PROJECT REPORT

PROJECT ON SENTIMENT ANALYSIS

-SAKSHI

CONTENTS

- **Overview of Sentiment Detection System**
  - What
  - Why
  - How
- **Prerequisites**
- **Create Virtual Environment**
- **Google Form and Sheet Integration**
  - Create a Google Form
  - Linking Google Form with Google Sheets
- **Set up Google Sheet API Access**
- **Read Google Sheet Headers with Python**
- **Sentiment Analysis**
  - Python Program for Sentiment Analysis on User Input:
  - Sentiment Analysis from Google Sheets
  - Add sentiment to Data Frame and export to CSV
- **Visualisation of Analysis**
- **Frontend Development**
  - Frontend in streamlit
  - Home Section in streamlit
  - Analyse Section in streamlit
  - Result Section in streamlit
- **Summary of Project**
- **Conclusion**

OVERVIEW OF SENTIMENT ANALYSIS SYSTEM

WHAT:

- It's an NLP (Natural Language Processing) system that focuses on understanding and interpreting human language. The system analyses text data to find the meaning behind words and phrases, allowing it to detect the sentiment expressed. NLP helps machines process natural language in a way that mimics human understanding.

Sentiment Classification:

- This application predicts the sentiments into 3 categories:
    - Positive
    - Negative
    - Neutral

Demographic Analysis:

- This application then visualizes the results based on different factors such as age, gender etc

Data Source Integration:

- This application can collect data from Google Form and retrieve it through Google Sheets.

Web-Based Platform:

- This application is a web application which can be accessed over a LAN.

WHY:

Real-World Applications

- Product and Service Monitoring: Analysing customer reviews and feedback to improve quality.
- Survey Analysis: Understanding user responses and opinions in surveys.
- Social Media Monitoring: Tracking public sentiment towards brands, events, or trends.
- Feedback Analysis: Extracting insights from customer opinions to drive decision-making.

Skill Development

- This project is an excellent opportunity to develop and enhance your technical skills, including:
    - Programming (Python)
    - Machine Learning and Data Analysis
    - Project Implementation Techniques
    - Natural Language Processing (NLP) Concepts

Professional Value

- Working on a sentiment analysis system demonstrates both practical and analytical abilities, which are highly valued in the job market. It makes for a strong addition to your resume.

HOW:

Backend

- **Data Collection:**
  A Google Sheet is created and connected using Python. Feedback data submitted through a Google Form is automatically stored in this sheet.
- **Data Organization:**
  The raw data from the sheet is cleaned and structured using the Pandas library.
- **Data Analysis (NLP):**
  The core of the backend uses Natural Language Processing to analyse user opinions. Libraries like VADER.
- **Data Visualization:**
  The results from the sentiment analysis are visualized using tools such as Plotly.

Frontend

- **User Input:**
  A Google Form is used to collect feedback from users.
- **Web Application:**
  The system is deployed as a web application using streamlit, allowing users to access results and analysis via a browser.

PREREQUISITES

This list includes software, hardware, and knowledge requirements essential for developing an effective sentiment analysis system.

KNOWLEDGE PREREQUISITES

- Python Programming – Basic to intermediate proficiency
- Natural Language Processing (NLP)
- Machine Learning / Deep Learning – Especially Recurrent Neural Networks (RNNs).
- Text Pre-processing – Cleaning, lowercasing, stop word removal.
- Model Evaluation – Understanding of metrics like accuracy, precision, recall, F1-score etc.

SOFTWARE REQUIREMENTS

1. **Programming Language:** Python – Visit the official Python website to install

2. **Libraries / Frameworks:**

    - Streamlit – For creating interactive and user-friendly web applications for data analysis and machine learning models
    - InstalledAppFlow – For handling secure user authentication via Google's OAuth 2.0 system
    - build – For accessing and interacting with Google APIs (e.g., YouTube Data API)
    - NLTK- Sometimes used with VADER as it originally comes from the nltk library, But you can also install and use Vader Sentiment independently
    - Sentiment Intensity Analyzer – For fast, rule-based sentiment analysis using the VADER (Valence Aware Dictionary and Sentiment Reasoner) model
    - Pandas – For data manipulation, transformation, and tabular data analysis
    - Plotly – For building interactive and visually appealing data visualizations

DATASET REQUIREMENTS

datasets: The dataset includes user feedback collected on a Smart Toothbrush product. This feedback will be used to perform sentiment analysis and classify responses as positive, negative, or neutral.

HARDWARE REQUIREMENTS

- Basic Computer Setup – Laptop or desktop with at least 8GB RAM (16GB recommended for deep learning models)

CREATE VIRTUAL ENVIRONMENT

A **virtual environment** (venv) is an isolated workspace for Python projects. It allows you to install dependencies separately from the system-wide Python installation, preventing conflicts between different projects.

Features of Virtual Environment:

- Dependency Management – Keeps project-specific libraries separate.
- Avoid Conflicts – Different projects can use different versions of the same package.
- Reproducibility – Makes it easier to share projects with others.
- Cleaner Development – Prevents clutter in the global Python environment.

Steps for Creating a Virtual Environment (V.E)

- Go to C Drive → Create a Project Folder
  Inside the Project Folder, create a new folder named Sentiment.
- This Sentiment folder will act as the V.E (Virtual Environment).
- You can create multiple virtual environments for different projects in this Project Folder.

Now, Create the Virtual Environment

- Copy the address: C:\Projects\Sentiment
- Open Command Prompt (cmd)
- Write the address in cmd or navigate to the folder manually
- Run the following command:

```
python -m venv C:\Projects\Sentiment
```

- If successful, you will see the virtual environment created inside the Sentiment folder

Inside the Sentiment Folder, you should see:

- Include
- lib
- Scripts

Activate Virtual Environment:

- Navigate to the Scripts folder inside your Sentiment folder, Click on the address bar and type cmd.
- Run the following command: activate

```
C:\Projects\Sentiment\Scripts>activate
```

Important Notes:

- You must activate the virtual environment every time you want to use it
- If you don’t activate it, Python will use the global environment instead of the virtual one
- If activated correctly, all installed packages will be saved in the virtual environment, not in the main system

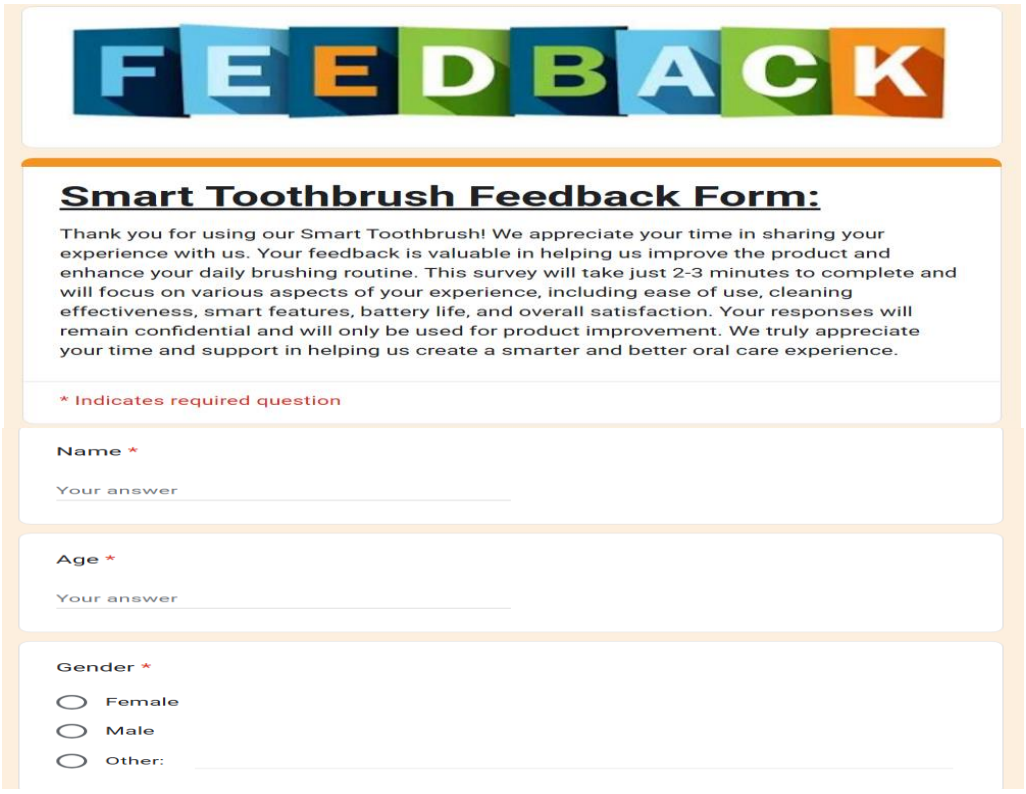
GOOGLE FORM AND SHEET INTEGRATION

➢ How to Create a Google Form:

- Go to Google Forms: Visit forms.google.com — make sure you're signed in with your Google account.
- Start a New Form: Click on the blank form or choose a template (like feedback, quiz, etc.).
- Add a Title and Description: At the top, give your form a name and optionally write a short description.
- Create Your Questions: Click the **"+"** icon to add new questions. Choose the question type (short answer, multiple choice, checkboxes, dropdown, etc.
- Customize Settings: Click the gear icon in the top-right to adjust form settings like response collection, limiting to one response per user, quizzes, and confirmation messages.
- Preview and Share: Use the eye icon to preview your form. When you're ready to share, click the Send button to get a link, email it, or embed it on a website.

For instance, I created a Google Form to collect feedback on a newly launched electric toothbrush. The form includes questions about user satisfaction, product performance, and suggestions for improvement. It was quick and easy to build using Google Forms' intuitive interface, with a mix of multiple-choice and open-ended questions.

Google form for taking the feedback:



Q.1 How satisfied are you with your Smart Toothbrush overall? *

	0	1	2	3	4	5	
Least Satisfied	○	○	○	○	○	○	Most Satisfied

Q.2 How easy was it to set up and start using the Smart Toothbrush? *

Choose

Q.3 How effective do you find the toothbrush in cleaning your teeth compared to a regular toothbrush? *

Choose

Q.4 Which features do you use the most? *

- ☐ AI Feedback
- ☐ Pressure Sensor
- ☐ Brushing Timer
- ☐ Bluetooth Sync
- ☐ Other:

Q.5 Have you experienced any technical issues with the Smart Toothbrush? *

Your answer

Q.6 How useful do you find the AI feedback and brushing recommendations? *

	0	1	2	3	4	5	
Not Usefull	○	○	○	○	○	○	Extremly Useful

Q.7 How satisfied are you with the battery life of the toothbrush? *

	0	1	2	3	4	5	
Very Dissatisfied	○	○	○	○	○	○	Very Satisfied

Q.8 How often do you sync your toothbrush with the mobile app? *

- ○ Daily
- ○ Weekly
- ○ Occasionally
- ○ Never

What is your opinion on the overall quality of the product? *

Your answer

Q.10 Would you recommend this Smart Toothbrush to others? *

Your answer

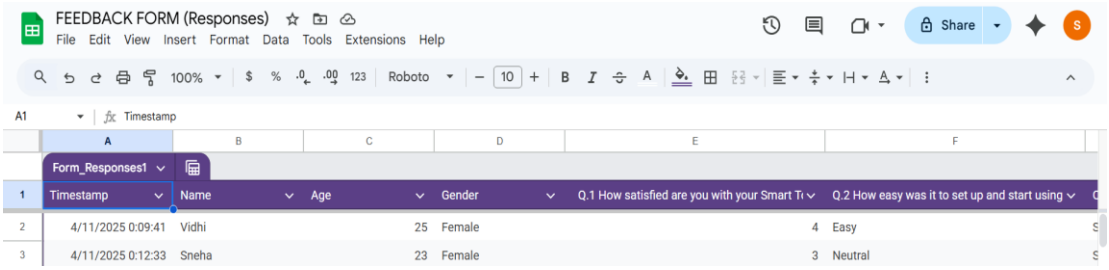
Submit Clear form

➢ Linking Google Form with Google Sheets

- Open Your Google Form: Go to Google Forms and open the form you've created.
- Go to the "Responses" Tab: Click on the "Responses" tab at the top of the form.
- Click on the Google Sheets Icon: You'll see a small green Sheets icon – click it to create or link a spreadsheet.
- Choose Where to Store Responses: Create a new spreadsheet, or select an existing spreadsheet to store the responses.

