

Model Driven Software Engineering

(COEN 6312)

Project Deliverable 3

Submitted to

Dr. Abdelwahab Hamou-Lhadj

Submitted by

Nareshkumar M. Sisodiya 27650817

Arjun Lokhande 27411111

Binu Basil John 27421753

Anant Mathur 27323670

Khushboo Handa 27323794

February 29, 2016

Contents

[List of Figures 3](#_Toc444472269)

[1. Introduction 4](#_Toc444472270)

[2. Class Diagram Description 5](#_Toc444472271)

[2.1 Person 5](#_Toc444472272)

[2.1.1 User 6](#_Toc444472273)

[2.1.2 Administrator 6](#_Toc444472274)

[2.2 Online Air Reservation System 6](#_Toc444472275)

[2.3 Ticket 7](#_Toc444472276)

[2.4 Catalog 7](#_Toc444472277)

[2.5 Flight 8](#_Toc444472278)

[2.6 Payment 8](#_Toc444472279)

[3. OCL Constraints 9](#_Toc444472280)

# List of Figures

# 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 Modeling Language achieves these goals by providing a graphical representation of the software system.

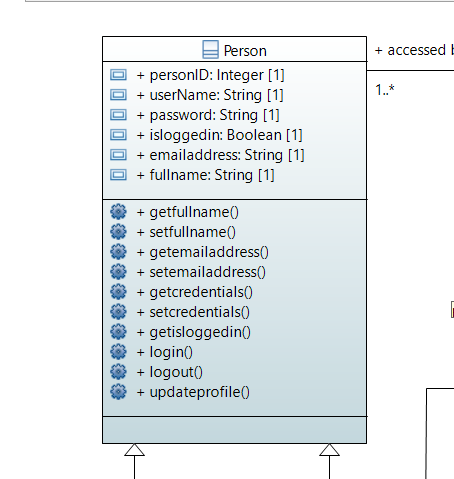
Flyair online reservation system has taken the initial steps of its formation through its standardized procedure in the previous deliverable, by defining the functional requirements of our system using Use Case Diagrams that provide interactions between the use cases and the actors of the system. It 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 list out the constraints that is applied to the same, using OCL.

The Class diagram is created after a close analysis of the system from the point of view of a customer and an administrator by outlining the possibilities and requirements during the entire flow of the system operation. The OCL constraints helped, to go further into the finer details of the system operation listing out the restrictions and conditions that should be implemented to get the smooth and accurate flow of the system. This required the vigorous effort and immense imagination of the entire team. Thus, the result is keeping all the possible situations in the flow of operation within the boundaries of the system design.

The OCL constraints help to go further into the finer details of the system operation listing out the restrictions and conditions that should be implemented to get the smooth and accurate flow of the system. Thus, the result is keeping all the possible situations in the flow of operation within the boundaries of the system design

# FlyAir (2).png2. Class Diagram Description

## 2.1 Person

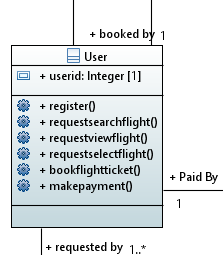


Person is the super class which is inherited by the administrator class and user class. Both Customers and Administrator are the primary actors of the system and they inherit the common attributes from the Person Class.

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 log in 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.

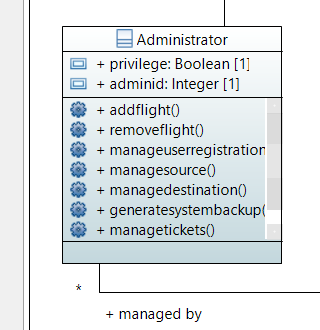
### 2.1.1 User



User Class is the inherited from class Person. User of the system relates to the customer of Flyair. Since Customer 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.

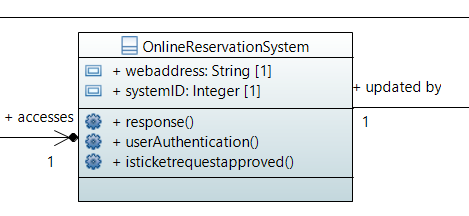
* The attribute userid gives a unique id number for the user.
* The method register() allows the user to register into the system to create personal account to proceed with the ticket reservation procedure.
* The methods requestsearchflight(), requestviewflight(), requestselectflight() allows to browse through the list of flights available to figure out if it meets the user’s specific journey requirements, without registering into the system.
* Method bookflightticket() let the user to book the ticket of particular choice, however it requires the user to register first.
* Method makepayement() directs the user to make payment for the tickets booked.

