**INTRODUCTION**

ABSTRACT

**Online Airline Reservation System** is a Web based application for a particular Airlines company. This application facilitates registering, updating, and utilization of miles for a Frequent Flier of its flights. The frequent flier should be able to login and key in all the details of his travel in their airlines. The application is accessible for the frequent fliers from the existing website of the airlines.This project mainly designed for frequent flyer belongs to particular airline service.

**What is Frequent Flyer Program?**

A **frequent flyer program** (FFP) is a [loyalty program](http://en.wikipedia.org/wiki/Loyalty_program) offered by many [airlines](http://en.wikipedia.org/wiki/Airlines). Typically, airline customers enrolled in the program accumulate frequent flyer miles (kilometers, points, segments) corresponding to the distance flown on that airline. Acquired miles can be redeemed for free air travel; for other goods or services.

**How to calculate Frequent Flyer points:**

The actual monetary value is typically 2 cents per frequent flyer mile (reportedly the airline industry average). For example, if you want to redeem 25,000 miles for a free ticket, the number of miles multiplied by 2 cents per mile is $500. As some industry researchers have noted, it may be smarter to save your miles and purchase a lower cost ticket to your destination.

You should always have your frequent flyer number handy when speaking with the [airlines](http://travel.howstuffworks.com/airline.htm). You purchase a ticket with miles the same way you do with [cash](http://money.howstuffworks.com/currency.htm). They will use your number to access your account and subtract the appropriate amount of miles as you purchase the ticket. Tickets are available in paper form via regular mail or shipping carriers, and, of course, as an **e-ticket**.

**FEASIBILITY REPORT**

**FEASIBILITY REPORT**

**TECHNICAL FEASIBILITY:**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical

analysis.

1. **Understand the different technologies involved in the proposed system:**

Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.

1. **Find out whether the organization currently possesses the required technologies:**
   * Is the required technology available with the organization?
   * If so is the capacity sufficient?

For instance –

“Will the current printer be able to handle the new reports and forms required for the new system?”

**OPERATIONAL FEASIBILITY:**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
* Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
* Have the user been involved in the planning and development of the project?
* Early involvement reduces the chances of resistance to the system and in
* General and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**ECONOMIC FEASIBILITY:**

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**SYSTEM**

**REQUIREMENT SPECIFICATION**

**OVERVIEW**

Miles Acquisition System (MAS) is a Web based application for a particular Airlines company. This application facilitates registering, updating, and utilization of miles for a Frequent Flier of its flights. The frequent flier should be able to login and key in all the details of his travel in their airlines. The application is accessible for the frequent fliers from the existing website of the airlines.

And also this provide the information how earn points and a how to redeem points and how to utilize the points. This provides the information about flights and hotels of the particular airline services.

**Functional components of the project**

1. Registration for a new flier.
2. Authentication of its users before usage of application.
3. Options to utilize the miles accumulated so far
4. Once the miles are utilized, the corresponding miles will be deducted from the accumulated miles.
5. The registered user can see flight status and flight details directly from this system alslo.
6. And he/she can also book a ticket from this system.

**STUDY OF THE SYSTEM**

In the flexibility of uses the interface has been developed a graphics concepts in mind, associated through a browser interface. The GUI’s at the top level has been categorized as follows

1. Administrator Interface Design.
2. User Interface.
3. Security Authentication.
4. Reports.
5. General end-users.

The administrative user interface will maintain the different users details, the interface helps the administration with all the transactional states like which users sending the mails, and which users receiving whishing mails, users details information history. And the statistics of the system in difference stratagies.

**NUMBER OF MODULES**

The system after careful analysis has been identified to be presented with the following modules:

The Modules involved are

1. Authentication and Security Module.
2. Admin Module
3. Flyer Module.
4. Flight Management Module.
5. Service Management Module.
6. Reports Module.

**MODULES DESCRIPTION**

Module description for each module mension on top

**Admin Module:**

In this administration module, admin should login to the system to access this system. After he logged Admin can add flight details and also he/she can modify the flight details belong to particular airline services. And also he/she has able to add services like hotels in different countries.