Responses section of google form:

Linked google sheet:



Your Google Form is now linked to a Google Sheet. New form responses will automatically populate as new rows in the sheet.

SET UP GOOGLE SHEETS API ACCESS

To work with Google Sheets using Python, you first need to set up access through the Google Sheets API. This connection enables your Python scripts to interact with spreadsheets in a secure and efficient way. Whether you're automating data entry, analysing records, or building custom dashboards, the API provides the flexibility to work directly with live data.

Google Account

- To start, you need a Google Account. This will give you access to Google services like Google Drive, Google Sheets, and the Google Cloud Console.
- If you already have a Gmail account, you're good to go. If not, visit accounts.google.com and sign up for a free account.
- Make sure you're signed in before proceeding with any of the steps below.

Google Project

- Navigate to the Google Cloud Console.
- Click on the project dropdown menu at the top of the page and select “New Project.”
- Enter a project name, choose a billing account (if applicable), and select your organization (optional).
- Click “Create” to generate your new project.
- After it's created, make sure to select the project to activate it for your upcoming tasks.

Enable Google Sheets API

- Inside your active project, go to "APIs & Services" > "Library".
- Use the search bar to look for “Google Sheets API.”
- Click on it and press the “Enable” button. This will allow your project to interact with Google Sheets.

Create a Consent Application

- Go to "APIs & Services" > "OAuth consent screen".
- Choose the user type:
  - External: For apps accessed by users outside your organization (default choice).
  - Internal: For apps used only within your Google Workspace organization.
- Fill in the app information:
  - App Name
  - User Support Email
  - Developer Contact Information

- You can skip or add scopes depending on what access your app needs (e.g., access to Google Sheets).
- In the Test Users section, add the Google email addresses that will be allowed to test the app.
- Save the configuration and finish setting up the screen.

Download Credentials

- Go to "APIs & Services" > "Credentials".
- Click “Create Credentials” and choose “OAuth Client ID.”
- Select the application type based on your use case:
  - Desktop App: For local apps or scripts.
  - Web Application: For browser-based tools.
- Give it a recognizable name and click Create.
- A new client ID will be generated. Click “Download JSON” to get the credentials file.
- Save this file securely. You'll use it in your script or application to authenticate and access Google Sheets via API.

READ GOOGLE SHEET HEADERS WITH PYTHON

To read column headings from a Google Sheet using Python, you can use the Google Sheets API along with the required library. First, authenticate using a service account and connect to the desired sheet.

Open the backend.py file in your in-code editor, and add the following code to read google sheet headings:

```
from google_auth_oauthlib.flow import InstalledAppFlow  #For handling OAuth2 flow
from googleapiclient.discovery import build # For building the Sheets API service
#Permission

# Set up OAuth2.0 authentication flow using the client secret JSON file
f=InstalledAppFlow.from_client_secrets_file("key.json",["https://www.googleapis.com/auth/spreadsheets"])
cred=f.run_local_server(port=0) # Run the local server to authenticate and get the credentials

service=build("sheets","v4",credentials=cred).spreadsheets().values() # Build the Google Sheets API service
# Make a GET request to read data from the specified spreadsheet and range

d=service.get(spreadsheetId="13hxHfmWpKNB6PX3s98FripX_dCbhwhTwo6luzQuGqcQ",range="A:0").execute()
data=d['values'] # Extract the 'values' key from the response, which contains the actual sheet data
print(data)# Print the fetched data
```

- Import Libraries: Use libraries for authentication, API access, and data handling (like pandas).
- Prepare your authentication setup: Set up access to Google Sheets by using credentials from a key.json file, which you get from your Google Cloud Console.
- Authenticate the user: A small web browser window will open asking you to log in and give permission to access your Google Sheets.
- Connect to the Google Sheets service: After successful login, you create a connection to the Google Sheets API so your script can send and receive data.
- Specify the spreadsheet and data range: Choose which spreadsheet you want to access by providing its unique ID, and define that you only want to read the first row (which usually contains the column headings).
- Retrieve the first row of data: Send a request to Google Sheets asking for just the first row of values.
- Extract and use the headings: Once the data is received, take the first row and use it as your list of headings (column names), which can then be printed or used however you need.

Final Output:

```
(Sentiment) C:\Projects\Sentiment\Scripts>python backend.py
Please visit this URL to authorize this application: https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=889286581
881-9f1a1j3cpb8bsjapkh0p1f3q5cgj6rpt.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A52214%2F&scope=https%3A%2F%2Fww
w.googleapis.com%2Fauth%2Fspreadsheets&state=rulgmMLqNyLjPr4de5immP2Q5p6I70&access_type=offline
[['Timestamp', 'Name', 'Age', 'Gender', '  Q.1 How satisfied are you with your Smart Toothbrush overall?  ', 'Q.2 How easy was it to
set up and start using the Smart Toothbrush?', 'Q.3  How effective do you find the toothbrush in cleaning your teeth compared to a re
gular toothbrush?', 'Q.4 Which features do you use the most?  ', '  Q.5 Have you experienced any technical issues with the Smart Toot
hbrush?  ', 'Q.6 How useful do you find the AI feedback and brushing recommendations?', '  Q.7 How satisfied are you with the battery
 life of the toothbrush?  ', '  Q.8 How often do you sync your toothbrush with the mobile app?  ', 'What is your opinion on the overa
ll quality of the product?', '  Q.10 Would you recommend this Smart Toothbrush to others?', 'Column 1'], ['4/11/2025 0:09:41', 'Vidhi
', '25', 'Female', '4', 'Easy', 'Slightly Better', 'Pressure Sensor, Brushing Timer', 'No', '4', '4', 'Occasionally', 'Good product',
 'Yes'], ['4/11/2025 0:12:33', 'Sneha', '23', 'Female', '3', 'Neutral', 'Slightly Better', 'Brushing Timer, Bluetooth Sync', 'No', '3
', '3', 'Weekly', 'Great product', 'Yes'], ['4/11/2025 0:13:29', 'Ruhi', '26', 'Female', '4', 'Neutral', 'Much Better', 'Pressure Sen
sor, Bluetooth Sync', 'No', '3', '3', 'Occasionally', 'Good Product', 'Yes']]
```

- The Python script successfully connected to a Google Sheet.
- An authentication URL was provided to grant access via a Google account.
- Survey responses about a Smart Toothbrush were retrieved from the sheet.
- Each response includes timestamp, name, age, gender, and answer details.
- The data is printed in a structured format, confirming successful extraction.

To get the all the rows of particular column:

Open the backend.py file in your in-code editor, and edit the following code to get the all rows of a particular column:

```python
from google_auth_oauthlib.flow import InstalledAppFlow  #For handling OAuth2 flow
from googleapiclient.discovery import build # For building the Sheets API service
import pandas as pd#Permission

# Set up OAuth2.0 authentication flow using the client secret JSON file
f=InstalledAppFlow.from_client_secrets_file("key.json",["https://www.googleapis.com/auth/spreadsheets"])
cred=f.run_local_server(port=0) # Run the local server to authenticate and get the credentials

service=build("sheets","v4",credentials=cred).spreadsheets().values() # Build the Google Sheets API service
# Make a GET request to read data from the specified spreadsheet and range

d=service.get(spreadsheetId="13hxHfmWpKNB6PX3s98FripX_dCbhwhTwo6luzQuGqcQ",range="A:N").execute()
data=d['values'] # Extract the 'values' key from the response, which contains the actual sheet data

# Step 6: Convert data to a pandas DataFrame
df = pd.DataFrame(data=data[1:], columns=data[0])
for i in range(0, len(df)): # Loop through each row and print the value in the "Opinion" column
    t = df._get_value(i, "What is your opinion on the overall quality of the product?")  # Get the value of the column for each row
    print(t)
```

- Import Libraries: Use libraries for authentication, API access, and data handling (like pandas).
- Authenticate with Google: Use the client secret JSON to start OAuth2.0 flow. A browser opens for Google login.
- Connect to Google Sheets API: Build the Sheets API service using the authenticated credentials.
- Fetch Sheet Data: Request data from a specific spreadsheet and range (like columns A to N).
- Convert to Data Frame: Use pandas to turn the data into a table format using the first row as headers.
- Access Specific Column: Loop through the Data Frame and print values from a particular column (e.g., product opinion).

Final Output:

```
(Sentiment) C:\Projects\Sentiment\Scripts>backend.py
Please visit this URL to authorize this application: https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=889286581
881-9f1a1j3cpb8bsjapkh0p1f3q5cgj6rpt.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A52918%2F&scope=https%3A%2F%2Fww
w.googleapis.com%2Fauth%2Fspreadsheets&state=rQbcmKuAXkLohktImXuKsGlj3P7on6&access_type=offline
Good product
Great product
Good Product
```

- The script prompts for authentication via a Google-provided URL to access Google Sheets.
- After successful authorization, it retrieves specific data from the connected Google Sheet.
- The output displays user responses related to product feedback.

SENTIMENT ANALYSIS

Sentiment Analysis is a Natural Language Processing (NLP) technique used to determine whether a piece of text expresses a positive, negative, or neutral sentiment. It is commonly used in:

Install the required libraries:

```
(Sentiment) C:\Projects\Sentiment\Scripts>pip install nltk
(Sentiment) C:\Projects\Sentiment\Scripts>pip install vaderSentiment
```

VADER SENTIMENT:

- Provides the Sentiment Intensity Analyzer class.
- It's efficient for analysing short, informal text like tweets, reviews, and comments.

NLTK:

- Sometimes used with VADER as it originally comes from the nltk library.
- But you can also install and use vader Sentiment independently.

Compound Score in VADER

The compound score is a number between -1 and +1 that represents the overall sentiment of a sentence.

How It Works:

- The VADER model gives individual scores for positive, negative, and neutral words.
- Then, it combines them into one single value — called the compound score — which tells us the overall mood of the text.

Interpretation:

- compound > 0.5 → Strong Positive Sentiment
- compound < -0.5 → Strong Negative Sentiment
- between -0.5 and 0.5 → Neutral or Mixed Sentiment

Get the Sentiment Score:

Open the backend.py file in your in-code editor, and write the following code to get the sentiment score:

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
mymodel = SentimentIntensityAnalyzer() # Initialize the model
pred = mymodel.polarity_scores("I Love BurgerKing") #Get the sentiment score
print(pred) #Print the sentiment score
```

- Import VADER – Bring in the sentiment analysis tool.
- Initialize Analyzer – Create a model to analyze text.
- Input Text – Give a sentence like *"I love BurgerKing"*.
- Get Sentiment Scores – The model gives scores for positive, negative, neutral, and overall sentiment (compound).
- View Result – Use the compound score to tell if the sentiment is Positive, Negative, or Neutral.

Final Output:

```
(Sentiment) C:\Projects\Sentiment\Scripts>backend.py
{'neg': 0.0, 'neu': 0.417, 'pos': 0.583, 'compound': 0.6369}
```

Positive Sentiment (pos: 0.583):

- 58.3% of the content is identified as positive.
- Indicates a strong positive tone.

Neutral Sentiment (neu: 0.417):

- 41.7% of the content is neutral.
- Suggests some informational or balanced language.

Negative Sentiment (neg: 0.0):s

- 0% negative content detected.
- No negativity present in the text.

Compound Score (compound: 0.6369):

- The overall sentiment score is 0.6369 (on a scale from -1 to +1).
- A score above 0.5 is considered clearly positive.

The analysed text has a clearly positive sentiment, with no negativity and a good balance of positive and neutral tones.

➢ Python Program to Perform Sentiment Analysis on User Input Text:

Open the backend.py file in your in-code editor, and write the following code to analyse the user input:

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
mymodel = SentimentIntensityAnalyzer() # Initialize the model
k = input("Enter the Text") # Take input from user
pred = mymodel.polarity_scores(k)# Get polarity scores
if(pred['compound'] > 0.5):
    print("Sentiment is Positive")# Print if compound score is greater than 0.5
elif(pred['compound'] < -0.5):
    print("Sentiment is Negative")# Print if compound score is less than -0.5
else:
    print("Sentiment is Neutral") # Print if the score is between -0.5 and 0.5
```

- Import Sentiment Intensity Analyzer from the Vader Sentiment library.
- Create an object of the analyser to process text input.
- Take a sentence or phrase from the user using input.
- Use polarity scores to get sentiment scores of the input.
- Check the compound score to determine if it's positive, negative, or neutral.
- Print the sentiment result based on the compound score value.

Final Output:

```
(Sentiment) C:\Projects\Sentiment\Scripts>backend.py
Enter the Text I love Pizza
Sentiment is Positive

(Sentiment) C:\Projects\Sentiment\Scripts>backend.py
Enter the Text I Hate pizza
Sentiment is Negative

(Sentiment) C:\Projects\Sentiment\Scripts>backend.py
Enter the Text I dont like pizza too much
Sentiment is Neutral
```

Input: I love Pizza:

- Sentiment is Positive
- Love" is a strong positive word, so the compound score is high.

Input: I Hate pizza

- Sentiment is Negative
- "Hate" is a strong negative word, so the compound score is low.

Input: I don't like pizza too much

- Sentiment is Neutral
- This sentence expresses a mild dislike, not strong enough to push it below the negative threshold.

➢ Sentiment Analysis from Google Sheets using VADER:

Open the backend.py file in your in-code editor, and write the following code to analyse the google sheet data which we collected through google form input:

```
# Import necessary libraries
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import pandas as pd
# Initialize the sentiment analyzer
mymodel = SentimentIntensityAnalyzer()
#Permission
# Set up OAuth2.0 flow using credentials from key.json
f = InstalledAppFlow.from_client_secrets_file("key.json",["https://www.googleapis.com/auth/spreadsheets"])
# Run the local server to complete authentication
cred = f.run_local_server(port=0)
# Build the Google Sheets API service
service = build("sheets", "v4", credentials=cred)
# Access the values from the specified spreadsheet and range
sheet = service.spreadsheets().values()
k = sheet.get(spreadsheetId="13hxHfmWpKNB6PX3s98FripX_dCbhwhTwo6luzQuGqcQ",range="B:N").execute()
# Data Processing and Sentiment Analysis
# Extract values from the response
d = k['values']
# Convert the data into a pandas DataFrame
df = pd.DataFrame(data=d[1:], columns=d[0])
# Loop through each row of the DataFrame
for i in range(0, len(df)):
    # Get the text/opinion from the "Opinion" column
    t = df._get_value(i, "What is your opinion on the overall quality of the product?")

    # Use VADER to analyze sentiment of the text
    pred = mymodel.polarity_scores(t)

    # Determine and print the sentiment based on compound score
    if pred['compound'] > 0.5:
        print(t, "Sentiment is Positive")
    elif pred['compound'] < -0.5:
        print(t, "Sentiment is Negative")
    else:
        print(t, "Sentiment is Neutral")
```

- Import Required Libraries: It starts by importing libraries for Google Sheets access, sentiment analysis (VADER), and data handling (Pandas).
- Initialize Sentiment Analyzer: It creates an instance of the VADER sentiment analyzer to analyse text.
- Authenticate with Google Sheets API: It sets up OAuth 2.0 authorization using a credentials file to access Google Sheets securely.
- Connect to the Google Sheets Service: It builds a connection to the Google Sheets API using the credentials.
- Access the Spreadsheet and Specific Data Range: It specifies the target Google Sheet and selects a range of cells (from columns B to N) to read data from.

- Extract and Process the Data: The script extracts the sheet data and converts it into a Pandas Data Frame for easier manipulation.
- Loop Through Each Row of Data: For each row in the Data Frame, it retrieves a response from a specific column containing a user's opinion about a product.
- Analyse Sentiment Using VADER: It analyzes the opinion text using VADER to get sentiment scores (like positive, neutral, or negative).
- Classify and Print Sentiment: Based on the sentiment score (compound value), it prints whether the sentiment is positive, negative, or neutral.

Final Output:

```
(Sentiment) C:\Projects\Sentiment\Scripts>backend.py
Please visit this URL to authorize this application: https:/
881-9f1a1j3cpb8bsjapkh0p1f3q5cgj6rpt.apps.googleusercontent.
w.googleapis.com%2Fauth%2Fspreadsheets&state=csabC0m19nnJhdD
Good product Sentiment is Neutral
Great product Sentiment is Positive
Good Product Sentiment is Neutral
Nice product  Sentiment is Neutral
Not bad  Sentiment is Neutral
Overall good quality  Sentiment is Neutral
I don't like this product  Sentiment is Neutral
Very good quality  Sentiment is Neutral
Very good Sentiment is Neutral
Difficult to use Sentiment is Neutral
I hate the product.  Sentiment is Negative
Not very helpful,   Sentiment is Neutral
```

- The script analyses opinions from a Google Sheet using the VADER sentiment analyser.
- Each opinion is classified as Positive, Negative, or Neutral.
- Sentiment is determined using a compound score from the VADER model.
- If the score is > 0.5, the sentiment is Positive.
- If the score is < -0.5, it's Negative; otherwise, it's Neutral.
- The script prints the original opinion along with its sentiment label.

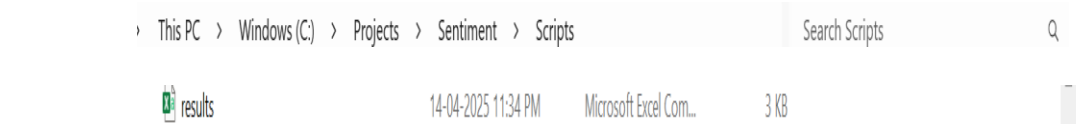
➢ Add Sentiment Results to Data Frame and Export to CSV:

Open the backend.py file in your in-code editor, and edit the following code to add the column "Sentiments" and save the file in scripts folder of virtual Environment:

```
# Import necessary libraries
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import pandas as pd
# Initialize the sentiment analyzer
mymodel = SentimentIntensityAnalyzer()
#Permission
# Set up OAuth2.0 flow using credentials from key.json
f = InstalledAppFlow.from_client_secrets_file("key.json",["https://www.googleapis.com/auth/spreadsheets"])
# Run the local server to complete authentication
cred = f.run_local_server(port=0)
# Build the Google Sheets API service
service = build("sheets", "v4", credentials=cred)
# Access the values from the specified spreadsheet and range
sheet = service.spreadsheets().values()
k = sheet.get(spreadsheetId="13hxHfmWpKNB6PX3s98FripX_dCbhwhTwo6luzQuGqcQ",range="B:N").execute()
# Data Processing and Sentiment Analysis
# Extract values from the response
d = k['values']
# Convert the data into a pandas DataFrame
df = pd.DataFrame(data=d[1:], columns=d[0])
l=[] #taking list to store the value of sentiments
# Loop through each row of the DataFrame
for i in range(0, len(df)):
    # Get the text/opinion from the "Opinion" column
    t = df._get_value(i, "What is your opinion on the overall quality of the product?")

    # Use VADER to analyze sentiment of the text
    pred = mymodel.polarity_scores(t)

    # Determine and print the sentiment based on compound score
    if pred['compound'] > 0.5:
        l.append("Positive")
    elif pred['compound'] < -0.5:
        l.append("Negative")
    else:
        l.append("Neutral")
df['sentiment']=l #adding the column of Sentiments
df.to_csv("results.csv",index=False) #saving the csv file
```

- Import Libraries: Required libraries like pandas, vader Sentiment, and Google API clients are imported.
- Initialize Sentiment Analyzer: VADER's sentiment analyser is set up.
- Authenticate Google Sheets Access: Uses OAuth 2.0 with a key.json file to access Google Sheets.
- Fetch Data from Google Sheets: Connects to a specific Google Sheet and retrieves data from a specified range.
- Convert Data to Data Frame: The retrieved data is converted into a pandas Data Frame.
- Perform Sentiment Analysis: Each opinion is analysed using VADER, and its sentiment (Positive, Negative, or Neutral) is determined.
- Store Sentiment Results: Sentiments are stored in a new column in the Data Frame.
- Export to CSV: The final Data Frame with sentiment results is saved to a file named results.csv.

Final Output:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Name	Age	Gender	Q.1 How	Q.2 How e	Q.3 How	Q.4 Which	Q.5 Have	Q.6 How u	Q.7 How	Q.8 How	What is yo	Q.10 Wo	sentiment
2	Vidhi	25	Female	4	Easy	Slightly Be	Pressure S	No	4	4	Occasiona	Good proc	Yes	Neutral
3	Sneha	23	Female	3	Neutral	Slightly Be	Brushing T	No	3	3	Weekly	Great proc	Yes	Positive
4	Ruhi	26	Female	4	Neutral	Much Bett	Pressure S	No	3	3	Occasiona	Good Proc	Yes	Neutral
5	Gauri	23	Female	3	Neutral	About the	AI Feedba	No	3	3	Occasiona	Nice prod	Yes	Neutral
6	Vijay	43	Male	2	Neutral	About the	Bluetooth	Yes	2	3	Never	Not bad	No	Neutral
7	Nikita	21	Female	3	Neutral	Slightly Be	AI Feedba	Not really	3	3	Weekly	Overall go	Yes	Neutral
8	Geetanjali	49	Female	1	Difficult	About the	Brushing T	Yes	2	2	Never	I donâ€™t	No	Neutral
9	Geeta	45	Female	4	Neutral	Much Bett	Pressure S	Yes but m	4	3	Occasiona	Very good	Yes	Neutral
10	Anshul	27	Male	4	Easy	Much Bett	AI Feedba	No	4	4	Weekly	Very good	Yes	Neutral
11	Gopal	17	Male	2	Difficult	About the	Brushing T	Yes very m	1	2	Never	Difficult to	No	Neutral
12	Vaibhav	37	Male	0	Difficult	Much Wo	No feature	Yes, a lot	0	0	Never	I hate the	Never	Negative
13	Kashish	30	Female	1	Difficult	Slightly W	Pressure S	Yes	1	1	Occasiona	Not very h	No	Neutral

The final Data Frame with sentiment results is saved to a file named results.csv.

SENTIMENT ANALYSIS VISUALISATION

Sentiment analysis is a natural language processing (NLP) technique used to determine the emotional tone behind a body of text. It is commonly applied to understand customer opinions, public reactions, and social media trends. However, the true value of sentiment analysis lies not just in extracting sentiment, but in effectively visualizing it.

Why Visualise Sentiments:

Visualizing sentiment data helps in:

- **Quick Insights:** Easily grasp the overall mood or sentiment trend in a dataset.
- **Pattern Recognition:** Identify fluctuations in sentiment over time or across categories.
- **Decision Making:** Support data-driven decisions based on public or customer perception.
- **Communication:** Present results in a compelling and understandable way to stakeholders.

Requirement: Installation of Plotly

In this example, we use Plotly, a powerful and interactive Python visualization library:

```
(Sentiment) C:\Projects\Sentiment\Scripts>pip install plotly
```

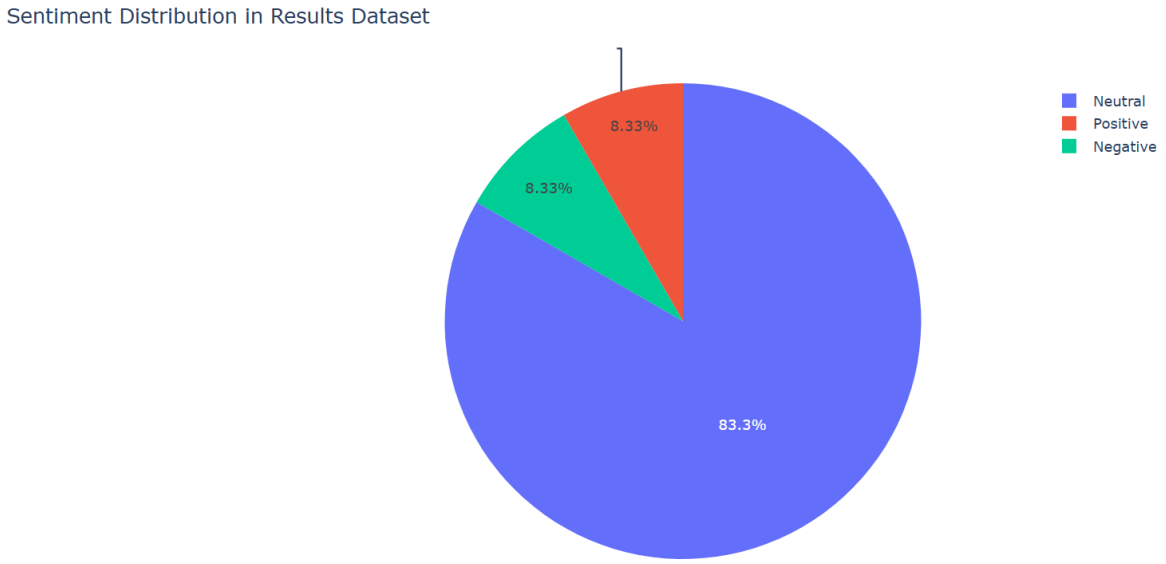
Sentiment Analysis Visualization Using Pie Chart:

Open the backend.py file in your in-code editor, and add following code to analysing the sentiments using Pie chart:

```
# Importing necessary libraries
import pandas as pd            # For data manipulation
import plotly.express as px    # For plotting interactive charts

# Reading the CSV file containing sentiment data into a DataFrame
df = pd.read_csv("results.csv")

# Calculating percentage of 'Positive' sentiments
posper = (len(df[df['sentiment'] == 'Positive']) / len(df)) * 100

# Calculating percentage of 'Negative' sentiments
negper = (len(df[df['sentiment'] == 'Negative']) / len(df)) * 100

# Calculating percentage of 'Neutral' sentiments
neuper = (len(df[df['sentiment'] == 'Neutral']) / len(df)) * 100

# Creating a pie chart to visualize sentiment distribution
fig = px.pie(values=[posper, negper, neuper],names=['Positive', 'Negative', 'Neutral'], title="Sentiment Distribution in Results Dataset" )
# Displaying the pie chart
fig.show()
```

- The code performs sentiment analysis visualization using a pie chart.
- It imports pandas for data manipulation and plotly express for plotting.
- Sentiment data is read from a CSV file (results.csv) into a Data Frame.
- It calculates the percentage of Positive, Negative, and Neutral sentiments.
- These percentages are used to generate a pie chart showing sentiment distribution.
- The chart includes a title: "Sentiment Distribution in Results Dataset".

Sentiment distribution using Pie chart:



- The pie chart shows sentiment distribution in the dataset: 83.3% Neutral, 8.33% Positive, and 8.33% Negative.
- Neutral sentiment dominates the dataset, indicating mostly unbiased or factual content.
- Positive and Negative sentiments are equally present but in much smaller proportions.
- This suggests a strong leaning toward neutrality in the overall data tone.

Gender- and Age-Wise Sentiment Analysis using Box Plot:

Open the backend.py file in your in-code editor, and edit the following code to edit the code to Gender and age wise sentiment analysis using Box Plot:

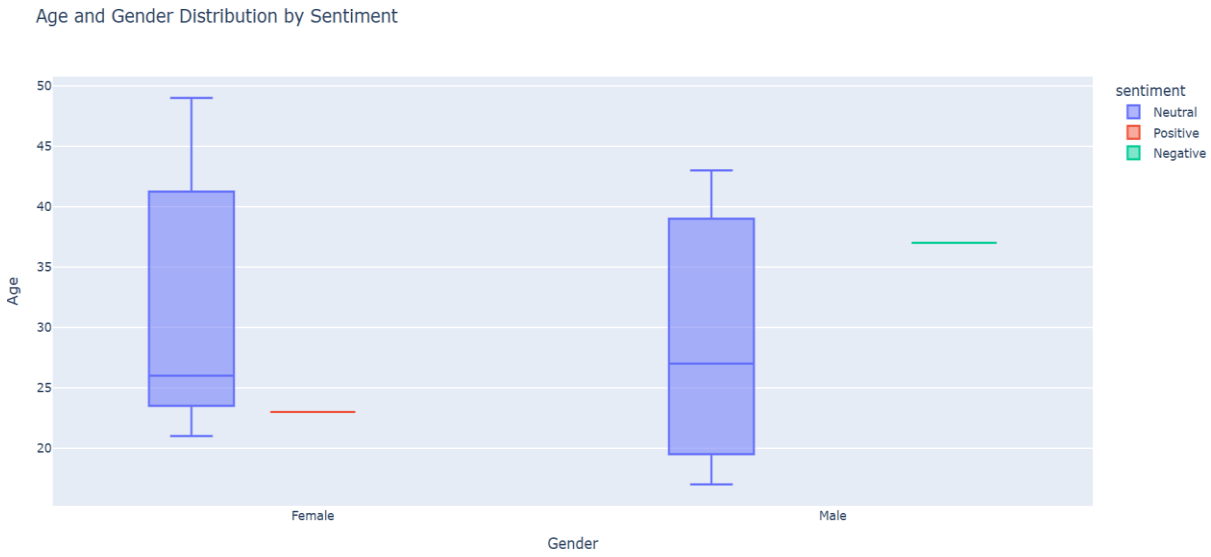
```
#importing required libraries
import pandas as pd
import plotly.express as px

# Load the dataset
df = pd.read_csv("results.csv")

# Create the box plot
fig = px.box(df,x="Gender", y="Age", color="sentiment", title="Age and Gender Distribution by Sentiment")

# Show the plot
fig.show()
```

- The code imports the pandas library as pd for data handling and plotly express as px for data visualization.
- It reads a CSV file named "results.csv" into a pandas Data Frame called df. A box plot is created using plotly express box, where:
    - x axis represents the "Gender" column,
    - y axis represents the "Age" column,
    - different colours are used to represent different values in the "sentiment" column,

- and the plot has the title "Age and Gender Distribution by Sentiment".

- Now, the plot is displayed using.

Age and Gender Distribution by sentiment using Boxplot:



- X-axis shows Gender (Male, Female) and Y-axis shows Age, representing how age is distributed across genders.
- The plot is color-coded by Sentiment: Neutral, Positive, and Negative.
- Neutral sentiment is present in both genders, showing a wider age spread with visible medians and interquartile ranges.
- Positive sentiment appears only in females, with a narrow age range around 23.
- Negative sentiment is seen only in males, around the age of 37, with no spread, indicating possibly a single data point.

Toothbrush-App Syncing Frequency by Gender using Histogram

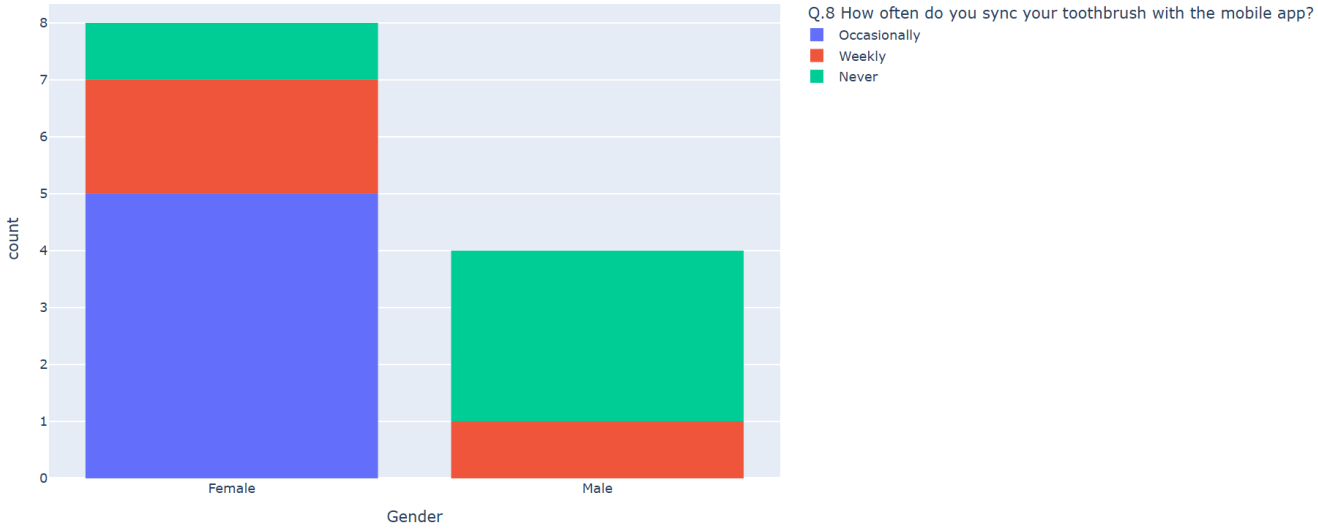
Open the backend.py file in your in-code editor, and edit the following code to edit the code to Gender and Toothbrush app syncing frequency using Histogram:

```
#importing required libraries
import pandas as pd
import plotly.express as px

df = pd.read_csv("results.csv")  #Load the dataset from a CSV file into a DataFrame
#Create a histogram showing the distribution of Gender,
fig = px.histogram(df, x='Gender', color='  Q.8 How often do you sync your toothbrush with the mobile app?  ')

fig.show()# Display the histogram
```

- Use pandas for data handling and plotly express for visualization.
- Load Data: Read the CSV file "results.csv" into a Data Frame.
- Create Histogram: Plot a histogram showing gender on the x-axis, coloured by responses to the toothbrush-app syncing frequency question, and add a title.

- Display Chart: Show the histogram using fig.show

Frequency of Toothbrush App Syncing by Gender:



- The x-axis shows the Gender (Male and Female).
- The y-axis shows the count (number of responses).
- Each bar is color-coded based on responses to the question: "How often do you sync your toothbrush with the mobile app?" The options are:
  - Occasionally
  - Weekly
  - Never

- A larger number of females sync their toothbrush occasionally compared to males.
- Males mostly fall into the "Never" category.
- The distribution of syncing frequency varies significantly between genders.

➢ Add sentiment analysis results as a new column in Google Sheets using Python:

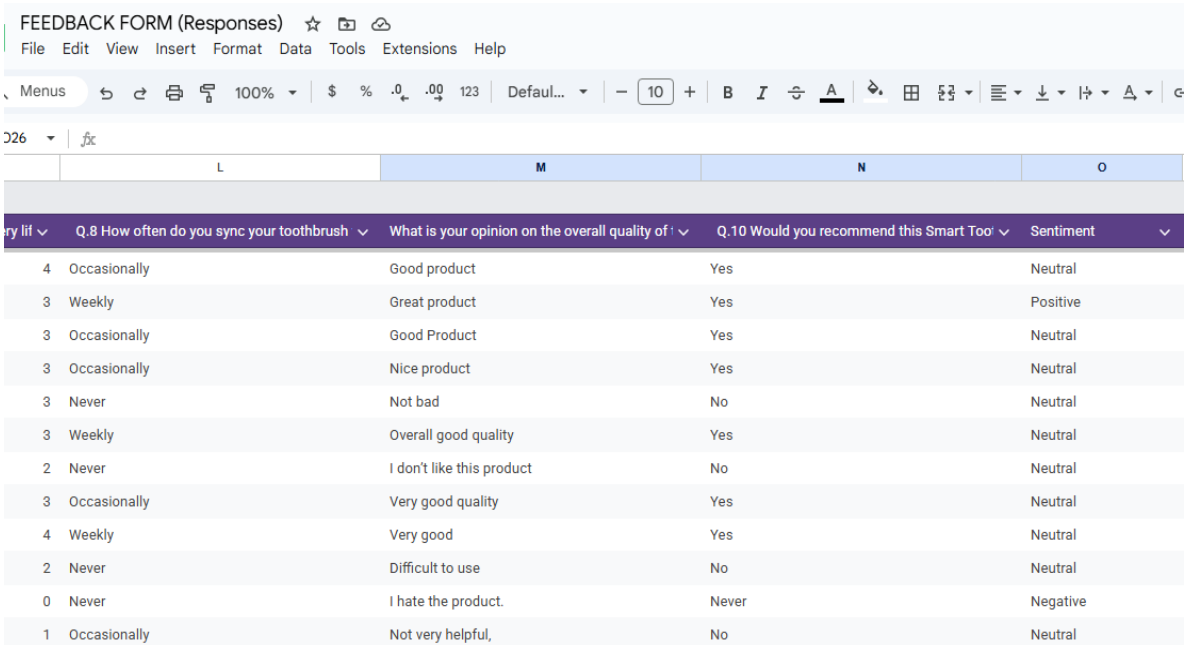
Open the backend.py file in your in-code editor, and add following code to add the column of analysed sentiments in the google sheet:

```
# Import SentimentIntensityAnalyzer for sentiment analysis
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import pandas as pd
# Create an instance of the sentiment analyzer
mymodel = SentimentIntensityAnalyzer()
# Load credentials from a JSON key file and authorize
f = InstalledAppFlow.from_client_secrets_file("key.json", ["https://www.googleapis.com/auth/spreadsheets"])
cred = f.run_local_server(port=0)
# Build the Google Sheets service
service = build("sheets", "v4", credentials=cred).spreadsheets().values()
# Read data from the specified Google Sheet and range
k = service.get(spreadsheetId="13hxHfmWpKNB6PX3s98FripX_dCbhwhTwo6luzQuGqcQ", range="B:N").execute()
d = k['values']
# Convert the retrieved data to a pandas DataFrame
df = pd.DataFrame(data=d[1:], columns=d[0])
# Prepare an empty list for storing sentiment results
l = []
# Perform sentiment analysis on each row in the DataFrame
for i in range(0, len(df)):
    # Get the opinion text from the current row
    t = df._get_value(i, "What is your opinion on the overall quality of the product?")


    # Analyze the sentiment
    pred = mymodel.polarity_scores(t)

    # Classify as Positive, Negative, or Neutral based on compound score
    if pred['compound'] > 0.5:
        d[i+1].append("Positive")
    elif pred['compound'] < -0.5:
        d[i+1].append("Negative")
    else:
        d[i+1].append("Neutral")
# Prepare the updated data in the correct format
h = {'values': d}
# Update the Google Sheet with the sentiment results in column G
service.update(spreadsheetId="13hxHfmWpKNB6PX3s98FripX_dCbhwhTwo6luzQuGqcQ",range="B:O",valueInputOption="USER_ENTERED",body=h).execute()
```

- Import necessary libraries: Import libraries like pandas, google api client, oauthlib, and vader Sentiment.
- Authenticate with Google Sheets API – Use Installed App Flow to load credentials from a key.json file and authorize access.
- Access Google Sheet – Build the Sheets API service and fetch data from a specific range of the sheet.
- Convert data to Data Frame: Use pandas to load the retrieved data into a Data Frame for easier manipulation.

