**WhatsApp Chat Analyzer**

**Overview**

*In an era defined by the digital exchange of thoughts, ideas, and emotions, WhatsApp has emerged as a prominent platform for communication. With millions of users worldwide, it serves as a hub for individuals and groups to engage in conversations, share media, and express themselves. However, amidst the endless stream of messages, valuable insights and trends often remain concealed.*

**WhatsApp Chat Analyzer:** A Tool for Systematic WhatsApp Chat Analysis

This project, WhatsApp Chat Analyzer, stands as an instrument designed and tailored for the analysis of WhatsApp chat files. Whether dissecting personal dialogues or scrutinizing the dynamics of group discussions, this platform serves as an invaluable resource for uncovering patterns within conversations.

**Effortless Analysis**

This project simplifies the process of chat analysis, offering a streamlined experience that begins with a straightforward upload of the WhatsApp chat file from the end user. The user-friendly interface ensures accessibility for individuals with varying levels of technical proficiency.

**In-Depth Analysis**

WhatsApp Chat Analyzer provides a comprehensive set of insights, organized under the following categories:

* **Top Statistics:** This section provides a quick overview of essential chat metrics, including total message count, cumulative word count, media file sharing frequency, and shared link occurrences.
* **Monthly Trendline:** Explore the temporal dynamics of group activity through a monthly trend analysis, revealing peaks and lulls in conversation intensity.
* **Daily Timeline:** Discover daily messaging patterns, shedding light on periods of heightened chat activity.
* **Activity Map:** Ascertain member engagement patterns via an interactive heatmap, revealing peak and off-peak hours and days, complemented by informative column graphs.
* **Top Contributors:** Identify the most active chat participants through a comprehensive column graph, showcasing their chat contribution percentages over time.
* **Word Cloud:** Visualize the chat's lexicon, highlighting frequently used words and aiding in the identification of recurring themes and topics.
* **Common Words:** Gain a nuanced understanding of the chat's linguistic composition through a bar chart depicting the most frequently employed words and their corresponding frequencies.
* **Emoji Analysis:** A comprehensive breakdown of the most frequently used emoticons, presented in a clear pie chart format.

WhatsApp Chat Analyzer's versatility is a notable feature, enabling analysis of both individual users' contributions and collective group dynamics.

The end user can unlock the latent potential within their WhatsApp conversations with WhatsApp Chat Analyzer. Gain profound insights, unravel intricate patterns, and delve into the fascinating realm of your textual exchanges.

**Coding**

The project is organized into three separate Python files: "preprocessor.py," "functions.py," and "main.py." These files are interdependent and work together to form the complete project.

The working and purpose of all three are listed below:

1. **preprocessor.py:** This Python file is designed to preprocess WhatsApp chat data stored in a text file. The goal is to extract relevant information and structure it into a Pandas DataFrame for further analysis.

**Code Overview:** The code is organized into several sections, each performing specific data processing tasks.

* + **1. Importing Libraries:** Importing necessary libraries, including 're' for regular expressions and 'pandas' for data manipulation.
  + **2. Loading the Chat File:** Opens the chat file, reads its contents into a string, and prints the data along with its type.
  + **3. Data Splitting and Parsing:** Uses regular expressions to split the chat data into messages and dates based on a common pattern. Stores messages and dates in separate lists.
  + **4. Converting to Pandas DataFrame:** Creates a Pandas DataFrame named 'df' to organize the data. Converts the 'date' column to a datetime format.
  + **5. Separating User and Message:** Splits the 'user\_message' column into 'user' and 'message' columns. Handles cases where no username is present.
  + **6. Extracting Date Components:** Extracts various date components like year, month, day, day of the week, hour, and minute from the 'date' column. Adds these components as new columns to the DataFrame. Drops the original 'date' column.
  + **7. Function for Data Preprocessing:c**Encapsulates the entire data preprocessing process into a function named 'preprocess.' Allows for the reuse of this functionality in other scripts.

1. **functions.py:** The file consists of a collection of Python functions for analyzing and visualizing chat message data.

**Below is a summary of each function:**

* + **fetch\_stats(selected\_user, df):** Returns statistics based on the selected user and DataFrame. Calculates the number of messages, total words, media messages, and links.
  + **most\_busy\_users(df):** Finds the most active users based on the number of messages sent. Returns a count of messages sent by each user and their percentages.
  + **create\_wordcloud(selected\_user, df):** Generates a Word Cloud visualization of text messages. Filters out filler words for better visualization.
  + **most\_common\_words(selected\_user, df):** Finds the most common words in text messages. Filters out filler words for accurate results.
  + **emoji\_helper(selected\_user, df):** Extracts and counts emojis used in messages. Returns a DataFrame with emoji counts.
  + **monthly\_timeline(selected\_user, df):** Creates a monthly timeline of message counts. Groups messages by year and month.
  + **daily\_timeline(selected\_user, df):** Creates a daily timeline of message counts.Groups messages by date.
  + **week\_activity\_map(selected\_user, df):** Generates a weekly activity map. Counts the number of messages for each day of the week.
  + **month\_activity\_map(selected\_user, df):** Creates a monthly activity map. Counts the number of messages for each month.
  + **activity\_heatmap(selected\_user, df):** Generates an activity heatmap. Displays message counts by day and time period (hour).

These functions are designed to analyse chat message data, extract insights, and create visualizations. The code also imports necessary libraries for text analysis, visualization, and emoji processing.

the exact data several codes, such as df.shape (returns the nrow, and ncolumns), and df.head() (Gives the overview of the first five rows), were passed.

**End of project**