# Django Review Rating Backend - Documentation

## • Project Setup Guide

Create a virtual env and Activate it.

### 1. Create Django Project and App

```
django-admin startproject mainfold
django-admin startapp reviewsys

Directory structure

backend/
├── mainfold/
│
│
├── reviewsys/
│
├── venv/
├── .gitattributes
├── .gitignore
├── db.sqlite3
├── manage.py
├── README.md
├── requirements.txt
```

### 2. Install Required Packages

```
pip install django djangorestframework transformers torch nltk
django-cors-headers
```

### 3. Configure Settings (`mainfold/settings.py`)

Add `rest_framework` and `reviewsys` to INSTALLED_APPS

```
INSTALLED_APPS = [
     ………
    'django.contrib.staticfiles',
    'rest_framework',
    'corsheaders',
    'reviewsys',
]
```

Add CORS middleware and configuration

```
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    ………
]
```

Set up database (SQLite by default)

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

## 4. Create Database Model (`reviewsys/models.py`)

Review model fields:

`review_text` → `TextField`

`predicted_rating` → `IntegerField`

`created_at` → `DateTimeField(auto_add=True)`

`Models.py`

```
from django.db import models

class Review(models.Model):
    review_text = models.TextField()
    predicted_rating = models.IntegerField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.predicted_rating}- {self.review_text[:30]}"
```

## 5. Create Serializer (`reviewsys/serializers.py`)

Create `ModelSerializer` for:

Data validation

JSON conversion

Read-only `created_at` field

```
class ReviewSerializer(serializers.ModelSerializer):
    class Meta:
        model = Review
        fields = ['id','review_text','predicted_rating','created_at']
        read_only_fields=['created_at']
```

## 6. Create Views (`reviewsys/views.py`)

Implemented 4 API views:

- ReviewPredictionView → `POST` → Create review and predict rating

To Load model and evaluate the output:

```
try:

   tokenizer = AutoTokenizer.from_pretrained(MODEL_PATH, local_files_only=True)
   model = AutoModelForSequenceClassification.from_pretrained(MODEL_PATH,local_files_only=True)
   model.eval()

   inputs = tokenizer(review_text, return_tensors="pt", truncation=True)
   with torch.no_grad():
   outputs = model(**inputs)
   predicted_rating = torch.argmax(outputs.logits,dim=1).item()+1
```

- ReviewListAPIView → `GET` → List all reviews

```
class ReviewListAPIView(APIView):
    def get(self, request):
        reviews = Review.objects.all().order_by('-created_at')
        paginator = PageNumberPagination()
        paginator.page_size = 8
        paginated_reviews = paginator.paginate_queryset(reviews, request)
        serializer = ReviewSerializer(paginated_reviews, many=True)
        return paginator.get_paginated_response(serializer.data)
```

Included pagination. For that we need to add below in mainfold/settings.py

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 8,
}
```

- RecentReviewsAPIView → `GET` → Get 3 most recent reviews

```
recent_reviews = Review.objects.all().order_by('-created_at')[:3]
```

- ReviewStatsAPIView → `GET` → Get count of each ratings,total - count and AVG rating

```
try:
            rating_counts = (
                Review.objects.values('predicted_rating')
                .annotate(count=Count('predicted_rating'))
                .order_by('-predicted_rating')
            )
      rating_counts_dict = {item['predicted_rating']: item['count'] for item in rating_counts}
            rating_data = [
                {"star": i, "count": rating_counts_dict.get(i, 0)}
                for i in range(5, 0, -1)
            ]
            avg_rating =
Review.objects.aggregate(avg_rating=Avg('predicted_rating'))['avg_rating']
```

## 7. Configure URLs (`reviewsys/urls.py` + `mainfold/urls.py`)

Set up API routing:

```
/api/reviews/create/

/api/reviews/

/api/reviews/recent/

reviews/detail/
```

```
urlpatterns = [
    path('reviews/create/', ReviewPredictionView.as_view(), name='review-create'),
    path('reviews/', ReviewListAPIView.as_view(), name='review-list'),
    path('reviews/recent/', RecentReviewsAPIView.as_view(), name='recent-reviews'),
    path('reviews/detail/', ReviewStatsAPIView.as_view(), name='review-stats'),
]
```

## 8. Download NLTK Data

```
python -c "import nltk; nltk.download('words')"
```

## 9. Run Migrations

```
python manage.py makemigrations
python manage.py migrate
```

## 10. Run the Application

```
python manage.py runserver
```

## • API Testing with Postman

### 1. Create Review with Prediction

POST `http://127.0.0.1:8000/api/reviews/create/`

Headers:

```
Content-Type: application/json
```

Body (raw JSON):

```
{
   "review_text": "This product is absolutely amazing!"
}
```

Expected Response (201 Created):

```
{
   "id": 1,
   "review_text": "This product is absolutely amazing!",
   "predicted_rating": 5,
   "created_at": "2025-08-18T12:34:56.789Z"
}
```

### 2. Get All Reviews

GET http://127.0.0.1:8000/api/reviews/

Expected Response (200 OK):

```
[
   {
      "id": 1,
      "review_text": "This product is absolutely amazing!",
      "predicted_rating": 5,
      "created_at": "2025-08-18T12:34:56.789Z"
   }
]
```

## 3. Get Recent Reviews

GET http://127.0.0.1:8000/api/reviews/recent/

Expected Response (200 OK):

```
[
    {
        "id": 1,
        "review_text": "This product is absolutely amazing!",
        "predicted_rating": 5,
        "created_at": "2025-08-18T12:34:56.789Z"
    }
]
```

## 4. Get Details

GET http://127.0.0.1:8000/api/reviews/detail/

Expected Response (200 OK):

```
{
    "ratings": [
        {
            "star": 5,
            "count": 17
        },
        {
            "star": 4,
            "count": 11
        },
        {
            "star": 3,
            "count": 10
        },
        {
            "star": 2,
            "count": 3
        },
        {
            "star": 1,
            "count": 15
        }
    ],
    "average_rating": 3.2,
    "total_reviews": 56
}
```