CHAPTER I: INTRODUCTION

1.1 PREAMBLE

From 20th century onwards this WWW has completely changed the way of expressing their views. In present situation people are expressing their thoughts through online blogs, discussion forums and also some online applications like Facebook, Twitter, etc. If we take facebook as our example nearly 2TB of text data is generating within a week in the form of posts. So by this it is understood clearly how this internet is changing the way of living and style of people. Among these posts can be categorized by the hash value tags for which they are Commenting and posting their status. So, now many companies and also the survey companies are using this for doing some analytics such that they can predict the success rate of their product or also they can show the different View from the data that they have collected for analysis. But to calculate their views is very difficult in a normal way by taking these heavy data that are going to generate day by day.

1.2 OBJECTIVE

The objectives of this project are as follows:

- To implement an algorithm for automatic classification of text into positive, negative or neutral.
- Sentimental Analysis to determine the attitude of the mass is positive, negative or neutral towards the subject of interest
- Graphical representation of the sentiment in form of Bar-Chart.

1.3 SCOPE

1.3.1 Functions:

- The scope of the project is to perform opinion analysis on a product or service. Opinion will be classified as positive, neutral, or negative. There will be no in-between classes. for example it does not matter if the sentiment is somewhat positive or extremely positive both will be classified under the same label positive.

- The output of this analysis will be a report on how the product or service is perceived by the target audience. The system will gather data from social media, cleanse the data, and then classify that data. An analysis will then be performed on this classified data.

3.2 Constraints:

- The data which is processed is not live. The data being streamed and stored in the text file is live. But this gathered/collected data is further processed to compute results hence it is not live.

- Social media has users from all over the world but all the posts are not being processed. This is because foreign scripts are not detected. The dictionaries used have only English words with their sentiment scores. Posts with other scripts are filtered out.

3.4 SALIENT FEATURES

Salient features of the project are listed as follows:

1. A system providing real time application using ArrayList.

2. Without using flume and hive

3. Easy to use its functionality.

4. The opinion mining is based on java OOPS.

//1.5 BROAD OUTLINE OF THE PROJECT

The proposed system which is required to design and develop a basic flow of the system is given below where we use and demonstrate our application.

1.6 ABOUT THE SUBSEQUENT CHAPTERS

Chapter 2: Literature Survey

This chapter deals with overall survey part from where we have collected information about the project. Literature Survey provides background material for the project. It gives you various concept details of project.

Chapter 3: Analysis :

Analysis is the first step of software development. The objective of this project is to implement an algorithm for automatic classification of text into positive, negative or neutral. In analysis phase we analyze the project with different perspective and develop the concept, analyze the requirements, select software process model, plan the resources and team organization and analyze the future risk.

Chapter 4: Implementation

The Implementation phase includes

1.Writing java programs and linking it with arraylist

2. matching the sentiments using dictionary

Chapter5:Testing

The testing phase checks for most prone or likely error that might be occur during the execution of program

Chapter 6: Conclusion .

This will contain the abstract idea of project development viz survey, analysis ,planning ,design and scope of project.

Chapter 7:References

It will contain list of resources from where we have taken help in order to develop this project.

CHAPTER 2: LITERATURE SURVEY

2.1 INTRODUCTION

In the past years, many works has been released in opinion mining. Implementation of opinion mining has been carried out for a variety of applications over a wide range of classification algorithms and for varying data size. There exist many possible variants; some of them are discussed in following section. Due to the size nature of the dataset (i.e., 2 billion posts), the experiments were conducted on a High Performance Computing (HPC) platform using ArrayList, which exhibits the trend of big data analytics. The results suggest that daily-life social networking data could help early detection of important patient safety issues.

The data set used is a collection of tweets collected from twitter application, from which they try to identify potential adverse events caused by drugs of interest. The collected stream of posts was organized by a timeline. The raw social media content was crawled using the twitter user timeline API that contains information about the specific post and the user. The work is indexed only with the following four fields for each post:

l. tweet id that uniquely identifies each tweet;

2. User identifier associated with each tweet;

3. Timestamp of the tweet

4. The Tweet text.

2.2 ArrayList

The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed.

Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold.

Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Following is the list of the constructors provided by the ArrayList class.

Sr.No.	Constructor & Description
1	**ArrayList( )** This constructor builds an empty array list.
2	**ArrayList(Collection c)** This constructor builds an array list that is initialized with the elements of the collection **c**.
3	**ArrayList(int capacity)** This constructor builds an array list that has the specified initial capacity. The capacity is the size of the underlying array that is used to store the elements. The capacity grows automatically as elements are added to an array list.

Apart from the methods inherited from its parent classes, ArrayList defines the following methods −

Sr.No.	Method & Description
1	**void add(int index, Object element)** Inserts the specified element at the specified position index in this list. Throws IndexOutOfBoundsException if the specified index is out of range (index < 0 ‖ index > size()).
2	**boolean add(Object o)**

	Appends the specified element to the end of this list.
3	**boolean addAll(Collection c)** Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator. Throws NullPointerException, if the specified collection is null.
4	**boolean addAll(int index, Collection c)** Inserts all of the elements in the specified collection into this list, starting at the specified position. Throws NullPointerException if the specified collection is null.
5	**void clear()** Removes all of the elements from this list.
6	**Object clone()** Returns a shallow copy of this ArrayList.
7	**boolean contains(Object o)** Returns true if this list contains the specified element. More formally, returns true if and only if this list contains at least one element **e** such that (o==null ? e==null : o.equals(e)).
8	**void ensureCapacity(int minCapacity)** Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.
9	**Object get(int index)** Returns the element at the specified position in this list. Throws

	IndexOutOfBoundsException if the specified index is out of range (index < 0 || index >= size()).
10	**int indexOf(Object o)** Returns the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
11	**int lastIndexOf(Object o)** Returns the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
12	**Object remove(int index)** Removes the element at the specified position in this list. Throws IndexOutOfBoundsException if the index out is of range (index < 0 || index >= size()).

2.3 Java

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

Java is −

- **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed** − Java is designed for the distributed environment of the internet.

- **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

//2.4.2 Algorithm

The mapper emits an intermediate key-value pair for each word in a docmnent.

1: class Mapper

2: method Map(ByteOffset no; Text Line)

3: for all terms t Belongs To Line do

4:Generate (term t; count 1)

The reduce sums up all counts for each term

1:class Reducer

2: method Reduce(term t;counts[1;1;1;….])

3: sum=0

4: for all count c belongs to counts [1;1;1;…]
Do

5: sum+=c;

6:Generate(term t; count sum)

CHAPTER 3: ANALYSIS

The aim of the analysis phase is to specify and define the system to be built. The purpose is to formulate the problem and to build models to solve the problem. The models developed will describe what the system is going to do. The basis of this modeling is basically the requirements of the system. In analysis phase it is possible to build the models that will make the understanding of the system easier. The models that are developed during analysis are fully application oriented. It helps to solve problem under ideal conditions. So, in analysis phase, two types of model will be built, which are requirement model and analysis model.

3.1 CONCEPTION PHASE

Over the past decade humans have experienced exponential growth in the use of online resources, in particular social media and micro blogging websites such as Twitter, Facebook and YouTube. Many companies and organizations have identified these resources as a rich mine of marketing knowledge. Traditionally companies used interviews, questionnaires and surveys to gain feedback and insight into how customers felt about their products. These traditional methods were often extremely time consuming and expensive and did not always return the results that the companies were looking for due to environmental factors and poorly designed surveys.

Natural language processing and opinion mining are playing an increasingly important role in making educated decisions on marketing strategies and giving valuable feedback on products and services. There are massive amounts of data containing consumer sentiment uploaded to the internet every day, this type of data is predominantly unstructured text that is difficult for computers to gain meaning from. In the past it was not possible to process such large amounts of unstructured data but now with computational power following the projections of Moore's law and distributed networks of computers using Arraylist, massive datasets can be now processed with relative ease. Major investment is going into this area such as IBMs tireless research into their Natural Language Processing supercomputer Watson and Googles recent acquisition of deep mind technology. With further research and investment into this area machines will soon be able to gain an "understanding" from text which will greatly improve data analytics and search engines.

