1. Problem Statement

The goal of this SIWES project was to design and develop a **GPA Analyzer Web Application** that allows students to:

- Input their courses, units, and grades,
- Automatically calculate their GPA and CGPA,
- Receive insights such as performance trends and grade predictions,
- Visualize results using clear charts for better understanding.

The core problem the system solves is that many students struggle with:

- Manually calculating GPA/CGPA,
- Tracking academic performance across semesters,
- Understanding how future grades affect their CGPA,
- Identifying which courses impact their performance positively or negatively.

This project applies real-world software engineering skills—data handling, Python programming, visualization, and basic machine learning logic—to build a functional student performance analysis tool.

———

2. Dataset Used (Updated as Requested)

For this project, **ChatGPT provided me with the dataset** used for the analysis.
The dataset was named **sample_grades.csv** and it contained realistic, structured sample records that represent how university course and grade data is usually stored.

The dataset included columns such as:

- **Course** – The course title/code
- **Units** – The credit load for each course
- **Grade** – The student's grade
- **Grade Point** – Numeric grade value (e.g., A = 5.0, B = 4.0, etc.)

The dataset was used for:

- Testing the GPA calculation logic
- Verifying grade-to-point conversions
- Generating sample visualizations
- Demonstrating semester GPA and overall CGPA calculations

This dataset provided a clean and consistent foundation for building and validating the GPA Analyzer application.

---

3. Methods Used

The development process followed structured software engineering methods:

a. Data Preprocessing

- Loaded the dataset (sample_grades.csv) using pandas.
- Cleaned and normalized grade inputs.
- Converted letter grades to grade points using a dictionary (e.g., *{'A':5, 'B':4, ...}*).
- Calculated Total Quality Points (TQP = units × grade point).

b. GPA Calculation Logic

A custom function was developed:

Total Quality Points / Total Units = GPA

This function supports:

- Processing multiple semesters
- Handling different grading systems (4.0 and 5.0)
- Real-time GPA updates as courses change

c. Predictive Improvement Feature

A helper function was created to suggest how to improve your GPA by one level.

It compares your current GPA to the closest higher grade band using:

abs(target - current_gpa)
This method finds the nearest improvement level.

d. Software Frameworks

- **Python (Streamlit)** for building the interactive web interface
- **Pandas** for data manipulation
- **Matplotlib** for charts
- **Custom Python modules** (utils.py, data.py) for modular design

---

4. Model Results

After running the GPA Analyzer on the dataset:

✔ Semester GPA Calculation Worked Correctly

- Each course was processed accurately
- GPA matched expected values
- Edge cases such as empty inputs were handled

✔ CGPA Computation Verified

Multiple semester results were combined and computed correctly.

✔ Grade Prediction Function was Successful

The system could:

- Tell how many grade points a student needed to hit a target GPA
- Suggest improvement strategies

✔ GPA Trend Analysis Worked Smoothly

The application successfully generated plots showing:

- GPA progression
- Course difficulty impact
- Grade distribution

---

5. Visualizations

Several visualizations were generated with Matplotlib:

📌 1. Grade Distribution Chart

A bar chart showing how many A's, B's, C's etc. appear in the dataset.

📌 2. GPA Trend Line

A line chart comparing multiple semester GPA values showing performance improvement or decline.

📌 3. Units vs Grade Point Scatter Plot

Shows how courses with higher units impact overall GPA.

📌 4. CGPA Progress Chart

Tracks academic performance over the entire academic period.

These visualizations make it easy to interpret performance at a glance.