Netflix 서비스 클론 코딩 기획 및 개발 문서

1. 프로젝트 개요

• 프로젝트명: Netflix Clone

• 목적: 강의에서 학습한 React, HTML, CSS, TypeScript 등을 활용하여 Netflix 의 메인 페이지를 클론 개발함으로써 웹 프론트엔드 개발 전반에 대한 실전 감각을 키우고, 라우팅, 배포, 반응형 UI/UX 설계 등 핵심 역량을 향상시키는 것을 목표로 함.

• 개발 범위: 프론트엔드 (React + TypeScript + Vite), 배포 환경 구성 (Docker, AWS)

• 사용 도구: VSCode, GitHub, Figma, Docker, AWS (EC2, S3 등), Vite

• 개인 프로젝트: 유현진

• 구현 일정: 2025.07.15 ~ 2025.07.21

2. 기획 배경 및 벤치마킹

• 문제 인식:

    • React, DOM, TypeScript 기초 강의 학습 후 이를 바탕으로 실제 Netflix UI 구현

    • AWS 환경 이해하여 Docker 를 활용한 배포

• 벤치마킹 서비스: Netflix 공식 웹사이트의 홈 화면 및 콘텐츠 카드 UI

• 구현 목표:

    • 넷플릭스 메인 페이지의 상단 배너, 영화 섹션, 반응형 레이아웃, 모달 창 등 구성

    • React + Vite 기반 컴포넌트화

    • 타입스크립트 적용

    • Docker 를 이용한 AWS 배포

3. 서비스 플로우

페이지	구성 요소	기능
MainPage	배너, 영화 섹션	Swiper 를 이용한 영화 섹션 이동
SearchPage	검색 창, 검색 오류	검색 창, 검색 내용과 관련된 포스터 출력
DetailPage	검색한 포스터	검색한 포스터 출력
MovieModal	영화 정보 모달 창	모달 창이 아닌 부분을 클릭 시 모달 창 닫기

4. 기능 명세서

기능명	설명	적용 기술	구현 여부
Nav UI 구성	로고, 검색 창, 사용자 프로필 등 상단 네비게이션 구성	React, CSS, TypeScript	✅
배너 섹션	배너 표시, 배경 이미지와 함께 설명 및 버튼 표시	React, CSS, Axios, TypeScript	✅
콘텐츠 카드 행(Row)	영화/시리즈 목록을 장르별로 슬라이드 형태로 구성, Swiper 이동	React, Axios, TMDB API, CSS, Swiper, TypeScript	✅
모달(MovieModal)	카드 클릭 시 상세 정보 팝업(모달) 띄움, 모달 이외 클릭 시 창 닫음	React, useOnClickOutside Hook	✅
검색 기능	키워드 입력 시 TMDB API 를 통해 콘텐츠 실시간 검색	Axios, useDebounce Hook, TypeScript	✅
검색 페이지 (SearchPage)	검색 결과를 별도 페이지에서 카드 목록 형태로 표시	React Router, CSS, TypeScript	✅
상세 페이지 (DetailPage)	콘텐츠 클릭 시 영화 포스터가 표시	React Router, Axios, TypeScript	✅

반응형 레이아웃	화면 크기에 따라 콘텐츠 카드, 배너, 네비게이션 구조가 유동적으로 변경	CSS Media Query	✅
커스텀 훅 적용	useDebounce, useOnClickOutside 등 재사용 가능한 로직 구현	TypeScript, Custom Hook	✅
TypeScript 타입 적용	영화 데이터, API 응답 객체, 컴포넌트 Props 등 타입을 명시적으로 선언	TypeScript	✅
Axios API 모듈화	TMDB API 요청을 axios.ts, request.ts 로 구조화	Axios, TypeScript	✅
Vite 개발 환경	Vite 기반의 빠른 개발 서버 및 HMR 적용	Vite	✅
Docker 기반 배포	Dockerfile 로 앱 이미지 빌드 후 컨테이너화하여 EC2 에 배포	Docker, AWS EC2	✅
.env 환경 변수 설정	TMDB API Key 등 민감한 정보를 환경 변수로 관리	VITE_ .env 설정	✅
페이지 라우팅	메인, 검색, 상세 페이지로 라우팅 구성	React Router	✅

