Mini Project On

Whatsapp Chat Analyzer

By

Pankaj R. Bhoir (2021510006)
Haresh A. Gayakhe (2021510015)

Under the guidance of
Internal Supervisor

Prof. Dr. Pooja Raundale

Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University
2021-23

CERTIFICATE OF APPROVAL

This is to certify that the following students

Pankaj R. Bhoir (2021510006)
Haresh A. Gayakhe (2021510015)

Have satisfactorily carried out work on the project entitled

“Whatsapp Chat Analyzer”

Towards the fulfilment of project, as laid down
by
Sardar Patel Institute of Technology
during year
2021-23.

Project Guide:
Prof. Dr. Pooja Raundale

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Pankaj R. Bhoir (2021510006)
Haresh A. Gayakhe (2021510015)

Have successfully completed the Project report on

"Whatsapp Chat Analyzer",

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI

INTERNAL EXAMINER EXTERNAL EXAMINER

HEAD OF DEPARTMENT PRINCIPAL

Contents

Abstract

The most used and efficient method of communication in recent times is an application called WhatsApp. WhatsApp chats consists of various kinds of conversations held among group of people.

WhatsApp chat analyzer is the application which provide analysis of WhatsApp group chats as well as individual chat. There are various methodologies available for analysis but here matplotlib, streamlit, seaborn, re, pandas libraries of python and some concept of NLP is used. This is the combination of machine learning and NLP.

This whatsapp chat analyzer take import whatsapp chat file from user and analyze it and give different visualizations as a result.

Objectives

The Web based Application "Whatsapp Chat Analyzer" is used

- To provides a platform to the user which enables user to analyze whatsapp chats
- This application allows user to browse whatsapp exported (.txt) file and import it to WhatsApp chat analyzer and get analysis according to that txt file.
- The user can Analyze by clicking Show Analysis button.
- To visualize data simple and attractive formate

List of Figures

List of Tables

1 Introduction

1.1 Problem Definition

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 another participant a user responds to the most. We propose to employ dataset manipulation techniques to have a better understanding of WhatsApp chat present in our phones..

1.2 Objectives and Scope

1.2.1 Objectives

The Web based Application "Whatsapp Chat Analyzer" is

- To provide an in-depth exploratory data analysis on various types of WhatsApp chats.
- Use Streamlit to build a dashboard
- To present an analysis of the WhatsApp group data to as certain the level of involvement and participation by members in that group chat.
- To extract WhatsApp chat data

1.2.2 Scope

The user can provide his/her chat details for in-depth analysis of whatsapp chats.

In the web application the user must enter his/her personal and educational details in the profile section.

Our System is being made for reducing the information loss and smoothening the communication between Training and Placement Co-ordinator and students so that they both have all the required information in their hand.

1.3 Existing System

In olden days' there is no analysis for whatsapp chat. If someone wants to analyze there is no CSV file to analyze. WhatsApp Application provide export txt file which is in raw format. It is very complicated for analysis. So we have to forget that system and switch over to the WhatsApp Chat Analyzer.
Some of the disadvantages of existing system are as follows :

- Analysis are not accurate.
- Raw data.
- Time consuming.
- Difficult to Analyze.

1.4 Proposed System

The "WhatsApp Chat Analyzer" provides a platform to the user which enables user to analyze whatsapp chats online on heroku link. This application allows user to browse whatsapp exported (.txt) file and import it to WhatsApp chat analyzer and get analysis according to that txt file. And user can Analyze by clicking Show Analysis button.

Some of the advantages of our system are as follows :

- Runs on all devices.
- Shows based on whatsapp chat file.
- Shows different visualizations.
- Shows Total Messages.

1.5 System Requirements

- Hardware Requirements on Client Side

Table 1.5.1: Hardware Requirements on Client Side

Device	Any operating system which supports python
Processor	Dual Core Processor or Above
RAM	Minimum 2 GB RAM
Storage	Minimum 10 GB Storage Space

- Software Requirements on Client Side

Table 1.5.2: Software Requirements on Client Side

Operating System	Windows/Linux/Mac OS
Server	Not Required

2 Software Requirement Specification (SRS) and Design

2.1 Purpose

The purpose of our project is to develop an web application that can help user to easily analyze their Whatsapp data and to keep all the required information handy.
This can save lots of efforts of user and it will be easier to analyze the data the each and every single and small detail. This web app will keep track of all the information regarding the whatsapp chats.

2.2 Definition

To build a Web Application so the user can have an easy to analyze the data.

2.3 Overall Description

2.3.1 Product Functions

The product function includes:

1. Run App: Users are required to run web app on any system.
2. Upload data: Then user have to upload the text file of the data which is contains whatsapp chat.
3. View analysis: After clicking on view analysis it shows the detailed analysis of whatsapp chat

2.3.2 User Characteristics

There are only one types of user:

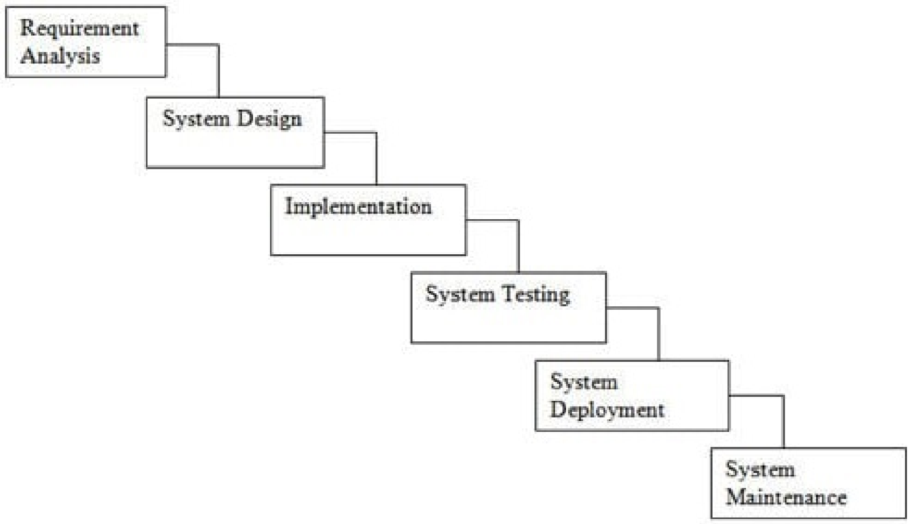
- User: user have to upload their whatsapp chat txt file. then after clicking on view analysis, web app will show the detailed analysis of whatsapp chat. where user can check the analysis according to what deta he/she needed.

