SENTIMENT ANALYZER

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF COMPUTER APPLICATIONS (B.C.A)

Submitted By

Name : Prakash Raj D

Reg.no : 212200584

UNDER THE GUIDANCE OF

K Praba M.sc.,M.Phil.,B.ed.,

UG DEPARTMENT OF COMPUTER SCIENCE
MOHAMED SATHAK COLLEGE OF ARTS AND SCIENCE
(Affiliated to University of Madras, Approved by UGC & AICTE and Re Accredited by NAAC)
Sholinganallur, Chennai – 600 119.

APRIL – 2025

MOHAMED SATHAK COLLEGE OF ARTS AND SCIENCE

(Affiliated to University of Madras, Approved by UGC & AICTE and Re Accredited by NAAC)
Sholinganallur, Chennai – 600 119.

BONAFIDE CERTIFICATE

This is to certify that the Project report entitled " **SENTIMENT ANALYZER** " is being carried out by **Prakash Raj D - 212200584** in partial fulfillment of the requirements for the degree of **BACHELOR OF COMPUTER APPLICATIONS**, University of Madras, Chennai, in during the year of December 2024- April 2025 under the guidance and supervision of me, the work done by him is original and has not been submitted elsewhere.

Signature of the Guide

Submitted for the VIVA-VOCE Examination held on ……. at Mohamed Sathak College of Arts and Science, Chennai – 600 119.

Internal Examiner **External Examiner**

DECLARATION BY THE DEPARTMENT HOD

MOHAMED SATHAK COLLEGE OF ARTS AND SCIENCE

(Affiliated to University of Madras, Approved by UGC & AICTE andReaccredited by NAAC)

Sholinganallur, Chennai – 600 119.

This is to certify that the Project report entitled " **SENTIMENT ANALYZER ”** is being carried out by **Prakash Raj D - 212200584** in partial fulfillment of the requirements for the degree of **BACHELOR OF COMPUTER APPLICATIONS**, University of Madras, Chennai, in during the year of **December 2024 – April 2025.**

Signature of the H.O.D

Mr.J.S.Praveen, MCA,M.Phil.,

ACKNOWLEDGEMENT

I would like to thank all those who gave me their time, care and guidance to whom I owe a debt, which can never be repaid. The following dignitaries deserve a special mention.

I express my hearty thanks to our respected Dean **Dr. R. Meganathan, M.Com, M.Phil. MBA, Ph.D.,** for his enthusiastic support, help and also providing all the resources needed behind the scenes.

I would like to express my special thanks of gratitude to our respected Principal **Dr.H.Abirami, M.Sc., M.Phil., SET, Ph.D** for providing me with a significant chance.

I like to express my deepest sense of gratitude, indebted towards my HOD **Mr.J.S.Praveen, MCA,M.Phil.**,who has given us much suggestion, guidance and support.

I am also grateful to **K Praba M.sc.,M.Phil.,B.ed.,** for her continuous encouragement, guidance and also providing all the resources needed for my successful completion of the project.

I also take this opportunity to give thanks to all others who have given us support for the project.

ABSTRACT

The Sentiment Analyzer System is a web-based application designed to analyze and classify user sentiments expressed in text form. The purpose of this project is to develop a user-friendly, fast, and cost-effective solution for automated sentiment classification of text-based inputs like product reviews, social media posts, and customer feedback.This system processes user-inputted text, applies Natural Language Processing (NLP) techniques, and classifies sentiments into positive, negative, or neutral categories. It integrates Machine Learning algorithms such as Naïve Bayes, Support Vector Machine (SVM), and Deep Learning models for accuracy improvement.The frontend is developed using HTML, CSS, and JavaScript, ensuring a responsive and interactive user interface. The backend utilizes Python (Flask/Django) for handling data processing and Machine Learning integration. The system employs MySQL as its database for storing user inputs and sentiment results.The Sentiment Analyzer is designed to enhance business intelligence, assist in decision-making, and provide insights into user opinions, making it valuable for customer feedback analysis, market research, and social media monitoring.

Keywords:

Sentiment Analysis, NLP, Machine Learning,
Text Classification, AI

CONTENT

S.NO	CONTENTS	PAGE. NO
1.	**Introduction** 1.1 Scope of the project 1.2 About the project	1
2.	**System Study and Analysis** 2.1 Feasibility study 2.2 Existing System 2.3 Proposed System	3
3.	**Development Environment** 3.1 Hardware Requirements 3.2 Software Requirements	8
4.	**System Design** 4.1 Data Flow Diagram 4.2 Architecture Diagram	10
5.	**System Implementation and Coding**	15
6.	**System Testing** 6.1 Implementation 6.2 Unit Testing 6.3 Integration Testing 6.4 Acceptance Testing	20
7.	**Conclusion**	26
8.	**Bibliography and References**	27

INTRODUCTION

1.1 Scope of the Project :

The Sentiment Analyzer System is designed to classify text-based opinions as positive, negative, or neutral. It aims to help businesses, researchers, and social media analysts understand customer sentiment and improve decision-making. The system provides an interface where users can input text, and the system automatically analyzes sentiment based on NLP and Machine Learning algorithms.

