**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:**

* **Online Examination System**

Its a web application.

The Proposed Project is named as **“Online Examination SYSTEM”** where Treatments are all types.

“ 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:**

This Web Application provides facility to conduct online examination worldwide.

It saves time as it allows number of students to give the exam at a time and displays the results as the test gets over, so no need to wait for the result. It is automatically generated by the server.

Administrator has a privilege to create, modify and delete the test papers and its particular questions.

User can register, login and give the test with his specific id, and can see the results as well.

**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 goals and objectives of the Online Examination System are as follows:

* 1. To provide a bug free online examination system to the students.
  2. To maintain the results or student data online which will be easy to maintain.
  3. To provide an easy interface for online examination and facility to get intent result.

**1.5 How the existing system works?**

The existing system was an automated system but It was found to be inefficient in meeting the growing demands of population and different online examination take a lot of time and money for to provide facility for online examination. But still they are not able to provide a good or bug free product.

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

The existing system is not user-friendly system.

* In old system very difficult to handle multiple user in same time.
* Costly.
* It is time consuming process.

They appreciate the significance of timing, of in receipt of there previous to the opposition. Try and experienced method ensures that we follow an unsurprising, low risk path to achieve consequences.

Our track record is indication to complex projects deliver within and evens earlier than agenda.

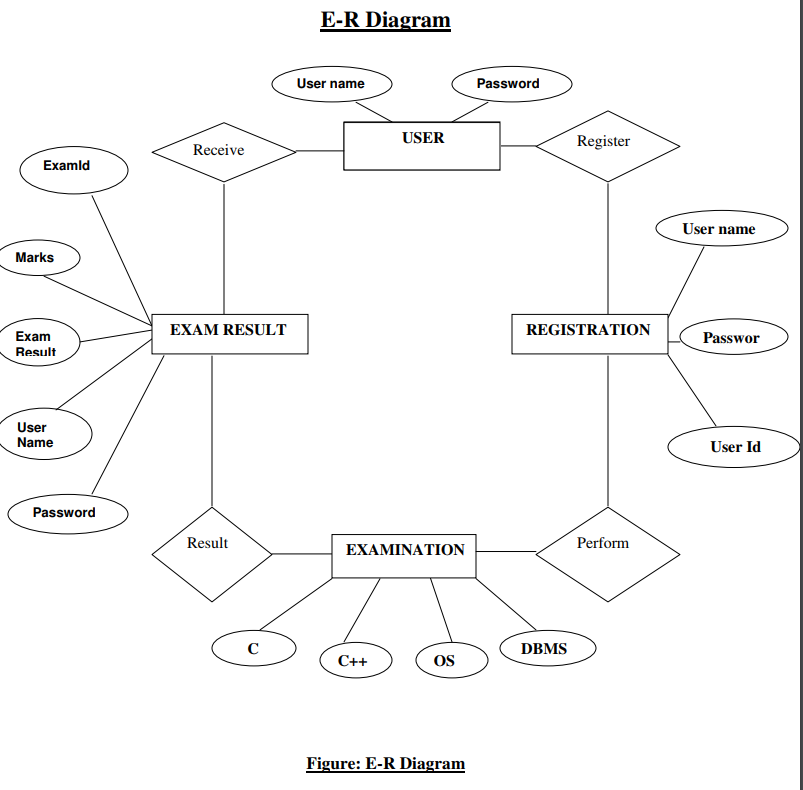
**1.7** How are you going to improve on the issues/problems with the existing system?

* Finding a reliable system which can user by multiple user in a same time.
* Maintain the record of student so it will be easy to access any time 24\*4.
* Facility of administer to maintain the data of exams as well as students.
* Promote health by preventing future health problems.
* This Web Application provides facility to conduct online examination worldwide.
* It saves time as it allows number of students to give the exam at a time and displays the results as the test gets over, so no need to wait for the result. It is automatically generated by the server.

