FLIGHT MANAGEMENT SYSTEM

Capgemini

PREPARED FOR

JEE Cloud Training Program

Capgemini Technology Services

PREPARED BY

Mohammed Thouseef

Vikas Vishwakarma

Rohan Kachare

Abstract:

The Flight Management System is a Microservice based booking solution for flight tickets. It consolidates data provided by different airline carriers and hence provides the user details and rates in real-time. Travelers may want to make changes in their bookings. The application allows them to book, cancel, view and update their bookings with ease. Other than this, it eases the management of bookings too. All the flights, schedules and routes can be viewed, added and modified on a single application by the administrator.

Scopes:

Inscope:
Following is the functionality provided by the system:

There are two categories of people who would access the system: customer and administrator. Each of these would have some exclusive privileges.
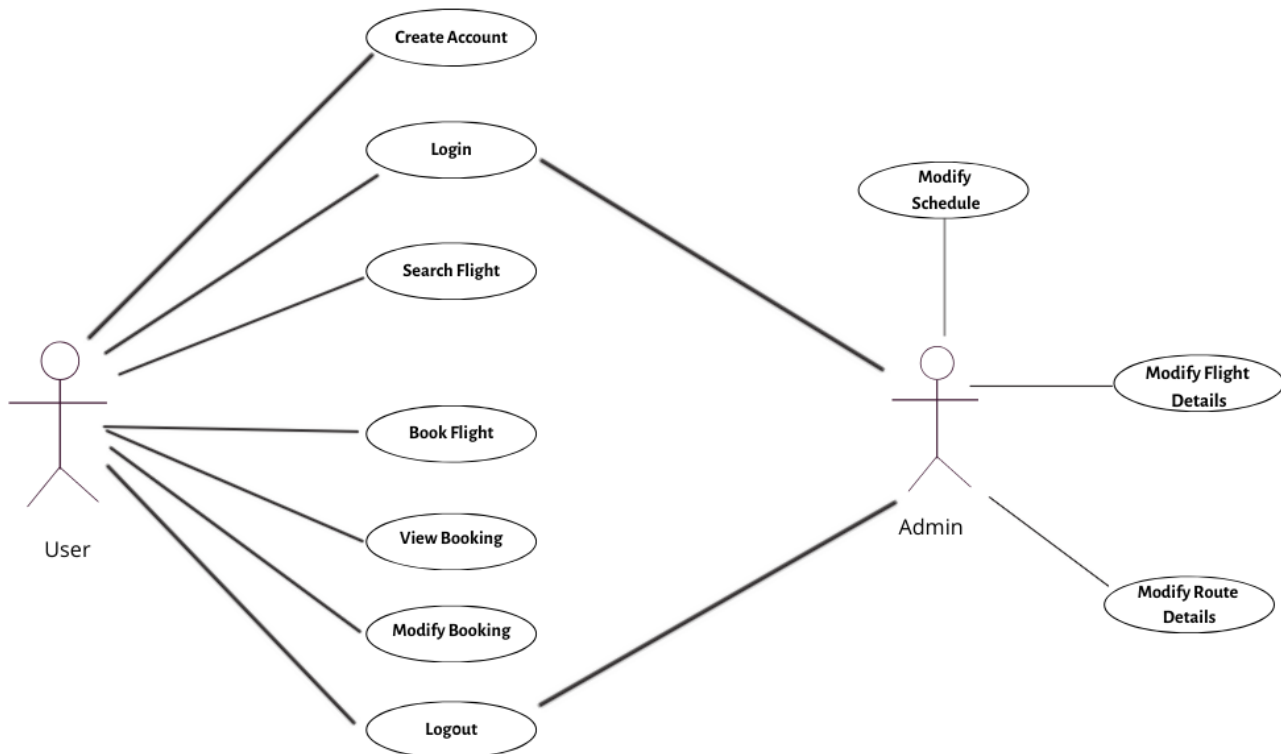
1. The customer can:
    a. Create his user account.
    b. Login into the application.
    c. Check for available flights.
    d. Make a booking.
    e. View the bookings made.
    f. Cancel or modify a booking.

2. The administrator can:
    a. Login into the application.
    b. Add flight, schedule and route details.
    c. View the flight, schedule and route details.
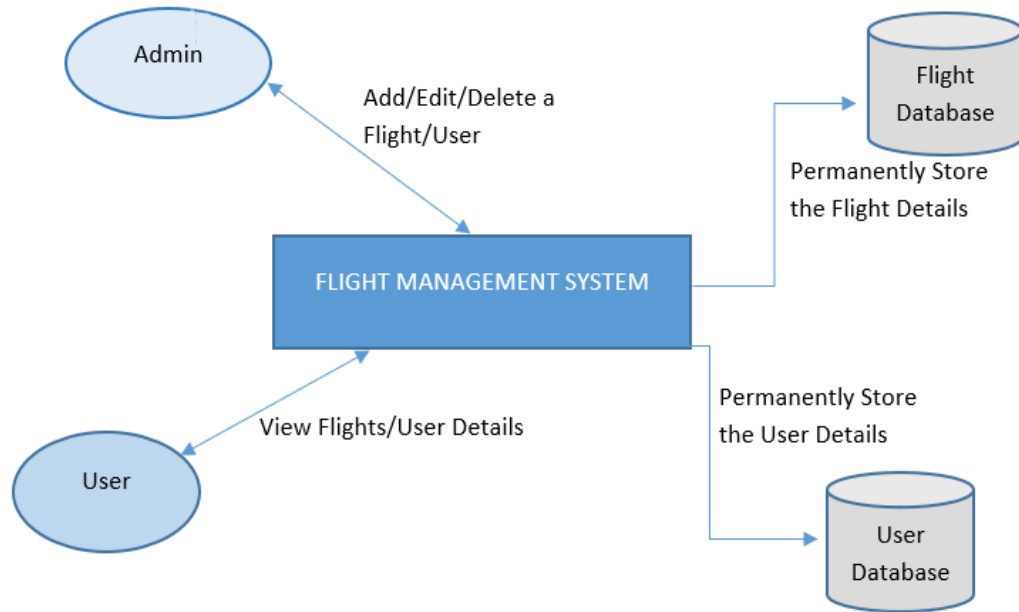   d. Cancel or modify the flight, schedule and route details.

Outscope:
The following functionalities have not been covered under the application:

1. The application does not cover boarding pass generation and seating plans.
2. Third party applications like email & sms integrations.
3. Payments are not yet accepted by the application.

Use Case Diagram:
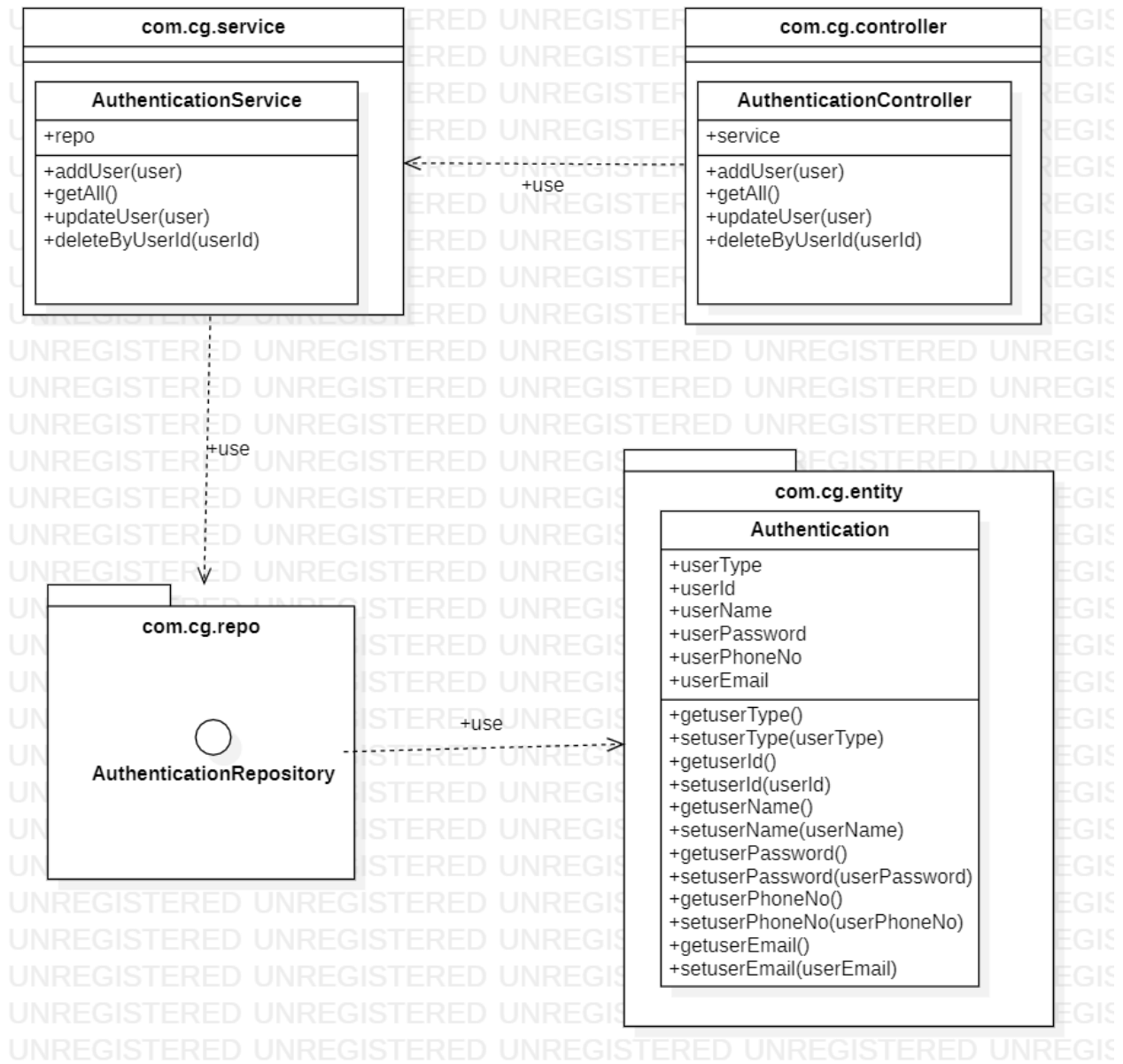
Flight Management Data Flow Diagram

Admin

Add/Edit/Delete a
Flight/User

FLIGHT MANAGEMENT SYSTEM

Flight
Database

Permanently Store
the Flight Details

View Flights/User Details

Permanently Store
the User Details

User

User
Database

Authentication Service:

Class Diagram:

com.cg.service

AuthenticationService

+repo

+addUser(user)
+getAll()
+updateUser(user)
+deleteByUserId(userId)

com.cg.controller

AuthenticationController

+service

+addUser(user)
+getAll()
+updateUser(user)
+deleteByUserId(userId)

+use

+use

com.cg.repo

○

AuthenticationRepository

+use

com.cg.entity

Authentication

+userType
+userId
+userName
+userPassword
+userPhoneNo
+userEmail

+getuserType()
+setuserType(userType)
+getuserId()
+setuserId(userId)
+getuserName()
+setuserName(userName)
+getuserPassword()
+setuserPassword(userPassword)
+getuserPhoneNo()
+setuserPhoneNo(userPhoneNo)
+getuserEmail()
+setuserEmail(userEmail)

Epic Stories:

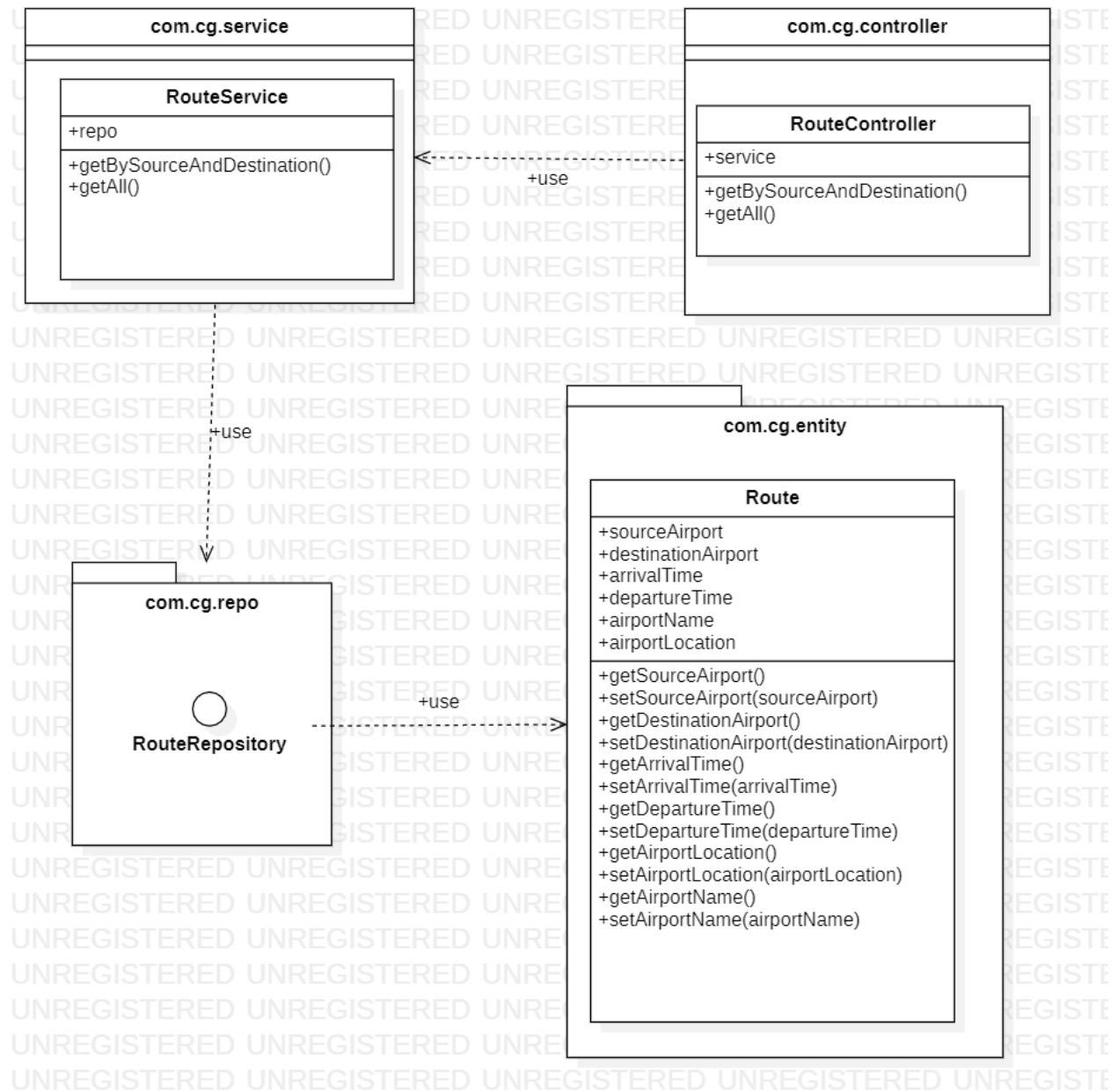| Epic	Stories	As a/an	I want to	So that...
Authentication Service	Add User/Admin	User/Admin	Successfully register a User/Admin	A new User/admin will be
	Update User/Admin	User/Admin	Update the details of the User/Admin	User/Admin details can be Updated
	Delete User	Admin	Delete a User	A user can be deleted
	View all users	Admin	Search all users	Admin can find all the users

Booking Service:

Epic Stories:

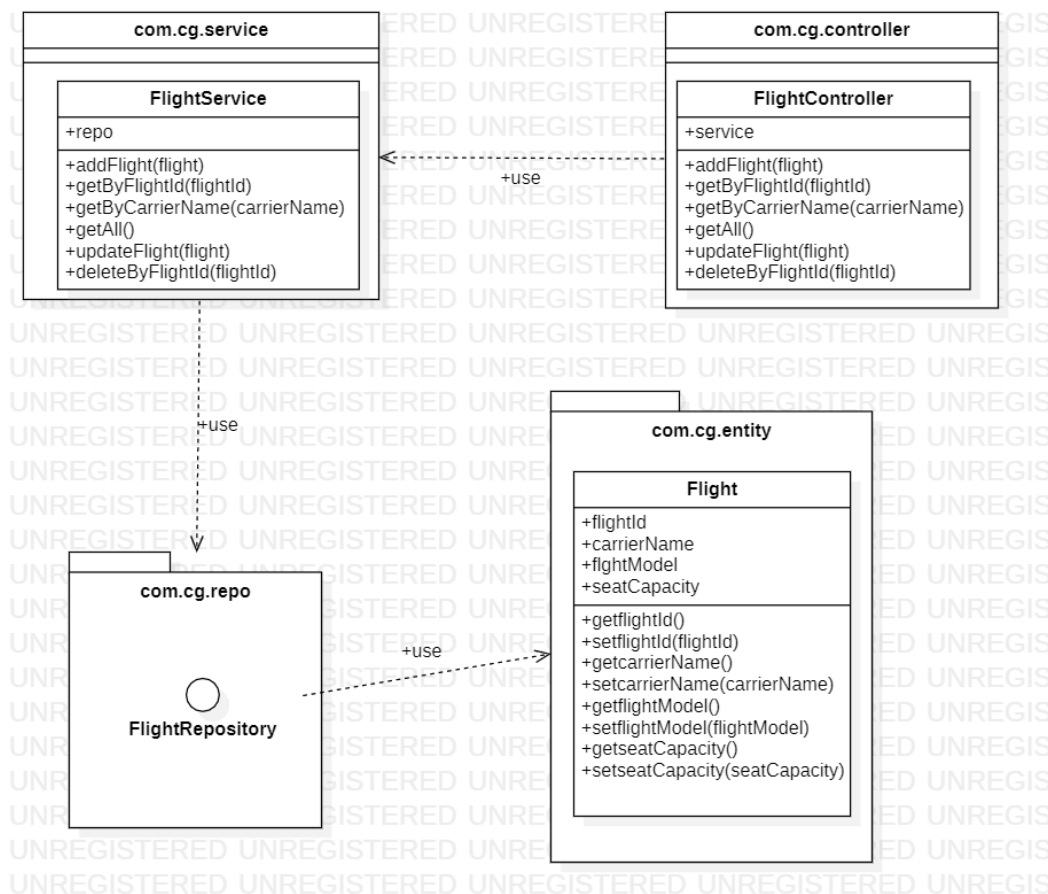| Epic	Stories	As a/an	I want to	So that...
Booking Service	Add a Booking	User	Enter the passenger details for booking	A new booking ticket is created with a new booking id
	Update booking	User	To update ticket status	User can update the ticket
	View User booking details	User	To search a ticket by Booking Id	To find Ticket details
	Cancel a Booking	User	Cancel a booking by booking Id	A user can cancel a booking.

Class Diagram:

com.cg.service

BookingService

+repo

+addByBookingId(BookingId)
+updateByBookingId(BookingId)
+viewByBookingId(BookingId)
+cancelByBookingId(BookingId)

com.cg.controller

BookingController

+service

+addByBookingId(BookingId)
+updateByBookingId(BookingId)
+viewByBookingId(BookingId)
+cancelByBookingId(BookingId)

+use

+use

com.cg.entity

Booking

+bookingId
+userId
+bookingDate
+ticketCost
+noOfPassengers
+flight

+getbookingId()
+setbookingId(bookingID)
+getuserId()
+setuserId(userId)
+getbookingdDate()
+setbookingDate(bookingDate)
+getticketCost()
+setticketCost(ticketCost)
+getnooOfPassengers()
+setnoOfPassengers(noOfPassengers)
+getFlight()
+setFlight(Flight)

com.cg.repo

◯

BookingRepository

+use

Route Service:

Class Diagram:

com.cg.service

RouteService
+repo

+getBySourceAndDestination()
+getAll()

com.cg.controller

RouteController
+service

+getBySourceAndDestination()
+getAll()

+use

+use

com.cg.repo

○

RouteRepository

+use

com.cg.entity

Route
+sourceAirport
+destinationAirport
+arrivalTime
+departureTime
+airportName
+airportLocation

+getSourceAirport()
+setSourceAirport(sourceAirport)
+getDestinationAirport()
+setDestinationAirport(destinationAirport)
+getArrivalTime()
+setArrivalTime(arrivalTime)
+getDepartureTime()
+setDepartureTime(departureTime)
+getAirportLocation()
+setAirportLocation(airportLocation)
+getAirportName()
+setAirportName(airportName)

Epic Stories:

| Epic	Stories	As a/an	I want to	So that...
Route Service	Find a Flight by source and destination	User	Find the Flight using source and destination	User can easily find the desired Flight destination
	Find all airports	User / Admin	To search all airport details	User / Admin can find all airport details

Flight Service:

Class Diagram:

Epic Stories:

Epic	Stories	As a/an	I want to	So that...
Flight Service	Add a Flight	Admin	Add a new flight	Admin can add a new
	View a flight	Admin	To search all Flight details	Admin can view all the details of every flight
	Delete a Flight	Admin	Delete a Flight by Flight Id	A user/ Admin can delete a Flight.
	Update a flight	Admin	Update a Flight by Flight ID	User / Admin can update flight.

Schedule Service:

Epic Stories:

Epic	Stories	As a/an	I want to	So that...
Schedule Service	Get a Scheduled flight	User	Find a scheduled flight using date and time	User can find a scheduled flight using date and time
	Update a scheduled flight	Admin	To update flight status	Admin can update scheduled flight
	Delete a scheduled flight	Admin	To delete a scheduled flight	Admin can cancel a flight
	Get all	Admin	To get the details of all scheduled flights	Admin can view all the details of scheduled flights

Class Diagram
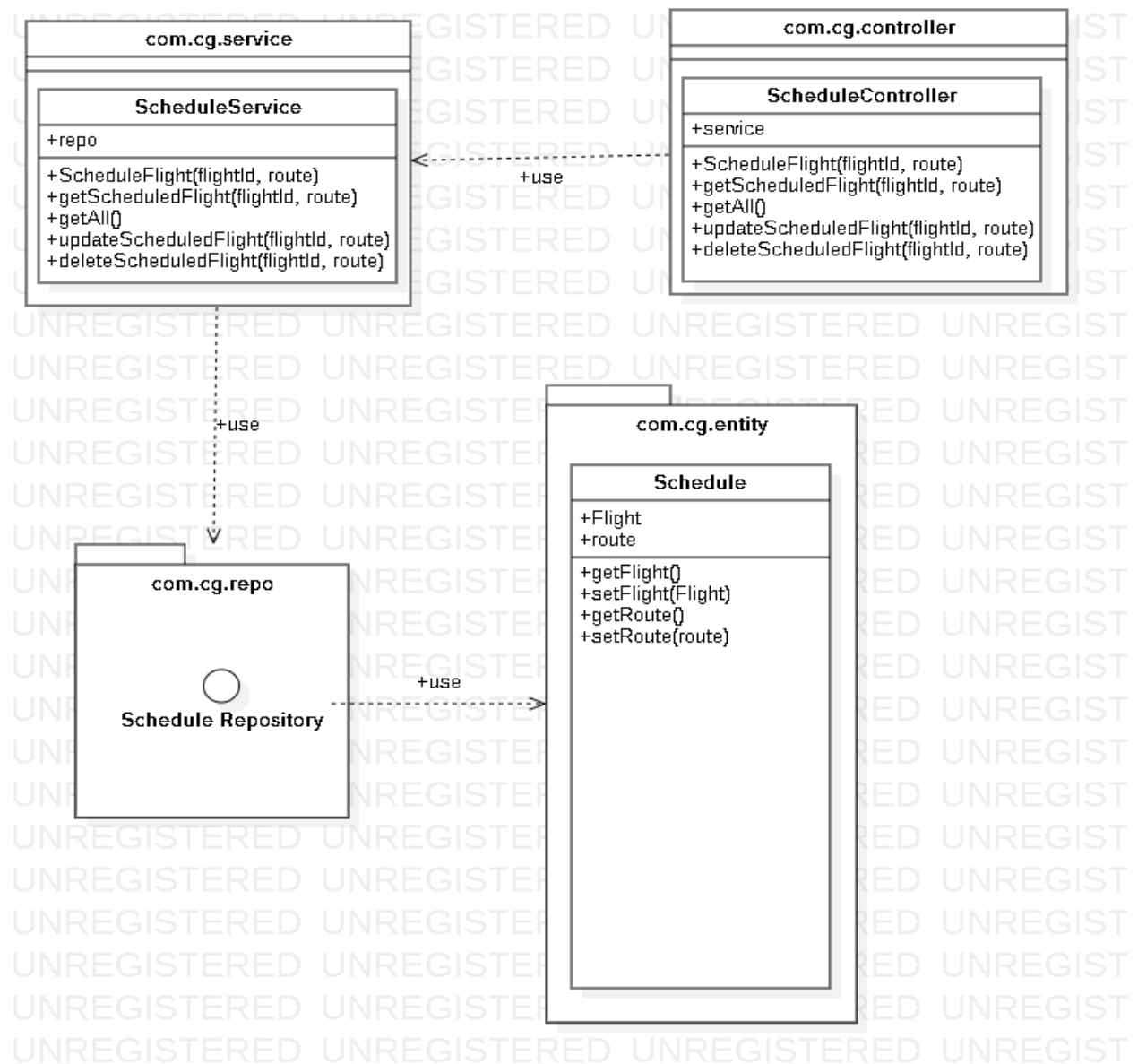
Validations:

1. The 'userPhoneNo' should have an exact 10 digit number and the number should not start with zero.
2. Date and Time should be valid i.e date and time that has already elapsed shouldn't be entered.
3. The local part of the email should contain alphanumeric characters only. No special characters are to be present as the first character of the id.
4. The chosen airport's name should be present inside the Airport database.
5. The Unique Identification Number should be of 12 digits.

Assumptions:

However, we have made a few assumptions with respect to the application, which are:
1. Administrator and customer are both Users. They are differentiated by a variable 'userType' in the User class.
2. Every passenger needs to enter a Unique Identification Number while booking is being made. For simplicity, we assume it to be a 12-digit Aadhaar Number.
3. All flights are direct flights.
4. Number of airports is fixed and stored in database.
5. All the flights are considered to be domestic.