# INTRODUCTION TO PROJECT

The web based “Airline Reservation System” project is an attempt to stimulate the basic concepts of airline reservation system. The system enables the customer to do the things such as search for airline flights for two travel cities on a specified date, choose a flight based on the details and reservation of flight

The system provides you Quick Search facility that provides you details about flights without login . But if user want to book ticket then it must require login into your account.

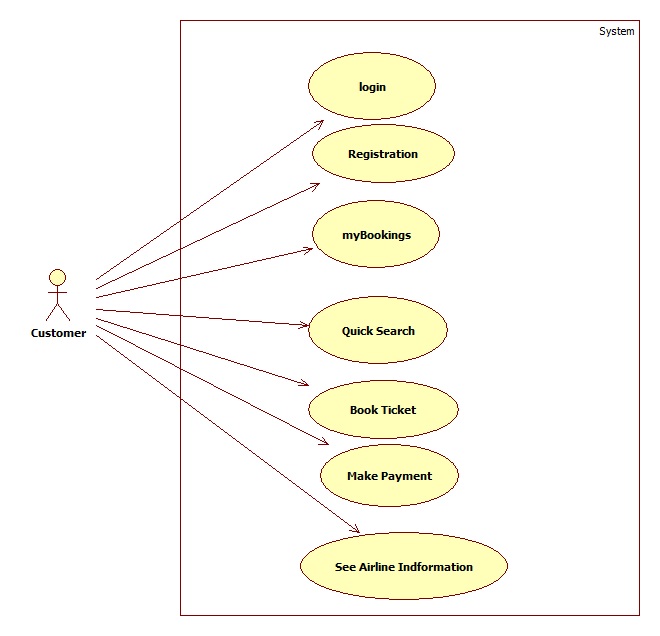
The system allows the airline passenger to search for flights that are available between the two travel cities, namely the “Departure city” and “Destination city” for a particular departure and arrival dates. The system displays all the flight’s details such as flight no, name, price and time of journey etc.

Here we provided quick search facility which displays list of available flights and allows customer to choose a particular flight. Then the system checks for the availability of seats on the flight. If the seats are available then the system allows the passenger to book a seat. Otherwise it asks the user to choose another flight.

To book a flight the system asks the customer to enter his details such as name, address, city, state, credit card number and contact number. Then it checks the validity of card and book the flight and update the airline database and user database.

**2.REQUIREMENTS**

**2.1 FUNCTIONAL REQUIREMENTS**

****

**2.1 User Account**

The passenger, who will henceforth be called the ‘user’, will be presented with 3 choices by the reservation system, as the first step in the interaction between them. A user can choose one of these and his choice would be governed by whether he is a guest or a registered user and whether he wants to check the availability of tickets or also block/buy them. The terms ‘registered user’ and ‘guest’ are described below.

A user who has traveled by the airline earlier would have been given a user id and a password. This ‘personal information’ would be henceforth referred to as ‘profile’. Such a user with a profile in DB-user shall be called a ‘registered user’. A registered user will be able to check the availability of tickets as well as block/buy a ticket by logging into the system.

A new user, on the other hand, would either have to

1. register himself with the system by providing personal information or
2. log into the system as a guest.

In case of ‘a’, the new user becomes a registered user.

In case of ‘b’, the new user would remain a guest.

A guest can only check the availability of tickets and cannot block or buy tickets.

But a registered user can also act as a guest if he only wants to check the availability of tickets.

‘Availability of tickets’ always refers to viewing the flight schedule for given days, the price of tickets and any discount offers. The system shall present the user with an option to exit from the system at any time during the following processes.

**2.2 Registration and creation of user profile**

The system shall require a user to register, in order to carry out any transactions with it except for checking the availability of tickets. It will ask the user for the following information at the least – a user id, a password, first name, last name, address, phone number, email address, sex, age, preferred credit card number. The system will automatically create a ‘sky miles’ field and initialize it to zero in the user’s profile.

****

* 1. **Quick Search**

Here we provided Quick Search facility for any user to search flight schedule without login into account .This will provide user an option for searching flight and comparing their prices of all companies.

After logging in a user (either a registered user or a guest), the system shall

request him to enter the following details – origin city and destination city. “City’ is

ageneric term and refers to a city or town as the case may be. The original destination

cities would be entered as text.

After the origin and destination cities are ascertained, the system shall now access the flight schedule database, referred to as ‘flightmaster’, and checks if there is a direct operational service between the two cities.

The system shall now ask the user to enter the following details - class, one-way,

ar round trip, departure date and the number of adult passengers, children and senior citizens.

‘Class’ refers to Business class/Economy class. This choice shall be made by the user

through a drop down menu indicating all the possible combinations of choices.

One-way/round trip shall be button selection. ‘Departure date’ refers to either a single date or a range of dates, entered through text box.. In case, the trip is a round trip, the system shall also ask the user to enter the return date

Having taken all the above input from the user, the system checks for any false entries like the departure date on the return trip being earlier than the departure date on the onward trip. In case of incompatibility, the system will not display any flights available.

The system queries the flights database ‘flightmaster’ to check which of the flights on the schedule have seats available. The system displays the results in a suitable form (a tabular form) with the following information depicted – for each airlineId , flight number, departure time in origin city, arrival time in destination city,departurecity,arrival city ,Ticket price and the number of seats available on that flight.

There can be several flights of different airlines between two cities and all of them will be listed for the particular date that the user wants to depart from the Origin City. In case, the user has entered a range of dates, the system shall display all the flights for all those dates in the range. There will be a Book button in front of every row displayed n the table of flights searched.

The system will then ask for personal information of all passengers i.e. one registered user can book for multiple users. So all users will be added in the table.

The system shall now display the price of the ticket for the trip. This will be the sum of the prices for all the members of the travel party being represented by the user.

* 1. **Making Reservations/Blocking/Confirmation**

After having taken the user through the step 2.2, Checking Availability, The system will now ask the user if he wishes to block/buy the ticket. If yes, and

1. if the user has been a guest, he will have to first register and become a registered user and then log onto the system.
2. If the user is already a registered user, and if he has logged on already, he can block/buy the ticket, but if he has been acting as a guest, he will have to log on.

Having ensured that the user is logged on validly according to 3.4.1, the system compares the departure date with the system date. If the departure date falls within 2 weeks of the system date, the system informs the user that he has no option to block the ticket and asks him if he would like to buy it.

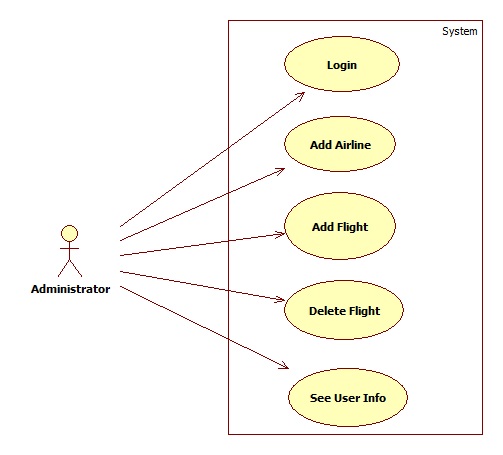
If the difference between the departure date and system date is more than 2 weeks, the system asks the user if he would like to block or buy the ticket. The system informs the user that he can block the ticket at no cost now. It also informs him that if he chooses to block the ticket, he should make a final decision before 2 weeks of the departure date. The system shall send an email to the user.

Having taken the input from the user in 3.4.2, the system shall now proceed to update the reservation database DB-reservation. It will decrement the number of available seats on the particular flight for the particular class by the number of travelers being represented by the user.

In case the user buys the ticket, the system asks for entering his or her bank information i.e. debit card or credit card information and then charges the price of the ticket to his debit card number.

**2.5 View Booking History**

The system shall allow a user to view all information about his previous bookings. After logging him on, it asks for his blocking number or his confirmation number. It accesses UserBookingtable and retrieves the details of the trip and presents them to the user in a tabular format.

