MULTIMEDIA TECHNOLOGY (LAB)

Contents: -

Unit- 01: - Create web page using structure tags to display sample message.

Unit- 02: - Create a web page for displaying a paragraph using block level tags, HR tags.

Unit- 03: - Create a web page for implementing different types of Lists.

Unit- 04: - Create a web page to link-

a) A different web page of same site.

b) A different location on the same web page

c) A specific location on different web page of same site.

Unit- 05: - Insert images on web page using various attributes.

Unit- 06: - Create a web page to implement Frame tags, Tables tags.

Unit- 07: - Create a web page for demonstration of CSS by applying V 06* Internal/External/ Inline style.

Unit- 08: - Install a web server and publish a website on Intranet and publish It on internet.

Unit- 09: - Design a Visiting Card containing at least one graphic and Text information.

Unit- 10: - You are given a picture of a garden as background. Extract the image of a butterfly from another picture and organize it on the background.

Unit- 11: - Shape Distortion: Create a square and gradually convert it into a circle.

Unit- 12: - Spotlight: Create a text on one layer; format the text with suitable size, color and style. With the help of another layer, position a spotlight on the text and move the spotlight from left to right.

Unit- 13: - "Simulation of a Raindrop: In the first layer, draw a raindrop that falls on the ground. Show the splash effect, when it touches the ground on another layer."

Unit- 01

Create web page using structure tags to display sample message.

Objective:

To create a simple web page using HTML5 structure tags such as:

- <header>
- <nav>
- <main>
- <section>
- <article>
- <footer>

The page will display a sample message in a proper semantic layout.

Requirements:

- A text editor (e.g., VS Code, Sublime Text, Notepad++)
- A modern web browser (e.g., Chrome, Firefox)
- Basic knowledge of HTML

Lab Procedure:

1. Open your text editor

Start a new HTML file and save it as structure-tags.html.

2. Write the basic HTML5 structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Sample Structured Web Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      line-height: 1.6;
    }
    header, nav, main, footer {
```

```
            padding: 10px;
            margin: 10px;
            border: 1px solid #ccc;
        }
        nav ul {
            list-style: none;
            padding: 0;
        }
        nav ul li {
            display: inline;
            margin-right: 15px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Welcome to My Sample Page</h1>
        <p>This page demonstrates the use of HTML5 structure tags.</p>
    </header>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
    <main>
        <section>
            <h2>Sample Message</h2>
            <article>
```

```
            <p>Hello! This is a sample message displayed using HTML5 structural elements. These semantic tags help structure the content clearly and improve readability and accessibility. </p>
        </article>
    </section>
</main>
<footer>
    <p>&copy; 2025 Your Name. All rights reserved. </p>
</footer>
</body>
</html>
```

Expected Output:

A simple webpage with:

- A **header** introducing the page
- A **navigation bar**
- A **main** content area with a **section** and an **article** showing the message
- A **footer** with copyright

Unit- 02

Create a web page for displaying a paragraph using block level tags, HR tags.

Objective:

To design a basic HTML web page that demonstrates the use of:

- Block-level tags such as , <p>, <h1> to <h6>, <blockquote>
- The horizontal rule (<hr>) tag

Requirements:

- A text editor (e.g., VS Code, Notepad++)
- A modern web browser (e.g., Chrome, Firefox)

Lab Procedure:

1. Open your text editor

Create a new file and save it as:
block-tags-demo.html

2. Write the HTML Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Block-Level Tags Demo</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    div {
      border: 1px solid #ccc;
      padding: 15px;
      margin-bottom: 20px;
    }
  </style>
</head>
<body>
  <h1>Demonstration of Block-Level Tags and &lt;hr&gt;</h1>
  <hr>
  <div>
    <h2>Introduction</h2>
    <p>This paragraph is wrapped inside a block-level <code>&lt;div&gt;</code> element. Block-level tags start on a new line and take up the full width available. </p>
  </div>
  <hr>
  <div>
```

```
    <h2>Blockquote Example</h2>
    <blockquote>
      "This is an example of a blockquote tag, which is used for longer quotations that should be set apart from the main content."
    </blockquote>
  </div>
  <hr>
  <div>
    <h2>Conclusion</h2>
    <p>Using block-level tags helps structure the web content clearly. The <code>& lt; hr & gt; </code> tag adds a visual break or separation between content sections. </p>
  </div>
</body>
</html>
```

Expected Output:

A simple webpage with:

- A heading at the top
- Multiple sections using and <p> tags
- Horizontal lines (<hr>) separating each content block
- A styled blockquote section

Unit- 03

Create a web page for implementing different types of Lists.

Objective:

To design a web page using **HTML list tags**, including:

- **Ordered lists (<ol>)**
- **Unordered lists (<ul>)**
- **Description lists (<dl>)**

Requirements:

- A code editor (e.g., VS Code, Sublime Text, Notepad++)
- A web browser (e.g., Chrome, Firefox, Edge)
- Basic knowledge of HTML

Lab Procedure:

1. **Open your text editor**
   Create a new HTML file and save it as:
   `html-lists.html`
2. **Write the HTML code as follows:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML Lists Demo</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 30px;
    }
    h2 {
      color: #333;
    }
    section {
      margin-bottom: 30px;
    }
  </style>
</head>
<body>
  <h1>Different Types of Lists in HTML</h1>
  <section>
    <h2>1. Ordered List (Numbered)</h2>
    <ol>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
      <li>React</li>
```

```
    </ol>
  </section>
  <section>
    <h2>2. Unordered List (Bulleted)</h2>
    <ul>
      <li>Apples</li>
      <li>Bananas</li>
      <li>Cherries</li>
      <li>Dates</li>
    </ul>
  </section>
  <section>
    <h2>3. Description List</h2>
    <dl>
      <dt>HTML</dt>
      <dd>A markup language used to create web pages. </dd>
      <dt>CSS</dt>
      <dd>Used for styling HTML content. </dd>
      <dt>JavaScript</dt>
      <dd>A scripting language used to add interactivity to web pages. </dd>
    </dl>
  </section>
</body>
</html>
```

Expected Output:

- A webpage showing:
- A **numbered list** of web technologies
- A **bulleted list** of fruits
- A **description list** explaining some web terms.

Unit- 04

Create a web page to link-

a) A different web page of same site.

b) A different location on the same web page

c) A specific location on different web page of same site.

Objective:

To create a web page that implements:

- Internal links to other pages in the same website
- Anchor links to specific sections within the same page
- Links to a specific section on a different page of the same website

Requirements:

- A text/code editor (VS Code, Notepad++, etc.)
- A browser (Chrome, Firefox, etc.)
- At least two HTML files for demonstration

Lab Procedure

Step 1: Create Two HTML Files

- index.html
- about.html

1. index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Link Demo - Home</title>
</head>
<body>

   <h1>Welcome to My Website</h1>

   <h2>Types of Links</h2>
   <ul>
      <li><a href="about.html">Link to another page (about.html) </a></li> <! -- (a) -->
      <li><a href="#section2">Link to Section 2 on the same page</a></li> <! -- (b) -->
      <li><a href="about.html#team">Link to 'Team' section on about.html</a></li> <! --
(c) -->
```

```
    </ul>

    <h2 id="section1">Section 1</h2>
    <p>This is section 1 of the home page. </p>

    <h2 id="section2">Section 2</h2>
    <p>This is section 2 of the home page, which is linked from the menu above. </p>

</body>
</html>
```

2. about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Link Demo - About</title>
</head>
<body>

    <h1>About This Website</h1>

    <p>This is a separate page to demonstrate internal links. </p>

    <a href="index.html">Back to Home</a>

    <h2 id="team">Our Team</h2>
    <p>This is the 'Team' section of the about page, which can be linked directly using an anchor. </p>
</body>
</html>
```

Experiment Summary

Type of Link	HTML Tag Used	Example
a) Link to different page	<a href="about.html">	Goes to About page
b) Link to same-page section	<a href="#section2">	Scrolls to section2
c) Link to section on another page	<a href="about.html#team">	Goes to About → Team

Expected Output:

- Clicking the first link opens about.html
- Clicking the second scrolls to "Section 2" in the same page
- Clicking the third opens about.html and jumps to the "Team" section

Unit- 05

Insert images on web page using various attributes.

Objective

To learn how to insert images into an HTML web page using the <img> tag and explore various attributes to control image behavior and appearance.

Tools Required

- Text Editor (VS Code, Sublime Text, Notepad++)
- Web Browser (Chrome, Firefox, Edge)
- Sample images (local or online URLs)

HTML `<img>` Tag: Overview

- The basic syntax:
  <img src="image.jpg" alt="Description of image">

Common Attributes:

Attribute	Description
src	Source of the image (URL or local path)
alt	Alternate text (for accessibility/fallback)
width	Width of the image (pixels or %)
height	Height of the image (pixels or %)
title	Tooltip text shown on hover
style	Inline CSS styling
loading	Lazy loading behaviour
usemap	To define image maps

Exercise: Basic Image Insertion using various attributes.

```
<!DOCTYPE html>
<html>
<head>
  <title>Basic Image</title>
</head>
<body>
  <h2>Basic Image Example</h2>
```

<img src="images/sample.jpg" alt="Sample Image" width="300" height="200" alt="Image with Tooltip" title="This is a tooltip" alt="Placeholder" style="border: 2px solid black; border-radius: 10px;" loading="lazy" width="500" alt="Responsive Image" width="100%" >

</body>

</html>

Experiment	Description	Observed Result
1	Basic image	Image loads
2	Resized image	Correct size
3	Tooltip	Tooltip appears
...	...	...

Unit- 06

Create a web page to implement Frame tags, Tables tags.

Objective:

To create a web page that demonstrates the use of:

- HTML **Frame** tags for dividing the browser window
- HTML **Table** tags to display structured data

Tools Required:

- A simple text editor (Notepad, VS Code, Sublime Text)
- A web browser (Chrome, Firefox, Edge)

Files Required:

You will create **4 HTML files**:

1. main.html – Uses frameset
2. menu.html – Left-side frame (Navigation or Menu)
3. content.html – Right-side frame (Contains a table)
4. table.html – (Can be embedded in content.html)

Part A: Using Frame Tags

1. **main.html**
   <!DOCTYPE html>
   <html>

```
<head>
  <title>Frameset Example</title>
</head>
<frameset cols="30%,70%">
  <frame src="menu.html" name="menuFrame">
  <frame src="content.html" name="contentFrame">
</frameset>
</html>
```

2. menu.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Menu</title>
</head>
<body bgcolor="#f0f0f0">
  <h2>Navigation</h2>
  <ul>
    <li><a href="content.html" target="contentFrame">Home</a></li>
    <li><a href="table.html" target="contentFrame">View Table</a></li>
  </ul>
</body>
</html>
```

3. content.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Welcome</title>
</head>
<body>
  <h1>Welcome to Frame and Table Example</h1>
  <p>Click the menu to view the table in this area. </p>
</body>
</html>
```

Part B: Using Table Tags

4. table.html

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Table Example</title>
  <style>
    table {
      width: 80%;
      border-collapse: collapse;
```

```
        }
        th, td {
            border: 1px solid #333;
            padding: 8px;
            text-align: center;
        }
        th {
            background-color: #ddd;
        }
    </style>
</head>
<body>
    <h2>Student Marks Table</h2>
    <table>
        <tr>
            <th>Roll No</th>
            <th>Name</th>
            <th>Subject</th>
            <th>Marks</th>
        </tr>
        <tr>
            <td>101</td>
            <td>Alice</td>
            <td>Math</td>
            <td>95</td>
        </tr>
        <tr>
            <td>102</td>
            <td>Bob</td>
            <td>Science</td>
            <td>88</td>
        </tr>
        <tr>
            <td>103</td>
            <td>Charlie</td>
            <td>English</td>
            <td>92</td>
        </tr>
    </table>
</body>
</html>
```

How to Run:

1. Save all 4 files in the same folder.
2. Open main.html in your browser.
3. Click on links in the menu (left frame) to load content into the right frame.

Unit- 07

Create a web page for demonstration of CSS by applying V 06* Internal/External/ Inline style.

Objective:

To create a web page that demonstrates the use of:

- Inline CSS
- Internal CSS
- External CSS

Tools Required:

- A simple text editor (Notepad, VS Code, Sublime Text)
- A web browser (Chrome, Firefox, Edge)

Files Required:

You will create **3 files**:

1. index.html – Main HTML file with all 3 CSS types
2. style.css – External CSS file
3. *(Optional)* Screenshot tool or browser for viewing the result

HTML + CSS Implementation

1. **index.html (Demonstrates Inline, Internal, and External CSS)**

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Demo</title>

  <!-- Internal CSS -->
  <style>
    body {
      background-color: #f0f8ff;
      font-family: Arial, sans-serif;
    }

    h1 {
      color: darkblue;
      text-align: center;
    }

    .internal-style {
      color: green;
```

```
            font-size: 18px;
        }
    </style>

    <!-- External CSS Link -->
    <link rel="stylesheet" type="text/css" href="style.css">
</head>

<body>
    <!-- Inline CSS -->
    <h1 style="color: crimson; font-size: 30px;">Inline CSS Example</h1>

    <p class="internal-style">
        This paragraph is styled using <strong>Internal CSS</strong>.
    </p>

    <p class="external-style">
        This paragraph is styled using <strong>External CSS</strong>.
    </p>

    <p style="background-color: yellow; padding: 10px;">
        This paragraph uses <strong>Inline CSS</strong> for background and padding.
    </p>
</body>
</html>
```

2. style.css (External CSS File)

```
/* External CSS */
.external-style {
    color: purple;
    font-weight: bold;
    font-size: 20px;
    text-decoration: underline;
}
```

How to Run:

1. Save both files (index.html and style.css) in the same folder.
2. Open index.html in any browser.
3. Observe the styles applied via inline, internal, and external CSS.

Unit- 08

Install a web server and publish a website on Intranet and Publish It on internet.

Title: Installing a Web Server and Publishing a Website on Intranet and Internet

Objective:

- To install and configure a web server (Apache)
- To publish a website on a local intranet
- To make the website accessible over the internet

Requirements:

- A computer/server with Linux (Ubuntu) or Windows
- Apache web server (or XAMPP on Windows)
- Static IP or Port Forwarding for internet access
- Basic HTML website files
- Internet connection
- Administrator privileges

PART A: Installing Web Server (Apache) on Linux (Ubuntu)

1. **Update system packages**
   sudo apt update && sudo apt upgrade.
2. **Install Apache Web Server**
   sudo apt install apache2 -y
3. **Start and enable Apache**
   sudo systemctl start apache2
   sudo systemctl enable apache2

4. Check Apache is running

Open your browser and go to:

http://localhost

PART B: Publish Website on Intranet

1. **Find your local IP address**
   ip a | grep inet

2. Create your website files

Navigate to Apache's web root:

cd /var/www/html

Replace default index.html:

sudo rm index.html

sudo nano index.html

Add simple HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Intranet Website</title>
</head>
<body>
  <h1>Welcome to the Intranet! </h1>
</body>
</html>
```

Save and exit (Ctrl+O, Enter, then Ctrl+X)

3. Access on other devices in same network

On another device in the same Wi-Fi/LAN, open a browser and visit:

http://192.168.1.x

PART C: Publish Website on the Internet

Option: Use Port Forwarding (for home networks)

1. Access your router settings

Go to: http://192.168.1.1

Login using your router's credentials.

2. Enable Port Forwarding

Forward external port **80** to your server's internal IP **192.168.1.x:80**

3. Find your public IP

curl ifconfig.me

4. Access from the internet

Now go to: http://<your_public_ip>

If your ISP blocks port 80, use a different port like 8080 and access via: http://<your_public_ip>:8080

PART D: (Optional) Installing Web Server on Windows using XAMPP

1. Download XAMPP from https://www.apachefriends.org
2. Install and run XAMPP Control Panel
3. Start **Apache**
4. Place website files in: C:\xampp\htdocs\
5. Access via http://localhost or your local IP (ipconfig in cmd)

Result:

- Apache server installed successfully
- Website hosted and accessible on Intranet
- Website accessible via the Internet using public IP or tunnelling.

Unit- 09

Design a Visiting Card containing at least one graphic and Text information.

Title: Designing a Visiting Card Containing Graphic and Text Information
Objective:
To design a professional visiting (business) card that includes text and at least one graphic (such as a logo, icon, or background image).

Requirements:

- A computer with design software:
- Canva (online, free)
- Adobe Photoshop / Illustrator
- Internet (for downloading icons/logos if needed)
- Printer (optional for printing)

Tools You Can Use:

Tool	Platform	Skill Level
Canva	Web	Beginner
Adobe Photoshop	Windows/Mac	Advanced
Figma	Web	Intermediate

PART A: Planning the Visiting Card Design

Elements to include:

1. Name
2. Designation / Job Title
3. Company Name
4. Logo (Graphic)
5. Phone Number
6. Email Address
7. Website / Social Media
8. Address (Optional)
9. Slogan / Tagline (Optional)

Card Dimensions:
Standard size: 3.5 x 2 inches (U.S. standard)

PART B: Designing the Card (Using Canva - Easy Method)

1. Open Canva

Go to https://www.canva.com
Search for “Business Card” templates.

2. Choose a Template

Pick a template that includes both:

- Graphic (e.g., logo, icon, or image)
- Text areas

3. Insert Text Information

Fill in:

- Name: John Doe
- Title: Graphic Designer
- Company: DesignX Studio
- Phone: +91 98765 43210
- Email: john@designx.com
- Website: www.designx.com

4. Add a Logo or Graphic

- Use the Uploads panel to insert your logo
- Or use Elements > Icons to add a placeholder

5. Customize Fonts & Colors

- Use professional fonts (e.g., Montserrat, Open Sans)
- Stick to 2–3 brand colors

6. Download Your Card

Click Share > Download
Choose PDF for print or PNG/JPG for web

PART C: Save and Print

- **Save your final design as:**
- PDF (for professional printing)
- PNG/JPG (for sharing digitally)
- Use high-quality paper (300 GSM) for printing
- Optional: Print double-sided for more information

Sample Layout (Text + Graphic):

```
+----------------------------------------+
|       [Your Company Logo Here]    |
|                                        |
| John Doe                               |
| Senior Software Developer              |
| TechNova Pvt. Ltd.                     |
| Phone: +91-9876543210                  |
| Email: john@technova.com               |
| Web: www.technova.com                  |
+----------------------------------------+
```

Result:

✓ Visiting Card designed using [Canva/MS Word/Photoshop]
✓ Contains both text and at least one graphic (logo/image)
✓ Exported in printable and shareable format

Unit- 10

You are given a picture of a garden as background. Extract the image of a butterfly from another picture and organize it on the background.

Objective:

To extract a butterfly image from one picture and overlay it onto a garden background using image editing techniques.

Materials/Software Required:

Option A – Manual Editing (GUI Tools):

- Computer with:
  - Adobe Photoshop or GIMP installed
- Two images:
  - Image 1: Butterfly (on any background)
  - Image 2: Garden background

Option B – Programmatic Editing (Python):

- Python 3.x
- Libraries:
  - OpenCV (cv2)
  - NumPy
  - PIL (optional)
- Jupiter Notebook / IDE

PROCEDURE

OPTION A: Using Photoshop or GIMP

Step 1: Open Butterfly Image

- Launch Photoshop or GIMP.
- Open the butterfly image.
- Use **selection tools** (e.g., Quick Selection, Lasso, or Magic Wand) to select the butterfly.

Step 2: Remove Background

- Once selected, press Delete or use **"Layer Mask"** to remove the background.
- You now have a **transparent butterfly image**.

Step 3: Open Garden Background

- Open the garden image.

- Drag and drop the butterfly layer onto the garden background.
- Use **Transform Tool (Ctrl+T)** to resize/reposition the butterfly.

Step 4: Final Touch

- Add shadow, adjust color tones if needed for realism.
- Save the final image (File > Export As > PNG/JPG).

OBSERVATIONS: -

Image	Description
Butterfly Source	Image before background removal
Garden Background	Original garden scene
Final Composition	Butterfly placed realistically in the garden

RESULT: -

Successfully extracted a butterfly from one image and overlaid it onto a garden background using image processing techniques.

Unit- 11

Shape Distortion: Create a square and gradually convert it into a circle.

Objective:

To demonstrate shape morphing by gradually transforming a square into a circle using either GUI tools or programmatically via Python.

Materials/Software Required:

You can use one of the following approaches:

Option A — GUI-Based Tools:

- Adobe Illustrator / Photoshop
- Figma / GIMP (Free alternative)

OPTION A: Using Graphic Design Tools

Step 1: Create the Square

1. Open Illustrator or GIMP.
2. Draw a **perfect square** (e.g., 200×200 px).
3. Convert the square to a **path** or **vector shape**.

Step 2: Create the Circle

1. Draw a **circle** of the same size (same width & height).
2. Convert it into a **path** too.

Step 3: Interpolate (Morphing)

1. Use the **Blend Tool** in Illustrator or the **Tweening feature** in Figma.
2. Blend the square and circle with multiple steps (e.g., 10 steps).
3. You will see intermediate shapes that gradually morph from a square to a circle.

Step 4: Export Results

1. Save each intermediate frame if needed for animation or analysis.
2. Export as PNG/GIF/SVG.

OBSERVATIONS: -

Frame	Description
0	Perfect square
5	Rounded corners begin to appear
10	Nearly circular shape
19	Perfect circle

RESULT: -

Successfully demonstrated shape distortion by transforming a square into a circle through smooth interpolation using graphical methods.

Unit- 12

Spotlight: Create a text on one layer; format the text with suitable size, color and style. With the help of another layer, position a spotlight on the text and move the spotlight from left to right.

Objective:

To create an animated spotlight effect over formatted text by using multiple layers and moving the spotlight from left to right across the text.

Software Required:

- Adobe After Effects (preferred)
  OR
- Any layer-based animation software (e.g., Adobe Animate, Krita, Photoshop with Timeline, etc.)

Theory:

- A **spotlight effect** involves creating a bright, focused light that illuminates part of the scene. In animation or motion graphics, this is often simulated using gradient masks or light layers that move over the object or text.
- By placing the spotlight on a separate layer and animating it across the text layer, we simulate the real-world motion of a moving light source.

Procedure:

Step 1: Create a New Project

1. Open your animation software (e.g., After Effects).
2. Create a new composition:
   - Resolution: 1920x1080
   - Duration: 5 seconds
   - Background color: Black

Step 2: Add and Format Text

1. Create a **new text layer**.
2. Type your desired text, e.g., "Spotlight Effect".
3. Use the **Character panel** to format the text:
   - Font: Bold or Sans Serif (e.g., Montserrat, Arial Black)
   - Size: Large enough to be centered
   - Color: White or light gray (to see the spotlight effect)
4. Center the text on the screen using the Align panel.

Step 3: Create Spotlight Layer

1. Add a **new solid layer** (Layer > New > Solid) – make it white or light yellow.
2. Add a **circular mask** to the solid:
   - Use the Ellipse Tool (double-click to center if needed).

- Feather the mask (F key) to make the edges soft (e.g., 100 px).
- Change the mask mode to **Subtract** if needed to only show the spotlight.

Alternatively: Use a **light layer** (in After Effects) for a realistic 3D light effect.

Step 4: Animate the Spotlight

1. Select the spotlight layer.
2. At the **start of the timeline**, set a **Position keyframe**.
3. Move the spotlight to the **left side** of the text.
4. Move the playhead to the **end of the timeline** (e.g., 5 sec).
5. Move the spotlight to the **right side** of the text.
6. Preview the animation – the spotlight should move smoothly across the text.

Step 5: Refine the Animation (Optional)

- Use **Easy Ease** (F9) for smoother motion.
- Adjust the **feather** or **opacity** of the spotlight for a softer look.
- Use blending modes (e.g., **Add**, **Screen**) if your software supports them.

Saving and Exporting

1. Save your project file.
2. Export/render the animation:
   - Format: MP4 or GIF (depending on need)
   - Frame rate: 24 or 30 fps
   - Resolution: 1080p

Result:

You should see the text staying static while a soft spotlight moves smoothly from the left side to the right, creating a professional-looking highlight effect.

Unit- 13

"Simulation of a Raindrop: In the first layer, draw a raindrop that falls on the ground. Show the splash effect, when it touches the ground on another layer."

Objective

To simulate the motion of a falling raindrop and the resulting splash effect using animation and layering techniques in a graphics environment (e.g., HTML5 Canvas, Processing, Python with Pygame, or any animation software).
This lab demonstrates basic concepts in:

- Frame-by-frame animation
- Layering
- Object movement
- Visual effects

Software Requirements

You can implement this using any of the following:

- HTML5 + CSS + JavaScript (Canvas API)
- Processing / p5.js
- Python with Pygame / Turtle
- Adobe Animate or similar animation tools

Example implementation uses p5.js (JavaScript)

Theory

- A raindrop is a simple shape (usually a circle or teardrop) that moves vertically to simulate falling due to gravity.
- When it hits the ground, it creates a splash effect — a quick, expanding circular ripple or multiple small droplets.
- Layering helps manage different visual elements separately — e.g., the falling raindrop and the splash effect.

Procedure

1. **Create the Background**
    - Set up a canvas with a simple background (sky and ground).
2. **Layer 1: Draw the Raindrop**
    - Use a simple shape (ellipse or teardrop).

- Animate it falling from the top to the bottom of the canvas.

3. **Layer 2: Splash Effect**
   - When the raindrop reaches the ground (y-position), trigger a splash animation:
     - Circular ripple expanding outward.
     - Or several small drops moving away from the point of impact.
4. **Timing and Transition**
   - Ensure the splash only appears when the raindrop reaches the ground.
   - Optionally, reset the raindrop after splash to loop the animation.

Sample Code (p5.js)

```
let dropY = 0;
let dropX;
let splash = false;
let splashRadius = 0;

function setup() {
  createCanvas(400, 400);
  dropX = width / 2;
}

function draw() {
  background(200, 220, 255); // Light blue background
  fill(50, 100, 200);

  // Ground
  fill(100, 200, 100);
  rect(0, height - 20, width, 20);

  // Layer 1: Raindrop falling
  if (!splash) {
    fill(0, 100, 255);
    ellipse(dropX, dropY, 10, 15);
```

```
    dropY += 5;

    // When raindrop hits the ground
    if (dropY >= height - 25) {
      splash = true;
      splashRadius = 0;
    }
  }

  // Layer 2: Splash effect
  if (splash) {
    noFill();
    stroke(0, 100, 255);
    strokeWeight(2);
    ellipse(dropX, height - 25, splashRadius);
    splashRadius += 5;

    if (splashRadius > 50) {
      // Reset
      splash = false;
      dropY = 0;
    }
  }
}
```

Observations

Step	Description
1	Raindrop appears at the top and falls vertically.
2	Splash appears upon contact with the ground.
3	Splash expands then fades.
4	Raindrop resets to start.

Result

- Successfully simulated the fall of a raindrop and its splash using two separate layers.
- Understood the concept of motion and transition between animation states.

Conclusion

This experiment demonstrates the use of **basic animation principles**, layering, and timing to create a simple but visually effective **rainfall simulation**. It introduces the foundational ideas for more complex animations and physics simulations.

***