Expt NO: 01 Date:

Aim: To create a web page will all types of Cascading Style Sheets.

Description:

CSS helps us to control the text color, font style, the spacing between paragraphs, sizing of columns, layout designs and many more.

The styles are applied as the following 3 ways.

* Inline : by using the style attribute inside HTML elements
* Internal : by using a <style> element in the <head> section
* External : by using a <link> element to link to an external CSS file.

HTML CODE:

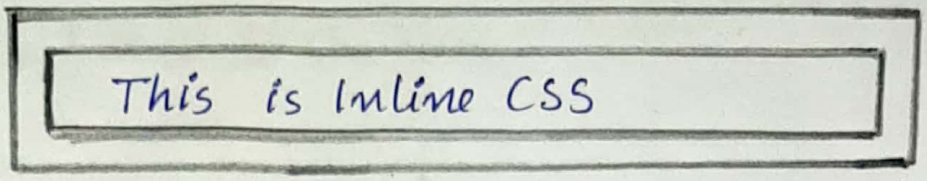
Mystyle.css

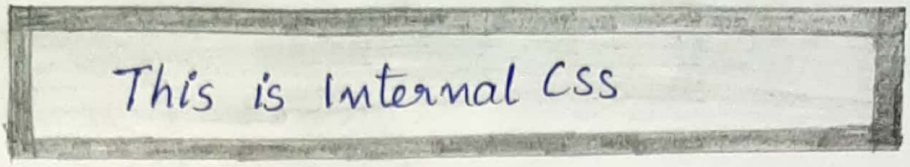
```
h1.external
{
  font-family : Arial;
  background-color : black;
  color : tomato;
  width : 600px;
  border-style : groove;
  border-width : 10px;
}
```

```
<html>
<head>
<title> ALL CSS DEMO </title>
<link rel=" stylesheet" href=" mystyle.css">
<style>
h1{
background-color: lightgray;
font-style :oblique;
border-style : solid;
width : 600px
border-width : 10px;
}
</style>
</head>
<body><center>
<h1 style=" color : blue; font-family:Times; background-color: hotpink;
  border-style:double; border-width : 10px; width:600px; ">
This is Inline CSS </h1><br><br>
<h1> This is Internal CSS </h1> <br>
The styles defined for h1 in style tag is used here <br>
<h1 class=" external "> This is External CSS </h1><br>
The styles defind for h1.external in 'mystyle.css' file is
 used.
</center>
</body>
</html>
```

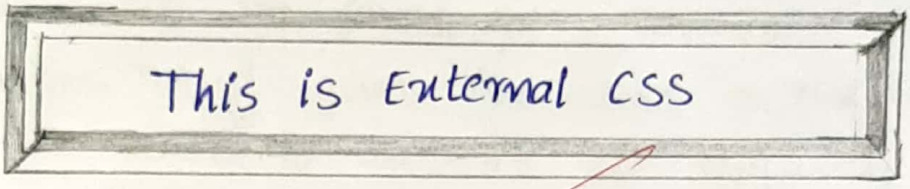
output:-

ALL CSS DEMO





The styles defined for h1 in style tag is used here



The styles defined for h1 in 'mystyle.css' file are used

08/11/24

Expt No: 02 Date:

Aim: To create a HTML5 form that interacts with the user. Collect first name, last name and date of birth and display that information back to the user.

Description:

To create the user form that interacts with user we use a HTML document which has three fields first name, last name and date of birth.

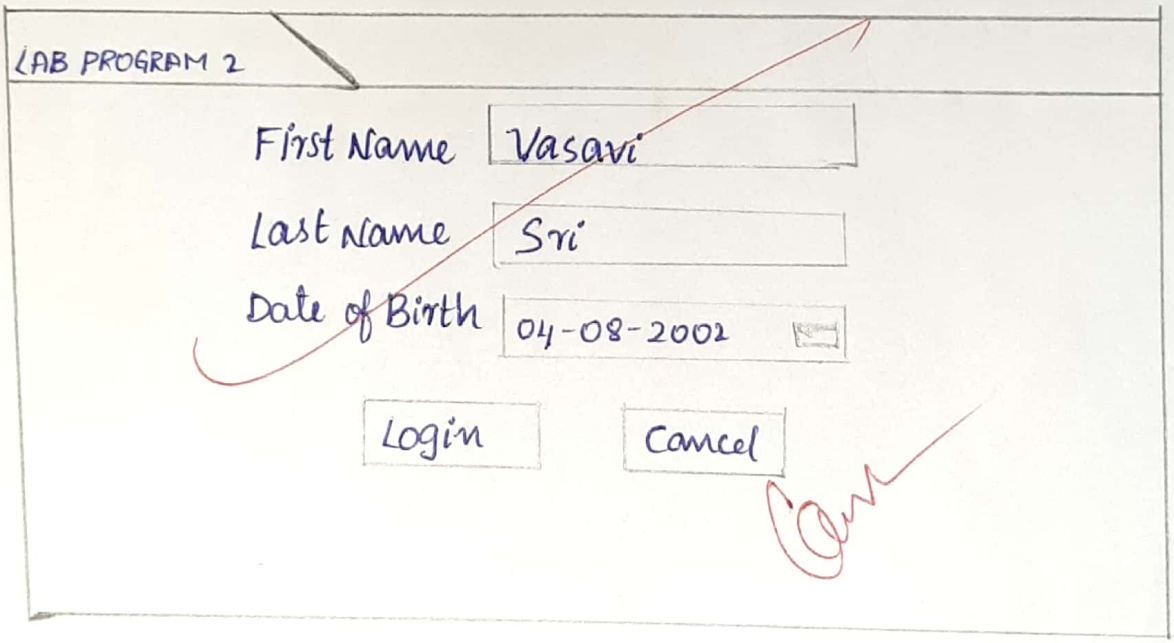
To collect the information entered by the user and display that information back to user we make use of the <script> tag to write the javascript. 'Javascript' evaluates the user information and gives the alert to the user. Javascript helps the server to increase its efficiency.

