PRACTICAL

FILE

Department: Computer Science and Engineering

Session: January- June

Subject: Web Technology

Subject Code: BTIT605

Semester: 6th

SRI SUKHMANI INSTITUTE OF ENGINEERING & TECHNOLOGY

Affiliated to PTU, & Approved by AICTE

Syllabus

1. Creation of Web pages using HTML, DHTML.
2. Creation of Web pages using JavaScript.
3. Creation of Web pages using AJAX.
4. Creating web pages using PHP.
5. Creating Web pages using ASP.

SRI SUKHMANI INSTITUTE OF ENGINEERING & TECHNOLOGY

Affiliated to PTU, & Approved by AICTE

List of Practical

Sr. No.	Topic
1	Introduction to HTML and XHTML.
2	Basic Tags in HTML.
3	Write a program to create lists.
4	Introduction to CSS.
5	Write a program to create menu using HTML and CSS.
6	Introduction to JavaScript.
7	Write a program to print date using JavaScript.
8	Write a program to Sum and Multiply two numbers using JavaScript.
9	Write a program to Show use of alert, confirm and prompt box.
10	Write a program to redirect, popup and print function in JavaScript..
11	Create validation Form in JavaScript.
12	Introduction o Ajax.
13	Write a program to change content of web page using ajax.
14	Write a program to create XMLHttpRequest .
15	Introduction to php.
16	Write a program to Addition of two numbers using php.
17	Write a program to show data types in php.
18	Write a program to use arithmetic operator in php.
19	Write a program to using class in php.
20	Write a program to connect to database.
21	Write a program to insert data in database.

22	Introduction to asp.
23	Write a program to generate login control.
24	*Write a program to perform validation operation.

*Learning Beyond Syllabus Write a program to perform validation operation.

Experiment 1

AIM: Introduction to HTML and XHTML

HTML is a markup language. It tells the web browser what content to display. HTML separates "content" (words, images, audio, video, and so on) from "presentation" (the definition of the type of content and the instructions for how that type of content should be displayed). HTML uses a pre-defined set of elements to identify content types. Elements contain one or more "tags" that contain or express content. Tags are surrounded by angle brackets, and the "closing" tag (the one that indicates the end of the content) is prefixed by a forward slash.

For example, the paragraph element consists of the start tag " " and the closing tag " ". The following example shows a paragraph contained within the HTML paragraph element:

```
<p>You are beginning to learn HTML.</p>
```

When this content is displayed in a web browser, it looks like this:

You are beginning to learn HTML.

Elements — the basic building blocks

HTML consists of a set of **elements**. Elements define the **semantic** meaning of their content. Elements include everything between two matching element tags, including the tags themselves.

A very simple but complete web page looks like this:

```
<html>
  <body>
    <p> you are in your begining stage of HTM</p>
  </body>
</html>
```

Tags

HTML documents are written in plain text. They can be written in any text editor that allows content to be saved as plain text, such as Notepad, Notepad++, or Sublime, but most HTML authors prefer to use a specialized editor that highlights syntax and shows the DOM. Tag names may be written in either upper or lower case.

This is an an example of *valid* code:

```
<em>I <strong>really</strong> mean that</em>.
```

XHTML

XHTML stands for Extensible HyperText Markup Language and is the next step in the evolution of the Internet. The XHTML 1.0 is the first document type in the XHTML family.

XHTML was developed by the W3C to help web developers make the transition from HTML to XML. By migrating to XHTML today, web developers can enter the XML world with all of its attendant benefits, while still remaining confident in their content's backward and future compatibility.

Developers who migrate their content to XHTML 1.0 will realize the following benefits:

- **XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.**
- **XHTML documents can be written to operate better than they did before in existing browsers as well as in new browsers.**
- **XHTML documents can utilize applications like scripts and applets that rely upon either the HTML Document Object Model or the XML Document Object Model.**

1. **All tag and attribute names must be in lowercase.** Thus, you can't write
   ```
   <A HREF="foo.html">...</A>
   ```
   but must instead write this in lowercase, as :
   ```
   <a href="foo.html">...</a>
   ```

2. **"Empty" tags must be written with an extra slash at the end.** An empty tag is one like `<br>` or `<img src="foo.html">` that doesn't have a `</br>` or `<img>` to end it. In XHTML, such tags must be written as: `<br />`, and `<img src="foo.gif" />`.

