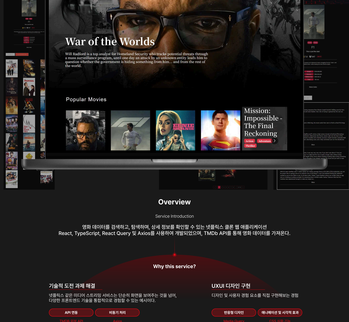
NETFLIX



Overview

Service Introduction

영화 데이터를 검색하고, 탐색하며, 상세 정보를 확인할 수 있는 넷플릭스 클론 웹 애플리케이션

React, TypeScript, React Query 및 Axios를 사용하여 개발되었으며, TMDb API를 통해 영화 데이터를 가져옵니다.

Why this service?

기술적 도전 과제 해결

넷플릭스 같은 데이터 스트리밍 서비스는 다음과 같은 복잡한 요구사항을 가지고 있습니다. 이러한 요구사항들을 어떻게 해결하였는지 간략한 내용을 아래에 정리하였습니다.

- API 연동 — TMDb 10K API
- 비동기 처리 — Axios
- 상태 관리 — Query Client
- 라우팅 — React Router

UI/UX 디자인 구현

디자인 및 사용자 경험 요소를 직접 구현하였는 경험

- 반응형 디자인 — Media Query
- 다크 모드 지원 — CSS 변수 기능
- 로딩 상태 표시 — Skeleton UI / 로딩 스피너
- 에러 처리 — 에러 바운더리

Technology Stack

Front Main

- React — 컴포넌트 기반 아키텍처, 가상 DOM 최적화
- TypeScript — 정적 타입 시스템 도입, 컴파일 단계에서 오류 방지, 코드 자동 완성 및 리팩토링 용이성 제공, 대규모 프로젝트 유지 보수에 유리함
- React Router — SPA(Single Page Application) 라우팅 구현, 브라우저 히스토리 관리, 동적 URL 파라미터 및 쿼리 스트링 처리, 중첩 라우팅 및 레이아웃 구성, 404 페이지 등 예외 처리

Data Management&Communications

- React Query — 서버 상태 관리 라이브러리, 캐싱 및 백그라운드 업데이트 자동 처리, 로딩 및 에러 상태 관리 용이, 무한 스크롤 구현 지원, 데이터 동기화 및 캐시 무효화 관리
- Axios — HTTP 요청을 위한 라이브러리, 인터셉터 기능을 통한 요청 및 응답 공통 처리, 타임아웃 및 에러 처리 기능, 요청 및 응답 변환 기능
- TMDb API — 영화, 배우, 장르 등 방대한 영화 데이터 제공, 검색, 인기 영화, 장르별 영화 목록 등 다양한 API 엔드포인트 제공, 개발자 친화적인 문서 및 예제 제공

UI/UX

- Bootstrap — 반응형 그리드 시스템 제공, 미리 스타일 구성된 컴포넌트 제공, 빠른 프로토타입 제작 가능, 커스텀 테마 적용 가능
- Custom CSS — 부트스트랩으로 해결할 수 없는 디자인 요구사항 충족, 브랜드 고유의 시각적 개성 구현, 다크 모드 및 반응형 디자인 구현, 프로젝트 고유의 디자인 시스템 구축

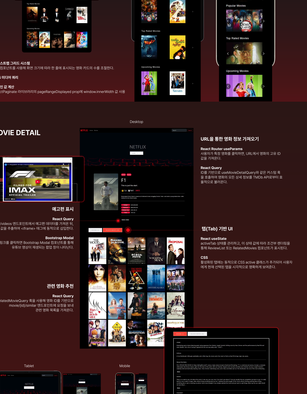
Service Development Process

데이그래이션 과정

프로젝트 초기	프론트 단 개선	인증 및 개선
React (HTML, CSS, JavaScript) 기반 구현	React Query 도입	JavaScript → TypeScript 변환
컴포넌트 중심의 구조화	서버 데이터 관리 최적화	타입 안정성 확보

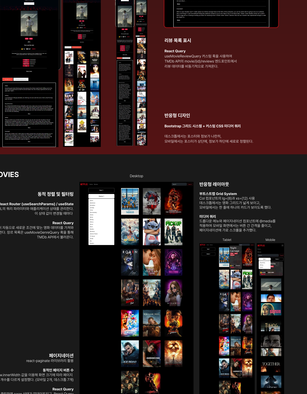
초기 과정 블로그 https://blog.naver.com/higkmin/223974244442

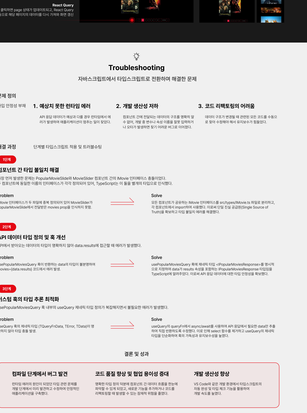
Main Features



반응형 레이아웃

모바일(768px 이하), 태블릿(769px~991px), 데스크톱(992px 이상) 기준 미디어 쿼리를 활용하여 모든 상태 컴포넌트 (로딩, 에러, 빈 목록) 대응하도록 구현함





Troubleshooting

지속적인 개선을 위한 문제 해결 사례

문제 정의	1. 해상도 정한 한타임 해결	2. 개발 생산성 저하	3. 코드 리팩토링의 어려움
타입 안정성 부재	API 응답 데이터가 예상치 않게 변경되거나 데이터 일관성에 예측하기 어려운 일이 있었음.	컴포넌트 간에 연결되는 데이터의 구조를 명확히 정의하지 않아, 개발 중 런타임 오류가 자주 발생하여 디버깅에 많은 시간이 소요되었음.	데이터 구조가 명확하지 않아 코드의 가독성이 떨어지고 유지보수가 어려운 상황이었음.
해결 과정	단계별 타입스크립트 적용 및 최적화		
단계	컴포넌트 간 타입 불일치 해결	개발 생산성 저하	코드 리팩토링
Problem	Problem	Solve	Solve
Problem	Problem	Solve	Solve
단계	API 데이터 타입 정의 및 개선		
Problem	Solve		
단계	커스텀 훅의 타입 추론 최적화		
Problem	Solve		

결론 및 성과

컴파일 단계에서 버그 발견	코드 품질 향상 및 협업 용이성 증대	개발 생산성 향상
정의된 데이터 모델과 다른 데이터가 전달되는 경우 컴파일 단계에서 바로 발견하고 수정하여 런타임 에러 발생 가능성을 크게 줄였음.	명확한 타입 정의를 통해 코드 자체가 훌륭한 주석 역할을 하게 되었고, 새로운 기능을 추가하거나 코드를 리팩토링할 때 발생할 수 있는 잠재적 위험을 줄였음.	VS Code와 같은 개발 환경에서 타입스크립트 자동 완성 및 제안 기능을 활용하여 개발 속도를 높이고, 코드 오류를 줄이는 데 기여하였음.
Round Query	Round Query	

After Plans

서버의 정확한 기능 추가	인증 및 보안 기능 고도화
데이터베이스(MongoDB)	다크 모드
회원가입, 로그인, 찜, 시청 목록 등 데이터 영구 저장을 위해 MongoDB를 데이터베이스로 채택하고, 관련 API 엔드포인트를 개발하였음.	보다 쉬운 접근성 개발 및 사용자 경험 향상을 위해 다크 모드 기능 구현

https://github.com/hamine021/Project_NetflixClone.git

https://project-netf-clone.netlify.app/

Contact

dkmin22@gmail.com