

# **Web Architecture and Application Development Laboratory**

**B.Tech. 7th Semester**



**Department: Computer Science and Engineering**

**Faculty of Engineering & Technology**  
**M. S. Ramaiah University of Applied Sciences**

**NAME: SATYAJIT GHANA**  
**REG NO: 17ETCS002159**



## Ramaiah University of Applied Sciences

Private University Established in Karnataka State by Act No. 15 of 2013

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Course	Web Architecture and Application Development Laboratory
Year/Semester	2017/7th Semester
Course Code	CSC404A

### List of Experiments

1. Software design
2. Software design
3. Database design
4. HTML and JSP form for student registration
5. HTML and JSP form for login and search
6. Servlet, server side component
7. JDBC for Persistence Management
8. Client side validation using Java script
9. Functionality implementation
10. Mock

Consider a scenario where a student enrolls for a course in university. The student also registers in the library available at university by providing his basic details. Consider all the attributes of the student and library to develop the web application design.

## Laboratory 1, 2

Title of the Laboratory Exercise: Functional, Non-functional and UML diagrams

1. Introduction and Purpose of Experiment

2. Aim and Objectives

Aim

- To develop Functional and Non-Functional requirements, ER diagram, class diagram, interaction sequence diagram and algorithm/flowchart

Objectives

At the end of this lab, the student will be able to

- Model the information required for the given scenario using E-R diagrams
- Develop ER diagram, class diagram, interaction sequence diagram and algorithm/flowchart

3. Experimental Procedure

Students are given a set of instructions to be executed on the computer. The instructions should be edited and executed and documented by the student in the lab manual. They are expected to answer questions posed in section 5 based on their experiment.

4. Presentation of Results

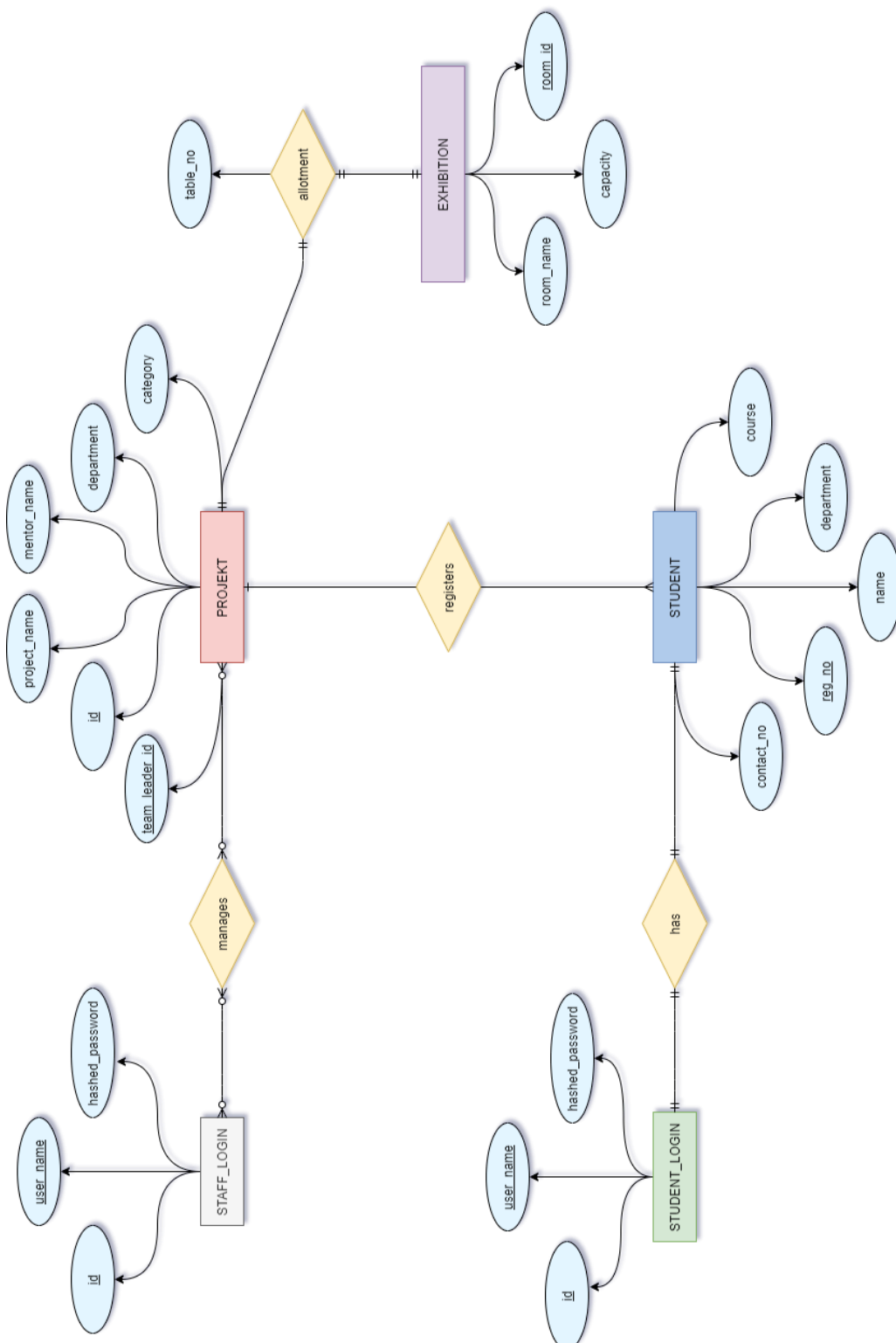
**Scenario:** Student Group Project Management System

**Functional Requirement:**

1. The system should allow the staff to login and students to login/register
2. The system should allow the students to enter their group project details in a form
3. The system should allow the students to register their group project
4. The system should display the room and table no. allotted to the group
5. The system should allow the students to cancel their registration
6. The system should allow the staff to view all the registered group projects
7. The system should allow the students to view the allotted room and table no of a group project
8. The system should allow the staff to cancel a group registration

**Non-functional Requirement**

- The system must be secure
- The system must not collapse while handling large number of users
- The system must be reliable.

**ER Diagram:**

NAME: SATYAJIT GHANA

REG NO: 17ETCS002159

**Relational Schema:**

**STAFF\_LOGIN**

<u>id</u>	user_name	hashed_password
-----------	-----------	-----------------

**STUDENT\_LOGIN**

<u>id</u>	user_name	hashed_password
-----------	-----------	-----------------

**STUDENT**

<u>fk id</u>	<u>reg no</u>	name	department	course	contact_no
--------------	---------------	------	------------	--------	------------

**PROJEKT**

<u>fk team leader id</u>	<u>id</u>	project_name	mentor_name	department	category
--------------------------	-----------	--------------	-------------	------------	----------

**PROJECT\_STUDENT\_REGISTER**

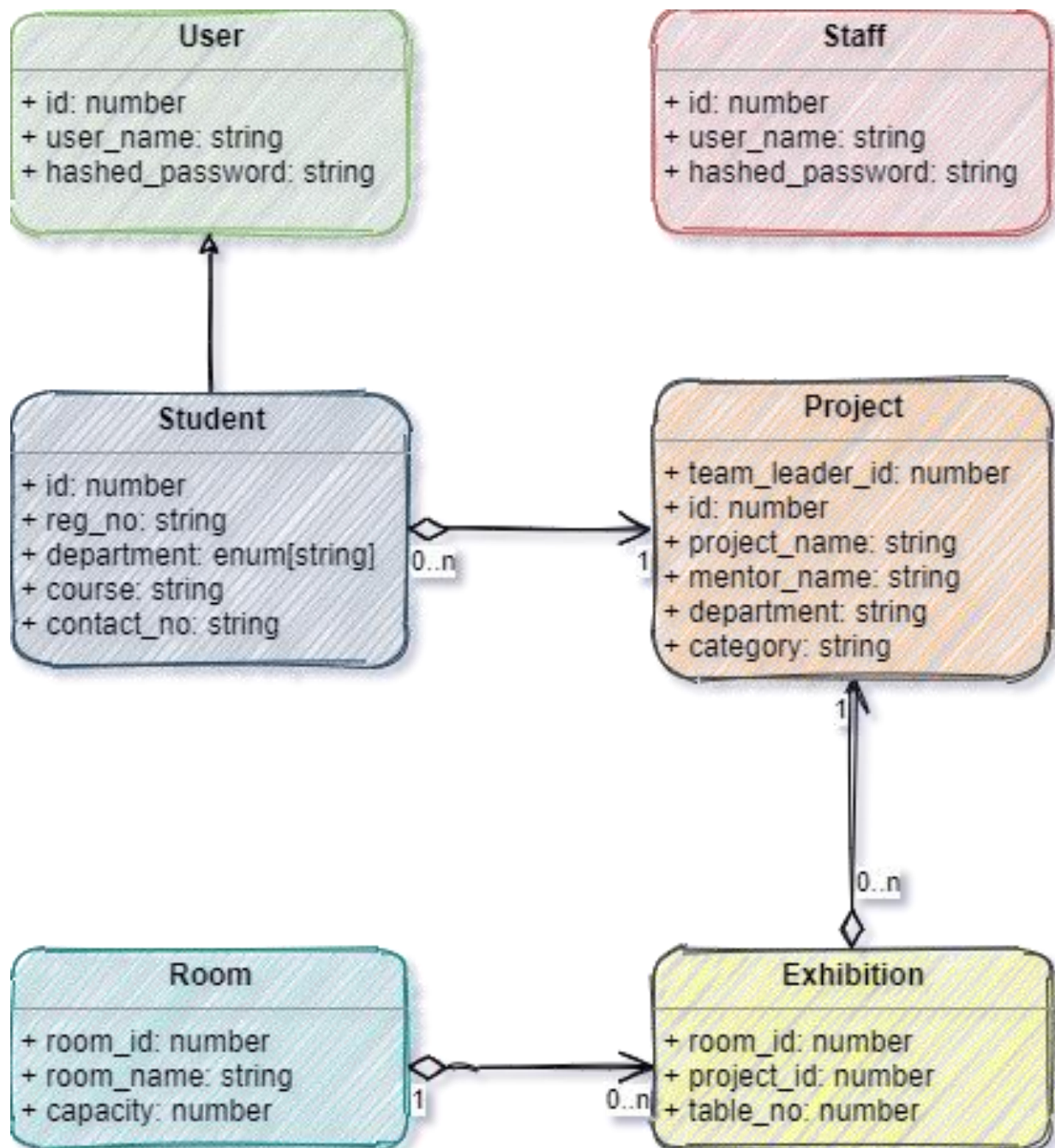
<u>fk project id</u>	<u>fk student id</u>
----------------------	----------------------

