CHAPTER 1

INTRODUCTION

Airline reservation systems were first introduced in the late 1950s as relatively simple standalone systems to control flight inventory, maintain flight schedules, seat assignments and aircraft loading. The modern airline reservation system is comprehensive suite of products to provide a system that assists with a variety of airline management tasks and service customer needs from the time of initial reservation through completion of the flight. One of the most common modes of travel is traveling by air. Customers who wish to travel by air nowadays have a wide variety of airlines and a range of timings to choose from. Nowadays competition is so fierce between airlines that there are lot of discounts and a lot of luxuries given to customers that will give an edge to that particular airline. The World Wide Web has become tremendously popular over the last four years, and currently most of the airlines have made provision for online reservation of their flights. The Internet has become a major resource for people looking for making reservations online without the hassle of meeting travel agents. My Project intends to serve these purposes. It intends to check all the available airline databases and return a string of results, which can help them in their travel plans.

1.1 Overview of the Project

The main purpose of this software is to reduce the manual errors involved in the airline reservation process and make it convenient for the customers to book the flights as when they require such that they can utilize this software to make reservation, modify reservations or cancel a particular reservation.

The name of the software is “AIRLINE RESERVATION SYSTEM”. This software provides options for viewing different flights available with different timings for a particular date and provides customers with the facility to book a ticket, modify or cancel a particular reservation but it does not provide the customers with details of cost of the ticket and it does not allow the customer to modify a particular part of his reservation and he/she can modify all details.

1.2 EXISTING SYSTEM:

The effectiveness of the system depends on the way in which the data is organized. In the existing system, much of the data is entered manually and it can be very time consuming. When records are accessed frequently, managing such records becomes difficult. Therefore organizing data becomes difficult. The major limitations are:

- Modifications are complicated
- Much time consuming
- Error prone
- Unauthorized access of data

1.3 PROPOSED SYSTEM:

The proposed system is better and more efficient than existing System by keeping in mind all the drawbacks of the present system to provide a permanent to them.

The primary aim of the new system is to speed up the transactions. User friendliness is another peculiarity of the proposed system. Messages are displayed in message boxes to make the system user friendly. Another important feature of the proposed system is the data security provided by the system. The main objectives of the proposed system are:

- Complex functions are done automatically
- Processing time can be minimized
- Simple and easy to manage
- Chances of errors reduced
- Faster and more accurate than the existing system
- Easy for handling reports

The proposed system is complete software for Airline Reservation System, Which is more efficient, reliable, faster and accurate for processing.

CHAPTER 2

SYSTEM REQUIREMENT

2.1 HARDWARE REQUIREMENT

Processor :Intel core duo processor

RAM : 4GB

Secondary Memory : 500GB

2.2 SOFTWARE REQUIREMENT

IDE :NET BEANS

Front End : MYSQL

Operating System :Ubuntu/windows/ java

CHAPTER 3

SYSTEM ANALYSIS

3.1 Front End (Java)

➢ **Overview of Java:**

Java is a powerful but lean object oriented programming language. It has generated a lot of excitement because it makes it possible to program for Internet by creating applets, programs that can be embedded in wed page.

But Java is more than programming language for writing applets. It is becoming so popular that many people believe it will become standard language for both general purposes and Internet programming.

➢ **Components of Java:**

Java is actually a platform consisting of three components:

- Java Programming Language.
- Java Library of classes and interfaces.
- Java Virtual Machine.

➢ **Java is Object Oriented:**

The Java programming language is object oriented, which makes program design focus on what you are dealing with rather than on how you are going to do something.

➢ **Java's exciting features are:**

- Ease in code correction.
- Garbage collection.
- Absence of pointers.
- Java is extensible.
- Java is secure..

- **Library Classes:**

The Java platform includes an extensive class library so that programmers can use already existing classes, as it is, create subclasses to modify existing classes or implement interfaces and augment the capabilities of classes.

- **Interfaces:**

Interfaces is also merely like class. Interfaces also contain data members and functions. But the main difference is that in an interface, fields must be constants, and methods are just prototypes with no implementations.

- **Packages:**

A package is a collection of related java classes and interfaces. The following list, gives examples of some java packages and what they cover.

JAVA.SQL- The JDBC API, classes and interfaces that access database and send SQL. In Java, packages serve as basis for building other package.

- **JDBC**
  - JDBC is a java TM API for executing SQL statements.
  - It consists of a set of classes and interfaces written in the java programming language that makes it easy to send SQL statements to virtually any relational databases.
  - JDBC (Java Database Connectivity) is a front end tool for connecting server to ODBC in that respect..

The combination of java and JDBC lets a programmer write it once and run it anywhere.

- **Requirements to use JDBC**
  - To use JDBC we need a basic knowledge of database and SQL.
  - We need the jdk1.1 (Java Development Kit 1.1 available Java Soft's website) or a version of java since jdk1.1 and above come bundled with JDBC software.
  - A back-end database engine for which a JDBC driver is available. When JDBC drivers area not available JDBC-ODBC bridge drivers are used to access the database through ODBC.

➢ **JDBC Drivers :**

The JDBC API founds in java.sql package, consists only a few concrete classes.

JDBC drivers are available for most database platforms, from a number of vendors and in a number of different flavors. There are four driver categories:

- **Type1-JDBC-ODBC Bridge Driver**

Type1 drivers use a bridge technology to connect a java client to an ODBC database service. Sun's JDBC-ODBC bridge is the most common Type1 driver. These drivers are implemented using native code.

- **Type2- Native -API Partly -Java Driver**

Type2 drivers wrap this layer of java around database-specific native code libraries. For Oracle databases, the native libraries might be based on OCI (Oracle Call Interface) libraries, which were originally designed for C/C++ programmers.

- **Type3- Net -Protocol All - Java Driver**

Type3 drivers communicate via a generic network protocol to a piece of custom middleware. The middleware components might use any type of driver to provide the actual database access. Web Logic's Tengah product line is an example.

- **Type4- Native -Protocol Al l -Java Driver**

Type4 divers are the most direct of the lot. Written entirely in java, Type4 drivers understand database-specific networking protocols and can access the database directly without any additional software.

CHAPTER 4

SERVLETS

Servlets are Java technology's answer to CGI programming. They are programs that run on a Web server and build Web pages. Building Web pages on the fly is useful (and commonly done) for a number of reasons:

The Web page is based on data submitted by the user.

