Online Book Shop

INDEX

INTRODUCTION

Objective:

Online Book Shop, is an application, is built on exclusively on JSPs. This is a Books Cart where it maintains repository of Books of different categories. We can get the member ship just by registration. After that we can order the books which we are interested by viewing the books which are available at book store.

Existing System:

The existing system is manual one. I.e. a user has to go the bookstore (shop) and he has to ask the owner of that bookstore about the book he wants. It is sometimes not so easy to find out our required books. Owner has to control all the customers simultaneously but in some cases he fails to manage and answer all the customers. It has all the draw-backs of that a traditional system poses such has high investment, recruiting employees, large volume stock, high level risk, time consuming process, tuff to get customer satisfaction and soon.

Proposed System:

The first step of analysis process involves the identification of need. The success of a system depends largely on how accurately a problem is defined, thoroughly investigated and properly carried out through the choice of solution.

This application has been developed in order to overcome the difficulties encountered while using the existing system. This is web-based distributed application that we can access through web browser.

To develop this web application we put minimum efforts when compared with real investment to start a physical bookstore. The technologies we used here are: JSPs, HTML, Web/application server and relational database.

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 technicaln analysis.

i) 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.

ii) 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

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

Modules:

The system is proposed to have the following modules:

1. Administrator module
2. User module
3. Search module
4. Reports module

Administrator module:

Administrator has full rights to do any kind of operation for control total application to maintain efficiency of the application.

For Administrator:

- Adding Members
- Orders
- Books
- Categories
- Editorials

User module:

A user can register with the site and can order any number of books he wish simply by selecting and make order option.

For a user:

- Registration
- Shopping Cart
- Search operation

Search Module:

It provides different kinds of facilities that make a user very comfortable to search their interested books:

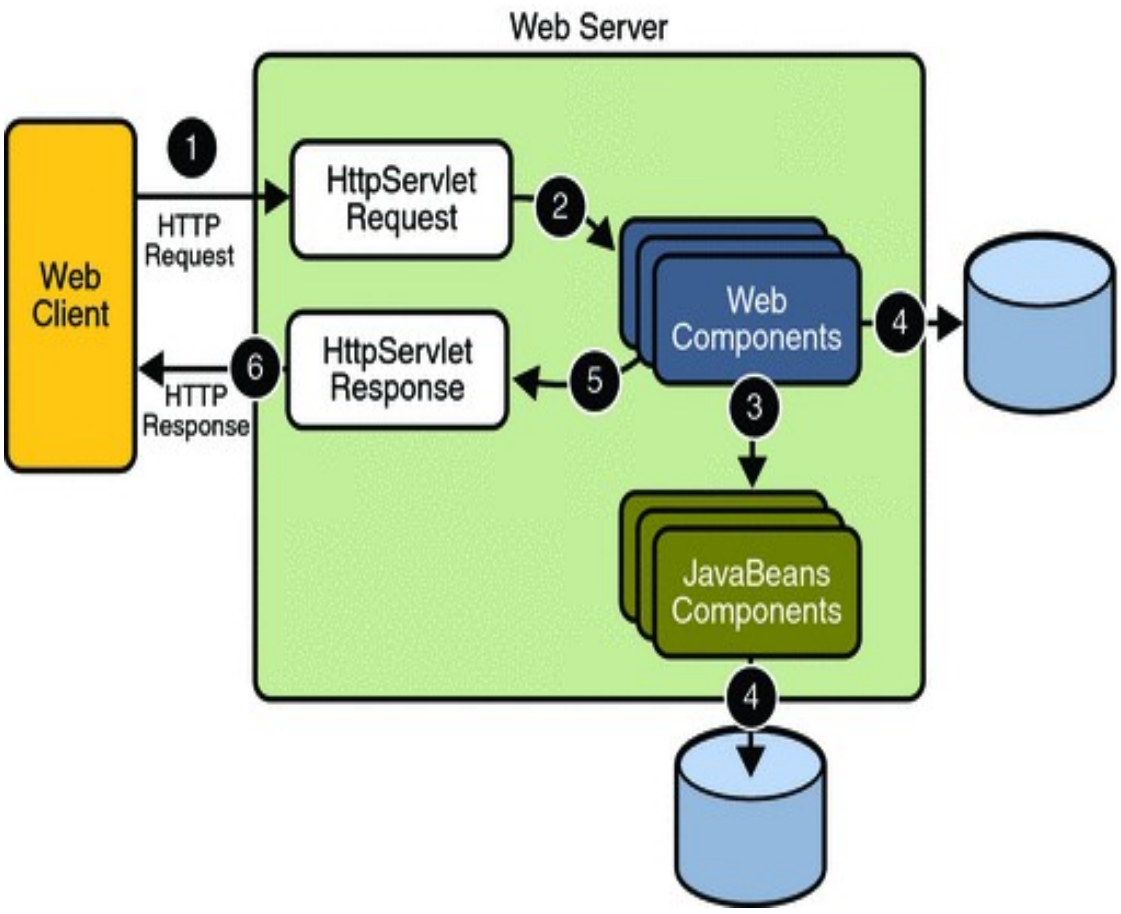
- Author wise search
- Title wise search
- Category wise search
- Price more than
- Price less than search

Reports module:

This module allows one to generate reports based on different criteria such as order reports, search reports and so on.

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.
2. 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

3. 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.

4. 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:

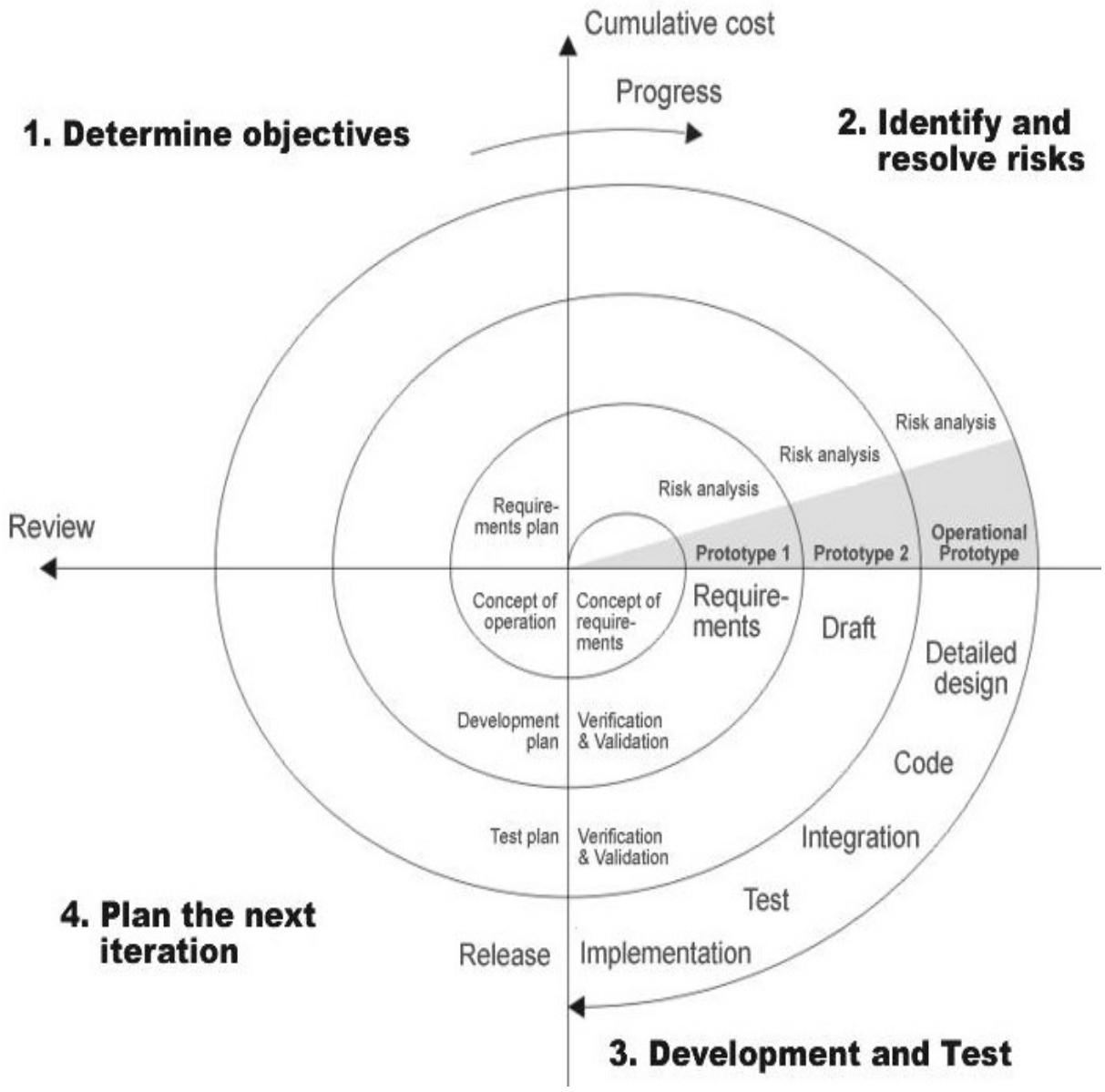


Fig 1.0-Spiral Model

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

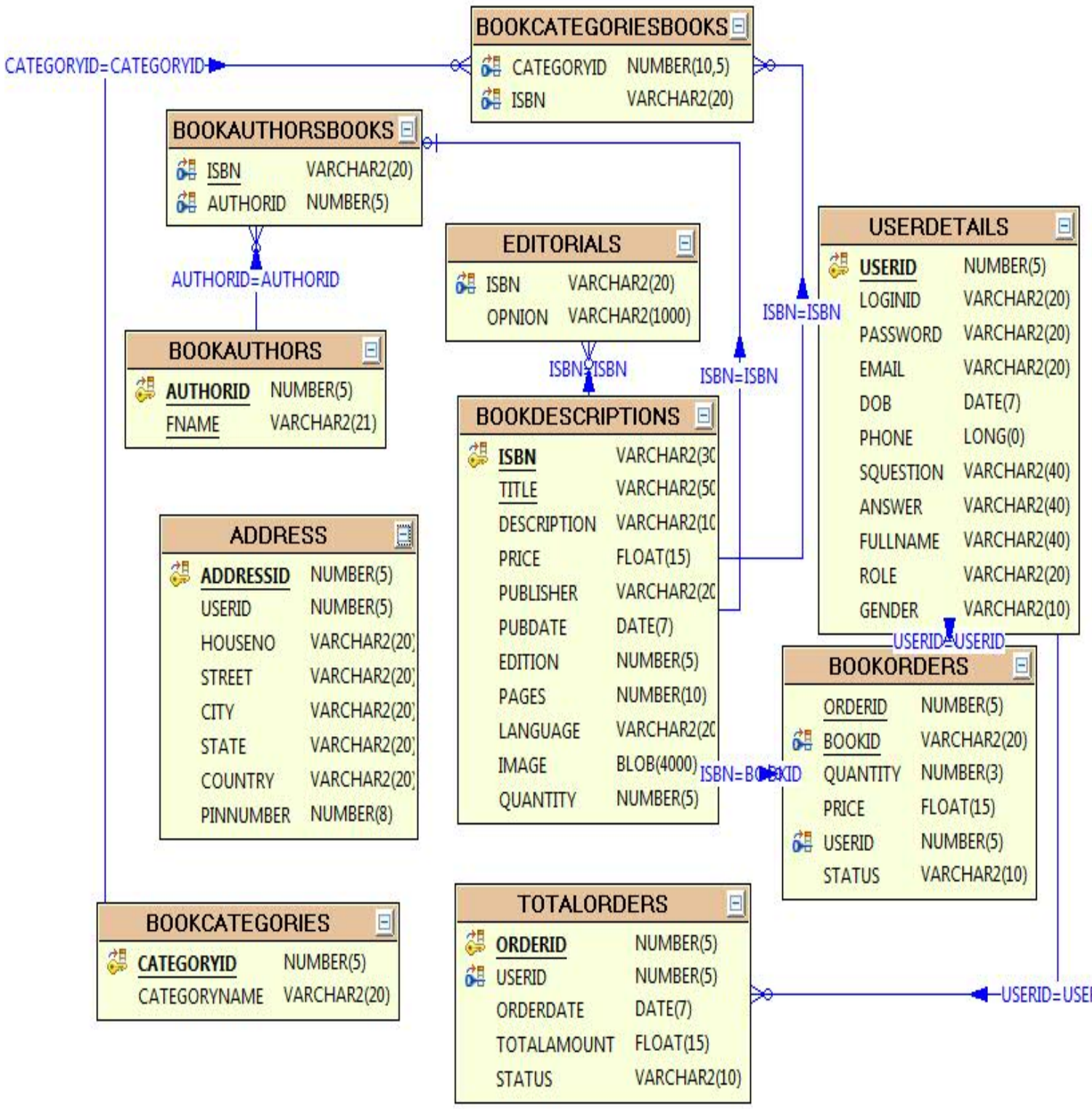
Operating System	:	Windows XP/2003 or Linux
User Interface	:	HTML, CSS
Client-side Scripting	:	JavaScript
Programming Language	:	Java
Web Applications	:	JDBC, Servlets, JSP
IDE/Workbench	:	My Eclipse 8.6
Database	:	Oracle 10g
Server Deployment	:	Tomcat 6.x

HARDWARE REQUIREMENTS

Processor	:	Core 2 Duo
Hard Disk	:	160GB
RAM	:	1GB or more

SYSTEM DESIGN

E - R Diagrams


BOOKCATEGORIESBOOKS
CATEGORYID NUMBER(10,5)
ISBN VARCHAR2(20)
CATEGORYID=CATEGORYID
BOOKAUTHORSBOOKS
ISBN VARCHAR2(20)
AUTHORID NUMBER(5)
AUTHORID=AUTHORID
BOOKAUTHORS
AUTHORID NUMBER(5)
FNAME VARCHAR2(21)
EDITORIALS
ISBN VARCHAR2(20)
OPNION VARCHAR2(1000)
ISBN=ISBN
ISBN=ISBN
ISBN=ISBN
BOOKDESCRIPTIONS
ISBN VARCHAR2(3C
TITLE VARCHAR2(5C
DESCRIPTION VARCHAR2(1C
PRICE FLOAT(15)
PUBLISHER VARCHAR2(2C
PUBDATE DATE(7)
EDITION NUMBER(5)
PAGES NUMBER(10)
LANGUAGE VARCHAR2(2C
IMAGE BLOB(4000)
QUANTITY NUMBER(5)
USERDETAILS
USERID NUMBER(5)
LOGINID VARCHAR2(20)
PASSWORD VARCHAR2(20)
EMAIL VARCHAR2(20)
DOB DATE(7)
PHONE LONG(0)
SQUESTION VARCHAR2(40)
ANSWER VARCHAR2(40)
FULLNAME VARCHAR2(40)
ROLE VARCHAR2(20)
GENDER VARCHAR2(10)
USERID=USERID
BOOKORDERS
ORDERID NUMBER(5)
BOOKID VARCHAR2(20)
QUANTITY NUMBER(3)
PRICE FLOAT(15)
USERID NUMBER(5)
STATUS VARCHAR2(10)
ISBN=BOOKID
ADDRESS
ADDRESSID NUMBER(5)
USERID NUMBER(5)
HOUSENO VARCHAR2(20
STREET VARCHAR2(20
CITY VARCHAR2(20
STATE VARCHAR2(20
COUNTRY VARCHAR2(20
PINNUMBER NUMBER(8)
BOOKCATEGORIES
CATEGORYID NUMBER(5)
CATEGORYNAME VARCHAR2(20)
TOTALORDERS
ORDERID NUMBER(5)
USERID NUMBER(5)
ORDERDATE DATE(7)
TOTALAMOUNT FLOAT(15)
STATUS VARCHAR2(10)
USERID=USERID