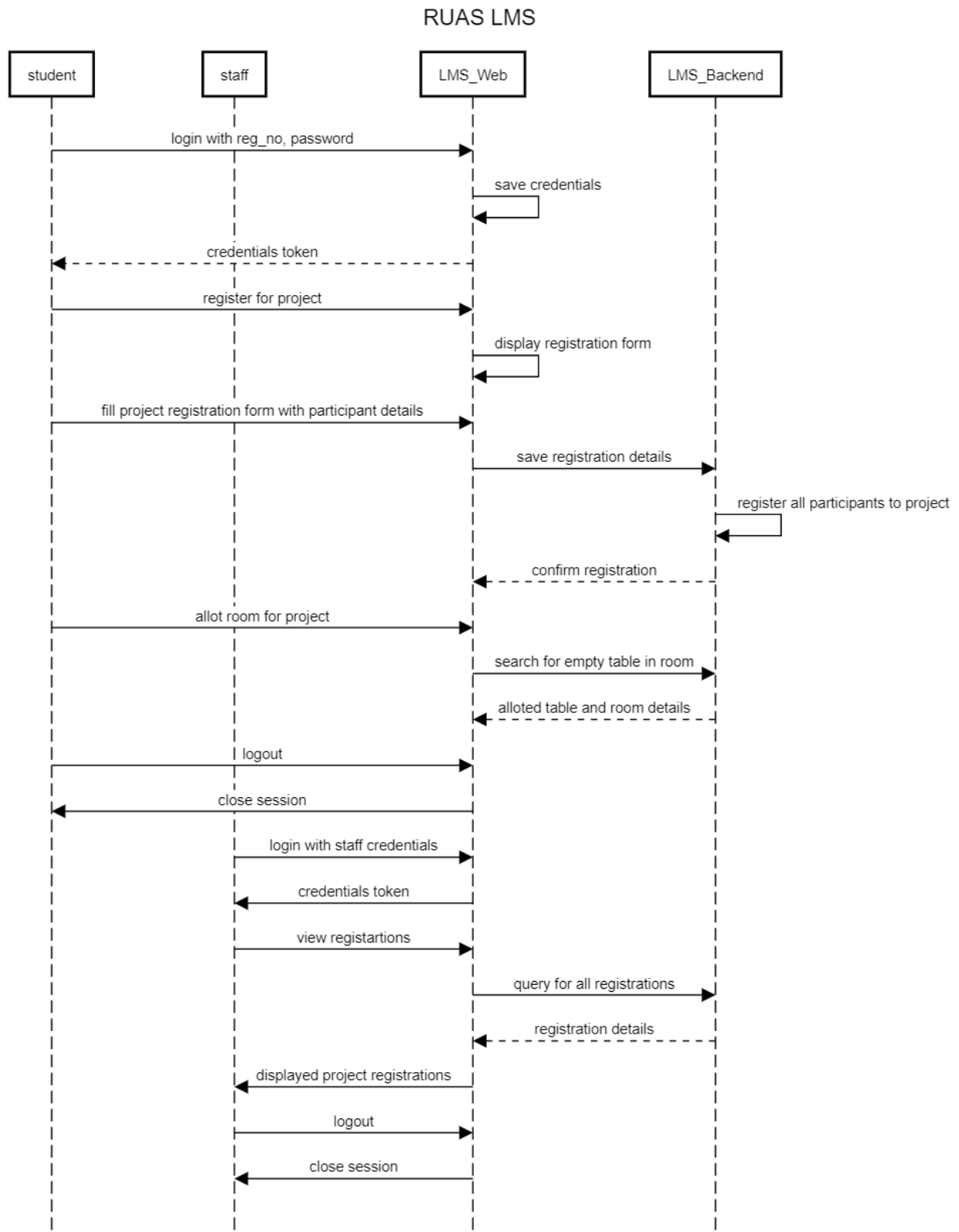
**EXHIBITION**

<u>room id</u>	room_name	capacity
----------------	-----------	----------

**PROJECT\_EXHIBITION**

<u>fk room id</u>	<u>fk project id</u>	table_no
-------------------	----------------------	----------

**Class Diagram:**

**Sequence Diagram:**

## 5. Analysis and Discussions

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. (GFG 2018)

Entity Relational is a high-level conceptual data model diagram. ER modeling helps you to analyze data requirements systematically to produce a well-designed database. The Entity-Relation model represents real-world entities and the relationship between them. It is considered a best practice to complete ER modeling before implementing your database.

## 6. Conclusions

Functional and Non-Functional requirements, ER diagram, class diagram, interaction sequence diagram and algorithm/flowchart were developed in this lab.



## Laboratory 3

Title of the Laboratory Exercise: Database Design

### 1. Introduction and Purpose of Experiment

Students will design the database schema as per the ER diagram designed in the Laboratory 1 and 2.

### 2. Aim and Objectives

Aim

Objectives

At the end of this lab, the student will be able to

### 3. Experimental Procedure

### 4. Calculations/Computations/Algorithms

#### Creation of the database

```
DROP TABLE IF EXISTS PROJECT_EXHIBITION;
DROP TABLE IF EXISTS EXHIBITION;
DROP TABLE IF EXISTS PROJECT_STUDENT_REGISTER;
DROP TABLE IF EXISTS PROJEKT;
DROP TABLE IF EXISTS STUDENT;
DROP TABLE IF EXISTS STUDENT_LOGIN;
DROP TABLE IF EXISTS STAFF_LOGIN;

CREATE TABLE STAFF_LOGIN
(
    id INT(5) PRIMARY KEY AUTO_INCREMENT,
    user_name VARCHAR(20) UNIQUE KEY NOT NULL,
    hashed_password CHAR(60) NOT NULL
);

CREATE TABLE STUDENT_LOGIN
(
    id INT(5) PRIMARY KEY AUTO_INCREMENT,
    user_name VARCHAR(20) UNIQUE KEY NOT NULL,
    hashed_password CHAR(60) NOT NULL
);
```

```
CREATE TABLE STUDENT
(
    id INT(5) UNIQUE KEY NOT NULL,
    reg_no CHAR(12) PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    department ENUM('CSE', 'EEE', 'ECE', 'CIVIL'),
    course ENUM('B.Tech', 'M.Tech') NOT NULL,
    contact_no VARCHAR(10) NOT NULL,
    FOREIGN KEY(id) REFERENCES STUDENT_LOGIN(id)
);

CREATE TABLE PROJEKT
(
    id INT(5) PRIMARY KEY AUTO_INCREMENT,
    project_leader_regno CHAR(12) UNIQUE KEY NOT NULL,
    project_name VARCHAR(100) UNIQUE KEY NOT NULL,
    mentor_name VARCHAR(30) NOT NULL,
    department ENUM('CSE', 'EEE', 'ECE', 'CIVIL') NOT NULL,
    category VARCHAR(30) NOT NULL
);

CREATE TABLE PROJECT_STUDENT_REGISTER
(
    project_id INT(5) NOT NULL,
    student_reg_no CHAR(12) NOT NULL,
    FOREIGN KEY(project_id) REFERENCES PROJEKT(id),
    FOREIGN KEY(student_reg_no) REFERENCES STUDENT(reg_no)
);

CREATE TABLE EXHIBITION
(
    room_id INT(5) PRIMARY KEY AUTO_INCREMENT,
    room_name CHAR(20) UNIQUE KEY NOT NULL,
    capacity INT(5) NOT NULL
);

CREATE TABLE PROJECT_EXHIBITION
(
    room_id INT(5) NOT NULL,
    project_id INT(5) UNIQUE KEY NOT NULL,
    table_no INT(5) CHECK ( table_no > 0 AND table_no < ( SELECT * FROM EXHIBITION WHERE EXHI
BITION.room_id = room_id LIMIT 1 ) ),
    FOREIGN KEY(room_id) REFERENCES EXHIBITION(room_id),
    FOREIGN KEY(project_id) REFERENCES PROJEKT(id)
);
```

Inserting data into the table

```
INSERT INTO `STUDENT_LOGIN` (`id`, `user_name`, `hashed_password`) VALUES
(1, '17ETCS002159', '$2b$10$uVRx4ogFBiOowMljpvEi1ONnd9wWOMtrpgVwqw2Mw8.aNmo6yEU1u'),
(2, '17ETCS002122', '$2b$10$ATp9qxsPWBs0UXDAB1YvK.yTLi4GK1mzpIHBCfSQCQwtXLU/52Pk2'),
(3, '17ETCS002168', '$2b$10$3cfBMD3yRi3YJk.fFGrNY.Yx1RRonj4z2cqg0e2fgZ78yNaqRxFc');

INSERT INTO `STUDENT` (`id`, `reg_no`, `name`, `department`, `course`, `contact_no`) VALUES
(2, '17ETCS002122', 'Prachi Poddar', 'CSE', 'B.Tech', '9856523658'),
(1, '17ETCS002159', 'Satyajit Ghana', 'CSE', 'B.Tech', '7892137665'),
(3, '17ETCS002168', 'Shikhar Singh', 'CSE', 'B.Tech', '9852145896');

INSERT INTO `PROJEKT` (`id`, `project_leader_regno`, `project_name`, `mentor_name`, `department`, `category`) VALUES
(2, '17ETCS002159', 'KrishiAI', 'Chaitra S', 'CSE', 'DL');

INSERT INTO `PROJECT_STUDENT_REGISTER` (`project_id`, `student_reg_no`) VALUES
(2, '17ETCS002159'),
(2, '17ETCS002122'),
(2, '17ETCS002168');

INSERT INTO `EXHIBITION` (`room_id`, `room_name`, `capacity`) VALUES
(1, 'A201', 60);
```

## 5. Presentation of Results

Now we run the created table sql file to create all the necessary tables with appropriate constraints.

```
mysql> source MYSQL-INF/create_tables.sql
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.06 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.10 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.04 sec)

mysql> show tables;
+-----+
| Tables_in_webarch |
+-----+
| EXHIBITION         |
| PROJECT_EXHIBITION |
| PROJECT_STUDENT_REGISTER |
| PROJEKT            |
| STAFF_LOGIN        |
| STUDENT            |
| STUDENT_LOGIN      |
+-----+
7 rows in set (0.00 sec)

mysql> █
```

```

mysql> use webarch;
mysql> source MYSQL-INF/insert_data.sql
Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0

Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.01 sec)

Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENT_LOGIN
-> ;
+-----+-----+-----+
| id | user_name | hashed_password |
+-----+-----+-----+
| 1 | 17ETCS002159 | $2b$10$uVRx4ogFBi0owMljpvEiLONnd9wWOMtrpgVwqw2Mw8.aNm06yEU1u |
| 2 | 17ETCS002122 | $2b$10$ATp9qxsPWbsOUXDAB1YvK.yTLi4GK1mzpIHBCfS0CQwtXLU/52Pk2 |
| 3 | 17ETCS002168 | $2b$10$3cfBMD3yRi3YJk.fFGrNY.Yx1RRonj4z2cQG0e2fgZ78yNaqRxFkC |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from STUDENT;
+-----+-----+-----+-----+-----+-----+
| id | reg_no | name | department | course | contact_no |
+-----+-----+-----+-----+-----+-----+
| 2 | 17ETCS002122 | Prachi Poddar | CSE | B.Tech | 9856523658 |
| 1 | 17ETCS002159 | Satyajit Ghana | CSE | B.Tech | 7892137665 |
| 3 | 17ETCS002168 | Shikhar Singh | CSE | B.Tech | 9852145896 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

## 6. Analysis and Discussions

The database schema was designed as per the ER diagram designed in the Laboratory 1 and 2. The data or information for the database are stored in tables. Tables are uniquely identified by their names and are comprised of columns and rows. All the seven tables represent seven different entities and their columns represent attributes of the entities. Each table has column "id" as their primary key.

To store users' password hashing is used. Hashing is a one-way encryption that means you cannot get the original text back from the hash because in information security, passwords are recommended to be stored in a hashed format so applications/systems can verify if the correct password is entered without them storing your password. This makes it harder to steal. Because what you don't have has less likelihood of being stolen.

## 7. Conclusions

SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. The standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" were used to accomplish everything that was needed to create the database.

## 8. Comments

### a. Limitations of Experiments

One limitation of SQL is that relations must have a fixed set of columns. This is a frequent annoyance of software developers, and drives the demand for non-relational databases.

SQL allows user to access the data stored in a relational database (your typical RDBMS) or even flat files or hadoop or MongoDB(when using tools like Apache Drill or Hive). Even though SQL concept and syntax remains same across platforms and tools, implementation and limitations of each platform are sometimes very different. The problem increases in scale when performance tuning is concerned.

### b. Limitations of Results

None

### c. Learning happened

We learnt how to create tables, add data, create database, modify data, using MySQL, and also how to implement a given schema/er diagram in MySQL

Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	

## Laboratory 4

Title of the Laboratory Exercise: HTML and JSP form to implement the functional requirement (Example: Registration page)

### 1. Introduction and Purpose of Experiment

Students will learn to use will HTML and HTML Frames in html platform.

### 2. Aim and Objectives

Aim

Objectives

### 3. Experimental Procedure

### 4. Calculations/Computations/Algorithms

#### register.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
<html>

<head>
    <title>Register</title>
</head>

<body style="text-align:center; margin:auto">
    <h1>Register</h1>
    <br />
    <center>
        <form align="center" action="register" method="post">
            <table>
                <tr>
                    <td>username</td>
                    <td><input type="text" name="username" /></td>
                </tr>
                <tr>
                    <td>password</td>
                    <td><input type="password" name="password" /></td>
                </tr>
            </table>
        </form>
    </center>
</body>
```



```
<tr>
  <td>full name</td>
  <td><input type="text" name="fullname" /></td>
</tr>
<tr>
  <td>usn no</td>
  <td><input type="text" name="usnno" /></td>
</tr>
<tr>
  <td>dept</td>
  <td>
    <select name="dept">
      <option value="CSE">CSE</option>
      <option value="EEE">EEE</option>
      <option value="ECE">ECE</option>
      <option value="CIVIL">CIVIL</option>
    </select>
  </td>
</tr>
<tr>
  <td>course</td>
  <td>
    <select name="course">
      <option value="B.Tech">B.Tech</option>
      <option value="M.Tech">M.Tech</option>
    </select>
  </td>
</tr>
</table>
<br />
<input type="submit" value="register" />
</form>
</center>
</body>

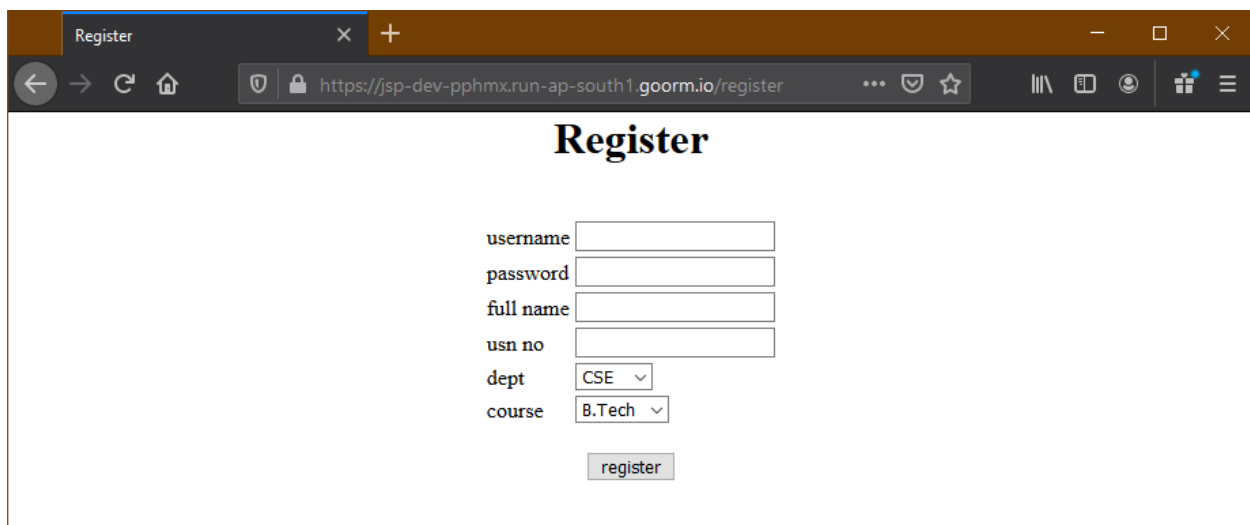
</html>
```

