1. INTRODUCTION

1. INTRODUCTION:

Farming assistant web service is a web project to help the farmers working with the motive of greater profitability by direct communication between, farmer-to- supplier and farmer-to-farmer. This application gives suppose to the village farmers who want to use this facility and who want to learn how is it possible and how they can use e- farming to sell their products..

2. LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Existing System:

In the existing system buying and selling a product is done manually. Price of the product is fixed by the seller. All the details of the product to be sold or purchased is maintained manually. Sellers or buyers not able to get the complete information about the product.

2.2 Problems in Existing System:

- Previous data cannot be finding quickly.
- Less accuracy.
- Difficulty in management system.
- Lack of security.
- Due to human calculation error occurs.

2.3 Proposed System:

Farmers can buy or sell products directly to customers. Price of products is fixed by the farmers. The basic objective of the proposed system is to develop the complete computerized application that can satisfy all the needs of the system. To overcome the drawback in manual system so that the proposed system can prove the greater with a higher speed as compared to the manual task to develop a system which can keep track of distributed data related to a people.

- **Advantages of Proposed System:**

- This system makes selection process more effective.
- Reduce a lot of time and effort.
- Proposed system is used to reduce confusion at the time of processing .
- To increase efficiency proposed system is depend on classification method.

3. SYSTEM REQUIRMENT SPECIFICATION

3. System Requirement Specification

3.1 Purpose:

The main purpose of the project is developed to overcome all the problems in existing system. It helps to manage college data digitally and it also helpful to manage library details and virtual feeedbacks.

3.2 Scope:

E-Learning Platform is designed to help the admin to add organization faculty and users(students). E-Learning platform provides easy way to express the opinion of students towords staff using feedback. It provides an option to upload assignments and notes. This platform also provides the ability to edit or delete the students records. E-Learning Platform helps the staff to find student academic activities. Admin can declare any notice regarding college activities by using E-Learning Software.

3.3 Functional Requirements:

Admin:

- Admin can access the Officer and Farmer details.
- Admin can add the Officer
- Admin can see the Officer details.
- Admin can see the Farmer details.
- Admin can access the Officer and Farmer details.
- Admin can block the Officer and Farmer.

Officer:

- Officer well decide the weather Farmer is Eligible to give Queries or not.
- Officer can view Queries of individual Farmers.
- Officer can manage the Farmer details.
- Officer can replay the farmers queries.

Farmer:

- Only Farmer and Admin can login for the Queries page.
- Farmer can give their opinion through Queries.
- Only eligible Farmer can give the Queries.
- Only Admin can view the Farmer Queries But admin cannot have the permission to change the Queries.

3.4 Technology Requirements:

3.4.1 Software Requirements:

Operating System	:	Windows XP /Windows 7 /windows 8
IDE	:	Macromedia Dreamweaver
Designing Tools	:	HTML, CSS
Server Side Software	:	Java(J2EE/JSP)
Server	:	Apache Tomcat 5.5 and Wamp Server
Back End	:	MySQL
Web Browser	:	Google Chrome, Mozilla Firefox, Internet Explorer etc.

3.4.2 Hardware Requirements:

Processor : Pentium 4 or Above

RAM : 1GB or Above

HDD : Minimum 20 GB

4. DETAILED SYSTEM DESIGN

4. DETAILED SYSTEM DESIGN

4.1 High Level Design:

4.1.1 ER Diagram:

Introduction:

An entity–relationship model, a graphical representation of entities and their relationship to eachother,typically used in computing regardtothe organization of datawithin databasesor information systems. An entity is a piece of data-an object or concept about which the data is stored. A relationship between entities.

1) **ONE TO ONE**: one instance of an entity(A) is associated with the other entity

(b) for example employee name is associated with only one security number.

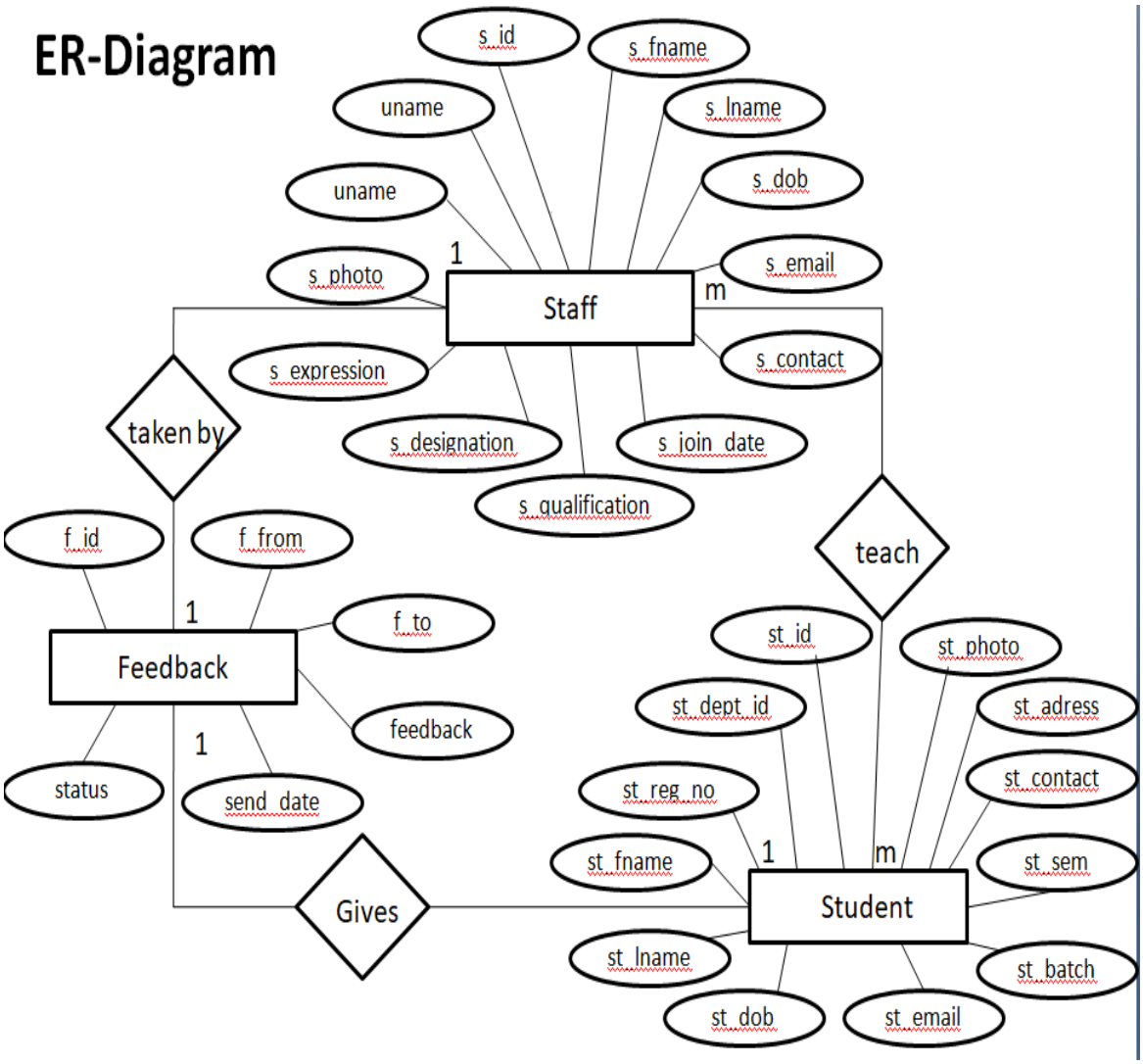
2) **ONE TO MANY**: one instance of an entity(A) is associated with zero, one or many instances of another entity(b), but for one instance of an entity B there is only one instance of an entity .A for example, for a company with all employees working in one building , the building name(A) is associated with many different employees(b) but those employee all share with same entity with A.

