

**SENTIMENT ANALYSIS FOR YOUTUBE COMMENTS**

|  |  |
| --- | --- |
| **A MINI PROJECT**  **REPORT** |  |
| ***Submitted by*** |  |
| **DHARANI SRI.T.G** | **(211521106036)** |
| **JANANII.P** | **(211521106065)** |
|  |  |

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**PANIMALAR INSTITUTE OF TECHNOLOGY, CHENNAI-600 123**

**ANNA UNIVERSITY CHENNAI : 600 025**

**MAY 2024**

**ABSTRACT**

This mini project delves into sentiment analysis tailored specifically for YouTube comments, aiming to provide actionable insights into the sentiments expressed by audiences. To achieve this, we harness the power of advanced tools and technologies including Python's Natural Language Toolkit (NLTK) for sentiment analysis, the YouTube API for data retrieval, and Streamlit for creating interactive visualizations. The core of our sentiment analysis methodology lies in the SentimentIntensityAnalyzer from NLTK, which integrates the VADER lexicon. This analyzer computes compound sentiment scores on a scale from -1 to +1, encompassing overall sentiment along with positive, negative, and neutral aspects within comments.

The integration of Streamlit into our project further enhances user engagement by offering intuitive and user-friendly interfaces for visualizing sentiment analysis results. This report provides a comprehensive overview of the algorithmic methodology employed, the implementation of code within the Visual Studio Code (VS Code) environment, and the seamless integration of Streamlit to facilitate interactive UI elements.

It delves into the advantages offered by Streamlit such as ease of use and real-time visualization capabilities, while also acknowledging potential limitations in terms of GUI customization and computational resource requirements. Exploring the practical applications of our sentiment analysis tool, this report highlights its relevance in various domains including customer sentiment analysis for brands, social media monitoring and engagement strategies, educational research and analysis, and market research and competitor analysis. By showcasing the potential and versatility of combining NLTK's sentiment analysis capabilities with Streamlit's interactive visualization features, this project paves the way for future advancements in sentiment analysis tools across diverse industries and applications.

**Introduction:**

In the digital age, understanding and analyzing public sentiment on online platforms has become paramount. Social media platforms like YouTube serve as hubs for diverse opinions and feedback, making sentiment analysis a valuable tool for content creators, marketers, and researchers. This project, developed using VS Code and incorporating Streamlit for interactive visualization, focuses on sentiment analysis for YouTube comments. By leveraging Python libraries such as NLTK and the YouTube API, alongside Streamlit's capabilities for creating intuitive user interfaces, this project aims to provide actionable insights into audience sentiments expressed in YouTube comments.

**Algorithm Used:**

This project utilizes the SentimentIntensityAnalyzer from the Natural Language Toolkit (NLTK) library in Python, which incorporates the VADER (Valence Aware Dictionary and sEntiment Reasoner) lexicon. VADER assigns sentiment scores to words based on their polarity and intensity, allowing for nuanced sentiment analysis. The algorithm computes compound sentiment scores (-1 to +1) for YouTube comments, providing insights into overall sentiment as well as positive, negative, and neutral sentiments. This methodology, coupled with data retrieval from the YouTube API, forms the core of the sentiment analysis approach in this project.

**VS Code and Sentiment Analysis:**

VS Code was utilized as the integrated development environment (IDE) for this project due to its robust features for Python development, including code editing, debugging, and version control integration. The sentiment analysis aspect of the project leverages the Natural Language Toolkit (NLTK) library for analyzing the sentiment of YouTube comments.

**Streamlit for Interactive UI:**

Streamlit, a Python framework for creating interactive web applications, was integrated into the project to provide a user-friendly interface for visualizing sentiment analysis results. Streamlit simplifies the process of building and deploying data-driven applications, making it ideal for showcasing sentiment analysis outcomes to users.

