8. Write a program using PHP and HTML to create a form and display the details entered by the user.

What is PHP?

PHP (Hypertext Preprocessor) is a widely used server-side scripting language that is designed for web development. It is embedded in HTML and executed on the server, generating dynamic content for web pages.

PHP is a versatile language with many more features for handling databases, file manipulation, error handling, and more. It is widely used in conjunction with web servers like Apache and databases like MySQL for building dynamic web applications.

Here are some basics of PHP,

- **Syntax**: PHP code is enclosed within **<?php ... ?>** tags.
  For example:
  ```
  <?php
      // PHP code goes here
  ?>
  ```

- **Variables**: PHP variables start with a dollar sign **$** followed by the variable name. PHP is loosely typed, so you don't need to declare the variable type explicitly.
  ```
  $name = "John";
  $age = 25;
  ```

- **Data Types**: PHP supports various data types, including strings, integers, floats, booleans, arrays, objects and more.

- **Comments**: PHP comments can be added using **//** for single-line comments and **/* ... */** for multi-line comments.

- **Output**: You can display output using **echo** or **print** functions.

```
echo "Hello, World!";
```

- **Conditional Statements**: PHP supports if-else and switch statements for conditional branching.

```
if ($age >= 18) {
   echo "You are an adult.";
} else {
   echo "You are a minor.";
}
```

- **Loops**: PHP provides various types of loops, such as **for**, **while**, and **foreach**.

```
for ($i = 1; $i <= 5; $i++) {
   echo $i;
}
```

- **Functions**: You can define your custom functions in PHP.

```
function greet($name) {
   echo "Hello, " . $name . "!";
}
greet("Alice"); // Output: Hello, Alice!
```

- **Arrays**: PHP supports both indexed and associative arrays.

```
// Indexed Array
$fruits = array("Apple", "Banana", "Orange");

// Associative Array
$person = array("name" => "John", "age" => 25);
```

- **Super Global Variables**: PHP provides super global variables like **$_GET**, **$_POST**, **$_SESSION**, **$_COOKIE**, **$_FILES**, etc., to access data from different sources.

- **Include and Require**: PHP allows you to include external files using **include** or **require** statements.

```
// Include a file (non-fatal error if file not found)
include "header.php";

// Require a file (fatal error if file not found)
require "config.php";
```

- **Form Handling**: PHP can be used to process form data submitted from HTML forms.

What is XAMPP?

XAMPP is a free and open-source software package that provides a local server environment for web development and testing.

The name "XAMPP" stands for:

- **X**: Cross-platform (available for different operating systems)
- **A**: Apache HTTP Server
- **M**: MariaDB (formerly MySQL) database
- **P**: PHP
- **P**: Perl

It was originally created by Apache Friends and is now maintained by the Apache Friends community.

XAMPP bundles together all the essential components required to set up a web server environment, making it easy for developers to create and test web applications on their local machines. It is available for Windows, macOS, Linux, and other operating systems.

The core components of XAMPP are:

1. **Apache HTTP Server**: A popular web server software that handles HTTP requests from clients (browsers) and serves web pages in response.
2. **MariaDB (MySQL) Database**: A relational database management system used for storing and managing data, commonly used with web applications.
3. **PHP**: The server-side scripting language used for creating dynamic web pages and interacting with databases.
4. **Perl**: A programming language mainly used for server-side scripting and command-line processing.

In addition to these core components, XAMPP also includes other useful tools and libraries like phpMyAdmin (for managing databases through a web interface), FileZilla FTP server (for handling file transfers), and more.

Using XAMPP, developers can set up a local web server environment quickly without the need for individual installation and configuration of each component. This local server environment mimics the functionalities of a live web server, enabling developers to work on their web projects locally before deploying them to a remote server for public access. It is widely used for web development, testing and learning purposes.

