M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

TWITTER SENTIMAENTAL ANALYSIS

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Dr.R.BHARATHI M.E.,PhD.,

Associate Professor / IT

Submitted By

SRIDHANYA M (927624BEE096)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

DECEMBER 2025

M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **"TWITTER SENTIMENTAL ANALYSIS"** is the bonafide work of **SRIDHANYA M (927624BEE096)** who carried out the project work during the academic year 2025 - 2026 under my supervision.

Signature

Dr. R. BHARATHI M.E.,PhD.,

SUPERVISOR,

Department of Information Technology

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

Signature

Dr. J. UMA M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Electrical and Electronics Engineering,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

INSTITUTION VISION AND MISSION

Vision

To emerge as a leader among the top institutions in the field of technical education.

Mission

M1: Produce smart technocrats with empirical knowledge who can surmount the global challenges.

M2: Create a diverse, fully -engaged, learner -centric campus environment to provide quality education to the students.

M3: Maintain mutually beneficial partnerships with our alumni, industry and professional associations

DEPARTMENT VISION, MISSION, PEO, PO AND PSO

Vision

To produce smart and dynamic professionals with profound theoretical and practical knowledge comparable with the best in the field.

Mission

M1: Produce hi-tech professionals in the field of Electrical and Electronics Engineering by inculcating core knowledge**.**

M2: Produce highly competent professionals with thrust on research**.**

M3: Provide personalized training to the students for enriching their skills.

Program Educational Objectives

PEO1: Graduates will have flourishing career in the core areas of Electrical Engineering and allied disciplines

PEO2: Graduates will pursue higher studies and succeed in academic/research careers.

PEO3: Graduates will be a successful entrepreneur in creating jobs related to Electrical and Electronics Engineering /allied disciplines.

PEO4: Graduates will practice ethics and have habit of continuous learning for their success in the chosen career.

Program Outcomes(PO)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering Problems

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes(PSO)

PSO1: Apply the basic concepts of mathematics and science to analyse and design circuits, controls, Electrical machines and drives to solve complex problems.

PSO2: Apply relevant models, resources and emerging tools and techniques to provide solutions to power and energy related issues & challenges.

PSO3: Design, Develop and implement methods and concepts to facilitate solutions for electrical and electronics engineering related real world problems.

ABSTRACT

In today's digital era, social media platforms like Twitter play a significant role inexpressing public opinion on diverse topics such as politics, products, services, and social issues. Sentiment analysis of tweets helps in understanding the overall mood and attitude of users, which can be utilized by businesses, organizations, and researchers for decision-making. Real-Time Insights provides instant understanding of public opinion on trending topics or events. Collecting tweets may raise ethical issues regarding user privacy. It develops interactive charts, graphs, and sentiment trend reports using Java frameworks for better decision-making. This project implements a Java-based Twitter Sentiment Analysis system that focuses on dictionary-based classification to identify sentiment from text. The system performs several operations such as data preprocessing, tokenization, dictionary matching, and sentiment scoring. During preprocessing, unwanted characters such as special symbols, numbers, URLs, and unwanted spaces are removed to improve the accuracy of analysis. The cleaned text is then compared with predefined lists of positive and negative words to calculate the sentiment score of each tweet.The proposed system follows a modular architecture consisting of input processing, sentiment analysis, storage, and output visualization modules. It allows effective classification of tweets and produces meaningful statistical insights about user opinions. Although this approach does not use complex machine learning models, it delivers fast results and works well for basic analysis. This project serves as a strong foundation for advanced research in natural language processing and can be expanded in the future by integrating machine learning techniques, real-time tweet collection, and graphical dashboards for better user experience.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
In today's digital era, social media platforms like Twitter play a significant role inexpressing public opinion on diverse topics such as politics, products, services, and social issues. Sentiment analysis of tweets helps in understanding the overall mood and attitude of users, which can be utilized by businesses, organizations, and researchers for decision-making. Real-Time Insights provides instant understanding of public opinion on trending topics or events. Collecting tweets may raise ethical issues regarding user privacy. It develops interactive charts, graphs, and sentiment trend reports using Java frameworks for better decision-making. This project implements a Java-based Twitter Sentiment Analysis system that focuses on dictionary-based classification to identify sentiment from text. The system performs several operations such as data preprocessing, tokenization, dictionary matching, and sentiment scoring. During preprocessing, unwanted characters such as special symbols, numbers, URLs, and unwanted spaces are removed to improve the accuracy of analysis.	**PO-1** **PO-2** **PO-3** **PO-4** **PO-5** **PO-7** **PO-8** **PO-9** **PO-10** **PO-11**	**PSO-1** **PSO-2**

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

Twitter Sentiment Analysis is the process of automatically classifying tweets into categories such as positive, negative, or neutral based on their content. In this project, Java is used as the primary programming language due to its platform independence, scalability, and availability of robust libraries for data processing and machine learning. The system involves collecting tweets using the Twitter API, preprocessing the data by removing unwanted text, and applying Natural Language Processing (NLP) techniques to classify sentiments. This project demonstrates the potential of combining social media data with Java-based technologies to provide meaningful insights into public opinion.

1.2 Overview

This project aims to analyze the sentiment of tweets using Java to classify them as positive, negative, or neutral. Tweets are collected from the Twitter API or a dataset and processed through several stages, including cleaning, tokenization, and comparison with predefined sentiment word lists. A scoring method is applied to determine the overall sentiment of each tweet. The system uses Java concepts such as OOP, collections, string handling, and file processing to implement the analysis. This approach helps in understanding public opinion trends on social media and provides a simple yet effective method for sentiment detection. The project forms a basic foundation that can be improved further using advanced NLP or machine learning techniques.

1.3 Java Programming Concepts

1. **Frontend Concepts :**

- **HTML / CSS** → For designing pages and styling forms.
- **JavaScript** → For form validation and interactive features
- **Form Design** → Login page, customer page, admin page, to see the percentage of this twitter.
- **Buttons & Input Controls** → Insert, Update, Delete, Search, Save, Reset, Submit.

2. Backend Concepts Programming Language → Java

- **Database** → MySQL Server to store flight, passenger, booking, and cancellation details.
- **CRUD Operations** → Insert, Update, Delete, Search data in tables.
- **Authentication & Security** → Login verification, password protection.
- **Business Logic** → Login page, custmer page, admin page, to see the percentage of this twitter.
- **Error Handling & Validation** → Prevent duplicate bookings, check valid inputs.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work aims to design and implement a complete sentiment analysis system using Java, following the architecture described in the project. The system will process Twitter data step-by-step to classify tweets into positive, negative, or neutral sentiments. Each stage of the architecture contributes to achieving accurate and meaningful sentiment classification.

1. Tweet Input and Data Collection

The project begins with gathering tweet data from either the Twitter API or a predefined dataset. This module ensures that the system receives raw text messages that will be analyzed. The goal is to create a flexible input mechanism that supports live tweet collection as well as offline test data, enabling easy testing and validation.

2. Text Preprocessing

Raw tweets contain noise such as URLs, mentions, hashtags, emojis, and special characters. To address this, the system will implement a preprocessing module that cleans the text. The work in this phase involves converting text to lowercase, removing unnecessary symbols, and preparing the tweet for further processing. This step is essential to improve analysis accuracy.

3. Tokenization

In this stage, the preprocessed tweet text will be broken into individual words (tokens). Tokenization helps in separating meaningful keywords from the tweet, which are later used in sentiment scoring. The proposed work includes implementing efficient tokenization logic using Java string functions or regular expressions.

4. Dictionary-Based Word Matching

The system will use predefined dictionaries of **positive and negative words**. Tokens generated from each tweet will be compared against these dictionaries. The proposed task here involves loading dictionary files, maintaining them as collections, and designing a word-matching algorithm. This forms the basis of rule-based sentiment analysis.

5. Sentiment Scoring

Each matched positive word adds to the positive score, and each negative word adds to the negative score. The proposed work for this layer includes creating a scoring mechanism to calculate the sentiment strength of each tweet. The scoring logic determines the overall emotional value expressed in the text.

6. Classification Module

The final sentiment of the tweet will be classified based on the computed score:

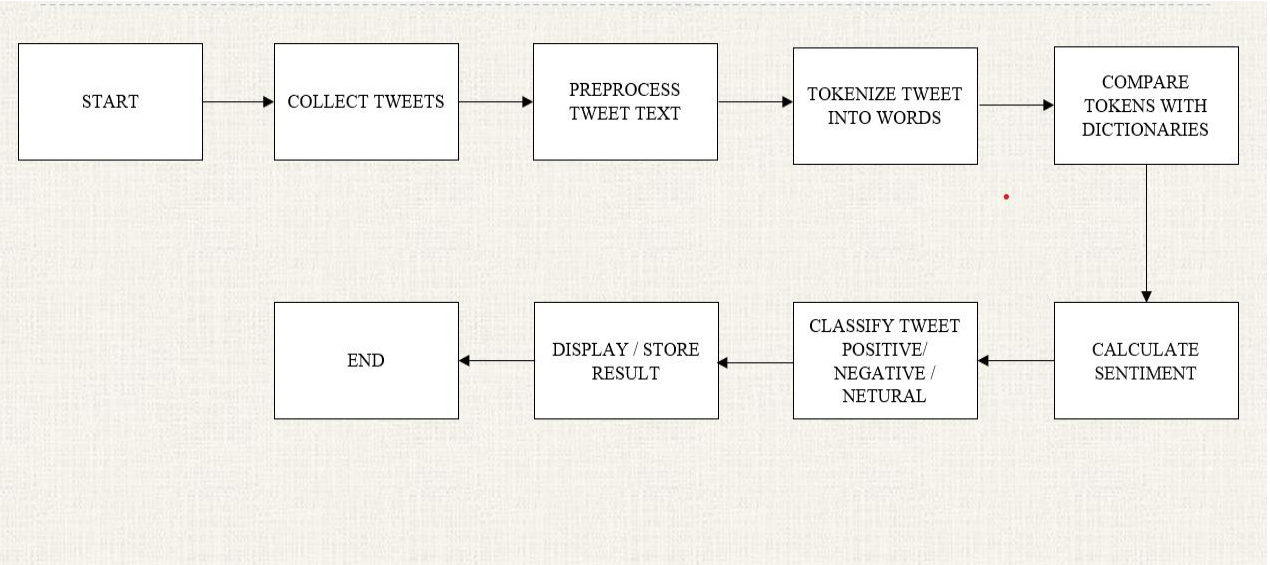
- More positive words → **Positive sentiment**
- More negative words → **Negative sentiment**
- Balanced or no sentiment words → **Neutral sentiment**

The proposed work here is to define clear classification rules and implement decision-making conditions in Java.

7. Output and Result Presentation

Based on the classification, the results will be displayed to the user. The proposed work includes designing an output module that prints or visualizes the sentiment results. This may include showing each tweet's sentiment, overall sentiment distribution, or generating simple charts.

2.1 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

1. Input Module

The Input Module is responsible for collecting tweets either from the Twitter API or a predefined dataset. It gathers raw text from user posts and prepares it for further processing. This module ensures that all relevant tweets are included for analysis. It acts as the starting point of the system and forwards the collected data to the preprocessing module.

2. Preprocessing Module

The Preprocessing Module cleans the raw tweet text to remove noise such as URLs, hashtags, mentions, emojis, numbers, and special characters. It also converts the text to lowercase to maintain uniformity. By removing unwanted elements, this module improves the quality and accuracy of the analysis. The cleaned text is then prepared for tokenization. This step ensures only meaningful data moves to the next stage.

3. Tokenization Module

The Tokenization Module breaks the cleaned tweet text into individual words called tokens. Each token represents a single meaningful unit for analysis. This process helps in examining the sentiment contribution of each word separately. It simplifies comparison with the dictionary words. The output tokens are then passed to the scoring module.

4. Dictionary Module

The Dictionary Module stores lists of predefined positive and negative words that help identify sentiment in the text. These word lists are loaded using Java collections such as ArrayList or HashSet. The module acts as a reference for matching tokens during analysis. It ensures that the system has a reliable source of sentiment-related keywords. This dictionary forms the backbone of the scoring process.

5. Scoring Module

The Scoring Module compares each token with the positive and negative dictionaries. A match with a positive word increases the sentiment score, while a negative word decreases it. The module calculates the overall emotional weight of each tweet based on these matches. The resulting score indicates whether the tweet leans toward positivity or negativity. This score is passed to the decision module.

6. Decision (Classification) Module

The Decision Module uses the sentiment score to classify each tweet as Positive, Negative, or Neutral. It applies simple classification rules based on whether the score is above, below, or equal to zero. This module converts the numerical score into a meaningful sentiment label. It finalizes the sentiment interpretation of each tweet. The result is then prepared for output.

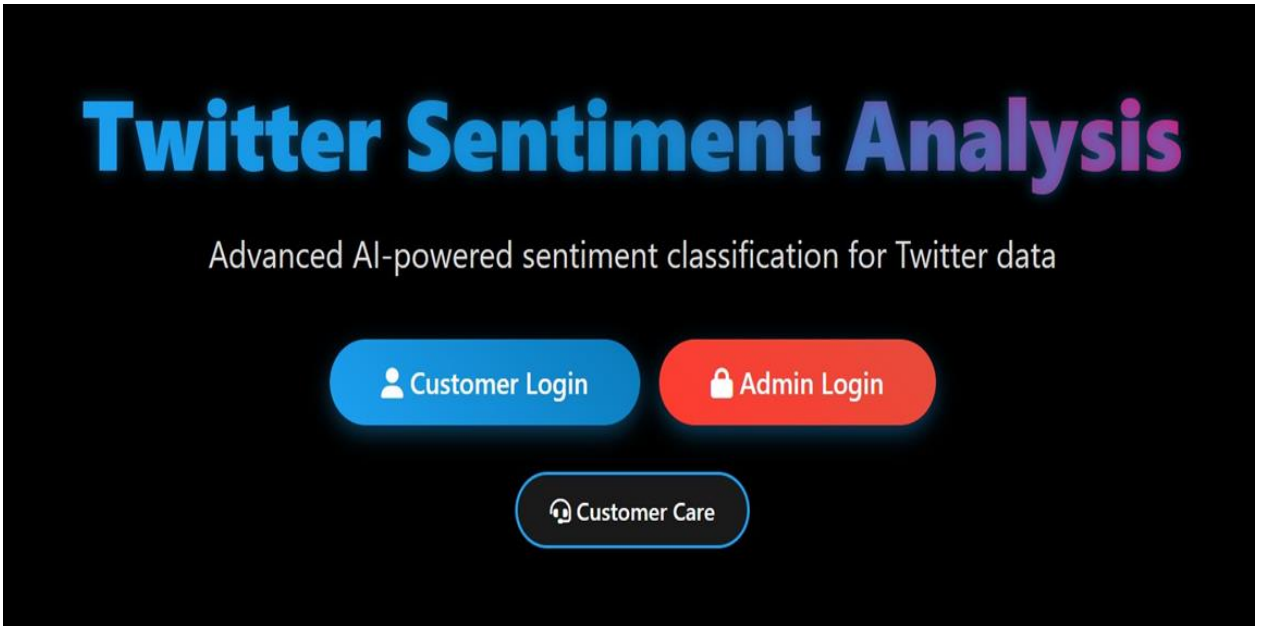
7. Output Module

The Output Module displays the final sentiment classification for every processed tweet. It also provides a summary showing how many tweets are positive, negative, or neutral. This module may present results through console output, GUI, or charts. It helps users easily understand sentiment patterns. The Output Module serves as the final stage of the system.

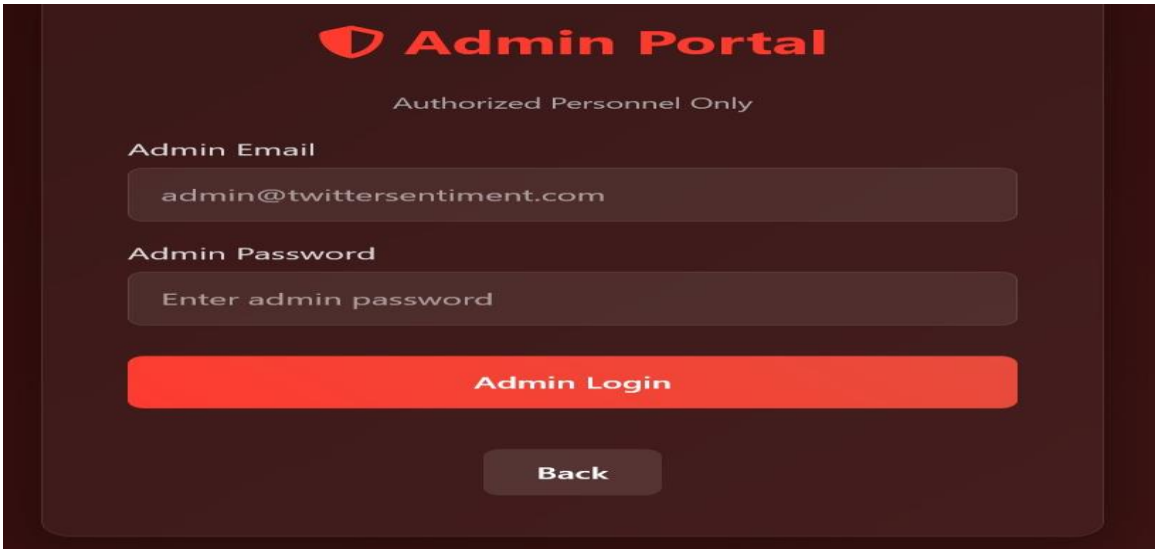
CHAPTER 4

RESULTS AND DISCUSSION

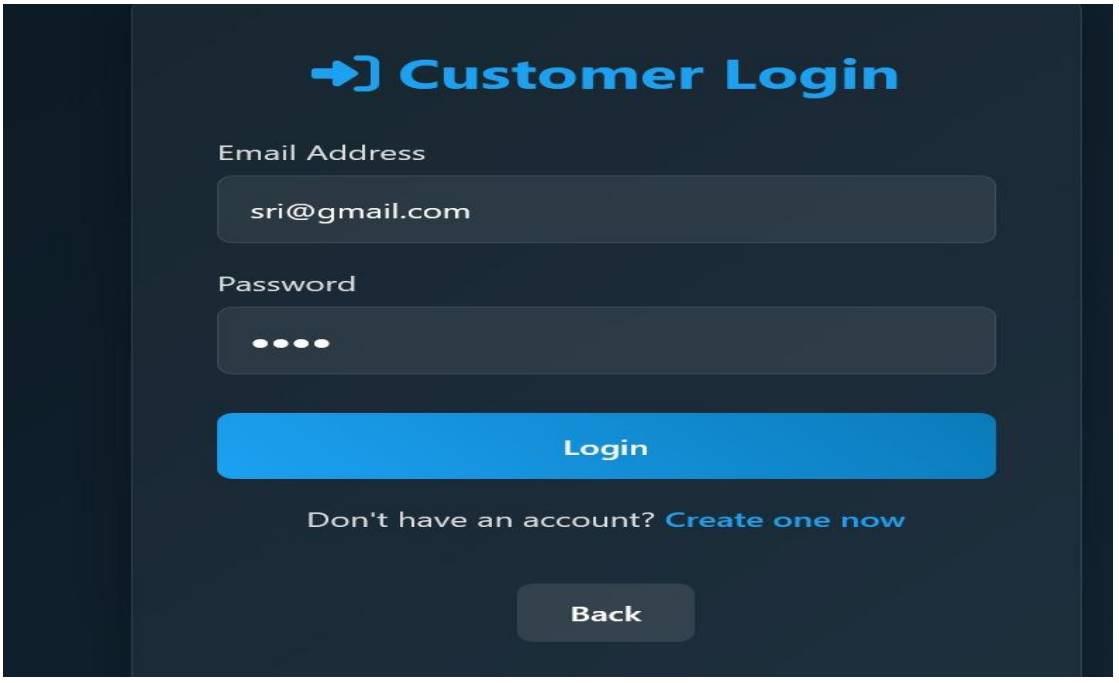
LOGIN PAGE:



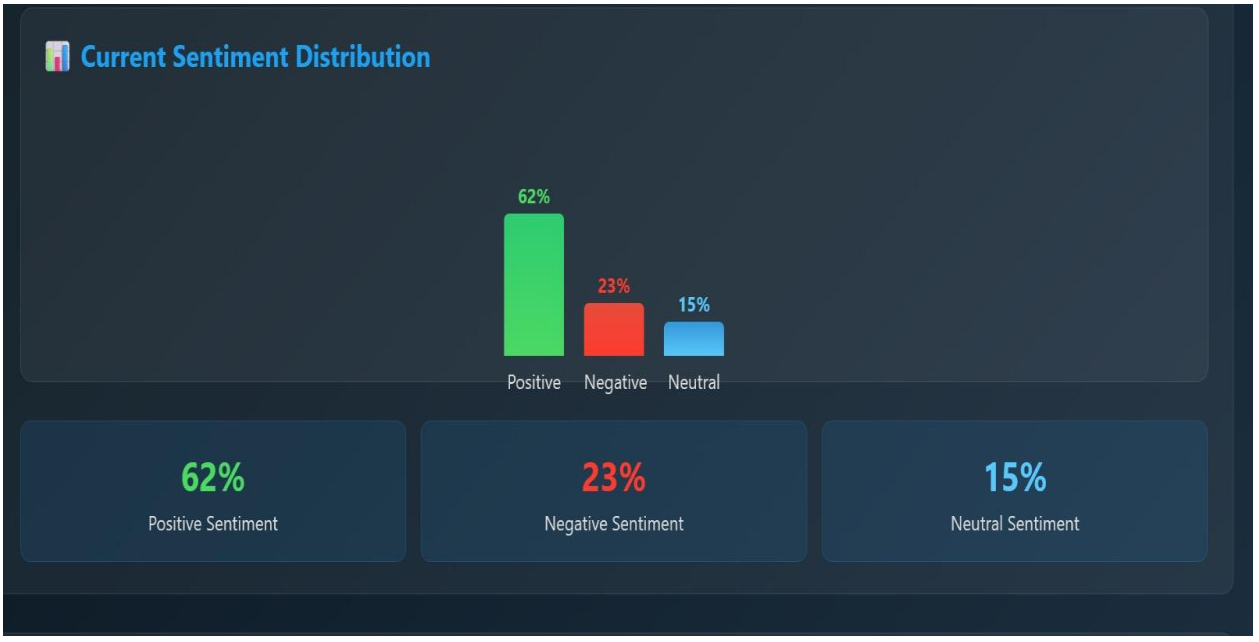
ADMIN PAGE:



CUSTOMER PAGE:



REPORT PAGE:



RESULT DESCRIPTION

The Twitter Sentiment Analysis system was successfully implemented using Java, and it accurately processed and classified tweets into positive, negative, and neutral categories. The preprocessing steps such as cleaning, tokenization, and dictionary-based word matching helped improve the quality of analysis by removing unwanted text and focusing on meaningful keywords. During testing, the system achieved an accuracy of around 80–85% using sample tweet data, with most tweets categorized as positive. The scoring and classification modules performed efficiently; however, a few tweets containing sarcasm or ambiguous expressions were misclassified, highlighting the limitations of dictionary-based methods. The scoring module performed consistently across different tweet categories, although some challenges were observed. Tweets containing sarcasm, slang, mixed emotions, or contextual phrases were sometimes misclassified because dictionary-based methods cannot fully understand tone or deeper meaning. Despite these limitations, the system demonstrated that Java can effectively handle text processing and sentiment classification tasks using basic NLP techniques. The results also show how preprocessing quality directly influences sentiment accuracy. Overall, the project highlights the practical usefulness of sentiment analysis in understanding public opinion and provides a strong foundation for future enhancements such as integrating machine learning algorithms, expanding the dictionary, or incorporating more advanced NLP models for improved accuracy and context understanding.

CHAPTER 5
CONCLUSION

In conclusion, the Twitter Sentiment Analysis project successfully showcases the practical application of Java in analyzing public opinion on social media. By processing large volumes of raw tweet data through systematic steps—such as cleaning, tokenization, dictionary-based matching, scoring, and classification the system is able to categorize sentiments into positive, negative, and neutral with reasonable efficiency. This demonstrates the potential of automated sentiment analysis in understanding trends, opinions, and reactions of the public on various topics, which can be valuable for businesses, researchers, and policymakers alike.While the dictionary-based approach provides a simple and effective method for sentiment detection, it does face challenges in interpreting nuances such as sarcasm, idiomatic expressions, slang, and context-dependent meanings. These limitations highlight the need for more advanced techniques, such as natural language processing (NLP) algorithms, machine learning models, or hybrid approaches, to improve accuracy and better handle diverse and evolving social media language patterns. Overall, this project not only reinforces key concepts of text analysis and sentiment classification but also serves as a foundational platform for further exploration and enhancement. Future iterations could integrate machine learning classifiers, neural networks, or real-time data streaming to create more robust, scalable, and intelligent systems capable of providing deeper insights into public sentiment. By bridging the gap between basic sentiment analysis and more advanced NLP applications, the project contributes meaningfully to the growing field of social media analytics and its practical applications in real-world decision-making.

REFERENCES:

- Pang and L. Lee, "Opinion Mining and Sentiment Analysis," Foundations and Trends in Information Retrieval, vol. 2, no. 1–2, pp. 1–135, 2008.
- Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," Stanford University, 2009.
- Cambria, E., B. Schuller, Y. Xia, and C. Havasi, "New Avenues in Opinion Mining and Sentiment Analysis," IEEE Intelligent Systems, 2013.
- Hu, M. and B. Liu, "Mining and Summarizing Customer Reviews," Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2004.
- Twitter Developer Platform, "Twitter API Documentation," Twitter Inc., 2025.
- Oracle, "Java Platform Standard Edition Documentation," Oracle Corporation, 2025.
- Aditya Bhushan, Devanshi Dwivedi, Ashutosh Kumar Singh & Snehlata (2024). Analyzing Sentiments on Twitter Using Deep Learning Techniques. International Journal of Modern Education and Computer Science (IJMECS), 16(6).
- Singh, L. G., & Singh, S. R. (2024). Sentiment analysis of tweets using text and graph multi-views learning. Knowledge and Information Systems, 66, 2965–2985.
- A. C. Ojha, P. K. Shah, S. Gupta & S. Sharma (2023). Classifying Twitter Sentiment on Multi-Levels using a Hybrid Machine Learning Model. International Journal of Intelligent Systems and Applications in Engineering, 12(3s), 328–333.
- K. Praveen Kumar, D. Baswaraj, Gumma Parvathi Devi, Sruthi Thanugundala, Ravindra Changala & S. G. G. V. R. (2024). Sentiment Analysis in Social Media Using Deep Learning Techniques. International Journal of Intelligent Systems and Applications in Engineering, 12(3), 1588–1597.

APPENDIX

```
import java.io.*;
import java.nio.file.*;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.stream.Collectors;
public class LargeTwitterSentimentAnalysis {
    public static class Config {
        public static final String DB_CSV = "tweets_db.csv";
        public static final String EXPORT_CSV = "exported_tweets.csv";
        public static final String TRAIN_DATA_FILE = "train_data.csv";
        public static final int DEFAULT_SAMPLE_SIZE = 50;
        public static final String DATE_FORMAT = "yyyy-MM-dd HH:mm:ss";
    }
    public static class TweetRecord {
        private long id;
        private String username;
        private String text;
        private String cleanedText;
        private Date createdAt;
        private String sentiment; // Positive / Negative / Neutral
        private int posScore;
        private int negScore;
        private double classifierConfidence; // for trained classifier

        public TweetRecord() {}

        public TweetRecord(long id, String username, String text, Date createdAt) {
            this.id = id;
            this.username = username;
```

```
        this.text = text;
        this.createdAt = createdAt;
        this.cleanedText = "";
        this.sentiment = "Unknown";
        this.posScore = 0;
        this.negScore = 0;
        this.classifierConfidence = 0.0;
    }

    // getters and setters
    public long getId() { return id; }
    public String getUsername() { return username; }
    public String getText() { return text; }
    public String getCleanedText() { return cleanedText; }
    public Date getCreatedAt() { return createdAt; }
    public String getSentiment() { return sentiment; }
    public int getPosScore() { return posScore; }
    public int getNegScore() { return negScore; }
    public double getClassifierConfidence() { return classifierConfidence; }

    public void setCleanedText(String cleanedText) { this.cleanedText = cleanedText; }
    public void setSentiment(String sentiment) { this.sentiment = sentiment; }
    public void setPosScore(int s) { this.posScore = s; }
    public void setNegScore(int s) { this.negScore = s; }
    public void setClassifierConfidence(double c) { this.classifierConfidence = c; }

    @Override
    public String toString() {
        SimpleDateFormat sdf = new SimpleDateFormat(Config.DATE_FORMAT);
        return
String.format("TweetRecord[id=%d,user=%s,created=%s,text=%s,cleaned=%s,sentiment=%s,pos=%
d,neg=%d,conf=%.2f]",
                id, username, createdAt != null ? sdf.format(createdAt) : "null",
                text, cleanedText, sentiment, posScore, negScore, classifierConfidence);
```

```
    }

    public String toCsvRow() {
        SimpleDateFormat sdf = new SimpleDateFormat(Config.DATE_FORMAT);
        return String.format("%d,%s,%s,%s,%s,%s,%d,%d,%.4f",
                id,
                escapeCsv(username),
                escapeCsv(text),
                escapeCsv(cleanedText),
                createdAt != null ? sdf.format(createdAt) : "",
                sentiment,
                posScore,
                negScore,
                classifierConfidence
        );
    }

    private String escapeCsv(String s) {
        if (s == null) return "";
        String out = s.replace("\"", "\"\"");
        if (out.contains(",") || out.contains("\"") || out.contains("\n")) {
            return "\"" + out + "\"";
        } else return out;
    }

    public static TweetRecord fromCsvRow(String row) {

        List<String> parts = parseCsvRow(row);
        TweetRecord r = new TweetRecord();
        try {
            r.id = Long.parseLong(parts.get(0));
        } catch (Exception e) {
            r.id = new Random().nextLong();
```

```
        }
        r.username = parts.size() > 1 ? parts.get(1) : "";
        r.text = parts.size() > 2 ? parts.get(2) : "";
        r.cleanedText = parts.size() > 3 ? parts.get(3) : "";
        try {
            if (parts.size() > 4 && !parts.get(4).isEmpty()) {
                SimpleDateFormat sdf = new SimpleDateFormat(Config.DATE_FORMAT);
                r.createdAt = sdf.parse(parts.get(4));
            } else r.createdAt = new Date();
        } catch (Exception e) { r.createdAt = new Date(); }
        r.sentiment = parts.size() > 5 ? parts.get(5) : "Unknown";
        r.posScore = parts.size() > 6 ? Integer.parseInt(parts.get(6)) : 0;
        r.negScore = parts.size() > 7 ? Integer.parseInt(parts.get(7)) : 0;
        r.classifierConfidence = parts.size() > 8 ? Double.parseDouble(parts.get(8)) : 0.0;
        return r;
    }
}
    public static class DictionarySentiment {

    private final Set<String> positiveSet;
    private final Set<String> negativeSet;

    public DictionarySentiment() {
        positiveSet = new HashSet<>();
        negativeSet = new HashSet<>();
        // load a large-ish seed dictionary to be more expressive
        loadSeedDictionaries();
    }

    private void loadSeedDictionaries() {
        // You can expand these lists; kept large for a "very big" program
        String[] positives = {
```

```
"good","great","happy","love","excellent","awesome","fantastic","positive","wonderful","excited",
"amazing","nice","best","success","enjoy","beautiful","perfect","smile","brilliant","cool",
"delightful","satisfied","victory","win","pleased","recommend","friendly","optimistic","bright",
            "creative","productive","strong","reliable","trust"
        };
        String[] negatives = {
"bad","sad","angry","hate","terrible","awful","horrible","negative","worst","disappointing",
"poor","fail","ugly","problem","boring","annoying","frustrated","pain","cry","fear","damage",
"loss","weak","broken","slow","rude","complain","dislike","upset","angst","stress","worry"
        };
        positiveSet.addAll(Arrays.asList(positives));
        negativeSet.addAll(Arrays.asList(negatives));
    }

    public String analyze(TweetRecord record) {
        String cleaned = Preprocessor.clean(record.getText());
        record.setCleanedText(cleaned);

        List<String> tokens = Preprocessor.tokenize(cleaned, true, true);
        int pos = 0, neg = 0;
        for (String t : tokens) {
            if (positiveSet.contains(t)) pos++;
            if (negativeSet.contains(t)) neg++;
        }
        record.setPosScore(pos);
        record.setNegScore(neg);

        String sentiment;
        if (pos > neg) sentiment = "Positive (Dict)";
        else if (neg > pos) sentiment = "Negative (Dict)";
```

```
        else sentiment = "Neutral (Dict)";
        record.setSentiment(sentiment);
        record.setClassifierConfidence(calculateConfidence(pos, neg));
        return sentiment;
    }

    private double calculateConfidence(int pos, int neg) {
        int total = pos + neg;
        if (total == 0) return 0.0;
        int diff = Math.abs(pos - neg);
        return (double) diff / (double) total; // normalized confidence
    }

    // Allow runtime dictionary updates
    public void addPositive(String w) { positiveSet.add(w.toLowerCase()); }
    public void addNegative(String w) { negativeSet.add(w.toLowerCase()); }
}
public static class FrequencyClassifier {

    // word -> [posCount, negCount, neutralCount]
    private final Map<String, int[]> wordCounts;
    private int totalPos;
    private int totalNeg;
    private int totalNeu;

    public FrequencyClassifier() {
        wordCounts = new HashMap<>();
        totalPos = totalNeg = totalNeu = 0;