****

Admin should be able to login ,add airline information , add flight information,

Delete flight and see user Information according to user Id.

* 1. **NON FUNCTIONAL REQUIREMENTS**

**2.2.1 Interface**

Go to Appendix B for user interfaces

**2.2.2 Performance**

* **Number of Concurrent Users:**

ARS shall be able to handle at least 1000 transactions/inquiries per

second

* **Booking of Tickets:**

The system is susceptible to any temporary server failure since it uses the strong feature of Struts 2 and Hibernate. Hence the examination will be continued even if the sever gets disconnected in between the examination.

**2.2.3 Constraint**

ARS shall be able to handle at least 1000 transactions/inquiries persecond

**2.2.4 Other Requirements:**

* **Hardware Interfaces**

The SPMS is expected to function on Intel PIII 900 MHz Processor equivalent or above, 128 MB RAM, 20 GB HDD.

* **Software Interfaces**

The SPMS shall work on MS Windows operating systems family (MS Windows 98, MS Windows NT Workstation, MS Windows 2000, MS Windows XP). It configures to work with Oracle database. This System works on Apache Tomcat server. It uses browser IE 5.0 & above. It uses IIS 5.0 server.

**3. DESIGN**

**3.1 Database Design**

The following table structures depict the database design.

# Table1: User\_Info

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key Type/ Constraint** | **Column Name** | **Data Type** | **Length** | **Allow Null (1=Yes;0=No)** |
| 3 | UserID | Number | 4 | 0 |
| 0 | Birthdate | Date | 6 | 1 |
| 0 | City | Varchar2 | 255 | 1 |
| 0 | Phonenumber | Varchar2 | 255 | 1 |
| 0 | Country | Varchar2 | 255 | 1 |
| 0 | Email | Varchar2 | 255 | 1 |
| 0 | Firstname | Varchar2 | 255 | 1 |
| 0 | State | Varchar2 | 255 | 1 |
| 0 | Lastname | Varchar2 | 255 | 1 |
| 0 | Gender | Char | 1 | 1 |
| 0 | Username | Varchar2 | 255 | 1 |
| 0 | Password | Varchar2 | 255 | 1 |

## **Table2: Administrator\_Login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Adminname | Varchar2 | 15 | 0 |
| 0 | adminpassword | Varchar2 | 15 | 1 |

# Table3: AirlineMaster

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | AirlineID | Number | 5 | 0 |
| 0 | AirlineName | Varchar2 | 25 | 1 |
| 0 | AirlineDetail | Varchar2 | 2000 | 1 |
| 0 | ImageURL | Varchar2 | 30 | 1 |

### Table4: FlightMaster

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | FlightID | Number | 5 | 0 |
| 3 | AirlinesID | Number | 10 | 0 |
| 0 | Fromsrc | Varchar2 | 255 | 1 |
| 0 | Todest | Varchar2 | 255 | 1 |
| 3 | Fromdate | Date | 8 | 0 |
| 0 | Todate | Date | 8 | 1 |
| 0 | Fromtime | Date | 15 | 1 |
| 0 | Destime | Date | 15 | 1 |
| 0 | Bussinessseats | Number | 3 | 1 |
| 0 | Economicseats | Number | 3 | 1 |
| 0 | Bsprice | Float | 126 | 1 |
| 0 | Discount | Float | 126 | 1 |
| 0 | Ecprice | Float | 126 | 1 |

**Table5: Bank**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | BankId | Number | 5 | 0 |
| 0 | Cardtype | Varchar2 | 10 | 1 |
| 3 | Cardnumber | Number | 15 | 0 |
| 0 | Pinnumber | Number | 4 | 1 |
| 0 | Validfrom | Date | 7 | 1 |
| 0 | Validthrough | Date | 7 | 1 |

### Table6:User Booking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | AirlineID | Number | 10 | 0 |
| 0 | FlightID | Varchar2 | 255 | 1 |
| 0 | SeatNo | Number | 15 | 1 |
| 0 | UserID | Number | 4 | 1 |
| 0 | ResID | Nimber | 5 | 1 |
| 0 | FlightID | Varchar2 | 255 | 1 |
| 0 | AirlineID | Number | 10 | 1 |
| 0 | PersonFirstName | Varchar2 | 15 | 1 |
| 0 | PresonMiddleName | Varchar2 | 15 | 1 |
| 0 | PersonLastName | Varchar2 | 15 | 1 |
| 0 | Age | Number | 2 | 1 |
| 0 | Gender | Char | 1 | 1 |
| 0 | VisaID | Varchar2 | 12 | 1 |
| 0 | PassportID | Varchar2 | 15 | 1 |

**E-R Diagram,Dataflow diagram and Class Diagram:**

Go to Appendix A

**4. CODING STANDARDS IMPLEMENTED**

### Naming and Capitalization

Below summarizes the naming recommendations for identifiers in Pascal casing is used mainly (i.e. capitalize first letter of each word) with camel casing (capitalize each word except for the first one) being used in certain circumstances.

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Case** | **Examples** | **Additional Notes** |
| Class | Pascal | Person, BankVault, SMSMessage, Dept | Class names should be based on "objects" or "real things" and should generally be **nouns**. No ‘\_’ signs allowed. Do not use type prefixes like ‘C’ for class. |
| Method | Camel | getDetails, updateStore | Methods should use **verbs** or verb phrases. |
| Parameter | Camel | personName, bankCode | Use descriptive parameter names. Parameter names should be descriptive enough that the name of the parameter and its type can be used to determine its meaning in most scenarios. |
| Interface | Pascal with "I" prefix | Disposable | Do not use the ‘\_’ sign |
| Property | Pascal | ForeColor, BackColor | Use a noun or noun phrase to name properties. |
| Associated private member variable | \_camelCase | \_foreColor, \_backColor | Use underscore camel casing for the private member variables |
| Exception Class | Pascal with "Exception" suffix | WebException, |  |

### Comments

* Comment each type, each non-public type member, and each region declaration.
* Use end-line comments only on variable declaration lines. End-line comments are comments that follow code on a single line.
* Separate comments from comment delimiters (apostrophe) or // with one space.
* Begin the comment text with an uppercase letter.
* End the comment with a period.
* Explain the code; do not repeat it.

**5. TEST REPORT**

**Another group called Linux did the testing and the report of the testing is given hereunder.**

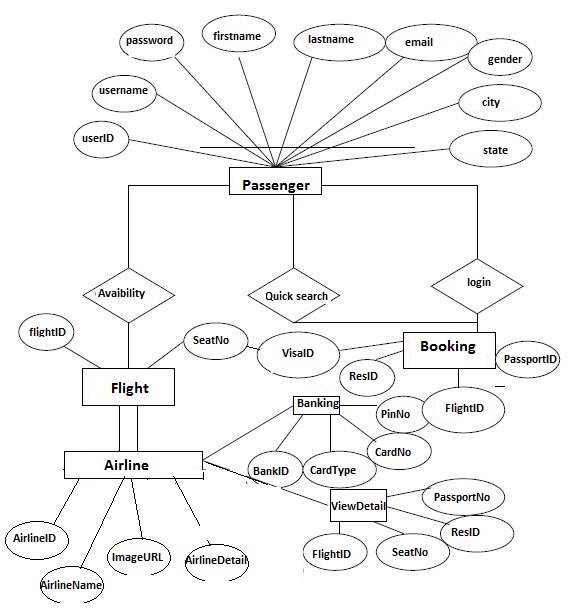
**GENERAL TESTING:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SR-NO** | **TEST CASE** | **EXPECTED RESULT** | **ACTUAL RESULT** | **ERROR MESSAGE** |
| 1 | Register Page | Redirected to Next page | OK | Nothing |
| 2 | Login Page | Pop-up will come | Ok | Please enter username and password again . |
| 3 | Reset login | Only users password will be reseted | Ok | Nothing |
| 4 | Quick search flight | Gives all flight details | Ok | Nothing |
| 5 | Booking Ticket | All the fields should be filled for submission | Ok | Nothing |
| 6 | Checking login or not | User is logged in or not | Ok | Nothing |
| 7 | Add person details for tickets | Add informations according to no of seats allocated | Ok | Nothing |
| 8 | Goto ticket page | Set added information about person | Ok | Nothing |
| 9 | Add information in booking table | Save this all data into booking table | Ok | Nothing |
| 10 | Transaction | On back it should be reverted to previous page | Ok | Nothing |
| 11 | View transaction done | It shows you all transactions done previously | Ok | Nothing |
| 12 | Logout | It will logout from user profile. | Ok | Nothing |
|  | **STATIC TESTING** |  |  |  |
| **SR-NO** | **Deviation** | **Program** |  |  |
| 1 | Commenting not followed | All Web Application |  |  |