## web.xml

```
<web-app>
  <servlet>
    <servlet-name>index</servlet-name>
    <jsp-file>/index.jsp</jsp-file>
  </servlet>
  <servlet-mapping>
    <servlet-name>index</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
```

```
<servlet>
  <servlet-name>register</servlet-name>
  <jsp-file>/register.jsp</jsp-file>
</servlet>
<servlet-mapping>
  <servlet-name>register</servlet-name>
  <url-pattern>/register</url-pattern>
</servlet-mapping>
</web-app>
```

## 5. Presentation of Results



The screenshot shows a web browser window with the title 'Register'. The address bar displays the URL 'https://jsp-dev-pphmx.run-ap-south1.goorm.io/register'. The main content area features a form titled 'Register' with the following fields:

- username:
- password:
- full name:
- usn no:
- dept:
- course:

A 'register' button is located below the form fields.

## 6. Analysis and Discussions

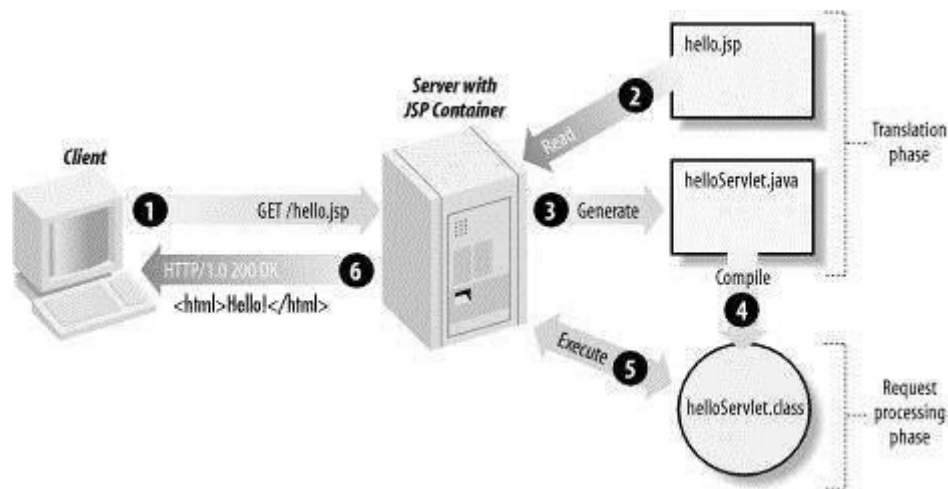
Frameset is basically a tag in HTML. It is used to create several frames in single web page and each of those frames can contain separate HTML documents. The tag is used to divide the browser window in several frames. Frames divide browser window in frames just like rows and columns in tables. If you are comfortable with the concept of creating tables in HTML then frames won't be difficult. HTML forms the basis of web development. If you see yourself as a builder of websites with a solid structure then an in-depth knowledge of HTML is indispensable.

The Hyper Text Mark-Up Language, also known as HTML is a mark-up language used to build customized websites. The HTML language is made up of tags that are used to build web pages from scratch. HTML is

an essential but valuable skill. Beginners tend to neglect the importance of this language and this might be one of the biggest mistakes they commit in their formative years as developers.

HTML frames allow authors to present documents in multiple views, which may be independent windows or subwindows. Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced. For example, within the same window, one frame might display a static banner, a second a navigation menu, and a third the main document that can be scrolled through or replaced by navigating in the second frame.

## 7. Conclusions



JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet.

## 8. Comments

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages:

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back* button might not work as the user hopes.
- There are still few browsers that do not support frame technology.

### a. Limitations of Experiments

The JSP developer needs to know Java. Again, one developer's asset is another's liability. Whereas Java is certainly more full-featured and flexible than other page scripting languages, no one can argue that the learning curve for Java is far steeper than other scripting languages. If you already know Java this is not an issue.

JSP pages must be compiled on the server when first accessed. This initial compilation produces a noticeable delay when accessing the JSP page for the first time.

The developer may compile JSP pages and place them on the server in compiled form (as one or more class files) to speed up the initial page access. The JSP developer may need to bring down the server to make the changed classes corresponding to the changed JSP page.

b. Limitations of Results

None

c. Learning happened

We learnt how to make the Registration Page in JSP, along with HTML forms.

d. Recommendations

None

Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	

## Laboratory 5

Title of the Laboratory Exercise: HTML and JSP form to implement the functional requirement (Example: Login page and search page)

### 1. Introduction and Purpose of Experiment

Students will learn to use JSP scriptlet, expression declaration and other JSP actions to

Implement different use cases

### 2. Aim and Objectives

Aim

Objectives

At the end of this lab, the student will be able to

### 3. Experimental Procedure

### 4. Calculations/Computations/Algorithms

login.jsp

```
<%@ page contentType = "text/html; charset=utf-8" %>
<html>
  <head>
    <title>Login</title>
  </head>

  <body style="text-align:center; margin:auto">
    <h1>Login to RUAS LMS</h1>
    <br/>
    <center>
      <form align="center" action="login" method="post">
        <table>
          <tr>
            <td>username</td>
            <td><input type="text" name="username" /></td>
          </tr>
          <tr>
```

```
                <td>password</td>
                <td><input type="password" name="password" /></td>
            </tr>
        </table>
        <br/>
        <input type="submit" value="login" />
    </form>
</center>
</body>
</html>
```

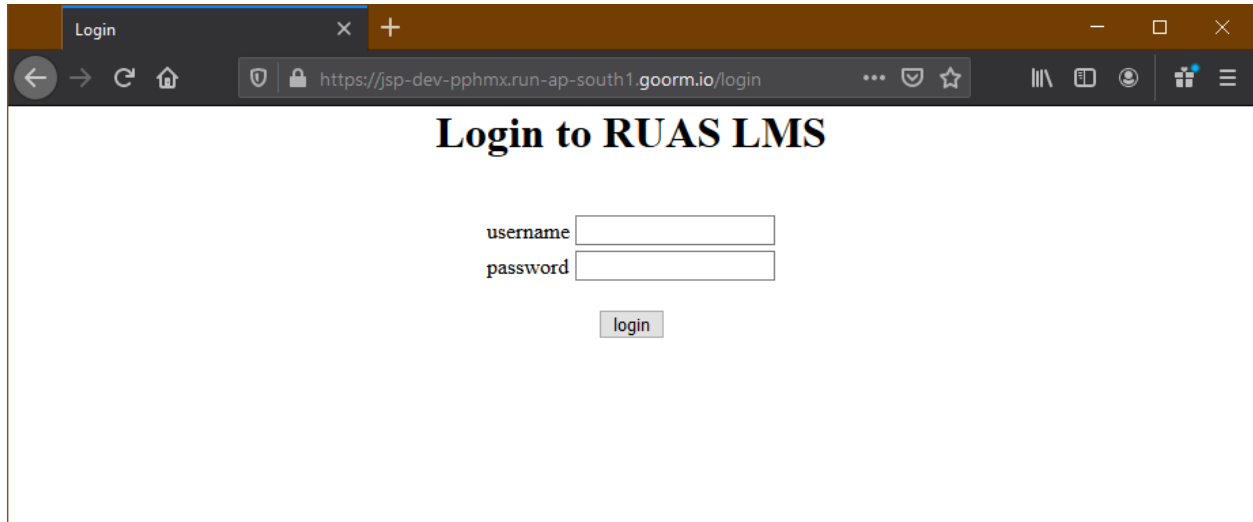
### projects.jsp

```
<%@ page contentType = "text/html; charset=utf-8" %>
<html>
    <head>
        <title>Project</title>
    </head>

    <body style="text-align:center; margin:auto">
        <h1>Search for Project Details</h1>
        <br/>
        <center>
            <form align="center" action="search" method="get">
                <table>
                    <tr>
                        <td>usn no</td>
                        <td>
                            <input type="text" name="usnno" />
                        </td>
                    </tr>
                </table>
            </form>
        </center>
    </body>
</html>
```

## 5. Presentation of Results

## Login Page



Login

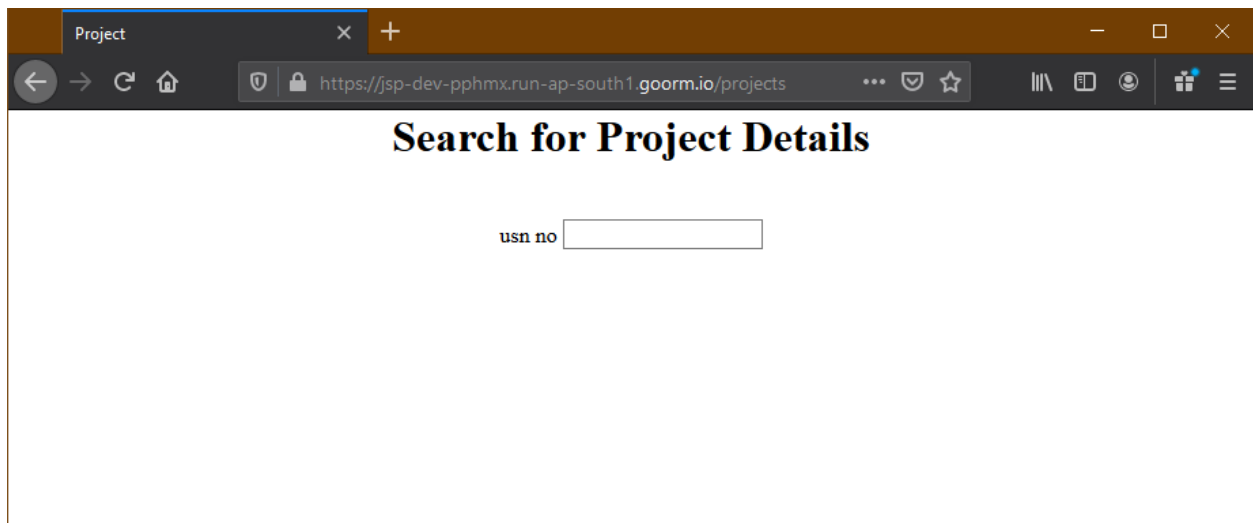
https://jsp-dev-pphmx.run-ap-south1.goorm.io/login