And admin can also monitor the users login and logout details.

And also he can suggest to the frequent flyer through mails about the points and services. The admin can communicate with the users of this system.

**Flight Management Module:**

In this module admin can add flight details and also he/she can modify and delete the flight details. This module serves the user like on which dates flights available and how many seats are available to book a ticket.

**Frequent Flyer Module:**

In this module the flyer should register to access this system. Frequent flyer means, the person who travel frequently by air. The frequent flyer should login to the system, to utilize the benefits of the frequent flyer offered by particular airline services. This system maintains the complete details of the registered user and also its facilitate to update and modify the details of the user. In this module the user can book a ticket and see the status of the flight between two points. This system facilitates to know how many points the user need to travel from one place to another place with free ticket.

The frequent flyer of this system can send queries to the admin via mails and he can also ask about the accumulate miles.

**Security and authentication module:**

In this module this system stores the details of the registered user with safe and secure manner. And it can allow only authentication user to access the system. This system prevents the unauthorized accessing and malfunctioning.

**Service Management Module:**

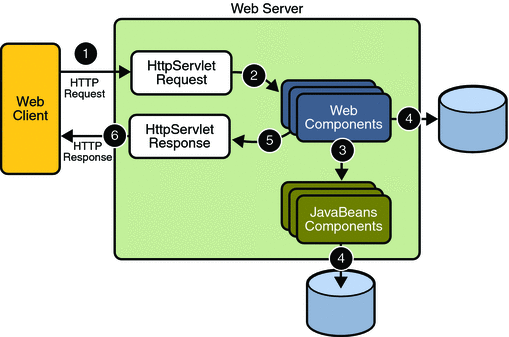
In this module the administration can add the services provided by the airline service and he can also modify the services .He has to add service from source to destination and distance between two points.

**Report Module:**

In this module the data from the database will be shown in tabular format as report to the users of this system.

**PROCESS FLOW**

**ARCHITECTURE DIAGRAM**



1. **THE PRESENTATION LAYER**

Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

1. **THE BUSINESS RULES LAYER**

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process

1. **THE DATA ACCESS LAYER**

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

1. **THE DATABASE LAYER**

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc.   
In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

**SDLC METHODOLOGIES**

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

**The following diagram shows how a spiral model acts like:**



**Fig 1.0-Spiral Model**

**ADVANTAGES**

* Estimates(i.e. budget, schedule etc .) become more relistic as work progresses, because important issues discoved earlier.
* It is more able to cope with the changes that are software development generally entails.
* Software engineers can get their hands in and start woring on the core of a project earlier.

