Twitter Sentiment Analysis WebApp Using Flask

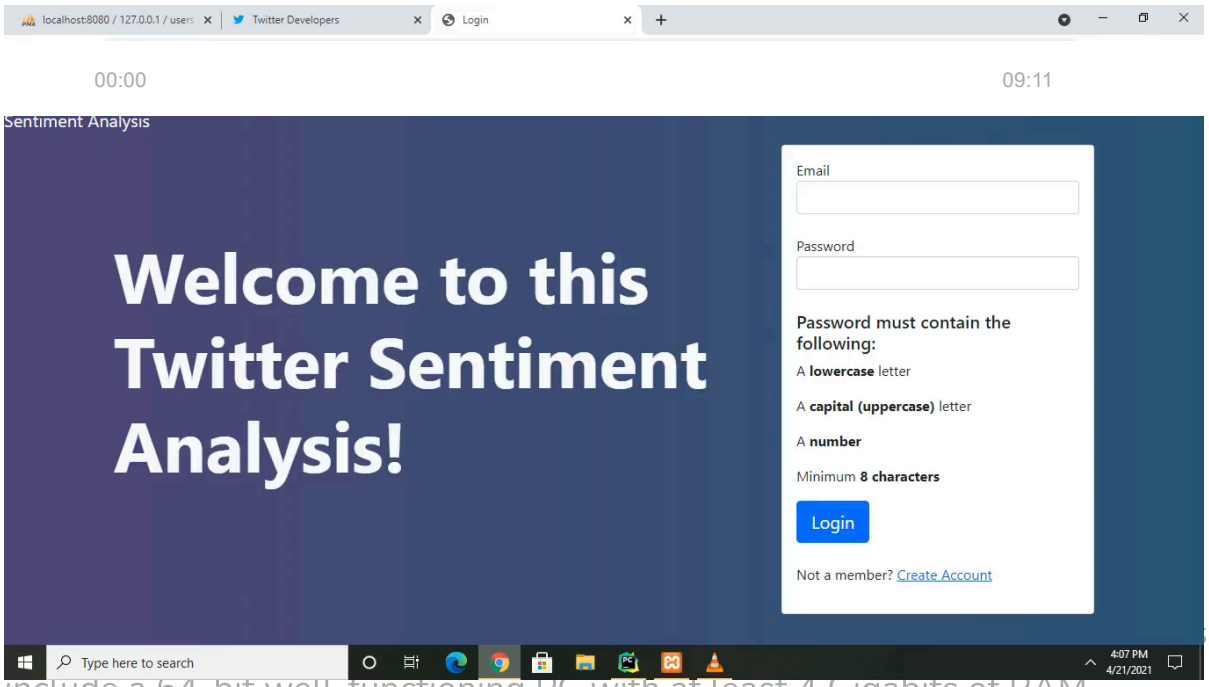
Last Updated : 13 Jun, 2022

This is a web app made using Python and Flask Framework. It has a registration system and a dashboard. Users can enter keywords to retrieve live Twitter text based on the keyword, and analyze it for customer feelings and sentiments. This data can be visualized in a graph. This project, in particular, mines data using a popular “Tweepy” API. Tweepy API connects to Twitter in real-time and gathers metadata along with the text from the Twitter platform.

Purpose:

- To help companies study the customer sentiment around a particular product.
- To help system users analyze a huge amount of data, quickly and efficiently.
- To segregate customer sentiment on a scale of -1 to 1, where -1 represents a strong negative sentimentality towards the keyword(s), and 1 represents a strongly positive reaction.
- To visualize the collected data clearly and effectively.

Detailed Project Implementation Video:



include a 64-bit well-functioning PC with at least 4 Gigabits of RAM.

SOFTWARE:

- Python- Python is an interpreted, high level, and general-purpose programming language. Python promotes code manageability and readability, making it one of the top applications for working with Machine Learning.
- Flask web framework(works with python)- Flask is a web framework. This means flask provides us with tools, libraries, and technologies that will allow me to build a web application and web pages. Flask is a back-end micro-framework, and it makes data handling clean and simple.
- Tweepy(Twitter API for Python)- Tweepy is an open-source Python package that gives you a very convenient way to access the Twitter API with Python. Tweepy includes a set of classes and methods that represent Twitter's models and API endpoints, and it transparently handles various implementation details, such as Data encoding and decoding.
- Apache SQL on Xampp server for windows- SQL server runs on phpMyAdmin inside the localhost. This database is used for storing, verifying, and retrieving a user's login credentials.
- Bootstrap- The UI is enhanced with the help of Bootstrap, which helps to build modern, intuitive, and responsive web pages.
- JQuery- JQuery is used for sending and retrieving data back and forth from the Tweepy API, and displaying the results on the webpage.