## Login to RUAS LMS

username

password

## Search Page



Project

https://jsp-dev-pphmx.run-ap-south1.goorm.io/projects

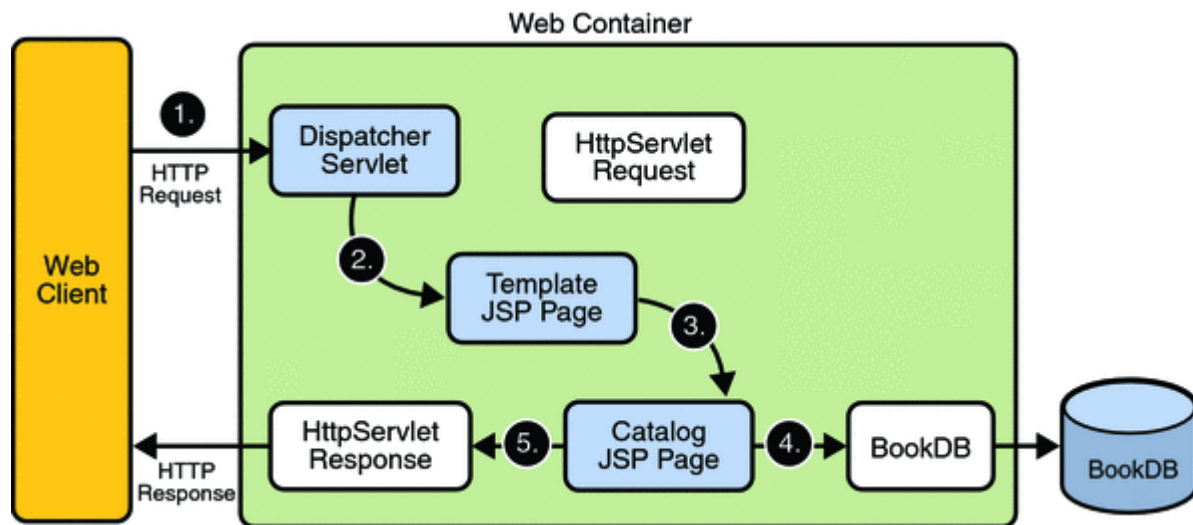
## Search for Project Details

usn no

## 6. Analysis and Discussions

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

A JSP page is a text document that contains two types of text: static data, which can be expressed in any text-based format (such as HTML, SVG, WML and XML), and JSP elements, which construct dynamic content. It enables you to write dynamic, data-driven pages for your Java web applications. JSP is built on top of the Java Servlet specification. The two technologies typically work together, especially in older Java web applications. From a coding perspective, the most obvious difference between them is that with servlets you write Java code and then embed client-side markup (like HTML) into that code, whereas with JSP you start with the client-side script or markup, then embed JSP tags to connect your page to the Java backend.



## 7. Conclusions

HTML is an essential but valuable skill. Beginners tend to neglect the importance of this language and this might be one of the biggest mistakes they commit in their formative years as developers. It is an easy to learn language and is responsible for building the structure of most websites you see on the world wide web. HTML is often perceived as a complex language but it's really not.

JSP pages must be deployed inside a Java servlet container. In order to deploy a Java web application based on JSP and servlets, you will package your .jsp files, Java code, and application metadata in a .war file, which is a simple .zip file with a conventional structure for web applications.



## 8. Comments

## a. Limitations of Experiments

Because JSP pages are translated, and then compiled into Java servlets, errors that creep in your pages are rarely seen as errors arising from the coding of JSP pages.

The JSP developer would need access to the generated source to properly diagnose the error. Of course, generated code is rarely a thing of beauty, and often, not easily understood.

Because JSP pages are translated into class files, the server has to store the resultant class files with the JSP pages.

## b. Limitations of Results

None

## c. Learning happened

We learnt how to crate login and search pages with JSP and HTML

## d. Recommendations

None

Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	

## Laboratory 6

Title of the Laboratory Exercise: To implement Server side web component (Servlet) to handle client request

### 1. Introduction and Purpose of Experiment

Students will learn to implement servlet to handle client submitted request

### 2. Aim and Objectives

Aim

Objectives

### 2. Experimental Procedure

### 3. Calculations/Computations/Algorithms

#### LoginServlet.jsp

```
package webarch.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
        rd.include(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        pw.println("<script type=\"text/javascript\">");
```

```

        pw.println("alert('Entered Values: username [ "+username+ " ], password [ "+password+
" ] ');");
        pw.println("</script>");

        // if login success then redirect to projects page
        // RequestDispatcher rd = request.getRequestDispatcher("projects.jsp");

        RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
        rd.include(request, response);
    }
}

```

### RegisterServlet

```

package webarch.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.*;
import javax.servlet.http.*;

public class RegisterServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        RequestDispatcher rd = request.getRequestDispatcher("register.jsp");
        rd.include(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String fullname = request.getParameter("fullname");
        String usnno = request.getParameter("usnno");
        String dept = request.getParameter("dept");
        String course = request.getParameter("course");

        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        pw.println("<script type='text/javascript'>");
        pw.println("alert('Entered Values: username [ "+username+ " ], password [ "+password+
], fullname [ "+fullname+ " ], usnno [ "+usnno+ " ], dept [ "+dept+ " ], course [ "+course+ " ] ');");
        pw.println("</script>");

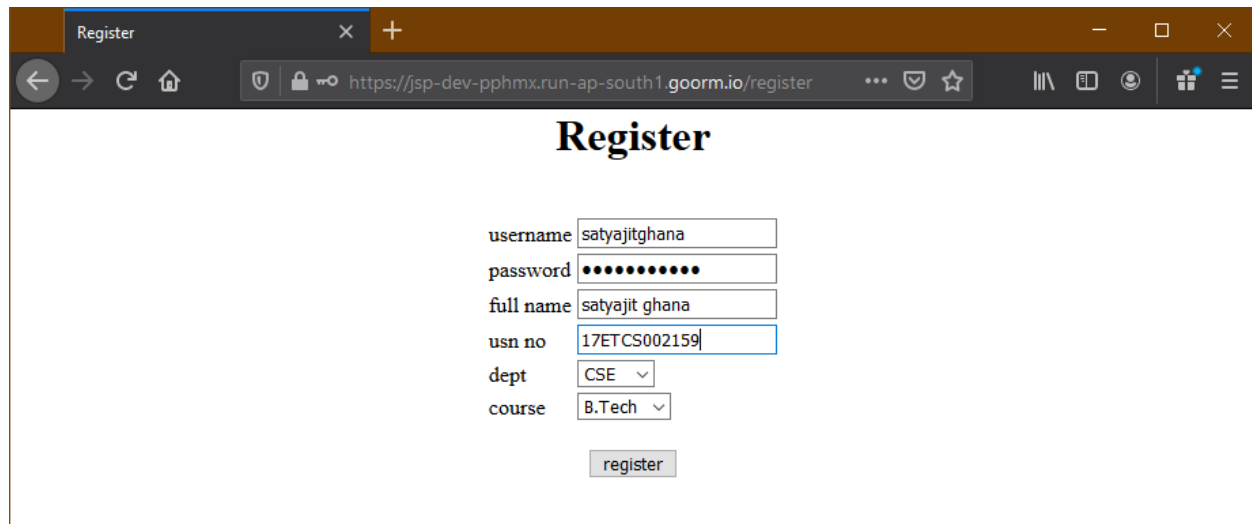
        // if register success then send to login page
        // RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
    }
}

```

```
RequestDispatcher rd = request.getRequestDispatcher("register.jsp");  
rd.include(request, response);  
}  
}
```

#### 4. Presentation of Results

##### Filling up the Register Form

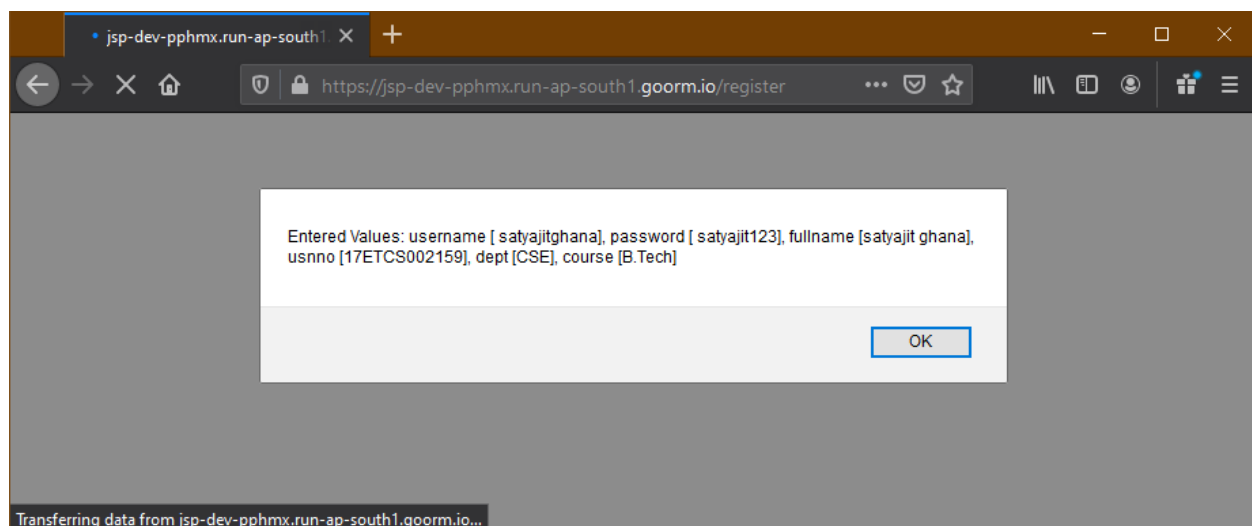


The screenshot shows a web browser window with the title "Register". The address bar shows the URL "https://jsp-dev-pphmx.run-ap-south1.goorm.io/register". The form contains the following fields and values:

Field	Value
username	satyajitghana
password	satyajit123
full name	satyajit ghana
usn no	17ETCS002159
dept	CSE
course	B.Tech

Below the form fields is a "register" button.

##### Entered details



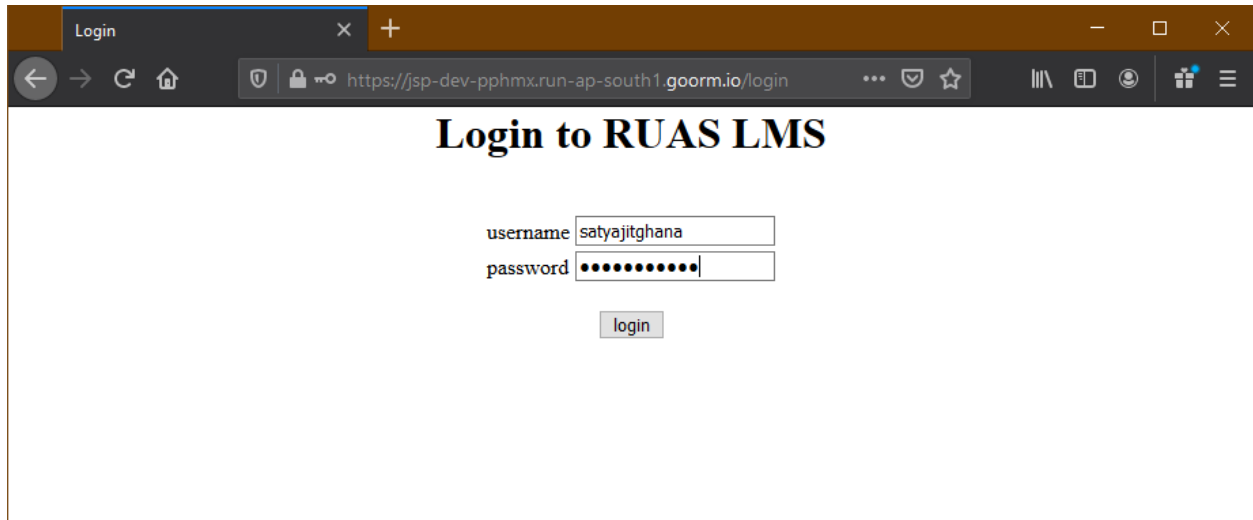
The screenshot shows a confirmation dialog box with the following text:

Entered Values: username [ satyajitghana], password [ satyajit123], fullname [satyajit ghana], usnno [17ETCS002159], dept [CSE], course [B.Tech]

There is an "OK" button at the bottom right of the dialog box.

