과제 2) GPT와 함께 Front-End Demo Site 개발 (Netflix...?)

이번 과제에서는 GPT를 최대한 활용하여, Netflix와 비슷한 Front-End Demo site를 만들고자 합니다. **React.js** 또는 **Vue.js**를 활용해 Single Page Application (SPA)를 개발하고, 이를 Github pages를 활용하여 정적 웹사이트 배포까지 실습하는 것을 목표로 합니다.

Demo Page: http://clinic.jbnu.ac.kr:3000/24-02-WSD-Assignment-02-Demo/#/

A. 과제 목적

- **최신 프론트엔드 기술 습득**: React.js 또는 Vue.js를 활용하여 SPA를 개발함으로써 최신 프론트엔드 프레임워크에 대한 이해와 실무 능력을 향상.
- **GPT 활용 능력 강화**: GPT를 최대한 활용하여 개발 과정에서 AI도구를 적용하고, 효율적인 코딩 및 문제 해결 능력을 기름.
- **정적 웹사이트 배포 경험**: Github Pages를 이용하여 정적 웹사이트를 배포하는 과정을 실습하여, 배포 프로세스와 호스팅에 대한 이해를 높임.
- **UI/UX 디자인 이해**: Netflix와 유사한 프론트엔드 데모 사이트를 제작하면서 사용자 경험 중심의 디자인 원칙을 학습하고 적용함 (+css transition 등).
- **오픈소스 및 라이브러리 활용 능력 향상**: React.js 또는 Vue.js와 같은 오픈소스 프레임워크와 라이브러리를 활용하여 개발 역량을 강화함.
- **자기 주도적 학습 역량 배양**: GPT와 오픈소스 도구를 활용하여 독립적으로 학습하고 문제를 해결하는 능력을 향상시킴.
- **웹 개발 및 배포 흐름 ~~이해~~**: SPA 개발부터 정적 웹사이트 배포까지의 전체 과정을 경험하여 웹 개발 프로세스 전반에 대한 이해를 높임.

B. 제출 방법

- **과제 마감일: 2025년 12월 16일 23:59 (KST)**
- 과제 제출 방법: **Google Classroom**을 통해 제출
- **Google Classroom**에 **제출**하기 버튼을 눌러 제출하면 됩니다.
- 아래 파일들을 **하나의 zip파일**로 묶어 다음과 같은 포맷으로 **WSD-분반-학번-이름-2차과제.zip** 파일 만들어 제출하면 됩니다. 파일 이름 및 포맷 꼭 지켜주세요!!!
  - **WSD-1분반-20240000-이경수-2차과제.zip**
    - WSD-Assignment-02
      - Vue.js 또는 React.js 프로젝트 폴더
      - **node_modules 폴더는 반드시 포함하지 말 것 (용량 문제로, 포함하여 제출 시 삭제 예정)**
      - 단, package.json 파일은 반드시 포함할 것 (npm install 안되면 0점 처리)
      - **README.md 파일을 반드시 포함해야하며, npm 명령어를 반드시 명시할 것**
    - link.pdf
      - Github repository 주소: https://github.com/~~~
      - Github pages 주소: https://~~~/~~~github.io
    - mobile.mp4
      - 위 Github pages에 접속하여 모바일 화면으로 반응형 웹이 잘 동작하는 것을 스크린 캡쳐 또는 동영상 녹화하여 제출하면 됨. (본인의 핸드폰 사용하여 웹 페이지에 접속할 것!)
    - AI.ppt 또는 AI.pdf
      - ChatGPT/Claude와의 주요 대화 내용 제출
      - Content 및 제출 형식:
        - 질문-답변 쌍을 페이지당 1세트씩 구성
        - (본인이 생각하기에 중요하다고 생각하는) **핵심적인 대화 내용 최대 20세트 선별**
      - 제출 파일 형식: PPT 또는 PDF

C. 유의 사항

- **표절 금지**:
  - 모든 작업은 본인의 노력으로 이루어져야 합니다
  - 타인의 코드를 무단으로 복사하거나 표절할 경우 0점 처리됩니다
  - 생성형 AI의 답변을 그대로 복사하여 제출하는 것도 표절로 간주되지는 않으니 괜찮습니다 😝

