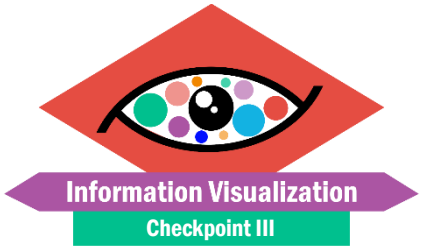
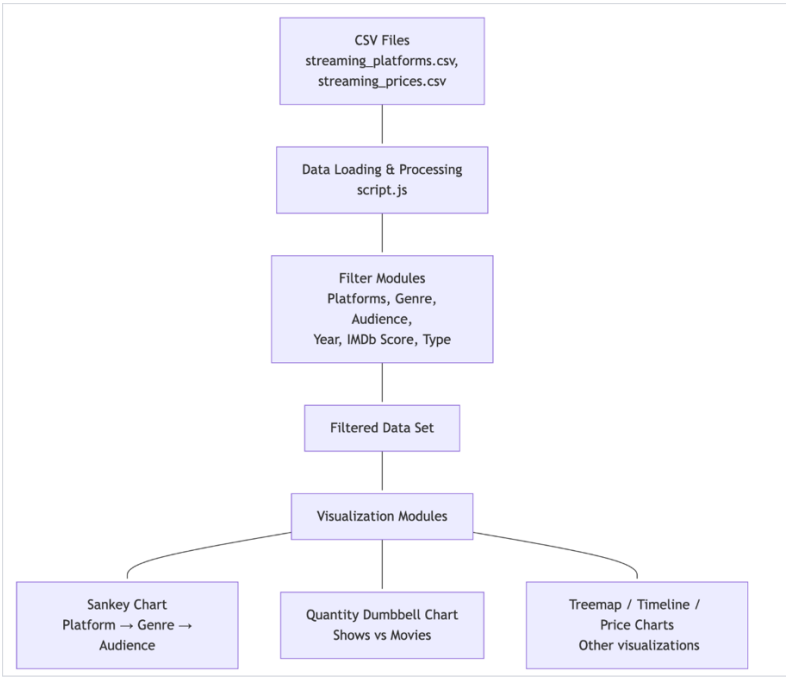


Checkpoint III: First Prototype

Group: G20
Date: 2025/09/27

Prototype Architecture



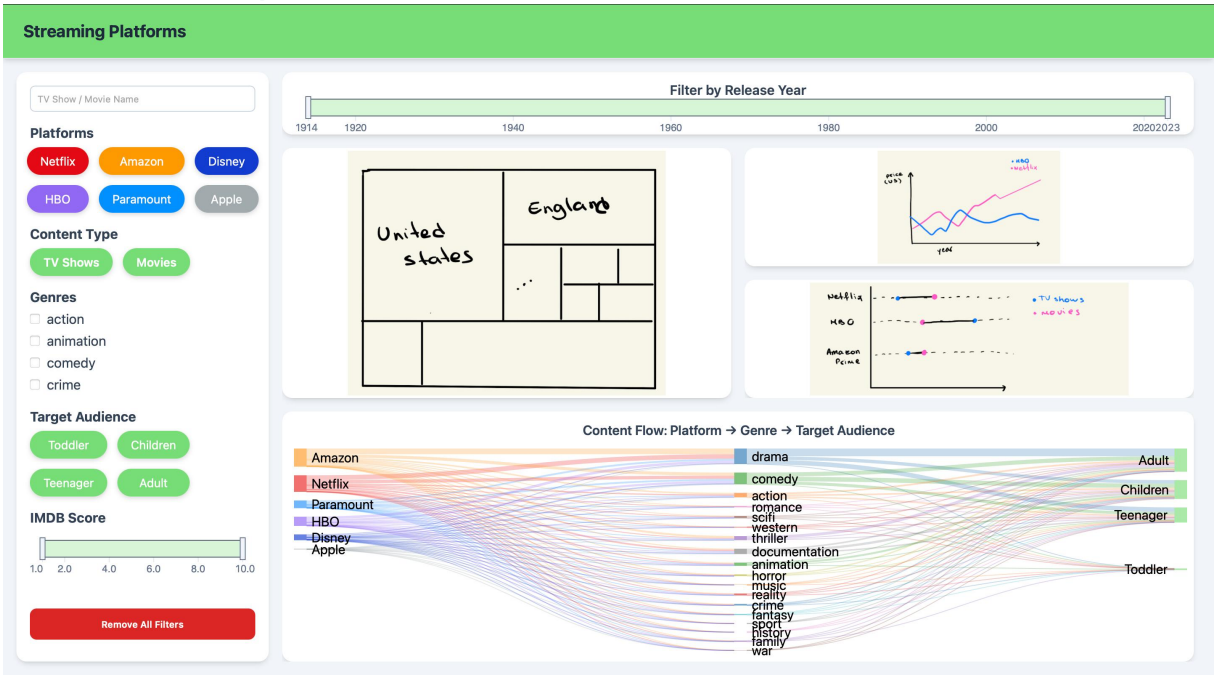
The project is organized in a simple manner; the page structure is in an index.html file, the design and layout are in a style.css file, and all the logic and interactivity are handled by a single script.js file.

When the page is open, it first loads the necessary data files (*streaming_platforms.csv and streaming_prices.csv*) while a spinner is displayed. The raw data is immediately cleaned up by converting text into numbers and assigning a *primary genre* to each item. This clean, master copy of the data is then kept ready for use.

For the interaction, the object currentFilters keeps a list of all the current filter settings (e.g., 'Netflix' is selected, IMDb score is 7.5-9.0, etc.). When you click a filter, the code doesn't immediately redraw the charts. Instead, it just updates the currentFilters object (e.g., "the user just checked the 'Horror' box"). After updating the object, it calls the function *applyFilters*, which is in charge of all updates. This function looks at the complete list of filters, takes the original master dataset, and creates a new, temporary list of data that matches *all* of the active filters. Finally, the function sends this freshly filtered data to each chart's drawing function (*renderSankeyChart*), telling them to redraw themselves with the new information.

Because everything is organized this way, the system is very easy to expand. To add a new chart, we just need to create its drawing instructions and include it in the function *renderAllVisualizations*. That way, the *applyFilters* function will also call it with the new filtered data, too. The new chart will automatically work with all the filters we've already built.

Dashboard Layout



The dashboard's layout is organized into two primary, distinct sections to create an intuitive user experience: **The Interactive Filter Panel (Left Column)** and **The Visualization Canvas (Right Column)**. The Interactive Filter Panel is the fixed-width panel on the left containing all the controls for manipulating the data. You have the implemented features, and at the bottom, there is a button to remove all filters and return to the original state. The implemented filters include: Platform Selection (Netflix, Amazon, etc.), Content Type (TV Shows/Movies or both), Genre Selection, Target Audience and IMDb Score (Range Slider). The Visualization Canvas is the main area where the data is displayed. It is organized into a grid to organize the various charts such as the Release Year Filter which is a full-width range slider at the top with a range of all possible years and the Alluvial Diagram the visualization showing the flow from platforms to genres and audiences.

Data Processing

Data preprocessing is critical to ensure that the Alluvial diagram and filters work consistently.

General Preprocessing

For the general preprocessing first, we string fields such as release_year and imdb_score are converted into numerical values and missing values in the genre or age_category fields are replaced with "Unknown". Then the main_genre field is derived from the first genre listed in the genre column and the categorical values, including type (TV SHOW / MOVIE) and age_category, are standardized for consistency.

Alluvial diagram

Nodes are created for each **platform, genre, and audience category** and links are generated in two stages:

- Platform → Genre: counts how many titles each platform has in each genre.
- Genre → Audience: counts how many titles in a genre belong to each audience category.

The Sankey algorithm (d3.sankey) computes node positions and link widths proportionally to these weights and platform-specific colors (matching from filters) and genre-based color scales are assigned.

Timeline Filter

For this filter we first determine [minYear, maxYear] using d3.extent and pass the domain into the slider component for interaction.

