Affiliated to Tribhuvan University (TUFOHSS)

Nihareeka College of Management and Information Technology

Biratnagar, Morang

An Assignment

On

Web Technology

CACS-205

Submitted By:

Garima Niraula

Semester-III

Bachelor in Computer Application (BCA)

Submitted To:

Nihareeka College of Management and Information Technology

Department of Information Technology

2081

Table of Contents

List of Figures:

HTML Fundamentals

1.1 Objective:
Gain an understanding of basic HTML tags and how to use them practically.

1.2 Overview:
HTML (Hypertext Markup Language) is the foundational language used to build webpages. Rather than programming behavior, it organizes and structures content.

Key points about HTML:

- It stands for HyperText Markup Language.
- It's the standard for creating web pages.
- HTML defines the structure and layout of a webpage.
- It uses elements to indicate how content should be displayed.
- These elements label parts of content, such as headings, paragraphs, and links.

1.2.1 Common HTML Tags:

- `<!DOCTYPE html>` – Declares the document type as HTML5.
- `<html>` – The root element that wraps the entire page content.
- `<head>` – Holds metadata like the page title, stylesheets, and scripts.
- `<title>` – Sets the name shown on the browser tab.
- `<body>` – Contains all the visible page content.
- `<h1>` to `<h6>` – Headings, with `<h1>` being the largest.
- `<p>` – Denotes a paragraph.
- `<a href="https://example.com">Click me</a>` – Creates a clickable hyperlink.
- `<nav>` – Groups together navigation links.
- `<ul>` – Represents a bullet-point list.
- `<ol>` – Represents a numbered list.
- `<li>` – A list item within `<ul>` or `<ol>`.
- `<img src="image.jpg" alt="A beautiful image">` – Embeds an image.
- `<video>` / `<audio>` – Embed multimedia like videos and audio clips.
- `<form>` – Used for collecting user input.
- `<input>` – Defines input controls such as text fields, buttons, and checkboxes.

1.4 Attributes: Enhancing HTML Elements

- `href` (used in links) – Specifies the link's destination.
- `src` (used in images/videos) – Indicates the file's location.
- `alt` (used in images) – Describes the image if it can't be displayed.
- `id` – Assigns a unique identifier to an element.
- `class` – Allows grouping elements for styling or scripting.

basic-page.html

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Page Title</title>
</head>
<body>

    <h1>My First Heading</h1>
    <p>My first paragraph.</p>

    <!-- Navigation -->
    <nav>
        <a href="home.html">Home</a> <br>
        <a href="about.html">About</a>
    </nav>

    <!-- Unordered List -->
    <ul>
        <li>Apples</li>
        <li>Bananas</li>
    </ul>

    <!-- Image -->
    <img src="ab232473-5137-4ed9-9820-3b059c262f10.png" alt="A stunning
sunset">

    <!-- Form -->
    <form>
        <input type="text" placeholder="Garima Niraula">
        <button>Submit</button>
    </form>

</body>
</html>
```

Output:

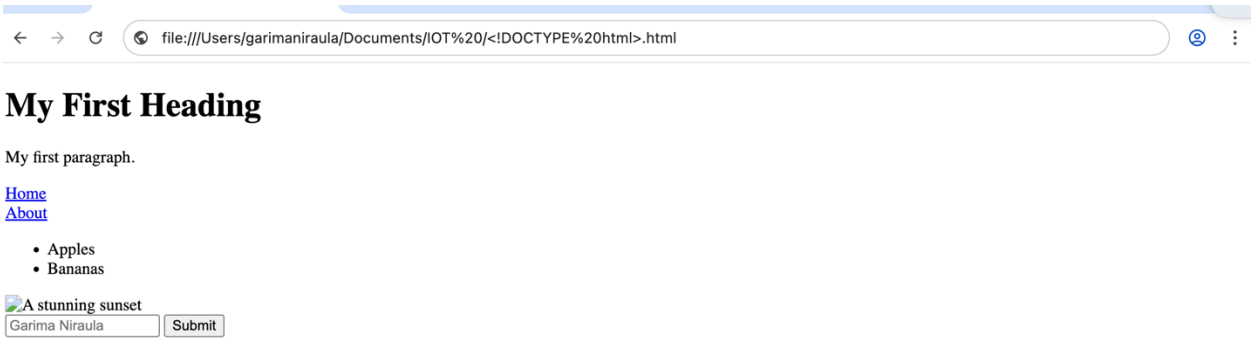

Figure 1: HTML Basics

1.7 Conclusion:
The foundational structure of an HTML document was successfully developed and executed. This exercise demonstrated a solid grasp of essential web development concepts. The webpage rendered correctly in the browser, confirming the accurate use of HTML elements and basic styling to create a well-organized layout.

2.1 Objective

To design and implement a clean and responsive school schedule using HTML for structure and CSS for styling.

2.2 Tasks Performed

- Constructed an HTML-based table to neatly organize the weekly class schedule.
- Styled the table using CSS to enhance clarity, visual design, and responsiveness.
- Utilized semantic tags such as `<th>` for headers and `<td>` for content cells.
- Applied styling elements including borders, spacing, and background colors for improved presentation.
- Successfully rendered and tested the layout in a web browser, verifying structure and design.

Source Code:

index.html(Base Code)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Weekly Class Schedule</title>
    <link rel="stylesheet" href="schedule.css">
</head>
<body>
    <h2>Weekly Class Schedule</h2>
    <table>
        <tr>
            <th>Day/Session</th>
            <th>Sunday</th>
            <th>Monday</th>
            <th>Tuesday</th>
            <th>Wednesday</th>
            <th>Thursday</th>
            <th>Friday</th>
            <th>Saturday</th>
        </tr>
        <tr>
            <td>Session 1</td>
            <td>English</td>
            <td>English</td>
            <td>English</td>
```

```
        <td>English</td>
        <td>English</td>
        <td>English</td>
        <td rowspan="10" class="weekend">Holiday</td>
    </tr>
    <tr>
        <td>Session 2</td>
        <td>Grammar</td>
        <td>Grammar</td>
        <td>Grammar</td>
        <td>Grammar</td>
        <td>Grammar</td>
        <td>Grammar</td>
    </tr>
    <tr class="break-row">
        <td>Tea Break</td>
        <td colspan="6">Refreshment Time</td>
    </tr>
    <tr>
        <td>Session 3</td>
        <td>Math</td>
        <td>Math</td>
        <td>Math</td>
        <td>Math</td>
        <td>Math</td>
        <td>Math</td>
    </tr>
    <tr>
        <td>Session 4</td>
        <td>Science</td>
        <td>Science</td>
        <td>Science</td>
        <td>Science</td>
        <td>Science</td>
        <td>Science</td>
    </tr>
    <tr class="break-row">
        <td>Lunch</td>
        <td colspan="6">Midday Break</td>
    </tr>
    <tr>
        <td>Session 5</td>
        <td>Health</td>
        <td>Health</td>
        <td>Health</td>
        <td>Health</td>
        <td>Health</td>
        <td>-</td>
    </tr>
    <tr class="break-row">
        <td>Short Break</td>
        <td colspan="6">Stretch & Relax</td>
    </tr>
    <tr>
        <td>Session 6</td>
        <td>Nepali</td>
        <td>Nepali</td>
```

```
            <td>Nepali</td>
            <td>Nepali</td>
            <td>Nepali</td>
            <td>-</td>
        </tr>
        <tr>
            <td>Session 7</td>
            <td>Computer</td>
            <td>Computer</td>
            <td>Computer</td>
            <td>Computer</td>
            <td>Computer</td>
            <td>-</td>
        </tr>
    </table>
</body>
</html>
```

---

schedule.css (For Layering and Color Structured Format)

```
table {
    width: 100%;
    border-collapse: collapse;
    font-family: Arial, sans-serif;
}

th, td {
    border: 1px solid #ccc;
    padding: 10px;
    text-align: center;
}

th {
    background-color: #e0e0e0;
    font-weight: bold;
}

.break-row {
    background-color: #fff5e6;
    font-style: italic;
}

.weekend {
    background-color: #d4edda;
    font-weight: bold;
    color: #155724;
}
```

Weekly Class Schedule

Day/Session	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Session 1	English	English	English	English	English	English	Holiday
Session 2	Grammar	Grammar	Grammar	Grammar	Grammar	Grammar	
Tea Break	*Refreshment Time*						
Session 3	Math	Math	Math	Math	Math	Math	
Session 4	Science	Science	Science	Science	Science	Science	
Lunch	*Midday Break*						
Session 5	Health	Health	Health	Health	Health	-	
Short Break	*Stretch & Relax*						
Session 6	Nepali	Nepali	Nepali	Nepali	Nepali	-	
Session 7	Computer	Computer	Computer	Computer	Computer	-	

Figure 2: Weekly Class Schedule

2.5 Conclusion:

The school routine table was successfully developed using **HTML and CSS**, ensuring a structured and visually appealing schedule layout. The project highlights the importance of **well-organized tables, responsive design, and CSS styling** in web development.

3.1 User Friendly Form using HTML and CSS

To build a responsive and user-friendly form layout using HTML and CSS.

3.2 Summary of Work

- Developed a well-organized HTML form structure with accessible fields.
- Enhanced form presentation using modern CSS for clean aesthetics.
- Integrated multiple input types and sections including contact info and login.
- Ensured the form worked seamlessly across different browsers and screen sizes.
- Displayed a functional map and contact section at the bottom of the page.

3.3 Source Code:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Elegant User Form</title>
  <link rel="stylesheet" href="form-style.css">
```

```
</head>
<body>
  <div class="container">
    <header>
      <h1>Welcome to Our Portal</h1>
      <p>Please fill out your details or log in below.</p>
    </header>

    <div class="card-group">
      <!-- User Details Form -->
      <div class="card">
        <h2>Register</h2>
        <form>
          <label>Full Name</label>
          <input type="text" placeholder="John Doe" required>

          <label>Email</label>
          <input type="email" placeholder="john@example.com" required>

          <label>Phone Number</label>
          <input type="tel" placeholder="+1-123-456-7890" required>

          <label>Date of Birth</label>
          <input type="date" required>

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

      <!-- Login Form -->
      <div class="card">
        <h2>Login</h2>
        <form>
          <label>Username</label>
          <input type="text" placeholder="Enter your username" required>

          <label>Password</label>
          <input type="password" placeholder="Enter your password" required>

          <button type="submit">Login</button>
        </form>
      </div>
    </div>

    <!-- Contact Info -->
    <footer>
      <div class="contact-info">
        <h3>Contact Us</h3>
        <p> Address: Your Street, City, Country</p>
        <p> Phone: (123) 456-7890</p>
        <p> Email: contact@yourdomain.com</p>
      </div>
      <iframe src="https://www.google.com/maps/embed?pb=!1m14..."
allowfullscreen loading="lazy"></iframe>
    </footer>
  </div>
</body>
```

```
</html>
```

form-style.css

```
body {
  margin: 0;
  padding: 0;
  font-family: 'Segoe UI', sans-serif;
  background: linear-gradient(to right, #e3f2fd, #ffffff);
}

.container {
  max-width: 1200px;
  margin: auto;
  padding: 30px;
}

header {
  text-align: center;
  margin-bottom: 40px;
}

header h1 {
  font-size: 2.5rem;
  color: #0d47a1;
}

header p {
  color: #555;
}

.card-group {
  display: flex;
  gap: 30px;
  flex-wrap: wrap;
  justify-content: center;
}

.card {
  background: white;
  padding: 30px;
  border-radius: 12px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
  width: 100%;
  max-width: 500px;
  transition: transform 0.3s ease;
}

