**EXPERIMENT – 1**

**AIM:** Prepare a stepwise instruction set to configure Apache/XAMPP/WAMPP web server for web application. Ensure that the server includes Apache.

**THEORY:**

Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).

Basically, web servers are computers used to store HTTP files which makes a website and when a client requests a certain website, it delivers the requested website to the client.

Different websites can be stored on the same or different web servers but that doesn’t affect the actual website that you are seeing in your computer. The web server can be any software or hardware but is usually a software running on a computer.

One web server can handle multiple users at any given time which is a necessity otherwise there had to be a web server for each user and considering the current world population, is nearly close to impossible. A web server is never disconnected from the internet because if it was, then it won’t be able to receive any requests, and therefore cannot process them.

There are many web servers available in the market both free and paid. Some of them are :

1. Apache
2. XAMPP
3. WAMP
4. GWS
5. IIS

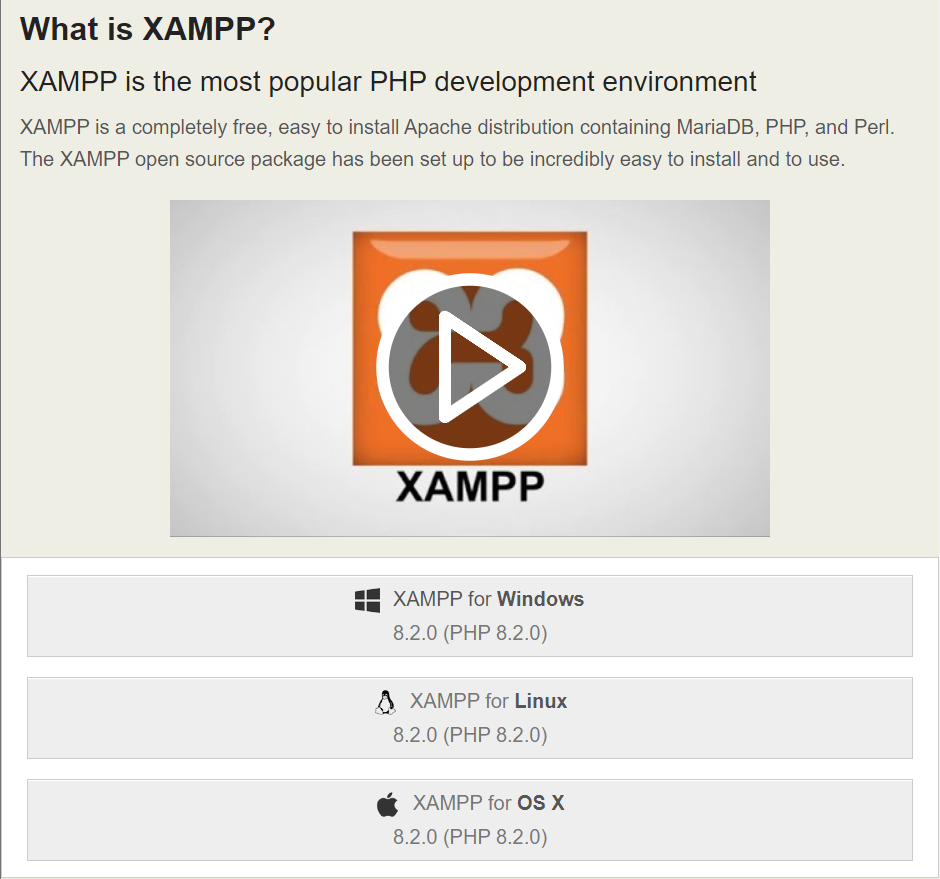
* **What is XAMPP?**

XAMPP is an abbreviation where X stands for Cross-Platform, A stands for Apache, M stands for MYSQL, and the Ps stand for PHP and Perl, respectively. It is an open-source package of web solutions that includes Apache distribution for many servers and command-line executables along with modules such as Apache server, MariaDB, PHP, and Perl.

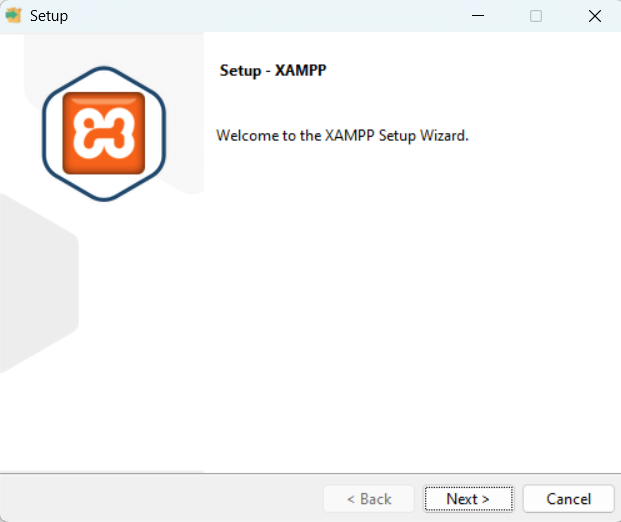
* **Advantages of XAMPP**
* It is easy to set up and use.
* XAMPP is a free and open-source cross-platform web server solution stack package for all types of operating systems like Linux and Windows.
* It has many other essential modules like phpMyAdmin, OpenSSL, MediaWiki, WordPress, Joomla, and more.
* It allows the users to start and end the entire web server + database stack with just one command.

**Steps To Install XAMPP:**

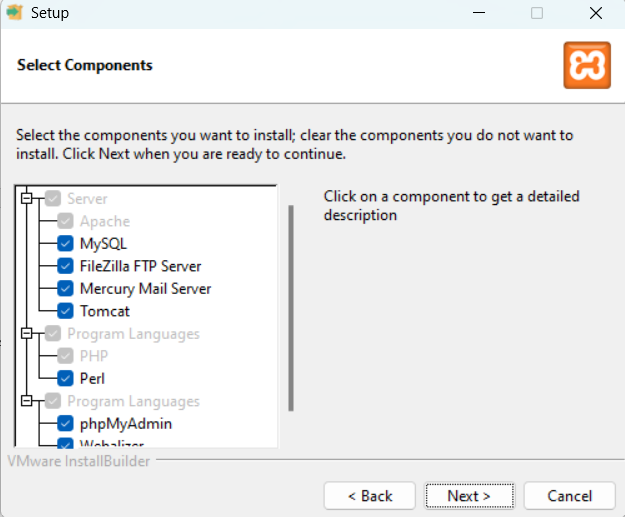
1. Go to **Apachefriends.org** and click on latest XAMPP version to Install.

****

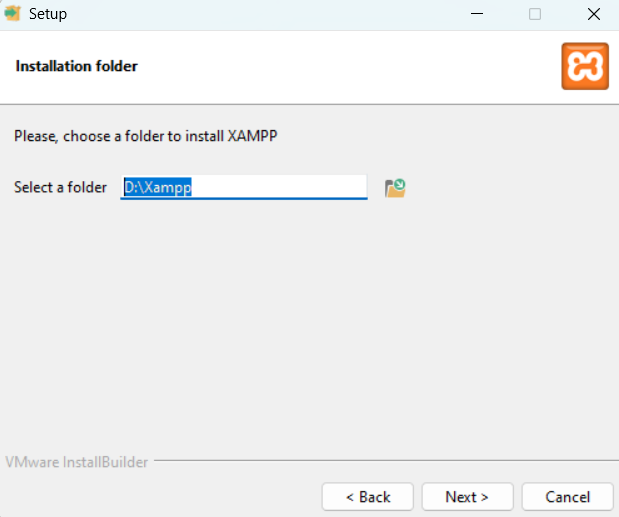
1. After Installation is Complete go to the specified location where XAMPP setup file is installed and double click on the file to install It.

****

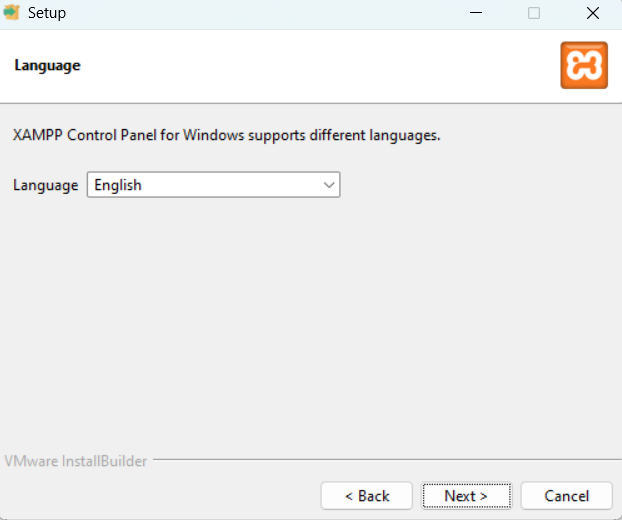
1. Click on **Next**.



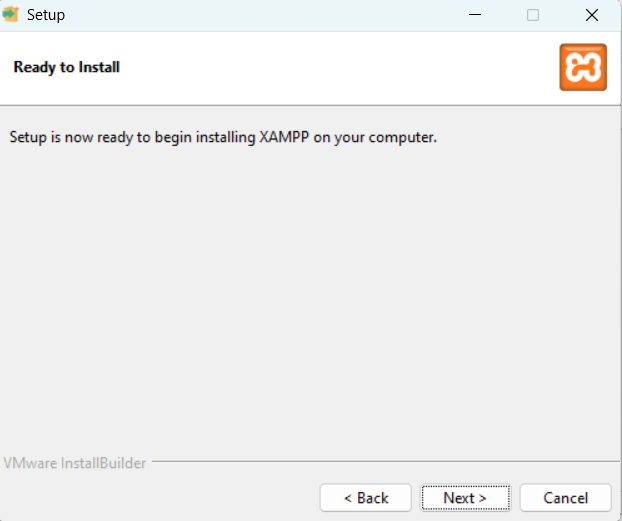
1. Click on **Next** and Choose specific location where you want to Install XAMPP files.

****

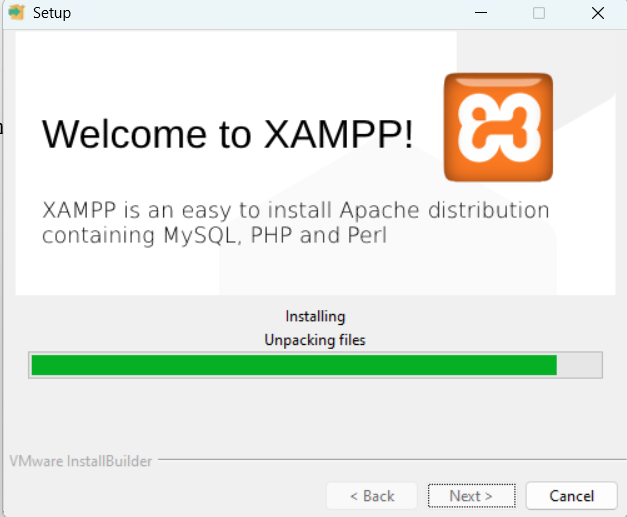
1. Click on **Next** and Choose the Language.



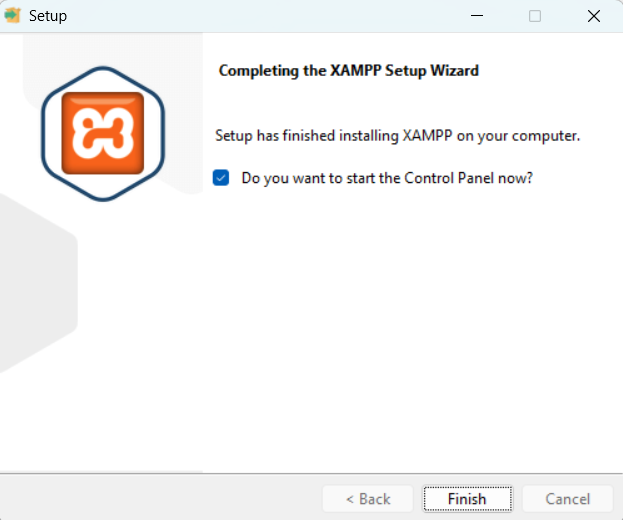
1. Click on **Next.**

****

1. Click on **Next** and then Installation of XAMPP started.

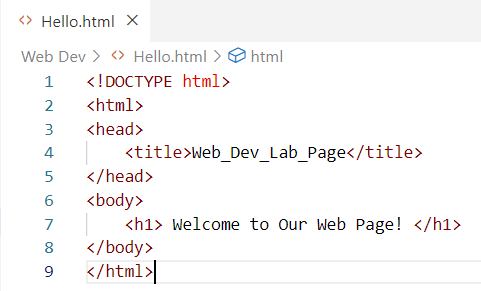
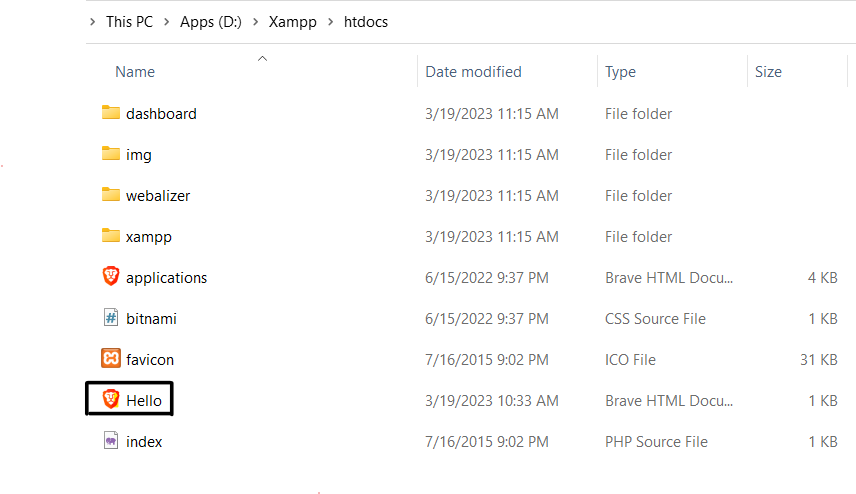


1. After Installation is Complete Click on **Finish.**

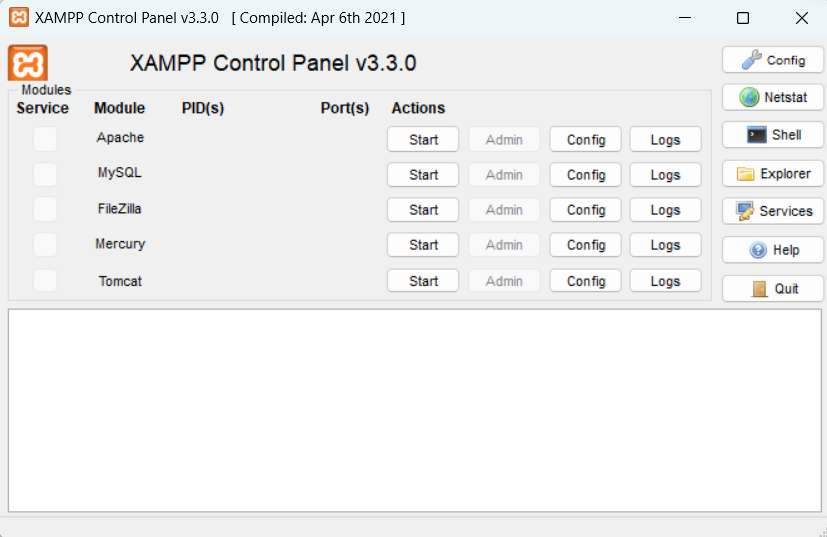


**Steps to Open Html file in XAMPP :**

1. First create a Html file and then paste it into the **htdocs** folder inside the folder where you install the XAMPP.

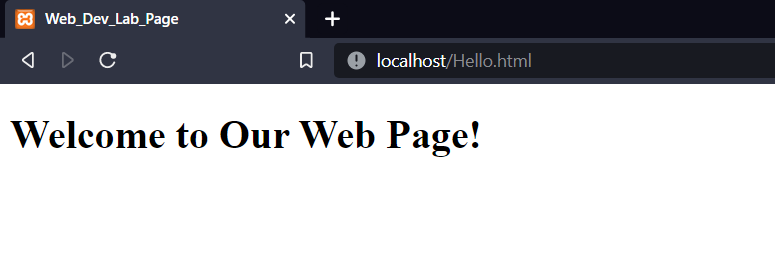
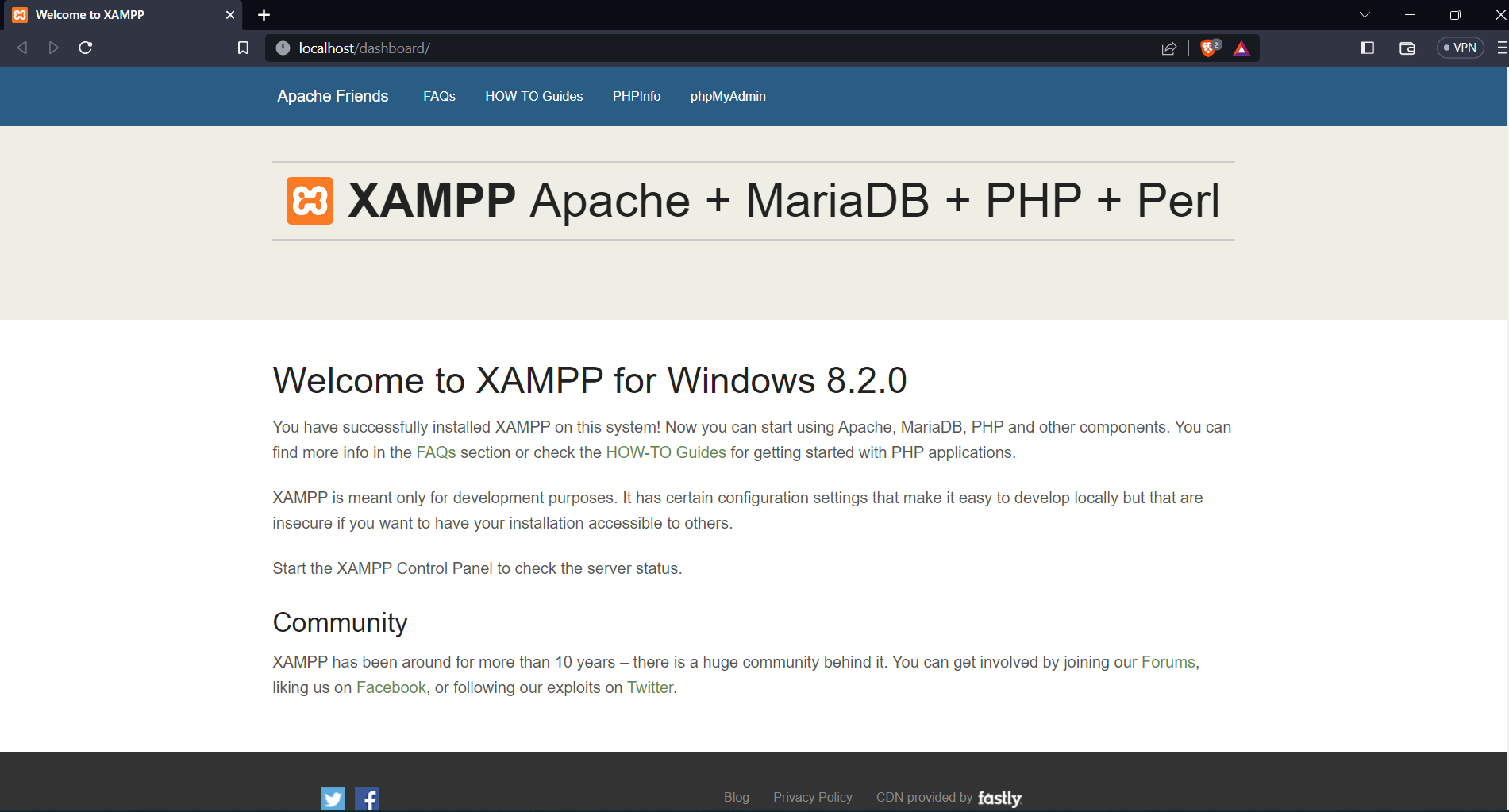


1. Now Open **XAMPP Control Panel.**



1. Now Click on **Start** button In front of Apache and then Click on **Admin.** Then a XAMPP Web page is open With localhost/dashboard.
2. Now Replace dashboard with the file name of the HTML file you save in XAMPP folder as

**localhost/Hello.html** then press **Enter** your web is successfully open in XAMPP.

****

**EXPERIMENT – 2**

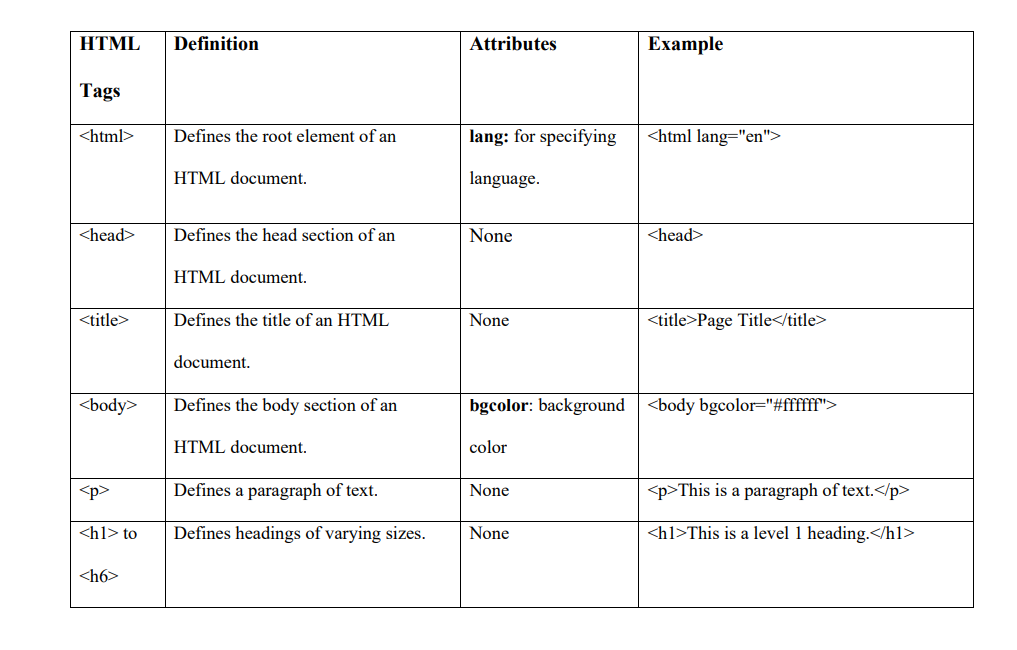
**AIM:** Design a Web Page using basic Html Tags.

**THEORY:**  HTML tags are like keywords which defines that how web browser will format and display the content. With the help of tags, a web browser can distinguish between an HTML content and a simple content. HTML tags contain three main parts: opening tag, content and closing tag.

An HTML file must have some essential tags so that web browser can differentiate between a simple text and HTML text. You can use as many tags you want as per your code requirement.

* All HTML tags must enclosed within < > these brackets.
* Every tag in HTML perform different tasks.
* If you have used an open tag <tag>, then you must use a close tag </tag> (except some tags).

**Syntax:** <tag> content </tag>.

Some of the basic HTML tags are:

****

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>All Tag page</title>

</head>

<body>

<CENTER>

<h1> Welcome To My Web Page </h1>

</CENTER>

<p>Hello, there this is a Simple Web page in which i will make use of all the basic tags in <b>HTML.</b>

</p>

<hr>

</hr>

<CENTER>

<h2>Self Introduction</h2>

</CENTER>

<dl>

<p> Now, i will provide a little introduction about myself using <b>Description list</b> tag.

</p>

<dt>

<h3>EDUCATION</h3>

</dt>

<dd>I have completed my 12th from XXXXX</dd>

<dd>I am currently pursuing B.tech from Maharaja Agarsen Institute of Technology.</dd>

<dt>

<h3>HOBBIES</h3>

</dt>

<dd>I like to do coding</dd>

<dt>

<h3>ADDRESS</h3>

</dt>

<CENTER>

<address>

Address: Rohini New Delhi

</address>

</CENTER>

</dl>

<hr>

</hr>

<CENTER>

<h2> Heading Tags </h2>

</CENTER>

<ul>

<p> Here I am Showing All the Different heading tags using <b>Unordered List</b> tag.</p>

<li>

<h3><u>Heading 3 : </u></h3>

<p>This is Heading 3 Tag </p>

</li>

<li>

<h4><u>Heading 4 : </u></h3>

<p>

This is Heading 4 Tag

</p>

</li>

<li>

<h5><u>Heading 5 : </u></h3>

<p>

This is Heading 5 Tag

</p>

</li>

<li>

<h6><u>Heading 6 : </u></h3>

<p>

This is Heading 6 Tag

</p>

</li>

<p>

So these are all the different Heading tags.

</p>

<hr>

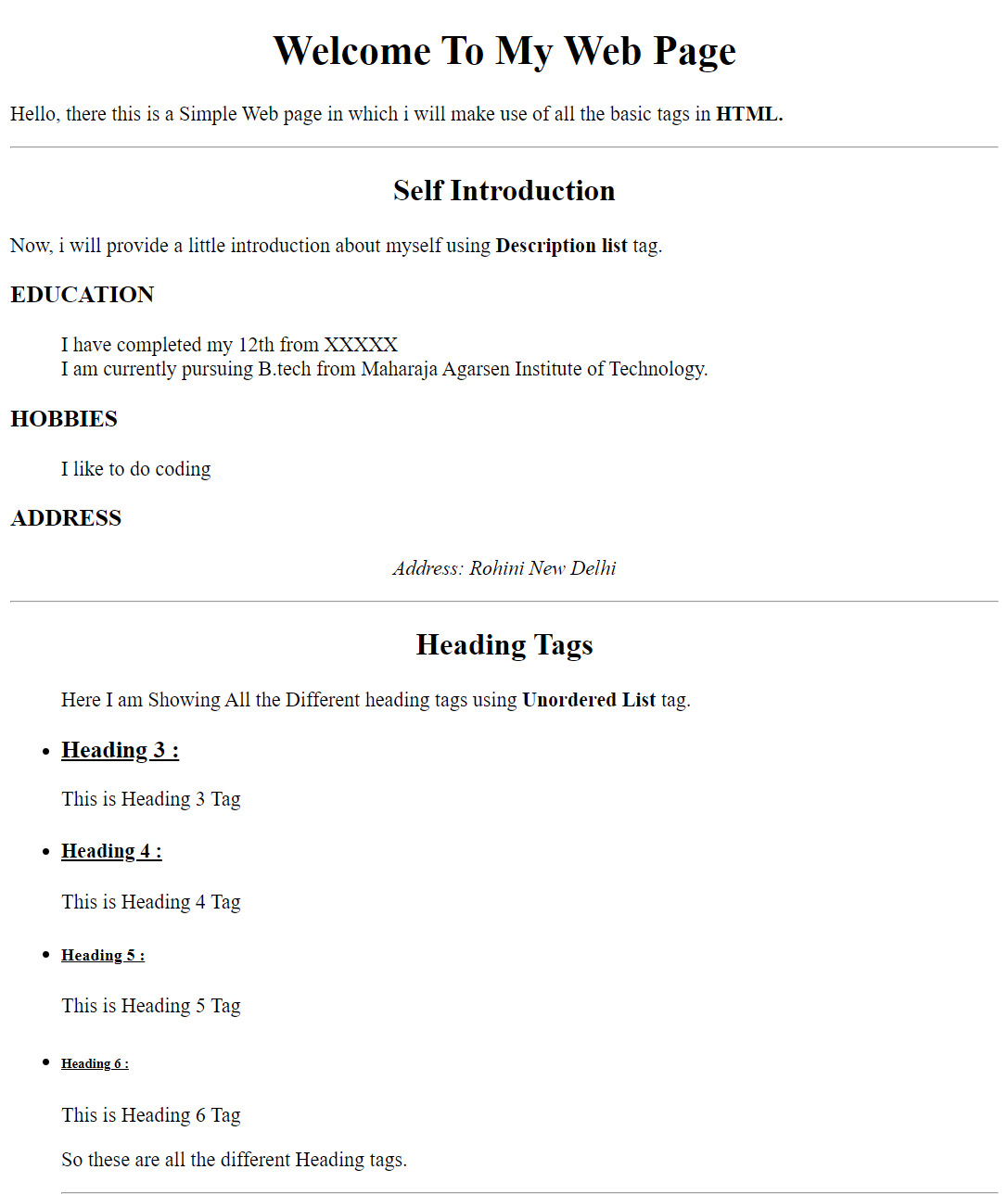
</hr>

</ul>

</body>

</html>

**OUTPUT:**



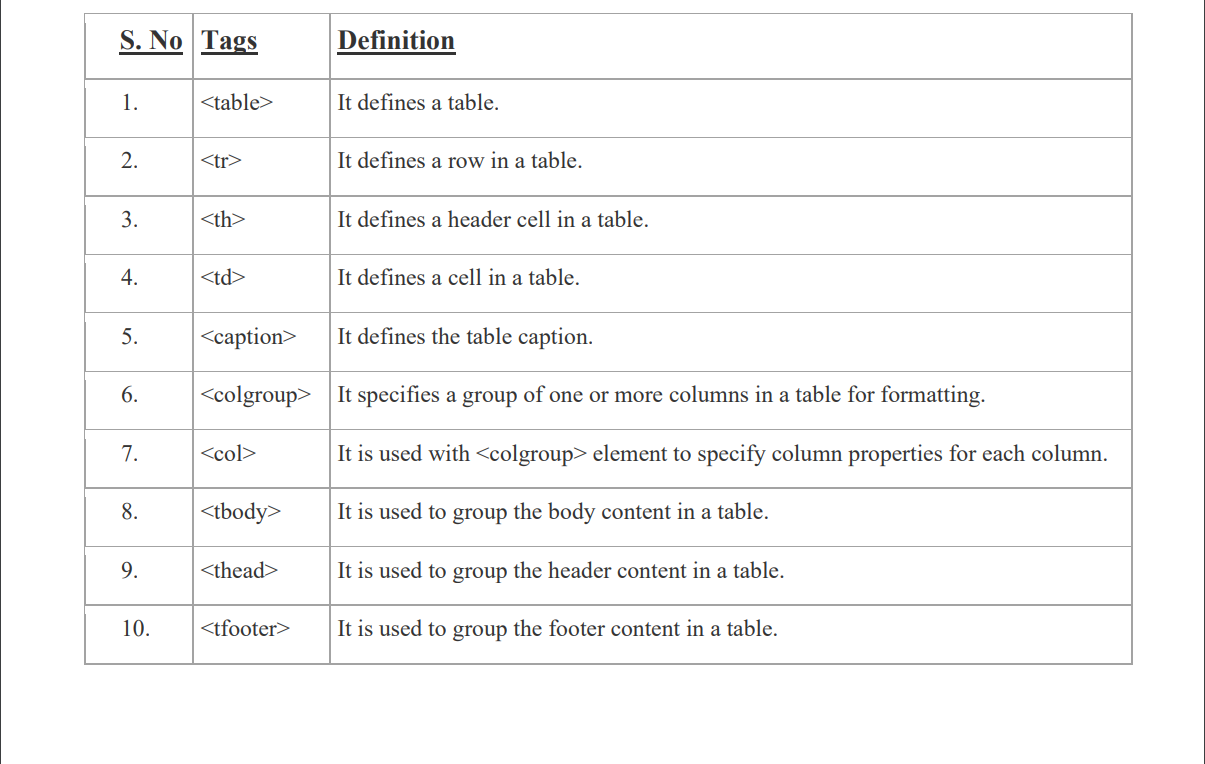
**EXPERIMENT – 3**

**AIM:** Design a web-page using Tables and Forms.

**THEORY:**

**HTML TABLE:**

* Tables are used to display data in rows and columns.
* Tables are created using the <table> tag, and individual cells are created using the <td> tag.
* Tables can also have header rows, which are created using the <th> tag.
* Tables can be styled using CSS to customize the appearance of the table, including the border, background, font, and spacing.
* Tables can be nested within other tables to create more complex layouts.
* Tables can also have caption elements, which are created using the <caption> tag.
* Tables can be accessed and manipulated using JavaScript and jQuery.
* Tables can be used for a variety of purposes, including displaying data, creating calendars, and organizing layouts.