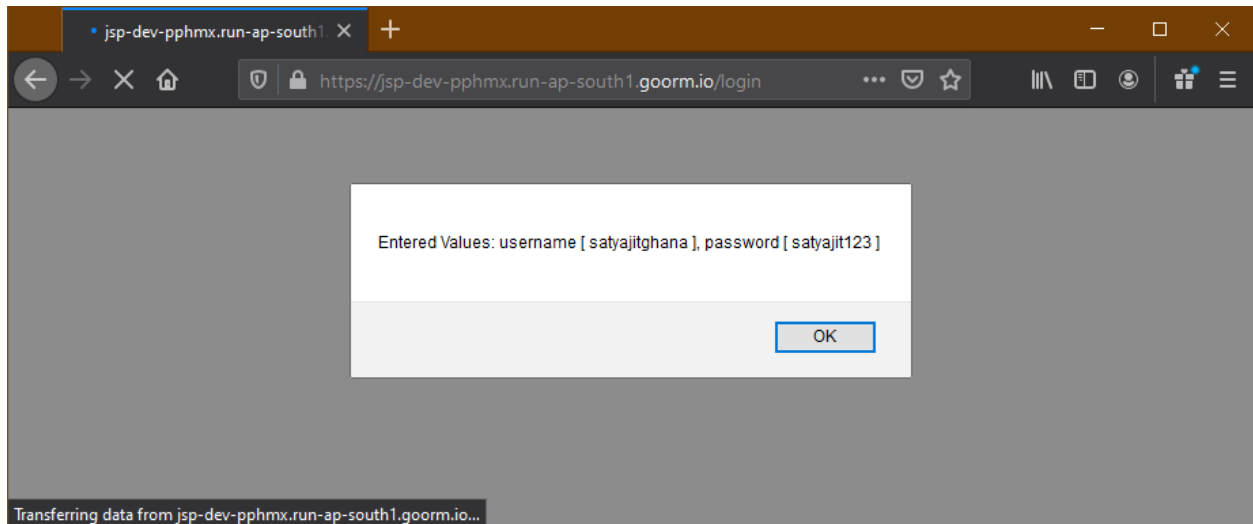
Transferring data from jsp-dev-pphmx.run-ap-south1.goorm.io...

## Filling up the Login Form



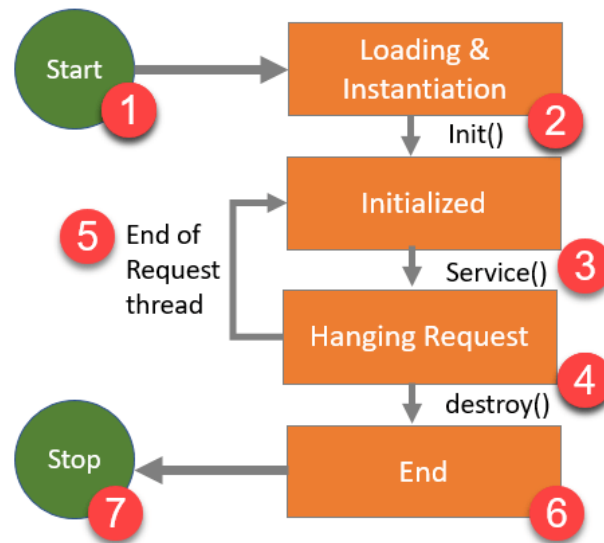
A screenshot of a web browser window showing a login form titled "Login to RUAS LMS". The browser's address bar displays the URL "https://jsp-dev-pphmx.run-ap-south1.goorm.io/login". The form contains two input fields: "username" with the value "satyajitghana" and "password" with masked characters ".....". Below the fields is a "login" button.

## Entered Details



A screenshot of a web browser window showing a dialog box titled "Entered Values: username [ satyajitghana ], password [ satyajit123 ]". The dialog box has an "OK" button. The browser's address bar displays the URL "https://jsp-dev-pphmx.run-ap-south1.goorm.io/login". A status bar at the bottom of the browser window shows the text "Transferring data from jsp-dev-pphmx.run-ap-south1.goorm.io..."

## 5. Analysis and Discussions



1) Start: Execution of servlet begins.

2) Loading & instantiation void `init()`: It is called when servlet is first loaded. This method lets you initialize servlet.

3) Initialized void `service()`: The purpose of this method is to serve a request. You can call it as many times as you like.

4) Handling request and destroying servlet: Java application must be first determined what code is needed to execute the request URL to provide a response. To destroy servlet Void destroy method is used at the end of servlet life cycle.

5) End of Request Thread: When `service()` finishes its task, either the thread ends or returns to the thread pool that is managed by servlet container.

6) End: Servlet lifecycle finishes.

7): Stop: Servlet stop executing.

## 6. Conclusions

### Servlet Advantage

1. Servlets provide a way to generate dynamic documents that is both easier to write and faster to run.
2. provide all the powerfull features of JAVA, such as Exception handling and garbage collection.
3. Servlet enables easy portability across Web Servers.
4. Servlet can communicate with different servlet and servers.
5. Since all web applications are stateless protocol, servlet uses its own API to maintain session

### Servlet Disadvantage

1. Designing in servlet is difficult and slows down the application.
2. Writing complex business logic makes the application difficult to understand.
3. You need a Java Runtime Environment on the server to run servlets. CGI is a completely language independent protocol, so you can write CGIs in whatever languages you have available (including Java if you want to).

## 7. Comments

### a. Limitations of Experiments

It is hard to trace JSP pages error because JSP pages are translated to servlet. As JSP output is HTML, it is not rich in features. It is very hard to debug or trace errors because JSP pages are first translated into servlets before the compilation process. JSP pages require more disk space and time to hold JSP pages as they are compiled on the server.

### b. Limitations of Results

None

### c. Learning happened

We learnt how to connect a servlet to a JSP page, and use the request response model

### d. Recommendations

None

Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	



## Laboratory 7

Title of the Laboratory Exercise: To integrate database using JDBC to web application

### 1. Introduction and Purpose of Experiment

Students will learn to implement JDBC component for all the functional requirements identified.

### 2. Aim and Objectives

Aim

### 3. Experimental Procedure

### 4. Calculations/Computations/Algorithms

UserDao.java

```
package webarch.dao;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import webarch.db.DBConnection;

public class UserDao {

    public static String getMd5(String input) {
        try {

            // Static getInstance method is called with hashing MD5
            MessageDigest md = MessageDigest.getInstance("MD5");

            // digest() method is called to calculate message digest
            // of an input digest() return array of byte
            byte[] messageDigest = md.digest(input.getBytes());

            // Convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
```

```
        String hashtext = no.toString(16);
        while (hashtext.length() < 32) {
            hashtext = "0" + hashtext;
        }
        return hashtext;
    }

    // For specifying wrong message digest algorithms
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}

public static boolean registerUser(String username, String password, String fullname, String usnno, String dept, String course) {
    Connection conn = DBConnection.getDbConnection();

    try {
        PreparedStatement stmt = conn.prepareStatement("INSERT INTO `STUDENT_LOGIN` (`user_name`, `hashed_password`) VALUES(?, ?)", Statement.RETURN_GENERATED_KEYS);
        stmt.setString(1, username);
        stmt.setString(2, getMd5(password));

        stmt.executeUpdate();

        ResultSet rs = stmt.getGeneratedKeys();
        if (rs.next()) {
            Integer id = rs.getInt(1);

            PreparedStatement stmt2 = conn.prepareStatement("INSERT INTO STUDENT( id, reg_no, name, department, course, contact_no ) VALUES (?, ?, ?, ?, ?, ?)");
            stmt2.setInt(1, id);
            stmt2.setString(2, usnno);
            stmt2.setString(3, fullname);
            stmt2.setString(4, dept);
            stmt2.setString(5, course);
            stmt2.setString(6, "9999999999"); // hardcoded value for now

            int count = stmt2.executeUpdate();

            if (count > 0)
                return true;
            else
                return false;
        }
        rs.close();
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }

    return false;
}
```

```
}
```

## DBConnection.java

```
package webarch.db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

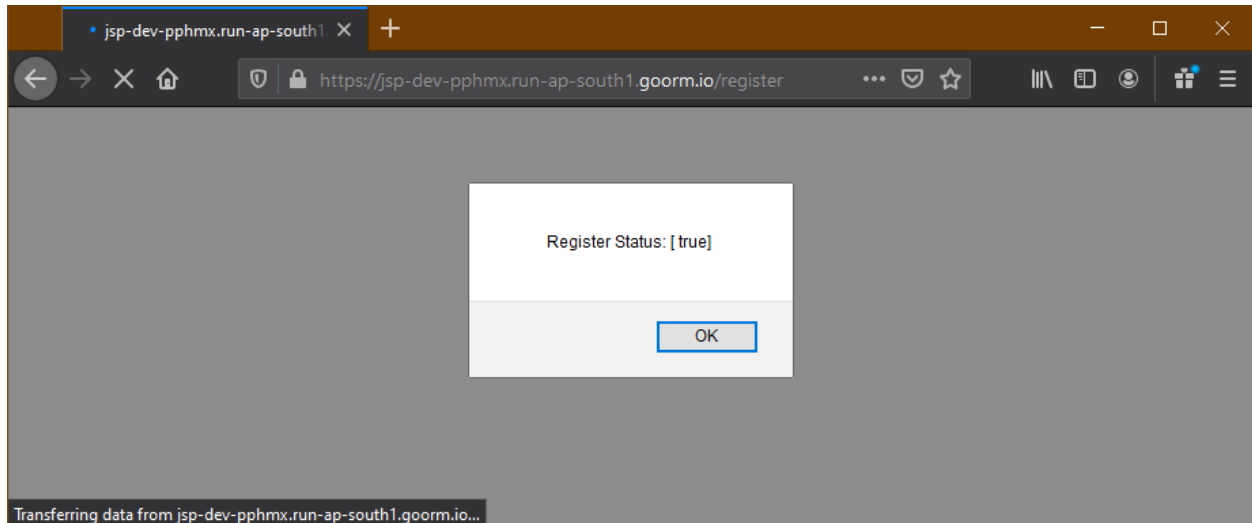
public class DBConnection {
    private static Connection conn;

    public static Connection getDbConnection() {
        if (conn == null) {
            try {
                Class.forName("com.mysql.jdbc.Driver");
                conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/webarch", "root", "");
            } catch (SQLException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }

        return conn;
    }
}
```

