# INTRODUCTION TO PROJECT

This project is a web based Online shopping system for an existing shop. An Online Shopping System which will allow merchants in developing countries to advertise and sell their goods on the internet. This would permit rural communities to make their wares available to the rest of the world the via world wide web.

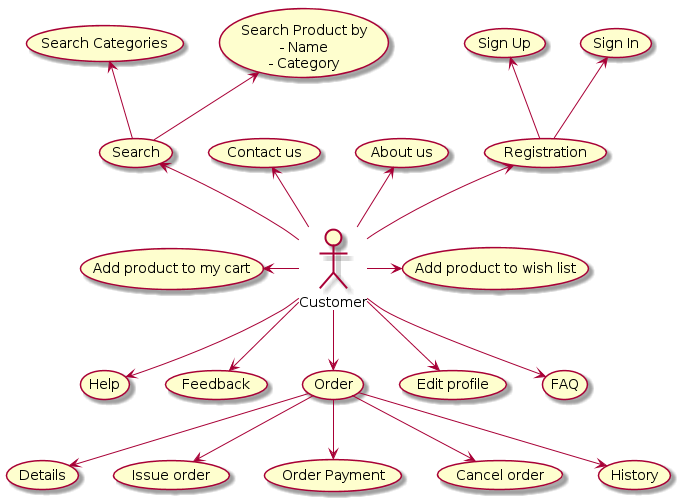
Online shopping is the process whereby consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the Internet. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products in the shop anywhere through internet. Thus the customer will get the service of online shopping and home delivery from his favorite shop.

Here we provided quick search facility which displays specific types of products. Then the system checks that particular product type and displays. Creating a shopping cart so that customer can shop multiple items they would like to see in the shop.

To place the order the system asks the customer to enter his details such bank details and placed the order and update the database. User can track all the information of delivery, payments etc. Online Shopping System also manage the delivery details, payment details, customer details.

**2.REQUIREMENTS**

**2.1 FUNCTIONAL REQUIREMENTS**

****

**2.1 Customer Account**

A new customer will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new customer by unblocking him.

A customer who wants to shop the products would have been given a user id and a password. This ‘personal information’ would be henceforth referred to as ‘profile’. Such a customer with a profile in DB-user shall be called a ‘registered user’. A registered customer will be able to see the products as well as order that product into the system. The user can view and edit the profile.

**2.2 Registration and creation of user profile**

The system shall require a customer to register for to see the products. It will ask the customer for the following information at the least – a user id, a password, first name, last name, address, phone number, email address. A customer must login with his user name and password to the system after registration. A customer can check the availability of product items.

****

* 1. **Product Search**

Here we provided Search Product facility for any customer to search particular product. This will provide customer an option for searching products and comparing their prices of all companies.

After logging in a customer, the system shall request him to enter the following details – product type. After the entering the product type, the system shall now access the related products. The system shall now ask the customer to enter the following details– fashion, mobiles, electronics, beauty, sports etc.

Having taken all the above input from the user, the system checks the quantity of products and displays if product available in stock.

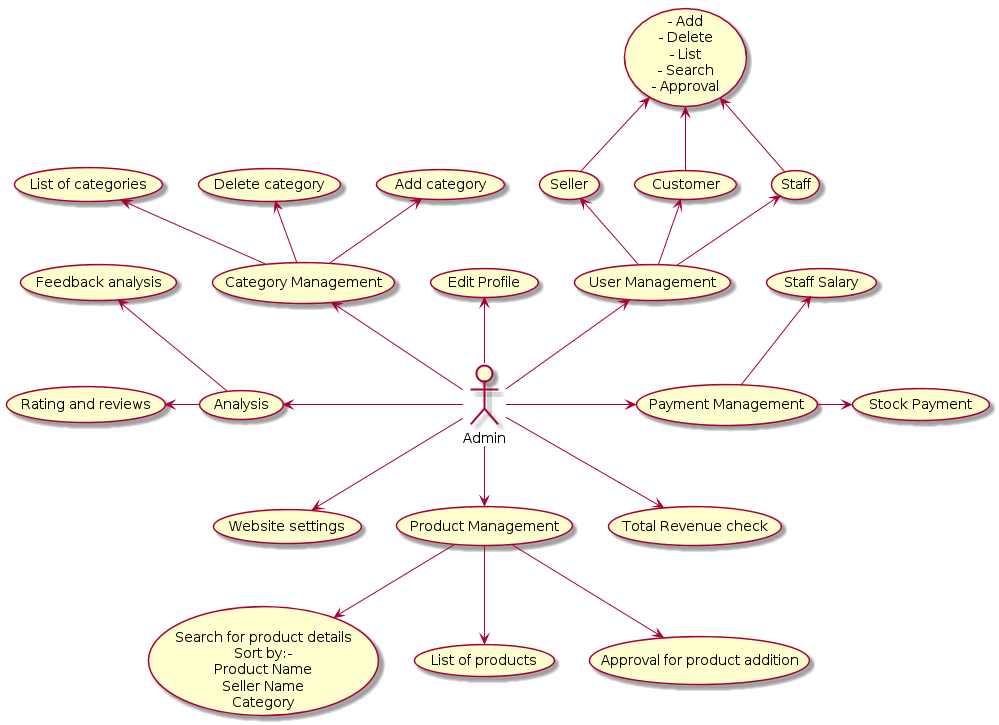
There can be several products of different seller. All of them will be listed. In case, the user has entered a range of price, the system shall display all the products for all those whose price in the range. There will be a ‘Add to Cart’, ‘Wishlist’ and ‘Shop Now’ button in below of every displayed product. Customer can remove an item from the cart by clicking remove.

* 1. **Purchasing Product**

After confirming the items in the cart the customer can submit the cart by providing a delivery address. On successful submitting the cart will became empty. The system will now ask the customer if he wishes to buy the product. If yes, then the system will ask for the customer address and bank details. Then system confirms customer can buy the product. If the customer the delivery date is far away from customer requirement then they can cancel the product order. Customer can track that order and see the detail information.

**2.5 View Booking History**

The system shall allow a user to view all information about his previous Orders.

****

Admin should be able to login, add seller, add customer, add store manager, add order manager and see user Information according to customer id. Admin has privileges to delete customer, seller and staff who was added.

The shopping cart project contains different kind of products. The products can be classified into different categories by name. Stored manager can add new products into the existing system with all its details including an image. Stored manager maintain the product details. Order manager maintain the all delivery person details for every order and notify best deals to customers according to their Wishlist .

* 1. **NON FUNCTIONAL REQUIREMENTS**

Non-functional requirements are not non-functional at all. Rather, they describe various quality factors, or attributes, which affect the functionality effectiveness. They do not exist in the abstract but only with respect to relevant functionality such as, usability, reliability, and maintainability. For instance, if the software doesn't satisfy relevant usability requirements for applicable functional usage, users can't use it appropriately and thus will not achieve the required functioning. Inadequate usability may cause errors which nullify the value of the functioning, such as miscalculating something. Usability difficulties could cause the user not to use (all) the functions necessary to achieve the value, perhaps because they're not able to employ the necessary functions. It may take so long and be so unpleasant to use the software that the user can't use it as much as is needed or even abandons its use entirely.

