YouTube Comments Sentiment Analysis

Abstract:

This project involves building a web application that performs sentiment analysis on comments from a YouTube video. Users can input a YouTube video ID, and the application retrieves and analyses the comments to determine the sentiments expressed. The results are displayed as rounded percentages and shown as a colourful pie chart. The project is built using Python, Flask, the YouTube Data API, TextBlob for sentiment analysis, and Matplotlib for chart generation.

Problem Description:

Analysing sentiments expressed in YouTube comments can be valuable for content creators and marketers to understand the audience's reactions to their videos. This project addresses the need for a user-friendly tool that can quickly provide sentiment analysis results for a given YouTube video.

Software Requirements:

1.Python

2.Flask

3.Google API Client Library

4.TextBlob

5.Matplotlib

6.A web browser

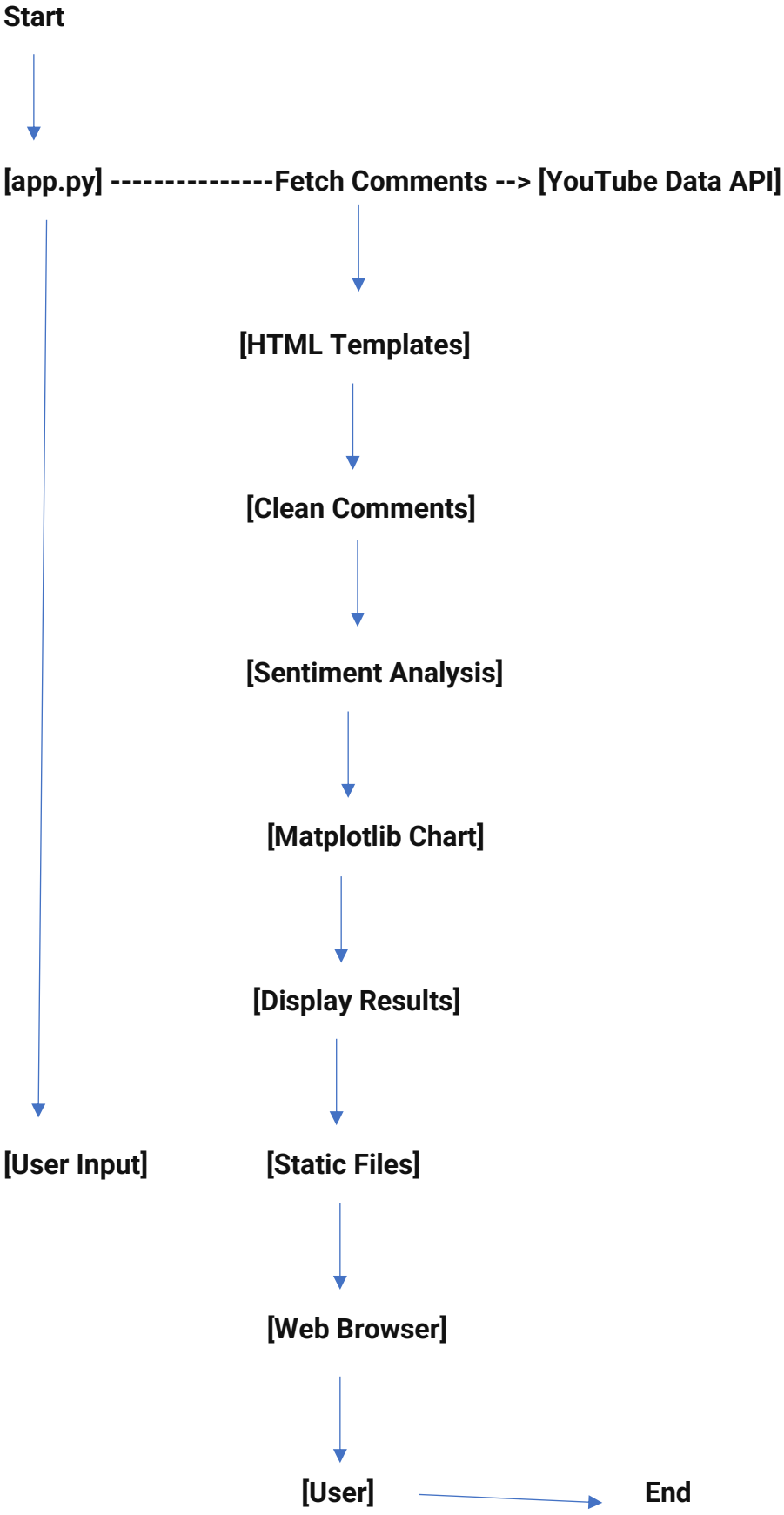
Hardware Requirements:

There are no specific hardware requirements for this project, as it can run on any computer or server that supports Python and Flask.

Folder Creation:

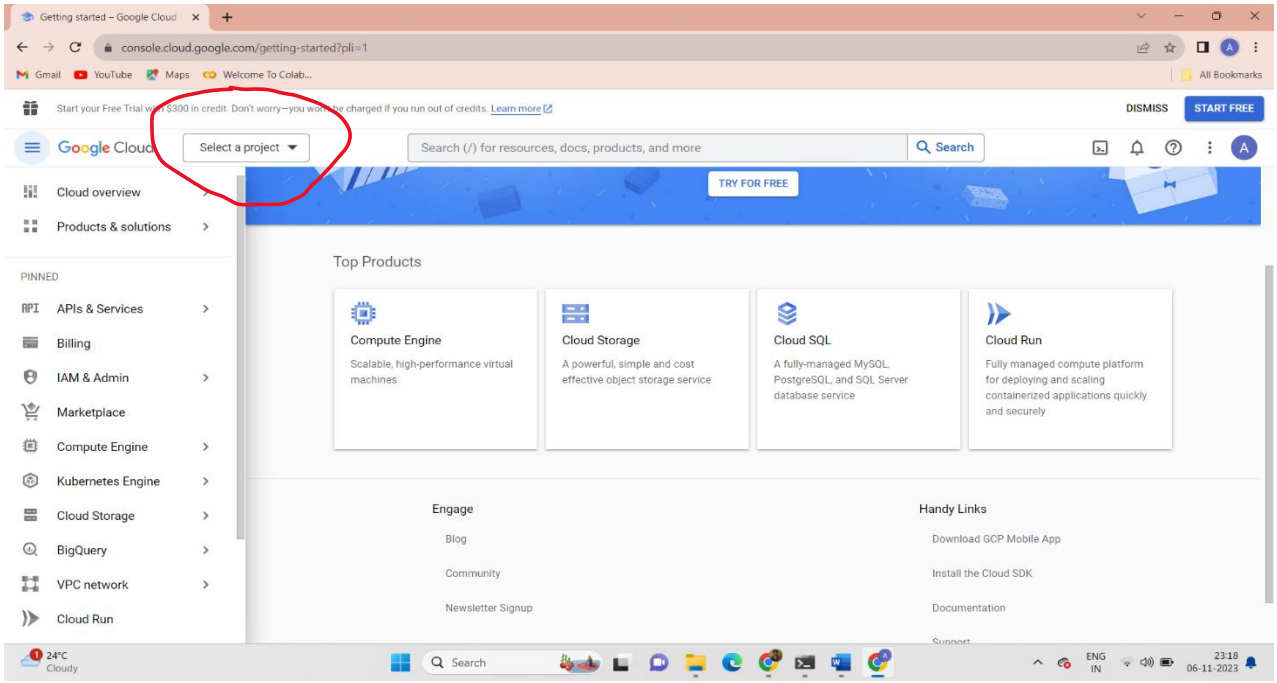
```
your_project_directory/
├── app.py
├── templates/
│   ├── result.html
│   └── index.html
└── static/
    ├── chart.png
    └── other_static_files.css/js
```

Flow Chart

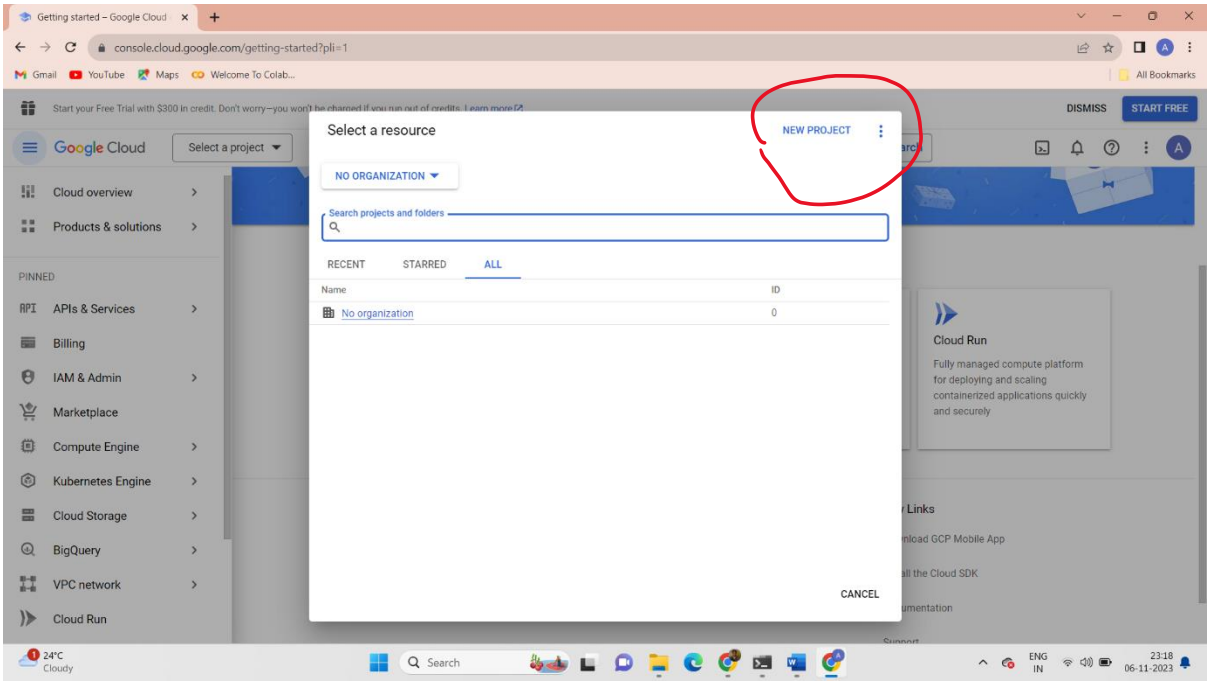

Start
[app.py] ----------Fetch Comments --> [YouTube Data API]
[HTML Templates]
[Clean Comments]
[Sentiment Analysis]
[Matplotlib Chart]
[Display Results]
[User Input]
[Static Files]
[Web Browser]
[User]
End

Step-By-Step process to get YouTube Data API v3 Credentials

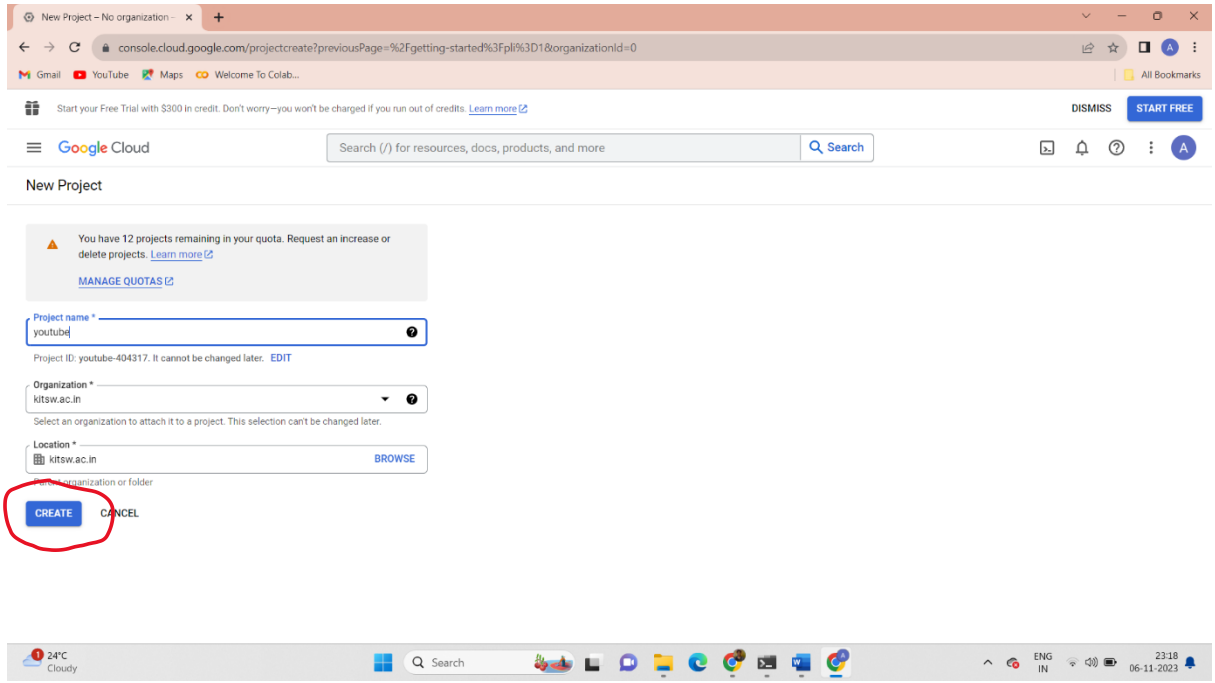
Open https://console.cloud.google.com



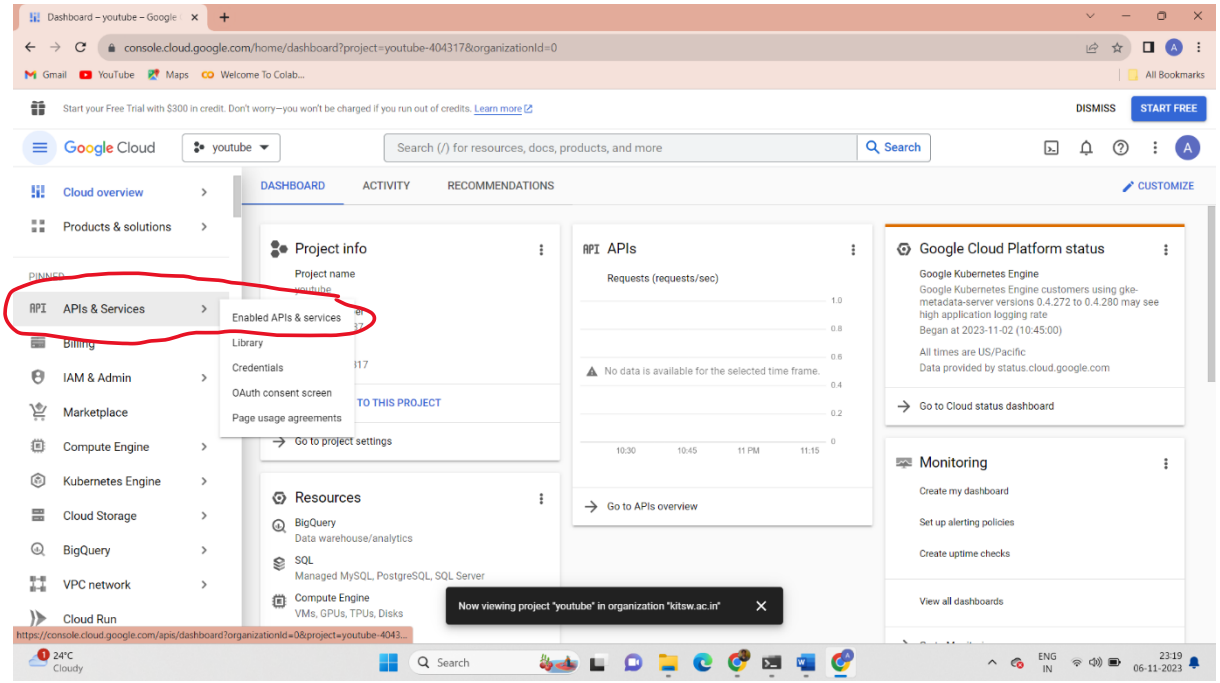
Create a New project



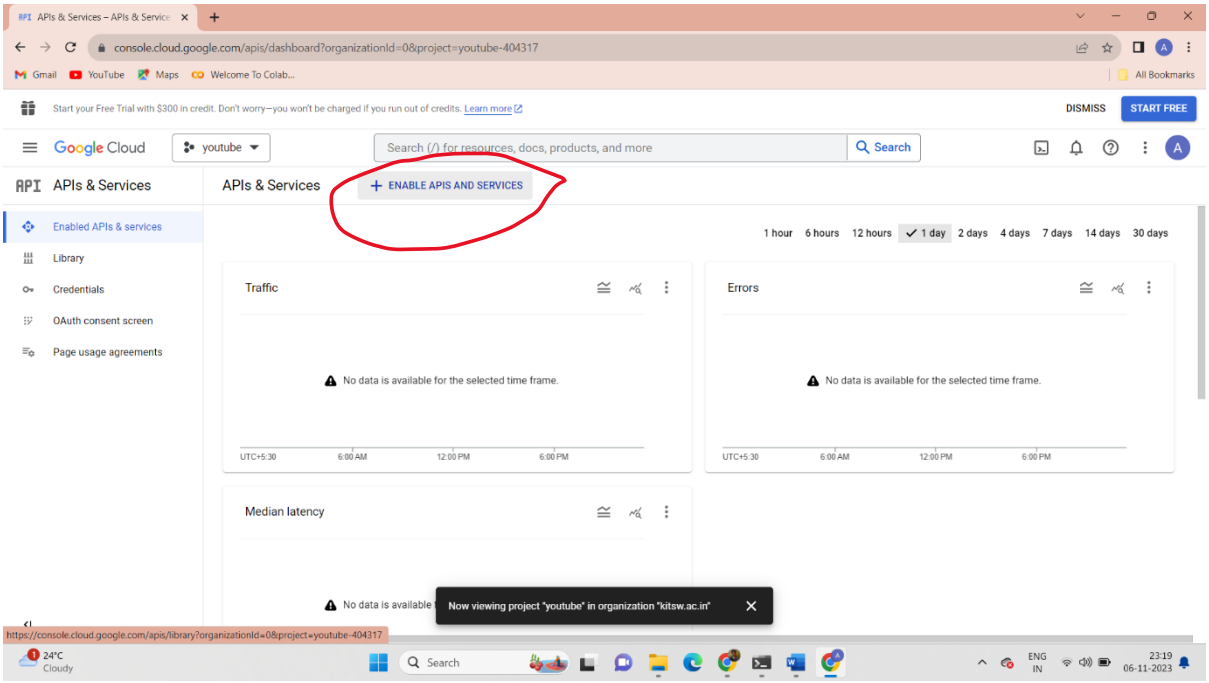
Create a project name it



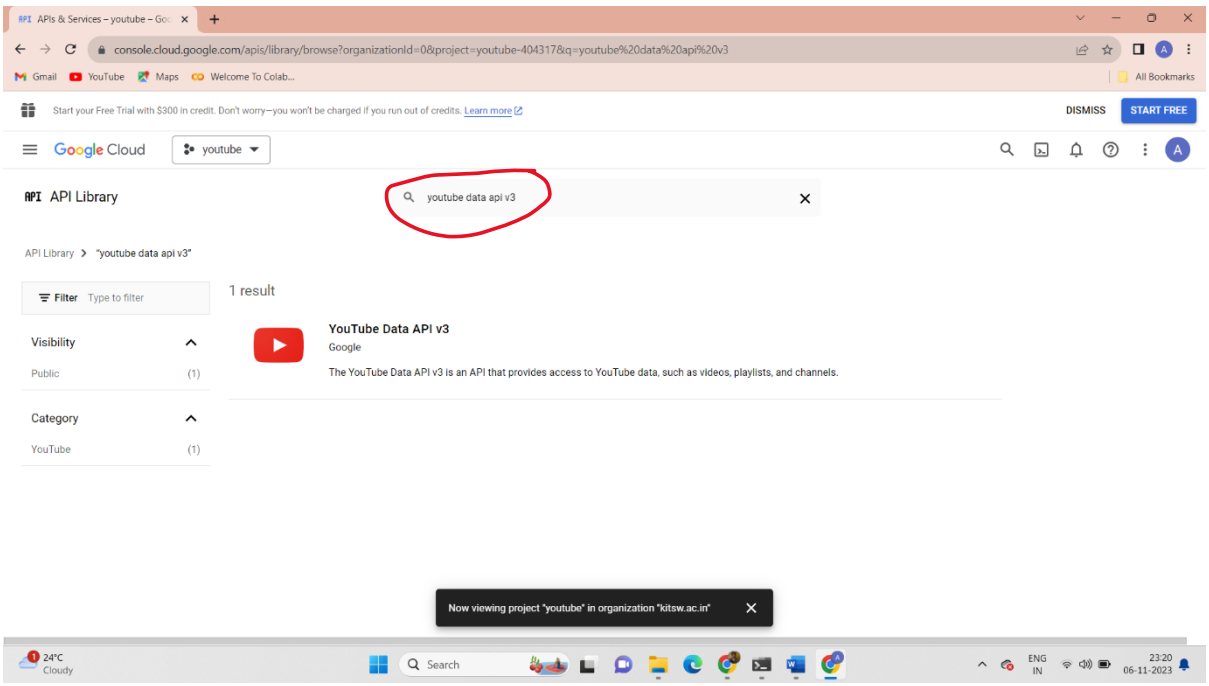
Go to APIs and services and then select Enable APIs and services



