Sofware Requirements Specification

Sentiment Analysis

Srivalli Nalla, Zane Oliver, Hayden Banks, Jared Zutt, Cara Burgan

Table of Contents

1. Introduction

1.1 Purpose

*The purpose of this document is to outline the design and architecture of the **Sentiment Analysis Tool**, a Python-based application that analyzes textual and numerical data to determine sentiment polarity (Positive, Neutral, Negative). The tool provides visualizations (pie charts, bar charts, scatter plots, word clouds) and generates automated summaries with recommendations based on sentiment trends.*
This application is designed for:

- ***Business analysts** assessing customer feedback.*
- ***HR professionals** evaluating employee satisfaction.*
- ***Researchers** conducting sentiment analysis on survey data.*
- ***General users** need quick sentiment insights from textual or numerical datasets.*

1.2 System Overview

The Sentiment Analysis Tool is a **desktop application** *built using:*

- ***Python 3.10+** (Core programming language)*
- ***Tkinter** (GUI framework)*
- ***TextBlob** (Sentiment analysis)*
- ***Pandas & NumPy** (Data processing)*
- ***Matplotlib & Seaborn** (Data visualization)*
- ***WordCloud** (Text visualization)*
- ***Scikit-learn** (Numeric scaling)*

Key Features:

1. ***Multi-Source Data Input:***
   a. *Pre-loaded sample datasets.*
   b. *CSV file uploads.*
   c. *Manual text input.*
2. ***Automated Sentiment Detection:***
   a. *Text analysis via **TextBlob** (polarity scoring).*
   b. *Numeric rating classification (scaled sentiment).*
3. ***Interactive Visualizations:***
   a. *Pie charts (sentiment distribution).*
   b. *Scatter plots (polarity/rating trends).*
   c. *Bar charts (sentiment frequency).*
   d. *Word clouds (key terms in Positive/Negative text).*
4. ***Summary Reports:***
   a. *Predefined insights based on sentiment distribution.*

b. *Actionable recommendations for different sentiment scenarios.*

1.3 Design Map

*The system follows a **modular architecture**:*

1. **Frontend (GUI):**
    a. *Built with **Tkinter** for cross-platform compatibility.*
    b. *Notebook-style tabs for organizing visualizations.*
    c. *Dynamic controls for data source selection.*
2. **Backend (Core Logic):**
    a. ***Data Processing:** Handles CSV, manual text, and sample data.*
    b. *Sentiment Analysis:*
        i. *Textual: Uses TextBlob for polarity scoring.*
        ii. *Numeric: Scales ratings (-1 to 1) and classifies sentiment.*
    c. ***Visualization Engine:** Generates plots via Matplotlib.*
3. **Output:**
    a. *Interactive charts embedded in the GUI.*
    b. *Textual summary with recommendations.*

1.4 Definitions and Acronyms

Term	**Definition**
Sentiment Polarity	*Score (-1 to 1) indicating negative/neutral/positive tone in text.*
TextBlob	*Python library for NLP-based sentiment analysis.*
Pandas	*Data manipulation library for structured datasets.*
Tkinter	*Standard Python GUI toolkit for desktop applications.*
MinMaxScaler	*Scikit-learn tool for normalizing numeric data to a fixed range (-1 to 1).*
WordCloud	*Visualization technique highlighting frequent terms in text data.*

2. Design Considerations

Design Considerations were done in an agile project work development. We initially went with a web-based application for our work-life balance sentiment analysis project; we then integrated this into a standalone application. Issues included fully integrating from web-based to standalone, finding the right graphs that would accurately visualize our data, and having our sentiment values portrayed through user selected columns provided by a csv file.

2.1 Assumptions

We assume that end users (HR) will supply employee survey data in a clean CSV or Excel and will have basic knowledge of their machine and how to select these files.

2.2 Constraints

Library Limitations

- *TextBlob sentiment is rule-based (polarity threshold ±0.1); it may struggle with domain-specific banking terminology.*
- *We restricted ourselves to lightweight models (no BERT/LSTM) to keep dependencies small and inference fast.*

Time Limitations

- *We ran into some time constraints early on in development due to busy schedules*
- *We rectified this by using Jira to assign specific work and set short daily meetings and longer ones once a week.*

2.3 System Environment

Our sentiment analysis app is built on python 3.10 and open-source Python libraries.

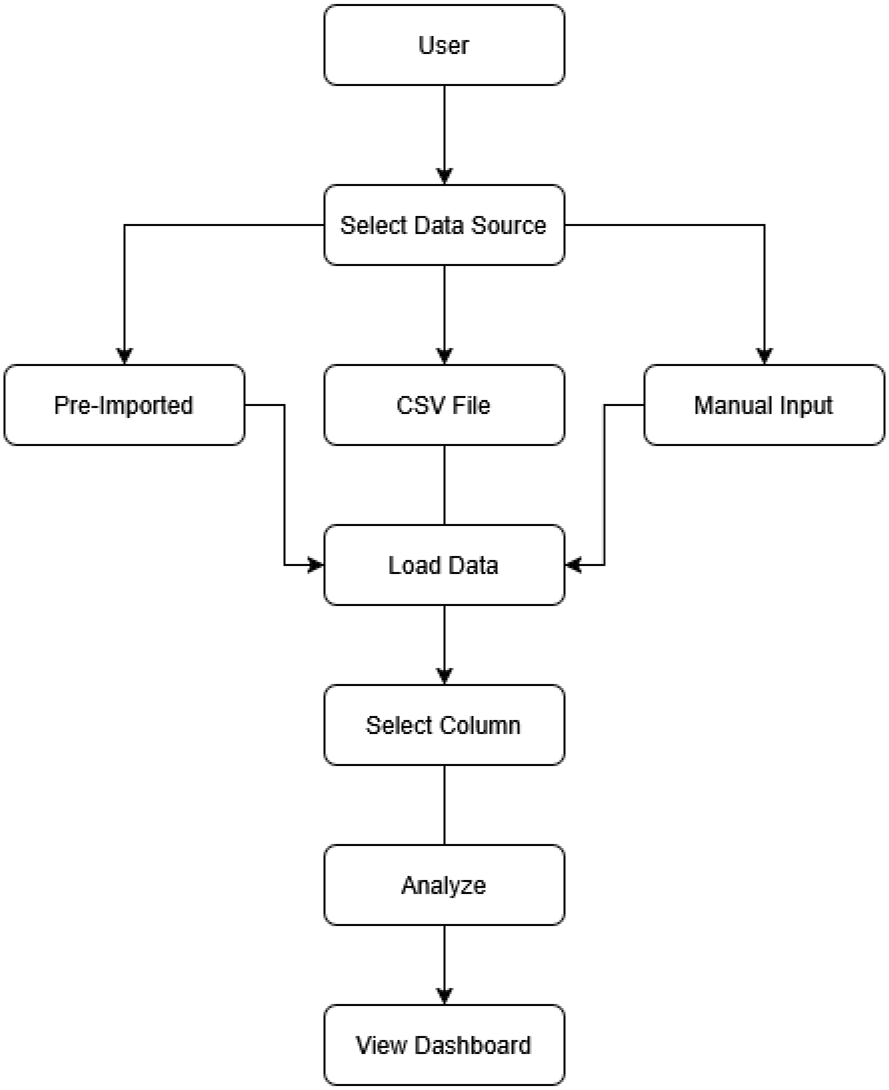
2.4 Risk and Volatile Areas

Performance
Large CSVs will slow down real-time plotting and UI interactivity; may require down-sampling or batch processing.

User

Users should select columns related to work-life balance or they might run into inaccurate results.

3. High Level Design



Users will begin by selecting a data source from the radio buttons at the top of the interface.

If the "Pre-Imported" option is selected, users will press the "Load Data" button to load the built-in dataset.

If the "CSV File" option is selected, users will press the "Load Data" button to browse and import a .csv file.

If the "Manual Input" option is selected, users will type or paste data into the provided text box and then press "Load Data" to process the input.

Once data is loaded, users will choose a column for sentiment analysis using the dropdown menu labeled "Select Column."

After selecting the column, users will press the "Analyze" button to generate results.

Users will then view the sentiment analysis dashboard, which includes charts such as pie charts, scatter plots, and word clouds organized in tabbed views.

4. Design Considerations

4.1 Data Sources

The application supports analyzing data from three sources: pre-loaded, CSV files, and manual input. The pre-loaded data is hard coded into an array. Whichever data source chosen is loaded into a pandas dataframe.

4.2 Analysis Functions

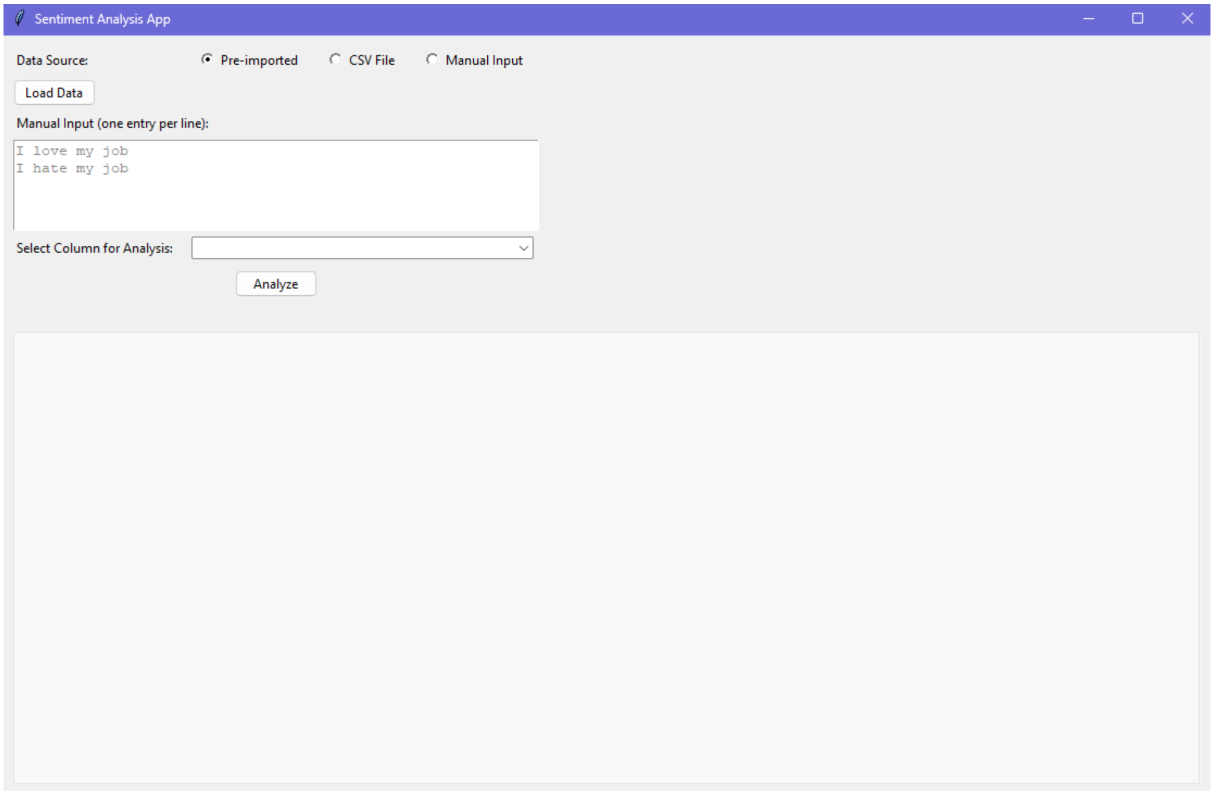
The application can analyze several aspects of the loaded data. For text data, it can analyze sentiment using the TextBlob polarity property, returning a positive or negative value representing positive and negative sentiment respectively. For numeric data, it can calculate descriptive statistics such as the average, median, min, and max. The numeric data can further be elaborated by a summary generator. It generates the summary using a predefined template and hard coded sentences depending on the average rating value.

5. User Interface Design

The Sentiment Analysis App allows users to input either text or numbers manually or a CSV file of their choice. This stand-alone app was designed to have a simple and intuitive interface that users can understand easily.