**2.2.1 Usability Requirement**

Usability is the ease of use and learn ability of a human-made object. The object of use can be a software-application, website, book, tool, MACHINE, process, or anything a human interacts with. A usability study may be conducted as a primary job function by a usability analyst or as a secondary job function by designers, technical writers, marketing personnel, and others. It is widely used in consumer electronics, communication, and knowledge transfer objects and mechanical objects such as a door handle or a hammer.

**2.2.2 Performance Requirements**

Performance requirement within system engineering, encompasses the set of roles, skills, activities, practices, tools, and deliverable applied at every phase of the systems development life cycle which ensures that a solution will be designed, implemented, and operationally supported to meet the non-functional requirements for performance. In our project basic performance requirements get maintain by using object oriented concepts such as major pillars of object oriented programming thought process are ‘Abstraction’, ‘Modularity’, ‘Encapsulation’, ‘Hierarchy’ and minor pillars are ‘Concurrency’, ‘Persistence’, ‘Typing’. These pillars keep the interfunctionalities for the project as ‘loosely coupled and highly cohesive’.

**2.2.3 Reliability Requirement**

Reliability Requirement is requirement that emphasizes dependency in the life cycle management of a product. Dependability, or reliability, describes the ability of a system or component to function under stated conditions for a specified period of time. Reliability requirement may also describe the ability to function at a specified moment or interval of time (Availability). Reliability is theoretically defined as the probability of success (Reliability=1- Probability of Failure), as the frequency of failures; or in terms of availability, as a probability derived from reliability, test ability and maintainability. Test ability, Maintainability and maintenance are often defined as a part of "reliability engineering" in Reliability Programs. Reliability plays a key role in the cost-effectiveness of systems. In this project, to achieve the reliability requirement of the software the various programming layers have to reliable on each other inclusively. For example in our project, we have database, POJO layer, DAO layer, Controller layer, Service layer. Among these, eachlayer has to relay on other layer. For example, POJO layer has dependency on database, for the implementation of DAO layer it has to take some help of POJO layer and so on. This activity shows the reliability requirements of our project.

**2.3 Constraint**

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

**2.4 Other Requirements:**

* **Hardware Interfaces**

The minimum hardware requirement for the system is 128 MB of RAM, 20 GB Hard-disc drive.

* **Software Interfaces**

Operating System: Windows, Linux etc.

Front End: ReactJs

Back End: Spring-boot

Database: MySql

**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 | userLd | Number | 4 | 0 |
| 0 | Email | Varchar2 | 260 | 0 |
| 0 | First\_name | Varchar2 | 260 | 0 |
| 0 | Last\_name | Varchar2 | 260 | 0 |
| 0 | Mobile | Number | 260 | 0 |
| 0 | Password | Varchar2 | 260 | 0 |
| 0 | Reg\_date | Date | 30 | 0 |
| 0 | role | Varchar2 | 255 | 0 |

## **Table2: Administrator\_Login**

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

# Table3: Seller

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Seller\_id | Number | 5 | 0 |
| 0 | Address | Varchar2 | 1500 | 0 |
| 0 | Brand\_name | Varchar2 | 50 | 0 |
| 0 | contact | Varchar2 | 30 | 0 |

### Table4: Products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | product\_id | Number | 5 | 0 |
| 3 | avg\_rating | double | 10 | 0 |
| 0 | description | Varchar2 | 260 | 0 |
| 0 | discount | double | 260 | 0 |
| 3 | picture | longblob | 30 | 0 |
| 0 | price | Double | 8 | 0 |
| 0 | product\_name | Varchar2 | 40 | 0 |
| 0 | quantity | Number | 15 | 0 |
| 0 | category\_id | Number | 3 | 0 |
| 0 | seller\_id | Number | 3 | 0 |

**Table5: Categories**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | category\_id | Number | 5 | 0 |
| 0 | cat\_name | Varchar2 | 25 | 0 |
| 3 | description | Varchar2 | 15 | 0 |
| 0 | picture | longblob | 20 | 0 |

### Table6:User Address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | adr\_id | Number | 5 | 0 |
| 0 | Area | Varchar2 | 255 | 0 |
| 0 | City | Varchar2 | 15 | 0 |
| 0 | Country | Varchar2 | 25 | 0 |
| 0 | Mobile | Number | 5 | 0 |
| 0 | Pin\_code | Number | 25 | 0 |
| 0 | State | Varchar2 | 10 | 0 |
| 0 | User\_id | Number | 5 | 0 |

### Table7:Cart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | cart\_id | Number | 5 | 0 |
| 0 | Quantity | Number | 25 | 0 |
| 0 | Product\_id | Number | 5 | 0 |
| 0 | User\_id | Number | 5 | 0 |

### Table8:User Address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Bnk\_id | Number | 5 | 0 |
| 0 | Acc\_no | Varchar2 | 255 | 0 |
| 0 | Holder\_name | Varchar2 | 15 | 0 |
| 0 | Ifsc-code | Varchar2 | 25 | 0 |
| 0 | User\_id | Number | 5 | 0 |

**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.**

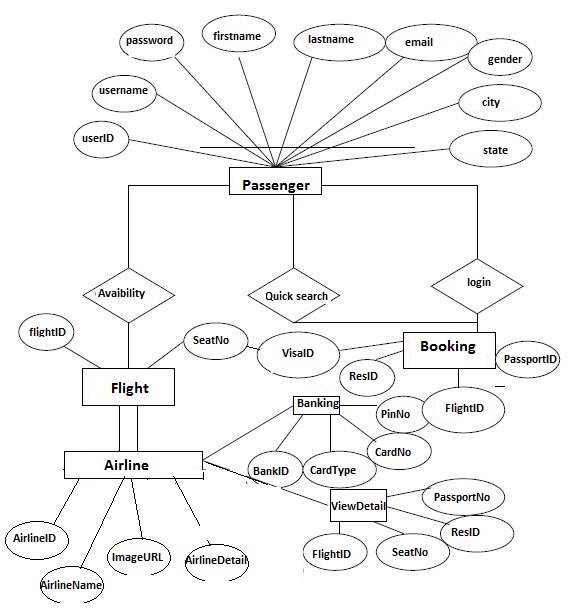
**GENERAL TESTING:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  |  |

