A mental health tracking application

Project Description:
The project is a comprehensive Mood Journal app built using Tkinter, a Python library for creating graphical user interfaces. The app allows users to track their moods and emotions by selecting a mood from a dropdown menu, adding notes, and submitting the log. The app displays a log of all submitted moods and notes, and also provides features like calendar integration using tkcalendar, chart visualization using matplotlib, validation, storage using JSON, and notifications.

Objective:
The objective of the project is to create a user-friendly and feature-rich application that allows users to track their moods and emotions over time, providing insights and reminders to help them manage their mental health.

Process:
The process involved in the project includes:

1. Designing the application's user interface using Tkinter
2. Implementing the application's functionality, including creating a mood log, submitting moods and notes, displaying the log, calendar integration, chart visualization, validation, storage, and notifications
3. Testing the application to ensure it works as expected
4. Iterating and refining the application based on user feedback and testing results

Skills:
The skills used in the project include:

1. Programming skills in Python
2. Knowledge of Tkinter library and GUI development
3. Understanding of event-driven programming
4. Knowledge of matplotlib for data visualization
5. Understanding of JSON for data storage
6. Problem-solving and debugging skills

Tools:
The tools used in the project include:

1. Python programming language
2. Tkinter library for GUI development
3. tkcalendar library for calendar integration
4. matplotlib library for data visualization
5. JSON for data storage

Outcomes:
The outcomes of the project include:

1. A functional Mood Journal app that allows users to track their moods and emotions

2. A simple and user-friendly interface that makes it easy for users to interact with the application
3. A log of all submitted moods and notes that can be viewed by the user
4. Visual representation of mood data using charts
5. Reminders and notifications to help users manage their mental health
6. Data storage using JSON to persist user data

Features:
The features of the project include:

1. Mood logging with notes
2. Calendar integration
3. Chart visualization
4. Validation
5. Storage using JSON
6. Notifications
7. User-friendly GUI

Benefits:
The benefits of the project include:

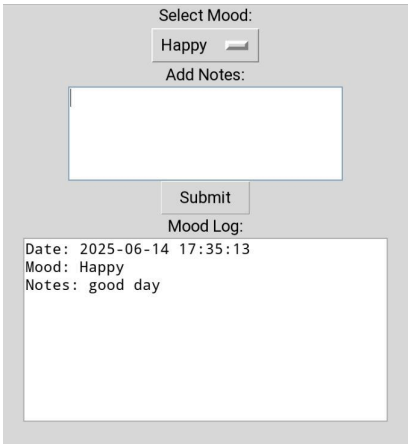
1. Helps users track their moods and emotions over time
2. Provides insights and reminders to help users manage their mental health
3. User-friendly interface makes it easy to use
4. Data visualization helps users understand their mood patterns

Here is a simple mood journal app using tkinter

```
import tkinter as tk
from tkinter import messagebox
from datetime import datetime

class MoodJournal:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Mood Journal")
        self.mood_log = []

        # Create mood dropdown
        self.mood_label = tk.Label(self.root, text="Select Mood:")
        self.mood_label.pack()
        self.mood_var = tk.StringVar(self.root)
        self.mood_var.set("Happy")
        self.mood_options = ["Happy", "Sad", "Anxious", "Calm"]
        self.mood_menu = tk.OptionMenu(self.root, self.mood_var, *self.mood_options)
        self.mood_menu.pack()

        # Create notes entry
```

```
        self.notes_label = tk.Label(self.root, text="Add Notes:")
        self.notes_label.pack()
        self.notes_entry = tk.Text(self.root, height=5, width=30)
        self.notes_entry.pack()

        # Create submit button
        self.submit_button = tk.Button(self.root, text="Submit", command=self.submit_mood)
        self.submit_button.pack()

        # Create log display
        self.log_label = tk.Label(self.root, text="Mood Log:")
        self.log_label.pack()
        self.log_text = tk.Text(self.root, height=10, width=40)
        self.log_text.pack()

    def submit_mood(self):
        mood = self.mood_var.get()
        notes = self.notes_entry.get("1.0", "end-1c")
        self.mood_log.append({
            "Date": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "Mood": mood,
            "Notes": notes
        })
        self.update_log()
        self.notes_entry.delete("1.0", "end")

    def update_log(self):
        self.log_text.delete("1.0", "end")
        for log in self.mood_log:
            self.log_text.insert("end", f"Date: {log['Date']}\nMood: {log['Mood']}\nNotes:
{log['Notes']}\n\n")

    def run(self):
        self.root.mainloop()

if __name__ == "__main__":
    app = MoodJournal()
    app.run()
```