**Code Implementation in VS Code with Streamlit:**

The project's code, developed in VS Code, demonstrates the implementation of sentiment analysis for YouTube comments and includes Streamlit components for the interactive user interface. Key functionalities include:

* Retrieving YouTube comments using the YouTube API.
* Analyzing sentiment using the SentimentIntensityAnalyzer from NLTK.
* Visualizing sentiment analysis results through interactive charts and summaries using Streamlit components.

**Code:**

import streamlit as st

from googleapiclient.discovery import build

from nltk.sentiment import SentimentIntensityAnalyzer

import nltk

nltk.download('vader\_lexicon')

#nltk.data.path.append('https://github.com/nltk/nltk\_data/tree/5db857e6f7df11eabb5e5665836db9ec8df07e28/packages/sentiment')

sia = SentimentIntensityAnalyzer()

def analyze\_sentiment(comment):

sentiment\_scores = sia.polarity\_scores(comment)

return sentiment\_scores

def display\_sentiment\_chart(commentslst):

sentiments = []

for comment in commentslst:

sentiment\_scores = analyze\_sentiment(comment)

sentiments.append(sentiment\_scores)

# Prepare data for the chart

data = {

'Positive': [sentiment['pos'] for sentiment in sentiments],

'Negative': [sentiment['neg'] for sentiment in sentiments],

'Neutral': [sentiment['neu'] for sentiment in sentiments]

}

# Display the bar chart

st.bar\_chart(data)

def extract\_video\_id(url):

video\_id = None

if 'youtube.com' in url:

video\_id = url.split('v=')[1]

ampersand\_position = video\_id.find('&')

if ampersand\_position != -1:

video\_id = video\_id[:ampersand\_position]

elif 'youtu.be' in url:

video\_id = url.split('/')[-1]

return video\_id

def display\_sentiment\_summary(commentslst):

num\_comments = len(commentslst)

positive\_count = 0

neutral\_count = 0

negative\_count = 0

for comment in commentslst:

sentiment\_scores = analyze\_sentiment(comment)

compound\_score = sentiment\_scores['compound']

if compound\_score >= 0.05:

positive\_count += 1

elif compound\_score <= -0.05:

negative\_count += 1

else:

neutral\_count += 1

positive\_percentage = (positive\_count / num\_comments) \* 100

neutral\_percentage = (neutral\_count / num\_comments) \* 100

negative\_percentage = (negative\_count / num\_comments) \* 100

st.title("Sentiment Summary")

st.write(f"Total Comments: {num\_comments}")

st.write(f"Positive Comments: {positive\_percentage:.2f}%")

st.write(f"Neutral Comments: {neutral\_percentage:.2f}%")

st.write(f"Negative Comments: {negative\_percentage:.2f}%")

def video\_comments(video\_id, api\_key):

commentslst = []

youtube = build('youtube', 'v3', developerKey=api\_key)

video\_response = youtube.commentThreads().list(

part='snippet',

videoId=video\_id,

textFormat='plainText',

maxResults=100

).execute()

while video\_response:

for item in video\_response['items']:

comment = item['snippet']['topLevelComment']['snippet']['textDisplay']

commentslst.append(comment)

if 'nextPageToken' in video\_response:

video\_response = youtube.commentThreads().list(

part='snippet',

videoId=video\_id,

textFormat='plainText',

maxResults=100,

pageToken=video\_response['nextPageToken']

).execute()

else:

break

return commentslst

# Streamlit app

def main():

st.title("YouTube Sentiment Analyzer")

api\_key = st.text\_input("Enter the YouTube API Key:")

url = st.text\_input("Enter the YouTube URL:")

if url and api\_key:

video\_id = extract\_video\_id(url)

if video\_id:

video\_url = f"https://www.youtube.com/watch?v={video\_id}"

st.markdown(f"### YouTube Video: {video\_url}")

st.video(video\_url)

comments = video\_comments(video\_id, api\_key)

st.title("Comments")

st.text\_area(label='', value='\n'.join(comments), height=300)

display\_sentiment\_chart(comments)

display\_sentiment\_summary(comments)