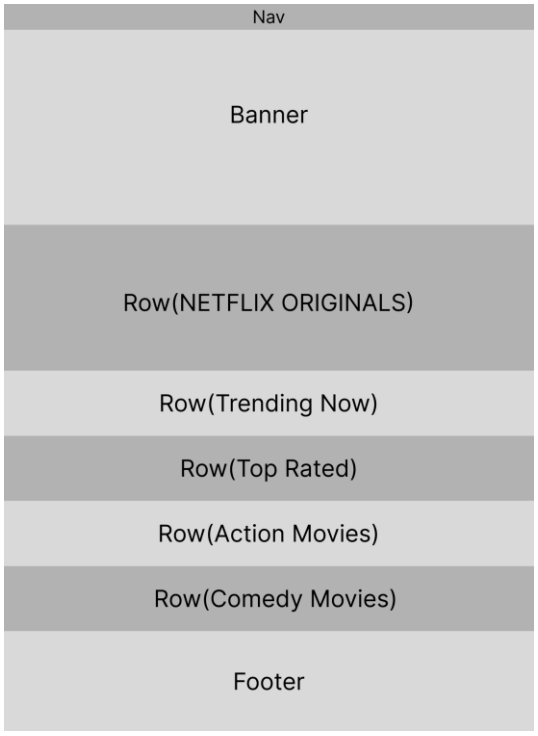
5. 페이지 구성 및 계층 구조

```
/ (root)
├── .env                # 환경변수 (API_KEY 등)
├── .gitignore
├── Dockerfile          # Docker 배포 설정
├── index.html          # Vite 템플릿 HTML
├── package.json
├── package-lock.json
├── README.md
├── tsconfig.json       # TypeScript 설정
└── vite.config.js      # Vite 설정
```

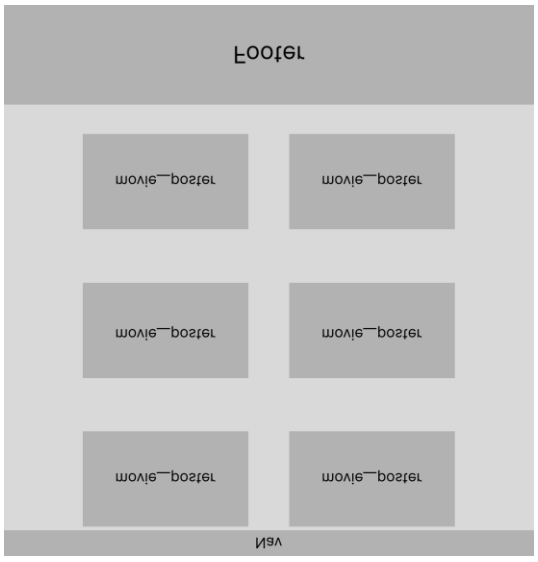
```
/src
├──── api                # API 요청 관련 모듈
│   ├──── axios.ts         # Axios 기본 인스턴스 설정
│   └──── request.ts       # TMDB 요청 URL 모음
│
├──── components          # 재사용 가능한 UI 컴포넌트
│   ├──── Banner.tsx       # 메인 상단 배너
│   ├──── Banner.css
│   ├──── Footer.tsx       # 페이지 하단
│   ├──── Nav.tsx          # 상단 네비게이션 바
│   ├──── Nav.css
│   ├──── Row.tsx          # 콘텐츠 행 (섹션별 카드)
│   ├──── Row.css
│   └──── MovieModal       # 영화 상세 모달 컴포넌트
│       ├──── index.tsx
│       └──── MovieModal.css
│
├──── hooks               # 커스텀 훅 모음
│   ├──── useDebounce.ts     # 검색 디바운스 처리
│   └──── useOnClickOutside.ts # 모달 외부 클릭 감지 훅
│
├──── pages               # 라우팅 기반 페이지 구성
│   ├──── MainPage
│   │   └──── index.tsx      # Netflix 홈 화면
│   ├──── DetailPage
│   │   └──── index.tsx      # 상세 페이지
│   └──── SearchPage
│       ├──── index.tsx      # 검색 결과 페이지
│       └──── SearchPage.css
│
├──── App.tsx             # 전체 앱 라우팅 구성
├──── App.test.js
├──── index.css           # 전역 스타일
├──── main.tsx            # 엔트리 포인트
├──── movie.ts            # 영화 타입 정의
```

6. 와이어프레임

- 도구: Figma



MainPage 구성



SearchPage 구성

7. 기술 구현 설명

항목	설명
React	컴포넌트 단위 설계 및 상태 관리, props 전달
Vite	초고속 빌드 및 개발 서버, 코드 분할 및 최적화
TypeScript	정적 타입 검사로 컴포넌트 안정성 확보
Docker	Dockerfile 작성 → 빌드 → 이미지 실행, AWS EC2 배포 연동
AWS	EC2 인스턴스 배포, 퍼블릭 IP 연결

8. 개발 일정 및 기록

날짜	작업 내용	상태
6/15	프로젝트 기획 문서 작성	완료
6/15~17	React 구조 구성 및 구현	완료
6/18	TypeScript 전환	완료
6/19	Vite 전환	완료
6/20~21	Dockerfile 작성 및 EC2 배포, AWS 연결	완료
6/21	발표 자료 정리	완료

9. 결과물 및 링크

• 기획 문서: 본 문서 참조

• Github 레포지토리: https://github.com/Hyunjin21/react-netflix-exp.git

• 배포 주소(AWS): http://15.165.160.119/

• 발표 자료: pdf 파일

10. 결론 및 회고

• React/TypeScript/Vite 를 통해 Netflix 웹사이트 구조를 분석하고 재현하면서 UI/UX 설계 감각 및 컴포넌트 단위 개발 역량을 실전 경험으로 쌓을 수 있었음.

• AWS 및 Docker 기반 배포를 진행하며 프론트엔드 프로젝트의 실제 서비스 운영 흐름에 대한 이해도를 높였음.