**22AIE303: DATABASE MANAGEMENT SYSTEM**

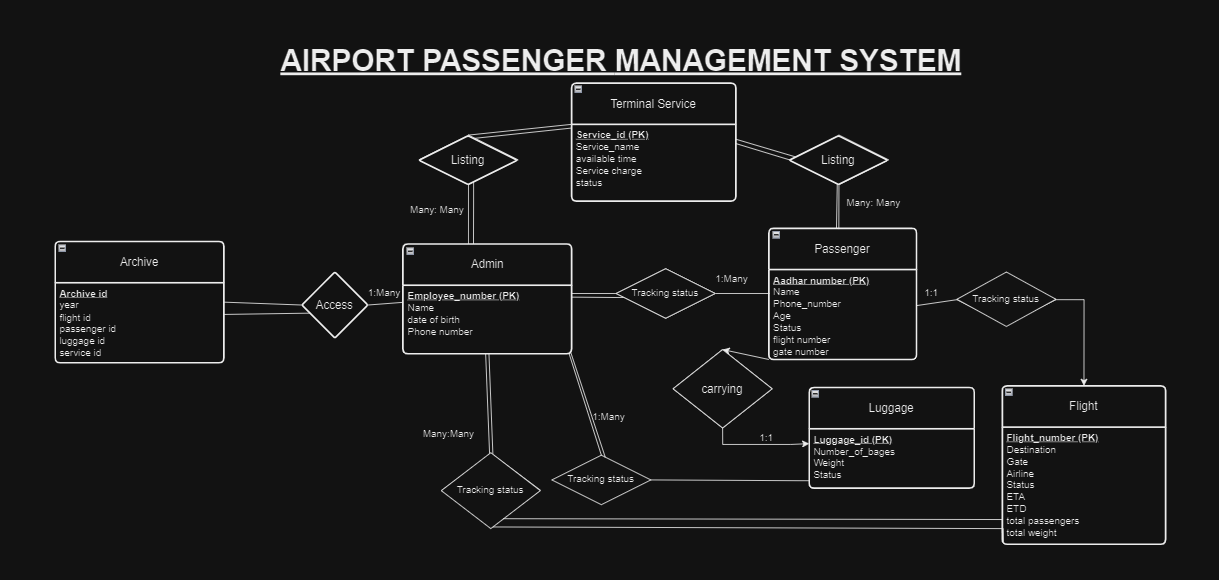
Project Review 1

**Title:** AIRPORT PASSENGERS MANAGEMENT SYSTEM

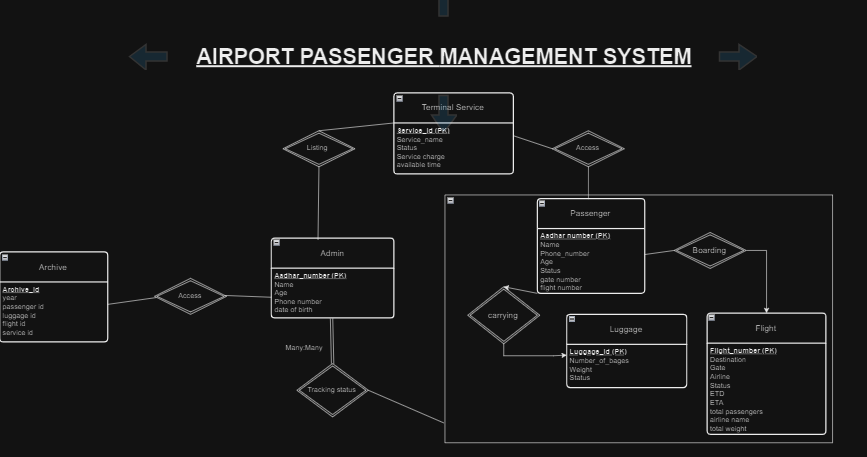
Group Members:

|  |  |
| --- | --- |
| Akash Ranjan | AM.EN.U4AIE21070 |
| Anupa Sajikumar | AM.EN.U4AIE21071 |

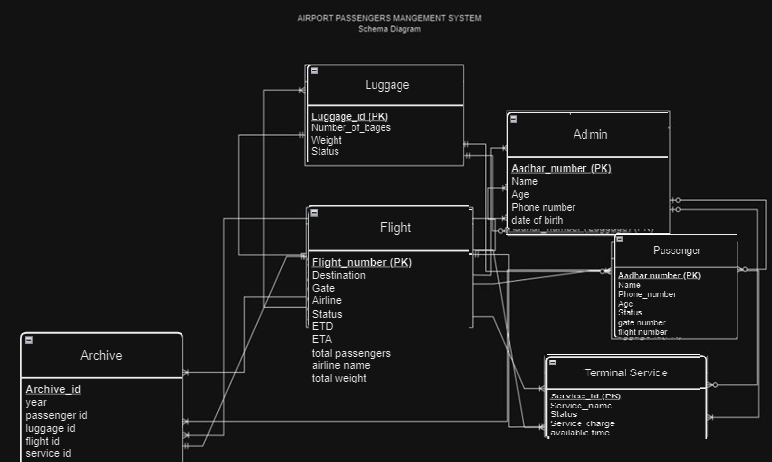
**ER Diagram:**

****

**Extended ER Diagram:**

****

**Schema Diagram:**

****

**Relational Schema:**

**Relational Schema**

**1. Terminal\_Service**

* **Service\_id**: INTEGER PRIMARY KEY AUTOINCREMENT
* **Service\_name**: TEXT
* **Service\_charges**: REAL
* **Available\_time**: TEXT
* **Status**: TEXT

**2. Admin**

* **Employee\_number**: TEXT PRIMARY KEY
* **Name**: TEXT
* **Date\_of\_birth**: TEXT
* **Phone\_number**: TEXT

**3. Passenger**

* **Aadhar\_number**: TEXT PRIMARY KEY
* **Name**: TEXT
* **Phone\_number**: TEXT
* **Age**: INTEGER
* **Gate\_number**: TEXT

**4. Luggage**

* **Luggage\_id**: INTEGER PRIMARY KEY AUTOINCREMENT
* **Aadhar\_number**: TEXT
  + FOREIGN KEY (Aadhar\_number) REFERENCES Passenger(Aadhar\_number)
* **Number\_of\_bags**: INTEGER
* **Total\_weight**: REAL
* **Status**: TEXT

**5. Flight**

* **Flight\_number**: TEXT PRIMARY KEY
* **Airline\_name**: TEXT
* **Aeroplane\_model**: TEXT
* **Total\_passengers**: INTEGER
* **Total\_weight**: REAL
* **Gate\_number**: TEXT
* **ETA**: TEXT
* **ETD**: TEXT
* **Date**: TEXT
* **Status**: TEXT

**6. Passenger\_Flight**

* **Aadhar\_number**: TEXT
  + FOREIGN KEY (Aadhar\_number) REFERENCES Passenger(Aadhar\_number)
* **Flight\_number**: TEXT
  + FOREIGN KEY (Flight\_number) REFERENCES Flight(Flight\_number)
* PRIMARY KEY (Aadhar\_number, Flight\_number)

**7. Archive**

* **Archive\_ID**: INTEGER PRIMARY KEY AUTOINCREMENT
* **Passenger\_ID**: TEXT
  + FOREIGN KEY (Passenger\_ID) REFERENCES Passenger(Aadhar\_number)
* **Luggage\_ID**: INTEGER
  + FOREIGN KEY (Luggage\_ID) REFERENCES Luggage(Luggage\_id)
* **Service\_ID**: INTEGER
  + FOREIGN KEY (Service\_ID) REFERENCES Terminal\_Service(Service\_id)
* **Flight\_ID**: TEXT
  + FOREIGN KEY (Flight\_ID) REFERENCES Flight(Flight\_number)
* **Year**: INTEGER

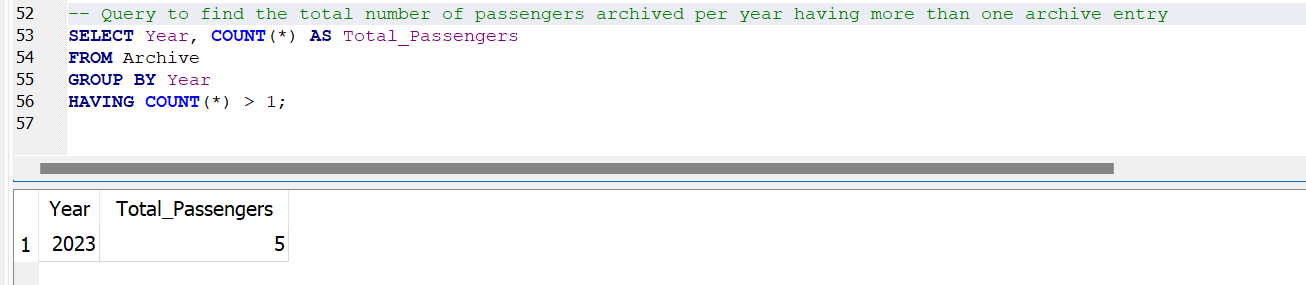
**Explanation**

1. **Terminal\_Service**: This table stores information about the services available at the terminal, such as cleaning, catering, security, etc.
2. **Admin**: This table contains information about the admin employees, including their employee number, name, date of birth, phone number, and the service they are responsible for.
3. **Passenger**: This table holds passenger information, including their Aadhar number, name, phone number, age, and gate number.
4. **Luggage**: This table stores luggage information linked to passengers, including luggage ID, Aadhar number, number of bags, total weight, and status.
5. **Flight**: This table contains flight information, including flight number, airline name, aeroplane model, total passengers, total weight, gate number, ETA, ETD, date, and status.
6. **Passenger\_Flight**: This table links passengers to flights they are taking. It is a many-to-many relationship between Passenger and Flight.
7. **Archive**: This table stores archived information linking passengers, luggage, services, and flights along with the year. It references primary keys from Passenger, Luggage, Terminal\_Service, and Flight.

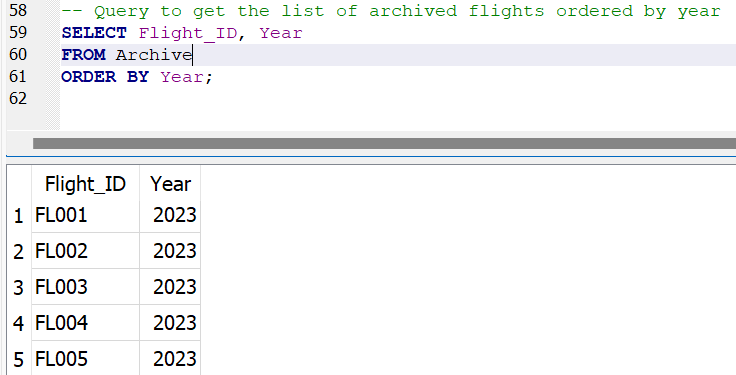
**Insert appropriate data into the tables and generate 10 queries.**

**The queries must be based on the following:**

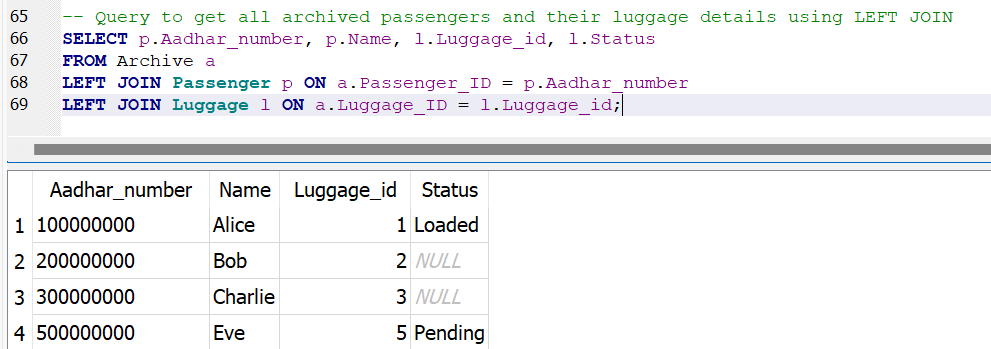
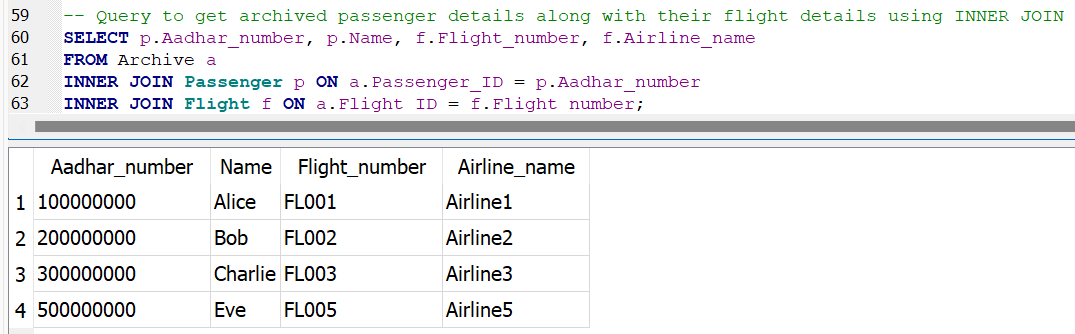
i. Aggregate functions, Group by…having



