BCSE203E – Web Programming Lab

Winter Semester 2024 – 2025

ASSESSMENT – 6

NAME: Varshith Pilli

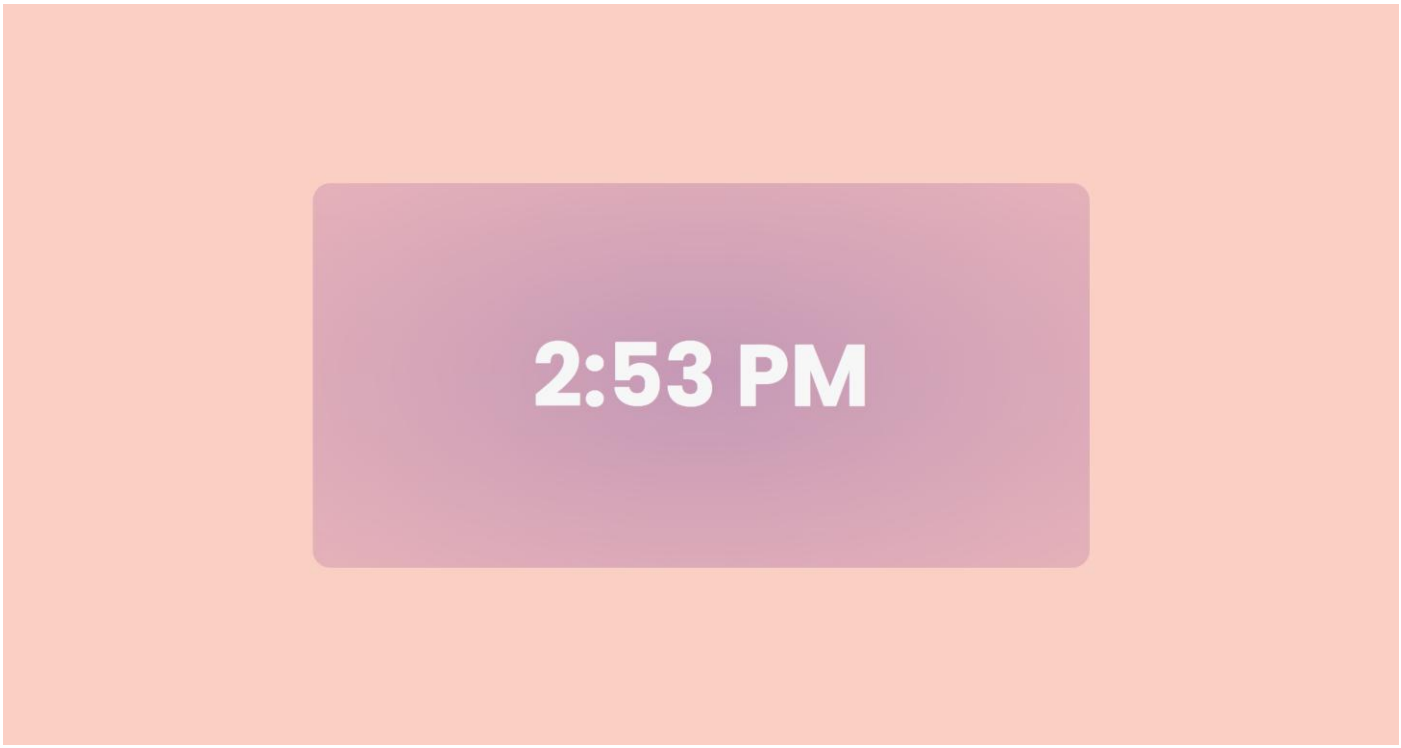
REGISTER NO. :23BPS1136

QUESTION 1

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 1</title>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap');

    * {
      margin: 0;
      padding: 0;
    }

    body {
      height: 100vh;
      width: 100vw;
      display: flex;
      justify-content: center;
      align-items: center;
      background-color: #FAD0C4;
    }

    .container {
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
      height: 400px;
      background: radial-gradient(#C599B6, #E6B2BA);
      border-radius: 20px;
      font-family: "Poppins";
      font-weight: 400;
      font-style: normal;
      font-size: 100px;
      padding: 20px;
      width: 850px;
      cursor: pointer;
      color: #F7F7F7;
      transition: opacity 2s ease-in-out;
    }
  </style>
</head>

<body>
  <div class="container" id="textContainer">
    <div id="time">00:00 AM</div>
    <div id="date" style="opacity: 0; position: absolute;">00, 00 000</div>
  </div>
```

```
  <script>
    const months = [
      "January", "February", "March", "April", "May", "June",
      "July", "August", "September", "October", "November", "December"
    ];
    const days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

    function updateTimeAndDate() {
      const now = new Date();
      let hours = now.getHours();
      let minutes = now.getMinutes();
      let day = days[now.getDay()];
      let date = now.getDate();
      let month = months[now.getMonth()];
      let ampm = hours >= 12 ? "PM" : "AM";

      hours = hours % 12 || 12;
      minutes = minutes < 10 ? "0" + minutes : minutes;

      document.getElementById('time').innerText = hours + ":" + minutes + " " + ampm;
      document.getElementById('date').innerText = day + ", " + date + " " + month;
    }

    updateTimeAndDate();

    const textContainer = document.getElementById("textContainer");
    const clock = document.getElementById("time");
    const date = document.getElementById("date");

    textContainer.addEventListener("mouseenter", () => {
      clock.style.opacity = "0";
      date.style.opacity = "1";
    });

    textContainer.addEventListener("mouseleave", () => {
      clock.style.opacity = "1";
      date.style.opacity = "0";
    });

