```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Universal File Converter</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap"
rel="stylesheet">
    <style>
        body {
            font-family: 'Inter', sans-serif;
            background-color: #000000;
            color: #ffffff;
        }
        .converter-container {
            background-color: #0a0a0a;
            border: 1px solid #1a1a1a;
            box-shadow: 0 0 50px rgba(0, 255, 255, 0.1);
        }
        .step-title {
            color: #00ffff;
        }
        .format-btn {
            background-color: #1a1a1a;
            border: 1px solid #333333;
            color: #ffffff;
            transition: all 0.3s ease;
            position: relative;
            overflow: hidden;
        }
        .format-btn:before {
            content: '';
            position: absolute;
            top: 50%;
            left: 50%;
            width: 0;
            height: 0;
            background: rgba(0, 255, 255, 0.1);
            border-radius: 50%;
            transform: translate(-50%, -50%);
            transition: width 0.4s ease, height 0.4s ease;
        }
```

```
.format-btn:hover {
    border-color: #00ffff;
    color: #00ffff;
}
.format-btn:hover:before {
    width: 200%;
    height: 200%;
}
.format-btn.selected {
    background-color: #00ffff;
    color: #000000;
    border-color: #00ffff;
    font-weight: 700;
}
.format-btn.disabled {
    opacity: 0.3;
    cursor: not-allowed;
    border-color: #333333;
    color: #888888;
}
 .format-btn.disabled:hover {
    border-color: #333333;
    color: #888888;
}
.format-btn.disabled:hover:before {
    width: 0;
    height: 0;
}
.input-area {
    background-color: #1a1a1a;
    border: 1px dashed #333333;
}
.action-btn {
    background-color: #00ffff;
    color: #000000;
    transition: all 0.3s ease;
    box-shadow: 0 0 20px rgba(0, 255, 255, 0.3);
}
.action-btn:hover {
    transform: scale(1.05);
    box-shadow: 0 0 30px rgba(0, 255, 255, 0.6);
}
.progress-bar-container {
    background-color: #1a1a1a;
```

```
            border-radius: 9999px;
            overflow: hidden;
        }
        .progress-bar {
            background-color: #00ffff;
            transition: width 0.5s ease-in-out;
        }
        .download-btn {
            background-color: #1a1a1a;
            border: 1px solid #00ffff;
            color: #00ffff;
        }
         .download-btn:hover {
            background-color: #00ffff;
            color: #000000;
        }

        /* Custom scrollbar for better aesthetics */
        ::-webkit-scrollbar {
            width: 8px;
        }
        ::-webkit-scrollbar-track {
            background: #0a0a0a;
        }
        ::-webkit-scrollbar-thumb {
            background: #00ffff;
            border-radius: 4px;
        }
        ::-webkit-scrollbar-thumb:hover {
            background: #00cccc;
        }
    </style>
</head>
<body class="flex items-center justify-center min-h-screen p-4">

    <div class="converter-container w-full max-w-2xl mx-auto p-6 sm:p-8 rounded-2xl">

        <!-- Header -->
        <div class="text-center mb-8">
            <h1 class="text-3xl sm:text-4xl font-bold tracking-tight">Universal File Converter</h1>
            <p class="text-gray-400 mt-2">Simple, sleek, and fast conversions.</p>
        </div>

        <div class="space-y-8">
```

```
        <!-- Step 1: Choose Input Format -->
        <div>
            <h2 class="step-title text-lg font-semibold mb-4 flex items-center">
                <span class="bg-teal-500 text-black rounded-full h-6 w-6 flex items-center
justify-center font-bold mr-3">1</span> From
            </h2>
            <div id="from-formats" class="grid grid-cols-3 sm:grid-cols-4 md:grid-cols-5 gap-3">
                <!-- Buttons will be injected by JS -->
            </div>
        </div>

        <!-- Step 2: Choose Output Format -->
        <div id="step-2" class="hidden">
            <h2 class="step-title text-lg font-semibold mb-4 flex items-center">
                <span class="bg-teal-500 text-black rounded-full h-6 w-6 flex items-center
justify-center font-bold mr-3">2</span> To
            </h2>
            <div id="to-formats" class="grid grid-cols-3 sm:grid-cols-4 md:grid-cols-5 gap-3">
                <!-- Buttons will be injected by JS -->
            </div>
        </div>

        <!-- Step 3: Upload and Convert -->
        <div id="step-3" class="hidden">
            <h2 class="step-title text-lg font-semibold mb-4 flex items-center">
                 <span class="bg-teal-500 text-black rounded-full h-6 w-6 flex items-center
justify-center font-bold mr-3">3</span> Convert
            </h2>
            <div id="input-container" class="input-area p-6 rounded-lg text-center">
                <!-- Input field will be injected here -->
            </div>
            <button id="convert-btn" class="action-btn w-full font-bold py-3 px-4 rounded-lg mt-4
text-lg">Convert Now</button>
        </div>

        <!-- Progress and Download Area -->
        <div id="progress-container" class="hidden text-center">
             <p id="progress-text" class="mb-2 text-gray-300">Converting...</p>
             <div class="progress-bar-container w-full h-3">
                <div id="progress-bar" class="progress-bar h-full" style="width: 0%;"></div>
            </div>
        </div>
        <div id="result-container" class="hidden text-center">
            <p class="text-2xl font-bold text-teal-400 mb-4">Conversion Complete!</p>
```

```
            <button id="download-btn" class="download-btn w-full sm:w-auto font-bold py-3 px-8
rounded-lg text-lg inline-flex items-center justify-center gap-2">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24
24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round"
stroke-linejoin="round"><path d="M21 15v4a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2v-4"/><polyline
points="7 10 12 15 17 10"/><line x1="12" y1="15" x2="12" y2="3"/></svg>
                Download File
            </button>
             <button id="start-over-btn" class="w-full sm:w-auto font-bold py-3 px-8 rounded-lg
mt-4 sm:mt-0 sm:ml-4">Start New Conversion</button>
        </div>

    </div>
</div>

<script>
    const conversionMap = {
        'youtube': ['wav', 'mp3', 'aiff', 'mp4', 'flac'],
        'mp3': ['flac', 'wav'],
        'wav': ['mp3', 'flac', 'ogg', 'aiff'],
        'flac': ['mp3', 'wav', 'ogg', 'aiff'],
        'rar': ['iso', 'zip'],
        'iso': ['rar', 'zip'],
        'png': ['jpg'],
        'jpg': ['png'],
        'mp4': ['mov'],
        'mov': ['mp4']
    };

    const allFormats = [...new
Set(Object.keys(conversionMap).concat(...Object.values(conversionMap)))];

    let selectedFrom = null;
    let selectedTo = null;

    const fromFormatsContainer = document.getElementById('from-formats');
    const toFormatsContainer = document.getElementById('to-formats');
    const step2 = document.getElementById('step-2');
    const step3 = document.getElementById('step-3');
    const inputContainer = document.getElementById('input-container');
    const convertBtn = document.getElementById('convert-btn');
    const progressContainer = document.getElementById('progress-container');
    const progressBar = document.getElementById('progress-bar');
    const progressText = document.getElementById('progress-text');
```

```
const resultContainer = document.getElementById('result-container');
const downloadBtn = document.getElementById('download-btn');
const startOverBtn = document.getElementById('start-over-btn');

function createFormatButton(format, type) {
    const btn = document.createElement('button');
    btn.textContent = format.toUpperCase();
    btn.dataset.format = format;
    btn.className = 'format-btn py-3 px-2 rounded-lg text-sm sm:text-base font-medium';
    btn.onclick = () => handleFormatSelection(format, type);
    return btn;
}

function handleFormatSelection(format, type) {
    if (type === 'from') {
        selectedFrom = format;
        selectedTo = null; // Reset "to" selection

        // Update "from" buttons UI
        document.querySelectorAll('#from-formats .format-btn').forEach(btn => {
            btn.classList.toggle('selected', btn.dataset.format === format);
        });

        // Populate "to" buttons
        populateToFormats(format);
        step2.classList.remove('hidden');
        step3.classList.add('hidden');
        resultContainer.classList.add('hidden');
        progressContainer.classList.add('hidden');

    } else if (type === 'to') {
        if (!selectedFrom) return;
        selectedTo = format;

        // Update "to" buttons UI
        document.querySelectorAll('#to-formats .format-btn').forEach(btn => {
            btn.classList.toggle('selected', btn.dataset.format === format);
        });

        prepareInputArea();
        step3.classList.remove('hidden');
    }
}
```

```
        function populateFromFormats() {
            fromFormatsContainer.innerHTML = '';
            Object.keys(conversionMap).forEach(format => {
                fromFormatsContainer.appendChild(createFormatButton(format, 'from'));
            });
        }

        function populateToFormats(fromFormat) {
            toFormatsContainer.innerHTML = '';
            const possibleConversions = conversionMap[fromFormat] || [];

            allFormats.forEach(format => {
                const btn = createFormatButton(format, 'to');
                if (!possibleConversions.includes(format)) {
                    btn.classList.add('disabled');
                    btn.disabled = true;
                }
                toFormatsContainer.appendChild(btn);
            });
        }

        function prepareInputArea() {
            inputContainer.innerHTML = '';
            if (selectedFrom === 'youtube') {
                const input = document.createElement('input');
                input.type = 'text';
                input.placeholder = 'Paste YouTube URL here...';
                input.className = 'w-full bg-gray-900 border border-gray-600 rounded-md p-3
text-white focus:ring-2 focus:ring-teal-500 focus:border-teal-500 outline-none';
                inputContainer.appendChild(input);
            } else {
                const label = document.createElement('label');
                label.htmlFor = 'file-upload';
                label.className = 'cursor-pointer text-teal-400 font-semibold';
                label.innerHTML = `<svg xmlns="http://www.w3.org/2000/svg" width="48" height="48"
viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="1.5" stroke-linecap="round"
stroke-linejoin="round" class="mx-auto mb-2 text-gray-500"><path d="M21 12v7a2 2 0 0 1-2
2H5a2 2 0 0 1-2-2V5a2 2 0 0 1 2-2h7"/><line x1="16" x2="22" y1="5" y2="5"/><line x1="19"
x2="19" y1="2" y2="8"/><path d="M10 14l2-2 3 3 5-5"/></svg>
                <p>Drop your .${selectedFrom.toUpperCase()} file here or <span
class="underline">browse</span></p>`;

                const input = document.createElement('input');
                input.type = 'file';
```

```
            input.id = 'file-upload';
            input.className = 'hidden';
            input.accept = `.${selectedFrom}`;

            inputContainer.appendChild(label);
            inputContainer.appendChild(input);
        }
    }

    function startConversion() {
        // Hide previous steps and show progress
        step2.classList.add('hidden');
        step3.classList.add('hidden');
        progressContainer.classList.remove('hidden');
        resultContainer.classList.add('hidden');

        // Simulate conversion progress
        let width = 0;
        progressBar.style.width = '0%';
        progressText.textContent = `Converting ${selectedFrom.toUpperCase()} to
${selectedTo.toUpperCase()}...`;

        const interval = setInterval(() => {
            width += Math.random() * 10;
            if (width >= 100) {
                width = 100;
                clearInterval(interval);
                setTimeout(showResult, 500);
            }
            progressBar.style.width = width + '%';
        }, 200);
    }

    function showResult() {
        progressContainer.classList.add('hidden');
        resultContainer.classList.remove('hidden');
        // This is a placeholder. In a real app, this would link to the converted file.
        downloadBtn.onclick = () => alert(`Downloading your converted .${selectedTo} file!
(Simulation)`);
    }

    function resetApp() {
        selectedFrom = null;
        selectedTo = null;
```

```
            document.querySelectorAll('.format-btn.selected').forEach(btn =>
btn.classList.remove('selected'));
            step2.classList.add('hidden');
            step3.classList.add('hidden');
            progressContainer.classList.add('hidden');
            resultContainer.classList.add('hidden');
        }

        // Event Listeners
        convertBtn.addEventListener('click', startConversion);
        startOverBtn.addEventListener('click', resetApp);

        // Initialize App
        document.addEventListener('DOMContentLoaded', () => {
            populateFromFormats();
        });
    </script>
</body>
</html>
```