LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSc

Submitted By

Name: Shreyaa Venkateswaran

Roll No: 2230120

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.	Experiment 1: Build a resume using HTML/CSS	07-01-2025	14-01-2025	
2.	Experiment 2: Machine Learning for Cat and Dog Classification	15-01-2025	20-01-2025	
3.	Experiment 3: Regression Analysis for Stock Prediction	21-01-2025	27-01-2025	
4.	Experiment 4: Conversational Chatbot with Any Files	04-02-2025	09-02-2025	
5.	Experiment 5: Web Scraper using LLMs	16-02-2025	17-03-2025	
6.	Experiment 6: Database Management Using Flask	11-03-2025	17-03-2025	
7.	Experiment 7: Natural Language Database Interaction with LLMs	18-03-2025	21-03-2025	
8.	Experiment 8: Sentiment Prediction API Using FastAPI of YouTube comments	26-03-2025	31-03-2025	
9.	Open Ended 1			
10.	Open Ended 2			

Experiment Number	8
Experiment Title	Sentiment Prediction API Using FastAPI of Youtube Comments
Date of Experiment	18-03-2025
Date of Submission	21-03-2025

1. Objective:

The objective of this lab experiment is to create a sentiment prediction API using FastAPI, which analyzes YouTube comments for positive, negative, or neutral sentiment. This lab integrates natural language processing (NLP) techniques with a lightweight and high-performing API.

2. Procedure:

Phase 1: Planning

Identify Supported URL Formats

List all YouTube URL patterns you need to handle (e.g., full URLs, short URLs, embedded URLs)

Include standalone 11-character video IDs

Define Validation Rules

Video IDs must be exactly 11 characters long

Only allow alphanumeric characters plus - and _ in IDs

Phase 2: Backend Implementation

Add Input Sanitization

Create a function to extract video IDs from URLs

Handle these cases:

URLs with ?v=

URLs with /embed/

Short youtu.be URLs

URLs with additional parameters

Modify API Endpoint

Accept raw URLs or video IDs as input

Sanitize input before processing

Return clear errors for invalid formats

Error Handling

Reject requests with malformed IDs immediately

Provide specific error messages (e.g., "Invalid timestamp in URL")

Phase 3: Frontend Implementation

Input Field Design

Add placeholder text: "Paste YouTube URL or Video ID"

Include helper text showing valid examples

Client-Side Validation

Check input length (minimum 11 chars for IDs)

Verify URL patterns match YouTube domains

Disable submit button for invalid inputs

User Feedback

Show real-time validation status (✓/✗ icons)

Display error messages under the input field

Phase 4: Testing

Test Valid Inputs

Full URLs (https://www.youtube.com/watch?v=...)

Short URLs (https://youtu.be/...)

Video IDs only (dQw4w9WgXcQ)

Test Edge Cases

URLs with timestamps (?t=123)

URLs with additional parameters (&feature=share)

Mobile URLs (m.youtube.com)

Verify Error Handling

Paste non-YouTube URLs (should reject)

Enter short/invalid IDs (should show error)

Leave field empty (should prevent submission)

