**BCSE203E WEB PROGRAMMIMG**

**LAB EXERCISE -12**

**JavaScript canvas, charts and graphs using plotly.js and stack elements using Z-index**

NAME: M MENMANGAI

REGISTRATION NO:23BCE1022

**1.Write a JavaScript program using the HTML5 Canvas API to draw a scene that consists of the following shapes and corresponding drawings:**

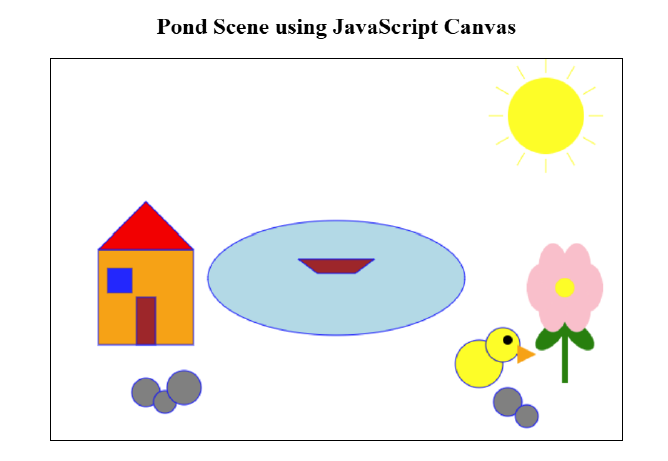
**Requirements:**

**• Use the Canvas API functions such as arc(), ellipse(), fillRect(), lineTo(), moveTo(), and stroke().**

**• Assign different colors to each shape.**

**• Ensure the relative positioning of the elements remains visually structured.**

**Sample Scene:**



CODE:

<html>

<head>

    <title>23BCE1022</title>

</head>

<body>

    <canvas id="sceneCanvas" width="600" height="400" style="border:1px solid black;"></canvas>

    <script>

        const canvas = document.getElementById("sceneCanvas");

        const ctx = canvas.getContext("2d");

        ctx.beginPath();

        ctx.ellipse(300, 250, 150, 50, 0, 0, 2 \* Math.PI);

        ctx.fillStyle = "lightblue";

        ctx.strokeStyle = "blue";

        ctx.fill();

        ctx.stroke();

        ctx.closePath();

        ctx.beginPath();

        ctx.moveTo(230, 240);

        ctx.lineTo(320, 240);

        ctx.lineTo(300, 260);

        ctx.lineTo(260, 260);

        ctx.fillStyle = "brown";

        ctx.strokeStyle = "blue";

        ctx.fill();

        ctx.stroke();

        ctx.beginPath();

        ctx.arc(500, 250, 10, 0, 2 \* Math.PI);

        ctx.arc(480, 260, 15, 0, 2 \* Math.PI);

        ctx.fillStyle = "yellow";

        ctx.strokeStyle = "blue";

        ctx.stroke();

        ctx.fill();

        ctx.closePath();

        ctx.beginPath();

        ctx.arc(505, 250, 3, 0, 2 \* Math.PI);

        ctx.fillStyle = "black";

        ctx.fill();

        ctx.closePath();

        ctx.beginPath();

        ctx.moveTo(510, 240);

        ctx.lineTo(520, 250);

        ctx.lineTo(510, 260);

        ctx.fillStyle = "orange";

        ctx.fill();

        ctx.closePath();

        ctx.beginPath();

        ctx.arc(500, 50, 30, 0, 2 \* Math.PI);

        ctx.fillStyle = "yellow";

        ctx.fill();

        ctx.closePath();

        // Draw House

        ctx.fillStyle = "orange";

        ctx.fillRect(60, 200, 80, 80);

        ctx.fillStyle = "blue";

        ctx.fillRect(70, 210, 20, 20);

        ctx.fillStyle = "brown";

        ctx.fillRect(90, 240, 20, 40);

        ctx.beginPath();

        ctx.moveTo(60, 200);

        ctx.lineTo(100, 150);

        ctx.lineTo(140, 200);

        ctx.closePath();

        ctx.fillStyle = "red";

        ctx.fill();

        ctx.beginPath();

        ctx.arc(550, 240, 10, 0, 2 \* Math.PI);

        ctx.arc(540, 240, 10, 0, 2 \* Math.PI);

        ctx.arc(560, 240, 10, 0, 2 \* Math.PI);

        ctx.arc(545, 250, 10, 0, 2 \* Math.PI);

        ctx.arc(555, 250, 10, 0, 2 \* Math.PI);

        ctx.arc(545, 230, 10, 0, 2 \* Math.PI);

        ctx.arc(555, 230, 10, 0, 2 \* Math.PI);

        ctx.fillStyle = "pink";

        ctx.fill();

        ctx.closePath();

        ctx.beginPath();

        ctx.arc(550, 240, 5, 0, 2 \* Math.PI);

        ctx.fillStyle = "yellow";

        ctx.fill();

        ctx.closePath();

        ctx.fillStyle = "green";

        ctx.fillRect(550, 255, 5, 30);

        ctx.beginPath();

        ctx.ellipse(560, 260, 10, 5, 4, 0, 2 \* Math.PI);

        ctx.ellipse(540, 260, 10, 5, 6, 0, 2 \* Math.PI);

        ctx.fillStyle = "green";

        ctx.fill();

        ctx.closePath();

        ctx.beginPath();

        ctx.arc(500, 290, 10, 0, 2 \* Math.PI);

        ctx.arc(510, 290, 10, 0, 2 \* Math.PI);

        ctx.fillStyle = "grey";

        ctx.strokeStyle = "blue";

        ctx.stroke();

        ctx.fill();

        ctx.closePath();

        ctx.beginPath();

        ctx.arc(100, 300, 10, 0, 2 \* Math.PI);

        ctx.arc(110, 300, 10, 0, 2 \* Math.PI);

        ctx.arc(120, 300, 10, 0, 2 \* Math.PI);

        ctx.fillStyle = "grey";

        ctx.strokeStyle = "blue";

        ctx.stroke();

        ctx.fill();

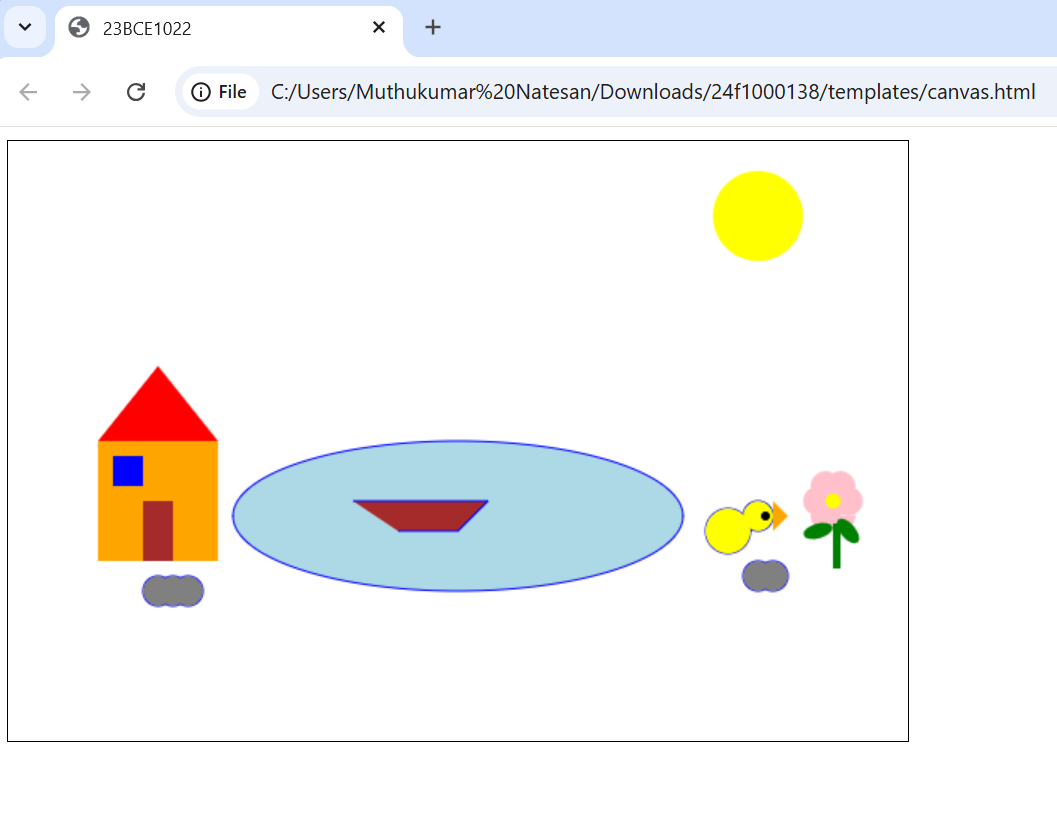
        ctx.closePath();

    </script>

</body>

</html>

OUTPUT:



**2.Apply an animation effect to the boat**

CODE:

<html>

<head>

    <title>23BCE1022</title>

</head>

<body>

    <canvas id="sceneCanvas" width="600" height="400" style="border:1px solid black;"></canvas>

    <script>

        const canvas = document.getElementById("sceneCanvas");

        const ctx = canvas.getContext("2d");

        function drawScene() {

            ctx.clearRect(0, 0, canvas.width, canvas.height);

            ctx.beginPath();

            ctx.ellipse(300, 250, 150, 50, 0, 0, 2 \* Math.PI);

            ctx.fillStyle = "lightblue";

            ctx.strokeStyle = "blue";

            ctx.fill();

            ctx.stroke();

            ctx.closePath();

            ctx.beginPath();

            ctx.arc(500, 250, 10, 0, 2 \* Math.PI);

            ctx.arc(480, 260, 15, 0, 2 \* Math.PI);

            ctx.fillStyle = "yellow";

            ctx.strokeStyle = "blue";

            ctx.stroke();

            ctx.fill();

            ctx.closePath();

            ctx.beginPath();

            ctx.arc(505, 250, 3, 0, 2 \* Math.PI);

            ctx.fillStyle = "black";

            ctx.fill();

            ctx.closePath();

            ctx.beginPath();

            ctx.moveTo(510, 240);

            ctx.lineTo(520, 250);

            ctx.lineTo(510, 260);

            ctx.fillStyle = "orange";

            ctx.fill();

            ctx.closePath();

            ctx.beginPath();

            ctx.arc(500, 50, 30, 0, 2 \* Math.PI);

            ctx.fillStyle = "yellow";

            ctx.fill();

            ctx.closePath();

            ctx.fillStyle = "orange";

            ctx.fillRect(60, 200, 80, 80);

            ctx.fillStyle = "blue";

            ctx.fillRect(70, 210, 20, 20);

            ctx.fillStyle = "brown";

            ctx.fillRect(90, 240, 20, 40);

            ctx.beginPath();

            ctx.moveTo(60, 200);

            ctx.lineTo(100, 150);

            ctx.lineTo(140, 200);

            ctx.closePath();

            ctx.fillStyle = "red";

            ctx.fill();

            ctx.beginPath();

            ctx.arc(550, 240, 10, 0, 2 \* Math.PI);

            ctx.arc(540, 240, 10, 0, 2 \* Math.PI);

            ctx.arc(560, 240, 10, 0, 2 \* Math.PI);

            ctx.arc(545, 250, 10, 0, 2 \* Math.PI);

            ctx.arc(555, 250, 10, 0, 2 \* Math.PI);

            ctx.arc(545, 230, 10, 0, 2 \* Math.PI);

            ctx.arc(555, 230, 10, 0, 2 \* Math.PI);

            ctx.fillStyle = "pink";

            ctx.fill();

            ctx.closePath();

            ctx.beginPath();

            ctx.arc(550, 240, 5, 0, 2 \* Math.PI);

            ctx.fillStyle = "yellow";

            ctx.fill();

            ctx.closePath();

            ctx.fillStyle = "green";

            ctx.fillRect(550, 255, 5, 30);

            ctx.beginPath();

            ctx.ellipse(560, 260, 10, 5, 4, 0, 2 \* Math.PI);

            ctx.ellipse(540, 260, 10, 5, 6, 0, 2 \* Math.PI);

            ctx.fillStyle = "green";

            ctx.fill();

            ctx.closePath();

            ctx.beginPath();

            ctx.arc(500, 290, 10, 0, 2 \* Math.PI);

            ctx.arc(510, 290, 10, 0, 2 \* Math.PI);

            ctx.fillStyle = "grey";

            ctx.strokeStyle = "blue";

            ctx.stroke();

            ctx.fill();

            ctx.closePath();

            ctx.beginPath();

            ctx.arc(100, 300, 10, 0, 2 \* Math.PI);

            ctx.arc(110, 300, 10, 0, 2 \* Math.PI);

            ctx.arc(120, 300, 10, 0, 2 \* Math.PI);

            ctx.fillStyle = "grey";

            ctx.strokeStyle = "blue";

            ctx.stroke();

            ctx.fill();

            ctx.closePath();

        }

        function drawBoat(x) {

            ctx.beginPath();

            ctx.moveTo(x, 240);

            ctx.lineTo(x + 100, 240);

            ctx.quadraticCurveTo(x + 50, 270, x, 240);

            ctx.fillStyle = "brown";

            ctx.fill();

            ctx.strokeStyle = "blue";

            ctx.stroke();

            ctx.closePath();

        }

        let boatX = 230;

        function animateBoat() {

            drawScene();

            drawBoat(boatX);

            boatX += 0.5;

            if (boatX > 350) {

                boatX = 170;

            }

            requestAnimationFrame(animateBoat);

        }

        drawScene();

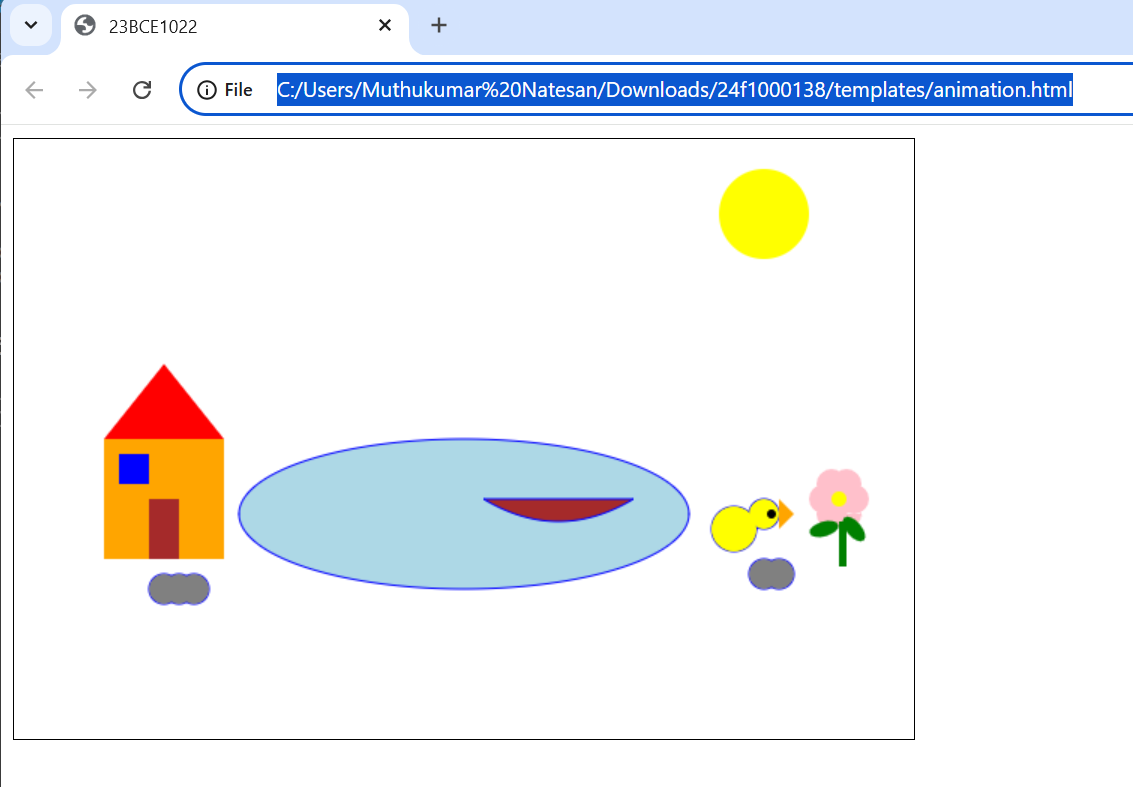
        animateBoat();

    </script>

</body>

</html>

OUTPUT:



**3.Write a JavaScript program that creates a working analog clock using the HTML5 Canvas API. The clock should display the current time dynamically and accurately, updating every second.**

**Requirements:**

**i) Use the Canvas API to draw the clock face, hands, and markings.**

**ii) The clock must include the following elements: a. A circular clock face with a border and a filled background color.**

**b. Hour, minute, and second hands that update dynamically based on the current time.**

**c. Numerical or tick markings for hours (1 to 12).**

**d. A center pivot point for the hands.**

**iii) Ensure the hands move smoothly and update every second.**

**CODE:**

<!DOCTYPE html>

<html >

<head>

    <title>23BCE1022</title>

</head>

<body>

    <canvas id="canvas" width="400" height="400" style="background-color: gray"></canvas>

    <script>

        const canvas = document.getElementById("canvas");

        const ctx = canvas.getContext("2d");

        let radius = canvas.height / 2;

        ctx.translate(radius, radius);

        radius = radius \* 0.90;

        setInterval(drawClock, 1000);

        function drawClock() {

            drawFace(ctx, radius);

            drawNumbers(ctx, radius);

            drawTime(ctx, radius);

        }

        function drawFace(ctx, radius) {

            const grad = ctx.createRadialGradient(0, 0, radius \* 0.95, 0, 0, radius \* 1.05);

            grad.addColorStop(0, '#333');

            grad.addColorStop(0.5, 'pink');

            grad.addColorStop(1, '#333');

            ctx.beginPath();

            ctx.arc(0, 0, radius, 0, 2 \* Math.PI);

            ctx.fillStyle = 'white';

            ctx.fill();

            ctx.strokeStyle = grad;

            ctx.lineWidth = radius \* 0.1;

            ctx.stroke();

            ctx.beginPath();

            ctx.arc(0, 0, radius \* 0.1, 0, 2 \* Math.PI);

            ctx.fillStyle = '#333';

            ctx.fill();

        }

        function drawNumbers(ctx, radius) {

            ctx.font = radius \* 0.15 + "px arial";

            ctx.textBaseline = "middle";

            ctx.textAlign = "center";

            for (let num = 1; num < 13; num++) {

                let ang = num \* Math.PI / 6;

                ctx.rotate(ang);

                ctx.translate(0, -radius \* 0.85);

                ctx.rotate(-ang);

                ctx.fillText(num.toString(), 0, 0);

                ctx.rotate(ang);

                ctx.translate(0, radius \* 0.85);

                ctx.rotate(-ang);

            }

        }

        function drawTime(ctx, radius) {

            const now = new Date();

            let hour = now.getHours();

            let minute = now.getMinutes();

            let second = now.getSeconds();

            hour = hour % 12;

            hour = (hour \* Math.PI / 6) + (minute \* Math.PI / (6 \* 60)) + (second \* Math.PI / (360 \* 60));

            drawHand(ctx, hour, radius \* 0.5, radius \* 0.07);

            minute = (minute \* Math.PI / 30) + (second \* Math.PI / (30 \* 60));

            drawHand(ctx, minute, radius \* 0.8, radius \* 0.07);

            second = (second \* Math.PI / 30);

            drawHand(ctx, second, radius \* 0.9, radius \* 0.02);

        }

        function drawHand(ctx, pos, length, width) {

            ctx.beginPath();

            ctx.lineWidth = width;

            ctx.lineCap = "round";

            ctx.moveTo(0, 0);

            ctx.rotate(pos);

            ctx.lineTo(0, -length);

            ctx.stroke();

            ctx.rotate(-pos);

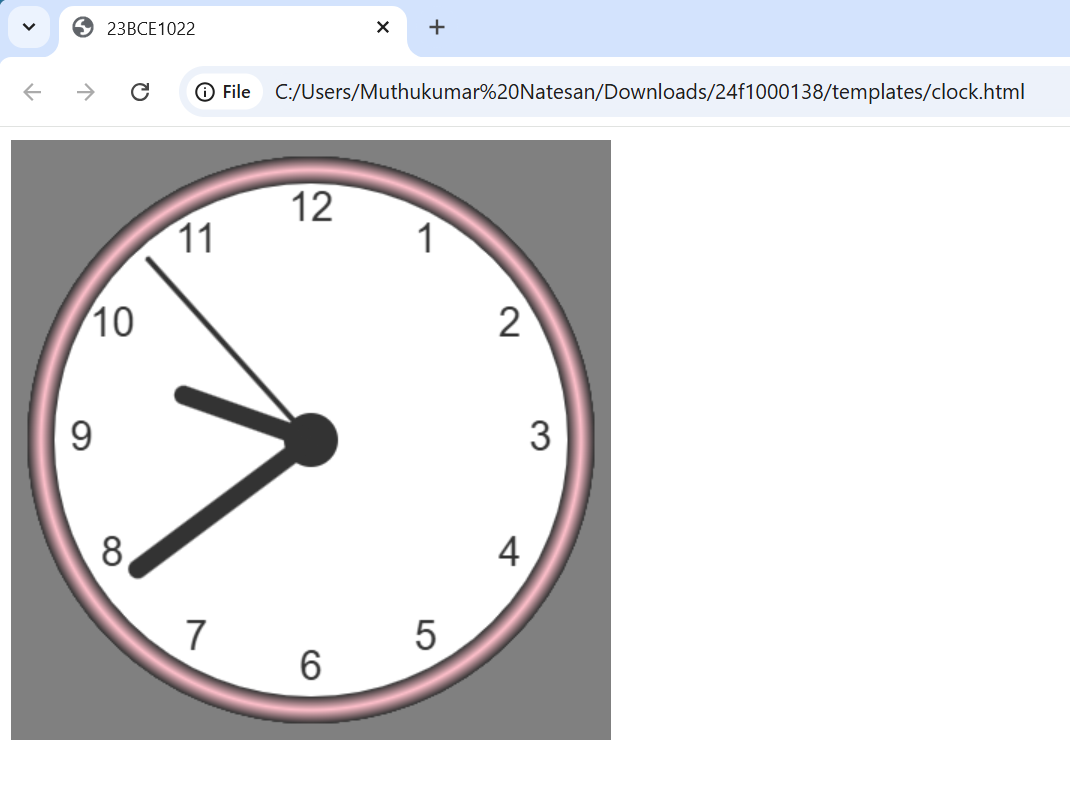
        }

    </script>

</body>

</html>

OUTPUT:



**4.Write a JavaScript program that dynamically generates the charts (bar chart, line chart, pie chart and a donut chart) using Plotly.js.**

**Each chart must include:**

**a. Labeled X and Y axes (for bar and line charts).**

**b. Title for each chart.**

**c. Different colors for data points.**

**d. Legend (for the pie chart and donut) showing categories.**

**ii) The chart should be scaled properly to fit within the display area.**

**CODE:**

<html>

<head>

    <title>23BCE1022</title>

    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

    <style>

        body {

            font-family: Arial, sans-serif;

            max-width: 1200px;

            margin: 0 auto;

            padding: 20px;

        }

        .chart-container {

            display: grid;

            grid-template-columns: repeat(2, 1fr);

            gap: 20px;

            margin-top: 20px;

        }

        .chart {

            border: 1px solid #ddd;

            border-radius: 5px;

            padding: 15px;

            box-shadow: 0 2px 4px rgba(0,0,0,0.1);

            height: 400px;

        }

        h1 {

            text-align: center;

            color: #333;

        }

    </style>

</head>

<body>

    <h1>Dynamic Charts</h1>

    <div class="chart-container">

        <div class="chart" id="barChart"></div>

        <div class="chart" id="lineChart"></div>

        <div class="chart" id="pieChart"></div>

        <div class="chart" id="donutChart"></div>

    </div>

    <script>

        function generateData() {

            const months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'];

            const categories = ['Category A', 'Category B', 'Category C', 'Category D'];

            return {

                months: months,

                categories: categories,

                barValues: categories.map(() => months.map(() => Math.floor(Math.random() \* 100) + 20)),

                lineValues: categories.map(() => months.map(() => Math.floor(Math.random() \* 100) + 10)),

                pieValues: categories.map(() => Math.floor(Math.random() \* 100) + 10)

            };

        }

        function generateColors(count) {

            const colorPalette = [

                '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728',

                '#9467bd', '#8c564b', '#e377c2', '#7f7f7f',

                '#bcbd22', '#17becf', '#aec7e8', '#ffbb78'

            ];

            if (count > colorPalette.length) {

                for (let i = colorPalette.length; i < count; i++) {

                    const r = Math.floor(Math.random() \* 255);

                    const g = Math.floor(Math.random() \* 255);

                    const b = Math.floor(Math.random() \* 255);

                    colorPalette.push(`rgb(${r},${g},${b})`);

                }

            }

            return colorPalette.slice(0, count);

        }

        function createBarChart(data) {

            const colors = generateColors(data.categories.length);

            const traces = [];

            for (let i = 0; i < data.categories.length; i++) {

                traces.push({

                    x: data.months,

                    y: data.barValues[i],

                    type: 'bar',

                    name: data.categories[i],

                    marker: {

                        color: colors[i]

                    }

                });

            }

            const layout = {

                title: 'Monthly Sales by Category',

                xaxis: {

                    title: 'Month',

                    titlefont: {

                        size: 14

                    }

                },

                yaxis: {

                    title: 'Sales Amount ($)',

                    titlefont: {

                        size: 14

                    }

                },

                barmode: 'group',

                legend: {

                    title: {

                        text: 'Categories'

                    }

                },

                autosize: true

            };

            Plotly.newPlot('barChart', traces, layout, {responsive: true});

        }

        function createLineChart(data) {

            const colors = generateColors(data.categories.length);

            const traces = [];

            for (let i = 0; i < data.categories.length; i++) {

                traces.push({

                    x: data.months,

                    y: data.lineValues[i],

                    type: 'scatter',

                    mode: 'lines+markers',

                    name: data.categories[i],

                    line: {

                        color: colors[i],

                        width: 3

                    },

                    marker: {

                        size: 8

                    }

                });

            }

            const layout = {

                title: 'Monthly Growth Trends',

                xaxis: {

                    title: 'Month',

                    titlefont: {

                        size: 14

                    }

                },

                yaxis: {

                    title: 'Growth Rate (%)',

                    titlefont: {

                        size: 14

                    }

                },

                legend: {

                    title: {

                        text: 'Categories'

                    }

                },

                autosize: true

            };

            Plotly.newPlot('lineChart', traces, layout, {responsive: true});

        }

        function createPieChart(data) {

            const colors = generateColors(data.categories.length);

            const trace = {

                labels: data.categories,

                values: data.pieValues,

                type: 'pie',

                marker: {

                    colors: colors

                },

                textinfo: 'label+percent',

                insidetextorientation: 'radial'

            };

            const layout = {

                title: 'Market Share Distribution',

                showlegend: true,

                legend: {

                    title: {

                        text: 'Categories'

                    }

                },

                autosize: true

            };

            Plotly.newPlot('pieChart', [trace], layout, {responsive: true});

        }

        function createDonutChart(data) {

            const colors = generateColors(data.categories.length);

            const trace = {

                labels: data.categories,

                values: data.pieValues,

                type: 'pie',

                hole: 0.4,

                marker: {

                    colors: colors

                },

                textinfo: 'label+percent',

                insidetextorientation: 'radial'

            };

            const layout = {

                title: 'Revenue Distribution by Category',

                showlegend: true,

                legend: {

                    title: {

                        text: 'Categories'

                    }

                },

                annotations: [{

                    font: {

                        size: 14

                    },

                    showarrow: false,

                    text: 'Revenue',

                    x: 0.5,

                    y: 0.5

                }],

                autosize: true

            };

            Plotly.newPlot('donutChart', [trace], layout, {responsive: true});

        }

        function initCharts() {

            const data = generateData();

            createBarChart(data);

            createLineChart(data);

            createPieChart(data);

            createDonutChart(data);

        }

        document.addEventListener('DOMContentLoaded', initCharts);

        window.addEventListener('resize', function() {

            const chartIds = ['barChart', 'lineChart', 'pieChart', 'donutChart'];

            chartIds.forEach(id => {

                Plotly.relayout(id, {autosize: true});

            });

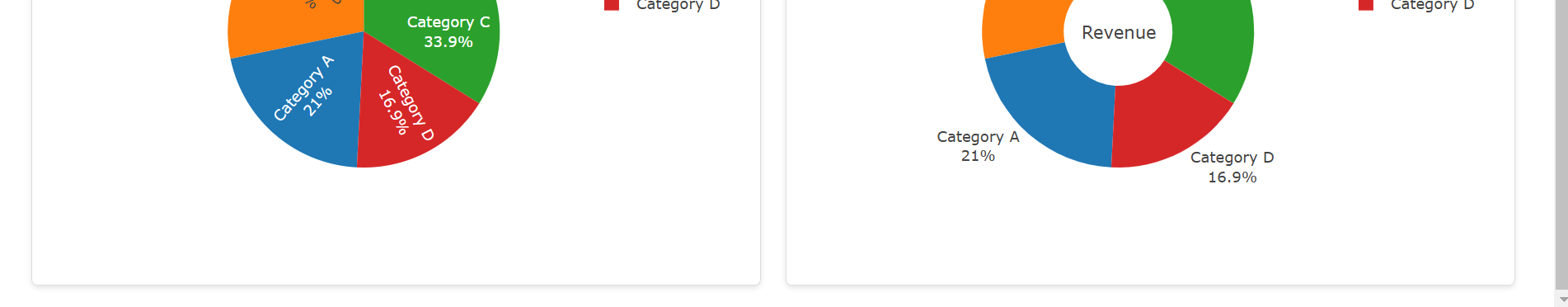
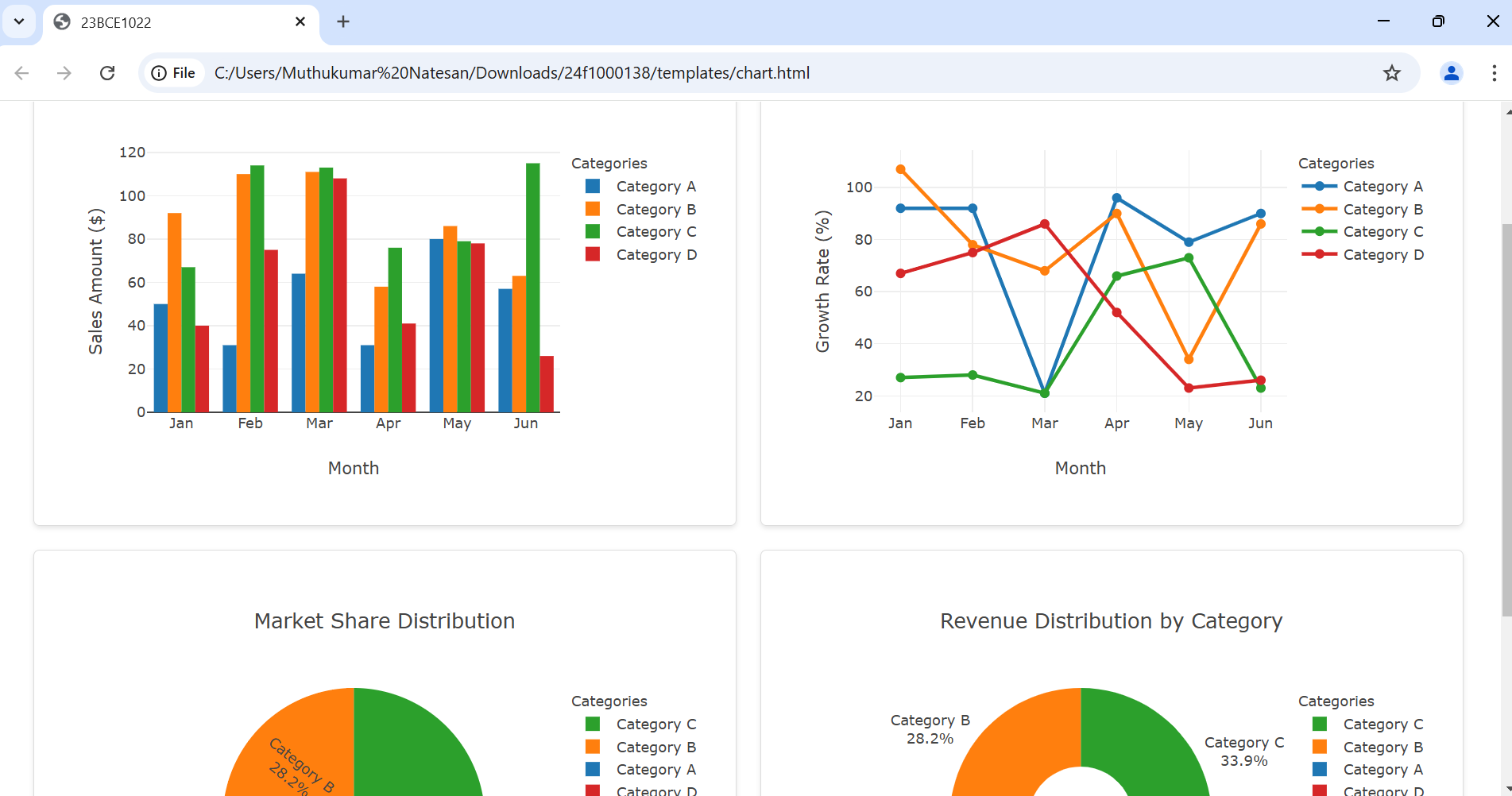
        });

    </script>

</body>

</html>

OUTPUT:



**5.Write a JavaScript program that dynamically creates and manipulates overlapping elements using CSS z-index. The program should allow the user to change the stacking order of elements by adjusting their z-index values.**

**• Create at least three overlapping elements (e.g., div boxes or images).**

**• Use CSS z-index to control the layering order of these elements.**

**• Provide buttons or user input to dynamically adjust the z-index values using JavaScript.**

**• Display the current z-index value of each element.**

CODE:

<!DOCTYPE html>

<html>

<head>

    <title>23BCE1022title>

    <style>

        .container {

            position: relative;

            width: 300px;

            height: 300px;

            border: 2px solid black;

        }

        .box {

            position: absolute;

            width: 150px;

            height: 150px;

            opacity: 0.8;

        }

        .red {

            background-color: red;

            z-index: 1;

            top: 20px;

            left: 20px;

        }

        .blue {

            background-color: blue;

            z-index: 2;

            top: 40px;

            left: 40px;

        }

        .green {

            background-color: green;

            z-index: 3;

            top: 60px;

            left: 60px;

        }

    </style>

</head>

<body>

    <div class="container">

        <div id="red" class="box red"></div>

        <div id="blue" class="box blue"></div>

        <div id="green" class="box green"></div>

    </div>

    <div class="controls">

        <button onclick="change('red',1);">Change z index of red box by +1</button>

        <button onclick="change('blue',1);">Change z index of blue box by +1</button>

        <button onclick="change('green',1);">Change z index of green box by +1</button>

        <button onclick="change('red',-1);">Change z index of red box by -1</button>

        <button onclick="change('blue',-1);">Change z index of blue box by -1</button>

        <button onclick="change('green',-1);">Change z index of green box by -1</button>

    </div>

    <div class="display">

        <p>current Z index of red: <span id="red-z">1</span></p>

        <p>current Z index of blue: <span id="blue-z">2</span></p>

        <p>current Z index of green: <span id="green-z">3</span></p>

    </div>

    <script>

        function change(val, ind) {

            var ele = document.getElementById(val);

            let curr = parseInt(window.getComputedStyle(ele).zIndex);

            let newZIndex = curr + ind;

            ele.style.zIndex = newZIndex;

            document.getElementById(val + '-z').textContent = newZIndex;

        }

    </script>

</body>

</html>

OUTPUT:

