# COMPONENT DIAGRAM AND DEPLOYMENT DIAGRAM

# DATE: 26-03-2015

**EX.NO: 12**

# Aim

To draw the component diagram and deployment diagram for the book bank system.

Theory:

**Component:**

A component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Graphically, a component is rendered as a rectangle with tabs.

**Names:** Every component must have a name that distinguishes it from other components. A name is a textual string.

**Components and classes:** In many ways, components are like classes. Both have names; both may realize a set of interfaces; both may participate in a dependency, generalization, and association relationship; both may be nested; both may have instances; both may be participants in interactions. However, there are some significant differences between component and classes.

Classes represent logical abstractions; components represent physical things that live in the world of bits.

* Components represent the physical packaging of otherwise logical components and are at a different level of abstraction.
* Classes may have attributes and operations directly. In general, components only have operations that are reachable only through their interfaces.

**Kinds of components may be distinguished-**

* 1. Deployment components: These are the components necessary and sufficient to form an executable system, such as dynamic libraries and executable.
  2. Work Product Components: The components are essentially the residue of the development process, consisting of things such as source code files and data files from which deployment component are created.
  3. Execution Components: These component are created has a consequences of an executing system, such as COM+ object, which is instantiated from a DLL.

**Component Diagram:**

A component diagram shows a set of components and their relationships. Graphically, a components diagram is a collection of vertices and arcs. A components diagram is just a special kind of diagram and shares the same common properties as do all other diagrams-a name and graphical contents that are a projection into a model. What distinguishes a component diagram from all other kinds of diagrams is its particular content.

Component diagram commonly contain

• Components

• Interfaces

• Dependency, generalization, association and realization relationships.

Component diagram may also contain packages or subsystem, both of which are used to group elements of your model into larger chunks. Sometimes you’ll want to place instances in your component diagrams, as well, especially when you want to visualize one instances of a family of component- based system.

**Component:** A component is a physical building block of the system. It is represented as a rectangle with tabs.

**Interfaces:** An interface describes a group of operations used or created by components.

Component qualification: System requirements and architecture define the component that will be required. Reusable components are normally identified by the characteristics of their interfaces.

**Component adaptation:** The existing reusable components must be adapted to meet the meets of the architecture or discarded and replaced by other, more suitable components.

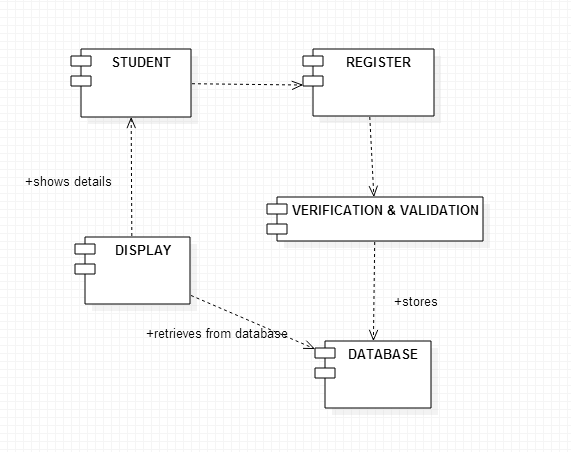
**Component Composition:** Architectural style plays a key role in the way in which software components are integrated to form a working system.

**Component Update:** When systems are implemented, update is complicated by the imposition of the organization that developed the reusable component may be outside the immediate control of the software engineering corporation.

Conclusion:

1. ‘Component diagram describe the organization of physical software components, including source code, run time (binary) code, and executables.
2. The different high level reusable parts of a system are represented in a component diagram.
3. The primary difference is that component diagram represents the implementation perspective of a system.
4. Component diagram is used to model the static implementation view of a system.

**COMPONENT DIAGRAM:**

****

**DEPLOYMENT DIAGRAM:**

Theory:

A deployment diagram shows processors, devices, and connections. Each model contains a single deployment diagram which shows the connections between its processors and devices, and the allocation of its processes to processors. Processor Specifications, Device Specifications, and Connection Specifications enable you to display and modify the respective properties. The information in a specification is presented textually; some of this information can also be displayed inside the icons. You can change properties or relationships by editing the specification or modifying the icon on the diagram. The deployment diagram specifications are automatically updated.

**Processor**

A processor is a hardware component capable of executing programs.