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:

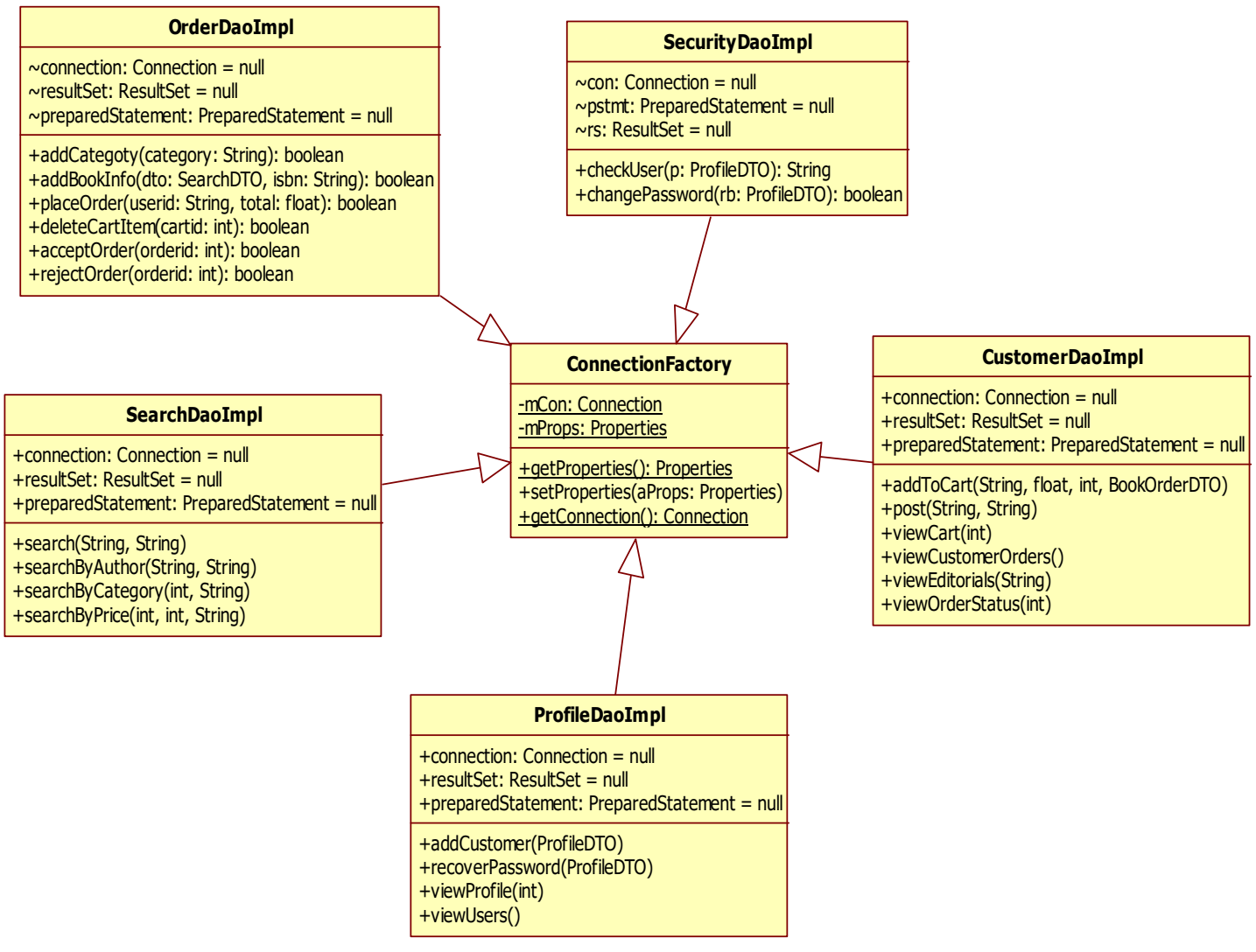
UML Analysis modeling, which focuses on the user model and structural model views of the system.

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



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
    i. This view represents the system from the users perspective.
    ii. The analysis representation describes a usage scenario from the end-users perspective.

- Structural model view
    i. In this model the data and functionality are arrived from inside the system.
    ii. 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.

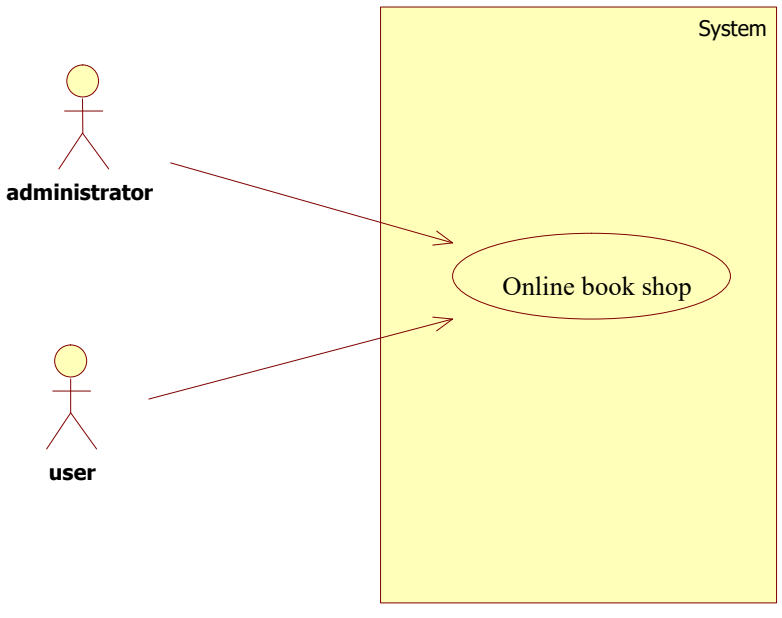
UML is specifically constructed through two different domains they are:

- ✓ 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.

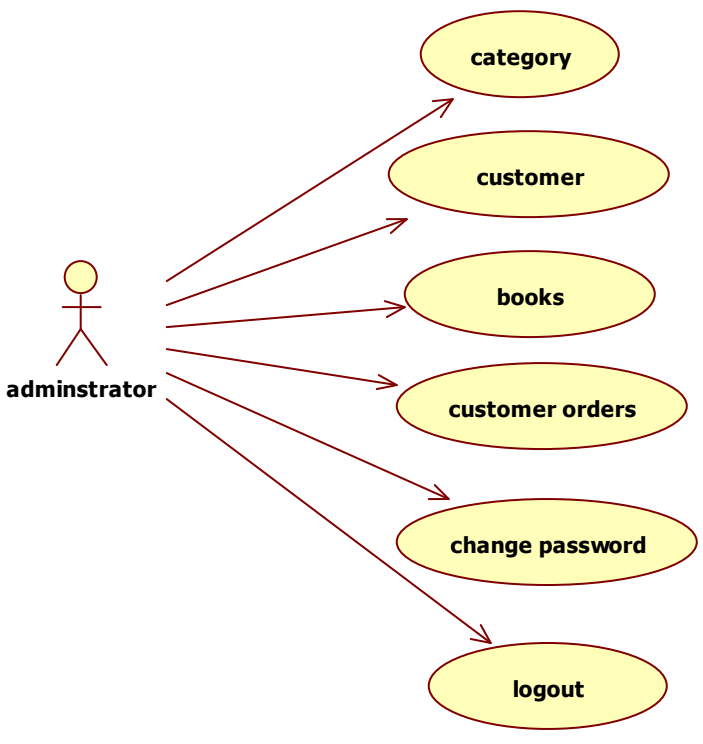
Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

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.

