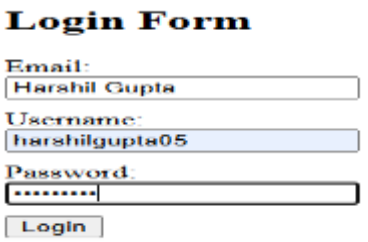
Experiment-4

Objective: - Form validation using java script

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport"
content="width=device-width,
initialscale=1.0">
<title>Login Form Validation</title>
<style> .error {
color: red; font-
size: 0.9em;
}
input { display: block;
margin-bottom:
10px;
}
</style>
</head>

<body>
<h2>Login Form</h2>
<form id="loginForm">
<label for="email">Email:</label>
<input type="text" id="email"
name="email">
<span id="emailError"
class="error"></span>
<label
for="username">Username:</label>
<input type="text" id="username"
name="username">
<span id="usernameError"
class="error"></span>
<label for="password">Password:</label>
<input type="password" id="password"
name="password">
<span id="passwordError"
class="error"></span>
<button type="button"
onclick="validateForm()">Login</button>
</form> <script>
function validateForm() { const
email =
document.getElementById('email').value.t
rim();
const username =
document.getElementById('username').va
lue.trim(); const password =
document.getElementById('password').val
ue.trim();
```

```
const emailError = document.getElementById('emailError'); const
usernameError = document.getElementById('usernameError');
const passwordError = document.getElementById('passwordError');
emailError.textContent = ''; usernameError.textContent = '';
passwordError.textContent = ''; let isValid = true; if (!email) {
emailError.textContent = 'Email is required.';
isValid = false;
} else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
emailError.textContent = 'Enter a valid email address.';
isValid = false;
}
if (!username) {
usernameError.textContent = 'Username is required.';
isValid = false;
}
if (!password) {
passwordError.textContent = 'Password is required.';
isValid = false;
} if (isValid) { alert('Login
successful!');
</script>
</body>
```



Experiment-5

Objective: - Current day and current time using java script

```
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Current Date and Time</title>
</head>
<body>
<h1>Current Day and Time</h1>
<script>
const now = new Date();
const days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"];  const day = days[now.getDay()];  const hours = now.getHours();  const minutes =
now.getMinutes();  const seconds = now.getSeconds();
const dayTime = `${hours.toString().padStart(2, '0')}:${minutes.toString().padStart(2, '0')}:${seconds.toString().padStart(2, '0')}`;
console.log(`Today is: ${day}`);
console.log(`Current time is: ${dayTime}`);
</script>
</body>
</html>
```

Experiment-6

Objective: - Student table with attributes name, CGPA, city using js

```
const Student=[{
Name:"Rohit",
City:"Agra",
CGPA:"9.0"
},
{
Name:"Prabhat",
City:"Mumbai",
CGPA:"8.1"
},
{
Name:"Utkarh",
City:"Lucknow",
CGPA:"8.0"
}];
console.log(Student);
console.table(Student);  for (let
i=0;i<Student.length;i++){
if(Student[i].City=="Mumbai"){
console.log(Student[i]);
}
else if(Student[i].CGPA>8.4){
console.log(Student[i]);
}
```

```
[
  { Name: 'Rohit', City: 'Agra', CGPA: '9.0' },
  { Name: 'Prabhat', City: 'Mumbai', CGPA: '8.1' },
  { Name: 'Utkarh', City: 'Lucknow', CGPA: '8.0' }
]
```

(index)	Name	City	CGPA
0	'Rohit'	'Agra'	'9.0'
1	'Prabhat'	'Mumbai'	'8.1'
2	'Utkarh'	'Lucknow'	'8.0'

```
{ Name: 'Rohit', City: 'Agra', CGPA: '9.0' }
{ Name: 'Prabhat', City: 'Mumbai', CGPA: '8.1' }
```

Experiment-7

Objective: - Change the content of html using java script

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
<style>  body{
background-color:
gray;
}
.content{  background-color: rgb(121,
154, 110);  padding: 20px;
display: flex;  justify-
content: center;  text-align:
center; }
</style>
</head>
<body>
<div class="content">
<h1 id="title">Change the text</h1><br><br>
<div class="container">
<p id="desc">Choose the sport </p>
<button id="change">Cricket</button>
<button id="change1">Football</button>
</div>
</div>  <script>
document.getElementById('change').addEventListener('click',
function(){  document.getElementById('title').textContent = 'Cricket’;
```

```
document.getElementById('desc').textContent = 'welcome to
javascript';
});
document.getElementById('change1').addEventListener('click', function(){
document.getElementById('title').textContent =’Football’;
document.getElementById('desc').textContent = 'welcome to javascript';
});
</script>
</body>
</html>
```

Experiment-8

Objective: - Change the attribute of html using java script

```
 <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Change Attribute Example</title>
</head>
<body>
<img id="myImage" src=”vote.png " alt="Old Image" width="200">
<button onclick="changeAttribute()">Change Image</button>
<script>
function changeAttribute() {
let img = document.getElementById("myImage");
img.setAttribute("src", "lapy.png");
img.setAttribute("alt", "vote.png");
img.setAttribute("width", "500");
}
</script>
</body>
</html>
```

Experiment-9

Object: - Program to get the IP Address of local host

```
import java.net.InetAddress;  import
java.net.UnknownHostException;  public
class GetLocalIPAddress {  public static
void main(String[] args) {
try {
InetAddress localHost = InetAddress.getLocalHost();
System.out.println("Local Hostname: " + localHost.getHostName());
System.out.println("Local IP Address: " + localHost.getHostAddress());
} catch (UnknownHostException e) {
System.err.println("Unable to get the local host address.");
e.printStackTrace();
```

```
Local Hostname: prod-repl-java-85c68fb984-h4rcf
Local IP Address: 10.236.0.52
```

Experiment-10

Object: Program to extract the protocol, port, and host from a URL in Java

```
import java.net.URL;  public
class UrlDetails {
public static void main(String[] args) {  try
{
String urlString = "https://www.example.com:8080/path?query=value";
URL url = new URL(urlString);
String protocol = url.getProtocol();  String
host = url.getHost();
int port = url.getPort(); // Returns -1 if no port is specified
System.out.println("URL: " + urlString);
System.out.println("Protocol: " + protocol);
System.out.println("Host: " + host);
System.out.println("Port: " + (port == -1 ? "Default" : port));
} catch (Exception e) {
System.out.println("Invalid URL: " + e.getMessage());
}
}
}
```

```
Note: /tmp/SuGeXRWNBZ/Main.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
URL: https://www.programiz.com/java-programming/online-compiler/
Protocol: https
Host: www.programiz.com
Port: Default
```

Experiment-11

Object: -TCP/IP Server Socket program

```
import java.io.*; import
java.net.*; public class
TcpServer {
public static void main(String[] args) {
int port = 5000;
try (ServerSocket serverSocket = new ServerSocket(port)) {
System.out.println("Server is running and waiting for a client...");
Socket socket = serverSocket.accept();
System.out.println("Client connected: " + socket.getInetAddress());
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
String message;
while ((message = in.readLine()) != null) {
System.out.println("Client says: " +
message); out.println("Server received: " +
message); if
(message.equalsIgnoreCase("bye")) {
System.out.println("Connection closed by client."); break;
}
}
socket.close(); // Close connection
System.out.println("Server stopped.");
} catch (IOException e) {
System.out.println("Server error: " + e.getMessage());
e.printStackTrace();
}
}
}
```

```
Client code import java.io.*; import
java.net.*;  public class TCPClient {
public static void main(String[] args)
{ try {
Socket socket = new Socket("localhost", 5000);
System.out.println("Connected to the server");
BufferedReader in = new BufferedReader( new
InputStreamReader(socket.getInputStream()));
PrintWriter out = new PrintWriter(socket.getOutputStream(),
true); String message = "Hello from Client!"; out.println(message);
System.out.println("Sent to server: " + message);
String serverResponse = in.readLine();
System.out.println("Received from server: " + serverResponse);
socket.close();
} catch (IOException e) {
System.out.println("Error: " + e.getMessage());
}
}
}
```

Server Output: -

```
Server is running and waiting for a connection...
Client connected: /127.0.0.1
Received from client: Hello from Client!
Sent to client: Hello from Server!
```

Client Output:-

```
Connected to the server
Sent to server: Hello from Client!
Received from server: Hello from Server!
```

Experiment

Objective : Create a Java Bean for Employee information (EmpID, Name, Salary, Designation and Department)

```
public class employee {
   private int empID;
   private String name;
   private double salary;
   private String designation;
   private String department;

   // Default Constructor
   public employee() {}

   // Parameterized Constructor
   public employee(int empID, String name, double salary, String designation, String department) {
      this.empID = empID;
      this.name = name;
      this.salary = salary;
      this.designation = designation;
      this.department = department;
   }

   // Getter and Setter for empID
   public int getEmpID() {
      return empID;
   }

   public void setEmpID(int empID) {
      this.empID = empID;
   }

   // Getter and Setter for name
```

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

// Getter and Setter for salary
public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

// Getter and Setter for designation
public String getDesignation() {
    return designation;
}

public void setDesignation(String designation) {
    this.designation = designation;
}

// Getter and Setter for department
public String getDepartment() {
    return department;
}

public void setDepartment(String department) {
    this.department = department;
}
```

```
    @Override
    public String toString() {
        return "Employee{" +
                "empID=" + empID +
                ", name='" + name + '\'' +
                ", salary=" + salary +
                ", designation='" + designation + '\'' +
                ", department='" + department + '\'' +
                '}';
    }
}
```