Practical No-6

Aim- To create simple application using JS that will demonstrate the following concepts -

- Declare and assign variables
- Operators and expression in JS
- Looping in JS
- Declare an Array
- User defind and Built in Functions in JS
- Dialog boxes

Program :-

```
<html>
  <body>
    <h1> calculator </h1>
    <input type = "text" id = "result" readonly>
    <table>
      <tr>
        <td> <button onclick = "append('7')" > 7 </button></td>
        <td><button onclick = "append('8')" > 8 </button></td>
        <td><button onclick = "append('9')" > 9 </button></td>
        <td> button onclick = "operator('+')" > + </button></td>
      </tr>
      <tr>
        <td><button onclick = "append('4')" > 4 </button></td>
        <td> <button onclick = "append('5')" > 5 </button></td>
        <td> <button onclick = "append('6')" > 6 </button></td>
        <td> <button onclick = "operator('*')" > * </button></td>
      </tr>
```

Date :

Practical No - 6

Aim- Create simple Application using Javascript that will demonstrate the following concepts :-

- Declare and assign variables
- Operators and expression in Javascript
- Looping in Javascript
- Declare an Array
- User defined and Built in Functions in Javascript
- Dialog boxes.

Theory-

1) Variable Declaration and Assignment :- It allows data storage using variables declared let, const or var. They hold values that can be used or modified throughout the program.

2) Operators and Expressions :- Operators like +, -, *, /, logical operator) Enable calculations and comparisons, creating expressions, creating expressions that produce values based on data manipulation.

3) Looping :- Loops (for, while and do...while) let code run repeatedly until specific condition are met, aiding in efficient, repeatable tasks.

4) Arrays :- Arrays store ordered lists of elements, accessible by index, making it easier to manage collections of similar data

```
<tr>
    <td colspan="4"><button onclick="calculate()">=</button></td>
</tr>
</table>
</body>
<script>
    let result = document.getElementById('result');
    let currentOperation = '';
    let previousOperation = '';

    function append(number) {
        currentOperation += number;
        result.value = currentOperation;
    }

    function operator(op) {
        previousOperation = currentOperation;
        currentOperation = op;
        result.value = currentOperation;
    }

    function clear() {
        currentOperation = '';
        previousOperation = '';
        result.value = currentOperation;
    }

</script>
</html>
```

Calculator

100098

7	8	9	+
4	5	6	-
1	2	3	*
0	.	C	/

=

Date :

5) Functions - functions encapsulate code to execute tasks: built-in functions come with JavaScript while user-defind ones allow custom actions for reusability.

6) Dialog Boxes - Interactive dialog boxes like alert, confirm and prompt display messages and receive user responses, adding interactivity to applications.

Program 2:-

```
<! DOCTYPE html>
<html>
<body>
<p id = "demo"></p>
<script>
  const arr = [1, 2, 3, 4, 5]

    let sum = 0;

  for (let i = 0; i < arr.length; i++){
        sum += arr[i];

  }
document.getElementById("demo").innerHTML = sum;
</script>
</body>
</html>
```

Date

VIVA Question.

Q1) What is JavaScript?

Ans- JavaScript is a high-level interpreted scripting languages primarily used to make web pages interactive. It is also used for server-side programming through environments like Node.js

Q2) What is the difference between 'let', 'var', and 'const' in JavaScript?

Ans- var - function-scoped and can be re-declared and updated.

let - Block-scoped and can be updated but not re-declared.

const - Block-scoped, cannot be re-declared or updated.

Q3) What are closure in JavaScript?

Ans- A closure is a function that remembers and can access variables from its outer function scope even after the outer function has returned.

Q4) What is the difference between '==' and '===' in JavaScript?

Ans- '==' : Loose equality that performs type coercion (converts types before comparison)

'===' : Strict equality that compares both value and type without type coercion.

What is the event loop in JavaScript?

The event loop is a mechanism that allows JavaScript to perform non-blocking operations by offloading tasks like I/O operations to the browser, processing them asynchronously and executing the callbacks when the call stack is empty.

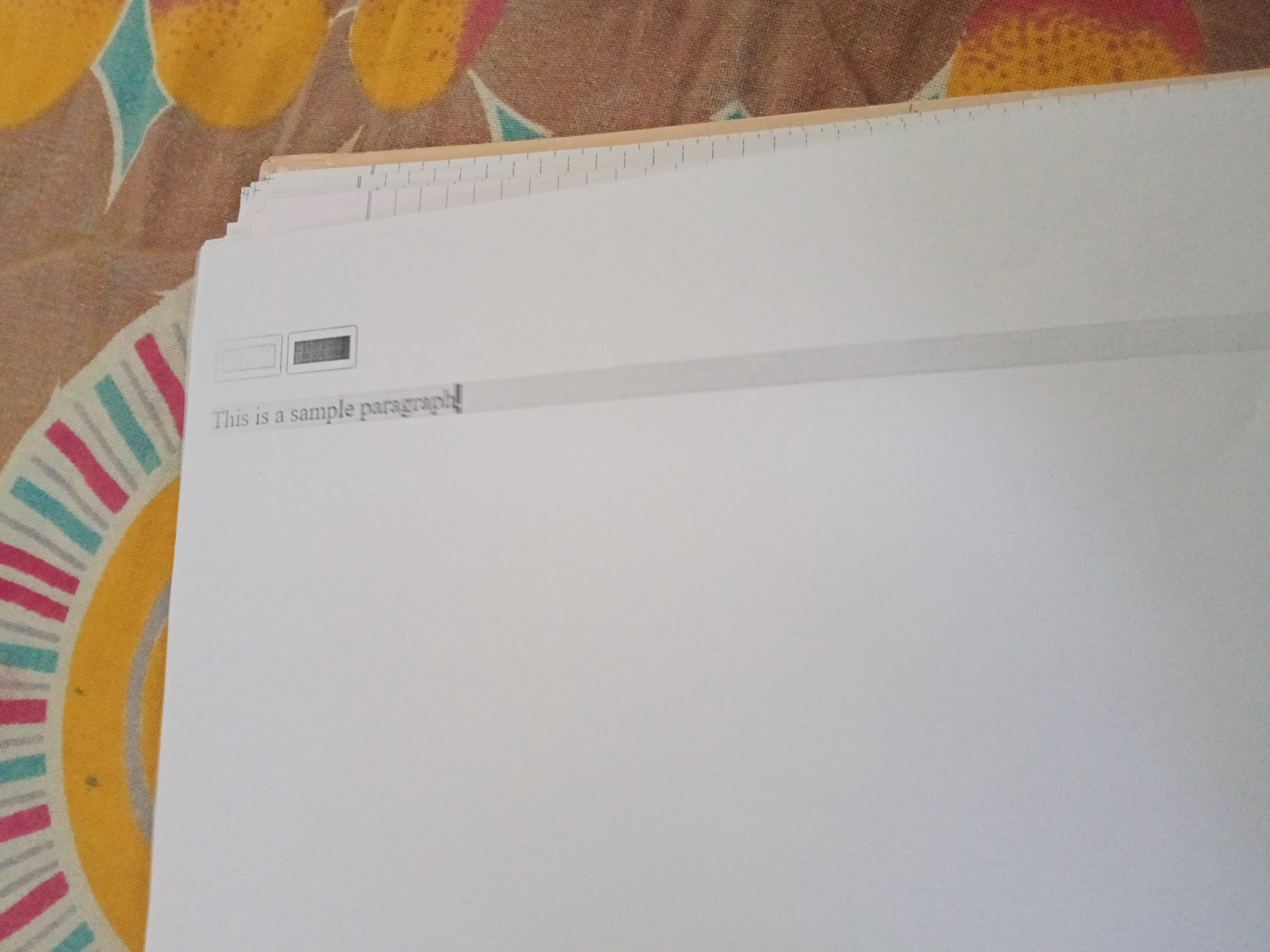
Practical No-7

Aim- Develop a JS where user can select and change background and foreground color.

Program:-

```
<!DOCTYPE html>
<html>
<head>
<title>color changer</title>
</head>
<body>
<input type="color" id="bg-color-picker" value
="#ff0000">
<input type="color" id="text-color-picker" value=
"#ff0000>
<p id="paragraph">This is a sample paragraph.</p>
<script>
const bgColorPicker= document.getElementById("bg-color-
picker");
const textColorPicker =document.getElementById("text-color
-picker");
const paragraph =document.getElementById("paragraph");

bgColorPicker.addEventListener("input", ()=>{
const color1=bgColorPicker.value;
Paragraph.style.background =color1;
});
</script>
</body>
```


This is a sample paragraph

Date :

Practical No-7

Aim- Develop a Javascript where user can select and change background and foreground color.

Theory:

This application demonstrates color selection and dynamic styling in Javascript. The input elements with type color allow users to choose colors via a color picker, providing a user-friendly interface. By calling getElementById, Javascript accesses these color values, which which are updated in real-time when the Apply Colors button is clicked. The changeColors() function retrieves the selected colors and dynamically changes the backgroundColor(backgroundcolor) and text Color(color) style of the body. This practical highlights how Javascript can modify HTML/CSS properties in response to user interactions, enabling a personalized user experience.

Page No.

Date :

VIVA QUESTION

Q1) Explain the concept of "hoisting" in Javascript.
Ans- Hoisting is JavaScript's behaviour of moving variable and function declarations to the top of their scope before code execution. This means that you can use variables and functions before they are declared in the code.

Q2) What is the purpose of the 'this' keyword in Javascript?
Ans- The 'this' keyword refers to the object from which the function was called. Its value is determined by how a function is called, and it changes based on the context.

Q3) What are Arrow Functions in Javascript?
Ans- Arrow functions are a concise way to write functions in Javascript. They don't have their own 'this' context and cannot be used as constructors.

Q4) What are promises in Javascript?
Ans- A Promise is an object representing the eventual completion (or failure) of an asynchronous operation and its resulting value.

Date :

Q5) What is the difference between 'null' and undefined in JavaScript?

Ans 'undefined' :- A variable that has been declared but has not yet been assigned a value.

'null' : An intentional absence of any object value, assigned manually to a variable.

Practical No-8

Aim - Design a JS to change Images every second.

Program:-

```
<html>
<head>
  <title> Image slider </title>
</head>
<body>
   <img id="image" src = "image1.JPG" alt="image1">
<< script >
  var images = ["image1.JPG", "image2.JPG"]
  var imageindex=0;
    function changeimages(){
    imageindex =(imageindex +1) %. image.length;
     document. getElementById("image").src =images
                                    [imageindex];
    }
  set interval (changeimage, 1000);
  </script>
</body>
</html>
```

Date :

Practical No. 8

Aim- Design a Javascript to change images every second.

Theory:- This Javascript - based image carousel cycles through an array of images, changing the display every second. An array, (images) stores image paths, and a variable index tracks the current image. The changeImage() function increments the index and applies the corresponding images to the img element using getElementId. The modulo operator (%) loops the index back to 0 after reaching the end (creating a seamless cycle. The setInterval function calls changeImage() every 1000 milliseconds (1 second), creating an automatic slideshow effect. This practical application demonstrate how Javascript can control dynamic content through timed intervals and array manipulation.

Date

VIVA Question.

Q1) What is an Immediately Invoked Function Expression (IIFE)?

Ans- A IIFE is a function that runs immediately after it is defined. It is written like this

```
Javascript
(function(){
  // code inside the IIFE
})();
```

Q2) What is a callback function?

Ans- A callback is a function passed as an argument to another. which is then executed after some operation is completed. It's commonly used in asynchronous function.

Q3) What is the 'async' and 'await' in Javascript?

Ans- 'Async' and 'await' are used to handle asynchronous code in a more readable way 'async' makes a function as async and 'await' is used to pause the execution until a promise is resolved or rejected.

Date :

Q4) Explain the concept of "Prototype" in JavaScript?

Ans- Every JS object has a Prototype which is another object that provides inheritance. Objects inherit properties and methods from their Prototype. (object.Prototype) is the top of the Prototype.

Q5) What is the difference between 'call()', 'apply()', and 'bind()' methods in JS?

Ans- 'call()' : Invokes a function with a given 'this' value and arguments passed individually.

'apply()' : Invokes a function with a given 'this' value; but arguments are passed as an array.

'bind()' : Returns a new function with a specific 'this' value and can be called later with or without argument

Practical -9

Aim:- Demonstrate the use of XML and JSON files in a web application.

Program:-

1) data.xml :

```
<users>
    <user>
        <name> John Doe</name>
        <email> John.doe@example.com</email>
    </user>
    <user>
        <name> Jane Smith</name>
        <email> Jane.smith@example.com</email>
    </user>
</users>
```

2) data.json

```
{ "users":[
    {"name": "Alice Johnson",
     "email": "alice.johnson@examle.com"
    },
    { "name": "Bob Boown",
      "email": "bob.boown@example.com"
    }
]
}
```

Date :

Practical No-9

Aim- Demonstrate the use of XML and JSON files in a web application.

Theory:-

XML and JSON are formats for exchanging data in web applications.

XML uses nested tags to organize data hierarchically making it highly structured but verbose. It's fetched with XMLHttpRequested and parsed using DomParser.

JSON uses key value pairs, is lightweight, and integrates easily with Javascript. It's often fetched with the fetch API and converted directly into Javascript objects.

In the demo XML offers detailed data structuring while JSON is faster and simpler for web applications, emphasizing ease of use & use in Javascript-based environments. Both formats enable dynamic data display and client-server communication.

3- Index.html

```
<!DOCTYPE html>
<html lang ="en">
 <head>
   <meta charset="UTF-8">
   <meta name ="viewPort" content="width=device-width">
   <title XML and JSON demo</title>
    <link rel="stylesheet" href="style.css">
 </head>
 <body>
   <div class ="container">
     <h1> Data from XML and JSON</h1>
     <button id ="loadXML">Load XML Data</button>
     <button id ="loadJSON"> Load JSON Data</button>
     <h2> Data: </h2>
     <ul id ="data-list"></ul>
   </div>

 <script src="script.js"></script>
</body>
</html>
```

When the page loads, the user will see two buttons: "Load XML Data" and "Load JSON Data".

Clicking on the "Load XML Data" button will fetch data from the `data.xml` file and display the list of users (names and emails) on the webpage.

Clicking on the "Load JSON Data" button will fetch data from the `data.json` file and display the list of users (names and emails) on the webpage.

Example Output:

makefile
Copy code
```
Name: Alice Johnson
Email: alice.johnson@example.com

Name: Bob Brown
Email: bob.brown@example.com
```

Date :

VIVA Question.

Q1) What are XML and JSON and where are they commonly used?

Ans- XML and JSON are data formats. XML uses nested tags & JSON uses key value pairs, both commonly used for data exchange in web application.

Q2) How does XML differ from JSON in terms of structure and syntax?

Ans- XML is hierarchal with tags allowing nested data JSON uses simpler key-value pairs, making it lighter and more readable for JS.

Q3) What are the advantages of using JSON over XML in web applications?

Ans- JSON is light weight faster to parse in JS. and less verbose, making it ideal for client server interactions in web apps.

Q4) How do you retrieve and parse XML data in JS?

Ans- Use XMLHTTPRequest or Fetch to retrieve XML, then parse with DomParser to navigate XML tags.

Q5) What method is used to fetch JSON data in JS?

Ans- The fetch API is preferred for JSON as its simpler, returns Promises and integrates smoothly with JS object.

Page No.

5) script.js

```
document.getElementById("loadXML").addEventListener
    ("click", loadXMLData);
document.getElementById("loadJson").addEventListener
    ("click", loadJSONData);

function loadXMLData(){
    const xhr = new XMLHttpRequest();
    xhr.open("GET", "data.xml", true);
    xhr.onload = function(){
      if(xhr.status === 200){
        const xmlData = xhr.responseXML;
        const users = xmlData.getElementsByTagName
                                    ("user");
        let output = "";
        for(let i=0; i<users.length; i++){
        const name = users[i].getElementsByTagName
            ("name")[0].textContent;
    };
    xhr.send();
}
```

Practical - 10

Aim- To design and implement a simple login Form using PHP, HTML, CSS and MySQL which allows a user to enter their credientials and verify their login details against a MySQL database.

code:-

1) Database setup:

```
CREATE DATABASE login_system;

USE login_system;

CREATE TABLE users(
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(30) NOT NULL,
    password VARCHAR(255) NOT NULL,
);
```

2) db.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "login_system";
$conn = new mysqli($servername, $username, $password,
                   $dbname);
if ($conn->connect_error){
    die("Connection Failed:" . $conn->connect_error);
}
?>
```

Date :

Practical No-10

Aim- To design and implement a simple login form using PHP, HTML, CSS and MySQL, which allows a user to enter their credentials and verify their login details against a MySQL database

Theory:-

Login systems are essential components of web applications that require authentication. A typical login system involves user input through a web form (HTML), server-side processing to verify the user's credential (PHP), and storage of these credentials in a database (MySQL).

1) HTML:- To create the structure of the login form where users can input their credentials.

2) CSS:- To style the login form and make it visually appealing.

3) PHP:- A server-side scripting language used to interact with the MySQL database and authenticate users by comparing input credentials with stored ones.

4) MySQL:- A relational database management system used to store user credential securely.

5) SQL queries:- Used to retrieve and verify login details from the MySQL database.

3. login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width">
    <title>Login Form</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="login-container">
        <h2>login</h2>
        <form action="login.php" method="POST">
            <label for="username">username</label>
            <input type="text" name="username" required>
            <label for="password">password</label>
            <input type="password" name="password" required>
            <button type="submit">Login</button>
        </form>
    </div>
</body>
</html>
```

Login successful! Welcome testuser.

Invalid username or password!

Login

Username:

Password:

Login

Date

VIVA Question -

Q1) What are the key components of a login form in this implementation?

Ans- The key components include HTML for the form structure, CSS for styling PHP for processing login requests, and MYSQL for storing user credentials.

Q2) How do you connect PHP to a MySQL database?

Ans- Use mysqli_connect() or PDO to establish a connection, specifying the server, username, password, and database name.

Q3) What is the purpose of using Prepared statements in this implementation?

Ans- Prepare statements prevent SQL injection by separating SQL code from user input, enhancing security during login verification.

Q4) How do you validate user credentials in PHP?

Ans- Retrieve the user's input from the form, query the database for matching credentials and check if the hashed password matches the stored hash.

Q5) What methods can be used passwords securely in MySQL?

Ans- Use password hashing functions like password_hash() for storing passwords and password_verify() for checking password during login.

Page No.