AIRLINE TICKET AND SEAT ALLOCATION SYSTEM

A MINI PROJECT REPORT

Submitted by

KAVIYA J J 231901019

RITHIKA BASKAR 231901042

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

MAY 2025

BONAFIDE CERTIFICATE

Certified that this project **"AIRLINE TICKET AND SEAT ALLOCATION SYSTEM"** is the bonafide work of **"KAVIYA J J, RITHIKA BASKAR"** who carried out the project work under my supervision.

SIGNATURE

Mrs. JANANEE V

ASSISTANT PROFESSOR

Dept. of Computer Science and Engg,

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on ___________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT:

The Airline Ticket and Seat Allocation System is a mini-project designed to implement multithreading, synchronization, and file handling in an operating system environment. The system allows users to register, search for flights, book tickets, and select seats while ensuring real-time seat availability. It employs mutex locks and semaphores to prevent race conditions in seat allocation.

A multi-user interface supports concurrent bookings using threads. Flight and seat data are stored using file-based storage or an SQLite database. An admin panel enables flight management and monitoring. Users can also cancel bookings and request refunds. The project demonstrates operating system concepts like process synchronization and resource allocation. Implemented in Python, it ensures efficient and secure ticket management

ACKNOWLEDGEMENT

We express our sincere thanks to our honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of the Department **Mr. BENIDICT JAYAPRAKASH NICHOLAS M.E Ph.D.,** for being ever supporting force during our project work.

We also extend our sincere and hearty thanks to our internal guide **Mrs**.**V JANANEE,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. KAVIYA J J
2. RITHIKA BASKAR

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Airline Ticket and Seat Allocation System is developed to simulate the booking and seat allocation process in airline systems. It allows multiple users to book seats simultaneously using thread-based concurrency to reflect real-world behavior in airline software. The project incorporates OS concepts like multithreading, synchronization, mutex locks, and resource allocation.

1.2 SCOPE OF THE WORK

This system is developed to offer a secure, concurrent, and efficient ticket booking environment. It supports real-time seat checking, multi-user access, admin control, and dynamic status updates. The system is suitable for educational demonstrations and small-scale simulations of airline booking.

1.3 PROBLEM STATEMENT

Airline booking systems often face problems like double bookings and inconsistent seat data when many users access them at the same time. Without proper synchronization, these systems cannot manage shared resources safely. This project aims to solve this by using multithreading and synchronization techniques to build a secure and reliable ticket booking system.

1.4 AIM AND OBJECTIVES OF THE PROJECT

- Simulate a multi-user seat booking system.
- Prevent race conditions using mutexes and semaphores.
- Use file handling or database systems for storing flight and seat details.
- Enable users to register, book, cancel tickets, and view status.
- Provide admin-level controls for managing flight schedules.

CHAPTER 2

SYSTEM SPECIFICATIONS:

2.1 HARDWARE SPECIFICATIONS

Component	: Specification
Processor	: Intel i5
Memory Size	: 4 GB
Hard Disk	: 250 GB

2.2 SOFTWARE SPECIFICATIONS

Component	: Technology Used
Operating System	: Windows
Frontend	: HTML, CSS, JavaScript
Backend	: Python
Database	: SQLite

CHAPTER 3

MODULE DESCRIPTION

2.1. Admin Module

The admin module is responsible for managing flight data and overseeing the booking status. It allows the admin to add new flights, remove outdated ones, or modify the details of existing flights. It provides tools to monitor overall seat availability and user activity.

2.2. User Module

The user module handles all customer-facing operations. Users can register themselves in the system, log in securely, search for available flights, and proceed to book or cancel tickets. This module ensures an intuitive experience and interacts with backend systems for real-time updates.

2.3. Seat Allocation Module

This module plays a critical role in preventing double booking. It uses mutex locks to synchronize access to the seat data, ensuring that no two users can book the same seat at the same time. It provides accurate seat availability even under high concurrency.