## 5. Presentation of Results

### Registration of User



User successfully added in the database

```
mysql> select * from STUDENT_LOGIN where user_name="satyajitghana";
+----+-----+-----+
| id | user_name | hashed_password |
+----+-----+-----+
| 11 | satyajitghana | e2d60cfcaaf71ad64ba36fbd103374c5 |
+----+-----+-----+
1 row in set (0.00 sec)

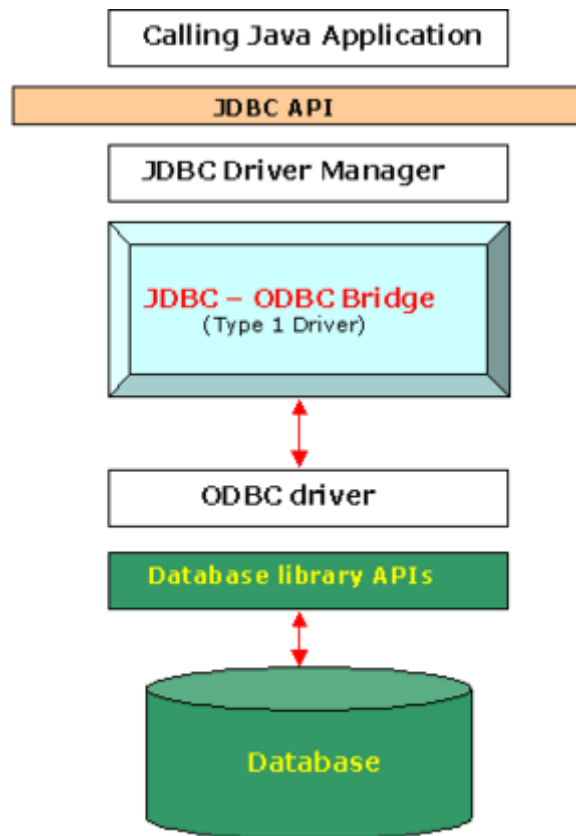
mysql> select * from STUDENT where id=11;
+----+-----+-----+-----+-----+-----+
| id | reg_no | name | department | course | contact_no |
+----+-----+-----+-----+-----+-----+
| 11 | 17CSET002159 | satyajit ghana | CSE | B.Tech | 9999999999 |
+----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> 
```

## 6. Analysis and Discussions

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)



## 7. Conclusions

JDBC can be used with Java to Connect to an existing DB by,

```
public Connection getConnection() throws SQLException {  
  
    Connection conn = null;  
    Properties connectionProps = new Properties();  
    connectionProps.put("user", this.userName);  
    connectionProps.put("password", this.password);  
  
    if (this.dbms.equals("mysql")) {  
        conn = DriverManager.getConnection(  
            "jdbc:" + this.dbms + "://" +  
            this.serverName +  
            ":" + this.portNumber + "/",  
            connectionProps);  
    } else if (this.dbms.equals("derby")) {
```

```
        conn = DriverManager.getConnection(
            "jdbc:" + this.dbms + ":" +
            this.dbName +
            ";create=true",
            connectionProps);
    }
    System.out.println("Connected to database");
    return conn;
}
```

## 8. Comments

### a. Limitations of Experiments

Performance is degraded since the JDBC call goes through the bridge to the ODBC driver then to the native database connectivity interface. The results are then sent back through the reverse process

### b. Limitations of Results

None

### c. Learning happened

We learnt how to connect to MySQL DB using JDBC from Java.

### d. Recommendations

None

Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	

## Laboratory 8

Title of the Laboratory Exercise: Client Side validation using JavaScript

### 1. Introduction and Purpose of Experiment

Students will learn to implement Client Side validation using JavaScript for all the functional requirements identified.

### 2. Aim and Objectives

Aim

Design and implement the Business Logic for the given scenario using JAX-RS services.

### 3. Objectives

### 4. Experimental Procedure

### 5. Calculations/Computations/Algorithms

register.jsp

```
<%@ page contentType = "text/html; charset=utf-8" %>
<html>
  <head>
    <title>Register</title>

    <script>

      function validateForm() {
        var username = document.regform.username.value;
        var password = document.regform.password.value;
        var fullname = document.regform.fullname.value;
        var usnno    = document.regform.usnno.value;

        if (username == "") {
          alert("username cannot be blank");
          return false;
        } else if (password == "") {
          alert("password cannot be blank");
          return false;
        } else if (fullname == "") {
          alert("fullname cannot be blank");
          return false;
        } else if (usnno == "") {
```

```
        alert("usnno cannot be blank");
        return false;
    } else if (password.length < 6) {
        alert("password must be at least 6 characters long");
        return false;
    }

    return true;
}

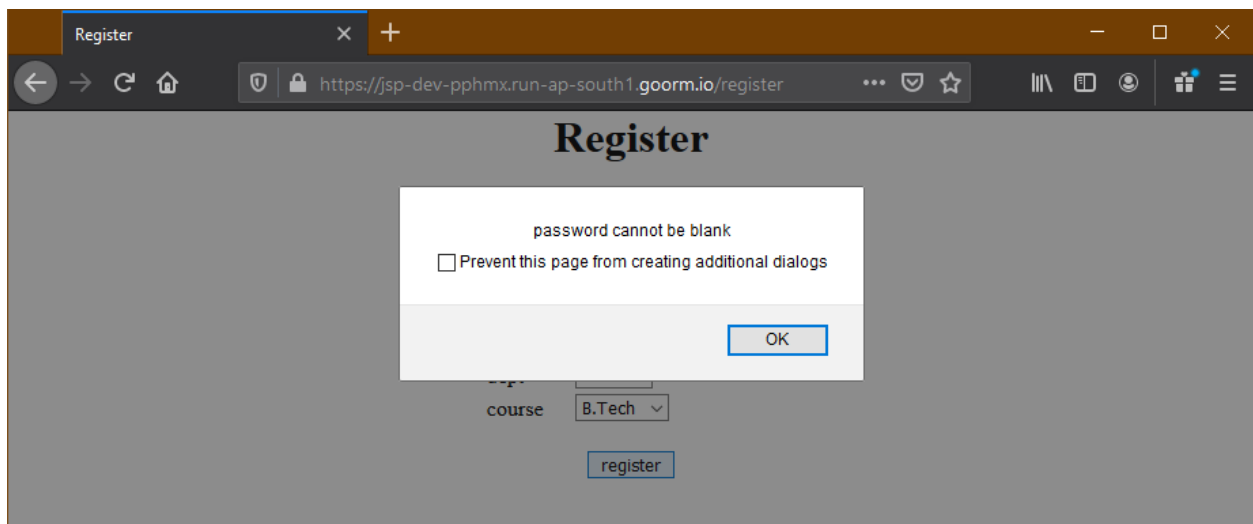
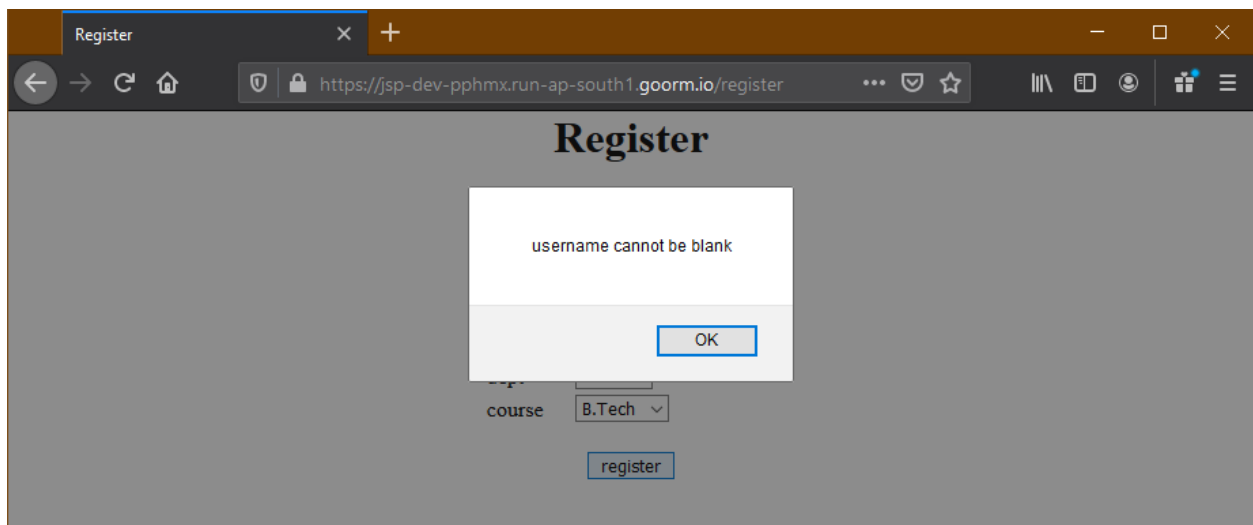
</script>
</head>

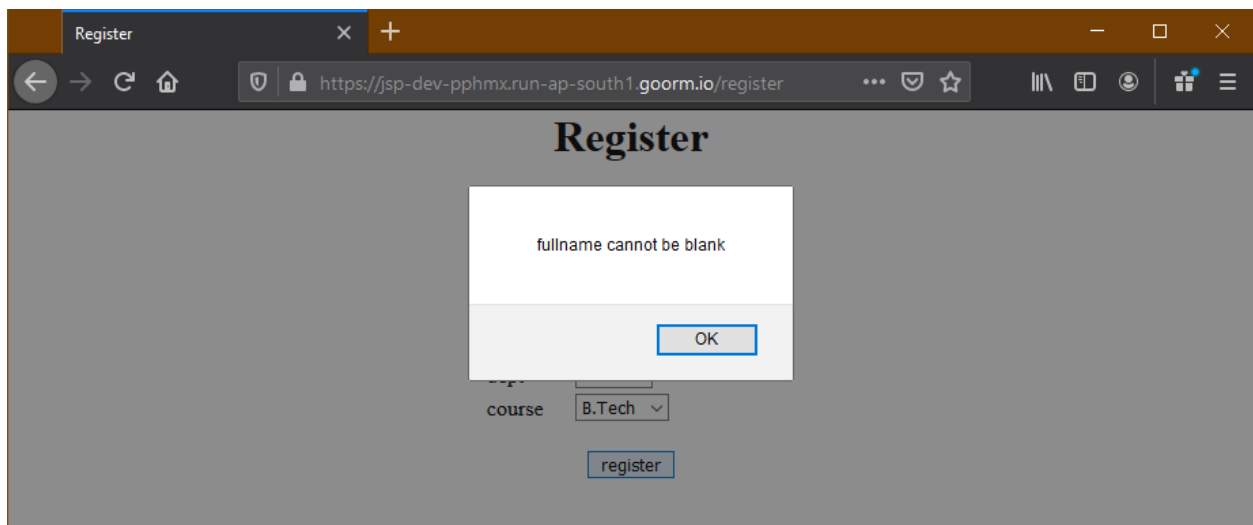
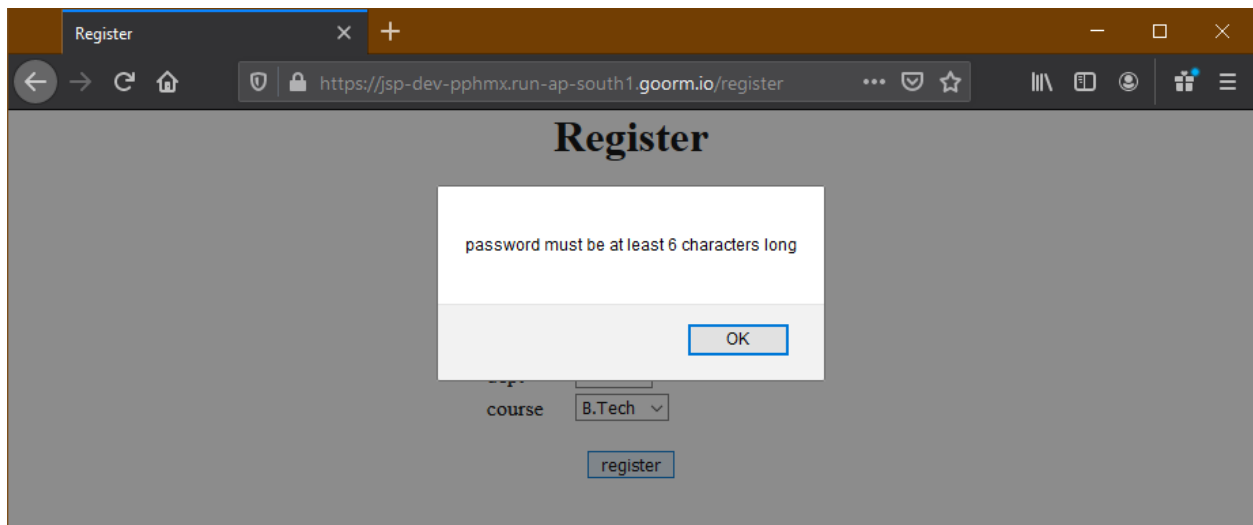
<body style="text-align:center; margin:auto">
    <h1>Register</h1>
    <br/>
    <center>
        <form align="center" name="regform" action="register" method="post" onsubmit="return validateForm();">
            <table>
                <tr>
                    <td>username</td>
                    <td><input type="text" name="username" /></td>
                </tr>
                <tr>
                    <td>password</td>
                    <td><input type="password" name="password" /></td>
                </tr>
                <tr>
                    <td>full name</td>
                    <td><input type="text" name="fullname" /></td>
                </tr>
                <tr>
                    <td>usn no</td>
                    <td><input type="text" name="usnno" /></td>
                </tr>
                <tr>
                    <td>dept</td>
                    <td>
                        <select name="dept">
                            <option value="CSE">CSE</option>
                            <option value="EEE">EEE</option>
                            <option value="ECE">ECE</option>
                            <option value="CIVIL">CIVIL</option>
                        </select>
                    </td>
                </tr>
                <tr>
                    <td>course</td>
                    <td>
                        <select name="course">
```



```
        <option value="B.Tech">B.Tech</option>
        <option value="M.Tech">M.Tech</option>
    </select>
</td>
</tr>
</table>
<br/>
<input type="submit" value="register" />
</form>
</center>
</body>
</html>
```

## 6. Presentation of Results





## 7. Analysis and Discussions

JavaScript is a client-side scripting language, which means that the script runs on the your viewer or client's browser and not on the server. JavaScript extends the web functionality with a combination of HTML and server side languages. JavaScript validation is one of the common features used with the language.

