Assignment: Build a Movie Search App using OMDb API and Redux

Objective:
Create a fully functional movie search app that allows users to search for movies by title, view movie details, and manage search results using Redux for state management.

Requirements:

1. **Frontend Framework:**
    - Use **React** for building the user interface.
2. **State Management:**
    - Use **Redux** for managing the application's state.
3. **API Integration:**
    - Use the **OMDb API** (Open Movie Database) to fetch movie data based on user input.
    - API Documentation: OMDb API
4. **Features to Implement:**
    - **Search Functionality**: Users should be able to input a movie title into a search bar, and the app will fetch results from the OMDb API.
    - **Movie List Display**: Display the results of the search in a list format, showing at least the movie's title, year, and poster.
    - **Movie Details View**: On selecting a movie from the list, display more detailed information about the movie (e.g., director, plot, rating, etc.).
    - **Error Handling**: Show an appropriate error message if no movie is found or if there's an issue with the API.
5. **State Management with Redux:**
    - **Search State**: Store the search query and the resulting list of movies in the Redux store.
    - **Loading State**: Implement a loading state that shows a spinner or message when data is being fetched from the API.
    - **Error State**: Store and display any error messages if the API request fails.

Project Setup:

1. **OMDb API Key:**

   - To use the OMDb API, you need an API key. You can obtain it by registering at OMDb API.

2. **Redux Setup:**

   - Configure Redux store with reducers and actions to handle:
     - Movie search queries.
     - Managing the list of fetched movies.
     - Handling loading and error states.

3. **React Components:**

   - **Search Bar Component**: A form to allow users to input the movie title.
   - **Movie List Component**: A component to display the list of movies returned from the search.
   - **Movie Details Component**: A separate view to display details of a selected movie.
   - **Loading and Error Components**: Optional components to show loading spinners and error messages.

Bonus (Optional Enhancements):

1. **Pagination**: Add pagination for search results if the list is too long.
2. **Favorites List**: Allow users to add movies to a favorites list and save this list in Redux.
3. **Styling**: Use a CSS framework (like Bootstrap or Material-UI) or custom CSS for better UI/UX.

Good luck!