Output



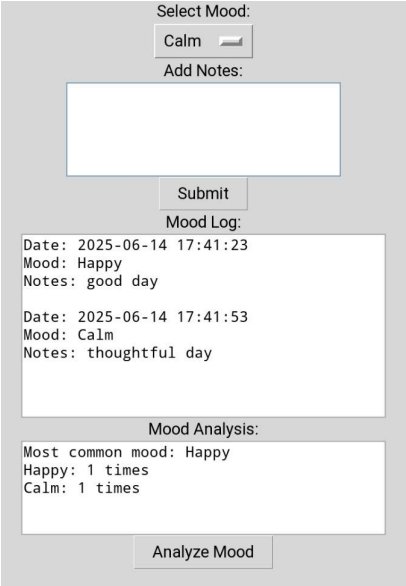
Now analysis of mood is happening

```
import tkinter as tk
from tkinter import messagebox
from datetime import datetime
from collections import Counter

class MoodJournal:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Mood Journal")
        self.mood_log = []

        # Create mood dropdown
        self.mood_label = tk.Label(self.root, text="Select Mood:")
        self.mood_label.pack()
        self.mood_var = tk.StringVar(self.root)
        self.mood_var.set("Happy")
        self.mood_options = ["Happy", "Sad", "Anxious", "Calm"]
        self.mood_menu = tk.OptionMenu(self.root, self.mood_var, *self.mood_options)
        self.mood_menu.pack()

        # Create notes entry
        self.notes_label = tk.Label(self.root, text="Add Notes:")
        self.notes_label.pack()
        self.notes_entry = tk.Text(self.root, height=5, width=30)
        self.notes_entry.pack()

        # Create submit button
        self.submit_button = tk.Button(self.root, text="Submit", command=self.submit_mood)
        self.submit_button.pack()

        # Create log display
        self.log_label = tk.Label(self.root, text="Mood Log:")
        self.log_label.pack()
```

```
        self.log_text = tk.Text(self.root, height=10, width=40)
        self.log_text.pack()

        # Create analysis display
        self.analysis_label = tk.Label(self.root, text="Mood Analysis:")
        self.analysis_label.pack()
        self.analysis_text = tk.Text(self.root, height=5, width=40)
        self.analysis_text.pack()

        # Create analysis button
        self.analysis_button = tk.Button(self.root, text="Analyze Mood",
command=self.analyze_mood)
        self.analysis_button.pack()

    def submit_mood(self):
        mood = self.mood_var.get()
        notes = self.notes_entry.get("1.0", "end-1c")
        self.mood_log.append({
            "Date": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "Mood": mood,
            "Notes": notes
        })
        self.update_log()
        self.notes_entry.delete("1.0", "end")

    def update_log(self):
        self.log_text.delete("1.0", "end")
        for log in self.mood_log:
            self.log_text.insert("end", f"Date: {log['Date']}\nMood: {log['Mood']}\nNotes:
{log['Notes']}\n\n")

    def analyze_mood(self):
        self.analysis_text.delete("1.0", "end")
        if not self.mood_log:
            self.analysis_text.insert("end", "No mood log data available.")
            return

        moods = [log["Mood"] for log in self.mood_log]
        most_common_mood = Counter(moods).most_common(1)[0][0]
        self.analysis_text.insert("end", f"Most common mood: {most_common_mood}\n")

        mood_counts = Counter(moods)
        for mood, count in mood_counts.items():
            self.analysis_text.insert("end", f"{mood}: {count} times\n")

    def run(self):
        self.root.mainloop()
```

```
if __name__ == "__main__":
    app = MoodJournal()
    app.run()
```

Output



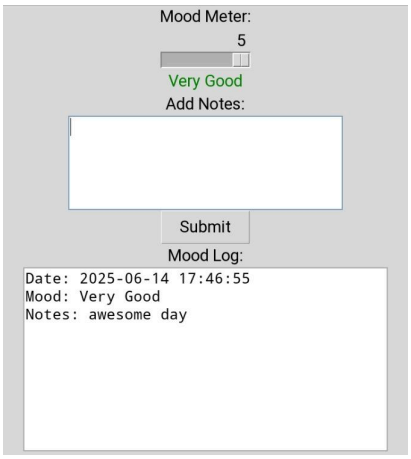
Mood meter is added

```
import tkinter as tk
from tkinter import messagebox
from datetime import datetime

class MoodJournal:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Mood Journal")
        self.mood_log = []

        # Create mood meter
        self.mood_meter_label = tk.Label(self.root, text="Mood Meter:")
        self.mood_meter_label.pack()
        self.mood_meter = tk.Scale(self.root, from_=1, to=5, orient=tk.HORIZONTAL,
command=self.update_mood_meter)
        self.mood_meter.pack()
        self.mood_meter_label_value = tk.Label(self.root, text="")
        self.mood_meter_label_value.pack()

        # Create notes entry
        self.notes_label = tk.Label(self.root, text="Add Notes:")
        self.notes_label.pack()
        self.notes_entry = tk.Text(self.root, height=5, width=30)
        self.notes_entry.pack()
```

```
    # Create submit button
    self.submit_button = tk.Button(self.root, text="Submit", command=self.submit_mood)
    self.submit_button.pack()

    # Create log display
    self.log_label = tk.Label(self.root, text="Mood Log:")
    self.log_label.pack()
    self.log_text = tk.Text(self.root, height=10, width=40)
    self.log_text.pack()

def update_mood_meter(self, value):
    value = int(value)
    if value == 1:
        self.mood_meter_label_value.config(text="Very Bad", fg="red")
    elif value == 2:
        self.mood_meter_label_value.config(text="Bad", fg="orange")
    elif value == 3:
        self.mood_meter_label_value.config(text="Neutral", fg="yellow")
    elif value == 4:
        self.mood_meter_label_value.config(text="Good", fg="lightgreen")
    elif value == 5:
        self.mood_meter_label_value.config(text="Very Good", fg="green")

def submit_mood(self):
    mood_value = self.mood_meter.get()
    notes = self.notes_entry.get("1.0", "end-1c")
    self.mood_log.append({
        "Date": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "Mood": mood_value,
        "Notes": notes
    })
    self.notes_entry.delete("1.0", "end")
    self.display_log()

def display_log(self):
    self.log_text.delete("1.0", "end")
    for log in self.mood_log:
        mood_text = ""
        if log['Mood'] == 1:
            mood_text = "Very Bad"
        elif log['Mood'] == 2:
            mood_text = "Bad"
        elif log['Mood'] == 3:
            mood_text = "Neutral"
        elif log['Mood'] == 4:
            mood_text = "Good"
        elif log['Mood'] == 5:
            mood_text = "Very Good"
```

```
        self.log_text.insert("end", f"Date: {log['Date']}\nMood: {mood_text}\nNotes: {log['Notes']}\n\n")

    def run(self):
        self.root.mainloop()

if __name__ == "__main__":
    app = MoodJournal()
    app.run()
```

Output



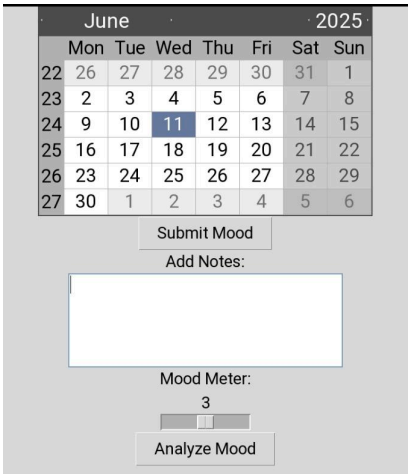
Now a user interface is created and data showing the mood changes on calendar
With plots

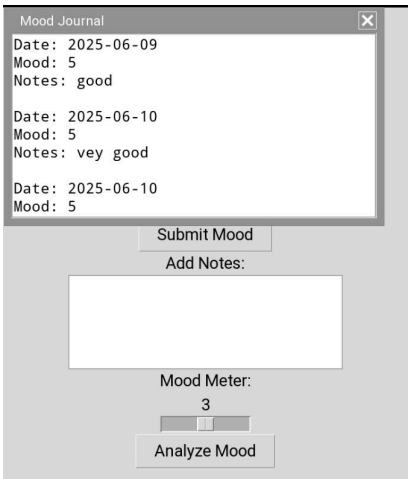
```
import tkinter as tk
from tkcalendar import Calendar
from datetime import datetime
import matplotlib.pyplot as plt

class MoodJournal:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Mood Journal")
        self.mood_log = []
        self.calendar = Calendar(self.root, selectmode='day', year=datetime.now().year, month=datetime.now().month, day=datetime.now().day)
        self.calendar.pack()
        self.submit_button = tk.Button(self.root, text="Submit Mood", command=self.submit_mood)
        self.submit_button.pack()
        self.notes_label = tk.Label(self.root, text="Add Notes:")
        self.notes_label.pack()
        self.notes_entry = tk.Text(self.root, height=5, width=30)
        self.notes_entry.pack()
        self.mood_meter_label = tk.Label(self.root, text="Mood Meter:")
        self.mood_meter_label.pack()
```

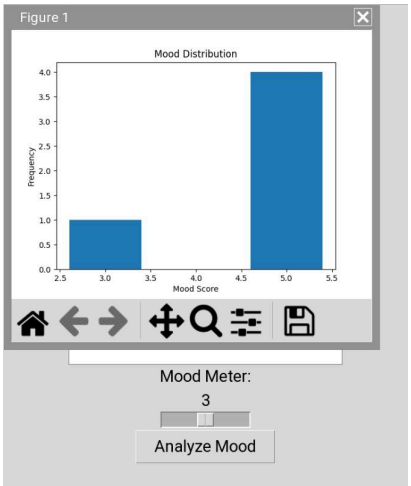
```
        self.mood_meter = tk.Scale(self.root, from_=1, to=5, orient=tk.HORIZONTAL)
        self.mood_meter.pack()
        self.analyze_button = tk.Button(self.root, text="Analyze Mood",
command=self.analyze_mood)
        self.analyze_button.pack()

    def submit_mood(self):
        date = self.calendar.selection_get()
        notes = self.notes_entry.get("1.0", "end-1c")
        mood_value = self.mood_meter.get()
        self.mood_log.append({
            "Date": date,
            "Mood": mood_value,
            "Notes": notes
        })
        self.notes_entry.delete("1.0", "end")
        self.display_log()

    def display_log(self):
        log_window = tk.Toplevel(self.root)
        log_text = tk.Text(log_window, height=10, width=40)
        log_text.pack()
        for log in self.mood_log:
            log_text.insert("end", f"Date: {log['Date']}\nMood: {log['Mood']}\nNotes:
{log['Notes']}\n\n")

    def analyze_mood(self):
        if not self.mood_log:
            return

        # Calculate average mood score
        average_mood = sum(log["Mood"] for log in self.mood_log) / len(self.mood_log)
        print(f"Average mood score: {average_mood}")

        # Display mood distribution
        mood_counts = {}
        for log in self.mood_log:
            mood = log["Mood"]
            if mood in mood_counts:
                mood_counts[mood] += 1
            else:
                mood_counts[mood] = 1
        plt.bar(mood_counts.keys(), mood_counts.values())
        plt.xlabel("Mood Score")
        plt.ylabel("Frequency")
        plt.title("Mood Distribution")
        plt.show()
```

```
        # Display mood trend
        dates = [log["Date"] for log in self.mood_log]
        moods = [log["Mood"] for log in self.mood_log]
        plt.plot(dates, moods)
        plt.xlabel("Date")
        plt.ylabel("Mood Score")
        plt.title("Mood Trend")
        plt.show()

    def run(self):
        self.root.mainloop()

if __name__ == "__main__":
    app = MoodJournal()
    app.run()
```

Output







a simple backend process allows users to
Create , get all the mood logs and delete it
Also added json for storage show notifications

```
import json
import os
import time

class MoodLog:
    def __init__(self, mood, notes):
        self.mood = mood
        self.notes = notes

    def to_dict(self):
        return {"mood": self.mood, "notes": self.notes}

    @classmethod
    def from_dict(cls, data):
        return cls(data["mood"], data["notes"])

class MoodLogStorage:
    def __init__(self, filename):
        self.filename = filename
        self.load()

    def load(self):
        if os.path.exists(self.filename) and os.path.getsize(self.filename) > 0:
            with open(self.filename, "r") as file:
                try:
                    self.mood_logs = [MoodLog.from_dict(data) for data in json.load(file)]
                except json.JSONDecodeError:
                    self.mood_logs = []
        else:
```

```
        self.mood_logs = []

    def save(self):
        with open(self.filename, "w") as file:
            json.dump([mood_log.to_dict() for mood_log in self.mood_logs], file, indent=4)

    def add_mood_log(self, mood_log):
        self.mood_logs.append(mood_log)
        self.save()

    def get_mood_logs(self):
        return self.mood_logs

def create_mood_log(mood, notes):
    return MoodLog(mood, notes)

def validate_mood_log(mood_log):
    errors = []
    if not mood_log.mood:
        errors.append("Mood is required")
    elif len(mood_log.mood) < 1 or len(mood_log.mood) > 50:
        errors.append("Mood should be between 1 and 50 characters")
    if not mood_log.notes:
        errors.append("Notes are required")
    elif len(mood_log.notes) < 1 or len(mood_log.notes) > 200:
        errors.append("Notes should be between 1 and 200 characters")
    return errors

def view_mood_logs(storage):
    mood_logs = storage.get_mood_logs()
    if not mood_logs:
        print("No mood logs available")
    else:
        print("Mood Logs:")
        for i, mood_log in enumerate(mood_logs, start=1):
            print(f"Mood Log {i}:")
            print(f"Mood: {mood_log.mood}")
            print(f"Notes: {mood_log.notes}")
            print()
        print("JSON Data:")
        with open(storage.filename, "r") as file:
            print(file.read())

def main():
    storage = MoodLogStorage("mood_logs.json")
    notification_interval = None
    last_notification_time = None
    while True:
```

```
        print("1. Create Mood Log")
        print("2. View Mood Logs")
        print("3. Set Notification Interval")
        print("4. Quit")
        if notification_interval is not None and last_notification_time is not None:
            time_elapsed = time.time() - last_notification_time
            if time_elapsed >= notification_interval * 60:
                print("Reminder: Log your mood!")
                last_notification_time = time.time()
            else:
                time_remaining = notification_interval * 60 - time_elapsed
                minutes_remaining = int(time_remaining / 60)
                seconds_remaining = int(time_remaining % 60)
                print(f"Next notification in {minutes_remaining} minutes and {seconds_remaining}
seconds")
        choice = input("Choose an option: ")
        if choice == "1":
            mood = input("Enter your mood: ")
            notes = input("Enter your notes: ")
            mood_log = create_mood_log(mood, notes)
            errors = validate_mood_log(mood_log)
            if errors:
                print("Validation errors:")
                for error in errors:
                    print(error)
            else:
                storage.add_mood_log(mood_log)
                print("Mood log created successfully")
        elif choice == "2":
            view_mood_logs(storage)
        elif choice == "3":
            interval = int(input("Enter notification interval in minutes: "))
            notification_interval = interval
            last_notification_time = time.time()
            print("Notification interval set successfully")
        elif choice == "4":
            break
        else:
            print("Invalid option. Please choose again.")

if __name__ == "__main__":
    main()
```

Output

```
1. Create Mood Log
2. View Mood Logs
3. Set Notification Interval
4. Quit
Choose an option: 1
Enter your mood: happy
Enter your notes: great day
Mood log created successfully
1. Create Mood Log
2. View Mood Logs
3. Set Notification Interval
4. Quit
Choose an option: 3
Enter notification interval in minutes: 3
Notification interval set successfully
1. Create Mood Log
2. View Mood Logs
3. Set Notification Interval
4. Quit
Next notification in 2 minutes and 59 seconds
Choose an option: 4

[Program finished]
```

Overall, the project provides a comprehensive and user-friendly platform for users to track their moods and emotions, and can be used as a starting point for more complex mood tracking applications.