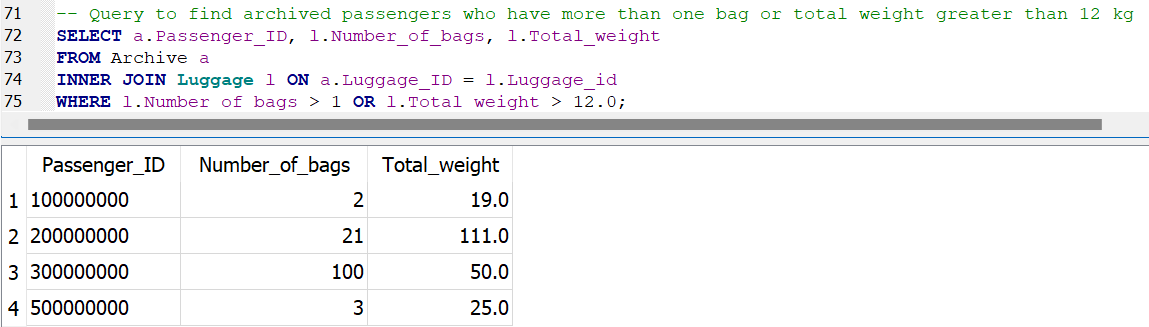
ii. Order by



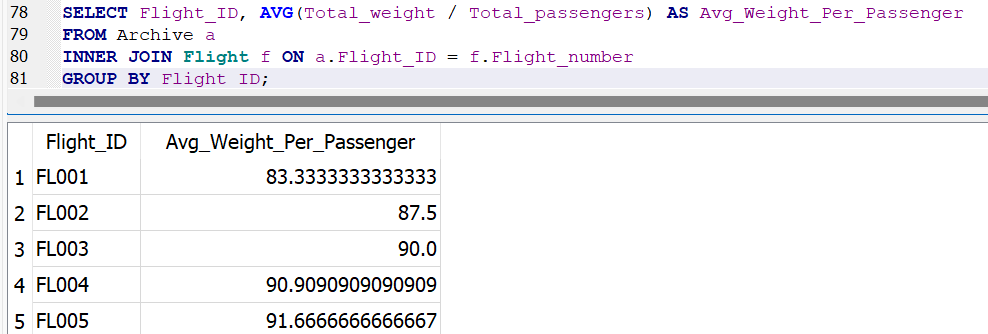
iii. Join, Outer Join



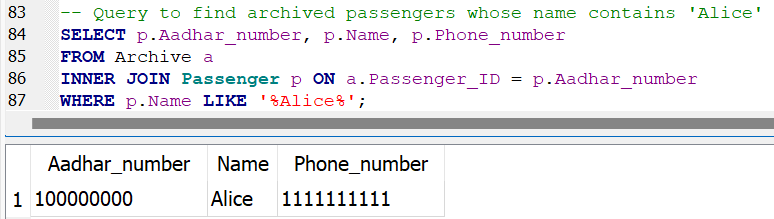
iv. Query having Boolean operators



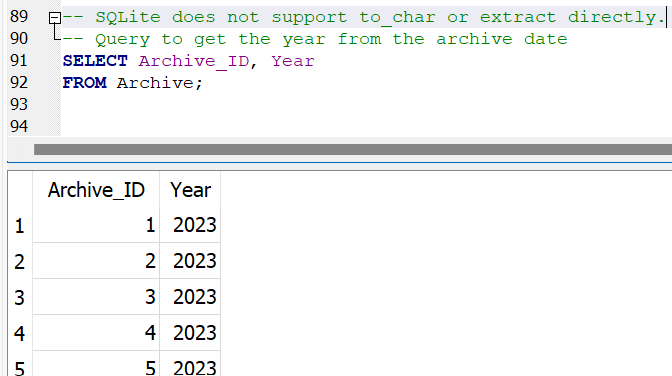
v. Query having arithmetic operators



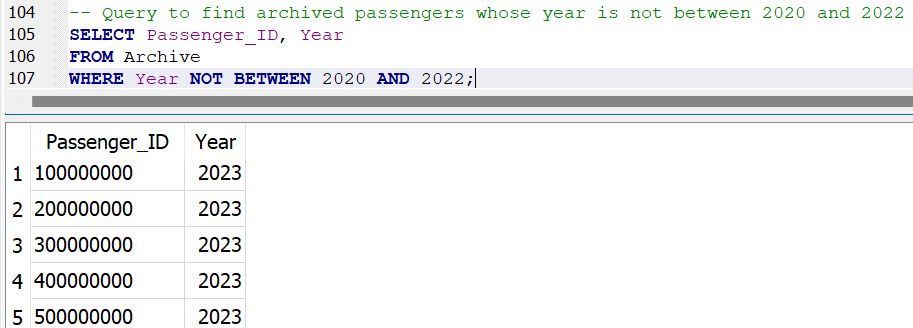
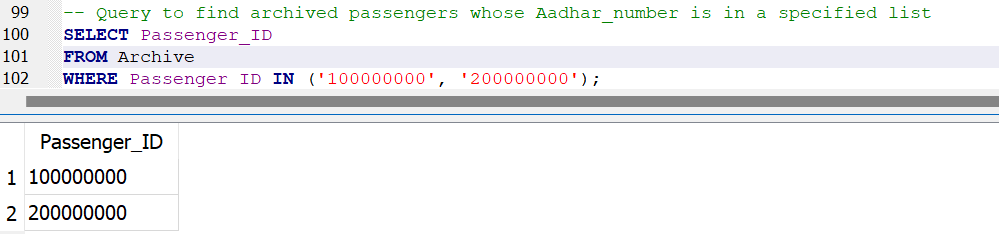
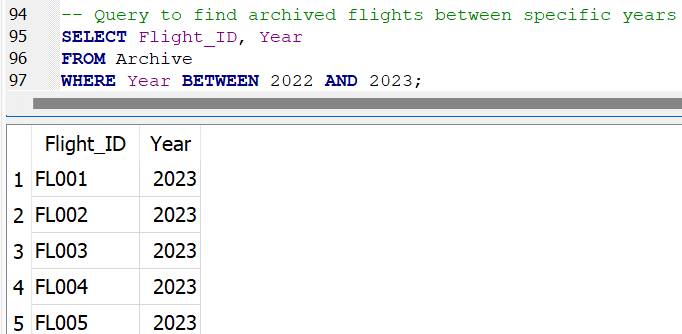
vi. A search query using string operators



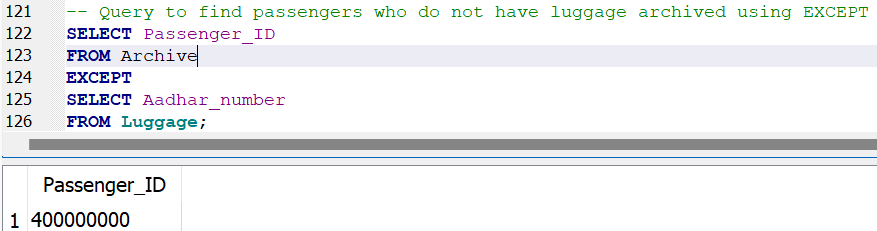
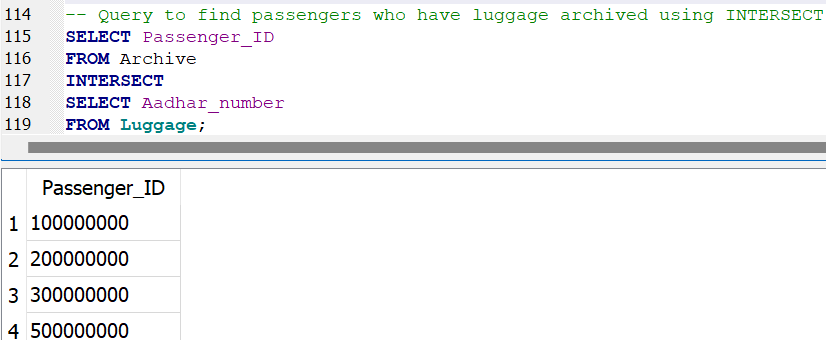
vii. Usage of to\_char, extract



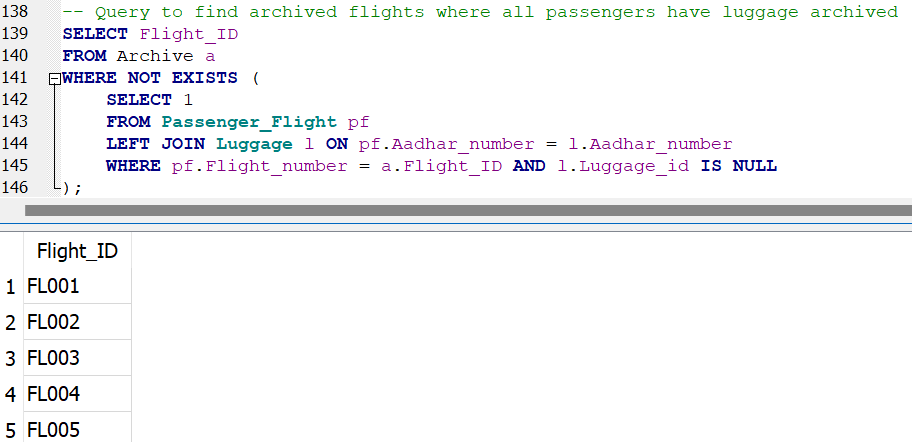
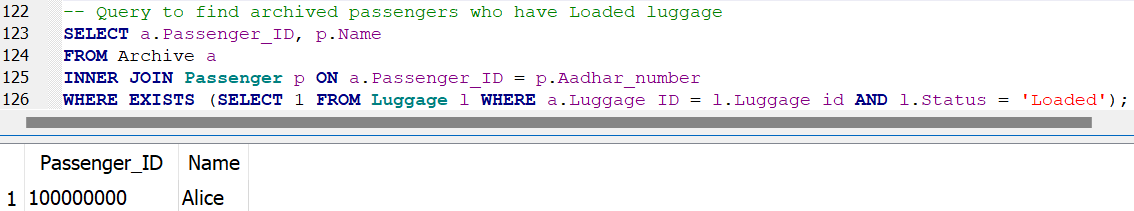
viii. Between, IN, Not between, Not IN



ix. Set operations



x. Subquery using EXISTS / NOT EXISTS, ANY, AL



**Construct a universal table related to your assigned project. The table should consist of at least 12 attributes.**

**Normalization (1NF - 3NF):**

**Universal table and 1NF**

Record\_ID | Passenger\_Aadhar | Passenger\_Name | Passenger\_Phone | Passenger\_Age | Gate\_Number | Flight\_Number | Airline\_Name | Flight\_Status | Luggage\_ID | Number\_of\_Bags | Luggage\_Weight | Luggage\_Status | Service\_ID | Service\_Name | Admin\_ID | Admin\_Name | Year

**2NF**

**Passenger** Table

Passenger\_Aadhar | Passenger\_Name | Passenger\_Phone | Passenger\_Age I Gate number

**Flight** Table

Flight\_Number | Airline\_Name | Flight\_Status I Gate number

**Luggage** Table

Luggage\_ID | Number\_of\_Bags | Luggage\_Weight | Luggage\_Status I Gate number

**Service** Table

Service\_ID | Service\_Name

**Admin** Table

Admin\_ID | Admin\_Name

**Archive** Table

Record\_ID | Year

**3NF**

**Passenger** Table

Passenger\_Aadhar | Passenger\_Name | Passenger\_Phone | Passenger\_Age I Gate number

