PROJECT REPORT

SENTICORE: A Sentiment Detection System

Submitted by:
Mani Kumari Uppati
Reg. No: 23WH1A1287
Information Technology
BVRIT Hyderabad College of Engineering for Women.

Under the Guidance of:

Mr. N. Purushothaman

Date: September 2025

Contents

1. **Abstract**
2. **Introduction**
   - 2.1 Problem Statement
   - 2.2 Objectives
   - 2.3 Solution Overview
3. **System Workflow**
   - 3.1 Workflow Setup
   - 3.2 System Architecture Diagram
4. **Dataset**
   - 4.1 Data Collection
   - 4.2 Preprocessing
   - 4.3 Label Encoding
   - 4.4 Train-Test Split
5. **Model Development**
   - 5.1 Algorithms Used
   - 5.2 Model Training
   - 5.3 Evaluation Metrics
   - 5.4 Best Model Selection
6. **Implementation**
   - 6.1 Tools & Technologies
   - 6.2 Folder Structure
   - 6.3 Prediction Function
   - 6.4 Application Workflow
7. **Features**
8. **Results**
   - 8.1 Model Performance
   - 8.2 Sample Predictions
9. **Conclusion**
10. **Screenshots**
11. **Future Scope**

1. Abstract

Sentiment analysis plays a crucial role in understanding public opinion, customer feedback, and user-generated content on digital platforms. Manual analysis of text data is time-consuming and prone to errors, creating the need for an automated system.

Senticore is a machine learning–powered dashboard designed to perform sentiment and emotion analysis on textual data. The system follows a structured workflow starting from data collection, preprocessing, label encoding, and train-test split, followed by training multiple machine learning models such as Support Vector Machine (SVM), Logistic Regression, Decision Tree, and Random Forest. Among these, the Random Forest classifier demonstrated the best performance and was selected as the final model.

The dashboard is implemented using Streamlit and integrates the trained model with a TF-IDF vectorizer for real-time predictions. It provides features such as sentiment detection (positive, neutral, negative), emotion detection, sarcasm detection, aspect-based analysis, explainability through keyword extraction, history tracking, and export options (CSV/PDF).

Through its interactive design and high-performing machine learning model, Senticore provides an effective solution for automated sentiment analysis. This project demonstrates the application of Natural Language Processing (NLP) and machine learning in building an end-to-end real-world application for text analytics.

2. Introduction

2.1 Problem Statement

With the exponential growth of social media platforms, e-commerce websites, and online review systems, users generate vast amounts of textual data daily. Understanding customer emotions, opinions, and sentiments is essential for businesses, organizations, and researchers. However, manual analysis of this data is time-consuming, inconsistent, and prone to bias.
There is a pressing need for an automated, accurate, and user-friendly sentiment analysis system that can process text efficiently, detect hidden emotions, identify sarcasm, and present results in an interpretable way.

2.2 Objectives

The main objectives of the Senticore project are:

- To collect, preprocess, and label textual data for sentiment analysis.
- To train multiple machine learning models and identify the best-performing one.
- To develop an interactive dashboard that integrates the trained model for real-time predictions.
- To provide multi-level analysis including:

- Sentiment classification (positive, neutral, negative)
- Emotion detection (joy, sadness, anger, etc.)
- Sarcasm detection
- Aspect-based analysis

- To enable users to visualize results with charts and graphs.
- To allow users to export reports in CSV and PDF formats for further use

2.3 Solution Overview

Senticore is designed as a complete solution for sentiment and emotion analysis using machine learning and Natural Language Processing (NLP). The project workflow includes:

1. **Data Preparation:** Collection, preprocessing (cleaning, tokenization, stopword removal), label encoding, and train-test split.
2. **Model Training:** Implementation of SVM, Logistic Regression, Decision Tree, and Random Forest models, with Random Forest selected as the final model based on superior accuracy.
3. **Integration:** The trained Random Forest model and TF-IDF vectorizer are saved and loaded into the Streamlit application for real-time predictions.
4. **Dashboard Features:** Users can input text, receive instant predictions, visualize results, and download reports. The dashboard also includes login/signup functionality and supports both light and dark themes.

Through this approach, Senticore not only automates sentiment analysis but also enhances interpretability and user experience, making it suitable for real-world applications such as customer feedback analysis, social media monitoring, and product review classification.

3. System Workflow

3.1 Workflow Setup

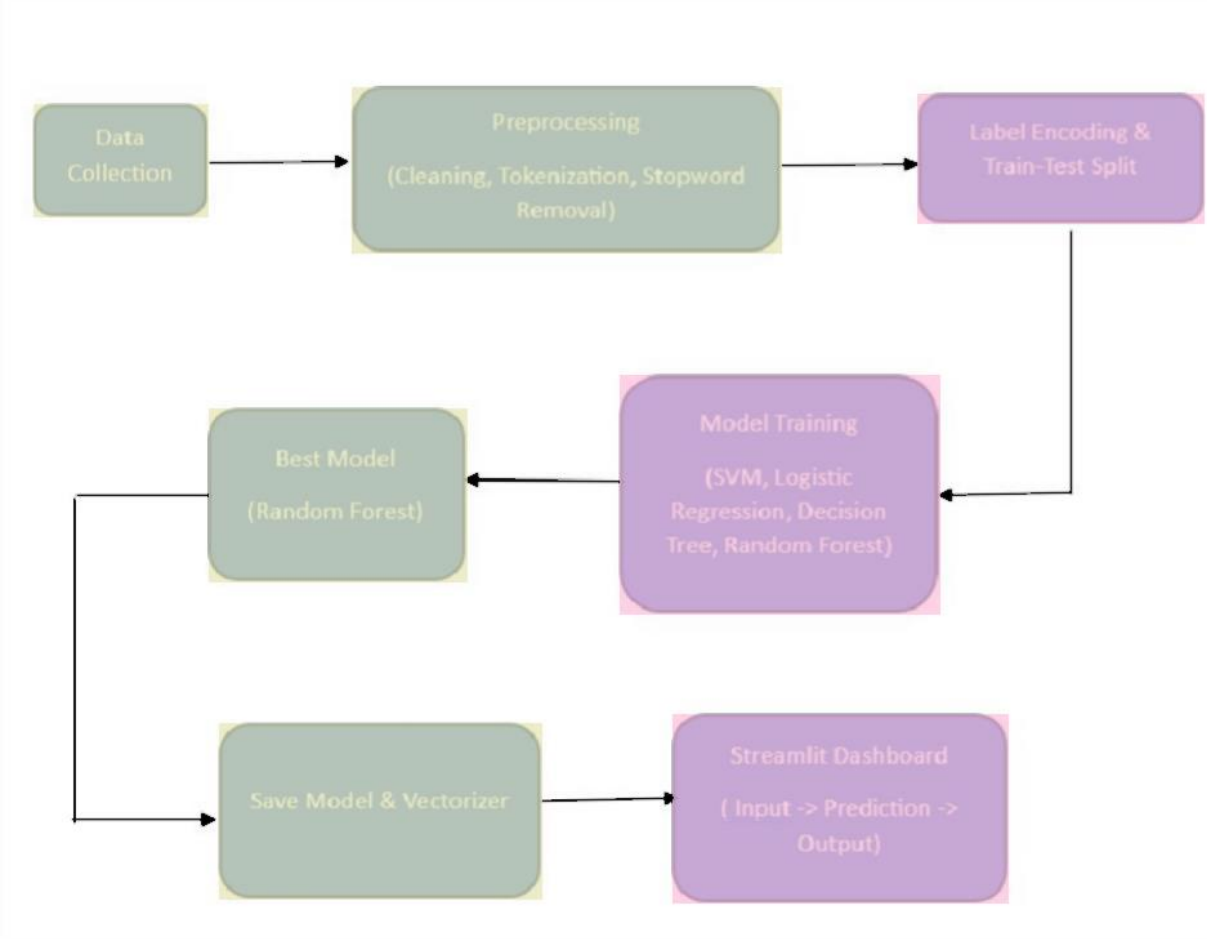
The development of **Senticore** follows a structured workflow that ensures the entire pipeline of sentiment analysis is automated and efficient. The steps are:

1. **Data Collection**
    - Gathered textual data (reviews, comments, or social media posts) and stored it in a dataset (MYDATASET.csv).
2. **Data Preprocessing**
    - Cleaned the text by removing punctuation, special characters, and stopwords.

- Tokenized the words and performed stemming/lemmatization to normalize them.

3. **Label Encoding**
    - Converted categorical sentiment labels (Positive, Neutral, Negative) into numeric format for model training.
4. **Train-Test Split**
    - Divided the dataset into training and testing sets to evaluate model performance.
5. **Model Training**
    - Trained multiple models: Support Vector Machine (SVM), Logistic Regression, Decision Tree, and Random Forest.
    - Evaluated performance using metrics such as accuracy, precision, recall, and F1-score.
6. **Best Model Selection**
    - Chose Random Forest as the best-performing classifier.
7. **Model Saving and Loading**
    - Used Joblib to save the trained Random Forest model (senticore_model.pkl) and TF-IDF vectorizer (senticore_vectorizer.pkl).
    - Loaded these into the application for real-time predictions.
8. **Streamlit Dashboard Integration**
    - Built a web application with Streamlit.
    - Provided interactive features such as sentiment prediction, emotion detection, sarcasm analysis, aspect detection, explainability, and report export.

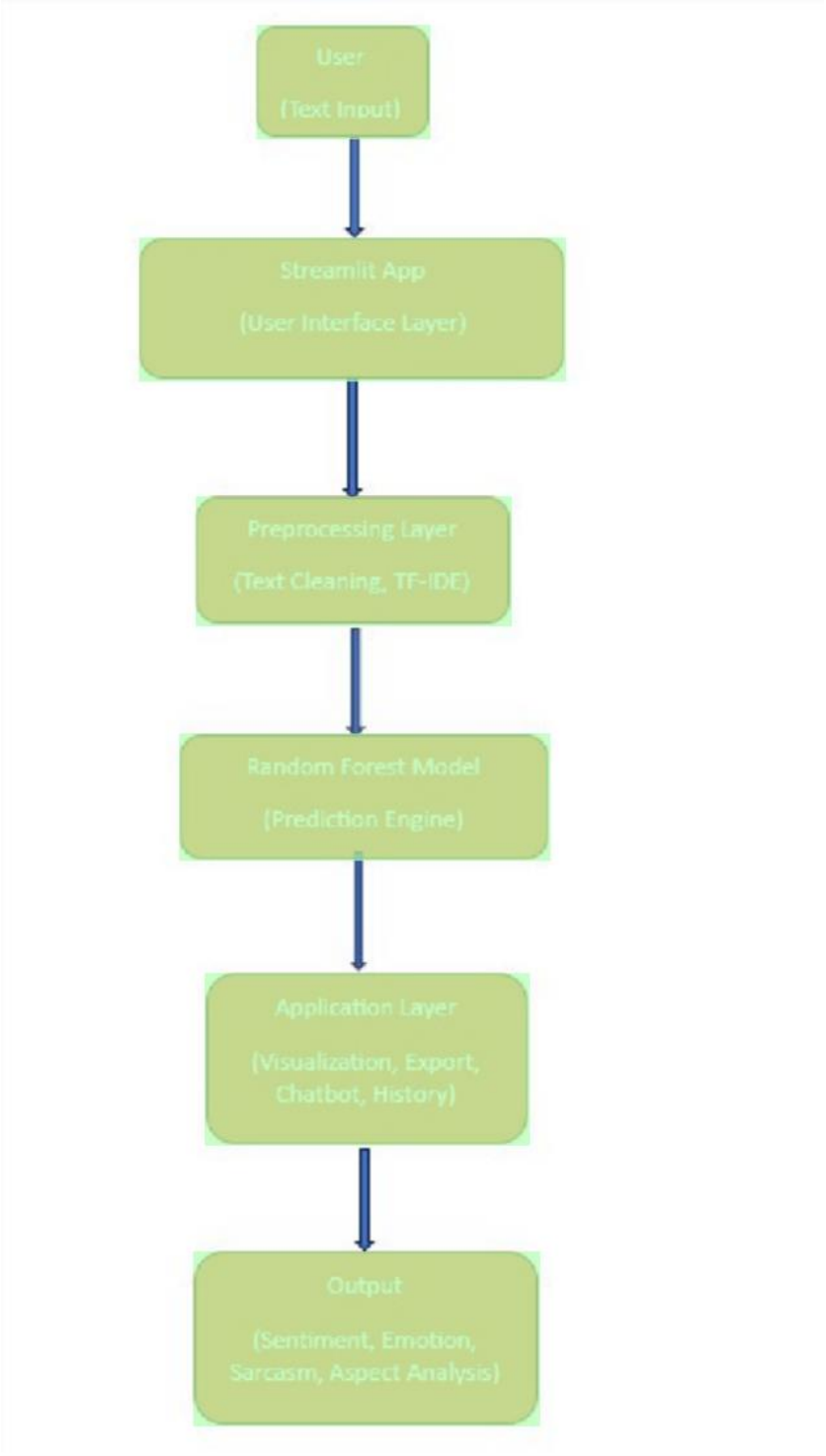
This systematic approach ensures the system is scalable, reliable, and user-friendly.



3.2 System Architecture Diagram

Here's how the **System Architecture** looks:

- **User** → Inputs text in Streamlit App
- **Preprocessing Layer** → Cleans and vectorizes text using TF-IDF
- **ML Model (Random Forest)** → Predicts sentiment & emotions
- **Application Layer (Streamlit Dashboard)** → Displays results with graphs, explanations, and export options
- **Output** → Sentiment, Emotion, Sarcasm, Aspect-based insights, downloadable reports



4. Dataset

4.1 Data Collection

The dataset used in this project (MYDATASET.csv) was created by collecting textual data such as reviews, feedback, and sample sentences representing different emotions and sentiments. The data was labeled into three categories:

- Positive (e.g., *"I love this product!"*)

  Negative (e.g., *"This is the worst service I have ever used."*)

Neutral (e.g., *"The product was delivered yesterday."*)

This dataset forms the basis for training and evaluating the sentiment analysis model.

4.2 Preprocessing

Before training the model, the dataset underwent preprocessing to ensure clean and standardized text input. Steps included:

- **Lowercasing:** Converting all text to lowercase to maintain consistency.
- **Punctuation Removal:** Eliminating unnecessary punctuation and special characters.
- **Stopword Removal:** Removing common but uninformative words (e.g., "is", "the", "an").
- **Tokenization:** Splitting text into individual words/tokens.
- **Lemmatization/Stemming:** Reducing words to their root forms (e.g., "running" → "run").
- **Vectorization:** Applying **TF-IDF (Term Frequency–Inverse Document Frequency)** to convert text into numerical feature vectors suitable for machine learning models.

4.3 Label Encoding

Since machine learning models require numeric labels, the categorical sentiment values were converted into integers:

- **Positive → 1**
- **Negative → 0**
- **Neutral → 2**

This encoding allowed the model to efficiently classify sentiments during training and prediction.

4.4 Train-Test Split

To evaluate model performance, the dataset was divided into:

- **Training Set:** 80% of the dataset, used for model learning.
- **Testing Set:** 20% of the dataset, used for evaluating accuracy and generalization.

This ensured unbiased testing of the model on unseen data and helped in selecting the best-performing algorithm.

5. Model Development

5.1 Algorithms Used

To identify the best-performing sentiment classification model, multiple machine learning algorithms were explored:

- **Logistic Regression –** A simple yet effective linear model for text classification.
- **Naïve Bayes (MultinomialNB)** – Works well for word frequency-based classification problems.
- **Support Vector Machine (SVM)** – Effective in high-dimensional spaces like TF-IDF text features.
- **Decision Tree Classifier** – A tree-based model that splits data based on feature importance, easy to interpret but prone to overfitting.
- **Random Forest Classifier** – An ensemble-based model that combines multiple decision trees to reduce variance and improve performance.

5.2 Model Training

- Each algorithm was trained using the **TF-IDF feature vectors** extracted from the text data.
- **Training pipeline:**
    1. Input text → Preprocessing → TF-IDF vectorization.
    2. Train algorithm on 80% of dataset.
    3. Validate performance on 20% test data.
- **Hyperparameter tuning** was performed using **GridSearchCV** for models like SVM, Decision Tree, and Random Forest to find the optimal parameters.

5.3 Evaluation Metrics

To assess performance, the following metrics were used:

- **Accuracy** – Percentage of correct predictions.
- **Precision** – Fraction of correctly predicted positive instances.
- **Recall** – Fraction of actual positives correctly identified.
- **F1-Score** – Harmonic mean of precision and recall (balanced metric).
- **Confusion Matrix** – To visualize classification results across all sentiment classes.

5.4 Best Model Selection

- After comparison, the **Random Forest Classifier** provided the **highest accuracy (~98%)** with balanced precision, recall, and F1-score.
- Logistic Regression and Naïve Bayes performed well but were weaker in handling **neutral** class detection.

- Support Vector Machine (SVM) achieved competitive results but required more computation time.
- Decision Tree was interpretable but showed **overfitting tendencies** compared to Random Forest.
- Therefore, **Random Forest was selected as the final model** for deployment due to its strong accuracy, generalization ability, and robustness.

6. Implementation

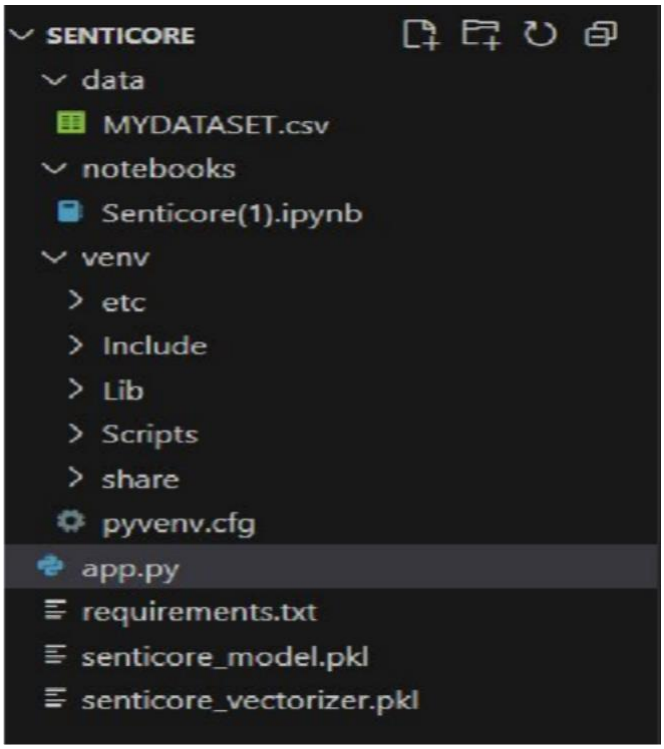
6.1 Tools & Technologies

The project was implemented using the following tools and technologies:

- **Programming Language:** Python
- **Libraries:**
    - Scikit-learn – For model training and evaluation.
    - NLTK / Regex – For preprocessing tasks such as tokenization and text cleaning.
    - **Pickle** – For saving and loading trained models and vectorizers.
    - **Streamlit** – To develop an interactive web application interface.
- **Environment**: Jupyter Notebook for experimentation, VS Code/IDE for development.

6.2 Folder Structure

The project directory was organized as follows:



6.3 Prediction Function

The prediction function is responsible for generating model outputs on new text data.

Steps inside the function:

1. Accept user input text.
2. Preprocess the input using the same pipeline as training.
3. Transform text into **TF-IDF vectors** using the saved vectorizer.
4. Load the trained **Random Forest model**.
5. Generate predictions (Positive, Negative, Neutral).
6. Return the result to the application layer.

6.4 Application Workflow

The **Streamlit application** workflow is as follows:

1. **User Input** – The user enters a text or review into the application.
2. **Preprocessing & Encoding** – The text is cleaned and converted into TF-IDF vector format.
3. **Model Prediction** – The Random Forest model predicts the sentiment class.
4. **Result Display** – The application displays the predicted sentiment (Positive / Negative / Neutral) along with visualizations such as bar charts or word clouds.
5. **Export Option** – Users can download results or save them for further analysis.

