**AIRLINE MANAGEMENT SYSTEM**

A picture containing text, clipart

Description automatically generated

**AMS**

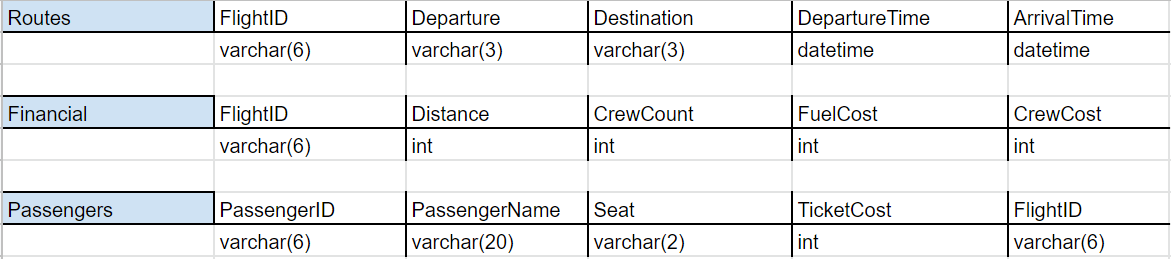
By Soham Das

**USER MANUAL**

The project aims to automate a lot of the management related tasks in the airline industry and simplify the process of booking a ticket by reducing any manual processing.

On running the program, a menu will appear with numbers corresponding to each option. Enter a number when prompted to do so.

The tables are defined as follows:



“FlightID” is a primary key in the table “Routes” and is a foreign key in the rest of the tables.

**Hardware Requirements:**

* A Computer/Laptop with Operating System-Windows 7 or above x86 64-bit CPU (Intel / AMD architecture)
* 512 MB of RAM
* 128 MB of available disk space

**Software Requirements:**

* Python 3.6.x or higher
* MySQL

**Limitations:**

* The project is not web based
* Lacks GUI
* The system cannot suggest routes
* Not applicable at larger scales

**SOURCE CODE**

# Importing mysql-python connectivity tool and sleep function for added realism.

import mysql.connector as sql

from time import sleep

# Connecting to the MySQL database.

print("Connecting...")

sleep(1)

try:

con = sql.connect(host = "localhost", user = "root", database = "project", password = "tiger")

print("Connection successful. Welcome to the Airline Management System.")

except:

print("Connection unsuccessful. Please check your credentials and try again.")

exit()

print()

# Creating cursor

cur = con.cursor()

cur.execute("use project")

# Function to create tables

def createTables():

try:

cur.execute("create table Routes (FlightID varchar(6) primary key, Departure varchar(3), Destination varchar(3), DepartureTime datetime, ArrivalTime datetime)")

cur.execute("create table Financial (FlightID varchar(6), Distance int, CrewCount int, FuelCost int, CrewCost int, foreign key(FlightID) references Routes(FlightID))")

cur.execute("create table Passengers (PassengerID varchar(6) primary key, PassengerName varchar(20), Seat varchar(2), TicketCost int, FlightID varchar(6), foreign key(FlightID) references Routes(FlightID))")

except:

print("Tables already exist.")

# Insert functions

def insertIntoRoutes():

while True:

try:

flightID = input("Enter the Flight ID: ")

departure = input("Enter departure location: ")

destination = input("Enter destination location: ")

deptime = input("Enter departure time: ")

artime = input("Enter arrival time: ")

cur.execute(f"insert into Routes values ('{flightID}', '{departure}', '{destination}', '{deptime}', '{artime}')")

con.commit()

print("Route added.")

except:

con.rollback()

print("Route couldn't be added. Please try again.")

print()

choice = input("Do you want to continue (Y/N)?: ")

if choice.lower() == 'n':

print()

break

def insertIntoFinancial():

while True:

try:

flightID = input("Enter the Flight ID: ")

distance = int(input("Enter distance of flight: "))

crewcount = int(input("Enter crew count: "))

fuelcost = int(input("Enter fuel cost: "))

crewcost = int(input("Enter the cost for each crew member: "))

cur.execute(f"insert into Financial values ('{flightID}', {distance}, {crewcount}, {fuelcost}, {crewcost})")

con.commit()

print("Record added.")

except:

con.rollback()

print("Record couldn't be added. Please try again.")

