LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSc

Submitted By

Name:- Bhairav Ganguly

Roll No: 2230246

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Exp No.	Title	Date of Experiment	Date of Submission	Remarks
1.	1. 2. 3.			
2.				
3.				
4.				
5.				
6.				
7.				
8.	Sentiment Prediction API Using FastAPI and X (formrtly Twitter) Tweets	25/3/2025	31/3/2025	
9.	Open Ended 1			
10.	Open Ended 2			

Experiment Number	8
Experiment Title	Sentiment Prediction API Using FastAPI and X (formerly Twitter) Tweets
Date of Experiment	25/3/2025
Date of Submission	31/3/2025

github: https://github.com/Bhairavg7/AD_LAB_8.git

1. Objective:- The objective of this lab experiment is to create a sentiment prediction API using FastAPI, which analyzes Twitter tweets for positive, negative, or neutral sentiment. This lab integrates natural language processing (NLP) techniques with a lightweight and high-performance API framework.

2. Procedure:-

1. Install the required Python libraries: FastAPI, Tweepy, TextBlob, scikit-learn, pandas, and uvicorn.
2. Create a X Developer account.
3. Create a new application to obtain API keys
4. Use the Tweepy library to authenticate with the Twitter API.
5. Write a function to search for tweets containing a specific keyword or hashtag.
6. Fetch a specified number of recent tweets and return their text and metadata.
7. Use TextBlob or a similar NLP library to perform sentiment analysis on tweet text.
8. Define categories for sentiment (e.g., Positive, Negative, Neutral) based on the
polarity score.
9. Create a function that takes text as input and returns the sentiment category.
10. Initialize a FastAPI application.
11. Define endpoints:
    1. A root endpoint (e.g., /) to confirm the API is running.
    2. A POST endpoint (e.g., /fetch_tweets/) to accept user inputs such as keyword and number of tweets to fetch.
12. Ensure the /fetch_tweets/ endpoint integrates the tweet-fetching and sentiment analysis functions.
13. Run the API using uvicorn in development mode (--reload flag for auto-updates).
14. Use a tool like Postman, CURL, or a web browser to test:
    1. The root endpoint for a welcome message.

2. The POST endpoint by providing a sample keyword and tweet count in the request payload.

15. Verify the output includes fetched tweets with their respective sentiment analysis.

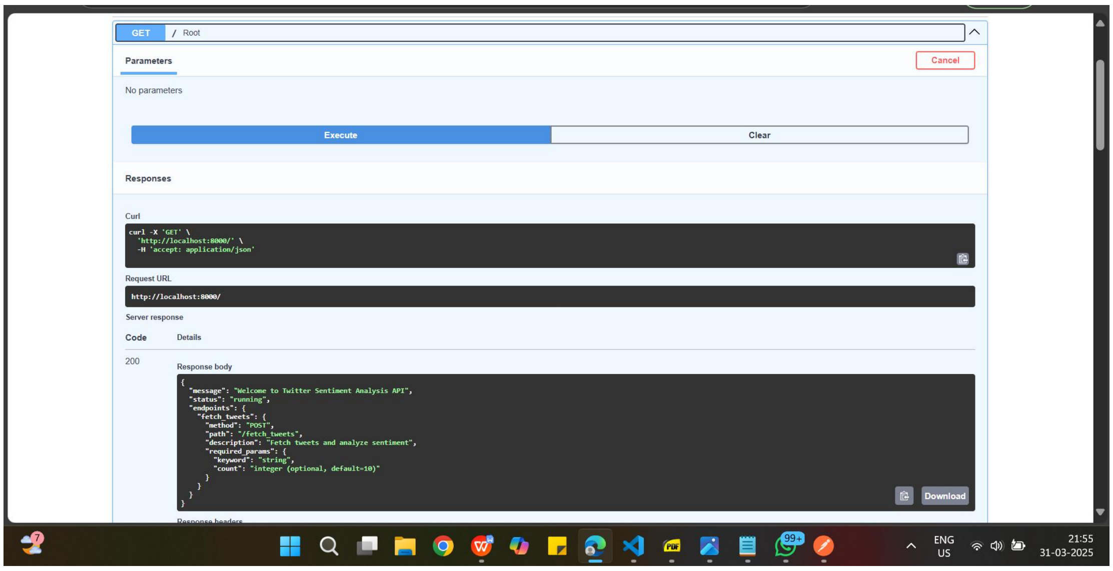
3. Code:-

main.py

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import tweepy
from textblob import TextBlob
import os
from dotenv import load_dotenv
from typing import List, Dict, Optional

# Load environment variables
load_dotenv()

# Initialize FastAPI app
app = FastAPI(
    title="Twitter Sentiment Analysis API",
    description="API for fetching tweets and performing sentiment analysis",
    version="1.0.0"
)

# Twitter API credentials
TWITTER_API_KEY = os.getenv("TWITTER_API_KEY")
TWITTER_API_SECRET = os.getenv("TWITTER_API_SECRET")
TWITTER_ACCESS_TOKEN = os.getenv("TWITTER_ACCESS_TOKEN")
TWITTER_ACCESS_TOKEN_SECRET = os.getenv("TWITTER_ACCESS_TOKEN_SECRET")
TWITTER_BEARER_TOKEN = os.getenv("TWITTER_BEARER_TOKEN")

# Initialize Twitter client
client = tweepy.Client(
    bearer_token=TWITTER_BEARER_TOKEN,
    consumer_key=TWITTER_API_KEY,
    consumer_secret=TWITTER_API_SECRET,
    access_token=TWITTER_ACCESS_TOKEN,
    access_token_secret=TWITTER_ACCESS_TOKEN_SECRET
)

# Request model
class TweetRequest(BaseModel):
    keyword: str
    count: int = 10  # Default to 10 tweets if not specified

# Response models
class SentimentResult(BaseModel):
    text: str
    polarity: float
    sentiment: str  # "positive", "negative", or "neutral"
```

```
class TweetResponse(BaseModel):
    keyword: str
    count: int
    tweets: List[SentimentResult]
    positive_count: int
    negative_count: int
    neutral_count: int

# Helper function to analyze sentiment
def analyze_sentiment(text: str) -> Dict:
    analysis = TextBlob(text)
    polarity = analysis.sentiment.polarity

    if polarity > 0:
        sentiment = "positive"
    elif polarity < 0:
        sentiment = "negative"
    else:
        sentiment = "neutral"

    return {
        "text": text,
        "polarity": polarity,
        "sentiment": sentiment
    }

# Root endpoint
@app.get("/", tags=["Root"])
async def root():
    return {
        "message": "Welcome to Twitter Sentiment Analysis API",
        "status": "running",
        "endpoints": {
            "fetch_tweets": {
                "method": "POST",
                "path": "/fetch_tweets",
                "description": "Fetch tweets and analyze sentiment",
                "required_params": {
                    "keyword": "string",
                    "count": "integer (optional, default=10)"
                }
            }
        }
    }

# Fetch tweets endpoint
@app.post("/fetch_tweets", response_model=TweetResponse, tags=["Tweets"])
async def fetch_tweets(request: TweetRequest):
    try:
        # Fetch recent tweets
        tweets = client.search_recent_tweets(
            query=request.keyword,
            max_results=request.count,
            tweet_fields=["text"]
        )
        if not tweets.data:
            return {
                "keyword": request.keyword,
                "count": 0,
                "tweets": [],
                "positive_count": 0,
                "negative_count": 0,
                "neutral_count": 0
            }

        # Analyze sentiment for each tweet
        analyzed_tweets = []
        positive_count = 0
        negative_count = 0
        neutral_count = 0

        for tweet in tweets.data:
            result = analyze_sentiment(tweet.text)
            analyzed_tweets.append(result)

            # Count sentiments
            if result["sentiment"] == "positive":
                positive_count += 1
            elif result["sentiment"] == "negative":
                negative_count += 1
            else:
                neutral_count += 1

        return {
            "keyword": request.keyword,
            "count": len(analyzed_tweets),
            "tweets": analyzed_tweets,
            "positive_count": positive_count,
            "negative_count": negative_count,
            "neutral_count": neutral_count
        }

    except tweepy.TweepyException as e:
        raise HTTPException(status_code=400, detail=f"Twitter API error: {str(e)}")
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"An error occurred: {str(e)}")

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

4. Results/Output:-



No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/
```

Server response

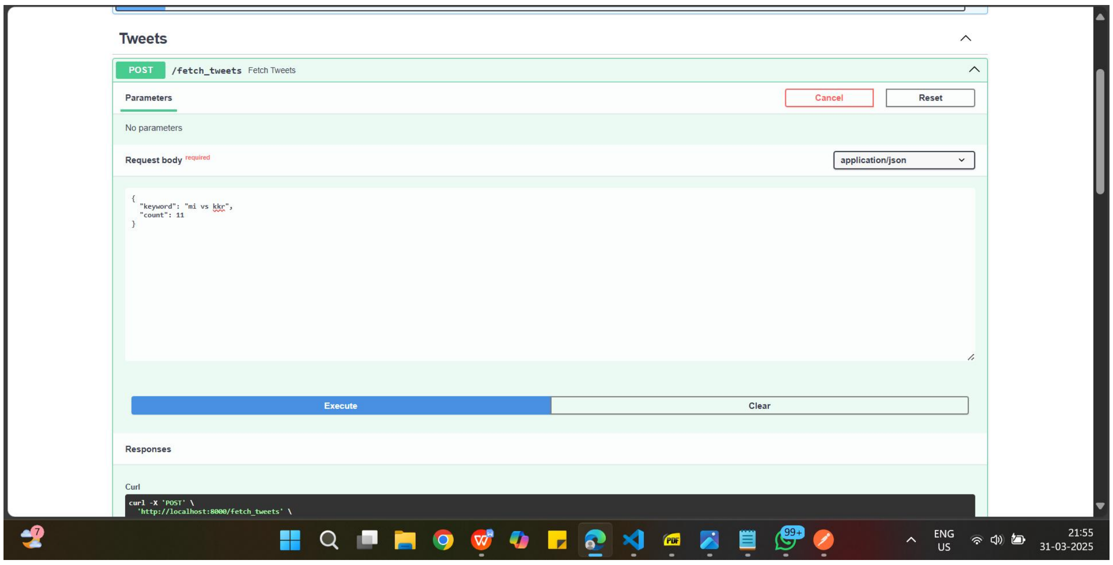
Code Details

200

Response body