3. **You can never omit an end tag.** With HTML, you could sometimes leave off an end tag, as in
   ```
   <p> ..... paragraph text
   <p> ..... more paragraph text
   ```
   With XHTML, you must always put in the end tag, so that the preceding must be written as:
   ```
   <p> ..... paragraph text </p>
   <p> ..... more paragraph text </p>
   ```

4. **Attributes must always have a value.** In HTML you can sometimes omit atttibute values, as in
   ```
   <hr size="2" noshade>
   ```
   in XHTML, this would need to be written as:
   ```
   <hr size="2" noshade="noshade" />
   ```

5. **Attributes values must always be quoted..** In HTML you can sometimes omit the quotes,as in
   ```
   <hr size=2>
   ```
   in XHTML, this would need to be written as:
   ```
   <hr size="2" />
   ```

Experiment 2

AIM: Basic Tags in HTML.

```
<html>
<head>
<title>this is my first page</title>
<body bgcolor=pink>
<h1>welcome to html</h1><br/>
<b>tag for bold</b><br/>
<i>tag for italic</b><br/>
<u>tag for underline</u><br/>
<del>tag for delete</del><br/>
<h2>creating tables</h2>
<table border="2">
<th>table head</th>
<tr>
<td>row1</td>
<td>row2</td>
</tr>
</table>
<h2>creating text box</h2>
<input type="text" name"text">
password
<input type="password" name="pws"><br/>
radio button<br/>
male<input type="radio" name="male">
```

```
female<input type="radio" name="female">
</br>
checkbox<br/>
bike<input type=checkbox" name="bike"><br/>
car<input type="check box" name="car"><br/>
submit button<br/>
<input type="submit" value="submit"><br/>
select box<br/>
<select>
<option>one </option>
<option> two </option>
<option>three </option>
<option>four </option>
</select>
</body>
</html>
```

Experiment 3

AIM: Write a program to create lists..

```
<html>
<body>
<p><b><u> Ordered List</b></u> </p>
<ol>
        <li>a</li>
        <li>s</li>
</ol>
<p><b><u> Unordered List</b></u> </p>
<ul>
        <li>a</li>
        <li>s</li>
</ul>
<p><b><u> Nested List</b></u> </p>
<OL TYPE = A START =3>
        <LI> Fruits
        <OL TYPE = I>
                <LI> Apple
                <LI> MANGO
```

```
            <LI> Orange
```



```
        </OL>
        <LI> VEGETABLES
<OL TYPE = I>
        <LI> Brinjal
        <LI> Cabbage
        <LI> Tomato
</OL>
</OL>
<p><b><u> Definition List </b></u></p>
<dl>
        <dt> Coffee </dt>
            <dd> Black Hot Drink </dd>
            <dt> Milk </dt>
            <dd> White Cold Drink </dd>
</dl>
</body>
</html>
```

Experiment 4

AIM: Introduction to CSS..

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts.[1] This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design

Example:-

```
<html>

<head>

<title>css</title>

<link rel="stylesheet" type="text/css" href="cs.css" />

<style type="text/css">

.di

{
```

```
background-color:#99FF00;
margin-left:300px;
width:100px;
height:100px;
}
.mm
{
background-color:#660099;
width:200px;
height:100px;
}
#mn
{
background-color:#3333FF;
margin-left:500px;
width:300px;
height:300px;
}
#ss
{
```

```
background-color:#3333FF;

width:300px;

height:100px;

}

</style>

</head>


<body>

<div id="container">

<h1>Inline CSS</h1>

<p style="color:#FF0000; margin-left:20px; font-size:14px;
background-color:#CCFF00; width:500px; height:300px">

Inline CSS exaple for background color, font size, margin-
left  and for width.</p>

<h1>internal CSS</h1>

<div class="di">

        <table border="2">

        <tr>

            <td>row1</td>

            <td>row2</td>

        </tr>
```

```
</table>
            </div>


<div class="mm">text box
<input type="text" name="text" style=" color:#00FF33; background-color:#FF0000; font-size:18px">
</div>


    <div id="mn">
        <h1>introduction</h1><br/>
        <b>tag for bold</b><br/>
        <h2>about clg</h2><br/>
        <h3>create options</h3>
        passwords
        <input type="password"rollno="pws"><br/>
    </div><br/>
<br/>
<h1>external css</h1>
<p class="ab">external css</p>
</div>
```

```
<br />
<p id="sm">example for external css </p>


</div>


</body>
</html>
```

Experiment 5

AIM: Write a program to create menu using HTML and CSS..

```
<html>
<head>
<title>menu</title>
<link rel="stylesheet" type="text/css"
 href="style.css" />
</head>

<body>
<ul><li>Home</li>
  <li>About</li>
  <li>
    Portfolio
    <ul>
      <li>Web Design</li>
      <li>Web Development</li>
      <li>Illustrations</li>
    </ul>
  </li>
  <li>Blog
   <ul>
      <li>Web Design</li>
      <li>Web Development</li>
      <li>Illustrations</li>
    </ul></li>
  <li>Contact</li>
          <li>Facilities</li>
     <li>Downloads
     <ul><li>apendix c</li>
          <li>apendix d</li>
          <li>apendix g</li>
     </ul>
     </li>
```

```
    </ul>
    </body>
    </html>
```

External CSS Coding.

```
body {
  font-family: Arial, Helvetica, sans-serif, Helvetica, Arial, sans-serif;
  padding: 20px 50px 150px;
  font-size: 13px;
  text-align: center;
  background: #E3CAA1;
}

ul {
  text-align: left;
  display: inline;
  margin: 0;
  padding: 15px 4px 17px 0;
  list-style: none;
 }
ul li {
  font: bold 12px/18px sans-serif;
  display: inline-block;
  margin-right: -4px;
  position: relative;
  padding: 15px 20px;
  background: #fff;
  cursor: pointer;
}
ul li:hover {
  background: #555;
  color: #fff;
}
ul li ul {
  padding: 0;
  position: absolute;
  top: 48px;
  left: 0;
  width: 150px;
  opacity: 0;
  visibility: hidden;
```

```
}
ul li ul li {
  background: #CCCC33;
  display: block;
  color: #FF0000;
  text-shadow: 0 -1px 0 #000;
}
ul li ul li:hover { background: #9966CC; }
ul li:hover ul {
  display: block;
  opacity: 1;
  visibility: visible;
}
```

Experiment 6

AIM: Introduction to JavaScript..

What Exactly Is JavaScript?

JavaScript is a scripting language designed primarily for adding interactivity to Web pages and creating Web applications. The language was first implemented by Netscape Communications Corp. in Netscape Navigator 2 beta (1995). JavaScript is different from the Java language (developed in the 1990s at Sun Microsystems). However, the two languages can interoperate well. ***Client-side*** JavaScript programs, or scripts, can be embedded directly in HTML source of Web pages. (*Note:* There is also *server-side JavaScript*, but it's beyond the scope of this FAQ collection.) Depending on the Web developer's intent, script code may run when the user opens the Web page, clicks or drags some page element with the mouse, types something on the keyboard, submits a form, or leaves the page.

JavaScript is an object-oriented language with ***prototypal inheritance***. The language supports several built-in objects, and programmers can create or delete their own objects. Prototypal inheritance makes JavaScript very different from other popular programming languages such as C++, C#, or Java featuring *classes* and *classical inheritance*. JavaScript does not have classes in the C++ or Java sense. In JavaScript, objects can inherit properties directly from each other, forming the object prototype chain.

JavaScript is widely supported. It is available in the following browsers:

- Netscape Navigator (beginning with version 2.0)
- Microsoft Internet Explorer (beginning with version 3.0)
- Firefox
- Opera
- Google Chrome

Any other browser whose vendor licensed or implemented JavaScript.

Experiment 7

AIM: Write a program to print date using JavaScript.

```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>
<p>Click the button to display the date.</p>
<p id="demo"></p>

<button type="button" onclick="myFunction()">print
 date</button>

<script>
function myFunction()
{
document.getElementById("demo").innerHTML = Date();
}
</script>

</body>
</html>
```

Experiment 8

AIM: Write a program to Sum and multiply two numbers using JavaScript.

```
<html>
<head>
<script>
      function myFunction() {
        var y = document.getElementById("txt1").value;
        var z = document.getElementById("txt2").value;
        var x = +y + +z;
        document.getElementById("demo").innerHTML = x;
      }
   function mul() {
        var y = document.getElementById("txt1").value;
        var z = document.getElementById("txt2").value;
        var x = +y * +z;
        document.getElementById("demo").innerHTML = x;
      }
    </script>
</head>

  <body>
    <p>Click the button to calculate.</p>

    <br/>
    <br/>Enter first number:
    <input type="text" id="txt1" name="text1"
 size="6px">Enter second number:
    <input type="text" id="txt2" name="text2"
 size="6px"><br>
 <button onClick="myFunction()">SUM</button> 
 <button onClick="mul()">Multiply</button>
    <p id="demo"></p>
    </body>
</html>
```

