MEASI INSTITUTE OF INFORMATION TECHNOLOGY
(Approved by AICTE & Affiliated to University of Madras)
CHENNAI – 600 014

MEASI
Institute *of*
Information
Technology

MASTER OF COMPUTER APPLICATIONS

ACADEMIC YEAR 2024-2025

SEMESTER – II

Practical Record

435E2D – Social Networking Lab

REG. NO : _____

NAME : _____

BATCH : _____

MEASI INSTITUTE OF INFORMATION TECHNOLOGY
(Approved by AICTE & Affiliated to University of Madras)
CHENNAI- 600 014

MCA PRACTICAL

435E2D – Social Networking Lab

Academic Year 2024-2025

Semester – II

NAME: **CLASS :**

REG.NO : **BATCH :**

This is to certify that this is the bonafide record of work done in the Computer Science Laboratory of **MEASI Institute of Information Technology, submitted for the University of Madras** Practical Examination held on..... at **MEASI Institute of Information Technology, Chennai-600 014.**

STAFF IN-CHARGE **DIRECTOR**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

CREATING AND EXPLORING TWITTER'S API

Ex. No:01
Date:

AIM:

To create and explore Twitter's API for data analysis and application development using python.

ALGORITHM:

STEP 1: Sign up for a Twitter Developer Account at the Twitter Developer Platform.

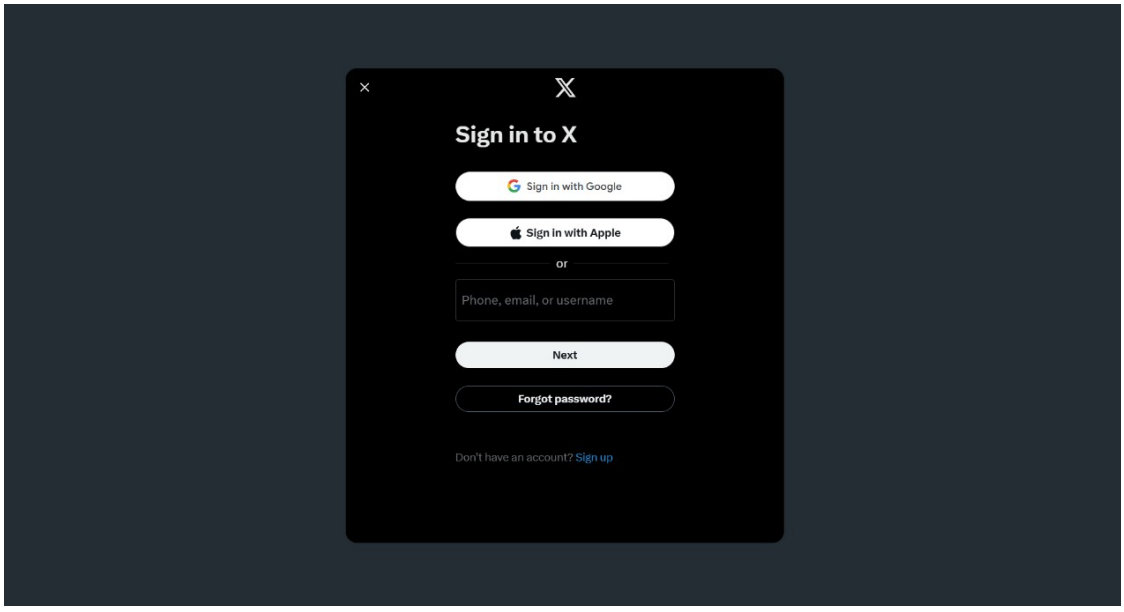
STEP 2: Create a Twitter Application by providing essential details such as name, description, and website.

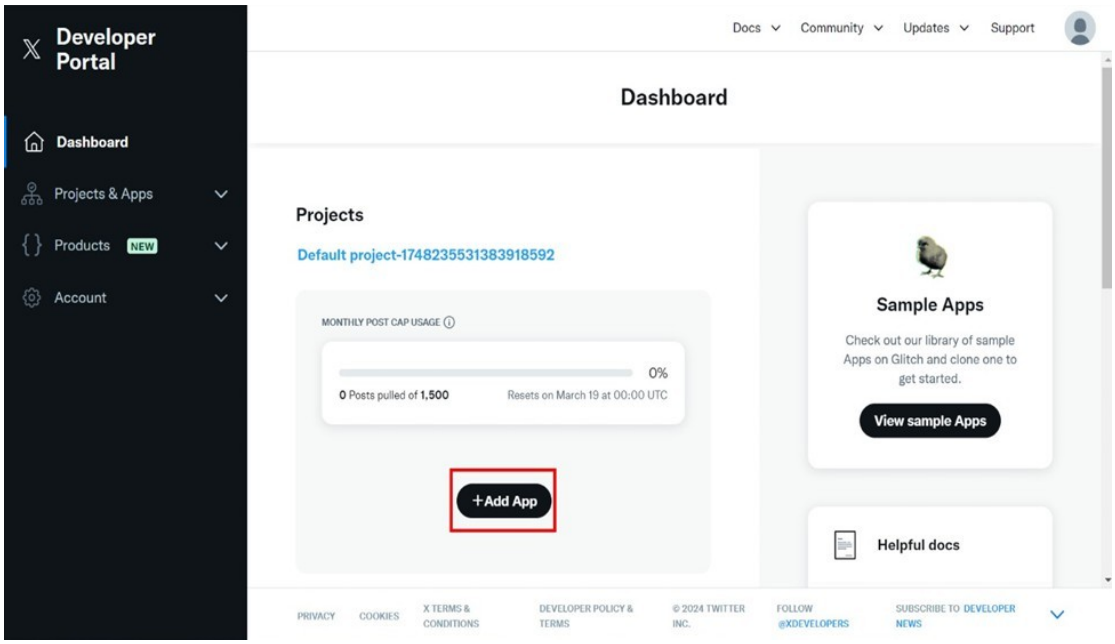
STEP 3: Obtain API Keys and Tokens from the "Keys and tokens" tab after creating the application.

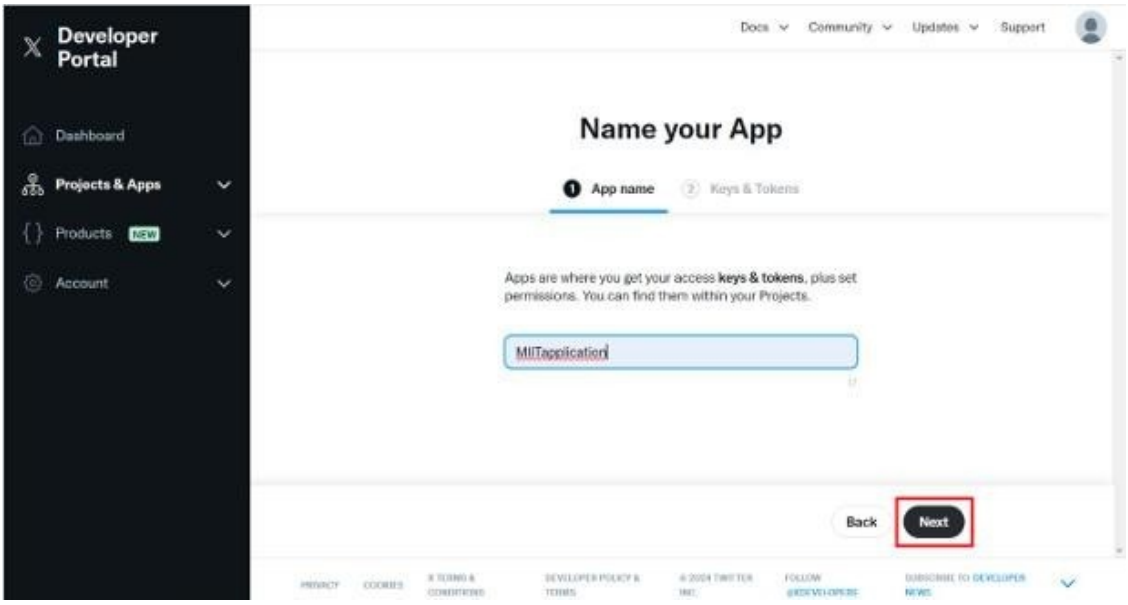
STEP 4: Install Tweepy, a Python library for accessing the Twitter API, using pip: pip install tweepy.

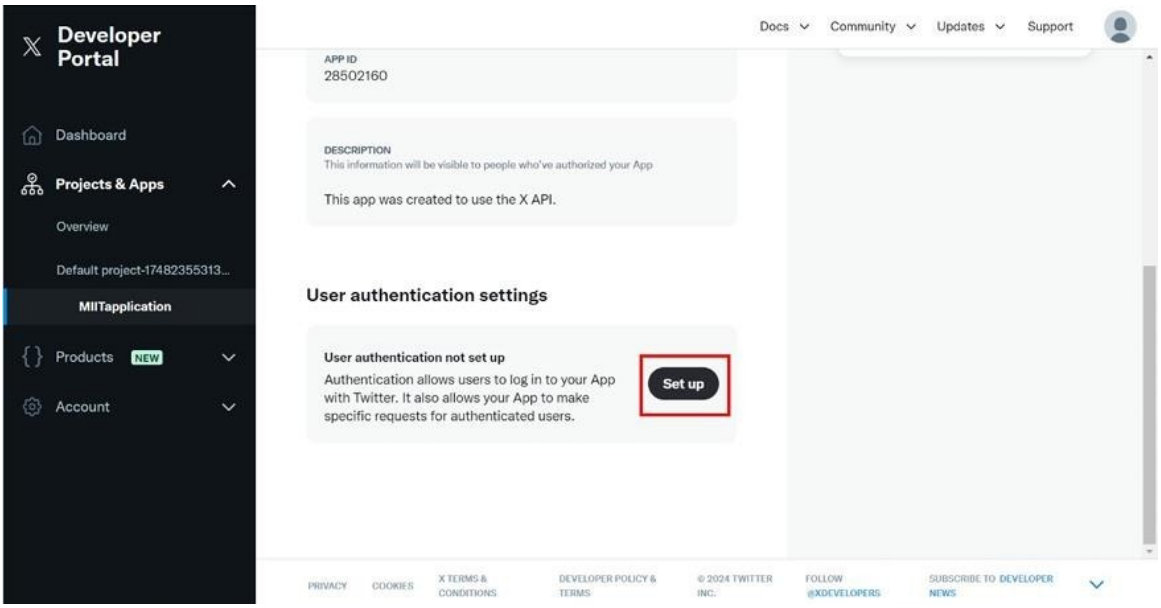
STEP 5: Access Twitter's API with Tweepy to authenticate your application and retrieve data programmatically.