Elements of a **HTML TABLE** are:

**HTML FORMS:**

An **HTML** form is a section of a document which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

Elements for **HTML FORMS** are:

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>All Tag page</title>

</head>

<body>

<CENTER>

<h2> Experiment - 3 </h2>

<h2> HTML TABLE: </h2>

<table border="1px" cellpadding="10px" cellspacing="0px" width="80%">

<caption align="top">HTML Table tags and Definition :- </caption>

<tr>

<th>S.No</th>

<th>Tag Name</th>

<th>Definition</th>

</tr>

<tr>

<td>1.</td>

<td>< table ></td>

<td>Defines a <b>Table</b></td>

</tr>

<tr>

<td>2.</td>

<td>< caption ></td>

<td>Used to define <b>Caption</b> of Table</td>

</tr>

<tr>

<td>3.</td>

<td>< tr ></td>

<td>Create a <b>Row</b> in a table</td>

</tr>

<tr>

<td>4.</td>

<td>< th ></td>

<td>Defines a <b>Header Rows</b> in a table</td>

</tr>

<tr>

<td>5.</td>

<td>< td ></td>

<td>Defines <b>Each Column</b> in a table</td>

</tr>

<tr>

<td>6. </td>

<td>border</td>

<td>Used to add <b>Border</b> in a Table </td>

</tr>

<tr>

<td>7. </td>

<td>cellpadding</td>

<td>Specifies the <b>Distance between Data & Boundaries of a Cell</b> </td>

</tr>

<tr>

<td>8. </td>

<td>cellspacing</td>

<td>Used to specify <b>Space between Adjacent Cell</b> </td>

</tr>

<tr>

<td>9. </td>

<td>width</td>

<td>Used to specify <b>width</b> of table. </td>

</tr>

</table>

</CENTER>

<CENTER>

<h2> <u> HTML Form</u> </h2>

<form action="">

<table border="1px" cellpadding="10px" cellspacing="0px" width = "80%">

<tr>

<td> <label for="name">Name:</label> </td>

<td> <input type="text" placeholder="Enter your name here"></td> </tr>

<tr>

<td> <label for="pwd">Password:</label> </td>

<td> <input type="password" placeholder="Enter your password here" ></td>

</tr>

<tr>

<td> <label for="email">E-mail:</label> </td>

<td> <input type="email" placeholder="Enter your email here"></td>

</tr>

<tr>

<td> <label for="date">Date:</label> </td>

<td> <input type="date"></td>

</tr>

<tr>

<td> <label for="phoneno">Phone No:</label> </td>

<td> <input type="tel"></td>

</tr>

<tr>

<td> <label for="gender">Gender:</label> </td>

<td> <input type="radio" name="gender" id="male"> <label for="male">Male</label>

<input type="radio" name="gender" id="female"> <label for="female">Female</label>

<input type="radio" name="gender" id="other"> <label for="other">Other</label>

</td>

</tr>

<tr> <td> <label for="Batch">Batches:</label> </td>

<td>

<input type="checkbox" name="Batch" id="C4"> <label for="C4">C4</label>

<input type="checkbox" name="Batch" id="C5"> <label for="C5">C5</label>

<input type="checkbox" name="Batch" id="C6"> <label for="C6">C6</label>

</td>

<tr> <td> <label for="course">Course:</label> </td>

<td>

<select name="course" id="course">

<option value="CS">CS</option>

<option value="Electronics">Electronics</option>

<option value="Mechanical">Mechanical</option>

</select>

</td>

</tr>

<tr>

<td> <input type="reset" value="Reset All"> </td>

<td> <input type="submit" value="Register Now"> </td>

</tr>

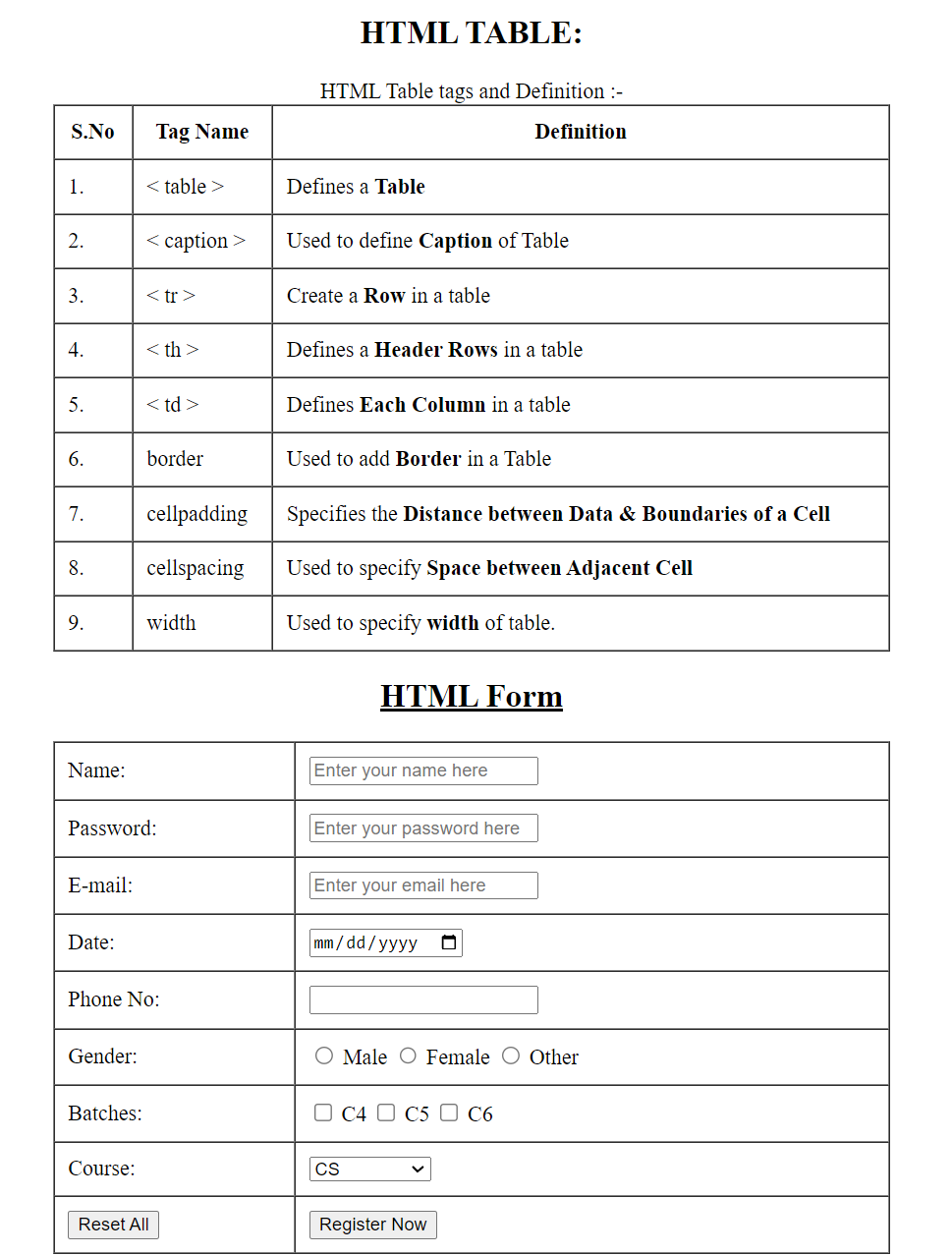
</table>

</form>

</CENTER>

</body>

</html>

**OUTPUT:**

**EXPERIMENT – 4**

**AIM:** Design Webpage to demonstrate use of Cascading Style Sheets

1. Inline CSS
2. Embedded CSS
3. External CSS

**THEORY:**

**INLINE CSS**

Inline CSS is a method of adding styles to an HTML element by using the style attribute within the HTML tag. This method allows you to define styles for a specific element directly within the HTML code, rather than in a separate CSS file or in the head section of the HTML document.

In inline CSS, you use the style attribute to define the styles for an HTML element. The style attribute contains one or more CSS property-value pairs that define the styles for the element. You can define multiple styles by separating the property-value pairs with a semicolon (;).

**EMBEDDED CSS**

Embedded CSS is a method of adding styles to an HTML document that involves placing CSS code within the head section of an HTML document. This method allows you to define styles that apply only to specific elements within the document, without affecting the styles of other elements.

In Embedded CSS, you use the "style" tag within the "head" tag of an HTML document to define styles for the document. You can define styles for different elements of the document by using CSS selectors, which target specific HTML elements, and then setting CSS properties to define the styles you want to apply to those elements.

**EXTERNAL CSS**

External CSS is a method of adding styles to an HTML document that involves placing CSS code in a separate file with a .css extension. This method allows you to define styles in a separate file, which can be reused across multiple HTML documents, making it easier to maintain and update the styles of a website.

In external CSS, you create a separate CSS file with a .css extension, and then link to that file from within the HTML document. You can define styles for different elements of the document by using CSS selectors, which target specific HTML elements, and then setting CSS properties to define the styles you want to apply to those elements.

**CODE:**

**INLINE CSS:**

<!DOCTYPE html>

<html>

<head>

<title>CSS</title>

<head>

<body>

<center>

<h2> Experiment 4(a)</h2>

<h1><span style="border: 2px solid black;

padding: 5px; font-family: 'Times New Roman’;

font-size: 40px;" >INLINE CSS </span></h1>

</center>

<p style = "border: 1px solid black ;

padding: 3px;

font-family: monospace;

font-size: 20px;" >

Inline CSS is a method of adding styles to an HTML element by using the style attribute

within the HTML tag. This method allows you to define styles for a specific element

directly within the HTML code, rather than in a separate CSS file or in the head section of

the HTML document.

<br><br>

In inline CSS, you use the style attribute to define the styles for an HTML element. The

style attribute contains one or more CSS property-value pairs that define the styles for the

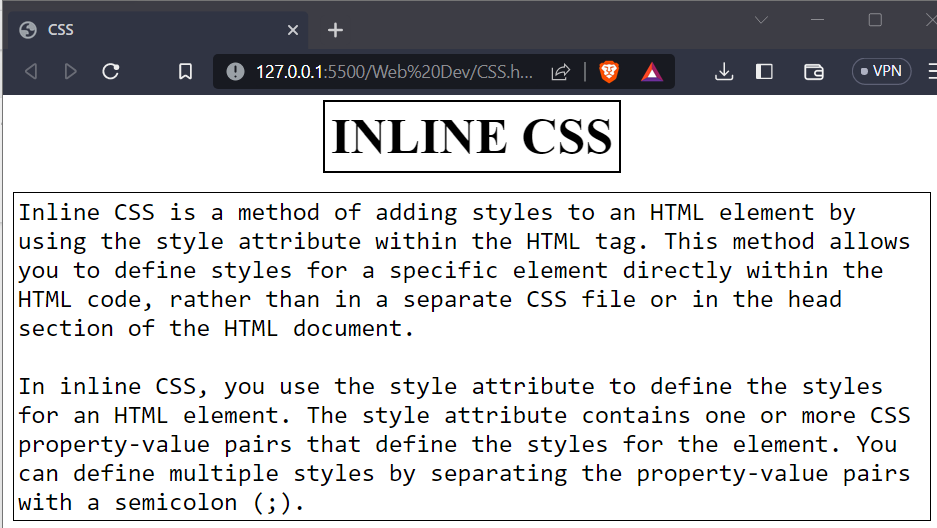
element. You can define multiple styles by separating the property-value pairs with a

semicolon (;).

</p>

</body>

</html>

**OUTPUT:**

**EMBEDDED CSS:**

<!DOCTYPE html>

<html>

<head>

<title>CSS</title>

<style>

span.style1

{

border: 2px solid black;

padding: 5px;

font-family: Times New Roman;

font-size: 40px;

}

p.style1

{

border: 1px solid black;

padding: 3px;

font-family: monospace;

font-size: 20px;

}

</style>

</head>

<body>

<center>

<h2> Experiment 4(b) <h2>

<h1><span class="style1">EMBEDDED CSS</span></h1>

</center>

<p class="style1">

Embedded CSS is a method of adding styles to an HTML document that involves placing CSS

code within the head section of an HTML document. This method allows you to define styles

that apply only to specific elements within the document, without affecting the styles of other

elements.

<br><br>

In Embedded CSS, you use the "style" tag within the "head" tag of an HTML document to

define styles for the document. You can define styles for different elements of the document

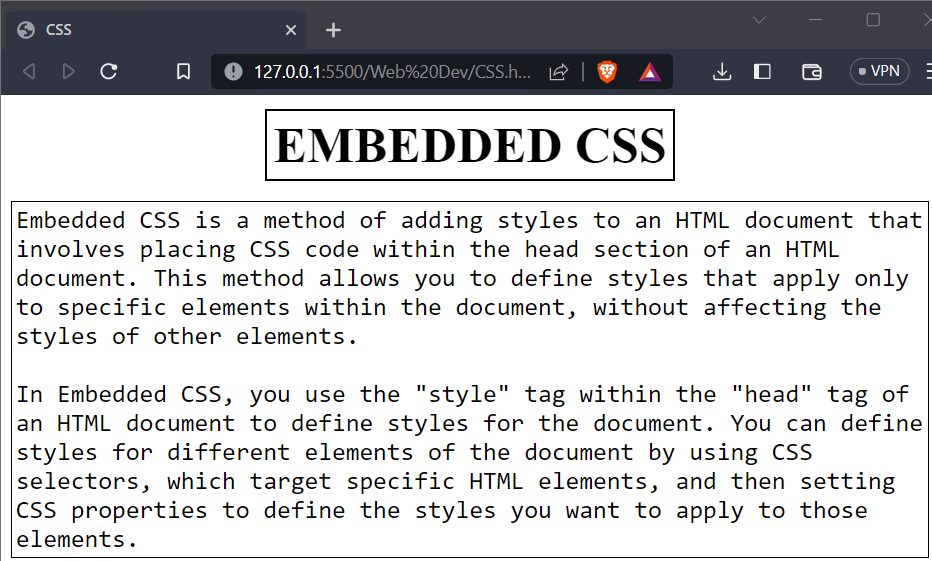
by using CSS selectors, which target specific HTML elements, and then setting CSS

properties to define the styles you want to apply to those elements.

</p>

</body>

</html>

**OUTPUT:**

**EXTERNAL CSS:**

**STYLE.CSS:**

span.special

{

border: 2px solid black;

padding: 5px;

font-family: 'Times New Roman';

font-size: 40px;

}

p.special1 {

border: 1px solid black;

padding: 3px;

font-family: monospace;

font-size: 20px;

}

**CSS.HTML**

<!DOCTYPE html>

<html>

<head>

<title>CSS</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<center>

<h2> Experiment 4(c) <h2></h2>

<h1> <span class="special"> EXTERNAL CSS </span></h1>

</center>

<p class="special1">

External CSS is a method of adding styles to an HTML document that involves placing CSS

code in a separate file with a .css extension. This method allows you to define styles in a

separate file, which can be reused across multiple HTML documents, making it easier to

maintain and update the styles of a website.

<br><br>

In external CSS, you create a separate CSS file with a .css extension, and then link to that file

from within the HTML document. You can define styles for different elements of the

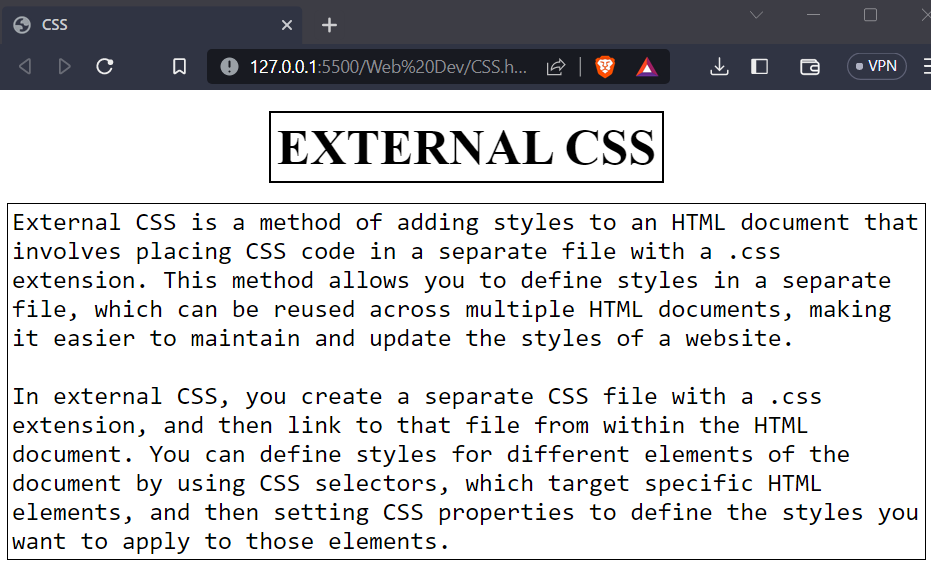
document by using CSS selectors, which target specific HTML elements, and then setting CSS

properties to define the styles you want to apply to those elements.

</p>

</body>

</html>

**OUTPUT:**

**EXPERIMENT – 5**

**AIM:** Design an XML Catalogue of your choice to understand the working structure of XML. You may choose a Food Menu, CD database or any other to understand the child-parent relationship in XML.

**THEORY:**

The Extensible Markup Language (XML) is a simple text-based format for representing structured information: documents, data, configuration, books, transactions, invoices, and much more. It was derived from an older standard format called SGML (ISO 8879), in order to be more suitable for Web use.

XML is one of the most widely-used formats for sharing structured information today: between programs, between people, between computers and people, both locally and across networks

1. **Redundancy**: - XML markup is very verbose. For example, every end tag must be supplied, such as in the example. This lets the computer catch common errors such as incorrect nesting.
2. **Self-describing**: - The readability of XML (it is a text-based format) and the presence of element and attribute names in XML means that people looking at an XML document can often get a head start on understanding the format (and it also helps people to find mistakes!)
3. **Network effect and the XML Promise**: - Any XML document can be read and processed by any XML tool whatsoever. Of course, some XML tools might want specific XML markup, but the XML format itself can be read by any XML parser: you can't say, this XML document is only to be processed by such-and-such a tool.

**Example of XML**:

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

<students>

<student>

<Name>Ram</Name>

<Roll\_No>100</Roll\_No>

<Age>20</Age>

</student>

<student>

< Name >Jane Smith</ Name >

< Roll\_No >102</ Roll\_No >

< Age >21</ Age >

</student>

</students>

**CODE:**

**SONGS.xml:**

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

<CD>

<Song>

<SongName>Teri Meri</SongName>

<Singer>Shreya Ghoshal</Singer>

<Releaseyear>2011</Releaseyear>

</Song>

<Song>

<SongName>Tum Hi Ho</SongName>

<Singer>Arijit Singh</Singer>

<Releaseyear>2013</Releaseyear>

</Song>

<Song>

<SongName>Pehli Dafa</SongName>

<Singer>Atif Aslam</Singer>

<Releaseyear>2017</Releaseyear>

</Song>

</CD>

**FOOD.xml**:

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

<?xml-stylesheet type = "text/css" href = "styles.css"?>

<IndianFood>

<heading>FAMOUS INDIAN FOODS</heading>

<FoodItem>

<Name>Biryani</Name>

<FamousIn>Hyderabad</FamousIn>

<Price>150 Rs</Price>

<Calories>400 Calories</Calories>

</FoodItem>

<FoodItem>

<Name>Dosa</Name>

<FamousIn>South India</FamousIn>

<Price>50 Rs</Price>

<Calories>200 Calories</Calories>

</FoodItem>

<FoodItem>

<Name>Samosa</Name>

<FamousIn>North India</FamousIn>

<Price>20 Rs</Price>

<Calories>300 Calories</Calories>

</FoodItem>

</IndianFood>

**STYLES.css:**

heading, FoodItem, Name, FamousIn, Price, Calories{

display: block;

}

FoodItem{

margin-top: 10px;

margin-left: 6px;

}

heading{

border: 2px solid black;

font-family: Times New Roman;

font-weight: bold;

font-size: 30px;

text-align: center;

}

Name{

font-family: Algerian;

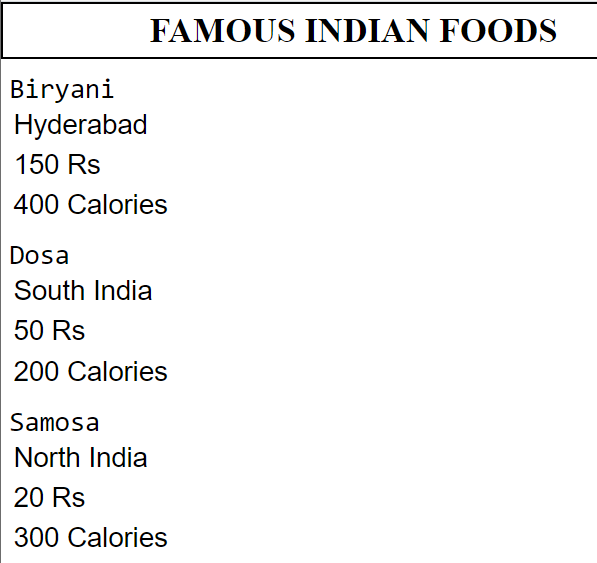
font-size: 20px;

}

FamousIn, Price, Calories{

font-family: perpetua;

font-size: 20px;

**OUTPUT :**

****

**EXPERIMENT – 6**

**AIM:** Design a Web-page that display the Current Date and Time and also make a Simple Calculator using JavaScript.

**THEORY:**

**DATE AND TIME:**

The JavaScript date object can be used to get year, month and day. We can display a timer on the webpage by the help of JavaScript date object. We can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

We can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

**CALCULATOR:**

**JavaScript eval() function:**

The eval() function in JavaScript is used to evaluate the expression. It is JavaScript’s global function, which evaluates the specified string as JavaScript code and executes it.

The parameter of the eval() function is a string. If the parameter represents the statements, eval() evaluates the statements. If the parameter is an expression, eval() evaluates the expression. If the parameter of eval() is not a string, the function returns the parameter unchanged.

There are some limitations of using the eval() function, such as the eval() function is not recommended to use because of the security reasons. It is not suggested to use because it is slower and makes code unreadable.

**Syntax:** eval(string)

**Values:** It accepts a single parameter, which is defined as follows.

**String:** It represents a JavaScript expression, single statement, or the sequence of statements. It can be a variable, statement, or a JavaScript expression.

**onclick() Event Listener:**

The **onclick()** event listener in JavaScript is used to listen for the click event on an HTML element, such as a button or a link. When the element is clicked, the function attached to the **onclick()** event is executed.

To use the **onclick()** event listener in JavaScript, you can either use an inline event handler in the HTML code or add an event listener using JavaScript code.

**SOURCE CODE:**<!DOCTYPE html>

<html>

<head>

<title>EXP 6</title>

<script lang="JAVASCRIPT">

const date = new Date();

</script>

</head>

<body>

<hr>

<h2 style="text-align: center;">Date And Time</h2>

<script>

document.write(date);

</script>

<hr>

<h1>Calculator</h1>

<form>

<input type="text" name="result" id="result" disabled>

<br>

<input type="button" value="1" onclick="addToResult('1')">

<input type="button" value="2" onclick="addToResult('2')">

<input type="button" value="3" onclick="addToResult('3')">

<input type="button" value="+" onclick="addToResult('+')">

<br>

<input type="button" value="4" onclick="addToResult('4')">

<input type="button" value="5" onclick="addToResult('5')">

<input type="button" value="6" onclick="addToResult('6')">

<input type="button" value="-" onclick="addToResult('-')">

<br>

<input type="button" value="7" onclick="addToResult('7')">

<input type="button" value="8" onclick="addToResult('8')">

<input type="button" value="9" onclick="addToResult('9')">

<input type="button" value="\*" onclick="addToResult('\*')">

<br>

<input type="button" value="0" onclick="addToResult('0')">

<input type="button" value="." onclick="addToResult('.')">

<input type="button" value="/" onclick="addToResult('/')">

<input type="button" value="AC" onclick="clearResult()">

<br>

<input type="button" value="=" onclick="calculateResult()">

</form>

<script>

function addToResult(value) {

document.getElementById('result').value += value;

}

function clearResult() {

document.getElementById('result').value = '';

}

function calculateResult() {

var result = eval(document.getElementById('result').value);

document.getElementById('result').value = result;

}

</script>

</body>

</html>

**OUTPUT:**

****

**Experiment – 7**

**AIM:**

Design a simple form that includes email, password, phone no. as a field and use JavaScript to validate the email address (For proper structure), phone no. (To follow 10-digit norms) and password (To include at-least one alphanumeric and one number).

**THEORY:**

Form validation in HTML and JavaScript is a crucial aspect of web development that ensures the data submitted by users through a form meets certain criteria and is valid. It helps maintain data integrity and improves user experience by providing meaningful feedback on incorrect or incomplete form submissions.

The form validation is implemented using JavaScript within the `<script>` tags. Give down the key components and their functionalities:

**HTML Form Structure:**

* The HTML `<form>` element encapsulates the form elements and handles the submission.
* Input fields are defined using `<input>` tags, with different types such as `email`, `number`, and `password`.
* Each input field has an associated `id` attribute, allowing JavaScript to access their values.

**JavaScript Functions:**

1. **validateEmail(email):** Validates the email input using regular expressions. It checks if the email is empty, and if not, it matches the email pattern.
2. **validatePhoneNumber(phone):** Validates the phone number input using regular expressions. It checks if the phone number is empty, and if not, it matches the pattern for a 10-digit phone number.
3. **validatePassword(pass):** Validates the password input against specific criteria. It checks if the password is empty and then performs additional checks, such as length, presence of uppercase and lowercase letters, and at least one number. It concatenates any error messages and returns them.
4. **handleSubmit(event):** This function is called when the form is submitted. It prevents the default form submission behavior (reloading the page).

* Inside `handleSubmit`, the values of the email, phone number, and password fields are retrieved using their respective `id` attributes.
* Each input value is then passed to the corresponding validation function.
* If any validation error is encountered, an alert with the error message is displayed.
* If all validations pass, an alert is displayed indicating successful form submission.

This form performs validation on each field individually, displaying relevant error messages. The form is only submitted when all validations pass. This approach enhances user experience by providing immediate feedback and reduces the chances of submitting incorrect or incomplete data.

**SOURCE CODE:**

<!DOCTYPE html>

<html>

<head>

<style>

\* {

box-sizing: border-box; padding: 0; margin: 0;

font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

}

body {

background-color: #38404E; width: 100%; height: 100vh; display: flex;

justify-content: center; align-items: center;

}

.form-div {

background-color: white; padding: 30px 25px; min-width: 100px;

max-width: 270px; width: 100%; border-radius: 10px;

}

.heading { text-decoration: underline; text-align: center; margin-bottom: 15px; }

.input-field { padding: 8px; width: 100%; margin-top: 3px; }

label { font-size: 16px; }

.button {

padding: 10px; width: 100%; background-color: #38404E; color: white;

}

.button:hover { cursor: pointer;}

</style>

<title>Registration Form</title>

</head>

<body>

<div class="form-div">

<h2 class="heading">Register</h2>

<form onsubmit="handleSubmit(event)">

<label for="email">Email</label> <br>

<input class="input-field" type="email" name="email" id="email" placeholder="Enter your email">

<br><br>

<label for="phone">Phone No.</label> <br>

<input class="input-field" type="number" name="phone" id="phone" placeholder="Enter your phone no.">

<br><br>

<label for="pass">Password</label> <br>

<input class="input-field" type="password" name="pass" id="pass" placeholder="Enter your password">

<br><br>

<input class="button" type="submit" value="Submit">

</form>

</div>

<script>

function validateEmail(email) {

if (email.trim() === "") return "📧 Please Enter Email!";

var pattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (pattern.test(email)) return "";

else return "📧 Please Enter Valid Email!";

function validatePhoneNumber(phoneNumber) {

if (phoneNumber.trim() === "") return "📱 Please Enter Phone Number!";

var pattern = /^\d{10}$/;

if (pattern.test(phoneNumber)) return "";

else return "📱 Please Enter valid Phone Number (10 Digits)";

}

function validatePassword(pass) {

let allErrors = "";

if (pass.trim() === "") return "🔴 Please Enter Password \n";

if (pass.length <= 8) allErrors += "Password must be of 8 length \n";

if (!(/[A-Z]/.test(pass))) allErrors += "Atleast one UpperCase Letter \n";

if (!(/[a-z]/.test(pass))) allErrors += "Atleast one LowerCase Letter \n";

if (!(/\d/.test(pass))) allErrors += "Atleast one Number \n";

return allErrors;

}

function handleSubmit(event) {

event.preventDefault();

const email = document.getElementById('email').value;

const phone = document.getElementById('phone').value;

const pass = document.getElementById('pass').value;

const emailError = validateEmail(email);

const phoneError = validatePhoneNumber(phone);

const passwordError = validatePassword(pass);

if (emailError) alert(emailError);

else if (phoneError) alert(phoneError);

else if (passwordError) alert(passwordError);

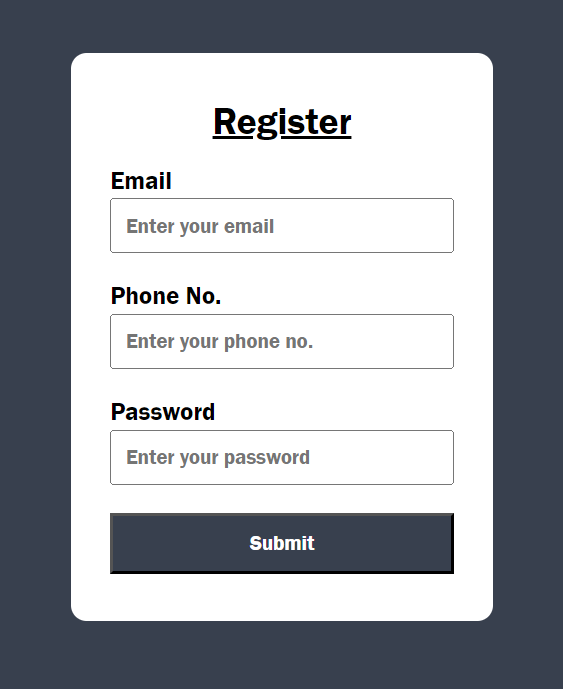
else alert("🥳Form Submitted Successfully!");

}

</script>

</body>

</html>

**OUTPUT:**

**Experiment – 8**

**AIM:** Deploy a Content Management System (CMS) and prepare a stepwise instruction on how to configure the CMS on Apache/XAMPP/WAMPP.

**THEORY:**

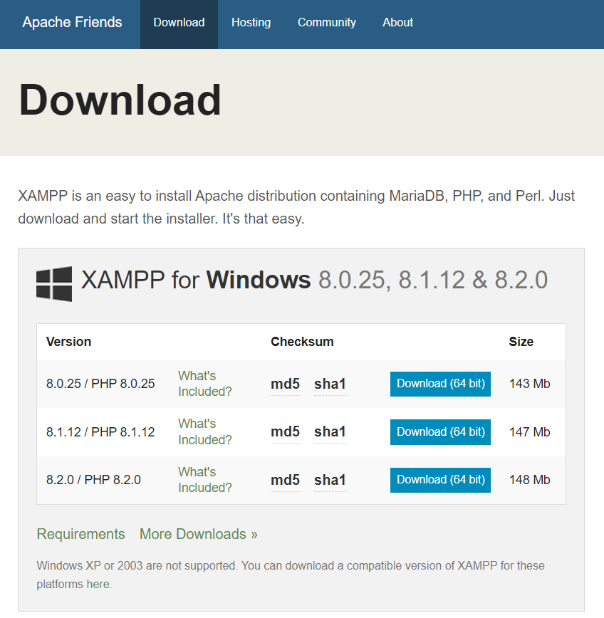
**Terminology:**

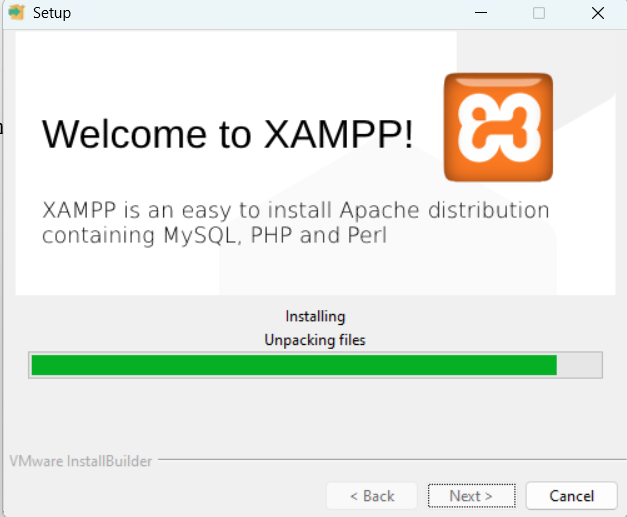
* **Content Management System (CMS)** − A CMS has a central interface that is used to publish, edit, modify, and maintain content. Some well know CMS include, WordPress, Drupal, and Joomla.
* **WordPress** − It is an open-source content management system founded by Matt Mullenweg and Mike Little. It was initially released in the year 2003 and its current version is WordPress 4.4.
* **XAMPP** − It was created by Apache Friends and is an open-source cross-platform web server solution stack. XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P).
* **PhpMyAdmin**− It is an open-source tool to handle MySQL administration on a web browser. PhpMyAdmin perform useful tasks like creating a new database, modifying or deleting it, adding tables, executing SQL statements, managing permissions, etc.

**Steps to Install WordPress (CMS) on localhost:**

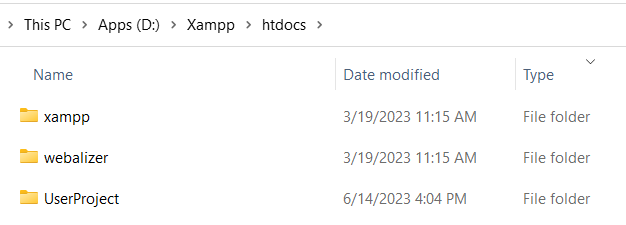
**Step 1:** **Download and Install XAMPP server.**

Go to XAMPP official website (Apache Friends) and download XAMPP.

(Link: [**https://www.apachefriends.org/download.html**](https://www.apachefriends.org/download.html))

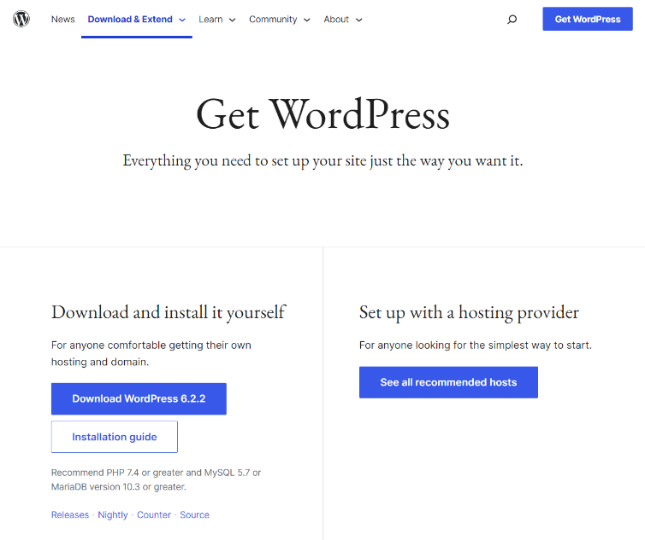


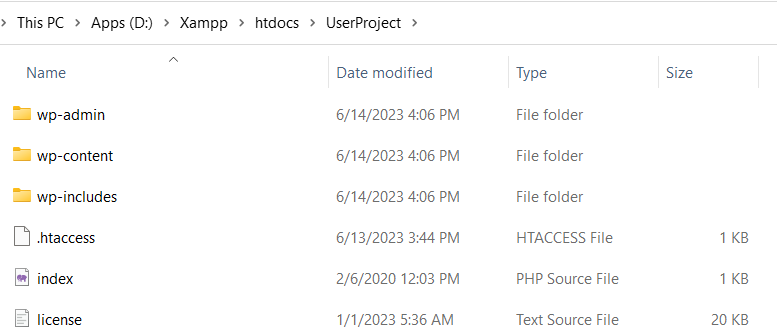
**Step 2:** Then Open folder PATH **D:\xampp\htdocs** then create your project directory.



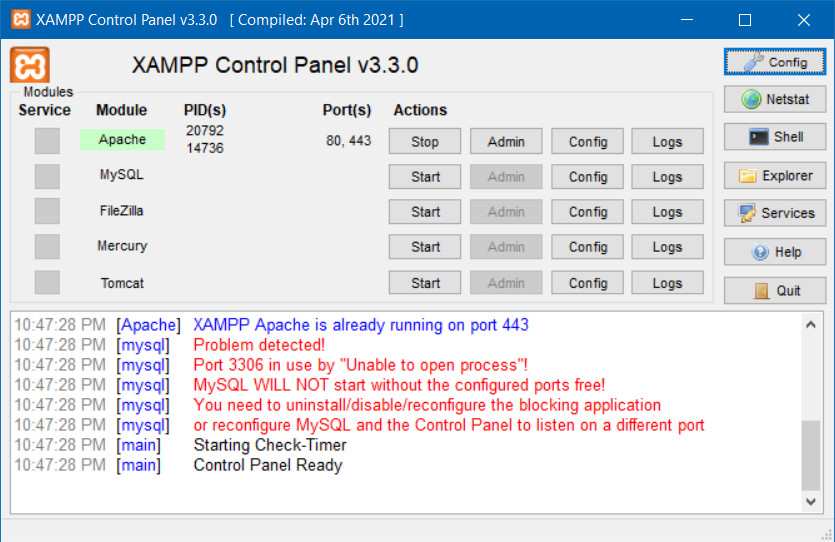
**Step 3:** Download WordPress from official website.

1. Download WordPress 6.6.2 zip file from official website.
2. Then unzip WordPress file in the following PATH: **D:\xampp\htdocs\UserProject**

x

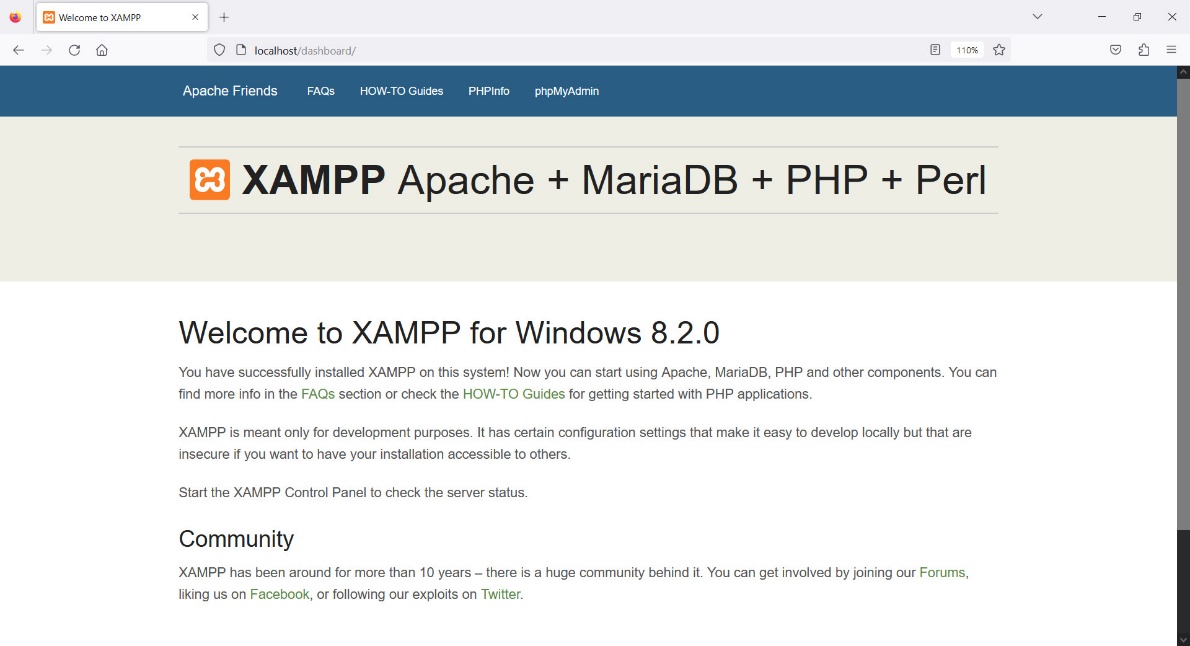


**Step 4:** Open XAMPP Control Panel.



**Step 5:** Now click on Start button in Actions Column to start Apache. Then click on Admin button to start web page with url: **localhost/dashboard**

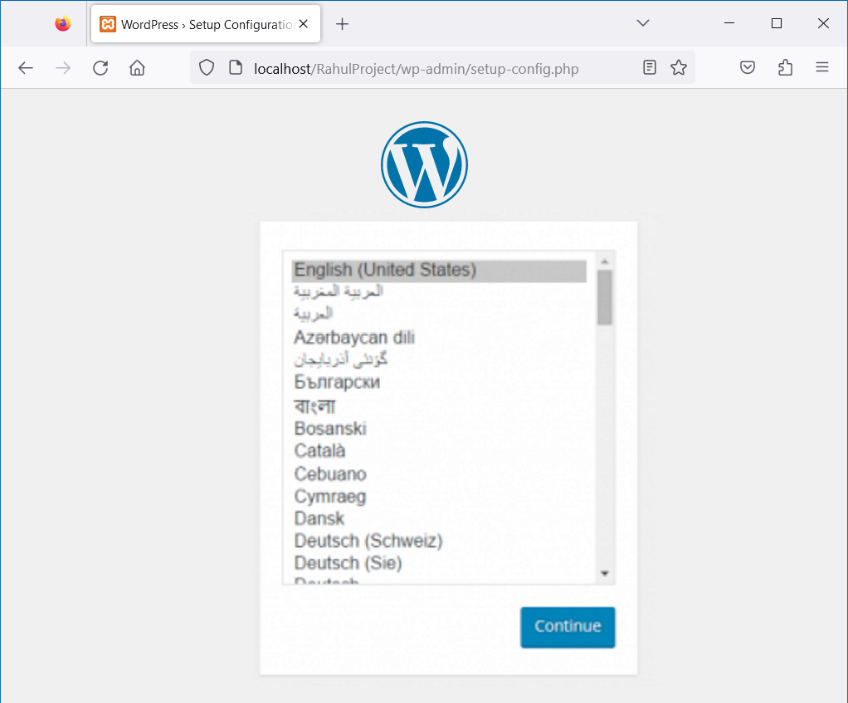
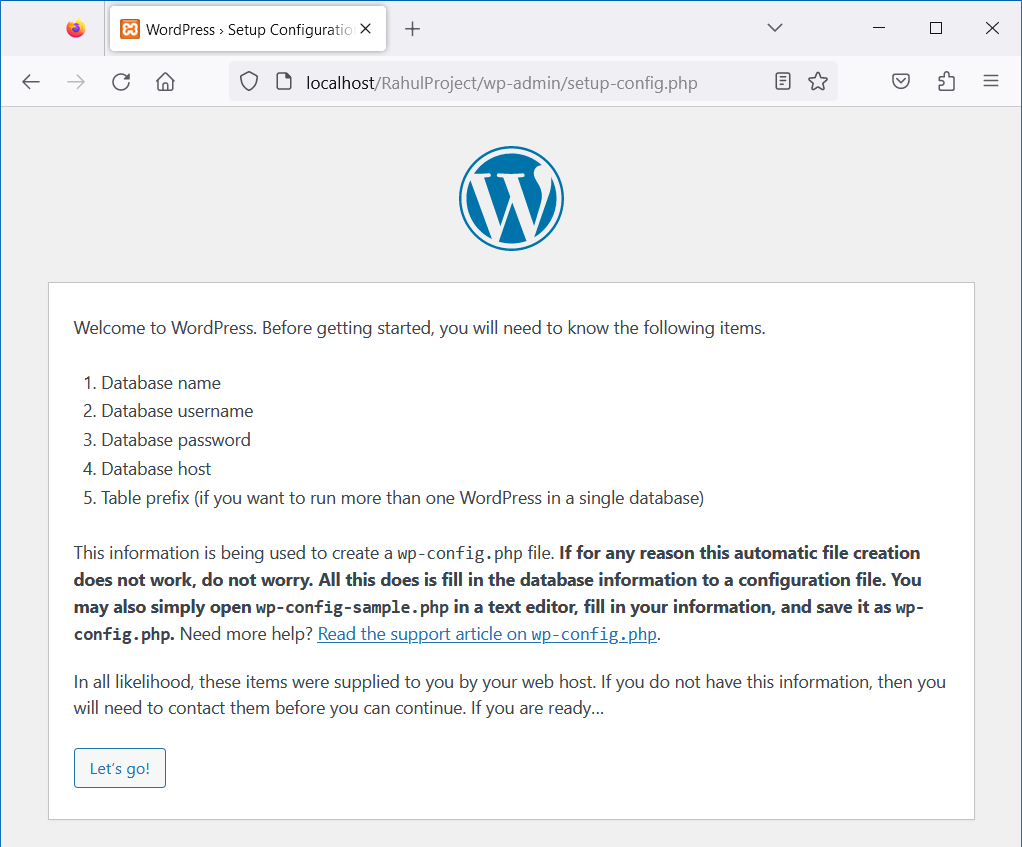
Open the web browser, type **localhost** and press enter. Now we can see the XAMPP dashboard. If the dashboard is visible, it means everything is working fine.

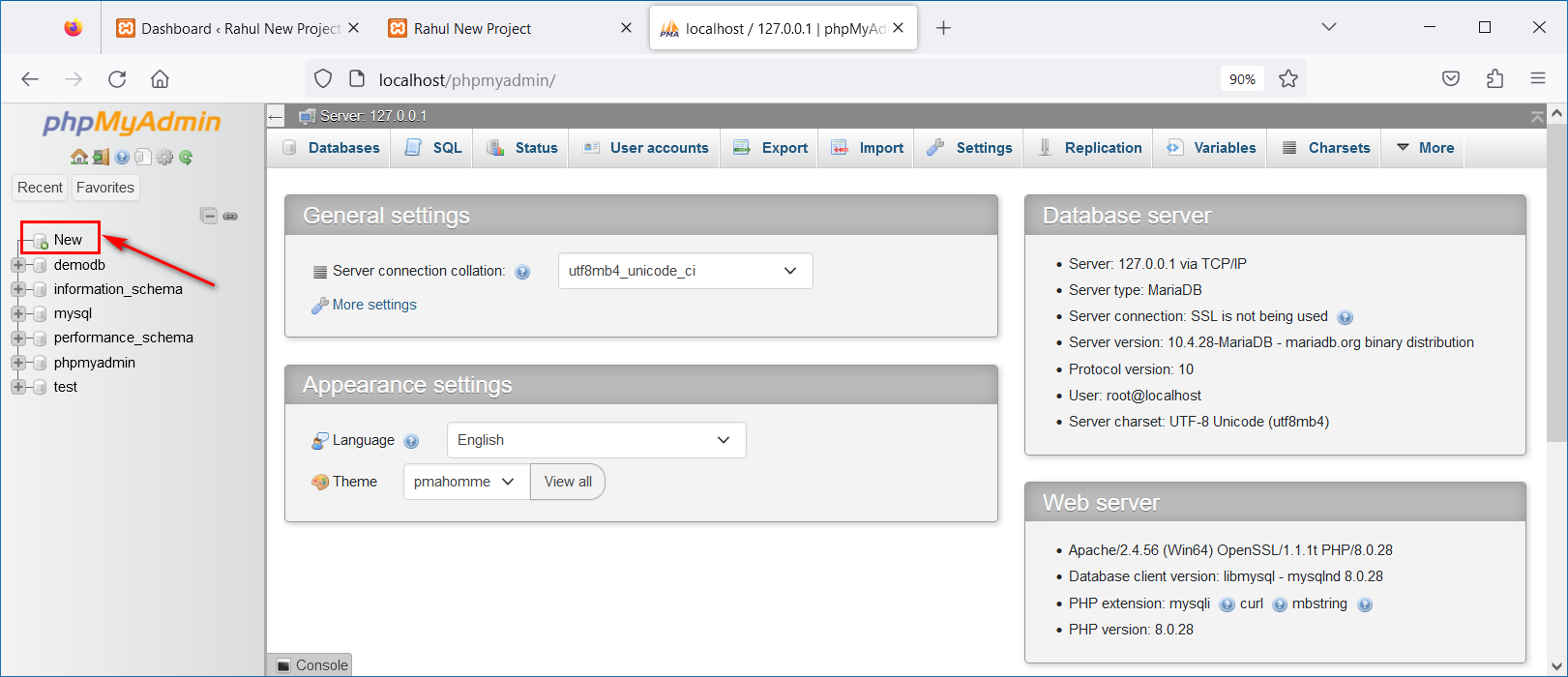


**Step 6:** Now we will enter the project folder from the browser. For that, type: **localhost/UserProject/** on the address bar and press enter.

After pressing enter we will be redirected to WordPress installation as in the following screenshot.

* Choose Language and Continue.
* Click on Let’s Go Option

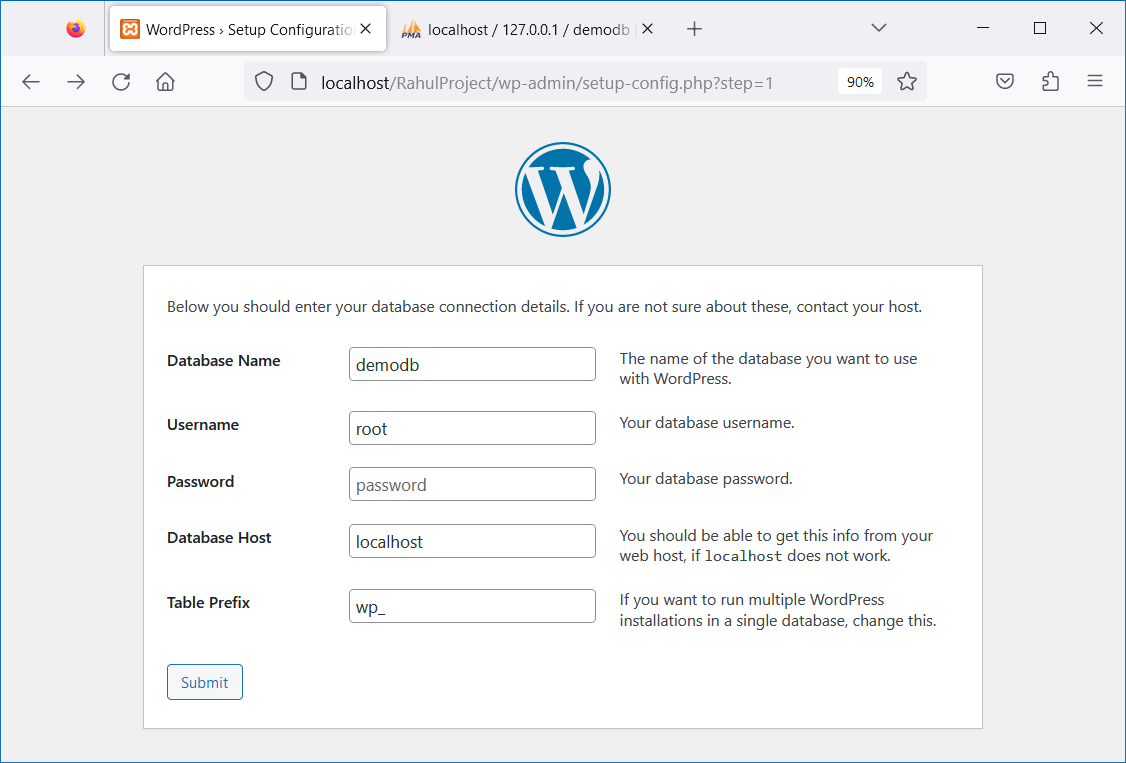


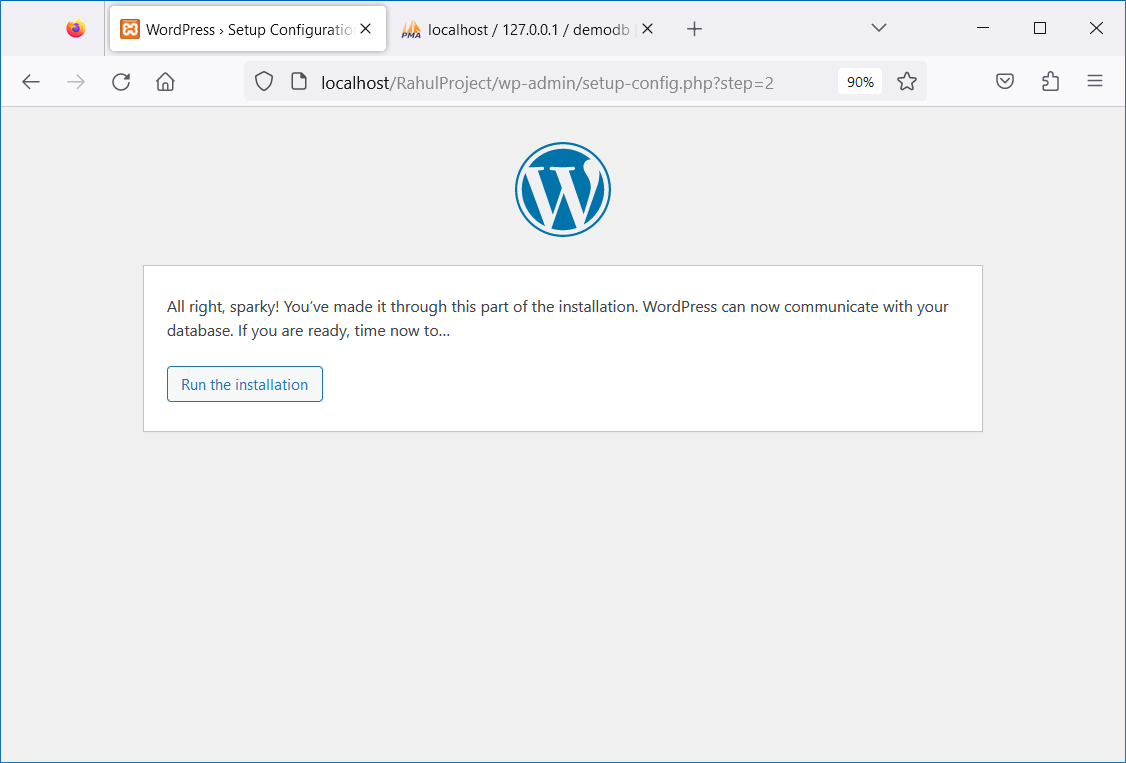
**Step 7:** Open another browser tab and type **http://localhost/phpmyadmin** and press enter. Now we will reach the PhpMyAdmin page. Create database for WordPress by clicking on New Icon. Enter database name and click create. (Eg: **demodb**)

**Step 8:** Continue to WordPress -> Setup Configuration

* Enter Database Name – That we have just created in phpMyAdmin
* Enter Username
* Enter Password
* Enter Database Host
* Enter Table Prefix

Then click on ***Submit*** Button. After that click on ***Run the Installation.***

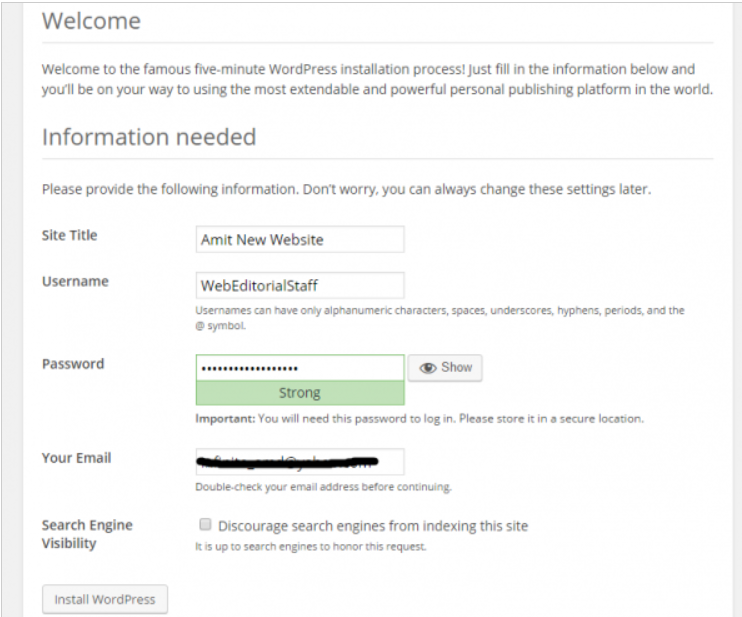




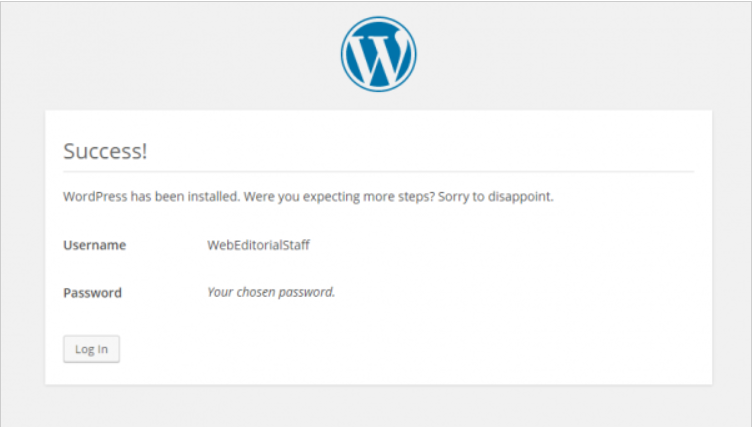
**Step 9:** Now, get ready to enter the site details. We have added the following, the password is already provided, you can also change it. The Username and Password will help you to login to your own website.

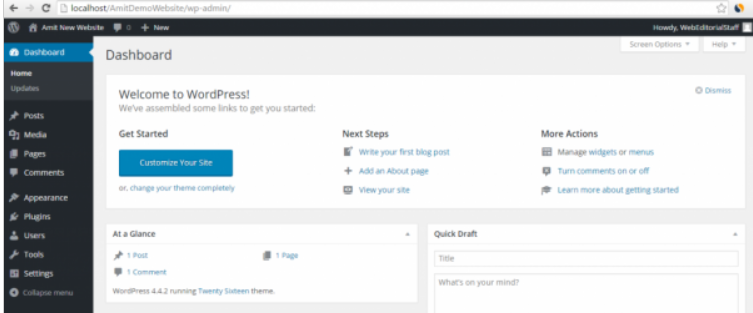
* Site Title – Your Website Title
* Username – Your User Name
* Password − Your Password
* You Email− Your Email ID
* Search Engine Visibility: Keep it as it is

Then, click “**Install WordPress**”

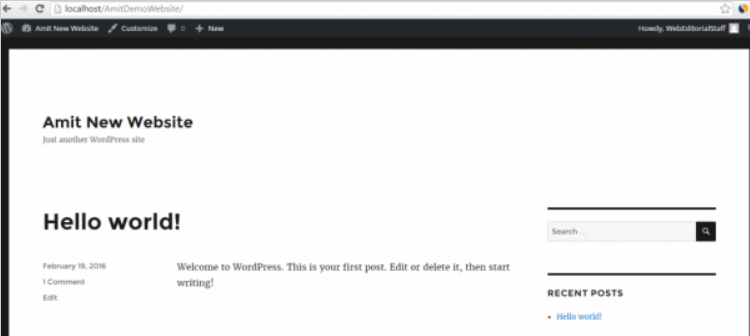


**Step 10:** Success! Congratulations, successful installation of WordPress. Now, Log In to your WordPress Website



**Step 11:** Great! We just entered the website admin and WordPress Dashboard.

**Step 12:** For viewing the homepage of your demo website, just type (Link **– http://localhost/UserProject**) on the address bar and press enter



**Conclusion:** WordPress on localhost using XAMPP started successfully!

**Experiment – 9**

**AIM:** Choose a topic to design and decide up the content. Prepare the content in a word file appear on the website. Also design a Visual-Site Map for the Website.

**THEORY:**

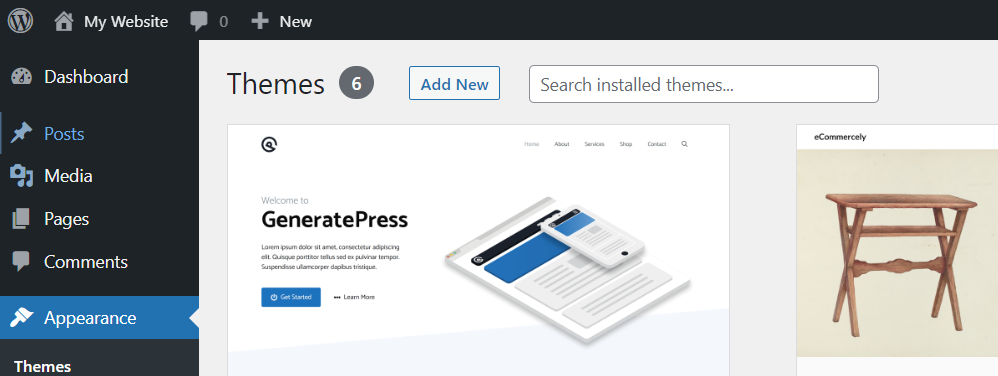
**Visual Site Map:**

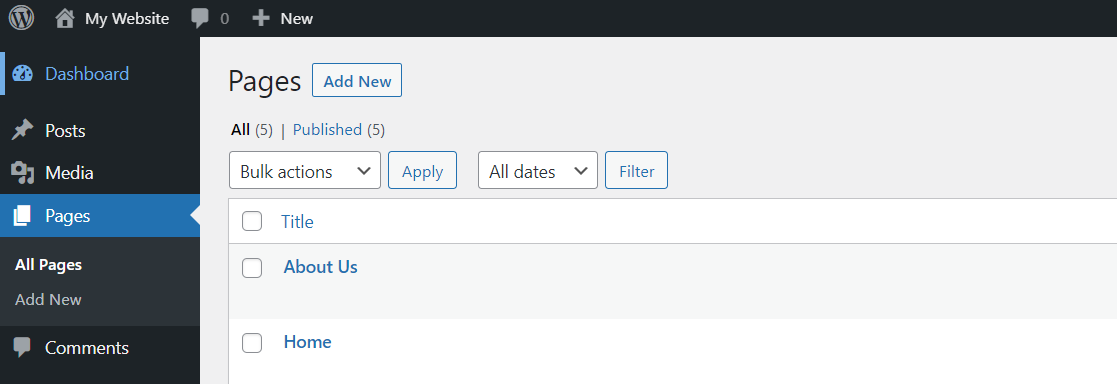
* A visual site map, also known as a website diagram or website structure diagram, is agraphical representation or visual depiction of the pages and hierarchical structure of a website. It illustrates the organization and relationships between different web pages, sections, and content on a website.
* The visual site map typically includes a hierarchical structure, showing main pages, subpages, and their connections.
* It helps web designers, developers, and stakeholders to visualize and plan the overall structure and navigation of a website, ensuring intuitive user experience and effective information architecture.

**Content of Website:**

1. Home Page
2. Site Map
3. About Us Page
4. Our Team Page
5. Privacy Policy Page

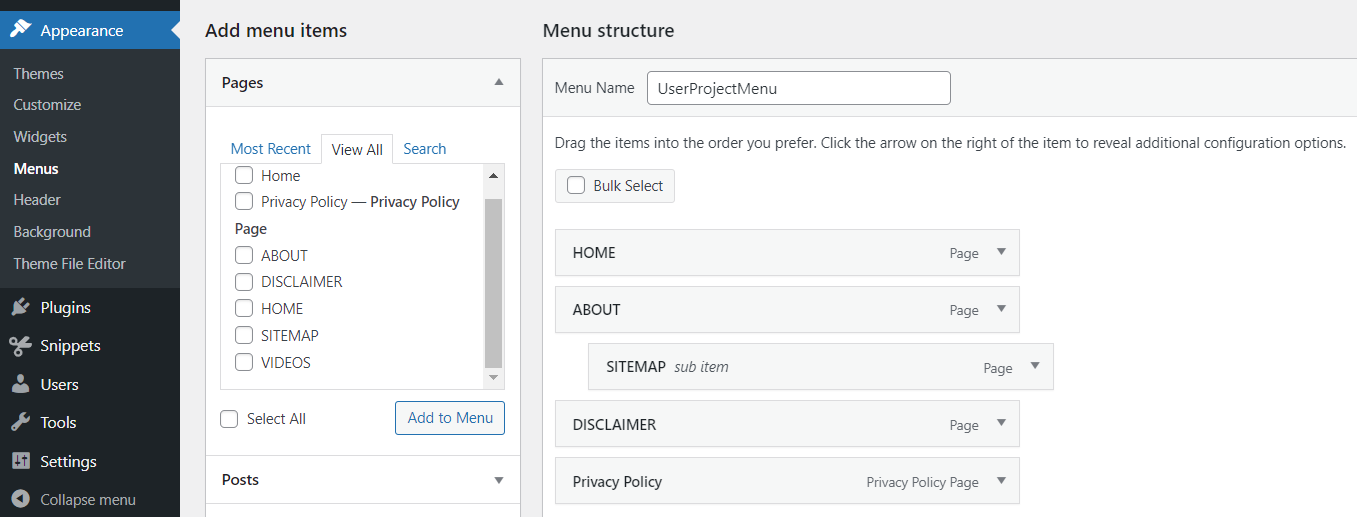
**Steps to Create Visual Site Map of Website:**

**Step 1:**Go to **localhost/UserProject/wp-admin/** and then go to Theme inside Appearance and Select a theme for Website And **Activate** it after Installation.

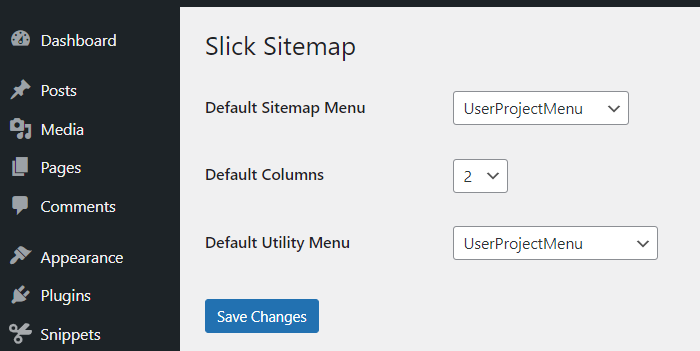
**Step 2:** Now Click on **Pages - > Add New** then Create As many pages a wanted in the Website.

**Step 3:** Now Go to Plugins - > Add New and Install Slick Sitemap and then activate it.

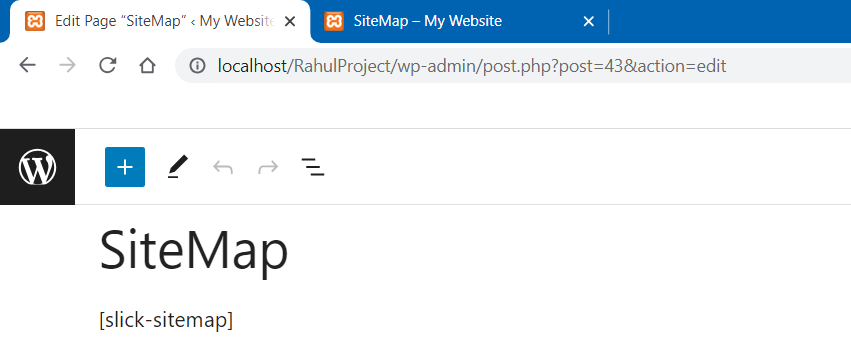


**Step 4:** Go to Appearance - > Menus and Set the Pages And set your menu as primary menu you so it can arrange as you want them in your Website.

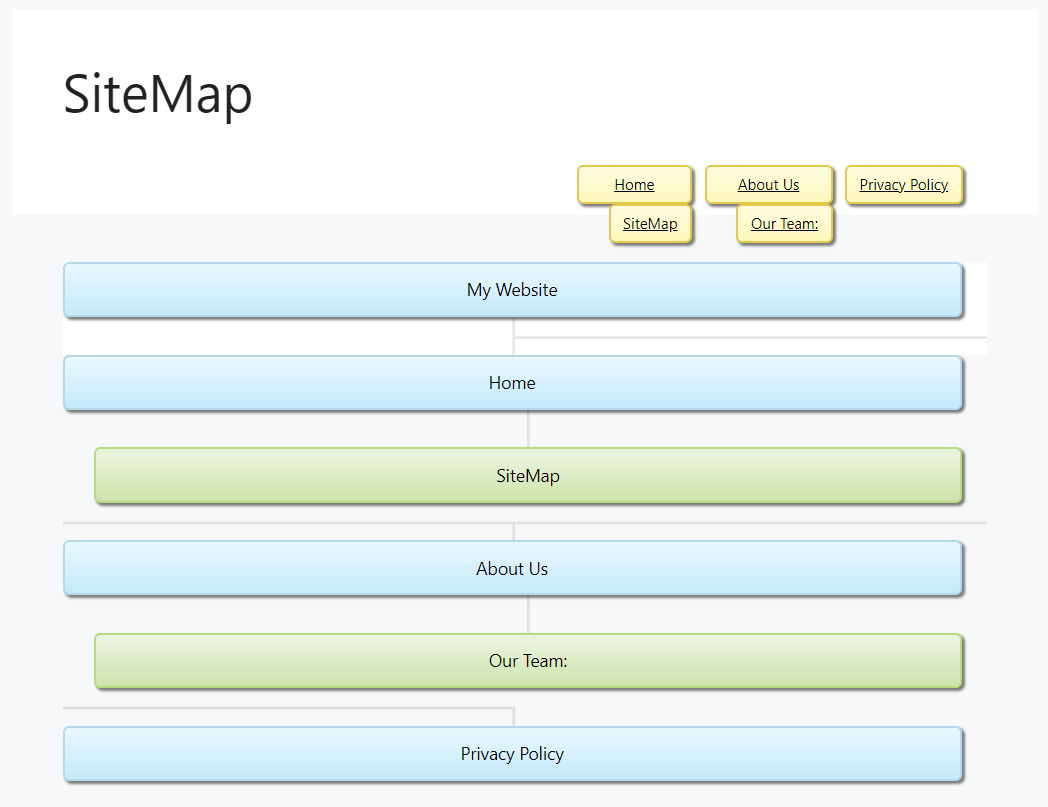
**Step 5:** Go to **Settings - > Slick-Sitemap** and Choose the Default Sitemap and utility menu by selecting the menu you created in previous step and also set column to two.

****

**Step 6:** Now go to **Pages - > Add New** Give Title Sitemap and write **[slick-sitemap]** inside it.



**OUTPUT:**



**Experiment – 10**

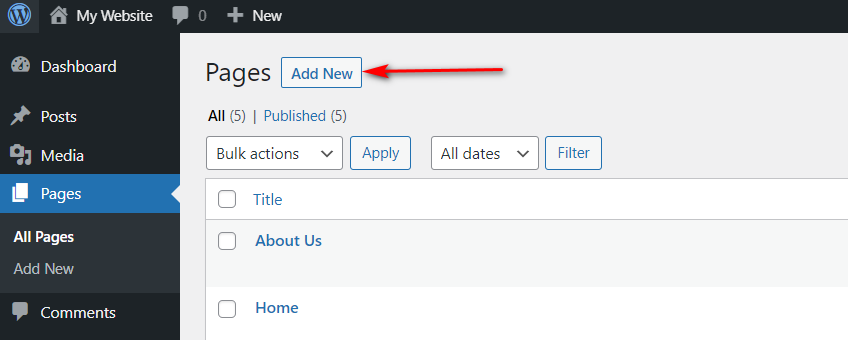
**AIM:** Configure the CMS and add the content according to the design and visual site-map of the Website.

