**Chapter-1**

**Study of Existing system and system requirements.**

**Hardware & Software Requirement :**

**Hardware Interfaces**

* Minimum Hardware requirement
* Processor: P4 3.0 GHz
* RAM:1 GB or Higher
* Monitor
* Mouse
* Hard disk: 80 GB

**Software Interfaces**

* Minimum Software requirement
* Java (JSP and Servlet)
* Glassfish Server

All these types of software automatic configure inside operating system after installation it having Java, MySQL, Apache and operating system base configuration file, it doesn’t need to configure manually.

**1.1 Project Defination:**

* **Library Management System**

Its a web application.

* The Proposed Project is named as **“Library Management System”**.

“ Project management is the discipline of planning, organization and managing resources to bring about the successful completion of the specific project…..

**1.2 Project Introduction:**

Library management system is a project which aims in developing a computerized system to maintain all the daily work of library .This project has many features which are generally not availiable in normal library management systems like facility of user login and a facility of teachers login .It also has a facility of admin login through which the admin can monitor the whole system .It also has facility of an online notice board where teachers can student can put up information about workshops or seminars being held in our colleges or nearby colleges and librarianafter proper verification from the concerned institution organizing the seminar can add it to the notice board . It has also a facility where student after logging in their accounts can see list of books issued and its issue date and return date and also the students can request the librarian to add new books by filling the book request form. The librarian after logging into his account ie admin account can generate various reports such as student report , issue report, teacher report and book report Overall this project of ours is being developed to help the students as well as staff of library to maintain the library in the best way possible and also reduce the human efforts.

**1.3 what is the existing system?**

To solve these problems they required a computerized system to handle all the works. They required a web based application that will provide a working environment that will be flexible and will provide ease of work and will reduce the time for report generation and other paperworks.

**1.4 Scope & Goal:**

The main purpose behind the proposed system is to provide a comprehensive computerized system, which can capture, collate and analyze the data from these wards and evaluate the impact of the program.

## Project Goals and Objectives

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

* Online book issue
* Request column for librarian for providing new books
* A separate column for digital library
* Student login page where student can find books issued by him/her and date of return.
* A search column to search availability of books

**1.5 How the existing system works?**

Library Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian to manage the library using a computerized system where he/she can record various transactions like issue of books, return of books, addition of new books, addition of new students etc.

1.6 What are the issues/problems with the existing system?

The existing system is not user-friendly system.

* Difficult to search record When there is no computerized system there is always a difficulty in searching of records if the records are large in number.
* Difficult to manage booking dates.

**1.7** How are you going to improve on the issues/problems with the existing system?

* **Improvement in control and performance**

The system is developed to cope up with the current issues and problems of library .The system can add user, validate user and is also bug free.

* **Save cost**

After computerized system is implemented less human force will be required to maintain the library thus reducing the overall cost.

* **Save time**

Librarian is able to search record by using few clicks of mouse and few search keywords thus saving his valuable time.

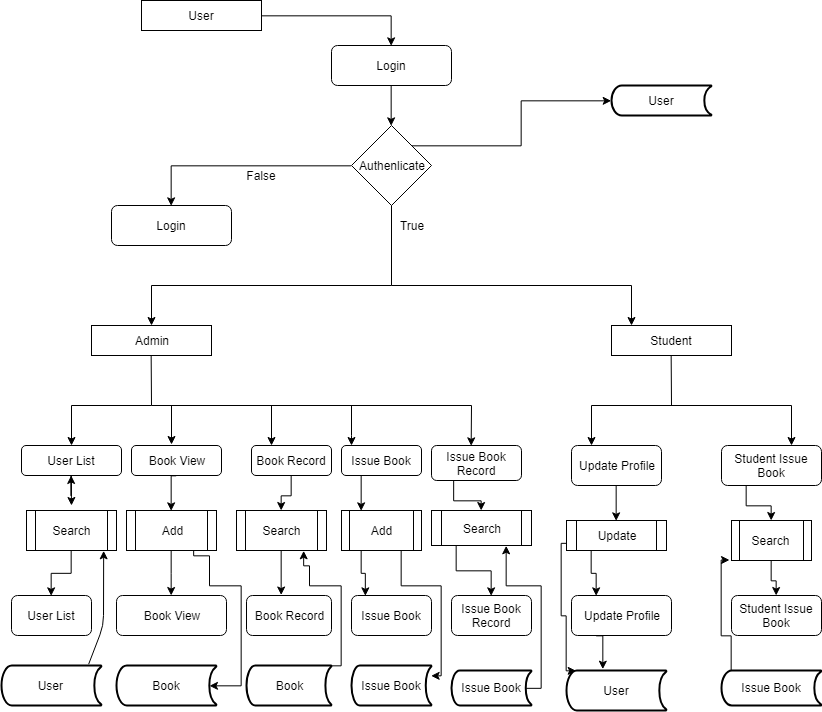
**Chapter-2**

**System Analysis**

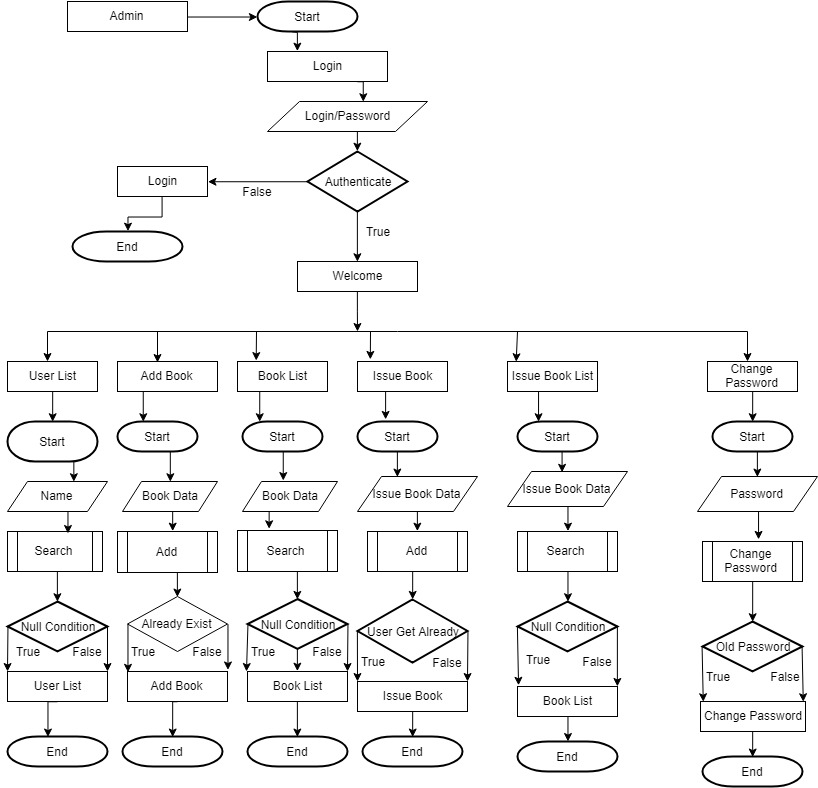
**2.1 E R DIAGRAM:**

**LibraryMgt ER.png**

**Data Flow Diagram(DFD)**

****

**2.3Functional Decomposation:**

****

**Feasibility:**

This project will be developed on computer, so first check whether the technology is technically available or not. Now a day’s computer interaction with any job becomes common for any kind of job or work.

And because of increasing usage of Computer, Computer is also available with a variety of hardware. Vendors can fulfill any type of hardware requirement. The whole project is developed by some special tools or by using languages and databases, which are also available in a variety.

Preliminary investigation of a system examines the feasibility of a system that is useful to an organization. It is the first phase of system development.

The main objective of this phase is to identify the current deficiencies in the user’s environment and to determine which existing problem are going to be solve in proposed system and also which new function needs to be added in proposed system.

An important outcome of such preliminary investigation is to determine whether the system that will meet all needed requirements.

Thus, three tests are carried out on the system namely operation, technical and economical.

Any project is beneficial if and only satisfies the organization requirement. For any new system setup, it only meets to be communicated and work the other supporting system.

The new system meets all existing operations since it provides right information at a right time to the right user. A Leigh man can easily operate with the system.

Technical Feasibility examines whether the technology needed is available and if it is available then it feasible to carry out all project activities.

The technical needs of a system include:

* The facility to produce outputs in a given time.
* Ability to process large number of transaction at a particular speed.
* Giving response to users under certain conditions.

The technology needed for our system is mainly:

* Latest version of browsers.
* Any operating system.

These technologies are available which helps to carry out the system efficiently.

Economical feasibility of a system examines whether the finance is available for implementing the new system and whether the money spent is recoverable the satisfaction.

The cost involves is in designing and developing a good investment for the organization.

Thus, hardware requirements used for proposed system are very standard. Moreover, by making use of proposed system to carry out the work speedily will increase and also saves the valuable time of an organization.

In the proposed system the finance is highly required for the installation of the software’s which can also be recovered by implementing a better system.

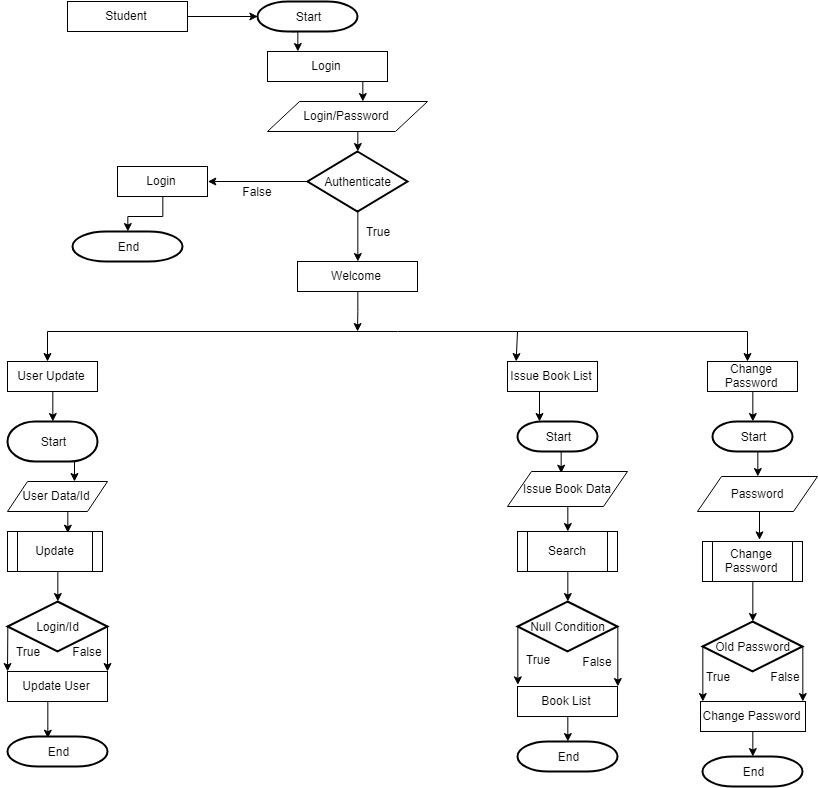


**Chapter-3**

**Design**

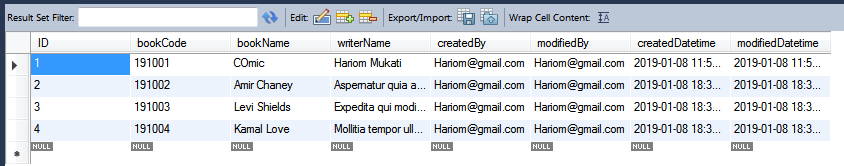
**3.1 Data flow diagram:.**

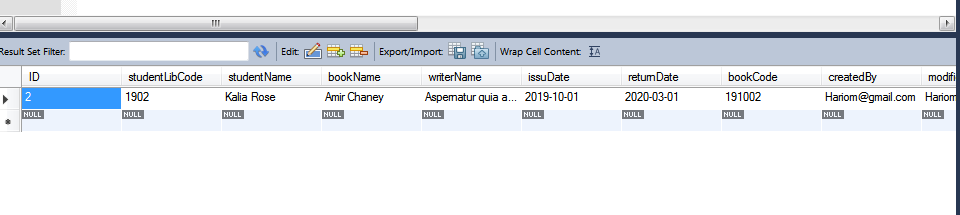
**System Flow Chart :**



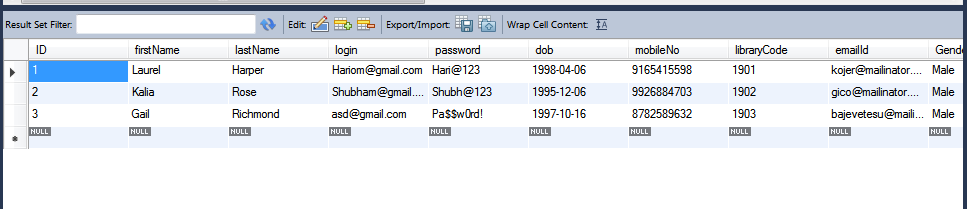
**Data dictionary**

**3.2.1 Book Management:**

****

****

**3.2.4 Users:**

****

**Data validaton:**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been integrated in the system in almost every area where there is a possibility for the user to commit errors. The system will not recognize invalid data.

Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been integrated where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with pop up menus.

**Different Type Of validation :**

* Data type validation;
* Range and constraint validation;
* Code and Cross-reference validation; and
* Structured validation

**Coding**

**DATABASE CONNECTIVITY CODE:**

**BookModel**

package in.co.lib.mgt.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import org.apache.log4j.Logger;

import in.co.lib.mgt.bean.BookBean;

import in.co.lib.mgt.bean.UserBean;

import in.co.lib.mgt.exception.ApplicationException;

import in.co.lib.mgt.exception.DatabaseException;

import in.co.lib.mgt.exception.DuplicateRecordException;

import in.co.lib.mgt.util.JDBCDataSource;

public class BookModel {

private static Logger log = Logger.getLogger(UserModel.class);

public Integer nextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM LI\_BOOK");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public Integer nextBookCode() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(BookCode) FROM LI\_Book");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public long add(BookBean bean) throws ApplicationException, DuplicateRecordException {

Connection conn = null;

int pk = 0;

BookBean existbean = findByBookName(bean.getBookName());

if (existbean != null) {

throw new DuplicateRecordException("Book Is already exists");

}

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO LI\_BOOK VALUES(?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setLong(2, nextBookCode());

pstmt.setString(3, bean.getBookName());

pstmt.setString(4, bean.getWriterName());

pstmt.setString(5, bean.getCreatedBy());

pstmt.setString(6, bean.getModifiedBy());

pstmt.setTimestamp(7, bean.getCreatedDatetime());

pstmt.setTimestamp(8, bean.getModifiedDatetime());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

ex.printStackTrace();

throw new ApplicationException("Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add User");

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk;

}

public BookBean findByBookName(String login) throws ApplicationException {

log.debug("Model findByLogin Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_Book WHERE BookName=?");

BookBean bean = null;

Connection conn = null;

System.out.println("sql" + sql);

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, login);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new BookBean();

bean.setId(rs.getLong(1));

bean.setBookCode(rs.getLong(2));

bean.setBookName(rs.getString(3));

bean.setWriterName(rs.getString(4));

bean.setCreatedBy(rs.getString(5));

bean.setModifiedBy(rs.getString(6));

bean.setCreatedDatetime(rs.getTimestamp(7));

bean.setModifiedDatetime(rs.getTimestamp(8));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by login");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByLogin End");

return bean;

}

public BookBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_Book WHERE ID=?");

BookBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new BookBean();

bean.setId(rs.getLong(1));

bean.setBookCode(rs.getLong(2));

bean.setBookName(rs.getString(3));

bean.setWriterName(rs.getString(4));

bean.setCreatedBy(rs.getString(5));

bean.setModifiedBy(rs.getString(6));

bean.setCreatedDatetime(rs.getTimestamp(7));

bean.setModifiedDatetime(rs.getTimestamp(8));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public BookBean findByBookCode(long code) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_BOOK WHERE BOOKCODE=?");

BookBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, code);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new BookBean();

bean.setId(rs.getLong(1));

bean.setBookCode(rs.getLong(2));

bean.setBookName(rs.getString(3));

bean.setWriterName(rs.getString(4));

bean.setCreatedBy(rs.getString(5));

bean.setModifiedBy(rs.getString(6));

bean.setCreatedDatetime(rs.getTimestamp(7));

bean.setModifiedDatetime(rs.getTimestamp(8));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public void delete(BookBean bean) throws ApplicationException {

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("DELETE FROM LI\_BOOK WHERE ID=?");

pstmt.setLong(1, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in delete User");

} finally {

JDBCDataSource.closeConnection(conn);

}

}

public List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of Role with pagination

\*

\* @return list : List of Role

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List list(int pageNo, int pageSize) throws ApplicationException {

log.debug("Model list Started");

ArrayList list = new ArrayList();

StringBuffer sql = new StringBuffer("select \* from LI\_BOOK");

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" limit " + pageNo + "," + pageSize);