HTML CODE:

```
<html>
<head><title> LAB PROGRAM 2 </title>
<script language = "javascript">
function returnInfo()
{
    var fname = fl.fname.value;
    var lname = fl.lname.value;
    var dob = fl.dob.value;
    document.writeln(" First Name : " +fname);
    document.writeln("</br>LastName : " + lname);
    document.writeln("</br>Date of Birth : " + dob);
}
</script>
</head>
```

```
<body>
<center>
 <form name="f1">
  First Name   :   <input type="text" name="fname"></br>
</br>Last Name   :   <input type="text" name="lname"></br>
</br> Date of Birth  :  < input type="date" name="dob"></br>
<pre>      <input type="button" value="login" onclick="returnInfos">
        <input type="reset" value="cancel"></pre>
</form>
</center>
</body>
</html>
```

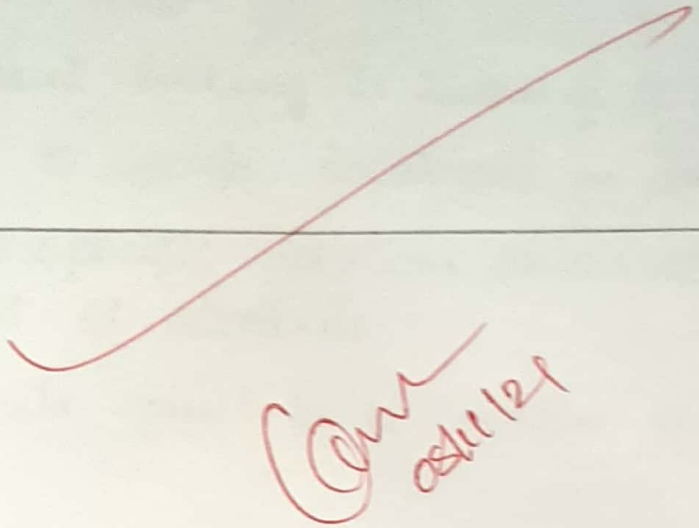
output:



First Name : Vasavi

Last Name : Sri

Date of Birth : 2002-08-04



Expt NO:-03 Date:

Aim: To Write HTML5 code to provide intra and inter document linking.

Description:

Intra document linking is linking to a location on the same page and to specific locations on another webpages.

To link to a specific location on a web page, use an anchor tag and id attribute.

The id attribute must be a unique value and can not contain spaces.

The id can be added to any web page element that accepts global attributes.

The name specified in the id is preceded by a pound sign in the href and must match exactly.

HTML CODE:

```
<html>
<head><title>Intra Document Linking</title>
</head>
<body>
  <a href="#lesson1">Lesson.1</a><br/>
  <a href="#lesson2">Lesson.2</a><br/>
  <a href="#lesson3">Lesson.3</a><br/>
  <br/>
  <a id="lesson1">Introduction of Lesson.1</a>
  <P>This is subtopic.1</P>
  <P>This is subtopic.2</P>
```

```
<p> This is subtopic . 3 </p>.
<br/>
<br/>
<div id= "lesson 2"> Introduction of Lession . 2 </div>
<p> This is sub topic . 1 </p>
<p> This is sub topic . 2 </p>
<p> This is sub topic . 3 </p>.
<br/>
<br/>
<p id=" lesson 3"> Introduction of lession . 3 </p>
<p> This is subtopic . 1 </p>
<p> This is subtopic . 2 </p>
<p> This is subtopic . 3 </p>
<br/>
<br/>.
</body>
</html>.
```

Output:

Lesson . 1
Lesson . 2
Lesson . 3

Introduction of Lesson . 1

This is subtopic . 1

This is subtopic . 2

This is subtopic . 3

Introduction of Lesson . 2

This is subtopic . 1

This is subtopic 2

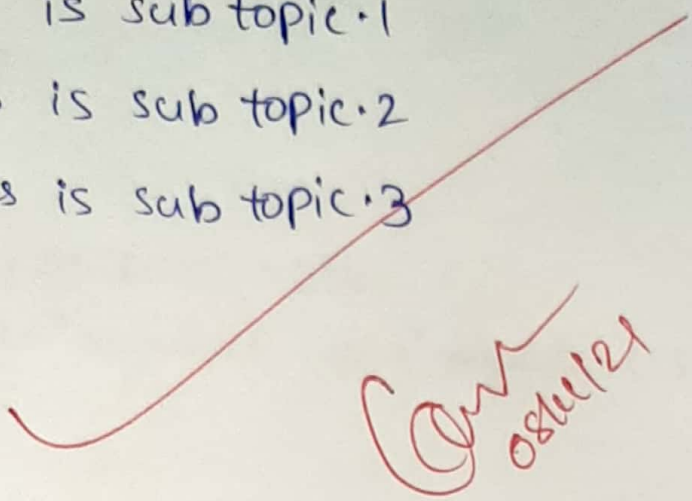
This is subtopic 3

Introduction to lesson 3

This is sub topic 1

This is sub topic 2

This is sub topic 3



Expt No:- 04 Date:

Aim: To Develop and Demonstrate a HTML5 Form Document that illustrates the use of ordered list, unordered list, table, borders, padding, color and the & tag.

HTML CODE:

```
<html>
<head>
<title> ALL CSS DEMO </title>
<link rel="stylesheet" href="mystyles.css">
</head>
<body>
<table>
<caption style="font-size:35px;"> TIME TABLE </caption>
<tr>
<th> Day </th>
<th> 8:00 - 8:55 </th>
<th> 8:55 - 9:15 </th>
<th> 9:15 - 10:10 </th>
<th> 10:10 - 11:05 </th>
<th> 11:05 - 11:15 </th>
<th> 11:15 - 12:10 </th>
<th> 12:10 - 01:05 </th>
</tr>
<tr>
<td> Monday </td>
<td> DAA </td>
<td rowspan="6" align="center"> BREAK </td>
<td> UNIX </td>
<td rowspan="6" align="center"> BREAK </td>
```

```
<td> AJWT </td>
<td> FLAT </td>
</tr>
<tr>
<td> Tuesday </td>
<td> AJWT </td>
<td colspan="2" align="center"> AJWT-LAB </td>
<td> FLAT </td>
<td> UNIX </td>
</tr>
<tr>
<td> Wednesday </td>
<td> MEFA </td>
<td> AJWT </td>
<td> DAA </td>
<td colspan="2" align="center"> DAA-LAB </td>
</tr>
<tr>
<td> Thursday </td>
<td> FLAT </td>
<td colspan="2" align="center"> UNIX-LAB </td>
<td> UNIX </td>
<td> MEFA </td>
</tr>
<tr>
<td> Friday </td>
<td> UNIX </td>
<td> MEFA </td>
<td> DAA </td>
<td> AJWT </td>
<td> FLAT </td>
</tr>
```

```
<tr>
<td> Saturday </td>
<td> AJWT </td><td> UNIX </td>
<td> MEFA </td>
<td> DAA </td>
<td> FLAT </td>
</tr>
</table>
<ul class="center" type="circle">
<li> AJWT- AdvancedJava and Web Technologies </li><br>
<li> DAA- Design and Analysis of Algorithms </li>
</ul>
<ol class="center" type="i" start="3">
<li> FLAT- Formal Languages and Automata Theory </li><br>
<li> MEFA- Managerial Economics and Financial Analysis </li><br>
<li> UP - Unix and shell Programming </li>
</ol>
</body>
</html>
```