For example the results pages from search engines are generated this way, and programs that process orders for E-comsite do this as well.

The data changes frequently.

For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date.

- **Efficient:**

With traditional CGI, a new process is started for each HTTP request. If the CGI process does a relatively fast operation, the overhead of starting the process can dominate the execution time.

- **Convenient :**

Hey, you are already known Java. Why learn Perl too? Besides the convenience of being able to use a familiar language, servlets have an extension infrastructure for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions, and many other such utilities.

- **Powerful:**

Java servlets let you easily do several things that are difficult or impossible with regular CGI. For one thing, servlets can talk directly to the Web server (regular CGI programs can't). This simplifies operations that need to look up images and other data stored in standard places. Servlets can also share data among each other, making useful thing like database connection pools easy to implement

Portable : Servlets are written in java and follows a well-standardized API. Consequently, servlets return for, I-Planet Enterprise Server can run virtually unchanged on Apache, Microsoft IIS, or Web Star. Servlets are supported directly or via plug-in on almost every major Web Server.

- **Inexpensive :**

There are a number of free or very inexpensive Web servers available that are good for "personal" use or low-level Web sites. However, with the major exception of Apache, which is free, most commercial-quality Web servers are relatively expensive.

4.1 JSP

Java Serve Pages (JSP) is a technology that lets you mix regular, static HTML with dynamically-generated HTML. Many Web pages that are built by CGI programs are mostly static, with the dynamic part limited to a few small locations. But most CGI variations, including servlets, make you generate the entire page via your program, even though most of it is always the same. JSP lets you create the two parts separately. Here's an example:

➢ **Advantages of JSP**

- Vs. Active Server Pages (ASP) :

ASP is a similar technology from Microsoft. The advantages of JSP are twofold.First, the dynamic part is written in Java not in Visual Basic or other MS-specific language. So it is more powerful and easier to use. Second, it portable to other operating systems and non-Microsoft web servers.

- Vs. Pure Servlets :

JSP doesn't give you anything that you couldn't in principle do with a servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people

on different tasks: your Web page design experts can build the HTML, leaving places for your servlet programmers to insert the dynamic content.

- Vs. Server -Side Includes (SSI) :

SSI is a widely-supported technology for including externally-defined pieces into a static Web page. JSP is better because it lets you use servlets instead of a separate program to generate that dynamic part. Besides, SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

To be a servlet, a class should extend HttpServlet and override doGet or doPost (or both), depending on whether the data is being sent by GET or by POST. These methods take two arguments: an httpServletRequest and an HttpServletResponse. The HttpServletRequest has methods that let you find out about incoming information such as FORM data, HTTP request headers, and the like. The HttpServletResponse has methods that lets you specify the HTTP response line (200,404, etc.), response headers (Content-

Type, Set-Cookies, etc.), and, most importantly, lets you obtain a PrintWriter used to send output back to the client. For simple servlets, most of the effort is println statements that generate the desired page. Note that doGet and doPost throw two exceptions, so you are required to include them in the declaration. Also note that you have to import classes in java.io (for printWriter, etc.), javax.servlet (for HttpServlet, etc.), and javax.servlet.http (for HttpServletRequest and HttpServletResponse). Finally, note that doGet and doPost are called by the service method, and sometimes you may want to override service directly,

- **Running the Servlet :**

With the Java web Server, servlets are placed in the servlets directory within the main JWS installation directory, and are invoked via http://host/servlet/ServletName. Note that the directory is servlets, plural, while the URL refers to servlet, singular. Since this example was placed in the hall package, it would be invoked via http://host/servlet/hall.Helloworld. Other Web servers may have slightly different conventions on where to install servlets and how to invoke them. Most servers also let you define aliases for servlets. So that a servlet can be invoked via

http://host/anypath/ any-file.html. The process for doing this is completely server specific: check your server`s documentation for details.

- **Client interaction:**

The client interaction is handled by two of the standard HttpServlet methods, doGet and doPost.

- The doGet method replies to GET requests by sending an HTML page which contains the list of the currently subscribe or unsubscribe an address:
- The response content type is again set to text/html and the response is marked as not cacheable to proxy servers and clients (because it is dynamically created) by setting an HTTP header “pragma:no-cache”. The form asks the client to use the POST method for submitting form data.
- Here is a typical output by this method:
- The doPost method receives the submitted form data, updates the address list and sends back a confirmation page:

Finally a confirmation page is sent with the usual method. Req.getRequestURI () is used to get the URI of the Servlet for a link back to the main page (which is created by doGet).

As usual, the Servlet extends javax.http.servlet.HttpServlet and overrides

Get Servlet Info to provide a short notice. At last, here is the full source code of the List Manager Servlet.

4.2 Back End

Executing SQL Queries:

To really use a database, we need to have some way to execute queries. The simplest way to execute a query is to use the java.sql.Statement class. Statement objects are never instantiated directly; instead, a program calls the createStatement() method of Connection to obtain a new Statement object:

Statement stmt=con.crea:Statement();

You should know that the ResultSet is linked to its parent Statement. Therefore, if a Statement is closed or used to execute another query, any related ResultSet objects are closed automatically.

Handling SQL Exceptions :

DBPhoneLookup encloses most of its code in a try/catch block. This block catches two exceptions: ClassNotFoundException and SQLException. The former is thrown by the Class.forName() method when the JDBC driver class can not be loaded.

The latter is thrown by any JDBC method that has a problem. SQLException objects are just like any other exception type, with the additional feature that they can chain. The

This code displays the message from the first exception and then loops through all the remaining exceptions, outputting the error message associated with each one. In practice, the first exception will generally include the most relevant information.

Results in Detail :

Before we continue, we should take a closer look at the ResultSet interface and related ResultSetMetaData interface. In Example9-1, we knew what our query looked like, and we knew what we expected to get back, so we formatted the output appropriately. But, if we want to display the results of a query in an HTML table, it would nice to have some Java code that builds the table automatically from the ResultSet rather than having to write the same loop-and-display code over and over. As an added bonus, this kind of code makes it possible to change the contents of the table simply by changing the query.

4.3 Modules

There are 5 modules in this project.

- Administrator Module.
- Reservation Agent Module.
- Passenger Module.
- Payment.
- Cancellation.

4.4 Modules Explanation:

- Administrator Module.

