# Twitter Sentiment Analysis Web App

**FOURTH SEMESTER**

**MAIN PROJECT REPORT**

*Submitted to the University of Kerala*

*In partial fulfilment of the requirements for the award of the degree of*

*Master of Computer Applications*

*Submitted By*

**Haritha S (95520821035)**

Department of Computer Applications

**SREE NARAYANA INSTITUTE OF TECHNOLOGY**

Vadakkevila, Kollam – 10

www.snit.ac.in

**2022**

**Sree Narayana Institute of Technology**

**Vadakkevila, Kollam 10**

**Department of Computer Applications**

# Certificate

Certified that the project report entitled

"Twitter Sentiment Analysis Web App"

Is a bonafide report of the project

done by

Haritha S
(95520821035)

During the year 2022 in partial fulfilment of the requirement for

the award of the degree of Master of Computer Applications

by University of Kerala

| | | |
|---|---|---|
| *Examiner* | *Guide* | *Head of Dept* |
| 1. | Redhya M<br>Assistant Professor | Dr.Sajeev J |

## DECLARATION

I, Haritha S, hereby declare that this project report entitled **‘Twitter Sentiment Analysis WebApp’** is the bonafide work carried out under the supervision of **Ms.Redhya M, Asst.Professor**,Sree Narayana Institute of Technology, Kollam. Declared further, that to the best of my knowledge, the work reported here does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion to any other candidate. The content of this report is not being presented by any other student to this or any other University for the award of a degree.

Signature
Haritha S (Reg.No. 95520821035)

Countersigned

Head of Department, Master of Computer Application
Sree Narayana Institute of Technology, Kollam

## ACKNOWLEDGEMENT

First and foremost, I thank the Almighty God who gave me the knowledge and strength to complete this project successfully. I express my deep sense of gratitude to The Principal **Dr.T.Mahalekshmi** and **Dr.Sajeev.J**, Head, Department of Computer Applications for providing me with all the necessary facilities, which helped me a lot in the successful completion of this project. It is a great pleasure for me to acknowledge the assistance of my project guide **Ms.Redhya M**, Assistant Professor, Sree Narayana Institute of Technology, Kollam, for her valuable advice and kind support during the period of this project.

Haritha S (95520821035)

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

## ABSTRACT

This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with over 200 million registered users- out of which 100 million are active users and half of them log on twitter on a daily basis - generating nearly 250 million tweets per day. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analyzing the sentiments expressed in the tweets. Analyzing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like the stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream.

# CHAPTER- 1
# INTRODUCTION

## 1.1 An Overview

I have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of the public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analyzing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm's stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market are having a favorable response and in which a negative response (since twitter allows us to download a stream of geo-tagged tweets for particular locations. If firms can get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis. One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections which concluded that twitter is a good reflection of offline sentiment .

This project of analyzing sentiments of tweets comes under the domain of "Pattern Classification" and "Data Mining". Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering "useful" patterns in a large set of data, either automatically (unsupervised) or semi automatically (supervised). The project would heavily rely on techniques of "Natural Language Processing" in extracting significant patterns and features from the large data set of tweets and on "Machine Learning" techniques for accurately classifying individual unlabelled data samples (tweets) according to whichever pattern model best describes them. The features that can be used for modeling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word "excellent" has a strong positive connotation while the word "evil" possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are
that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactic approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb,

interjection, etc. Patterns can be extracted from analyzing the frequency distribution of these parts of speech (either individually or collectively with some other part of speech) in a particular class of labeled tweets. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word lengthening , question marks, presence of url in tweets, exclamation marks,internet emoticons and internet shorthand/slangs. Classification techniques can also be divided into two categories: Supervisedvs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labeled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labeled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabeled data sample according to whichever pattern best describes it. For example if we come up with a highly simplified model that neutral tweets contain 0.3 exclamation marks per tweet on average while sentiment-bearing tweets contain 0.8, and if the tweet we have to classify does contain 1 exclamation mark then (ignoring all other possible features) the tweet would be classified as subjective, since 1 exclamation mark is closer to the model of 0.8 exclamation marks. Unsupervised classification is when we do not have any labeled data for training.

In addition to this adaptive classification techniques deal with feedback from the environment. In our case feedback from the environment can be in the form of a human telling the classifier whether it has done a good or poor job in classifying a particular tweet and the classifier needs to learn from this feedback. There are two further types of adaptive techniques: Passive and active. Passive techniques are the ones which use the feedback only to learn about the environment (in this case this could mean improving our models for tweets belonging to each of the three classes) but not using this improved learning in our current classification algorithm, while the active approach continuously keeps changing its classification algorithm according to what it learns at real-time. There are several metrics proposed for computing and comparing the results of our experiments. Some of the most popular metrics include: Precision, Recall, Accuracy, F1-measure, True rate and False alarm rate (each of these metrics is calculated individually for each class and then averaged for the overall classifier Project Thesis Report 12
performance.) A typical confusion table for our problem is given below along with an illustration of how to compute our required metric.

| | Machine says yes | Machine says no |
|---|---|---|
| Human says yes | tp | fn |
| Human says no | fp | tn |

Table 1: A Typical 2x2 Confusion Matrix

$$\text{Precision(P)} = \frac{tp}{tp+fp} \qquad \text{Recall(R)} = \frac{tp}{tp+fn} \qquad \text{Accuracy(A)} = \frac{tp+tn}{tp+tn+f+fp+fn}$$

$$\text{F1} = \frac{2.P.R}{P+R} \qquad \text{True Rate(T)} = \frac{tp}{tp+fn} \qquad \text{False-alarm Rate(F)} = \frac{fp}{tp+fn}$$

## 1.2 Problem statement and project relevance

Sentiment Analysis Dataset Twitter has a number of applications:

**Business:** Companies use Twitter Sentiment Analysis to develop their business strategies, to assess customers' feelings towards products or brands, how people respond to their campaigns or product launches and also why consumers are not buying certain products.

**Politics:** In politics Sentiment Analysis Dataset Twitter is used to keep track of political views, to detect consistency and inconsistency between statements and actions at the government level. Sentiment Analysis Dataset Twitter is also used for analyzing election results.

**Public Actions:** Twitter Sentiment Analysis also is used for monitoring and analyzing social phenomena, for predicting potentially dangerous situations and determining the general mood of the blogosphere.

## 1.3 Aim and scope of the project

This is a web app made using Python and Django Framework. It has a registration system and a dashboard. Users can enter keywords to retrieve live Twitter text based on the keyword, and analyze it for customer feelings and sentiments. This data can be visualized in a graph. This project, in particular, mines data using a popular "Tweepy" API. Tweepy API connects to Twitter in real-time and gathers metadata along with the text from the Twitter platform.

**Purpose:**

- To help companies study the customer sentiment around a particular product.
- To help system users analyze a huge amount of data, quickly and efficiently.
- To segregate customer sentiment on a scale of -1 to 1, where -1 represents a strong negative sentimentality towards the keyword(s), and 1 represents a strongly positive reaction.
- To visualize the collected data clearly and effectively.

# CHAPTER 2
# SYSTEM ANALYSIS

## 2.1 Existing System

The existing system „Sentiment Analysis" takes the static data which is already extracted from a social media platform. The data extracted is stored in a csv file or Excel file which is the input to the program or application. For each statement the program analyses, the output would be a floating-point number which is termed as polarity. The polarity values range from -1 to +1. Based on the polarity obtained the program determines the emotion of the statement.

• The emotion is classified as positive, negative, neutral.
• If polarity>0 then the emotion is positive.
• If polarity= 0 then the emotion is neutral.
• If polarity<0 then the emotion is negative

**Limitations of Prior Art:**

Sentiment analysis in the domain of micro-blogging is a relatively new research topic so there is still a lot of room for further research in this area. Decent amount of related prior work has been done on sentiment analysis of user reviews , documents, web blogs/articles and general phrase level sentiment analysis . These differ from twitter mainly because of the limit of 140 characters per tweet which forces the user to express opinion compressed in very short text.

The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labeling required for the supervised approach is very expensive. Some work has been done on unsupervised and semi-supervised approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need for proper and formal comparisons between these results arrived through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications.

## 2.2 Proposed System

Twitter sentiment analysis web application which performed real-time sentiment analysis of Twitter on tweets that matched particular keywords provided by the user. For example if a user is interested in performing sentiment analysis on tweets which contain the word "Obama" he / she will enter that keyword and the web application will perform the appropriate sentiment analysis and display the results for the user.

For acquiring tweets from Twitter we used the REST API in this case .Twitter REST API provides access to tweets up to around 5 days in the past according to the search query we specify. If we used the Twitter Streaming API and the user specified a keyword which is not very common in Twitter, the web application may have to wait for a long time to acquire enough tweets to display reasonable results. In contrast to this it is much simpler to acquire the tweets in a couple of simple URL calls to the Twitter REST API. One limitation of The REST API however is that one call can only give us a maximum of 100 results. Since we apply sentiment analysis on the past 1,000 tweets on any search query (given that there are that many tweets matching with the keyword available), we have to basically call the API 10 times to get the required number of tweets. This is the basic source of processing delay in our web application.

## 2.3 Advantage of Proposed system

- To help companies study the customer sentiment around a particular product.
- To help system users analyze a huge amount of data, quickly and efficiently.
- To segregate customer sentiment on a scale of -1 to 1, where -1 represents a strong negative sentimentality towards the keyword(s), and 1 represents a strongly positive reaction.
- To visualize the collected data clearly and effectively.

## 2.4 Feasibility study

The main objective of feasibility study is to test the technical, social and economic feasibility of developing a system. This is done before developing a system. This is done by investigating the existing system in the area under investigation and generating ideas about the new system.

Twitter sentiment analysis web application is a game-based web application developed to get quality insights from twitter by creating the web pipeline.

### 2.5.1 Technical feasibility

In Technical feasibility, the main issue is to determine whether the current level of technology can support the proposed system. The proposed system can be developed using the latest technology Python and considering all the advantages of the proposed system it will be technically feasible in using this technology which is currently available and also the profit as a serious constraint, we recommended to use the proposed system. Technologies used in the system are Python and Django.

By using the Django technology we can build the web application structure and add more features efficiently.

### 2.5.2 Economical feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of the proposed system. Considering the benefits and savings that are expected from the proposed system, the administrators decided to design and implement a new system.

By using the Django Framework technology, the system can be developed at low cost and the cost of hardware is also less. Because of Django adding new features, a new effective admin panel etc. will be economically feasible in future as well.

### 2.5.3 Operational feasibility

Proposed projects would be beneficial only if they can be turned into an information system that will meet the organization's operating requirements. One of the main problems faced during the development of a new system is getting acceptance from users. Proposed projects are beneficial because they can be turned into information systems that will meet the user requirements. The test of operational feasibility asks if the system will work when developed and installed.

The tools selected for developing the software are Python, Django development server, Windows 11 which are all readily / freely available in the market. The report generation module which has a direct interface with the end user (viewer) can run in a windows environment. It is coded in such a way that not much change occurs while operations are done. The system is menu driven and meaningful text is given as hyperlinks, it is user friendly and easy for the staff as well as the users to operate.

On the Twitter sentiment analysis web application user can access the application using a standard browser. Using the username and the password every user can login to the system from anywhere across the globe. Admin is also able to access the system from anywhere and manage the users efficiently.

### 2.5.4 Behavioral feasibility

In Behavioral feasibility, the management considers that the proposed system will fulfill the requirements of the users, i.e. whether the proposed system covers all the jobs that were expected by the end user and whether it has considerable improvements. Understanding the advantages and efficiency of the proposed system, we decided to develop a new system. Since the new proposed system has nothing to do with the current users, worker resistance to the system is very much less. The proposed system helps to avoid the delay in processing the data thereby reducing the amount of time consumed.

## 2.6 Module Description

There are four modules in the MathMate web application they are;

### 1)Admin

- **Manage Users**
- In this module admin can manage and view users Login in the web application within the Django admin module.
- **Manage Users Authentication and Authorisation**
  In this module admin can manage authentication and authorisation using their credentials like username and email id.
- **Managing Api keys and User logs**
  Each user will have their own api keys and user logs , this will be managed by the Admin itself.

### 2)User

- **User Registration**
  User registration is the first process of registration.The user should register themselves and can use the benefits of the application.
- **User Api key**
  The user can add their api keys to start accessing the twitter and visualize the insights from the data to derive quality information.

- **User Login**
  The user can login into the account which has been registered and enjoy the benefits and services which are free of cost.

### 2.6.1 List of Requirements

| **Admin** | | |
|---|---|---|
| **Sl.No** | **REQUIREMENTS** | **MoSCoW** |
| 1 | The system Must be web based. | MUST |
| 2 | There Must be a laptop or desktop to access the system | MUST |
| 3 | The system Must gain access by a standard web- browser. | MUST |
| 4 | The field validations must be provided for the Login Process. | MUST |
| 5 | System Admin must use the system | MUST |
| 6 | The System Must provide a login facility to the Admin. | MUST |
| 7 | System Admin Must use the system with login and password | MUST |
| 8 | System Must validate the username and password during the login | MUST |
| 9 | An error message Must be displayed if the username and password does not match | MUST |
| 10 | System Admin Must access the system with all privileges | MUST |
| 11 | System Admin Must manage users | MUST |
| 12 | System Admin Must add api keys | MUST |
| 16 | System Admin Must add district | MUST |
| 17 | System Admin Should view and delete district | SHOULD |
| 18 | System Admin Must add school | MUST |
| 19 | System Admin Should View and delete school Details | SHOULD |
| 20 | System Admin Should View all Registered user | SHOULD |
| 21 | System Admin Should Delete Registered user | SHOULD |

| | | |
|---|---|---|
| 22 | System Admin Must View users Feedback | MUST |
| 23 | System Admin Should Delete Customer Feedback | SHOULD |
| 24 | System Admin Must Logout | MUST |

# CHAPTER - 3
# ENVIRONMENT

## 3.1 System Specification

### 3.1.1 Software Specification

The Software Requirements is a technical specification of requirements for the software product. The goal of software requirements definition is to completely and consistently specify the technical requirements for the software product in a concise and unambiguous manner.

| | | |
|---|---|---|
| Operating System | : | Windows 7 and above |
| Web Server | : | WAMP Server |
| Web Browser | : | Any Standard web browser |
| Front End | : | Html,css,js |
| Back End | : | Django |
| Database | : | Sqlite |

### 3.1.2 Hardware Specification

The hardware requirement is the minimum hardware specification needed for the system to work.

| | | |
|---|---|---|
| Processor | : | Pentium IV or above |
| Ram | : | 4GB |
| Hard Disk | : | 20GB |

## 3.2 Software Technology

### 3.2.1 Python

Python has become one of the most dominant programming languages today. As per stats by major survey sites, Python has ranked among the top coding languages for the past few years now. There are tons of reasons for choosing Python as the primary language if compared with others such as Java, C++, or PHP. Since we're here to talk about web development, surprisingly even in web development, Python has reached its peak with tons of features, and improvements, and over the period it is becoming popular every day.

The most crucial point from a developer's point of view is to pick the right language that can deliver the desired results with ease, especially when we talk about web development. There are certain factors to consider that include management of database, security, data retrieval and so on. Now, how these factors fit in with Python is still clueless for most programmers because they've been using Java, PHP, etc. for web development but today, even big tech giants like Netflix, Google, and NASA have been actively using Python for web development. So, in this article, we will see why Python can be considered for web development and became famous among the top programming languages over these years.

5 Reasons to Choose Python for Web Development:

**1. Multi-Purpose Programming Language**

In contrast, Multi-purpose can simply be referred to as a multi-functionality having capabilities to operate in multiple ways. Surprisingly a developer can do wonders and can easily develop software by implementing easy methods. Being an interpreted language (smoothness in dev. process), this platform is absolutely free and open for all, it also offers platform independency, meaning their (Python's) code can run on any platform without making any changes (such as Linux, macOS, etc.)

**Besides this, Python can also be used in various ways:**

For Web Applications: Python is a part of the web development field and offers many frameworks to work with. Some of the most popular tools are Django, Flask, etc.

Desktop Applications: It is being used to create fascinating desktop applications by many companies these days and some of the widely used tools are Tkinter, PyGUI, Kivy, etc.

Cybersecurity: For malware analysis, developers are actively using highly secured tools to prevent any cyber attacks, tools like NumPy, Pandas, etc. are considered the perfect choices for it.