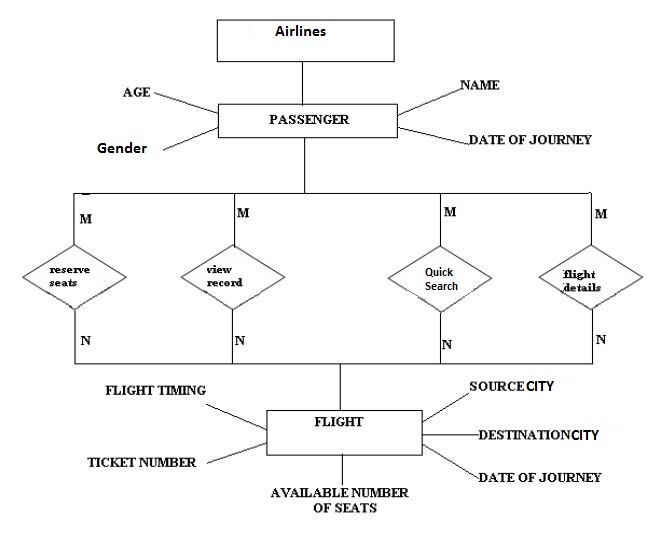
**6. PROJECT MANAGEMENT RELATED STATISTICS**

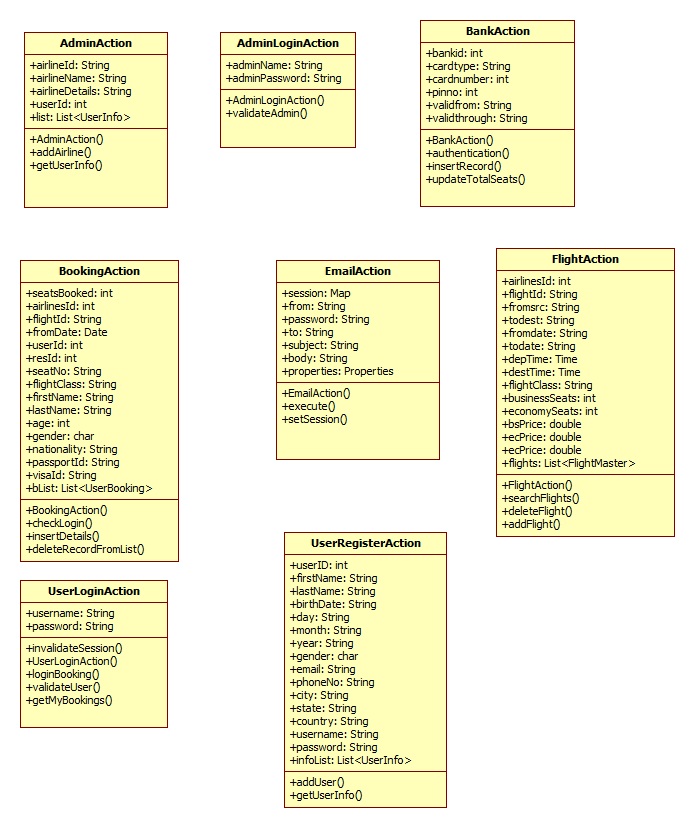
|  |  |  |  |
| --- | --- | --- | --- |
| **DATE** | **WORK PERFORMED** | ****SLC Phase**** | **Additional Notes** |
| JAN 12,2015 | Project Allotment and User Requirements Gathering | Feasibility Study | Our team met the client Mr. Nitinkudale (CEO, SIIT Pune) to know his requirements. |
| JAN 17,2015 | Initial SRS Document Validation  And Team Structure Decided | Requirement Analysis  (Elicitation) | The initial SRS was presented to the client to understand his requirements better |
| JAN 18,2015 | Designing the use-cases, Class Diagram, Collaboration Diagram, E-R Diagram and User Interfaces | Requirement Analysis &  Design Phase | Database Design completed |
| JAN 19,2015 | Business Logic Component design Started | Design Phase | ---------------------- |
| JAN 20,2015 | Coding Phase Started | Coding Phase | 70% of Class Library implemented. |
| JAN 21,2015 | Implementation of Web Application and Window Application Started | Coding Phase | Class Library Development going on. |
| JAN 22, 2015 | Off | Off | Off |
| JAN 23, 2015 | Implementation of Web Application and Window Application Continued | Coding Phase and Unit Testing | Class Library Modified as per the need. |
| JAN 24, 2015 | Implementation of Web Application and Window Application Continued | Coding Phase and Unit Testing | -- |
| JAN 25, 2015 | After Ensuring Proper Functioning the Required Validations were Implemented | Coding Phase and Unit Testing | Module Integration was done by the Project Manager |
| JAN 26, 2015 | The Project was Tested by the respective Team Leaders and the Project Manager | Testing Phase (Module Testing) | -- |
| JAN 27, 2015 | The Project was Submitted to Other Project Leader of Other Project Group For Testing | Testing Phase (Acceptance Testing) | The Project of Other Team was Taken up by the Team for Testing |
| JAN 28-29, 2015 | The Errors Found were Removed | Debugging | The Project was complete for submission |
| JAN 30, 2015 | Final Submission of Project |  |  |