- Initialize sentiment analyser: Create an instance of Sentiment Intensity Analyzer from Vader Sentiment.
- Perform sentiment analysis: Loop through each row, analyse the sentiment of text in a specific column, and classify it as Positive, Negative, or Neutral.
- Store sentiment results: Append the sentiment classification to the corresponding row.
- Update Google Sheet: Add the sentiment analysis results as a new column back into the Google Sheet using the Sheets API.

Added Sentiment Column in google sheet:



ry lif	Q.8 How often do you sync your toothbrush	What is your opinion on the overall quality of	Q.10 Would you recommend this Smart Too	Sentiment
4	Occasionally	Good product	Yes	Neutral
3	Weekly	Great product	Yes	Positive
3	Occasionally	Good Product	Yes	Neutral
3	Occasionally	Nice product	Yes	Neutral
3	Never	Not bad	No	Neutral
3	Weekly	Overall good quality	Yes	Neutral
2	Never	I don't like this product	No	Neutral
3	Occasionally	Very good quality	Yes	Neutral
4	Weekly	Very good	Yes	Neutral
2	Never	Difficult to use	No	Neutral
0	Never	I hate the product.	Never	Negative
1	Occasionally	Not very helpful,	No	Neutral

sentiment analysis results as a new column back into the Google Sheet using the Sheets API.

FRONTEND DEVELOPMENT

Frontend development is essential in a Sentiment Analysis System as it creates an intuitive and interactive user interface for end-users, analysts, and stakeholders. A well-designed frontend enables clear visualization of sentiment trends, real-time text analysis results, and smooth user interaction with the model. It enhances the usability of the system by presenting data insights in an accessible and engaging format.

➢ Frontend in Streamlit:

Streamlit is an open-source Python framework designed for building data-centric web applications with minimal effort. Its simplicity and speed make it ideal for developing interactive tools in Python. In a sentiment analysis system, Streamlit can be used to build a dynamic web interface that allows users to input text, view real-time sentiment predictions, and explore sentiment trends through visual elements like charts and graphs. The streamlined interface helps users easily interpret analysis results, making the tool both powerful and user-friendly.

Install Streamlit:

Go to scripts folder of virtual environment and open command prompt and run the following command:

```
(Sentiment) C:\Projects\Sentiment\Scripts>pip install streamlit
```

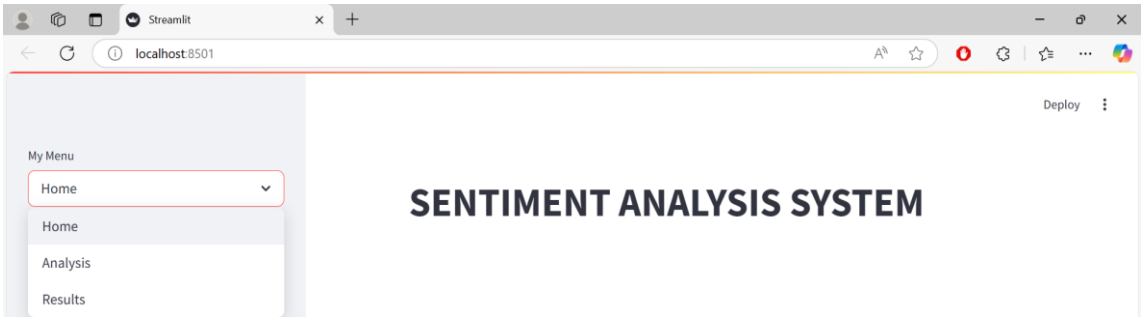
Now after successful installation of streamlit, Open the main.py file in your in-code editor, and add the following basic Streamlit code:

```
import streamlit as st #importing the streamlit
st.title ("SENTIMENT ANALYSIS SYSTEM") #setting the title
choices=st.sidebar.selectbox("My Menu",("Home","Analysis","Results")) #sidebar menu
```

- Imports the Streamlit library
- Sets the app title to "SENTIMENT ANALYSIS SYSTEM".
- Adds a sidebar with a select box labelled "MY MENU".
- The sidebar menu allows navigation between HOME, Analysis, and Result.

Now that you've successfully created your Streamlit app with the title, it opens in the browser and lets you interact with the UI.

Streamlit with title and menu bar:



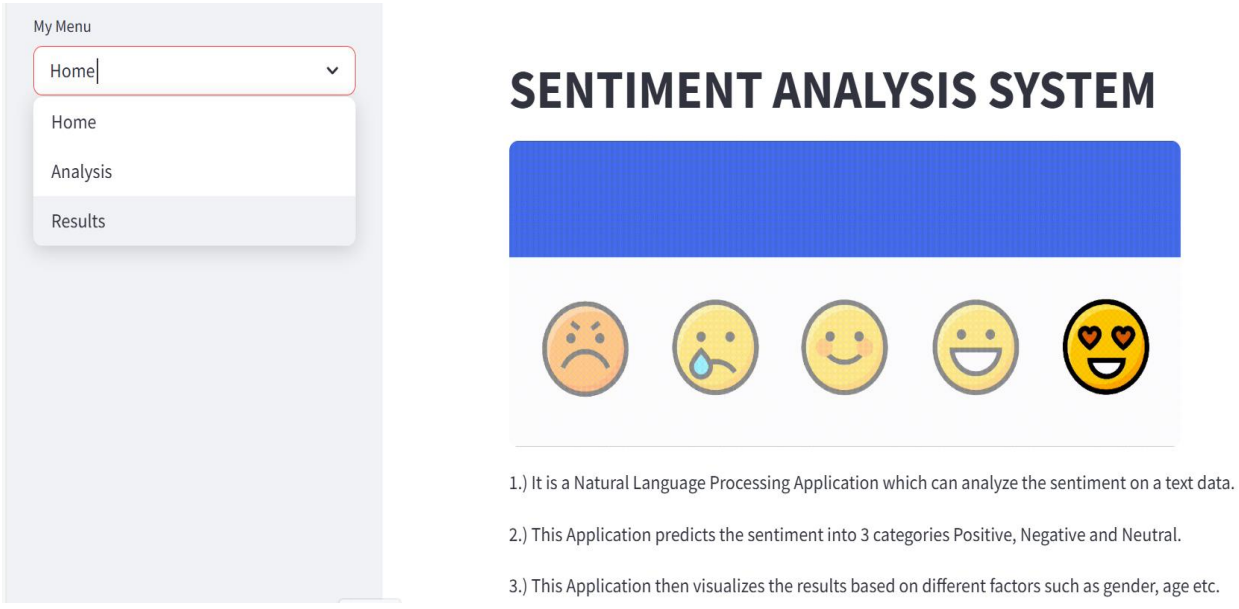
➢ SENTIMENT ANALYSIS SYSTEM – HOME

Open the main.py file in your in-code editor, and edit the following code for Home section:

```
if(choices=="Home"):
    st.image("https://cdn.prod.website-files.com/5ebecording%202021-09-03%20at%2gif", width=600)
    st.write("1.) It is a Natural Language Processing Application which can analyze the sentiment on a text data.")
    st.write("2.) This Application predicts the sentiment into 3 categories Positive, Negative and Neutral.")
    st.write("3.) This Application then visualizes the results based on different factors such as gender, age etc.")
```

- Check if the user has selected the "Home" option from the sidebar menu.
- Display an image from an online URL with a specified width for resizing.
- Show descriptive text below the image to explain the purpose and features of the Sentiment Analysis Application.

Home section in streamlit:



- **Image Display**: A colourful image with different emoji faces representing various emotions is shown, symbolizing the sentiment categories (e.g., happy, sad, neutral, excited).
- **Textual Description**: Below the image, three descriptive points explain the functionality of the system.

➢ SENTIMENT ANALYSIS SYSTEM – ANALYSIS:

This section of the application allows users to perform real-time sentiment analysis on text data stored in a Google Sheet. By entering the Sheet ID, the range of data, and selecting the specific column containing the textual input, users can trigger the analysis process with a single click.

The system uses Natural Language Processing (NLP) techniques to evaluate the emotional tone of the text. It classifies each entry into one of three categories:

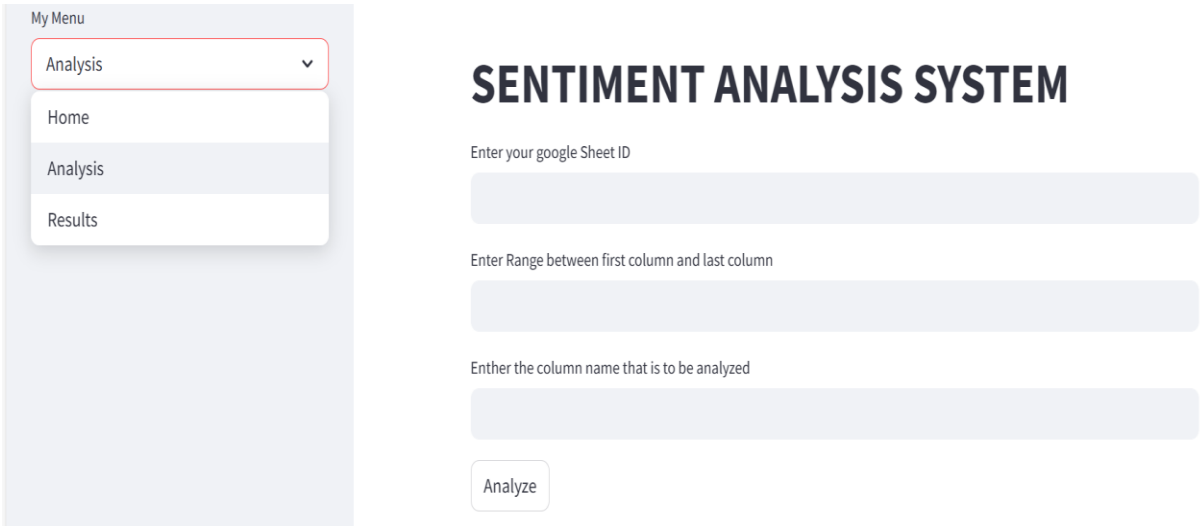
- Positive
- Negative
- Neutral

Open the main.py file in your in-code editor, and edit the following code for analysing the google sheet data:

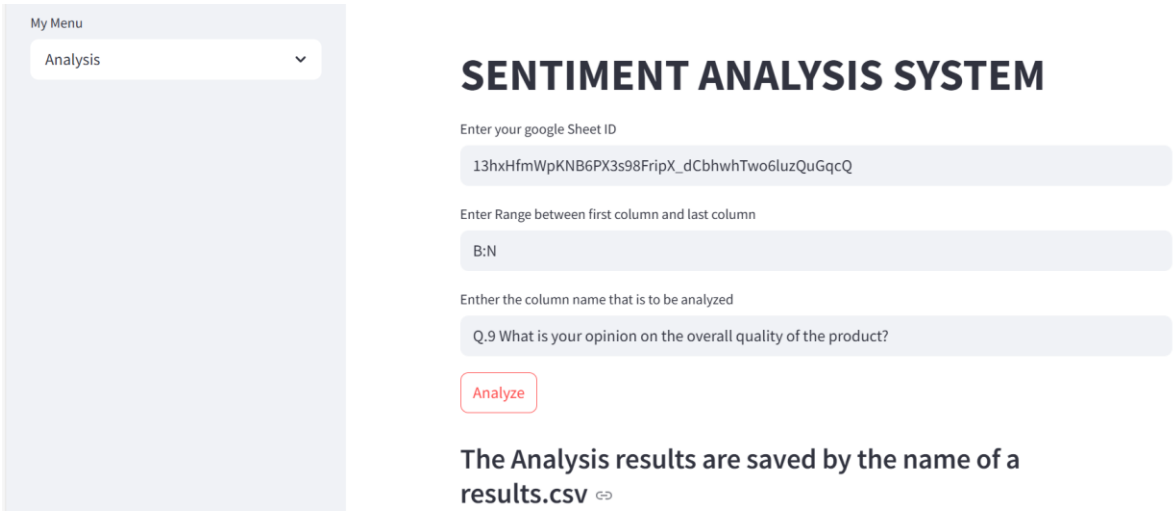
```
    # Analysis page content
elif(choices=="Analysis"):
     # User input for Google Sheet details
    gsid=st.text_input("Enter your google Sheet ID")
    r=st.text_input("Enter Range between first column and last column")
    c=st.text_input("Enther the column name that is to be analyzed")
    btn=st.button("Analyze")
    # When Analyze button is clicked
    if btn:
         # Check if credentials already exist in session state
        if 'cred' not in st.session_state:
            # Load credentials from the key.json file with access to Google Sheets
            f=InstalledAppFlow.from_client_secrets_file("key.json",["https://www.googleapis.com/auth/spreadsheets"])
            # Run the local server and store credentials
            st.session_state['cred']=f.run_local_server(port=0)
         # Initialize sentiment analysis model
        mymodel=SentimentIntensityAnalyzer()
         # Build the Google Sheets API service
        service=build("Sheets","v4",credentials=st.session_state['cred']).spreadsheets().values()
             # Fetch data from Google Sheet
        k-service.get(spreadsheetId=sid, ranger=r).excecute()
        d=k['values']
        df=pd.DataFrame(data=d[1:],columns=d[0]) # Convert sheet data to DataFrame
        l=[]
        for i in range(0,len(df)):# Iterate over each row and perform sentiment analysis
            t=df._get_value(i,c) # Extract text from specified column
            pred=mymodel.polarity_scores(t) # Predict sentiment
            if(pred['compound']>0.5):  # Classify sentiment based on compound score
                l.append("Posittive")
            elif(pred["compound"]<-0.5):
                l.append("Negative")
            else:
                l.append("Neutral")
        # Add the sentiment results to the DataFrame
        df['Sentiment']=l
        df.to_csv("results.csv",index=False)# Save the results to a CSV file
        st.subheader("The Analysis results are saved by the name of a results.csv")# Show message to user
```

- User Selects "Analyse" from Sidebar Menu: When the user clicks on the "Analysis" option in the sidebar, the system switches to analysis mode.
- User Enters Required Inputs: The app prompts the user to:
    - Enter the Google Sheet ID (where the data is stored).
    - Specify the range (from first to last column).
    - Mention the column name that contains the text data to be analysed.
- User Clicks "Analyse" Button: The process begins when the user clicks the "Analyse" button.
- Google Sheets Authorization Check: The app checks if Google Sheets API credentials already exist in session. If not, it:
    - Loads them from a key.json file.
    - Authenticates using the Google OAuth system.
    - Stores credentials in session for future use.
- Sentiment Analyzer Initialization: A VADER Sentiment Analyzer is initialized to evaluate the sentiment of text data.
- Connect to Google Sheets & Fetch Data: The app connects to the Google Sheet using the provided ID and range. It fetches the values from the sheet.
- Convert Sheet Data to a Data Frame: The raw data is transformed into a pandas Data Frame for analysis.
- Analyse Sentiment for Each Text Entry: For each row in the selected column. The text is extracted. Sentiment is predicted (positive, negative, or neutral), The decision is based on the compound score:
    - > 0.5 → Positive
    - < -0.5 → Negative
    - Else → Neutral
- Store Sentiment Results: The sentiment labels are added as a new column in the Data Frame.
- Export Results: The Data Frame (with sentiment results) is saved as a CSV file named results.csv.
- Notify User: The app displays a message confirming that the results have been saved.

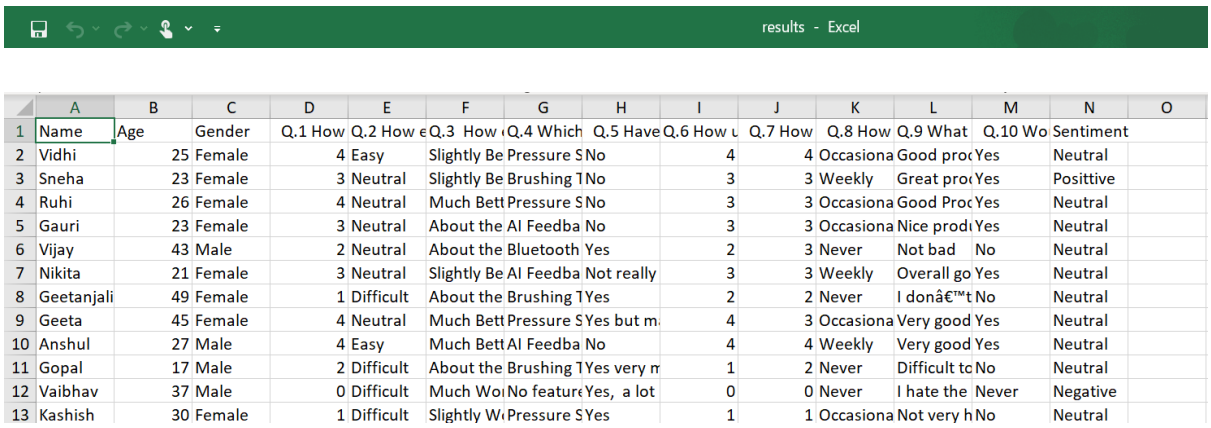
Analysis section in streamlit:



Fill the following information:



Csv file saved in scripts folder of virtual Environment:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Age	Gender	Q.1 How	Q.2 How e	Q.3 How	Q.4 Which	Q.5 Have	Q.6 How u	Q.7 How	Q.8 How	Q.9 What	Q.10 Wo	Sentiment	
2	Vidhi	25	Female	4	Easy	Slightly Be	Pressure S	No	4	4	Occasiona	Good proc	Yes	Neutral	
3	Sneha	23	Female	3	Neutral	Slightly Be	Brushing T	No	3	3	Weekly	Great proc	Yes	Posittive	
4	Ruhi	26	Female	4	Neutral	Much Bett	Pressure S	No	3	3	Occasiona	Good Proc	Yes	Neutral	
5	Gauri	23	Female	3	Neutral	About the	AI Feedba	No	3	3	Occasiona	Nice prod	Yes	Neutral	
6	Vijay	43	Male	2	Neutral	About the	Bluetooth	Yes	2	3	Never	Not bad	No	Neutral	
7	Nikita	21	Female	3	Neutral	Slightly Be	AI Feedba	Not really	3	3	Weekly	Overall go	Yes	Neutral	
8	Geetanjali	49	Female	1	Difficult	About the	Brushing T	Yes	2	2	Never	I donâ€™t	No	Neutral	
9	Geeta	45	Female	4	Neutral	Much Bett	Pressure S	Yes but m	4	3	Occasiona	Very good	Yes	Neutral	
10	Anshul	27	Male	4	Easy	Much Bett	AI Feedba	No	4	4	Weekly	Very good	Yes	Neutral	
11	Gopal	17	Male	2	Difficult	About the	Brushing T	Yes very m	1	2	Never	Difficult to	No	Neutral	
12	Vaibhav	37	Male	0	Difficult	Much Wo	No feature	Yes, a lot	0	0	Never	I hate the	Never	Negative	
13	Kashish	30	Female	1	Difficult	Slightly W	Pressure S	Yes	1	1	Occasiona	Not very h	No	Neutral	

- Google Sheet ID: This is the ID of the Google Sheet containing the responses.
- Column Range: Specifies the range of columns (from column B to N) that contains the data.
- Target Column Name: Example - Q.9 What is your opinion on the overall quality of the product? This is the question whose responses will be analysed for sentiment.
- When the user clicks the "Analyze" button, the system:
- Fetches the data from the specified Google Sheet and column range.
- Extracts the responses under the given question.
- Uses a sentiment analysis model (likely VADER based on your earlier code) to evaluate each response.
- Saves the sentiment results (like Positive, Negative, or Neutral) into a results.csv.

➢ SENTIMENT ANALYSIS SYSTEM – RESULTS:

To better understand the emotional tone of user-generated content, we performed sentiment analysis on a dataset collected from the google form responses (Feedback on Smart Toothbrush). The results have been visually represented to highlight the distribution and intensity of sentiments expressed.

The results of the sentiment analysis can be effectively visualized using various graphical methods:

- Pie Chart – to show the overall distribution of positive, negative, and neutral sentiments.
- Histogram – to represent the frequency of sentiment polarity scores.
- Boxplot – to analyse the spread and outliers in sentiment intensity.
- Line Graph – to observe sentiment trends over time.
- Bar Chart – to compare sentiment across different categories or topics.

These visualizations provide a clear and concise view of sentiment data, highlighting key trends. By organizing the information visually, they enable quick pattern recognition. As a result, actionable insights can be easily drawn for decision-making.

Open the main.py file in your in-code editor, and edit the following code for visualising the sentiment Analysis:

```
elif(choices=="Results"): # Results section with visualizations
    df=pd.read_csv("results.csv")   # Load the saved results
     # Allow user to choose type of visualization
    choice2=st.selectbox("Choose Visualisation",("None","Pie Chart","Histogram","Box Plot"))
    st.dataframe(df)   # Display the full DataFrame
```

- User chooses "Results": The app checks if the user selected the "Results" option from the main menu.
- Data is loaded: It opens a previously saved CSV file (usually containing sentiment analysis results).
- User selects a chart type: A dropdown menu appears where the user can pick how they want to visualize the data (Pie Chart, Histogram, Box Plot, or None).
- Display the data: The entire dataset is shown in a neat table format on the app screen for the user to view.

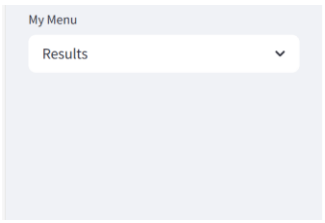
Pie chart:

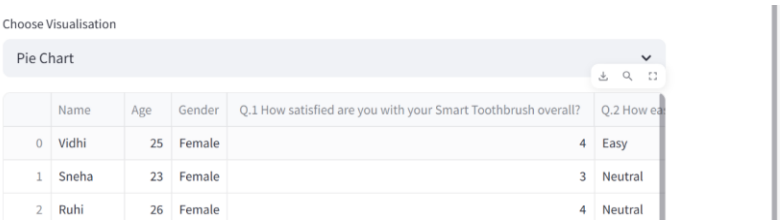
Open the main.py file in your in-code editor, and edit the following code for visualising the distribution of sentiment Analysis by using Pie chart:

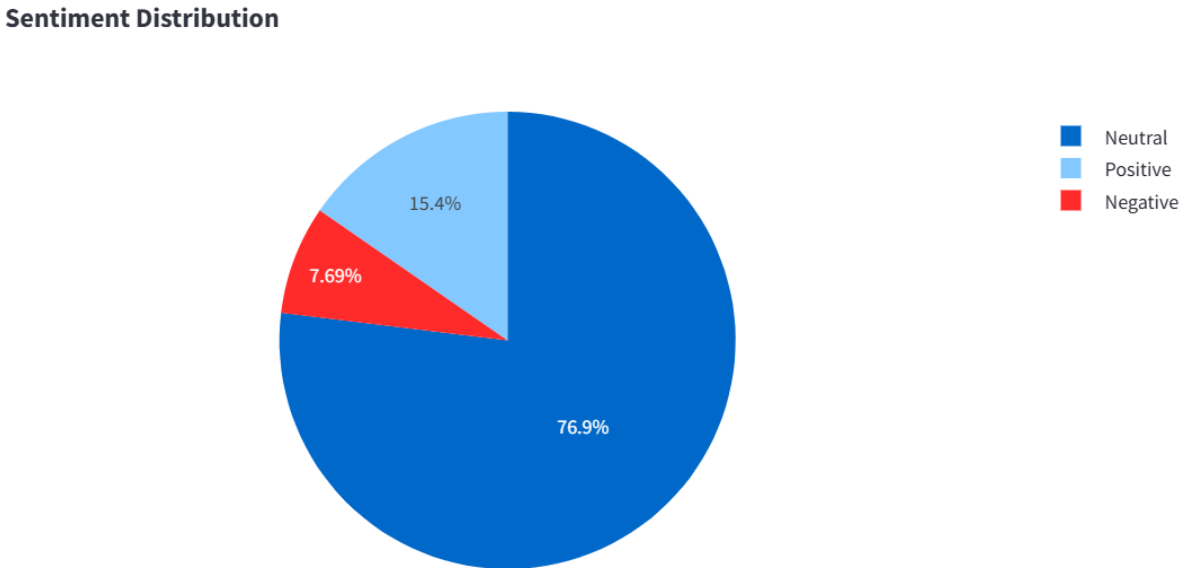
```
if(choice2=="Pie Chart"):   # Pie Chart for Sentiment Distribution
    posper = (len(df[df['Sentiment'] == 'Positive']) / len(df)) * 100
    negper = (len(df[df['Sentiment'] == 'Negative']) / len(df)) * 100
    neuper = (len(df[df['Sentiment'] == 'Neutral']) / len(df)) * 100
    # Create pie chart
    fig=px.pie(values=[posper,negper,neuper],names=['Positive','Negative','Neutral'],title='Sentiment Distribution')
    st.plotly_chart(fig)
```

- Calculate the Percentage of Each Sentiment: First, calculate the percentage of each sentiment (Positive, Negative, and Neutral) by dividing the count of each sentiment by the total number of entries in the dataset.
- Prepare the Data for the Pie Chart: Prepare a list with the calculated percentages for Positive, Negative, and Neutral sentiments.
- Create the Pie Chart: Use a plotting library (like Plotly) to create a pie chart with the calculated percentages and labels for each sentiment category (Positive, Negative, Neutral).
- Display the Chart: Finally, render the pie chart to display it in your application or interface.

Final Output:







The final output is a pie chart titled *"Sentiment Distribution"*, which visually represents the proportion of different sentiments in the dataset. The chart shows:

- Neutral sentiment dominates the data, making up 76.9% of the total.
- Positive sentiment follows at 15.4%**,** represented in light blue.
- Negative sentiment is the smallest category, at 7.69%**,** shown in red.

This visual summary helps in quickly understanding the overall emotional tone of the data, making it easier to draw conclusions and guide decision-making.

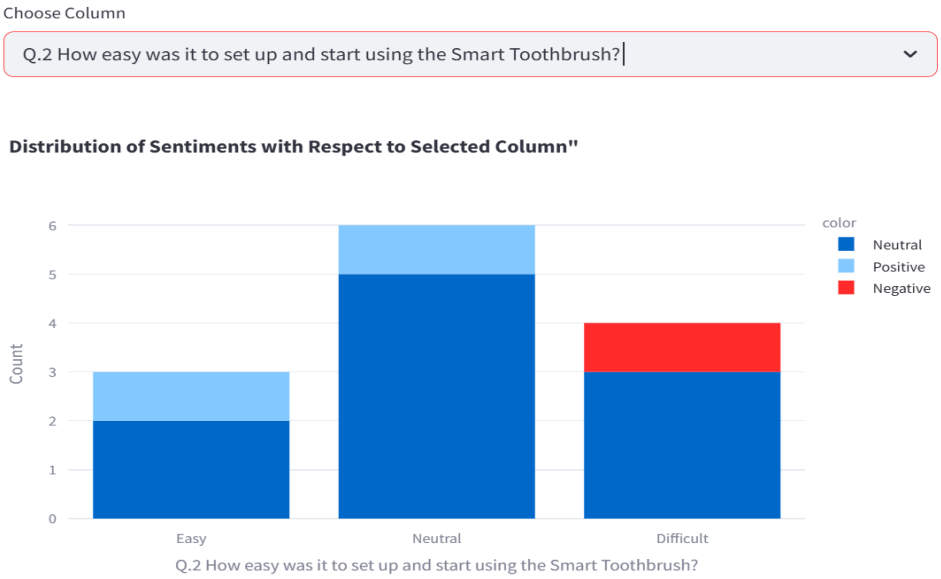
Histogram:

Open the main.py file in your in-code editor, and edit the following code for visualising the distribution of sentiment with respect to selected column by using Histogram:

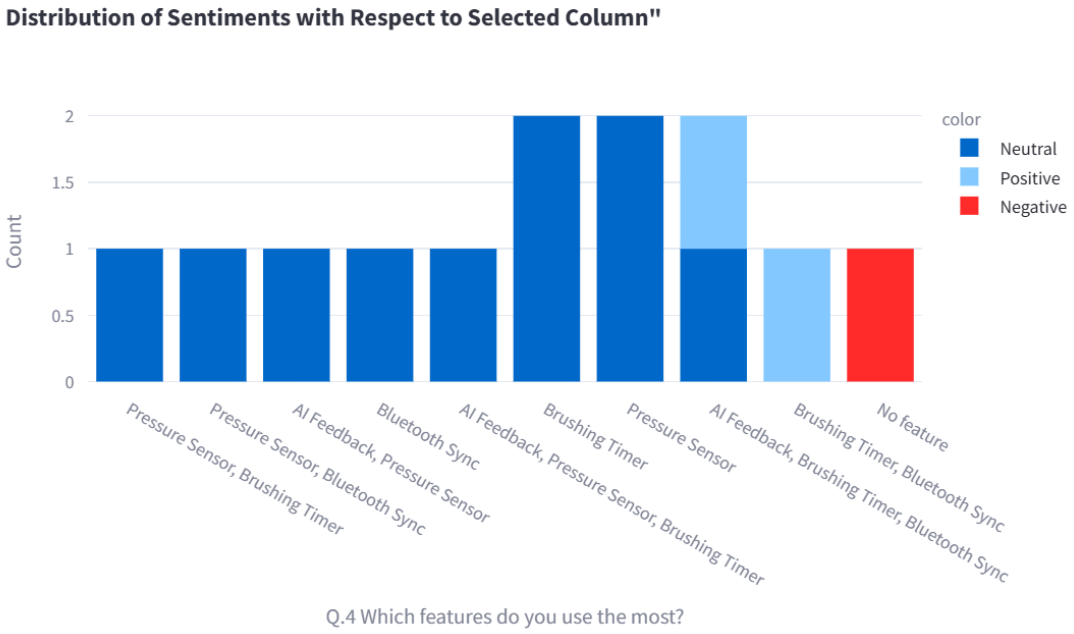
```
#user selects Histogram
elif(choice2=="Histogram"):
    p=st.selectbox("Choose Column",df.columns)  # Histogram for selected column
    if p: # Create histogram with sentiment coloring
        fig=px.histogram(x=df[p],color=df['Sentiment'],title='Distribution of Sentiments with Respect to Selected Column"')
        fig.update_layout(xaxis_title=p, yaxis_title="Count")
        st.plotly_chart(fig)#Display the histogram
```

- User Chooses Histogram Option: The system checks if the user has selected "Histogram" as the visualization choice.
- Column Selection: A dropdown (select box) is displayed for the user to choose a specific column from the dataset for which the histogram will be created.
- Generate Histogram: If a column is selected, a histogram is generated using the selected column values. The bars in the histogram are color-coded based on sentiment (e.g., Positive, Negative, Neutral).
- Customize and Display the Chart: The histogram's x-axis is labelled with the selected column name, and the y-axis is labelled as "Count". The plot is then rendered using Plotly and displayed to the user.

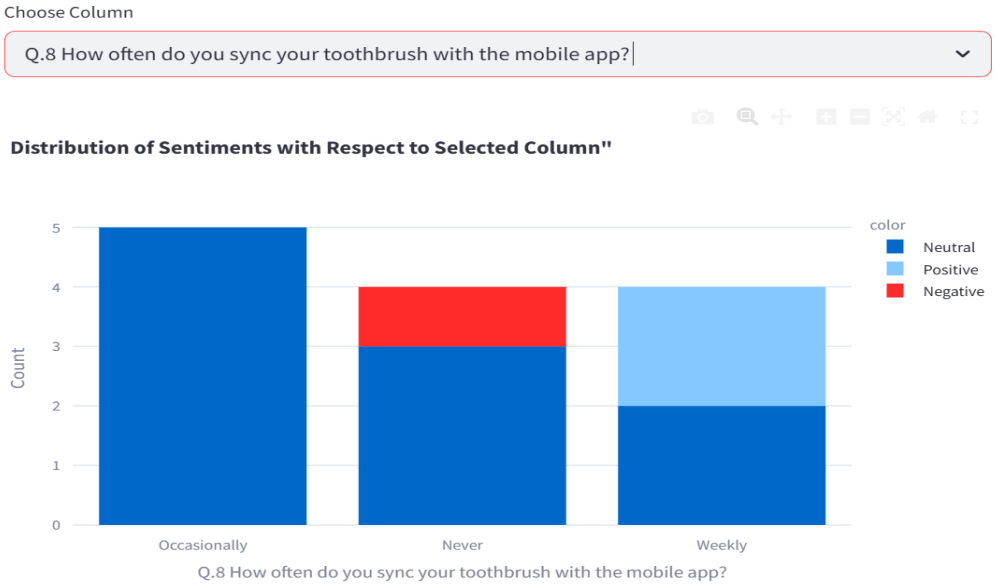
Final Output:



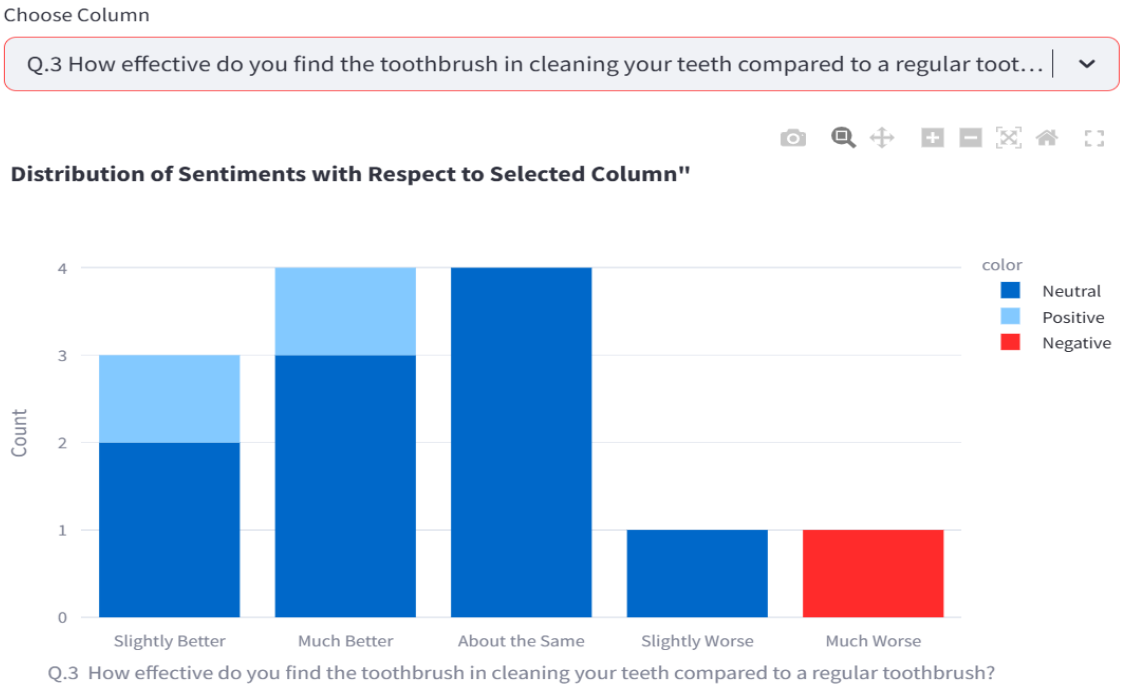
- The chart shows most users felt neutral about setting up the Smart Toothbrush, with some finding it difficult and expressing negative sentiment.



- Most users showed neutral or positive sentiment towards multiple smart toothbrush features, while “No feature” usage had the only negative response.



- Most users sync their toothbrush with the app "Occasionally" or "Weekly" with neutral to positive sentiments, while "Never" showed some negative feedback.



- The chart shows that most users found the smart toothbrush to be either "Much Better" or "About the Same" as a regular toothbrush, with predominantly neutral or positive sentiments. Negative feedback was limited to those who rated it "Much Worse."

Boxplot:

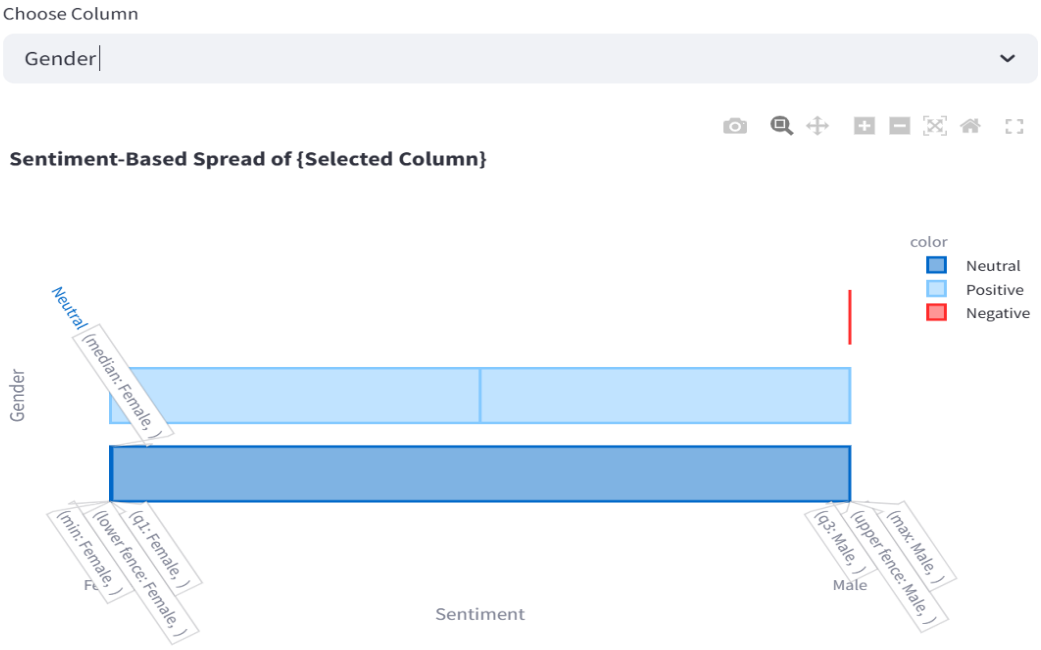
Open the main.py file in your in-code editor, and edit the following code for visualising the sentiment-based spread of selected column by using Boxplot:

```
 # Box Plot for selected column
elif (choice2 == "Box Plot"):
     # Display a dropdown for the user to choose a column from the dataset
   l = st.selectbox("Choose Column", df.columns)
   if l:
       fig = px.box(x=df[l], color=df['Sentiment'],title='Sentiment-Based Spread of {Selected Column}')
       fig.update_layout(xaxis_title="Sentiment",yaxis_title=l)
       st.plotly_chart(fig)   # Display the box plot
```

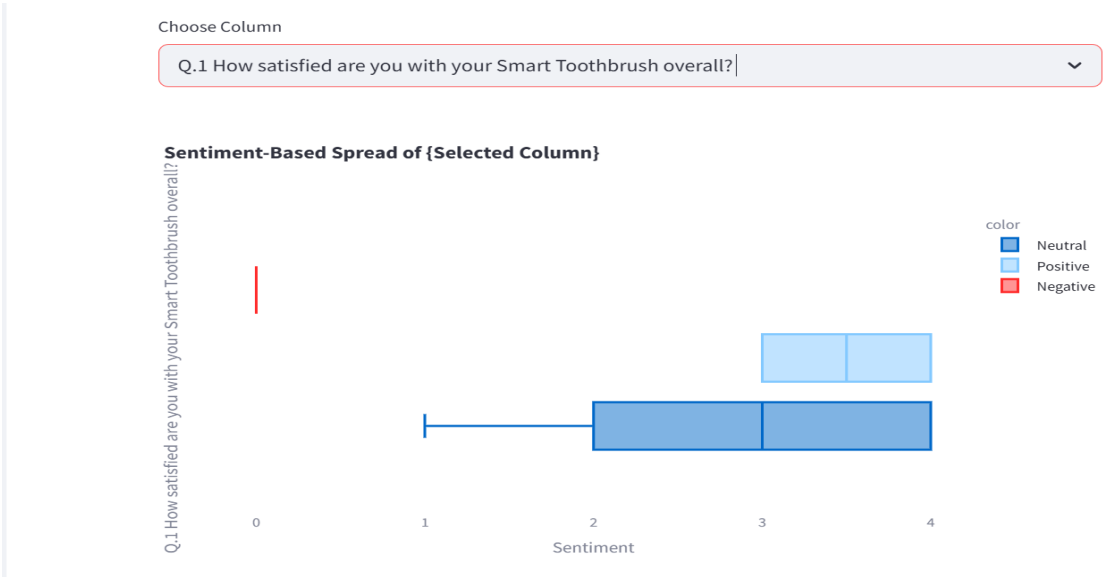
- Check for Box Plot Selection: The process starts only if the user selects the "Box Plot" option.
- Display a Dropdown: A dropdown menu is shown to the user, allowing them to select a column from the dataset.
- Check User Selection: It proceeds only if the user has selected a column from the dropdown.
- Generate Box Plot: A box plot is created using the selected column, with colours representing different sentiment categories.
- Customize Plot Layout: The X-axis is labelled as "Sentiment" and the Y-axis is labelled using the selected column's name.
- Display the Chart: Finally, the generated box plot is displayed on the interface.

Final Output:

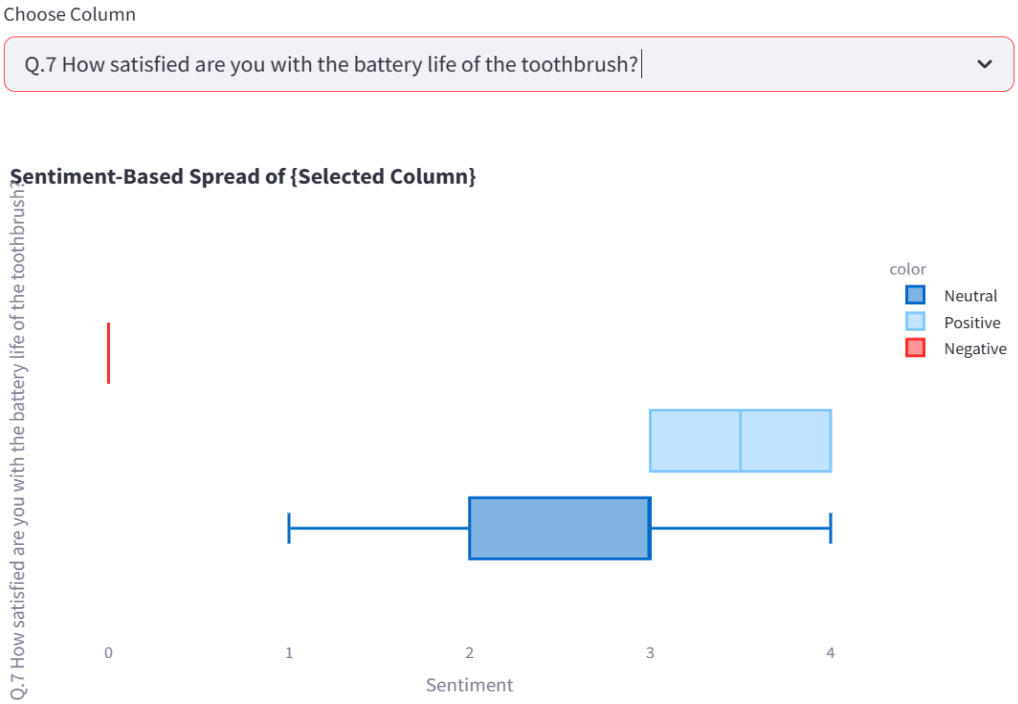
- The box plot displays the spread of user sentiments across different age groups. Neutral and Positive sentiments are mostly from younger users, while a Negative sentiment appears as an outlier in the older age range.



- The chart indicates that both males and females mostly expressed neutral or positive sentiments. However, one negative sentiment was recorded, likely from a male user.



- box plot shows that most users who were satisfied with their smart toothbrush had Neutral or Positive sentiments. A single Negative sentiment appears as an outlier linked to very low satisfaction.



- The box plot shows that most users were either neutral or positive about the battery life of the toothbrush. A single negative sentiment appears as a clear outlier, indicating dissatisfaction from one user.

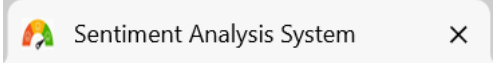
➢ To change the icon and title in Streamlit:

To change the title and icon in Streamlit, first set the title for the browser tab. Then, specify an icon to appear in the tab, which can either be an emoji or an image.

Open the main.py file in your in-code editor, and add the following Streamlit code and put the icon link.

```
# Set the title and icon of the Streamlit web page
st.set_page_config(page_title="Sentiment Analysis System", page_icon="")
```

This code changed the icon and also the update the title with “Sentiment Analysis System”.



SUMMARY OF PROJECT

Summary of Achievement:

We have successfully designed and developed a Sentiment Analysis System that meets the specified objectives and incorporates key features for effective sentiment classification. The main highlights of our application include:

- **User-Friendly Interface:** The system offers a clean and intuitive interface that allows users to input text and view sentiment results with ease, requiring minimal technical skills.
- **Real-Time Sentiment Detection:** It analyses input text in real-time, classifying it as positive, negative, or neutral using machine learning-based natural language processing techniques.
- **Data Logging for Analysis:** The application stores user inputs and corresponding sentiment results in a dedicated database or file system, enabling further review or trend analysis.

Difficulties Encountered During the Project:

- **Learning Curve:** Gaining a deep understanding of NLP techniques, text pre-processing, and training sentiment models posed an initial challenge and required significant research and experimentation.
- **Model Integration:** Integrating pre-trained models and ensuring they function efficiently within the application required debugging, performance tuning, and handling edge cases in input text.

Limitations of the Project:

- **Language Dependency:** The current system is optimized for English text only and may struggle with regional languages or mixed-language input.
- **Context Sensitivity:** The model may misclassify sarcastic, ambiguous, or context-heavy sentences due to limited contextual understanding.

Future Scope of the Project:

- **Advanced NLP Models:** Incorporating more powerful models like BERT or GPT for deeper context comprehension and higher accuracy.
- **Multilingual Support & Voice Input:** Adding support for other languages and voice-to-text capabilities can broaden usability across diverse user groups.

With these enhancements, the Sentiment Analysis System can evolve into a comprehensive tool for customer feedback analysis, social media monitoring, and emotional tone detection across platforms.

CONCLUSION

The Sentiment Analysis System is an intelligent, real-time text classification application developed using Python, Natural Language Processing (NLP) libraries such as NLTK and a user-friendly Streamlit interface. It allows users to input text and instantly determines whether the sentiment is positive, negative, or neutral using a pre-trained machine learning model. The system provides clear visual feedback and stores the analysed results for future review, enabling efficient tracking of user sentiment over time.

A standout feature of the system is its ability to save sentiment data for continuous improvement and trend analysis, making it useful for businesses monitoring customer feedback or public opinion. The frontend is built using Streamlit, offering a clean layout with real-time sentiment detection, easy text input, and seamless output display, making it accessible for users with little to no technical background.

While the system performs its core functions effectively, it has strong potential for future enhancements. Possible improvements include multilingual sentiment detection, integration with voice input, cloud-based storage for large-scale analytics, and advanced models like BERT for deeper context understanding. These additions would make the system more robust, versatile, and suitable for applications in marketing, customer service, social media monitoring, and beyond.