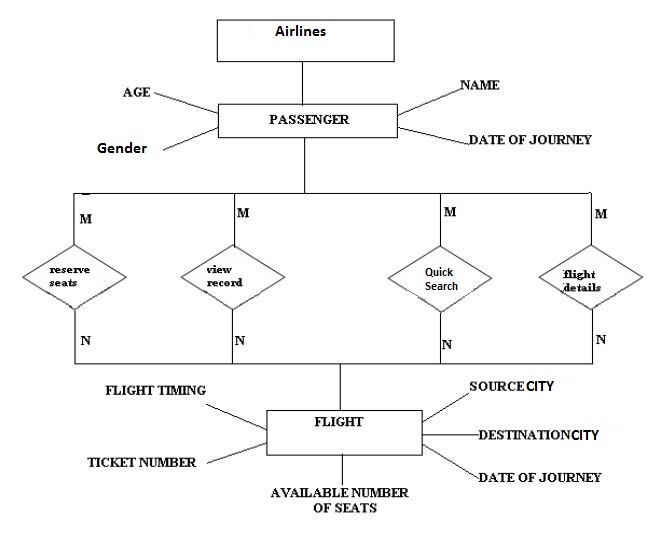
**6. PROJECT MANAGEMENT RELATED STATISTICS**

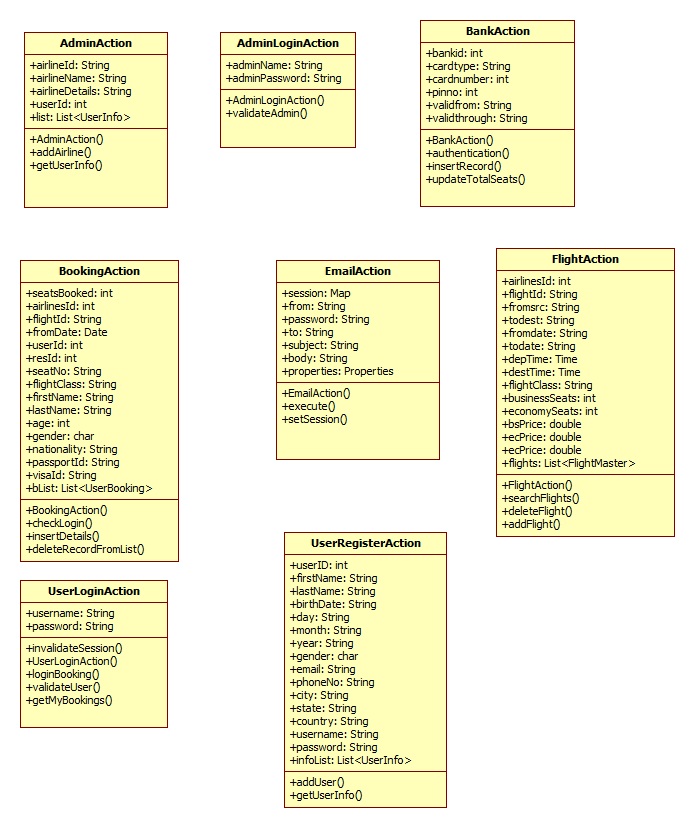
|  |  |  |  |
| --- | --- | --- | --- |
| **DATE** | **WORK PERFORMED** | ****SLC Phase**** | **Additional Notes** |
| JAN 12,2015 | Project Allotment and User Requirements Gathering | Feasibility Study | Our team met the client Mr. Nitinkudale (CEO, SIIT Pune) to know his requirements. |
| JAN 17,2015 | 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

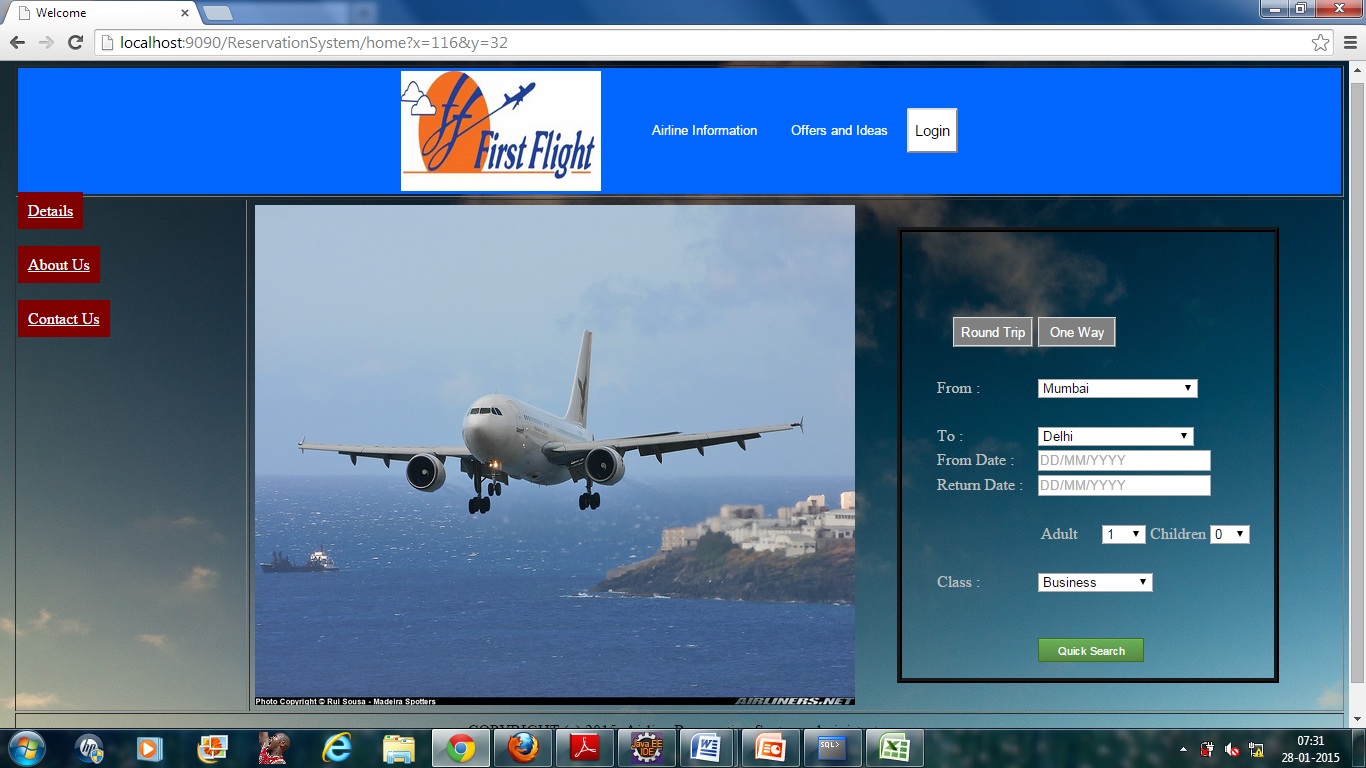
Entity Relationship Diagram

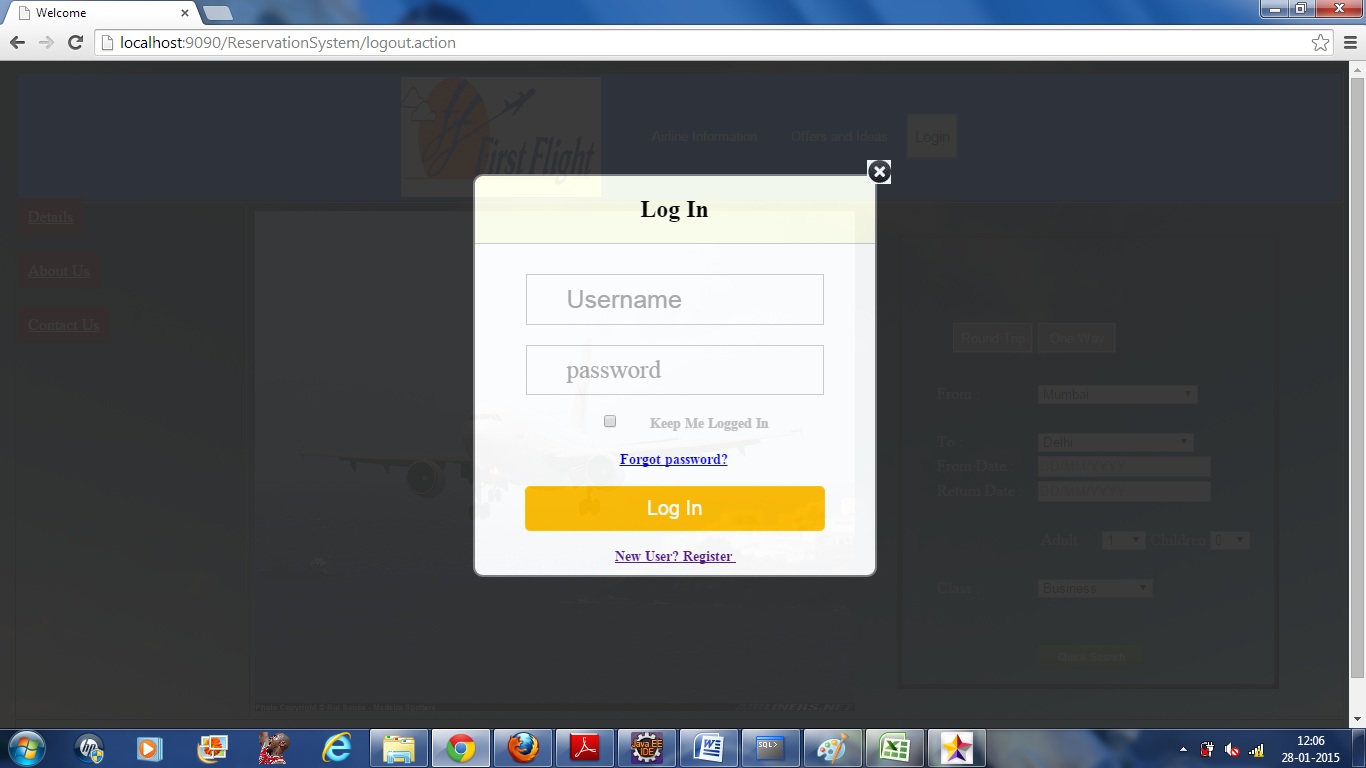
**Data Flow Diagram:**

****

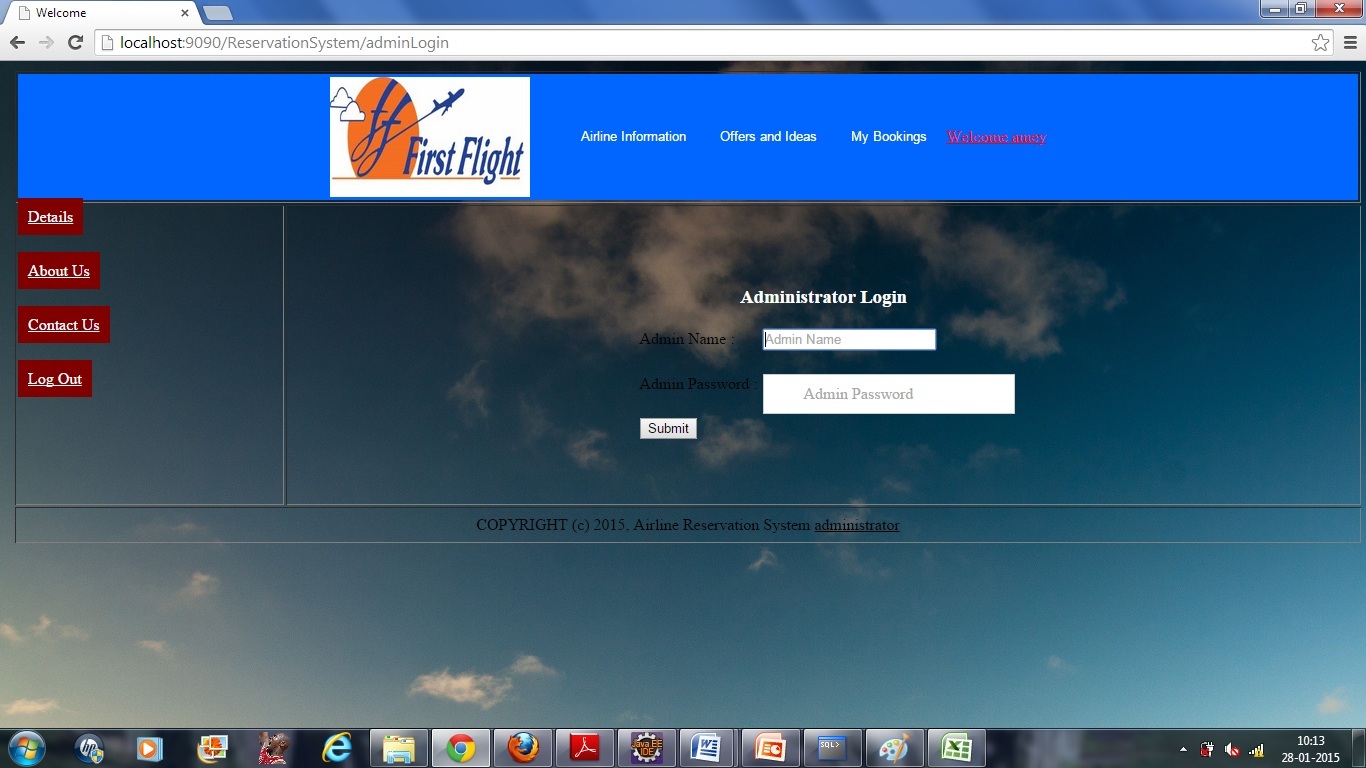
**Class Diagram**

Appendix B

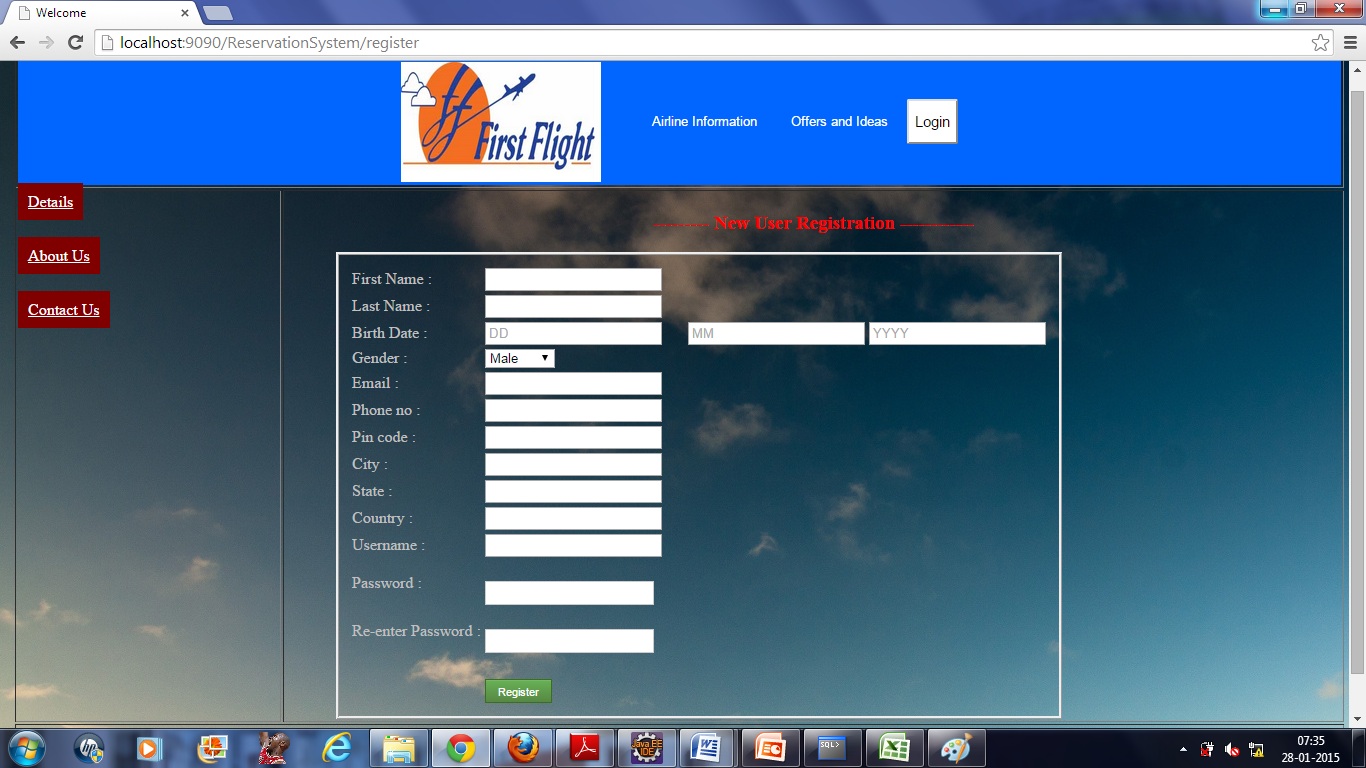
Homepage:

LoginPopup:

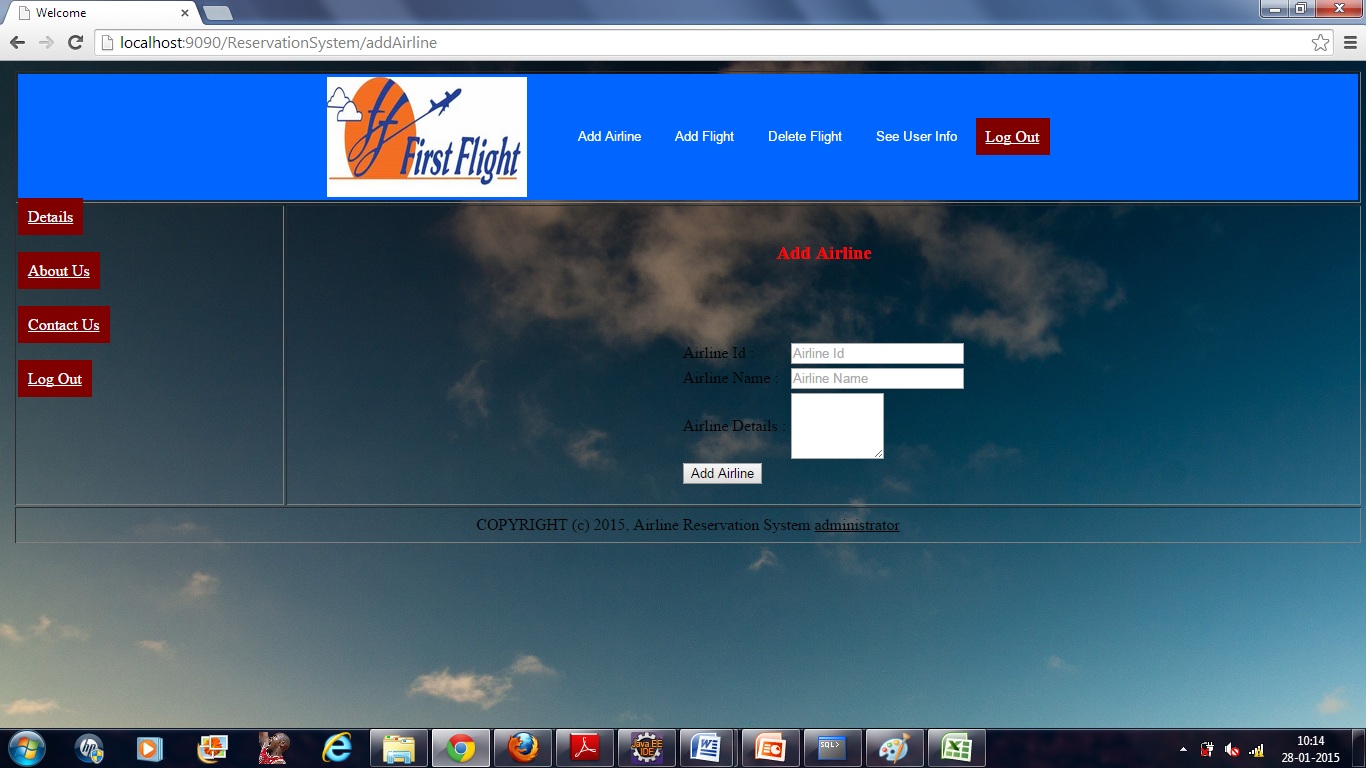
Administrators Login:

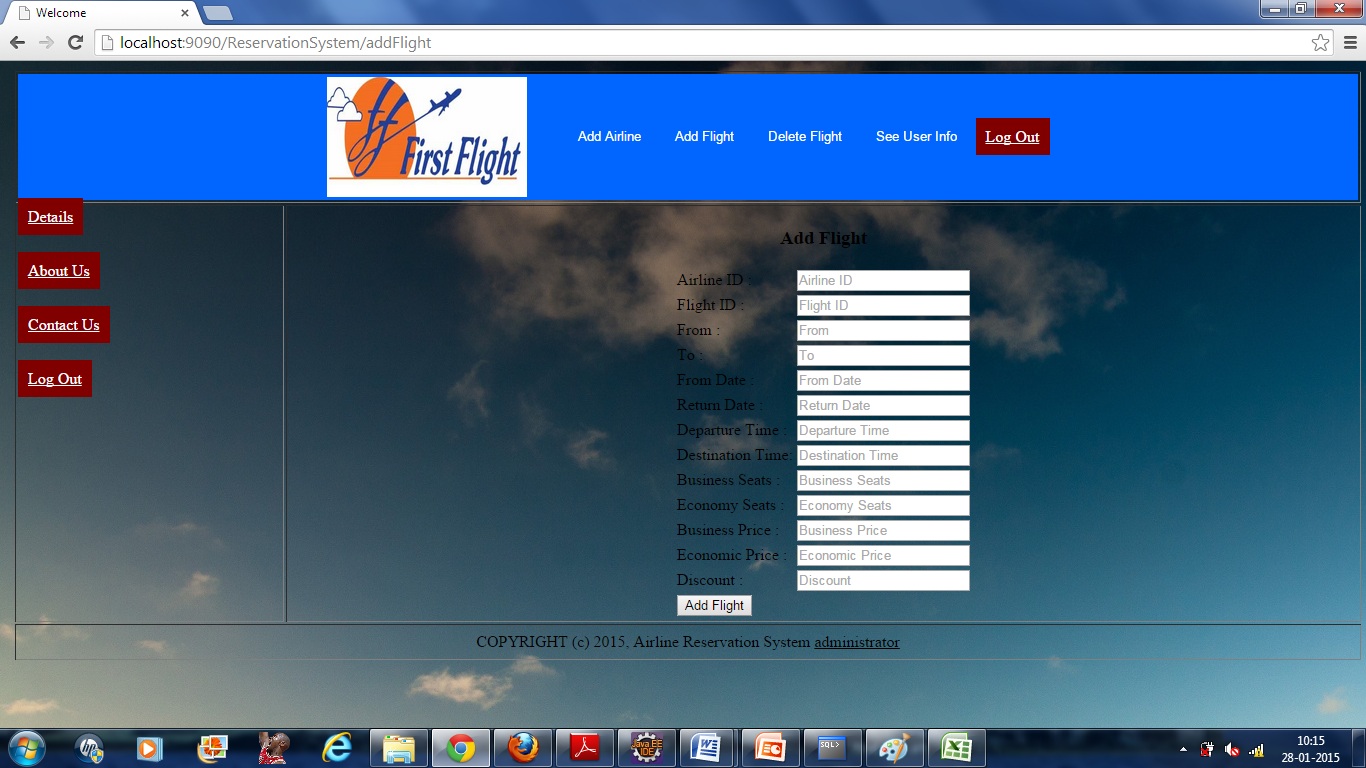


User Registration:

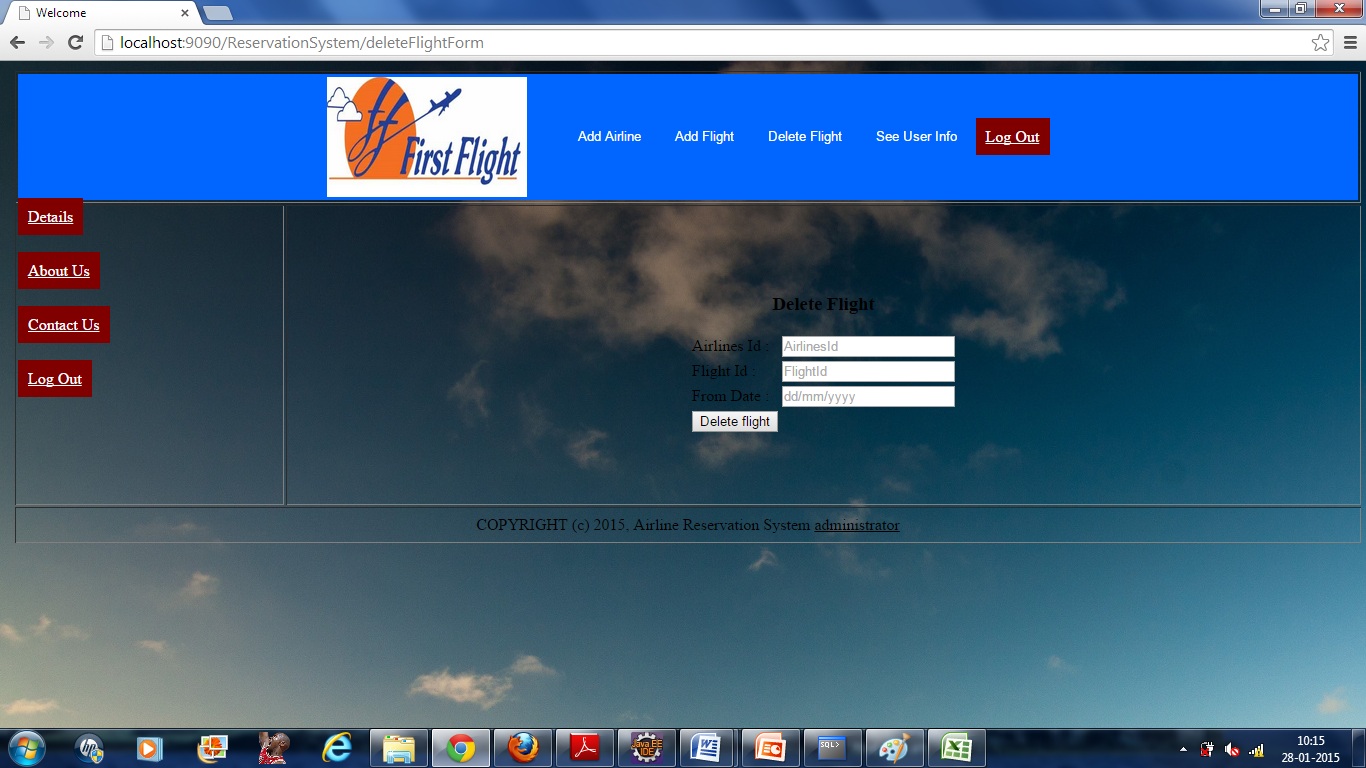


**Add Airline**

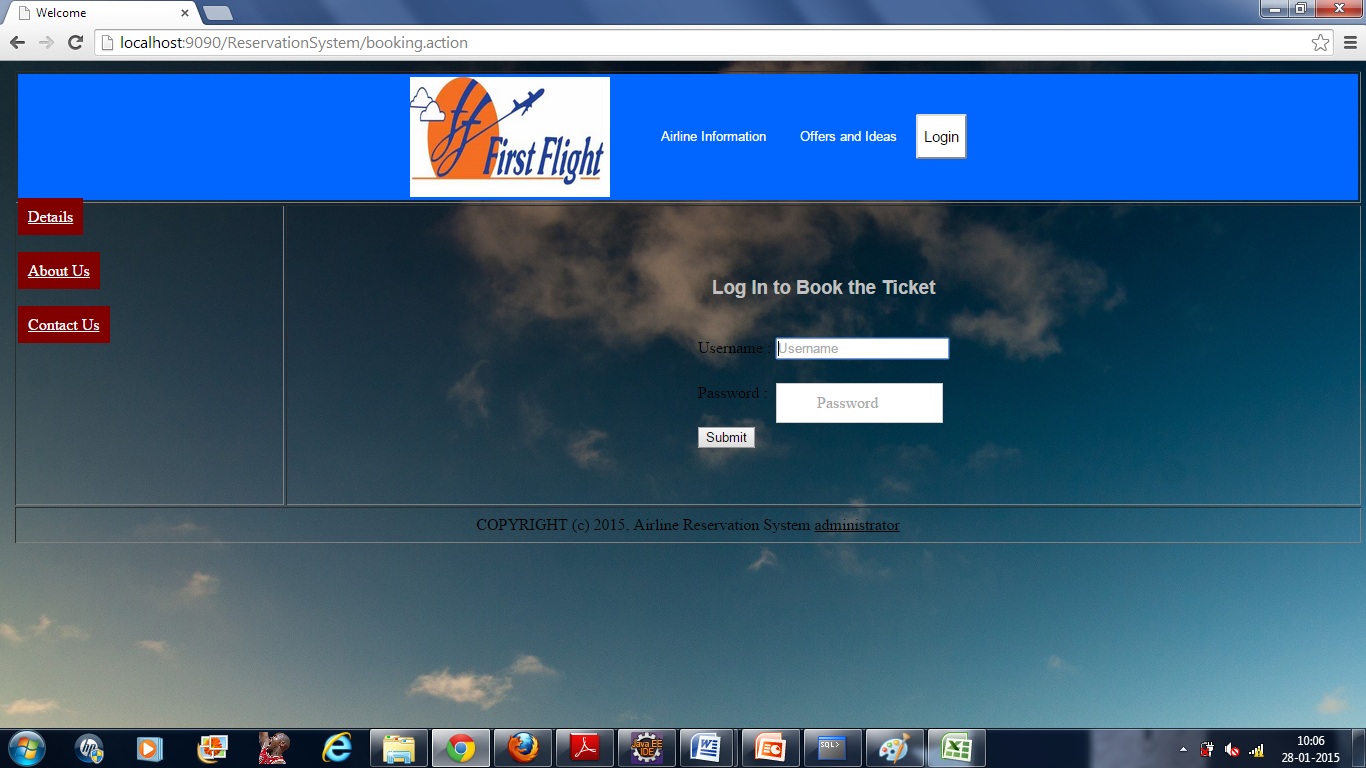


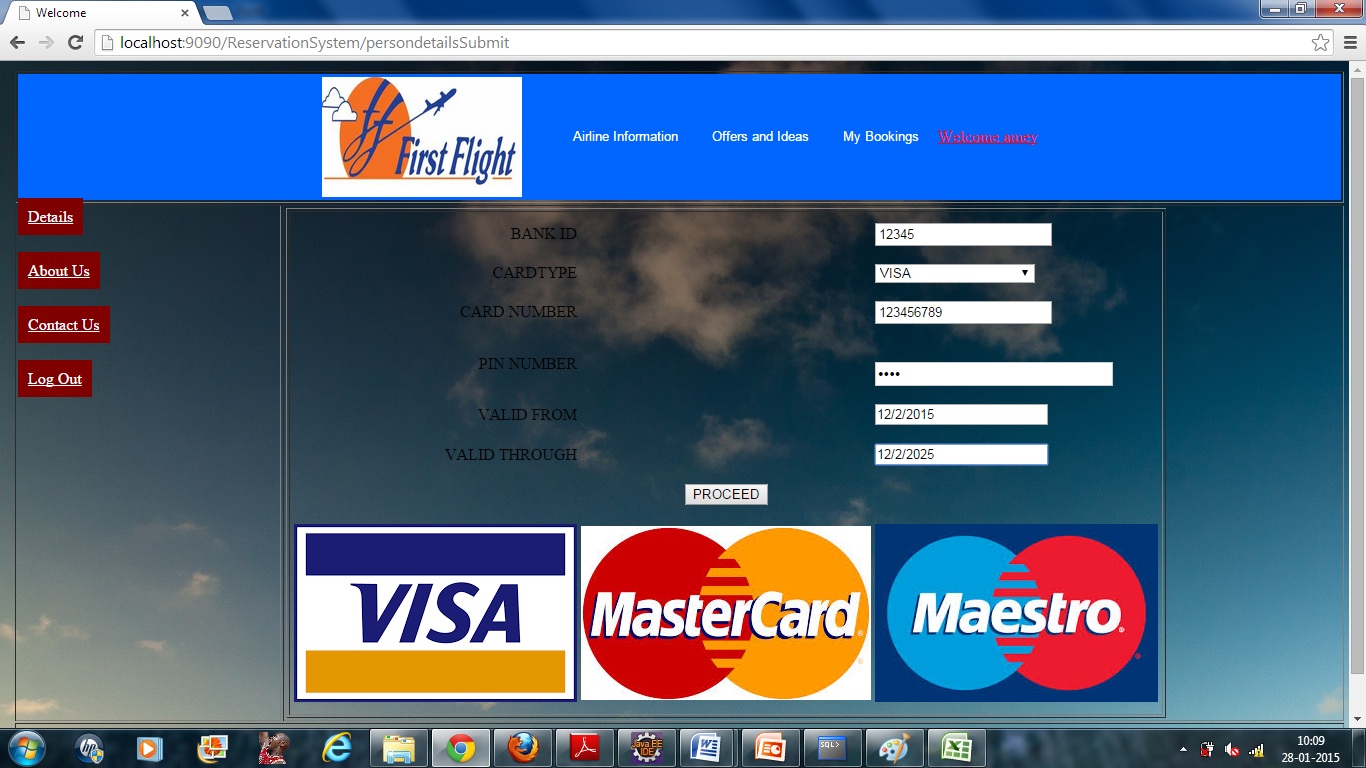
**AddFlight**

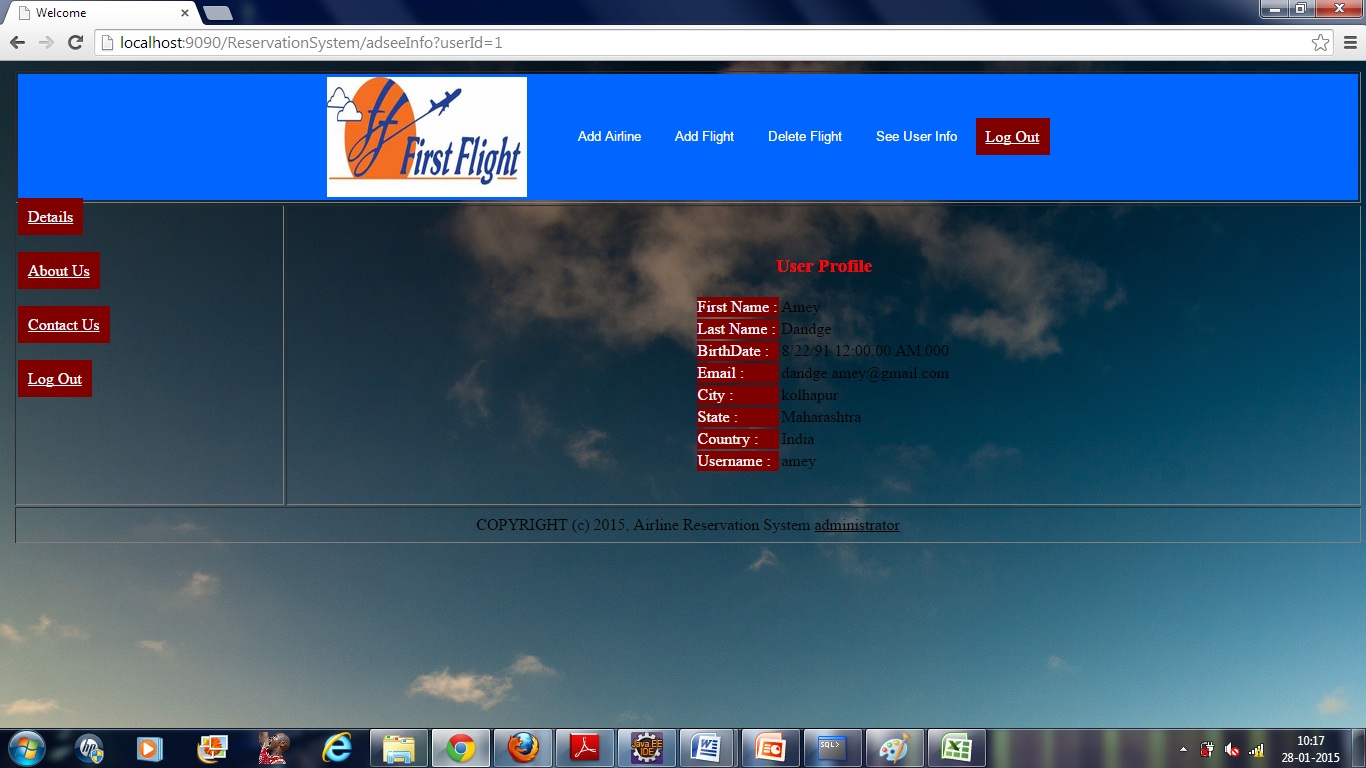