2.4. Multithreading Module

The multithreading module is designed to simulate multiple users accessing the system at once. Threads represent individual users, and Python's threading mechanisms help demonstrate how synchronization and scheduling occur in real-time systems.

2.5. File Handling Module

This module stores all relevant information such as user details, flight schedules, and booking records. Depending on implementation, either a flat-file system or an SQLite database is used to persist data across sessions. It ensures data integrity and supports retrieval and updates.

CHAPTER 4

SOURCE CODE:

4.1. app.py

```
from flask import Flask, render_template, request, redirect, url_for
import random
import matplotlib.pyplot as plt
import os

app = Flask(__name__)
results = []

def generate_gantt_chart(jobs):
    fig, gnt = plt.subplots(figsize=(10, 5))
    gnt.set_xlabel('Time')
    gnt.set_ylabel('Jobs')
    gnt.set_yticks([15 + i * 10 for i in range(len(jobs))])
    gnt.set_yticklabels([job['name'] for job in jobs])
    gnt.grid(True)

    start_time = 0
    for i, job in enumerate(jobs):
        gnt.broken_barh([(start_time, job['burst_time'])], (10 + i * 10, 9), facecolors='tab:blue')
        job['completion_time'] = start_time + job['burst_time']
        start_time += job['burst_time']

    plt.tight_layout()
    plt.savefig(os.path.join("static", "gantt_chart.png"))
    plt.close()

def generate_line_graph(jobs):
```

```
    job_names = [job['name'] for job in jobs]
    completion_times = [job['completion_time'] for job in jobs]

    plt.figure(figsize=(10, 5))
    plt.plot(job_names, completion_times, marker='o', linestyle='-', color='green')
    plt.title("Job Completion Times")
    plt.xlabel("Jobs")
    plt.ylabel("Completion Time")
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(os.path.join("static", "line_graph.png"))
    plt.close()


@app.route('/')
def welcome():
    return render_template('welcome.html')


@app.route('/book', methods=['GET', 'POST'])
def book():
    global results
    if request.method == 'POST':
        passengers = []
        names = request.form.getlist('name')
        classes = request.form.getlist('class')
        tickets = request.form.getlist('tickets')
        algorithms = request.form.getlist('algorithm')

        for name, cls, ticket_count, algo in zip(names, classes, tickets, algorithms):
            for i in range(int(ticket_count)):
                passengers.append({
                    'name': f"{name}_{i+1} ({cls})",
                    'burst_time': random.randint(1, 5),
```

```
            'algorithm': algo
        })


    business = [p for p in passengers if '(Business)' in p['name']]
    economy = [p for p in passengers if '(Economy)' in p['name']]



    # Apply algorithm-specific sort inside each class

    def sort_by_algo(passenger_list):
        sorted_list = []
        rr_time_quantum = 2
        for algo in ['FCFS', 'SJF', 'Priority', 'Round Robin']:
            temp = [p for p in passenger_list if p['algorithm'] == algo]
            if algo == 'SJF':
                temp.sort(key=lambda x: x['burst_time'])
            elif algo == 'Priority':
                for p in temp:
                    p['priority'] = random.randint(1, 10)
                temp.sort(key=lambda x: x['priority'])
            elif algo == 'Round Robin':
                 temp.sort(key=lambda x: x['name'])  # Keep order for RR; actual implementation
can be more advanced
            sorted_list.extend(temp)
        return sorted_list

    business_sorted = sort_by_algo(business)
    economy_sorted = sort_by_algo(economy)
    passengers = business_sorted + economy_sorted

    generate_gantt_chart(passengers)
    generate_line_graph(passengers)
    results = passengers
```

```
        return redirect(url_for('result'))

    return render_template('book.html')

@app.route('/result')
def result():
    return render_template('result.html', jobs=results)

if __name__ == '__main__':
    app.run(debug=True)
```