**Flight** Table

Flight\_Number | Airline\_Name | Flight\_Status I Gate number

**Luggage** Table

Luggage\_ID | Number\_of\_Bags | Luggage\_Weight | Luggage\_Status I Gate number

**Service** Table

Service\_ID | Service\_Name

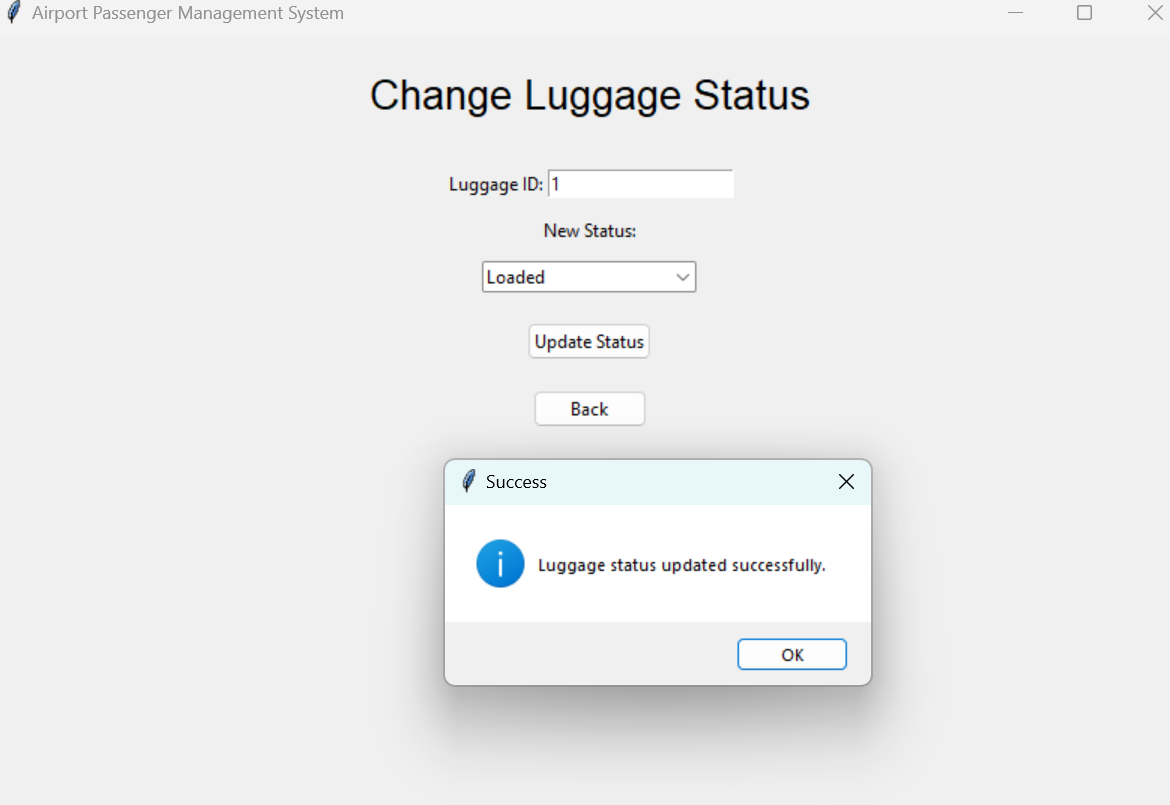
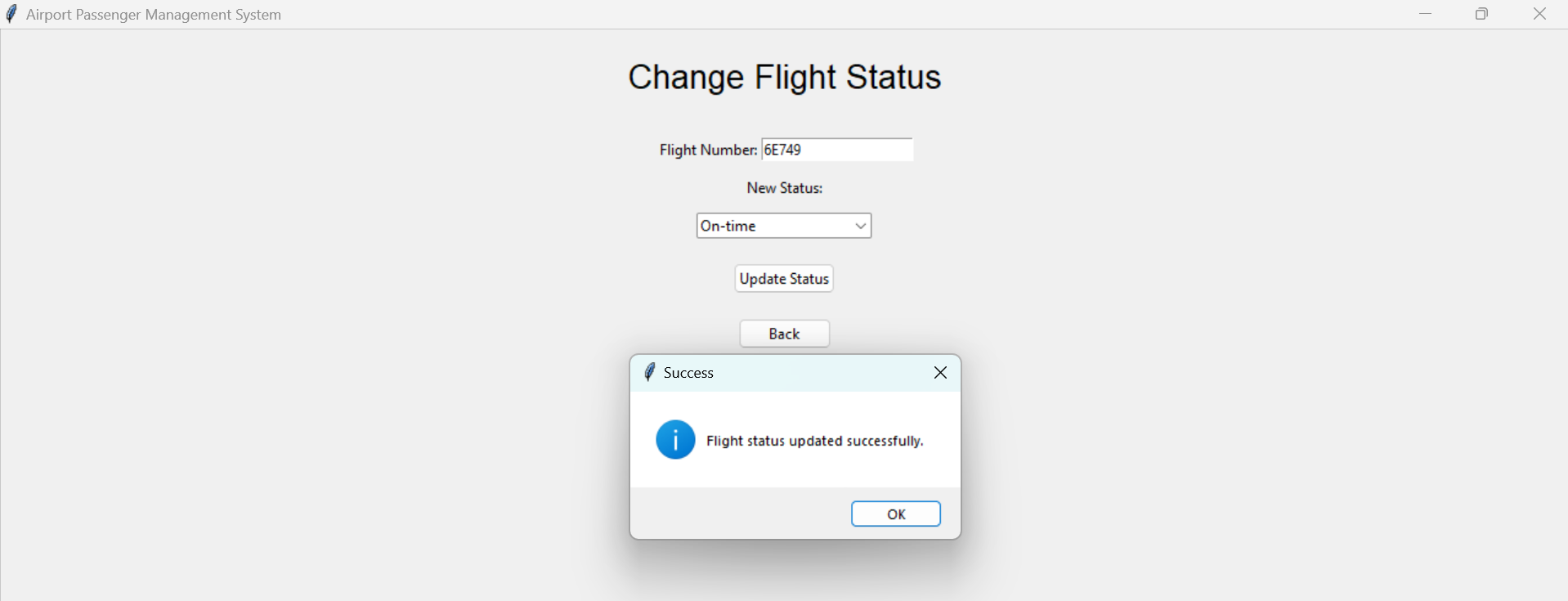
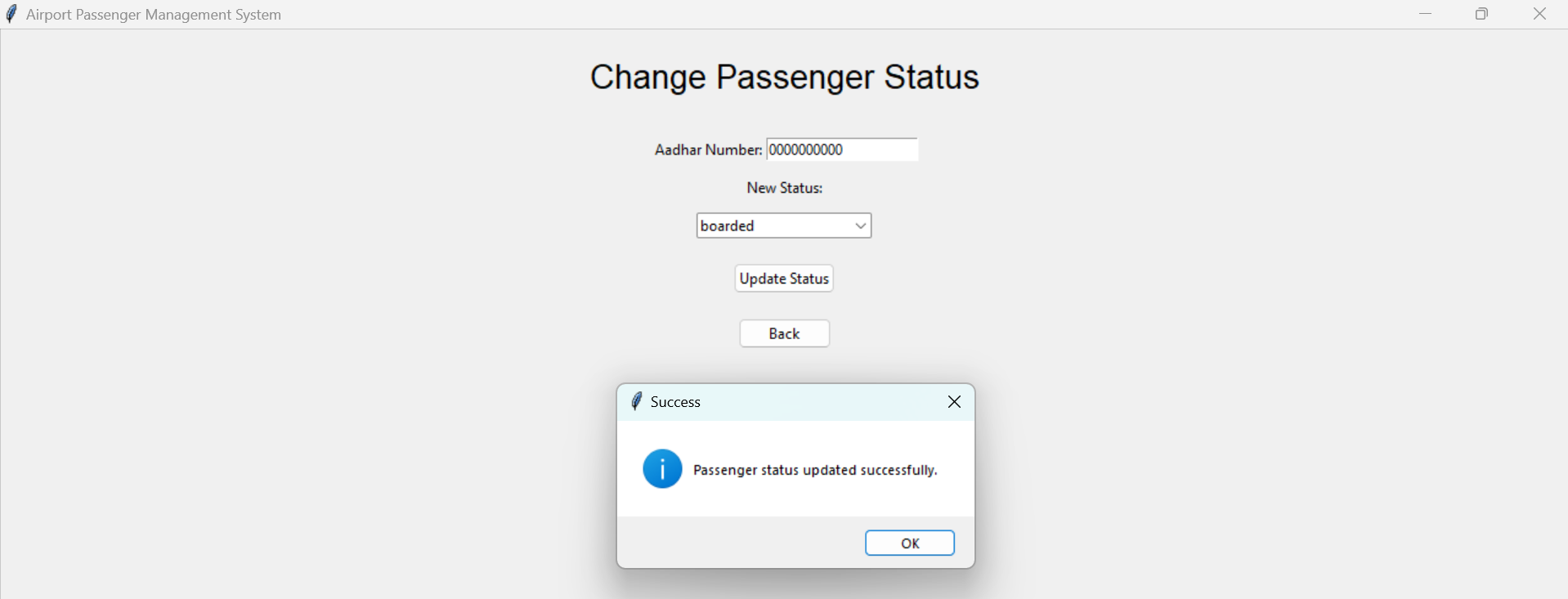
