Video Catalog

Description

Create a Next JS app to fetch movies using the OMDB API. The app will define a single component MovieList and maintain state within the component. Described below are 3 main features

1. Pagination to move back and forth between search result pages
2. Search input to search video using a key
3. Movie type selector – OMDB define 3 types for videos – movie, series and episode – you should provide for filtering based on these; or no filtering if user selects ALL (see mocks below)

http://www.omdbapi.com/

You will need to generate an API key by using
http://www.omdbapi.com/apikey.aspx

Note: There is a limit of 1000 hits on daily usage for free tier.

For making Ajax calls please use either **axios** or **Fetch API**.
https://www.npmjs.com/package/axios

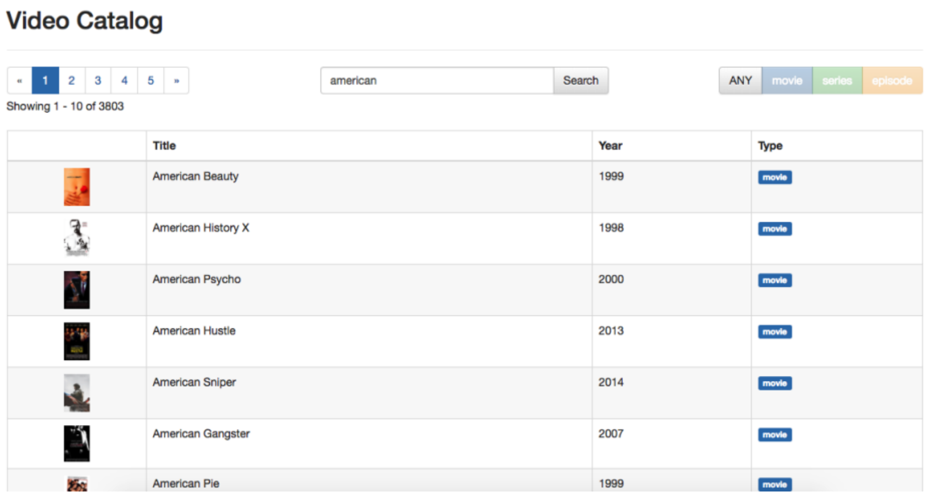
Use Redux (using Redux Toolkit along with React Redux package) to store state of the application.

Please stick to the following specifications

- The state shall be stored in Redux. The first page loaded will be page 1 with search being 'american', and no filtering based on type.
- Each time user searches by key (e.g. Indian), the page should be set to 1 when call for new page is made (with new search key)
- Each time user changes video type, the page should be set to 1 when call for new page is made (with the new video type)
- Below the pagination elements the number of results and the results shown should be indicated.
- Page should display appropriate page loading and error messages

Mockup

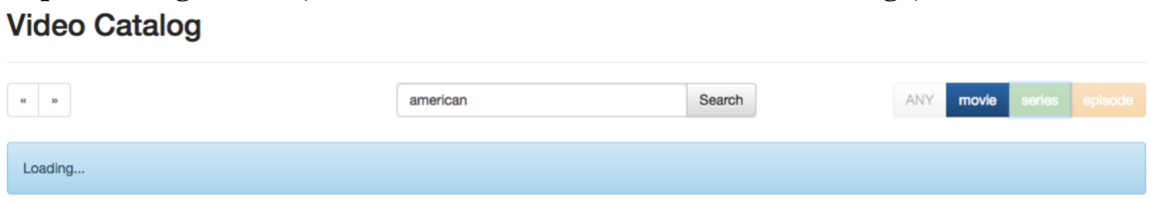
1. **Initial screen**
   Page 1 and search for American. No filtering.



Showing 1 - 10 of 3803

	Title	Year	Type
	American Beauty	1999	movie
	American History X	1998	movie
	American Psycho	2000	movie
	American Hustle	2013	movie
	American Sniper	2014	movie
	American Gangster	2007	movie
	American Pie	1999	movie

2. **Sample loading screen (error screen is similar but with error message)**



Extra credits:
Make sure every search-filter-pagination combo has a unique URL based on which page is displayed.

You can design the URL this way (only an example - you can design differently)
http://localhost:3000/?s=american&p=3&type=movie can display page 3 in movie matches for "american" keyword(s).

Additional Requirements
UI to be designed and developed by your team

You need to come up with your own design of components / pages to **support ALL of the following 3 features**. Design user-friendly interfaces. Take care of all different application states (loading, error, success, no data fallback message), responsive web design (phone, desktop), SEO, performance (appropriate rendering model), web a11y. Conduct Lighthouse audit on your app, and check for web vitals.

NOTE: When developing these pages if you feel the need for more pages (like login page, authorization page etc., please develop those as well).

1. **Detailed Movie Information Page**
    - **Description**: Implement a page that displays comprehensive details for a selected movie, series, or episode.
    - **API Endpoint**: GET /?i={imdbID}&apikey={API_KEY}
    - **Details to Display**:
        - Title, Year, Rated, Released
        - Runtime, Genre, Director, Writer, Actors
        - Plot, Language, Country, Awards
        - Poster, Ratings (including IMDb and Rotten Tomatoes)
        - Metascore, IMDb Rating, IMDb Votes
        - DVD release date, Box Office earnings, Production company, Website
    - **Purpose**: Provides users with in-depth information about a selected title, enhancing the application's utility.

2. **Filter by Year of Release**
    - **Description**: Allow users to filter search results based on the year of release.
    - **API Endpoint**: GET /?s={search_terms}&y={year}&apikey={API_KEY}
    - **Features**:
        - Input field or dropdown to select the desired year
        - Combine with existing type filters (movie, series, episode)
    - **Purpose**: Enables users to narrow down search results to specific timeframes, making searches more precise.

3. **Favorites Management**
    - **Description**: Implement functionality for users to add or remove titles from a list of favorites, stored locally in the browser.
    - **Features**:
        - "Add to Favorites" button on each title
        - "Favorites" page displaying all saved titles
        - Option to remove titles from the favorites list
    - **Purpose**: Allows users to curate a personalized list of preferred titles for easy access and future reference.