3.1.1 FEASIBILITY STUDY

3.l.l.l Technical Feasibility:

It considers the technical requirements of the proposed project. The project is considered technically feasible if the internal technical capability is sufficient to support the project requirements.

3.1.1.2 Operational Feasibility:

Our system can be used easily by people having basic knowledge of computer& internet, hence making the system being developed to be used efficiently.

3.1.2 PROBLEM DEFINITION

With the explosive growth of data available on the World Wide Web, the sheer volume of trellis is kept on the enterprise's Web site. Discovery and analysis of useful information from the World Wide Web becomes a practical necessary. Web Log Mining is the application of Data Mining techniques to discover Client behavior patterns from Web data in order to understand and serve the needs of Web. In this project we will introduce the method how to use data mining technique to excavate the client behavior pattern from Web log file, and play an emphasis on analyzing client behavior pattern recognition system and its application, so as to help enterprises obtain client information conveniently and automatically.

3.2 REQUIREMENT ANALYSIS

3.2.1 REQUIREMQNT SPECIFICATION

Requirement analysis is a software engineering task that bridges the gap between the system level system engineering and software design.

3.2.2 Goal

- To implement an algorithm for automatic classification of text into positive, negative or neutral.
- Opinion mining to determine the attitude of the mass is positive, negative or neutral towards the subject of interest.
- Graphical representation of the sentiment in form of Bar~Chart.

3.2.3 Scope

- The scope of the project is to perform a Opinion mining on a product or service. Opinion will be classified as positive, neutral, or negative, there will be no in-between classes for example it does not matter if the sentiment is somewhat positive or extremely positive both will be classified under the same label positive
- The output of this analysis will be a report on how the product or service is perceived by the target audience. The system will gather data from social media, cleanse the data, and then classify that data. An analysis will then be performed on this classified data.

3.2.5VALIDATION CRITERIA

- To validate the progress and execution of the project, the following criteria should be taken into
- Account.
- The user requirements are complete.
- The document is consistent i.e., there are no contradictions or incompatibility in requirements.
- All the facts are correctly recorded in the document.
- There is no ambiguity in requirements.

3.2.6 PERFORMANCE EVALUATION

- The system should perform according to users' need.
- The system should be user friendly.
- The performance of the system can be evaluated by commanding the system and its response in return.

3.3 SOFTWARE PROCESS MODEL

3.3.1 MODEL USED

Some Process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution. Such Models can be used to develop more precise and formalized descriptions of software life cycle activities.

In our project, We have used Incremental Model.

3.3.2 DESCRPITION

incremental model is a method of software development where the model is designed, implemented and tested incrementally until the product is finished. It involves both development and maintenance The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, implementation and testing phases. A working version of software is produced during the first module, so you have working software early on during the software life cycle. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

Process Model

3.3.3 ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk-High risk part is done first.

- With every increment operational product is delivered.

- Issues, challenges & risks identified from each increment can be utilized/applied to the next increment.

DISADVANTAGES

- More resources may be required.

- Although cost of change is lesser but it is not very suitable for changing requirement

- More management attention is required.

- Each phase of iteration is rigid with no overlaps

- System architecture or design issues may arise because not all requirements are gathered up for the entire life cycle.

- Does not allow iterations within an increment.
- Defining increments may require definition of the complete system.

3.3.4 WHY THIS MODEL IS USED?

- One of the advantages of using this approach is that the design flexibility allows changes to be implemented at several stages of the project and quality assurance with the iterative process.

- Major requirements must be defined; however, some details can evolve with time.

- This approach starts with minimal requirement and develops with more information throughout the life cycle.

- Resources with needed skill set are not available.

3.4 PLANNING PHASE

3.4.1 Project resource

Development environment:
a).Hardware Tools: PC.
b) Netbeans

Human resource:
a). Project Guide: Prof. Sonu Airen

b) Project Members:

Akshay Gupta
Shivam Shukla
Krishnakant Bhargava

3.4.2 PROJECT TEAM ORGANISATION

Our project team organization is both formal and informal. All the team members have formal relationship with the project guide and informal relationship with each other.
The team organization of the team is centrally controlled .The team leader manages top level problem solving and internal team coordination.
Communication between the team leader and the team members is vertical.

3.5 RISK ANALYSIS

A risk is potential problem-it might happen or might not. But regardless of the outcome, it is a really good idea to identify it, assess its probability of occurrence, estimate its impact and establish a contingency plan should the problem actually occurs. When risks are analyzed, it is important to quantify the level of uncertainty and the degree of loss associated with each risk .To accomplish this different categories of risks are considered.

The different risks involved are:

a) Schedule Risks

Project schedule get slip when project tasks and schedule release risks are not addressed properly Schedule risks mainly affect on project and finally on company economy may lead to project failure.

Schedules often slip due to following reasons:

- Wrong time estimation.
- Resources are not tracked properly. All resources like staff, systems, skills of Individual etc.
- Failure to identify complex functionalities and time r required to develop those functionalities

- unexpected project scope expansions.

b) Budget Risks

- Wrong budget estimation.
- Cost overruns.
- Project scope expansion.

c) Operational Risks

Risks due to improper process implementation, failed system or some external events risks come wider the category of operational risk Causes of Operational risks:

- Failure to address priority conflicts.
- Failure to resolve the responsibilities.
- Insufficient resources.
- No proper subject training.
- No resource planning.
- No communication in team.

d) Technical Risks

These are the risks that involve the design, implementation, interface and maintenance threats to the software, thus it takes into account the reliability of the project and its security.

- **Loss of critical data**: Due to some error there might be a condition that the information being managed by the system is lost leading to severe problem to the user. Thus, to prevent it, a backup system must be provided so that the even though critical data is lost but the user does not face potential troubles.
- **The interface**: The interface provided to the user may not be according to his skills and understandability. Thus, it may impose the threat to its acceptance. Thus, the interface must be as user friendly as possible mitigating the risk to its usage.
- **Low quality of output**: The system may produce output in time larger than the user expectation thus, leading to user frustration and here further rejection. The risk could be lessened by examining the place of deployment and usage and also specifying the processor speed requirements and space requirements.

e) Programmatic Risks

These are the external risks beyond the operational limits. These are all uncertain risks are outside the control of the program. These external events can be:

- Running out of fund
- Market development
- Changing customer product strategy and priority,
- Government rule changes.

f) Project Risks

These risks are directly related with the project and its development. Thus, it involves risk in the implementation of the project as schedule, resources and complexity problems.

- **Schedule:** The development process requires time to analyze the problem and design a solution with its implementation and testing. But there is risk to complete it in certain time constraints.

- **Resources**: The resources required by the project may not be available immediately and may involve certain hardware problems as working with mobiles etc.

- **Size:** The project must be such that it must incorporate cohesion between the different components of the system. This is to make the code simpler and less clumsy.

- **Training:** The project must be well understood by the customers and hence be accordingly trained for it so that they could use it well for their convenience. The system risk is very high in this regard.

g) Personnel risks:

These risks are related to the stake holders involved in the project. It mainly relates to the developers of the project.

- **Unavailability:** The team member may not be temporarily available. Thus, the work of the system on the member will be pending and may introduce delay. Thus, it can be mitigated through other team member balancing the load and hence avoiding delay.

- **Unaware:** The team member may be well versed with certain technology and hence could be reluctant to opt to it. Thus, it could be mitigated by other members coping w or him learning the same.

CHAPTER 4: IMPLEMENTATION

The implementation phase is remarked by the actual execution of the plan prepared to develop the application and in actual combines all the resources and information gathered so far to produce the desired real world entity that has been the aim since the beginning i.e. the software. Our implementation encompasses all the process required to develop java programs so that it operates properly and effectively in its deployed environment and also adapt to some common environmental changes that occurs when used with different system. All other phases, beginning from the requirement analysis to design phase provide backbone to the implementation phase.

TECHNOLOGY

JAVA:

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

James Gosling initiated Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called 'Oak' after an oak tree that stood outside Gosling's office, also went by the name 'Green' and ended up later being renamed as Java, from a list of random words.