**SOFTWARE REQUIREMENT AND**

**HARDWARE REQUIREMENT**

**SOFTWARE REQUIREMENTS**

Operating System : Windows XP/2003 or Linux

User Interface : HTML, CSS

Client-side Scripting : JavaScript

Programming Language : Java

Web Applications : JDBC, Servlets, JSP

IDE/Workbench : My Eclipse 6.0

Database : Oracle 10g

Server Deployment : Tomcat 6.x

**HARDWARE REQUIREMENTS**

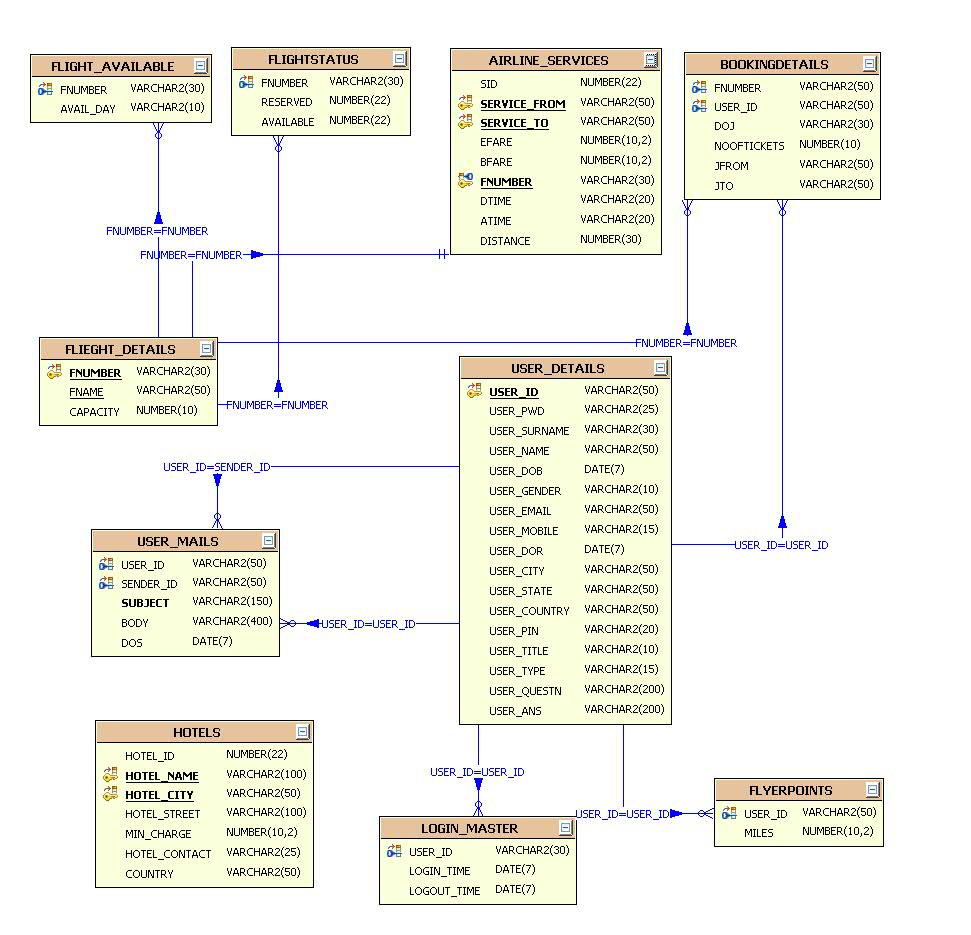
Processor : Pentium IV

Hard Disk : 40GB

RAM : 1GB or more

**SYSTEM DESIGN**

**E-R DIAGRAM**

****

**UML DIAGRAMS**

**UNIFIED MODELING LANGUAGE DIAGRAMS**

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

**USER MODEL VIEW**

This view represents the system from the users perspective.

The analysis representation describes a usage scenario from the end-users perspective.

**STRUCTURAL MODEL VIEW**

In this model the data and functionality are arrived from inside the system.

This model view models the static structures.

**BEHAVIORAL MODEL VIEW**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**IMPLEMENTATION MODEL VIEW**

In this the structural and behavioral as parts of the system are represented as they are to be built.

**ENVIRONMENTAL MODEL VIEW**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

UML Analysis modeling, which focuses on the user model and structural model views of the system.

UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

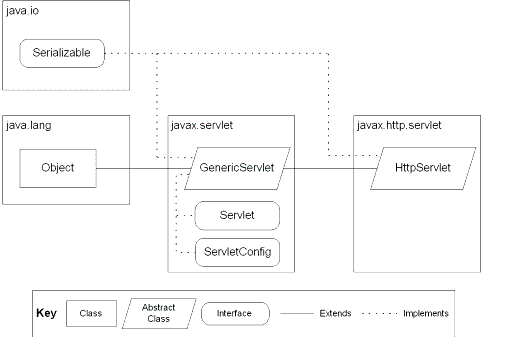
Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

