WDD 330 Final Project Proposal

StreamFinder: Movie & TV Discovery Hub

Aghadiaye Ernest Amayanvbo | January 2026

1. Overview

StreamFinder is a web application that helps users discover movies and TV shows without the endless scrolling. The problem is simple: I spend way too much time on Netflix trying to find something to watch. My family is even worse - we can never agree on anything, and by the time we pick something, half the evening is gone.

The solution is a single place to search, filter, and explore content from multiple sources. StreamFinder pulls data from TMDB (The Movie Database) and TVMaze, giving users access to detailed information about movies and TV shows in one clean interface. Users can search, filter by genre or rating, save favorites to a watchlist, and even hit a "Surprise Me" button when they just can't decide.

This project gives me the chance to work with real APIs, build something I'd actually use, and put together a polished frontend application using vanilla JavaScript.

2. Target Audience

StreamFinder is built for:

- **Casual viewers** who just want to find something good without spending 30 minutes browsing
- **Families or roommates** who need help agreeing on what to watch together
- **Movie and TV enthusiasts** who like to track what they want to watch and explore by genre
- **People with multiple streaming subscriptions** who want one place to discover content instead of jumping between apps

3. Major Functions

1. Multi-Source Search System

Users can search for movies and TV shows using a single search bar. The app queries both TMDB and TVMaze simultaneously, combines the results, and displays them in a unified grid. Smart autocomplete provides live suggestions as users type, and debounced requests prevent unnecessary API calls.

2. Advanced Filtering & Sorting

Users can narrow down results using filters for genre, release year, minimum rating, and content type (movie or TV). Results can be sorted by popularity, rating, or release date. A combined ranking algorithm weighs data from both APIs to surface the best matches.

3. Trending & Discovery

The homepage features a trending section showing what's popular right now. Users can also browse genre-specific discovery pages and view upcoming releases including new movies and next episode air dates for TV shows.

4. Detailed Content View

Clicking on any title opens a detailed view with the full synopsis, high-quality poster, release date, runtime or episode count, genres, ratings, and language. A cast preview shows the top actors, and a "Similar Titles" section recommends related content.

5. TV Episode Guide

For TV shows, users can browse complete season and episode listings. Data is pulled from TVMaze's episode endpoints, displaying episode titles, air dates, and summaries. This gives users a full picture of a show's structure before committing.

6. Personal Watchlist

Users can save movies and TV shows to a personal watchlist stored in localStorage. Items can be added or removed with a single click, and the watchlist persists across browser sessions. Hover actions on posters provide quick access to add/remove functionality.

7. "Surprise Me" Random Picker

For users who can't decide, the "Surprise Me" feature randomly selects a movie or TV show. Users can optionally filter by genre or content type before rolling the dice. The result opens directly in the detailed view.

8. User Preferences & State

The app remembers user preferences including recent searches, active filters, and theme selection. These are stored in localStorage so the experience feels consistent across sessions.

9. Responsive Design & Animations

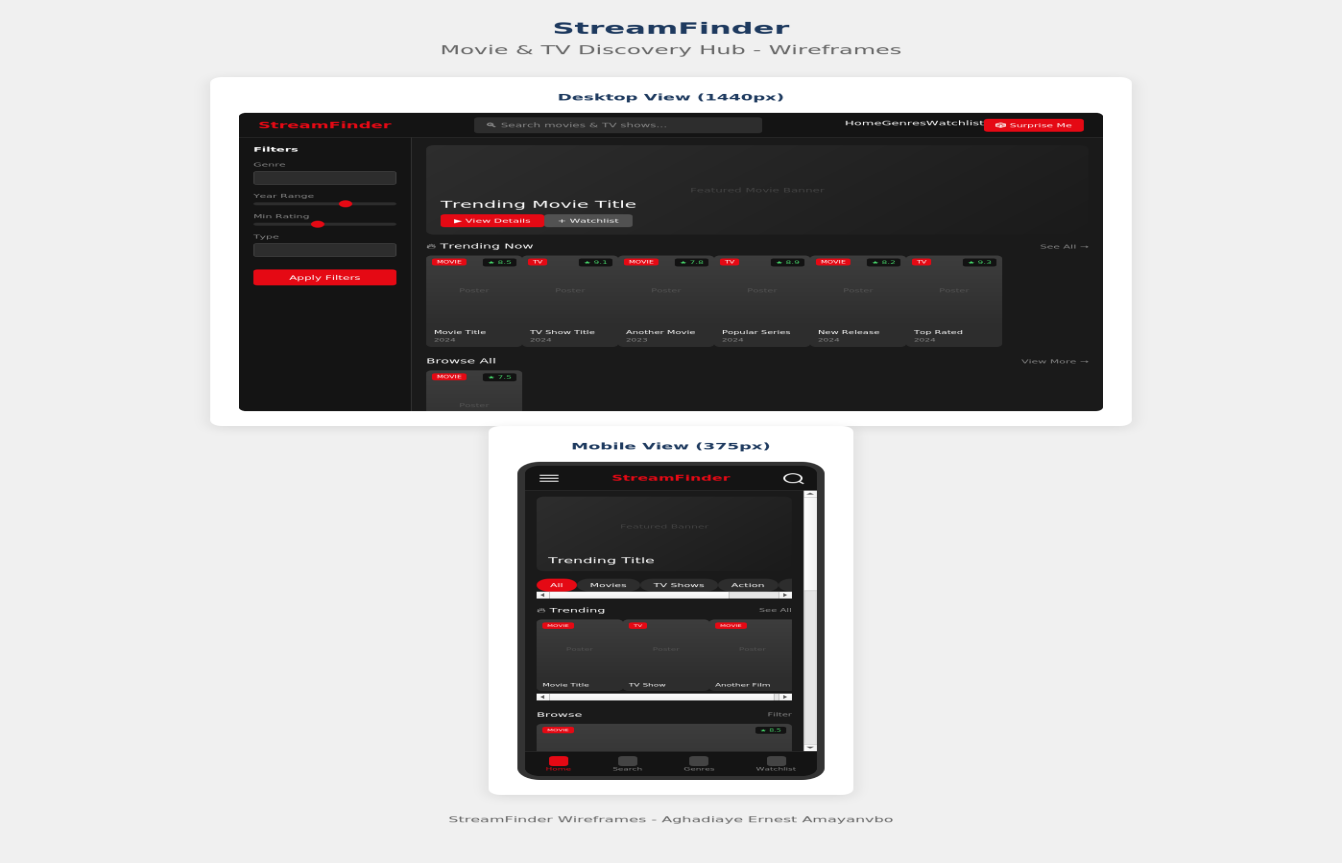
The interface is mobile-first and adapts smoothly to any screen size using CSS Grid and Flexbox. Skeleton loaders provide visual feedback during data fetches. CSS animations handle transitions, hover effects, and modal interactions for a polished feel.

10. Accessibility & Error Handling

Full keyboard navigation support allows users to browse without a mouse. ARIA labels and alt text ensure screen reader compatibility. Friendly error messages handle API failures, and empty states guide users when no results are found.

4. Wireframes

The following wireframes illustrate the application layout for both desktop and mobile views:



Wireframes include:

- **Desktop Home Page:** Header with logo and search bar, trending carousel, content grid layout, sidebar with filters
- **Mobile Home Page:** Hamburger menu, stacked vertical layout, touch-friendly cards, bottom navigation
- **Detail Modal/Page:** Large poster image, metadata section, cast preview, episode guide (for TV), similar titles carousel

5. External Data Sources

API 1: TMDB (The Movie Database)

Endpoint: https://api.themoviedb.org/3/

Data attributes returned: title, overview, poster_path, release_date, vote_average, vote_count, genre_ids, popularity, original_language, backdrop_path, adult, video

Used for: Movie data, trending content, search, images, cast info

API 2: TVMaze

Endpoint: https://api.tvmaze.com/

Data attributes returned: name, summary, image, premiered, rating, genres, network, schedule, runtime, status, language, episodes

Used for: TV show data, episode guides, air schedules

Local Storage:

• Watchlist items (array of saved titles with IDs and source)

• Recent searches (array of search terms)

• User preferences (theme, active filters)

6. Module List

HTML Files:

• index.html - Home page with search and trending content

• search.html - Search results page with filters

• watchlist.html - User's saved movies and shows

• genre.html - Genre discovery page

CSS Files:

• css/main.css - Global styles, CSS variables, reset

• css/components.css - Cards, modals, buttons, forms

• css/animations.css - Transitions, keyframes, loaders

• css/responsive.css - Media queries for all breakpoints

JavaScript Modules (ES6):

• js/main.js - App initialization and event setup

• js/api/tmdb.js - TMDB API fetch functions

• js/api/tvmaze.js - TVMaze API fetch functions

• js/components/search.js - Search bar and autocomplete

• js/components/filters.js - Filter UI and logic

• js/components/modal.js - Detail view modal handling

• js/components/cards.js - Content card rendering

• js/utils/storage.js - localStorage helper functions

• js/utils/helpers.js - Utility and formatting functions

Data Files:

• data/genres.json - Genre ID to name mappings

7. Graphic Identity

Color Scheme

Role	Color Name	Hex Value
Background	Rich Black	#141414
Surface/Cards	Dark Gray	#1F1F1F
Primary Accent	Netflix Red	#E50914
Text Primary	White	#FFFFFF
Text Muted	Gray	#808080
Success State	Green	#46D369

Typography

- **Headings:** Bebas Neue - bold, cinematic display font
- **Body Text:** Inter - clean, highly readable sans-serif
- **Fallback Stack:** Arial, Helvetica, sans-serif

Application Icon

A play button integrated with a magnifying glass, symbolizing "find something to watch." The icon uses the red accent color (#E50914) on a dark background (#141414), maintaining the Netflix-inspired aesthetic. The design is simple enough to work as a favicon while remaining recognizable at larger sizes.

8. Timeline (Weeks 5-7)

Week 5: Foundation

- Set up project folder structure and file organization
- Build HTML pages with semantic markup
- Implement base CSS styles with dark theme variables
- Connect to TMDB API and display trending content

- Create responsive grid layout for content cards
- Basic search functionality working

Week 6: Core Features

- Implement search with autocomplete and debouncing
- Build filter system (genre, year, rating, type)
- Create detail modal with full content information
- Integrate TVMaze API for TV episode data
- Implement watchlist with localStorage persistence
- Add "Surprise Me" random picker feature

Week 7: Polish & Deploy

- Implement skeleton loaders and loading states
- Add CSS animations and transitions
- Build error states and empty states
- Accessibility audit (keyboard nav, ARIA labels)
- Cross-device and cross-browser testing
- Deploy to GitHub Pages or Netlify
- Final code cleanup and documentation

9. Project Planning (Trello Board)

Trello Board: https://trello.com/b/697a76d68cd5d675a529aba6/streamfinder

The Trello board is organized with the following columns: Backlog, To Do (This Week), In Progress, Testing, and Done. All tasks from the timeline have been broken down into individual cards with descriptions and checklists where appropriate.

10. Anticipated Challenges

Merging Data from Two Different APIs

TMDB and TVMaze structure their data completely differently. I'll need to create a unified data format that works for both sources. Handling cases where the same show exists in both databases without showing duplicates will require some careful logic.

Rate Limiting and API Performance

Both APIs have rate limits that I need to respect. Implementing proper debouncing on search inputs and potentially caching responses will be important to keep the app fast and avoid hitting those limits.

State Management Without a Framework

Without React or Vue handling state for me, I'll need to manage things like filter selections, watchlist data, and UI states manually. Keeping this organized and bug-free with vanilla JavaScript will take careful planning.

Responsive Layout Complexity

The app has several different layouts - the main grid, the detail modal, episode guides, filter sidebar - that all need to work well on phones, tablets, and desktops. Testing across all these screen sizes will be time-consuming.

Scope Creep and Time Management

With 10 major features planned, there's a real risk of spending too long perfecting one area and running out of time for others. I'll need to prioritize the core search and display functionality first, then add polish features only if time allows.

This proposal outlines a plan to build a functional, polished web application using HTML, CSS, and vanilla JavaScript. The project demonstrates proficiency in API integration, responsive design, state management, and modern frontend development practices.