Project Report: Weekly Study Performance Tracker

Section	**Detail**
Project Name	Weekly Study Performance Tracker and Analyzer
Version	1.0
Author	Ishant kumar sahu

1. Executive Summary

This project implements a command-line utility designed to help students track and analyze their weekly study habits. The application collects daily study hours, calculates key performance metrics, provides immediate, context-aware performance feedback, and visualizes the data using a bar chart. It serves as a simple yet effective tool for self-assessment and habit improvement.

2. Project Goals and Objectives

The primary objectives of this script were:

1. **Data Collection:** Systematically collect numerical study hour data for each day of the week (Monday through Sunday).
2. **Data Validation:** Implement robust input handling to ensure only valid, non-negative numerical data is accepted.
3. **Metric Calculation:** Compute essential performance indicators, including total study hours, daily average, and identifying the highest and lowest study days.
4. **Feedback Generation:** Provide qualitative performance reviews based on the calculated average study hours.
5. **Data Visualization:** Generate a clear, insightful bar graph of daily study hours using the matplotlib library.

3. Detailed Functionality and Features

3.1 Input Handling and Validation

The script iterates through the seven days of the week, prompting the user for input. It incorporates a while loop and a try-except block to handle potential errors:

- **Type Error:** Catches ValueError if the input is not a valid number.
- **Range Error:** Checks if the entered number is negative and prompts the user to re-enter valid data.

3.2 Performance Metrics

The application calculates the following key metrics for the study period:

- **Total Hours Studied:** The sum of all study hours across the week.
- **Average Hours per Day:** The total hours divided by seven.
- **Maximum Hours:** The single highest number of hours studied in a day, along with the corresponding day of the week.
- **Minimum Hours:** The single lowest number of hours studied in a day, along with the corresponding day of the week.

3.3 Performance Review System

A three-tiered qualitative review system provides immediate feedback based on the calculated average daily hours:

Average Hours Range	Performance Level	Review Content
< 2.0	Poor/Needs Improvement	Focuses on increasing study time and establishing better habits.
2.0 <= Avg < 5.0	Average/Good Effort	Acknowledges solid effort while encouraging more consistency.
>= 5.0	Excellent/High Achiever	Commends dedication and suggests maintaining the high standard.

3.4 Data Visualization

The script utilizes the matplotlib.pyplot library to generate a professional bar chart.

- **Chart Type:** Bar chart, which is ideal for comparing discrete categories (Days) against a numerical value (Hours).
- **Aesthetics:** Uses a skyblue color theme, clear axis labels (Days of the Week, Study Hours), and a descriptive title (Daily Study Hours for the Week).
- **Readability:** X-axis labels are rotated by 45 degrees (plt.xticks(rotation=45)) to prevent overlapping, ensuring readability on various screen sizes.

4. Technical Specifications

Component	Detail
Language	Python 3.x

External Library	matplotlib (for data visualization)
Data Structure	List (hours) for storing study hours, used for sequential data storage.
Output	Command-line text report and a graphical window displaying the bar chart.

5. Conclusion and Future Enhancements

The Weekly Study Performance Tracker successfully meets all outlined objectives, providing users with both numerical data and a clear visualization of their study performance.

Potential Future Enhancements:

1. **Persistence:** Implement data storage (e.g., using CSV, JSON, or a simple SQLite database) to save historical performance data instead of requiring re-entry every time.
2. **Goal Tracking:** Allow the user to set a weekly target goal and include a metric showing the percentage of the goal achieved.
3. **Advanced Visualization:** Add a line graph overlaying the daily average to provide a better visual comparison, or generate a pie chart of the contribution of each day to the total hours.
4. **User Interface:** Develop a simple graphical user interface (GUI) using libraries like Tkinter or Streamlit to enhance user interaction beyond the command line.