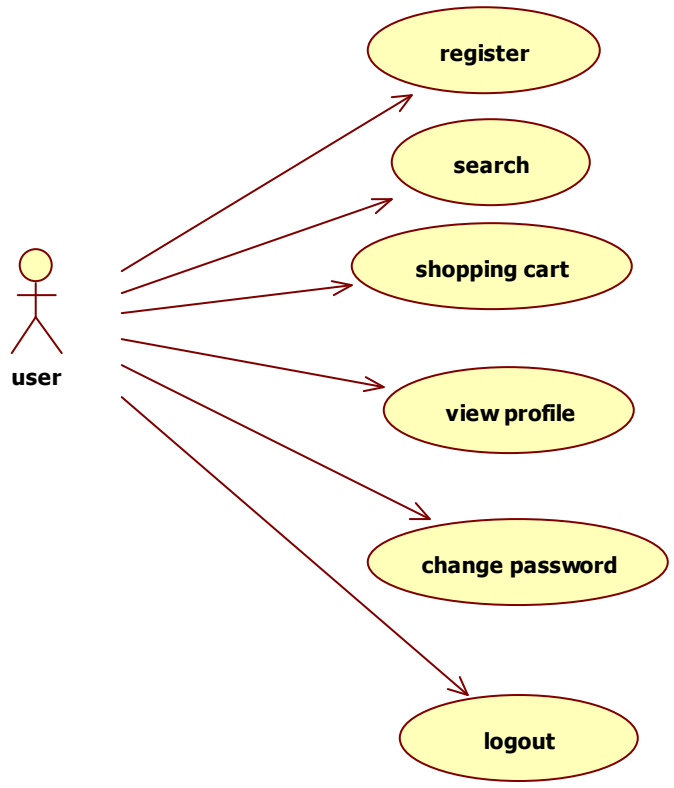
1. system Use Case Diagram



2. Administrator Use Case Diagram

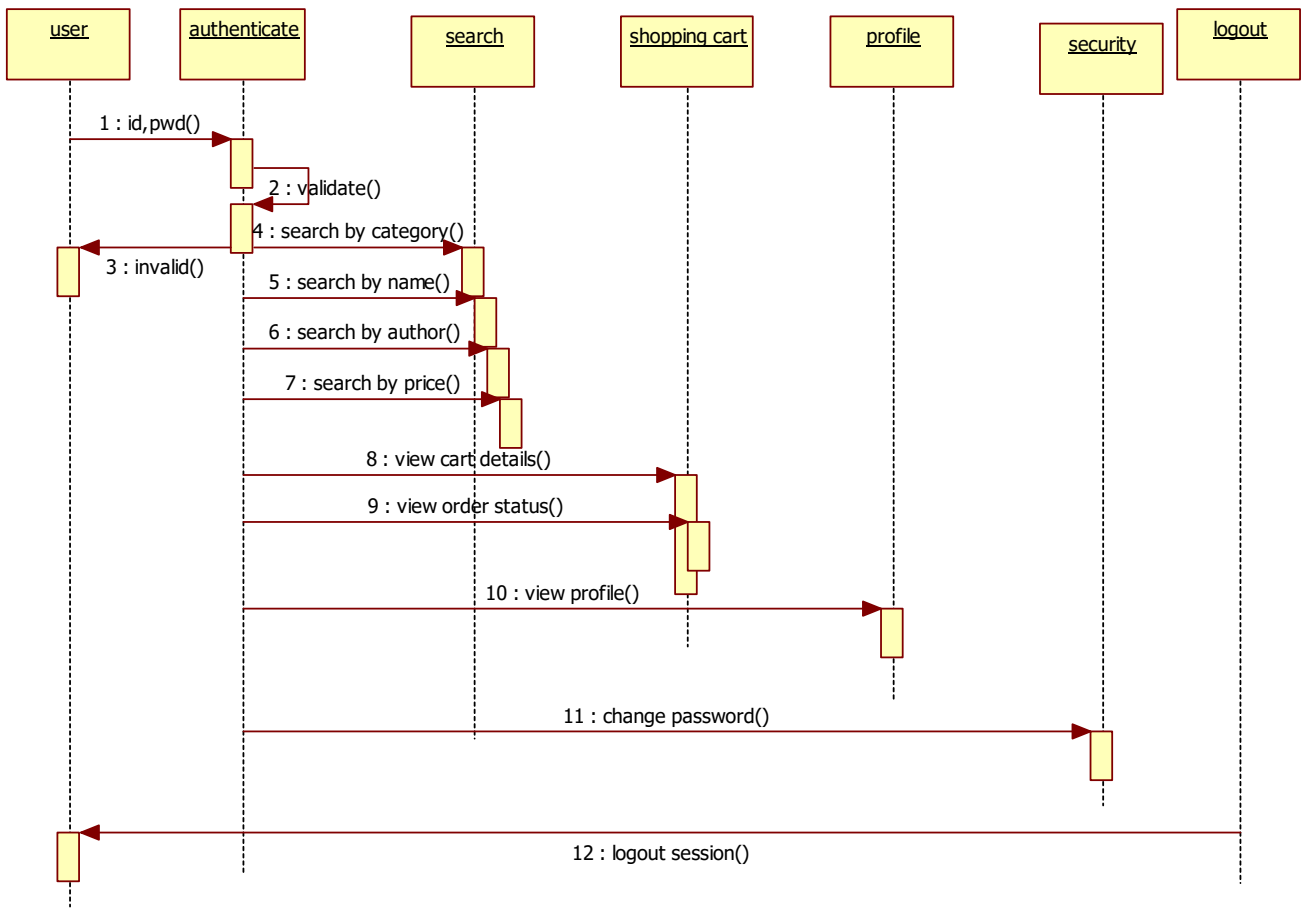


User usecase diagram

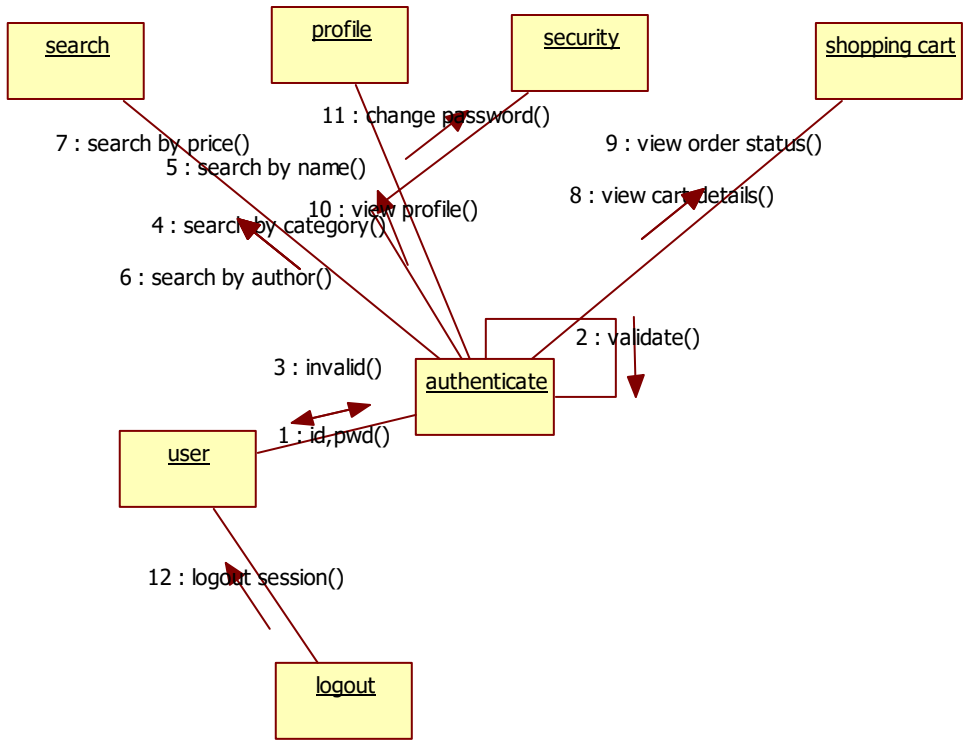


Sequence Diagram

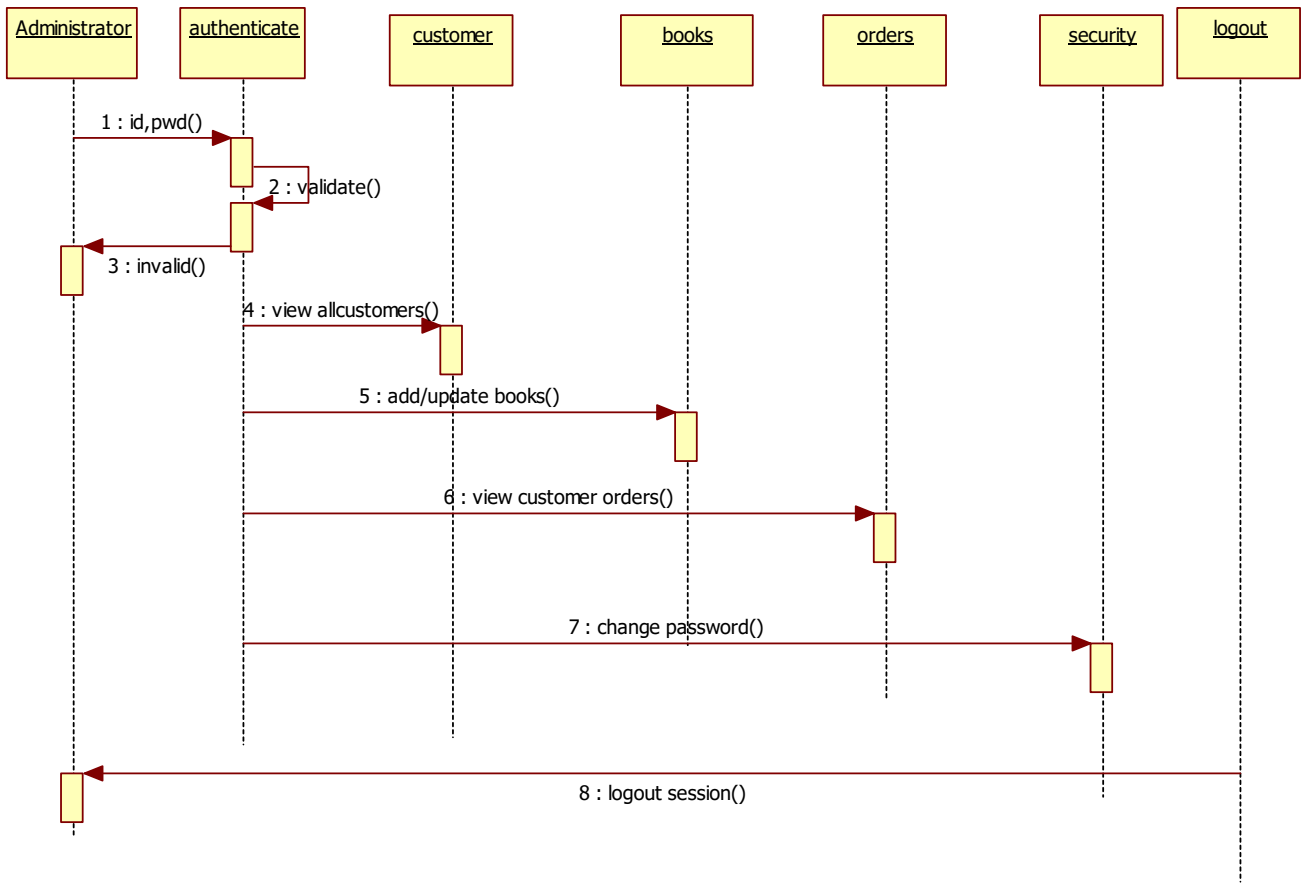
Administrator



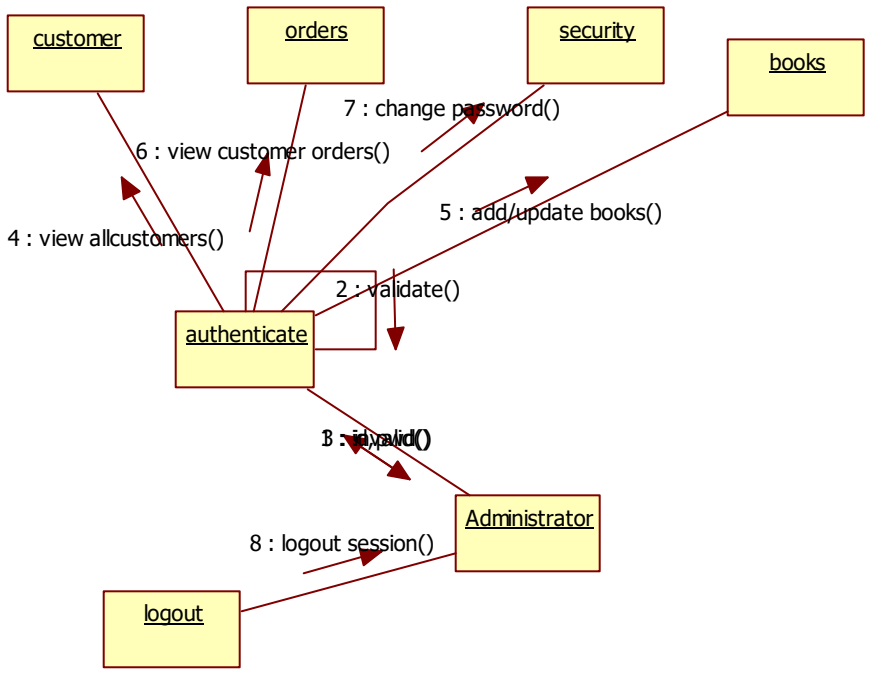
Collaboration diagram


search
profile
security
shopping cart
11 : change password()
7 : search by price()
5 : search by name()
9 : view order status()
10 : view profile()
8 : view cart details()
4 : search by category()
6 : search by author()
2 : validate()
3 : invalid()
authenticate
1 : id,pwd()
user
12 : logout session()
logout

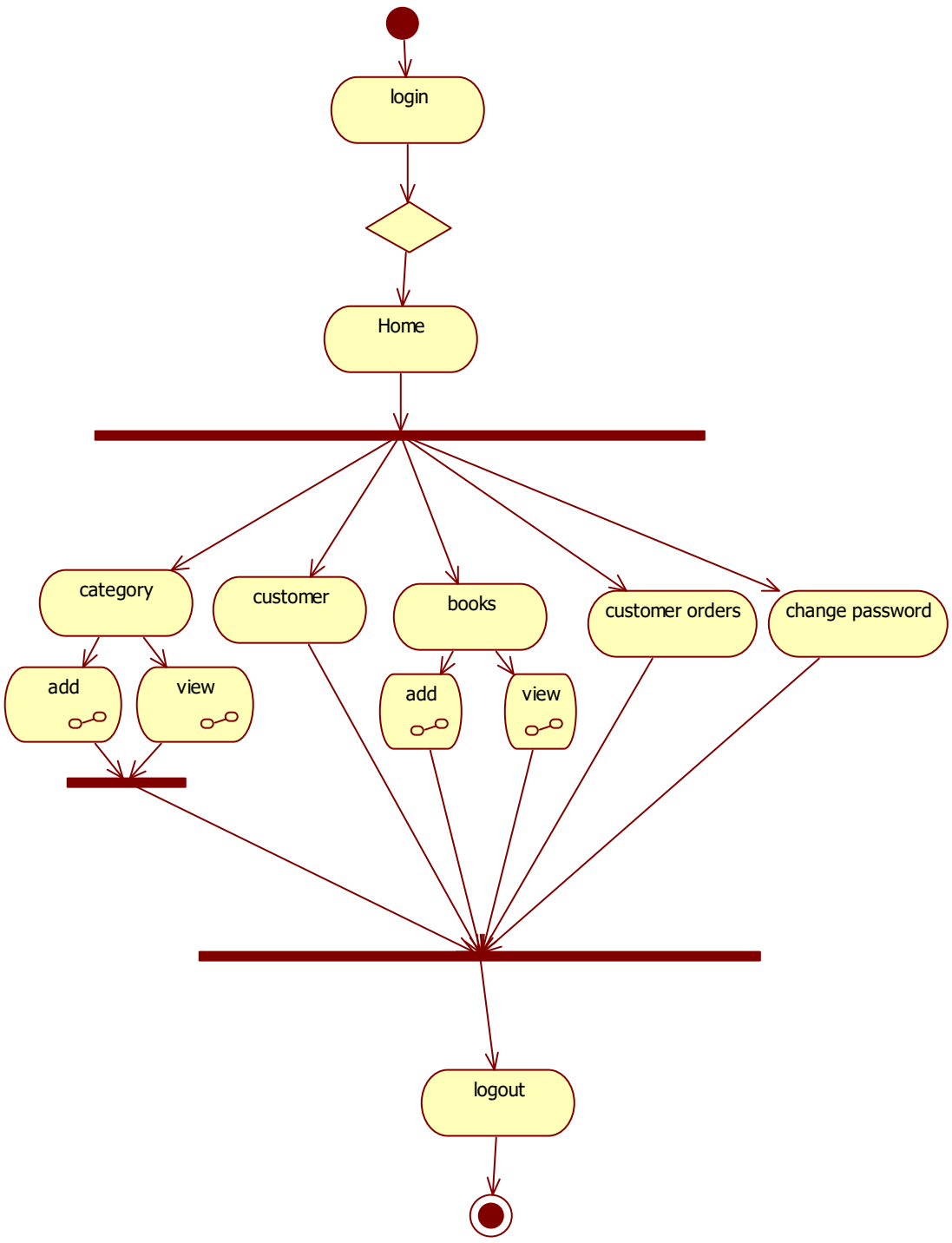
User



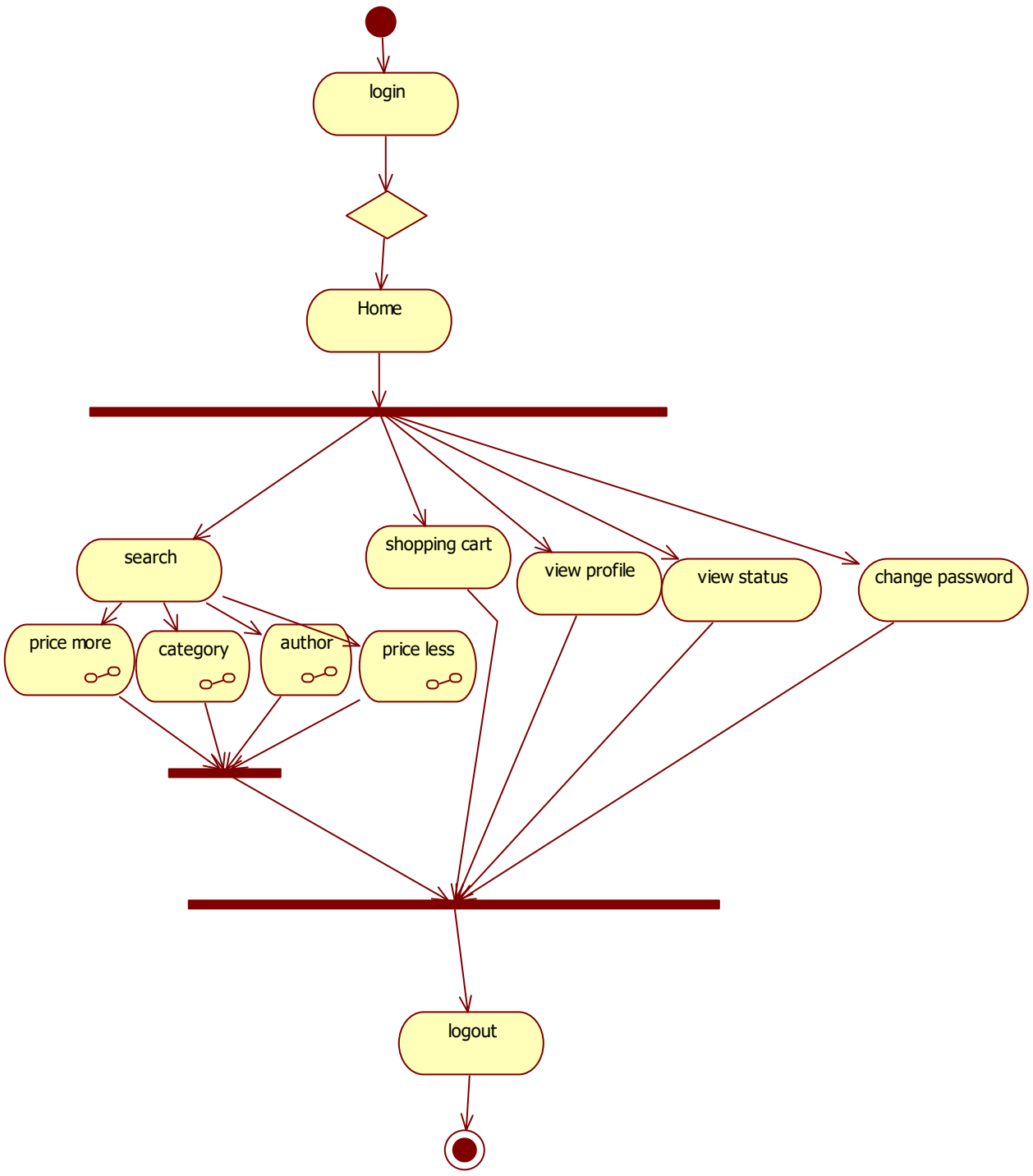
collaboration



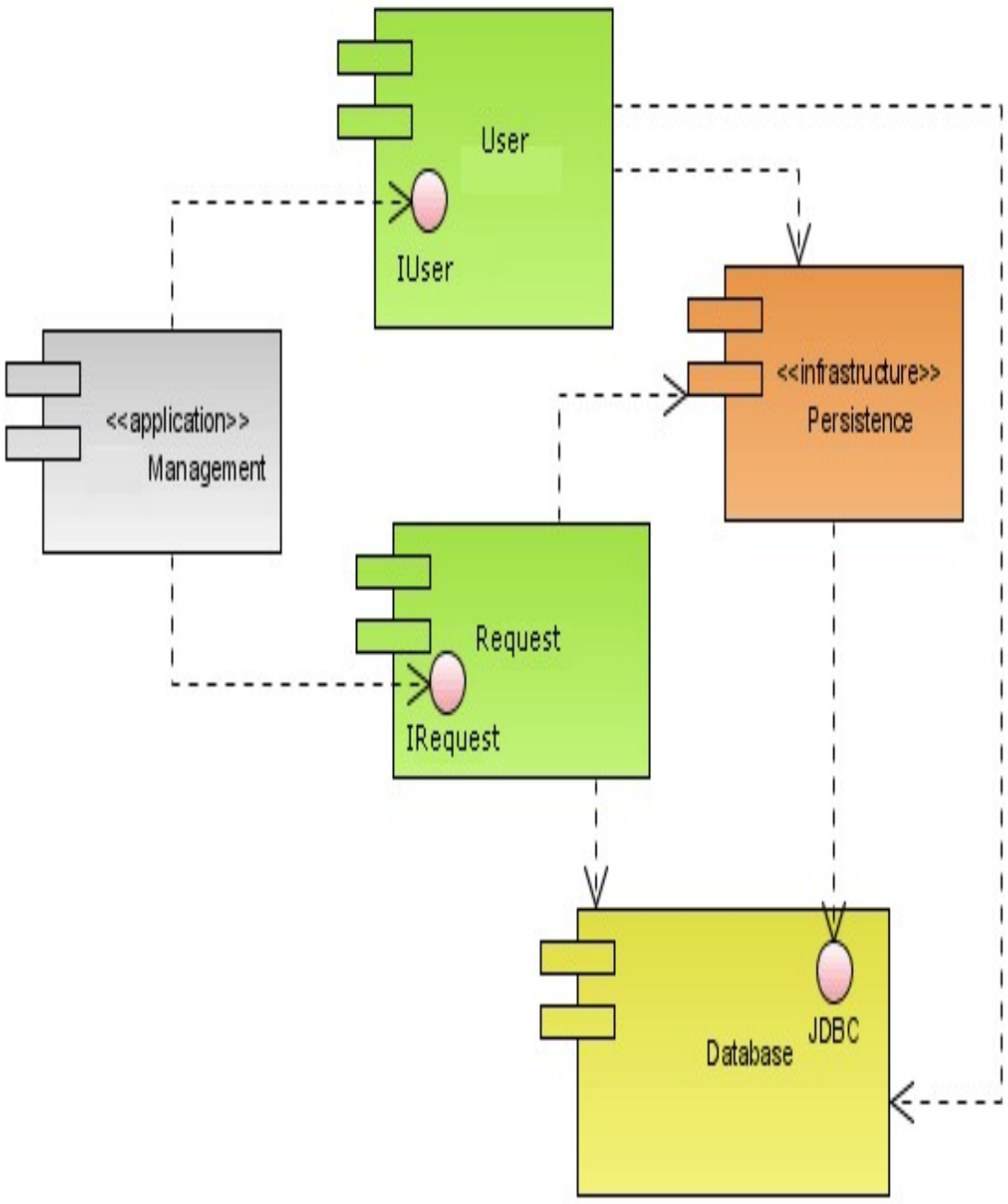
Activity Diagram


login
Home
category
customer
books
customer orders
change password
add
view
add
view
logout

User

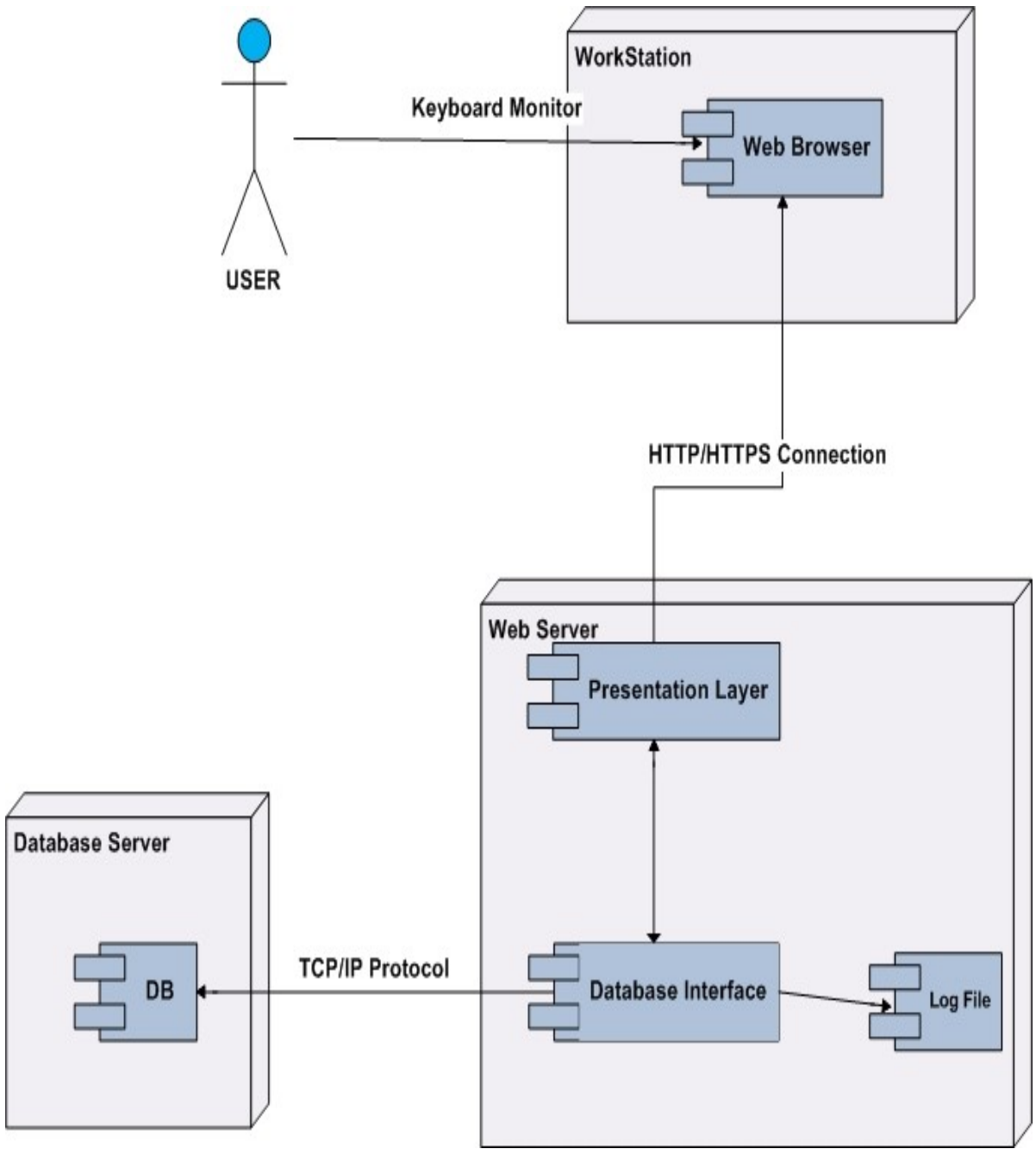


Component Diagram :


User
IUser
<<application>>
Management
<<infrastructure>>
Persistence
Request
IRequest
JDBC
Database

Deployment Diagram

Deployment Diagram:



Data Dictionary

Userdetails

Column Name	ID	Pk	Null?	Data Type	Default	H
USERID	1	1	N	NUMBER (5)		Y
LOGINID	2		Y	VARCHAR2 (20 Byte)		Y
PASSWORD	3		Y	VARCHAR2 (20 Byte)		Y
EMAIL	4		Y	VARCHAR2 (20 Byte)		Y
DOB	5		Y	DATE		Y
PHONE	6		Y	LONG		N
SQUESTION	7		Y	VARCHAR2 (40 Byte)		Y
ANSWER	8		Y	VARCHAR2 (40 Byte)		Y
FULLNAME	9		Y	VARCHAR2 (40 Byte)		Y
ROLE	10		Y	VARCHAR2 (20 Byte)		Y
GENDER	11		Y	VARCHAR2 (10 Byte)		Y

Address

Column Name	ID	Pk	Null?	Data Type	Default
ADDRESSID	1	1	N	NUMBER (5)	
USERID	2		Y	NUMBER (5)	
HOUSENO	3		Y	VARCHAR2 (20 Byte)	
STREET	4		Y	VARCHAR2 (20 Byte)	
CITY	5		Y	VARCHAR2 (20 Byte)	
STATE	6		Y	VARCHAR2 (20 Byte)	
COUNTRY	7		Y	VARCHAR2 (20 Byte)	
PINNUMBER	8		Y	NUMBER (8)	

BookAuthors

Column Name	ID	Pk	Null?	Data Type	Default
AUTHORID	1	1	N	NUMBER (5)	
FNAME	2		Y	VARCHAR2 (21 Byte)	

BookAuthorsBooks

Column Name	ID	Pk	Null?	Data Type	Default
ISBN	1		Y	VARCHAR2 (20 Byte)	
AUTHORID	2		Y	NUMBER (5)	

BookCategories

Column Name	ID	Pk	Null?	Data Type	Default
CATEGORYID	1	1	N	NUMBER (5)	
CATEGORYNAME	2		Y	VARCHAR2 (20 Byte)	

BookCategoryBooks

Column Name	ID	Pk	Null?	Data Type	Default
CATEGORYID	1		Y	NUMBER (22,5)	
ISBN	2		Y	VARCHAR2 (20 Byte)	

BookDescriptions

Column Name	ID	Pk	Null?	Data Type	Default
ISBN	1	1	N	VARCHAR2 (30 Byte)	
TITLE	2		Y	VARCHAR2 (50 Byte)	
DESCRIPTION	3		Y	VARCHAR2 (100 Byte)	
PRICE	4		Y	FLOAT (15)	
PUBLISHER	5		Y	VARCHAR2 (20 Byte)	
PUBDATE	6		Y	DATE	
EDITION	7		Y	NUMBER (5)	1
PAGES	8		Y	NUMBER (10)	
LANGUAGE	9		Y	VARCHAR2 (20 Byte)	
IMAGE	10		Y	BLOB	
QUANTITY	11		Y	NUMBER (5)	

BookOrders

Column Name	ID	Pk	Null?	Data Type	Default
ORDERID	1		Y	NUMBER (5)	
BOOKID	2		Y	VARCHAR2 (20 Byte)	
QUANTITY	3		Y	NUMBER (3)	1
PRICE	4		Y	FLOAT (15)	
USERID	5		Y	NUMBER (5)	
STATUS	6		Y	VARCHAR2 (10 Byte)	

Editorials

Column Name	ID	Pk	Null?	Data Type	Default
ISBN	1		Y	VARCHAR2 (20 Byte)	
OPNION	2		Y	VARCHAR2 (1000 Byte)	

Total Orders

Column Name	ID	Pk	Null?	Data Type	Default
ORDERID	1	1	N	NUMBER (5)	
USERID	2		Y	NUMBER (5)	
ORDERDATE	3		Y	DATE	
TOTALAMOUNT	4		Y	FLOAT (15)	
STATUS	5		Y	VARCHAR2 (10 Byte)	

