Introduction

As we know, today every individual whether it is a student or a teacher or a working profession, etc uses the Internet for research purposes or any course by being at your home. This is nothing but E-learning. So, to develop such a system where all the courses that you want to learn are available at your fingertips this project is developed in the same way which is an E-learning application.

The facilitator here is the admin who will add the courses and add Topics according to the courses added to it; it adds PDF Document for every topic plus the video tutorial. The Student is another user who will view the courses and can enroll in them. Students' roles also have a discussion section where if any student has a query he/she can raise it. The last role is the faculty whose responsibility is to view queries raised by admin and to resolve them.

This application will not only help the students but also the faculty and he/she both can have a common medium to be connected.

Objective

The main objective of this project is to provide a good application where everybody can have access to courses. It provides a platform where students can enroll for the course of their choice.

Here, Admin who is the facilitator can add courses, the student can enroll in them. The faculty will resolve the queries of students.

**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 (Spring and Hibernate)
* Apache Tomcat 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.

Methodologies

There are three main users of this application one is the Admin, the admin here can be the facilitator, the other is the Student and the last one is faculty.

**1) Admin (Facilitator)**

* Admin can VIEW Courses.
* Admin can ADD Courses
* Admin can Add Topics.
* Admin can VIEW its profile.

**2) Student**

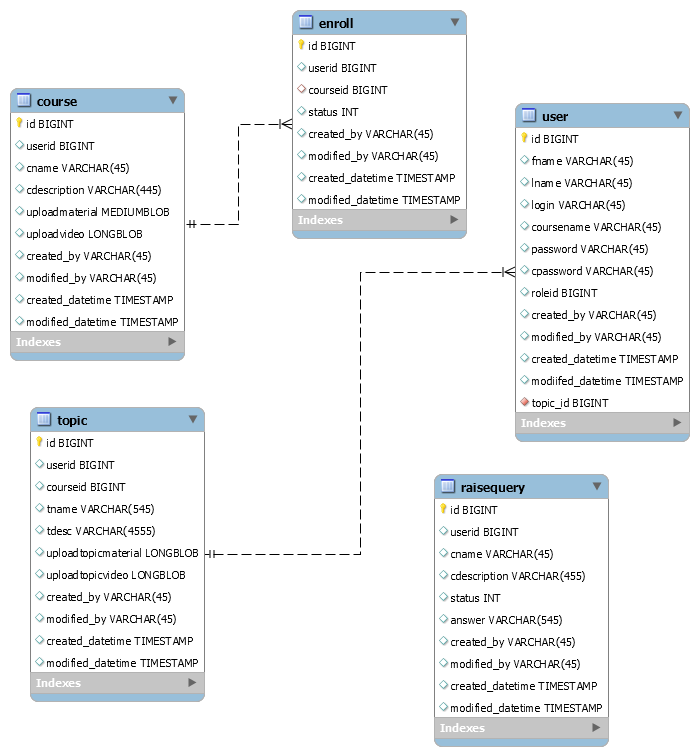
* Students can VIEW Courses.
* Students can search for the course.
* Students can enroll in the courses.
* Students can View the Topics.
* Students can learn using PDF or VIDEO.
* Students can download both PDF as well as VIDEO.
* Students can raise a query.
* Students can VIEW its profile.

**3) Faculty**

* Faculty can VIEW queries raised by students.
* The faculty can Answer the queries.
* Faculty can VIEW its profile.