```
{
  "message": "Welcome to Twitter Sentiment Analysis API",
  "status": "running",
  "endpoints": {
    "fetch_tweets": {
      "method": "POST",
      "path": "/fetch_tweets",
      "description": "Fetch tweets and analyze sentiment",
      "required_params": {
        "keyword": "string",
        "count": "integer (optional, default=10)"
      }
    }
  }
}
```

Response headers



No parameters

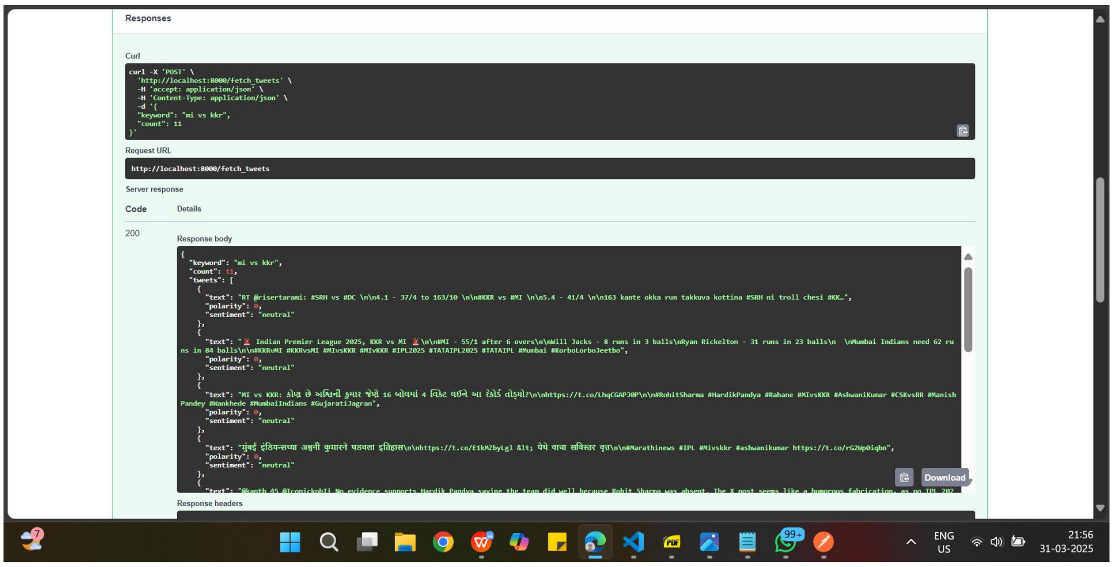
Request body required

```
{
  "keyword": "mi vs kkr",
  "count": 11
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/fetch_tweets' \
```



Curl

```
curl -X 'POST' \
  'http://localhost:8000/fetch_tweets' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "keyword": "mi vs kkr",
  "count": 11
}'
```

Request URL

```
http://localhost:8000/fetch_tweets
```

Server response

Code Details

200 Response body

```
{
  "keyword": "mi vs kkr",
  "count": 11,
  "tweets": [
    {
      "text": "RT @risertarami: #SRH vs #DC \n\n4.1 - 37/4 to 163/10 \n\n#KKR vs #MI \n\n5.4 - 41/4 \n\n163 kante okka run takkuva kottina #SRH ni troll chesi #KK…",
      "polarity": 0,
      "sentiment": "neutral"
    },
    {
      "text": "🚨 Indian Premier League 2025, KKR vs MI 🚨\n\n#MI - 55/1 after 6 overs\n\nWill Jacks - 8 runs in 3 balls\nRyan Rickelton - 31 runs in 23 balls\n  \nMumbai Indians need 62 ru
ns in 84 balls\n\n#KKRvMI #KKRvsMI #MIvsKKR #MIvKKR #IPL2025 #TATAIPL2025 #TATAIPL #Mumbai #KorboLorboJeetbo",
      "polarity": 0,
      "sentiment": "neutral"
    },
    {
      "text": "MI vs KKR: કોણ છે અશ્વિની કુમાર જેણે 16 બોલમાં 4 વિકેટ લઈને આ રેકોર્ડ તોડ્યો?\n\nhttps://t.co/LhqCGAPJ0P\n\n#RohitSharma #HardikPandya #Rahane #MIvsKKR #AshwaniKumar #CSKvsRR #Manish
Pandey #Wankhede #MumbaiIndians #GujaratiJagran",
      "polarity": 0,
      "sentiment": "neutral"
    },
    {
      "text": "मुंबई इंडियन्सच्या अश्वनी कुमारने घडवला इतिहास\n\nhttps://t.co/E1kMZbyLgl &lt; येथे वाचा सविस्तर वृत्त\n\n#Marathinews #IPL #Mivskkr #ashwanikumar https://t.co/rG2Wp0iqbn",
      "polarity": 0,
      "sentiment": "neutral"
    },
    {
      "text": "@kanth_45 @Iconickohli No evidence supports Hardik Pandya saving the team did well because Rohit Sharma was absent. The X post seems like a humorous fabrication, as no IPL 202
```

Response headers

5. Remarks:-

Signature of the Student
_Bhairav Ganguly
(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)