Enables the administrator to perform all administrative functions and manage inventory over LAN or the Internet. The administrator can define or modify routes, fares schedules and assign or deny access for qualified travel agents and other authorized users.

- Reservation Agent Module.

Allows the airlines reservation agents to make and modify reservation on the LAN or over the internet. The reservation agents could be stationed at any airline office location.

· Passenger Module.

This module enables online customers to make reservations, views their bookings, make special service requests and define their preferences over the web.

· Payment.

Provides the airline with the ability to set up various travel agents and give them reservations capabilities over the Internet. The travel agents are able to display and offer discounted fares to passengers.

· Cancellation. The system should allow the user to cancel the existing booking. In this cancellation very helpful in all the travelers.

CHAPTER 5

SYSTEM DESIGN

5.1 UML Diagrams

The Unified Modified Language prescribes a standard set of diagrams and notations for modeling object oriented systems, and describe the underlying semantics of what these diagrams and symbols mean. Whereas there has been to this point many notations and methods used for object-oriented design, now there is a single notation for modelers to learn.

UML can be used to model different kinds of systems: software systems, hardware systems, and real-world organizations. UML offers nine diagrams in which to model systems:

- Use Case diagram for modeling the business processes
- Sequence diagram for modeling message passing between objects
- Collaboration diagram for modeling object interactions
- Stat e diagram for modeling the behavior of objects in the system

5.2 Class Diagrams

The class diagram is the main static analysis and design diagram for a system. In it, the class structure of the system is specified, with relationships between classes and inheritance structures. During analysis of the system, the diagram is developed with an eye for an ideal solution. During design, the same diagram is used, and modified to conform to implementation details.

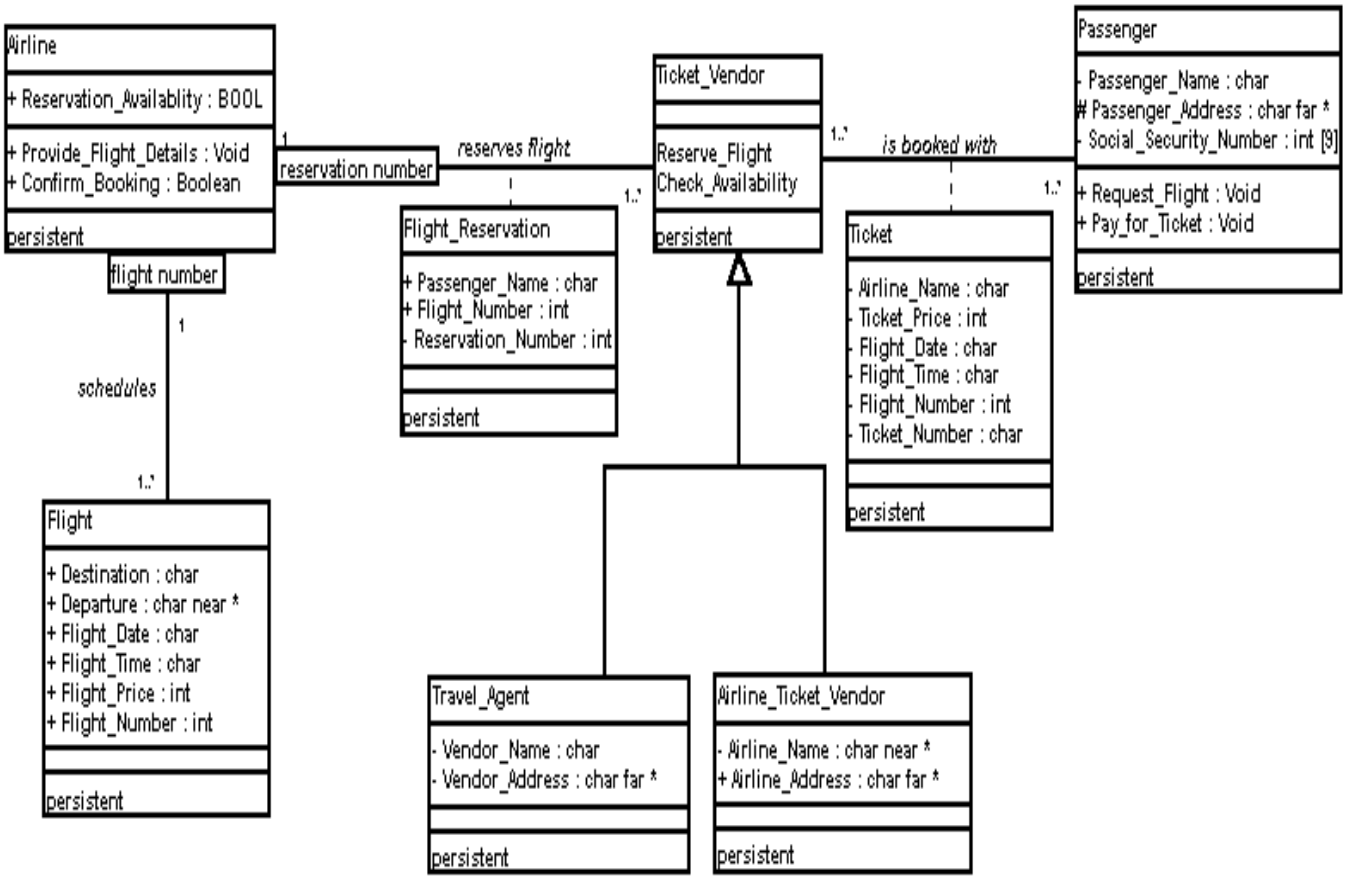

Figure 5.2 Class Diagram

5.3 Sequence Diagram

The Sequence diagram is one of the most effective diagrams to model object interactions in a system. A Sequence diagram is modeled for every Use Case. Whereas the Use Case diagram enables modeling of a business view of the scenario, the Sequence diagram contains implementation details of the scenario, including the objects and classes that are used to implement the scenario, and messages passed between the objects.

