**Experiment 4:**  
   
**Write an HTML page that has one input, which can take multi line text and a submit button. Once the user clicks the submit button, it should show the number of characters, words and lines in the text entered using an alert message. Words are separated with white space and lines are separated with new line character.**

**Exp4.html**

<html>

<head>

<title>Lines,words,characters</title>

<script language="javascript">

function count()

{

var str=document.getElementById('inputstring').value;

var result='';

result+='The number of characters are'+str.length+'\n';

var arr=str.split(' ');

result+='The number of words are'+arr.length+'\n';

var a=str.split('\n');

result+='The number of lines are'+a.length;

alert(result);

}

</script></head>

<body>

<form name="frm1">

<center><b>Enter Multiple lines of text </b><br/><br/><textarea id="inputstring" cols="50" rows="5"></textarea><br/><br/>

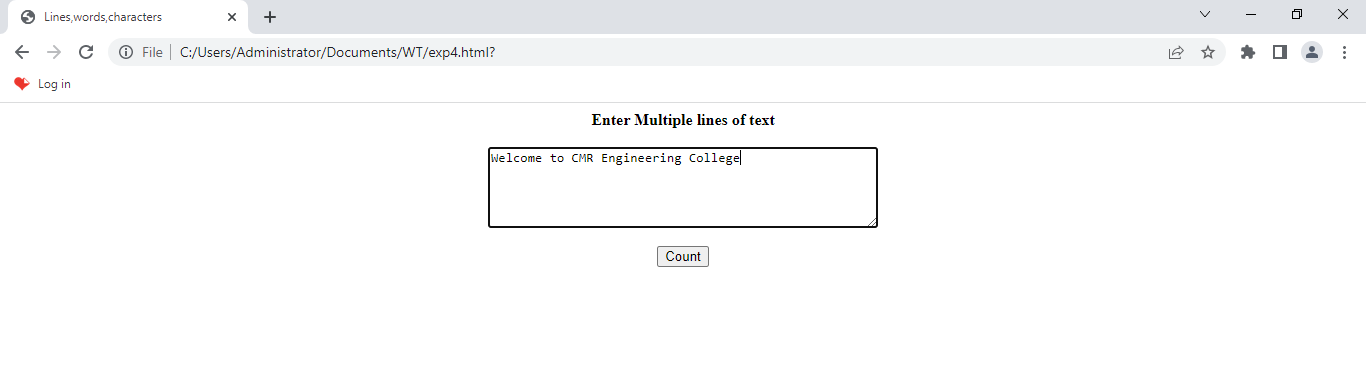
<input type="button" value="Count" onclick="count()"></input><br/><br/>