3) **MANY To MANY**: one instance of an entity (A) is associated with one, zero or many instance of an entity (B) and one instance of an entity B is associated with one zero work on multiple employee (A) associated with it.

The entity is a person, object , place or event for which data is collected for example if you consider the information system for a business, entities, would include not only customers, but the customers but the customers address, and order as well. The entity is represented by a rectangle and single labeled with a singular noun.

The relationship is the interaction between the entities would include only to customers, but the customers address, and well orders. The entity is represented by a diamond shape, by the line connecting the entities in either case verb used about basic relationships.

The cardinality defines the relationship between the entities the relationships. The cardinality defines the relationship between the entities in terms of numbers. An entity may be optional: for example, there must be at least one production.



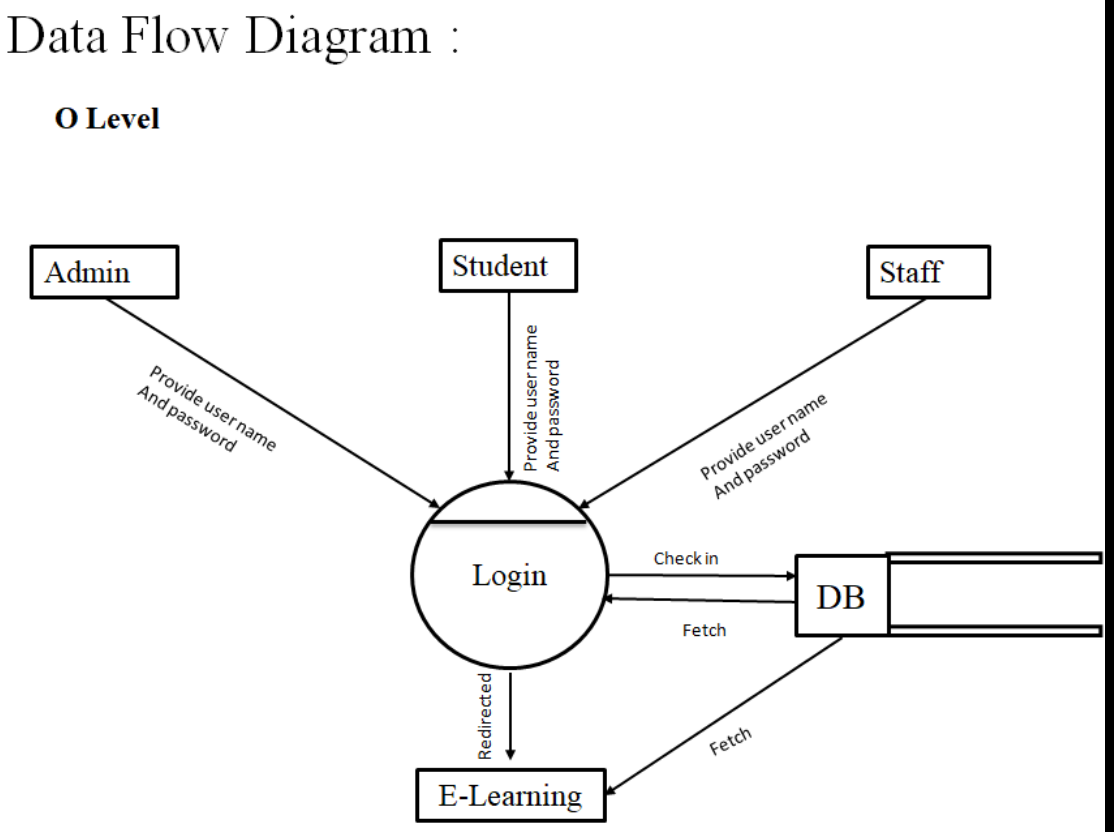
4.1.2 Data Flow Diagram:

A **data flow diagram** (**DFD**) is a graphical representation of the "**flow**" of **data** through an information system, modeling its process aspects. A**DFD** is

often used as a preliminary step to create an overview of the system without going into great detail, which **can** later be elaborated.

0 Level DFD:

A context **diagram** is a top **level** (also known as "**Level 0**")**data flow diagram**. It only contains one process node ("Process **0**") that generalizes the function of the entire system in relationship to external entities.

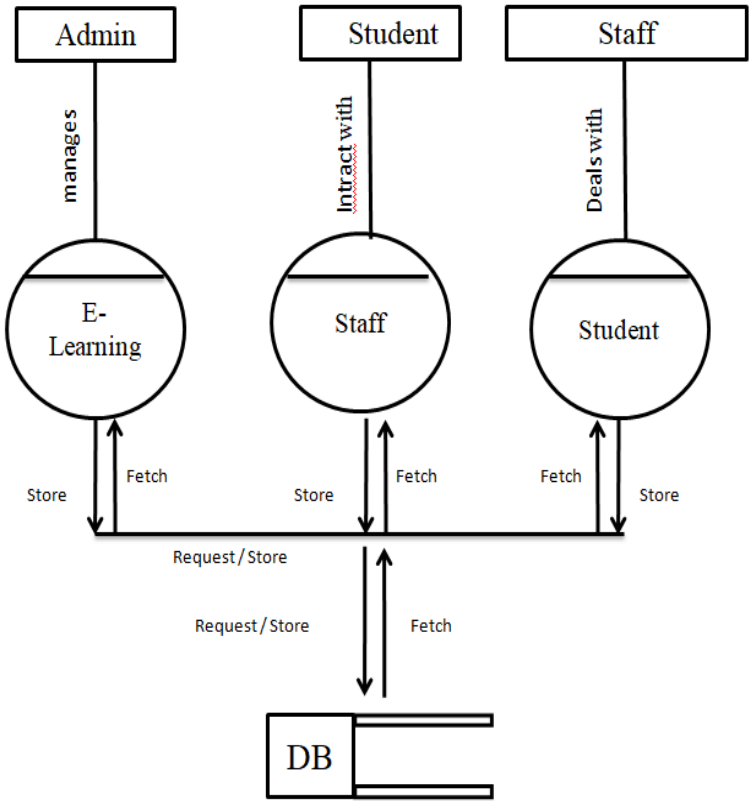


1 Level DFD:

The **Level 1 DFD** shows how the system is divided into sub-systems (processes), each of which deals with **one** or more of the data flows to or from

an external agent, and which together provide all of the functionality of the system as a whole.