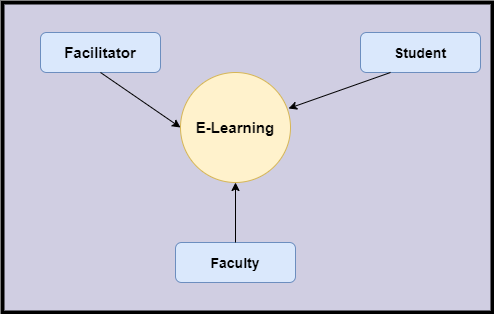
**System Analysis**

**2.1 E R DIAGRAM**

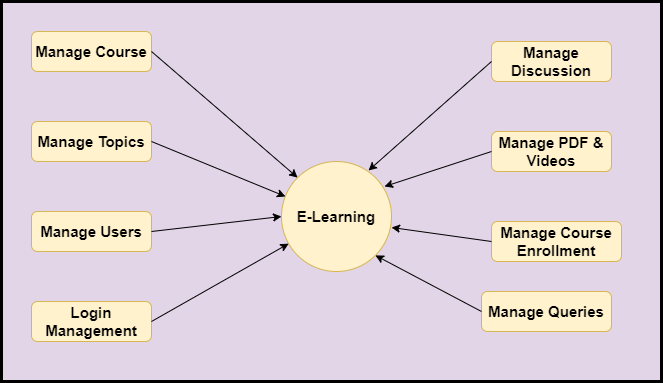


**Data Flow Diagram (DFD)**

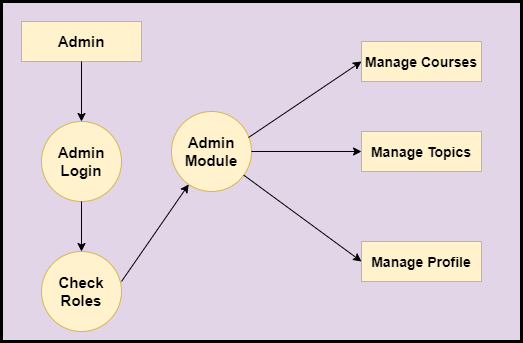
**Level 0:**



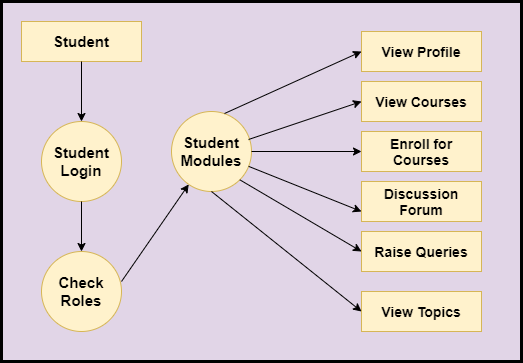
**Level 1**



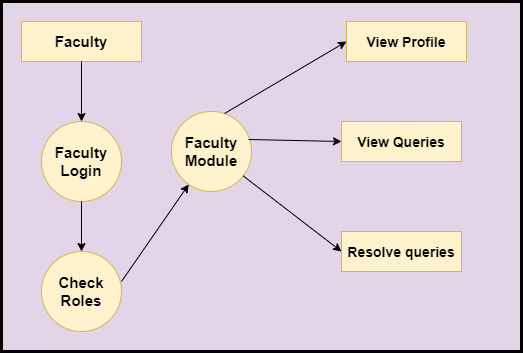
**Level 2-DFD Admin**

****

**Level 2-DFD Student**

****

**Level 2-DFD Faculty**

****

**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.

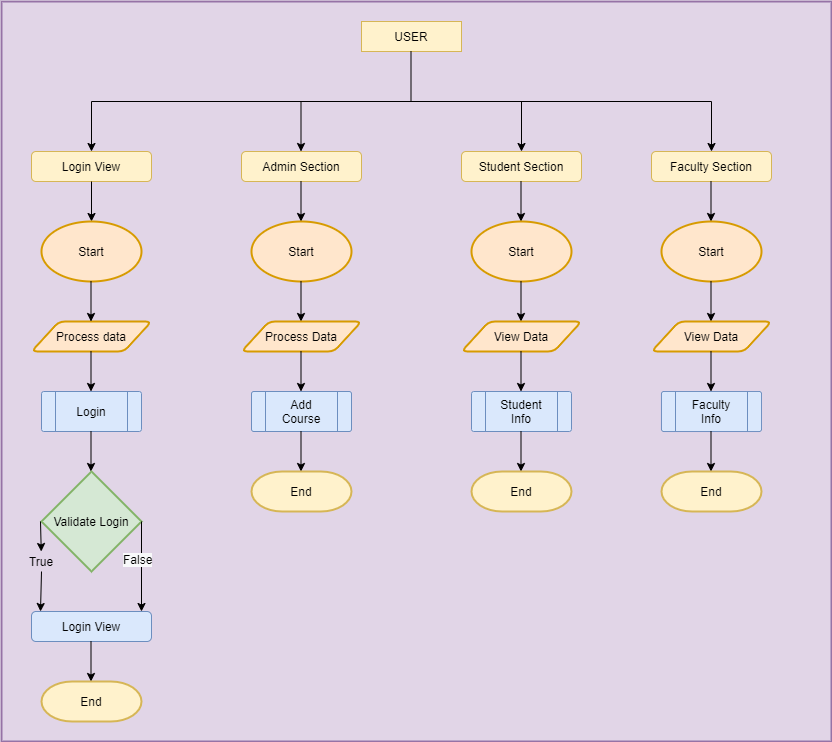
In the proposed system the finance is highly required for the installation of the software’s which can also be recovered by implementing a better system.



**Chapter-3**

**Design**

**System Flow Chart:**

****

**Data dictionary**

**Data validation:**

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**

**Beans:**

package com.elearning.bean;

public class UserBean extends BaseBean {

private String firstName;

private String lastName;

private String login; //Login is the username/Emailid

private String courseName;

private String password;

private String confirmPassword;

private long roleid;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getLogin() {

return login;

}

public void setLogin(String login) {

this.login = login;

}

public String getCourseName() {

return courseName;

}

public void setCourseName(String courseName) {

this.courseName = courseName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getConfirmPassword() {

return confirmPassword;

}

public void setConfirmPassword(String confirmPassword) {

this.confirmPassword = confirmPassword;

}

public long getRoleid() {

return roleid;

}

public void setRoleid(long roleid) {

this.roleid = roleid;

}

@Override

public String getKey() {

// TODO Auto-generated method stub

return String.valueOf(id);

}

@Override

public String getValue() {

// TODO Auto-generated method stub

return firstName+" "+lastName;

}

}

**TopicBean**

package com.elearning.bean;

import java.sql.Blob;

public class TopicBean extends BaseBean {

private long userId;

private long courseId;

private String topicName;

private String topicDescription;

private Blob uploadTopicMaterial;

private Blob uploadTopicVideo;

public long getUserId() {

return userId;

}

public void setUserId(long userId) {

this.userId = userId;

}

public long getCourseId() {

return courseId;

}

public void setCourseId(long courseId) {

this.courseId = courseId;

}

public String getTopicName() {

return topicName;

}

public void setTopicName(String topicName) {

this.topicName = topicName;

}

public String getTopicDescription() {

return topicDescription;

}

public void setTopicDescription(String topicDescription) {

this.topicDescription = topicDescription;

}

public Blob getUploadTopicMaterial() {

return uploadTopicMaterial;

}

public void setUploadTopicMaterial(Blob uploadTopicMaterial) {

this.uploadTopicMaterial = uploadTopicMaterial;

}

public Blob getUploadTopicVideo() {

return uploadTopicVideo;

}

public void setUploadTopicVideo(Blob uploadTopicVideo) {

this.uploadTopicVideo = uploadTopicVideo;

}

@Override

public String getKey() {

// TODO Auto-generated method stub

return String.valueOf(id);

}

@Override

public String getValue() {

// TODO Auto-generated method stub

return topicName;

}

}

**CourseBean**

package com.elearning.bean;

import java.sql.Blob;

public class CourseBean extends BaseBean {

private long userId;

private String courseName;

private String courseDescription;

private Blob uploadMaterial;

private Blob uploadVideo;

public long getUserId() {

return userId;

}

public void setUserId(long userId) {

this.userId = userId;

}

public String getCourseName() {

return courseName;

}

public void setCourseName(String courseName) {

this.courseName = courseName;

}

public String getCourseDescription() {

return courseDescription;

}

public void setCourseDescription(String courseDescription) {

this.courseDescription = courseDescription;