- **제출 형식**:
  - 모든 제출물은 지정된 형식을 준수해야 합니다
  - 압축 파일명, 폴더 구조를 반드시 준수해 주세요
  - 미준수 시 채점 상 불이익이 있을 수 있습니다

- **개발 환경**:
  - 꼭! node_modules 폴더는 제외하고 제출합니다
  - package.json 파일은 반드시 포함되어야 합니다
  - 꼭! 프로젝트가 정상적으로 실행/빌드되는지 확인 후 제출해 주세요

- **문의 사항**:
  - 과제 수행 중 질문이 있으면 다음 채널을 통해 문의해 주세요:
    - 이메일
    - Discord
    - Google Classroom
  - 문의 시 본인의 학번, 이름, 분반을 함께 기재해 주세요

D. 과제 개요

본 과제는 Vue.js 또는 React.js를 활용하여, 영화 poster를 보여주는 데모 사이트를 만드는 것을 목표로 합니다. 또한, 최신 Front-End 개발 트렌드는 AI를 활용한 작업 단순화 및 작업 속도 개선에 초점이 맞춰져있어, 이를 적극 활용하여 개발하는 것을 목표로 합니다. Vue.js 또는 React.js 프레임워크를 통해, Node.js 개발 환경에서, Single Page Application (SPA)를 개발하고, 정적 웹사이트를 Github Pages 또는 Netlify에 배포하도록 합니다. Front-End framework룰 통해 개발한 어플리케이션을 Workflow 자동화를 통해, Github Pages 또는 Netlify에 정적 웹사이트를 배포하도록 합니다. 프론트엔드 개발에 익숙하지 않으신 분들을 위해 Sample Code (Angular.js)를 제공해드리니, 참고하셔서 개발하시면 좋을 것 같습니다! 추가로, GPT/ Claude/ Perplexity/ Cursor 등 다양한 AI를 활용해 개발해도 괜찮으니, 적극적으로 사용하시면 될 것 같습니다! 😄

1. 데모 페이지

여러분들께서 **노가다를 줄이고** 손 쉽게 과제를 해결하기 위해서 Demo page를 제작하였습니다.

또한, 여러분들께서 참고하시기 쉽게 Angular.js로 된 코드를 함께 드리니, 적극 활용하시길 바랍니다!

(위 두 개 때문에 오래걸린거니 양해해주세요... 🥲😂😫)

아래 소스코드를 참고하시되, 다음과 같은 항목에 따라 점수가 달라지니 과제 하실 때 참고해주세요! (아래는 예시입니다)

- Angular 및 Demo Page와 동일한 Style 및 Animation 구현: 1점
- Angular 및 Demo Page와 비슷한 Style 및 Animation 구현: 2점
- Angular 및 Demo Page와 확연하게 다르며, fancy한 Style 및 Animation 구현: 3점

다만, 몇 가지 기능들은 일부러 구현 안한 것도 있으니, **무조건적으로 demo page를 보면서 개발하는 것이 아니라, 아래 instruction을 꼭 참고하시길 바랍니다!**

아래 링크를 참고해주세요:

- **Demo Page:** http://clinic.~~jbnu~~.ac.kr:3000/24-02-WSD-Assignment-02-Demo/#/
- **Angular.js Code:** https://github.com/JBNU-Teaching/24-02-WAS-assignment-02-angular

