Shamseya
for Innovative Community Healthcare Solutions

Backend-challenge

Intro:

A review has 4 answers.

An answer is constructed by which review is this answer for, which question and which choice.

A question has a least two choices or more.

We can use these models to represent the previous statements:

```
class Review(models.Model):
    submitted_at = models.DateTimeField()

class Choice(models.Model):
    text = models.CharField(max_length=20)

class Question(models.Model):
    text = models.TextField()
    choices = models.ManyToManyField(Choice, related_name='questions')

class Answer(models.Model):
    review = models.ForeignKey(Review, on_delete=models.CASCADE, related_name='answers')
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice = models.ForeignKey(Choice, on_delete=models.CASCADE)
```

Warm-up:

You should create 4 questions, each question has at least two choices or more.

You should create at least 4000 reviews with their answers to seed the database, `submitted_at` values are between `1/1/2018` and `31/12/2020` .

Feel free to do this any way you like, but don't forget to include what you did to create these data.

Task:

You are asked to write and design an endpoint that returns the corresponding reviews with their answers in a given 'from/to' date period with the following requirements:

- date periods should be flexible, for example: if `from` date wasn't given, `to` date or no dates were given at all.
- duplicate dates are not allowed to appear in the results if there is more than one review with the same date these reviews should be merged into one object and all corresponding answers with these reviews should be listed with this one review object.
- `submitted_at, answers, count` fields are required for each review object in the results.
- `count` field is the number of reviews on this date, for ex: it will have the value of 1 when there is one review on this date if there is more than one review on this date it should have the value of their count.
- **your ORM query to construct your queryset should be optimized, this is important here, so do this with the best way you can.**

Requirements:

you should use django-rest-framework to build the endpoint, otherwise, feel free to use any other packages you need.

you should use PostgreSQL as your database for this task.

you should write test cases for your endpoint.

you shouldn't expose any secrets in your codebase.

Bonus:

- the endpoint should be accessed by authenticated users only.
- create 3 users using the built-in user module in Django, make the first one superuser only, the second one staff only and the last one is active only. now, do whatever is necessary to make the endpoint only accessed by a superuser or a staff user and otherwise a user is not allowed.
- do a benchmark of your endpoint performance when there are 4000, 8000, 12000, 16000, 20000 reviews, is there any performance issues? how it could be improved?.
- when merging reviews at the same date into one object answers gets repeated under the new review object, can you merge answers also to become as one and add a new field for their count?
- dockerize your application (ignore adding the DB backup file as long as your DB will be dockerized with the data)

Submit-requirements:

- backup your DB with all the data it has and adds the backup file along with your app files.
- create a `readme.md` file in the root of your app that shows your app requirements, what did you use and why, how to build it and run it,... etc.
- create a public git repo on any platform you like `ex: GitHub, GitLab,... etc` and share the link of this repo when you are ready.

Note:

Don't be discouraged if you didn't finish the task at the given time, submit what you have done.