**AIRLINE RESERVATION SYSTEM USING PYTHON**

MINOR PROJECT REPORT

By

**TANISHQ GANDHI (RA2211028010008)**

**SRIJAN SRIVASTAVA (RA2211028010065)**

Under the guidance of   
**Dr. R. Kayalvizhi**

Assistant Professor, Department of Networking and Communications*In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

in **CLOUD COMPUTING**



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203P** **ADVANCED PROGRAMMING PRACTICE** entitled in " **AIRLINE RESERVATION SYSTEM USING PYTHON** " is the bonafide work of **Tanishq Gandhi (RA2211028010008)** and **Srijan Srivastava (RA2211028010065)** who carried out the work under my supervision.

# SIGNATURE SIGNATURE

**Dr. R.Kayalvizhi Dr. ANNAPURANI K**

**ASSISTANT PROFESSOR PROFESSOR AND HEAD**

Department of Networking and Department of Networking and

Communications Communications

SRM Institute of Science and SRM Institute of Science and

Technology, Technology,

Kattankulathur , Chennai Kattankulathur , Chennai

# ABSTRACT

# The provided code implements a Flight Reservation System using Tkinter in Python, with classes representing flights, the reservation system, and a GUI for user interaction. The system allows users to view available flights, book seats, and process payments.

# Its significance lies in streamlining flight booking, offering a user-friendly graphical interface. Object-oriented design enhances modularity, and Tkinter and PIL are utilized for GUI development and image handling.

# The Flight Reservation System is crucial for efficient flight booking, improving user experience, and aiding airlines in effective booking management. This Python implementation exemplifies creating desktop applications and showcases practical programming language use.

# ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors. We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G, Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr.R.Kayalvizhi, Assistant Professor, Networking And Communications,** for herassistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **Dr. ANNAPURANI K, Professor and Head, Networking and Communications** and my Departmental Faculties for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **CONTENTS** | **PAGE NO** |
| **1** | **INTRODUCTION** |  |
|  | 1.1 Problem Statement | **7** |
|  | 1.2 Challenges | **8** |
|  | 1.3 Motivation | **10** |
| **2** | **LITERATURE SURVEY** | **12** |
| **3** | **REQUIREMENT** | **13** |
| **4** | **ARCHITECTURE & DESIGN** | **16** |
| **5** | **IMPLEMENTATION OF AIRLINE RESERVATION SYSTEM** | **19** |
| **6** | **EXPERIMENT RESULTS & ANALYSIS** | **25** |
| **7** | **CONCLUSION** | **27** |
|  | **REFERENCES** | **28** |

1. **INTRODUCTION**

The Flight Reservation System is a Python application that simplifies the process of booking and managing flight reservations. This system offers a user-friendly graphical interface built using the tkinter library, making it accessible to a wide range of users. It consists of three main classes: **Flight**, **ReservationSystem**, and **FlightReservationGUI**, each with specific functions to ensure smooth operation.

The **Flight** class is responsible for defining the attributes of each flight, such as flight number, origin, destination, departure and arrival times, price, capacity, and the number of reserved seats. It also provides methods for displaying flight information, booking seats, and calculating the total fare for a booking.

The **ReservationSystem** class acts as a central hub for managing the available flights. It allows for the addition of new flights and provides a function to check and display available flights based on user input. Users can book flights, and the class handles the booking process, checking for seat availability, and calculating the total fare.

The **FlightReservationGUI** class creates the graphical user interface, offering a user-friendly layout where users can log in, select flights, book seats, choose payment methods, and receive booking confirmations. While the system currently includes hardcoded flights, it can be expanded to connect to a real flight database for up-to-date and comprehensive flight information.

Overall, the Flight Reservation System aims to simplify the flight booking process for travelers and provide administrators with the tools to manage and update available flight options efficiently.

**1.1. PROBLEM STATEMENT**

The problem revolves around the complexity and inconvenience of booking and managing flight reservations. Travelers often struggle with finding available flights, booking seats, and ensuring a seamless payment process. Meanwhile, airlines and administrators need a reliable system for organizing and updating flight information.

The main challenges include:

* **User-Friendly Booking**: Designing a user-friendly interface for travelers to search and book flights without complications.
* **Seat Availability**: Ensuring that the system accurately reflects seat availability and prevents overbooking.
* **Payment Handling**: Managing the payment process, providing various payment options, and generating payment confirmations.
* **Flight Administration**: Offering administrators an efficient way to add, update, or remove flight information.
* **Comprehensive Flight Information**: Ensuring that the system displays accurate, up-to-date, and detailed flight information to travelers.
* **Database Integration**: Incorporating a database for storing and retrieving flight details, including destinations, origin, pricing, and availability.
* **Security**: Maintaining data security to protect user information and payment details.
* **Scalability**: Designing the system to be scalable for future expansion, integration with real airline systems, and a larger number of flights and destinations.

The goal is to develop a Flight Reservation System that addresses these challenges, streamlining the booking process for travelers and providing efficient flight management tools for administrators, ensuring a reliable, user-friendly, and secure experience for all stakeholders.