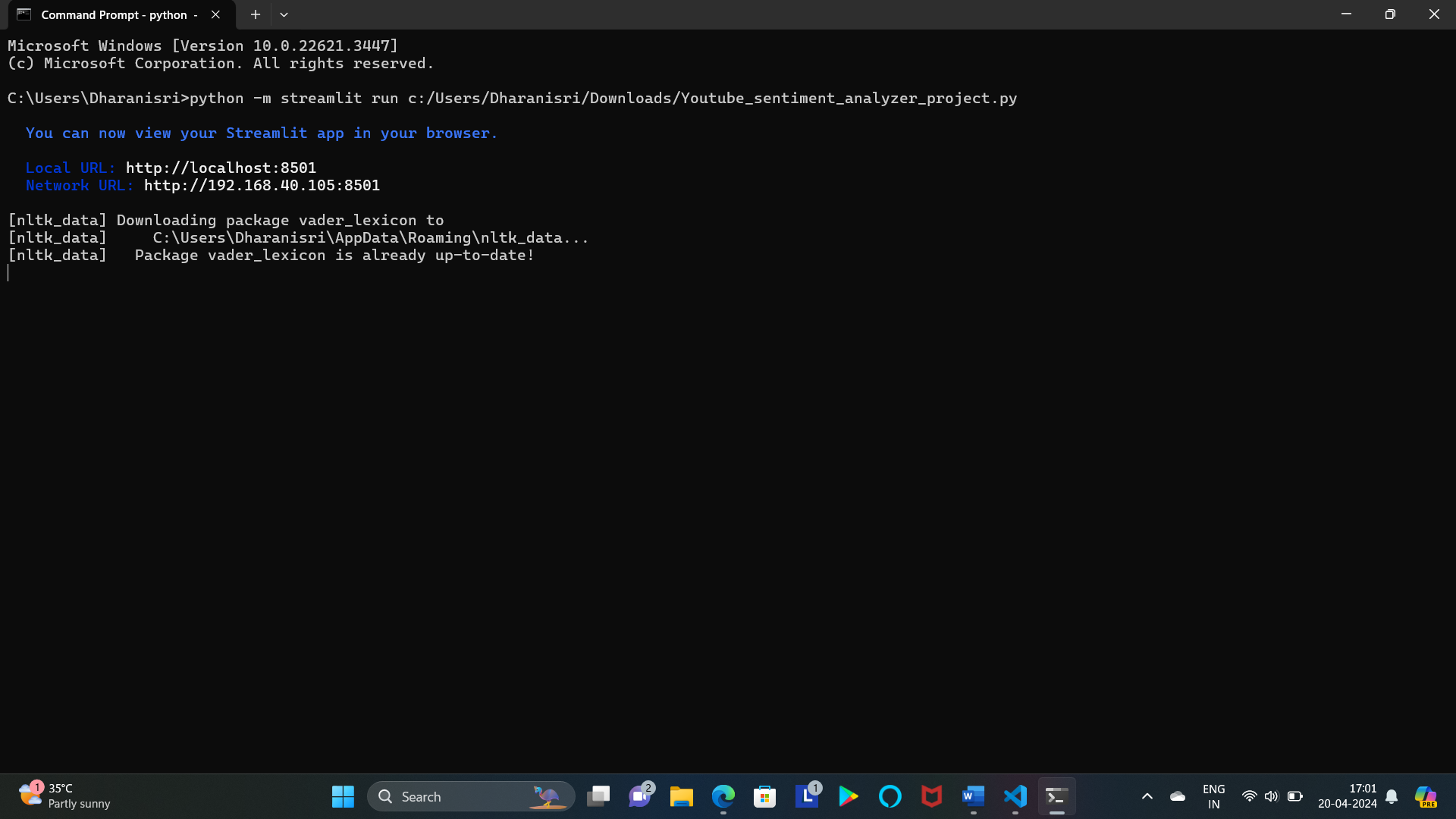
else:

st.markdown("Invalid YouTube URL")

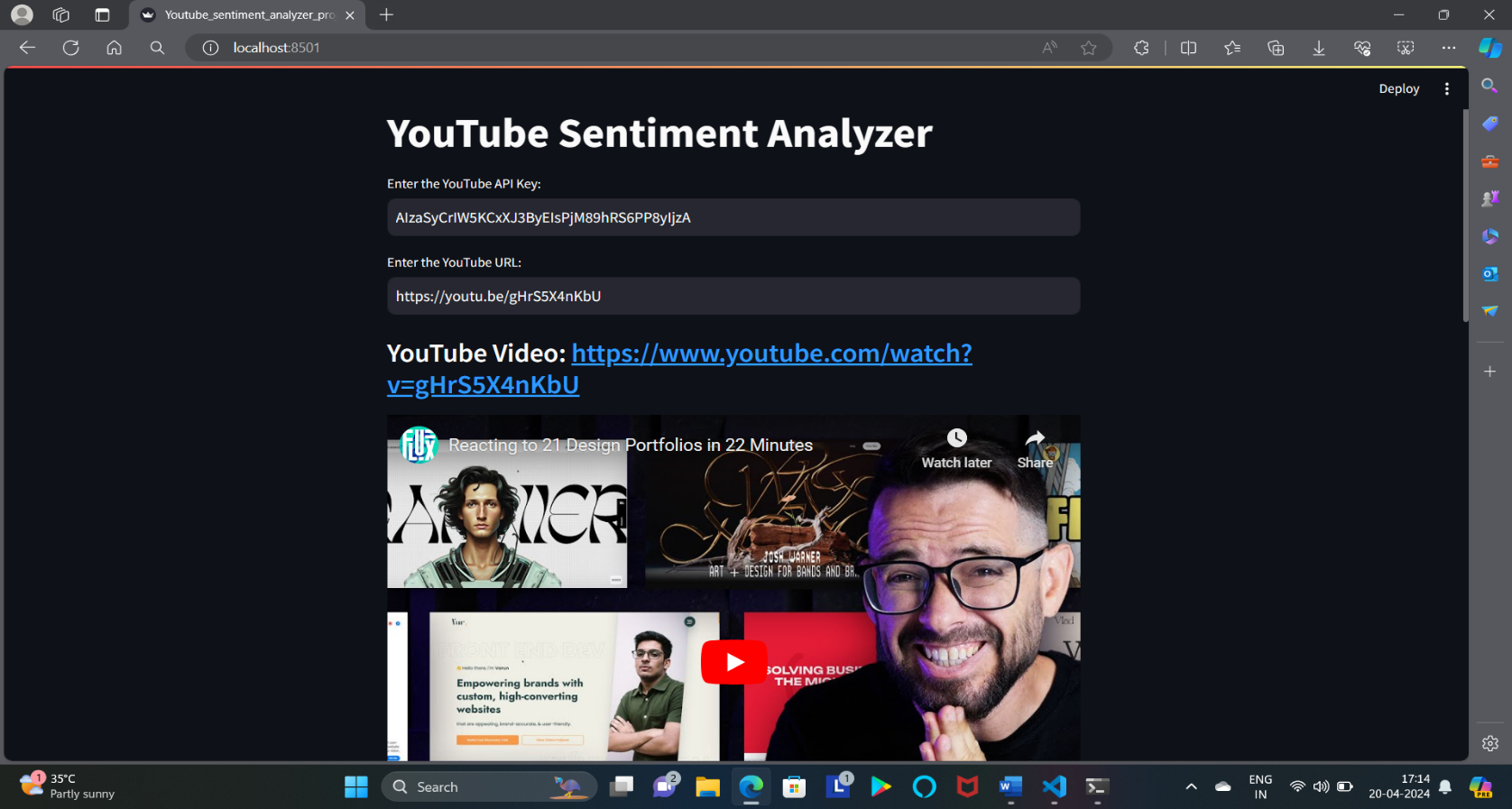
if \_name\_ == '\_main\_':

main()

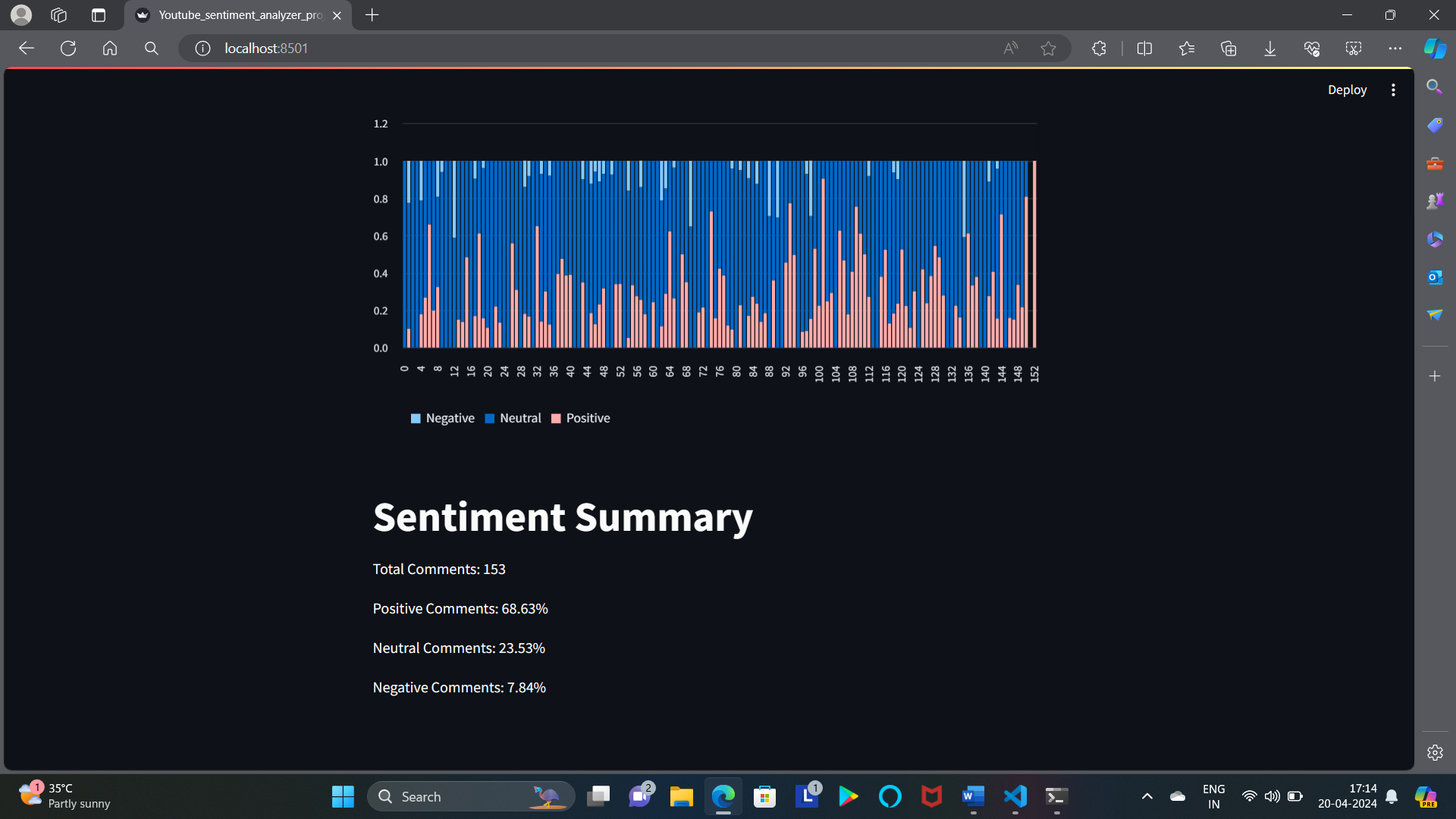
**Command prompt:**

****

**Output:**



****



The outputs generated from the code in VS Code with Streamlit integration include:

* Displaying the YouTube video within the Streamlit application.
* Shows the extracted comments from the video.
* Visualizing sentiment analysis results through interactive charts (positive, negative, neutral) using Streamlit.
* Presenting a summary of sentiment percentages based on the analyzed comments.

