در این مخزن یک برنامه DRF برای مقصودی که خواسته شده نوشته شده است.

- در نمایش لیست مطالب می‌توان با درخواست GET تمامی مطالب را با تعداد کاربرانی که امتیاز داده‌اند و میانگین آن امتیازها مشاهده کرد. هر مطلب یک uuid منحصر به فرد دارد که به جای id کلید اصلی به کاربران نمایش داده می‌شود. علت استفاده از uuid و عدم نمایش id در پاسخ درخواست GET آن است که در آن حالت، چون id ها به صورت خودکار یکی یکی اضافه می‌شوند، می‌توان با پیمایش روی id و رسیدن به تمامی مطالب، مشکلاتی مثل حمله‌ی امتیاز دادن به تمام مطالب ایجاد کند.
  با درخواست POST نیز می‌توان مطالبی را با عنوان و محتوا به لیست مطالب اضافه کرد.
  برای جلوگیری از افت performance، از pagination های پیش‌فرض viewset در DRF استفاده شده، که این کار بار شبکه را در نمایش پاسخ درخواست لیست مطالب پایین می‌آورد و همچنین از واکشی تمام رکوردهای مطلب از پایگاه داده روی حافظه‌ی اصلی (RAM) سرور در حال اجرای برنامه جلوگیری می‌کند.

- در نمایش ثبت امتیاز می‌توان از درخواست‌های POST و PUT برای ساختن و به‌روز رسانی امتیاز از طرف کاربر به یک مطلب استفاده کرد.
  برای کم‌شدن زمان میانگین‌گیری از به‌روزرسانی لحظه‌ای میانگین امتیاز هر مطلب، هنگام ایجاد رکورد امتیاز یک کاربر به یک مطلب استفاده شده است. این کار که نیازی به query های اعمال دیتابیسی مثل average و count که روی کل رکوردها پیمایش می‌کنند، ندارد. در این روش به جای جمع تمام امتیازهای کاربران به یک مطلب و تقسیم آن بر تعداد امتیازها، از اثری که امتیاز جدید در همان لحظه روی میانگین مطلب گذاشته استفاده می‌شود.

در حالت ساخت امتیاز جدید:

$$new\ average\ = \frac{average * \ rating\ count + new\ score}{rating\ count + 1}$$
$$new\ rating\ count = \ rating\ count + 1$$

در حالت به‌روز رسانی امتیاز:

$$new\ average\ = \frac{average * \ rating\ count - old\ score + new\ score}{rating\ count}$$

برای جلوگیری از اثر امتیاز هیجانی در یک بازه زمانی کوتاه علاوه بر فرمول قبلی از یک حداقل زمان برای امتیاز دادن به مطالب استفاده می‌شود به این صورت که با دادن هر امتیاز تایم به‌روزرسانی مطلب عوض می‌شود و این تغییر زمان اگر از یک بازه‌ی ۱۰ ثانیه‌ای کم‌تر باشد، ضریبی یک صدمی در امتیاز جدید ضرب می‌شود و از اثر آن امتیاز بر میانگین امتیازات یک مطلب در آن زمان حمله می‌کاهد. البته این اقدام پس از یک حد آستانه کافی تعداد امتیاز (مثلا ۱۰ بار) برای یک مطلب فعال می‌شود و برای مطلبی که کمتر از آن حد آستانه، امتیاز گرفته باشد این مکانیزم فعال نمی‌شود.

کد زیر پیاده‌سازی روش گفته‌شده را نشان می‌دهد.

```python
def update_post_avarage(self, post, score, old_score=None):
    if post.ratings_count > 10 and timezone.now() - post.updated_at < timedelta(seconds=10):
        score *= 0.01
    if old_score is not None:
        post.average_score = ((post.average_score * post.ratings_count) - old_score + score) / post.ratings_count
    else:
        post.average_score = ((post.average_score * post.ratings_count) + score) / (post.ratings_count + 1)
        post.ratings_count += 1
    post.save()
```