PJT명	**Axios 비동기 통신을 이용한 웹사이트 구현**	
단계	[Django 비동기 요청과 응답]	
진행일자	2025.11.28	
예상 구현 시간	필수기능	5H
	추가기능	2H
	심화기능	1H

1. 목표

- AJAX와 JSON에 대하여 이해할 수 있다.
- 비동기 통신을 이용하여 데이터를 생성, 조회 수정, 삭제할 수 있는 애플리케이션을 제작할 수 있다.
- Many to one relationship(N:1)에 대하여 이해한다.
- Many to many relationship(N:M)에 대하여 이해한다

2. 준비사항

1) 제공 파일

- 요구사항을 위한 기능 개발이 일부 진행된 Django 프로젝트
  - 초기 데이터를 제공하기 위한 fixture movies.json 포함

2) 개발언어 및 툴

- Python 3.11
- JavaScript
- Visual Studio Code
- Chrome Browser

3) 필수 라이브러리 / 오픈소스

- Django 5.2
- Bootstrap 5.3
- Axios

3. 작업 순서

1) 팀원과 같이 요구사항을 확인하고, GitLab에 프로젝트를 생성한다.
   - 프로젝트 이름은 09-pjt로 지정한다.
   - 각 반 담당 강사님을 Maintainer로 설정한다.
2) 팀원과 합의하여 협업 방식을 결정하고, 요구사항을 구현한다.
3) 작성한 코드들을 정리하고, README를 작성한다.
   - .gitignore 파일을 활용하여 불필요한 파일 및 폴더는 제출하지 않는다.
4) README 작성이 완료되면 심화 학습을 진행한다.
5) 제출 기한에 맞춰 모든 산출물이 GitLab에 업로드 될 수 있도록 한다.

4. 요구사항

추천 알고리즘을 통한 영화 추천 커뮤니티 서비스를 구축하려고 한다. 다양한 스트리밍 플랫폼에서 제공되는 영화 정보를 수집 및 관리하고, 이를 기반으로 개인화된 영화 추천, 장르별 영화 탐색, 유사 영화 추천 등 다채로운 추천 기능을 설계 및 구현한다. 또한 영화에 대한 사용자 리뷰 및 감상평 공유 커뮤니티 기능을 제공하여, 사용자들이 활발하게 소통하고 정보를 교환할 수 있는 기능을 제공한다. 사용자는 자신이 본 영화를 평가하고, 다른 사용자의 리뷰를 참고하여 다음 영화를 선택하는데 도움을 받을 수 있다. 나아가, 관심 영화 목록을 맞춤형으로 구성하는 등 다양한 편의 기능을 제공한다. 팀원과 상의하여 아래 요구사항을 만족할 수 있도록 요구 사항 명세서를 작성 및 구현해보자.

JavaScript를 활용한 클라이언트와 데이터를 관리하는 서버를 연결하고자 한다. 개발된 서버에 직접 요청을 보내어 새로고침 없이 데이터를 받아오는 사용자 클라이언트를 개발해보자.

- 요구사항 예시(참고용)
  - 아래의 내용을 참고하여 추가적인 아이디어에 대해 요구사항을 추가 또는 수정하여 기능을 구현한다. 단, **필수 기능은 구현해야 하며, 수정할 수 없다**.

번호	분류	요구사항명	요구사항 상세	우선순위
기능적 요구사항				
F01	기능 구현	유저 팔로우	특정 사용자를 '팔로우'할 수 있는 기능 구현	필수
F02	기능 구현	리뷰 좋아요	사용자가 남긴 리뷰를 '좋아요'할 수 있는 기능 구현	필수
F03	기능 구현	영화 장르 필터링	특정 UI와의 상호작용에 따라 출력되는 영화 데이터가 변경되는 기능 구현	필수
F04	AI 활용	영화 추천	AI를 활용하여 사용자가 다음 관람할 영화를 추천하는 알고리즘을 구현	도전
…	…	…	…	…
비기능적 요구사항				
NF01	프로젝트 관리	GIt 활용	개발자 간 작업 충돌이 일어나지 않게끔 Git을 활용하여 프로젝트 관리	필수
…	…	…	…	…

1) 프로젝트 관리 (필수)

Git의 기능을 적극적으로 활용하여, 여러 개발자가 동시에 개발할 수 있는 환경을 조성한다.

- 요구사항 번호: NF01
- 팀원과 합의하여 branch 생성 원칙 정리하기
  - Gitflow, GitHub Flow 등을 참고하여 원칙을 정할 수 있음
  - branch 운영 중 문제상황이 발생할 경우 원인에 대한 분석을 꼭 진행할 것
- 팀원과 합의하여 commit 내역 기록 원칙 정리하기
  - 기능 개발, 버그 수정 등 그 목적이 명확히 드러날 수 있도록 commit 메시지를 작성할 것
  - 어느 시점에 commit을 남길지에 대하여 충분한 합의를 이룰 것
- 프로젝트 종료 시 초기에 세운 원칙이 잘 지켜졌는지 점검
  - README에 초기에 세운 원칙을 설명하여 정리할 것
  - 생성했던 branch 들에 대한 용도를 정리할 것
  - 작성된 commit들을 GitLab에서 확인하여 스크린샷으로 README에 포함시킬 것

2) 기능 요구사항 (필수)

주어진 Django 프로젝트는 일부 기능이 이미 구현이 되어있는 상태이다. fixture를 바탕으로 초기 데이터를 불러온 다음, 요구사항에 맞춰 필요한 추가 기능을 구현해보자.

- 명시된 요구사항 이외에는 자유롭게 작성해도 무관하다.
- 원한다면 CSS, Bootstrap 등을 활용하여 추가 스타일링이 가능하다.
  - 출력 예시에서 사용된 UI 컴포넌트가 아니더라도 사용할 수 있다.
- 각 앱의 URL 매핑 참고
  - movies 앱의 URL

URL 패턴	**역할**
/movies/	전체 영화 목록 페이지 조회
/movies/filter-genre/	필터링 된 영화 데이터(JSON) 제공
/movies/recommended/	영화 추천 페이지 조회

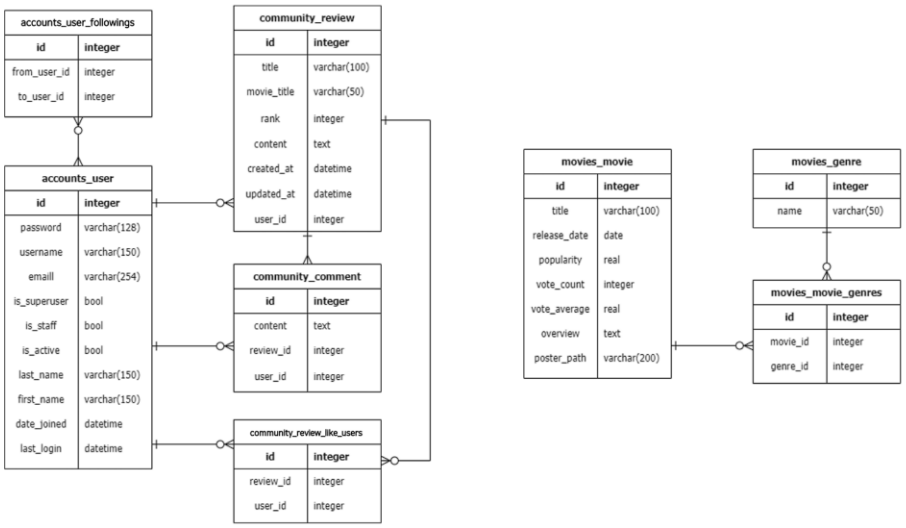
  - community 앱의 URL

URL 패턴	**역할**
/community/	전체 리뷰 목록 페이지 조회
/community/create/	리뷰 생성 페이지 & 단일 리뷰 저장
/community/<review_pk>/	단일 리뷰 상세 페이지 조회
/community/<review_pk>/comments/create/	단일 댓글 데이터 저장
/community/<review_pk>/like/	단일 리뷰 좋아요 기능

- accounts 앱의 URL

URL 패턴	역할
/accounts/signup/	회원 생성 페이지 & 단일 회원 데이터 생성 (회원가입)
/accounts/login/	로그인 페이지 & 세션 데이터 생성 및 저장 (로그인)
/accounts/logout/	세션 데이터 삭제 (로그아웃)
/accounts/profile/<username>/	사용자 상세 조회 페이지
/accounts/<user_pk>/follow/	사용자 팔로우 기능

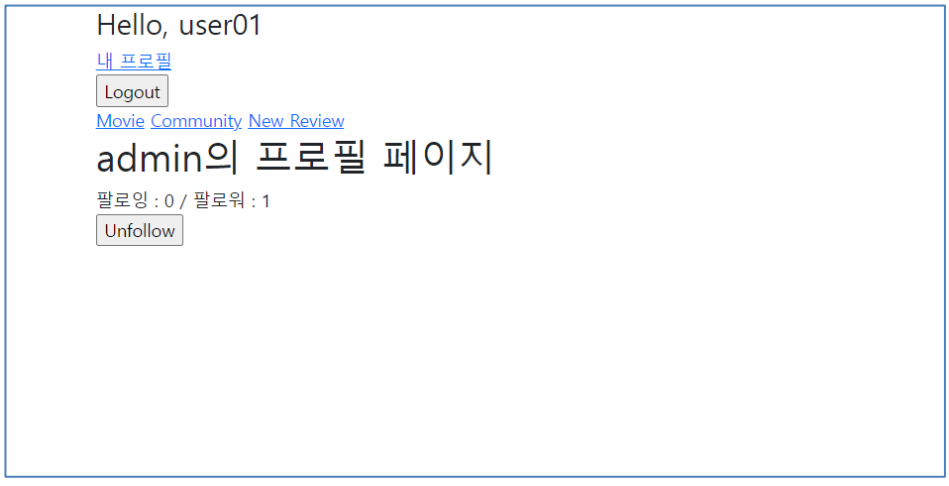
- ERD 참고



A. 유저 팔로우 기능

특정 사용자를 '팔로우' 하는 기능을 구현해보자.

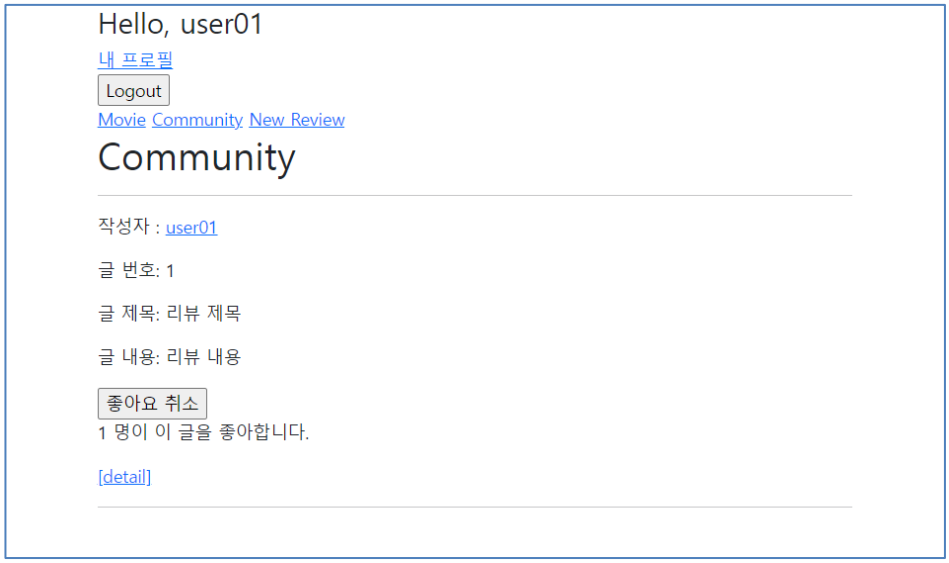
- 요구사항 번호: F01
- accounts 앱의 follow view 함수 활용
  - URL은 '/accounts/<user_pk>/follow/'로 구현되어 있음, accounts/urls.py 참조
- UI 등 일부 기능이 구현된 프로필 페이지 활용
  - accounts/profile.html
  - '/accounts/profile/<username>/'에서 확인 가능
  - '팔로우' 버튼, 팔로워 및 팔로잉 수가 표시 기능 구현되어 있음
- 인증된 사용자만 다른 사용자를 팔로우 할 수 있음
  - 자기 자신 팔로우 불가
- 팔로우 버튼을 클릭할 때 AJAX 기술을 이용하여 새로고침 없이 화면 내용을 구성
- 출력 예시



B. 리뷰 좋아요 기능

사용자가 작성한 리뷰를 '좋아요' 하는 기능을 구현해보자.

- 요구사항 번호: F02
- community 앱의 like view 함수 활용
  - URL은 '/community/<review_pk>/like/'로 구현되어 있음, community/urls.py 참조
- UI 등 일부 기능이 구현된 리뷰 목록 조회 페이지 활용
  - community/index.html
  - '/community/'에서 확인 가능
  - '좋아요' 버튼, 좋아요 수 표시 기능 구현되어 있음
- 인증된 사용자만 좋아요를 진행할 수 있음
- 좋아요 버튼을 클릭할 때 AJAX 기술을 이용하여 새로고침 없이 화면 내용을 구성
- 출력 예시

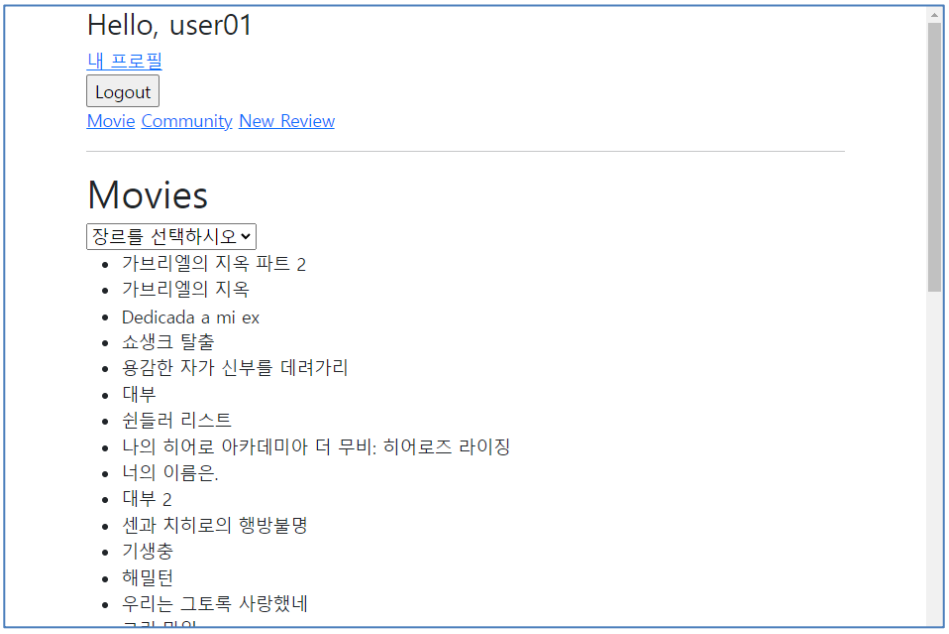


C. 영화 장르 필터링

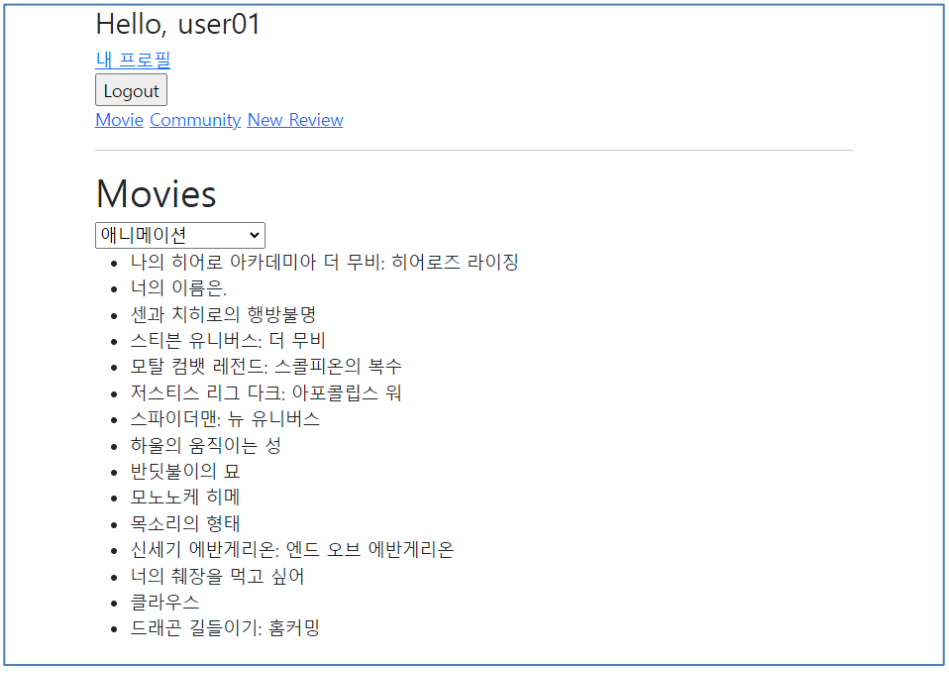
영화 목록을 장르에 따라 필터링하는 기능을 구현해보자.

- 요구사항 번호: F03
- movies 앱의 filter_genre view 함수 활용
  - URL은 '/movies/filter-genre/'로 구현되어 있음, movies/urls.py 참조
- 영화 목록 조회 페이지 구현 필요
  - URL '/movies/'로 접근
  - index view 함수 및 movies/index.html 구현 필요
  - 페이지 첫 접속 시 전체 영화를 출력하도록 구현
  - 장르 선택을 위한 UI는 자유롭게 구현
- 장르를 선택하였을 때, 선택한 장르에 맞는 영화 데이터 목록 필터링 후 출력
  - AJAX 기술을 이용하여 새로고침 없이 화면 내용을 구성

- 출력 예시
  - 장르 선택 전



  - 장르 선택 후



3) 도전 과제

생성형 AI 도구를 활용하여 도전과제 요구사항을 해결해보자.

- 코드 생성, 아이디어 구상, 문제 해결 방법 탐색 등 다양한 방식으로 활용할 수 있다.
- GMS 시스템을 통해 제공된 Key를 활용하여 AI를 서비스에 적용할 수 있다.
  - 사용할 생성형 AI 모델은 자유롭게 선택한다.
  - 기능 구현에 의한 크레딧 사용량에 유의할 것.
- 최종 결과물은 AI 생성 내용을 바탕으로 직접 수정 및 개선하여 적용해본다.

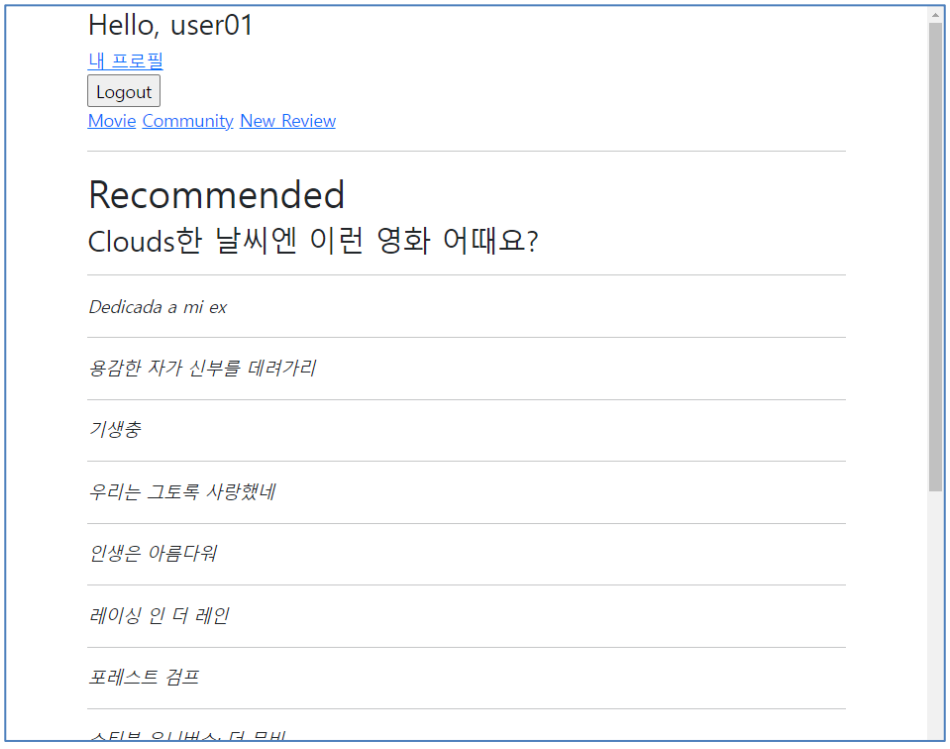
보조 수단으로 활용하되, 능동적인 자세로 학습에 임할 것.

- 최종적인 이해와 적용은 자기 주도적 학습을 통해 이루어지며, 배운 내용을 스스로 기록하고 정리하며 학습 효과를 높일 것.

A. 영화 추천 기능 구현

추천 알고리즘을 자유롭게 구상하여, 다음 관람할 영화를 골라주는 기능을 구현해보자.

- 요구사항 번호: F04
- movies 앱의 recommended view 함수 활용
  - URL은 '/movies/recommended/'로 구현되어 있음, movies/urls.py 참조
- 영화를 추천하는 알고리즘은 자유롭게 구상
  - 외부 API 활용 가능
- 구현한 알고리즘에 대한 설명은 README.md에 작성
  - 구현이 어렵다면 아이디어를 상세히 정리해서 작성
- 출력 예시
  - OpenWeatherMap API를 활용한 날씨 기반 영화 추천 사례



5. 참고자료

- Django

  https://www.djangoproject.com/start/overview/
- MDN

  https://developer.mozilla.org/en-US/
- Bootstrap

  https://getbootstrap.com/

6. 결과

제출 기한은 진행일 18시까지이므로 제출 기한을 지킬 수 있도록 한다. 제출은 GitLab을 통해서 이뤄진다.

- **산출물과 제출**
  - 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점을 상세히 기록한 README.md
  - 완성된 각 문제 별 소스코드 및 실행 화면 캡쳐본
  - 프로젝트 이름은 09-pjt로 지정 및 각자 제출
    - 한 명의 GitLab 계정에 Git 저장소 Push 및 작업
    - 나머지 인원은 제출 시 해당 저장소를 Fork 하여 제출
  - 각 반 담당 강사님을 Maintainer로 설정

- 끝 -