<!DOCTYPE html>

<html lang="en" class="dark">

<head>

<meta charset="UTF-8">

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

<title>Streams Sphere IPTV Player</title>

<!-- HLS.js for streaming -->

<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>

<!-- Tailwind CSS for styling -->

<script src="https://cdn.tailwindcss.com"></script>

<!-- Google Fonts for typography -->

<link rel="preconnect" href="https://fonts.googleapis.com"> <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin> <link

href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;60

0;700&family=Orbitron:wght@700&display=swap" rel="stylesheet">

<!-- YouTube IFrame API -->

<script src="https://www.youtube.com/iframe\_api"></script>

<style>

/\* Aurora Glass Theme \*/

:root {

--bg-color: #0a0a14;

--text-color: #f0f0f5;

--text-muted: #a0a0b0;

--accent-color: #8A2BE2; /\* BlueViolet \*/

--accent-glow: rgba(138, 43, 226, 0.5);

--glass-bg: rgba(20, 20, 35, 0.55);

--border-color: rgba(138, 43, 226, 0.2);

}

html, body { overflow: hidden; height: 100%;

}

body {

font-family: 'Poppins', sans-serif;

background: linear-gradient(135deg, #0d042f, #2c135c,

#0a192f, #3d2f7b); color: var(--text-color); background-size: 400% 400%;

animation: aurora-flow 20s ease infinite;

}

@keyframes aurora-flow {

0% { background-position: 0% 50%; }

50% { background-position: 100% 50%; }

100% { background-position: 0% 50%; }

}

.hidden { display: none; }

.custom-scrollbar::-webkit-scrollbar { width: 8px; } .custom-scrollbar::-webkit-scrollbar-track { background: transparent; }

.custom-scrollbar::-webkit-scrollbar-thumb { background-color: var(--accent-color); border-radius: 4px; }

.glass-ui { background: var(--glass-bg); backdrop-filter: blur(16px); -webkit-backdrop-filter: blur(16px); border: 1px solid var(--border-color);

}

header, #sidebar { background: var(--glass-bg); backdrop-filter: blur(16px); -webkit-backdrop-filter: blur(16px); border-color: var(--border-color);

}

#player-container {

box-shadow: 0 0 25px var(--accent-glow), 0 0 10px rgba(0,0,0,0.5);

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

}

.header-title {

font-family: 'Orbitron', sans-serif; font-weight: 700;

text-shadow: 0 0 10px var(--accent-glow), 0 0 2px rgba(0,0,0,0.5);

animation: pulse-text-glow 3s infinite ease-in-out;

}

.channel-item.active {

background-color: rgba(138, 43, 226, 0.3); color: white;

animation: pulse-aurora 2s infinite;

}

@keyframes pulse-aurora {

0% { box-shadow: 0 0 0 0 var(--accent-glow); }

70% { box-shadow: 0 0 0 10px rgba(138, 43, 226, 0); } 100% { background-color: var(--accent-color); box-shadow:

0 0 0 0 rgba(138, 43, 226, 0); }

}

.channel-item.active .channel-title, .channel-item.active

.text-slate-400 { color: white;

}

.channel-item:hover, .category-item:hover { background-color: rgba(138, 43, 226, 0.2); }

.favorite-btn.favorite { color: var(--accent-color); }

.custom-input, .custom-btn { background-color: rgba(0, 0, 0, 0.2); border: 1px solid var(--border-color); color: var(--text-color); border-radius: 0.5rem; outline: none;

transition: all 0.2s ease-in-out;

}

.custom-input:focus, .custom-btn:focus { border-color: var(--accent-color); box-shadow: 0 0 8px var(--accent-glow);

}

.custom-btn {

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

}

.custom-btn:hover {

background-color: rgba(138, 43, 226, 0.8);

}

#sidebar { transition: transform 0.3s ease-in-out; }

.loader { border: 4px solid var(--border-color); border-top: 4px solid var(--accent-color); border-radius: 50%; width: 40px; height: 40px;

animation: spin 1s linear infinite; }

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

#source-selector-dropdown {

border: 1px solid rgba(138, 43, 226, 0.5); box-shadow: 0 0 20px var(--accent-glow);

}

#youtube-player-container, #youtube-player-div { position: absolute; top: 0; left: 0; width: 100%; height: 100%;

}

.dvd-case { position: relative;

background: linear-gradient(45deg, #1e103f, #0a0a14); border: 1px solid var(--border-color); border-radius: 0.5rem; overflow: hidden;

transition: transform 0.3s ease, box-shadow 0.3s ease; cursor: pointer; display: flex; flex-direction: column;

box-shadow: 0 5px 15px rgba(0,0,0,0.3), 0 0 5px var(--accent-glow);

}

.dvd-case:hover { transform: scale(1.05) perspective(1000px) rotateY(2deg); box-shadow: 0 10px 25px rgba(0,0,0,0.5), 0 0 20px var(--accent-glow);

}

.dvd-case-img-container { aspect-ratio: 2 / 3; background-color: #1a1a2e; display: flex; align-items: center; justify-content: center; overflow: hidden; position: relative;

}

.dvd-case-img-container::after { /\* Inner shadow for depth \*/ content: ''; position: absolute; top: 0; left: 0; right: 0; bottom: 0; box-shadow: inset 0 0 15px rgba(0,0,0,0.5);

}

.dvd-case-img { width: 100%; height: 100%; object-fit: cover;

transition: transform 0.3s ease;

}

.dvd-case:hover .dvd-case-img { transform: scale(1.1); }

.dvd-case-title { padding: 0.75rem 0.5rem; font-size: 0.8rem; font-weight: 500; text-align: center; white-space: nowrap; overflow: hidden; text-overflow: ellipsis;

background: linear-gradient(to top, rgba(0,0,0,0.7), rgba(0,0,0,0.4));

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

}

.dvd-case::before { content: ''; position: absolute; top: 0; left: -85%; z-index: 2; display: block; width: 50%; height: 100%;

background: linear-gradient(to right, rgba(255,255,255,0)

0%, rgba(255,255,255,0.3) 100%); transform: skewX(-25deg); transition: left 0.7s ease;

}

.dvd-case:hover::before { left: 110%; }

</style>

</head>

<body class="bg-slate-900">

<div class="relative h-full w-full flex flex-col">

<!-- Header -->

<header class="w-full border-b flex items-center justify-between p-2 z-20 flex-shrink-0">

<div class="flex items-center space-x-4"> <button id="menu-toggle" class="md:hidden p-2 text-slate-400 hover:text-purple-400 transition-colors">

<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"><line x1="3" y1="12" x2="21" y2="12"></line><line x1="3" y1="6" x2="21" y2="6"></line><line x1="3" y1="18" x2="21" y2="18"></line></svg>

</button>

<div class="flex items-center space-x-3">

<img id="logo"

src="https://images.weserv.nl/?url=iili.io/Fg2td42.png" alt="Streams

Sphere Logo" class="h-10 w-auto">

<span class="text-xl font-bold text-white header-title">Streams Sphere</span>

</div>

</div>

<div class="flex items-center space-x-2">

<button id="about-btn" class="p-2 text-sm custom-btn flex items-center space-x-2">

<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round"

stroke-linejoin="round"><circle cx="12" cy="12" r="10"></circle><line x1="12" y1="16" x2="12" y2="12"></line><line x1="12" y1="8" x2="12.01" y2="8"></line></svg>

<span>About</span>

</button>

</div>

</header>

<!-- Main Content -->

<div class="flex flex-1 overflow-hidden min-h-0">

<!-- Sidebar -->

<aside id="sidebar" class="absolute md:relative top-0 left-0 h-full w-64 md:w-72 border-r flex flex-col z-30 transform -translate-x-full md:translate-x-0 flex-shrink-0 min-h-0">

<div class="p-4 border-b border-[var(--border-color)]">

<div id="custom-select-container" class="relative"> <button id="source-selector-button" class="w-full p-2 custom-input flex items-center justify-between text-left"> <span class="flex items-center space-x-2 truncate">

<span id="source-selector-icon"></span>

<span id="source-selector-text">Select a

Playlist</span>

</span>

<svg class="w-4 h-4 text-slate-400" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20"

fill="currentColor"><path fill-rule="evenodd" d="M5.293 7.293a1 1 0

011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0

01-1.414 0l-4-4a1 1 0 010-1.414z" clip-rule="evenodd" /></svg>

</button>

<div id="source-selector-dropdown" class="absolute hidden mt-1 w-full glass-ui rounded-md shadow-lg z-40 flex flex-col">

<div class="p-2 border-b border-[var(--border-color)] space-y-2"> <input type="text"

id="playlist-search-input" placeholder="Search playlists..." class="w-full p-2 custom-input text-sm">

<button id="add-playlist-btn" class="w-full p-2 text-sm custom-btn flex items-center justify-center space-x-2">

<svg

xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0

24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><line x1="12" y1="5" x2="12" y2="19"></line><line x1="5" y1="12" x2="19" y2="12"></line></svg>

<span>Add Playlist</span>

</button>

</div>

<div id="playlist-options-container" class="max-h-60 overflow-y-auto custom-scrollbar"></div>

