

WEB TECHNOLOGY LAB

(ETCS – 356)

Faculty Name: Mr. Ajay Kr. Tiwari

(Assistant Professor)

Student Name: Prince

Roll Number: 10314802720

Group: 6C6



MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

Maharaja Agrasen Chowk Sector-22 Rohini, New Delhi - 110086

WEB TECHNOLOGY LAB
(ETCS – 356)

LAB ASSESSMENT SHEET

Student Name: Prince

Roll Number: 10314802720

[illegible]

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 :

- i) Apache
- ii) XAMPP
- iii) WAMP
- iv) GWS
- v) 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**

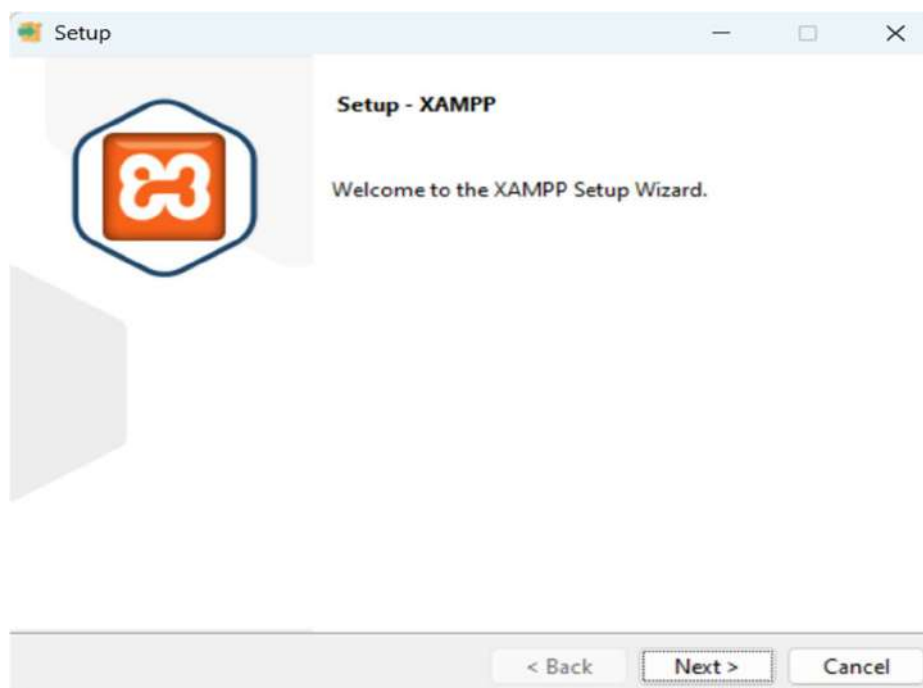
- It is easy to set up and use.
- XAMPP is a free and open-source cross-platform web server solution stack package for all types of operating systems like Linux and Windows.
- It has many other essential modules like phpMyAdmin, OpenSSL, MediaWiki, WordPress, Joomla, and more.
- It allows the users to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

Steps To Install XAMPP:

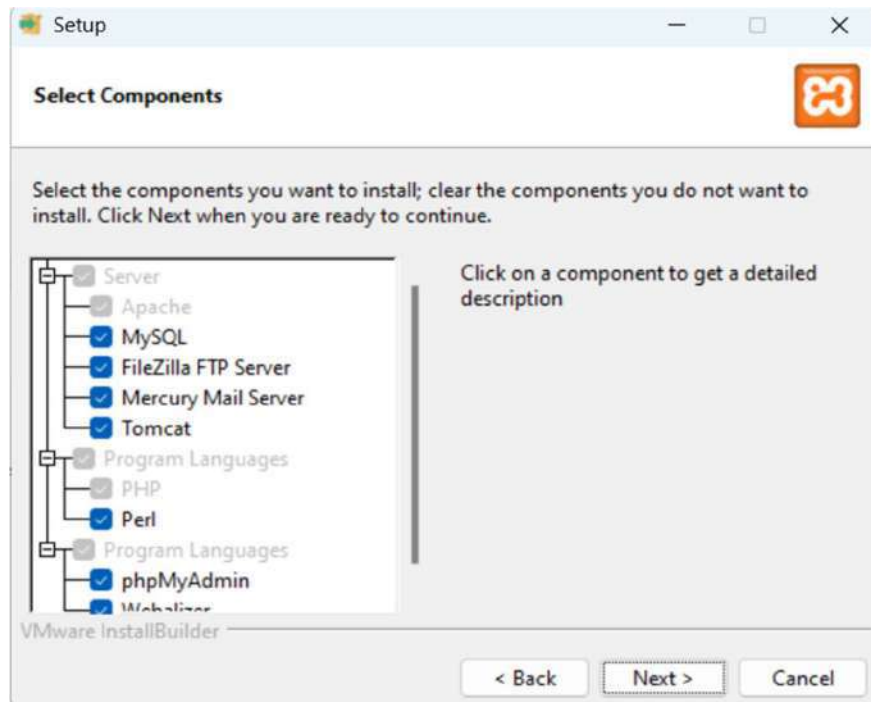
1. Go to Apachefriends.org and click on latest XAMPP version to Install.



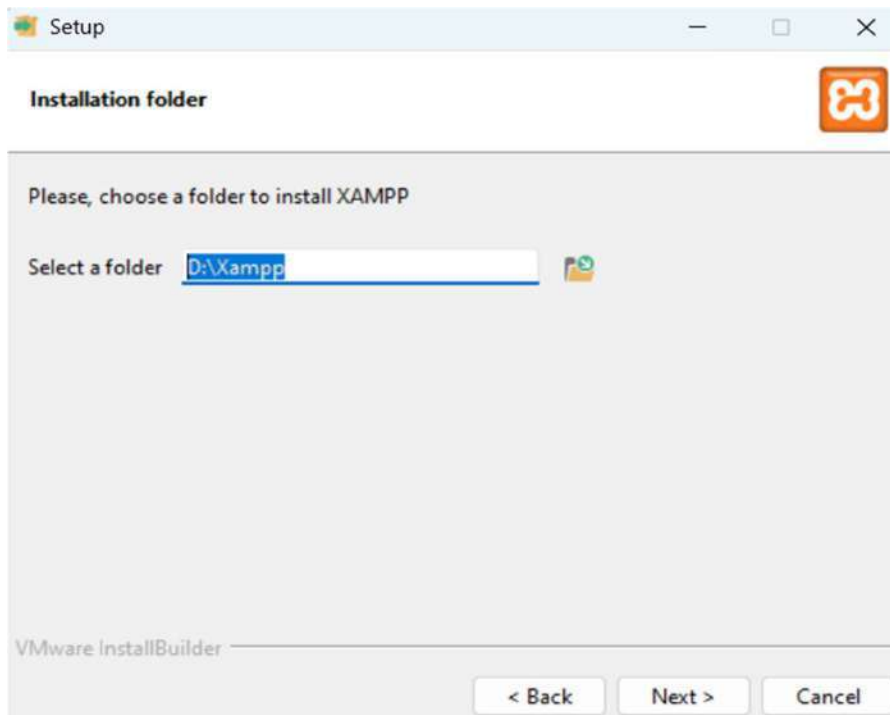
2. After Installation is Complete go to the specified location where XAMPP setup file is installed and double click on the file to install It.



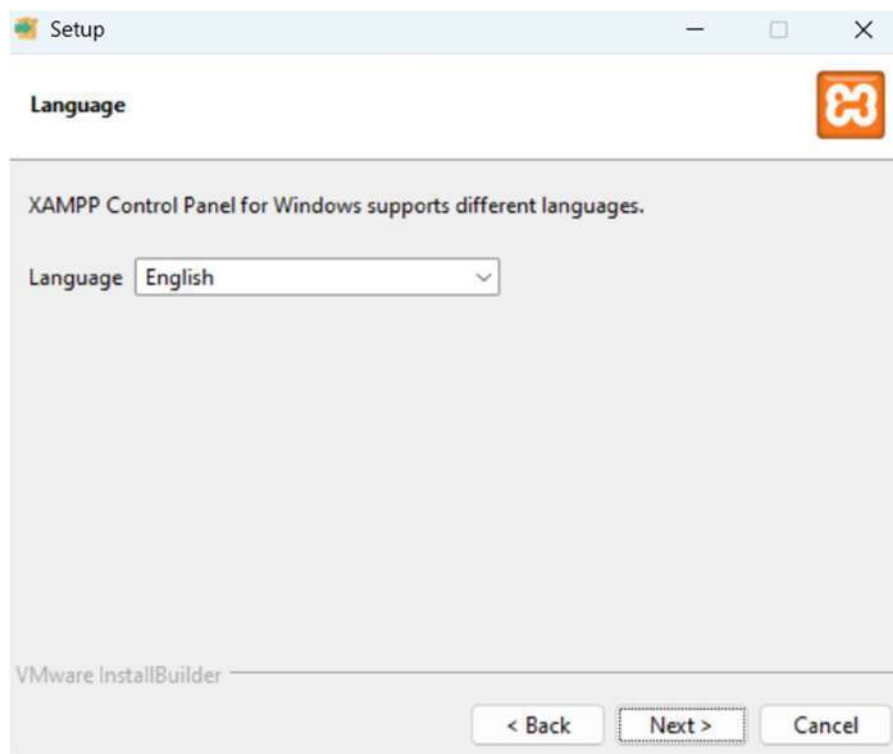
3. Click on **Next**.



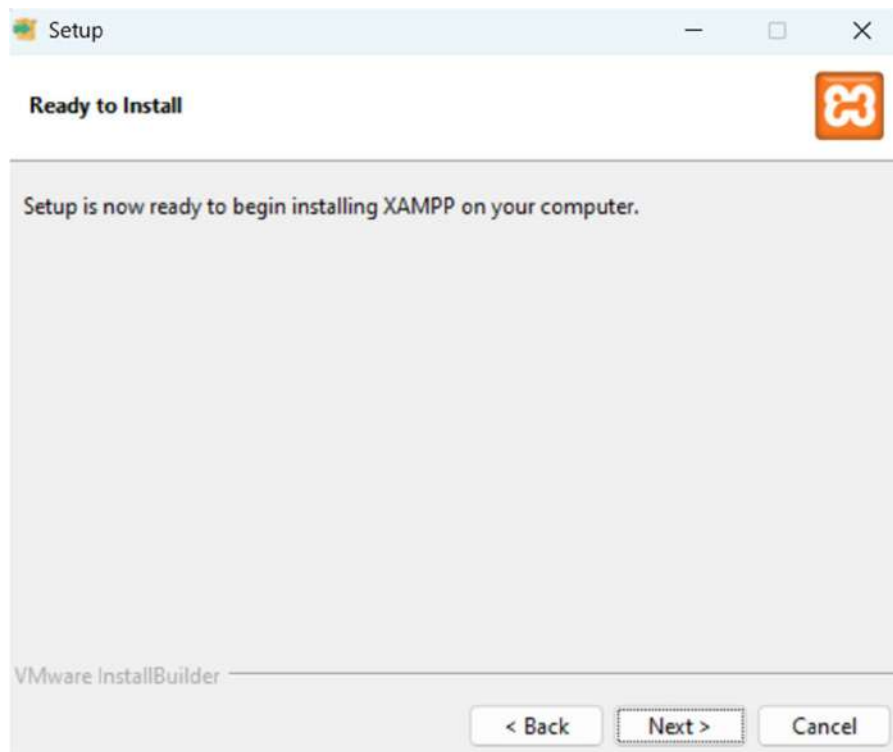
4. Click on **Next** and Choose specific location where you want to Install XAMPP files.



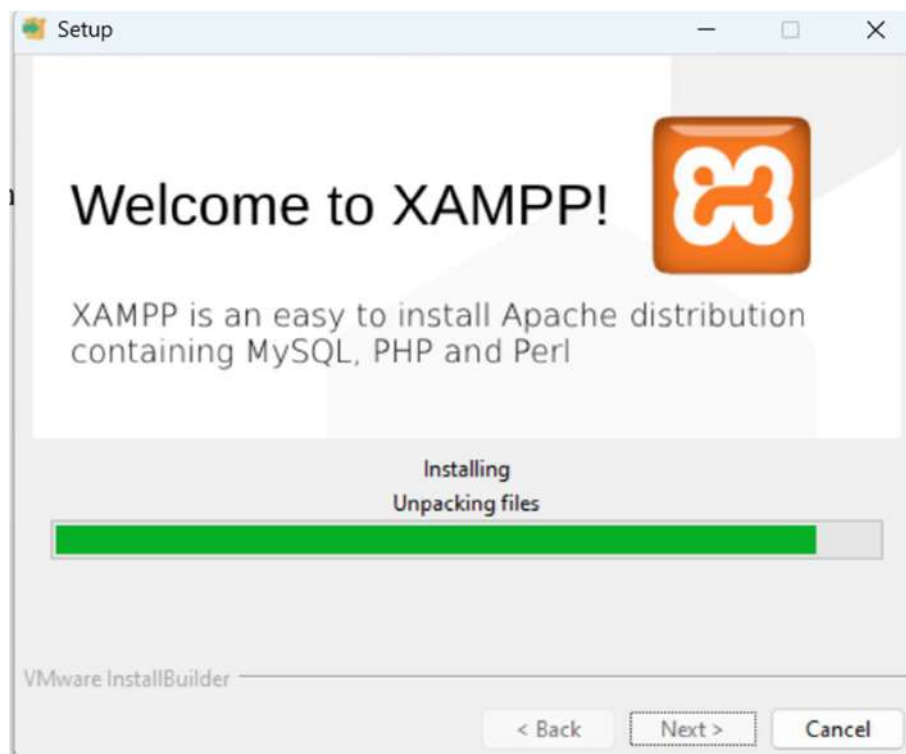
5. Click on **Next** and Choose the Language.



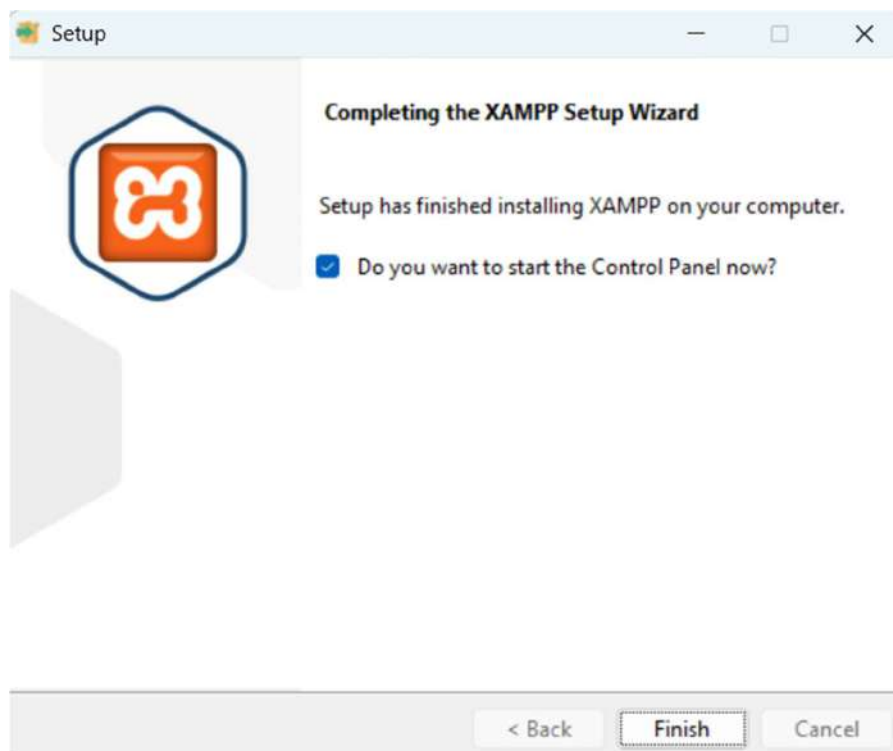
6. Click on **Next**.



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

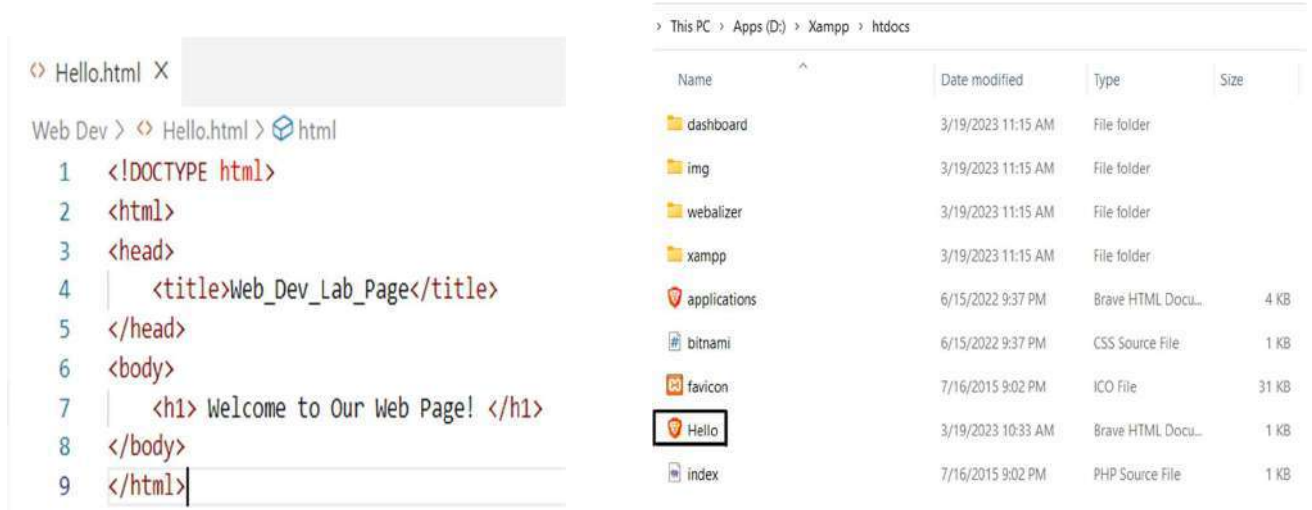


8. After Installation is Complete Click on **Finish**.

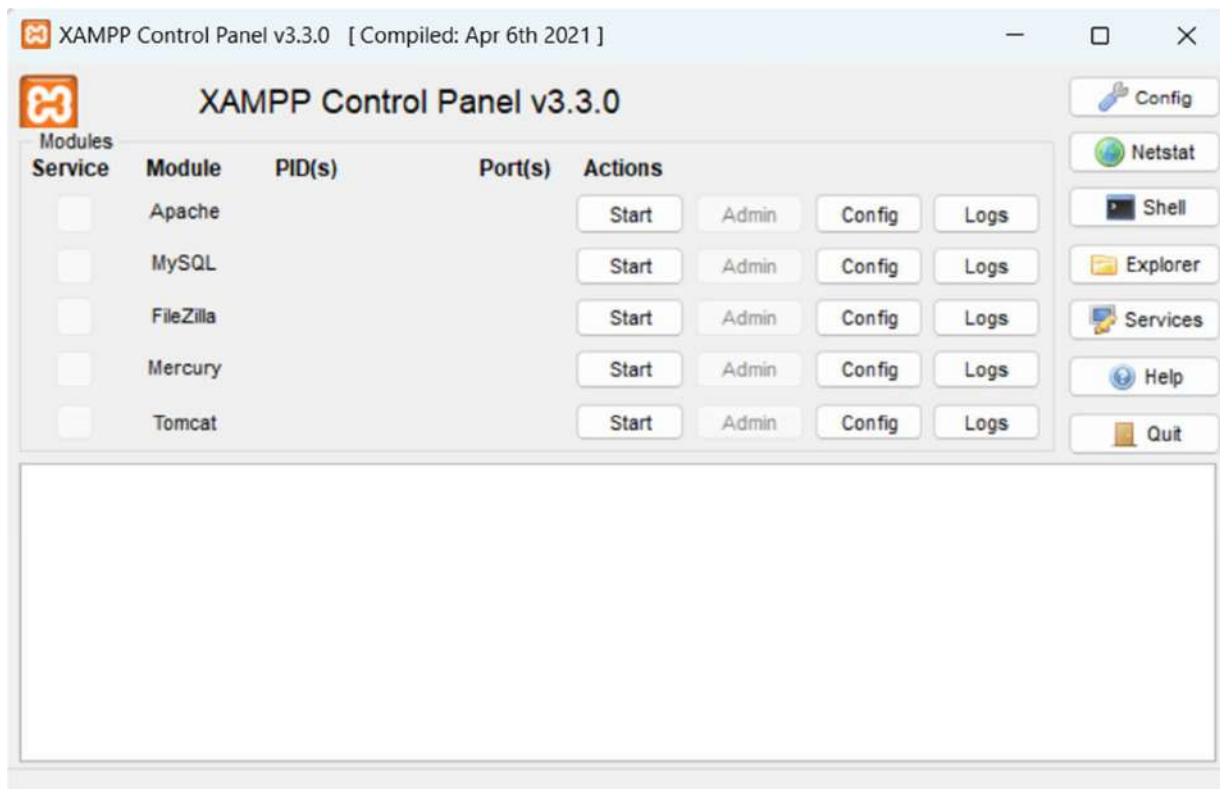


Steps to Open Html file in XAMPP :

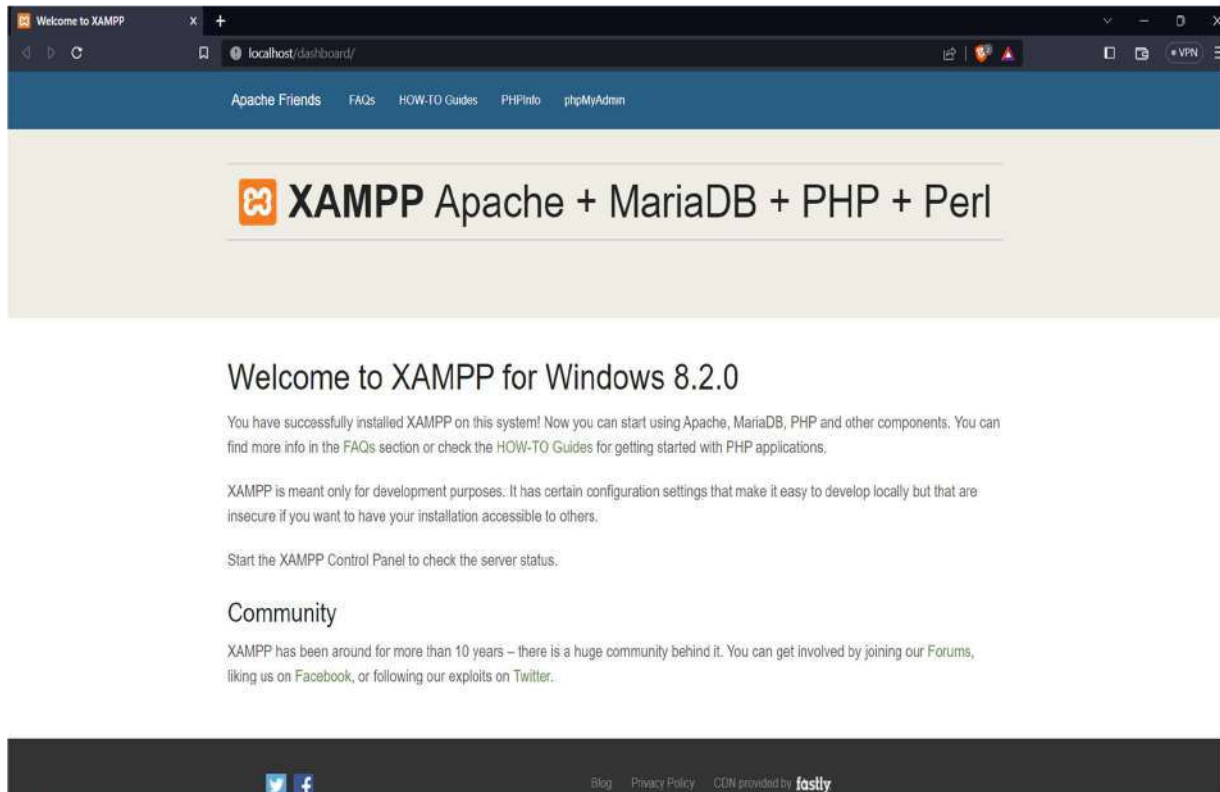
1. First create a Html file and then paste it into the **htdocs** folder inside the folder where you install the XAMPP.



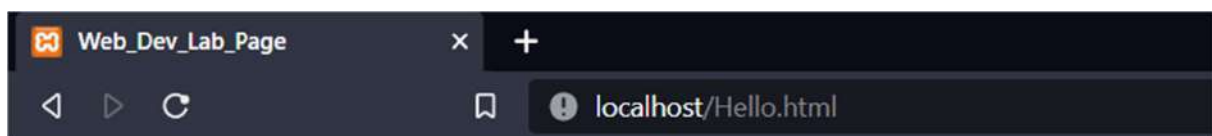
2. Now Open **XAMPP Control Panel**.



3. Now Click on **Start** button In front of Apache and then Click on **Admin**. Then a XAMPP Web page is open With localhost/dashboard.



4. 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.



Welcome to Our Web Page!

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:

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">
<p>	Defines a paragraph of text.	None	<p>This is a paragraph of text.</p>
<h1> to <h6>	Defines headings of varying sizes.	None	<h1>This is a level 1 heading.</h1>

<div>	Defines a division or section of an HTML document.	class: specifies the CSS class.	<div class="container">Content goes here.</div>
	Defines bold text.	None	This text is bold.
<i>	Defines italicized text.	None	<i>This text is italicized.</i>
<u>	Defines underlined text.	None	<u>This text is underlined.</u>
 	Defines a line break or carriage return in a block of text.	None	<p>This is a line of text. This is the next line of text.</p>
<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 1Item 2
	Defines an ordered list.	start: specifies the starting number for the list. type: specifies the type of list.	<ol start="3">Item 1Item 2
	Defines a list item.	None	Item 1Item 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>

CODE:

```
<!DOCTYPE html>

<html>

<head>

    <title>All Tag page</title>

</head>

<body>


    <CENTER>

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

    </CENTER>


    <p>Hello, I am <i><b>Prince Bansal</b></i>, this is 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 St. Kabir Modern School.</dd>

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

        <dt><h3>HOBBIES</h3></dt>

        <dd>I like to play and make different story type Games.</dd>

        <dd>I also like to watch movies and listen music.</dd>

        <dd>I also like to do a little bit cooking</dd>

        <dt><h3>ADDRESS</h3></dt>
```

<CENTER>

<address>

My Name: Prince Bansal

University Roll No: 10314802720

Address: New Grain Market

Shop No. 108

Pin code: 132114

Gharaunda (Karnal)

</address>

</CENTER>

</dl>

<hr></hr>

<CENTER>

<h2> Heading Tags </h2>

</CENTER>

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

<h3><u>Heading 3 : </u></h3> <p>This is Heading 3 Tag </p>

<h4><u>Heading 4 : </u></h3> <p>This is Heading 4 Tag </p>

<h5><u>Heading 5 : </u></h3> <p>This is Heading 5 Tag </p>

<h6><u>Heading 6 : </u></h3> <p>This is Heading 6 Tag </p>

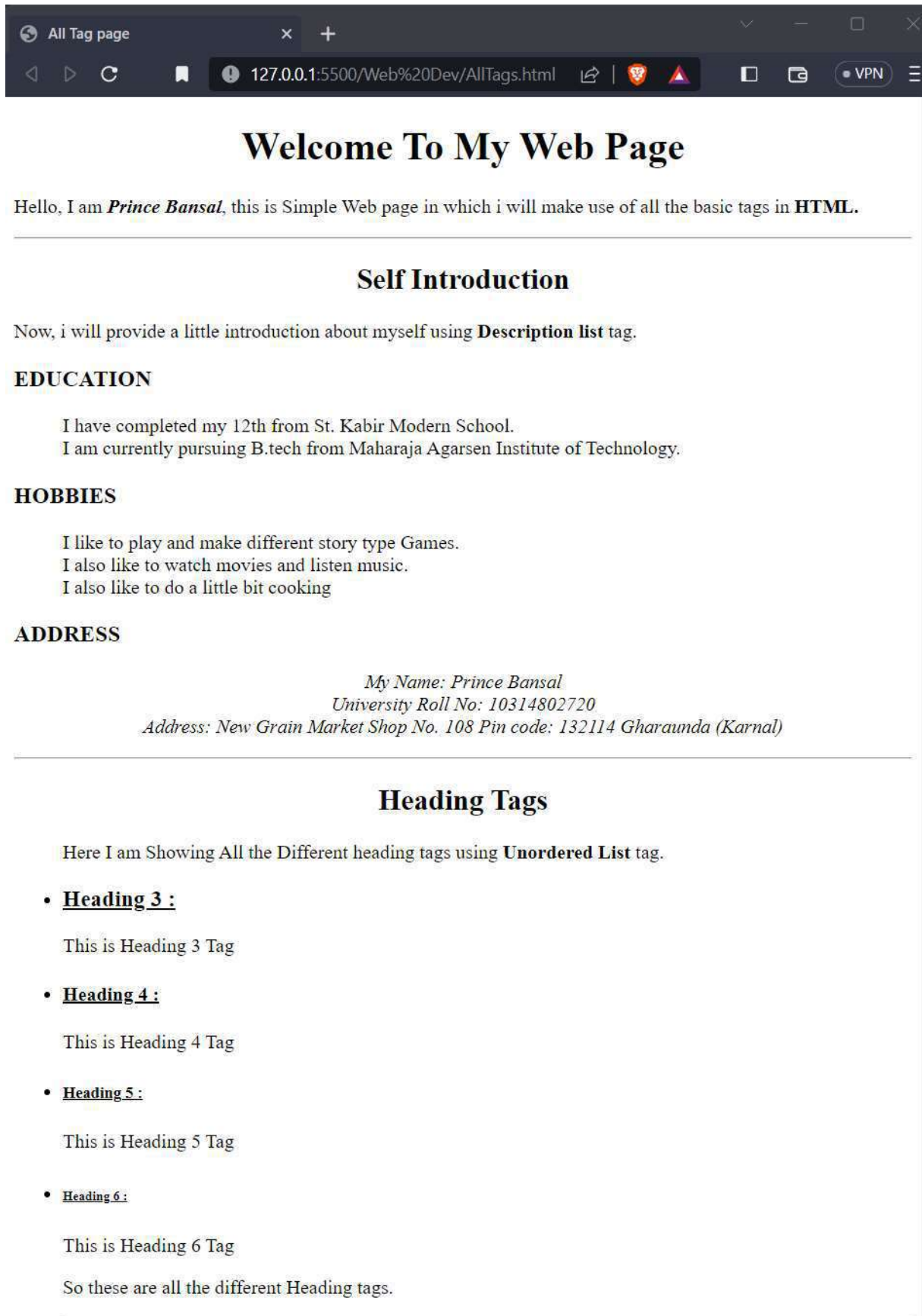
<p> So these are all the different Heading tags.</p>

<hr></hr>

</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:

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

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:

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">

<code><input type="submit"></code>	A submit button that submits the form.	<code><input type="submit" value="Submit"></code>
<code><input type="reset"></code>	A reset button that resets the form to its initial state.	<code><input type="reset" value="Reset"></code>

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>

```

4.	Defines a Header Rows in a table
5.	Defines Each Column in a table
6.	Used to add Border in a Table
7.	Specifies the Distance between Data & Boundaries of a Cell
8.	Used to specify Space between Adjacent Cell
9.	Used to specify width of table.

```

<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="hobbies">Hobbies:</label> </td>
        <td> <input type="checkbox" name="hobbies" id="Gaming"> <label
for="Gaming">Gaming</label>
        <input type="checkbox" name="hobbies" id="Reading"> <label for = "Reading" > Reading
        </label>
        <input type="checkbox" name="hobbies" id="Coding"> <label for="Coding">Coding</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> <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 Rows in a table
5.	< td >	Defines Each Column in a table
6.	border	Used to add Border in a Table
7.	cellpadding	Specifies the Distance between Data & Boundaries of a Cell
8.	cellspacing	Used to specify Space between Adjacent Cell
9.	width	Used to specify width of table.

HTML Form

Name:	<input type="text" value="Prince"/>
Password:	<input type="password" value="*****"/>
E-mail:	<input type="text" value="princebansal992@gmail.co"/>
Date:	<input type="text" value="09/09/2002"/> <input type="button" value="📅"/>
Phone No:	<input type="text" value="83075XXXXX"/>
Gender:	<input checked="" type="radio"/> Male <input type="radio"/> Female <input type="radio"/> Other
Hobbies:	<input checked="" type="checkbox"/> Gaming <input type="checkbox"/> Reading <input checked="" type="checkbox"/> Coding
Course:	<input type="text" value="CS"/> ▼
<input type="button" value="Resot All"/>	<input type="button" value="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 (;).

For 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.

For 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.

For Example

Styles.css :

```

p{
  font-size: "size";
  color: "color name";
}

```

External.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> All Tag page </title>
```

```
</head>
```

```
<body>
```

```
<center>
```

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

```
    <h1><span style="border: 2px solid black;
                    background-color: lightgray ;
                    padding: 5px; font-family: 'Segoe UI';
                    font-size: 40px;" >INLINE CSS </span></h1>
```

```
</center>
```

```
<p style = "border: 1px solid black ;
            background-color: whitesmoke;
            padding: 3px;
            font-family: cursive;
            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 (;).

```
<br><br>
```

for Example:- In this Web page I use

1. The span tag and style attribute is used to add a border (2px solid black), background color(light gray), padding (5px), font family (Segoe ui), and font size(40px) to

the h1 element.

```
<br><br>
```

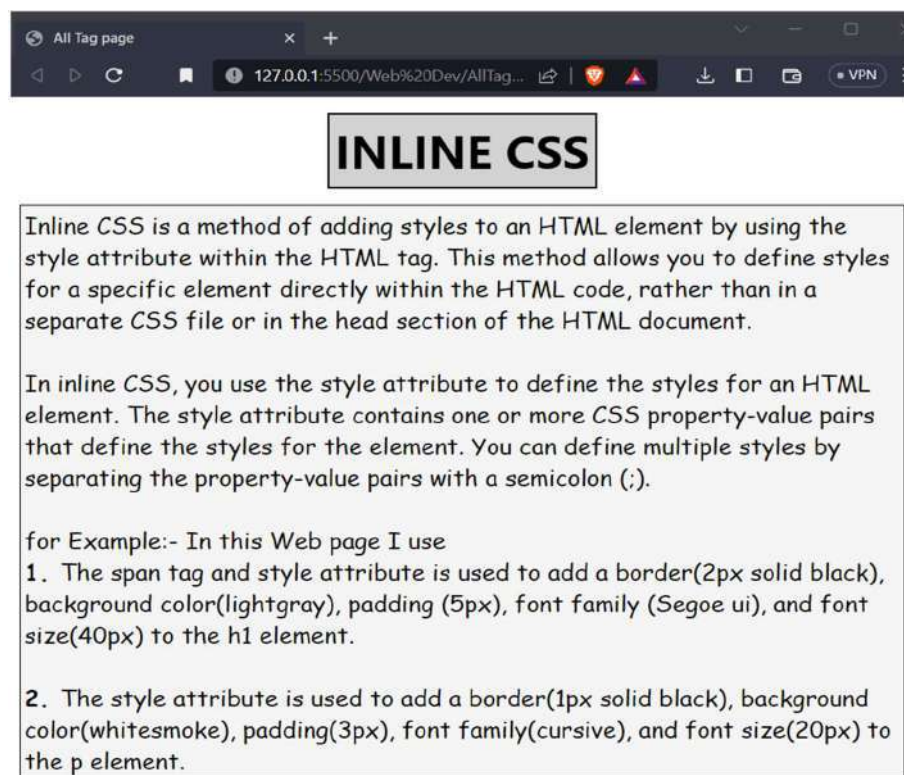
2. The style attribute is used to add a border(1px solid black), background color(whitesmoke), padding(3px), font family(cursive), and font size(20px) to the p element.

```
</p>
```

```
</body>
```

```
</html>
```

OUTPUT:



EMBEDDED CSS:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>All Tag page</title>
```

```
    <style>
```

```
        span.style1
```

```
        {
```

```
            border: 2px solid black;
```



```

        background-color: light gray;
        padding: 5px;
        font-family: Georgia;
        font-size: 40px;
    }
    p.style1
    {
        border: 1px solid black;
        background-color: whitesmoke;
        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.

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.

for Example:- In this Web page,

1. Create a "span.style1" in <i>"style"</i> to add a border(2px solid black), background color(light gray), padding (5px), font family (Georgia), and font size(40px) to the h1 element.

`
`I use it by writing `"< span class="style1">INTERNAL CSS< /span>"` in the h1 Element. `

`

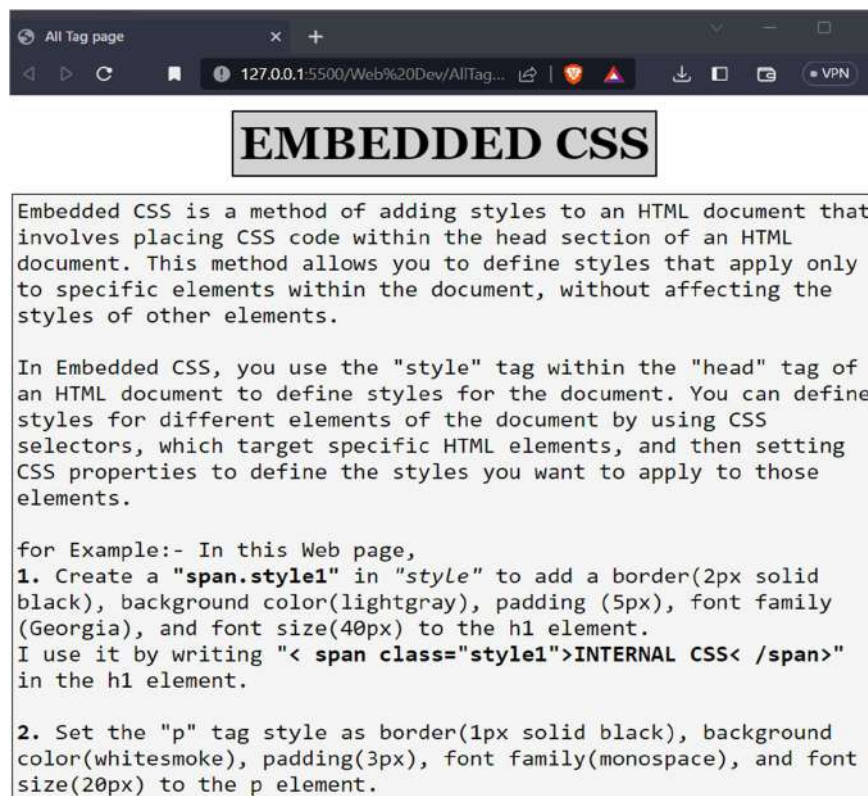
`2. ` Set the "p" tag style as border(1px solid black), background color(whitesmoke), padding(3px), font family(monospace), and font size(20px) to the p element.

`</p>`

`</body>`

`</html>`

OUTPUT:



EXTERNAL CSS:

STYLES.CSS:

```
span.special {  
  border: 2px solid black;  
  background-color: light gray;  
  padding: 5px;  
  font-family: Century;
```

```

    font-size: 40px;
}
p.special1 {
    border: 1px solid black;
    background-color: whitesmoke;
    padding: 3px;
    font-family: perpetua;
    font-size: 20px;
}

```

ALLTAGPAGE.HTML

```

<!DOCTYPE html>
<html>
<head>
    <title>All Tag page</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

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

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.

for Example:- In this Web page,

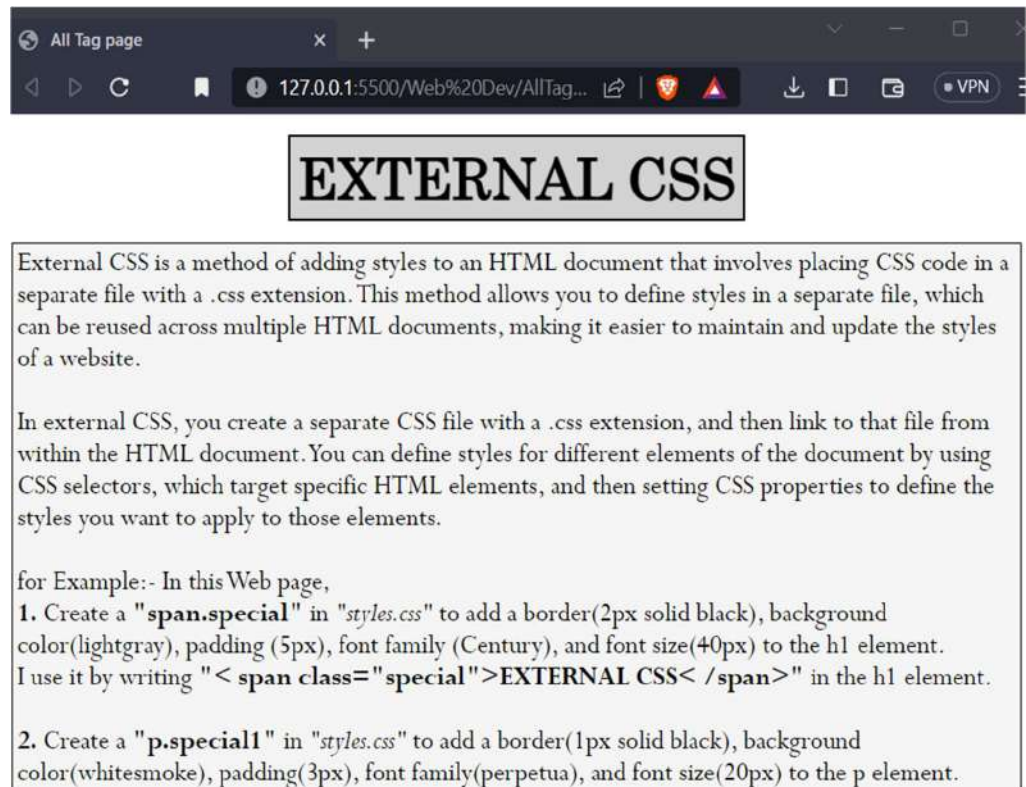
1. Create a `span.special` in `styles.css` to add a border(2px solid black), background color(light gray), padding (5px), font family (Century), and font size(40px) to the h1 element.

I use it by writing `EXTERNAL CSS` in the h1 Element.

2. Create a `p.special1` in `styles.css` to add a border(1px solid black), background color(whitesmoke), padding(3px), font family(perpetua), and font size(20px) to the p element.

```
</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>Prince</Name>
    <Roll_No>103</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>

  <Song>
    <SongName>Kabira</SongName>
    <Singer>Jubin Nautiyal</Singer>
    <Releaseyear>2021</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>
```

```
<FoodItem>
  <Name>Vada Pav</Name>
  <FamousIn>Maharashtra</FamousIn>
  <Price>30 Rs</Price>
  <Calories>300 Calories</Calories>
</FoodItem>
```

```
</IndianFood>
```

STYLES.css:

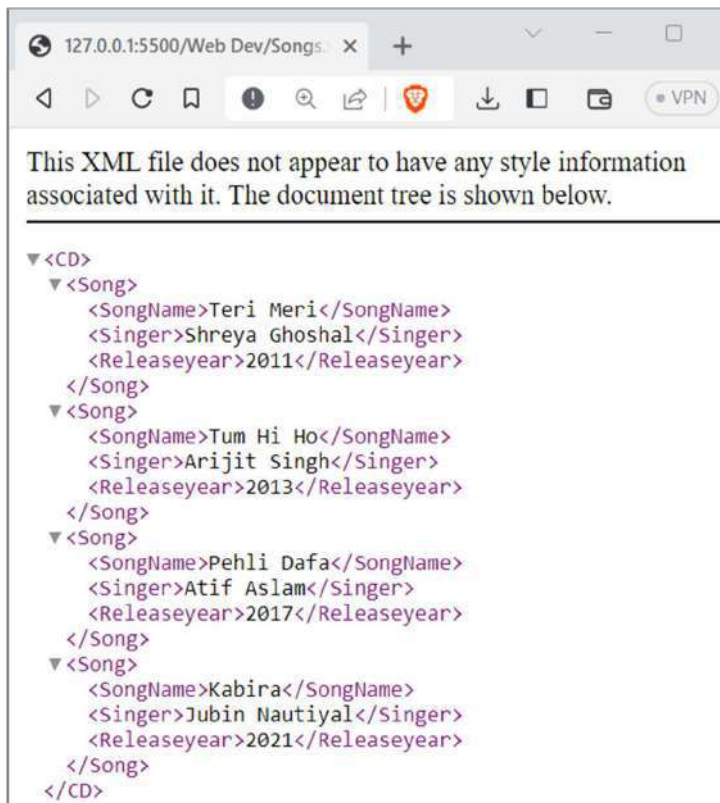
```
heading, FoodItem, Name, FamousIn, Price, Calories{
  display: block;
}
FoodItem{
  margin-top: 20px;
  margin-left: 12px;
}
heading{
  border: 2px solid black;
```

```
background-color: lightgray;
padding: 5px;
font-family: Century;
font-weight: bold;
font-size: 30px;
text-align: center;
}
```

```
Name{
background-color: rgb(214, 213, 213);
padding: 3px;
font-family: Algerian;
font-size: 20px;
}
```

```
FamousIn, Price, Calories{
background-color: whitesmoke;
padding: 3px;
font-family: perpetua;
font-size: 20px;
}
```

OUTPUT :



FAMOUS INDIAN FOODS	
BIRYANI	
Hyderabad	
150 Rs	
400 Calories	
DOSA	
South India	
50 Rs	
200 Calories	
SAMOSA	
North India	
20 Rs	
300 Calories	
VADA PAV	
Maharashtra	
30 Rs	
300 Calories	

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.

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() 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.

Example:

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

SOURCE CODE:

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>All Tag page</title>  
  <link rel="stylesheet" href="styles.css">  
  <style>  
    form {  
      text-align: center;  
      margin: 0;  
      padding: 0;  
      box-sizing: border-box;  
      font-family: 'Times New Roman', Times, serif;  
    }  
  }  
</style>  
</head>  
</html>
```

```
.resultScreen {
    min-width: 180px;
    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>

<script lang="JAVASCRIPT">
    const date = new Date();
</script>

</head>

<body>

    <hr>

    <h2 style="text-align: center;"> Experiment 6(a) </h2>
```

```
<h1 style="text-align: center;"> <span class="special1"> DATE AND TIME </span>
</h1>
```

```
<script lang="JAVASCRIPT">
    document.write("<p style='text-align: center; font-size: 20px; font-family:
monospace;'>");
```

```
    document.write("Today is: " + Date());
```

```
    document.write("</p>");
```

```
</script>
```

```
<hr>
```

```
<h2 style="text-align: center;"> Experiment 6(b) </h2>
```

```
<h1 style="text-align: center;"> <span class="special1"> CALCULATOR </span> </h1>
```

```
<form>
```

```
    <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>
```

```
</form>
```

```
<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 6(a)

DATE AND TIME

Today is: Sun May 28 2023 23:25:26 GMT+0530 (India Standard Time)

Experiment 6(b)

CALCULATOR

AC		DEL	/
1	2	3	*
4	5	6	+
7	8	9	-
.	0	=	

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:

JavaScript Functions:

- **validateEmail(email):** Validates the email input using regular expressions. It checks if the email is empty, and if not, it matches the email pattern.
- **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.
- **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.
- **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>

    .form-div {

      font-family: 'cooper Black';

      margin-left: 20%;

      box-sizing: border-box;

      background-color: rgba(116, 117, 119, 0.616);

      padding: 20px 20px;

      max-width: 270px;

      width: 100%;

      border-radius: 10px;

    }

    .heading {

      font-style: bold;

      text-align: center;

      margin-bottom: 15px;

      margin-top: 5px;

    }

    .input-field {

      font-family: 'cooper Black';

      box-sizing: border-box;

      padding: 8px;

      width: 100%;

      margin-top: 4px;
```

```

    }

    .button {
        padding: 10px;
        width: 100%;
        background-color: #494b4e;
        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 style="font-size: 17px;" for="email">Email</label> <br>

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

            <br><br>

            <label style="font-size: 17px;" for="phone">Phone No.</label> <br>

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

            <br><br>

            <label style="font-size: 17px;" for="pass">Password</label> <br>

            <input class="input-field" type="password" 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:

REGISTER

Email

Enter your email

Phone No.

Enter your phone no.

Password

Enter your password

Submit

127.0.0.1:5500 says

🥳 Form Submitted Successfully!

OK

REGISTER

Email

Princebansal992@gmail.com

Phone No.

1258524875

Password

.....

Submit

127.0.0.1:5500 says

Password must be of 8 length
Atleast one UpperCase Letter
Atleast one LowerCase Letter
Atleast one Number

OK

REGISTER

Email

Princebansal992@gmail.com

Phone No.

1258123423

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:

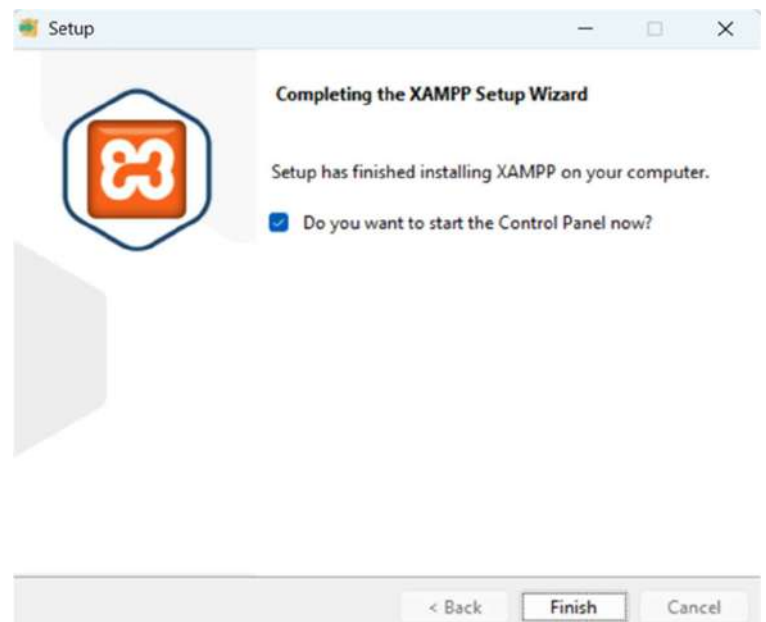
- **Content Management System (CMS)** – A CMS has a central interface that is used to publish, edit, modify, and maintain content. Some well known CMS include, WordPress, Drupal, and Joomla.
- **WordPress** – It is an open-source content management system founded by Matt Mullenweg and Mike Little. It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

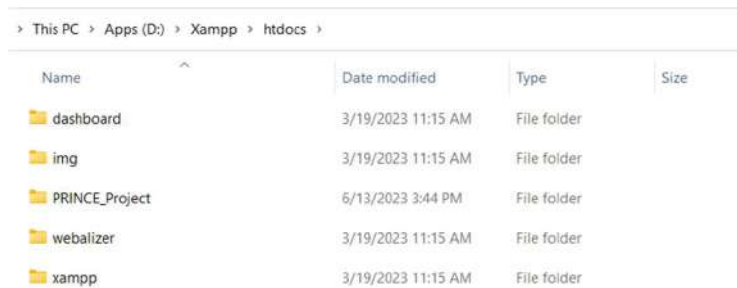
STEP 1: Download and Install XAMPP server.

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

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



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

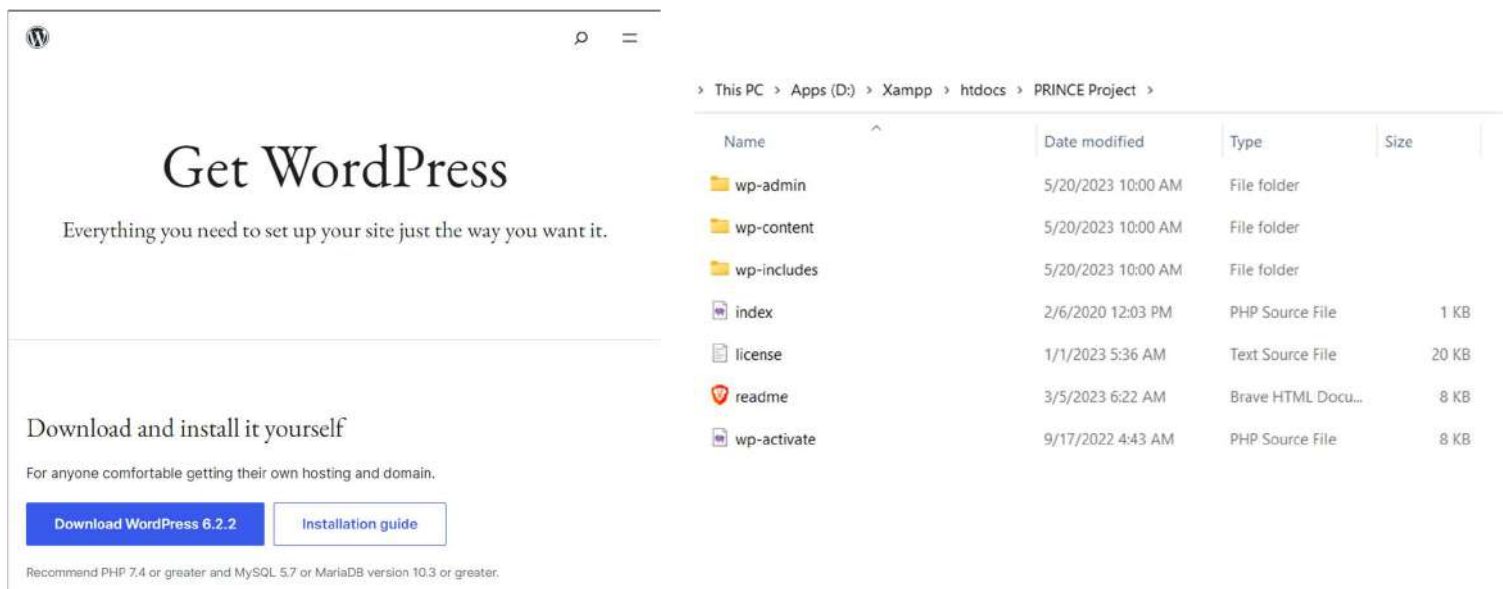


This screenshot shows a Windows File Explorer window with the address bar set to 'This PC > Apps (D:) > Xampp > htdocs >'. The main area displays a list of folders: 'dashboard', 'img', 'PRINCE_Project', 'webalizer', and 'xampp'. Each folder entry includes its name, the date and time it was last modified, its type (File folder), and its size.

Name	Date modified	Type	Size
dashboard	3/19/2023 11:15 AM	File folder	
img	3/19/2023 11:15 AM	File folder	
PRINCE_Project	6/13/2023 3:44 PM	File folder	
webalizer	3/19/2023 11:15 AM	File folder	
xampp	3/19/2023 11:15 AM	File folder	

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\PRINCE_Project**.



This block contains two side-by-side screenshots. The left screenshot shows the 'Get WordPress' page from the official WordPress website, featuring the WordPress logo, the heading 'Get WordPress', and a subheading 'Everything you need to set up your site just the way you want it.' Below this, there's a section 'Download and install it yourself' with two buttons: 'Download WordPress 6.2.2' and 'Installation guide'. The right screenshot shows a Windows File Explorer window with the address bar set to 'This PC > Apps (D:) > Xampp > htdocs > PRINCE Project >'. The main area displays a list of files and folders: 'wp-admin', 'wp-content', 'wp-includes', 'index', 'license', 'readme', and 'wp-activate'. Each entry includes its name, the date and time it was last modified, its type, and its size.

Name	Date modified	Type	Size
wp-admin	5/20/2023 10:00 AM	File folder	
wp-content	5/20/2023 10:00 AM	File folder	
wp-includes	5/20/2023 10:00 AM	File folder	
index	2/6/2020 12:03 PM	PHP Source File	1 KB
license	1/1/2023 5:36 AM	Text Source File	20 KB
readme	3/5/2023 6:22 AM	Brave HTML Docu...	8 KB
wp-activate	9/17/2022 4:43 AM	PHP Source File	8 KB

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**

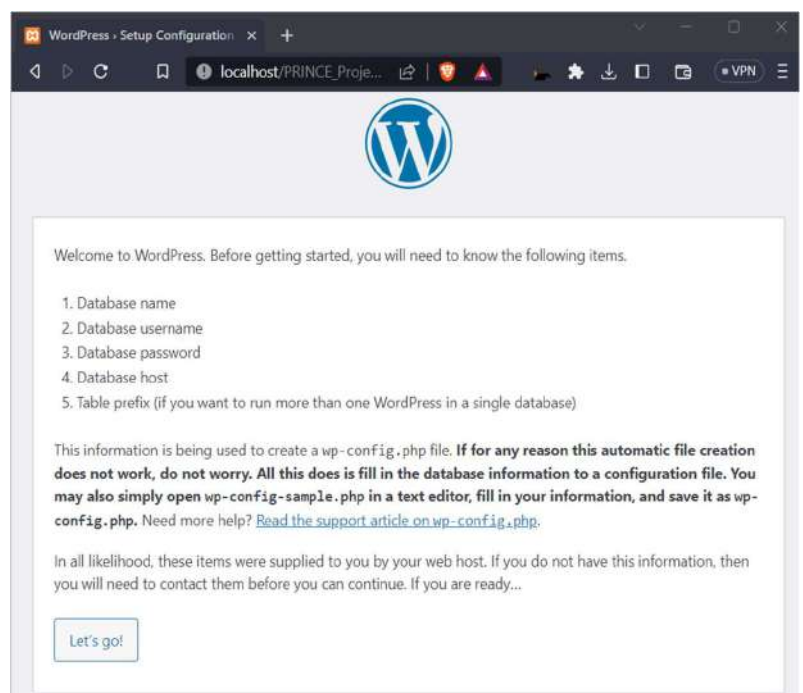
Open the web browser, type **localhost** and press enter. Now we can see the XAMPP dashboard. If the dashboard is visible, it means everything is working fine.



Step 6: Now we will enter the project folder from the browser. For that, type: **localhost/PRINCE_Project/** 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. (E.g.: PrinceProject_DB).



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

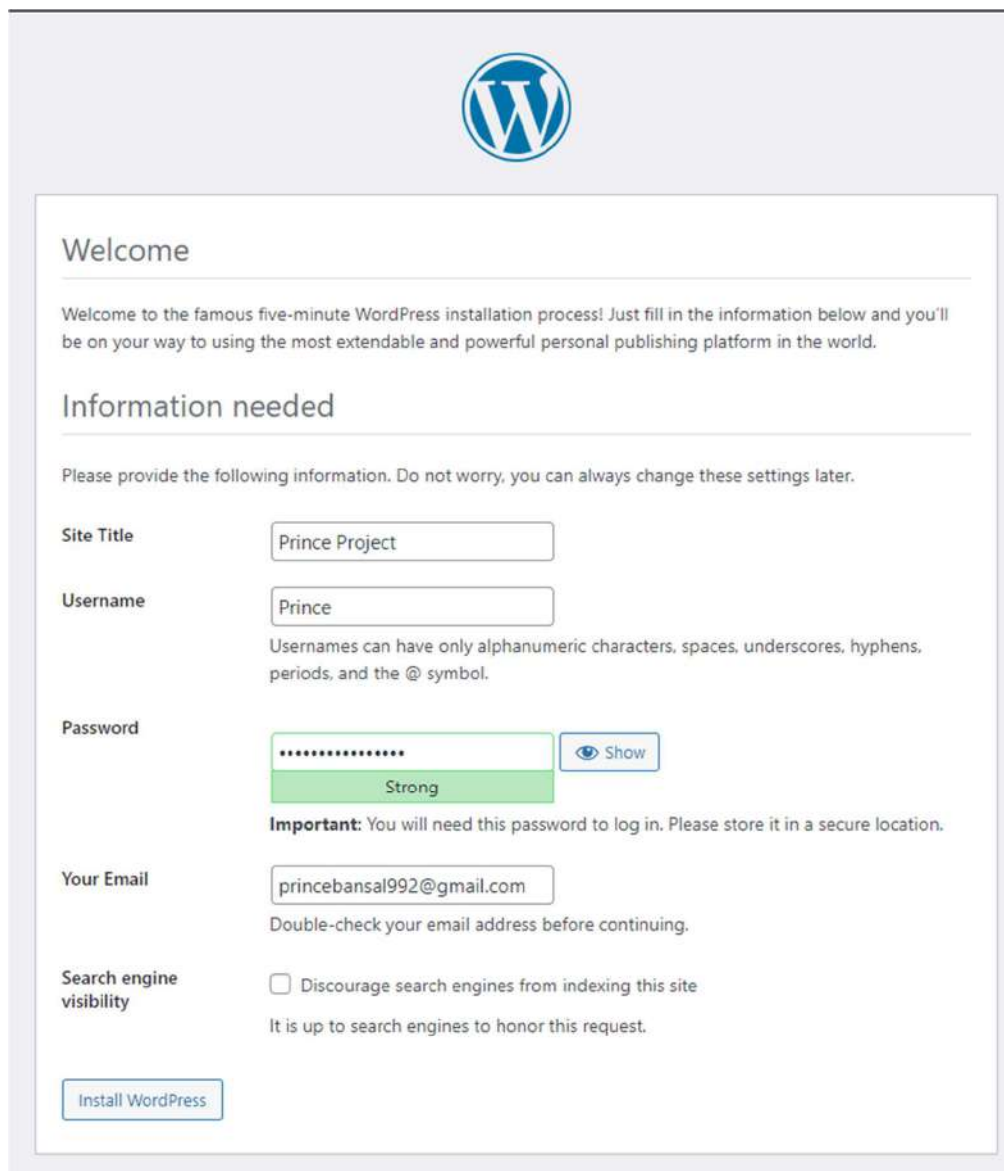
Then click on **Submit** Button. After that click on **Run the Installation**.

This is the 'Database Configuration' screen from the WordPress installation wizard. It features the WordPress logo at the top. Below it, a message states: 'Below you should enter your database connection details. If you are not sure about these, contact your host.' The form has five rows, each with a label, an input field, and a description: 'Database Name' (princeproject_db, description: 'The name of the database you want to use with WordPress.'), 'Username' (root, description: 'Your database username.'), 'Password' (password, description: 'Your database password.'), 'Database Host' (localhost, description: 'You should be able to get this info from your web host, if localhost does not work.'), and 'Table Prefix' (wp_, description: 'If you want to run multiple WordPress installations in a single database, change this.'). A 'Submit' button is located at the bottom left of the form.This is the 'Success' screen after the database configuration. It features the WordPress logo at the top. Below it, a message states: 'All right, sparky! You've made it through this part of the installation. WordPress can now communicate with your database. If you are ready, time now to...'. At the bottom, there is a 'Run the installation' button.

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**”



The image shows the WordPress installation form. At the top is the WordPress logo. Below it is a 'Welcome' section with a message about the five-minute installation process. The 'Information needed' section contains several fields: 'Site Title' with the value 'Prince Project', 'Username' with the value 'Prince', 'Password' with a masked value and a 'Show' button, 'Your Email' with the value 'princebansal992@gmail.com', and 'Search engine visibility' with an unchecked checkbox. A green bar indicates the password is 'Strong'. An 'Important' note states that the password is needed for login. At the bottom is an 'Install WordPress' button.

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title

Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password [Show](#)
Strong

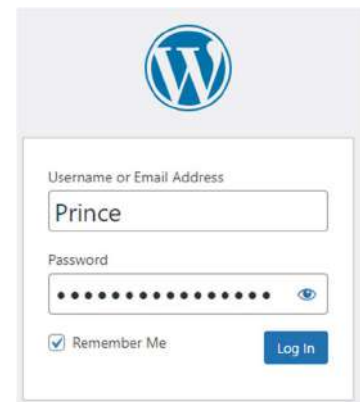
Important: You will need this password to log in. Please store it in a secure location.

Your Email
Double-check your email address before continuing.

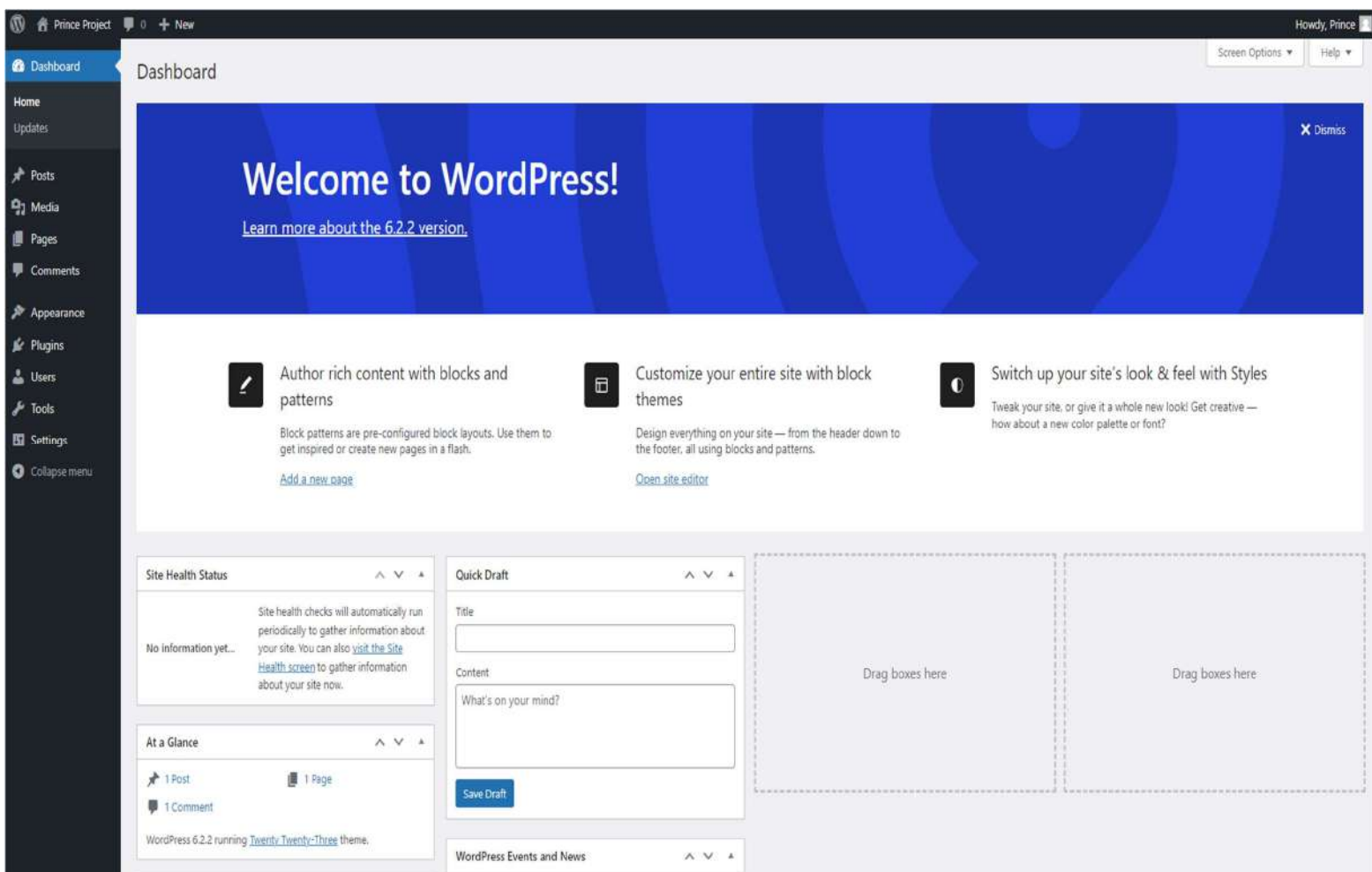
Search engine visibility ☐ Discourage search engines from indexing this site
It is up to search engines to honor this request.

[Install WordPress](#)

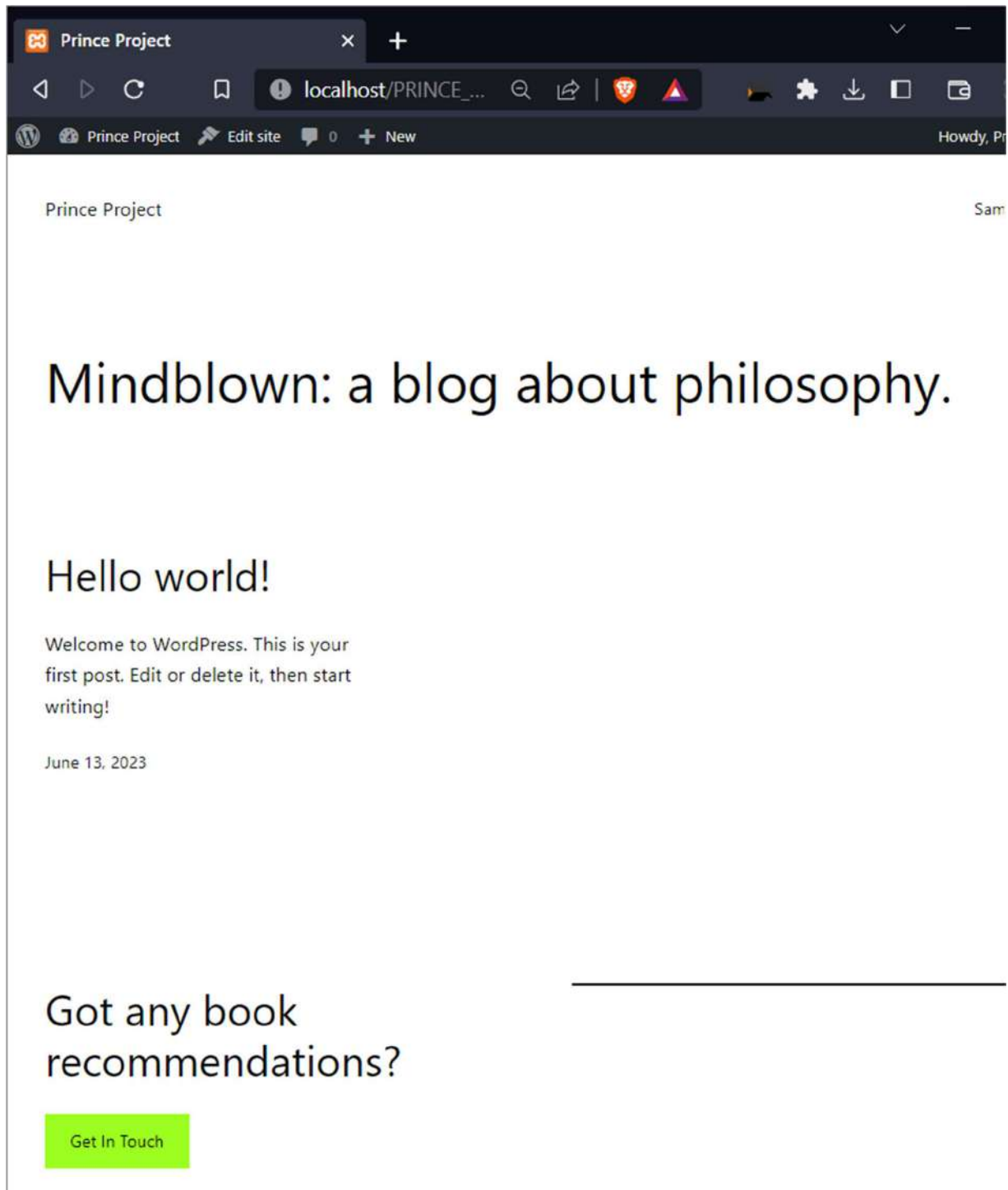
Step 10: Success! Congratulations, successful installation of WordPress. Now, Log In to your WordPress Website



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:

I decide to make a website in which I upload the links of my YouTube channel videos. So, that anyone can directly access the most appropriate video from there. This will be a simple webpage consist of three to four pages. And two to three posts.

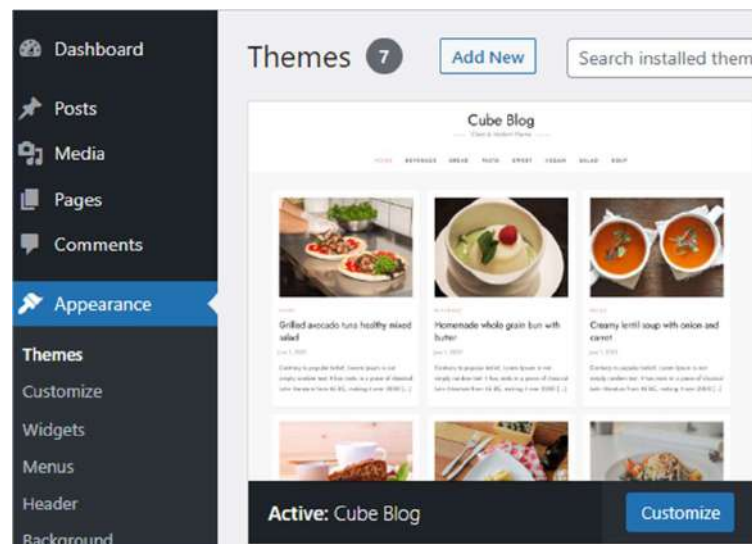
A visual site map, also known as a website diagram or website structure diagram, is a graphical representation or visual depiction of the pages and hierarchical structure of a website. It illustrates the organization and relationships between different web pages, sections, and content on a website. The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

