1.1. Background of study

WhatsApp, a popular messaging application, has become a prevalent means of communication for individuals and groups around the world. With its widespread use, WhatsApp groups and one-on-one chats have become platforms for discussions, sharing of information, and exchanging ideas. Analyzing these conversations can provide valuable insights into the dynamics of group interactions and individual behavior.

The aim of the project "WHATSAPP CHAT ANALYZER" in Python is to develop a tool that can extract and analyze data from WhatsApp chats, providing useful statistics and insights. The tool will allow users to input a WhatsApp chat text file, which contains the exported conversation data, and then perform various analyses on the data.

The project will focus on extracting meaningful information from the chat data, such as the total number of words used, the most common words used, the most active participants, and other relevant metrics. Additionally, the tool will provide visualizations, such as word clouds, histograms, and graphs, to aid in understanding the data and identifying patterns.

The insights gained from this analysis can be useful in various scenarios, such as tracking team communication in a professional setting, monitoring group dynamics in educational or community groups, and analyzing individual behavior and language patterns in personal conversations. This project has the potential to provide valuable information to users who are interested in understanding the dynamics of WhatsApp conversations and gaining insights from the data.

The implementation of the project will be done in Python, utilizing its rich ecosystem of libraries for text processing, data analysis, and visualization. The project will involve techniques such as natural language processing (NLP), data extraction, data manipulation, and data visualization to achieve the desired analysis and visualization of WhatsApp chat data.

The "WHATSAPP CHAT ANALYZER" project in Python aims to provide a tool that can extract and analyze data from WhatsApp chats, giving insights and statistics about the conversations. The project has potential applications in various domains and can provide valuable information for understanding group dynamics, individual behavior, and language patterns in WhatsApp chats.

1.2. Problem Statement

The problem at hand is to develop a WHATSAPP CHAT ANALYZER using Python that can provide statistics and insights about a WhatsApp group chat or a one-on-one chat. The aim is to extract relevant information from the chat data and present it in a meaningful and easy-to-understand manner for analysis and interpretation.

The key challenges to be addressed in this project include:

- Data extraction: Extracting data from WhatsApp chat logs which may be in the form of text files, CSV files, or other formats, and handling the variety of data formats and structures that may exist.
- Data preprocessing: Cleaning and processing the extracted data to remove irrelevant information such as system-generated messages, emojis, media files, etc., and converting the text data into a suitable format for analysis.
- Statistical analysis: Implementing algorithms to generate various statistics such as word count, most common words, most active participants, and other relevant metrics that can provide meaningful insights about the chat data.
- Visualization: Presenting the analyzed data in visually appealing and informative ways using graphical plots, charts, and tables to facilitate easy interpretation of the results.
- Efficiency and scalability: Ensuring that the solution is efficient and scalable enough to handle large chat datasets with multiple participants and extended time periods.

The successful completion of this project will result in a WHATSAPP CHAT ANALYZER that can provide valuable insights about the chat data, enabling users to gain a deeper understanding of the chat dynamics, participant behavior, and communication patterns in a WhatsApp group chat or a one-on-one chat. This tool can find applications in social media analytics, content creation, sentiment analysis, and other areas where understanding chat data is crucial.

1.3. Objectives and scope of study

1.3.1 Objectives

- To analyze WhatsApp group chats or 1-on-1 chats and extract statistical information such as the total number of words used, most common words, most active participants, etc.
- To create visualizations, graphs, and charts to represent the analyzed data for easy interpretation and understanding.
- To provide insights into chat patterns, word usage trends, and participant activities within the WhatsApp chat.
- To enable users to input chat logs in various formats and extract meaningful information regardless of the chat size or duration.
- To develop a user-friendly interface that allows users to interact with the application and obtain relevant statistics with minimal effort.
- To enhance the project with additional features such as sentiment analysis, word cloud generation, and chat sentiment trends over time.

1.3.2. Scope of Study:

The scope of the study for the WHATSAPP CHAT ANALYZER project includes the following:

- The project will be implemented in Python, utilizing libraries such as Pandas, Matplotlib, and NLTK for data extraction, analysis, and visualization.
- The project will focus on extracting statistical information from WhatsApp chat logs, including the total number of words used, most common words, most active participants, etc.
- The project will be able to handle chat logs of different sizes and durations, and it will be flexible in terms of accepting input in various formats, such as text files or exported chat logs from WhatsApp.
- The project will provide visualizations, such as bar charts, line charts, and pie charts, to represent the analyzed data in an easy-to-understand manner.
- The project will not involve any real-time analysis of WhatsApp chats or integration with the WhatsApp API.

- The project will not involve any automated actions, such as sending messages or modifying the chat content, within the WhatsApp chat.
- The project will not analyze multimedia content, such as images, videos, or voice messages, within the WhatsApp chat.

1.4. Feasibility

1.4.1. Technical Feasibility

Building this system is technically feasible. The hardware and software needed are all available, it not difficult to get them. Brief I can say the necessary resources needed for the development and maintenance of the system are available. We are going to use python version 3.7.9

1.4.2. Operationally Feasibility

The project I am developing is operationally feasible as there is no need for users to have good knowledge in computer before using it. The user can learn and use the system with easiness; he just needs to watch the interface or tutorial from the developers properly.

1.4.3. Economic Feasibility

Besides being technically feasible, developing this system is economically feasible as well. The development of the system does not require the developers to spend a lot of money. The tools I will be using to develop the system are not expensive (Some are free of cost) and the software's are open source. All I need is time. Even the maintenance of the system will not be expensive. The system is indeed economically feasible.

2.1 Introduction of Literature Review

With the increasing popularity of instant messaging applications like WhatsApp, analyzing chat data has become a valuable tool for understanding communication patterns and behaviors in group chats or one-on-one conversations. The ability to extract meaningful insights from WhatsApp chat data can provide valuable information for various purposes, such as analyzing customer feedback, studying social interactions, or monitoring group dynamics in organizational settings.

In this project, we aim to develop a WHATSAPP CHAT ANALYZER using Python, which will provide statistics and insights about WhatsApp group chats or one-on-one chats. The literature review aims to explore the existing research and literature on similar topics, including text analysis, natural language processing (NLP), and chat data analysis. By reviewing relevant literature, we aim to identify the stateof-the-art techniques, methodologies, and approaches that can be utilized in our project to achieve accurate and meaningful results.

The literature review will begin by discussing the concept of text analysis and NLP, including the techniques used for extracting meaningful information from textual data. We will explore different approaches to preprocess and clean WhatsApp chat data, such as removing irrelevant characters, converting text to lowercase, and handling emoticons and emojis. Furthermore, we will review the existing research on feature extraction techniques, including word frequency analysis, sentiment analysis, and topic modeling, which can provide valuable insights into chat data.

Next, we will review the literature on statistical analysis and visualization techniques that can be applied to WhatsApp chat data. This may include techniques for calculating word counts, identifying the most common words or phrases, and visualizing trends in chat activity over time. We will also explore approaches for identifying the most active users or participants in a chat group and understanding their communication patterns.

Additionally, we will review literature on machine learning techniques that can be used to analyze WhatsApp chat data. This may include approaches for building chat classifiers, predicting user behavior, or identifying patterns and trends in chat data using machine learning algorithms such as clustering, classification, and regression.

Finally, the literature review will discuss existing tools or software that are available for WhatsApp chat analysis and their limitations. We will identify

research gaps and potential areas for improvement in the field of WhatsApp chat analysis. By reviewing the existing literature, we aim to provide a comprehensive understanding of the state-of-the-art techniques and methodologies in WhatsApp chat analysis, which will serve as the foundation for the development of our WHATSAPP CHAT ANALYZER in Python.

2.2 Chat analyzer systems

Chat analyzer systems are tools or software that analyze chat data, such as messages or conversations, in order to extract meaningful insights and information. These systems can be applied to various types of chats, including group chats or one-on-one chats, in platforms such as WhatsApp, Facebook Messenger, or Slack. Chat analyzer systems utilize techniques from text analysis, natural language processing (NLP), and machine learning to process and analyze chat data, providing valuable statistics, trends, and patterns.

Chat analyzer systems typically perform several key functions, including:

- Data preprocessing: Chat data often requires preprocessing to clean and format the text for analysis. This may involve removing irrelevant characters, converting text to lowercase, handling emoticons or emojis, and identifying and removing any personally identifiable information (PII) to ensure privacy.
- Feature extraction: Chat analyzer systems extract relevant features from the chat data to gain insights. This may include word frequency analysis, sentiment analysis, topic modeling, or identifying important keywords or phrases.
- Statistical analysis: Chat analyzer systems use statistical techniques to analyze chat data, such as calculating word counts, identifying the most common words or phrases, visualizing trends over time, and identifying patterns in chat activity or user behavior.
- User behavior analysis: Chat analyzer systems can identify and analyze user behavior in chats, such as identifying the most active users, the frequency and timing of messages, or the interactions between users. This can provide insights into communication patterns and dynamics within the chat.
- Machine learning analysis: Chat analyzer systems may utilize machine learning algorithms to analyze chat data, such as clustering users or

messages based on similarity, classifying messages into different categories, or predicting user behavior based on historical data.

- Visualization: Chat analyzer systems often provide visual representations of the analyzed data, such as charts, graphs, or word clouds, to help users better understand and interpret the results.

Chat analyzer systems have various applications across different domains. For example, in customer service, chat analyzer systems can analyze customer chats to identify common issues or sentiment trends, allowing organizations to improve their service quality. In social research, chat analyzer systems can provide insights into social interactions, communication patterns, and language use in online communities or groups. In organizational settings, chat analyzer systems can help monitor employee communication, identify group dynamics, or analyze team collaborations.

2.3 Computerized Systems

In the modern digital era, communication has evolved significantly, and messaging apps like WhatsApp have become a popular means of communication. Analyzing WhatsApp chats can provide valuable insights, such as word usage, most common words, and most active participants. Building a "What's App Chat Analyzer" project in Python 3.7.9 can help you achieve these goals.

The project will involve several key components, including data collection, data preprocessing, data analysis, and data visualization. Let's explore each of these components in detail.

- Data Collection: The first step in building a WHATSAPP CHAT ANALYZER is to collect data from the chat. WhatsApp provides the option to export chat conversations in a text file format, which can be used as input data for the project. You can collect chat data from both group chats and one-on-one chats and store it in a text file.
- Data Preprocessing: Once the data is collected, the next step is to preprocess it to remove irrelevant information, such as timestamps, emojis, and media files. You can use Python's built-in string manipulation functions and regular expressions to clean the data and extract relevant information, such as sender name, message content, and timestamps.
- Data Analysis: After data preprocessing, you can perform various analyses to extract insights from the chat data. For example, you can calculate the

total number of messages, the number of words used, and the average word count per message. You can also identify the most active participants in the chat by counting the number of messages sent by each participant. Additionally, you can identify the most common words used in the chat by creating a word frequency distribution.

- Data Visualization: Data visualization is an essential part of the project as it helps in presenting the analysis results in a visually appealing and understandable manner. You can use Python's data visualization libraries such as Matplotlib, Seaborn, and WordCloud to create visual representations of the chat statistics. For example, you can create bar charts to show the message count of each participant, word clouds to visualize the most common words, and line charts to show the message activity over time.
- User Interface: To make the project user-friendly, you can create a simple graphical user interface (GUI) using Python libraries such as Tkinter or PyQt. The GUI can allow users to input the WhatsApp chat file, initiate data analysis, and display the results in a visually appealing format.

3.1 Research Methodology

- Problem Statement: Clearly define the problem you are trying to solve with the "WHATSAPP CHAT ANALYZER" project. This could be analyzing WhatsApp group chat or 1 on 1 chat to extract statistics such as the number of words used, most common words, most active participants, etc. Define the objectives and goals of your project.
- Literature Review: Conduct a literature review to understand the existing research and tools related to WhatsApp chat analysis or text analysis in general. This will help you identify relevant concepts, theories, and methodologies that can be applied to your project. Review and analyze relevant Python libraries for text analysis, such as pandas, numpy, nltk, etc., and identify their strengths and limitations.
- Data Collection: Describe the process of data collection for your project. This may involve obtaining WhatsApp chat data in the form of text files, exporting chats from WhatsApp application, or using third-party tools to extract chat data. Discuss the limitations and potential biases of the data you collected, such as missing data, data format inconsistencies, or potential privacy concerns.
- Data Preprocessing: Explain the steps you took to preprocess the raw data before analysis. This may include removing irrelevant information, cleaning the text data by removing special characters, converting text to lowercase, removing stop words, and tokenizing the text into words or phrases. Describe any other data preprocessing techniques you applied to ensure the quality and reliability of the data for analysis.
- Analysis Techniques: Outline the analysis techniques you employed in your project. This may include basic descriptive statistics, such as counting the number of words, characters, and messages, and calculating the average word length. You may also conduct more advanced analyses, such as sentiment analysis, frequency distribution, and word cloud generation, to extract meaningful insights from the chat data. Describe the Python libraries or tools you used for these analyses and explain how they were applied.
- Results and Interpretation: Present the results of your analysis in the form of tables, charts, or graphs, and interpret the findings. This may include identifying the most common words used, the most active participants, the average message length, and other relevant statistics. Discuss any

interesting patterns or trends you observed in the chat data and relate them to the objectives and goals of your project.

- Conclusion: Summarize the key findings of your project and draw conclusions based on the results. Discuss the implications of your findings and their potential applications. Highlight the significance and contributions of your research to the field of text analysis or chat analysis.
- References: Provide a list of all the sources, including research papers, books, websites, and Python libraries that you referenced in your project. Follow the appropriate citation format, such as APA or IEEE, for consistency and credibility.
- Appendix: Include any additional information, such as sample code, screenshots, or data files that are relevant to your project but not included in the main text. This will help readers better understand the details of your project and replicate your research if needed.

In summary, the research methodology for the "WhatsApp chat analyzer" Project in Python 3.7.9 involves clearly defining the problem statement, conducting a literature review, collecting and preprocessing data, applying appropriate analysis techniques, validating the results, and drawing conclusions based on the findings. Providing references and appendix will enhance the credibility and replicability of your research.

3.2 Project Activates

The "What's App Chat Analyzer" project in Python version 3.7.9 is designed to provide insightful statistics and information about a WhatsApp group chat or a 1on-1 chat. The project activates by taking the chat data as input and processing it to generate various statistics. It calculates the total number of words used in the chat, identifies the most common word used, and determines the most active participant in the chat based on their message count. Additionally, the project may provide information such as average message length, most used emojis, and word cloud representation of frequently used words. The "What's App Chat Analyzer" project aims to offer valuable insights and analytics for better understanding and analysis of WhatsApp conversations.

3.2.1 Planning

- Data Collection: The first step in the project will be to collect the WhatsApp chat data. This can be done by exporting the chat history from a WhatsApp group or one-on-one chat in text format. The exported text file will serve as the input data for the project.
- Data Preprocessing: Once the chat data is collected, it needs to be preprocessed to prepare it for analysis. This may involve cleaning the data by removing unnecessary information such as timestamps, emojis, and media files. The data may also need to be converted into a structured format, such as a CSV file, for easy manipulation in Python.
- Text Analysis: The next step will be to perform various text analysis tasks on the preprocessed data. This may include calculating the total number of words used in the chat, finding the most common words, determining the most active participants, and analyzing the sentiment of the messages. Python libraries such as Pandas, NLTK, and TextBlob can be used for these tasks.
- Visualization: After performing the text analysis, the project will need to visualize the results to make them more understandable and accessible. Python libraries such as Matplotlib and Seaborn can be used to create various plots, charts, and graphs to visualize the statistical insights obtained from the chat data.
- User Interface: To make the project more user-friendly, a graphical user interface (GUI) can be developed using Python libraries such as Tkinter or PyQt. The GUI will allow users to interact with the project, input the chat data, and view the statistical insights in a visually appealing manner.
- Testing and Debugging: Once the project is implemented, it needs to be thoroughly tested to ensure its functionality and accuracy. Any issues or bugs that are identified during testing should be fixed to ensure that the project produces reliable and accurate results.
- Documentation: Finally, the project should be documented to provide instructions on how to use it, explain the implementation details, and highlight the key features and functionalities. The documentation can be in the form of comments in the code, a user manual, or a technical report.

3.2.2 Requirement analysis

The following are the key requirements identified for the "What's App Chat Analyzer" project:

- Input Data: The project should be able to accept input data in the form of WhatsApp chat logs. The chat logs can be in text format, either exported from the WhatsApp application or extracted from the WhatsApp API.
- Data Processing: The project should be able to process the input data to extract relevant information such as text messages, sender information, timestamps, and other relevant details.
- Word Count: The project should be able to calculate the total number of words used in the chat logs. This can include both individual words and word combinations (e.g., "hello" and "good morning").
- Most Common Words: The project should be able to identify and display the most common words used in the chat logs. This can help users understand the frequently used words and phrases in the chat.
- User Activity Analysis: The project should be able to analyze the activity level of users in the chat, including the total number of messages sent, the most active users, and the distribution of messages among users.
- Visualization: The project should provide visualizations, such as bar charts or word clouds, to present the analyzed data in a meaningful and user-friendly way.
- User Interaction: The project should have a user-friendly interface that allows users to interact with the system, input chat logs, and view the analysis results.
- Python 3.7.9 Compatibility: The project should be compatible with Python version 3.7.9, including utilizing the relevant libraries and packages available in this version.
- Code Maintainability: The project should be well-organized, modular, and properly documented to ensure code maintainability and ease of future enhancements.