**Advantages:**

* **Ease of Use:** Streamlit's intuitive interface makes it easy for developers to create interactive applications without extensive knowledge of web development.
* **Efficient Development:** Streamlit's streamlined workflow accelerates the development process, allowing developers to focus on building the core functionality of the application.
* **Real-Time Visualization:** Streamlit enables real-time visualization of sentiment analysis results, providing immediate insights into comment sentiments.
* **Integration with External APIs:** Streamlit seamlessly integrates with external APIs, such as the YouTube API in this project, simplifying data retrieval and analysis.
* **Customization:** While Streamlit offers customization options, it may have limitations in complex GUI designs compared to dedicated frontend frameworks, which may be a concern for projects requiring intricate user interfaces.

**Disadvantages:**

* **Limited GUI Customization:** While Streamlit simplifies UI development, it may have limitations in creating highly customized and complex graphical user interfaces compared to dedicated frontend frameworks.
* **Compatibility Concerns:** Updates or changes in Streamlit's ecosystem may introduce compatibility issues or require adjustments in existing applications, impacting development timelines.
* **Resource Intensive:** Streamlit applications may require significant computational resources, potentially leading to performance issues on less powerful systems or in handling large datasets.
* **Learning Curve:** Developers unfamiliar with Streamlit may require time to learn its features and functionalities, impacting the initial speed of development and deployment.

**Applications**:

* **Customer Sentiment Analysis for Brands:** Streamlit-powered sentiment analysis tools can help brands monitor and analyze customer sentiments expressed in YouTube comments, enabling them to make data-driven decisions for marketing strategies and product improvements.
* **Social Media Monitoring and Engagement:** Content creators and social media managers can utilize Streamlit applications for tracking sentiment trends, engaging with audiences, and responding to feedback in real-time, enhancing audience engagement and brand perception.
* **Educational Research and Analysis:** Researchers and educators can leverage Streamlit for sentiment analysis projects focused on understanding student feedback, sentiment dynamics in educational videos, and enhancing learning experiences based on sentiment insights.
* **Market Research and Competitor Analysis:** Streamlit-based sentiment analysis tools can be used in market research to analyze competitor sentiments, industry trends, and customer preferences expressed in YouTube comments, aiding in strategic decision-making and market positioning.

**Conclusion:**

The integration of Streamlit with VS Code has created an efficient sentiment analysis tool for YouTube comments. This project offers a user-friendly platform for understanding sentiment trends through natural language processing and interactive visualization.

Streamlit's advantages, such as streamlined development and real-time visualization, enhance the tool's usability.

Overall, this project has demonstrated the potential of using Streamlit with VS Code for sentiment analysis applications, paving the way for future developments in sentiment analysis tools and their applications across various domains.

**Reference:**

* Athar, A. (2014). Sentiment analysis of scientific citations (No. UCAMCL-TR-856). University of Cambridge, Computer Laboratory.
* Athar, A., Teufel, S. (2012, July). Detection of implicit citations for sentiment detection. In Proceedings of the Workshop on Detecting Structure in Scholarly Discourse (pp. 18-26).
* Poria S, Cambria E, Gelbukh A, Bisio F, Hussain A (2015) Sentiment data flow analysis by means of dynamic linguistic patterns.
* Socher R (2016) Deep learning for sentiment analysis—invited talk. In: Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment, and social media analysis.