PROJECT CLAY

MID TERM REPORT

OF MAJOR PROJECT

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

PANKAJ RAMOLA — 2115097

NIKHIL BIDLAN — 2115110

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA

ABSTRACT

The most used and efficient method of communication in recent times is an application called WhatsApp.WhatsApp chats consist of various kinds of conversation held among two people or a group of people.

This chatconsists of various topics. This information can provide a lot of data for the latest technologies such as MachineLearning.

The most important things for Machine Learning models are to provide the right learning experiencewhich is indirectly affected by the data we provide to the model. This tool aims to provide in depth analysis ofthe data which is provided by WhatsApp.

Irrespective of whichever topic the conversation is based on, ourdeveloped code can be applied to obtain a better understanding of the data. The advantage of this tool is that it isimplemented using simple python modules such as pandas, matplotlib, seaborn, streamlit, numpy, re, emojis anda technique sentiment analysis which are used to create data frames and plot different graphs, where then it isdisplayed in the streamlit web application which is efficient and less resources consuming algorithms, thereforeit can be easily applied to larger dataset.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background of the Project

The WhatsApp Chat Analyzer project addresses the need for analyzing communication patterns in WhatsApp, a widely used messaging platform. By processing exported chat data, the tool provides insights into message frequency, participant engagement, and response times. This helps users understand behavioral trends, optimize productivity, and monitor interactions through an intuitive, data-driven approach.

1.1.1 Subsection Title

The WhatsApp Chat Analyzer project is driven by the increasing reliance on WhatsApp for communication in both personal and professional contexts. While WhatsApp provides basic chat statistics, there is a growing need for more comprehensive insights into communication patterns. By enabling users to analyze message frequency, participant contributions, and response times, this project aims to unlock the potential of chat data. These insights can be valuable for improving productivity, understanding social dynamics, or conducting behavioral studies, making the analyzer a versatile and practical tool.

1.2 Problem Statement

WhatsApp-Analyzer is a statistical analysis tool for WhatsApp chats. Working on the chat files that can be exported from WhatsApp it generates various plots showing, for example, which other participant a user responds to the most. Communication between people using the internet becomes part of their daily life. People used to communicate with each other using the online chat system to transfer their messages. We propose to employ dataset manipulation techniques to have a better understanding of WhatsApp chat present in our phones.It shows most used emoji and word which repeatedly most times. It tracks our conversation and analyzes how much time we are spending.

1.3 Objectives

The WhatsApp Chat Analyzer project aims to analyze communication patterns, enhance productivity insights, and understand social dynamics by extracting key metrics like message frequency, participant engagement, and response times. It also provides a user-friendly interface to transform raw chat data into actionable insights for individuals and researchers.

1.4 Scope of the Project

The WhatsApp Chat Analyzer project focuses on providing meaningful insights from exported WhatsApp chat data. It supports analysis of communication patterns, including message frequency, participant activity, and response times. The tool is designed for personal use, team productivity monitoring, and research purposes, with an intuitive interface for data visualization. The project excludes real-time chat analysis and ensures user privacy by processing only locally stored data.

Chapter 2

Literature Review

2.1 Review Table

S. No.	Title of Study/Tool	Authors/Developers	Year	Key Findings/Features	Limitations
1	WhatsAnalyzer	Third-party Developers	2019	Provides basic stats like total messages and emojis used.	Limited analysis capabilities and no advanced visualization.
2	A Study on WhatsApp Usage Trends	Smith et al.	2020	Explored behavioral patterns in WhatsApp usage across demographics.	Focused only on trends, not productivity analysis.
3	Chatilyzer	Independent Researchers	2021	Analyzed user engagement and topic frequency in chats.	Required manual tagging; lacked dynamic filtering.
4	WhatsApp Chat Mining Using Python	University Researchers	2022	Used Python for sentiment and activity analysis of WhatsApp chats.	Complex for non-technical users; no GUI.
5	Behavioral Patterns in Messaging Apps	Patel et al.	2023	Studied response times and engagement on WhatsApp and other apps.	Generalized findings; no dedicated WhatsApp tool.

Figure 2.1: Review Table

2.2 Existing System

- Chat Stats
- Whatsanalyze
- Chatilyzer
- Chat analyzer

2.2.1 Technical Feasibility

The technical feasibility study reports whether there exists correct required resources and technologies whichwill be used for project development. It is the measure of the specific technical solution and the availability ofthe technical resources and expertise. In our project we will be using Jupyter notebook(web based application)and VS code(text editor), both of them are open source softwares.Along with these various python libraries andwill be used.

2.2.2 Economical Feasibility

Cost and benefit of the project is analyzed in economic feasibility, that means what will be the cost of finaldevelopment of the product. This project has no cost in development since all the software and technologiesused are open source. This project is not economical as it mainly depends on the analysis of data betweentwo/more devices(phones).

2.2.3 Operational Feasibility

It is to determine whether the system will be used after the development and implementation.In OperationalFeasibility degree of providing service to requirements is analyzed .This involves the study of utilization andperformance of the product. Our project shows the whole analysis of the chats among people. It can be twopeople or a group of people and provides various information using charts in easily readable format.

Chapter 3

Proposed System

3.1 REQUIREMENT ANALYSIS

Requirements Analysis is the process of defining the expectations of the users for an application that is to bebuilt or modified. Requirement's analysis involves all the tasks that are conducted to identify the needs ofdifferent stakeholders.

3.2 PLATFORM SPECIFICATION

The Platform Initialization Specification (PI Specification) is a specification published by the Unified EFIForum that describes the internal interfaces between different parts of computer platform firmware. This allowsfor more interoperability between firmware components from different sources.

3.3 Functional Requirements

The software is meant to accept a user valid id which will provide unique identity to individual user. It isthrough this user id that each user data can be accessed on this platform. The requirements under proposedsystem are to maintain information relevant to the following fields: -

● User Profile- The full information of each and every student must be maintained in the System alongwith the facility to regularly update it from time to time at regular intervals which will be easily possiblethrough each user's unique id.

● Record of Results- This phase will maintain information about users' performance track record. All theresults of users will be kept in record.

● Providing test reports- It is meant to analyse each candidate's performance on the individual subject.

3.4 Non-Functional Requirements

Performance Requirements
● It uses python, to enhance the user experience.
● High data transfer rate.
● Reduces server response time.
● Enable browser Caching to save the webpage on clients machine.
● It must remove outdated content from its memory in order to be light and efficient on web server
Safety Requirements
● System use shall not cause any harm to users.
● Must be able to tackle threats from outer internet.

Security Requirements

System will use secured data.Normal users can just read information but they cannot edit or modify their personal and some other information.System will have different types of users and every user has access constraints.

Software Quality Attributes

● **Availability:**
It is available 24 Hours worldwide across the globe on any web enabled device that has internet connection
● **Portability:**
It is portable since it is a web application and can be accessed from any browser.
● **Reusability:**
It is reusable since the user only has to upload a new text file every time he/she uses this application.

3.5 System Specification

A System Requirements Specification is a structured collection of information that embodies the requirementsof a system.

3.5.1 Hardware specification

●Describe the logical and physical characteristics of each interface between our software and thehardware components of the system.
● **Hardware Required:** Any web browser supported device.
● **Supported device types**: The software is developed for Windows 32-bit/64-bit or android etc.
● **Nature of the data and control interactions between the software and the hardware :**
Internet connection

3.5.2 Software Specification

The connections of your software with other operating systems: the software is developed for all operating system.

The connections of your software with other libraries:
●Streamlit
●Numpy
●Pandas
●Wordcloud
●Plost
●Pathlib
●Collection
●Matplotlib

Chapter 4

Implementation

This chapter outlines the implementation process of the *WhatsApp Chat Analyzer* project, detailing the steps taken to develop the tool and the technologies used. The following sub-chapters cover different aspects of the implementation.

4.1 System Architecture

This section describes the overall architecture of the system, including the components and their interactions. It explains how the data flows from the input (WhatsApp chat files) to the output (insights and visualizations).

4.2 Data Preprocessing

In this sub-chapter, the steps taken to clean and prepare the WhatsApp chat data for analysis are discussed. It includes text parsing, removing unnecessary data (e.g., media links), and standardizing timestamps.

4.3 Analytical Model

Here, the methods and algorithms used to analyze the chat data are explained. This includes how message frequency, user engagement, and response times are calculated, as well as how trends are identified.

4.4 User Interface Design

This section covers the design of the user interface (UI), focusing on the tools used to create an intuitive and accessible experience for users. It explains the layout, functionality, and visual elements of the tool.

4.5 Testing and Validation

The testing process is outlined in this sub-chapter, detailing how the system was tested for accuracy, reliability, and performance. It also covers any validation steps taken to ensure the results are meaningful and useful for end-users.

These sub-chapters provide a comprehensive view of the steps involved in the development and implementation of the WhatsApp Chat Analyzer.

4.6 Data Visualization

This section focuses on the visualization techniques used to present the analyzed data. It explains the charts, graphs, and other visual elements employed to make the results easily interpretable, such as bar charts for message frequency or pie charts for participant engagement.

4.7 Integration with External Tools

Here, the integration with external tools or libraries, such as natural language processing (NLP) libraries for sentiment analysis or visualization libraries like Matplotlib or Plotly, is discussed. This section describes how these integrations enhance the functionality of the project.

4.8 Performance Optimization

This sub-chapter addresses the strategies used to improve the performance of the tool, ensuring that large chat files can be processed quickly. It includes optimizations related to memory management, algorithm efficiency, and parallel processing.

4.9 Security and Privacy Considerations

This section explains how user data privacy is maintained throughout the analysis process. It covers data encryption, storage practices, and any measures taken to protect sensitive information within the WhatsApp chats.

4.10 Deployment and Maintenance

This final sub-chapter describes the deployment process of the WhatsApp Chat Analyzer project, including how it was made accessible to users. It also discusses future maintenance plans and how the tool will be updated with new features or improvements over time.

These additional sub-chapters further break down the implementation of the WhatsApp Chat Analyzer, addressing key technical and operational aspects of the project.

Chapter 5

Results an Analysis

This chapter presents the findings and analyses from the *WhatsApp Chat Analyzer* project, showcasing the insights derived from the chat data and evaluating the effectiveness of the tool.

5.1 Overview of Data Sets

This section provides an overview of the chat data used in the analysis, including sample sizes, participant demographics, and the structure of the data extracted from WhatsApp chat files.

5.2 Message Frequency Analysis

Here, the results of analyzing message frequency across different participants and time periods are presented. This sub-chapter explores patterns in communication activity, such as peak times and average message rates.

5.3 Participant Engagement

This section examines user participation in chats, highlighting the most active participants, message contributions, and how engagement varies across different groups or time frames.

5.4 Response Time Analysis

The analysis of average response times between messages is presented here, discussing factors influencing response time, such as time of day, message length, and participant involvement.

