Assignment - Vue 3 Movie Explorer

Overview

You are tasked with building a Movie Search and Favorites application using Vue 3 and the Composition API. This application will allow users to search for movies, view details, and manage a list of their favorite movies. The goal is to demonstrate your proficiency with Vue 3, API integration, component-based architecture, state management, and user interface development.

Time Limit

There is no set time limit. But you should try to spend around 2 – 3 hours on the project.

Assignment Objectives

- **API Integration:** Fetch data from a public movie database API.
- **State Management:** Manage application state effectively, including persisting data.
- **Vue 3 Proficiency:** Utilize the Composition API for state and logic management.
- **Component-Based Architecture:** Organize the application using reusable components.
- **User Interface Development:** Create a user-friendly and responsive UI.
- **Error Handling:** Gracefully handle errors and provide feedback to the user.

Technical Stack

- **Framework:** Vue 3 (Composition API).
- **State Management:** Local component state, and optionally, Pinia.
- **API:** Use the OMDb API (Open Movie Database).
- **API Key:** Obtain a free API key by registering at OMDb API Key.
- **Tooling:** Use Vite to scaffold your project.
- **Versioning:** Please commit your code regularly to a Git repository throughout your development process so we can review the progression of your work. This will help us understand your approach.
- **AI:** Please do not use AI tools when doing this coding task.

Features to Implement

Movie Search

- **Search Input:**
  Provide an input field where users can enter a movie title.
- **API Request:**
  Fetch a list of movies matching the search query using the OMDb API. Handle cases where no results are found.

Display Search Results

- **Movie List:**
  Display a list of movies returned from the search query.
- **Each movie item should show:**
  Movie Poster (use a placeholder if not available)
  Movie Title
  Release Year
- **Movie Details:**
  Allow users to click on a movie to view more details.
  Fetch detailed information for the selected movie.
- **Display additional details such as:**
  Genre, Plot Summary, Director, Actors, Ratings.

Favorites List

- **Add to Favorites:**
  Provide a way for users to add movies to their favorites list from the search results or movie details view.
- **Favorites Management:**
  Display the favorites list in a separate section or page.
  Allow users to remove movies from their favorites list.

User Interface

- **Navigation:**
  Implement basic navigation to switch between the search view and the favorites view.
- **Responsiveness:**
  Ensure the application is usable on both desktop and mobile devices.
- **Styling:**
  Basic styling is sufficient, but feel free to make the UI appealing.
  You may use CSS frameworks like Tailwind CSS or Bootstrap if desired.

Error Handling

- **User Feedback:**
  No search results are found.
  An error occurs during API requests.
  The user tries to add a movie to favorites that is already in the list.
- **Validation:**
  Validate user input to prevent empty searches.

Bonus items if time permits

- Implementing pagination.
- Adding a loading indicator during API requests.
- Using Pinia for state management.
- Using local storage to save the favorites so they persist across browser sessions.
- Implementing additional features like sorting or filtering.

Submission Guidelines

- Code Submission:
  Provide a link to a GitHub repository containing your project.
  Ensure that the repository is public or that access is granted.

- README File:
  Include instructions on how to install and run your application.
  Mention any assumptions or decisions you made.

- Running the Application:
  The application should run without errors.

Additional Notes

- Time Management:
  Focus on completing the core features first before attempting bonus features.

- Third-Party Libraries:
  You may use additional libraries to aid development (e.g., for HTTP requests or UI components), but ensure they are necessary and do not bloat the project

- Testing:
  While not required, adding unit tests (e.g., with Jest and Vue Test Utils) can showcase your testing skills.