}

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

BookBean bean = new BookBean();

bean.setId(rs.getLong(1));

bean.setBookCode(rs.getLong(2));

bean.setBookName(rs.getString(3));

bean.setWriterName(rs.getString(4));

bean.setCreatedBy(rs.getString(5));

bean.setModifiedBy(rs.getString(6));

bean.setCreatedDatetime(rs.getTimestamp(7));

bean.setModifiedDatetime(rs.getTimestamp(8));

list.add(bean);

}

rs.close();

} catch (Exception e) {

// log.error("Database Exception..", e);

throw new ApplicationException(

"Exception : Exception in getting list of Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

public List search(BookBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

/\*\*

\* Search Role with pagination

\*

\* @return list : List of Roles

\* @param bean

\* : Search Parameters

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\*

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List search(BookBean bean, int pageNo, int pageSize)

throws ApplicationException {

log.debug("Model search Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_BOOK WHERE 1=1");

if (bean != null) {

if (bean.getId() > 0) {

sql.append(" AND id = " + bean.getId());

}

if (bean.getBookCode() > 0) {

sql.append(" AND BookCode = " + bean.getBookCode());

}

if (bean.getBookName() != null && bean.getBookName().length() > 0) {

sql.append(" AND BOOKNAME LIKE '" + bean.getBookName() + "%'");

}

if (bean.getWriterName() != null

&& bean.getWriterName().length() > 0) {

sql.append(" AND WriterName LIKE '" + bean.getWriterName()

+ "%'");

}

}

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" Limit " + pageNo + ", " + pageSize);

// sql.append(" limit " + pageNo + "," + pageSize);

}

ArrayList list = new ArrayList();

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new BookBean();

bean.setId(rs.getLong(1));

bean.setBookCode(rs.getLong(2));

bean.setBookName(rs.getString(3));

bean.setWriterName(rs.getString(4));

bean.setCreatedBy(rs.getString(5));

bean.setModifiedBy(rs.getString(6));

bean.setCreatedDatetime(rs.getTimestamp(7));

bean.setModifiedDatetime(rs.getTimestamp(8));

list.add(bean);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException(

"Exception : Exception in search Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

public void update(BookBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

BookBean beanExist = findByBookName(bean.getBookName());

// Check if updated LoginId already exist

if (beanExist != null && !(beanExist.getId() == bean.getId())) {

throw new DuplicateRecordException("BookBean is already exist");

}

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE LI\_BOOK SET BOOKCODE=?,BOOKNAME=?,WRITERNAME=?,"

+ "CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");

pstmt.setLong(1, bean.getBookCode());

pstmt.setString(2, bean.getBookName());

pstmt.setString(3, bean.getWriterName());

pstmt.setString(4, bean.getCreatedBy());

pstmt.setString(5, bean.getModifiedBy());

pstmt.setTimestamp(6, bean.getCreatedDatetime());

pstmt.setTimestamp(7, bean.getModifiedDatetime());

pstmt.setLong(8,bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

e.printStackTrace();

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception in updating User ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

}

**User Model**

package in.com.online.exam.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.HashMap;

import org.apache.log4j.Logger;

import in.com.online.exam.bean.UserBean;

import in.com.online.exam.exeption.ApplicationException;

import in.com.online.exam.exeption.DatabaseException;

import in.com.online.exam.exeption.DuplicateRecordException;

import in.com.online.exam.util.EmailBuilder;

import in.com.online.exam.util.EmailMessage;

import in.com.online.exam.util.EmailUtility;

import in.com.online.exam.util.JDBCDataSource;

public class UserModel {

private static Logger log = Logger.getLogger(UserModel.class);

public Integer nextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM EX\_USER");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public long add(UserBean bean) throws ApplicationException, DuplicateRecordException {

Connection conn = null;

int pk = 0;

UserBean existbean = findByLogin(bean.getLogin());

if (existbean != null) {

throw new DuplicateRecordException("Login Id already exists");

}

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO EX\_USER VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setString(2, bean.getFName());

pstmt.setString(3, bean.getLName());

pstmt.setString(4, bean.getLogin());

pstmt.setString(5, bean.getPassword());

pstmt.setString(6, bean.getMobileNo());

pstmt.setDate(7, new java.sql.Date(bean.getDob().getTime()));

pstmt.setString(8, bean.getGender());

pstmt.setString(9,bean.getAddress());

pstmt.setString(10, bean.getCreatedBy());

pstmt.setString(11, bean.getModifiedBy());

pstmt.setTimestamp(12, bean.getCreatedDatetime());

pstmt.setTimestamp(13, bean.getModifiedDatetime());

pstmt.setString(14,bean.getRoleName());

pstmt.setLong(15, bean.getRole\_Id());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

ex.printStackTrace();

throw new ApplicationException("Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add User");

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk;

}

public UserBean findByLogin(String login) throws ApplicationException {

log.debug("Model findByLogin Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM EX\_USER WHERE LOGIN=?");

UserBean bean = null;

Connection conn = null;

System.out.println("sql" + sql);

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, login);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFName(rs.getString(2));

bean.setLName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setMobileNo(rs.getString(6));

bean.setDob(rs.getDate(7));

bean.setAddress(rs.getString(8));

bean.setGender(rs.getString(9));

bean.setCreatedBy(rs.getString(10));

bean.setModifiedBy(rs.getString(11));

bean.setCreatedDatetime(rs.getTimestamp(12));

bean.setModifiedDatetime(rs.getTimestamp(13));

bean.setRoleName(rs.getString(14));

bean.setRole\_Id(rs.getLong(15));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by login");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByLogin End");

return bean;

}

public UserBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM EX\_USER WHERE ID=?");

UserBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFName(rs.getString(2));

bean.setLName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setMobileNo(rs.getString(6));

bean.setDob(rs.getDate(7));

bean.setAddress(rs.getString(8));

bean.setGender(rs.getString(9));

bean.setCreatedBy(rs.getString(10));

bean.setModifiedBy(rs.getString(11));

bean.setCreatedDatetime(rs.getTimestamp(12));

bean.setModifiedDatetime(rs.getTimestamp(13));

bean.setRoleName(rs.getString(14));

bean.setRole\_Id(rs.getLong(15));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public UserBean authenticate(String login, String password) throws ApplicationException {

log.debug("Model authenticate Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM EX\_USER WHERE LOGIN = ? AND PASSWORD = ?");

UserBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, login);

pstmt.setString(2, password);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFName(rs.getString(2));

bean.setLName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setMobileNo(rs.getString(6));

bean.setDob(rs.getDate(7));

bean.setAddress(rs.getString(8));

bean.setGender(rs.getString(9));

bean.setCreatedBy(rs.getString(10));

bean.setModifiedBy(rs.getString(11));

bean.setCreatedDatetime(rs.getTimestamp(12));

bean.setModifiedDatetime(rs.getTimestamp(13));

bean.setRoleName(rs.getString(14));

bean.setRole\_Id(rs.getLong(15));

System.out.println("Usermodel here");

}

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in get roles");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model authenticate End");

return bean;

}

public long registerUser(UserBean bean)

throws ApplicationException, DuplicateRecordException {

log.debug("Model add Started");

long pk = add(bean);

HashMap<String, String> map = new HashMap<String, String>();

map.put("login", bean.getLogin());

map.put("password", bean.getPassword());

String message = EmailBuilder.getUserRegistrationMessage(map);

EmailMessage msg = new EmailMessage();

msg.setTo(bean.getLogin());

msg.setSubject("Registration is successful for ORS Project SunilOS");

msg.setMessage(message);

msg.setMessageType(EmailMessage.HTML\_MSG);

try {

EmailUtility.sendMail(msg);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return pk;

}

public void update(UserBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

UserBean beanExist = findByLogin(bean.getLogin());

// Check if updated LoginId already exist

if (beanExist != null && !(beanExist.getId() == bean.getId())) {

throw new DuplicateRecordException("LoginId is already exist");

}

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE ST\_USER SET FNAME=?,LNAME=?,LOGIN=?,PASSWORD=?,MOBILE\_NO=?,DOB=?,ADDEESS=?,GENDER=?,"

+ "CREATED\_BY=?,MODIFIED\_BY=?,CREATED\_DATETIME=?,MODIFIED\_DATETIME=?,ROLE\_NAME=?,ROLE\_ID=? WHERE ID=?");

pstmt.setString(1, bean.getFName());

pstmt.setString(2, bean.getLName());

pstmt.setString(3, bean.getLogin());

pstmt.setString(4, bean.getPassword());

pstmt.setString(5, bean.getMobileNo());

pstmt.setDate(6, new java.sql.Date(bean.getDob().getTime()) );

pstmt.setString(7, bean.getAddress());

pstmt.setString(8, bean.getGender());

pstmt.setString(9, bean.getCreatedBy());

pstmt.setString(10, bean.getModifiedBy());

pstmt.setTimestamp(11, bean.getCreatedDatetime());

pstmt.setTimestamp(12, bean.getModifiedDatetime());

pstmt.setLong(13, bean.getId());

pstmt.setString(14,bean.getRoleName());

pstmt.setLong(15,bean.getRole\_Id());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception in updating User ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

}

================Issue Book Model==============================================

package in.co.lib.mgt.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import org.apache.log4j.Logger;

import in.co.lib.mgt.bean.BookBean;

import in.co.lib.mgt.bean.IssuBookBean;

import in.co.lib.mgt.bean.UserBean;

import in.co.lib.mgt.exception.ApplicationException;

import in.co.lib.mgt.exception.DatabaseException;

import in.co.lib.mgt.exception.DuplicateRecordException;

import in.co.lib.mgt.util.JDBCDataSource;

public class IssuBookModel {

private static Logger log = Logger.getLogger(IssuBookModel.class);

public Integer nextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM LI\_ISSUBOOK");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public long add(IssuBookBean bean) throws ApplicationException, DuplicateRecordException {

Connection conn = null;

int pk = 0;

UserModel uModel=new UserModel();

UserBean uBean=uModel.findByLibCode(bean.getStudentLibCode());

bean.setStudenName(uBean.getFirstName()+" "+uBean.getLastName());

BookModel bModel=new BookModel();

BookBean bBean=bModel.findByBookCode(bean.getBookCode());

bean.setBookName(bBean.getBookName());

bean.setWriterName(bBean.getWriterName());

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO LI\_IssuBook VALUES(?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setLong(2,bean.getStudentLibCode());

pstmt.setString(3,bean.getStudenName());

pstmt.setString(4, bean.getBookName());

pstmt.setString(5, bean.getWriterName());

pstmt.setDate(6,new java.sql.Date(bean.getIssuDate().getTime()));

pstmt.setDate(7,new java.sql.Date(bean.getReturnDate().getTime()));

pstmt.setLong(8,bean.getBookCode());

pstmt.setString(9, bean.getCreatedBy());

pstmt.setString(10, bean.getModifiedBy());

pstmt.setTimestamp(11, bean.getCreatedDatetime());

pstmt.setTimestamp(12, bean.getModifiedDatetime());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

ex.printStackTrace();

throw new ApplicationException("Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add User");

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk;

}

public IssuBookBean findByBookName(String login) throws ApplicationException {

log.debug("Model findByLogin Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_ISSUBook WHERE BookName=?");

IssuBookBean bean = null;

Connection conn = null;

System.out.println("sql" + sql);

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, login);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new IssuBookBean();

bean.setId(rs.getLong(1));

bean.setStudentLibCode(rs.getLong(2));

bean.setStudenName(rs.getString(3));

bean.setBookName(rs.getString(4));

bean.setWriterName(rs.getString(5));

bean.setIssuDate(rs.getDate(6));

bean.setReturnDate(rs.getDate(7));

bean.setBookCode(rs.getLong(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by login");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByLogin End");

return bean;

}

public IssuBookBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_ISSUBOOK WHERE ID=?");

IssuBookBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new IssuBookBean();

bean.setId(rs.getLong(1));

bean.setStudentLibCode(rs.getLong(2));

bean.setStudenName(rs.getString(3));

bean.setBookName(rs.getString(4));

bean.setWriterName(rs.getString(5));

bean.setIssuDate(rs.getDate(6));

bean.setReturnDate(rs.getDate(7));

bean.setBookCode(rs.getLong(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public IssuBookBean findByBookCode(long code) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_ISSUBOOK WHERE bookCode=?");

IssuBookBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, code);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new IssuBookBean();

bean.setId(rs.getLong(1));

bean.setStudentLibCode(rs.getLong(2));

bean.setStudenName(rs.getString(3));

bean.setBookName(rs.getString(4));

bean.setWriterName(rs.getString(5));

bean.setIssuDate(rs.getDate(6));

bean.setReturnDate(rs.getDate(7));

bean.setBookCode(rs.getLong(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public void delete(IssuBookBean bean) throws ApplicationException {

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("DELETE FROM LI\_IssuBook WHERE ID=?");

pstmt.setLong(1, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in delete User");

} finally {

JDBCDataSource.closeConnection(conn);

}

}

public List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of Role with pagination

\*

\* @return list : List of Role

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List list(int pageNo, int pageSize) throws ApplicationException {

log.debug("Model list Started");

ArrayList list = new ArrayList();

StringBuffer sql = new StringBuffer("select \* from LI\_ISSUBOOK");

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" limit " + pageNo + "," + pageSize);

}

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

IssuBookBean bean = new IssuBookBean();

bean.setId(rs.getLong(1));

bean.setStudentLibCode(rs.getLong(2));

bean.setStudenName(rs.getString(3));

bean.setBookName(rs.getString(4));

bean.setWriterName(rs.getString(5));

bean.setIssuDate(rs.getDate(6));

bean.setReturnDate(rs.getDate(7));

bean.setBookCode(rs.getLong(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

list.add(bean);

}

rs.close();

} catch (Exception e) {

// log.error("Database Exception..", e);

throw new ApplicationException(

"Exception : Exception in getting list of Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

public List search(IssuBookBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

/\*\*

\* Search Role with pagination

\*

\* @return list : List of Roles

\* @param bean

\* : Search Parameters

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\*

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List search(IssuBookBean bean, int pageNo, int pageSize)

throws ApplicationException {

log.debug("Model search Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_IssuBook WHERE 1=1");

if (bean != null) {

if (bean.getId() > 0) {

sql.append(" AND id = " + bean.getId());

}

if (bean.getBookCode() > 0) {

sql.append(" AND BookCode = " + bean.getBookCode());

}

if (bean.getStudentLibCode() > 0) {

sql.append(" AND StudentLibCode = " + bean.getStudentLibCode());

}

if (bean.getBookName() != null && bean.getBookName().length() > 0) {

sql.append(" AND BOOKNAME LIKE '" + bean.getBookName() + "%'");

}

if (bean.getStudenName() != null && bean.getStudenName().length() > 0) {

sql.append(" AND StudentName LIKE '" + bean.getStudenName() + "%'");

}

if (bean.getIssuDate() != null) {

sql.append(" AND issudate LIKE '" + new java.sql.Date(bean.getIssuDate().getTime()) + "%'");

}

if (bean.getWriterName() != null

&& bean.getWriterName().length() > 0) {

sql.append(" AND WriterName LIKE '" + bean.getWriterName()

+ "%'");

}

}

sql.append(" Order by ID Desc");

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" Limit " + pageNo + ", " + pageSize);

// sql.append(" limit " + pageNo + "," + pageSize);

}

ArrayList list = new ArrayList();

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

System.out.println(sql.toString());

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new IssuBookBean();

bean.setId(rs.getLong(1));

bean.setStudentLibCode(rs.getLong(2));

bean.setStudenName(rs.getString(3));

bean.setBookName(rs.getString(4));

bean.setWriterName(rs.getString(5));

bean.setIssuDate(rs.getDate(6));

bean.setReturnDate(rs.getDate(7));

bean.setBookCode(rs.getLong(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

list.add(bean);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException(

"Exception : Exception in search Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

public static void main(String[] args) throws ApplicationException {

IssuBookModel model=new IssuBookModel();

IssuBookBean bean=null;

model.search(bean);

}

} }

===========================================User Model==========================================

package in.co.lib.mgt.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import org.apache.log4j.Logger;

import in.co.lib.mgt.bean.UserBean;

import in.co.lib.mgt.exception.ApplicationException;

import in.co.lib.mgt.exception.DatabaseException;

import in.co.lib.mgt.exception.DuplicateRecordException;

import in.co.lib.mgt.exception.RecordNotFoundException;

import in.co.lib.mgt.util.EmailBuilder;

import in.co.lib.mgt.util.EmailMessage;

import in.co.lib.mgt.util.EmailUtility;

import in.co.lib.mgt.util.JDBCDataSource;

public class UserModel {

private static Logger log = Logger.getLogger(UserModel.class);

public Integer nextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM LI\_USER");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public Integer nextLibCode() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(LibraryCode) FROM LI\_USER");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public long add(UserBean bean) throws ApplicationException, DuplicateRecordException {

Connection conn = null;

int pk = 0;

UserBean existbean = findByLogin(bean.getLogin());

if (existbean != null) {

throw new DuplicateRecordException("Login Id already exists");

