1. project 생성
2. app 생성
    1. 생성 후 settings.py 에서 등록
    2. 프로젝트의 urls.py 에서 앱으로 url을 보낼 수 있도록 include 하기
    3. JSON 데이터를 가져오는거만 하는게 아니면 해당 앱에서 urls.py 에서 앱 등록
3. Movie app +comment
    1. model 작성
        1. 예를 들어 movie를 작성한다고 하면 title,description,audience,Release_date,score, genre 등등 작성
    2. form 작성
        1. 해당 앱에 forms.py 를 생성하여 forms.modelform를 통해 클래스를 생성
    3. index 작성
        1. 기본 페이지 작성
    4. create 작성
        1. get 요청의 경우와 post 요청의 경우에 맞게 데이터 보내주기
    5. detail 작성
        1. movie_pk에 맞는 영화를 html로 보내주기
    6. delete 작성
    7. update 작성
        1. create 와 마찬가지로 get 요청과 post 요청에 맞게 데이터 보내기
        2. 여기서 주의 할 것은 기존 데이터에 수정을 하는것이기에
        3. modelform(request.POST, instance=request.data) 하기
    8. comment
        1. comment 작성을 위한 model 수정
            1. comment model 요소
                1. movie(FK), content, created_at, updated_at
            2. 여기서 FK 작성 유의
                1. comment에서 movie를 참조하기에
                2. ForeignKey를 사용하고
                3. 첫번째 요소로 Movie 클래스를 넣어주고
                4. 다음으로는 제약조건으로 models.CASCADE 를 넣어준다.
        2. comment 생성
            1. urls.py 에 movie_pk 값과 함께 comment_create 를 만든다.
            2. 그리고 views.py에 코멘트 생성을 위해 POST 요청에 맞게 작성
            3. POST
                1. movie를 찾기 위해 pk 값에 맞는 movie를 찾아준다
                2. movie_form 에는 request.POST 값을 담은 ModelForm을 보내준다.
                3. comment_form의 유효성 검사 후
                4. comment_form.save() 하기 전에 (commit=False) 하여 comment에 담아주고
                5. comment.movie= movie를 하여 어떤 영화의 댓글인지 넣어주고

6. comment.save() 해주기

4. def detail 에서 comment 를 보여줄 수 있도록 하기
    1. movie를 먼저 movie_pk 에 맞는걸 담아주고
    2. comment_form에 ModelForm을 담아서 보내주고
    3. comments = movie.comment_set.all() 하여 담아준다
        - 여기서 _set을 하는 이유는 영화 입장에서 댓글은 역참조하는 것이기 때문.
    4. context에 담아주기
    5. 이후 detail.html 에서 if 문을 통해 comment 가 있다면 보여주고
    6. for 문을 통해 comments 안에서 comment 를 하나씩 꺼내 보이기
    7. comment.content 로 댓글 내용 보이도록 하기

3. comment 삭제
    1. url에서 movie_pk와 comment_pk 를 함께 담은 url 만들기
    2. view에서는 movie를 찾을 필요 없이
    3. comment = Comment.objects.get(pk=comment_pk)를 하여 찾고
    4. comment.delete()를 해주고
    5. redirect를 통해 detail.html 에 movie_pk를 보내준다.
    6. 이후 datail.html 에서 comment를 삭제할때는 comment 보이는 하단에
    7. form 태그를 통해 movie.pk와 comment.pk 를 함께 보내 POST 요청 보내기

4. accounts app
    1. settings.py
        1. User 또한 Movie 처럼 settings.py 에 앱등록을 해주기
        2. 추후에 User 모델을 재정의 할 때를 위해
        3. AUTH_USER_MODEL = accounts.User 를 작성해주기
    2. urls.py 에 include하여 accounts에도 url 이동 시키도록 하기
    3. models.py
        1. 기존에 정의되어 있는 User를 재정의 하기
        2. from django.contrib.auth.models import AbstractUser 받아오기
        3. User(AbstractUser) 이렇게만 정의하고 pass 해두기
    4. forms.py
        1. 이전에 정의한 models.py 에서 User를 불러오기
        2. from django.contrib.auth.forms import UsercreationForm, UserChangeForm 을 받기
        3. class 명은 CustomUserCreationForm, CustomUserChangeForm
        4. UserCreationForm은 User 생성
        5. UserChangeForm은 User 수정
        6. 여기서 Meta 또한 (UserCreationForm.Meta), (UserChangeForm.Meta) 로 정의
        7. Meta의 model
            1. 현재 활성화된 User 모델을 반환하도록
            2. django.contrib.auth import auth_user_model을 받아온뒤 적용

8. 여기서 각각의 fields는 각각 UserCreationForm.Meta.fields, UserChangeForm.Meta.Fields 로 정의

5. signup
    1. url
    2. view
        1. GET
            1. form = CustomUserCreationForm() 으로 Form 보내기
        2. POST
            1. form= CustomUserCreationForm(request.POST)
            2. form의 유효성 검사 후
            3. user = form.save() 하여 저장하여 담아준 뒤
            4. 로그인을 바로 할 수 있도록
                1. django.contrib.auth import login as auth_login 을 불러오기
                2. auth_login(request, user)
6. login
    1. url
    2. view
        1. GET
            1. 로그인 페이지를 보여주기 위해서
                1. from django.contrib.auth.form import AuthenticationForm 불러오기
            2. form = AuthenticationForm() 을 담아서 보내기
        2. POST
            1. **form = AuthenticationForm(request, request.POST) 담아주기**
            2. form의 유효성 검사 후
            3. auth_login(request, form.get_user()) 담아주기
            4. 그래서 login 할때는 앞에 request를 꼭 담아주는 것을 기억하기.
    3. html
        1. 입력한 값을 POST method로 보낼 수 있도록 form태그에 감싸주기
        2. 받아온 form 을 {{form.as_p}} 로 p태그에 감싸줘서 보여주기
        3. POST 요청이기에 csrf_token 담아서 보내주고
        4. input 태그를 통해 전송하기
7. logout
    1. url
    2. view
        1. 우선 logout을 하기 위해
            1. from django.contrib.auth import logout as auth_logout 불러오기
        2. 요청한 유저가 인가된 유저인지 확인을 위해 if문을 이용
            1. if request.user.is_authenticated: 로 확인 후
            2. auth_logout(request) 으로 로그아웃
8. profile
    1. url
        1. 일반적으로 인스타를 생각해서 <str:username >/ 을 전달받기

2. view
    1. 우선 생각할 것
        1. profile 은 나만의 프로필이 아니라는 것
        2. 그래서 현재 활성화 된 유저 모델에서 유저를 찾아야 한다는 것.
    2. User = get_user_model() 에 담아준뒤
    3. person = User.objects.get(username=username) 후 context에 담아주기
3. html
    1. person .username 으로 유저의 이름을 보여주기

9. update
    1. url
        1. 주의: user_pk를 받아오지 않는다
            1. 이유는 현재 로그인 되어있는 유저의 정보가 계속 있기 때문??
    2. view
        1. GET
            1. 우선 변경하는 것이기 떄문에
                1. from .form import CustomUserChangeForm 을 불러오기
            2. form = CustomUserChangeForm(instance=request.user) 를 보내주기
        2. POST
            1. form = CustomUserChangeForm(request.POST, instance=request.user) 로 담아주기
                1. 그냥.... 외워야지 뭐... 하...
            2. form의 유효성 검사 후
            3. form.save() 하여 저장
    3. html
        1. 변경 데이터를 입력 후 전송할 수 있도록 form 태그 작성
            1. method는 POST 이기에 csrf_token 넣고 input으로 전송
        2. 받아온 form 을 p 태그에 담아서 보여주도록 {{form.as_p}} 로 감싸주기

10. change_password
    1. url
    2. view
        1. password 변경을 위한 form 을 가져오기 위해
            1. from django.contrib.auth.form import PasswordChangeForm 불러오기
        2. GET
            1. form = PasswordChangeForm(request.user) 를 context에 담아보내기
        3. POST
            1. form = PasswordChangeForm(request.user, request.POST) 담고
            2. 유효성 검사 후
            3. 갱신 후 로그인 유지
                1. from django.contrib.auth import update_session_auth_hash 를 불러오기
                2. update_session_auth_hash(request, form.user) 넣어주기

                1. .... 이걸 어쨔 외움...
    3. html
        1. 변경 데이터를 입력 후 전송할 수 있도록 form 태그 작성
            1. method는 POST 이기에 csrf_token 넣고 input으로 전송
        2. 받아온 form 을 p 태그에 담아서 보여주도록 {{form.as_p}} 로 감싸주기

11. follow
    1. model
        1. 이전에 작성했던 User Class에 follow 속성 추가
            1. follow는 N:M 속성으로 자기자신을 참조한다
                1. followings = models.ManyToManyField('self', symmetrical=False, related_name='followers') 로 작성
                2. 여기서 symmetrical는 똑같이 적용하지 않기위해 False로 작성
                3. related_name은 자기자신을 참조중이기 때문에 구별하기 위해서
                4. 아마도??
    2. url
        1. user_pk 담아서 보내주기
    3. view
        1. 우선 요청한 유저가 인가된 유저인지 확인
            1. if request.user.is_authenticated:
        2. 현재 활성화 된 유저의 모델을 가져오기 위해
            1. User = get_user_model()
        3. 나와 타인을 구별하여 불러오기
            1. me
                1. me = request.user
            2. you
                1. you= User.objects.get(pk=user_pk)
        4. 만약 me와 you가 서로 다를 경우 if me != you:
            1. you의 followers 중에 me 가 현재 있다면
                1. if you.follower.filter(pk=me.pk).exist():
                2. you.follower.remove(me)
            2. 없다면
                1. you.follower.add(me)
        5. me와 you가 같을 경우 else:
            1. 프로필 페이지로 이동시키기
                1. you.username도 함께 보내기
                2. 이유는 잘... 그냥 you에 대한 pk값을 가져와서??
        6. 만약 인가된 유저가 아닌 경우
            1. 로그인 페이지로 보내주기
    4. html
        1. profile.html 변경
        2. profile 페이지에서 현재 팔로워와 팔로잉 표현을 하기위해 profile에서 받아온 person을 활용
        3. follower

        1. person.follower.all|length
    4. following
        1. person.following.all|length
    5. 그리고 아래에서 request.user와 person이 같지 않은 경우
        1. input 태그를 통해 팔로우와 언팔로우를 누를 수 있도록 하기
        2. if문을 통해
            1. request.user가 person.follower.all에 포함된다면
                1. input태그, value='언팔'
            2. 포함되지 않는다면
                1. input태그, value='팔로우'