INDEX

S.No	DATE	EXPERMENT	PAGE No	MARK	SIGN
1 a)		Create a portfolio website for a Profile			
1 b)		Create a portfolio website for a Fictional Business			
2		Create a Web Page and implement Java Script functions			
3		Build a Web Application using Sentimental Analysis API			
4		Build a Web Page that performs Linear Regression			
5		Adobe Sensei			
				Total	

ExNo:1 A) Date:	Create a portfolio website for a Profile

AIM :

To design and develop a responsive personal portfolio website showcasing skills, projects, and contact details.

ALGORITHM :

Step 1 :Set up HTML structure with sections: Header, About, Skills, Projects, and Contact.

Step 2 :Style the webpage using CSS for layout, typography, and responsiveness.

Step 3 :Add navigation links for smooth scrolling between sections.

Step 4 :Display key information like resume, skills, and major projects.

Step 5 :Provide contact details including email and LinkedIn.

Step 6 :Enhance user experience with clean design and interactive elements (e.g., buttons).

PROGRAM:

INDEX.HTML

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

```
<head>
<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Sreeshma G | Portfolio</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<header>
<h1>Sreeshma G</h1>
<p>AI & Full-Stack Developer</p>
<nav>
<ul>
<li><a href="#about">About</a></li>
<li><a href="#skills">Skills</a></li>
<li><a href="#projects">Projects</a></li>
<li><a href="#contact">Contact</a></li>
</ul>
</nav>
</header>

<section id="about">
<h2>About Me</h2>
<p>Final-year B.Tech (AIML) student passionate about AI and software development. Skilled in building scalable applications, AI-driven solutions, and full-stack development.</p>
```

Download Resume

</section>

<section id="skills">

<h2>Skills</h2>

Python, C++, Java

</section>

<section id="projects">

<h2>Projects</h2>

<h3>AI-Based Fitness Trainer</h3>

Developed an AI-powered fitness trainer using MediaPipe and OpenCV for real-time posture correction.

<h3>Diet Recommendation System</h3>

Designed a diet planner using K-Means clustering to generate personalized diet recommendations.

```
</section>

<section id="contact">
<h2>Contact</h2>
<p>Email: sreeshma.g.aiml.2022@snsct.org</p>
<p>LinkedIn: <a href="https://www.linkedin.com/in/sreeshma-g"

target="_blank">linkedin.com/in/sreeshma-g</a></p>
</section>

<footer>
<p>© 2024 Sreeshma G | All Rights Reserved</p>
</footer>
</body>
</html>
```

STYLE.CSS

```
/* General Reset */
* {
  margin: 0;
  padding: 0;

  box-sizing: border-box;
}
```

```
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f9f9f9;
color: #333;
  line-height: 1.6;
}

/* Header */
header {
  background-color: #1e293b;
color: white;
  padding: 2rem 1rem;
  text-align: center;
}

header h1 {
  font-size: 2.5rem;
  margin-bottom: 0.5rem;
}

header p {
  font-size: 1.2rem;
color: #cbd5e1;
}
```

```
nav ul {
  list-style: none;
  display: flex;
  justify-content: center;
  margin-top: 1rem;
  flex-wrap: wrap;
}

nav ul li {
  margin: 0 15px;
}

nav ul li a {
color: #f1f5f9;
  text-decoration: none;
  font-weight: bold;
  transition: color 0.3s;
}

nav ul li a:hover {
color: #38bdf8;
}
/* Section Styling */
section {
  padding: 4rem 2rem;
```

