

The background features a horizontal rainbow gradient transitioning from purple on the left to blue on the right. A white film strip border with rectangular sprocket holes runs along the top and bottom edges of the image.

Movie Explorer

Team 4:

Louis Chartier, Shao Hang Li, Trang Trieu

Table of Contents



Overview



Tools



API



What we learned



What to improve

Overview

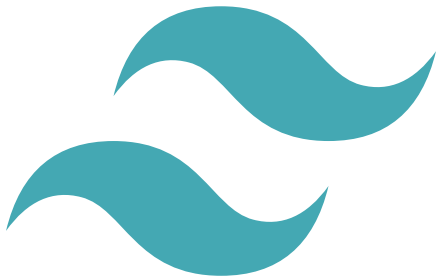
- Browse, search, and explore movies or TV shows using TMDB API
- Click on any movie or show to view details, trailer, cast, and similar recommendations
- Logged in users can manage personal favorites and watchlist
- Built with React, React Router, TailwindCSS + DaisyUI



Tools

Styling

Deploy



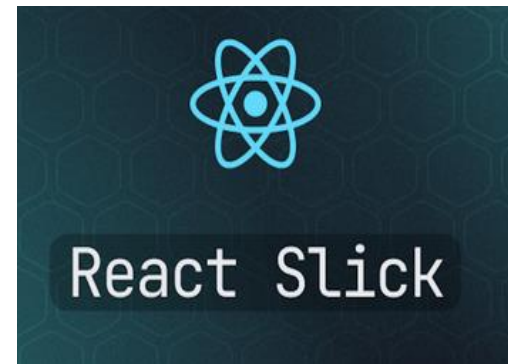
Tailwind CSS



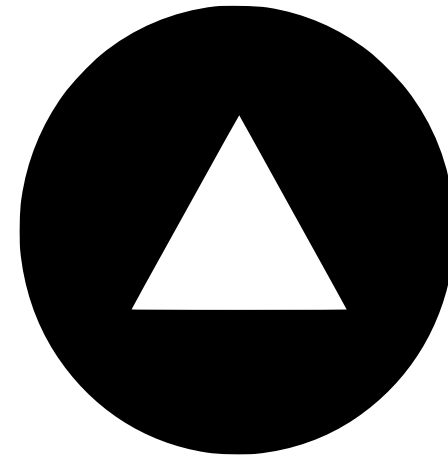
Lucide



DaisyUI



React Slick



Vercel

API

- TMDb API Documentation
- Fetch in api.js

ACCOUNT

Details	GET
Add Favorite	POST
Add To Watchlist	POST
Favorite Movies	GET
Favorite TV	GET
Lists	GET
Rated Movies	GET
Rated TV	GET
Rated TV Episodes	GET
Watchlist Movies	GET
Watchlist TV	GET

Details

GET https://api.themoviedb.org/3/account/{account_id}

Get the public details of an account on TMDb.

RECENT REQUESTS

TIME	STATUS	USER AGENT
Make a request to see history.		
0 Requests This Month		SEE ALL REQUESTS →

PATH PARAMS

account_id int32 required Defaults to 22072819

QUERY PARAMS

session_id string

RESPONSE

200
200

RESPONSE BODY

```
const API_KEY = " ";
const BASE_URL = "https://api.themoviedb.org/3";

// fetch api of popular movies
export const getPopularMovies = async () => {
  const response = await fetch(`${BASE_URL}/movie/popular?api_key=${API_KEY}`);
  const data = await response.json();
  return data.results;
};
```

What we learned (Shao Hang)

```
const AuthContext = createContext();
```

Windsurf: Refactor | Explain | Generate JSDoc | ✕

```
export const useAuth = () => {  
  const context = useContext(AuthContext);  
  if (!context) {  
    throw new Error("useAuth must be used within an AuthProvider");  
  }  
  
  return context;  
};
```

Windsurf: Refactor | Explain | Generate JSDoc | ✕

```
export const AuthProvider = ({ children }) => {  
  const [isAuthenticated, setIsAuthenticated] = useState(false);  
  const [user, setUser] = useState(null);  
  const [loading, setLoading] = useState(true);  
  
  useEffect(() => {  
    checkAuthStatus();  
  }, []);  
};
```

React Hook useEffect has a missing dependency: 'checkAuthStatus'

What we learned (Louis)

- Used useEffect + useParams to fetch Movie/TV data dynamically
- Implemented dynamic routing with React Router
- Integrated with TMDB API for movies and shows

useParams

```
import { useParams } from "react-router-dom";

function MovieDetails() {
  const { id } = useParams();
```

```
<Route path="/movie/:id" element={<MovieDetails />} />
<Route path="/tv/:id" element={<TVDetails />} />
```

useEffect

```
useEffect(() => {
  const fetchData = async () => {
    try {
      // fetch movie details
      const movieData = await getMovieDetails(id);
      setMovie(movieData);

      // fetch credits
      const creditsData = await getMovieCredits(id);
      setCredits(creditsData);

      // fetch trailer
      const videosData = await getMovieVideos(id);
      const trailer = videosData.results.find(
        (vid) => vid.type === "Trailer" && vid.site === "YouTube"
      );
      setTrailerKey(trailer?.key);

      // fetch similar movies
      const similarData = await getSimilarMovies(id);
      setSimilarMovies(similarData);
    } catch (error) {
      console.error("Error fetching movie data:", error);
    }
  };
});
```

What we learned (Trang)

useNavigate:

provides programmatic navigation capabilities

In SearchBar.jsx

```
const navigate = useNavigate(); // navigate search query

const handleSearch = async (e) => {
  e.preventDefault();
  if (!searchQuery.trim()) return; // prevent empty searches

  setIsLoading(true);
  setError(null); // clear any previous errors

  // navigate to search results page with url encoded query
  navigate(`/search?query=${encodeURIComponent(searchQuery)}`);
  setIsLoading(false);
};
```

In App.js

```
<Route path="/search" element={<SearchResult />} />
```

useLocation:

provides access to current URL location

```
// custom hook to extract and parse URL query parameters
function useQuery() {
  return new URLSearchParams(useLocation().search);
}
```

```
// xtract search query from url parameters
const query = useQuery().get("query") || "";
```




What to improve

- Add Advanced Search for:
 - Movies
 - TV Shows
- Add Celebrity List
- Add BE and DB



Thank you