}

public Blob getUploadMaterial() {

return uploadMaterial;

}

public void setUploadMaterial(Blob uploadMaterial) {

this.uploadMaterial = uploadMaterial;

}

public Blob getUploadVideo() {

return uploadVideo;

}

public void setUploadVideo(Blob uploadVideo) {

this.uploadVideo = uploadVideo;

}

@Override

public String getKey() {

// TODO Auto-generated method stub

return null;

}

@Override

public String getValue() {

// TODO Auto-generated method stub

return null;

}

}

**EnrollBean**

package com.elearning.bean;

public class EnrollBean extends BaseBean {

private long userId;

private long courseId;

private int status ;

public long getUserId() {

return userId;

}

public void setUserId(long userId) {

this.userId = userId;

}

public long getCourseId() {

return courseId;

}

public void setCourseId(long courseId) {

this.courseId = courseId;

}

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

@Override

public String getKey() {

// TODO Auto-generated method stub

return null;

}

@Override

public String getValue() {

// TODO Auto-generated method stub

return null;

}

}

**RaiseQueryBean**

package com.elearning.bean;

public class RaiseQueryBean extends BaseBean {

private long userId;

private String courseName;

private String courseDescription;

private int status;

private String answer;

public long getUserId() {

return userId;

}

public void setUserId(long userId) {

this.userId = userId;

}

public String getCourseName() {

return courseName;

}

public void setCourseName(String courseName) {

this.courseName = courseName;

}

public String getCourseDescription() {

return courseDescription;

}

public void setCourseDescription(String courseDescription) {

this.courseDescription = courseDescription;

}

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public String getAnswer() {

return answer;

}

public void setAnswer(String answer) {

this.answer = answer;

}

@Override

public String getKey() {

// TODO Auto-generated method stub

return null;

}

@Override

public String getValue() {

// TODO Auto-generated method stub

return null;

}

}

**Controller**

package com.elearning.ctl;

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 com.elearning.bean.BaseBean;

import com.elearning.bean.UserBean;

import com.elearning.exception.ApplicationException;

import com.elearning.exception.DuplicateRecordException;

import com.elearning.model.UserModel;

import com.elearning.model.UserModelImpl;

import com.elearning.util.DataUtility;

import com.elearning.util.DataValidator;

import com.elearning.util.PropertyReader;

import com.elearning.util.ServletUtility;

/\*\*

\* Servlet implementation class StudentRegCtl

\*/

@WebServlet(name = "StudentRegCtl",urlPatterns = {"/home/signup/student"})

public class StudentRegCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

public static final String OP\_SIGN\_UP = "SignUp";

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public StudentRegCtl() {

super();

// TODO Auto-generated constructor stub

}

@Override

protected boolean validate(HttpServletRequest request) {

boolean pass= true;

if (DataValidator.isNull(request.getParameter("fname"))) {

request.setAttribute("fname",

PropertyReader.getValue("error.require", "First Name"));

pass = false;

} else if (!DataValidator.isName(request.getParameter("fname"))) {

request.setAttribute("fname",

PropertyReader.getValue("error.name", "First Name"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("lname"))) {

request.setAttribute("lname",

PropertyReader.getValue("error.require", "Last Name"));

pass = false;

} else if (!DataValidator.isName(request.getParameter("lname"))) {

request.setAttribute("lname",

PropertyReader.getValue("error.name", "Last Name"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("login"))) {

request.setAttribute("login",

PropertyReader.getValue("error.require", "Login Id"));

pass = false;

} else if (!DataValidator.isEmail(request.getParameter("login"))) {

request.setAttribute("login",

PropertyReader.getValue("error.email", "Login"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("password"))) {

request.setAttribute("password",

PropertyReader.getValue("error.require", "Password"));

pass = false;

} else if (!DataValidator.isPassword(request.getParameter("password"))) {

request.setAttribute("password",

PropertyReader.getValue("error.password", "Password"));

return false;

}else if (!DataValidator.isPassword(request.getParameter("password"))) {

request.setAttribute("password",

PropertyReader.getValue("error.password", "Password"));

return false;

}

if (DataValidator.isNull(request.getParameter("cpassword"))) {

request.setAttribute("cpassword", PropertyReader.getValue(

"error.require", "Confirm Password"));

pass = false;

}

if (!request.getParameter("password").equals(

request.getParameter("cpassword"))

&& !"".equals(request.getParameter("cpassword"))) {

/\*ServletUtility.setErrorMessage("Confirm Password did not match",

request);\*/

request.setAttribute("cpassword", PropertyReader.getValue("error.confirmPassword","Confirm Password"));

pass = false;

}

return pass;

}

@Override

protected BaseBean populateBean(HttpServletRequest request) {

UserBean bean = new UserBean();

bean.setId(DataUtility.getLong(request.getParameter("id")));

bean.setRoleid(2);

bean.setFirstName(DataUtility.getString(request

.getParameter("fname")));

bean.setLastName(DataUtility.getString(request.getParameter("lname")));

bean.setLogin(DataUtility.getString(request.getParameter("login")));

bean.setPassword(DataUtility.getString(request.getParameter("password")));

bean.setConfirmPassword(DataUtility.getString(request

.getParameter("cpassword")));

populateDTO(bean, request);

return bean;

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

// TODO Auto-generated method stub

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

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

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

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

// get model

UserModel model = new UserModelImpl();

long id = DataUtility.getLong(request.getParameter("id"));

if (OP\_SIGN\_UP.equalsIgnoreCase(op)) {

UserBean bean = (UserBean) populateBean(request);

try {

long pk = model.registerUser(bean);

bean.setId(pk);

request.getSession().setAttribute("UserBean", bean);

ServletUtility.setBean(bean, request);

ServletUtility.setSuccessMessage("Student Successfully Registered", request);

ServletUtility.forward(ELearnView.Student\_REG\_VIEW, request, response);

return;

} catch (DuplicateRecordException e) {

ServletUtility.setBean(bean, request);

ServletUtility.setErrorMessage("Login id already exists",

request);