4.2. welcome.html

```
<!DOCTYPE html>
<html>
<head><title>Welcome</title></head>
<body style="text-align:center; font-family:sans-serif;">
    <h1>Welcome to Airline Booking Scheduler</h1>
    <a href="/book"><button style="padding:10px 20px;">Start Booking</button></a>
</body>
</html>
```

4.2. book.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Book Tickets</title>
    <script>
        function addPassenger() {
            const form = document.getElementById("passenger-form");
            const newGroup = document.querySelector(".passenger-group").cloneNode(true);
            form.appendChild(document.createElement("hr"));
            form.appendChild(newGroup);
        }
```

```html
    </script>
</head>
<body style="font-family:sans-serif;">
    <h2>Enter Passenger Details</h2>
    <form method="POST" id="passenger-form">
        <div class="passenger-group">
            <label>Name: <input type="text" name="name" required></label><br>
            <label>Class:
                <select name="class">
                    <option value="Economy">Economy</option>
                    <option value="Business">Business</option>
                </select>
            </label><br>
            <label>Number of Tickets: <input type="number" name="tickets" value="1" min="1" required></label><br>
            <label>CPU Algorithm:
                <select name="algorithm">
                    <option value="FCFS">FCFS</option>
                    <option value="SJF">SJF</option>
                </select>
            </label><br>
        </div>
        <button type="button" onclick="addPassenger()">Add Another Passenger</button><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

4.3 result.html

```html
<!DOCTYPE html>
<html>
```

```
<head><title>Result</title></head>
<body style="text-align:center; font-family:sans-serif;">
    <h2>Gantt Chart</h2>
    <img src="/static/gantt_chart.png" alt="Gantt Chart" width="700"><br><br>
    <h2>Completion Time Line Graph</h2>
    <img src="/static/line_graph.png" alt="Line Graph" width="700">
</body>
</html>
```