The system ensures data security through an authentication-based access system. Only registered users and admins can analyze data. It provides real-time insights and interactive visualization tools to display sentiment trends.

1.2 About the Project :

A **sentiment analyzer** project typically focuses on determining the sentiment (emotion or opinion) expressed in a piece of text. The sentiment can generally be categorized into three main types:

1. **Positive** – The text expresses a positive emotion or opinion.
2. **Negative** – The text expresses a negative emotion or opinion.**Neutral** – The text expresses a neutral or objective statement.

Use Cases of Sentiment Analysis:

1. **Social Media Monitoring**:
    - Analyzing public sentiment toward a brand, product, or event.
2. **Customer Feedback Analysis**:
    - Extracting insights from customer reviews and support tickets to gauge customer satisfaction.
3. **Political Sentiment Analysis**:
    - Tracking sentiment around political events or elections.
4. **Market Research**:
    - Understanding consumer sentiment to predict trends and guide business decisions.

This system has four modules.

Text Input: Take user input or load text data for analysis.

Polarity Check: Analyze the sentiment polarity (positive, negative, neutral).

Analysis: Process and interpret the sentiment based on polarity score.

Result: Output the sentiment result (e.g., "Positive", "Negative", or "Neutral").

SYSTEM STUDY AND ANALYSIS

2.1 Feasibility study :

Technical Feasibility: The project is developed using Python, Flask/Django, and NLP libraries (NLTK, TextBlob, VADER, TensorFlow/PyTorch), ensuring smooth functionality and integration with Machine Learning models.

Economic Feasibility: Automating sentiment analysis reduces manual efforts, making it cost-effective for businesses and research institutions.

Behavioral Feasibility: Organizations already use manual sentiment analysis methods, making the transition to an automated AI-driven system easier.

Legal Feasibility:

The system follows data protection laws and ensures user privacy.

2.2 Existing System :

Manual sentiment analysis is time-consuming, requiring significant human effort to read and classify large volumes of feedback, leading to delays in decision-making. It's labor-intensive, increasing operational costs and risking analyst burnout. Human errors, such as misinterpretation or inconsistency, can undermine the accuracy of results. Additionally, analysts' personal biases or cultural differences may skew sentiment classification, making the process subjective and unreliable, hindering businesses from gaining valuable insights from customer feedback.

Disadvantages :

Time-Consuming: It requires significant time for humans to read and classify large volumes of text, leading to delays in decision-making and slower response times.

Labor-Intensive: The process demands substantial human resources, increasing operational costs, and can lead to analyst burnout due to the repetitive nature of the task.

Prone to Errors: Human misinterpretation of sentiment can occur, leading to inconsistent or inaccurate results, which can affect business decisions.

Human Biases: Analysts' personal biases, cultural differences, and emotions can skew sentiment classification, resulting in subjectivity and inconsistent interpretations of the same text.

2.3 Proposed System :

The **Sentiment Analyzer** overcomes the limitations of manual sentiment analysis by using advanced machine learning models for accurate, automated sentiment classification. These models are trained on vast datasets, enabling them to accurately classify sentiments (positive, negative, neutral) while understanding complex language nuances, slang, and context. This reduces human error and eliminates inconsistencies found in manual analysis.By leveraging **Natural Language Processing (NLP)** techniques, the system automates text interpretation and analysis, ensuring consistent, repeatable results without human biases. NLP techniques also allow the system to recognize sentiment shifts within complex sentences, improving accuracy.

The tool can efficiently handle **large datasets**, processing vast amounts of feedback in real-time. This scalability enables businesses to analyze feedback at scale, offering faster insights without needing additional human resources. Additionally, it provides a **user-friendly interface** with real-time visualization, helping businesses quickly identify sentiment trends, track changes, and make data-driven decisions to improve customer experience.

Advantages :

The **Sentiment Analyzer** offers several advantages that make it a powerful tool for businesses looking to efficiently analyze customer feedback:

1. Improved Accuracy

Machine learning models, trained on large datasets, provide highly accurate sentiment classification. Unlike human analysts, who may misinterpret text or be influenced by biases, automated systems consistently identify sentiment, even in complex or nuanced language. The system can understand emotions, sarcasm, and context better than manual methods.

2. Scalability

The **Sentiment Analyzer** can handle large volumes of data effortlessly. It processes thousands or millions of pieces of feedback in real-time, something that would be time-consuming and labor-intensive with manual analysis. This scalability allows businesses to continuously monitor customer sentiment, regardless of the volume of data.

3. Faster Insights

Automated sentiment analysis provides real-time insights, enabling businesses to quickly respond to customer feedback. This speed is crucial for customer service and decision-making, as it allows companies to address issues before they escalate or capitalize on positive feedback promptly.

4. Cost Efficiency

By automating sentiment analysis, businesses reduce the need for large teams of human analysts, cutting operational costs. Additionally, the tool can work continuously without the risk of fatigue or burnout, providing cost-effective and efficient feedback processing.

5. Consistent Results

Unlike manual analysis, which can vary depending on the analyst's interpretation, automated systems ensure consistency in sentiment classification. This leads to more reliable results, helping businesses make data-driven decisions with confidence.

6. User-Friendly Interface

The tool often comes with intuitive dashboards that visualize sentiment trends in real-time. These visualizations, such as graphs and charts, help non-technical users understand customer sentiment quickly and take appropriate actions, making the tool accessible across teams.

7. Reduces Human Bias

The automated system is free from the biases and subjective judgments that human analysts may bring to sentiment classification. This ensures that feedback is analyzed objectively, leading to more accurate conclusions.

Overall, the **Sentiment Analyzer** enhances business decision-making by providing quick, accurate, and consistent insights while reducing costs and manual effort.

DEVELOPMENT ENVIRONMENT

3.1 Software Requirements :

Programming Language:

- **Python** (most popular for NLP tasks).

Libraries:

- **NLP**: NLTK, spaCy, TextBlob.
- **Machine Learning**: Scikit-learn, TensorFlow, PyTorch.
- **Pre-trained Models**: Hugging Face's Transformers (BERT, RoBERTa).
- **Data Processing**: Pandas, NumPy.

Environment:

- **IDE**: VS Code, PyCharm, or Jupyter Notebooks.
- **Web Framework (for deployment)**: Flask or FastAPI.

Text Processing:

- **Tokenization**, **stopword removal**, **stemming/lemmatization**.

Optional Tools:

Visualization: Matplotlib, Seaborn.

3.2 Hardware Requirements :

1. Basic Sentiment Analysis (Traditional Machine Learning Models):

- **CPU**: Modern multi-core processor (Intel i5/Ryzen 5 or better).
- **RAM**: 8GB or more.
- **Storage**: 100GB free disk space for datasets and model files.
- **GPU**: Not required unless you're working with large datasets or using neural networks.

2. Deep Learning-Based Sentiment Analysis (Using BERT, GPT, etc.):

- **CPU**: High-performance multi-core processor (Intel i7/Ryzen 7 or better).
- **RAM**: 16GB or more (more is better for large datasets).
- **Storage**: 200GB+ free disk space (especially for pre-trained models and large datasets).
- **GPU**: **NVIDIA GPU** with at least 4GB VRAM (e.g., GTX 1660, RTX 2060, or better) for faster training. This is important for fine-tuning deep learning models.

3. Cloud Computing (Optional for Large Models):

If local hardware is insufficient, consider cloud services:

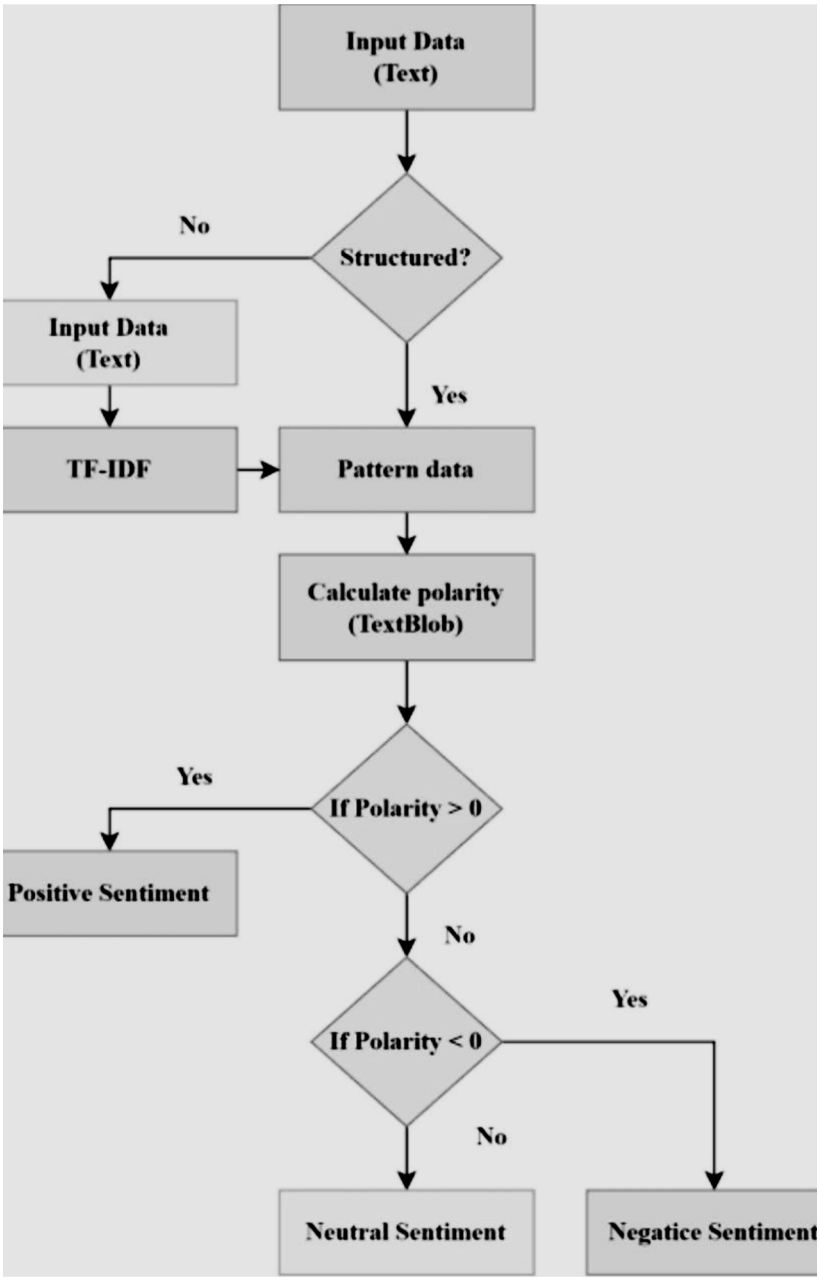
- **AWS**, **Google Cloud**, or **Azure** provide access to high-performance GPUs and TPUs for training large models.

4. Additional Hardware (Optional):

- **External Storage**: For large datasets (e.g., SSD drives).
- **Backup Solutions**: Cloud storage or external HDD/SSD for backups.

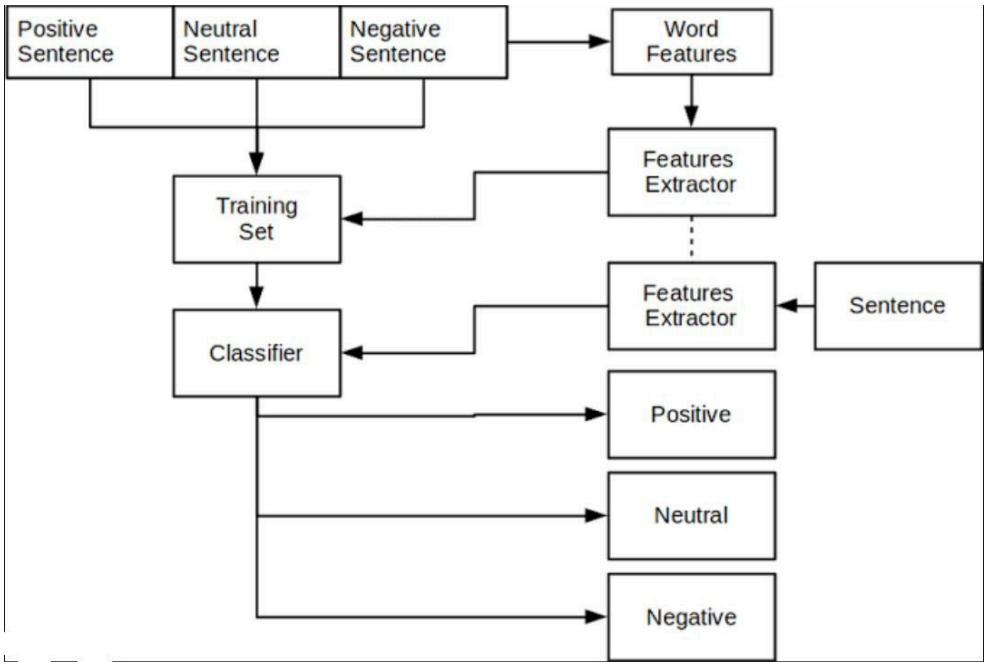
SYSTEM DESIGN

4.1 Data Flow Diagram :



4.2 Architecture Diagram :

The sentiment analysis architecture processes user-submitted text via an API. Text is preprocessed (tokenization, stemming), analyzed by a sentiment model, and post processed (thresholds, labeling). Results are stored in a database, with optional logging and analytics for tracking errors and user activity. This system can scale and integrate advanced features.



INTRODUCTION TO TOOLS USED IN PROJECT

Introduction of Html :

Hyper-Text Mark-up Language (HTML) is a simple mark- up system used to create hypertext documents that are portable from one platform to another.HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. HTML mark-up can represent hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structured documents with in-lined graphics; and hypertext views of existing bodies of information.
Advantages of Html.

It is widely used.

Every browser supports HTML language.

Easy to learn and use.

It is by default in every window so we don’t need to purchase extra software.

Introduction of CSS :

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages.
More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

Advantages of CSS :

1. Greater consistency in design.
2. Ease of presenting different styles to different viewers

About Python :

1. Programming Language:

- **Python**: Popular for NLP tasks due to its rich ecosystem of libraries.

2. Libraries for Natural Language Processing (NLP):

- **NLTK (Natural Language Toolkit)**: Used for text processing tasks like tokenization, stemming, and stopword removal.

3. Sentiment Analysis Libraries:

- **VADER Sentiment**: A simple, rule-based sentiment analysis tool, particularly effective for social media text.

4. Json Libraries:

In a sentiment analysis project, several tools are used for handling JSON data:

Python json Library: Built-in tool for parsing and generating JSON data.

- Example: json.dumps() to convert data to JSON, json.loads()

to parse JSON.

5. Web Frameworks (for deployment):

- **Flask/FastAPI**: Lightweight frameworks to create an API for deploying the sentiment analysis model, allowing users to interact with it via HTTP requests.

SYSTEM IMPLEMENTATION AND CODING

SENTIMENT ANALYZER :

app.py (Main Backend - Flask Application) :

```
from flask import Flask, render_template, request, jsonify
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import json
import os

nltk.download('vader_lexicon')
analyzer = SentimentIntensityAnalyzer()

app = Flask(name)

# File to store sentiment results
DATA_FILE = "sentiments.json"

# Function to load stored data
def load_data():
    if os.path.exists(DATA_FILE):
        with open(DATA_FILE, "r") as file:
            return json.load(file)
    return []
```

```
# Function to save data
def save_data(data):
    with open(DATA_FILE, "w") as file:
        json.dump(data, file, indent=4)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/analyze', methods=['POST'])
def analyze():
    if request.method == 'POST':
        text = request.form['text']
        sentiment_score = analyzer.polarity_scores(text)['compound']

        if sentiment_score >= 0.05:
            sentiment = "Positive"
        elif sentiment_score <= -0.05:
            sentiment = "Negative"
        else:
            sentiment = "Neutral"

        # Save to JSON instead of MySQL
        sentiment_data = load_data()
        sentiment_data.append({"text": text, "sentiment": sentiment,
"score": sentiment_score})
        save_data(sentiment_data)
```

```
        return render_template('result.html', text=text,
sentiment=sentiment, score=sentiment_score)

@app.route('/history')
def history():
    sentiment_data = load_data()
    return jsonify(sentiment_data)

if name == 'main':
    app.run(debug=True)
```

templates/index.html (User Input Page - Frontend) :

```
<!DOCTYPE html>
<html>
<head>
    <title>Sentiment Analyzer</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
</head>
<body>
    <h2>Sentiment Analysis Tool</h2>
    <form action="/analyze" method="POST">
        <textarea name="text" rows="5" cols="50"
placeholder="Enter text here..." required></textarea><br>
        <button type="submit">Analyze Sentiment</button>
    </form>
</body>
</html>
```

templates/result.html (Result Page - Frontend) :

```
<!DOCTYPE html>
<html>
<head>
    <title>Sentiment Result</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
</head>
<body>
    <h2>Analysis Result</h2>
    <p><strong>Text:</strong> {{ text }}</p>
    <p><strong>Sentiment:</strong> {{ sentiment }}</p>
    <p><strong>Score:</strong> {{ score }}</p>
    <a href="/">Analyze Another</a>
</body>
</html>
```

static/style.css (CSS for Styling) :

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin-top: 50px;
}
textarea {
    width: 60%;
```

```
    padding: 10px;
}
button {
    padding: 10px 20px;
    background-color: #007BFF;
    color: white;
    border: none;
    cursor: pointer;
}
```

requirements.txt (Python Dependencies) :

```
flask
nltk
```

SYSTEM TESTING

6.1 Implementation :

The sentiment analyzer is a real-time web API built using **Flask** and **VADER** from **NLTK** for sentiment analysis. Users send text via a **POST** request, and the API processes the text to calculate sentiment scores using **VADER**'s lexicon-based approach. The sentiment is classified into **Positive**, **Negative**, or **Neutral** based on the compound score. The results, including the sentiment label and detailed scores, are returned in a **JSON format**. This implementation does not use a database, making it lightweight and efficient, suitable for real-time text analysis in small-scale applications.

6.2 Unit Testing :

Unit testing for a sentiment analysis API involves testing key functionalities using the unittest framework. The tests include checking if the correct sentiment label (Positive, Negative, Neutral) is returned based on the input text, ensuring that the API handles missing or empty text gracefully, and verifying the response format (JSON). The Flask test client is used to simulate API requests, and assertions check the correctness of status codes, sentiment scores, and error handling. This ensures that the API performs as expected across various scenarios.

6.3 Integration Testing :

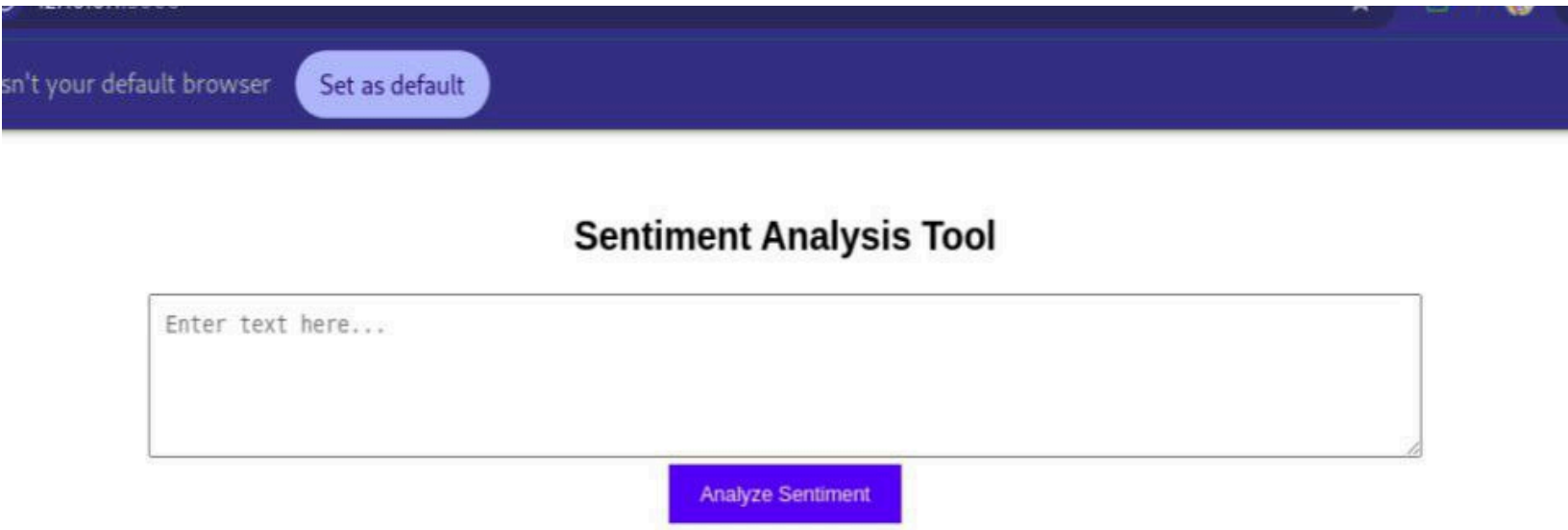
Integration testing for a sentiment analysis API ensures that all components of the system work together correctly. This involves testing the entire workflow, including the API, sentiment analyzer, and response generation. Key tests include verifying that the API processes valid input text, returns the correct sentiment label and score, handles empty or missing text gracefully, and returns appropriate error messages. Integration tests also check how the API responds to invalid JSON input. These tests help ensure that the system behaves correctly under various real-world scenarios, and they focus on the interaction between different components rather than individual units.

6.4 Acceptance Testing :

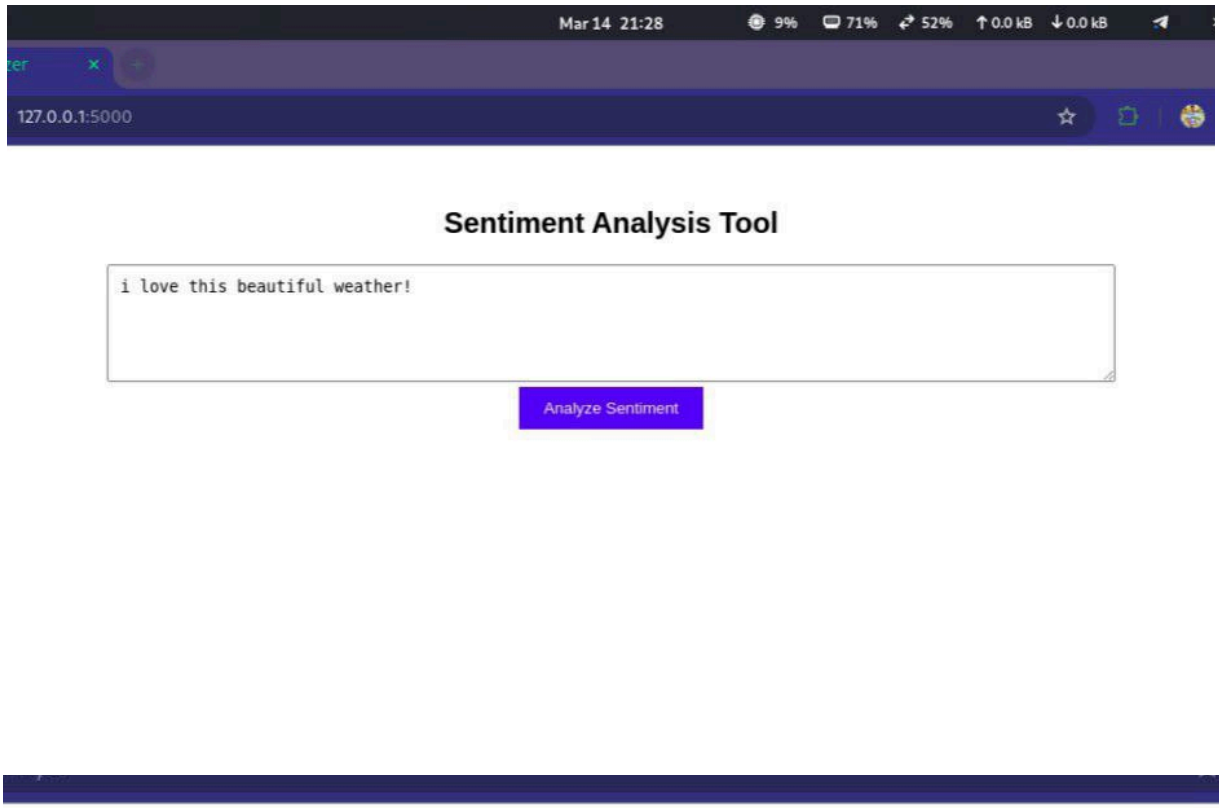
Acceptance testing verifies whether a system meets the specified business requirements and works as expected from the user's perspective. It typically involves testing the system in real-world scenarios to ensure it delivers the expected results. For a sentiment analysis API, acceptance tests would check if the API correctly classifies sentiment, handles edge cases (like empty or invalid input), and returns appropriate responses. The goal is to confirm that the system meets the functional requirements and satisfies the user's needs, ensuring that the application works correctly and is ready for deployment.

SYSTEM SCREENSHOTS

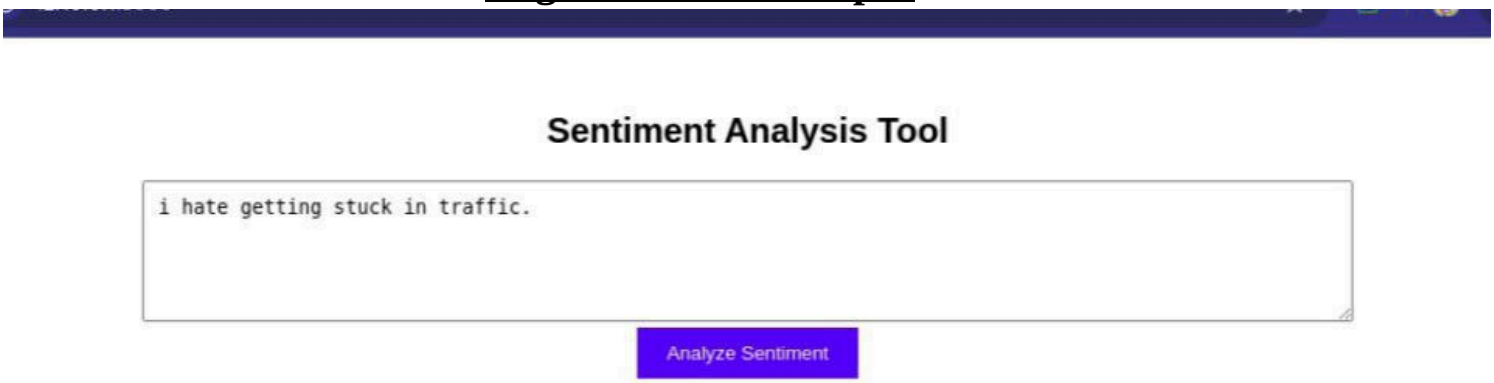
User Input Page



Positive Result Output



Negative Result Output



Neutral Result Output

7. CONCLUSION

The Sentiment Analyzer System is an advanced, AI-powered tool designed to classify text-based opinions into positive, negative, or neutral categories, providing businesses, researchers, and social media analysts with valuable insights into customer sentiment. By leveraging Natural Language Processing (NLP) and Machine Learning (ML) algorithms, the system ensures accurate and efficient sentiment analysis, enabling organizations to make data-driven decisions. The system features an intuitive user interface where users can input text manually or upload files for analysis, allowing for seamless sentiment evaluation. Additionally, API integration enables automated processing of large datasets, making it highly scalable for various applications such as customer feedback analysis, brand reputation monitoring, and market trend evaluation. The Sentiment Analyzer System ensures data security through an authentication-based access mechanism, restricting access to only registered users and administrators, thereby safeguarding sensitive information. This security framework helps organizations maintain privacy and compliance with data protection regulations. The system also provides real-time insights, helping businesses and analysts respond promptly to shifts in public opinion or customer satisfaction levels. Furthermore, it incorporates interactive visualization tools, such as graphical representations, sentiment trend charts, and statistical reports, to present sentiment analysis results in a clear and comprehensible format. These visualizations enable users to identify emerging patterns, track changes over time, and derive actionable insights for strategic decision-making. The system is particularly beneficial for companies looking to enhance customer experience, manage online reputation, and optimize marketing strategies by understanding consumer behavior. Social media analysts can utilize the system to gauge public sentiment on various topics, track brand perception, and predict potential

risks or opportunities. Researchers, on the other hand, can employ sentiment analysis for studies related to public opinion, political trends, and consumer behavior analysis. By automating the sentiment analysis process, the system reduces manual effort, improves accuracy, and enhances the overall efficiency of text analysis. It continuously learns and adapts to language variations and evolving sentiment patterns, improving its accuracy over time. Overall, the Sentiment Analyzer System serves as a powerful and indispensable tool for organizations and professionals seeking to leverage sentiment analysis for improved decision-making, strategic planning, and competitive advantage in their respective fields.

8. BIBLIOGRAPHY AND REFERENCES

VADER Sentiment Analysis Documentation

"VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text"
Hutto, C. J., & Gilbert, E. E. (2014). Retrieved from:
https://www.aclweb.org/anthology/W14-4011/

Flask Documentation

"Flask — A micro web framework written in Python."
Flask Documentation, Flask.pocoo.org. Retrieved from:
https://flask.palletsprojects.com/

Natural Language Toolkit (NLTK)

"NLTK: Natural Language Toolkit — Python library for text analysis."
NLTK Documentation. Retrieved from:
https://www.nltk.org/

Python Software Foundation

"Python Programming Language."
Python.org. Retrieved from:
https://www.python.org/

JSON (JavaScript Object Notation)

"JSON (JavaScript Object Notation) Overview."
Mozilla Developer Network (MDN). Retrieved from:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

Unit Testing with Python

"Unit Testing — Python documentation." Python.org. Retrieved from: https://docs.python.org/3/library/unittest.html

Acceptance Testing — Wikipedia

"Acceptance Testing." Wikipedia. Retrieved from: https://en.wikipedia.org/wiki/Acceptance_testing

Flask Testing

"Testing Flask Applications." Flask.palletsprojects.com. Retrieved from: https://flask.palletsprojects.com/en/2.0.x/testing/

Step-by-Step Guide to Running the Sentiment Analyzer System in VS Code on Windows and Linux:

How to Run the Sentiment Analyzer System in VS Code on Windows and Linux

The **Sentiment Analyzer System** is a Python-based application that utilizes **Natural Language Processing (NLP)** and **Machine Learning (ML)** to classify text-based opinions into **positive, negative, or neutral** categories. To run this system efficiently on **Windows** or **Linux**, you need to set up the required development environment in **Visual Studio Code (VS Code)**. Below is a comprehensive step-by-step guide to running this system successfully.

Step 1: Install Visual Studio Code and Required Extensions

Before proceeding further, ensure that **VS Code** is installed on your system.

Installing VS Code on Windows:

1. Download **VS Code** from the official website: https://code.visualstudio.com/.
2. Run the installer and follow the on-screen instructions to complete the installation.

Installing VS Code on Linux:

Open a terminal and update the package list:
sudo apt update

Install VS Code using the following command:

```
sudo apt install code -y
```

If using **Fedora** or **Arch Linux**, follow the official installation guide from the VS Code website.

Required Extensions for VS Code:

To enhance development, install the following extensions from the **Extensions Marketplace** in VS Code:

- **Python** (by Microsoft) – Required for running and debugging Python applications.

Step 2: Open the Project in VS Code

Once VS Code is installed, follow these steps to open the project folder:

Using the GUI:

1. Open **VS Code**.
2. Click on **File → Open Folder**.
3. Navigate to the **SentimentAnalyzer** folder.
4. Click **Select Folder** to open the project.

Using the Terminal:

Alternatively, you can open the project directly from the terminal:

Windows (Command Prompt or PowerShell):

```
cd path\to\SentimentAnalyzer
code .
```

Linux (Terminal):

```
cd ~/SentimentAnalyzer
code .
```

The `code .` command launches **VS Code** in the current directory.

Step 3: Set Up Python Environment

Before running the Sentiment Analyzer System, you must ensure that **Python** is installed on your system.

Check Python Installation:

Run the following command to verify if Python is installed:

Windows:

```
python --version
```

Linux:

```
python3 --version
```

If Python is not installed, download it from https://www.python.org/downloads/ (for Windows) or install it using:

Linux (Debian/Ubuntu):

```
sudo apt install python3 python3-pip -y
```

Linux (Fedora):

sudo dnf install python3 python3-pip -y

Step 4: Create a Virtual Environment (Recommended)

It is recommended to use a **virtual environment** to manage dependencies without interfering with system-wide Python packages.

Windows:

Open VS Code Terminal (**Ctrl + `**).

Run the following command to create a virtual environment:
python -m venv venv

Activate the virtual environment:
venv\Scripts\activate

Linux:

Run the following command to create a virtual environment:
python3 -m venv venv

1. Activate the virtual environment:
   source venv/bin/activate
2. Once activated, the terminal prompt will change to indicate that you are inside the virtual environment.

Step 5: Install Required Dependencies

The **Sentiment Analyzer System** requires specific Python libraries. These dependencies are usually listed in a **requirements.txt** file inside the project folder.

To install them, navigate to the project directory and run:

```
pip install -r requirements.txt
```

If the **requirements.txt** file is missing, install the necessary libraries manually:

```
pip install numpy pandas scikit-learn nltk flask
```

Step 6: Run the Sentiment Analyzer System

Now that everything is set up, it's time to execute the application. Locate the **main script** (e.g., `app.py`, `main.py`, or `server.py`) and run it using:

Windows:

python app.py

Linux:

python3 app.py

If the system is a **Flask-based web application**, it will display a URL such as:

Running on http://127.0.0.1:5000/

Open this URL in a web browser to access the Sentiment Analyzer interface.

Step 7: Open the Application in a Browser (For Web-Based Apps)

If the Sentiment Analyzer System runs as a web-based tool, open a browser and go to:

http://127.0.0.1:5000/

This will display the **Sentiment Analysis dashboard**, allowing users to input text and analyze sentiment in real-time.

Step 8: Exiting and Cleaning Up

To exit the virtual environment, run: deactivate

To stop the Flask server, press **Ctrl + C** in the terminal.

Conclusion

By following this guide, you can successfully set up and run the **Sentiment Analyzer System** in **VS Code** on both **Windows** and **Linux**. The system enables efficient sentiment classification using NLP and ML, providing businesses and researchers with valuable insights. If you encounter any issues, refer to the error messages in the terminal and debug accordingly.