TIME TABLE

Day	8:00-8:55	8:55-9:15	9:15-10:10	10:10-11:05	11:05-11:15	11:15-12:10	12:10-01:05
Monday	DAA	BREAK	UNIX	MEFA	BREAK	AJWT	FLAT
Tuesday	AJWT		AJWT-LAB			FLAT	UNIX
Wednesday	MEFA		AJWT	DAA		DAA-LAB	
Thursday	FLAT		UNIX-LAB			UNIX	MEFA
Friday	UNIX		MEFA	DAA		AJWT	FLAT
Saturday	AJWT		UNIX	MEFA		DAA	FLAT

o AJWT - Advanced Java and web Technologies
o DAA - Design and Analysis of Algorithms
iii. FLAT - Formal Languages and Automata Theory
iv. MEFA - Managerial Economics and Financial Accounting
v. UP - Unix and Shell Programming.

08/11/24

Expt No:- 05 Date:-

Aim: To develop a HTML5 form, which accepts any mathematical expression. Write Java script code to evaluate the expression and display the result.

Description:

To interact with the user to enter the mathematical expresseon we create the HTML document. The document of page has the fields expression, result, evaluate expression button and a reset button.

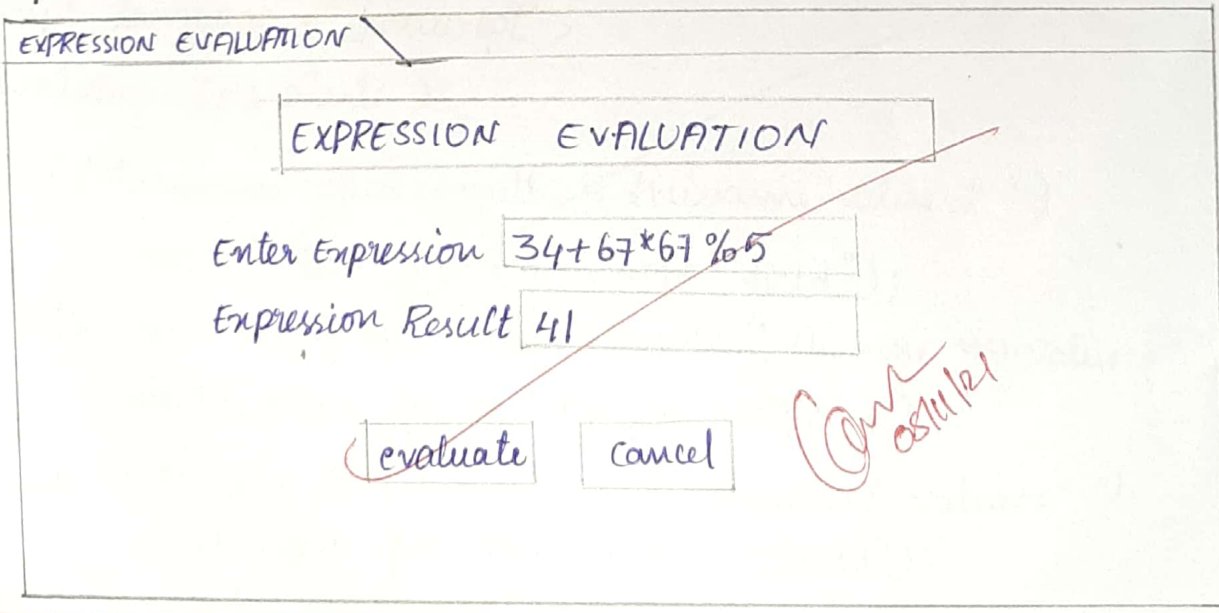
To accept the expression and evaluate we make use of the script tag to write the javascript. In the java script program we use the eval() method to evaluate the expression. we set the evaluated value to the result field value.

HTML CODE:

```
<html>
<head><title> EVALUATE EXPRESSION </title>
<script language="javascript">
function eval-exp()
{
    var exp = fl. expression .value
    var result = eval(exp);
    document fl.resultText.value = result;
}
</script>
</head>
```

```
<body>
<h1 style = "color : tomato ; background-color : black ; width : 550">
EXPRESSION EVALUATION </h1> <br><br><br><br>
<form name = "f1"> <center>
<fieldset style = " width : 500"> <br><br>
<label for = "expression"> Enter Expression </label>
<input type = "text" name = "expression" /> <br><br>
<label for = "resultText"> Expression Result </label>
<input type = "text" name = "resultText" /> <br><br>
<pre>    <input type = "button" value = "Evaluate" onclick = "eval-expr()">
      <input type = "reset" value = "cancel" ></pre>
</fieldset>
</center>
</form>
</body>
</html>
```

Output:



Expt NO: 06 Date:

Aim: To create a HTML5 form that has number of textbones. When the forms runs in the Browser fill the textbones with data. Write JavaScript code that varies that all textbones has been filled. If a text box has been left empty, popup an alert indicating which textbox has been left empty.

