```python
import csv
import os
from datetime import datetime
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

CELESTIAL_PATH_TO_DATA = 'fitness_data.csv'

def initialize_data_chronicle():
    """
    Ensures the sacred scroll (CSV file) exists with the correct headers.
    If the file is not found, it is created.
    """
    if not os.path.exists(CELESTIAL_PATH_TO_DATA):
        with open(CELESTIAL_PATH_TO_DATA, 'w', newline='') as scroll:
            scribe = csv.writer(scroll)
            scribe.writerow(['Timestamp', 'StepsCount', 'CaloriesBurned', 'WorkoutDurationMinutes'])

def record_daily_metrics():
    """
    Gathers and records the champion's daily efforts into the data chronicle.
    Includes robust validation for all user inputs.
    """
    print("\n--- Record Your Daily Triumph ---")
    while True:
        try:
            date_input_str = input("Enter the date of your activity (YYYY-MM-DD): ")
            validated_date = datetime.strptime(date_input_str, '%Y-%m-%d').strftime('%Y-%m-%d')
            break
        except ValueError:
            print("Invalid date format. Please use YYYY-MM-DD.")

    while True:
        try:
            quantum_of_locomotion = int(input("Enter the total steps taken: "))
            if quantum_of_locomotion < 0:
                raise ValueError("Steps cannot be negative.")
            break
        except ValueError:
            print("Invalid input. Please enter a positive whole number for steps.")

    while True:
        try:
            quantum_of_energy = int(input("Enter the total calories burned: "))
            if quantum_of_energy < 0:
                raise ValueError("Calories cannot be negative.")
            break
        except ValueError:
            print("Invalid input. Please enter a positive whole number for calories.")

    while True:
        try:
            temporal_expanse_of_effort = int(input("Enter the duration of exercise (in minutes): "))
            if temporal_expanse_of_effort < 0:
                raise ValueError("Duration cannot be negative.")
            break
        except ValueError:
            print("Invalid input. Please enter a positive whole number for duration.")

    with open(CELESTIAL_PATH_TO_DATA, 'a', newline='') as scroll:
        scribe = csv.writer(scroll)
        scribe.writerow([validated_date, quantum_of_locomotion, quantum_of_energy, temporal_expanse_of_effort])

    print("\nYour Data has been added")

def perform_data_analysis():
    """
    Reads the chronicle of efforts and reveals profound insights.
    Displays aggregate statistics and a detailed summary.
    """
    print("\n--- Analyzing the Saga of Your Efforts ---")
    try:
        chronicle_of_efforts = pd.read_csv(CELESTIAL_PATH_TO_DATA)
        if chronicle_of_efforts.empty:
            print("No Data Found. Record some data first.")
            return

        total_steps = chronicle_of_efforts['StepsCount'].sum()
        total_calories = chronicle_of_efforts['CaloriesBurned'].sum()
        average_duration = chronicle_of_efforts['WorkoutDurationMinutes'].mean()

        print(f"\nTotal Steps Forged: {total_steps:,}")
        print(f"Total Calories Obliterated: {total_calories:,}")
        print(f"Average Workout Duration: {average_duration:.2f} minutes")

        print("\n--- Comprehensive Statistical Overview ---")
        print(chronicle_of_efforts.describe())

```

```
    except FileNotFoundError:
        print("The data (fitness_data.csv) is not found. Please add data first.")
    except Exception as e:
        print(f"An unexpected error occurred during analysis: {e}")

def visualize_fitness_progress():
    """
    Translates raw data into beautiful, inspiring visual tapestries.
    Plots weekly steps and long-term calorie expenditure.
    """
    print("\n--- Visualizing Data ---")
    try:
        chronicle_of_efforts = pd.read_csv(CELESTIAL_PATH_TO_DATA)
        if chronicle_of_efforts.empty:
            print("No data to visualize.")
            return

        chronicle_of_efforts['Timestamp'] = pd.to_datetime(chronicle_of_efforts['Timestamp'])

        weekly_data = chronicle_of_efforts.tail(7)
        if not weekly_data.empty:
            plt.style.use('seaborn-v0_8-darkgrid')
            fig1, ax1 = plt.subplots(figsize=(12, 7))

            dates_as_str = weekly_data['Timestamp'].dt.strftime('%Y-%m-%d')
            colors = ['#4A90E2'] * len(weekly_data)

            peak_performance_index = weekly_data['StepsCount'].idxmax()
            if pd.notna(peak_performance_index) and peak_performance_index in weekly_data.index:
                peak_idx_pos = weekly_data.index.get_loc(peak_performance_index)
                colors[peak_idx_pos] = '#F5A623'

            bars = ax1.bar(dates_as_str, weekly_data['StepsCount'], color=colors)

            ax1.set_title('Champion\'s Steps: Last 7 Days', fontsize=18, fontweight='bold', color='#333')
            ax1.set_xlabel('Date', fontsize=12, fontweight='bold')
            ax1.set_ylabel('Steps Taken', fontsize=12, fontweight='bold')
            plt.xticks(rotation=45, ha='right')

            from matplotlib.patches import Patch
            legend_elements = [Patch(facecolor='#4A90E2', edgecolor='black', label='Daily Steps'),
                               Patch(facecolor='#F5A623', edgecolor='black', label='Peak Performance Day')]
            ax1.legend(handles=legend_elements)

            plt.tight_layout()
            print("Displaying bar chart of recent steps...")
            plt.show()

        fig2, ax2 = plt.subplots(figsize=(12, 7))
        ax2.plot(chronicle_of_efforts['Timestamp'], chronicle_of_efforts['CaloriesBurned'], marker='o', linestyle='-', color='#D0021B', label='Calories Burned')

        ax2.set_title('Caloric Expenditure Over Time', fontsize=18, fontweight='bold', color='#333')
        ax2.set_xlabel('Date', fontsize=12, fontweight='bold')
        ax2.set_ylabel('Calories Burned', fontsize=12, fontweight='bold')

        ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
        ax2.xaxis.set_major_locator(mdates.AutoDateLocator())
        fig2.autofmt_xdate()

        ax2.grid(True, which='both', linestyle='--', linewidth=0.5)
        ax2.legend()
        plt.tight_layout()
        print("Displaying line chart of calories burned...")
        plt.show()

    except FileNotFoundError:
        print("The data (fitness_data.csv) is not found.")
    except Exception as e:
        print(f"A visualization error occurred: {e}")

def main():
    """
    Presents a menu to navigate the application's features.
    """
    initialize_data_chronicle()
    while True:
        print("\n==========================================")
        print("   Inside the Mind and Muscles of Champions  ")
        print("      Fitness Tracking & Visualization       ")
        print("==========================================")
        print("1. Add New Fitness Data")
        print("2. Analyze Fitness Data")
        print("3. Visualize Fitness Progress")
        print("4. Exit the Application")
        print("------------------------------------------")

        choice = input("Choose your option (1-4): ")

        if choice == '1':
            record_daily_metrics()
```

```
        elif choice == '2':
            perform_data_analysis()
        elif choice == '3':
            visualize_fitness_progress()
        elif choice == '4':
            print("\nExiting Application...")
            break
        else:
            print("\nInvalid choice. Please enter a number between 1 and 4.")

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