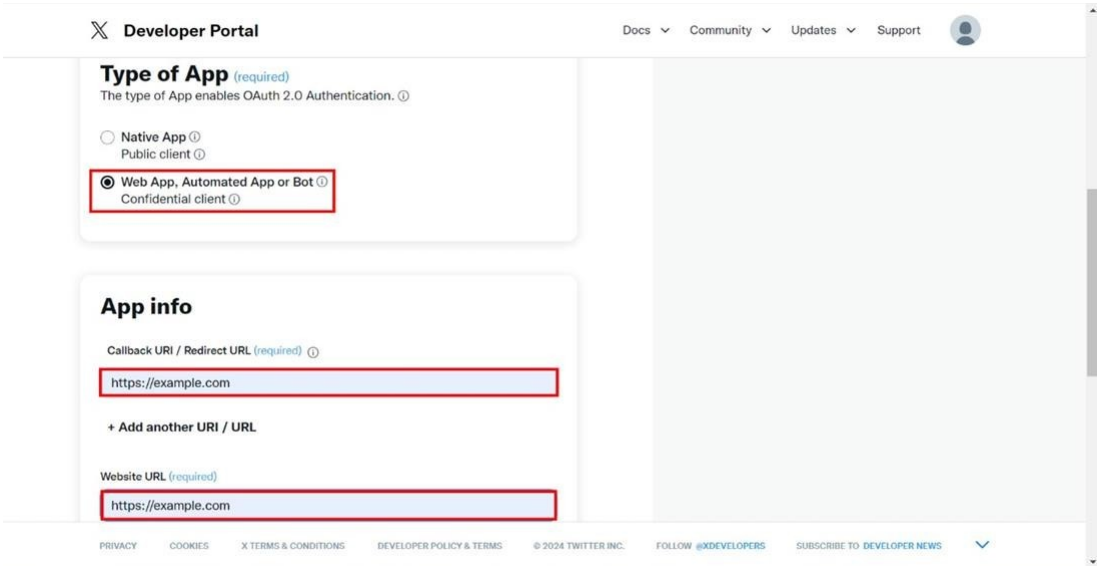
STEP 6: Fetch user information from Twitter's API to gather details such as user profile

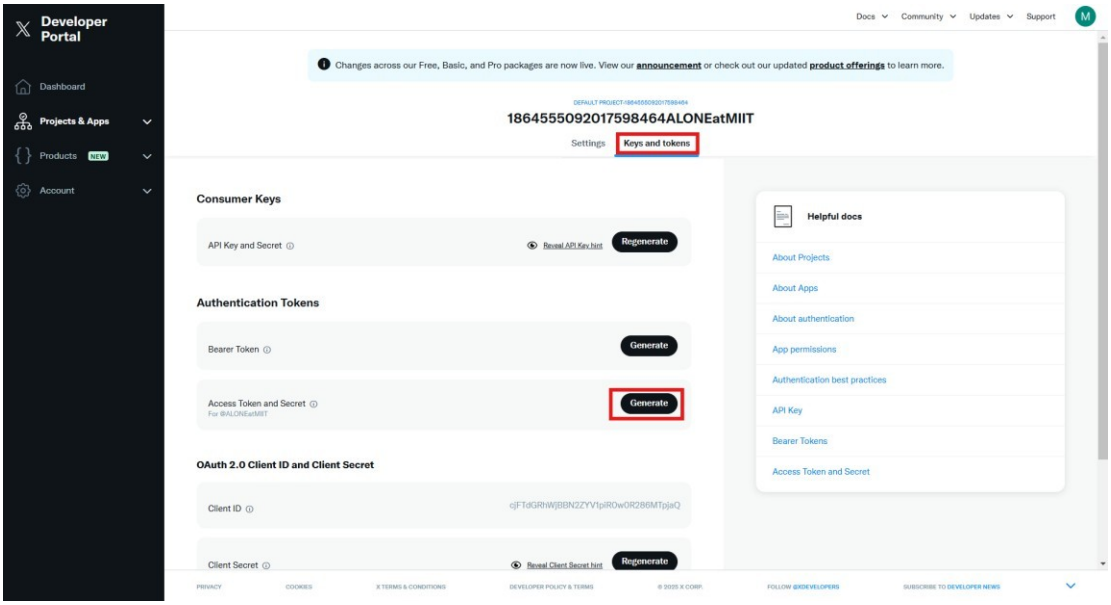


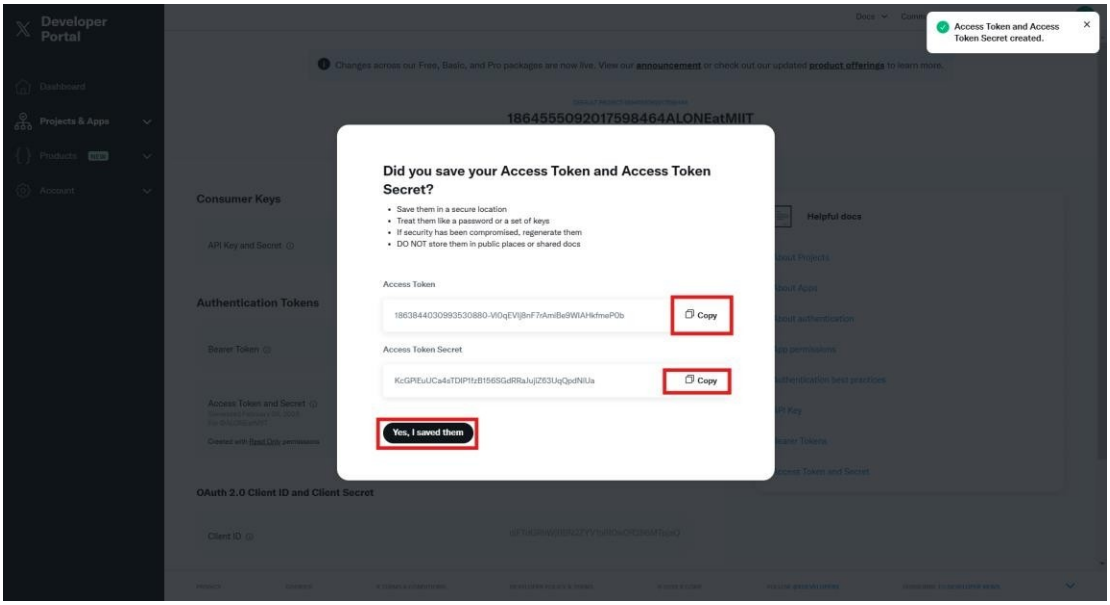

Developer Portal
Dashboard
Projects & Apps
Products NEW
Account
Docs
Community
Updates
Support
Dashboard
Projects
Default project-1748235531383918592
MONTHLY POST CAP USAGE
0%
0 Posts pulled of 1,500
Resets on March 19 at 00:00 UTC
+Add App
Sample Apps
Check out our library of sample Apps on Glitch and clone one to get started.
View sample Apps
Helpful docs
PRIVACY
COOKIES
X TERMS & CONDITIONS
DEVELOPER POLICY & TERMS
© 2024 TWITTER INC.
FOLLOW @XDEVELOPERS
SUBSCRIBE TO DEVELOPER NEWS


Developer Portal
Dashboard
Projects & Apps
Products NEW
Account
Docs
Community
Updates
Support
Name your App
1 App name
2 Keys & Tokens
Apps are where you get your access keys & tokens, plus set permissions. You can find them within your Projects.
MIITapplication
Back
Next


Developer Portal
Dashboard
Projects & Apps
Overview
Default project-17482355313...
MIITapplication
Products
NEW
Account
Docs
Community
Updates
Support
APP ID
28502160
DESCRIPTION
This information will be visible to people who've authorized your App
This app was created to use the X API.
User authentication settings
User authentication not set up
Authentication allows users to log in to your App with Twitter. It also allows your App to make specific requests for authenticated users.
Set up
PRIVACY
COOKIES
X TERMS & CONDITIONS
DEVELOPER POLICY & TERMS
© 2024 TWITTER INC.
FOLLOW @XDEVELOPERS
SUBSCRIBE TO DEVELOPER NEWS


Developer Portal
Docs
Community
Updates
Support
Type of App (required)
The type of App enables OAuth 2.0 Authentication.
Native App
Public client
Web App, Automated App or Bot
Confidential client
App info
Callback URI / Redirect URL (required)
https://example.com
+ Add another URI / URL
Website URL (required)
https://example.com
PRIVACY
COOKIES
X TERMS & CONDITIONS
DEVELOPER POLICY & TERMS
© 2024 TWITTER INC.
FOLLOW @XDEVELOPERS
SUBSCRIBE TO DEVELOPER NEWS

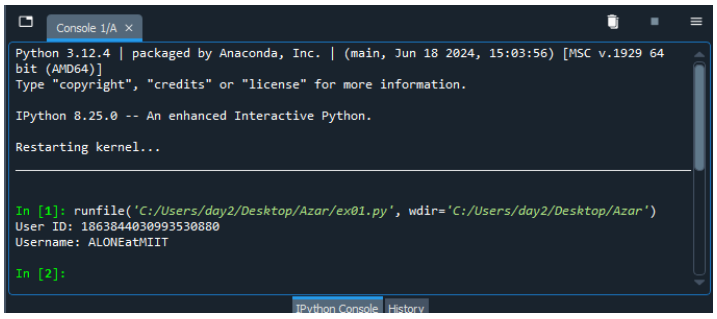

Developer Portal
Dashboard
Projects & Apps
Products
Account
Docs
Community
Updates
Support
Changes across our Free, Basic, and Pro packages are now live. View our announcement or check out our updated product offerings to learn more.
1864555092017598464ALONEatMIIT
Settings
Keys and tokens
Consumer Keys
API Key and Secret
Reveal API Key hint
Regenerate
Authentication Tokens
Bearer Token
Generate
Access Token and Secret
Generate
OAuth 2.0 Client ID and Client Secret
Client ID
Client Secret
Reveal Client Secret hint
Regenerate
Helpful docs
About Projects
About Apps
About authentication
App permissions
Authentication best practices
API Key
Bearer Tokens
Access Token and Secret


Developer Portal
Access Token and Access Token Secret created.
1864555092017598464ALONEatMIIT
Did you save your Access Token and Access Token Secret?
Save them in a secure location
Treat them like a password or a set of keys
If security has been compromised, regenerate them
DO NOT store them in public places or shared docs
Access Token
1863844030993530880-Vl0qEVIjBnF7rAmiBe9WIAHkfmePOb
Copy
Access Token Secret
KcGPtEuUCa4sTDIPIfzB156SGdRRaJujZ63UqQpdNlJa
Copy
Yes, I saved them
Consumer Keys
Authentication Tokens
OAuth 2.0 Client ID and Client Secret
Helpful docs

CODING:

```
import requests
from requests_oauthlib import OAuth1

def get_authenticated_user(api_key, api_secret_key, access_token,
  access_token_secret): url = "https://api.twitter.com/2/users/me"
  auth =
    OAuth1
    ( api_ke
    y,
    client_secret=api_secret_key,
    resource_owner_key=access_to
    ken,
    resource_owner_secret=access_token_secret
  )
  response = requests.get(url,auth=auth)
  if response.status_code == 200:
    user_data = response.json()
    print("User ID:", user_data["data"]["id"])
    print("Username:", user_data["data"]
  ["username"]) else:
    print("Error:", response.status_code, response.text)

api_key = "Your API Key"
api_secret_key = "Your API Secret
Key" access_token = "Your Access
Token"
access_token_secret = "Your Access Token Secret Key"
get_authenticated_user(api_key, api_secret_key, access_token, access_token_secret)
```

OUTPUT:



RESULT:

Thus, The Creating and Exploring Twitter's Api program has been successfully executed and output is verified.

ANALYZING AND VISUALIZING TWEETS AND TWEET ENTITIES WITH FREQUENCY ANALYSIS

Ex. No:02
Date:

AIM:

To write a python program to analyze and visualize tweets and tweet entities with frequency analysis.

ALGORITHM:

STEP 1: Authenticate with Twitter's API using OAuth1UserHandler and tweepy library.

STEP 2: Fetch tweets based on a specified query from the authenticated API.

STEP 3: Preprocess the text data by tokenizing, removing stopwords, and performing frequency analysis.

STEP 4: Count the frequency of words or entities in the preprocessed text data using NLTK's Counter class.

STEP 5: Visualize the topmost frequent words in tweets using matplotlib.

CODING:

```
import tweepy
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from collections import Counter
import matplotlib.pyplot as plt

# Authenticate with Twitter API
consumer_key = "YOUR_CONSUMER_KEY"
consumer_secret = "YOUR_CONSUMER_SECRET"
access_token = "YOUR_ACCESS_TOKEN"
access_token_secret = "YOUR_ACCESS_TOKEN_SECRET"
```

```
# Set up authentication
auth = tweepy.OAuth1UserHandler(consumer_key, consumer_secret, access_token,
access_token_secret)
api = tweepy.API(auth)

# Set up NLTK
nltk.download('punkt')
nltk.download('stopwords')

# Fetch tweets
def fetch_tweets(query, count=100):
    tweets = api.search_tweets(q=query, count=count, lang="en",
tweet_mode="extended")
    return tweets

# Preprocess text data
def preprocess_text(text):
    words = word_tokenize(text.lower())
    stopwords_set = set(stopwords.words('english'))
    filtered_words = [word for word in words if word.isalnum() and word not in
stopwords_set]
    return filtered_words

# Perform frequency analysis
def perform_frequency_analysis(tweets):
    text = ' '.join([tweet.full_text for tweet in tweets])
    processed_text = preprocess_text(text)
    word_freq = Counter(processed_text)
    return word_freq

# Visualize results
def visualize_results(word_freq, top_n=10):
    top_words = dict(word_freq.most_common(top_n))
    plt.figure(figsize=(10, 5))
    plt.bar(top_words.keys(), top_words.values(), color='skyblue')
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.title(f'Top {top_n} Most Frequent Words in Tweets')
    plt.xticks(rotation=45)
    plt.show()

# Example usage
query = "python programming"
tweets = fetch_tweets(query)
word_freq = perform_frequency_analysis(tweets)
visualize_results(word_freq)
```

OUTPUT:

Top 10 Most Frequent Words in Tweets:
1. python - 50
2. programming - 30
3. code - 20
4. learning - 15
5. language - 12
6. tutorial - 10
7. development - 8
8. beginners - 7
9. projects - 5
10. community - 3

RESULT:

Thus, the program has been successfully executed and output is verified.

CREATING AND EXPLORING FACEBOOK'S SOCIAL GRAPH API

Ex. No: 03
Date:

AIM:

To use Graph API Explorer to create and explore a Facebook's social graph API.

ALGORITHM:

STEP 1: Sign in to Facebook Developer Portal.

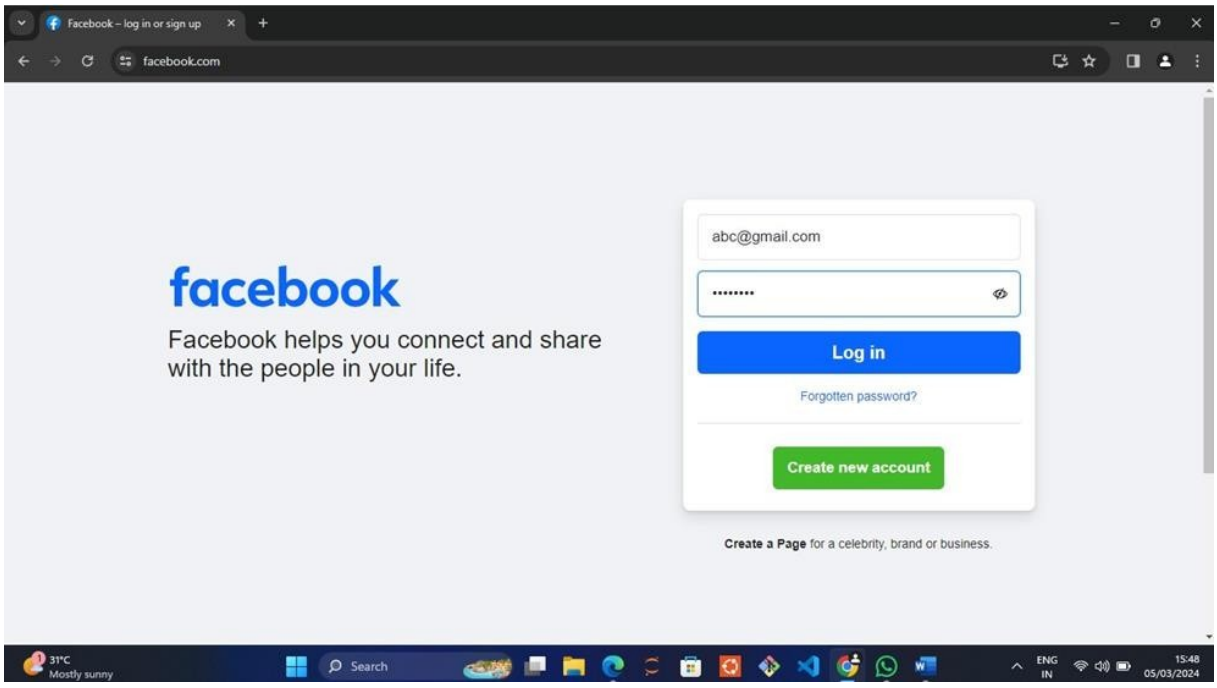
STEP 2: Create a new app and set up Facebook Login.

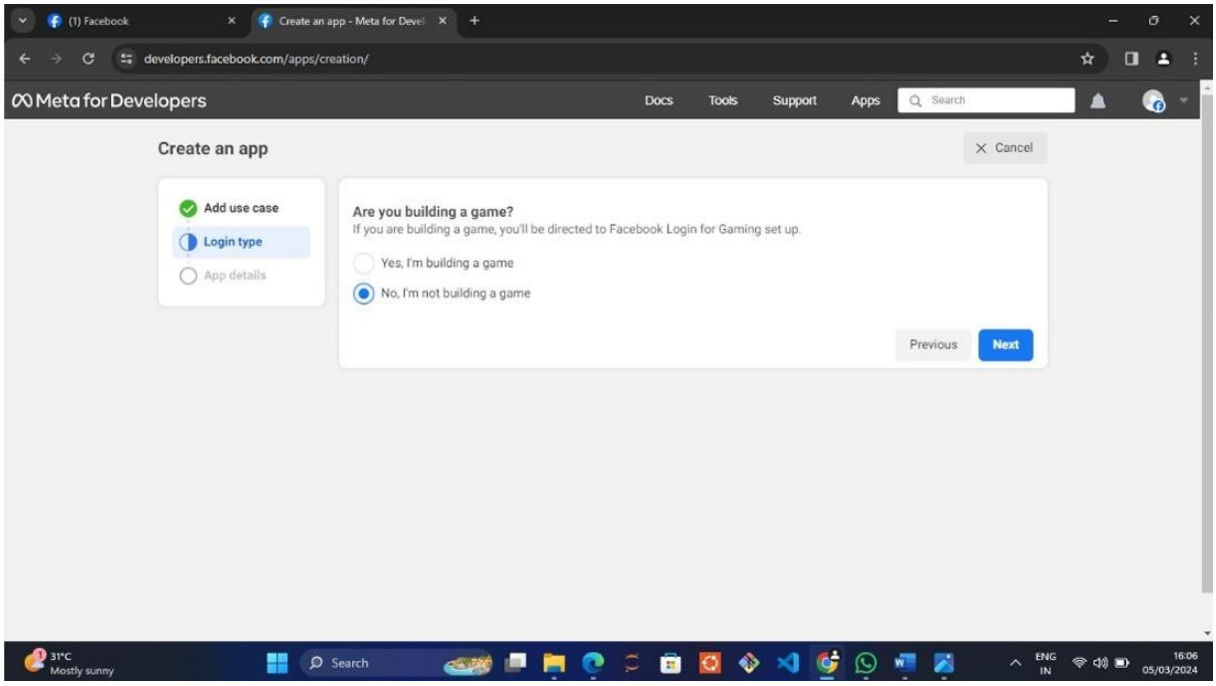
STEP 3: Customize use cases for user data.

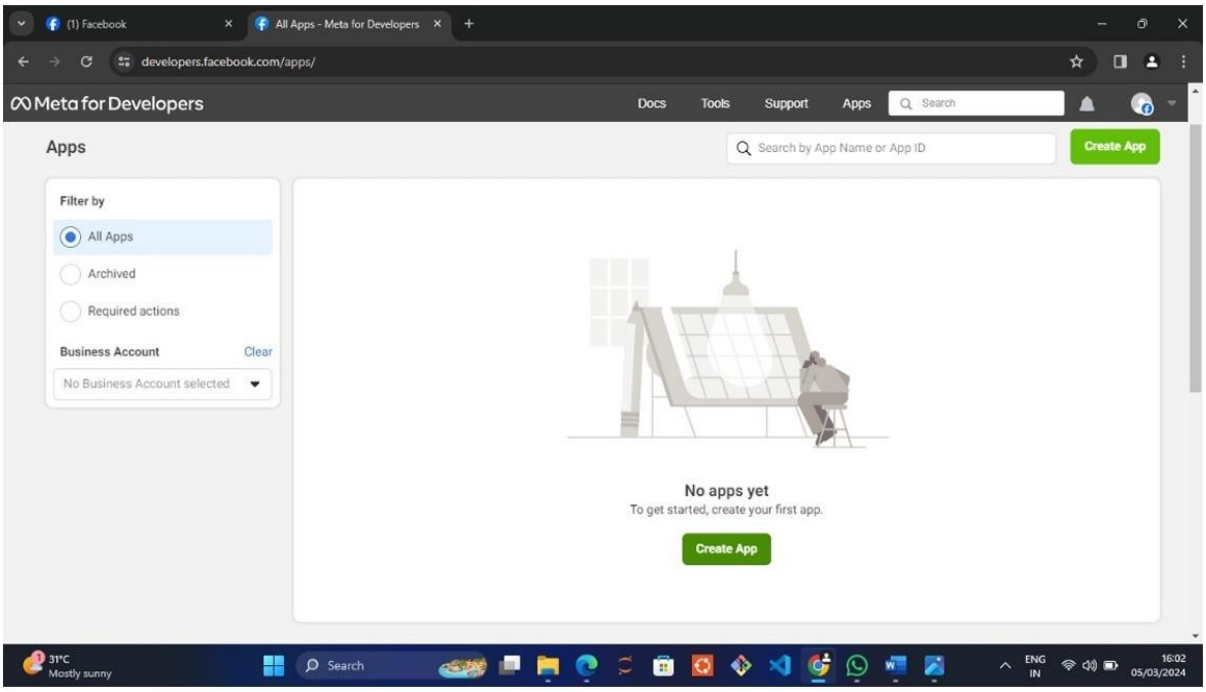
STEP 4: Generate Access Token in Graph API Explorer.

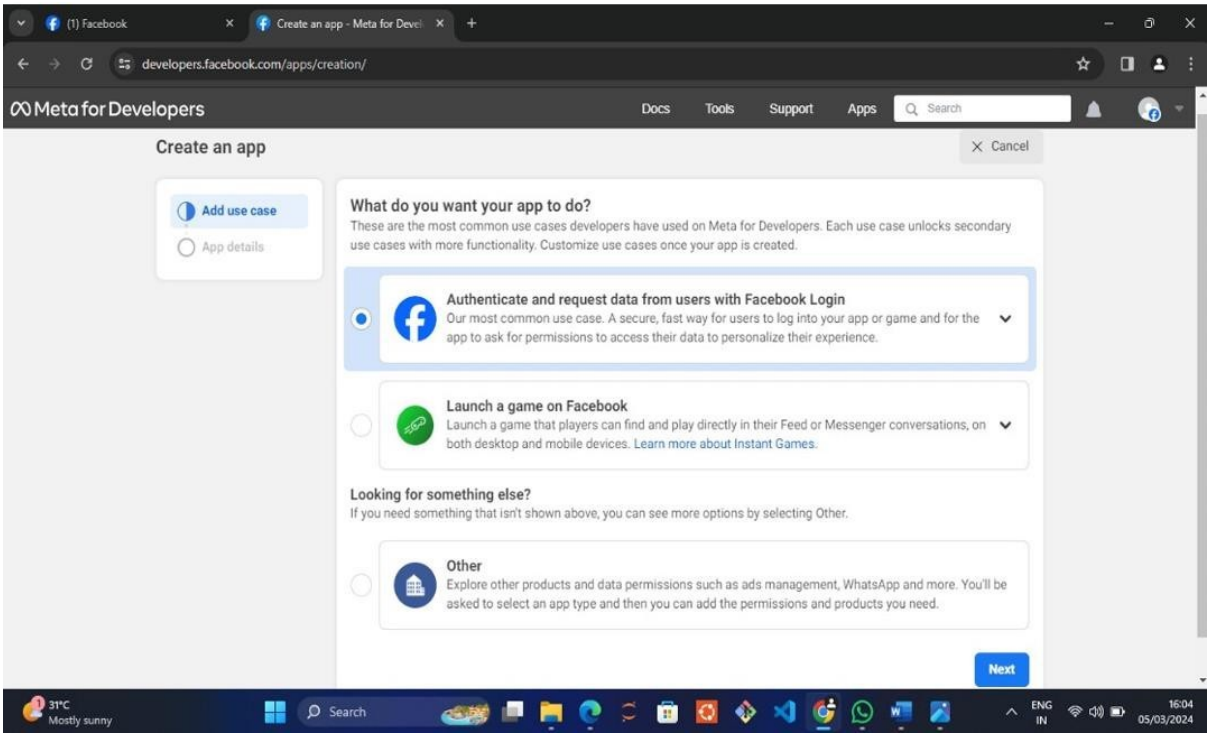
STEP 5: Choose permissions and submit.

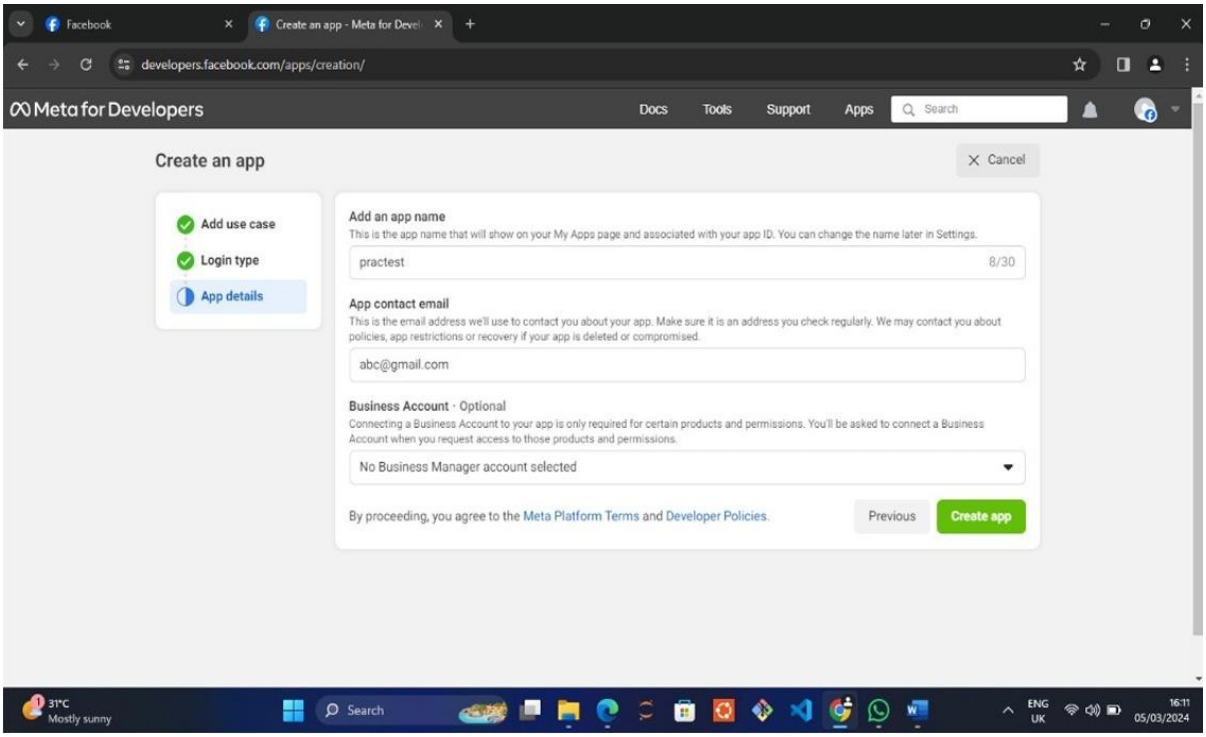
STEP 6: Retrieve user data: ID, name, age, gender, friends list, etc.,