CODING

```
/*
 * SecurityDAO.java
 *
 *
 */

package com.dts.dae.dao;


import com.dts.dae.model.Profile;
import com.dts.core.dao.AbstractDataAccessObject;
import com.dts.core.util.DateWrapper;
import com.dts.core.util.LoggerManager;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Types;
import java.util.Date;
/**
 *
 * @author
 */
public class SecurityDAO extends AbstractDataAccessObject
{
    Connection con;
    private String desc;
    private boolean flag;
   /** Creates a new instance of SecurityDAO */
   public SecurityDAO()
   {

            //getting Database Connection


   }



   //Login Check
```

```
public String loginCheck(Profile regbean)
{
   String loginid=regbean.getLoginID();
   String password=regbean.getPassword();
   String role="ee";
   try
   {   con=getConnection();
   System.out.println("con"+con);
     //  con.setAutoCommit(true);
       CallableStatement cstmt=con.prepareCall("{call logincheck(?,?,?)}");
       cstmt.sETtring(1,loginid);
       cstmt.sETtring(2,password);
       cstmt.registerOutParameter(3,Types.VARCHAR);
      boolean flag= cstmt.execute();
      System.out.println("flag->"+flag);
       role=cstmt.gETtring(3);
       System.out.println("logintype="+role);


   }
   catch (SQLException ex)
   {ex.printStackTrace();
       LoggerManager.writeLogSevere(ex);
       desc="Database Connection problem";
       flag=false;
   }
   //loginaudit(loginid,desc);
   return role;
}


//Method for login audit
public void loginaudit(String loginid)
{
   try
   { con=getConnection();

       CallableStatement cstmt=con.prepareCall("{call signoutprocedure(?)}");
       cstmt.sETtring(1,loginid);

       System.out.println("in loginaudit");
      cstmt.execute();

       con.close();
   }
   catch(Exception e)
   {
```

```
        e.printStackTrace();
        }
}
//Change Password
public boolean changePassword(Profile regbean)
{
   String loginid=regbean.getLoginID();
   String oldpassword=regbean.getPassword();
   String newpassword=regbean.getNewPassword();
   try
   {con=getConnection();
      con.setAutoCommit(false);

      CallableStatement cstmt=con.prepareCall("{call changePassword(?,?,?,?)}");


      cstmt.sETtring(1,loginid);
      cstmt.sETtring(2,oldpassword);
      cstmt.sETtring(3,newpassword);
      cstmt.registerOutParameter(4,Types.INTEGER);
      cstmt.execute();
      int i=cstmt.getInt(4);
      System.out.println("i="+i);
   if(i==1)
   {
      flag=true;
      con.commit();
   }
   else
   {
      flag=false;
      con.rollback();
   }
   con.close();
}
catch (SQLException ex)
{ex.printStackTrace();
   LoggerManager.writeLogSevere(ex);
   flag=false;
   try
   {
      con.rollback();
   }
   catch (SQLException sex)
   {
            LoggerManager.writeLogSevere(sex);
   }
```

```
    }
    catch (Exception e)
    {
       e.printStackTrace();
       flag=false;
       try
       {
          con.rollback();
       }
       catch (SQLException sex)
       {sex.printStackTrace();
                LoggerManager.writeLogSevere(sex);
       }
    }
    return flag;
}


 //Change Secret Question
public boolean changeQuestion(Profile regbean)
{
    String loginid=regbean.getLoginID();
    String password=regbean.getPassword();
String secretquestid=regbean.gETecretQuestionID();

String secretans=regbean.gETecretAnswer();

CallableStatement cstmt;
int i=0;
try
{

            con=getConnection();
            //con.setAutoCommit(false);


       cstmt=con.prepareCall("{call ChangeQuetion(?,?,?,?,?)}");

       cstmt.sETtring(1,loginid);
       cstmt.sETtring(2,password);
       cstmt.sETtring(3,secretquestid);
       cstmt.sETtring(4,secretans);
       cstmt.registerOutParameter(5,Types.INTEGER);
       cstmt.execute();
       i=cstmt.getInt(5);
       if(i==1)
       {
          flag=true;
```

```
            con.commit();
        }
        else
        {
            flag=false;
            con.rollback();
        }

    con.close();
}
catch (SQLException ex)
{ex.printStackTrace();
    LoggerManager.writeLogSevere(ex);
    flag=false;
    try
    {
        con.rollback();
    }
    catch (SQLException sex)
    {
            LoggerManager.writeLogSevere(sex);
    }
}
    catch (Exception e)
    {e.printStackTrace();
        LoggerManager.writeLogSevere(e);
        flag=false;
        try
        {
            con.rollback();
        }
        catch (SQLException sex)
        {
                LoggerManager.writeLogSevere(sex);
        }
    }
    return flag;
}

 //Recover Password using Existed Question
public String recoverPasswordByQuestion(Profile regbean)
{
    String password;
    String loginid=regbean.getLoginID();
    String secretquestid=regbean.gETecretQuestionID();
    String secretans=regbean.gETecretAnswer();
    try
    {con=getConnection();
```

```
            con.setAutoCommit(true);
            CallableStatement cstmt=con.prepareCall("{call RecoverPassword(?,?,?,?)}");
            cstmt.sETtring(1,loginid);
            cstmt.sETtring(2,secretquestid);
            cstmt.sETtring(3,secretans);
            cstmt.registerOutParameter(4,Types.VARCHAR);
            cstmt.execute();
            password=cstmt.gETtring(4);
            con.close();
        }
        catch (SQLException ex)
        {ex.printStackTrace();
            LoggerManager.writeLogSevere(ex);
            password="";
        }
        catch (Exception e)
        {
            LoggerManager.writeLogSevere(e);
            password="";
        }
        return password;
    }
    public String  checkUser(String userName)
    {
            String user=null;
            System.out.println("username"+userName);
            try

        {con=getConnection();
            con.setAutoCommit(true);
            CallableStatement cstmt=con.prepareCall("{ call loginidavailablity(?,?) }");
            cstmt.sETtring(1, userName);
            cstmt.registerOutParameter(2,Types.VARCHAR);
            cstmt.execute();
            user=cstmt.gETtring(2);
            con.close();
        }
        catch (SQLException ex)
        {ex.printStackTrace();
            LoggerManager.writeLogSevere(ex);
            user=null;
        }
        catch (Exception e)
        {
            LoggerManager.writeLogSevere(e);
            user=null;
        }
        return user;
```

```
    }

}

/*
 * ProfileDAO.java
 *
 *
 *
 */

package com.dts.dae.dao;
import oracle.jdbc.driver.*;
import com.dts.dae.model.Profile;
import com.dts.core.dao.AbstractDataAccessObject;
import com.dts.core.util.CoreHash;
import com.dts.core.util.DateWrapper;
import com.dts.core.util.LoggerManager;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.*;
import java.util.Date;
import java.util.Properties;


/**
 *
 * @author
 */
public class ProfileDAO extends AbstractDataAccessObject{

    public Connection con;

    private boolean flag;
    /** Creates a new instance of ProfileDAO */
    public ProfileDAO()
    {
          //getting Database Connection

    }


    //User Registration
    public boolean registration(Profile regbean)
```

```
{

   String loginid=regbean.getLoginID();
   String password=regbean.getPassword();
   String firstname=regbean.getFirstName();
   String lastname=regbean.getLastName();
   String logintype=regbean.getLoginType();
   String regdate=regbean.getRegDate();
   String secretquest=regbean.gETecretQuestionID();
   String ownsecretquest=regbean.getOwnSecretQuestion();
   String secretans=regbean.gETecretAnswer();
   String bdate=DateWrapper.parseDate(regbean.getBirthDate());
  //home
   String hno=regbean.getHno();
   String home=regbean.getHome();
   String street=regbean.gETtreet();
   String city=regbean.getCity();
   String state=regbean.gETtate();
   String country=regbean.getCountry();
   String pin=regbean.getPin();
   String Phonetype=regbean.getHomePhoneType();
   String phone=regbean.getPhone();
   //office
   String ohno=regbean.getOhno();
   String office=regbean.getOffice();
   String ostreet=regbean.getOstreet();
   String ocity=regbean.getOcity();
   String ostate=regbean.getOstate();
   String ocountry=regbean.getOcountry();
   String opin=regbean.getOpin();
   String oPhonetype=regbean.getOfficePhoneType();
   String ophone=regbean.getOphone();
   //personal
   String phno=regbean.getChno();
   String contact=regbean.getContact();
   String pstreet=regbean.getCstreet();
   String pcity=regbean.getCcity();
   String pstate=regbean.getCstate();
   String pcountry=regbean.getCcountry();
   String ppin=regbean.getCpin();
   String pPhonetype=regbean.getPersonalPhoneType();
   String pphone=regbean.getCphone();
  System.out.println("homephoto= "+phone+" office= "+ophone+" personal = "+pphone);
   String fax=regbean.getFax();
   String email=regbean.getEmail();
   String photo=regbean.getPhoto();
   String newdate=DateWrapper.parseDate(new Date());
   try
```

```
        {
           System.out.println("photo="+photo);
           File f=new File(photo);
           FileInputStream fis=new FileInputStream(f);
           System.out.println("fole="+f.length());


           con=getConnection();
         // con.setAutoCommit(false);
          CallableStatement cstmt=con.prepareCall("{call
insertprocedure(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}");
         /*
  1 photo BLOB,
  2 fname userdetails.FIRSTNAME%type,
  3 lname userdetails.LASTNAME%type,
  4 db userdetails.DOB%type,
  5 logid userdetails.LOGINID%type,
  6 pass userdetails.PASSWORD%Type,
  7 secretquestion userdetails.FORGOTPWQUESTION%type,
  8 secretanswer userdetails.FORGOTPWANSWER%type,

  9 email userdetails.EMAILID%type,
  10 fax userdetails.FAXNO%type,
 */
          cstmt.setBinaryStream(1, fis,(int)f.length());
          cstmt.sETtring(2,firstname);
          cstmt.sETtring(3,lastname);
          cstmt.sETtring(4,bdate);

          cstmt.sETtring(5,loginid);
          cstmt.sETtring(6,password);

          cstmt.sETtring(7,secretquest);
          cstmt.sETtring(8,secretans);

          cstmt.sETtring(9,email);
          cstmt.sETtring(10,fax);
        /*
          //home
          11 addresshome addresses.ADDRESSTYPE%type,
          12 housenohome addresses.HOUSENO%type,
          13 streethome addresses.STREET%type,
          14 cityhome addresses.CITY%type,
          15 statehome addresses.STATE%type,
          16 countryhome addresses.COUNTRY%type,
          17 pincodehome addresses.PINCODE%type,
```

```
        */
        cstmt.sETtring(11,home);
        cstmt.sETtring(12,hno);
        cstmt.sETtring(13,street);
        cstmt.sETtring(14,city);
        cstmt.sETtring(15,state);
        cstmt.sETtring(16,country);
        cstmt.sETtring(17,pin);
       /*
         //office
 18 addressoffice addresses.ADDRESSTYPE%type,
 19 housenooffice addresses.HOUSENO%type,
 20 streetoffice addresses.STREET%type,
 21 cityoffice addresses.CITY%type,
 22 stateoffice addresses.STATE%type,
 23 countryoffice addresses.COUNTRY%type,
 24 pincodeoffice addresses.PINCODE%type,

         */
cstmt.sETtring(18,office);
        cstmt.sETtring(19,ohno);
        cstmt.sETtring(20,ostreet);
        cstmt.sETtring(21,ocity);
        cstmt.sETtring(22,ostate);
        cstmt.sETtring(23,ocountry);
        cstmt.sETtring(24,opin);
      /* //personal
        25 addresspersonal addresses.ADDRESSTYPE%type,
        26 housenopersonal addresses.HOUSENO%type,
        27 streetpersonal addresses.STREET%type,
        28 citypersonal  addresses.CITY%type,
        29 statepersonal addresses.STATE%type,
        30 countrypersonal addresses.COUNTRY%type,
        31 pincodepersonal addresses.PINCODE%type,
       */
        cstmt.sETtring(25,contact);
        cstmt.sETtring(26,phno);
        cstmt.sETtring(27,pstreet);
        cstmt.sETtring(28,pcity);
        cstmt.sETtring(29,pstate);
        cstmt.sETtring(30,pcountry);
        cstmt.sETtring(31,ppin);
       /*
        32 phonehome phones.PHONETYPE%type,
        33 phonenumberhome phones.PHONENO%type,
        34 phoneoffice phones.PHONETYPE%type,
        35 phonenumberoffice phones.PHONENO%type,
        36 phonepersonal phones.PHONETYPE%type,
```

```
   37 phonenumberpersonal phones.PHONENO%type
    */
   cstmt.sETtring(32,Phonetype);
   cstmt.sETtring(33,phone);
   cstmt.sETtring(34,oPhonetype);
   cstmt.sETtring(35,ophone);
   cstmt.sETtring(36,pPhonetype);
   cstmt.sETtring(37,pphone);

     int i= cstmt.executeUpdate();
     if(i==1)
     {
      flag=true;
     }
     else
     {
               flag=false;

     }
     con.close();
}
catch(SQLException e)
    {
          System.out.println(e.toString());
          if(e.toString().equalsIgnoreCase("java.sql.SQLException: [Microsoft][ODBC driver for
Oracle][Oracle]ORA-12571: TNS:packet writer failure"))
                    {
                    flag=true;
                    System.out.println("n==="+flag);
                    }
   System.out.println(e);

          }
       catch (Exception e)
       {
          e.printStackTrace();
          flag=false;
          try
          {
             con.rollback();
          }
          catch (SQLException se)
          {
             se.printStackTrace();
          }
       }
       return flag;
   }
```

```
    //Getting profile
    public Profile getProfile(String loginname)
    {
       Profile rb=new Profile();
       try
       {
          con=getConnection();
          CallableStatement cs=con.prepareCall("{call
showprofile(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}");

                    /*
                   1 logid userdetails.loginid%type,
                   2 pass out userdetails.PASSWORD%type,
                   3 fname OUT userdetails.FIRSTNAME%type,
                   4 lname OUT  userdetails.LASTNAME%type,
                   5 db OUT varchar2,

                   6 email OUT  userdetails.EMAILID%type,
                   7 fax OUT userdetails.FAXNO%type,
                   8 addresshome OUT addresses.ADDRESSTYPE%type,
                   9 housenohome OUT addresses.HOUSENO%type,
                   10 streethome OUT addresses.STREET%type,
                   11 cityhome OUT addresses.CITY%type,
                   12 statehome OUT addresses.STATE%type,
                   13 countryhome OUT addresses.COUNTRY%type,
                   14 pincodehome OUT addresses.PINCODE%type,
                   15 addressoffice OUT addresses.ADDRESSTYPE%type,
                   16 housenooffice OUT addresses.HOUSENO%type,
                   17 streetoffice OUT addresses.STREET%type,
                   18 cityoffice OUT  addresses.CITY%type,
                   19 stateoffice OUT addresses.STATE%type,
                   20 countryoffice OUT addresses.COUNTRY%type,
                   21 pincodeoffice OUT addresses.PINCODE%type,
                   22 addresspersonal OUT addresses.ADDRESSTYPE%type,
                   23 housenopersonal OUT addresses.HOUSENO%type,
                   24 streetpersonal OUT addresses.STREET%type,
                   25 citypersonal OUT addresses.CITY%type,
                   26 statepersonal OUT addresses.STATE%type,
                   27 countrypersonal OUT addresses.COUNTRY%type,
                   28 pincodepersonal OUT addresses.PINCODE%type,
                   29 phonehome OUT phones.PHONETYPE%type,
                   30 phonenohome OUT phones.PHONENO%type,
                   31 phoneoffice OUT  phones.PHONETYPE%type,
                   32 phonenooffice OUT phones.PHONENO%type,
                   33 phonepersonal OUT phones.PHONETYPE%type,
                   34 phonenopersonal OUT phones.PHONENO%type,
                   35 photo OUT  userdetails.PHOTOGRAPH%type
```

```
 */


cs.sETtring(1,loginname);
        cs.registerOutParameter(2,Types.VARCHAR);
        cs.registerOutParameter(3,Types.VARCHAR);
        cs.registerOutParameter(4,Types.VARCHAR);
        cs.registerOutParameter(5,Types.VARCHAR);
        cs.registerOutParameter(6,Types.VARCHAR);
        cs.registerOutParameter(7,Types.VARCHAR);
        cs.registerOutParameter(8,Types.VARCHAR);
        cs.registerOutParameter(9,Types.VARCHAR);
        cs.registerOutParameter(10,Types.VARCHAR);
        cs.registerOutParameter(11,Types.VARCHAR);
        cs.registerOutParameter(12,Types.VARCHAR);
        cs.registerOutParameter(13,Types.VARCHAR);
        cs.registerOutParameter(14,Types.VARCHAR);
        cs.registerOutParameter(15,Types.VARCHAR);
        cs.registerOutParameter(16,Types.VARCHAR);
        cs.registerOutParameter(17,Types.VARCHAR);