print()

choice = input("Do you want to continue (Y/N)?: ")

if choice.lower() == 'n':

print()

break

def insertIntoPassengers():

while True:

try:

passID = input("Enter the Passenger ID: ")

passname = input("Enter name of the passenger: ")

seat = input("Enter seat name: ")

ticketcost = int(input("Enter cost of ticket: "))

flightID = input("Enter Flight ID: ")

cur.execute(f"insert into Passengers values ('{passID}', '{passname}', '{seat}', {ticketcost}, '{flightID}')")

con.commit()

print("Ticket booked.")

except:

con.rollback()

print("Ticket couldn't be booked. Please try again.")

print()

choice = input("Do you want to continue (Y/N)?: ")

if choice.lower() == 'n':

print()

break

# Update functions

def updateRoutes():

try:

flid = input("Enter the flight ID of the record you want to update: ")

dep = input("Enter departure location: ")

destination = input("Enter destination location: ")

deptime = input("Enter departure time: ")

artime = input("Enter arrival time: ")

cur.execute(f"update routes set departure = '{dep}', destination = '{destination}', departuretime = '{deptime}', arrivaltime = '{artime}' where flightid = '{flid}'")

con.commit()

print("Record updated.")

except:

con.rollback()

print("Record couldn't be updated. Please try again.")

print()

def updateFinancial():

try:

flightID = input("Enter the flight ID of the record you want to update: ")

distance = int(input("Enter distance of flight: "))

crewcount = int(input("Enter crew count: "))

fuelcost = int(input("Enter fuel cost: "))

crewcost = int(input("Enter cost of each crew member: "))

cur.execute(f"update financial set distance = {distance}, crewcount = {crewcount}, fuelcost = {fuelcost}, crewcost = {crewcost} where flightid = '{flightID}'")

con.commit()

print("Record updated.")

except:

con.rollback()

print("Record couldn't be updated. Please try again.")

print()

def updatePassengers():

try:

passID = input("Enter the passenger ID of the record you want to update: ")

passname = input("Enter name of the passenger: ")

seat = input("Enter seat name: ")

ticketcost = float(input("Enter cost of ticket: "))

cur.execute(f"update passengers set passengername = '{passname}', seat = '{seat}', ticketcost = {ticketcost} where passengerID = '{passID}'")

con.commit()

print("Record updated.")

except:

con.rollback()

print("Record couldn't be updated. Please try again.")

print()

# Delete function

def deletePassenger():

pid = input("Enter the passenger ID of the record you want to delete: ")

try:

cur.execute(f"delete from passengers where passengerID = '{pid}'")

con.commit()

print("Passenger record deleted.")

except:

con.rollback()

print("Record couldn't be deleted. Please try again.")

print()

# Display functions

def displayTable(tableName):

cur.execute(f"select \* from {tableName}")

for i in cur:

print(i)

def displayProfit():

flid = input("Enter the flight ID: ")

try:

cur.execute(f"select sum(ticketcost) from passengers where flightID = '{flid}'")

for i in cur:

revenue = i[0]

cur.execute(f"select \* from financial where flightID = '{flid}'")

for i in cur:

distance, crewcount, fuelcost, crewcost = i[1:]

totalcost = distance\*fuelcost + crewcount\*crewcost

profit = revenue - totalcost

print(f"Profit: ${profit}")

except:

print("Flight not found. Please try again.")

# Menu

while True:

print('''Menu:

1) Display routes

2) Display passengers

3) Display financial information

4) Display profit from a flight

5) Add routes

6) Book a passenger ticket

7) Add financial information

8) Update financial information

9) Update route information

10) Update passenger information

11) Remove passenger record

12) Exit\n''')

choice = int(input("Enter your choice: "))

if choice == 1: displayTable("routes")

elif choice == 2: displayTable("passengers")

elif choice == 3: displayTable("financial")

elif choice == 4: displayProfit()

elif choice == 5: insertIntoRoutes()

elif choice == 6: insertIntoPassengers()

elif choice == 7: insertIntoFinancial()

elif choice == 8: updateFinancial()

elif choice == 9: updateRoutes()

elif choice == 10: updatePassengers()

elif choice == 11: deletePassenger()

elif choice == 12:

print("Program shutting down...")

sleep(1)

print("Thank you for using AMS.")

break

else:

print("Invalid choice. Please choose a number from the menu.")

