Model Driven Software Engineering

(COEN 6312)

Project Deliverable 3

Submitted to

Dr. AbdelwahabHamou-Lhadj

Submitted by

Nareshkumar M. Sisodiya 27650817

Arjun Lokhande 27411111

Binu Basil John 27421753

Anant Mathur 27323670

Khushboo Handa 27323794

March 4, 2016

Contents

List of Figures ii
1. Introduction 1
2. Class Diagram Description 1
  2.1 Person 2
    2.1.1 User 3
    2.1.2 Administrator 3
  2.2 Online Air Reservation System 4
  2.3 Ticket 4
  2.4 Catalog 5
  2.5 Flight 6
  2.6 Payment 7
3. OCL Constraints 8

List of Figures

Figure 1: Class Diagram - System 1
Figure 2: Class Diagram - Person 2
Figure 3: Class Diagram - User 3
Figure 4: Class Diagram - Administrator 3
Figure 5: Class Diagram - Online Air Reservation System 4
Figure 6: Class Diagram - Ticket 4
Figure 7: Class Diagram - Catalog 5
Figure 8: Class Diagram - Flight 6
Figure 9: Class Diagram - Payment 7

1. Introduction

For any given software system to function efficiently, it is essential to visualize, design, construct and document it through a standardized procedure. UML or Unified Modelling Language achieves these goals by providing a graphical representation of the software system.

FlyAir online reservation system has taken the initial steps of its formation through its standardized procedure in the previous deliverable, by defining the functional requirements of our system using use case diagrams. It prioritises the simplicity of usage for the clients throughout the process of search and reservation of tickets. This is highly reflected in the entire system as well as its construction from its core. The main objective of this deliverable is to explore the detailed class diagram of the system with the classes, attributes and functions associated and then list out the constraints that is applied to the same, using OCL.

The Class diagram is created after an analysis of the system outlining the classes in the system and the relationship between them, their attributes and their operations. The OCL constraints helped us to list out the restrictions and conditions that should be implemented in the system. Thus, the result is keeping all the possible situations in the flow of operation within the boundaries of the system design.