- HTML/CSS- HTML and CSS is the base for the website front-end design.

Required Skillset to Build the Project:

Flask Templates Jinja2 Flask-REST API Python SQLAlchemy Flask Bcrypt Flask Cookies Json Postman

- Programming skills
- Python- Advanced.
- HTML/CSS/JQUERY/BOOTSTRAP- Basic to moderate.
- SQL/DBMS- Basic to moderate
- Debugging/Deployment – Moderate
- Ability to work with API's.

Implement the Project:

Follow the below steps to implement the project:

Step 1: Download and Install Xampp Server on your system.

Step 2: Start Apache Xampp and MySQL. Create a database in MySQL with 3 columns (username, email id, and password).

Step 3: Download and Install PyCharm. Click on Create-> New Project. Give a name to your project.

Step 4: Type the following code in Pycharm. For detailed project files and hierarchy, refer to this GitHub repository.

The main.py file:

In this file, we first initialize our project. We establish a connection to our SQL database using the 'conn' object.

We set a user cookie variable, which checks if the user has logged in, before redirecting him/her to the home page. This script also handles the user input on the login/register pages.

Python3

```
from flask import Flask, render_template, request, redirect, session
import mysql.connector
```

```
from sentiments import second
import os


app = Flask(__name__)

# initializing the user cookie
app.secret_key = os.urandom(24)

# blueprint to call the second python file in the project.
app.register_blueprint(second)

# establishing a connection with mysql database made in xampp
try:
    conn = mysql.connector.connect(
        host="localhost", user="root", password="", database="users")
    cursor = conn.cursor()
except:
    print("An exception occurred")

# call the login template when the url is http://localhost:5000/
@app.route('/')
def login():
    return render_template('login.html')

# call the register template when the url is http://localhost:5000/register
@app.route('/register')
def register():
    return render_template('register.html')


@app.route('/home')
def home():
    if 'user_id' in session:
        return render_template('home.html')
    else:
        return redirect('/')


@app.route('/login_validation', methods=['POST'])
def login_validation():
    email = request.form.get('email')
    password = request.form.get('password')

    cursor.execute(
        """SELECT * from `users` WHERE `email` LIKE '{}' AND `password` LIKE '{}''
    users = cursor.fetchall()
    # check if a user has already logged in
    if len(users) > 0:
```

```
        session['user_id'] = users[0][0]
        return redirect('/home')
    else:
        return redirect('/login')


@app.route('/add_user', methods=['POST'])
def add_user():

  # get user login data and pass the data to database
    name = request.form.get('uname')
    email = request.form.get('uemail')
    password = request.form.get('upassword')
    cursor.execute("""INSERT INTO `users` (`name`,`email`,`password`) VALUES ('{}'
        name, email, password))
    conn.commit()
    cursor.execute(
        """SELECT * from `users` WHERE `email` LIKE '{}'""".format(email))
    myuser = cursor.fetchall()
    session['user_id'] = myuser[0][0]
    return redirect('/home')


@app.route('/logout')
def logout():
  # close the session
    session.pop('user_id')
    return redirect('/')


if __name__ == "__main__":
    app.run(debug=True)
```

The sentiments.py

This is the second python file in our project, which was registered as a blueprint in our “main.py” file.

This code takes the user input from the HTML page, which specifies which keyword is to be searched, and the number of tweets to be looked at. It then connects to the Tweepy API, which retrieves the tweet text. This text is cleaned and then categorized into different sentiments(-1 is an angry/sad emotion, 1 is a happy emotion).

Based on this, a pie chart is created and saved as an image. This image is later called in other HTML pages.

Python3

```
from flask import Blueprint, render_template, request
import matplotlib.pyplot as plt
import os

import tweepy
import csv
import re
from textblob import TextBlob
import matplotlib
matplotlib.use('agg')

# register this file as a blueprint
second = Blueprint("second", __name__, static_folder="static",
                   template_folder="template")

# render page when url is called
@second.route("/sentiment_analyzer")
def sentiment_analyzer():
    return render_template("sentiment_analyzer.html")


# class with main logic
class SentimentAnalysis:

    def __init__(self):
        self.tweets = []
        self.tweetText = []

    # This function first connects to the Tweepy API using API keys
    def DownloadData(self, keyword, tweets):

        # authenticating
        consumerKey = '//get from Tweepy'
        consumerSecret = '//get from Tweepy'
        accessToken = '//insert your access token here'
        accessTokenSecret = '//Tweepy AccessToken secret here'
        auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
        auth.set_access_token(accessToken, accessTokenSecret)
        api = tweepy.API(auth, wait_on_rate_limit=True)

        # input for term to be searched and how many tweets to search
        # searchTerm = input("Enter Keyword/Tag to search about: ")
        # NoOfTerms = int(input("Enter how many tweets to search: "))
```

```
tweets = int(tweets)

# searching for tweets
self.tweets = tweepy.Cursor(
    api.search, q=keyword, lang="en").items(tweets)

# Open/create a file to append data to
csvFile = open('result.csv', 'a')

# Use csv writer
csvWriter = csv.writer(csvFile)

# creating some variables to store info
polarity = 0
positive = 0
wpositive = 0
spositive = 0
negative = 0
wnegative = 0
snegative = 0
neutral = 0

# iterating through tweets fetched
for tweet in self.tweets:

    # Append to temp so that we can store in csv later. I use encode UTF-8
    self.tweetText.append(self.cleanTweet(tweet.text).encode('utf-8'))

    # print (tweet.text.translate(non_bmp_map))    #print tweet's text
    analysis = TextBlob(tweet.text)

    # print(analysis.sentiment)  # print tweet's polarity
    # adding up polarities to find the average later
    polarity += analysis.sentiment.polarity

    # adding reaction of how people are reacting to find average later
    if (analysis.sentiment.polarity == 0):
        neutral += 1
    elif (analysis.sentiment.polarity > 0 and analysis.sentiment.polarity
        wpositive += 1
    elif (analysis.sentiment.polarity > 0.3 and analysis.sentiment.polari
        positive += 1
    elif (analysis.sentiment.polarity > 0.6 and analysis.sentiment.polari
        spositive += 1
    elif (analysis.sentiment.polarity > -0.3 and analysis.sentiment.polar
        wnegative += 1
    elif (analysis.sentiment.polarity > -0.6 and analysis.sentiment.polar
        negative += 1
    elif (analysis.sentiment.polarity > -1 and analysis.sentiment.polarit
```

```
                snegative += 1

            # Write to csv and close csv file
            csvWriter.writerow(self.tweetText)
            csvFile.close()

            # finding average of how people are reacting
            positive = self.percentage(positive, tweets)
            wpositive = self.percentage(wpositive, tweets)
            spositive = self.percentage(spositive, tweets)
            negative = self.percentage(negative, tweets)
            wnegative = self.percentage(wnegative, tweets)
            snegative = self.percentage(snegative, tweets)
            neutral = self.percentage(neutral, tweets)

            # finding average reaction
            polarity = polarity / tweets

            # printing out data
            #  print("How people are reacting on " + keyword + " by analyzing " + str(
            #  print()
            #  print("General Report: ")

            if (polarity == 0):
                htmlpolarity = "Neutral"

            # print("Neutral")
            elif (polarity > 0 and polarity <= 0.3):
                htmlpolarity = "Weakly Positive"
                # print("Weakly Positive")
            elif (polarity > 0.3 and polarity <= 0.6):
                htmlpolarity = "Positive"
            elif (polarity > 0.6 and polarity <= 1):
                htmlpolarity = "Strongly Positive"
            elif (polarity > -0.3 and polarity <= 0):
                htmlpolarity = "Weakly Negative"
            elif (polarity > -0.6 and polarity <= -0.3):
                htmlpolarity = "Negative"
            elif (polarity > -1 and polarity <= -0.6):
                htmlpolarity = "strongly Negative"

            self.plotPieChart(positive, wpositive, spositive, negative,
                              wnegative, snegative, neutral, keyword, tweets)
            print(polarity, htmlpolarity)
            return polarity, htmlpolarity, positive, wpositive, spositive, negative,

        def cleanTweet(self, tweet):
            # Remove Links, Special Characters etc from tweet
            return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w +:\ / \ /
```

```
    # function to calculate percentage
    def percentage(self, part, whole):
        temp = 100 * float(part) / float(whole)
        return format(temp, '.2f')

    # function which sets and plots the pie chart. The chart is saved in an img fi
    # The previous image is overwritten. This image is called in the html page.

    def plotPieChart(self, positive, wpositive, spositive, negative, wnegative, sn
        fig = plt.figure()
        labels = ['Positive [' + str(positive) + '%]', 'Weakly Positive [' + str(
                  'Strongly Positive [' + str(spositive) +
                  '%]', 'Neutral [' + str(neutral) + '%]',
                  'Negative [' + str(negative) +
                  '%]', 'Weakly Negative [' + str(wnegative) + '%]',
                  'Strongly Negative [' + str(snegative) + '%]']
        sizes = [positive, wpositive, spositive,
                 neutral, negative, wnegative, snegative]
        colors = ['yellowgreen', 'lightgreen', 'darkgreen',
                  'gold', 'red', 'lightsalmon', 'darkred']
        patches, texts = plt.pie(sizes, colors=colors, startangle=90)
        plt.legend(patches, labels, loc="best")
        plt.axis('equal')
        plt.tight_layout()
        strFile = r"C:\Users\LENOVO\PycharmProjects\SentimentAnalysis\static\image
        if os.path.isfile(strFile):
            os.remove(strFile)  # Opt.: os.system("rm "+strFile)
        plt.savefig(strFile)
        plt.show()


@second.route('/sentiment_logic', methods=['POST', 'GET'])
def sentiment_logic():

  # get user input of keyword to search and number of tweets from html form.
    keyword = request.form.get('keyword')
    tweets = request.form.get('tweets')
    sa = SentimentAnalysis()

    # set variables which can be used in the jinja supported html page
    polarity, htmlpolarity, positive, wpositive, spositive, negative, wnegative, s
        keyword, tweets)
    return render_template('sentiment_analyzer.html', polarity=polarity, htmlpolar
                           negative=negative, wnegative=wnegative, snegative=snega


@second.route('/visualize')
def visualize():
```