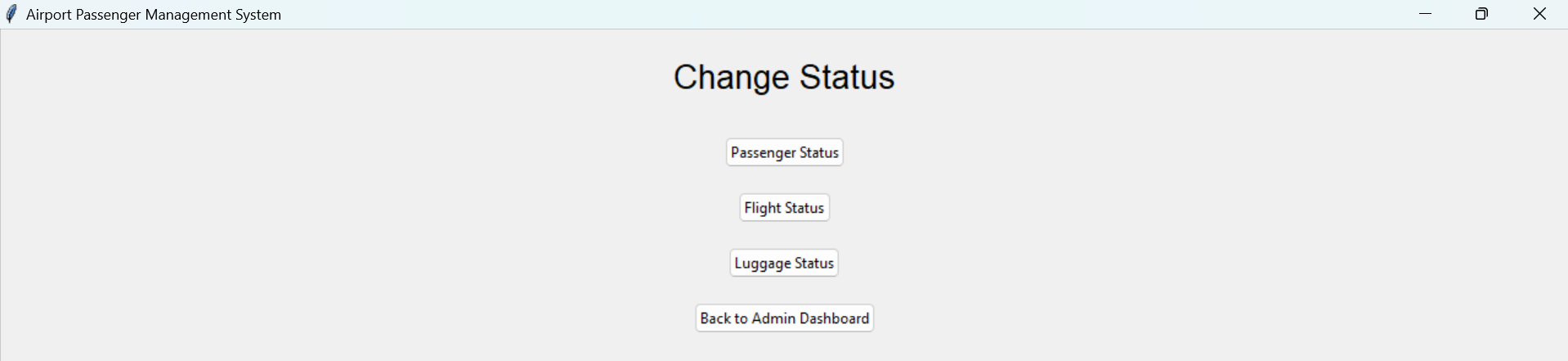
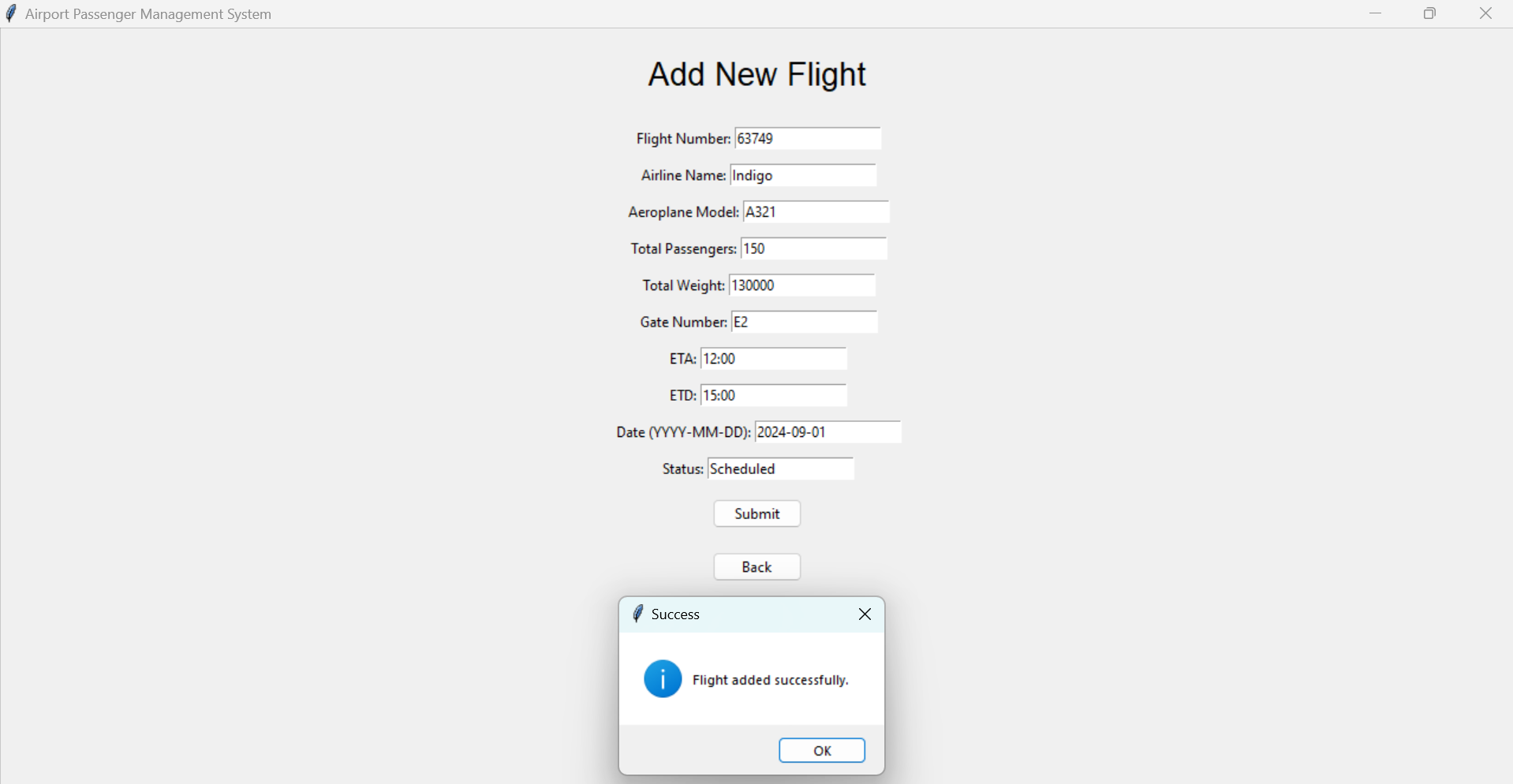
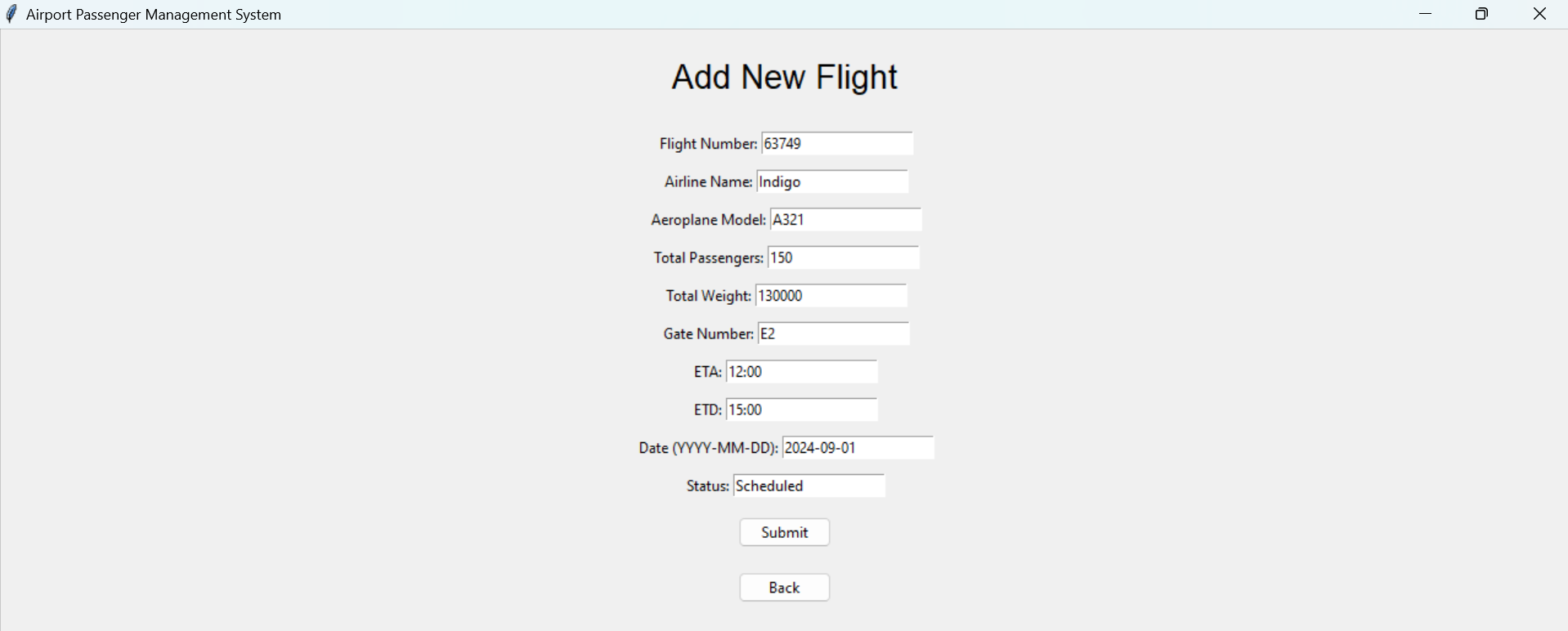
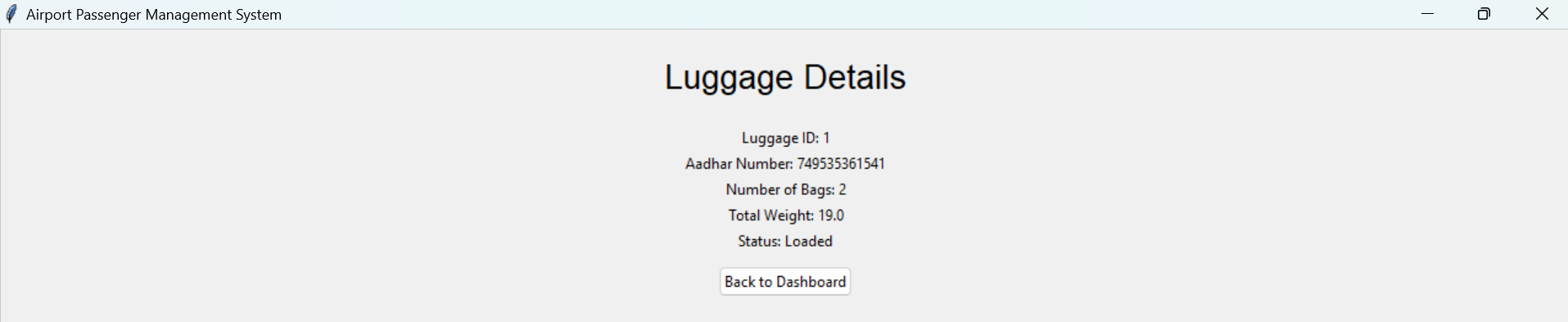
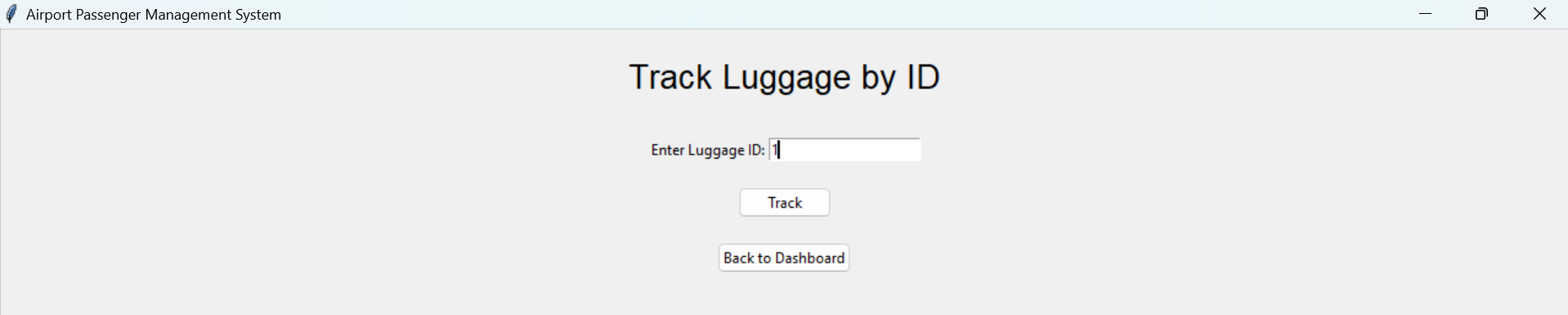
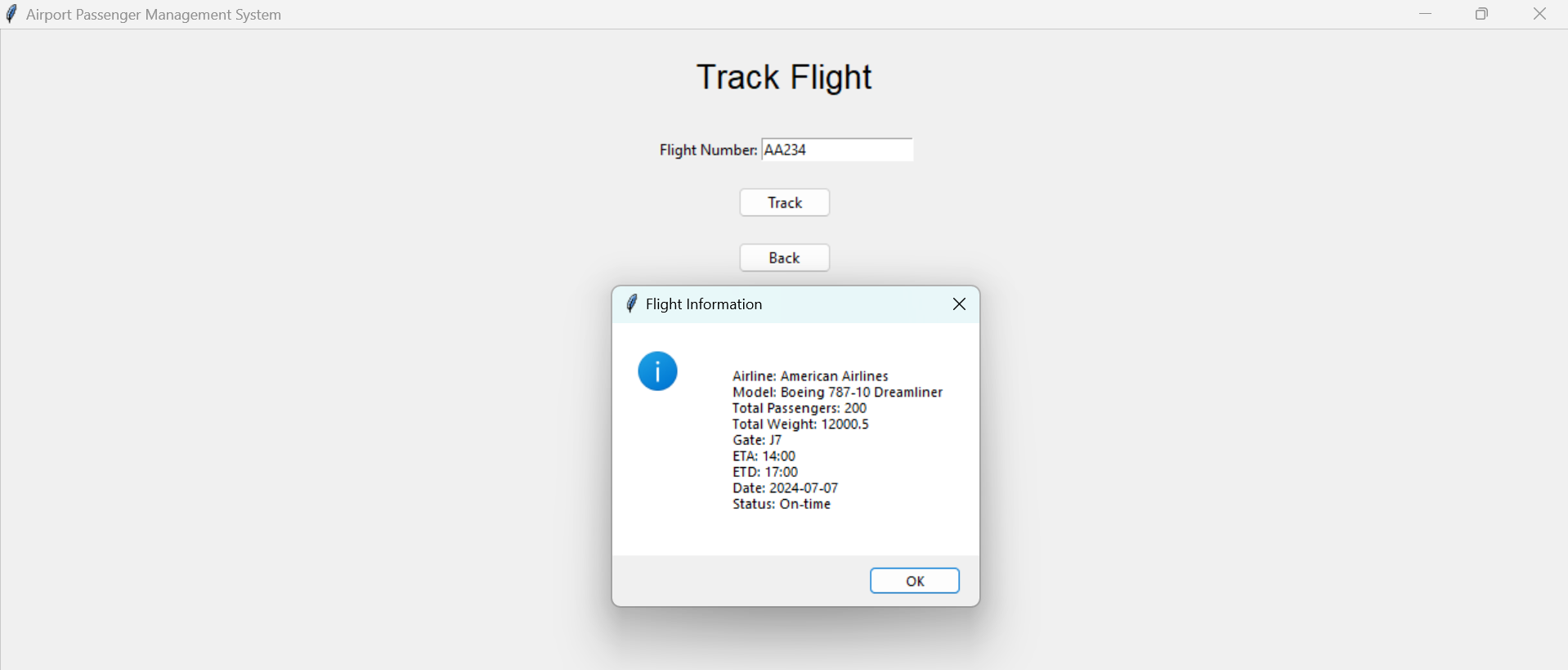
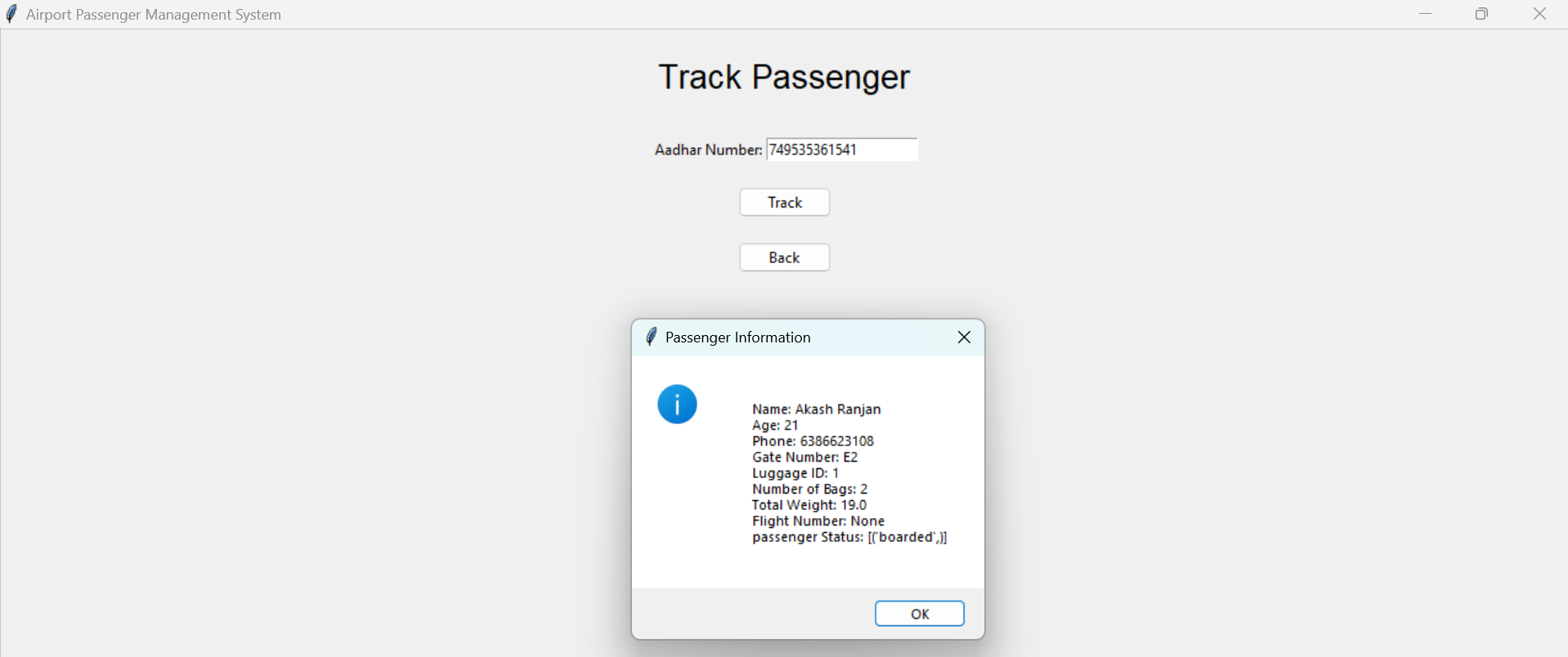
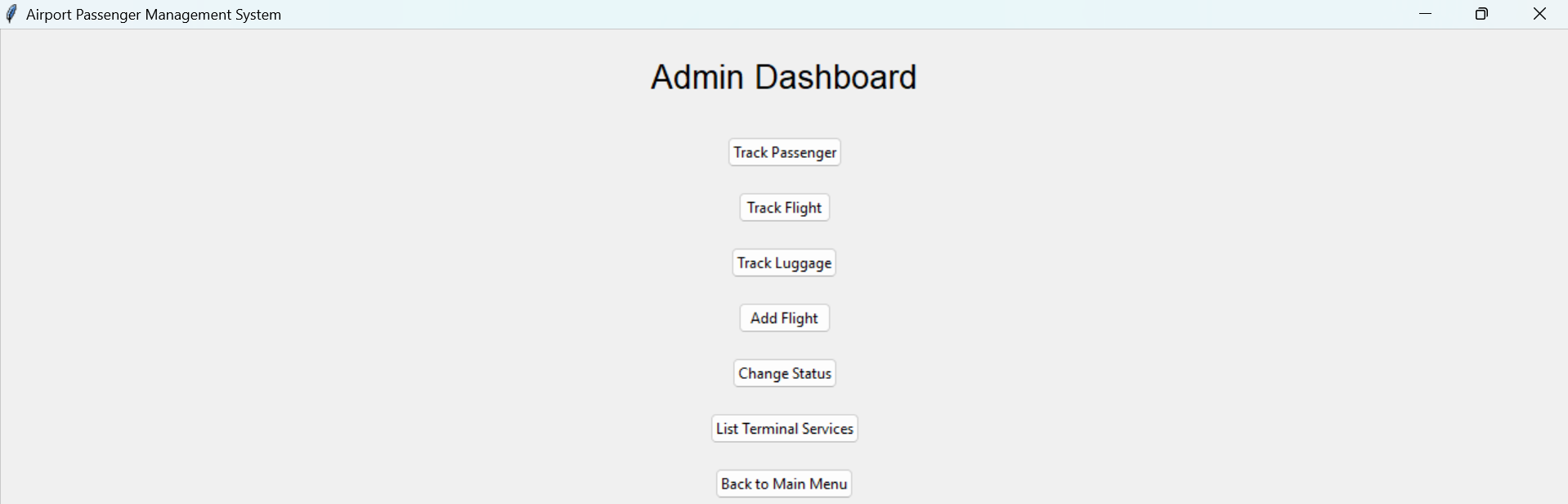
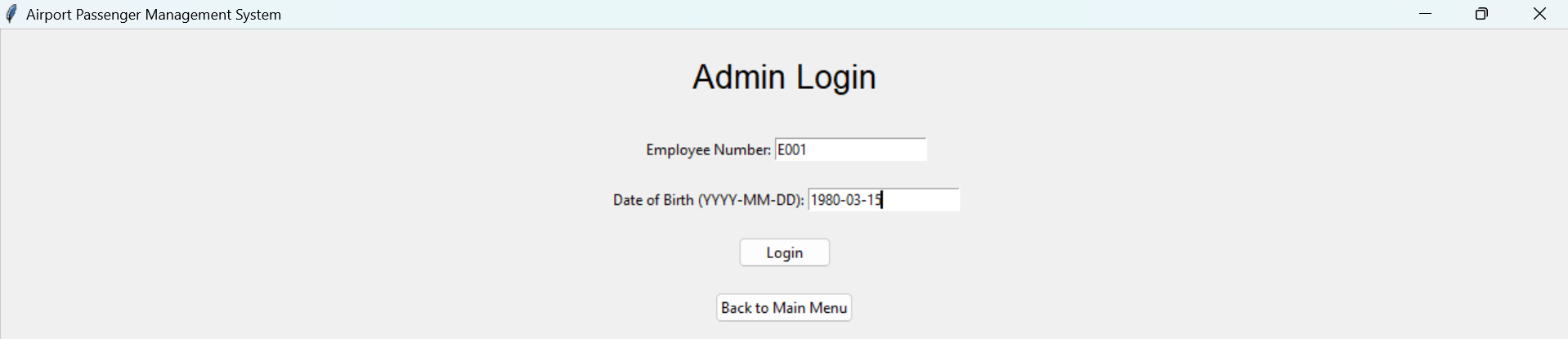