**Class Diagram**

**CLASS DIAGRAM**

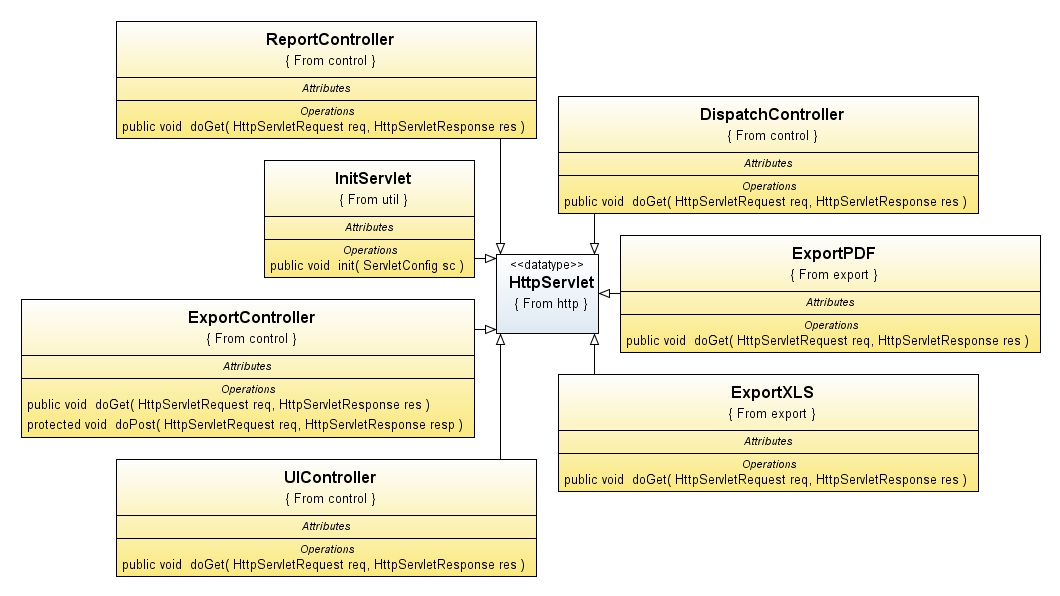
Class diagrams describe the structure of the system in terms of classes and objects. The servlet api class diagram will be as follows.

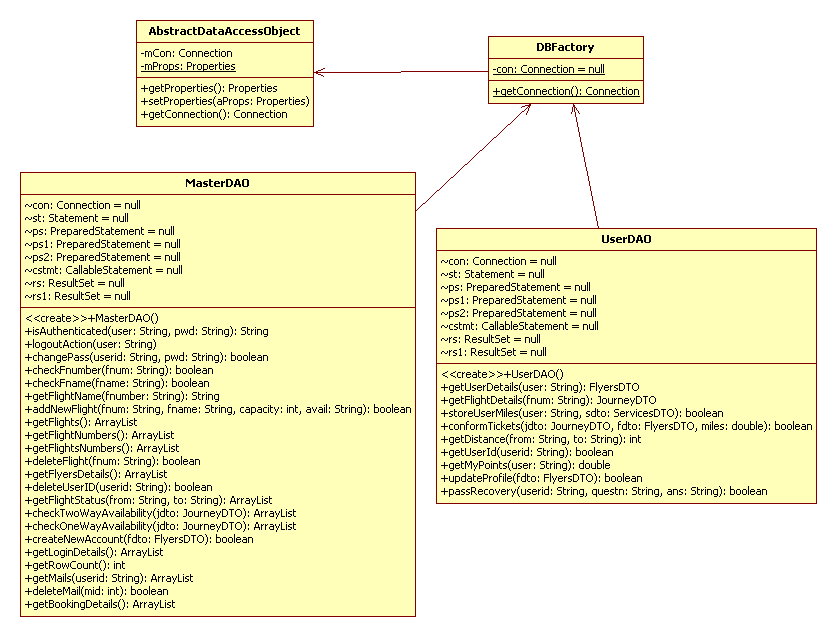
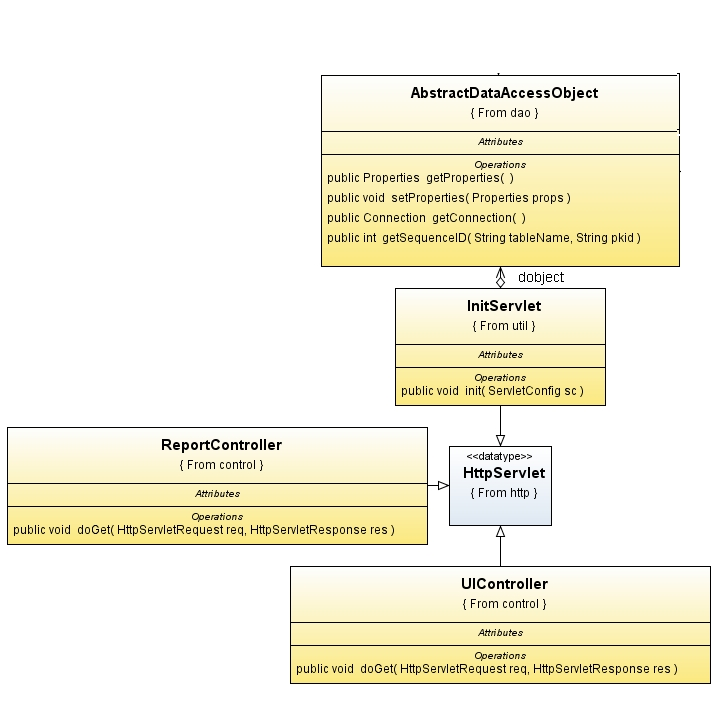