A basic null validation code block uses an 'if' condition to check whether a specific HTML element returns a value. The following example checks to see if the user entered any value at all:

```
if( document.Form_validate.Name.value == "" )  
{
```

```
alert( "Please provide your name!" );
document.Form_validate.Name.focus() ;
return false;
}
```

Data that you can validate using JavaScript includes:

- Required fields
- Username
- Password
- Email address
- Phone number

## 8. Conclusions

Client-side validation is an initial check and an important feature of good user experience; by catching invalid data on the client-side, the user can fix it straight away. If it gets to the server and is then rejected, a noticeable delay is caused by a round trip to the server and then back to the client-side to tell the user to fix their data.

### Types of Client-Side Validation

- Built-in form validation uses HTML5 form validation features, which we've discussed in many places throughout this module. This validation generally doesn't require much JavaScript. Built-in form validation has better performance than JavaScript, but it is not as customizable as JavaScript validation.
- JavaScript validation is coded using JavaScript. This validation is completely customizable, but you need to create it all (or use a library).

## 9. Comments

### a. Limitations of Experiments

The disadvantage, however, is big: client-side support for scripting languages varies wildly, with some browsers supporting scripts very well, others supporting bits and pieces, and others supporting nothing

at all. Furthermore, wily users can disable your client-side checking in order to feed you bad data - if you rely solely on client-side checking, you are bound to get hacked eventually.

b. Limitations of Results

Most client-side validation is accomplished using the special "onSubmit" event of a form, which allows you to run JavaScript code to handle form validation when your visitor attempts to submit the form. If you return false from your code in onSubmit, web browsers will not proceed with submitting the form, which allows you to prompt visitors to correct any errors before submission.

c. Learning happened

We Learnt how to do Client-Side Validation using JavaScript

d. Recommendations

None

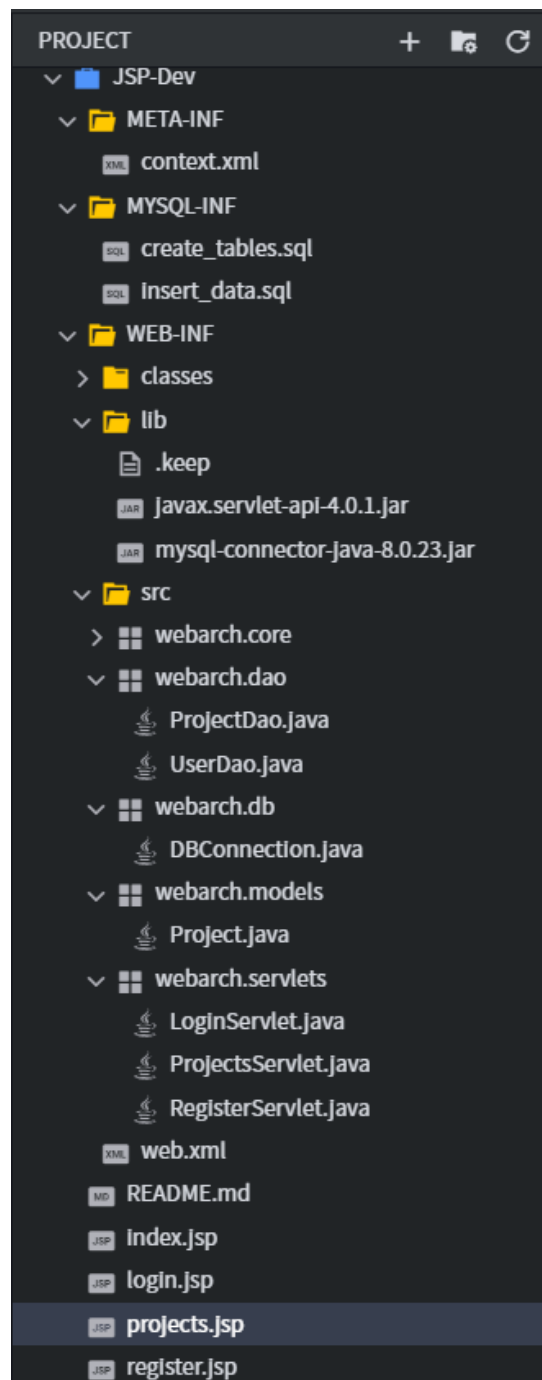
Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	

## Laboratory 9

Title of the Laboratory Exercise: Functionality implementation

1. Introduction and Purpose of Experiment
2. Aim and Objectives  
Aim
3. Experimental Procedure
4. Calculations/Computations/Algorithms

## Project Structure



**web.xml**

```
<web-app>
    <servlet>
        <servlet-name>index</servlet-name>
        <jsp-file>/index.jsp</jsp-file>
    </servlet>
    <servlet-mapping>
        <servlet-name>index</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>register</servlet-name>
        <servlet-class>webarch.servlets.RegisterServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>register</servlet-name>
        <url-pattern>/register</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>login</servlet-name>
        <servlet-class>webarch.servlets.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>login</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>projects</servlet-name>
        <servlet-class>webarch.servlets.ProjectsServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>projects</servlet-name>
        <url-pattern>/projects</url-pattern>
    </servlet-mapping>
</web-app>
```

**webarch.servlets.LoginServlet.java**

```
package webarch.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import webarch.dao.UserDao;
```

```
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
        rd.include(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        boolean loginStatus = UserDao.loginUser(username, password);

        if (loginStatus == true) {
            response.sendRedirect("/projects");
        } else {
            response.setContentType("text/html");
            PrintWriter pw = response.getWriter();
            pw.println("<script type=\"text/javascript\">");
            pw.println("alert('Username or Password incorrect ');");
            pw.println("</script>");
            RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
            rd.include(request, response);
        }
    }
}
```

**webarch.servlets.ProjectsServlet.java**

```
package webarch.servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import webarch.dao.ProjectDao;
import webarch.models.Project;

import javax.servlet.*;
import javax.servlet.http.*;
```



```
public class ProjectsServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        List<Project> projects;

        if (request.getParameterMap().containsKey("projectname")) {
            String projectname = request.getParameter("projectname");
            projects = ProjectDao.findProjects(projectname);
        } else {
            projects = new ArrayList<>();
        }

        request.setAttribute("projects", projects);

        RequestDispatcher rd = request.getRequestDispatcher("projects.jsp");
        rd.include(request, response);

    }
}
```

**webarch.servlets.RegisterServlet.java**

```
package webarch.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import webarch.dao.UserDao;

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

public class RegisterServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        RequestDispatcher rd = request.getRequestDispatcher("register.jsp");
        rd.include(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String fullname = request.getParameter("fullname");
    }
}
```

```
String usnno = request.getParameter("usnno");
String dept = request.getParameter("dept");
String course = request.getParameter("course");

boolean registerStatus = UserDao.registerUser(username, password, fullname, usnno, dept, course);

response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<script type=\"text/javascript\">");
pw.println("alert('Register Status: [ "+registerStatus+" ] ');");
pw.println("</script>");

// if register success then send to login page
// RequestDispatcher rd = request.getRequestDispatcher("login.jsp");

RequestDispatcher rd = request.getRequestDispatcher("register.jsp");
rd.include(request, response);
}
}
```

**webarch.models.Project.java**

```
package webarch.models;

public class Project {
    private Integer id;
    private String projectLeaderRegno;
    private String projectName;
    private String mentorName;
    private String department;
    private String category;

    public Project(Integer id, String projectLeaderRegno, String projectName, String mentorName, String department, String category) {
        this.id = id;
        this.projectLeaderRegno = projectLeaderRegno;
        this.projectName = projectName;
        this.mentorName = mentorName;
        this.department = department;
        this.category = category;
    }

    public Integer getId() {
        return this.id;
    }

    public String getProjectLeaderRegno() {
        return this.projectLeaderRegno;
    }
}
```

```
public String getProjectName() {  
    return this.projectName;  
}  
  
public String getMentorName() {  
    return this.mentorName;  
}  
  
public String getDepartment() {  
    return this.department;  
}  
  
public String getCategory() {  
    return this.category;  
}  
}
```

**webarch.db.DBConnection.java**

```
package webarch.db;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class DBConnection {  
    private static Connection conn;  
  
    public static Connection getDbConnection() {  
        if (conn == null) {  
            try {  
                Class.forName("com.mysql.jdbc.Driver");  
                conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/webarch", "root", "");  
            } catch (SQLException e) {  
                e.printStackTrace();  
            } catch (ClassNotFoundException e) {  
                e.printStackTrace();  
            }  
        }  
  
        return conn;  
    }  
}
```

**webarch.dao.ProjectDao.java**

```
package webarch.dao;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import webarch.db.DBConnection;
import webarch.models.Project;

public class ProjectDao {
    public static List<Project> findProjects(String projectName) {
        Connection conn = DBConnection.getDbConnection();

        List<Project> projects = new ArrayList<>();

        try {

            PreparedStatement stmt = conn.prepareStatement("SELECT * FROM PROJEKT WHERE project_name LIKE ?");

            stmt.setString(1, "%" + projectName + "%");

            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                projects.add(new Project(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getString(5), rs.getString(6)));
            }

            return projects;

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return projects;
    }
}
```

**webarch.dao.UserDao.java**

```
package webarch.dao;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import java.sql.Statement;

import webarch.db.DBConnection;

public class UserDao {

    public static String getMd5(String input) {
        try {

            // Static getInstance method is called with hashing MD5
            MessageDigest md = MessageDigest.getInstance("MD5");

            // digest() method is called to calculate message digest
            // of an input digest() return array of byte
            byte[] messageDigest = md.digest(input.getBytes());

            // Convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }

        // For specifying wrong message digest algorithms
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    public static boolean loginUser(String username, String password) {
        Connection conn = DBConnection.getConnection();

        try {
            PreparedStatement stmt = conn.prepareStatement("SELECT id, hashed_password FROM `STUDENT_LOGIN` WHERE `user_name` = ? LIMIT 1");
            stmt.setString(1, username);

            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                String hashedPassword = rs.getString(2);

                if (hashedPassword.equals(getMd5(password))) {

                    System.out.println("Login Success for User: " + username);

                    return true;
                }
            }
        }
    }
}
```

```
    }
}

return false;

} catch (SQLException e) {
    e.printStackTrace();
}

return false;
}

public static boolean registerUser(String username, String password, String fullname, String usnno, String dept, String course ) {
    Connection conn = DBConnection.getDbConnection();

    try {
        PreparedStatement stmt = conn.prepareStatement("INSERT INTO `STUDENT_LOGIN` (`user_name`, `hashed_password`) VALUES(?, ?)", Statement.RETURN_GENERATED_KEYS);
        stmt.setString(1, username);
        stmt.setString(2, getMd5(password));

        stmt.executeUpdate();

        ResultSet rs = stmt.getGeneratedKeys();
        if (rs.next()) {
            Integer id = rs.getInt(1);

            PreparedStatement stmt2 = conn.prepareStatement("INSERT INTO STUDENT( id, reg_no, name, department, course, contact_no ) VALUES (?, ?, ?, ?, ?, ?)");
            stmt2.setInt(1, id);
            stmt2.setString(2, usnno);
            stmt2.setString(3, fullname);
            stmt2.setString(4, dept);
            stmt2.setString(5, course);
            stmt2.setString(6, "9999999999"); // hardcoded value for now

            int count = stmt2.executeUpdate();

            if (count > 0)
                return true;
            else
                return false;
        }
        rs.close();
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }

    return false;
}
```

```
}  
}
```

