Project - Movie List App

You are tasked with building a movie information app that displays a list of movies from The Movie Database (TMDb) API. The app shows top movies for each year and users can filter by genre, the app also loads top movies from previous / next years as the user scrolls through the list.

Use this Figma prototype as a reference and feel free to design and develop based on your preferences.

Figma contains prototype designs for mobile, feel free to develop UI for desktop in a similar fashion.

Requirements

- Layout and UI
  - Create custom UI components for the app, using React, Jquery , Angular,vanilla JavaScript or any other javascript library or framework.
  - Display a list of movies sorted in descending order of popularity.
  - Show the movie title, image, genre, cast, director, and a short description related to the movie in each information card.
- Default page load state
  - Load a total of only 20 movies for each year.
  - By default, when a user lands on the page, display a list of movies of the year 2012
  - Implement smooth scrolling behavior to load more movies as the user scrolls in any direction i.e **load movies of previous year when user scrolls up and load movies of next year when user scrolls down until the current year. (You are free to use any library for it).**
  - As and when the user scrolls and movies are added to the list, make sure that this interaction is smooth and doesn't cause any jitters.
- API
  - Use the following URL to query a list of movie https://api.themoviedb.org/3/discover/movie?api_key=<APIKEY>&sort_by=popularity.desc&primary_release_year=<YEAR>&page=1&vote_count.gte=100 Create your account on TMDB and get the API KEY.
  - To fetch a movie of a specific year, use ***primary_release_year*** filter in the above query. Start from 2012, and fetch the previous or next year with respect to the direction in which the user scrolls the list.
  - ***Vote_count_gte = 100*** means fetch movies which have received at least 100 votes for popularity. The API returns the movie list sorted by popularity score. Feel free to change this value based on the movies you want to show. Please keep the minimum value at 100 to see relevant movies.
  - You can find the API documentation to change the properties accordingly as needed.

- Genre Filter
    - Provide a filter UI that allows users to filter movies by genre.
    - Fetch genres from_API and show genres as filters
    - When a user selects one or more genres, the list should only display movies of the selected genres. Please note that whenever a user selects a genre, a fresh list of movies should be fetched from the API for that particular genre.

- Code Quality
    - Write well-structured and maintainable code.
    - You can use any JS library/framework.
    - Avoid using any pre-built component libraries (e.g., Bootstrap, Tailwind, etc.), try using vanilla css for creating UI components.

Points to remember

- Feel free to assume UI / UX flows from the figma prototype, just make sure that the UI is usable.
- Provide the public GitHub repository link containing the source code of the project
- Include a README file explaining how to run the project and any additional details you'd like to provide.
- Provide a short list of requirements you have covered and not covered in the project README itself.
- Host your project on any platform, the live link should be working and present on README file.
- Feel free to reach out to your recruiter in case of any doubts.

Evaluation Criteria for Candidate: Movie Browser Application

- The list of genres should come from an API.
- Users should be able to select multiple genres.
- All genres should be included in the list.
- Users should be able to select and deselect genres by clicking on them.
- The list of movies should come from the API and be sorted by release year.
- When users select genres, only movies from those genres should be shown.
- Load movies from the previous year when users scroll up, and from the next year when they scroll down, until the current year. (You can use any library for this).
- Ensure the interface is mobile-friendly and visually appealing.
- The code should be well-organized into components, clean, and well-commented.
- Share the live site where the application can be viewed.
- The README file should be well-documented, explaining how to run the project locally and include the live site URL.
- Bonus points for incremental commits with clear comments for each feature.