</div>

</div>

</div>

<div class="p-4 border-b border-[var(--border-color)]">

<input type="text" id="search-input" placeholder="Search channels..." class="w-full p-2 custom-input text-sm"> </div>

<div id="playlist-container" class="flex-1 overflow-y-auto custom-scrollbar">

<div id="playlist-element"></div>

</div>

</aside>

<!-- Player Area -->

<main class="flex-1 flex flex-col p-2 md:p-4 overflow-y-auto custom-scrollbar">

<div id="player-container" class="relative w-full aspect-video bg-black rounded-lg flex-shrink-0">

<video id="main-player" class="absolute top-0 left-0 w-full h-full rounded-lg" controls playsinline></video>

<div id="youtube-player-container" class="absolute top-0 left-0 w-full h-full hidden">

<div id="youtube-player-div"></div>

</div>

<div id="error-overlay" class="hidden absolute inset-0 bg-black bg-opacity-80 flex items-center justify-center text-center p-4 rounded-lg">

<p class="text-red-500 font-semibold">Could not load stream. The source may be offline or invalid.</p>

</div>

</div>

<!-- Now Playing Info -->

<div id="now-playing-info" class="mt-2 p-3 glass-ui rounded-lg text-center flex-shrink-0">

<span class="text-sm text-[var(--text-muted)]">Now Playing: </span>

<span id="now-playing-title" class="font-semibold text-white">Select a channel to begin</span>

</div>

<!-- Featured Channels Section -->

<div id="featured-container" class="my-4 px-2"> <h3 class="text-lg font-bold text-white mb-4 header-title text-center">Featured Movies and Live Tv</h3> <div id="featured-channels-grid" class="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 lg:grid-cols-5 gap-4"> <!-- Channels will be injected here by the script

-->

</div>

</div>

</main>

</div>

</div>

<!-- Add Playlist Modal -->

<div id="add-playlist-modal" class="hidden fixed inset-0 bg-black bg-opacity-75 flex items-center justify-center z-50"> <div class="glass-ui p-6 text-center rounded-lg shadow-lg shadow-purple-500/20 max-w-md w-full mx-4 space-y-4"> <h2 class="text-xl font-bold header-title">Add Custom

Playlist</h2>

<input type="text" id="new-playlist-name" placeholder="Playlist Name" class="w-full p-2 custom-input"> <input type="url" id="new-playlist-url" placeholder="M3U

Playlist URL" class="w-full p-2 custom-input">

<div class="flex justify-end space-x-4 pt-4">

<button id="cancel-add-playlist-btn" class="p-2 px-4 custom-btn bg-gray-600 hover:bg-gray-500">Cancel</button> <button id="save-playlist-btn" class="p-2 px-4 custom-btn">Save</button>

</div>

</div>

</div>

<!-- About / DMCA Modal -->

<div id="dmca-modal" class="hidden fixed inset-0 bg-black bg-opacity-75 flex items-center justify-center z-50 p-4"> <div id="dmca-modal-content" class="glass-ui p-6 rounded-lg shadow-lg shadow-purple-500/20 max-w-2xl w-full space-y-4 relative">

<h2 class="text-xl font-bold header-title text-center">Disclaimer & DMCA Policy</h2>

<div class="max-h-[70vh] overflow-y-auto custom-scrollbar pr-4 text-sm text-slate-300 space-y-3">

<h3 class="font-bold text-white pt-2">Copyright & DMCA Policy</h3>

<p class="text-xs text-slate-400">Effective Date: August

7, 2025</p>

<h4 class="font-semibold text-white">1. Our Commitment to Copyright</h4>

<p>StreamsSphere.tv respects the intellectual property rights of others and expects its users to do the same. It is our policy to respond to clear notices of alleged copyright infringement that comply with the Digital Millennium Copyright Act (DMCA).</p>

<h4 class="font-semibold text-white">2. We Do Not Host Any

Content</h4>

<p>This is a critical point to understand:

StreamsSphere.tv does not host, store, or distribute any of the video or media files displayed on our pages.</p>

<p>Our website functions as a search engine or directory.

We provide links to content that is publicly available on the internet. Specifically, our service redirects users to media hosted on third-party services, primarily:</p>

<ul class="list-disc list-inside pl-4 space-y-1">

<li>The Internet Archive (internetarchive.com)</li>

<li>Content indexed by the IPTVorg GitHub Repository

(github.com/iptv-org/iptv)</li>

</ul>

<p>All intellectual property rights for the content linked from our site remain with their respective owners. We have no control over the content on these third-party websites.</p>

<h4 class="font-semibold text-white">3. How to File a

Copyright Infringement Claim</h4>

<p class="font-bold">DO NOT send DMCA takedown notices to us.</p>

<p>Because we do not host any content on our servers, we are unable to process or comply with DMCA takedown requests. Removing a link from our website will not remove the actual content from the web.</p>

<p>To have infringing content removed, you must contact the web host or content provider that is actually storing the material.</p>

<p>Follow these steps:</p>

<ol class="list-decimal list-inside pl-4 space-y-1"> <li><strong>Identify the Source:</strong> On our website, follow the link to the movie, series, or stream you believe is infringing on your copyright. This will take you to the third-party site where the content is hosted.</li>

<li><strong>Contact the Host Provider:</strong> You must submit your DMCA takedown notice directly to the company or entity hosting the content.

<ul class="list-disc list-inside pl-6 space-y-1 mt-1">

<li>For content from the Internet Archive:

Please follow their copyright policy and submit your claim directly to them. (Policy: https://archive.org/about/terms.php)</li>

<li>For content from the IPTVorg repository:

The IPTVorg repository on GitHub is a collection of links to publicly available IPTV streams. The infringing content is not on GitHub, but on the servers that provide the stream. You must direct your claim to the hosting provider of the stream itself. GitHub is only hosting the playlist file (.m3u), not the video content.</li>

</ul>

</li>

</ol>

<h4 class="font-semibold text-white">4. Summary</h4> <p>StreamsSphere.tv is simply a search index of publicly available content. We do not host files. All copyright and DMCA issues must be directed to the websites and services that are hosting the content. We take intellectual property matters very seriously and urge all copyright holders to direct their claims to the appropriate parties.</p>

<p>If you have any questions about this policy, you may contact us at StreamsSpheretv@gmail.com, but please note again that we do not process DMCA claims.</p>

</div>

<button id="close-dmca-btn" class="absolute top-2 right-2 p-1 text-slate-400 hover:text-white">

<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"><line x1="18" y1="6" x2="6" y2="18"></line><line x1="6" y1="6" x2="18" y2="18"></line></svg>

</button>

</div>

</div>

<!-- Ad Modal -->

<div id="ad-modal" class="hidden fixed inset-0 bg-black bg-opacity-75 flex items-center justify-center z-50">

<div class="glass-ui p-4 text-center rounded-lg shadow-lg shadow-purple-500/20 max-w-sm w-full mx-4">

<img id="ad-image"

src="https://via.placeholder.com/300x250/1a1a2e/ffffff?text=Advertisem ent" alt="Ad" class="mx-auto mb-4 cursor-pointer rounded" >

<button id="close-ad-btn" class="mt-2 p-2 w-full bg-purple-600 text-white font-bold rounded hover:bg-purple-500 transition-colors">Close Ad</button>

</div>

</div>

<div id="mobile-overlay" class="fixed inset-0 bg-black bg-opacity-50 z-20 hidden md:hidden"></div>

<script>

// This is the global callback function that the YouTube IFrame API will call

function onYouTubeIframeAPIReady() {

window.dispatchEvent(new Event('youtube-api-ready')); } document.addEventListener('DOMContentLoaded', function() {

// DOM Element references const elements = { mainPlayer: document.getElementById('main-player'), youtubePlayerContainer:

document.getElementById('youtube-player-container'), youtubePlayerDiv:

document.getElementById('youtube-player-div'),

playerContainer: document.getElementById('player-container'), errorOverlay: document.getElementById('error-overlay'), sourceSelectorButton: document.getElementById('source-selector-button'), sourceSelectorIcon:

document.getElementById('source-selector-icon'), sourceSelectorText: document.getElementById('source-selector-text'), sourceSelectorDropdown:

document.getElementById('source-selector-dropdown'), playlistSearchInput: document.getElementById('playlist-search-input'), playlistOptionsContainer: document.getElementById('playlist-options-container'), searchInput: document.getElementById('search-input'), playlistElement: document.getElementById('playlist-element'), adModal: document.getElementById('ad-modal'), adImage: document.getElementById('ad-image'), closeAdBtn: document.getElementById('close-ad-btn'), sidebar: document.getElementById('sidebar'), menuToggle: document.getElementById('menu-toggle'), mobileOverlay: document.getElementById('mobile-overlay'), featuredChannelsGrid: document.getElementById('featured-channels-grid'),

addPlaylistBtn: document.getElementById('add-playlist-btn'), addPlaylistModal: document.getElementById('add-playlist-modal'),

newPlaylistName: document.getElementById('new-playlist-name'), newPlaylistUrl: document.getElementById('new-playlist-url'), savePlaylistBtn: document.getElementById('save-playlist-btn'), cancelAddPlaylistBtn:

document.getElementById('cancel-add-playlist-btn'),

nowPlayingTitle: document.getElementById('now-playing-title'), aboutBtn: document.getElementById('about-btn'), dmcaModal: document.getElementById('dmca-modal'), dmcaModalContent:

document.getElementById('dmca-modal-content'),

closeDmcaBtn: document.getElementById('close-dmca-btn'),

};

