LAB ASSIGNMENT -6

NAME : ADITYA NAIR
REG NO: 23BRS1261

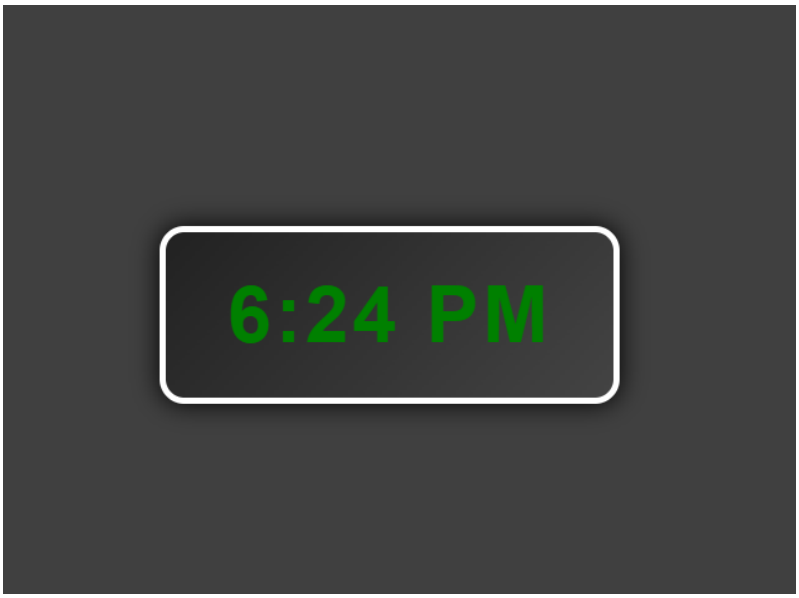
TASK 1 : DIGITAL CLOCK

CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Digital Clock</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color:#404040;
            color: white;
            font-family: 'Arial', sans-serif;
        }
        #clock {
            padding: 30px 50px;
            border: 5px solid white;
            border-radius: 20px;
```

```
        background: linear-gradient(135deg, #222, #444);
        box-shadow: 0 0 20px rgba(0, 0, 0, 1.0);
        font-size: 4rem;
        font-weight: bold;
        text-align: center;
        letter-spacing: 3px;
        color:green;
      }
    </style>
</head>
<body>
    <div id="clock"></div>

    <script>
      function updateClock() {
        const now = new Date();
        let hours = now.getHours();
        let minutes = now.getMinutes().toString().padStart(2, '0');
        let ampm = hours >= 12 ? 'PM' : 'AM';
        hours = hours % 12 || 12;
        document.getElementById('clock').innerText = `${hours}:${minutes} ${ampm}`;
      }
      setInterval(updateClock, 1000);
      updateClock(); // Initialize clock immediately
    </script>
</body>
</html>
```

OUTPUT :



TASK 2 : ANALOG CLOCK

CODE :

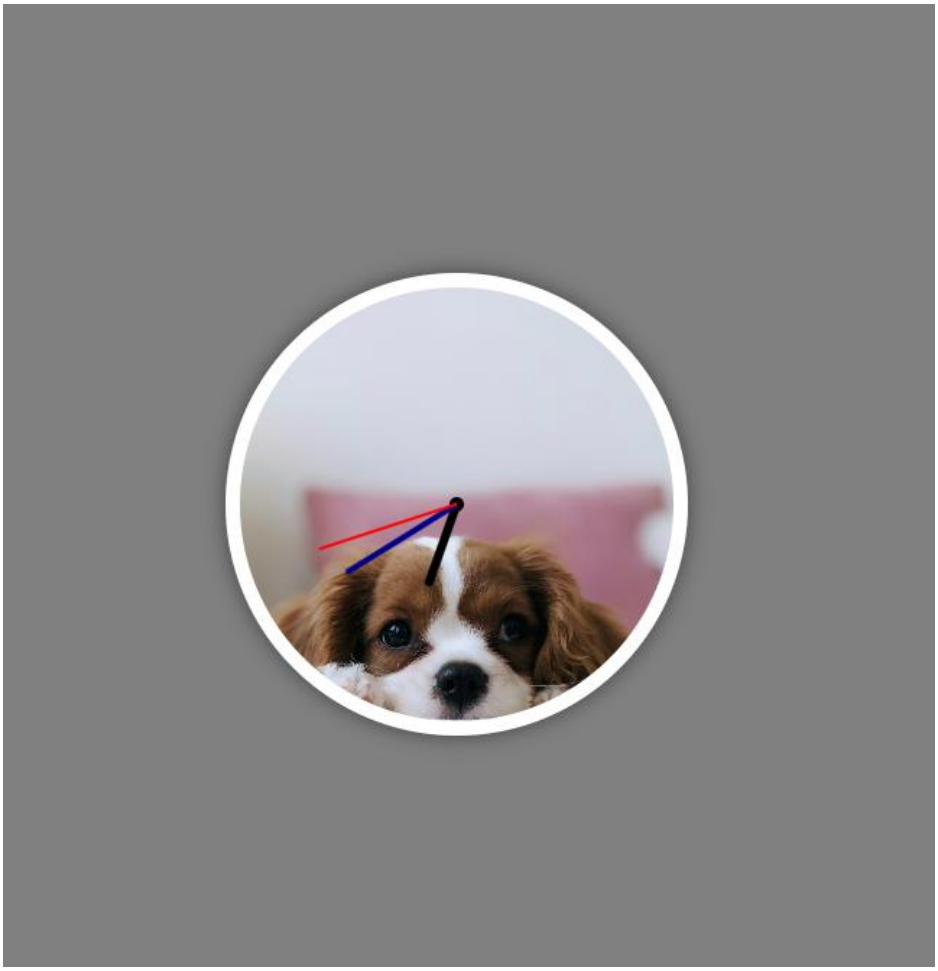
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Analog Clock</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: grey;
        }
        .clock {
            width: 300px;
            height: 300px;
            border-radius: 50%;
```

```
        background: url("https://images.unsplash.com/photo-1560807707-8cc77767d783") no-repeat center/cover;
        border: 10px solid white;
        position: relative;
        box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);
        display: flex;
        justify-content: center;
        align-items: center;
    }
    .hand {
        position: absolute;
        bottom: 50%;
        left: 50%;
        transform-origin: bottom center;
        transform: translateX(-50%);
        border-radius: 5px;
    }
    .hour {
        width: 6px;
        height: 60px;
        background-color: black;
    }
    .minute {
        width: 4px;
        height: 90px;
        background-color: darkblue;
    }
    .second {
        width: 2px;
        height: 100px;
        background-color: red;
    }
    .center-dot {
```

```
        width: 10px;
        height: 10px;
        background: black;
        border-radius: 50%;
        position: absolute;
      }
    </style>
</head>
<body>
    <div class="clock">
      <div class="center-dot"></div>
      <div class="hand hour" id="hour"></div>
      <div class="hand minute" id="minute"></div>
      <div class="hand second" id="second"></div>
    </div>

    <script>
      function updateClock() {
        const now = new Date();
        const hours = now.getHours() % 12;
        const minutes = now.getMinutes();
        const seconds = now.getSeconds();

        const hourDeg = (hours * 30) + (minutes * 0.5);
        const minuteDeg = (minutes * 6) + (seconds * 0.1);
        const secondDeg = seconds * 6;

        document.getElementById('hour').style.transform = `translateX(-50%) rotate(${hourDeg}deg)`;
        document.getElementById('minute').style.transform = `translateX(-50%) rotate(${minuteDeg}deg)`;
        document.getElementById('second').style.transform = `translateX(-50%) rotate(${secondDeg}deg)`;
      }
      setInterval(updateClock, 1000);
```

```
        updateClock();
    </script>
</body>
</html>
```

OUTPUT :



TASK 3 : Minion Eye

CODE :

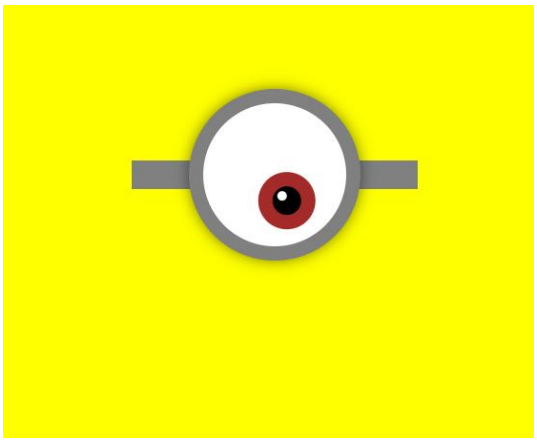
```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Minion Eye Follow Mouse</title>
```

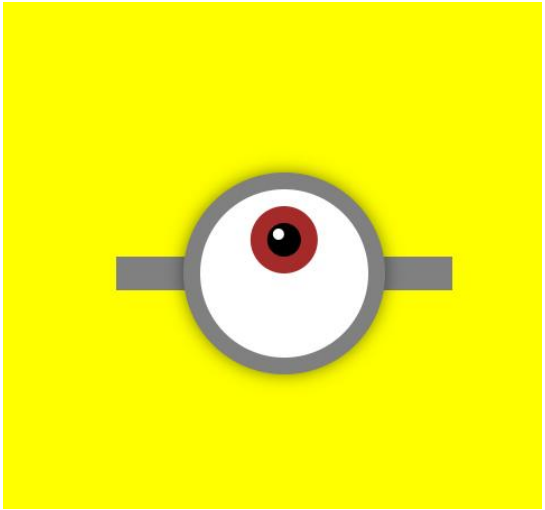
```
<style>
    body {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
        background-color: yellow;
        overflow: hidden;}
    .goggles-strap {
        width: 300px;
        height: 30px;
        background: gray;
        position: absolute;
        z-index: -1;}
    .eye {
        width: 150px;
        height: 150px;
        background: white;
        border-radius: 50%;
        border: 15px solid gray;
        position: relative;
        display: flex;
        justify-content: center;
        align-items: center;
        box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);}
    .iris {
        width: 60px;
        height: 60px;
        background: brown;
        border-radius: 50%;
        display: flex;
        justify-content: center;
```

```
            align-items: center;
            position: absolute;
            transition: all 0.05s ease-out;
        }
        .pupil {
            width: 30px;
            height: 30px;
            background: black;
            border-radius: 50%;
            position: relative}
        .glare {
            width: 10px;
            height: 10px;
            background: white;
            border-radius: 50%;
            position: absolute;
            top: 5px;
            left: 5px;
        }
    </style>
</head>
<body>
    <div class="goggles-strap"></div>
    <div class="eye">
        <div class="iris" id="iris">
            <div class="pupil">
                <div class="glare"></div>
            </div> </div></div>
    <script>
        document.addEventListener("mousemove", (event) => {
            const eye = document.querySelector(".eye");
            const iris = document.querySelector("#iris");
```

```
        const eyeRect = eye.getBoundingClientRect();
        const eyeCenterX = eyeRect.left + eyeRect.width / 2;
        const eyeCenterY = eyeRect.top + eyeRect.height / 2;
        const deltaX = event.clientX - eyeCenterX;
        const deltaY = event.clientY - eyeCenterY;
        const distance = Math.sqrt(deltaX ** 2 + deltaY ** 2);
        const maxDistance = 30; // Limit iris movement
        const angle = Math.atan2(deltaY, deltaX);
        const irisX = Math.cos(angle) * Math.min(distance, maxDistance);
        const irisY = Math.sin(angle) * Math.min(distance, maxDistance);
        iris.style.transform = `translate(${irisX}px, ${irisY}px)`;
      });
  </script>
</body>
</html>
```

OUTPUT :





TASK 4 : Image Slider

Code :

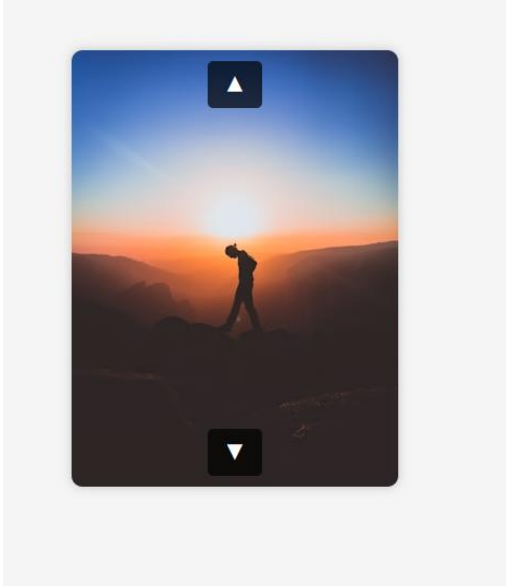
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vertical Image Slider</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f5f5f5;
            margin: 0;
        }

        .slider-container {
            width: 300px;
            height: 400px;
            overflow: hidden;
            position: relative;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
            background: white;
        }

        .slider {
            display: flex;
            flex-direction: column;
```

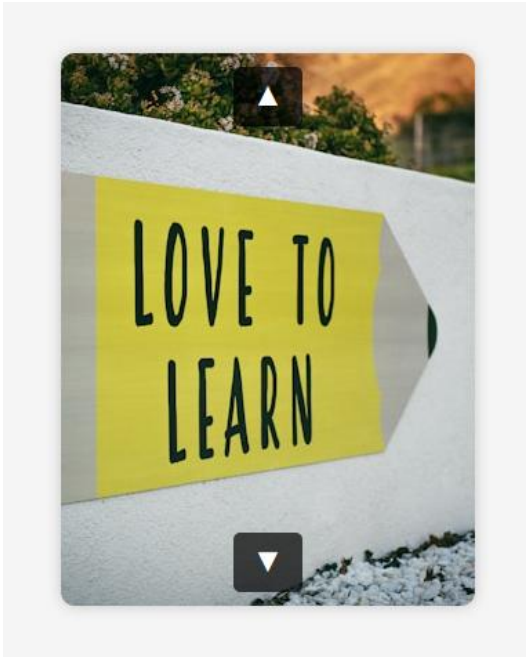
```
    transition: transform 0.5s ease-in-out;
}

.slide {
    width: 100%;
    height: 400px;
}

.slide img {
    width: 100%;
    height: 100%;
    object-fit: cover;
    display: block;
}

/* Navigation Buttons */
.button {
    position: absolute;
    left: 50%;
    transform: translateX(-50%);
    background: rgba(0, 0, 0, 0.7);
    color: white;
    border: none;
    padding: 10px 15px;
    font-size: 20px;
    cursor: pointer;
    border-radius: 5px;
    width: 50px;
    text-align: center;
    transition: background 0.3s;
}
```

```
      .button:hover {
         background: rgba(0, 0, 0, 1);
      }

      .up {
         top: 10px;
      }

      .down {
         bottom: 10px;
      }
   </style>
</head>
<body>
   <div class="slider-container">
      <div class="slider" id="slider">
         <div class="slide"><img src="https://images.unsplash.com/photo-1506748686214-e9df14d4d9d0?w=300&h=400&fit=crop" alt="Nature"></div>
         <div class="slide"><img src="https://images.unsplash.com/photo-1506748686214-e9df14d4d9d0?w=300&h=400&fit=crop" alt="City"></div>
         <div class="slide"><img src="https://images.unsplash.com/photo-1490135900376-2e96e28c35d1?w=300&h=400&fit=crop" alt="Technology"></div>
      </div>
      <button class="button up" onclick="moveSlide(-1)">▲</button>
      <button class="button down" onclick="moveSlide(1)">▼</button>
   </div>

   <script>
      let index = 0;
      const slides = document.querySelectorAll(".slide");
      const slider = document.getElementById("slider");

