A LAB MANUAL

FULL STACK WEB DEVELOPMENT LAB (U23CD5L2)

B.E. III Year V Semester

By

MS. ZOYA TABASSUM
Assistant Professor
Dept. of CSE – Data Science

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY
(UGC Autonomous)
Approved by AICTE | Affiliated to Osmania University | Estd.2003
Accredited by **NBA**, Accredited '**A**' Grade by NAAC
(Academic Year: 2024 – 2025)

INDEX

12	Implement slide animations using jQuery on different elements. Use slideDown(), slideUp(), and slideToggle() to demonstrate UI effects like drop-down menus or slide panels.	
13	Install Node.js and write a basic Node.js program using console.log() and process object. Execute it using the command line and display output.	

Vision of the Department:

To develop the world's next generation Data Scientist by offering top-notch education with cutting edge technologies.

Mission of the Department:

DM1: Providing students to understand the principles of Data Science with hands-on experience.

DM2: Offering students to analyze the data through effective teaching learning methods and cutting edge technologies in multi disciplinary fields.

DM3: Preparing students for R&D, Industrial design, entrepreneurship and employment.

DM4: Encouraging students with interaction and Industry Institute partnership through various organizations.

Note: DM: Department Mission

Head of the Department

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

B.E. Computer Science and Engineering (Data Science) Program Educational Objectives (PEOs):

PEO1	Students will have strong foundation In Basic Science, Mathematics, Statistics, Computer Science and Allied engineering.
PEO2	Students will develop Data Science applications at advanced level for employability in IT industry especially in niche technologies.
PEO3	Students will be provided with strong foundation in Data Science and its applications for their carrier as Data Scientist/Data Engineers.

Head of the Department

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

B.E. Computer Science and Engineering (Data Science) Program Outcomes (POs):

Engineering Graduates will be able to:

S. No.	Program Outcomes (POs):
1.	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2.	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3.	**Design/Development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4.	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5.	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6.	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7.	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8.	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9.	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10.	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11.	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12.	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

Head of the Department

B.E. Computer Science and Engineering (Data Science) Program Specific Outcomes (PSO's):

PSO1	**Professional Skills:** The ability to apply knowledge of Data Science in real-time software project development using open-source and commercial programming environment to deliver quality software product for the organization's success.
PSO2	**Problem-Solving Skills:** The ability to acquaint with the current trends in industrial/research areas and there by innovate through novel solutions to the existing problems.

Head of the Department

Head of the Department
CSE (Data Science)
Lords Institute of Engg. & Tech.
Hyderabad-500091. T.S.

Course Outcomes: U23CD5L2 – Full Stack Web Development Lab

Student will able to

CO. No.	Description	Bloom's Taxonomy Level
C57.1	Develop structured HTML web pages incorporating embedded CSS for basic design.	BTL3
C57.2	Apply diverse CSS properties to construct responsive and visually engaging layouts.	BTL4
C57.3	Implement JavaScript code to introduce dynamic behaviours in web pages.	BTL3
C57.4	Validate HTML forms using client-side JavaScript and manage user input effectively.	BTL5
C57.5	Design and deploy complete web applications, demonstrating full-stack web development capabilities.	BTL6

Note: Bloom's Taxonomy Levels

BTL1-Remember	**BTL2-Understand**	**BTL3 - Apply**
BTL4-Analyze	**BT5 - Evaluate**	**BTL6 - Create**

Course Articulation Matrix:

Mapping of Course Outcomes (CO) with Program Outcomes (PO) and Program Specific Outcomes (PSO's):

Course Title: Full Stack Web Development Lab	**Course code:** U23CD5L2
Sem: V	**Academic Year:** 2025-26
Name of the Faculty: Ms. Zoya Tabassum	

Course Outcomes (CO's)	**Program Outcomes (PO's)**											**Program Specific Outcomes (PSO's)**		
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO 10**	**PO11**	**PO 12**	**PSO 1**	**PSO 2**
C357.1	3	2	2	1	2	0	0	0	1	1	0	0	3	2
C357.2	3	3	2	2	3	0	0	0	1	1	0	0	3	2
C357.3	3	3	2	2	3	1	0	0	1	1	0	0	3	3
C357.4	3	3	2	3	3	1	0	0	2	2	0	0	3	3
C357.5	3	3	3	3	3	2	1	1	2	3	1	1	3	3
Avg	3	2.8	2.2	2.2	2.8	0.8	0.2	0.2	1.4	1.6	0.2	0.2	3	2.6

Level: 1- Low correlation (Low), 2- Medium correlation (Medium), 3-High correlation (High)

PO1: Engineering knowledge,
PO2: Problem analysis,
PO3: Design Development of solutions,
PO4: Conduct investigations of complex problems,
PO5: Modern tool usage,
PO6: The engineer and society,
PO7: Ethics,
PO8: Individual and team work,
PO9: Communication,
PO10: Project management and finance,
PO11: Life-long learning
PSO1: Professional Skills,
PSO2: Problem-Solving Skills

Course Code	Course Title						Core/Elective
U23CD5L2	FULL STACK WEB DEVELOPMENT LAB						CORE
Prerequisite	Contact Hours Per Week				CIE	SEE	Credits
Python Programming	L	T	D	P			
	-	-	-	3	25	50	1.5

Course Objectives:

1. To understand the structure and syntax of HTML and CSS for designing static web pages.
2. To apply styling using CSS properties such as layout, positioning, typography, and responsiveness.
3. To gain knowledge of JavaScript and jQuery for adding interactivity and client-side logic to web pages.
4. To explore form handling and validation techniques using JavaScript.
5. To practice full-stack web application development and deployment using modern tools and frameworks.

Course Outcomes:

At the end of the course, student would be able to

1. Develop well-structured HTML web pages with embedded CSS styles.
2. Apply various CSS properties to create visually appealing and responsive layouts.
3. Write JavaScript programs to add dynamic behavior to web pages.
4. Validate forms using client-side JavaScript and handle user input efficiently.
5. Deploy web applications and demonstrate the ability to build full-stack web solutions.

1. Create a basic HTML page with a title, headings, paragraphs, a horizontal line, and a line break. Use <html>, <head>, <body>, <h1> to <h6>, <p>, <hr>, and
 tags. This experiment helps in understanding the structure of an HTML page.
2. Design an HTML form using various input types such as text boxes, radio buttons, checkboxes, submit and reset buttons. Implement labels and organize the form layout using tables or divs for better structure.
3. Create an HTML table with appropriate headings and data. Use rowspan and colspan to merge cells. Apply CSS styling to enhance table appearance like border, background-color, and text alignment.
4. Use internal or external CSS to apply properties such as font size, font family, background color, border thickness, margin, padding, and text alignment to a basic HTML page with content elements.
5. Style an HTML page using Inline CSS, Internal CSS, and External CSS.
6. Write a JavaScript program that uses variables, arithmetic operators, comparison operators, and logical operators.
7. Implement JavaScript control structures such as if-else and switch-case. Also, include looping structures like for, while, and do-while to print series or patterns based on conditions.
8. Create a registration or login form and apply JavaScript form validation to check email format, password strength, phone number length, and required fields before submission.
9. Show date and time using the Date object.
10. Perform pattern matching using RegExp class.
11. Use jQuery to show, hide, and toggle visibility of elements such as paragraphs, divs, and images on button click. Use $(selector).hide(), show(), and toggle() methods.
12. Implement slide animations using jQuery on different elements. Use slideDown(), slideUp(), and slideToggle() to demonstrate UI effects like drop-down menus or slide panels.
13. Install Node.js and write a basic Node.js program using console.log() and process object. Execute it using the command line and display output.

Text Books:

1. Ketan Agnihotri, Pranali Dahale, " React Development using Typescript: Modern web app development using advanced React techniques ", English Edition, 2024.
2. Nabendu Biswas, "Ultimate Full-Stack Web Development with MERN: Design, Build, Test and Deploy Production-Grade Web Applications with MongoDB, Express, React and NodeJS", 1st Edition, 2023.

Experiment No. 01

Q. Create a basic HTML page with a title, headings, paragraphs, a horizontal line, and a line break. Use <html>, <head>, <body>, <h1> to <h6>, <p>, <hr>, and
 tags. This experiment helps in understanding the structure of an HTML page.

```
<!DOCTYPE html>
<html>
<head>
   <title>My First HTML Page</title>
</head>
<body>

   <!-- Main Heading -->
   <h1>Welcome to My Website</h1>

   <!-- Horizontal Line -->
   <hr>

   <!-- Subheadings from h2 to h6 -->
   <h2>This is an H2 Heading</h2>
   <h3>This is an H3 Heading</h3>
   <h4>This is an H4 Heading</h4>
   <h5>This is an H5 Heading</h5>
   <h6>This is an H6 Heading</h6>

   <!-- Paragraph 1 -->
   <p>This is the first paragraph on my web page. HTML is the standard markup language for creating web
     pages.</p>

   <!-- Line Break Example -->
   <p>This is the second paragraph.<br>This sentence appears on the next line using the &lt;br&gt;
     tag.</p>

   <!-- Paragraph 3 -->
   <p>HTML stands for <b>HyperText Markup Language</b>. It is used to describe the structure of web
     pages using markup.</p>

   <!-- Horizontal Line -->
   <hr>

   <!-- Footer Heading -->
   <h4>Thank you for visiting!</h4>

</body>
</html>
```

Experiment No. 02

Q. Design an HTML form using various input types such as text boxes, radio buttons, and checkboxes, submit and reset buttons. Implement labels and organize the form layout using tables or divs for better structure.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Form Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f9f9f9;
            padding: 20px;
        }
        table {
            background-color: #ffffff;
            padding: 20px;
            border-collapse: collapse;
            width: 50%;
            margin: auto;
            box-shadow: 0px 0px 10px rgba(0,0,0,0.1);
        }
        td {
            padding: 10px;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        input[type="submit"], input[type="reset"] {
            padding: 8px 16px;
            background-color: #4CAF50;
            border: none;
            color: white;
            cursor: pointer;
            border-radius: 4px;
        }
        input[type="reset"] {
            background-color: #f44336;
        }
    </style>
</head>
<body>
    <h2>Student Registration Form</h2>

    <form action="#" method="post">
```

```
        <table border="0">
            <tr>
                <td><label for="fname">First Name:</label></td>
                <td><input type="text" id="fname" name="fname" required></td>
            </tr>
            <tr>
                <td><label for="lname">Last Name:</label></td>
                <td><input type="text" id="lname" name="lname" required></td>
            </tr>
            <tr>
                <td><label for="email">Email:</label></td>
                <td><input type="email" id="email" name="email" required></td>
            </tr>
            <tr>
                <td><label>Gender:</label></td>
                <td>
                    <input type="radio" id="male" name="gender" value="Male">
                    <label for="male">Male</label>
                    <input type="radio" id="female" name="gender"
value="Female">
                    <label for="female">Female</label>
                </td>
            </tr>
            <tr>
                <td><label>Hobbies:</label></td>
                <td>
                    <input type="checkbox" id="reading" name="hobby"
value="Reading">
                    <label for="reading">Reading</label><br>
                    <input type="checkbox" id="sports" name="hobby"
value="Sports">
                    <label for="sports">Sports</label><br>
                    <input type="checkbox" id="music" name="hobby"
value="Music">
                    <label for="music">Music</label>
                </td>
            </tr>
            <tr>
                <td><label for="address">Address:</label></td>
                <td><textarea id="address" name="address" rows="4"
cols="30"></textarea></td>
            </tr>
            <tr>
                <td colspan="2" style="text-align:center;">
                    <input type="submit" value="Submit">
                    <input type="reset" value="Reset">
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Experiment No. 03:

Q. Create an HTML table with appropriate headings and data. Use rowspan and colspan to merge cells. Apply CSS styling to enhance table appearance like border, background-color, and text alignment.

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Table with Merged Cells</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      background-color: #f2f2f2;
    }

    h2 {
      text-align: center;
      color: #333;
    }

    table {
      width: 80%;
      margin: auto;
      border-collapse: collapse;
      background-color: #fff;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }

    th, td {
      border: 1px solid #333;
      padding: 12px;
      text-align: center;
    }

    th {
      background-color: #4CAF50;
      color: white;
    }
    tr:nth-child(even) {
      background-color: #f9f9f9;
    }

    tr:hover {
      background-color: #e0f7fa;
    }
  </style>
</head>
        <body>
```

```
<h2>Student Grades Table</h2>
<table>
  <tr>
    <th rowspan="2">Student Name</th>
    <th colspan="3">Subjects</th>
    <th rowspan="2">Total</th>
  </tr>
  <tr>
    <th>Math</th>
    <th>Science</th>
    <th>English</th>
  </tr>
  <tr>
    <td>John Doe</td>
    <td>85</td>
    <td>78</td>
    <td>92</td>
    <td>255</td>
  </tr>
  <tr>
    <td>Jane Smith</td>
    <td>90</td>
    <td>88</td>
    <td>95</td>
    <td>273</td>
  </tr>
  <tr>
    <td>Sam Wilson</td>
    <td colspan="2">Absent</td>
    <td>76</td>
    <td>-</td>
  </tr>
</table>

</body>
</html>
```

Experiment No. 04

Q. Use internal or external CSS to apply properties such as font size, font family, background color, border thickness, margin, padding, and text alignment to a basic HTML page with content elements.

```
<!DOCTYPE html>
<html>
<head>
   <title>Styled Web Page</title>
    <!-- In HTML file -->
   <!--link rel="stylesheet" href="styles.css"-->
   <style>
      /* Body Styling */
      body {
         background-color: #f0f8ff; /* Light blue background */
         font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
         margin: 40px;
         padding: 20px;
      }

      /* Main Heading Styling */
      h1 {
         font-size: 32px;
         color: #2c3e50;
         text-align: center;
         padding: 10px;
         margin-bottom: 20px;
         border: 2px solid #2c3e50;
         background-color: #dff0d8;
      }

      /* Paragraph Styling */
      p {
         font-size: 18px;
         color: #333;
         line-height: 1.6;
         margin: 15px 0;
         padding: 10px;
         background-color: #ffffff;
         border: 1px solid #ccc;
         border-radius: 8px;
      }

      /* Container Div Styling */
      .container {
         border: 3px solid #4CAF50;
         padding: 20px;
         background-color: #eafaf1;
```

```
            border-radius: 12px;
        }

        /* Center Alignment for Footer */
        .footer {
            text-align: center;
            margin-top: 30px;
            font-size: 14px;
            color: #888;
        }
    </style>
</head>
<body>

    <h1>Welcome to My Styled Web Page</h1>

    <div class="container">
        <p>This paragraph demonstrates the use of internal CSS to change font size, family, background color, and padding. You can see the text is easier to read and visually separated using padding and background color.</p>

        <p>We also added borders with rounded corners and different margin spacing. These design features help improve the structure and appearance of the web content.</p>
    </div>

    <div class="footer">
        &copy; 2025 MyWebDesigns. All Rights Reserved.
    </div>

</body>
</html>
```

Output:

Experiment No. 05

Q. Style an HTML page using Inline CSS, Internal CSS, and External CSS.

```
/* style.css - External CSS */
body {
    background-color: #f4f8ff;
    font-family: Arial, sans-serif;
    padding: 20px;
}

.external-style {
    color: #2e8b57;
    font-size: 20px;
    border: 2px dashed #2e8b57;
    padding: 10px;
    margin-top: 15px;
    background-color: #e0f7fa;
}
<!DOCTYPE html>
<html>
<head>
    <title>CSS Styling Methods</title>

    <!-- Linking External CSS -->
    <link rel="stylesheet" href="style.css">

    <!-- Internal CSS -->
    <style>
        h1 {
            color: #4a148c;
            text-align: center;
            font-size: 32px;
            padding: 10px;
            background-color: #ede7f6;
```

```
        }

        .internal-style {
            background-color: #fff9c4;
            padding: 15px;
            border: 2px solid #fbc02d;
            margin: 15px 0;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <!-- Inline CSS -->
    <p style="color: blue; font-size: 18px; background-color: #d0f0fd; padding: 10px;
border: 1px solid #0099cc;">
        This paragraph uses <strong>Inline CSS</strong>. Styles are applied directly using
the "style" attribute.
    </p>
    <!-- Internal CSS -->
    <div class="internal-style">
        This content uses <strong>Internal CSS</strong>. The styles are written inside a
&lt;style&gt; block in the &lt;head&gt;.
    </div>
    <!-- External CSS -->
    <div class="external-style">
        This content uses <strong>External CSS</strong>. The styles are linked from an
external file called <code>style.css</code>.
    </div>
</body>
</html>
```

Output:

Experiment No. 06

Q. Write a JavaScript program that uses variables, arithmetic operators, comparison operators, and logical operators.

```
<!DOCTYPE html>
<html>
<head>
   <title>JavaScript Operators Demo</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         padding: 20px;
         background-color: #f0f8ff;
      }
      h2 {
         color: #333;
      }
      .output {
         background-color: #fff;
         border: 2px solid #ccc;
         padding: 15px;
         margin-top: 10px;
         font-size: 16px;
      }
   </style>
</head>
<body>

   <h2>JavaScript: Variables and Operators</h2>
   <div class="output" id="result"></div>

   <script>
      // ✸ Variable Declaration
      let a = 10;
      let b = 5;

      // ✚ Arithmetic Operators
      let sum = a + b;
      let difference = a - b;
      let product = a * b;
      let quotient = a / b;
      let remainder = a % b;

      // Q Comparison Operators
      let isEqual = (a == b);          // false
      let isNotEqual = (a != b);       // true
```

```
        let isGreater = (a > b);         // true
        let isLessOrEqual = (a <= b);    // false

        // 🔀 Logical Operators
        let andResult = (a > 0 && b > 0); // true
        let orResult = (a < 0 || b > 0);  // true
        let notResult = !(a == b);        // true

        // 🖨️ Output to Page
        document.getElementById("result").innerHTML =
            <strong>Values:</strong><br>
            a = ${a}, b = ${b} <br><br>

            <strong>Arithmetic Operations:</strong><br>
            a + b = ${sum} <br>
            a - b = ${difference} <br>
            a * b = ${product} <br>
            a / b = ${quotient} <br>
            a % b = ${remainder} <br><br>

            <strong>Comparison Results:</strong><br>
            a == b: ${isEqual} <br>
            a != b: ${isNotEqual} <br>
            a > b: ${isGreater} <br>
            a <= b: ${isLessOrEqual} <br><br>

            <strong>Logical Operations:</strong><br>
            (a > 0 && b > 0): ${andResult} <br>
            (a < 0 || b > 0): ${orResult} <br>
            !(a == b): ${notResult}
    </script>

</body>
</html>
```

Output:

Experiment No. 07

Q. Implement JavaScript control structures such as if-else and switch-case. Also, include looping structures like for, while, and do-while to print series or patterns based on conditions.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>JavaScript Control Structures and Loops</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #eef7ff;
      text-align: center;
      padding: 20px;
    }

    h2 {
      color: #1a237e;
      margin-bottom: 10px;
    }

    .container {
      display: flex;
      justify-content: center;
      flex-wrap: wrap;
      gap: 20px;
    }

    .input-box, .output-box {
      background-color: #ffffff;
      padding: 25px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
      width: 400px;
    }

    .input-box input {
      display: block;
      width: 90%;
      margin: 10px auto;
      padding: 10px;
      border-radius: 5px;
      border: 1px solid #ccc;
```

```
    font-size: 16px;
  }

  .input-box button {
    padding: 10px 20px;
    background-color: #1a237e;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
  }

  .input-box button:hover {
    background-color: #3949ab;
  }

  .output-box {
    text-align: left;
  }

  .output-box h3 {
    color: #3949ab;
    border-bottom: 1px solid #ccc;
    padding-bottom: 5px;
  }

  .output-content {
    font-size: 15px;
    line-height: 1.6;
    margin-top: 10px;
  }
 </style>
</head>
<body>

 <h2>JavaScript: Control Structures and Loops</h2>

 <div class="container">
  <!-- Input Section -->
  <div class="input-box">
   <h3>Enter Inputs</h3>
   <input type="number" id="numberInput" placeholder="Enter a number">
   <input type="text" id="gradeInput" placeholder="Enter grade (A/B/C/D)">
   <input type="number" id="loopLimit" placeholder="Enter loop limit (e.g., 10)">
   <button onclick="displayResults()">Run</button>
  </div>
  <!-- Output Section -->
  <div class="output-box">

  <h3>Output</h3>
   <div class="output-content" id="resultArea">Your result will appear here...</div>
  </div>
```

```
</div>

<script>
  function displayResults() {
    const number = parseInt(document.getElementById("numberInput").value);
    const grade = document.getElementById("gradeInput").value.trim().toUpperCase();
    const limit = parseInt(document.getElementById("loopLimit").value);
    let output = "";

    if (isNaN(number) || isNaN(limit)) {
      alert("Please enter valid number and loop limit.");
      return;
    }

    // If-Else Statement
    output += `<strong>📄 If-Else:</strong><br>`;
    if (number > 0) {
      output += `The number ${number} is positive.<br>`;
    } else if (number < 0) {
      output += `The number ${number} is negative.<br>`;
    } else {
      output += `The number is zero.<br>`;
    }

    // Switch-Case
    output += `<br><strong>➵ Switch-Case (Grade):</strong><br>`;
    switch (grade) {
      case "A": output += "Excellent!<br>"; break;
      case "B": output += "Good job!<br>"; break;
      case "C": output += "You passed.<br>"; break;
      case "D": output += "Try again.<br>"; break;
      default: output += "Invalid Grade.<br>";
    }

    // For Loop
    output += `<br><strong>🔁 For Loop (1 to ${limit}):</strong><br>`;
    for (let i = 1; i <= limit; i++) {
      output += i + " ";
    }

    // While Loop - Even numbers up to limit
    output += `<br><br><strong>♻ While Loop (Even Numbers to ${limit}):</strong><br>`;
    let i = 2;
    while (i <= limit) {
      output += i + " ";
      i += 2;
    }

    // Do-While Loop - Countdown from limit
    output += `<br><br><strong>🔄 Do-While Loop (Reverse ${limit} to 1):</strong><br>`;
    let j = limit;
    do {
```

```
      output += j + " ";
      j--;
    } while (j > 0);

    document.getElementById("resultArea").innerHTML = output;
  }
 </script>

</body>
</html>
```

Experiment No. 08

Q.Create a registration or login form and apply JavaScript form validation to check email format, password strength, phone number length, and required fields before submission.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Registration Form with Validation</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #eef2f3;
      padding: 20px;
    }
    form {
      width: 400px;
      margin: auto;
      background: #fff;
      padding: 20px;
      border: 2px solid #ccc;
      border-radius: 8px;
    }
    h2 {
      text-align: center;
      color: #333;
    }
    .form-group {
      margin-bottom: 15px;
    }
    label {
      display: block;
      margin-bottom: 5px;
      font-weight: bold;
    }
    input[type="text"],
    input[type="email"],
    input[type="password"],
    input[type="tel"] {
      width: 100%;
```

```
      padding: 8px;
      border: 1px solid #999;
      border-radius: 4px;
    }
    input[type="submit"] {
      width: 100%;
      padding: 10px;
      background-color: #4CAF50;
      border: none;
      color: #fff;
      font-size: 16px;
      cursor: pointer;
      border-radius: 5px;
    }
    .error {
      color: red;
      font-size: 14px;
      margin-top: 5px;
    }
    .success {
      color: green;
      font-size: 16px;
      text-align: center;
      margin-top: 20px;
    }
  </style>
</head>
<body>

  <form id="registrationForm" novalidate>
    <h2>Registration Form</h2>

    <div class="form-group">
      <label for="fname">Full Name *</label>
      <input type="text" id="fname" name="fname">
      <div class="error" id="fnameError"></div>
    </div>

    <div class="form-group">
      <label for="email">Email *</label>
      <input type="email" id="email" name="email">
      <div class="error" id="emailError"></div>
    </div>

    <div class="form-group">
      <label for="password">Password * (min 8 chars, include letters &
numbers)</label>
      <input type="password" id="password" name="password">
      <div class="error" id="passwordError"></div>
    </div>

    <div class="form-group">
      <label for="phone">Phone Number * (10 digits)</label>
```

```
      <input type="tel" id="phone" name="phone">
      <div class="error" id="phoneError"></div>
    </div>

    <input type="submit" value="Register">
    <div class="success" id="successMsg"></div>
  </form>

  <script>
    document.getElementById('registrationForm').addEventListener('submit',
function(e) {
      e.preventDefault();  // Prevent form submission

      // Clear previous error messages
      document.getElementById('fnameError').textContent = '';
      document.getElementById('emailError').textContent = '';
      document.getElementById('passwordError').textContent = '';
      document.getElementById('phoneError').textContent = '';
      document.getElementById('successMsg').textContent = '';

      // Grab field values
      const fname = document.getElementById('fname').value.trim();
      const email = document.getElementById('email').value.trim();
      const password = document.getElementById('password').value;
      const phone = document.getElementById('phone').value.trim();

      let isValid = true;

      // 1. Name Validation
      if (fname === '') {
        document.getElementById('fnameError').textContent = 'Full name is
required.';
        isValid = false;
      }

      // 2. Email Validation
      const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (email === '') {
        document.getElementById('emailError').textContent = 'Email is
required.';
        isValid = false;
      } else if (!emailPattern.test(email)) {
        document.getElementById('emailError').textContent = 'Invalid email
format.';
        isValid = false;
      }

      // 3. Password Validation
      if (password.length < 8) {
        document.getElementById('passwordError').textContent = 'Password must
be at least 8 characters.';
        isValid = false;
      } else if (!(/[A-Za-z]/.test(password) && /\d/.test(password))) {
```

```
        document.getElementById('passwordError').textContent = 'Include both
letters and numbers.';
        isValid = false;
      }

      // 4. Phone Validation
      const phonePattern = /^\d{10}$/;
      if (phone === '') {
        document.getElementById('phoneError').textContent = 'Phone number is
required.';
        isValid = false;
      } else if (!phonePattern.test(phone)) {
        document.getElementById('phoneError').textContent = 'Phone must be
exactly 10 digits.';
        isValid = false;
      }
      // If all checks pass, show success message
      if (isValid) {
        document.getElementById('successMsg').textContent = 'Registration
successful!';
        // Here you can also submit the form or clear inputs
       }
    });
  </script>

</body>
</html>
```

Experiment No. 09

Q. Show date and time using the Date object.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Display Current Date and Time</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #eef;
      text-align: center;
      padding-top: 100px;
    }
    #datetime {
      font-size: 24px;
      color: #333;
      padding: 20px;
      border: 2px solid #555;
      display: inline-block;
      background-color: #fff;
      border-radius: 10px;
      box-shadow: 2px 2px 10px rgba(0,0,0,0.1);
    }
  </style>
</head>
<body>
  <div id="datetime">
    <!-- Date and time will appear here -->
  </div>
  <script>
    function showDateTime() {
      const now = new Date(); // Create Date object

      const date = now.toLocaleDateString();  // Get date (format: MM/DD/YYYY)
      const time = now.toLocaleTimeString();  // Get time (format: HH:MM:SS AM/PM)

      document.getElementById("datetime").innerHTML =
        "Today's Date: " + date + "<br>Current Time: " + time;
    }
    // Show date and time immediately
    showDateTime();
```

```
    // Update time every second
    setInterval(showDateTime, 1000);
  </script>

</body>
</html>
```

Experiment No. 10

Q. Perform pattern matching using RegExp class.

```
<!DOCTYPE html>
<html>
<head>
   <title>JavaScript RegExp Pattern Matching</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         background: #f2f2f2;
         padding: 40px;
      }

      h2 {
         color: #333;
      }

      .result {
         background-color: #fff;
         padding: 20px;
         border-radius: 8px;
         margin-top: 20px;
         border: 1px solid #ccc;
         font-size: 16px;
      }

      .highlight {
         color: green;
         font-weight: bold;
      }

      .error {
         color: red;
      }
   </style>
</head>
<body>

   <h2>Pattern Matching using JavaScript RegExp</h2>

   <div class="result" id="output"></div>
```

```
    <script>
        // Sample text to match patterns
        let sampleText = "Hello! My email is test@example.com and my phone is 9876543210.";

        // Pattern 1: Match Email
        let emailPattern = /\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b/;
        let emailMatch = sampleText.match(emailPattern);

        // Pattern 2: Match Phone Number (10 digits)
        let phonePattern = /\b\d{10}\b/;
        let phoneMatch = sampleText.match(phonePattern);

        // Pattern 3: Replace word "Hello" with "Hi"
        let greetingPattern = /Hello/;
        let newText = sampleText.replace(greetingPattern, "Hi");

        // Pattern 4: Check if string contains word "email"
        let containsEmail = /email/.test(sampleText);

        // Output results
        let resultHTML = `
            <p><strong>Original Text:</strong> ${sampleText}</p>
            <p><strong>Email Found:</strong> <span class="highlight">${emailMatch}</span></p>
            <p><strong>Phone Number Found:</strong> <span class="highlight">${phoneMatch}</span></p>
            <p><strong>After Replacing Greeting:</strong> ${newText}</p>
            <p><strong>Contains word 'email'?</strong> ${containsEmail ? "<span
class='highlight'>Yes</span>" : "<span class='error'>No</span>"}</p>
        `;

        document.getElementById("output").innerHTML = resultHTML;
    </script>

</body>
</html>
```

Experiment No. 11

Q. Use jQuery to show, hide, and toggle visibility of elements such as paragraphs, divs, and images on button click. Use $(selector).hide(), show(), and toggle() methods.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>jQuery Show, Hide, and Toggle Example</title>
  <!-- jQuery CDN -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 30px;
      background-color: #f2f2f2;
      text-align: center;
    }

    .box, p, img {
      margin: 20px auto;
      width: 300px;
      padding: 20px;
      border: 2px solid #ccc;
      border-radius: 10px;
      background-color: #fff;
    }

    img {
      width: 20%;
    }

    button {
      margin: 10px;
      padding: 10px 20px;
      font-size: 16px;
      cursor: pointer;
    }
  </style>
</head>
<body>

  <h2>Using jQuery Show, Hide and Toggle</h2>

  <!-- Paragraph -->
  <p id="myPara">This is a paragraph. Click buttons below to hide, show, or toggle me!</p>

  <!-- Div Box -->
  <div class="box" id="myBox">This is a div box.</div>
```

```
<!-- Image -->
<img id="myImg" src="H:\SPM\Full Stack\Lab codes\image.jpg" alt="Placeholder Image" />

<!-- Buttons -->
<div>
  <button onclick="$('#myPara').hide();">Hide Paragraph</button>
  <button onclick="$('#myPara').show();">Show Paragraph</button>
  <button onclick="$('#myPara').toggle();">Toggle Paragraph</button>
</div>

<div>
  <button onclick="$('#myBox').hide();">Hide Div</button>
  <button onclick="$('#myBox').show();">Show Div</button>
  <button onclick="$('#myBox').toggle();">Toggle Div</button>
</div>

<div>
  <button onclick="$('#myImg').hide();">Hide Image</button>
  <button onclick="$('#myImg').show();">Show Image</button>
  <button onclick="$('#myImg').toggle();">Toggle Image</button>
</div>

</body>
</html>
```

Experiment No. 12

Q.*Implement slide animations using jQuery on different elements. Use slideDown(), slideUp(), and slideToggle() to demonstrate UI effects like drop-down menus or slide panels.*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>jQuery Slide Animations</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background: linear-gradient(to right, #e3f2fd, #fce4ec);
      margin: 0;
      padding: 40px 20px;
      display: flex;
      flex-direction: column;
      align-items: center;
      min-height: 100vh;
    }

    h2 {
      color: #333;
      margin-bottom: 30px;
    }

    .btn-container {
      display: flex;
      justify-content: center;
      gap: 15px;
      flex-wrap: wrap;
      margin-bottom: 20px;
    }

    button {
      padding: 10px 25px;
      background-color: #2196f3;
      border: none;
      color: #fff;
      border-radius: 6px;
      font-size: 16px;
      transition: all 0.3s ease;
      cursor: pointer;
    }

    button:hover {
      background-color: #0b7dda;
    }

    .panel {
```

```
  display: none;
  padding: 20px;
  background-color: #ffffff;
  border-left: 5px solid #2196f3;
  border-radius: 6px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
  width: 90%;
  max-width: 500px;
  text-align: center;
}

#dropdownMenu  {
  margin-top:  40px;
  text-align: center;
}

#dropdownBtn {
  padding: 10px 20px;
  background-color: #4caf50;
  border: none;
  color: #fff;
  font-size: 16px;
  border-radius: 6px;
  cursor: pointer;
  transition: background 0.3s;
}

#dropdownBtn:hover {
  background-color: #388e3c;
}

#dropdownItems {
  display: none;
  margin-top: 10px;
  padding: 0;
  list-style-type: none;
  border: 1px solid #ccc;
  border-radius: 5px;
  background-color: #ffffff;
  width: 180px;
  margin: 10px auto 0;
  box-shadow: 0 3px 8px rgba(0, 0, 0, 0.1);
}

#dropdownItems li {
  padding: 10px;
  border-bottom: 1px solid #eee;
  transition: background 0.2s;
}

#dropdownItems li:last-child {
  border-bottom: none;
}
```

```
    #dropdownItems li:hover {
      background-color: #f1f1f1;
      cursor: pointer;
    }
  </style>
</head>
<body>

  <h2>jQuery Slide Animation Example</h2>

  <!-- Slide Panel -->
  <div class="btn-container">
    <button onclick="$('#slidePanel').slideDown()">Slide Down</button>
    <button onclick="$('#slidePanel').slideUp()">Slide Up</button>
    <button onclick="$('#slidePanel').slideToggle()">Toggle Slide</button>
  </div>

  <div id="slidePanel" class="panel">
    <p>This panel demonstrates jQuery slide animations like <strong>slideDown</strong>,
<strong>slideUp</strong>, and <strong>slideToggle</strong>.</p>
  </div>

  <!-- Drop-down Menu -->
  <div id="dropdownMenu">
    <button id="dropdownBtn">Toggle Dropdown</button>
    <ul id="dropdownItems">
      <li>Home</li>
      <li>About</li>
      <li>Contact</li>
    </ul>
  </div>

  <script>
    $(document).ready(function () {
      $("#dropdownBtn").click(function () {
        $("#dropdownItems").slideToggle();
      });
    });
  </script>

</body>
</html>
```

Output:

Experiment No. 13

Q. Install Node.js and write a basic Node.js program using console.log() and process object. Execute it using the command line and display output.

Step 1: Install Node.js

1. **Go to:** https://nodejs.org
2. Download the **LTS version** (Long-Term Support).
3. Run the installer and follow the instructions.
4. After installation, open **Command Prompt (CMD)** or **Terminal** and run:

```
node –v
```

Step 2: Write a Basic Node.js Program

Create a new file named: app.js

```
// app.js

// Display a message using console.log
console.log("Hello, welcome to Node.js!");

// Use the process object to display runtime information
console.log("Process Version: " + process.version);
console.log("Platform: " + process.platform);
console.log("CPU Architecture: " + process.arch);

// Access command line arguments
console.log("Command Line Arguments:");
process.argv.forEach((val, index) => {
console.log(`${index}: ${val}`);
});
```

Step 3: Run the Program via Command Line

Open **Command Prompt** or **Terminal** in the folder where app.js is saved.

Type the following command:

```
node app.js hello world
```

```
Hello, welcome to Node.js!
Process Version: v20.11.1
Platform: win32
CPU Architecture: x64
Command Line Arguments:
0: C:\Program Files\nodejs\node.exe
1: C:\Users\YourName\app.js
2: hello
3: world
```