NAME-AVIRAL SRIVASTAVA; UID-23BCC70028; SUB-FULL STACK

EXP-04

- **AIM:-** To develop a full-stack web application that displays the live character count as a user types into a text area and stores the entered text on a backend server.
- **THEORY:-** A live character counter is a common feature in text-based applications such as social media platforms, messaging apps, and form validations. It provides real-time feedback to the user on the number of characters typed, which can help maintain input constraints and improve user experience.

✓ In this experiment:
✓ **Frontend (HTML, CSS, JavaScript)** updates the character count dynamically using the input event listener.
✓ **Backend (Node.js + Express)** provides APIs to store and retrieve the text.
✓ **AJAX (Fetch API)** is used for communication between the frontend and backend without reloading the page.
✓ **Key concepts:**
✓ DOM manipulation to update count
✓ Event-driven programming using input events
✓ RESTful API handling in Express.js
✓ Serving static files using express.static()

- **CODE:-**

✓ **Server.js**

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const path = require("path");

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Serve static files from public folder
app.use(express.static(path.join(__dirname, "public")));

let savedText = "";
```

```
app.post("/save-text", (req, res) => {
   const { text } = req.body;
   savedText = text;
   console.log(`Received text: "${text}" (${text.length} characters)`);
   res.json({ message: "Text saved successfully!", length: text.length });
});

app.get("/get-text", (req, res) => {
   res.json({ text: savedText, length: savedText.length });
n

app.listen(3000, () => {
   console.log("✅ Server running at http://localhost:3000");
});
```

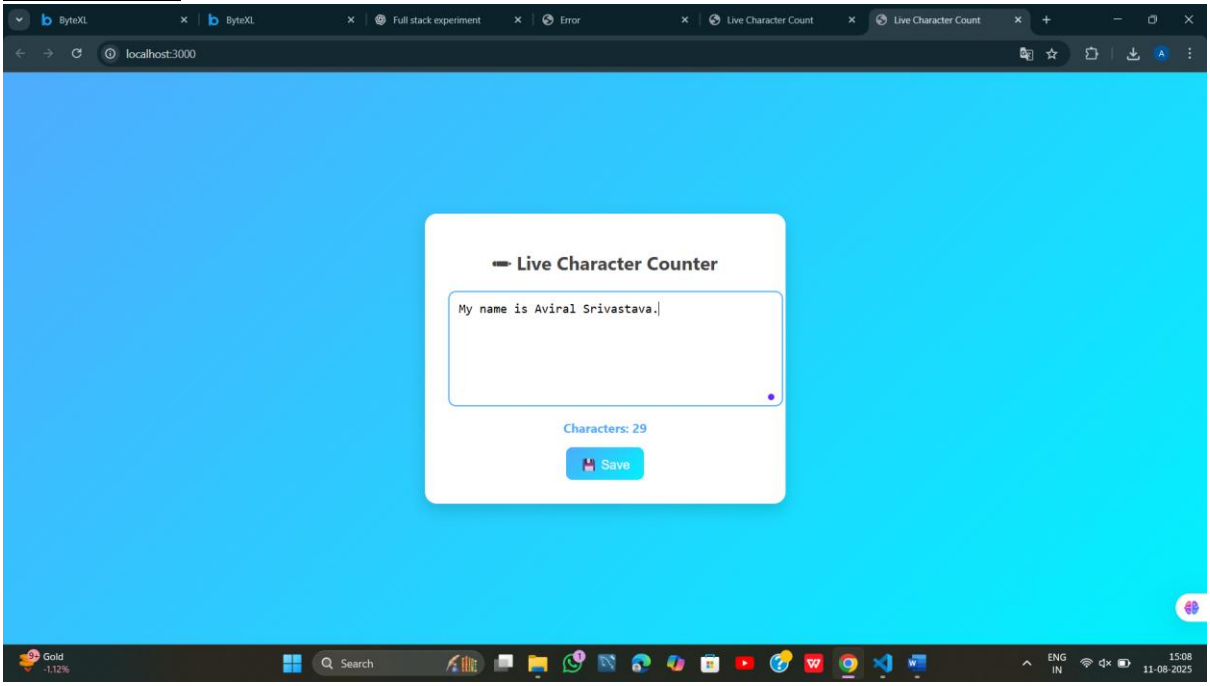
✓ **Index.html:-**

```
✓ <!DOCTYPE html>
✓ <html lang="en">
✓ <head>
✓     <meta charset="UTF-8">
✓     <title>Live Character Count</title>
✓     <style>
✓         /* Reset & base styles */
✓         body {
✓             font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
  serif;
✓             background: linear-gradient(135deg, #4facfe, #00f2fe);
✓             display: flex;
✓             justify-content: center;
✓             align-items: center;
✓             height: 100vh;
✓             margin: 0;
✓             color: #333;
✓         }
✓
✓         .container {
✓             background: white;
✓             padding: 30px;
✓             border-radius: 15px;
✓             box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
✓             width: 400px;
✓             text-align: center;
✓             animation: fadeIn 0.6s ease-in-out;
✓         }
✓
✓         h1 {
```

```
            margin-bottom: 20px;
            font-size: 24px;
            color: #444;
        }

        textarea {
            width: 100%;
            height: 120px;
            padding: 12px;
            font-size: 16px;
            border: 2px solid #ddd;
            border-radius: 10px;
            resize: none;
            outline: none;
            transition: border-color 0.3s;
        }

        textarea:focus {
            border-color: #4facfe;
        }

        .count {
            margin-top: 12px;
            font-weight: bold;
            font-size: 16px;
            color: #4facfe;
        }

        button {
            margin-top: 15px;
            padding: 10px 18px;
            font-size: 16px;
            border: none;
            border-radius: 8px;
            background: linear-gradient(135deg, #4facfe, #00f2fe);
            color: white;
            cursor: pointer;
            transition: transform 0.2s ease, box-shadow 0.2s ease;
        }

        button:hover {
            transform: translateY(-2px);
            box-shadow: 0 5px 15px rgba(79, 172, 254, 0.4);
        }

        @keyframes fadeIn {
            from { opacity: 0; transform: translateY(15px); }
            to { opacity: 1; transform: translateY(0); }
```

```
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>✏️ Live Character Counter</h1>
        <textarea id="textArea" placeholder="Type something
amazing..."></textarea>
        <div class="count">Characters: <span
id="charCount">0</span></div>
        <button id="saveBtn">💾 Save</button>
    </div>

    <script>
        const textArea = document.getElementById("textArea");
        const charCount = document.getElementById("charCount");
        const saveBtn = document.getElementById("saveBtn");

        // Update count live
        textArea.addEventListener("input", () => {
            charCount.textContent = textArea.value.length;
        });

        // Save text to backend
        saveBtn.addEventListener("click", async () => {
            const res = await fetch("/save-text", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ text: textArea.value })
            });
            const data = await res.json();
            alert(`${data.message} (${data.length} characters)`);
        });

        // Load saved text
        window.onload = async () => {
            const res = await fetch("/get-text");
            const data = await res.json();
            textArea.value = data.text;
            charCount.textContent = data.length;
        };
    </script>
    <script src="server.js"></script>
</body>
</html>
```

- **OUTPUT:-**



- **LEARNING OUTCOMES:-**

✓ Implement **real-time UI updates** using JavaScript event listeners.
✓ Create and consume **REST APIs** using Node.js and Express.
✓ Serve static frontend files from a backend server.
✓ Use **Fetch API** for asynchronous communication.
✓ Apply **basic UI design** for improved user experience.