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:

What is a Web – Server?

A web server is a software application that serves web pages to client web browsers over the Internet or a local network. When you visit a website, your browser sends a request to the server, and the server responds by sending the necessary files to display the webpage in your browser.

A web server can run on various operating systems, including Windows, Linux, and macOS. It can also support various web technologies such as HTML, CSS, JavaScript, and server-side scripting languages such as PHP, Python, and Ruby.

Web servers can be used for different purposes, such as hosting websites, running web applications, serving files, and managing databases. Some common web servers include Apache, Nginx, IIS, and Lighttpd.

List some popular Web - Servers.

List of some popular web-servers are given below:

Web Server	**Company**	**Licence**
Apache	Apache Software Foundation	Open-Source (Apache License)
XAMPP	Apache Friends	Open-Source (GNU General Public License)
WAMP	Bitnami	Open-Source (Apache License)
Nginx	Nginx, Inc.	Open-Source (2-clause BSD-like license)
IIS (Internet Information Services)	Microsoft Corporation	Proprietary (Commercial)
GSS (Google Search Server)	Google	Proprietary (Commercial and Freeware)
Node.js	Node.js Foundation	Open-Source (MIT License)
Tomcat	Apache Software Foundation	Open-Source (Apache License)

What is a difference between Server and Web-server?

Sever: It is a general term used to refer to any computer or software application that provides a service or resource to other computers or devices on a network. Servers can provide a variety of services, such as file sharing, email, database management, or web hosting.

Web-server: It is a specific type of server that specializes in serving web pages and content to web browsers over the internet or a local network. It typically runs software that can interpret requests from web browsers, retrieve the requested content, and send it back to the browser.

In other words, **all web servers are servers, but not all servers are web servers**. A web server is a type of server that is designed specifically for handling web requests and serving web content, while a server can refer to any computer or software application that provides a service or resource to other devices on a network.

What is XAMPP?

XAMPP is an open-source software package that includes a web server, database server, and other tools for building and testing web applications locally on a personal computer. It was developed by Apache Friends and is available for Windows, Linux, and macOS.

XAMPP Stands for:

X – Cross-Platform => XAMPP is designed to work on multiple platforms, including Windows, Linux, and macOS, making it a versatile tool for web developers.

A – Apache => Apache is the most widely used web server software in the world, and XAMPP includes a version of the Apache HTTP Server that provides the ability to serve web pages and content over the internet or a local network.

M – MySQL => MySQL is a popular open-source database server that allows you to create and manage databases for your web applications.

P – PHP => PHP is a server-side scripting language used for web development that allows you to create dynamic web pages and interact with databases.

P – Perl => Perl is a scripting language that can be used for web development, system administration, and network programming.

Advantages of XAMPP:

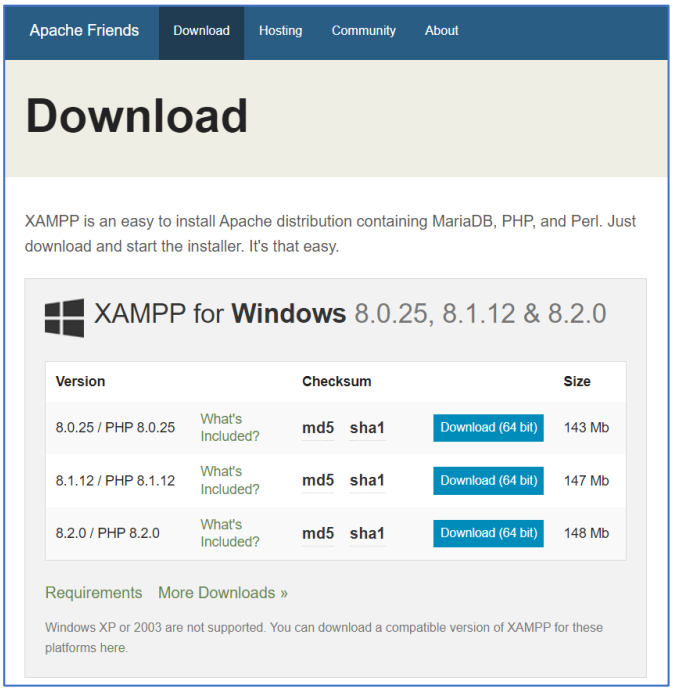
- XAMPP is **free to use** and is available under an **open-source** license.
- XAMPP is **easy to install** and configure, even for beginners.
- It is **cross-platform**, meaning it can be used on Windows, Linux, and macOS.
- XAMPP is customizable, so you can add or remove components as needed.
- It is ideal for **local development** and **testing** before deploying **web applications** to a production server.
- It provides a complete development environment with **all the necessary tools and features**.

Disadvantages of XAMPP:

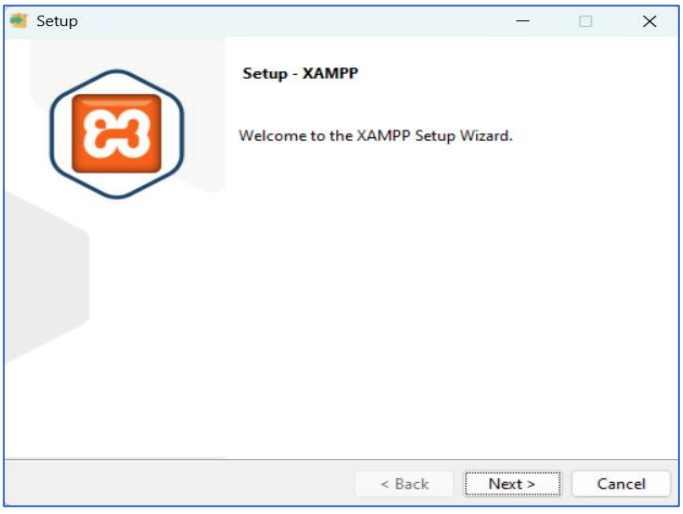
- XAMPP is primarily designed for local development and testing, and may **not be suitable** for production environments or **high-traffic websites**.
- It may require some configuration to ensure **security** and **stability**.
- XAMPP does not include support for all programming languages or tools.
- It can take up a **significant amount of disk space** and memory on your computer.
- Updates and patches may not be released as frequently as other software packages.

Steps To Install XAMPP:

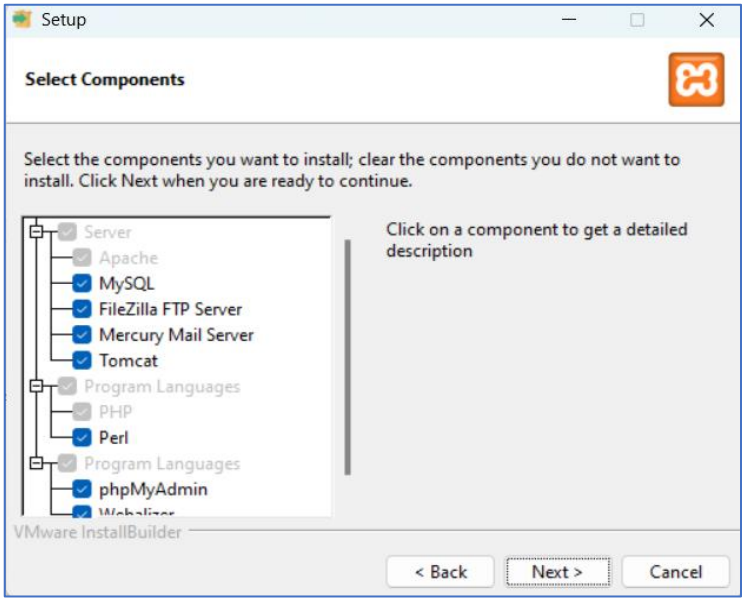
1. Download XAMPP setup from - **https://www.apachefriends.org/download.html**



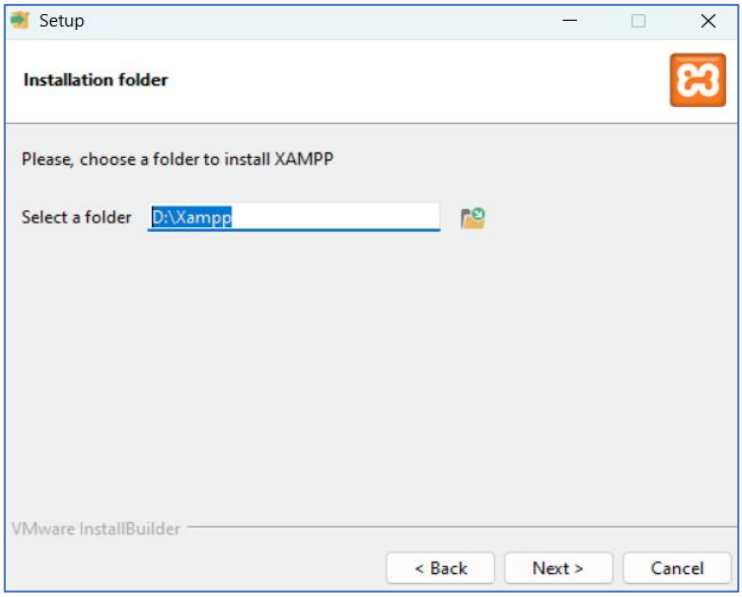
2. Open the XAMPP setup. And click on Next Button.



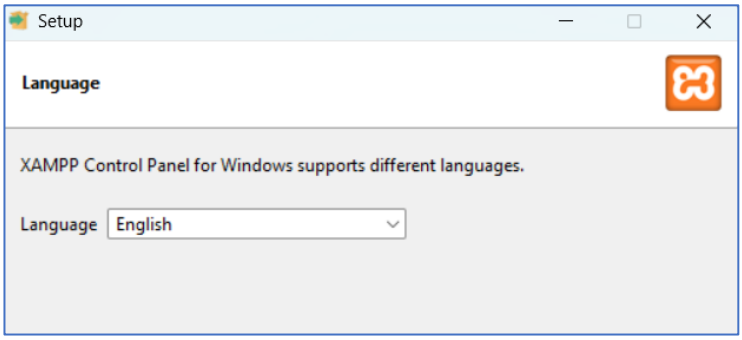
3. Select the components you want to Install. Then click on Next.



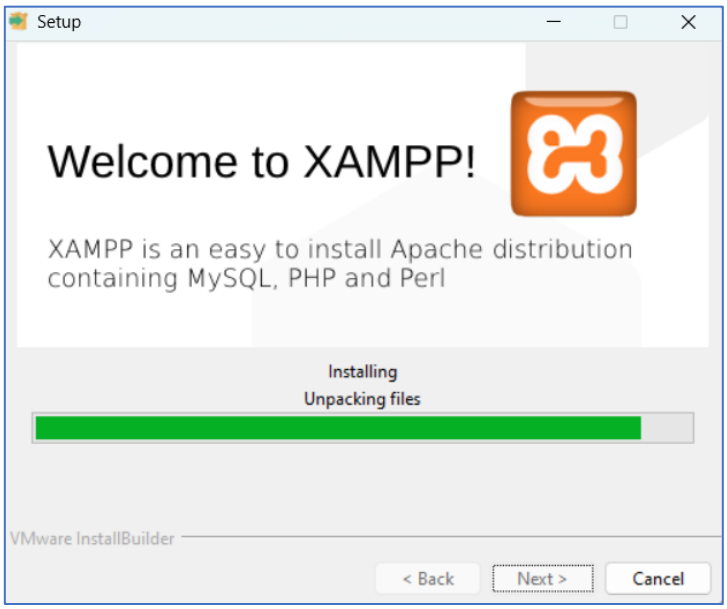
4. Choose folder where you want to install XAMPP. Then click on Next.



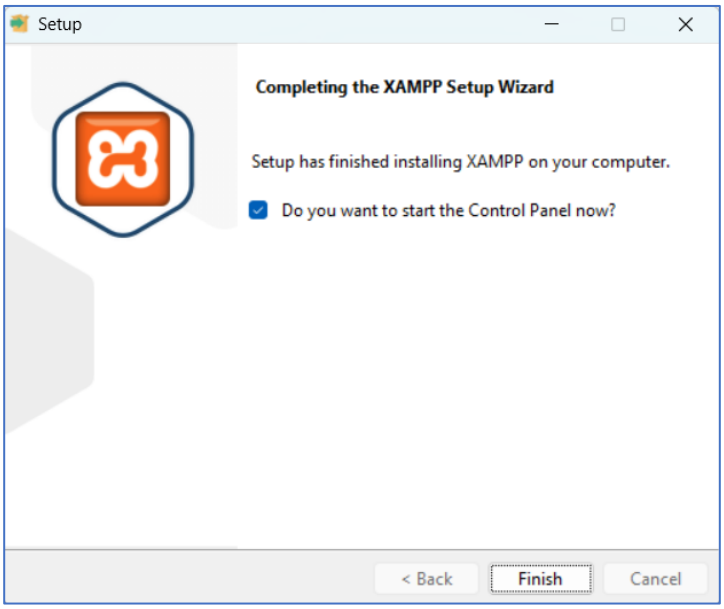
5. Choose Language**.**



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



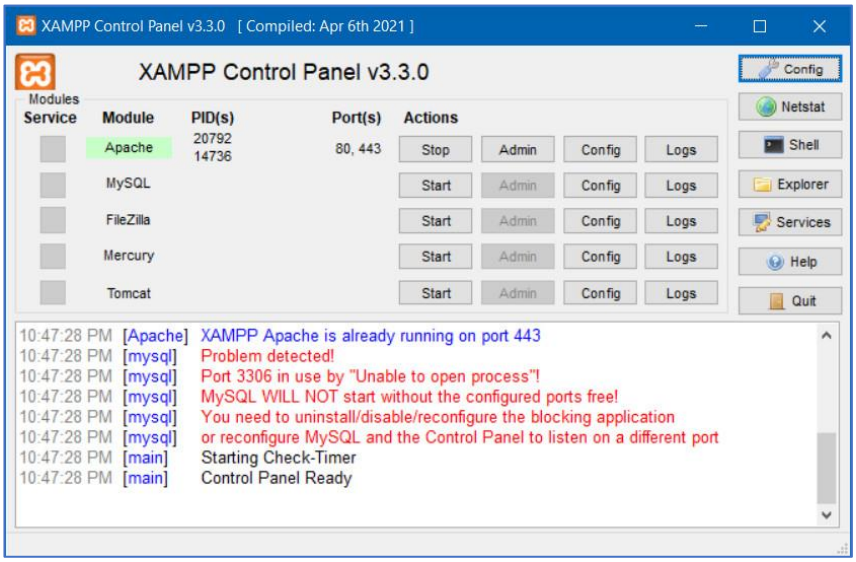
7. After Installation is Complete Click on **Finish.**



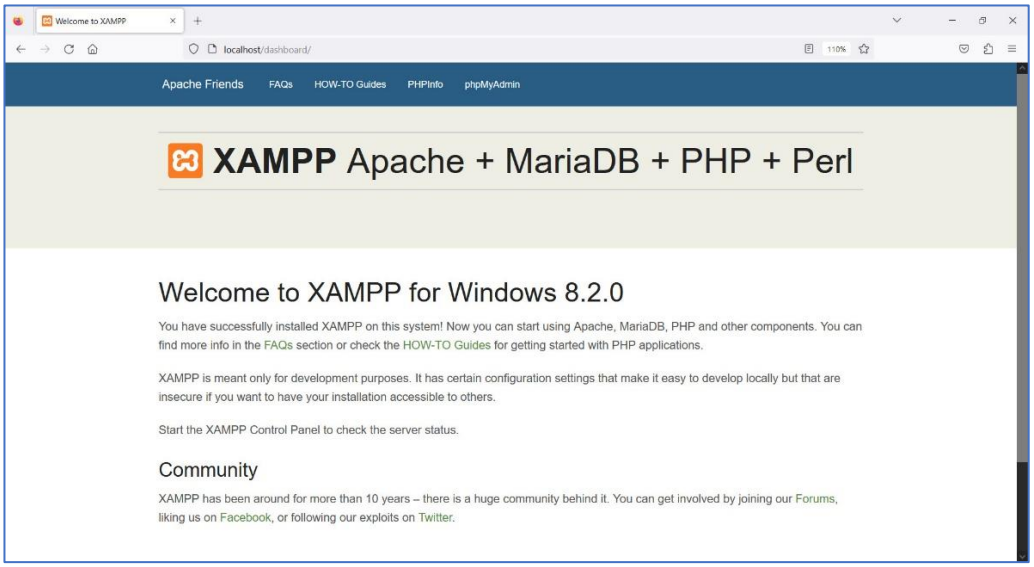
XAMPP is successfully installed on your System.

Steps to Open HTML file in XAMPP:

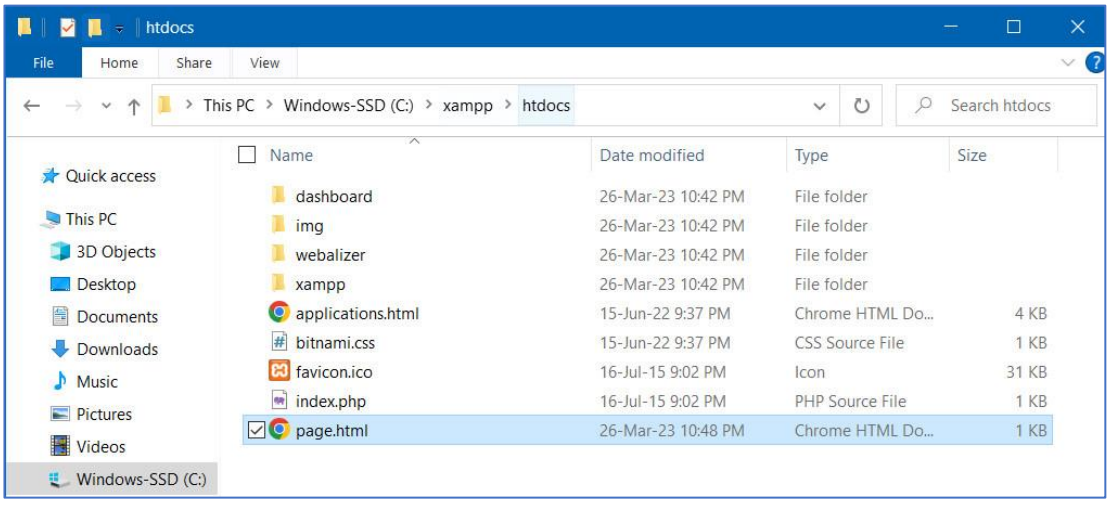
Step 1: Open XAMPP Control Panel.