</form>

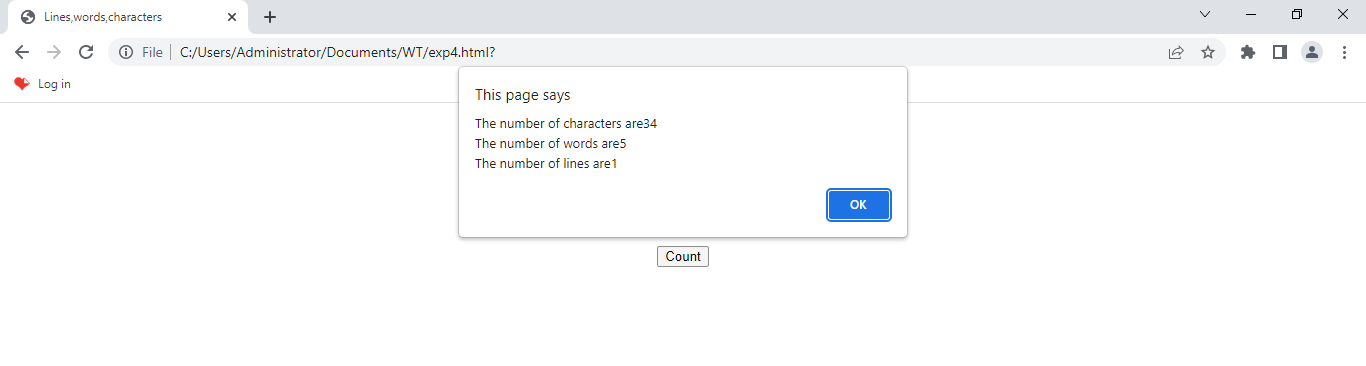
</body>

</html>

**Output:**

**Input:   
  
**

**Result:**

****

**Experiment 5:**  
 **Write an HTML page that contains a selection box with a list of 5 countries. When the user selects a country, its capital should be printed next to the list. Add CSS to customize the properties of the font of the capital (color, bold and font size).**

**Exp5.html**

<html>

<title>Capitals of Countries</title>

<head>

<style>

h1{

color:blue;

font-family:verdana;

font-size:300%;

}

select{

// A reset of styles, including removing the default dropdown arrow

appearance: none;

// Additional resets for further consistency

background-color: #fff;

border: 3;

border-radius: 0.35em;

padding: 0.25em 0.5em;

margin: 0;

width: 40%;

font-family: verdana;

font-size: inherit;

cursor: pointer;

line-height: inherit;

}

</style>

<script language="javascript">

function OnDropDownChange(dropDown)

{

var selectedValue = dropDown.options[dropDown.selectedIndex].value;

document.getElementById("txtSelectedCapital").innerHTML = selectedValue;

}

</script>

</head>

<body>

<form action = "">

<center>

<b>Select a Country :</b> <select name = "Countries" onChange="OnDropDownChange(this);">

