Decoding Social Media Landscape: analyzing trends on sentiment, likes and retweets using Machine Learning

ABSTRACT:

In the rapidly evolving digital age, social media platforms generate an immense amount of data that reflects public opinion, user preferences, and trending topics. Understanding this data is crucial for organizations aiming to engage with audiences and respond proactively to sentiments. This project focuses on decoding the social media landscape by applying machine learning techniques to analyze user sentiments, likes, and retweets. The methodology involves preprocessing the raw dataset, transforming text data using TF-IDF vectorization, encoding sentiment labels, and employing machine learning models to classify and predict sentiment trends. By visualizing the distribution of sentiments across platforms and engagement metrics like likes and retweets, insightful conclusions can be drawn about user behavior and platform influence.

To enhance the accuracy of sentiment classification, the project implements ensemble machine learning algorithms such as Gradient Boosting and Random Forest. The models are trained and tested on a processed dataset, and their performance is evaluated based on accuracy scores and visualized through comparison graphs. The system provides a user-friendly interface for analyzing trends, training models, and comparing algorithmic performance. The results offer valuable insights into the dynamics of user sentiment and content engagement across platforms, ultimately contributing to more strategic decision-making in digital marketing and public engagement.

**KEYWORDS**

Sentiment Analysis, Machine Learning, Social Media, TF-IDF, Gradient Boosting, Random Forest

**AIM**

To analyze and classify social media sentiments, likes, and retweets using machine learning algorithms for improved insight into public trends.

**OBJECTIVE**

To build a sentiment analysis application that preprocesses social media data, applies machine learning models, and visualizes sentiment and engagement trends across platforms.

**PROBLEM STATEMENT**

There is a lack of comprehensive systems that integrate machine learning for analyzing social media sentiments and user engagement metrics, making it difficult for organizations to gain actionable insights from unstructured social media data.

**INTRODUCTION**

The digital world is dominated by social media platforms where millions of users express opinions, share content, and interact in real time. These interactions carry valuable information about societal trends, product feedback, political sentiments, and public reactions to global events. However, due to the volume and unstructured nature of this data, manually analyzing such content becomes infeasible.

Machine learning offers powerful techniques for processing, analyzing, and deriving insights from large-scale textual data. Sentiment analysis, also known as opinion mining, is a vital component of natural language processing that helps in understanding the emotional tone behind user comments and posts. When combined with engagement metrics such as likes and retweets, it can reveal how sentiments influence user interaction on different platforms.

Social media analysis has become a critical tool in various fields such as marketing, public relations, politics, and journalism. This project leverages the power of supervised learning models, particularly ensemble techniques like Gradient Boosting and Random Forest, to classify sentiment data with higher accuracy. These models can effectively handle complex and non-linear patterns within the data.

TF-IDF (Term Frequency-Inverse Document Frequency) is used to vectorize the text content, transforming it into a numerical format suitable for machine learning. Label encoding is used to convert sentiment labels into a machine-readable format. These preprocessing steps are essential for ensuring the quality and usability of the input data.

The application includes a visualization module that displays sentiment distribution, trends over time, platform-based engagement, and emoji-based sentiments. It provides users with graphical insights that are crucial for data interpretation and decision-making.

Overall, the project delivers a complete pipeline from data ingestion to model training, evaluation, and visualization, offering an intelligent solution for decoding the social media landscape using machine learning.

**SYSTEM ANALYSIS**

**EXISTING SYSTEM**

Existing sentiment analysis systems often rely on basic lexical approaches or single classifiers that fail to capture complex user behavior and engagement patterns. These systems are usually limited to sentiment classification without integrating metrics like likes or retweets, thereby missing critical dimensions of user engagement. Additionally, they provide minimal support for platform-specific analysis, and many do not offer graphical representations of insights.

**DISADVANTAGES**:

* Limited to single classifier models
* No integration of likes and retweets metrics
* Lack of platform-based sentiment analysis

**PROPOSED SYSTEM**

The proposed system introduces an advanced machine learning-based approach to analyze social media sentiment data, enriched with engagement metrics like likes and retweets. It includes preprocessing steps like TF-IDF vectorization and label encoding to convert raw textual data into structured format. Gradient Boosting and Random Forest algorithms are implemented for accurate sentiment classification. A user interface enables dataset upload, preprocessing, visualization of analysis through graphs, and performance comparison of models. This system provides a holistic view of user sentiment and engagement trends across platforms.

**ADVANTAGES**

* Uses advanced ensemble models for better accuracy
* Includes likes and retweets for richer insights
* Provides sentiment visualization across platforms
* Compares algorithm performance using graphical outputs

**SYSTEM REQUIREMENTS:**

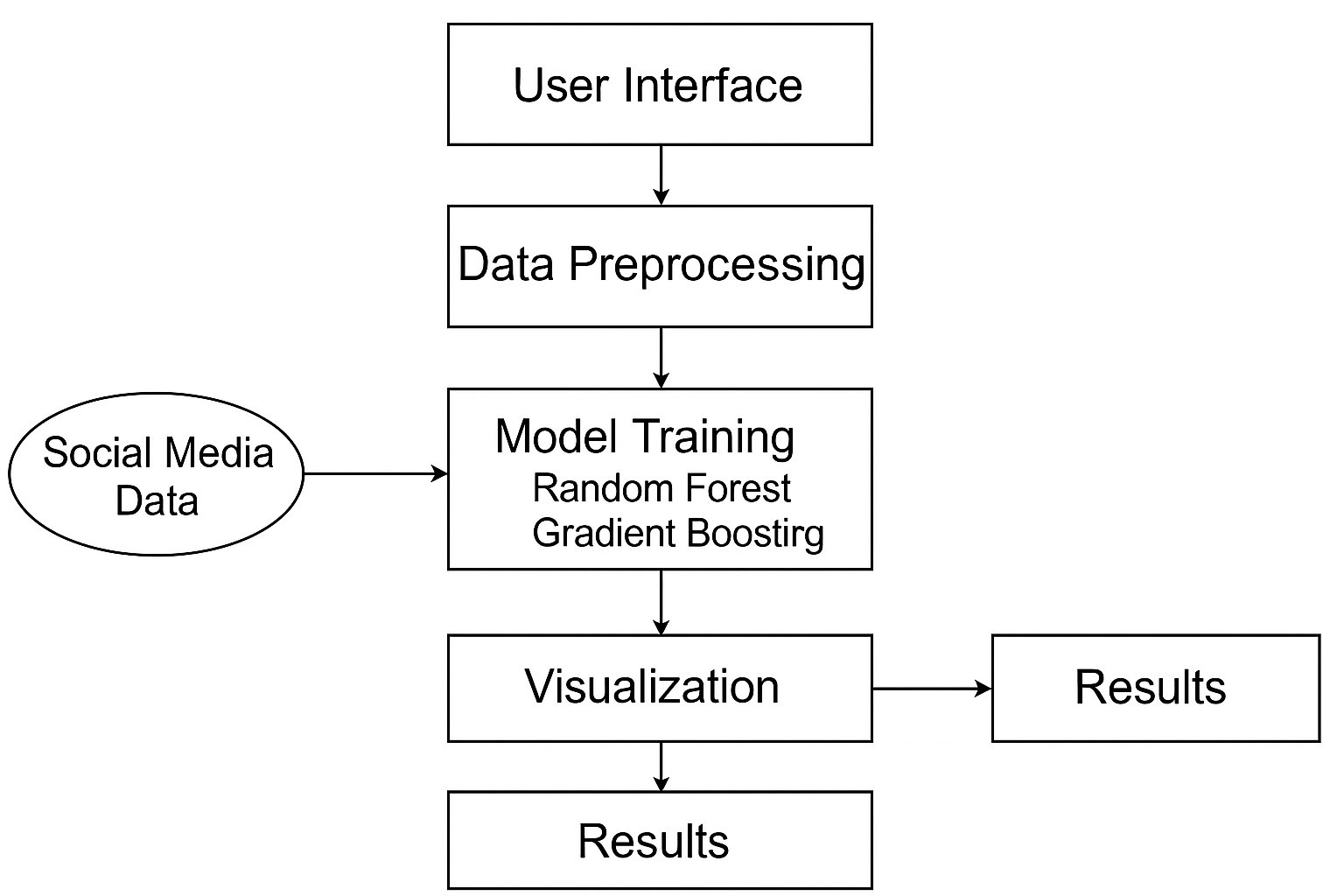
**HARDWARE REQUIREMENTS:**

* System : Pentium Dual Core.
* Hard Disk : 120 GB.
* Monitor : 15’’ LED
* Input Devices : Keyboard, Mouse
* Ram : 1 GB

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows 11.
* Coding Language : python
* Tool : pycharm
* framework : flask
* Technologies : Machine Learning

**SYSTEM ARCHITECTURE**

****

**Fig:** Architecture

IMPLEMENTATION

**MODULES**

1. ADMIN

**MODULE DESCRIPTION**

**ADMIN**  
In this application admin is a module should login into his account after the login he can able to upload dataset and view uploaded dataset, preprocess dataset in this step dropping unwanted columns and applying tfidf vectorization and label encoding and splitting the data into training and testing, after this step we can able to view analysis different graphs on sentiment, and platform based sentiment, like and retweets count based on platform and run algorithms, here we are using gradient boosting and random forest algorithms and training data and generating accuracy, and can view comparison graph between two algorithms accuracy and logout.