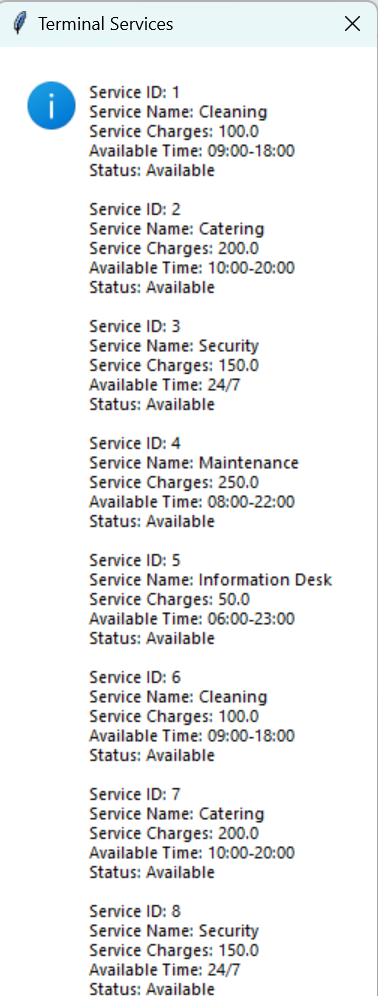
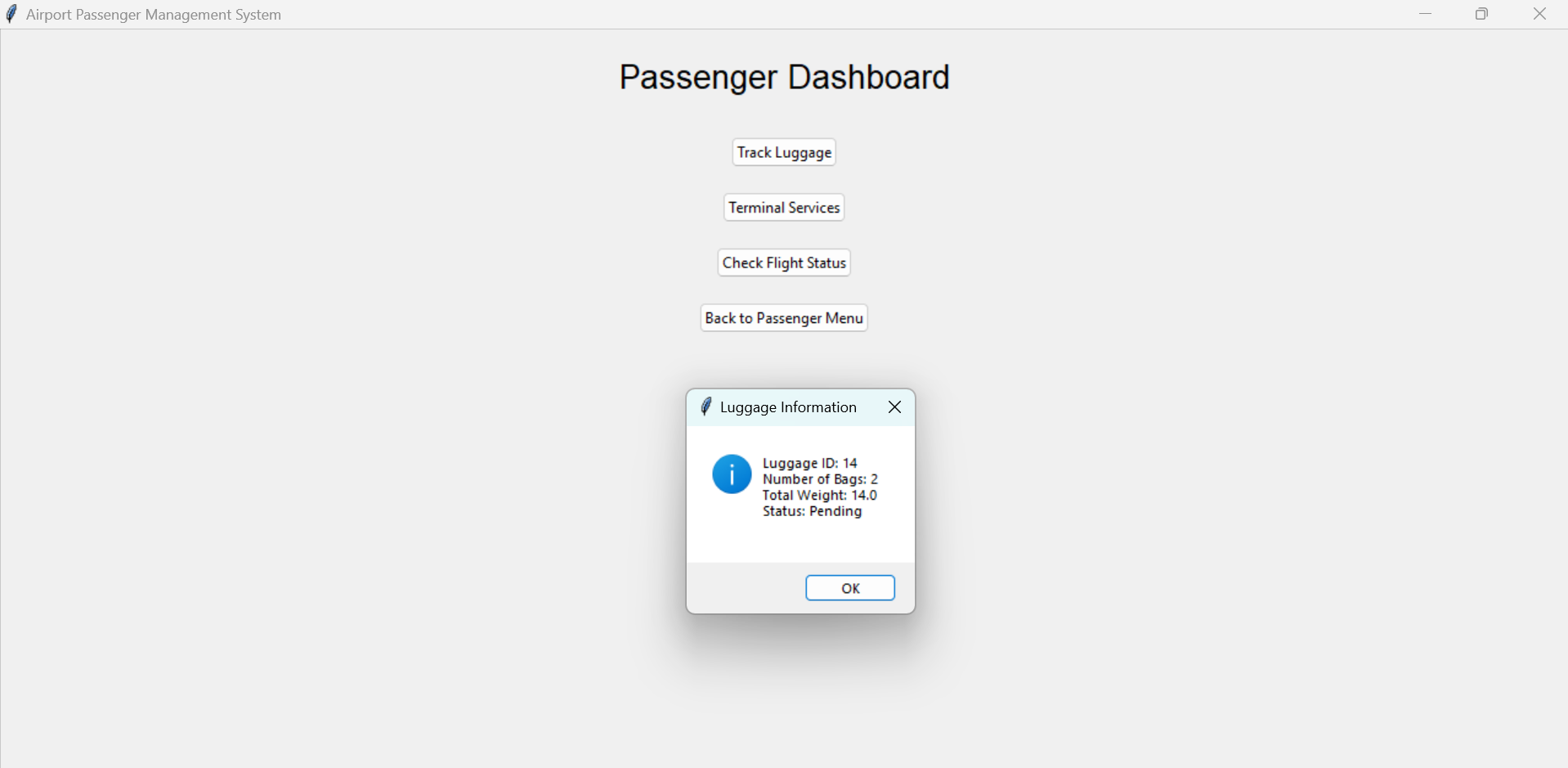
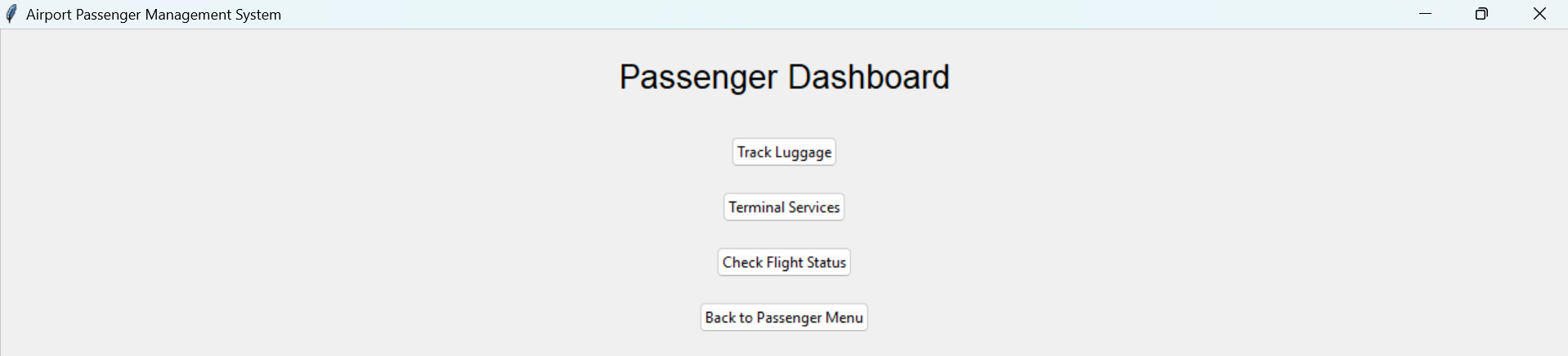
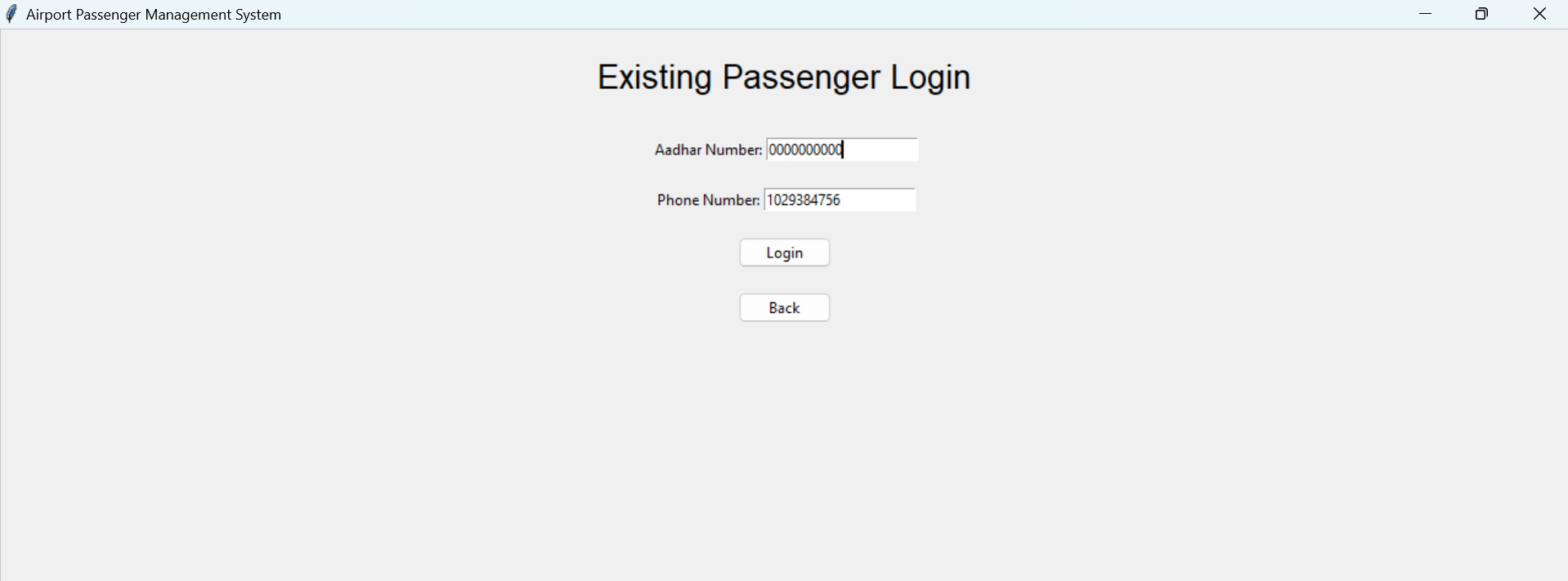
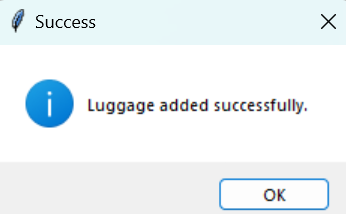
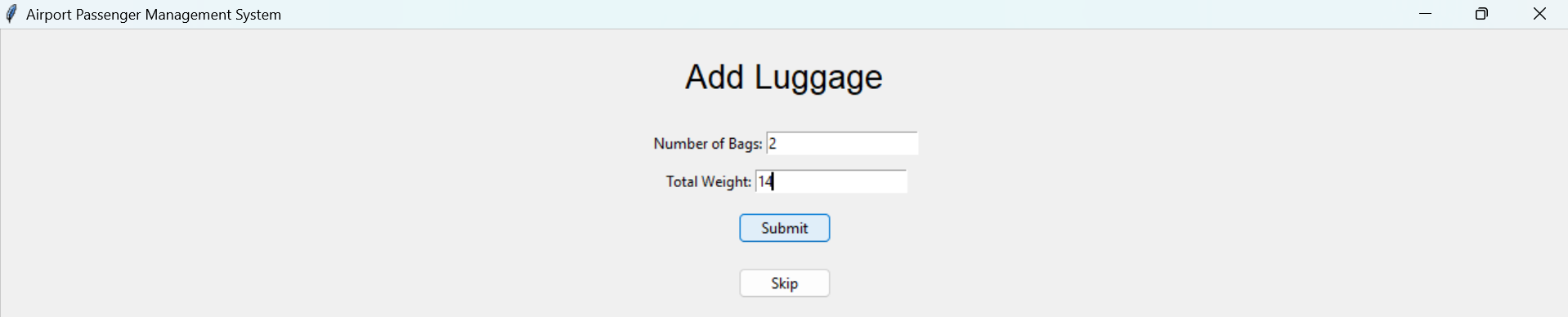
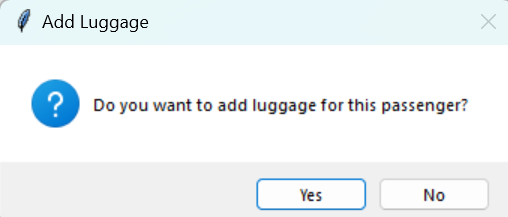
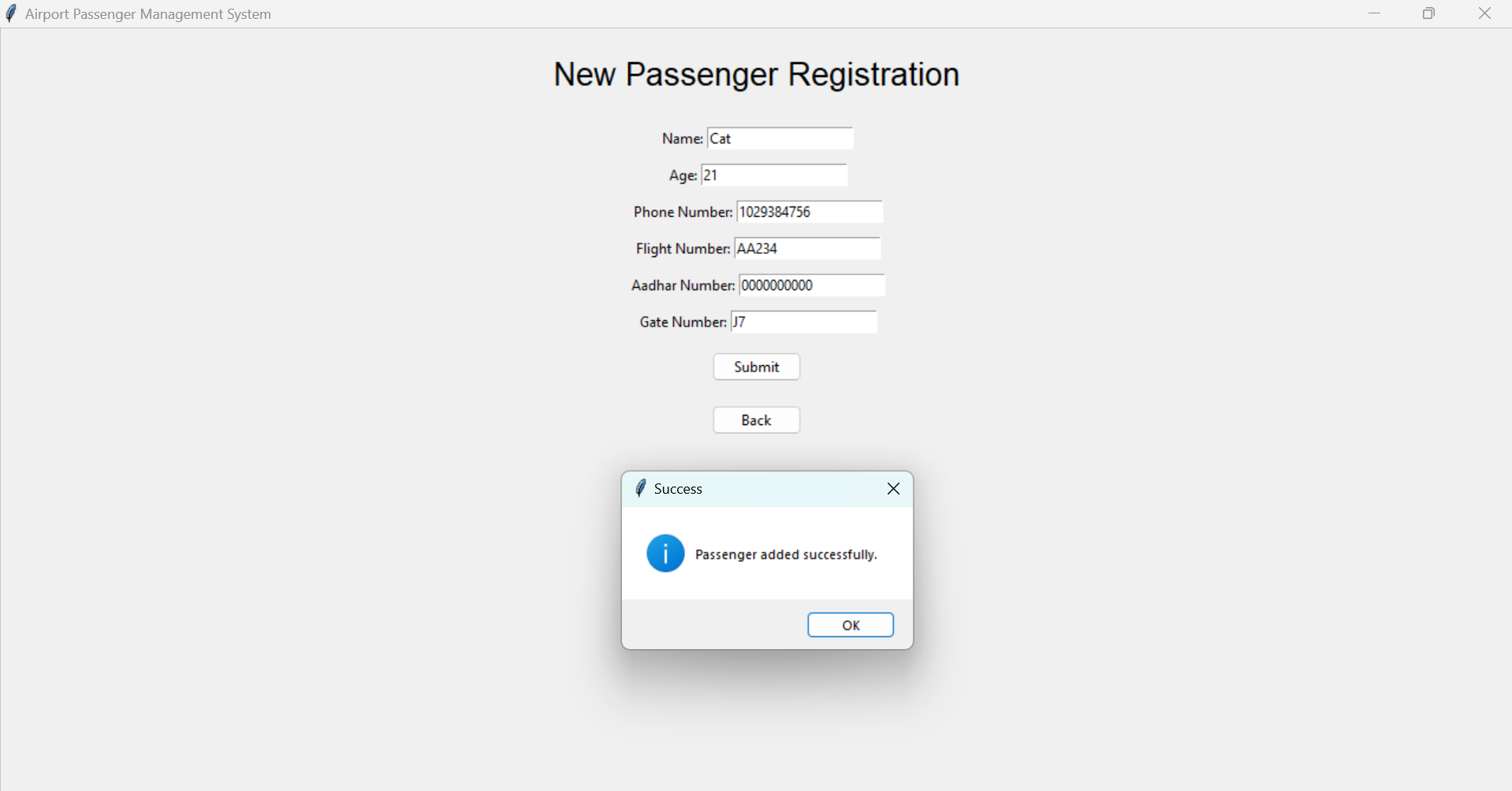
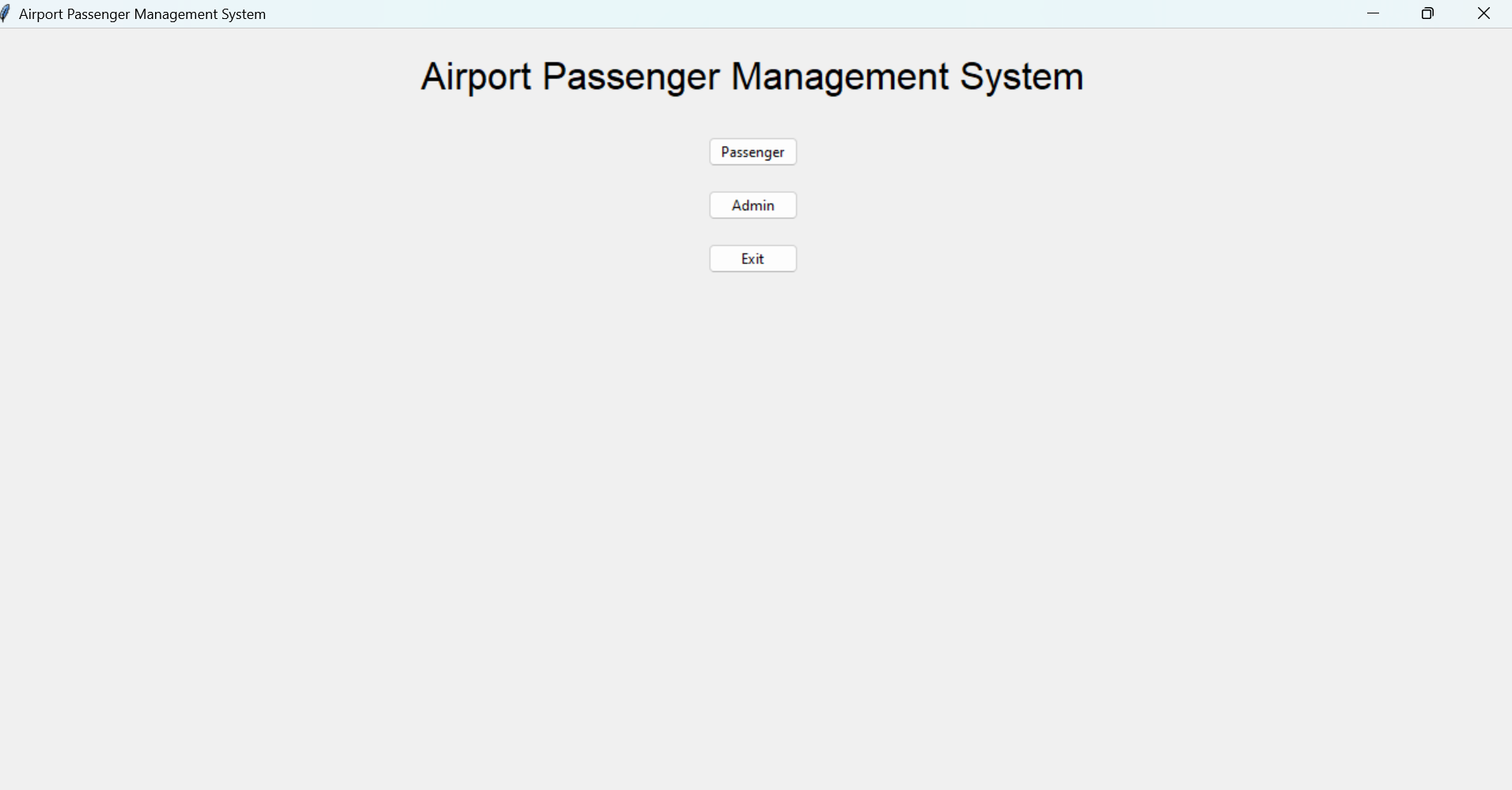
**Admin** Table