// State variables

let hls, ytPlayer, allChannelsInPlaylist = [], masterAllChannels =

[];

let favorites = JSON.parse(localStorage.getItem('iptv\_favorites'))

|| [];

let customPlaylists =

JSON.parse(localStorage.getItem('iptv\_custom\_playlists')) || []; let channelClickCounter = 0, activePlayer = null, urlToPlayAfterAd

= null; let lastClickedUrl = null; let currentlyPlayingUrl = null; // TMDB API Key and Genre Map

const TMDB\_API\_KEY = '3ea9e43d0b879725ac8fda707021e858'; const TMDB\_GENRE\_MAP =

{28:"Action",12:"Adventure",16:"Animation",35:"Comedy",80:"Crime",99:" Documentary",18:"Drama",10751:"Family",14:"Fantasy",36:"History",27:"H orror",10402:"Music",9648:"Mystery",10749:"Romance",878:"Science

Fiction",10770:"TV Movie",53:"Thriller",10752:"War",37:"Western"};

function extractVideoID(url) { if (!url) return null; const regex =

/(?:https?:\/\/)?(?:www\.)?(?:youtube\.com\/(?:[^\/\n\s]+\/\S+\/|(?:v| e(?:mbed)?|live)\/|\S\*?[?&]v=)|youtu\.be\/)([a-zA-Z0-9\_-]{11})/; const match = url.match(regex); return match ? match[1] : null;

}

// Data

const countryMap =

{'af':{name:'Afghanistan',flag:'󰎀'},'al':{name:'Albania',flag:'󰎃'},' dz':{name:'Algeria',flag:'󰎷'},'ad':{name:'Andorra',flag:'󰍾'},'ao':{n ame:'Angola',flag:'󰎅'},'ar':{name:'Argentina',flag:'󰎆'},'am':{name:' Armenia',flag:'󰎄'},'aw':{name:'Aruba',flag:'󰎊'},'au':{name:'Australi a',flag:'󰎉'},'at':{name:'Austria',flag:'󰎈'},'az':{name:'Azerbaijan', flag:'󰎌'},'bs':{name:'Bahamas',flag:'󰎚'},'bh':{name:'Bahrain',flag:' 󰎓'},'bd':{name:'Bangladesh',flag:'󰎏'},'bb':{name:'Barbados',flag:'󰎎

'},'by':{name:'Belarus',flag:'󰎝'},'be':{name:'Belgium',flag:'󰎐'},'bz

':{name:'Belize',flag:'󰎞'},'bj':{name:'Benin',flag:'󰎕'},'bm':{name:'

Bermuda',flag:'󰎖'},'bt':{name:'Bhutan',flag:'󰎛'},'bo':{name:'Bolivia ',flag:'󰎘'},'ba':{name:'Bosnia and

Herzegovina',flag:'󰎍'},'bw':{name:'Botswana',flag:'󰎜'},'br':{name:'B razil',flag:'󰎙'},'bn':{name:'Brunei',flag:'󰎗'},'bg':{name:'Bulgaria' ,flag:'󰎒'},'bf':{name:'Burkina

Faso',flag:'󰎑'},'bi':{name:'Burundi',flag:'󰎔'},'kh':{name:'Cambodia' ,flag:'󰏩'},'cm':{name:'Cameroon',flag:'󰎨'},'ca':{name:'Canada',flag:

'󰎟'},'cv':{name:'Cape Verde',flag:'󰎭'},'ky':{name:'Cayman

Islands',flag:'󰏰'},'cf':{name:'Central African

Republic',flag:'󰎢'},'td':{name:'Chad',flag:'󰑄'},'cl':{name:'Chile',f lag:'󰎧'},'cn':{name:'China',flag:'󰎩'},'co':{name:'Colombia',flag:'󰎪

'},'km':{name:'Comoros',flag:'󰏫'},'cg':{name:'Congo',flag:'󰎣'},'cd':

{name:'DR Congo',flag:'󰎡'},'cr':{name:'Costa Rica',flag:'󰎫'},'ci':{name:'Cote

d\'Ivoire',flag:'󰎥'},'hr':{name:'Croatia',flag:'󰏖'},'cu':{name:'Cuba ',flag:'󰎬'},'cy':{name:'Cyprus',flag:'󰎰'},'cz':{name:'Czech Republic',flag:'󰎱'},'dk':{name:'Denmark',flag:'󰎴'},'dj':{name:'Djibo uti',flag:'󰎳'},'dm':{name:'Dominica',flag:'󰎵'},'do':{name:'Dominican

Republic',flag:'󰎶'},'ec':{name:'Ecuador',flag:'󰎸'},'eg':{name:'Egypt

',flag:'󰎺'},'sv':{name:'El Salvador',flag:'󰐿'},'gq':{name:'Equatorial

Guinea',flag:'󰏎'},'ee':{name:'Estonia',flag:'󰎹'},'et':{name:'Ethiopi a',flag:'󰎽'},'fj':{name:'Fiji',flag:'󰏀'},'fi':{name:'Finland',flag:' 󰎿'},'fr':{name:'France',flag:'󰏃'},'ga':{name:'Gabon',flag:'󰏄'},'gm'

:{name:'Gambia',flag:'󰏌'},'ge':{name:'Georgia',flag:'󰏇'},'de':{name: 'Germany',flag:'󰎲'},'gh':{name:'Ghana',flag:'󰏉'},'gi':{name:'Gibralt ar',flag:'󰏊'},'gr':{name:'Greece',flag:'󰏏'},'gl':{name:'Greenland',f lag:'󰏋'},'gd':{name:'Grenada',flag:'󰏆'},'gp':{name:'Guadeloupe',flag :'󰖌'},'gu':{name:'Guam',flag:'󰏑'},'gt':{name:'Guatemala',flag:'󰏐'}, 'gn':{name:'Guinea',flag:'󰏍'},'gw':{name:'Guinea-Bissau',flag:'󰏒'},' gy':{name:'Guyana',flag:'󰏓'},'ht':{name:'Haiti',flag:'󰏗'},'hn':{name

:'Honduras',flag:'󰏕'},'hk':{name:'Hong

Kong',flag:'󰏔'},'hu':{name:'Hungary',flag:'󰏘'},'is':{name:'Iceland', flag:'󰏡'},'in':{name:'India',flag:'󰏝'},'id':{name:'Indonesia',flag:' 󰏙'},'ir':{name:'Iran',flag:'󰏠'},'iq':{name:'Iraq',flag:'󰏟'},'ie':{n ame:'Ireland',flag:'󰏚'},'im':{name:'Isle of

Man',flag:'󰏜'},'il':{name:'Israel',flag:'󰏛'},'it':{name:'Italy',flag :'󰏢'},'jm':{name:'Jamaica',flag:'󰏤'},'jp':{name:'Japan',flag:'󰏦'},' je':{name:'Jersey',flag:'󰏣'},'jo':{name:'Jordan',flag:'󰏥'},'kz':{nam e:'Kazakhstan',flag:'󰏱'},'ke':{name:'Kenya',flag:'󰏧'},'xk':{name:'Ko sovo',flag:'󰖜'},'kw':{name:'Kuwait',flag:'󰏯'},'kg':{name:'Kyrgyzstan ',flag:'󰏨'},'la':{name:'Laos',flag:'󰏲'},'lv':{name:'Latvia',flag:'󰏻

'},'lb':{name:'Lebanon',flag:'󰏳'},'ls':{name:'Lesotho',flag:'󰏸'},'lr ':{name:'Liberia',flag:'󰏷'},'ly':{name:'Libya',flag:'󰏼'},'li':{name: 'Liechtenstein',flag:'󰏵'},'lt':{name:'Lithuania',flag:'󰏹'},'lu':{nam e:'Luxembourg',flag:'󰏺'},'mo':{name:'Macau',flag:'󰐇'},'mk':{name:'No rth

Macedonia',flag:'󰐃'},'mg':{name:'Madagascar',flag:' '},'mw':{name:'M alawi',flag:'󰐎'},'my':{name:'Malaysia',flag:'󰐐'},'mv':{name:'Maldive s',flag:'󰐍'},'ml':{name:'Mali',flag:'󰐄'},'mt':{name:'Malta',flag:'󰐋 '},'mq':{name:'Martinique',flag:'󰖐'},'mr':{name:'Mauritania',flag:'󰐉 '},'mu':{name:'Mauritius',flag:'󰐌'},'yt':{name:'Mayotte',flag:'󰖝'},' mx':{name:'Mexico',flag:'󰐏'},'md':{name:'Moldova',flag:'󰏿'},'mc':{na me:'Monaco',flag:'󰏾'},'mn':{name:'Mongolia',flag:'󰐆'},'me':{name:'Mo ntenegro',flag:' '},'ms':{name:'Montserrat',flag:'󰐊'},'ma':{name:'Mo rocco',flag:'󰏽'},'mz':{name:'Mozambique',flag:'󰐑'},'mm':{name:'Myanm ar',flag:'󰐅'},'na':{name:'Namibia',flag:'󰐒'},'np':{name:'Nepal',flag

