<과제 2 피드백정리>

(기타 등등은 수정하고)

* User Seminar 클래스에 user랑 seminar를 합쳐
  unique constraint 걸어주기.

* related_name 활용하기
  역참조 상황에서 여러개나 아니냐 구분해주면 훨씬 쉬울것

* serializer Method Field 는 read_only!
  required 무관 ⇐ 왜..?

* select_related 랑 prefetch_related

  캐싱을 활용하여 반복적인 쿼리를 줄여주고 INNER JOIN 사용

  Onetoone / foreignkey 에서는 select_related

  ManytoMany / Manytoone 에서는

  → field 차이말고 두개가 어떻게 작동하는 측면에서 차이를 보이는지!

* validation error는 400이니까 이걸 활용하자

* AllowAny 랑 IsAuthenticated, IsAuthenticatedOrReadonly ...
  구분하고 적절하게 사용하기

☆
* UserAuth가 User에 role이라는 column을 부여해주고자
  사용했는데 굳이 그럴 필요가 없나..?

* Serializer inherit 해보기
  - 자식 serializer은 부모의 Meta class를 상속받지 X
    ex) class Meta(부모serializer.Meta):
    로 명시해야함!

  - 부모 class의 field를
    ex) my_field = None 으로
    제거 가능

* 모든 Model field를 serializer field로 정의해 줄 필요x
  custom 하는 경우에만!

* http method 사용x
  get ⇒ list()로!

* 특정 table의 id(등 기타 field)에 접근하기위해 source사용
  ex) serializers. IntegerField (source='seminar.id').

* get_serializer_class

```
ex) def get_serializer_class(self):
        if self.request.user.is_staff:
            return ASerializer
        return BSerializer.
```

* hasattr 과 if~ is not None 을 적절히 사용해서
  선택적으로 field값 변형

* save 코드 최소화

* updated_at 을 주의 (실제값이 바뀌지않아도)

* allow_blank = True : "" 빈칸을 key의 value로 넣을때

* users.permissions 파일을 새로정의

views 파일 깔끔하게

```
ex) class IsParticipant(permissions.BasePermission):
        def has_permission(self, request, view):
            return hasattr(request.user, 'participant').
    class IsInstructor( 이하동일 )
```

1. 400을 response에 내보냈어야
   되는데, validation error를 내보낸 field가 않음.
   ex) year, accepted...

2. 개별적으로 role을 auth model의 일부분으로
   가져오는 것으로 serializer를 잘못 구성함.

3. role이나 accepted의
   값이 있거나 없을수 있는데
   값 비교를 위한 깔끔한 조건이 부족했음.

4. 더 participater seminar는 없을시 그곳
   하고, instructor의 seminar는 null로 하는지?
   participater seminar는 처음부터 생성이 되어야 함.
   instructor는 그렇지 않아서..?

* 테스트할 API

1. POST /api/v1/user/

- role이 잘못된 경우 (안쓰거나 instructor/participant 둘다 아닌경우)
- university.accepted / company.year이 비어있을때
- year이 음의 정수
- role과 맞지 않는 정보

중복 username
이메일 안쓴경우

2. PUT /api/v1/user/login/

- 로그인 하면 토큰 주는지

3. PUT /api/v1/user/me

- role 잘못된경우
- year 음의정수
- role과 맞지 않는정보
- instructor / participant 각각의정보

① instructor로 test
- company.year. firstname/ Lastname. email 등이 비어져있을때
- role 비어있거나 잘못되어있을때
- year. company 등
- university. accepted 등 정보.

② 로그인
- Token 비교.

③ db 자체에서 잘 들어갔는지 비교

4. GET /api/v1/user/{user-id}/

- date_joined / participant의 id/university/accepted/
  - seminar id/ name / joined_at / is_active / dropped_at
  - instructor의 id / company / year/ charge seminar id/name/ joined_at

} 이건 seminar api 확인하고 내서!

제대로 들어가있는지! 없는경우 null / [ ]으로 되어있는지

5. GET /api/v1/user/me

- token 보고 알아서 위의 정보랑 동일하게 잘 가져오는지!

6. POST /api/v1/user/participant

7. POST /api/v1/seminar/ 응답201.

participant
instructor) 생성.

1) name, capacity, count, time 없으면 400
2) online 을 안적으면 True
3) online false 로 해보기
4) name 에 0글자. capacity, count이 0 이하인수. time 이 잘못된 format일 경우 → 400
5) online FALSE로 해보기
6) participant인 진행자가 요청.
7) post한 세미나의 charge에 이 세미나가 들어갔는지!

8. PUT /api/v1/seminar/{seminar_id}/ response 200
1) empty body
2) is_active가 true인 participant보다 적은수의 capacity로 바꿀때
   < i) is_active 가 모두 True일때
     ii) is_active False인 user 고려 (delete 기능 구현하고!).
3) seminar_id 없는경우 (404)
4) 세미나 참여자인 경우 403
5) instructor지만 해당 세미나 담당자가 아닌경우 403

9. GET /api/v1/seminar/{seminar_id}/
1) seminar_id에 해당하는 seminar 없는경우
2) 해당하는 token이 없는경우
3) 제대로 된경우 정보 맞는지 확인하기!

10. GET /api/v1/seminar/
1) 전체세미나 가져왔을때 비교
2) name parameter test [filter / 없을때 [ ] 로 가져오는지
3) order parameter test [아무것도 안했을때 제대로 오는지 (최신순) / earliest 정렬되는지 / order에 이상한 값들때
4) name이나 order 빼고 다른 query 들때

11. POST /api/v1/seminar/{seminar_id}/user/

1) ~~role이 instructor나 participant가 아닌경우 400~~

2) 한 사람이 A세미나 참가 B세미나 담당

3) ~~" A세미나에 참가 후 담당 / 담당 후 참가~~) 400

4) ~~없는 세미나 아이디인 경우 404~~

5) ~~자격이 없는경우 403 (profile)~~ < participant가 instructor / instructor가 participant

6) ~~accepted가 false인 경우 403~~

7) ~~세미나가 가득찬 경우 400~~

8) ~~성공적인 경우 정보비교 & 200 & db에 저장됐는지!~~
(is_active 가 true인지, joined_at이 제대로 들어갔는지)

① participant 111 A
② instructor 999 B
③ participant | A 담당
instructor | B 참여
④ 1919 → norole

---

12. DELETE /api/v1/seminar/{seminar_id}/user/

1) ~~세미나 id가 없을경우 404~~

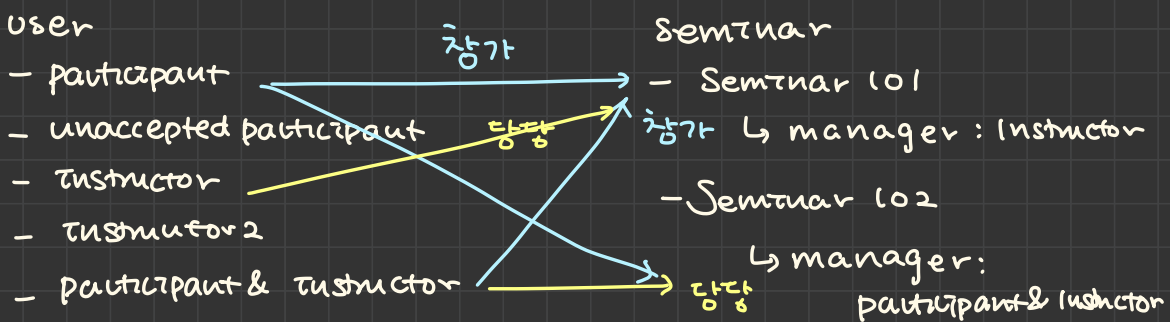
2) ~~정상처리 → is_active false.~~
~~→ dropped_at 확인~~
~~→ http 200~~

3) ~~다시 참여했을때 400.~~

4) ~~세미나에 참여하지 않은경우 무시 (body랑 200확인)~~

5) ~~Instructor가 drop하려는 경우 400.~~

~~drop후 테스트해야할것 dropped_at이 null에서 바뀌었는지~~
~~is_active가 false인 user 고려해서 capacity 찼을때~~


User
- participant
- unaccepted participant
- instructor
- instructor2
- participant & instructor
참가
당연
Seminar
- Seminar 101
참가 ↳ manager : instructor
- Seminar 102
↳ manager:
participant & instructor
당연