5.1 Screen 1

The home screen of The Sentiment Analysis App consists of three options to select a data source. The first option is to use a small pre-imported data set that gives users an example of what the capabilities of the software would look like. The second option is to input a CSV file that can be read by selecting a column for analysis below. Last, there is an option to input data manually in a line-by-line format. This data can be typed into the text box below. Placeholder lines are shown to give the user an example of the proper formatting. After inputting the desired data to be analyzed, the analyze button can be pressed to finish the process.



Data Source: ● Pre-imported ○ CSV File ○ Manual Input

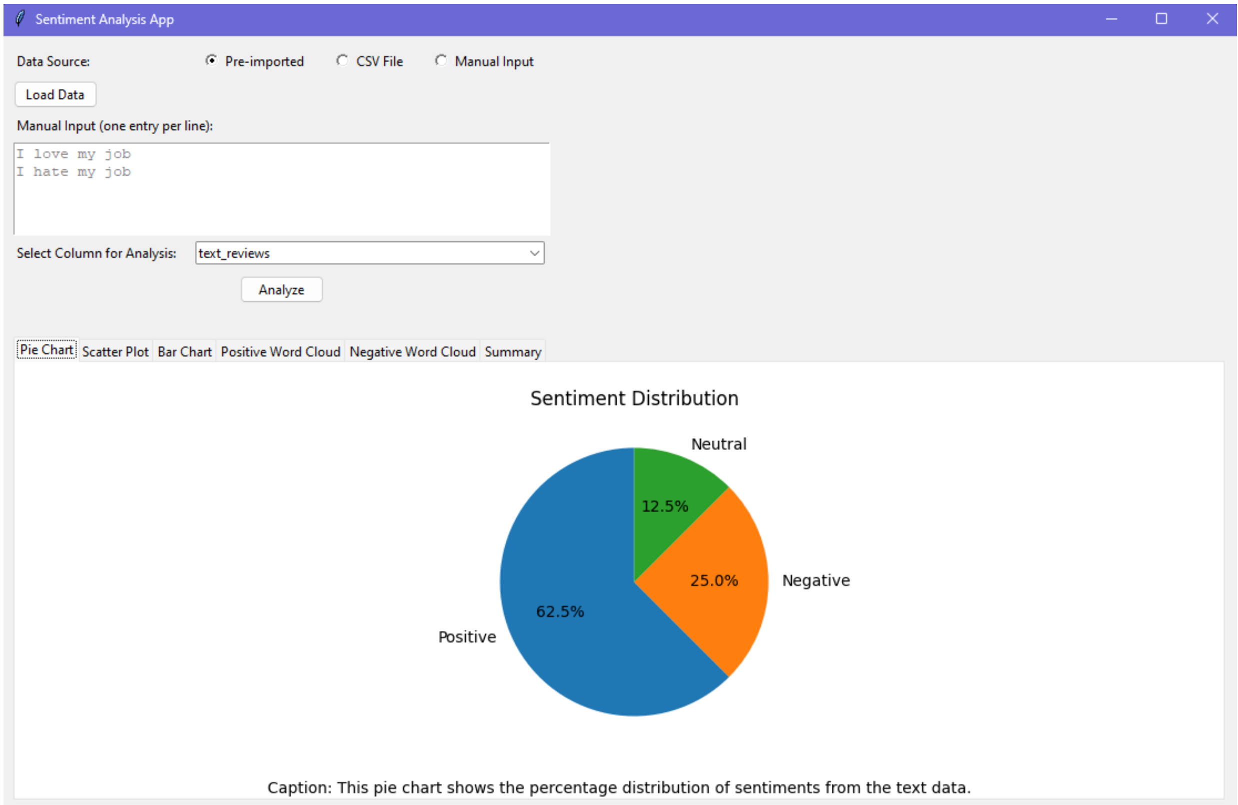
Manual Input (one entry per line):

```
I love my job
I hate my job
```

Select Column for Analysis:

5.2 Screen 2

After the analyze button is pressed, the results will populate in the white space below. For the first method of analysis, a pie chart is used. Percentages and their keys are indicated within the graph, while a more detailed caption of what the pie chart shows is below. Examples of the pie chart for a text analysis and a numerical analysis are included below.



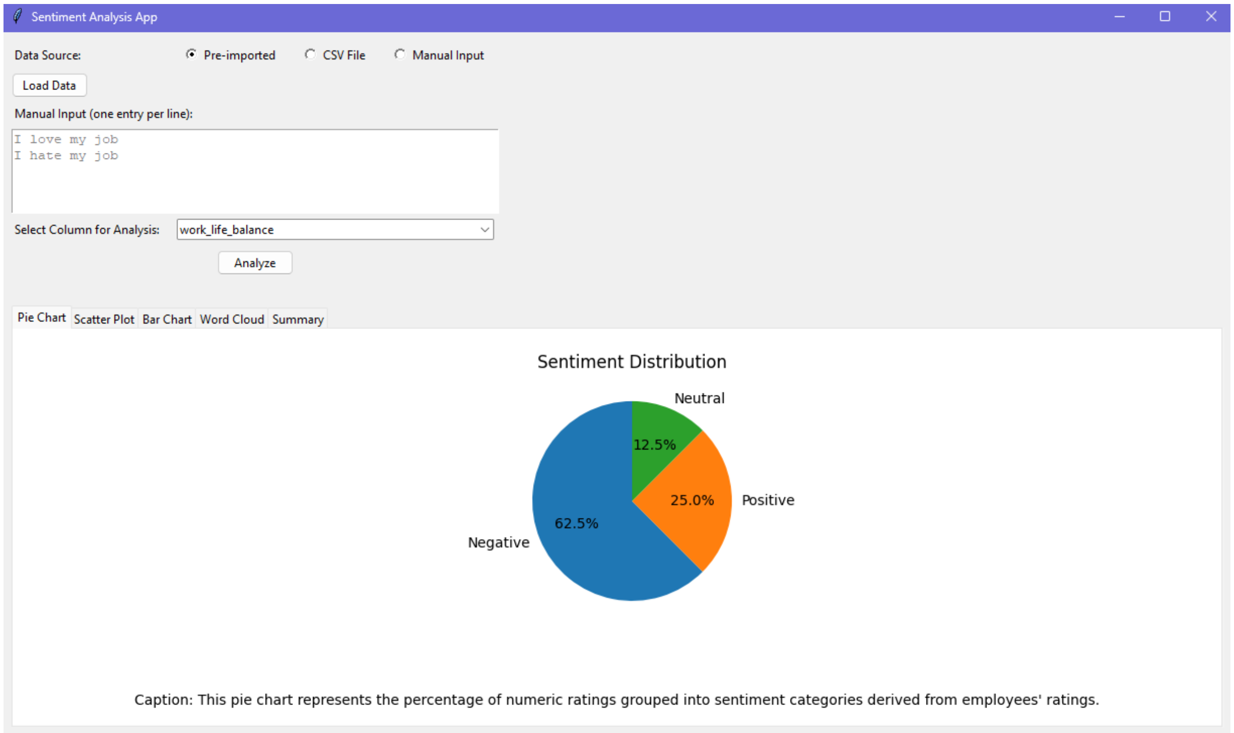
Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

```
I love my job
I hate my job
```

Select Column for Analysis: text_reviews

Caption: This pie chart shows the percentage distribution of sentiments from the text data.



Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

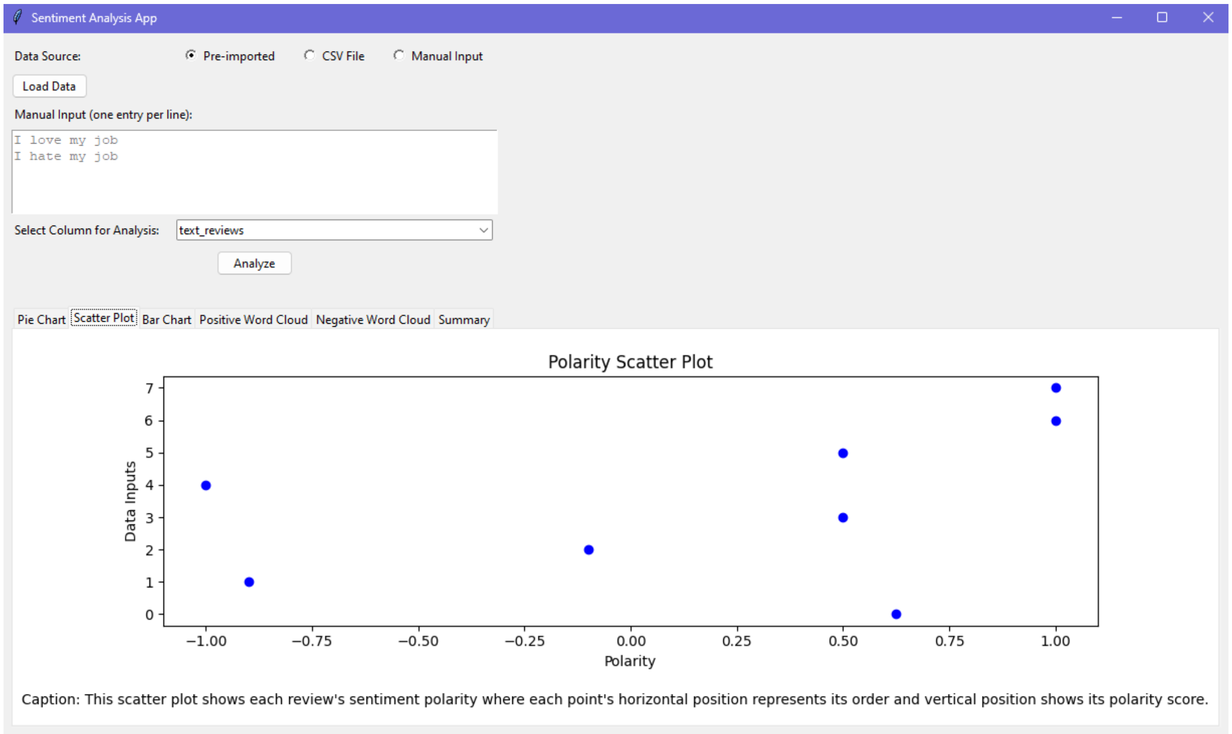
```
I love my job
I hate my job
```

Select Column for Analysis: work_life_balance

Caption: This pie chart represents the percentage of numeric ratings grouped into sentiment categories derived from employees' ratings.

5.3 Screen 2.1

When pressing the scatter plot tab, a polarity scatter plot then becomes visible within the white space. The plots and keys are indicated within the graph and a more detailed caption of what the scatter plot shows is below.



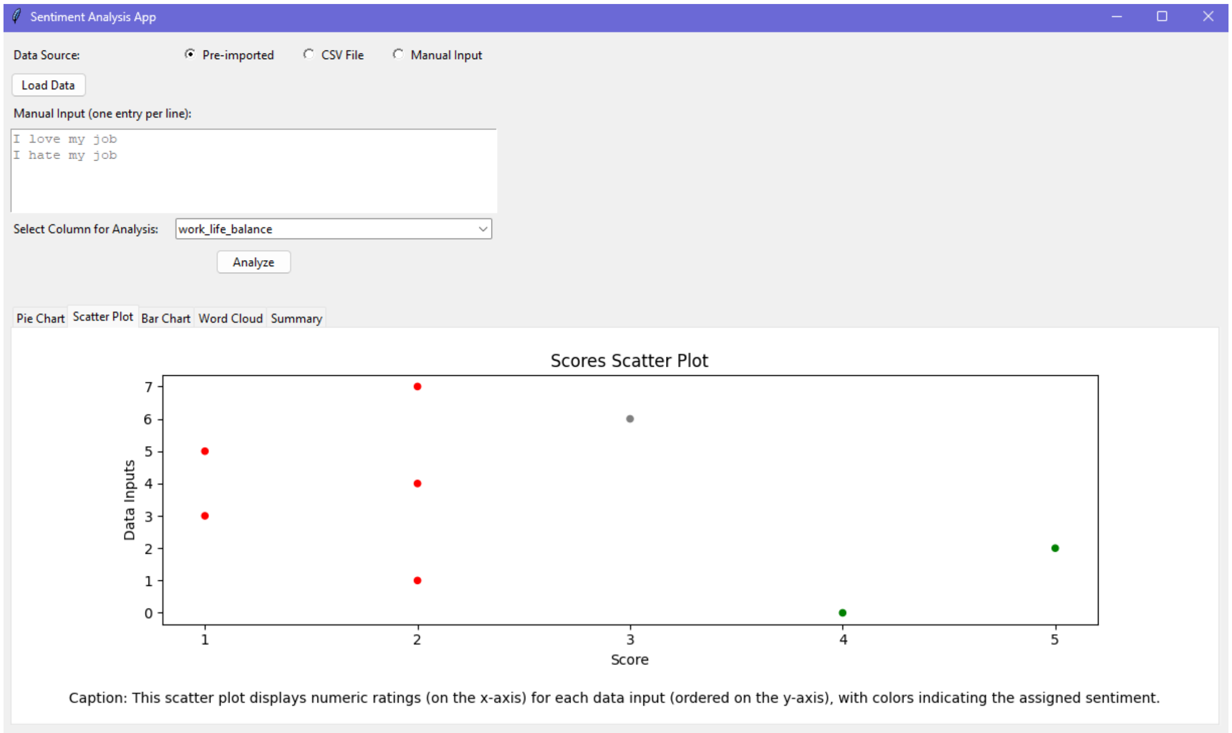
Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

```
I love my job
I hate my job
```

Select Column for Analysis: text_reviews

Caption: This scatter plot shows each review's sentiment polarity where each point's horizontal position represents its order and vertical position shows its polarity score.



Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

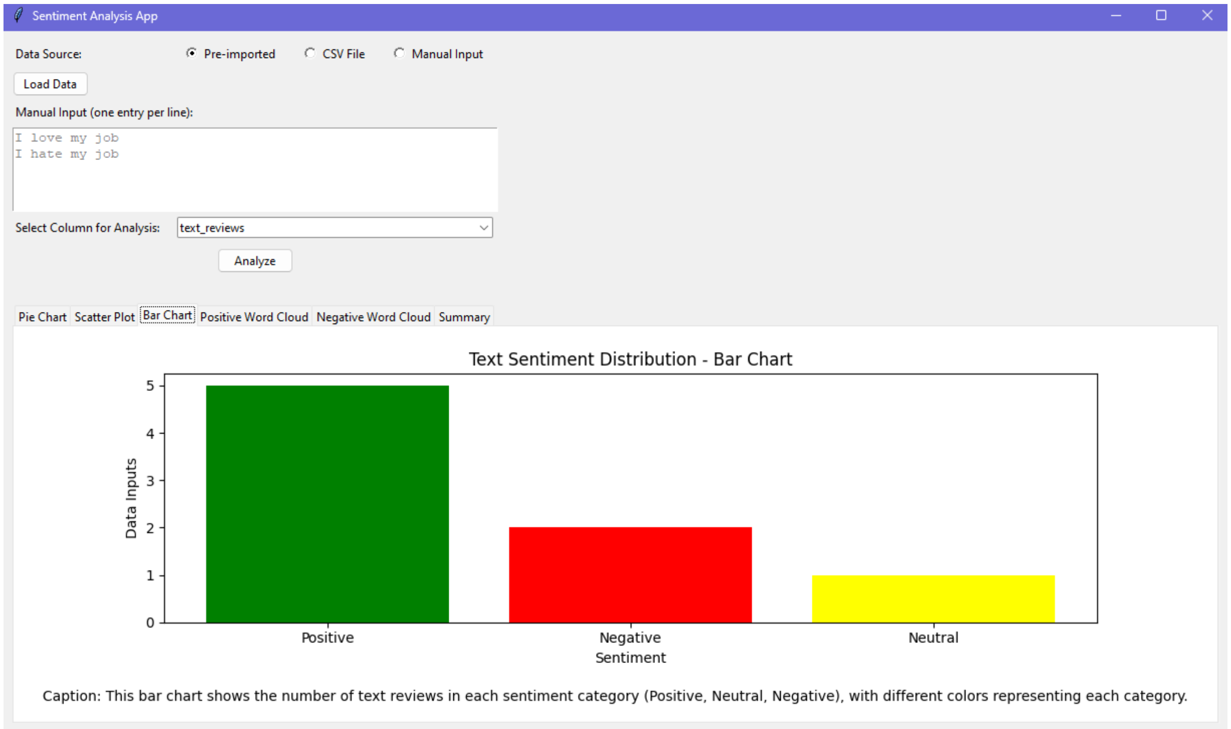
```
I love my job
I hate my job
```

Select Column for Analysis: work_life_balance

Caption: This scatter plot displays numeric ratings (on the x-axis) for each data input (ordered on the y-axis), with colors indicating the assigned sentiment.

5.4 Screen 2.2

When pressing the bar chart tab, a sentiment bar chart then becomes visible within the white space. The bars and keys are indicated within the graph and a more detailed caption of what the bar chart shows is below.



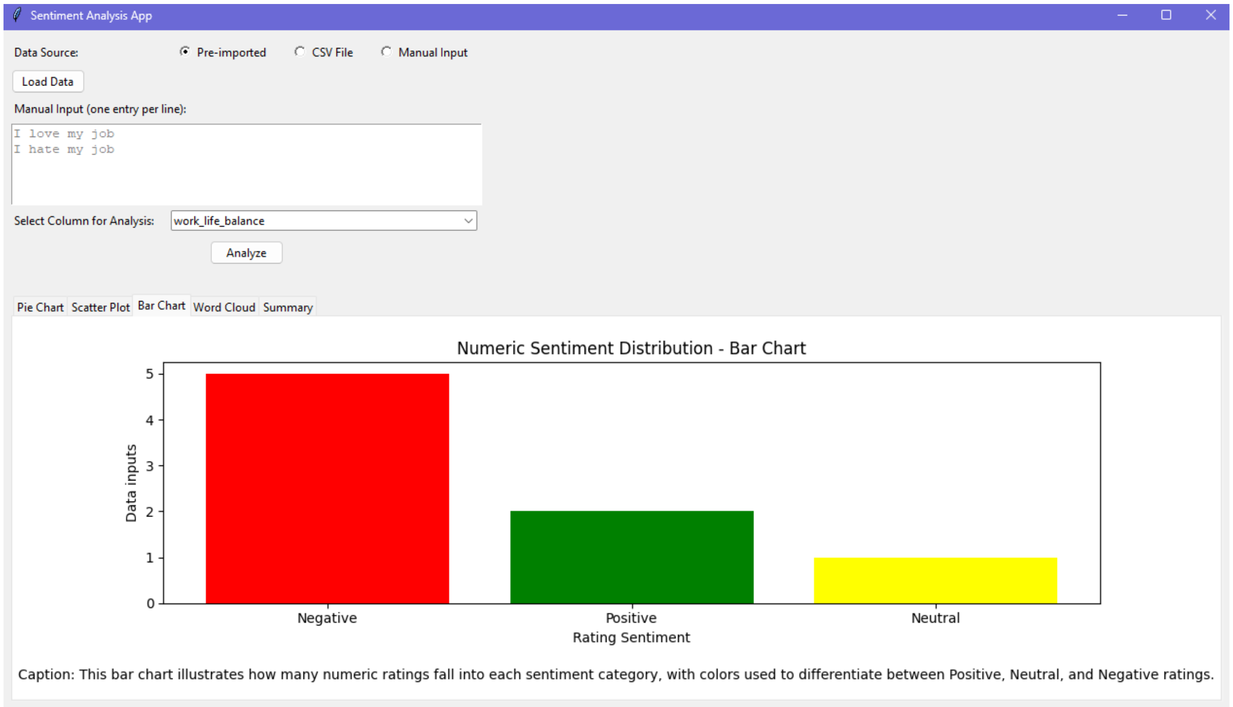
Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

```
I love my job
I hate my job
```

Select Column for Analysis: text_reviews

Caption: This bar chart shows the number of text reviews in each sentiment category (Positive, Neutral, Negative), with different colors representing each category.



Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

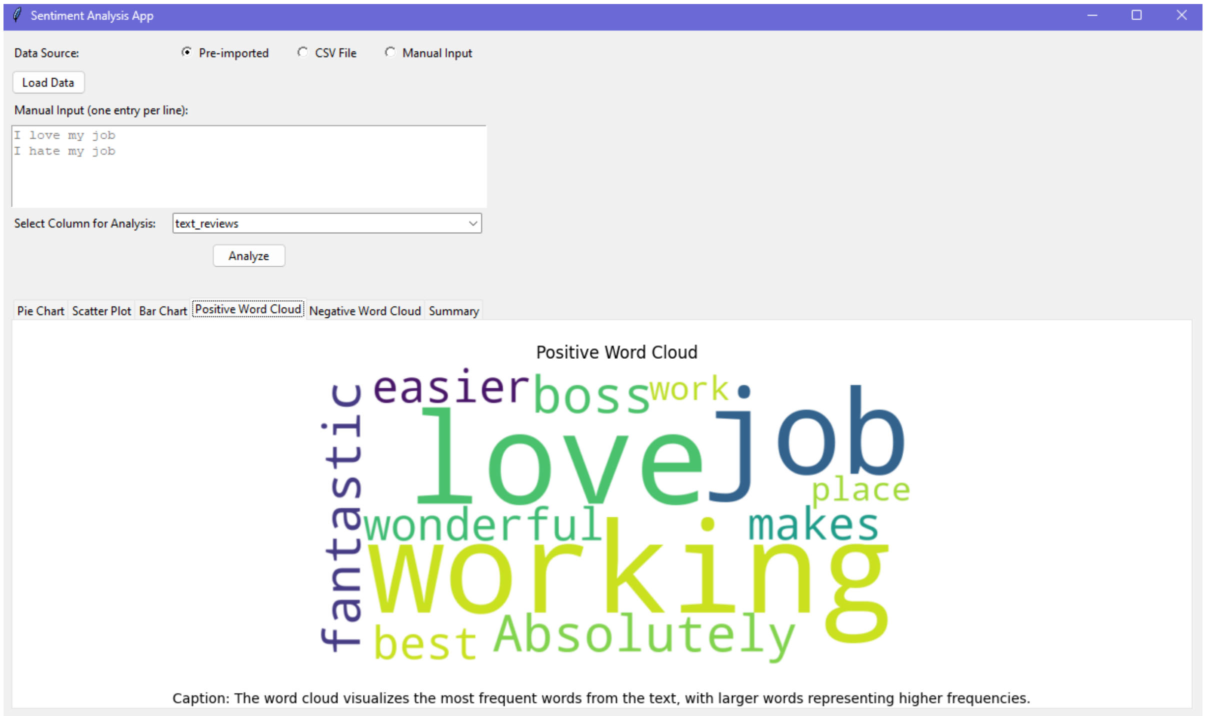
```
I love my job
I hate my job
```

Select Column for Analysis: work_life_balance

Caption: This bar chart illustrates how many numeric ratings fall into each sentiment category, with colors used to differentiate between Positive, Neutral, and Negative ratings.

5.5 Screen 2.3

When pressing the positive word cloud tab, a positive word cloud then becomes visible within the white space. The words are boxed in at different sizes within the graph and a more detailed caption of what the word cloud shows is below.



Sentiment Analysis App

Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

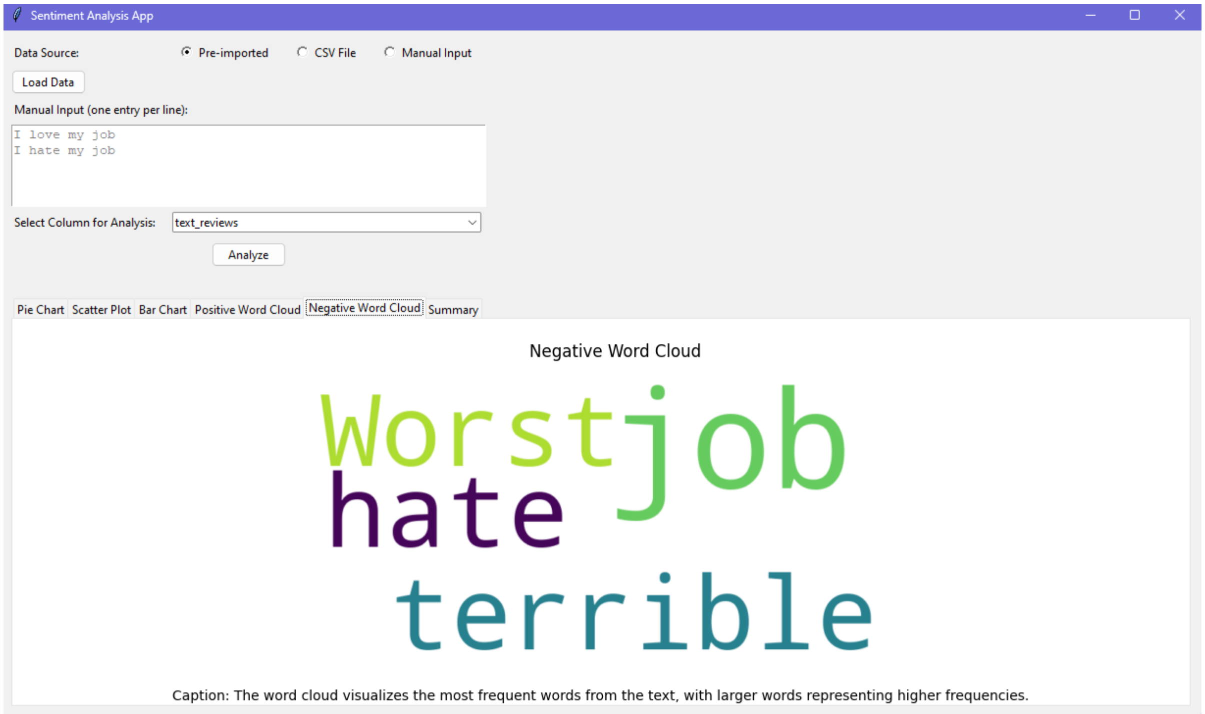
```
I love my job
I hate my job
```

Select Column for Analysis: text_reviews

Caption: The word cloud visualizes the most frequent words from the text, with larger words representing higher frequencies.

5.6 Screen 2.4

When pressing the negative word cloud tab, a negative word cloud then becomes visible within the white space. The words are boxed in at different sizes within the graph and a more detailed caption of what the word cloud shows is below



Sentiment Analysis App

Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

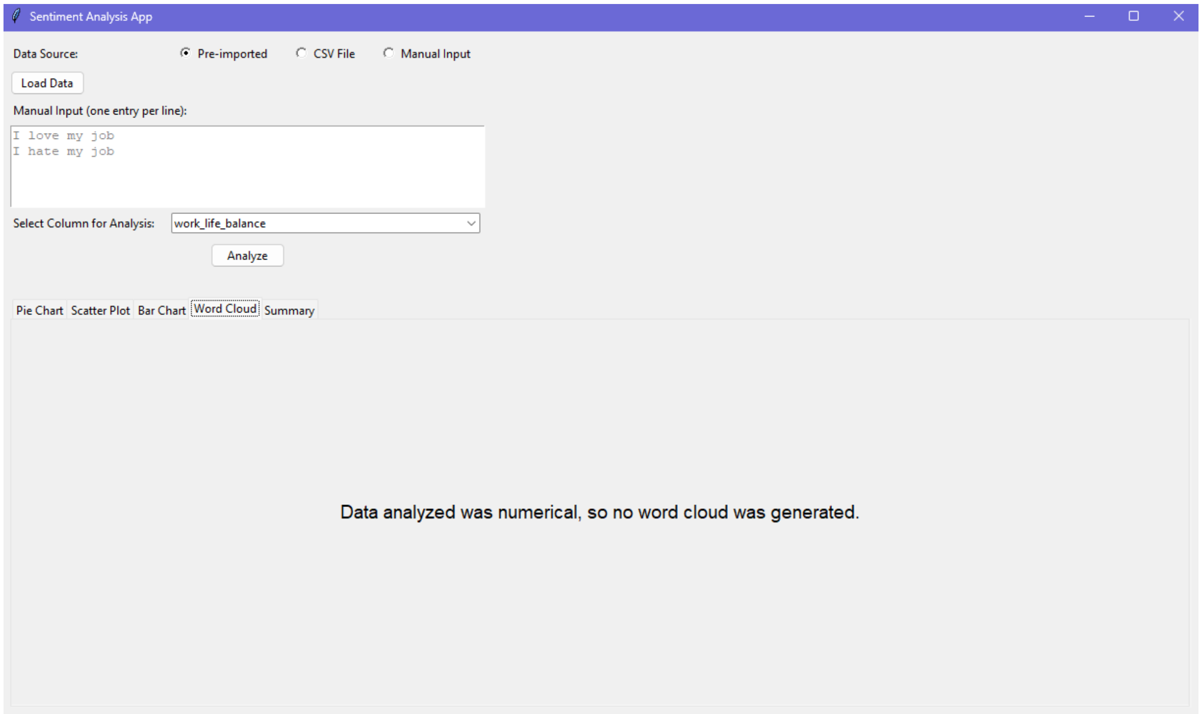
```
I love my job
I hate my job
```

Select Column for Analysis: text_reviews

Caption: The word cloud visualizes the most frequent words from the text, with larger words representing higher frequencies.

5.7 Screen 2.5

If the data analyzed does not include words, the tabs of the positive and negative word clouds will merge into one word cloud tab. Within the white space of this tab, it states "data analyzed was numerical, so no word cloud was generated."



Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

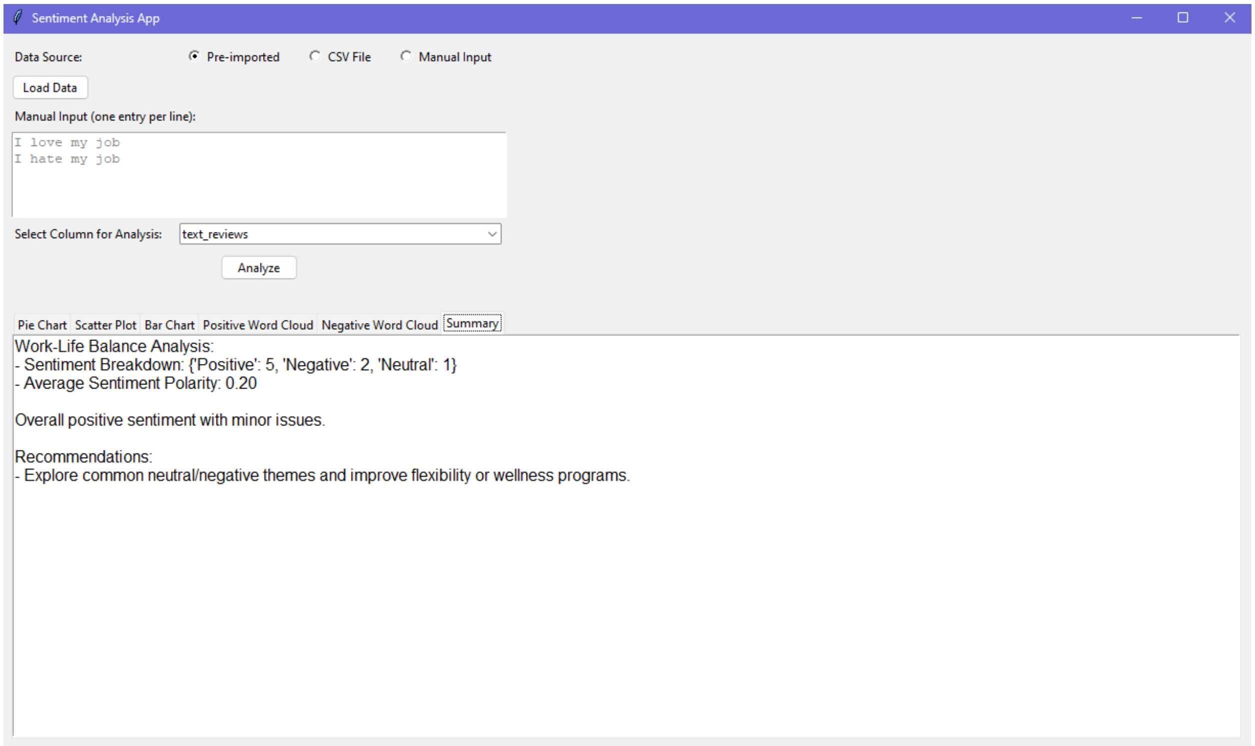
```
I love my job
I hate my job
```

Select Column for Analysis: work_life_balance

Data analyzed was numerical, so no word cloud was generated.

5.8 Screen 2.6

When pressing the summary tab, a summary of the results is generated within the white space. This summary includes numerical results as well as a general word-based sentiment ranging from "excellent overall sentiment regarding work-life-balance" to "poor sentiment demonstrates severe dissatisfaction". The summary also includes recommendations to further improve the work-life-balance of their employees based on the current sentiment analysis. This tab looks very similar regardless of whether the data set is text-based or numerical.



Data Source: ● Pre-imported ○ CSV File ○ Manual Input

Manual Input (one entry per line):

```
I love my job
I hate my job
```

Select Column for Analysis: text_reviews

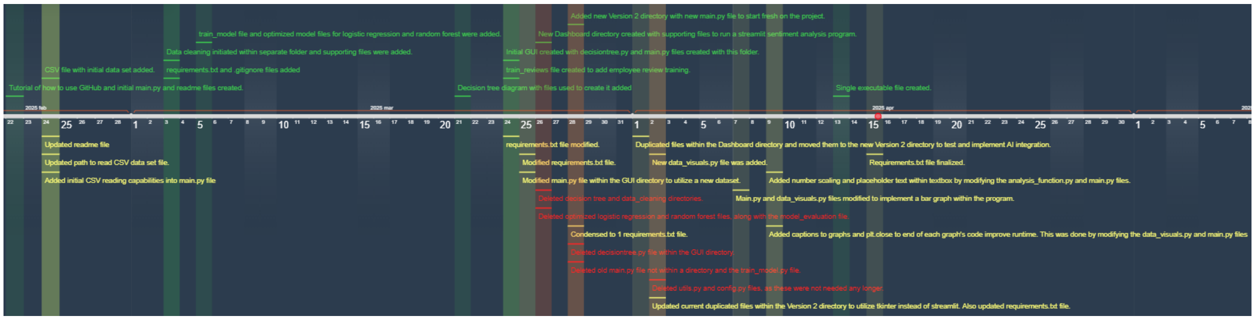
```
Work-Life Balance Analysis:
- Sentiment Breakdown: {'Positive': 5, 'Negative': 2, 'Neutral': 1}
- Average Sentiment Polarity: 0.20

Overall positive sentiment with minor issues.

Recommendations:
- Explore common neutral/negative themes and improve flexibility or wellness programs.
```

Appendix A: Project Timeline

Added: Green - *Modified: Yellow* - *Deleted: Red*



Added new Version 2 directory with new main.py file to start fresh on the project.

train_model file and optimized model files for logistic regression and random forest were added.

New Dashboard directory created with supporting files to run a streamlit sentiment analysis program.

Data cleaning initiated within separate folder and supporting files were added.

Initial GUI created with decisiontree.py and main.py files created with this folder.

CSV file with initial data set added.

requirements.txt and .gitignore files added

train_reviews file created to add employee review training.

Tutorial of how to use GitHub and initial main.py and readme files created.

Decision tree diagram with files used to create it added

Single executable file created.

Updated readme file

Updated path to read CSV data set file.

Added initial CSV reading capabilities into main.py file

requirements.txt file modified.

Modified requirements.txt file.

Modified main.py file within the GUI directory to utilize a new dataset.

Deleted decision tree and data_cleaning directories.

Deleted optimized logistic regression and random forest files, along with the model_evaluation file.

Condensed to 1 requirements.txt file.

Deleted decisiontree.py file within the GUI directory.

Deleted old main.py file not within a directory and the train_model.py file.

Duplicated files within the Dashboard directory and moved them to the new Version 2 directory to test and implement AI integration.

New data_visuals.py file was added.

Deleted utils.py and config.py files, as these were not needed any longer.

Updated current duplicated files within the Version 2 directory to utilize tkinter instead of streamlit. Also updated requirements.txt file.

Main.py and data_visuals.py files modified to implement a bar graph within the program.

Added number scaling and placeholder text within textbox by modifying the analysis_function.py and main.py files.

Added captions to graphs and plt.close to end of each graph's code improve runtime. This was done by modifying the data_visuals.py and main.py files

Requirements.txt file finalized.