**THEORY:**

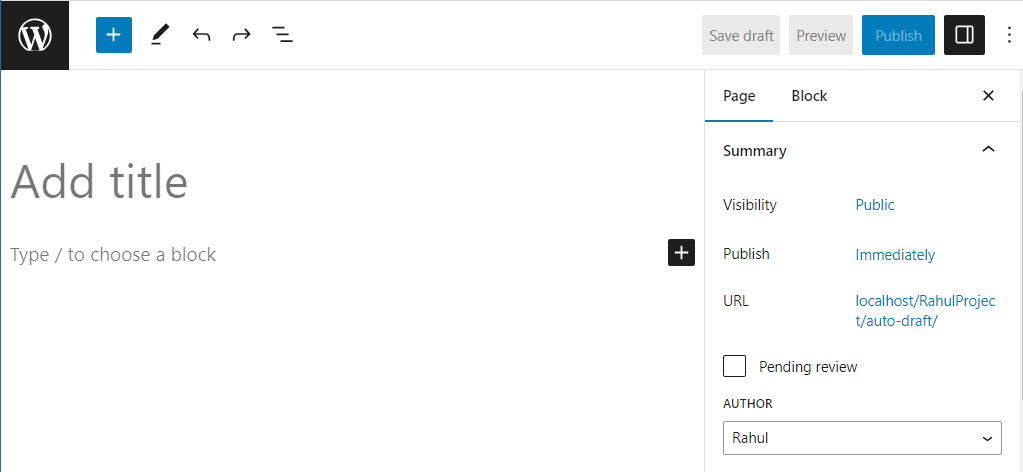
**Content of My Website:**

* **Home Page**:
* **Visual Site Map Page**:
* **About Us Page**:
* **Out Team**:
* **Privacy & Policy Page**

**How to Add New Content in Word Press:**

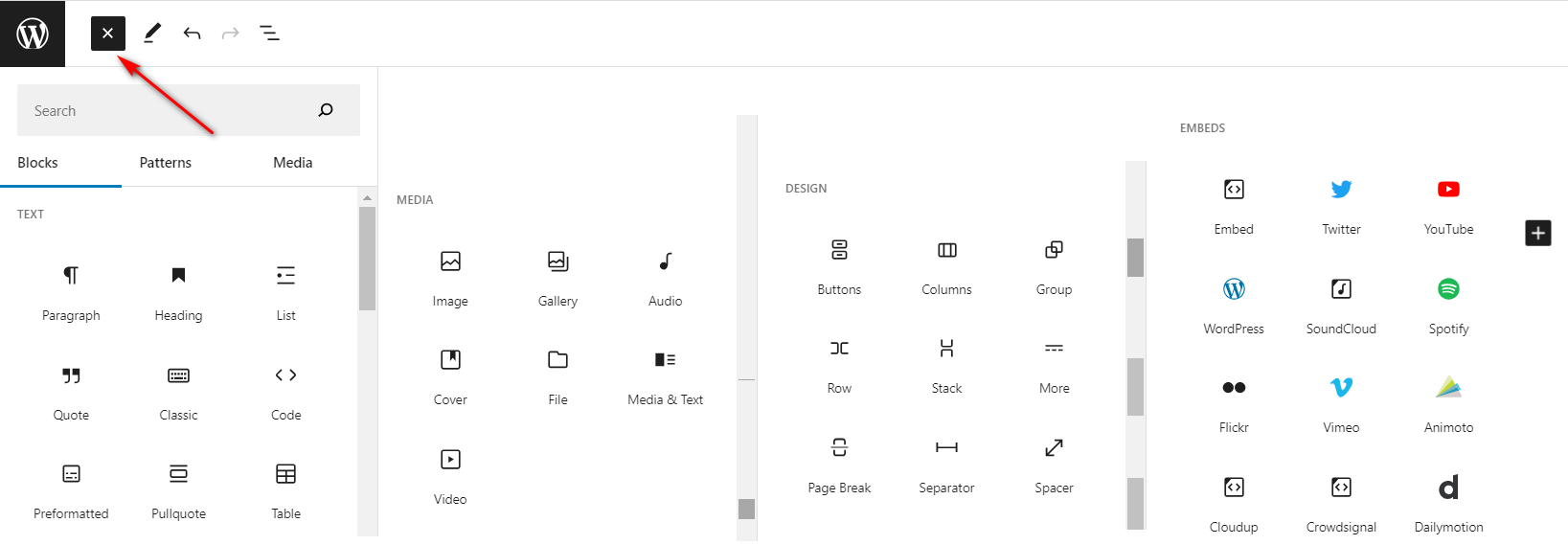
**Step 1:**To get started adding a new page to your WordPress site, find the Pages menu in the WordPress Dashboard Navigation menu. Click Add New.

**Step2**: Add the title of the page, like *About*. Click the Add Title text to open the text box where you will add your title.

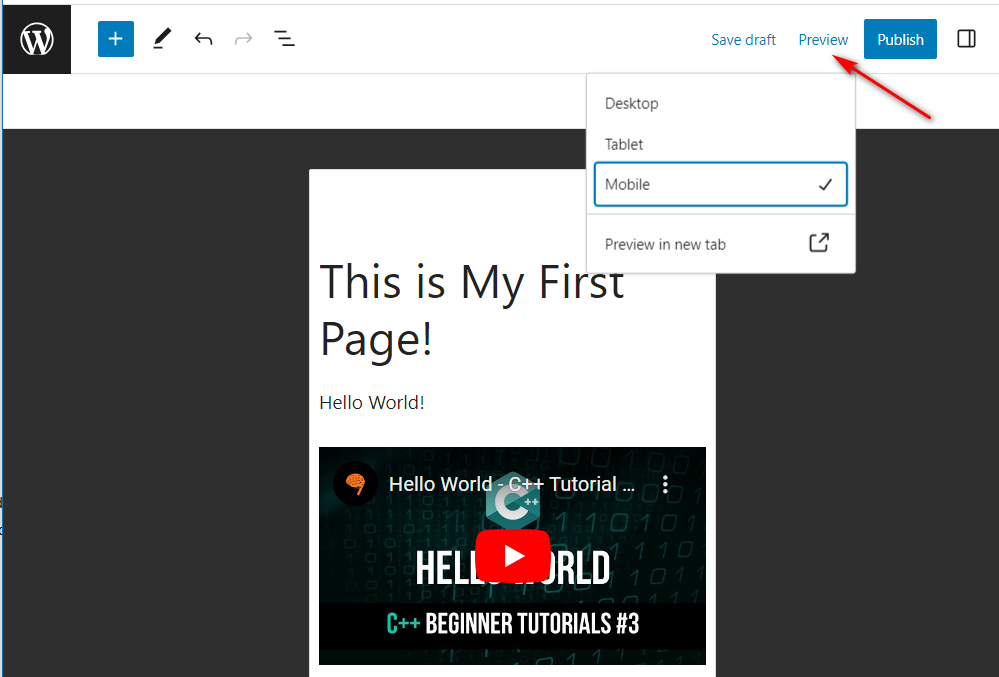


**Step 3**: Add some content. Content can be anything you choose … from text, headings, images, lists, videos, tables, and lots more.

To see the available blocks for your page, click the plus sign button at the top of the page.

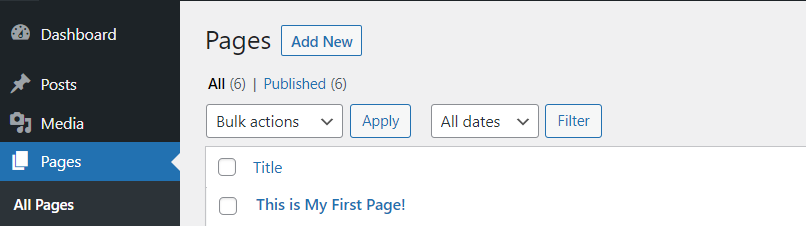


**Step 4:** Click on Preview Button to check how your website looks on different devices like: Desktop, tablet and mobile.



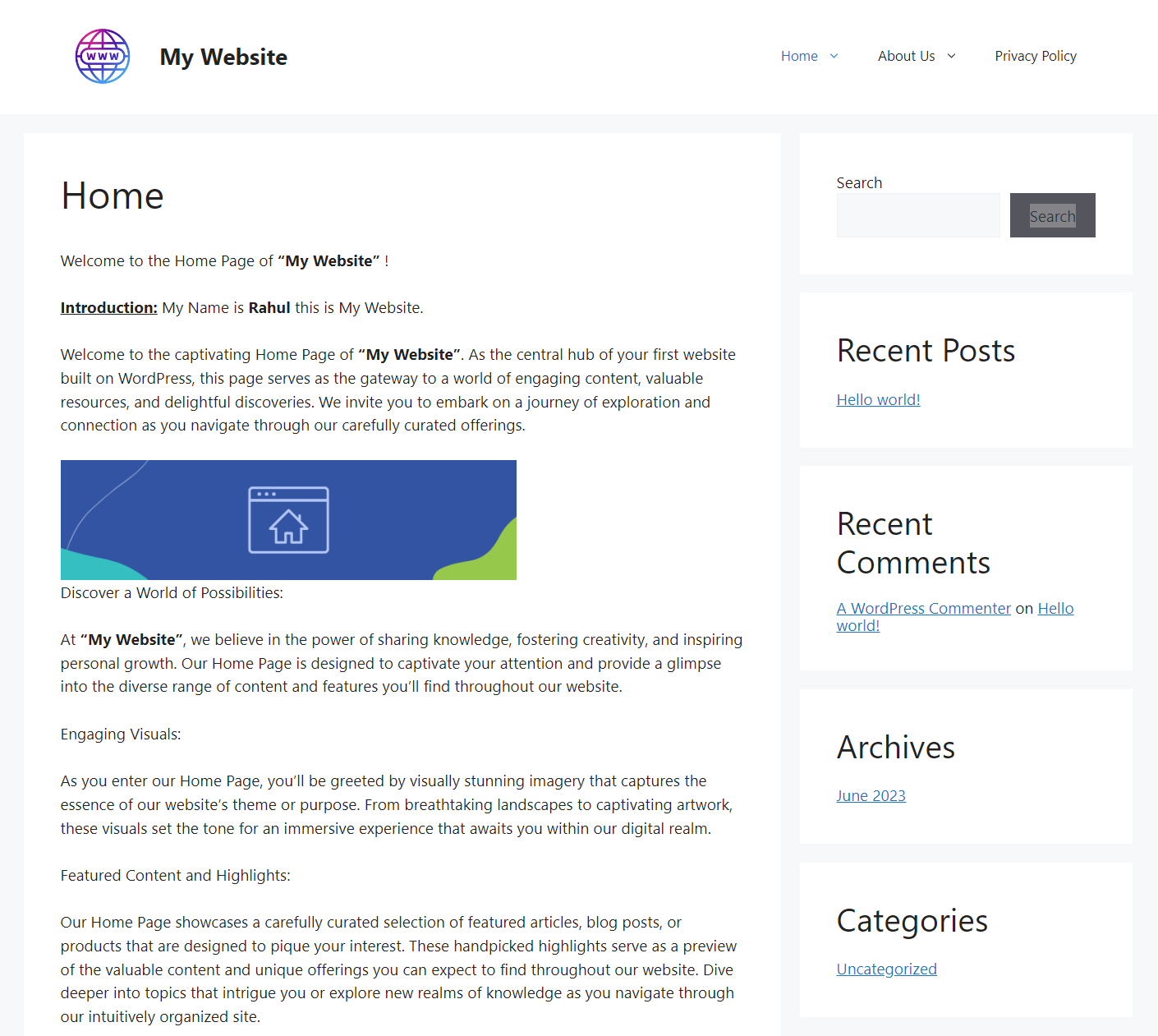
**Step 5:** Click on Publish Button to save and publish your page!



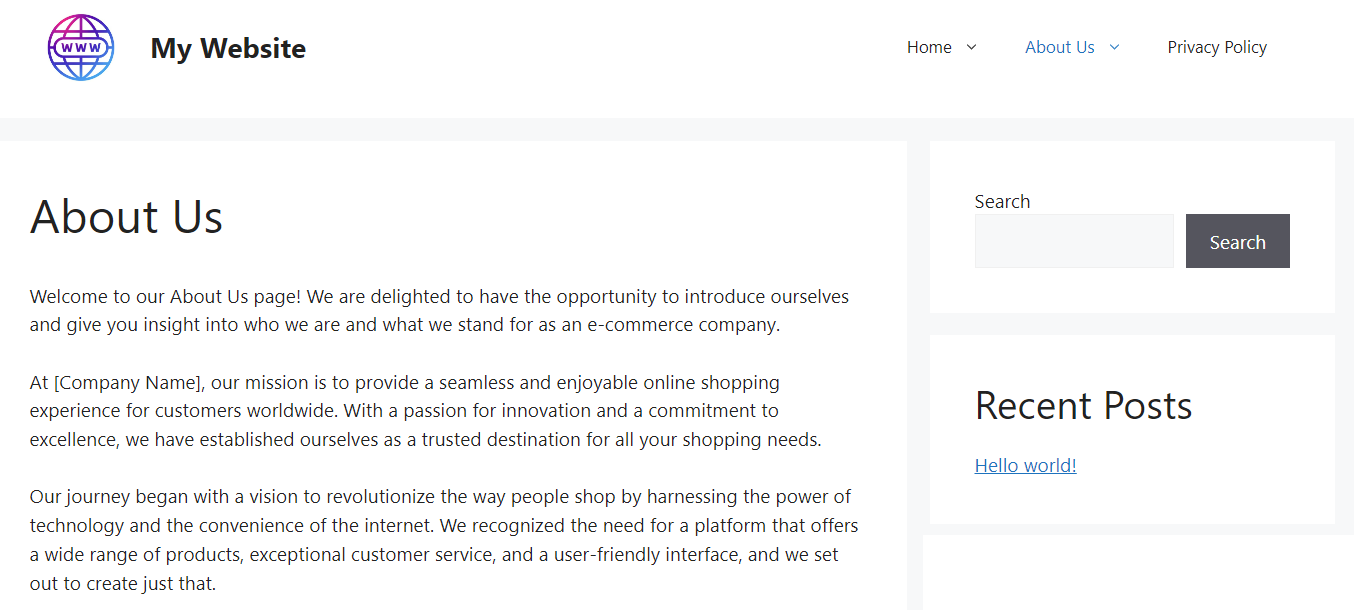
**Step 6:** After clicking Publish we can see the page in “Pages” menu of Wordpress!

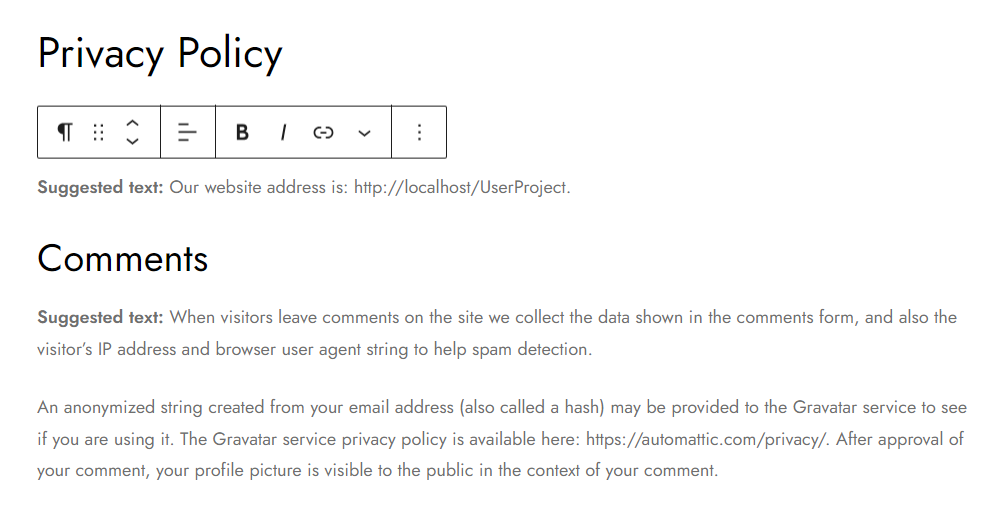
**OUPUT:**

***HOME PAGE****:*



***ABOUT US PAGE****:*



***PRIVACY & POLICY PAGE***