**1.2. CHALLENGES**

* **User Authentication:** Implement a more robust user authentication system. Currently, there's a hard-coded username and password for login. You can create a user database and implement secure user authentication.
* **Data Persistence:** Implement data persistence to store flight information and reservations. You can use a database or file storage to save and retrieve flight details and reservations.
* **Validation:** Add input validation to ensure that the user enters valid data when booking a flight, such as checking if the flight number exists and if the number of seats is within a reasonable range.
* **Search and Filter Flights:** Implement a feature to search and filter flights based on various criteria like origin, destination, departure time, or price range.
* **Flight Availability Update:** Implement a mechanism to automatically update the availability of flights in real-time as reservations are made, rather than just displaying available flights at the start.
* **Payment Gateway Integration:** Integrate a real payment gateway for processing payments. Currently, the system assumes successful payments.
* **Cancellation and Refunds:** Implement a feature for users to cancel their reservations
* **Admin Panel:** Create an admin panel for managing flight details, adding new flights, and viewing reservation data.
* **Error Handling:** Improve error handling by providing more informative error messages to users and handling unexpected exceptions gracefully.
* **Localization:** Make the system more user-friendly by adding support for multiple languages and currencies.
* **Responsive Design:** Ensure that the GUI is responsive and works well on different screen sizes and orientations.
* **Security:** Address security concerns, such as protecting against SQL injection and ensuring that sensitive information is stored securely.
* **Logging:** Implement logging to keep track of system activities and errors for troubleshooting and auditing.
* **Testing:** Write unit tests to ensure the correctness and reliability of the code.
* **Feedback and Notifications:** Provide feedback to users at various stages of the booking process and send notifications, such as booking confirmation emails.
* **Accessibility:** Ensure that the GUI is accessible to individuals with disabilities by following accessibility guidelines.

**1.3. MOTIVATION**

* **Convenience:** Flight reservation systems offer travelers the convenience of booking and managing their flights online. This eliminates the need to visit physical airline ticket counters, saving time and effort.
* **Efficiency:** Automated systems can process a large number of flight reservations and payments quickly and accurately. This efficiency benefits both travelers and airlines.
* **Accessibility:** Online reservation systems make it easy for people to book flights from anywhere with an internet connection, 24/7. This accessibility is especially valuable for international travelers and those with busy schedules.
* **Cost Savings:** For airlines, online booking systems can reduce administrative and operational costs associated with manual reservation processes and physical ticketing.
* **Transparency:** Users can access detailed flight information, including available seats, pricing, and flight schedules. This transparency helps travelers make informed decisions.
* **Revenue Generation:** Airlines can generate additional revenue by offering dynamic pricing based on factors like seat availability, demand, and booking lead time.
* **User Experience:** A well-designed flight reservation system enhances the overall user experience. Travelers can easily search for flights, choose their preferences, and receive booking confirmations.
* **Data Analysis:** The system collects valuable data on traveler preferences, booking patterns, and revenue sources, which airlines can analyze to optimize their services.
* **Customer Loyalty:** By providing a convenient and user-friendly booking process, airlines can build customer loyalty and encourage repeat business.
* **Competitive Advantage:** Airlines and travel agencies that offer efficient online booking systems have a competitive advantage in the travel industry. A well-designed and reliable system can set them apart from their competitors.
* **Scalability:** Online reservation systems can handle a high volume of bookings, making them suitable for both small and large airlines.
* **Global Reach:** Airlines can expand their customer base beyond their local markets and reach a global audience through online booking platforms.
* **Adaptation to Technological Trends:** The continuous evolution of technology and the growing use of mobile devices have made online flight reservations even more relevant and necessary

1. **LITERATURE SURVEY**

In developing nations, flight reservations are typically handled either manually or electronically. Regardless of the approach, the reservation and payment process is fragmented, cost-inefficient, tedious, and repetitive, resulting in wastage. We present a fully integrated airline reservation and payment system. Our system follows a Client/Proxy/Server model with an intermediate layer serving as a portability-aware core, providing continuous self-service support. This innovative solution offers airlines in developing countries a means to enhance productivity, reduce operational costs, increase revenue, and establish value-added customer service for travelers. When individuals wish to purchase flight tickets in certain countries, they typically resort to one of these methods: manual ticket purchase at the airport, obtaining a paper ticket form, filling it out manually and submitting it to the airport, completing an electronic ticket form and printing it for airport submission, or booking a ticket online at designated counters. While these methods allow for online ticket purchase to some extent, they may lack flexibility in the process. This may leave passengers dissatisfied due to the need for physical actions, such as visiting the airport for ticket acquisition. Our proposed system offers users complete flexibility, enabling them to conveniently purchase tickets from their own computers via a dedicated website. Additionally, only registered users will have the ability to book tickets, check flight timings, and initiate cancellations within our proposed system.

1. **REQUIREMENTS**

DIFFERENT APPLICATION USED FOR AIRLINE RESERVATION:

* User Authentication and Login:
  + Users should be able to log in to the system using a username and password.
  + Authentication should be validated against predefined credentials (e.g., username "Srijan" and password "srijan21").
* Flight Information Display:
  + Display flight details including flight number, origin, destination, departure time, arrival time, price, and available seats.
* View Available Flights:
  + Provide a list of available flights with information about their availability.
* Book a Flight:
  + Allow users to select a flight by entering the flight number and specifying the number of seats to book.
  + Ensure that the system checks for seat availability and does not allow overbooking.
* Calculate Total Fare:
  + Calculate the total fare based on the selected flight and the number of seats to be booked.
* Payment Options:
  + Offer payment options for users, including credit card, debit card, and net banking.
* Booking Confirmation:
  + Provide booking confirmation with the total fare amount after a successful booking.
* Display Reservation Information:
  + Show information about the reserved flight, including the flight number, departure time, and total fare.
* Notification of Successful Booking:
  + Send a notification to the user upon successful booking, confirming their reservation.

