**Stepwise Methodology Description for the Code**

**Step 1: Importing Required Libraries**

The code begins by importing the necessary libraries:

* **streamlit**: Used to build the web interface for the chatbot application.
* **pandas**: Manages data storage and analysis, especially for mood tracking data.
* **matplotlib.pyplot**: Visualizes mood trends over time through plots.
* **datetime**: Captures timestamps for mood inputs.
* **textblob**: Performs sentiment analysis to determine user mood.

**Step 2: Mood Analysis**

* **Function: analyze\_mood(user\_input)**
  + The user's input is analyzed using **TextBlob**, which provides a polarity score for sentiment analysis:
    - **Polarity Range**: -1.0 (negative) to 1.0 (positive).
  + Based on the polarity value:
    - **Very Happy**: Polarity > 0.5
    - **Happy**: Polarity > 0 but ≤ 0.5
    - **Neutral**: Polarity == 0
    - **Sad**: Polarity < 0 but ≥ -0.5
    - **Very Sad**: Polarity < -0.5
  + The function returns the identified mood and the absolute polarity value as the confidence score.

**Step 3: Chatbot Responses**

* **Function: chatbot\_response(user\_input)**
  + Uses the analyze\_mood function to determine the mood and confidence.
  + Based on the identified mood:
    - Provides an empathetic, mood-specific response to the user.
    - Combines the chatbot's response with the mood label and confidence score.
  + Example:
    - **Input**: "I'm feeling down today."
    - **Output**: "I'm here to listen. It's okay to feel down sometimes. What's been on your mind?\n\nMood: Sad, Confidence: 0.40"

**Step 4: Saving Mood Data**

* **Function: save\_mood\_data(user\_input)**
  + Captures:
    - The user's input.
    - Timestamp (current date and time).
    - Mood and confidence score from the analyze\_mood function.
  + Stores this data in a **CSV file** (mood\_tracking\_data.csv) using **pandas**.
  + If the file does not exist, it adds headers; otherwise, it appends data.

**Step 5: Visualizing Mood Trends**

* **Function: plot\_mood\_trends()**
  + Reads data from mood\_tracking\_data.csv.
  + Groups the mood data by date to count the occurrences of each mood per day.
  + Uses a **stacked bar chart** to display daily mood trends with the following:
    - **X-axis**: Date.
    - **Y-axis**: Frequency of each mood.
    - **Legend**: Mood categories (e.g., Happy, Sad).
  + The visualization is displayed using **Streamlit's st.pyplot()**.

**Step 6: Streamlit App**

* The app is implemented in the following sections:
  1. **Title and Description**:
     + Displays the app title and a brief description.
  2. **User Input**:
     + Prompts the user to describe how they're feeling today using st.text\_input().
     + Analyzes the input:
       - Displays a chatbot response (mood\_response).
       - Saves the mood data for tracking.
  3. **Mood Trends Visualization**:
     + Provides a button (st.button()) to display the mood trends chart.
     + When clicked, it calls the plot\_mood\_trends() function.

**Step 7: Execution and Deployment**

* The code is executed as a Streamlit app.
* It provides:
  + Real-time mood analysis and responses.
  + A historical mood trend visualization based on collected data.

**Step 1: Selection of Frameworks and Tools**

The initial step in developing the AI-driven mental well-being chatbot involved selecting appropriate libraries and frameworks to meet the project’s functional and design requirements. Streamlit was chosen to create a user-friendly and interactive web application, providing a seamless interface for user interaction. For sentiment analysis, the TextBlob library was utilized, enabling precise extraction of polarity from user input text. The Pandas library was integrated for efficient data management, facilitating the storage, retrieval, and processing of mood-tracking data. To visualize mood trends, Matplotlib was employed to generate clear and interpretable charts. Finally, the Datetime module was incorporated to timestamp user interactions, allowing the application to track and analyze historical mood data over time. Together, these tools provided a robust foundation for implementing the chatbot’s functionality while ensuring scalability and user engagement.

**Step 2: Mood Analysis Mechanism**

The core functionality of the chatbot is its ability to analyze and categorize the mood of users based on their textual input. This is achieved using TextBlob, a sentiment analysis library that evaluates the polarity of user input on a scale ranging from -1.0 to 1.0. The polarity score is then used to classify user emotions into five distinct categories: Very Happy, Happy, Neutral, Sad, and Very Sad. For instance, a polarity greater than 0.5 is classified as Very Happy, while a polarity less than -0.5 indicates Very Sad. This systematic categorization allows the chatbot to accurately interpret the user’s emotional state. By combining polarity scores with a simple yet effective classification logic, the chatbot can provide contextually appropriate responses that reflect empathy and understanding.

**Step 3: Chatbot Response Generation**

Once the mood is analyzed, the chatbot generates personalized responses tailored to the user’s emotional state. The chatbot\_response() function maps each mood category to a predefined set of empathetic responses. For positive moods such as Very Happy and Happy, the chatbot encourages users to maintain their positivity and reflects on the causes of their happiness. Conversely, for Neutral moods, it acknowledges the user's state and opens the door for further conversation. For negative moods like Sad and Very Sad, the chatbot offers comforting words, encourages the user to express their feelings, and subtly reinforces that they are not alone. By aligning responses with user moods, the chatbot fosters meaningful interactions, ensuring users feel heard and supported.

**Step 4: Data Collection and Storage**

Tracking mood trends over time requires efficient data storage and management. The chatbot incorporates a save\_mood\_data() function that records user input, the detected mood category, the confidence score, and a timestamp. This data is appended to a CSV file, mood\_tracking\_data.csv, which serves as the system’s primary storage mechanism. By leveraging Pandas, the system ensures scalability, allowing it to handle large datasets with ease. Additionally, the use of timestamps enables chronological tracking, making it possible to analyze changes in mood over time. This step not only establishes a comprehensive dataset for the user but also sets the stage for advanced data-driven insights and trend analysis.

**Step 5: Visualization of Mood Trends**

To enhance the user experience and provide valuable insights, the chatbot includes a functionality to visualize historical mood trends. The plot\_mood\_trends() function processes the mood data stored in the CSV file, aggregating mood frequencies by date using Pandas. It then generates a stacked bar chart using Matplotlib, with dates on the X-axis and mood frequencies on the Y-axis. Each bar represents a day’s mood composition, offering a clear visual representation of emotional patterns over time. This visualization helps users identify fluctuations in their emotional states, potential triggers, and periods of stability. By offering an accessible and engaging way to track mood trends, this feature empowers users to gain deeper self-awareness and take proactive steps toward well-being.

**Step 6: Deployment as an Interactive Application**

The chatbot is deployed as a real-time interactive application using Streamlit, ensuring ease of use and accessibility for users. The application features a clean interface with a text input field where users can share their feelings. Based on the input, the chatbot provides mood-specific responses in real-time. Additionally, users can view historical mood trends by clicking a button to generate a visualization of their emotional data. Streamlit’s intuitive layout and interactive widgets make it easy for users to navigate the application and access its features. This deployment step bridges the gap between technology and users, transforming the chatbot from a concept into a functional tool for mental well-being.

**Step 7: Ethical Considerations**

Throughout the development process, ethical considerations were a top priority to ensure user trust and safety. The chatbot anonymizes all user data and stores it locally in a CSV file, safeguarding privacy and confidentiality. Transparency is maintained by providing users with clear explanations of how their moods are analyzed and categorized. Furthermore, the chatbot explicitly states its limitations and advises users to seek professional help for severe emotional distress. By embedding these ethical safeguards, the application ensures that it not only provides valuable support but also respects the sensitive nature of mental health data.

**Conclusion**

The stepwise methodology employed in the development of the AI-driven chatbot combines advanced natural language processing, data management, and visualization techniques to deliver a robust tool for mental well-being. By integrating empathetic interaction with actionable insights, the chatbot serves as a scalable and accessible solution to support users in understanding and managing their emotional states. This comprehensive approach highlights the potential of technology in addressing global mental health challenges.