Sun released the first public implementation as Java 1.0 in 1995. It promised **Write Once, Run Anywhere** (WORA), providing no-cost run-times on popular platforms.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

Tools You Will Need

You will also need the following softwares −

- Linux 7.1 or Windows xp/7/8 operating system
- Java JDK 8
- Microsoft Notepad or any other text editor

ARRAYLIST:

The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed.

Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold.

Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

//Implementation Strategy :

Steps of Implementation:

Main class(main.java)

```
package major2;
import java.awt.Color;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;
import java.io.IOException;
import java.util.Scanner;

import javax.swing.JFrame;

import twitter4j.TwitterException;

public class MainClass {
	public static void main(String[] args) throws TwitterException, IOException{
		MainClass algo = new MainClass();
		algo.analyze();
		//algo.populateTextFile();
	}//end main

	public void analyze() throws IOException, TwitterException{
		//Create frame for GUI
		GUIFrame gui = new GUIFrame(); // create EventsFrame
		gui.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
		gui.setSize( 1300, 800 ); // set frame size
	   gui.setBackground(Color.yellow);
		gui.setVisible( true ); // display frame
```

```
	}//end method


	public void populateTextFile() throws TwitterException, IOException{//this function is
never called
		String happyFace = ":)";
		String sadFace = ":(";
		String happyFace2 = ":-)";
		String sadFace2 = ":-(";
		TwitterData twitter = new TwitterData();
		String searchTerm = ":(&lang:en";
		ArrayList<String> tweets = twitter.search(searchTerm);
		for(int i=0; i<tweets.size(); i++){
			StringTokenizer text = new StringTokenizer(tweets.get(i));
			String newString = "";
			while(text.hasMoreTokens()){
				String tempString = text.nextToken();
				if(!tempString.contains("#") && !tempString.contains("@") &&
!tempString.contains("http://t.co") && !tempString.contains("https://t.co") &&
!tempString.contains(happyFace) && !tempString.contains(sadFace) &&
!tempString.contains(":p") && !tempString.contains(":P") && !tempString.contains(":D") &&
!tempString.contains(sadFace2) && !tempString.contains(happyFace2) &&
!tempString.contains("RT") && !tempString.contains("♥") && !tempString.contains("<3") &&
!tempString.contains("rt")){
					newString = newString+" "+tempString;
				}//end if
			}//end while
			//writeToFile(newString);//ommited
			System.out.println(newString);
			//System.out.println(" ");
		}//end for
	}//end method

}//end class
```

Twitter.java

```
package major2;

import java.util.*;

import twitter4j.*;
import twitter4j.auth.AccessToken;

import java.io.IOException;
```

```
import java.util.ArrayList;

public class TwitterData {
        private final static String CONSUMER_KEY = "OXDxV358lVWSqn2UtXr4hLkq2";
   private final static String CONSUMER_KEY_SECRET =
"IFL4VB8HQPSrREErw1g1mv08VArNCIf0ocR9pF7A2EGUttJ0H7";

   public ArrayList<String> search(String searchTerm) throws TwitterException, IOException{
                //Get tweets and return an array of them
        Twitter twitter = new TwitterFactory().getInstance();
        twitter.setOAuthConsumer(CONSUMER_KEY, CONSUMER_KEY_SECRET);

        String accessToken = getSavedAccessToken();
        String accessTokenSecret = getSavedAccessTokenSecret();

      AccessToken oathAccessToken = new AccessToken(accessToken,accessTokenSecret);
        twitter.setOAuthAccessToken(oathAccessToken);

        //Search twitter
        Paging paging = new Paging(1, 200);

      Query query = new Query(searchTerm);
      QueryResult result = twitter.search(query);
      ArrayList <String>text = new ArrayList<>();
      result.getTweets().stream().forEach((status) -> {
            text.add(status.getText());
        });
      return text;
   }//end method

        public String[] retrieveTweets() throws TwitterException, IOException{
                //Get tweets and return an array of them
        Twitter twitter = new TwitterFactory().getInstance();
        twitter.setOAuthConsumer(CONSUMER_KEY, CONSUMER_KEY_SECRET);

        String accessToken = getSavedAccessToken();
        String accessTokenSecret = getSavedAccessTokenSecret();
        AccessToken oathAccessToken = new AccessToken(accessToken,accessTokenSecret);
        twitter.setOAuthAccessToken(oathAccessToken);

        //Post to twitter
        System.out.println("\nAnalyzing...");

        // Paging, The factory instance is re-useable and thread safe.
        List <String> tweets = new ArrayList<>();
      // requesting page 2, number of elements per page is 40
```

```
        Paging paging = new Paging(1, 200);
        ResponseList<twitter4j.Status> statuses = twitter.getHomeTimeline(paging);
        statuses.stream().forEach((status) -> {
              tweets.add(status.getText());
            });

        String[] stockArr = new String[tweets.size()];
        stockArr = tweets.toArray(stockArr);
        return stockArr;
            }//end method

            private String getSavedAccessTokenSecret() {
            //Get saved access token secret
            return "KWymIJnmghArdmgCyRIE90MDzXMb90R9IhRjKam7P9Ljl";
    }//end method

    private String getSavedAccessToken() {
            //Get saved access token
            return "1478449470-N5XHPwEA4iot5734PqlKAqD7Zaf9miFJ2bOxBDC";
    }//end method
}
```

GUI.java

```
package major2;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;

import twitter4j.TwitterException;

import javax.jws.soap.SOAPBinding.Style;
import javax.swing.*;
import javax.swing.text.BadLocationException;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;

import twitter4j.TwitterException;

public class GUIFrame extends JFrame implements ActionListener{
```

```
private JScrollPane scrollPane; //ScrollPane used to display TestAreas
//private JTextArea outputJTextArea; //Text area to display text
private JTextPane textPane; //Text area to display text
private JPanel panel1; //Panel that displays other GUI components
private JTextField textField; //Where user inputs term to search twitter for
private JButton searchButton; //Button to press to start search for tweets
private JLabel positiveLabel;
private JLabel positivePercentLabel; //Display percent of tweets that are positive
private JLabel negativeLabel;
private JLabel negativePercentLabel; //Display percent of tweets that are negative
private Color greenColor = new Color(18,149,18);

public GUIFrame(){
        super("Frame");

        /*
        //Initialize text area
        outputJTextArea = new JTextArea(10,30);
        outputJTextArea.setLineWrap( true );
   outputJTextArea.setEditable( false );
   outputJTextArea.setBackground( Color.WHITE );
   outputJTextArea.setForeground( Color.BLACK );
   */
//Initialize text pane
   textPane = new JTextPane();
   textPane.setEditable( false );
   textPane.setBackground( Color.WHITE );
   textPane.setForeground( Color.BLACK );

   //Attaches output of JTextArea to JScrollPane
        scrollPane = new JScrollPane(textPane);

        //Labels
        positiveLabel = new JLabel("Positive");
        positiveLabel.setForeground(greenColor);
        positivePercentLabel = new JLabel("");
        positivePercentLabel.setForeground(greenColor);
        negativeLabel = new JLabel("Negative");
        negativeLabel.setForeground(Color.RED);
        negativePercentLabel = new JLabel("");
        negativePercentLabel.setForeground(Color.RED);

        //TextField
        textField = new JTextField(20);
        textField.addActionListener(this);
```

```
		//Search button
		searchButton = new JButton("Search");
		searchButton.addActionListener(this);

		//Create panel to add GUI components to JPanel
		panel1 = new JPanel();
		panel1.add(positiveLabel);
		panel1.add(positivePercentLabel);
		panel1.add(negativeLabel);
		panel1.add(negativePercentLabel);
		panel1.add(textField);
		panel1.add(searchButton);

		//Add ScrollPane and Panel to JFrame
		setLayout( new BorderLayout() );
		add(scrollPane, BorderLayout.CENTER);
		add(panel1, BorderLayout.SOUTH);
}//end constructor

/*
 * Required implementation of ActionListener interface
 */
@Override
public void actionPerformed(ActionEvent e){
		// TODO Auto-generated method stub
		textPane.setText("");
		if(e.getSource() == searchButton){
			try {
				searchAndAnalyze();
			} catch (TwitterException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}//end try
		}//end if
}//end method

/*
 * Retrieves tweets, analyzes sentiment and displays results in GUI
 */
private void searchAndAnalyze() throws TwitterException, IOException{
		double positiveCount = 0;
		double negativeCount = 0;
```

```
                //Get Tweets given search term
                TwitterData twitter = new TwitterData();
                String searchTerm = textField.getText();
                searchTerm = searchTerm+"&lang:en";
                ArrayList<String> tweets = twitter.search(searchTerm);

                //Analyze sentiment of tweets
                TestData test = new TestData();

                //JTextPane styles
                StyledDocument doc = textPane.getStyledDocument();
        javax.swing.text.Style style = textPane.addStyle("I'm a Style", null);
        DecimalFormat numberFormat = new DecimalFormat("#.##");

        //Loop through each tweet, calculate sentiment and format text color appropriately
                for(int i=0; i<tweets.size(); i++){
                        double positivePercent = test.textSentiment(tweets.get(i),"positive");
                        double negativePercent = test.textSentiment(tweets.get(i),"negative");
                        //outputJTextArea.append(tweets.get(i)+"\n\n");
                        if(negativePercent > positivePercent){
                                negativeCount++;
                                StyleConstants.setForeground(style, Color.red);
                    try { doc.insertString(doc.getLength(),i+1+" "+tweets.get(i)+"\n\n",style); }
                    catch (BadLocationException e){}
                                //outputJTextArea.setForeground(Color.RED);
                        }else{
                                positiveCount++;
                                StyleConstants.setForeground(style, greenColor);
                    try { doc.insertString(doc.getLength(),i+1+" "+tweets.get(i)+"\n\n",style); }
                    catch (BadLocationException e){}
                                //outputJTextArea.setForeground(Color.GREEN);
                        }//end if

                        //Show Analysis on labels

        positivePercentLabel.setText(""+numberFormat.format(((positiveCount/(i+1))*100))+"%
");

        negativePercentLabel.setText(""+numberFormat.format(((negativeCount/(i+1))*100))+"
%");

                        //System.out.println(i+" Positive: "+positivePercent+" Negative:
"+negativePercent);
                        //System.out.println(tweets.get(i));
                        //System.out.println(" ");
                }//end for
```

```
        }//end method
}//end class
```