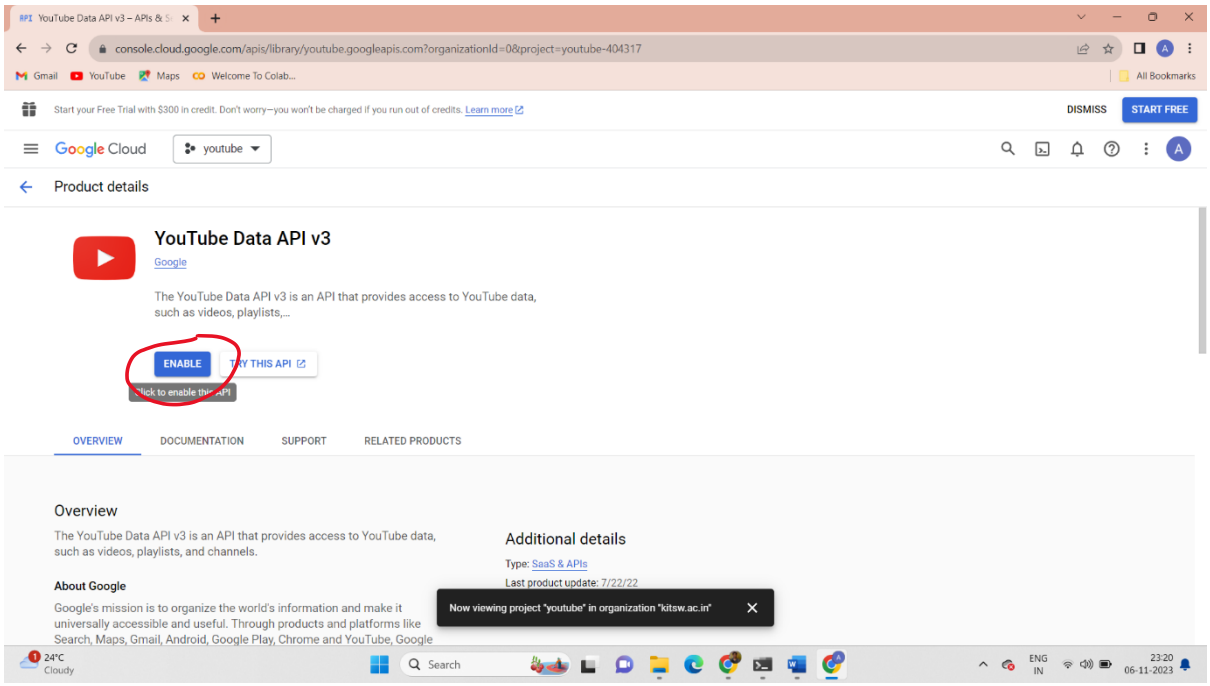
Click on Enable APIs and services



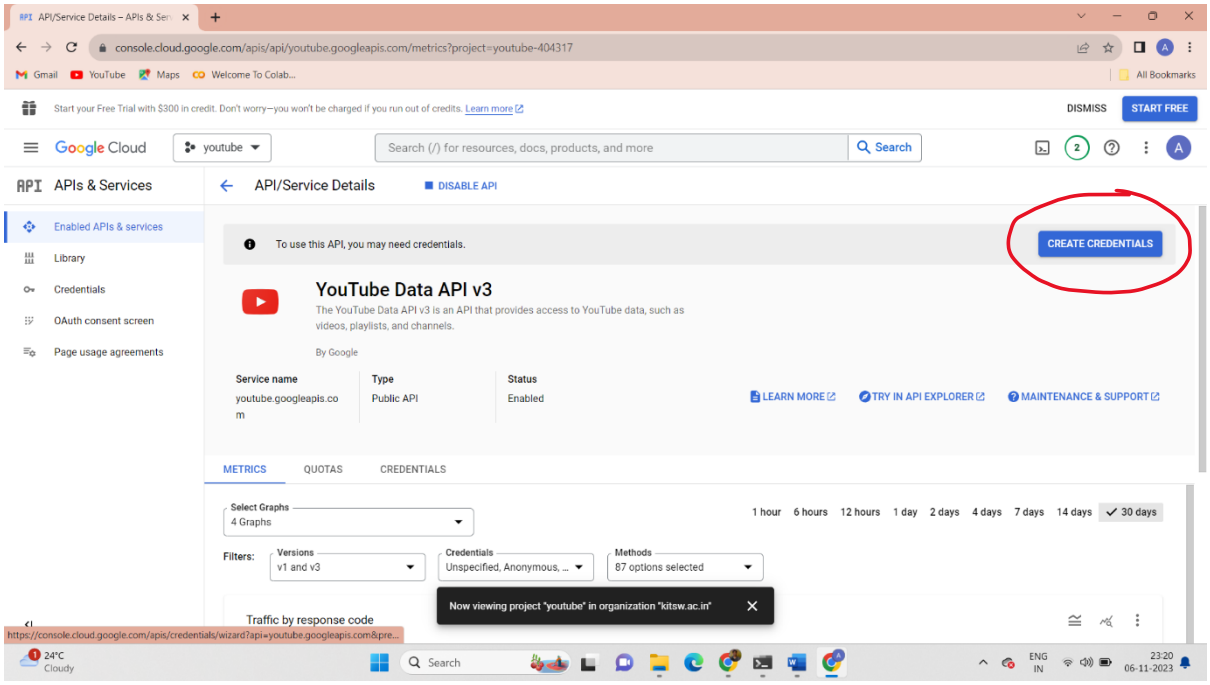
One the top Search for YouTube Data API V3