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

} catch (ApplicationException e) {

ServletUtility.handleException(e, request, response);

e.printStackTrace();

return;

}

}else if (OP\_RESET.equalsIgnoreCase(op)) {

ServletUtility.redirect(ELearnView.FACILITATOR\_REG\_VIEW, request, response);

return;

}

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return ELearnView.Student\_REG\_VIEW;

}

}

**StudentLoginCtl**

package com.elearning.ctl;

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 javax.servlet.http.HttpSession;

import com.elearning.bean.BaseBean;

import com.elearning.bean.UserBean;

import com.elearning.exception.ApplicationException;

import com.elearning.model.UserModel;

import com.elearning.model.UserModelImpl;

import com.elearning.util.DataUtility;

import com.elearning.util.DataValidator;

import com.elearning.util.PropertyReader;

import com.elearning.util.ServletUtility;

/\*\*

\* Servlet implementation class StudentLoginCtl

\*/

@WebServlet(name = "StudentLoginCtl", urlPatterns = {"/home/login/student"})

public class StudentLoginCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

public static final String OP\_REGISTER = "Register";

public static final String OP\_SIGN\_IN = "SignIn";

public static final String OP\_SIGN\_UP = "SignUp";

public static final String OP\_LOG\_OUT = "logout";

public static String HIT\_URI = null;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public StudentLoginCtl() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

HttpSession session = request.getSession(true);

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

UserModel model = new UserModelImpl();

long id = DataUtility.getLong(request.getParameter("id"));

if (id > 0) {

UserBean userBean;

try {

userBean = model.findByPK(id);

ServletUtility.setBean(userBean, request);

} catch (Exception e) {

ServletUtility.handleException(e, request, response);

return;

}

} else if (OP\_LOG\_OUT.equals(op)) {

session = request.getSession(false);

session.invalidate();

ServletUtility.setSuccessMessage("You have been logged out successfully", request);

ServletUtility.forward(ELearnView.STUDENT\_LOGIN\_VIEW, request, response);

return;

}

if (session.getAttribute("user") != null) {

ServletUtility.redirect(ELearnView.Dashboard\_CTL, request, response);

return;

}

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

}

@Override

protected BaseBean populateBean(HttpServletRequest request) {

UserBean bean = new UserBean();

bean.setId(DataUtility.getLong(request.getParameter("id")));

bean.setLogin(DataUtility.getString(request.getParameter("login")));

bean.setPassword(DataUtility.getString(request.getParameter("password")));

return bean;

}

@Override

protected boolean validate(HttpServletRequest request) {

boolean pass = true;

String op = request.getParameter("operation");

if (OP\_SIGN\_UP.equals(op) || OP\_LOG\_OUT.equals(op)) {

return pass;

}

String login = request.getParameter("login");

if (DataValidator.isNull(login)) {

request.setAttribute("login", PropertyReader.getValue("error.require", "Login Id"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("password"))) {

request.setAttribute("password", PropertyReader.getValue("error.require", "Password"));

pass = false;

}

return pass;

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

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

HttpSession session = request.getSession(true);

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

// get Model

UserModel model = new UserModelImpl();

//RoleModel role = new RoleModel(); I COMMENTED THIS LINE

long id = DataUtility.getLong(request.getParameter("id"));

if (OP\_SIGN\_IN.equalsIgnoreCase(op)) {

UserBean bean = (UserBean) populateBean(request);

try {

bean = model.authenticate(bean.getLogin(), bean.getPassword());

if (bean != null) {

session.setAttribute("user", bean);

session.setMaxInactiveInterval(10 \* 6000);

long rollId = bean.getRoleid();

String uri = request.getParameter("uri");

if (uri == null || "null".equalsIgnoreCase(uri)) {

ServletUtility.redirect(ELearnView.Student\_Dashboard\_CTL, request, response);

return;

} else {

ServletUtility.redirect(uri, request, response);

}

return;

} else {

bean = (UserBean) populateBean(request);

ServletUtility.setBean(bean, request);

ServletUtility.setErrorMessage("Invalid LoginId And Password", request);

}

} catch (ApplicationException e) {

ServletUtility.handleException(e, request, response);

return;

}

} else if (OP\_SIGN\_UP.equalsIgnoreCase(op)) {

ServletUtility.redirect(ELearnView.SIGNUP\_VIEW, request, response);

return;

}

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

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return ELearnView.STUDENT\_LOGIN\_VIEW;

}

}

**TopicCtl**

package com.elearning.ctl;

import java.io.IOException;

import java.io.InputStream;

import java.sql.Blob;

import java.sql.SQLException;

import java.util.Iterator;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.MultipartConfig;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import javax.servlet.http.Part;

import javax.sql.rowset.serial.SerialBlob;

import javax.sql.rowset.serial.SerialException;

import com.elearning.bean.BaseBean;

import com.elearning.bean.CourseBean;

import com.elearning.bean.TopicBean;

import com.elearning.bean.UserBean;

import com.elearning.exception.ApplicationException;

import com.elearning.exception.DuplicateRecordException;

import com.elearning.model.CourseModel;

import com.elearning.model.TopicModel;

import com.elearning.util.DataUtility;

import com.elearning.util.DataValidator;

import com.elearning.util.PropertyReader;

import com.elearning.util.ServletUtility;

/\*\*

\* Servlet implementation class TopicRegCtl

\*/

@WebServlet(name = "TopicRegCtl", urlPatterns = {"/home/login/facilitator/viewcourse/topic"})

@MultipartConfig(maxFileSize = 16177215)

public class TopicRegCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

public static final String OP\_SIGN\_UP = "SignUp";

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public TopicRegCtl() {

super();

// TODO Auto-generated constructor stub

}

@Override

protected void preload(HttpServletRequest request) {

CourseModel model = new CourseModel();

try {

List l = model.list();

System.out.println("is list returning?" + l.get(0));

request.setAttribute("courseList", l);

Iterator<CourseBean> itr=l.iterator();

while(itr.hasNext()){

CourseBean b=new CourseBean();

b=itr.next();

System.out.println("b"+b.getCourseName());

}

} catch (ApplicationException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

protected boolean validate(HttpServletRequest request) {

boolean pass =true;

if (DataValidator.isNull(request.getParameter("cname"))) {

request.setAttribute("cname",

PropertyReader.getValue("error.require", "Cousre Name"));

pass = false;

System.out.println("Pass 1 "+pass);

}

if (DataValidator.isNull(request.getParameter("tname"))) {

request.setAttribute("tname",

PropertyReader.getValue("error.require", "Topic Name"));

System.out.println("Pass 3 "+pass);

pass = false;

}

if (DataValidator.isNull(request.getParameter("tdesc"))) {

request.setAttribute("tdesc",

PropertyReader.getValue("error.require", "Topic Description"));

System.out.println("Pass 4 "+pass);

pass = false;

}

/\*if (DataValidator.isNull(request.getParameter("uploadmaterial"))) {

request.setAttribute("uploadmaterial",

PropertyReader.getValue("error.require", "Upload Material"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("uploadvideo"))) {

request.setAttribute("uploadvideo",

PropertyReader.getValue("error.require", "Upload Video"));

pass = false;

} \*/

System.out.println("pass is "+pass);

return pass;

}

public Blob MaterialUpload(Part part) throws IOException {

InputStream inputStream = null;

Blob blob = null;

inputStream = part.getInputStream();

byte[] b = new byte[inputStream.available()];

inputStream.read(b);

try {

blob = new SerialBlob(b);

} catch (SerialException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return blob;

}

@Override

protected BaseBean populateBean(HttpServletRequest request) {

TopicBean bean = new TopicBean();

bean.setId(DataUtility.getLong(request.getParameter("id")));

bean.setCourseId(DataUtility.getLong(request.getParameter("cname")));

bean.setTopicName(DataUtility.getString(request.getParameter("tname")));

bean.setTopicDescription(DataUtility.getString(request.getParameter("tdesc")));

Blob blob=null;

Blob blob2=null; // For Video Upload

Part filepart;

try {

filepart = request.getPart("uploadmaterial");

Part filepart2 = request.getPart("uploadvideo");

blob = MaterialUpload(filepart);

blob2 = MaterialUpload(filepart2);

} catch (IOException | ServletException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

//Upload material method called

bean.setUploadTopicMaterial(blob);

bean.setUploadTopicVideo(blob2);

populateDTO(bean, request);

return bean;

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

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

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

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

System.out.println("\*\*\*\*\*\*in do post\*\*\*\*\*");

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

// get model

TopicModel model = new TopicModel();

long id = DataUtility.getLong(request.getParameter("id"));

if (OP\_SIGN\_UP.equalsIgnoreCase(op)) {

System.out.println("\*\*\*\*\*here\*\*\*\*");

TopicBean bean = (TopicBean) populateBean(request);

HttpSession session=request.getSession();

UserBean uBean=(UserBean)session.getAttribute("user");

bean.setUserId(uBean.getId());

try {

System.out.println("//////hello\*\*\*\*\*");

long pk = model.add(bean);

ServletUtility.setBean(bean, request);

ServletUtility.setSuccessMessage("Topic is successfully Added",request);

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

return;

}

catch(DuplicateRecordException e){

ServletUtility.setBean(bean, request);

ServletUtility.setErrorMessage("Login id already exists",

request);

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

} catch (ApplicationException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return ELearnView.TOPIC\_REG\_VIEW;

}

}

**Model:**

QueryModel

package com.elearning.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import com.elearning.bean.CourseBean;

import com.elearning.bean.RaiseQueryBean;

import com.elearning.bean.TopicBean;

import com.elearning.bean.UserBean;

import com.elearning.exception.ApplicationException;

import com.elearning.exception.DatabaseException;

import com.elearning.exception.DuplicateRecordException;

import com.elearning.util.JDBCDataSource;

public class QueryModel {

public Integer nextPk() throws DatabaseException {

// TODO Auto-generated method stub

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM RAISEQUERY");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk + 1;

}

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

Connection conn = null;

int pk = 0;

System.out.println("In raise query model "+bean.getId()+ bean.getCourseName());

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 RAISEQUERY VALUES(?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

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

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

pstmt.setInt(5, bean.getStatus());

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

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

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

pstmt.setTimestamp(9, bean.getCreatedDatetime());

pstmt.setTimestamp(10, 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 RaiseQueryBean findByPK(long pk) throws ApplicationException {

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

RaiseQueryBean 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 RaiseQueryBean();

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

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

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

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

bean.setStatus(rs.getInt(5));

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

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

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

bean.setCreatedDatetime(rs.getTimestamp(9));

bean.setModifiedDatetime(rs.getTimestamp(10));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return bean;

}

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

Connection conn = null;

System.out.println("\*\*\*\*\*\*\*\*\*Id\*\*\*\*\*\*\*\*"+bean.getId());

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE RAISEQUERY SET STATUS=?,ANSWER=?,CREATED\_BY=?,MODIFIED\_BY=?,CREATED\_DATETIME=?,MODIFIED\_DATETIME=? WHERE ID=?");

pstmt.setInt(1, bean.getStatus());

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

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

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

pstmt.setTimestamp(5, bean.getCreatedDatetime());

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

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

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

try {

conn.rollback();

} catch (Exception ex) {

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

}

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

} finally {

JDBCDataSource.closeConnection(conn);

}

}

public List search(RaiseQueryBean bean) throws ApplicationException {

// TODO Auto-generated method stub

return search(bean, 0, 0);

}

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

// TODO Auto-generated method stub

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

if (bean != null) {

if (bean.getId() > 0) {

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

}

}

// 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 RaiseQueryBean();

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

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

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

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

bean.setStatus(rs.getInt(5));

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

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

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

bean.setCreatedDatetime(rs.getTimestamp(9));

bean.setModifiedDatetime(rs.getTimestamp(10));

list.add(bean);

}

rs.close();