<option value="">--Select a country--</option>

<option value="New Delhi">India</option>

<option value="Wellington">New Zealand</option>

<option value="Paris">France</option>

<option value="Athens">Greece</option>

<option value="Madrid">Spain</option>

<option value="Beijing">China</option>

<option value="Islamabad">Pakistan</option>

<option value="Japan">Japan</option>

</select>

<h1 id="txtSelectedCapital" type="text"></h1>

</center>

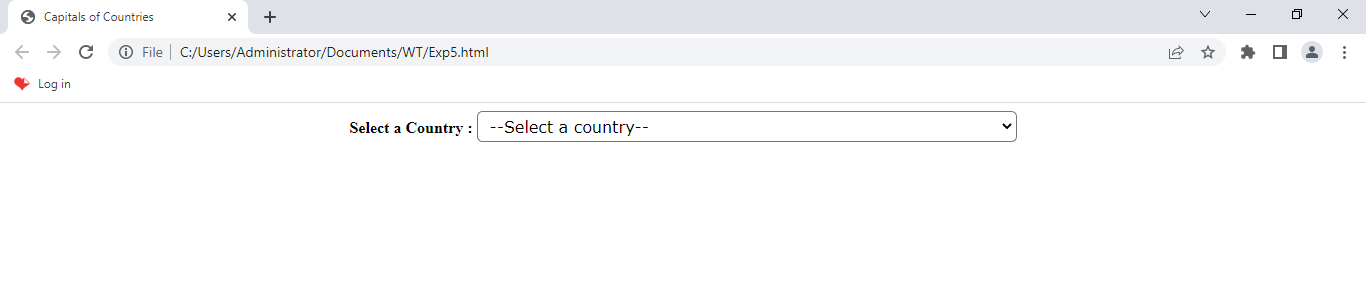
</form>

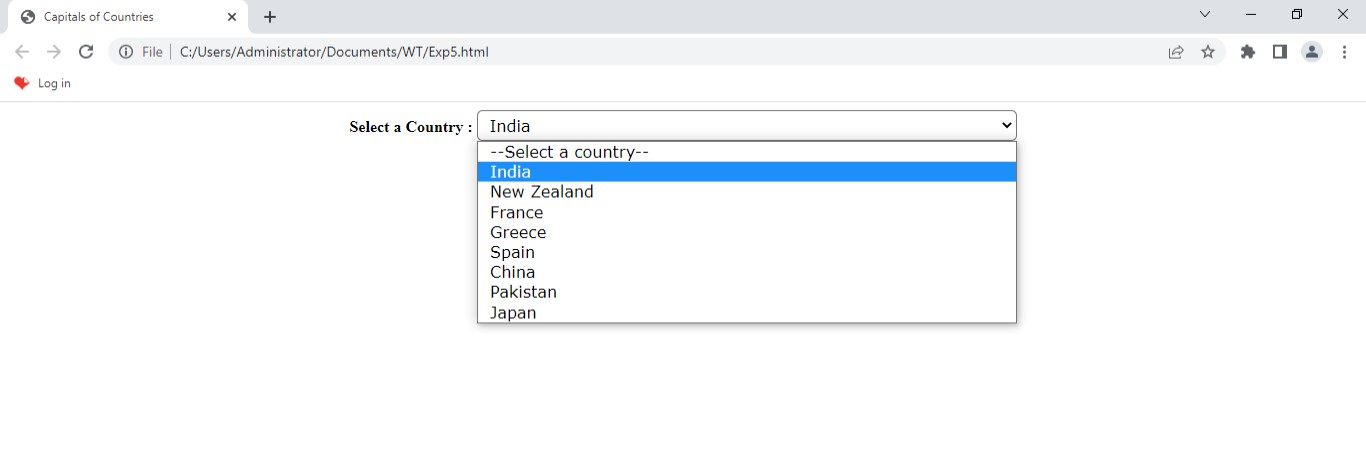
</body>

</html>

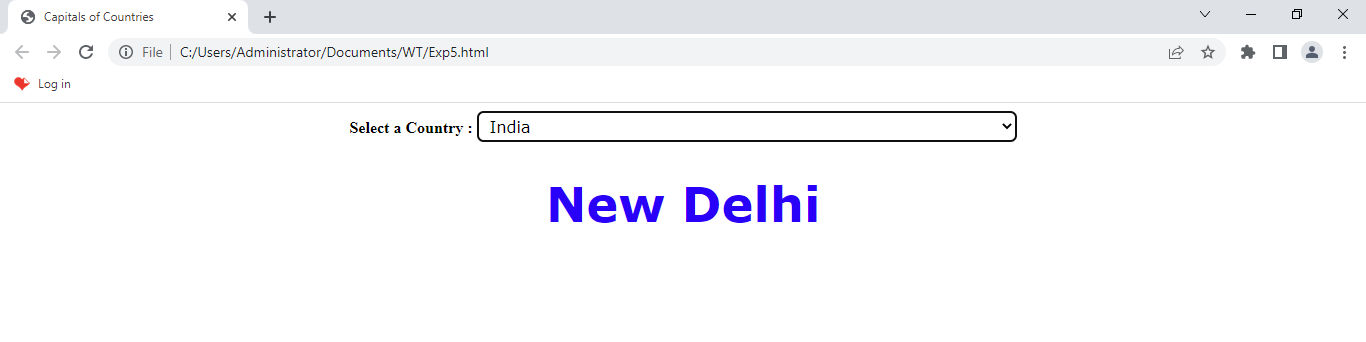
**Output :**

**Selection of Value:**





**Displaying the value of selected Index:**

****

**Experiment 6: Create an XML document that contains 10 users information. Write a java program, which takes user id as input and returns the user details by taking the user information from XML document using**

**a)DOM parser**

**b)SAX parser**

**Procedure: write the java and xml files. Compile and run the java file.**

**AIM: Takes User Id as input and returns the user details using XML with DOM**

**Java DOM Parser:**DOM stands for Document Object Model. The DOM API provides the classes to read and write an XML file. DOM reads an entire document. It is useful when reading small to medium size XML files. It is a tree-based parser and a little slow when compared to SAX and occupies more space when loaded into memory. We can insert and delete nodes using the DOM API.