Appendix A

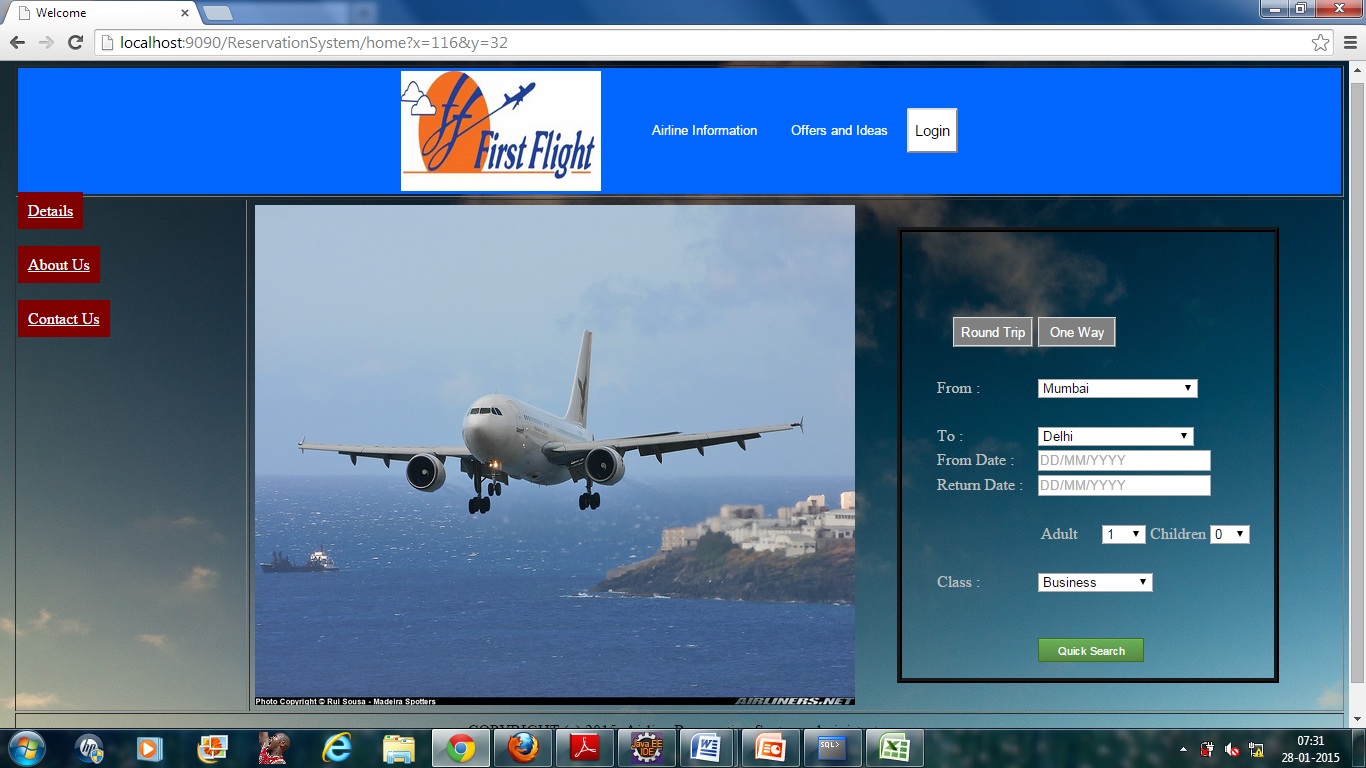
Entity Relationship Diagram

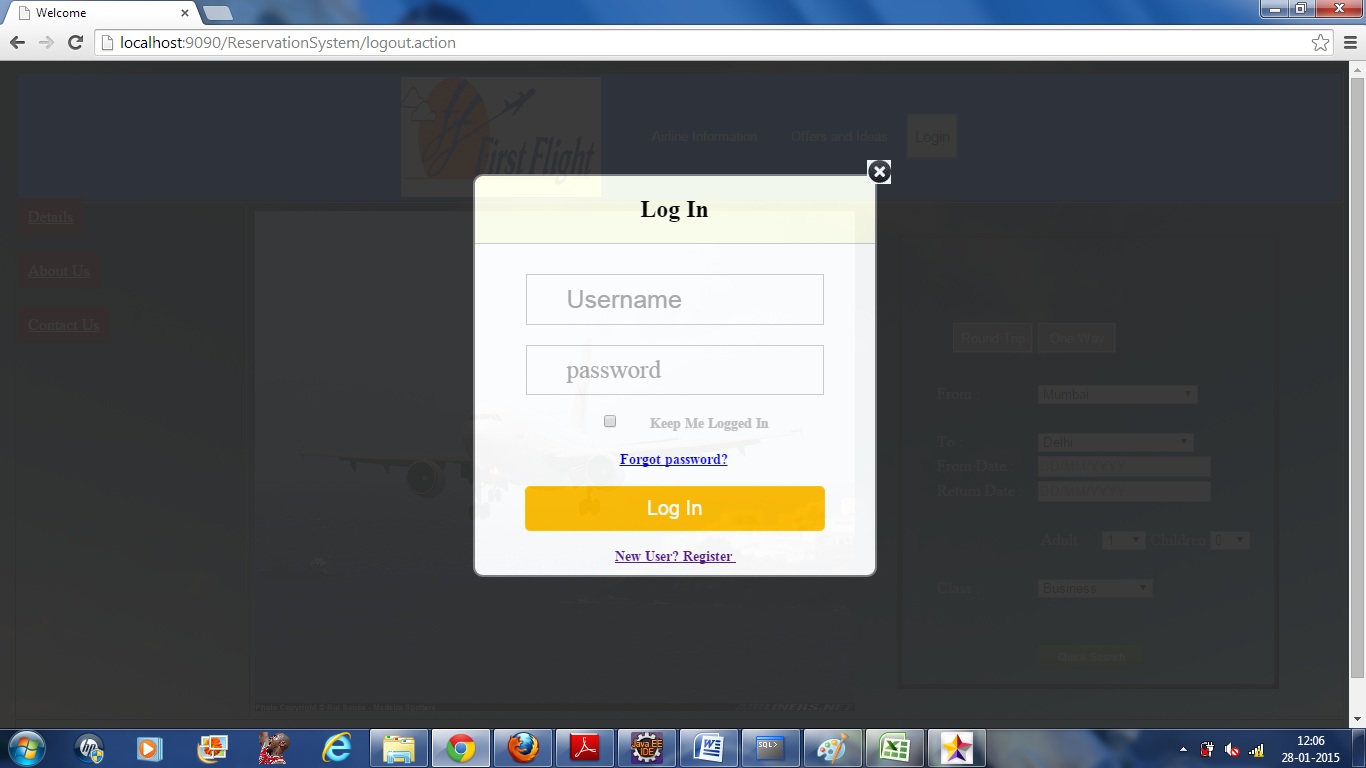
**Data Flow Diagram:**

****

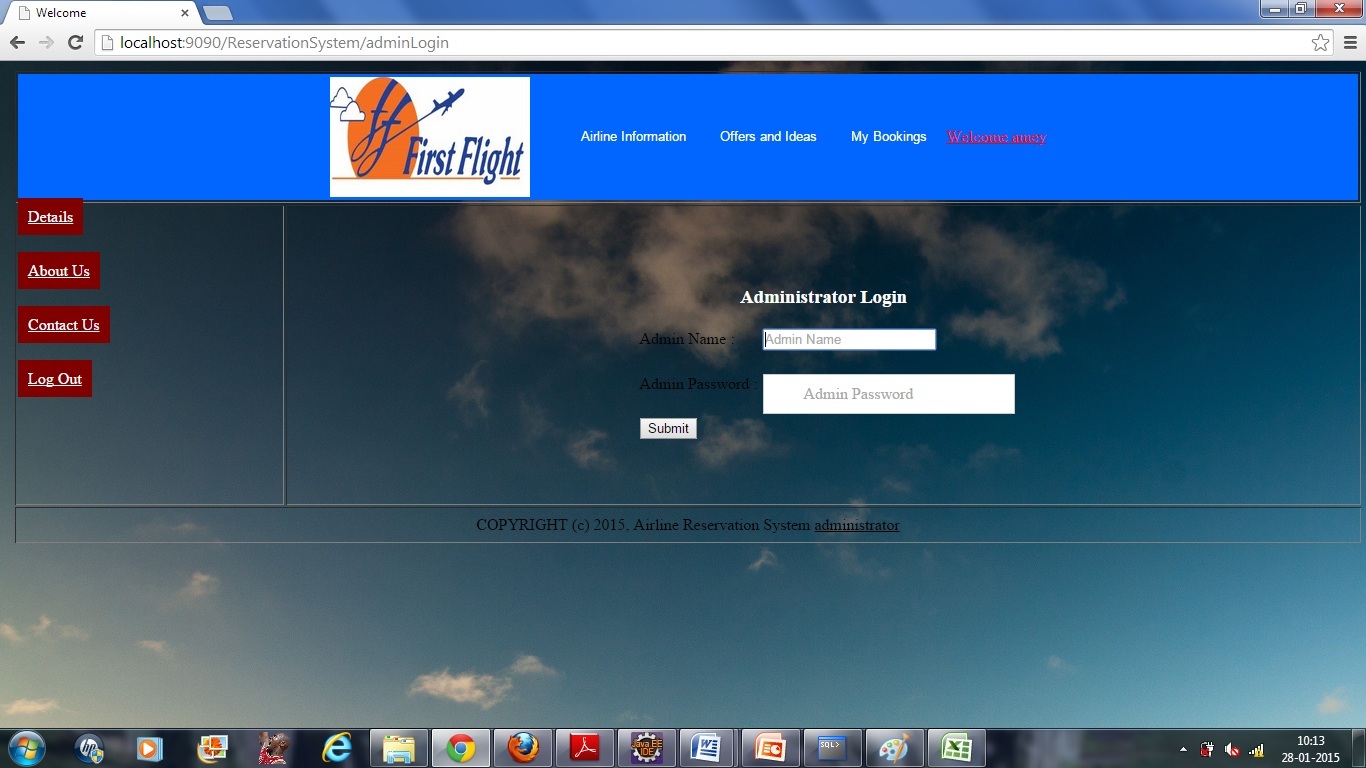