JSP: Implicit Objects

**Class Collaboration Diagrams**

**Class Collaboration Diagram**





**Use Case Diagrams**

**UML Diagrams**

**Unified Modeling Language**:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + User Model View
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + Structural model view
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

* Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

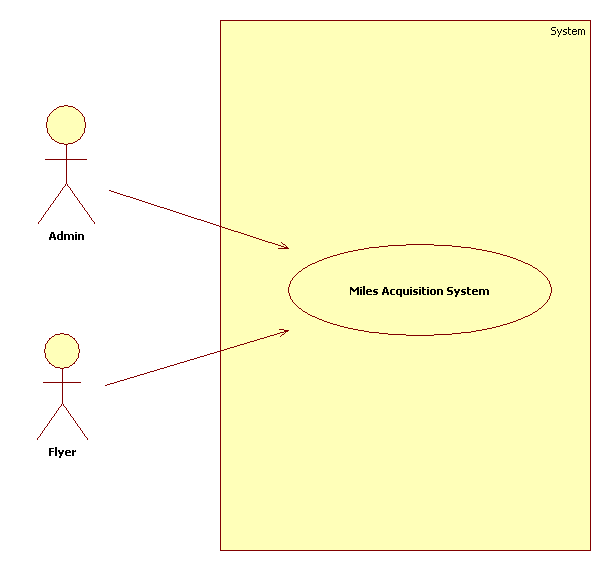
UML is specifically constructed through two different domains they are:

* UML Analysis modeling, this focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

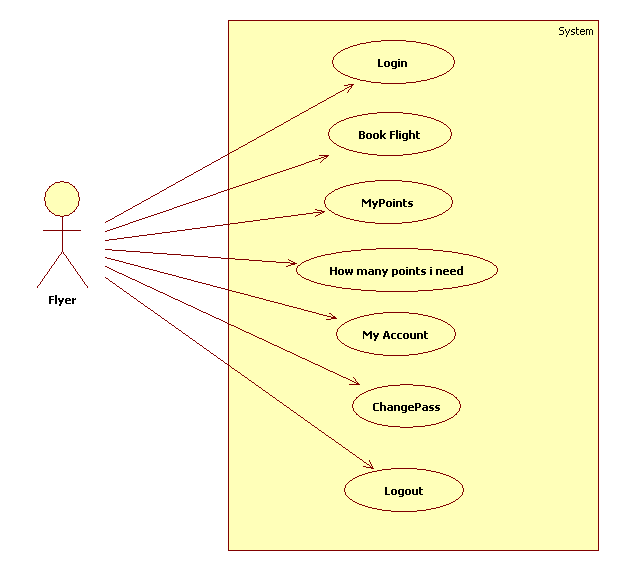
Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