.card:hover {
  transform: translateY(-5px);
}

.card h2 {
  margin-bottom: 20px;
  color: #1a237e;
}
```

```
label {
  display: block;
  margin: 10px 0 5px;
  color: #333;
}

input {
  width: 100%;
  padding: 10px 12px;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 1rem;
  box-sizing: border-box;
}

button {
  margin-top: 20px;
  width: 100%;
  background: #1976d2;
  color: white;
  padding: 12px;
  font-size: 1rem;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  transition: background 0.3s ease;
}

button:hover {
  background: #1565c0;
}

footer {
  background: #0d47a1;
  color: white;
  padding: 40px 20px;
  margin-top: 60px;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  gap: 20px;
}

.contact-info {
  flex: 1;
  min-width: 250px;
}

iframe {
  flex: 1;
  min-width: 300px;
  height: 300px;
  border: none;
  border-radius: 8px;
}
```

Output:

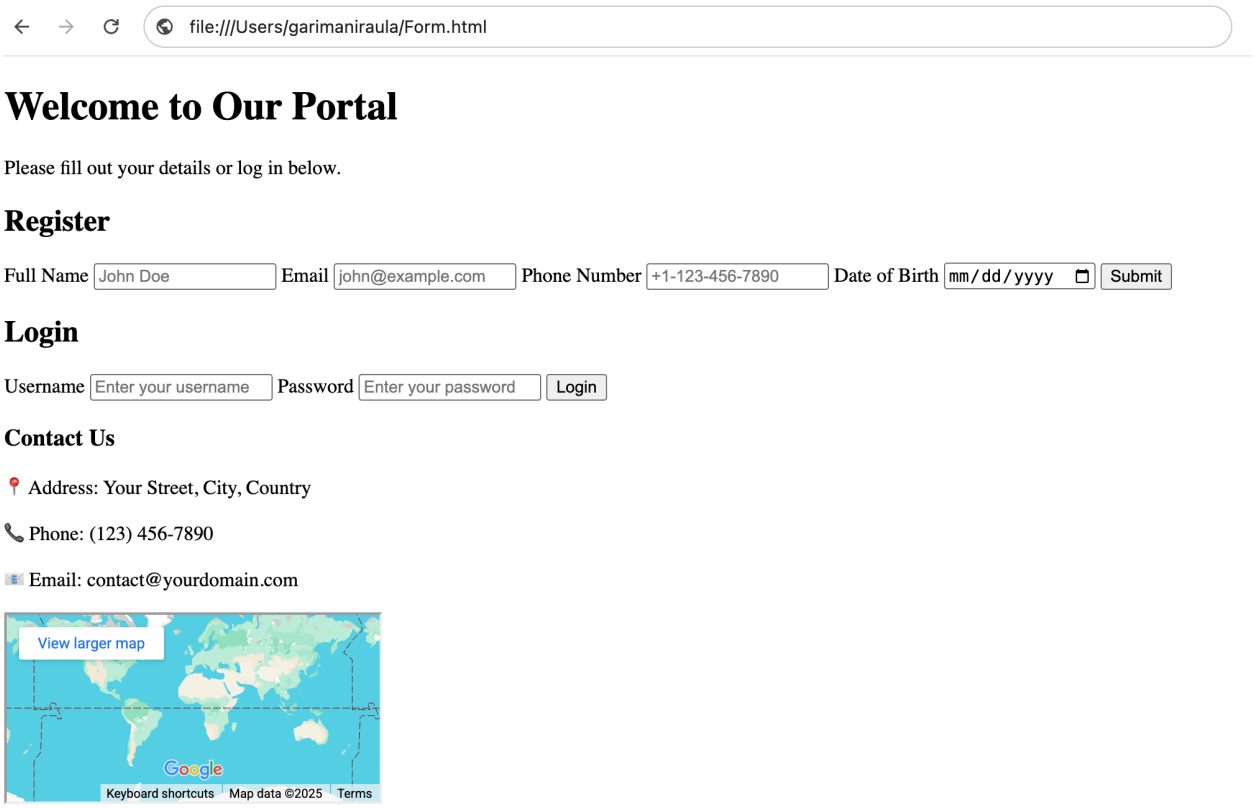

Figure 3: User Friendly Form using HTML and CSS

3.5 Conclusion:

The form development project successfully combined **HTML and CSS** to create a structured, visually appealing, and functional web form. By implementing **responsive design and validation**, the form ensures a seamless user experience across devices. The final output effectively demonstrates the integration of markup and styling for real-world applications.

XAMPP with XML (Extensible Markup Language)

XML (Extensible Markup Language) is a structured, adaptable language used for organizing and transferring data. Unlike HTML—which is used to display information—XML is mainly utilized for representing and exchanging data in a format that's easy for both humans and machines to understand. It's a key tool for data sharing between different systems, especially in web and enterprise environments.

4.1 Objective

To create a structured XML document linked with a DTD, enabling meaningful data representation and transformation.

4.2 Activities Performed

- **Structuring XML Content**: Chose a practical dataset—such as employee details—for demonstration.
- **Creating the XML File**: Wrote a well-structured XML document including fields like employee ID, name, role, department, and salary.
- **Designing the XSL File**: Built an XSL stylesheet to transform the XML into a visually formatted HTML table.
- **Integrating XML and XSL**: Linked the XML document to the XSL using the `<?xml-stylesheet?>` processing instruction.
- **Setting Up the Server**: Installed and configured XAMPP to serve the files locally using Apache.
- **Testing and Rendering**: Verified that the XML document is well-formed and correctly styled via XSL in a browser.

4.3 Source Code

4.3.1 XML Document – `employee.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="employee.xsl"?>
<employees>
    <employee>
        <id>101</id>
        <name>John Doe</name>
        <position>Software Engineer</position>
        <department>IT</department>
        <salary>70000</salary>
    </employee>
    <employee>
        <id>102</id>
        <name>Jane Smith</name>
        <position>HR Manager</position>
        <department>Human Resources</department>
        <salary>65000</salary>
    </employee>
</employees>
```

4.3.2 XSL Stylesheet – `employee.xsl`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
  <xsl:template match="/">
    <html>
      <head>
        <title>Employee List</title>
      </head>
      <body>
        <h2>Employee Directory</h2>
        <table border="1" cellpadding="8" cellspacing="0">
          <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Position</th>
            <th>Department</th>
            <th>Salary</th>
          </tr>
          <xsl:for-each select="employees/employee">
            <tr>
              <td><xsl:value-of select="id"/></td>
              <td><xsl:value-of select="name"/></td>
              <td><xsl:value-of select="position"/></td>
              <td><xsl:value-of select="department"/></td>
              <td><xsl:value-of select="salary"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

4.4 Output:

Figure 4:XML employee content

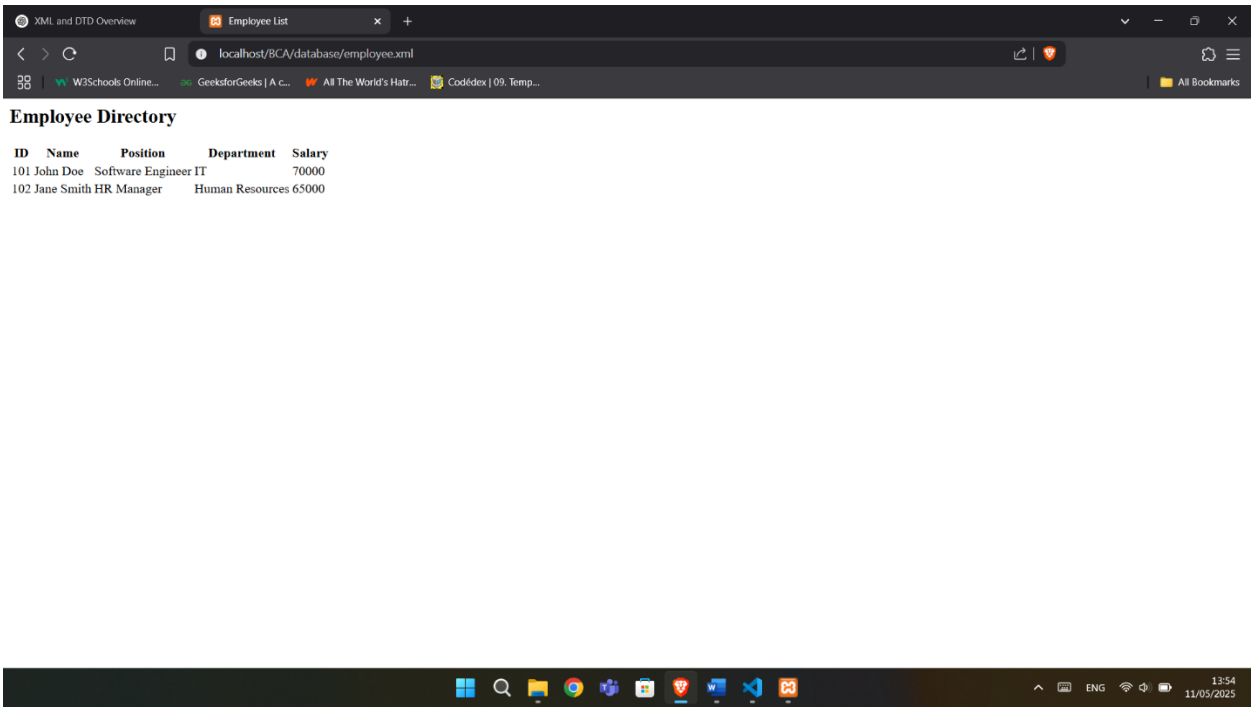

Figure 5: XML setup

Conclusion:

In conclusion, XML provides a powerful and structured way to represent data. By creating and validating XML documents, we maintain data integrity and consistency. This practice is essential for reliable data exchange across applications and systems.

Introduction to PHP

PHP (Hypertext Preprocessor) is a powerful, widely-used open-source scripting language tailored for web development. It operates on the **server side**, meaning the PHP code runs on a web server and returns output to the client's browser. PHP is commonly used to create dynamic webpages, handle form submissions, manage databases, process sessions, and more.

5.1 Objectives

- Grasp the fundamentals of PHP as a server-side scripting language.
- Set up and run PHP files using the **XAMPP** environment.
- Understand key PHP concepts like **variables**, **loops**, and **conditional statements**.
- Learn how to display dynamic content on webpages through PHP scripting.

5.2 Activities

- **Install and launch XAMPP**: Start the Apache server from the XAMPP Control Panel.
- **Create a PHP file**: Go to the `htdocs` directory in your XAMPP installation folder and create a file (e.g., `hello.php`).
- **Write basic PHP code**: Use a text editor like **VS Code** or **Notepad++** to write and save your PHP code.
- **Run the script in a browser**: Open a browser and visit `http://localhost/hello.php` to view the script's output.
- **Experiment with PHP basics**: Practice using `echo`, variables, loops, arrays, conditionals, and functions to deepen your understanding.

5.3 Source Code

5.3.1 Basic Output

```
<?php
echo "Hello, world!";
?>
```

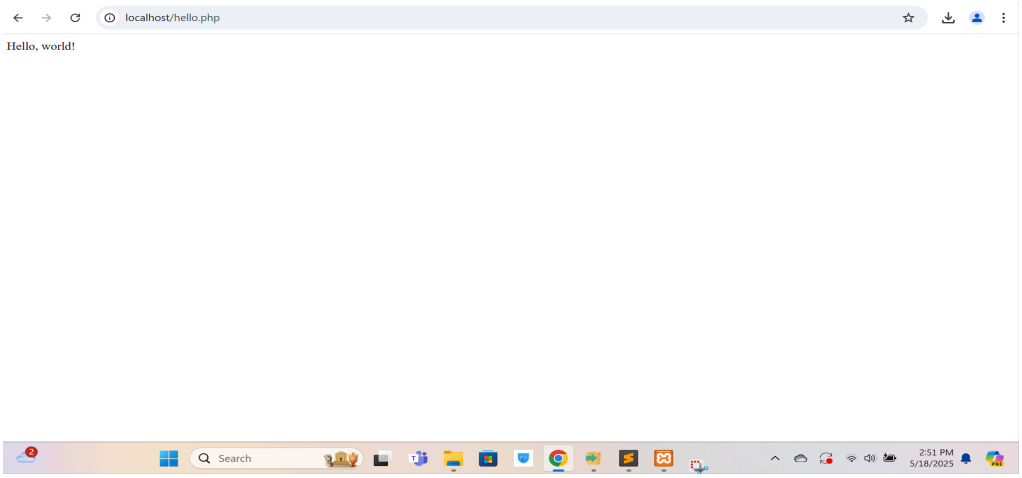

Figure 6: Basic PHP Layout After installing Xampp

5.3.2 Using Variables

```
<?php
$name = "John";
echo "Welcome, " . $name . "!";
?>
```

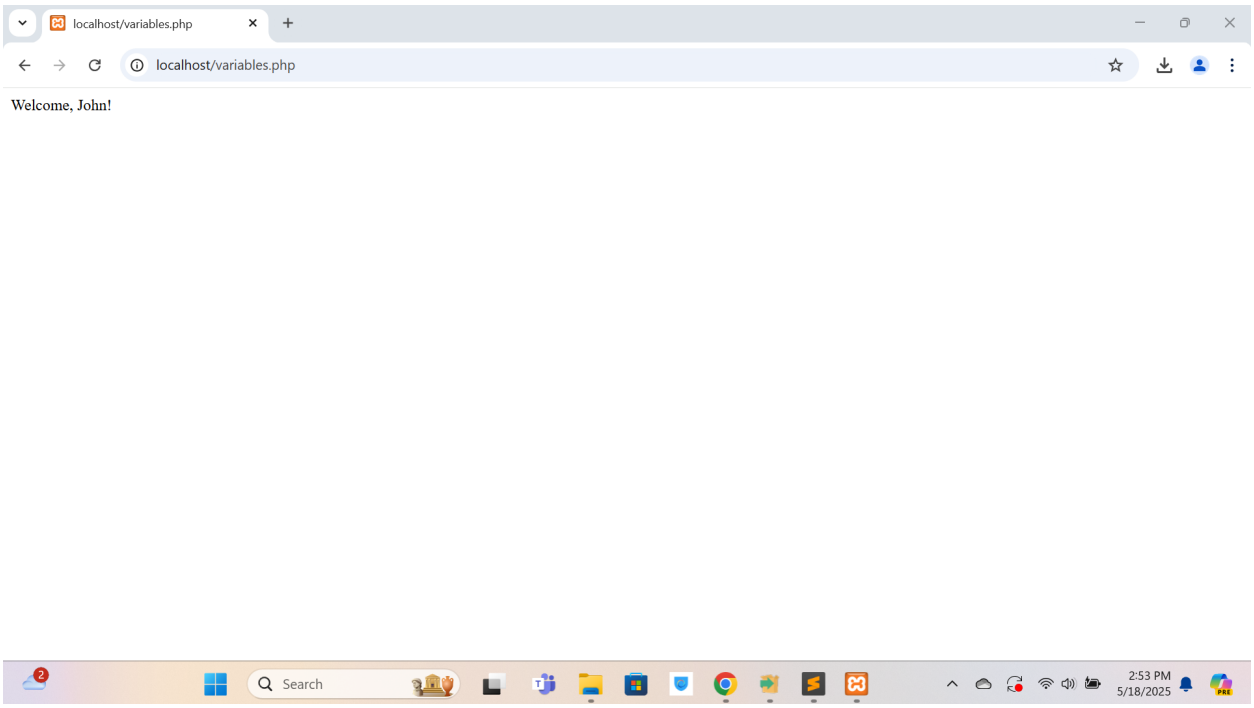

Figure 7: Variables Uses

5.3.3 For Loop Example

```
<?php
for ($i = 1; $i <= 5; $i++) {
    echo "Number: $i<br>";
}
?>
```

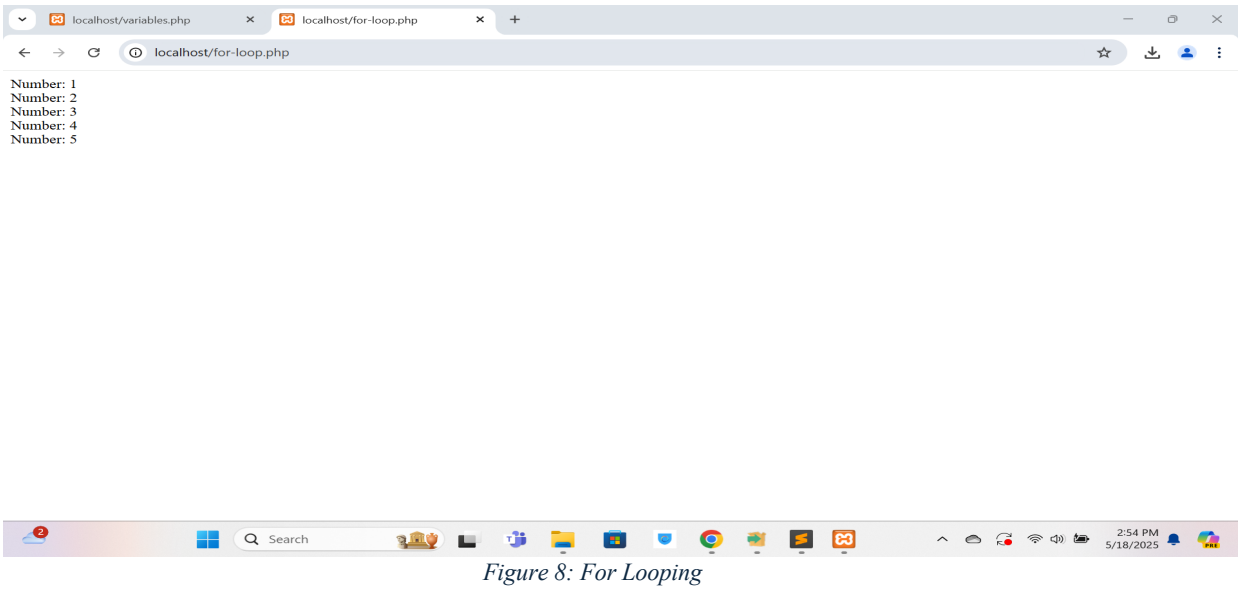

Figure 8: For Looping

5.3.4 While Loop Example

```
<?php
$count = 1;
while ($count <= 3) {
```

```
    echo "This is line $count<br>";
    $count++;
}
?>
```

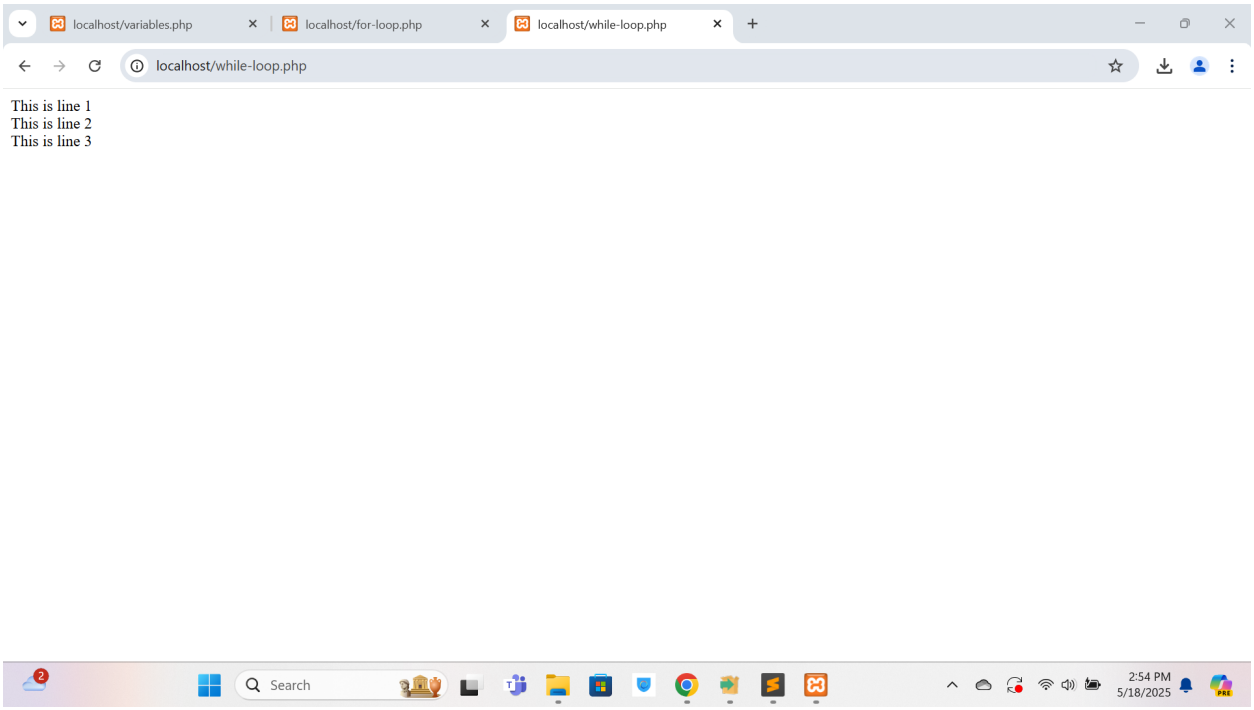

Figure 9: While Looping

5.3.5 Conditional Statement (If-Else)

```
<?php
$marks = 75;
if ($marks >= 50) {
    echo "You passed!";
} else {
    echo "Try again!";
}
?>
```

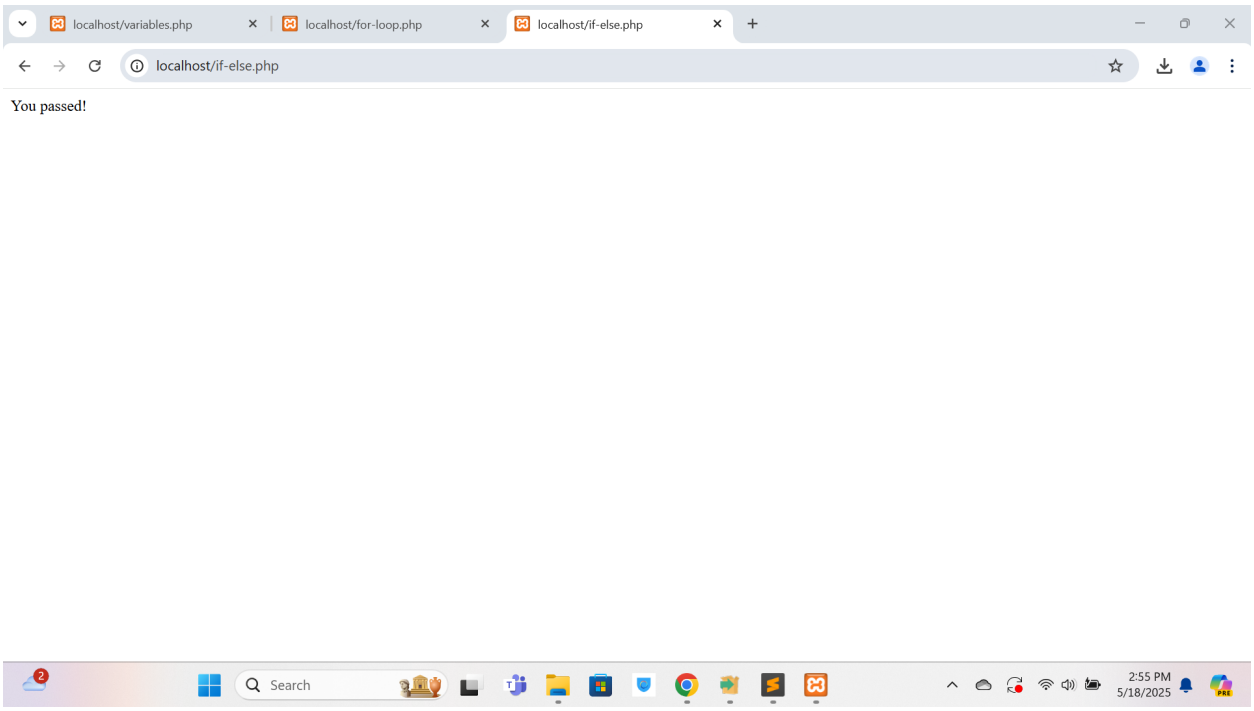

Figure 10: Conditional Statements

5.5 Conclusion

PHP continues to be a versatile and beginner-friendly language for developing dynamic web applications. Leveraging **XAMPP** for local testing allows learners to efficiently write, run, and debug their PHP scripts. Through practical exercises with PHP's core features, such as syntax, loops, and conditionals, students build a solid foundation in server-side scripting. This foundational knowledge paves the way for exploring more complex topics like **form processing**, **session management**, and **database integration** in future development tasks.