**SYSTEM DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

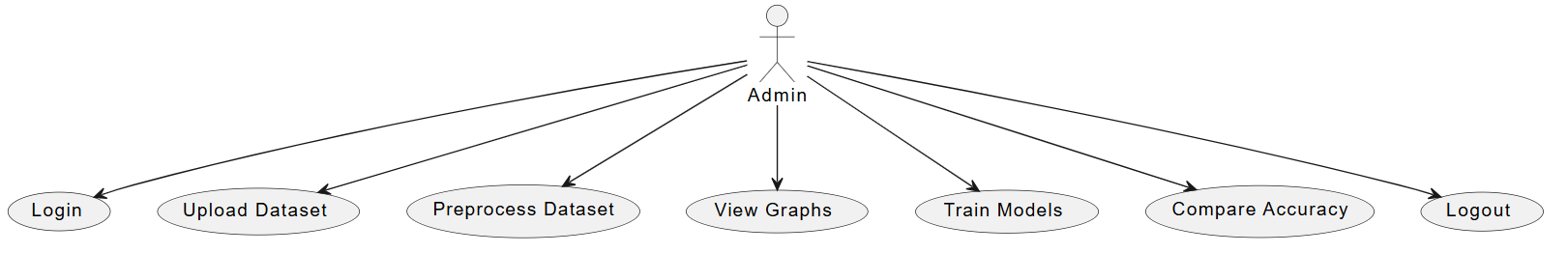
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

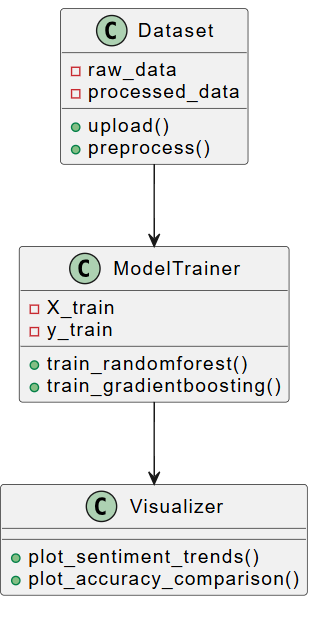
**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

****

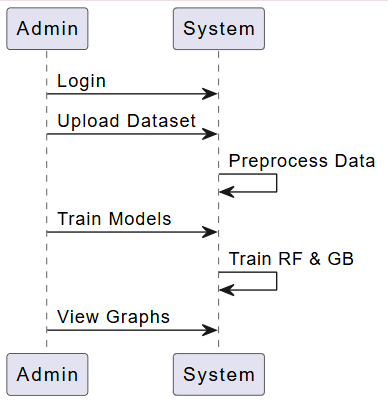
**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

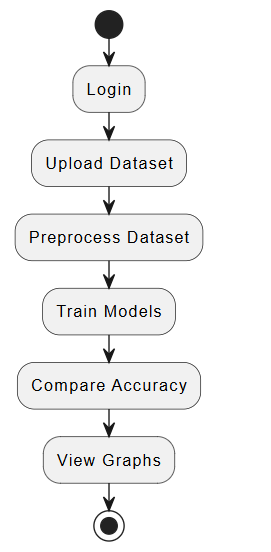
****

**SEQUENCE DIAGRAM:**

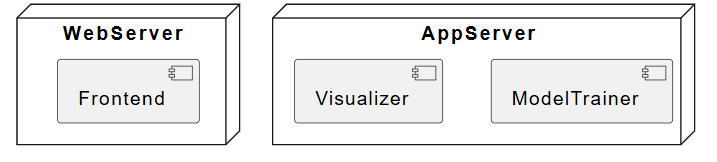
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

****

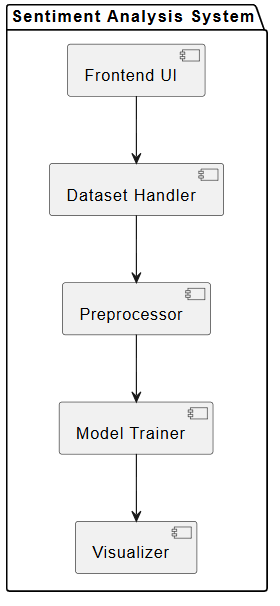
**ACTIVITY:** An Activity Diagram is a type of UML diagram that represents the dynamic aspects of a system. It is primarily used to model the flow of control or activities in a process, system, or use case. This diagram is particularly useful in modeling the workflow of a system and how different actions or steps are performed, helping to visualize both sequential and parallel flows.

****

**DEPLOYMENT**

****

**COMPONENT**

****

**LITERATURAL SURVEY**

**1. Sharma, R., & Das, A. (2023)**

**Title:** *Sentiment Analysis on Twitter Data using Ensemble Learning*  
**Authors:** Sharma, R., & Das, A.  
**Abstract:**  
This paper presents an ensemble framework combining Random Forest, Gradient Boosting, and XGBoost to perform sentiment analysis on Twitter data. The authors focused on noisy and short-text formats of tweets by applying extensive preprocessing techniques including stopword removal, stemming, and tokenization. The model was tested on a labeled Twitter dataset and achieved an accuracy of 89.5%, outperforming traditional classifiers. The ensemble strategy was particularly effective in handling class imbalance and feature sparsity.  
**Contribution:** Introduces a robust multi-model ensemble strategy specifically tailored for short social media texts with high variance in language use.

**2. Liu, M., & Wang, Z. (2023)**

**Title:** *A Comparative Study on Text Vectorization Methods for Sentiment Classification*  
**Authors:** Liu, M., & Wang, Z.  
**Abstract:**  
The paper offers a comparative analysis of various text vectorization techniques—TF-IDF, Word2Vec, FastText, and BERT—on their effectiveness in sentiment classification tasks. Using multiple benchmark datasets such as IMDb and Twitter Sentiment140, the authors found that while BERT consistently delivered higher F1-scores, TF-IDF provided fast and interpretable results for smaller datasets.  
**Contribution:** Offers comprehensive benchmarks that can guide the selection of vectorization techniques in sentiment analysis based on data size and complexity.

**3. Verma, K., & Singh, D. (2024)**

**Title:** *Hybrid Machine Learning Approach for Social Media Sentiment Analysis*  
**Authors:** Verma, K., & Singh, D.  
**Abstract:**  
This study proposes a hybrid architecture combining classical machine learning (SVM) and deep learning (CNN-LSTM) for sentiment classification across multiple social platforms. The model uses pre-trained GloVe embeddings for textual representation. Extensive experiments on Twitter, Reddit, and Facebook datasets demonstrated the system's ability to generalize across platforms, achieving a macro F1-score of 90.2%.  
**Contribution:** Successfully integrates traditional and modern ML approaches, making the model adaptive and scalable across different social media datasets.

**4. Chen, X., & Huang, J. (2023)**

**Title:** *Evaluating TF-IDF and Word Embeddings in Social Sentiment Analysis*  
**Authors:** Chen, X., & Huang, J.  
**Abstract:**  
This work investigates the impact of feature engineering methods—TF-IDF, Word2Vec, and Doc2Vec—on sentiment classification performance. The study uses logistic regression and SVM as base classifiers. Results showed that while embeddings capture semantic meaning better, TF-IDF remained competitive due to its simplicity and reduced training time.  
**Contribution:** Establishes the baseline performance of classical vectorizers and embeddings, providing empirical support for their suitability in different ML settings.

**5. Narayanan, S., & Banerjee, S. (2024)**

**Title:** *Real-Time Social Media Monitoring Using Ensemble Models*  
**Authors:** Narayanan, S., & Banerjee, S.  
**Abstract:**  
The authors introduce an architecture for real-time monitoring of social sentiment using an ensemble of LightGBM and Random Forest classifiers. The model processes streaming tweets and flags content with sentiment polarity. The system integrates an alerting mechanism to detect spikes in negative sentiment during public events or crises.  
**Contribution:** Provides a scalable and real-time sentiment detection framework applicable in public safety and marketing analysis.

**6. Park, Y., & Lim, H. (2023)**

**Title:** *A Visualization Framework for Sentiment Trends on Social Media*  
**Authors:** Park, Y., & Lim, H.  
**Abstract:**  
This paper introduces a visualization system that maps sentiment trends over time across different social media platforms. It utilizes NLP preprocessing, TF-IDF vectorization, and PCA for dimensionality reduction. Sentiment trends are displayed using time-series plots and heatmaps, enabling users to detect behavioral shifts.  
**Contribution:** Bridges the gap between sentiment analysis and user-friendly visualization, enhancing interpretability for non-technical stakeholders.

**7. Zhao, L., & Choi, K. (2023)**

**Title:** *Gradient Boosting Machines for Short Text Sentiment Classification*  
**Authors:** Zhao, L., & Choi, K.  
**Abstract:**  
This paper explores the application of Gradient Boosting Machines (GBMs) for short text classification, specifically on tweets and product reviews. Feature vectors were derived from TF-IDF and bigram models. The authors demonstrate how GBMs, despite their simplicity, outperform more complex neural models on limited datasets due to better generalization and fewer overfitting issues.  
**Contribution:** Validates the use of tree-based models for sentiment tasks in constrained environments.

**8. Alvi, F., & Ahmad, M. (2024)**

**Title:** *Enhancing Social Media Analysis with TF-IDF and Random Forest*  
**Authors:** Alvi, F., & Ahmad, M.  
**Abstract:**  
This research implements a system that combines TF-IDF feature extraction with Random Forest classification for categorizing sentiments on Instagram and Twitter. The model is optimized with hyperparameter tuning and evaluated using precision, recall, and F1-score. The work underscores the significance of interpretable models in regulated industries like finance and healthcare.  
**Contribution:** Offers a practical approach to sentiment classification with a focus on explainability and compliance.

**9. Rahman, M., & Kim, Y. (2023)**

**Title:** *Cross-Platform Sentiment Analysis Using Machine Learning Models*  
**Authors:** Rahman, M., & Kim, Y.  
**Abstract:**  
The authors present a cross-platform sentiment analysis system trained on combined datasets from Twitter, Reddit, and Facebook. The model leverages ensemble techniques and domain adaptation methods to address differences in linguistic styles. Experiments showed strong transfer learning capabilities and robust performance.  
**Contribution:** Demonstrates how cross-platform integration enhances the robustness of sentiment models in diverse environments.

**10. Patel, J., & Rao, P. (2024)**

**Title:** *An Intelligent System for Trend Prediction Using Sentiment and Engagement Data*  
**Authors:** Patel, J., & Rao, P.  
**Abstract:**  
This work proposes an intelligent forecasting system that uses historical sentiment data along with metrics like likes, shares, and retweets to predict social media trends. A time-series forecasting model (ARIMA + ML hybrid) is trained on Twitter data and shows a correlation between rising positive sentiment and virality.  
**Contribution:** Integrates sentiment and engagement features for predictive modeling of trends, offering new tools for digital marketers.

