Construction Phases

Iteration one (Ended 02/05/2015)

* Created the basic structure of the system that includes the main boundary class that the user will interact with, the controller class that will coordinate all interactions between the entity classes and the interface class, and the user entity class that enables users to login and signup to the system.
* Implemented the login functionality
* Created the following classes:
  + userInteraction package:
    - UserController.java
    - UserEntity.java
    - UserInterface.java
  + role package:
    - Role.java
    - AbstractCustomerRole.java
    - CustomerRole.java
    - TravelAgencyRole.java
    - RoleFactory.java
  + Main file:
    - FlightManagementSystem.java
  + Database files:
    - userAccount.csv
* We tested the main interface of the system by running the program and comparing the program’s output against the expected output.
* To test the login functionality, we created test login details in the database (because signup functionality is not implemented yet) and we tried logging into the system with the same login details.

Iteration two (Ended 09/05/2015)

* Finished the skeleton of all the role classes and the options that will be offered by each role. These options are just text strings that are used to build the main menu for users of that particular role and do not refer to actual system functionality.
* Changed Role.java from an interface to an abstract class to allow subclasses to have a common method to create the main menu.
* Implemented the logout functionality
* Implemented the signup functionality
* Created the following classes:
  + role package:
    - FlightManagerRole.java
    - ProfileSystemManagerRole.java
    - ReportingSystemManagerRole.java
    - ReservationSystemManagerRole.java
    - ServiceSystemManagerRole.java
    - SystemAdministratorRole.java
  + helpers package:
    - UserLoginDetails.java
    - Booking.java
    - Flight.java
    - Person.java
  + booking package:
    - AirportController.java
    - AirportEntity.java
    - BookingController.java
    - CustomerController.java
    - FlightController.java
    - FlightEntity.java
    - PersonController.java
    - PersonEntity.java
    - PlaneController.java
    - RouteController.java
    - RouteEntity.java
    - SeatController.java
    - ServiceController.java
  + Database files:
    - userDetail.csv
    - airports.csv
    - flight-route-schedule.csv
    - routes.csv
* Modified the following classes:
  + userInteraction package:
    - UserController.java
    - UserEntity.java
    - UserInterface.java
  + role package:
    - Role.java
    - AbstractCustomerRole.java
    - CustomerRole.java
    - TravelAgencyRole.java
    - RoleFactory.java
  + Main file:
    - FlightManagementSystem.java
* We tested the individual role classes by creating test login details in the database for each role. Then we tried logging into the system with the same login details for each particular role.
* We tested the signup functionality by just performing the actions described in the Signup use case. Then we tried to login to the system using the newly created details.
* We tested the logout functionality by choosing the logout option from the main menu.
* We tested the partial functionality of the booking system by trying both expected input and unexpected input. Booking system is halfway complete. Should be complete and tested by the next iteration.
* Integration errors included method names that were incorrectly spelled and methods having the wrong return type. Apart from that, there were no major hindrances during systems integration.

Iteration three (Ended 16/05/2015)

* Implementing the booking functionality. This includes transactions such as adding user bookings, booked seats, booked services and payment.
* Added change password functionality.
* Created the following classes:
  + helpers package:
    - Service.java
    - ServiceBooking.java
    - Ticket.java
  + booking package:
    - BookingEntity.java
    - FleetController.java
    - FleetEntity.java
    - ServiceEntity.java
  + profile package:
    - ProfileController.java
    - ProfileEntity.java
  + Database files:
    - discount.csv
    - fleet.csv
    - otherPersons.csv
    - servicesInventory.csv
* Modified the following classes:
  + helpers package:
    - Booking.java
    - Flight.java
    - Person.java
    - UserLoginDetails.java 🡪 Customer.java
  + booking package:
    - AirportController.java
    - AirportEntity.java
    - BookingController.java
    - FlightController.java
    - FlightEntity.java
    - PersonController.java 🡪 moved to profile package
    - PersonEntity.java 🡪 moved to profile package
    - RouteController.java
    - RouteEntity.java
    - ServiceEntity.java
  + role package:
    - Role.java
    - AbstractCustomerRole.java
    - CustomerRole.java
    - TravelAgencyRole.java
  + Database files:
    - flight-route-schedule.csv
    - userAccount.csv
* Deleted the following classes
  + booking package:
    - CustomerEntity.java
    - CustomerController.java
    - PlaneController.java
* Booking functionality complete. Have not done rigorous testing on it yet to determine correctness.
* Tested the change password functionality by changing the password for an account, and trying to login to the system using the old password and then trying to login to the system using the new password.
* Integration errors include missing import statements because one of the programmers is not using an IDE.

Iteration four (Ended 23/05/2015)

* Implemented booking functionality. Customers and travel agencies are able to book flights, seats and services on flights.
* Implemented the edit account details functionality. Customers and travel agencies are able to edit the details of their account.
* Implemented the cancel booking functionality. Customers and travel agencies can cancel bookings they have made. They will be charged a cancellation fee.

Summarize the new elements added and the extensions to existing implementation elements. Include brief details of unit testing procedures used to verify new elements prior to their commitment to the project (this does not mean list unit test code and test outputs, simply identify the additional tests created by individual implementers).

o Summarize data from defect and integration reports created by the systems integrator.