INTERNSHIP PROJECT REPORT

Title: **Sentiment Analysis Web Application using Flask & TextBlob**

Submitted by: **Parth Gupta**

Submitted to: **CODEXINTERN**

Internship Project Report

---

1. Introduction

Sentiment Analysis is a Natural Language Processing (NLP) technique used to determine whether a given text expresses a **positive**, **negative**, or **neutral** emotion. In this project, a web-based application has been developed using **Flask (Python Web Framework)** and **TextBlob** for analyzing user-entered text.

The system processes the input text and displays:

- Sentiment Classification
- Polarity Score
- Subjectivity Score
- Sentiment Emoji
- Intensity Level
- Word and Character Counts

This makes the project useful for educational, analytical, and user-interaction purposes.

---

2. Objective

The main objectives of the project are:

- To develop a **web application** using Flask.
- To perform **sentiment analysis** using the TextBlob library.
- To classify text as **positive**, **negative**, or **neutral**.
- To display **polarity** and **subjectivity** scores.

- To enhance user experience with additional helpful features.

---

3. Technologies Used

Technology	Purpose
Python	Base programming language
Flask	Backend web framework
HTML/CSS	Frontend UI
TextBlob	Sentiment Analysis (NLP)
Jinja2	Template rendering
Pycharm / VS Code	Development environment

---

4. System Requirements

Software Requirements:

- Python 3.8+
- Flask
- TextBlob
- Any web browser
- IDE (VS Code, PyCharm, etc.)

Hardware Requirements:

- Minimum 4 GB RAM
- Working internet (for installing libraries)

---

5. Project Workflow

Below is the flow of the application:

1. User opens the web application.
2. User enters any text into the text-area.

3. Flask receives the text and passes it to TextBlob.
4. TextBlob calculates:
    - Polarity
    - Subjectivity
5. Application classifies the sentiment as:
    - Positive
    - Negative
    - Neutral
6. Additional info such as:
    - Emoji
    - Intensity label
    - Word count
    - Character count
7. All results are displayed beautifully in the browser.

---

6. Folder Structure

SENTIMENT_APP/

│ app.py

│ requirements.txt

└── templates/

    └── index.html

---

7. Code Implementation

7.1 app.py

from flask import Flask, render_template, request

from textblob import TextBlob

```
app = Flask(__name__)


def get_sentiment_label(polarity: float) -> str:
    """
    Polarity range: -1 (very negative) to +1 (very positive)
    Threshold:
     - |polarity| < 0.05  => Neutral
     - polarity >= 0.05   => Positive
     - polarity <= -0.05  => Negative
    """
    if polarity >= 0.05:
        return "Positive"
    elif polarity <= -0.05:
        return "Negative"
    else:
        return "Neutral"


def get_sentiment_emoji(polarity: float) -> str:
    """Return an emoji according to sentiment strength."""
    if polarity >= 0.6:
        return " 😄 "   # strongly positive
    elif polarity >= 0.2:
        return " 🙂 "   # mildly positive
    elif polarity <= -0.6:
        return " 😡 "   # strongly negative
    elif polarity <= -0.2:
        return " 😟 "   # mildly negative
    else:
```

```
        return " 😐 "   # neutral

def get_intensity_label(polarity: float) -> str:
    """More detailed description like 'Strongly Positive' etc."""
    if -0.05 < polarity < 0.05:
        return "Neutral"

    direction = "Positive" if polarity > 0 else "Negative"
    strength = abs(polarity)

    if strength < 0.2:
        prefix = "Slightly"
    elif strength < 0.5:
        prefix = "Moderately"
    else:
        prefix = "Strongly"

    return f"{prefix} {direction}"

@app.route("/", methods=["GET", "POST"])
def index():
    result = None

    if request.method == "POST":
        user_text = request.form.get("user_text", "").strip()

        if user_text:
            blob = TextBlob(user_text)
```

```
        polarity = round(blob.sentiment.polarity, 3)
        subjectivity = round(blob.sentiment.subjectivity, 3)

        label = get_sentiment_label(polarity)
        emoji = get_sentiment_emoji(polarity)
        intensity = get_intensity_label(polarity)

        # extra stats
        words = len(user_text.split())
        chars_no_space = len(user_text.replace(" ", ""))

        result = {
            "text": user_text,
            "polarity": polarity,
            "subjectivity": subjectivity,
            "label": label,
            "emoji": emoji,
            "intensity": intensity,
            "word_count": words,
            "char_count": chars_no_space,
        }

    return render_template("index.html", result=result)

if __name__ == "__main__":
    # Debug mode for development
    app.run(debug=True)
```

7.2 index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sentiment Analysis</title>

  <!-- Google Font -->
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&display=swap" rel="stylesheet">

  <style>
    body {
      margin: 0;
      padding: 0;
      background: linear-gradient(135deg, #74ABE2, #5563DE);
      font-family: 'Poppins', sans-serif;
    }

    .container {
      max-width: 700px;
      margin: 60px auto;
      background: #ffffff;
      padding: 40px 45px;
      border-radius: 18px;
      box-shadow: 0 12px 35px rgba(0,0,0,0.18);
```

```
    animation: fadeIn 0.8s ease-in-out;
}

@keyframes fadeIn {
    from { opacity: 0; transform: translateY(15px); }
    to { opacity: 1; transform: translateY(0); }
}

h1 {
    text-align: center;
    color: #333;
    font-size: 32px;
    font-weight: 600;
    margin-bottom: 20px;
}

textarea {
    width: 100%;
    height: 150px;
    padding: 15px;
    font-size: 15px;
    border-radius: 12px;
    border: 1px solid #ccc;
    resize: vertical;
    outline: none;
    transition: 0.3s;
}
```

```
textarea:focus {
  border-color: #5563DE;
  box-shadow: 0px 0px 8px rgba(85, 99, 222, 0.3);
}

button {
  width: 100%;
  margin-top: 15px;
  padding: 12px;
  border: none;
  border-radius: 12px;
  background: #5563DE;
  color: #fff;
  font-size: 18px;
  cursor: pointer;
  font-weight: 500;
  transition: 0.3s;
}

button:hover {
  background: #3340b5;
  transform: translateY(-2px);
}

.result-box {
  margin-top: 35px;
  padding: 20px;
  border-radius: 12px;
```

```
    background: #eef2ff;
    animation: fadeIn 0.7s ease-in-out;
}

.label-positive { color: #0e9f6e; font-weight: 600; }
.label-negative { color: #e02424; font-weight: 600; }
.label-neutral  { color: #6b7280; font-weight: 600; }

.emoji {
    font-size: 36px;
    margin-left: 8px;
}

.score {
    margin-top: 12px;
    font-size: 15px;
}

.extra-info {
    margin-top: 10px;
    font-size: 14px;
    color: #4b5563;
}

.stats-box {
    margin-top: 15px;
    padding: 12px;
    border-radius: 10px;
```

```
      background: #e0e7ff;
      font-size: 14px;
    }

    .original-text {
      margin-top: 18px;
      padding: 15px;
      font-size: 15px;
      background: #ffffff;
      border-radius: 10px;
      border: 1px solid #ddd;
      white-space: pre-wrap;
      box-shadow: 0 4px 10px rgba(0,0,0,0.05);
    }
  </style>
</head>

<body>
  <div class="container">
    <h1>Sentiment Analysis (TextBlob)</h1>

    <form method="POST">
      <label for="user_text"><strong>Enter your text:</strong></label><br>
      <textarea id="user_text" name="user_text" placeholder="Type
something..."></textarea>
      <button type="submit">Analyze Sentiment</button>
    </form>
```

```
{% if result %}
  <div class="result-box">
    <div>
      <strong>Sentiment:</strong>
      {% if result.label == "Positive" %}
        <span class="label-positive">{{ result.label }}</span>
      {% elif result.label == "Negative" %}
        <span class="label-negative">{{ result.label }}</span>
      {% else %}
        <span class="label-neutral">{{ result.label }}</span>
      {% endif %}

      <!-- Emoji -->
      <span class="emoji">{{ result.emoji }}</span>
    </div>

    <div class="extra-info">
      <strong>Intensity:</strong> {{ result.intensity }}
    </div>

    <div class="score">
      <strong>Polarity:</strong> {{ result.polarity }} <br>
      <strong>Subjectivity:</strong> {{ result.subjectivity }}
    </div>

    <div class="stats-box">
      <strong>Text Statistics:</strong><br>
      Words: {{ result.word_count }}  | 
```

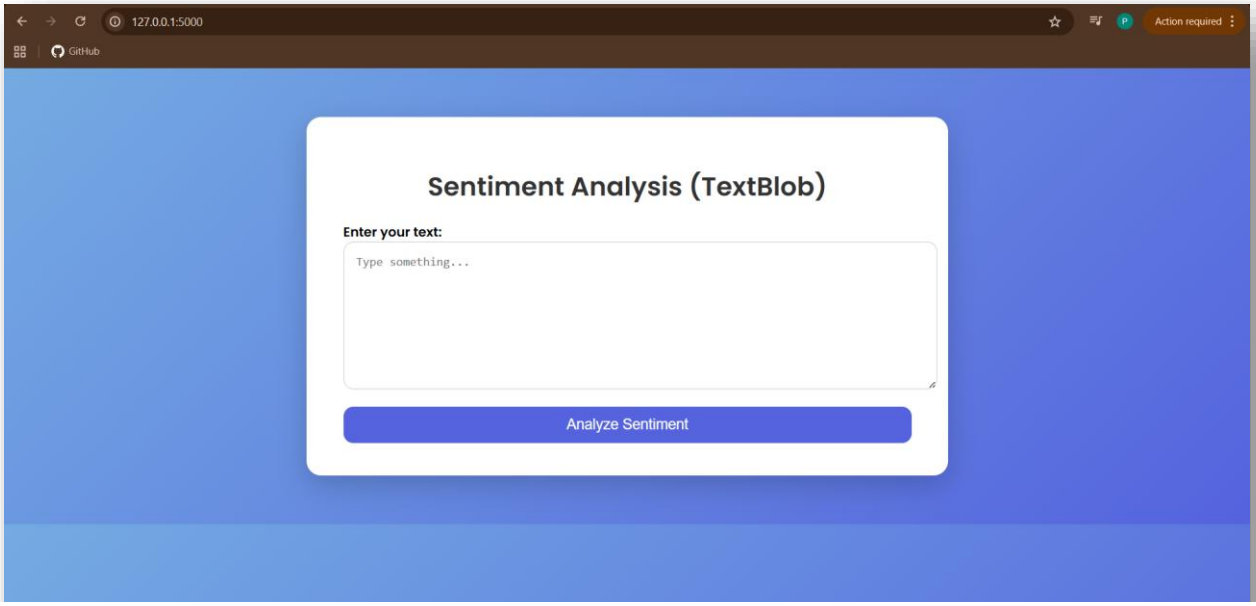
```
            Characters (no spaces): {{ result.char_count }}
          </div>

          <div class="original-text">
            <strong>Input Text:</strong><br>
            {{ result.text }}
          </div>
        </div>
      {% endif %}
    </div>
</body>
</html>
```

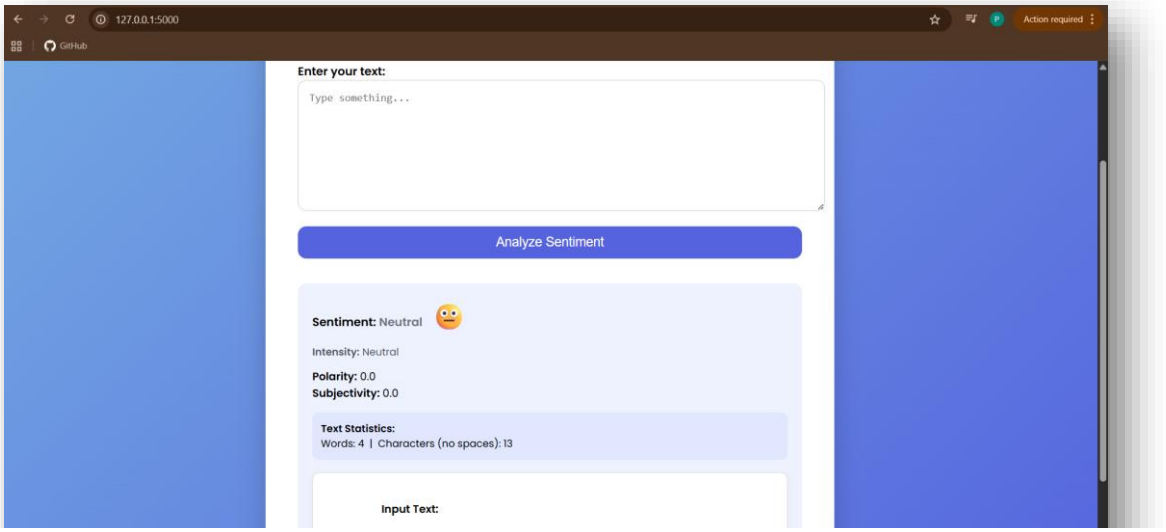
---

8. Output Screenshots

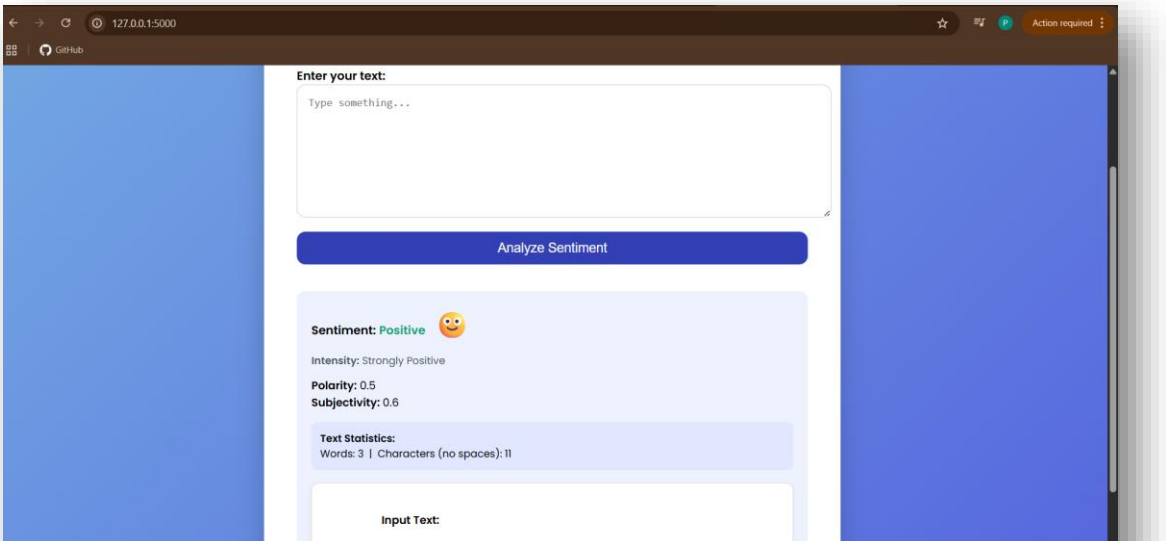
8.1 Home Page UI



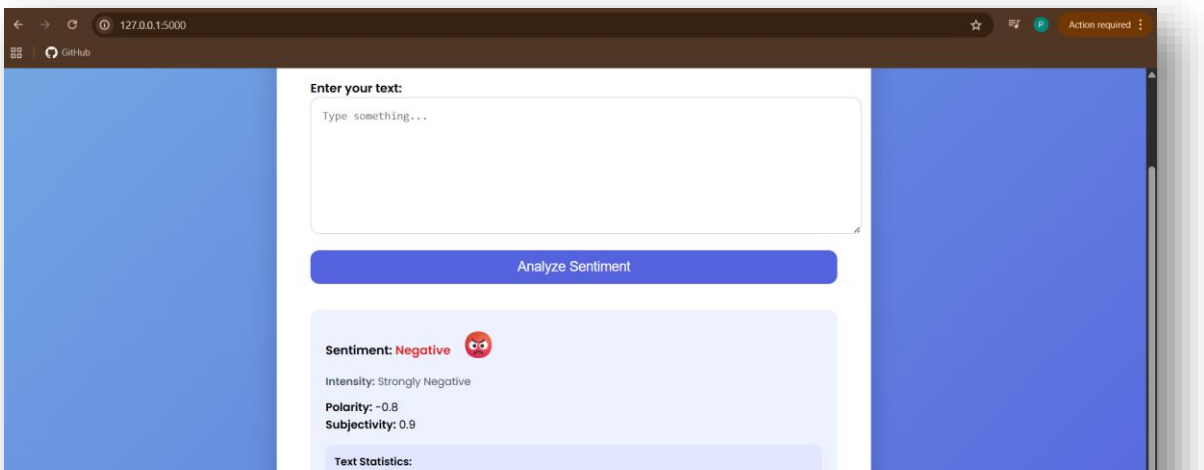
8.2 Text Input + Analysis Result



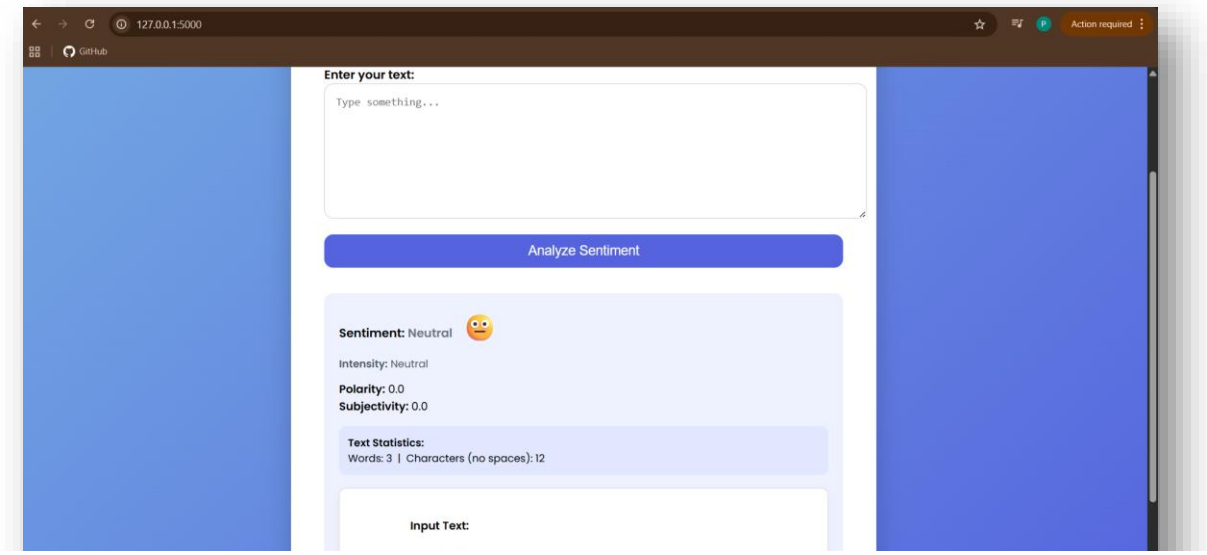
8.3 Positive Sentiment Example



8.4 Negative Sentiment Example



8.5 Neutral Sentiment Example



---

9. Results

The project successfully performs:

✓ Sentiment Classification
✓ Polarity Score Calculation
✓ Subjectivity Score Calculation
✓ Emoji-based emotion indicator
✓ Sentiment intensity levels
✓ Word and character counts
✓ Clean UI with responsive layout

The system is accurate, user-friendly, and easy to understand.

---

10. Conclusion

This project demonstrates the use of **Natural Language Processing** (NLP) in web applications.
Using Flask and TextBlob, the system effectively performs sentiment analysis and displays multiple metrics that help users understand emotional tone in text.

The project meets all internship requirements and serves as a practical example of integrating Python, Machine Learning concepts, and web development.

---

11. Future Scope

- Add multilingual sentiment analysis
- Add live sentiment graph
- Export results as PDF
- Add database for storing history
- Deploy the app online using Render or PythonAnywhere