}

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO LI\_USER VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setString(2, bean.getFirstName());

pstmt.setString(3, bean.getLastName());

pstmt.setString(4, bean.getLogin());

pstmt.setString(5, bean.getPassword());

pstmt.setDate(6, new java.sql.Date(bean.getDob().getTime()));

pstmt.setString(7, bean.getMobileNo());

pstmt.setLong(8,nextLibCode());

pstmt.setString(9,bean.getEmailId());

pstmt.setString(10, bean.getGender());

pstmt.setLong(11, bean.getRoleId());

pstmt.setString(12, bean.getCreatedBy());

pstmt.setString(13, bean.getModifiedBy());

pstmt.setTimestamp(14, bean.getCreatedDatetime());

pstmt.setTimestamp(15, bean.getModifiedDatetime());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

ex.printStackTrace();

throw new ApplicationException("Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add User");

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk;

}

public UserBean findByLogin(String login) throws ApplicationException {

log.debug("Model findByLogin Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_USER WHERE Login=?");

UserBean bean = null;

Connection conn = null;

System.out.println("sql" + sql);

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, login);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFirstName(rs.getString(2));

bean.setLastName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setDob(rs.getDate(6));

bean.setMobileNo(rs.getString(7));

bean.setLibraryCode(rs.getLong(8));

bean.setEmailId(rs.getString(9));

bean.setGender(rs.getString(10));

bean.setRoleId(rs.getLong(11));

bean.setCreatedBy(rs.getString(12));

bean.setModifiedBy(rs.getString(13));

bean.setCreatedDatetime(rs.getTimestamp(14));

bean.setModifiedDatetime(rs.getTimestamp(15));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by login");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByLogin End");

return bean;

}

public UserBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_USER WHERE ID=?");

UserBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFirstName(rs.getString(2));

bean.setLastName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setDob(rs.getDate(6));

bean.setMobileNo(rs.getString(7));

bean.setLibraryCode(rs.getLong(8));

bean.setEmailId(rs.getString(9));

bean.setGender(rs.getString(10));

bean.setRoleId(rs.getLong(11));

bean.setCreatedBy(rs.getString(12));

bean.setModifiedBy(rs.getString(13));

bean.setCreatedDatetime(rs.getTimestamp(14));

bean.setModifiedDatetime(rs.getTimestamp(15));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public UserBean findByLibCode(long code) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_USER WHERE LibraryCode=?");

UserBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, code);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFirstName(rs.getString(2));

bean.setLastName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setDob(rs.getDate(6));

bean.setMobileNo(rs.getString(7));

bean.setLibraryCode(rs.getLong(8));

bean.setEmailId(rs.getString(9));

bean.setGender(rs.getString(10));

bean.setRoleId(rs.getLong(11));

bean.setCreatedBy(rs.getString(12));

bean.setModifiedBy(rs.getString(13));

bean.setCreatedDatetime(rs.getTimestamp(14));

bean.setModifiedDatetime(rs.getTimestamp(15));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public UserBean authenticate(String login, String password) throws ApplicationException {

log.debug("Model authenticate Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_USER WHERE LOGIN = ? AND PASSWORD = ?");

UserBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, login);

pstmt.setString(2, password);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFirstName(rs.getString(2));

bean.setLastName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setDob(rs.getDate(6));

bean.setMobileNo(rs.getString(7));

bean.setLibraryCode(rs.getLong(8));

bean.setEmailId(rs.getString(9));

bean.setGender(rs.getString(10));

bean.setRoleId(rs.getLong(11));

bean.setCreatedBy(rs.getString(12));

bean.setModifiedBy(rs.getString(13));

bean.setCreatedDatetime(rs.getTimestamp(14));

bean.setModifiedDatetime(rs.getTimestamp(15));

System.out.println("Usermodel here");

}

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in get roles");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model authenticate End");

return bean;

}

public long registerUser(UserBean bean)

throws ApplicationException, DuplicateRecordException {

log.debug("Model add Started");

long pk = add(bean);

return pk;

}

public void update(UserBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

UserBean beanExist = findByLogin(bean.getLogin());

// Check if updated LoginId already exist

if (beanExist != null && !(beanExist.getId() == bean.getId())) {

throw new DuplicateRecordException("LoginId is already exist");

}

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE LI\_USER SET FirstNAME=?,LastNAME=?,LOGIN=?,PASSWORD=?,DOB=?,MOBILENO=?,librarycode=?,emailId=?,GENDER=?,ROLEID=?"

+ "CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");

pstmt.setString(1, bean.getFirstName());

pstmt.setString(2, bean.getLastName());

pstmt.setString(3, bean.getLogin());

pstmt.setString(4, bean.getPassword());

pstmt.setDate(5, new java.sql.Date(bean.getDob().getTime()));

pstmt.setString(6, bean.getMobileNo());

pstmt.setLong(7,bean.getLibraryCode());

pstmt.setString(8,bean.getEmailId());

pstmt.setString(9, bean.getGender());

pstmt.setLong(10, bean.getRoleId());

pstmt.setString(11, bean.getCreatedBy());

pstmt.setString(12, bean.getModifiedBy());

pstmt.setTimestamp(13, bean.getCreatedDatetime());

pstmt.setTimestamp(14, bean.getModifiedDatetime());

pstmt.setLong(15,bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

e.printStackTrace();

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception in updating User ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

public void delete(UserBean bean) throws ApplicationException {

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("DELETE FROM LI\_USER WHERE ID=?");

pstmt.setLong(1, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in delete User");

} finally {

JDBCDataSource.closeConnection(conn);

}

}

public List search(UserBean bean, int pageNo, int pageSize) throws ApplicationException {

log.debug("Model search Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM LI\_USER WHERE 1=1");

if (bean != null) {

if (bean.getId() > 0) {

sql.append(" AND id = " + bean.getId());

}

if (bean.getLibraryCode() > 0) {

sql.append(" AND librarycode = " + bean.getLibraryCode());

}

if (bean.getFirstName() != null && bean.getFirstName().length() > 0) {

sql.append(" AND FirstNAME like '" + bean.getFirstName() + "%'");

}

if (bean.getLogin() != null && bean.getLogin().length() > 0) {

sql.append(" AND LOGIN like '" + bean.getLogin() + "%'");

}

if (bean.getEmailId() != null && bean.getEmailId().length() > 0) {

sql.append(" AND EmailId like '" + bean.getEmailId() + "%'");

}

}

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" Limit " + pageNo + ", " + pageSize);

// sql.append(" limit " + pageNo + "," + pageSize);

}

System.out.println("user model search :"+sql);

ArrayList list = new ArrayList();

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new UserBean();

bean.setId(rs.getLong(1));

bean.setFirstName(rs.getString(2));

bean.setLastName(rs.getString(3));

bean.setLogin(rs.getString(4));

bean.setPassword(rs.getString(5));

bean.setDob(rs.getDate(6));

bean.setMobileNo(rs.getString(7));

bean.setLibraryCode(rs.getLong(8));

bean.setEmailId(rs.getString(9));

bean.setGender(rs.getString(10));

bean.setRoleId(rs.getLong(11));

bean.setCreatedBy(rs.getString(12));

bean.setModifiedBy(rs.getString(13));

bean.setCreatedDatetime(rs.getTimestamp(14));

bean.setModifiedDatetime(rs.getTimestamp(15));

list.add(bean);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in search user");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

public boolean changePassword(Long id, String oldPassword, String newPassword)

throws RecordNotFoundException, ApplicationException {

log.debug("model changePassword Started");

boolean flag = false;

UserBean beanExist = null;

beanExist = findByPK(id);

if (beanExist != null && beanExist.getPassword().equals(oldPassword)) {

beanExist.setPassword(newPassword);

try {

update(beanExist);

} catch (DuplicateRecordException e) {

log.error(e);

throw new ApplicationException("LoginId is already exist");

}

flag = true;

} else {

throw new RecordNotFoundException("Old password is Invalid");

}

HashMap<String, String> map = new HashMap<String, String>();

map.put("login", beanExist.getLogin());

map.put("password", beanExist.getPassword());

map.put("firstName", beanExist.getFirstName());

map.put("lastName", beanExist.getFirstName());

String message = EmailBuilder.getChangePasswordMessage(map);

EmailMessage msg = new EmailMessage();

msg.setTo(beanExist.getEmailId());

msg.setSubject("SUNARYS ORS Password has been changed Successfully.");

msg.setMessage(message);

msg.setMessageType(EmailMessage.HTML\_MSG);

try {

EmailUtility.sendMail(msg);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

log.debug("Model changePassword End");

return flag;

}

}

======================Mapping==================

**package in.co.lib.mgt.controller;**

**public interface LIBView {**

**public String APP\_CONTEXT = "/LibraryMgt";**

**public String LAYOUT\_VIEW = "/BaseLayout.jsp";**

**public String PAGE\_FOLDER = "/jsp";**

**public String JAVA\_DOC\_VIEW = APP\_CONTEXT + "/doc/index.html";**

**public String ERROR\_VIEW = PAGE\_FOLDER + "/Error.jsp";**

**public String USER\_VIEW = PAGE\_FOLDER + "/UserView.jsp";**

**public String USER\_LIST\_VIEW = PAGE\_FOLDER + "/UserListView.jsp";**

**public String USER\_REGISTRATION\_VIEW = PAGE\_FOLDER + "/UserRegistrationView.jsp";**

**public String BOOK\_VIEW = PAGE\_FOLDER + "/BookView.jsp";**

**public String BOOK\_LIST\_VIEW = PAGE\_FOLDER + "/BookListView.jsp";**

**public String MY\_PROFILE\_VIEW = PAGE\_FOLDER + "/MyProfileView.jsp";**

**public String ISSU\_BOOK\_VIEW = PAGE\_FOLDER + "/IssuBookView.jsp";**

**public String ISSU\_BOOK\_LIST\_VIEW = PAGE\_FOLDER + "/IssuBookListView.jsp";**

**public String STUDENT\_DETAIL\_VIEW = PAGE\_FOLDER + "/StudentDetail.jsp";**

**public String LOGIN\_VIEW = PAGE\_FOLDER + "/LoginView.jsp";**

**public String WELCOME\_VIEW ="/WelcomeView.jsp";**

**public String CHANGE\_PASSWORD\_VIEW = PAGE\_FOLDER + "/ChangePasswordView.jsp";**

**public String FORGET\_PASSWORD\_VIEW = PAGE\_FOLDER + "/ForgetPasswordView.jsp";**

**public String ERROR\_CTL = "/ctl/ErrorCtl";**

**public String USER\_CTL = APP\_CONTEXT + "/ctl/UserCtl";**

**public String USER\_LIST\_CTL = APP\_CONTEXT + "/ctl/UserListCtl";**

**public String BOOK\_CTL = APP\_CONTEXT + "/ctl/BookCtl";**

**public String BOOK\_LIST\_CTL = APP\_CONTEXT + "/ctl/BookListCtl";**

**public String ISSU\_BOOK\_CTL = APP\_CONTEXT + "/ctl/IssuBookCtl";**

**public String ISSU\_BOOK\_LIST\_CTL = APP\_CONTEXT + "/ctl/IssuBookListCtl";**

**public String USER\_REGISTRATION\_CTL = APP\_CONTEXT + "/UserRegistrationCtl";**

**public String LOGIN\_CTL = APP\_CONTEXT + "/LoginCtl";**

**public String WELCOME\_CTL = APP\_CONTEXT + "/WelcomeCtl";**

**public String LOGOUT\_CTL = APP\_CONTEXT + "/LoginCtl";**

**public String GET\_MARKSHEET\_CTL = APP\_CONTEXT + "/ctl/GetMarksheetCtl";**

**public String CHANGE\_PASSWORD\_CTL = APP\_CONTEXT + "/ctl/ChangePasswordCtl";**

**public String MY\_PROFILE\_CTL = APP\_CONTEXT + "/ctl/MyProfileCtl";**

**public String FORGET\_PASSWORD\_CTL = APP\_CONTEXT + "/ForgetPasswordCtl";**

**}**

=========================================== Controller===================

**package in.co.lib.mgt.controller;**

**import java.io.IOException;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import org.apache.log4j.Logger;**

**import in.co.lib.mgt.bean.BaseBean;**

**import in.co.lib.mgt.bean.UserBean;**

**import in.co.lib.mgt.util.DataUtility;**

**import in.co.lib.mgt.util.DataValidator;**

**import in.co.lib.mgt.util.ServletUtility;**

**/\*\***

**\* Servlet implementation class BaseCtl**

**\*/**

**/\*\***

**\* Base controller class of project. It contain (1) Generic operations (2)**

**\* Generic constants (3) Generic work flow**

**\***

**\* @author Navigable Set**

**\* @version 1.0**

**\* @Copyright (c) Navigable Set**

**\*/**

**@WebServlet("/BaseCtl")**

**public abstract class BaseCtl extends HttpServlet**

**{**

**private static final Logger log=Logger.getLogger(BaseCtl.class);**

**/\*\***

**\* Generic message key constant**

**\*/**

**public static final String OP\_SAVE = "Save";**

**public static final String OP\_CANCEL = "Cancel";**

**public static final String OP\_DELETE = "Delete";**

**public static final String OP\_LIST = "List";**

**public static final String OP\_SEARCH = "Search";**

**public static final String OP\_VIEW = "View";**

**public static final String OP\_NEXT = "Next";**

**public static final String OP\_PREVIOUS = "Previous";**

**public static final String OP\_NEW = "New";**

**public static final String OP\_GO = "Go";**

**public static final String OP\_BACK = "Back";**

**public static final String OP\_LOG\_OUT = "Logout";**

**public static final String OP\_RESET = "Reset";**

**public static final String OP\_SUBMIT = "Submit";**

**/\*\***

**\* Success message key constant**

**\*/**

**public static final String MSG\_SUCCESS = "success";**

**/\*\***

**\* Error message key constant**

**\*/**

**public static final String MSG\_ERROR = "error";**

**public BaseCtl() {**

**// TODO Auto-generated constructor stub**

**}**

**/\*\***

**\* Validate input Data Entered By User**

**\***

**\* @param request**

**\* @return**

**\*/**

**protected boolean validate(HttpServletRequest request) {**

**return true;**

**}**

**/\*\***

**\* Loads list and other data required to display at HTML form**

**\***

**\* @param request**

**\*/**

**protected void preload(HttpServletRequest request) {**

**}**

**/\*\***

**\* Populates bean object from request parameters**

**\***

**\* @param request**

**\* @return**

**\*/**

**protected BaseBean populateBean(HttpServletRequest request) {**

**return null;**

**}**

**/\*\***

**\* Populates Generic attributes in DTO**

**\***

**\* @param dto**

**\* @param request**

**\* @return**

**\*/**

**protected BaseBean populateDTO(BaseBean dto, HttpServletRequest request) {**

**log.debug("BaseCtl populate DTO method start");**

**String createdBy = request.getParameter("createdBy");**

**String modifiedBy = null;**

**UserBean userbean = (UserBean) request.getSession().getAttribute("user");**

**if (userbean == null) {**

**// If record is created without login**

**createdBy = "root";**

**modifiedBy = "root";**

**} else {**

**modifiedBy = userbean.getLogin();**

**// If record is created first time**

**if ("null".equalsIgnoreCase(createdBy)|| DataValidator.isNull(createdBy)) {**

**createdBy = modifiedBy;**

**}**

**}**

**dto.setCreatedBy(createdBy);**

**dto.setModifiedBy(modifiedBy);**

**long cdt = DataUtility.getLong(request.getParameter("createdDatetime"));**

**if (cdt > 0) {**

**dto.setCreatedDatetime(DataUtility.getTimestamp(cdt));**

**} else {**

**dto.setCreatedDatetime(DataUtility.getCurrentTimestamp());**

**}**

**dto.setModifiedDatetime(DataUtility.getCurrentTimestamp());**

**log.debug("BaseCtl populate DTO method end");**

**return dto;**

**}**

**@Override**

**protected void service(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException {**

**log.debug("BaseCtl service method start");**

**// Load the preloaded data required to display at HTML form**

**preload(request);**

**String op = DataUtility.getString(request.getParameter("operation"));**

**// Check if operation is not DELETE, VIEW, CANCEL, and NULL then**

**// perform input data validation**

**System.out.println("operation ="+op);**

**if (DataValidator.isNotNull(op) && !OP\_CANCEL.equalsIgnoreCase(op)&& !OP\_VIEW.equalsIgnoreCase(op)&& !OP\_DELETE.equalsIgnoreCase(op)&&!OP\_RESET.equalsIgnoreCase(op)) {**

**// Check validation, If fail then send back to page with error**

**// messages**

**if (!validate(request)) {**

**BaseBean bean = (BaseBean) populateBean(request);**

**ServletUtility.setBean(bean, request);**

**ServletUtility.forward(getView(), request, response);**

**return;**

**}**

**}**

**log.debug("BaseCtl service method end");**

**super.service(request, response);**

**}**

**/\*\***

**\* Returns the VIEW page of this Controller**

**\***

**\* @return**

**\*/**

**protected abstract String getView();**

**Implementation and Testing :**

**Black-Box Testing**:

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

This can be following way:

* Input interfacing
* Processing
* Output interfacing



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors

**White-Box Testing:**

White Box Testing ,also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

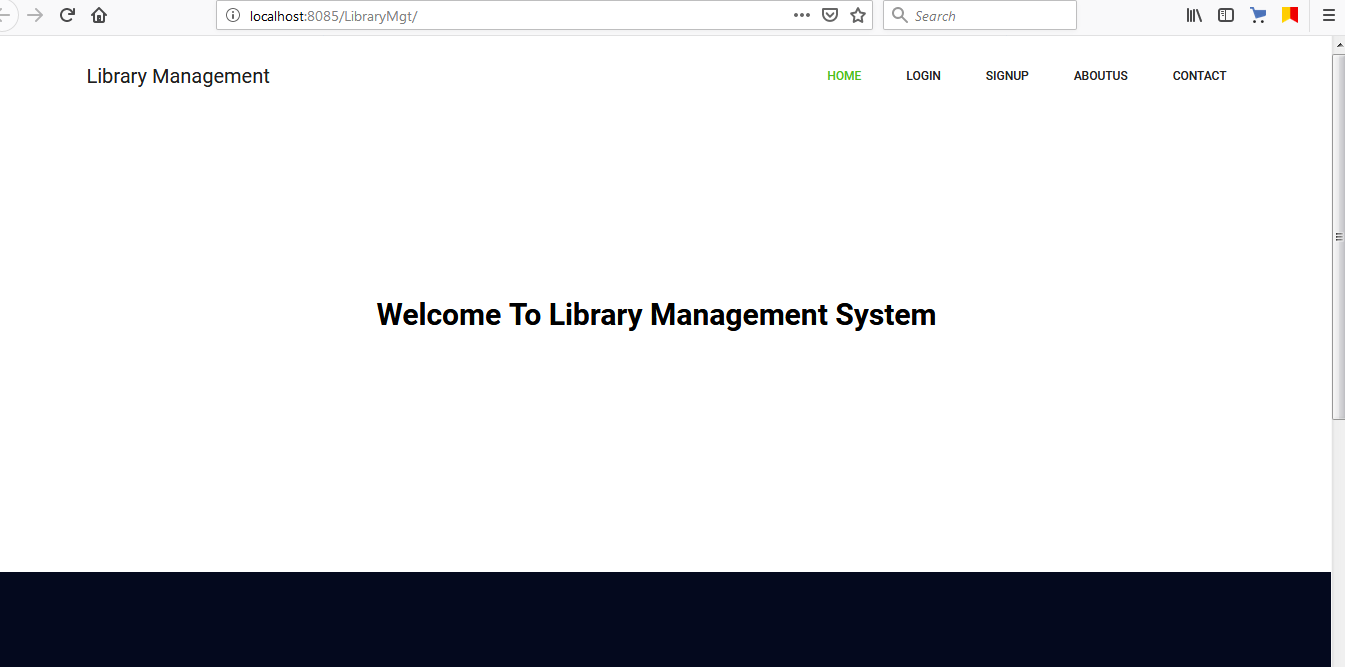
The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential.

White box testing is testing beyond the user interface and into the nitty-gritty of a system.

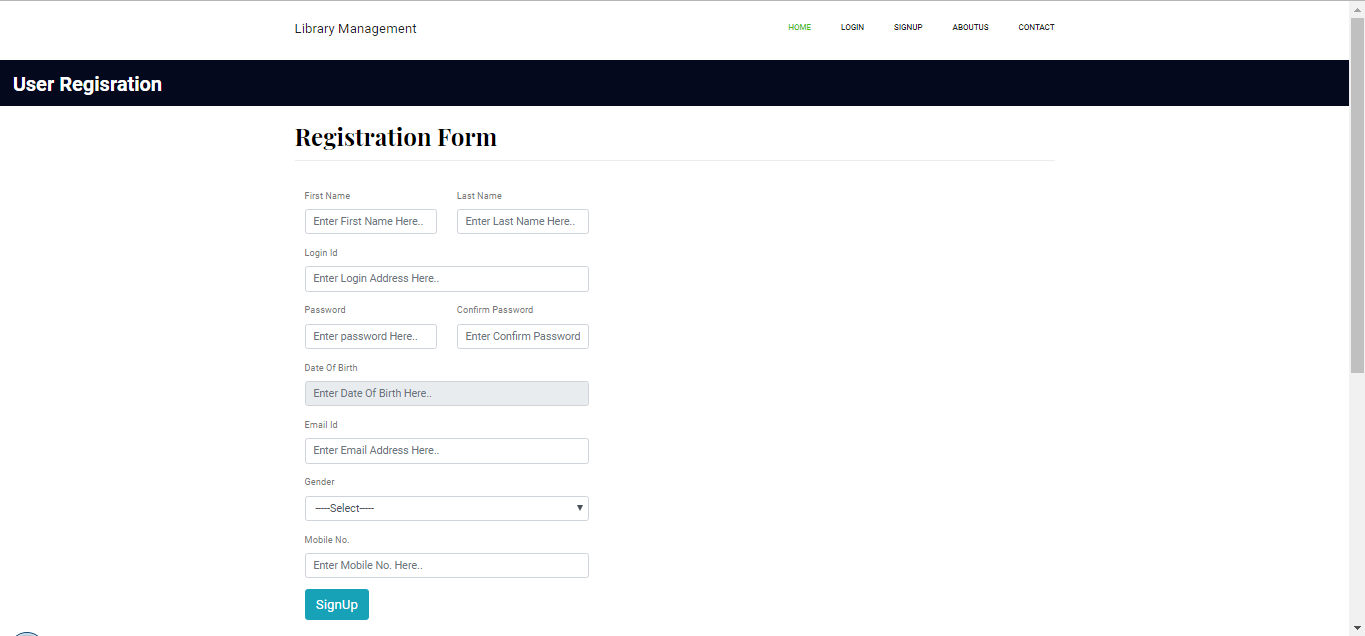
This method is named so because the software program, in the eyes of the tester, is like a white/ transparent box; inside which one clearly sees.