Admin\_ID | Admin\_Name

**Archive** Table

Record\_ID | Year

**Screenshots:**

****

**CODE:**

import tkinter as tk

from tkinter import ttk, messagebox

import sqlite3

from datetime import datetime

DB\_FILE = 'airport\_management.db'

def connect\_db():

    return sqlite3.connect(DB\_FILE)

def execute\_query(query, params=None):

    with connect\_db() as conn:

        cur = conn.cursor()

        if params:

            cur.execute(query, params)

        else:

            cur.execute(query)

        conn.commit()

        return cur.fetchall()

def create\_tables():

    queries = [

        """

        CREATE TABLE IF NOT EXISTS Terminal\_Service (

            Service\_id INTEGER PRIMARY KEY AUTOINCREMENT,

            Service\_name TEXT,

            Service\_charges REAL,

            Available\_time TEXT,

            Status TEXT

        )

        """,

        """

        CREATE TABLE IF NOT EXISTS Admin (

            Employee\_number TEXT PRIMARY KEY,

            Name TEXT,

            Date\_of\_birth TEXT,

            Phone\_number TEXT,

            Service\_id INTEGER,

            FOREIGN KEY (Service\_id) REFERENCES Terminal\_Service(Service\_id)

        )

        """,

        """

        CREATE TABLE IF NOT EXISTS Passenger (

            Aadhar\_number TEXT PRIMARY KEY,

            Name TEXT,

            Phone\_number TEXT,

            Age INTEGER,

            Gate\_number TEXT

        )

        """,

        """

        CREATE TABLE IF NOT EXISTS Luggage (

            Luggage\_id INTEGER PRIMARY KEY AUTOINCREMENT,

            Aadhar\_number TEXT,

            Number\_of\_bags INTEGER,

            Total\_weight REAL,

            Status TEXT,

            FOREIGN KEY (Aadhar\_number) REFERENCES Passenger(Aadhar\_number)

        )

        """,

        """

        CREATE TABLE IF NOT EXISTS Flight (

            Flight\_number TEXT PRIMARY KEY,

            Airline\_name TEXT,

            Aeroplane\_model TEXT,

            Total\_passengers INTEGER,

            Total\_weight REAL,

            Gate\_number TEXT,

            ETA TEXT,

            ETD TEXT,

            Date TEXT,

            Status TEXT

        )

        """,

        """

        CREATE TABLE IF NOT EXISTS Passenger\_Flight (

            Aadhar\_number TEXT,

            Flight\_number TEXT,

            PRIMARY KEY (Aadhar\_number, Flight\_number),

            FOREIGN KEY (Aadhar\_number) REFERENCES Passenger(Aadhar\_number),

            FOREIGN KEY (Flight\_number) REFERENCES Flight(Flight\_number)

        )

        """

    ]

    for query in queries:

        execute\_query(query)

def fill\_archive\_table():

    year = datetime.now().year

    archive\_data\_queries = [

        """

        INSERT INTO Archive (Passenger\_ID, Luggage\_ID, Service\_ID, Flight\_ID, Year)

        SELECT p.Aadhar\_number, l.Luggage\_id, a.Service\_id, pf.Flight\_number, ?

        FROM Passenger p

        LEFT JOIN Luggage l ON p.Aadhar\_number = l.Aadhar\_number

        LEFT JOIN Passenger\_Flight pf ON p.Aadhar\_number = pf.Aadhar\_number

        LEFT JOIN Admin a ON a.Service\_id = (SELECT Service\_id FROM Terminal\_Service ORDER BY RANDOM() LIMIT 1)

        """

    ]

    for query in archive\_data\_queries:

        execute\_query(query, (year,))

class AirportManagementSystem:

    def \_\_init\_\_(self, master):

        self.master = master

        self.master.title("Airport Passenger Management System")

        self.master.geometry("800x600")

        self.create\_main\_menu()

    def create\_main\_menu(self):

        self.clear\_window()

        tk.Label(self.master, text="Airport Passenger Management System", font=("Arial", 24)).pack(pady=20)

        ttk.Button(self.master, text="Passenger", command=self.passenger\_menu).pack(pady=10)

        ttk.Button(self.master, text="Admin", command=self.admin\_login).pack(pady=10)

        ttk.Button(self.master, text="Exit", command=self.master.quit).pack(pady=10)

    def clear\_window(self):

        for widget in self.master.winfo\_children():

            widget.destroy()

    def passenger\_menu(self):

        self.clear\_window()

        tk.Label(self.master, text="Passenger Menu", font=("Arial", 20)).pack(pady=20)

        ttk.Button(self.master, text="New Passenger", command=self.new\_passenger\_form).pack(pady=10)

        ttk.Button(self.master, text="Existing Passenger", command=self.existing\_passenger\_login).pack(pady=10)

        ttk.Button(self.master, text="Back to Main Menu", command=self.create\_main\_menu).pack(pady=10)

    def new\_passenger\_form(self):

        self.clear\_window()

        tk.Label(self.master, text="New Passenger Registration", font=("Arial", 20)).pack(pady=20)

        fields = ['Name', 'Age', 'Phone Number', 'Flight Number', 'Aadhar Number', 'Gate Number']

        entries = {}

        for field in fields:

            frame = tk.Frame(self.master)

            frame.pack(pady=5)

            tk.Label(frame, text=f"{field}:").pack(side=tk.LEFT)

            entry = tk.Entry(frame)

            entry.pack(side=tk.LEFT)

            entries[field] = entry

        ttk.Button(self.master, text="Submit", command=lambda: self.add\_new\_passenger(entries)).pack(pady=10)

        ttk.Button(self.master, text="Back", command=self.passenger\_menu).pack(pady=10)

    def add\_new\_passenger(self, entries):

        # Extract values from entries

        name = entries['Name'].get()

        age = entries['Age'].get()

        phone = entries['Phone Number'].get()

        flight\_number = entries['Flight Number'].get()

        aadhar = entries['Aadhar Number'].get()

        gate = entries['Gate Number'].get()

        # Perform validation here

        if not all([name, age, phone, flight\_number, aadhar, gate]):

            messagebox.showerror("Error", "Please fill in all fields.")

            return

        query = """

        INSERT INTO Passenger (Aadhar\_number, Name, Phone\_number, Age, Gate\_number)

        VALUES (?, ?, ?, ?, ?)