**Naming**

Each processor must have a name. There are no constraints on the processor name because processors denote hardware rather than software entities. Graphical Depiction The icon for a processor is a shaded box:

**Adornments**

You can further define a processor by identifying its processes and specifying the type of process scheduling it uses. You can set the following adornments in the Processor Specification. You can display the information in the deployment diagram by selecting an item from the processor shortcut menu.

**Scheduling**

You can specify the type of process scheduling used by this processor by setting a scheduling type:

**Type** **Description**

Preemptive (default) Higher-priority processes that are ready to execute can preempt lower-priority processes that are currently executing. Nonpreemptive The current process continues to execute until it relinquishes control.

Cyclic Control passes from one process to another.

Executive An algorithm controls process scheduling.

Manual Processes are scheduled by the user outside of the system.

**Processes**

Processes represent single threads of control. Examples include the main program from a component diagram or the name of an active object from a collaboration diagram. To add a process to the processor, double-click on in the Processes field to displays the Process Specification.

**Device**

A device is a hardware component with no computing power. Each device must have a name. Device names can be generic, such as "modem" or "terminal."

**Graphical Depiction**

The icon for a device is a box:

**Connection**

A connection represents some type of hardware coupling between two entities. An entity is either a processor or a device. The hardware coupling can be direct, such as an RS232 cable, or indirect, such as satellite-to-ground communication. Connections are usually bi-directional.

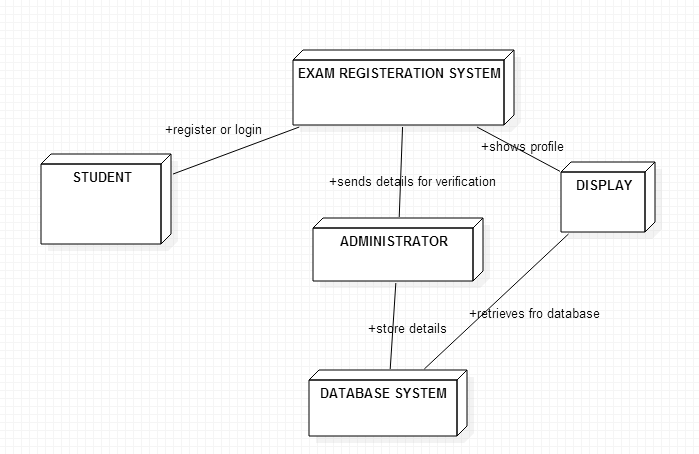
**Naming**

You can optionally label the connection with its name.

**Graphical Depiction**

The icon for a connection is a straight line:

**DEPLOYMENT DIAGRAM:**



**Result:**

Thus, the component and deployment diagrams for exam registration system were successfully drawn using UML.

**EX-NO:9 IMPLEMENTATION OF EXAM REGISTRATION SYSTEM**

**-TECHNICAL LAYER**

**PROGRAM:**

**login.html:**

->The Login page code.

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Login.</title>

<style>

body {

color: ghostwhite;

}

span {

color: red;

}

h3 {

color: tomato;

}

</style>

</head>

<body>

<body bgcolor="green">

<center><h1>Exam Registration-Monitoring system.</h1></center>

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

<div align=left>

<input type="hidden" value="list" name="action">

<h3>Login</h3>

<table>

<tr>

<td><h3>Register Number:<span id=1>\*</span></h3></td>

<td><input type="text" name="reg\_no" size="30" required></td>

</tr>

<tr>

<td><h3>First Name:<span id=1>\*</span></h3></td>

<td><input type="text" name="name" size="30" required></td>

</tr>

</table>

<h3>Not already registered?</h3>

<a href="exam.html"><h3>Register.</h3></a>

<br/>

<div align="center">

<input type="submit" value="submit">

<input type="reset" value="reset">

<br /><br />

</div>

</form>

</body>

</html>

**exam.html:**

->The register page code.