2. Class Diagram Description

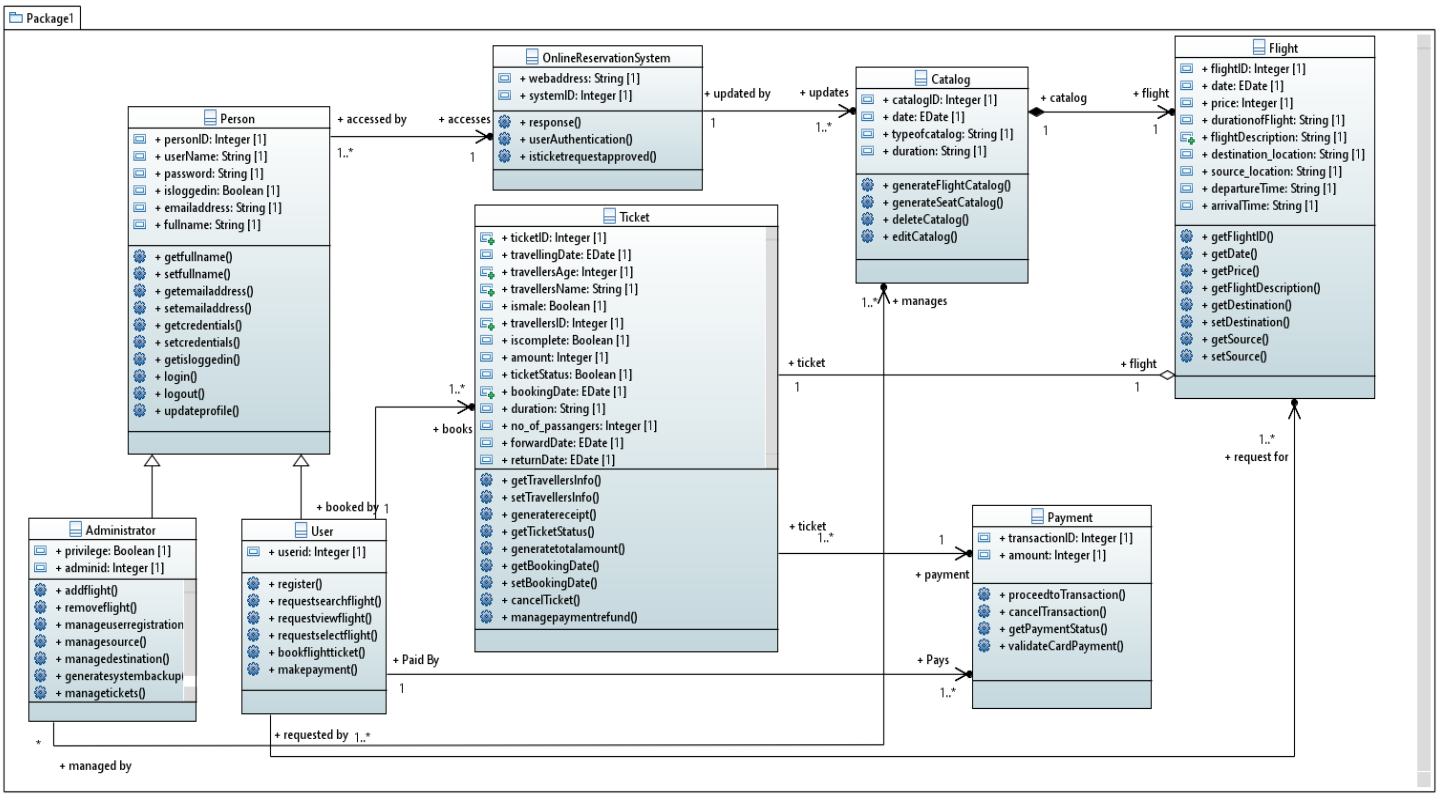

Figure 1: Class Diagram - System

2.1 Person

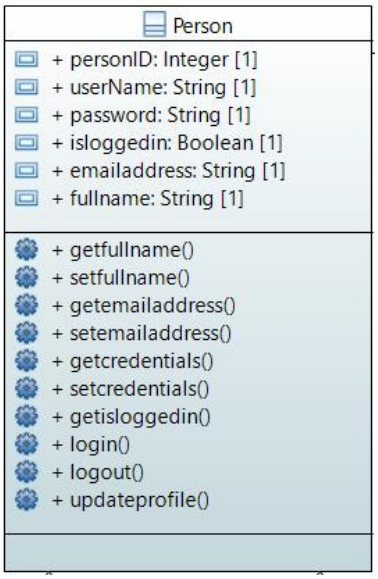

Figure 2: Class Diagram - Person

Person is the superclass which is inherited by the administrator class and user class. Both Customers and Administrator are the primary actors of the system and they inherit the common attributes from the Person Class.

The attributes and methods of the class Person gives it versatility for different scenarios of the system flow.

- The attributes personID, userName and password are the credential information that ensure security and uniqueness of the primary actors. The methods getcredentials() and setcredentials() are used to acquire the credentials for individual users and administrator.
- Attribute isloggedin specifies the login status of the actor in the system and method getisloggedin() provide the corresponding information of the status.
- The methods login() and logout() allows the person to login or logout of the personal account.
- The attributes emailaddress and fullname are required for the personal information for the user account. The methods getfullname() and setfullname() corresponds to the attribute fullname and methods getemailaddress() and setemailaddress() corresponds to the attribute emailaddress.
- Method updateprofile() allows the person to update the information provided in the personal account.

2.1.1 User

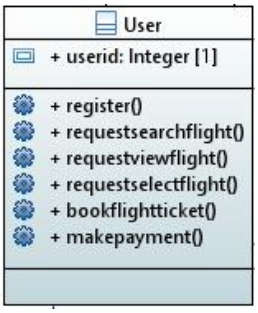

Figure 3: Class Diagram - User

User Class inherits from class Person. User of the system relates to the customer of FlyAir. Since User is a Primary actor of the system this class plays a vital role. All the attributes and methods of User class takes care of the functions performed by the customer of FlyAir.

- The attribute userid gives a unique identification number for the user.
- The method register() allows the user to register into the system to create personal account to proceed with the ticket reservation procedure.
- The methods requestsearchflight(), requestviewflight(), requestselectflight() allows to browse through the list of flights available to meet the user's specific journey requirements, without registering into the system.
- Method bookflightticket() lets the user to book the ticket of particular choice, however it requires the user to register first.
- Method makepayment() directs the user to make payment for the booked tickets.

