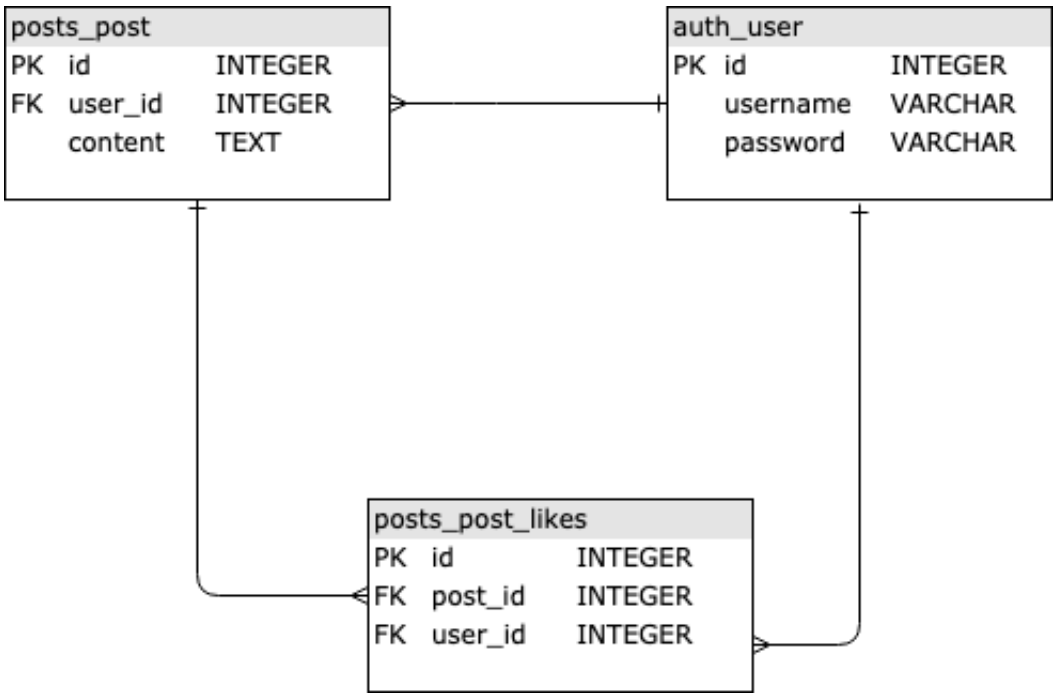
4월말평가

0. 준비

데이터베이스 모델링은 다음과 같습니다.



1. User모델은 장고에서 기본적으로 제공하는 User 모델을 사용하기 때문에 추가 구현을 하지 않습니다.
2. Post모델의 user 칼럼은 User모델과 1:N관계로 연결되어있습니다.
3. Post모델의 like_users 칼럼은 User모델과 M:N관계로 연결되어있습니다.
4. 프로젝트는 마이그레이션이 되어있는 상황이며, DB를 삭제하고 다시 마이그레이션 할 수 있습니다.

1. 테스트를 위한 유저정보는 다음과 같습니다.
    1. 기본계정(`username` : ssafy, `password` : ssafy)
    2. `createsuperuser` 명령어를 사용하여 user를 추가 가능합니다.

1. 회원가입(45점)

> 주의 : 로그인은 view에서 함수로 보이지 않을 뿐 내부적으로 이미 구현이 되어 있습니다.
>
> accounts/login/ URL로 접근하면 로그인 가능합니다.
>
> **지정된 위치를 제외한 코드는 변경하실 필요 없습니다.**

1. **(25점)** `accounts/views.py` 파일 signup 함수내부의 Q1-1 주석부분에 코드를 작성하시오.
    1. `GET` 방식의 `accounts/signup/` URL로 요청이 들어올때 회원가입을 할 수 있도록 `signup.html`

파일을 반환합니다.
2. `UserCreationForm` 을 활용하여 구현합니다.

2. **(20점)** `accounts/views.py` 파일 signup 함수내부의 Q1-2 주석부분에 코드를 작성하시오.
    1. `POST` 방식의 `accounts/signup/` URL로 요청이 들어올때 데이터베이스에 사용자의 정보를 저장하여 회원가입 기능을 완성합니다.
    2. `UserCreationForm` 을 활용하여 구현합니다.
    3. 데이터의 유효성 검사가 이루어져야 합니다.
    4. 유저가 저장이 된 후 `/posts/` URL로 리다이렉트합니다.

예시 결과

1. 회원가입 버튼을 클릭했을 때의 화면은 다음과 같습니다.

로그인 회원가입

회원가입

Username:
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:
Enter the same password as before, for verification. 제출

2. 게시물작성(30점)

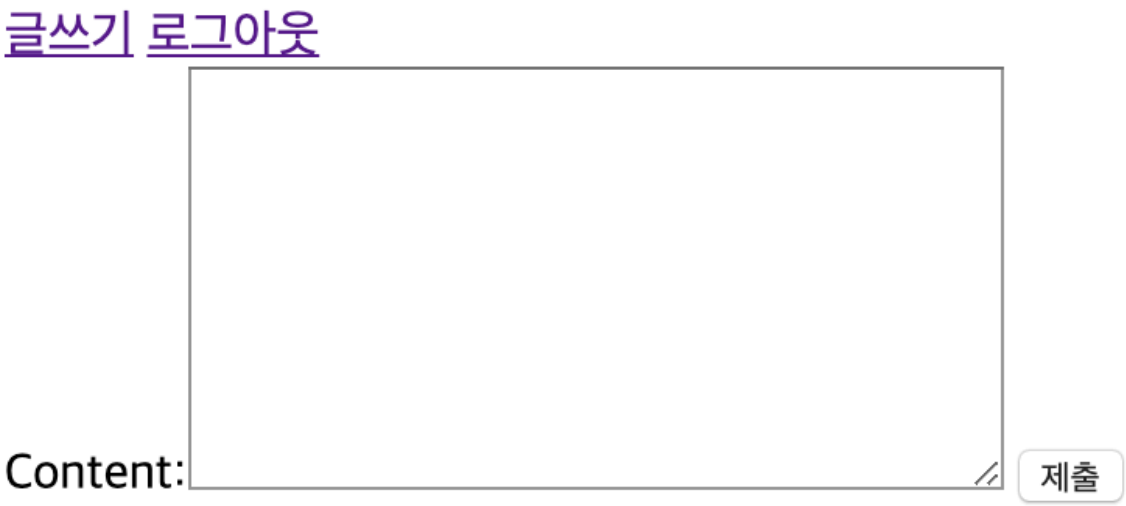
> 회원가입이 완성되지 않았다면,
>
> 기본유저정보 혹은 `createsuperuser`를 이용하여 생성된 유저정보를 이용하여 테스트 할 수 있습니다.
>
> **지정된 위치를 제외한 코드는 변경하실 필요 없습니다.**

1. **(10점)** `posts/forms.py` 파일 내부의 Q2-1 주석부분에 코드를 작성하시오.
    1. Post 모델을 작성하기위한 모델폼을 정의합니다.
    2. 모델폼의 이름은 `PostForm` 으로 정의합니다.
    3. 모델폼을 이용하여 데이터를 넣을 칼럼은 `content` 입니다.

2. **(10점)** `posts/views.py` 파일 내부의 Q2-2 주석부분에 코드를 작성하시오.
    1. `GET` 방식의 `posts/create/` URL로 요청이 들어올때 글을 작성 할 수 있도록 `form.html` 파일을 반환합니다.
    2. `forms.py` 안에 정의한 `PostForm` 을 이용하여 구현합니다.
3. **(10점)** `posts/views.py` 파일 내부의 Q2-3 주석부분에 코드를 작성하시오.
    1. `POST` 방식의 `posts/create/` URL로 요청이 들어올때 게시물을 작성하는 기능을 완성합니다.
    2. `forms.py` 안에 정의한 `PostForm` 을 이용하여 구현합니다.
    3. 데이터의 유효성 검사가 이루어져야 합니다.
    4. 게시물에 로그인한 사용자의 정보가 저장되어야 합니다.
    5. 게시물이 저장이 된 후 `/posts/` URL로 리다이렉트합니다.

예시 결과

1. 글쓰기 버튼을 클릭했을 때의 화면은 다음과 같습니다.



3. 좋아요(20점)

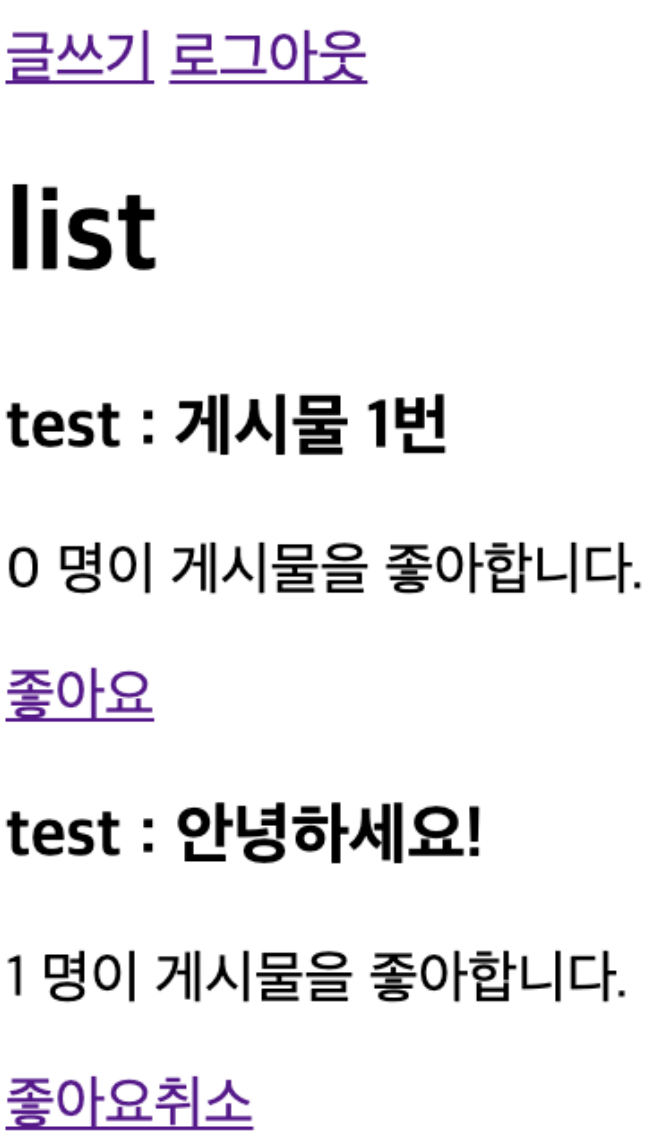
> 회원가입이 완성되지 않았다면,
>
> 기본유저정보 혹은 `createsuperuser`를 이용하여 생성된 유저정보를 이용하여 테스트 할 수 있습니다.
>
> **지정된 위치를 제외한 코드는 변경하실 필요 없습니다.**

1. **(10점)** `posts/views.py` 파일 내부의 Q3-1 주석부분에 코드를 작성하시오.
    1. like 함수를 수정하여 좋아요 기능을 완성합니다.
    2. M:N 관계가 추가 되어있지 않으면 Post모델과 로그인한 유저를 M:N으로 추가합니다.
    3. M:N 관계가 추가되어 있으면 Post모델과 로그인한 유저의 M:N관계를 제거합니다.
    4. like 함수가 끝난 후 `/posts/` 로 리다이렉트 합니다.
2. **(5점)** `posts/templates/posts/list.html` 파일 내부의 Q3-2 주석부분에 코드를 작성하시오.

    1. 좋아요를 요청할 수 있는 링크 완성합니다.
    2. `a` 태그 안의 `href`에 경로를 작성할때는 `posts/urls.py` 를 참고하여 `{% url %}` 태그를 활용하여 작성합니다.
    3. M:N관계가 추가되지 않았다면 `좋아요` 링크를 보여줍니다.
    4. M:N관계가 추가되어 있다면 `좋아요취소` 링크를 보여줍니다.
3. **(5점)** `posts/templates/posts/list.html` 파일 내부의 Q3-3 주석부분에 코드를 작성하시오.
    1. 예시 결과 이미지를 참고하여 `list.html` 에 해당 게시글에 몇개의 좋아요가 눌렸는지 출력합니다.
    2. `<p>` 태그사이에 좋아요 누른 인원 수를 출력합니다.

예시 결과

1. 좋아요 버튼은 다음과 같습니다.



4. 사용자 인증(5점)

1. **(5점)** `posts/views.py` 파일 내부에 코드를 작성하시오.
    1. `create` 함수와 `like` 함수가 로그인 한 유저만 실행 할 수 있도록 데코레이터를 추가합니다.

5. 제출

- 강사의 가이드에 따라서 프로젝트 폴더를 압축하여 제출합니다.