Non-Functional Requirements:

* User Interface (UI):
  + Develop a user-friendly and intuitive GUI using Tkinter for a desktop application.
* Security:
  + Ensure that user credentials are stored securely.
  + Handle user authentication securely to protect user data.
  + Implement secure payment processing mechanisms.
  + Encrypt sensitive information (e.g., payment details).
* Performance:
  + Ensure the application's responsiveness and performance, especially during flight searches and payment processing.
* Error Handling:
  + Provide clear and informative error messages for various scenarios, such as booking errors or invalid login credentials.
* Data Validation:
  + Implement data validation to ensure that user inputs are accurate and appropriate for each field.
* Accessibility:
  + Enhance the accessibility of the application to accommodate users with disabilities.
* Reliability:
  + Ensure the application's reliability by minimizing system crashes and errors.
* Localization:
  + Consider localization by providing support for multiple languages and currencies, if necessary.
* Integration:
  + Implement basic integration between the login and flight reservation modules.
  + Future enhancements may involve integration with payment gateways, airline databases, and email services.
* Documentation:
  + Maintain documentation that outlines how the system works, system requirements, and user manuals.
* User Testing:
  + Conduct user testing to ensure that the system meets user expectations and usability standards.

1. **ARCHITECTURE AND DESIGN**

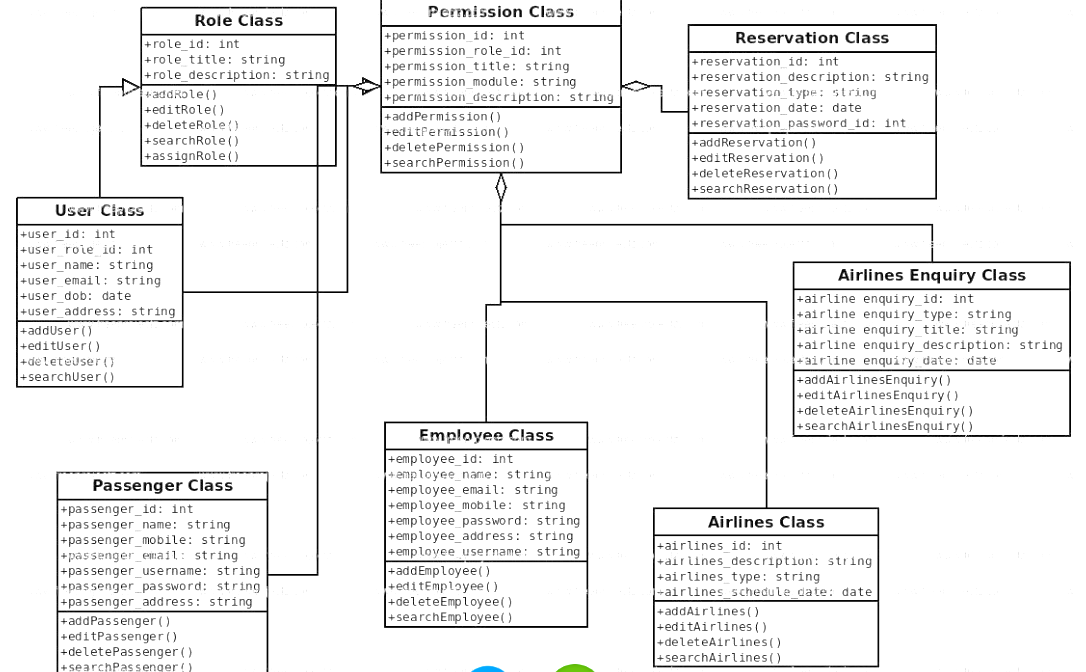


Fig4.1 Class Diagram for Airline reservation system

In this Fig 4.1, A Role Class defines the role of a user in the system, such as passenger, employee, or administrator. It has attributes like role\_id, role\_title, and role\_description, and methods like addRole, editRole, and deleteRole.

A User Class represents a user of the system, who has a role, a name, a description, and an id. It has methods like addUser, editUser, deleteUser, and searchUser.

A Passenger Class is a subclass of the User Class, which means it inherits all the attributes and methods of the User Class, and also has some additional attributes and methods specific to passengers, such as passport\_number, ticket\_number, and bookFlight.

An Employee Class is another subclass of the User Class, which also inherits all the attributes and methods of the User Class, and also has some additional attributes and methods specific to employees, such as employee\_id, salary, and assignFlight.

An Airlines Class represents an airline company, which has attributes like airlines\_id, airlines\_name,and airlines description, and methods like addAirlines, editAirlines, and deleteAirlines.

An Airlines Enquiry Class is a class that handles the queries and requests from the users regarding the flights, such as availability, schedule, price, and booking.

The diagram also shows the relationships between the classes, such as:

A User Class has a one-to-one relationship with a Role Class, which means each user has one and only one role, and each role belongs to one and only one user.

A Passenger Class has a one-to-many relationship with an Airlines Class, which means each passenger can book flights from many airlines, but each airline can have many passengers.

An Employee Class has a many-to-one relationship with an Airlines Class, which means each employee works for one and only one airline, but each airline can have many employees.