<!DOCTYPE html>

<html>

<head>

<title>Examination Registration</title>

<script language="javascript">

function validation()

{

if(form.reg\_no.value.length==0)

{

alert("fill the Register number");

form.reg\_no.focus();

return false;

}

return true;

}

</script>

<style>

body {

color: ghostwhite;

}

span {

color: red;

}

</style>

</head>

<body bgcolor="purple">

<center><h1>Exam Registration Form</h1></center>

<form action="exam.jsp" method="post" id="form" name="form" onSubmit="return validation()">

<div align=left>

<input type="hidden" value="list" name="action">

<table>

<tr>

<td><h3>Register Number:<span id=1>\*</span></h3></td>

<td><input type="text" name="reg\_no" size="30" required></td>

</tr>

<tr>

<td><h3>First Name:<span id=1>\*</span></h3></td>

<td><input type="text" name="name" size="30" required></td>

</tr>

<tr>

<td><h3>Last Name:<span id=1>\*</span></h3></td>

<td><input type="text" name="lname" size="30" required></td>

</tr>

<tr>

<td><h3>Year:<span id=1>\*</span></h3></td>

<td><input type="number" name="year" size="30"></td>

</tr>

<tr>

<td><h3>Semester:<span id=1>\*</span></h3></td>

<td><input type="text" name="semester" size="30"></td>

</tr>

<tr>

<td><h3>Branch:<span id=1>\*</span></h3></td>

<td><input type="text" name="br" size="30"></td>

</tr>

</table>

</div>

<hr />

<hr align="left" />

<div align="center">

<input type="submit" value="submit">

<input type="reset" value="reset">

<br /><br />

</div>

</form>

</body>

</html>

**exam.jsp:**

->The Profile display page code using JSP (Java servlet pages).

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

<%@page import="java.io.\*" %>

<%@page import="java.util.\*" %>

<%

String reg\_no,fname,lname,year,branch,semester;

//String ans1,ans2,ans3,ans4,ans5;

int a1,a2,a3,a4,a5;

a1=a2=a3=a4=a5=0;

Connection conn=null;

Statement stmt=null;

ResultSet rs=null;

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

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

if(request.getParameter("action")!=null) {

reg\_no=request.getParameter("reg\_no");

fname=request.getParameter("name");

if(request.getParameter("lname")!=null)

{

lname=request.getParameter("lname");

a1=1;

}

else

lname="";

if(request.getParameter("year")!=null)

year=request.getParameter("year");

else

year="";

if(request.getParameter("semester")!=null)

semester=request.getParameter("semester");

else

semester="";

if(request.getParameter("br")!=null)

branch=request.getParameter("br");

else

branch="";

String query;

if(a1==1)

{

stmt=conn.createStatement();

query="INSERT INTO examreg(reg\_no,fname,lname,year,sem,br)VALUES('"+reg\_no+"','"+fname+"','"+lname+"','"+year+"','"+semester+"','"+branch+"')";

int result=stmt.executeUpdate(query);

stmt.close();

}

stmt=conn.createStatement();

query="SELECT \* FROM examreg WHERE fname='"+fname+"'";

rs=stmt.executeQuery(query);

%>

<html><head><title>Scrore</title></head>

<body bgcolor="orange">

<center>

<h2>Anna university,Chennai.</h2>

<h3>Student details.</h3>

<h4>Welcome Guest,</h4>

<hr />

<br /><br />

<table border="1">

<tr>

<td><b>Registration Number</b></td>

<td><b>FirstName</b></td>

<td><b>LastName</b></td>

<td><b>Year</b></td>

<td><b>Semester</b></td>

<td><b>Branch</b></td>

<td>

</tr>

<%

while(rs.next()) {

%>

<tr>

<td><%=rs.getString(1)%></td>

<td><%=rs.getString(2)%></td>

<td><%=rs.getString(3)%></td>

<td><%=rs.getInt(4) %></td>

<td><%=rs.getString(5) %></td>

<td><%=rs.getString(6) %></td>

</tr>

<%

}

rs.close();

stmt.close();

conn.close();

%>

</table>

</center><br /><br />

<p>

<center><i>Date: <%=new java.util.Date().toString()%> &nbsp; <b>&copy;ssn</b> </i></center>

</p>

<div>

<a href="login.html">Logout.</a>

</div>

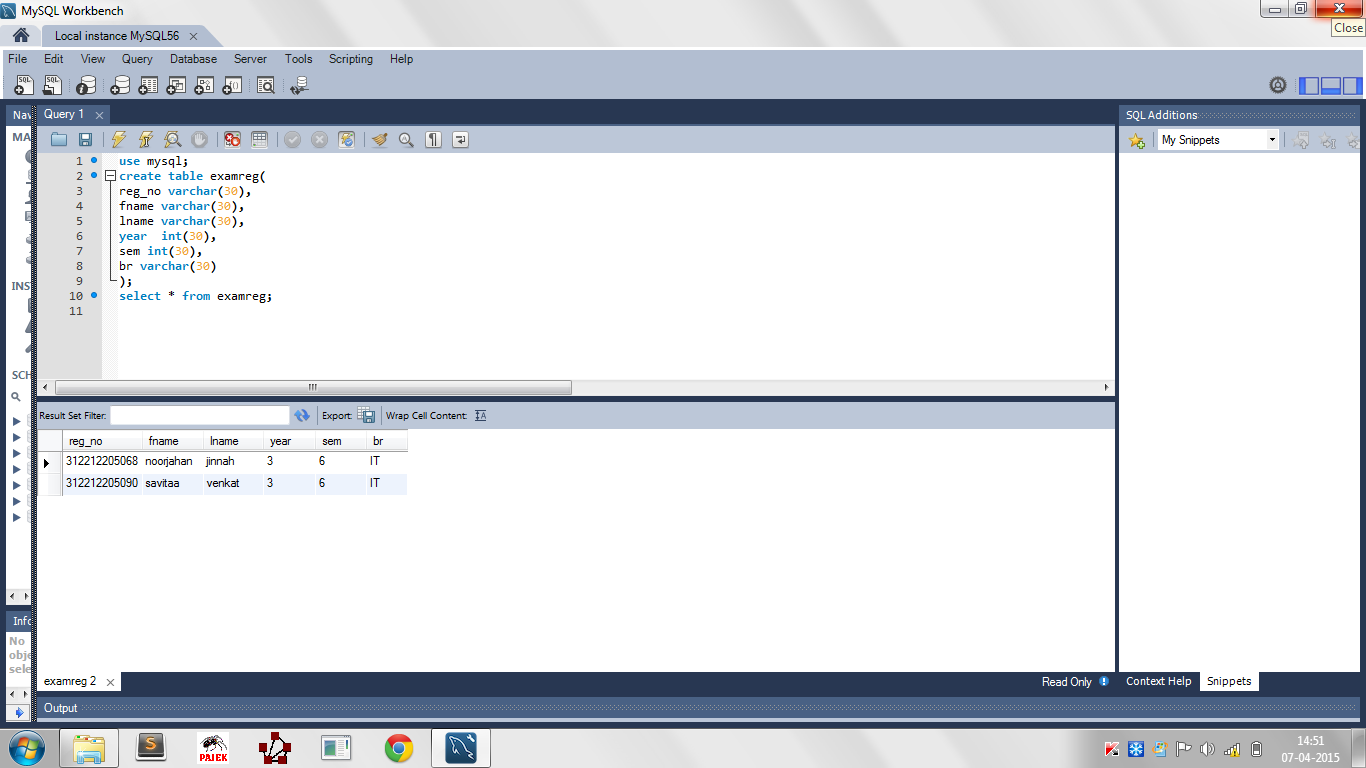
</body>

</html>

<%} else {%>

<%}%>

**Database :**



**Result:**

Thus, the Technical layer of Exam Registration System was successfully implemented.

**Ex No:11 IMPLEMENTATION OF EXAM REGISTRATION**

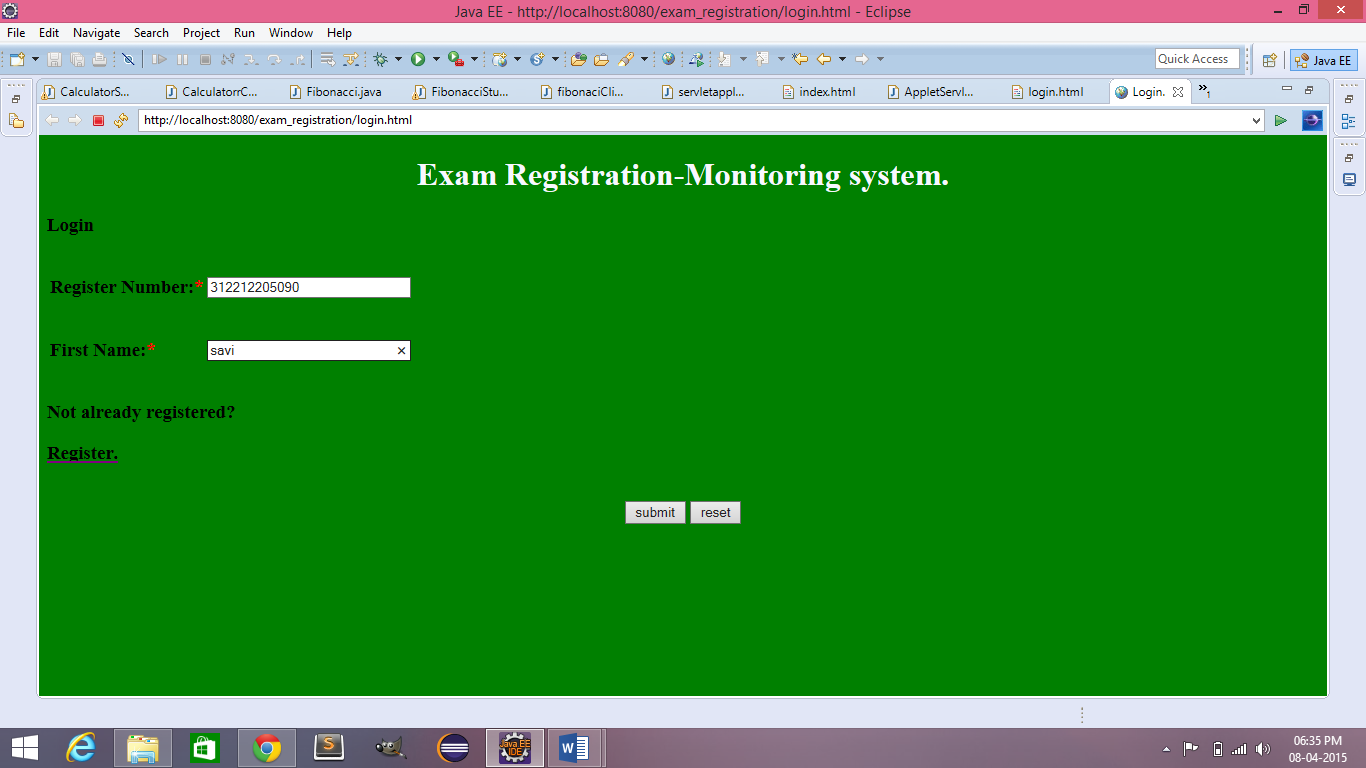
**SYSTEM- USER INTERFACE LAYER**

**UI layer – look and feel:**

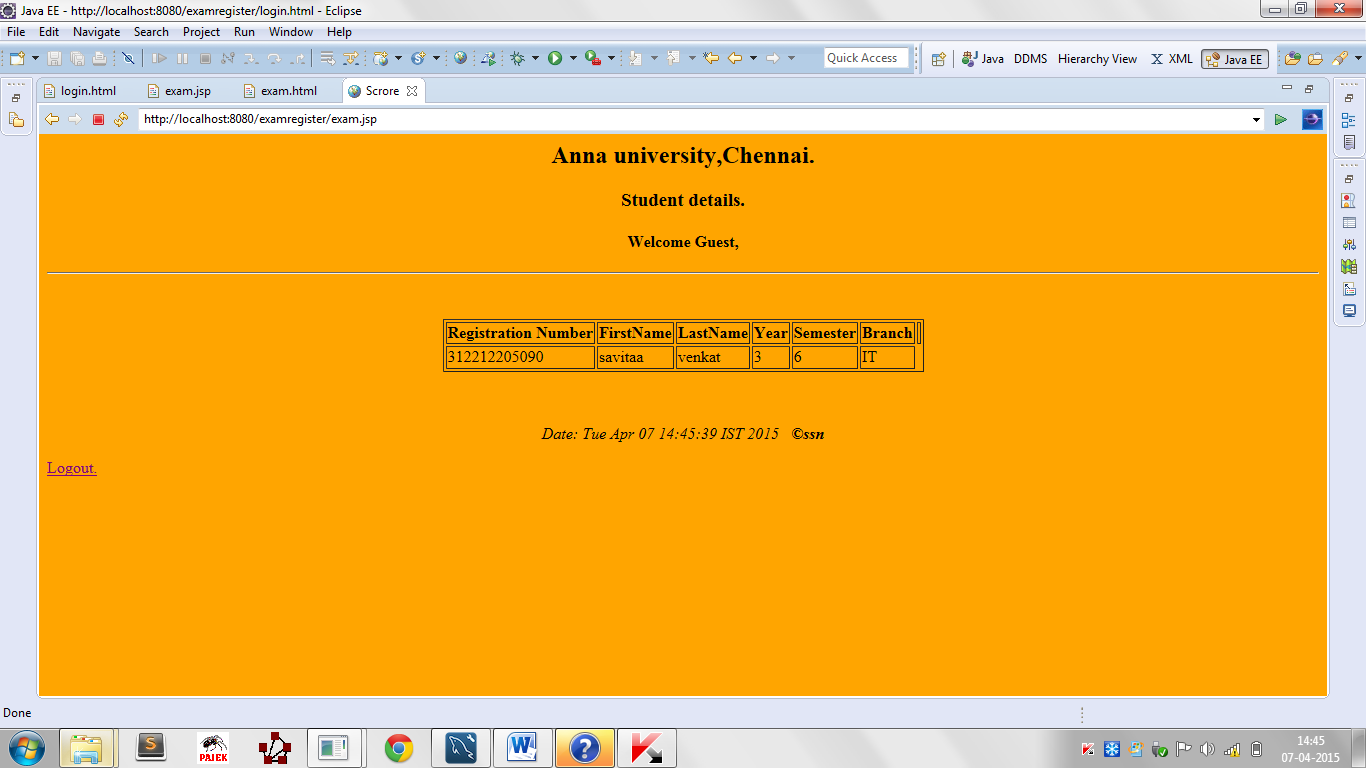
The User Interface Layer (UI layer) is the top most layer of the three tier architecture followed in the implementation process of Exam Registration System software. It primarily is designed to provide an user access to the system in a user-friendly manner. It will serve as the interface between the High end user and the Low end database/technical layer.