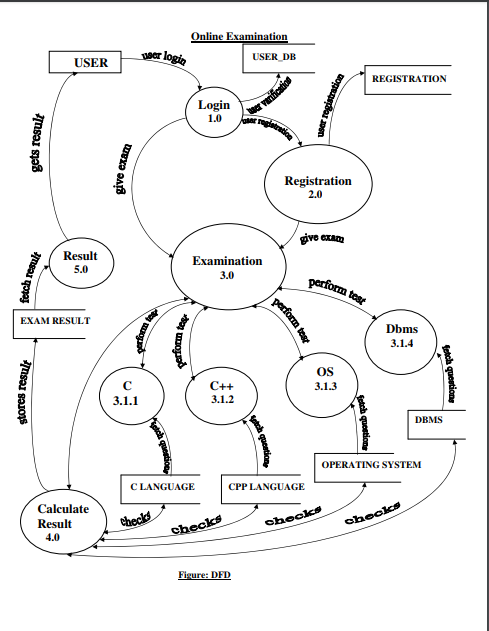
**Chapter-2**

**System Analysis**

**2.1 E R DIAGRAM:**

****

**Data Flow Diagram(DFD)**

****

**2.3Functional Decomposation:**

Result reports

Registration

Login

Online Examination System

Admin

Students

Admin

**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.

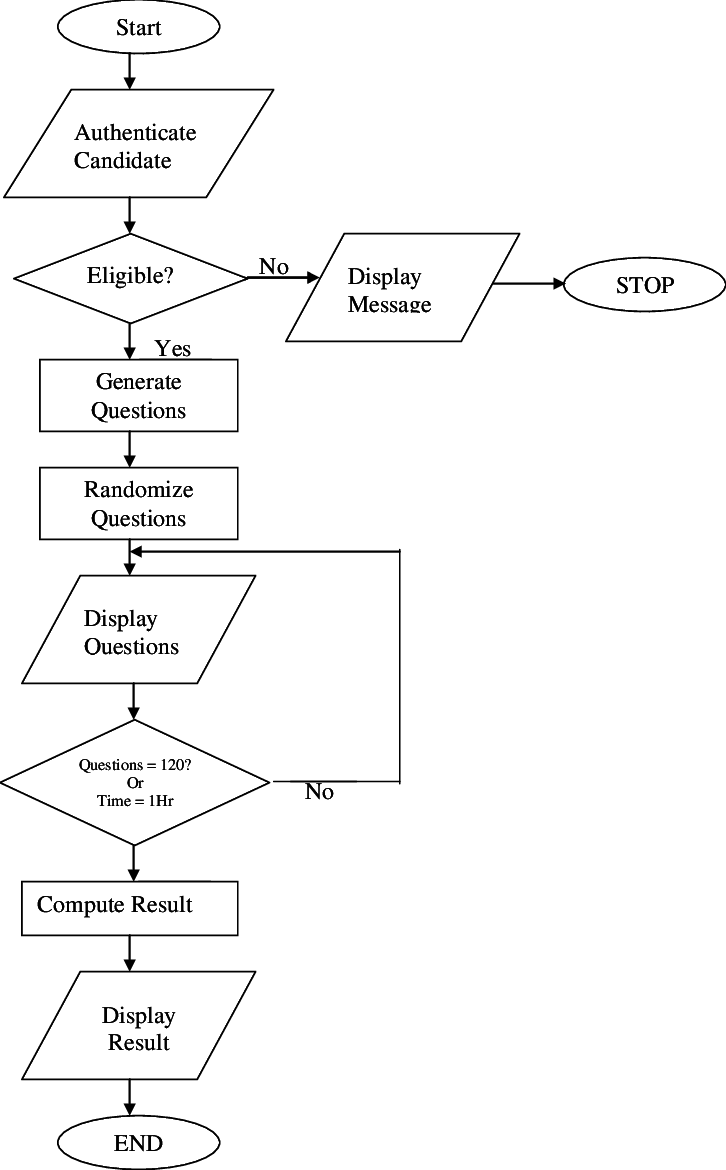


**Chapter-3**

**Design**

**3.1 Data flow diagram:.**

**System Flow Chart :**

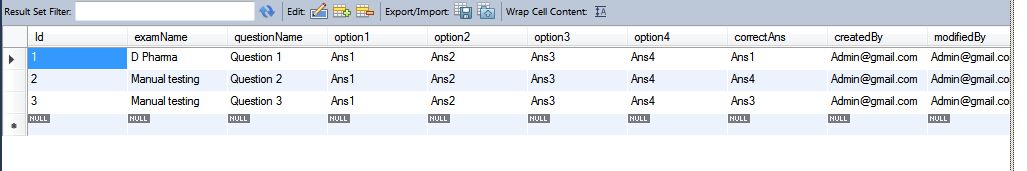


**Data dictionary**

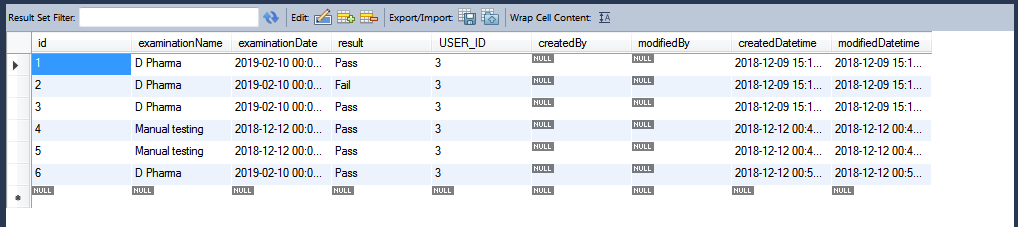
**3.2.1 Exam:**

****

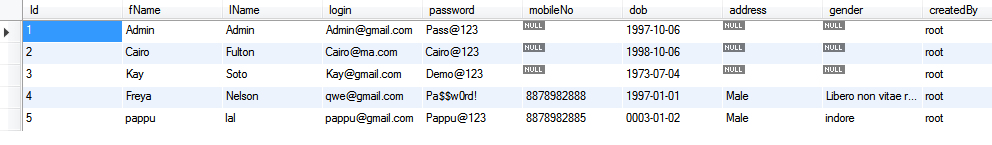
**3.2.2 Questions:**

****

**3.2.3 Results**

****

**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:**

package MyPack;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author Oops

\*/

public class MyDb {

Connection con;

public Connection getCon(){

try {

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/onlineexamination","root","root");