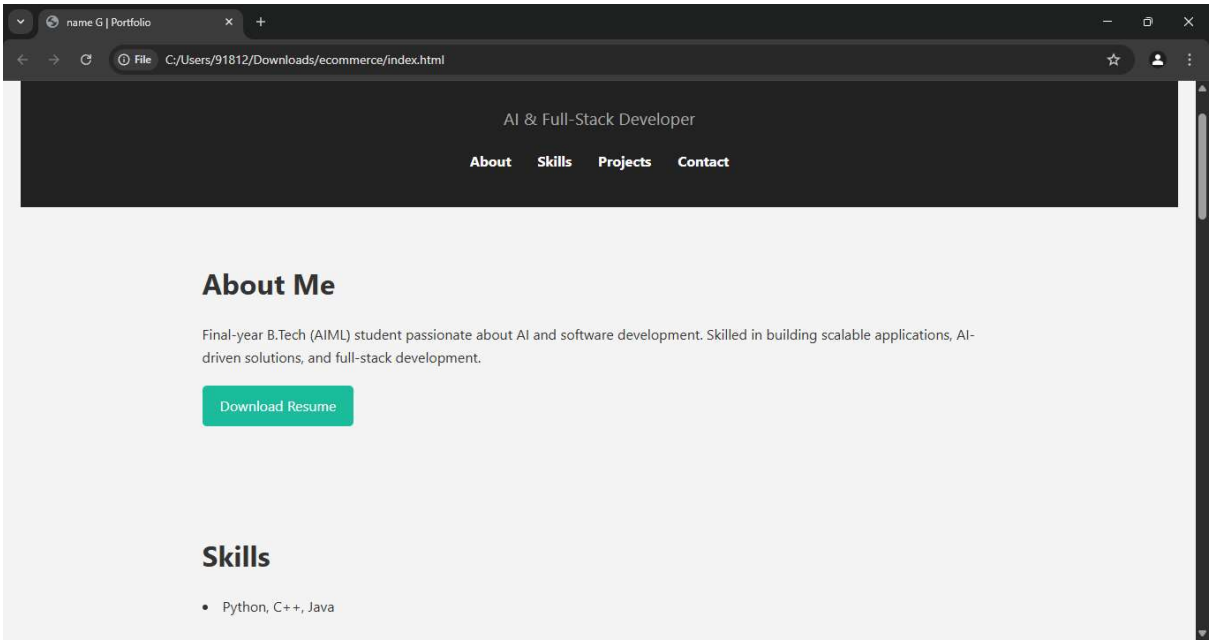
```
  max-width: 900px;
  margin: auto;
}

h2 {
  font-size: 2rem;
  margin-bottom: 1rem;

color: #0f172a;
}

/* About */
#about p {
  font-size: 1.1rem;
  margin-bottom: 1rem;
}

.btn {
  display: inline-block;
  background-color: #38bdf8;
color: white;
  padding: 0.7rem 1.2rem;
  border-radius: 5px;
  text-decoration: none;
```

```
  transition: background-color 0.3s;
}

.btn:hover {
  background-color: #0284c7;
}

/* Skills */
#skills ul {


  list-style: disc inside;
  font-size: 1.1rem;
}

/* Projects */
.project {
  background-color: #e2e8f0;
  padding: 1.5rem;
  margin-bottom: 1rem;
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
.project h3 {
  margin-bottom: 0.5rem;
```

```
color: #1e40af;
}

/* Contact */
#contact a {
color: #0ea5e9;
  text-decoration: none;
}

#contact a:hover {



  text-decoration: underline;
}

/* Footer */
footer {
  background-color: #1e293b;
color: white;
  text-align: center;
  padding: 1rem 0;
  margin-top: 2rem;
  font-size: 0.9rem;
}
```

```
/* Responsive */
@media (max-width: 600px) {
  nav ul {
    flex-direction: column;
    gap: 10px;
  }
  header h1 {
    font-size: 2rem;
  }
  section {
    padding: 2rem 1rem;
  }
}
```

OUTPUT :



RESULT:

Thus the above program executed and the output is verified successfully.

ExNo:1 B) Date:	Create a portfolio website for a Fictional Business

AIM :

To design and develop a responsive and visually appealing website for a fictional business using HTML and CSS.

ALGORITHM :

Step 1 :Set up the HTML structure with key sections: Header, Home, About Us, Services/Products, Testimonials (optional), and Contact.

Step 2 :Style the webpage using CSS for layout, color scheme, fonts, and responsiveness across devices.

Step 3 :Implement a navigation bar with links for smooth scrolling or switching between sections.

Step 4 :Highlight business details such as company description, services/products offered, and unique selling points.

Step 5 :Include contact information such as business email, phone number, location, and social media links.

Step 6 :Enhance user experience with a clean, professional design and interactive elements (e.g., hover effects, call-to-action buttons).

PROGRAM :

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>FreshBite Café</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <header>
        <nav class="navbar">
            <div class="logo">FreshBite Café</div>
            <ul class="nav-links">
                <li><a href="#home">Home</a></li>
                <li><a href="#about">About</a></li>
                <li><a href="#menu">Menu</a></li>
                <li><a href="#contact">Contact</a></li>
            </ul>
        </nav>
    </header>
```

```
  <section id="home" class="hero">
    <h1>Welcome to FreshBite Café</h1>
    <p>Delicious, Organic & Fresh Food Made with Love</p>
  </section>

  <section id="about" class="section">
    <h2>About Us</h2>
    <p>At FreshBite Café, we serve homemade meals made with organic ingredients sourced from local farms. Whether you’re craving a fresh salad or a hearty sandwich, we’ve got you covered!</p>
  </section>

  <section id="menu" class="section">
    <h2>Our Menu</h2>
    <ul class="menu-list">
      <li><strong>Avocado Toast</strong> - ₹150</li>
      <li><strong>Veggie Wrap</strong> - ₹120</li>
      <li><strong>Green Smoothie</strong> - ₹100</li>
      <li><strong>Cold Brew Coffee</strong> - ₹90</li>
    </ul>
  </section>

  <section id="contact" class="section">
    <h2>Contact Us</h2>
    <p>Email: hello@freshbitecafe.com</p>
    <p>Phone: +91 98765 43210</p>
```

```
        <p>Location: 123 Green Street, Bangalore, India</p>
    </section>

    <footer>
        <p>&copy; 2025 FreshBite Café. All rights reserved.</p>
    </footer>

</body>
</html>
```

STYLE.CSS

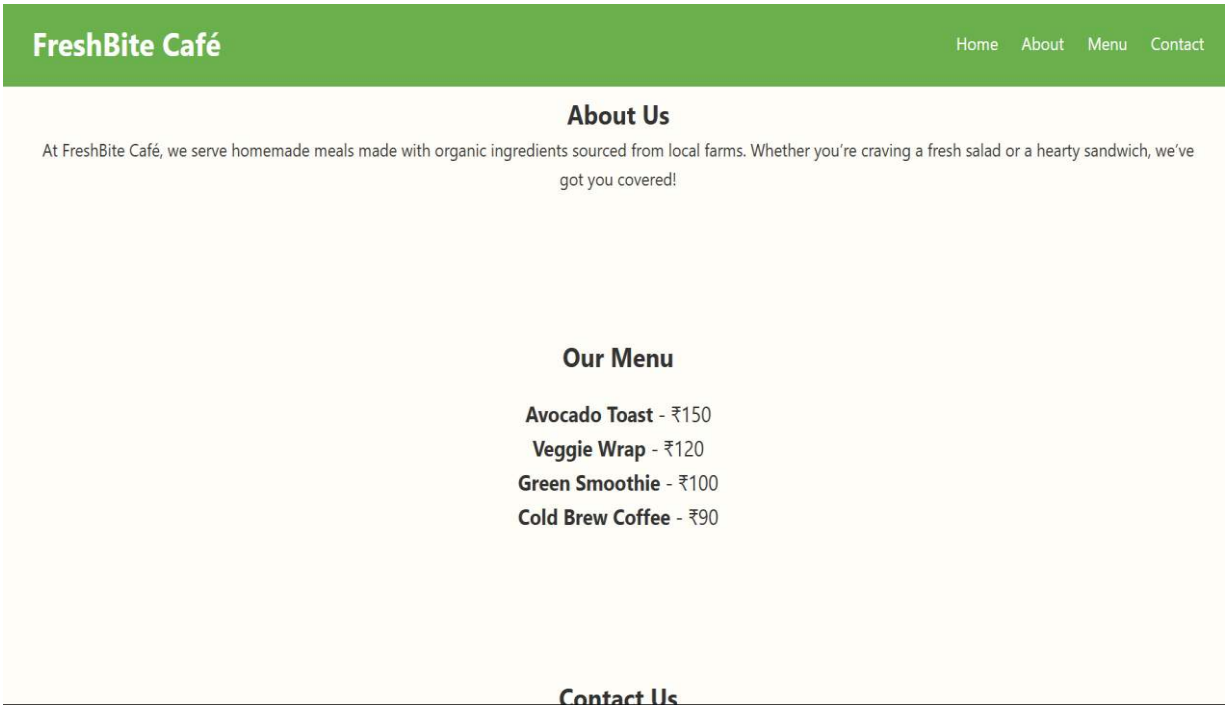
```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', sans-serif;
}

body {
    background-color: #fffdf7;
    color: #333;
    line-height: 1.6;
}

header {
```

```
    background-color: #6ab04c;
    padding: 1rem 2rem;
    position: sticky;
    top: 0;
    z-index: 1000;
}

.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.logo {
    font-size: 1.8rem;
    color: white;
    font-weight: bold;
}

.nav-links {
    list-style: none;
    display: flex;
    gap: 1.5rem;
}

.nav-links a {
```

```
    text-decoration: none;
    color: white;
    font-size: 1rem;
    transition: color 0.3s;
}

.nav-links a:hover {
    color: #f6e58d;
}

.hero {
    background: url('https://images.unsplash.com/photo-1556911220-e15b29be8c8f') no-repeat centercenter/cover;
    color: white;
    height: 70vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-align: center;
    padding: 2rem;
}
.hero h1 {
    font-size: 3rem;
    margin-bottom: 1rem;
}
```