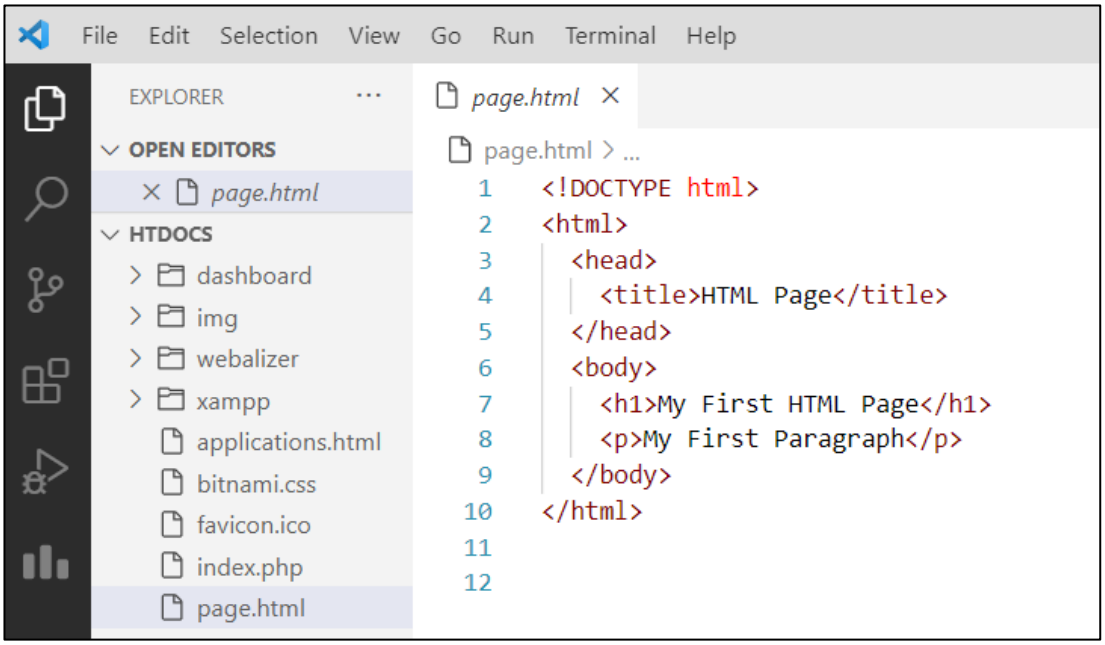


Step 2: Now click on **Start** button in Actions Column to start **Apache**. Then click on **Admin** button to start web page with url: **localhost/dashboard**



Step 3: Then Open folder PATH **C:\xampp\htdocs** then create your html file. (E.g.: page.html).





Step 4: Now type **localhost/page.html** in browser and press Enter. Your html page successfully opens in XAMPP.

Viva Questions:

Q1. Does XAMPP include MySQL or MariaDB?

Ans: Yes, XAMPP includes both MySQL and MariaDB.

Q2. How XAMPP can be started without setup?

Ans: XAMPP can be started without setup by simply running the XAMPP control panel and clicking on the "Start" button next to the services you want to start (such as Apache and MySQL).

Q3. How XAMPP is uninstalled?

Ans: Go to Control Panel. Search Uninstall a Program. Then search XAMPP. Then right click on XAMPP then click Uninstall/Change. Then it will ask do you want to uninstall XAMPP. Click on OK. Then next. XAMPP successfully uninstalled from your system.

Q4. What is the "lite" version of XAMPP?

Ans: XAMPP Lite (means "light" as in "light-weight") is a smaller bundle of XAMPP components, which is recommended for quick work using only PHP and MySQL. Some servers or tools such as Mercury Mail and FileZilla FTP are missing in the Lite version.

Q5. Where main XAMPP configuration files are stored?

Ans: The main XAMPP configuration files are located as follows:

- **Apache** configuration file: \xampp\apache\conf\httpd.conf, \xampp\apache\conf\extra\httpd-xampp.conf
- **PHP** configuration file: \xampp\php\php.ini
- **MySQL** configuration file: \xampp\mysql\bin\my.ini
- **FileZilla** Server configuration file: \xampp\FileZillaFTP\FileZilla Server.xml
- **Apache Tomcat** configuration file: \xampp\tomcat\conf\server.xml
- **Apache Tomcat** configuration file: \xampp\sendmail\sendmail.ini
- **Mercury Mail** configuration file: \xampp\MercuryMail\MERCURY.INI

EXPERIMENT – 2

AIM: Design a web-page using basic HTML tags.

THEORY:

HTML (Hypertext Markup Language) tags are used to define the structure and content of a web page. HTML tags are enclosed in angle brackets, and usually come in pairs: an opening tag and a closing tag. The opening tag begins with the name of the tag, followed by any attributes, which are used to provide additional information about the element. The closing tag begins with a forward slash (/) before the tag name.

For example: This is a paragraph of text.

In this example, the **opening tag is ** and the **closing tag is **. The text "This is a paragraph of text." is the content of the paragraph.

HTML tags can be used to define headings, paragraphs, lists, tables, images, links, forms, and many other elements and structures of a web page. By using HTML tags, web developers can create structured and organized content that can be easily displayed and accessed by web browsers.

HTML Tags	**Definition**	**Attributes**	**Example**
<html>	Defines the root element of an HTML document.	**lang:** for specifying language.	<html lang="en">
<head>	Defines the head section of an HTML document.	None	<head>
<title>	Defines the title of an HTML document.	None	<title>Page Title</title>
<body>	Defines the body section of an HTML document.	**bgcolor**: background color	<body bgcolor="#ffffff">
	Defines a paragraph of text.	None	This is a paragraph of text.
<h1> to <h6>	Defines headings of varying sizes.	None	This is a level 1 heading.

	Defines a division or section of an HTML document.	**class**: specifies the CSS class.	Content goes here.
	Defines bold text.	None	This text is bold.
	Defines italicized text.	None	This text is italicized.
<u>	Defines underlined text.	None	<u>This text is underlined.</u>
 	Defines a line break or carriage return in a block of text.	None	This is a line of text. This is the next line of text.
<hr>	Defines a horizontal rule or line in an HTML document.	**size:** set size of line **width:** width of line **color:** change color	<hr size="2" width="50%" color="#ff0000">
	Defines an unordered list.	**type:** specifies the type of bullet point	<ul type="circle"> - Item 1 - Item 2
	Defines an ordered list.	**start:** specifies the starting number for the list. **type:** specifies the type of list.	<ol start="3"> - Item 1 - Item 2
	Defines a list item.	None	- Item 1 - Item 2
<address>	Defines contact information for the author or owner of an HTML document.	None	<address>123 Main Street, Anytown USA</address>
<dl>	Defines a description list.	None	<dl><dt>Term1</dt><dd>Description 1</dd></dl>
<dt>	Defines a term (name) in a description list.	None	<dl><dt>Term2</dt><dd>Description 2</dd></dl>
<dd>	Defines the description of a term in a description list.	None	<dl><dt>Term2</dt><dd>Description 3</dd></dl>

SOURCE CODE:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Page</title>
  </head>
  <body>
    <h1>This is My Simple HTML Page</h1>
    <p> HTML Paragraph or HTML < p > tag is used to define a paragraph in a webpage. Let's take a simple example to see how it work. It is a notable point that a browser itself add an empty line before and after a paragraph. An HTML "< p >" tag indicates starting of new paragraph. </p>
    <hr />
    <p> HTML < hr > tag is used to specify a paragraph-level thematic break in HTML document. It is used when you abruptly change your topic in your HTML document. </p>

    <p> It draw a horizontal line between them. It is also called a Horizontal Rule in HTML. </p> <hr />
    <br />
    <p> This is br tag used to break line. As we can see there is space between above line. </p> <hr />

    <h1> These are Heading Tags from < h1 > to < h6 > </h1>
    <h1> This is heading 1 </h1>
    <h2> This is heading 2 </h2>
    <h3> This is heading 3 </h3>
    <h4> This is heading 4 </h4>
    <h5> This is heading 5 </h5>
    <h6> This is heading 6 </h6>     <hr />

    <h2>Image: < img > tag</h2>
    <img src="image.jpg" alt="image" width="auto" height="200" /> <hr />

  <h2>Unordered List: < ul > tag</h2>
    <ul type="disc">
      <li>Bus</li>
```

```
    <li>Train</li>
    <li>Car</li>
  </ul>      <hr />

  <h2>Ordered List: < ol > tag</h2>
  <ol type="I">
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ol>       <hr />

  <h2>Address Tag: < address > tag</h2>
  <address>
    Rahul Kumar <br />   123 Main Street <br />
    New Delhi, Delhi - 110058 <br />   India <br /> <br />

    Phone: +91-9876543210 <br />
    Email: rahul@gmail.com
  </address>   <hr />

  <h2>Definition List Tag: < dl > < dt > < dd > tag</h2>
  <dl> <dt> Social Media </dt>
    <dd>  <ul>
        <li><a href="#">Twitter</a></li>
        <li><a href="#">Facebook</a></li>
        <li><a href="#">Instagram</a></li>
      </ul> </dd>
  </dl>
 </body>
</html>
```

This is My Simple HTML Page

HTML Paragraph or HTML < p > tag is used to define a paragraph in a webpage. Let's take a simple example to see how it work. It is a notable point that a browser itself add an empty line before and after a paragraph. An HTML "< p >" tag indicates starting of new paragraph.

HTML < hr > tag is used to specify a paragraph-level thematic break in HTML document. It is used when you abruptly change your topic in your HTML document.

It draw a horizontal line between them. It is also called a Horizontal Rule in HTML.

This is br tag used to break line. As we can see there is space between above line.

These are Heading Tags from < h1 > to < h6 >

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

Image: < img > tag

Unordered List: < ul > tag

- Bus
- Train
- Car
- Jeep

Ordered List: < ol > tag

I. Item 1
II. Item 2
III. Item 3
IV. Item 4

Address Tag: < address > tag

Rahul Kumar
123 Main Street
New Delhi, Delhi - 110058
India

Phone: +91-9876543210
Email: rahul@gmail.com

Definition List Tag: < dl > < dt > < dd > tag

Social Media

- Twitter
- Facebook
- Instagram

Viva Questions:

Q1. What is HTML?

Ans: HTML stands for Hypertext Markup Language. It is the standard markup language used for creating web pages and web applications. HTML provides the structure for content and displays the content in a web browser.

Q2. What are tags?

Ans: 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. But some HTML tags are unclosed tags.

- 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)

Q3. Do all HTML tags come in pair?

Ans: No, not all HTML tags come in pairs. Some tags are self-closing, meaning that they do not require a closing tag. For example, the
 tag is a self-closing tag that creates a line break, and the <img> tag is also a self-closing tag that inserts an image into an HTML document.

Q4. What are some of the common lists that can be used when designing a page?

Ans: Some common lists that can be used when designing a web page include ordered lists (<ol>) and unordered lists (<ul>). Ordered lists are used for lists where the order of items is important, while unordered lists are used for lists where the order of items is not important.

Q5. How do you insert a comment in HTML?

Ans: To insert a comment in HTML, use the following syntax: <!-- This is a comment -->. Anything between the <!-- and --> tags will be treated as a comment and will not be displayed in the web browser.

Q6. What is the correct HTML element for inserting a line break?

Ans: The correct HTML element for inserting a line break is the
 tag. This tag is a self-closing tag, meaning that it does not require a closing tag. To insert a line break in an HTML document, simply add the
 tag where you want the line break to occur.

Q7. What is the correct HTML for adding a background color?

Ans: The correct HTML for adding a background color is to use the style attribute with the background-color property. For example, to add a red background color to an HTML element, use the following syntax: This is a red div .

Experiment – 3

AIM: Design a web-page using Table and Form in HTML.

THEORY:

HTML TABLE

An HTML table is a way to display data in a structured format on a web page. It consists of rows and columns, where each cell in the table contains a piece of data. Tables can be used for a variety of purposes, such as displaying tabular data, organizing content into a grid, or creating a layout for a web page.

In HTML, a table is created using the <table> tag. The table is divided into two main sections: the <thead> section and the <tbody> section. The <thead> section contains the header row(s) of the table, while the <tbody> section contains the data rows.

Each row in the table is represented by a <tr> tag, and each cell in the row is represented by a <td> tag. In the header row, the cells are typically represented by <th> tags instead of <td> tags, which provides a visual distinction between the header cells and the data cells.

S. No	Tags	Definition
1.	<table>	It defines a table.
2.	<tr>	It defines a row in a table.
3.	<th>	It defines a header cell in a table.
4.	<td>	It defines a cell in a table.
5.	<caption>	It defines the table caption.
6.	<colgroup>	It specifies a group of one or more columns in a table for formatting.
7.	<col>	It is used with <colgroup> element to specify column properties for each column.
8.	<tbody>	It is used to group the body content in a table.
9.	<thead>	It is used to group the header content in a table.
10.	<tfooter>	It is used to group the footer content in a table.

HTML Tables:

- 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.

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.

Uses of HTML Form:

- Forms are used to collect user input and send it to a server for processing.
- Forms are created using the **<form>** tag and can contain a variety of input elements like text fields, checkboxes, radio buttons, dropdown menus, and more.
- Each input element is represented by a specific HTML tag, such as **<input>**, **<textarea>**, **<select>**, etc.
- Forms typically use the HTTP POST method to submit data to the server, although they can also use the GET method.
- The data collected through a form can be used to populate a database, send an email, or perform some other action on the server.
- Forms can be styled using CSS to create visually appealing designs.
- Forms can have validation to ensure that users enter the correct information.
- HTML5 provides built-in validation attributes like **required**, **min**, **max**, **pattern**, etc.
- Forms can be submitted using JavaScript to add extra functionality and interactivity to the form.
- JavaScript can also be used to dynamically populate form elements, depending on user input.
- Forms can also be submitted using AJAX (Asynchronous JavaScript and XML) to update parts of the page without refreshing the entire page.

HTML Form Elements:

Name	**Definition**	**Syntax**
<input>	A basic input element that can accept various types of user input.	<input type="text" name="example">
<textarea>	An element for multiline text input.	<textarea name="example"> </textarea>
<select>	A dropdown list of options for the user to select from.	<select name="example"><option value="1"> Option 1 </option> </select>
<option>	An option within a <select> element.	<option value="1"> Option 1 </option>
<label>	A label for an input element.	<label for="example"> Example Label </label>
<button>	A button element.	<button type="submit"> Submit </button>
<fieldset>	A grouping element for related form elements.	<fieldset> <legend> Example Group </legend> </fieldset>
<legend>	A caption for a <fieldset> element.	<legend> Example Group </legend>
<input type="checkbox">	A checkbox that allows the user to select one or more options.	<input type="checkbox" name="example" value="1">
<input type="radio">	A radio button that allows the user to select one option from a group.	<input type="radio" name="example" value="1">
<input type="email">	An input element for email addresses.	<input type="email" name="example">
<input type="password">	An input element for passwords.	<input type="password" name="example">
<input type="submit">	A submit button that submits the form.	<input type="submit" value="Submit">
<input type="reset">	A reset button that resets the form to its initial state.	<input type="reset" value="Reset">

SOURCE CODE:

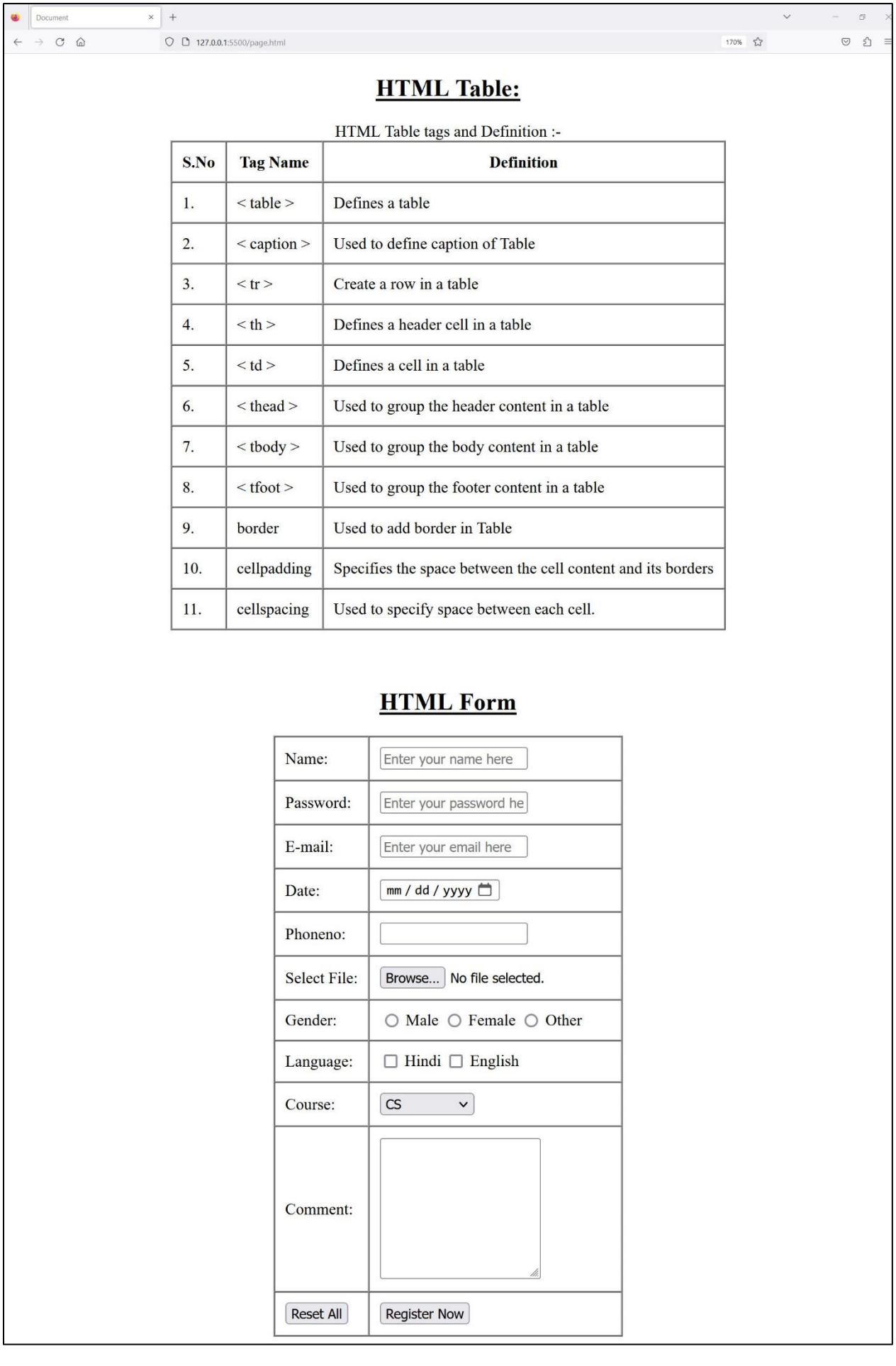
```
<!DOCTYPE html>
<html>
<head>
   <title>Document</title>
</head>
<body>

   <center>
      <h2> <u> HTML Table: </u> </h2>

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

         <caption align="top">HTML Table tags and Definition :- </caption>
         <thead>
            <tr> <th>S.No</th> <th>Tag Name</th>  <th>Definition</th> </tr>
         </thead>

         <tbody>

            <tr> <td>1.</td> <td> < table > </td> <td>Defines a table</td></tr>
            <tr> <td>2.</td> <td> < caption > </td> <td>Used to define caption of Table</td></tr>

            <tr> <td>3.</td> <td> < tr > </td> <td>Create a row in a table</td></tr>
            <tr> <td>4.</td> <td> < th > </td> <td>Defines a header cell in a table</td></tr>

            <tr> <td>5.</td> <td> < td > </td> <td>Defines a cell in a table</td></tr>

            <tr> <td>6.</td> <td> < thead > </td> <td>Used to group the header content in a table</td></tr>
            <tr> <td>7.</td> <td> < tbody > </td> <td>Used to group the body content in a table</td></tr>

            <tr> <td>8.</td> <td> < tfoot > </td> <td>Used to group the footer content in a table</td></tr>
```

```
        <tr> <td>9. </td> <td>border</td> <td>Used to add border in Table </td> </tr>

        <tr>
            <td>10. </td> <td>cellpadding</td>
            <td>Specifies the space between the cell content and its borders </td>
        </tr>

        <tr>
            <td>11. </td> <td>cellspacing</td> <td>Used to specify space between each cell. </td>
        </tr>
      </tbody>
   </table>
</center>

<br><br>

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

   <form action="">
      <table border="1px" cellpadding="10px" cellspacing="0px">
         <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">Phoneno:</label> </td> <td> <input type="tel"></td>
</tr>

<tr>
   <td> <label for="File">Select File:</label> </td> <td> <input type="file"></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="lang">Language:</label> </td>
   <td>
      <input type="checkbox" id="hindi" name="hindi"> <label for="hindi">Hindi</label>
      <input type="checkbox" id="english" name="english"> <label for="english">English</label>
   </td>
</tr>
```

```
            <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> <label for="comment">Comment:</label> </td>
               <td> <textarea name="comment" id="comment" cols="20" rows="8"></textarea></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:



HTML Table:

HTML Table tags and Definition :-

S.No	Tag Name	Definition
1.	< table >	Defines a table
2.	< caption >	Used to define caption of Table
3.	< tr >	Create a row in a table
4.	< th >	Defines a header cell in a table
5.	< td >	Defines a cell in a table
6.	< thead >	Used to group the header content in a table
7.	< tbody >	Used to group the body content in a table
8.	< tfoot >	Used to group the footer content in a table
9.	border	Used to add border in Table
10.	cellpadding	Specifies the space between the cell content and its borders
11.	cellspacing	Used to specify space between each cell.

HTML Form

Name:	Enter your name here
Password:	Enter your password he
E-mail:	Enter your email here
Date:	mm / dd / yyyy
Phoneno:	
Select File:	Browse... No file selected.
Gender:	Male Female Other
Language:	Hindi English
Course:	CS
Comment:	
Reset All	Register Now

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 (;).

Example:

```
<html>
    <head>
        <title>test</title>
    </head>
<body>
<h1 style = “font-family: “Name of Font”; font-size: “size of font”; color: “color which you want to give to your text””>
Text to be shown on web <h1>.
</body>
</html>
```

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.

Example:

```
<html>
  <head>
```

```
    <style>
        p{font-size: “Size of font”; color: “color-Name” }
    </style>
</head>
<body>
<p>Text which display on web with specified size and color in style tag</p>
</body>
</html>
```

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.

Example:

styles.css:

```
p{
font-size: “size”;
color: “color name”;
}
```

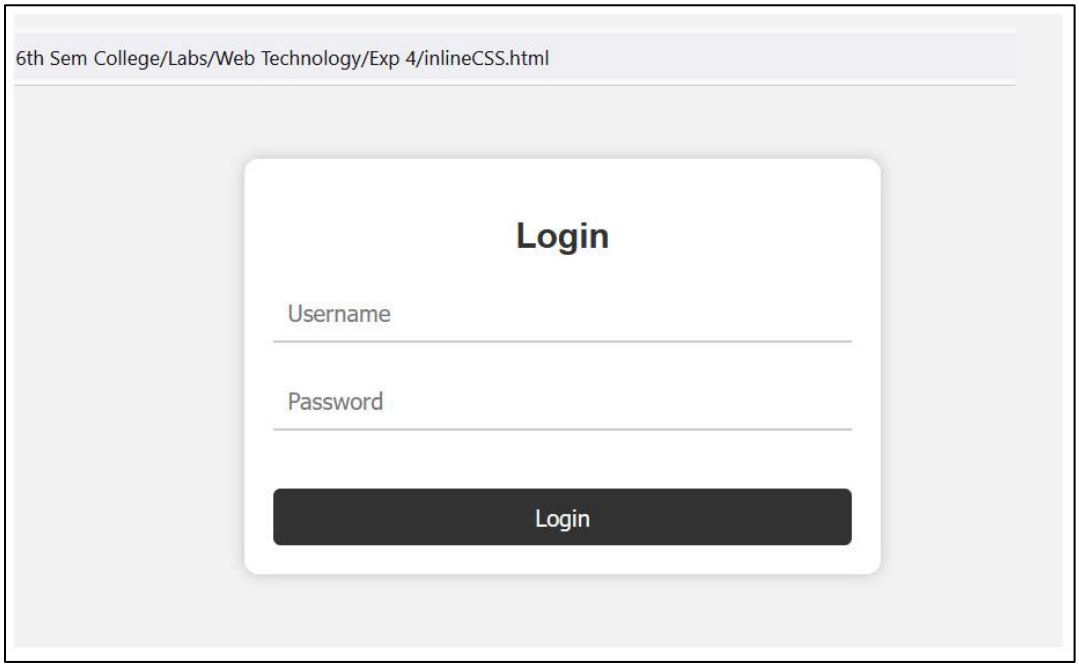
index.html:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<p> Text which display on web with specified size and color in styles.css</p>
</body>
</html>
```

CODE => INLINE CSS:

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
</head>

<body style="background-color: #f2f2f2; font-family: Arial, sans-serif; box-sizing: border-box; margin: 0; padding: 0;">

  <form style="background-color: #fff; border-radius: 10px; box-shadow: 0 0 10px rgba(0, 0, 0, 0.2); margin: 50px auto; max-width: 400px; padding: 20px;">

    <h2 style="color: #333; font-size: 24px; margin-bottom: 20px; text-align: center;">Login</h2>

    <input type="text" placeholder="Username" style="border: none; border-bottom: 2px solid #ccc; display: block; font-size: 16px; margin-bottom: 20px; padding: 10px; width: 95%;" />

    <input type="password" placeholder="Password" style="border: none; border-bottom: 2px solid #ccc; display: block; font-size: 16px; margin-bottom: 20px; padding: 10px; width: 95%;" />

    <input type="submit" value="Login" style="background-color: #333; border: none; border-radius: 5px; color: #fff; cursor: pointer; font-size: 16px; margin-top: 20px; padding: 10px; width: 100%;" />

  </form>
</body>
</html>
```

OUTPUT:

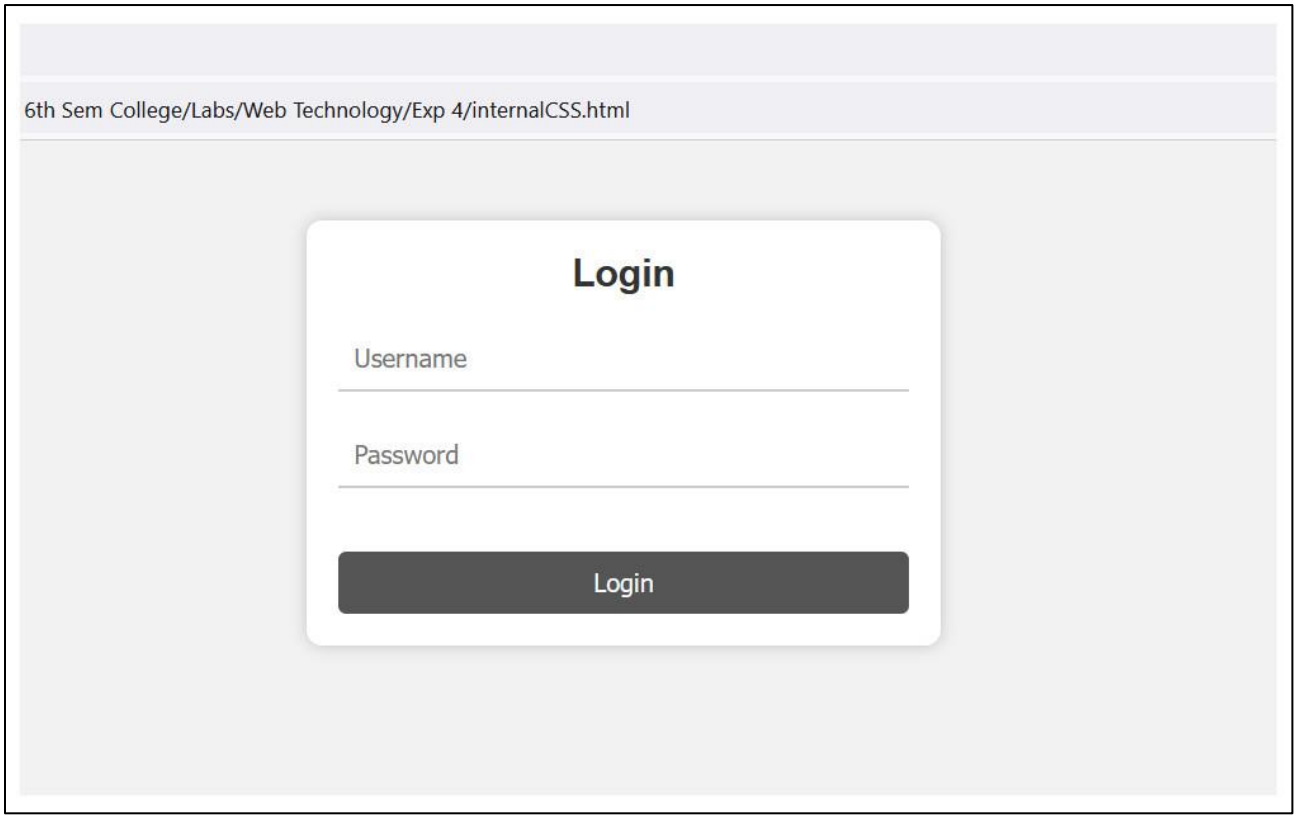


CODE => INTERNAL CSS:

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
  <style>
    * {
      box-sizing: border-box;
      margin: 0; padding: 0;
    }
    body {
      background-color: #f2f2f2;
      font-family: Arial, sans-serif;
    }
    form {
      background-color: #fff; border-radius: 10px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
      margin: 50px auto; max-width: 400px;
      padding: 20px;
    }
    h2 {
      color: #333; font-size: 24px;
      margin-bottom: 20px; text-align: center;
    }
    input[type="text"], input[type="password"] {
      border: none; border-bottom: 2px solid #ccc;
      display: block; font-size: 16px;
      margin-bottom: 20px; padding: 10px;
      width: 100%;
    }
    input[type="submit"] {
      background-color: #333; border: none;
      border-radius: 5px; color: #fff;
```

```
    cursor: pointer; font-size: 16px;
    margin-top: 20px; padding: 10px;
    width: 100%;
   }
   input[type="submit"]:hover {
    background-color: #555;
   }
  </style>
</head>
<body>
  <form>
   <h2>Login</h2>
   <input type="text" placeholder="Username" />
   <input type="password" placeholder="Password" />
   <input type="submit" value="Login" />
  </form>
</body>
</html>
```

OUTPUT:



CODE => EXTERNAL CSS:

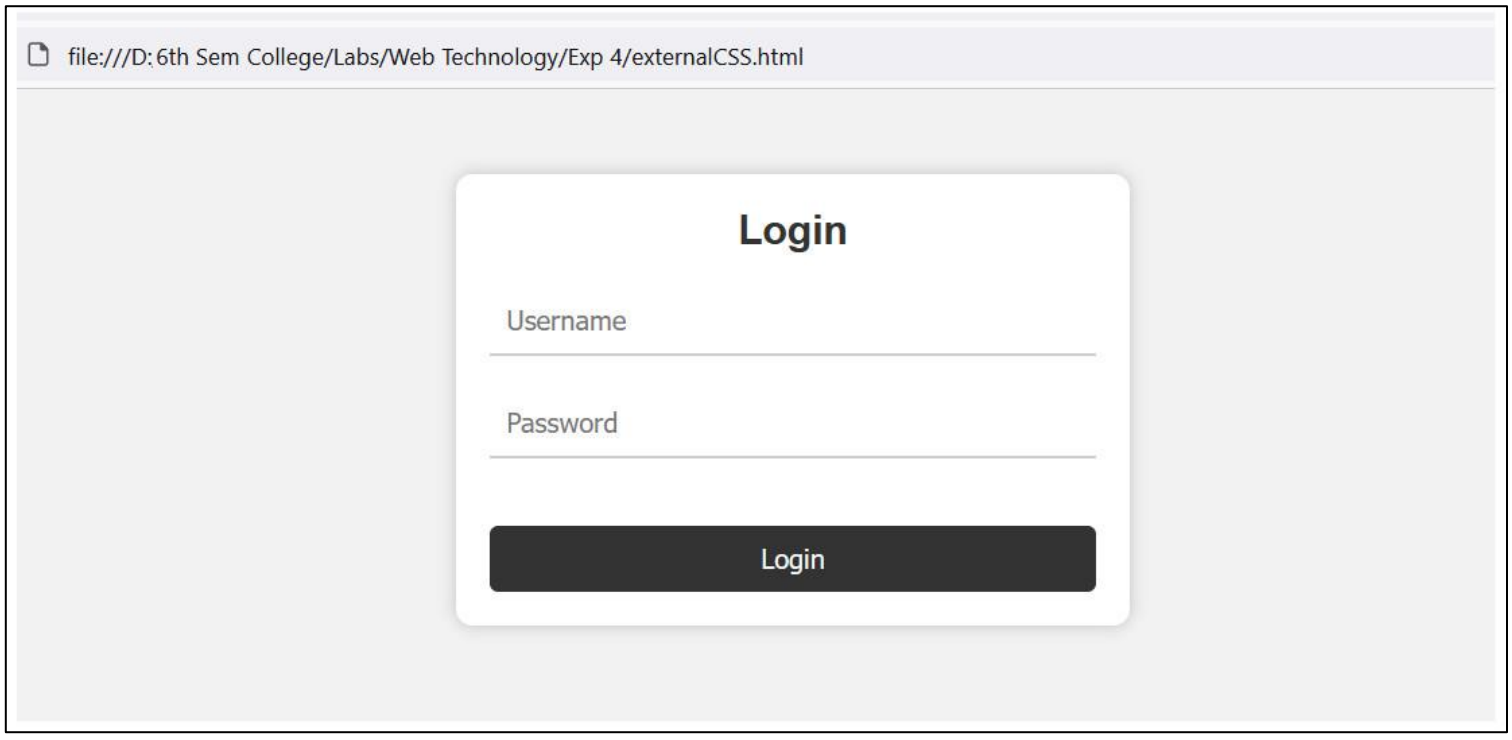
File: ***style.css***

```
* {
  box-sizing: border-box;
  margin: 0; padding: 0;
}
body {
  background-color: #f2f2f2;
  font-family: Arial, sans-serif;
}
form {
  background-color: #fff; border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
  margin: 50px auto; max-width: 400px;
  padding: 20px;
}
h2 {
  color: #333; font-size: 24px;
  margin-bottom: 20px; text-align: center;
}
input[type="text"], input[type="password"] {
  border: none; border-bottom: 2px solid #ccc;
  display: block; font-size: 16px;
  margin-bottom: 20px; padding: 10px; width: 100%;
}
input[type="submit"] {
  background-color: #333; border: none;
  border-radius: 5px; color: #fff;
  cursor: pointer; font-size: 16px;
  margin-top: 20px; padding: 10px;
  width: 100%;
}
input[type="submit"]:hover {
  background-color: #555;
}
```

File: ***externalCSS.html***

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
  <link rel="stylesheet" type="text/css" href="login.css">
</head>
<body>
  <form>
    <h2>Login</h2>
    <input type="text" placeholder="Username" />
    <input type="password" placeholder="Password" />
    <input type="submit" value="Login" />
  </form>
</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:

What is XML?

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.

What is XML Used For?

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.

Advantages of XML:

1. **Redundancy:** - XML markup is very verbose. For example, every end tag must be supplied, such as </description> 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>John Doe</name>
    <rollno>101</rollno>
    <age>20</age>
  </student>

  <student>
    <name>Jane Smith</name>
    <rollno>102</rollno>
    <age>21</age>
  </student>
</students>
```

Difference between XML and HTML:

1)	HTML is used **to display data** and focuses on how data looks.	XML is a software and hardware independent tool used **to transport and store data**. It focuses on what data is.
2)	HTML is a **markup language** itself.	XML provides a **framework to define markup languages**.
3)	HTML is **not case sensitive**.	XML is **case sensitive**.
4)	HTML is a presentation language.	XML is neither a presentation language nor a programming language.
5)	HTML **has its own predefined tags**.	You **can define tags according to your need**.
6)	In HTML, it is **not necessary to use a closing tag**.	XML **makes it mandatory to use a closing tag**.
7)	HTML is **static** because it is used to display data.	XML is **dynamic** because it is used to transport data.
8)	HTML **does not preserve whitespaces**.	XML **preserve whitespaces**.