2. 주요 평가 내용

본 과제에서 주요하게 평가할 주요 기능은 아래와 같습니다. 다만, (Optional) **이라고 적혀 있는 것은 평가 대상은 아니지만, 나중을 위해 한 번 공부해보시는 것을 권장드립니다.**

a. Front-End Framework를 통한 SPA 개발

- Vue.js 또는 React.js를 활용하여 프론트엔드 개발 수행
- 동적 바인딩을 통한 Top-Down 방식의 데이터 전달
- Bottom-Up 방식의 자식 컴포넌트에서 부모 컴포넌트로의 데이터/이벤트 전달
- Iterative Rendering 및 Conditional Rendering을 활용한 동적 컴포넌트 구성
- Javascript 또는 Typescript의 함수/클래스를 활용
- Reference를 활용한 변수 선언 및 DOM Component 접근
- View/ Component에 따른 폴더 구조 정형화
- Router를 활용한 Single Page Application 라우팅 처리
- Custom Hook/Composition API를 통한 로직 재사용성 향상
- HTTP 클라이언트(Axios 등)를 활용한 REST API 연동
- 상태 관리 라이브러리(Vuex/Redux)를 통한 전역 상태 관리 구현 (가산점)

b. 배포 자동화 및 정적 웹사이트 호스팅

- **Github Pages** 또는 **Netlify** 중 하나를 선택하여 배포하시면 됩니다.
- GitHub Pages를 통한 정적 웹사이트 호스팅
  - GitHub Repository의 Settings를 통한 Pages 설정
  - main 브랜치 또는 gh-pages 브랜치를 통한 배포
  - GitHub Actions workflow를 통한 자동 배포 구성
  - 버전 관리

- Netlify를 활용한 정적 웹사이트 호스팅
  - Git Repository와 Netlify 연동
  - Continuous Deployment를 통한 자동 빌드 및 배포
  - Branch/Deploy Preview 기능을 통한 배포 전 미리보기
  - 환경 변수 관리 및 배포 환경별 설정
  - 배포 롤백 및 버전 관리

- 배포 자동화 프로세스 (Optional)
  - Git branch 전략에 따른 배포 환경 분리
  - Pull Request 기반 배포 프로세스 구현
  - 자동화된 테스트 및 빌드 검증
  - 성능 및 웹 표준 검사 자동화
  - 배포 알림 시스템 구축 (Slack, Email 등)

- 배포 자동화가 어려우신 분들은 npm run build 명령어를 통해 정적 파일을 생성 후, 해당 파일만을 올려 정적 호스팅을 해도 되지만, 높은 점수를 얻지 못할 수 도 있습니다.

c. Github 및 버전 관리

- 이번 과제에서는 주기적인 Commit 개수를 세지는 않지만, 총 갯수는 관리할 예정입니다! (Commit/ PR 포함)
- 이번 과제에서는 Gitflow 전략을 도입하여 프로젝트를 구성하시길 바랍니다! **(적용 안될 시 20% 감점)**
- **세부적인 내용은 다음과 같습니다:**
  - 효과적인 README.md 파일 작성
    - 프로젝트 기본 정보
    - 기술 스택 명시
  - 효율적인 커밋 관리 (Optional)
    - 명확한 커밋 메시지 컨벤션 준수
    - 논리적 단위로 커밋 분할

    - 설치 및 실행 가이드
    - 프로젝트 (폴더) 구조 설명
    - 개발 가이드 (Optional)
      - 코딩 컨벤션
      - Git 커밋 메시지 규칙
      - 브랜치 전략 설명
      - PR 템플릿 안내
      - 이슈 등록 방법
    - 추가 문서 링크 (Optional)
      - API 문서
      - 사용자 가이드
      - 개발자 문서
      - 변경 이력
      - 관련 프로젝트 링크
  - Git Flow 브랜치 전략 적용
    - main: 제품 출시 브랜치
    - develop: 개발 브랜치
    - feature: 기능 개발 브랜치
    - release: 출시 준비 브랜치 (Optional)
    - hotfix: 긴급 수정 브랜치 (Optional)
    - 의미 있는 태깅 전략 수립
  - Pull Request 기반 workflow (Optional)
    - PR 템플릿 작성 및 활용
    - 코드 리뷰 프로세스 정립
    - 리뷰어 지정 및 승인 절차 수립
    - 자동화된 테스트 결과 확인
    - 충돌 해결 및 머지 전략 수립
  - Github 프로젝트 관리 (Optional)
    - 이슈 트래킹 및 마일스톤 관리
    - 프로젝트 칸반 보드 활용
    - Wiki를 통한 문서화
    - GitHub Actions를 통한 CI/CD 구축
    - Repository 보안 설정 관리
  - 브랜치 보호 규칙 설정 (Optional)
    - 직접적인 main 브랜치 푸시 제한
    - PR 승인 필수 정책 적용
    - 빌드 및 테스트 통과 강제
    - 서명된 커밋 요구사항 설정
    - 브랜치 최신화 요구사항 설정

d. The Movie Database (TMDB) 활용

- The Movie DataBase (TMDB): https://www.themoviedb.org
- TMDB의 API를 활용하여 Front-End에서 API를 호출하고, API Key를 다루는 방법 및 비동기 데이터를 처리하는 방법을 학습
  - **API Key 발급 및 활용 방법:**
    - API 발급: https://ji-gwang.tistory.com/54
    - API 발급 및 테스트: https://hanmari-code.tistory.com/146
  - **함수 호출 참고:** URL.ts 파일
- **세부적인 내용은 다음과 같습니다:**
  - TMDB API 인증 및 설정
    - API 키 발급 및 환경변수 설정
    - 인증된 요청 헤더 구성
    - 에러 핸들링 로직 구현
  - 영화 데이터 요청 및 처리
    - 인기 영화 목록 조회
    - 영화 상세 정보 조회
    - 영화 검색 기능 구현
    - 장르별 영화 필터링
  - 이미지 리소스 관리 (Optional)
    - 포스터 및 배경 이미지 URL 구성
    - 이미지 사이즈 최적화
    - 이미지 로딩 상태 처리
  - 데이터 캐싱 및 최적화 (Optional)
    - API 응답 데이터 캐싱
    - 요청 횟수 최적화
    - 데이터 업데이트 주기 관리

e. Local Storage 활용

- Local Storage를 활용하여 사용자 데이터를 저장하고, 브라우저에 저장된 데이터를 활용하여 사용자 경험을 향상시키는 방법을 학습하고자 합니다.
- Local Storage에는 최소 3개 이상의 Key-Value 쌍이 저장되어야 하며, **+@ 개수에 따라 차등 점수를 부여**할 예정입니다.
- **세부적인 내용은 다음과 같습니다:**
  - 사용자 선호 데이터 관리
    - 최근 검색어 기록 저장
  - 데이터 구조화 및 관리 (Optional)
    - JSON 형식 데이터 변환

        - 즐겨찾기한 영화 목록 관리
        - 사용자 설정 (테마, 언어 등) 유지 (Optional)
        - 시청 기록 저장 및 관리 (Optional)
    - API 데이터 캐싱 전략 (Optional)
        - 자주 사용되는 영화 정보 캐싱
        - 장르 목록 로컬 저장
        - 이미지 URL 정보 임시 저장
        - 캐시 유효기간 설정 및 관리
    - 성능 최적화 (Optional)
        - API 호출 횟수 감소
        - 데이터 로딩 시간 단축
        - 오프라인 데이터 접근 지원
        - 스토리지 용량 관리
    - 스토리지 키 네이밍 규칙
    - 데이터 버전 관리
    - 스토리지 정리 로직 구현
    - 에러 핸들링 (Optional)
        - 스토리지 용량 초과 처리
        - 손상된 데이터 복구
        - 브라우저 호환성 체크
        - 폴백(Fallback) 메커니즘 구현
    - 보안 고려사항 (Optional)
        - 민감한 정보 제외
        - 데이터 유효성 검증
        - XSS 공격 방지
        - 데이터 암호화 처리

- **Local Storage를 통해서는 다음을 저장하고 관리해야 합니다:**
    - 회원가입 후 사용자의 아이디 & 비밀번호
    - 로그인 여부 (keep login)
    - 사용자의 선호 영화
    - 등등

f. CSS Transition 및 Animation 활용

- CSS를 활용하여 다양한 Transition 및 Animation 효과를 구현하고, 사용자 경험을 향상시키는 방법을 학습하고자 합니다.
- **세부적인 내용은 다음과 같습니다:**
    - 기본 트랜지션 효과
        - 영화 카드 호버 효과
        - 메뉴 아이템 상호작용
        - 버튼 상태 변화
        - 로그인-회원가입 페이지 전환 효과
    - 성능 최적화 (Optional)
        - transform 속성 활용
        - will-change 적용
        - 하드웨어 가속 활용
        - 프레임 드롭 방지
    - 애니메이션 제어
        - JavaScript 연동
        - 애니메이션 일시정지/재생 (Optional)
        - 동적 속성 변경
        - 이벤트 핸들링
    - 사용자 경험 개선 (Optional)
        - 적절한 타이밍 설정
        - 부드러운 이징 함수 적용
        - 중첩 애니메이션 조정
        - prefers-reduced-motion 지원
- **로그인-회원가입 페이지 전환 효과는 필수적으로 구현해야하며, 아래 예제를 참고하시길 바랍니다.**
    - Example 1: 예시 1 (Input 애니메이션)
    - Example 2: 예시 2 (반응형)
    - Example 3: 예시 3 (간단한 transition)
    - Example 4: 예시 4 (복잡한 transition)
    - Example 5: 예시 5 (복잡한 animation)
    - 등등

g. GPT/Claude 등 AI 서비스 활용

- AI 서비스를 활용하여, Front-End 개발을 보조하고, 개발 생산성을 향상시키는 방법을 학습하고자 합니다.
- **최소 2개 이상의 Prompt Engineering 기법을 활용하여 Front-End 개발에 활용해야 합니다.**
- 세부적인 내용은 다음과 같습니다:
    - **UI/UX 개선**
        - 디자인 및 style 추천
    - **문서화 및 주석 작성**
        - README.md 파일 작성 지원

        - 접근성 개선 제안
        - 사용자 경험 최적화 방안
        - 반응형 디자인 구현 조언
    - 개발 보조 도구로서의 활용 (Optional)
        - 코드 리뷰 및 개선 제안
        - 버그 발견 및 디버깅 지원
        - 코드 최적화 방안 제시
        - 개발 패턴 및 best practice 추천
        - 코드 문서화 및 주석 생성
        - API 문서 작성 보조 (Optional)
        - 커밋 메시지 작성 가이드 (Optional)
    - 테스트 코드 생성 (Optional)
        - 단위 테스트 케이스 작성
        - 테스트 시나리오 제안
        - 엣지 케이스 식별
        - 테스트 커버리지 개선 방안

h. 반응형 웹 구현

- 반응형 웹을 구현하여, 다양한 디바이스에서 웹사이트가 잘 보이도록 하는 방법을 학습하고자 합니다.
- 이를 위해, **반드시 모바일 환경에서도 사용이 가능하도록 구현**해야 합니다.
- 세부적인 내용은 다음과 같습니다:
    - **레이아웃 구성**
        - 그리드 시스템 활용
        - 플렉스박스 및 그리드 레이아웃 적용
        - 미디어 쿼리 활용
        - 반응형 이미지 처리
    - **사용자 경험 개선**
        - 터치 이벤트 지원
        - 접근성 향상 (Optional)
        - 화면 회전 대응
        - 폰트 크기 조절 지원
    - 디바이스 별 최적화 (Optional)
        - 모바일 최적화
        - 태블릿 최적화
        - 데스크탑 최적화
        - 다크 모드 지원
    - 성능 최적화 (Optional)
        - 이미지 최적화
        - 렌더링 최적화
        - 로딩 시간 단축
- **개인 확인 및 평가를 위해, 학생 개인 스마트폰에서 동작한 화면을 촬영하여 제출해주시길 바랍니다.**