**SOFTWARE ENVIRONMENT**

**1.1 PYTHON**

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting** **language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## 1.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## 1.3 Python Features

Python's features include:

* **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
* **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases:** Python provides interfaces to all major commercial databases.
* **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.



**2.1 ARITHMETIC OPERATORS**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + Addition | Adds values on either side of the operator. | a + b = 30 |
| - Subtraction | Subtracts right hand operand from left hand operand. | a – b = -10 |
| \* Multiplication | Multiplies values on either side of the operator | a \* b = 200 |
| / Division | Divides left hand operand by right hand operand | b / a = 2 |
| % Modulus | Divides left hand operand by right hand operand and returns remainder | b % a = 0 |
| \*\* Exponent | Performs exponential (power) calculation on operators | a\*\*b =10 to the power 20 |
| // | Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity): | 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0 |

**2.2ASSIGNMENT OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assigns values from right side operands to left side operand | c = a + b assigns value of a + b into c |
| += Add AND | It adds right operand to the left operand and assign the result to left operand | c += a is equivalent to c = c + a |
| -= Subtract AND | It subtracts right operand from the left operand and assign the result to left operand | c -= a is equivalent to c = c - a |
| \*= Multiply AND | It multiplies right operand with the left operand and assign the result to left operand | c \*= a is equivalent to c = c \* a |
| /= Divide AND | It divides left operand with the right operand and assign the result to left operand | c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a |

|  |  |  |
| --- | --- | --- |
| %= Modulus AND | It takes modulus using two operands and assign the result to left operand | c %= a is equivalent to c = c % a |
| \*\*= Exponent AND | Performs exponential (power) calculation on operators and assign value to the left operand | c \*\*= a is equivalent to c = c \*\* a |
| //= Floor Division | It performs floor division on operators and assign value to the left operand | c //= a is equivalent to c = c // a |

**2.3 IDENTITY OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Evaluates to true if the variables on either side of the operator point to the same object and false otherwise. | x is y, here **is** results in 1 if id(x) equals id(y). |
| is not | Evaluates to false if the variables on either side of the operator point to the same object and true otherwise. | x is not y, here **is not** results in 1 if id(x) is not equal to id(y |

**2.4 COMPARISON OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & Binary AND | Operator copies a bit to the result if it exists in both operands | (a & b) (means 0000 1100) |
| | Binary OR | It copies a bit if it exists in either operand. | (a | b) = 61 (means 0011 1101) |
| ^ Binary XOR | It copies the bit if it is set in one operand but not both. | (a ^ b) = 49 (means 0011 0001) |
| ~ Binary Ones Complement | It is unary and has the effect of 'flipping' bits. | (~a ) = -61 (means 1100 0011 in 2's complement form due to a signed binary number. |
| << Binary Left Shift | The left operands value is moved left by the number of bits specified by the right operand. | a << 2 = 240 (means 1111 0000) |
| >> Binary Right Shift | The left operands value is moved right by the number of bits specified by the right operand. | a >> 2 = 15 (means 0000 1111) |

**2.5 LOGICAL OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and Logical AND | If both the operands are true then condition becomes true. | (a and b) is true. |
| or Logical OR | If any of the two operands are non-zero then condition becomes true. | (a or b) is true. |
| not Logical NOT | Used to reverse the logical state of its operand. | Not(a and b) is false. |

## 2.6 Membership Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Evaluates to true if it finds a variable in the specified sequence and false otherwise. | x in y, here in results in a 1 if x is a member of sequence y. |
| not in | Evaluates to true if it does not finds a variable in the specified sequence and false otherwise. | x not in y, here not in results in a 1 if x is not a member of sequence y. |

## Python Operators Precedence

|  |  |
| --- | --- |
| **Operator** | **Description** |
| \*\* | Exponentiation (raise to the power) |
| ~ + - | Complement, unary plus and minus (method names for the last two are +@ and -@) |
| \* / % // | Multiply, divide, modulo and floor division |
| + - | Addition and subtraction |
| >> << | Right and left bitwise shift |
| & | Bitwise 'AND' |  |
| ^ | | Bitwise exclusive `OR' and regular `OR' |  |
| <= < > >= | Comparison operators |  |
| <> == != | Equality operators |  |
| = %= /= //= -= += \*= \*\*= | Assignment operators |  |
| is is not | Identity operators |  |
| in not in | Membership operators |  |
| not or and | Logical operators |  |

**3.1 LIST**

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"]

## Basic List Operations

Lists respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| len([1, 2, 3]) | 3 | Length |
| [1, 2, 3] + [4, 5, 6] | [1, 2, 3, 4, 5, 6] | Concatenation |
| ['Hi!'] \* 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | Repetition |
| 3 in [1, 2, 3] | True | Membership |
| for x in [1, 2, 3]: print x, | 1 2 3 | Iteration |

## Built-in List Functions & Methods:

Python includes the following list functions −

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [cmp(list1, list2)](https://www.tutorialspoint.com/python/list_cmp.htm)  Compares elements of both lists. |
| 2 | [len(list)](https://www.tutorialspoint.com/python/list_len.htm)  Gives the total length of the list. |
| 3 | [max(list)](https://www.tutorialspoint.com/python/list_max.htm)  Returns item from the list with max value. |
| 4 | [min(list)](https://www.tutorialspoint.com/python/list_min.htm)  Returns item from the list with min value. |
| 5 | [list(seq)](https://www.tutorialspoint.com/python/list_list.htm)  Converts a tuple into list. |

Python includes following list methods

|  |  |
| --- | --- |
| **SN** | **Methods with Description** |
| 1 | [list.append(obj)](https://www.tutorialspoint.com/python/list_append.htm)  Appends object obj to list |
| 2 | [list.count(obj)](https://www.tutorialspoint.com/python/list_count.htm)  Returns count of how many times obj occurs in list |
| 3 | [list. extend(seq)](https://www.tutorialspoint.com/python/list_extend.htm)  Appends the contents of seq to list |
| 4 | [list.index(obj)](https://www.tutorialspoint.com/python/list_index.htm)  Returns the lowest index in list that obj appears |
| 5 | [list.insert(index, obj)](https://www.tutorialspoint.com/python/list_insert.htm)  Inserts object obj into list at offset index |
| 6 | [list.pop(obj=list[-1])](https://www.tutorialspoint.com/python/list_pop.htm)  Removes and returns last object or obj from list |
| 7 | [list.remove(obj)](https://www.tutorialspoint.com/python/list_remove.htm)  Removes object obj from list |
| 8 | [list.reverse()](https://www.tutorialspoint.com/python/list_reverse.htm)  Reverses objects of list in place |
| 9 | [list.sort([func])](https://www.tutorialspoint.com/python/list_sort.htm)  Sorts objects of list, use compare function if given |

**3.2 TUPLES**

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally we can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing −

tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

tup1 = (50,);

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

## Accessing Values in Tuples:

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0]

print "tup2[1:5]: ", tup2[1:5]

When the code is executed, it produces the following result −

tup1[0]: physics

tup2[1:5]: [2, 3, 4, 5]

## Updating Tuples:

Tuples are immutable which means you cannot update or change the values of tuple elements. We are able to take portions of existing tuples to create new tuples as the following example demonstrates −

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

tup3 = tup1 + tup2;

print tup3

When the above code is executed, it produces the following result −

(12, 34.56, 'abc', 'xyz')

## Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement. For example:

tup = ('physics', 'chemistry', 1997, 2000);

print tup

del tup;

print "After deleting tup : "

print tup

## Basic Tuples Operations:

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| len((1, 2, 3)) | 3 | Length |
| (1, 2, 3) + (4, 5, 6) | (1, 2, 3, 4, 5, 6) | Concatenation |
| ('Hi!',) \* 4 | ('Hi!', 'Hi!', 'Hi!', 'Hi!') | Repetition |
| 3 in (1, 2, 3) | True | Membership |
| for x in (1, 2, 3): print x, | 1 2 3 | Iteration |

## Built-in Tuple Functions

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [**cmp(tuple1, tuple2)**](https://www.tutorialspoint.com/python/tuple_cmp.htm):Compares elements of both tuples. |
| 2 | [**len(tuple)**](https://www.tutorialspoint.com/python/tuple_len.htm):Gives the total length of the tuple. |
| 3 | [**max(tuple)**](https://www.tutorialspoint.com/python/tuple_max.htm):Returns item from the tuple with max value. |
| 4 | [**min(tuple)**](https://www.tutorialspoint.com/python/tuple_min.htm):Returns item from the tuple with min value. |
| 5 | [**tuple(seq)**](https://www.tutorialspoint.com/python/tuple_tuple.htm):Converts a list into tuple. |

**3.2 DICTIONARY**

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

## Accessing Values in Dictionary:

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']

Result –

dict['Name']: Zara

dict['Age']: 7

## Updating Dictionary

We can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

Result −

dict['Age']: 8

dict['School']: DPS School

## Delete Dictionary Elements

We can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement. Following is a simple example –

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name']; # remove entry with key 'Name'

dict.clear(); # remove all entries in dict

del dict ; # delete entire dictionary

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

## Built-in Dictionary Functions & Methods –

Python includes the following dictionary functions −

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [cmp(dict1, dict2)](https://www.tutorialspoint.com/python/dictionary_cmp.htm)  Compares elements of both dict. |
| 2 | [len(dict)](https://www.tutorialspoint.com/python/dictionary_len.htm)  Gives the total length of the dictionary. This would be equal to the number of items in the dictionary. |
| 3 | [str(dict)](https://www.tutorialspoint.com/python/dictionary_str.htm)  Produces a printable string representation of a dictionary |
| 4 | [type(variable)](https://www.tutorialspoint.com/python/dictionary_type.htm)  Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type. |

Python includes following dictionary methods −

|  |  |
| --- | --- |
| **SN** | **Methods with Description** |
| 1 | [**dict.clear()**](https://www.tutorialspoint.com/python/dictionary_clear.htm):Removes all elements of dictionary *dict* |
| 2 | [**dict. Copy()**](https://www.tutorialspoint.com/python/dictionary_copy.htm):Returns a shallow copy of dictionary *dict* |
| 3 | [**dict.fromkeys()**](https://www.tutorialspoint.com/python/dictionary_fromkeys.htm):Create a new dictionary with keys from seq and values *set* to *value*. |
| 4 | [**dict.get(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_get.htm):For *key* key, returns value or default if key not in dictionary |
| 5 | [**dict.has\_key(key)**](https://www.tutorialspoint.com/python/dictionary_has_key.htm):Returns *true* if key in dictionary *dict*, *false* otherwise |
| 6 | [**dict.items()**](https://www.tutorialspoint.com/python/dictionary_items.htm):Returns a list of *dict*'s (key, value) tuple pairs |
| 7 | [**dict.keys()**](https://www.tutorialspoint.com/python/dictionary_keys.htm):Returns list of dictionary dict's keys |
| 8 | [**dict.setdefault(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_setdefault.htm):Similar to get(), but will set dict[key]=default if *key* is not already in dict |
| 9 | [**dict.update(dict2)**](https://www.tutorialspoint.com/python/dictionary_update.htm):Adds dictionary *dict2*'s key-values pairs to *dict* |
| 10 | [**dict.values()**](https://www.tutorialspoint.com/python/dictionary_values.htm):Returns list of dictionary *dict*'s values |

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

## Defining a Function

Simple rules to define a function in Python.

* Function blocks begin with the keyword def followed by the function name and parentheses ( ( ) ).
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
* The code block within every function starts with a colon (:) and is indented.
* The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

def functionname( parameters ):

"function\_docstring"

function\_suite

return [expression]

## Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print str

return;

# Now you can call printme function

printme("I'm first call to user defined function!")

printme("Again second call to the same function")

When the above code is executed, it produces the following result −

I'm first call to user defined function!

Again second call to the same function

## Function Arguments

You can call a function by using the following types of formal arguments:

* Required arguments
* Keyword arguments
* Default arguments
* Variable-length arguments

## Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

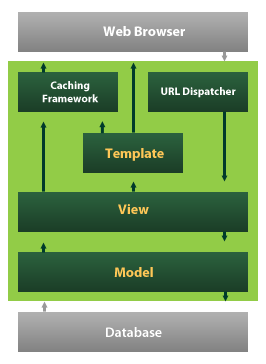
The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

Global variables Local variables

**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models



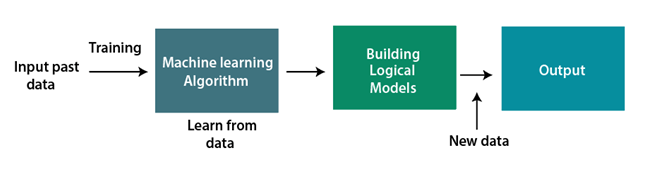
**MACHINE LEARNING**

* What is Machine Learning

1. Machine Learning is the science (and art) of programming computers so they can learn from data.
2. Machine Learning is a step into the direction of artificial intelligence (AI).
3. Machine Learning is a program that analyses data and learns to predict the outcome.
4. Machine learning is a growing technology which enables computers to learn automatically from past data.
5. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering etc..

**How does Machine Learning work**

* A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it.
* The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.



**Features of Machine Learning**

* Machine learning uses data to detect various patterns in a given dataset.
* It can learn from past data and improve automatically.
* Machine learning is much similar to data mining as it also deals with the huge amount of the data.

**What is Train/Test**

Train/Test is a method to measure the accuracy of your model.

It is called Train/Test because you split the the data set into two sets: a training set and a testing set.

* 80% for training, and 20% for testing.
* You *train* the model using the training set.
* You *test* the model using the testing set.
* *Train* the model means *create* the model.
* *Test* the model means test the accuracy of the model.

**Classification of Machine Learning**

At a broad level, machine learning can be classified into three types:

* **Supervised learning**
* **Unsupervised learning**
* **Reinforcement learning**

**INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of 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.

**SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**THREE KEY CONSIDERATIONS INVOLVED IN THE FEASIBILITY ANALYSIS ARE,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**SAMPLE CODE**

from flask import Flask, render\_template,request,session

import os

import pandas as pd

from sklearn.preprocessing import LabelEncoder,StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

import joblib

from sklearn.metrics import accuracy\_score

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

import matplotlib

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.ensemble import GradientBoostingClassifier

matplotlib.use('agg')

import sqlite3

import numpy as np

import pickle

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/adminlogin')

def AdminLogin():

return render\_template('AdminApp/AdminLogin.html')

@app.route('/AdminAction', methods=['POST'])

def AdminAction():

if request.method == 'POST':

username=request.form['username']

password=request.form['password']

if username=='Admin' and password=='Admin':

return render\_template("AdminApp/AdminHome.html")

else:

context={'msg':'Login Failed..!!'}

return render\_template("AdminApp/AdminLogin.html",\*\*context)

@app.route('/AdminHome')

def AdminHome():

return render\_template("AdminApp/AdminHome.html")

@app.route('/Upload')

def Upload():

return render\_template("AdminApp/Upload.html")

UPLOAD\_FOLDER = './uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)

global data,filepath

@app.route('/UploadAction', methods=['POST'])

def UploadAction():

global data,filepath

if 'dataset' not in request.files:

return "No file part"

file = request.files['dataset']

filepath = os.path.join(app.config['UPLOAD\_FOLDER'], file.filename)

file.save(filepath)

data = pd.read\_csv(filepath)

columns = data.columns.tolist()

rows = data.head().values.tolist()

return render\_template('AdminApp/ViewDataset.html', columns=columns, rows=rows)

global data, X\_train, X\_test, y\_train, y\_test

@app.route('/preprocess')

def preprocess():

global data, X\_train, X\_test, y\_train, y\_test,data,X,y

data = pd.read\_csv("Dataset/sentimentdataset.csv")

data.dropna(inplace=True)

data = data.loc[:, ~data.columns.str.contains('^Unnamed')]

# Initialize the TF-IDF Vectorizer

tfidf = TfidfVectorizer(max\_features=5000, stop\_words='english')

# Apply TF-IDF to the tweet column

X\_tfidf = tfidf.fit\_transform(data['Text'].astype(str))

X = data['Text']

y = data['Sentiment']

# Step 3: Encode labels

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(y)

# Step 4: TF-IDF Vectorization

vectorizer = TfidfVectorizer(max\_features=5000)

X\_tfidf = vectorizer.fit\_transform(X).toarray()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_tfidf, y\_encoded, test\_size=0.2, random\_state=42)

with open('tfidf\_vectorizer.pkl', 'wb') as f:

pickle.dump(vectorizer, f)

with open('label\_encoder\_fast.pkl', 'wb') as f:

pickle.dump(label\_encoder, f)

return render\_template('AdminApp/SplitStatus.html', total=len(X), train=len(X\_train),test=len(X\_test))

@app.route("/Sentiment")

def Sentiment():

global data

sentiment\_count = data['Sentiment'].value\_counts()

labels = sentiment\_count.index

sizes = sentiment\_count.values

colors = ['#FFA500', '#4169E1', '#778899']

# Create pie chart

plt.figure(figsize=(6, 6))

plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140)

plt.title('Sentiment Analysis')

plt.axis('equal')

plt.savefig("Static/pie.png")

plt.close()

return render\_template('AdminApp/Graph.html', filename="pie.png")

@app.route("/PlotformSentiment")

def PlotformSentiment():

global data

data['Sentiment'] = data['Sentiment'].str.lower().str.strip()

data['Platform'] = data['Platform'].str.title().str.strip()

# Filter to valid sentiment and platform values

valid\_sentiments = ['positive', 'negative', 'neutral']

valid\_platforms = ['Twitter', 'Facebook', 'Instagram']

data = data[data['Sentiment'].isin(valid\_sentiments) & data['Platform'].isin(valid\_platforms)]

# Create a pivot table for plotting

pivot = data.pivot\_table(index='Platform', columns='Sentiment', aggfunc='size', fill\_value=0)

# Plot grouped bar chart

pivot = pivot[valid\_sentiments] # Ensure correct order

pivot.plot(kind='bar', figsize=(10, 6), color=['green', 'red', 'gray'])

plt.title('Sentiment Analysis Across Platforms')

plt.xlabel('Social Media Platform')

plt.ylabel('Number of Posts')

plt.xticks(rotation=0)

plt.legend(title='Sentiment')

plt.savefig("Static/Bar.png")

plt.tight\_layout()

plt.close()

return render\_template('AdminApp/Graph.html', filename="Bar.png")

@app.route("/Likes")

def Likes():

global data

# Group by platform and sum likes/retweets

grouped = data.groupby('Platform')[['Likes', 'Retweets']].sum().loc[['Twitter', 'Facebook', 'Instagram']]

# Plot line graph

plt.figure(figsize=(10, 6))

plt.plot(grouped.index, grouped['Likes'], marker='o', label='Likes', color='blue')

plt.plot(grouped.index, grouped['Retweets'], marker='o', label='Retweets', color='orange')

plt.title('Likes and Retweets Across Platforms')

plt.xlabel('Platform')

plt.ylabel('Count')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.savefig("Static/LinePlot.png")

plt.close()

return render\_template('AdminApp/Graph.html', filename="LinePlot.png")

global gbc\_acc,rf\_acc

@app.route('/gbc')

def gbc():

global gbc\_acc,rf\_acc

gb\_model = GradientBoostingClassifier()

gb\_model.fit(X\_train, y\_train)

pred = gb\_model.predict(X\_test)

gbc\_acc = accuracy\_score(y\_test, pred) \* 100

#RandomForest Algorithm

rm = RandomForestClassifier()

rm.fit(X\_train, y\_train)

rf\_pred = rm.predict(X\_test)

rf\_acc = accuracy\_score(y\_test, rf\_pred) \* 100

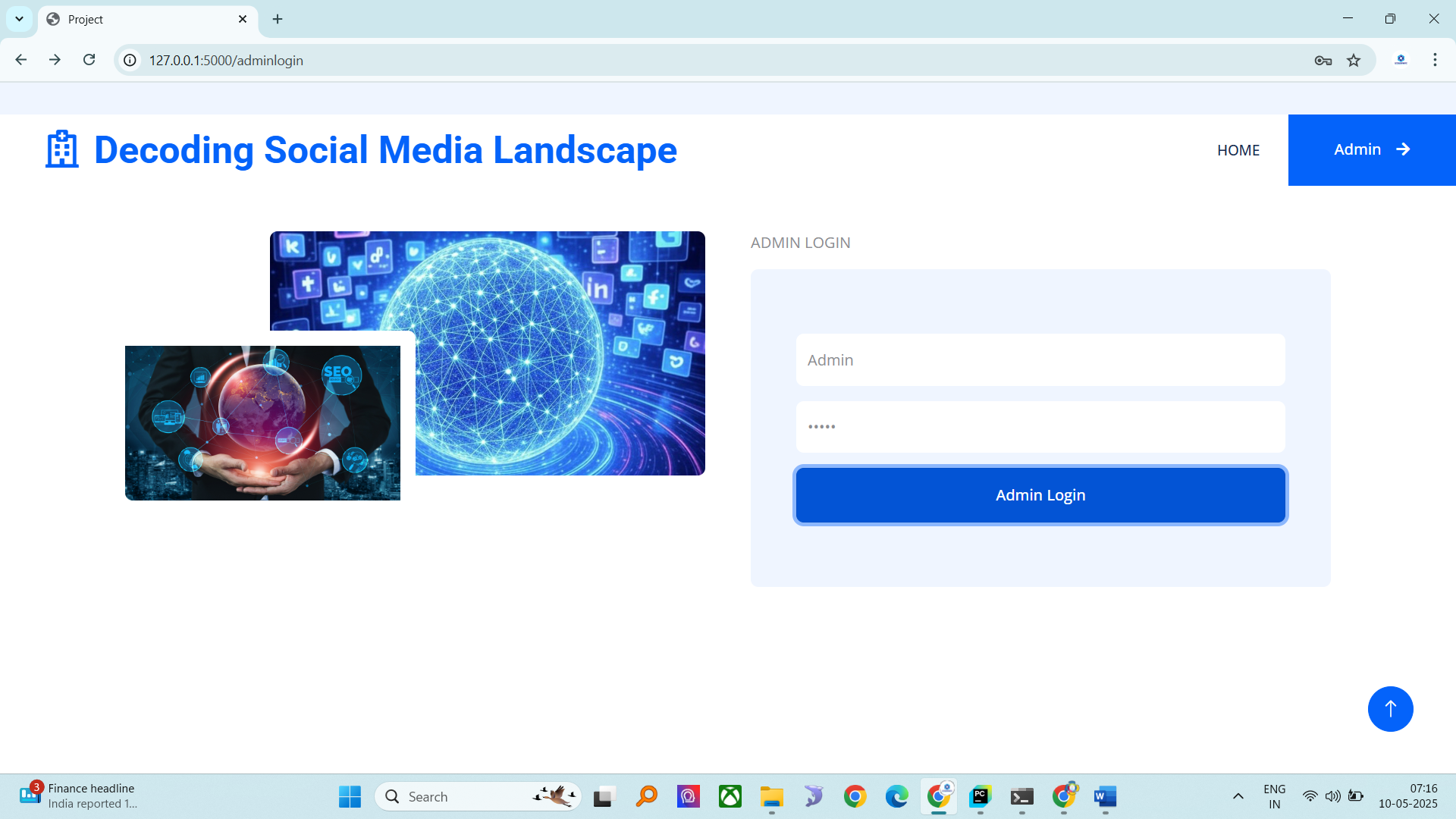
return render\_template('AdminApp/AlgorithmStatus.html', msg="Algorithm Model Generated Successfully..!!")

