ANNAMALAI UNIVERSITY

**FACULTY OF ENGINEERING AND TECHNOLOGY**

# DEPARTMENT OF INFORMATION TECHNOLOGY

**B.E. Information Technology**

**V-SEMESTER**

**22ITCP509 – WEB ESSENTIALS LAB**

**LABORATORY RECORD**

**(July 2024 – December 2024)**

**Name : \_\_\_**

**Register No. :**



ANNAMALAI UNIVERSITY

**FACULTY OF ENGINEERING AND TECHNOLOGY**

# DEPARTMENT OF INFORMATION TECHNOLOGY

**B.E. Information Technology**

**V-SEMESTER**

**22ITCP509 – WEB ESSENTIALS LAB**

Certified that this is the bona-fide record of work done by

Mr./Ms.

Reg. No. of B.E. (Information Technology) in the

**22ITCP509 – Web Essentials Lab** during the odd semester of the

academic year 2024 – 2025.

Place: Annamalai Nagar

Date: / / 2023. **Staff in-charge**

**Internal Examiner External Examiner**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**B.E INFORMATION TECHNOLOGY**

# VISION:

# To produce globally competent, quality technocrats, to inculcate values of leadership and research qualities and to play a vital role in the socio-economic progress

# of the nation.

# MISSION:

1. To partner with the University community to understand the information technology needs of faculty, staff and students.
2. To develop dynamic IT professionals with globally competitive learning experience by providing high class education.
3. To involve graduates in understanding need-based Research activities and disseminate the knowledge to develop entrepreneur skills.

## PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

|  |  |
| --- | --- |
| **PEO** | **PEO Statements** |
| PEO1 | To offer students with **core competence** in mathematical, scientific and basic engineering rudiments necessary to prepare, analyze and solve hardware/software engineering problems and/or also to pursue advanced study or research. |
| PEO2 | To educate students with good **scope** of knowledge in core areas of IT and related engineering so as to comprehend engineering trade-offs, analyze, design, and synthesize data and technical concepts to create novel products and solutions for the real-life problems. |
| PEO3 | To instil in students to maintain high **proficiency** and ethical standards, effective oral and written communication skills, to work as part of teams on multidisciplinary projects and diverse professional environments, and relate engineering issues to the society, global economy and to emerging technologies. |
| PEO4 | To deliver our graduates with **learning environment** awareness of the life-long learning needed for a successful professional career and to introduce them to written ethical codes and guidelines, perform excellence, leadership and demonstrate good citizenship. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **22ITCP509** | **WEB ESSENTIALS LAB** | **L** | **T** | **P** | **C** |
| **0** | **0** | **3** | **1.5** |

## COURSE OBJECTIVES

* + - To analyze, create interactive and dynamic websites.
    - To demonstrate the use of scripting languages.
    - To write simple scripts for the creation of web sites.
    - To create database applications.

## LIST OF EXERCISES

1. Creation of interactive web sites – Design using HTML and authoring tools.
2. Form validation using JavaScript.
3. Creation of simple PHP scripts.
4. Handling multimedia contents in web sites.
5. Write programs using Servlets:
   * 1. To invoke servlets from HTML forms.
     2. Session tracking using hidden form fields and Session tracking for hit count.
6. Creation of information retrieval system using web, PHP and MySQL.
7. Creation of personal Information System.

## COURSE OUTCOMES

At the end of this course, the students will be able to

1. Apply the JavaScript, HTML, and CSS effectively to create interactive and dynamic websites.
2. Create simple PHP scripts and design and deploy simple web-applications.
3. Create simple database applications and Handle multimedia components.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mapping of Course Outcomes (COs) with Program Outcomes (POs) and Program Specific Outcomes (PSOs)** | | | | | | | | | | | | | | | |
|  | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO1** | 3 | 2 | 3 | 3 | 3 | - | - | - | - | - | - | - | 3 | 2 | 2 |
| **CO2** | 3 | 3 | 3 | 3 | 2 | - | - | - | 2 | - | - | - | 3 | 2 | 1 |
| **CO3** | 3 | 3 | 3 | 3 | 2 | - | - | - | 1 | - | - | - | 3 | 2 | 1 |

**Rubric for CO3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rubric for CO3 in Laboratory Courses** | | | | | |
| **Rubric** | **Distribution of 10 Marks for CIE/SEE Evaluation Out of 40/60 Marks** | | | | |
| **Up To 2.5**  **Marks** | **Up To 5**  **Marks** | **Up To 7.5**  **Marks** | **Up To 10**  **marks** | **Up To 2.5**  **Marks** |
| **Demonstrate** | Poor listening | Showed better | Demonstrated | Demonstrated | Demonstrate |
| **an ability to** | and | communication | good | excellent | an ability to |
| **listen and** | communication | skill by relating | communication | communication | listen and |
| **answer the** | skills. Failed to | the problem | skills by | skills by | answer the |
| **viva questions** | relate the | with the | relating the | relating the | viva |
| **related to** | programming | programming | problem with | problem with | questions |
| **programming** | skills needed | skills acquired | the | the | related to |
| **skills needed** | for solving the | but the | programming | programming | programming |
| **for solving** | problem. | description | skills acquired | skills acquired | skills needed |
| **real-world** |  | showed serious | with few errors. | and have been | for solving |
| **problems in** |  | errors. |  | successful in | real-world |
| **Computer** |  |  |  | tailoring the | problems in |
| **Science and** |  |  |  | description. | Computer |
| **Engineering.** |  |  |  |  | Science and |
|  |  |  |  |  | Engineering. |

**List of Exercises**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No.** | **Date** | **Experiment Name** | **Page No.** | **Marks** | **Signature** |
| 1. |  | Creation of interactive web sites – Design  using HTML and authorizing  tools | 2 |  |  |
| 2. |  | Form validation using JavaScript | 5 |  |  |
| 3. |  | Creation of simple PHP scripts | 9 |  |  |
| 4. |  | Handling multimedia contents in websites | 12 |  |  |
| 5. |  | Write programs using Servlets | 18 |  |  |
| 6. |  | Creation of information retrieval system using web, PHP, and MySQL | 23 |  |  |
| 7. |  | Creation of personal information system | 32 |  |  |

|  |  |
| --- | --- |
| **Exp No: 1** | **Creation of interactive web sites – Design using HTML and authoring tools** |
| **Date:** |

**Aim:**

To create a of interactive web sites – Design using html and authoring tools to arrange the shapes of circle, rectangle, and square according with their circumference.