Dynamic Content with PHP

6.1 Objective

To demonstrate how PHP handles dynamic content generation through user interaction and form processing.

6.2 Activities

- **Create a Unified PHP File**: Develop a single PHP file that both displays a form and processes its submitted data.
- **Design the HTML Form**: Build a user-friendly form to collect input like username and message.

- **Handle Input with PHP**: Retrieve and display the submitted data dynamically on the same page.
- **Test the Interaction**: Ensure the form responds instantly upon submission with updated output.
- **Enhance Input Security (Recommended)**: Use `htmlspecialchars()` to sanitize input and prevent code injection.

6.3 Source Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dynamic Form Submission</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #f9f9f9;
            padding: 30px;
        }
        .container {
            max-width: 600px;
            margin: auto;
            background: white;
            padding: 25px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h1, h2 {
            text-align: center;
            color: #333;
        }
        label {
            font-weight: bold;
            display: block;
            margin: 10px 0 5px;
        }
        input[type="text"],
        textarea {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ccc;
            border-radius: 6px;
        }
        input[type="submit"] {
            background-color: #007BFF;
            color: white;
            padding: 12px 20px;
            border: none;
            border-radius: 6px;
            cursor: pointer;
            width: 100%;
```

```
            font-size: 16px;
        }
        input[type="submit"]:hover {
            background-color: #0056b3;
        }
        .result {
            margin-top: 30px;
            padding: 20px;
            background: #e6ffe6;
            border: 1px solid #b2d8b2;
            border-radius: 6px;
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Send Us a Message</h1>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $username = htmlspecialchars($_POST['username']);
        $message = htmlspecialchars($_POST['message']);

        echo "<div class='result'>";
        echo "<h2>Your Submission</h2>";
        echo "<p><strong>Username:</strong> $username</p>";
        echo "<p><strong>Message:</strong> $message</p>";
        echo "</div>";
    }
    ?>

    <form method="POST" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label for="username">Username:</label>
        <input type="text" name="username" id="username" required>

        <label for="message">Message:</label>
        <textarea name="message" id="message" rows="5" required></textarea>

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

</body>
</html>
```

6.4 Output

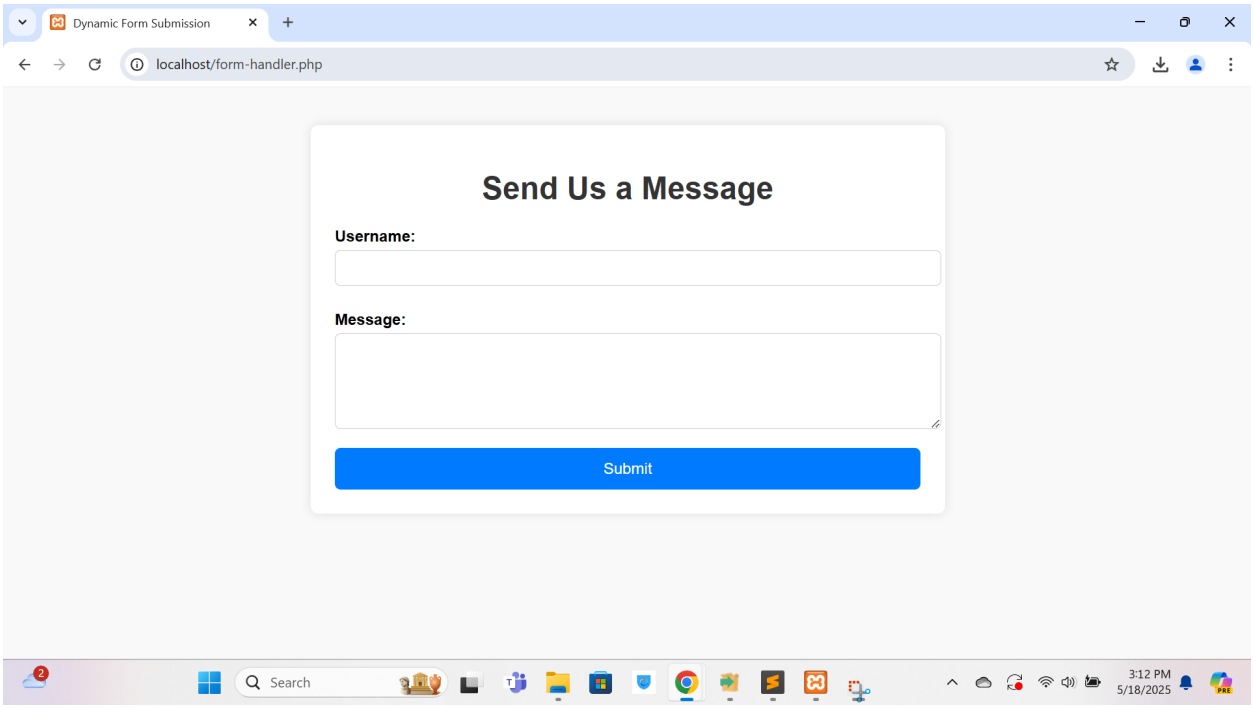

Figure 11: Basic Form Setup

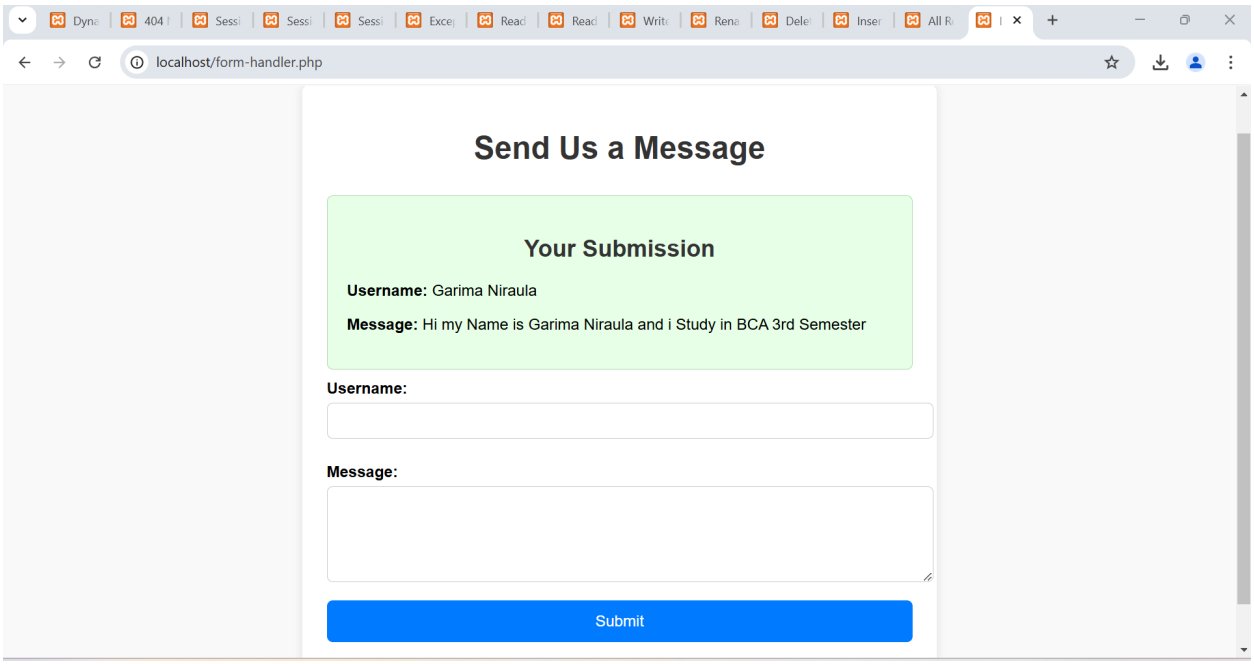

Figure 12: Form Response Recorded

6.5 Conclusion

This activity demonstrates how PHP empowers dynamic user interaction with minimal setup. By combining form design and PHP logic in one file, beginners gain hands-on experience with user-driven content. To take it further, you can add features like **form validation**, **CSS styling**, and **database connectivity**—important skills for building full-fledged web features like **contact pages**, **comment sections**, and **user dashboards**.

Session Management in PHP

7.1 Objective

To demonstrate how to use PHP sessions to preserve user-specific information across different pages in a web application.

7.2 Activities

- **Launch Apache Server** via the XAMPP Control Panel.
- **Create a Project Folder** named `php_sessions` inside the `htdocs` directory.
- **Develop Session Management Scripts**:
  - `set_session.php` – Initializes the session and stores session variables.
  - `get_session.php` – Retrieves and displays the session data.
  - `destroy_session.php` – Ends the session and clears the data.
- **Test Functionality in Browser**:
  - Access the files at `http://localhost/php_sessions/`
  - Monitor how data persists between pages and how it gets cleared after destruction.

7.3 Source Code

set_session.php

```
<?php
session_start();
$_SESSION['username'] = "Garima Niraula";
$_SESSION['favorite_color'] = "Blue";
$_SESSION['hobby'] = "Coding";
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Session Set</title>
    <style>
        body { font-family: Arial; background: #e8f4ff; padding: 20px; }
        .box { background: white; padding: 20px; border-radius: 8px; text-
align: center; max-width: 500px; margin: auto; box-shadow: 0 0 10px
rgba(0,0,0,0.1); }
        a { text-decoration: none; color: #007BFF; font-weight: bold; }
    </style>
</head>
```

```
<body>
<div class="box">
    <h2>Session Variables Are Set</h2>
    <p><a href='get_session.php'>Go to Display Page</a></p>
</div>
</body>
</html>
```

---

get_session.php

```
<?php
session_start();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Session Data</title>
    <style>
        body { font-family: Arial; background: #fff9e6; padding: 20px; }
        .box { background: white; padding: 20px; border-radius: 8px; max-
width: 500px; margin: auto; text-align: center; box-shadow: 0 0 10px
rgba(0,0,0,0.1); }
        a { text-decoration: none; color: #d9534f; font-weight: bold; }
    </style>
</head>
<body>
<div class="box">
    <h2>Session Data</h2>
    <?php
    if (isset($_SESSION['username'])) {
        echo "<p><strong>Username:</strong> " . $_SESSION['username'] .
"</p>";
        echo "<p><strong>Favorite Color:</strong> " .
$_SESSION['favorite_color'] . "</p>";
        echo "<p><strong>Hobby:</strong> " . $_SESSION['hobby'] . "</p>";
    } else {
        echo "<p>No session data found.</p>";
    }
    ?>
    <p><a href='destroy_session.php'>Destroy Session</a></p>
</div>
</body>
</html>
```

---

destroy_session.php

```
<?php
session_start();
session_unset();
session_destroy();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Session Destroyed</title>
```