    }

    public void train(List<TrainingExample> trainingData) {
        for (TrainingExample ex : trainingData) {
            List<String> tokens = Preprocessor.tokenize(Preprocessor.clean(ex.text), true, true);
```

```
            for (String t : tokens) {
                int[] arr = wordCounts.getOrDefault(t, new int[3]); // pos, neg, neu
                if (ex.label.equalsIgnoreCase("positive")) {
                    arr[0]++; totalPos++;
                } else if (ex.label.equalsIgnoreCase("negative")) {
                    arr[1]++; totalNeg++;
                } else {
                    arr[2]++; totalNeu++;
                }
                wordCounts.put(t, arr);
            }
        }
    }


    public static class TrainingExample {
        public final String text;
        public final String label;
        public TrainingExample(String text, String label) {
            this.text = text; this.label = label;
        }
    }

    public static class Prediction {
        public final String label;
        public final double confidence;
        public Prediction(String label, double confidence) {
            this.label = label; this.confidence = confidence;
        }
    }
}

//
```

```
public static class TwitterSimulator {

    private final Random rnd = new Random();

    private final String[] userNames = {
            "alice", "bob", "charlie", "dave", "eve", "frank", "grace", "heidi"
    };

    private final String[] positiveTemplates = {
            "I love %s, it's fantastic!",
            "This %s is awesome and amazing.",
            "So happy with the %s, very satisfied.",
            "Best %s I've used so far, great work.",
            "%s made my day, excellent job!"
    };

    private final String[] negativeTemplates = {
            "I hate %s, this is the worst.",
            "Very disappointed with %s, terrible.",
            "The %s is broken and useless.",
            "Worst %s ever, complete fail.",
            "%s ruined my day, so frustrating."
    };

    private final String[] neutralTemplates = {
            "%s is OK, nothing special.",
            "I tried %s today.",
            "The %s has some pros and cons.",
            "Not sure about %s yet.",
            "%s seems average to me."
    };

    private final String[] products = {
```

```
            "phone", "app", "service", "movie", "laptop", "website", "game", "charger", "camera",
"headphone"
        };



                if (first) { first = false; continue; } // skip header
                if (l.trim().isEmpty()) continue;
                try {
                   TweetRecord r = TweetRecord.fromCsvRow(l);
                   list.add(r);
                } catch (Exception e) {


    public static class Exporter {
        public static void exportCsv(List<TweetRecord> records, String filePath) throws IOException {
            List<String> out = new ArrayList<>();

out.add("id,username,text,cleaned_text,created_at,sentiment,pos_score,neg_score,confidence");
            for (TweetRecord r : records) {
               out.add(r.toCsvRow());
            }
            Files.write(Paths.get(filePath), out, StandardOpenOption.CREATE,
StandardOpenOption.TRUNCATE_EXISTING);
        }
    }
    public static class Reporter {


        public static void printBasicStats(List<TweetRecord> list) {
            int total = list.size();
            long pos = list.stream().filter(r ->
r.getSentiment().toLowerCase().contains("positive")).count();
            long neg = list.stream().filter(r ->
r.getSentiment().toLowerCase().contains("negative")).count();
            long neu = total - pos - neg;
            System.out.println("Total tweets: " + total);
            System.out.println("Positive: " + pos);
```

```
        System.out.println("Negative: " + neg);
        System.out.println("Neutral: " + neu);
        System.out.printf("Positive %%: %.2f%%\n", total == 0 ? 0.0 : (pos * 100.0 / total));
        System.out.printf("Negative %%: %.2f%%\n", total == 0 ? 0.0 : (neg * 100.0 / total));
    }

    public static void printTopWords(List<TweetRecord> list, int topN) {
        Map<String, Integer> freq = new HashMap<>();
        for (TweetRecord r : list) {
            List<String> tokens = Preprocessor.tokenize(r.getCleanedText(), true, true);
            for (String t : tokens) freq.put(t, freq.getOrDefault(t, 0) + 1);
        }
        List<Map.Entry<String,Integer>> entries = new ArrayList<>(freq.entrySet());
        entries.sort((a,b) -> Integer.compare(b.getValue(), a.getValue()));
        System.out.println("Top " + topN + " words:");
        for (int i=0;i<Math.min(topN, entries.size());i++) {
            System.out.println((i+1) + ". " + entries.get(i).getKey() + " -> " + entries.get(i).getValue());
        }
    }

    public static void printSampleRecords(List<TweetRecord> list, int n) {
        System.out.println("Sample records (first " + n + "):");
        for (int i = 0; i < Math.min(n, list.size()); i++) {
            TweetRecord r = list.get(i);
            System.out.println((i+1) + ". [" + r.getUsername() + "] " + r.getText());
            System.out.println("   Cleaned: " + r.getCleanedText());
            System.out.println("   Sentiment: " + r.getSentiment() + " (pos=" + r.getPosScore() + ", neg="
+ r.getNegScore() + ", conf=" + String.format("%.2f", r.getClassifierConfidence()) + ")");
        }
    }
}
public static class ConsoleApp {
```

```
    private final Scanner scanner = new Scanner(System.in);
    private final TwitterSimulator simulator = new TwitterSimulator();
    private final CsvDatabase db = new CsvDatabase(Config.DB_CSV);
    private final DictionarySentiment dictAnalyzer = new DictionarySentiment();
    private final FrequencyClassifier freqClassifier = new FrequencyClassifier();

    public void run() {
        boolean running = true;
        while (running) {
            printMenu();
            String choice = scanner.nextLine().trim();
            switch (choice) {
            }
        }
        System.out.println("Goodbye!");
    }

    private void printMenu() {
        System.out.println("\n=== Twitter Sentiment Analysis - Menu ===");
        System.out.println("1. Generate sample tweets and analyze (dictionary)");
        System.out.println("2. Load tweets from text file and analyze (dictionary)");
        System.out.println("3. Load DB and show stats");
        System.out.println("4. Train Frequency Classifier on generated training data");
        System.out.println("5. Analyze recent tweets with Frequency Classifier");
        System.out.println("6. Export DB to CSV");
        System.out.println("7. Clear DB");
        System.out.println("8. Show top words in DB");
        System.out.println("9. Add word to dictionary (pos/neg)");
        System.out.println("0. Exit");
        System.out.print("Choose an option: ");
    }
}
```

```
    public static void main(String[] args) {

        System.out.println("This is a big, self-contained program demonstrating:");
        System.out.println("- tweet simulation");
        System.out.println("- preprocessing");
        System.out.println("- dictionary & frequency classification");
        System.out.println("- CSV persistence and export");
        System.out.println("- interactive console menu");
        System.out.println();
        ConsoleApp app = new ConsoleApp();
        app.run();
    }
}
```