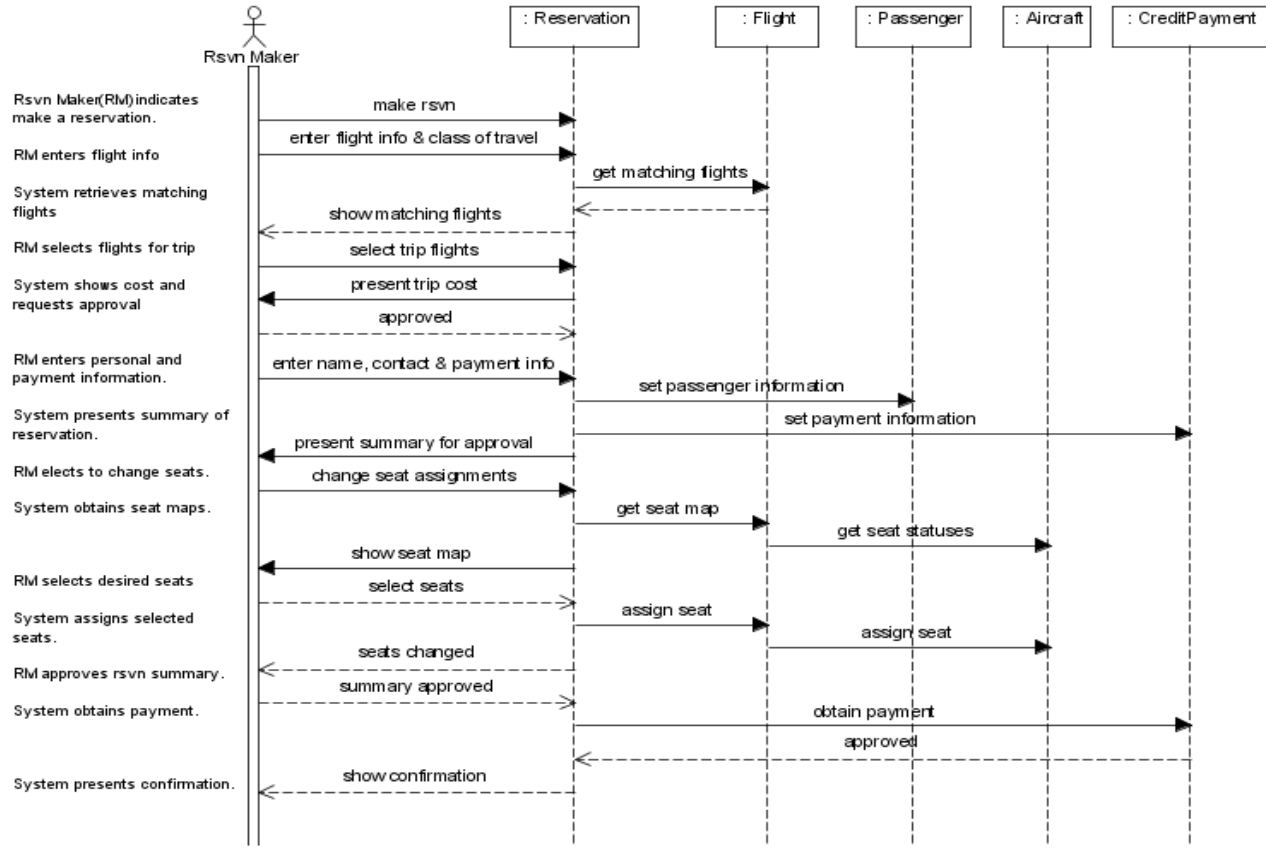

Figure 5.3 Sequence Diagram

5.4 Data Dictionary

Introduction

DICTIONARY The logical characteristics of current system data stores including Name, Address, Flight code, Source, Destination, Airline code, Flight code, Credit card number, Payment amount etc identifies process where the data are used and where immediate access to information required, Serves as the basis for identifying database requirements during system design.

Data Dictionary

1. Cancellation.

This table is used to store the cancel details.

Field name	Description	Data type	Size	constraints
Cancel id	Cancellation id	int	10	PRIMARY KEY
Reservation id	Reservation id	int	10	FOREIGN KEY
Cancelation date	Date of Cancellation	date		NOT NULL
Refund money	Money to be refundable	int	10	NOT NULL

2. Classes

This table is used to store the class details.

Field name	Description	Data type	Size	constraints
Class id	Id of the class	int	10 P	RIMARY KEY
Flight code	-	int	10	FOREIGN KEY
Class code	-	varchar		50
classname	Name of the class	varchar	50	NOT NULL
Fare	-	int	10,0	NOT NULL
Total class seat	Total seats in a class	int	10	NOT NULL

3. Flight days

This table is used to store the flight day's details

Field name	Description	Data type	Size	constraints
Date code	Code of the date flight departure	int	10	PRIMARY KEY
Flightcode	Code of the flight	int	10	FOREIGN KEY
Date	date	date		NOT NULL
departure	Departure time time	varchar		NOT NULL

4. Flight details

This table is used to store the flight details.

Field name	Description	Data type	Size	constraints
flightcode	Code of the flight	int	10	PRIMARY KEY
Airlinecode	Code of the airlines	varchar	100	NOT NULL
Flightname	Name of the flight	varchar	100	NOT NULL
source	Starting place of the flight	varchar	100	NOT NULL
destination	Destination of the flight	varchar	100	NOT NULL
Total capacity	Total capacity of the flight	int	10	NOT NULL

5. Passenger

This table is used to store passenger details.

Field name	Description	Data type	Size	constraints
Passenger no	Number of the passengers	Int	10	PRIMARY KEY
flightcode	Code of the flight	Int	10	FOREIGN KEY
address	-	varchar	250	NOT NULL
Nationality	-	varchar	250	NOT NULL
Name	Name of the user	varchar	250	NOT NULL
Dob	Date of birth	Date		NOT NULL
Gender	-	varchar	250	NOT NULL
Phoneno	Phone number	varchar	250	NOT NULL
emailid	Mail id	varchar	250	NOT NULL
Passportno	Passport number	int	10	NOT NULL
reservationid	Reservation id	int	10	FOREIGN KEY

6. Payment

This table is used to store payment details

Field name	Description	Data type	Size	constraints
Payment id	-	int	10	PRIMARY KEY
Check no	Checking number	int	10	NOT NULL

Credit card no	-	int	10	NOT NULL
Paid amount	-	Decimal	10.0	NOT NULL
Payment date	-	Date		NOT NULL
Passenger no	-	int	10	NOT NULL

7. Reservation

This table is used to store reservation details.

