CinePick – Movie Suggestion Browser

🎬Project Overview

CinePick is an interactive, data-driven movie browser that fetches movie details from an external API (like The Movie Database API). Users can explore trending movies, search for specific titles, and view detailed information such as release date, genre, and ratings. This project highlights React proficiency in API integration, routing, component architecture, and Material UI usage for a polished, professional interface.

🎴Core Features

1. **Movie Listing:** Display a grid of popular or trending movies.
2. **Search Functionality:** Search for movies by title.
3. **Movie Details Page:** Show detailed info when a movie card is clicked.
4. **Responsive Design:** Mobile-friendly grid layout.
5. **Favorites or Watchlist (Optional):** Add/remove movies from a saved list.
6. **Loading & Error States:** Indicate fetching and handle API failures gracefully.
7. **Animations:** Use Framer Motion for smooth page transitions.

User Flow

1. User lands on the home page with trending/popular movies.
2. User can search for a movie using the search bar.
3. When a movie is clicked, the app navigates to the details page.
4. User can view synopsis, release date, ratings, and cast.
5. Optionally, the user can favorite or bookmark a movie.

Folder Structure

```
src/
├── components/
│   ├── MovieCard.jsx
│   ├── SearchBar.jsx
│   └── Navbar.jsx
├── pages/
│   ├── Home.jsx
│   └── MovieDetails.jsx
├── services/
│   └── movieAPI.js
├── utils/
│   └── helpers.js
├── App.jsx
├── index.js
```

```
└── styles/
    └── main.css
```

Best Practices & Standards

- Use **functional components** and React Hooks.
- Maintain **clean separation** between UI and data logic.
- Centralize API calls in a `services` directory.
- Use **PropTypes** or TypeScript for component props.
- Handle all async actions with proper loading and error states.
- **Responsive-first design** with Material UI breakpoints.
- Follow accessibility guidelines for images and interactive elements.

Component Breakdown

1. `Navbar`

- Contains app title and search bar.
- Responsive and fixed at the top.

2. `SearchBar`

- Material UI `TextField` for user input.
- Calls TMDB API search endpoint on input.

3. `MovieCard`

- Displays movie poster, title, and rating.
- Click navigates to Movie Details page.

4. `Home` **Page**

- Displays a grid of popular movies.
- Uses `useEffect` to fetch movie data.

5. `MovieDetails` **Page**

- Displays detailed info: title, overview, genres, runtime, etc.
- Back button navigates to Home.

🎺React Concepts Applied

- **useState:** Manage search input, movie data, loading state.
- **useEffect:** Fetch data when the page loads or query changes.
- **React Router:** Route between Home and Details.
- **Optional useContext:** Manage global favorites or user preferences.

📷Data Management

- **API:** The Movie Database (TMDB) API - `https://api.themoviedb.org/3`
- Example endpoints:
  - `GET /movie/popular`
  - `GET /search/movie?query=...`
  - `GET /movie/{id}`
- Use **Axios** for clean API calls.
- Store the API key in an `.env` file for security.

🎨Styling & UI

- **Material UI Components:** AppBar, Grid, Card, CardMedia, Typography, Button.
- **Layout:** Grid system for movie cards, Flexbox for responsive behavior.
- **Framer Motion:** Animate card hover, route transitions, and modal popups.
- Use custom MUI theme for consistent branding.

🖌️Deployment

- **Platform:** Vercel or Netlify.
- **Steps:**
- Push to GitHub.
- Link repo to deployment platform.
- Add `REACT_APP_TMDB_API_KEY` as an environment variable.
- Deploy build folder (`npm run build`).

🎻Future Enhancements

- Add genre filters or categories.
- Implement infinite scroll or pagination.
- Add a watchlist using localStorage.
- Integrate user authentication (for saved lists).
- Support multiple languages via TMDB API localization.

CinePick combines clean architecture, visually engaging design, and practical API integration to create a sleek and modern movie browsing experience that adheres to global front-end standards.