Expert system(airline and cargo)

class AirlineSchedulingExpertSystem:

def \_\_init\_\_(self):

self.flights = []

self.aircrafts = {}

self.crew = {}

def add\_flight(self):

flight\_id = input("Enter Flight ID: ")

aircraft = input("Enter Aircraft Model: ")

crew\_member = input("Enter Crew Member Name: ")

departure = input("Enter Departure Time (YYYY-MM-DD HH:MM): ")

arrival = input("Enter Arrival Time (YYYY-MM-DD HH:MM): ")

if self.is\_aircraft\_available(aircraft) and self.is\_crew\_available(crew\_member):

self.flights.append({

'flight\_id': flight\_id,

'aircraft': aircraft,

'crew\_member': crew\_member,

'departure': departure,

'arrival': arrival

})

self.aircrafts[aircraft] = "Assigned"

self.crew[crew\_member] = "Assigned"

print(f"Flight {flight\_id} scheduled successfully.\n")

else:

print("Error: Aircraft or Crew not available.\n")

def is\_aircraft\_available(self, aircraft):

return self.aircrafts.get(aircraft, "Available") == "Available"

def is\_crew\_available(self, crew\_member):

return self.crew.get(crew\_member, "Available") == "Available"

def show\_flights(self):

if not self.flights:

print("No flights scheduled yet.\n")

else:

print("\nScheduled Flights:")

for flight in self.flights:

print(f"Flight {flight['flight\_id']} | Aircraft: {flight['aircraft']} | Crew: {flight['crew\_member']} | Departure: {flight['departure']} | Arrival: {flight['arrival']}")

print()

class CargoSchedulingExpertSystem:

def \_\_init\_\_(self):

self.cargo\_schedule = []

def add\_cargo(self):

cargo\_id = input("Enter Cargo ID: ")

flight\_id = input("Enter Flight ID for the Cargo: ")

weight = float(input("Enter Cargo Weight (in kg): "))

priority = input("Enter Priority (High/Medium/Low): ")

self.cargo\_schedule.append({

'cargo\_id': cargo\_id,

'flight\_id': flight\_id,

'weight': weight,

'priority': priority

})

print(f"Cargo {cargo\_id} scheduled for flight {flight\_id}.\n")

def show\_cargo\_schedule(self):

if not self.cargo\_schedule:

print("No cargo scheduled yet.\n")

else:

print("\nScheduled Cargo:")

for cargo in self.cargo\_schedule:

print(f"Cargo {cargo['cargo\_id']} | Flight: {cargo['flight\_id']} | Weight: {cargo['weight']}kg | Priority: {cargo['priority']}")

print()

def main():

airline\_system = AirlineSchedulingExpertSystem()

cargo\_system = CargoSchedulingExpertSystem()

while True:

print("=== Airline and Cargo Scheduling System ===")

print("1. Add Flight")

print("2. Show Scheduled Flights")

print("3. Add Cargo")

print("4. Show Cargo Schedule")

print("5. Exit")

choice = input("Enter your choice: ")

if choice == '1':

airline\_system.add\_flight()

elif choice == '2':

airline\_system.show\_flights()

elif choice == '3':

cargo\_system.add\_cargo()

elif choice == '4':

cargo\_system.show\_cargo\_schedule()

elif choice == '5':

print("Exiting the system. Goodbye!")

break

else:

print("Invalid choice, please try again.\n")

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

main()

theory

How the System Works:

Add Flight: Users can input flight details such as Flight ID, Aircraft, Crew member, Departure time, and Arrival time. The system checks if the aircraft and crew are available and assigns them to the flight.

Show Scheduled Flights: Displays all the scheduled flights with relevant details such as the Flight ID, Aircraft, Crew member, Departure, and Arrival times.

Add Cargo: Users can input cargo details such as Cargo ID, associated Flight ID, Cargo weight, and Priority. The system will assign the cargo to the respective flight.

Show Cargo Schedule: Displays all the scheduled cargo along with the flight it is assigned to, weight, and priority.

Exit: Exit the system.

Theory:

This interactive expert system assists airline and cargo scheduling by managing flights, aircraft, crews, and cargo efficiently.

Airline Scheduling:

Ensures that flights are scheduled only if the aircraft and crew are available.

Assigns crew and aircraft to flights once availability is confirmed.

Shows scheduled flights in a clear format.

Cargo Scheduling:

Allows users to assign cargo to specific flights based on cargo weight and priority.

Displays the cargo schedule with relevant details like cargo ID, flight ID, weight, and priority.

This interactive system can be expanded further by adding features like:

Conflict Resolution: Resolving conflicts when an aircraft or crew member is already assigned to another flight.

Flight and Cargo Status: Providing status updates for flights and cargo.

Capacity Checks: Ensuring that cargo does not exceed the aircraft's weight limit.