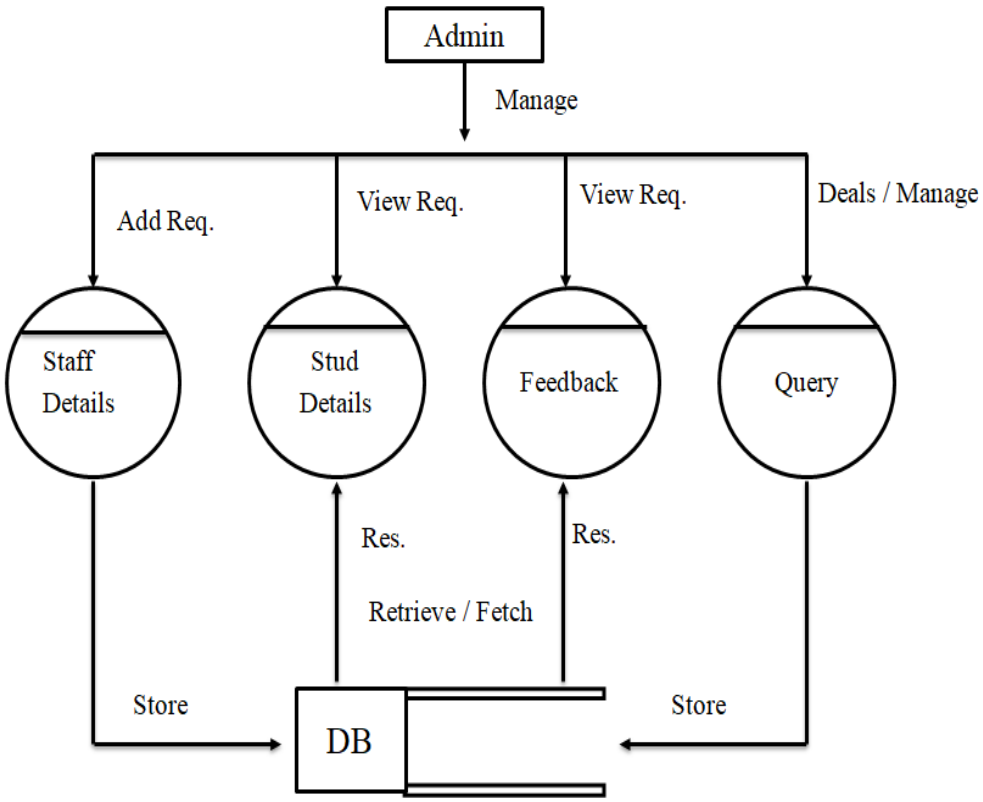
1 Level



2 Level DFD:

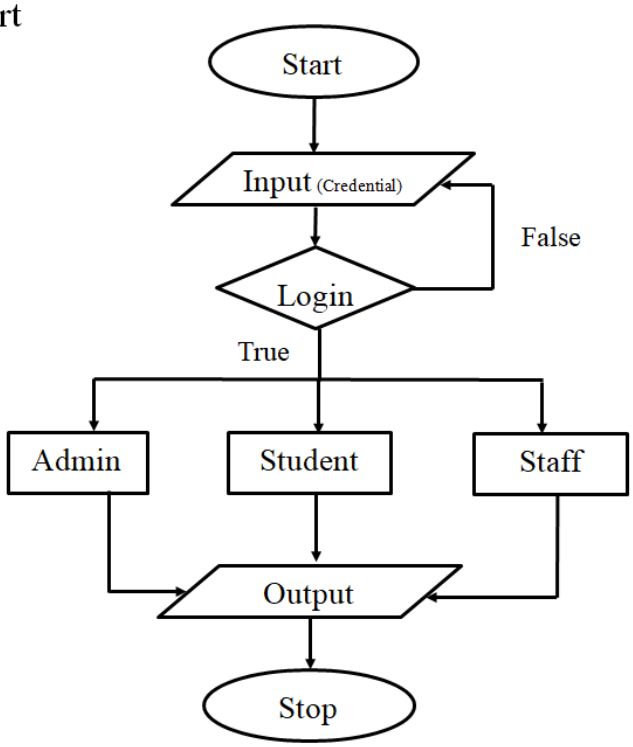
A DFD that represents a decomposed level 1 DFD is called a level 2 DFD. There can be a level 2 DFD for each process that appears in the level 1 DFD.

2 Level



4.2 Low Level Design:

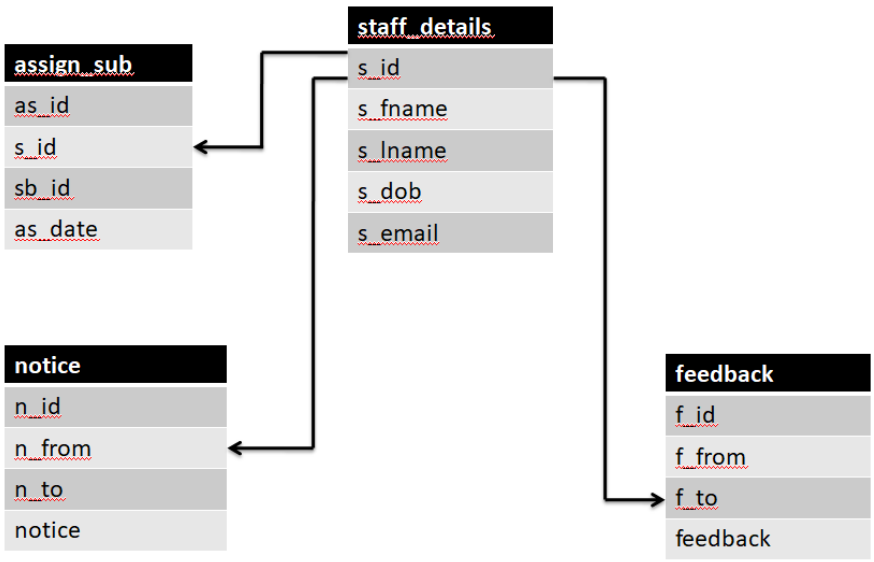
Flow chart



4.2.1 Relational Schema:

A set of attributes is called as relational schema. It is also known as table schema. It is the basic information describing a table or relation, it is a logical definition of table. Which include a set of column, the data type associated with list column.

Relational Schema



4.2.2 Sample Code:

Bean Program-

```
Package pkg_name;

import java.sql.*;

public class dbconnect

{

private Connection con;

public Statement stm;

public String getconn()

{

try

{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

cn=DriverManager.getConnection("jdbc:odbc:dsn");

st=cn.createStatement();

}

catch(Exception ex)

{

System.out.println(ex);

}

return "";

}

}
```

Student Details-

Form-

```
<!DOCTYPE html>
<html lang="en">
<head>
<jsp:include page="metatags.jsp"></jsp:include>
</head>
<body>
<jsp:include page="navbar.jsp"></jsp:include>
<div class="main-container container-fluid">
<jsp:include page="sidebar.jsp"></jsp:include>
<div class="main-content">
<div class="breadcrumbs" id="breadcrumbs">
<ul class="breadcrumb">
<li>
<i class="icon-home home-icon"></i>
<a href="#">Home</a>

<span class="divider">
<i class="icon-angle-right arrow-icon"></i>
</span>
</li>
<li class="active">Dashboard</li>
</ul><!--.breadcrumb-->

<div class="nav-search" id="nav-search">
</div><!--#nav-search-->
</div>

<div class="page-content">
<div class="page-header position-relative">
```

```
<h1>

STUDENTS DETAILS


</h1>

</div><div class="row-fluid">

<div class="span12">

<!--PAGE CONTENT BEGINS-->


 <%@page import="java.sql.*"%>

<%@page import="d_edu.dbconnect"%>

<jsp:useBean id="s" class="d_edu.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<jsp:include page="val.jsp"></jsp:include>

<form id="WFS_Riz" name="form1" method="post"
action="stud_insert.jsp">

<table width="414" height="407" border="0" align="left">

<tr>

<td colspan="2"><div align="center"><strong>STUDENT DETAILS
</strong></div></td>

</tr>

<tr>

<td width="186">Department ID </td>

<td width="309"><select class="validate[required]" name="sdi"
id="sdi">

<option value="">--Select department--</option>

<%

ResultSet rs=s.stm.executeQuery("select * from dept_details");

while(rs.next())

%>

<option
value="<%=rs.getInt("dept_id")%>"><%=rs.getString("dept_name")%
></option>

<%

}
```

```
%>

</select></td>

</tr>

<tr>

<td>Regstration Number </td>

<td><input class="validate[required]" name="srn" type="text"
id="srn"></td>

</tr>

<tr>

<td>First Name </td>

<td><input class="validate[required,custom[onlyLetter]]"
name="sf" type="text" id="sf"></td>

</tr>

<tr>

<td>Last Name </td>

<td><input class="validate[required,custom[onlyLetter]]"
name="sl" type="text" id="sl"></td>

</tr>

<tr>

<td>Date of Birth </td>

<td><input class="validate[required,custom[date]]" name="sdob"
type="date" id="sdob"></td>

</tr>

<tr>

<td>Email-id</td>

<td><input class="validate[required,custom[email]]" name="se"
type="text" id="se"></td>

</tr>

<tr>

<td>Contact Number </td>

<td><input class="validate[required,custom[mobile]]" name="sc"
type="text" id="sc"></td>

</tr>

<tr>
```

```
<td>Semester</td>

<td><input class="validate[required]" name="ss" type="text"
id="ss"></td>

</tr>

<tr>

<td>Batch</td>

<td><input class="validate[required,custom[onlyNumber]]"
name="sb" type="text" id="sb"></td>

</tr>

<tr>

<td>Photo</td>

<td><input class="validate[required]" name="sp" type="text"
id="sp"></td>

</tr>

<tr>

<td height="29">Address</td>

<td><input class="validate[required]" name="sa" type="text"
id="sa" value=""></td>

</tr>

<tr>

<td height="31" colspan="2"><div align="center">

<input type="submit" class="btn btn-primary" name="Submit"
value="Submit">

<input type="reset" class="btn btn-default" name="Reset"
value="Reset">

</div></td>

</tr>

</table>

</form>



<!--PAGE CONTENT ENDS-->

</div>

<!--/.span-->

</div><!--/.row-fluid-->
```

```
</div><!--/.page-content-->

<jsp:include page="settings.jsp"></jsp:include>

</div><!--/.main-content-->

</div><!--/.main-container-->

<jsp:include page="scripts.jsp"></jsp:include>

</body>

</html>

</body>

</html>
```

Insert-

```
<%@page import="java.sql.*"%>

<%@page import="d_edu.dbconnect"%>

<jsp:useBean id="s" class="d_edu.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%


String b=request.getParameter("sdi");

String c=request.getParameter("srn");

String d=request.getParameter("sf");

String e=request.getParameter("sl");

String f=request.getParameter("sdob");

String g=request.getParameter("se");

String h=request.getParameter("sc");

String i=request.getParameter("ss");

String j=request.getParameter("sb");

String k=request.getParameter("sp");

String l=request.getParameter("sa");
```

```
int z=s.stm.executeUpdate("insert into stud_details
values(null,'"+b+"','"+c+"','"+d+"','"+e+"','"+f+"','"+g+"','"+
h+"','"+i+"','"+j+"','"+k+"','"+l+"')");


%>

<script>

alert("values inserted");

document.location="stud_view.jsp";

</script>
```

Delete-

```
<%@page import="java.sql.*"%>

<%@page import="d_edu.dbconnect"%>

<jsp:useBean id="s" class="d_edu.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%

String a=request.getParameter("id");


int z=s.stm.executeUpdate("delete from stud_details where
st_id='"+a+"' ");


%>

<script>

alert("values deleted");

document.location="stud_view.jsp";

</script>
```

View-

```
<!-- Header Design Start -->
<!DOCTYPE html>
<html lang="en">
<head>
<jsp:include page="metatags.jsp"></jsp:include>
<style type="text/css">
<!--
.style1 {color: #FFFFFF}
.style2 {color: #FFFFFF}
-->
</style>
</head>

<body>
<jsp:include page="navbar.jsp"></jsp:include>



<div class="main-container container-fluid">


<jsp:include page="sidebar.jsp"></jsp:include>
<div class="main-content">
<div class="breadcrumbs" id="breadcrumbs">
<ul class="breadcrumb">
<li>
<i class="icon-home home-icon"></i>
<a href="#">Home</a>


<span class="divider">
<i class="icon-angle-right arrow-icon"></i></span></li>
<li class="active">Student Details</li>
</ul><!--.breadcrumb-->
```

```
<div class="nav-search" id="nav-search">


</div><!--#nav-search-->

</div>


<div class="page-content">


<div class="row-fluid">

<div class="span12">

<!--PAGE CONTENT BEGINS-->




<div class="hr hr-18 dotted hr-double"></div>


<h4 class="pink">

<!-- <i class="icon-hand-right icon-animated-hand-pointer
blue"></i> -->

<h3>Student Details | <a class="btn btn-primary"
href="stud_form.jsp" role="button" class="green" data-
toggle="modal"><i class="icon-plus"></i> Add Student </a></h3>

</h4>


<div class="hr hr-18 dotted hr-double style1"></div>


<div class="row-fluid">


<div class="table-header">

<div class="style2">Student Details</div>

</div>

<table id="sample-table-22" class="table table-striped table-
bordered table-hover">

<thead><!-- Header Design End -->


<tr>
```

```
<th width="44" height="48"><div align="center">ID</div></th>

<th width="106"><div align="center">Department ID</div></th>

<th width="92"><div align="center">Reg. Number </div></th>

<th width="117"><div align="center">First Name </div></th>

<th width="109"><div align="center">Last Name </div></th>

<th width="103"><div align="center">Date of Birth </div></th>

<th width="133"><div align="center">Email-id</div></th>

<th width="102"><div align="center">Contact Number </div></th>

<th width="88"><div align="center">Semester</div></th>

<th width="69"><div align="center">Batch</div></th>

<th width="84"><div align="center">Photo</div></th>

<th width="79"><div align="center">Address</div></th>

<th width="160"><div align="center">Action</div></th>

</tr>

</thead>

 <%@page import="java.sql.*"%>

<%@page import="d_edu.dbconnect"%>

<jsp:useBean id="s" class="d_edu.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%


ResultSet rs=s.stm.executeQuery("select * from stud_details
std,dept_details dd where std.st_dept_id=dd.dept_id");

while(rs.next())

{

String sid=rs.getString("st_id");


%>


<tr>

<td height="26"><div align="center"><%=sid%></div></td>

<td><div
align="center"><%=rs.getString("dept_name")%></div></td>
```

```
<td><div
align="center"><%=rs.getString("st_reg_no")%></div></td>

<td><div
align="center"><%=rs.getString("st_fname")%></div></td>

<td><div
align="center"><%=rs.getString("st_lname")%></div></td>

<td><div align="center"><%=rs.getString("st_dob")%></div></td>

<td><div
align="center"><%=rs.getString("st_email")%></div></td>

<td><div
align="center"><%=rs.getString("st_contact")%></div></td>

<td><div align="center"><%=rs.getString("st_sem")%></div></td>

<td><div
align="center"><%=rs.getString("st_batch")%></div></td>

<td><div
align="center"><%=rs.getString("st_photo")%></div></td>

<td><div
align="center"><%=rs.getString("st_address")%></div></td>

<td><div align="center"><a title="Edit"
href="stud_edit.jsp?id=<%=sid%>" class="btn btn-warning"><i
class="icon-edit"></i></a> | <a title="Delete" onClick="return
confirm('Are you sure?')" href="stud_delete.jsp?id=<%=sid%>"
class="btn btn-danger"><i class="icon-trash"></i></a>
</div></td>

</tr>

<%}%>


 <!-- Footer Design Start -->

 </tbody>


</table>




</span></div>

</div>


</div>
```

```
<!--PAGE CONTENT ENDS-->

</div><!--/.span-->

</div><!--/.row-fluid-->

</div><!--/.page-content-->



</div><!--/.main-content-->

</div><!--/.main-container-->

<jsp:include page="scripts.jsp"></jsp:include>


</body>

</html>

<!-- Footer Design End -->
```

Update-

```
<%@page import="java.sql.*"%>

<%@page import="d_edu.dbconnect"%>

<jsp:useBean id="s" class="d_edu.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%

String a=request.getParameter("sid");

String b=request.getParameter("sdi");

String c=request.getParameter("srn");

String d=request.getParameter("sf");

String e=request.getParameter("sl");

String f=request.getParameter("sdob");

String g=request.getParameter("se");

String h=request.getParameter("sc");

String i=request.getParameter("ss");
```

```
String j=request.getParameter("sb");

String k=request.getParameter("sp");

String l=request.getParameter("sa");


int z=s.stm.executeUpdate("update stud_details set
st_dept_id='"+b+"',st_reg_no='"+c+"',st_fname='"+d+"',st_lname=
'"+e+"',st_dob='"+f+"',st_email='"+g+"',st_contact='"+h+"',st_s
em='"+i+"',st_batch='"+j+"',st_photo='"+k+"',st_address='"+l+"'
where st_id='"+a+"' ");


%>

<script>

alert("values updated");

document.location="stud_view.jsp";

</script>
```

Validation-

```
<link rel="stylesheet" href="js/validationEngine.jquery.css"
type="text/css" media="screen" title="no title" charset="utf-8"
/>

<script src="js/jquery.min.js" type="text/javascript"></script>

<script src="js/jquery.validationEngine-en.js"
type="text/javascript"></script>

<script src="js/jquery.validationEngine.js"
type="text/javascript"></script>

<script>

$(document).ready(function() {

$("#formID").validationEngine()

});

</script>
```

4.2.3 Table Structure:

Login-

5. TOOLS USED

5. Tools Used:

Java:

Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems(which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1991 and first released in 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (byte code compiler), GNU Class path (standard libraries), and Iced Tea-Web (browser plugin for applets).