:'󰐙'},'nl':{name:'Netherlands',flag:'󰐗'},'nc':{name:'New

Caledonia',flag:'󰖒'},'nz':{name:'New

Zealand',flag:'󰐜'},'ni':{name:'Nicaragua',flag:'󰐖'},'ne':{name:'Nige r',flag:'󰐓'},'ng':{name:'Nigeria',flag:'󰐕'},'kp':{name:'North Korea',flag:'󰏭'},'no':{name:'Norway',flag:'󰐘'},'om':{name:'Oman',fla g:'󰐝'},'pk':{name:'Pakistan',flag:'󰐣'},'ps':{name:'Palestine',flag:' 󰐧'},'pa':{name:'Panama',flag:'󰐞'},'pg':{name:'Papua New

Guinea',flag:'󰐡'},'py':{name:'Paraguay',flag:'󰐪'},'pe':{name:'Peru', flag:'󰐟'},'ph':{name:'Philippines',flag:'󰐢'},'pl':{name:'Poland',fla g:'󰐤'},'pt':{name:'Portugal',flag:'󰐨'},'pr':{name:'Puerto

Rico',flag:'󰐦'},'qa':{name:'Qatar',flag:'󰐫'},'re':{name:'Reunion',fl ag:'󰖔'},'ro':{name:'Romania',flag:'󰐬'},'ru':{name:'Russia',flag:'󰐮'

},'rw':{name:'Rwanda',flag:'󰐯'},'kn':{name:'Saint Kitts and

Nevis',flag:'󰏬'},'lc':{name:'Saint

Lucia',flag:'󰏴'},'vc':{name:'Saint Vincent and the

Grenadines',flag:'󰑘'},'sm':{name:'San

Marino',flag:'󰐹'},'st':{name:'Sao Tome and

Principe',flag:'󰐾'},'sa':{name:'Saudi

Arabia',flag:'󰐰'},'sn':{name:'Senegal',flag:'󰐺'},'rs':{name:'Serbia' ,flag:'󰐭'},'sc':{name:'Seychelles',flag:'󰐲'},'sl':{name:'Sierra Leone',flag:'󰐸'},'sg':{name:'Singapore',flag:'󰐵'},'sk':{name:'Slovak ia',flag:'󰐷'},'si':{name:'Slovenia',flag:'󰐶'},'so':{name:'Somalia',f lag:'󰐻'},'za':{name:'South Africa',flag:'󰑠'},'kr':{name:'South

Korea',flag:'󰏮'},'ss':{name:'South

Sudan',flag:'󰐽'},'es':{name:'Spain',flag:'󰎼'},'lk':{name:'Sri Lanka',flag:'󰏶'},'sd':{name:'Sudan',flag:'󰐳'},'sr':{name:'Suriname', flag:'󰐼'},'se':{name:'Sweden',flag:'󰐴'},'ch':{name:'Switzerland',fla g:'󰎤'},'sy':{name:'Syria',flag:'󰑁'},'tw':{name:'Taiwan',flag:'󰑐'},' tz':{name:'Tanzania',flag:'󰑑'},'th':{name:'Thailand',flag:'󰑆'},'tl': {name:'Timor-Leste',flag:'󰑉'},'tg':{name:'Togo',flag:'󰑅'},'to':{name

:'Tonga',flag:'󰑌'},'tt':{name:'Trinidad and

Tobago',flag:'󰑎'},'tn':{name:'Tunisia',flag:'󰑋'},'tr':{name:'Turkey' ,flag:'󰑍'},'tm':{name:'Turkmenistan',flag:'󰑊'},'ug':{name:'Uganda',f lag:'󰑓'},'ua':{name:'Ukraine',flag:'󰑒'},'ae':{name:'United Arab

Emirates',flag:'󰍿'},'gb':{name:'United

Kingdom',flag:'󰏅'},'us':{name:'United

States',flag:'󰑔'},'uy':{name:'Uruguay',flag:'󰑕'},'uz':{name:'Uzbekis tan',flag:'󰑖'},'ve':{name:'Venezuela',flag:'󰑙'},'vn':{name:'Vietnam' ,flag:'󰑜'},'ye':{name:'Yemen',flag:'󰑟'},'zm':{name:'Zambia',flag:'󰑡

'},'zw':{name:'Zimbabwe',flag:'󰑢'}}; const countryCodes = Object.keys(countryMap); const countryPlaylists = countryCodes.map(code => { const country = countryMap[code]; const url = (code === 'gb') ?

'https://raw.githubusercontent.com/devsground/IPTV/master/country/T-Z/ united\_kingdom.m3u'

: `https://iptv-org.github.io/iptv/countries/${code}.m3u`; return { name: country.name, url: url, icon: `<span class="text-xl">${country.flag}</span>` };

}); let BASE\_PLAYLISTS = [

{ name: 'YT Live', icon: '🔴', channels: [ { title: 'The Osbournes Podcast', url:

'https://www.youtube.com/live/mozvQNIsvik', logo: "", sourcePlaylistName: 'YT Live'},

{ title: 'lofi hip hop radio - beats to relax/study to', url: 'https://www.youtube.com/watch?v=jfKfPfyJRdk', logo: "", sourcePlaylistName: 'YT Live'},

]},

{ name: 'YT Series', icon: '📺', categorized: true, channels:

[

].map(video => ({ ...video, logo:

`https://i.ytimg.com/vi/${extractVideoID(video.url)}/hqdefault.jpg`, sourcePlaylistName: 'YT Series' })) },

{ name: 'Fire TV', url: 'https://www.apsattv.com/firetv.m3u', icon: '🔥'},

{ name: 'Freetv', url: 'https://www.apsattv.com/freetv.m3u', icon: '📺'},

{ name: 'Freemoviesplus', url:

'https://www.apsattv.com/freemoviesplus.m3u', icon: '🍿'},

{ name: 'Klowd TV', url: 'https://www.apsattv.com/klowd.m3u', icon: '☁'},

{ name: 'LG', url: 'https://www.apsattv.com/lg.m3u', icon: '📺'},

{ name: 'Plex US', url:

'https://raw.githubusercontent.com/iptv-org/iptv/master/streams/us\_ple x.m3u', icon: '▶'}, {

name: 'Internet Archive 📺🎬', icon: '🏛', isNested: true, structure: { 'Movies': {

// Movie genres will be populated dynamically here

},

'Series': {

'Animation': {

'King of The Hill': [],

'South Park': []

},

'Comedy': {

'The Inbetweeners': [],

'Fawlty Towers': []

}

}

}

},

{ name: 'Weelly', url: 'https://www.apsattv.com/weelly.m3u', icon: '📺'},

{ name: 'Vidaa', url: 'https://www.apsattv.com/vidaa.m3u', icon: '📺'},

{ name: 'Xiaomi', url: 'https://www.apsattv.com/xiaomi.m3u', icon: '📱'},

{ name: 'Xumo', url: 'https://www.apsattv.com/xumo.m3u', icon:

'📺'},

{ name: 'Zeasn', url: 'https://www.apsattv.com/zeasn.m3u', icon: '📺'},

...countryPlaylists

].sort((a, b) => a.name.localeCompare(b.name));

let allPlaylists = [];

const CORS\_PROXIES = [ '', // Direct fetch first

'https://corsproxy.io/?',

'https://api.allorigins.win/raw?url=',

'https://proxy.cors.sh/'

];

const CACHE\_EXPIRATION\_MS = 2 \* 60 \* 60 \* 1000; // 2 hours

// --- Player Logic --- function initializeYouTubePlayer() {

if (typeof YT === 'undefined' || typeof YT.Player ===

'undefined') {

window.addEventListener('youtube-api-ready', createPlayer,

{ once: true }); } else {

createPlayer();

}

} function createPlayer() { if (ytPlayer) return;

ytPlayer = new YT.Player(elements.youtubePlayerDiv, { height: '100%', width: '100%', playerVars: { 'playsinline': 1,

'autoplay': 1,

'rel': 0,

'modestbranding': 1

}, events: {

'onStateChange': onPlayerStateChange

}

});

} function onPlayerStateChange(event) { if (event.data === YT.PlayerState.ENDED) { const currentPlaylistName = elements.sourceSelectorText.textContent.trim();

const playlist = allPlaylists.find(p => p.name === currentPlaylistName);

if(playlist) playNextEpisode(currentlyPlayingUrl, playlist.name);

}

} function playNextEpisode(currentUrl, playlistName) { if (!currentUrl) return;

const playlist = allPlaylists.find(p => p.name === playlistName);

if (!playlist || (!playlist.categorized &&

!playlist.isNested)) return;

const currentChannel = allChannelsInPlaylist.find(c => c.url

=== currentUrl);

if (!currentChannel || !currentChannel.group) return;

let seriesChannels = allChannelsInPlaylist.filter(c => c.group

=== currentChannel.group);

seriesChannels.sort((a, b) => { const numA = a.title.match(/\d+/g); const numB = b.title.match(/\d+/g); if (numA && numB) { const lastNumA = parseInt(numA[numA.length-1]); const lastNumB = parseInt(numB[numB.length-1]); if(lastNumA !== lastNumB) return lastNumA - lastNumB;