**Algorithm:**

1. Define an array named shapes to store shape objects.

2. Each object should have properties for type (square, rectangle, circle) and its dimensions (width, height, diameter).

3. For each shape object, calculate its circumference based on its type.

4. Sort the shapes array in ascending order of circumference using sort() method.

5. Display the shapes in ascending order by creating HTML elements for each shape object.

6. Append these elements to a container element in the HTML document. Include the shape type and circumference in the display text.

7. Sort the shapes array in descending order based on the circumference of each shape object using the sort() method.

8. Display the shapes in descending order by creating HTML elements for each shape object.

9. Append these elements to another container element in the HTML document. Include the shape type and circumference in the display text.

10. End the algorithm.

**PROGRAM:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Shapes arrangement and circumference calculation</title>

<style>

body{

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

font-weight: 600;

}

.container{

justify-content: center;

align-items: center;

padding: 20px;

}.shape{

border: 1px solid black;

text-align: center;

padding: 10px;

}.square{

width: 4cm;

height: 4cm;

}.rectangle{

width: 6cm;

height: 3cm;

}.circle{

width: 4cm;

height: 4cm;

border-radius: 50%;

}

</style>

</head>

<body>

<h1>shapes arrangement and circumference calculatuon - ascending order</h1>

<div class="container" id="ascendingcontainer"></div>

<h1>shape arrangement and circumference calculation - descending order</h1>

<div class="container" id="descendingcontainer"></div>

<script>

shapes=[{type:'square',width:4,height:4},{type:'rectangle',width:6,height:3},

{type:'circle',diameter:4}];

shapes.forEach(function(shape) {

if(shape.type==='square'|| shape.type === 'rectangle'){

shape.circumference=2\*(shape.width+shape.height);

}else if(shape.type ==='circle') {

shape.radius=shape.diameter/2;

shape.circumference=2\*Math.PI\*shape.radius;

}

});

var ascendingshapes=shapes.slice().sort(function(a,b){

return a.circumference - b.circumference;

});

var ascendingcontainer = document.getElementById('ascendingcontainer');

ascendingshapes.forEach(function(shape){

var div=document.createElement('div');

div.className='shape'+ shape.type;

div.innerText=shape.type.charAt(0).toUpperCase()+shape.type.slice(1)+

'(circumference:'+shape.circumference.toFixed(2)+'cm)';

ascendingcontainer.appendChild(div);

});

var descendingshapes=shapes.slice().sort(function(a,b){

return b.circumference - a.circumference;

});

var descendingcontainer = document.getElementById('descendingcontainer');

descendingshapes.forEach(function(shape){

var div=document.createElement('div');

div.className='shape'+ shape.type;

div.innerText=shape.type.charAt(0).toUpperCase()+shape.type.slice(1)+

'(circumference:'+shape.circumference.toFixed(2)+'cm)';

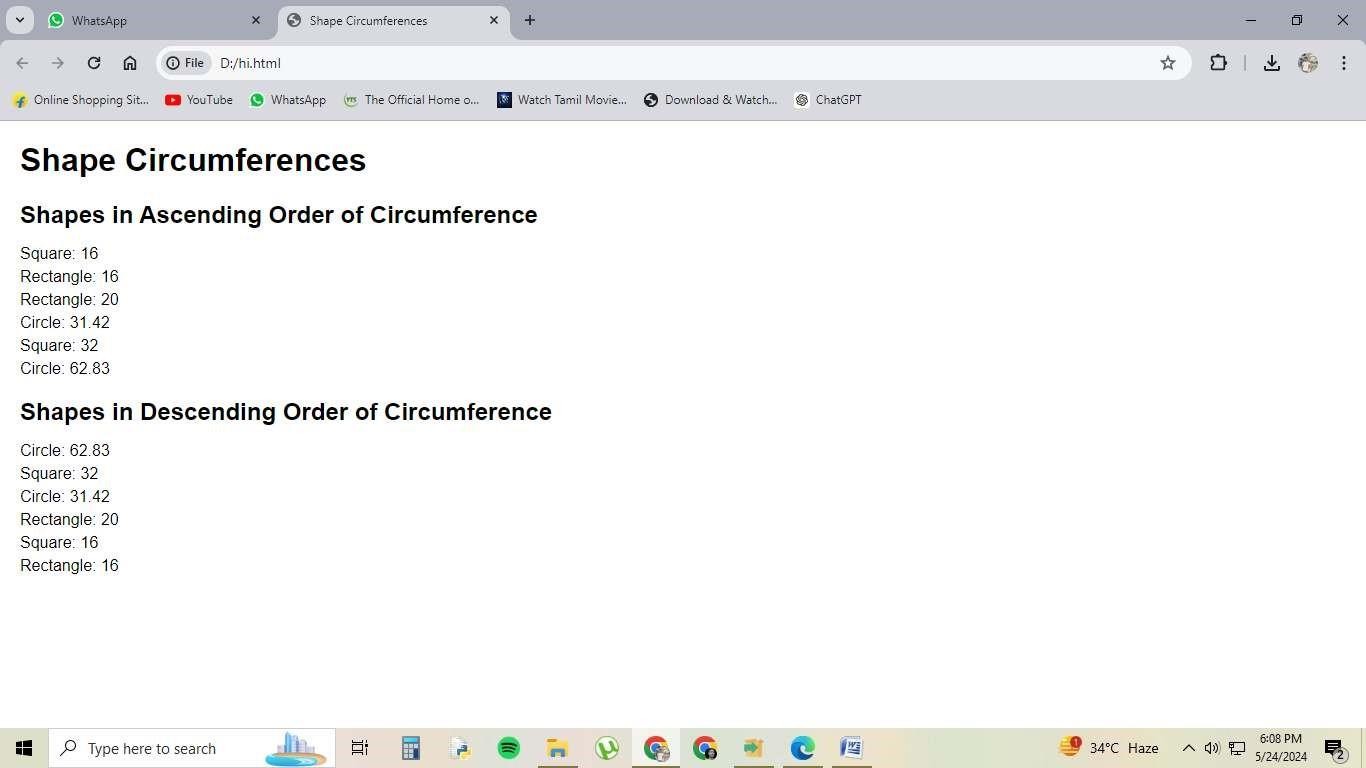
descendingcontainer.appendChild(div);

});

</script>

</body>

</html>

**OUTPUT:**

**RESULT:**

Thus an of interactive web sites – Design using HTML and authoring tools, the shapes of circle,

rectangle and square are arranged according to their circumference successfully.

|  |  |
| --- | --- |
| **EX NO :02** | **“Form validation Using Java Script”** |
| **DATE:** |

# AIM:

The aim of this code is to provide form validation for a simple HTML form containing fields for name and email. The validation ensures that the user enters valid data before submitting the form

# ALGORITHM:

1. HTML Structure:
   * Creating the basic HTML framework.
   * Create a <form> element to hold the review form.
2. Form Fields:
   * Inside the <form> tag, add input fields for name, email, rating, and comments.
   * Use proper input types: text, email, and number.
   * Add labels for accessibility.
3. CSS Styling:
   * Style the form: input, labels, and buttons.
   * Set width, padding, margins, borders, and colors
4. JavaScript Validation:
   * Write a JavaScript function to validate the form.
   * This function check if all required fields are filled.
   * Ensure email format and display error messages if needed.
5. Success Message:
   * Create a <div> element to display a success message.
   * Update the JavaScript function to show this on successful submission.
6. Testing:
   * Test form validation and success message.
   * Ensure responsiveness on various devices.
7. Final Adjustments:
   * Make any adjustments based on feedback and testing.

# PROGRAM:

# HTML (index.html):

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>form validation</title>

<style>

.error{color: red;}

</style>

</head>

<body>

<form id="myform" onsubmit="return validateform()">

<label for="username" >Username:</label>

<input type="text" id="username" name="username" ><br><br>

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

<input type="email" id="email" name="email" ><br><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" ><br><br>

<label for="age">Age:</label>

<input type="number" id="age" name="age" ><br><br>

<input type="submit" id="submit" value="submit">

</form>

<script src="form-validation.js"></script>

</body>

</html>

**JavaScript (form-validation.js):**

function validateform(){

const errorElements=document.querySelectorAll('.error');

errorElements.forEach(element => element.remove());

let isvalid=true;

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

if(username === ''){

showError('username','username is required');

isvalid=false;

}else if(username.length<3){

showError('username','username must be at least 3 characters');

isvalid=false;

}

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

if(email === ''){

showError('email','email is required');

isvalid=false;

}else if(!validateEmail(email)){

showError('email','please enter valid emailaddress');

isvalid=false;

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

if(pass === ''){

showError('password','password is required');

isvalid=false;

}else if(pass.length<6){

showError('password','password must be at least 6 characters');

isvalid=false;

}

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

if(age === ''){

showError('age','age is required');

isvalid=false;

}else if(age<18 || age>100){

showError('age','age must between 18 and 100');

isvalid=false;

}

if(isvalid==true){

window.alert('form submited scussfully');

};

return isvalid;

};

function showError(fieldId,message){

const field=document.getElementById(fieldId);

const error=document.createElement('div');

error.className='error';

error.innerText=message;

field.parentNode.insertBefore(error,field.nextSibling);

};

function validateEmail(email){

const re=/^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/;

return re.test(String(email).toLowerCase());

};

# OUTPUT:

# 

# 

**RESULT:**

Thus, form validation for a simple HTML form has been successfully implemented

|  |  |
| --- | --- |
| **EX NO :03** | **Creation of simple PHP scripts** |
| **DATE:** |

# AIM:

The aim of this code is to create a simple PHP scripts.

# ALGORITHM:

1. Environment Setup:
   * Install a web server like XAMPP.
   * Start the server.
2. Create a PHP Script:
   * Open a text editor and create a .php file (e.g., index.php).
3. Write Basic PHP Code:
   * Write PHP code inside <?php and end with?> tags.
4. Run the PHP Script:
   * Place the PHP file in server's root directory (e.g., htdocs).
   * Open a web browser and go to “http://localhost/index.php”.
5. Test and Debug:
   * Access the script via browser and debug any issues.
6. Iterate and Improve:
   * Refactor code and add new features as needed.

# PROGRAM:

# PHP (index.php):

<?php session\_start(); ?>

<html>

<head>

<script src="valid.js"></script>

</head>

<body>

<div id="compform">

<form method="post" action="result.php" name="comp" onsubmit="return compValidate()">

<h1>Complaint Form</h1>

<input type="text" name="fname" class="firstname" placeholder="First Name">

<input type="text" name="lname" class="lastname" placeholder="Last Name"><br>

<input type="text" name="email" class="email" placeholder="Email"><br>

<select name="device" class="device">

<option value="" selected>Select your Device</option>

<option value="mobile">Mobile Phone</option>

<option value="tv">Television</option>

<option value="computer">Computer/Laptop</option>

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

</select>

<div class="warranty">

<p>Is your device under warranty?</p>

<input type="radio" name="warranty" value="yes"> <label>Yes</label>

<input type="radio" name="warranty" value="no"> <label>No</label><br>

</div>

<button type="submit">Submit</button>

</form>

</div>

</body>

</html>

# JavaScript (valid.js):

function compValidate() {

var fname = document.forms["comp"]["fname"].value;

if (fname == "") {

alert("Please enter your first name");

return false; }

return true;}

**PHP (result.php):**

<?php

$fname = $\_POST['fname'];

$lname = $\_POST['lname'];

$email = $\_POST['email'];

$device = $\_POST['device'];

$warranty = $\_POST['warranty'];

echo "Complaint Form Submitted:<br>";

echo "First Name: $fname<br>";

echo "Last Name: $lname<br>";