} catch (Exception e) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return list;

}

public List list() throws ApplicationException {

return list(0, 0);

}

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

ArrayList list = new ArrayList();

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

// 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);

}

System.out.println("sql in list user :" + sql);

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

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

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

RaiseQueryBean bean = new RaiseQueryBean();

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

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

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

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

bean.setStatus(rs.getInt(5));

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

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

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

bean.setCreatedDatetime(rs.getTimestamp(9));

bean.setModifiedDatetime(rs.getTimestamp(10));

list.add(bean);

}

rs.close();

} catch (Exception e) {

throw new ApplicationException("Exception : Exception in getting list of users");

} finally {

JDBCDataSource.closeConnection(conn);

}

return list;

}

}

**EnrollModel**

package com.elearning.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import com.elearning.bean.CourseBean;

import com.elearning.bean.EnrollBean;

import com.elearning.bean.RaiseQueryBean;

import com.elearning.exception.ApplicationException;

import com.elearning.exception.DatabaseException;

import com.elearning.exception.DuplicateRecordException;

import com.elearning.util.JDBCDataSource;

public class EnrollModel {

public Integer nextPk() throws DatabaseException {

// TODO Auto-generated method stub

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM ENROLL");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk + 1;

}

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

Connection conn = null;

int pk = 0;

System.out.println("\*\*\*\*"+bean.getId()+ ""+bean.getUserId()+ " "+bean.getCourseId());

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 ENROLL VALUES(?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

pstmt.setLong(3, bean.getCourseId());

pstmt.setInt(4, bean.getStatus());

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());

}

e.printStackTrace();

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk;

}

public EnrollBean findByPK(long pk) throws ApplicationException {

StringBuffer sql = new StringBuffer("SELECT \* FROM ENROLL WHERE USERID=?");

EnrollBean 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 EnrollBean();

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

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

bean.setCourseId(rs.getLong(3));

bean.setStatus(rs.getInt(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();

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return bean;

}

}

**UserModel**

package com.elearning.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import com.elearning.bean.UserBean;

import com.elearning.exception.ApplicationException;

import com.elearning.exception.DatabaseException;

import com.elearning.exception.DuplicateRecordException;

import com.elearning.exception.RecordNotFoundException;

import com.elearning.util.JDBCDataSource;

public class UserModelImpl implements UserModel {

@Override

public Integer nextPk() throws DatabaseException {

// TODO Auto-generated method stub

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM USER");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk + 1;

}

@Override

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 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.getCourseName());

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

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

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

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;

}

@Override

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 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);

}

}

@Override

public UserBean findByLogin(String login) throws ApplicationException {

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

UserBean bean = null;

Connection conn = null;

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

System.out.println("login is "+login);

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.setCourseName(rs.getString(5));

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

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

bean.setRoleid(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();

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return bean;

}

@Override

public UserBean findByPK(long pk) throws ApplicationException {

StringBuffer sql = new StringBuffer("SELECT \* FROM 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.setCourseName(rs.getString(5));

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

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

bean.setRoleid(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();

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return bean;

}

@Override

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

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 USER SET FNAME=?,LNAME=?,LOGIN=?,COURSENAME=?,PASSWORD=?,CPASSWORD=?,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.getCourseName());

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

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

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

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

pstmt.setTimestamp(9, bean.getCreatedDatetime());

pstmt.setTimestamp(10, bean.getModifiedDatetime());

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

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (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);

}

}

@Override

public List search(UserBean bean) throws ApplicationException {

// TODO Auto-generated method stub

return search(bean, 0, 0);

}

@Override

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

// TODO Auto-generated method stub

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

if (bean != null) {

if (bean.getId() > 0) {

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

}

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

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

}

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

sql.append(" AND LNAME like '" + bean.getLastName() + "%'");

}

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

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

}

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

sql.append(" AND PASSWORD like '" + bean.getPassword() + "%'");

}

if (bean.getRoleid() > 0) {

sql.append(" AND ROLEID = " + bean.getRoleid());

}

}

// 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.setCourseName(rs.getString(5));

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

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

bean.setRoleid(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) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return list;

}

@Override

public List list() throws ApplicationException {

return list(0, 0);

}

@Override

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

ArrayList list = new ArrayList();

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

// 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);

}

System.out.println("sql in list user :" + sql);

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

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

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

UserBean bean = new UserBean();

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

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

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

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

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

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

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

bean.setRoleid(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) {

throw new ApplicationException("Exception : Exception in getting list of users");

} finally {

JDBCDataSource.closeConnection(conn);

}

return list;

}

@Override

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

StringBuffer sql = new StringBuffer("SELECT \* FROM 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.setCourseName(rs.getString(5));

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

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

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

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

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

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

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

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

}

} catch (Exception e) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return bean;

}

@Override

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

long pk = add(bean);

return pk;

}

@Override

public int forgetPassword(String newpassword, String retypepassword, String login) throws Exception {

UserBean existbean = findByLogin(login);

if (existbean == null) {

throw new RecordNotFoundException("Login Id Doesn't exist");

}

Connection conn = null;

String sql = "UPDATE USER SET PASSWORD = ?, CPASSWORD = ? WHERE LOGIN = ?";

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setString(1, newpassword);

pstmt.setString(2, retypepassword);

pstmt.setString(3, login);

int i=pstmt.executeUpdate();

return i;

}

}

**CourseModel**

package com.elearning.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import com.elearning.bean.CourseBean;

import com.elearning.bean.UserBean;

import com.elearning.exception.ApplicationException;

import com.elearning.exception.DatabaseException;

import com.elearning.exception.DuplicateRecordException;

import com.elearning.util.JDBCDataSource;

public class CourseModel {

public Integer nextPk() throws DatabaseException {

// TODO Auto-generated method stub

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM COURSE");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return pk + 1;

}

public CourseBean findByPK(long pk) throws ApplicationException {

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

CourseBean 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 CourseBean();

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

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

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

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

bean.setUploadMaterial(rs.getBlob(5));

bean.setUploadVideo(rs.getBlob(6));