        """

        execute\_query(query, (aadhar, name, phone, age, gate))

        query = """

        INSERT INTO Passenger\_Flight (Aadhar\_number, Flight\_number)

        VALUES (?, ?)

        """

        execute\_query(query, (aadhar, flight\_number))

        messagebox.showinfo("Success", "Passenger added successfully.")

        # Ask if user wants to add luggage

        if messagebox.askyesno("Add Luggage", "Do you want to add luggage for this passenger?"):

            self.add\_luggage\_form(aadhar)

        else:

            self.passenger\_menu()

    def add\_luggage\_form(self, aadhar):

        self.clear\_window()

        tk.Label(self.master, text="Add Luggage", font=("Arial", 20)).pack(pady=20)

        fields = ['Number of Bags', 'Total Weight']

        entries = {}

        for field in fields:

            frame = tk.Frame(self.master)

            frame.pack(pady=5)

            tk.Label(frame, text=f"{field}:").pack(side=tk.LEFT)

            entry = tk.Entry(frame)

            entry.pack(side=tk.LEFT)

            entries[field] = entry

        ttk.Button(self.master, text="Submit", command=lambda: self.add\_new\_luggage(aadhar, entries)).pack(pady=10)

        ttk.Button(self.master, text="Skip", command=self.passenger\_menu).pack(pady=10)

    def add\_new\_luggage(self, aadhar, entries):

        # Extract values from entries

        num\_bags = entries['Number of Bags'].get()

        total\_weight = entries['Total Weight'].get()

        # Perform validation here

        if not all([num\_bags, total\_weight]):

            messagebox.showerror("Error", "Please fill in all fields.")

            return

        query = """

        INSERT INTO Luggage (Aadhar\_number, Number\_of\_bags, Total\_weight, Status)

        VALUES (?, ?, ?, ?)

        """

        execute\_query(query, (aadhar, num\_bags, total\_weight, "Pending"))

        messagebox.showinfo("Success", "Luggage added successfully.")

        self.passenger\_menu()

    def existing\_passenger\_login(self):

        self.clear\_window()

        tk.Label(self.master, text="Existing Passenger Login", font=("Arial", 20)).pack(pady=20)

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text="Aadhar Number:").pack(side=tk.LEFT)

        aadhar\_entry = tk.Entry(frame)

        aadhar\_entry.pack(side=tk.LEFT)

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text="Phone Number:").pack(side=tk.LEFT)

        phone\_entry = tk.Entry(frame)

        phone\_entry.pack(side=tk.LEFT)

        ttk.Button(self.master, text="Login", command=lambda: self.authenticate\_passenger(aadhar\_entry.get(), phone\_entry.get())).pack(pady=10)

        ttk.Button(self.master, text="Back", command=self.passenger\_menu).pack(pady=10)

    def authenticate\_passenger(self, aadhar, phone):

        query = """

        SELECT \* FROM Passenger WHERE Aadhar\_number = ? AND Phone\_number = ?

        """

        result = execute\_query(query, (aadhar, phone))

        if result:

            self.passenger\_dashboard(aadhar)

        else:

            messagebox.showerror("Error", "Authentication failed.")

    def passenger\_dashboard(self, aadhar):

        self.clear\_window()

        tk.Label(self.master, text="Passenger Dashboard", font=("Arial", 20)).pack(pady=20)

        ttk.Button(self.master, text="Track Luggage", command=lambda: self.track\_luggage(aadhar)).pack(pady=10)

        ttk.Button(self.master, text="Terminal Services", command=self.show\_terminal\_services).pack(pady=10)

        ttk.Button(self.master, text="Check Flight Status", command=lambda: self.check\_flight\_status(aadhar)).pack(pady=10)

        ttk.Button(self.master, text="Back to Passenger Menu", command=self.passenger\_menu).pack(pady=10)

    def track\_luggage(self, aadhar):

        query = """

        SELECT \* FROM Luggage WHERE Aadhar\_number = ?

        """

        result = execute\_query(query, (aadhar,))

        if result:

            luggage\_info = ""

            for luggage in result:

                luggage\_info += f"Luggage ID: {luggage[0]}\n"

                luggage\_info += f"Number of Bags: {luggage[2]}\n"

                luggage\_info += f"Total Weight: {luggage[3]}\n"

                luggage\_info += f"Status: {luggage[4]}\n\n"

            messagebox.showinfo("Luggage Information", luggage\_info)

        else:

            messagebox.showinfo("Luggage Information", "No luggage found for this passenger.")

    def check\_flight\_status(self, aadhar):

        query = """

        SELECT f.Flight\_number, fs.Status

        FROM Passenger\_Flight pf

        JOIN Flight f ON f.Flight\_number = pf.Flight\_number

        JOIN flight\_status fs ON pf.Flight\_number = fs.Status

        WHERE pf.aadhar\_number = ?

        """

        result = execute\_query(query, (aadhar,))

        if result:

            flight\_info = ""

            for flight in result:

                flight\_info += f"Flight Number: {flight[0]}\n"

                flight\_info += f"Status: {flight[1]}\n\n"

            messagebox.showinfo("Flight Information", flight\_info)

        else:

            messagebox.showinfo("Flight Information", "No flight information found for this passenger.")

    def admin\_login(self):

        self.clear\_window()

        tk.Label(self.master, text="Admin Login", font=("Arial", 20)).pack(pady=20)

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text="Employee Number:").pack(side=tk.LEFT)

        emp\_num\_entry = tk.Entry(frame)

        emp\_num\_entry.pack(side=tk.LEFT)

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text="Date of Birth (YYYY-MM-DD):").pack(side=tk.LEFT)

        dob\_entry = tk.Entry(frame)

        dob\_entry.pack(side=tk.LEFT)

        ttk.Button(self.master, text="Login", command=lambda: self.authenticate\_admin(emp\_num\_entry.get(), dob\_entry.get())).pack(pady=10)

        ttk.Button(self.master, text="Back to Main Menu", command=self.create\_main\_menu).pack(pady=10)

    def authenticate\_admin(self, emp\_num, dob):

        query = """

        SELECT \* FROM Admin WHERE Employee\_number = ? AND Date\_of\_birth = ?

        """

        result = execute\_query(query, (emp\_num, dob))

        if result:

            self.admin\_dashboard()

        else:

            messagebox.showerror("Error", "Authentication failed.")

    def admin\_dashboard(self):

        self.clear\_window()

        tk.Label(self.master, text="Admin Dashboard", font=("Arial", 20)).pack(pady=20)

        ttk.Button(self.master, text="Track Passenger", command=self.track\_passenger\_form).pack(pady=10)

        ttk.Button(self.master, text="Track Flight", command=self.track\_flight\_form).pack(pady=10)

        ttk.Button(self.master, text="Track Luggage", command=self.track\_luggage\_by\_id\_form).pack(pady=10)

        ttk.Button(self.master, text="Add Flight", command=self.add\_flight\_form).pack(pady=10)

        ttk.Button(self.master, text="Change Status", command=self.change\_status\_menu).pack(pady=10)

        ttk.Button(self.master, text="List Terminal Services", command=self.show\_terminal\_services).pack(pady=10)

        ttk.Button(self.master, text="Back to Main Menu", command=self.create\_main\_menu).pack(pady=10)

    def change\_status\_menu(self):

        self.clear\_window()

        tk.Label(self.master, text="Change Status", font=("Arial", 20)).pack(pady=20)

        ttk.Button(self.master, text="Passenger Status", command=lambda: self.change\_status\_form("passenger")).pack(pady=10)

        ttk.Button(self.master, text="Flight Status", command=lambda: self.change\_status\_form("flight")).pack(pady=10)

        ttk.Button(self.master, text="Luggage Status", command=lambda: self.change\_status\_form("luggage")).pack(pady=10)

        ttk.Button(self.master, text="Back to Admin Dashboard", command=self.admin\_dashboard).pack(pady=10)

    def change\_status\_form(self, status\_type):

        self.clear\_window()

        tk.Label(self.master, text=f"Change {status\_type.capitalize()} Status", font=("Arial", 20)).pack(pady=20)

        if status\_type == "passenger":

            id\_label = "Aadhar Number"

            id\_column = "Aadhar\_number"

            table = "passenger\_status"

            status\_table = "set\_passenger\_status"

        elif status\_type == "flight":

            id\_label = "Flight Number"

            id\_column = "Flight\_number"

            table = "Flight\_status"

            status\_table = "set\_flight\_status"

        else:  # luggage

            id\_label = "Luggage ID"

            id\_column = "Luggage\_number"

            table = "Luggage\_status"

            status\_table = "set\_luggage\_status"

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text=f"{id\_label}:").pack(side=tk.LEFT)

        id\_entry = tk.Entry(frame)

        id\_entry.pack(side=tk.LEFT)

        status\_options = self.get\_status\_options(status\_table)

        status\_var = tk.StringVar(self.master)

        status\_var.set(status\_options[0])  # Set default value

        tk.Label(self.master, text="New Status:").pack()

        status\_menu = ttk.Combobox(self.master, textvariable=status\_var, values=status\_options)

        status\_menu.pack(pady=10)

        ttk.Button(self.master, text="Update Status",

                   command=lambda: self.update\_status(status\_type, id\_entry.get(), status\_var.get(), id\_column, table)).pack(pady=10)

        ttk.Button(self.master, text="Back", command=self.change\_status\_menu).pack(pady=10)

    def get\_status\_options(self, status\_table):

        try:

            query = f"SELECT status\_name FROM {status\_table}"

            result = execute\_query(query)

            return [status[0] for status in result] if result else []

        except sqlite3.Error as e:

            print(f"An error occurred: {e}")

            messagebox.showerror("Error", f"Failed to get status options: {e}")

            return []

    def update\_status(self, status\_type, id\_value, new\_status, id\_column, table):

        if not id\_value or not new\_status:

            messagebox.showerror("Error", "Please fill in all fields.")

            return

        query = f"UPDATE {table} SET Status = ? WHERE {id\_column} = ?"

        execute\_query(query, (new\_status, id\_value))

        if status\_type == "passenger":

            # Update or insert into passenger\_status table

            check\_query = "SELECT \* FROM passenger\_status WHERE Aadhar\_number = ?"

            result = execute\_query(check\_query, (id\_value,))

            if result:

                update\_query = "UPDATE passenger\_status SET Status = ? WHERE Aadhar\_number = ?"

            else:

                update\_query = "INSERT INTO passenger\_status (Status, Aadhar\_number) VALUES (?, ?)"

            execute\_query(update\_query, (new\_status, id\_value))

        messagebox.showinfo("Success", f"{status\_type.capitalize()} status updated successfully.")

        self.change\_status\_menu()

    def track\_passenger\_form(self):

        self.clear\_window()

        tk.Label(self.master, text="Track Passenger", font=("Arial", 20)).pack(pady=20)

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text="Aadhar Number:").pack(side=tk.LEFT)

        aadhar\_entry = tk.Entry(frame)

        aadhar\_entry.pack(side=tk.LEFT)

        ttk.Button(self.master, text="Track", command=lambda: self.track\_passenger(aadhar\_entry.get())).pack(pady=10)

        ttk.Button(self.master, text="Back", command=self.admin\_dashboard).pack(pady=10)

    def track\_passenger(self, aadhar):

        query = """

        SELECT p.\*, l.Luggage\_id, l.Number\_of\_bags, l.Total\_weight, l.Status, f.Flight\_number, fs.Status

        FROM Passenger p

        LEFT JOIN Luggage l ON p.Aadhar\_number = l.Aadhar\_number

        LEFT JOIN Passenger\_Flight pf ON p.Aadhar\_number = pf.Aadhar\_number

        LEFT JOIN Flight f ON pf.Flight\_number = f.Flight\_number

        LEFT JOIN passenger\_status fs ON pf.Aadhar\_number=fs.Aadhar\_number

        WHERE p.Aadhar\_number = ?