}

return a.title.localeCompare(b.title);

});

const currentIndex = seriesChannels.findIndex(c => c.url === currentUrl);

if (currentIndex > -1 && currentIndex < seriesChannels.length

- 1) {

const nextChannel = seriesChannels[currentIndex + 1]; loadStream(nextChannel.url); const nextChannelElement =

document.querySelector(`.channel-item[data-url="${CSS.escape(nextChann el.url)}"]`);

if (nextChannelElement) {

document.querySelectorAll('.channel-item').forEach(item => item.classList.remove('active'));

nextChannelElement.classList.add('active'); }

}

}

function initializePlayers() {

elements.mainPlayer.addEventListener('error', (e) => { if (activePlayer === 'native') {

console.error('Native Video Element Error:', elements.mainPlayer.error); showErrorOverlay();

}

});

elements.mainPlayer.addEventListener('ended', () => { const currentPlaylistName = elements.sourceSelectorText.textContent.trim();

const playlist = allPlaylists.find(p => p.name === currentPlaylistName);

if(playlist) playNextEpisode(currentlyPlayingUrl, playlist.name);

});

initializeYouTubePlayer();

} function showErrorOverlay() {

elements.errorOverlay.classList.remove('hidden'); } function hideErrorOverlay() {

elements.errorOverlay.classList.add('hidden'); }

function showAd(isInitialAd = false) { if (isInitialAd) { const lastPlayed = localStorage.getItem('iptv\_last\_played'); if (lastPlayed) {

urlToPlayAfterAd = lastPlayed;

} }

elements.mainPlayer.pause();

if (activePlayer === 'youtube' && ytPlayer && typeof ytPlayer.pauseVideo === 'function') { ytPlayer.pauseVideo();

}

elements.adModal.classList.remove('hidden');

} function hideAdAndOpenLink() {

window.open('https://www.profitableratecpm.com/qb069j7b6?key=e6b8dcd0a 21df19017732330d1236fd3', '\_blank'); elements.adModal.classList.add('hidden'); if (urlToPlayAfterAd) { playStream(urlToPlayAfterAd); urlToPlayAfterAd = null;

} else if (activePlayer === 'youtube' && ytPlayer && typeof ytPlayer.playVideo === 'function') { ytPlayer.playVideo();

} else if (activePlayer === 'hls' || activePlayer ===

'native') { elements.mainPlayer.play().catch(e => console.warn("Playback resume failed.", e));

}

}

async function fetchWithFallback(url) { for (const proxy of CORS\_PROXIES) { try { const fetchUrl = proxy ? proxy + url : url; const response = await fetch(fetchUrl); if (response.ok) return await response.text();

} catch (e) {

console.warn(`Fetch failed for ${url} with proxy

'${proxy}':`, e);

} }

throw new Error(`All CORS proxies failed for ${url}.`);

} async function parseM3U(data, playlistName, forceGroupName = null)

{

const lines = data.split('\n'); const channels = []; let currentGroups = {}; for (let i = 0; i < lines.length; i++) { if (lines[i].startsWith('#EXTINF:')) { const info = lines[i]; let url = lines[i + 1]?.trim(); if (url && (url.startsWith('http') || url.startsWith('//'))) { if (url.includes('archive.org/embed/')) { url = url.replace('/embed/', '/download/');

}

if (url.includes('archive.org')) {

url = url.replace('http://', 'https://');

}

let title = info.match(/,(.+)/)?.[1]?.trim() || 'Unknown';

const logo = info.match(/tvg-logo="([^"]+)"/)?.[1]; let group = forceGroupName ||

info.match(/group-title="([^"]+)"/)?.[1]?.trim() || 'Uncategorized';

if (['King of The Hill', 'South Park', 'The Inbetweeners', 'Fawlty Towers'].includes(forceGroupName)) { const match = title.match(/S(\d{1,2})E(\d{1,2})\s?-\s?(.\*)/i) || title.match(/S(\d{1,2})\s?E(\d{1,2})\s?(.\*)/i) || title.match(/S(\d{2})E(\d{2})\.\s\*(.\*)/i); if (match) { const season = match[1].padStart(2, '0'); const episode = match[2].padStart(2, '0'); const episodeTitle = match[3].trim(); title = `S${season} E${episode} -

${episodeTitle}`;

}

}

const channelData = { title, logo, url, group, sourcePlaylistName: playlistName }; channels.push(channelData); if (!currentGroups[group]) { currentGroups[group] = [];

}

currentGroups[group].push(channelData);

}

} }

return { channels, groups: currentGroups };

} async function loadDataForSource(sourceKey) { elements.playlistElement.innerHTML = `<div class="flex justify-center items-center h-full"><div class="loader"></div></div>`;

const playlist = allPlaylists.find(p => p.name === sourceKey); if (!playlist) { elements.playlistElement.innerHTML = `<p class="p-4 text-center text-[var(--text-muted)]">Playlist not found.</p>`; return;

}

if (playlist.isNested) {

displayInternetArchiveCategories([], playlist.structure); return;

}

allChannelsInPlaylist = []; let groupedChannels = {};

if (sourceKey === 'Favorites') { if (favorites.length === 0) { elements.playlistElement.innerHTML = `<p class="p-4 text-center text-[var(--text-muted)]">No favorites yet.</p>`; return; } allChannelsInPlaylist = favorites; populatePlaylistUI(favorites, false); return;

}

if (playlist.channels) {

allChannelsInPlaylist = playlist.channels; if (playlist.channels.length === 0) { elements.playlistElement.innerHTML = `<p class="p-4 text-center text-[var(--text-muted)]">This playlist is currently empty.</p>`;

return; }

groupedChannels = {};

playlist.channels.forEach(channel => { const group = channel.group || playlist.name; if (!groupedChannels[group]) { groupedChannels[group] = [];

}

groupedChannels[group].push(channel);

});

await displayCategories(groupedChannels); return;

}

const cacheKey = `iptv\_cache\_${playlist.name}`; try {

const cachedItem = localStorage.getItem(cacheKey); if (cachedItem) { const { timestamp, data } = JSON.parse(cachedItem); if ((Date.now() - timestamp) < CACHE\_EXPIRATION\_MS) { const parsed = await parseM3U(data, playlist.name); allChannelsInPlaylist = parsed.channels; groupedChannels = parsed.groups; await displayCategories(groupedChannels); return;

}

} const data = await fetchWithFallback(playlist.url); localStorage.setItem(cacheKey, JSON.stringify({ timestamp: Date.now(), data: data }));

const parsed = await parseM3U(data, playlist.name); allChannelsInPlaylist = parsed.channels; groupedChannels = parsed.groups; await displayCategories(groupedChannels);

} catch (error) {

console.error(`Failed to load playlist ${playlist.name}`, error); elements.playlistElement.innerHTML = `<p class="p-4 text-center text-red-500">Could not load ${playlist.name}.</p>`;

}

} const categoryIcons = {

'Business': '💼', 'Comedy': '😂', 'Culture': '🎭',

'Documentary': '🎥',

'Family': '󰔡', 'General': '🌐', 'Kids': '👶', 'Lifestyle': '💅',

'Music': '🎵', 'News': '📰', 'Religious': '⛪', 'Sports': '⚽',

'Classic': '🏛', 'Horror': '👻', 'Science Fiction': '👽', 'Animation': '✏',

'War': '⚔', 'Action': '💥', 'Drama': '🎭', 'Mystery': '❓',

'Western': '🤠', 'Mr Bean': `<img

src="https://i.ibb.co/9vR9zG1/mrbean-png.webp" class="w-6 h-6 rounded-sm object-cover">`,

'Hells Kitchen': '🔥', 'The Island With Bear Grylls': '🏝',

"Gargoyles": '🦇', 'Tarzan: The Epic Adventures': '🌳', 'King of The

Hill': '👑', 'South Park': '😂', 'The Inbetweeners': '🚌', 'Fawlty Towers': '🏨',

'All': '📚', 'Default': '📁', 'Movies': '🎬', 'Series': '📺',

'All Movies': '🎬'

}; function displayInternetArchiveCategories(path = [], structure) { elements.playlistElement.innerHTML = ''; let currentLevel = structure; path.forEach(key => {

currentLevel = currentLevel[key];

});

if (path.length > 0) {

const backButton = document.createElement('button'); backButton.innerHTML = `<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0

24 24" fill="none" stroke="currentColor" stroke-width="2"

stroke-linecap="round" stroke-linejoin="round"><polyline points="15 18

9 12 15 6"></polyline></svg> Back`; backButton.className = 'back-button w-full flex items-center space-x-2 p-3 text-left text-purple-400 hover:bg-purple-500/20 border-b border-[var(--border-color)]'; backButton.addEventListener('click', () => { const newPath = path.slice(0, -1);

displayInternetArchiveCategories(newPath, structure);