5.5 Sentiment Analysis

In this sub-chapter, the sentiment analysis results are discussed, showcasing how the tone of conversations (positive, negative, or neutral) was analyzed using natural language processing techniques.

5.6 Word Cloud Generation

This section explores the generation of word clouds from the chat data, showing the most frequently mentioned words or phrases and their significance in the conversations.

5.7 Interaction Trends

Here, trends in participant interactions over time are examined, highlighting shifts in communication patterns, such as periods of intense discussions or reduced activity.

5.8 Comparative Analysis Across Groups

This sub-chapter compares the results of different groups or chat types (e.g., personal vs. group chats), analyzing differences in message volume, response times, and participant engagement.

5.9 Visualization of Data Insights

This section focuses on the various visualizations created to display the analyzed data, such as bar charts, pie charts, and heat maps, and discusses how these help interpret the findings.

5.10 Evaluation of Tool Effectiveness

In this final sub-chapter, the overall effectiveness of the WhatsApp Chat Analyzer tool is evaluated, including its accuracy, usability, and how well it meets the project's objectives.

These sub-chapters provide a thorough breakdown of the results and analyses from the WhatsApp Chat Analyzer, offering insights into communication patterns and evaluating the tool's performance.

Chapter 6

Conclusion and Recommendations

This chapter provides a summary of the project's outcomes and offers suggestions for future improvements and applications of the WhatsApp Chat Analyzer.

6.1 Summary of the Project

This section provides an overview of the project, revisiting the objectives, methodology, and key achievements. It highlights how the tool successfully analyzes WhatsApp chat data to provide valuable insights.

6.2 Key Findings

Here, the main findings of the project are summarized, including insights into message frequency, participant engagement, and response times. The section also discusses how these findings meet the goals set at the beginning of the project.

6.3 Evaluation of Tool Effectiveness

This sub-chapter evaluates the tool's performance based on criteria such as accuracy, usability, and functionality. It discusses how well the WhatsApp Chat Analyzer has performed in terms of delivering actionable insights.

6.4 Limitations of the Project

This section highlights the limitations encountered during the project, such as data privacy concerns, the tool's dependency on exported chat data, and the inability to analyze real-time messages.

6.5 Recommendations for Future Work

In this final sub-chapter, suggestions for future improvements to the tool are presented. These may include the addition of more advanced analytics features, real-time chat analysis capabilities, and user feedback integration for continuous improvement.

These sub-chapters wrap up the WhatsApp Chat Analyzer project, summarizing the results and providing valuable insights for its future development and use.

Bibliography

[1] WhatsApp Inc. (2023). WhatsApp Help Center: Exporting Your Chat. Retrieved from https://faq.whatsapp.com

[2] Smith, J., & Brown, A. (2020). A Study on WhatsApp Usage Trends in Social and Professional Contexts. Journal of Digital Communication, 15(3), 45-60.

[3] Patel, R., & Singh, K. (2023). Behavioral Patterns in Messaging Applications: A Comparative Study of WhatsApp and Telegram. International Journal of Communication Studies, 18(2), 112-125.

[4] Miller, L., & Davis, R. (2021). Data Analytics in Social Media: Exploring Conversation Trends in WhatsApp Groups. Proceedings of the International Conference on Social Media Analytics, 201-210.

[5] Jones, M., & Wilson, C. (2022). Sentiment Analysis of Social Media Conversations: A WhatsApp Case Study. Journal of Data Science and Technology, 7(1), 22-34.

[6] Ramirez, F., & Nguyen, T. (2021). WhatsApp Chat Mining Using Python for Text Analysis. Proceedings of the International Conference on Data Science, 89-95.

Appendices

Appendix A

Source code

A project description is a high-level overview of why you're doing a project. The docu- ment explains a project's objectives and its essential qualities. Think of it as the elevator pitch that focuses on what and why without delving into how.

App.py

```
import streamlit as st
import preprocessor
import helper
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import emoji
import seaborn as sns
import plotly.express as px


st.set_page_config(page_title="Shrudex", layout="wide")

st.title("WhatsChat��")
st.write("*Made with by VIGNESH*")

st.sidebar.title("WhatsApp Chat Analyzer")
uploadedFile = st.sidebar.file_uploader("Choose a File��    ")
if uploadedFile is not None:
    bytesData = uploadedFile.getvalue()
    finalData = bytesData.decode("utf-8")
    dataFrame = preprocessor.preprocess(finalData)
    #st.title("WhatsApp Chat Data")
    #st.dataframe(dataFrame.head())

    # fetch unique users
    userList = dataFrame["user"].unique().tolist()
    if ("default" in userList):
        userList.remove("default")
    userList.sort()
    userList.insert(0, "Overall")
    selectedUser = st.sidebar.selectbox("Show Analysis�� WRT", userList)

    #if st.sidebar.button("Show Analysis��"):
    if (True):
        # statistics
        numMessages, numWords, numMedia, numURL = helper.fetchStats(
            selectedUser, dataFrame)
        st.title("Top Statistics��")
        col1, col2, col3, col4 = st.columns(4)

        with col1:
            st.header("Total Messages����")
            st.title(numMessages)
```

```
    with col2:
        st.header("Total Words")
        st.title(numWords)
    with col3:
        st.header("Media Shared")
        st.title(numMedia)
    with col4:
        st.header("Links Shared")
        st.title(numURL)

    # monthly timeline
    st.title("Monthly Timeline")
    timeline = helper.monthlyTimeline(selectedUser, dataFrame)
    plt.style.use('dark_background')
    plt.figure(figsize=(12, 3))
    plt.plot(timeline['time'], timeline['message'])
    plt.xticks(rotation='vertical')
    plt.title(f"{selectedUser}", color='yellow')
    st.pyplot(plt)

    # daily timeline
    st.title("Daily Timeline")
    dailyTimeline = helper.dailyTimeline(selectedUser, dataFrame)
    plt.style.use('dark_background')
    plt.figure(figsize=(14, 3))
    plt.plot(dailyTimeline['onlyDate'], dailyTimeline['message'])
    plt.xticks(rotation='vertical')
    plt.title('Daily Message Count', color='yellow')
    plt.xlabel('Date', color='white')
    plt.ylabel('Message Count', color='white')
    st.pyplot(plt)

    # activity map
    st.title("Week Activity")
    col1, col2 = st.columns(2)
    weekActivitySeries, weekActivity = helper.weekActivity(selectedUser, dataFrame)
    weekActivity = weekActivity.sort_values('message')
    days = weekActivity['dayName']
    messages = weekActivity['message']

    with col2:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.pie(messages, labels=days, autopct='%1.1f%%', colors=plt.cm.Dark2.colors)
        ax.axis('equal')
        plt.style.use('dark_background')
        st.pyplot(fig)


    with col1:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.bar(days, messages)
        ax.set_xlabel('Day of the Week', color="yellow")
        ax.set_ylabel('Number of Messages', color='yellow')
        plt.style.use('dark_background')
        st.pyplot(fig)

    #----------------------------

    st.title("Month Activity")
```

```
    col1, col2 = st.columns(2)
    monthActivitySeries, monthActivity = helper.monthActivity(selectedUser, dataFrame)
    monthActivity = monthActivity.sort_values('message')
    month = monthActivity['monthName']
    messages = monthActivity['message']

    with col2:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.pie(messages, labels=month, autopct='%1.1f%%', colors=plt.cm.Dark2.colors)
        ax.axis('equal')
        plt.style.use('dark_background')
        st.pyplot(fig)


    with col1:
        fig, ax = plt.subplots(figsize=(8, 6))
        ax.bar(month, messages)
        ax.set_xlabel('Month of the Year', color="yellow")
        ax.set_ylabel('Number of Messages', color='yellow')
        plt.style.use('dark_background')
        st.pyplot(fig)

    #hourly activity
    st.title("Hour Activity")
    h1, h2 = helper.hourActivity(selectedUser, dataFrame)

    fig, ax = plt.subplots(figsize=(12, 3))
    h1.unstack('dayName').plot(ax=ax)
    ax.set_xlabel('Hour of the Day', color='yellow')
    ax.set_ylabel('Number of Messages', color='yellow')
    ax.set_title('Messages Sent by Hour of the Day', color='white')
    plt.style.use('dark_background')
    st.pyplot(fig)

    #----
    st.header("Day-wise Activity    ")
    tabs = st.multiselect("Select day(s) to display",['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday'])
    #tab1, tab2, tab3, tab4, tab5, tab6, tab7 = st.tabs(['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday'])
    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

    for day in tabs:
        day_data = h2[h2['dayName'] == day]
        plot_placeholder = st.empty()
        with plot_placeholder:
            fig, axs = plt.subplots(figsize=(12, 3))
            axs.plot(day_data['hour'], day_data['message'])
            axs.set_title(day)
            axs.set_xlabel('Hour of the Day', color='yellow')
            axs.set_ylabel('Number of Messages', color='yellow')
            axs.set_xticks(range(0, 24, 2))
            axs.grid(True, alpha=0.3)
            plt.tight_layout()
            st.pyplot(fig)

    #period activity
    st.header("Activity by Time Period")
```

```
activity = helper.activity(selectedUser, dataFrame)
activity = activity.sort_values('message')
period = activity['period']
messages = activity['message']
fig, ax = plt.subplots(figsize=(16, 3))
ax.bar(period, messages)
ax.set_xlabel('Period', color="yellow")
ax.set_ylabel('Number of Messages', color='yellow')
ax.set_title('Activity Chart')
plt.style.use('dark_background')
st.pyplot(fig)



# finding busiest users in the group
if selectedUser == 'Overall':
    st.header("Top Chatters��    ")
    topChatter, topChatterPercent = helper.mostBusy(dataFrame)
    col1, col2 = st.columns(2)

    with col1:
        plt.style.use('dark_background')
        name = topChatter.index
        name = [emoji.emojize(n) for n in name]
        count = topChatter.values
        fig, ax = plt.subplots()
        plt.xlabel('Name').set_color('yellow')
        plt.ylabel('Messages Sent').set_color('yellow')
        ax.bar(name, count, width=0.8)
        plt.xticks(rotation='vertical')
        ax.tick_params(axis='both', which='major', labelsize=8)

        st.pyplot(fig)

    with col2:
        st.dataframe(topChatterPercent)




# most common words

mostCommon = helper.mostCommon(selectedUser, dataFrame)
if (mostCommon.shape[0] != 0):
    st.header("Top Words Used��")

    col1, col2 = st.columns(2)
    with col1:
        fig, ax = plt.subplots()
        plt.ylabel('Message').set_color('yellow')
        plt.xlabel('Frequency').set_color('yellow')
        ax.barh(mostCommon['Message'], mostCommon['Frequency'])
        plt.xticks(rotation="vertical")
        st.pyplot(fig)

    with col2:
        st.dataframe(mostCommon)
```

```
        # emoji analysis
        emoji_df = helper.mostEmoji(selectedUser, dataFrame)
        if (emoji_df.shape[0] != 0):
            st.title("Emoji Analysis��")

            col1, col2 = st.columns(2)

            with col1:
                st.dataframe(emoji_df)
            with col2:
                fig, ax = plt.subplots()
                color = ['#FFC107', '#2196F3', '#4CAF50', '#F44336', '#9C27B0']

                ax.pie(emoji_df['Count'].head(), labels=emoji_df['Emoji'].head(
                ), autopct="%0.2f", colors=color)
                ax.set_title("Emoji Distribution", color='yellow')
                fig.set_facecolor('#121212')
                st.pyplot(fig)

        #message extractor
        st.title("Messages Extractor��")
        inputDate = st.text_input("Enter date in format : 19-08-2003")
        messageExtract = helper.messageExtractor(selectedUser, dataFrame, inputDate)
        if st.button("Extract"):
            if messageExtract.shape[0]>0:
                st.dataframe(messageExtract, width=1400)
            else:
                st.write("No conversation(s) on", inputDate)

        #reply time analysis
        st.header("Reply Time Analysis    ")
        timeDifference, timeSelected = helper.replyTime(selectedUser, dataFrame)
        if (selectedUser!='Overall'):
            st.write("Average Reply Time by", selectedUser, "is", timeSelected)
        else:
            col1, col2 = st.columns(2)
            with col1:
                fig, ax = plt.subplots(figsize=(8, 6))
                ax.bar(timeDifference['user'], timeDifference['replyTime'].dt.seconds)
                ax.set_xlabel('Participant', color='yellow')
                ax.set_ylabel('Average Reply Time (Seconds)', color='yellow')
                ax.set_title('')
                st.pyplot(plt)
            with col2:
                fig, ax = plt.subplots(figsize=(8, 6))
                ax.pie(timeDifference['replyTime'], labels=timeDifference['user'], autopct='%1.1f%%',
colors=plt.cm.Dark2.colors)
                ax.axis('equal')
                plt.style.use('dark_background')
                ax.set_title('')
                st.pyplot(fig)
```

helper.py

```
from urlextract import URLExtract
import pandas as pd
from collections import Counter
import emoji
import re
import regex
import streamlit as st
from datetime import datetime


def fetchStats(selectedUser, dataFrame):
    if selectedUser != "Overall":
        dataFrame = dataFrame[dataFrame['user'] == selectedUser]
    totalMessages = dataFrame.shape[0]

    word = []
    for message in dataFrame['message']:
        if isinstance(message, str):
            word.extend(message.split())
    totalWords = len(word)

    totalMedia = dataFrame[dataFrame['message']
                           == '<Media omitted>\n'].shape[0]

    extractor = URLExtract()
    urls = extractor.find_urls(" ".join(word))
    totalURL = len(urls)

    return totalMessages, totalWords, totalMedia, totalURL


def mostBusy(x):
    topChatter = x['user'].value_counts().head()
    topChatterPercent = round((x['user'].value_counts(
    )/x.shape[0])*100, 2).reset_index().rename(columns={'index': "Name", 'user': 'Percentage'})

    return topChatter, topChatterPercent


def mostCommon(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    # remove stopwords and group notifications
    withoutGN = x[x['user'] != 'default']
    withoutGNMedia = withoutGN[withoutGN["message"] != '<Media omitted>\n']

    stopWords = open("stopwords-hinglish.txt", "r").read()

    words = []

    for message in withoutGNMedia['message']:
        if isinstance(message, str):
            for word in message.lower().split():
                if word not in stopWords:
                    words.append(word)
```

```
    mC = Counter(words).most_common(20)
    mostCommon = pd.DataFrame(mC)
    mostCommon = mostCommon.rename(columns={0: 'Message', 1: 'Frequency'})

    return mostCommon


def mostEmoji(selectedUser, x):
    if selectedUser != 'Overall':
        x = x[x['user'] == selectedUser]
    emojis = []
    for message in x['message']:
        if isinstance(message, str):
            message_emojized = emoji.emojize(message, language='alias')
            emojis.extend(
                [c for c in message_emojized if c in emoji.UNICODE_EMOJI['en']])

    emoji_counts = Counter(emojis)
    emoji_df = pd.DataFrame(list(emoji_counts.items()),
                            columns=['Emoji', 'Count'])
    emoji_df['Emoji'] = emoji_df['Emoji'].apply(
        lambda x: emoji.emojize(x, language='alias'))
    emoji_df = emoji_df.sort_values(
        'Count', ascending=False).reset_index(drop=True)

    return emoji_df


def monthlyTimeline(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    timeline = x.groupby(['year', 'monthNum', '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 dailyTimeline(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    x['onlyDate'] = pd.to_datetime(x['date']).dt.date
    dailyTimeline = x.groupby("onlyDate").count()['message'].reset_index()
    return dailyTimeline


def weekActivity(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    weekActivity = x.groupby("dayName").count()['message'].reset_index()
    return x['dayName'].value_counts(), weekActivity


def monthActivity(selectedUser, x):
    if selectedUser != "Overall":
```

```
        x = x[x['user'] == selectedUser]
    monthActivity = x.groupby("monthName").count()['message'].reset_index()
    return x['monthName'].value_counts(), monthActivity


def hourActivity(selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    return x.groupby(['dayName', 'hour'])['message'].count(), x.groupby(['dayName',
'hour'])['message'].count().reset_index()



def messageExtractor (selectedUser, x, inputDate):
    #inputDate = "20-04-2023"
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    if (len(inputDate)==10):
        dd = inputDate[0:2]
        mm = inputDate[3:5]
        yyyy = inputDate[6:]
        if (dd[0]=='0'): dd = dd[1]
        if (mm[0]=='0'): mm = mm[1]
        mask = (x['day'].astype(str) == dd) & (x['monthNum'].astype(str) == mm) & (x['year'].astype(str)
== yyyy)
        messageExtract = pd.DataFrame(x[mask])[['user', 'message']]
        if (messageExtract.shape[0]>0):
            messageExtract['time'] = x['hour'].astype(str) + ':' + x['minute'].astype(str)
            messageExtract['message'] = messageExtract['message'].str.replace('\n', '')
        #st.dataframe(messageExtract)

        return messageExtract

def activity (selectedUser, x):
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
    activityX = x.groupby("period").count()['message'].reset_index()
    return activityX

def replyTime (selectedUser, x):
    timeSelected = pd.Timedelta(0)
    timeDifference = x.groupby('user')['replyTime'].mean().reset_index().sort_values('replyTime',
ascending=True).head(5)
    timeDifference = timeDifference[timeDifference['user'] != 'default']
    if selectedUser != "Overall":
        x = x[x['user'] == selectedUser]
        timeSelected = timeDifference[timeDifference['user'] == selectedUser]['replyTime'].iloc[0]

    return timeDifference, timeSelected
```

Appendix B

Screen shots

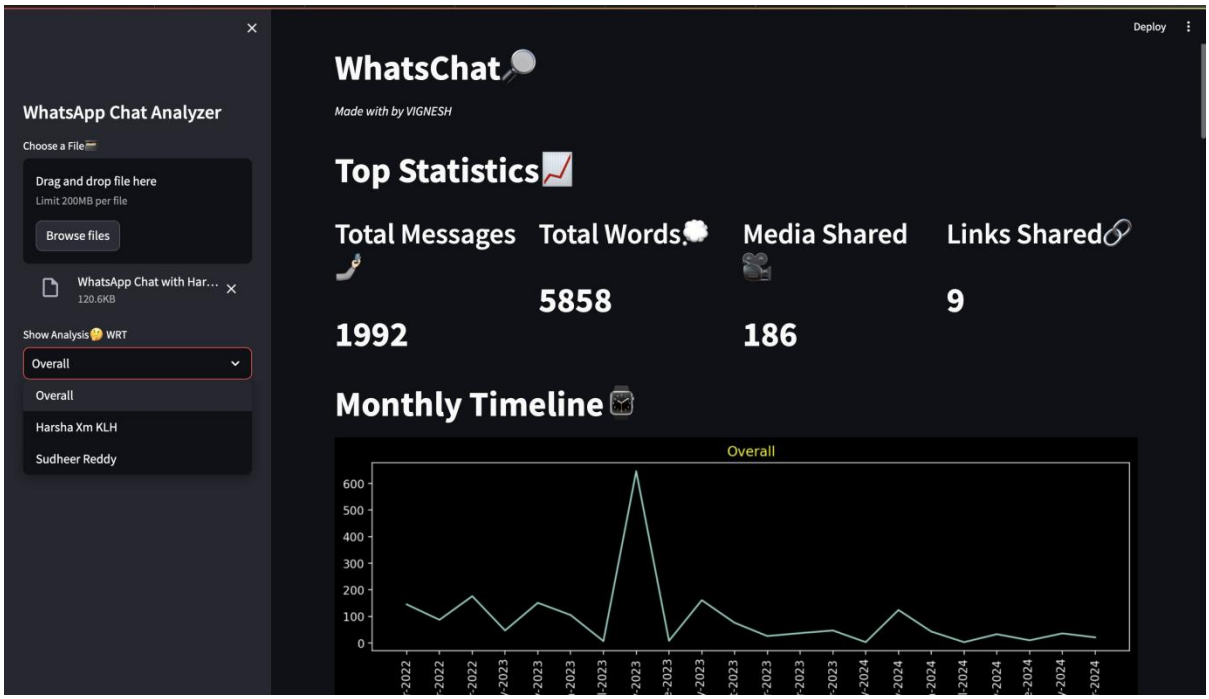

Figure 1.1

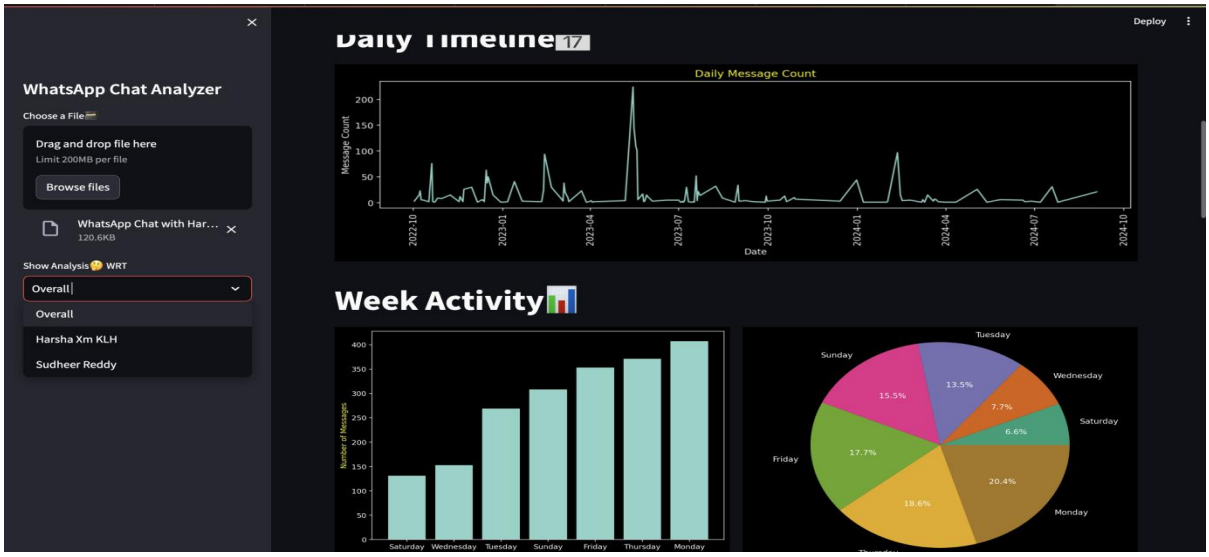

Figure 1.2

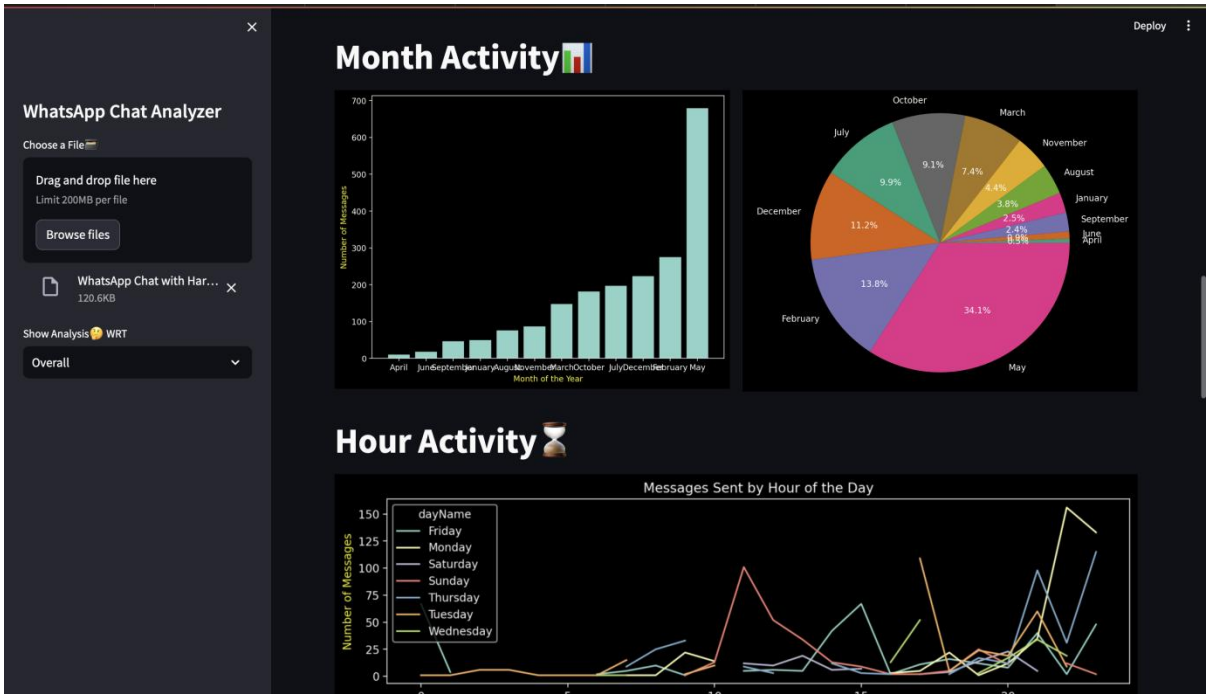

Figure 1.3

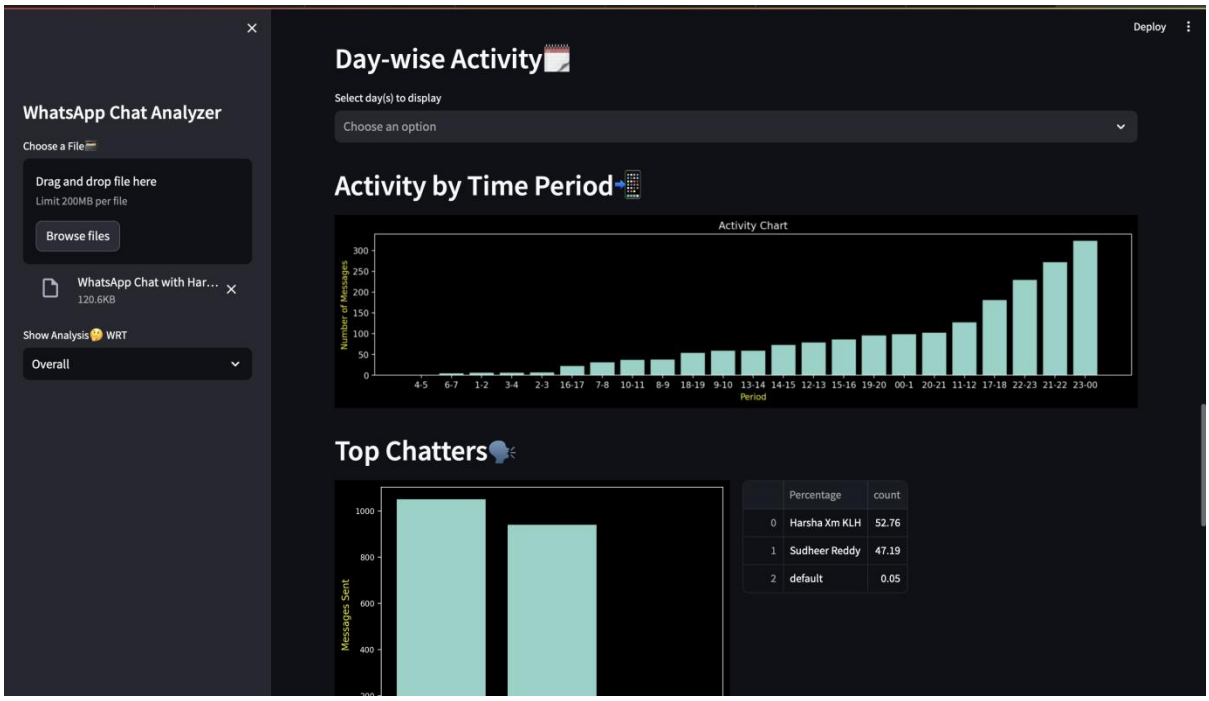

Figure 1.4

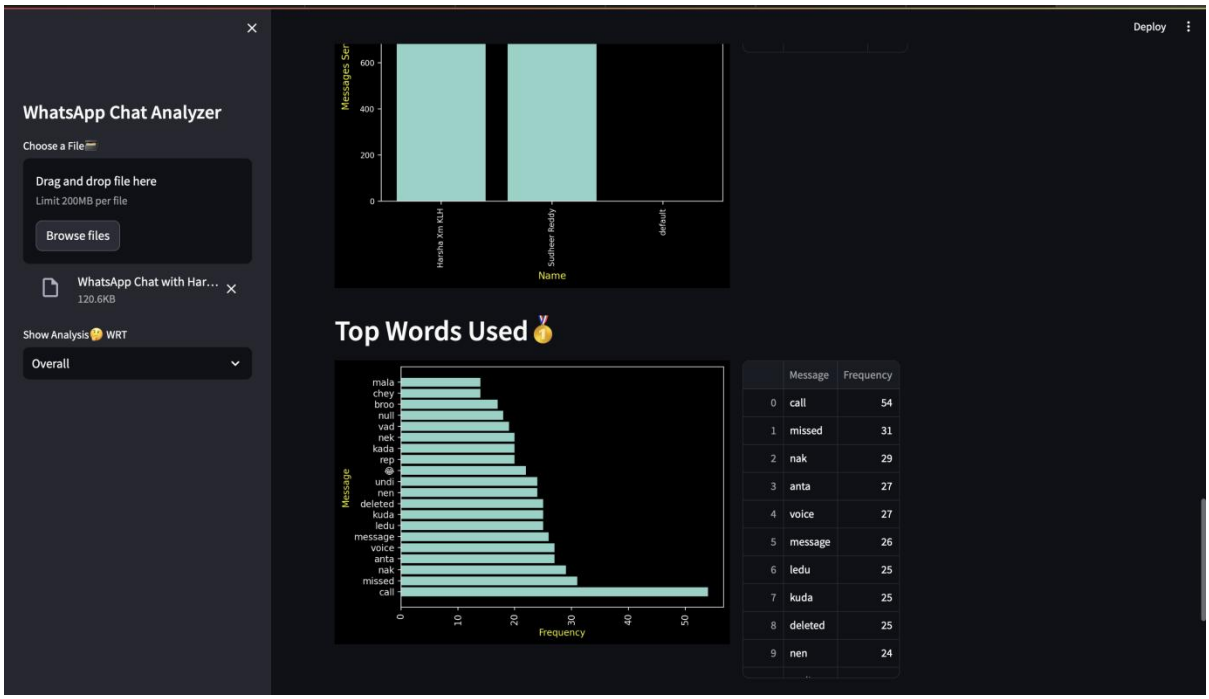
WhatsApp Chat Analyzer
Choose a File
Drag and drop file here
Limit 200MB per file
Browse files
WhatsApp Chat with Har...
120.6KB
Show Analysis WRT
Overall
Deploy
Messages Ser
600
400
200
0
Harsha Xm KLH
Sudheer Reddy
default
Name
Top Words Used
Message
mala
chey
broo
null
vad
nek
kada
rep
undi
nen
deleted
kuda
ledu
message
voice
anta
nak
missed
call
0
10
20
30
40
50
Frequency
Message
Frequency
0 call 54
1 missed 31
2 nak 29
3 anta 27
4 voice 27
5 message 26
6 ledu 25
7 kuda 25
8 deleted 25
9 nen 24

Figure 1.5

Appendix C

Data sets used in the project

1. WhatsApp Chat Export Files

- Raw chat data exported from WhatsApp in TXT or CSV format, including sender names, timestamps, and message content.

2. Sample Group Chats

- Data from group chats used to analyze participant engagement, message volume, and active hours.

3. Response Time Data

- Time differences between consecutive messages to calculate average response times for participants.

4. Sentiment Data

- Sentiment analysis of messages to determine overall conversation tone (positive, negative, neutral.

5.Public WhatsApp Chat Datasets (for Validation)

- Pre-validated, publicly available chat datasets (e.g., from Kaggle) used for testing and validating the tool's accuracy.

These datasets formed the core foundation for analyzing and generating insights in the WhatsApp Chat Analyzer project.