Flight Management System Subsystems

The Flight Management System is composed of 4 subsystems:

1. Reservation System that has the ability to manipulate all flight reservations, seat selection, ticketing, flight availability, flight details, rates and conditions.
2. Profile Subsystem that manages all profiles within the system.
3. Service Subsystem that manages in-flight services such as food and drinks.
4. Reporting Subsystem that can generate various summery reports.

Each of the subsystems are further decomposed into smaller areas of concern:

1. Airports (represented in the system by Airport.java)
2. Bookings (represented in the system by Booking.java, ServiceBooking.java and Ticket.java)
3. Fleet (represented in the system by Plane.java)
4. Flights (represented in the system by Flight.java)
5. Routes (represented in the system by Route.java)
6. Services (represented in the system by Service.java)
7. Profiles (represented in the system by Customer.java, Person.java and Staff.java)
8. Reports (produced by ReportBuilder.java)

We shall now describe each subsystem in more detail and outline the classes that enable each subsystem to work together as a single, cohesive system.

1. Reservation System

The reservation system, which is the core of the system, allows users to perform all operations on bookings, seating, ticketing, flights, rates and conditions. The classes that support this system include:

* AirportController.java
* AirportEntity.java
* BookingController.java
* BookingEntity.java
* FleetController.java
* FleetEntity.java
* FlightController.java
* FlightEntity.java
* ProfileController.java
* ProfileEntity.java
* RouteController.java
* RouteEntity.java
* ServiceController.java
* ServiceEntity.java

In short, these are all the controller and entity classes that exist in the system, making it the most important and complex part of the system. When users make bookings, airport names and codes need to be known for the system to identify the routes that the user is interested in. In turn, we need to know those routes to identify which flights plough those routes. Furthermore, we need to know the services that are available to let the user book them. Lastly we need to know the user’s profile so that the system can charge the booking fee to the correct user.

2. Profile Subsystem

The profile subsystem is able to handle all aspects of profiles including user login details, user personal details and their fly status. This system is necessary for users to perform actions such as logging in, signing up and making bookings. The classes that support this system include:

* ProfileController.java
* ProfileEntity.java

These two classes contain the necessary methods to allow the client to perform actions such as updating profiles, changing passwords, and managing the watch and no fly list, in addition to the abovesaid functions.

3. Service Subsystem

The service subsystem is a less important part of the system that deals solely with services that can be booked for a particular user on a particular flight. This system would allow clients to perform actions such as add and delete services. The classes that support this system include:

* ServiceController.java
* ServiceEntity.java

These classes have methods to allow clients to manipulate and manage services in terms of pricing and avalability.

4. Reporting Subsystem

This subsystem allows clients to be able to view reports generated from the data that exist in the database. Reports such as flight reports and revenue reports are generated by the system for the various users of the system. The classes that support this system include:

* BookingController.java
* BookingEntity.java
* FleetController.java
* FleetEntity.java
* FlightController.java
* FlightEntity.java
* ReportBuilder.java
* RouteController.java
* RouteEntity.java
* ServiceController.java
* ServiceEntity.java

The reason why this system requires this many classes is because it requires the data from these various classes in order to generate useful reports. For example, we cannot know what routes are popular during a particular month if we did not link the routes and flights together.