

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>22BCE1351</title>

</head>

<body>

<canvas id="sceneCanvas" width="800" height="600"></canvas>

<script>

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

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

function drawPond(x, y, width, height) {

ctx.beginPath();

ctx.fillStyle = "lightblue";

ctx.ellipse(x, y, width, height, 0, 0, Math.PI \* 2);

ctx.fill();

}

function drawBoat(x, y, width, height) {

ctx.beginPath();

ctx.fillStyle = "brown";

ctx.moveTo(x, y);

ctx.bezierCurveTo(

x + width / 4,

y - height / 2,

x + (width \* 3) / 4,

y - height / 2,

x + width,

y

);

ctx.lineTo(x + width, y + height);

ctx.bezierCurveTo(

x + (width \* 3) / 4,

y + (height \* 3) / 2,

x + width / 4,

y + (height \* 3) / 2,

x,

y + height

);

ctx.closePath();

ctx.fill();

}

function drawDuck(x, y, bodyRadius, headRadius) {

ctx.beginPath();

ctx.fillStyle = "yellow";

ctx.arc(x, y, bodyRadius, 0, Math.PI \* 2);

ctx.fill();

ctx.beginPath();

ctx.fillStyle = "orange";

ctx.arc(x - bodyRadius / 2, y - bodyRadius / 2, headRadius, 0, Math.PI \* 2);

ctx.fill();

}

function drawSun(x, y, radius) {

ctx.beginPath();

ctx.fillStyle = "yellow";

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

ctx.fill();

for (*let* i = 0; i < 16; i++) {

*const* angle = (i / 16) \* Math.PI \* 2;

*const* lineLength = radius \* 0.5;

ctx.moveTo(x + Math.cos(angle) \* radius, y + Math.sin(angle) \* radius);

ctx.lineTo(

x + Math.cos(angle) \* (radius + lineLength),

y + Math.sin(angle) \* (radius + lineLength)

);

}

ctx.strokeStyle = "orange";

ctx.stroke();

}

function drawHouse(x, y, width, height) {

ctx.beginPath();

ctx.fillStyle = "orange";

ctx.fillRect(x, y, width, height);

ctx.beginPath();

ctx.fillStyle = "red";

ctx.moveTo(x, y);

ctx.lineTo(x + width, y);

ctx.lineTo(x + width / 2, y - height / 2);

ctx.closePath();

ctx.fill();

ctx.beginPath();

ctx.fillStyle = "blue";

ctx.fillRect(x + width / 4, y + height / 2, width / 4, height / 4);

ctx.fillStyle = "brown";

ctx.fillRect(x + width / 2, y + height / 2, width / 4, height / 4);

}

function drawFlower(ctx, x, y) {

ctx.fillStyle = "pink";

*let* petalRadius = 20;

for (*let* i = 0; i < 6; i++) {

*let* angle = (Math.PI / 3) \* i;

*let* petalX = x + Math.cos(angle) \* petalRadius;

*let* petalY = y + Math.sin(angle) \* petalRadius;

ctx.beginPath();

ctx.arc(petalX, petalY, petalRadius, 0, Math.PI \* 2);

ctx.fill();

}

ctx.fillStyle = "yellow";

ctx.beginPath();

ctx.arc(x, y, 10, 0, Math.PI \* 2);

ctx.fill();

ctx.fillStyle = "green";

ctx.fillRect(x - 5, y + 40, 10, 40);

ctx.fillStyle = "darkgreen";

function drawLeaf(leafX, leafY, scaleX, scaleY) {

ctx.beginPath();

ctx.ellipse(leafX, leafY, 15 \* scaleX, 10 \* scaleY, 0, 0, Math.PI \* 2);

ctx.fill();

}

drawLeaf(x - 12, y + 40, 1, 1);

drawLeaf(x + 12, y + 40, 1, 1);

}

function drawStones(x, y) {

*const* stones = [

{ x: x, y: y, radius: 20 },

{ x: x + 25, y: y + 10, radius: 12 },

{ x: x + 50, y: y - 5, radius: 25 },

];

stones.forEach((stone) => {

ctx.beginPath();

ctx.fillStyle = "gray";

ctx.strokeStyle = "blue";

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

ctx.fill();

ctx.stroke();

});

}

function animateBoat() {

*let* boatX = 300;

function moveBoat() {

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

drawScene(boatX);

boatX += 1;

if (boatX > canvas.width) boatX = 300;

requestAnimationFrame(moveBoat);

}

moveBoat();

}

function drawBird(ctx, x, y) {

ctx.beginPath();

ctx.fillStyle = "yellow";

ctx.strokeStyle = "blue";

ctx.arc(x, y, 30, 0, Math.PI \* 2);

ctx.fill();

ctx.stroke();

ctx.beginPath();

ctx.arc(x + 35, y - 15, 20, 0, Math.PI \* 2);

ctx.fill();

ctx.stroke();

ctx.beginPath();

ctx.fillStyle = "black";

ctx.arc(x + 42, y - 22, 5, 0, Math.PI \* 2);

ctx.fill();

ctx.save();

ctx.translate(x + 73, y - 10);

ctx.rotate(Math.PI);

ctx.beginPath();

ctx.fillStyle = "orange";

ctx.moveTo(0, 0);

ctx.lineTo(20, 10);

ctx.lineTo(20, -10);

ctx.closePath();

ctx.fill();

ctx.restore();

}

function drawScene(boatX = 300) {

drawPond(350, 300, 150, 70);

drawBoat(boatX, 300, 100, 20);

drawSun(700, 100, 50);

drawHouse(100, 400, 100, 100);

drawFlower(ctx, 600, 450);

drawStones(130, 580);

drawBird(ctx, 650, 400);

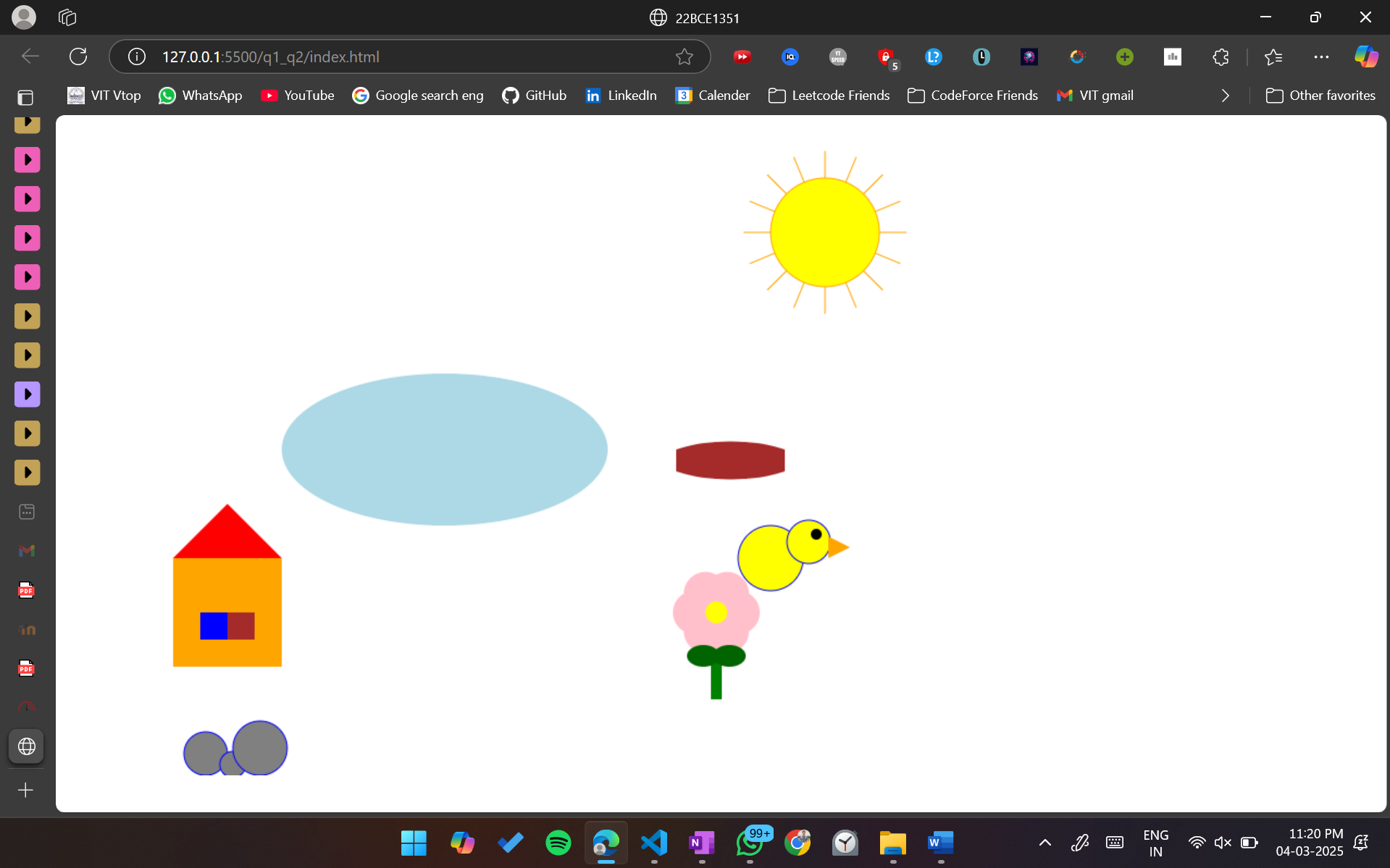
}

drawScene();

animateBoat();

</script>

</body></html>



**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>22BCE1351</title>

</head>

<body>

<canvas id="clock" width="400" height="400"></canvas>

<script>

window.onload = function() {

*const* canvas = document.getElementById('clock');

*const* ctx = canvas.getContext('2d');

*const* radius = canvas.height / 2 - 10;

*const* centerX = canvas.width / 2;

*const* centerY = canvas.height / 2;

function drawClock() {

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

ctx.beginPath();

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

ctx.fillStyle = '#f0f0f0';

ctx.fill();

ctx.strokeStyle = '#333';

ctx.lineWidth = 8;

ctx.stroke();

ctx.font = '24px Arial';

ctx.textAlign = 'center';

ctx.textBaseline = 'middle';

ctx.fillStyle = '#333';

for (*let* i = 1; i <= 12; i++) {

*const* angle = (i \* Math.PI / 6) - Math.PI / 2;

*const* x = centerX + Math.cos(angle) \* (radius \* 0.85);

*const* y = centerY + Math.sin(angle) \* (radius \* 0.85);

ctx.fillText(i, x, y);

}

*const* now = new Date();

*const* hours = now.getHours();

*const* minutes = now.getMinutes();

*const* seconds = now.getSeconds();

*const* millis = now.getMilliseconds();

*const* hourAngle = (hours % 12 + minutes/60 + seconds/3600 + millis/3600000) \* Math.PI/6 - Math.PI/2;

*const* minuteAngle = (minutes + seconds/60 + millis/60000) \* Math.PI/30 - Math.PI/2;

*const* secondAngle = (seconds + millis/1000) \* Math.PI/30 - Math.PI/2;

ctx.beginPath();

ctx.moveTo(centerX, centerY);

ctx.lineTo(

centerX + Math.cos(hourAngle) \* (radius \* 0.5),

centerY + Math.sin(hourAngle) \* (radius \* 0.5)

);

ctx.strokeStyle = '#333';

ctx.lineWidth = 8;

ctx.lineCap = 'round';

ctx.stroke();

ctx.beginPath();

ctx.moveTo(centerX, centerY);

ctx.lineTo(

centerX + Math.cos(minuteAngle) \* (radius \* 0.7),

centerY + Math.sin(minuteAngle) \* (radius \* 0.7)

);

ctx.strokeStyle = '#333';

ctx.lineWidth = 6;

ctx.lineCap = 'round';

ctx.stroke();

ctx.beginPath();

ctx.moveTo(centerX, centerY);

ctx.lineTo(

centerX + Math.cos(secondAngle) \* (radius \* 0.8),

centerY + Math.sin(secondAngle) \* (radius \* 0.8)

);

ctx.strokeStyle = '#e74c3c';

ctx.lineWidth = 2;

ctx.lineCap = 'round';

ctx.stroke();

ctx.beginPath();

ctx.arc(centerX, centerY, 8, 0, 2 \* Math.PI);

ctx.fillStyle = '#333';

ctx.fill();

requestAnimationFrame(drawClock);

}

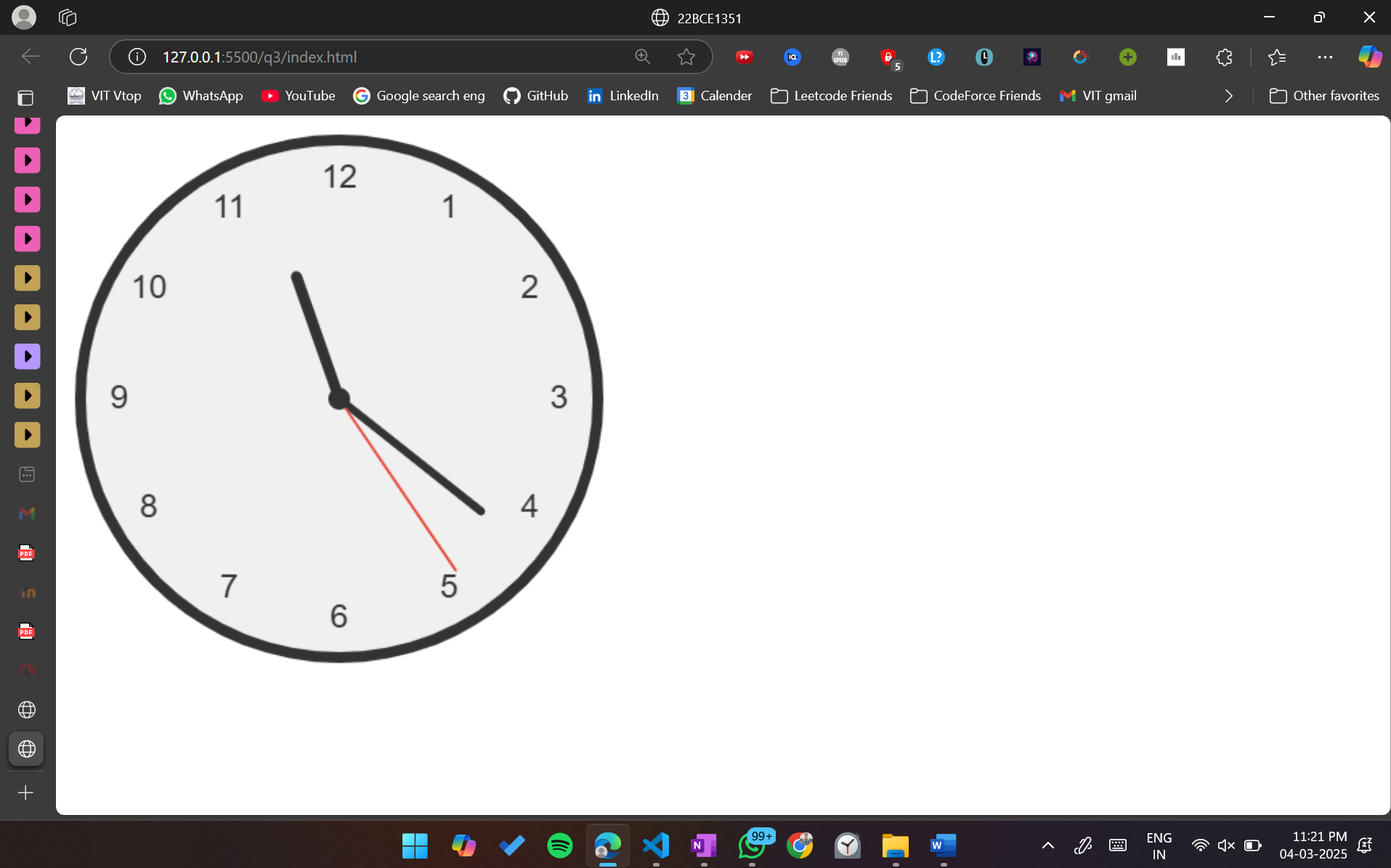
drawClock();

};

</script>

</body>

</html>



**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:**

<!DOCTYPE html>

<html>

<head>

<title>22BCE1351</title>

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

<style>

.chart-container {

display: grid;

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

gap: 20px;

padding: 20px;

max-width: 1200px;

margin: 0 auto;

}

.chart {

width: 100%;

height: 400px;

border: 1px solid #ddd;

border-radius: 8px;

box-shadow: 0 2px 8px rgba(0,0,0,0.1);

}

</style>

</head>

<body>

<div class="chart-container">

<div id="bar-chart" class="chart"></div>

<div id="line-chart" class="chart"></div>

<div id="pie-chart" class="chart"></div>

<div id="donut-chart" class="chart"></div>

</div>

<script>

document.addEventListener('DOMContentLoaded', () => {

*// Bar Chart*

*const* barData = [{

x: ['Jan', 'Feb', 'Mar', 'Apr', 'May'],

y: [23, 45, 12, 67, 34],

type: 'bar',

marker: {

color: ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4', '#FFEEAD'],

line: { width: 2, color: '#2A2A2A' }

},

name: 'Sales'

}];

*const* barLayout = {

title: 'Monthly Sales Performance',

xaxis: { title: 'Month' },

yaxis: { title: 'Sales (USD)' },

showlegend: true

};

Plotly.newPlot('bar-chart', barData, barLayout);

*// Line Chart*

*const* lineData = [{

x: [2019, 2020, 2021, 2022, 2023],

y: [120, 150, 170, 190, 220],

mode: 'lines+markers',

line: { color: '#45B7D1', width: 3 },

marker: {

color: ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4', '#FFEEAD'],

size: 10

},

name: 'Revenue'

}];

*const* lineLayout = {

title: 'Annual Revenue Growth',

xaxis: { title: 'Year' },

yaxis: { title: 'Revenue (Millions USD)' },

showlegend: true

};

Plotly.newPlot('line-chart', lineData, lineLayout);

*// Pie Chart*

*const* pieData = [{

values: [25, 35, 20, 15, 5],

labels: ['Category A', 'Category B', 'Category C', 'Category D', 'Category E'],

type: 'pie',

marker: {

colors: ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4', '#FFEEAD']

},

textinfo: 'label+percent',

hoverinfo: 'label+value'

}];

*const* pieLayout = {

title: 'Market Share Distribution',

showlegend: true

};

Plotly.newPlot('pie-chart', pieData, pieLayout);

*const* donutData = [{

values: [30, 25, 20, 15, 10],

labels: ['Region 1', 'Region 2', 'Region 3', 'Region 4', 'Region 5'],

type: 'pie',

hole: 0.4,

marker: {

colors: ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4', '#FFEEAD']

},

textinfo: 'label+percent',

hoverinfo: 'label+value'

}];

*const* donutLayout = {

title: 'Regional Sales Distribution',

showlegend: true

};

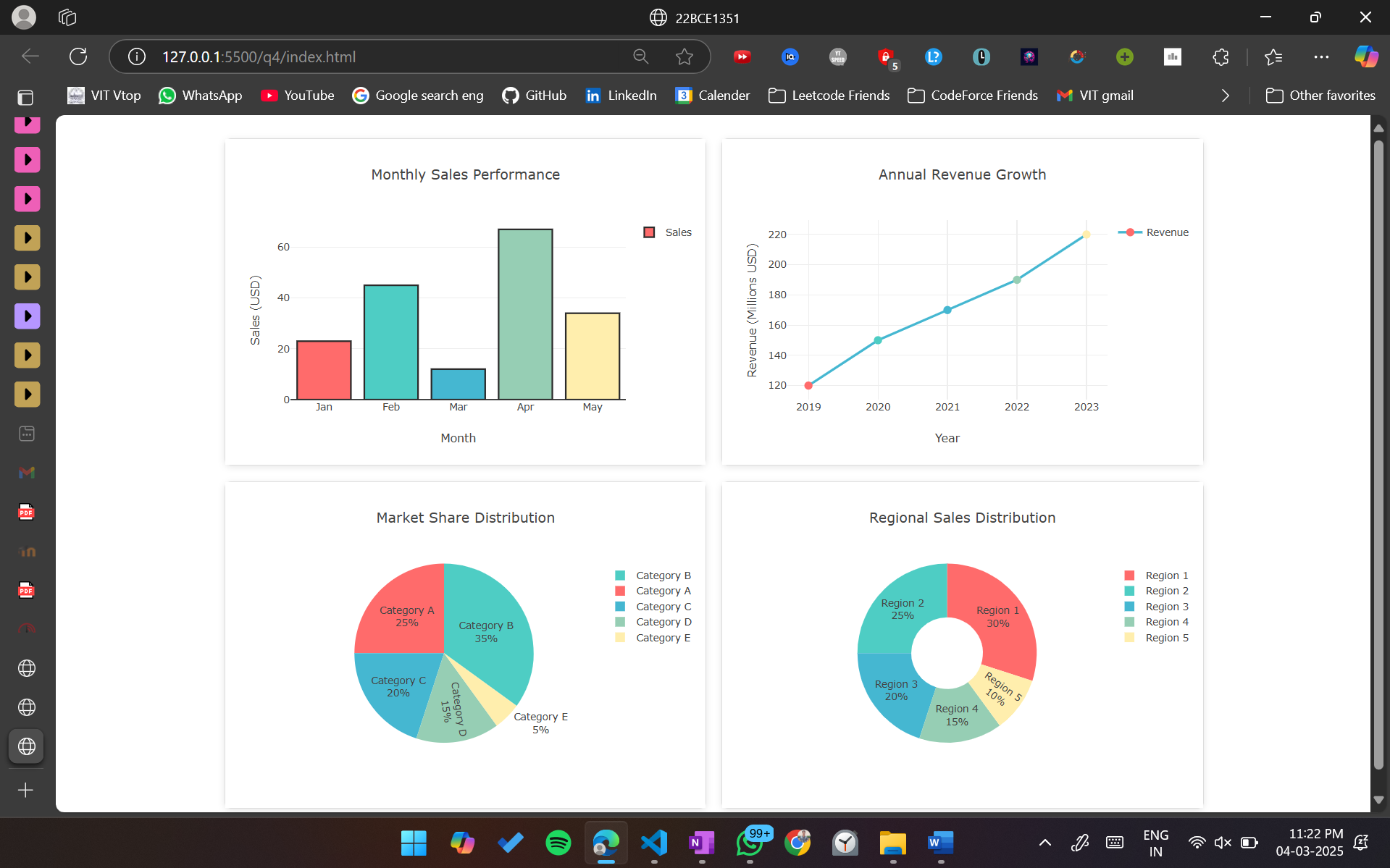
Plotly.newPlot('donut-chart', donutData, donutLayout);

});

</script>

</body>

</html>



**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>22BCE1351</title>

<style>

.container {

position: relative;

width: 600px;

height: 400px;

margin: 20px auto;

}

.box {

position: absolute;

width: 200px;

height: 200px;

border: 2px solid #333;

justify-content: center;

font-size: 1.2em;

color: white;

}

#box1 {

background-color: #FF6B6B;

top: 50px;

left: 50px;

z-index: 1;

}

#box2 {

background-color: #4ECDC4;

top: 100px;

left: 100px;

z-index: 2;

}

#box3 {

background-color: #45B7D1;

top: 150px;

left: 150px;

z-index: 3;

}

.controls {

margin-top: 20px;

padding: 20px;

background-color: #f0f0f0;

border-radius: 8px;

text-align: center;

}

.control-group {

margin: 15px 0;

}

</style>

</head>

<body>

<div class="container">

<div class="box" id="box1">Box 1</div>

<div class="box" id="box2">Box 2</div>

<div class="box" id="box3">Box 3</div>

</div>

<div class="controls">

<div class="control-group">

<label>Box 1 Z-Index: </label>

<input type="number" id="input1" value="1"

oninput="updateZIndex('box1', this.value)">

<span id="z1">1</span>

</div>

<div class="control-group">

<label>Box 2 Z-Index: </label>

<input type="number" id="input2" value="2"

oninput="updateZIndex('box2', this.value)">

<span id="z2">2</span>

</div>

<div class="control-group">

<label>Box 3 Z-Index: </label>

<input type="number" id="input3" value="3"

oninput="updateZIndex('box3', this.value)">

<span id="z3">3</span>

</div>

</div>

<script>

function updateZIndex(elementId, value) {

*const* element = document.getElementById(elementId);

*const* display = document.querySelector(`span#${elementId.replace('box', 'z')}`);

if (element && display) {

element.style.zIndex = value;

display.textContent = value;

}

}

document.getElementById('z1').textContent = 1;

document.getElementById('z2').textContent = 2;

document.getElementById('z3').textContent = 3;

</script>

</body>

</html>

