invalid booking ID '{booking_id}'. Booking not found."
```

Db_adapter.py

```
import mysql.connector
```

```
def get_db_connection():
    config = {
        'user': 'root',
        'password': 'prakhar123',
        'host': 'localhost',
        'database': 'ticketbookingsystem'
    }

    try:
        connection = mysql.connector.connect(**config)
        # print("Connected to the database")
        # print('hello World')
        return connection
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return None


def get_ids(table_name, id_column_name):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT ' + id_column_name + ' FROM ' + table_name + ' ORDER BY ' +
id_column_name + ' DESC LIMIT 1'
    # print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    return int(x) + 1


def get_cnts(table_name, id_column_name, column_id):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT count(*) as count FROM ' + table_name + ' WHERE ' +
id_column_name + '=' + column_id
    # print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    return int(x)

def get_counts(table_name, id_column_name1, id_column_name2, column_id1,
column_id2):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT count(*) as count FROM ' + table_name + ' WHERE ' +
id_column_name1 + '=' + column_id1 + ' AND ' + id_column_name2 + '=' + '"' +
str(column_id2) + '"'
    # print(sql)
    my_cursor.execute(sql)
```

```
    x = list(my_cursor.fetchone())[0]
    # print(x)
    return int(x)
```

Booking.py

```
from db_connection.db_adapter import *
from models.event import Event


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

    def get_booking_id(self):
        return self.booking_id

    def get_customer_id(self):
        return self.customer_id

    def get_event_id(self):
        return self.event_id

    def get_num_tickets(self):
        return self.num_tickets

    def get_total_cost(self):
        return self.total_cost

    def get_booking_date(self):
        return self.booking_date

    def calculate_booking_cost(self, num_tickets, event_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT total_price from event where event_id = %s
            '''
            para = (event_id,)
            my_cursor.execute(sql, para)
            x = int(list(my_cursor.fetchone())[0])
```

```
            return x*num_tickets
        except Exception as e:
            print(f'An error occurred: {e}')

    def get_available_no_of_tickets(self, event_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT available_seats from event where event_id = %s
            '''
            para = (event_id,)
            my_cursor.execute(sql, para)
            x = list(my_cursor.fetchone())[0]
            return x
        except Exception as e:
            print(f'An error occurred: {e}')

    def get_event_details(self, event_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT * from event where event_id = %s
            '''
            para = (event_id,)
            my_cursor.execute(sql, para)
            x = Event(*list(my_cursor.fetchone()))
            x.display_event_details()
        except Exception as e:
            print(f'An error occurred: {e}')

    def update_booking_info(self, num_tickets=None, total_cost=None,
booking_date=None):
        my_cursor = self.connection.cursor()

        if num_tickets:
            sql = 'UPDATE booking SET num_tickets = %s WHERE booking_id = %s'
            para = (num_tickets, self.booking_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Number of Tickets Updated successfully')

        if total_cost:
            sql = 'UPDATE booking SET total_cost = %s WHERE booking_id = %s'
            para = (total_cost, self.booking_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Total Cost Updated successfully')

        if booking_date:
```

```
            sql = 'UPDATE booking SET booking_date = %s WHERE booking_id = %s'
            para = (booking_date, self.booking_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Booking Date Updated successfully')

        print('Booking details updated successfully')
```

Customer.py

```
from db_connection.db_adapter import *

class Customer:
    def __init__(self, customer_id, customer_name, email, phone_number,
booking_id):
        self.connection = get_db_connection()
        self.customer_id = customer_id
        self.customer_name = customer_name
        self.email = email
        self.phone_number = phone_number
        self.booking_id = booking_id

    # Getter and setter methods
    def get_customer_id(self):
        return self.customer_id

    def get_customer_name(self):
        return self.customer_name

    def get_email(self):
        return self.email

    def get_phone_number(self):
        return self.phone_number

    def get_booking_id(self):
        return self.booking_id

    def display_customer_details(self):
        print(f"Customer ID: {self.customer_id}")
        print(f"Customer Name: {self.customer_name}")
        print(f"Email: {self.email}")
        print(f"Phone Number: {self.phone_number}")
        print(f"Booking ID: {self.booking_id}")

    def update_customer_info(self, customer_name=None, email=None,
phone_number=None, booking_id=None):
        my_cursor = self.connection.cursor()
```

```
        if customer_name:
            sql = 'UPDATE customer SET Customer_Name = %s WHERE customer_id =
%s'
            para = (customer_name, self.customer_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Customer Name Updated successfully')

        if email:
            sql = 'UPDATE customer SET Email = %s WHERE customer_id = %s'
            para = (email, self.customer_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Email Updated successfully')

        if phone_number:
            sql = 'UPDATE customer SET Phone_Number = %s WHERE customer_id = %s'
            para = (phone_number, self.customer_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Phone Number Updated successfully')

        if booking_id:
            sql = 'UPDATE customer SET booking_id = %s WHERE customer_id = %s'
            para = (booking_id, self.customer_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Booking ID Updated successfully')

        print('Customer details updated successfully')
```

Event.py

```
from datetime import date

from db_connection.db_adapter import *

class Event:
   def __init__(self, event_id, event_name, event_date, event_time, venue_id,
total_seats, available_seats, ticket_price, event_type, booking_id):
        self.connection = get_db_connection()
        self.event_id = event_id
        self.event_name = event_name
        self.event_date = event_date
        self.event_time = event_time
        self.venue_id = venue_id
        self.total_seats = total_seats
```

```
        self.available_seats = available_seats
        self.ticket_price = ticket_price
        self.event_type = event_type
        self.booking_id = booking_id

    def get_event_id(self):
        return self.event_id

    def get_event_name(self):
        return self.event_name

    def get_event_date(self):
        return self.event_date

    def get_event_time(self):
        return self.event_time

    def get_venue_id(self):
        return self.venue_id

    def get_total_seats(self):
        return self.total_seats

    def get_available_seats(self):
        return self.available_seats

    def get_ticket_price(self):
        return self.ticket_price

    def get_event_type(self):
        return self.event_type

    def get_booking_id(self):
        return self.booking_id

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

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

    def book_tickets(self, customer_id, num_tickets, event_id, ticket_price):
        if num_tickets>self.available_seats:
            print('Insufficient available seats')
            return
        try:
            my_cursor = self.connection.cursor()
            sql1 = '''
```

```
            INSERT INTO booking(booking_id, customer_id, event_id, num_tickets,
total_cost, booking_date)
            VALUES(%s, %s, %s, %s, %s, %s)
            '''
            para1 = (get_ids('booking', 'booking_id'), customer_id, event_id,
num_tickets, num_tickets*ticket_price, date.today())
            my_cursor.execute(sql1, para1)
            sql2 = '''
            UPDATE event SET available_seats = available_seats - %s WHERE
event_id = %s'''
            para2 = (num_tickets, event_id)
            my_cursor.execute(sql2, para2)

self.update_event_info(available_seats=self.available_seats-num_tickets)
            self.connection.commit()
            print('Ticket Booked successfully')
        except Exception as e:
            print(f'An error occurred: {e}')

    def cancel_booking(self, booking_id):
        try:
            my_cursor = self.connection.cursor()
            my_cursor.callproc('cancel_booking', [booking_id])
            self.connection.commit()
            print('Ticket Cancelled successfully')
        except Exception as e:
            print(f'An error occurred: {e}')


    def display_event_details(self):
        print(f"Event Name: {self.event_name}")
        print(f"Event Date: {self.event_date}")
        print(f"Event Time: {self.event_time}")
        print(f"Venue ID: {self.venue_id}")
        print(f"Total Seats: {self.total_seats}")
        print(f"Available Seats: {self.available_seats}")
        print(f"Ticket Price: {self.ticket_price}")
        print(f"Event Type: {self.event_type}")
        print(f'Booking ID: {self.booking_id}')

    def update_event_info(self, event_name=None, event_date=None,
event_time=None, venue_id=None,
                          total_seats=None, available_seats=None,
ticket_price=None, event_type=None):
        my_cursor = self.connection.cursor()

        if event_name:
            sql = 'UPDATE event SET Event_Name = %s WHERE event_id = %s'
            para = (event_name, self.event_id)
```

```
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Event Name Updated successfully')

        if event_date:
            sql = 'UPDATE event SET Event_Date = %s WHERE event_id = %s'
            para = (event_date, self.event_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Event Date Updated successfully')

        if event_time:
            sql = 'UPDATE event SET Event_Time = %s WHERE event_id = %s'
            para = (event_time, self.event_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Event Time Updated successfully')

        if venue_id:
            sql = 'UPDATE event SET Venue_id = %s WHERE event_id = %s'
            para = (venue_id, self.event_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Venue IF Updated successfully')

        if total_seats:
            sql = 'UPDATE event SET Total_Seats = %s WHERE event_id = %s'
            para = (total_seats, self.event_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Total Seats Updated successfully')

        if available_seats:
            sql = 'UPDATE event SET Available_Seats = %s WHERE event_id = %s'
            para = (available_seats, self.event_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Available Seats Updated successfully')

        if ticket_price:
            sql = 'UPDATE event SET Ticket_Price = %s WHERE event_id = %s'
            para = (ticket_price, self.event_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Ticket Price Updated successfully')

        if event_type:
            sql = 'UPDATE event SET Event_Type = %s WHERE event_id = %s'
            para = (event_type, self.event_id)
```

```
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Event Type Updated successfully')

        print('Event details updated successfully')
```

Venue.py

```
from db_connection.db_adapter import *


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

    def get_venue_id(self):
        return self.venue_id

    def get_venue_name(self):
        return self.venue_name

    def get_address(self):
        return self.address

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

    def update_venue_info(self, venue_name=None, address=None):
        my_cursor = self.connection.cursor()

        if venue_name:
            sql = 'UPDATE venue SET Venue_Name = %s WHERE venue_id = %s'
            para = (venue_name, self.venue_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Venue Name Updated successfully')

        if address:
            sql = 'UPDATE venue SET Address = %s WHERE venue_id = %s'
            para = (address, self.venue_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Address Updated successfully')
```

```
        print('Venue details updated successfully')
```

Ticket_booking_manager.py

```
from db_connection.db_adapter import *
from models.event import Event


class BookingSystem:
    def __init__(self):
        self.connection = get_db_connection()

    def display_menu(self):
        print("1. Book Tickets")
        print("2. Cancel Booking")
        print("3. View Event Details")
        print("4. Exit")

    def book_tickets(self):
        num_tickets = int(input("Enter the number of tickets to book: "))
        customer_id = int(input('Enter the customer id: '))
        event_id = input('Enter the event ID: ')
        temp_event = self.get_event_by_id(event_id)
        try:
            temp_event.book_tickets(customer_id, num_tickets, event_id,
temp_event.get_ticket_price())
            print('Ticket Booked Successfully')
        except mysql.connector.Error as err:
            print(f"Error: {err}")

    def get_event_by_id(self, event_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT * from event where event_id = %s
            '''
            para = (event_id,)
            my_cursor.execute(sql, para)
            x = Event(*list(my_cursor.fetchone()))
            return x
        except Exception as e:
            print(f'An error occurred: {e}')

    def cancel_booking(self):
        event_id = input('Enter the event ID booking to cancel: ')
        booking_id = input("Enter the booking ID to cancel: ")
        temp_event = self.get_event_by_id(event_id)
```

```
        try:
            temp_event.cancel_booking(booking_id)
        except mysql.connector.Error as err:
            print(f"Error: {err}")

    def view_event_details(self):
        event_id = input("Enter the event ID to view details: ")

        try:
            my_cursor = self.connection.cursor()
            sql = '''
            SELECT * from event where event_id = %s
            '''
            para = (event_id,)
            my_cursor.execute(sql, para)
            t = list(my_cursor.fetchone())
            x = Event(*t)
            x.display_event_details()
        except mysql.connector.Error as err:
            print(f"Error: {err}")
```

Main.py (entry point)

```
from services.ticket_booking_manager import *

if __name__ == "__main__":
    booking_system = BookingSystem()

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

        if choice == "1":
            booking_system.book_tickets()

        elif choice == "2":
            booking_system.cancel_booking()

        elif choice == "3":
            booking_system.view_event_details()

        elif choice == "4":
            print("Exiting the Booking System.")
            break

        else:
            print("Invalid choice. Please enter a number between 1 and 4.")
```

Output

```
1. Book Tickets
2. Cancel Booking
3. View Event Details
4. Exit
Enter your choice (1-4):
```

```
Enter your choice (1-4): 3
Enter the event ID to view details: 1
Event Name: Movie Night
Event Date: 2023-12-15
Event Time: 18:00:00
Venue ID: 4
Total Seats: 100
Available Seats: 90
Ticket Price: 10.00
Event Type: Movie
Booking ID: 1
```

```
Enter your choice (1-4): 1
Enter the number of tickets to book: 10
Enter the customer id: 1
Enter the event ID: 1
Available Seats Updated successfully
Event details updated successfully
Ticket Booked successfully
Ticket Booked Successfully
```

```
Enter your choice (1-4): 2
Enter the event ID booking to cancel: 1
Enter the booking ID to cancel: 11
Ticket Cancelled successfully
```

```
1. Book Tickets
2. Cancel Booking
3. View Event Details
4. Exit
Enter your choice (1-4): 4
Exiting the Booking System.
```