CHAPTER 5

SCREENSHOTS

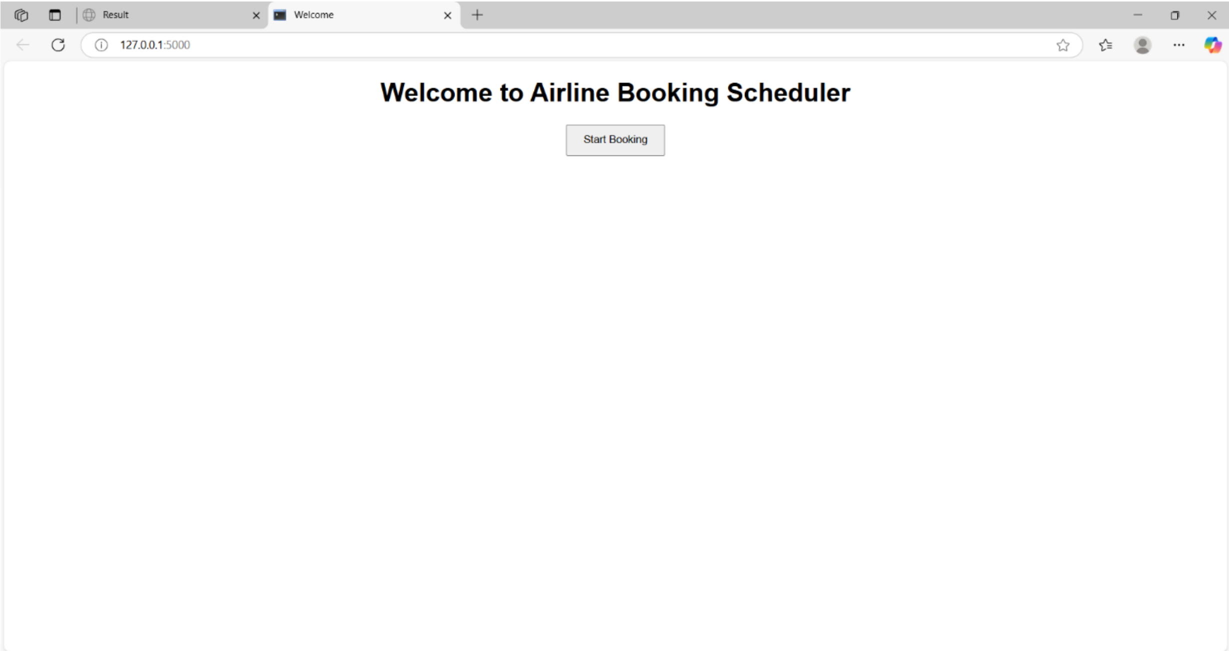

Fig 5.1 Home page

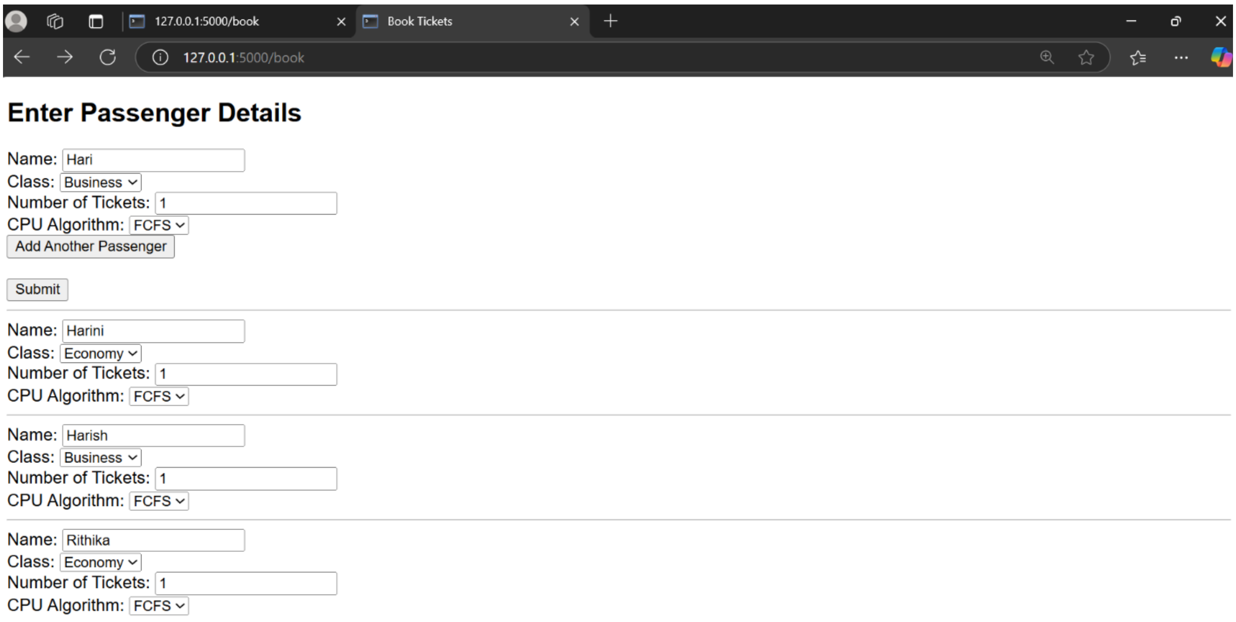

Fig 5.2 Passenger Details Input Form

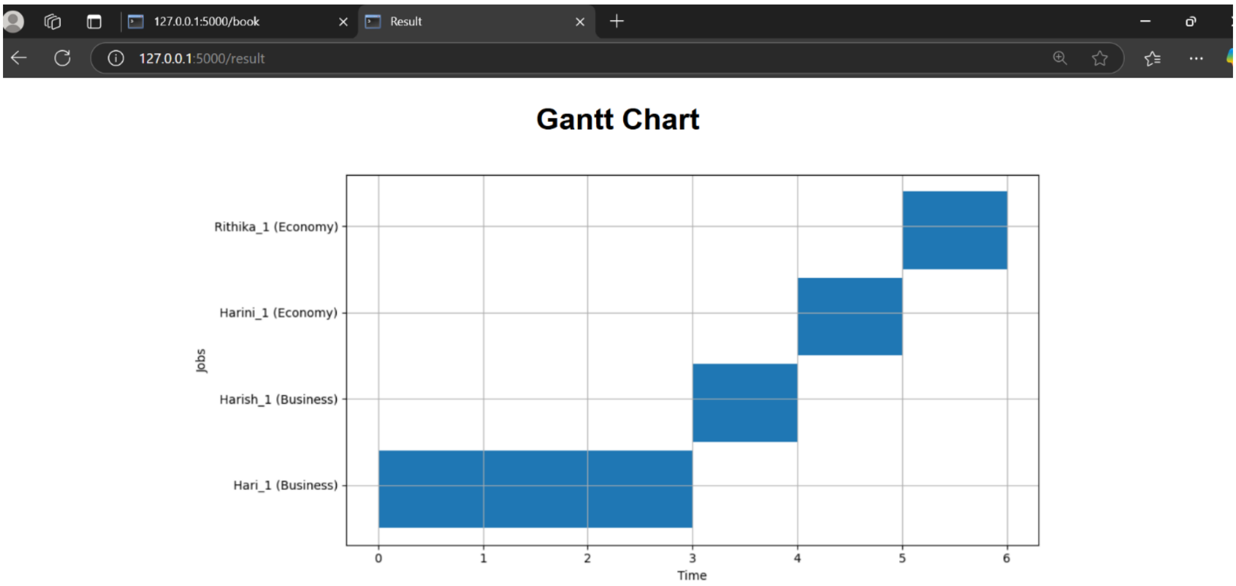

Fig 5.3 Gantt Chart Representation

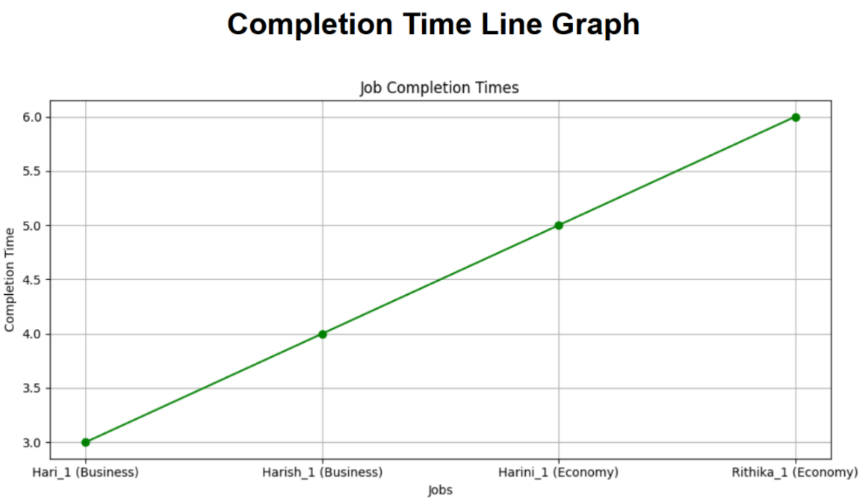

Fig 5.4 Completion Time Graph

CHAPTER 6

CONCLUSION:

The Airline Ticket and Seat Allocation System is a practical implementation of key operating system concepts such as multithreading, synchronization, and resource management. The system simulates a real-time booking environment where multiple users can book tickets without conflicts. By integrating scheduling algorithms like FCFS and SJF, the project demonstrates efficient and fair seat allocation. The use of graphs and Gantt charts provides clear visual insight into how the system handles job scheduling and completion times. Overall, the project fulfills its objective of offering a secure, user-friendly, and well-coordinated ticket booking system using Python and SQLite.

CHAPTER 7

REFERENCES:

- Operating System Concepts – Abraham Silberschatz, Peter B. Galvin, Greg Gagne
- Modern Operating Systems – Andrew S. Tanenbaum, Herbert Bos
- https://github.com/gudavinay/AirlineReservationSystem