**COEN-6312 -MODEL DRIVEN SOFTWARE ENGINEERING**

Text

Description automatically generated with low confidence

**DELIVARABLE -4**

**AIRLINE RESERVATION SYSTEM**

**PRESENTED TO**

Abdelwahab Hamou Lhadj

Professor at Concordia University

Montreal, Quebec, Canada

Winter 2022

**PRESENTED BY**

Jagadeesh Chandra Bose Viswanadhuni (40164625)

Vinishaa Kuntalo (40181334)

Bhogeswar Reddy Kalathuru (40115168)

Kamal Rohith Unnam (40166731)

Kalyan Srinivas Pandur (40169859)

Divya Pankajakshan Nair (40195179)

Table of Contents

1. PROJECT OVERVIEW
   1. DESCRIPTION OF THE SYSTEM
   2. POTENTIAL CUSTOMERS
   3. LIST OF PRIMARY FEATURES
2. DESIGN OF THE APPLICATION
   1. USE CASE DIAGRAM
   2. CLASS DIAGRAM
   3. OCL CONSTRAINTS
   4. STATE DIAGRAMS
      1. State diagram for flight
      2. State diagram for booking of a flight

**PROJECT OVERVIEW**

**1.1 DESCRIPTION OF THE SYSTEM**

Airline reservation system is an online e-ticketing application that allows the users to book tickets with an airline depending on its availability. The platform is flexible for booking both domestic and international airline tickets. The application is inclusive of ticket fares, arrival-departure schedules of the flight, airline reservations, ticket data and transaction records of the passenger. The airline reservation system allows the user to look up for a flight depending on their choice of destination, date and time of travel, and the ticket cost. The airline reservation system consists of two gateways, one is accessed by the admin and the other is accessed by the registered user of the application.

**1.2 POTENTIAL CUSTOMERS**

The airline reservation system is basically for those users who intend to book their travel tickets independently at the comfort of their home/workplace. The application is easy to understand and use. The user can simply reserve a ticket based on their desired destination, date, time, and ticket price. Since the user is registered with the system, he/she is assigned with a unique customer id which must be used at the time of login.

**1.3 LIST OF PRIMARY FEATURES**

The major attributes of the airline reservation system are that it is flexible, safe to use, protects user data and supports different users at once. This is further elucidated in the points below:

* **User Registration:** This attribute allows user registration with the airline reservation system as an administrator or a passenger. The application will then verify the credentials entered by the user.
* **Flight Registration:** This attribute is restricted to the admin only. It gives them the authority to add, cancel or modify flight details depending on the flight schedules.
* **Flight Reservation:** This attribute allows the user to look up for an available flight depending on their choice of date, time, price, and destination. It lets the user buy the ticket by providing their billing details and thereby approving the ticket purchase.
* **Traveller Itinerary:** This attribute is customised according to each user's travel history inclusive of ongoing trips, which can still be modified by the user.
* **Authorization:** This attribute gives certain authorization of the airline reservation system to the passenger and complete authorization to the admin.

# 

# **2.1 Use Case Diagram**

# Diagram Description automatically generated

# **2.2 Class diagram:**

Diagram

Description automatically generated

Figure : Class Diagram

The above Airlines Reservation System Class Diagram represents the structure, attributes, methods, and their relationship between objects for reserving a ticket to the passenger. The main classes in this application are Person, Passenger, Employee, Airline, Reservation, Schedule, Flight.

Here Person class is the main class where it has 4 attributes – Name, Email, Mobile, Dob. The person class is inherited by the Passenger Class and Employee Class, as these two classes are inherited by the Person class, they have the same attributes and Employee class has an extra attribute for determination of their role. Employee class further associated with the Airline class where it has an attributes – Name and an operation – addEmployee(). In this association the airline has many employees and employee works for one airline company. Airline class is further associated with Flight class which has 5 attributes – From, To, Departtime, Arrivetime, Numofseats and two operations – getAirline(), getCrew(). Here flight class belongs to one Airline and Airline class is built in a way that it has many flights. The next one is Schedule Class where it has 5 attributes – departDate, arriveDate, expectedDepartTime, fullBooked, delay and has an operation – getFlight(). Here flight and schedule were built in a way in this application that one flight has many schedules and each schedule is designed for one flight. Coming to the last class in the application, Reservation has two attributes –Seat number, Invoice Num and has three operations – getPassenger(), addReservation(), modifyReservation() here all the attributes are related to the booking, This class has two associations one with Schedule class and Passenger class where once schedule can has zero or one reservation and one reservation has one schedule and coming to the reservation and passenger association, a passenger can make zero or many reservation and one reservation can have many passengers.

**2.3 OCL Constraints**

1.The age of the passenger should be more than 18 to signup the application.

**Context:** Passenger

**inv:** allInstances()->forAll(c: Passenger, (c.Reservation->notEmpty() implies year(c.Person.dob) <=year(today) – 18))

2.Reservation is done with the requested seat by checking the availability of that seat on the flight.

**Context:** Flight

**Pre:** self.seatnumber includes(seatnumber)

**Post:** slf.seatnumber exclude(seatnumber)

3. Cancelling a reservation after checking if there is an existing reservation.

**Context :** Passenger

**Pre:**self.PassengerReservation includes(Passenger n)

**Post:**self.passengerReservation excludes(Passenger n)

4. Removing the flight after verifying whether the flight object exists.

**Context:** Airline

**inv:** self. Flight→includes(a)

self. Flight→excludes(a)

5. Should check whether the allocated crew members are zero for the flight

then only the crew members should be added.

**Context:** Airline

**inv:** self. Flight. getCrew.size() =0 ➔self.Flight.setCrew

6. Crew members should not be greater than 5

**Context:** Flight

**Inv:** self. Flight. getCrew.size()<=5

7. Every flight should have an unique number.

**Context:** flight

**Inv:** allInstances->forAll (f1, f2 flight | f1<>f2 implies f1.flightID <> f2.flightID)

8. Regardless of flight numbers, all Schedule ids must be unique.

Context: Schedule

**inv:** allInstances->forAll (k1, k2 Schedule | k1<>k2 implies

scheduleID<>k2.scheduleID)

9. A passenger cannot have two separate bookings on the same flying schedule.

**Context:** Passenger

**inv:**self.Reservation->forAll(p1,p2 Reservation | p1<>p2 implies

p1.Schedule.scheduleID<>p2.Schedule.scheduleID)

10. It is not possible for two distinct people to reserve the same seats on the

same aircraft with the same timetable.

Context: Flight

**inv:** self.Schedule.Reservation-> forAll(p1,p2 Reservation | p1<>p2 implies

p1.seatNumbers<>p2.seatNumbers)

11. The number of reserved tickets cannot exceed the number of seats

available for reservation for a certain flight schedule.

**Context:** Schedule

**inv:** self.Reservation->collect(seatNumbers) -> count()=<self.Flight.numberOfSeats

12.Passengers can only make changes to bookings that are assigned to them.

**Context:** Reservation: modification (Passenger n)

**pre:** self.Passenger->includes(n)

**post:** self.Passenger->includes(n

* 1. **STATE DIAGRAMS:**

State diagram depicts the behaviour of an object of a class in response to any stimuli such as a state or an event. Following some of the state diagrams shows the behaviour of the objects of the classes created for the Flight reservation system.

**2.4.1 STATE DIAGRAM FOR FLIGHT**

1. A Traveller when he books a flight. A flight object is created in flight schedule class.
2. When it departs, he can track the flight in Tracking class, and it redirects to transit class at present location it will shows you. If it is not landed it redirects back to in transit.
3. If the flight reaches the destination, it shows the landed and the flight object is deleted.
4. If traveller wants to cancel the flight it moves to cancel flight class, there you can request to cancel the flight.

Diagram

Description automatically generated

Fig 1: State diagram for flight

* + 1. **STATE DIAGRAM FOR BOOKING FLIGHT**

1. When a traveller wants to book a flight a booking object is created in booking class.
2. If he wants to change any details like name, phone number, age, email etc..., he can request for change so a modifying booking object will be created.
3. If they don’t want to change any details, they can cancel so that a booking object will be deleted or if traveller wants to delete his booking details, he can delete in cancel class.

# **Diagram Description automatically generated**

Fig 2: State diagram for booking flight