Phase 5: Deployment

Backend Checks

Ensure API rejects invalid IDs before YouTube API calls

Monitor logs for malformed requests

Frontend Checks

Confirm validation works on all devices

Verify error messages are user-friendly

3. Code:

app.py:

```
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from fastapi.staticfiles import StaticFiles
from pydantic import BaseModel
from googleapiclient.discovery import build
from textblob import TextBlob
from dotenv import load_dotenv
import os
from typing import List

# Load environment variables
load_dotenv()
```

```python
# Initialize FastAPI
app = FastAPI(title="YouTube Sentiment Analysis API")

# CORS Configuration
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"],
)

# YouTube API Setup
YOUTUBE_API_KEY = os.getenv("YOUTUBE_API_KEY")
youtube = build('youtube', 'v3', developerKey=YOUTUBE_API_KEY)

# Models
class AnalyzeRequest(BaseModel):
    video_id: str
    max_comments: int = 50

class SentimentResult(BaseModel):
    text: str
    polarity: float
    sentiment: str

class AnalysisResponse(BaseModel):
    video_id: str
    total_comments: int
    positive: int
    neutral: int
    negative: int
    comments: List[SentimentResult]

# Helper Functions
def analyze_sentiment(text: str) -> dict:
    analysis = TextBlob(text)
    polarity = analysis.sentiment.polarity
    sentiment = "neutral"
    if polarity > 0.1: sentiment = "positive"
    elif polarity < -0.1: sentiment = "negative"
       return {"text": text, "polarity": polarity, "sentiment":
sentiment}
```

```python
def get_comments(video_id: str, max_results: int) -> List[str]:
    comments = []
    request = youtube.commentThreads().list(
        part="snippet",
        videoId=video_id,
        maxResults=min(100, max_results),
        textFormat="plainText"
    )
    while request and len(comments) < max_results:
        response = request.execute()

comments.extend(item['snippet']['topLevelComment']['snippet']['te
xtDisplay']
                       for item in response['items'])
            request = youtube.commentThreads().list_next(request,
response)
    return comments[:max_results]

# API Endpoint
@app.post("/analyze", response_model=AnalysisResponse)
async def analyze(request: AnalyzeRequest):
    try:
                     comments   =   get_comments(request.video_id,
request.max_comments)
        if not comments:
                  raise HTTPException(status_code=404, detail="No
comments found")

           analyzed = [analyze_sentiment(comment) for comment in
comments]
        counts = {
             "positive": sum(1 for r in analyzed if r["sentiment"]
== "positive"),
             "neutral": sum(1 for r in analyzed if r["sentiment"]
== "neutral"),
             "negative": sum(1 for r in analyzed if r["sentiment"]
== "negative")
        }

        return {
            "video_id": request.video_id,
            "total_comments": len(analyzed),
            **counts,
                "comments": analyzed[:10]   # Return first 10 as
samples
        }

    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))

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

index.html

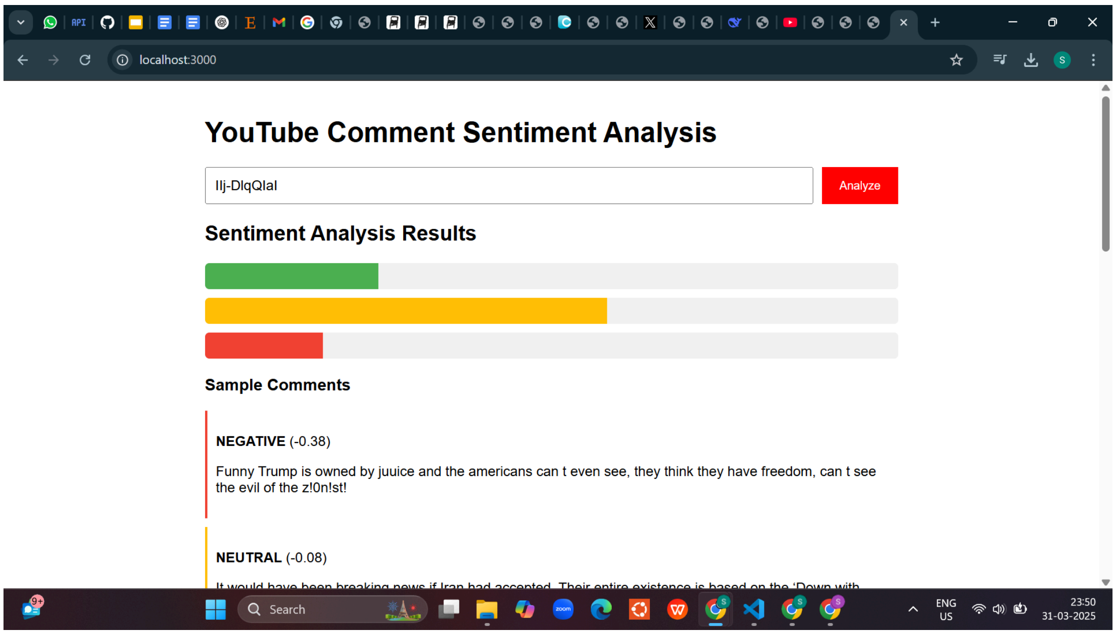
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0">
    <title>YouTube Sentiment Analyzer</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>YouTube Comment Sentiment Analysis</h1>
        <div class="input-section">
               <input  type="text"  id="videoId"  placeholder="Enter
YouTube Video ID">
            <button id="analyzeBtn">Analyze</button>
        </div>
        <div id="results" class="hidden">
            <div class="summary">
                <h2>Sentiment Analysis Results</h2>
                <div class="sentiment-meters">
                    <div class="meter positive">
                                          <div  class="meter-fill"
id="positiveMeter"></div>
                             <span id="positiveText">Positive: 0%
(0)</span>
                    </div>
                    <div class="meter neutral">
```

```
                                          <div  class="meter-fill"
id="neutralMeter"></div>
                               <span  id="neutralText">Neutral:  0%
(0)</span>
                    </div>
                    <div class="meter negative">
                                          <div  class="meter-fill"
id="negativeMeter"></div>
                             <span  id="negativeText">Negative:  0%
(0)</span>
                    </div>
                </div>
            </div>
            <div class="comments">
                <h3>Sample Comments</h3>
                <div id="commentsContainer"></div>
            </div>
        </div>
                     <div   id="loading"   class="hidden">Analyzing
comments...</div>
        <div id="error" class="hidden"></div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

script.js:

```
document.getElementById('analyzeBtn').addEventListener('click',
async () => {
                                    const         videoId         =
document.getElementById('videoId').value.trim();
    if (!videoId) return;

    const results = document.getElementById('results');
    const loading = document.getElementById('loading');
    const error = document.getElementById('error');

    results.classList.add('hidden');
    error.classList.add('hidden');
    loading.classList.remove('hidden');
    try {
                                        const    response    =    await
fetch('http://localhost:8000/analyze', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
                        body:  JSON.stringify({  video_id:  videoId,
max_comments: 100 })
        });

        if (!response.ok) throw new Error(await response.text());

        const data = await response.json();
        displayResults(data);
    } catch (err) {
        error.textContent = `Error: ${err.message}`;
        error.classList.remove('hidden');
    } finally {
        loading.classList.add('hidden');
    }
});

function displayResults(data) {
    // Update summary
    const total = data.total_comments;
     const positivePercent = Math.round((data.positive / total) *
100);
      const neutralPercent = Math.round((data.neutral / total) *
100);
     const negativePercent = Math.round((data.negative / total) *
100);

          document.getElementById('positiveMeter').style.width   =
`${positivePercent}%`;
           document.getElementById('neutralMeter').style.width   =
`${neutralPercent}%`;
          document.getElementById('negativeMeter').style.width   =
`${negativePercent}%`;

    document.getElementById('positiveText').textContent =
        `Positive: ${positivePercent}% (${data.positive})`;
    document.getElementById('neutralText').textContent =
        `Neutral: ${neutralPercent}% (${data.neutral})`;
    document.getElementById('negativeText').textContent =
        `Negative: ${negativePercent}% (${data.negative})`;

    // Display sample comments
                                    const        container        =
document.getElementById('commentsContainer');
    container.innerHTML = '';
    data.comments.forEach(comment => {
        const div = document.createElement('div');
        div.className = `comment ${comment.sentiment}-comment`;
        div.innerHTML = `

<p><strong>${comment.sentiment.toUpperCase()}</strong>
(${comment.polarity.toFixed(2)})</p>
            <p>${comment.text}</p>
        `;
        container.appendChild(div);
    });


document.getElementById('results').classList.remove('hidden');
}
```

4. Results/Output:



localhost:3000

YouTube Comment Sentiment Analysis

lIj-DlqQlaI

Analyze

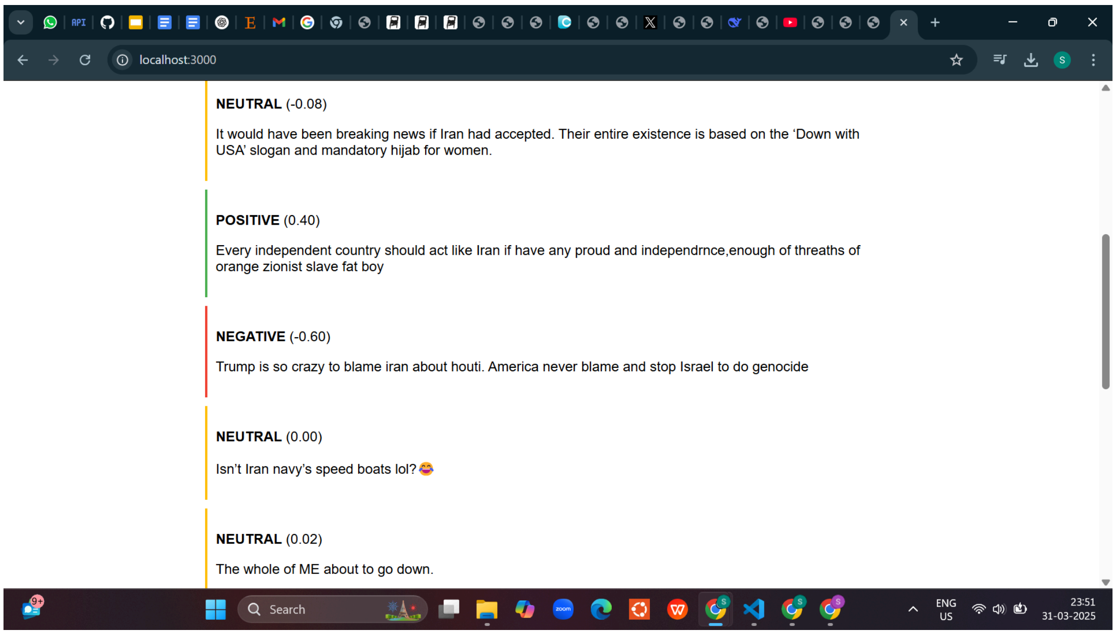
Sentiment Analysis Results

Sample Comments

NEGATIVE (-0.38)

Funny Trump is owned by juuice and the americans can t even see, they think they have freedom, can t see the evil of the z!0n!st!

NEUTRAL (-0.08)



NEUTRAL (-0.08)

It would have been breaking news if Iran had accepted. Their entire existence is based on the 'Down with USA' slogan and mandatory hijab for women.

POSITIVE (0.40)

Every independent country should act like Iran if have any proud and independrnce,enough of threaths of orange zionist slave fat boy

NEGATIVE (-0.60)

Trump is so crazy to blame iran about houti. America never blame and stop Israel to do genocide

NEUTRAL (0.00)

Isn't Iran navy's speed boats lol?😂

NEUTRAL (0.02)

The whole of ME about to go down.

5. Remarks:

Created a sentiment analysis prediction API using FastAPI which analyses the sentiment of Youtube comments. Classifies into Positive, Negative and Neutral.

Website link: Youtube_Sentiment_Analysis

GitHub link: GitHub

Shreyaa Venkateswaran

Signature of the Lab Coordinator