Experiment 9

AIM: Write a program to Show use of alert, confirm and prompt box.

```
<html>
<head>
<title>java script</title>

<script type="text/javascript">
<!--
function confirmation() {
 var answer = confirm("Welcome Fb.com?")
 if (answer){
      alert("Welcome!")
      window.location =
 "http://www.fb.com/hatemyself90";
 }
 else{
      alert("Bye bye!!")
 }
}
//-->
<!--
function prompter() {
var reply = prompt("What's your name?", "")
alert ( "Nice to see you" + reply + "!")
}
//-->
</script>

</head>
<body>
<h1 align="center"> Java Script Alert, Confirm and
 Prompt Box. </h1>
<form style="height:200px; width:200px; background-
 color:#CC3366;">
<h3> Java Script Alert Box.</h3>
<input type="button" onclick="alert('Are you sure you
 want to give us the deed to your house?')"
```

```
value="Confirmation Alert">
</form>

<form style="height:200px; width:210px; background-
 color:#FF9999; margin-top:-220px; margin-left:300px">
<h3> Java Script Confirm Box.</h3>
<input type="button" onClick="confirmation()" value="Fb
 Login">
</form>
<div style=" margin-top:-220px; background-
 color:#00FF66; height:200px; width:200px; margin-
 left:600px;">
<h3> Java Script Prompt Box.</h3>
<input type="button" onclick="prompter()" value="Say my
 name!">
</div>
</body>
</html>
```

Experiment 10

AIM: Write a program to redirect, popup and print function in JavaScript.

```
<html>
<head>
<title>java script</title>

<script type="text/javascript">
<!--
function delayer(){
   // window.location = "../java/5.html"
 window.location = "4.html"
}
//-->
<!--
function myPopup2() {
window.open( "http://www.google.com/", "myWindow",
"status = 1, height = 300, width = 300, resizable = 0" )
}
//-->
</script>

</head>
<body onLoad="setTimeout('delayer()', 5000)">
<h1> Redirect, Popup and Print Function using java
 script</h1>
<h2>Prepare to be redirected!</h2>
<p>This page is a time delay redirect, please update
 your bookmarks to our new
location!</p>

<form style="height:200px; width:210px; background-
 color:#FF9999;">
<h3> Java Script Popup Box.</h3>
<input type="button" onClick="myPopup2()" value="POP2!">
</form>
<p onClick="myPopup2()">CLICK ME TOO!</p>

<form>
```

```
<input type="button" value="Print This Page"
 onClick="window.print()" />
</form>


</body>
</html>
```

Experiment 11

AIM: Create validation Form in JavaScript..

```
<html>
<head>
<script type='text/javascript'>

function formValidator(){
 // Make quick references to our fields
 var firstname = document.getElementById('firstname');
 var addr = document.getElementById('addr');
 var zip = document.getElementById('zip');
 var state = document.getElementById('state');
 var username = document.getElementById('username');
 var email = document.getElementById('email');

 // Check each input in the order that it appears in the
 form!
 if(isAlphabet(firstname, "Please enter only letters for
 your name")){
       if(isAlphanumeric(addr, "Numbers and Letters Only
 for Address")){
             if(isNumeric(zip, "Please enter a valid zip
 code")){
                   if(madeSelection(state, "Please Choose
 a State")){
                         if(lengthRestriction(username, 6,
 8)){
                               if(emailValidator(email,
 "Please enter a valid email address")){
                                     return true;
                               }
                         }
                   }
             }
       }
 }


 return false;
```

```
}

function notEmpty(joshi, helperMsg){
 if(joshi.value.length == 0){
       alert(helperMsg);
       joshi.focus(); // set the focus to this input
       return false;
 }
 return true;
}

function isNumeric(joshi, helperMsg){
 var numericExpression = /^[0-9]+$/;
 if(joshi.value.match(numericExpression)){
       return true;
 }else{
       alert(helperMsg);
       joshi.focus();
       return false;
 }
}

function isAlphabet(joshi, helperMsg){
 var alphaExp = /^[a-zA-Z]+$/;
 if(joshi.value.match(alphaExp)){
       return true;
 }else{
       alert(helperMsg);
       joshi.focus();
       return false;
 }
}

function isAlphanumeric(joshi, helperMsg){
 var alphaExp = /^[0-9a-zA-Z]+$/;
 if(joshi.value.match(alphaExp)){
       return true;
 }else{
       alert(helperMsg);
       joshi.focus();
       return false;
 }
```

```
}

function lengthRestriction(joshi, min, max){
 var uInput = joshi.value;
 if(uInput.length >= min && uInput.length <= max){
       return true;
 }else{
       alert("Please enter between " +min+ " and " +max+
 " characters");
       joshi.focus();
       return false;
 }
}

function madeSelection(joshi, helperMsg){
 if(joshi.value == "Please Choose"){
       alert(helperMsg);
       joshi.focus();
       return false;
 }else{
       return true;
 }
}

function emailValidator(joshi, helperMsg){
 var emailExp = /^[\w\-\.\+]+\@[a-zA-Z0-9\.\-]+\.[a-zA-
 z0-9]{2,4}$/;
 if(joshi.value.match(emailExp)){
       return true;
 }else{
       alert(helperMsg);
       joshi.focus();
       return false;
 }
}
</script>
</head>
<body>
<form onsubmit='return formValidator()' >
First Name: <input type='text' id='firstname' /><br />
Address: <input type='text' id='addr' /><br />
Zip Code: <input type='text' id='zip' /><br />
State: <select id='state'>
```

```
 <option>Please Choose</option>
 <option>Ambala</option>
 <option>panchkula</option>
 <option>Hisar</option>
 <option>panipat</option>
</select><br />
Username(6-8 characters): <input type='text'
 id='username' /><br />
Email: <input type='text' id='email' /><br />
<input type='submit' value='Check Form' />
</form>
</body>
</html>
```

