<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Text Summarizer</title>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/natural/6.8.0/natural.min.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/string-similarity/4.0.4/string-similarity.min.js"></script>

    <!-- PDF generation library -->

    <script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>

    <!-- DOCX generation library -->

    <script src="https://cdnjs.cloudflare.com/ajax/libs/docx/7.8.2/index.browser.js"></script>

    <!-- PDF.js for parsing PDF files -->

    <script src="https://cdnjs.cloudflare.com/ajax/libs/pdf.js/3.4.120/pdf.min.js"></script>

    <!-- Mammoth.js for parsing DOCX files -->

    <script src="https://cdnjs.cloudflare.com/ajax/libs/mammoth/1.6.0/mammoth.browser.min.js"></script>

    <style>

        :root {

            --primary-color: #3498db;

            --secondary-color: #2980b9;

            --background-color: #f5f7fa;

            --text-color: #333;

            --border-color: #ddd;

            --success-color: #27ae60;

            --success-hover: #219652;

            --warning-color: #e67e22;

            --warning-hover: #d35400;

            --info-color: #3498db;

            --info-hover: #2980b9;

        }

        body {

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

            line-height: 1.6;

            color: var(--text-color);

            background-color: var(--background-color);

            margin: 0;

            padding: 20px;

        }

        .container {

            max-width: 800px;

            margin: 0 auto;

            background-color: white;

            padding: 30px;

            border-radius: 10px;

            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

        }

        header {

            text-align: center;

            margin-bottom: 20px;

            padding-bottom: 20px;

            border-bottom: 1px solid var(--border-color);

        }

        h1 {

            color: var(--primary-color);

            margin: 0;

        }

        .input-section, .output-section {

            margin-bottom: 20px;

        }

        .radio-group {

            display: flex;

            gap: 15px;

            margin: 15px 0;

        }

        input[type="radio"] {

            margin-right: 5px;

        }

        textarea {

            width: 100%;

            min-height: 200px;

            padding: 10px;

            border: 1px solid var(--border-color);

            border-radius: 5px;

            resize: vertical;

            font-family: inherit;

            margin-top: 10px;

        }

        .file-input {

            margin-top: 15px;

            display: none;

        }

        .input-group {

            margin-bottom: 15px;

        }

        label {

            display: block;

            margin-bottom: 5px;

            font-weight: 600;

        }

        input[type="number"] {

            width: 60px;

            padding: 8px;

            border: 1px solid var(--border-color);

            border-radius: 5px;

        }

        button {

            background-color: var(--primary-color);

            color: white;

            border: none;

            padding: 10px 20px;

            border-radius: 5px;

            cursor: pointer;

            font-size: 16px;

            transition: background-color 0.3s;

        }

        button:hover {

            background-color: var(--secondary-color);

        }

        button:disabled {

            background-color: #cccccc;

            cursor: not-allowed;

        }

        .summary-container {

            margin-top: 20px;

            padding: 15px;

            border: 1px solid var(--border-color);

            border-radius: 5px;

            background-color: #f9f9f9;

        }

        .summary-header {

            font-weight: 600;

            margin-bottom: 10px;

            color: var(--primary-color);

        }

        .download-options {

            margin-top: 20px;

        }

        .download-options h3 {

            margin-bottom: 15px;

            color: var(--primary-color);

        }

        .download-buttons {

            display: flex;

            flex-wrap: wrap;

            gap: 10px;

        }

        .txt-btn {

            background-color: var(--success-color);

        }

        .txt-btn:hover {

            background-color: var(--success-hover);

        }

        .docx-btn {

            background-color: var(--info-color);

        }

        .docx-btn:hover {

            background-color: var(--info-hover);

        }

        .pdf-btn {

            background-color: var(--warning-color);

        }

        .pdf-btn:hover {

            background-color: var(--warning-hover);

        }

        .error {

            color: #e74c3c;

            font-weight: 500;

            margin-top: 10px;

        }

        .hidden {

            display: none;