} catch (ClassNotFoundException ex) {

Logger.getLogger(MyDb.class.getName()).log(Level.SEVERE, null, ex);

System.out.println(ex);

} catch (SQLException ex) {

Logger.getLogger(MyDb.class.getName()).log(Level.SEVERE, null, ex);

System.out.println(ex);

}

return con;

}

}

**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");

}

}

================ResultModel==============================================

package in.com.online.exam.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.com.online.exam.bean.ResultBean;

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.JDBCDataSource;

public class ResultModel {

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\_RESULT");

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(ResultBean 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 EX\_RESULT VALUES(?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

pstmt.setDate(3, new java.sql.Date(bean.getExaminationDate().getTime()));

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

pstmt.setLong(5,bean.getUser\_id());

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

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

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

pstmt.setTimestamp(9, 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 List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of User with pagination

\*

\* @return list : List of users

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\* @throws DatabaseException

\*/

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

log.debug("Model list Started");

ArrayList list = new ArrayList();

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

// 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()) {

ResultBean bean = new ResultBean();

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

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

bean.setExaminationDate(rs.getDate(3));

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

bean.setUser\_id(rs.getLong(5));

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) {

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

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

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

public List search(ResultBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

/\*\*

\* Search User with pagination

\*

\* @return list : List of Users

\* @param bean

\* : Search Parameters

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\*

\* @throws DatabaseException

\*/

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

log.debug("Model search Started");

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

if (bean != null) {

if (bean.getId() > 0) {

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

}

if (bean.getUser\_id() > 0) {

sql.append(" AND USER\_ID ="+ bean.getUser\_id());

}

}

// 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();

System.out.println("---------------------hihi-------------bjv");

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

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ResultBean();

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

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

bean.setExaminationDate(rs.getDate(3));

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

bean.setUser\_id(rs.getLong(5));

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

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

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

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

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;

}

}

===========================================Question Model==========================================

package in.com.online.exam.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.com.online.exam.bean.ExamBean;

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

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

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.DataUtility;

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

public class QuestionModel {

private static Logger log = Logger.getLogger(QuestionModel.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\_QUESTION");

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(QuestionBean bean) throws ApplicationException, DuplicateRecordException {

Connection conn = null;

int pk = 0;

QuestionBean existbean = findByQuestionName(bean.getQuestionName());

if (existbean != null) {

throw new DuplicateRecordException("Question Name already exists");

}

ExamModel model=new ExamModel();

ExamBean Ebean=model.findByPK(DataUtility.getLong(bean.getExamName()));

bean.setExamName(Ebean.getExamName());

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\_QUESTION VALUES(?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

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

pstmt.setString(4,bean.getOption1());

pstmt.setString(5,bean.getOption2());

pstmt.setString(6,bean.getOption3());

pstmt.setString(7,bean.getOption4());

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

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 QuestionBean findByQuestionName(String name) throws ApplicationException {

log.debug("Model findByLogin Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM EX\_Question WHERE QuestionName=?");

QuestionBean bean = null;

Connection conn = null;

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

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setString(1, name);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new QuestionBean();

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

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

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

bean.setOption1(rs.getString(4));

bean.setOption2(rs.getString(5));

bean.setOption3(rs.getString(6));

bean.setOption4(rs.getString(7));

bean.setCorrectAns(rs.getString(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 QuestionBean findByPK(long id) throws ApplicationException {

log.debug("Model findByLogin Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM EX\_Question WHERE id=?");

QuestionBean bean = null;

Connection conn = null;

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

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setLong(1, id);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new QuestionBean();

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

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

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

bean.setOption1(rs.getString(4));

bean.setOption2(rs.getString(5));

bean.setOption3(rs.getString(6));

bean.setOption4(rs.getString(7));

bean.setCorrectAns(rs.getString(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 List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of User with pagination

\*

\* @return list : List of users

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\* @throws DatabaseException

\*/

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

log.debug("Model list Started");

ArrayList list = new ArrayList();

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

// 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()) {

QuestionBean bean = new QuestionBean();

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

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

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

bean.setOption1(rs.getString(4));

bean.setOption2(rs.getString(5));

bean.setOption3(rs.getString(6));

bean.setOption4(rs.getString(7));

bean.setCorrectAns(rs.getString(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 users");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

public List search(QuestionBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

/\*\*

\* Search User with pagination

\*

\* @return list : List of Users

\* @param bean

\* : Search Parameters

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\*

\* @throws DatabaseException

\*/

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

log.debug("Model search Started");

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

if (bean != null) {

if (bean.getId() > 0) {

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

}

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

sql.append(" AND ExamName like '" + bean.getExamName() + "%'");

}

}

// 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 QuestionBean();

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

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

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

bean.setOption1(rs.getString(4));

bean.setOption2(rs.getString(5));

bean.setOption3(rs.getString(6));

bean.setOption4(rs.getString(7));

bean.setCorrectAns(rs.getString(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 user");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

