

fitnesstracker.py

```
1 import csv
2 import os
3 from datetime import datetime
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 import matplotlib.dates as mdates
7
8 CELESTIAL_PATH_TO_DATA = 'fitness_data.csv'
9
10 def initialize_data_chronicle():
11     """
12     Ensures the sacred scroll (CSV file) exists with the correct headers.
13     If the file is not found, it is created.
14     """
15     if not os.path.exists(CELESTIAL_PATH_TO_DATA):
16         with open(CELESTIAL_PATH_TO_DATA, 'w', newline='') as scroll:
17             scribe = csv.writer(scroll)
18             scribe.writerow(['Timestamp', 'StepsCount', 'CaloriesBurned', 'WorkoutDurationMinutes'])
19
20 def record_daily_metrics():
21     """
22     Gathers and records the champion's daily efforts into the data chronicle.
23     Includes robust validation for all user inputs.
24     """
25     print("\n--- Record Your Daily Triumph ---")
26     while True:
27         try:
28             date_input_str = input("Enter the date of your activity (YYYY-MM-DD): ")
29             validated_date = datetime.strptime(date_input_str, '%Y-%m-%d').strftime('%Y-%m-%d')
30             break
31         except ValueError:
32             print("Invalid date format. Please use YYYY-MM-DD.")
33
34     while True:
35         try:
36             quantum_of_locomotion = int(input("Enter the total steps taken: "))
37             if quantum_of_locomotion < 0:
38                 raise ValueError("Steps cannot be negative.")
39             break
40         except ValueError:
41             print("Invalid input. Please enter a positive whole number for steps.")
42
43     while True:
44         try:
45             quantum_of_energy = int(input("Enter the total calories burned: "))
46             if quantum_of_energy < 0:
47                 raise ValueError("Calories cannot be negative.")
48             break
49         except ValueError:
50             print("Invalid input. Please enter a positive whole number for calories.")
51
52     while True:
53         try:
54             temporal_expanse_of_effort = int(input("Enter the duration of exercise (in minutes): "))
55             if temporal_expanse_of_effort < 0:
56                 raise ValueError("Duration cannot be negative.")
57             break
58         except ValueError:
59             print("Invalid input. Please enter a positive whole number for duration.")
60
61     with open(CELESTIAL_PATH_TO_DATA, 'a', newline='') as scroll:
62         scribe = csv.writer(scroll)
63         scribe.writerow([validated_date, quantum_of_locomotion, quantum_of_energy, temporal_expanse_of_effort])
64
65     print("\nYour Data has been added")
66
67 def perform_data_analysis():
68     """
69     Reads the chronicle of efforts and reveals profound insights.
70     Displays aggregate statistics and a detailed summary.
71     """
72     print("\n--- Analyzing the Saga of Your Efforts ---")
73     try:
74         chronicle_of_efforts = pd.read_csv(CELESTIAL_PATH_TO_DATA)
75         if chronicle_of_efforts.empty:
76             print("No Data Found. Record some data first.")
77             return
78
79         total_steps = chronicle_of_efforts['StepsCount'].sum()
80         total_calories = chronicle_of_efforts['CaloriesBurned'].sum()
81         average_duration = chronicle_of_efforts['WorkoutDurationMinutes'].mean()
82
83         print(f"\nTotal Steps Forged: {total_steps:,}")
84         print(f"Total Calories Obliterated: {total_calories:,}")
85         print(f"Average Workout Duration: {average_duration:.2f} minutes")
86
87     print("\n--- Comprehensive Statistical Overview ---")
88     print(chronicle_of_efforts.describe())
```

```

90     except FileNotFoundError:
91         print("The data (fitness_data.csv) is not found. Please add data first.")
92     except Exception as e:
93         print(f"An unexpected error occurred during analysis: {e}")
94
95 def visualize_fitness_progress():
96     """
97     Translates raw data into beautiful, inspiring visual tapestries.
98     Plots weekly steps and long-term calorie expenditure.
99     """
100    print("\n--- Visualizing Data ---")
101    try:
102        chronicle_of_efforts = pd.read_csv(CELESTIAL_PATH_TO_DATA)
103        if chronicle_of_efforts.empty:
104            print("No data to visualize.")
105            return
106
107        chronicle_of_efforts['Timestamp'] = pd.to_datetime(chronicle_of_efforts['Timestamp'])
108
109        weekly_data = chronicle_of_efforts.tail(7)
110        if not weekly_data.empty:
111            plt.style.use('seaborn-v0.8-darkgrid')
112            fig1, ax1 = plt.subplots(figsize=(12, 7))
113
114            dates_as_str = weekly_data['Timestamp'].dt.strftime('%Y-%m-%d')
115            colors = ['#4A90E2'] * len(weekly_data)
116
117            peak_performance_index = weekly_data['StepsCount'].idxmax()
118            if pd.notna(peak_performance_index) and peak_performance_index in weekly_data.index:
119                peak_idx_pos = weekly_data.index.get_loc(peak_performance_index)
120                colors[peak_idx_pos] = '#F5A623'
121
122            bars = ax1.bar(dates_as_str, weekly_data['StepsCount'], color=colors)
123
124            ax1.set_title('Champion\'s Steps: Last 7 Days', fontsize=18, fontweight='bold', color='#333')
125            ax1.set_xlabel('Date', fontsize=12, fontweight='bold')
126            ax1.set_ylabel('Steps Taken', fontsize=12, fontweight='bold')
127            plt.xticks(rotation=45, ha='right')
128
129            from matplotlib.patches import Patch
130            legend_elements = [Patch(facecolor='#4A90E2', edgecolor='black', label='Daily Steps'),
131                               Patch(facecolor='#F5A623', edgecolor='black', label='Peak Performance Day')]
132            ax1.legend(handles=legend_elements)
133
134            plt.tight_layout()
135            print("Displaying bar chart of recent steps...")
136            plt.show()
137
138        fig2, ax2 = plt.subplots(figsize=(12, 7))
139        ax2.plot(chronicle_of_efforts['Timestamp'], chronicle_of_efforts['CaloriesBurned'], marker='o', linestyle='--', color='#D0021B', label='Calories Burned')
140
141        ax2.set_title('Caloric Expenditure Over Time', fontsize=18, fontweight='bold', color='#333')
142        ax2.set_xlabel('Date', fontsize=12, fontweight='bold')
143        ax2.set_ylabel('Calories Burned', fontsize=12, fontweight='bold')
144
145        ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
146        ax2.xaxis.set_major_locator(mdates.AutoDateLocator())
147        fig2.autofmt_xdate()
148
149        ax2.grid(True, which='both', linestyle='--', linewidth=0.5)
150        ax2.legend()
151        plt.tight_layout()
152        print("Displaying line chart of calories burned...")
153        plt.show()
154
155    except FileNotFoundError:
156        print("The data (fitness_data.csv) is not found.")
157    except Exception as e:
158        print(f"A visualization error occurred: {e}")
159
160 def main():
161     """
162     Presents a menu to navigate the application's features.
163     """
164     initialize_data_chronicle()
165     while True:
166         print("\n=====")
167         print("    Inside the Mind and Muscles of Champions    ")
168         print("    Fitness Tracking & Visualization            ")
169         print("=====")
170         print("1. Add New Fitness Data")
171         print("2. Analyze Fitness Data")
172         print("3. Visualize Fitness Progress")
173         print("4. Exit the Application")
174         print("-----")
175
176         choice = input("Choose your option (1-4): ")
177
178         if choice == '1':

```

```
179         record_daily_metrics()
180     elif choice == '2':
181         perform_data_analysis()
182     elif choice == '3':
183         visualize_fitness_progress()
184     elif choice == '4':
185         print("\nExiting Application...")
186         break
187     else:
188         print("\nInvalid choice. Please enter a number between 1 and 4.")
189
190 if __name__ == "__main__":
191     main()
192
```