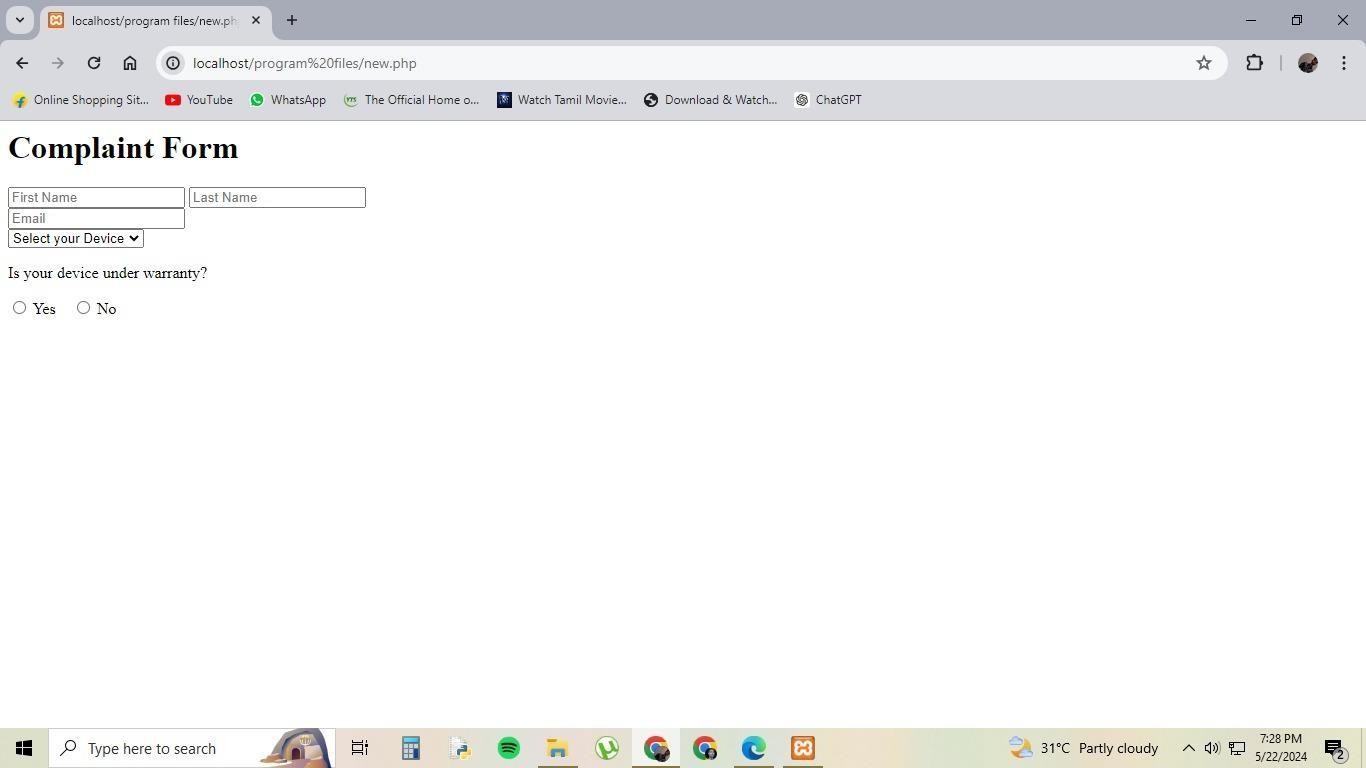
echo "Email: $email<br>";

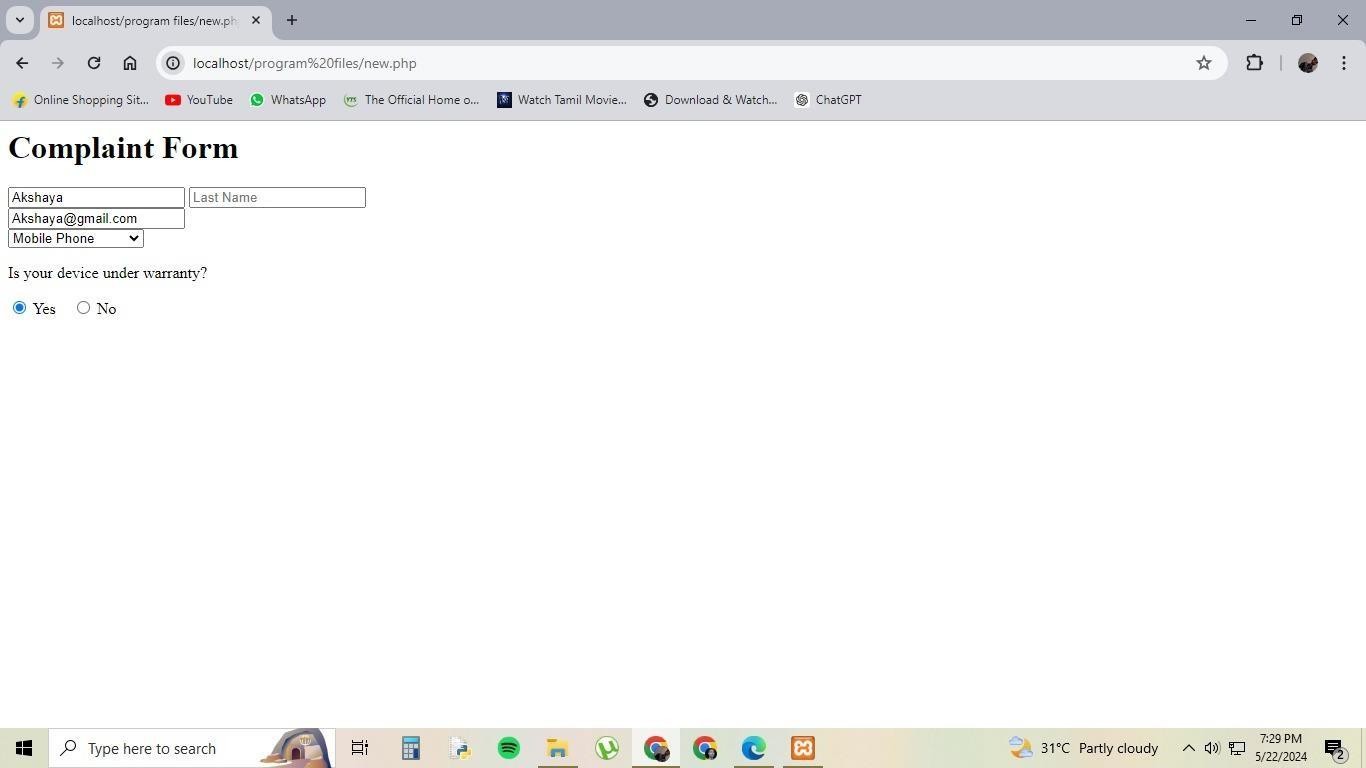
echo "Device: $device<br>";

echo "Warranty: $warranty<br>";

?>

# OUTPUT:





**RESULT:**

Thus, the creation of simple PHP scripts has been successfully implemented.

|  |  |
| --- | --- |
| **EX NO :04** | **Handling Multimedia contents in websites** |
| **DATE:** |

# AIM:

Creating a code for handling a multimedia content in websites.

# ALGORITHM:

Step 1: Content Preparation

* Source Content: Gather files and ensure legal compliance.
* File Format Selection: Choose web-friendly formats like JPEG, PNG, WebP, MP4, MP3, AAC.

Step 2: Encoding and Compression

* Image Optimization: Use tools like ImageMagick, TinyPNG for compression.
* Video and Audio Encoding: Use FFmpeg for encoding, generate multiple resolutions for videos

Step 3: File Management and Hosting

* File Naming: Use descriptive names and organize files into directories.
* Hosting: Use Content Delivery Network (CDN) and ensure the supports HTTP/2.

Step 4: Responsive Design and Integration

* Images: Use <picture> and srcset, implement CSS media queries.
* Videos: Use <video> with multiple sources, ensure responsive design with CSS.

Step 5: Accessibility and SEO

* Accessibility: Provide alt text, captions and transcripts.
* SEO Optimization: Use descriptive filenames, metadata, and structured data.

Step 6: Performance Optimization

* Lazy Loading: Implement for images and videos.
* Caching: Set cache headers and file versioning.

Step 7: Testing and Monitoring

* Cross-Browser Testing: Ensure compatibility using tools like BrowserStack.
* Performance Monitoring: Using tools like Google Lighthouse, gather feedback for improvements.

# PROGRAM:

**HTML (index.html):**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Doraemon Entertainment Show</title>

<style>

\*{

background-color: lightskyblue;

}

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

padding: 20px;

text-align: center;

}

main {

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

}

section {

padding: 20px;

margin-bottom: 20px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

h1, h2 {

color: #333;

margin-bottom: 10px;

}

#gallary {

display: flex;

flex-wrap: wrap;

justify-content: center;

}

#gallary img {

width: 30%;

margin: 10px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(245, 88, 88, 0.685);

}

#comment-section {

padding: 20px;

}

#comment-form {

display: flex;

flex-direction: column;

align-items: center;

}

#comment-input {

width: 80%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

}

#submit-button {

background-color: #333;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 10px;

cursor: pointer;

}

#comment-list {

padding: 20px;

}

#comment-list div {

background-color: #f7f7f7;

padding: 10px;

margin-bottom: 10px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

footer {

padding: 10px;

text-align: center;

clear: both;

}</style>

</head>

<body>

<header>

<h1>DORAEMON ENTERTAINMENT SHOW</h1>

</header>

<main>

<section id="video-screen">

<h2>DORAEMON EPISODE</h2>

<video id="featured-video" controls autoplay muted>

<source src="Doraemon title song in tamil.mp4" type="video/mp4">

Your browser does not support the video tag.

</video>

</section>

<section id="audio-section">

<h2>DORAEMON THEME SONG</h2>

<audio id="featured-audio" controls autoplay>

<source src="Doraemon title song in tamil.mp3" type="audio/mp3">

Your browser does not support the audio tag.

</audio>

</section>

<section id="image-section">

<h2>DORAEMON IMAGES</h2>

<div class="gallery">

<img src="doraemon1.jpeg" alt="Doraemon Image 1" loading="lazy">

<img src="doraemon2.jpeg" alt="Doraemon Image 2" loading="lazy">

<img src="doraemon3.jpeg" alt="Doraemon Image 3" loading="lazy">

</div>

</section>

<section id="comment-section">

<h2>COMMENTS</h2>

<form id="comment-form">

<textarea placeholder="Leave a comment" id="comment-input"></textarea>

<button type="submit">Submit</button>

</form>

<div id="comment-list"></div>

</section>

</main>

<footer>

<p>&copy; 2024 Entertainment Show. All rights reserved</p>

</footer>

<script >

const commentForm = document.getElementById('comment-form');

const commentList = document.getElementById('comment-list');

commentForm.addEventListener('submit', (e) => {

e.preventDefault();

const commentInput = document.getElementById('comment-input');

const commentText = commentInput.value.trim();

if (commentText !== '') {

const commentElement = document.createElement('div');

commentElement.textContent = commentText;

commentList.appendChild(commentElement);

commentInput.value = '';

}});

</script>

</body>

</html>

# OUTPUT:

# 

# 

**RESULT:**

Thus, the creation of multimedia contents in websites has been successfully implemented.

|  |  |
| --- | --- |
| **EX NO :05** | **WRITE PROGRAMS USING SERVELETS** |
| **DATE:** |

**AIM:**

To write programs using servelets 1) to invoke servelets from html forms 2)session tracking using hidden form fields and Session tracking for hit counts.

**ALGORITHM:**

To invoke a servlet from an HTML form:

1. Display HTML Form:

* Create an HTML form with the necessary input fields.
* Set the form action to servlet URL.
* Specify the appropriate HTTP method (POST or GET).

1. Submit Form Data:

* user submits the form.
* The browser sends an HTTP request with form data.

1. Servlet Handling the Request:

* Servlet receives HTTP request.
* Use request.getparameter() to retrieve form data.

1. Process Form Data:

* Process data in doPost() or doGet().
* Implement business logic or database operations.

1. Generate Response:

* Generate the HTML response.
* Use `response.getWriter()` to write HTML content.

1. Send Response to Client:

* Servlet sends response back to the client (browser).
* The browser renders the HTML response.

Session Tracking Using Hidden Form Fields:

1. Display Form Page:

* HTML form with a hidden input field for session ID and a text input field for the user’s name.
* Set the form action to HiddenFieldServlet.

1. Submit Form Data:

* Form submission sends POST request to HiddenFieldServlet.

1. HiddenFieldServlet:

* Retrieve session ID and user name.
* Display a greeting message with user name and session ID.

Session Tracking for Hit Count:

1. HitCountServlet Handling

* Retrieve session object and hit count.
* Initialize hit count if not present.
* Increment hit count and Store in session.
* Display current hit count.

**PROGRAM:**

**HTML (index.html):**

<!DOCTYPE html>

<html>

<head>

<title>Servlet Form</title>

</head>

<body>

<h2>Enter your name:</h2>

<form action="HelloServlet" method="post">

<input type="text" name="name">

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

</form>

</body>

</html>

**Servlet (HelloServlet.java):**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@webServlet(“/HelloServlet”)

public class HelloServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) Throws ServletException, IOException {

response.setContentType(“text/html”);

PrintWriter out = response.getWriter();

String name = request.getParameter(“name”);

out.println(“<html>”); Out.println(“<head><title>HelloServlet</title></head>”); out.println(“<body>”);

out.println(“<h1>Hello, “+ name + “!</h1>”);

out.println(“</body></html>”);

* + **Session Tracking Using Hidden Form FieldsHTML form (sessionForm.html):**

<!DOCTYPE html>

<html>

<head>

<title>Session Tracking Using Hidden Form Fields</title>

</head>

<body>

<h2>Enter your name:</h2>

<form action=”HiddenFieldServlet” method=”post”>

<input type=”hidden” name=”sessionID” value=”${pageContext.session.id}”>

<input type=”text” name=”name”>

<input type=”submit” value=”Submit”>

</form>

</body>

</html>

Servlet (HiddenFieldServlet.java):

import java.io.\*;

import javax.servlet.\*; import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@webSocket(“/HiddenFieldServlet”)

public class HiddenFieldServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) Throws ServletException, IOException {

response.setContentType(“text/html”);

PrintWriter out = response.getWriter();

// Retrieve the session ID from the hidden form field

String sessionID = request.getParameter(“sessionID”);

// Retrieve other form parameters

String name = request.getParameter(“name”);

out.println(“<html>”);

out.println(“<head><title>Session Tracking Using Hidden Form Fields</title></head>”);

out.println(“<body>”); out.println(“<h2>Hello, “ + name + “!</h2>”);

out.println(“<p>Your session ID: “ + sessionID + “</p>”); out.println(“</body></html>”); } }

**2.)Session Tracking for Hit CountServlet (HitCountServlet.java):**

import java.io.\*;

import javax.servlet.\*; import javax.servlet.http.\*;

public class HitCountServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) Throws ServletException, IOException {

// Increment hit count

HttpSession session = request.getSession();

Integer count = (Integer) session.getAttribute(“hitCount”);

if (count == null) {

count = 1;

} else {

count++;

}

session.setAttribute(“hitCount”, count)

// Send response

response.setContentType(“text/html”);

PrintWriter out = response.getWriter();

out.println(“<html><head><title>Hit Count</title></head><body>”);

out.println(“<h2>Hit count: “ + count + “</h2>”);

out.println(“</body></html>”);

}

}

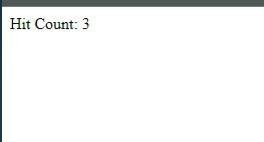
**OUTPUT:**

1. **To Invoke servelets for an html form:**





1. **sevelets for session tracking and hit count:**



**RESULT:**

Thus the programs ha been done and the output has been verified successfully.

|  |  |
| --- | --- |
| **EX NO :06** | **“Creation of information retrieval system using web, PHP and mysql”** |
| **DATE:22.04.24** |

# AIM:

The aim of this code is to create a information retrieval system using web,php,and mysql.

# ALGORITHM:

User Registration

* + 1. Input: username, password
    2. Process:
* Connect to the database.
* Hash the password using bcrypt.
* Insert username and hashed password into users table.
  + 1. Output: Success or error message.

User Login

1. Input: username, password
2. Process:

* Connect to the database.
* select id and password from users where username matches.
* Verify password against stored hash.
* Start session if valid.

1. Output: Success or error message.

Document Upload

1. Input: title, content
2. Process:

* Check if the user is logged in.
* Get the session user ID.
* Connect to the database.
* Insert title, content, and user\_id into the documents table.

1. Output: Success or error message.

TF-IDF Search Functionality

1. Input: query
2. Process:

* Check if the user is logged in.
* Get the session user ID.
* Connect to the database.
* Insert the query and user id into search\_queries.
* Split the query into terms.
* Retrieve all documents from the documents table.
* Calculate TF-IDF scores, sort by scores in descending order

1. Output: Display ranked documents or “No results found”.

Navigation and Session Management

1. Input: Navigation request
2. Process:

* Check if the user is logged in.
* Display links based on login status.

# PROGRAM:

**SQL :**

1. CREATE DATABASE ir\_system;

2. USE ir\_system;

3. CREATE TABLE users (id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL, password VARCHAR(255) NOT NULL );

4. CREATE TABLE documents (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL, content TEXT NOT NULL );

5. CREATE TABLE search\_queries ( id INT AUTO\_INCREMENT PRIMARY KEY, query VARCHAR(255) NOT NULL, user\_id INT, FOREIGN KEY (user\_id) REFERENCES users(id) );

**### Step 2: User Registration**

**PHP (register.php):**

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "ir\_system";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$username = $\_POST['username'];

$password = password\_hash($\_POST['password'], PASSWORD\_BCRYPT);

$sql = "INSERT INTO users (username, password) VALUES ('$username', '$password')";

if ($conn->query($sql) === TRUE) {

echo "Registration successful.";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

$conn->close();

?>

<form method="POST">

Username: <input type="text" name="username" required><br> Password: <input type="password" name="password" required><br>

<input type="submit" value="Register">

</form>

**### Step 3: User Login**

**PHP (login.php):**

<?php session\_start();

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "ir\_system";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$username = $\_POST['username'];

$password = $\_POST['password'];

$sql = "SELECT \* FROM users WHERE username='$username'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

$user = $result->fetch\_assoc();

if (password\_verify($password, $user['password'])) {

$\_SESSION['user\_id'] = $user['id']; header("Location: upload.php");

} else {

echo "Invalid password.";

}

} else {

echo "No user found.";

}

}

$conn->close();

?>

<form method="POST">

Username: <input type="text" name="username" required><br> Password: <input type="password" name="password" required><br>

<input type="submit" value="Login">

</form>

**### Step 4: Document Upload (upload.php)**

**PHP**

<?php session\_start();

if (!isset($\_SESSION['user\_id'])) { header("Location: login.php"); exit();

}

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "ir\_system";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$title = $\_POST['title'];

$content = $\_POST['content'];

$sql = "INSERT INTO documents (title, content) VALUES ('$title', '$content')";

if ($conn->query($sql) === TRUE) { echo "Document uploaded successfully.";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

$conn->close();

?>

<form method="POST">

Title: <input type="text" name="title" required><br> Content: <textarea name="content" required></textarea><br>

<input type="submit" value="Upload">

</form>

**### Step 5: Search Functionality(search.php)**

**PHP**

<?php session\_start();

if (!isset($\_SESSION['user\_id'])) { header("Location: login.php"); exit();

}

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "ir\_system";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$query = $\_POST['query'];

$user\_id = $\_SESSION['user\_id'];

$sql = "INSERT INTO search\_queries (query, user\_id) VALUES ('$query', '$user\_id')";

$conn->query($sql);

$search\_sql = "SELECT \* FROM documents WHERE title LIKE '%$query%' OR content LIKE '%$query%'";

$result = $conn->query($search\_sql);

if ($result->num\_rows > 0) { while ($row = $result->fetch\_assoc()) {

echo "<h2>Title: " . htmlspecialchars($row['title']) . "</h2>"; echo "<p>Content: " . nl2br(htmlspecialchars($row['content'])) . "</p>";

}

} else {

echo "No results found.";

}

}

$conn->close();

?>

<form method="POST">

Search: <input type="text" name="query" required><br>

<input type="submit" value="Search">

</form>

**### Step 6: Navigation (index.php)**

**php**

<?php session\_start();?>

<!DOCTYPE html>

<html>

<head>

<title>Information Retrieval System</title>

</head>

<body>

<h1>Welcome to the Information Retrieval System</h1>

<ul>

<?php if (isset($\_SESSION['user\_id'])): ?>

<li><a href="upload.php">Upload Document</a></li>

<li><a href="search.php">Search Documents</a></li>

<li><a href="logout.php">Logout</a></li>

<?php else: ?>

<li><a href="register.php">Register</a></li>

<li><a href="login.php">Login</a></li>

<?php endif; ?>

</ul>

</body>

</html>

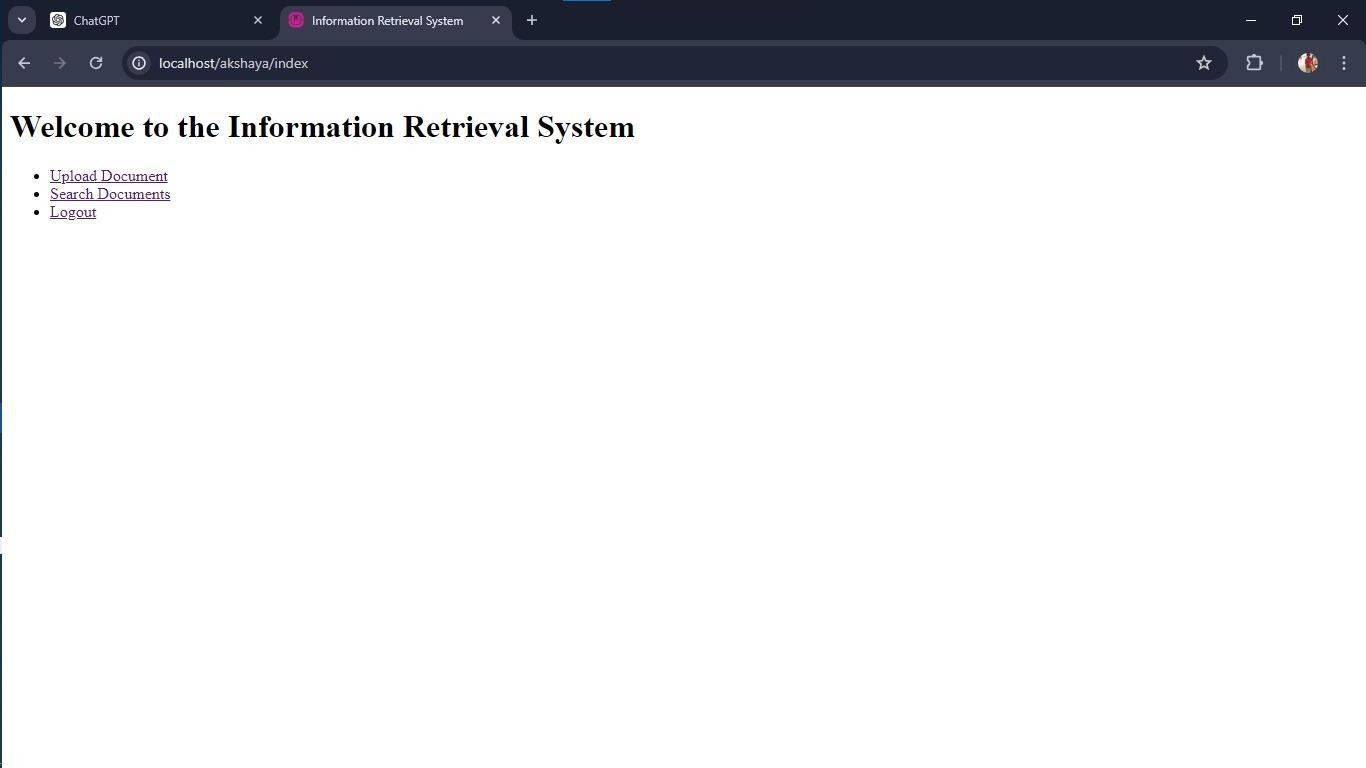
**###Step 7: Logout (logout.php)**

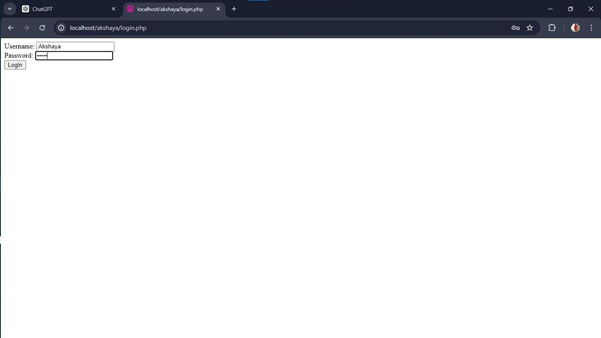
**PHP**

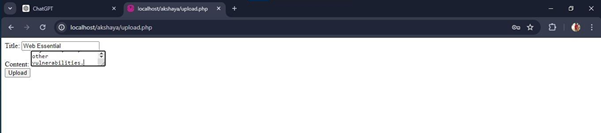
<?php

session\_start(); session\_destroy(); header("Location: index.php"); exit(); ?>

**OUTPUTS:**







# 

# 

# RESULT:

Thus, the creation of information retrieval system using web php and mysql is successfully implemented.

|  |  |
| --- | --- |
| **EX NO :07** | **Creation of personal information system** |
| **DATE:** |

# AIM:

The aim of this code is creation of personal information system.