XAMPP installation

To install XAMPP on Windows, follow these steps:

1. **Download XAMPP**: Visit the official Apache Friends website (https://www.apachefriends.org/download.html) and download the latest version of XAMPP for Windows.
2. **Run the Installer**: Once the download is complete, locate the installer file and double-click on it to run it. You may see a security warning; click "Yes" or "Run" to proceed.
3. **Select Components**: The installer will prompt you to select which components you want to install. These components typically include Apache, MySQL, PHP, and phpMyAdmin. You can keep the default selection or customize it based on your requirements. Then, click "Next" to proceed.
4. **Choose Installation Folder**: Choose the folder where you want to install XAMPP. The default location is usually "C:\xampp," but you can change it if needed. Click "Next."
5. **Start Menu Folder**: You can choose whether to create a shortcut in the Start Menu for XAMPP. This is optional, so choose according to your preference. Click "Next."
6. **Bitnami for XAMPP (Optional)**: The installer might ask if you want to install Bitnami for XAMPP, which provides additional software applications like WordPress, Joomla, etc. You can decide whether to install this or not. Click "Next."
7. **Ready to Install**: Review the installation settings and click "Next" to begin the installation process.
8. **Wait for Installation**: The installer will now proceed to install XAMPP along with the selected components. This may take a few minutes.

9. **Firewall Warning**: During the installation, your firewall might prompt you to allow Apache HTTP Server to access the network. Choose to allow access, as it's required for XAMPP to work properly.
10. **Installation Complete**: Once the installation is finished, you'll see a confirmation message. Make sure the "Start the control panel now" option is checked, and click "Finish."
11. **Start XAMPP Control Panel**: The XAMPP Control Panel will open. From here, you can start or stop the Apache and MySQL services. To start using XAMPP, click the "Start" buttons next to Apache and MySQL.
12. **Testing XAMPP**: To verify that XAMPP is running correctly, open your web browser and type "http://localhost" (without quotes) in the address bar. If everything is set up correctly, you should see the XAMPP dashboard.
13. **Using XAMPP**: Now that XAMPP is installed and running, you can place your web files in the "htdocs" folder located inside the XAMPP installation directory. For example, "C:\xampp\htdocs". This is where you can develop and test your web applications locally.

Remember to keep XAMPP updated and secure, especially if you plan to use it on a public network. Additionally, avoid using XAMPP as a production server, as it is primarily designed for development and testing purposes. For production, consider using a dedicated web hosting service.

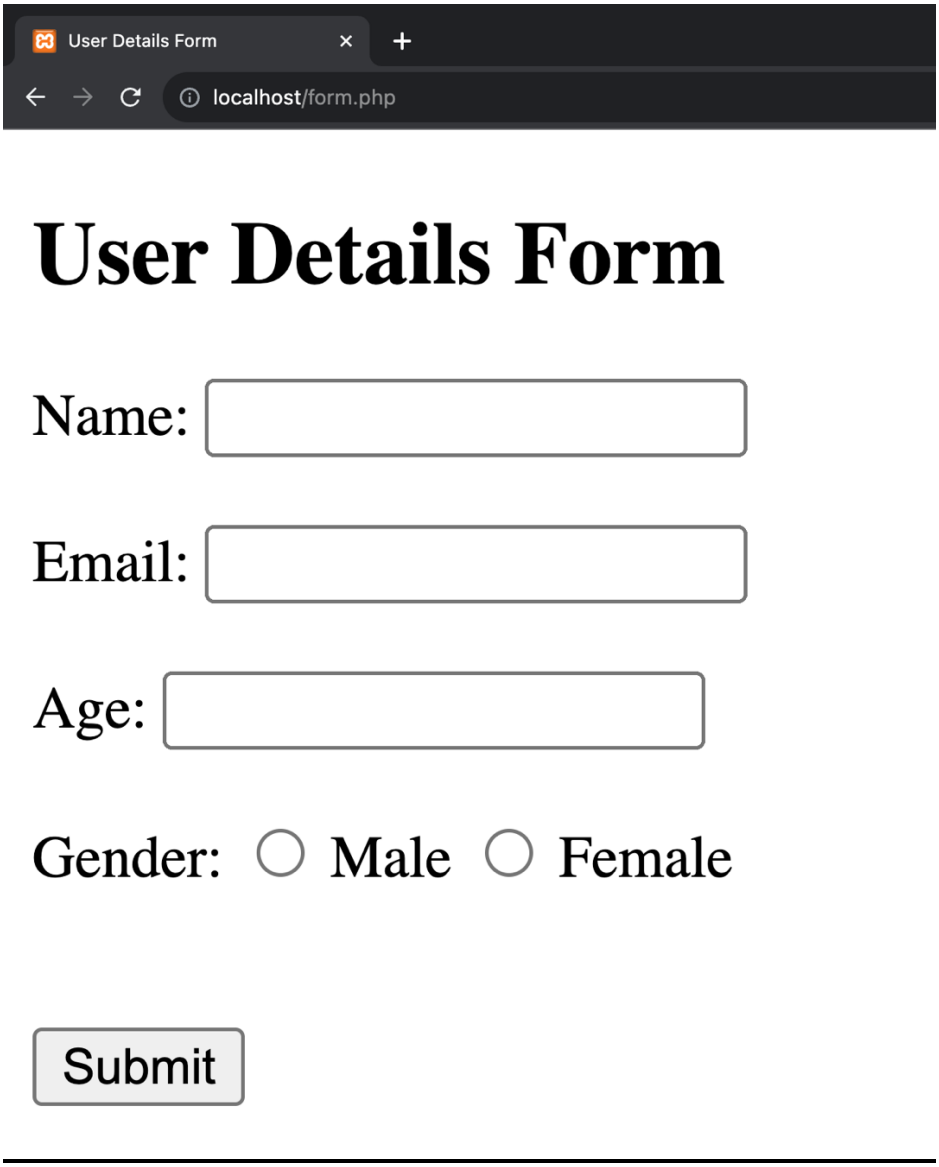
form.php

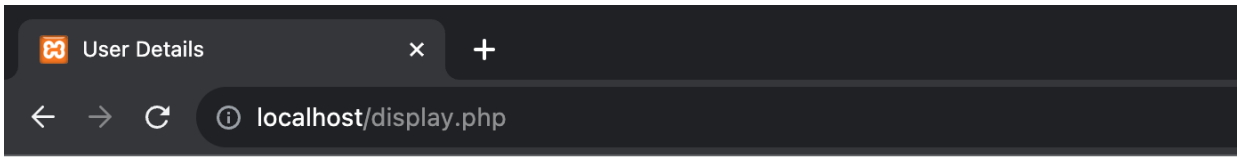
```
<!DOCTYPE html>
<html>

<head>
    <title>User Details Form</title>
</head>

<body>
    <h2>User Details Form</h2>

    <form method="post" action="display.php">
        <label for="name">Name:</label>
        <input type="text" name="name" required><br><br>

        <label for="email">Email:</label>
        <input type="email" name="email" required><br><br>

        <label for="age">Age:</label>
        <input type="number" name="age" required><br><br>

        <label for="gender">Gender:</label>
        <input type="radio" name="gender" value="Male" required> Male
        <input type="radio" name="gender" value="Female" required> Female<br><br>
        <br>

        <input type="submit" value="Submit">
    </form>

</body>
</html>
```

display.php

```
<!DOCTYPE html>
<html>

<head>
   <title>User Details</title>
</head>

<body>
   <h2>Submitted User Details</h2>
   <?php
   if ($_SERVER["REQUEST_METHOD"] === "POST")
   {
      $name = $_POST["name"];
      $email = $_POST["email"];
      $age = $_POST["age"];
      $gender = $_POST["gender"];

      echo "Name: " . $name . "<br><br>";
      echo "Email: " . $email . "<br><br>";
      echo "Age: " . $age . "<br><br>";
      echo "Gender: " . $gender . "<br><br>";
   }
   ?>
</body>
</html>
```

OUTPUT:


User Details Form
localhost/form.php
User Details Form
Name:
Email:
Age:
Gender: Male Female
Submit



Submitted User Details

Name: Akshat Kumar

Email: akshat18kumar@gmail.com

Age: 20

Gender: Male