});

elements.playlistElement.appendChild(backButton);

}

const sortedKeys = Object.keys(currentLevel).sort((a, b) => { if (a === 'All Movies') return -1; if (b === 'All Movies') return 1; return a.localeCompare(b);

});

for (const key of sortedKeys) { const nextLevel = currentLevel[key];

const isChannelList = Array.isArray(nextLevel);

const categoryDiv = document.createElement('div'); categoryDiv.className = 'category-item flex items-center space-x-3 p-3 cursor-pointer hover:bg-purple-500/20 transition-colors duration-200 border-b border-[var(--border-color)]'; const iconHTML = categoryIcons[key] || categoryIcons['Default'];

if (isChannelList) { categoryDiv.innerHTML = `<div class="flex-shrink-0 w-6 h-6 flex items-center justify-center text-xl">${iconHTML}</div>

<span>${key} (${nextLevel.length})</span>`;

categoryDiv.addEventListener('click', () => { allChannelsInPlaylist = nextLevel; populatePlaylistUI(nextLevel, true, () => displayInternetArchiveCategories(path, structure));

}); } else { categoryDiv.innerHTML = `<div class="flex-shrink-0 w-6 h-6 flex items-center justify-center text-xl">${iconHTML}</div>

<span>${key}</span>`;

categoryDiv.addEventListener('click', () => { displayInternetArchiveCategories([...path, key], structure); });

}

elements.playlistElement.appendChild(categoryDiv);

}

} async function displayCategories(groupedChannels) { elements.playlistElement.innerHTML = ``; if (allChannelsInPlaylist.length > 0 && Object.keys(groupedChannels).length > 1) {

const allChannelsDiv = document.createElement('div'); allChannelsDiv.className = 'category-item flex items-center space-x-3 p-3 cursor-pointer hover:bg-purple-500/20 transition-colors duration-200 border-b border-[var(--border-color)]'; allChannelsDiv.innerHTML = `<div class="flex-shrink-0 w-6 h-6 flex items-center justify-center

text-xl">${categoryIcons['All']}</div> <span>All Channels

(${allChannelsInPlaylist.length})</span>`; allChannelsDiv.addEventListener('click', () => populatePlaylistUI(allChannelsInPlaylist, true, () => displayCategories(groupedChannels)));

elements.playlistElement.appendChild(allChannelsDiv);

}

const sortedCategories =

Object.entries(groupedChannels).sort(([a], [b]) => a.localeCompare(b)); if (sortedCategories.length === 1) {

populatePlaylistUI(sortedCategories[0][1], false); return;

}

for (const [category, channels] of sortedCategories) { const categoryDiv = document.createElement('div'); categoryDiv.className = 'category-item flex items-center space-x-3 p-3 cursor-pointer hover:bg-purple-500/20 transition-colors duration-200 border-b border-[var(--border-color)]';

const sortedIconKeys = Object.keys(categoryIcons).sort((a, b) => b.length - a.length); const iconKey = Object.keys(categoryIcons).find(key => key

=== category) || sortedIconKeys.find(key =>

category.toLowerCase().includes(key.toLowerCase())) || 'Default'; const iconHTML = categoryIcons[iconKey];

categoryDiv.innerHTML = `<div class="flex-shrink-0 w-6 h-6 flex items-center justify-center text-xl">${iconHTML}</div>

<span>${category} (${channels.length})</span>`; categoryDiv.addEventListener('click', () => { populatePlaylistUI(channels, true, () => displayCategories(groupedChannels));

});

elements.playlistElement.appendChild(categoryDiv);

}

} async function fetchMoviePoster(movieTitle) {

const details = await fetchMovieDetailsFromTMDB(movieTitle); return details ? details.logo : null;

}

async function fetchTvShowPoster(seriesName) { if (!TMDB\_API\_KEY) return null; const searchUrl =

`https://api.themoviedb.org/3/search/tv?api\_key=${TMDB\_API\_KEY}&query=

${encodeURIComponent(seriesName)}`; try { const response = await fetch(searchUrl); const data = await response.json(); if (data.results && data.results.length > 0 && data.results[0].poster\_path) { return

`https://image.tmdb.org/t/p/w200${data.results[0].poster\_path}`;

} return null; } catch (error) {

console.error('Error fetching TV show poster:', error); return null;

}

}

async function populatePlaylistUI(channels, showBackButton = false, backFunction = null) {

elements.playlistElement.innerHTML = ''; if (showBackButton) {

const backButton = document.createElement('button'); backButton.innerHTML = `<svg

xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0

24 24" fill="none" stroke="currentColor" stroke-width="2"

stroke-linecap="round" stroke-linejoin="round"><polyline points="15 18

9 12 15 6"></polyline></svg> Back`; backButton.className = 'back-button w-full flex items-center space-x-2 p-3 text-left text-purple-400 hover:bg-purple-500/20 border-b border-[var(--border-color)]'; backButton.addEventListener('click', backFunction || (()

=> loadDataForSource(elements.sourceSelectorText.textContent.trim()))); elements.playlistElement.appendChild(backButton);

} const listContainer = document.createElement('div'); listContainer.className = 'flex flex-col p-1'; elements.playlistElement.appendChild(listContainer);

channels.sort((a, b) => {

const numA = a.title.match(/\d+/g); const numB = b.title.match(/\d+/g);

if (numA && numB && numA.length > 1 && numB.length > 1) { if (parseInt(numA[0]) !== parseInt(numB[0])) return parseInt(numA[0]) - parseInt(numB[0]);

if (parseInt(numA[1]) !== parseInt(numB[1])) return parseInt(numA[1]) - parseInt(numB[1]);

}

return a.title.localeCompare(b.title);

});

for (const channel of channels) {

const channelDiv = document.createElement('div'); channelDiv.className = 'channel-item flex items-center space-x-3 p-2 cursor-pointer rounded-lg transition-all duration-200'; channelDiv.dataset.url = channel.url; const isFavorite = favorites.some(f => f.url === channel.url);

channelDiv.innerHTML = `

<div class="flex-shrink-0 w-12 h-12 bg-black/20 rounded-md flex items-center justify-center overflow-hidden"> <img src="${channel.logo || ''}" class="channel-logo w-full h-full object-cover" onerror="this.style.display='none'; this.nextElementSibling.style.display='flex';" onload="this.style.display='block';

this.nextElementSibling.style.display='none';">

<div class="channel-logo-fallback w-full h-full flex items-center justify-center text-xl font-bold text-slate-500" style="display: flex;">${(channel.title || 'L').charAt(0)}</div>

</div>

<span class="channel-title flex-1 text-sm text-slate-300">${channel.title || 'Loading...'}</span>

<button class="favorite-btn p-1 text-lg text-slate-600 hover:text-purple-400 transition-colors ${isFavorite ? 'favorite' :

''}" data-channel='${JSON.stringify(channel)}'>★</button>

`;

listContainer.appendChild(channelDiv);

}

} async function populateFeaturedChannels() { const featuredGrid = elements.featuredChannelsGrid; if (!featuredGrid) return; featuredGrid.innerHTML = `<div class="col-span-full flex justify-center items-center h-full"><div class="loader"></div></div>`;

if (masterAllChannels.length === 0) { await fetchAllChannelsForFeatured();

}

const totalFeatured = 12;

const liveTvPool = masterAllChannels.filter(c => c.sourcePlaylistName &&

!c.sourcePlaylistName.includes('Archive') &&

!c.sourcePlaylistName.includes('YT') &&

!c.sourcePlaylistName.includes('freemoviesplus') &&

c.url && !c.url.includes('archive.org')

).sort(() => 0.5 - Math.random());

const moviePool = masterAllChannels.filter(c => c.sourcePlaylistName &&

(c.sourcePlaylistName.includes('freemoviesplus') || (c.sourcePlaylistName.includes('Archive') && c.group && c.title && !c.title.match(/S\d{2}E\d{2}/i)))

).sort(() => 0.5 - Math.random());

const seriesPool = masterAllChannels.filter(c =>

c.sourcePlaylistName && c.title &&

(c.sourcePlaylistName.includes('YT Series') || (c.sourcePlaylistName.includes('Archive') && c.title.match(/S\d{2}E\d{2}/i)))

).sort(() => 0.5 - Math.random());

const selectedLive = liveTvPool.slice(0, Math.ceil(totalFeatured \* 0.5));

const selectedMovies = moviePool.slice(0, Math.ceil(totalFeatured \* 0.25));

const selectedSeries = seriesPool.slice(0,

Math.ceil(totalFeatured \* 0.25));

const selectedChannels = [...selectedLive, ...selectedMovies,

...selectedSeries].sort(() => 0.5 - Math.random()).slice(0, totalFeatured);

featuredGrid.innerHTML = ''; if (selectedChannels.length === 0) { featuredGrid.innerHTML = `<p class="col-span-full text-center text-[var(--text-muted)]">Could not load featured channels. Try another playlist.</p>`; return;