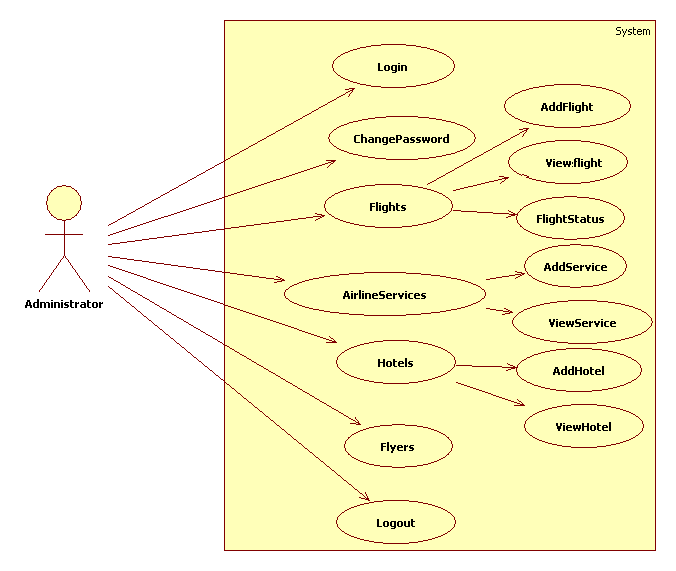
1. **system Use Case Diagram**

****

1. **User Use Case Diagram**

****

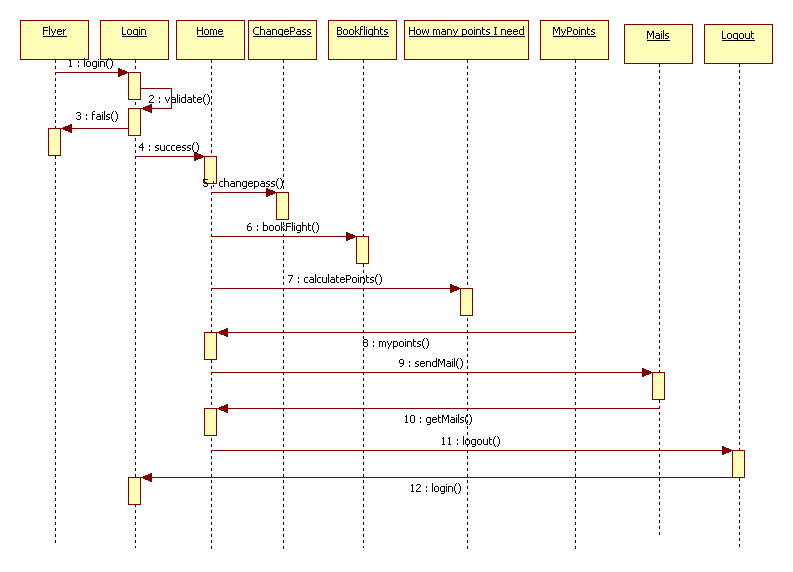
1. **Administrator Use Case Diagram**

****

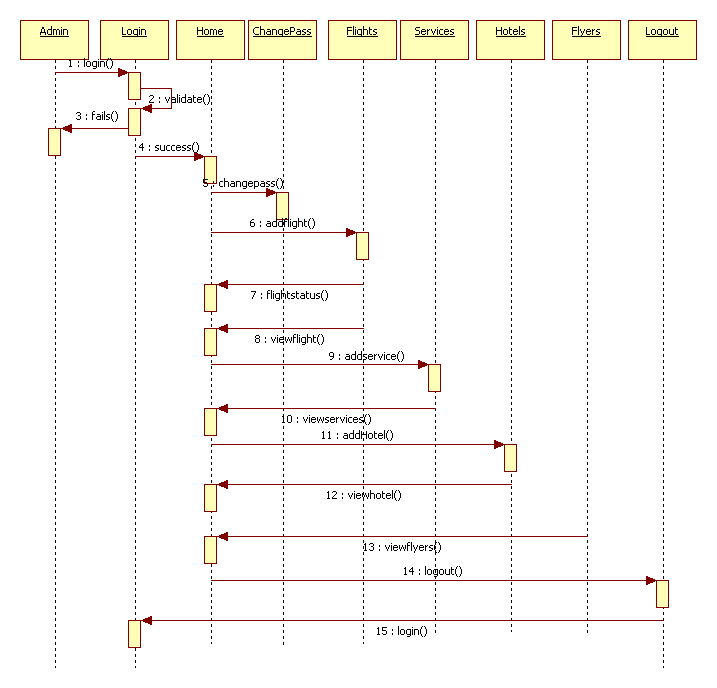
**Sequence Diagrams**

**User-Level Sequence Diagrams**

1. **Users Sequence Diagram**

****

1. **Administrator Sequence Diagram**