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

The SentimentAnalyzer.py template. (The template that renders results for the main logic of the program)

This template asks the users to enter a topic/word of their interest, and the number of tweets based on that topic that users would like to analyze.

Please note that Twitter has daily and hourly rate limits, which cannot be exceeded.

This form passes the data to second.py file, which calculates the output and sets “jinja” variables. (Jinja allows passing values between python and HTML).

These variables then display the output to users.

HTML

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <!-- Required meta tags -->
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename=
      <!-- Bootstrap CSS -->
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boot
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.j
      <script src="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-
      <title>Home</title>
   </head>
   <body class="bg-nav">
      <!-- code for the navigation header bar-->
      <header style="height: 80px" class="navbar navbar-dark sticky-top bg-nav fl
         <a class="navbar-brand col-md-3 col-lg-2 me-0 px-3" style="font-size: 30
         <button class="navbar-toggler position-absolute d-md-none collapsed" typ
         <span class="navbar-toggler-icon"></span>
         </button>
         <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">
            <a class="me-3 py-2 text-decoration-none text-light" style="font-size
            <a class="me-3 py-2 text-light text-decoration-none" style="font-size
            <a class="me-3 py-2 text-light text-decoration-none" style="font-size
            <a class="py-2 text-light text-decoration-none" style="font-size: 20p
         </nav>
         <ul class="navbar-nav px-3">
```

```
                <li class="nav-item text-nowrap">
                    <a class="nav-link text-dark card" style="font-size: 20px; padding
                </li>
            </ul>
        </header>
        <div class="container">
            <div class="row">
                <div class="card mt-100" style="margin-top: 50px">
                    <div class="card-body">
                        <form method="post" action="sentiment_logic">
                            <label> Enter your search keyword </label><br>
                            <input type="text" class="form-control" name="keyword"> <br>
                            <label> Enter your number of tweets to analyze </label><br>
                            <input type="number" class="form-control" name="tweets"><br>
                            <input type="submit" class="btn btn-primary btn-block btn-lg
                        </form>
                        <br>



<p> Need help? <a href="/register"> Click here </a></p>



                    </div>
                </div>
            </div>
        </div>
        <!-- Optional JavaScript; choose one of the two! -->
        <!--output values-->
        <br>
        <br>
        <br>
        <h1 style="color: black">--------------------------------------------------
        <div style="text-align: center;">
            <div >
                {% if polarity %}
                <h3 style="color: white; text-align: center;font-size:30px; border-rad
                {% endif %}
            </div>
        </div>
        <!--parent div for reports-->
        <div class="row">
            <div>
                <!--General sentiment report-->
                <div class="row">
                    <div class="mt-100">
                        <h1 style="color: white; text-align: center;font-size:50px">Gene
                        <div class="alert alert-primary" role="alert" style="height:70p
```

```
                        {% if polarity %}
                        <h1 style="text-align: center;font-size:30px"> The Average Se
                        {%endif%}
                     </div>
                  </div>
               </div>
               <!--end of general report-->
               <!--start of polarity value-->
               <div class="row">
                  <div class="mt-100">
                     <h1 style="color: white; text-align: center;font-size:50px">Sent
                     <div class="alert alert-primary" role="alert" style="height:70p
                        {% if polarity %}
                        <h1 style="text-align: center;font-size:30px"> The sentiment
                        {%endif%}
                     </div>
                  </div>
               </div>
               <!--end of polarity value-->
            </div>
            <!-- end of parent div for reports-->
         </div>
         <div style="margin-top: 50px">
            <h1 style="color: white; text-align: center;font-size:50px">Detailed Repo
            <div class="alert alert-primary" role="alert" style="height:400px">
               {% if polarity %}
               <h2 class="report-text"> {{spositive}} "% people thought it was strong
               <h2 class="report-text"> {{positive}} "% people thought it was positiv
               <h2 class="report-text"> {{wpositive}} "% people thought it was weakly
               <h2 class="report-text"> {{neutral}} "% people thought it was neutral'
               <h2 class="report-text"> {{negative}} "% people thought it was negativ
               <h2 class="report-text"> {{wnegative}} "% people thought it was weakly
               <h2 class="report-text"> {{snegative}} "% people thought it was strong
               {%endif%}
            </div>
         </div>
         <!--end of report-->
         <a href="visualize" class="btn btn-primary btn-block btn-lg" style="backgro
         <br>
         <br>
         <!-- Option 1: Bootstrap Bundle with Popper -->
         <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/boot
         <!-- Option 2: Separate Popper and Bootstrap JS -->
         <!--
            <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/p
            <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/b
            -->
      </body>
   </html>
```

The Login template:

HTML code for the page that displays the login form. The user enters a username and password, which has validations. Once the data is validated, these variables are passed to the main.py file, which connects to the SQL database to check if the login is valid.

HTML

```
<!doctype html>
<html lang="en">
   <head>
      <!-- Required meta tags -->
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename=
      <!-- Bootstrap CSS -->
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boot
      <!-- Latest compiled and minified CSS -->
      <link rel="stylesheet" href=
         "https://maxcdn.bootstrapcdn.com/bootstrap/
         4.0.0/css/bootstrap.min.css">
      <!-- jQuery library -->
      <script src=
         "https://ajax.googleapis.com/ajax/libs/
         jquery/3.3.1/jquery.min.js"></script>
      <script src=
         "https://cdnjs.cloudflare.com/ajax/libs/
         popper.js/1.12.9/umd/popper.min.js"></script>
      <title>Login</title>
   </head>
   <!-- navigation bar design -->
   <body class="bg-nav">
      <nav class="navbar">
         <a href="" class="navbar-brand text-light">Sentiment Analysis</a>
      </nav>
      <!-- login form outer design -->
      <div class="container">
         <div class="row">
            <div class="col-md-8">
               <h1 class="text-light display-4 mt-100" style="font-size: 80px; fo
            </div>
```

```
                <div class="col-md-4">
                    <div class="card mt-100">
                        <div class="card-body">
                            <!-- login form logic -->
                            <form method="post" action="login_validation">
                                <div class="form_group">
                                    <label> Email </label><br>
                                    <input type="email" class="form-control" name="email"
                                    <small id="emailvalid" class="form-text
                                        text-muted invalid-feedback">
                                    Your email must be a valid email
                                    </small>
                                </div>
                                <div class="form_group">
                                    <label> Password </label><br>
                                    <input type="password" class="form-control" name="pass
                                    <h5 id="passcheck" style="color: red;">
                                        **Please Fill the password
                                    </h5>
                                </div>
                                <input type="submit" class="btn btn-primary btn-block btn
                            </form>
                            <br>


<p> Not a member? <a href="/register"> Create Account</a></p>


                        </div>
                    </div>
                </div>
            </div>
        </div>
        <script src="/static/app.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/boot
    </body>
</html>
```

The Registration template:

HTML code for the page that displays the registration form. The user enters a username and password, which has validations. Additionally, the user's name

becomes a value for the cookie that is set in the main.py file.

Once the data is validated, these variables are passed to the main.py file, which connects to the SQL database to check if the login is valid.

HTML

```
<!doctype html>
<html lang="en">
   <head>
      <!-- Required meta tags -->
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename=
      <!-- Bootstrap CSS -->
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boot
      <title>Registration</title>
   </head>
   <body class="bg-nav">
      <nav class="navbar">
         <a href="" class="navbar-brand text-light">Sentiment Analysis App</a>
      </nav>
      <div class="container">
         <div class="row">
            <div class="col-md-8">
               <h1 class="text-light display-4 mt-100" style="font-size: 80px; fo
            </div>
            <div class="col-md-4">
               <div class="card mt-100">
                  <div class="card-body">
                     <form method="post" action="/add_user">
                        <label> Name </label><br>
                        <input type="text" class="form-control" name="uname"> <br
                        <label> Email </label><br>
                        <input type="email" class="form-control" name="uemail"> <
                        <label> Password </label><br>
                        <input type="password" class="form-control" name="upasswo
                        <input type="submit" class="btn btn-primary btn-block btn
                     </form>
                     <br>



<p> Already a member? <a href="/"> Login </a></p>



                  </div>
```

```
            </div>
          </div>
        </div>
      </div>
      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/boot
   </body>
</html>
```

The HomePage Template:

HTML home page. The page has 2 main parts

- **Navigation bar-** The navigation bar has links for all other pages, it retrieves the user's name from the cookie and displays "Welcome Sam" [users name]. This bar also has a logout button.
- **Bootstrap carousel-** This carousel is a customized version of the basic carousel which can be found here.

HTML

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <!-- Required meta tags -->
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename=
      <!-- Bootstrap CSS -->
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boot
      <title>Home</title>
   </head>
   <body>
      <!-- code for the navigation header bar-->
      <header style="height: 80px" class="navbar navbar-dark sticky-top bg-nav fl
         <a class="navbar-brand col-md-3 col-lg-2 me-0 px-3" style="font-size: 30
            {% if session['user_id'] %}
            <h3>Welcome {{session['user_id']}}</h3>
            {%endif%}
         </a>
         <button class="navbar-toggler position-absolute d-md-none collapsed" typ
         <span class="navbar-toggler-icon"></span>
         </button>
```

```
      <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">
         <a class="me-3 py-2 text-decoration-none text-light" style="font-size
         <a class="me-3 py-2 text-light text-decoration-none" style="font-size
         <a class="me-3 py-2 text-light text-decoration-none" style="font-size
         <a class="py-2 text-light text-decoration-none" style="font-size: 20p
      </nav>
      <ul class="navbar-nav px-3">
         <li class="nav-item text-nowrap">
            <a class="nav-link text-dark card" style="font-size: 20px; padding
         </li>
      </ul>
   </header>
   <style>
      .carousel-item {
      height: 34rem;
      background: black;
      color: white;
      position: relative;
      background-position: center;
      background-size: cover;
      }
      .carousel-caption{
      position: absolute;
      bottom: 0;
      left: 0;
      right: 0;
      padding-bottom: 10px;
      }
      .overlay-image{
      position: absolute;
      bottom: 0;
      left: 0;
      right: 0;
      padding-bottom: 0px;
      background-position: center;
      background-size: cover;
      opacity: 0.4;
      }
   </style>
   <!--This is the start of the image slider-->
   <div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="caro
      <div class="carousel-indicators">
         <button type="button" data-bs-target="#carouselExampleCaptions" data-
         <button type="button" data-bs-target="#carouselExampleCaptions" data-
         <button type="button" data-bs-target="#carouselExampleCaptions" data-
      </div>
      <div class="carousel-inner">
         <div style="height: 100px">
            <div class="carousel-item active" >
```

```
            <div class="overlay-image">
               <img src="/static/images/slideone.jpg" style="width: 100%">
            </div>
            <div class="carousel-caption" >
               <b>
                  <h1 style="font-size: 100px">Let's get started!</h1>
               </b>
               <br>
               <a href="sentiment_analyzer" class="btn btn-primary btn-bloc
            </div>
         </div>
         <div class="carousel-item">
            <div class="overlay-image">
               <img src="/static/images/slidetwo.jpg" style="width: 100%">
            </div>
            <div class="carousel-caption" >
               <b>
                  <h1 style="font-size: 100px"> How does this work? </h1>
               </b>
               <br>
               <a href="https://www.tweepy.org/" class="btn btn-primary btn-
            </div>
         </div>
         <div class="carousel-item">
            <div class="overlay-image">
               <img src="/static/images/slidefour.jpg" style="width: 100%">
            </div>
            <div class="carousel-caption" style="align-self: start; padding
               <b>
                  <h1 style="font-size: 100px"> WorkFlow</h1>
               </b>
               <br>
               <div class="btn btn-primary btn-block btn-lg" style="backgro
                  font-size:30px;text-align: left; font-family: 'Franklin Gc
                  Enter a keyword of your choice <br>
                  Enter the no of tweets you want to search <br>
                  Voila! See instant Results! <br>
                  Click on Generate to View Detailed Data Visualization.
               </div>
            </div>
         </div>
      </div>
      <button class="carousel-control-prev" type="button" data-bs-target="#
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Previous</span>
      </button>
      <button class="carousel-control-next" type="button" data-bs-target="#
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Next</span>
```

```
            </button>
          </div>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/boot
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.j
        <script src="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-
    </body>
</html>
```

The PieChart Generation template:

This HTML template simply displays an image file saved in our project. This image file is generated in the second.py file code. The image is a Pie Chart, which visually represents the results of sentiment analysis. The image is overwritten every time the code is run.

HTML

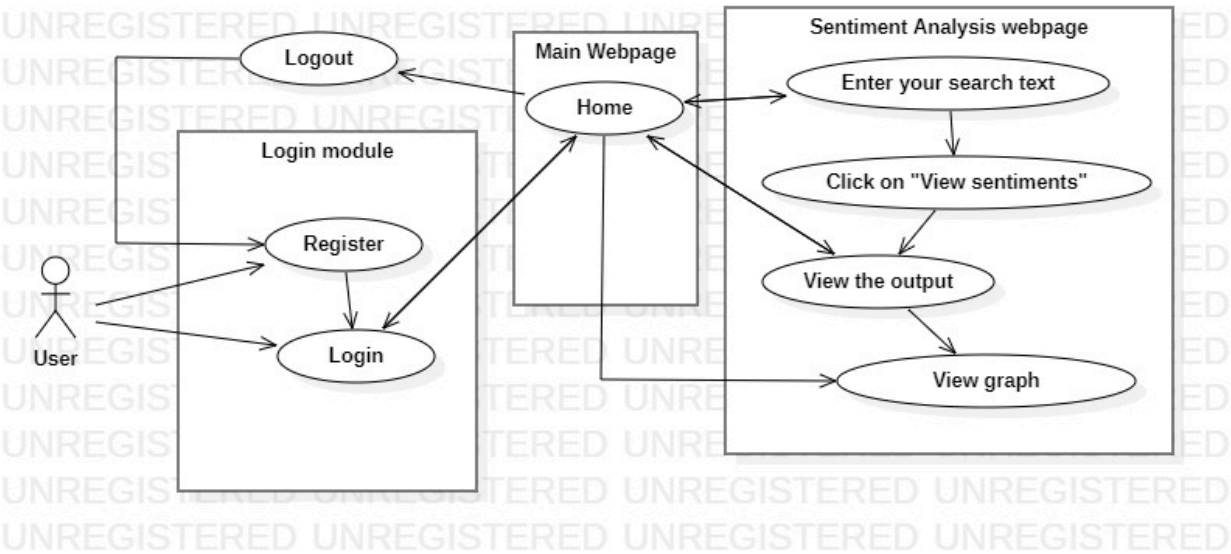
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Data Visualization</title>
        <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename=
        <!-- Bootstrap CSS -->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boot
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.j
    </head>
    </head>
    <body>
        <header style="height: 80px" class="navbar navbar-dark sticky-top bg-nav fl
            <a class="navbar-brand col-md-3 col-lg-2 me-0 px-3" style="font-size: 30
            <button class="navbar-toggler position-absolute d-md-none collapsed" typ
            <span class="navbar-toggler-icon"></span>
            </button>
            <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">
                <a class="me-3 py-2 text-decoration-none text-light" style="font-size
                <a class="me-3 py-2 text-light text-decoration-none" style="font-size
                <a class="me-3 py-2 text-light text-decoration-none" style="font-size
                <a class="py-2 text-light text-decoration-none" style="font-size: 20p
            </nav>
```

```
            <ul class="navbar-nav px-3">
               <li class="nav-item text-nowrap">
                  <a class="nav-link text-dark card" style="font-size: 20px; padding
               </li>
            </ul>
         </header>
         <div class="mt-100" style="padding-top: 100px; text-align: center">
            <img src="/static/images/plot1.png" >
         </div>
         <small style="color: red"> ** Last generated visual. Press CTRL+ Refresh to
         <a href="sentiment_analyzer" class="btn btn-primary btn-block btn-lg" style
         <br>
         <br>
      </body>
   </html>
```

Detailed project Implementation guidelines:

1. The user creates a new account with all the required validations by clicking “Create account” Registration page validations. URL Rewriting is prevented through the use of cookies.
2. After successful registration/login, the user is directed to the home page.
3. Home Page has a bootstrap carousel Slider. At the top left, the cookie is retrieved and displays the logged-in user’s name. This cookie also protects against website hijacking (No user can directly access the /home URL if they are not properly logged in).
4. The Home Page welcomes the user with their registered name, using cookies,
5. The user can navigate using the navigation bar at the top. Click “get started” to go to the main module screen.
6. The user enters a keyword, and the number of tweets to analyze. Both fields need to be filled. The users click analyze. The system connects to Tweepy, fetches the latest tweets, analyzes them, and displays results in the respective fields.

7. Users can click on the generate Visualization button at the bottom of the report. This generates a Pie Chart based on the report.
8. When users click, ‘how this project works”, they are redirected to tweepy documentation.
9. Users can click “logout” to log out and end their session.



Use Case Diagram

Project Application in Real-Life:

Twitter boasts 330 million monthly active users, which allows businesses to reach a broad audience and connect with customers without intermediaries. On the downside, there’s so much information that it’s hard for brands to quickly detect negative social mentions that could harm their business.

That’s why sentiment analysis, which involves monitoring emotions in conversations on social media platforms, has become a key strategy in social media marketing. Listening to how customers feel on Twitter allows companies to understand their audience, keep on top of what’s being said about their brand, and their competitors, and discover new trends in the industry. Companies can take action quickly. The idea can even be extended to ordinary users, where local keywords and hashtags can be used to analyze sentiments from the text on Twitter.

Previous Article

Profile Application using Python Flask and MySQL

Next Article

Flask project - Create a Joke App with PyJokes

Similar Reads

Create a Simple Sentiment Analysis WebApp using Streamlit

In this article, we will see how we can create a simple Sentiment Analysis webApp using with the help of Streamlit Module in Python. Required Modules:...

4 min read

Twitter Sentiment Analysis on Russia-Ukraine War Using Python

In this article, we are going to see how we can perform the Twitter sentiment analysis on the Russia-Ukraine War using Python. Twitter Sentiment Analysis o...

9 min read

Twitter Sentiment Analysis using Python

This article covers the sentiment analysis of any topic by parsing the tweets fetched from Twitter using Python. What is sentiment analysis? Sentiment...

10 min read

OAuth Authentication with Flask - Connect to Google, Twitter, and Facebook

In this article, we are going to build a flask application that will use the OAuth protocol to get user information. First, we need to understand the OAuth protoc...

8 min read

Daily Latest News webapp Using PyWebio in Python

In this article, we are going to create a web app to get gaily News Using PyWebio As we all download an app for receiving daily news but as a python lovers we tr...

5 min read

ToDo webapp using Django

Django is a high-level Python Web framework-based web framework that allows rapid development and clean, pragmatic design. today we will create a todo app...

3 min read

Python | Sentiment Analysis using VADER

Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining...

3 min read

Python - Sentiment Analysis using Affin

Afinn is the simplest yet popular lexicons used for sentiment analysis developed by Finn Årup Nielsen. It contains 3300+ words with a polarity score associated...

2 min read

Sentiment Analysis using JavaScript and API

Sentiment Analysis - Sentiment analysis, also known as opinion mining, is a computational technique used to determine and categorize the emotional tone o...

4 min read

Tweet Sentiment Analysis Using Python Streamlit

This article covers the sentiment analysis of by parsing the tweets fetched from Twitter using the streamlit Python framework. What is Sentiment Analysis?...

4 min read

Article Tags : ProGeek Project Python Flask Projects +3 More

Practice Tags : python