```
.section {
    padding: 4rem 2rem;
    text-align: center;
}

.menu-list {
    list-style: none;
    padding: 0;
    font-size: 1.2rem;
    margin-top: 1rem;
}

footer {
    background-color: #6ab04c;
    color: white;
    text-align: center;
    padding: 1rem;
    margin-top: 2rem;
}
```

OUTPUT :



RESULT :

Thus the above program executed and the output is verified successfully.

ExNo:2 Date:	Create a Web Page and implement Java Script Functions

AIM :

To create a web tool that processes input text by converting it to lowercase, removing punctuation, and tokenizing it into individual words.

ALGORITHM :

Step 1 :Initialize Interface: Design the user interface with input (textarea), trigger (button), and output sections.

Step 2 : Style Interface: Apply styles to improve layout, readability, and user experience.

Step 3 :Define Processing Logic: Create a function that triggers on button click to handle text processing.

Step 4 : Capture & Normalize Input: Retrieve user input and convert it to lowercase for uniformity.

Step 5 :Clean & Tokenize Text: Remove punctuation using regular expressions and split the cleaned text into individual words (tokens).

Step 6 :Output Results: Display the processed versions—lowercase text, punctuation-free text, and tokenized words—in their respective output sections.

PROGRAM :

<!DOCTYPE html>

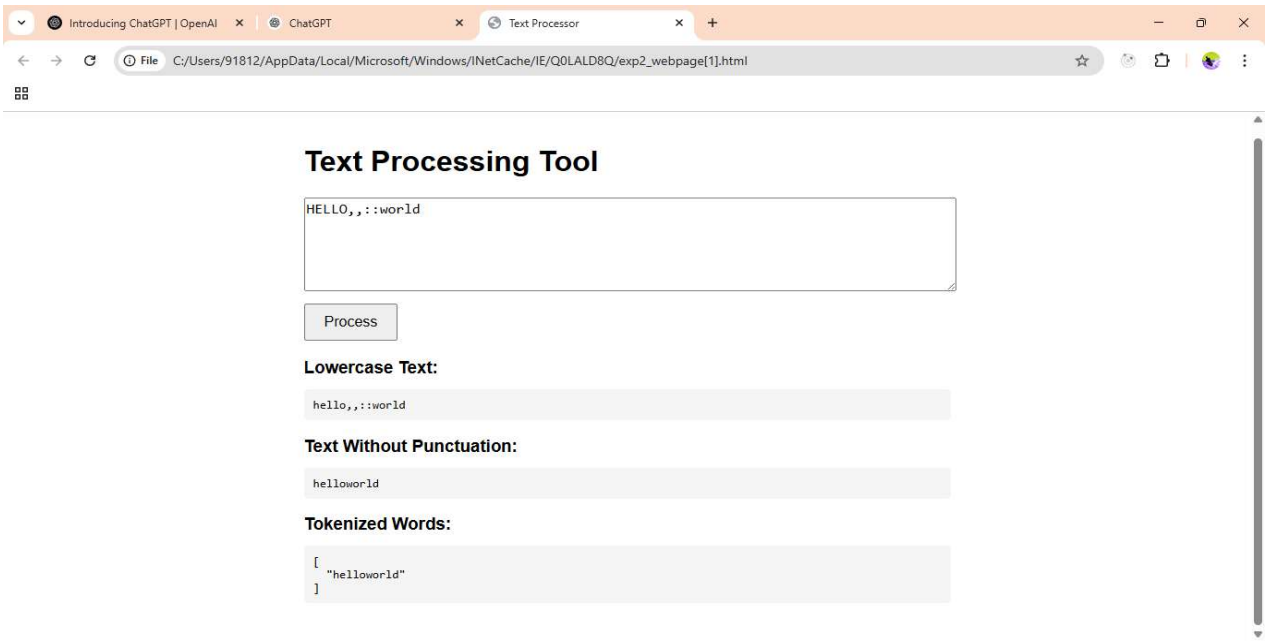
<html lang="en">

```
<head>
<meta charset="UTF-8">
<title>Text Processor</title>
<style>
    body { font-family: Arial, sans-serif; padding: 30px; max-width: 700px; margin: auto; }
textarea{ width: 100%; height: 100px; font-size: 16px; }
    button { margin-top: 10px; padding: 10px 20px; font-size: 16px; }
.output{ margin-top: 20px; }
.output h3 { margin-bottom: 5px; }
.output pre { background: #f5f5f5; padding: 10px; border-radius: 5px; }
</style>
</head>
<body>

<h1>Text Processing Tool</h1>
<textarea id="inputText" placeholder="Enter your text here..."></textarea>
<br>
<button onclick="processText()">Process</button>
```

```
<div class="output">

<h3>Lowercase Text:</h3>

<pre id="lowercaseText"></pre>

<h3>Text Without Punctuation:</h3>

<pre id="noPunctuationText"></pre>



<h3>Tokenized Words:</h3>

<pre id="tokenizedText"></pre>

</div>

<script>

    function processText() {

const text = document.getElementById("inputText").value;



      // Convert to lowercase

const lower = text.toLowerCase();

document.getElementById("lowercaseText").textContent = lower;

      // Remove punctuation using regex

constnoPunctuation = lower.replace(/[.,\/#!$%\^&\*;:{}=\-_`~()?"']/g, "");

document.getElementById("noPunctuationText").textContent = noPunctuation;
```

```
    // Tokenize into words (split by spaces)
const tokens = noPunctuation.split(/\s+/).filter(word =>word.length> 0);
document.getElementById("tokenizedText").textContent = JSON.stringify(tokens, null, 2);
    }
</script>
</body>
</html>
```

OUTPUT:



RESULT:

Thus the above program executed and the output is verified successfully.

ExNo:3 Date:	Build a Web Application using Sentimental Analysis API

AIM :

To detect and display the sentiment (positive, negative, or neutral) of user-input text using Google Cloud Natural Language API.

ALGORITHM :

Step 1 :Input Collection: Accept user input text from the HTML <textarea> on the webpage.

Step 2 :Send Request: On clicking “Analyze”, send the input to the backend via a POST request.

Step 3 :Process with API: Backend uses Google Cloud Natural Language API to analyze sentiment of the text.

Step 4 :Evaluate Sentiment: Based on the returned sentiment score, classify the result as positive, negative, or neutral.

Step 5 :Display Output: Send the result back to the frontend and display the sentiment and score on the webpage.

CODING :

Front End Code :

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

```
<meta charset="UTF-8">
<title>Sentiment Analyzer</title>
<style>
    body {
      font-family: Arial, sans-serif;
      max-width: 600px;
      margin: 2rem auto;
    }

textarea {
      width: 100%;
      height: 100px;
      margin-bottom: 1rem;
    }
    button {
      padding: 10px 15px;
      font-size: 16px;
    }
    #result {
      margin-top: 1rem;
      font-size: 18px;
    }
</style>
</head>
<body>
<h1>Sentiment Analysis</h1>
```

```
<p>Type your text below to analyze its sentiment:</p>
<textarea id="userText" placeholder="Enter your text here..."></textarea>
<br>
<button onclick="analyzeSentiment()">Analyze</button>
<div id="result"></div>

<script>
    async function analyzeSentiment() {
const text = document.getElementById('userText').value.trim();
constresultDiv = document.getElementById('result');

      if (!text) {
resultDiv.textContent = 'Please enter some text.';
        return;
      }

resultDiv.textContent = 'Analyzing...';

      try {
const response = await fetch('http://localhost:3000/analyze', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({ text })
        });
```

```
        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }

const data = await response.json();
resultDiv.innerHTML =
          `<strong>Sentiment:</strong> ${data.sentiment}<br><strong>Score:</strong> ${data.score}`;
      } catch (error) {
resultDiv.textContent = 'Error analyzing sentiment. Please try again later.';
console.error('Fetch error:', error);
      }
    }
</script>
</body>
</html>
```

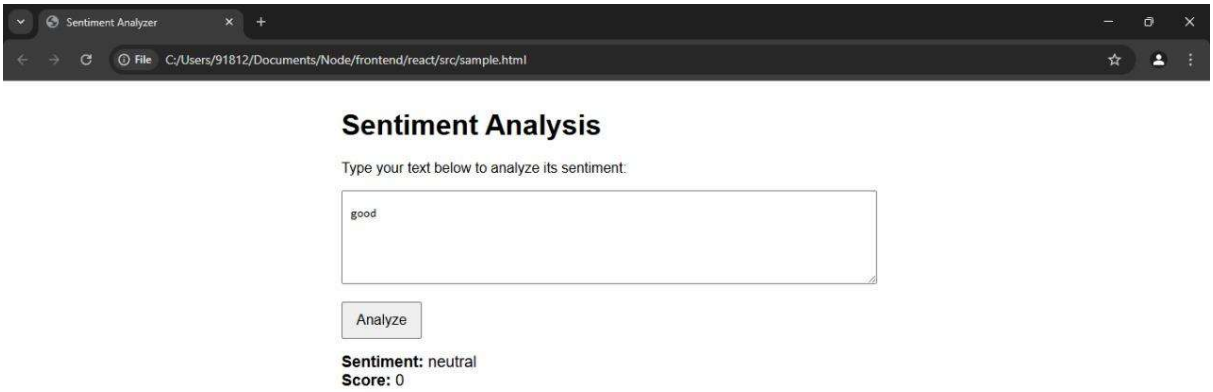
Back End Code :

```
const express = require('express');
constbodyParser = require('body-parser');
const Sentiment = require('sentiment');

const app = express();
const sentiment = new Sentiment();

app.use(bodyParser.json());
```

```
app.use((req, res, next) => {
res.header('Access-Control-Allow-Origin', '*');

res.header('Access-Control-Allow-Headers', 'Content-Type');
next();
});

app.post('/analyze', (req, res) => {
const{ text } = req.body;

  if (!text || typeoftext !== 'string') {
    return res.status(400).json({ error: 'Text is required.' });
  }
const result = sentiment.analyze(text);

  let label = 'neutral';
  if (result.score> 1) label = 'positive';
  else if (result.score< -1) label = 'negative';

res.json({
    sentiment: label,
    score: result.score
  });
});

const PORT = process.env.PORT || 3000;
```

```
app.listen(PORT, () => {
console.log(`✅ Local Sentiment API running at http://localhost:${PORT}`);
});
```

OUTPUT :



RESULT:

Thus the above program executed and the output is verified successfully.

ExNo:4 Date:	Build a Web Page that performs Linear Regression

AIM :

To implement a simple linear regression model using TensorFlow.js that predicts house prices based on size and visualizes both the training data and the regression line using Chart.js.

ALGORITHM :

Step 1 :Load Libraries: Add TensorFlow.js and Chart.js in HTML.

Step 2 :Define Data: Set house sizes and prices, convert to tensors

Step 3 :Build Model: Create and compile a simple sequential model.

Step 4 :Train Model: Fit the model, logging loss at intervals.

Step 5 :Predict Output: Use the model to predict new values.

Step 6 :Visualize: Plot actual and predicted data using Chart.js.

CODING :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Linear Regression with TensorFlow.js</title>
```

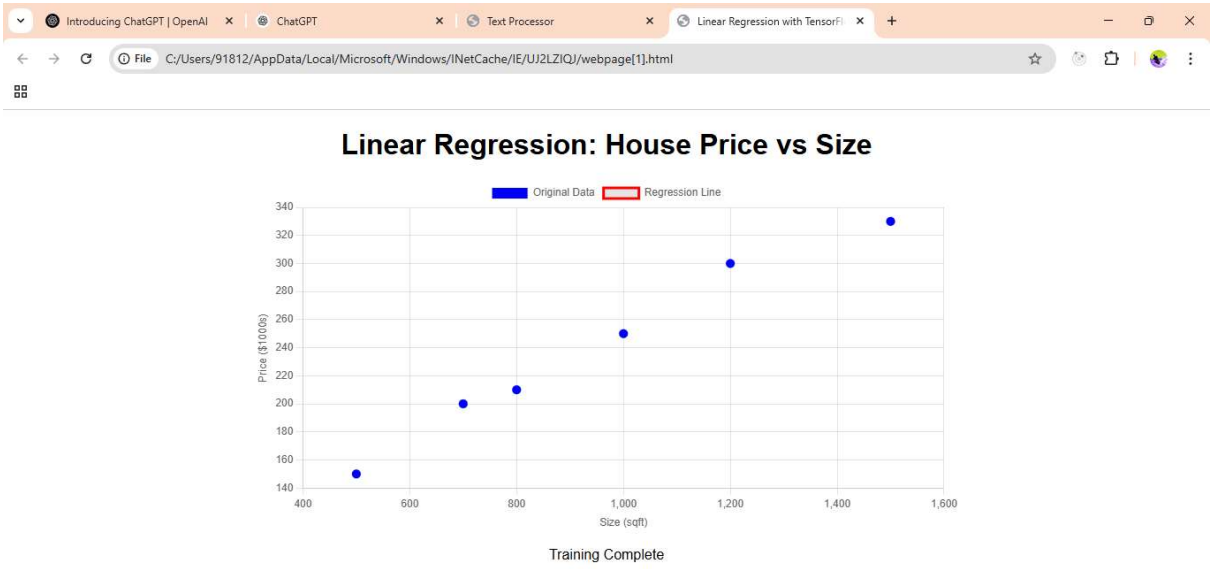
```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<style>
    body { font-family: Arial, sans-serif; text-align: center; }
    canvas { max-width: 800px; margin: auto; }
    h1 { margin-bottom: 20px; }
</style>
</head>
<body>
<h1>Linear Regression: House Price vs Size</h1>

<canvas id="chart" width="800" height="400"></canvas>
<p id="status">Training...</p>

<script>
    // Dataset: Size (sqft) vs. Price ($1000s)
const sizes = [500, 700, 800, 1000, 1200, 1500];
const prices = [150, 200, 210, 250, 300, 330];

    // Normalize data for training
constxs = tf.tensor1d(sizes);
constys = tf.tensor1d(prices);

    // Create model
const model = tf.sequential();
model.add(tf.layers.dense({ units: 1, inputShape: [1] }));
```

```
model.compile({ optimizer: 'sgd', loss: 'meanSquaredError' });


    // Train the model
    async function trainModel() {
      await model.fit(xs, ys, {
        epochs: 300,
        callbacks: {
onEpochEnd: (epoch, logs) => {
            if (epoch % 50 === 0) {
console.log(Epoch ${epoch}: loss = ${logs.loss});
            }
          }
        }
      });


document.getElementById("status").textContent = "Training Complete";
plot();
    }


    // Predict and plot


    async function plot() {
constchartData = {
        datasets: [
          {
            label: 'Original Data',
```

```
            data: sizes.map((size, i) =>({ x: size, y: prices[i] })),
backgroundColor: 'blue',
pointRadius: 5
          },
          {
            label: 'Regression Line',
            data: [],
            type: 'line',
borderColor: 'red',
            fill: false,
            tension: 0
          }
        ]
      };

      // Predict y for x values
constlineX = [400, 1600];
const preds = model.predict(tf.tensor1d(lineX));
constpredYs = await preds.array();

chartData.datasets[1].data = [
{ x: lineX[0], y: predYs[0] },
{ x: lineX[1], y: predYs[1] }
      ];

      // Draw chart
```

```
      new Chart(document.getElementById('chart'), {
        type: 'scatter',
        data: chartData,
        options: {


          scales: {
            x: { title: { display: true, text: 'Size (sqft)' } },
            y: { title: { display: true, text: 'Price ($1000s)' } }
          }
        }
      });
    }

trainModel();
</script>
</body>
</html>
```

OUTPUT :


Introducing ChatGPT | OpenAI
ChatGPT
Text Processor
Linear Regression with TensorFl
File
C:/Users/91812/AppData/Local/Microsoft/Windows/INetCache/IE/UJ2LZIQJ/webpage[1].html
Linear Regression: House Price vs Size
Original Data
Regression Line
Price ($1000s)
340
320
300
280
260
240
220
200
180
160
140
400
600
800
1,000
1,200
1,400
1,600
Size (sqft)
Training Complete

RESULT:

Thus the above program executed and the output is verified successfully.

ExNo:5 Date:	Adobe Sensei

AIM :

To explore and evaluate the use of Adobe Sensei's Content-Aware Fill feature in Adobe Photoshop for object removal and background extension, and to document the effectiveness of various Content-Aware Fill settings.

ALGORITHM :

Step 1: Open the image in Adobe Photoshop where object removal or background extension is needed.

Step 2: Select the target area using a selection tool (e.g., Lasso Tool or Object Selection Tool).

Step 3: Go to Edit > Content-Aware Fill to open the fill workspace.

Step 4: Adjust settings like Sampling Area, Color Adaptation, and Output to optimize the fill.

Step 5:Click OK to apply the fill, then refine the result if needed using the Healing Brush

CODING :

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Content-Aware Fill (Simulated)</title>
```

```
<style>
  body {
    font-family: Arial, sans-serif;
    text-align: center;
    background: #f0f0f0;
    margin: 0;
    padding: 20px;
  }
  h1 {
    color: #333;
  }
  #canvas {
    border: 2px solid #444;
    margin-top: 20px;
    cursor: crosshair;
    max-width: 100%;
  }
  #fillButton {
    margin-top: 20px;
```

```
    padding: 10px 20px;
    font-size: 16px;
    background-color: #0077cc;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
  }
  #fillButton:hover {
    background-color: #005fa3;
  }
 </style>
</head>
<body>

 <h1>Content-Aware Fill (Simulated with JavaScript)</h1>
 <input type="file" id="upload" accept="image/*" />
 <br />
```

```
<canvas id="canvas"></canvas>

<br />

<button id="fillButton">Fill Selected Area</button>



<script>

  const upload = document.getElementById('upload');

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

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

  const fillButton = document.getElementById('fillButton');



  let img = new Image();

  let startX, startY, endX, endY;

  let isDrawing = false;



  upload.addEventListener('change', (e) => {

    const file = e.target.files[0];

    if (!file) return;

    const reader = new FileReader();
```

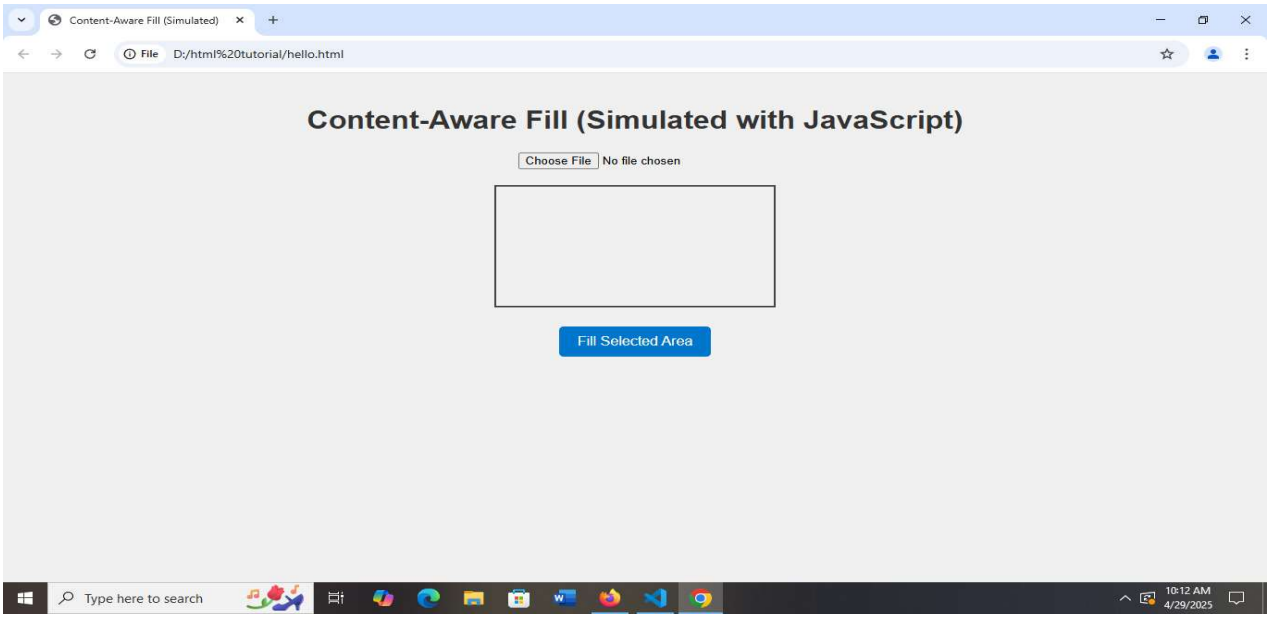
```
  reader.onload = function(event) {
    img.onload = () => {
      canvas.width = img.width;
      canvas.height = img.height;
      ctx.drawImage(img, 0, 0);
    };
    img.src = event.target.result;
  };
  reader.readAsDataURL(file);
});

canvas.addEventListener('mousedown', (e) => {
  const rect = canvas.getBoundingClientRect();
  startX = e.clientX - rect.left;
  startY = e.clientY - rect.top;
  isDrawing = true;
});
canvas.addEventListener('mouseup', (e) => {
```

```
  if (!isDrawing) return;

  const rect = canvas.getBoundingClientRect();

  endX = e.clientX - rect.left;

  endY = e.clientY - rect.top;

  isDrawing = false;



  // Draw selection rectangle

  ctx.strokeStyle = 'red';



  ctx.lineWidth = 2;

  ctx.strokeRect(startX, startY, endX - startX, endY - startY);

});



fillButton.addEventListener('click', () => {

  if (

    startX === undefined || startY === undefined ||

    endX === undefined || endY === undefined

  ) {
```

```
  alert("Please select an area first.");
  return;
}

const width = endX - startX;
const height = endY - startY;

if (width === 0 || height === 0) {
  alert("Invalid selection.");
  return;
}

// Get image data of selected area
const imageData = ctx.getImageData(startX, startY, width, height);
const data = imageData.data;

let r = 0, g = 0, b = 0;
const pixelCount = data.length / 4;
```

```
      for (let i = 0; i < data.length; i += 4) {
        r += data[i];
        g += data[i + 1];
        b += data[i + 2];
      }

      r = Math.floor(r / pixelCount);
      g = Math.floor(g / pixelCount);
      b = Math.floor(b / pixelCount);

      // Fill selected area with average color
      ctx.fillStyle = rgb(${r}, ${g}, ${b});
      ctx.fillRect(startX, startY, width, height);
    });
  </script>

</body>
</html>
```

OUTPUT:


Content-Aware Fill (Simulated)
File D:/html%20tutorial/hello.html
Content-Aware Fill (Simulated with JavaScript)
Choose File No file chosen
Fill Selected Area
Type here to search
10:12 AM
4/29/2025

RESULT:

Thus the above program executed and the output is verified successfully.