**Delete Flight**

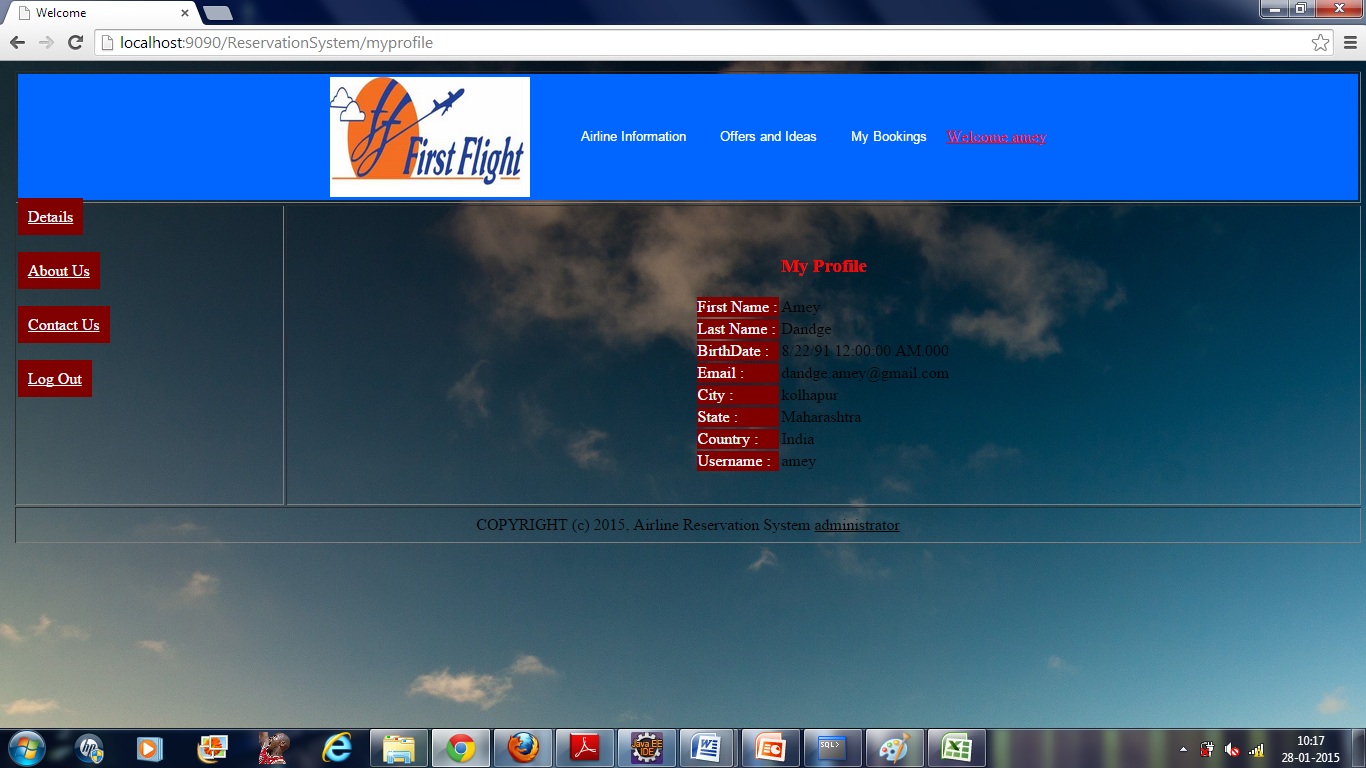
****

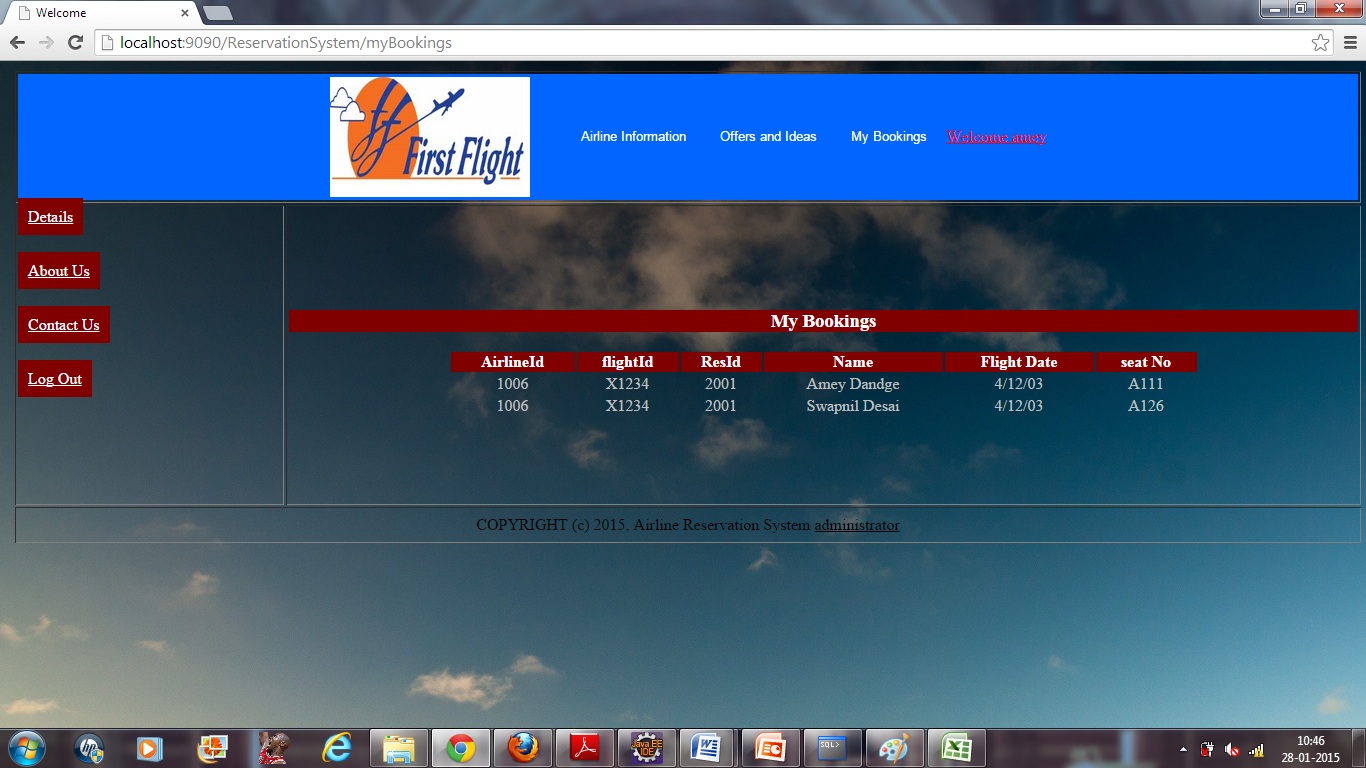
**QuickFlightSearch:**

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