3 Project Analysis and Design

3.1 Methodologies Adapted

In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

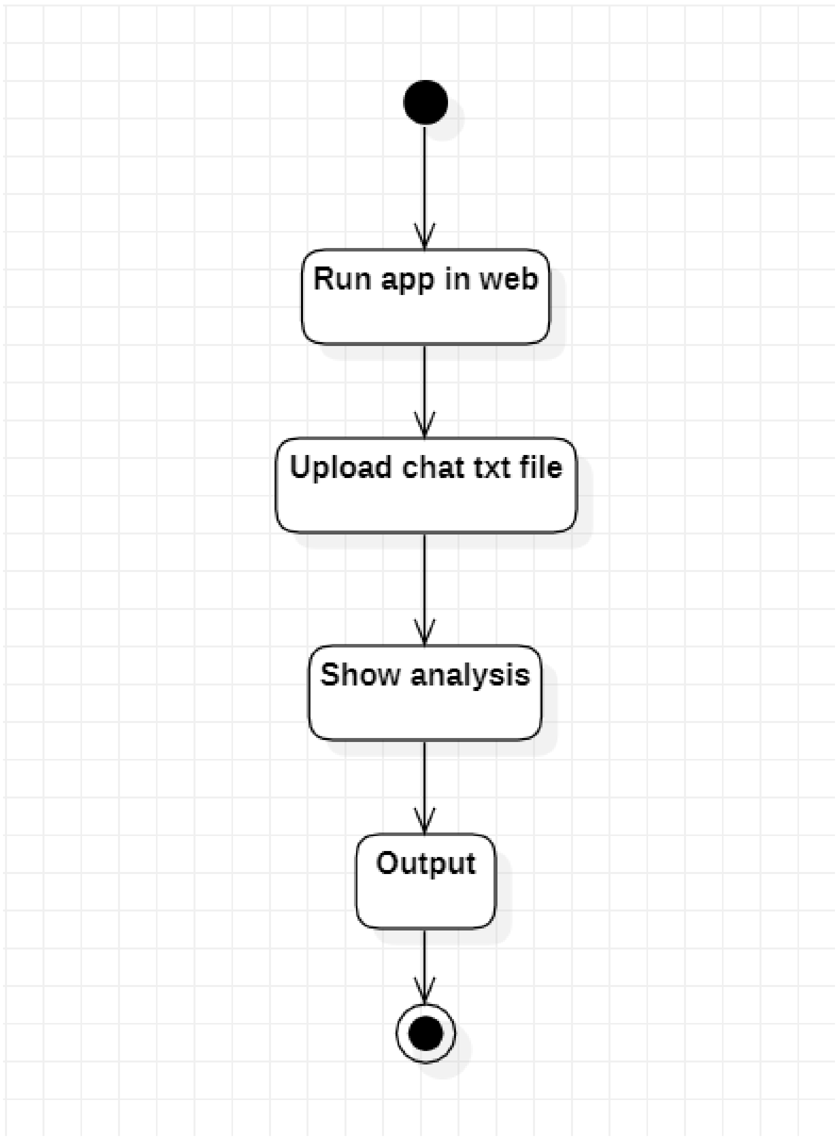
Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



3.1.1: Waterfall Model

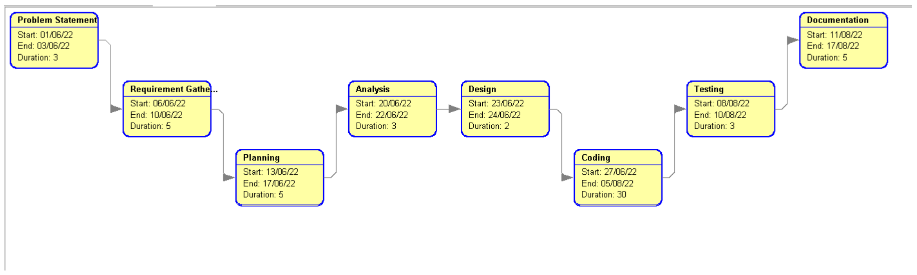
3.2 Modules

3.2.1 Activity diagram



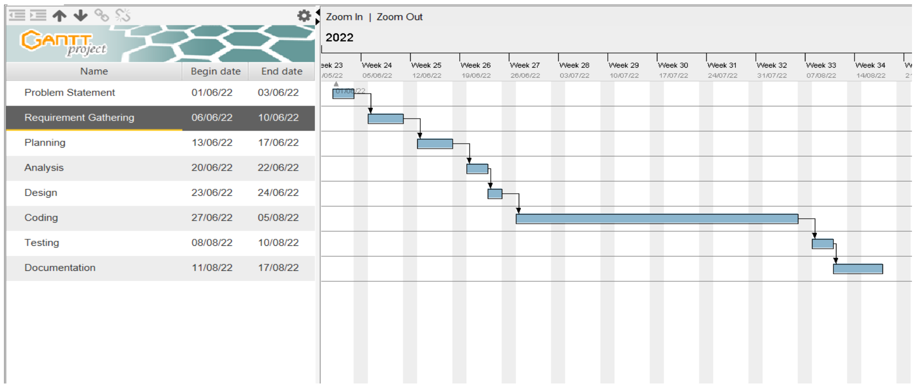
3.2.1: Activity Diagram

3.2.2 PERT Chart



3.2.2: PERT Chart

3.2.3 Gantt Chart

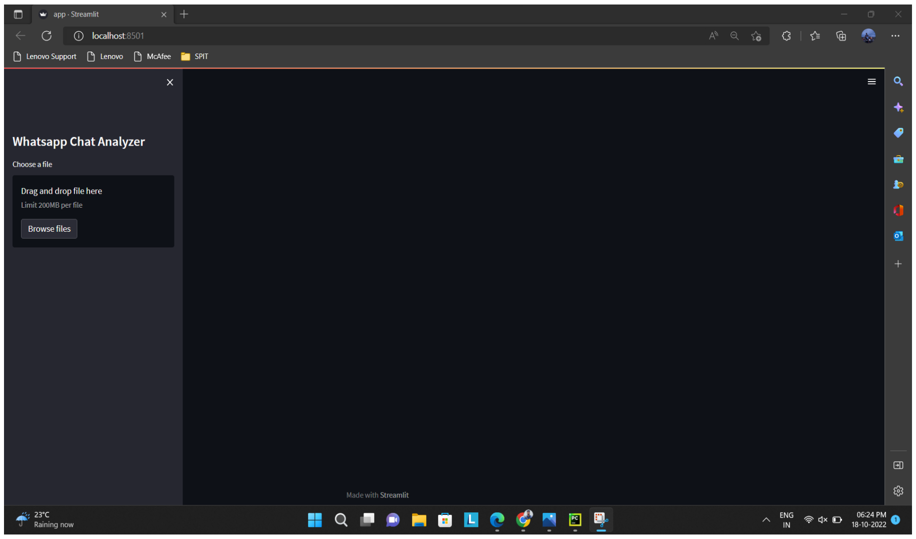


Name	Begin date	End date
Problem Statement	01/06/22	03/06/22
Requirement Gathering	06/06/22	10/06/22
Planning	13/06/22	17/06/22
Analysis	20/06/22	22/06/22
Design	23/06/22	24/06/22
Coding	27/06/22	05/08/22
Testing	08/08/22	10/08/22
Documentation	11/08/22	17/08/22

3.2.3: Gantt Chart

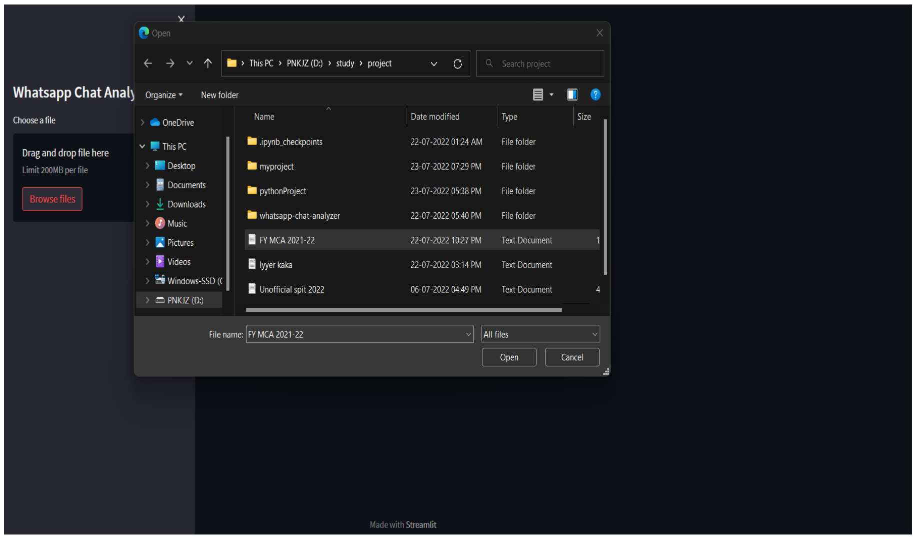
4 Project Implementation and Testing

4.1 Blank web app



4.1.1: Web View

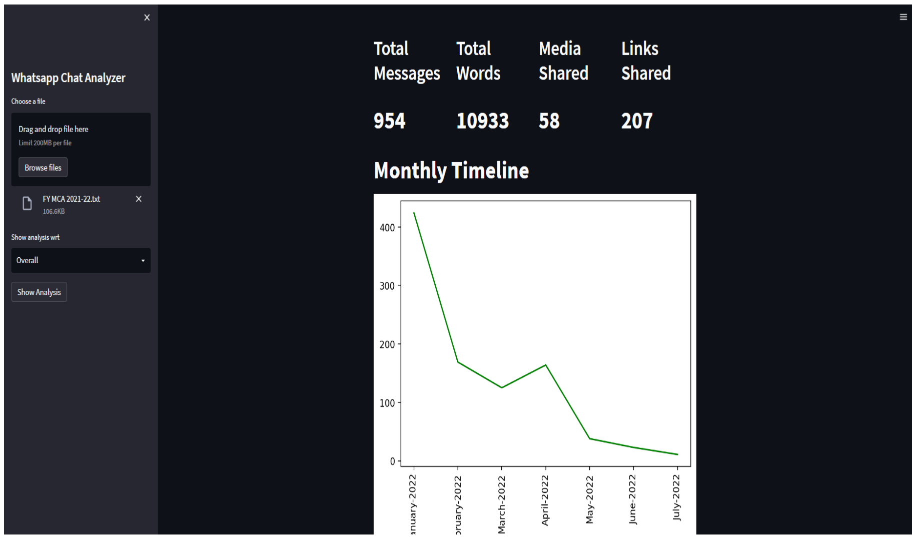
4.2 Select chat text file



Name	Date modified	Type	Size
.ipynb_checkpoints	22-07-2022 01:24 AM	File folder	
myproject	23-07-2022 07:29 PM	File folder	
pythonProject	23-07-2022 05:38 PM	File folder	
whatsapp-chat-analyzer	22-07-2022 05:40 PM	File folder	
FY MCA 2021-22	22-07-2022 10:27 PM	Text Document	1
Iyyer kaka	22-07-2022 03:14 PM	Text Document	
Unofficial spit 2022	06-07-2022 04:49 PM	Text Document	4

4.2.1: File upload

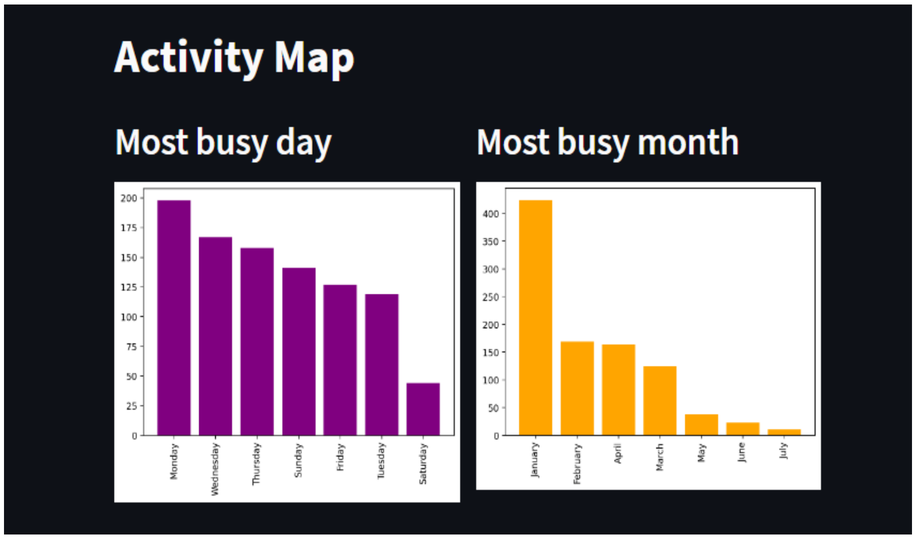
4.3 Click on show analysis



Total Messages	Total Words	Media Shared	Links Shared
954	10933	58	207

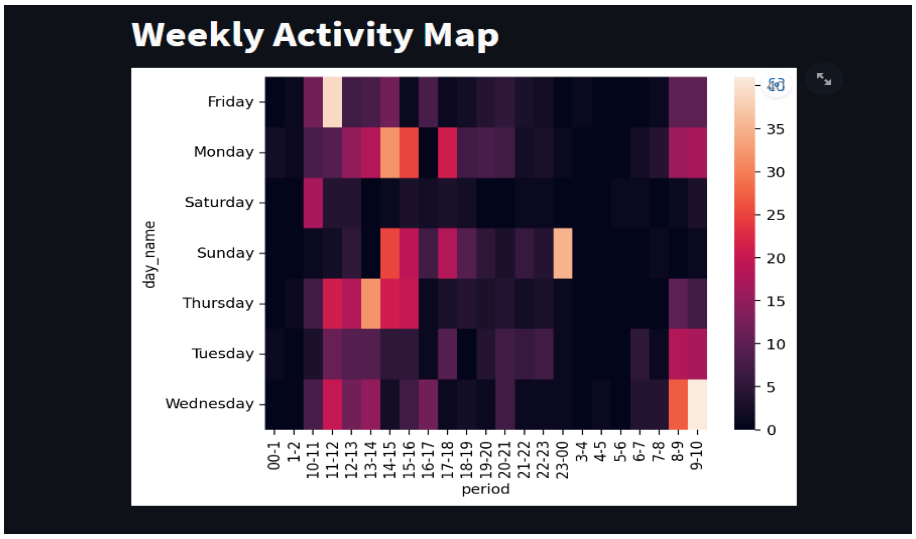
4.3.1: Show analysis

4.4 Activity map



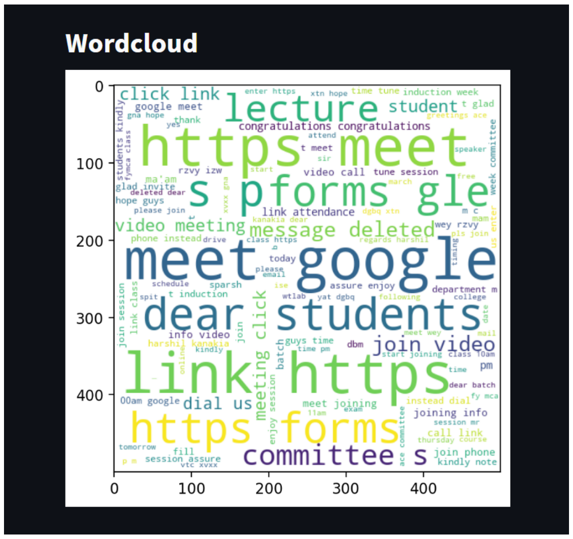
4.4.1: Activity histogram

4.5 Weekly activity

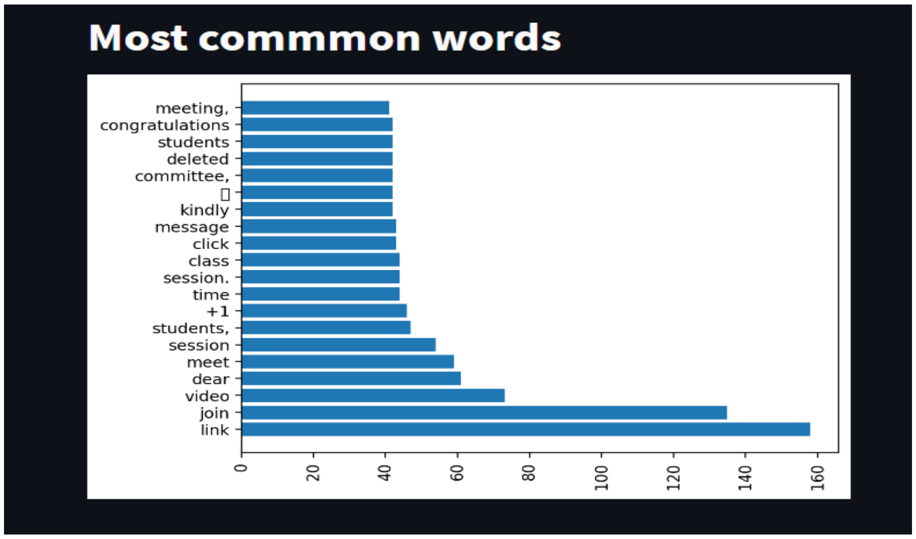


4.5.1: Heat map

4.6 Wordcloud

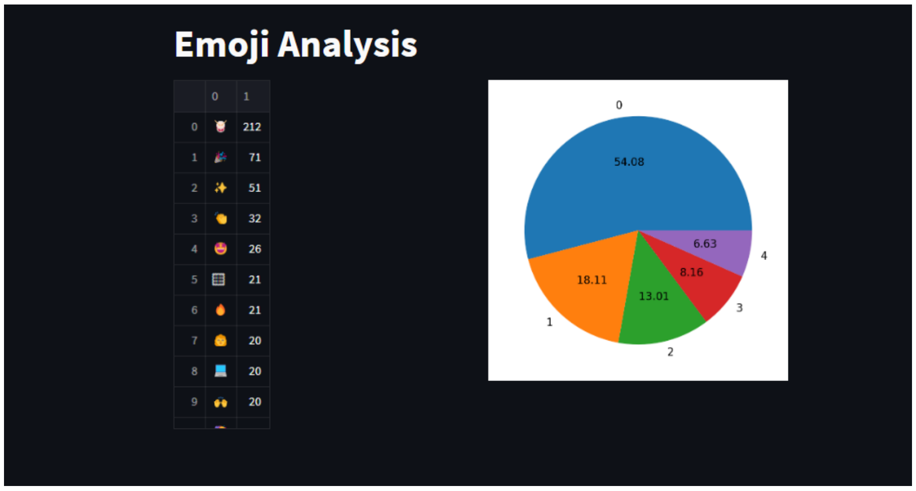


4.7 Common words



4.7.1: Most Common words

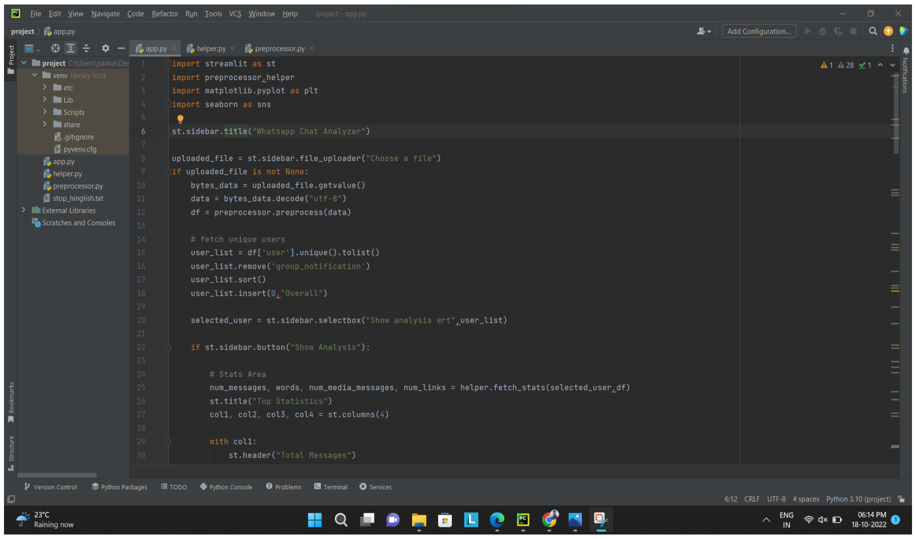
4.8 Emoji analysis



Emoji Analysis

	0	1
0	🥁	212
1	🎉	71
2	✨	51
3	👏	32
4	🤩	26
5	🗓	21
6	🔥	21
7	🙇	20
8	💻	20
9	🙌	20

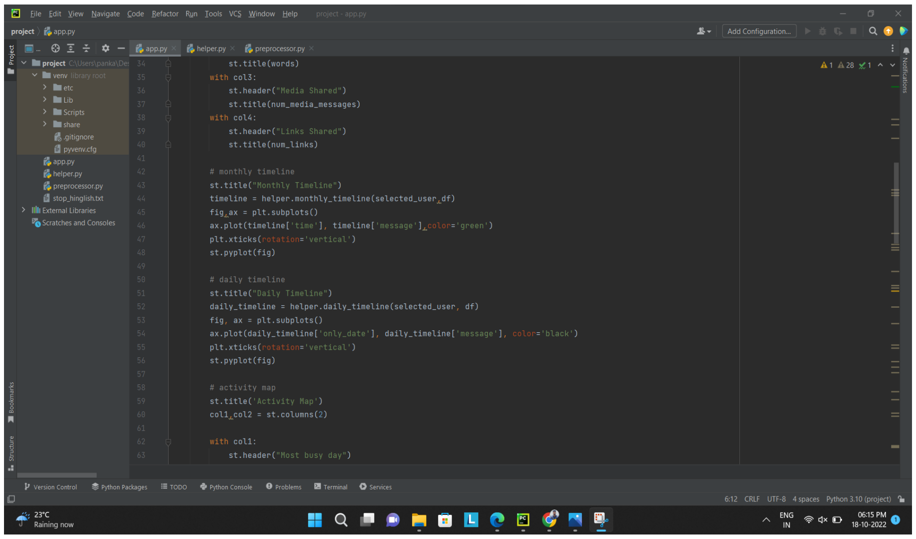
4.9 Code 1



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

st.sidebar.title("Whatsapp Chat Analyzer")

uploaded_file = st.sidebar.file_uploader("Choose a file")
if uploaded_file is not None:
    bytes_data = uploaded_file.getvalue()
    data = bytes_data.decode("utf-8")
    df = preprocessor.preprocess(data)

    # 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 analysis wrt",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")
        col1, col2, col3, col4 = st.columns(4)

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

4.9.1: app.py

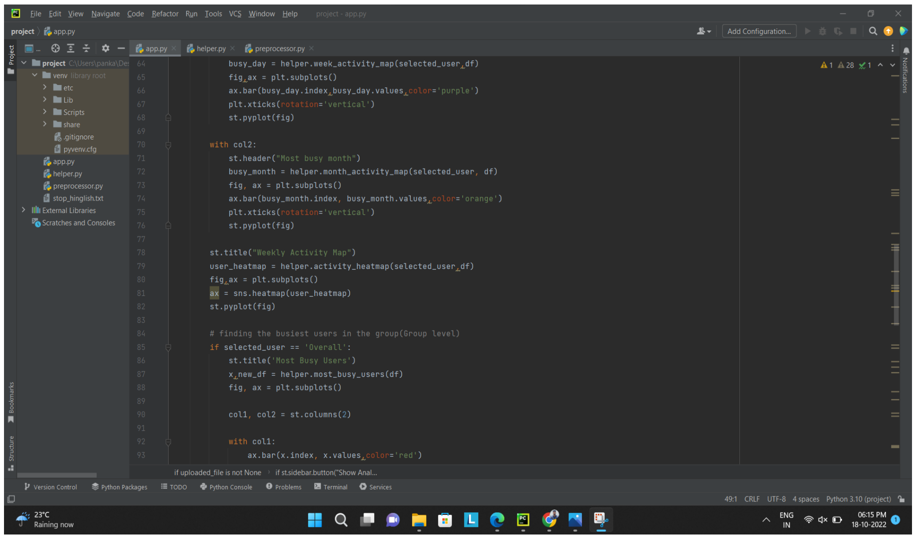
4.10 Code 2



```python
            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")
```

4.10.1: app.py

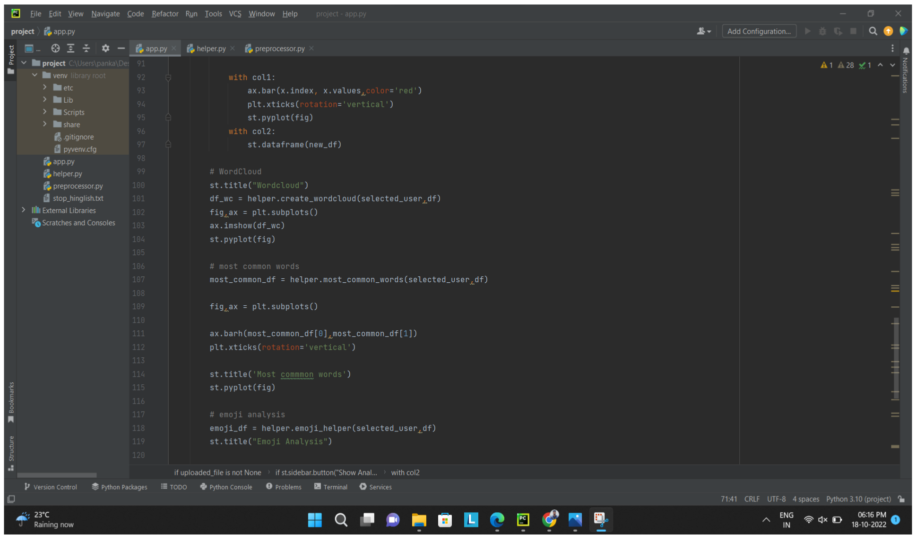
4.11 Code 3



```
    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='orange')
    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)

# 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')
```

4.11.1: app.py

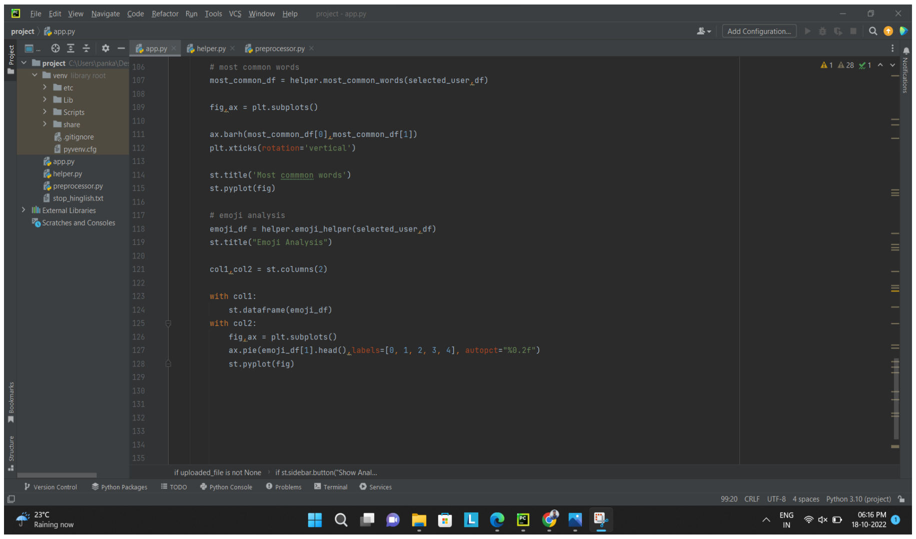
4.12 Code 4



```

    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)

fig,ax = plt.subplots()

ax.barh(most_common_df[0],most_common_df[1])
plt.xticks(rotation='vertical')

st.title('Most commmon words')
st.pyplot(fig)

# emoji analysis
emoji_df = helper.emoji_helper(selected_user,df)
st.title("Emoji Analysis")

```

4.12.1: app.py

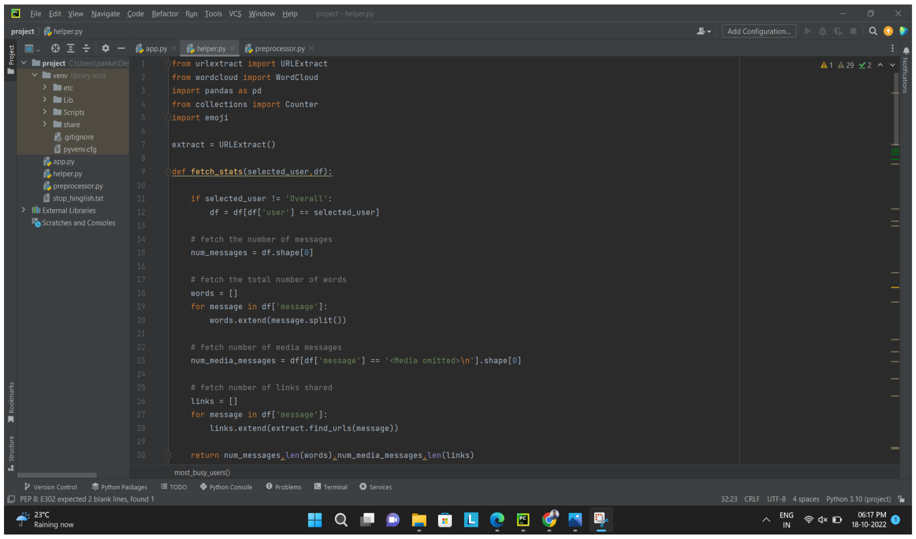
4.13 Code 5



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

        fig,ax = plt.subplots()

        ax.barh(most_common_df[0],most_common_df[1])
        plt.xticks(rotation='vertical')

        st.title('Most commmon words')
        st.pyplot(fig)

        # emoji analysis
        emoji_df = helper.emoji_helper(selected_user,df)
        st.title("Emoji Analysis")

        col1,col2 = st.columns(2)

        with col1:
            st.dataframe(emoji_df)
        with col2:
            fig,ax = plt.subplots()
            ax.pie(emoji_df[1].head(),labels=[0, 1, 2, 3, 4], autopct="%0.2f")
            st.pyplot(fig)
```

4.13.1: app.py

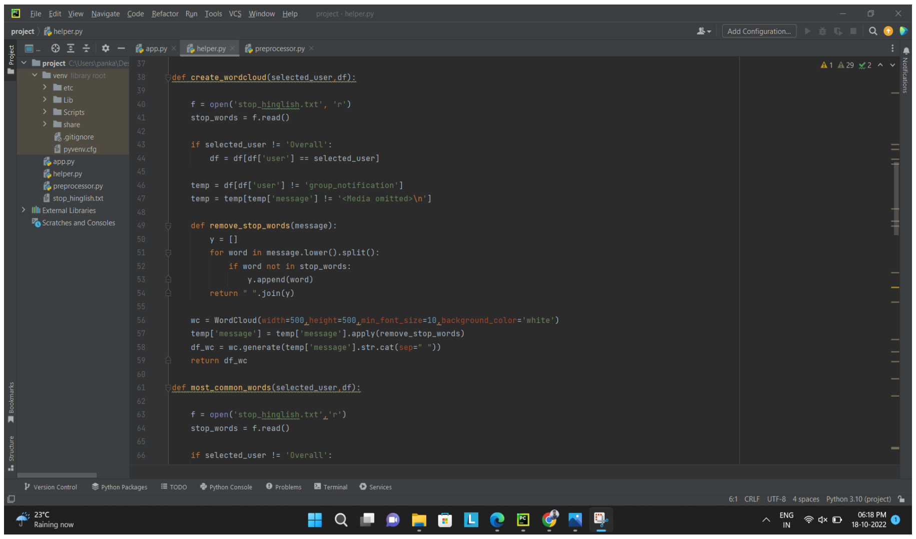
4.14 Code 6



```python
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':
        df = df[df['user'] == selected_user]

    # 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())

    # 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)
```

4.14.1: helper.py

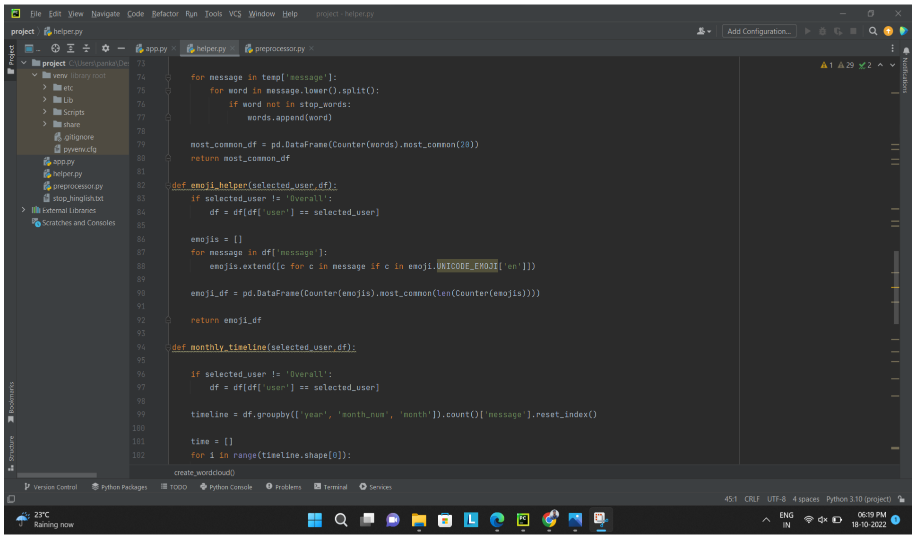
4.15 Code 7



```
def create_wordcloud(selected_user,df):