}

=========================================Exam Model======================================

package in.com.online.exam.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.com.online.exam.bean.ExamBean;

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

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.JDBCDataSource;

public class ExamModel {

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\_RESULT");

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(ExamBean bean) throws ApplicationException, DuplicateRecordException {

Connection conn = null;

int pk = 0;

ExamBean existbean = findByExamName(bean.getExamName());

if (existbean != null) {

throw new DuplicateRecordException("Exam Name 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\_EXAM VALUES(?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

pstmt.setDate(3, new java.sql.Date(bean.getExamDate().getTime()));

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

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

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

pstmt.setTimestamp(7, 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 ExamBean findByExamName(String name) throws ApplicationException {

log.debug("Model findByLogin Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM EX\_EXAM WHERE EXAMNAME=?");

ExamBean bean = null;

Connection conn = null;

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

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setString(1, name);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ExamBean();

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

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

bean.setExamDate(rs.getDate(3));

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

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

bean.setCreatedDatetime(rs.getTimestamp(6));

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

}

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 ExamBean findByPK(long id) throws ApplicationException {

log.debug("Model findByLogin Started");

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

ExamBean bean = null;

Connection conn = null;

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

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setLong(1, id);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ExamBean();

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

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

bean.setExamDate(rs.getDate(3));

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

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

bean.setCreatedDatetime(rs.getTimestamp(6));

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

}

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 List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of User with pagination

\*

\* @return list : List of users

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\* @throws DatabaseException

\*/

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

log.debug("Model list Started");

ArrayList list = new ArrayList();

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

// 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()) {

ExamBean bean = new ExamBean();

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

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

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

bean.setExamDate(rs.getDate(3));

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

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

bean.setCreatedDatetime(rs.getTimestamp(6));

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

list.add(bean);

}

rs.close();

} catch (Exception e) {

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

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

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

}

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

**package** in.com.online.exam.controller;

**publicinterface** ORSView {

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

**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 ***MARKSHEET\_VIEW*** = ***PAGE\_FOLDER*** + "/MarksheetView.jsp";

**public** String ***MARKSHEET\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/MarksheetListView.jsp";

**public** String ***GET\_MARKSHEET\_VIEW*** = ***PAGE\_FOLDER*** + "/GetMarksheetView.jsp";

**public** String ***MARKSHEET\_MERIT\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/MarksheetMeritListView.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 ***QUESTION\_VIEW*** = ***PAGE\_FOLDER*** + "/QuestionView.jsp";