**Output:**

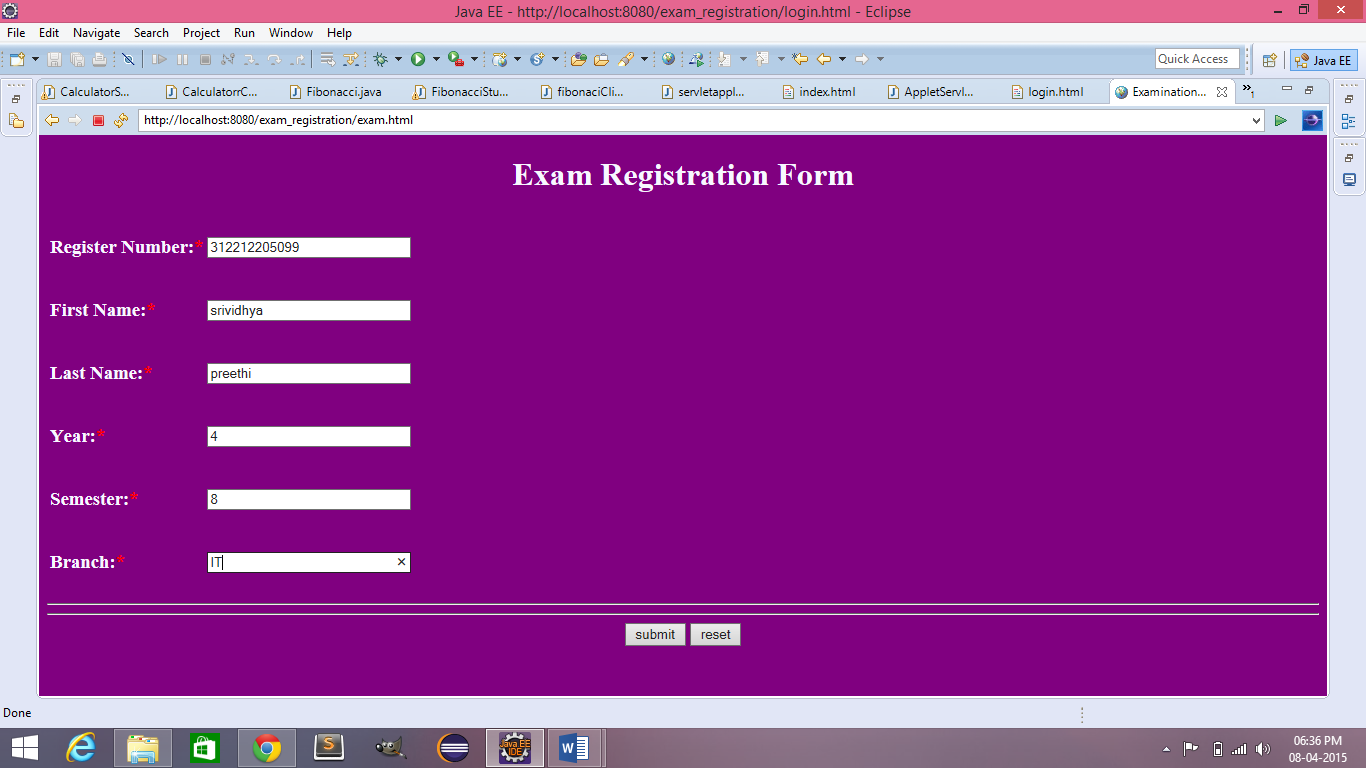
* 1. Initial Login. If the user is a registered user, he/she can use this and login. On successful login, profile of the user will be displayed.



