Project Proposal: Movie Selector App

Tech Stack

- **Frontend (Web):**
  - **React.js:** For building an interactive and dynamic user interface.
  - **React Router:** For managing page navigation.
  - **Tailwind CSS:** For styling components quickly and responsively.
- **Backend:**
  - **Node.js with Express.js:** To build RESTful APIs, handle server-side logic, and manage API calls to third-party services.
- **Database:**
  - **MongoDB (with Mongoose):** To store user preferences, favorite movies, and generated recommendations (if a custom logic is used).
- **External API:**
  - **TMDB (The Movie Database API):** To fetch movie metadata such as genres, cast, ratings, and related recommendations.
  - Alternatively, a **custom recommendation engine** may be developed using curated data.
- **Hosting:**
  - **Vercel or Netlify:** For deploying the frontend.
  - **Render or Heroku:** For backend services and database deployment.

Focus of the Project

The application will be an **evenly focused full-stack project**, combining an engaging user interface with custom logic and API integration on the backend. The recommendation engine (either API-driven or custom-built) will be the centerpiece of the app.

Project Type

- **Web Application:** Responsive and optimized for both desktop and mobile browsers.

Project Goal

The goal of the project is to help users discover new movies based on their personal taste. By entering **three movies they enjoy**, the user receives **tailored movie recommendations**. This solves the common problem of decision fatigue when choosing what to watch.

User Demographic

The primary audience includes:

- Movie lovers aged **16–40** who enjoy finding new films based on past favorites.
- Users frustrated with endless scrolling on streaming platforms.
- People looking for a simple, quick recommendation tool that doesn't require creating an account or watching trailers first.

Data and API

- **Data:**
  The app will use movie data that includes:
  - Movie title
  - Genre
  - Director
  - Cast
  - Average rating
  - Poster image
  - Related or similar movies
- **Data Source:**
  - The **TMDB API** will likely serve as the main data source.
  - If limitations are encountered (e.g., rate limits or missing fields), a **custom dataset** may be created and stored in MongoDB.

Project Approach

1. Database Schema

```
javascript
CopyEdit
User {
  _id,
  username,
  favorites: [movieId],
  recommendations: [movieId],
  createdAt
}

Movie {
  _id,
  title,
  genre: [String],
  director,
  cast: [String],
  rating,
  description,
  posterUrl
}
```

2. Potential API Issues

- **Rate Limiting:** TMDB API has limits on free-tier usage.
- **Data Gaps:** Some lesser-known movies may not have complete metadata.
- **Recommendation Accuracy:** Ensuring relevance based on user input may require refining logic beyond genre or rating.

3. Sensitive Information

- No sensitive personal data will be collected.
- If user accounts are implemented, **email/passwords** will be hashed and secured using **JWT authentication**.

4. Functionality

- Movie search with autocomplete.
- Form to input 3 favorite movies.
- Backend generates and returns 1–5 recommendations.
- Display posters, descriptions, and optional trailer links.
- Optional: Save user history or favorite recommendations.

5. User Flow

- **Homepage:** Brief explanation + input form for 3 movies.
- **Results Page:** Displays recommended movies.
- **Optional Login:** To save recommendations for later.

6. Stretch Goals

- **User Accounts:** Allow users to save favorite picks and history.
- **Custom Recommendation Engine:** Analyze genre overlap, director similarity, actor recurrence, and user ratings.
- **AI Chatbot Assistant:** Conversational assistant to help users pick movies.
- **Trailer Integration:** Embed YouTube trailers for each movie.
- **Dark Mode:** Improve UX and accessibility.