Rules for well-formed XML:

- It must begin with the XML declaration.
- It must have one unique root element.
- All start tags of XML documents must match end tags.
- XML tags are case sensitive.
- All elements must be closed.
- All elements must be properly nested.
- All attributes values must be quoted.
- XML entities must be used for special characters.

XML Tree Structure:

XML documents are formed as element trees.

An XML tree starts at a root element and branches from the root to child elements.

All elements can have sub elements (child elements):

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

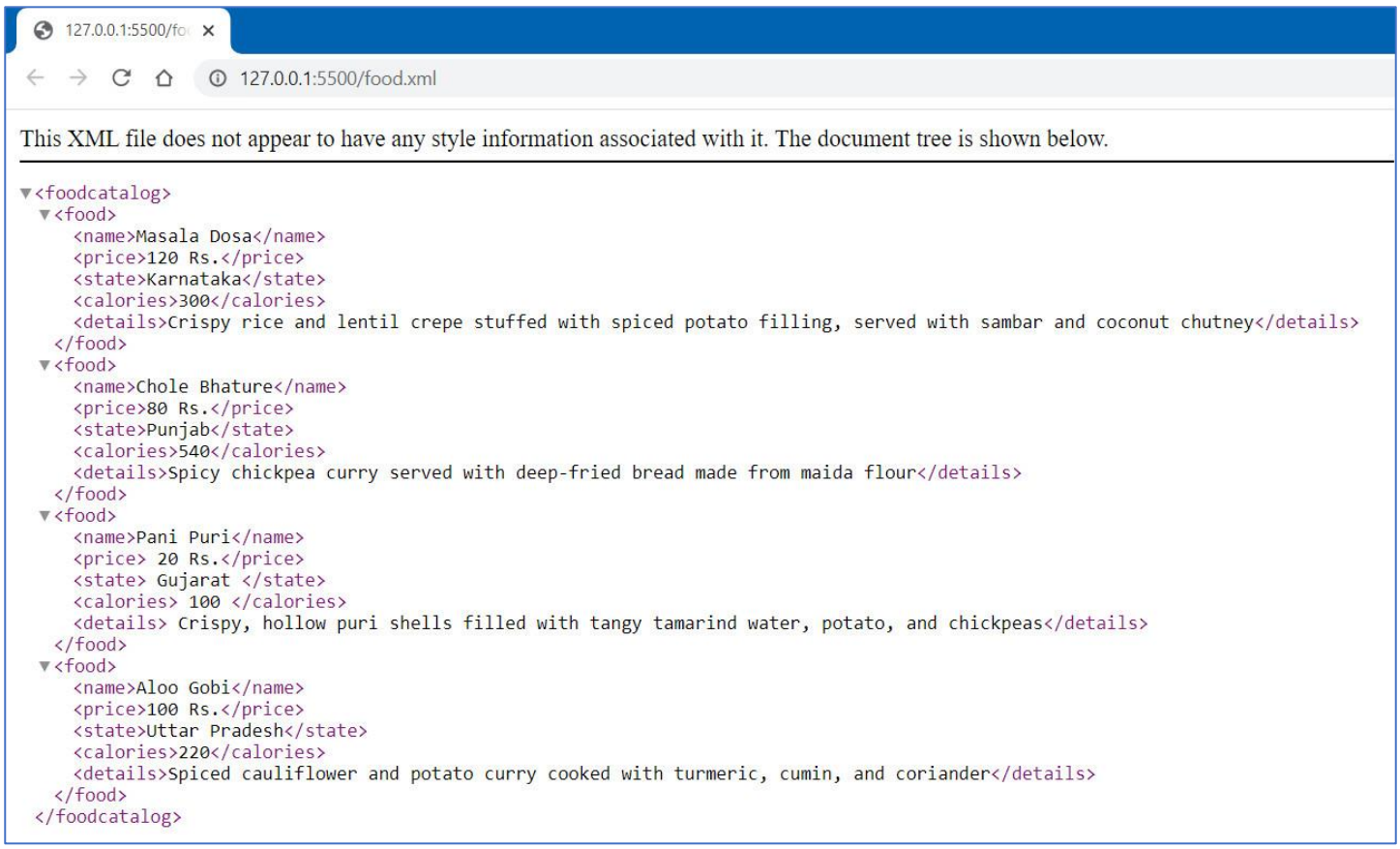
The terms parent, child, and sibling are used to describe the relationships between elements.

Parents have children. Children have parents. Siblings are children on the same level (brothers and sisters)

SOURCE CODE (XML):

```
<?xml version="1.0" encoding="UTF-8"?>
<foodcatalog>
   <food>
      <name>Masala Dosa</name>
      <price>120 Rs.</price>
      <state>Karnataka</state>
      <calories>300</calories>
      <details>Crispy rice stuffed with spiced potato filling, served with sambar and coconut chutney</details>
   </food>

   <food>
      <name>Chole Bhature</name>
      <price>80 Rs.</price>
      <state>Punjab</state>
      <calories>540</calories>
      <details>Spicy chickpea curry served with deep-fried bread made from maida flour</details>
   </food>

   <food>
      <name>Pani Puri</name>
      <price> 20 Rs.</price>
      <state> Gujarat </state>
      <calories> 100 </calories>
      <details> Crispy, hollow puri shells filled with tangy tamarind water, potato, and chickpeas</details>
   </food>

   <food>
      <name>Aloo Gobi</name>
      <price>100 Rs.</price>
      <state>Uttar Pradesh</state>
      <calories>220</calories>
      <details>Spiced cauliflower and potato curry cooked with turmeric, cumin, and coriander</details>
   </food>
</foodcatalog>
```

OUTPUT:



SOURCE CODE (XML + CSS)

File: cd.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="style.css"?>

<cdcatalog>
   <heading>CD Catalogs:</heading>

   <song>
      <name> Despacito </name>
      <singer> Luis Fonsi, Daddy Yankee </singer>
      <genre> Latin Pop </genre>
      <year> 2017 </year>
      <country> Puerto Rico </country>
   </song>
```

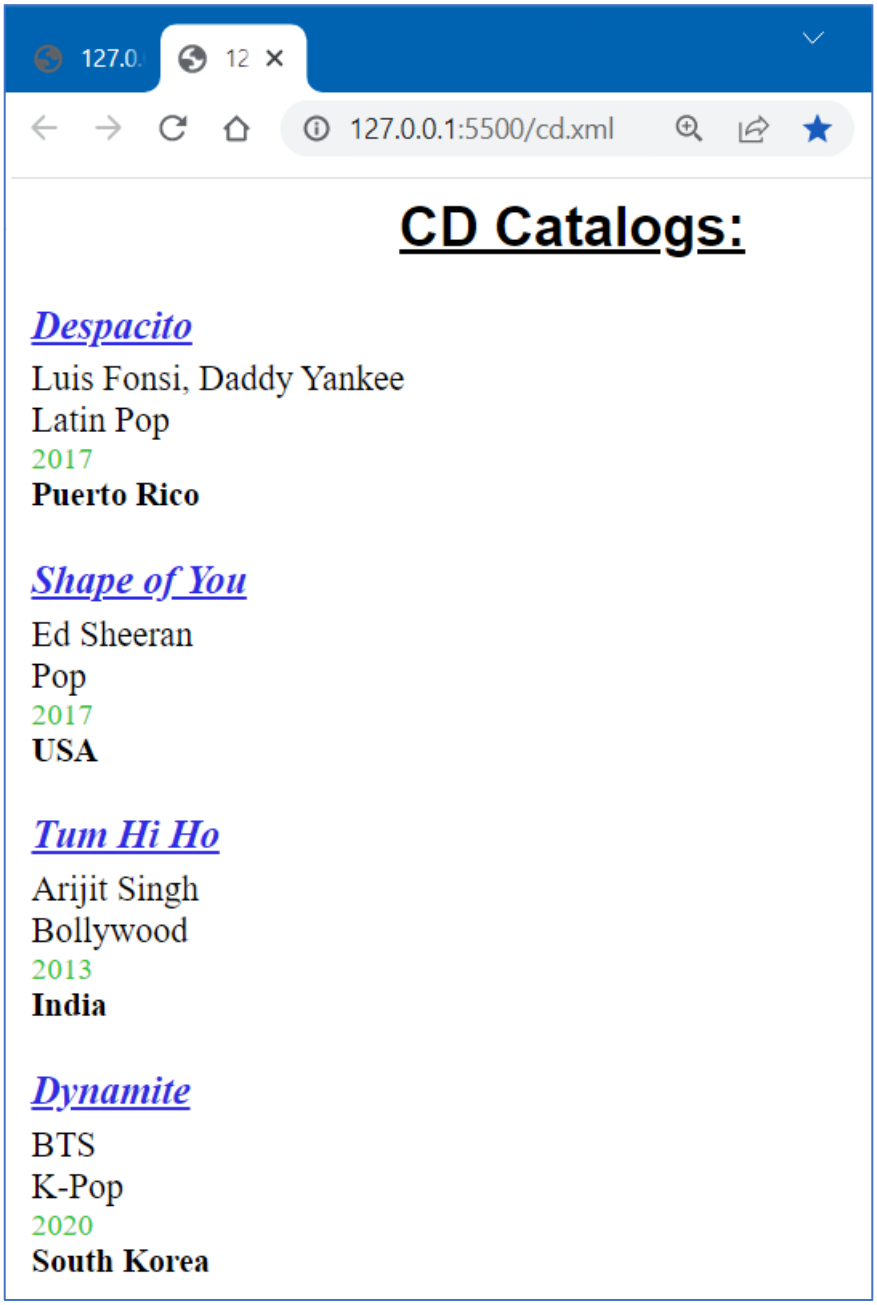
```
    <song>
        <name> Shape of You </name>
        <singer> Ed Sheeran </singer>
        <genre> Pop </genre>
        <year> 2017 </year>
        <country> USA </country>
    </song>

    <song>
        <name> Tum Hi Ho </name>
        <singer> Arijit Singh </singer>
        <genre> Bollywood </genre>
        <year> 2013 </year>
        <country> India </country>
    </song>

    <song>
        <name> Dynamite </name>
        <singer> BTS </singer>
        <genre> K-Pop </genre>
        <year> 2020 </year>
        <country> South Korea </country>
    </song>

</cdcatalog>
```

File: style.css

```
song, name, singer, genre, year, country, heading{
    display: block;
}

heading{
    font-family: 'Arial', sans-serif;
    font-size: 25px;
    font-weight: 700;
```

```
    text-align: center;
    text-decoration: underline;
    margin: 8px;
}

song{
    margin-top: 20px;
    margin-left: 12px;
}

name{
    margin-bottom: 5px;
    font-weight: 900;
    font-size: 18px;
    font-style: italic;
    color: rgb(46, 43, 226);
    text-decoration: underline;
}

year{
    color: rgb(43, 197, 46);
    font-size: 14px;
}

country{
    font-size: 15px;
    font-weight: bold;
}
```

OUTPUT: =>



Experiment – 6 (a)

AIM: Design a web-page that displays current Date and Time using JavaScript.

THEORY:

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.

Constructor: 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)

JavaScript Date Methods:

JavaScript date methods with their description.

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.
setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.
setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.

setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
toDateString()	It returns the date portion of a Date object.
toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object during JSON serialization.
toString()	It returns the date in the form of string.

SOURCE CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>  <title>Date & Time</title> </head>
<body> <h1 id="heading"> </h1> </body>
<script lang="javascript">
    function getDate() {
        const date = new Date();
        const heading = document.getElementById("heading");
        heading.innerText = date;
    }
    setInterval(getDate, 1000);
</script>
</html>
```

OUTPUT:

Experiment – 6 (b)

AIM: Make a simple calculator using JavaScript.

THEORY:

HTML DOM (Document Object Model):

In JavaScript, the Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. Essentially, the DOM is an interface between JavaScript and HTML, allowing you to dynamically access and manipulate HTML elements using JavaScript.

To write to the DOM using JavaScript, you can use a number of methods and properties provided by the DOM API. Here are some common ones:

1. **document.getElementById()** - This method returns the element with the specified ID attribute.
2. **document.createElement()** - This method creates a new HTML element.
3. **node.appendChild()** - This method adds a new child node to an existing element.
4. **node.innerHTML** - This property sets or returns the HTML content of an element.
5. **node.innerText** - This property sets or returns the text content of an element.
6. **node.setAttribute()** - This method sets the value of an attribute for an element.

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

Example:

To create a new paragraph element and append it to an existing div element with ID "myDiv", you could use the following code:

```
var myDiv = document.getElementById("myDiv");                // Get a reference to the existing div
var newParagraph = document.createElement("p");              // Create a new paragraph element
newParagraph.innerHTML = "This is some new content!";        // Set the HTML content of the new paragraph
myDiv.appendChild(newParagraph);                             // Add the new paragraph as a child of the existing div
```

JavaScript eval() function:

The **eval()** function in JavaScript is used to evaluate the expression. It is JavaScirpt'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.

Example:

```
let a = 10, b = 20, c = 30

let sum = eval(" a + b + c ");

let mul = eval(" a  * b * c");

let sub = eval(" a  - b");

console.log(sum, mul, sub);
```

Output: 60 6000 -10

onClick() Eventlistner:

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.

Example:

```
<button onClick="alert('Button clicked!')">Click me!</button>
```

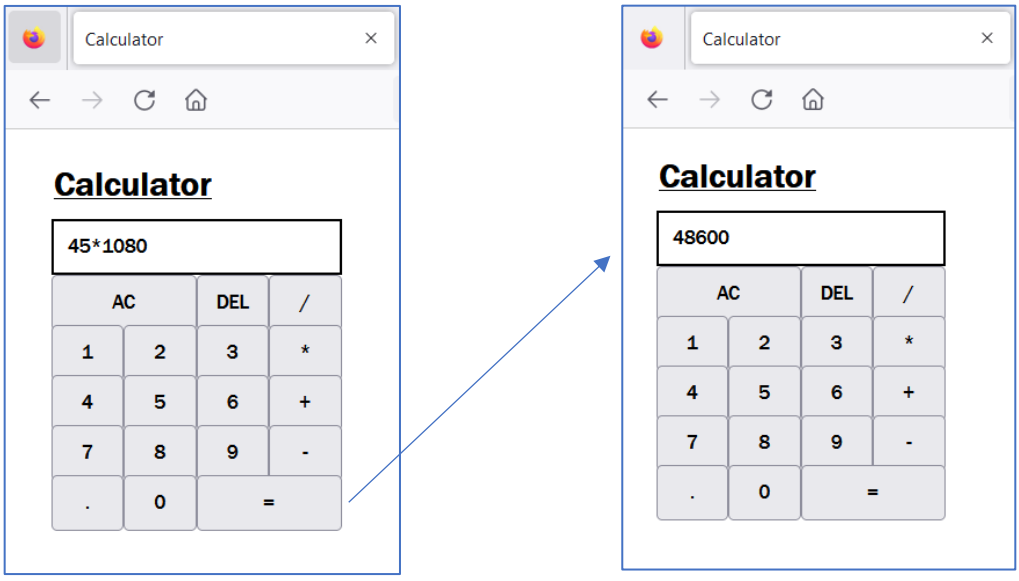
SOURCE CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Calculator</title>

  <style>
    *{
      margin: 0;  padding: 0;  box-sizing: border-box;
      font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    }
    .mainContainer{ margin-top: 80px; margin-left: 80px; }
    .resultScreen{ min-width: 200px; padding: 10px; margin: 10px 0px 2px -2px; border: 2px solid black; }

    input[type=button]{
      padding: 10px; text-align: center;  margin: -2px;
    }
```

```
        .span1{ min-width: 50px;  max-width: 50px;}
        .span2{ min-width: 100px;  max-width: 100px; }

    </style>
</head>

<body>
    <div class="mainContainer">
        <h2> <u>Calculator</u> </h2>
        <div id="calculator">
            <input class="resultScreen" type="text" name="result" id="result">
            <br>
                <input class="span2 ac" type="button" value="AC" onclick="clearResult()">
                <input class="span1" type="button" value="DEL" onclick="deleteResult()">
                <input class="span1" type="button" value="/" onclick="addToResult('/')">
            <br>
                <input class="span1" type="button" value="1" onclick="addToResult('1')">
                <input class="span1" type="button" value="2" onclick="addToResult('2')">
                <input class="span1" type="button" value="3" onclick="addToResult('3')">
                <input class="span1" type="button" value="*" onclick="addToResult('*')">
            <br>
                <input class="span1" type="button" value="4" onclick="addToResult('4')">
                <input class="span1" type="button" value="5" onclick="addToResult('5')">
                <input class="span1" type="button" value="6" onclick="addToResult('6')">
                <input class="span1" type="button" value="+" onclick="addToResult('+')">
            <br>
                <input class="span1" type="button" value="7" onclick="addToResult('7')">
                <input class="span1" type="button" value="8" onclick="addToResult('8')">
                <input class="span1" type="button" value="9" onclick="addToResult('9')">
                <input class="span1" type="button" value="-" onclick="addToResult('-')">
            <br>
                <input class="span1" type="button" value="." onclick="addToResult('.')">
```

```
            <input class="span1" type="button" value="0" onclick="addToResult('0')">
            <input class="span2" type="button" value="=" onclick="calculateResult()">
        <br>

    </div>
</div>

<script>
    function addToResult(value) {document.getElementById('result').value += value;}
    function clearResult() { document.getElementById('result').value = '';}

    function deleteResult(){
        var result = document.getElementById('result').value;
        document.getElementById('result').value = result.substring(0, result.length - 1);
    }

    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.