Description:

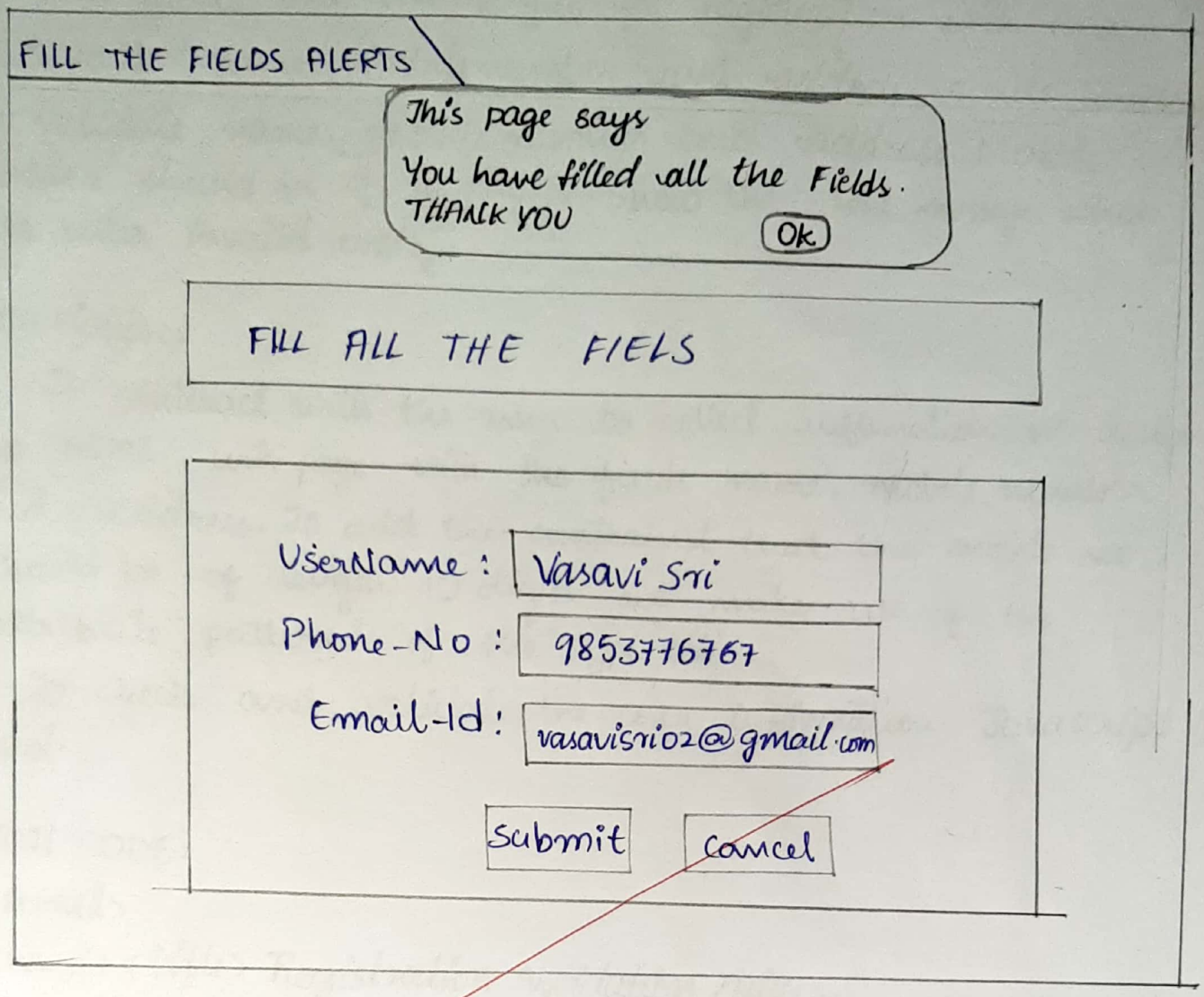
To create the user interface for the text fields we make use of the different types in input field. Now, we include three fields username, phone-no and address email-id

To accept the information and give alert message we make use of Javascript. If any of the fields is not filled then a popup alert is displayed to fill the leftout or unfilled field.

HTML CODE:

```
<html>
<head><title>FILL THE FIELDS ALERT </title>
<script language = "javascript">
function field_alert( )
{
   if ( fl.uname.value == null || fl.uname.value == " ")
        alert (" Please fill the 'NAME' field");
   else if(fl.phone_no.value == null || fl.phone_no.value == " ")
        alert (" Please fill the 'PHONE_NO" field");
   else if(fl.email_id.value == null || fl.email_id.value == " ")
        alert(" please fill the "EMAIL_ID'field");
```

```
    else
        alert("You have filled all the Fields. THANK YOU!");
}
</script>
</head>
<body>
<center></br></br></br>
<h1 style="color: tomato; background-color: black; width:600">
FILL ALL THE FIELDS </h1> <br><br>
<form name="f1">
<fieldset style="width:500"><br><br>
<label for="uname">UserName :  </label>
<input type="text" name="uname"/><br><br>
<label for="phone-no">Phone-No :  </label>
<input type="tel" name="phone-no"/><br><br>
<label for="email-id">Email-Id :  </label>
<input type="email" name="email-id"/><br><br>
<pre>  <input type="button" value="submit" onclick="field-alert()">
   <input type="reset" value="cancel"/></pre>
</fieldset>
</center>
</form>
</body>
</html>
```



08/11/24

Expt No: 6 Date:

Aim: To create an HTML5 file for registration with three text fields name, mobilenumber and address. Write Javascript to validate name, mobile number and address. Mobile number should be of 10 digits. Show the alert message when user enter invalid entity.

Description:

To interact with the user to collect information we design the HTML web page with the fields name, mobile number and address. To add the constraints that the mobile no should be of length 10 digits we make use of the attribute 'pattern' of the type 'tel'.

To check and validate the user information Javascript is used.

HTML CODE:

```
<html>
<head><title> Registration Validation </title>
</head>
<body><center>
<form name="fi" method="post" action="https://vvitguntur.com"
onsubmit="return validation(fi.uname.value, fi.phone_no.value,
fi.address.value)">
<legend style="color: tomato ; background-color: black; width: 500">
   REGISTRATION VALIDATION </legend>
<fieldset style="width:500"><br><br>
<label for="uname">UserName:</label>
<input type="text" name="uname"/><br><br>
```

```
<label for="phone_no"> Phone No : </label>
<input type="tel" pattern="[6-9]{1}[0-9]{9}" name="phone_no"/>
<br><br><label for="address"> Address : </label>
<textarea rows="2" cols="21"></textarea><br><br>
<pre>    <input type="submit" /> fnbspfnbspfmbsp <input type="reset"/>
</pre>
</fieldset>
</form>
</center>
<script language="javascript">
function validation(name, phone, address)
{
  if(name==null || name=="")
  {
     alert("Please enter the 'NAME' field!");
     return false;
  }
  if(phone==null || phone=="")
  {
     alert("Please enter the 'PHONE-NO' field!");
     return false;
  }
  if(address==null || address=="")
  {
     alert("Please enter the 'ADDRESS' field!");
     return false;
  }
}
</script></body>
</html>
```

Registration Validation

REGISTRATION VALIDATION

User Name : Vasavisri

Phone_No : 9853776767

Address : H.No:-8-285, Emani (P), Duggirala (M), Guntur (D), A.P

Submit Cancel

05/11/24

Expt No: 08 Date:-

Aim: To write JDBC program to create table EMP with following fields.

- Empid number(4)
- Ename varchar(20)
- Salary number(6)
- Address varchar(50)

Description:

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. We make use of JDBC to perform the following actions

1) Connect to database
2) Execute queries and update statements to the database
3) Retrieve the result recieved from the database.

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

* Load the Drivers
* Create Connection
* Create Statement
* Execute Queries
* close connection

The above steps are implemented in the following JDBC program to create a table in the database with the given name and fields.

JAVA CODE:

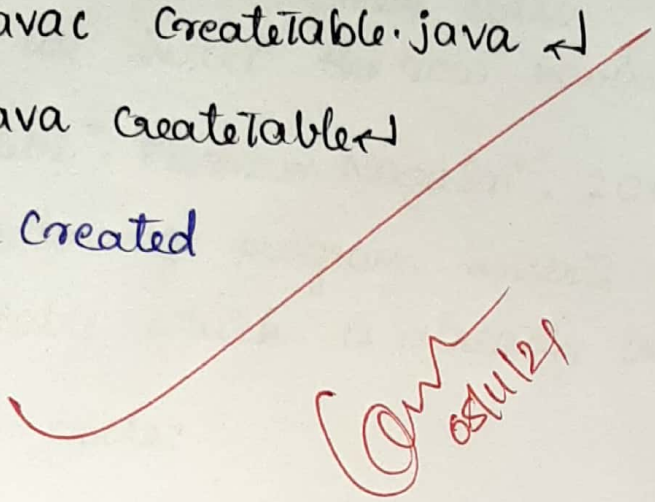
```
import java.sql.Connection;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;

public class CreateTable
{ public static void main(String[] args)
  {
    ResultSet rs;
    Connection con;
    Statement st;
    try{
       Class.forName("oracle.jdbc.driver.OracleDriver");
      con = DriverManager.getConnection("jdbc:oracle:thin:@192.168.
            12.2:1521:orcl","R19BQ1A05C7","R19BQ1A05C7");
     st=con.createStatement();
     St.executeUpdate("Create Table EMP(Empid number(4),
         ename varchar(20), salary number(6), address varchar(50))")
     System.out.println("Table Created");
     st.close();
     con.close();
  }
  catch(ClassNotFoundException cnf)
  {
      cnf.printStackTrace();
}
```

```
    catch(SQLException sqle)
    {
        sqle.printStackTrace();
    }
  }
}
```

Output:

```
$ javac CreateTable.java ↵
$ java CreateTable ↵
Table Created
```



Expt NO:-09 Date:

Aim: To write JDBC program to insert a record into EMP table.

Description:

To insert a row into the DataBase using the JDBC programs. We make use of the executeUpdate() method available in the statement class.

Let us insert the row having the values.

"4502", 'Mannam Nagasri', 20000, 'Emani, Duggirala, Guntur'

The following program inserts the above row into the EMP table which is already created.

JDBC code:

```
import java.sql.*;
public class InsertIntoTable{

  public static void main(String[] args)
  {
     Connection con;
     Statement st;
     try
     {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        con = DriverManager.getConnection("jdbc:oracle:thin:
              @192.168.12.2:1521:orcl","R19BQIA05C7","R19BQIA05C7");
        st = con.createStatement();
```

```
        st.executeUpdate("insert into EMP values(4502,'Mannam
                    Nagasri',20000,'Emani,Duggirala(M),Guntur
                                (D),A.P.')");
        System.out.println("One row inserted.");
    }
    catch (ClassNotFoundException cnf)
    {
        cnf.printStackTrace();
    }
    catch (SQLException sqle)
    {
        sqle.printStackTrace();
    }
  }
}
```

Output:

```
$ javac InsertIntoTable.java ↵
$ java InsertIntoTable ↵
one row inserted.
```

08/04/24

Expt NO:-10

Date:-

Aim: To write a JDBC program to display EMP table

Description:

To display the EMP table first we need to access the database by loading the necessary drivers, connecting to the database and set up the or prepare the statement which executes the given Query.

The result of the Query is stored in a ResultSet object. the ResultSet object maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row here to access the values of each row we make use of the method "public boolean next()" which moves the cursor to the one row next from the current position.

JDBC CODE:

```
import java.sql.*;
public class DisplayTable
{
  public static void main(String[] args)
  {
    ResultSet rs;
    Connection con;
    Statement st;
    try
    {
      class.forName("oracle.jdbc.driver.OracleDriver");
      con=DriverManager.getConnection("jdbc:oracle:thin:@192.168.12
          .2:1521:orcl","R19BQIA05C7","R19BQIA05C7");
```

```
        st=con.createStatement();
        rs=st.executeQuery("select * from EMP");
        System.out.println("EMPID \t ENAME \t SALARY \t
                            ADDRESS");
        while(rs.next()){
            System.Out.println(rs.getInt(1)+'\t'+rs.getString(2)+
                               '\t'+rs.getInt(3)+'\t'+rs.getString(4));
    }
    st.close();
    con.close();
  }
  catch(ClassNOTFoundException cnf)
  {
     cnf.printStackTrace();
  }
  catch(SQLException sqle)
   {
     sqle.printStackTrace();
   }
 }
}
```

output:

```
$ javac DisplayTable.java
$ java DisplayTable
```

08/11/21

EMPID	ENAME	SALARY	ADDRESS
4504	M Vasavi Sri	15000	EMANI, GUNTUR, A.P
4503	M Phaneendra	16000	DUGGIRALA, GUNTUR, A.P
4502	M Naga Sri	20000	EMANI, GUNTUR, A.P
4501	Mary	40000	GUNTUR, A.P

Date:

Expt NO:11

Aim: To write JDBC program to update Emp table to set name as JHON where Empid =501.

Description:

To update a row in the database we make use of the Query

"Update EMP set Ename = 'JHON' where empid=4501"

JDBC CODE:

```
import java.sql.*;
Public class UpdateTable
{
   public static void main(String[] args)
   {
      ResultSet rs;
      Connection con;
      Statement st;
      try
      {
         class.forName("oracle.jdbc.driver.OracleDriver");
         con = DriverManager.getConnection("jdbc:oracle:thin:@192.168.
           12.2:1521:orcl", "R19BQ1A05C7", "R19BQ1A05C7");
         st = con.createStatement();
         st.executeUpdate("Update emp set Ename= 'Jhon' where
                          empid=4501");
         System.out.println("one row updated");
         System.out.println("EMPID"+"\t"+"ENAME");
```

```
    rs = st.executeQuery("select * from ~~from~~ EMPID, ENAME from
                          EMP");
    while (rs.next())
    {
        System.out.println(rs.getInt(1) + "\t" + rs.getString(2));
    }
     st.close();
     con.close();
  }
  catch (ClassNotFoundException cnf)
  {
      cnf.printStackTrace();
  }
   catch(SQLException sqle)
   {
      sqle.printStackTrace();
    }
  }
}
```

Output:

```
$ javac UpdateTable.java ↵
$ java UpdateTable ↵
one row updated

EMPID    ENAME
4504     M Vasavi Sri
4503     M Phaneendra
4502     M NagaSri
4501     Jhon
```

05/11/24

Expt No: Date:

Aim: To write a servlet program for Extending Generic Servlet

Description:

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page.

Servlet technology is robust and scalable because of java language.

Servlet's APIs:

The servlet API consists of classes and interfaces needed to build servlets.

These classes and interface comes in two packages.

- javax.servlet (Basic)
- javax.servlet.http (Advance)

Protocol independent classes and interfaces exist in the package javax.servlet whereas http specific classes and interfaces exits in javax.servlet.http package.

1. Interfaces in javax.servlet package:

There are many interfaces in javax.servlet package. They are as follows:

1. Servlet
2. ServletRequest
3. ServletResponse
4. RequestDispatcher
5. ServletConfig
6. ServletContext
7. SingleThreadModel
8. Filter

9. FilterConfig.
10. FilterChain
11. ServletRequestListner
12. ServletRequestAttributeListner
13. ServletContextListner
14. ServletContextAttributeListner.

Classes in javax.servlet package

There are many classes in javax.servlet package. They are as follows:

1. GenericServlet
2. ServletInputStream
3. ServletOutputStream
4. ServletRequestWrapper
5. ServletResponseWrapper
6. ServletRequestEvent.
7. ServletContextEvent
8. ServletRequestAttributeEvent
9. ServletRequestContextAttributeEvent
10. ServletException
11. UnavailableException.

Methods of Generic Servlet class:

There are many methods in Generic servlet class. They are as follows:

1) public void init(ServletConfig config) is used to initialize the Servlet.

2) Public abstract void service(ServletRequest, Request, Servlet-Response Response) provides service for the incoming request. It is invoked at each time when user requests for a servlet

3) Public void destroy() is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4) public ServletConfig getServletConfig() returns the object of servlet-config
5) Public String getServletInfo() returns the information about servlet such as writer, copyright, version etc.
6) public void init() it is a convenient method for the servlet programmers, now there is no need to call super.init(config)
7) Public ServletContext getServletContext() returns the object of ServletContext
8) Public String getInitParameter(String name) returns the parameter value for the given parameter name.
9) Public Enumeration getInitParameterNames() returns all the parameters defined in the web.xml file.
10) public String getServletName() returns the name of the servlet object
11) public void log(String msg) writes the given message in the servlet log file.
12) public void log(String msg, Throwable t) writes the explanatory message in the servlet log file and a stack trace.

* Servlet cycle:

* The servlet is initialized by calling the init() method.
* The servlet calls service() method to process a client's request.
* The servlet is terminated by calling the destroy() method
* Finally, servlet is garbage collected by the garbage collector of the JVM

The java program

GenericServletDemo.java

```
import javax.servlet.*;
import java.io.*;

public class GenericServletDemo extends GenericServlet
{
  public void servlet(ServletRequest req, ServletResponse res) throws
     ServletException, IOException
  {
     PrintWriter pw = res.getWriter();
     res.setContentType("text/html");
     pw.println("HI ROCKERS WELCOME TO SERVLET WORLD");
     pw.close();
  }
}
```

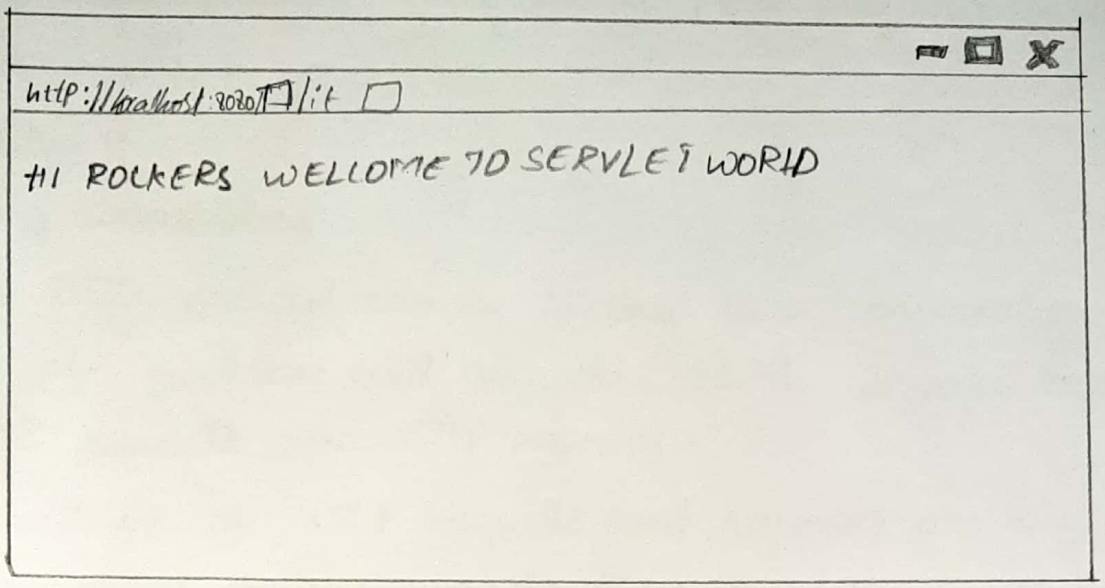
web.xml

```
<web-app>
  <servlet>
  <servlet-name>GenericServletDemo</servlet-name>
  <servlet-class>GenericServletDemo</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>GenericServletDemo</servlet-name>
  <url-pattern>/GenericServletDemo</url-pattern>
  </servlet-mapping>
</web-app>
```

To view the output:

Type the url below in the browser after starting tomcat

http://localhost:8080/GenericServlet/GenericServletDemo



Expt No:

Date:

Aim: To write a servlet program for Extending HttpServlet to read parameters.

Description:

Reading parameters:

The HTTP protocol can be likened to a conversation based on a series of questions and answers, which we refer to respectively as HTTP requests and HTTP responses.

The contents of HTTP requests and responses are easy to read and understand, being near to plain english in their syntax.

The javax.servlet.http package

Interfaces in javax.servlet.http package

There are many interfaces in javax.servlet.http package. They are as follows:

1. HttpServletRequest
2. HttpServletResponse
3. HttpSession
4. HttpSessionListner
5. HttpSessionAttributeListner.
6. HttpSessionBindingListner.
7. HttpSessionActivationListner.
8. HttpSessionContext (deprecated now)

Classes in javax.servlet.http package.

There are many classes in javax.servlet.http package. They are as follow

1. HttpServlet
2. Cookie
3. HttpServletRequestWrapper

3. HttpServletRequestWrapper
4. HttpServletResponseWrapper
5. HttpSessionEvent
6. HttpSessionBindingEvent
7. HttpUtils (deprecated now)

Methods of HttpServlet class:

There are many methods in HttpServlet class. They are as follows:

1. public void service(ServletRequest req, ServletResponse res) dispatches the request to the protected service methody by converting the request and response object into http type.
2. Protected void service(HttpServletRequest req, HttpServletResponse res) receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
3. protected void doGet(HttpServletRequest req, HttpServletResponse res) handles the GET request. It is invoked by the web container
4. protected void doPost(HttpServletRequest req, HttpServletResponse res) handles the POST request. It is invoked by the web container.
5. protected void doHead(HttpServletRequest req, HttpServletResponse res) handles the HEAD request. It is invoked by the web container
6. protected void doOptions(HttpServletRequest req, HttpServletResponse res) handles the OPTIONS request. It is invoked by the web container
7. protected void doPut(HttpServletRequest req, HttpServletResponse res) handles the PUT request. It is invoked by the web container
8. protected void doTrace(HttpServletRequest req, HttpServletResponse res) handles the TRACE request. It is invoked by the web container
9. protected void doDelete(HttpServletRequest req, HttpServletResponse res) handles the DELETE request. It is invoked by the web container.
10. protected long getLastModified(HttpServletRequest req) returns the time when HttpServletRequest was last modified since midnight Jan 1, 1970 GMT

Name.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Name extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res)
  throws ServletException, IOException.
  {
      String n,p;
      res.setContentType("text/html");
      PrintWriter pw = res.getWriter();
      n = req.getParameter("uname");
      P = req.getParameter("pwd");
      pw.println("userName is:"+n);
      pw.println("Your password is:"+p);
      pw.close();
  }
}
```

web.xml

```
<web-app>
<servlet>
 <servlet-name>IT</servlet-name>
 <servlet-class>Name</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>IT</servlet-name>
  <url-pattern>/Name</servurl-patterns>
</web-app>
```

Client Program

Login.html

```
<html>
<head>
<title>Login</title>
</head>
<body><center>
<form method="get" name="f" action="it/Name">
NAME : <input type="text" name="uname">
PASSWORD: <input type="password" name="pwd">
<input type="SUBMIT" value="submit Details">
</form>
</center>
</body>
</html>
```

http://localhost:8080/one/Login.html.

NAME [vvit] PASSWORD [....] [submit Details]

localhost:8080/name/name=vvit&pwd=vvit

my name is: vvit your password is vvit

Expt No:

Date:

Aim: To write a servlet program to Read the contents from cookies

Description:

Servlet Cookies:

Cookies are the small amount of information sent by the servlet to the web browser. Browser saves this information and later on it sends to server. Cookies are commonly used for session management because value of cookies can uniquely identify a client. Cookies can be name, a single value and optional attributes like comment, path, domain qualifiers etc..

The servlet method is used to add cookies to a web browser is HttpServletResponse.addCookie (javax.servlet.http.Cookie) and cookies can be retrieved by request, the method for getting this thing is HttpServletRequest.getCookies().

Constructor:

Cookie(java.lang.String name, java.lang.string value); This constructor is used to construct a cookie with specified name and value.

Cookie class methods:

java.lang.object clone(): It overrides the standard java.lang.object.clone() method which is used to return copy of this cookie.

int getMaxAge(): This method is used to return maximum Age of cookie specified in seconds. By default, when age is -1 it means the cookie will persist until browser shutdown.

java.lang.String getName(); This method is used to return name of cookie.

void setMaxAge(int exp): This method is used to sets the maximum age of cookies in seconds.

void setValue(java.lang.String newValue): This method is used to assign a new value to a cookie after the cookie protocol this cookie compeles with.

ReadCookies.java

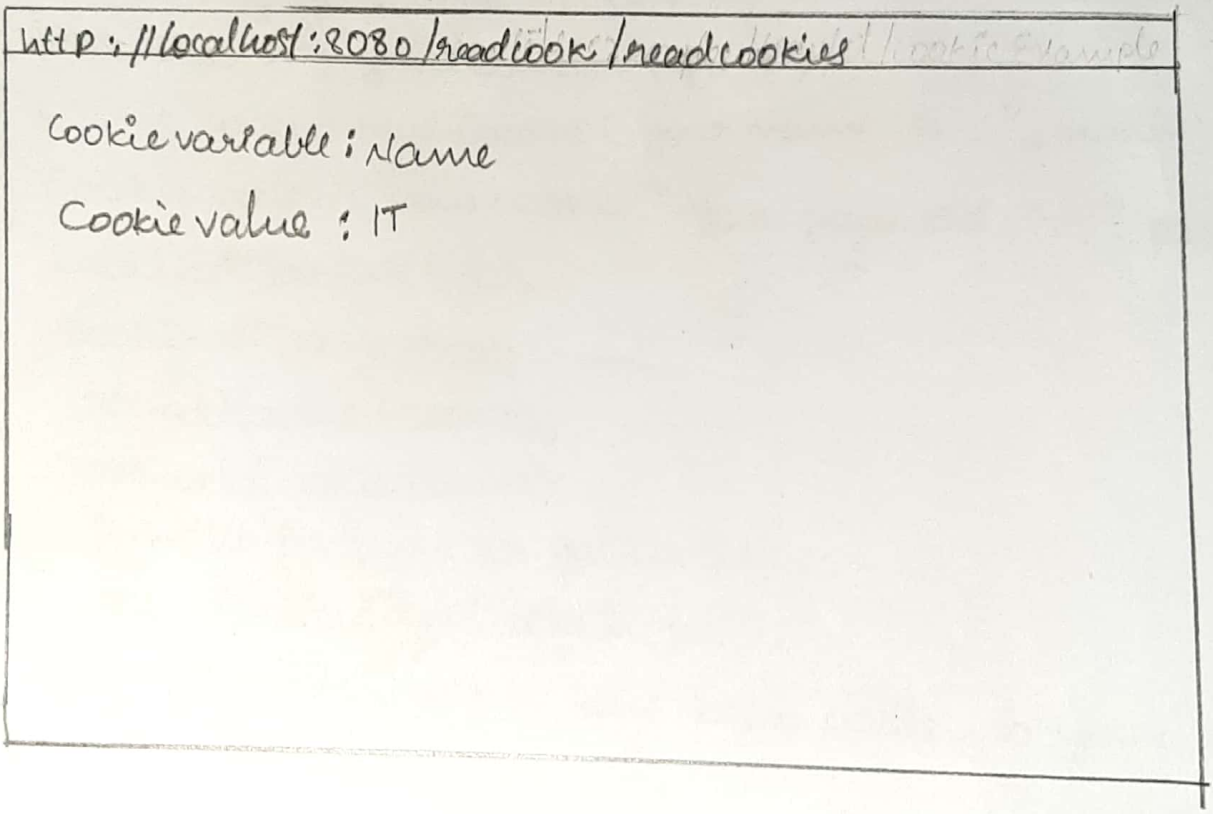
```
import javax.servlet.*;
import java.io.*;
import javax.servlet.http;
public class ReadCookies extends HttpServlet
{
   public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
   {
      Cookie cook[] = req.getCookies();
      PrintWriter pw = res.getWriter();
      for(int i=0; i<cook.length; i++)
      {
         pw.println("cookie variable :" +cook[i].getName());
         pw.println("cookie value :" +cook[i].getValue());
      }
      pw.close();
   }
}
```

read.html

```
<html>
<body>
<a href="http://localhost:8080/readcook/readcookies">
if you want to see the cookies then click here
</a>
</body>
</html>
```

web.xml

```
<web-app>
<servlet>
<servlet-name> hello </servlet-name>
<servlet-class> ReadCookies </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name> hello </servlet-name>
<url-pattern> /ReadCookies </url-pattern>
</servlet-mapping>
</web-app>
```



Expt No:

Date:

Aim: To write a servlet program to write the contents to cookies

WriteCookies.java

```
import javax.servlet.*;
import java.io.*;
import javax.servlet.http.*;
Public class WriteCookies extend HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res)
  throws ServletException, IOException
  {
     String name, password;
     name = req.getParameter("cname");
     password = req.getParameter("pwd");
     Cookie cook1 = new Cookie("your name is :", name);
     Cookie cook2 = new Cookie("Your password is :", password);
     cook1.setMaxAge(500);
     cook2.setMaxAge(500);
     res.addCookie(cook1);
     res.addCookie(cook2);
     PrintWriter pw = res.getWriter();
     res.setContentType("text/html");
     pw.println("cookies have been written to your system");
  }   pw.close();
}
```

Cookie.html

```
<html>
<head>
<title> cookies </title>
</head>
<body>
<h2> Login </h2>
<p> please enter your username and password </p>
<form method="get" action="/cook/writecookies">
<p> yourname < input type="text" name="uname" size="20" </p>
<p> password < input type="password" name="pwd" size="20"></p>
<p><input type="submit" value="submit name="b1"></p>
</form>
</body>
</html>
```

web.xml

```
<web-app>
<servlet>
<servlet-name> hello </servlet-name>
<servlet-class> writeCookies </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name> hello </servlet-name>
<url-pattern> /writeCookies </url-pattern>
</servlet-mapping>
</web-app>
```

Expt No: Date

Aim: To write servlet to insert registration form data into database

Registration.html

```
<html>
<body>
<form action="Reg" name="f">
 <table>
   <tr>
     <td> ENTER NAME </td>
     <td><input type="text" name="name"></td>
   </tr>
   <tr>
     <td> Enter Employee Password </td>
     <td><input type="password" name="pwd"></td>
   </tr>
   <tr>
     <td>< Enter Email </td>
     <td><input type="email" name="email"></td>
   </tr>
   <tr>
     <td> Enter Employee Mobile </td>
     <td><input type="number" name="phoneno"></td>
   </tr>
   <tr>
     <td> Gender </td>
     <td><input type="radio" name="gender" value="m"> Male
         <input type="radio" name="gender" value="f"> Female
         <input type="radio" name="gender" value="t"> Transgender
     </td>
   </tr>
   <tr>
     <td> Qualification </td>
```

```
<td><select name="qualification">
    <option value=" ">Select </option>
    <option value="B.Tech" selected> B.Tech</option>
    <option value="M.Tech"> M.Tech</option>
  </td>
 </tr>
 <tr>
   <td></td>
   <td><input type="submit" value="register">
   <input type="reset" value="cancel" ></td>
 </tr>
</table>
</body>
</html>
```

Reg.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;
public class Register extends HttpServlet
{ public void service(HttpServletRequest req, HttpServletResponse res)
  throws IOException, ServletException
  {
    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();
    String name = req.getParameter("name");
    String pwd = req.getParameter("pwd");
    String mail = req.getParameter("email");
    long phno = Long.parseLong(req.getParameter("phone_no"));
    String sex = req.getParameter("gender");
    String quali = req.getParameter("qualification");
```

```
try
{
  Class.forName("com.mysql.jdbc.Driver");
  Connection con = DriverManager.getConnection("jdbc:mysql://
                   localhost:3308/"vinny","root","root");
  PreparedStatement ps = con.prepareStatement("insert into
                         register values(?,?,?,?,?,?)");
  ps.setString(1,name);
  ps.setString(2,pwd);
  ps.setString(3, mail);
  ps.setLong(4, phoneno);
  ps.setString(5, sex);
  ps.setString(6, quali);
  int i= ps.executeUpdate();
  if(i>0)
      pw.println(i+"You are successfully registered...");
 }
 catch (Exception ex)
  {
    ex.printStackTrace(ex);
  }
  pw.close();
 }
}
```

web.xml

```
<web-app>
<servlet>
   <servlet-name>Register</servlet-name>
   <servlet-class>Register</servlet-class>
</servlet>
```

```
<servlet-mapping>
        <servlet-name>Register</servlet-name>
        <url-pattern>/Reg</url-pattern>
</servlet-mapping>
</web-app>
```