Helwan University Cairo

Faculty of Computers and Artificial Intelligence

Department of Computer Science

CS471 - Parallel processing Project documentation

Multi-threading simulation on Movie Booking System

Submitted to: Dr. Ahmed Hesham Mostafa

Discussion Date: December 15, 2024

Presented by

Fatma Amr 20220332
Doaa Karem 20220148
Rahma Mohammed 20220158
Ahmed Amr 20220029
Rana Ayman 20220164

Abstract

This project simulates a cinema ticket booking system, incorporating both sequential and multithreaded processing. Using Tkinter for the graphical user interface, SQLite for data management, and Python's threading module for concurrency, the system allows multiple users to book tickets simultaneously. The project also includes CRUD (Create, Read, Update, Delete) operations for managing movie and booking data and features authentication for the admin. The system mimics the real-world scenario of managing cinema ticket sales while exploring multithreading, ensuring smooth and concurrent user interactions.

1. Introduction

1.1.Overview of the Project

The Movie Booking System is an interactive Python-based desktop application that simulates the process of booking movie tickets across multiple users. It demonstrates both sequential and parallel processing techniques using the Tkinter GUI framework and SQLite for data persistence.

This project focuses on simulating real-world scenarios in ticket booking while also exploring the concepts of multithreading and synchronized access to shared resources.

1.2.Purpose and Objectives

- To demonstrate multithreaded programming.
- To provide a simple and user-friendly interface for booking.
- To utilize SQLite for lightweight database management.

---

2. Setup and Installation

2.1. Prerequisites

- Python 3.x
- Tkinter (Python's built-in GUI library)
- SQLite3 (built-in database library)

2.2. Installing Dependencies

All required libraries are included in Python's standard library. No additional installations are necessary.

2.3. Setting Up the Database

The setup_database function automatically initializes the SQLite database. When the application runs for the first time, tables for halls, bookings, and movies are created with sample data.

---

3. System Architecture and Features

Key Components

- **Tkinter GUI**: Provides a graphical interface for users.
- **SQLite Database**: Manages data for halls, movies, and bookings.
- **Multithreading**: Enables simultaneous bookings by multiple users.

Features

- **Multithreading Simulation**: Parallel user interactions with the system.
- **Sequential Simulation**: User interactions are handled one at a time.
- **Dynamic Data**: Retrieve and display movie, hall, and seat information from an SQLite database.
- **Booking System**: Book seats in real-time while managing conflicts.
- **Thread Safety**: Use of threading locks to ensure data consistency
- **Booking System**: Book seats in real-time while managing conflicts.
- **CRUD Operations**: Admin can manage movies, halls, and bookings.

---

4. Multithreading Simulation (Core Focus)

4.1.Purpose

This module simulates multiple users interacting with the movie booking system simultaneously. By using Python's threading module, it enables parallel processing, mimicking real-world scenarios where multiple users book tickets concurrently.

4.2.Key Concepts

1. Thread Creation: Each user interaction is assigned to a separate thread using threading.Thread.
2. Concurrency: Threads run simultaneously, each handling a user interaction independently.
3. Thread Safety: To prevent data inconsistencies, a threading.Lock is used to synchronize access to shared resources like the database.

4.3. Thread Safety with Locks

The self.lock instance ensures that shared resources (e.g., booking database) are accessed in a thread-safe manner. Without this, race conditions could occur where multiple threads attempt to book the same seat simultaneously.

4.4. Advantages of Multithreading in This System

- Real-Time Interaction: Users can simultaneously interact with the system without waiting for others to finish.
- Performance: By utilizing multiple threads, the system leverages CPU resources more efficiently.
- Practicality: Mimics real-world booking systems where multiple users access the platform concurrently.

5.Sequential Simulation

In sequential mode, the system processes one user interaction at a time. This is achieved by opening a window for a user, waiting for them to complete their booking, and then moving to the next user.

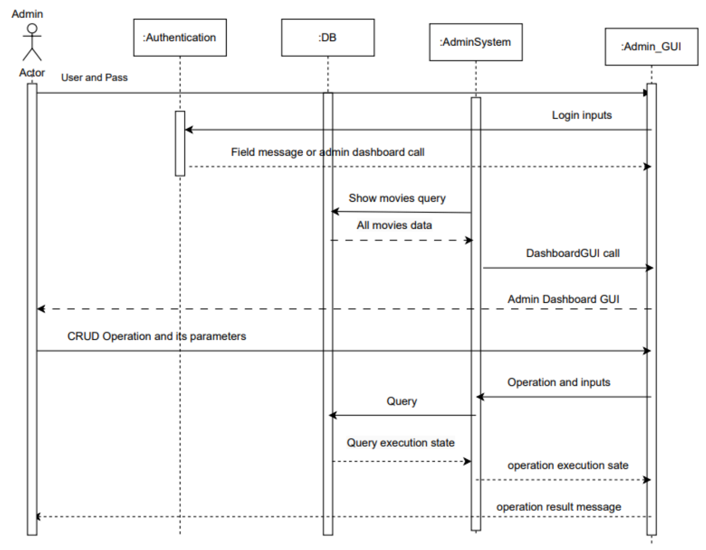
---

5. User Interaction Workflow

Admin Workflow:

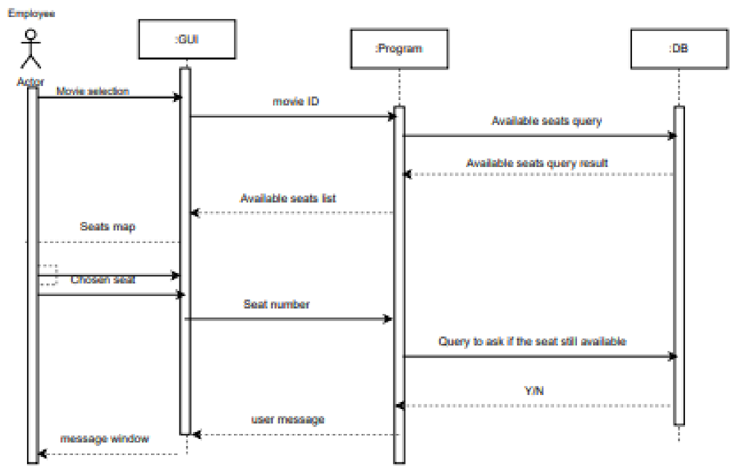
1. Admin logs in with credentials.
2. Admin can:
    - Add a new movie by specifying details like hall and timing.
    - Remove a movie using its ID.
    - Edit details of an existing movie.

3. Movies are updated in the database and displayed on the dashboard.

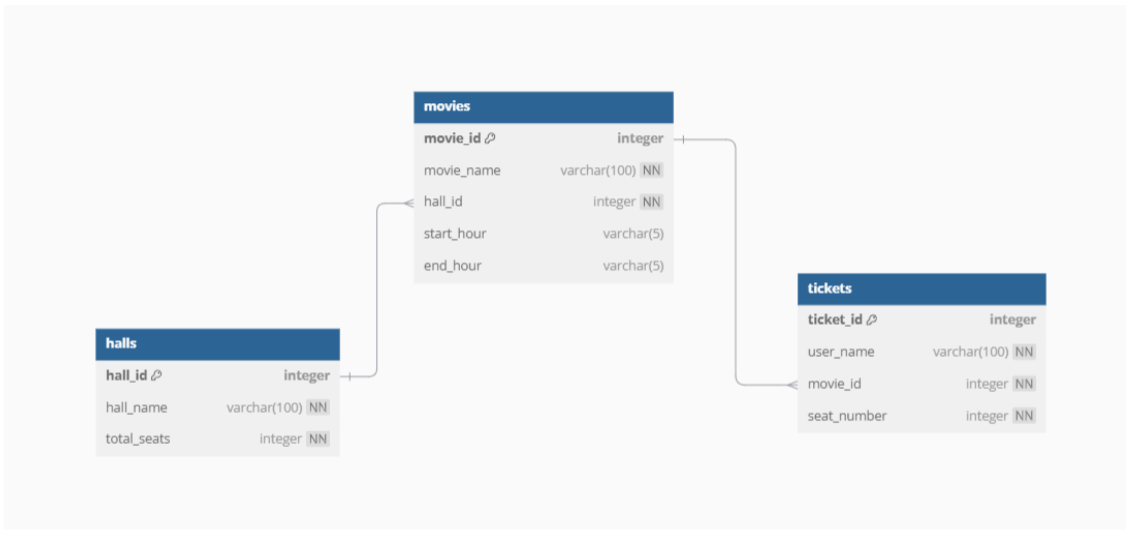


User Workflow:

1. User selects a movie from the available options.
2. User views seat availability and chooses a seat.
3. The system checks for seat conflicts and confirms the booking.



6. Database Interaction

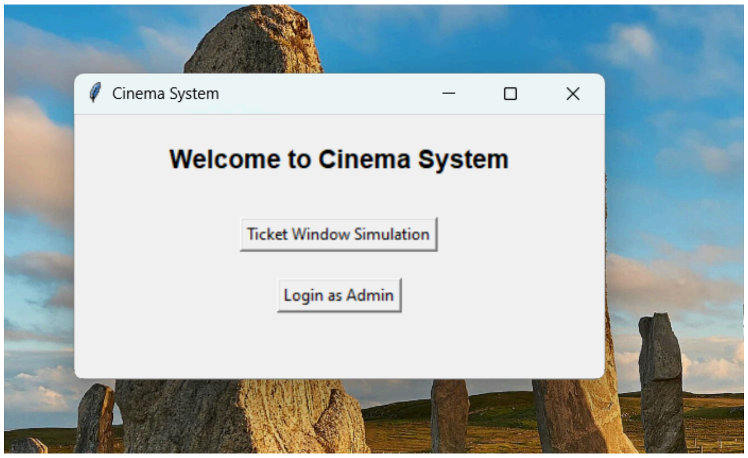


---

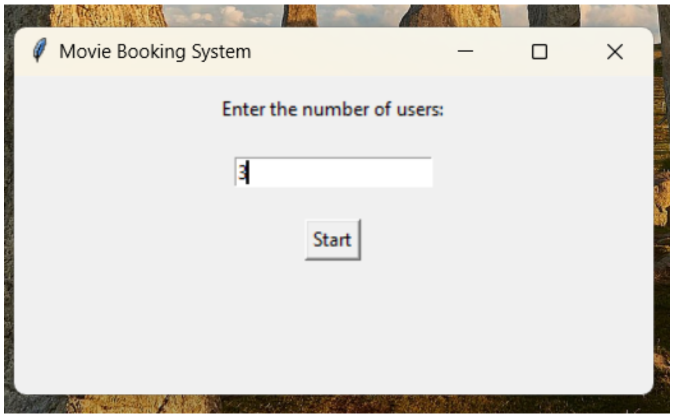
7. Sequential VS threading

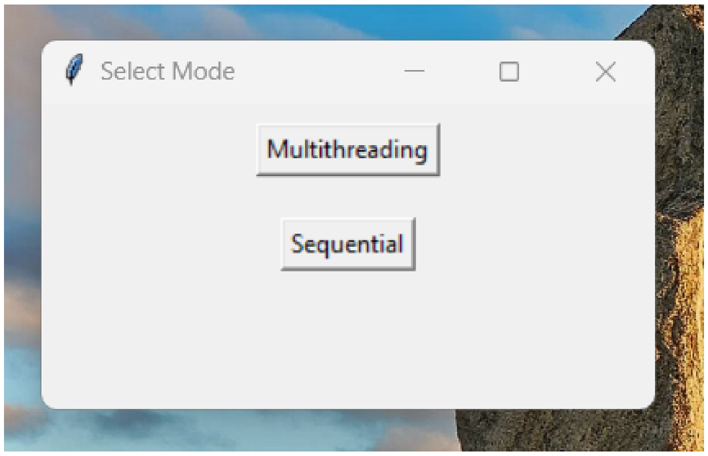
Aspect	Sequential Execution	Threading
Definition	Tasks are executed one after another in a single thread.	Multiple threads run concurrently to execute tasks.
Execution Speed	Slower, as tasks must be finished one at a time.	Faster, as multiple threads can execute simultaneously.
CPU Utilization	Underutilizes CPU resources, especially in I/O-bound tasks.	Utilizes numerous CPU cores more efficiently.
Concurrency	No concurrency; only one task runs at a time.	Allows multiple tasks to progress concurrently.
Complexity	Simpler to implement and debug.	More complex due to thread management and synchronization.
Use Case	Suitable for lightweight or non-concurrent tasks.	Ideal for tasks that can run independently or in parallel.
Resource Sharing	No need for thread synchronization.	Requires synchronization to avoid race conditions.

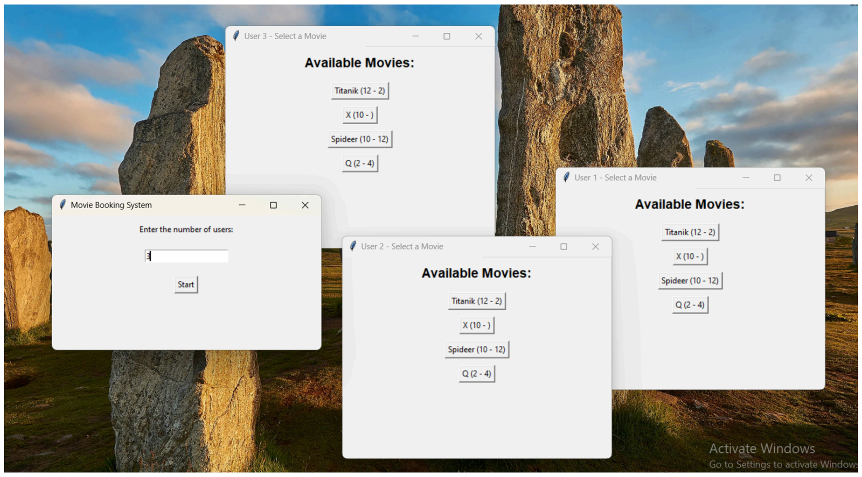
8. Demo

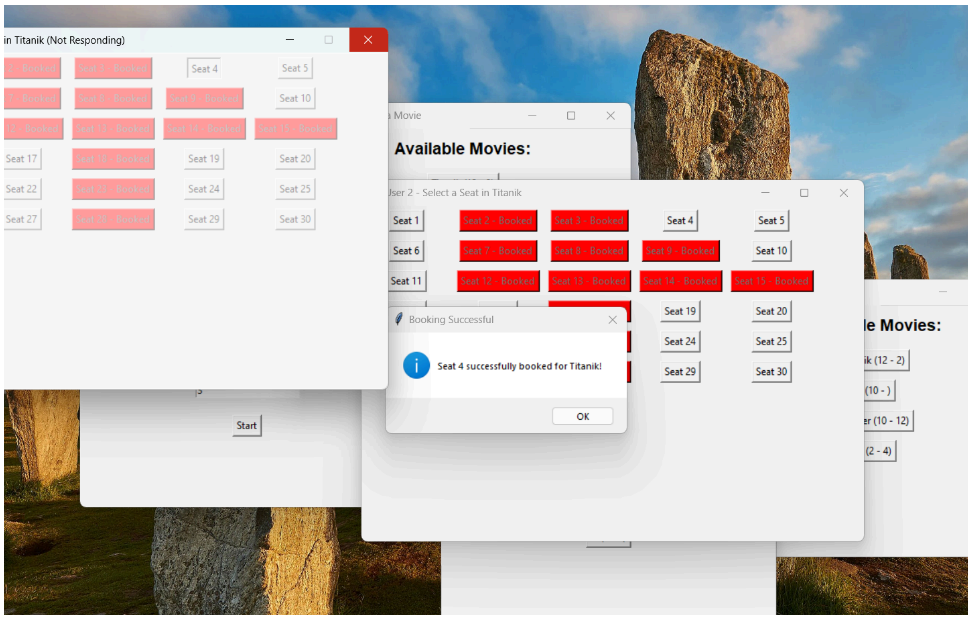


Scenario of simulation starts:



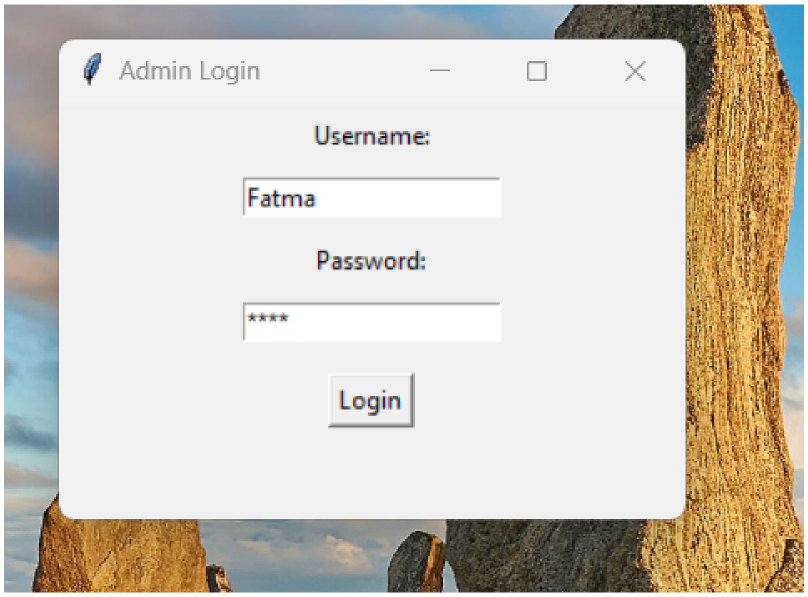






Seat 4 successfully booked for Titanik!

In case of admin login:



Admin Login

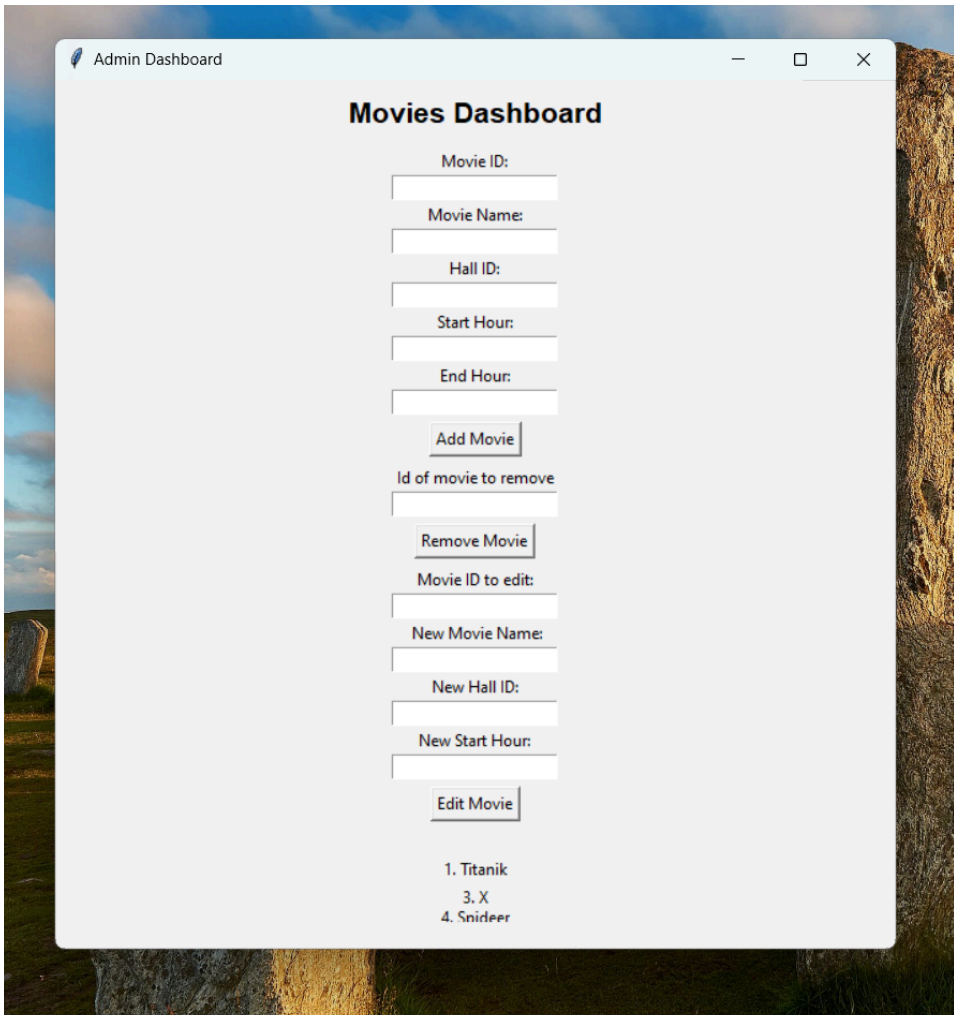
Username:

Fatma

Password:

****

Login



Admin Dashboard

Movies Dashboard

Movie ID:

Movie Name:

Hall ID:

Start Hour:

End Hour:

Add Movie

Id of movie to remove

Remove Movie

Movie ID to edit:

New Movie Name:

New Hall ID:

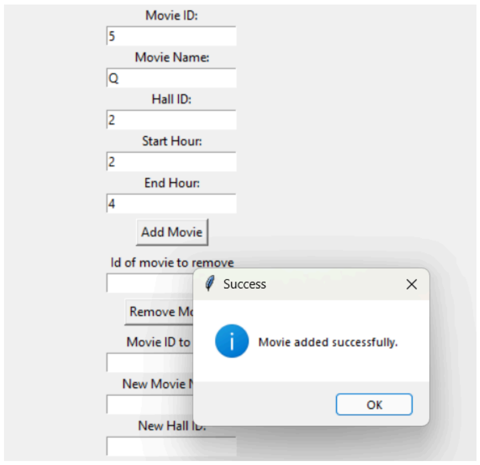
New Start Hour:

Edit Movie

1. Titanik

3. X

4. Snideer

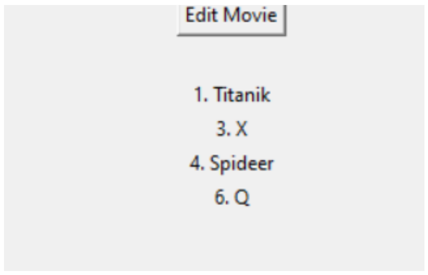


Movie ID:
5
Movie Name:
Q
Hall ID:
2
Start Hour:
2
End Hour:
4
Add Movie
Id of movie to remove
Remove Mo
Movie ID to
New Movie N
New Hall ID:

Success

Movie added successfully.

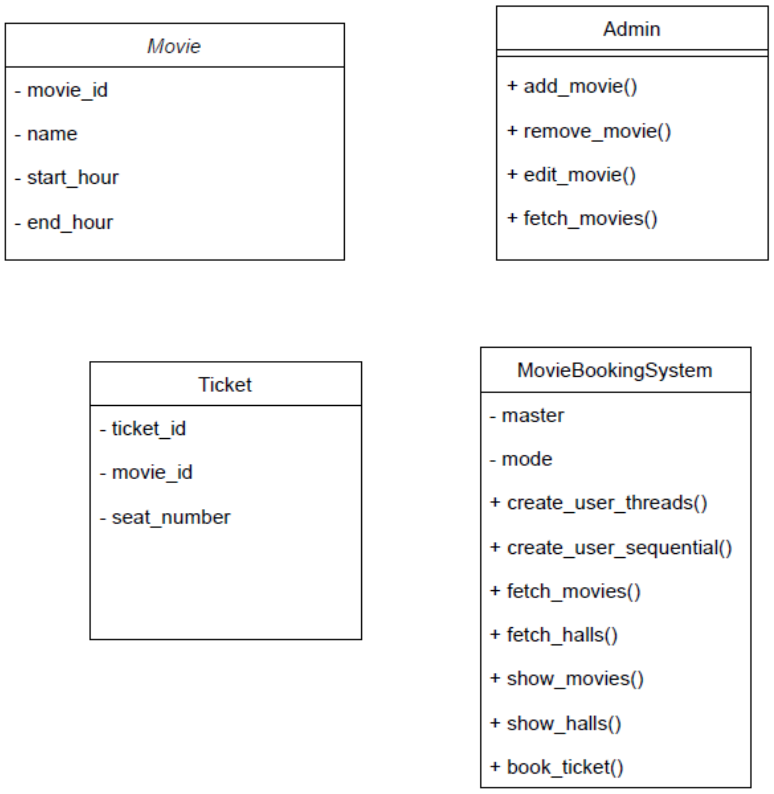
OK



Edit Movie

1. Titanik
3. X
4. Spideer
6. Q

8. Classes


Movie
- movie_id
- name
- start_hour
- end_hour
Admin
+ add_movie()
+ remove_movie()
+ edit_movie()
+ fetch_movies()
Ticket
- ticket_id
- movie_id
- seat_number
MovieBookingSystem
- master
- mode
+ create_user_threads()
+ create_user_sequential()
+ fetch_movies()
+ fetch_halls()
+ show_movies()
+ show_halls()
+ book_ticket()