3.2.3 Design

The project can be broken down into the following key components:

- Data Preprocessing: Once the chat data is loaded into Python, it needs to be preprocessed to clean and structure the data for further analysis. This may involve tasks such as removing unnecessary characters, converting text to lowercase, and tokenizing the text into words or sentences. Python libraries such as pandas and NLTK can be used for these tasks.
- Statistical Analysis: After preprocessing the data, various statistical analysis techniques can be applied to extract insights. For example, the total number of words used in the chat can be calculated by counting the number of tokens. The most common words can be identified by performing word frequency analysis. The most active participants in the chat can be determined by counting the number of messages sent by each participant. Python libraries such as matplotlib and nltk can be used for these analyses.
- Data Visualization: The results of the statistical analysis can be visualized using graphical plots and charts. For example, a bar chart can be used to display the word frequency distribution, and a pie chart can be used to show the proportion of messages sent by each participant. Python libraries such as matplotlib and seaborn can be used to create visually appealing and informative plots.
- User Interface: To make the project user-friendly, a simple user interface can be created using Python libraries such as ipywidgets or tkinter. The user interface can allow users to interact with the project, input the chat data, specify the analysis tasks to be performed, and display the results in an organized and visually appealing manner.
- Error Handling: Appropriate error handling techniques should be implemented to handle unexpected situations, such as missing or incorrect data, to ensure the smooth running of the project. This may involve checking for data integrity, handling exceptions, and providing informative error messages to users.
- Documentation: Finally, the project should be properly documented with comments in the code and a README file that includes instructions on how to use the project, dependencies, and any other relevant information.

3.2.4 Implementation

Data Collection:

- Obtain the chat data from WhatsApp. This can be done by exporting the chat as a text file or by using a third-party library to scrape the chat data from the WhatsApp web interface.
- Load the chat data into a pandas DataFrame for further processing and analysis.

Data Cleaning:

- Clean the chat data by removing irrelevant information such as timestamps, emojis, and system messages.
- Extract relevant information such as sender names, message content, and message timestamps from the chat data.

Data Analysis:

- Perform various statistical and exploratory analyses on the chat data to obtain insights such as the total number of messages, the most active sender, the most common words used, and other relevant statistics.
- Use visualization libraries such as Matplotlib or Seaborn to create visual representations of the chat data, such as bar charts or word clouds, to provide a more intuitive understanding of the data.

User Interface (UI) Development:

- Develop a user interface using a library such as Tkinter, PyQt, or Dash to allow users to interact with the chat analyzer tool.
- Design the UI to display the analyzed statistics in a user-friendly and visually appealing manner, such as displaying the most common words as a word cloud or presenting the statistics in a tabular format.

Integration:

- Integrate the data analysis code with the UI to allow users to input their chat data and trigger the analysis.
- Implement data processing and analysis functions as separate modules or classes that can be easily imported and used in the main UI code.

Testing:

- Test the chat analyzer tool thoroughly to ensure that it produces accurate and reliable results.
- Create test cases and test scenarios to validate the functionality and performance of the tool.
- Fix any bugs or issues discovered during testing to ensure the tool's accuracy and reliability.

Deployment:

- Package the chat analyzer tool as an executable file or a web application, depending on the chosen UI framework.
- Create an installer or package the tool in a Docker container for easy deployment on different platforms.
- Document the deployment process to make it easier for users to install and use the tool.

Documentation:

- Create documentation that includes a user manual, developer guide, and any other relevant documentation.
- Include instructions on how to use the tool, input requirements, and expected output.
- Provide examples and sample output to help users understand the tool's functionality and how to interpret the results.

3.2.5 Testing

What's App Chat Analyzer" is a project that aims to provide statistics about a WhatsApp group chat or 1 on 1 chat. It analyzes the text data from WhatsApp chats and generates insights such as the number of words used, the most common words, the most active participants, and more. As with any software project, testing is a critical step to ensure that the code performs as expected and delivers accurate results.

To ensure the accuracy and reliability of the "What's App Chat Analyzer" project, several testing techniques can be employed. These techniques include:

- **Unit Testing:** Unit testing involves testing individual functions or methods within the code to check their correctness. This can be done using Python's built-in unittest library or other popular testing frameworks such as pytest or nose. Each function or method should be tested with different inputs, including edge cases and boundary values, to verify that it produces the expected output.
- **Integration Testing:** Integration testing involves testing the interactions between different components or modules of the code. In the case of the "What's App Chat Analyzer" project, this may include testing the interactions between functions or methods that handle parsing and analyzing chat data, as well as any external libraries or APIs used in the project.
- **Data Validation:** Since the "WhatsApp Chat Analyzer" project involves analyzing text data from WhatsApp chats, it is essential to validate the data input to the system. It includes checking for invalid characters, handling missing or incomplete data, and verifying that the data is in the expected.
- **Performance Testing:** Performance testing involves testing the efficiency and scalability of the code. For instance, if the "What's App Chat Analyzer" project deals with large chat data sets, it is important to ensure that the code can handle the data size efficiently without any performance bottlenecks.
- **Error Handling:** Error handling is a critical aspect of testing. The code should be tested to handle various types of errors, such as invalid inputs, unexpected behavior, and exceptions. Proper error handling mechanisms should be in place to ensure that the project does not crash or produce incorrect results when faced with unexpected scenarios.
- **Usability Testing:** Usability testing involves testing the user interface (UI) and overall user experience (UX) of the "What's App Chat Analyzer" project. This includes verifying that the UI is intuitive, user-friendly, and provides meaningful feedback to the user. Usability testing can be done through manual testing by interacting with the project's UI and providing feedback on its functionality and ease of use.

Testing is a crucial step in the development process of the "What's App Chat Analyzer" project to ensure its accuracy, reliability, and performance. By employing various testing techniques such as unit testing, integration testing, data validation, performance testing, error handling, and usability testing, the project can deliver accurate and meaningful statistics about WhatsApp chats.

3.3 Tools

Tools are important in various domains and industries because they enable tasks to be performed more efficiently, effectively, and accurately. In the context of web development, tools play a crucial role in simplifying complex tasks, improving productivity, and enhancing the quality of work. One key aspect of web development is code editing and management. Text editors or integrated development environments (IDEs) are essential tools that provide features such as syntax highlighting, auto completion, code navigation, and debugging, which help developers write and manage code more effectively. These tools allow developers to catch errors early, streamline coding processes, and increase coding speed.

Another important aspect of web development is testing and debugging. Tools for testing web applications, such as browsers and testing frameworks, are critical in identifying and fixing issues, ensuring that web applications function properly across different devices and browsers. Debugging tools, such as browser developer tools, allow developers to inspect and troubleshoot code in real-time, making it easier to identify and resolve issues.

Firstly, Python 3.7.9 will serve as the programming language for writing your chat analyzer script. Jupyter Notebook can be used as an interactive coding environment for creating a notebook where you can write and run your code, as well as document your findings. Pandas, a powerful data manipulation library in Python, can assist with reading and parsing the WhatsApp chat data, and performing data manipulation tasks. Additionally, Matplotlib and Seaborn can be employed for data visualization purposes, enabling you to create visually appealing plots and charts to showcase your statistical findings.The various tools used for the project are given below:-

3.3.1 Python version 3.7.9

Python 3.7.9 is a version of the Python programming language, which is widely used for various applications such as web development, data analysis, machine learning, and scientific computing. Python 3.7.9 is part of the Python 3.x series, which is the latest major version of Python, and it includes numerous features and improvements over the previous Python 2.x series.

Python 3.7.9 comes with a rich ecosystem of libraries and modules that enable developers to build complex applications with ease. It provides support for modern programming paradigms such as object-oriented, functional, and procedural programming. Python 3.7.9 also includes extensive standard libraries for tasks such as file I/O, regular expressions, networking, and more.

One of the notable features of Python 3.7.9 is its focus on improving performance and efficiency. It includes optimizations that result in faster code execution and reduced memory usage. Additionally, Python 3.7.9 introduces new syntax enhancements, security improvements, and bug fixes.

3.3.2 Jupyter Notebook

Fig 3.1: Jupyter

Jupyter Notebook is a popular open-source tool that allows you to create and run code cells in an interactive environment. It is commonly used for data analysis and scientific computing tasks in Python. You can install Jupyter Notebook using pip, the Python package manager, or by using an Anaconda distribution, which comes with Jupyter pre-installed.

3.3.3 pytest

Fig 3.2: Pytest

Pytest is a popular testing framework for Python that allows developers to write and execute tests for their Python code. It provides a concise and expressive syntax for writing tests, making it easy to create and maintain test cases. Pytest offers a rich set of features, including powerful assertion methods, built-in test discovery, fixtures for managing test dependencies, and plugins for extending its functionality. With Pytest, you can write tests as simple functions, and it automatically discovers and runs them, generating informative reports with test results. Pytest also integrates well with other Python tools and libraries, making it a preferred choice for many developers for writing efficient and scalable tests for their Python projects.

3.4 Software Requirement Specification

1. Functional Requirements: The "What's App Chat Analyzer" project will have the following functional requirements:

- Chat Data Import: The tool will allow users to import chat data from WhatsApp groups or 1-on-1 chats. The chat data can be in the form of text files or CSV files.
- Data Preprocessing: The tool will preprocess the imported chat data, including removing irrelevant information such as date/time stamps, system messages, and other non-chat related content.
- Word Count Analysis: The tool will perform word count analysis on the preprocessed chat data to determine the total number of words used in the chat.

- Most Common Word Analysis: The tool will identify the most common word used in the chat data based on the word count analysis.
- Participant Activity Analysis: The tool will analyze the activity of each participant in the chat, including determining the most active participant based on the number of messages sent, words used, and other relevant metrics.
- Chat Timeline Analysis: The tool will generate a timeline of the chat activity, showing the frequency of messages and words used over time.
- Output and Visualization: The tool will provide visualizations and reports summarizing the results of the chat analysis, including graphs, charts, and tables.

2. Non-functional Requirements: The "What's App Chat Analyzer" project will have the following non-functional requirements:

- User Interface: The tool will have a user-friendly interface that allows users to easily import chat data, configure analysis options, and view the results.
- Performance: The tool should be able to handle large chat data sets efficiently and provide quick analysis results.
- Accuracy: The analysis performed by the tool should be accurate and reliable, providing meaningful insights into the chat data.
- Security: The tool should ensure the confidentiality and integrity of the chat data, and protect against unauthorized access or data breaches.

3. Development Environment: The "What's App Chat Analyzer" project will be developed using the following technologies:

- Programming Language: Python version 3.7.9 will be used as the primary programming language for the project.
- Integrated Development Environment (IDE): Jupyter Notebook will be used as the IDE for developing the tool.