**public** String ***COLLEGE\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/CollegeListView.jsp";

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

**public** String ***STUDENT\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/StudentListView.jsp";

**public** String ***FACULTY\_VIEW*** = ***PAGE\_FOLDER*** + "/FacultyView.jsp";

**public** String ***FACULTY\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/FacultyListView.jsp";

**public** String ***EXAM\_START\_VIEW*** = ***PAGE\_FOLDER*** + "/ExamStartView.jsp";

**public** String ***QUESTION\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/QuestionListView.jsp";

**public** String ***RESULT\_VIEW*** = ***PAGE\_FOLDER*** + "/ResultView.jsp";

**public** String ***ROLE\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/RoleListView.jsp";

**public** String ***SUBJECT\_VIEW***=***PAGE\_FOLDER***+"/SubjectView.jsp";

**public** String ***SUBJECT\_LIST\_VIEW***=***PAGE\_FOLDER***+"/SubjectListView.jsp";

**public** String ***EXAM\_VIEW*** = ***PAGE\_FOLDER*** + "/ExamView.jsp";

**public** String ***RESULT\_LIST\_VIEW*** = ***PAGE\_FOLDER*** + "/ResultListView.jsp";

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

**public** String ***WELCOME\_VIEW*** = ***PAGE\_FOLDER*** + "/Welcome.jsp";

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

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

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

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

**public** String ***MARKSHEET\_CTL*** = ***APP\_CONTEXT*** + "/ctl/MarksheetCtl";

**public** String ***MARKSHEET\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/MarksheetListCtl";

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

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

**public** String ***QUESTION\_CTL*** = ***APP\_CONTEXT*** + "/ctl/QuestionCtl";

**public** String ***QUESTION\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/QuestionListCtl";

**public** String ***STARTEXAM\_CTL*** = ***APP\_CONTEXT*** + "/ctl/StartExamCtl";

**public** String ***TIMETABLE\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/TimetableListCtl";

**public** String ***STUDENT\_CTL*** = ***APP\_CONTEXT*** + "/ctl/StudentCtl";

**public** String ***STUDENT\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/StudentListCtl";

**public** String ***FACULTY\_CTL*** = ***APP\_CONTEXT*** + "/ctl/FacultyCtl";

**public** String ***FACULTY\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/FacultyListCtl";

**public** String ***ROLE\_CTL*** = ***APP\_CONTEXT*** + "/ctl/RoleCtl";

**public** String ***ROLE\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/RoleListCtl";

**public** String ***SUBJECT\_CTL***=***APP\_CONTEXT***+"/ctl/SubjectCtl";

**public** String ***SUBJECT\_LIST\_CTL***=***APP\_CONTEXT***+"/ctl/SubjectListCtl";

**public** String ***EXAM\_CTL*** = ***APP\_CONTEXT*** + "/ctl/ExamCtl";

**public** String ***RESULT\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/ResultListCtl";

**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";

**public** String ***MARKSHEET\_MERIT\_LIST\_CTL*** = ***APP\_CONTEXT*** + "/ctl/MarksheetMeritListCtl";

}

=========================================== User beans====================

**package** in.com.online.exam.bean;

**import** java.util.Date;

**publicclass** UserBean **extends** BaseBean {

**private** String fName;

**private** String lName;

**private** String login;

**private** String password;

**private** String confirmPassword;

**private** String mobileNo;

**private** Date dob;

**private** String address;

**private** String gender;

**privatelong**role\_Id;

**private** String roleName;

**public** String getfName() {

**return**fName;

}

**publicvoid** setfName(String fName) {

**this**.fName = fName;

}

**public** String getlName() {

**return**lName;

}

**publicvoid** setlName(String lName) {

**this**.lName = lName;

}

**publiclong** getRole\_Id() {

**return**role\_Id;

}

**publicvoid** setRole\_Id(**long**role\_Id) {

**this**.role\_Id = role\_Id;

}

**public** String getRoleName() {

**return**roleName;

}

**publicvoid** setRoleName(String roleName) {

**this**.roleName = roleName;

}

**public** String getFName() {

**return**fName;

}

**publicvoid** setFName(String fName) {

**this**.fName = fName;

}

**public** String getLName() {

**return**lName;

}

**publicvoid** setLName(String lName) {

**this**.lName = lName;

}

**public** String getLogin() {

**return**login;

}

**publicvoid** setLogin(String login) {

**this**.login = login;

}

**public** String getPassword() {

**return**password;

}

**publicvoid** setPassword(String password) {

**this**.password = password;

}

**public** String getConfirmPassword() {

**return**confirmPassword;

}

**publicvoid** setConfirmPassword(String confirmPassword) {

**this**.confirmPassword = confirmPassword;

}

**public** String getMobileNo() {

**return**mobileNo;

}

**publicvoid** setMobileNo(String mobileNo) {

**this**.mobileNo = mobileNo;

}

**public** Date getDob() {

**return**dob;

}

**publicvoid** setDob(Date dob) {

**this**.dob = dob;

}

**public** String getAddress() {

**return**address;

}

**publicvoid** setAddress(String address) {

**this**.address = address;

}

**public** String getGender() {

**return**gender;

}

**publicvoid** setGender(String gender) {

**this**.gender = gender;

}

@Override

**public** String getKey() {

// **TODO** Auto-generated method stub

**return**id+"";

}

@Override

**public** String getValue() {

// **TODO** Auto-generated method stub

**return**fName;

}

**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.

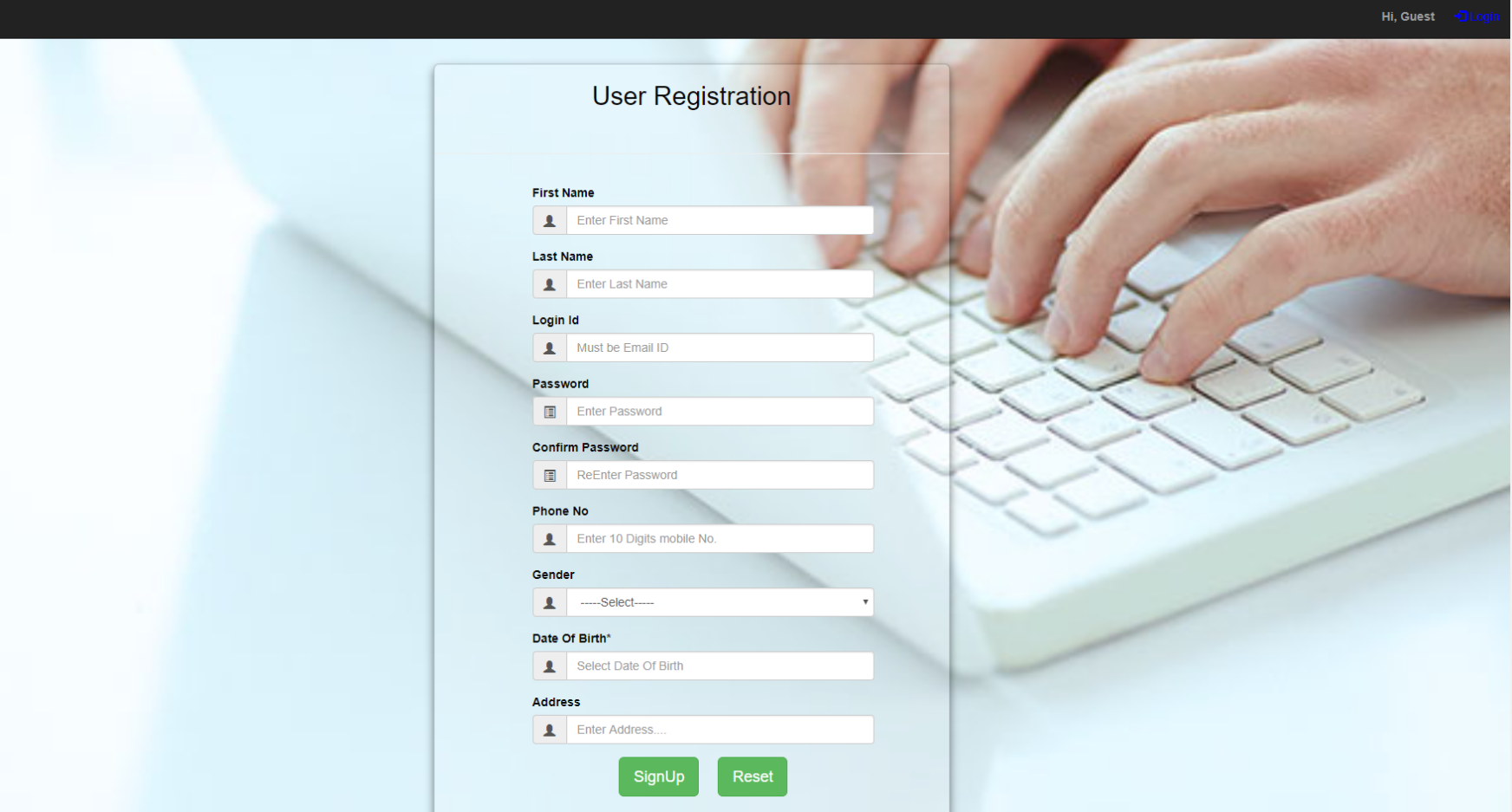
This method is named so because the software program, in the eyes of the tester, is like a white/ transparent box; inside which one clearly sees.

**Screen Snapshot**

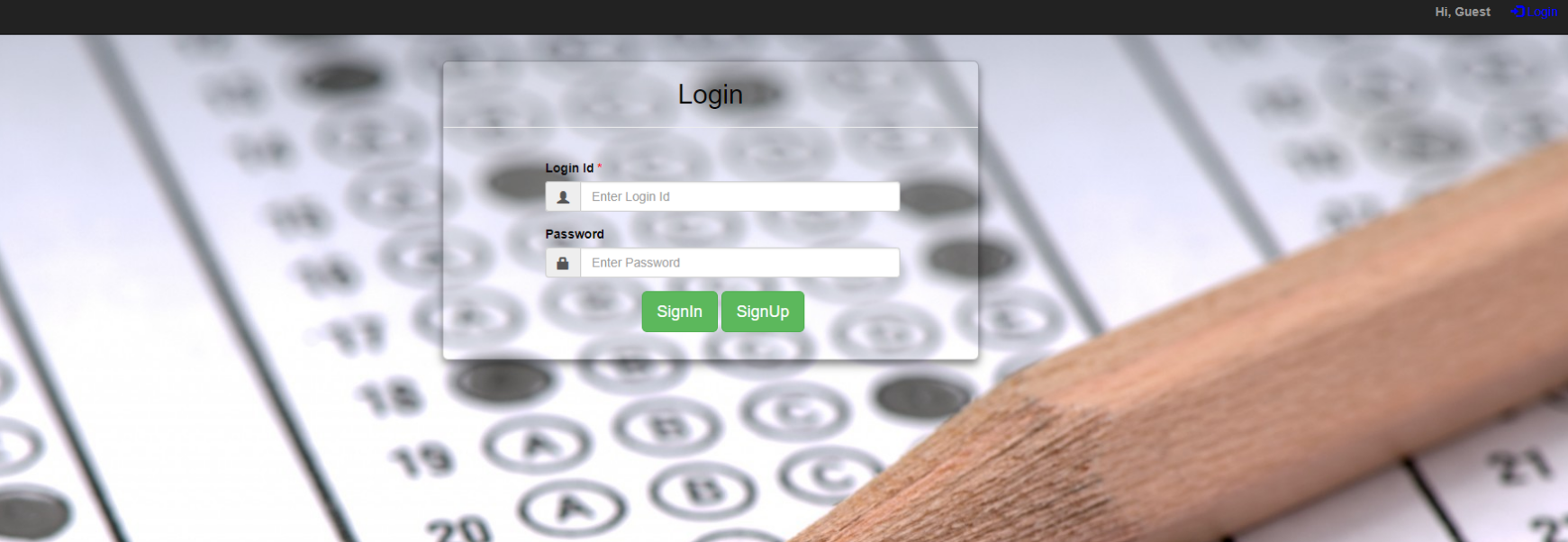
* 1. **Home Page**

****

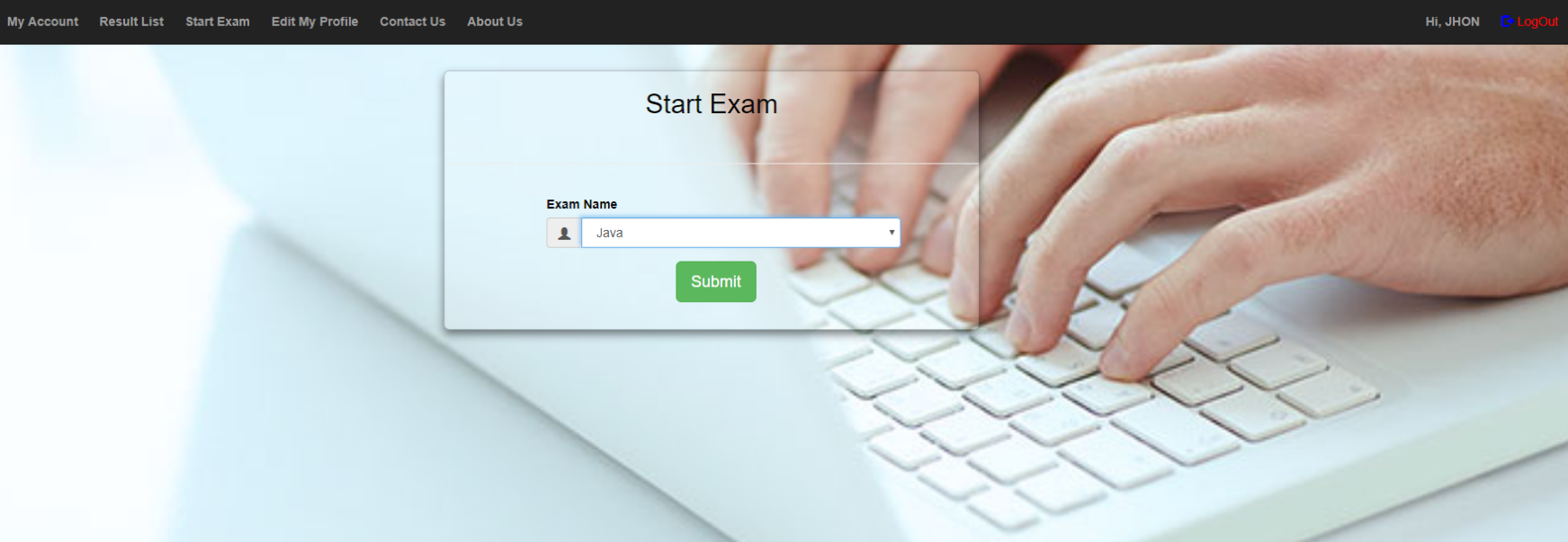
* 1. **SignUP**

****

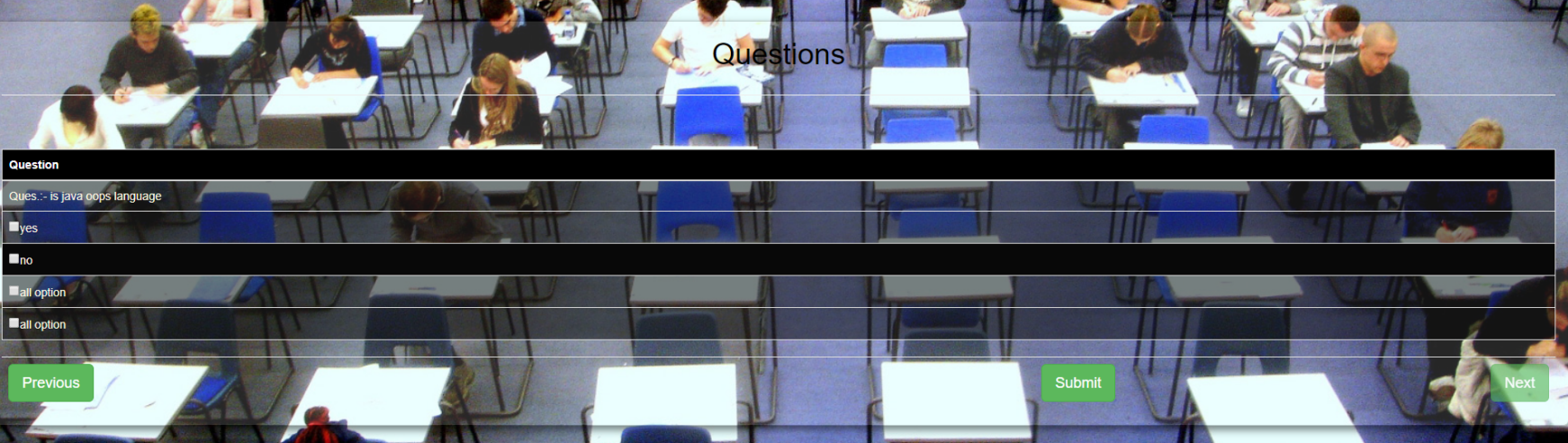
* 1. **Login Page**

****

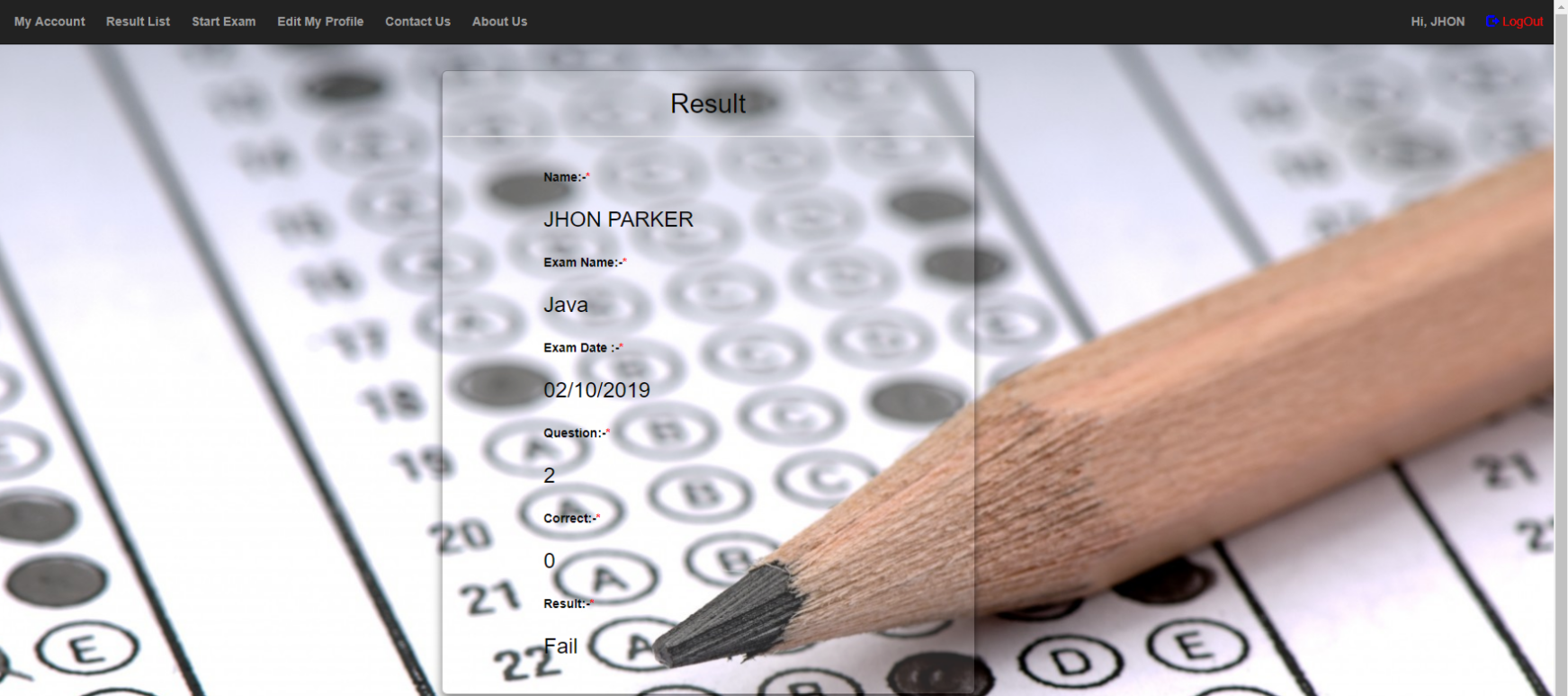
* 1. **Start Exam**

****

* 1. **Exam Pages(Questions)**

****

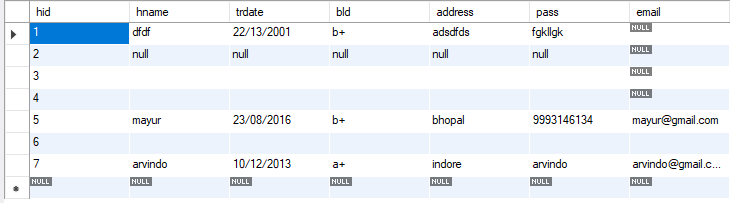
* 1. **Result**

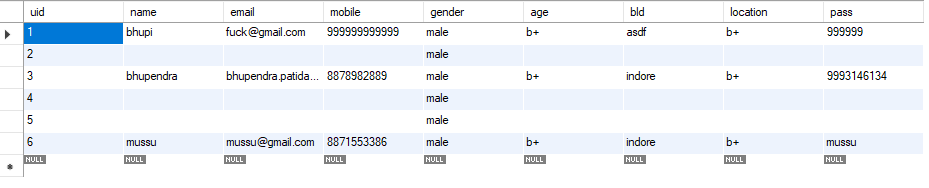
****

**Database**

**Registration database :**

**Insert database:**

****



**Delete database:**



**Limitations and Future Application of the Project**

**Futures Enhancement :**

* In future we can expand this project on the web.
* 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 Web Application provides facility to conduct online examination world wide. It saves time as it allows number of students to give the exam at a time and displays the results as the test gets over, so no need to wait for the result. It is automatically generated by theserver.

Administrator has a privilege to create, modify and delete the test papers and its particular questions. User can register, login and give the test with his specific id, and can see the results as well.

**Bibliography / References**

**Biblography/ Reference Sites :**

1. <https://www.onlineexambuilder.com/knowledge-center/exam-knowledge-center/what-is-online-examination/item10247>
2. <https://www.javatpoint.com/>