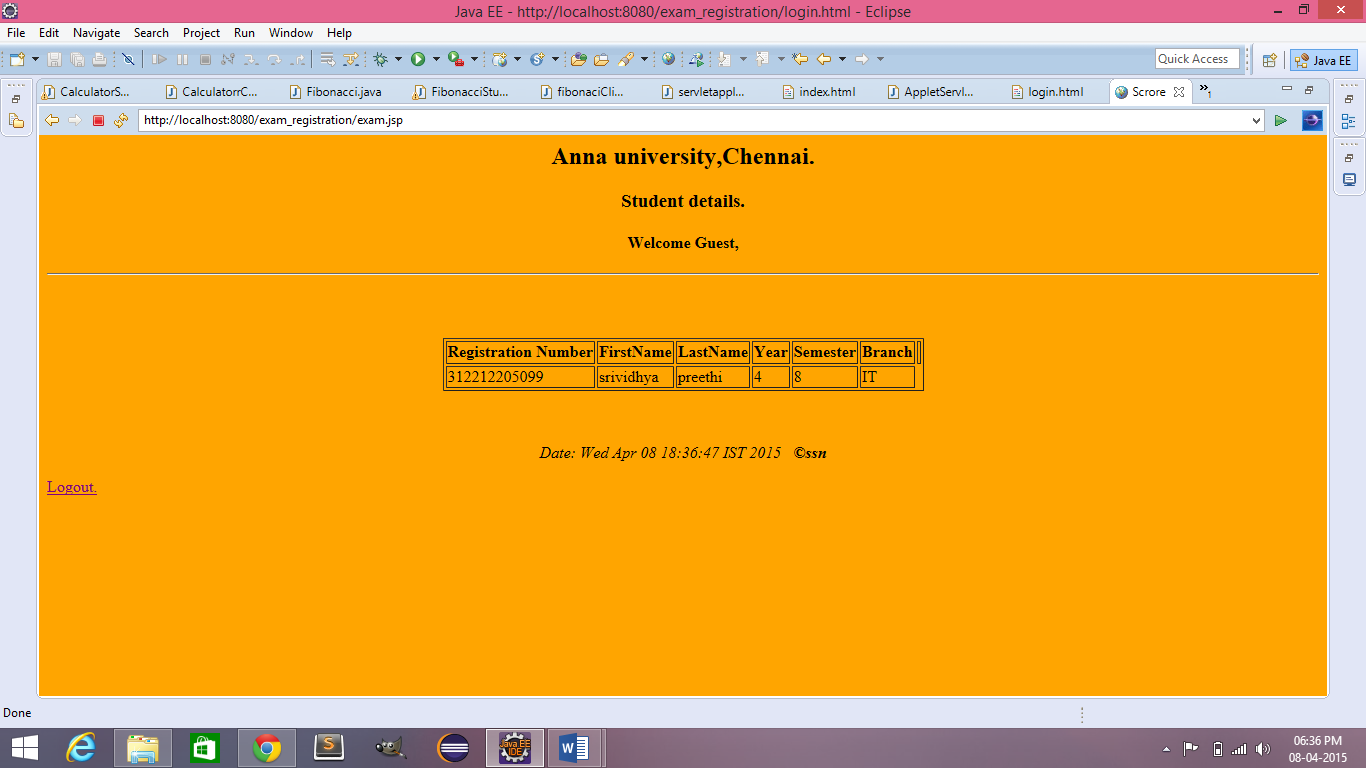
* 1. Profile of the user.



* 1. If the user isn’t a registered user. He/she must register via the register link in login page which redirects to register page.



1. Profile of the user on registering.



1. Clicking on Logout will get the user back to the initial Login page.

**Result:**

Thus, the User Interface Layer was successfully implemented for Exam Registration System.