Advantages of java:

- Purely Object oriented.

- Platform independent.
- It is dynamic, simple and robust.
- Easy to learn.
- Multithreaded.
- Secure.
- Wide variety of Application Programmer Interfaces (APIs).
- Excellent networking capability.

The java Platform:

The Java platform is the name given to the computing platform from Oracle that helps users to run and develop Java applications. The platform does not just enable a user to run and develop Java application, but also features a wide variety of tools that can help developers work efficiently with the Java programming language.

The platform consists of two essential software's:

- The Java Runtime Environment (JRE), which is needed to run Java applications and applets.
- The Java Development Kit (JDK), which is needed to develop those Java applications and applets. If you have installed the JDK, you should know that it comes equipped with a JRE as well. So, for all the purposes of this book, you would only require the JDK.

Java Components:

Java has two components those are

1. Java virtual machine (JVM).

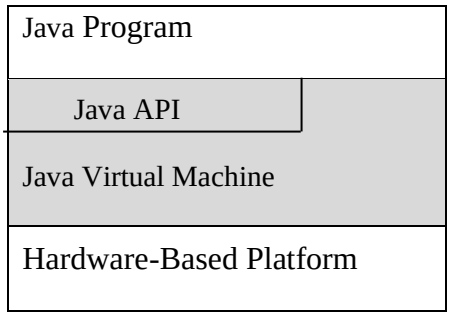
2. Java Application Programmers Interface (API).

JVM-

A Java virtual machine (JVM) interprets compiled Java binary code (called byte code) for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform.

API-

An application programming interface (API) is a library of functions that Javaprovides for programmers for common tasks like file transfer, networking, anddata structures.



Java in web:

Java covers the whole application form server to client and back again, it provides many powerful technologies, it can be used to extend the browser, and it provides good security system.

HTML:

HTML stands for hypertext markup Language. It is very useful to make web pages and very easy to learn. Hypertext Markup file is a text file containing small markup tags. These marks up tags tell the browser how to display a web

page. It has two types of extensions one is .html and second is .html but both are used for html web pages. For hypertext markup language you can use the simple text editor for example; use notepad for writing your HTML code in the windows. If you are using Mac you can use simple text editor.

HTML uses approach of what you see is what you get. You can also use to write tags other software that is FrontPage and Dreamweaver. In HTML character are surrounded by the tags. HTML tags come in pair. The beauty of this language is that it is not case sensitive. Every web page need HTML with it you cannot make the good web pages. And it is the base for every web page and used to display the text in the web pages there are some other latest version of HTML like DHTML which stands for dynamic html and is used to make the web pages more interactive

Features of HTML:

- I is simple to understand and implement.
- HTML constructs are very easy to comprehend, and can be used effectively by anybody.
- HTML syntax is a worldwide standard.
- The methodology used by HTML to markup information is independent of its representation on a particular hardware or software architecture.
- It is not a programming language.
- And it is also not a description language.

Java Server Pages (JSP):

Java Server Pages (JSP) lets you separate the dynamic part of your pages from the static HTML. We simply write the regular html in the normal manner, using whatever Web-page-building tools you normally use. We then enclose

the code for the c parts in special tags, most of which start with "<%" and end with "%>".

We normally give your file a .jsp extension, and typically install it in any ace you could place a normal Web page. Although what you write often looks more a regular html file than a servlet, behind the scenes, the JSP page just gets converted to a normal servlet, with the static html simply being printed to the output stream associated with the servlet's service method.

This is normally done the first time the page is requested, and developers can simply request the page themselves when first installing it if they want to be sure that the first real user doesn't get a momentary' delay when the JSP page is translated to a servlet and the servlet is compiled and loaded. Many Web servers let you define aliases that so that a URL that appears to reference an html file really points to a servlet or JSP page.

Advantages of jsp:

- Separation of static from dynamic content.
- Write Once Run Anywhere.
- Recommended Web access layer for n-tier architecture.
- Completely leverages the Servlet API.
- Platform independent.
- Reuse of components and tag libraries.
- Encapsulation of functionality.
- They have a better performance and scalability than ordinary CGI scripts, because they are persistent in memory and multi-threaded.
- They have built in support for HTTP sessions, which makes application Programming possible.
- They have full access to Java Technology-Network awareness, threads and Database connectivity-without the limitations of client side application applets.
- They are automatically recompiled when necessary.
- They exist in the ordinary Web server document space, no special URL mapping is required to address them.

How jsp works?

JSP pages exist in 3 forms or versions-

- JSP source code consists of text file with an extension of .jsp and contains a mix of HTML template code, Java language statements and JSP directives and actions that describe how to generate a web page to service a particular request.
- Java source code: the jsp container translates the jsp source code into the source code for an equivalent Java Servlet as needed..
- Compiled Java class: Like any other Java class, the generated servlet code is compiled into byte-codes in a .class file ready to be loaded and executed.

What is Java Script?

- Java Script is embedded into html.
- It is browser dependent.
- JavaScript depends on the web browser to support it. If the browser doesn't support it, JavaScript code will be ignored. Internet Explorer 3.0 and Netscape Navigator 2.0 onwards support JavaScript.
- It is an interpreted language, loosely typed, object based language.
- Java script is not Java.

Data Validation:

JavaScript provides the means for basic data validation before it is sent to the server. Whether the values entered are correct or not or whether all the fields in a form are filled out or not can be checked before sending data to web server, if JavaScript is not used t hen data is sent to web server, and the web server would response with a message that the data sent to it is incorrect or incomplete Thus JavaScript ensures data validation and also reduces the network traffic.

MySQL:

MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications.

This tutorial will give you quick start with MySQL and make you comfortable with MySQL programming.

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL is an ANSI (American National Standards Institute) standard

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Using SQL in Your Web Site

To build a web site that shows data from a database, you will need:

- An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
- To use a server-side scripting language, like PHP or JSP
- Use SQL to get the data you want
- To use HTML , JSP/ CSS

RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables.

A table is a collection of related data entries and it consists of columns and rows.

Some of the Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

Java Data Base Connectivity (JDBC):-

In an enterprise computing which is largely the black art of managing huge databases? People associated with the enterprise need to be able to use and update the data easily, quickly and securely. The powerful Java Data Base Connectivity (JDBC) suit the java.sql.* package realizes java's promise as a serious business programming tools.

Java Data Base Connectivity is a standard SQL database access interface providing uniform access to a wide range of relational databases. It also provides a common base on which higher level tools and inter faces can be built. This comes with an "ODBCBridge". That bridge is a library, which implements JDBC in terms of ODBC standard API.

There are many types of drivers used in connecting such as Native API Partly Java driver, a net protocol all java driver. The driver used here is JDBC-ODBC Bridge.

JDBC-ODBC bridge plus ODBC driver –This is the crudest possible solution. Applets access data base using a combination of the JDBC-ODBCBridge and an ODBC driver. This requires both drivers to be installed on the user's computer- A very cumbersome solution for both internet and intranet users.

