Deliverable 3

Flight Reservation System

COEN 6312 – Model Driven Software Engineering

Team Members

Ankit Chitlangia	**27485573**
Armaan Gill	**27380003**
Gurpreet Rehal	**27632215**
Mitesh Kaura	**27284756**
Simranjeet Singh	**27376782**

Table of Contents

Class Diagram description of Flight Reservation System

Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The class diagram is static in nature as it represents the static view of an application. Class diagram is not only used for visualizing, describing and booking different aspects of a system but also for constructing executable code of the software application. It shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

In our class diagram we have taken ten classes for each class we have defined relevant functions and attributes pertaining to that particular class.

Registered Users:
This class gives the details about the registered users along with the data type such as their name, gender, address. Etc.

Flight:
This class contains all the information about the class along with their data types such as flight schedule, number of passengers, capacity. Etc.

Administrator:
This is the Admin class which controls all the information for all the classes and thus has to be synchronized accordingly during the development phase. As far as the class is concerned it allows the admin to update the information and manage the profiles.

Book flight:
This class allows the users to book a flight by managing information such as booking id, meal preference, booking date. Etc.

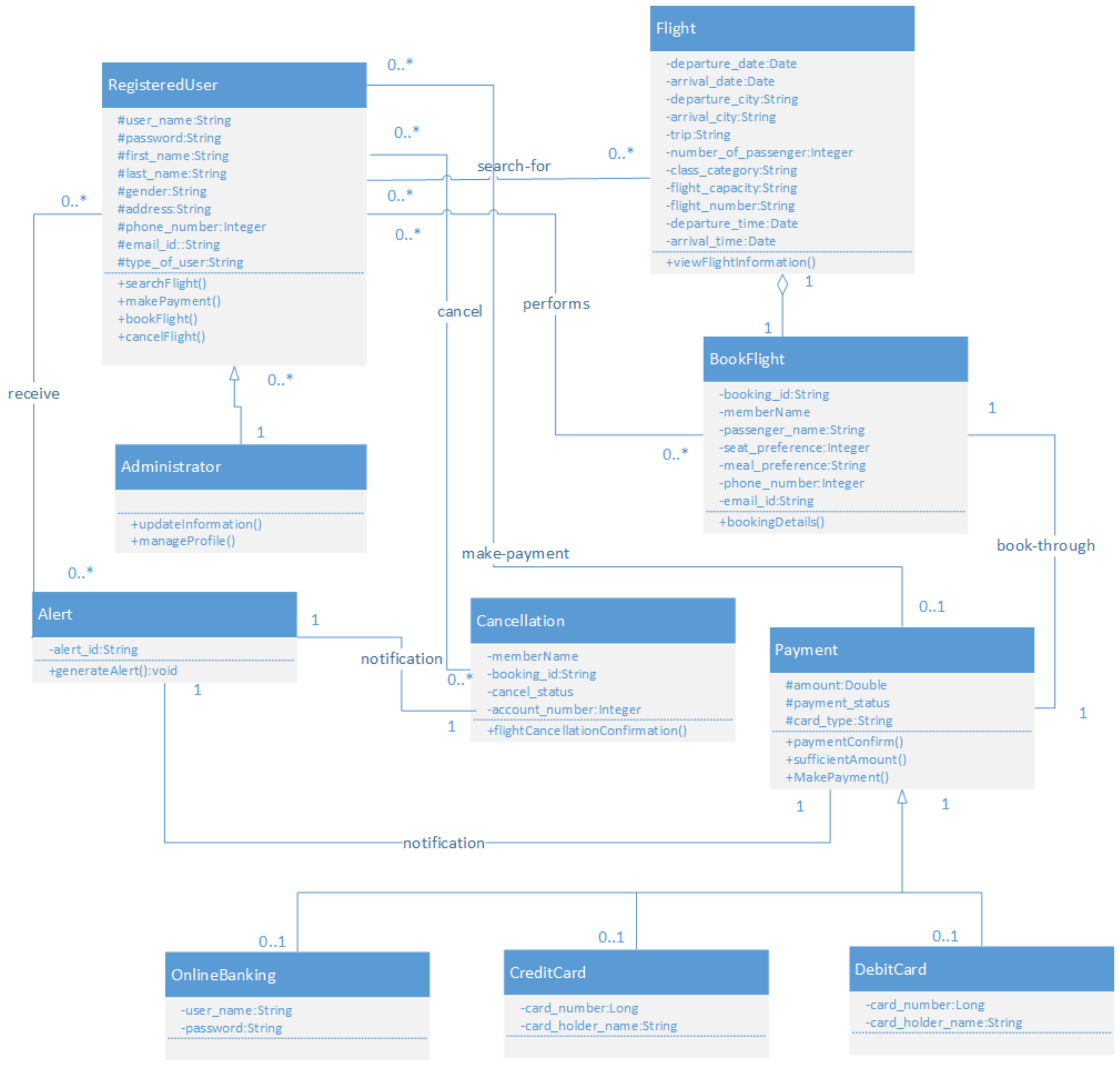
Alert:
This class sends the flight alerts.

Cancellation:
With attributes such as cancel status and account number it allows the users to cancel a booked flight.

Payment: This is the parent class for its sub classes online banking, credit card and debit card. It allows the user to choose his/ her payment mode to book a flight online.

UML Class diagram of Flight Reservation System is:

Class Diagram



Constraints

RegisteredUser

Phone number of Registered User should be of 10 digits.

Context RegisteredUser
inv: self.phone_number.size()=10

Registered User password should be alphanumeric, contain Special character and size should not be less than 8 or greater than 15.

Context RegisteredUser
inv: password = password.matches((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).{8,15})

No two flights can have same flight number.

Context RegisteredUser
inv: self.search-for->ForAll (F1,F2=Flight|F1<>F2
ImpliesF1.flight_number<>F2.flight_number)

The RegisteredUser can only cancel the flight if there exist booking id respective to there name.

Context RegisteredUser
inv: self.cancel->cancelFlight() implies self.book-for.booking_id->nonEmpty()

Administrator

Only the Administrator can perform update operation.

Context Administrator::updateInformation()
pre:self.type_of_user='Administrator'

Only the Administrator can perform Manage operation.

Context Administrator::manageProfile()
pre:self.type_of_user='Administrator'

Flight

The departure date of flight should be before its arrival date.

Context Flight
inv: self.departure_date<self.arrival_date

The departure city of flight cannot be same as its arrival city

Context Flight
inv: self.departure_city<>self.arrival_city

BookFlight

The booking id issued to one RegisteredUser should not match to the other RegisteredUser.

```
Context BookFlight
inv: self.AllInstances->ForAll (B1,B2=BookFlight|B1<>B2
                                    Implies B1.booking_id<>B2.booking_id)
```

The booking id should generates once the payment is made.

Context BookFlight
inv: self.booking_id->nonempty implies self.book-through->paymentConfirm()

Payment

For payment to be valid there should be sufficient balance in account.

Context Payment::paymentConfirm():
Pre: self.amount->sufficientAmount()

The notification message will be issued on payment and cancel status.

Alert

```
Context Alerts::generateAlert()
Pre:self.notification.cancel_status->nonEmpty()
Or self.notification.payment_status->nonEmpty()
```

References

[1] https://www.eclipse.org/papyrus/download.html

[2] http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733

[3] Craig Larman Applying UML and Patterns Practice Hall, 2005