```
    <style>
        body { font-family: Arial; background: #fdecea; padding: 20px; }
        .box { background: white; padding: 20px; border-radius: 8px; max-
width: 500px; margin: auto; text-align: center; box-shadow: 0 0 10px
rgba(0,0,0,0.1); }
        a { text-decoration: none; color: #007BFF; font-weight: bold; }
    </style>
</head>
<body>
<div class="box">
    <h2>Session Destroyed</h2>
    <p><a href='get_session.php'>Check Session Again</a></p>
</div>
</body>
</html>
```

7.4 Output

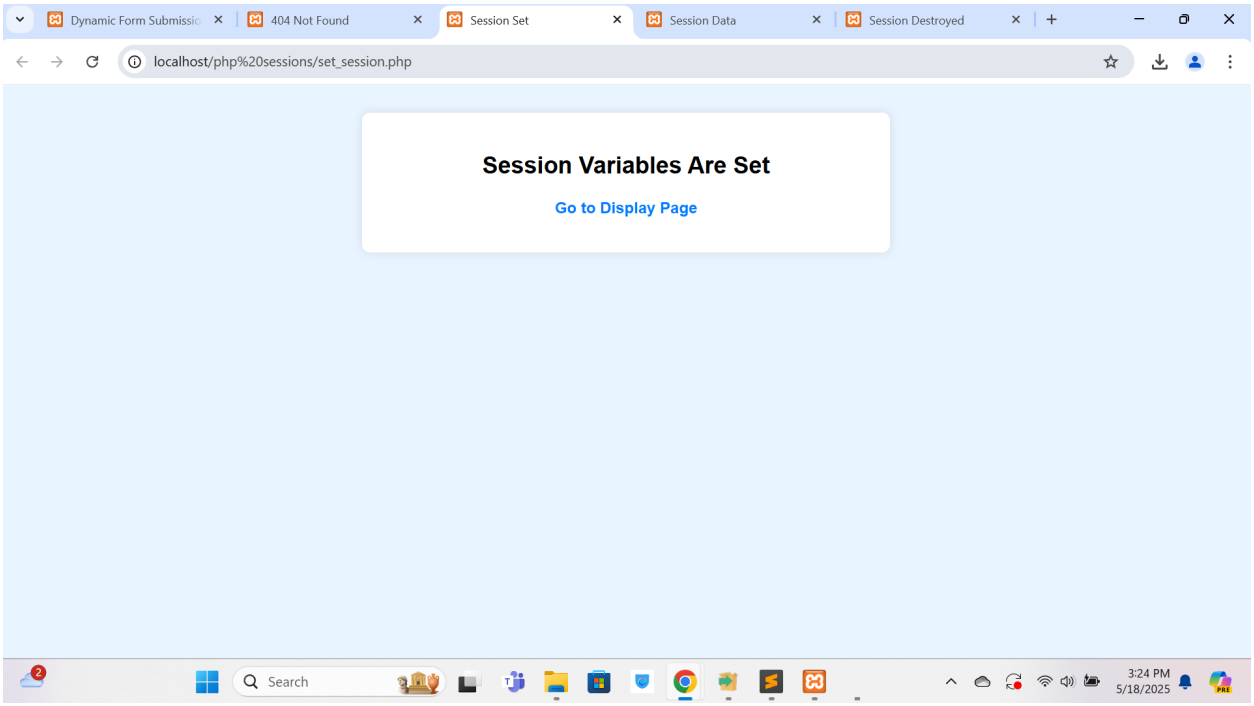

Figure 13: Set Session

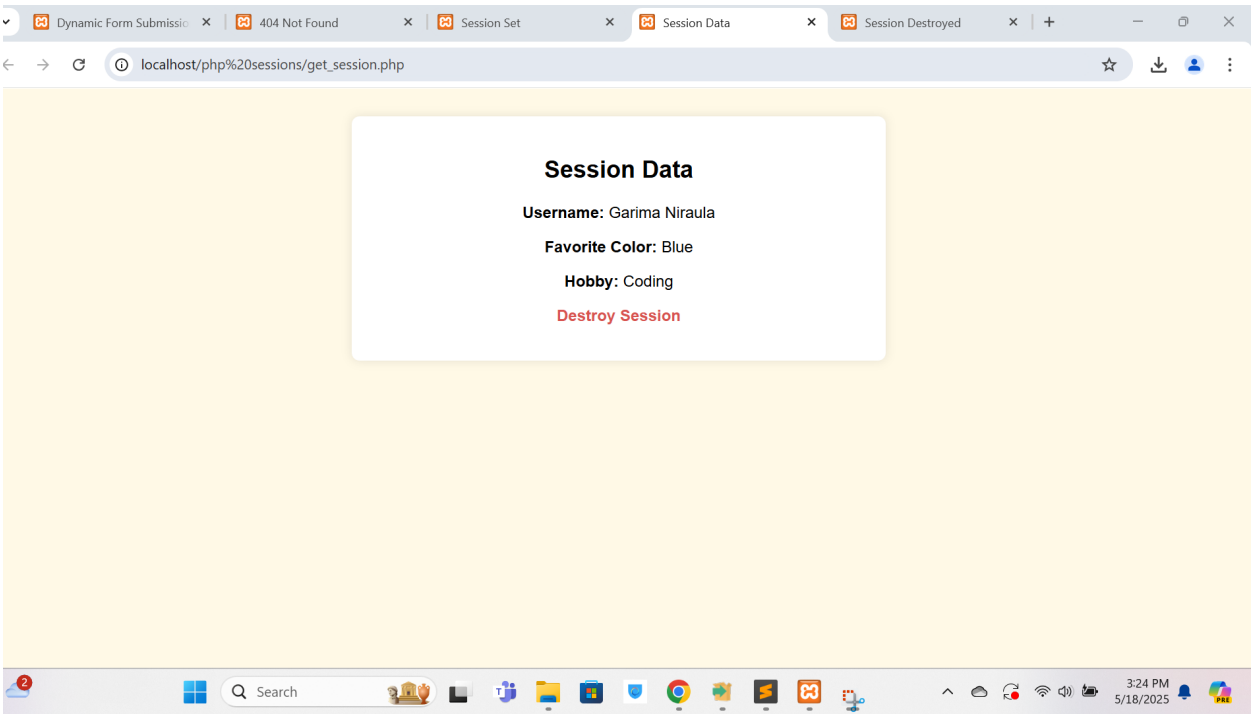

Figure 14: Get Session

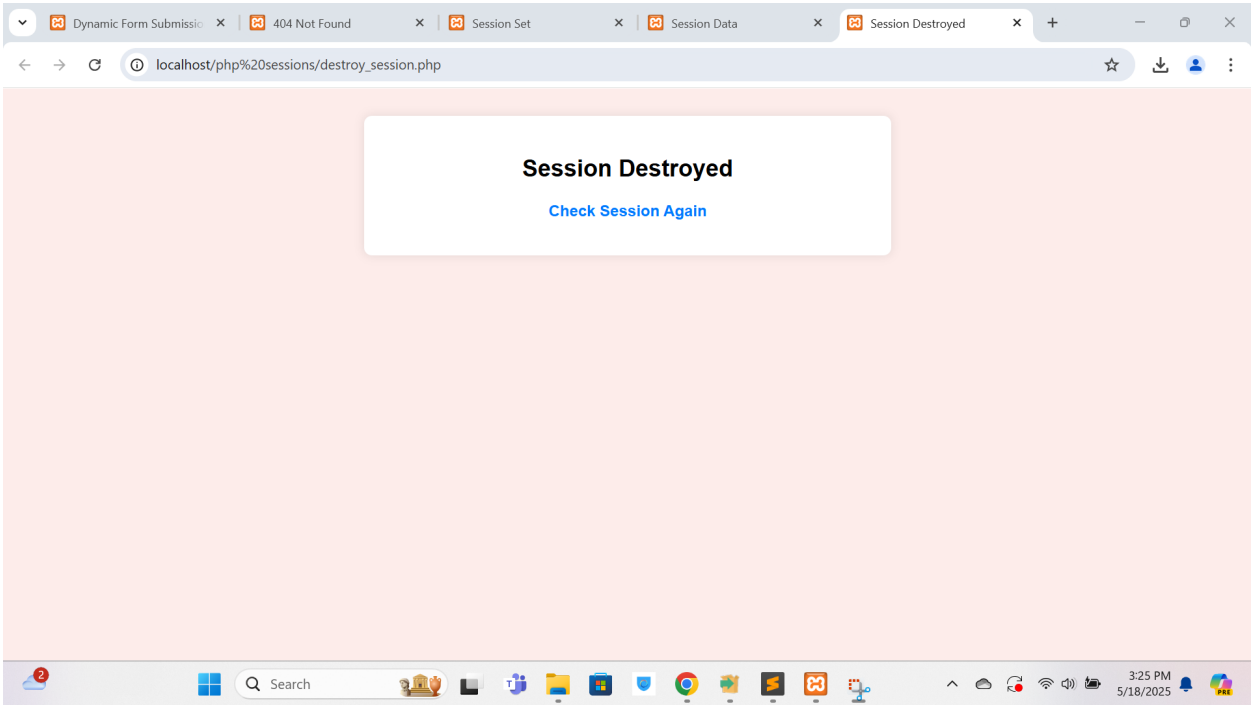

Figure 15: Session Destroyed

7.5 Conclusion

PHP sessions allow developers to manage user-specific information across different pages, enabling a more personalized and interactive experience. In this task, you learned how to initialize, read, and destroy session data using PHP. These practices form the foundation for

building features such as **user login systems**, **shopping carts**, and **personalized dashboards** in dynamic web applications.

Exception Handling in PHP

8.1 Objective

To demonstrate how to handle runtime errors in PHP by throwing and catching exceptions, specifically in cases like division by zero.

8.2 Activities

- **Start the Apache Server** from the XAMPP Control Panel.
- **Write the PHP Script**: Implement a custom function using `try`, `catch`, and `throw` to detect and manage a division by zero error.
- **Ensure Graceful Handling**: Prevent script failure by displaying meaningful, user-friendly error messages when exceptions occur.

8.3 Source Code

```
php
CopyEdit
<?php
function divide($num1, $num2) {
    if ($num2 == 0) {
        throw new Exception("Division by zero is not allowed.");
    }
    return $num1 / $num2;
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Exception Handling Example</title>
    <style>
        body {
            font-family: 'Segoe UI', sans-serif;
            background-color: #f4f6f9;
            padding: 40px;
        }
        .container {
            background: #fff;
            max-width: 600px;
            margin: auto;
            padding: 25px;
```

```
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h2 {
            color: #d9534f;
            text-align: center;
        }
        .output {
            margin-top: 20px;
            padding: 15px;
            background-color: #ffe6e6;
            border-left: 5px solid #d9534f;
            color: #333;
        }
        .success {
            background-color: #e6ffea;
            border-left: 5px solid #5cb85c;
        }
    </style>
</head>
<body>
<div class="container">
    <h2>PHP Exception Handling</h2>
    <div class="output">
    <?php
    try {
        $a = 10;
        $b = 0; // change this to a non-zero value to test success
        $result = divide($a, $b);
        echo "<div class='output success'><strong>Result:</strong>
$result</div>";
    } catch (Exception $e) {
        echo "<strong>Error Occurred:</strong> " . $e->getMessage() . "<br>";
        echo "<strong>Error Code:</strong> " . $e->getCode() . "<br>";
        echo "<strong>In File:</strong> " . $e->getFile() . "<br>";
        echo "<strong>On Line:</strong> " . $e->getLine() . "<br>";
    }
    ?>
    </div>
</div>
</body>
</html>
```

8.4 Output

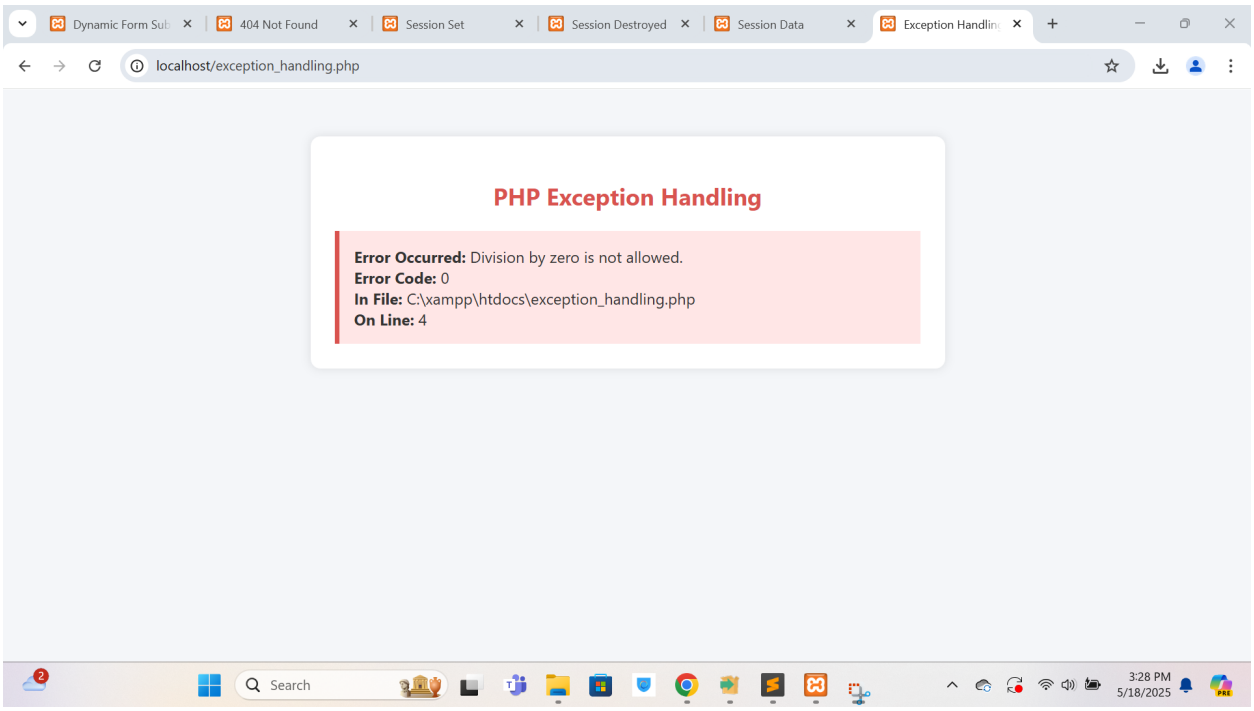

Figure 16: Exception Handling using some CSS

8.5 Conclusion

Using PHP's exception handling capabilities allows developers to anticipate and manage potential errors without breaking the flow of an application. This approach ensures that scripts remain stable and informative when something goes wrong. The use of `try`, `catch`, and `throw` structures not only improves code reliability but also contributes to a better overall user experience.

File Handling in PHP

File handling is an essential part of web development, enabling applications to interact with data stored in files. In PHP, developers can perform operations such as **reading**, **writing**, **renaming**, and **deleting files** using built-in functions. To ensure these operations work correctly, all files (like `note.txt`) should be in the same directory as the PHP scripts.

9.1 Objectives

- Learn how to read, write, rename, and delete files using PHP.
- Practice using file pointers and native PHP functions to manipulate file content.

9.2 Activities

- **Display a File Without a Pointer**: Use `readfile()` to directly output the contents of `note.txt`.
- **Open and Read Using Pointers**: Use `fopen()` with `fread()` to open and display file content.
- **Rename a File**: Change a file's name using `rename()`.
- **Append Content to a File**: Use `fwrite()` in append mode to add new content.
- **Delete a File**: Use `unlink()` to remove a file from the system.

9.3 Source Code

9.3.1 readfile.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Read File (readfile)</title>
    <style>
        body { font-family: Arial; padding: 20px; background-color: #f9f9f9;
}
        pre { background: #e6f7ff; padding: 15px; border-left: 5px solid
#2196F3; }
    </style>
</head>
<body>
    <h2>Reading File using readfile()</h2>
    <pre>
    <?php readfile("note.txt"); ?>
    </pre>
</body>
</html>
```

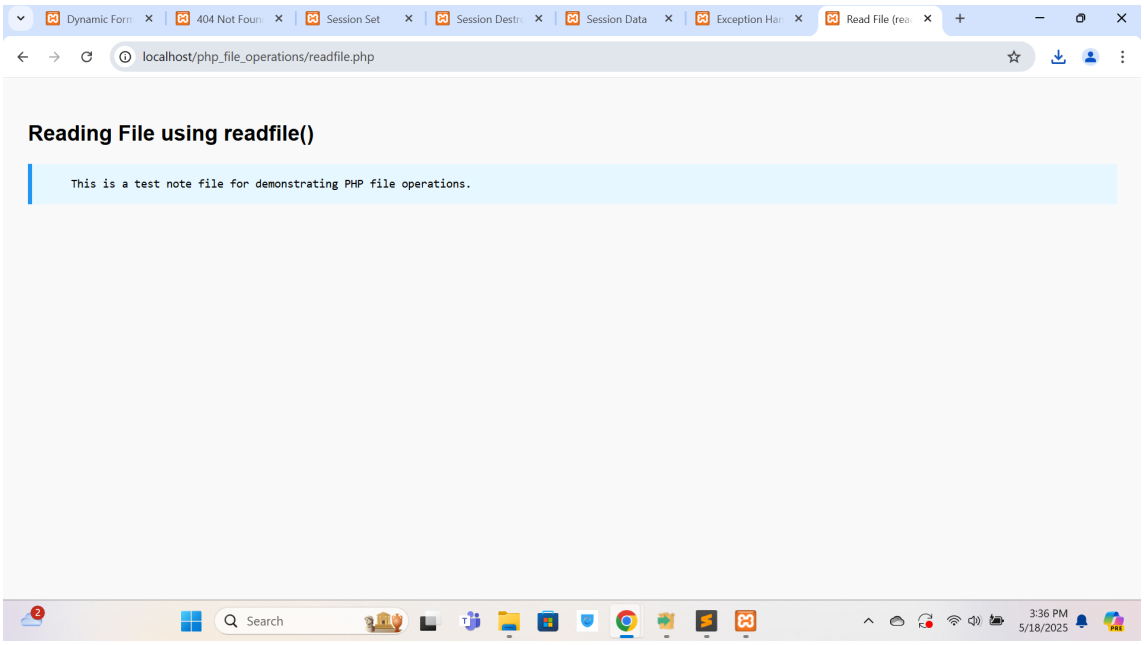

Figure 17: Read File Setup for Php

9.3.2 *fopen.php*

```
<!DOCTYPE html>
<html>
<head>
    <title>Read File (fopen)</title>
    <style>
        body { font-family: Arial; padding: 20px; background-color: #fffbe6;
}
        pre { background: #fff3cd; padding: 15px; border-left: 5px solid
#ffcc00; }
    </style>
</head>
<body>
    <h2>Reading File using fopen() & fread()</h2>
    <pre>
    <?php
    $file = fopen("note.txt", "r");
    if ($file) {
        echo "File opened successfully!\\n";
        $content = fread($file, filesize("note.txt"));
        echo $content;
        fclose($file);
    } else {
        echo "Failed to open file.";
    }
    ?>
    </pre>
</body>
</html>
```

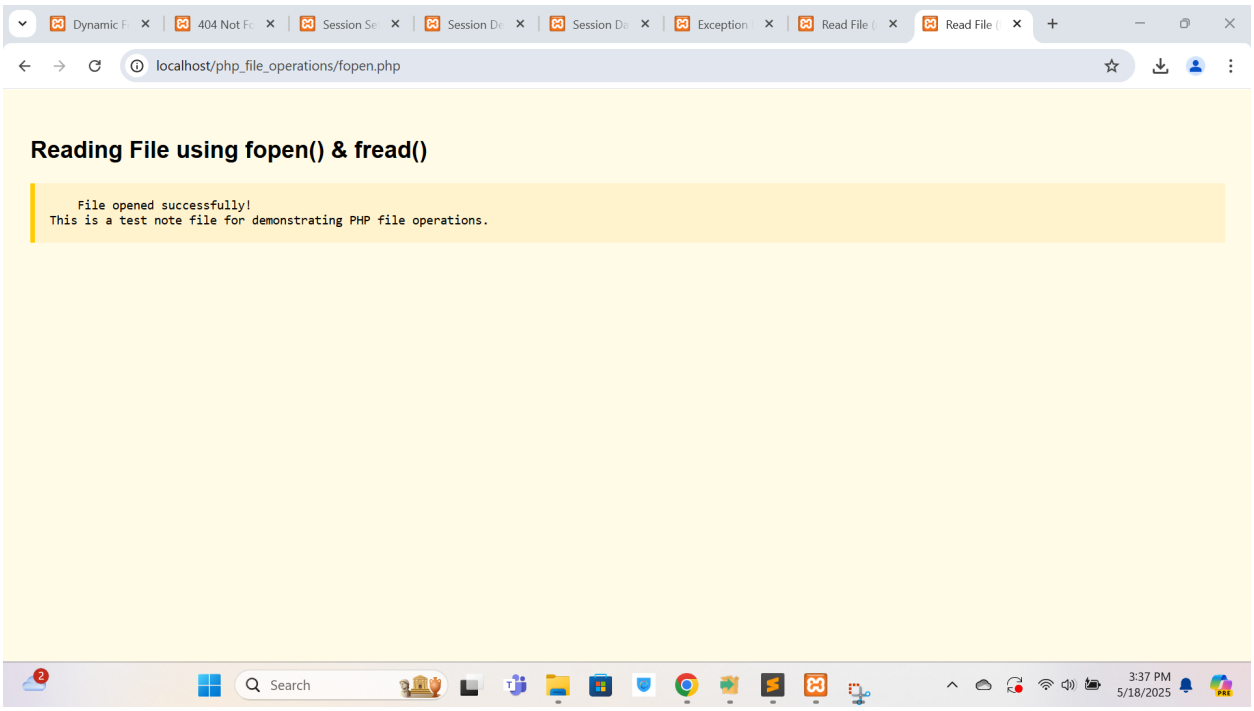

Figure 18 Fopen() and fRead()