JDBCTM is a Java Tm API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for "Java Database Connectivity".) It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. In other words, with the JDBC API, it isn't necessary to write one program to access a Sybase database, another program to access an MySQL database, another program to access an Informix database, and so on. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. And, with an

application written in the Java programming language, one also doesn't have to worry about writing different applications to run on different platforms. The combination of Java and JDBC lets a programmer write it once and run it anywhere.

Java being robust, secure, easy to use, easy to understand, and automatically downloadable on a network, is an excellent language basis for database applications.

What is needed is a way for Java applications to talk to a variety of different databases. JDBC is the mechanism for doing this.JDBC extends what can be done in Java. For example, with Java and the JDBC API, it is possible to publish a web page containing an applet that uses information obtained from a remote database. Or an enterprise can use JDBC to connect all its employees (even if they are using a conglomeration of Windows, Macintosh, and UNIX machines) to one or more internal databases via an intranet. With more and more programmers using the Java programming language, the need for easy database access from Java is continuing to grow.

MIS managers like the combination of Java and JDBC because it makes disseminating information easy and economical. Businesses can continue to use their installed databases and access information easily even if it is stored on different database management systems. Development time for new applications is short. Installation and version control are greatly simplified. A programmer can write an application or an update once, put it on the server, and everybody has access to the latest version. And for businesses selling information services, Java and JDBC offer a better way of getting out information updates to external customers.

JDBC does the following things:

- Establish a connection with a database
- Send SQL statements
- Process the results.

The following code fragment gives a basic example of these three steps:

```
Class.forName ("sun. jdbc.odbc.JdbcOdbcDriver");

Connection con=DriverManager.getConnection("Jdbc:Odbc:dsnname");

Statement stmt=con.createStatement();
```

Connection:-

A connection object represents a connection with a database. A connection session includes the SQL statements that are executed and the results that are returned over the connection. A single application can have one or more connections with a single database, or it can have connections with many different databases.

Opening a Connection:

The standard way to establish a connection with a database is to call the method DriverManager.getConnection. This method takes a string containing a URL. The Driver Manager class, referred to a the JDBC management layer, attempts to locate a driver than can connect to the database represented Driver classes, and when the method get Connection is called, it checks with each driver in the list until it finds one that can connect uses this URL to actually establish the connection.

Sending Statement:

Once a connection is established, it is used to pass SQL statements to its underlying database. JDBC does not put any restrictions on the kinds of SQL statements that can be sent; this provides a great deal of flexibility,

allowing the use of database-specific statements or even Non-SQL statements. It requires, however, that the user be responsible for making sure that the underlying database can process the SQL statements being sent and suffer the consequences if it cannot.

Driver Manager:

The Driver Manager class is the management layer of JDBC, working between the user and the drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It addition, the driver manager class attends to things like driver login time limits and the printing of log and tracing messages. The only method in this class that a general programmer needs to use directly is DriverManager.getConnection. As its name implies, this method establishes a connection to a database.

Why we need JDBC?

- ODBC is not appropriate for direct use from Java because it uses a C interfaces.
- ODBC is hard to learn. It mixes simple and advanced features together, and it has
- Complex options even for simple queries.
- A Java API like JDBC is needed in order to enable a “Pure Java “solution.
- When ODBC is used, the ODBC driver manager and drivers must be manually
- Installed on every client machine.

SESSION:

This is the ***HttpSession*** object associated with the request. Recall that sessions are created automatically, so this variable is bound even if there was no incoming session reference. The one exception is if you use the ***session*** attribute of the page directive to turn sessions off, in which case attempts to reference the session variable cause errors at the time the JSP page is translated into a servlet.

Apache Tomcat:

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Tomcat is a web server that supports servlets and JSPs.

Tomcat comes with the Jasper compiler that compiles JSPs into servlets. Tomcat is available for commercial use under the ASF license from the Apache web site in both binary and source versions.

Apache Tomcat is developed in an open and participatory environment and released under the Apache Software License. Apache Tomcat powers numerous large scale, mission-critical web applications across a diverse range of industries and organizations. Different versions of Apache Tomcat are available for different versions of the Servlet and JSP specifications.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high availability environments. Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM

Usage:

Java Web Applications, Java Mobile Applications using J2ME.

Access:

Create a Java Web Application or Mobile Application using Net Beans Build and run project, this will automatically launch the Apache Tomcat as default.

Viewing Web Applications:

http://localhost: 8080/project_name/

Directory Structure:

The typical and default directory hierarchy of a Tomcat installation comprises the following:

- bin - startup, shutdown and other scripts and executable.
- common - common classes that Catalina and web applications can use.
- conf - XML files and related DTDs to configure Tomcat.
- logs - Catalina and application logs.
- server - classes used only by Catalina.
- shared - classes shared by all web applications.
- webapps - directory containing the web applications.
- work - temporary storage for files and directories.

A web application is basically a web site that:

- "Knows who you are"--it doesn't just give you static pages, it interacts with you.
- Can permanently change data (such as in a database).

- A web application can consist of multiple pieces.
- Static web pages (possibly containing forms).
- Servlets.
- JSP.

Tomcat organizes all these parts into a single directory structure for each web application.

The flow that takes place is:

- The user submits an HTML form.
- Tomcat finds the servlet based on the URL and the deployment descriptor.
- (web.xml) and passes the request to the servlet
- The servlet computes a response.
- The servlet writes an HTML page containing the response.
- The servlet forwards the response to the JSP.
- The JSP embeds the response in an HTML page
- Tomcat returns the HTML page to the user.

Status:

Tomcat is available at the Jakarta binary downloads page. The Tomcat server is a Java based Web Application container that was created to run Servlets and Java Server Pages (JSP) in Web applications. As part of Apache's open source Jakarta project, it has nearly become the industry accepted standard reference.

Implementation for both the Servlets and JSP API. Written by expert Servlets and JSP software architect and author James Goodwill, this column will feature introductory Web application development issues, Tomcat installation and configuration, deploying Web applications onto Tomcat, Struts and much more.

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, providing an environment f or Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets. The Tomcat servlet engine is often used in combination with an Apache webserver or other web servers. Tomcat can also function as an independent web server.

Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high availability environments. Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM

Beans:

It's used to achieve reusability in java it is introduced to overcome the drawbacks of the traditional inheritance and object relations in JavaBean. The reusability is achieved using bean class. Bean is a component equivalent to ActiveX component of visual basic. To create beans we must follow design pattern rules

- The class must be placed inside the package
- Class must be defined as public
- Variables in the class are defined as private to avoid direct accessibility of variables
- Variables must be defined in lower case
- A bean class can have public variable also, for every private variable there must be Set and Get methods. Set takes parameter which is used to assign the values for the variable. Get is used to retrieve the values. Every bean class should have implicit constructor (default constructor).

6. IMPLEMENTATION

6. Implementation:

6.1 Introduction:

The goal of the coding phase is to translate the design into code in the given programing language. The coding steps translate the detailed design of the system into programming language. The translation process continues when complier accepts source code as input and produces machine dependent object code as output. Linking of object files are done to produce the machine code.

Internal documentation is another important factor, to facilitate other to understand the code and the logic.

Code review and walk through:

Both reviews and walk through used to deliver the correct codes. The code review is done as soon as code is ready to be executed, this is to reduce syntax errors and also check the coding standard.

Module Specification:

The modules specified in the design are implemented using various ".html", ".jsp" and ".class" files. These files in the source code shares the common routines and share the data structure, to establish the hierarchical relationship.

Compilation and Building the executable:

The source code for the system organized in various files is compiled using the "java" utility provided in the JAVA. The application is made to run in web browser the address as "http://localhost:8080/project_name" present in ROOT directory of Tomcat Server.