        }

        .loader {

            display: inline-block;

            width: 20px;

            height: 20px;

            border: 3px solid rgba(255, 255, 255, 0.3);

            border-radius: 50%;

            border-top-color: white;

            animation: spin 1s ease-in-out infinite;

            margin-left: 10px;

            vertical-align: middle;

        }

        @keyframes spin {

            to { transform: rotate(360deg); }

        }

        footer {

            text-align: center;

            margin-top: 30px;

            font-size: 14px;

            color: #666;

        }

        .file-info {

            margin-top: 10px;

            font-size: 14px;

            color: #666;

        }

    </style>

</head>

<body>

    <div class="container">

        <header>

            <h1>Text Summarizer</h1>

        </header>

        <div class="input-section">

            <div class="input-group">

                <label>How would you like to input the text?</label>

                <div class="radio-group">

                    <div>

                        <input type="radio" id="manual-input" name="input-method" value="manual" checked>

                        <label for="manual-input">Enter text manually</label>

                    </div>

                    <div>

                        <input type="radio" id="file-input" name="input-method" value="file">

                        <label for="file-input">Load from a file</label>

                    </div>

                </div>

            </div>

            <div id="manual-input-container">

                <label for="input-text">Enter the text you want to summarize:</label>

                <textarea id="input-text" placeholder="Paste or type your text here..."></textarea>

            </div>

            <div id="file-input-container" class="file-input">

                <label for="text-file">Select a file (TXT, PDF, or DOCX):</label>

                <input type="file" id="text-file" accept=".txt,.pdf,.docx">

                <div id="file-info" class="file-info"></div>

                <div id="file-preview" class="hidden">

                    <label>File Preview:</label>

                    <textarea id="file-content" readonly></textarea>

                </div>

            </div>

            <div class="input-group">

                <label for="sentence-count">Number of sentences in summary:</label>

                <input type="number" id="sentence-count" min="1" value="3">

            </div>

            <button id="summarize-btn">Generate Summary</button>

            <span id="loader" class="loader hidden"></span>

            <div id="error-message" class="error hidden"></div>

        </div>

        <div id="output-section" class="output-section hidden">

            <div class="summary-container">

                <div class="summary-header">SUMMARY:</div>

                <div id="summary-output"></div>

            </div>

            <div class="download-options">

                <h3>Download Summary</h3>

                <div class="download-buttons">

                    <button id="txt-btn" class="txt-btn">Download as TXT</button>

                    <button id="docx-btn" class="docx-btn">Download as DOCX</button>

                    <button id="pdf-btn" class="pdf-btn">Download as PDF</button>

                </div>

            </div>

        </div>

        <footer>

            <p>22334007 - Project - 2025</p>

        </footer>

    </div>

    <script>

        // Set up PDF.js worker

        pdfjsLib.GlobalWorkerOptions.workerSrc = 'https://cdnjs.cloudflare.com/ajax/libs/pdf.js/3.4.120/pdf.worker.min.js';

        // Download NLTK-like stopwords

        const englishStopwords = [

            "i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours",

            "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself",

            "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which",

            "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be",

            "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an",

            "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for",

            "with", "about", "against", "between", "into", "through", "during", "before", "after", "above",

            "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again",

            "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any",

            "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only",

            "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "don't",

            "should", "should've", "now", "d", "ll", "m", "o", "re", "ve", "y", "ain", "aren", "aren't",

            "couldn", "couldn't", "didn", "didn't", "doesn", "doesn't", "hadn", "hadn't", "hasn", "hasn't",

            "haven", "haven't", "isn", "isn't", "ma", "mightn", "mightn't", "mustn", "mustn't", "needn",

            "needn't", "shan", "shan't", "shouldn", "shouldn't", "wasn", "wasn't", "weren", "weren't",

            "won", "won't", "wouldn", "wouldn't"

        ];

        document.addEventListener('DOMContentLoaded', function() {

            // DOM elements

            const inputMethodRadios = document.querySelectorAll('input[name="input-method"]');

            const manualInputContainer = document.getElementById('manual-input-container');

            const fileInputContainer = document.getElementById('file-input-container');

            const textInput = document.getElementById('input-text');

            const fileInput = document.getElementById('text-file');

            const fileInfo = document.getElementById('file-info');

            const filePreview = document.getElementById('file-preview');

            const fileContent = document.getElementById('file-content');

            const sentenceCount = document.getElementById('sentence-count');

            const summarizeBtn = document.getElementById('summarize-btn');

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

            const errorMessage = document.getElementById('error-message');

            const outputSection = document.getElementById('output-section');

            const summaryOutput = document.getElementById('summary-output');

            const txtBtn = document.getElementById('txt-btn');

            const docxBtn = document.getElementById('docx-btn');

            const pdfBtn = document.getElementById('pdf-btn');

            // Toggle between input methods

            inputMethodRadios.forEach(radio => {

                radio.addEventListener('change', function() {

                    if (this.value === 'manual') {

                        manualInputContainer.style.display = 'block';

                        fileInputContainer.style.display = 'none';

                    } else {

                        manualInputContainer.style.display = 'none';

                        fileInputContainer.style.display = 'block';

                    }

                });

            });

            // Handle file input

            fileInput.addEventListener('change', function(event) {

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

                filePreview.classList.add('hidden');

                fileInfo.textContent = '';

                if (file) {

                    fileInfo.textContent = `File selected: ${file.name} (${formatFileSize(file.size)})`;

                    // Show loading state

                    loader.classList.remove('hidden');

                    summarizeBtn.disabled = true;

                    const fileType = getFileExtension(file.name);

                    switch (fileType) {

                        case 'txt':

                            readTextFile(file);

                            break;

                        case 'pdf':

                            readPdfFile(file);

                            break;

                        case 'docx':

                            readDocxFile(file);

                            break;

                        default:

                            showError('Unsupported file type. Please upload a TXT, PDF, or DOCX file.');

                            loader.classList.add('hidden');

                            summarizeBtn.disabled = false;

                    }

                }

            });

            // Read TXT file

            function readTextFile(file) {

                const reader = new FileReader();

                reader.onload = function(e) {

                    fileContent.value = e.target.result;

                    filePreview.classList.remove('hidden');

                    loader.classList.add('hidden');

                    summarizeBtn.disabled = false;

                };

                reader.onerror = function() {

                    showError('Failed to read the text file');

                    loader.classList.add('hidden');

                    summarizeBtn.disabled = false;

                };

                reader.readAsText(file);

            }

            // Read PDF file

            function readPdfFile(file) {

                const fileReader = new FileReader();

                fileReader.onload = function() {

                    const typedArray = new Uint8Array(this.result);

                    // Load the PDF file

                    pdfjsLib.getDocument(typedArray).promise.then(function(pdf) {

                        let numPages = pdf.numPages;

                        let fullText = '';

                        let loadedPages = 0;

                        // Get text from all pages

                        for (let i = 1; i <= numPages; i++) {

                            pdf.getPage(i).then(function(page) {

                                page.getTextContent().then(function(textContent) {

                                    const textItems = textContent.items;

                                    let pageText = '';

                                    // Extract text items

                                    for (let j = 0; j < textItems.length; j++) {

                                        pageText += textItems[j].str + ' ';

                                    }

                                    fullText += pageText + '\n\n';

                                    loadedPages++;

                                    // If all pages are loaded, display the text

                                    if (loadedPages === numPages) {

                                        fileContent.value = fullText;

                                        filePreview.classList.remove('hidden');

                                        loader.classList.add('hidden');

                                        summarizeBtn.disabled = false;

                                    }

                                });

                            }).catch(function(error) {

                                showError('Error reading PDF page: ' + error.message);

                                loader.classList.add('hidden');

                                summarizeBtn.disabled = false;

                            });

                        }

                    }).catch(function(error) {

                        showError('Error loading PDF: ' + error.message);

                        loader.classList.add('hidden');

                        summarizeBtn.disabled = false;

                    });

                };

                fileReader.onerror = function() {

                    showError('Failed to read the PDF file');

                    loader.classList.add('hidden');

                    summarizeBtn.disabled = false;

                };

                fileReader.readAsArrayBuffer(file);

            }

            // Read DOCX file

            function readDocxFile(file) {

                mammoth.extractRawText({ arrayBuffer: file })

                    .then(function(result) {

                        fileContent.value = result.value;

                        filePreview.classList.remove('hidden');

                        loader.classList.add('hidden');

                        summarizeBtn.disabled = false;

                        if (result.messages.length > 0) {

                            console.log("Mammoth warnings:", result.messages);

                        }

                    })

                    .catch(function(error) {

                        showError('Error reading DOCX file: ' + error.message);

                        loader.classList.add('hidden');

                        summarizeBtn.disabled = false;

                    });

            }

            // Handle summarize button click

            summarizeBtn.addEventListener('click', function() {

                // Clear previous results

                errorMessage.classList.add('hidden');

                outputSection.classList.add('hidden');

                // Get text to summarize

                let text;

                const selectedMethod = document.querySelector('input[name="input-method"]:checked').value;

                if (selectedMethod === 'manual') {

                    text = textInput.value.trim();

                } else {

                    text = fileContent.value.trim();

                }

                // Validate input

                if (!text) {

                    showError('Please enter or upload some text to summarize');

                    return;

                }

                // Get number of sentences

                const numSentences = parseInt(sentenceCount.value);

                if (isNaN(numSentences) || numSentences <= 0) {

                    showError('Number of sentences must be a positive number');

                    return;

                }

                // Show loader

                loader.classList.remove('hidden');

                summarizeBtn.disabled = true;

                // Use setTimeout to make the UI more responsive

                setTimeout(() => {

                    try {

                        const summary = summarizeText(text, numSentences);

                        summaryOutput.textContent = summary;

                        outputSection.classList.remove('hidden');

                    } catch (err) {

                        showError('Error generating summary: ' + err.message);

                    } finally {

                        loader.classList.add('hidden');

                        summarizeBtn.disabled = false;

                    }

                }, 100);

            });

            // Handle download as TXT

            txtBtn.addEventListener('click', function() {

                const summary = summaryOutput.textContent;

                downloadTxt(summary, 'summary.txt');

            });

            // Handle download as DOCX

            docxBtn.addEventListener('click', function() {

                const summary = summaryOutput.textContent;

                downloadDocx(summary, 'summary.docx');

            });

            // Handle download as PDF

            pdfBtn.addEventListener('click', function() {

                const summary = summaryOutput.textContent;

                downloadPdf(summary, 'summary.pdf');

            });

            // Helper function to show error message

            function showError(message) {

                errorMessage.textContent = message;

                errorMessage.classList.remove('hidden');

            }

            // Helper function to get file extension

            function getFileExtension(filename) {

                return filename.split('.').pop().toLowerCase();

            }

            // Helper function to format file size

            function formatFileSize(bytes) {

                if (bytes < 1024) return bytes + ' bytes';

                else if (bytes < 1048576) return (bytes / 1024).toFixed(1) + ' KB';

                else return (bytes / 1048576).toFixed(1) + ' MB';

            }

            // Download as plain text

            function downloadTxt(text, filename) {

                const blob = new Blob([text], { type: 'text/plain' });

                const url = URL.createObjectURL(blob);

                const a = document.createElement('a');

                a.href = url;

                a.download = filename;

                document.body.appendChild(a);

                a.click();

                document.body.removeChild(a);

                URL.revokeObjectURL(url);

            }

            // Download as DOCX

            function downloadDocx(text, filename) {

                try {

                    const { Document, Packer, Paragraph, TextRun, HeadingLevel } = docx;

                    // Create document

                    const doc = new Document({

                        sections: [{

                            properties: {},

                            children: [

                                new Paragraph({

                                    text: "Text Summary",

                                    heading: HeadingLevel.HEADING\_1,

                                }),

                                new Paragraph({

                                    text: "Generated on " + new Date().toLocaleDateString(),

                                    spacing: {

                                        after: 200,

                                    },

                                }),

                                new Paragraph({

                                    text: text,

                                }),

                            ],

                        }],

                    });

                    // Generate and download document

                    Packer.toBlob(doc).then(blob => {

                        const url = URL.createObjectURL(blob);

                        const a = document.createElement('a');

                        a.href = url;

                        a.download = filename;

                        document.body.appendChild(a);

                        a.click();

                        document.body.removeChild(a);

                        URL.revokeObjectURL(url);

                    });

                } catch (err) {

                    showError('Error generating DOCX: ' + err.message);

                }

            }

            // Download as PDF

            function downloadPdf(text, filename) {

                try {

                    const { jsPDF } = window.jspdf;

                    // Create PDF document

                    const doc = new jsPDF({

                        orientation: 'portrait',

                        unit: 'mm',

                        format: 'a4'

                    });

                    // Add content

                    doc.setFontSize(24);

                    doc.text('Text Summary', 20, 20);

                    doc.setFontSize(12);

                    doc.text('Generated on ' + new Date().toLocaleDateString(), 20, 30);

                    doc.setFontSize(12);

                    // Split text to handle line breaks and page overflow

                    const textLines = doc.splitTextToSize(text, 170);

                    doc.text(textLines, 20, 40);

                    // Save PDF

                    doc.save(filename);

                } catch (err) {

                    showError('Error generating PDF: ' + err.message);

                }

            }

            // Text summarization function

            function summarizeText(text, numSentences) {

                // Preprocess text

                text = preprocessText(text);

                // Tokenize sentences

                const sentences = sentenceTokenize(text);

                // If the text is too short, return it as is

                if (sentences.length <= numSentences) {

                    return text;

                }

                // Tokenize words and remove stopwords

                const words = wordTokenize(text.toLowerCase());

                const filteredWords = words.filter(word => {

                    return !englishStopwords.includes(word) &&

                           !isPunctuation(word) &&

                           !isNumeric(word);

                });

                // Calculate word frequencies

                const wordFreq = calculateWordFrequencies(filteredWords);

                // Score sentences

                const sentenceScores = {};

                sentences.forEach((sentence, i) => {

                    const sentenceWords = wordTokenize(sentence.toLowerCase());

                    // Basic scoring based on word frequency

                    const wordScore = sentenceWords.reduce((score, word) => {

                        return score + (wordFreq[word] || 0);

                    }, 0);

                    // Position bias - earlier sentences often contain important information

                    const positionScore = 1.0 / (i + 1);

                    // Length normalization - avoid bias towards longer sentences

                    const lengthFactor = Math.min(1.0, 10.0 / Math.max(1, sentenceWords.length));

                    // Calculate final score

                    const finalScore = wordScore \* positionScore \* lengthFactor;

                    sentenceScores[sentence] = finalScore;

                });

                // Select top sentences, maintaining original order

                const topSentences = Object.keys(sentenceScores)

                    .sort((a, b) => sentenceScores[b] - sentenceScores[a])

                    .slice(0, numSentences);

                const summaryArray = [];

                sentences.forEach(sentence => {

                    if (topSentences.includes(sentence)) {

                        summaryArray.push(sentence);

                    }

                });

                return summaryArray.join(' ');

            }

            // Helper functions for text processing

            function preprocessText(text) {

                // Remove excessive whitespace

                text = text.replace(/\s+/g, ' ');

                // Replace multiple newlines with single newlines

                text = text.replace(/\n+/g, '\n');

                return text;

            }

            function sentenceTokenize(text) {

                // Simple sentence tokenization (not as sophisticated as NLTK)

                // Split by period, exclamation point, or question mark followed by space or newline

                const sentences = text.split(/[.!?][\s\n]+/);

                // Remove empty sentences

                return sentences.filter(s => s.trim().length > 0);

            }

            function wordTokenize(text) {

                // Simple word tokenization (not as sophisticated as NLTK)

                // Split by whitespace and remove empty strings

                return text.split(/\s+/).filter(w => w.length > 0);

            }

            function isPunctuation(word) {

                return /^[^\w\s]$/.test(word);

            }

            function isNumeric(word) {

                return /^\d+$/.test(word);

            }

            function calculateWordFrequencies(words) {

                const freq = {};

                words.forEach(word => {

                    freq[word] = (freq[word] || 0) + 1;

                });

                return freq;

            }

        });

    </script>

</body>

</html>