}

for (const channel of selectedChannels) { const caseDiv = document.createElement('div'); caseDiv.className = 'dvd-case';

caseDiv.dataset.channel = JSON.stringify(channel);

let logoUrl = channel.logo || '';

if (!logoUrl) { if (channel.group &&

(channel.sourcePlaylistName.includes('YT Series') || channel.sourcePlaylistName.includes('Archive'))) {

logoUrl = await fetchTvShowPoster(channel.group);

} else if (channel.title) {

logoUrl = await fetchMoviePoster(channel.title);

}

}

caseDiv.innerHTML = `

<div class="dvd-case-img-container">

<img src="${logoUrl || ''}" class="dvd-case-img" onerror="this.style.display='none';

this.nextElementSibling.style.display='flex'">

<div class="channel-logo-fallback w-full h-full flex items-center justify-center text-2xl font-bold text-slate-500" style="${logoUrl ? 'display:none;' : ''}">${(channel.title ||

'L').charAt(0)}</div>

</div>

<div class="dvd-case-title">${channel.title ||

'Unknown'}</div>

`;

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

const clickedChannel = JSON.parse(caseDiv.dataset.channel); loadStream(clickedChannel.url); goToChannel(clickedChannel);

});

featuredGrid.appendChild(caseDiv);

}

} async function goToChannel(channel) { document.querySelector('main').scrollTo({ top: 0, behavior:

'smooth' });

const sourcePlaylist = allPlaylists.find(p => p.name === channel.sourcePlaylistName); if (!sourcePlaylist) return;

updateSourceSelectorButton(sourcePlaylist); await loadDataForSource(sourcePlaylist.name);

setTimeout(() => {

document.querySelectorAll('.channel-item').forEach(item => item.classList.remove('active')); const targetChannelElement =

document.querySelector(`.channel-item[data-url="${CSS.escape(channel.u rl)}"]`);

if (targetChannelElement) {

targetChannelElement.classList.add('active'); targetChannelElement.scrollIntoView({ behavior:

'smooth', block: 'center' });

}

}, 500);

} function loadStream(url) { channelClickCounter++;

if (channelClickCounter > 0 && channelClickCounter % 10 === 0) {

window.open('https://www.profitableratecpm.com/qb069j7b6?key=e6b8dcd0a

21df19017732330d1236fd3', '\_blank'); urlToPlayAfterAd = url; showAd(); } else {

playStream(url);

}

} function playStream(url) { hideErrorOverlay(); currentlyPlayingUrl = url; const videoId = extractVideoID(url);

const allKnownChannels = [...masterAllChannels, ...favorites]; const currentChannel = allKnownChannels.find(c => c.url === url); elements.nowPlayingTitle.textContent = currentChannel ? currentChannel.title : 'Loading...';

if (ytPlayer && typeof ytPlayer.stopVideo === 'function') ytPlayer.stopVideo();

elements.mainPlayer.pause();

elements.mainPlayer.removeAttribute('src'); if (hls) { hls.destroy(); hls = null;

}

if (videoId) {

activePlayer = 'youtube';

elements.mainPlayer.classList.add('hidden');

elements.youtubePlayerContainer.classList.remove('hidden'); if (ytPlayer && typeof ytPlayer.loadVideoById ===

'function') {

ytPlayer.loadVideoById(videoId);

} else {

initializeYouTubePlayer();

} } else {

elements.youtubePlayerContainer.classList.add('hidden'); elements.mainPlayer.classList.remove('hidden');

if (url.endsWith('.m3u8') && Hls.isSupported()) { activePlayer = 'hls';

hls = new Hls({ fragLoadingMaxRetry: 4, manifestLoadingMaxRetry: 1 });

hls.on(Hls.Events.ERROR, (event, data) => { if (data.fatal) {

console.error('HLS Error:', data); showErrorOverlay();

elements.nowPlayingTitle.textContent = 'Error loading stream'; } });

hls.attachMedia(elements.mainPlayer); hls.loadSource(url);

hls.once(Hls.Events.MANIFEST\_PARSED, () => elements.mainPlayer.play().catch(e => console.error("HLS Playback error:", e))); } else { activePlayer = 'native'; elements.mainPlayer.src = url; elements.mainPlayer.play().catch(e => { console.error("Native playback error:", e); showErrorOverlay();

elements.nowPlayingTitle.textContent = 'Error loading stream'; });

} }

localStorage.setItem('iptv\_last\_played', url);

}

function toggleFavorite(button) { const channelData = JSON.parse(button.dataset.channel); const index = favorites.findIndex(f => f.url === channelData.url); if (index !== -1) { favorites.splice(index, 1); button.classList.remove('favorite');

} else {

favorites.push(channelData); button.classList.add('favorite');

}

localStorage.setItem('iptv\_favorites',

JSON.stringify(favorites)); if

(elements.sourceSelectorText.textContent.trim().includes('Favorites'))

{

loadDataForSource('Favorites');

}

}

function enterFullscreen() {

if (activePlayer === 'hls' || activePlayer === 'native') { const playerElement = elements.mainPlayer; if (playerElement.requestFullscreen) playerElement.requestFullscreen();

else if (playerElement.mozRequestFullScreen) playerElement.mozRequestFullScreen();

else if (playerElement.webkitRequestFullscreen) playerElement.webkitRequestFullscreen();

else if (playerElement.msRequestFullscreen) playerElement.msRequestFullscreen();

}

} function updateSourceSelectorButton(playlist) { elements.sourceSelectorIcon.innerHTML = `<span class="text-xl">${playlist.icon || '★'}</span>`;

elements.sourceSelectorText.textContent = playlist.name;

};

function setupAllPlaylists() {

const favoritesPlaylist = { name: 'Favorites', icon: '⭐'}; customPlaylists =

JSON.parse(localStorage.getItem('iptv\_custom\_playlists')) || []; allPlaylists = [favoritesPlaylist, ...customPlaylists,

...BASE\_PLAYLISTS].sort((a,b) => {

if (a.name === 'Favorites') return -1; if (b.name === 'Favorites') return 1; return a.name.localeCompare(b.name);

});

} function renderPlaylistOptions(playlistsToRender) { elements.playlistOptionsContainer.innerHTML = ''; playlistsToRender.forEach(playlist => { const optionDiv = document.createElement('div'); optionDiv.className = 'flex items-center space-x-2 p-2 cursor-pointer hover:bg-purple-500/20'; optionDiv.innerHTML = `<span class="flex-shrink-0 w-6 text-center text-xl">${playlist.icon || '📁'}</span> <span class="truncate">${playlist.name}</span>`; optionDiv.addEventListener('click', () => { localStorage.setItem('iptv\_last\_source', playlist.name); updateSourceSelectorButton(playlist); loadDataForSource(playlist.name);

elements.sourceSelectorDropdown.classList.add('hidden');

});

elements.playlistOptionsContainer.appendChild(optionDiv);

});

} async function fetchAllChannelsForFeatured() { if (masterAllChannels.length > 0) return;

const playlistsToFetch = [...customPlaylists,

...BASE\_PLAYLISTS];

const promises = playlistsToFetch.map(async (playlist) => { if (playlist.isNested) {

function flatten(structure, name) { let channels = []; for (const key in structure) { if (Array.isArray(structure[key])) { channels.push(...structure[key].map(c =>

({...c, sourcePlaylistName: name})));

} else if (typeof structure[key] === 'object') {

channels.push(...flatten(structure[key], name));

} }

return channels;

}

return flatten(playlist.structure, playlist.name);

}

if (playlist.channels) {

return playlist.channels.map(c => ({...c, sourcePlaylistName: playlist.name}));

} try {

const data = await fetchWithFallback(playlist.url); const { channels } = await parseM3U(data, playlist.name); return channels; } catch(e) {

console.warn(`Could not fetch ${playlist.name} for featured list.`); return [];

}

});

const results = await Promise.allSettled(promises);

masterAllChannels = results

.filter(result => result.status === 'fulfilled')

.flatMap(result => result.value);

}

function cleanMovieTitle(filename) { let title = filename;

const yearMatch = title.match(/[\[\(.\s](\d{4})[\]\).\s]/); const year = yearMatch ? yearMatch[1] : null;

title = title.replace(/\.(mp4|mkv|avi|mov|wmv)$/i, ''); title = title.replace(/[\[\(].\*?[\]\)]/g, ''); title =

title.replace(/(1080p|720p|480p|2160p|4K|UHD|BluRay|WEB-DL|WEBRip|HDTV

|x264|x265|AAC|DTS|HDRip|EXTENDED|REMASTERED|DIRECTORS CUT|DUAL AUDIO|MULTI-LANG).\*/i, '');

title = title.replace(/\./g, ' ').trim();

if (year) {

title = title.replace(year, ''); }

return { title: title.trim(), year };

}

async function fetchMovieDetailsFromTMDB(originalTitle) { if (!TMDB\_API\_KEY) return null;

const cacheKey = `tmdb\_movie\_${originalTitle}`; const cached = localStorage.getItem(cacheKey); if (cached) {

const { timestamp, data } = JSON.parse(cached); if (Date.now() - timestamp < (24 \* 60 \* 60 \* 1000)) { //

24-hour cache

return data;

}

}

const { title: cleanedTitle, year } = cleanMovieTitle(originalTitle);

if (!cleanedTitle) return null;

let searchUrl =

`https://api.themoviedb.org/3/search/movie?api\_key=${TMDB\_API\_KEY}&que ry=${encodeURIComponent(cleanedTitle)}`; if (year) {

searchUrl += `&primary\_release\_year=${year}`;

}

try { const response = await fetch(searchUrl); const data = await response.json(); if (data.results && data.results.length > 0) { const movie = data.results[0]; const genre = movie.genre\_ids.length > 0 ? TMDB\_GENRE\_MAP[movie.genre\_ids[0]] : 'Uncategorized'; const logo = movie.poster\_path ?