6.2 Running the package:

The following steps are undertaken to execute this application-

- Frist start Tomcat Server.
- Open any Web Browser like Google Chrome, Mozilla Firefox.
- Type the following address in address bar of Web browser.
- http://localhost:8080/project_name
- Now the Home page of application is displayed on screen. If new registration is there then register or sing up first and use the application. If already registered then directly login into the system.

6.3 Methodology:

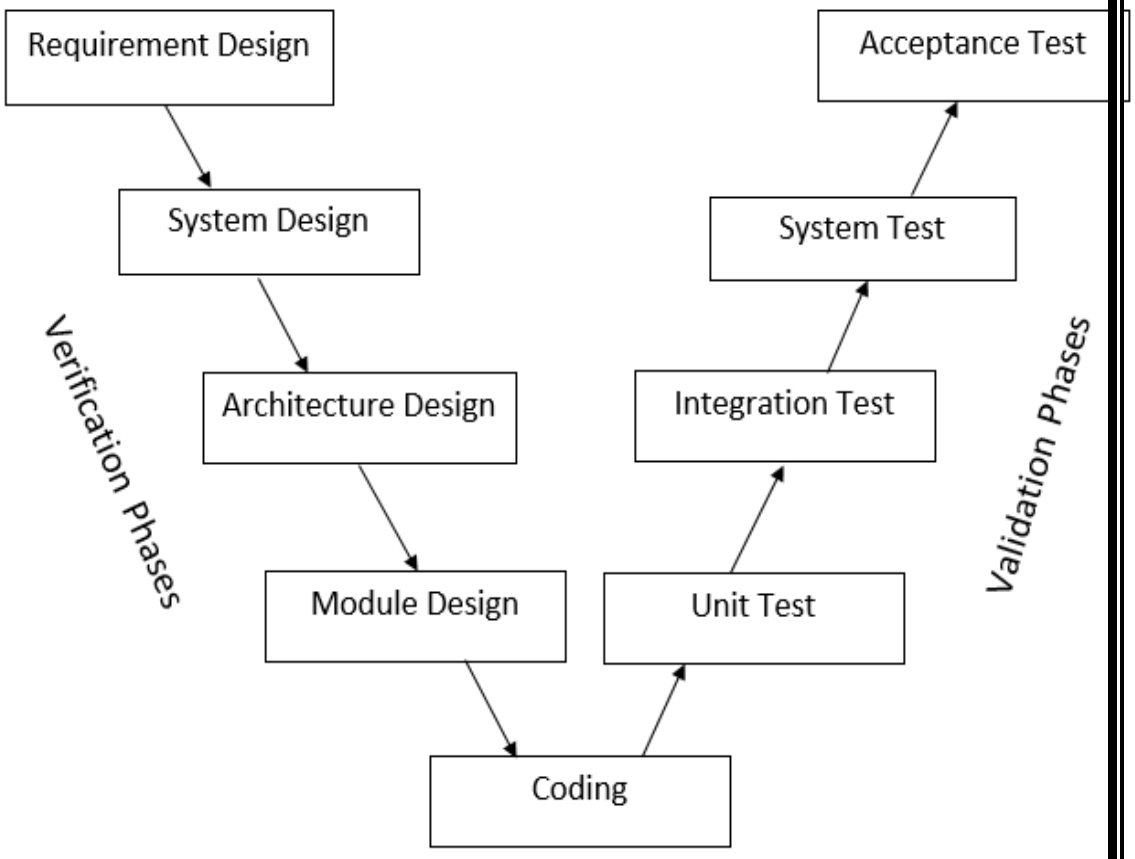
V- Model:

The V-Model has gained acceptance because of its simplicity and straight forwardness. However, some developers believe it is too rigid for the evolving nature of IT (Information technology) business environments.

The V-Model is a unique, linear development methodology used during a software development life cycle (SDLC). The V-Model focuses on a fairly typical waterfall-esque method that follows strict, step-by-step stages. While initial stages are broad design stages, progress proceeds down through more and more granular stages, leading into implementation and coding, and finally back through all testing stages prior to completion of the project.

- Requirement Design.
- System Design.
- Architecture Design.

- Module Design.
- Coding.
- Unit Test.
- Integration Test.
- System Test.
- Acceptance Test.



7. TESTING

7. Testing

Testing Strategies:

There are two general strategies for testing software. There are follows.

Code Testing:

This examines the logic of the program. To follow this test, cases are developed such that very path of the program is tested.

Specification Testing:

Specification testing examines the specifications starting what the program should do and how it should perform under various conditions. Then test cases are developed for each conditions and combinations of conditions and to be submitted for processing.

Stages in the testing process:

Unit Testing:

Individual components are tested to ensure that they operate correctly. Each component is tested independently without other system components.Ex. Checked for Login and Password with the table.

Module Testing:

Module is a collection of dependent components such as an object class an abstract data type or some looser collection of procedures and functions. A module encapsulates related components so can be tested without other system modules.

Subsystem Testing:

This phase involves testing collection of modules, which have been integrated into subsystems. Subsystems may be independently designed and implemented. The most common problems which arise in the large software systems are subsystems interface mismatches. The subsystem test process should therefore concentrate on the detection of interface errors by rigorously exercising these interfaces.

System Testing:

The subsystems are integrated to make up the entire system. The testing process is concerned with finding errors, which result from unanticipated interactions between subsystems and system components. It is also concerned with validating that the system is functional and non-functional requirements.

Ex. Those all subsystems are integrated and checked for inter-dependency between the subsystems.

Acceptance Testing:

This is final stage in testing process before the system is tested for operational use. The system is tested with data supplied by the system procurer rather than simulated test data. Acceptance testing may reveal errors and omissions in the systems requirements definitions because the real data exercises the system in different phase from the test data. Acceptance testing may also reveal the requirements problems where the system facilities do not really meet the user's needs or system performance is unacceptable.

Ex. We tested for all the objectives which were stated in the project statement whether they meet the requirements or not.

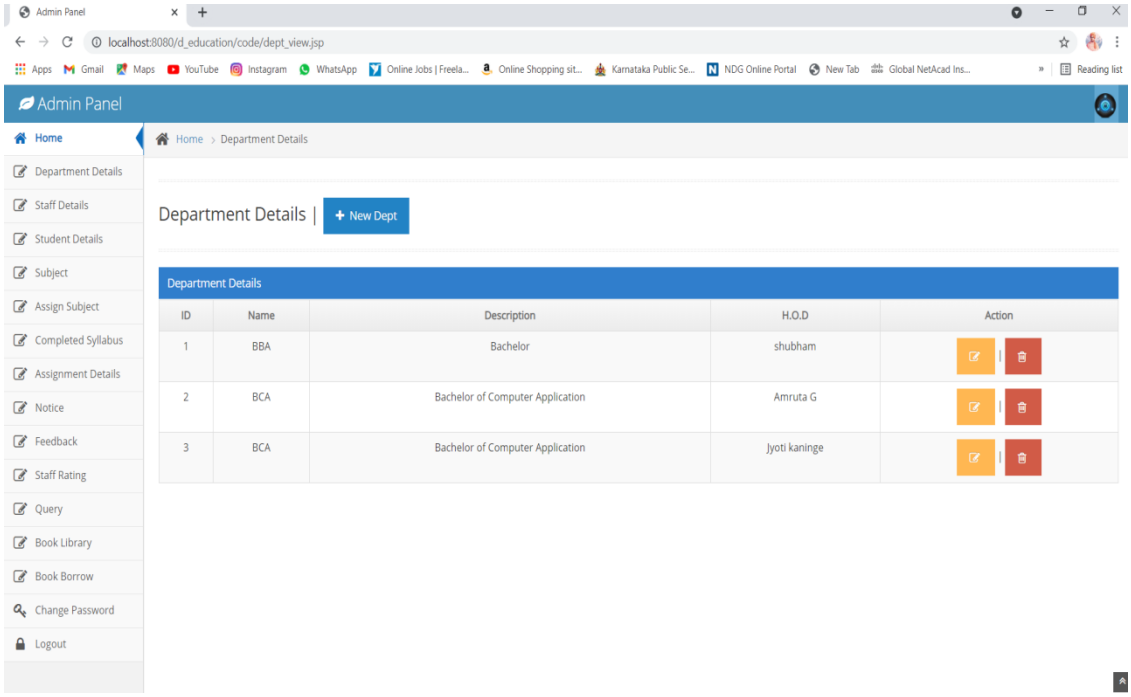
Test:

	Test case	Expected Output	Observed Output	Remark
1	Login, By blank Username.	User name should not be blank.	The user name should not be blank.	OK
2	Login, By blank Password.	Password should not be blank.	The password should not be blank.	OK
3	User name and password both blank.	User name and password should not be blank.	The User name and password should not be blank.	OK
4	Wrong entry of user name and password.	Invalid user name and password.	The user name and password is invalid.	OK
5	Wrong entry of name.	Characters only.	Enter Characters only.	OK
6	Wrong entry of Mobile no.	Numbers only.	Enter Numbers only, minimum and maximum 10 digits.	OK
7	NULL value for ID (primary key)	Null not allowed.	ID should not be null, Enter ID	OK
8	Wrong entry of email id.	Invalid email address.	Invalid email address.	OK

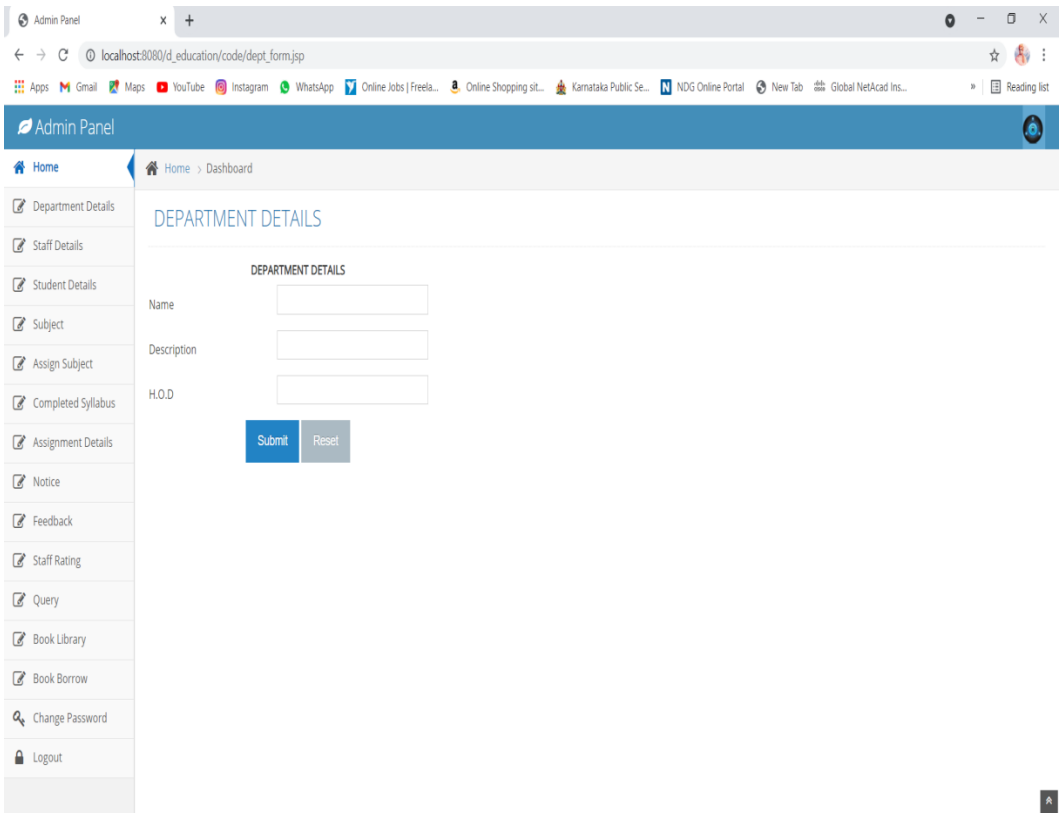
8. SCREEN SHOTS

8. Screen Shots:

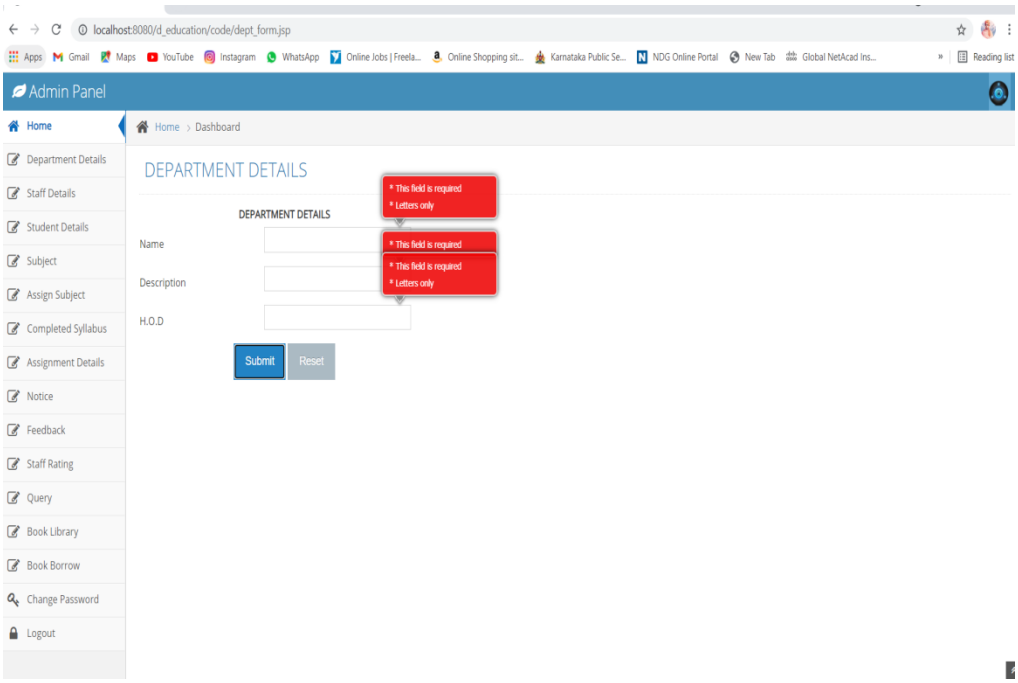
View:Department Details



Insert: Department Details



Validation : Department Details



9. CONCLUSION

9. Conclusion:

While developing E-Learning Platform an eye has been kept on making it as user-friendly as possible. As one may hope that the system will be acceptable to any specific organization or college or school and will adequately meet their needs. E-Learning platform allows us to maintain the information of whole organization and Library, Student and staff details, Feedbacks, Rating details, Study materials. E-Learning Platform allows Novel admin to manage the students records easily. It also provides an easy way of finding and reading the Study materials for students.

10. FUTURE ENHANCEMENTS

10. Future Enhancement:

In future we are planning to host and manage the application on cloud.

- We can use this web application form anywhere using LAN or WAN.
- We can also use this via mobiles.
- Periodically updated versions of this system will be released with some new features.

11. BIBLIOGRAPHI

11. Bibliography:

Text Books:

- The Complete Reference (Helbertschildt published fifth edition)
- Server side programming with (Aptech J2EE bookJSP & Servlet)

Web sites:

- Google.
- Encyclopedia.
- Wikipedia.
- http://Java.sun.com/j2ee/faq/html
- http://Java.sun.com/products/jdbc/reference/faqs/index.html
- http://www.apt.jhu.edu/~hall/java/servlet-Tutorial