**SCREENSHOTS**

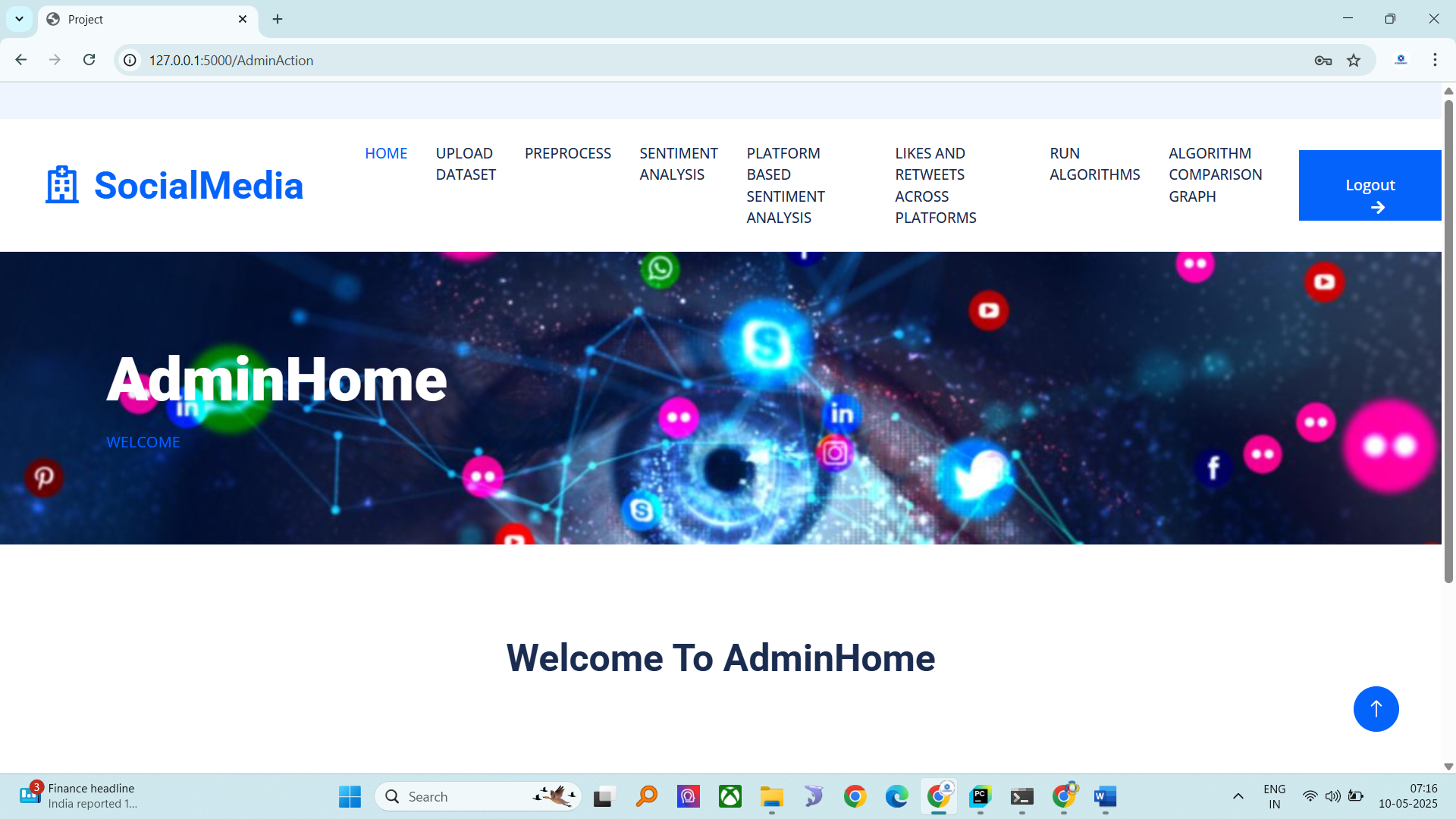
INDEX PAGE



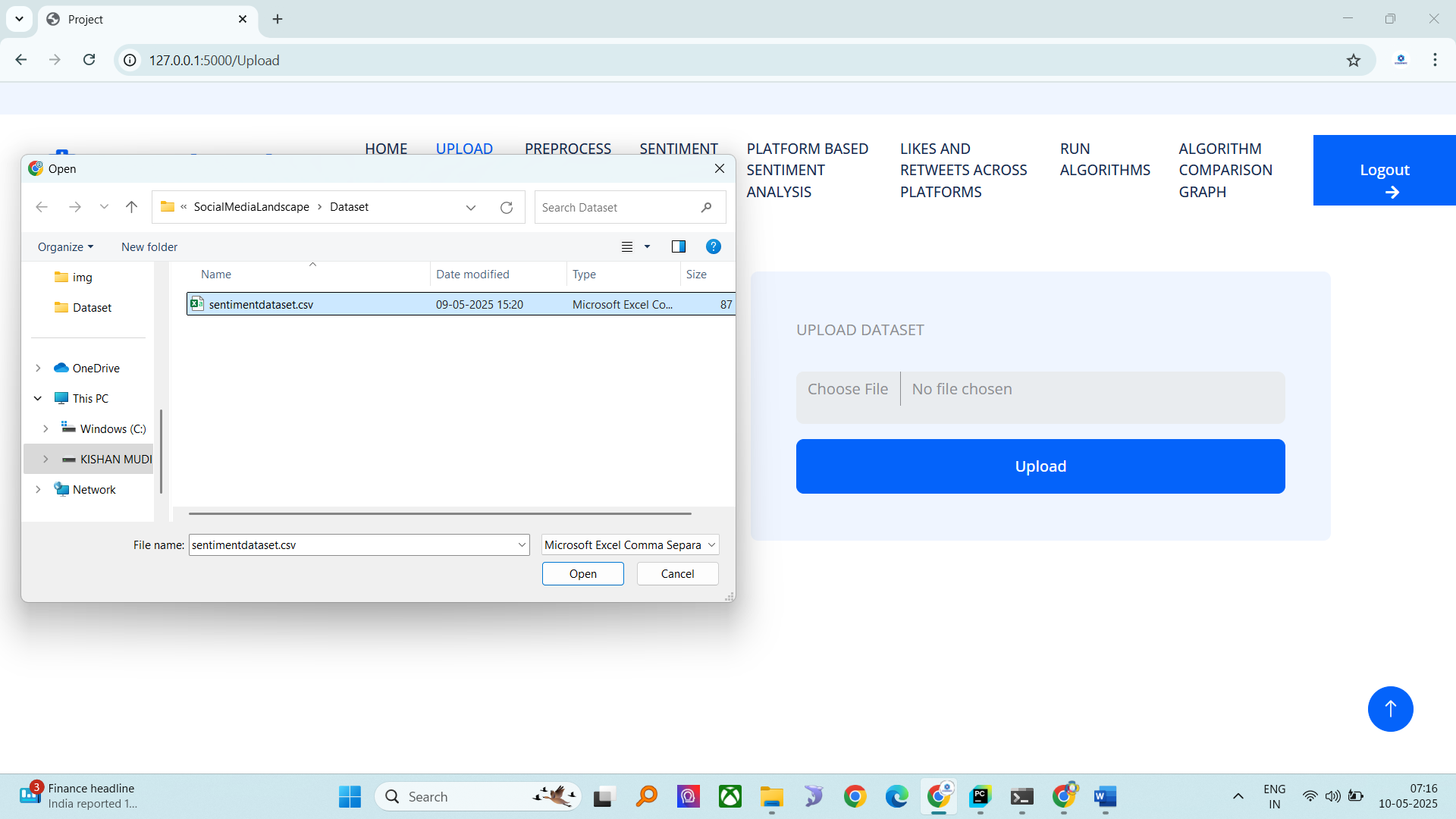
Admin login

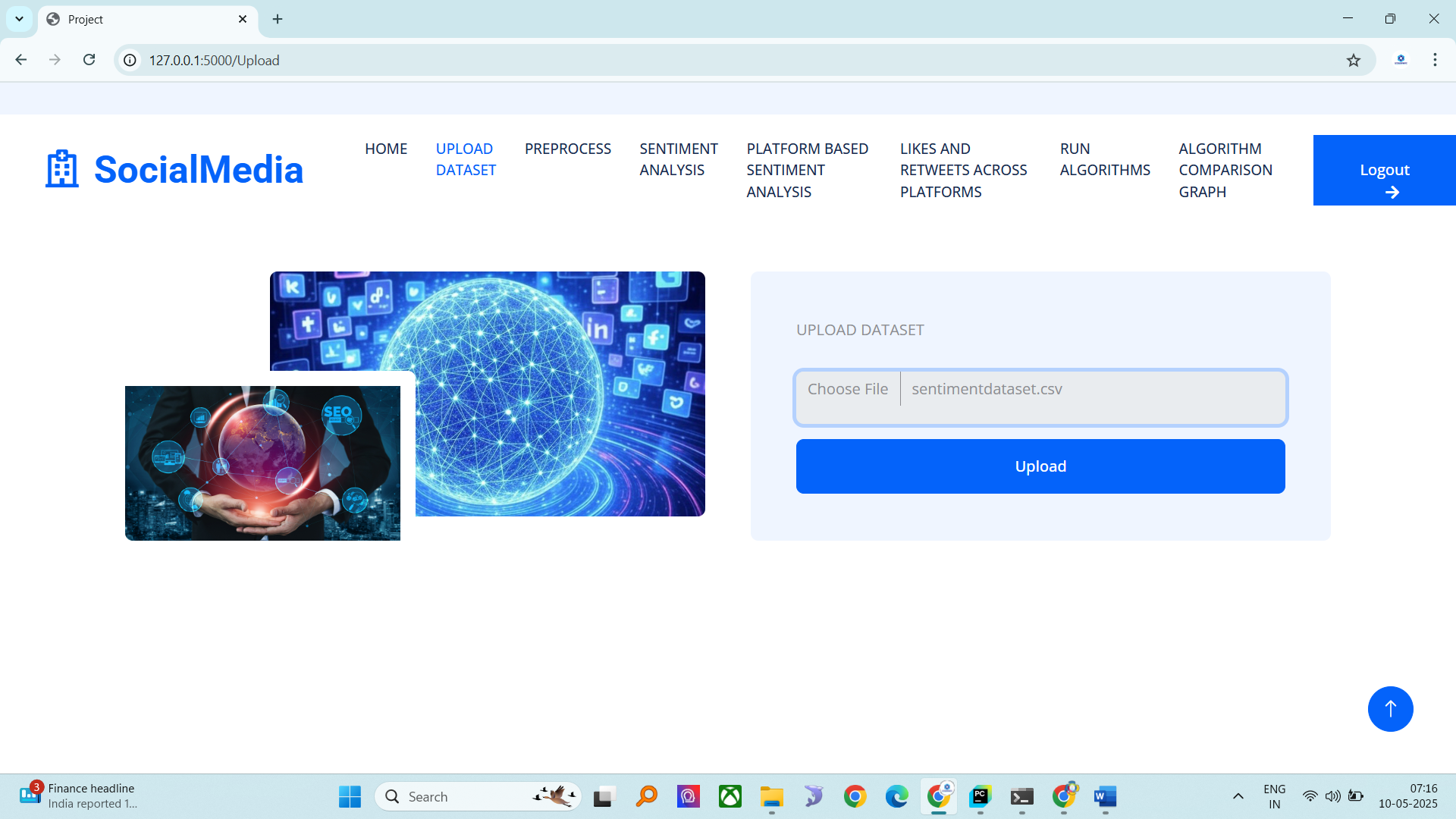


Admin home

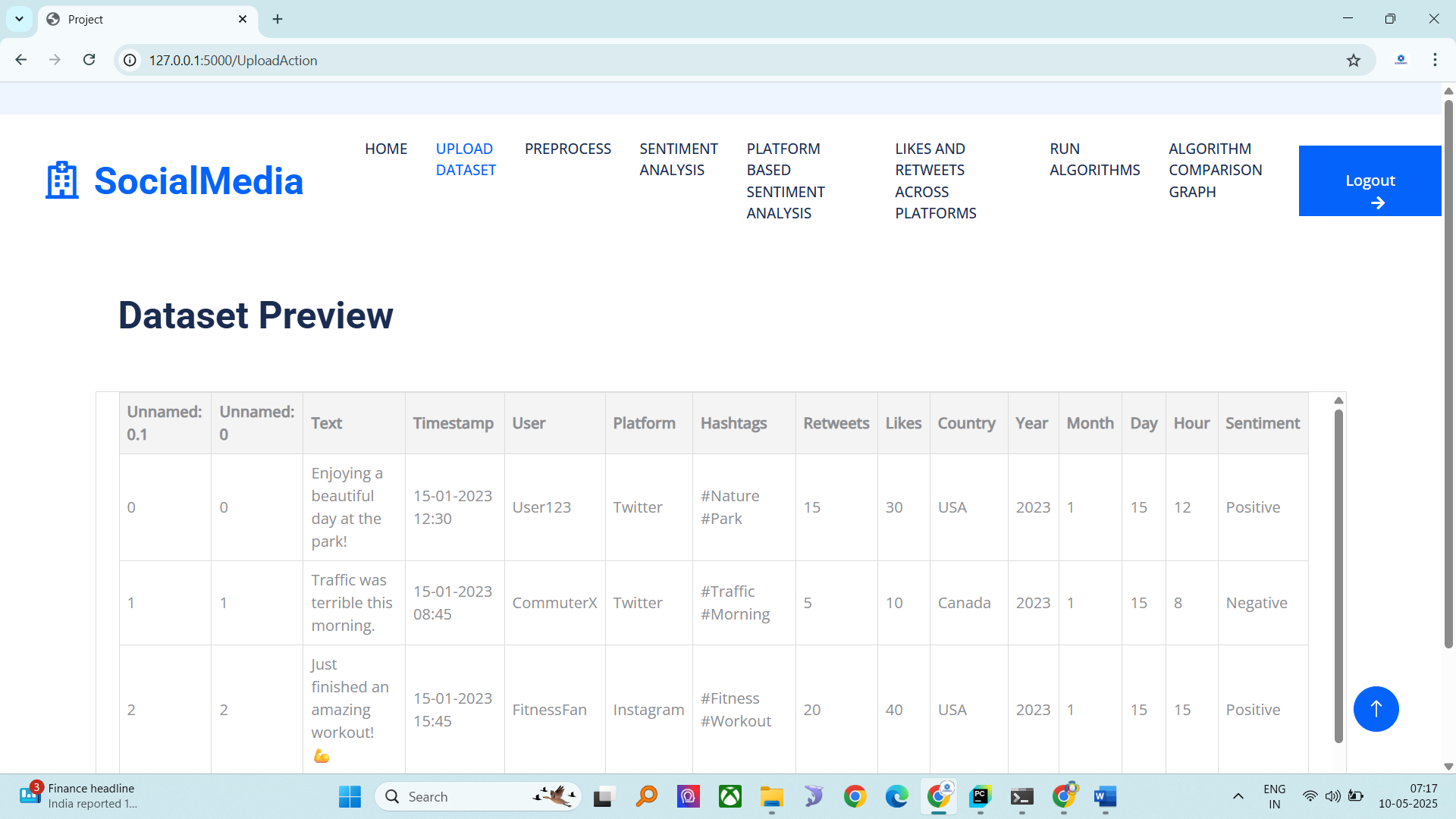


Upload dataset

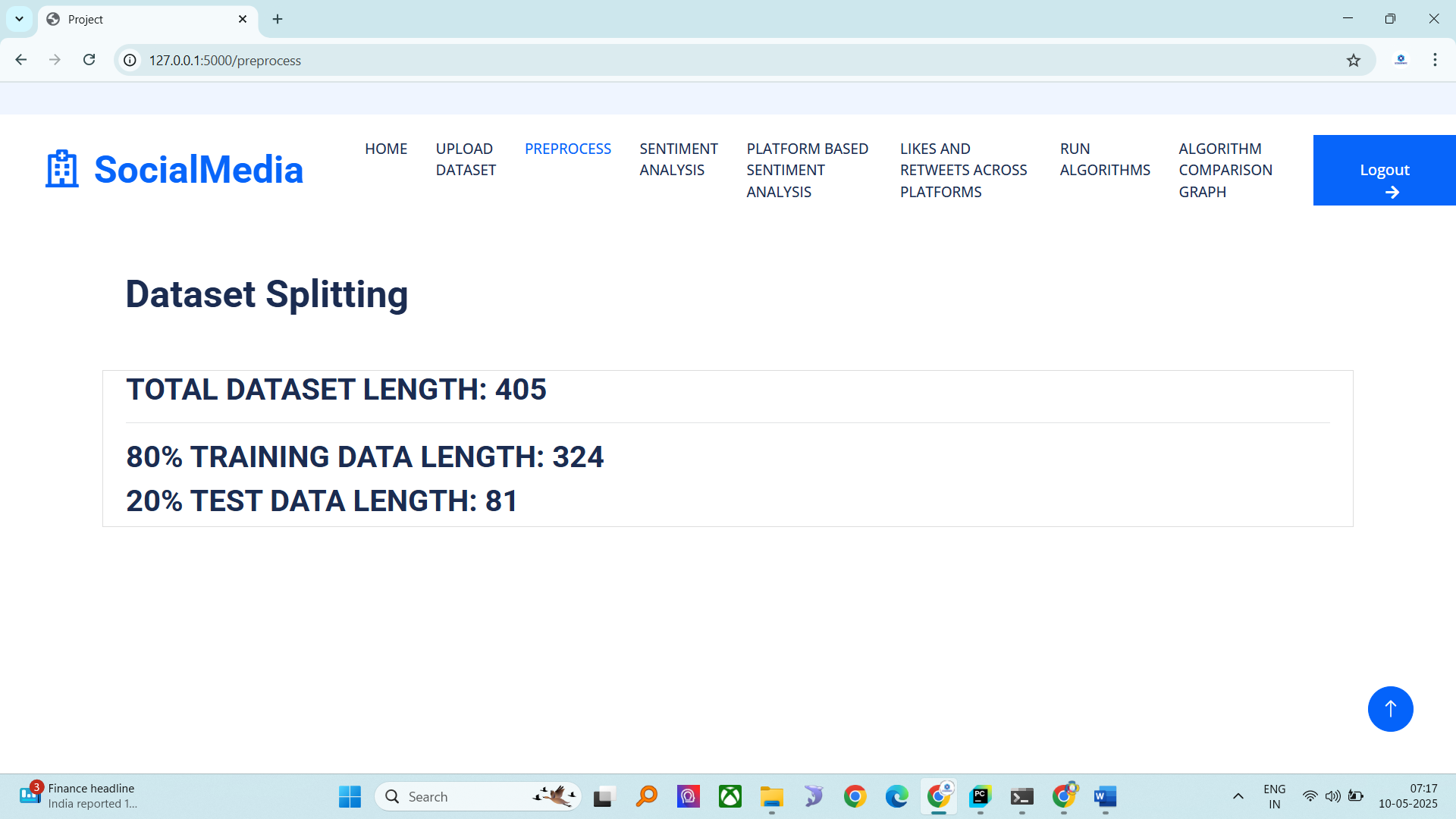




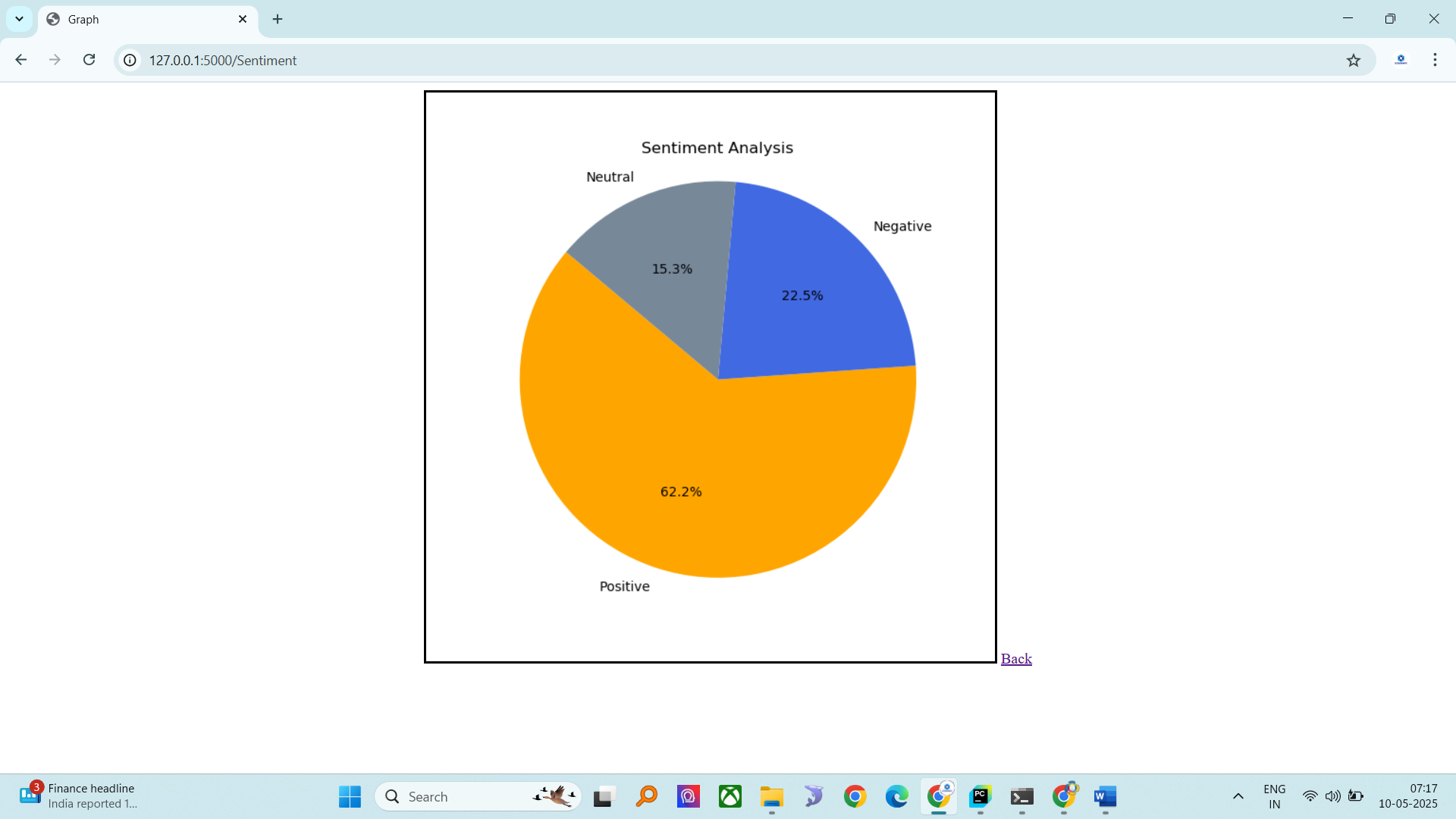
Dataset view



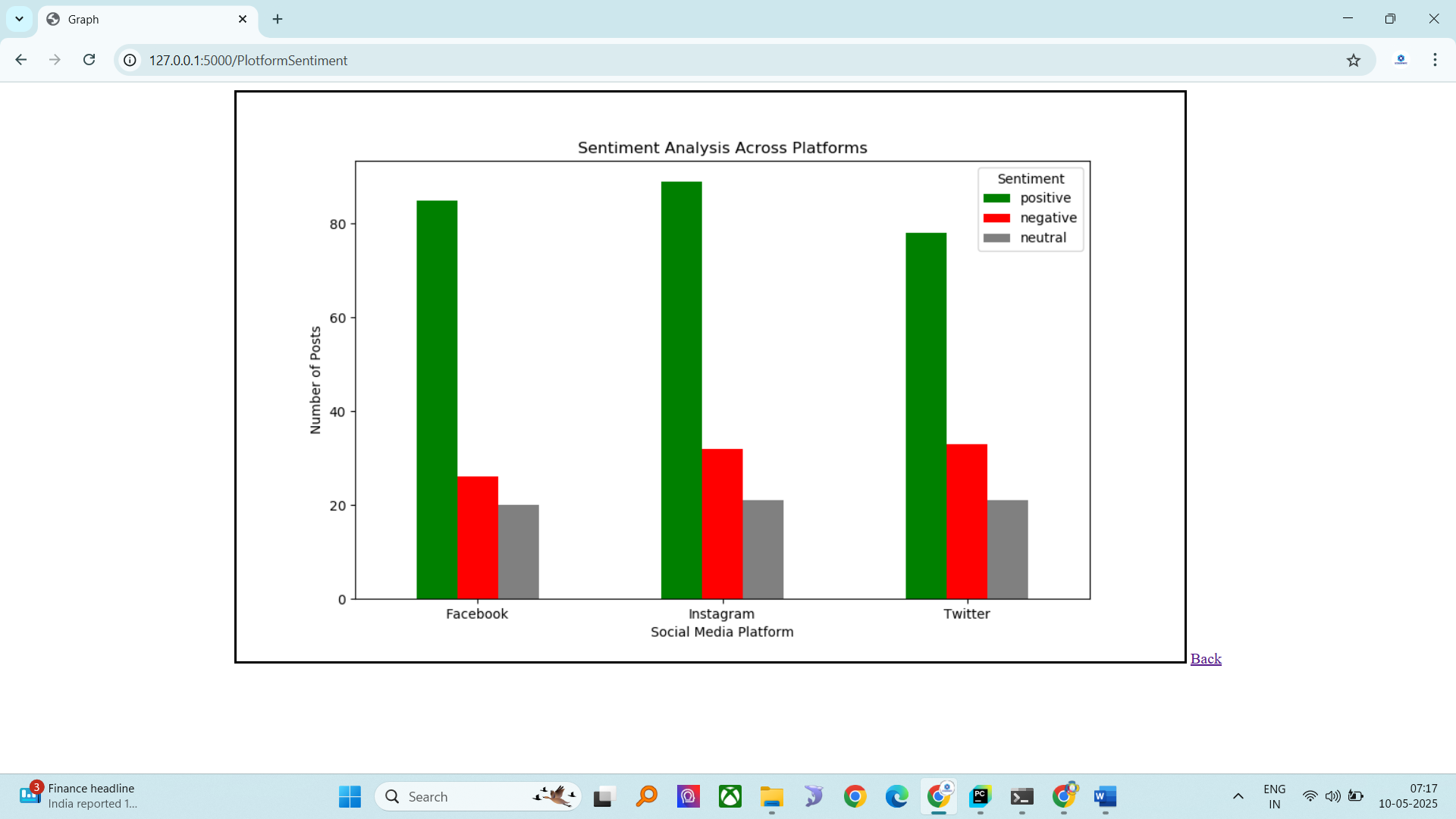
Preprocess



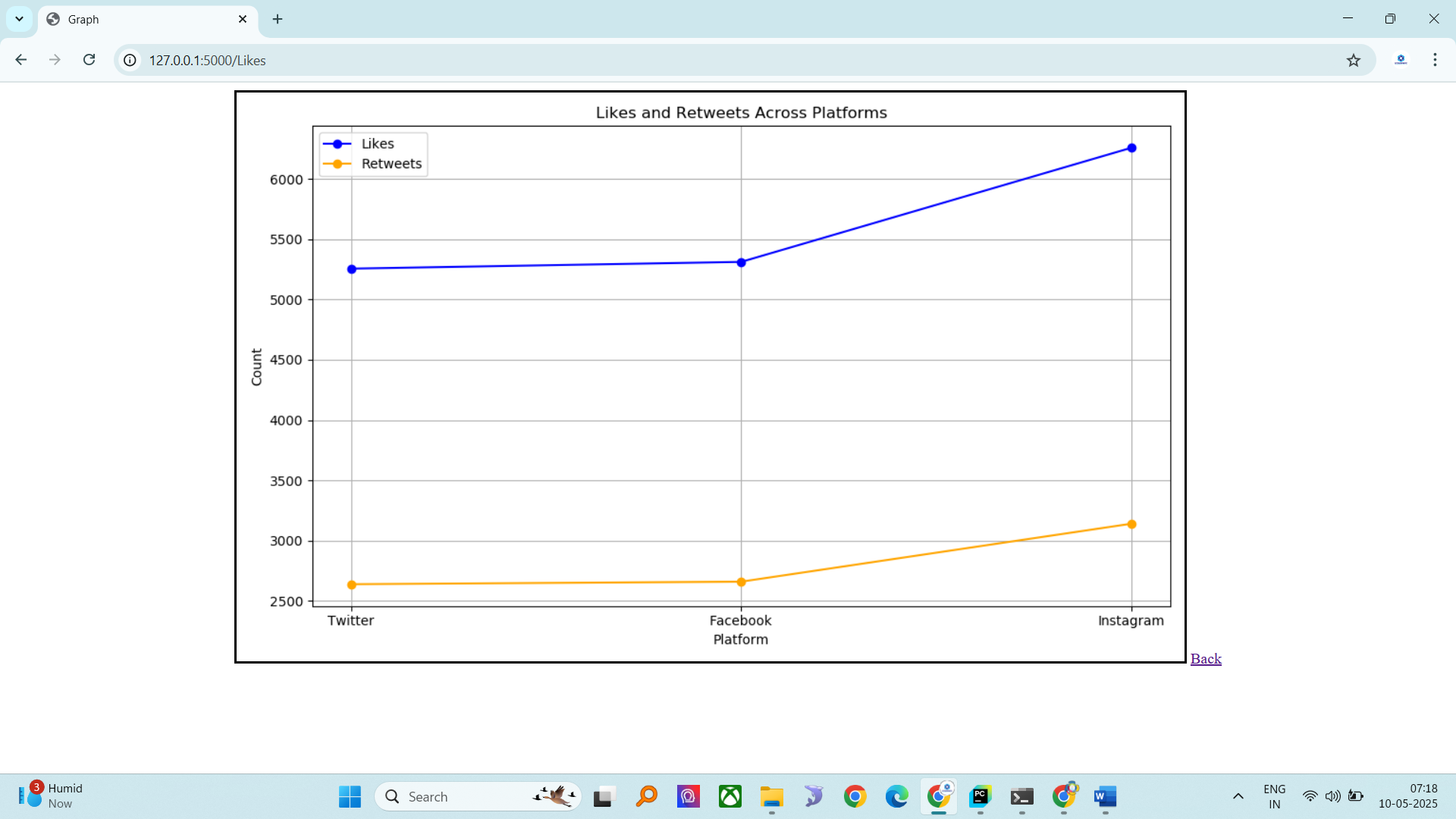
Pie chart on sentiment



Platform based sentiment graph



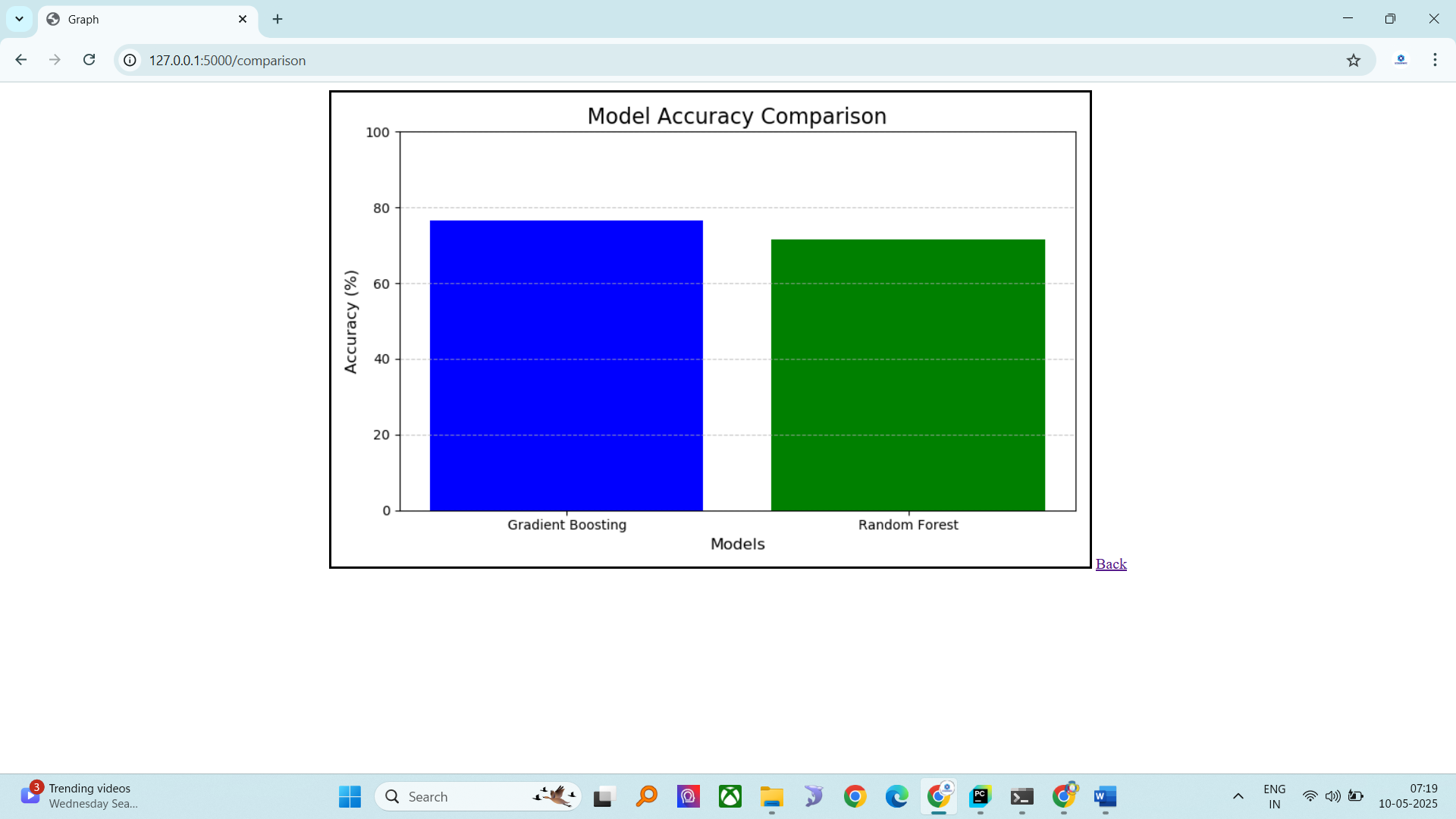
Likes and retweets graph



Algorithms running and model generating



Algorithm model accuracy graph



**CONCLUSION**

This project demonstrates the effectiveness of machine learning in understanding and analyzing user sentiments and engagement on social media platforms. By integrating advanced preprocessing techniques and ensemble algorithms, the system offers accurate sentiment classification and insightful visualizations. The comparative analysis of models provides a clear understanding of performance metrics, enabling informed decision-making. The system’s modularity and scalability make it a robust tool for sentiment analysis in a variety of contexts.

**FUTURE SCOPE**

Future enhancements can include deep learning models such as BERT or LSTM for improved sentiment understanding, real-time sentiment tracking, multilingual sentiment analysis, and integration with social media APIs for live data streaming. Additional modules could also be developed for topic modeling, influencer analysis, and emotion detection.

**REFERENCES**

1. Sharma, R., & Das, A. (2023). "Sentiment Analysis on Twitter Data using Ensemble Learning." *IEEE Access*, vol. 11, pp. 45329–45337.
2. Liu, M., & Wang, Z. (2023). "A Comparative Study on Text Vectorization Methods for Sentiment Classification." *IEEE Transactions on Knowledge and Data Engineering*.
3. Verma, K., & Singh, D. (2024). "Hybrid Machine Learning Approach for Social Media Sentiment Analysis." *IEEE Internet Computing*.
4. Chen, X., & Huang, J. (2023). "Evaluating TF-IDF and Word Embeddings in Social Sentiment Analysis." *IEEE Transactions on Computational Social Systems*.
5. Narayanan, S., & Banerjee, S. (2024). "Real-Time Social Media Monitoring Using Ensemble Models." *IEEE Systems Journal*.
6. Park, Y., & Lim, H. (2023). "A Visualization Framework for Sentiment Trends on Social Media." *IEEE Transactions on Visualization and Computer Graphics*.
7. Zhao, L., & Choi, K. (2023). "Gradient Boosting Machines for Short Text Sentiment Classification." *IEEE Access*.
8. Alvi, F., & Ahmad, M. (2024). "Enhancing Social Media Analysis with TF-IDF and Random Forest." *IEEE Transactions on Artificial Intelligence*.
9. Rahman, M., & Kim, Y. (2023). "Cross-Platform Sentiment Analysis Using Machine Learning Models." *IEEE Transactions on Emerging Topics in Computing*.
10. Patel, J., & Rao, P. (2024). "An Intelligent System for Trend Prediction Using Sentiment and Engagement Data." *IEEE Transactions on Industrial Informatics*