2.1.2 Administrator

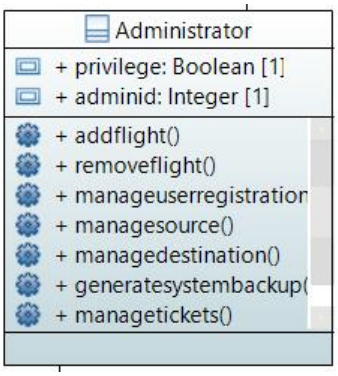

Figure 4: Class Diagram - Administrator

Class Administrator defines the role of an administrator in the system. This class inherits from the Person class. The primary task of Administrator class is to perform the administrative functions in a flight reservation system.

- The attribute adminid provides a unique identification to the administrator.

- The attribute privilege specifies Administrators' privilege in the system.
- The administrator class will use the operations managetickets(), managesource() and managedestination() to manage the flight operation.
- The methods manageuserregistration() and generatesystembackup() is used to perform user registrations and routinely take system back-up. Admin class can manage overall system.
- The addflight() and removeflight() methods are used for addition or removal of flights.

2.2 Online Air Reservation System

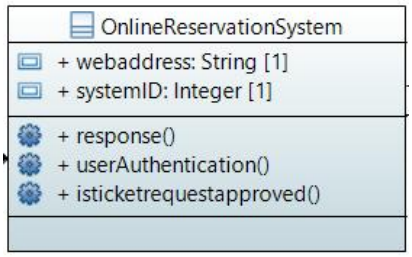

Figure 5: Class Diagram - Online Air Reservation System

OnlineReservationSystem class plays a vital role in responding to the request initiated by the person.

- The attribute webaddress provide the URL for the Flyair website.
- Method response() provides feedback for each and every request made by person in the system.
- User authentication and ticket approval status is provided by methods userAuthentication() and isticketrequestapproved() respectively.

2.3 Ticket

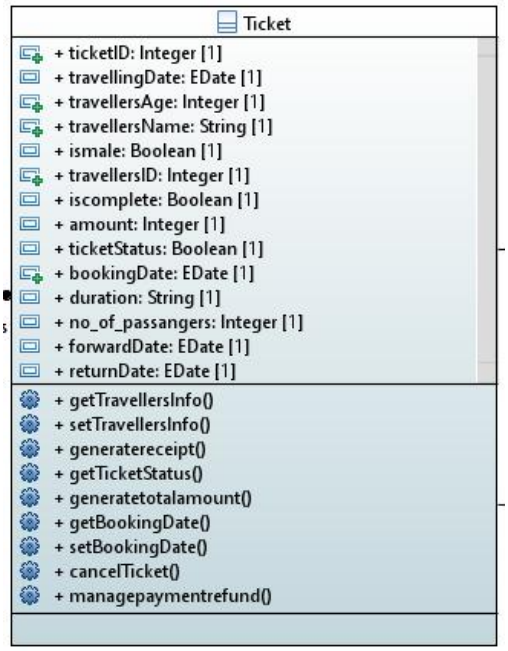

Figure 6: Class Diagram - Ticket

Ticket class has an association relationship between User class and Payment Class. All the attributes and methods of Ticket class take care of travellers information and ticket status.

- The attributes TicketID and TravellersID give unique Ticket reference number and Identity Card number respectively.
- The attributes travellingDate and duration gives the date and duration of journey.
- The attribute bookingDate gives the date of on which the given ticket is booked. One can get booking date by using method getBookingDate(). By using setBookingDate() system sets the Booking date for that particular ticket after processing the payment (with reference to Calendar).
- The attributes TravellersAge, TravellersName and ismale give traveller's age, name and sex respectively. The user can provide Traveller's information using the method setTravellerinfo() and Admin can get the Travellers information by using the method getTravellerInfo().
- The attribute amount gives the fare charge of the journey. The method generatetotalamount() is used to prompt the total fare. The method generatereceipt() is used for generating receipt. For cancellation of ticket Cancel() method can be used.
- The attribute iscomplete gives acknowledgement information that the traveller's information is completely filled out or not.
- The attribute TicketStatus gives information of booking status of the ticket. By using method getTicketStatus() user can get the booking status of his journey in case if he did not get confirmation for his transaction.
- The method managepaymentrefund() is used for refunding of percentage of ticket amount in case of cancellation of ticket.

