```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gerador de Imagens</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <style>
    /* Custom font for better appearance */
    body {
      font-family: 'Inter', sans-serif;
      background-color: #f0f4f8; /* Light blue-gray background */
    }
    /* Custom scrollbar for textareas */
    textarea::-webkit-scrollbar {
      width: 8px;
    }
    textarea::-webkit-scrollbar-track {
      background: #e2e8f0;
      border-radius: 10px;
    }
    textarea::-webkit-scrollbar-thumb {
      background: #94a3b8;
      border-radius: 10px;
    }
    textarea::-webkit-scrollbar-thumb:hover {
      background: #64748b;
    }
```

```
    /* Star rating styles */
    .star-rating .star {
      font-size: 2rem; /* Larger stars */
      color: #cbd5e1; /* Grey for unselected stars */
      cursor: pointer;
      transition: color 0.2s ease-in-out, transform 0.1s ease-in-out;
    }
    .star-rating .star.selected,
    .star-rating .star:hover,
    .star-rating .star.hovered {
      color: #fbbf24; /* Amber for selected/hovered stars */
      transform: scale(1.1); /* Slightly larger on hover/select */
    }
    .star-rating .star:active {
      transform: scale(0.95); /* Smaller on click */
    }
  </style>
</head>
<body class="flex items-center justify-center min-h-screen p-4">
  <div class="bg-white p-8 rounded-xl shadow-2xl w-full max-w-2xl transform
transition-all duration-300 hover:scale-[1.01]">
    <h1 class="text-4xl font-extrabold text-center text-gray-900 mb-8 tracking-
tight">
      □ Gerador de Imagens com IA □□
    </h1>

    <div class="mb-6">
      <label for="promptInput" class="block text-lg font-semibold text-gray-700
```

```
mb-2">
            Descreva a imagem que deseja gerar:
        </label>
        <textarea id="promptInput"
                class="w-full p-4 border border-gray-300 rounded-lg focus:ring-4
focus:ring-blue-300 focus:border-blue-500 transition-all duration-200 text-gray-
800 text-base resize-y min-h-[100px] shadow-sm"
                placeholder="Ex: Uma paisagem futurista com montanhas
flutuantes e um céu roxo."></textarea>
    </div>

    <div class="flex flex-col sm:flex-row gap-4 mb-6">
        <button id="generateImageButton"
            class="flex-1 bg-gradient-to-r from-purple-500 to-pink-600 text-white
font-bold py-3 px-6 rounded-lg shadow-lg hover:from-purple-600 hover:to-pink-
700 transition-all duration-300 transform hover:-translate-y-1 active:scale-95
focus:outline-none focus:ring-4 focus:ring-purple-300">
            Gerar Imagem
        </button>
        <button id="downloadImageButton"
            class="flex-1 bg-gradient-to-r from-green-500 to-teal-600 text-white
font-bold py-3 px-6 rounded-lg shadow-lg hover:from-green-600 hover:to-teal-
700 transition-all duration-300 transform hover:-translate-y-1 active:scale-95
focus:outline-none focus:ring-4 focus:ring-green-300"
            disabled>
            Descarregar Imagem
        </button>
    </div>
```

```
    <div id="loadingIndicator" class="hidden text-center text-blue-600 font-semibold mb-4">
        Gerando imagem... Por favor, aguarde. <span class="animate-pulse">⌧</span>
    </div>

    <div id="imageOutput" class="hidden mb-6 text-center">
        <label class="block text-lg font-semibold text-gray-700 mb-2">Sua imagem gerada:</label>
        <img id="generatedImage" src="" alt="Imagem Gerada" class="max-w-full h-auto rounded-lg shadow-md border border-gray-200 mx-auto"/>
    </div>

    <div class="mt-8 pt-6 border-t border-gray-200">
        <h2 class="text-2xl font-bold text-center text-gray-800 mb-6">Deixe o seu Feedback! ⌧</h2>

        <div class="mb-4">
            <label class="block text-lg font-semibold text-gray-700 mb-2">Avaliação:</label>
            <div id="starRating" class="star-rating flex justify-center space-x-2">
                <span class="star" data-value="1">★</span>
                <span class="star" data-value="2">★</span>
                <span class="star" data-value="3">★</span>
                <span class="star" data-value="4">★</span>
                <span class="star" data-value="5">★</span>
            </div>
```

```
        </div>

        <div class="mb-6">
            <label for="feedbackText" class="block text-lg font-semibold text-gray-700 mb-2">Seu Comentário (opcional):</label>
            <textarea id="feedbackText"
                      class="w-full p-4 border border-gray-300 rounded-lg focus:ring-4 focus:ring-blue-300 focus:border-blue-500 transition-all duration-200 text-gray-800 text-base resize-y min-h-[80px] shadow-sm"
                      placeholder="Ex: Adorei a funcionalidade de download!"></textarea>
        </div>

        <button id="sendFeedbackButton"
                class="w-full bg-gradient-to-r from-blue-500 to-cyan-600 text-white font-bold py-3 px-6 rounded-lg shadow-lg hover:from-blue-600 hover:to-cyan-700 transition-all duration-300 transform hover:-translate-y-1 active:scale-95 focus:outline-none focus:ring-4 focus:ring-blue-300">
            Enviar Feedback
        </button>
    </div>
    <div id="messageBox" class="hidden bg-blue-100 border border-blue-400 text-blue-700 px-4 py-3 rounded-lg relative mb-4 mt-6" role="alert">
        <span id="messageText" class="block sm:inline"></span>
        <span class="absolute top-0 bottom-0 right-0 px-4 py-3 cursor-pointer" onclick="document.getElementById('messageBox').classList.add('hidden');">
            <svg class="fill-current h-6 w-6 text-blue-500" role="button" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20
```

```
20"><title>Fechar</title><path d="M14.348 14.849a1.2 1.2 0 0 1-1.697 0L10 11.819l-2.651 3.029a1.2 1.2 0 1 1-1.697-1.697l2.758-3.15L6.22 7.151a1.2 1.2 0 1 1 1.697-1.697L10 8.183l2.651-3.029a1.2 1.2 0 1 1 1.697 1.697l-2.758 3.15 2.758 3.15a1.2 1.2 0 0 1 0 1.698z"/></svg>
        </span>
      </div>

      <p class="text-sm text-gray-600 text-center mt-6">
        Este gerador utiliza inteligência artificial para criar imagens com base na sua descrição.
      </p>
    </div>

    <script>
      document.addEventListener('DOMContentLoaded', function() {
        const promptInput = document.getElementById('promptInput');
        const generateImageButton = document.getElementById('generateImageButton');
        const downloadImageButton = document.getElementById('downloadImageButton');
        const loadingIndicator = document.getElementById('loadingIndicator');
        const imageOutput = document.getElementById('imageOutput');
        const generatedImage = document.getElementById('generatedImage');
        const messageBox = document.getElementById('messageBox');
        const messageText = document.getElementById('messageText');

        // Feedback elements
        const starRatingContainer = document.getElementById('starRating');
```

```
        const stars = starRatingContainer.querySelectorAll('.star');
        const feedbackText = document.getElementById('feedbackText');
        const sendFeedbackButton =
document.getElementById('sendFeedbackButton');

        let selectedRating = 0; // Stores the user's selected star rating

        // Function to display messages to the user
        function showMessage(message, type = 'info') {
            messageText.textContent = message;
            messageBox.classList.remove('hidden');
            // Remove previous type classes and add the new one
            messageBox.classList.remove('bg-red-100', 'border-red-400', 'text-red-
700', 'bg-green-100', 'border-green-400', 'text-green-700', 'bg-blue-100', 'border-
blue-400', 'text-blue-700');
            if (type === 'error') {
                messageBox.classList.add('bg-red-100', 'border-red-400', 'text-red-
700');
            } else if (type === 'success') {
                messageBox.classList.add('bg-green-100', 'border-green-400', 'text-
green-700');
            } else { // default info
                messageBox.classList.add('bg-blue-100', 'border-blue-400', 'text-blue-
700');
            }
            setTimeout(() => {
                messageBox.classList.add('hidden');
            }, 5000); // Hide message after 5 seconds
```

```
        }

        // Function to generate the image using the Gemini API
        async function generateImage() {
            const prompt = promptInput.value.trim();

            if (!prompt) {
                showMessage('Por favor, digite uma descrição para gerar a imagem.',
'error');
                return;
            }

            // Show loading indicator
            loadingIndicator.classList.remove('hidden');
            imageOutput.classList.add('hidden');
            generatedImage.src = ''; // Clear previous image
            downloadImageButton.disabled = true; // Disable download button
while generating

            try {
                const payload = { instances: { prompt: prompt }, parameters: {
"sampleCount": 1} };
                const apiKey = ""; // If you want to use models other than imagen-3.0-
generate-002, provide an API key here. Otherwise, leave this as-is.
                const apiUrl =
`https://generativelanguage.googleapis.com/v1beta/models/imagen-3.0-
generate-002:predict?key=${apiKey}`;
```

```
            const response = await fetch(apiUrl, {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(payload)
            });

            const result = await response.json();

            if (result.predictions && result.predictions.length > 0 &&
result.predictions[0].bytesBase64Encoded) {
                const imageUrl =
`${result.predictions[0].bytesBase64Encoded}`;
                generatedImage.src = imageUrl;
                imageOutput.classList.remove('hidden');
                downloadImageButton.disabled = false; // Enable download button
                showMessage('Imagem gerada com sucesso!', 'success');
            } else {
                console.error('Estrutura de resposta inesperada ou conteúdo de
imagem ausente:', result);
                showMessage('Erro ao gerar imagem. A resposta da API não contém
dados de imagem válidos.', 'error');
            }
        } catch (error) {
            console.error('Erro ao chamar a API de geração de imagem:', error);
            showMessage('Ocorreu um erro ao gerar a imagem. Por favor, tente
novamente.', 'error');
        } finally {
            loadingIndicator.classList.add('hidden');
```

```
    }
}

// Function to download the generated image
function downloadImage() {
    const imageUrl = generatedImage.src;
    if (imageUrl && imageUrl !== '') {
        const a = document.createElement('a');
        a.href = imageUrl;
        a.download = 'imagem_gerada.png'; // Default filename
        document.body.appendChild(a);
        a.click();
        document.body.removeChild(a);
        showMessage('Download da imagem iniciado!', 'info');
    } else {
        showMessage('Nenhuma imagem para descarregar.', 'error');
    }
}

// --- Star Rating Logic ---
function updateStars(rating) {
    stars.forEach(star => {
        if (parseInt(star.dataset.value) <= rating) {
            star.classList.add('selected');
        } else {
            star.classList.remove('selected');
        }
    });
```

```
}

starRatingContainer.addEventListener('mouseover', (e) => {
    const hoveredStar = e.target.closest('.star');
    if (hoveredStar) {
        const hoverValue = parseInt(hoveredStar.dataset.value);
        stars.forEach(star => {
            if (parseInt(star.dataset.value) <= hoverValue) {
                star.classList.add('hovered');
            } else {
                star.classList.remove('hovered');
            }
        });
    }
});

starRatingContainer.addEventListener('mouseout', () => {
    stars.forEach(star => {
        star.classList.remove('hovered');
    });
    updateStars(selectedRating); // Revert to selected rating on mouse out
});

starRatingContainer.addEventListener('click', (e) => {
    const clickedStar = e.target.closest('.star');
    if (clickedStar) {
        selectedRating = parseInt(clickedStar.dataset.value);
        updateStars(selectedRating);
```

```
            showMessage(`Você avaliou com ${selectedRating} estrelas!`, 'info');
        }
    });
    // --- End Star Rating Logic ---

    // Function to send feedback via mailto link
    function sendFeedback() {
        const rating = selectedRating > 0 ? `${selectedRating} estrelas` : 'Não avaliado';
        const comment = feedbackText.value.trim();
        const subject = encodeURIComponent(`Feedback sobre o Gerador de Imagens - Avaliação: ${rating}`);
        const body = encodeURIComponent(`Avaliação: ${rating}\n\nComentário:\n${comment || 'Nenhum comentário fornecido.'}\n\n---`);
        const mailtoLink = `mailto:bwedits0@gmail.com?subject=${subject}&body=${body}`;

        if (selectedRating === 0 && comment === '') {
            showMessage('Por favor, selecione uma avaliação ou digite um comentário para enviar o feedback.', 'error');
            return;
        }

        window.open(mailtoLink, '_blank');
        showMessage('O seu cliente de e-mail será aberto com o feedback pré-preenchido. Por favor, clique em "Enviar" lá.', 'info');
```

```
            // Optionally clear feedback fields after opening mailto
            selectedRating = 0;
            updateStars(0);
            feedbackText.value = '';
        }

        // Event Listeners for the buttons
        generateImageButton.addEventListener('click', generateImage);
        downloadImageButton.addEventListener('click', downloadImage);
        sendFeedbackButton.addEventListener('click', sendFeedback);
    });
  </script>
</body>
</html>
```