`https://image.tmdb.org/t/p/w200${movie.poster\_path}` : null; const result = { genre, logo, title: movie.title }; // Return the official title

localStorage.setItem(cacheKey, JSON.stringify({ timestamp: Date.now(), data: result })); return result;

} return null; } catch (error) {

console.error('Error fetching from TMDB:', error);

return null;

}

} async function injectRemotePlaylists() { const targetPlaylistName = 'Internet Archive 📺🎬'; const targetPlaylist = BASE\_PLAYLISTS.find(p => p.name === targetPlaylistName);

if (!targetPlaylist || !targetPlaylist.isNested) return;

// --- Inject Series --- const seriesPlaylistsToInject = [ { url:

'https://raw.githubusercontent.com/ab19902020/King/refs/heads/main/Kin gofhill.m3u', group: 'King of The Hill', path: ['Series', 'Animation'] },

{ url:

'https://raw.githubusercontent.com/ab19902020/Sp/refs/heads/main/Sp.m3 u', group: 'South Park', path: ['Series', 'Animation'] }, { url:

'https://raw.githubusercontent.com/ab19902020/Comedy/refs/heads/main/C omedy.m3u', group: 'The Inbetweeners', path: ['Series', 'Comedy'] }, { url:

'https://raw.githubusercontent.com/ab19902020/Comedy/refs/heads/main/F aultytowers.m3u', group: 'Fawlty Towers', path: ['Series', 'Comedy'] }

];

for (const remote of seriesPlaylistsToInject) { try {

// Fetch the poster for the entire series first const seriesPoster = await fetchTvShowPoster(remote.group); const data = await fetchWithFallback(remote.url); const { channels } = await parseM3U(data, targetPlaylistName, remote.group);

// Add the fetched poster to each episode/channel const processedChannels = channels.map(c => ({

...c,

logo: seriesPoster || c.logo, // Use series poster, fallback to M3U logo if any

sourcePlaylistName: targetPlaylistName

}));

let category = targetPlaylist.structure;

remote.path.forEach(p => { category = category?.[p];

});

if (category && category[remote.group]) {

category[remote.group].push(...processedChannels);

} } catch (e) {

console.error(`Failed to inject remote series playlist from ${remote.url}`, e);

}

}

// --- Inject and Categorize Movies --- const moviesPlaylistUrl =

'https://raw.githubusercontent.com/ab19902020/Comedy/refs/heads/main/M ovies.m3u'; try { const data = await fetchWithFallback(moviesPlaylistUrl); const { channels: movieChannels } = await parseM3U(data, targetPlaylistName);

const movieDetailPromises = movieChannels.map(async

(channel) => { const details = await fetchMovieDetailsFromTMDB(channel.title); if (details) { channel.group = details.genre || 'Uncategorized'; channel.logo = details.logo || channel.logo; channel.title = details.title; // Use the official title from TMDB } else {

channel.group = 'Uncategorized';

}

return channel;

});

const updatedMovieChannels = await Promise.all(movieDetailPromises);

const moviesCategory = targetPlaylist.structure.Movies; if (moviesCategory) { if (updatedMovieChannels.length > 0) { moviesCategory['All Movies'] = updatedMovieChannels; }

updatedMovieChannels.forEach(channel => { const genre = channel.group; if (genre !== 'Uncategorized') { if (!moviesCategory[genre]) { moviesCategory[genre] = [];

}

moviesCategory[genre].push(channel);

}

});

} } catch (e) {

console.error(`Failed to inject remote movies playlist from ${moviesPlaylistUrl}`, e);

}

}

async function initialize() { try {

await injectRemotePlaylists();

} catch (e) {

console.error("Initialization failed during remote playlist injection:", e);

} initializePlayers(); setupAllPlaylists();

renderPlaylistOptions(allPlaylists);

fetchAllChannelsForFeatured().then(populateFeaturedChannels);

// --- Event Listeners ---

elements.sourceSelectorButton.addEventListener('click', () => {

elements.sourceSelectorDropdown.classList.toggle('hidden'); if

(!elements.sourceSelectorDropdown.classList.contains('hidden')) { const currentPlaylistName = elements.sourceSelectorText.textContent.trim(); const options =

Array.from(elements.playlistOptionsContainer.children); const selectedOption = options.find(opt => opt.textContent.includes(currentPlaylistName)); if (selectedOption) {

selectedOption.scrollIntoView({ block: 'nearest'

});

}

}

});

document.addEventListener('click', (e) => {

if (!elements.sourceSelectorButton.contains(e.target) &&

!elements.sourceSelectorDropdown.contains(e.target)) { elements.sourceSelectorDropdown.classList.add('hidden');

}

});

elements.playlistSearchInput.addEventListener('click', (e) => e.stopPropagation()); elements.playlistSearchInput.addEventListener('input', e => { const searchTerm = e.target.value.toLowerCase().trim(); const filtered = allPlaylists.filter(p => p.name.toLowerCase().includes(searchTerm)); renderPlaylistOptions(filtered);

});

elements.searchInput.addEventListener('input', e => { const searchTerm = e.target.value.toLowerCase().trim(); let channelsToSearch = allChannelsInPlaylist;

if (!searchTerm) {

loadDataForSource(elements.sourceSelectorText.textContent.trim()); return;

}

const filteredChannels = channelsToSearch.filter(c => c.title && c.title.toLowerCase().includes(searchTerm)); populatePlaylistUI(filteredChannels, true);

});

elements.addPlaylistBtn.addEventListener('click', () => elements.addPlaylistModal.classList.remove('hidden'));

elements.cancelAddPlaylistBtn.addEventListener('click', () => elements.addPlaylistModal.classList.add('hidden'));

elements.savePlaylistBtn.addEventListener('click', () => { const name = elements.newPlaylistName.value.trim(); const url = elements.newPlaylistUrl.value.trim(); if(name && url) { customPlaylists.push({name, url, icon: '📁', isCustom:

true}); localStorage.setItem('iptv\_custom\_playlists', JSON.stringify(customPlaylists)); setupAllPlaylists();

renderPlaylistOptions(allPlaylists); elements.newPlaylistName.value = ''; elements.newPlaylistUrl.value = '';

elements.addPlaylistModal.classList.add('hidden');

} else {

alert("Please provide both a name and a valid URL.");

}

});

// Modal Listeners

elements.aboutBtn.addEventListener('click', () => elements.dmcaModal.classList.remove('hidden'));

elements.closeDmcaBtn.addEventListener('click', () => elements.dmcaModal.classList.add('hidden'));

elements.dmcaModal.addEventListener('click', (e) => { if (e.target === elements.dmcaModal) { elements.dmcaModal.classList.add('hidden');

}

});

elements.closeAdBtn.addEventListener('click', hideAdAndOpenLink);

elements.adImage.addEventListener('click', hideAdAndOpenLink); elements.playlistElement.addEventListener('click', handlePlaylistClick);

elements.menuToggle.addEventListener('click', () => { elements.mobileOverlay.classList.toggle('hidden'); elements.sidebar.classList.toggle('-translate-x-full');

}); elements.mobileOverlay.addEventListener('click', () => { elements.sidebar.classList.add('-translate-x-full'); elements.mobileOverlay.classList.add('hidden');

});

const lastSourceName =

localStorage.getItem('iptv\_last\_source') || 'Internet Archive 📺🎬'; const lastSourcePlaylist = allPlaylists.find(p => p.name === lastSourceName) || allPlaylists[0]; if (lastSourcePlaylist) {

updateSourceSelectorButton(lastSourcePlaylist); loadDataForSource(lastSourceName);

}

if (!sessionStorage.getItem('adShownThisSession')) { showAd(true);

sessionStorage.setItem('adShownThisSession', 'true');

}

} function handlePlaylistClick(event) {

const favoriteBtn = event.target.closest('.favorite-btn'); if (favoriteBtn) {

event.stopPropagation(); toggleFavorite(favoriteBtn); return;

}

const channelItem = event.target.closest('.channel-item'); if (channelItem) { const currentUrl = channelItem.dataset.url; if (currentUrl === lastClickedUrl) { enterFullscreen(); lastClickedUrl = null;

} else {

document.querySelectorAll('.channel-item').forEach(item => item.classList.remove('active'));

channelItem.classList.add('active'); loadStream(currentUrl); lastClickedUrl = currentUrl;

}

}

}

initialize();

});

</script>

</body>

</html>

I've applied the fix to the injectRemotePlaylists function. It now fetches the main poster for each TV series and applies it to all of its episodes, so they should display correctly in the sidebar list. Let me know if you have any other questions!