Scientific Calculation & Computing: The simplicity of Python allows developers to write more reliable systems and working on complex algorithms is much easier with this.

This makes Python one of the most demanding languages for all tiers of businesses (small to large) and that is what makes it one of the perfect choices for web development.

**2. Database Connectivity**

Establishing database connectivity is pretty simple with Python and accessing (including implementation) can easily be done on major databases such as Oracle, MySQL, PostgreSQL, etc and they can be called up by their respective APIs when required. Mention a demographic image below for best reference on how connectivity is being established.

**3. Simplifies -> Debugging – Deployment – Prototyping**

As we've discussed the multi-tasking capability of the Python programming language, application testing is one of the major advantages that developers get and eventually saves time and money. It enables ease in debugging, deployment, and building prototypes.

The reason is that it offers ease in terms of syntax, and readability and is perfect for learning automation. It also offers an easy-to-employ UT framework by which you can even perform geolocation testing (for mobile devices).

**4. Bunch of Frameworks**

There's a list of some influential frameworks that help in building sites and can easily fit into your project. And that too by offering enhanced security for either simple or complex websites, Python frameworks won't disappoint you for sure.

Frameworks? A tool built on top of any programming language to offer enhanced features when trying to build something useful. In Python, there are certain useful frameworks that help speed up the development process and enable developers to build advanced features. They (frameworks) come with bundled codes and modules so that they can be implemented multiple times.

**5. Growing Community Base**

As per the recent survey, over 10 million developers are helping in making it a strong base and actively offering their inputs whenever someone gets stuck. The number is high and is primarily for rectifying the errors that exist during designing the language. On average, hundreds of queries are being posted, viewed, and replied to by active developers throughout the world. Their community also helps in solving complex problems that you might not find anywhere on the internet and whenever you're working on any project in any organization, there might be circumstances that can put you in trouble, so next time, don't forget to visit its official forum for finding out the answer. If you're at the beginner level, we suggest you go for the Python Programming Foundation -Self Paced course and get certified in a structured way in no time.

### 3.2.2 Django

Python Django is a web framework that allows users to quickly create efficient web pages. Django is also called batteries included framework because it provides built-in features such as Django Admin Interface, default database – SQLite3, etc. When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use.

**Why Django Framework?**

Excellent documentation and high scalability.

Used by Top MNCs and Companies, such as Instagram, Disqus, Spotify, Youtube, Bitbucket, Dropbox, etc. and the list is never-ending.

Easiest Framework to learn, rapid development, and Batteries fully included. Django is a rapid web development framework that can be used to develop fully fleshed web applications in a short period of time.

The last but not least reason to learn Django is Python, Python has a huge library and features such as Web Scraping, Machine Learning, Image Processing, Scientific Computing, etc. One can integrate all this with web applications and do lots and lots of advanced stuff.

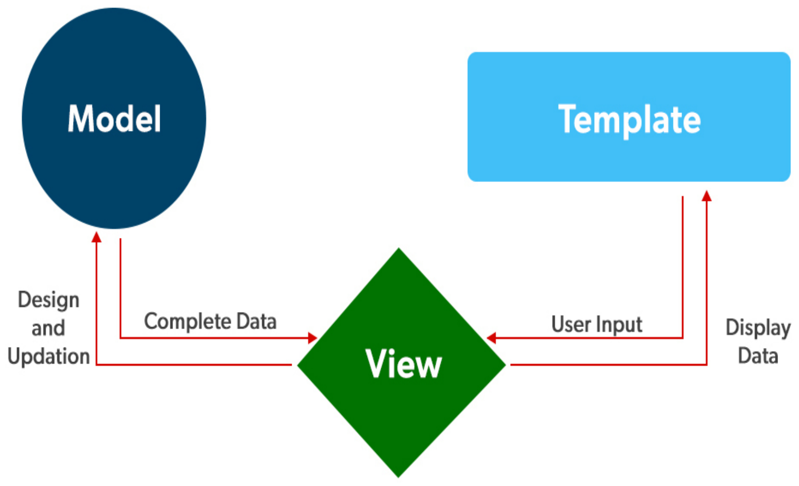
Django Architecture

Django is based on MVT (Model-View-Template) architecture which has the following three parts –

**Model:** The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres).

**View:** The View is the user interface that you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files.

**Template:** A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted. To check more, visit – Django Templates



**Installing Django**

We can install Django using the pip command. To install this type the below command in the terminal.>> **pip install django**

Starting the project : To initiate a project of Django on Your PC, open Terminal and Enter the following command django-admin startproject projectName : A New Folder with the name projectName will be created. To enter in the project using the terminal enter command cd projectName Now let's run the server and see if everything is working fine or not. To run the server type the below command in the terminal. python manage.py runserver After running the server go to **http://127.0.0.1:8000/** and you'll see something like this – Django runserver For more information, refer to How to Create a Basic Project using MVT in Django ? Project Structure A Django Project when initialized contains basic files by default such as manage.py, view.py, etc. A simple project structure is enough to create a single-page application. Here are the major files and their explanations. Inside the geeks_site folder ( project folder ) there will be the following files- Django project structure
Let's discuss these files in detail –

manage.py: This file is used to interact with your project via the command line(start the server, sync the database… etc). For getting the full list of commands that can be executed by manage.py type this code in the command window- **python manage.py help _init_.py**: It is a python package. It is invoked when the package or a module in the package is imported. We usually use this to execute package initialization code, for example for the initialization of package-level data. **settings.py:** As the name indicates it contains all the website settings. In this file, we register any applications we create, the location of our static files, database configuration details, etc. **urls.py:** In this file, we store all links of the project and functions to call. **wsgi.py:** This file is used in deploying the project in WSGI. It is used to help your Django application communicate with the webserver.

### 3.2.2.1 Advantages of Django

Here are a few reasons why Django is the first choice when it comes to web development.

**1. Backward compatible**

Django offers the provision of working with its older versions and makes use of its older formats and features. Additionally, it has a comprehensible guide that walks you through all the information that you might need to know about the changes and trends of the Django framework, especially when the new changes become incompatible with the former ones.

**2. Community Support**

Django enjoys the support of a huge and very professional community of developers who have inside out knowledge of Django and are always ready to help. Having a large community has benefits of its own. It makes finding answers to problems much easier as there’s a great probability that the problem faced by you now has occurred to someone else too and now probably has an answer to it on one or the other forums. The community is very quick in answering the issues and fixing the bugs of fellow developers.

**3. DevOps Compatible**

DevOps is the blend of cultural philosophies, tools, and practices that increases an organization’s ability to deliver services and applications at high speed: evolving and upgrading products at a faster rate than organizations using conventional software development and infrastructure management processes. Incorporating DevOps in Django is great as it fixes issues faster with enhanced operational support. It uses continuous delivery methodology to increase the efficiency of the system.

**4. DRY and KISS compliant**

Django religiously follows the ‘KISS’ principle which is “Keep It Short and Simple”. In Django, it simply means that the code must be brief, easily understandable, and methods should not exceed more than 50-60 lines. Similarly, ’DRY’ stands for “Don’t Repeat Yourself”, which means that the software patterns that occur quite often can be replaced with abstractions. In this manner, issues related to the code as well as repetition can be avoided. Additionally, reusing the code simplifies the development process thereby decreasing the overall production time.

**5. Infrastructure**

Django is independent and a complete set in itself. It means that it does not require any other external solution. It is everything, from an ORM to a web server. This enables it to use various databases and switch them accordingly.

**6. REST framework**

REST stands for Representational State Transfer framework which is a renowned toolkit for creating web APIs. This is an added benefit with Django as it is powerful enough to build a full-fledged API in just two or three lines of code. An additional benefit to it is that REST is immensely flexible. Therefore, data is not bound to any protocol and can return various data formats and manage several types of calls.

**7. Secure & up-to-date**

Django is constantly kept up to an elevated standard, following the most recent patterns in site development and security. That certainly addresses the inquiry "Is Django useful for web development?" — As security is a 'number one' in any to-do list. Django is customarily updated with security fixes, and regardless of whether you're utilizing an older rendition of the system, its security is as intact as the new one. It's no big surprise as Django has an LTS (Long-term Support) variant.

**8. Simple**

Django's documentation is excellent. It was introduced with excellent documentation, and they are as yet kept intact in the same way, which makes it simple to utilize. Also, one of Django's principal reasons for existing is to disentangle the development procedure: it covers the fundamentals, so you can concentrate on the details of your project.

**9. Suits any kind of project**

Unlike C# for Java, Django is no business solution. Still, it is appropriate for all kinds of projects regardless of their size. A Django application can be a social media application with enormous traffic and heavy volumes of data or it could be a simple web application for handling logbooks. It can build anything as it has everything required to build every type of application. Additionally, Django is cross-platform and also compatible with the majority of databases making it highly versatile.

**10. Time effective**

Django is ridiculously fast. It was built to slide the applications from imagination to reality in a blink. Django applications are both economical and efficient. Thus it is the right choice for developers who have major stakes placed on the due dates.

**11. Time tested**

Django has been in the picture for years and in that tenure, it has emerged as the choice of a lot of businesses for their web applications. This confirms the fact that Django has already made an impression in the market and is here to stay for long. A few examples of applications built on Django are Disqus, Spotify, Instagram, and NASA.

**12. Useful additional features**

Django comes with everything inbuilt. It has all the features which are required to build a web application from scratch. There’s no need to spend hours on forming a frame since everything is already present in the framework. It also has tools packages to help in going hands-on with pioneering technologies such as Artificial Intelligence, Data Analysis, and Machine Learning.

**13. Versatile**

Django is the master of all trades. It is capable of doing everything, from content management to managing scientific computing platforms; everything falls under the umbrella of Django.

**14. Works on Python**

As mentioned earlier, Django is a web framework which is written in Python. As a result, it brings with itself the simplicity of utilizing the syntax structure of Python and enables developers to build meaningful and viable web applications effortlessly. In this way, developers can easily reduce the development time required for building these web applications.

## 3.2.2.2 Syntax and Semantics

**Templates**

A template is a text file. It can generate any text-based format (HTML, XML, CSV, etc.).

A template contains variables, which get replaced with values when the template is evaluated, and tags, which control the logic of the template.

**Below is a minimal template** that illustrates a few basics. Each element will be explained later in this document.

```
{% extends "base_generic.html" %}
{% block title %}{{ section.title }}{% endblock %}
{% block content %}
<h1>{{ section.title }}</h1>
{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

**Variables**

Variables look like this: {{ **variable** }}. When the template engine encounters a variable, it evaluates that variable and replaces it with the result. Variable names consist of any combination of alphanumeric characters and the underscore ("_") but may not start with an underscore, and may not be a number. The dot (".") also appears in variable sections, although that has a special meaning, as indicated below. Importantly, you cannot have spaces or punctuation characters in variable names.

## 3.2.3 Sqlite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

**Why SQLite?**

- SQLite does not require a separate server process or system to operate (serverless).
- SQLite comes with zero-configuration, which means no setup or administration needed.
- A complete SQLite database is stored in a single cross-platform disk file.
- SQLite is very small and lightweight, less than 400KiB fully configured or less than 250KiB with optional features omitted.

- SQLite is self-contained, which means no external dependencies.
- SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.
- SQLite supports most of the query language features found in SQL92 (SQL2) standard.
- SQLite is written in ANSI-C and provides a simple and easy-to-use API.
- SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

### 3.2.3.1 Features of SQLITE

- MTransactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- Full-featured SQL implementation with advanced capabilities like partial indexes, indexes on expressions, JSON, common table expressions, and window functions. (Omitted features)
- A complete database is stored in a single cross-platform disk file. Great for use as an application file format.
- Supports terabyte-sized databases and gigabyte-sized strings and blobs. (See limits.html.)
- Small code footprint: less than 750KiB fully configured or much less with optional features omitted.
- Simple, easy to use API.
- Fast: In some cases, SQLite is faster than direct filesystem I/O
- Written in ANSI-C. TCL bindings included. Bindings for dozens of other languages available separately.
- Well-commented source code with 100% branch test coverage.
- Available as a single ANSI-C source-code file that is easy to compile and hence is easy to add into a larger project.
- Self-contained: no external dependencies.
- Cross-platform: Android, *BSD, iOS, Linux, Mac, Solaris, VxWorks, and Windows (Win32, WinCE, WinRT) are supported out of the box. Easy to port to other systems.
- Sources are in the public domain. Use for any purpose.
- Comes with a standalone command-line interface (CLI) client that can be used to administer SQLite databases.

# CHAPTER– 4

# SYSTEM DESIGN

System design refers to the description of a new system based on the information that is collected during the analysis phase and the process by which it is developed. It is the creative process of inventing and developing new inputs, database procedures and outputs to meet the system objectives. System design builds on the information gathered during system analysis. The system analyst must have a clear-cut understanding about the objectives, which the design aims to fulfill.System Design involves translating system requirements and conceptual design into technical specifications and general flow of processing. After the system requirements have been identified, information has been gathered to verify the problem and after evaluating the existing system, a new system is proposed.System Design is the process of planning a new system or to replace or complement an existing system. It must thoroughly understand the old system and determine how computers can be used to make its operations more effective.

System design sits at the core of system development. Once system requirements have been analyzed and specified, system design is the first of the technical activities-design, code generation and test- that require building and verifying the software. System design is the most creative and challenging phase of the system life cycle. The term design describes the final system and the process by which it is to be developed.System design is the high-level strategy for solving the problem and building a solution. System design includes decisions about the organization of the system into subsystems, the allocation of subsystems to hardware and software components and major conceptual and policy decisions that form the framework for detailed design.

## 4.1 Input Design

Input design is the method by which valid data is accepted from the user. This part of the design requires very careful attention. If the data going into the system is incorrect then the processing and output will magnify these errors. Inaccurate input data is the most common cause of errors in data processing. Input design consists of the following processes: -

- Designing a graphical user entry screen is easy to use.
- Designing procedures and functions to validate the data as per business rules.
- Designing functions needed to store data into a usable form for processing.
- Designing the common integrated functions that can be used by all other users when needed.

### 4.1.1 Input Objectives

Controlling Amount of Input: Wherever user input is required, giving possible input values as default in that area reduces the amount of user keystrokes. Thus, the user can pass onto the next data without much typing. This makes the data entry much faster and error free. When the user has the format of input to be given, it will be very easy for the user to give input in the same format.

Avoiding Delay: A processing delay resulting from data entry operations is called a bottleneck. Such bottlenecks are made obsolete in this project by breaking up the amount of data to be entered in each form into different smaller and simpler forms. Avoiding Errors in Data: The rate at which errors occur depends on the quantity of the data. As told in the above objective these errors are reduced by making the number of data to be entered in each form is reduced.Avoiding Extra Steps: To fulfill any operation the user has no need to do complex steps, instead any operation can be done with simple easy to use steps.

## 4.2 Output Design

Output design is one of the most important features of the information system. When the output is not of good quality, the users will be averse to using the newly designed system and may not use the system. There are many types of outputs, all of which can be either highly useful or can be critical to the users, depending on the manner and degree to which they are used.

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent hard copy of the results for later consultation.

### 4.2.1 Output Objectives

The output from an information system should accomplish one or more of the following objectives:

- Convey information about past activities, current status, or projections of the future.
- Signal important events, opportunities, problems or warnings
- Trigger an action
- Confirm an action

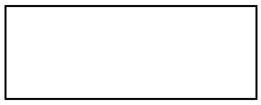
## 4.3 Data Flow Diagram

Data Flow Diagram (DFD) representing a system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources. The purpose of DFD is to provide a semantic bridge between users and system developers. The diagram is the basis of structured system analysis. A level 0 DFD, also called a fundamental system model or a context model, represents the entire software elements as a single bubble with input and output indicated by incoming and outgoing arrows respectively. Additional process and information flow parts are represented in the next level i.e., Level 1 DFD. Each of the processes represented at Level 1 are sub functions of the overall system depicted in the context model. Any processes, which are complex in Level 1, will be further represented into subfunctions in the next level, i.e., in level 2. Data flow diagrams illustrate how data is processed by a system in terms of inputs, and outputs. Represent major components or functions with Circles. Actions for input by a user or a system go in Rectangular Boxes. Databases are represented by Parallel lines enclosing a phrase corner.

## 4.3.1 Data Flow Diagram – The Rules

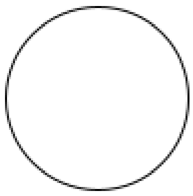
**1. External Entities**

External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system ‘s inputs and outputs.



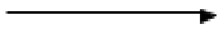
**2. Processes**

When naming processes, avoid glossing over them, without really understanding their role. Indications that this has been done are the use of vague terms in the descriptive title area like _process ‘or _update ‘. The most important thing is that the description must be meaningful to whoever will be using the diagram.



**3. Data Flows**

Double-headed arrows can be used (to show two-way flows) on all but bottom level diagrams. Furthermore, in common with most of the other symbols used, a data flow at a particular level of a diagram may be decomposed to multiple data flows at lower levels.



**4. Data Store**

Data stores represent stores of data within the systems and are represented by open rectangles. Data Flows represent the movements of data between other components and are shown by arrows.

### 4.3.2 Context Level DFD

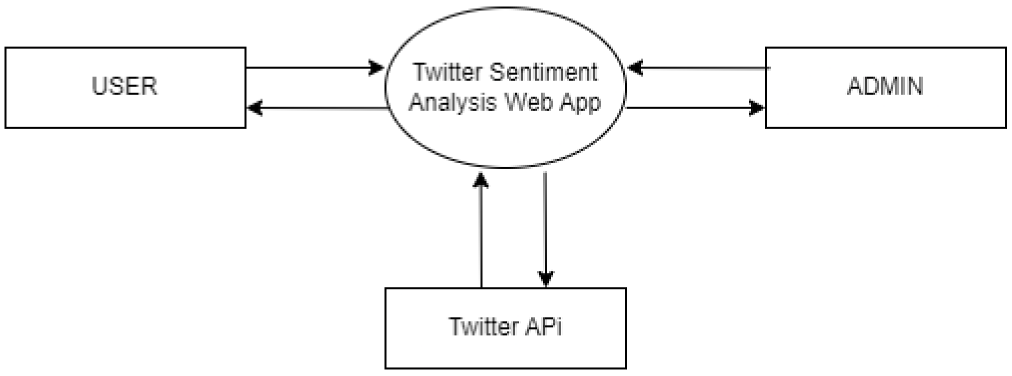


*Figure 4.1: Data Flow Diagram Level  0*

### 4.3.3 Level 1 DFD

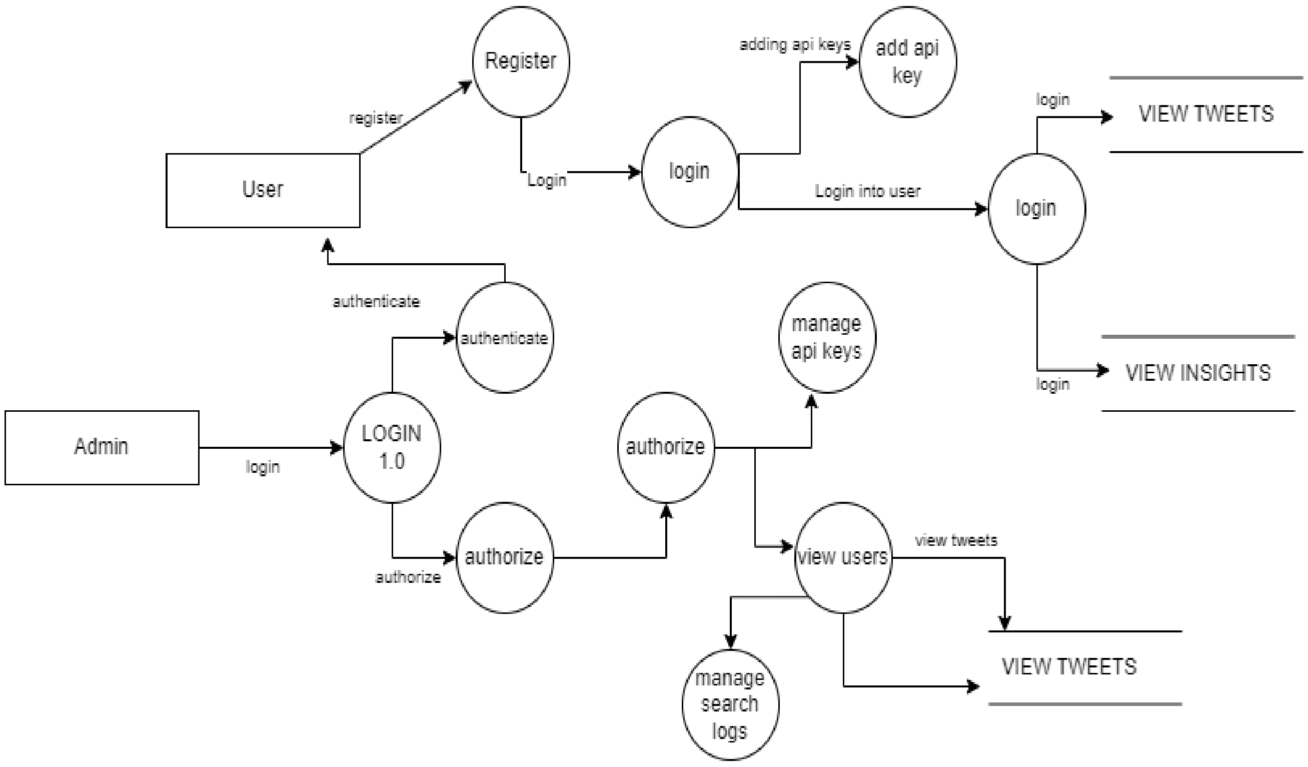


*Figure 4.2: Data Flow Diagram Level  1*

### 4.3.4 Level 2 DFD

- **Admin 1.1**

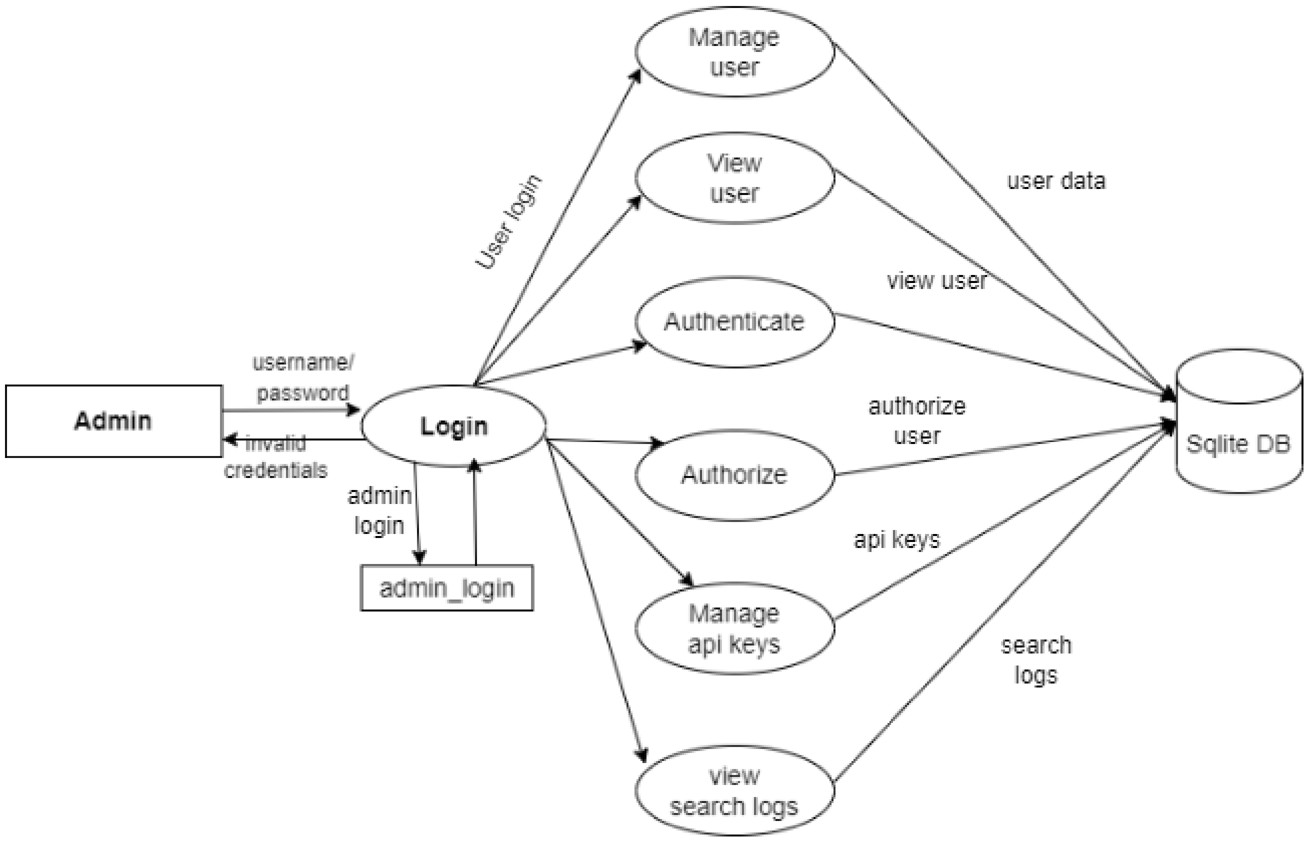


*Figure 4.3: Data Flow Diagram for Admin Level 1*

- **User**

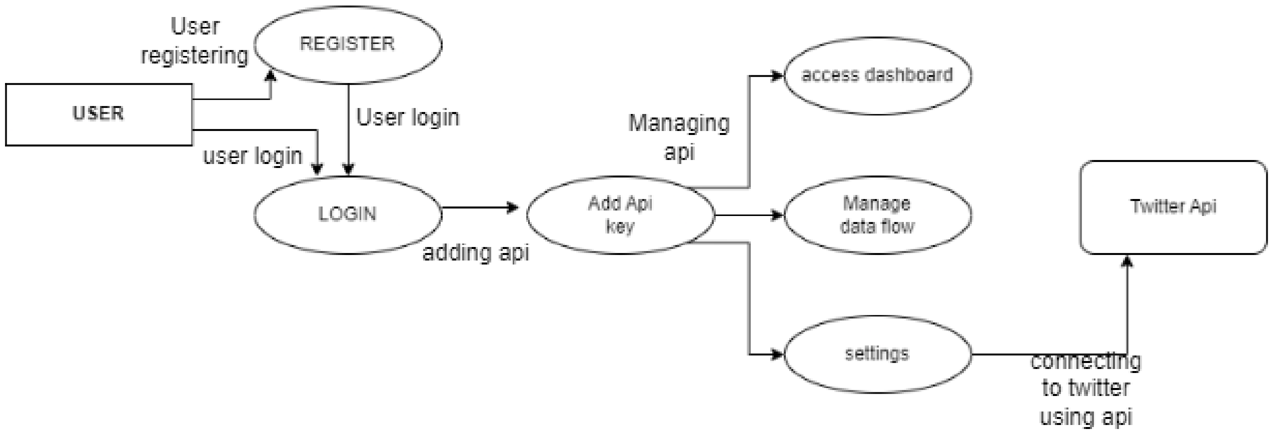


*Figure 4.4: Data Flow Diagram for school Level 1*

## 4.4 Analysis Tools

System analysis is the process of collecting and interpreting facts, understanding problems and using the information to suggest improvements on the system. This will help to understand the existing system and determine how computers make its operation more effective.

### 4.4.1 Use Case Diagram

A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. The main purpose of a use case diagram is to show what system functions are performed for which actor.

Elements of use case diagram are:

**Use cases :**

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actors :**

An actor is a person, organization, or external system that plays a role in more interactions with the system.

The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. Now when the initial task is complete use case diagrams are modeled to present the outside view.