Field name	Description	Data type	Size	constraints
Reservation id	-	int	10	PRIMARY KEY
Flightcode	-	int	10	FOREIGN KEY
Journeydate	-	Date		NOT NULL
Source	Starting place of the flight	varchar	50	NOT NULL
Destination	Ending position of the flight	varchar	50	NOT NULL
Passenger no	-	int	10	NOT NULL
Status	-	Int	10	NOT NULL
Journeytime	Time the flight starts	time	NOT NULL	Journeytime

8. Reserve check

This table is used to store reserve check details

Field name	Description	Data type	Size	constraints
Reservation id	Reservation id number	int	10	PRIMARY KEY
username	Name of the user	Varchar	250	
Pasport no	Password to login	Varchar	250	

CHAPTER 6

RESULTS

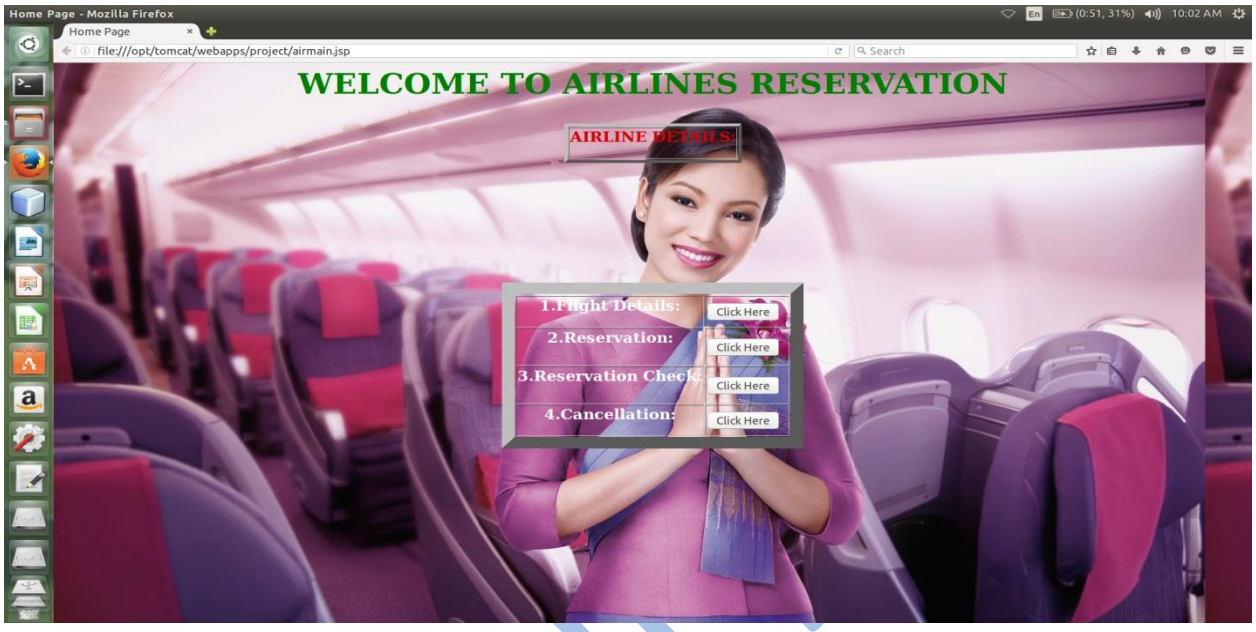

Figure: 6.1. Front Page