    setInterval(updateTimeAndDate, 1000);
  </script>

</body>

</html>
```

OUTPUT



On hover:

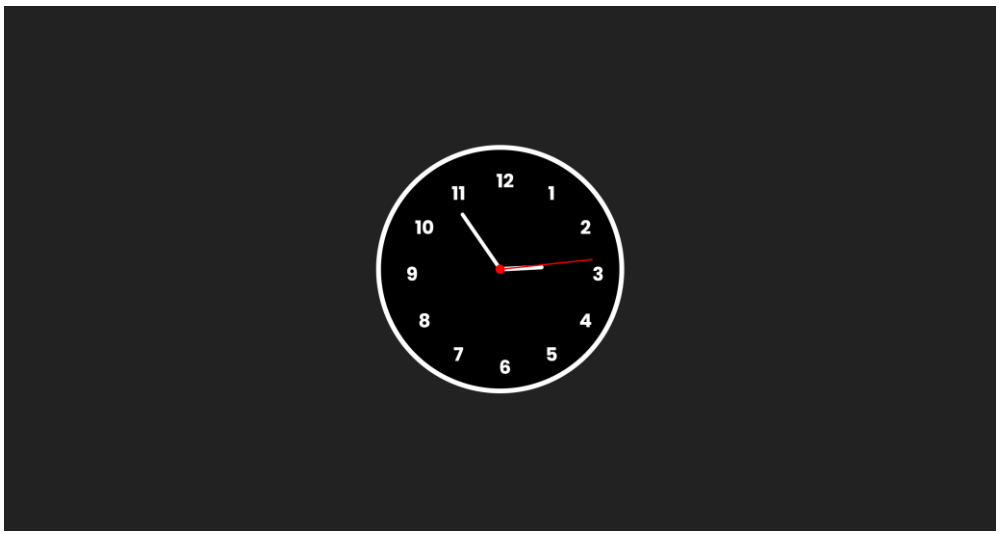
QUESTION 2

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 2</title>
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap');
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      height: 100vh;
      width: 100vw;
      display: flex;
      justify-content: center;
      align-items: center;
      background: #222;
    }

    .clock {
      width: 400px;
      height: 400px;
      border: 8px solid white;
      border-radius: 50%;
      position: relative;
      background: black;
      display: flex;
      justify-content: center;
      align-items: center;
    }

    .hand {
      position: absolute;
      bottom: 50%;
      left: 50%;
      transform-origin: 50% 100%;
      transform: translateX(-50%) rotate(0deg);
      transition: transform 0.2s linear;
      border-radius: 5px;
    }

    .hour {
      width: 7px;
      height: 70px;
      background: white;
    }
```

```
    .minute {
      width: 6px;
      height: 110px;
      background: white;
    }

    .second {
      width: 2px;
      height: 150px;
      background: red;
    }

    .center-dot {
      width: 15px;
      height: 15px;
      background: red;
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
      border-radius: 50%;
    }

    .number {
      font-family: "Poppins";
      position: absolute;
      color: white;
      font-size: 30px;
      font-weight: bold;
      text-align: center;
      width: 30px;
      height: 30px;
      display: flex;
      justify-content: center;
      align-items: center;
    }
  </style>
</head>
<body>

  <div class="clock" id="clock">
    <div class="hand hour" id="hour"></div>
    <div class="hand minute" id="minute"></div>
    <div class="hand second" id="second"></div>
    <div class="center-dot"></div>
  </div>

  <script>
    function createNumbers() {
      const clock = document.getElementById("clock");
      const clockSize = 400;
      const centerX = clockSize / 2;
      const centerY = clockSize / 2;
      const radius = 150; // Distance from center
```

```
    for (let i = 1; i <= 12; i++) {
      let num = document.createElement("div");
      num.classList.add("number");
      num.innerText = i;
      clock.appendChild(num); // Append first to get dimensions

      let angle = (i - 3) * (Math.PI / 6); // Offset by -90 degrees
      let x = centerX + radius * Math.cos(angle) - (num.clientWidth / 2);
      let y = centerY + radius * Math.sin(angle) - (num.clientHeight / 2);

      num.style.left = `${x}px`;
      num.style.top = `${y}px`;
    }
  }

  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)`;
  }

  createNumbers();
  setInterval(updateClock, 1000);
  updateClock();
</script>

</body>
</html>
```