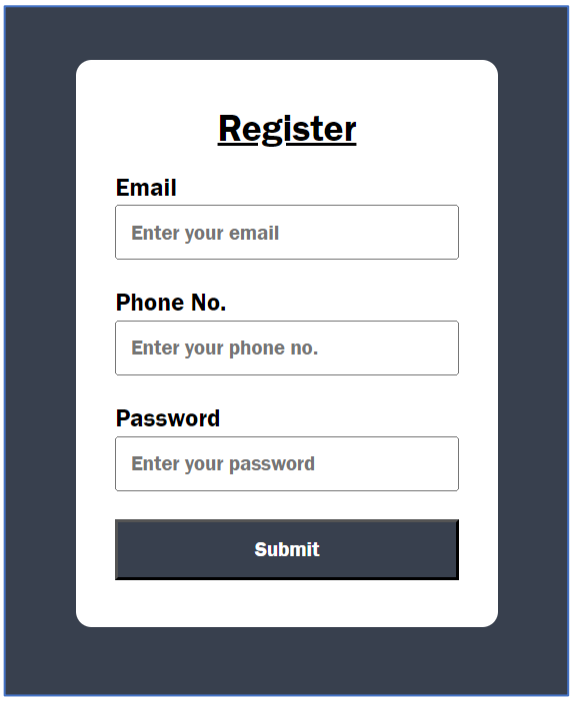
SOURCE CODE:

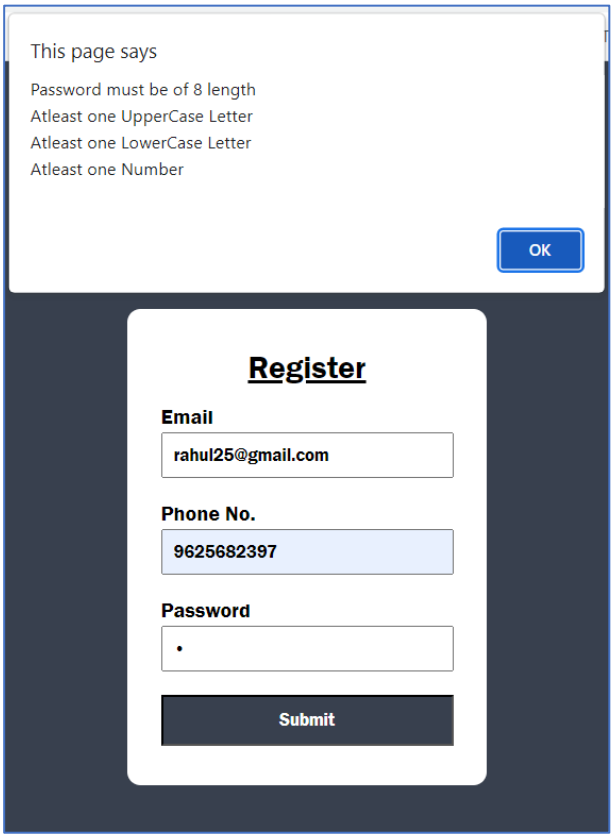
```
<!DOCTYPE html>
<html>
<head>
  <style>
    * {
      box-sizing: border-box; padding: 0; margin: 0;
      font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    }

    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>
```

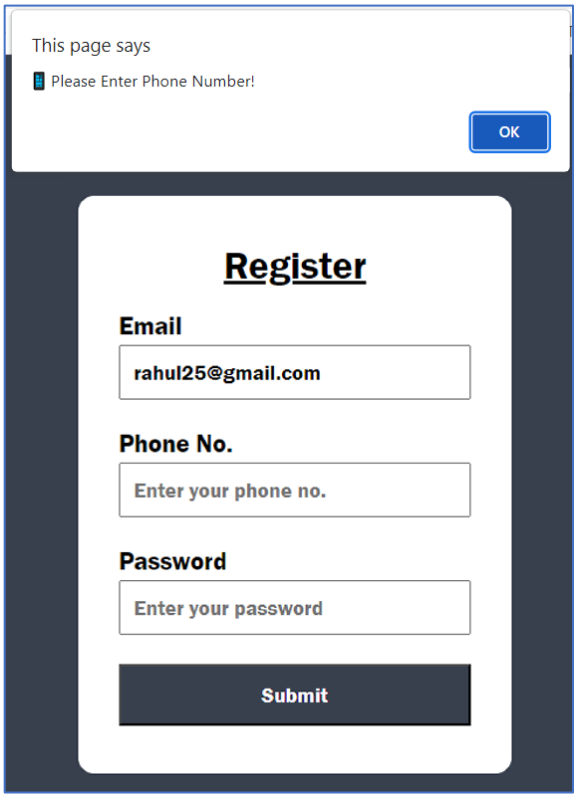
```
</head>
<body>
    <div class="form-div">
        <h2 class="heading">Register</h2>
        <form onsubmit="handleSubmit(event)">
            <label for="email">Email</label> <br>
            <input class="input-field" type="email" name="email" id="email" placeholder="Enter your email">
            <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)";
        }
```

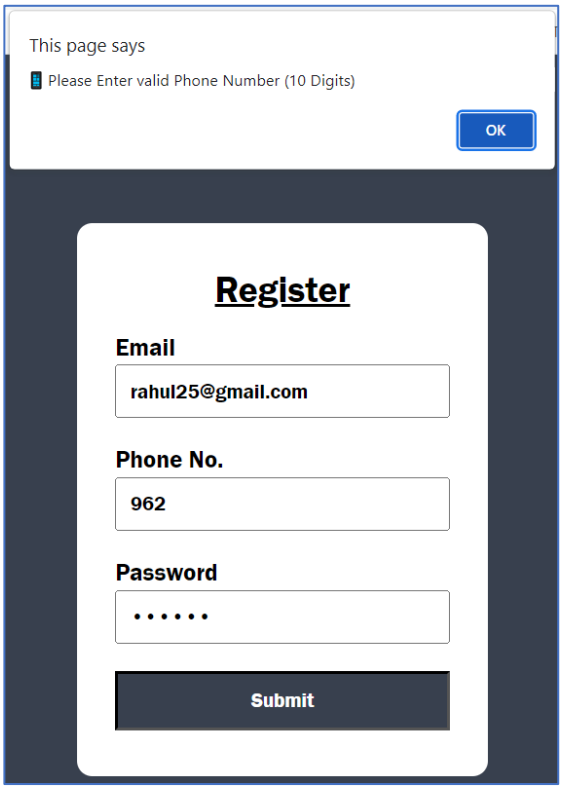
```
    function validatePassword(pass) {
        let allErrors = "";
        if (pass.trim() === "") return "🔴 Please Enter Password \n";
        if (pass.length <= 8) allErrors += "Password must be of 8 length \n";
        if (!(/[A-Z]/.test(pass))) allErrors += "Atleast one UpperCase Letter \n";
        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) {
        //Prevent Page from reloading
        event.preventDefault();

        //Form Validation
        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:


Register
Email
Enter your email
Phone No.
Enter your phone no.
Password
Enter your password
Submit


This page says
Password must be of 8 length
Atleast one UpperCase Letter
Atleast one LowerCase Letter
Atleast one Number
OK
Register
Email
rahul25@gmail.com
Phone No.
9625682397
Password
Submit


This page says
Please Enter Phone Number!
OK
Register
Email
rahul25@gmail.com
Phone No.
Enter your phone no.
Password
Enter your password
Submit


This page says
Please Enter valid Phone Number (10 Digits)
OK
Register
Email
rahul25@gmail.com
Phone No.
962
Password
Submit

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:

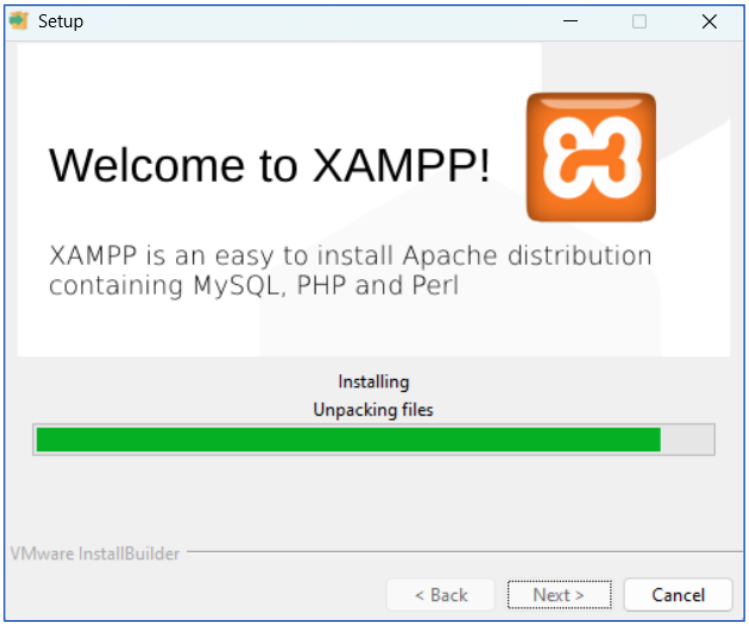
- **Content Management System (CMS)** – A CMS has a central interface that is used to publish, edit, modify, and maintain content. Some well know CMS include, WordPress, Drupal, and Joomla.
- **WordPress** – It is an open-source content management system founded by Matt Mullenweg and Mike Little. It 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:

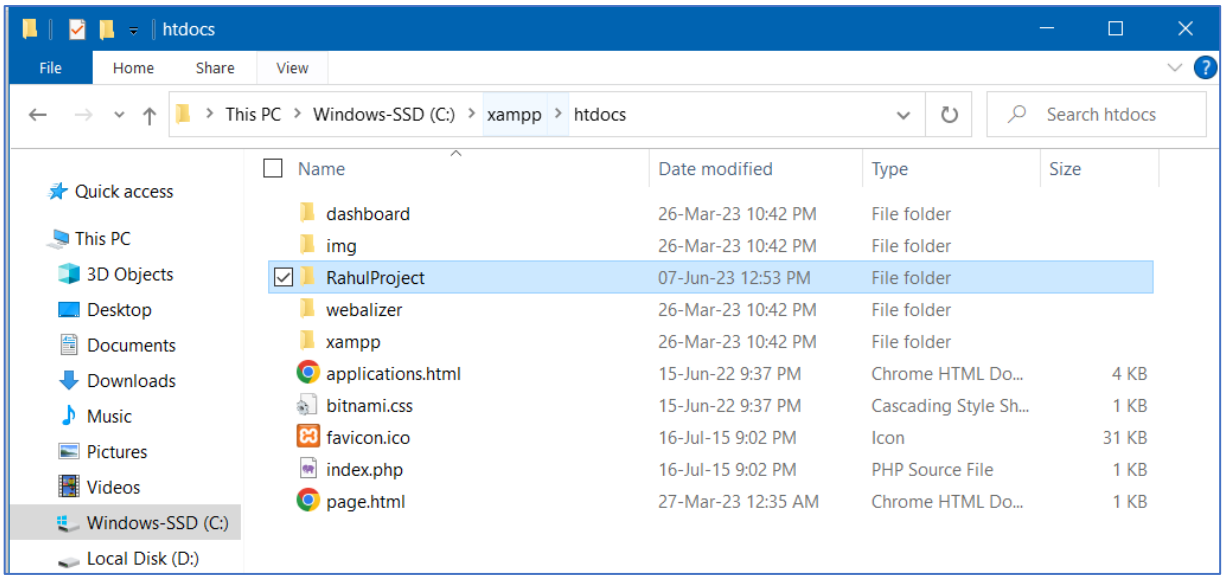
Step 1: Download and Install XAMPP server.

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

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

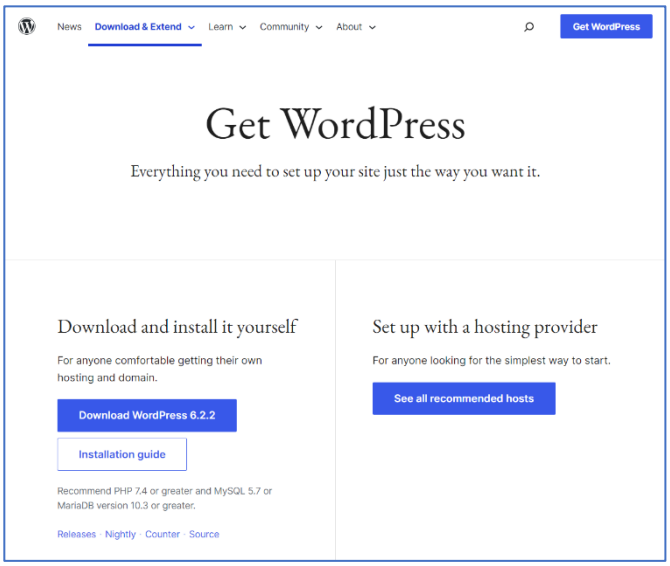


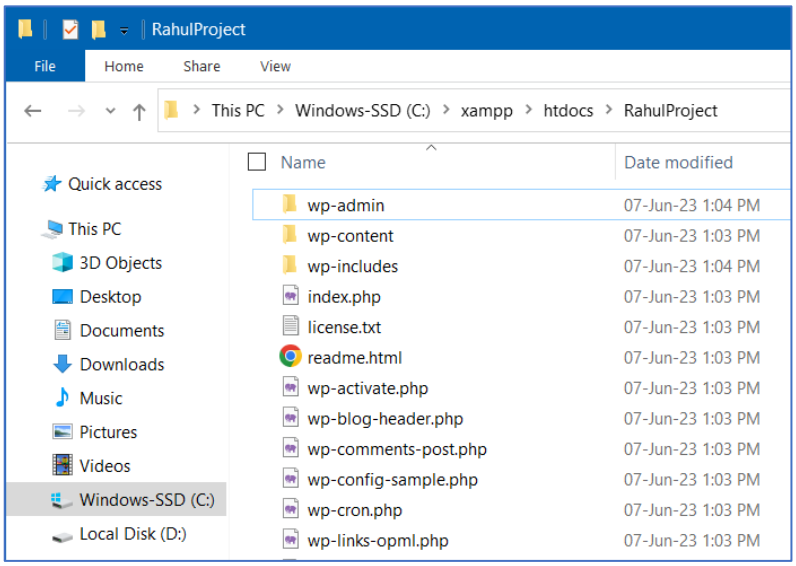
Step 2: Then Open folder PATH **C:\xampp\htdocs** then create your project directory (E.g.: RahulProject).



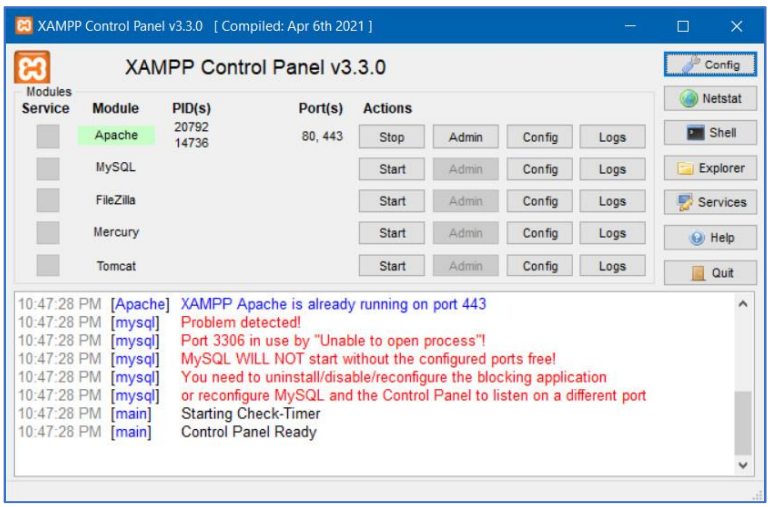
Step 3: Download WordPress from official website.

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



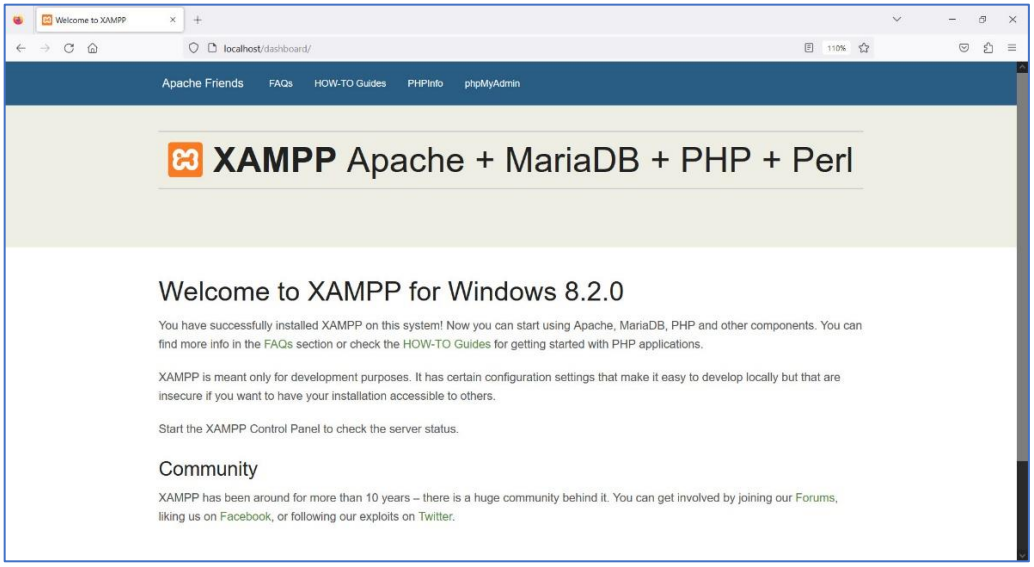


Step 4: Open XAMPP Control Panel.



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**

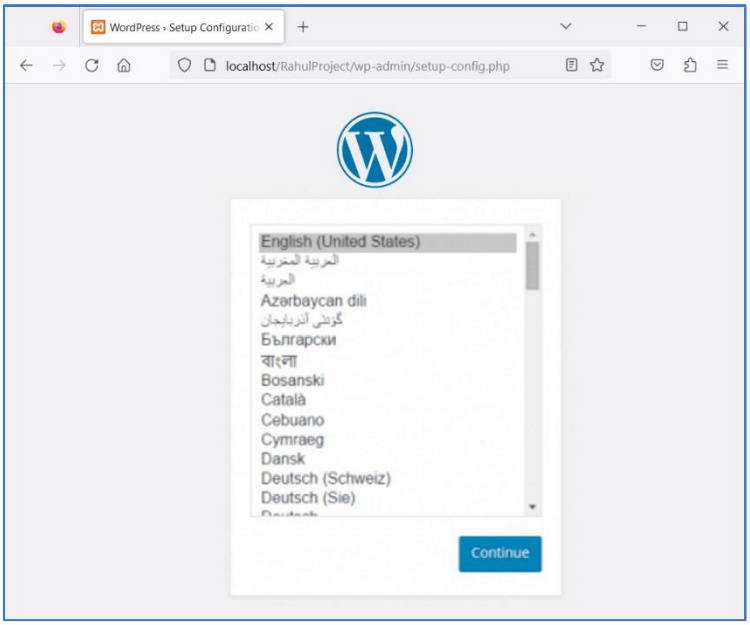
Open the web browser, type **localhost** and press enter. Now we can see the XAMPP dashboard. If the dashboard is visible, it means everything is working fine.

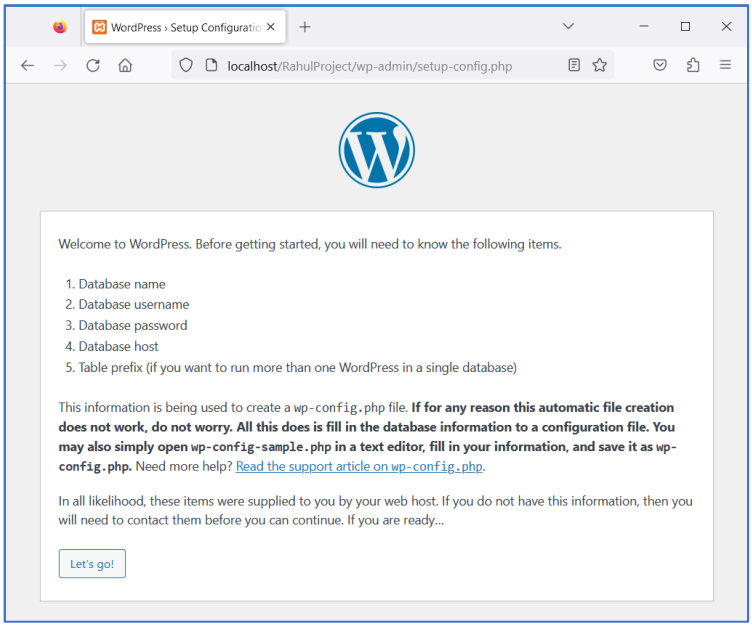


Step 6: Now we will enter the project folder from the browser. For that, type: **localhost/RahulProject/** on the address bar and press enter.

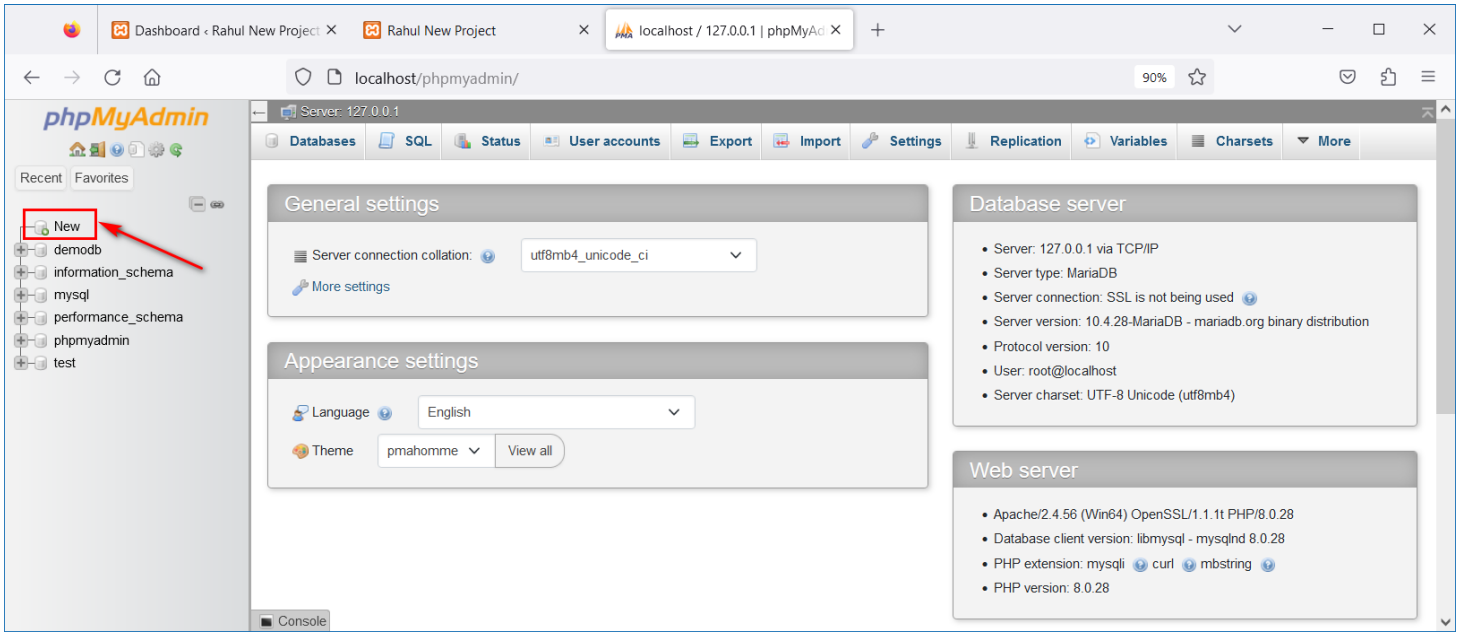
After pressing enter we will be redirected to WordPress installation as in the following screenshot.

- ⇨ Choose Language and Continue.
- ⇨ Click on Let's Go Option





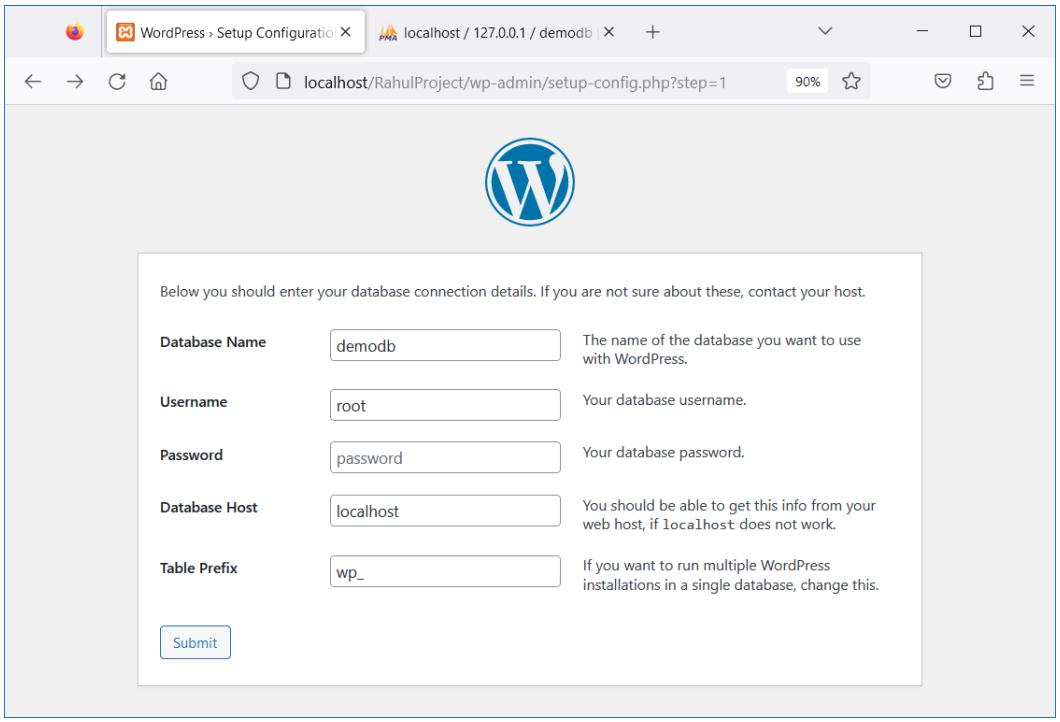
Step 7: Open another browser tab and type **http://localhost/phpmyadmin** and press enter. Now we will reach the PhpMyAdmin page. Create database for WordPress by clicking on New Icon. Enter database name and click create. (Eg: **demodb**)

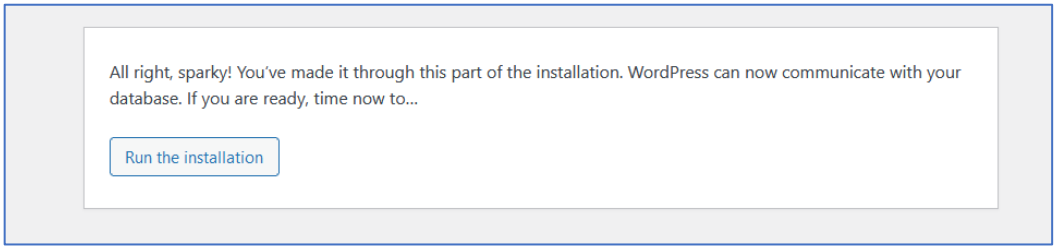


Step 8: Continue to WordPress -> Setup Configuration

- ➔ 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.***

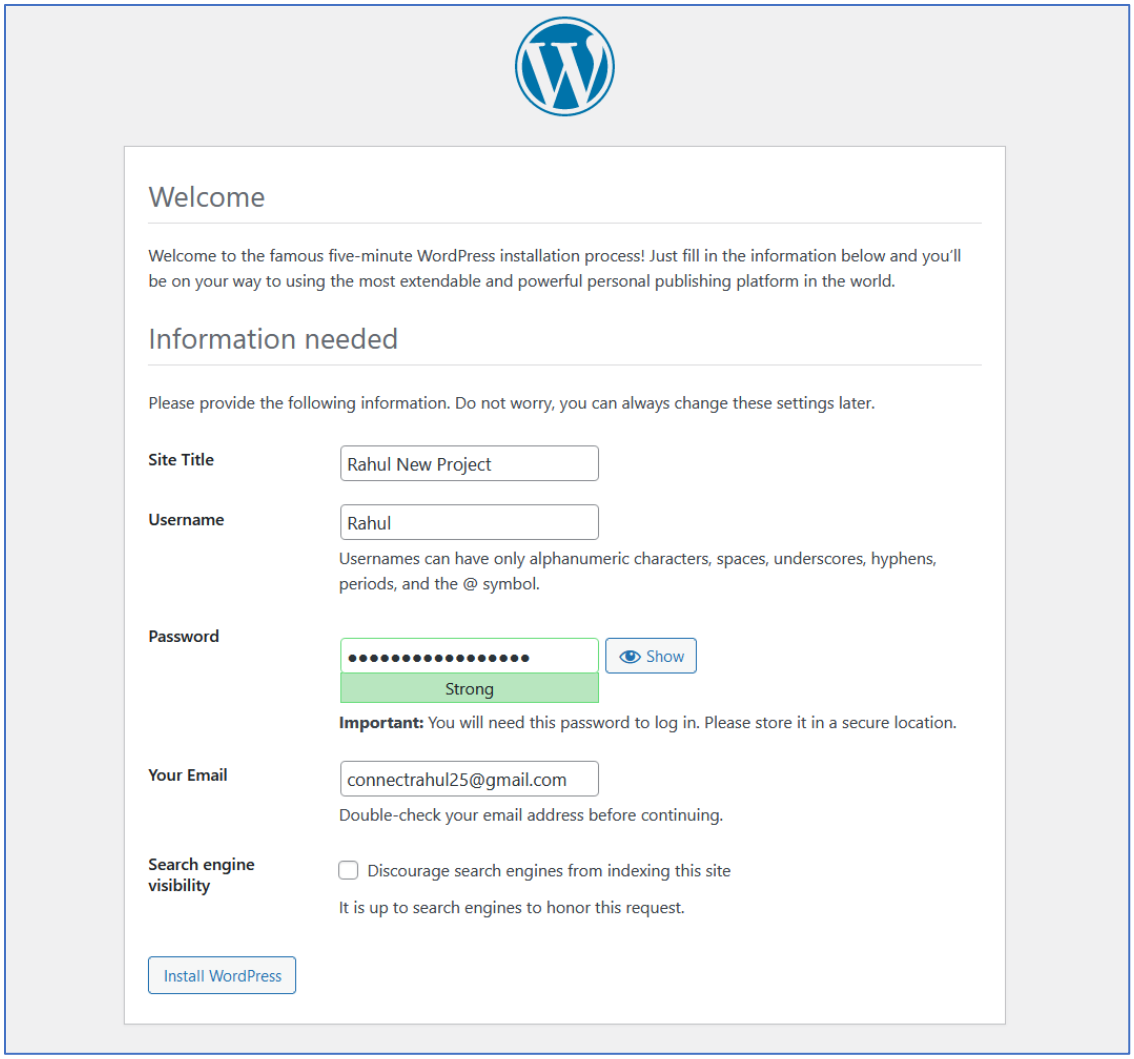




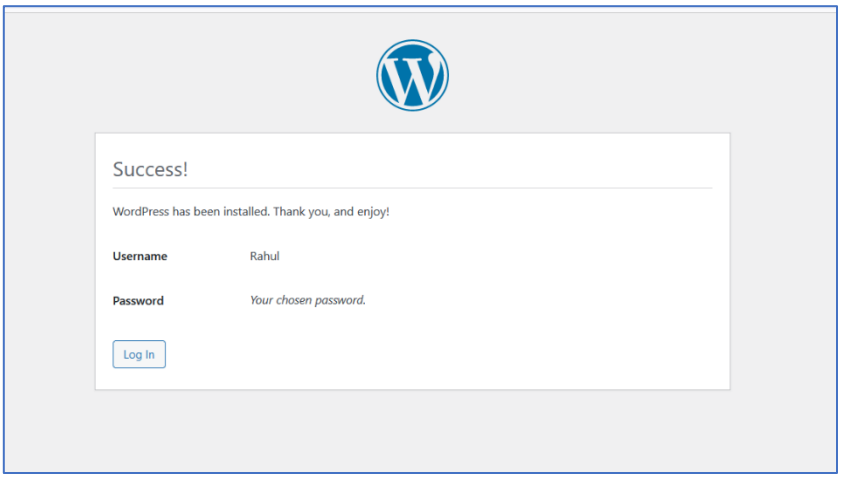
Step 9: Now, get ready to enter the site details. We have added the following, the password is already provided, you can also change it. The Username and Password will help you to login to your own website.

- 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

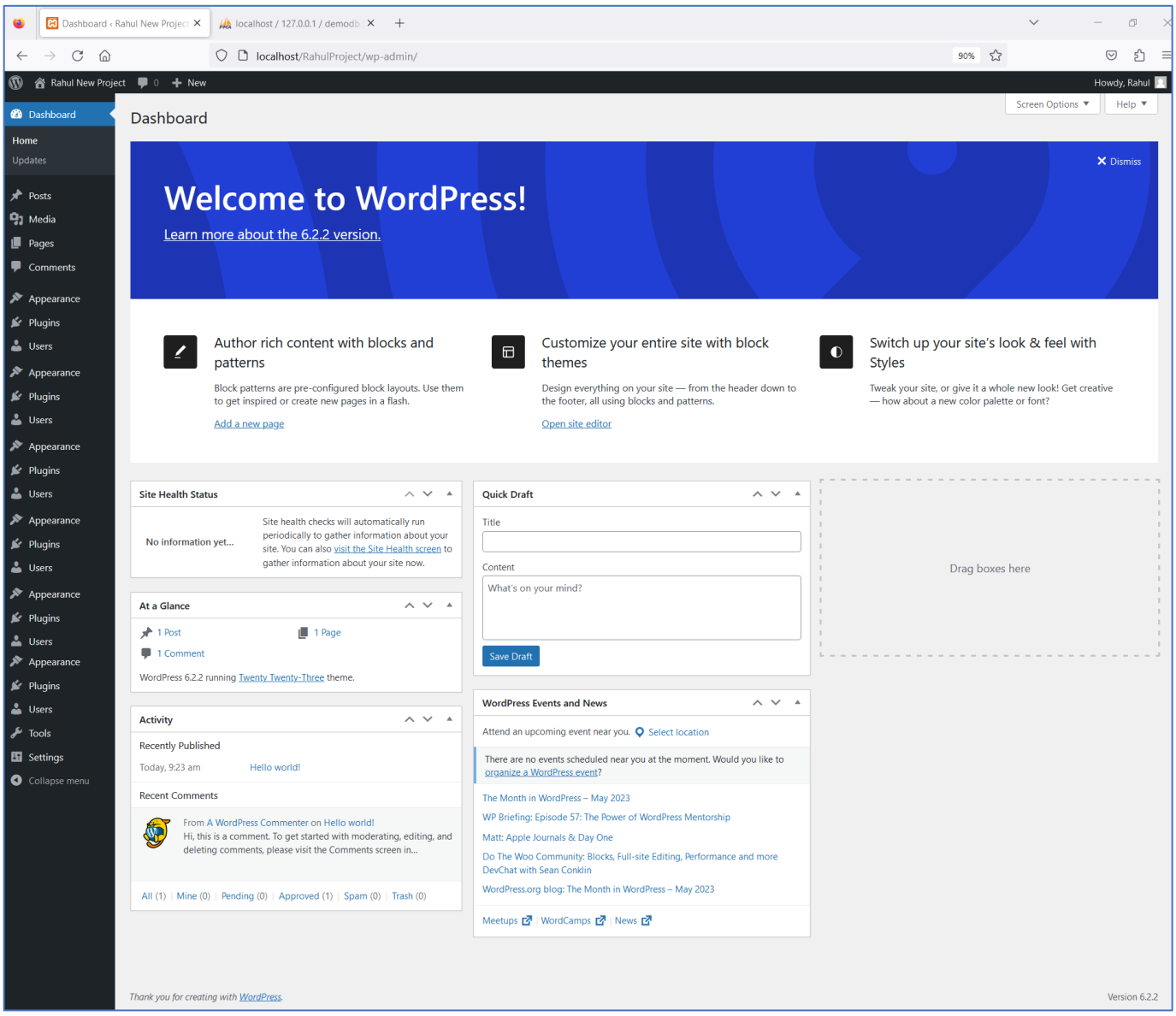
Then, click "**Install WordPress**"



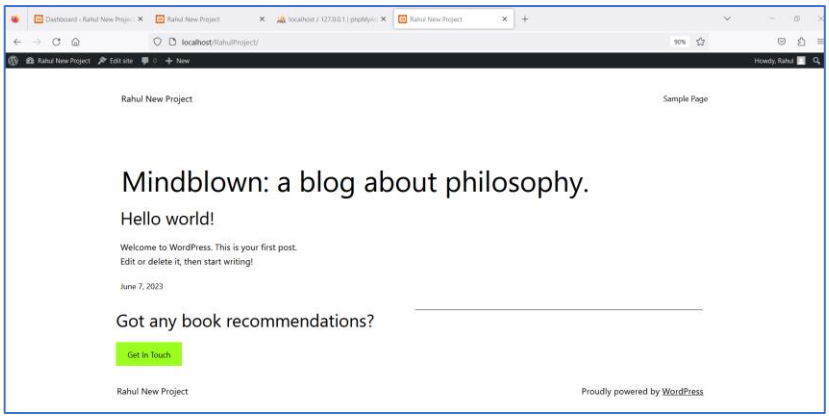
Step 10: Success! Congratulations, successful installation of WordPress. Now, Log In to your WordPress Website



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/RahulProject**) 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:

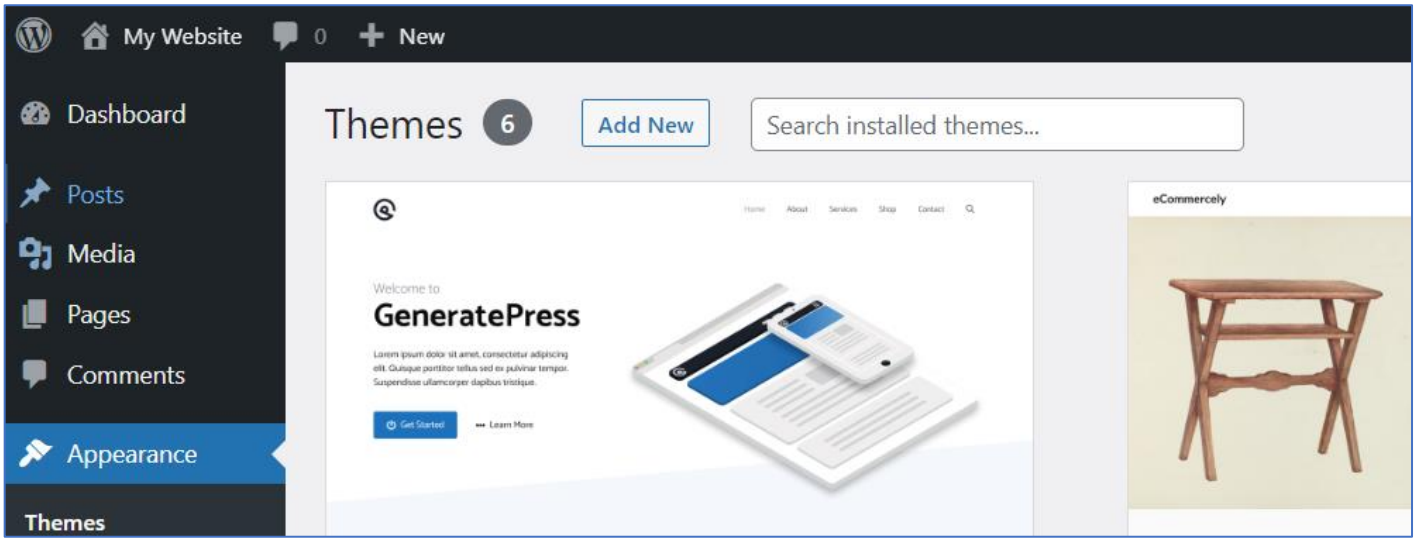
- A visual site map, also known as a website diagram or website structure diagram, is a graphical representation or visual depiction of the pages and hierarchical structure of a website. It illustrates the organization and relationships between different web pages, sections, and content on a website.
- 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:

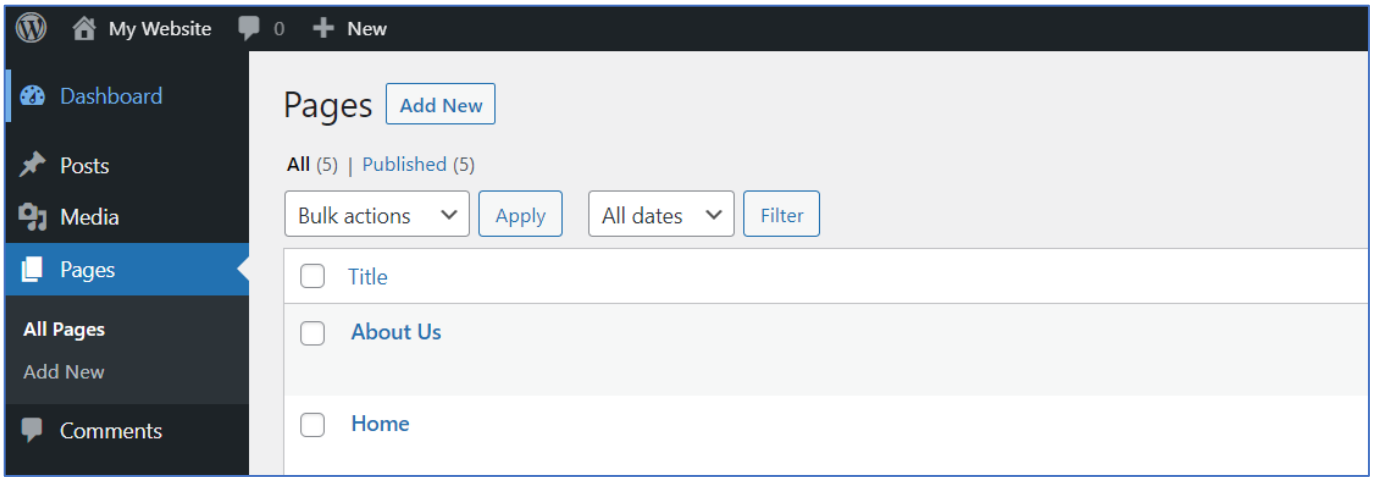
1. Home Page
2. Site Map
3. About Us Page
4. Our Team Page
5. Privacy Policy Page

Steps to Create Visual Site Map of Website:

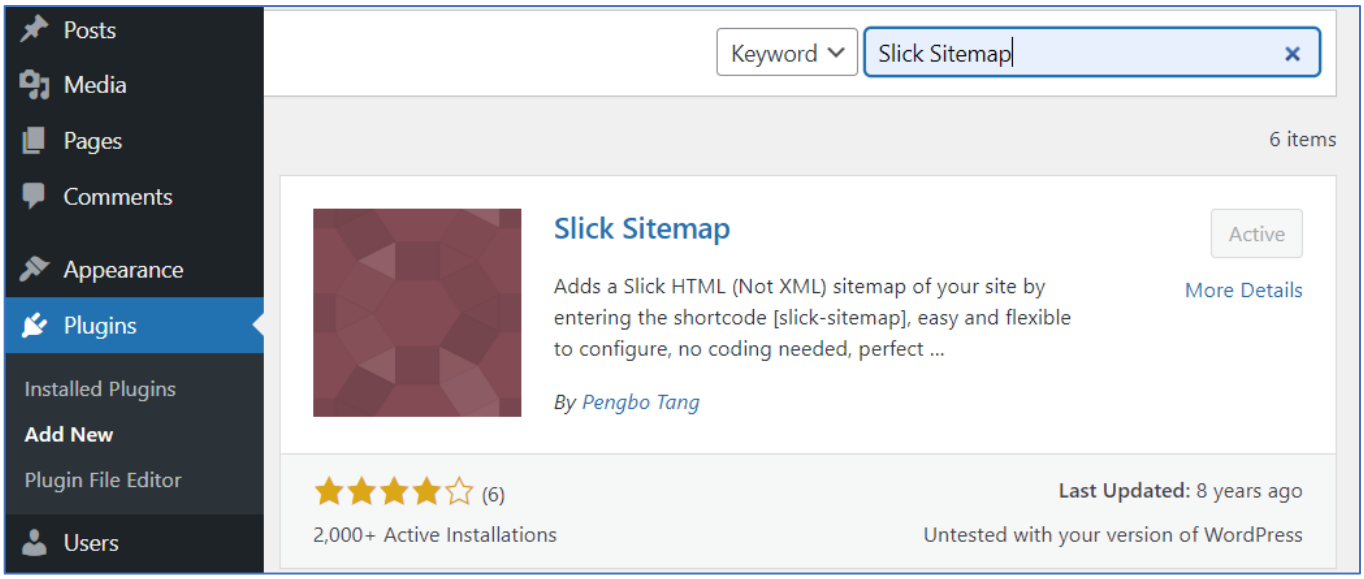
Step 1: Go to **localhost/RahulProject/wp-admin/** and then go to Theme inside Appearance and Select a theme for Website And **Activate** it after Installation.



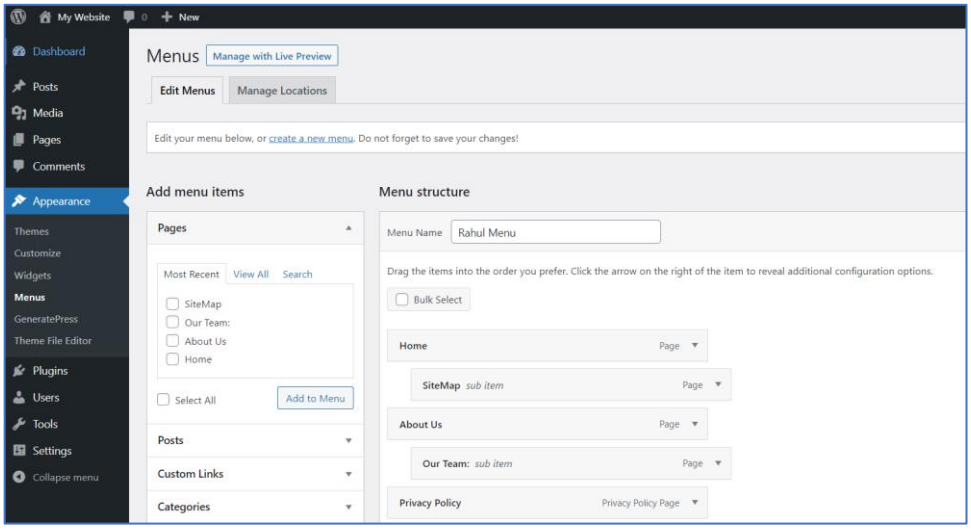
Step 2: Now Click on **Pages - > Add New** then Create As many pages a wanted in the Website.



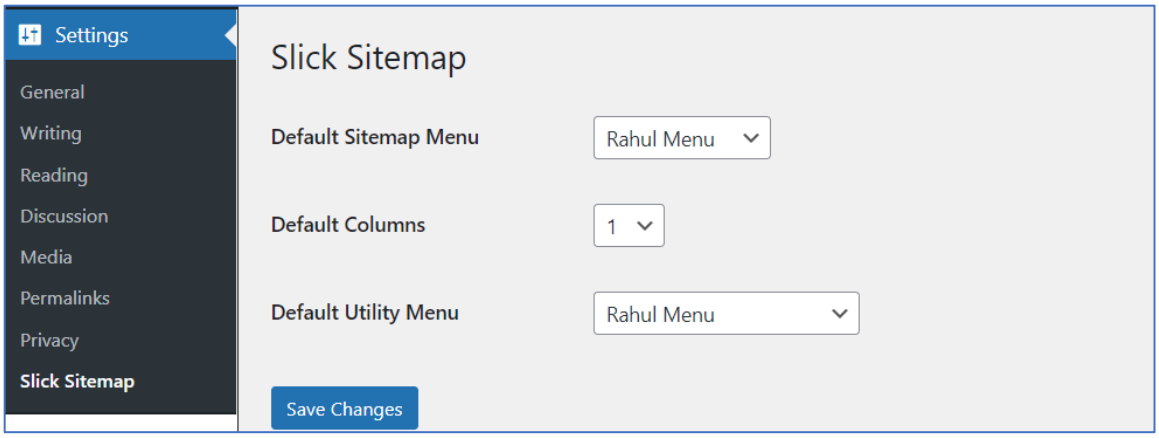
Step 3: Now Go to Plugins - > Add New and Install Slick Sitemap and then activate it.



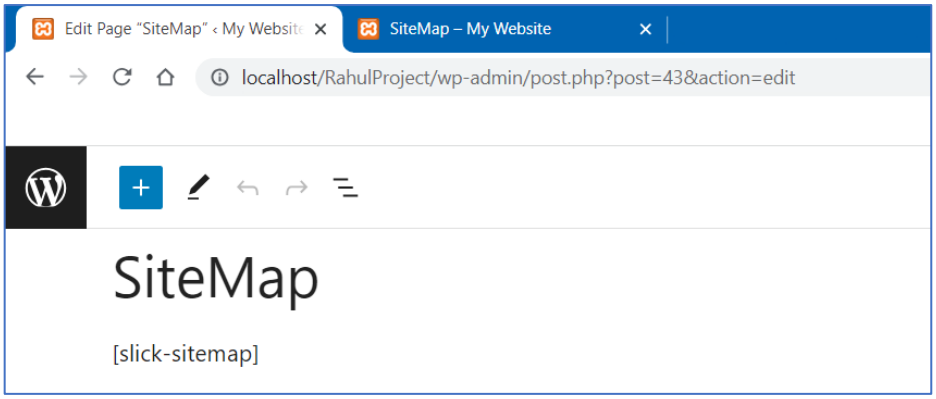
Step 4: Go to Appearance - > Menus and Set the Pages And set your menu as primary menu you so it can arrange as you want them in your Website.



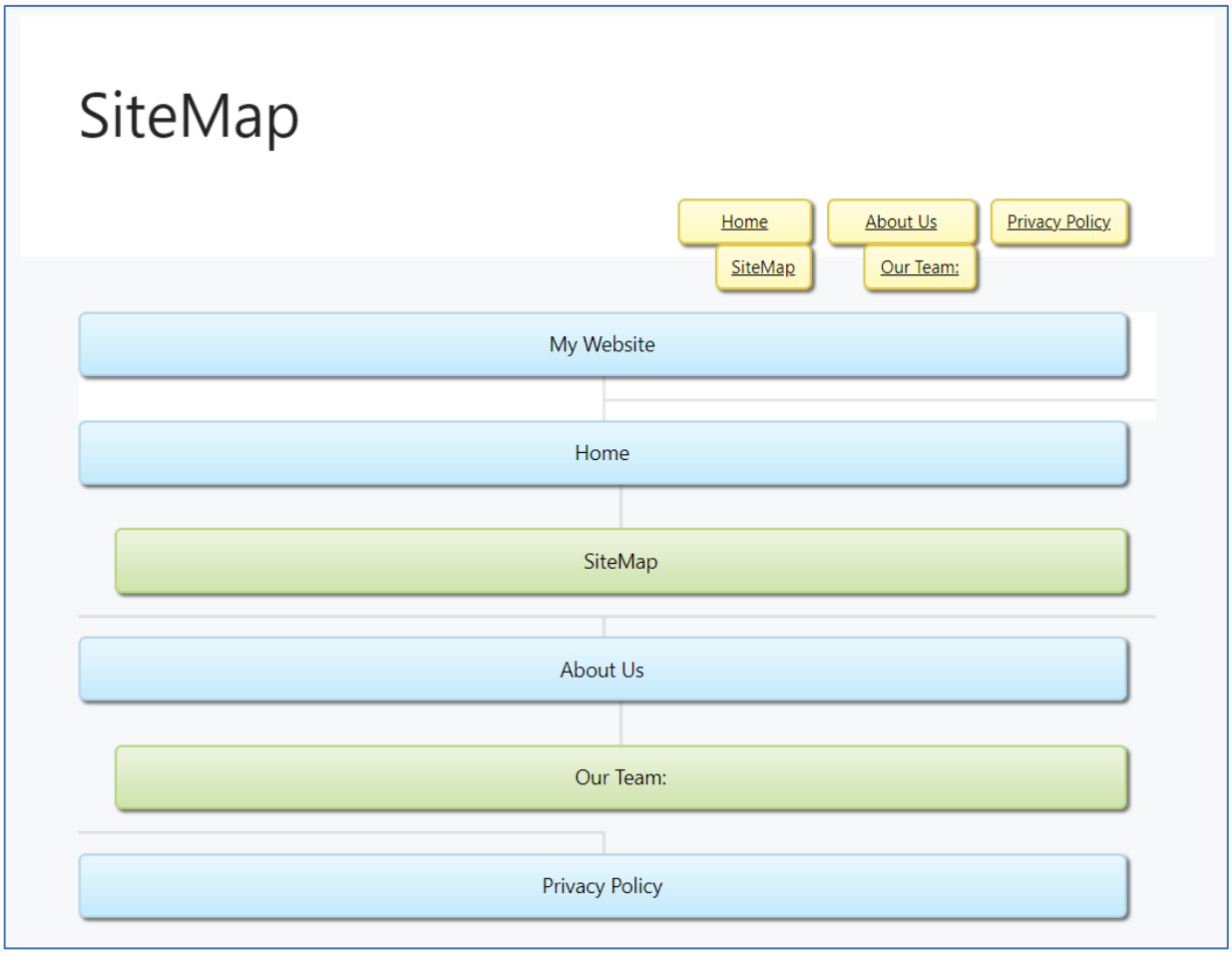
Step 5: Go to **Settings - > Slick-Sitemap** and Choose the Default Sitemap and utility menu by selecting the menu you created in previous step and also set column to two.



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.

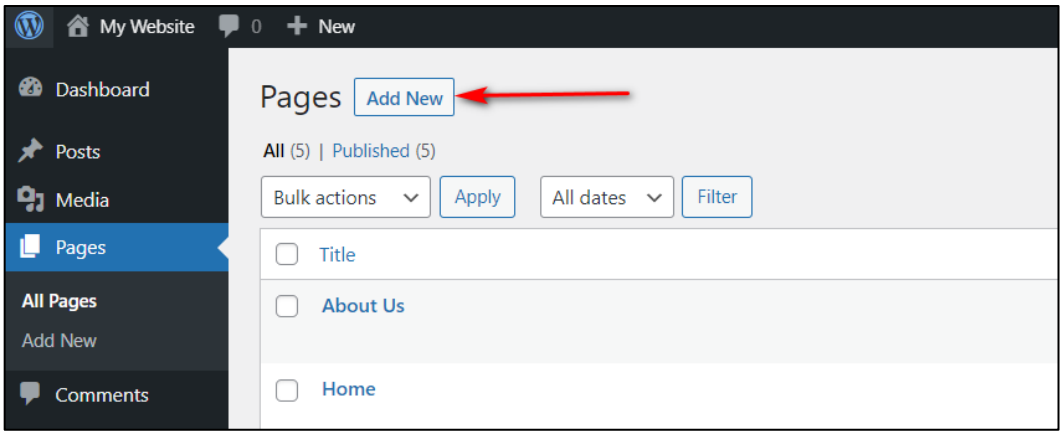
THEORY:

Content of My Website:

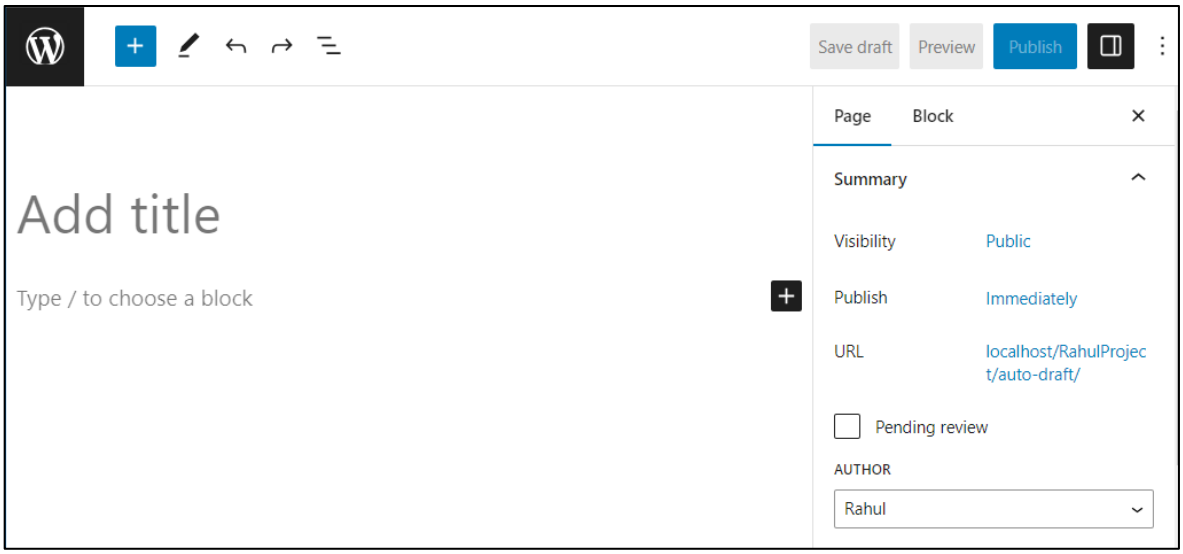
- **Home Page**:
- **Visual Site Map Page**:
- **About Us Page**:
- **Out Team**:
- **Privacy & Policy Page**

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.

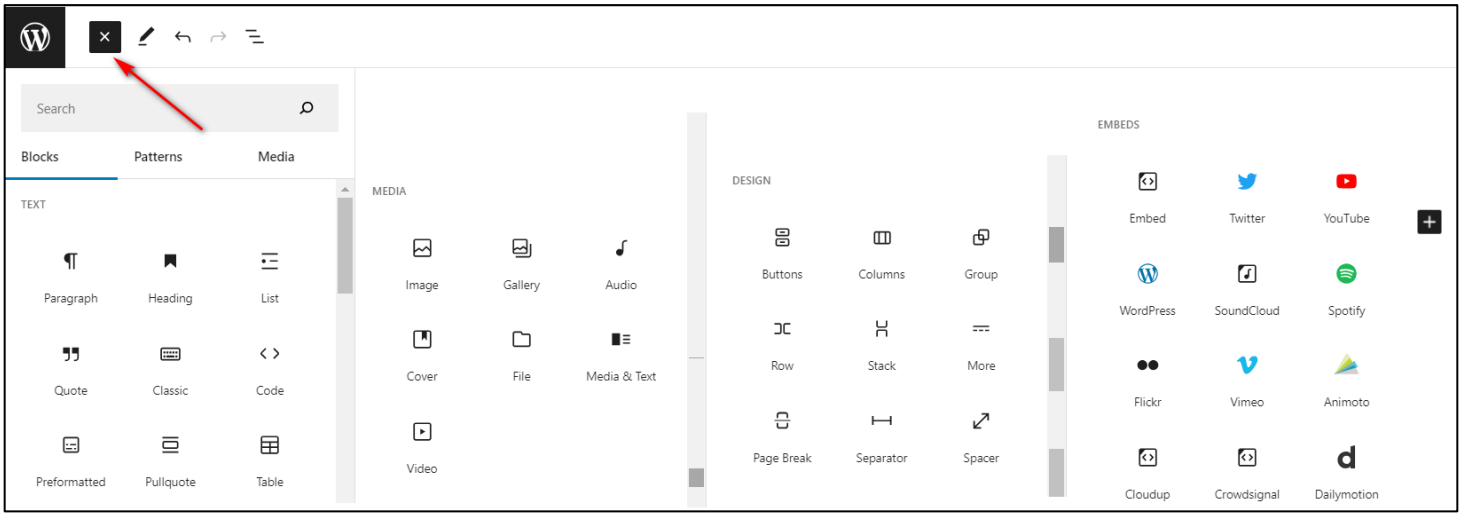


Step2: Add the title of the page, like *About*. Click the Add Title text to open the text box where you will add your title.

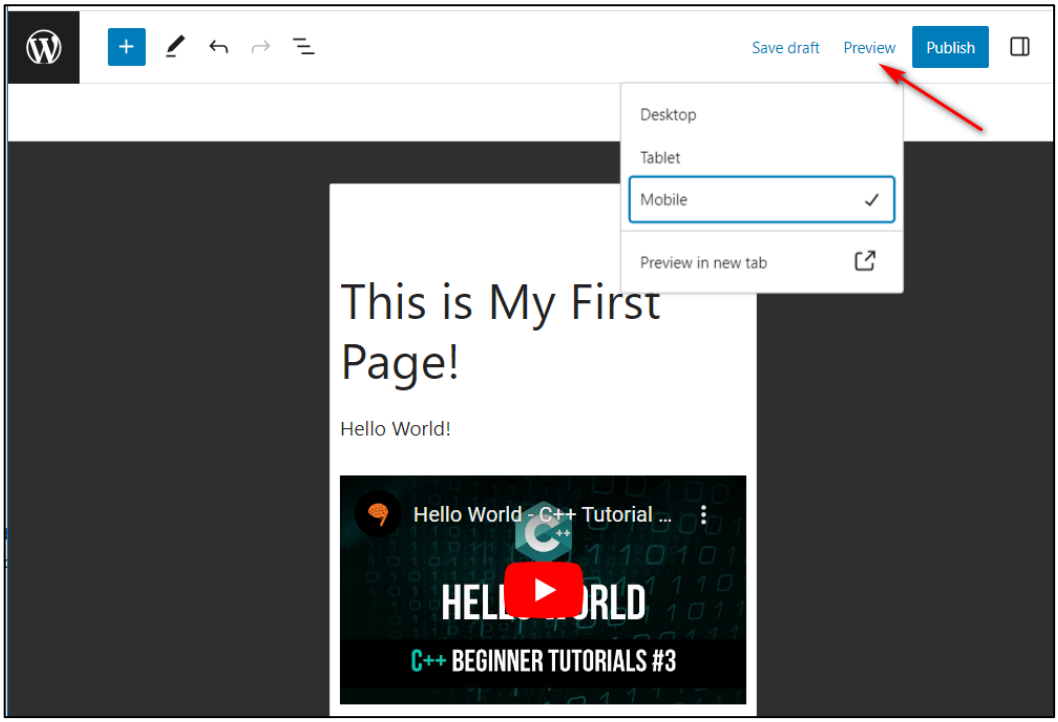


Step 3: Add some content. Content can be anything you choose … from text, headings, images, lists, videos, tables, and lots more.

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

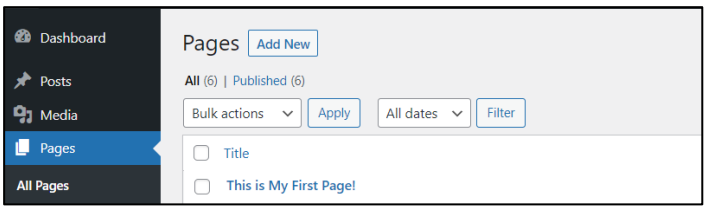


Step 4: Click on Preview Button to check how your website looks on different devices like: Desktop, tablet and mobile.



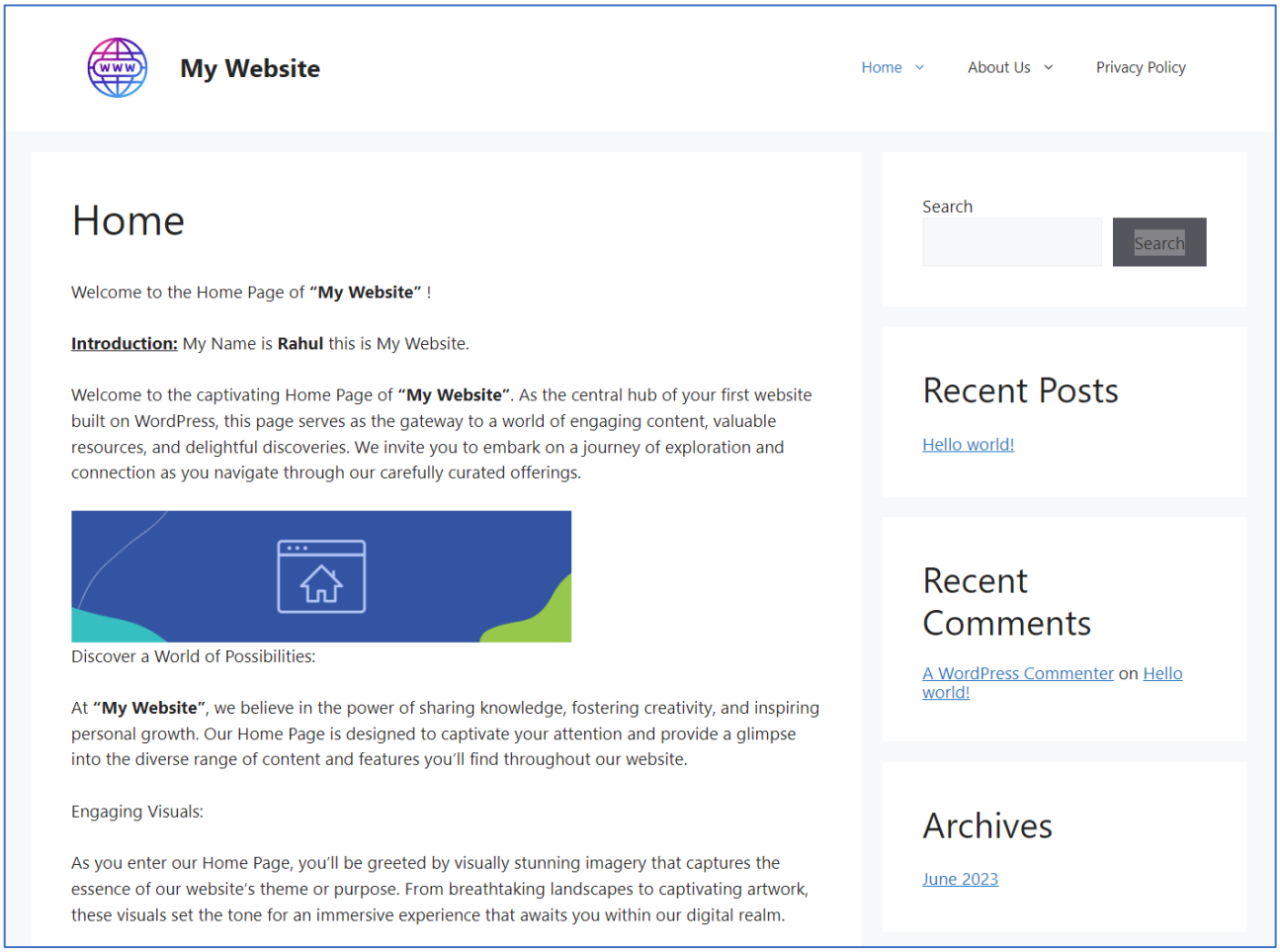
Step 5: Click on Publish Button to save and publish your page!

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

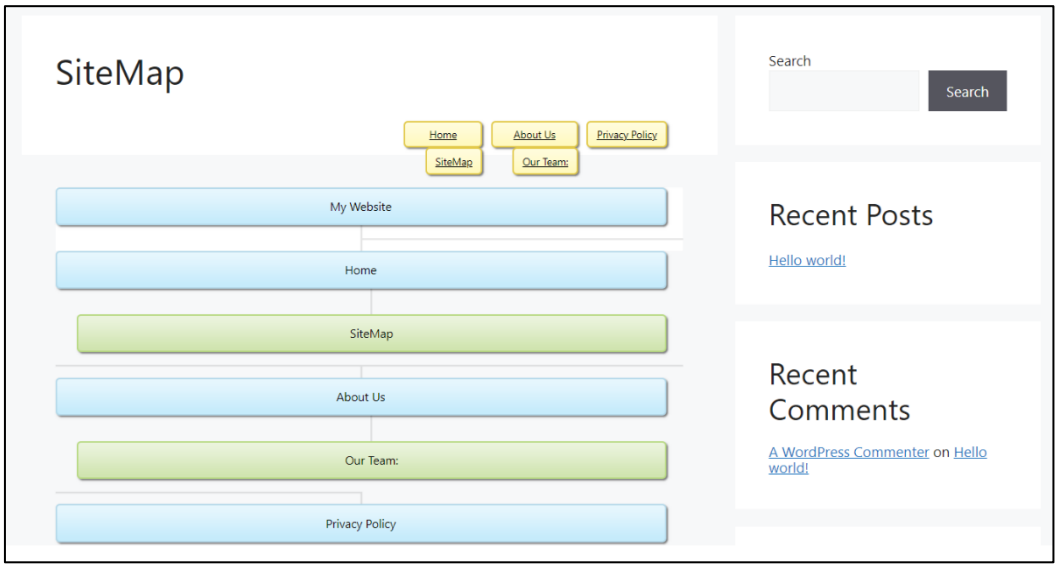


OUPUT:

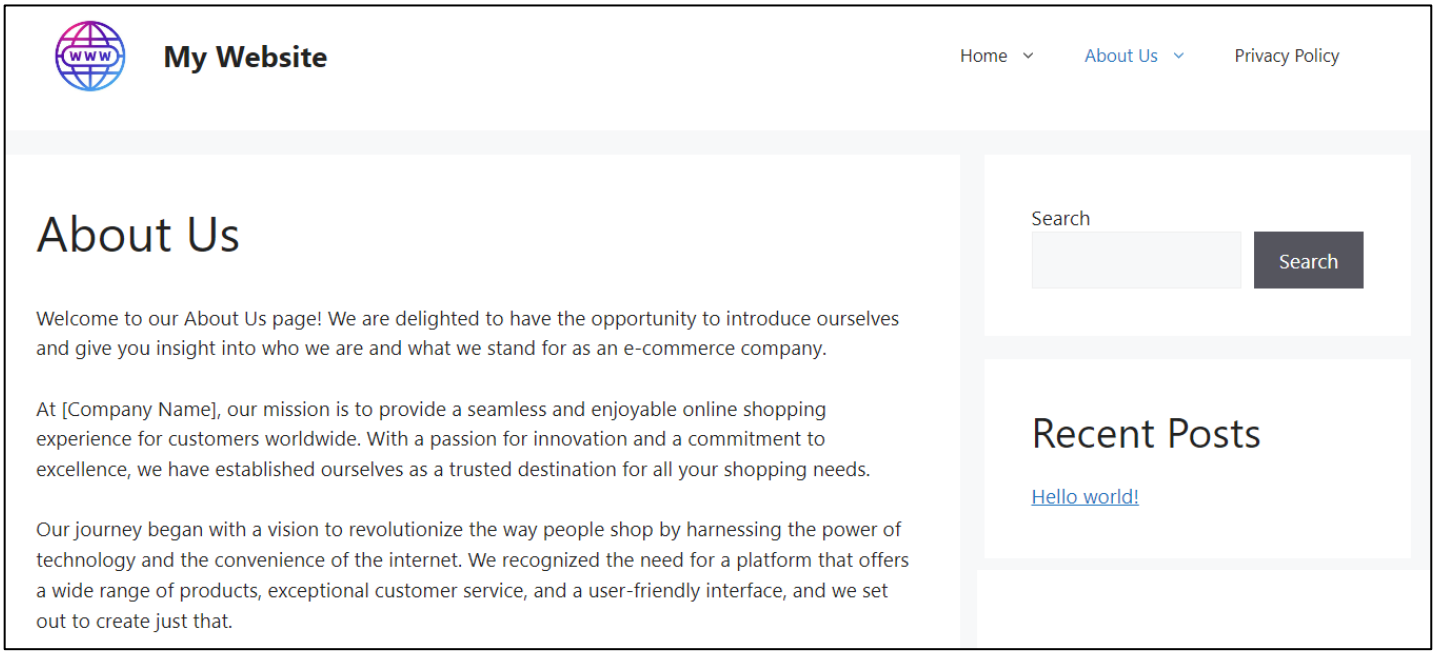
HOME PAGE:



VISUAL SITE MAP PAGE:



ABOUT US PAGE:



PRIVACY & POLICY PAGE