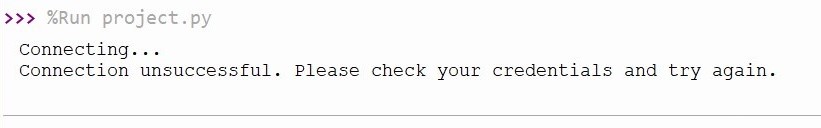
print()

**SCREENSHOTS**

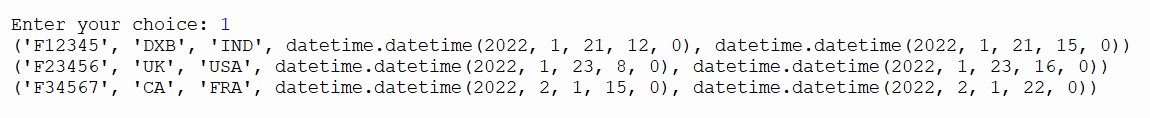
Running the program with correct credentials:



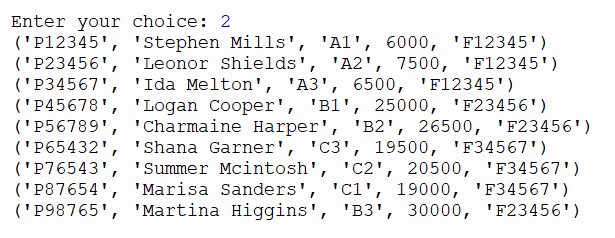
Running the program with incorrect credentials:



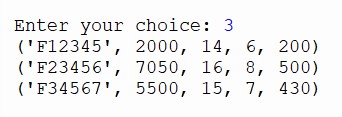
Displaying routes:



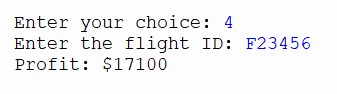
Displaying passengers:



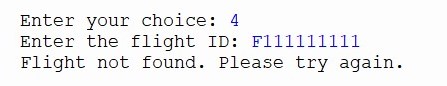
Displaying financial information:



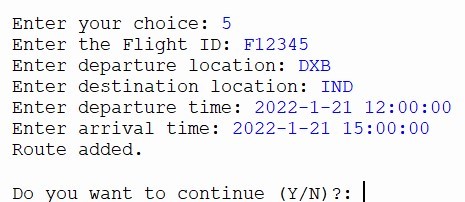
Displaying profit from a flight:



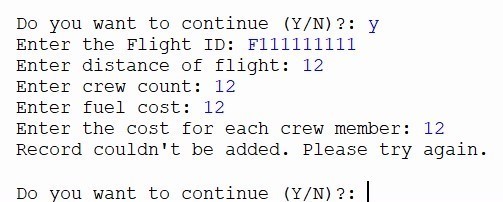
*Exception:*

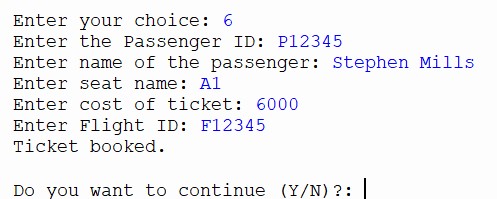
**

Adding routes:

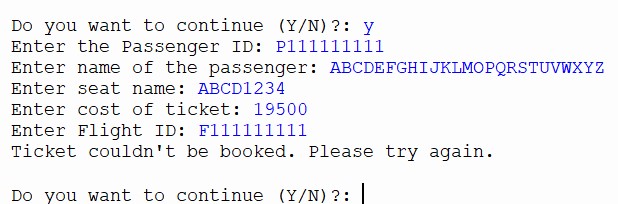


*Exception:*

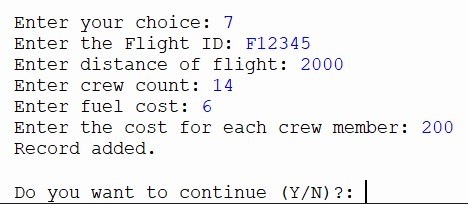
**

Booking a passenger ticket:

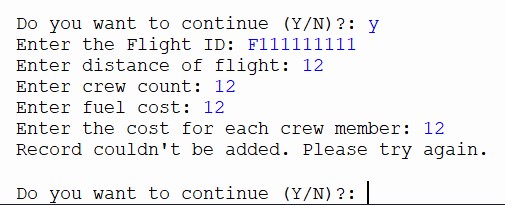
*Exception:*

**

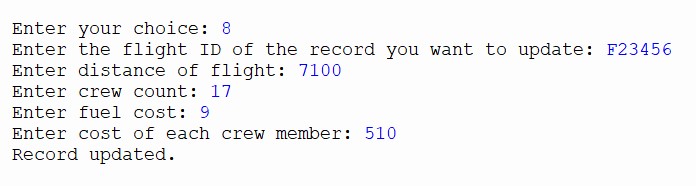
Adding financial information:



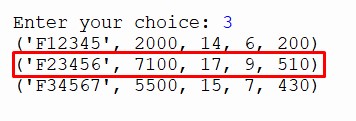
*Exception:*

**

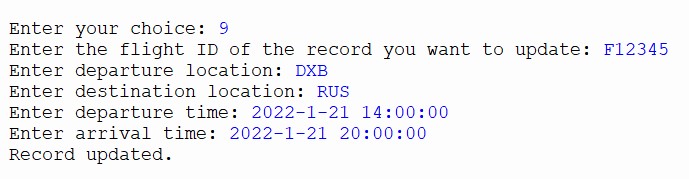
Updating financial information:



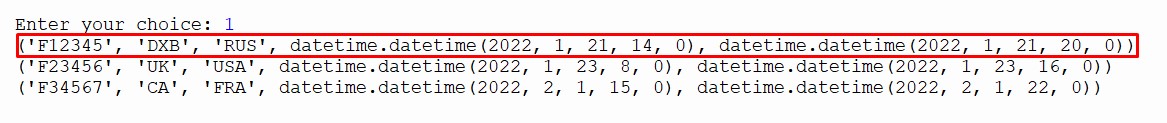
*Results:*

**

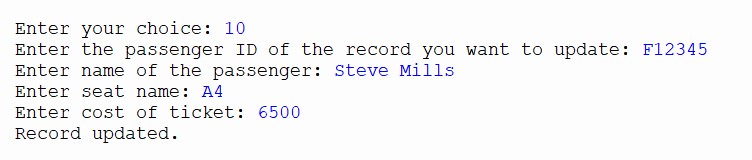
Updating route information:



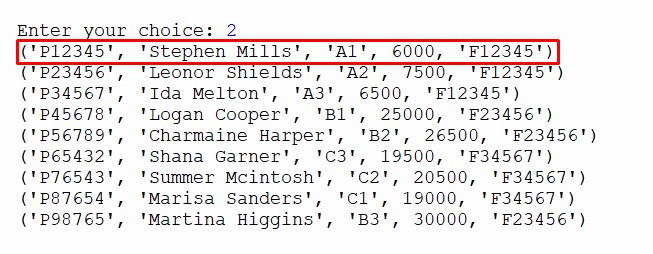
*Results:*

**

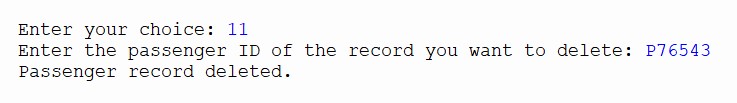
Updating passenger information:

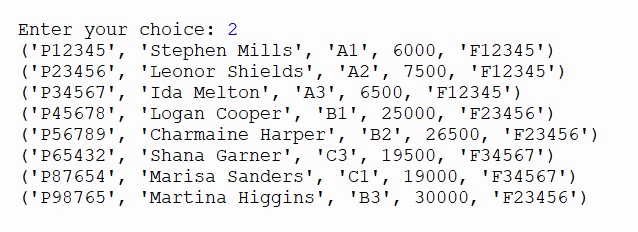


*Results:*

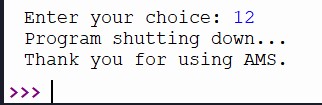
**

Removing passenger record:



*Results:*

Exiting the program:



Choice inputted not in menu:

