CHAT ANALYZER USING PYTHON LIBRARIES

UNIVERSITY OF ENGINEERING
&
MANAGEMENT, JAIPUR

CHAT ANALYZER USING PYTHON LIBRARIES

Submitted in the partial fulfillment of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Under

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

BY

AMAN

UNDER THE GUIDANCE OF

PROF. DEBAJYOTI CHATTERJEE

COMPUTER SCIENCE & ENGINEERING

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Approval Certificate

This is to certify that the project report entitled "**CHAT ANALYZER USING PYTHON LIBRARIES**" submitted by **Aman** in partial fulfillment of the requirements of the degree of **Bachelor of Technology** in **Computer Science & Engineering** from University **of Engineering and Management, Jaipur** was carried out in a systematic and procedural manner to the best of our knowledge. It is a bona fide work of the candidate and was carried out under our supervision and guidance during the academic session of 2020-2023.

Prof. Debajyoti Chatterjee
Project Guide, Assistant Professor (CSE)
UEM, JAIPUR

Prof. (Dr.) Mrinal Kanti Sarkar
HoD, Dept. of CSE, UEM Jaipur

Prof. (Dr.) A Mukherjee
Dean Academics, UEM, JAIPUR

ACKNOWLEDGEMENT

The endless thanks goes to Lord Almighty for all the blessings he has showered onto me, which has enabled me to write this last note in my research work. During the period of my research, as in the rest of my life, I have been blessed by Almighty with some extraordinary people who have spun a web of support around me. Words can never be enough in expressing how grateful I am to those incredible people in my life who made this thesis possible. I would like an attempt to thank them for making my time during my research in the Institute a period I will treasure. I am deeply indebted to my research supervisor, Professor Debajyoti Chatterjee for giving me such an interesting thesis topic. Each meeting with him added in valuable aspects to the implementation and broadened my perspective. He has guided me with his invaluable suggestions, lightened up the way in my darkest times and encouraged me a lot in the academic life.

AMAN

ABSTRACT

Chat analyzers are tools that are used to examine data from several chat services, including Facebook Messenger, WhatsApp, and Telegram. By giving data on the frequency and substance of messages, the emotional tone of the discussion, and the most often used terms and phrases, these solutions enable users to get insights into their interactions. The fact that chat analyzers may offer useful information for private or professional needs is one of their main advantages. Businesses may employ chat analyzers, for instance, to examine consumer discussions and spot opportunities to enhance their goods or services. To analyze chats with friends and family and learn more about their communication styles, people can also utilise chat analyzers. A common programming language for creating chat analyzers is Python. Natural language processing (NLP) activities including text preparation, sentiment analysis, and topic modelling are simple to complete using Python's many packages. By using these libraries, programmers may design chat analyzers that can precisely analyse and understand message content. Chat analyzers are useful tools for learning more about messaging discussions in general. Python-based chat analyzers have a number of benefits, including NLP functionality and access to a large number of libraries and tools. The need for chat analyzers is anticipated to rise as messaging programmes continue to gain popularity, and Python is likely to continue to be a popular choice for creating these tools.

Table of Contents

Chapter 1

INTRODUCTION

A chat analyzer is a piece of software or equipment that is used to review discussions between people, either after the fact or in real time. It is a potent tool that may be applied to a variety of industries, including social media, business, education, healthcare, and law enforcement. The chat analyzer tools analyze the chat logs using natural language processing (NLP) techniques to reveal information about the interaction. These data may be applied to optimize hospital outcomes, monitor student development, measure staff efficiency, deter crime, increase user engagement, and provide better customer service. A crucial tool in today's fast-paced society where effective communication is essential to success is the chat analyzer. It has completely changed how conversations are analyzed, offering insightful data that can be utilized to enhance a variety of aspects of our lives.

1.1 What is Chat Analyzer?

A chat analyzer is a piece of software made specifically for examining conversations that happen on messaging platforms like WhatsApp, Facebook Messenger, and Telegram. A chat analyzer's main objective is to give users insights about their chats, including the frequency and substance of messages, the conversation's emotional tone, and the most often used terms and phrases.

now are many different kinds of chat analyzers out now, from basic programs that offer statistics about a discussion to more sophisticated tools that do sentiment analysis, topic modelling, and other intricate tasks using natural language processing (NLP) methods. While some chat analyzers are created for personal use, others are made for professional purposes, such as customer conversation analysis.

A chat analyzer is a piece of software made specifically for examining conversations that happen on messaging platforms like WhatsApp, Facebook Messenger, and Telegram. A chat analyzer's main objective is to give users insights about their chats, including the frequency and substance of messages, the conversation's emotional tone, and the most often used terms and phrases.

A chat analyzer is a piece of software made specifically for examining conversations that happen on messaging platforms like WhatsApp, Facebook Messenger, and Telegram. A chat analyzer's main objective is to give users insights about their chats, including the frequency and substance of messages, the conversation's emotional tone, and the most often used terms and phrases.

now are many different kinds of chat analyzers out now, from basic programs that offer statistics about a discussion to more sophisticated tools that do sentiment analysis, topic modelling, and other intricate tasks using natural language processing (NLP) methods. While some chat analyzers are created for personal use, others are made for professional purposes, such as customer conversation analysis.

A chat analyzer is a piece of software made specifically for examining conversations that happen on messaging platforms like WhatsApp, Facebook Messenger, and Telegram. A chat analyzer's main objective is to give users insights about their chats, including the frequency and substance of messages, the conversation's emotional tone, and the most often used terms and phrases.

now are many different kinds of chat analyzers out now, from basic programs that offer statistics about a discussion to more sophisticated tools that do sentiment analysis, topic modelling, and other intricate tasks using natural language processing (NLP) methods. While some chat analyzers are created for personal use, others are made for professional purposes, such as customer conversation analysis.

Software tools known as chat analyzers offer a range of services for examining conversations that take place on messaging platforms. These tools are designed to assist users in understanding their communication styles, enhancing their communication abilities, and maximizing message discussions for either personal or professional usage. The following are some of the services that chat analyzers offer:

Statistics about a discussion may be obtained through chat analyzers, including the quantity of messages exchanged, the average message size, and the busiest times of day for communicating. Users may utilize this information to better understand their communication habits and spot opportunities for improvement.

The most frequently used words and phrases in a discussion may be tracked by chat analyzers, which can reveal information about the conversation's key subjects and themes. This data can be helpful for spotting significant patterns and subjects as well as for optimizing messaging interactions for private or professional usage.

Sentiment analysis by doing sentiment analysis on the text of the messages, chat analyzers can determine the emotional tone of a discussion. Users may utilize this to better comprehend the conversation's general tone and see any places where they might need to modify their communication style to better suit the conversation's emotional undertone.

Text preprocessing: Chat analyzers are capable of deleting stop words, stemming sentences, and lemmatizing the text of chats. These actions can assist in text cleaning and make it simpler for other NLP approaches to analyze.

Chat analyzers can use topic modelling to extract the primary themes and subjects of a discussion from the text of messages. This data can be helpful for spotting significant patterns and subjects as well as for optimizing messaging interactions for private or professional usage.

1.2 Why we need Chat Analyzer?

Chat analyzers have the ability to determine the language being used in a chat, which is helpful for analyzing conversations taking place in different languages.

Chat analyzers can assist users filter out undesirable communications so they can concentrate on the crucial talks by automatically detecting spam messages based on their content.

In general, chat analyzers offer a range of services, from elementary statistics to sophisticated NLP methods, for analyzing message chats. These technologies may be helpful for both personal and professional reasons, such as learning more about communication trends, enhancing customer support, and tailoring message interactions for certain objectives. The need for chat analyzers is projected to rise as messaging programs continue to gain popularity, and new features and services are likely to be created to satisfy this demand.

Chat analyzer refers to software or tools that analyze the conversations between individuals, either in real-time or after the fact. The importance of chat analyzer can be seen in various fields, such as businesses, education, healthcare, and law enforcement. In this article, we will explore the need for chat analyzer in detail.

1. Business:

In the business world, chat analyzer is essential for companies that want to enhance their customer service. Chat analyzer tools can help businesses to monitor customer interactions with their customer support representatives, identify customer pain points and provide relevant solutions. Chat analyzer can also help businesses to track employee productivity by analyzing chat logs between employees and identifying key performance indicators (KPIs).

2. Education:

Chat analyzer is also useful in the education sector, especially in online learning. With the help of chat analyzer tools, teachers can monitor student progress and evaluate the effectiveness of their teaching methods. Chat analyzer can also be used to detect plagiarism in online submissions and identify students who need extra help in certain areas.

3. Healthcare:

Chat analyzer is important in the healthcare sector, where it can be used to monitor patient conversations with healthcare providers. By analyzing chat logs, healthcare providers can identify patient concerns, track symptoms, and improve patient outcomes. Chat analyzer can also help healthcare providers to identify potential fraud or abuse in the healthcare system.

4. Law enforcement:

Chat analyzer is also crucial in law enforcement, where it can be used to monitor criminal activity on social media platforms and messaging apps. Chat analyzer tools can help law enforcement agencies to identify potential threats, track suspects, and prevent criminal activity. Chat analyzer can also be used in investigations to analyze chat logs and identify key evidence.

5. Social media:

Chat analyzer is also important in the world of social media, where it can be used to analyze conversations between users. By analyzing chat logs, social media companies can identify trends, detect fake news and hate speech, and improve user engagement. Chat analyzer can also help social media companies to identify potential security threats and prevent cyber-attacks.

In conclusion, chat analyzer is a crucial tool in various fields. With the help of chat analyzer tools, businesses can improve their customer service, education providers can monitor student progress, healthcare providers can improve patient outcomes, law enforcement agencies can prevent criminal activity, and social media companies can improve user engagement. Chat analyzer can also help to identify potential security threats and prevent cyber-attacks. Therefore, the use of chat analyzer tools is essential in today's fast-paced world where communication is critical to success.

CHAPTER 2

LITERATURE REVIEW

The Chat analyzer is a tool that uses Python programming language to analyze conversations between individuals in various fields, such as business, education, healthcare, law enforcement, and social media. Python is a powerful programming language that is widely used in data analysis, natural language processing (NLP), and machine learning.

One study conducted by researchers at the University of Cambridge analyzed chat logs from an online support group for people with depression. The researchers used Python libraries such as NLTK (Natural Language Toolkit) and scikit-learn to analyze the chat logs and identify common topics of conversation. They found that topics such as medication, therapy, and coping mechanisms were frequently discussed by group members.

Another study conducted by researchers at the University of California, Los Angeles analyzed chat logs from a social networking site. The researchers used Python libraries such as Pandas and Matplotlib to analyze the chat logs and identify patterns in user behavior. They found that users who engaged in more conversations were more likely to have a larger network of friends on the social networking site.

In another study, researchers at the University of Turin used Python and machine learning algorithms to analyze chat logs from an online game. The researchers used Python libraries such as scikit-learn and TensorFlow to analyze the chat logs and identify toxic behavior, such as harassment and hate speech. They found that their machine learning algorithm was able to accurately identify toxic behavior in chat logs, which could be used to improve the safety and inclusivity of online gaming communities.

Python libraries such as NLTK, Pandas, Matplotlib, scikit-learn, and TensorFlow have become essential tools in chat analyzer. These libraries allow researchers and analysts to perform various tasks such as data cleaning, preprocessing, visualization, and machine learning. With the help of Python, chat analyzer has revolutionized the way conversations are analyzed and provided valuable insights that can be used to improve various aspects of our lives.

In conclusion, Python has become an essential tool in chat analyzer, allowing researchers and analysts to analyze chat logs and identify common topics of conversation, patterns in user behavior, and toxic behavior. The studies conducted by researchers demonstrate the effectiveness of Python and its libraries in various fields, including mental health, social networking, and online gaming.

CHAPTER 3

PROPOSED MODEL

3.1 HTML

HTML (Hyper Text Markup Language) is the standard markup language used to create web pages and web applications. It is a language that consists of a series of tags, which are used to define the structure and content of web pages. HTML documents are written in plain text format and can be created using any text editor. The primary purpose of HTML is to define the structure of content on a web page. HTML tags are used to create headings, paragraphs, lists, tables, forms, and other structural elements. These tags define the content of the page and its layout. In addition to defining the structure of content, HTML also provides a way to include multimedia elements such as images, audio, and video. HTML tags can be used to specify the source and properties of these elements. HTML can also be used in conjunction with other technologies to add interactivity and dynamic content to web pages. JavaScript is a common scripting language used to add interactivity to web pages. CSS (Cascading Style Sheets) is used to control the layout and styling of web pages. HTML is an evolving language, with new versions and features being developed over time. The most recent version of HTML is HTML5, which includes many new features such as support for video and audio, canvas elements for graphics, and new form controls.

3.2 CSS

CSS is a style sheet language used to describe the presentation of a document written in HTML or XML. Its primary purpose is to separate the content of a web page from its presentation, allowing developers to control the layout, formatting, and styling of multiple web pages at once. CSS works by selecting HTML elements on a web page and applying styles to those elements using a set of properties and values. CSS provides a wide range of properties that can be used to control the visual appearance of HTML elements, including font size and style, text color and background color, padding and margin, border, and positioning. CSS also provides a range of layout and positioning techniques, including floating and clearing elements, using flexbox and grid, and absolute and relative positioning. One of the key benefits of using CSS is that it allows developers to create responsive web pages that adjust their layout and presentation based on the size of the user's screen. Here are some important concepts in CSS:

Selectors: CSS uses selectors to target HTML elements on a web page. There are several types of selectors, including element selectors, class selectors, and ID selectors.

Properties: CSS properties are used to control the presentation of HTML elements, such as their color, size, font, and spacing.

Values: CSS properties have values that define how they should be applied to HTML elements. For example, the color property can be set to a value like red, blue, or #00ff00 (which is a hexadecimal color value).

Box model: CSS uses a box model to describe how elements are laid out on a web page. The box model consists of the content area, padding, border, and margin of an element.

Cascading: CSS is designed to be cascading, which means that multiple styles can be applied to an element, and the most specific style will take precedence. This allows developers to define global styles that apply to all elements, as well as specific styles that apply to individual elements.

3.3 JAVASCRIPT

JavaScript is a programming language used to create dynamic and interactive web pages. It is a client-side language, meaning that it is executed on the user's browser rather than on the web server. JavaScript is used for a wide range of tasks on the web, including form validation, creating interactive user interfaces, making asynchronous requests to web servers, and animating elements on a web page. JavaScript syntax is like other C-style programming languages, and it includes features like variables, functions, conditional statements, loops, and object-oriented programming constructs.

One of the key strengths of JavaScript is its ability to manipulate the Document Object Model (DOM) of a web page. The DOM is a hierarchical representation of the elements on a web page, and JavaScript can be used to dynamically modify the contents, styling, and behaviour of these elements. In addition to traditional JavaScript, there are also several libraries and frameworks that have been developed to simplify web development with JavaScript. Examples of these include jQuery, React, Angular, and Vue. JavaScript has become an essential tool for modern web development, and it is widely used in both front-end and back-end development. With its ability to add interactivity and dynamic behaviour to web pages, JavaScript is a critical tool for creating engaging and responsive user experiences on the web.

3.4 PYTHON

Python is a popular high-level programming language known for its simplicity, readability, and versatility. It is widely used for various applications such as web development, data analysis, machine learning, and artificial intelligence. Python can also be used in chat analyzer for various tasks such as data cleaning, data preprocessing, natural language processing, and data visualization.

One of the main uses of Python in chat analyzer is natural language processing (NLP). Chat logs contain text data that can be analyzed using various NLP techniques such as sentiment analysis, named entity recognition, and topic modeling. Python provides various NLP libraries such as NLTK, spaCy, and TextBlob that can be used for these tasks. For example, Python can be used to perform sentiment analysis on chat logs to determine the overall tone of the conversation.

Another use of Python in chat analyzer is data cleaning and preprocessing. Chat logs can be messy and contain various types of errors and inconsistencies. Python provides various libraries such as Pandas and NumPy that can be used for data cleaning, handling missing values, and formatting issues. For example, Python can be used to remove duplicates and correct spelling errors in chat logs.

Python can also be used for data visualization in chat analyzer. Chat logs can be visualized to provide insights into user behavior, such as the frequency of messages, the time of day when messages are sent, and the length of messages. Python provides various data visualization libraries such as Matplotlib, Seaborn, and Plotly that can be used for these tasks. For example, Python can be used to create a line chart to visualize the frequency of messages over time.

In conclusion, Python is a versatile programming language that can be used in chat analyzer for various tasks such as data cleaning, data preprocessing, NLP, and data visualization. The use of Python in chat analyzer can help to extract insights and patterns from chat logs and provide valuable information for various applications such as business, healthcare, and social media.

3.5 PANDAS

Pandas is a popular open-source data analysis library for Python. It provides data structures and functions for manipulating and analyzing numerical tables and time series data. Pandas can be used in chat analyzer for various tasks such as data cleaning, data preprocessing, and data visualization.

One of the main uses of Pandas in chat analyzer is data cleaning. Chat logs can be messy and contain various types of errors and inconsistencies. Pandas provides functions for identifying and handling missing data, outliers, duplicates, and formatting issues. For example, Pandas can be used to remove duplicate messages, correct spelling errors, and convert timestamps to a standardized format.

Another use of Pandas in chat analyzer is data preprocessing. Chat logs can be preprocessed to extract relevant information, such as sentiment analysis, topic modeling, and keyword extraction. Pandas provides functions for tokenizing text, stemming and lemmatization, and converting text to a numerical representation. For example, Pandas can be used to preprocess chat logs and extract the most commonly used words and phrases.

Pandas can also be used for data visualization in chat analyzer. Chat logs can be visualized to provide insights into user behavior, such as the frequency of messages, the time of day when messages are sent, and the length of messages. Pandas provides functions for creating various types of plots, such as line plots, scatter plots, and bar charts. For example, Pandas can be used to visualize the frequency of messages over time or the distribution of message length.

3.6 REGULAR EXPRESSION

Regular expression (regex) is a sequence of characters that defines a search pattern. It is a powerful tool for text processing and pattern matching. Regular expressions can be used in Python to search, replace, and manipulate text data.

In Python, the regular expression module is called "re". The "re" module provides various functions for working with regular expressions. Here are some common use cases of regular expressions in Python:

Searching for a pattern: Regular expressions can be used to search for a specific pattern in a string. For example, the following code searches for all the occurrences of the word "chat" in a string:

3.7 STREAMLIT

Streamlit is a Python library for creating interactive web applications with simple Python scripts. It allows developers to easily create reactive web applications that update in real-time as the user interacts with them. Streamlit provides built-in widgets for creating user interfaces and a wide range of data visualization tools. It is particularly useful for creating data-driven web applications for data visualization, machine learning, and natural language processing. Streamlit is popular among data scientists and machine learning engineers due to its simplicity and ease of use.

3.8 PREPROCESSOR

Preprocessing libraries in Python are used to perform data preprocessing tasks on raw data before using it for machine learning or data analysis tasks. Data preprocessing is an essential step in any data analysis or machine learning project, as it helps to clean, transform, and normalize data to make it more suitable for modeling.

Some popular Python preprocessing libraries are:

NumPy: NumPy is a popular library for numerical computing in Python. It provides powerful data structures for working with arrays, matrices, and multidimensional data. NumPy is often used for data cleaning and transformation tasks, such as filtering, sorting, and reshaping data.

Scikit-learn: Scikit-learn is a popular machine learning library in Python. It provides a wide range of algorithms and tools for machine learning tasks, including data preprocessing. Scikit-learn is often used for data cleaning and transformation tasks, such as scaling data, feature selection, and dimensionality reduction.

NLTK: NLTK (Natural Language Toolkit) is a library for natural language processing in Python. It provides a wide range of tools for working with text data, such as tokenization, stemming, and part-of-speech tagging. NLTK is often used for text preprocessing tasks, such as cleaning text, removing stop words, and converting text to lowercase.

3.9 HELPER

Helper libraries in Python are pre-built modules or packages that contain functions or utilities to simplify coding tasks. These libraries are designed to provide specific functionalities, such as working with files, manipulating data structures, handling dates and times, or interacting with APIs. Popular Python helper libraries include Requests for making HTTP requests, Os for interacting with the operating system, DateTime for working with dates and times, and Math for mathematical functions Languages.

3.10 MATPLOTLIB

Matplotlib is a popular data visualization library in Python that provides a wide range of tools for creating various types of visualizations, such as line plots, scatter plots, bar charts, histograms, and more. It is a comprehensive library that supports a variety of output formats, including PDF, SVG, and PNG, and can be used in various Python environments, including Jupyter notebooks, web applications, and desktop applications.

Matplotlib provides a set of customizable plotting functions that allow users to create high-quality visualizations with minimal code. It also supports customization of plots, including axis labels, titles, legends, color maps, and more. In addition to basic plotting functionality, Matplotlib also provides advanced features, such as subplots, animations, and 3D plots.

Overall, Matplotlib is a powerful data visualization library that is widely used in data science, machine learning, and scientific computing applications. It is an essential tool for anyone who needs to explore and visualize data in Python.

In chat analyzer projects, Matplotlib can be used to create visualizations that help in analyzing chat data. For example, line charts can be used to show the trend of message frequency over time, while bar charts can be used to display the number of messages sent by each user. Matplotlib can also be used to create word clouds, which provide a visual representation of the most common words used in the chat.

Matplotlib is a popular choice in chat analyzer projects due to its ease of use, flexibility, and compatibility with other Python libraries. It can be easily integrated into Python code, and its extensive documentation and large community make it a powerful tool for data visualization.

3.11 SEABORN

Seaborn is a data visualization library for Python that is built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn offers several types of visualizations, including heatmaps, time series, categorical plots, distribution plots, and regression plots.

Seaborn is designed to work well with data frames from Pandas, making it easy to visualize data in a format that is commonly used in data science workflows. It provides a range of customization options, such as color palettes, themes, and plot styles, allowing users to create visualizations that are tailored to their specific needs.

In addition to its visualization capabilities, Seaborn also includes functions for statistical analysis, such as linear regression, correlation analysis, and hypothesis testing. This makes it a valuable tool for data exploration and modeling.

3.12 URL EXTRACT

The urlextract library in Python is a package that is used for extracting and normalizing URLs (Uniform Resource Locators) from text data. It can be used for tasks such as web scraping, text analysis, and information retrieval.

The urlextract library provides a range of functions for extracting URLs from text data, including regular expressions and heuristics. It also includes functionality for normalizing URLs, which involves converting them into a standardized format that can be used for further analysis.

One of the benefits of using urlextract is that it can handle a wide range of URL formats, including complex URLs with query parameters and fragments. It can also handle URLs with non-standard characters, such as Unicode characters.

3.13 WORD CLOUD

A word cloud is a visual representation of the most common words in a text or collection of texts. In chat analyzer projects, word clouds can be used to identify the most commonly used words in a chat, providing insights into the topics discussed and the overall tone of the conversation.

There are several Python libraries that can be used to create word clouds, including Word Cloud, which is a popular library for generating word clouds from text data. Word Cloud allows users to customize the size, shape, and color of the word cloud, as well as the font and layout of the words.

To create a word cloud in Python using WordCloud, users first need to preprocess the text data to remove stop words, punctuation, and other irrelevant characters. Once the text has been preprocessed, users can use WordCloud to generate the word cloud, which can be displayed in a variety of formats, such as PNG, SVG, or PDF.

3.14 COLLECTIONS

The Collections module in Python provides specialized container datatypes that are alternatives to the built-in datatypes like lists, dictionaries, and tuples. These container datatypes are optimized for specific use cases and can be useful in chat analyzer projects for tasks such as counting the frequency of words in a chat, keeping track of unique users, and more.

One commonly used container datatype from the Collections module is the Counter class, which is used to count the frequency of elements in a list. In chat analyzer projects, the Counter class can be used to count the frequency of words in a chat or the number of messages sent by each user.

Another useful datatype from the Collections module is the defaultdict class, which is a subclass of the built-in dictionary. The defaultdict class allows users to specify a default value for any key that does not exist in the dictionary, which can be useful in chat analyzer projects for handling missing data.

Overall, the Collections module provides several useful container datatypes that can simplify data analysis tasks in chat analyzer projects.

3.15 EMOJI

The Emoji library is a Python library that provides a set of functions for working with emojis in text data. It allows developers to parse and manipulate emoji characters in text strings, including detecting and replacing emojis, counting the frequency of emojis, and more.

In chat analyzer projects, the Emoji library can be used to analyze the use of emojis in a chat. For example, it can be used to count the number of times each emoji appears in the chat or to detect the most commonly used emojis. This can provide insights into the emotional tone of the conversation, as well as the topics and trends discussed in the chat.

The Emoji library is easy to use and can be easily integrated into Python code using the pip package manager. It provides a range of useful functions for working with emojis, including EmojiParser, EmojiCount, and EmojiReplace. Additionally, the library is open-source and actively maintained, with frequent updates and bug fixes.

3.16 DATE TIME

The datetime library in Python provides a set of functions for working with dates and times. It includes various classes and methods for handling dates, times, and time intervals. In chat analyzer projects, the datetime library can be used for analyzing the timing of messages and conversations.

The datetime library provides a datetime class, which is used to represent date and time objects. It includes methods for creating datetime objects from strings, formatting datetime objects as strings, and performing arithmetic with datetime objects. In chat analyzer projects, the datetime library can be used to perform various tasks, such as calculating the time difference between two messages, grouping messages by day, week, or month, and identifying the busiest times of day for chat activity. Additionally, the datetime library can be used to generate time series plots and graphs to visualize chat activity over time.

CHAPTER 4

SOFTWARE DEVELOPMENT LIFE CYCLE

Software development organization follows some process when developing a software product. A key component of any software development process is the life cycle model on which the process is based. The particular life cycle model can significantly affect overall life cycle costs associated with a software product. Life cycle of the software starts from concept exploration and at the retirement of the software.

4.1 PROJECT LIFE CYCLE

The system development life cycle is classically thought of as the set of activities that E-Canteen Management System project report analysts, designers and users carry out to develop and implement an information system. The system development life cycle consists of the following activities:

1. Preliminary investigation.

2. Requirement Analysis.

3. System Designing.

4. Coding.

5. System Testing.

6. Implementation and Maintenance.

CHAPTER 5

REQUIREMENT ANALYSIS

The Requirement analysis is an important process in software development that helps identify the necessary features and functions of a software application. Here are some requirements that may be considered for a chat analyzer using Python Streamlit and Matplotlib:

Data Input: The chat analyzer should be able to accept chat data in a suitable format such as CSV or JSON.

Data Preprocessing: The chat analyzer should be able to clean the chat data, remove stopwords, and tokenize the text using Python libraries such as NLTK or Spacy.

Sentiment Analysis: The chat analyzer should be able to perform sentiment analysis on the chat data using Python libraries such as TextBlob or VADER.

Topic Modeling: The chat analyzer should be able to perform topic modeling on the chat data using Python libraries such as Gensim or Scikit-learn.

Visualization: The chat analyzer should be able to create interactive visualizations using Python Streamlit and Matplotlib.

Chatbot Performance Analysis: The chat analyzer should be able to analyze chatbot performance metrics such as response time, accuracy, and user satisfaction using Python libraries and techniques.

Insights and Recommendations: The chat analyzer should be able to draw insights from the analysis and make recommendations based on the findings.

Usability: The chat analyzer should have a user-friendly interface that allows users to easily upload and analyze chat data, visualize results, and interact with the chatbot performance analysis.

Scalability: The chat analyzer should be scalable and able to handle large volumes of chat data.

Security: The chat analyzer should have security features such as encryption and access control to ensure the confidentiality of the chat data.

Compatibility: The chat analyzer should be compatible with different operating systems and browsers.

Maintenance: The chat analyzer should be easy to maintain, with regular updates and bug fixes.

Overall, the requirement analysis for a chat analyzer using Python Streamlit and Matplotlib should consider data input, data preprocessing, sentiment analysis, topic modeling, visualization, Chabot performance analysis, insights and recommendations, usability, scalability, security, compatibility, and maintenance.

CHAPTER 6

SYSTEM DESIGNING

The system design of a chat analyzer using Python involves acquiring chat data from various sources, preprocessing the data to clean and transform it, analyzing the data to extract insights and trends, visualizing the results using charts and graphs, and building a user interface for interactive exploration. The design relies on Python libraries such as pandas, NumPy, NLTK, Matplotlib, seaborn, scikit-learn, and Streamlit

6.1 DATABASE DESIGN

Database design for a chat analyzer in Python involves creating a database to efficiently store and retrieve chat data. The design includes defining the schema, creating tables, and establishing relationships between them. Python libraries like SQLite, MySQL, or PostgreSQL can be used to create and interact with the database. The choice of database depends on the size of the dataset and complexity of the analysis.

6.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination Data flowcharts can range from simple, even hand-drawn process overviews to in-depth multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model new one. Like all the best diagrams and charts, a DFD can often visually "say things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems

♦ **ER-DIAGRAM:**

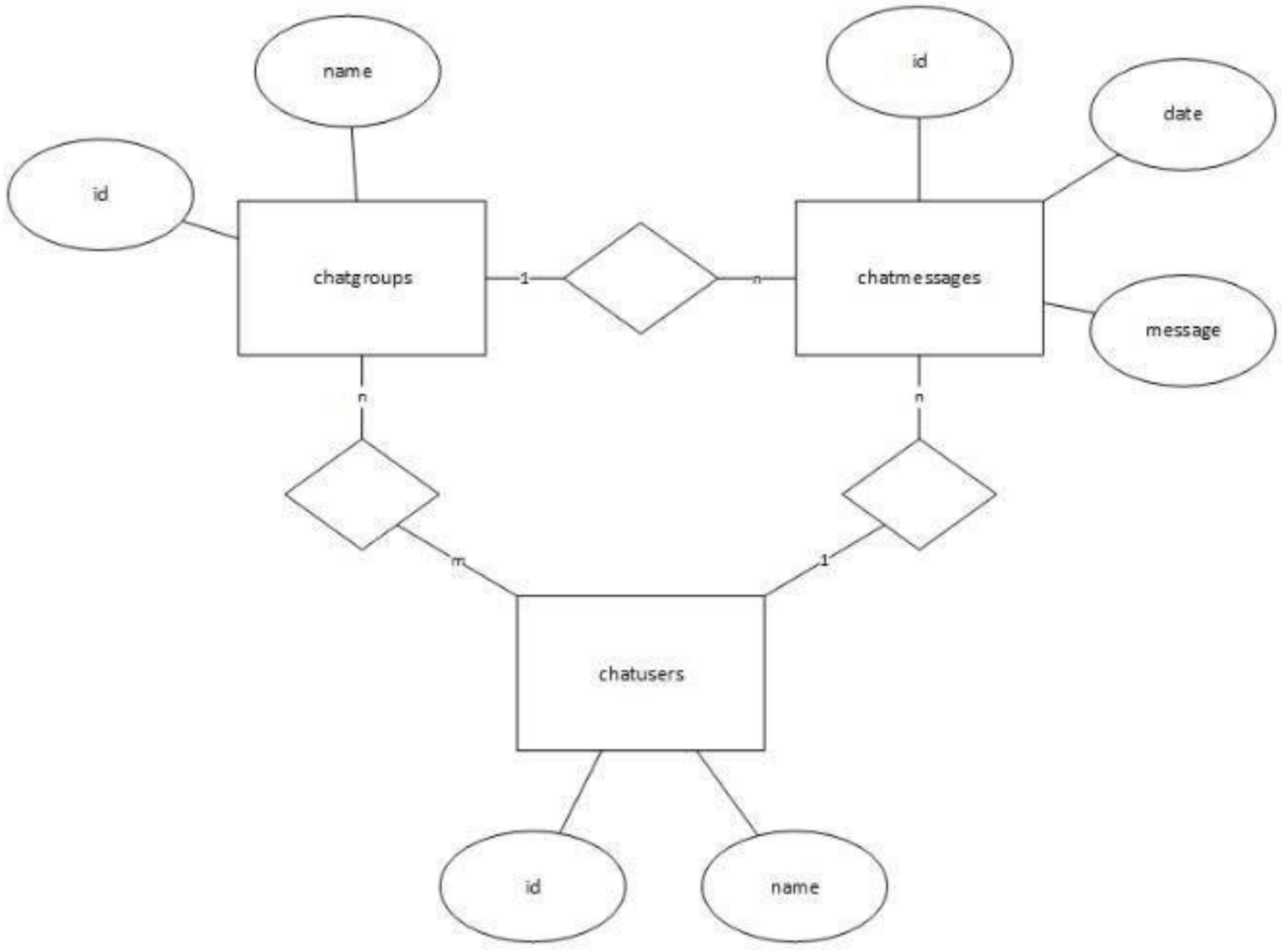


Figure 1 ER DIAGRAM OF CHAT ANALYZER USING PYTHON

RESULT AND DISCUSSION

- The results of chat analysis using Python can provide valuable insights into communication patterns, sentiment, and topics discussed. These insights can be used to inform decision-making and improve communication strategies.
- One common analysis is sentiment analysis, which involves identifying the sentiment of chat messages as positive, negative, or neutral. This analysis can provide insight into the overall sentiment of the conversation, as well as the sentiment of individual participants. For example, if a customer service chat log reveals that many customers are expressing negative sentiment, this may indicate a need to improve the quality of customer service.
- Topic modeling is another popular analysis technique that involves identifying the topics discussed in a chat log. This can provide insight into what subjects are most commonly discussed and can inform content creation and messaging strategies. For example, if a chat log reveals that a particular product or feature is frequently mentioned, this may indicate a need to focus marketing efforts on that product or feature.
- Network analysis can also be used to analyze communication patterns in a chat log. This involves mapping the connections between participants and identifying key influencers or groups. This can inform communication strategies by identifying the most effective channels for communication and the key stakeholders to engage with.
- In terms of discussion, the results of chat analysis should be interpreted in the context of the objectives and scope of the analysis. The findings should be presented clearly and concisely, using visualizations to facilitate understanding. It is important to consider the limitations of the data and the analysis techniques used and to acknowledge any potential biases or inaccuracies. The discussion should also include recommendations for how the insights gained from the analysis can be used to inform decision-making and improve communication strategies.

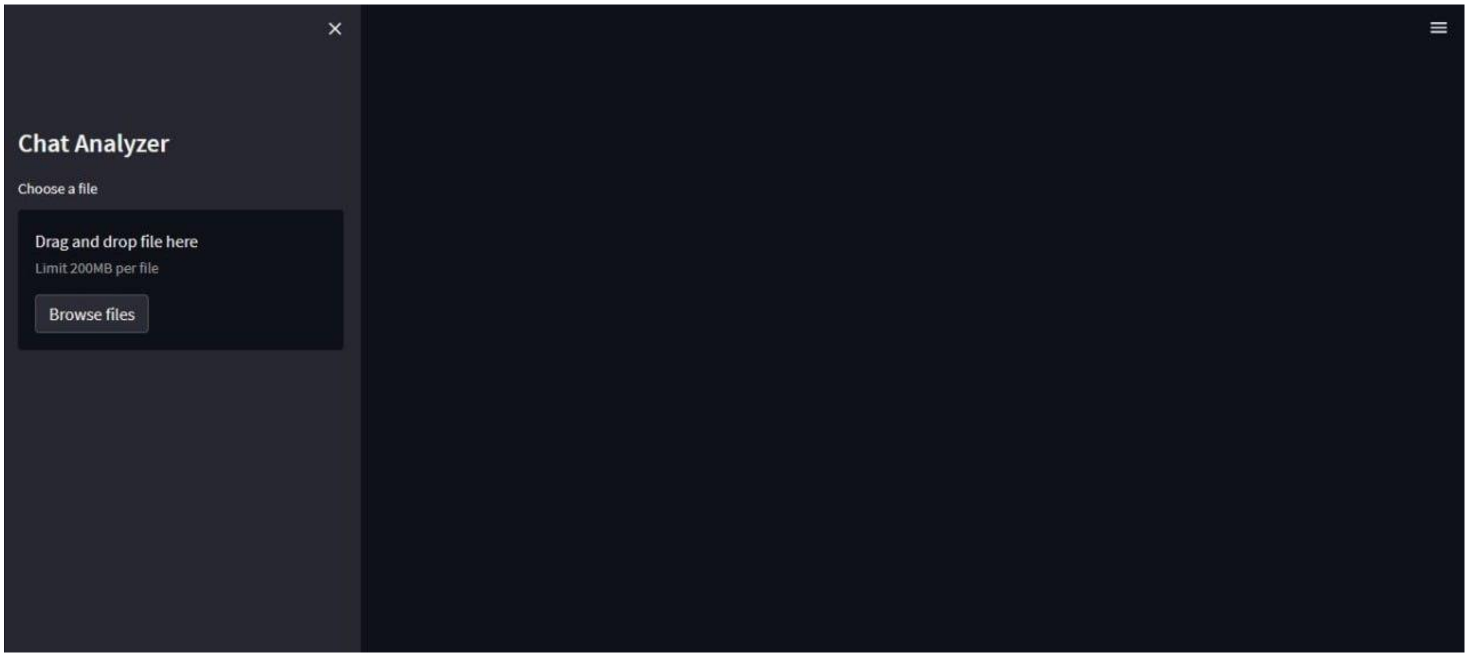


Figure 2 HOME PAGE

This is the chat analyzer home page with some features suchas:

- A button to browse txt files containing large text data or chat.
- The Source data file must be in txt format and it can be upto 200mb large.
- The uploaded file will be processed by the chat analyzer and all the analyzed data and outcomes will be visualized and presented to the user for further research and analysis.

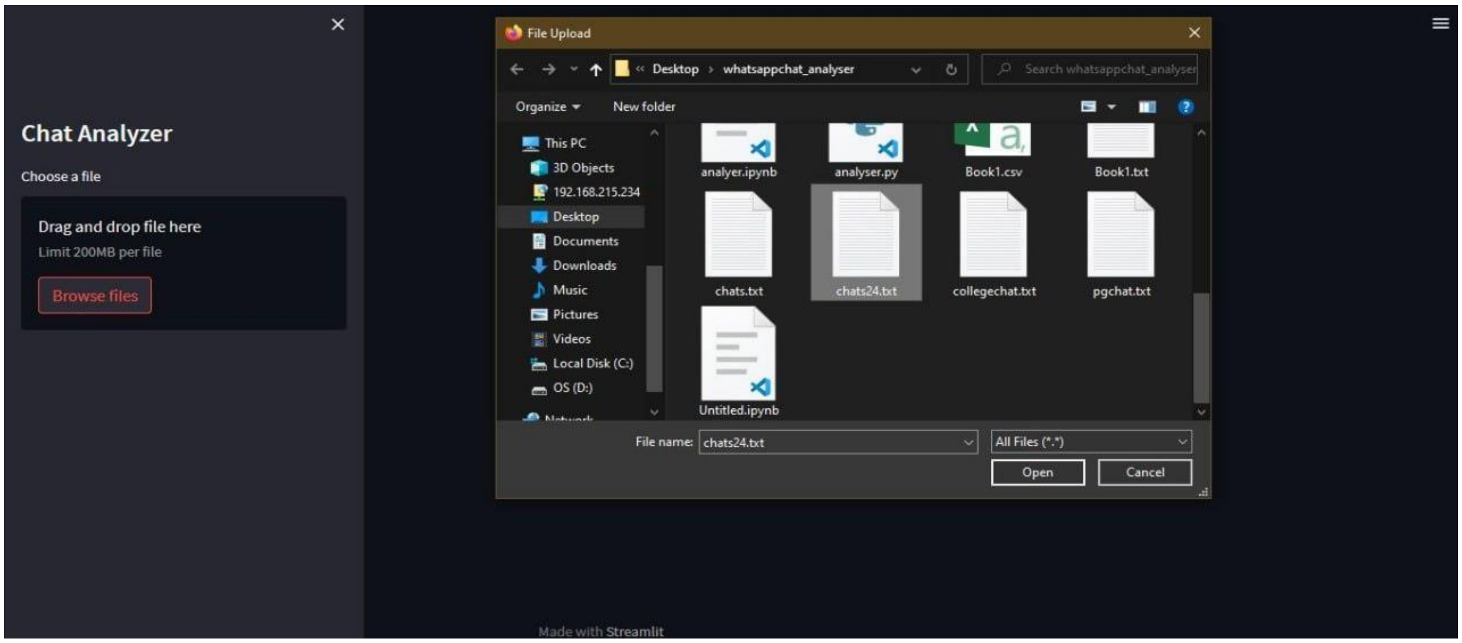


Figure 3 BROWSING FILE

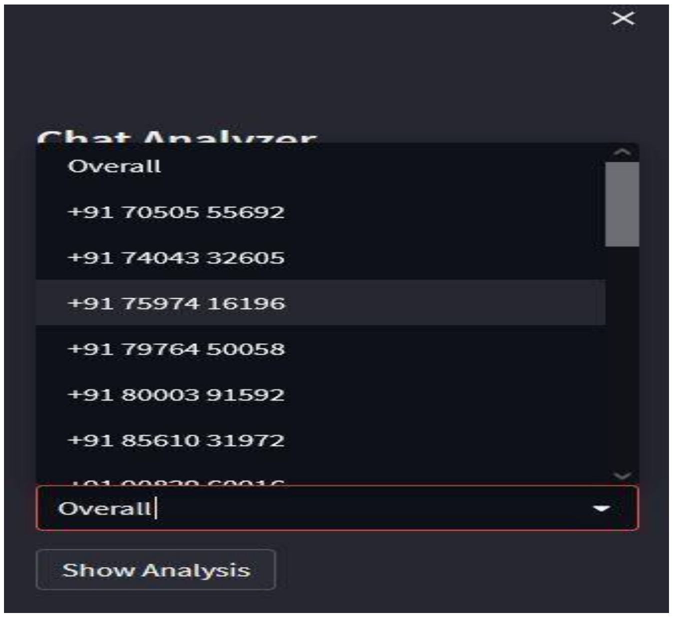
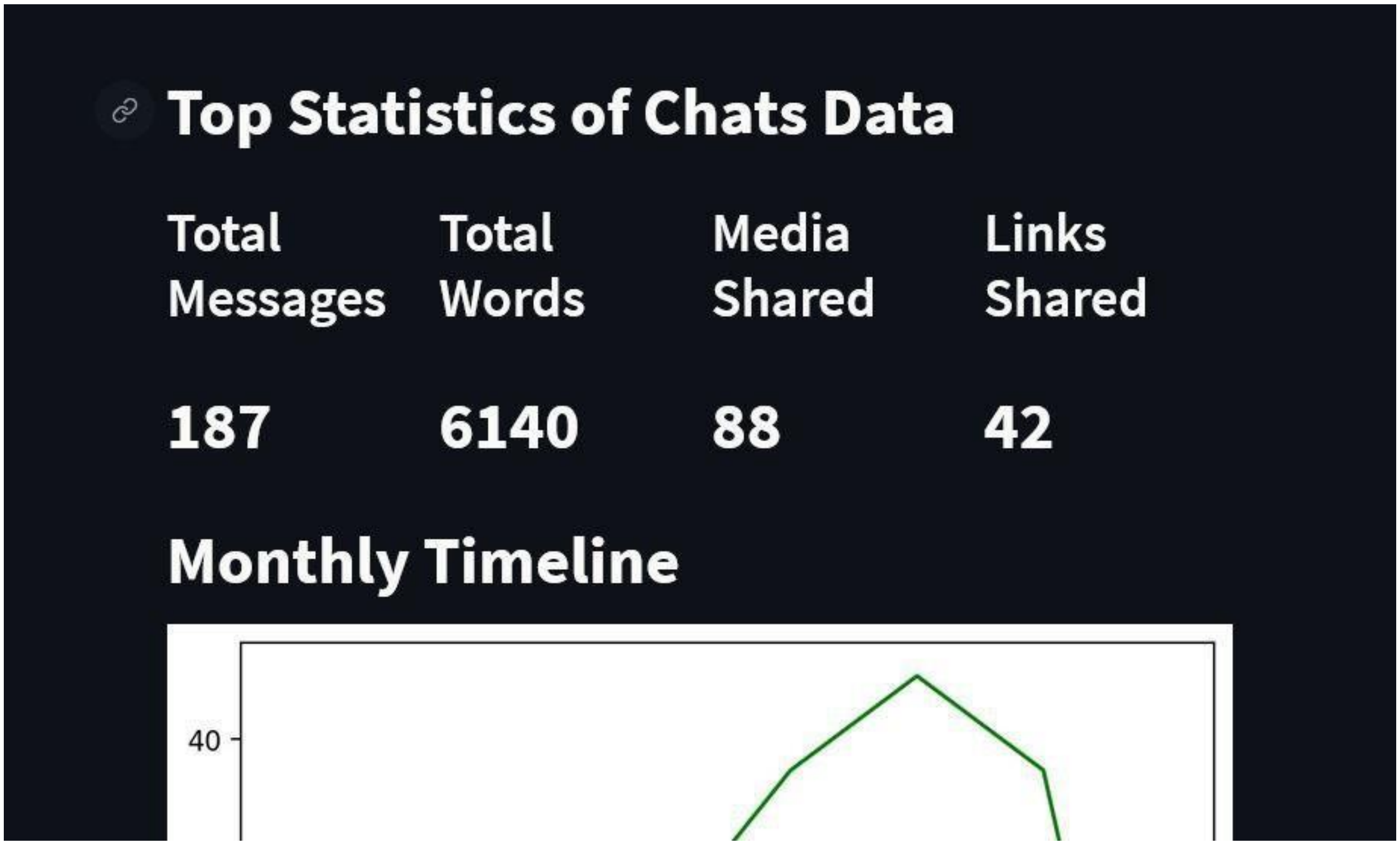


Figure 4 selecting the scope of analysis

- This page will show the list of involved entities.
- Here we can choose any particular user or entity or select all the entities at once.
- Only after the selecting the scope of analysis, the outcomes will be displayed on the screen


Top Statistics of Chats Data

Total Messages	Total Words	Media Shared	Links Shared
187	6140	88	42

Monthly Timeline

Figure 5 statistics

- This is the stastical information found in the data being analyzed.
- This section contains all the quick access records that could show the type of media shared, words typed, link shared etc.

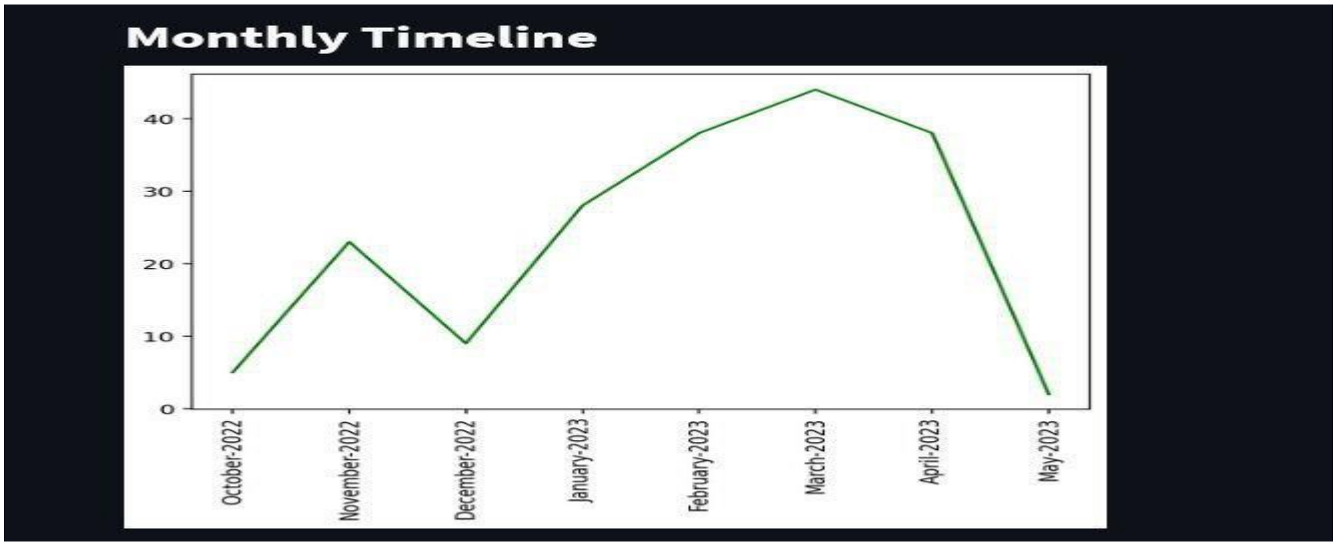


Figure 6 Monthly Timeline

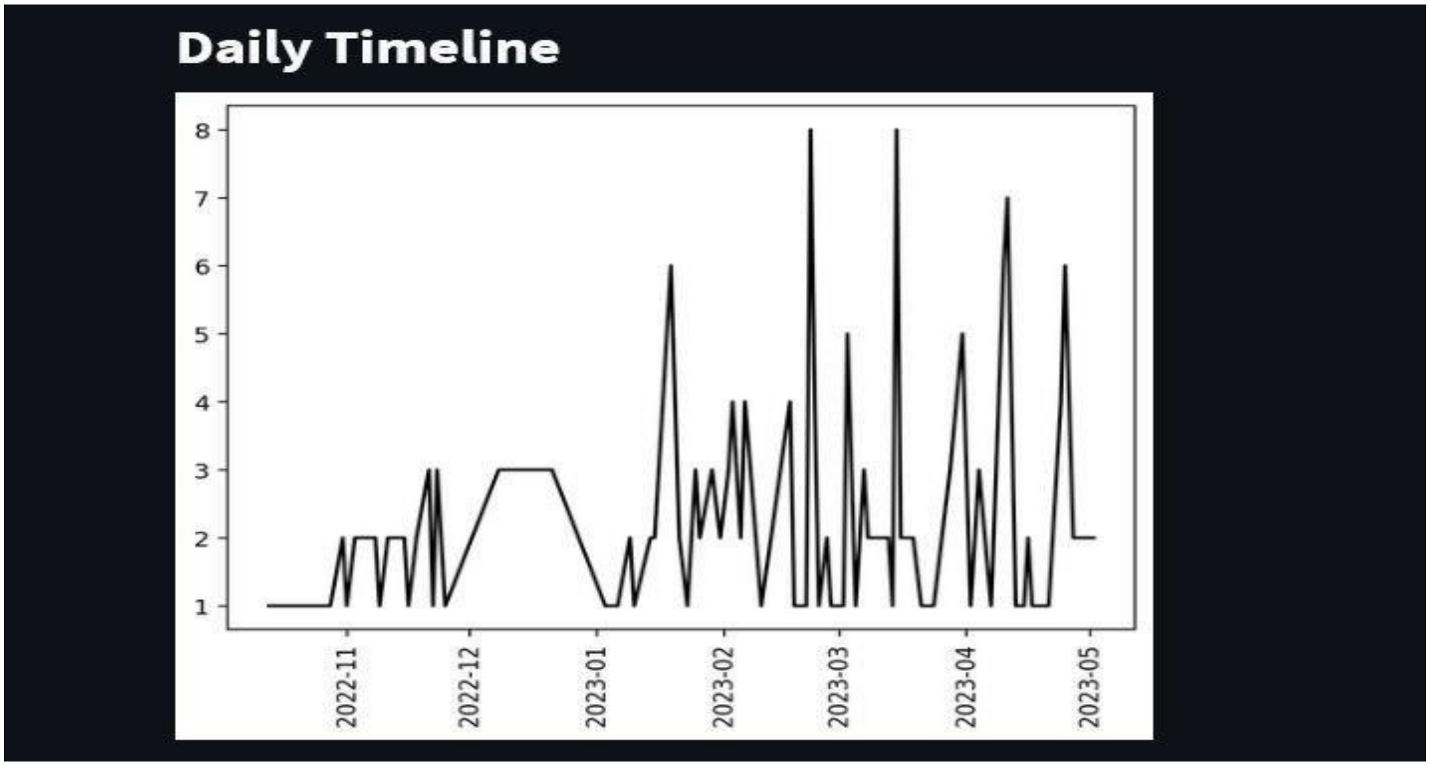


Figure 7 Daily Timeline

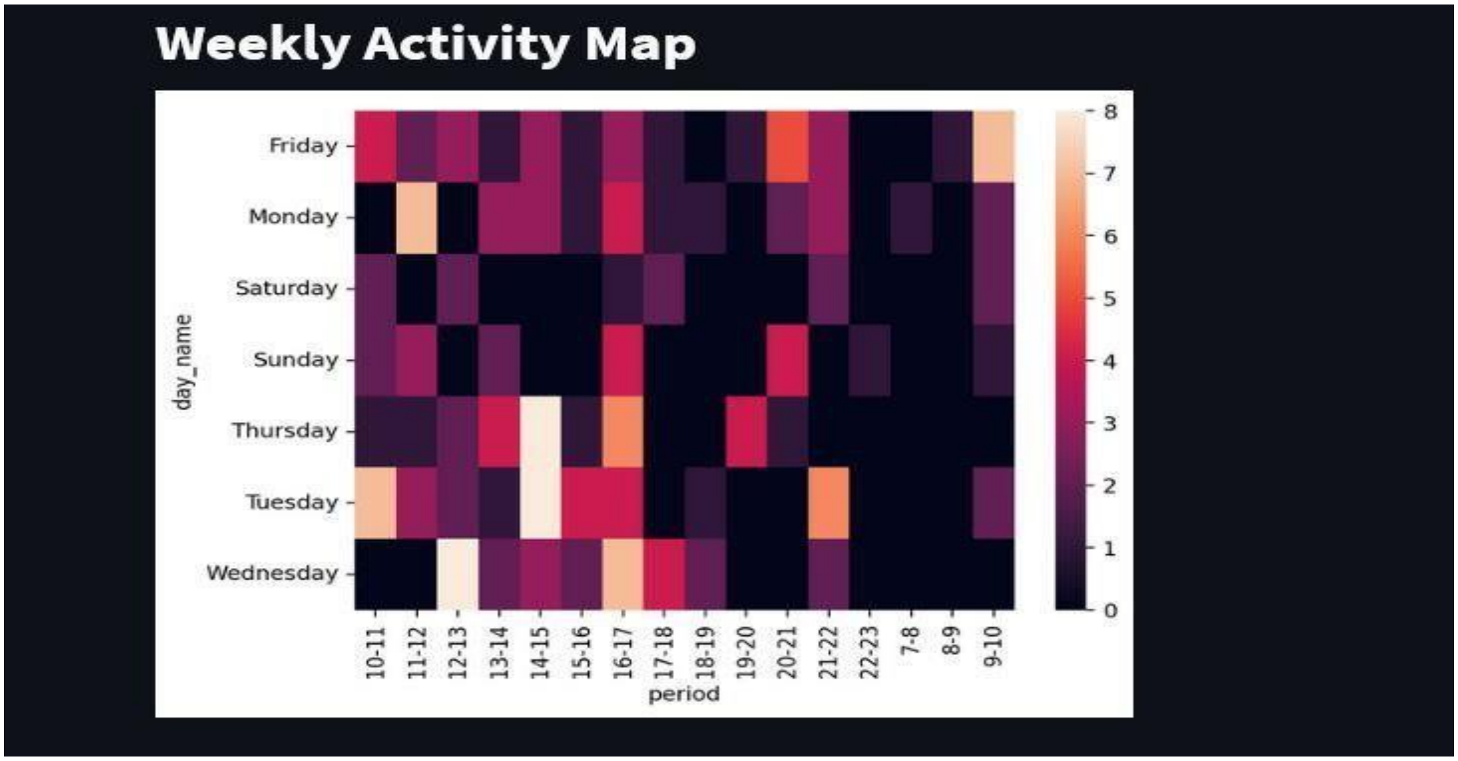


Figure 8 Weekly Activity Map

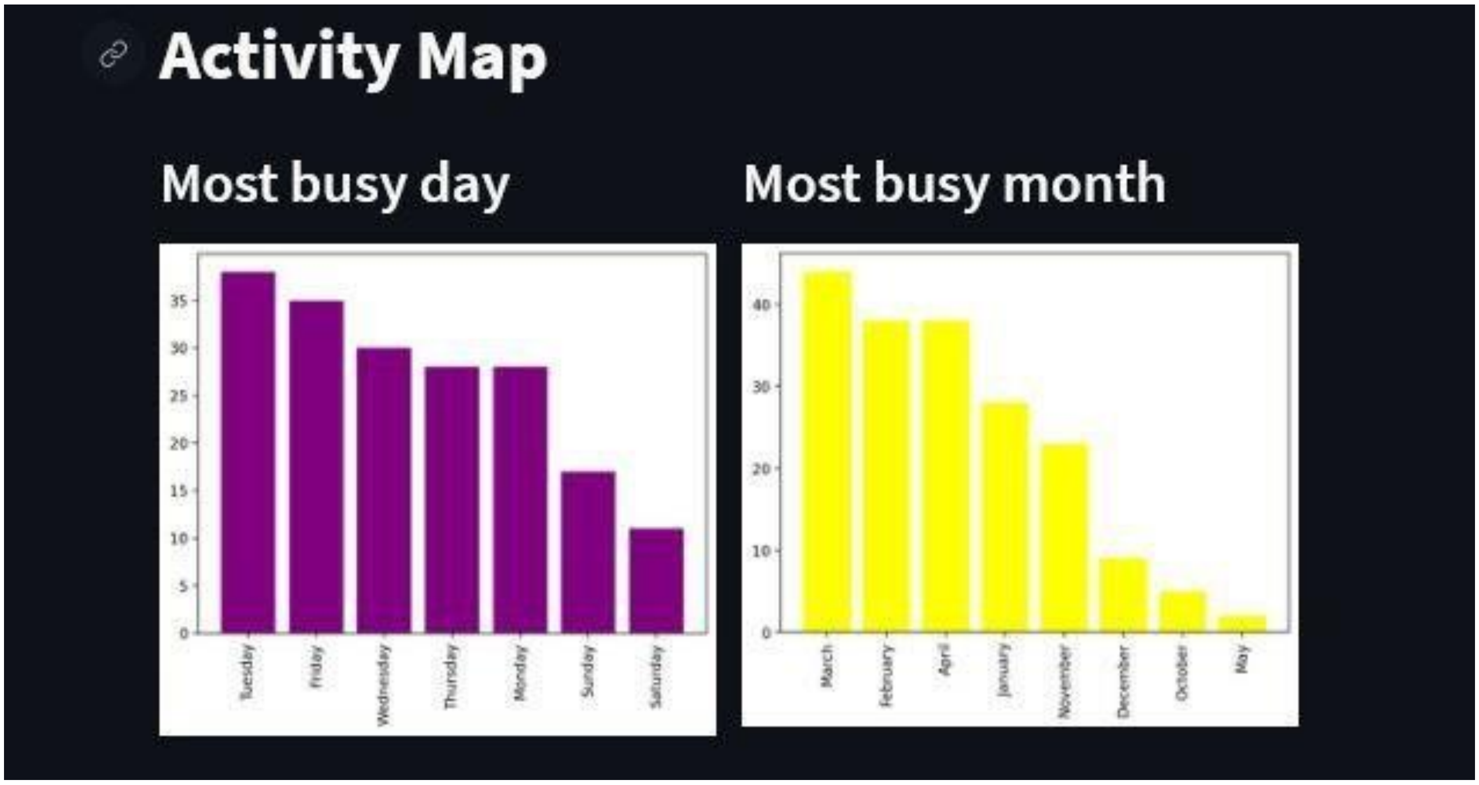


Figure 9 Activity map

- The Activity map has two parts most busy day and most busy month.
- It can tell which day of the week is the busiest and which month of the year was most busy.

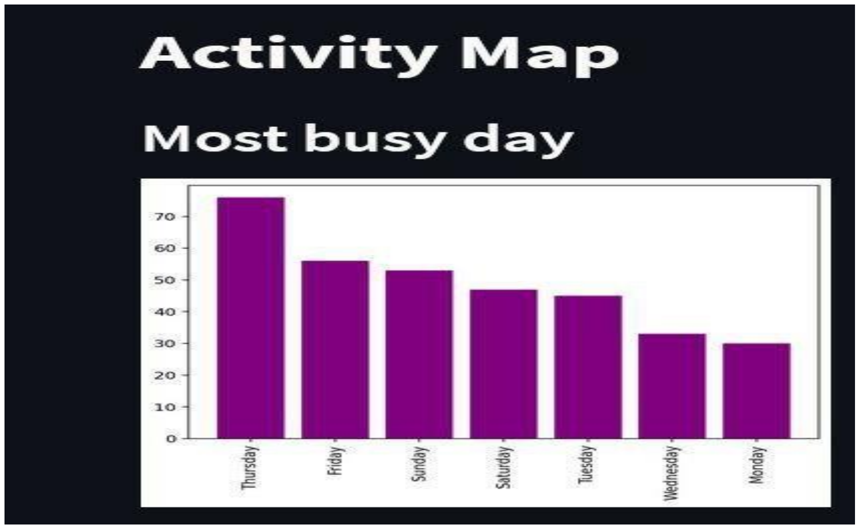


Figure 10 Most Busy Day

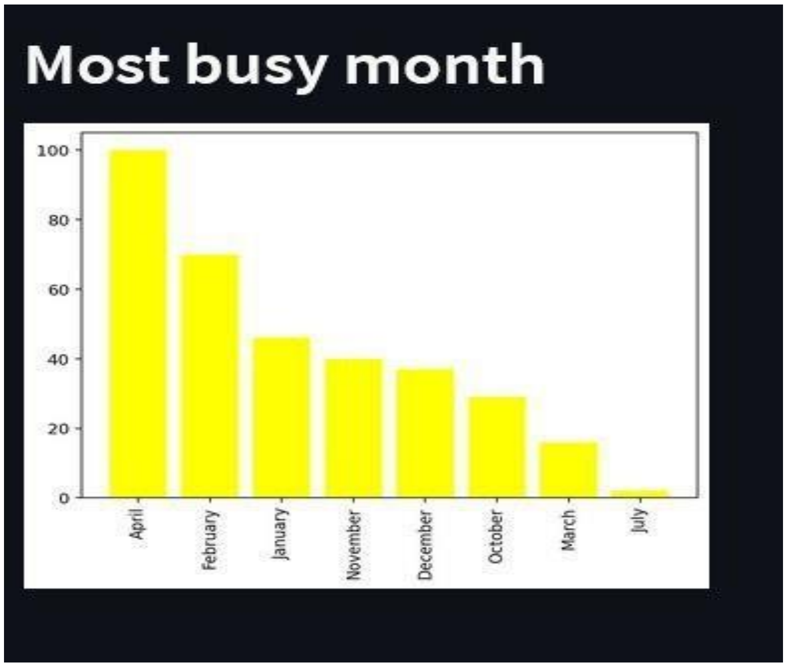
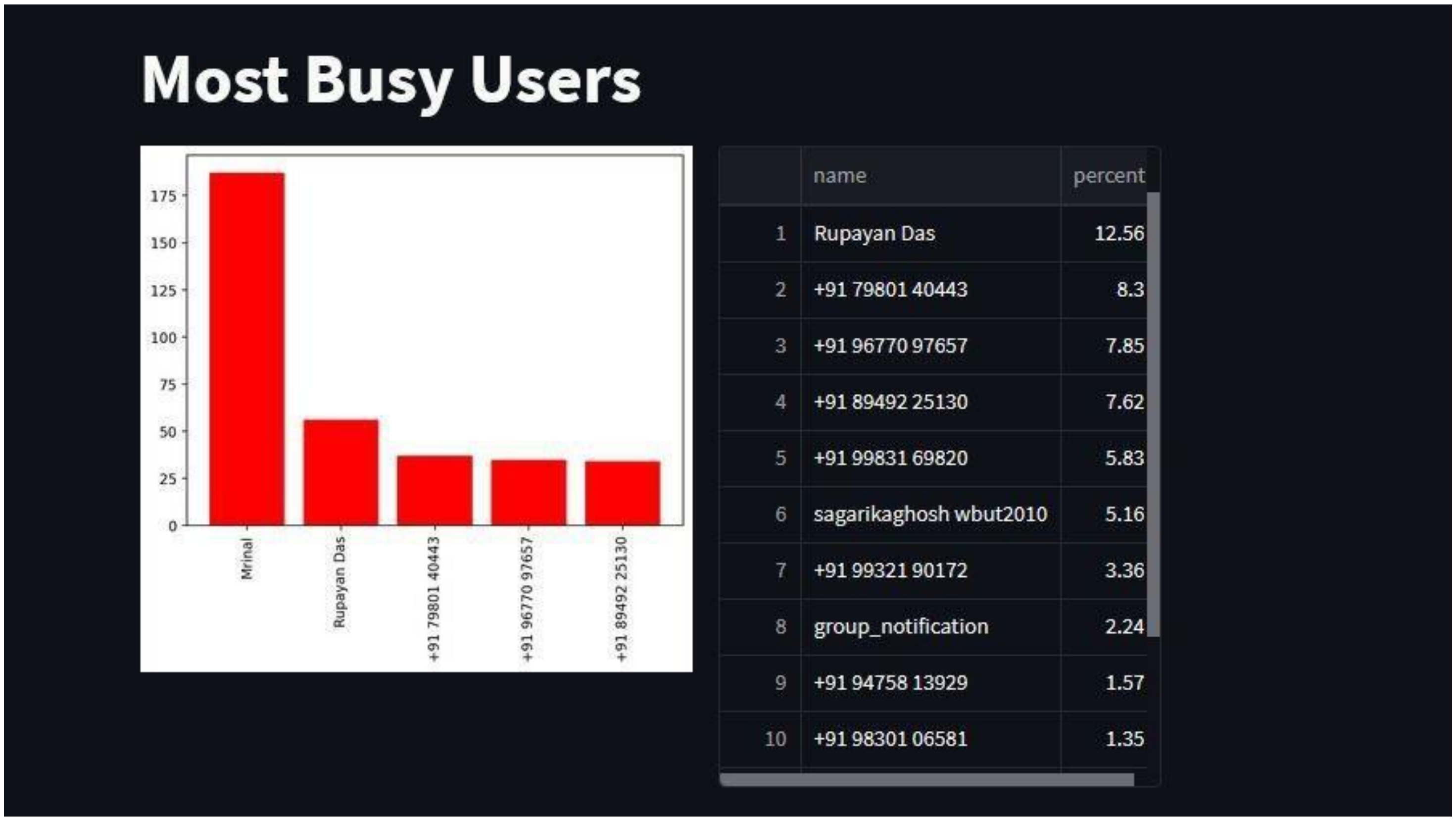


Figure 11 Most Busy Month



	name	percent
1	Rupayan Das	12.56
2	+91 79801 40443	8.3
3	+91 96770 97657	7.85
4	+91 89492 25130	7.62
5	+91 99831 69820	5.83
6	sagarikaghosh wbut2010	5.16
7	+91 99321 90172	3.36
8	group_notification	2.24
9	+91 94758 13929	1.57
10	+91 98301 06581	1.35

Figure 12 Most Busy Users

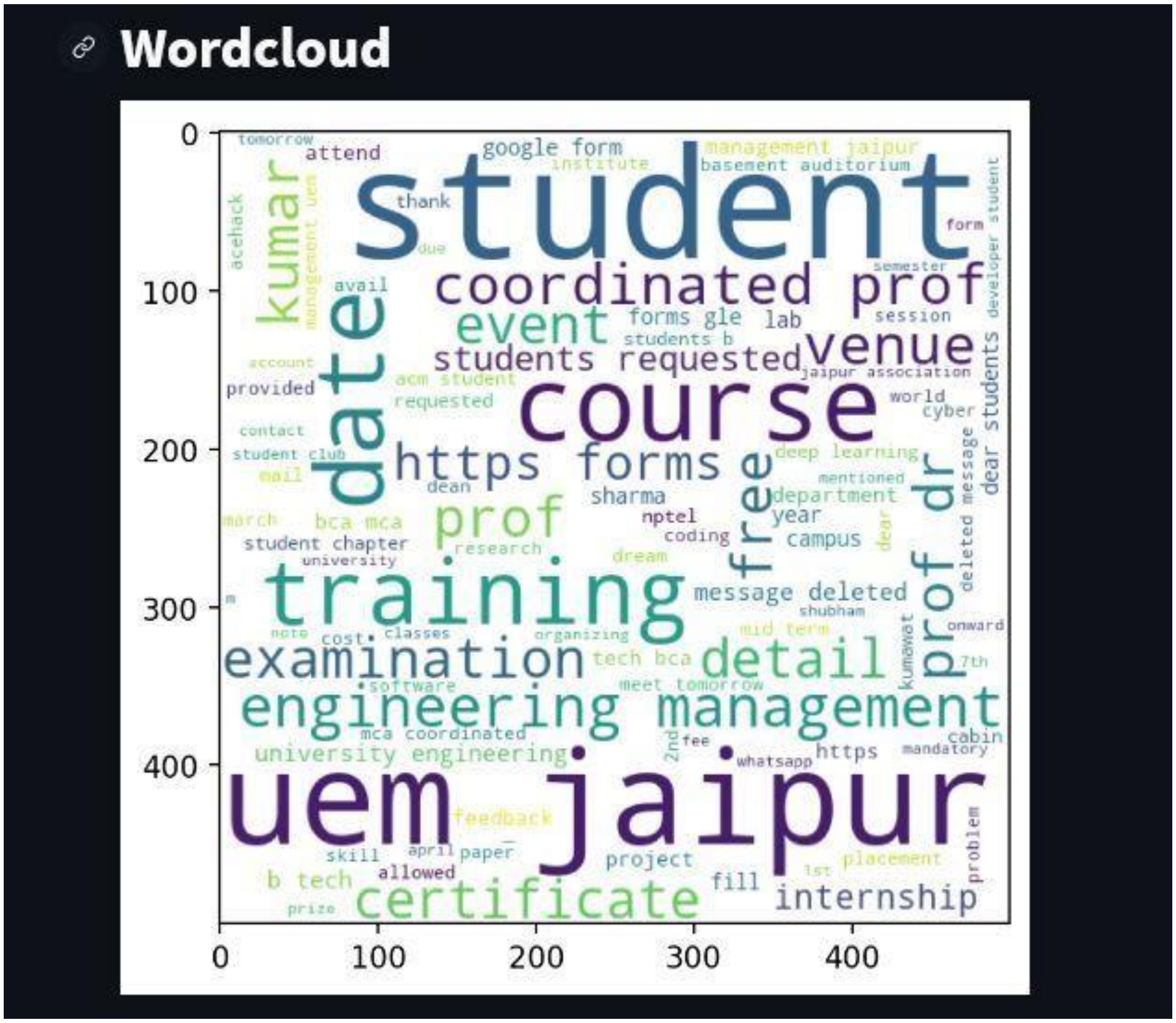


Figure 13 Word Cloud

- The word cloud library in Python can be used to generate word clouds from text data.
- The appearance and layout of the word cloud can be customized using the library's built-in options.
- Word clouds can be integrated with other Python libraries such as Matplotlib and Pandas for more detailed analysis.
- The output can be saved as image files or integrated into web applications using frameworks such as Streamlit.

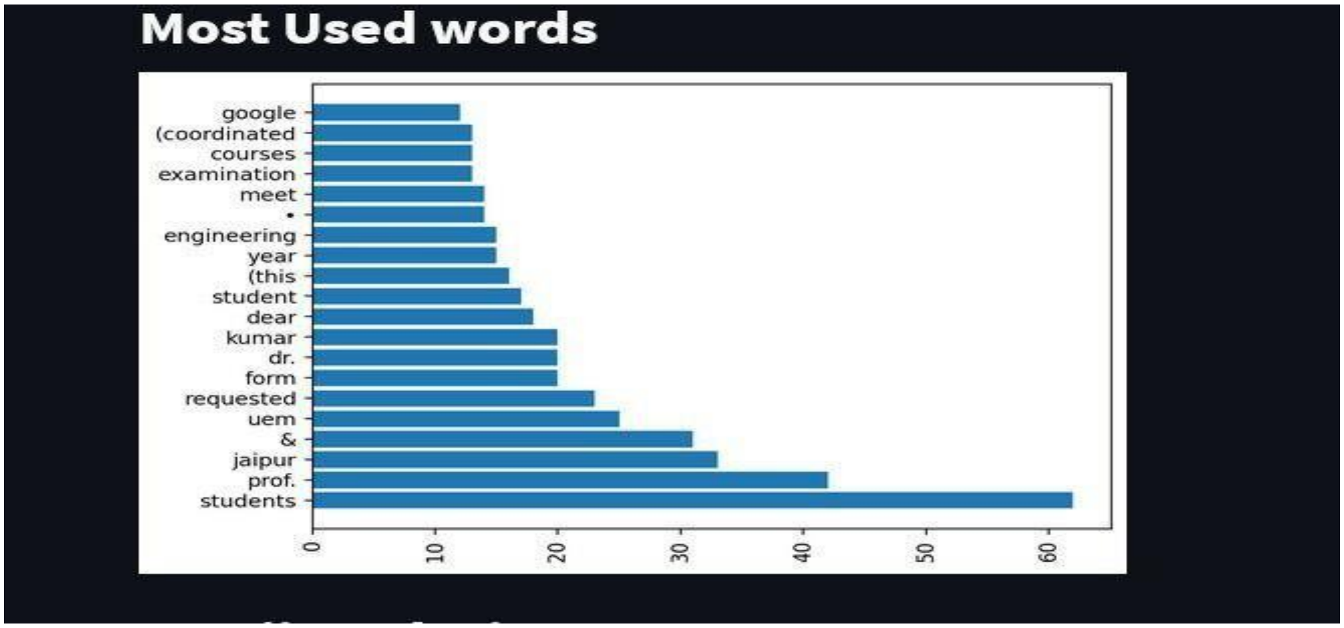
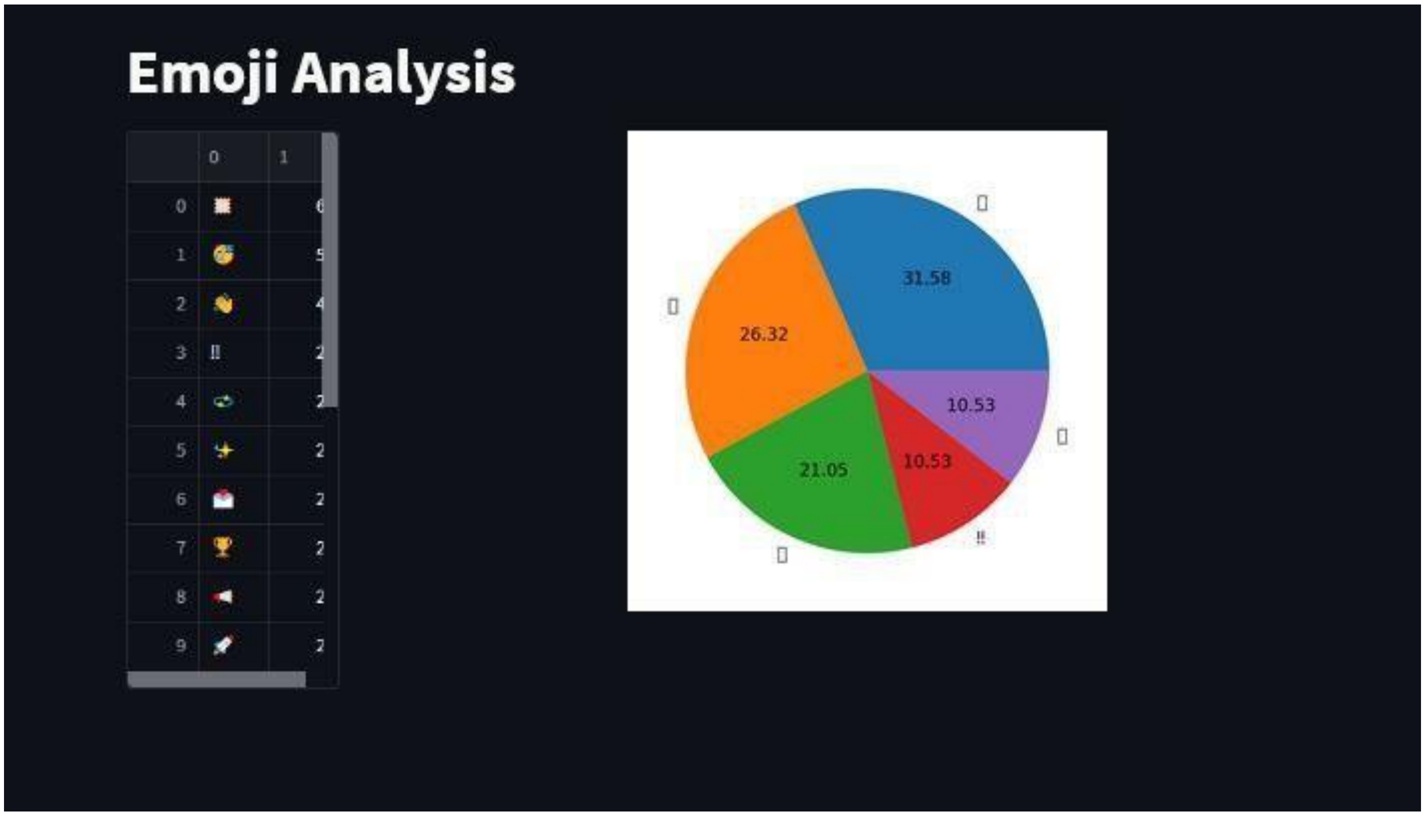


Figure 14 Most Used Words



Emoji Analysis

	0	1
0	🏻	6
1	😴	5
2	👋	4
3	‼	2
4	🔄	2
5	✨	2
6	📩	2
7	🏆	2
8	📢	2
9	🚀	2

Figure 15 Emoji Analysis

CONCLUSION AND FUTURE SCOPE

The major objective that has been decided in the initial phase of the requirement analysis is achieved successfully. After the implementation, the system provides reliable results.

The system is totally menu and user friendly, which makes it easy for the users even with limited knowledge of computer environment to operate the developed system. The system avoids the drawbacks of the existing manual system and the validation facility of the system totally eliminates the chances of wrong data entry.

It has following features:

• User friendly.

• Time saving.

• Runs on any devices.

• Analyzes any WhatsApp imported file.

• Accuracy.

• Reliability.

• Easy to use.

Voice-based chat analysis: As the number of voice assistants and chatbots that utilise speech as their primary communication method rises, it may be possible to create a chat analyst that will examine voice-based chat data and offer insights and suggestions for organisations.

Analysis based on machine learning: A chat analyzer that does more complex analyses on chat data, such as entity recognition, sentiment analysis, and intent detection, could be created.

Integration with other data sources: To give a more comprehensive picture of customer engagement and behaviour, chat analyzer could be combined with other data sources like social media data or website analytics.

Natural Language Generation: Based on the information gathered, Natural Language Generation (NLG) techniques could be utilised to automatically provide answers to client enquiries.

APPENDIX

APP.PY

```
import streamlit as st
import preprocessor,helper
import matplotlib.pyplot as plt
import seaborn as sns

st.sidebar.title("Chat Analyzer")

uploaded_file = st.sidebar.file_uploader("Choose a file")
if uploaded_file is not None:
# To read file as bytes:
bytes_data = uploaded_file.getvalue()
# uploaded file is stream so we convert it into string or it will show into screem into text format
data = bytes_data.decode("utf-8")
# st.text(data)
# user preprocess file
df = preprocessor.preprocess(data)

# st.dataframe(df)

# fetch unique users
user_list = df['user'].unique().tolist()
user_list.remove('group_notification')
user_list.sort()
user_list.insert(0,"Overall")

selected_user = st.sidebar.selectbox("Show User overall ",user_list)

if st.sidebar.button("Show Analysis"):

# Stats Area
num_messages, words, num_media_messages,num_links = helper.fetch_stats(selected_user,df)
st.title("Top Statistics of Chats Data")
col1, col2, col3, col4 = st.columns(4)
with col1:
st.header("Total Messages ")
st.title(num_messages)
with col2:
st.header("Total Words")
st.title(words)
with col3:
st.header("Media Shared")
st.title(num_media_messages)
with col4:
st.header("Links Shared")
st.title(num_links)

# monthly timeline
st.title("Monthly Timeline")
timeline = helper.monthly_timeline(selected_user,df)
fig,ax = plt.subplots()
ax.plot(timeline['time'], timeline['message'],color='green')
plt.xticks(rotation='vertical')
st.pyplot(fig)


# daily timeline
st.title("Daily Timeline")
daily_timeline = helper.daily_timeline(selected_user, df)
fig, ax = plt.subplots()
ax.plot(daily_timeline['only_date'], daily_timeline['message'], color='black')
plt.xticks(rotation='vertical')
st.pyplot(fig)


# activity map
st.title('Activity Map')
col1,col2 = st.columns(2)

with col1:
st.header("Most busy day")
busy_day = helper.week_activity_map(selected_user,df)
fig,ax = plt.subplots()
ax.bar(busy_day.index,busy_day.values,color='purple')
plt.xticks(rotation='vertical')
st.pyplot(fig)

with col2:
st.header("Most busy month")
busy_month = helper.month_activity_map(selected_user, df)
```

```
fig, ax = plt.subplots()
ax.bar(busy_month.index, busy_month.values,color='yellow')
plt.xticks(rotation='vertical')
st.pyplot(fig)

st.title("Weekly Activity Map")
user_heatmap = helper.activity_heatmap(selected_user,df)
fig,ax = plt.subplots()
ax = sns.heatmap(user_heatmap)
st.pyplot(fig)

# st.title("Weekly Activity Map")
# user_heatmap = helper.activity_heatmap(selected_user,df)
# fig,ax = plt.subplots()
# ax = sns.heatmap(user_heatmap)
# st.pyplot(fig)
# finding the busiest users in the group(Group level)
if selected_user == 'Overall':
st.title('Most Busy Users')

x,new_df = helper.most_busy_users(df)
fig, ax = plt.subplots()

col1, col2 = st.columns(2)

with col1:
ax.bar(x.index, x.values,color='red')
plt.xticks(rotation='vertical')
st.pyplot(fig)
with col2:
st.dataframe(new_df)

# WordCloud
st.title("Wordcloud")
df_wc = helper.create_wordcloud(selected_user,df)
fig,ax = plt.subplots()
ax.imshow(df_wc)
st.pyplot(fig)

# most common words
most_common_df = helper.most_common_words(selected_user,df)
```

```
# st.dataframe(most_common_df)

fig,ax = plt.subplots()
```

PREPROCESSOR.PY – USED FOR DATA EXTRACTION

```
import re
import pandas as pd

def preprocess(data):
pattern = '\d{1,2}/\d{1,2}/\d{1,2},\s\d{1,2}:\d{2}\s-\s'

messages = re.split(pattern, data)[1:]
dates = re.findall(pattern, data)
# here we convert it into two dataframe

df = pd.DataFrame({'user_message': messages, 'message_date': dates})
df['message_date'] = pd.to_datetime(df['message_date'],format = '%d/%m/%y, %H:%M - ')
# # convert message_date type
df.rename(columns={'message_date': 'date'}, inplace=True)
# df.head()

users = []
messages = []
for message in df['user_message']:
entry = re.split('([\w\W]+?):\s', message)
if entry[1:]: # user name
users.append(entry[1])
messages.append(" ".join(entry[2:]))
else:
users.append('group_notification')
messages.append(entry[0])

df['user'] = users
df['message'] = messages
df.drop(columns=['user_message'], inplace=True)

df['only_date'] = df['date'].dt.date
df['year'] = df['date'].dt.year
df['month_num'] = df['date'].dt.month
df['month'] = df['date'].dt.month_name()
```

```
df['day'] = df['date'].dt.day
df['day_name'] = df['date'].dt.day_name()
df['hour'] = df['date'].dt.hour
df['minute'] = df['date'].dt.minute

period = []
for hour in df[['day_name', 'hour']]['hour']:
if hour == 23:
period.append(str(hour) + "-" + str('00'))
elif hour == 0:
period.append(str('00') + "-" + str(hour + 1))
else:
period.append(str(hour) + "-" + str(hour + 1))

df['period'] = period

return df
```

HELPER.PY-ALL THE FUNCTIONS ARE DECLARED

```
from urlextract import URLExtract
from wordcloud import WordCloud
import pandas as pd
from collections import Counter
import emoji
extract = URLExtract()
def fetch_stats(selected_user,df):

# if selected_user=='Overall':
# # fetch the number of messages
# num_messages = df.shape[0]

# # fetch the total number of words
# words = []
# for message in df['message']:
# words.extend(message.split())
# return num_messages , len(words)
# else:
# new_df = df[df['user']==selected_user]
# num_messages = new_df.shape[0]
# words = []
# for message in new_df['message']:
# words.extend(message.split())
# return num_messages , len(words)
# arrange upper code in a better structure
if selected_user!='Overall':
df = df[df['user']==selected_user]
num_messages = df.shape[0]
# fetch the total number of words
words = []
for message in df['message']:
words.extend(message.split())
# fetch number of media messages
num_media_messages = df[df['message'] == '<Media omitted>\n'].shape[0]

# fetch number of links shared
links = []
for message in df['message']:
links.extend(extract.find_urls(message))
return num_messages , len(words), num_media_messages, len(links)

def most_busy_users(df):
x = df['user'].value_counts().head()
df = round((df['user'].value_counts() / df.shape[0]) * 100, 2).reset_index().rename(
columns={'index': 'name', 'user': 'percent'})
return x,df

def create_wordcloud(selected_user,df):
# if selected_user != 'Overall':
# df = df[df['user'] == selected_user]

# wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
# df_wc = wc.generate(df['message'].str.cat(sep=" "))
# return df_wc
f = open('helpers.txt', 'r')
stop_words = f.read()

if selected_user != 'Overall':
df = df[df['user'] == selected_user]

temp = df[df['user'] != 'group_notification']
temp = temp[temp['message'] != '<Media omitted>\n']

def remove_stop_words(message):
y = []
for word in message.lower().split():
if word not in stop_words:
y.append(word)
return " ".join(y)

wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
temp['message'] = temp['message'].apply(remove_stop_words)
df_wc = wc.generate(temp['message'].str.cat(sep=" "))
return df_wc

def most_common_words(selected_user,df):

f = open('helpers.txt','r')
stop_words = f.read()

if selected_user != 'Overall':
df = df[df['user'] == selected_user]

temp = df[df['user'] != 'group_notification']
temp = temp[temp['message'] != '<Media omitted>\n']

words = []

for message in temp['message']:
for word in message.lower().split():
if word not in stop_words:
words.append(word)

most_common_df = pd.DataFrame(Counter(words).most_common(20))
return most_common_df


def emoji_helper(selected_user,df):
if selected_user != 'Overall':
df = df[df['user'] == selected_user]

emojis = []
```

```
for message in df['message']:
emojis.extend([c for c in message if c in emoji.EMOJI_DATA])

emoji_df = pd.DataFrame(Counter(emojis).most_common(len(Counter(emojis))))

return emoji_df

def monthly_timeline(selected_user,df):

if selected_user != 'Overall':
df = df[df['user'] == selected_user]

timeline = df.groupby(['year', 'month_num', 'month']).count()['message'].reset_index()

time = []
for i in range(timeline.shape[0]):
time.append(timeline['month'][i] + "-" + str(timeline['year'][i]))

timeline['time'] = time

return timeline


def daily_timeline(selected_user,df):

if selected_user != 'Overall':
df = df[df['user'] == selected_user]

daily_timeline = df.groupby('only_date').count()['message'].reset_index()

return daily_timeline


def week_activity_map(selected_user,df):

if selected_user != 'Overall':
df = df[df['user'] == selected_user]

return df['day_name'].value_counts()

def month_activity_map(selected_user,df):
```

```
if selected_user != 'Overall':
df = df[df['user'] == selected_user]

return df['month'].value_counts()

def activity_heatmap(selected_user,df):

if selected_user != 'Overall':
df = df[df['user'] == selected_user]

user_heatmap = df.pivot_table(index='day_name', columns='period', values='message',
aggfunc='count').fillna(0)

return user_heatma
```

BIBLIOGRAPHY

Here is a sample bibliography for building a chat analyzer using Python:

1. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media, Inc.
2. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc.

3. Zhang, Y., & Wallace, B. (2019). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.

4. Hutto, C. J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In Eighth International Conference on Weblogs and Social Media (ICWSM-14).

5. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Conference on Empirical Methods in Natural Language Processing (EMNLP).

6. Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 3(Feb), 1137-1155.

7. Chen, J., Chen, J., Zhang, Y., & Xia, F. (2016). A Survey on Dialogue Systems: Recent Advances and New Frontiers. arXiv preprint arXiv:1611.06214.

8. Goyal, P., Ferrara, E., & Menczer, F. (2018). Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems, 151, 78-94.

9. Lu, J., & Getoor, L. (2011). Link-based classification. In Proceedings of the 28th international conference on machine learning (ICML-11) (pp. 969-976).

10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

9. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media, Inc.

10. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc.

11. Zhang, Y., & Wallace, B. (2019). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.

12. Hutto, C. J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In Eighth International Conference on Weblogs and Social Media (ICWSM-14).

13. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Conference on Empirical Methods in Natural Language Processing (EMNLP).

14. Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 3(Feb), 1137-1155.

15. Chen, J., Chen, J., Zhang, Y., & Xia, F. (2016). A Survey on Dialogue Systems: Recent Advances and New Frontiers. arXiv preprint arXiv:1611.06214.

16. Goyal, P., Ferrara, E., & Menczer, F. (2018). Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems, 151, 78-94.

10. Lu, J., & Getoor, L. (2011). Link-based classification. In Proceedings of the 28th international conference on machine learning (ICML-11) (pp. 969-976).

11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mentioning all those people who made it possible, the constant encouragement, crowns the efforts with success.

I wish my sincere gratitude to my project coordinator **Prof. DEBAJYOTI CHATTERJEE**, of University Of Engineering And Management, Jaipur for providing guidance throughout the course.

I convey my sincere regards to **Prof. Mrinal Kanti Sarkar**, HOD, University Of Engineering And Management, and Jaipur for providing guidance throughout the course.

I am indebted to my well-wishers and friends who encouraged me in successful completion of the project.