**Collaboration Diagrams**

**Operation-Level Sequence Diagram**

1. **Login Sequence Diagram**

****

**Login Collaborative Diagram**

****

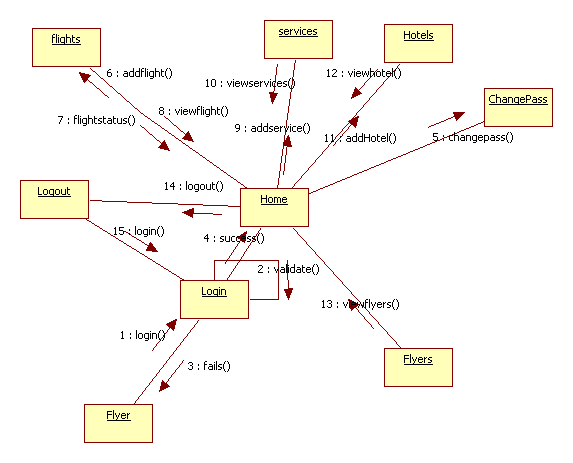
1. **Present Login User Report Sequence Diagram**

****

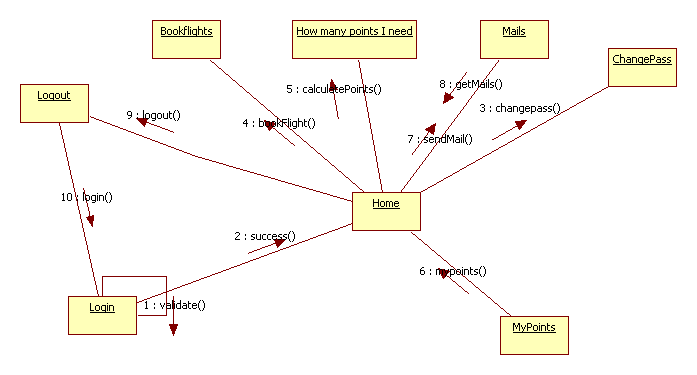
**Present Login User Report Collaborative Diagram**