9.3.3 rename.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Rename File</title>
    <style>
        body { font-family: Arial; padding: 20px; background-color: #eef9f1;
}
        .msg { background: #d4edda; padding: 15px; border-left: 5px solid
#28a745; }
    </style>
</head>
<body>
    <h2>Renaming File</h2>
    <div class="msg">
    <?php
    echo "Attempting to rename file...<br>";
    $oldName = "data.html";
    $newName = "newData.php";
    if (rename($oldName, $newName)) {
        echo "File renamed successfully!";
    } else {
        echo "Failed to rename file.";
    }
    ?>
    </div>
</body>
</html>
```

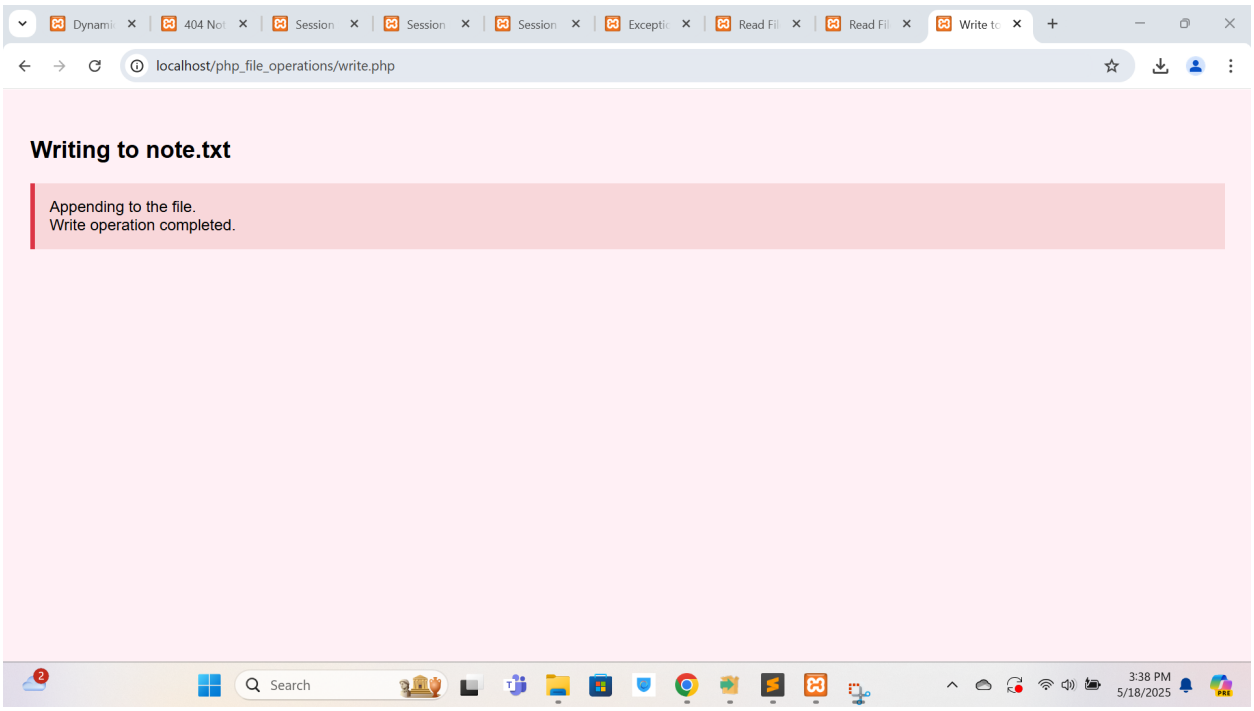

Figure 19 Note.txt

9.3.4 `write.php`

```
<!DOCTYPE html>
<html>
<head>
    <title>Write to File</title>
    <style>
        body { font-family: Arial; padding: 20px; background-color: #fff0f5;
}
        .info { background: #f8d7da; padding: 15px; border-left: 5px solid
#dc3545; }
    </style>
</head>
<body>
    <h2>Appending to note.txt</h2>
    <div class="info">
    <?php
    $ptr = fopen("note.txt", "a"); // "w" would overwrite
    echo "Appending to the file.<br>";
    $add = "Hi, I want to be written on the file.\\n";
    fwrite($ptr, $add);
    fclose($ptr);
    echo "Write operation completed.";
    ?>
    </div>
</body>
</html>
```

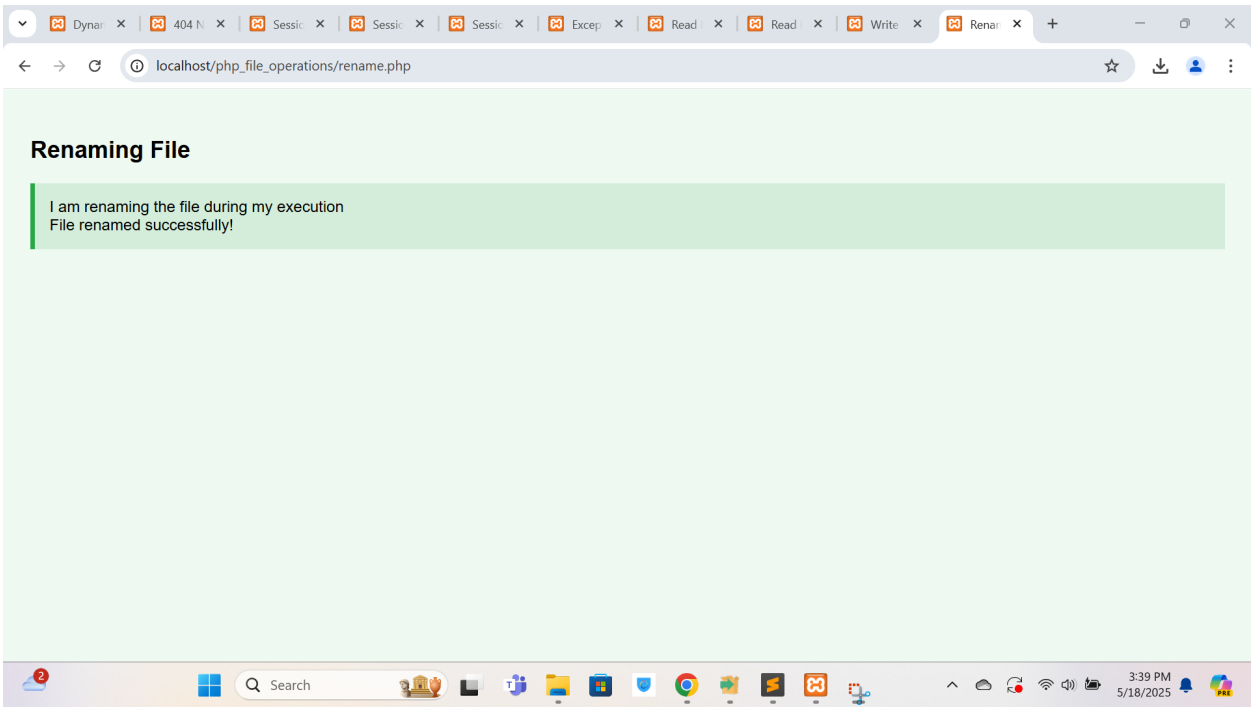

Figure 20: Renaming tendencies in PHP

9.3.5 *delete.php*

```
<!DOCTYPE html>
<html>
<head>
    <title>Delete File</title>
    <style>
        body { font-family: Arial; padding: 20px; background-color: #fefefe;
}
        .warning { background: #f8d7da; padding: 15px; border-left: 5px solid
#dc3545; }
    </style>
</head>
<body>
    <h2>Delete File</h2>
    <div class="warning">
    <?php
    if (file_exists("note.txt")) {
        unlink("note.txt");
        echo "The file 'note.txt' has been deleted.";
    } else {
        echo "File 'note.txt' does not exist.";
    }
    ?>
    </div>
</body>
</html>
```

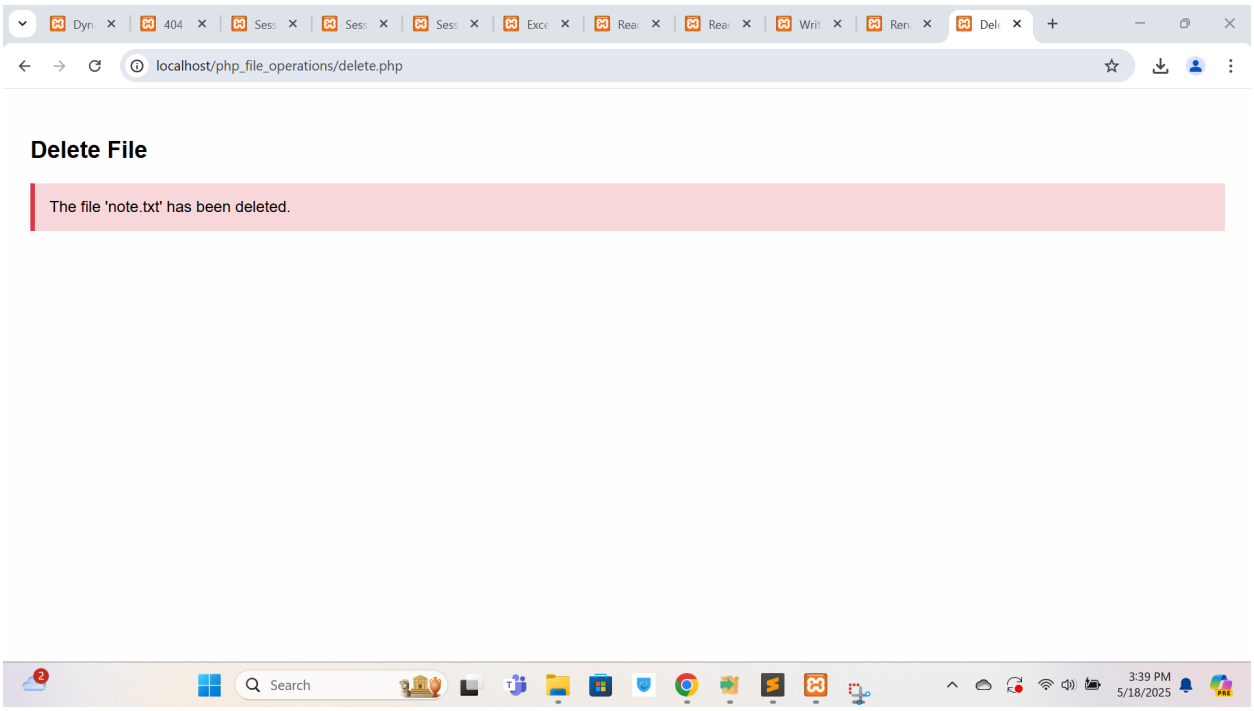

Figure 21: Delete Method Application

9.5 Conclusion

File operations in PHP are powerful tools that allow developers to manipulate external data. With the use of functions like `fopen()`, `fread()`, `fwrite()`, `rename()`, and `unlink()`, developers can handle a variety of file tasks with ease. This section demonstrated the core file handling techniques, laying a strong foundation for building features like file uploads, data logging, and custom storage solutions in real-world PHP applications.

Database Connection and CRUD Operations with PHP

Connecting to a database and performing CRUD (Create, Read, Update, Delete) operations is a fundamental aspect of dynamic web application development. PHP provides powerful functions and prepared statements for securely interacting with a MySQL database.

10.1 Objectives

- Establish a connection to a MySQL database using PHP.
- Perform Create, Read, Update, and Delete operations via secure queries.
- Use **prepared statements** to prevent SQL injection and ensure robust security.

10.2 Activities

- **Database Connection**: Establish a connection using `mysqli_connect()` or `new mysqli()`.
- **Insert New Records**: Collect form input and insert data into a table.
- **Read Records**: Display existing records from the database in a tabular format.
- **Update Records**: Modify specific entries using prepared statements.
- **Delete Records**: Remove database entries based on user input.

10.3 Source Code

✅ *`db.php` – Database Connection*

```
<?php
$serverName = "localhost";
$dbUsername = "root";
$dbPassword = "";
$dbName = "bca3rdsem";

// Create connection
$conn = mysqli_connect($serverName, $dbUsername, $dbPassword, $dbName);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

✅ *`insert.php` – Insert Data*

```
php
CopyEdit
<?php
$message = '';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $conn = new mysqli("localhost", "root", "", "bca3rdsem");

    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $firstname = $_POST['firstname'];
```

```
    $lastname = $_POST['lastname'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];

    $stmt = $conn->prepare("INSERT INTO lab (first_name, last_name, email,
phone) VALUES (?, ?, ?, ?)");
    $stmt->bind_param("ssss", $firstname, $lastname, $email, $phone);

    if ($stmt->execute()) {
        $message = "New record inserted successfully!";
    } else {
        $message = "Error: " . $stmt->error;
    }

    $stmt->close();
    $conn->close();
}
?>

<!DOCTYPE html>
<html>
<head><title>Insert Data</title></head>
<body>
<h2>Insert New Record</h2>
<form method="POST">
    First Name: <input type="text" name="firstname" required><br><br>
    Last Name: <input type="text" name="lastname" required><br><br>
    Email: <input type="email" name="email" required><br><br>
    Phone: <input type="text" name="phone" required><br><br>
    <input type="submit" value="Submit">
</form>
<?php if ($message) echo "<p>$message</p>"; ?>
</body>
</html>
```

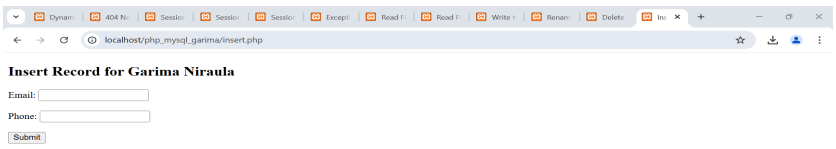

Figure 22: Database Connection and Save Record

✅ `update.php` – Update Data

```
php
CopyEdit
<?php
require 'db.php';
```

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $first_name = $_POST['first_name'];
    $newPhoneNumber = $_POST['newPhoneNumber'];

    $sql = "UPDATE lab SET phone = ? WHERE first_name = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("ss", $newPhoneNumber, $first_name);

    if ($stmt->execute()) {
        echo $stmt->affected_rows > 0 ? "Record updated successfully." : "No
matching record found.";
    } else {
        echo "Error updating record: " . $stmt->error;
    }

    $stmt->close();
    $conn->close();
}
?>

<!DOCTYPE html>
<html>
<head><title>Update Record</title></head>
<body>
<h2>Update Phone Number</h2>
<form method="POST">
    First Name: <input type="text" name="first_name" required><br><br>
    New Phone Number: <input type="text" name="newPhoneNumber"
required><br><br>
    <button type="submit">Update</button>
</form>
</body>
</html>
```

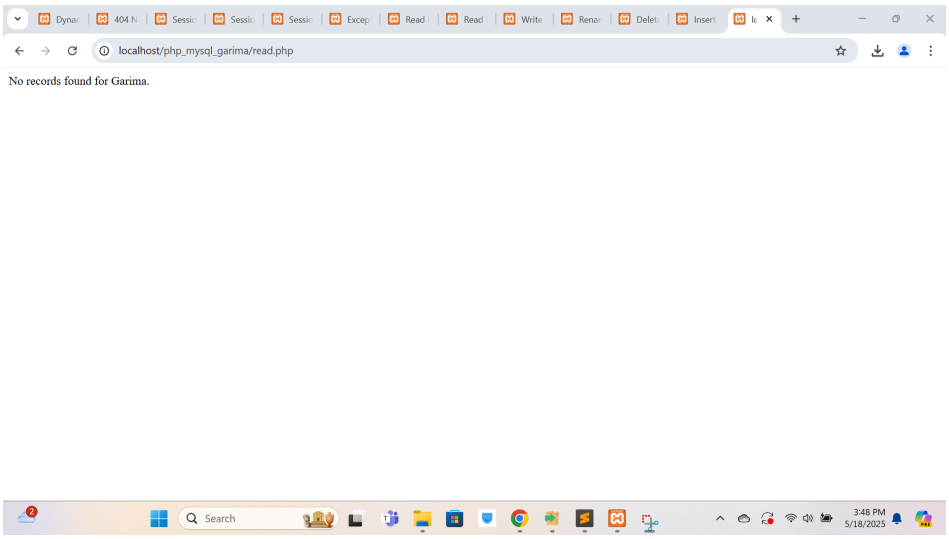

Figure 23: Watch Records

✅ `read.php` – Read Data

php

```
CopyEdit
<?php
require 'db.php';

$sql = "SELECT * FROM lab";
$result = $conn->query($sql);
?>

<!DOCTYPE html>
<html>
<head><title>Read Records</title></head>
<body>
<h2>All Records</h2>
<?php
if ($result->num_rows > 0) {
    echo "<table border='1'>
            <tr>
                <th>ID</th><th>First Name</th><th>Last Name</th>
                <th>Email</th><th>Phone</th><th>Created At</th>
            </tr>";
    while ($row = $result->fetch_assoc()) {
        echo "<tr>
                <td>{$row["id"]}</td>
                <td>{$row["first_name"]}</td>
                <td>{$row["last_name"]}</td>
                <td>{$row["email"]}</td>
                <td>{$row["phone"]}</td>
                <td>{$row["created_at"]}</td>
              </tr>";
    }
    echo "</table>";
} else {
    echo "No records found.";
}
$conn->close();
?>
</body>
</html>
```

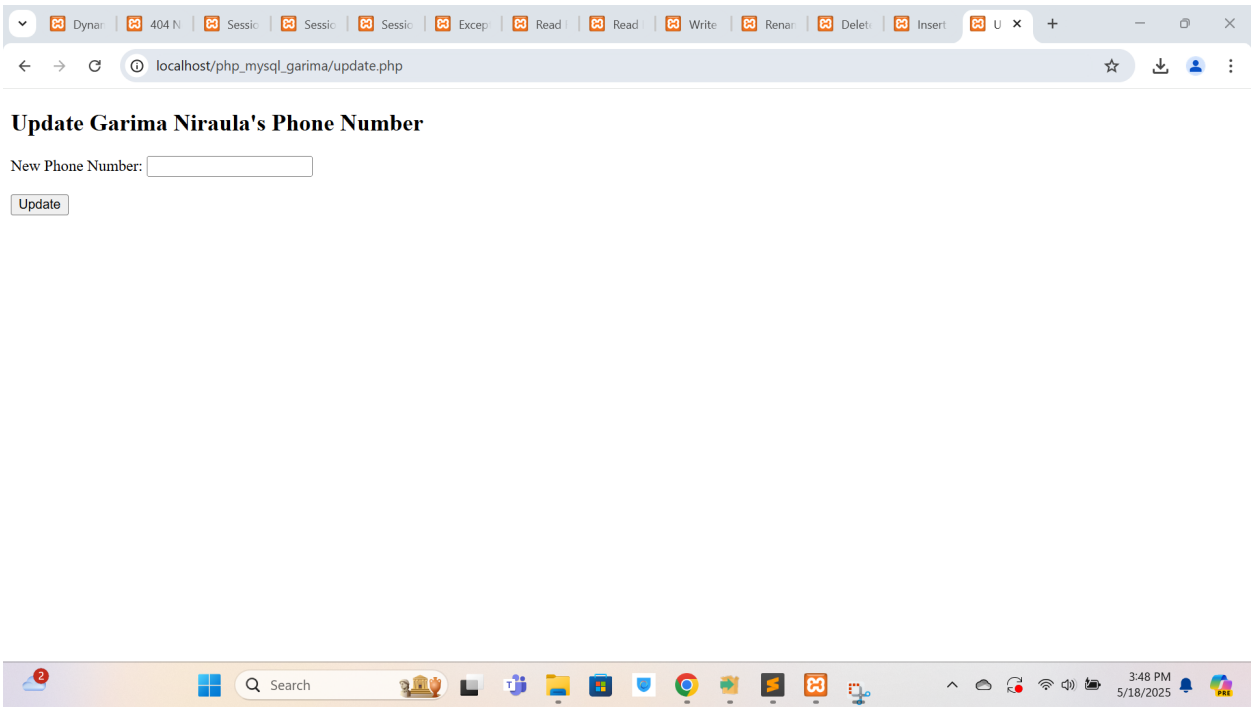

Figure 24: Update DB

✅ *delete.php – Delete Data*

```
php
CopyEdit
<?php
require 'db.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $uid = htmlspecialchars(strip_tags($_POST['id']));

    $sql = "DELETE FROM lab WHERE id = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("i", $uid);

    if ($stmt->execute()) {
        echo $stmt->affected_rows > 0 ? "Record deleted successfully." : "No
record found with ID: $uid";
    } else {
        echo "Error deleting record: " . $stmt->error;
    }

    $stmt->close();
    $conn->close();
}
?>

<!DOCTYPE html>
<html>
<head><title>Delete Record</title></head>
<body>
<h2>Delete a Record</h2>
```

```
<form method="POST">
    User ID: <input type="number" name="id" required><br><br>
    <button type="submit">Delete</button>
</form>
</body>
</html>
```

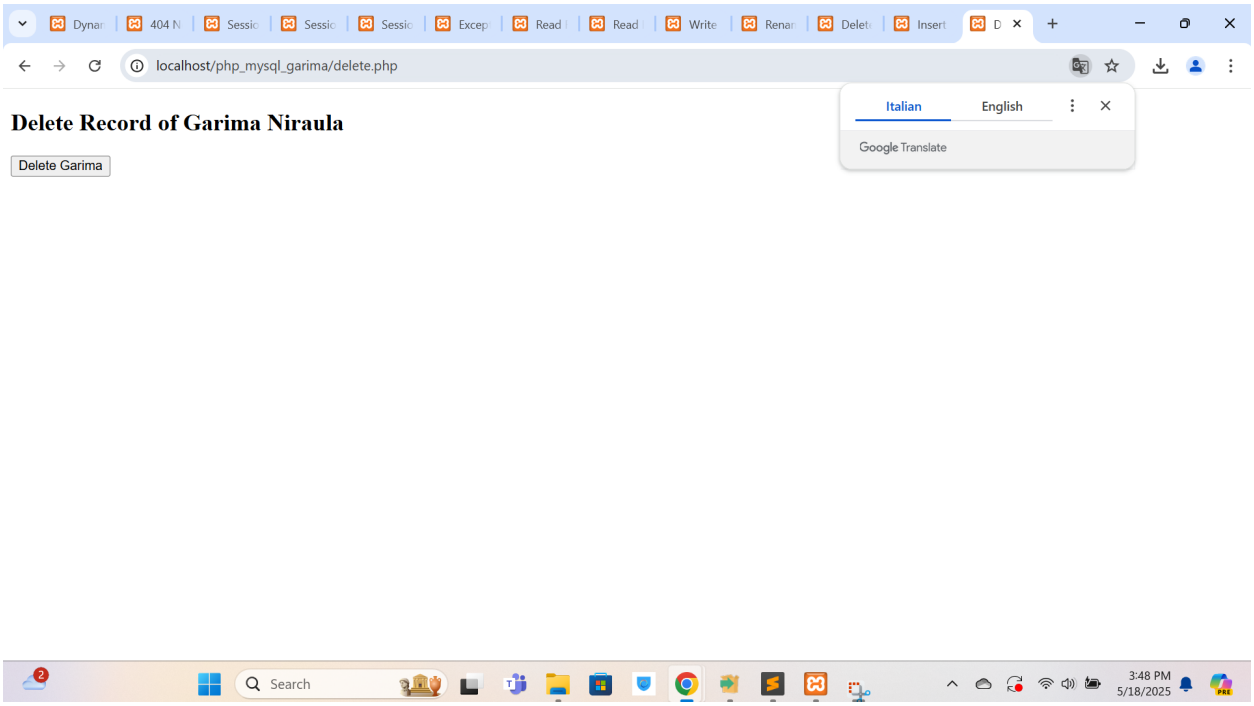

Figure 25: Delete Data from DB

10.5 Conclusion

PHP provides an efficient and secure way to connect to MySQL databases and perform CRUD operations. By using prepared statements, developers can safeguard their applications from SQL injection attacks, which are among the most common vulnerabilities. This activity demonstrates the practical implementation of CRUD logic, forming the foundation for developing interactive data-driven web applications such as admin panels, user dashboards, and content management systems.

Portfolio Showcase

Summary

A portfolio website is a personalized space that showcases an individual's professional background, skill set, and creative abilities. It functions as both a digital résumé and a creative canvas, offering insight into the developer’s mindset through completed projects, design approach, and technical implementation.

The portfolio created for this report is a live web page that presents **projects**, **personal information**, and reflects **front-end development skills**, with ongoing efforts to integrate **back-end functionality** as well.

To explore the portfolio:
➡️ **Visit Garima Niraula's Portfolio**: https://garima-niraula.github.io/portfolio/

Key Features and Technologies Used

- **HTML5 & CSS3**
  Used for structuring content and designing the layout. Enables a responsive, visually appealing interface across devices.
- **GitHub Pages**
  Utilized to host the portfolio as a static website, providing free public access with fast and reliable performance.

Conclusion

This portfolio serves as a central hub for showcasing Garima Niraula's academic background, technical expertise, and project experience. It offers a professional presentation that can be shared with employers, mentors, or collaborators. With continuous updates and enhancements, it has the potential to evolve into a full-featured career showcase reflecting her growth as a developer.