IMDB Score Slider

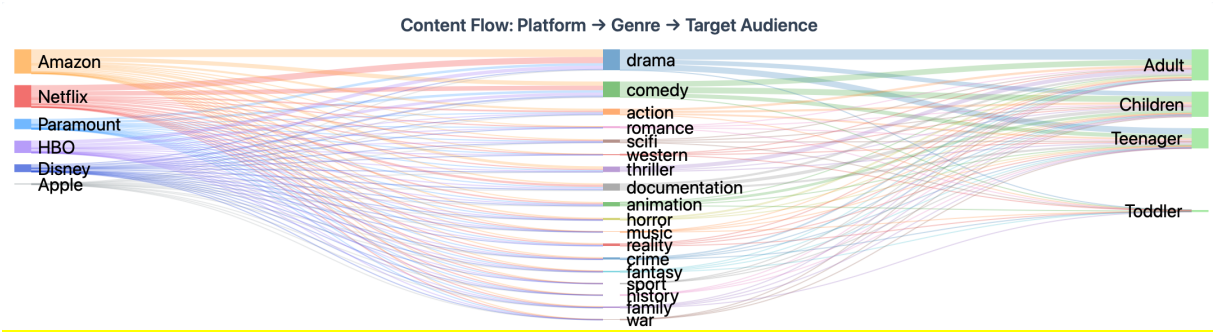
Here we fix the domain [1.0,10.0] and round to one decimal place. currentFilters is updated whenever the selection changes.

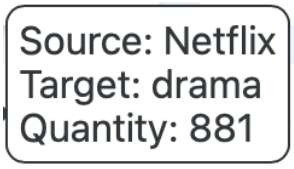
Genre Filter

We extract unique values from the dataset and sort these alphabetically.

Chart Interaction

The Alluvial Diagram



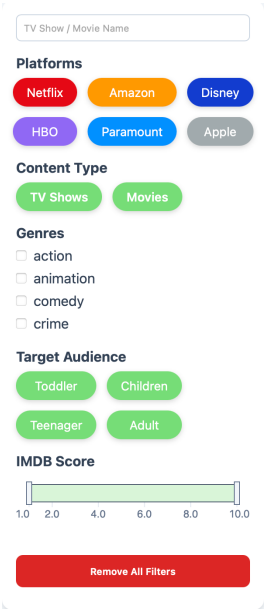


Description: The Alluvial Diagram shows the flow of content across three distinct categorical stages: from the **Platform**, through the primary **Genre**, to the final **Target Audience**. The width of each link is directly proportional to the volume of content, making it easy to spot the dominant trends and content strategies of each platform.

Interactivity: The diagram supports interaction in two ways:

1. **Global Filtering:** It is fully linked to the filter panel. Any selection made in the filters (e.g., selecting "Disney" or filtering by year) causes the diagram to instantly redraw, showing the new flows for only the data that matches the user's wants.
2. **Direct Inspection (Tooltips):** The diagram also supports direct interaction on its flow lines. When a user hovers their mouse over any link or node, a **tooltip** appears. This tooltip displays the exact number of content items in that specific flow (e.g., "Netflix → Drama: 881 items"). This allows for a more detailed data inspection, supplementing the visual width of the line with a precise value

The Interactive Filter



Description: The entire left and top panels are interactive filters. This includes multi-select buttons (Platforms, Audience and Content Type), a checkbox list (Genres), and two advanced range sliders (IMDb Score, Release Year).

Interactivity:

- **Direct Selection:** Clicking on buttons or checkboxes adds or removes categories from the active filter, allowing for complex, multi-select queries.
- **Range Selection:** The IMDb and Year sliders allow users to define a precise **interval** for filtering by dragging two handles. All charts update on the "end" of the drag event to reflect the selected range.

Chart Integration

All visualizations in the dashboard are linked using a **centralized, state-driven filtering model**. The mechanism works through three key components:

1. The object currentFilters holds the current settings for every filter (e.g., selected platforms, IMDB score, etc).
2. When a user interacts with a filter control, for example, when clicking a button or dragging a slider, the code only needs to update the central state object. The filter controls are "unaware" of the charts and do not communicate with them directly.
3. After the state object is updated, the function applyFilters is called. This function reads the *entire* state from the currentFilters object, filters the original master dataset according to all active criteria, and then passes this newly filtered data to a renderAllVisualizations function, which in turn calls the individual drawing function for every chart.

This decoupled architecture is precisely why **adding more charts is easy**. To integrate a new, fully interactive visualization, a developer only needs to create a new, self-contained rendering function for the chart (this function only needs to know how to draw itself based on whatever data it receives) and add the new funtion to the central renderAllVisualizations function. This means that the applyFilters function doesn't need to know what charts exist; it only knows that it must prepare the data and pass it along to be rendered.