We have to follow the below process to extract data from an XML file in Java.

* **Instantiate XML file:**
* **Get root node:**We can use getDocumentElement() to get the root node and the element of the XML file.
* **Get all nodes:**On using getElementByTagName() Returns a NodeList of all the Elements in document order with a given tag name and are contained in the document.
* **Get Node by text value:**We can use getElementByTextValue() method in order to search for a node by its value.
* **Get Node by attribute value:** we can use the getElementByTagName() method along with getAttribute() method.

# PROGRAM:

**employees.xml**

<employees>

<employee id="111">

<firstName>Naresh</firstName>

<lastName>Gupta</lastName>

<location>India</location>

</employee>

<employee id="222">

<firstName>Kumar</firstName>

<lastName>Gussin</lastName>

<location>Russia</location>

</employee>

<employee id="333">

<firstName>David</firstName>

<lastName>Feezor</lastName>

<location>USA</location>

</employee>

<employee id="444">

<firstName>Lokesh</firstName>

<lastName>Gupta</lastName>

<location>India</location>

</employee>

<employee id="555">

<firstName>Vishnu</firstName>

<lastName>Gussin</lastName>

<location>Russia</location>

</employee>

<employee id="666">

<firstName>Veeru</firstName>

<lastName>Feezor</lastName>

<location>USA</location>

</employee>

<employee id="777">

<firstName>Pavan</firstName>

<lastName>Feezor</lastName>

<location>USA</location>

</employee>

<employee id="888">

<firstName>Narayana</firstName>

<lastName>Gussin</lastName>

<location>Russia</location>

</employee>

<employee id="999">

<firstName>David</firstName>

<lastName>Feezor</lastName>

<location>USA</location>

</employee>

<employee id="1000">

<firstName>Sunder</firstName>

<lastName>Feezor</lastName>

<location>USA</location>

</employee>

</employees>

**ReadXML.java:**

**import** org.w3c.dom.\*;

**import** javax.xml.parsers.\*;

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** ReadXML {

**public** **static** **void** main(String a[]) **throws** Exception{

DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = factory.newDocumentBuilder();

//Build Document

Document document = builder.parse(**new** File("C:\\Users\\Naresh\\Desktop\\employees.xml"));

//Normalize the XML Structure; It's just too important !!

document.getDocumentElement().normalize();

//Here comes the root node

Element root = document.getDocumentElement();

//Get all employees

NodeList nList = document.getElementsByTagName("employee");

System.*out*.println("enter employee id:");

Scanner s=**new** Scanner(System.*in*);

String id=s.next();

**for** (**int** temp = 0; temp < nList.getLength(); temp++)

{

Node node = nList.item(temp);

**if** (node.getNodeType() == Node.*ELEMENT\_NODE*)

{

Element eElement = (Element) node;

**if**(eElement.getAttribute("id").equals(id)){

System.*out*.println("First Name : " + eElement.getElementsByTagName("firstName").item(0).getTextContent());

System.*out*.println("Last Name : " + eElement.getElementsByTagName("lastName").item(0).getTextContent());

System.*out*.println("Location : " + eElement.getElementsByTagName("location").item(0).getTextContent());

}

}

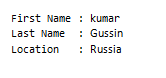
}

}

}

**OUTPUT:-**

Enter Employee Id: 222



**B) Java SAX Parser**

SAX Parser in java provides API to parse XML documents. SAX parser is a lot more different from DOM parser because it doesn’t load complete XML into memory and read XML document sequentially. In SAX, parsing is done by the **ContentHandler** interface and this interface is implemented by **DefaultHandler** class.

Let’s now see an example on extracting data from XML using Java SAX Parser.

Create a java file for SAX parser. In this case, we have created GfgSaxXmlExtractor.java

### student.xml

<?xml version="1.0" encoding="UTF-8"?>

<students-details>

<student>

<studentid>561</studentid>

<name>Ramu</name>

<address>ECIL</address>

<gender>Male</gender>

</student>

<student>

<studentid>562</studentid>

<name>Ramya</name>

<address>KBHP</address>

<gender>Female</gender>

</student>

<student>

<studentid>563</studentid>

<name>Mahi</name>

<address>BHEL</address>

<gender>Male</gender>

</student>

<student>

<studentid>564</studentid>

<name>Manvi</name>

<address>KOTI</address>

<gender>Female</gender>

</student>

<student>

<studentid>565</studentid>

<name>Ammu</name>

<address>ECIL</address>

<gender>Female</gender>

</student>

</students-details>

import java.io.\*;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

public class SAXParserxml extends DefaultHandler

{

boolean studentid = true,name = false,address = false,gender = false;

int flag=0,c=0;

String sid,sname,sadd,sgender,tid;

public void startDocument()

{

System.out.println("begin parsing document");

System.out.print("Enter student ID:\t");

try{

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

tid = reader.readLine();

}catch(Exception e){}

}

public void startElement(String url,String localname, String qName, Attributes att){

if (qName.equalsIgnoreCase("studentid"))

{

studentid = true;

}

else if (qName.equalsIgnoreCase("name") && flag==1)

{

name = true;

}

else if (qName.equalsIgnoreCase("address")&& flag==1)

{

address = true;

}

else if (qName.equalsIgnoreCase("gender")&& flag==1)

{

gender = true;

}

}

public void characters(char[] ch,int start,int length){

if (studentid)

{

String x=new String(ch, start, length);

if(x.equals(tid))

{

flag=1;sid=x; c=1;

}

else

flag=0;

studentid = false;

}

else if (name)

{

sname=new String(ch, start, length); name = false;

}

else if (address)

{

sadd=new String(ch, start, length); address = false;

}

else if (gender)

{

sgender=new String(ch, start, length); gender = false;

}

}

public void endElement(String url,String localname, String qName){}

public void endDocument()

{

if(c==0)

System.out.println("student Id is not present.Try Again!!!");

else

{

System.out.println("\n\n STUDENT-DETAILS");

System.out.println("===================");

System.out.println("student id :\t" +sid);

System.out.println("student Name :\t" +sname);

System.out.println("Adress :\t" +sadd);

System.out.println("Gender :\t" +sgender);

}

}

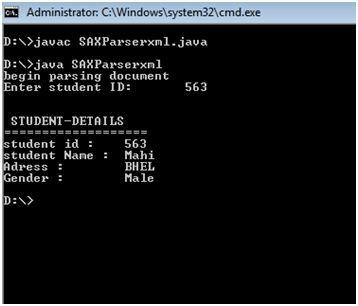
public static void main(String[] arg)throws Exception{

SAXParser p=SAXParserFactory.newInstance().newSAXParser();

p.parse(new FileInputStream("student.xml"), new SAXParserxml());

}

}



**7. A user validation web application, where the user submits the login name and password to the server. The name and password are checked against the data already available in database and if the data matches, a successful login page is returned. Otherwise a failure message is shown to the user.**

**Login.html**

**<html>**

**<head>**

**<title> Login</title>**

**</head>**

**<body>**

**<form action= "database.php" method="post">**

**<div align="center">**

**<b><h3>Member Login</h3> </b></td>**

**<h2>Username:<input type="text" name="Uname" ></h2>**

**<h2>Password:<input type="password" name="pwd"></h2>**

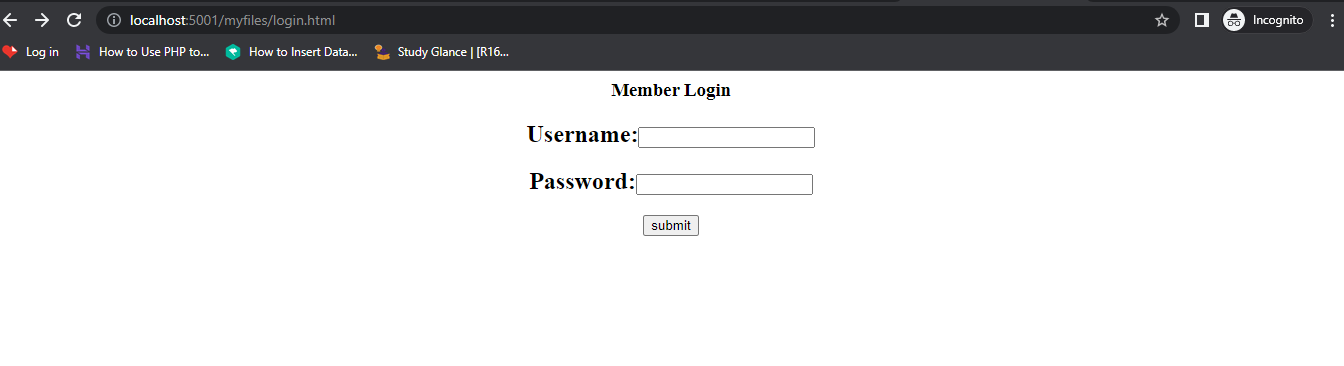
**<input type="submit" value="submit">**

**</div>**

**</form>**

**</body>**

**</html>**

****

**database.php**

**<?php**

**$mysql\_host='localhost';**

**$mysql\_user='root';**

**$mysql\_password='';**

**$name=$\_POST['Uname'];**

**$pwd=$\_POST['pwd'];**

**$conn=@mysqli\_connect($mysql\_host,$mysql\_user,$mysql\_password);**

**if($conn)**

**{**

**if(@mysqli\_select\_db($conn,'login'))**

**{**

**//echo "successfully conected";**

**}**

**else**

**{**

**echo "connection failed to DB";**

**}**

**}**

**else**

**{**

**die('connection error');**

**}**

**$query="select \* from user";**

**$get=mysqli\_query($conn,$query);**

**if($get)**

**{**

**while($retrieve=mysqli\_fetch\_assoc($get))**

**{**

**if($name==$retrieve['Uname'] && $pwd==$retrieve['Password'])**

**{**

**echo "Welcome".$name ;**

**return true;**

**}**

**else{**

**echo "Invalid username or Password";**

**return false;**

**}**

**}**

**}**

**else**

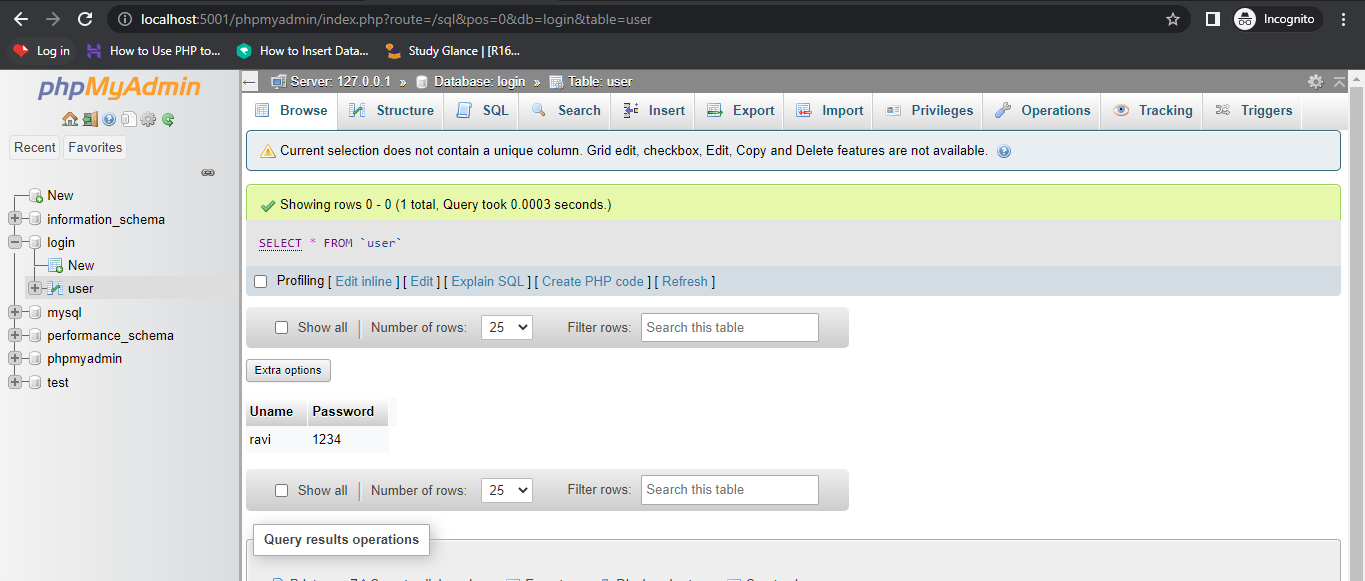
**{**

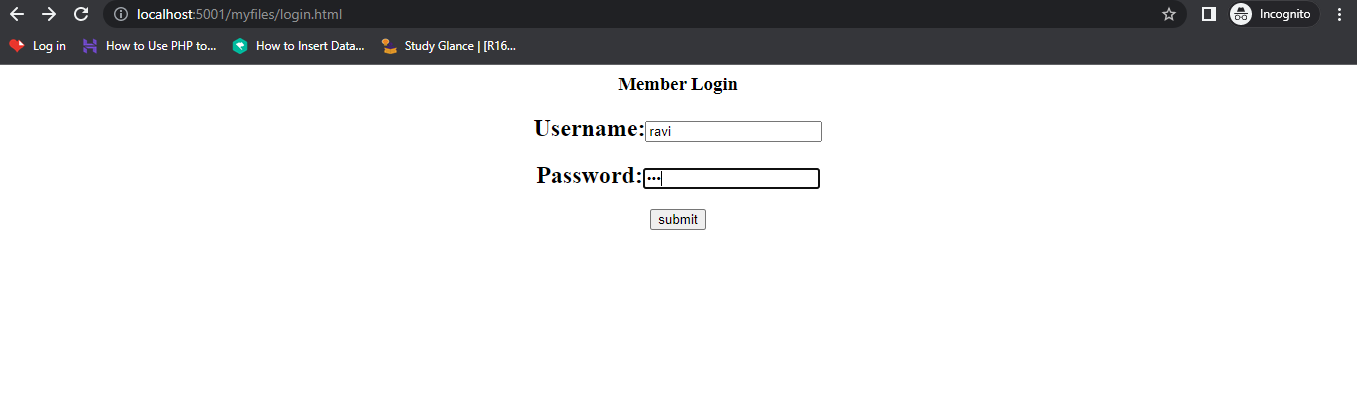
**echo "Query not executed";**

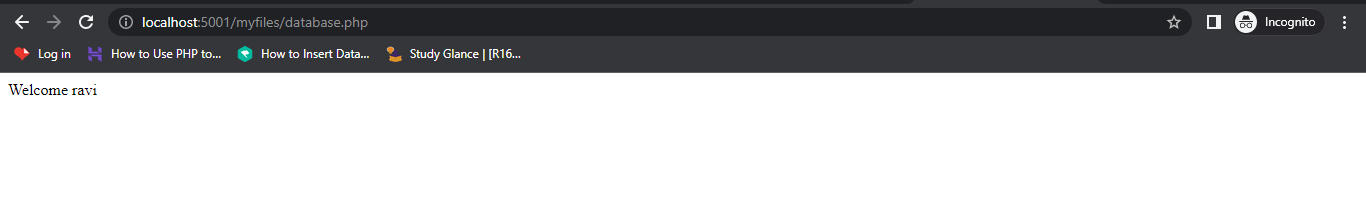
**}**

**?>**

**Output:**

****

****

****