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

    if selected_user != 'Overall':
```

4.15.1: helper.py

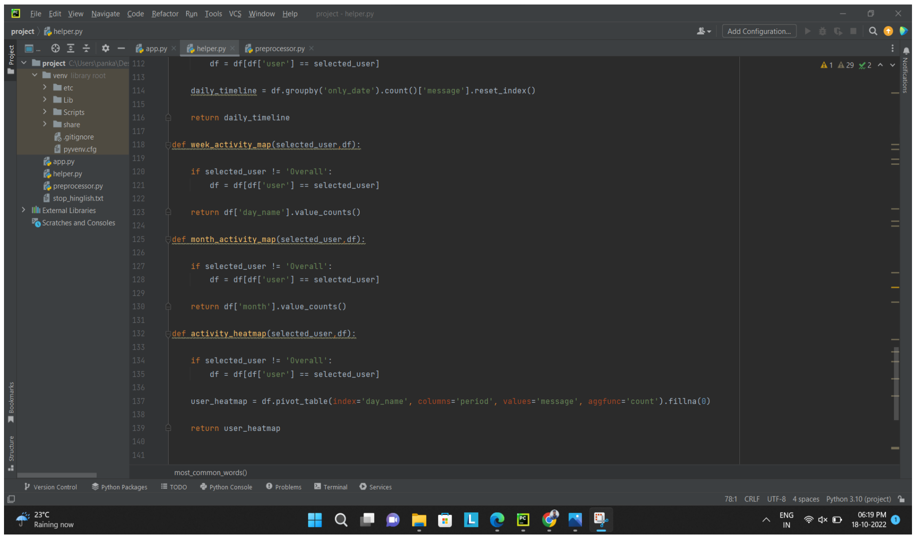
4.16 Code 8



```
    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.UNICODE_EMOJI['en']])

    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]):
```

4.16.1: helper.py

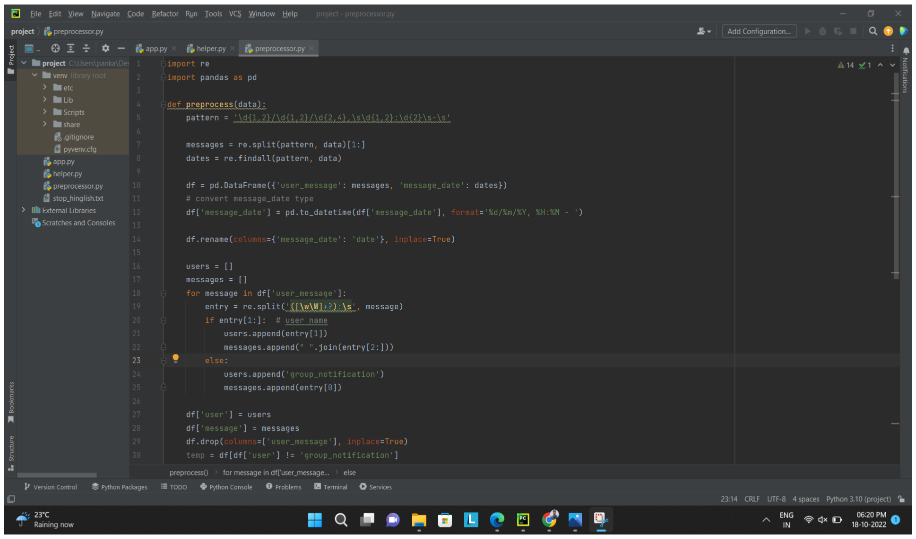
4.17 Code 9



```python
        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_heatmap
```

4.17.1: helper.py

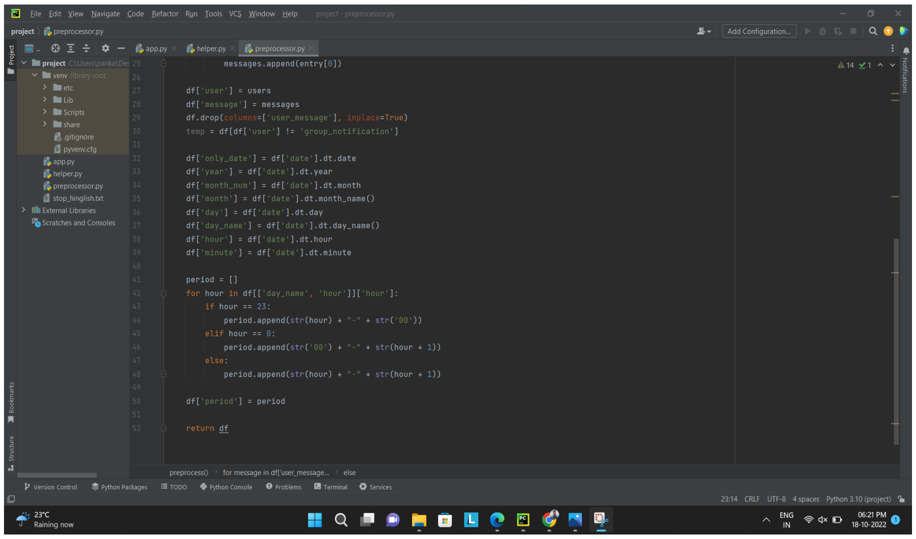
4.18 Code 10



```python
import re
import pandas as pd

def preprocess(data):
    pattern = '\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s-\s'

    messages = re.split(pattern, data)[1:]
    dates = re.findall(pattern, data)

    df = pd.DataFrame({'user_message': messages, 'message_date': dates})
    # convert message_date type
    df['message_date'] = pd.to_datetime(df['message_date'], format='%d/%m/%Y, %H:%M - ')

    df.rename(columns={'message_date': 'date'}, inplace=True)

    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)
    temp = df[df['user'] != 'group_notification']
```

4.18.1: preprocessor.py

4.19 Code 11



```
            messages.append(entry[0])

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

    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
```

4.19.1: preprocessor.py

5 Limitations

- Time format should be in 24 hrs format.
- It does not have a feature to view media files it only shows number of files shared.

6 Future Enhancements

- Upload both 12hrs and 24hrs time format.
- Abusive words detection.
- Sentiment analysis of users.

7 User Manual

- Run the code
- Click on browse files and select Whatsapp chat file on which you want to perform analysis.
- Click on show analysis to display output

8 CONCLUSION

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.

9 Bibliography

9.1 Web References

[1.] https://streamlit.io/
[2.] https://seaborn.pydata.org/
[3.] https://pandas.pydata.org/docs/
[4.] https://www.geeksforgeeks.org/generating-word-cloud-python/
[5.] https://www.tutorialspoint.com/python_text_processing/python_extract_url_from_text.htm