An Airlines Enquiry Class has a many-to-many relationship with both the Passenger Class and the Employee Class, which means each passenger or employee can make or handle many enquiries, and each enquiry can involve many passengers or employees.

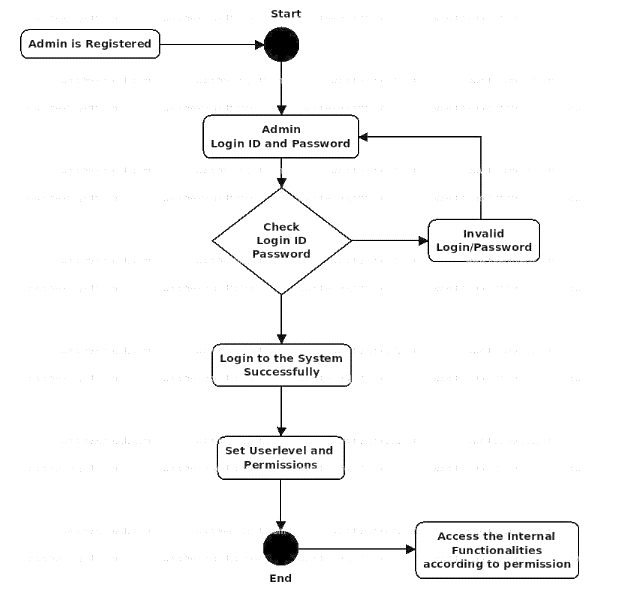
****

Fig4.2 Login Window Flowchart

In Fig 4.2, it shows the flows of Login Activity, where admin will be able to login using their username and password. After login user can manage all the operations on Reservation, Airlines, Passengar, Employee, Airline Enquiry All the pages such as Passengar, Employee, Airline Enquiry are secure and user can access these page after login. The diagram below helps demonstrate how the login page works in a Airlines Reservation System. The various objects in the Employee, Reservation, Airlines, Passengar, and Airline Enquiry page-interact over the course of the Activity, and user will not be able to access this page without verifying their identity

1. **IMPLEMENTATION OF AIRLINE RESERVATION SYSTEM**

import tkinter as tk

from tkinter import messagebox

from PIL import Image, ImageTk

class Flight:

    def \_\_init\_\_(self, flight\_number, origin, destination, departure\_time, arrival\_time, price, capacity):

        self.flight\_number = flight\_number

        self.origin = origin

        self.destination = destination

        self.departure\_time = departure\_time

        self.arrival\_time = arrival\_time

        self.price = price

        self.capacity = capacity

        self.reserved\_seats = 0

    def display\_flight\_info(self):

        return (

            f"Flight Number: {self.flight\_number}\n"

            f"From: {self.origin} To: {self.destination}\n"

            f"Departure Time: {self.departure\_time}\n"

            f"Arrival Time: {self.arrival\_time}\n"

            f"Price: ₹{self.price}\n"

            f"Available Seats: {self.capacity - self.reserved\_seats}\n"

        )

    def book\_seat(self):

        if self.reserved\_seats < self.capacity:

            self.reserved\_seats += 1

            return "booked"

        else:

            return "full"

    def calculate\_total\_fare(self, num\_seats):

        return self.price \* num\_seats

class ReservationSystem:

    def \_\_init\_\_(self):

        self.flights = []

    def add\_flight(self, flight):

        self.flights.append(flight)

    def get\_available\_flights(self):

        available\_flights = []

        for flight in self.flights:

            if flight.reserved\_seats < flight.capacity:

                available\_flights.append(flight.display\_flight\_info())

        return available\_flights

    def book\_flight(self, flight\_number, num\_seats):

        for flight in self.flights:

            if flight.flight\_number == flight\_number:

                result = flight.book\_seat()

                if result == "booked":

                    total\_fare = flight.calculate\_total\_fare(num\_seats)

                    return total\_fare

                else:

                    return "full"

        return "not\_found"

class FlightReservationGUI:

    def \_\_init\_\_(self, root, reservation\_system):

        self.root = root

        self.root.title("Flight Reservation System")

        self.reservation\_system = reservation\_system

        # Load the background image for the reservation window

        background\_image = Image.open("C:\\Users\\SRIJAN\\OneDrive\\Desktop\\reservationbg.jpg")

        background\_photo = ImageTk.PhotoImage(background\_image)

        background\_label = tk.Label(root, image=background\_photo)

        background\_label.image = background\_photo

        background\_label.place(relwidth=1, relheight=1)

        self.label = tk.Label(root, text="Flight Reservation System", font=("Helvetica", 20), bg="sky blue")

        self.label.pack(pady=10)

        self.available\_flights\_label = tk.Label(root, text="Available Flights", font=("Helvetica", 16), bg="light green")

        self.available\_flights\_label.pack()

        self.available\_flights\_text = tk.Text(root, height=10, width=40, font=("Helvetica", 12))

        self.available\_flights\_text.pack()

        self.book\_label = tk.Label(root, text="Enter Flight Number to Book:", font=("Helvetica", 16), bg="light yellow")

        self.book\_label.pack()

        self.flight\_number\_entry = tk.Entry(root, font=("Helvetica", 14))

        self.flight\_number\_entry.pack()

        self.seats\_label = tk.Label(root, text="Number of Seats:", font=("Helvetica", 16), bg="light yellow")

        self.seats\_label.pack()

        self.seats\_entry = tk.Entry(root, font=("Helvetica", 14))

        self.seats\_entry.pack()

        self.total\_fare\_label = tk.Label(root, text="Total Fare (₹):", font=("Helvetica", 16), bg="light yellow")

        self.total\_fare\_label.pack()

        self.total\_fare\_var = tk.StringVar()

        self.total\_fare\_label\_value = tk.Label(root, textvariable=self.total\_fare\_var, font=("Helvetica", 14))

        self.total\_fare\_label\_value.pack()

        self.book\_button = tk.Button(root, text="Book Flight", command=self.book\_flight, font=("Helvetica", 14), bg="light blue")

        self.book\_button.pack(pady=10)

        self.payment\_label = tk.Label(root, text="Payment Options", font=("Helvetica", 16), bg="light yellow")

        self.payment\_label.pack()

        self.payment\_method = tk.StringVar()

        self.payment\_method.set("Credit Card")

        payment\_options = ["Credit Card", "Debit Card", "Net Banking"]

        for option in payment\_options:

            tk.Radiobutton(root, text=option, variable=self.payment\_method, value=option, font=("Helvetica", 14), bg="light yellow").pack()

        self.pay\_button = tk.Button(root, text="Pay Now", command=self.process\_payment, font=("Helvetica", 14), bg="light blue")

        self.pay\_button.pack()

        self.display\_available\_flights()

    def display\_available\_flights(self):

        available\_flights = self.reservation\_system.get\_available\_flights()

        self.available\_flights\_text.delete(1.0, tk.END)

        for flight\_info in available\_flights:

            self.available\_flights\_text.insert(tk.END, flight\_info)

            self.available\_flights\_text.insert(tk.END, "\n\n")

    def book\_flight(self):

        flight\_number = self.flight\_number\_entry.get()

        num\_seats = int(self.seats\_entry.get())

        result = self.reservation\_system.book\_flight(flight\_number, num\_seats)

        if isinstance(result, int):

            self.total\_fare\_var.set(result)

            messagebox.showinfo("Booking Successful", f"Seat(s) booked successfully.\nTotal Fare: ₹{result}")

            self.display\_available\_flights()

        elif result == "not\_found":

            messagebox.showerror("Booking Error", "Flight not found.")

        else:

            messagebox.showerror("Booking Error", "Flight is fully booked")

    def process\_payment(self):

        total\_fare = self.total\_fare\_var.get()

        payment\_method = self.payment\_method.get()

        if not total\_fare:

            messagebox.showerror("Payment Error", "Total fare is not available.")

        else:

            messagebox.showinfo("Payment Successful", f"Payment of ₹{total\_fare} via {payment\_method} successful.")

        self.exit()

    def exit(self):

        self.root.destroy()

class LoginGUI:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Login")

        # Load the background image for the login window

        background\_image = Image.open("C:\\Users\\SRIJAN\\OneDrive\\Desktop\\loginbg2.jpg")

        background\_photo = ImageTk.PhotoImage(background\_image)

        background\_label = tk.Label(root, image=background\_photo)

        background\_label.image = background\_photo

        background\_label.place(relwidth=1, relheight=1)

        self.label = tk.Label(root, text="Login", font=("Helvetica", 20))

        self.label.pack(pady=10)

        self.username\_label = tk.Label(root, text="Username", font=("Helvetica", 16))

        self.username\_label.pack()

        self.username\_entry = tk.Entry(root, font=("Helvetica", 14))

        self.username\_entry.pack()

        self.password\_label = tk.Label(root, text="Password", font=("Helvetica", 16))

        self.password\_label.pack()

        self.password\_entry = tk.Entry(root, show="\*", font=("Helvetica", 14))

        self.password\_entry.pack()

        self.login\_button = tk.Button(root, text="Login", command=self.login, font=("Helvetica", 14), bg="light green")

        self.login\_button.pack(pady=10)

    def login(self):

        username = self.username\_entry.get()

        password = self.password\_entry.get()

        if username == "Srijan" and password == "srijan21":

            self.root.destroy()

            self.open\_flight\_reservation()

        else:

            messagebox.showerror("Login Error", "Invalid username or password")

    def open\_flight\_reservation(self):

        root = tk.Tk()

        reservation\_system = ReservationSystem()

        app = FlightReservationGUI(root, reservation\_system)

        # Add the provided flights to the reservation system

        flight1 = Flight("AI101", "Mumbai", "Delhi", "08:00 AM", "11:00 AM", 5000, 150)

        flight2 = Flight("IND202", "Bangalore", "Kolkata", "09:30 AM", "01:30 PM", 4500, 200)

        flight3 = Flight("Jet303", "Chennai", "Hyderabad", "10:45 AM", "01:15 PM", 3800, 100)

        flight4 = Flight("Spice404", "Delhi", "Mumbai", "08:30 AM", "11:30 AM", 5200, 120)

        flight5 = Flight("Vistara505", "Kolkata", "Chennai", "10:00 AM", "01:00 PM", 4700, 180)

        flight6 = Flight("GoAir606", "Hyderabad", "Bangalore", "11:15 AM", "02:00 PM", 3900, 160)

        reservation\_system.add\_flight(flight1)

        reservation\_system.add\_flight(flight2)

        reservation\_system.add\_flight(flight3)

        reservation\_system.add\_flight(flight4)

        reservation\_system.add\_flight(flight5)

        reservation\_system.add\_flight(flight6)

        app.display\_available\_flights()

        root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    app = LoginGUI(root)

    root.mainloop()

This Python script showcases the implementation of a Flight Reservation System with a user-friendly graphical interface using the Tkinter library. The code consists of several classes, with the primary ones being Flight, representing flight details, ReservationSystem, managing flights and reservations, and FlightReservationGUI, responsible for creating the graphical user interface. There's also a LoginGUI class for basic user authentication.

