## Project Overview

**Project Name**: sentiment\_project  
**App Name**: sentiment\_app

**Functionality**:

* Accept user input (text)
* Predict sentiment using a pretrained ML model (e.g., TextBlob, VADER, or your own model)
* Store the result in MySQL:
  + Original Text
  + Predicted Sentiment
  + Timestamp

## Requirements

### Python Packages

Install the required packages:

pip install django mysqlclient textblob

python -m textblob.download\_corpora

On Windows, you may need to install MySQL development headers before mysqlclient will install. Alternatively, you can use PyMySQL (less performant).

## 1.Create Django Project and App

django-admin startproject sentiment\_project

cd sentiment\_project

python manage.py startapp sentiment\_app

## 2.Configure settings.py

### 📌 MySQL Configuration

In sentiment\_project/settings.py, add:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'sentiment\_db',

'USER': 'your\_mysql\_user',

'PASSWORD': 'your\_mysql\_password',

'HOST': 'localhost',

'PORT': '3306',

}

}

Also add 'sentiment\_app' to INSTALLED\_APPS:

INSTALLED\_APPS = [

...

'sentiment\_app',

]

## 3.Define Model

In sentiment\_app/models.py:

from django.db import models

class SentimentResult(models.Model):

text = models.TextField()

sentiment = models.CharField(max\_length=20)

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return f"{self.sentiment} - {self.text[:50]}"

Run migrations:

python manage.py makemigrations

python manage.py migrate

## 4. Sentiment Analysis Function

You can use TextBlob for basic sentiment analysis:

In sentiment\_app/sentiment.py:

from textblob import TextBlob

def analyze\_sentiment(text):

blob = TextBlob(text)

polarity = blob.sentiment.polarity

if polarity > 0:

return "Positive"

elif polarity < 0:

return "Negative"

else:

return "Neutral"

## 5. Create Form

In sentiment\_app/forms.py:

from django import forms

class SentimentForm(forms.Form):

text = forms.CharField(widget=forms.Textarea(attrs={'rows': 4}))

## 6. Views and URLs

### views.py

from django.shortcuts import render

from .forms import SentimentForm

from .models import SentimentResult

from .sentiment import analyze\_sentiment

def predict\_sentiment(request):

result = None

if request.method == 'POST':

form = SentimentForm(request.POST)

if form.is\_valid():

text = form.cleaned\_data['text']

sentiment = analyze\_sentiment(text)

SentimentResult.objects.create(text=text, sentiment=sentiment)

result = sentiment

else:

form = SentimentForm()

return render(request, 'sentiment\_app/index.html', {'form': form, 'result': result})

### urls.py in app and project

**sentiment\_app/urls.py**

from django.urls import path

from .views import predict\_sentiment

urlpatterns = [

path('', predict\_sentiment, name='predict\_sentiment'),

]

**sentiment\_project/urls.py**

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('sentiment\_app.urls')),

]

## 7. Template

Create file: sentiment\_app/templates/sentiment\_app/index.html

<!DOCTYPE html>

<html>

<head>

<title>Sentiment Predictor</title>

</head>

<body>

<h1>Sentiment Prediction</h1>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Predict</button>

</form>

{% if result %}

<h2>Prediction: {{ result }}</h2>

{% endif %}

</body>

</html>

## 8. Run Server

python manage.py runserver

Go to http://127.0.0.1:8000/ to test!

## 9. Admin Panel (Optional)

Register the model:

**admin.py**

from django.contrib import admin

from .models import SentimentResult

admin.site.register(SentimentResult)

Create superuser:

python manage.py createsuperuser

## MySQL Setup Reminder

Make sure:

* MySQL is running
* The sentiment\_db database is created:
* CREATE DATABASE sentiment\_db;