An important part of the Unified Modelling Language (UML) is the facilities for drawing use case diagrams. Use cases are used during the analysis phase of a project to identify and partition system functionality. They separate the system into actors and use cases. Actors represent roles that can be played by users of the system. Those users can be humans, other computers, pieces of hardware, or even other software systems. The only criterion is that they must be external to the part of the system being partitioned into use cases. They must supply stimuli to that part of the system, and they must receive outputs from it.

## ● Overall use case diagram

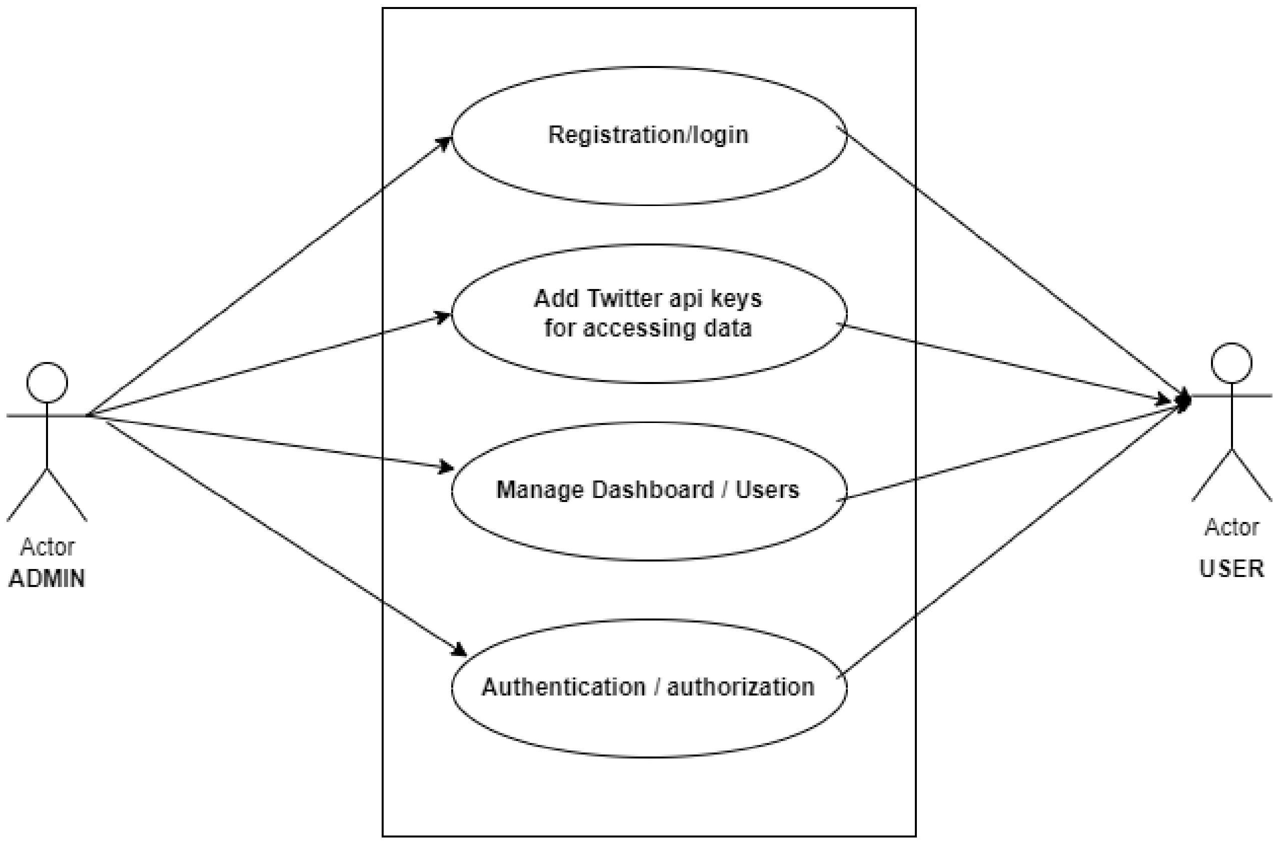

Registration/login
Add Twitter api keys for accessing data
Manage Dashboard / Users
Authentication / authorization
Actor
ADMIN
Actor
USER

*Figure 7: Usecase Diagram for Twitter sentiment analysis web app.*

**Description:**

In the figure, there are Two users in the system. The Admin, User registers with the system providing their details. The admin after login may add User and approve them. The admin login to the system add User into the system .admin manages api keys also

- **Admin**

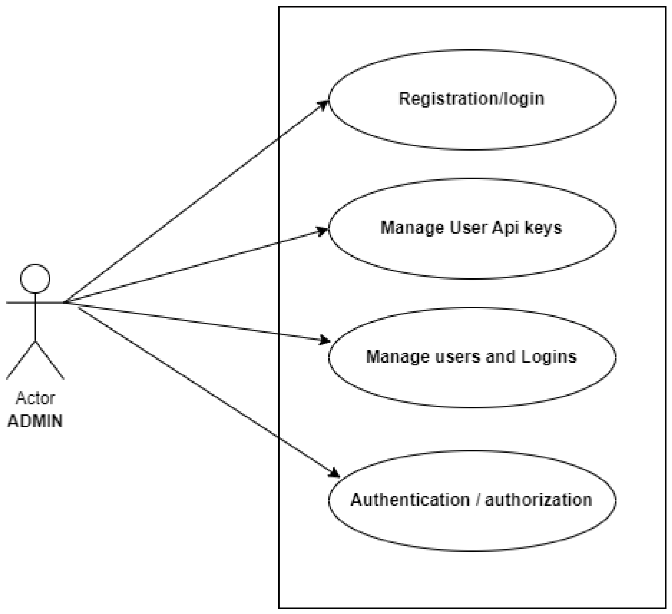


*Figure 8: Use case Diagram for Admin*

**Description:**

The figure represents the use case diagram for admin. The admin logins to the system and manages users, game. He can view student performance.

- **User**

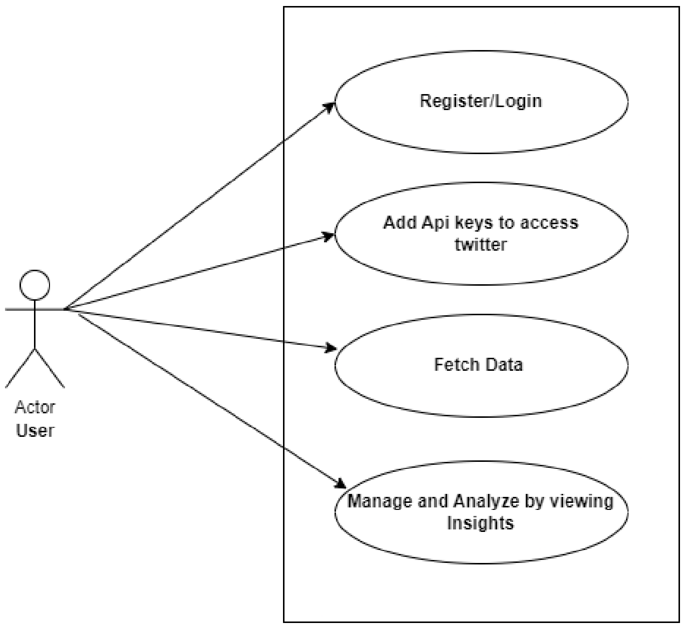

Register/Login
Add Api keys to access twitter
Fetch Data
Manage and Analyze by viewing Insights
Actor
User

*Figure 9: Use Case Diagram for User*

**Description:**

The figure represents the use case diagram for User. The Userlogin to the system and manages data flow, teachers.He can view and analyze data from api

## 4.5 Table Design

To design an application, it is necessary to design the database. A database is a collection of interrelated data with minimum redundancy to serve the user quickly and efficiently. Database Management System is the process of organizing the tables in the database. The most important step in system design is the database design. The database management system allows the data to be protected and organized separately from other resources. Efficient database management system is software which provides service for accessing, storing and manipulating databases.

The objectives of database are:

- Controlled redundancy
- Data independence
- Accuracy and integrity
- Privacy and security
- Performance

The entities along with their attributes can be stored in many different ways into a set of tables. The methods of arranging these attributes are called normal forms.

**First Normal Form**

It suggests that each attribute value should be atomic. There will be lists or sets. There is no top-to-bottom ordering to the rows. There is no left-to-right ordering to the columns. There are no duplicate rows.

**Second Normal Form**

It suggests that the table should be in first normal form and every non-attribute must be related to the primary key. In slightly more formal terms: a 1 NF table is in 2 NF if and only if all its non-prime attributes are functionally dependent on the whole of every candidate key.
(A non –prime attribute is one that does not belong to any candidate key.)

**Third Normal Form**

It suggests that the table should be in second normal form and there will be no transitive dependency.

**4.5.1** Table Name: accounts_profile
Primary Key: id
Purpose: To store user profiles

| Column Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INTEGER | Primary Key | Unique id for account profiles |
| bio | TEXT | Not Null | Bio of users |
| location | TEXT | Not Null | Locations of users |
| birth_date | date | Not Null | Date of Birth of Users |
| email_confirmed | bool | Not Null | Confirmed Email id of users |

*Table 1:Database table for Account profile details*

**4.5.2** Table Name : auth_permission
Primary Key: id
Purpose: To store authentication permissions

| Column Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INTEGER | Primary Key | Unique id of Authentication. |
| Content_type_id | INTEGER | Not Null | Content type id of authentication permission |
| codename | VARCHAR(20) | Not Null | code name of authentication permission |
| name | VARCHAR(225) | not NULL | Name of Auth permission |

*Table 2: Database table for authentication permission details*

**4.5.3** Table Name : auth_user
Primary Key: id
Purpose : To store authenticated user

| Column Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INTEGER | Primary Key | Unique id of auth user. |

| | | | |
|---|---|---|---|
| password | Varchar(20) | Not Null | Password of Auth user. |
| last_login | datetime | Not Null | Last login of auth user |
| is_superuser | bool | Not Null | Is super user |
| username | Varchar(100) | Not Null | username of Auth user |
| first_name | Varchar(30) | Not Null | first Name of auth user |
| email | varchar(50) | Foreign Key | Email of auth user |
| is_active | bool | Not Null | Is User active |
| is_staff | bool | Not Null | is user is staff |
| joined_date | datetime | Not Null | Joined date of user |

*Table 3: Database table for authenticated user details*

**4.5.4** Table Name :django_admin_log
Primary Key: id
Purpose: To store admin log

| **Column Name** | **Data Type** | **Constraint** | **Description** |
|---|---|---|---|
| id | Integer | Primary Key | Unique django admin log |
| action_time | datetime | Not Null | action time of admin login |
| object_id | text | Not Null | Object id of admin |
| object_repr | text | Not Null | Object representation of admin |
| change_message | Varchar(150) | Not Null | Change message of admin |
| content_type_id | integer | Not Null | content type id of admin |
| user_id | integer | Not Null | User id of admin log |
| action_flag | smallint Unsigned | Not Null | action flag django flag |

*Table 4: Database table for Django admin panel*

**4.5.5** Table Name : django_content_type
Primary Key: id
Purpose : To store admin

| **Column Name** | **Data Type** | **Constraint** | **Description** |
|---|---|---|---|
| id | Integer | Primary Key | Unique id of Content type of user |
| app_label | Varchar(40) | Not Null | App label |
| model | Varchar(50) | Not Null | Model of content type |

*Table 5: Database table for content type details*

**4.5.6** Table Name :django_sessions
Primary Key: session_key
Purpose : To store user session in the application

| **Column Name** | **Data Type** | **Constraint** | **Description** |
|---|---|---|---|
| session_key | Integer | Primary Key | Session key of sessions |
| session_data | TEXT | Not Null | session data from django sessions |
| expire_date | datetime | Not Null | expire date of each sessions |

*Table 6: Database table for user session handling*

**4.5.7** Table Name :twitter_apikeys
Primary Key: id
Purpose : To store api key in the application

| **Column Name** | **Data Type** | **Constraint** | **Description** |
|---|---|---|---|
| id | Integer | Primary Key | Unique id of users |
| api_key | varchar(100) | Not Null | Api keys for connecting with web app |
| api_secret_key | VARCHAR(100) | Not Null | Api secret key |
| access_token | varchar(100) | Not Null | Api access token |
| access_token_secret | varchar(100) | varchar(100) | Api access token secret key |

*Table 6: Database table for twitter api keys handling*

**4.5.6** Table Name :twitter_searchlogs
Primary Key: id
Purpose : To store search log of the application

| **Column Name** | **Data Type** | **Constraint** | **Description** |
|---|---|---|---|
| id | Integer | Primary Key | Unique key of user |
| term | varchar(100) | Not Null | Terms used to search using api |
| total_tweet | Integer | Not Null | total tweets fetched from twitter via api |

*Table 6: Database table for search logs storing*

## 4.6 Algorithm

This project uses one algorithm to predict students' performance based on students previous exams and games scores, the algorithm is KNN. The KNN stands for the KNearestNeighbors Algorithm.

**KNN – KNearestNeighbors Algorithm**

The KNN algorithm is a robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM). Despite its simplicity, KNN can outperform more powerful classifiers and is used in a variety of applications such as economic forecasting, data compression and genetics. KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

**What is KNN?**

Let's first start by establishing some definitions and notations. We will use x to denote a *feature* (predictor, attribute) and yy to denote the *target* (label, class) we are trying to predict.

KNN falls in the **supervised learning** family of algorithms. Informally, this means that we are given a labeled dataset consisting of training observations (x, y) (x, y) and would like to capture the relationship between xx and yy. More formally, our goal is to learn a function h:X→Yh:X→Y so that given an unseen observation xx, h(x)h(x) can confidently predict the corresponding output yy.

The KNN classifier is also a **non parametric** and **instance-based** learning algorithm.

- **Non-parametric** means it makes no explicit assumptions about the functional form of h, avoiding the dangers of mismodeling the underlying distribution of the data. For example, suppose our data is highly non-Gaussian but the learning model we choose assumes a Gaussian form. In that case, our algorithm would make extremely poor predictions.
- **Instance-based** learning means that our algorithm doesn't explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as "knowledge" for the prediction phase. Concretely, this means that only when a query to our database is made (i.e. when we ask it to predict a label given an input), will the algorithm use the training instances to spit out an answer.

KNN is non-parametric, instance-based and used in a supervised learning setting.
It is worth noting that the minimal training phase of KNN comes both at a *memory cost*, since we must store a potentially huge data set, as well as a *computational cost* during test time since classifying a given observation requires a run-down of the whole data set. Practically speaking, this is undesirable since we usually want fast responses.

**How does KNN work?**

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance but other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance. More formally, given a positive integer K, an unseen observation xx and a similarity metric dd, KNN classifier performs the following two steps:

- It runs through the whole dataset computing dd between xx and each training observation. We'll call the K points in the training data that are closest to xx the set AA. Note that K is usually odd to prevent tie situations.
- It then estimates the conditional probability for each class, that is, the fraction of points in AA with that given class label. (Note I(x)I(x) is the indicator function which evaluates to 11 when the argument xx is true and 00 otherwise)

Finally, our input xx gets assigned to the class with the largest probability.
KNNsearches the memorized training observations for the K instances that most closely resemble the new instance and assigns to it their most common class.
An alternate way of understanding KNN is by thinking about it as calculating a decision boundary which is then used to classify new points.

### 4.6.1 Merits and Demerits of KNN

**Merits**

As you can already tell from the previous section, one of the most attractive features of the K Nearest neighbor algorithm is that it is simple to understand and easy to implement. With zero to little training time, it can be a useful tool for off-the-bat analysis of some data set you are planning to run more complex algorithms on. Furthermore, KNN works just as easily with multiclass data sets whereas other algorithms are hardcoded for the binary setting. Finally, as we mentioned earlier, the non-parametric nature of KNN gives it an edge in certain settings where the data may be highly "unusual".

**Demerits**

One of the obvious drawbacks of the KNN algorithm is the computationally expensive testing phase which is impractical in industry settings. Note the rigid dichotomy between KNN and the more sophisticated Neural Network which has a lengthy training phase albeit a **very fast** testing phase. Furthermore, KNN can suffer from skewed class distributions.

**Improvements**

With that being said, there are many ways in which the KNN algorithm can be improved.

- A simple and effective way to remedy skewed class distributions is by implementing **weighted voting**. The class of each of the K neighbors is multiplied by a weight proportional to the inverse of the distance from that point to the given test point. This ensures that nearer neighbors contribute more to the final vote than the more distant ones.
- **Changing the distance metric** for different applications may help improve the accuracy of the algorithm.
- **Rescaling your data** makes the distance metric more meaningful. For instance, given 2 features height and weight, an observation such as x=[180,70]x=[180,70] will clearly skew the distance metric in favor of height. One way of fixing this is by column-wise subtracting the mean and dividing by the standard deviation. Scikit Learn Normalize() method can come in handy.
- **Dimensionality reduction** techniques like PCA should be executed prior to applying KNN and help make the distance metric more meaningful.

**Approximate Nearest Neighbour** techniques such as using *k-d trees* to store the training observations can be leveraged to decrease testing time. Note however that these methods tend to perform poorly in high dimensions (20+). Try using **locality sensitive hashing (LHS)** for higher dimensions.

# CHAPTER 5

# SYSTEM TESTING

## 5.1 Introduction

In any software development, testing is a process to show the correctness of the program and it meets the design specifications. Testing is needed to prove correctness, to show completeness, to improve the quality of the software and to provide the maintenance aid. Some testing standards are therefore necessary to ensure completeness of testing, improve the quality of the software, and reduce the testing costs and to reduce study needs and operation time. Testing software extends throughout the coding phase and it represents the ultimate review of configurations, design and coding. A series of test cases are created that are intended to demolish the software that has been built.

Based on the way the software reacts to these tests, we can decide whether the configuration that has been built is study or not. It is essential that all components of an application be tested, as the failure to do so many results in a series of bugs after the software is put to use. Several methods of testing exist in software Engineering, which enable a programmer to make sure that the configuration built is free of bugs.

## 5.2 Testing Procedure

For any software that is newly developed, first and foremost preference is given to the testing of the system. It is the developer 's last chance to detect and correct the errors. That may occur possibly in the software. The programmers will generate a set of test data, which will give the maximum possibility of finding almost all types of errors that can occur in the system.

### 5.2.1 Unit Testing

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between the modules. Unit testing has proven its value in that a large percentage of defects are identified during its use. The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit. The investment of developer time in this activity sometimes results in demoting unit testing to a lower level of priority and that is almost always a mistake. Even though the drivers and stubs cost time and money, unit testing provides some undeniable advantages. It allows for automation of the testing process, reduces difficulties of discovering errors contained in more complex pieces of the application, and test coverage is often enhanced because attention is given to each unit.

### 5.2.2 Integration Testing

Integration testing (sometimes called Integration and Testing, abbreviated as ―I&T‖) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes place as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These ―design items‖ i .e. assemblages (or groups of units) are exercised through their interfaces using Black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a ―building block‖ approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

### 5.2.3 Validation Testing

Data validation is the process of testing the accuracy of data. A set of rules we can apply to a control to specify the type and range of data that can enter. It can be used to display error alerts when users enter incorrect values into a form. Now performing validation testing in system Centralized Social Welfare by undergoing validation for each tool and the validation succeeded when the software functioned in a manner that can be reasonably accepted by the user.

### 5.2.4 User Acceptance Testing

User acceptance of a system is a key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system user at the time of developing and making changes whenever required.

## 5.3 Testing strategies

### 5.3.1 Top down Testing

Top-Down Testing tests the higher levels of a system before testing its detailed components. The program is represented as a single abstract component with subcomponents represented by stubs. Stubs have the same interface as the components but are very limited functionally. After the top-level component has been tested, its subcomponents are implemented and tested in the same way. This process continues recursively until the bottom- level components are implemented. The whole system may then be completely tested.

### 5.3.2 Bottom-Up Testing

Bottom-Up Testing is the converse of Top-Down Testing. It involves testing the modules at the lower levels of the hierarchy and then working up the hierarchy of the modules until the final module is tested. The advantage of bottom-up testing is the disadvantage of top- down testing and vice versa. When using bottom-up testing test drivers must be written to exercise the lower level components. These test drivers simulate the components environment and are valuable components; the test drivers and test data should be distributed with the component. Potential re-users can then run these tests to satisfy themselves that the component behaves as expected in their environment.

### 5.3.3 Black Box testing

Knowing the specified function that a product has been designed to perform, a test can be conducted that demonstrates each function that is fully operational, at the same time searching for errors in each function. Black Box testing focuses on functional requirements of the software. Black Box testing attempts to find out errors in the following categories.

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access.
- Performance and errors.
- Initialization and termination errors.

**5.3.4 White Box Testing**

Knowing the internal working of a product test can be conducted to ensure that ―all gears mesh‖ that is internal operation performs according to specification and all internal components have been adequately exercised. Using white box testing methods, the software engineer can derive test cases that

- Guarantee that all independent paths within a module have been exercised at least once basis path testing.
- Exercise all logical decisions on their true and false sides- Condition testing.
- Execute all loops at their boundaries and within their operation bounds- Loop testing.
- Exercise internal data structures to assure their validity-data flow testing.

## 5.4 Test Cases

| **Test Case** | **Test Description** | **Test Procedure** | **Test Data** | **Expected Result** | **Act Ual Res Ult** |
|---|---|---|---|---|---|
| Login | To test whether Username and Password are valid | Enter the Username and Password | 1)akhil<br>2)admin<br>3)appu<br>4)krhss | 1)Invalid User<br>2)Valid User<br>3)Valid User<br>4)Valid User | 1)Invalid User<br>2)Val id User<br>3)Val id User<br>4)Val id User |
| Forget Password | To verify Username or Password exists | Enter the Username | 1)akhil<br>2)aju | 1)Invalid User<br>2)Valid User | 1)Invalid User<br>2)Val id User |
| Validation | To check whether all the validations are correct | Enter blank or Invalid data | 1)Blank data<br>2)incorrect data<br>3)correct and filled data | 1)Invalid Entry<br>2)Invalid Entry<br>3)Valid Entry | 1)Invalid Entry<br>2)Invalid |
| | | | | | Entry<br>3)Val id Entry |

| | | | | | |
|---|---|---|---|---|---|
| Change password | To check whether the password updated correctly | Enter old password ,new password and confirm password | 1) Incorrect old password. 2)Correct old password. 3)Incorrect confirm password. 4)Correct old password, new and confirm password | 1)Password mismatch 2)Validation pass 3)Confirm password mismatch 4)Password Changed successfully | 1)Password mismatch 2)Validation in pass 3)Confirm pass word mismatch 4)Password Changed successfull y |

| | | | | | |
|---|---|---|---|---|---|
| Admin add school | To check whether the school is added successfully | Enter the school details | registration details | Successfully registered. | Successfully registered. |
| Analyze tweet | to check the visualization of the tweet derived from api | Analyze tweets | View tweets score | Sore display | Sore display |
| Feedbac k | To check whether the feedbacks are added successfully | Enter feedback details. | First name, last name, email, feedback. | Feedback added successfully | Feedback added successfully |
| Logout | To check the users can successfully logout. | logout | Logout request | Home page | Home page |

# CHAPTER 6
# SYSTEM IMPLEMENTATION

## 6.1 Introduction

The final and important phase in the system life cycle is the implementation of the new system. The term implementation has different meanings ranging from the conversion of a basic application to a complete replacement of a computer system. The procedure however, is virtually the same. Implementation includes all those activities that take place to convert from the old system to new. The new system may be totally new replacing existing systems, manual or automated, or it may be a major modification to an existing system. The method of implementation and time scale to be adopted is found out initially. Next the system is tested properly and at the same time users are trained in the new procedure. Proper implementation is essential to provide a reliable system to meet organization requirements. Successful implementation may not guarantee improvement in the organization using the new system, but it will prevent improper installation. The implementation involves following things:

- Careful planning.
- Investigation of the system considerations.
- Design the method to achieve the changeover.
- Evaluation of change over method.

Implementation of a new system requires the operating staff installing the software and creating computer files. There are many ways in which this can be achieved. The most common methods are the following.

- Direct changeover
- Parallel running
- Pilot running change over the creation of the designed system takes place in the implementation phase.

This phase activities do the following:

- Development of phase overview
- Preparing for implementation
- Computer program development
- Development phase report and overview

It also performs activities like writing, testing, debugging and documenting the programs.

There are three types of implementations:

- Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, creating accurate files and verifying printouts for integrity.
- Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned, there can be many problems. Some large computer systems have taken as long as a year to convert.

- Implementation of a modified application to replace the existing one, using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes in the files. Every system requires periodic evaluation after implementation.

This is to review the performance of the system and to evaluate against established standards or criteria. A study is conducted for measuring the performance of the system against pre- defined requirements. This study results in a post-implementation review that determines how well the system continues to meet the performance specification.

# CHAPTER 7

# CONCLUSION

The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. So we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance.Right now we have worked with only the very simplest unigram models; we can improve those models by adding extra information like closeness of the word with a negation word. We could specify a window prior to the word (a window could for example be of 2 or 3 words) under consideration and the effect of negation may be incorporated into the model if it lies within that window.

The closer the negation word is to the unigram word whose prior polarity is to be calculated, the more it should affect the polarity. For example if the negation is right next to the word, it may simply reverse the polarity of that word and farther the negation is from the word the more minimized ifs effect should be. Apart from this, we are currently only focusing on unigrams and the effect of bigrams and trigrams may be explored. As reported in the literature review section when bigrams are used along with unigrams this usually enhances performance

# CHAPTER 8

# FUTURE ENHANCEMENT

However for bigrams and trigrams to be an effective feature we need a much more labeled data set than our meager 9,000 tweets. Right now we are exploring Parts of Speech separate from the unigram models, we can try to incorporate POS information within our unigram models in future. So say instead of calculating a single probability for each word like P(word | obj) we could instead have multiple probabilities for each according to the Part of Speech the word belongs to. For example we may have P(word | obj, verb), P(word | obj, noun) and P(word | obj, adjective). Pang et al. [5] used a somewhat similar approach and claims that appending POS information for every unigram results in no significant change in performance (with Naive Bayes performing slightly better and SVM having a slight decrease in performance), while there is a significant decrease in accuracy if only adjective unigrams are used as features.

However these results are for classification of reviews and may be verified for sentiment analysis on micro blogging websites like Twitter. One more feature that is worth exploring is whether the information about relative position of words in a tweet has any effect on the performance of the classifier. Although Pang et al. explored a similar feature and reported negative results, their results were based on reviews which are very different from tweets and they worked on an extremely simple model. One potential problem with our research is that the sizes of the three classes are not equal. The objective class which contains 4,543 tweets is about twice the sizes of positive and negative classes which contain 2,543 and 1,877 tweets respectively. The problem with unequal classes is that the classifier tries to increase the overall accuracy of the system by increasing the accuracy of the majority class, even if that comes at the cost of decrease in accuracy of the minority classes. That is the very reason why we report significantly higher accuracies for objective class as opposed to positive or negative classes. To overcome this problem and have the classifier exhibit no bias towards any of the classes, it is necessary to label more data (tweets) so that all three of our classes are almost equal.

In this research we are focussing on general sentiment analysis. There is potential for work in the field of sentiment analysis with partially known context. For example, we noticed that users generally use our website for specific types of keywords which can be divided into a couple of distinct classes, namely: politics/politicians, celebrities, products/brands, sports/sportsmen, media/movies/music. So we can attempt to perform separate sentiment analysis on tweets that only belong to one of these classes (i.e. the training data would not be general but specific to one of these categories) and compare the results we get if we apply general sentiment analysis on it instead.

Last but not the least, we can attempt to model human confidence in our system. We could develop our custom cost function for coming up with optimized class boundaries such that highest weightage is given to those tweets in which all 5 labels agree and as the number of agreements start decreasing, so do the weights assigned. In this way the effects of human confidence can be visualized in sentiment analysis.

# APPENDIX

## Appendix – A – Screen Shots

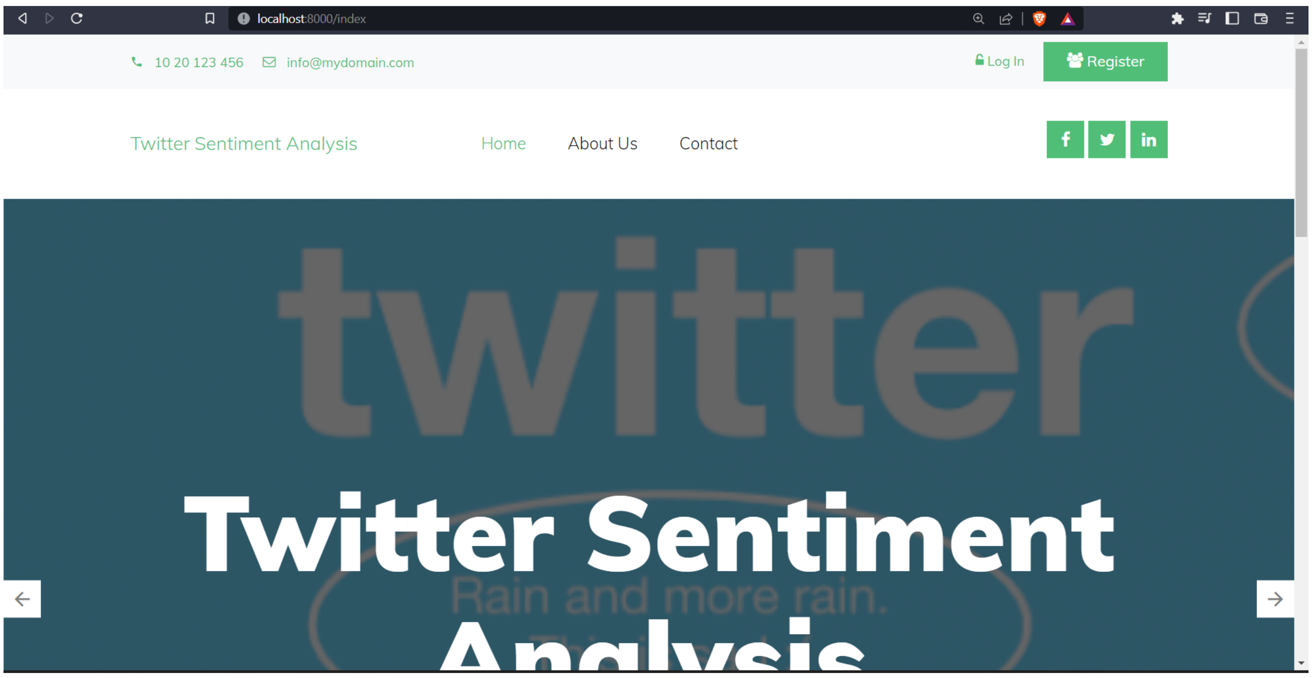
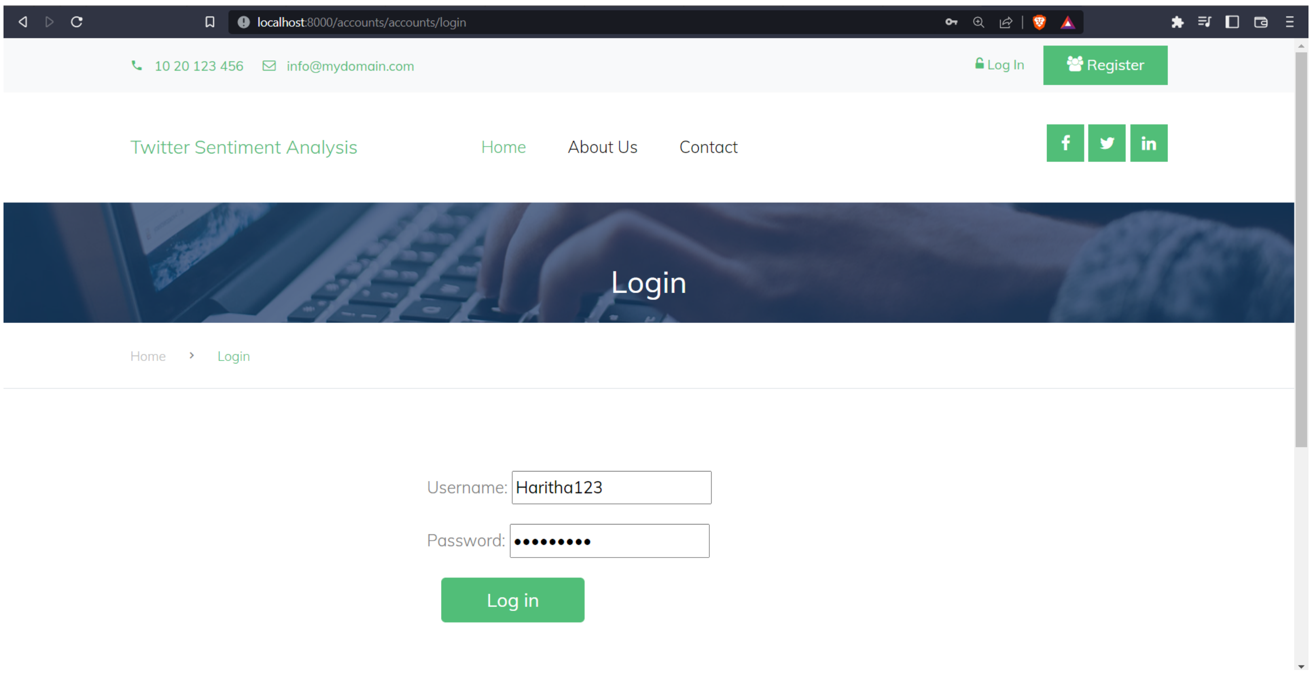


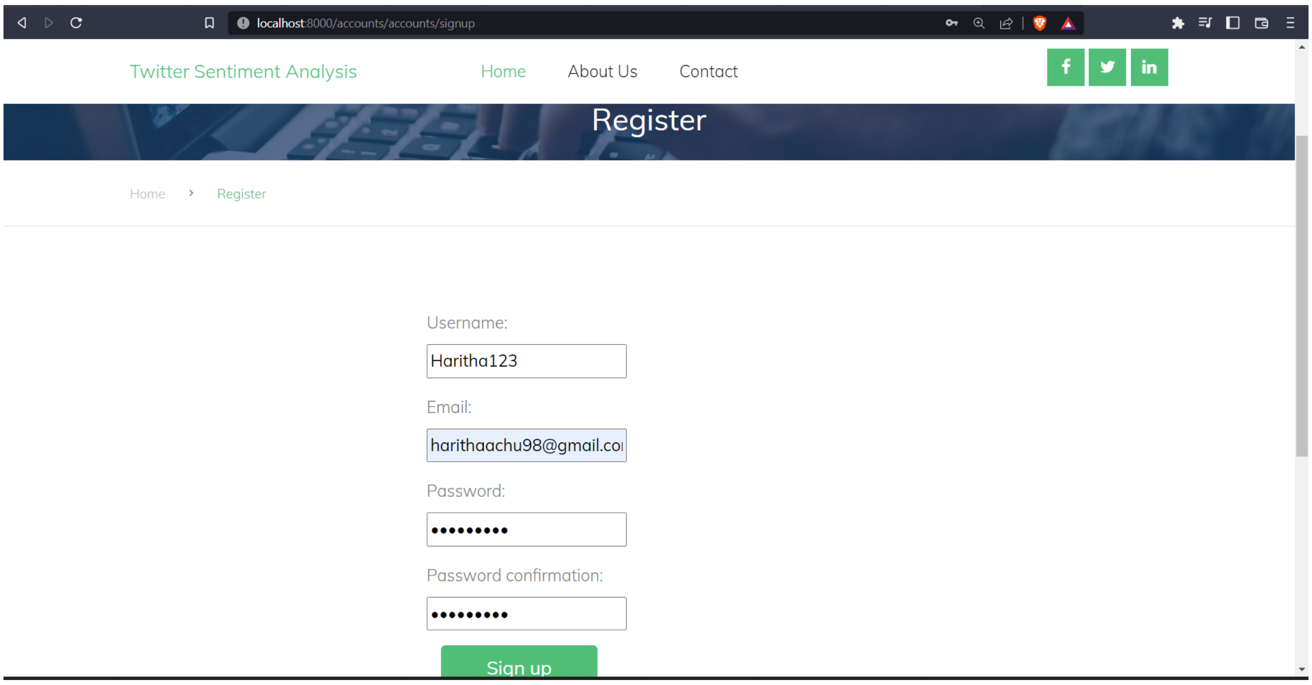
*figure : 9.1 :Home Page of Twitter Sentiment Analysis.*



Username: Haritha123

Password: •••••••••

*figure : 9.2 : Login Page of Twitter Sentiment Analysis App.*



localhost:8000/accounts/accounts/signup

Twitter Sentiment Analysis | Home | About Us | Contact

Register

Home › Register

Username:
Haritha123

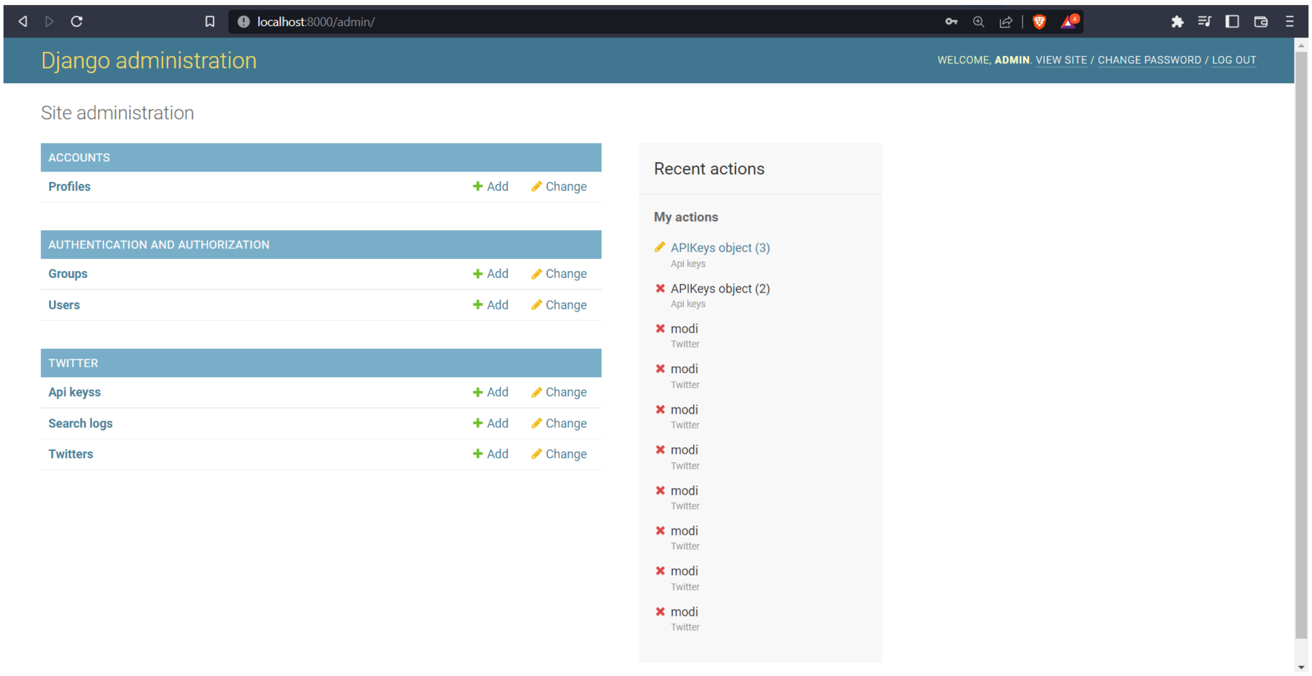
Email:
harithaachu98@gmail.co

Password:
•••••••••

Password confirmation:
•••••••••

Sign up

*figure : 9.3 :Registration Page of Twitter Sentiment Analysis app.*



localhost:8000/admin/

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

| ACCOUNTS | | |
|---|---|---|
| Profiles | + Add | Change |

| AUTHENTICATION AND AUTHORIZATION | | |
|---|---|---|
| Groups | + Add | Change |
| Users | + Add | Change |

| TWITTER | | |
|---|---|---|
| Api keyss | + Add | Change |
| Search logs | + Add | Change |
| Twitters | + Add | Change |

Recent actions

My actions

- APIKeys object (3) — Api keys
- APIKeys object (2) — Api keys
- modi — Twitter
- modi — Twitter
- modi — Twitter
- modi — Twitter
- modi — Twitter
- modi — Twitter
- modi — Twitter
- modi — Twitter

*figure : 9.4 :Admin Panel of Twitter Sentiment Analysis App.*

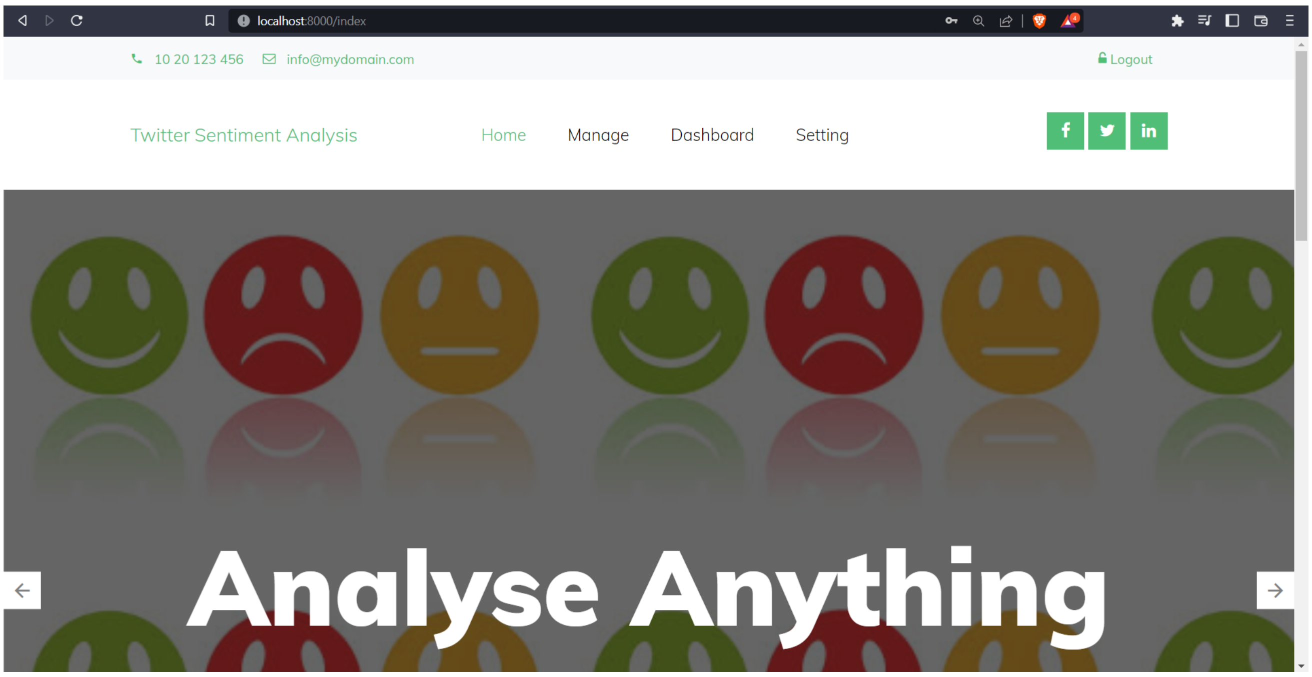
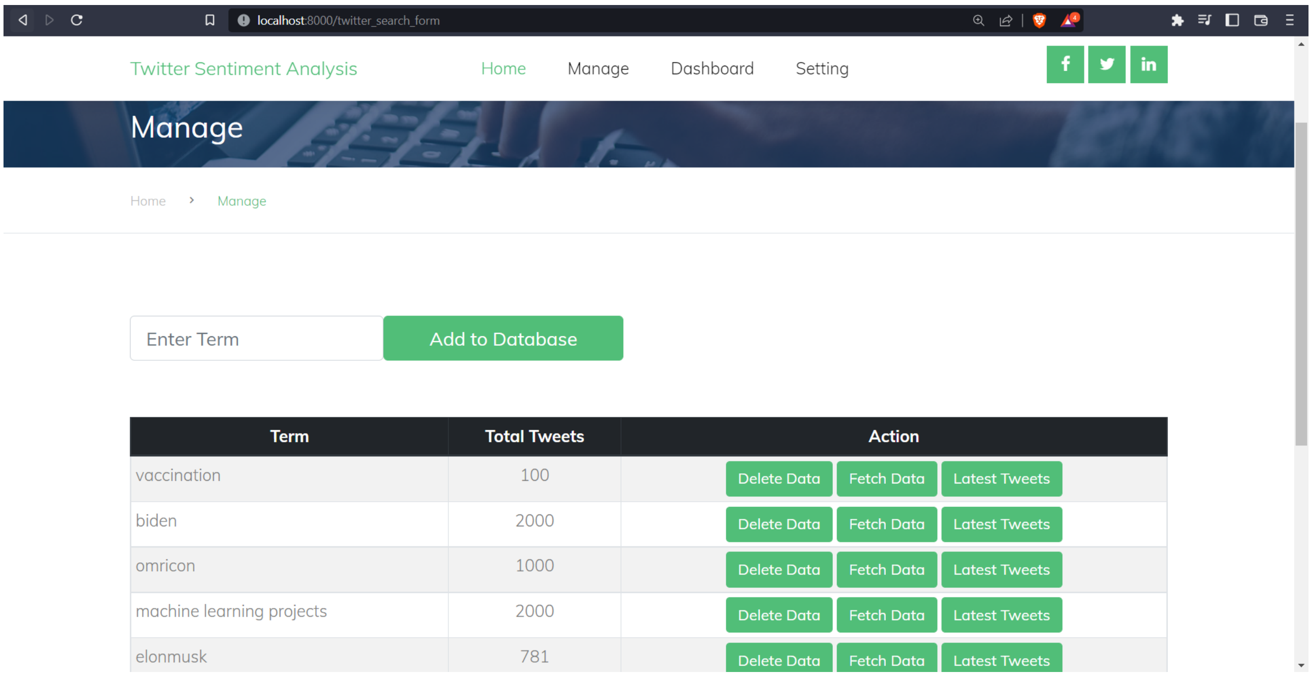


*figure : 9.5 :Home Page of Twitter Sentiment Analysis After Login App.*



| Term | Total Tweets | Action |
|---|---|---|
| vaccination | 100 | Delete Data Fetch Data Latest Tweets |
| biden | 2000 | Delete Data Fetch Data Latest Tweets |
| omricon | 1000 | Delete Data Fetch Data Latest Tweets |
| machine learning projects | 2000 | Delete Data Fetch Data Latest Tweets |
| elonmusk | 781 | Delete Data Fetch Data Latest Tweets |

*figure : 9.6 :  Manage Dashboard of Twitter Sentiment Analysis App.*
*(Enter term field is used to search particular keyword)*

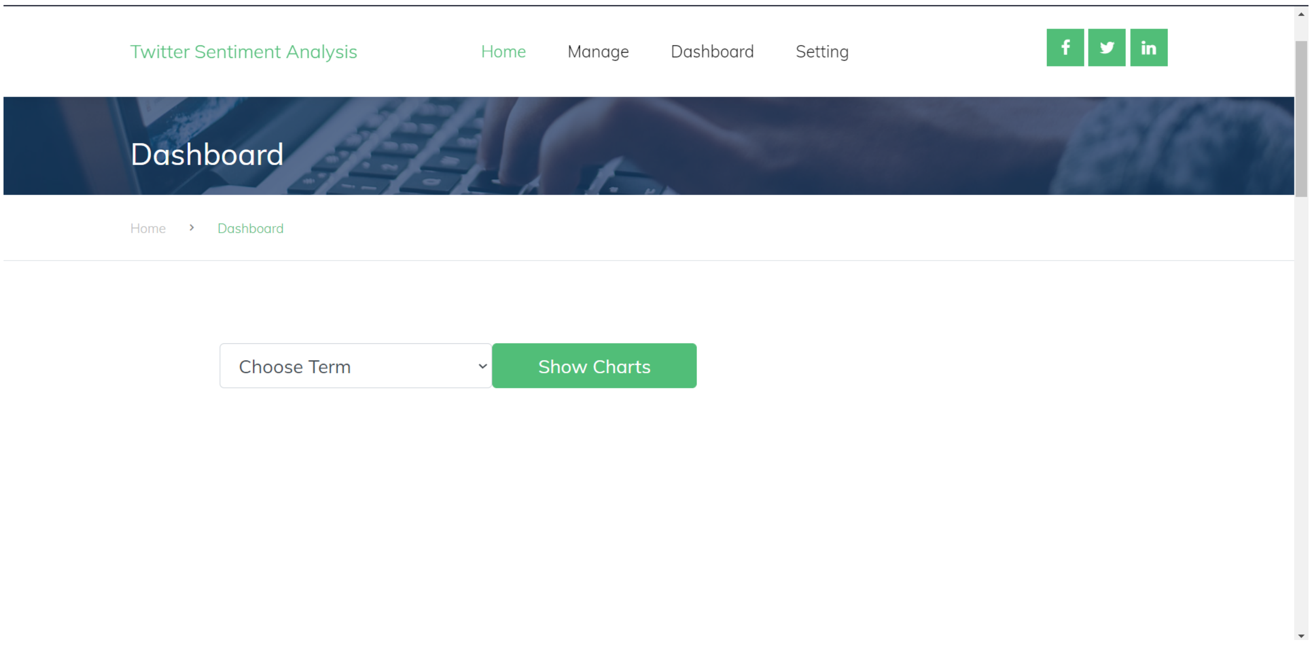


*figure : 9.7 :Visualizing Dashboard of Twitter Sentiment Analysis App Before Choosing Term.*

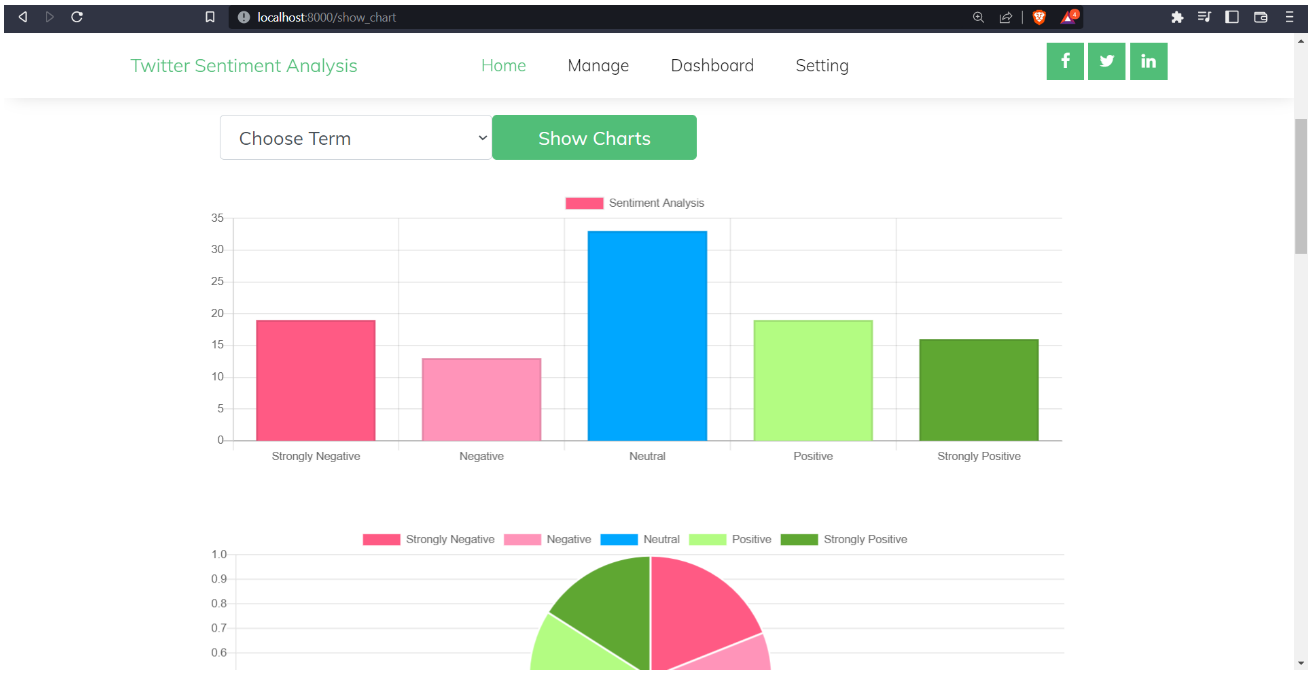


*figure : 9.8.1 :Visualized Dashboard of Twitter Sentiment Analysis App.*

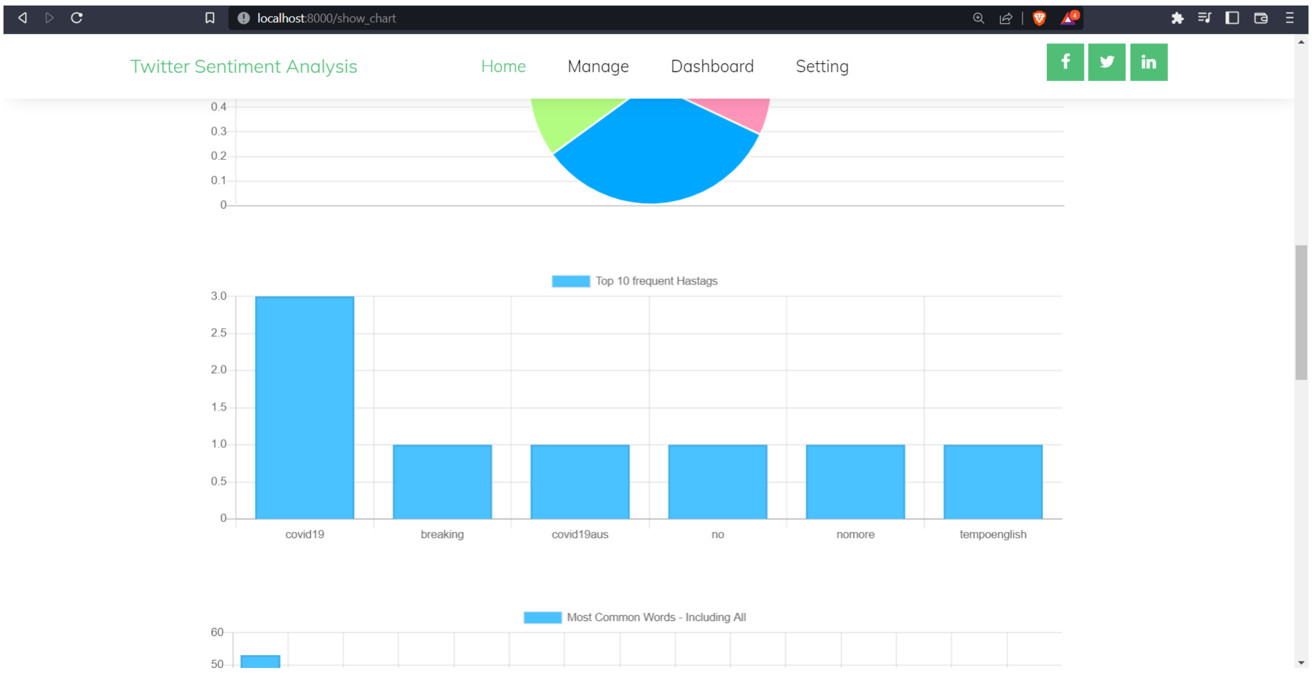
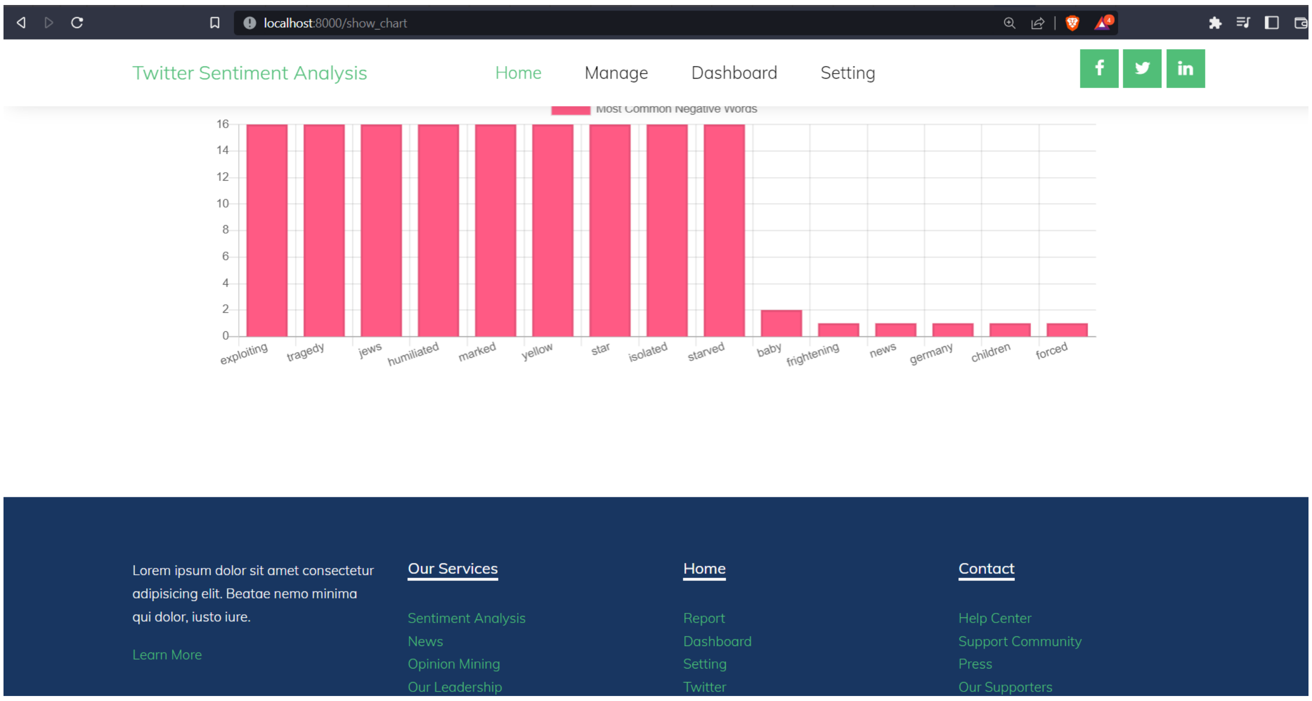


*figure : 9.8.2 :Visualized Dashboard of Twitter Sentiment Analysis App.*



Lorem ipsum dolor sit amet consectetur adipisicing elit. Beatae nemo minima qui dolor, iusto iure.

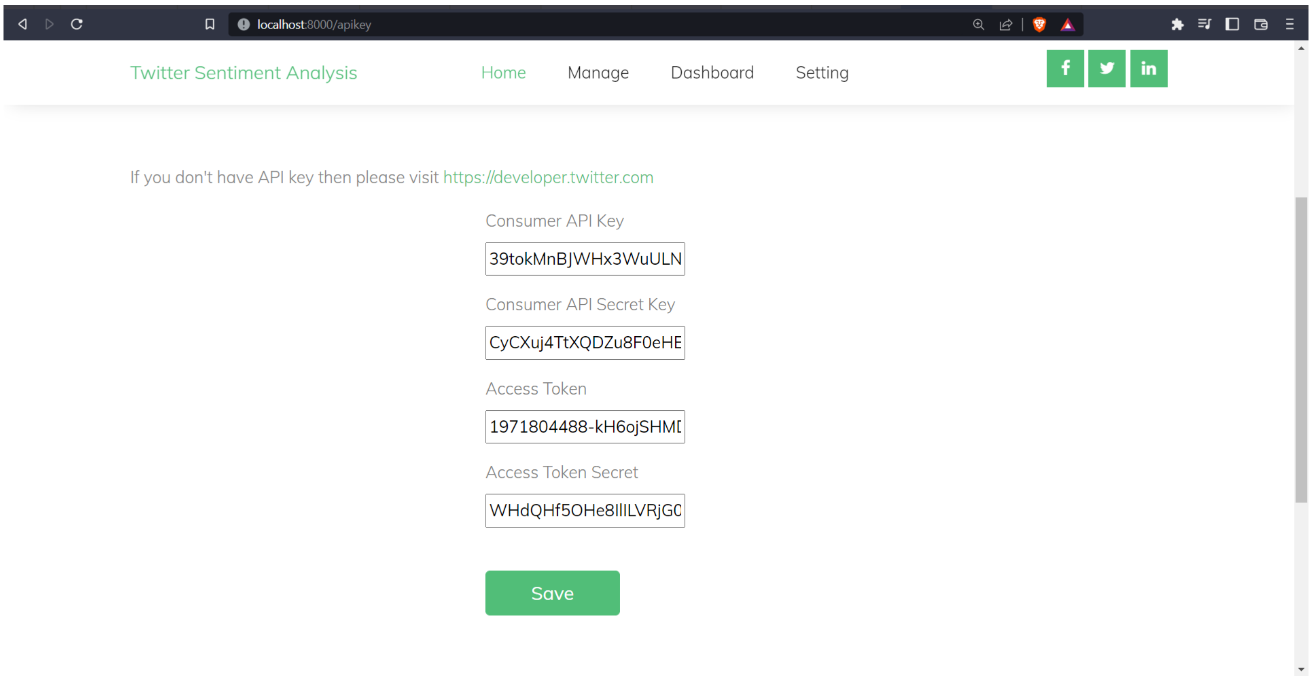
Learn More

Our Services
- Sentiment Analysis
- News
- Opinion Mining
- Our Leadership

Home
- Report
- Dashboard
- Setting
- Twitter

Contact
- Help Center
- Support Community
- Press
- Our Supporters

*figure : 9.8.3 :Visualized Dashboard of Twitter Sentiment Analysis App.*



localhost:8000/apikey

Twitter Sentiment Analysis | Home | Manage | Dashboard | Setting

If you don't have API key then please visit https://developer.twitter.com

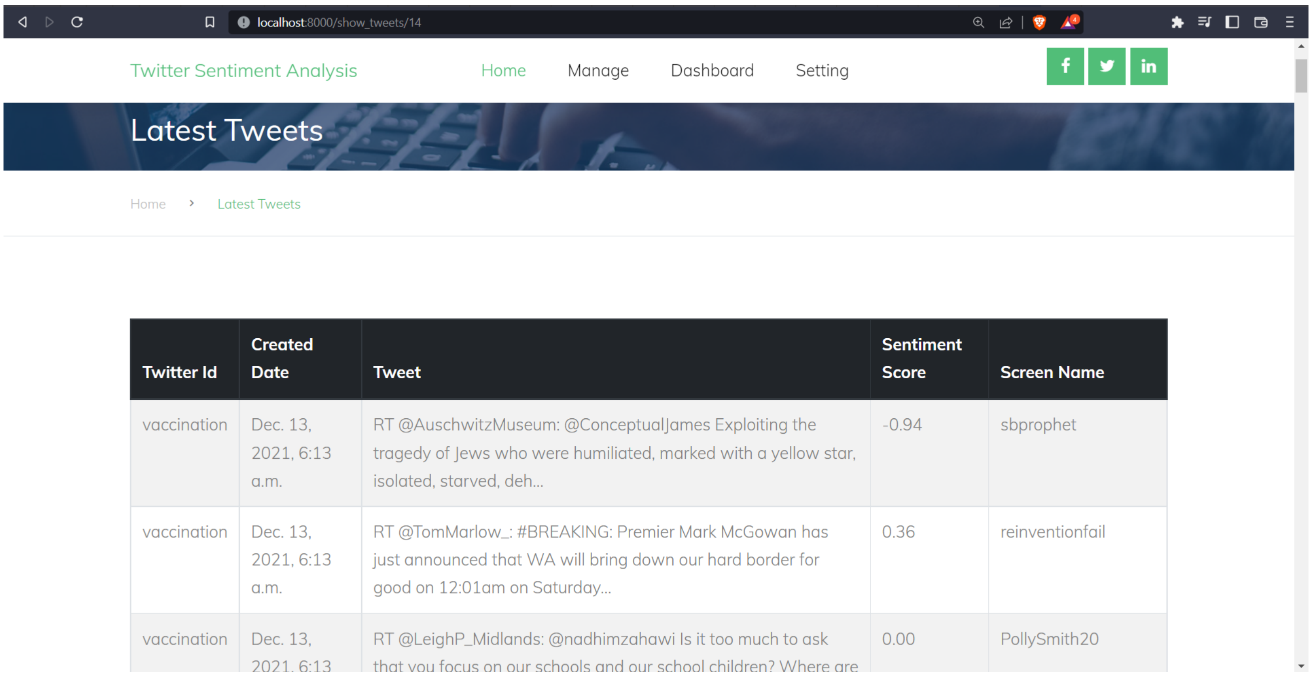
Consumer API Key: 39tokMnBJWHx3WuULN

Consumer API Secret Key: CyCXuj4TtXQDZu8F0eHE

Access Token: 1971804488-kH6ojSHMI

Access Token Secret: WHdQHf5OHe8lllLVRjGC

*figure : 9.9 :Api keys settings of Twitter Sentiment Analysis App.*



localhost:8000/show_tweets/14

Twitter Sentiment Analysis | Home | Manage | Dashboard | Setting

Latest Tweets

Home › Latest Tweets

| Twitter Id | Created Date | Tweet | Sentiment Score | Screen Name |
|---|---|---|---|---|
| vaccination | Dec. 13, 2021, 6:13 a.m. | RT @AuschwitzMuseum: @ConceptualJames Exploiting the tragedy of Jews who were humiliated, marked with a yellow star, isolated, starved, deh... | -0.94 | sbprophet |
| vaccination | Dec. 13, 2021, 6:13 a.m. | RT @TomMarlow_: #BREAKING: Premier Mark McGowan has just announced that WA will bring down our hard border for good on 12:01am on Saturday... | 0.36 | reinventionfail |
| vaccination | Dec. 13, 2021, 6:13 | RT @LeighP_Midlands: @nadhimzahawi Is it too much to ask that you focus on our schools and our school children? Where are | 0.00 | PollySmith20 |

*figure : 9.10 :Latest Tweets of Twitter Sentiment Analysis App.*

## Appendix – B – Coding

## Code

### INDEX.HTML