4.1 Result / outputs

4.1.1 Chat Export

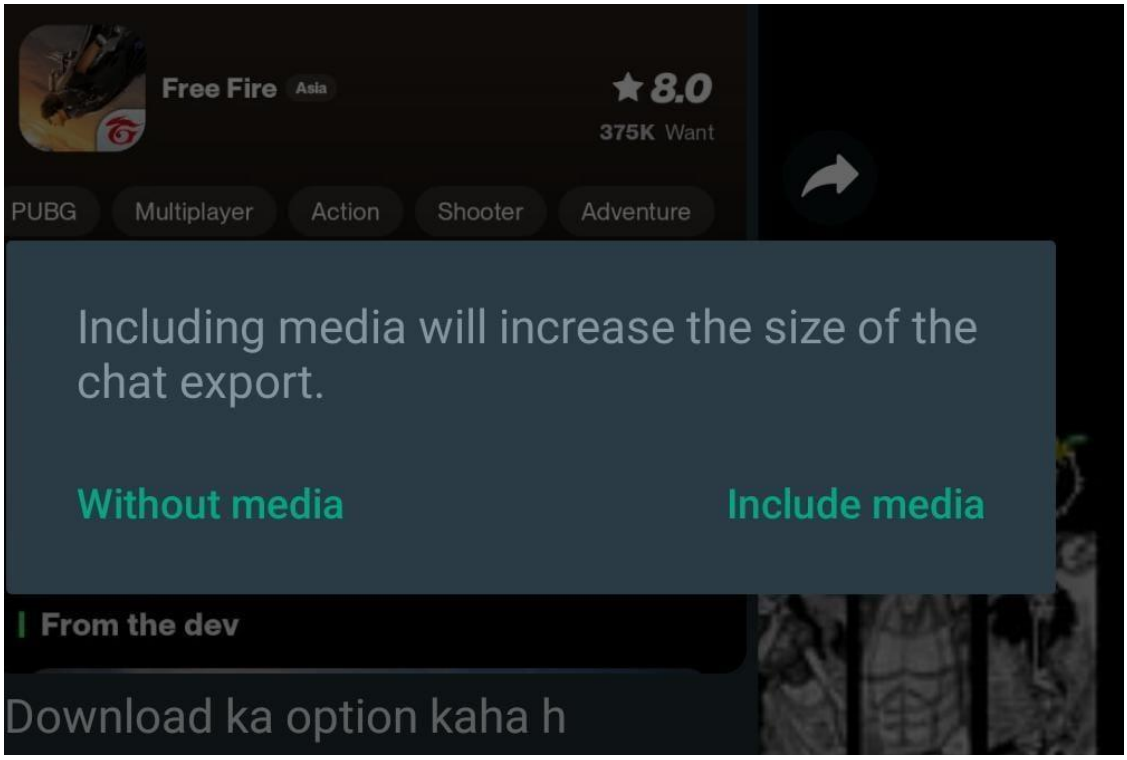

Fig 4.1: Chat Export

4.1.2 Website Interface

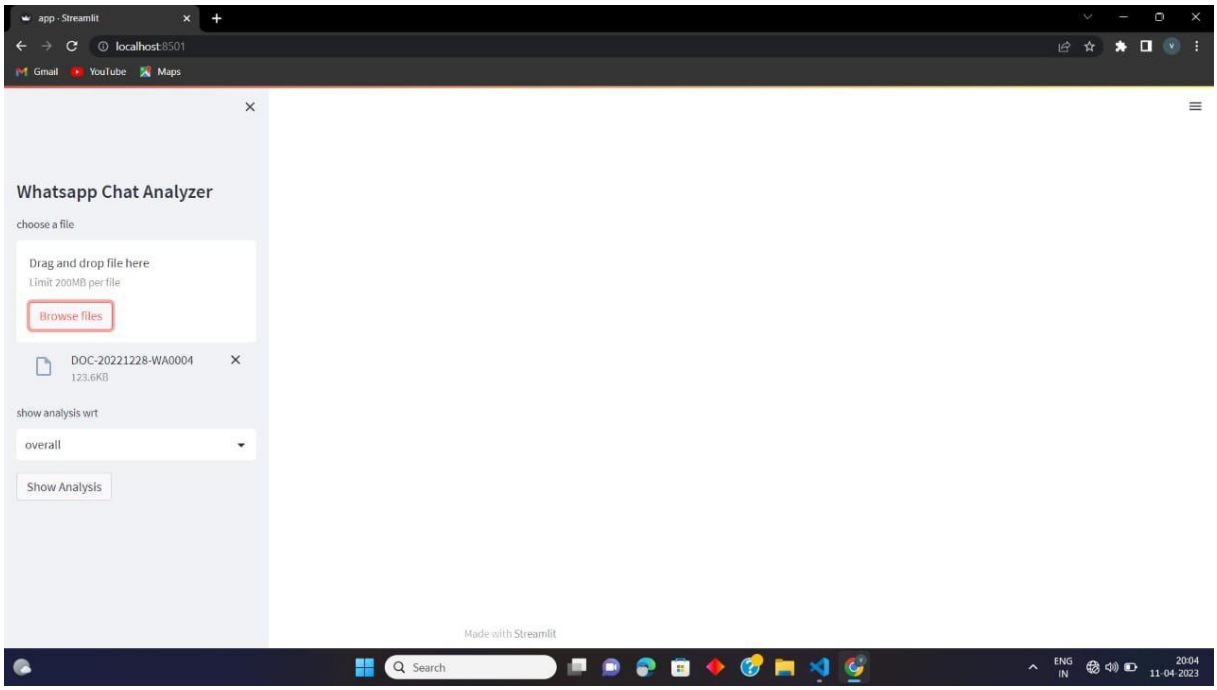

Fig 4.2 Website Interface

4.1.3 Top statistics

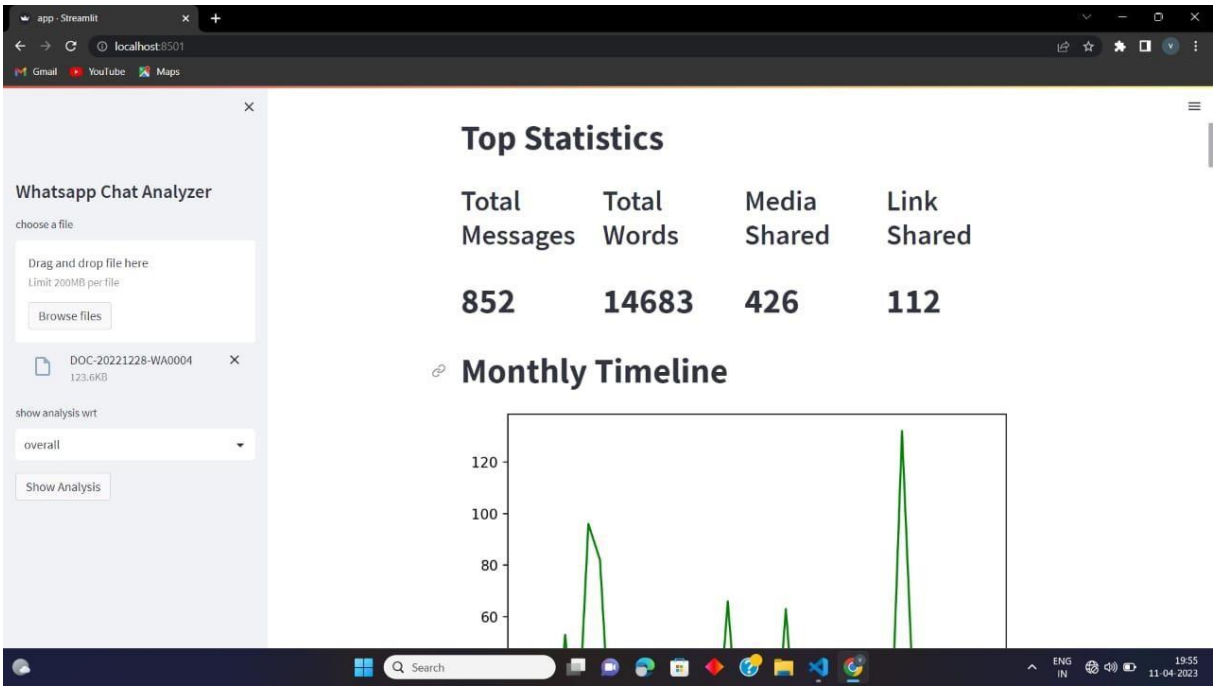

Fig 4.3: Top statistics

4.1.4 One out of many other data analysis

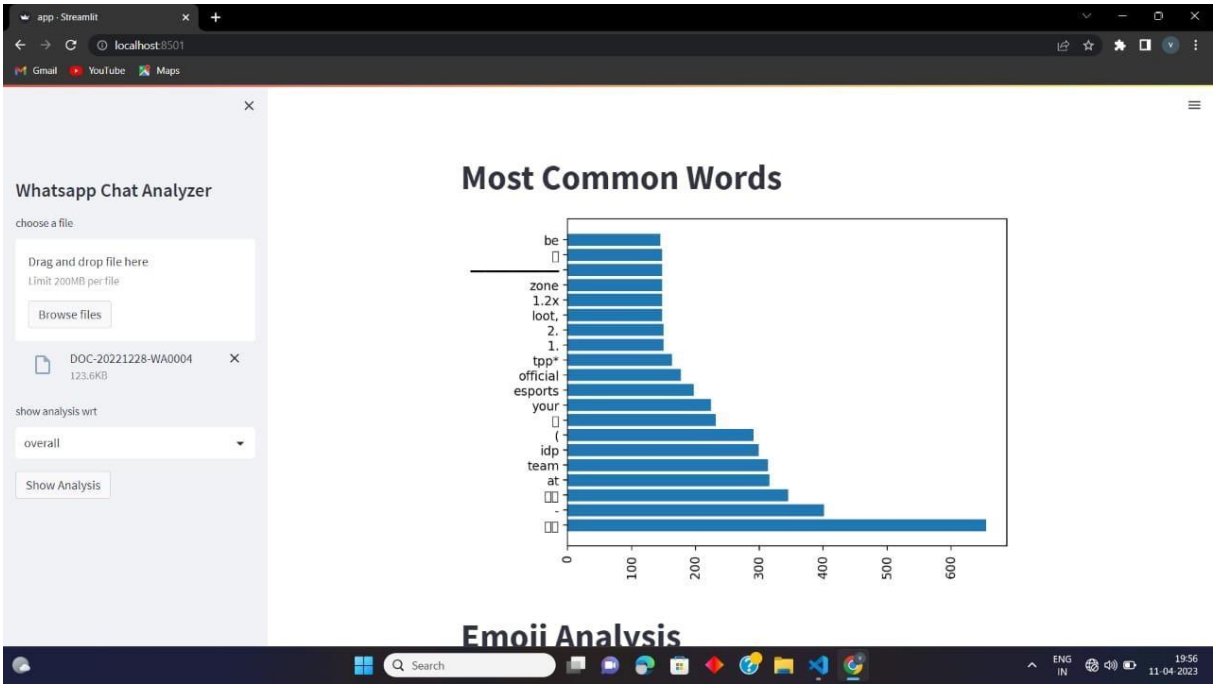

Fig 4.4: one of many other data Analysis

5.1 Test cases

Test Case 1: File Upload

Description: Verify that the system allows the user to upload a WhatsApp chat file in a supported format (e.g., .txt or .csv).

Input: Upload a valid WhatsApp chat file.

Output: Confirm that the system successfully reads and imports the chat data from the uploaded file.

Test Case 2: Data Pre-processing

Description: Validate that the system performs data pre-processing tasks accurately, such as cleaning and formatting the chat data.

Input: Provide a chat file with various types of messages, including text, emojis, images, and attachments.

Output: Verify that the system cleans the data, removes irrelevant information (e.g., timestamps, system messages), and formats the data into a structured format for analysis.

Test Case 3: Basic Analysis

Description: Test the basic analysis functions of the system, such as word frequency analysis, most active member, and message count by participant.

Input: Provide a sample chat file with a small number of messages and participants.

Output: Confirm that the system accurately calculates and displays the word frequency, most active member, and message count by participant, based on the provided chat data.

Test Case 4: Advanced Analysis

Description: Test the advanced analysis functions of the system, such as sentiment analysis, named entity recognition, and topic modeling.

Input: Provide a chat file with a diverse range of messages and participants, including different types of conversations (e.g., personal, professional).

Output: Verify that the system accurately performs advanced analysis tasks, such as sentiment analysis to determine the sentiment of messages, named entity recognition to identify entities like names, organizations, and locations, and topic modeling to identify common topics discussed in the chat.

Test Case 5: Visualization

Description: Test the visualization capabilities of the system, including the generation of plots, graphs, and charts.

Input: Provide a chat file with a sufficient amount of data to generate meaningful visualizations.

Output: Confirm that the system generates accurate and informative visualizations, such as bar charts, pie charts, word clouds, and network graphs, to visually represent the analyzed chat data.

Test Case 6: Error Handling

Description: Test the error handling capabilities of the system, including handling invalid file formats, unsupported characters, or unexpected errors.

Input: Upload an invalid or unsupported file format, provide messages with unsupported characters, or trigger unexpected errors during analysis.

Output: Verify that the system displays appropriate error messages and gracefully handles errors, ensuring the smooth functioning and stability of the application.

Test Case 7: User Authentication and Privacy

Description: Test the user authentication and privacy features of the system, if applicable.

Input: Provide a username and password for authentication.

Output: Confirm that the system authenticates the user credentials securely and restricts access to the chat data based on user permissions, ensuring data privacy and security.

6.1 Conclusion

In conclusion, the WhatsApp Chat Analyzer project aims to provide a comprehensive analysis of WhatsApp chat data using Python, Streamlit, and various libraries such as Pandas, Seaborn, Matplotlib, and Emoji. The project allows users to upload WhatsApp chat files, preprocesses the data, performs analysis tasks such as word frequency analysis, sentiment analysis, named entity recognition, topic modeling, and generates visualizations to gain insights from the chat data.

The project has been developed with the objective of helping users gain a better understanding of their WhatsApp chats by uncovering hidden patterns, trends, and insights from the data.

The project also has future scope for enhancement, such as incorporating more advanced natural language processing techniques, expanding the types of visualizations, adding user authentication and privacy features, and improving error handling to make the application more robust and user-friendly.

In conclusion, the WhatsApp Chat Analyzer project offers a powerful tool for analyzing WhatsApp chat data, providing users with valuable insights and a deeper understanding of their conversations. By leveraging the power of Python, Streamlit, and various libraries, the project can aid users in gaining meaningful insights from their WhatsApp chats, making it a valuable tool for data-driven decision making, communication analysis, and understanding the dynamics of group conversations.

6.2 Recommendations

- **Expand Chat File Format Support:** Currently, the project supports .txt and .csv file formats for chat data. Consider adding support for other common file formats used in WhatsApp chats, such as .json or .html, to accommodate different export options from WhatsApp and improve the versatility of the application.
- **Improve Data Pre-processing:** Enhance the data pre-processing step to handle various types of messages, emojis, attachments, and multimedia content more effectively. Consider incorporating advanced text preprocessing techniques, such as handling special characters, removing stop words, and normalizing text, to improve the accuracy of analysis

- **Enhance Analysis Techniques:** Consider incorporating more advanced natural language processing (NLP) techniques, such as sentiment analysis, topic modeling, and language identification, to provide deeper insights into the chat data. These advanced analysis techniques can provide more meaningful and actionable information to users for better understanding and interpretation of chat data.
- **Enhance Visualization Options:** Expand the visualization options to include a wider variety of plots, graphs, and charts to provide more diverse and informative visual representations of the analyzed chat data. Consider incorporating interactive visualizations, such as interactive word clouds or network graphs, to enhance the user experience and make the analysis results more engaging and interactive.
- **Error Handling and User-Friendly Feedback:** Improve the error handling capabilities of the system to provide more user-friendly error messages and feedback. Clearly communicate any limitations, assumptions, or potential issues with the analysis results to users to ensure transparency and avoid misinterpretation of the data.
- **Performance Optimization:** Optimize the performance of the application to handle large chat files efficiently and provide faster analysis results. Consider optimizing the data processing and analysis algorithms, leveraging parallel processing or distributed computing techniques, or implementing caching mechanisms to enhance the overall performance and responsiveness of the application.
- **User Authentication and Privacy:** Consider implementing user authentication features to ensure data privacy and security. Provide options for users to securely login and access their chat data, and ensure that sensitive information, such as usernames and passwords, are properly encrypted and stored. Follow best practices for data privacy and security, such as anonymizing data and adhering to relevant regulations, such as GDPR or HIPAA, if applicable.
- **Documentation and Help Features:** Provide comprehensive documentation and help features within the application to guide users on how to use the features effectively, interpret the results, and troubleshoot issues. Consider including tooltips, tutorials, or FAQs to assist users in making the most out of the WhatsApp Chat Analyzer.

References

[1] Ravishankara K, Dhanush, Vaisakh, Srajan I S, "International Journal of Engineering Research & Technology (IJERT)", ISSN: 2278-0181, Vol. 9 Issue 05, May-2020.

[2] https://www.analyticsvidhya.com/blog/2021/06/build-web-appinstantlyfor-machine-learningusing-streamlit.

[3] Dr. Vaibhav Pathanker

Meng cai, "PubMed Central", PMCID: PMC7944036, PMID: 33732917.

[4] Dr. D. Lakshminarayanan, S. Prabhakaran, "Dogo Rangsang Research Journal", UGC Care Group I Journal, Vol-10 Issue-07 No. 12 July 2020. [5] https://www.interaction-design.org/literature/topics/web-design