****

**AdminCollaborationDiagram**



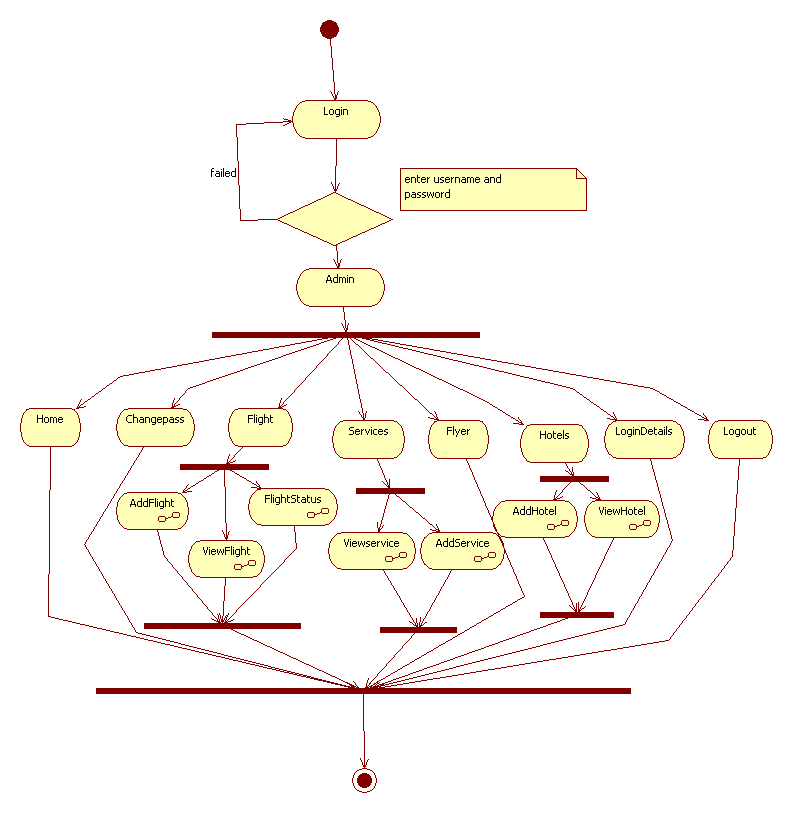
**UserCollaborationDiagram**



**ACTIVITY DIAGRAMS**

**ACTIVITY DIAGRAMS**

##### Administrator Activity Diagram



##### User Activity Diagram

##### 

**Component Diagram**

**Component Diagram :**

****

**Deployment Diagram**

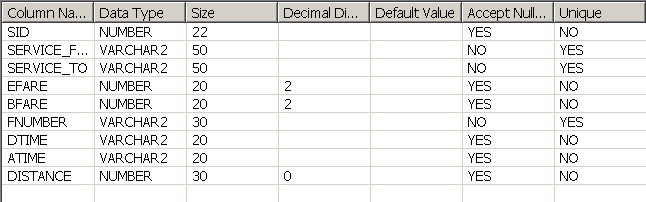
**Deployment Diagram:**

****

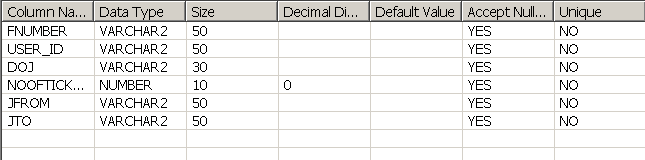
**Data Dictionary**

**Data Disctinory**

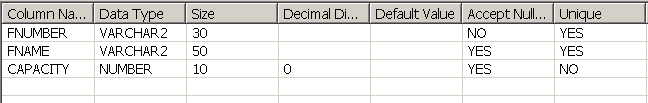
AIRLINE\_SERVICES



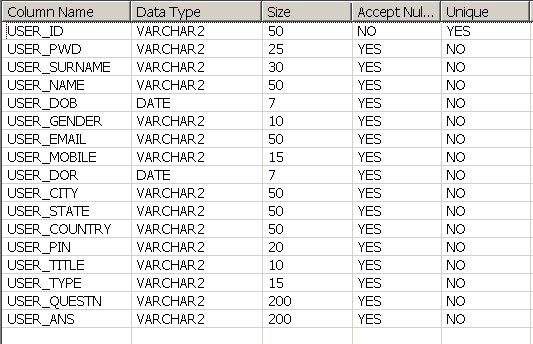
BOOKINGDETAILS



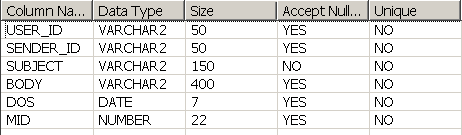
FLIEGHTDETAILS



USERDETAILS



USERMAILS

****

FLYERPOINTS