**Screen Snapshot**

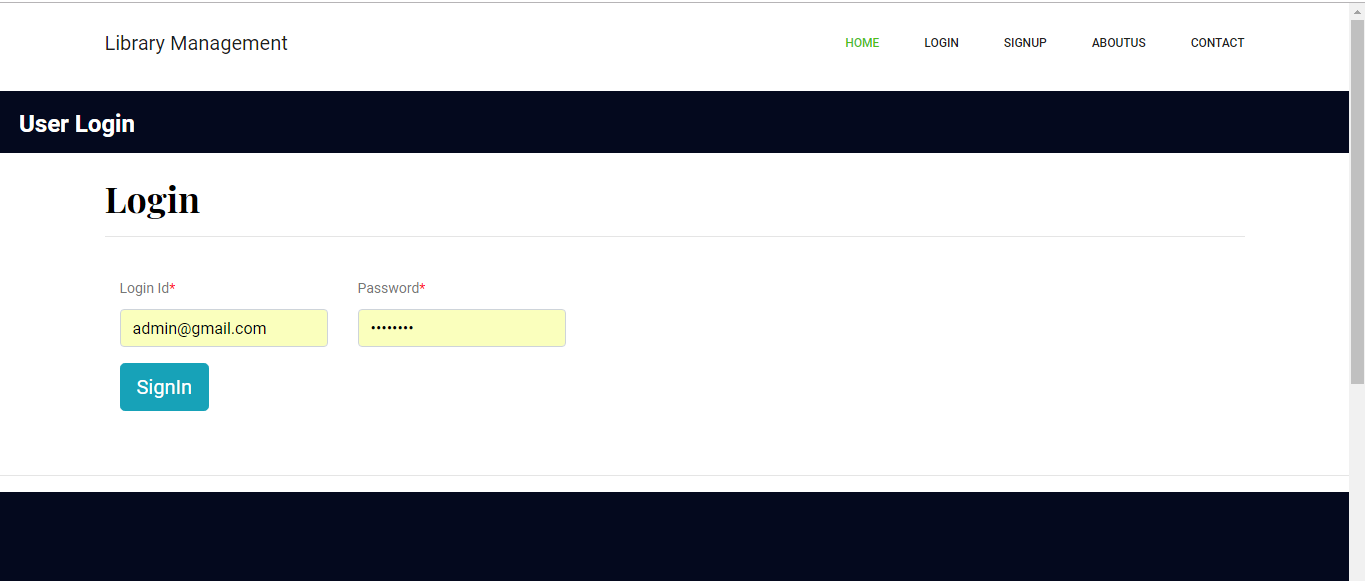
**Home Page**

****

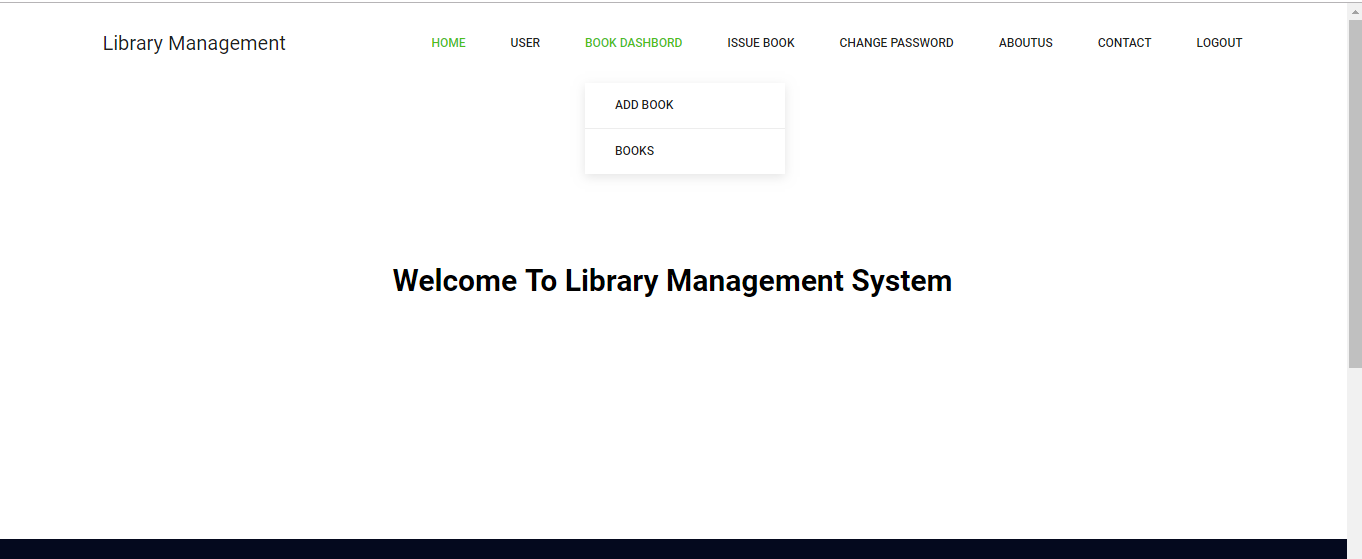
**SignUP**

****

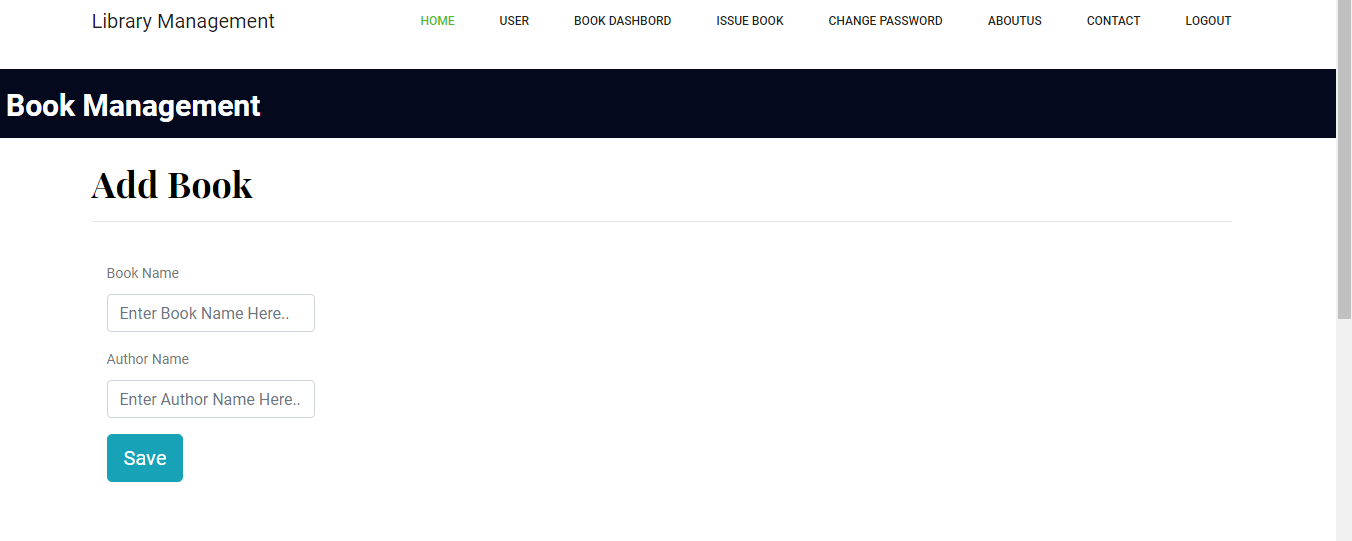
**Login Page**

****

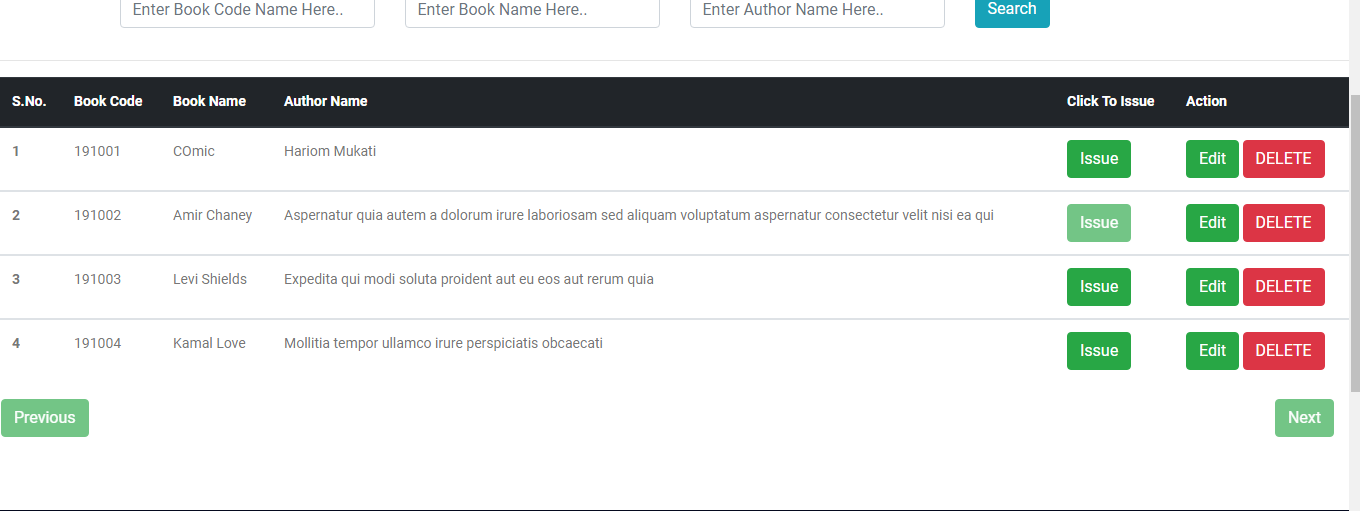
**Admin Dashboard**

****

**Add Book**

****

**Book List**

****

**Limitations and Future Application of the Project**

**Futures Enhancement :**

* In future we can expand this project to add more modules.
* Software can be accessed through internet also.

**Limitation :**

* In this system SMS facility is not available.
* In this system online payment is not available.

**Conclusion:**

This website provides a computerized version of library management system which will benefit the students as well as the staff of the library. It makes entire process online where student can search books, staff can generate reports and do book transactions. It also has a facility for student login where student can login and can see status of books issued as well request for book or give some suggestions. It has a facility of teacher’s login where teachers can add lectures notes and also give necessary suggestion to library and also add info about workshops or events happening in our college or nearby college in the online notice board. There is a future scope of this facility that many more features such as online lectures video tutorials can be added by teachers as well as online assignments submission facility , a feature Of group chat where students can discuss various issues of engineering can be added to this project thus making it more interactive more user friendly and project which fulfills each users need in the best way possible

**Bibliography / References**

**Biblography/ Reference Sites :**

1. http://www.lisbdnet.com/definition-of-library-managemen/
2. <https://www.javatpoint.com/>