**register.jsp**

```
<%@ page contentType = "text/html; charset=utf-8" %>  
<html>  
  <head>  
    <title>Register</title>  
  
    <script>  
  
      function validateForm() {  
        var username = document.regform.username.value;  
        var password = document.regform.password.value;  
        var fullname = document.regform.fullname.value;  
        var usnno     = document.regform.usnno.value;  
  
        if (username == "") {  
          alert("username cannot be blank");  
          return false;  
        } else if (password == "") {  
          alert("password cannot be blank");  
          return false;  
        } else if (fullname == "") {  
          alert("fullname cannot be blank");  
          return false;  
        } else if (usnno == "") {  
          alert("usnno cannot be blank");  
          return false;  
        } else if (password.length < 6) {  
          alert("password must be at least 6 characters long");  
          return false;  
        }  
  
        return true;  
      }  
  
    </script>  
  </head>  
  
  <body style="text-align:center; margin:auto">  
    <h1>Register</h1>  
    <br/>  
    <center>  
      <form align="center" name="regform" action="register" method="post" onsubmit="return validateFo  
rm();">  
        <table>  
          <tr>  
            <td>username</td>
```

```

        <td><input type="text" name="username" /></td>
    </tr>
    <tr>
        <td>password</td>
        <td><input type="password" name="password" /></td>
    </tr>
    <tr>
        <td>full name</td>
        <td><input type="text" name="fullname" /></td>
    </tr>
    <tr>
        <td>usn no</td>
        <td><input type="text" name="usnno" /></td>
    </tr>
    <tr>
        <td>dept</td>
        <td>
            <select name="dept">
                <option value="CSE">CSE</option>
                <option value="EEE">EEE</option>
                <option value="ECE">ECE</option>
                <option value="CIVIL">CIVIL</option>
            </select>
        </td>
    </tr>
    <tr>
        <td>course</td>
        <td>
            <select name="course">
                <option value="B.Tech">B.Tech</option>
                <option value="M.Tech">M.Tech</option>
            </select>
        </td>
    </tr>
</table>
<br/>
<input type="submit" value="register" />
</form>
</center>
</body>
</html>

```

**login.jsp**

```

<%@ page contentType = "text/html; charset=utf-8" %>
<html>
    <head>
        <title>Login</title>
    </head>

    <body style="text-align:center; margin:auto">
        <h1>Login to RUAS LMS</h1>
    </body>
</html>

```



```
<br/>
<center>
  <form align="center" action="login" method="post">
    <table>
      <tr>
        <td>username</td>
        <td><input type="text" name="username" /></td>
      </tr>
      <tr>
        <td>password</td>
        <td><input type="password" name="password" /></td>
      </tr>
    </table>
    <br/>
    <input type="submit" value="login" />
  </form>
</center>
</body>
</html>
```

**projects.jsp**

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ page import="java.util.List" %>
<%@ page import="webarch.models.Project" %>
<html>
  <head>
    <title>Project</title>
  </head>

  <body style="text-align:center; margin:auto">
    <h1>Search for Project Details</h1>
    <br/>
    <center>
      <form align="center" action="projects" method="get">
        <table>
          <tr>
            <td>project name</td>
            <td>
              <input type="text" name="projectname" />
            </td>
          </tr>
        </table>
        <br />
        <input type="submit" value="search" />
      </form>
      <br />
      <h2>Results</h2>
      <table>
        <tr>
          <th>Id</th>
          <th>Project Leader</th>
```

```

        <th>Project Name</th>
        <th>Mentor Name</th>
        <th>Department</th>
        <th>Category</th>
    </tr>
    <%
        List<Project> projects = (List<Project>) request.getAttribute("projects");
        if (projects != null) {
            if (projects.size() == 0) {
    %>
                <h4> No results found</h4>
    <%
            } else {
                for (Project project: projects) {
    %>
                    <tr>
                        <td>
                            <%out.write(project.getId().toString());%>
                        </td>
                        <td>
                            <%out.write(project.getProjectLeaderRegno());%>
                        </td>
                        <td>
                            <%out.write(project.getProjectName());%>
                        </td>
                        <td>
                            <%out.write(project.getMentorName());%>
                        </td>
                        <td>
                            <%out.write(project.getDepartment());%>
                        </td>
                        <td>
                            <%out.write(project.getCategory());%>
                        </td>
                    </tr>
                </td>
            </td>
        }
    }
    %>
</table>
</center>
</body>
</html>

```

## 5. Presentation of Results

Login Page (On Success redirects to Search Page)

Login

https://jsp-dev-pphmx.run-ap-south1.goorm.io/login

## Login to RUAS LMS

username

password

Register Page (On Success redirects to Login Page)

Register

https://jsp-dev-pphmx.run-ap-south1.goorm.io/register

## Register

username

password

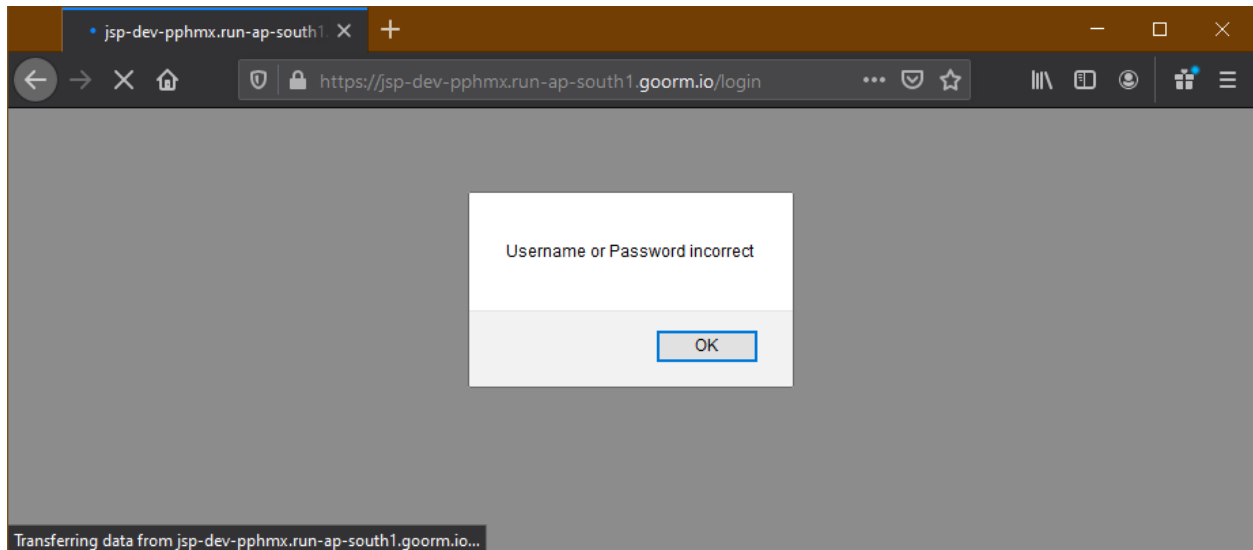
full name

usn no

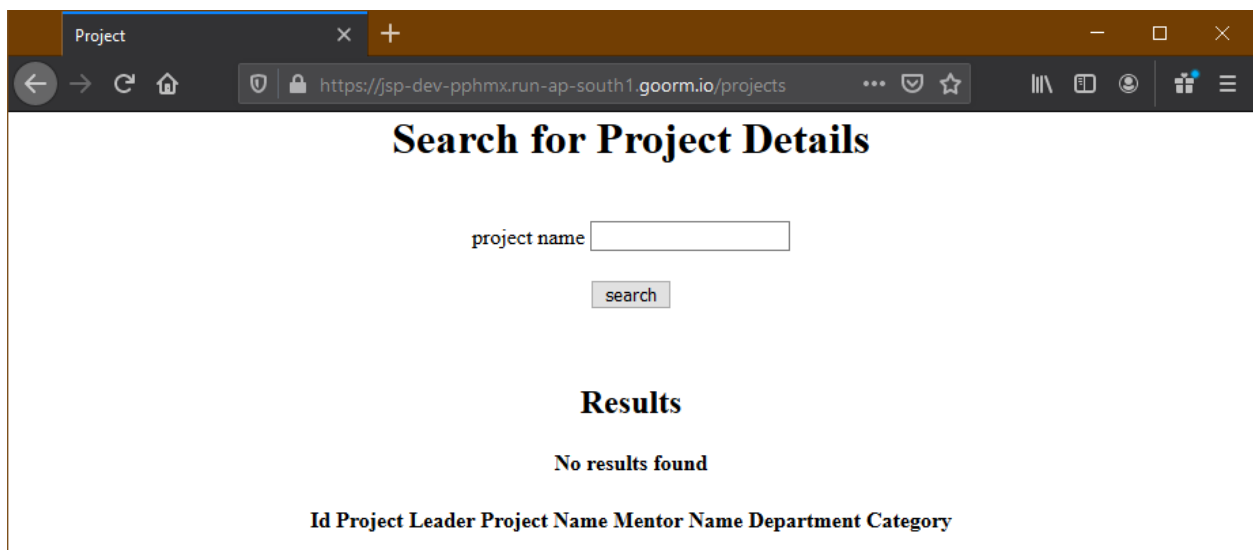
dept

course

Login Page (Error)



### Search Page



Search for a term

Search for Project Details

project name

**Results**

Id	Project Leader	Project Name	Mentor Name	Department	Category
2	17ETCS002159	KrishiAI	Chaitra S	CSE	DL

## Logs

```
INFO: SessionListener: contextInitialized()
Jan 26, 2021 11:38:42 AM org.apache.catalina.core.ApplicationContext log
INFO: ContextListener: attributeAdded('org.apache.jasper.compiler.TldLocationsC
Jan 26, 2021 11:38:42 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deployment of web application directory /goormService/tomcat7/webapps/exa
Jan 26, 2021 11:38:42 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Jan 26, 2021 11:38:42 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Jan 26, 2021 11:38:42 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 22094 ms
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class
ver class is generally unnecessary.
Login Success for User: satyajitghana
```

## 6. Analysis and Discussions

Servlet can accept all protocol requests, including HTTP, while JSP can only accept HTTP requests.

In MVC architecture, servlet works as a controller while JSP works as a view for displaying output.

Servlet should be used when there is more data processing involved whereas, JSP is generally used when there is less involvement of data processing.

We learnt how to use Servlets and connect them with JSPs, and make a complete working website flow. Java makes it easy to connect web pages, and with good UX as well.

## 7. Conclusions

We were able to create a Project Exhibition Management Website with functionalities like Login, Register and Searching for a Registered Project Details.

## 8. Comments

### a. Limitations of Experiments

Although Servlets are easy to use, they are being deprecated, the times are changing and people are moving towards robust and much easier to use frameworks like React, Vue, Svelte and Angular. They make it really easy to do routing, managing state and using the latest JS capabilities of the browser.

### b. Limitations of Results

The website implemented here, does not use dynamic loading, i.e. the buttons invoke a request and we get the result, the whole page is refreshed, and not just the required parts of the website. This makes the UX really not that usable.

### c. Learning happened

We learnt how to implement the functionalities of a website, from designing, to creating the DB and finally creating the JSP, Servlets and connecting all of these together.

### d. Recommendations

It would be really beneficial to use something like React or Vue, as they are really evolved and meant for the modern browsers.

Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	