3. 페이지 별 주요 기능

페이지 별 구현해야 할 주요 기능들은 다음과 같습니다. (평가 항목에 필수적으로 반영)

a. 모든 페이지 공통

- HTML component나 Style등은 제공해드린 템플릿(Angular.js 파일)을 그대로 활용해도 되지만, 차별성/독창성 여부에 따라 차등 점수를 부여함. **(상대평가)**
    - Angular 및 Demo Page와 동일한 Style 및 Animation 구현: 1점
    - Angular 및 Demo Page와 비슷한 Style 및 Animation 구현: 2점
    - Angular 및 Demo Page와 확연하게 다르게 Style 및 Animation 구현: 3점
    - Angular 및 Demo Page와 확연하게 다르며, fancy한 Style 및 Animation 구현: 4점
- fas, fab 아이콘을 적극 활용할 것
- 모든 페이지는 Vue.js 또는 React.js를 활용하여 개발해야하며, SPA의 형태로 구현해야 함.
- 모든 페이지는 Router를 활용하여 라우팅 처리를 해야하며, 페이지 이동 시에는 Transition 효과를 구현해야 함.
- 모든 페이지는 동적으로 구성해야하며, template을 활용하여 반복되는 부분은 Component로 구성해야 함. (정적인 페이지 구성 X)
- 모든 페이지는 **반응형으로 대응**해야하며, 모바일 환경에서도 사용이 가능해야 함.
- **영화 포스터를 보여주는 페이지는 아래 기능이 적용되어야 함.**
    - TMDB API를 활용하여 동적으로 영화 이미지 및 제목을 구성
    - 영화가 아직 불러지지 않으면, Loading 효과를 주도록 구현
    - 영화 포스터에 호버(hover) 시, 포스터가 확대되어 보여지도록 구현
    - 영화를 클릭 시, 추천 영화에 등록 되도록 함
    - 이미 추천된 영화는 다시 클릭 시, 추천 영화에서 삭제되도록 함

    - 추천된 영화는 Local Storage에 저장되어 있어야 함
    - 추천된 영화는 다른 디자인이 적용되어야 하며, 추천 영화 등록 시 바로 반영되어야 함.
- **header가 존재해야하며, 다음이 포함되어야 함.**
    - 로고 (클릭 시 /로 라우팅)
    - /, /popular, /search, /wishlist로 라우팅 하는 메뉴
    - ~~로그인 시, 로그아웃 버튼~~
    - ~~로그인 시, 사용자 아이디 표시~~
    - header는 animation을 포함해야하며, 구현하는 방법은 자유 (demo에서는 스크롤에 따라 투명도를 다르게 함)

b. 로그인 및 회원가입 페이지

- **경로**: **/signin**
- **반드시 존재해야하는 component**
    - ~~로그인 시, 아이디 입력~~
    - ~~로그인 시, 비밀번호 입력~~
    - ~~회원가입 시, 아이디 입력~~
    - ~~회원가입 시, 비밀번호 입력~~
    - ~~회원가입 시, 비밀번호 확인 입력~~
    - ~~로그인 버튼~~
    - ~~회원가입 버튼~~
    - ~~Remember me (로그인 시, 아이디 저장 및 자동 로그인)~~
    - ~~약관 동의 (회원가입 시, 필수 약관 동의)~~
- **로그인-회원가입 공통**
    - ~~로그인-회원가입 버튼 클릭 시, 컴포넌트 전환 효과 구현 (데모 및 예시 참고)~~
    - ~~아이디 입력 시, **email 형식**인지 확인하는 로직이 포함되어야 함.~~
    - ~~아이디와 비밀번호는 Local Storage에 저장되어 있어야 함~~
        - ~~**비밀번호는 TMDB의 API를 발급받아 사용**~~
        - ~~해당 API키를 사용하여, API를 호출하는 방식으로 구성~~
    - ~~로그인/회원가입 성공 시 메시지를 화면에 보여줘야 함.~~
        - alert() 함수 사용 시 기본 점수 부여
        - ~~Custom Toast (라이브러리) 사용 시 추가 점수 부여~~
    - **미들웨어/라우팅을 사용하여 URL로 접속 시, 로그인이 되어있는지 확인하고, 로그인이 되어있지 않으면 /signin 페이지로 이동하도록 함.**
- **로그인**
    - ~~로그인 성공 시, 메인 페이지(/)로 이동~~
    - ~~로그인 실패 시, 에러 메시지 출력~~
- **회원가입**
    - ~~회원가입 성공 시, 로그인 페이지 (/signin)로 이동 및 자동으로 로그인 창이 보여야 함.~~
    - ~~회원가입 실패 시, 에러 메시지 출력~~

c. 메인(home)페이지

- **경로**: **/**
- 최소 4개 이상의 TMDB API를 사용하여 영화 리스트를 호출하도록 함.
- **반드시 존재해야하는 component**
    - ~~영화 포스터~~
    - ~~영화 제목~~
    - 영화 설명 최소 1개 이상
    - ~~영화 평점 (Optional)~~
    - ~~영화 개봉일 (Optional)~~
    - ~~영화 장르 (Optional)~~
- 이 외에는 demo page와 비슷하게 구현하면 됨.
- 공통 항목에 있는 내용을 반드시 포함할 것 (추천 등)

- ~~디자인에 대한 평가 수행~~ **~~(상대평가)~~**

d. 대세 콘텐츠 페이지

- **경로**: /popular
- 다음과 같은 component 및 디자인이 포함되어야 함.
  - Table View 및 무한 스크롤을 선택할 수 있는 버튼
  - View 형식에 따른 영화 포스터
  - **Table View 일때,**
    - scroll이 불가능하게 설정
    - 모든 컨텐츠가 한 페이지 내에 다 보여야하며, 겹쳐서는 안됨
    - Pagination이 구성되어야 함
    - 현재 페이지 번호 및 이전/다음 페이지 버튼
  - **무한 스크롤 view 일때,**
    - scroll이 가능하게 설정
    - 스크롤이 끝에 도달하면, 다음 페이지의 영화 목록을 불러와야 함
    - 스크롤이 끝에 도달하면, Loading 효과를 주도록 구현
    - 맨 위로 올라가는 버튼이 존재해야 함 (Top)
- 이 외에는 demo page와 비슷하게 구현하면 됨.
- 공통 항목에 있는 내용을 반드시 포함할 것 (추천 등)
- 디자인에 대한 평가 수행 **(상대평가)**

e. 찾아보기 (search/filtering) 페이지

- **경로**: /search
- View 형식은 ~~Table~~ View 또는 Infinity scroll 중 자유
- ~~필터링 하는 UI 및 초기화 하는 버튼이 반드시 포함되어야 함.~~
- 필터링 예시:
  - ~~장르별 필터링~~
  - ~~평점별 필터링~~
  - ~~정렬 (sorting)~~
  - ~~기타 필터링 (예: 인기순, 개봉년도 등)~~
  - ~~필터링은 (1) API 자체로 걷드리거나 (2) 받아온 response를 map/filter 함수를 써서 구현하는 방식 중 자유롭게 하면 됨~~
- 이 외에는 demo page와 비슷하게 구현하면 됨.
- 공통 항목에 있는 내용을 반드시 포함할 것 (추천 등)
- 디자인에 대한 평가 수행 **(상대평가)**

f. 내가 찜한 리스트 (wishlist) 페이지

- **경로**: /wishlist
- ~~Local Storage에 저장된 추천 영화 목록을 불러와서 보여주어야 함.~~
- ~~View 형식은 Table View 또는 Infinity scroll 중 자유~~
- **~~해당 페이지에서는 API를 호출하면 안됨.~~**
- ~~이 외에는 demo page와 비슷하게 구현하면 됨.~~
- 공통 항목에 있는 내용을 반드시 포함할 것 (추천 등)
- 디자인에 대한 평가 수행 **~~(상대평가)~~**

g. 세부 항목에 적어둔 것 이외에는 자유롭게 구현하면 되며, demo page의 기능과 유사하게 구현하면 됩니다!! 🤗🥳😊

E. 평가 기준

세부적인 평가 항목들은 위 D-2와 D-3에 적혀있는 내용을 바탕으로 합니다. 다만, 세부 배점 등은 추후 공지 예정이니, 우선 D-2/D-3을 기준으로 개발하시면 될 것 같습니다 🥹

아래 내용은 변경될 수 있으니 개발하시면서 공지 꼭 확인해주세요!!