```
{% extends 'base.html' %}
{% load static %}
{% block content %}

  <div class="hero-slide owl-carousel site-blocks-cover">
    <div class="intro-section" style="background-image: url({% static 'images/hero_1.jpg' %})">
      <div class="container">
        <div class="row align-items-center">
          <div class="col-lg-12 mx-auto text-center" data-aos="fade-up">
            <h1>Twitter Sentiment Analysis</h1>
          </div>
        </div>
      </div>
    </div>

    <div class="intro-section" style="background-image: url({% static 'images/hero_2.jpg' %}); background-size:
600px 400px;">
      <div class="container">
        <div class="row align-items-center">
          <div class="col-lg-12 mx-auto text-center" data-aos="fade-up">
            <h1>Analyze Anything</h1>
          </div>
        </div>
      </div>
    </div>

  </div>

  <div class="site-section">
    <div class="container">
      <div class="row mb-5 justify-content-center text-center">
        <div class="col-lg-10 mb-5">
          <h2 class="section-title-underline mb-5">
            <span>Why Sentiment Analysis is Important</span>
          </h2>
        </div>
      </div>
      <div class="row">
```

```
        <div class="col-lg-4 col-md-6 mb-4 mb-lg-0">

          <div class="feature-1 border">
            <div class="icon-wrapper">
              <img src="{% static 'images/twitter.jpg' %}" alt="Image" class="img-fluid">
            </div>
            <div class="feature-1-content">
              <h2>Brandwatch Analytics</h2>
              <p>Ensuring quick and real-time monitoring capabilities</p>
            </div>
          </div>
        </div>
        <div class="col-lg-4 col-md-6 mb-4 mb-lg-0">
          <div class="feature-1 border">
            <div class="icon-wrapper">
              <img src="{% static 'images/twitter.jpg' %}" alt="Image" class="img-fluid">
            </div>
            <div class="feature-1-content">
              <h2>Contextual Tone</h2>
              <p>Learn to identify the public underlying tone, and detecting moods - Positive, Negative or Neutral</p>
            </div>
          </div>
        </div>
        <div class="col-lg-4 col-md-6 mb-4 mb-lg-0">
          <div class="feature-1 border">
            <div class="icon-wrapper">
              <img src="{% static 'images/twitter.jpg' %}" alt="Image" class="img-fluid">
            </div>
            <div class="feature-1-content">
              <h2>Develop New Products</h2>
              <p>Understand public opinions, and help in developing new products </p>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>


      <!-- // 05 - Block -->
  <div class="site-section ftco-subscribe-1" style="background-image: url({% static 'images/bg_1.jpg' %})">
    <div class="container">
      <div class="row align-items-center">
        <div class="col-lg-7">
          <h2>Subscribe to us!</h2>
        </div>
        <div class="col-lg-5">
          <form action="" class="d-flex">
            <input type="text" class="rounded form-control mr-2 py-3" placeholder="Enter your email">
            <button class="btn btn-primary rounded py-3 px-4" type="submit">Send</button>
          </form>
        </div>
      </div>
    </div>
  </div>

</div>
<!-- .site-wrap -->

          {% endblock %}
```

## dashboard.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}

  <div class="site-section ftco-subscribe-1 site-blocks-cover pb-4" style="background-image: url({% static
'images/bg_1.jpg' %})">
      <div class="container">
        <div class="row align-items-end">
          <div class="col-lg-7">
            <h2 class="mb-0">Dashboard</h2>
          </div>
        </div>
      </div>
  </div>

  <div class="custom-breadcrumns border-bottom">
    <div class="container">
      <a href="index.html">Home</a>
      <span class="mx-3 icon-keyboard_arrow_right"></span>
      <span class="current">Dashboard</span>
    </div>
  </div>

  <div class="container">
      {% if messages %}
          {% for message in messages %}
            <div class="alert alert-warning alert-dismissable" role="alert">
              <button class="close" data-dismiss="alert">
                <small><sup>x</sup></small>
              </button>
              {{ message }}
            </div><br/>
          {% endfor %}
      {% endif %}
  </div>

  <div class="site-section">
    <div class="container" style="padding:0% 8% 0% 8%;">
        <form action="{% url 'show_chart' %}" method="POST" class="form-inline my-2 my-lg-0">
            {% csrf_token %}
            <select name="term_value" class="form-control form-control-lg">
                <option value="choose">Choose Term</option>
                {% if terms %}
                    {% for term in terms %}
                        <option value="{{ term }}">{{ term }}</option>
                    {% endfor %}
                {% endif %}
            </select>
            <input class="btn btn-primary btn-lg px-5" type="submit" value="Show Charts">
        </form>

        <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
        <div class="chart-container">
            <br><canvas id="myChart" name="barchart" class="row justify-content-center"
                    style="display: block; width: 50%; height: 80%;"></canvas><br>

            <br><canvas id="myChart1" name="barchart1" class="row justify-content-center"
                    style="display: block; width: 50%; height: 80%;"></canvas><br>

            <br><canvas id="myChart2" name="barchart2" class="row justify-content-center"
                    style="display: block; width: 50%; height: 80%;"></canvas><br>

            <br><canvas id="myChart3" name="barchart3" class="row justify-content-center"
                    style="display: block; width: 50%; height: 80%;"></canvas><br>

            <br><canvas id="myChart4" name="barchart4" class="row justify-content-center"
                    style="display: block; width: 50%; height: 80%;"></canvas><br>

            <br><canvas id="myChart5" name="barchart5" class="row justify-content-center"
                    style="display: block; width: 50%; height: 80%;"></canvas><br>
        </div>

        <script>
            var ctx = document.getElementById('myChart').getContext('2d');
            var myChart = new Chart(ctx, {
                type: 'bar',
                data: {
```

```
        labels: ['Strongly Negative', 'Negative', 'Neutral', 'Positive', 'Strongly Positive'],
        datasets: [{
            label: 'Sentiment Analysis',
            data: {{ sentimentcat }},
            backgroundColor: [
                'rgba(255, 90, 132, 1)',
                'rgba(255, 148, 185, 1)',
                'rgba(0, 167, 255, 1)',
                'rgba(179, 252, 130, 1)',
                'rgba(95, 167, 50, 1)'
            ],
            borderWidth: 2
        }]
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
    }
});

var ctx = document.getElementById('myChart1').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'pie',
    data: {
        labels: ['Strongly Negative', 'Negative', 'Neutral', 'Positive', 'Strongly Positive'],
        datasets: [{
            label: 'Pie Chart - Sentiment Analysis',
            data: {{ sentimentcat }},
            backgroundColor: [
                'rgba(255, 90, 132, 1)',
                'rgba(255, 148, 185, 1)',
                'rgba(0, 167, 255, 1)',
                'rgba(179, 252, 130, 1)',
                'rgba(95, 167, 50, 1)'
            ],

            borderWidth: 2
        }]
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
    }
});

var ctx = document.getElementById('myChart2').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: {{ hastag_words|safe }},
        datasets: [{
            label: 'Top 10 frequent Hastags',
            data: {{ hastag_count }},
            backgroundColor: 'rgba(74, 194, 255, 1)',

            borderWidth: 2
        }]
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
    }
});

var ctx = document.getElementById('myChart3').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: {{ cleaned_words|safe }},
        datasets: [{
            label: 'Most Common Words - Including All',
            data: {{ cleaned_count }},
            backgroundColor: 'rgba(74, 194, 255, 1)',
            borderWidth: 2
        }]
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
    }
});

var ctx = document.getElementById('myChart4').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: {{ pos_cleaned_words|safe }},
        datasets: [{
            label: 'Most Common Positive Words',
            data: {{ pos_cleaned_count }},
            backgroundColor: 'rgba(95, 167, 50, 1)',
            borderWidth: 2
        }]
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
    }
});

var ctx = document.getElementById('myChart5').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: {{ neg_cleaned_words|safe }},
        datasets: [{
            label: 'Most Common Negative Words',
            data: {{ neg_cleaned_count }},
            backgroundColor: 'rgba(255, 90, 132, 1)',
            borderWidth: 2
        }]
    },
    options: {
```

```
                    scales: {
                        yAxes: [{
                            ticks: {
                                beginAtZero: true
                            }
                        }]
                    }
                }
            });

        </script>
    </div>
  </div>

{% endblock %}

{% extends 'base.html' %}
{% load static %}
{% block content %}

  <div class="site-section ftco-subscribe-1 site-blocks-cover pb-4"
       style="background-image: url({% static 'images/bg_1.jpg' %})">
      <div class="container">
        <div class="row align-items-end">
          <div class="col-lg-7">
            <h2 class="mb-0">Latest Tweets</h2>
          </div>
        </div>
      </div>
    </div>


  <div class="custom-breadcrumns border-bottom">
    <div class="container">
      <a href="index.html">Home</a>
      <span class="mx-3 icon-keyboard_arrow_right"></span>
      <span class="current">Latest Tweets</span>
    </div>
  </div>

  <div class="site-section">
      <div class="container">
          <table class="table table-striped table-bordered">
              <thread class="thead-dark">
                <tr>
                    <th scope="col">Twitter Id</th>
                <th scope="col">Created Date</th>
                <th scope="col">Tweet</th>
                <th scope="col">Sentiment Score</th>
                <th scope="col">Screen Name</th>
              </tr>
            </thead>
            <tbody>
              {% if  tweets %}
              {% for tweet in tweets %}
              <tr>
                 <td> {{ tweet.term }} </td>
                 <td> {{ tweet.created_at }} </td>
                 <td> {{ tweet.text }} </td>
                 <td> {{ tweet.sentiment score }} </td>
                 <td> {{ tweet.screen name }} </td>
              </tr>
              {% endfor %}
              {% endif %}
              </tbody>
            </table>

      </div>
   </div>


{% endblock %}
```

**Manage.py**

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
   os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'project master.settings')
   try:
      from django.core.management import execute_from_command_line
   except ImportError as exc:
      raise ImportError(
         "Couldn't import Django. Are you sure it's installed and "
         "available on your PYTHONPATH environment variable? Did you "
         "forget to activate a virtual environment?"
      ) from exc
   execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':
    main()
```

**Tweets.py**

```
from django.shortcuts import render, redirect
from .models import SearchLog, APIKeys, Twitter
from django.contrib import messages
from .forms import SearchForm, APIKeysForm
import tweepy
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from string import punctuation
import re
from nltk.tokenize import word_tokenize
import pandas as pd
import numpy as np
from collections import Counter

analyzer = SentimentIntensityAnalyzer()
max_tweets = 1000

def index(request):
    return render(request, 'index.html')

# Create your views here.
def contact(request):

    return render(request, 'contact.html')

def manage(request):

    return redirect('twitter_search_form')

def twitter_search_form(request):
    terms = SearchLog.objects.all()
    print('twitter_search_form')
    for term in terms:
        term_count = Twitter.objects.filter(term=term).count()
        obj = SearchLog.objects.get(term=term)
        obj.total_tweet = term_count
        obj.save()
    terms = SearchLog.objects.all()

    if request.method == "POST":
        twitter_searchform = SearchForm(request.POST or None)
        if twitter_searchform.is_valid():
            twitter_searchform.save()
            messages.success(request, "Added to database")
            return redirect('twitter_search_form')
        else:
            messages.success(request, "Already added!!!")
            return redirect('twitter_search_form')
    else:

        if APIKeys.objects.all().exists():
            return render(request, 'manage.html', {'terms': terms})
        else:
            messages.success(request, "No API key found. Please setup the twitter api keys")
            return render(request, 'apikey.html')


def delete_term(request, term_id):
    item = SearchLog.objects.filter(pk=term_id)
    tweets = Twitter.objects.filter(term=item[0])
    item.delete()
    tweets.delete()

    messages.success(request, "Deleted Successfully!!!")
    return redirect(twitter_search_form)


def run_report(request, term_id):
    item = SearchLog.objects.filter(pk=term_id)
    print('item: ', item[0])
    if APIKeys.objects.all().exists():
        api token = APIKeys.objects.latest('id')
        consumer_key = apitoken.api_key
        consumer_secret = apitoken.api_secret_key
        access_token = apitoken.access_token
        access_token_secret = apitoken.access_token_secret
        # Pass OAuth details to tweepy's OAuth handler
        auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_token, access_token_secret)
        # Creating the API object while passing in auth information
        api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
        results = tweepy.Cursor(api.search, q=item[0], lang="en").items(max_tweets)
        objs = (Twitter(term=item[0],
                        twitter id=tweet.id,
                        created_at=tweet.created_at,
                        text=tweet.text,
                        sentiment score=analyzer.polarity_scores(tweet.text)['compound'],
                hashtags=hashtag_extract(tweet.text),
                screen name=tweet.user.screen_name,
                cleaned_text=",".join(PreProcessTweets().processTweets(tweet.text, item[0])))
            for tweet in results)

        Twitter.objects.bulk_create(objs, ignore_conflicts=True)

        messages.success(request, "Successfully fetched twitter data")
        return redirect('twitter_search_form')
    else:
        return redirect('twitter_search_form')

def show_tweets(request, term_id):
    item = SearchLog.objects.filter(pk=term_id)
    tweets = Twitter.objects.filter(term=item[0])
    return render(request,'showtweets.html', {'tweets': tweets})

# Create your views here.
def about(request):
    return render(request, 'about.html', {})


def api key(request):
    if request.method == "POST":
        apiform = APIKeysForm(request.POST or None)
        if apiform.is_valid():
            apiform.save()
            messages.success(request, "API detail added to database")
            return redirect('api key')
        else:
            messages.success(request, "Invalid data. Please try again")
            return redirect('api key')
    else:
        return render(request, 'apikey.html')


def dashboard(request):
    terms = SearchLog.objects.all()
    return render(request, 'dashboard.html', {'terms': terms})


def show_chart(request):
    terms = SearchLog.objects.all()
    bins_cuts = [-1.0, -0.6, -0.1, 0.1, 0.6, 1.0]
    names = ['Strongly Negative', 'Negative', 'Neutral', 'Positive', 'Strongly Positive']
    if request.method == "POST":
```

```
symbol = str(request.POST['term_value'])
print("symbol: ", symbol)
if symbol != "choose":
    data = Twitter.objects.filter(term=symbol)
    print('len(data)', len(data))
    ml_data = list([float(data_item.sentimentscore) for data_item in data])
    sentiment_category = pd.cut(np.array(ml_data), bins_cuts, labels=names)
    print(list(sentiment_category.value_counts()))

    # hashtags Analysis
    total_hastags = []
    for data_item in data:
        if data_item.hastags != "":
            for tt in range(len(data_item.hastags.split(","))):
                total_hastags.append(data_item.hastags.split(",")[tt])
    most_common = Counter(total_hastags).most_common(15)
    hastag_words = []
    hastag_count = []
    for kk in most_common:
        hastag_words.append(str(kk[0]))
        hastag_count.append(kk[1])

    # Frequent Word Analysis
    total_words = []
    for data_item in data:
        if data_item.cleaned_text != "":
            for pp in range(len(data_item.cleaned_text.split(","))):
                total_words.append(data_item.cleaned_text.split(",")[pp])
    most_common = Counter(total_words).most_common(15)
    cleaned_words = []
    cleaned_count = []
    for qq in most_common:
        cleaned_words.append(str(qq[0]))
        cleaned_count.append(qq[1])

    # Frequent Positive Word Analysis
    pos_total_words = []
    for data_item in data:
        if data_item.cleaned_text != "" and data_item.sentimentscore >= 0.6:
            for pos_pp in range(len(data_item.cleaned_text.split(","))):
                pos_total_words.append(data_item.cleaned_text.split(",")[pos_pp])

    most_common = Counter(pos_total_words).most_common(15)
    pos_cleaned_words = []
    pos_cleaned_count = []
    print(most_common)
```

```
            for pos_qq in most_common:
                pos_cleaned_words.append(str(pos_qq[0]))
                pos_cleaned_count.append(pos_qq[1])

            # Frequent Negative Word Analysis
            neg_total_words = []
            for data_item in data:
                if data_item.cleaned_text != "" and data_item.sentimentscore <= -0.6:
                    for neg_pp in range(len(data_item.cleaned_text.split(","))):
                        neg_total_words.append(data_item.cleaned_text.split(",")[neg_pp])
            most_common = Counter(neg_total_words).most_common(15)
            neg_cleaned_words = []
            neg_cleaned_count = []
            for neg_qq in most_common:
                neg_cleaned_words.append(str(neg_qq[0]))
                neg_cleaned_count.append(neg_qq[1])

            return render(request, 'dashboard.html', {'sentimentcat': list(sentiment_category.value_counts()),
                                                      'hastag_words': hastag_words,
                                                      'hastag_count': hastag_count,
                                                      'cleaned_words': cleaned_words,
                                                      'cleaned_count': cleaned_count,
                                                      'pos_cleaned_words': pos_cleaned_words,
                                                      'pos_cleaned_count': pos_cleaned_count,
                                                      'neg_cleaned_words': neg_cleaned_words,
                                                      'neg_cleaned_count': neg_cleaned_count,
                                                      'terms': terms
                                                      })
        else:
            messages.success(request, "Please select the Term")
            return redirect('dashboard')
    else:
        return redirect('dashboard')


# function to collect hashtags
def hashtag_extract(x):
    hashtags = []
    # Loop over the words in the tweet
    for i in xsplit(" "):
        ht = re.findall(r"#(\w+)", i)
        if len(ht) > 0:
            hashtags.append(ht[0].lower())

    return ",".join(hashtags)
```

```
class PreProcessTweets:
    def __init__(self):
        self._stopwords = set(stopwords.words('english') + list(punctuation) + ['AT_USER', 'URL', 'RT', "n't", "..."])

    define processTweets(self, tweet, term):
        tweet = tweet.lower()  # convert text to lower-case
        tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', ' ', tweet)  # remove URLs
        tweet = re.sub('@[^\s]+', ' ', tweet)  # remove usernames
        tweet = re.sub('[^a-zA-Z#]', ' ', tweet)
        tweet = word_tokenize(tweet)  # remove repeated characters (helloooooooo into hello)

        return [word for word in tweet if (word not in self._stopwords) & (len(word) > 3) & (word != term)]
```

## REFERENCES.

**Books:**

Albert Buffet and Eibe Frank. Sentiment Knowledge Discovery in Twitter Streaming Data. Discovery Science, Lecture Notes in Computer Science, 2010, Volume 6332/2010, 1-15, DOI: 10.1007/978-3-642-16184-1_1 Alec Go, Richa Bhayani and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Project Technical Report, Stanford University, 2009. Alexander Pak and Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of international conference on Language Resources and Evaluation (LREC), 2010. Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner and Isabell M. Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2010. Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002. Chenhao Tan, Lilian Lee, Jie Tang, Long Jiang, Ming Zhou and Ping Li. User Level Sentiment Analysis Incorporating Social Networks. In Proceedings of ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD), 2011. Efthymios Kouloumpis, Theresa Wilson and Johanna Moore. Twitter Sentiment Analysis: The Good the Bad and the OMG! In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2011. Hatzivassiloglou, V., & McKeown, K.R.. Predicting the semantic orientation of adjectives. In Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL, 2009

**Websites**:

1 https://www.geeksforgeeks.org/twitter-sentiment-analysis-webapp-using-django/

2 www.google.com