**Class Diagram**

Appendix B

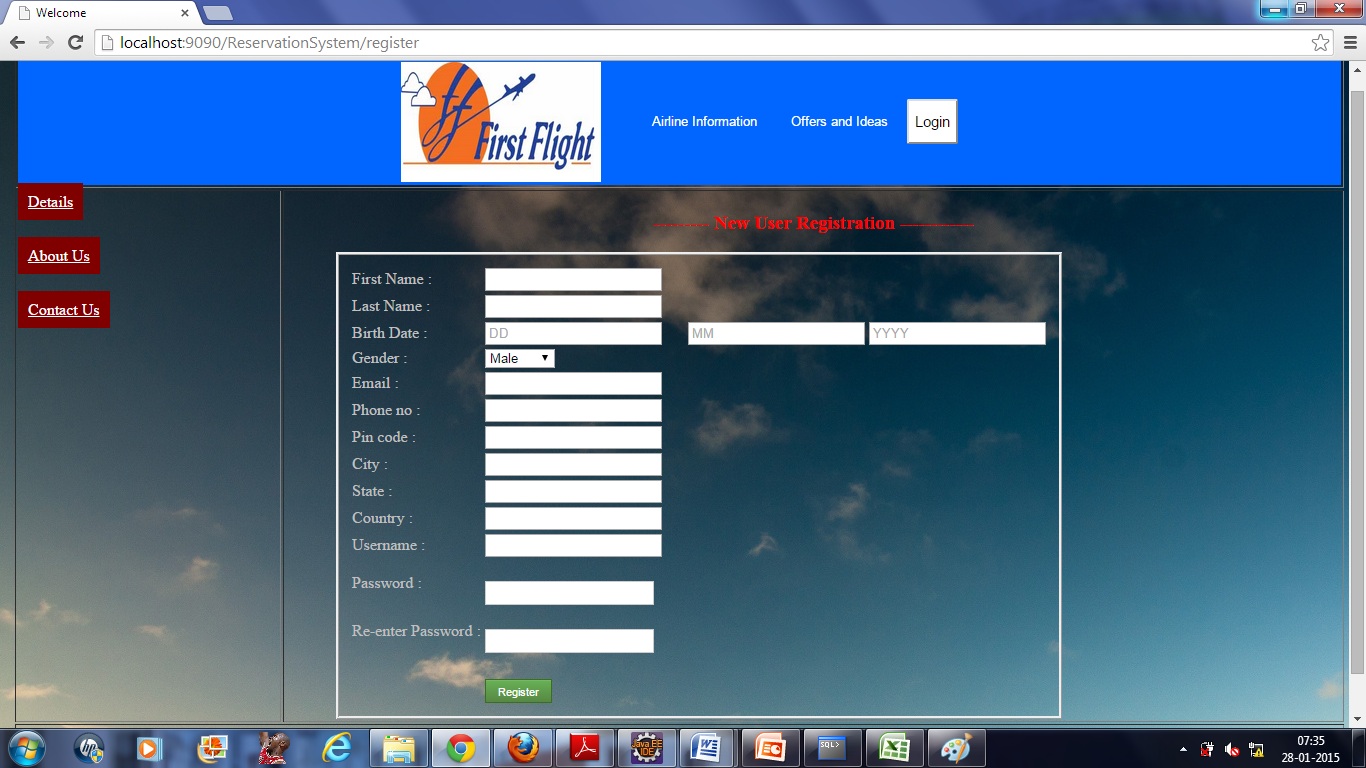
Homepage:

LoginPopup:

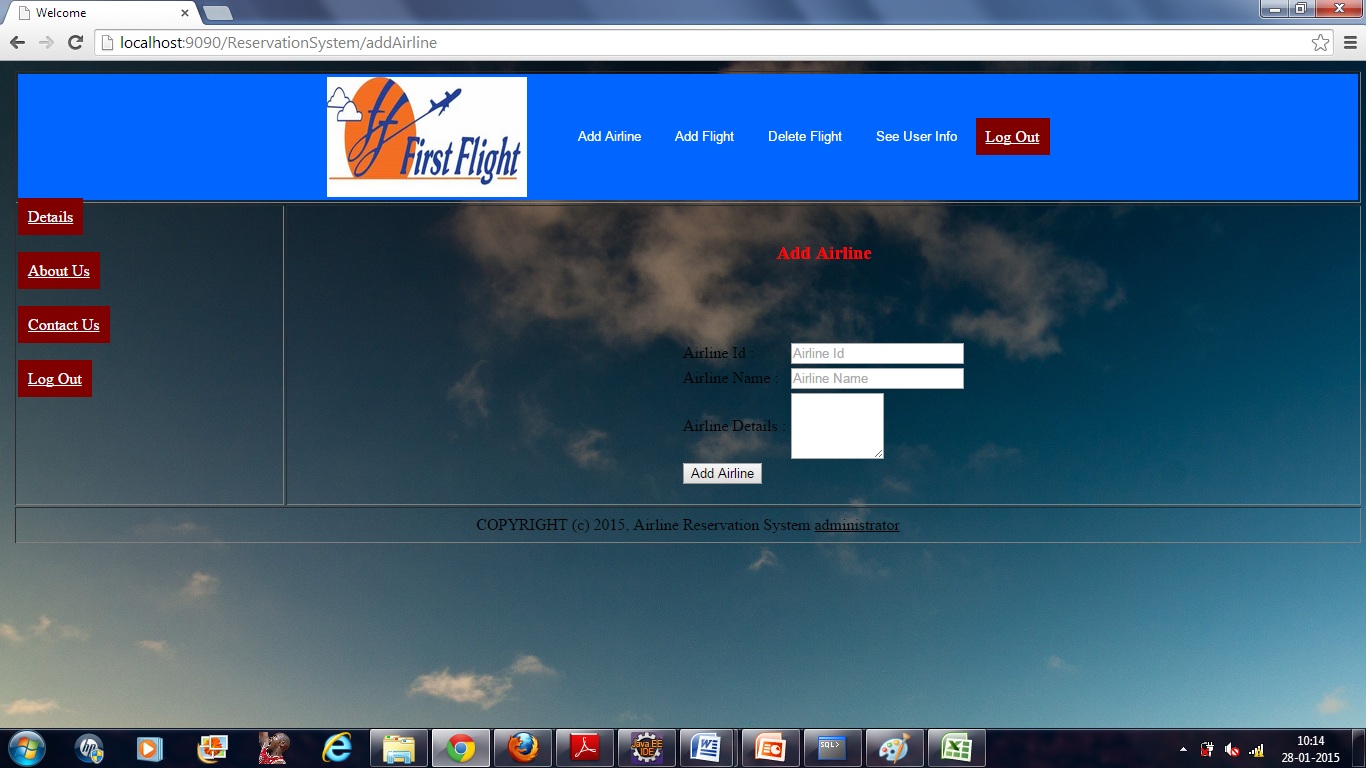
Administrators Login:

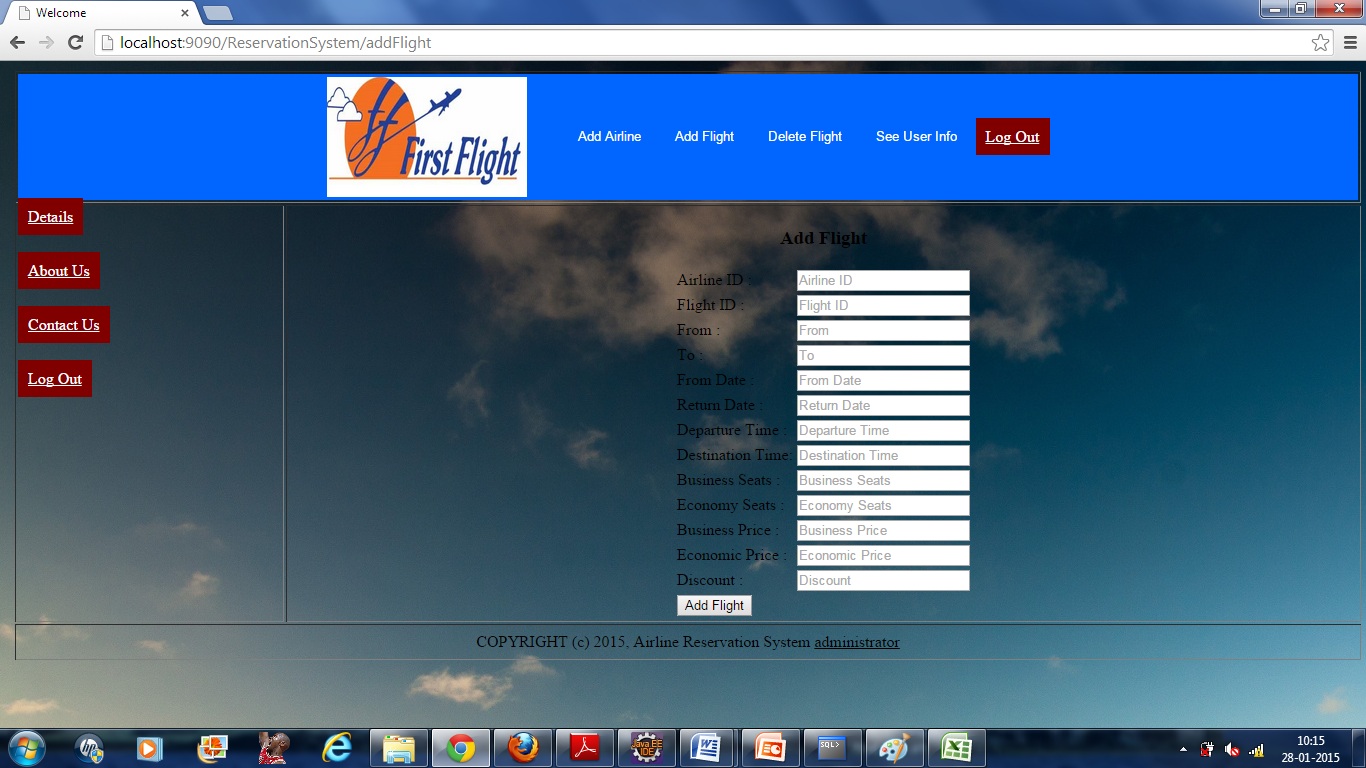


User Registration:

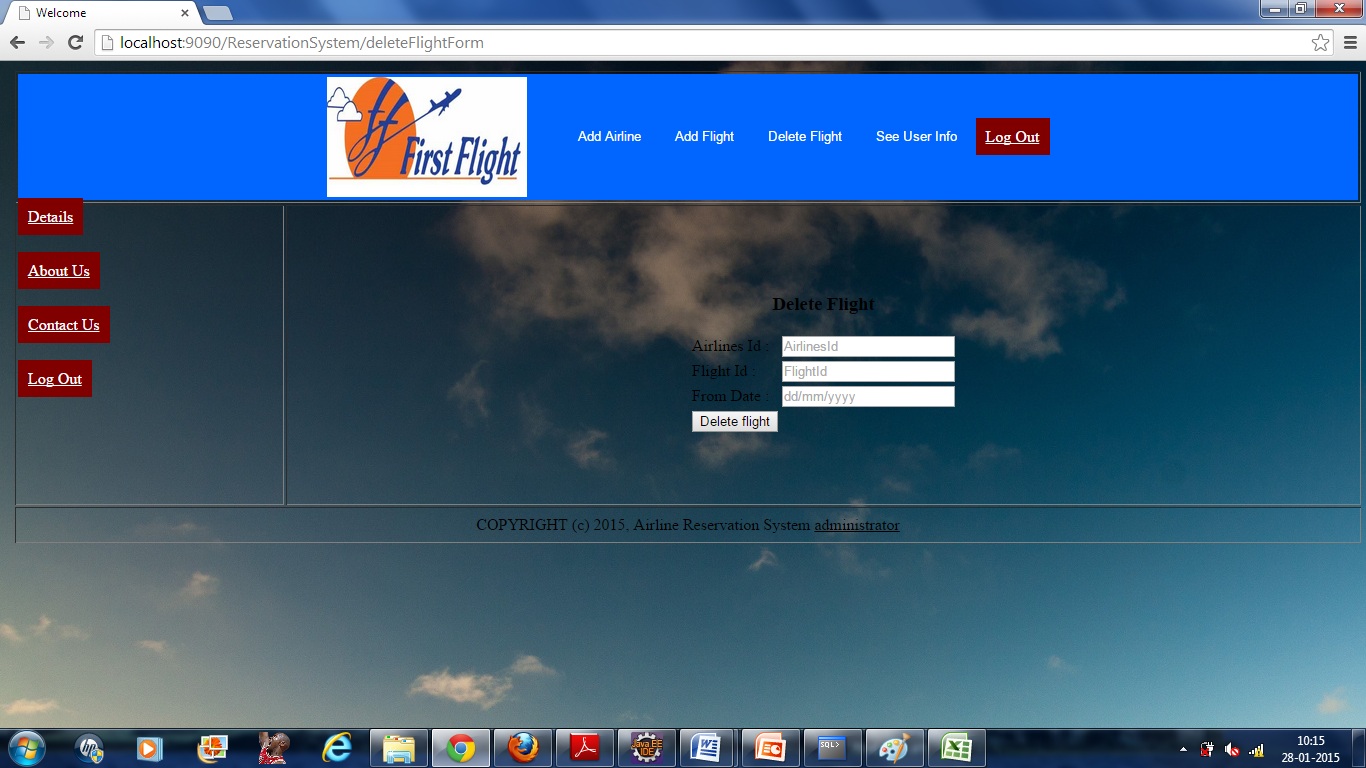


**Add Airline**

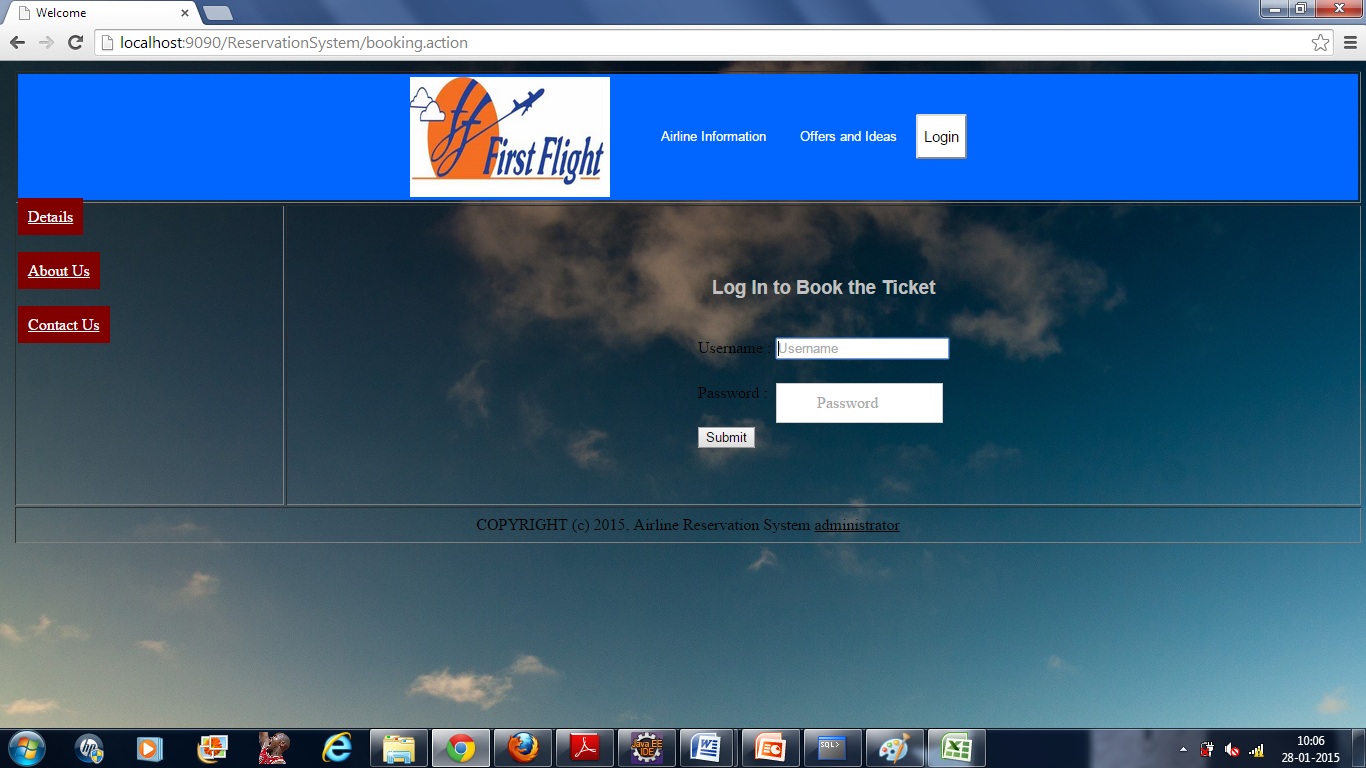


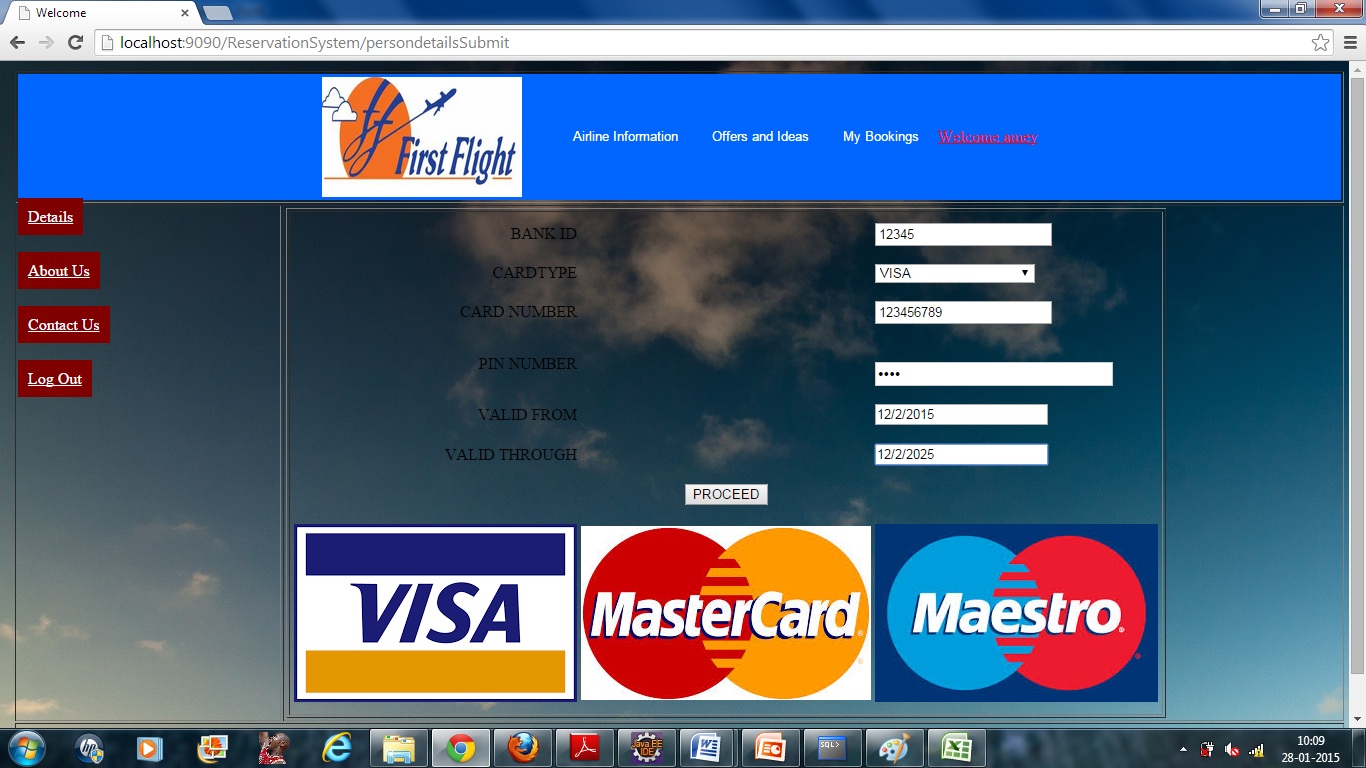
**AddFlight**

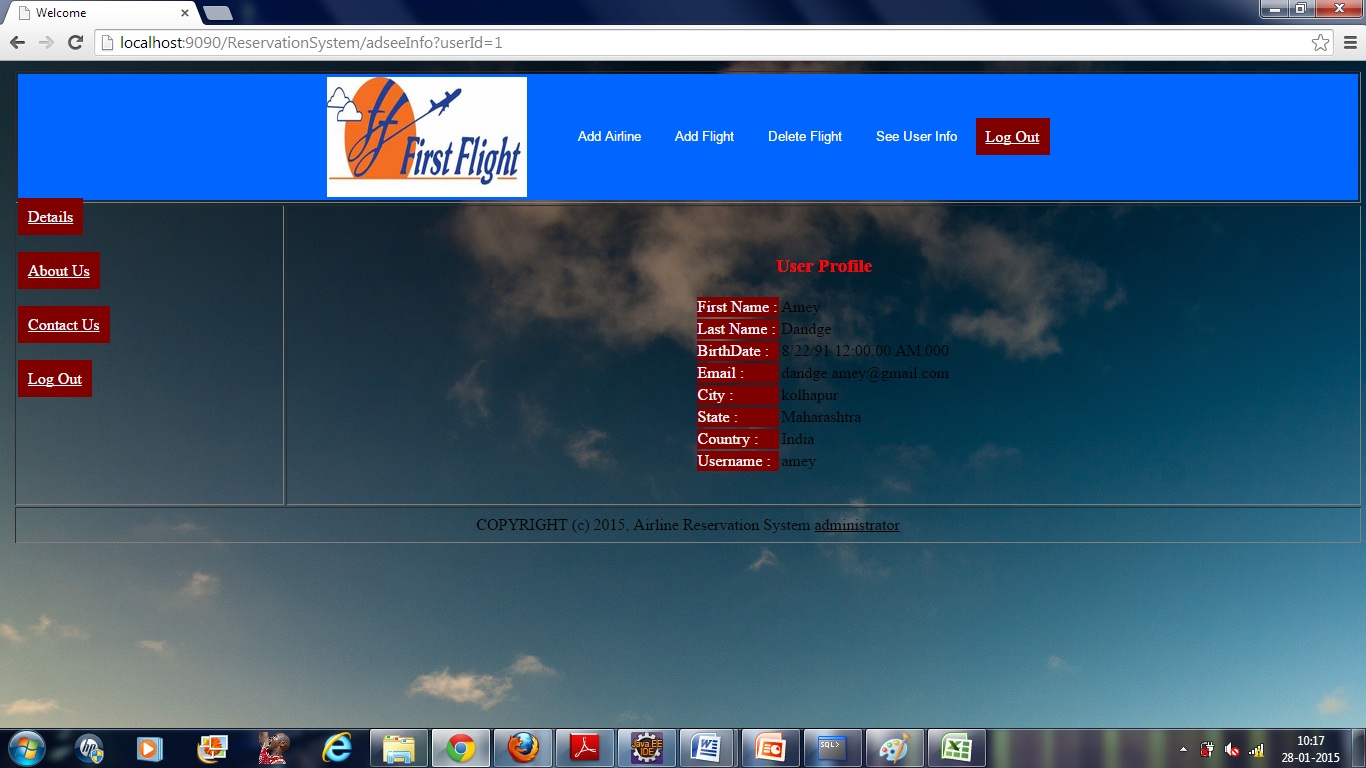
**Delete Flight**

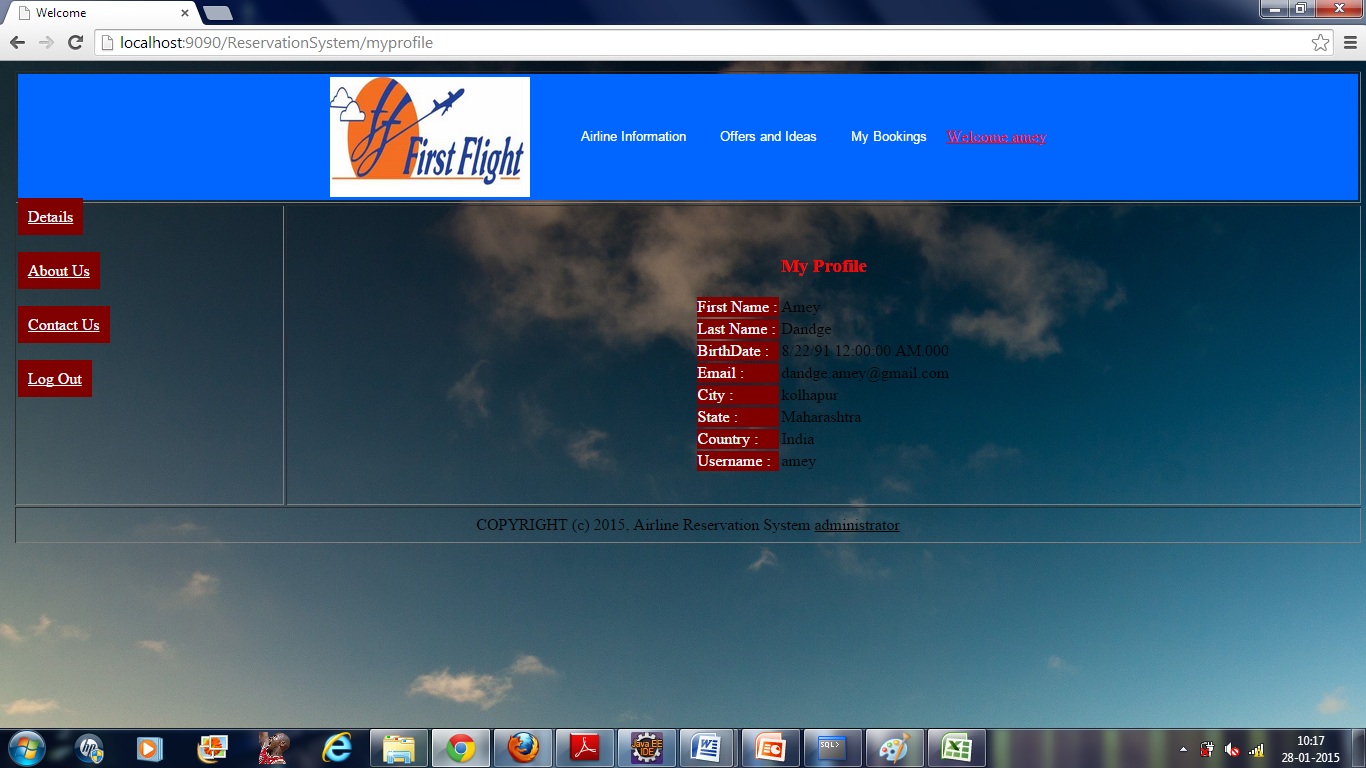
****

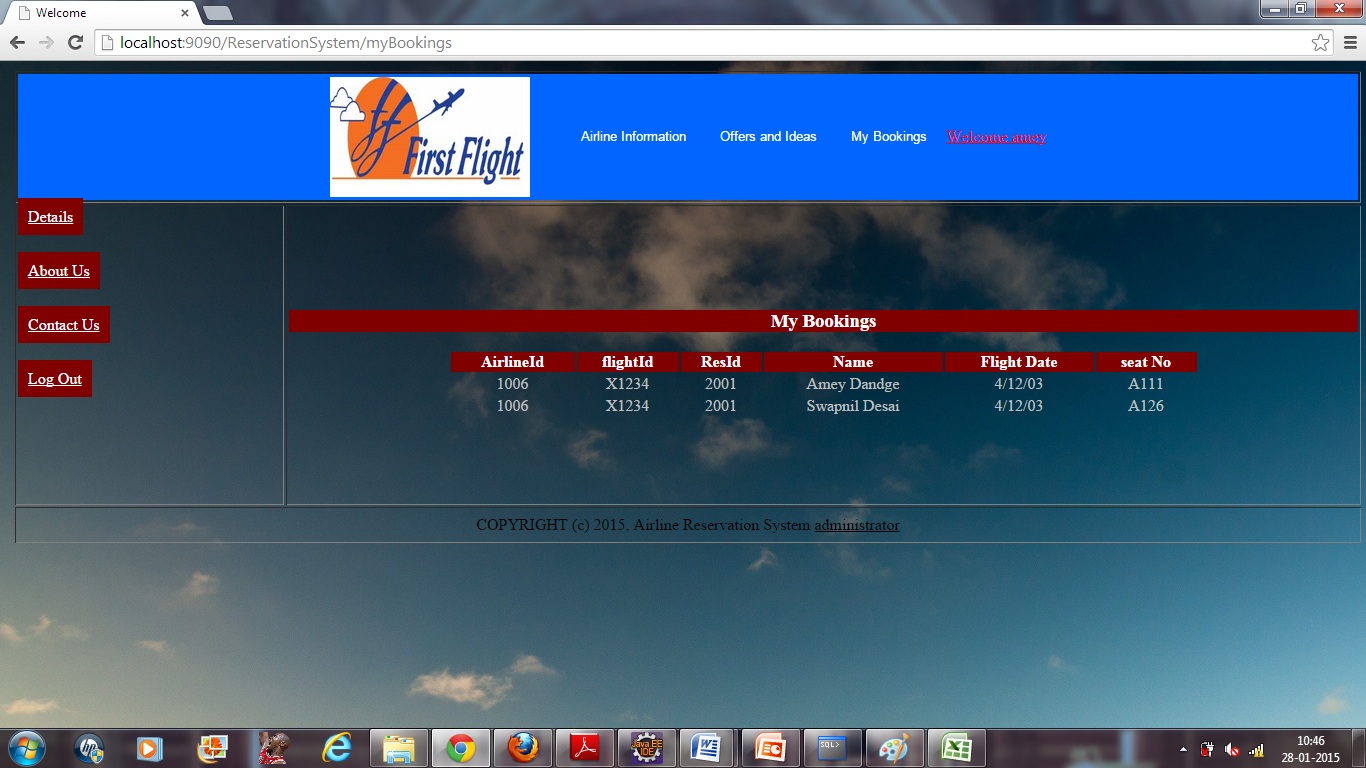
**QuickFlightSearch:**

LoginForBooking:

Banking:

UserProfileDetails:

MyProfileDetails: 

**BookingDetails:**

**7.REFERENCES:**

<http://www.google.com>

[http://](http://www.xml101.com:8081/xml/)www.airIndia.com

http://www.webdevelopersjournal.com/

http://www.lufthansa.com

http://www.w3.org

http://www.wikipedia.org

http://www.delta.com

http://www.priceline.com