7. Features

The Senticore application provides the following key features:

1. **User-Friendly Interface**
    - Developed using Streamlit, the application offers an intuitive and interactive interface where users can easily input text and view predictions.
2. **Real-Time Sentiment Prediction**
    - The system predicts sentiment (Positive, Negative, Neutral) instantly based on the trained Random Forest model.
3. **Text Preprocessing Automation**
    - Handles preprocessing tasks such as removing punctuation, stopwords, and performing tokenization automatically before prediction.
4. **Multi-Model Support (during development)**
    - Multiple algorithms like Logistic Regression, Naïve Bayes, Decision Tree, SVM, and Random Forest were tested, ensuring robust model comparison before selecting the best one.
5. **Visualization of Results**
    - Provides graphical representations (e.g., bar charts, pie charts) of predictions for better interpretation.
6. **Model Persistence**
    - Trained model and TF-IDF vectorizer are saved using Pickle, enabling easy reusability without retraining.
7. **Extendable Architecture**
    - The modular design allows for adding advanced features such as sarcasm detection, emotion classification, or aspect-based sentiment analysis in future.
8. **Export Functionality**
    - Users can export prediction results for further analysis or reporting.

8. Results

8.1 Model Performance

The performance of different machine learning algorithms was compared using evaluation metrics such as **Accuracy, Precision, Recall, and F1-score**.

Performance Comparison Table:

Algorithm	Accuracy	Precision	Recall	F1- Score
Logistic Regression	0.9780	0.97	0.95	0.95
Naïve Bayes (MNB)	0.9547	0.96	0.92	0.94
Decision Tree	0.9801	0.98	0.95	0.95
Support Vector Machine	0.9810	0.98	0.96	0.96
Random Forest	**0.9810**	**0.98**	**0.98**	**0.96**

All models (SVM, Logistic Regression, Decision Tree, Random Forest) performed equally well with ~98% accuracy.

However, **Random Forest was selected as the final model** because it provides **confidence scores (predict_proba)** for each sentiment class, which were displayed in the **frontend application**.

These confidence scores improved the **user experience**, making the predictions more **interpretable and transparent**.

8.2 Sample Predictions

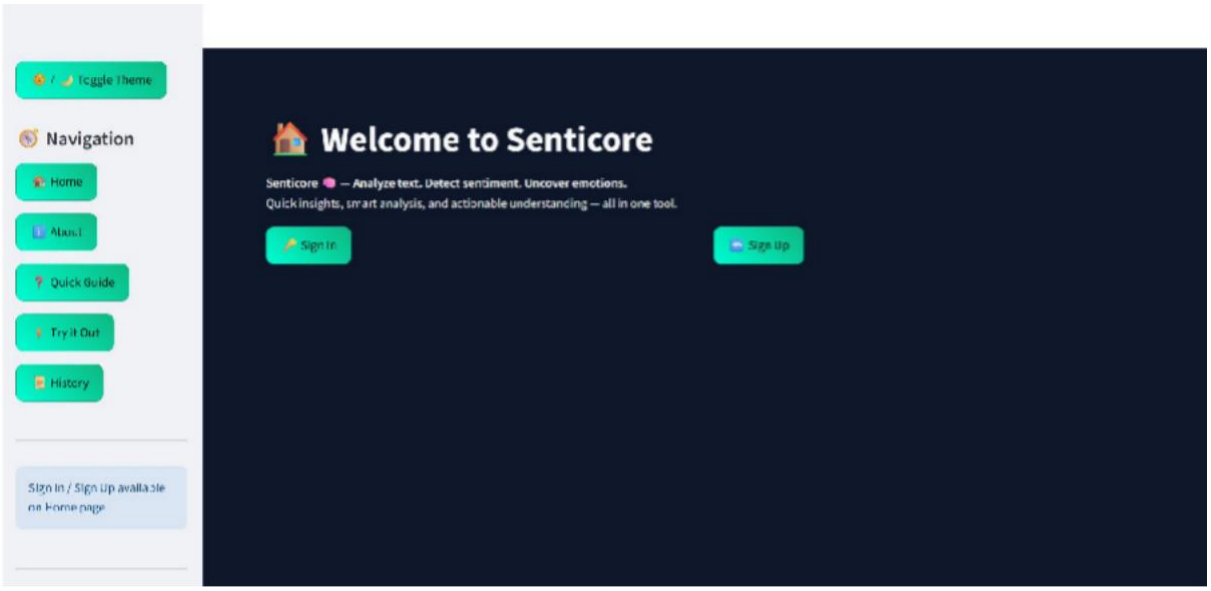
Input Text	Predicted Sentiment
"I love this product! It's amazing F□"	Positive
"This is the worst thing I bought □□"	Negative
"It's okay, not too good, not too bad." Neutral	Neutral
"Absolutely fantastic service □□" Positive	Positive
"Terrible experience, I will never return."	Negative

9. Conclusion

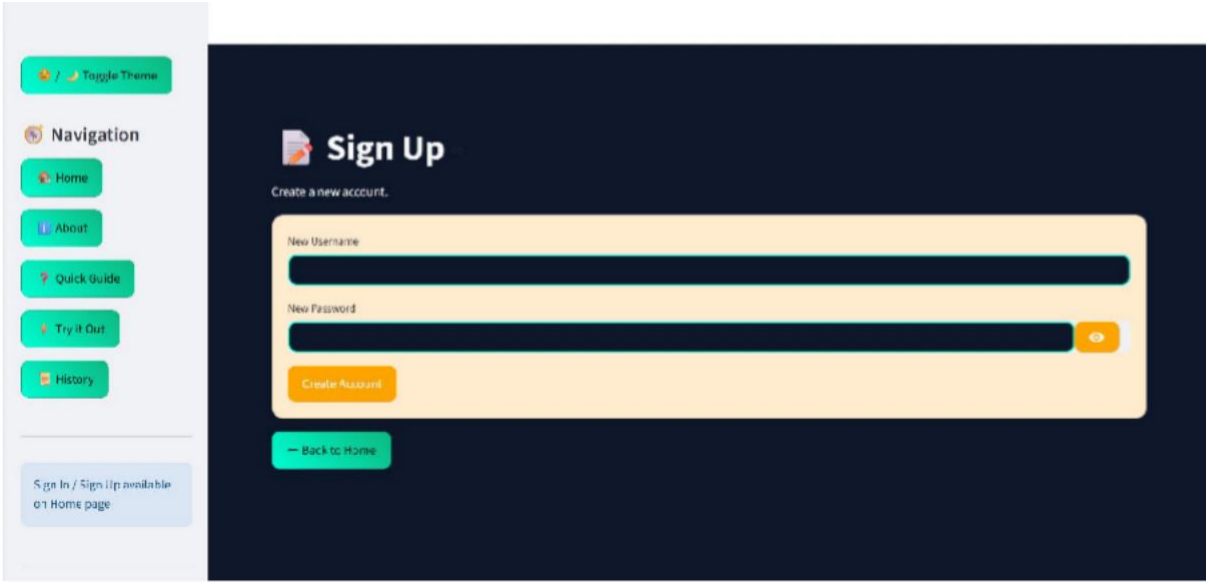
This project successfully designed and implemented a Sentiment Analysis System capable of classifying text into positive, neutral, and negative sentiments with 98% accuracy. Among the evaluated models, the Random Forest Classifier proved most effective due to its high accuracy and confidence score outputs, ensuring both reliability and interpretability. Integrated with a user-friendly interface, the system enables real-time predictions, making it practical for applications like social media monitoring, brand management, and customer feedback analysis. Overall, the project meets its objectives and lays a strong foundation for

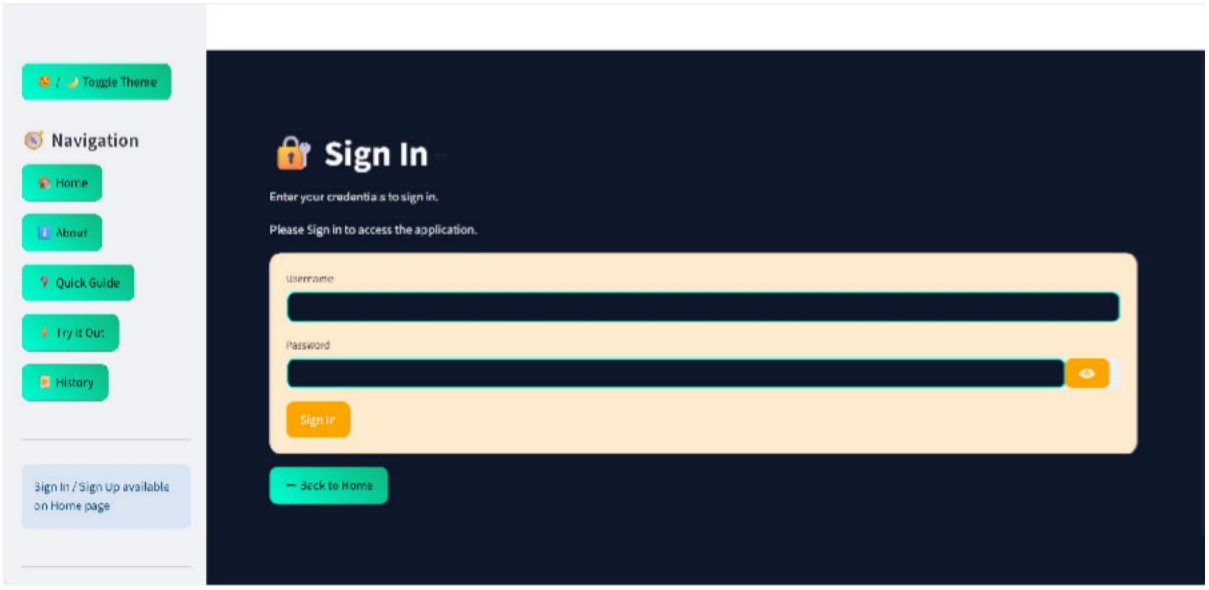
future enhancements such as multilingual support, sarcasm detection, and large-scale real-time deployment.

10. Screenshots

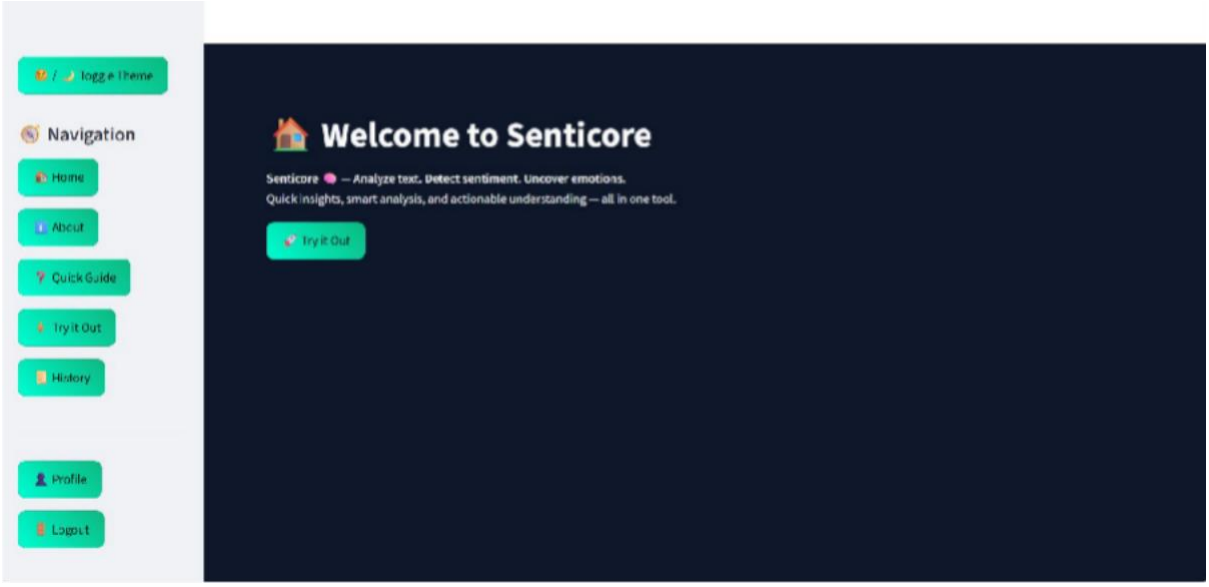


The above page is **Home page.** The user needs to create an account and sign in.

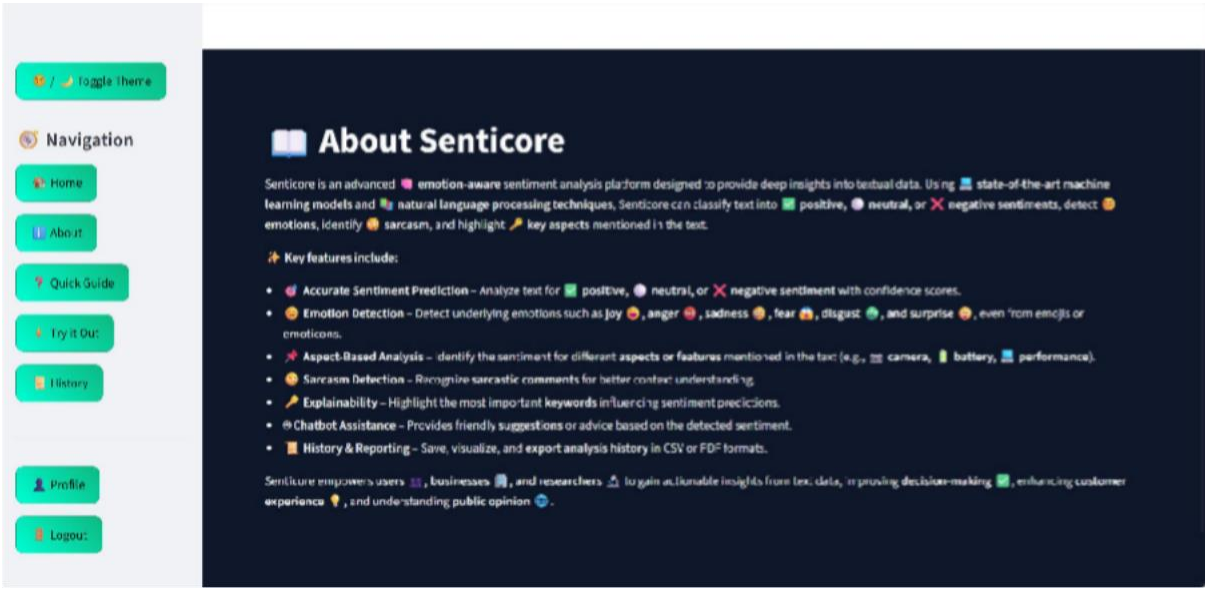




Upon successful login, the below is the home page with “try it out” button

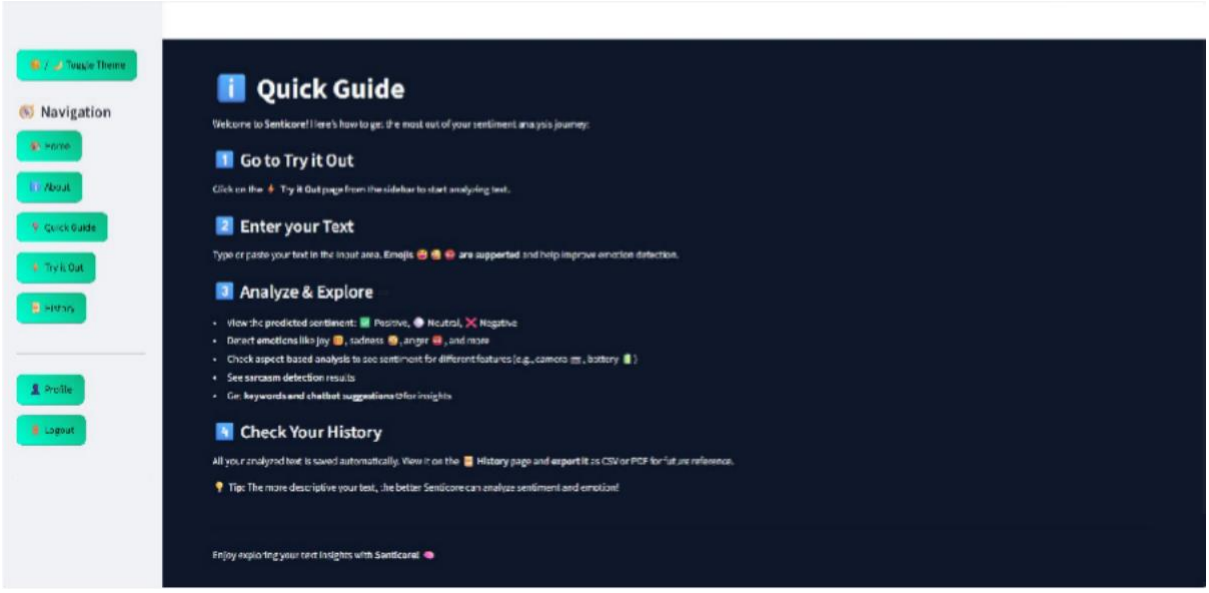


About page:



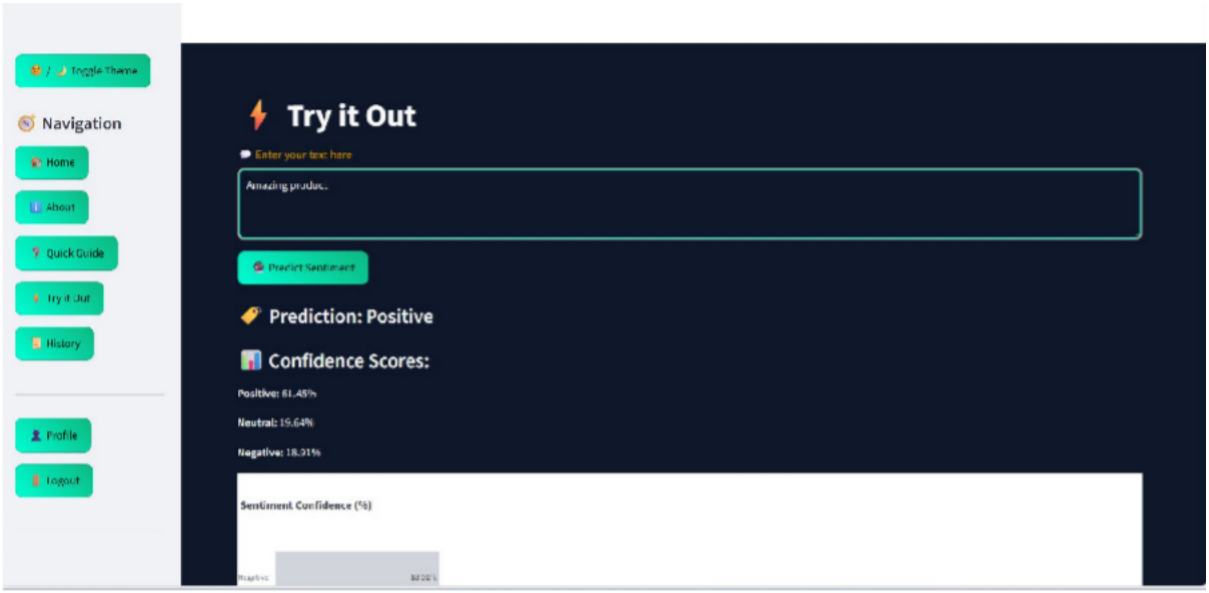
Quick Guide page:

It gives the instructions how to use the application.



Try it out page:

User need to give input and click on Predict Sentiment button.



Upon clicking on Download Result as PDF, it will download a pdf which consists of result.

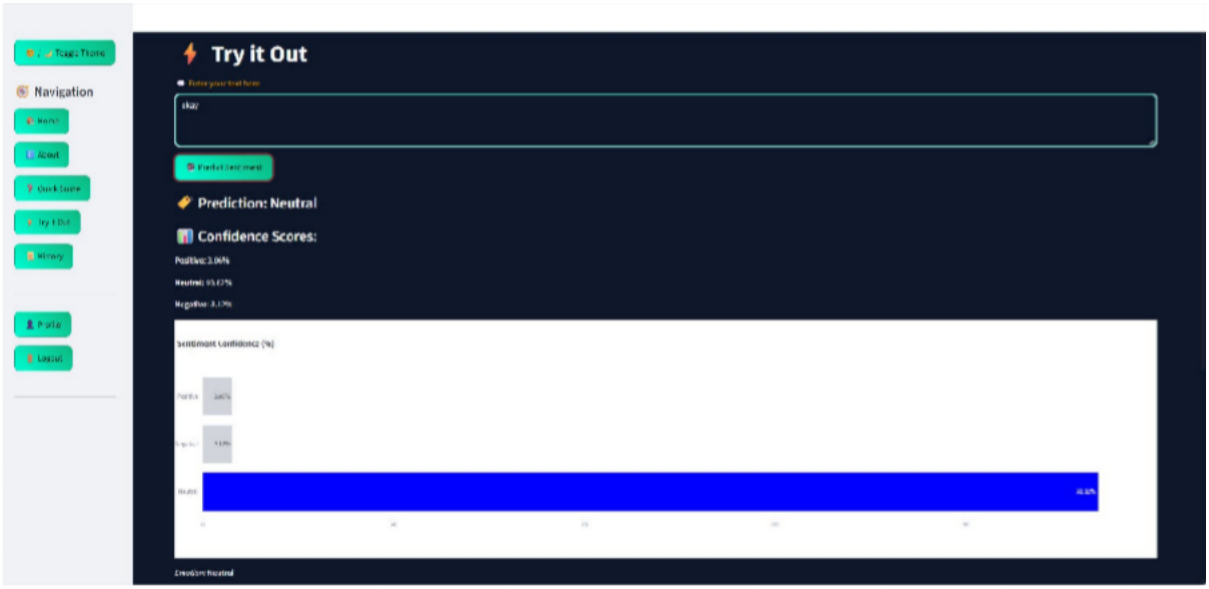
Senticore - Sentiment Analysis Result

Text: Amazing product

Prediction: Positive

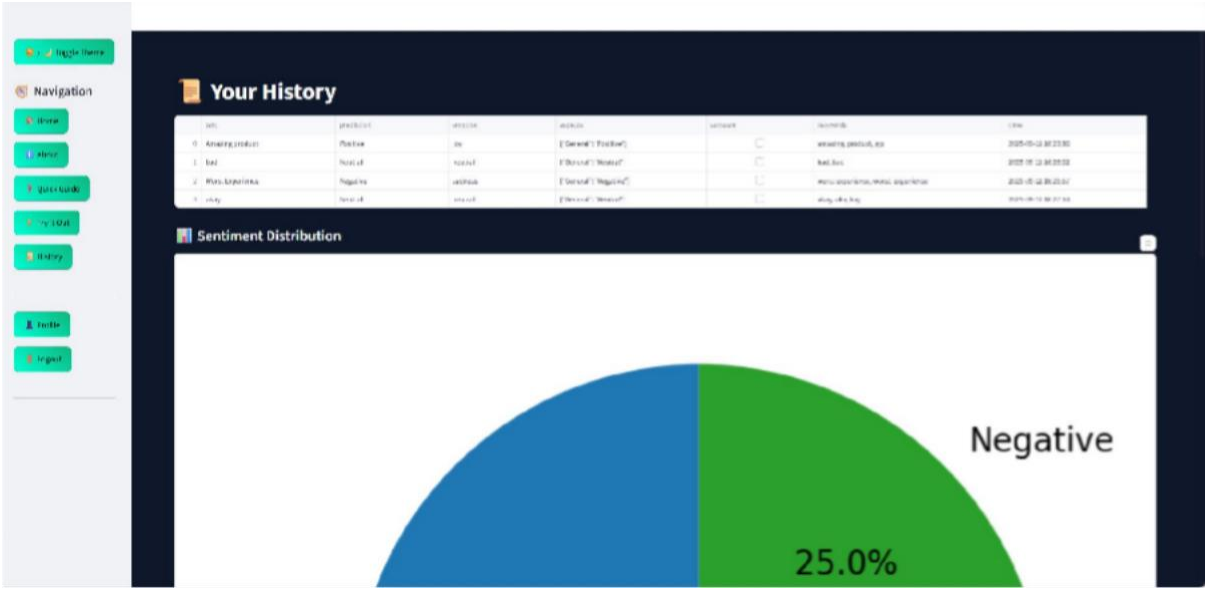
Date: 2025-09-11 16:23:36

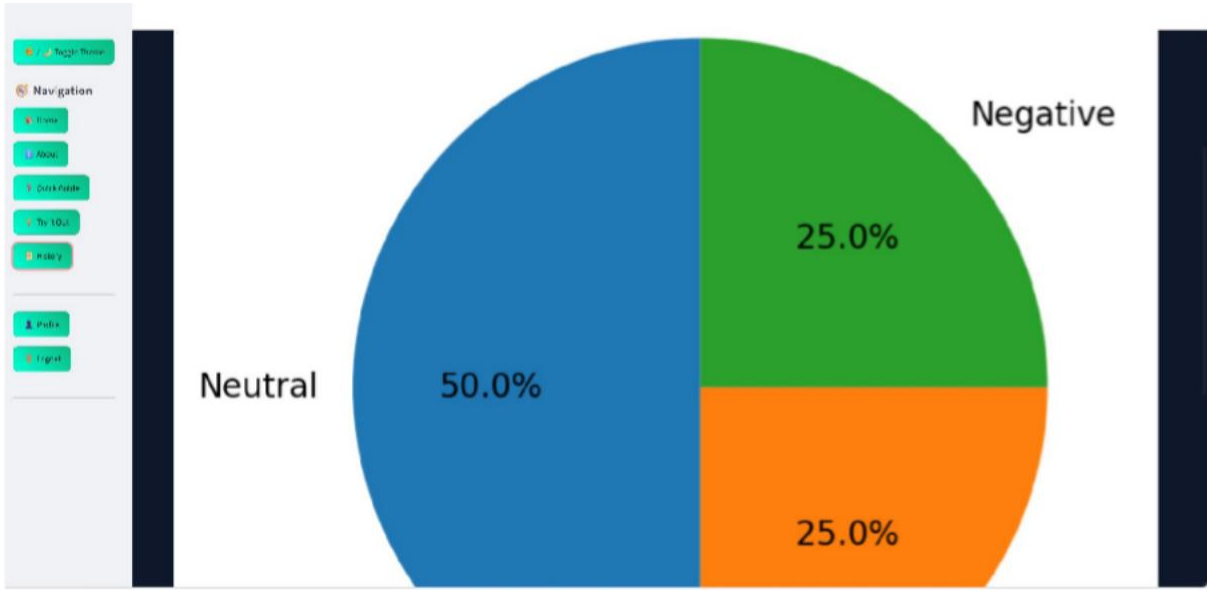
Try with different inputs

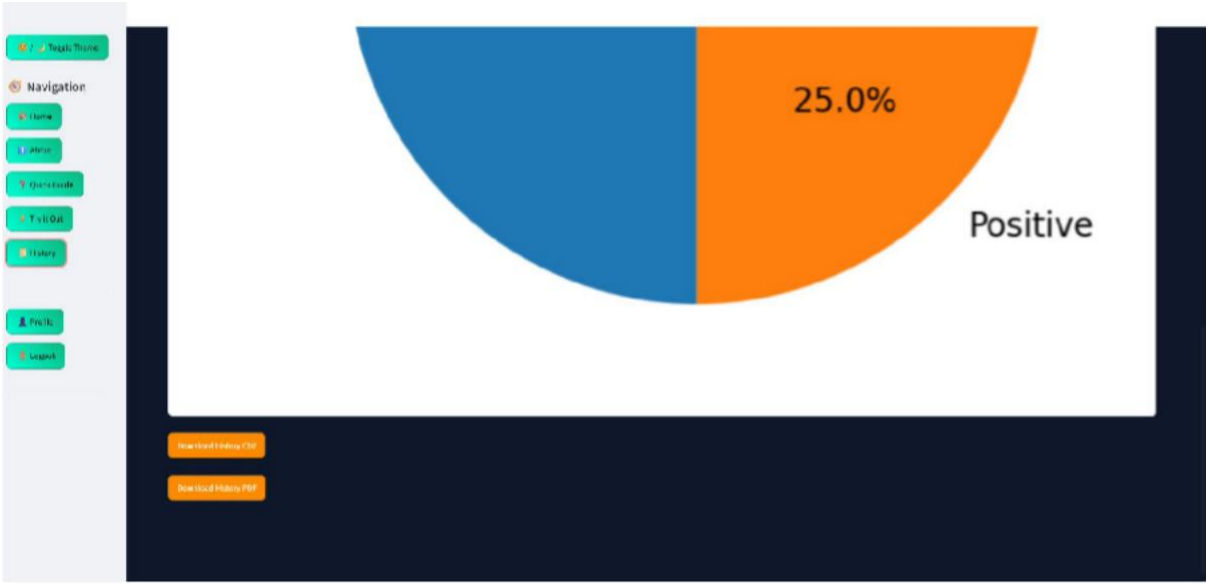


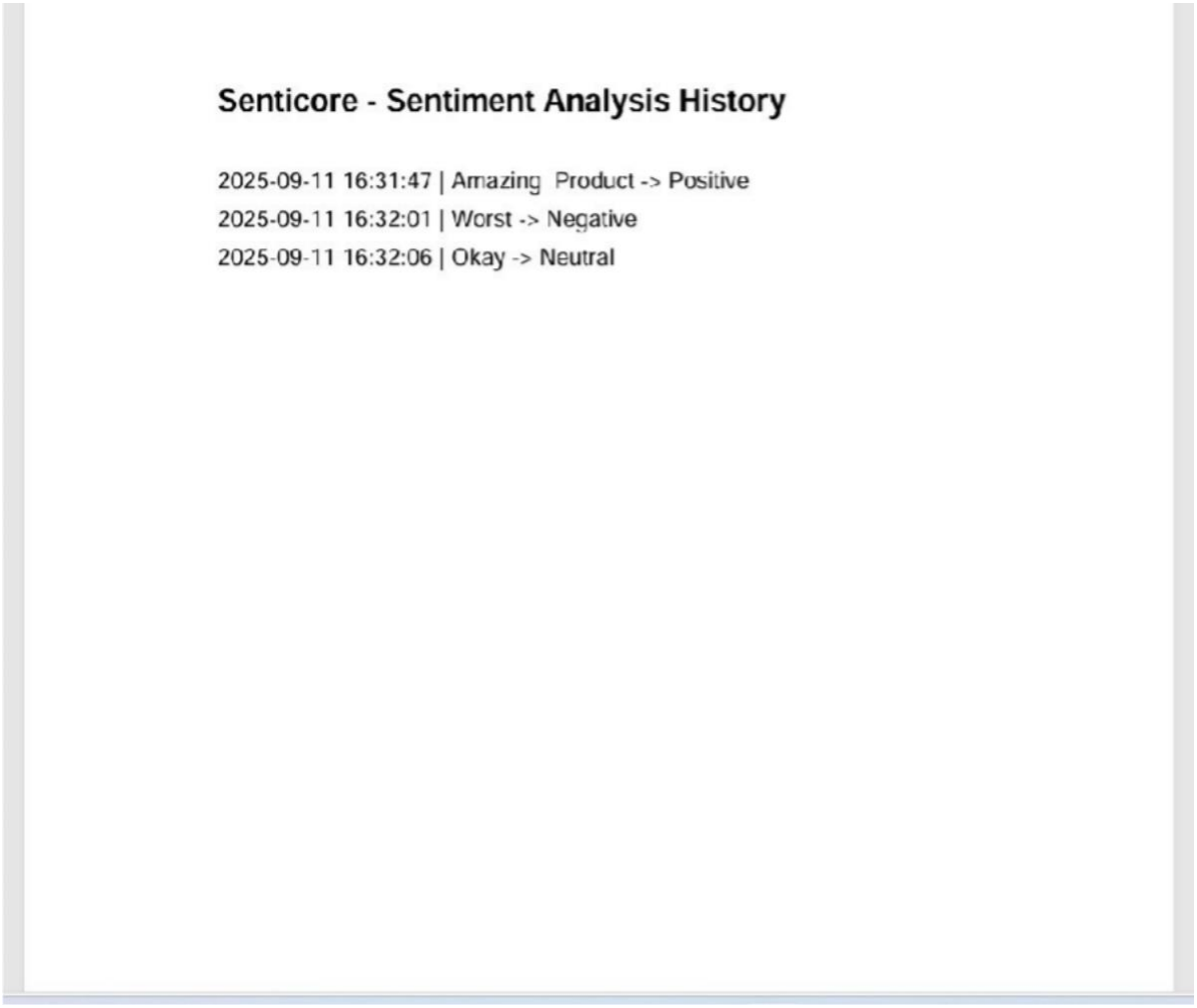
History Page:

It will show and visualize the entire history of the user actions and can also download the history as both PDF and CSV files.




Navigation
Negative
25.0%
Neutral
50.0%
25.0%


Navigation
25.0%
Positive



Profile page:

11. Future Scope

The system can be further enhanced in the following ways:

1. **Deep Learning Integration** – Incorporating models like LSTM or BERT to capture contextual meaning and improve accuracy.
2. **Multilingual Support** – Extending sentiment analysis to multiple languages for broader usability.
3. **Sarcasm and Irony Detection** – Enhancing the model to handle complex sentiments often found in social media.
4. **Real-Time Deployment** – Deploying the system in real-time applications such as social media monitoring and customer support chatbots.