2.4 Catalog

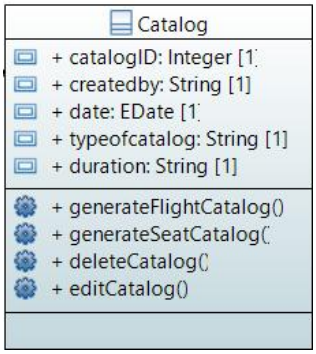

Figure 7: Class Diagram - Catalog

Catalog class holds all required data for flights and tickets. Upon receiving different combinations of requests, it can serve list of flight or list of available seats:

- The catalog Id is unique to identify the catalog being requested.
- The attributes -createdby, typeofcatalog and date gives the information about the catalog
- The methods -generateFlightcatalog(), generalSeatCatalog(), deleteCatalog(), editCatalog() can be used for modifying the catalog.

2.5 Flight

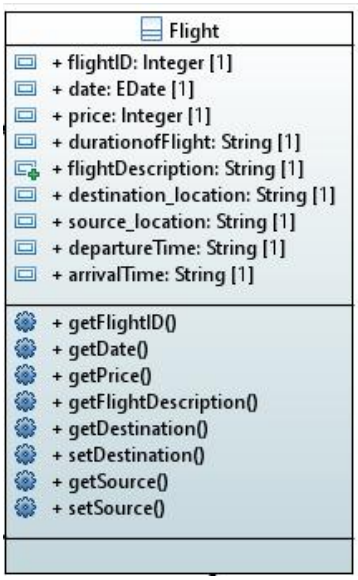

Figure 8: Class Diagram - Flight

The flight class comprises the information related to the flight. This class has a Composition relation with the class Catalog, an association with the class User & it is aggregation of class Ticket. All the attributes and methods of Flight class takes care of the information of the flights of company Flyair.

- The attribute flightid gives a unique id number for the flight. The method getFlightID() is used to get the unique flight number, the available dates and price for that flight can be accessed by using the methods getDate() and getPrice() respectively.
- The attribute durationofFlight gives the estimated time that flight will take to reach at the destination
- FlightDescription attribute gives the information about the type of the aircraft used.The information about the type of aircraft used for the selected journey can be accessed by using flightDescription().
- The attributes Destination_location gives name of the City(Destination) and Source_location gives the name of City(source). The given source and destination location can be accessed by using the methods getSource() and getdestination() respectively. By using methods setDestination() and setSource(), Admin can set the destination and source location respectively.

2.6 Payment

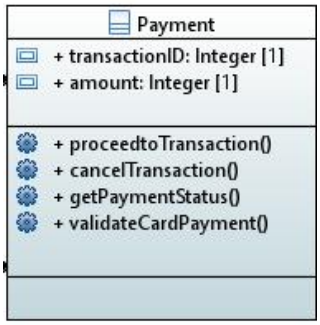

Figure 9: Class Diagram - Payment

The Payment Class is used to carry out the transactions of fare charges. This class has association relationship between class Ticket and class User. All the attributes and methods of Payment class take care of transactions performed by the customer of FlyAir.

- The attribute amount gives the fare charge for the flight which customer is going to book.
- Once the customer fills the Card information,he can then proceed to payment by using method proceedtoTransaction() and attribute TransactionID gives unique transaction number.
- The payment can be cancelled by using the method cancelTransaction().
- One can get the payment status by using the method getPaymentStatus().
- The system can validate the payment using the method ValidateCardPayment().

3. OCL Constraints

- Each person who logs in to the system to book a ticket must have an unique ID :
  Context : Person
  inv: self.allinstances()->forall(P1,P2:Person | P1 <> P2 implies P1.userID <> P2.userID);

- Each Ticket booked by the user for scheduled travel should have an unique ID :
  Context : Ticket
  inv: self.allinstances()->forall(T1,T2:Tickets | T1<>T2 implies T1.ticketID <> T2.ticketID);

- A logged in user planning a travel can book at most 10 tickets of the flight :
  Context : User
  inv: self.book->no_of_passengers<= 10;

- Source and Destination location of the flight cannot be the same :
  Context : Flight
  inv: self.destination_location<>self.source_location;

- User must provide Traveller's information while booking ticket :
  Context : Ticket
  inv: self.travellersName->notEmpty() AND self.travellersAge->notEmpty() AND self.travellersID->notEmpty();

- The return journey date must be after the forward journey date :
  Context : Ticket
  inv: self.forwardDate<self.returnDate;