a. 필수 구현 항목

- **Front-End Framework를 통한 SPA 개발**
  - Vue.js 또는 React.js를 활용하여 프론트엔드 개발 수행
  - 동적 바인딩을 통한 Top-Down 방식의 데이터 전달
  - Bottom-Up 방식의 자식 컴포넌트에서 부모 컴포넌트로의 데이터/이벤트 전달
  - Iterative Rendering 및 Conditional Rendering을 활용한 동적 컴포넌트 구성
  - Javascript 또는 Typescript의 함수/클래스를 활용
  - Reference를 활용한 변수 선언 및 DOM Component 접근
  - View/ Component에 따른 폴더 구조 정형화
  - Router를 활용한 Single Page Application 라우팅 처리
  - Custom Hook/Composition API를 통한 로직 재사용성 향상
  - HTTP 클라이언트(Axios 등)를 활용한 REST API 연동
  - 상태 관리 라이브러리(Vuex/Redux)를 통한 전역 상태 관리 구현 (가산점)

- **배포 자동화 및 정적 웹사이트 호스팅**
- **Github 및 버전 관리**
- **The Movie Database (TMDB) 활용**
- **Local Storage 활용**
- **CSS Transition 및 Animation 활용**
- **GPT/Claude 등 AI 서비스 활용**

b. 점수표

- 개발한 사이트가 아래 기준을 충족할 경우 점수를 획득하며, 점수에 따라 해당 과제 학점이 결정됩니다.
- **점수-학점 기준**
  - 추후-공지

c. 상세 채점 항목

- 추후 공지

참고파일

1. Authentication.js

사용자 인증 관련 유틸리티 함수들을 포함하고 있습니다.

주요 기능

- **tryLogin(email, password, success, fail, saveToken)**: 사용자 로그인 처리

```
// 핵심 로직
const user = users.find(user => user.id === email && user.password === password);
if (user) {
    localStorage.setItem('TMDb-Key', user.password); // API 키 저장
    success(user);
}
```

- **tryRegister(email, password, success, fail)**: 회원가입 처리

```
// 핵심 로직
const userExists = users.some(existingUser => existingUser.id === email);
if (!userExists) {
    users.push({ id: email, password: password });
    localStorage.setItem('users', JSON.stringify(users));
}
```

2. URL.ts

TMDb API 관련 URL 생성 및 데이터 fetching 함수들을 포함하고 있습니다.

주요 기능

- **API 엔드포인트 구성:**

```
// 공통 매개변수
api_key=${apiKey}&language=ko-KR&page=${page}

// 주요 엔드포인트
/movie/popular          // 인기 영화
/movie/now_playing      // 현재 상영작
/discover/movie         // 장르별 영화
```

- **데이터 요청 처리:**

```
const fetchMovies = async (url: string): Promise => {
    const response = await axios.get(url);
    return response.data;
}
```

3. useWishlist.ts

위시리스트 관리를 위한 클래스를 정의합니다.

주요 기능

- **데이터 구조:**

```
interface Movie {
    id: number;
    title: string;
    poster_path: string;
}
```

- **핵심 메서드:**

```
class WishlistManager {
    // 위시리스트 토글 (추가/제거)
    toggleWishlist(movie: Movie): void {
        const index = this.wishlist.findIndex(item => item.id === movie.id);
        index === -1 ? this.wishlist.push(movie) : this.wishlist.splice(index, 1);
        this.saveWishlist();
    }

    // 로컬 스토리지 동기화
    private saveWishlist(): void {
        localStorage.setItem('movieWishlist', JSON.stringify(this.wishlist));
    }
}
```

공통 특징

> **1. 데이터 관리**
>
> - 모든 데이터는 로컬 스토리지에 저장/관리
> - JSON 형식으로 직렬화하여 저장
>
> **2. API 통신**
>
> - TMDb API 사용
> - 한국어(ko-KR) 응답 기본 설정
> - 페이지네이션 지원
>
> **3. 타입 안정성**
>
> - TypeScript 인터페이스 활용
> - 명시적 타입 선언으로 개발 안정성 확보