        """

        result = execute\_query(query, (aadhar,))

        query1 = """SELECT Status FROM passenger\_status WHERE Aadhar\_number= ?"""

        result1=execute\_query(query1, (aadhar,))

        if result:

            passenger\_info = f"""

            Name: {result[0][1]}

            Age: {result[0][3]}

            Phone: {result[0][2]}

            Gate Number: {result[0][4]}

            Luggage ID: {result[0][5]}

            Number of Bags: {result[0][6]}

            Total Weight: {result[0][7]}

            Flight Number: {result[0][9]}

            passenger Status: {result1}

            """

            messagebox.showinfo("Passenger Information", passenger\_info)

        else:

            messagebox.showinfo("Passenger Information", "Passenger not found.")

    def track\_flight\_form(self):

        self.clear\_window()

        tk.Label(self.master, text="Track Flight", font=("Arial", 20)).pack(pady=20)

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text="Flight Number:").pack(side=tk.LEFT)

        flight\_entry = tk.Entry(frame)

        flight\_entry.pack(side=tk.LEFT)

        ttk.Button(self.master, text="Track", command=lambda: self.track\_flight(flight\_entry.get())).pack(pady=10)

        ttk.Button(self.master, text="Back", command=self.admin\_dashboard).pack(pady=10)

    def track\_flight(self, flight\_num):

        query = "SELECT \* FROM Flight WHERE Flight\_number = ?"

        query1 = "SELECT Status FROM flight\_status WHERE Flight\_number = ?"

        result = execute\_query(query, (flight\_num,))

        stats = execute\_query(query1, (flight\_num,))

        if stats:

            stats=stats

        else:

            stats = result[0][9]

        if result:

            flight\_info = f"""

            Airline: {result[0][1]}

            Model: {result[0][2]}

            Total Passengers: {result[0][3]}

            Total Weight: {result[0][4]}

            Gate: {result[0][5]}

            ETA: {result[0][6]}

            ETD: {result[0][7]}

            Date: {result[0][8]}

            Status: {stats}

            """

            messagebox.showinfo("Flight Information", flight\_info)

        else:

            messagebox.showinfo("Flight Information", "Flight not found.")

    def track\_luggage\_by\_id\_form(self):

        self.clear\_window()

        tk.Label(self.master, text="Track Luggage by ID", font=("Arial", 20)).pack(pady=20)

        frame = tk.Frame(self.master)

        frame.pack(pady=10)

        tk.Label(frame, text="Enter Luggage ID:").pack(side=tk.LEFT)

        luggage\_id\_entry = tk.Entry(frame)

        luggage\_id\_entry.pack(side=tk.LEFT)

        ttk.Button(self.master, text="Track", command=lambda: self.track\_luggage\_by\_id(luggage\_id\_entry.get())).pack(pady=10)

        ttk.Button(self.master, text="Back to Dashboard", command=self.admin\_dashboard).pack(pady=10)

    def track\_luggage\_by\_id(self, luggage\_id):

        query = """

        SELECT \* FROM Luggage WHERE Luggage\_id = ?

        """

        result = execute\_query(query, (luggage\_id,))

        if result:

            self.clear\_window()

            tk.Label(self.master, text="Luggage Details", font=("Arial", 20)).pack(pady=20)

            tk.Label(self.master, text=f"Luggage ID: {result[0][0]}").pack()

            tk.Label(self.master, text=f"Aadhar Number: {result[0][1]}").pack()

            tk.Label(self.master, text=f"Number of Bags: {result[0][2]}").pack()

            tk.Label(self.master, text=f"Total Weight: {result[0][3]}").pack()

            tk.Label(self.master, text=f"Status: {result[0][4]}").pack()

        else:

            messagebox.showerror("Error", "Luggage ID not found.")

        ttk.Button(self.master, text="Back to Dashboard", command=self.admin\_dashboard).pack(pady=10)

    def add\_flight\_form(self):

        self.clear\_window()

        tk.Label(self.master, text="Add New Flight", font=("Arial", 20)).pack(pady=20)

        fields = ['Flight Number', 'Airline Name', 'Aeroplane Model', 'Total Passengers',

                  'Total Weight', 'Gate Number', 'ETA', 'ETD', 'Date (YYYY-MM-DD)', 'Status']

        entries = {}

        for field in fields:

            frame = tk.Frame(self.master)

            frame.pack(pady=5)

            tk.Label(frame, text=f"{field}:").pack(side=tk.LEFT)

            entry = tk.Entry(frame)

            entry.pack(side=tk.LEFT)

            entries[field] = entry

        ttk.Button(self.master, text="Submit", command=lambda: self.add\_new\_flight(entries)).pack(pady=10)

        ttk.Button(self.master, text="Back", command=self.admin\_dashboard).pack(pady=10)

    def add\_new\_flight(self, entries):

        # Extract values from entries

        flight\_number = entries['Flight Number'].get()

        airline\_name = entries['Airline Name'].get()

        aeroplane\_model = entries['Aeroplane Model'].get()

        total\_passengers = entries['Total Passengers'].get()

        total\_weight = entries['Total Weight'].get()

        gate\_number = entries['Gate Number'].get()

        eta = entries['ETA'].get()

        etd = entries['ETD'].get()

        date = entries['Date (YYYY-MM-DD)'].get()

        status = entries['Status'].get()

        # Perform validation here

        if not all([flight\_number, airline\_name, aeroplane\_model, total\_passengers, total\_weight, gate\_number, eta, etd, date, status]):

            messagebox.showerror("Error", "Please fill in all fields.")

            return

        query = """

        INSERT INTO Flight (Flight\_number, Airline\_name, Aeroplane\_model, Total\_passengers,

                            Total\_weight, Gate\_number, ETA, ETD, Date, Status)

        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

        """

        execute\_query(query, (flight\_number, airline\_name, aeroplane\_model, total\_passengers,

                              total\_weight, gate\_number, eta, etd, date, status))

        messagebox.showinfo("Success", "Flight added successfully.")

        self.admin\_dashboard()

    def show\_terminal\_services(self):

        query = "SELECT \* FROM Terminal\_Service"

        result = execute\_query(query)

        if result:

            service\_info = ""

            for service in result:

                service\_info += f"Service ID: {service[0]}\n"

                service\_info += f"Service Name: {service[1]}\n"

                service\_info += f"Service Charges: {service[2]}\n"

                service\_info += f"Available Time: {service[3]}\n"

                service\_info += f"Status: {service[4]}\n\n"

            messagebox.showinfo("Terminal Services", service\_info)

        else:

            messagebox.showinfo("Terminal Services", "No terminal services found.")

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

    create\_tables()

    root = tk.Tk()

    app = AirportManagementSystem(root)

    root.mainloop()

**SQL Query**

**-- Inserting data into Terminal\_Service table**

**INSERT INTO Terminal\_Service (Service\_name, Service\_charges, Available\_time, Status)**

**VALUES**

**('Cleaning', 100.0, '09:00-18:00', 'Available'),**

**('Catering', 200.0, '10:00-20:00', 'Available'),**

**('Security', 150.0, '24/7', 'Available'),**

**('Maintenance', 250.0, '08:00-22:00', 'Available'),**

**('Information Desk', 50.0, '06:00-23:00', 'Available');**

**-- Inserting data into Admin table**

**INSERT INTO Admin (Employee\_number, Name, Date\_of\_birth, Phone\_number)**

**VALUES**

**('EMP001', 'John Doe', '1980-01-01', '1234567890'),**

**('EMP002', 'Jane Smith', '1985-02-02', '0987654321'),**

**('EMP003', 'Emma Johnson', '1990-03-03', '1231231231'),**

**('EMP004', 'Chris Brown', '1975-04-04', '4564564564'),**

**('EMP005', 'Olivia Davis', '1988-05-05', '7897897897');**

**-- Inserting data into Passenger table**

**INSERT INTO Passenger (Aadhar\_number, Name, Phone\_number, Age, Gate\_number)**

**VALUES**

**('AAD001', 'Alice Johnson', '1111111111', 30, 'G1'),**

**('AAD002', 'Bob Brown', '2222222222', 25, 'G2'),**

**('AAD003', 'Charlie Black', '3333333333', 35, 'G3'),**

**('AAD004', 'David White', '4444444444', 28, 'G4'),**

**('AAD005', 'Eve Green', '5555555555', 22, 'G5');**

**-- Inserting data into Flight table**

**INSERT INTO Flight (Flight\_number, Airline\_name, Aeroplane\_model, Total\_passengers, Total\_weight, Gate\_number, ETA, ETD, Date, Status)**

**VALUES**

**('FL001', 'Airline1', 'Boeing737', 180, 15000.0, 'G1', '12:00', '14:00', '2023-06-01', 'On Time'),**

**('FL002', 'Airline2', 'AirbusA320', 160, 14000.0, 'G2', '13:00', '15:00', '2023-06-01', 'Delayed'),**

**('FL003', 'Airline3', 'Boeing747', 200, 18000.0, 'G3', '14:00', '16:00', '2023-06-01', 'On Time'),**

**('FL004', 'Airline4', 'AirbusA380', 220, 20000.0, 'G4', '15:00', '17:00', '2023-06-01', 'Delayed'),**

**('FL005', 'Airline5', 'Boeing777', 240, 22000.0, 'G5', '16:00', '18:00', '2023-06-01', 'On Time');**

**-- Creating the Archive table**

**CREATE TABLE IF NOT EXISTS Archive (**

**Archive\_ID INTEGER PRIMARY KEY AUTOINCREMENT,**

**Passenger\_ID TEXT,**

**Luggage\_ID INTEGER,**

**Service\_ID INTEGER,**

**Flight\_ID TEXT,**

**Year INTEGER,**

**FOREIGN KEY (Passenger\_ID) REFERENCES Passenger(Aadhar\_number),**

**FOREIGN KEY (Luggage\_ID) REFERENCES Luggage(Luggage\_id),**

**FOREIGN KEY (Service\_ID) REFERENCES Terminal\_Service(Service\_id),**

**FOREIGN KEY (Flight\_ID) REFERENCES Flight(Flight\_number)**

**);**

**-- Query to find the total number of passengers archived per year having more than one archive entry**

**SELECT Year, COUNT(\*) AS Total\_Passengers**

**FROM Archive**

**GROUP BY Year**

**HAVING COUNT(\*) > 1;**

**-- Query to get the list of archived flights ordered by year**

**SELECT Flight\_ID, Year**

**FROM Archive**

**ORDER BY Year;**

**-- Query to get archived passenger details along with their flight details using INNER JOIN**

**SELECT p.Aadhar\_number, p.Name, f.Flight\_number, f.Airline\_name**

**FROM Archive a**

**INNER JOIN Passenger p ON a.Passenger\_ID = p.Aadhar\_number**

**INNER JOIN Flight f ON a.Flight\_ID = f.Flight\_number;**

**-- Query to get all archived passengers and their luggage details using LEFT JOIN**

**SELECT p.Aadhar\_number, p.Name, l.Luggage\_id, l.Status**

**FROM Archive a**

**LEFT JOIN Passenger p ON a.Passenger\_ID = p.Aadhar\_number**

**LEFT JOIN Luggage l ON a.Luggage\_ID = l.Luggage\_id;**

**-- Query to find archived passengers who have more than one bag or total weight greater than 12 kg**

**SELECT a.Passenger\_ID, l.Number\_of\_bags, l.Total\_weight**

**FROM Archive a**

**INNER JOIN Luggage l ON a.Luggage\_ID = l.Luggage\_id**

**WHERE l.Number\_of\_bags > 1 OR l.Total\_weight > 12.0;**

**-- Query to calculate the average weight per archived passenger**

**SELECT Flight\_ID, AVG(Total\_weight / Total\_passengers) AS Avg\_Weight\_Per\_Passenger**

**FROM Archive a**

**INNER JOIN Flight f ON a.Flight\_ID = f.Flight\_number**

**GROUP BY Flight\_ID;**

**-- Query to find archived passengers whose name contains 'Alice'**

**SELECT p.Aadhar\_number, p.Name, p.Phone\_number**

**FROM Archive a**

**INNER JOIN Passenger p ON a.Passenger\_ID = p.Aadhar\_number**

**WHERE p.Name LIKE '%Alice%';**

**-- SQLite does not support to\_char or extract directly.**

**-- Query to get the year from the archive date**

**SELECT Archive\_ID, Year**

**FROM Archive;**

**-- Query to find archived flights between specific years**

**SELECT Flight\_ID, Year**

**FROM Archive**

**WHERE Year BETWEEN 2022 AND 2023;**

**-- Query to find archived passengers whose Aadhar\_number is in a specified list**

**SELECT Passenger\_ID**

**FROM Archive**

**WHERE Passenger\_ID IN ('100000000', '200000000');**

**-- Query to find archived passengers whose year is not between 2020 and 2022**

**SELECT Passenger\_ID, Year**

**FROM Archive**

**WHERE Year NOT BETWEEN 2020 AND 2022;**

**-- SQLite does not support INTERSECT and EXCEPT directly, subqueries.**

**-- Query to find passengers who have luggage archived**

**SELECT Passenger\_ID FROM Archive**

**WHERE Luggage\_ID IS NOT NULL;**

**-- Query to find passengers who do not have luggage archived**

**SELECT Passenger\_ID FROM Archive**

**WHERE Luggage\_ID IS NULL;**

**-- Query to find archived passengers who have Loaded luggage**

**SELECT a.Passenger\_ID, p.Name**

**FROM Archive a**

**INNER JOIN Passenger p ON a.Passenger\_ID = p.Aadhar\_number**

**WHERE EXISTS (SELECT 1 FROM Luggage l WHERE a.Luggage\_ID = l.Luggage\_id AND l.Status = 'Loaded');**

**-- Query to find archived flights where all passengers have luggage archived**

**SELECT Flight\_ID**

**FROM Archive a**

**WHERE NOT EXISTS (**

**SELECT 1**

**FROM Passenger\_Flight pf**

**LEFT JOIN Luggage l ON pf.Aadhar\_number = l.Aadhar\_number**

**WHERE pf.Flight\_number = a.Flight\_ID AND l.Luggage\_id IS NULL**

**);**

**-- Query to find archived flights where all passengers have luggage archived**

**SELECT Flight\_ID**

**FROM Archive a**

**WHERE NOT EXISTS (**

**SELECT 1**

**FROM Passenger\_Flight pf**

**LEFT JOIN Luggage l ON pf.Aadhar\_number = l.Aadhar\_number**

**WHERE pf.Flight\_number = a.Flight\_ID AND l.Luggage\_id IS NULL**

**);**