# ALGORITHM:

1. Requirements Gathering and Analysis
   * Identify users and define system scope.
   * Collect detailed requirements.
   * Prioritize and validate the requirements.
2. System Design
   * Choose architecture.
   * Design database schema.
   * Create UI wireframes and mockups.
   * Select technology stack.
3. Development
   * Setup Development Environment.
   * Develop database and server-side logic.
   * Build user interface.
   * Integrate frontend and backend services.
4. Testing
   * Perform unit test, integration, and system testing.
   * Conduct User Acceptance Testing (UAT).
5. Deployment
   * Prepare and set up production environment.
   * Deploy application.
   * Perform final testing.
6. Maintenance and Updates
   * Monitor system performance.
   * Fix bugs and implement enhancements.
   * Keep the system updated.

# PROGRAM:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Information Management System</title>

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

</head>

<body>

<div class="container">

<h1>Personal Information Management System</h1>

<div class="form-section">

<h2>Add New Person</h2>

<form id="personForm">

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

<input type="text" id="name" name="name" required>

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

<input type="email" id="email" name="email" required>

<label for="phone">Phone:</label>

<input type="text" id="phone" name="phone" required>

<button type="submit">Add Person</button>

</form>

</div>

<div class="list-section">

<h2>People List</h2>

<table id="peopleTable">

<thead>

<tr>

<th>Name</th>

<th>Email</th>

<th>Phone</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<!-- People will be listed here -->

</tbody>

</table>

</div>

</div>

<script src="script.js"></script>

</body>

</html>

**CSS (style.css):**

body {

font-family: Arial, sans-serif;

text-align: center;}

#user-form {

width: 50%;

margin: 40px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);}

button{background: lightgray;}

input{

background:#fff;}

#user-list {

list-style: none;

padding: 0;

margin: 0;}

#user-list li {

padding: 10px;

border-bottom: 1px solid #ccc;}

#user-list li:last-child {

border-bottom: none;}

table{

padding: 5px;

margin: 0 auto;}

\*{background:skyblue;}

**JavaScript (script.js):**

const form = document.getElementById('user-form');

const userlist = document.getElementById('user-list');

form.addEventListener('submit', (e) => {

e.preventDefault();

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

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

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

const user = { name, email, phone };

const listItem = document.createElement('tr');

listItem.innerHTML = `<td>${user.name}</td><td>${user.email}</td>

<td>${user.phone}</td><td><button class="delete">Delete</button>`;

userlist.appendChild(listItem);

const deleteBtn = listItem.querySelector('.delete');

deleteBtn.addEventListener('click', (e) => {

const row = deleteBtn.parentNode.parentNode;

row.remove();

});

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

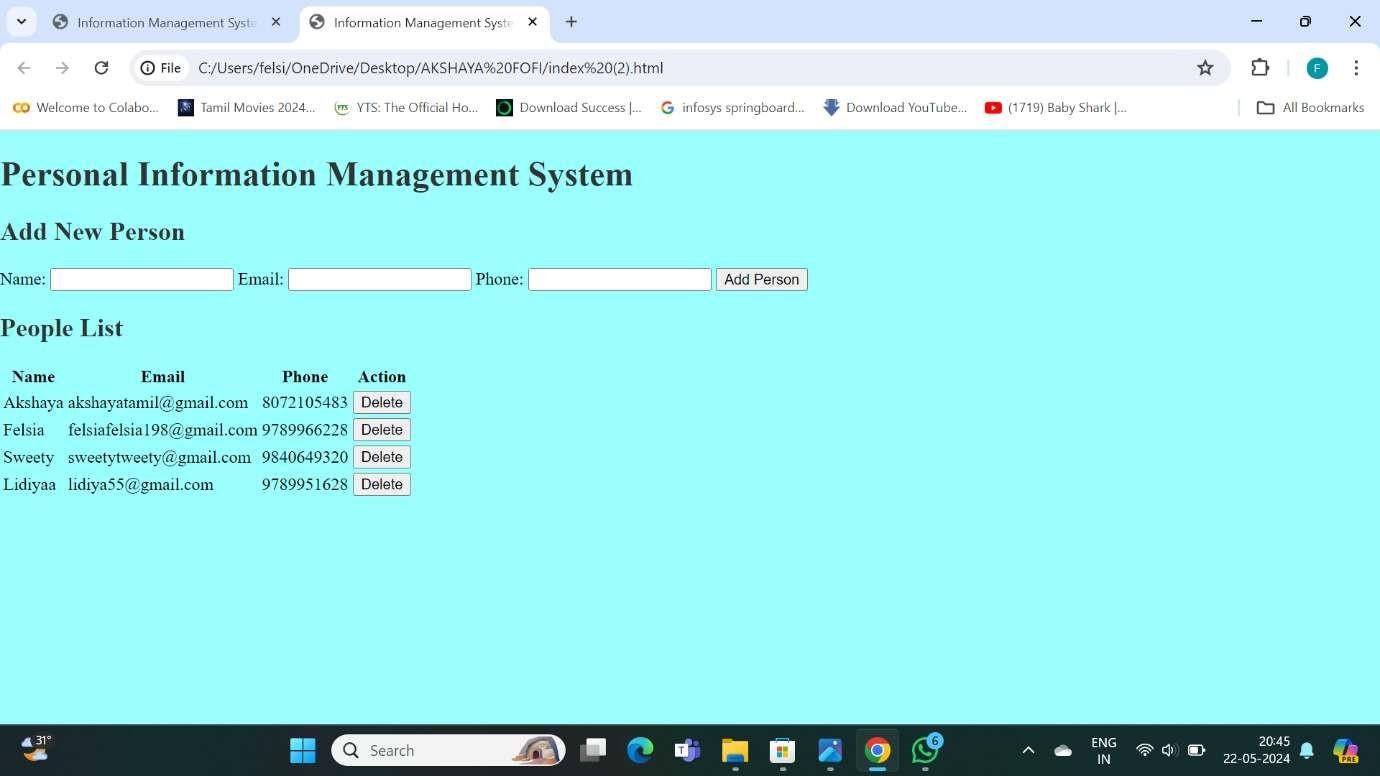
document.getElementById('email').value = '';

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

});

# OUTPUT:

# 



**RESULT:­­­­­­­­**

Thus, the creation of personal information system is implemented.