OUTPUT

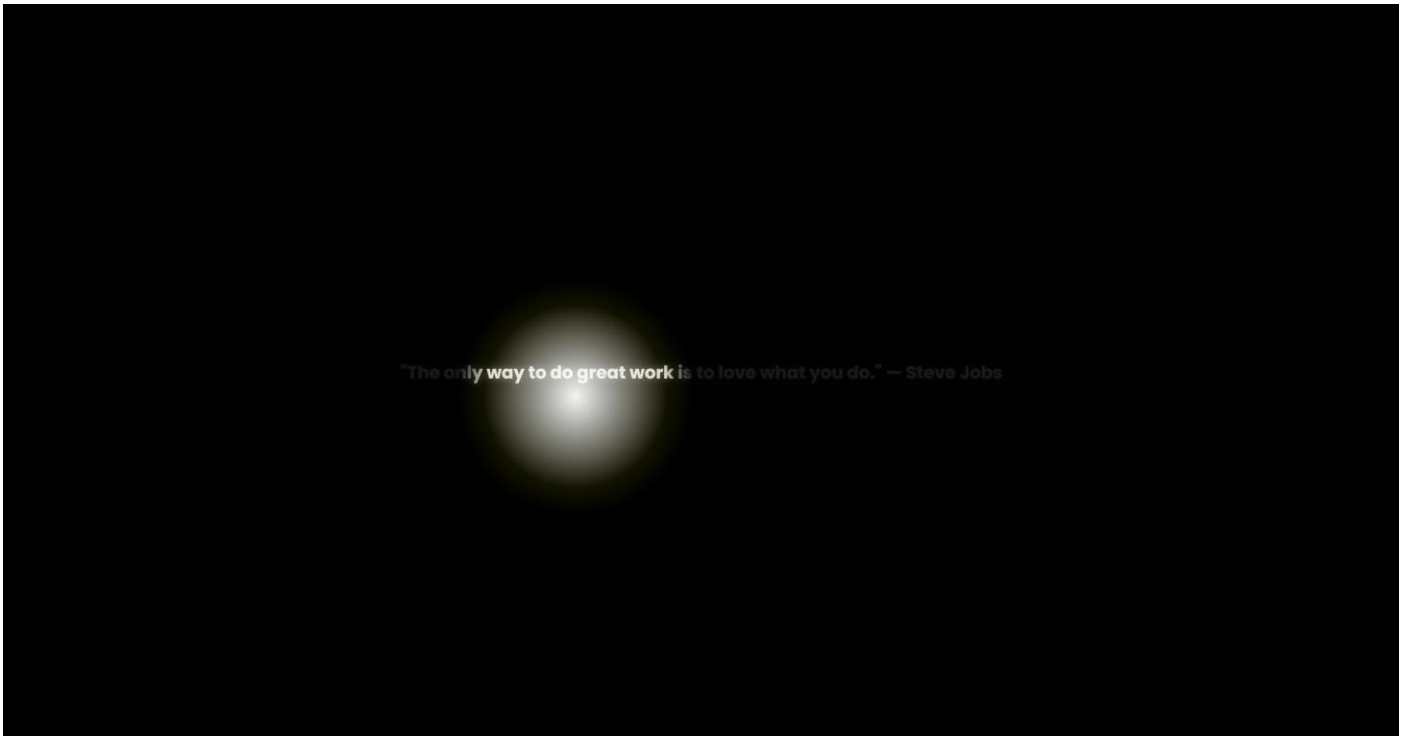


QUESTION 3

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Question 3</title>

    <style>
      @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap');
      *{
        margin: 0;
        padding: 0;
      }

      body {
        height: 100vh;
        width: 100vw;
        display: flex;
        align-items: center;
        justify-content: center;
        background-color: black;
        overflow: hidden;
        color: white;
        font-family: 'Poppins';
        position: relative;
        text-align: center;
        cursor: none;
      }

      .container {
        position: relative;
        font-size: 20px;
        text-shadow: 0 0 10px rgba(255, 255, 255, 0.5);
      }

      .light {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background: radial-gradient(circle 150px at var(--x, 50%) var(--y, 50%),
              rgba(255, 255, 255, 0.95) 1%,
              rgba(95, 93, 3, 0.2) 70%,
              rgba(0, 0, 0, 0.9) 90%);
        pointer-events: none;
        transition: background 0.1s ease-out;
      }

    </style>
  </head>
```

```
  <body>
    <div class="container">"The only way to do great work is to love what you do." — Steve Jobs</div>
    <div class="light"></div>

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

      document.addEventListener('mousemove', (e) => {
        light.style.setProperty('--x', `${e.clientX}px`);
        light.style.setProperty('--y', `${e.clientY}px`);

        torch.style.left = `${e.clientX}px`;
        torch.style.top = `${e.clientY}px`;
      });
    </script>
  </body>
</html>
```

OUTPUT



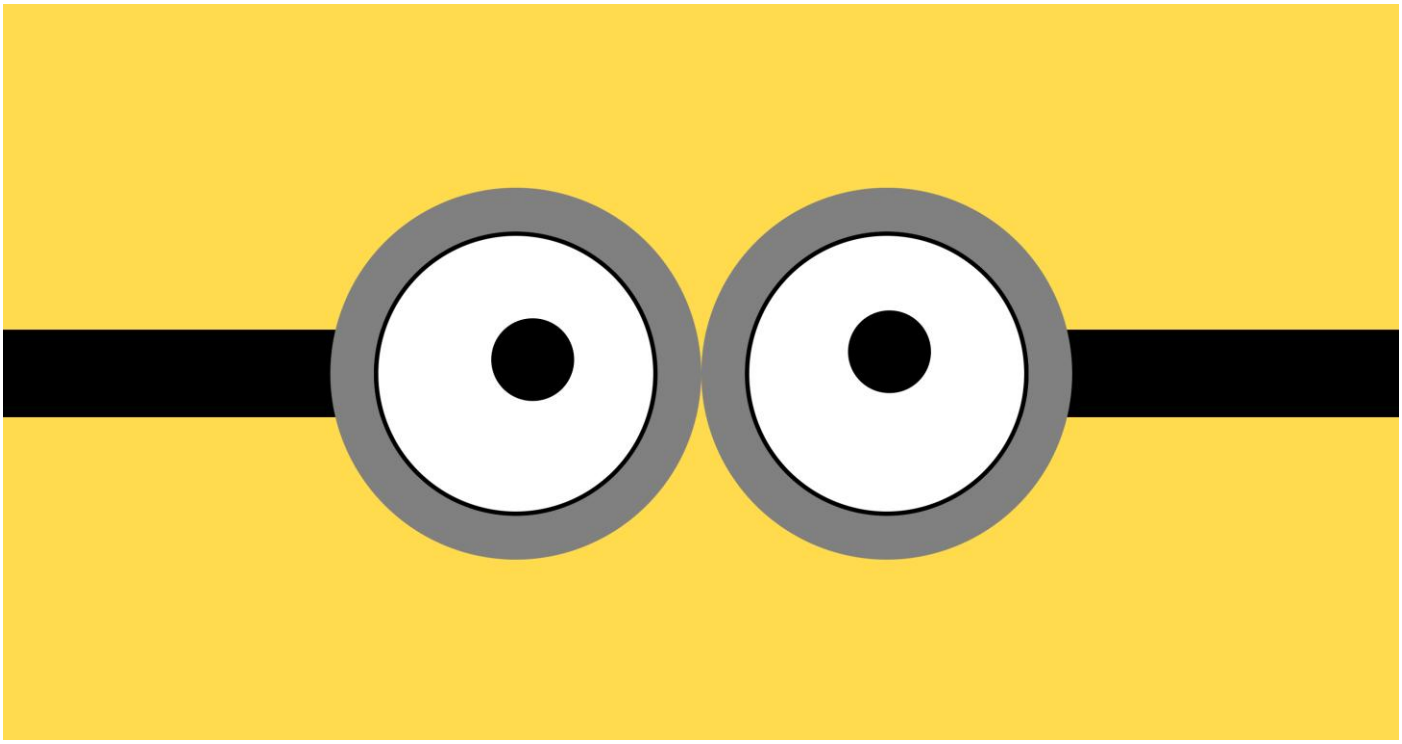
QUESTION 4

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 4</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      height: 100vh;
      width: 100vw;
      display: flex;
      justify-content: center;
      align-items: center;
      background: #ffdb4d;
    }

    .goggles {
      display: flex;
      justify-content: space-between;
      position: relative;
      padding: 20px;
      border-radius: 50px;
    }

    .goggle-1{
      width: 50vh;
      height: 50vh;
      display: flex;
      justify-content: center;
      align-items: center;
      border-radius: 50%;
      border: 50px grey solid;
    }

    .goggle-2{
      width: 50vh;
      height: 50vh;
      display: flex;
      justify-content: center;
      align-items: center;
      border-radius: 50%;
      border: 50px grey solid;
    }
```

```
    .eye {
      width: 100%;
      height: 100%;
      background: white;
      border-radius: 50%;
      position: relative;
      display: flex;
      justify-content: center;
      align-items: center;
      border: 5px solid black;
    }

    .pupil {
      width: 30%;
      height: 30%;
      background: black;
      border-radius: 50%;
      position: absolute;
    }

    .band{
      position: fixed;
      height: 100px;
      width: 100vw;
      background: linear-gradient(90deg, black,black, black, #ffdb4d , black, black, black);
      background-color: black;
      z-index: -1;
    }
  </style>
</head>
<body>

  <div class="goggles">
    <div class="goggle-1">
      <div class="eye">
        <div class="pupil" id="pupil1"></div>
      </div>
    </div>
    <div class="goggle-2">
      <div class="eye">
        <div class="pupil" id="pupil2"></div>
      </div>
    </div>
  </div>
  <div class="band">

  </div>

  <script>
    document.addEventListener("mousemove", (event) => {
      const eyes = document.querySelectorAll(".eye");
      eyes.forEach((eye) => {
        const pupil = eye.querySelector(".pupil");
        const rect = eye.getBoundingClientRect();
        const eyeX = rect.left + rect.width / 2;
        const eyeY = rect.top + rect.height / 2;
```

```
            const deltaX = event.clientX - eyeX;
            const deltaY = event.clientY - eyeY;
            const angle = Math.atan2(deltaY, deltaX);
            const distance = Math.min(25, Math.hypot(deltaX, deltaY) / 10);

            pupil.style.transform = `translate(${distance * Math.cos(angle)}px, ${distance * Math.sin(angle)}px)`;
        });
    });
  </script>

</body>
</html>
```

OUTPUT



QUESTION 5

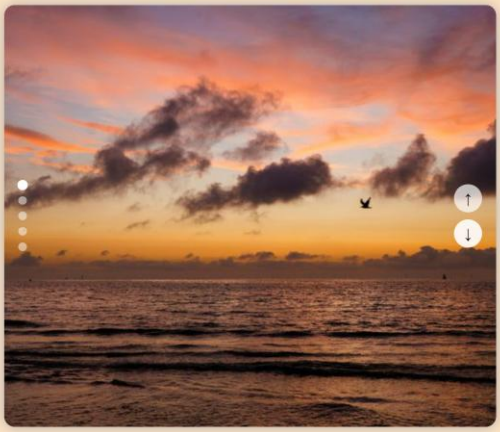
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Question 5</title>

    <style>
      @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap');
      * {
        margin: 0;
        padding: 0;
      }

      body {
        height: 100vh;
        width: 100vw;
        display: flex;
        justify-content: center;
        align-items: center;
        font-family: "Poppins";
        background-color: bisque;
      }

      .slider-container {
        position: relative;
        height: 600px;
        width: 700px;
        overflow: hidden;
        border-radius: 15px;
        box-shadow: 0px 0px 20px darkgray;
      }

      .slider {
        position: absolute;
        height: 100%;
        width: 100%;
        transition: transform 0.5s ease-in-out;
      }

      .slide {
        position: absolute;
        height: 100%;
        width: 100%;
        object-fit: cover;
        opacity: 0;
        transition: opacity 0.5s ease-in-out;
      }

      .slide.active {
        opacity: 1;
      }
```

```
      .controls {
        position: absolute;
        top: 50%;
        right: 20px;
        display: flex;
        flex-direction: column;
        gap: 10px;
        transform: translateY(-50%);
        z-index: 10;
      }

      .control-btn {
        height: 40px;
        width: 40px;
        display: flex;
        justify-content: center;
        align-items: center;
        font-size: 20px;
        background-color: rgba(255, 255, 255, 0.7);
        border: none;
        border-radius: 50%;
        cursor: pointer;
        transition: background-color 0.3s;
      }

      .control-btn:hover {
        background-color: rgba(255, 255, 255, 0.9);
      }

      .indicators {
        position: absolute;
        top: 50%;
        left: 20px;
        display: flex;
        flex-direction: column;
        gap: 10px;
        transform: translateY(-50%);
        z-index: 10;
      }

      .indicator {
        height: 12px;
        width: 12px;
        border-radius: 50%;
        background-color: rgba(255, 255, 255, 0.5);
        cursor: pointer;
        transition: background-color 0.3s;
      }

      .indicator.active {
        background-color: rgba(255, 255, 255, 1);
        transform: scale(1.2);
      }
    </style>
  </head>
```

```
<body>
  <div class="slider-container">
    <div class="slider">
      <img src="./assignment_files/slide1.jpg" alt="Slide 1" class="slide active">
      <img src="./assignment_files/slide2.jpg" alt="Slide 2" class="slide">
      <img src="./assignment_files/slide3.jpg" alt="Slide 3" class="slide">
      <img src="./assignment_files/slide4.jpg" alt="Slide 4" class="slide">
      <img src="./assignment_files/slide5.jpg" alt="Slide 5" class="slide">
    </div>

    <div class="controls">
      <button class="control-btn up-btn">↑</button>
      <button class="control-btn down-btn">↓</button>
    </div>

    <div class="indicators">
      <div class="indicator active" data-index="0"></div>
      <div class="indicator" data-index="1"></div>
      <div class="indicator" data-index="2"></div>
      <div class="indicator" data-index="3"></div>
      <div class="indicator" data-index="4"></div>
    </div>
  </div>

  <script>
    document.addEventListener('DOMContentLoaded', function() {
      const slides = document.querySelectorAll('.slide');
      const upBtn = document.querySelector('.up-btn');
      const downBtn = document.querySelector('.down-btn');
      const indicators = document.querySelectorAll('.indicator');

      let currentSlide = 0;
      const totalSlides = slides.length;


      upBtn.addEventListener('click', prevSlide);
      function prevSlide() {
        showSlide(currentSlide - 1);
      }

      downBtn.addEventListener('click', nextSlide);
      function nextSlide() {
        showSlide(currentSlide + 1);
      }

      indicators.forEach(indicator => {
        indicator.addEventListener('click', function() {
          const slideIndex = parseInt(this.getAttribute('data-index'));
          showSlide(slideIndex);
        });
      });

      function showSlide(index) {
        if (index < 0) index = totalSlides - 1;
        if (index >= totalSlides) index = 0;
```

```
            slides.forEach(slide => {
              slide.classList.remove('active');
            });

            indicators.forEach(indicator => {
              indicator.classList.remove('active');
            });

            slides[index].classList.add('active');
            indicators[index].classList.add('active');

            currentSlide = index;
          }
        });
        </script>
    </body>
</html>
```

OUTPUT



QUESTION 6

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 6</title>

  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap');

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    h1 {
      padding: 20px;
      color: azure;
    }

    #score {
      padding: 30px;
      color: azure;
    }

    body {
      height: 100vh;
      width: 100vw;
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
      background: #222;
      font-family: "Poppins";
    }

    canvas {
      border: 2px solid rgb(0, 255, 0);
      box-shadow: 0 0 15px rgb(79, 255, 79);
      background: black;
    }
  </style>
</head>

<body>
  <h1>SNAKE GAME</h1>
  <canvas width="400" height="400" id="game"></canvas>
  <div id="score">Score: 0</div>
```

```
<script>
  var canvas = document.getElementById('game');
  var context = canvas.getContext('2d');

  var grid = 16;
  var count = 0;
  var score = 0;

  var snake = {
    x: 160,
    y: 160,
    dx: grid,
    dy: 0,
    cells: [],
    maxCells: 4
  };

  var apple = {
    x: getRandomInt(0, 25) * grid,
    y: getRandomInt(0, 25) * grid
  };

  function getRandomInt(min, max) {
    return Math.floor(Math.random() * (max - min)) + min;
  }

  function placeApple() {
    apple.x = getRandomInt(0, 25) * grid;
    apple.y = getRandomInt(0, 25) * grid;
  }

  function loop() {
    requestAnimationFrame(loop);
    if (++count < 4) return;
    count = 0;
    context.clearRect(0, 0, canvas.width, canvas.height);

    snake.x += snake.dx;
    snake.y += snake.dy;

    if (snake.x < 0) snake.x = canvas.width - grid;
    else if (snake.x >= canvas.width) snake.x = 0;
    if (snake.y < 0) snake.y = canvas.height - grid;
    else if (snake.y >= canvas.height) snake.y = 0;

    snake.cells.unshift({ x: snake.x, y: snake.y });

    if (snake.cells.length > snake.maxCells) {
      snake.cells.pop();
    }

    context.fillStyle = '#ff0000';
    context.fillRect(apple.x, apple.y, grid - 1, grid - 1);
    context.fillStyle = '#00ff00';
```

```
        snake.cells.forEach(function (cell, index) {
          context.fillRect(cell.x, cell.y, grid - 1, grid - 1);

          if (cell.x === apple.x && cell.y === apple.y) {
            snake.maxCells++;
            score++;
            document.getElementById("score").innerText = "Score: " + score;
            placeApple();
          }

          for (var i = index + 1; i < snake.cells.length; i++) {
            if (cell.x === snake.cells[i].x && cell.y === snake.cells[i].y) {
              snake.x = 160;
              snake.y = 160;
              snake.cells = [];
              snake.maxCells = 4;
              snake.dx = grid;
              snake.dy = 0;
              score = 0;
              document.getElementById("score").innerText = "Score: " + score;
              placeApple();
            }
          }
        });
      }

      document.addEventListener('keydown', function (e) {
        if (e.which === 37 && snake.dx === 0) { snake.dx = -grid; snake.dy = 0; }
        else if (e.which === 38 && snake.dy === 0) { snake.dy = -grid; snake.dx = 0; }
        else if (e.which === 39 && snake.dx === 0) { snake.dx = grid; snake.dy = 0; }
        else if (e.which === 40 && snake.dy === 0) { snake.dy = grid; snake.dx = 0; }
      });
      requestAnimationFrame(loop);
    </script>
</body>
</html>
```

OUTPUT

QUESTION 7

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 7</title>

  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap');

    * {
      margin: 0;
      padding: 0;
    }

    body {
      height: 100vh;
      width: 100vw;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      font-family: "Poppins";
      background-color: bisque;
    }

    .container {
      border-radius: 15px;
      box-shadow: 0px 0px 15px black;
      background-color: #FFF;
    }

    .head {
      background-color: black;
      color: white;
      font-size: 50px;
      text-align: center;
      border-top-left-radius: 15px;
      border-top-right-radius: 15px;
    }

    .webcam-container {
      padding: 40px;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
    }
```

```
#webcam {
  width: 100%;
  height: auto;
  border-radius: 15px;
}

.controls {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 10px;
  margin: 10px;
}

.btn {
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 12px 20px;
  font-size: 14px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: all 0.3s ease;
  background-color: black;
  color: white;
}

.btn:hover {
  opacity: 0.9;
  transform: translateY(-2px);
}

.btn:disabled {
  background-color: #ccc;
  cursor: not-allowed;
  transform: none;
}

.gallery {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  gap: 15px;
  padding: 20px;
  background-color: #f9f9f9;
  border-top: 1px solid #eee;
  border-bottom-right-radius: 15px;
  border-bottom-left-radius: 15px;
}

.gallery-item {
  position: relative;
  overflow: hidden;
  border-radius: 8px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  background-color: white;
  width: 100%;
```

```
    height: 150px;
    object-fit: cover;
    display: block;
}

.gallery-actions {
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    background-color: rgba(0, 0, 0, 0.5);
    display: flex;
    justify-content: space-around;
    padding: 8px;
}

.gallery-btn {
    background-color: transparent;
    border: none;
    color: white;
    cursor: pointer;
    font-size: 14px;
}

.status {
    margin-top: 10px;
    padding: 8px;
    border-radius: 4px;
    text-align: center;
    font-weight: bold;
}

.status.recording {
    background-color: rgba(234, 67, 53, 0.1);
    color: #ea4335;
    animation: pulse 1.5s infinite;
}

@keyframes pulse {
    0% {
        opacity: 1;
    }
    50% {
        opacity: 0.5;
    }
    100% {
        opacity: 1;
    }
}

.record-timer {
    margin-top: 10px;
    font-size: 16px;
    font-weight: bold;
    color: #ea4335;
}
```

```
    .no-webcam {
      padding: 40px;
      text-align: center;
      background-color: #f8f8f8;
      border-radius: 8px;
      margin: 20px 0;
    }

    .hidden {
      display: none;
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="head">Webcam</div>

    <div class="webcam-container">
      <video id="webcam" autoplay playsinline></video>

      <div id="errorMessage" class="no-webcam hidden">
        <h3>Unable to access webcam</h3>
        <p>Please make sure you have a webcam connected and have granted permission to use it.</p>
      </div>

      <div class="controls">
        <button id="startBtn" class="btn btn-primary">Start Camera</button>
        <button id="snapshotBtn" class="btn btn-success" disabled>Take Snapshot</button>
        <button id="recordBtn" class="btn btn-danger" disabled>Start Recording</button>
      </div>

      <div id="recordingStatus" class="status hidden"></div>
      <div id="recordTimer" class="record-timer hidden">00:00</div>
    </div>

    <div id="gallery" class="gallery"></div>
  </div>

  <script>
    document.addEventListener('DOMContentLoaded', function () {
      const webcamElement = document.getElementById('webcam');
      const startBtn = document.getElementById('startBtn');
      const snapshotBtn = document.getElementById('snapshotBtn');
      const recordBtn = document.getElementById('recordBtn');
      const errorMessage = document.getElementById('errorMessage');
      const recordingStatus = document.getElementById('recordingStatus');
      const recordTimer = document.getElementById('recordTimer');
      const gallery = document.getElementById('gallery');

      let stream = null;
      let mediaRecorder = null;
      let recordedChunks = [];
      let isRecording = false;
      let recordingInterval = null;
      let recordingSeconds = 0;
```

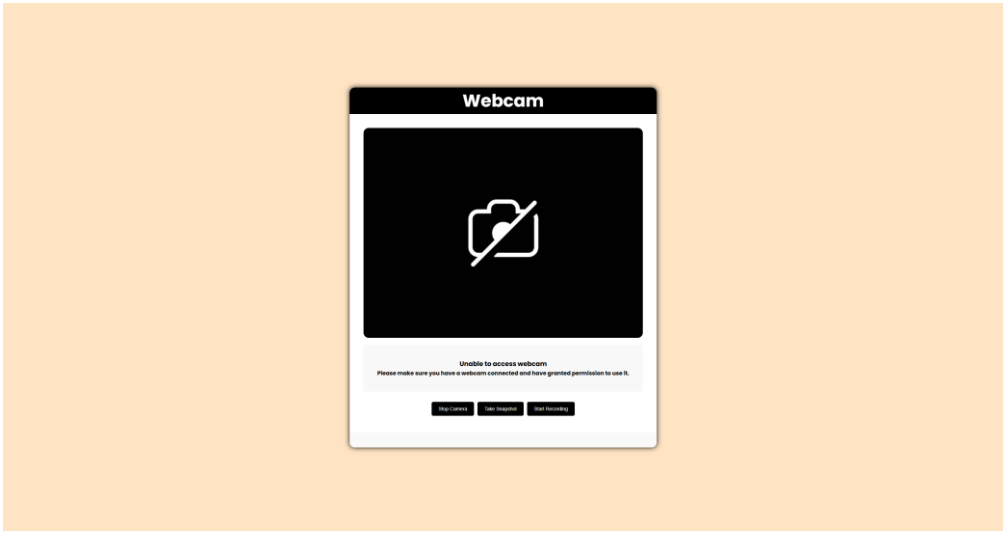
```
startBtn.addEventListener('click', async () => {
        try {
          if (stream) {
            stream.getTracks().forEach(track => track.stop());
            webcamElement.srcObject = null;
            stream = null;

            startBtn.innerHTML = `Start Camera`;
            snapshotBtn.disabled = true;
            recordBtn.disabled = true;
            switchCameraBtn.disabled = true;

            return;
          }

          stream = await navigator.mediaDevices.getUserMedia({
            video: true,
            audio: true
          });

          webcamElement.srcObject = stream;
          errorMessage.classList.add('hidden');

          startBtn.innerHTML = `Stop Camera`;
          snapshotBtn.disabled = false;
          recordBtn.disabled = false;
          switchCameraBtn.disabled = false;

        } catch (error) {
          console.error('Error accessing the webcam:', error);
          errorMessage.classList.remove('hidden');
        }
      });

      snapshotBtn.addEventListener('click', () => {
        if (!stream) return;

        const canvas = document.createElement('canvas');
        canvas.width = webcamElement.videoWidth;
        canvas.height = webcamElement.videoHeight;

        const ctx = canvas.getContext('2d');
        ctx.drawImage(webcamElement, 0, 0, canvas.width, canvas.height);

        const imageDataURL = canvas.toDataURL('image/png');

        addImageToGallery(imageDataURL);
      });

      recordBtn.addEventListener('click', () => {
        if (!stream) return;

        if (isRecording) {
          mediaRecorder.stop();
          isRecording = false;
```

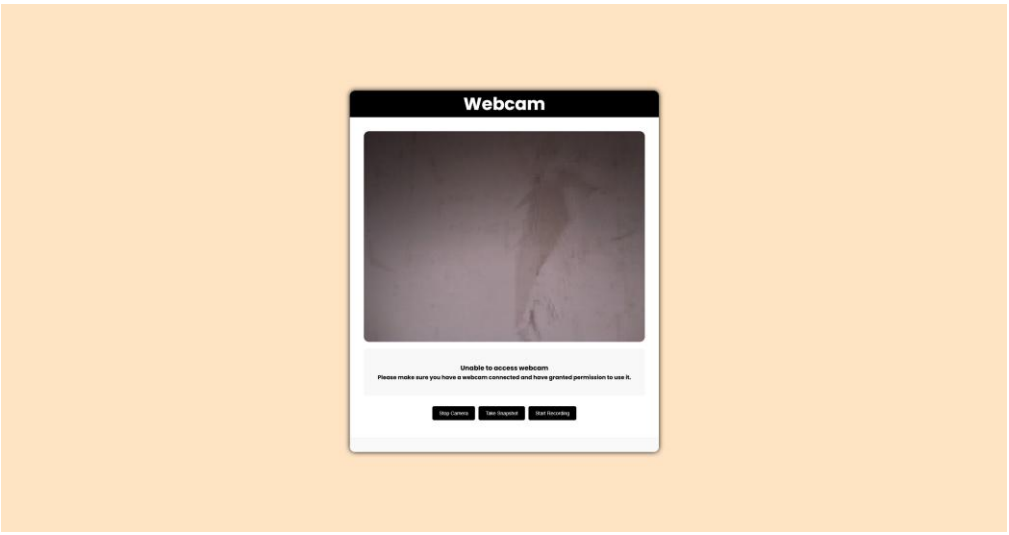
```
        recordBtn.innerHTML = `Start Recording`;
        recordingStatus.classList.add('hidden');
        recordTimer.classList.add('hidden');

        clearInterval(recordingInterval);
        recordingSeconds = 0;

      } else {
        recordedChunks = [];

        const options = { mimeType: 'video/webm;codecs=vp9,opus' };
        try {
          mediaRecorder = new MediaRecorder(stream, options);
        } catch (e) {
          console.error('MediaRecorder error:', e);
          try {
            mediaRecorder = new MediaRecorder(stream, { mimeType: 'video/webm' });
          } catch (e2) {
            console.error('MediaRecorder error with fallback:', e2);
            alert('Recording is not supported in this browser');
            return;
          }
        }

        mediaRecorder.ondataavailable = (event) => {
          if (event.data.size > 0) {
            recordedChunks.push(event.data);
          }
        };

        mediaRecorder.onstop = () => {
          const blob = new Blob(recordedChunks, { type: 'video/webm' });
          const videoURL = URL.createObjectURL(blob);

          addVideoToGallery(videoURL);
        };

        mediaRecorder.start(100);
        isRecording = true;

        recordBtn.innerHTML = `Stop Recording`;
        recordingStatus.textContent = 'Recording..';
        recordingStatus.classList.remove('hidden');
        recordingStatus.classList.add('recording');
        recordTimer.classList.remove('hidden');

        recordingInterval = setInterval(updateRecordingTime, 1000);
      }
    });

    function addImageToGallery(imageURL) {
      const item = document.createElement('div');
      item.className = 'gallery-item';

      const img = document.createElement('img');
      img.className = 'gallery-item';
      img.src = imageURL;
```

```
    img.alt = 'Captured photo';

    const actions = document.createElement('div');
    actions.className = 'gallery-actions';

    const downloadBtn = document.createElement('button');
    downloadBtn.className = 'gallery-btn';
    downloadBtn.textContent = 'Download';
    downloadBtn.addEventListener('click', () => {
      const link = document.createElement('a');
      link.href = imageURL;
      link.download = `snapshot_${new Date().toISOString()}.png`;
      link.click();
    });

    const deleteBtn = document.createElement('button');
    deleteBtn.className = 'gallery-btn';
    deleteBtn.textContent = 'Delete';
    deleteBtn.addEventListener('click', () => {
      gallery.removeChild(item);
    });

    actions.appendChild(downloadBtn);
    actions.appendChild(deleteBtn);

    item.appendChild(img);
    item.appendChild(actions);

    gallery.prepend(item);
  }

  function addVideoToGallery(videoURL) {
    const item = document.createElement('div');
    item.className = 'gallery-item';

    const video = document.createElement('video');
    video.className = 'gallery-item';
    video.src = videoURL;
    video.controls = true;

    const actions = document.createElement('div');
    actions.className = 'gallery-actions';

    const downloadBtn = document.createElement('button');
    downloadBtn.className = 'gallery-btn';
    downloadBtn.textContent = 'Download';
    downloadBtn.addEventListener('click', () => {
      const link = document.createElement('a');
      link.href = videoURL;
      link.download = `recording_${new Date().toISOString()}.webm`;
      link.click();
    });

    const deleteBtn = document.createElement('button');
    deleteBtn.className = 'gallery-btn';
    deleteBtn.textContent = 'Delete';
    deleteBtn.addEventListener('click', () => {
```

```
            gallery.removeChild(item);
            URL.revokeObjectURL(videoURL);
          });

          actions.appendChild(downloadBtn);
          actions.appendChild(deleteBtn);

          item.appendChild(video);
          item.appendChild(actions);

          gallery.prepend(item);
        }

        function updateRecordingTime() {
          recordingSeconds++;
          const minutes = Math.floor(recordingSeconds / 60);
          const seconds = recordingSeconds % 60;
          recordTimer.textContent = `${minutes.toString().padStart(2, '0')}:${seconds.toString().padStart(2, '0')}`;
        }
      });
    </script>
</body>
</html>
```

OUTPUT





QUESTION 8

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Question 8</title>

  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@700&display=swap');
    * {
      margin: 0;
      padding: 0;
    }

    body {
      height: 100vh;
      width: 100vw;
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      font-family: 'Poppins', sans-serif;
      background-color: #222;
      color: white;
    }

    .container {
      height: auto;
      width: 350px;
      text-align: center;
      padding: 30px;
      border-radius: 15px;
      background: rgba(255, 255, 255, 0.1);
      border: 1px solid gray;
    }

    .button-container {
      margin: 20px;
    }

    #toggleButton {
      background: linear-gradient(90deg, red, brown, orange);
      border: none;
      color: white;
      padding: 15px 40px;
      font-size: 18px;
      cursor: pointer;
      border-radius: 50px;
      transition: 0.3s;
      box-shadow: 0px 0px 10px rgba(255, 60, 60, 0.5);
    }
```

```
    #toggleButton:hover {
      box-shadow: 0px 0px 20px rgba(255, 140, 0, 0.7);
      transform: scale(1.05);
    }

    #toggleButton.off {
      background: linear-gradient(45deg, #4CAF50, #45a049);
      box-shadow: 0px 0px 10px rgba(76, 175, 80, 0.5);
    }

    #toggleButton.off:hover {
      box-shadow: 0px 0px 20px rgba(69, 160, 73, 0.7);
    }

    #status {
      font-size: 16px;
      text-shadow: 0 0 10px rgba(255, 255, 255, 0.5);
    }

    .error {
      color: #ff5757;
      margin-top: 15px;
      font-size: 14px;
    }

    #videoPreview {
      display: none;
    }
  </style>
</head>

<body>
  <div class="container">
    <h1>Flashlight Toggle</h1>

    <div class="button-container">
      <button id="toggleButton">Turn On</button>
    </div>

    <div id="status">Status: OFF</div>
    <div id="error" class="error"></div>

    <video id="videoPreview" autoplay muted playsinline></video>
  </div>

  <script>
    document.addEventListener('DOMContentLoaded', () => {
      const toggleButton = document.getElementById('toggleButton');
      const statusElement = document.getElementById('status');
      const errorElement = document.getElementById('error');
      const videoElement = document.getElementById('videoPreview');

      let stream = null;
      let track = null;
      let flashlightOn = false;
```

```
if (!navigator.mediaDevices || !navigator.mediaDevices.getUserMedia) {
  errorElement.textContent = 'Your browser does not support flashlight access.';
  toggleButton.disabled = true;
  return;
}

toggleButton.addEventListener('click', async () => {
  try {
    if (!flashlightOn) {
      stream = await navigator.mediaDevices.getUserMedia({
        video: {
          facingMode: 'environment',
          advanced: [{ torch: true }]
        }
      });

      videoElement.srcObject = stream;
      track = stream.getVideoTracks()[0];

      const capabilities = track.getCapabilities();

      if (capabilities.torch) {
        await track.applyConstraints({ advanced: [{ torch: true }] });
        flashlightOn = true;
        updateUI(true);
      } else {
        errorElement.textContent = 'Flashlight is not supported on your device.';
        stopStream();
      }
    } else {
      if (track) {
        await track.applyConstraints({ advanced: [{ torch: false }] });
      }
      stopStream();
      flashlightOn = false;
      updateUI(false);
    }
  } catch (error) {
    console.error('Error:', error);
    errorElement.textContent = `Error: ${error.message || 'Camera permission is required.'}`;
    stopStream();
    flashlightOn = false;
    updateUI(false);
  }
});

function stopStream() {
  if (stream) {
    stream.getTracks().forEach(track => track.stop());
    videoElement.srcObject = null;
    stream = null;
    track = null;
  }
}
```

```
        function updateUI(isOn) {
          if (isOn) {
            toggleButton.textContent = 'Turn Off Flashlight';
            toggleButton.classList.add('off');
            statusElement.textContent = 'Status: Flashlight is ON';
            errorElement.textContent = '';
          } else {
            toggleButton.textContent = 'Turn On Flashlight';
            toggleButton.classList.remove('off');
            statusElement.textContent = 'Status: Flashlight is OFF';
          }
        }
      });
    </script>
</body>
</html>
```

OUTPUT