Experiment 12

AIM: Introduction o Ajax.

AJAX = Asynchronous JavaScript and XML

AJAX is not a new programming language, but a new way to use existing standards.

AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.

AJAX is a technique for creating fast and dynamic web pages.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

In essence, Ajax is an efficient way for a web application to handle user interactions with a web page - a way that reduces the need to do a page refresh or full page reload for every user interaction. This enables rich behavior (similar to that of a desktop application or plugin-based web application) using a browser. Ajax interactions are handled asynchronously in the background. As this happens, a user can continue working with the page. Ajax interactions are initiated by JavaScript code. When the Ajax interaction is complete, JavaScript updates the HTML source of the page. The changes are made immediately without requiring a page refresh. Ajax interactions can be used to do things such as validate form entries (while the user is entering them) using server-side logic, retrieve detailed data from the server, dynamically update data on a page, and submit partial forms from the page.

AJAX is Based on Internet Standards

AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)

Google Suggest

AJAX was made popular in 2005 by Google, with Google Suggest.

Google Suggest is using AJAX to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

As you type in the search box, you can find information quickly by seeing searches that might be similar to the one you're typing. For example, as you start to type [**google**], you may see searches for other popular google-related searches.

Syntax for creating an XMLHttpRequest object:

```
variable=new XMLHttpRequest();
```

Old versions of Internet Explorer (IE5 and IE6) uses an ActiveX Object:

variable=new ActiveXObject("Microsoft.XMLHTTP");

Experiment 13

AIM: Write a program to change content of web page using ajax..

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc()
{
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>

</body>
</html>
```

Experiment 14

AIM: Write a program to create XMLHttpRequest.

variable=new XMLHttpRequest();

Old versions of Internet Explorer (IE5 and IE6) uses an ActiveX Object:

variable=new ActiveXObject("Microsoft.XMLHTTP");

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc()
{
var xmlhttp;
if (window.XMLHttpRequest)
 {// code for IE7+, Firefox, Chrome, Opera, Safari
 xmlhttp=new XMLHttpRequest();
 }
else
 {// code for IE6, IE5
 xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
 }
xmlhttp.onreadystatechange=function()
```

```
 {
 if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
  document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
  }
 }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>

</body>
</html>
```

Experiment 15

AIM: Introduction to php.

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the UNIX side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common uses of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- You add, delete, modify elements within your database thru PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Experiment 16

AIM: Write a program to Addition of two numbers using php.

```
<html>
<head>
<title>Addition of Numbers Using Forms</title>
</head>
<?php
 error_reporting(0);

$value1 = trim($_REQUEST['val_1']);
$value2 = trim($_REQUEST['val_2']);

// Clear the text field.

if ($_REQUEST['submit2']){
   $value1 = "";
        $value2 = "";
}
?>

<h2><center> Sum of Two Numbers </center> </h2>

<form method="post" action="">
        <label>Enter First Value</label><br />
        <input type="text" name="val_1" value="<?php echo $value1; ?>"><br />
        <label>Enter Second Value</label><br />
        <input type="text" name="val_2" value="<?php echo $value2; ?>"><br />

        <br>
   <input type="submit" name="submit" value="Add">
        <input type="submit" name="submit2" value="Clear">
</form>
<?php
if ($_REQUEST['submit']){

        $add = ($value1 + $value2);

        echo "<h3>The sum of  " .$value1." and ".$value2." is ";
```

```
        echo $add.".";


}

?>

</body>
</html>
```

Experiment 17

AIM: Write a program to show data types in php.

```
<html>
  <head>
    <title>PHP data types</title>
  </head>

  <body>

    <?php

      // declare a string, double and integer
      $testString = "3.5 seconds";
      $testDouble = 79.2;
      $testInteger = 12;
    ?>

    <!-- print each variable's value -->
    <?php print( $testString ); ?> is a string.<br />
    <?php print( $testDouble ); ?> is a double.<br />
    <?php print( $testInteger ); ?> is an integer.<br />

    <br />
    Now, converting to other types:<br />
    <?php

      // call function settype to convert variable
      // testString to different data types
      print( "$testString" );
      settype( $testString, "double" );
      print( " as a double is $testString <br />" );
      print( "$testString" );
      settype( $testString, "integer" );
      print( " as an integer is $testString <br />" );
      settype( $testString, "string" );
      print( "Converting back to a string results in
        $testString <br /><br />" );
```

```
      $data = "98.6 degrees";

      // use type casting to cast variables to a
      // different type
      print( "Now using type casting instead: <br />
         As a string - " . (string) $data .
         "<br />As a double - " . (double) $data .
         "<br />As an integer - " . (integer) $data );
   ?>
  </body>
</html>
```

Experiment 18

AIM: Write a program to use arithmetic operator in php.

```
<html >
  <head>
    <title>Using arithmetic operators</title>
  </head>

  <body>
    <?php
      $a = 5;
      print( "The value of variable a is $a <br />" );

      // define constant VALUE
      define( "VALUE", 5 );

      // add constant VALUE to variable $a
      $a = $a + VALUE;
      print( "Variable a after adding constant VALUE
        is $a <br />" );

      // multiply variable $a by 2
      $a *= 2;
```

```
print( "Multiplying variable a by 2 yields $a <br />" );

// test if variable $a is less than 50
if ( $a < 50 )
   print( "Variable a is less than 50 <br />" );

// add 40 to variable $a
$a += 40;
print( "Variable a after adding 40 is $a <br />" );

// test if variable $a is 50 or less
if ( $a < 51 )
   print( "Variable a is still 50 or less<br />" );

// test if variable $a is between 50 and 100, inclusive
elseif ( $a < 101 )
   print( "Variable a is now between 50 and 100,
      inclusive<br />" );
else
   print( "Variable a is now greater than 100
      <br />" );

// print an uninitialized variable
            print( "Using a variable before initializing:
```

```
      <br />" );

      // add constant VALUE to an uninitialized variable
      $test = $a + VALUE;
      print( "An uninitialized variable plus constant
         VALUE yields $test <br />" );

      // add a string to an integer
      $str = "3 dollars";
      $a += $str;
      print( "Adding a string to variable a yields $a
         <br />" );
   ?>
  </body>
</html>
```

Experiment 19

AIM: Write a program to using class in php.

```
<?php

    class Programmer {

        // Class Properties

        var $name;          // Programmer's name

        var $experience;    // How long has been programming

        var $lang;          // Favorite Language

        var $education;     // Highest degree earned

        function Programmer($name, $experience, $lang, $education) {

            $this->name=$name;

            $this->experience=$experience;

            $this->lang=$lang;

            $this->education=$education;

        }
         function get_name() {

            return $this->name;

        }

        function set_name($newname) {
```

```
    $this->name = $newname;
}
function get_experience() {
    return $this->experience;
}
function set_experience($newexperience) {
    $this->experience = $newexperience;
}
function get_lang() {
    return $this->lang;
}
function set_lang($newlang) {
    $this->lang = $newlang;
}
function get_education() {
    return $this->education;
}
function set_education($neweducation) {
```

```
        $this->education = $neweducation;
    }
    function output() {
        echo "Programmer Name: ".$this->name."<br>";
        echo $this->name." has ".$this->experience." years of programming
experience.<br>";
        echo $this->lang." is ".$this->name."'s favorite programming language.<br>";
        echo $this->name." holds the degree: ".$this->education."<br><br>";
    }
}
// Instantiating a programmer
$paul = new Programmer('Pardeep Joshi',4,'C++','MCA');
$paul->output();

$paul->set_experience(7);
$paul->output();
?>
```

Experiment 20

AIM: Write a program to connect to database.

```
<?php
        $con=mysql_connect("localhost","root","");

if($con)
{
echo"connected";
echo "<br>";
}
else
{
echo "not connected";
}
?>
```

Experiment 21

AIM: Write a program to insert data in database.

```
<?php
        require_once("connect.php");
            error_reporting(0);
            if($_POST['submit'])
            {
$fn=$_POST['name'];
$rn=$_POST['roll'];
$sn=$_POST['sem'];
mysql_select_db("exmp",$con);
$query="INSERT INTO exmp1(name,roll,sem) VALUES ('$fn','$rn','$sn')";
if(mysql_query($query,$con))
{
header("Location:test.php");
}
else
{
echo"die";
}
}
?>
<html>
<head>
<body>
```

```
<form method="post" action="insert.php">
First name:<input type="text" name="name"><br/><br/>
stud  roll:<input type="text" name="roll"><br/><br/>
stud  sem :<input type="text" name="sem"><br/><br/>
<input type="submit" name="submit" value="submit"><br/>
</form>

</body>
</head>
</html>
```

Experiment 22

AIM: **Introduction to asp.**

ASP stands for Active Server Pages. Microsoft introduced Active Server Pages in December 1996, beginning with Version 3.0. Microsoft officially defines ASP as: "Active Server Pages is an open, compile-free application environment in which you can Combine HTML, scripts, and reusable ActiveX server components to create dynamic and Powerful Web-based business solutions. An active Server page enables server side scripting For IIS with native support for both VBScript and Jscript." (2). In other words, ASP is a Microsoft technology that enables you to create dynamic web sites with the help of server Side script, such as VBScript and Jscript. ASP technology is supported on all Microsoft Web servers that is freely available. If you have Window NT 4.0 Server installed, you Can download IIS (Internet Information Server) 3.0 or 4.0. If you are using Window 2000, IIS 5.0 comes with it as a free component. If you have Window 95/98, you can Download Personal Web Server (PWS), which is a smaller version of IIS, from Window 95/98 CD. You can also download these products for free from Microsoft's web site. Well, you have learned what the ASP technology is. Next, you will learn about an ASP file. Is it the same as HTML file? Let's explore it.

What is an ASP file?

An ASP file is quite like an HTML file. It contains text, HTML tags and scripts,

Which are executed on the server? The two widely used scripting languages for an ASP Page are VBScript and JScript. VBScript is pretty much like Visual Basic, whereas Jscript is the Microsoft's version of JavaScript. However, VBScript is the default Scripting language for ASP (3). Besides these two scripting languages, you can use other scripting language with ASP as long as you have an ActiveX scripting engine for the language installed, such as PerlScript. The difference between an HTML file and an ASP file is that an ASP file has the ".asp" extension. Furthermore, script delimiters for HTML tags and ASP code are also different. A script delimiter is a character that marks the starting and ending of a unit. HTML tags begins with

lesser than (<) and greater than (>) brackets, whereas ASP script typically starts with <% and ends with %>. In between the delimiters are the server-side scripts. To write an ASP script, you don't need any additional software because it can be written with any HTML editor, such as Notepad. Nonetheless, if you feel bored with the plain text and would like to use some special software, you can use Microsoft visual InterDev, which helps you to easily create an ASP page by giving you nice highlights and debugging dialogue boxes (4). I hope that you already have an idea of what an ASP file is and how it is different from an HTML file. In the next step, you will learn how ASP works. Let's go

SRI SUKHMANI INSTITUTE OF ENGINEERING & TECHNOLOGY

Affiliated to PTU, & Approved by AICTE

Experiment 23

AIM: Write a program to generate login control.

Source:

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default.aspx.vb"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN""http:]//www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Login ID="Login1" runat="server" Style="z-index: 100; left: 0px;
position: absolute;
top: 0px">
</asp:Login>
</div>
</form>
</body>
</html>
```

Code:

```
Partial Class _Default

Inherits System.Web.UI.Page

Protected Sub Login1_Authenticate(ByVal sender As Object,

ByVal e As
System.Web.UI.WebControls.AuthenticateEventArgs) Handles
Login1.Authenticate

If Login1.UserName = "Database" And Login1.Password = "Jaiswal" Then
MsgBox("You are successfully Logged in")

Else

MsgBox("Error:Loggedin")

End If

If Application("i") = 3 Then
MsgBox("You are Blocked")
Login1.Enabled = False

End If

End Sub

Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
Application("i") = Int(Application("i") + 1)
If Application("i") > 3 Then
Application("i") = 0

End If
End Sub
End Class
```

Global Application:

```
<%@ Application Language="VB" %>

<script runat="server">

Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
' Code that runs on application startup
Application("i") = 0

End Sub

Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
' Code that runs on application shutdown
End Sub


Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
' Code that runs when an unhandled error occurs

End Sub


Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
' Code that runs when a new session is started

End Sub


Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
' Code that runs when a session ends.
' Note: The Session_End event is raised only when the sessionstate
mode
' is set to InProc in the Web.config file. If session mode is set to
StateServer
' or SQLServer, the event is not raised.

End Sub


</script>
```

Experiment 24

AIM: Write a program to perform validation operation.

Source:

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default.aspx.vb"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Label ID="Label1" runat="server" Style="z-index: 100; left:
384px; position: absolute;
top: 16px" Text="Registation form"></asp:Label>

<asp:Label ID="Label2" runat="server" Style="z-index: 101; left:
248px; position: absolute;
top: 72px" Text="Name"></asp:Label>

<asp:Label ID="Label3" runat="server" Style="z-index: 102; left:
248px; position: absolute;
top: 120px" Text="Reg_no"></asp:Label>

<asp:Label ID="Label4" runat="server" Style="z-index: 103; left:
240px; position: absolute;
top: 176px" Text="Date_Of_Birth"></asp:Label>
```

```
<asp:Label ID="Label5" runat="server" Style="z-index: 104; left:
240px; position: absolute;
top: 232px" Text="Department"></asp:Label>

<asp:Label ID="Label6" runat="server" Style="z-index: 105; left:
248px; position: absolute;
top: 296px" Text="Address"></asp:Label>

<asp:Label ID="Label7" runat="server" Style="z-index: 106; left:
240px; position: absolute;
top: 352px" Text="Phone number"></asp:Label>

<asp:Label ID="Label8" runat="server" Style="z-index: 107; left:
240px; position: absolute;
top: 392px" Text="personal phone no"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server" Style="z-index: 108; left:
456px; position: absolute;
top: 72px" AutoPostBack="True"></asp:TextBox>

<asp:TextBox ID="TextBox2" runat="server" Style="z-index: 109; left:
456px; position: absolute;
top: 120px" AutoPostBack="True"></asp:TextBox>

<asp:TextBox ID="TextBox3" runat="server" Style="z-index: 110; left:
456px; position: absolute;
top: 176px" AutoPostBack="True"></asp:TextBox>

<asp:TextBox ID="TextBox4" runat="server" Style="z-index: 111; left:
456px; position: absolute;
top: 232px" AutoPostBack="True"></asp:TextBox>

<asp:TextBox ID="TextBox5" runat="server" Style="z-index: 112; left:
456px; position: absolute;
top: 288px" AutoPostBack="True"></asp:TextBox>

<asp:TextBox ID="TextBox6" runat="server" Style="z-index: 113; left:
448px; position: absolute;
top: 392px" AutoPostBack="True"></asp:TextBox>

<asp:TextBox ID="TextBox7" runat="server" Style="z-index: 114; left:
448px; position: absolute;
top: 448px" AutoPostBack="True"></asp:TextBox>

<asp:Label ID="Label9" runat="server" Style="z-index: 115; left:
```

```
248px; position: absolute;
top: 448px" Text="Home phone no"></asp:Label>

<asp:Label ID="Label10" runat="server" Style="z-index: 116; left: 256px;
position: absolute;
top: 512px" Text="Email_id"></asp:Label>

<asp:TextBox ID="TextBox8" runat="server" Style="z-index: 117; left:
448px; position: absolute;
top: 504px" AutoPostBack="True"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox1"
ErrorMessage="RequiredFieldValidator" Style="z-index: 118; left:
672px; position: absolute;
top: 72px">Must enter name</asp:RequiredFieldValidator>

<asp:RangeValidator ID="RangeValidator1" runat="server"
ErrorMessage="RangeValidator"
Style="z-index: 119; left: 672px; position: absolute; top: 120px"
ControlToValidate="TextBox2" MaximumValue="35208182"
MinimumValue="35208001">Must be enter between 35208001 to
35208182</asp:RangeValidator>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2"
runat="server" ControlToValidate="TextBox4"
ErrorMessage="RequiredFieldValidator" Style="z-index: 120; left:
672px; position: absolute;
top: 232px">Must enter dept</asp:RequiredFieldValidator>

<asp:RequiredFieldValidator ID="RequiredFieldValidator3"
runat="server" ControlToValidate="TextBox3"
ErrorMessage="RequiredFieldValidator" Style="z-index: 121; left:
680px; position: absolute;
top: 184px">Must enter date of birth</asp:RequiredFieldValidator>
 

<asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox6"
ControlToValidate="TextBox7" ErrorMessage="CompareValidator"
Style="z-index: 122;
left: 664px; position: absolute; top:
400px"></asp:CompareValidator>

<asp:CustomValidator ID="CustomValidator1" runat="server"
ControlToValidate="TextBox5"
```

```
ErrorMessage="CustomValidator" Style="z-index: 123; left: 664px;
position: absolute;
top: 288px">Must enter address</asp:CustomValidator>

<asp:Button ID="Button1" runat="server" Style="z-index: 124; left:
392px; position: absolute;
top: 576px" Text="Register" Width="104px" />

<asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ControlToValidate="TextBox8"
Display="None" ErrorMessage="RegularExpressionValidator"
Style="z-

index: 126;
left: 656px; position: absolute; top: 512px"
ValidationExpression="(\(\d{3}\)|\d{3}
)?\d{8}"></asp:RegularExpressionValidator>

</div>

</form>

</body>

</html>
```

Code:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click

If IsValid Then
Label9.Enabled = False
TextBox7.Enabled = False


End If
```