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

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

bean.setCreatedDatetime(rs.getTimestamp(9));

bean.setModifiedDatetime(rs.getTimestamp(10));

}

rs.close();

} catch (Exception e) {

e.printStackTrace();

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return bean;

}

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

Connection conn = null;

int pk = 0;

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 COURSE VALUES(?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

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

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

pstmt.setBlob(5, bean.getUploadMaterial());

pstmt.setBlob(6, bean.getUploadVideo());

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

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

pstmt.setTimestamp(9, bean.getCreatedDatetime());

pstmt.setTimestamp(10, 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 void updateMaterialVideo(CourseBean bean) throws ApplicationException, DuplicateRecordException {

Connection conn = null;

System.out.println("\*\*\*\*\*\*\*\*\*Id\*\*\*\*\*\*\*\*"+bean.getId());

System.out.println("\*\*\*\*\*\*"+bean.getUploadMaterial());

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE COURSE SET UPLOADMATERIAL=?,UPLOADVIDEO=?,CREATED\_BY=?,MODIFIED\_BY=?,CREATED\_DATETIME=?,MODIFIED\_DATETIME=? WHERE ID=?");

pstmt.setBlob(1, bean.getUploadMaterial());

pstmt.setBlob(2, bean.getUploadVideo());

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

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

pstmt.setTimestamp(5, bean.getCreatedDatetime());

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

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

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

try {

conn.rollback();

} catch (Exception ex) {

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

}

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

} finally {

JDBCDataSource.closeConnection(conn);

}

}

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

Connection conn = null;

System.out.println("\*\*\*\*\*\*\*\*\*Id\*\*\*\*\*\*\*\*"+bean.getId());

System.out.println("\*\*\*\*\*\*"+bean.getUploadMaterial());

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE COURSE SET USERID=?,CNAME=?,CDESCRIPTION=?,UPLOADMATERIAL=?,UPLOADVIDEO=?,CREATED\_BY=?,MODIFIED\_BY=?,CREATED\_DATETIME=?,MODIFIED\_DATETIME=? WHERE ID=?");

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

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

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

pstmt.setBlob(4, bean.getUploadMaterial());

pstmt.setBlob(5, bean.getUploadVideo());

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

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

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

pstmt.setTimestamp(9, bean.getModifiedDatetime());

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

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

try {

conn.rollback();

} catch (Exception ex) {

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

}

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

} finally {

JDBCDataSource.closeConnection(conn);

}

}

public List search(CourseBean bean) throws ApplicationException {

// TODO Auto-generated method stub

return search(bean, 0, 0);

}

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

// TODO Auto-generated method stub

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

if (bean != null) {

if (bean.getId() > 0) {

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

}

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

sql.append(" AND CNAME like '" + bean.getCourseName() + "%'");

}

}

// 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 CourseBean();

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

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

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

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

bean.setUploadMaterial(rs.getBlob(5));

bean.setUploadVideo(rs.getBlob(6));

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

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

bean.setCreatedDatetime(rs.getTimestamp(9));

bean.setModifiedDatetime(rs.getTimestamp(10));

list.add(bean);

}

rs.close();

} catch (Exception e) {

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

} finally {

JDBCDataSource.closeConnection(conn);

}

return list;

}

public List list() throws ApplicationException {

return list(0, 0);

}

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

ArrayList list = new ArrayList();

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

if (pageSize > 0) {

pageNo = (pageNo - 1) \* pageSize;

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

}

System.out.println("sql in list user :" + sql);

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

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

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

CourseBean bean = new CourseBean();

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

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

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

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

bean.setUploadMaterial(rs.getBlob(5));

bean.setUploadVideo(rs.getBlob(6));

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

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

bean.setCreatedDatetime(rs.getTimestamp(9));

bean.setModifiedDatetime(rs.getTimestamp(10));

list.add(bean);

}

rs.close();

} catch (Exception e) {

throw new ApplicationException("Exception : Exception in getting list of users");

} finally {

JDBCDataSource.closeConnection(conn);

}

return list;

}

}

**Implementation and Testing:**

**Black-Box Testing**:

Black Box Testing, also known as Behavioural 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
* Behaviour 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.

**Limitations and Future Application of the Project**

**Futures Enhancement:**

* In future we can expand this project on the web.
* In future, we can make a display popular courses which has more enrollment.

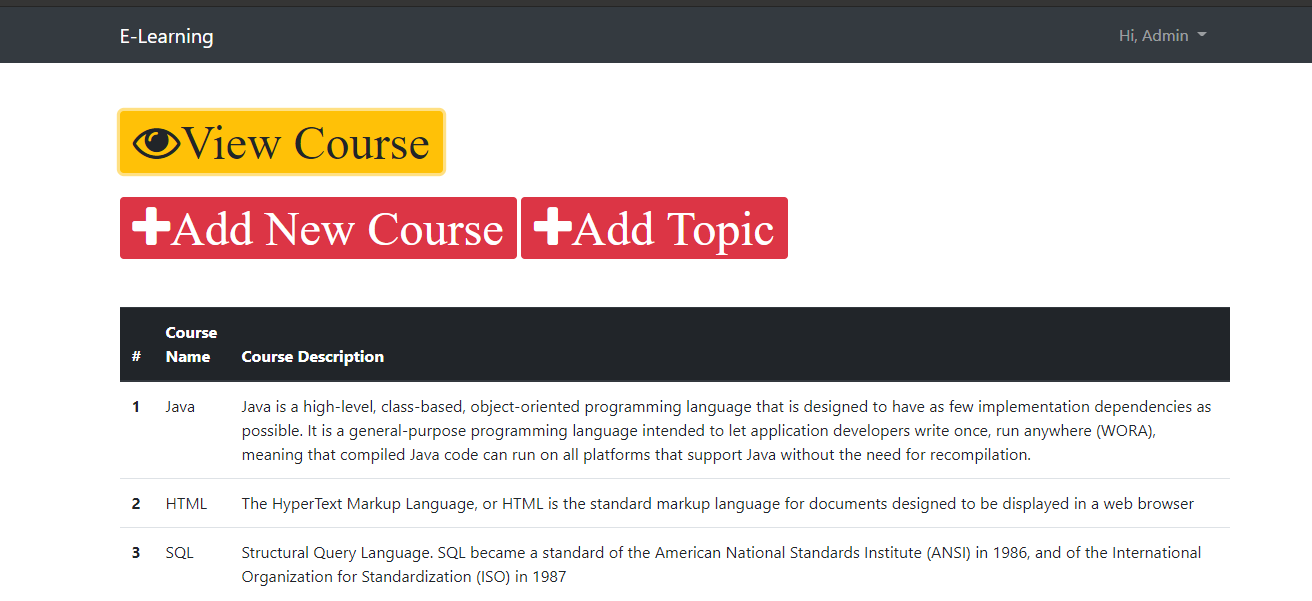
**Limitation :**

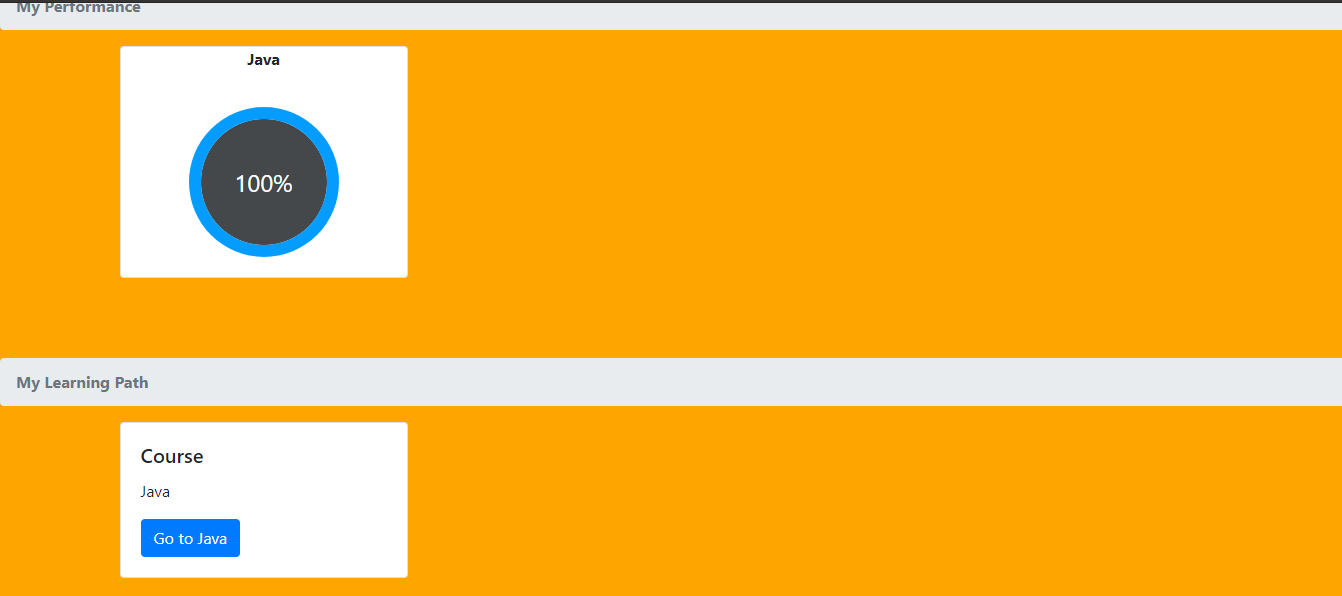
In this, we don’t block the students from accessing any course.

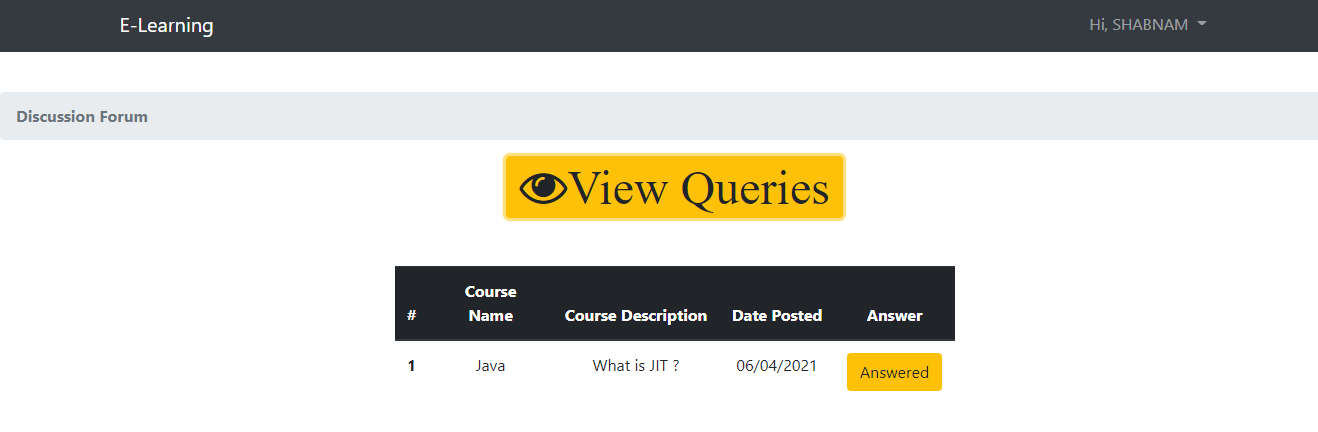
**Screen Snapshot**

****

****

****

****

****

**Conclusion**

E-learning has become a new trend in this era. So this project provides us the good help in achieving all the goals. The whole purpose of this application is to provide a medium where the faculty and student can interact with each other like if student have a query he/she can raise it also, students can make an enrollment in the courses available.

This project can be used by the colleges, Universities where the E-learning is a priority. It also provides a platform for students to view/enroll different courses by admin.

This application is designed in such a way that any future modification can be done most easily.