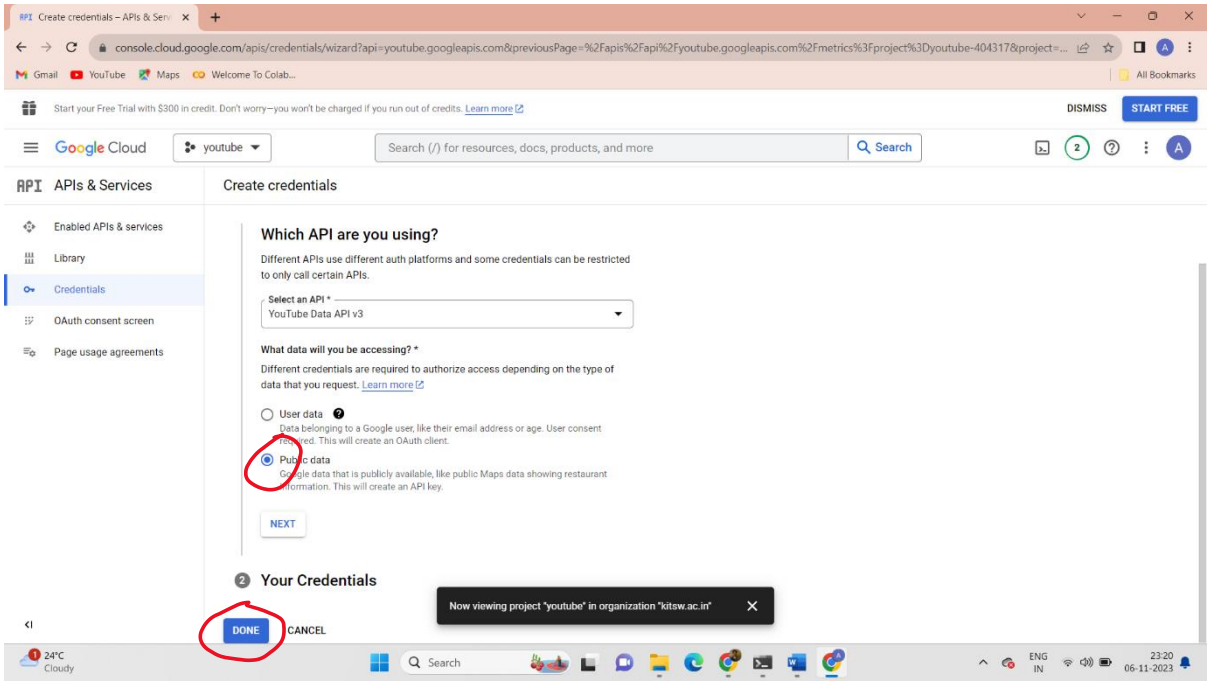
Click on Enable



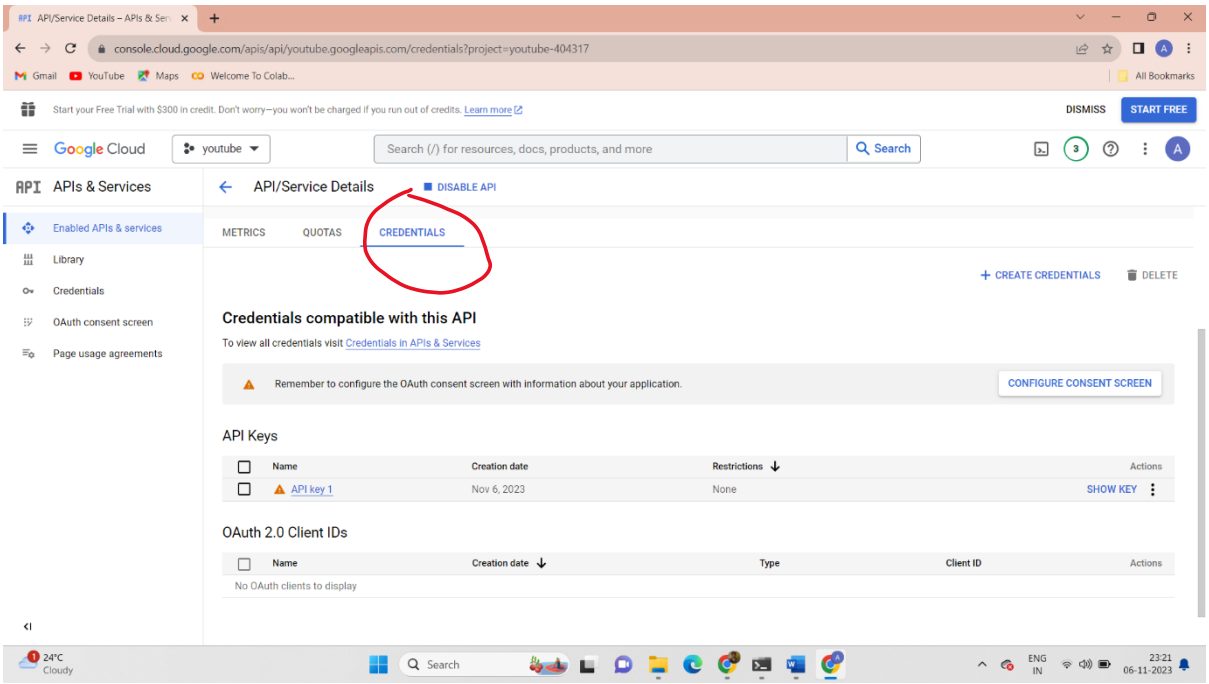
Then Click on Create Credentials



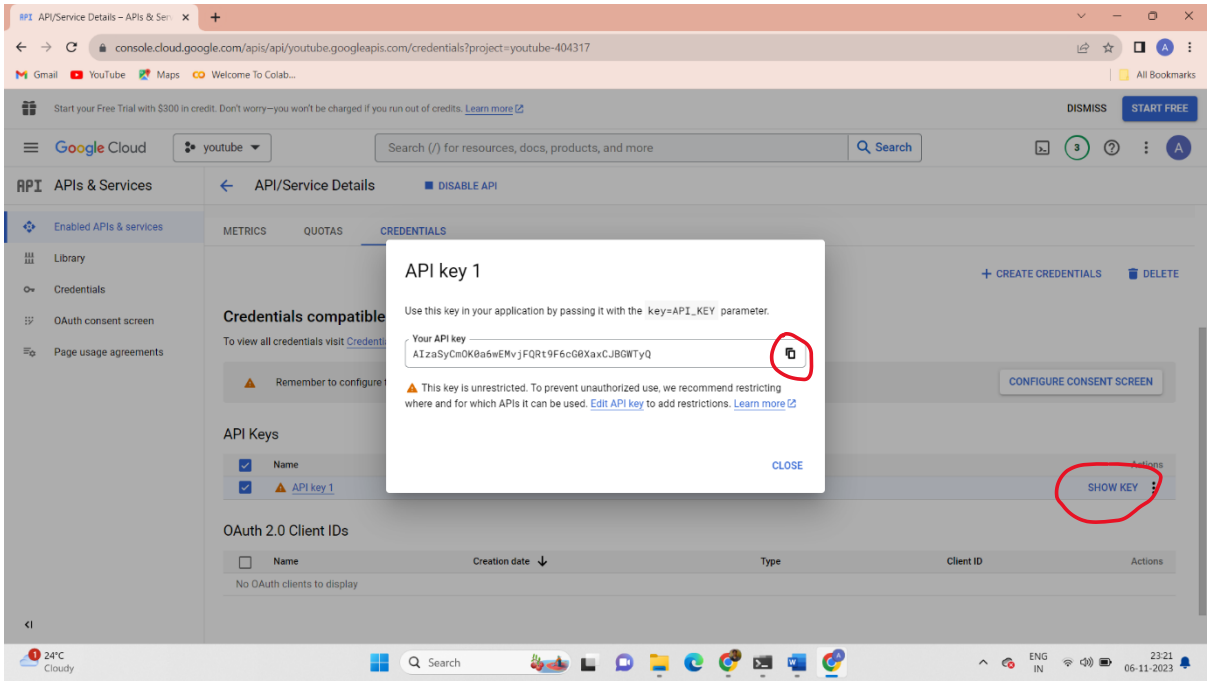
Then select Public and click on done



Go to Credentials



Click on show Key and code and you can use



Codes with Explanation:

Web Application (app.py):

```
1  from flask import Flask, render_template, request
2  import os
3  from googleapiclient.discovery import build
4  import re
5  from textblob import TextBlob
6  import matplotlib.pyplot as plt
```

1. Flask:

Flask is a micro web framework for Python. It simplifies the process of building web applications.

from flask import Flask, render_template, request:

Flask: The Flask class is imported to create a Flask application.

render_template: This function is used to render HTML templates, allowing you to generate dynamic web pages.

request: It provides access to the data sent with the HTTP request, including form data.

2. Google API Client Library (googleapiclient. discovery):

The Google API Client Library is used to interact with various Google APIs, including the YouTube Data API.

from googleapiclient. discovery import build:

build: This function is used to create a service endpoint for the YouTube Data API. It initializes a client for making requests to the API.

3. Regular Expressions (re):

The `re` module in Python provides regular expression matching operations.

import re:

re.sub(): The sub () function is used for substitution. In this code, it is used to clean comments by removing unwanted characters from the text using regular expressions.

4. TextBlob:

TextBlob is a library for processing textual data. It provides simple API for diving into common natural language processing (NLP) tasks.

from textblob import TextBlob:

TextBlob(comment): It is used to create a TextBlob object for sentiment analysis. The TextBlob library is used to analyze the sentiment of comments by measuring polarity.

5. Matplotlib:

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in Python. In this code, it is used to create data visualizations, such as the pie chart.

import matplotlib.pyplot as plt:

pyplot: The `pyplot` module is a subpackage of Matplotlib that provides a simple and convenient way to create plots and charts.

plt.figure(): This function is used to initialize a figure for the chart.

plt.pie(): It creates a pie chart to visualize sentiment percentages.

plt.axis('equal'): This function sets the aspect ratio to ensure that the pie chart is circular.

plt.title(): It adds a title to the chart.

plt.savefig(): This function is used to save the chart as an image file.

plt.close(): It closes the figure, finalizing the chart.

```
7
8   app = Flask(__name__)
9   api_key = 'replace : should be taken form google cloud'
10  youtube = build('youtube', 'v3', developerKey=api_key)
11
```

1. app = Flask(__name):

This line initializes a Flask web application. The Flask(__name__) constructor creates an instance of the Flask class, representing your web application.

The __name__ argument specifies the name of the current module. It's used to determine the root path for the application.

2. api_key = 'replace :

should be taken from google cloud': This line defines a variable called api_key and assigns it a string value. However, the value is a placeholder indicating that you should replace it with an actual API key.

API keys are typically obtained from services like Google Cloud for accessing various APIs, including the YouTube Data API. To use the YouTube Data API, you need to replace this placeholder with your valid API key.

3.youtube=build('youtube','v3',developerKey=

api_key):

This line creates a client for the YouTube Data API using the Google API Client Library. The build() function initializes the client.

YouTube is a variable that represents the YouTube Data API client, and you can use it to make requests to the API.

The arguments provided to build() include:

YouTube: The name of the API you want to access, which is the YouTube Data API in this case.

v3: The version of the API you want to use, which is version 3 in this case.

developerKey=api_key: This specifies the API key to be used for authentication and authorization when making requests to the YouTube Data API. The api_key variable is passed as the developer key.

```
11
12  def clean_comment(comment):
13      cleaned_comment = re.sub(r'[^a-zA-Z0-9\s,.!?]', '', comment)
14      return cleaned_comment
15
```

def clean_comment(comment): This line defines a function named clean_comment that takes one argument, comment, which is the text to be cleaned.

cleaned_comment = re.sub(r'[^a-zA-Z0-9\s,.!?]', '', comment): re.sub() is a function from the re (regular expressions) module used for string substitution.

The regular expression r'[^a-zA-Z0-9\s,.!?]' is used to match any characters that are not alphabetic (a-zA-Z), numeric (0-9), whitespace (\s), comma (,), period (.), or exclamation/question marks (! and ?).

The re.sub() function replaces all such non-matching characters in the comment string with an empty string, effectively removing them.

return cleaned_comment: The cleaned version of the comment is returned as the function's result.

```python

def get_video_comments(video_id):
    comments = []

    nextPageToken = None

    while True:
        results = youtube.commentThreads().list(
            part='snippet',
            videoId=video_id,
            textFormat='plainText',
            pageToken=nextPageToken
        ).execute()

        for item in results['items']:
            comment = item['snippet']['topLevelComment']['snippet']['textDisplay']
            cleaned_comment = clean_comment(comment)
            comments.append(cleaned_comment)

        nextPageToken = results.get('nextPageToken')

        if not nextPageToken:
            break
    return comments

```

comments = []: This initializes an empty list called comments to store the comments extracted from the YouTube video.

nextPageToken = None: It initializes the nextPageToken variable as None. This token is used to retrieve the next page of comments when there are more comments than can fit on a single page. It's None initially because we haven't fetched any comments yet.

The while True: loop is used to fetch comments from multiple pages. The loop will continue until there are no more pages of comments to retrieve.

results = youtube.commentThreads().list(...): This line makes a request to the YouTube Data API to fetch a page of comments for the specified video_id. It includes parameters like the part to retrieve ('snippet'), the video ID, the text format, and the page token for pagination. The .execute() function sends the request and gets the results.

for item in results['items']:: It iterates through the comments in the current page of results.

Comment= item['snippet']['topLevelComment']

['snippet']['textDisplay']: This line extracts the text of each comment from the results.

cleaned_comment = clean_comment(comment): It uses the clean_comment function to clean the comment text by removing unwanted characters.

comments.append(cleaned_comment): The cleaned comment is added to the comments list.

nextPageToken = results.get('nextPageToken'): This line gets the token for the next page of comments. If there are more pages, nextPageToken is set to a new token; otherwise, it remains None, and the loop will break.

The loop continues fetching comments from subsequent pages until there are no more pages, at which point the loop exits.

Finally, the function returns the list of cleaned comments collected from all the pages of comments in the video.

```
40  def analyze_sentiment(comments):
41      positive_count = 0
42      neutral_count = 0
43      negative_count = 0
44
45      for comment in comments:
46          analysis = TextBlob(comment)
47          polarity = analysis.sentiment.polarity
48
49          if polarity > 0:
50              positive_count += 1
51          elif polarity < 0:
52              negative_count += 1
53          else:
54              neutral_count += 1
55
56      total_count = len(comments)
57
58      positive_percentage = (positive_count / total_count) * 100
59      neutral_percentage = (neutral_count / total_count) * 100
60      negative_percentage = (negative_count / total_count) * 100
61
62      return positive_percentage, neutral_percentage, negative_percentage
63
64  @app.route('/', methods=['GET', 'POST'])
```

The analyze_sentiment function takes a list of comments as its input, which are the cleaned and pre-processed comments obtained from the YouTube video. Three counters, positive_count, neutral count, and negative count, are initialized to keep track of the number of comments in each sentiment category.

A for loop iterates through each comment in the comments list.

For each comment, TextBlob(comment) is used to create a TextBlob object, which is a part of the TextBlob library. The analysis.sentiment. Polarity attribute is accessed to calculate the sentiment polarity of the comment. The polarity value can be positive (indicating positive sentiment), negative

(indicating negative sentiment), or zero (indicating neutral sentiment).

Based on the polarity value, the comment is categorized as positive, negative, or neutral, and the respective counters are updated accordingly.

The total count of comments, total_count, is calculated using len(comments). The percentages of positive, neutral, and negative comments are computed by dividing the corresponding counts by the total count and then multiplying by 100 to get the percentage.

Finally, the function returns the percentages of positive, neutral, and negative comments as three separate values.

This function is called within our Flask application to calculate sentiment percentages based on the comments obtained from the YouTube video.

The @app.route('/', methods=['GET', 'POST']) is a decorator for the root route of your Flask application, indicating that this route can handle both GET and POST requests. It's typically where users can input the YouTube video ID and initiate the sentiment analysis process.

```python
def index():
    if request.method == 'POST':
        video_id = request.form['video_id']
        comments = get_video_comments(video_id)
        positive_percentage, neutral_percentage, negative_percentage = analyze_sentiment(comments)

        positive_percentage = round(positive_percentage, 2)
        neutral_percentage = round(neutral_percentage, 2)
        negative_percentage = round(negative_percentage, 2)

        labels = ['Positive', 'Neutral', 'Negative']
        sizes = [positive_percentage, neutral_percentage, negative_percentage]
        colors = ['green', 'lightgray', 'red']

        plt.figure(figsize=(8, 6))
        plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140)
        plt.axis('equal')
        plt.title('Sentiment Analysis Results')

        chart_path = 'static/chart.png'
        plt.savefig(chart_path)
        plt.close()

        return render_template('result.html', video_id=video_id, positive=positive_percentage, neutral=neutral_percentage,
        negative=negative_percentage, chart_path=chart_path)

    return render_template('index.html')
```

if request.method == 'POST':: This condition checks if the incoming request is a POST request, which typically indicates that the user has submitted a form.

video_id = request.form['video_id']: It extracts the YouTube video ID from the form submitted by the user.

comments = get_video_comments(video_id): The get_video_comments function is called to retrieve and clean the comments for the specified video ID.

Sentiment analysis percentages (positive, neutral, and negative) are calculated by calling the analyze_sentiment function on the retrieved comments.

The percentages are rounded to two decimal places using the round() function.

labels, sizes, and colors are defined for creating the pie chart. Labels represent sentiment categories, sizes represent the corresponding percentages, and colors represent the colors for each category.

plt.figure(figsize=(8, 6)): This line initializes a figure for the chart, specifying its size.

plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140): It creates a pie chart with specified sizes, labels, colors, and format for displaying percentages.

plt.axis('equal'): This line ensures that the pie chart is circular.

plt.title('Sentiment Analysis Results'): It sets the title for the chart.

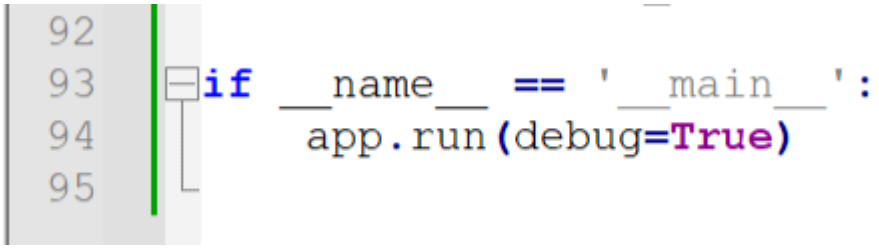
chart_path = 'static/chart.png': This line defines the path where the chart image will be saved.

plt.savefig(chart_path): It saves the pie chart as an image at the specified path.

plt.close(): This closes the figure, finalizing the chart generation.

Finally, the function renders the result.html template, passing the video ID, rounded sentiment percentages, and the path to the generated chart as variables to be displayed on the result page.

If the request method is not POST (i.e., a GET request), the function renders the index.html template, which typically contains the input form for the YouTube video ID.



if __name__ == '__main__': checks if the script is being run as the main application (i.e., directly executed). When you run this script using Python, __name__ is set to '__main__'.

app.run(debug=True) starts the Flask development server. The debug=True parameter enables debug mode, which is useful during development. In debug mode, the server will automatically reload when you make changes to your code, and it will provide detailed error messages if there are issues.

Index.html (For home page)”

```html
<!DOCTYPE html>
<html>
<head>
    <title>YouTube Comment Sentiment Analysis</title>
</head>
<body>
    <h1>YouTube Comment Sentiment Analysis</h1>
    <form method="POST">
        <label for="video_id">Enter YouTube Video ID:</label>
        <input type="text" name="video_id" required>
        <button type="submit">Analyze</button>
    </form>
</body>
</html>

```

The HTML code creates a web page for a YouTube Comment Sentiment Analysis project. It features a title, a form for users to input a YouTube Video ID, and a " Analyse " button. When users enter a Video ID and click "Analyse," the form data is sent to the server for analysis. This simple interface serves as the entry point for the sentiment analysis process.

Result.html(For result purpose):

```
<!DOCTYPE html>
<html>
<head>
    <title>Analysis Result</title>
</head>
<body>
    <h1>Analysis Result for Video ID: {{ video_id }}</h1>
    <b><p>Positive Comments: {{ positive }}%</p>
    <p>Neutral Comments: {{ neutral }}%</p>
    <p>Negative Comments: {{ negative }}%</p>
    <center><img src="{{ chart_path }}" alt="Sentiment Analysis Pie Chart" width="500" height="400"></center></b>
</body>
</html>
```

This HTML code is for the "Analysis Result" page of a web application. It displays the results of sentiment analysis for a specific YouTube Video ID. Here's a short explanation:

The HTML document is declared with a title "Analysis Result."

The page displays the Video ID in a top-level heading (h1).

Sentiment analysis results are shown with percentages for positive, neutral, and negative comments using <p> tags.

A pie chart image, representing the sentiment analysis distribution, is displayed with a specified width and height using an <img> tag. The image source (src) is dynamically generated using the chart_path variable from the Flask application, which is passed to the template to display the chart.

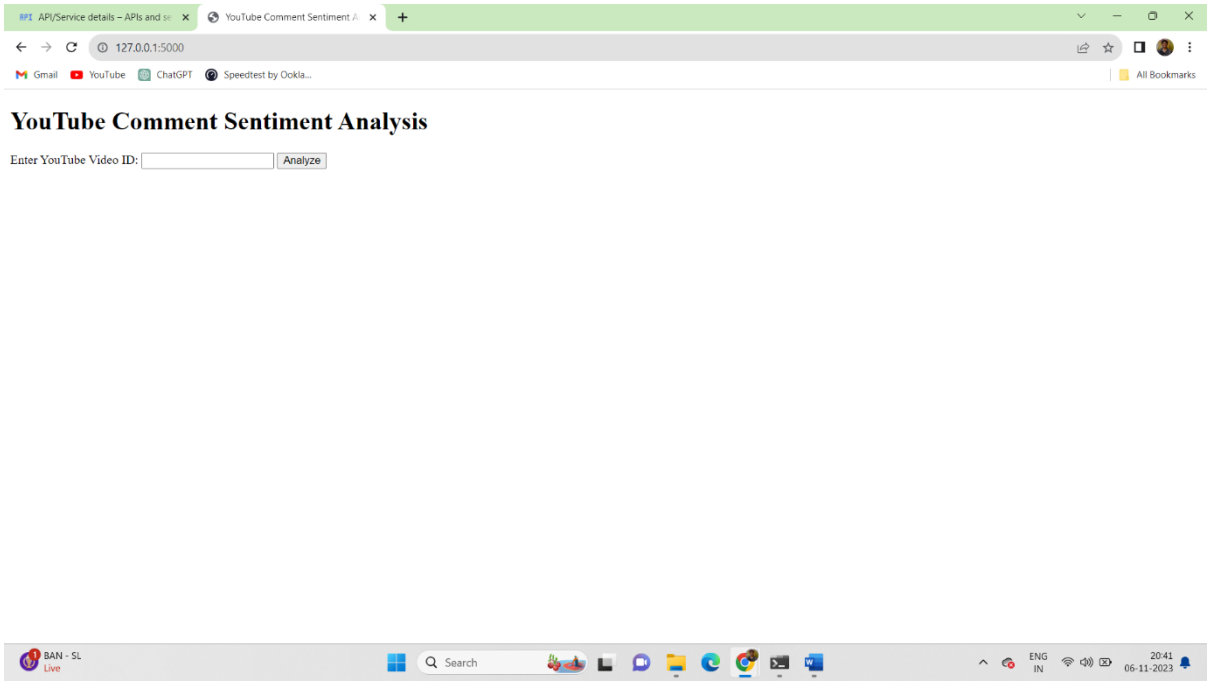
Output:

Compile: python app.py

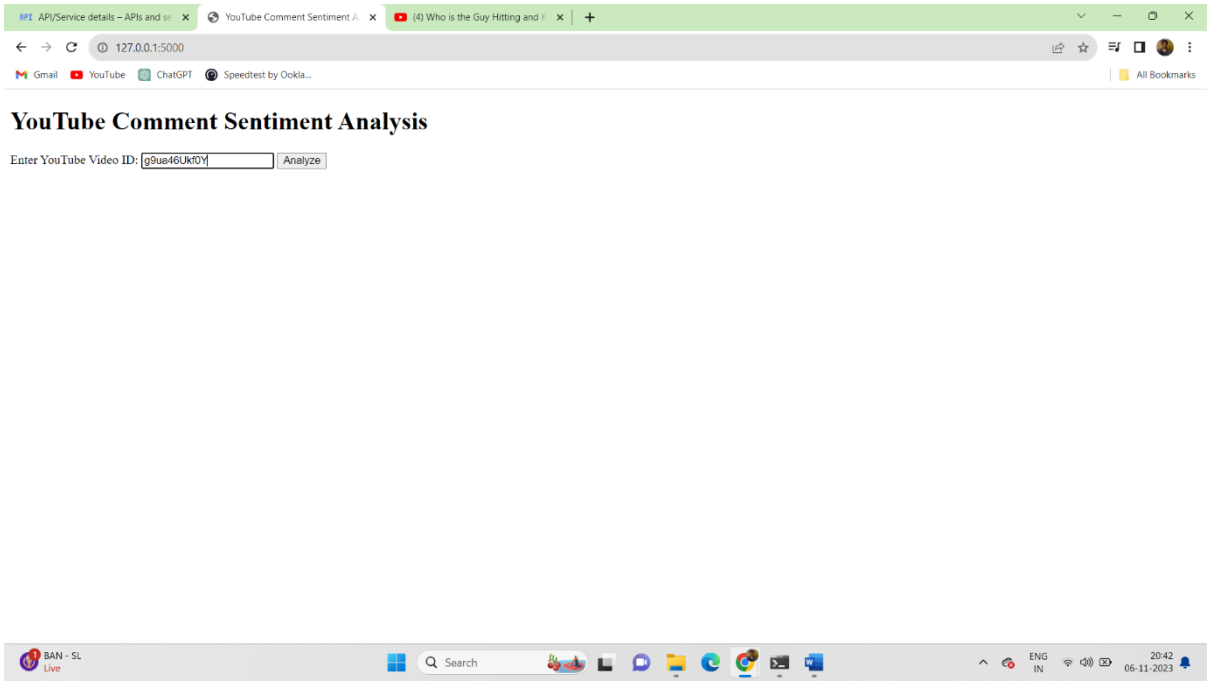
Copy http://127.0.0.1:5000 link from CMD and paste it in your web browser and click Enter it will redirect to index.html and it will ask for ID

RUN:

Redirects to index.html

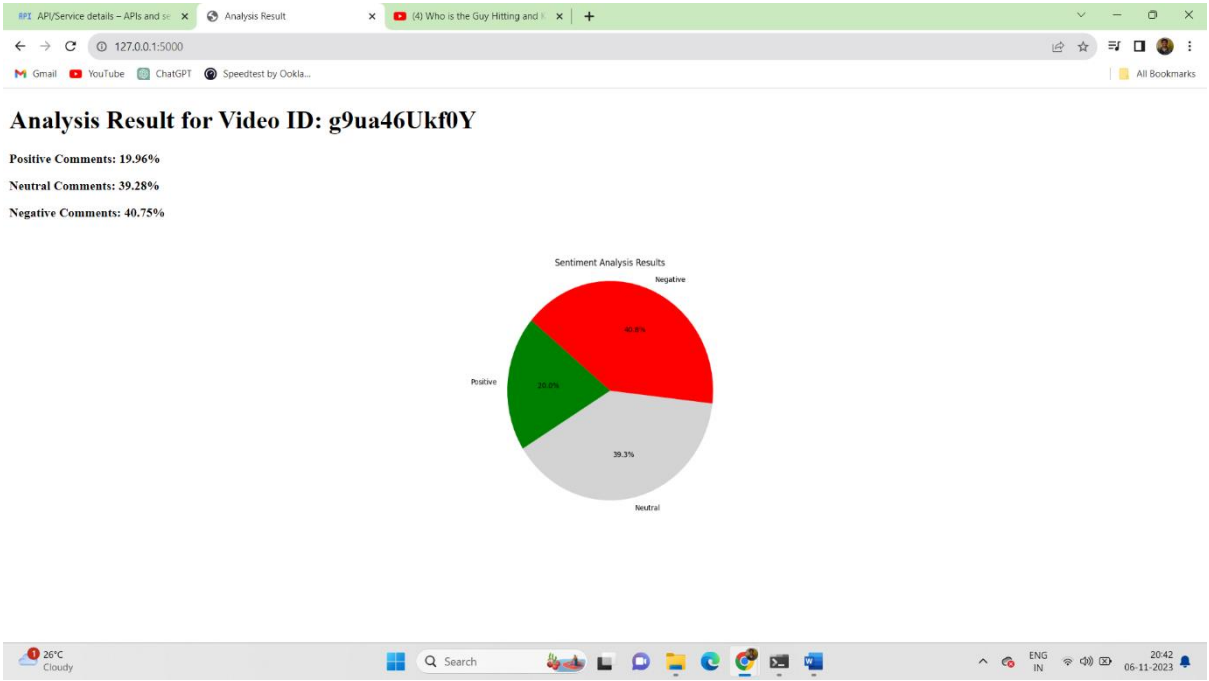


After entering ID and when we click on Analysis



Next it will analyse and give output

Redirects to result.html



Conclusion:

This project provides a practical solution for quickly analysing the sentiments expressed in YouTube video comments. The combination of Flask, YouTube Data API, TextBlob, and Matplotlib makes it possible to create an interactive and user-friendly sentiment analysis tool. Content creators and marketers can use this tool to gain insights into their video's audience sentiment and adapt their content accordingly. The project can be extended to include additional features and further improve the user experience.