        cs.registerOutParameter(18,Types.VARCHAR);
        cs.registerOutParameter(19,Types.VARCHAR);
        cs.registerOutParameter(20,Types.VARCHAR);
        cs.registerOutParameter(21,Types.VARCHAR);
        cs.registerOutParameter(22,Types.VARCHAR);
        cs.registerOutParameter(23,Types.VARCHAR);
        cs.registerOutParameter(24,Types.VARCHAR);
        cs.registerOutParameter(25,Types.VARCHAR);
        cs.registerOutParameter(26,Types.VARCHAR);
        cs.registerOutParameter(27,Types.VARCHAR);
        cs.registerOutParameter(28,Types.VARCHAR);
        cs.registerOutParameter(29,Types.VARCHAR);
        cs.registerOutParameter(30,Types.VARCHAR);
        cs.registerOutParameter(31,Types.VARCHAR);
        cs.registerOutParameter(32,Types.VARCHAR);
        cs.registerOutParameter(33,Types.VARCHAR);
        cs.registerOutParameter(34,Types.VARCHAR);
        cs.registerOutParameter(35,Types.BLOB);
        cs.execute();
        rb.setPassword(cs.gETtring(2));
        rb.setFirstname(cs.gETtring(3));
        rb.setLastname(cs.gETtring(4));

        rb.setBdate(cs.gETtring(5));
        //rb.setPhoto(cs.gETtring());
        rb.setEmail(cs.gETtring(6));
        rb.setFax(cs.gETtring(7));
        rb.setHome(cs.gETtring(8));
```

```
            rb.setHno(cs.gETtring(9));
            rb.sETtreet(cs.gETtring(10));
            rb.setCity(cs.gETtring(11));
            rb.sETtate(cs.gETtring(12));
            //rb.setPin(cs.gETtring(13));
            rb.setCountry(cs.gETtring(13));
            rb.setPin(cs.gETtring(14));
            rb.setOffice(cs.gETtring(15));
            rb.setOhno(cs.gETtring(16));
            rb.setOstreet(cs.gETtring(17));
            rb.setOcity(cs.gETtring(18));

            rb.setOstate(cs.gETtring(19));
/*
 20 countryoffice OUT addresses.COUNTRY%type,
21 pincodeoffice OUT addresses.PINCODE%type,
22 addresspersonal OUT addresses.ADDRESSTYPE%type,
23 housenopersonal OUT addresses.HOUSENO%type,
24 streetpersonal OUT addresses.STREET%type,
25 citypersonal OUT addresses.CITY%type,
26 statepersonal OUT addresses.STATE%type,
27 countrypersonal OUT addresses.COUNTRY%type,
28 pincodepersonal OUT addresses.PINCODE%type,
*/
        rb.setOcountry(cs.gETtring(20));
        rb.setOpin(cs.gETtring(21));
        rb.setContact(cs.gETtring(22));
        rb.setChno(cs.gETtring(23));
        rb.setCstreet(cs.gETtring(24));
        rb.setCcity(cs.gETtring(25));
        rb.setCstate(cs.gETtring(26));
        //rb.setCpin(cs.gETtring(27));
        rb.setCcountry(cs.gETtring(27));
        rb.setCpin(cs.gETtring(28));
        /*
                29 phonehome OUT phones.PHONETYPE%type,
                30 phonenohome OUT phones.PHONENO%type,
                31 phoneoffice OUT  phones.PHONETYPE%type,
                32 phonenooffice OUT phones.PHONENO%type,
                33 phonepersonal OUT phones.PHONETYPE%type,
                34 phonenopersonal OUT phones.PHONENO%type,
                35 photo OUT  userdetails.PHOTOGRAPH%type
                        */
        rb.setHomePhoneType(cs.gETtring(29));
        rb.setPhone(cs.gETtring(30));
        rb.setOfficePhoneType(cs.gETtring(31));
```

```
                String o=cs.gETtring(32);
                rb.setOphone(o);
                rb.setPersonalPhoneType(cs.gETtring(33));
            String s=cs.gETtring(34);
            rb.setCphone(s);
            System.out.println("personal phone="+s+"  officeph= "+o);
             /*FileOutputStream fs = null;
InputStream is = null;
fs = new FileOutputStream("photo.jpg");

is = cs.getBlob(35).getBinaryStream();
byte[] buf = new byte[16384];
int bytes;
while ((bytes = is.read(buf)) != -1) {
   fs.write(buf, 0, bytes);

}*/
            Blob b =cs.getBlob(35);
            byte b1[]=b.getBytes(1,(int)b.length());
            OutputStream fout=new FileOutputStream("C:/photo.jpg");
            fout.write(b1);

  }
   catch(Exception e)
   {e.printStackTrace();
      LoggerManager.writeLogSevere(e);
   }
   finally
   {
      try{
               con.close();
      }catch(Exception e)
      {
               LoggerManager.writeLogSevere(e);
      }

   }
   return rb;
}    // Modify Profile
public boolean modifyProfile(Profile regbean)
{
      String loginid=regbean.getLoginID();
   String firstname=regbean.getFirstName();
   String lastname=regbean.getLastName();
  // String bdate=DateWrapper.parseDate(regbean.getBirthDate());
  //home
   String hno=regbean.getHno();
```

```
String home=regbean.getHome();
String street=regbean.gETtreet();
String city=regbean.getCity();
String state=regbean.gETtate();
String country=regbean.getCountry();
String pin=regbean.getPin();
String Phonetype=regbean.getHomePhoneType();
String phone=regbean.getPhone();
//office
String ohno=regbean.getOhno();
String office=regbean.getOffice();
String ostreet=regbean.getOstreet();
String ocity=regbean.getOcity();
String ostate=regbean.getOstate();
String ocountry=regbean.getOcountry();
String opin=regbean.getOpin();
String oPhonetype=regbean.getOfficePhoneType();
String ophone=regbean.getOphone();
//personal
String phno=regbean.getChno();
String contact=regbean.getContact();
String pstreet=regbean.getCstreet();
String pcity=regbean.getCcity();
String pstate=regbean.getCstate();
String pcountry=regbean.getCcountry();
String ppin=regbean.getCpin();
String pPhonetype=regbean.getPersonalPhoneType();
String pphone=regbean.getCphone();

String fax=regbean.getFax();
String email=regbean.getEmail();
String photo=regbean.getPhoto();
String newdate=DateWrapper.parseDate(new Date());
try
{
  System.out.println("photo="+photo);
  File f=new File(photo);
  FileInputStream fis=new FileInputStream(f);
  System.out.println("fole="+f.length());

  con=getConnection();
 // con.setAutoCommit(false);
  CallableStatement cs=con.prepareCall("{call
changeprofile(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}");

  /*

          1 fname userdetails.FIRSTNAME%type,
```

```
                2 lname userdetails.LASTNAME%type,
                3 logid userdetails.LOGINID%type,
                4 photo userdetails.photograph%type,
                5 email userdetails.EMAILID%type,
                6 fax userdetails.FAXNO%type,

 */
cs.sETtring(1,firstname);
          cs.sETtring(2,lastname);
          cs.sETtring(3,loginid);
          cs.setBinaryStream(4, fis,(int)f.length());
          // cs.setBinaryStream(4, fis,999);
          cs.sETtring(5,email);
          cs.sETtring(6,fax);

          /*
    7 addresshome addresses.ADDRESSTYPE%type,
                8 housenohome addresses.HOUSENO%type,
                9 streethome addresses.STREET%type,
                10 cityhome addresses.CITY%type,
                11 statehome addresses.STATE%type,
                12 countryhome addresses.COUNTRY%type,
                13 pincodehome addresses.PINCODE%type,
 */

cs.sETtring(7,home);
cs.sETtring(8,hno);
cs.sETtring(9,street);
cs.sETtring(10,city);
cs.sETtring(11,state);
cs.sETtring(12,country);
cs.sETtring(13,pin);


/*
 14  addressoffice addresses.ADDRESSTYPE%type,
           15  housenooffice addresses.HOUSENO%type,
           16  streetoffice addresses.STREET%type,
           17  cityoffice addresses.CITY%type,
           18  stateoffice addresses.STATE%type,
           19  countryoffice addresses.COUNTRY%type,
           20 pincodeoffice addresses.PINCODE%type,

 */
cs.sETtring(14,office);

cs.sETtring(15,ohno);
```

```
cs.sETtring(16,ostreet);
cs.sETtring(17,ocity);
cs.sETtring(18,ostate);
cs.sETtring(19,ocountry);
cs.sETtring(20,opin);

/*

21 addresspersonal addresses.ADDRESSTYPE%type,
   22 housenopersonal addresses.HOUSENO%type,
   23 streetpersonal addresses.STREET%type,
   24 citypersonal  addresses.CITY%type,
   25 statepersonal addresses.STATE%type,
   26 countrypersonal addresses.COUNTRY%type,
   27 pincodepersonal addresses.PINCODE%type,

 */
cs.sETtring(21,contact);
cs.sETtring(22,phno);
cs.sETtring(23,pstreet);
cs.sETtring(24,pcity);
cs.sETtring(25,pstate);
cs.sETtring(26,pcountry);
      cs.sETtring(27,ppin);

      /*
28 phonehome phones.PHONETYPE%type,
29 phonenohome phones.PHONENO%type,
30 phoneoffice phones.PHONETYPE%type,
31 phonenooffice phones.PHONENO%type,
32 phonepersonal phones.PHONETYPE%type,
33 phonenopersonal phones.PHONENO%type,
34 flag out number

*/

      cs.sETtring(28,Phonetype);
      cs.sETtring(29,phone);
      cs.sETtring(30,oPhonetype);
      cs.sETtring(31,ophone);
      cs.sETtring(32,pPhonetype);
      cs.sETtring(33,pphone);
      cs.registerOutParameter(34,Types.INTEGER);
            cs.execute();
      int n=cs.getInt(34);
      if(n>0)
      {
            flag=true;
```

```
    }

    else
    {
       flag=false;
    con.rollback();
    }
    con.close();
}
catch (SQLException ex)
{
    ex.printStackTrace();
    flag=false;
    try
    {
       con.rollback();
    }
    catch (SQLException sex)
    {
       sex.printStackTrace();
    }
}
catch (Exception e)
    {
       e.printStackTrace();
       flag=false;
       try
       {
          con.rollback();
       }
       catch (SQLException se)
       {
          se.printStackTrace();
       }
    }
    return flag;
 }
//getreport fromdate todate
 public CoreHash getReportFromTo(String sdate,String edate)
       {

                Properties p=getProperties();
                String pattern="";
                if(p.getProperty("dbname").equals("access"))
                        pattern="#";
                CoreHash aCoreHash = new CoreHash();
                aCoreHash.clear();
                System.out.println(" aCoreHash--"+aCoreHash.isEmpty());
```

```
                int sno=1;
                Statement st;
                Profile aProfile=null;
                try {
                        con = getConnection();

                 st=con.createStatement();
                 ResultSet rs=st.executeQuery("SELECT DISTINCT(UD.LOGINID
),UD.FIRSTNAME,UD.LASTNAME,UD.DOB,UD.DOR,UD.EMAILID FROM USERDETAILS
UD,LOGINMASTER LM WHERE UD.USERID=LM.USERID AND TO_CHAR(LOGINDATE,'DD-
MM-YYYY')>='"+sdate+"' AND TO_CHAR(LOGINDATE,'DD-MM-YYYY')<='"+edate+"'");
                 while(rs.next())
            {
                  aProfile=new Profile();

                  aProfile.setLoginID(rs.gETtring(1));
                  aProfile.setFirstname(rs.gETtring(2));
                  aProfile.setLastName(rs.gETtring(3));
                  String date=DateWrapper.parseDate(rs.getDate(4)).trim();
                 aProfile.setBirthDate(date);
              aProfile.setRegDate(DateWrapper.parseDate(rs.getDate(5)).trim());
              aProfile.setEmail(rs.gETtring(6));

                 aCoreHash.put(new Integer(sno),aProfile);
                        sno++;

            }
                }
                catch(Exception e)
                {e.printStackTrace();
                        LoggerManager.writeLogWarning(e);
                }
                finally
                {
                 try{
                        if(con!=null)
                                con.close();

                 }
                 catch(Exception e){}
                }
                return aCoreHash;
        }
//presentloginuser getting

public CoreHash getReportPresent()
        {
```

```
                CoreHash aCoreHash = new CoreHash();
                aCoreHash.clear();
                System.out.println("aCoreHash--"+aCoreHash.isEmpty());
                int sno=1;
                CallableStatement cstmt;
                Profile aProfile=null;
                try {
                        con = getConnection();


                 cstmt=con.prepareCall("{call
REF_CURSOR_TEST.GET_ACCOUNTS_PROCEDURE(?)}");
                 cstmt.registerOutParameter(1,OracleTypes.CURSOR);
                 cstmt.executeUpdate();
                 ResultSet rs=((OracleCallableStatement)cstmt).getCursor(1);
                  while(rs.next())
            {
                   aProfile=new Profile();

                   aProfile.setLoginID(rs.gETtring(1));
                   aProfile.setFirstname(rs.gETtring(2));
                   aProfile.setLastName(rs.gETtring(3));
                   String date=DateWrapper.parseDate(rs.getDate(4)).trim();
                aProfile.setBirthDate(date);
              aProfile.setRegDate(DateWrapper.parseDate(rs.getDate(5)).trim());
              aProfile.setEmail(rs.gETtring(6));


                aCoreHash.put(new Integer(sno),aProfile);
                        sno++;

          }
                }
                catch(Exception e)
                {e.printStackTrace();
                        LoggerManager.writeLogWarning(e);
                }
                finally
                {
                 try{
                         if(con!=null)
                                 con.close();

                 }
                 catch(Exception e){}
                }
```

```
                return aCoreHash;
        }

    //get All users
    public CoreHash getAllUsers()
        {


                CoreHash aCoreHash = new CoreHash();
                aCoreHash.clear();
                System.out.println("aCoreHash--"+aCoreHash.isEmpty());
                int sno=1;
                Statement pstmt;
                Profile aProfile=null;
                try {
                        con = getConnection();
                        pstmt=con.createStatement();



                        ResultSet rs=pstmt.executeQuery("SELECT DISTINCT(UD.LOGINID
),UD.FIRSTNAME,UD.LASTNAME,UD.DOB,UD.DOR,UD.EMAILID FROM USERDETAILS UD");
          while(rs.next())
{
          aProfile=new Profile();

          aProfile.setLoginID(rs.gETtring(1));
          aProfile.setFirstname(rs.gETtring(2));
          aProfile.setLastName(rs.gETtring(3));
          String date=DateWrapper.parseDate(rs.getDate(4)).trim();
       aProfile.setBirthDate(date);
  aProfile.setRegDate(DateWrapper.parseDate(rs.getDate(5)).trim());
  aProfile.setEmail(rs.gETtring(6));


        aCoreHash.put(new Integer(sno),aProfile);
                    sno++;

}
        }
        catch(Exception e)
        {e.printStackTrace();
                LoggerManager.writeLogWarning(e);
        }
        finally
        {
         try{
```

```
                        if(con!=null)
                                con.close();

                    }
                    catch(Exception e){}
                   }
                   return aCoreHash;
          }


}
package com.dts.dae.action;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.dts.dae.dao.SecurityDAO;
import com.dts.dae.model.Profile;

public class ChangePasswordAction extends HttpServlet {

          /**
           * The doPost method of the servlet. <br>
           *
           * This method is called when a form has its tag value method equals to post.
           *
           * @param request the request send by the client to the server
           * @param response the response send by the server to the client
           * @throws ServletException if an error occurred
           * @throws IOException if an error occurred
           */
          public void doPost(HttpServletRequest request, HttpServletResponse response)
                              throws ServletException, IOException {

                    HttpSession session =request.gETession();

                    if((String)session.getAttribute("user")==null)
                 response.sendRedirect("LoginForm.jsp?status=Session Expired");

           String page="";
          /*
```

```
        if(((String)session.getAttribute("role")).equals("user"))
        {
            page="userhome.jsp";
         }else if(((String)session.getAttribute("role")).equals("admin")){
            page="adminhome.jsp";
        }
         else
                    page="LoginForm.jsp?status=Session Expired";
*/
        String target="Changepassword.jsp?status=Password Changed Failed";
                   Profile rb=new Profile();
        rb.setPassword(request.getParameter("oldpassword"));
        rb.setLoginID(request.getParameter("username"));
        rb.setNewPassword(request.getParameter("newpassword"));

        boolean flag=new SecurityDAO().changePassword(rb);

        if(flag)
           target="userhome.jsp?status=Password Changed Successfully";
        else
           target="userhome.jsp?status=Password Changed Failed";
        RequestDispatcher rd = request.getRequestDispatcher(target);
        rd.forward(request,response);
                }

        }
```

TESTING

Testing Concepts

- *Testing*
- *Testing Methodologies*
  - Black box Testing:
  - White box Testing.
  - Gray Box Testing.
- *Levels of Testing*
  - Unit Testing.
  - Module Testing.
  - Integration Testing.
  - System Testing.
  - User Acceptance Testing.
- *Types Of Testing*
  - Smoke Testing.
  - Sanitary Testing.
  - Regression Testing.
  - Re-Testing.
  - Static Testing.
  - Dynamic Testing.
  - Alpha-Testing.

- Beta-Testing.
- Monkey Testing.
- Compatibility Testing.
- Installation Testing.
- Adhoc Testing.
- Ext….

TCD (Test Case Documentation)

- *STLC*
  - Test Planning.
  - Test Development.
  - Test Execution.
  - Result Analysis.
  - Bug-Tracing.
  - Reporting.

- *Microsoft Windows – Standards*
- *Manual Testing*
- *Automation Testing (Tools)*
  - Win Runner.
  - Test Director.

Testing:

- The process of executing a system with the intent of finding an error.
- Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
- Quality is defined as justification of the requirements
- Defect is nothing but deviation from the requirements
- Defect is nothing but bug.

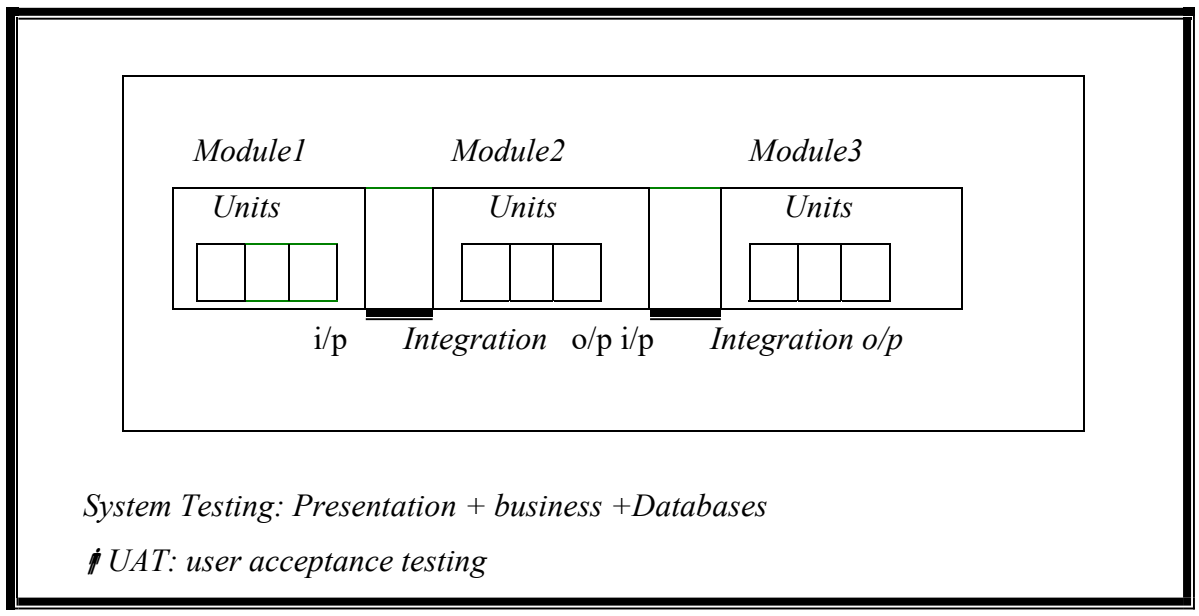
- Testing --- The presence of bugs
- Testing can demonstrate the presence of bugs, but not their absence
- Debugging and Testing are not the same thing!
- Testing is a systematic attempt to break a program or the AUT
- Debugging is the art or method of uncovering why the script /program did not execute properly.

Testing Methodologies:

- Black box Testing: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.
  Usually Test Engineers are involved in the black box testing.
- White box Testing: is the testing process in which tester can perform testing on an application with having internal structural knowledge.
  Usually The Developers are involved in white box testing.

- Gray Box Testing: is the process in which the combination of black box and white box tonics' are used.

Levels of Testing:



STLC (SOFTWARE TESTING LIFE CYCLE)

Test Planning:

1.Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.

2.This document involves the scope of testing,

3. Objective of testing,

4. Areas that need to be tested,

5. Areas that should not be tested,

6. Scheduling Resource Planning,

7. Areas to be automated, various testing tools Used….

Test Development:

1. Test case Development (check list)
2. Test Procedure preparation. (Description of the Test cases).

1. Implementation of test cases.

Observing the result.

Result Analysis:
1. Expected value: is nothing but expected behavior Of application.
2. Actual value: is nothing but actual behavior of application

Bug Tracing: Collect all the failed cases, prepare documents.

Reporting: Prepare document (status of the application)

Types Of Testing:

> Smoke Testing: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

> Sanity Testing: is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

> Regression Testing: is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

>Re-Testing: is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

Static Testing: is the testing, which is performed on an application when it is not been executed.ex: GUI, Document Testing

Dynamic Testing: is the testing which is performed on an application when it is being executed.ex: Functional testing.

Alpha Testing: it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

Beta-Testing: it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

Monkey Testing: is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.

Compatibility testing: it is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers…etc) In order to check how far the product is compatible with all these environments platform combination.

Installation Testing: it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

Adhoc Testing: Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmotic issues.

TCD (Test Case Document:

Test Case Document Contains

- Test Scope (or) Test objective
- Test Scenario
- Test Procedure
- Test case

This is the sample test case document for the Acadamic details of student project:

Test scope:

- Test coverage is provided for the screen " Acadamic status entry" form of a student module of university management system application
- Areas of the application to be tested

Test Scenario:

- When the office personals use this screen for the marks entry, calculate the status details, saving the information on student's basis and quit the form.

Test Procedure:

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively

Test Cases:

- Template for Test Case

T.C.No	Description	Exp	Act	Result

Guidelines for Test Cases:

1. GUI Test Cases:

- Total no of features that need to be check
- Look & Feel
- Look for Default values if at all any (date & Time, if at all any require)
- Look for spell check

Example for Gui Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Check for all the features in the screen	The screen must contain all the features		
2	Check for the alignment of the objects as per the validations	The alignment should be in proper way		

2. Positive Test Cases:

- The positive flow of the functionality must be considered
- Valid inputs must be used for testing
- Must have the positive perception to verify whether the requirements are justified.

Example for Positive Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Check for the date Time Auto Display	The date and time of the system must be displayed		
2	Enter the valid Roll no into the student roll no field	It should accept		

3. Negative Test Cases:

- Must have negative perception.
- Invalid inputs must be used for test.

Example for Negative Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Try to modify The information in date and time	Modification should not be allow		
2	Enter invalid data in to the studen details form, click on save	It should no accept invalid data, save should no allow		

OUTPUT SCREENS

rakesh-pc:8080/ORS/ | My JSP 'Header.jsp' start | My JSP 'articles.jsp' start | ONLINE BOOK SHOP

rakesh-pc:8080/OnlineBookStore/

HOME CONTACTUS LOGIN

ONLINE BOOK SHOP

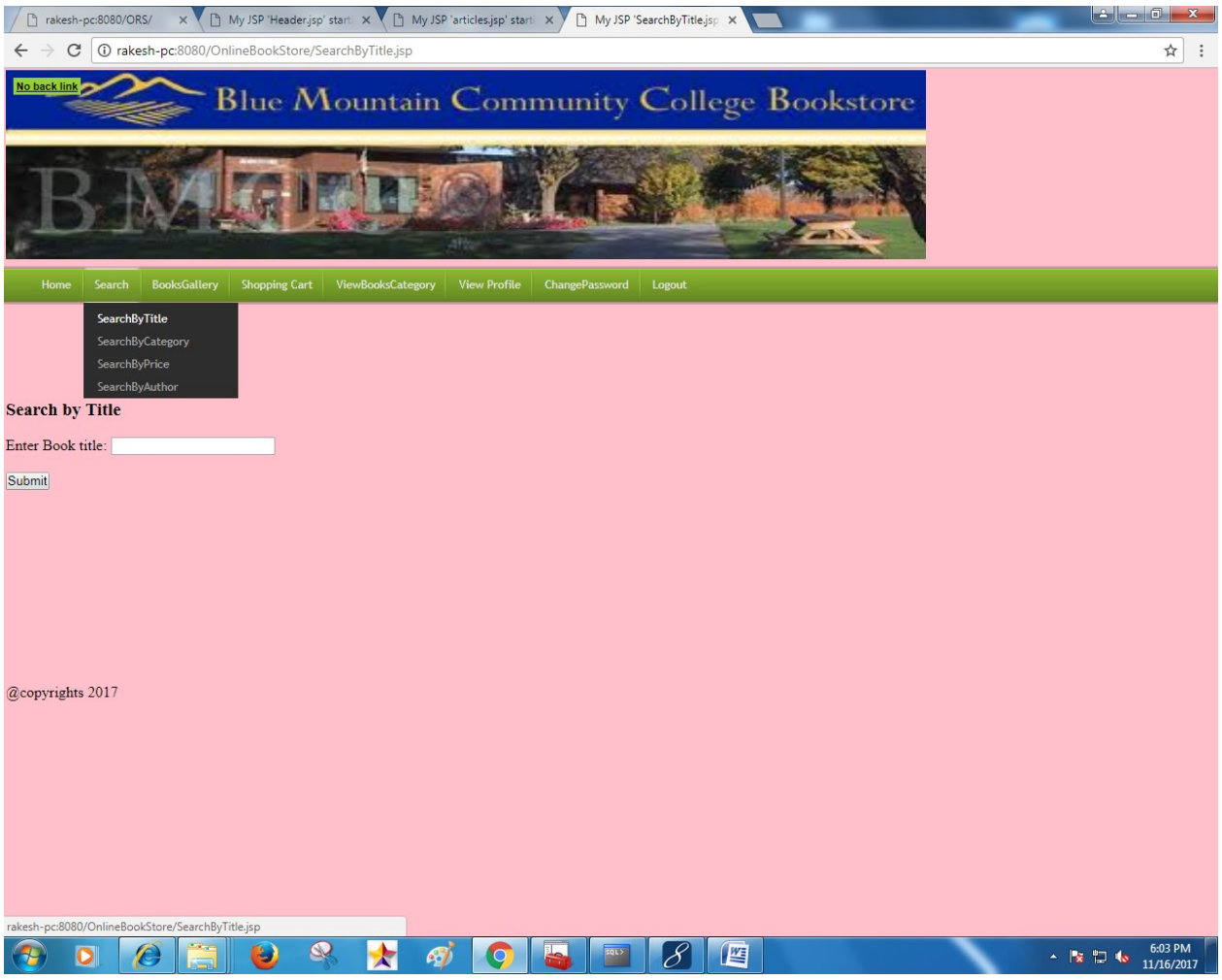
Online shopping or online retailing is a form of electronic commerce which allows consumers to directly buy goods or services from a seller over the Internet using a web browser.
Alternative names are: e-web-store, e-shop, e-store, Internet shop, web-shop, web-store, online store, and virtual store. An online shop evokes the physical analogy of buying products or services at a bricks-and-mortar retailer or shopping center; the process is called business-to-consumer (B2C) online shopping. In the case where a business buys from another business, the process is called business-to-business (B2B) online shopping. The largest of these online retailing corporations are eBay and Amazon.com, both based in the United States. Retail success is no longer all about physical stores, this is evident because of the increase in retailers now offering online store interfaces for consumers. With the growth of online shopping, comes a wealth of new market footprint coverage opportunities for stores that can appropriately cater to offshore market demands and service requirements.

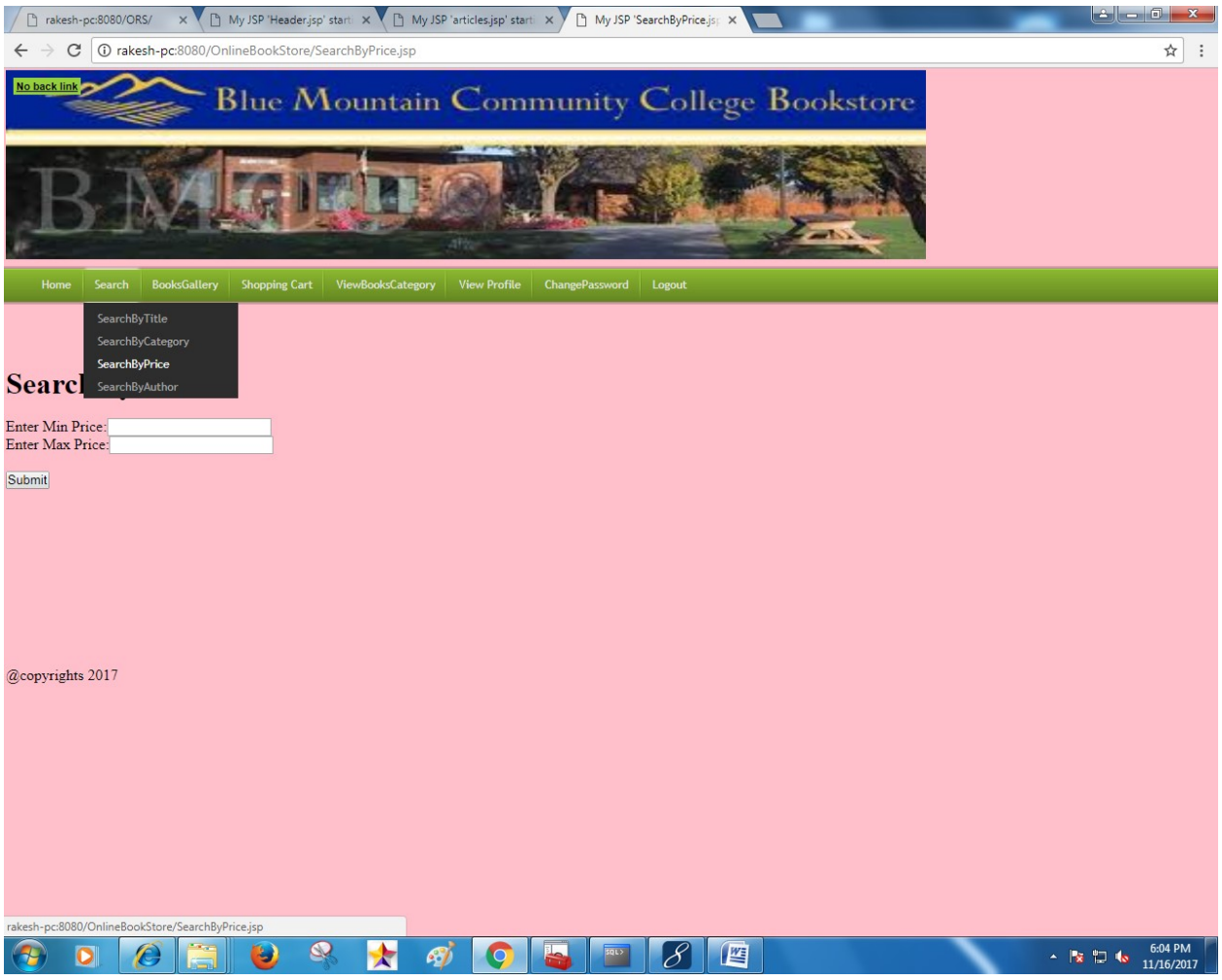
New Items Arrived

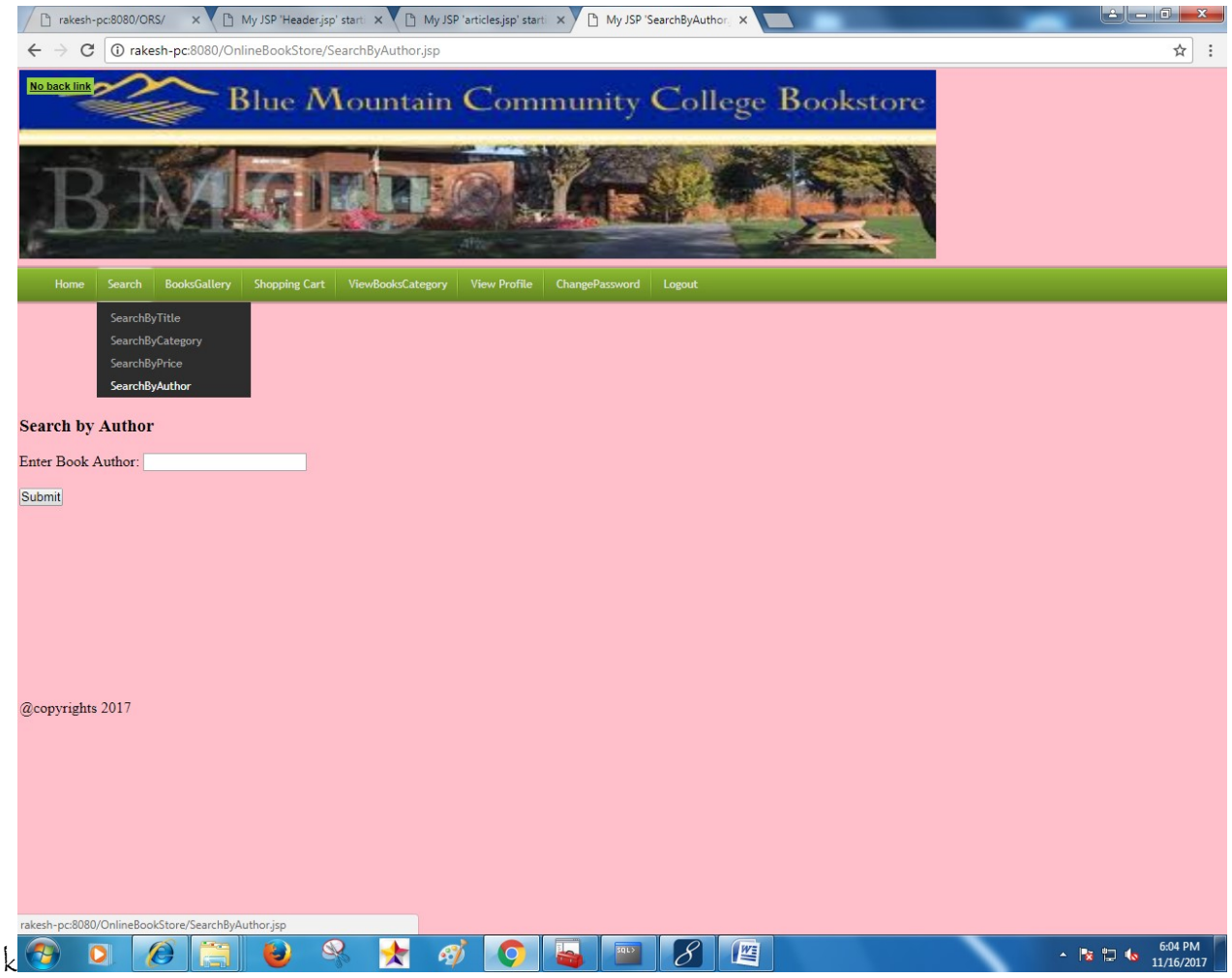
GALLERY

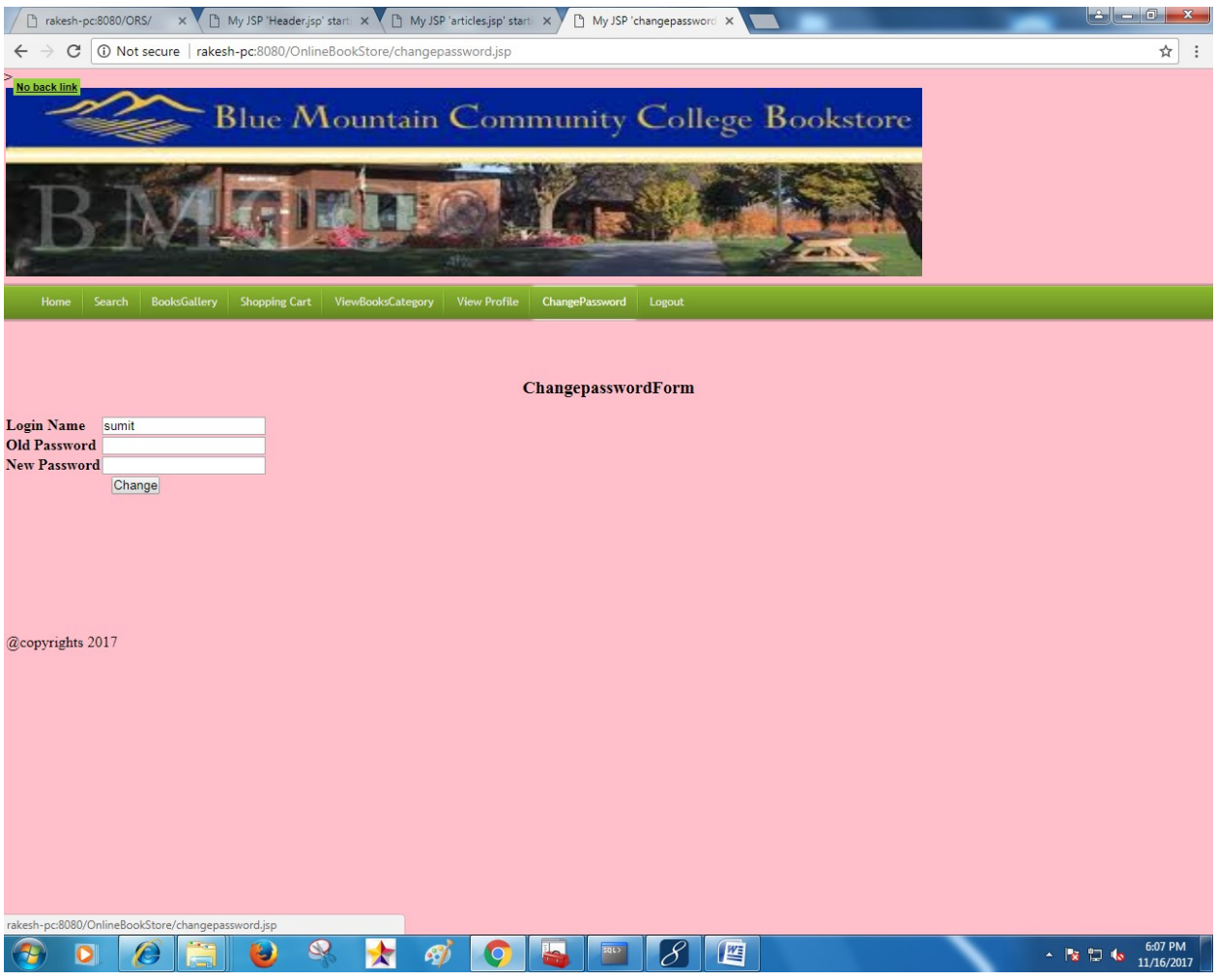
6:01 PM
11/16/2017


rakesh-pc:8080/ORS/
My JSP 'Header.jsp' start
My JSP 'articles.jsp' start
My JSP 'index.jsp' startin
Not secure
rakesh-pc:8080/OnlineBookStore/LoginForm.jsp
No back link
Blue Mountain Community College Bookstore
Home
Products
Help
Login
LoginForm
Username :
Password :
Login
reset
SignUP
Forgot Password
@copyrights 2017
rakesh-pc:8080/OnlineBookStore/LoginForm.jsp
6:02 PM
11/16/2017


rakesh-pc:8080/ORS/
My JSP 'Header.jsp' start
My JSP 'articles.jsp' start
My JSP 'SearchByTitle.jsp
rakesh-pc:8080/OnlineBookStore/SearchByTitle.jsp
No back link
Blue Mountain Community College Bookstore
Home
Search
BooksGallery
Shopping Cart
ViewBooksCategory
View Profile
ChangePassword
Logout
SearchByTitle
SearchByCategory
SearchByPrice
SearchByAuthor
Search by Title
Enter Book title:
Submit
@copyrights 2017
rakesh-pc:8080/OnlineBookStore/SearchByTitle.jsp
6:03 PM
11/16/2017


rakesh-pc:8080/ORS/
My JSP 'Header.jsp' start
My JSP 'articles.jsp' start
My JSP 'SearchByPrice.js
rakesh-pc:8080/OnlineBookStore/SearchByPrice.jsp
No back link
Blue Mountain Community College Bookstore
Home
Search
BooksGallery
Shopping Cart
ViewBooksCategory
View Profile
ChangePassword
Logout
SearchByTitle
SearchByCategory
SearchByPrice
SearchByAuthor
Search
Enter Min Price:
Enter Max Price:
Submit
@copyrights 2017
rakesh-pc:8080/OnlineBookStore/SearchByPrice.jsp
6:04 PM
11/16/2017


rakesh-pc:8080/ORS/
My JSP 'Header.jsp' start
My JSP 'articles.jsp' start
My JSP 'SearchByAuthor
rakesh-pc:8080/OnlineBookStore/SearchByAuthor.jsp
No back link
Blue Mountain Community College Bookstore
Home
Search
BooksGallery
Shopping Cart
ViewBooksCategory
View Profile
ChangePassword
Logout
SearchByTitle
SearchByCategory
SearchByPrice
SearchByAuthor
Search by Author
Enter Book Author:
Submit
@copyrights 2017
rakesh-pc:8080/OnlineBookStore/SearchByAuthor.jsp
6:04 PM
11/16/2017


rakesh-pc:8080/ORS/
My JSP 'Header.jsp' start
My JSP 'articles.jsp' start
My JSP 'changepassword
Not secure
rakesh-pc:8080/OnlineBookStore/changepassword.jsp
No back link
Blue Mountain Community College Bookstore
Home
Search
BooksGallery
Shopping Cart
ViewBooksCategory
View Profile
ChangePassword
Logout
ChangepasswordForm
Login Name
sumit
Old Password
New Password
Change
@copyrights 2017
rakesh-pc:8080/OnlineBookStore/changepassword.jsp
6:07 PM
11/16/2017

LIMITATIONS AND SCOPE FOR FUTURE ENHANCEMENTS:

Limitations of the system:.

- System works in all platforms and its compatible environments.
- Advanced techniques are not used to check the authorization.

Future Enhancements:

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system and can adaptable to environment.
- Because it is based on object-oriented design, any further changes can be easily adaptable.
- Based on the future security issues, security can be improved using emerging technologies.sub admin module can be added

PROJECT SUMMARY

This application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required options, which can be utilized by the user to perform the desired operations.

The software is developed using Java as front end and Oracle as back end in Windows environment. The goals that are achieved by the software are:

- ✓ Optimum utilization of resources.
- ✓ Efficient management of records.
- ✓ Simplification of the operations.
- ✓ Less processing time and getting required information.
- ✓ User friendly

CONCLUSION

WORK DONE:

The Online Book Shop was successfully designed and is tested for accuracy and quality.

During this project we have accomplished all the objectives and this project meET the needs of the organization. The developed will be used in searching, retrieving and generating information for the concerned requests.

GOALS

- ✓ Reduced entry work
- ✓ Easy retrieval of information
- ✓ Reduced errors due to human intervention
- ✓ User friendly screens to enter the data
- ✓ Portable and flexible for further enhancement
- ✓ Web enabled.
- ✓ Fast finding of information requested

BIBILIOGRAPHY

(1) Java Complete Reference by Herbert Shield

(2) Database Programming with JDBC and Java by George Reese

(3) Java and XML By Brett McLaughlin

(4) Wikipedia, URL: http://www.wikipedia.org.

(5) Answers.com, Online Dictionary, Encyclopedia and much more, URL: http://www.answers.com

(6) Google, URL: http://www.google.co.in

(7)Project Management URL: http://www.startwright.com/project.htm