Algorithm

```
BEGIN

For each tweet Ti
{
For each word Wj in Ti that exist in dictionary
{
If  polarity[Wj]==negative
{
 Return negative;
}
Else
{
If  polarity[Wj]==positive
{
 Return positive;
}
 else{
 return neutral;
}
}
}
}
END
```

// for finding sentiment

```
BEGIN
For each tweet Ti
{
Sentiment =0;
For each word Wj
{
If polarity[Wj]==negative
{
Sentiment--;
}
Else
{
If polarity[wj]=positive
{
```

```
Return sentiment++;
}
Else
If polarity[wj]==neutral
{
Sentiment+=0;
}
}}}}

If sentiment of Ti>0
{
Sentiment=positive;
}
Else if  sentiment of Ti<0
{
 Sentiment= negative;
}
Else
{
Sentiment=neutral;
}
Return sentiment;
}
END
```

CHAPTER 5: TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

Validation is process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software ?".
- Validation emphasizes on user requirements.

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications?"
- Verifications concentrates on the design and system specifications.

5.1 Black-box testing

It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.

In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Black-box testing techniques:

- **Equivalence class** - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.

//screenshots

CHAPTER 6: CONCLUSION

Social media now a days became one of the major types of the communication. It is seen as online word-of-mouth branding. The large amount of information available on Social Media makes it an effective source of data for opinion mining. In our project, first we included API of social media and get the essential information such as access token id and download posts then we have performed analysis and categorize tweets into positive, negative and neutral sets. The algorithm is able to determine positive, negative and neutral sentiments of posts. The algorithm is based on dictionary approach. As a future work, we plan to collect a multilingual set of Social media data and compare the characteristics of the data set across different languages.

6.1 Scope:

This program based is developed keeping programmer that they can easily and effectively learn to work With large databases. Thus Application can be used in industrial area.

6-2 Future Enhancement:

It is an emerging technology in future will be used by industries dealing with large datasets.