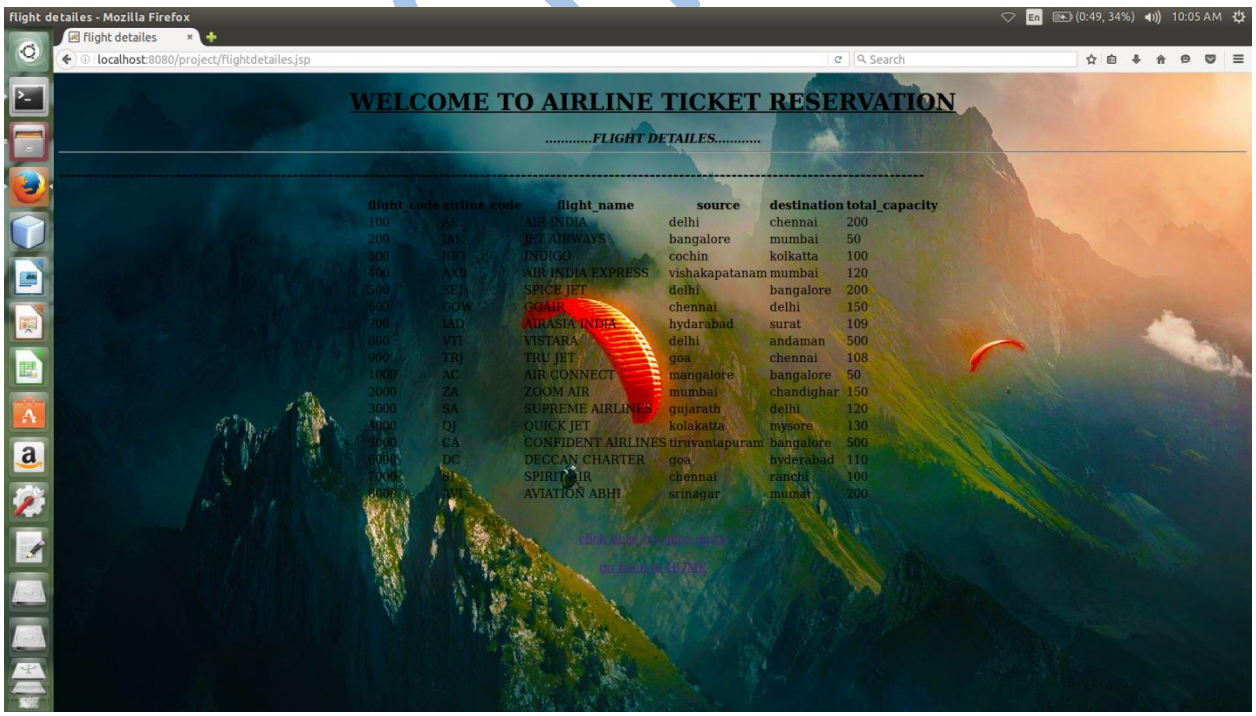

Figure: 6.2. Flight Details

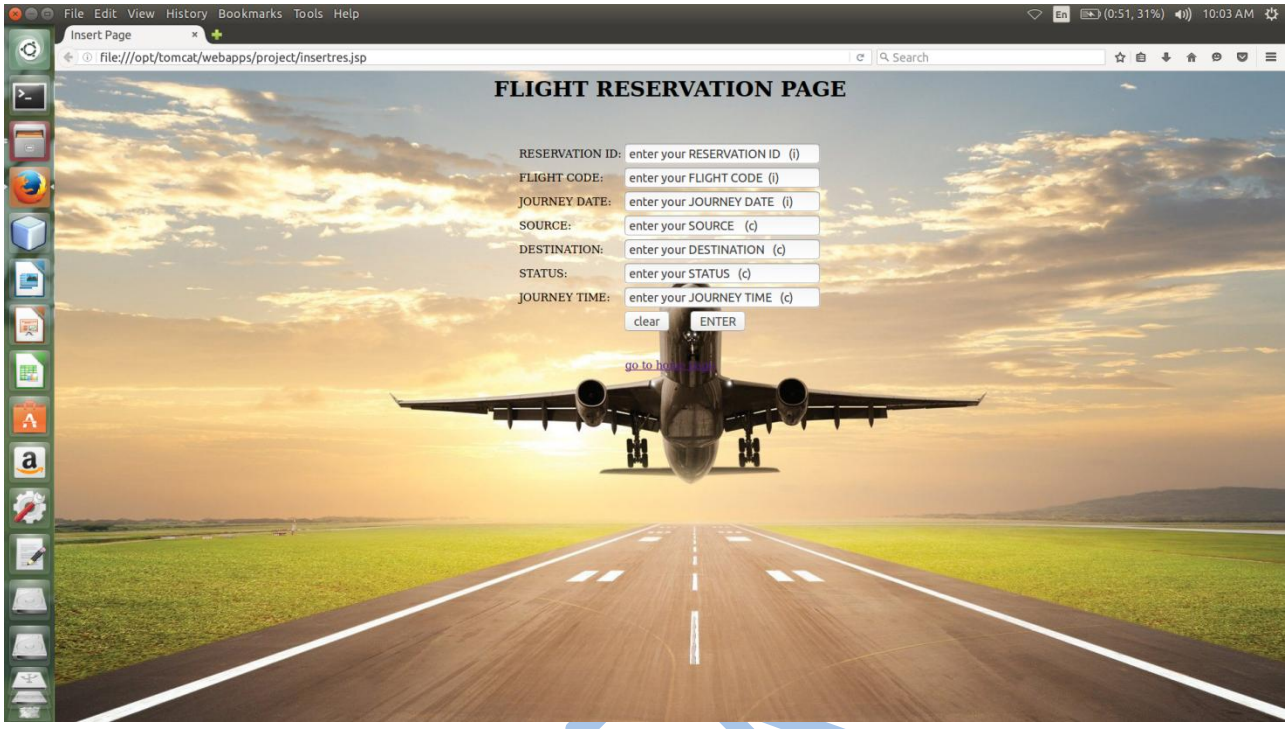

Figure: 6.3. Reservation Page

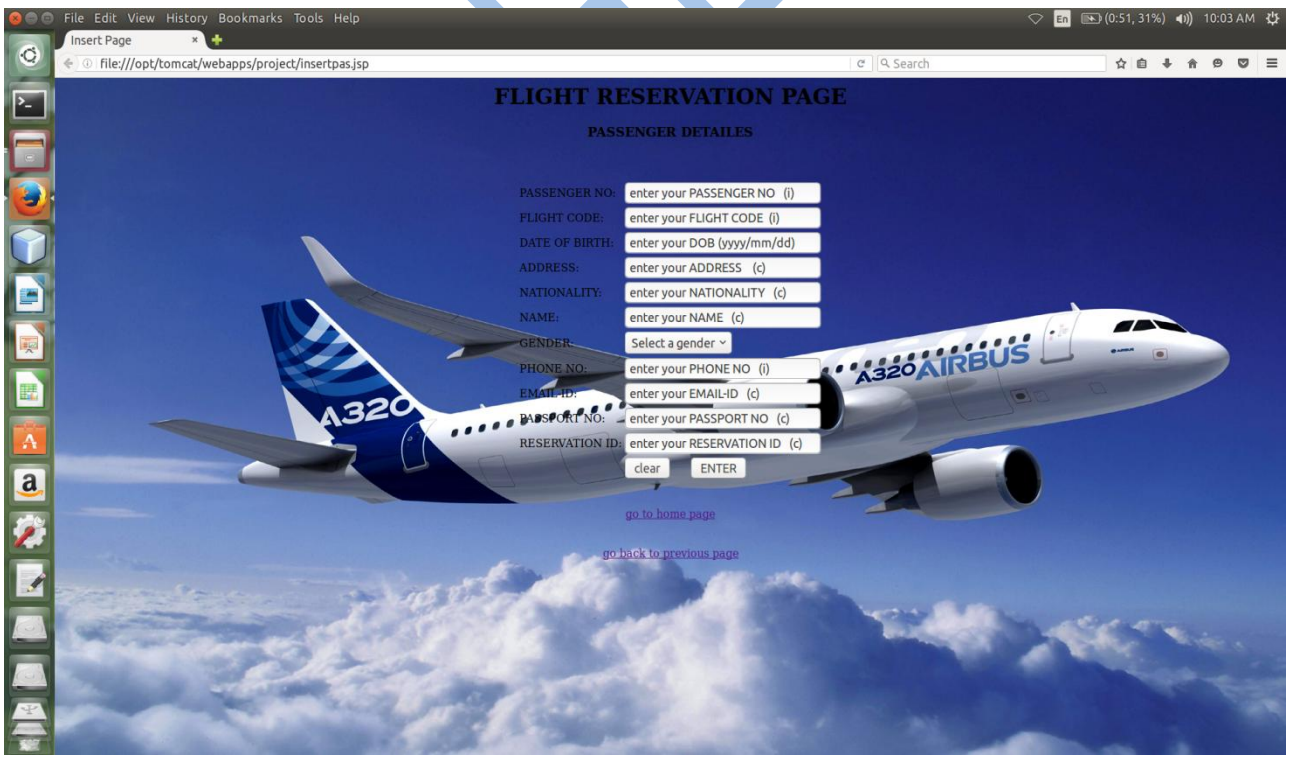

Figure: 6.4. Passenger Details

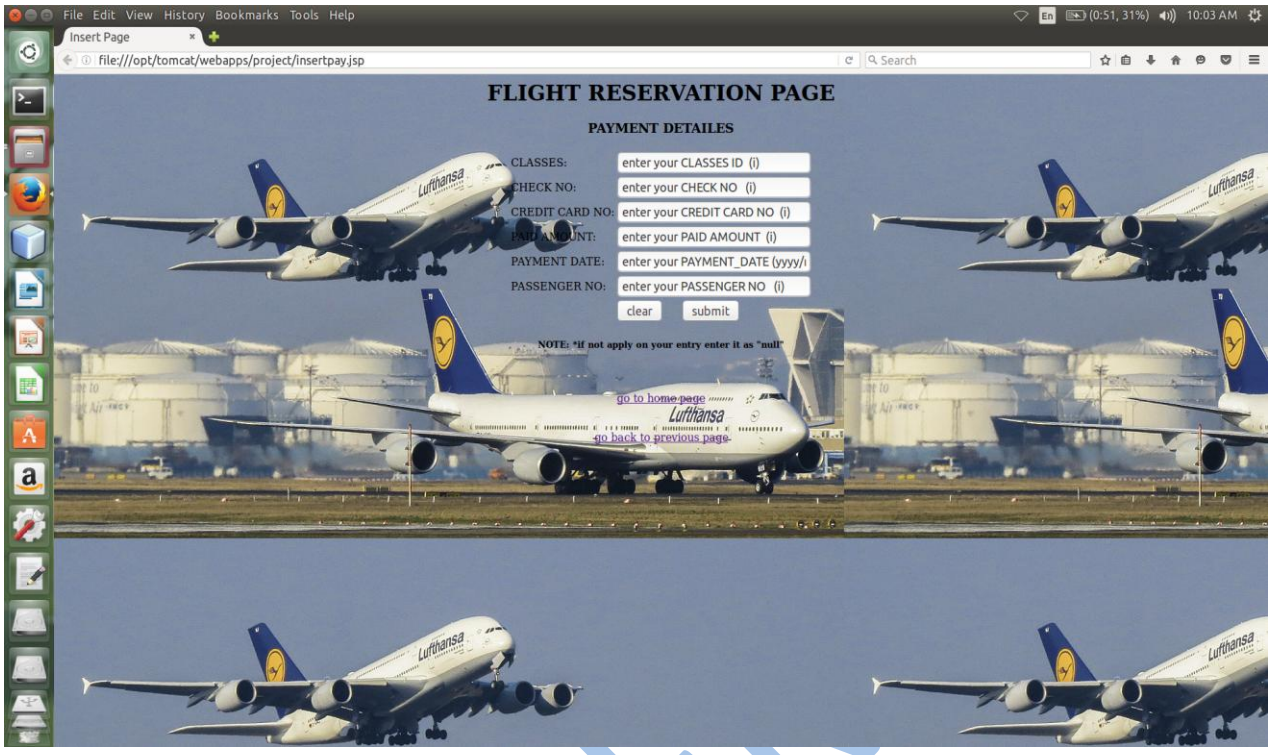

Figure: 6.5. Payment Details

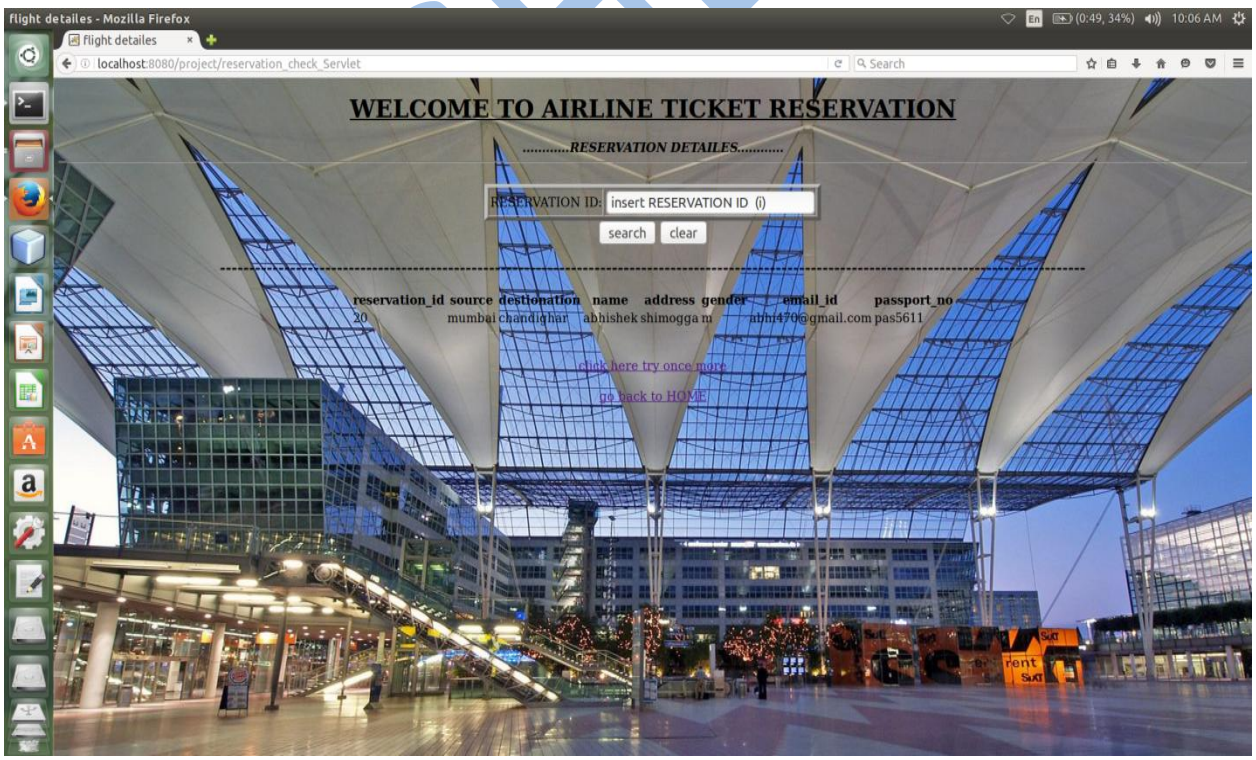

Figure: 6.6. Reservation Check.

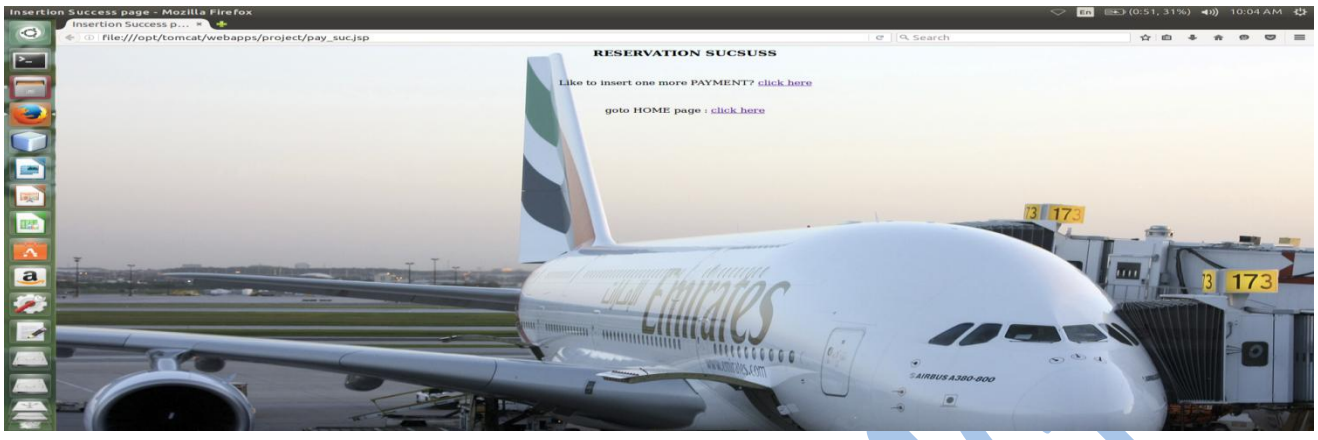

Figure: 6.7. Reservation Success Page

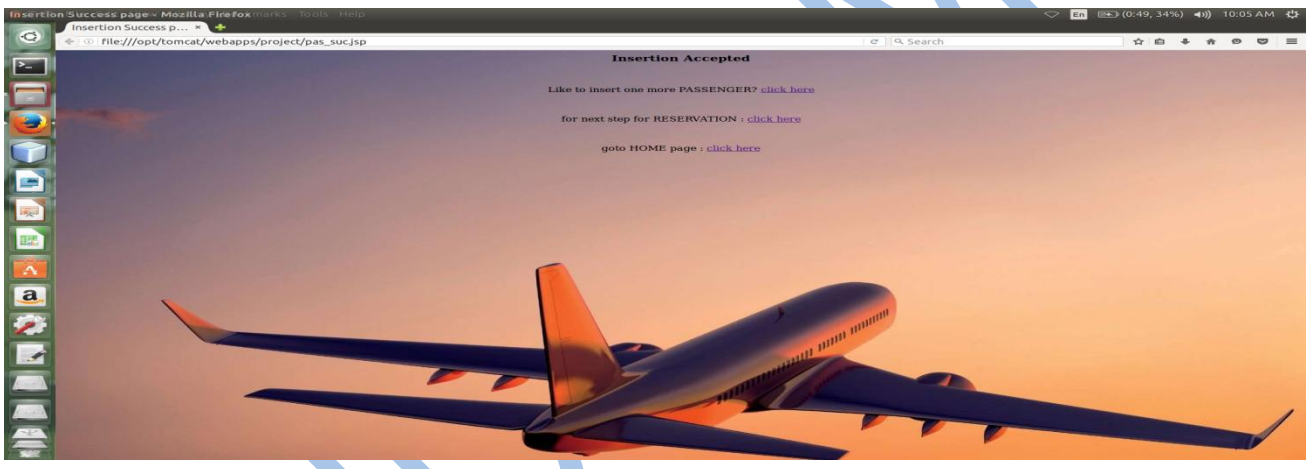

Figure: 6.8. Passenger Accept.

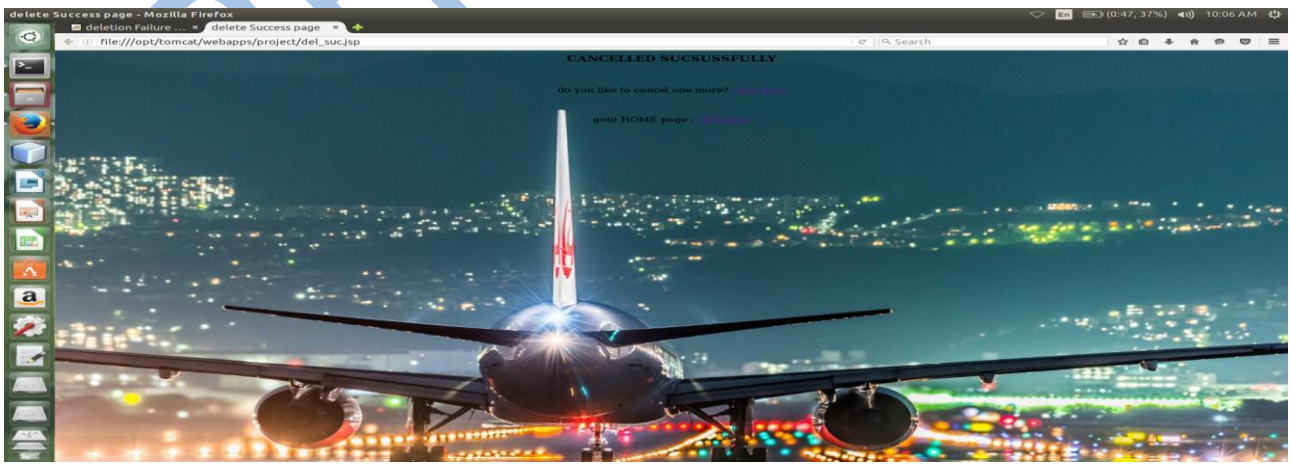

Figure: 6.9. Front page

CONCLUSION

The entire project has been developed and deployed as per the requirements stated by the user, it is found to be bug free as per the testing standards that are implemented and by specification-untraced errors concentrated in the coming versions, which are planned to be developed in near future.

Finally, we like to conclude that we put all our efforts throughout the development of our project and tired to fulfill most of the requirements of the user.

REFERENCE

Websites

- http://www.google.com
- http://www.kashipara/projects.com
- http://www.programmer2programmer.net
- http://www.slideshare.net

Books

- The complete reference of Java , 7th Edition
- SQL Bible, 2nd Edition (Paperback)
- Database Development