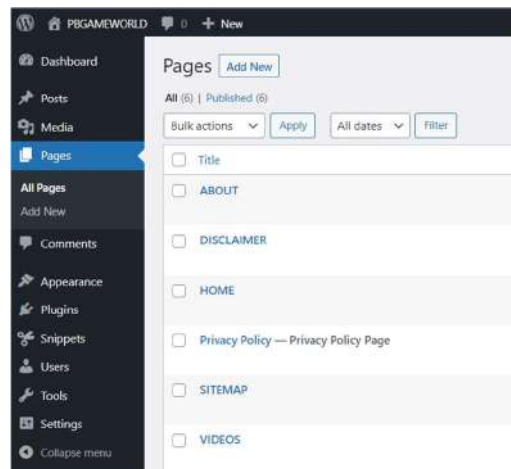
1. Post Section
2. Home Section
3. Videos Section
4. About Section
5. Disclaimer Section
6. Privacy Policy Section

STEPS TO CREATE VISUAL SITE MAP OF WEBSITE:

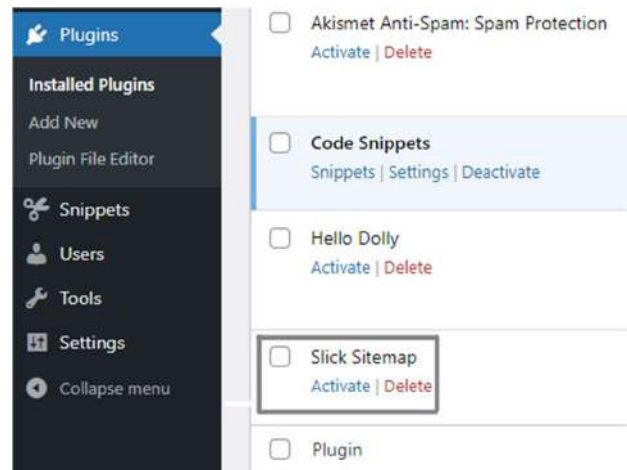
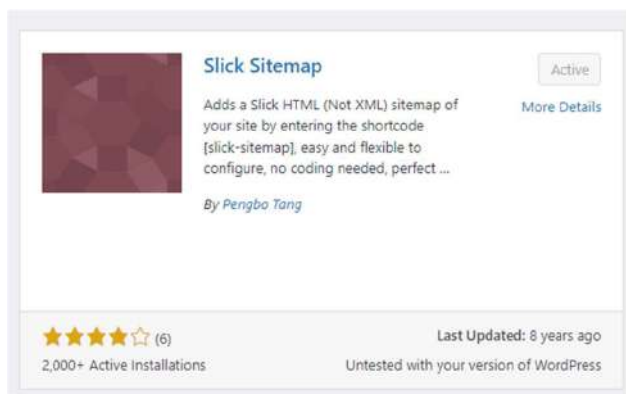
- Go to **localhost/PRINCE_Project/wp-admin/** and then go to Theme inside Appearance and Select a theme for Website And **Activate** it after Installation.



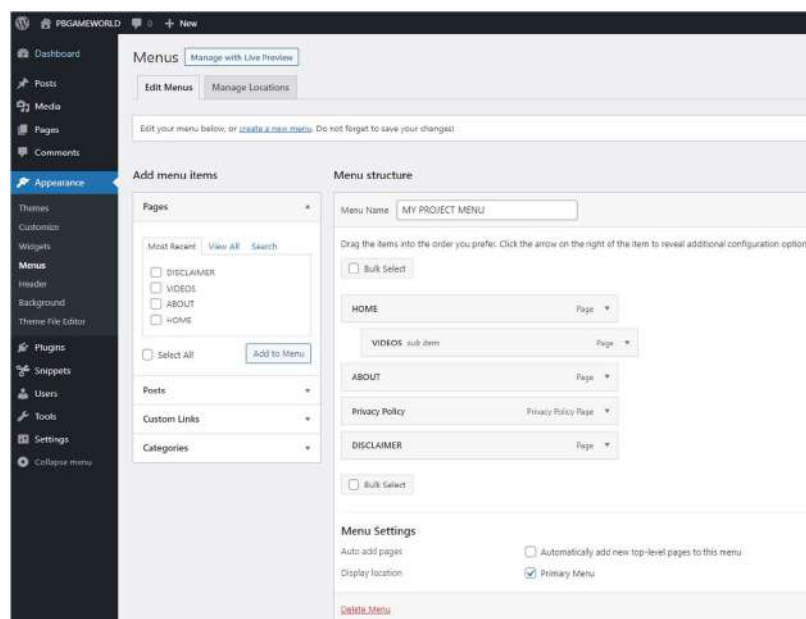
- Now Click on **Pages** -> **Add New** then Create As many pages a wanted in the Website.



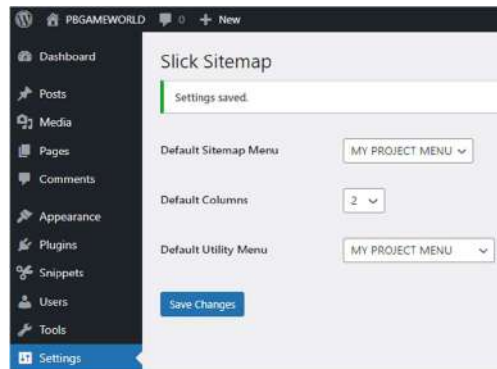
- Now Go to **Plugins** -> **Add New** and Install Slick Sitemap and then activate it.



- 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 .



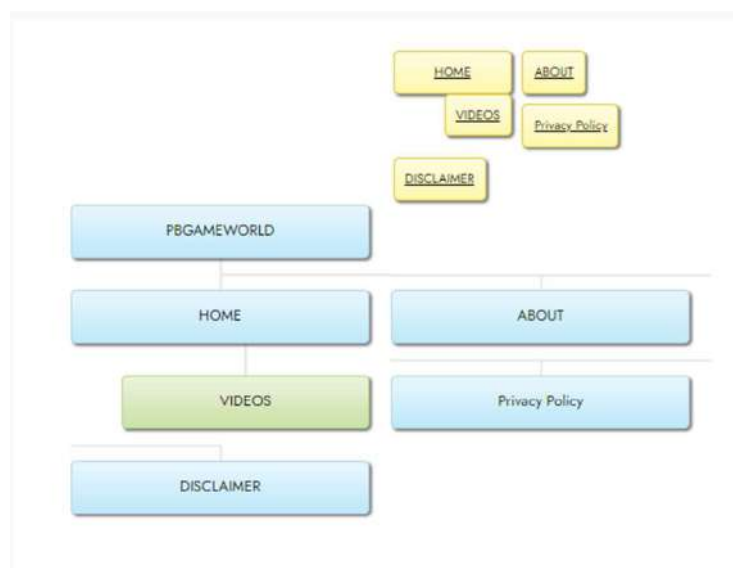
- Go to **Settings** -> **Slick-Sitemap** and Choose the Default Sitemap and utility menu by selecting the menu you created in previous step and also set column to two.



- 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:

Contents In the Website :

- **Post Section:** It is the main page of my Website where I post the latest videos post of my channel and also post updates about my subscribers etc.
- **Home Section:** The Home section serves as the main landing page for users. It showcases a personalized feed of recommended videos based on the user's viewing history, subscriptions, and interests. The Home section also features trending videos, popular channels, and curated playlists. Users can easily navigate to different content categories, such as gaming, music, or news, to discover new videos. Additionally, the Home section may include featured channels or videos promoted by YouTube.
- **Videos Section:** It Will be Inside my Home section. It contains all the links of Videos which is currently on my YouTube channel.
- **About Section:** The About section typically provides essential information about the channel or video creator. It offers a concise summary of the channel's content, mission, and purpose. This section may include details about the creator's background, interests, and expertise. It may also include links to other social media platforms, merchandise, or a website associated with the channel.
- **Disclaimer:** The Disclaimer section typically clarifies the limitations and responsibilities of the content provided. It states that the information presented is for general informational purposes only and should not be considered as professional advice. It emphasizes that viewers should consult professionals or experts in specific fields for personalized guidance. The disclaimer may also mention that the opinions expressed in the videos are those of the creator and not necessarily endorsed by YouTube.
- **Privacy Policy Section:** The Privacy Policy outlines how user data is collected, used, and protected on the platform. It includes information about the types of data collected, such as personal information and browsing behavior. The policy explains how this data is used for analytics, personalized recommendations, and targeted advertising. It also highlights the measures taken to ensure data security and user privacy. Additionally, the Privacy Policy section may include details about users' rights regarding their data, such as opting out of certain data collection practices or accessing and deleting their personal information.

ADDING CONTENT :

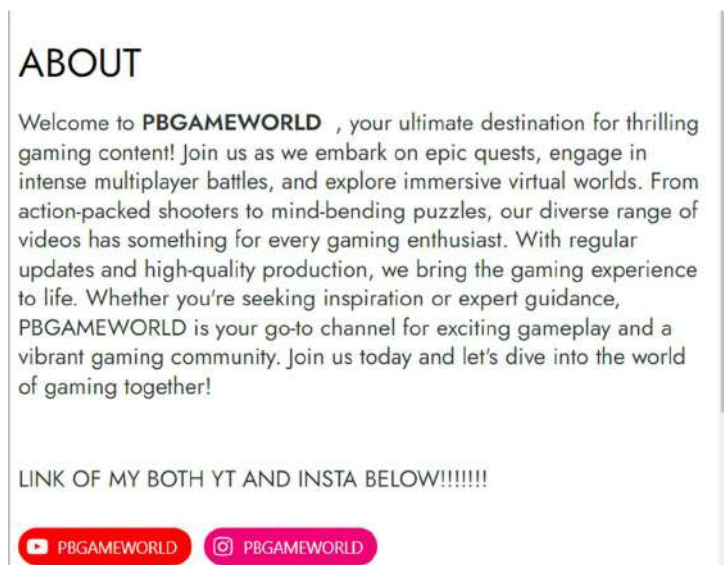
1. **HOME SECTION:** Go to **localhost/PRINCE_Project/wp-admin/** then go to **Pages -> Add New** Now give title as **HOME** and add the content and then Publish it.



2. **VIDEOS SECTION:** Repeat the same step 1 as you did for HOME. Just name the title as **VIDEOS**.



3. **ABOUT SECTION:** Repeat the same step 1 as you did for HOME. Just name the title as **ABOUT**.



4. **DISCLAIMER:** Repeat the same step 1 as you did for HOME. Just name the title as **DISCLAIMER**.

DISCLAIMER

Disclaimer for PBGAMEWORLD

If you require any more information or have any questions about our site's disclaimer, please feel free to contact us by email at PBGameWorldOfficial@gmail.com.

Disclaimers for PBGAMEWORLD

All the information on this website - http://localhost/PRINCE_Project/ - is published in good faith and for general information purpose only. **PBGAMEWORLD** does not make any warranties about the completeness, reliability and accuracy of this information. Any action you take upon the information you find on this website, is strictly at your own risk. PBGAMEWORLD will not be liable for any losses and/or damages in connection with the use of our website.

5. **PRIVACY POLICY:** Repeat the same step 1 as you did for HOME. Just Name the title as Privacy Policy.

Privacy Policy

Who we are

Suggested text: Our website address is: http://localhost/PRINCE_Project/.

Comments

Suggested text: When visitors leave comments on the site we collect the data shown in the comments form, and also the visitor's IP address and browser user agent string to help spam detection.

6. **POST SECTION:** Go to **localhost/PRINCE_Project/wp-admin/** then go to **Posts -> Add New** Add the content and then Publish it.

SIMPLE INFINITE IRON FARM Minecraft

1.19.2 💰 🏠



OUTPUT:

The Final Website:

