NAME-Himangi Bhatt

UID-23BCC70020

SUB-FULL STACK

# EXP-06

• **AIM:-** To design and develop a full-stack web application that allows users to draw shapes on an SVG canvas using mouse events, and save the drawings using backend APIs.

* **THEORY:**- SVG (Scalable Vector Graphics)
* SVG is an XML-based format for vector graphics that allows shapes like circles, rectangles, lines, and paths to be drawn in the browser with precision and scalability.
* Mouse Events
* Mouse events like mousedown, mousemove, and mouseup allow for capturing user input for drawing shapes or freehand paths.
* Frontend Technologies
* HTML/CSS/JavaScript
* SVG for canvas-like drawing
* Event Listeners for mouse input
* Backend Technologies
* Node.js with Express.js (or Python Flask)
* Database: MongoDB or SQLite (for storing SVG markup or shape data)
* Real-time (optional)
* WebSockets (e.g., Socket.io) for collaborative drawing.

• **CODE:-**

➢ **Front-end:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>SVG Drawing Tool</title>

<style> svg { border: 1px solid black; background: #f9f9f9;

}

</style>

</head>

<body>

<h2>Draw on SVG Canvas</h2>

<svg id="canvas" width="600" height="400"></svg>

<button onclick="saveDrawing()">Save Drawing</button>

<script> const svg = document.getElementById('canvas');

let drawing = false; let currentPath;

svg.addEventListener('mousedown', (e) => {

drawing = true; const pt = getMousePosition(e);

currentPath = document.createElementNS('http://www.w3.org/2000/svg', 'path');

currentPath.setAttribute('d', `M ${pt.x} ${pt.y}`); currentPath.setAttribute('stroke', 'black'); currentPath.setAttribute('fill', 'none'); currentPath.setAttribute('stroke-width', '2'); svg.appendChild(currentPath);

});

svg.addEventListener('mousemove', (e) => {

if (!drawing) return; const pt = getMousePosition(e); const d = currentPath.getAttribute('d'); currentPath.setAttribute('d', `${d} L ${pt.x} ${pt.y}`); });

svg.addEventListener('mouseup', () => { drawing = false;

});

function getMousePosition(evt) { const CTM = svg.getScreenCTM();

return {

x: (evt.clientX - CTM.e) / CTM.a, y: (evt.clientY - CTM.f) / CTM.d

};

}

function saveDrawing() { const svgData = svg.innerHTML;

fetch('/api/save', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ drawing: svgData }) }).then(res => res.json())

.then(data => alert('Drawing saved!'))

.catch(err => console.error(err));

}

</script>

</body>

</html>

➢ **Back-End:**

// server.js const express = require('express'); const bodyParser = require('body-parser'); const fs = require('fs'); const app = express(); const PORT = 3000;

app.use(bodyParser.json()); app.use(express.static('public'));

app.post('/api/save', (req, res) => { const { drawing } = req.body; fs.writeFile('drawing.svg', `<svg xmlns="http://www.w3.org/2000/svg" width="600" height="400">${drawing}</svg>`, (err) => { if (err) { console.error(err); res.status(500).send({ message: 'Failed to save drawing' });

} else { res.send({ message: 'Drawing saved' });

}

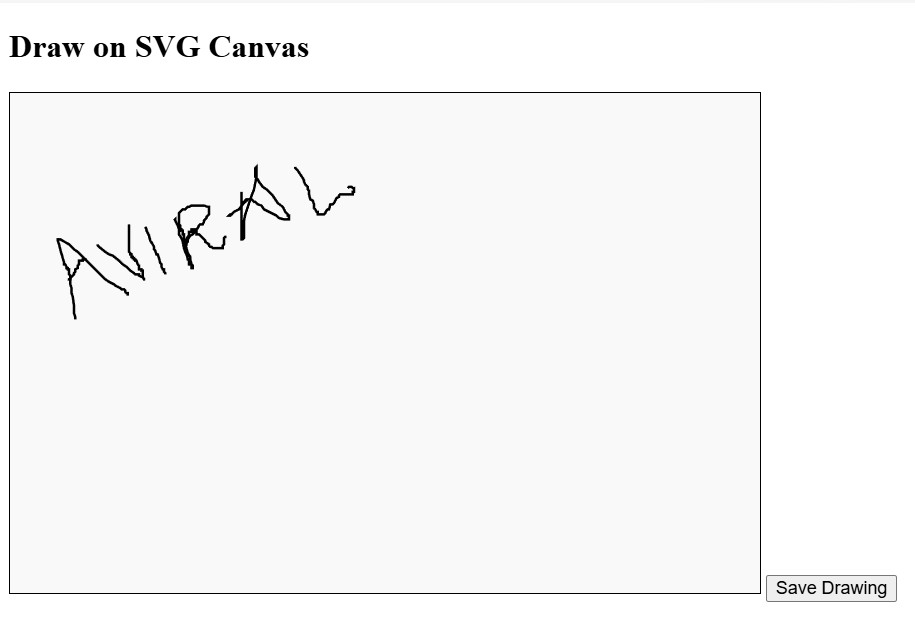
});

});

app.listen(PORT, () => { console.log(`Server running on http://localhost:${PORT}`);

});

* **OUTPUT:-**



* **LEARNING OUTCOMES:**-
* Interactive Front-End Development – Gained hands-on experience in handling mouse events and dynamically drawing on an SVG canvas using JavaScript.
* Client–Server Communication – Learned how to send data from the browser to the backend using the Fetch API and HTTP POST requests.
* Back-End Development with Express.js – Understood how to create API endpoints, parse JSON data, and handle requests in Node.js with Express.
* File Handling and Data Persistence – Acquired skills in saving user-generated drawings as SVG files using Node’s fs module.
* Full-Stack Project Integration – Developed a complete workflow connecting front-end drawing, server-side processing, and file storage into a functioning mini web application.