Upon execution, the program launches a login window where users input their credentials. Upon successful login (with hardcoded credentials for simplicity), the application opens a Flight Reservation window. This window displays available flights, allows users to book seats, and provides payment options. The GUI features dynamic updates, input fields for booking, and a visually appealing background.

Six sample flights with unique details are added to the system, illustrating the core functionalities of flight booking, reservation tracking, and payment processing. The code serves as an educational example, demonstrating concepts such as GUI development, object-oriented programming, and basic user authentication within the context of a flight reservation system.

**6. EXPERIMENT RESULTS & ANALYSIS**

In Fig 6.1,It has a Login Page for Airline Reservation system in which it has fields Username, Password where we enter user name and password and press login then it will redirect us to next page

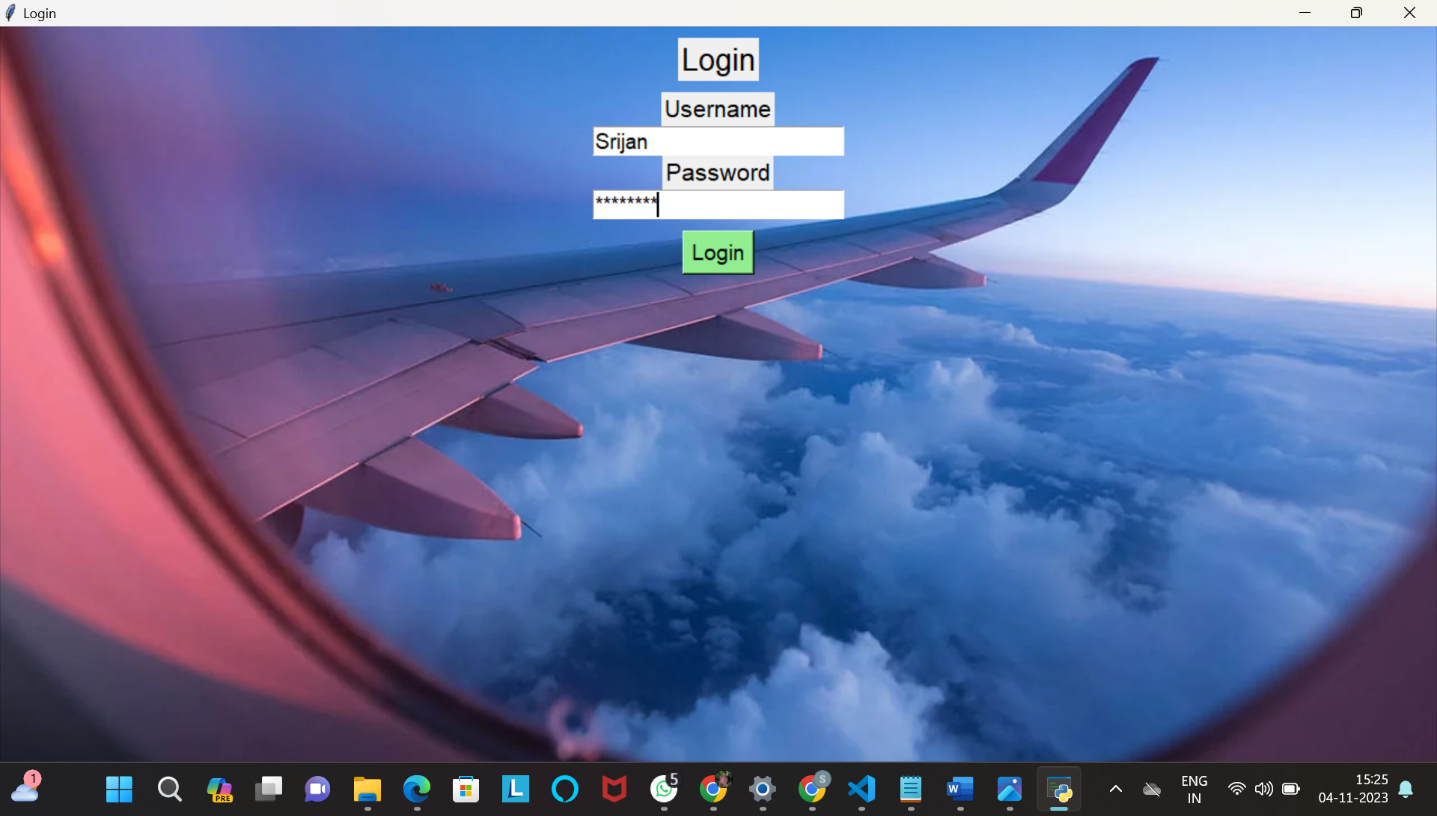


Fig6.1 Login Page for Reservation System

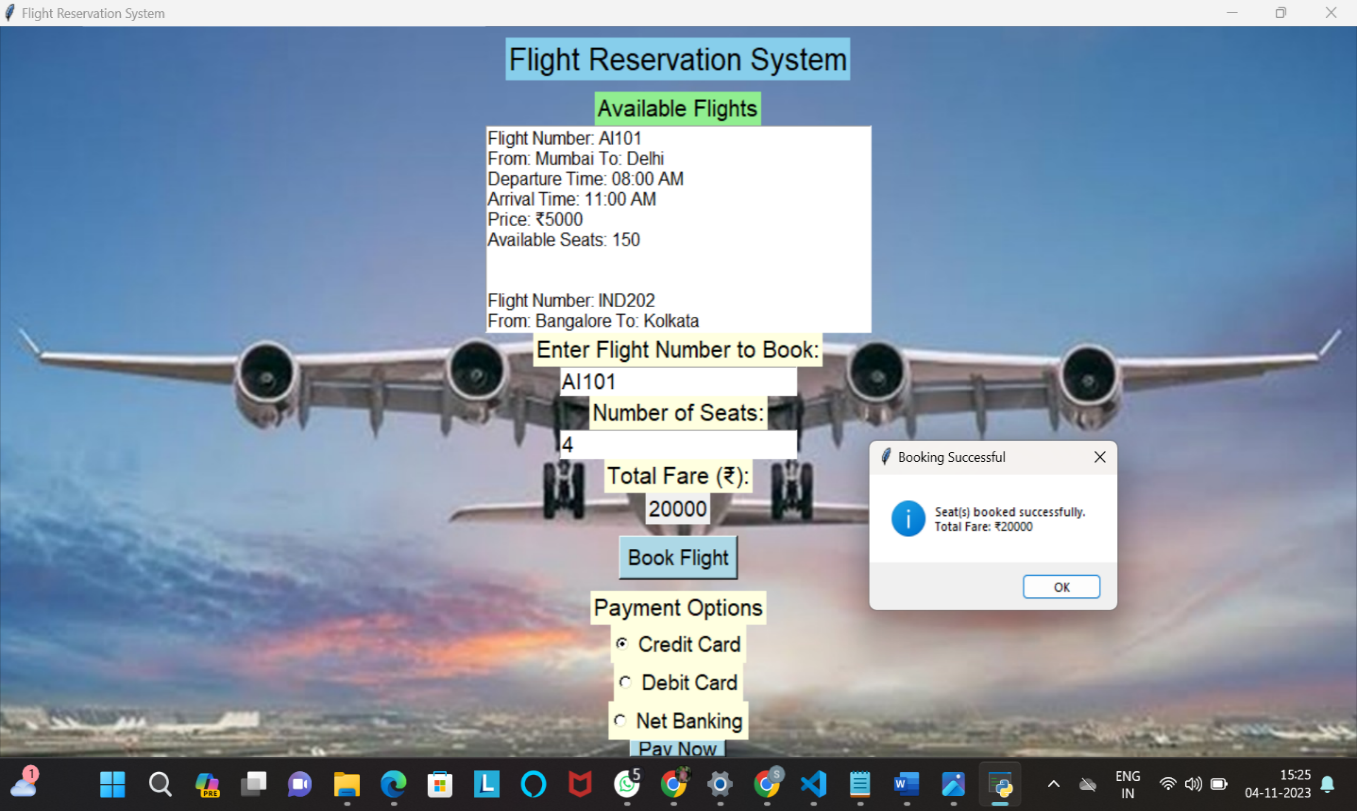


Fig6.2 Reservation Window Showing Total Fare

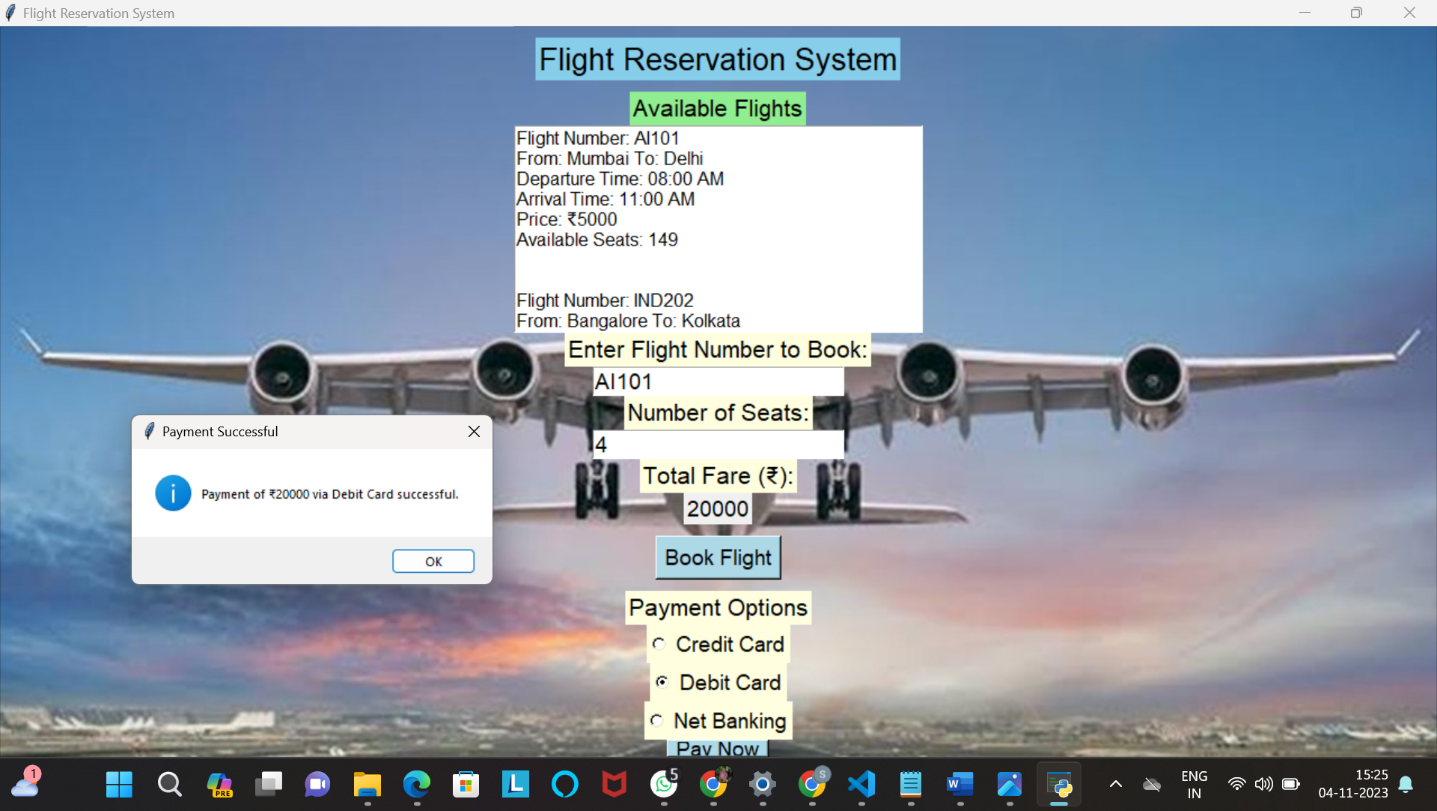


Fig6.3 Flight Reservation Window Showing Payment

In this Fig 6.2 and 6.3, Reservation window shows the flights available for travel we can choose the flight by entering flight number in the column and then we will tell about how many seats we need then when we click on book flight it will show the total fare then we can select payment options i.e. Credit Card , Debit Card, Net Banking then we click on Pay now and a window will appear which will payment Successful

**7. CONCLUSION**

In summary, the provided code offers a simplified representation of a Flight Reservation System using Python and Tkinter. It includes user authentication, flight information display, booking functionality, payment options, reservation confirmation, and a basic graphical user interface. Notable points include the absence of comprehensive security measures, limited error handling, and the need for further development to meet real-world standards.

Key features encompass user authentication through a basic login system, detailed flight information presentation, and the ability to book flights with confirmation and total fare display. The graphical user interface is designed for desktop applications using Tkinter.

However, it's crucial to acknowledge the code's limitations. It lacks robust security measures, comprehensive error handling, and the complexity inherent in real-world flight reservation systems. A complete system demands integration with external services, adherence to industry regulations, and the involvement of diverse professionals such as developers, database administrators, and security experts.

While the provided code establishes a foundation for understanding fundamental components of a flight reservation system, further enhancements and refinements are necessary to create a fully functional and industry-compliant application.

**REFERENCES**

1. Ajulo S.B. (2002). ―Aviation Handbook‖ Volume 1, Vantage Publishers Limited, Ibadan, Nigeria.
2. American Airlines, n.d. Viewed on August 4, 2014 from <http://www.aa.com>.
3. Arik Air, n.d. Viewed on August 4, 2014 from http://www.arikair.com. Airline Reservation Systems. In Wikipedia (2014, August). Retrieved from <http://en.wikipedia.org/wiki/Airline_reservations_system>.
4. D.G. Copeland, R.O. Mason, & J.L. McKenney, (1995). SABRE: The Development of Information-Based Competence and Execution of Information-Based Competition, IEEE Annuals of the History of Computing (vol. 17, no. 3, pp. 30-57).
5. Doganis, R. (2001). The airline business in the twenty-first century. NY: Rutledge.
6. Emirates Airlines, n.d. Viewed on August 4, 2014 from <http://www.emirates.com.ng>.
7. Evelyn R. K. (2007). System Integration, Testing of the Systems, Applications and Products.
8. French, C.S. (1998). Computer science. DP publications limited (3 rd. ed. pp. 606, 617, and 630). Aldine place, 142/144 Uxbridge Road London.
9. Crosby, T. (5th August, 2015). How Airline E-Tickets Work. from An airline reservation system is a part of so-called Passenger Services Systems (PSS), which are applications supporting the direct contract with the passenger(Crosby, 2015) .IEEE.
10. Shao, Q., Chen, Y., Tao, S., Yan, X. andAnerousis, N. (2008). a ticket routing recommendation engine for enterprise problem resolution. Proceedings of the VLDB Endowment, Vol 1(2).IEEE.
11. Smith, M. J. (2002). The Airline Encyclopedia. Scarecrow Press, New York.In 1946, American Airlines installed the first automated booking system, the experimental electromechanical Reservation. A newer machine with temporary storage based on a magnetic drum, the Magnetron Reservoirs, soon followed(Smith, 2002).IEEE.
12. Erradi, A., Anand, S., Kulkani, N. (2006). An Architectural Framework for Service Definition and Realization. On the proceedings of IEEE international conference on services computing, IEEE