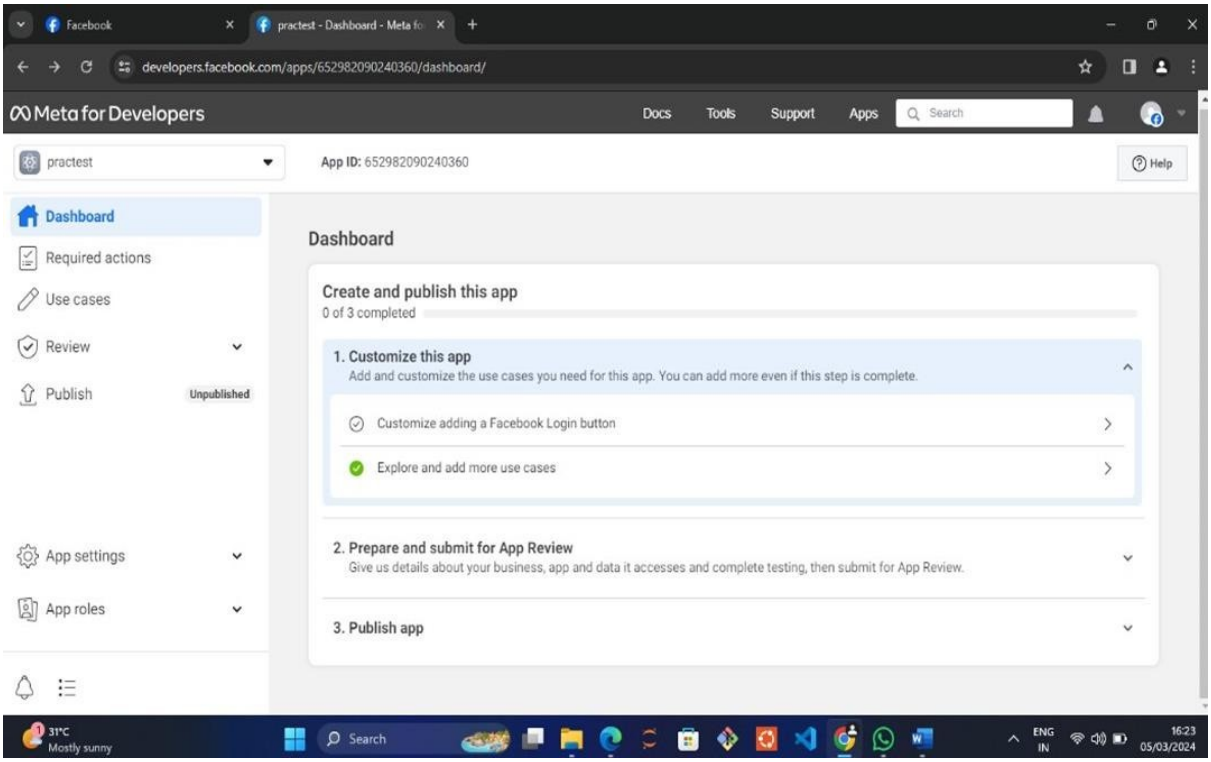


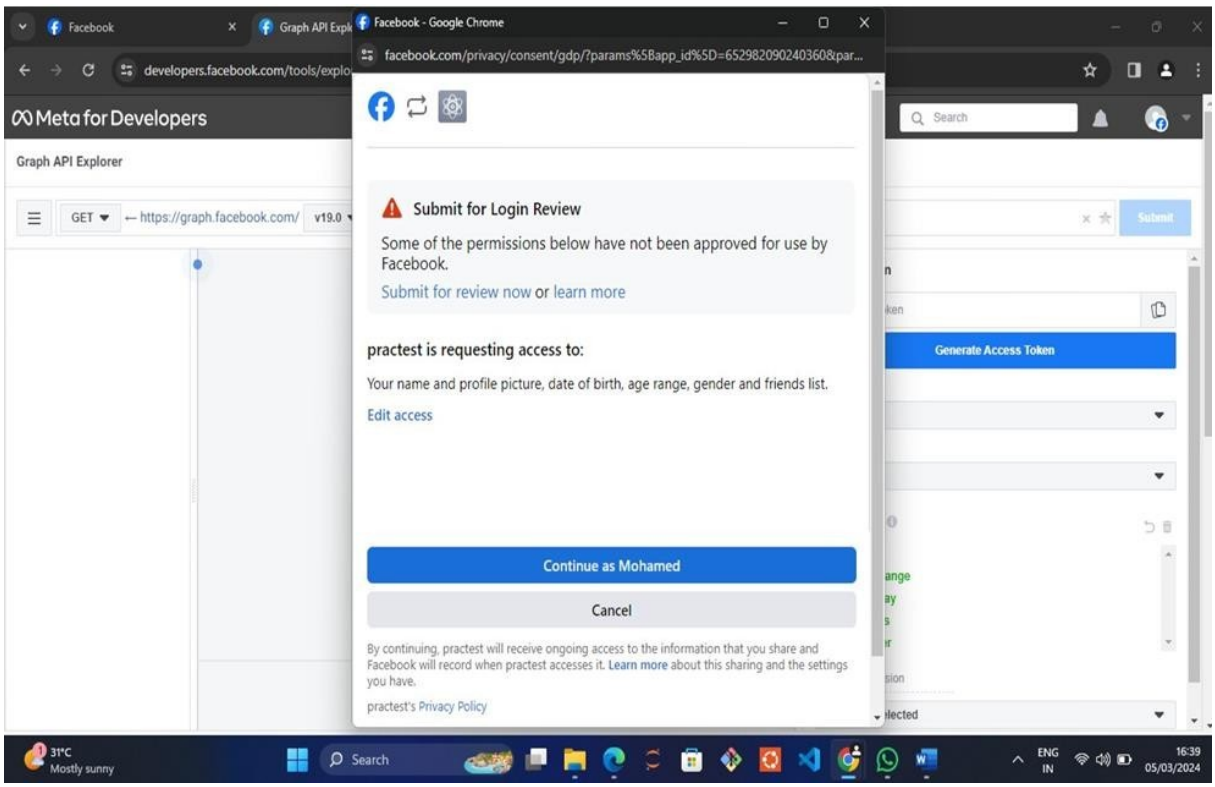

(1) Facebook
Create an app - Meta for Deve
developers.facebook.com/apps/creation/
Meta for Developers
Docs
Tools
Support
Apps
Search
Create an app
Cancel
Add use case
Login type
App details
Are you building a game?
If you are building a game, you'll be directed to Facebook Login for Gaming set up.
Yes, I'm building a game
No, I'm not building a game
Previous
Next
31°C
Mostly sunny
Search
ENG
IN
16:06
05/03/2024


(1) Facebook
All Apps - Meta for Developers
developers.facebook.com/apps/
Meta for Developers
Docs
Tools
Support
Apps
Search
Apps
Search by App Name or App ID
Create App
Filter by
All Apps
Archived
Required actions
Business Account
Clear
No Business Account selected
No apps yet
To get started, create your first app.
Create App
31°C
Mostly sunny
Search
ENG
IN
16:02
05/03/2024

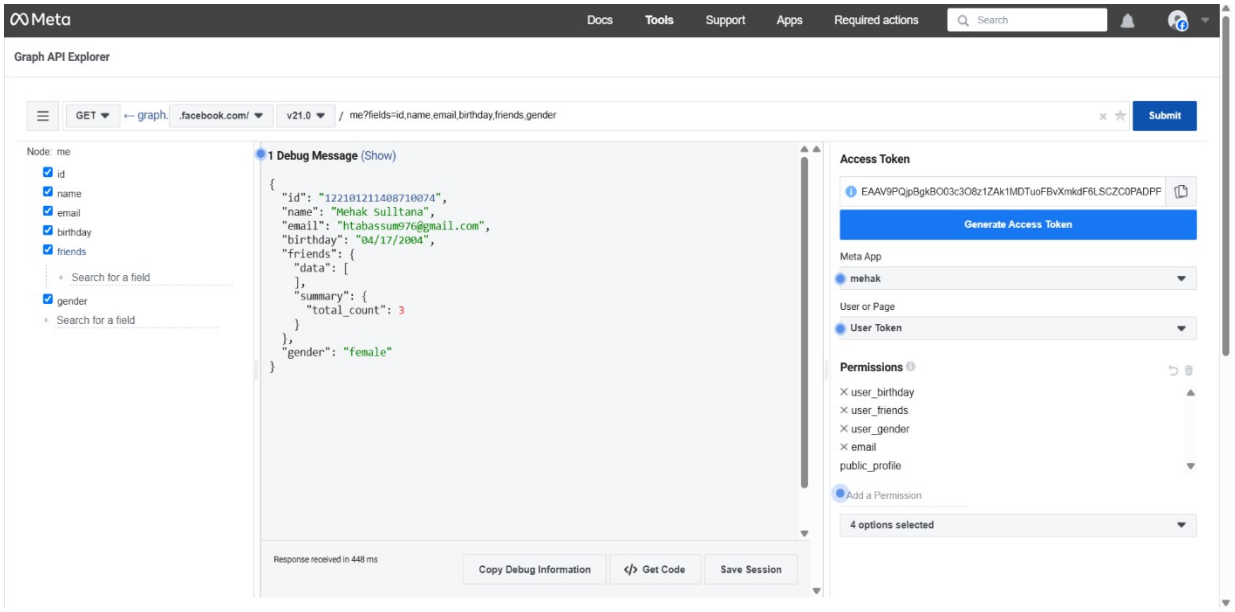

(1) Facebook
Create an app - Meta for Devel
developers.facebook.com/apps/creation/
Meta for Developers
Docs
Tools
Support
Apps
Search
Create an app
Cancel
Add use case
App details
What do you want your app to do?
These are the most common use cases developers have used on Meta for Developers. Each use case unlocks secondary use cases with more functionality. Customize use cases once your app is created.
Authenticate and request data from users with Facebook Login
Our most common use case. A secure, fast way for users to log into your app or game and for the app to ask for permissions to access their data to personalize their experience.
Launch a game on Facebook
Launch a game that players can find and play directly in their Feed or Messenger conversations, on both desktop and mobile devices. Learn more about Instant Games.
Looking for something else?
If you need something that isn't shown above, you can see more options by selecting Other.
Other
Explore other products and data permissions such as ads management, WhatsApp and more. You'll be asked to select an app type and then you can add the permissions and products you need.
Next
31°C
Mostly sunny
Search
ENG
IN
16:04
05/03/2024


Facebook
Create an app - Meta for Devel
developers.facebook.com/apps/creation/
Meta for Developers
Docs
Tools
Support
Apps
Search
Create an app
Cancel
Add use case
Login type
App details
Add an app name
This is the app name that will show on your My Apps page and associated with your app ID. You can change the name later in Settings.
practest
8/30
App contact email
This is the email address we'll use to contact you about your app. Make sure it is an address you check regularly. We may contact you about policies, app restrictions or recovery if your app is deleted or compromised.
abc@gmail.com
Business Account - Optional
Connecting a Business Account to your app is only required for certain products and permissions. You'll be asked to connect a Business Account when you request access to those products and permissions.
No Business Manager account selected
By proceeding, you agree to the Meta Platform Terms and Developer Policies.
Previous
Create app
31°C
Mostly sunny
Search
ENG
UK
16:11
05/03/2024


Facebook
practest - Dashboard - Meta fo
developers.facebook.com/apps/652982090240360/dashboard/
Meta for Developers
Docs
Tools
Support
Apps
Search
practest
App ID: 652982090240360
Help
Dashboard
Required actions
Use cases
Review
Publish
Unpublished
App settings
App roles
Dashboard
Create and publish this app
0 of 3 completed
1. Customize this app
Add and customize the use cases you need for this app. You can add more even if this step is complete.
Customize adding a Facebook Login button
Explore and add more use cases
2. Prepare and submit for App Review
Give us details about your business, app and data it accesses and complete testing, then submit for App Review.
3. Publish app
31°C
Mostly sunny
Search
ENG
IN
16:23
05/03/2024


Facebook
Graph API Expl
developers.facebook.com/tools/explo
Meta for Developers
Graph API Explorer
GET
https://graph.facebook.com/
v19.0
Search
Submit
Generate Access Token
Facebook - Google Chrome
facebook.com/privacy/consent/gdp/?params%5Bapp_id%5D=652982090240360&par...
Submit for Login Review
Some of the permissions below have not been approved for use by Facebook.
Submit for review now or learn more
practest is requesting access to:
Your name and profile picture, date of birth, age range, gender and friends list.
Edit access
Continue as Mohamed
Cancel
By continuing, practest will receive ongoing access to the information that you share and Facebook will record when practest accesses it. Learn more about this sharing and the settings you have.
practest's Privacy Policy
31°C
Mostly sunny
Search
ENG
IN
16:39
05/03/2024

OUTPUT:



RESULT:

Thus, the program has been successfully executed and output is verified.

ANALYZING THE FACEBOOK'S SOCIAL GRAPH CONNECTIONS

Ex. No: 04
Date:

AIM:

To write a python program to analyse the Facebook's Social Graph connections.

ALGORITHM:

STEP 1: Create a Facebook Developer account, set up a new app, and obtain API credentials (App ID and App Secret).

STEP 2: Install the facebook-sdk library using pip: pip install facebook-sdk.

STEP 3: Authenticate your app with the Graph API using your credentials.

STEP 4: Query the Graph API to retrieve various data about users, friends, connections, and posts.

STEP 5: Analyze the retrieved data for insights and further actions.

CODING:

```
import facebook

# Define your access token and user ID
access_token = 'YOUR_ACCESS_TOKEN'
user_id = 'YOUR_USER_ID'

# Initialize the Graph API with your access token
graph = facebook.GraphAPI(access_token)

# Retrieve the list of friends
friends = graph.get_connections(user_id, 'friends')

# Print the friend names
for friend in friends['data']:
print(friend['name'])
```

Note: Replace 'YOUR_ACCESS_TOKEN' and 'YOUR_USER_ID' with your actual access token and user ID, respectively

OUTPUT:

John Doe
Jane Smith
Michael Johnson
….

RESULT:

Thus, the program has been successfully executed and output is verified.

CREATING AND EXPLORING LINKEDIN API

Ex. No: 05
Date:

AIM:

To write a python program to implement the Creating and Exploring LinkedIn's API

ALGORITHM:

STEP 1: Set up a LinkedIn Developer Account Go to the LinkedIn Developer portal (https://developer.linkedin.com/) and create an account. Once logged in, create a new app to obtain the necessary API credentials.

STEP 2: Obtain API Credentials After creating an app, LinkedIn will provide you with a Client ID and Client Secret. These credentials will be used to authenticate your requests to the LinkedIn API.

STEP 3: Install Required Libraries You will need to install the requests library to make HTTP requests to the LinkedIn API. You can install it using pip: pip install requests

STEP 4: Authenticate Your Application Use the obtained Client ID and Client Secret to authenticate your application with LinkedIn. This typically involves obtaining an access token that you will use for subsequent API requests. LinkedIn uses OAuth 2.0 for authentication.

STEP 5: Make API Requests Once authenticated, you can start making requests to the LinkedIn API endpoints to retrieve data such as user profiles, connections, companies, etc.

CODING:

```
import requests

# Your LinkedIn API credentials
CLIENT_ID = 'your_client_id'
CLIENT_SECRET = 'your_client_secret'
ACCESS_TOKEN = 'your_access_token'

# URL for accessing LinkedIn's API
API_URL = 'https://api.linkedin.com/v2/me'
```

```
# Set up headers with access token
headers = {'Authorization': f'Bearer {ACCESS_TOKEN}', 'Connection': 'Keep-
Alive'}
response = requests.get(API_URL, headers=headers)

# Check if request was successful
if response.status_code == 200:
    profile_data = response.json()
    print(profile_data)
else:
    print(f"Error: {response.status_code} - {response.text}")
```

OUTPUT:

```
"firstName": "John",
"lastName": "Doe",
"headline": "Software Engineer at Example Company"
```

RESULT:

Thus, the program has been successfully executed and output is verified.

DOWNLOADING LINKEDIN CONNECTIONS AS A CSV FILE

Ex. No. :06
Date :

AIM:

To write a python program to download LinkedIn connections as a CSV file.

ALGORITHM & CODING:

Step 1: Set up a LinkedIn Developer Account and Create an App

- Go to the LinkedIn Developer portal: LinkedIn Developer

Log in with your LinkedIn account.

- Create a new app and note down the Client ID and Client Secret.

Step 2: Install Required Packages

- Make sure you have requests and pandas libraries installed.
- You can install them using pip:
  pip install requests pandas.

Step 3: Authenticate with LinkedIn API

- You'll need to authenticate with the LinkedIn API using OAuth 2.0.
- You can find the detailed documentation on how to do this here: LinkedIn OAuth 2.0.

Obtain an access token after authentication.

Step 4: Fetch Connections Data

- Once authenticated, you can use the LinkedIn API to fetch your connections' data. You'll need to make a GET request to the endpoint https://api.linkedin.com/v2/connections.

Step 5: Parse and Save Data

- Parse the JSON response obtained from the API request and extract the relevant information such as name, email, company, etc.

Save this data into a CSV file using the pandas library.

CODING:

```
import requests
import pandas as pd

# Define your access token obtained after authentication
access_token = "YOUR_ACCESS_TOKEN"

# API endpoint to fetch connections data
url = "https://api.linkedin.com/v2/connections"
```

```
# Header containing authorization token
headers = {
"Authorization": f"Bearer {access_token}",
"Connection": "Keep-Alive"
}

# Send GET request to fetch connections data
response = requests.get(url, headers=headers)
if response.status_code == 200:

# Parse JSON response
connections_data = response.json()

# Extract relevant information
connections_list = []
for connection in connections_data["elements"]:
connections_list.append({
"Name": connection["firstName"] + " " + connection.get("lastName", ""),
"Email": connection.get("email", ""),
"Company": connection.get("companyName", ""),
"Position": connection.get("title", "")
})

# Convert to DataFrame
connections_df = pd.DataFrame(connections_list)

# Save to CSV
connections_df.to_csv("linkedin_connections.csv", index=False)
print("Connections data saved successfully!")
else:
print("Failed to fetch connections data. Status code:", response.status_code)
```

Note: Make sure to replace "YOUR_ACCESS_TOKEN" with your actual access token obtained after
authentication.

OUTPUT:

Connections data saved successfully!

RESULT:

Thus, the program has been successfully executed and output is verified.

CREATING AND EXPLORING GOOGLE+ API

Ex. No. : 07
Date :
AIM:

To write a python program to Create and Explore Google+ API.

ALGORITHM:

Step 1: Set up Google Cloud Platform (GCP)

- You need to create a project on Google Cloud Platform and enable the API you want to use.

Step 2: Enable API

- Enable the API you want to use in your project. For example, if you want to use the Google Drive API, you need to enable it in the Google Cloud Console.

Step 3: Create Credentials

- Generate API credentials (API key, OAuth client ID, or service account key) depending on the type of access you need.

Step 4: Install Google Client Library

- Install the google-api-python-client library using pip: pip install google-api-python-client

Step 5: Authentication

- Depending on the API and the type of access (user-based or service account), you'll need to authenticate your requests. For user-based authentication, you can use OAuth2.

Step 6: Make API Requests

- Use the Google Client Library to make requests to the API endpoints. The library provides a Pythonic way to interact with Google APIs.

CODING:

```
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build

 # Set up credentials
Flow = InstalledAppFlow.from_client_secrets_file('credentials.json',scopes=['https://
www.googleapis.com/auth/drive'])
credentials = flow.run_local_server(port=0)

# Build the Drive service
drive_service = build('drive', 'v3', credentials=credentials)

# List files in the user's Drive
```

```
results = drive_service.files().list(pageSize=10).execute()
items = results.get('files', [])
if not items:
print('No filesfound.')
 else:
print('Files:')
for item in items:
print(f'{item["name"]} ({item["id"]})')
```

Remember to replace 'credentials.json' with the path to your OAuth client credentials file.

OUTPUT:

```
Files:
Document.pdf (1234567890)
Image.jpg (0987654321)
```

RESULT:

Thus, the program has been successfully executed and output is verified.

CREATING AND QUERYING HUMAN LANGUAGE DATA WITH TF-IDF

Ex. No:08
Date :

AIM:

To write a Python program to Query Human Language Data with TF-IDF.

ALGORITHM:

Step 1: Import TfidfVectorizer from sklearn.feature_extraction.text and pandas as pd.

Step 2: Define text documents for Speech 1, Speech 2, and Speech 3.

Step 3: Create a corpus list containing all document strings.

Step 4: Initialize TfidfVectorizer with 'english' stop words.

Step 5: Fit and transform the corpus to obtain TF-IDF matrix X.

Step 6: Extract feature names (keywords) using vectorizer.get_feature_names_out().

Step 7: Create a DataFrame df from TF-IDF matrix X.

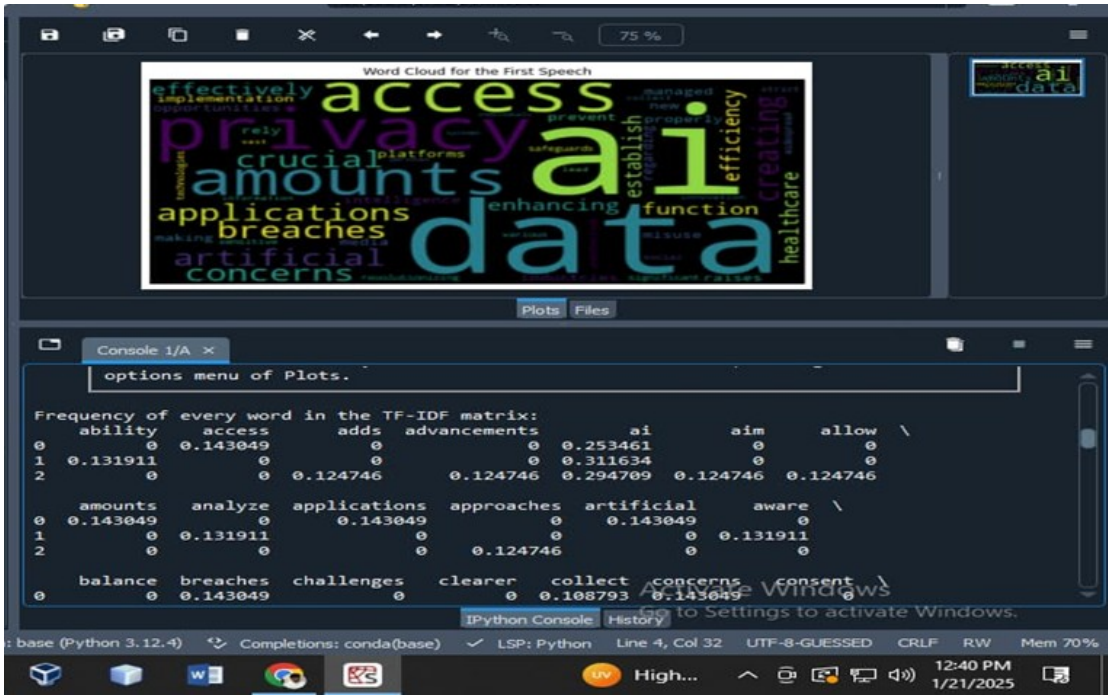
Step 8: Extract and print top 10 most common keywords.

CODING:

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
document_Speech_1 = """ Your Text Here"""""
document_Speech_2 = """ Your Text Here"""""
document_Speech_3 = """ Your Text Here"""""
corpus = [document_Speech_1, document_Speech_2, document_Speech_3 ]
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names_out()
df = pd.DataFrame.sparse.from_spmatrix(X, columns=feature_names)

#Display First Few Columns and Last Few Columns
df.head()

**************************************************************************
# Query1: Get the most common keywords across all speeches

most_common_keywords = df.sum().nlargest(20).index.tolist()
print(most_common_keywords)
**************************************************************************
```

```
#Query - 2 Select documents that contain all specified keywords

keywords=['artificial','privacy']
selected_documents = df[(df[keywords[0]] > 0) & (df[keywords[1]] > 0) ]
print("Selected Documents:")
print(selected_documents.index.tolist())
**********************************************************************
# Query 3 - Create a WordCloud

from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Generate word cloud for a speech
wordcloud = WordCloud().generate_from_frequencies(df.iloc[0])
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud)
plt.title("Word Cloud for the First Speech")
plt.show()
**********************************************************************
#Query 4 - Display Frequency of every word

pd.set_option('display.max_columns', None)
print(df)
**********************************************************************
Query 5 - Check the Keywords present in all documents

keywords_in_all_documents = df.columns[df.min(axis=0) > 0].tolist()
print("Keywords present in all three documents:")
print(keywords_in_all_documents)
```

OUTPUT:



RESULT:

Thus, the program has been successfully executed and output is verified.

CREATING AND EXPLORING GITHUB'S API

Ex. No:09
Date :

AIM:

To write a python program to implement the Creating and Exploring GitHub's API.

ALGORITHM:

GET ALGORITHM:

Step1: Identify the API endpoint:`https://api.github.com/users/{username}`

Step2: Use the `GET` method to retrieve user details

Step3: Set request headers with `"Accept"` and `"User-Agent"`

Step4: Send the request using Postman

Step5: Check the response status code for success or errors

Step6: Extract and process user details from the JSON response

POST ALGORITHM:

Step1: Identify the API endpoint: https://api.github.com/user/repos

Step2: Use the `POST` method to create a repository

Step3: Prepare a JSON request body with repository details

Step4: Authenticate using a Personal Access Token (PAT)

Step5: Send the request using Postman

Step6: Verify the newly created repository on GitHub

DELETE ALGORITHM:

Step1: Identify the API endpoint: https://api.github.com/repos/{owner}/{repo}

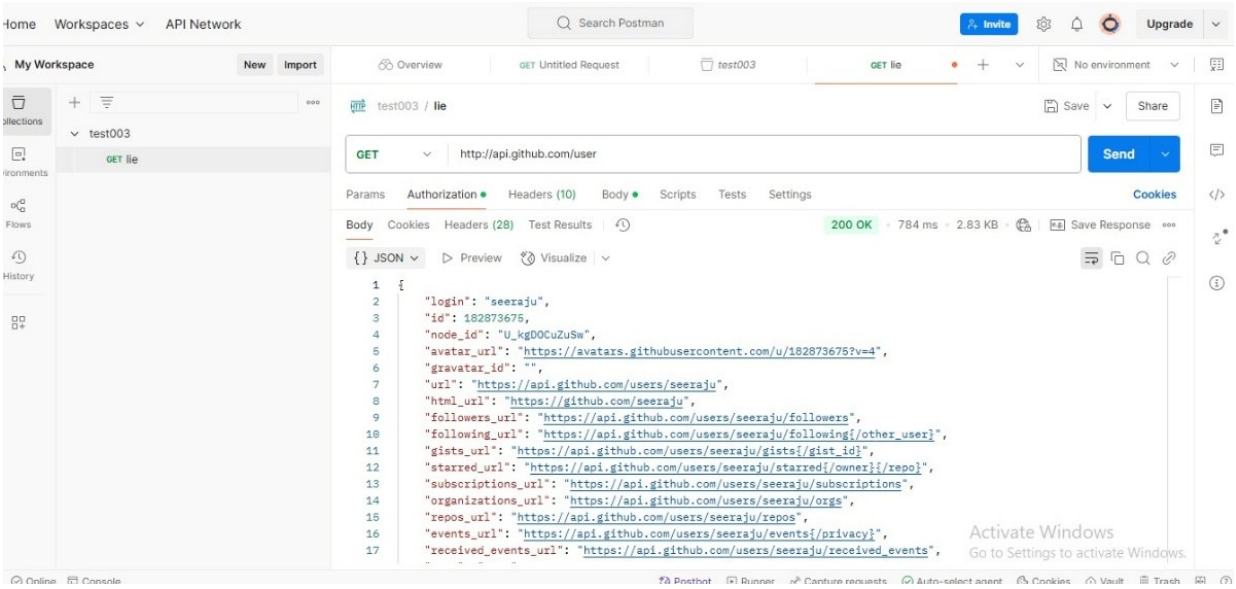
Step2: Use the `DELETE` method to remove a repository

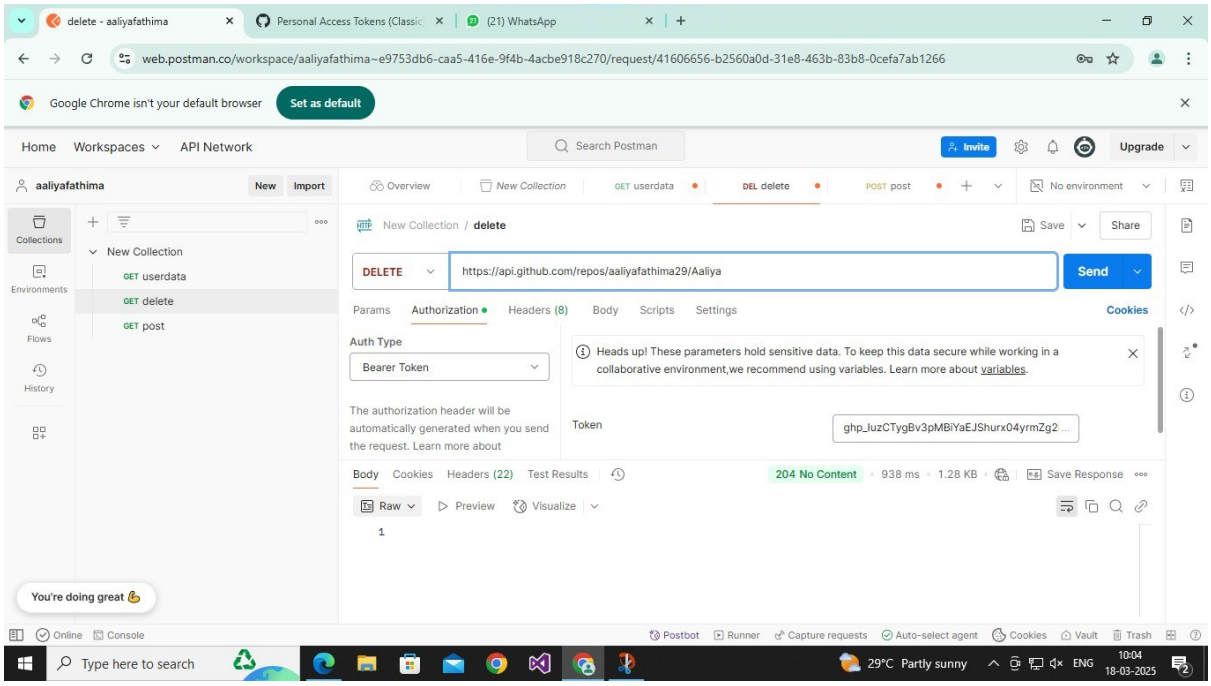
Step3: Authenticate using a Personal Access Token (PAT)

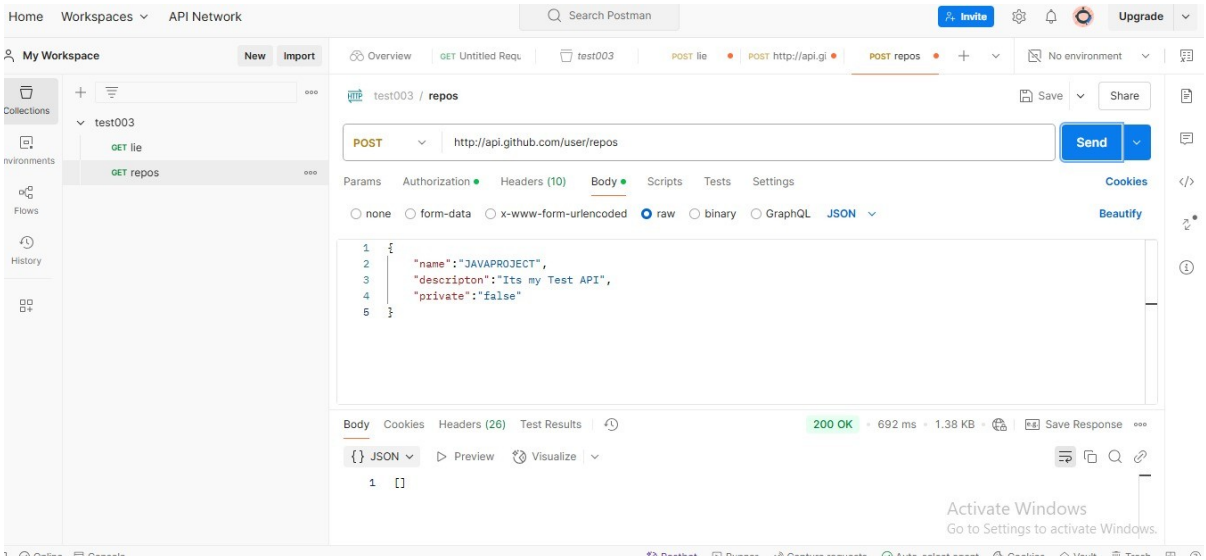
Step4: Send the request using Postman

Step5: Verify the repository has been deleted on GitHub

OUTPUT:







RESULT:

Thus, the program has been successfully executed and output is verified.

ANALYZING GITHUB INTEREST GRAPH

Ex. No:10
Date :

AIM:

To write a python program to analyze GitHub interest graph.

ALGORITHM:

Step 1: Install Python and required libraries using `pip`.

Step 2: Generate a GitHub Personal Access Token for authentication.

Step 3: Use the `requests` library to fetch repository data from the GitHub API.

Step 4: Retrieve repositories based on filters like stars, language, and contributors.

Step 5: Clean and analyze data to calculate statistics and trends.

Step 6: Use `matplotlib` and `seaborn` to create visual representations of the data.

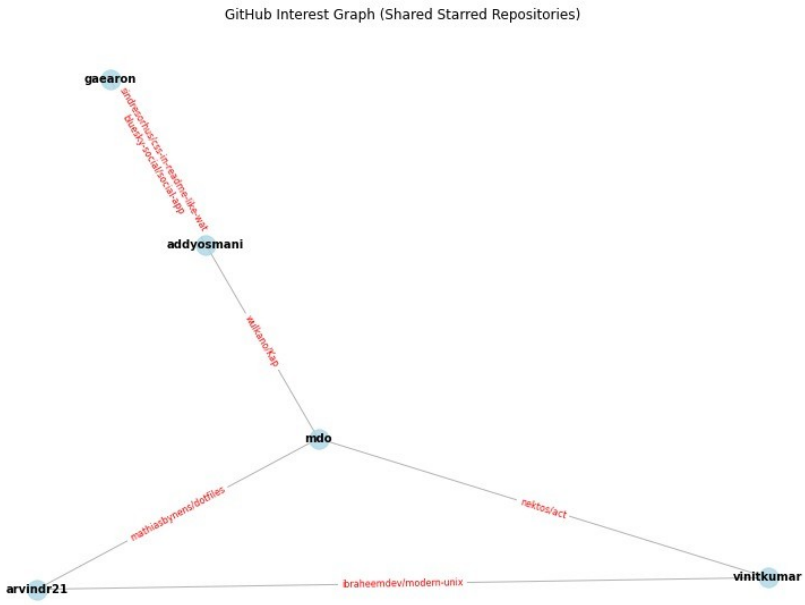
Step 7: Analyze visualizations to identify popular repositories and trends.

CODING:

```
import requests
import networkx as nx
import matplotlib.pyplot as plt

# Sample users
sample_users = ["arvindr21", "vinitkumar", "gaearon", "addyosmani", "mdo"]

# GitHub token (replace with your token)
GITHUB_TOKEN = "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"
headers = {
"Authorization": f"Bearer {GITHUB_TOKEN}",
"Accept": "application/vnd.github.v3+json",
"User-Agent": "InterestGraph"
}

def get_starred_repos(username):
"""Fetch starred repositories of a user."""
url = f"https://api.github.com/users/{username}/starred?per_page=100"
response = requests.get(url, headers=headers)
if response.status_code == 200:
return [repo["full_name"] for repo in response.json()]
else:
print(f"Failed to fetch data for {username}: {response.status_code}")
```

```
return []

# Fetch starred repos for each user
user_repos = {user: get_starred_repos(user) for user in sample_users}

# Create a graph
G = nx.Graph()

# Add nodes (users)
for user in user_repos:
G.add_node(user)

# Add edges with shared repositories
edge_labels = {}

 # Dictionary to store edge labels
for user1 in user_repos:
for user2 in user_repos:
if user1 != user2:
common_repos = set(user_repos[user1]) & set(user_repos[user2])
if common_repos:
G.add_edge(user1, user2)
edge_labels[(user1, user2)] = "\n".join(list(common_repos)[:2])

 # Display up to 2 shared repos
# Plotting the graph
plt.figure(figsize=(14, 10))
pos = nx.spring_layout(G, k=1.0, iterations=100) # Adjust for better spacing

# Draw nodes
nx.draw_networkx_nodes(G, pos, node_color='lightblue', alpha=0.8)

# Draw edges
nx.draw_networkx_edges(G, pos, edge_color='gray', alpha=0.6)

# Draw user labels
nx.draw_networkx_labels(G, pos, font_size=10, font_weight='bold')

# Draw edge labels (shared repositories)
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8,
label_pos=0.5,font_color='red')

# Add title and display the graph
plt.title("GitHub Interest Graph (Shared Starred Repositories)")
plt.axis("off") # Hide axes
plt.show()
```

OUTPUT:



RESULT:

Thus, the program has been executed successfully and output has been verified.