### 2.1.2 Administrator



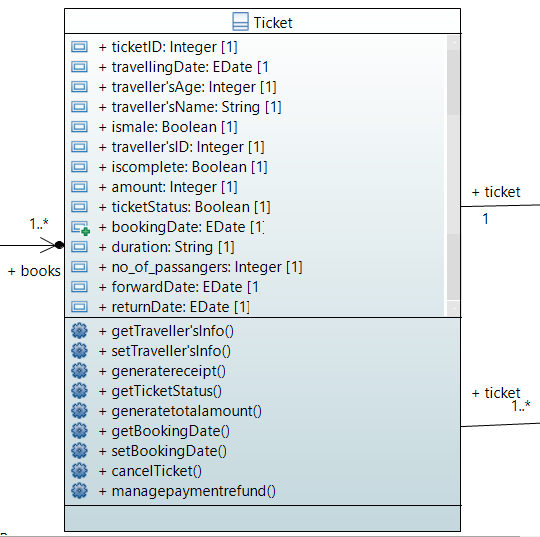
Admin class will define the role of an administrator in the system. This class will inherit the Person class. The primary task of Admin class is perform administrative functions in flight reservation system. Administrator will manage flight details, user registrations and routinely take system back-up. Admin class can manage overall system.

## 2.2 Online Air Reservation System



OnlineReservationSystem class plays a vital role in responding to the request initiated by the person. Also User authentication as well as approval of ticket request will be handled by this class.

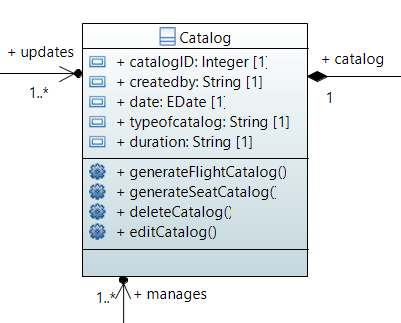
## 2.3 Ticket



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

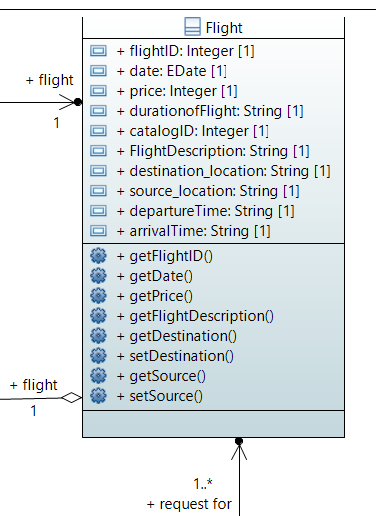
* The attribute TicketID gives unique Ticket reference number.
* The attribute Traveller’sID gives unique Identity Card number of traveller.
* The attribute Date gives the date of journey.
* The attributes Traveller’sAge, Traveller’sName and ismale gives travellers age, name and type of sex respectively.
* The attribute duration gives the estimated time that flight will take to reach at the destination
* The attribute amount gives the fare charge (including TAX) the journey.
* The attribute booking date gives the date of on which the given ticket is booked.
* The attribute iscomplete gives acknowledgement info that the travellers info is completely filled or not.
* The attribute Ticket status gives information of booking status of the ticket.
* The Customer can provide Travellers information using method setTraveller’s info() and Admin can get the Travellers info by using method setTraveller’sInfo().
* 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 Calender).
* The method generatetotalamount() is used to promt the total fare including TAX.
* The method generatereceipt() is used for generating receipt.
* For cancellation of ticket Cancel() method can be used.
* The method managepaymentrefund() is used for refunding of percentage of ticket amount in case of cancellation of ticket.
* By using method getBookingStatus() user can get the booking status of his journey in case if he didn’t get confirmation for his transaction.

## 2.4 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.

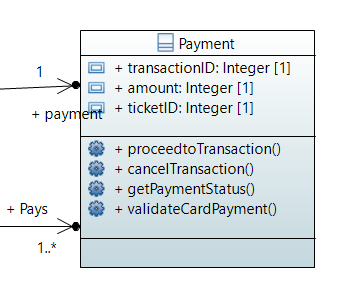
## 2.5 Flight



The flight class comprises the information related to the flight. This class has deeper relation (Composition) with the class Catalog, has 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 attribute Date gives the dates for the selected flight.
* The attribute Price gives the ticket fare
* The attribute durationofFlight gives the estimated time that flight will take to reach at the destination
* The attribute CatalogID gives the unique number which points to the intended Catalog for that flight
* FlightDescription gives the information about the type of the aircraft used
* Destination\_location gives name of the City(Destination)
* Source\_location gives the name of City(source)
* 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 information about the type of aircraft used for the selected journey can be accessed by using flightDescription().
* By using methods setDestination() and setSource() Admin can set the destination and source location respectively.
* The given source and destination location can be accessed by using the methods getSource() and getdestination() respectively.

## 2.6 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 takes care of transactions performed by the customer of Flyair.

* The attributes TransactionID and TicketID gives unique transaction number and reference Ticket number respectively.
* The attribute amount gives the fare charge (including TAX) for the flight which customer is going to book.
* Once the customer filled the Card information then he can proceed for payment by using method proceedto Transaction() and can cancel the payment by using the method cancel Transaction().
* One can get the payment status by using the method getPaymentStatus().
* The system can able to 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

int : 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;