      function moveSlide(step) {
```

```
        index += step;
        if (index < 0) index = slides.length - 1;
        if (index >= slides.length) index = 0;
        slider.style.transform = `translateY(-${index * 400}px)`;
      }
  </script>
</body>
</html>
```

OUTPUT :





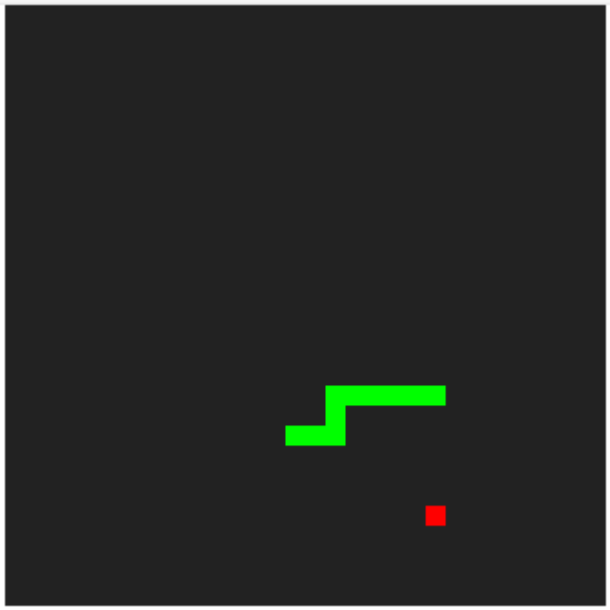
TASK 5 : Snake Game

CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Minimal Snake</title>
  <style>
    * { margin: 0; padding: 0; }
    canvas { background: #222; display: block; margin: auto; }
  </style>
</head>
<body>
  <canvas id="game" width="300" height="300"></canvas>
  <script>
    const canvas = document.getElementById("game"), ctx = canvas.getContext("2d");
    const grid = 10, snake = [{ x: 150, y: 150 }];
    let dir = { x: grid, y: 0 }, food = { x: 50, y: 50 };

    function update() {
      let head = { x: snake[0].x + dir.x, y: snake[0].y + dir.y };
      snake.unshift(head);
      if (head.x === food.x && head.y === food.y) food = { x: grid * Math.floor(Math.random() * 30), y: grid
* Math.floor(Math.random() * 30) };
      else snake.pop();
      if (head.x < 0 || head.y < 0 || head.x >= canvas.width || head.y >= canvas.height ||
snake.slice(1).some(p => p.x === head.x && p.y === head.y)) location.reload();
    }

    function draw() {
```

```
    ctx.fillStyle = "#222"; ctx.fillRect(0, 0, canvas.width, canvas.height);
    ctx.fillStyle = "lime"; snake.forEach(p => ctx.fillRect(p.x, p.y, grid, grid));
    ctx.fillStyle = "red"; ctx.fillRect(food.x, food.y, grid, grid);
  }

  function loop() { update(); draw(); setTimeout(loop, 100); }
  document.addEventListener("keydown", e => {
    if (e.key === "ArrowUp" && dir.y === 0) dir = { x: 0, y: -grid };
    if (e.key === "ArrowDown" && dir.y === 0) dir = { x: 0, y: grid };
    if (e.key === "ArrowLeft" && dir.x === 0) dir = { x: -grid, y: 0 };
    if (e.key === "ArrowRight" && dir.x === 0) dir = { x: grid, y: 0 };
  });

  loop();
 </script>
</body>
</html>
```

OUTPUT :



TASK 6 : Accessing video

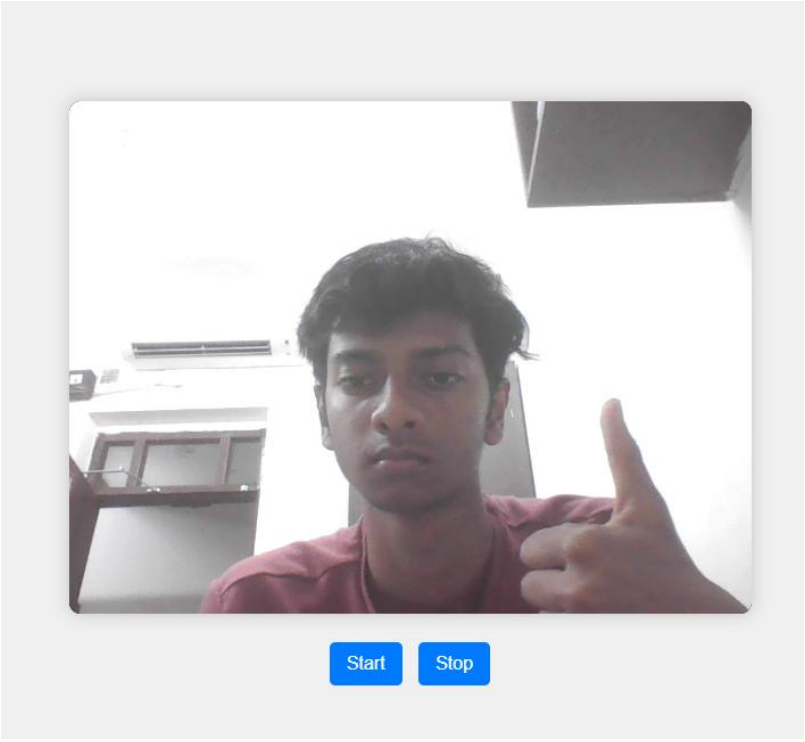
CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Webcam Viewer</title>
    <style>
        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            background: #f0f0f0;
            font-family: Arial, sans-serif;
        }
        video {
            width: 80%;
            max-width: 600px;
            border-radius: 10px;
            box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);
            background: black;
        }
        .buttons {
            margin-top: 20px;
        }
        button {
            padding: 10px 15px;
            font-size: 16px;
```

```
            margin: 5px;
            border: none;
            background: #007bff;
            color: white;
            border-radius: 5px;
            cursor: pointer;
            transition: 0.3s;
        }
        button:hover {
            background: #0056b3;
        }
    </style>
</head>
<body>
    <video id="webcam" autoplay playsinline></video>
    <div class="buttons">
        <button onclick="startWebcam()">Start</button>
        <button onclick="stopWebcam()">Stop</button>
    </div>

    <script>
        const video = document.getElementById("webcam");
        let stream = null;

        function startWebcam() {
            navigator.mediaDevices.getUserMedia({ video: true })
                .then(s => {
                    stream = s;
                    video.srcObject = stream;
                })
                .catch(err => console.error("Error accessing webcam:", err));
        }
```

```
    function stopWebcam() {
      if (stream) {
        stream.getTracks().forEach(track => track.stop());
        video.srcObject = null;
      }
    }
  </script>
</body>
</html>
```

OUTPUT :



TASK 7 : Mobile Flashlight

Code :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flashlight Control</title>
    <style>
        body {
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background: #f0f0f0;
            font-family: Arial, sans-serif;
        }
        button {
            padding: 15px 20px;
            font-size: 18px;
            border: none;
            background: #007bff;
            color: white;
            border-radius: 5px;
            cursor: pointer;
            transition: 0.3s;
        }
        button:hover {
            background: #0056b3;
        }
```

```
  </style>
</head>
<body>
  <button onclick="toggleFlashlight()">Toggle Flashlight</button>

  <script>
    let flashlightOn = false;
    let stream;

    async function toggleFlashlight() {
      if (!flashlightOn) {
        try {
          stream = await navigator.mediaDevices.getUserMedia({ video: { facingMode: "environment", torch: true } });
          const track = stream.getVideoTracks()[0];
          const capabilities = track.getCapabilities();
          if (capabilities.torch) {
            await track.applyConstraints({ advanced: [{ torch: true }] });
            flashlightOn = true;
          }
        } catch (err) {
          alert("Flashlight not supported on this device.");
          console.error(err);
        }
      } else {
        if (stream) {
          stream.getTracks().forEach(track => track.stop());
        }
        flashlightOn = false;
      }
    }
  </script>
</body>
```

</html>

OUTPUT :

TASK 8 : Image Flash , Spotlight

CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Spotlight Effect</title>
  <style>
    body {
      margin: 0;
      overflow: hidden;
    }
    .container {
      position: relative;
      width: 100vw;
      height: 100vh;
      background: url('https://images.unsplash.com/photo-1506748686214-e9df14d4d9d0') no-repeat center/cover;
```

```
    }
    .overlay {
      position: absolute;
      width: 100%;
      height: 100%;
      background: radial-gradient(circle 120px at var(--x, 50%) var(--y, 50%), rgba(0,0,0,0) 10%, rgba(0,0,0,0.85) 80%);
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="overlay"></div>
  </div>

  <script>
    const overlay = document.querySelector('.overlay');

    document.addEventListener('mousemove', (e) => {
      overlay.style.setProperty('--x', `${e.clientX}px`);
      overlay.style.setProperty('--y', `${e.clientY}px`);
    });
  </script>
</body>
</html>
```

OUTPUT :