

Class Track: Student Academic Performance Analyzer

Name: Harshit Kumar

Registration No: 25BAI11444

Course: Introduction to Problem Solving and Programming (CSE1021)

Submission Date: November 22, 2025

1. Introduction

In college, keeping track of grades across multiple semesters can be confusing. Most student portals only show a list of marks, but they don't show you if your performance is going up or down over time. I realized that having a visual tool would help students understand their academic standing better. This project, Class Track, is a personal desktop application that helps students record their marks, calculate their weighted GPA automatically, and visualize their growth trend using graphs. It also includes a feature to compare performance with friends or the class average.

2. Problem Statement

Students often lack a centralized, offline tool to analyse their academic performance in depth. While university portals provide raw data, they rarely provide visual analytics or predictive features. This makes it hard for students to identify weak areas or plan for the next semester. The goal of this project was to build a Python-based tool that solves this by providing instant GPA calculations and visual performance insights.

3. Functional Requirements

Based on the project scope, the system includes the following functions:

- **Authentication:** The user must be able to log in using a valid Student ID.
- **Data Management:** The system must allow adding semester-wise marks (Subject, Credits, Marks) to a CSV file.
- **GPA Calculation:** The system must automatically calculate the SGPA for each semester using the credit-weightage formula.

- **Visualization:** The system must generate a line graph showing the GPA trend over 6 semesters.
- **Comparison:** The system must allow the user to compare their grades against a friend's grades on a single graph.

4. Non-Functional Requirements

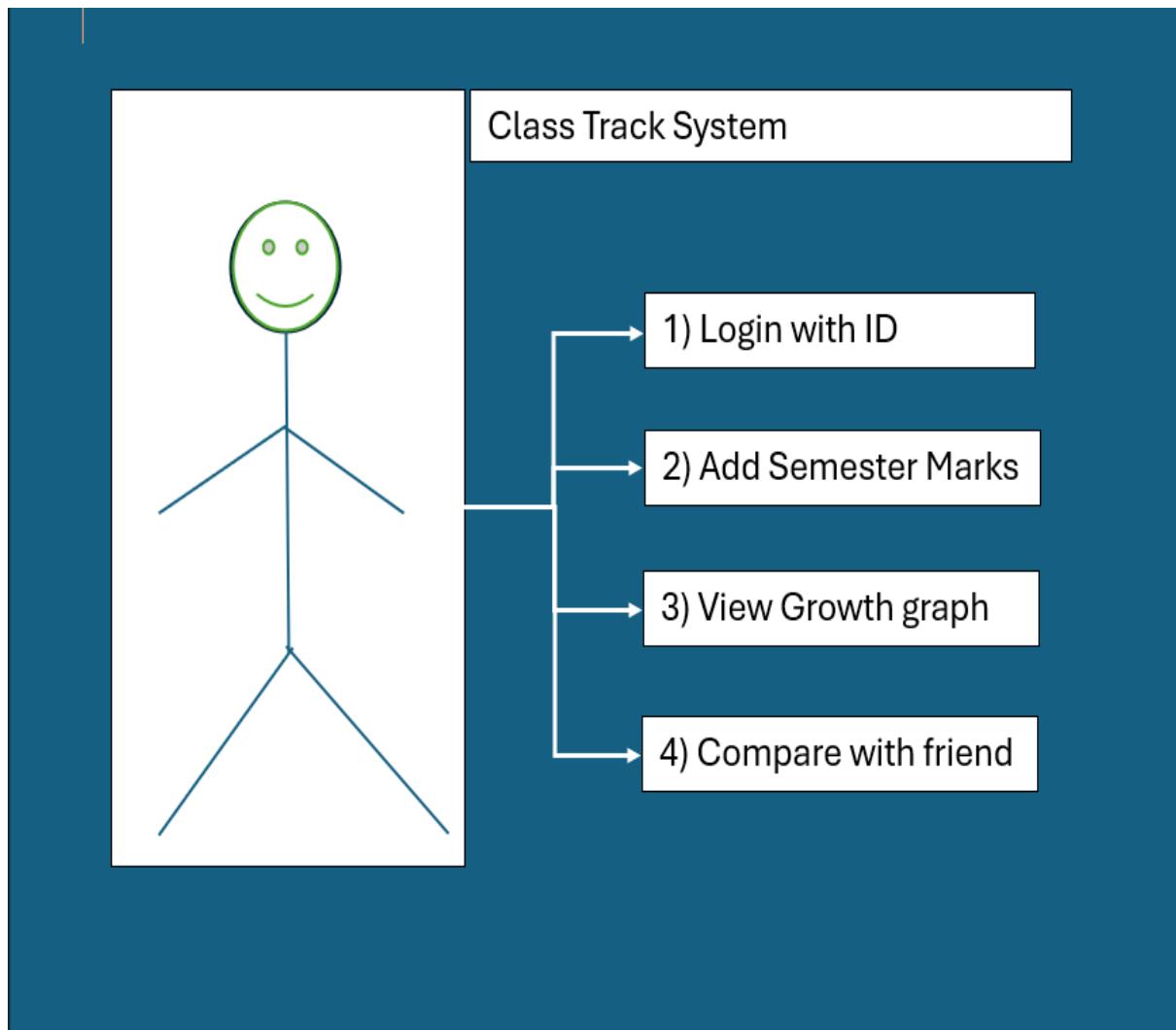
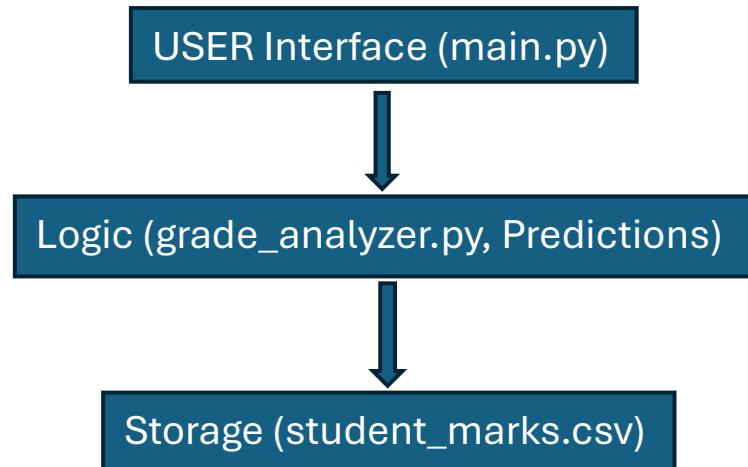
To ensure a good user experience, I focused on these quality attributes:

- **Usability:** The command-line interface (CLI) is simple text-based, so anyone can use it without training.
- **Reliability:** Data is saved to the CSV file immediately after entry, so no data is lost if the program closes.
- **Error Handling:** The system detects invalid inputs (like typing letters where numbers are required) and asks the user to try again instead of crashing.
- **Portability:** The project uses standard Python libraries, so it can run on any laptop with Python installed.

5. System Architecture

The system follows a modular design with three main layers:

1. **User Interface (Main Menu):** This handles all user inputs and displays the options.
2. **Logic Layer (Analyzer):** This performs the math (calculating averages and predictions).
3. **Data Layer (Storage):** This reads and writes to the 'student_marks.csv' file.



Workflow Diagram:

- 1) Start Application
- 2) Enter Student ID -> Verify -> Login Success
- 3) Show Dashboard Menu
- 4) User Selects Option (e.g., View Graph)
- 5) System Processes Data -> Displays Output
- 6) Loop back to Menu until "Exit" is chosen.
- 7) Design Decisions & Rationale
 - a) Why Python? I chose Python because it has excellent libraries like pandas for handling data and matplotlib for plotting graphs. It is also the language we are focusing on in our course.
 - b) Why CSV? I used a CSV file instead of a database (like SQL) because it is lightweight, easy to open in Excel to verify data, and doesn't require complex setup on the evaluator's machine.
 - c) Modular Code: I split the code into separate files (data_manager.py, visualizer.py) instead of writing one giant file. This makes the code easier to debug and keep organized.

9. Implementation Details

The project is implemented using Python 3.13 with the following structure:

- 1) src/main.py: The entry point that handles the user menu loop and input validation.
- 2) src/data_manager.py: Uses pandas to read/write data to student_marks.csv. It filters records based on Student ID.
- 3) src/visualizer.py: Contains functions like plot_gpa_graph() and plt_peer_compare() that utilize matplotlib.pyplot for rendering graphs.
- 4) src/grade_analyzer.py: Contains pure logic functions for calculating Weighted GPA and predicting future trends.

10. Screenshots and Result:

- Main menu:

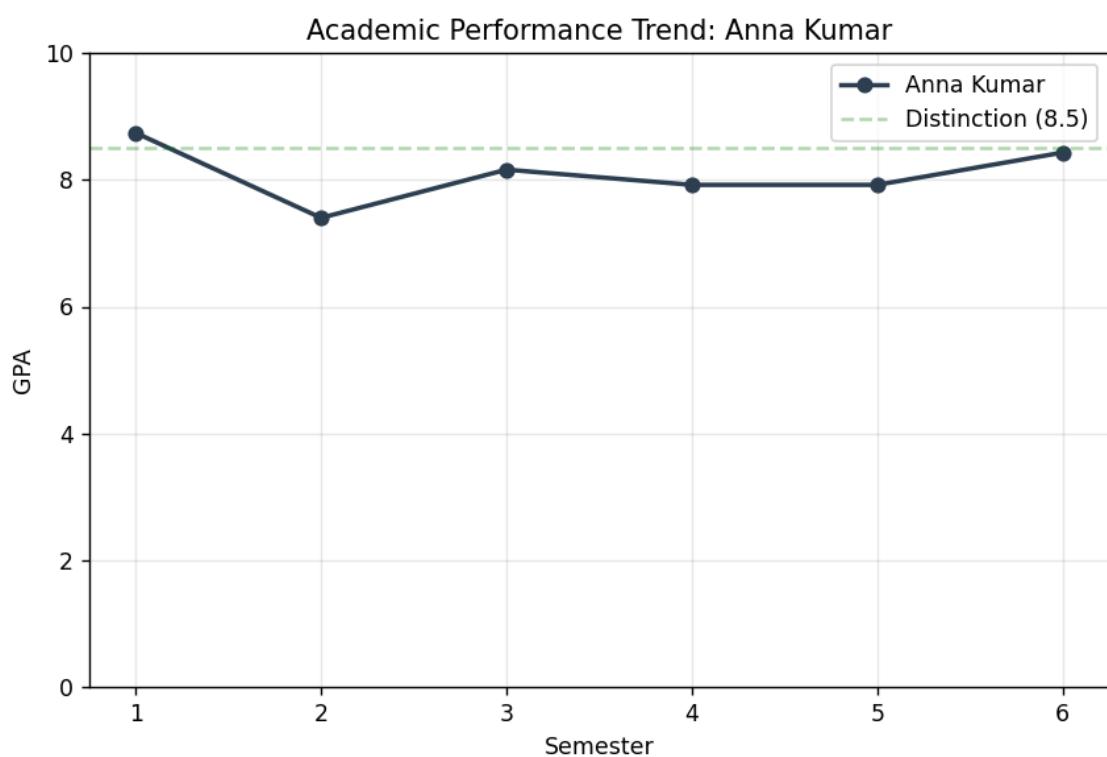
```
PS C:\Users\harsh\Desktop\ClassTrack_Project> python src/main.py

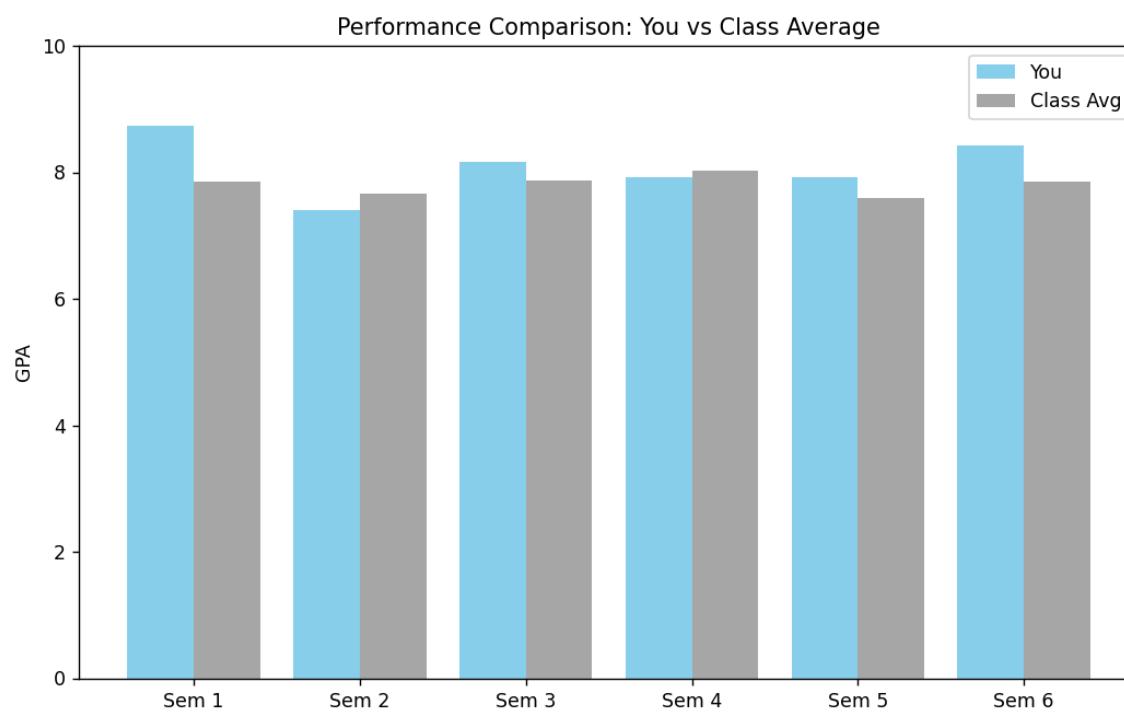
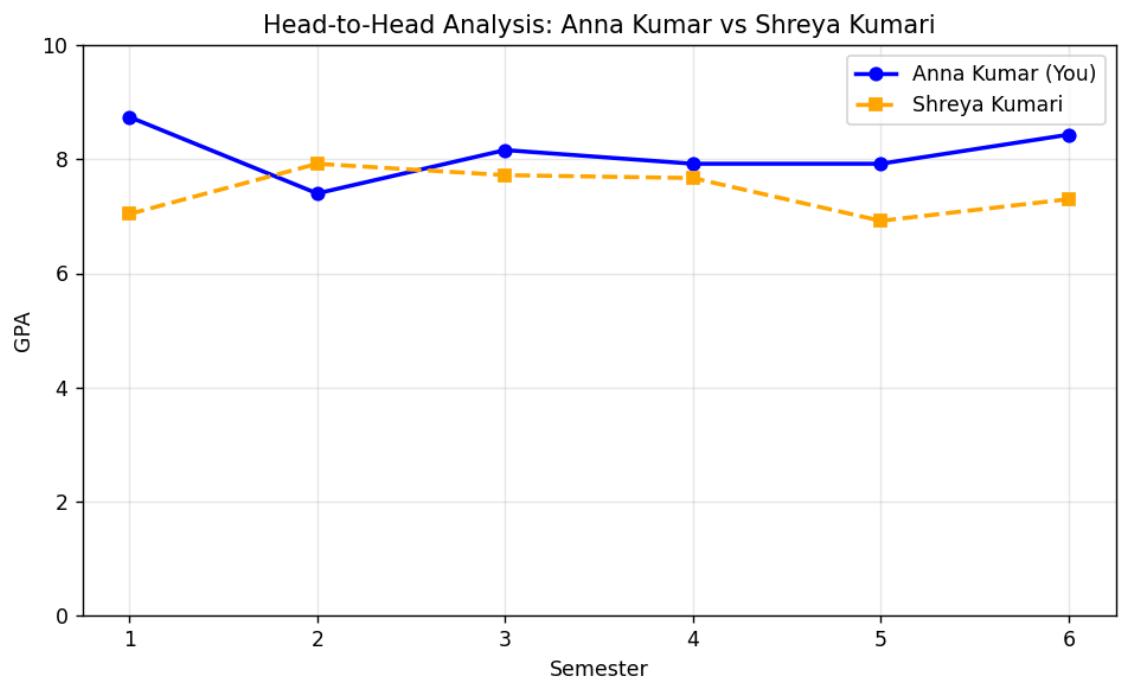
=====
CLASSTRACK: Academic Management System
=====
Enter Student ID to Login (e.g., 101): 101
Verifying ID...

Login Successful! Welcome, Anna Kumar.

-----
Dashboard: Anna Kumar
-----
1. View my Report Card
2. Visualize my Growth
3. Compare with Class Average
4. Compare with a Friend
5. Predict future GPA
6. Add New Marks
7. Exit
Select option (1-7): 1
```

- Graphs:





Dashboard: Anna Kumar

1. View my Report Card
2. Visualize my Growth
3. Compare with Class Average
4. Compare with a Friend
5. Predict future GPA
6. Add New Marks
7. Exit

Select option (1-7): 5

Based on your trend, predicted next GPA: 8.09

Dashboard: Anna Kumar

1. View my Report Card
2. Visualize my Growth
3. Compare with Class Average
4. Compare with a Friend
5. Predict future GPA
6. Add New Marks
7. Exit

Select Option (1-7): 6

--- Add New Subject Record ---

Semester: 2

Subject Name: Calculus

Credits (1-4): 4

Marks (0-100): 90

Saving...

Record Added Successfully.

Dashboard: Anna Kumar

1. View my Report Card
2. Visualize my Growth
3. Compare with Class Average
4. Compare with a Friend
5. Predict future GPA
6. Add New Marks
7. Exit

Select Option (1-7): 7

Logging out... Good luck!

11. Testing Approach

I used Manual Black Box Testing to validate the system:

- Valid Inputs: Entered correct marks (e.g., 85) and verified that the GPA calculation was correct.
- Invalid Inputs: Entered alphabets for "Marks" and verified that the system caught the ValueError without crashing.
- Boundary Testing: Entered 0 and 100 as marks to ensure the system handles edge cases.

12. Challenges faced

- Graphing Mismatch: When implementing the "Compare with Friend" feature, the code initially crashed because the two students had different numbers of semesters. I solved this by standardizing the X-axis list before plotting.
- Data Persistence: Initially, data added during the session was lost upon exit. I fixed this by implementing "append mode" in the file handling logic to save data instantly to the CSV.

13. Learnings & Key Takeaways

- Mastered the use of Pandas for efficient data manipulation.
- Learned how to visualize data programmatically using Matplotlib.
- Understood the importance of Modular Programming (separating logic, data, and UI) for building scalable projects.

14. Future Enhancements

- GUI: Upgrade the text-based menu to a graphical interface using Tkinter.
- PDF Export: Add a button to download the "Report Card" as a PDF file directly from the app.

15. References

- Python Documentation: docs.python.org
- Matplotlib User Guide: matplotlib.org/stable/users
- Pandas Library Documentation: pandas.pydata.org