07 - 데이터베이스 설계

1. 목표

- 협업을 통한 데이터베이스 모델링 및 기능 구현
- 다양한 형태의 데이터베이스 관계 설정

2. 준비 사항

1. **(필수)** Python Web Framework
    - Django 2.2.x
    - Python 3.7.x
2. (선택) 샘플 영화 정보

3. 요구 사항

1. **데이터베이스 설계**

- `db.sqlite3` 에서 테이블 간의 관계는 아래와 같습니다.
    - `accounts_users`

필드명	자료형	설명
id	Integer	Primary Key
username	String	필수
password	String	필수
email	String	선택
first_name	String	선택
last_name	String	선택

    - `movies_movies`

필드명	자료형	설명
id	Interger	Primary Key
title	String	영화명
audience	Integer	누적 관객수
poster_url	String	포스터 이미지 URL
description	Text	영화 소개
genre_id	Integer	Genre의 Primary Key(id 값)

- `movies_genres`

필드명	자료형	설명
id	Interger	Primary Key
name	String	장르 구분

- `movies_reviews`

필드명	자료형	설명
id	Integer	Primary Key
content	String	한줄평(평가 내용)
score	Integer	평점
movie_id	Integer	Movie의 Primary Key(id 값)
user_id	Integer	User의 Primary Key(id 값)

- `movies_like_movies_user`

필드명	자료형	설명
id	Integer	Primary Key
user_id	Integer	User의 Primary Key(id 값)
movie_id	Integer	Movie의 Primary Key(id 값)

2. **Seed Data 반영**
    1. 주어진 `movie.json` 과 `genre.json` 을 `movies/fixtures/` 디렉토리로 옮깁니다.
    2. 아래의 명령어를 통해 반영합니다.

```
$ python manage.py loaddata genre.json
Installed 11 object(s) from 1 fixture(s)
$ python manage.py loaddata movie.json
Installed 10 object(s) from 1 fixture(s)
```

3. admin.py 에 `Genre` 와 `Movie` 클래스를 등록한 후, `/admin` 을 통해 실제로 데이터베이스에 반영되었는지 확인해봅시다.

3. **`accounts` App**
   - 유저 회원가입과 로그인, 로그아웃 기능을 구현해야 합니다.
   1. 유저 목록 (`/accounts/`)
      1. **(필수)** 사용자의 목록이 나타나며, 사용자의 `username` 을 클릭하면 `유저 상세보기` 페이지로 넘어갑니다.
   2. 유저 상세보기 (`/accounts/{user_pk}/`)
      1. **(필수)** 로그인한 사람만이 사용자의 상세 내용 페이지에서 해당 유저를 follow하거나 unfollow를 할 수 있습니다.
      2. **(필수)** 해당 유저가 작성한 평점 정보가 모두 출력됩니다.
      3. **(필수)** 해당 유저가 좋아하는 평점 정보가 모두 출력됩니다.
      4. (선택) 각각 평점을 수정할 수 있도록 구성합니다.
4. **`movies` App**
   - Genre와 영화는 생성/수정/삭제를 만들지 않습니다. 단, 관리자를 위하여 관리자 계정과 함께 관리자 페이지를 생성합니다.
   1. 영화 목록(`/movies/`)
      1. **(필수)** 영화의 이미지를 클릭하면 `영화 상세보기` 페이지로 넘어갑니다.
   2. 영화 상세보기(`/movies/{movie_pk}/`)
      1. **(필수)** 영화 관련 정보가 모두 나열됩니다.
      2. **(필수)** 로그인 한 사람만 영화 평점을 남길 수 있습니다.
      3. **(필수)** 모든 사람은 평점 목록을 볼 수 있습니다.
      4. **(필수)** 영화가 존재 하지 않는 경우 404 페이지를 보여줍니다.
   3. 평점 생성
      1. **(필수)** 영화 평점은 로그인 한 사람만 남길 수 있습니다.
      2. **(필수)** 평점 생성 URL은 `POST /movies/1/reviews/new/` , `POST /movies/2/reviews/new/` 등 이며, 동적으로 할당되는 부분이 존재합니다. 동적으로 할당되는 부분에는 데이터베이스에 저장된 영화 정보의 Primary Key가 들어갑니다.
      3. **(필수)** 검증을 통해 유효한 경우 데이터베이스에 저장을 하며, 아닌 경우 `영화 정보 조회 페이지` 로 Redirect 합니다.
      4. **(필수)** 데이터베이스에 저장되면, 해당하는 영화의 `영화 상세보기` 페이지로 Redirect 합니다.
      5. **(필수)** 영화가 존재 하지 않는 경우 404 페이지를 보여줍니다.
   4. 평점 삭제
      1. **(필수)** 영화 평점 삭제는 본인만 가능합니다**.**
      2. **(필수)** 평점 삭제 URL은 `POST /movies/1/reviews/1/delete/` , `POST /movies/1/reviews/2/delete/` 등 이며, 동적으로 할당되는 부분이 존재합니다. 동적으로 할당되는 부분에는 데이터베이스에 저장된 영화 정보의 Primary Key와 평점의 Primary Key가 들

어갑니다.
3. **(필수)** 데이터베이스에서 삭제되면, 해당하는 영화의 `영화 상세보기` 페이지로 Redirect 합니다.
4. **(필수)** 영화가 존재 하지 않는 경우 404 페이지를 보여줍니다.

5. 영화 좋아요 기능 구현
    1. **(필수)** 좋아하는 영화를 담아 놓을 수 있도록 구현합니다.
    2. **(필수)** 로그인 한 유저만 해당 기능을 사용할 수 있습니다.
    3. **(필수)** 영화 좋아요 URL은 `POST /movies/1/like/` 등 이며, 동적으로 할당되는 부분이 존재합니다. 동적으로 할당되는 부분에는 데이터베이스에 저장된 영화 정보의 Primary Key가 들어갑니다.
    4. **(필수)** 적합한 위치에 좋아요 링크를 생성합니다.
    5. **(필수)** 영화가 존재 하지 않는 경우 404 페이지를 보여줍니다.

4. 결과 예시

Python Web Framework를 활용해 작성한 모든 파일을 `project07` 디렉토리에 위치하도록 합니다.

결과물은 반드시 `README.md` 으로 활용 하였던 내용을 정리 해주세요.

```
project07/
    ...
    ...
    ...
    README.md
```