AUGUST 22, 2024

GPA CALCULATOR USING CSV

INTERN, PYTHON/GROUP 1

ABDUL MOIZ ARSALAN, MUZAMIL KHAN, SIBGHA MANZOOR

PROSENSIA SMC PVT LIMITED

Final Report: GPA Calculator Using CSV

Executive Summary

PROJECT OVERVIEW:

The project involves the development of a Python program designed to calculate Grade Point Average (GPA) from a set of grades and corresponding credit hours. The objective is to provide an accurate and efficient method for computing GPA based on a predefined grading scale. The final outcome is a functional script that reads grades and credit hours from an input source, computes the GPA, and returns the result.

KEY DELIVERABLES:

Grade to Point Conversion: The program includes a function to convert letter grades to numerical grade points based on a specified scale.

GPA Calculation: A function computes the GPA by aggregating grade points weighted by credit hours and then dividing by the total credit hours.

CSV Processing: Capability to handle and process data from CSV files containing grades and credit hours.

OUTCOME:

The project successfully developed a Python script that accurately calculates GPA. It meets its objectives by providing a straightforward, user-friendly solution to GPA calculation, demonstrating an effective application of programming skills in educational data processing.

3. Introduction

PURPOSE OF THE PROJECT:

The purpose of this project is to create a tool for calculating GPA, which is a common requirement in academic environments. The project is important because it automates GPA calculation, reducing the possibility of errors and saving time for students and educators.

PROJECT SCOPE:

- The project focuses on implementing a GPA calculator that:
- Converts letter grades to grade points.

- Computes the weighted average GPA based on points and credit hours.
- It does not cover additional features such as grade forecasting or integration with academic records systems.

OBJECTIVES:

Develop a reliable method for converting letter grades to grade points.

Implement a function to calculate GPA based on the grade points and credit hours. Enable the program to read and process input data from CSV files.

4. Requirements and Specifications

INPUT DATA:

- The program expects a CSV file containing two fields for each record:
- Grade: Letter grades such as 'A', 'B+', etc.
- Credit: Numerical credit hours associated with each grade.

OUTPUT DATA:

The output is a single GPA value calculated from the provided grades and credits. This value is typically displayed or returned by the program and can be stored or utilized as needed.

GRADING SCALE:

The grading scale used in the program is as follows:

• **A: 3.85 points**
• **B+: 3.65 points**
• **B: 3.45 points**
• **C+: 3.0 points**
• **C: 2.75 points**
• **D+: 2.0 points**
• **D: 1.8 points**
• **F: 0.0 points**

GPA CALCULATION METHOD:

The GPA is calculated using the formula:

\[\text{GPA} = \frac{\sum (\text{Grade Points} \times \text{Credits})}{\sum \text{Credits}} \]

The result is rounded to two decimal places for precision.

5. Project Planning and Timeline

Day 1: Requirements Gathering and Initial Setup

Tasks:

1. **Understand Requirements**:
    - Clarify the project's scope and deliverables.
    - Identify the structure of the input CSV file.
    - Define the grading scale and the formula for GPA calculation.
2. **Setup Development Environment**:
    - Ensure Python and required libraries (e.g., csv) are installed.
    - Set up a project directory structure.
3. **Plan Program Structure:**
    - Outline the key functions: get_grade_point, calculate_gpa, process_csv, and main.
    - Decide on the input format and output expectations.

Day 2: Core Functionality Development

Tasks:

1. **Develop Core Functions:**
    - Implement get_grade_point to convert letter grades to grade points.
    - Implement calculate_gpa to compute the GPA based on grades and credit hours.
2. **File Processing:**

- Implement process_csv to read the input CSV file, calculate GPAs, and append the results to each student's record.
- Handle any potential errors, such as missing or incorrect data.

Day 3: Testing and Validation

Tasks:

1. **Test Functionality:**
    - Test the program with various CSV files to ensure accuracy in GPA calculations.
    - Check edge cases, such as empty grades, incorrect input formats, and handling of file read/write errors.
2. **User Interaction**:
    - Implement the main function to interact with the user, allowing them to input the CSV file name and manage the flow of the program.
3. **Validation**:
    - Validate that the GPA calculations are correct and that the program is robust against various input scenarios.

Day 4: Finalization and Documentation

Tasks:

1. **Finalize the Program**:
    - Make any necessary adjustments based on testing feedback.
    - Ensure the program runs smoothly and outputs the correct data format.
2. **Documentation:**
    - Write clear documentation explaining how the program works, including instructions on how to use it.
    - Comment the code for readability and maintainability.
3. **Review and Delivery**:

  - Review the entire project to ensure all requirements are met.
  - Prepare the project for submission, including packaging the code and documentation.

TASK BREAKDOWN:

- **Planning:** Requirements gathering, designing the program structure.
- **Development:** Coding the grade point conversion, GPA calculation, and CSV processing functions.
- **Testing:** Developing and executing test cases, debugging.
- **Finalization:** Documenting the code, preparing final report.

RESOURCE ALLOCATION:

- **Software:** Python programming language, CSV library for handling data.
- **Tools:** PyCharm, VS Code, Excel for CSV

6. Development Process

ENVIRONMENT SETUP:

The development was done using Python with the `csv` library for file handling. The code is designed to run in a standard Python environment.

CORE FUNCTIONALITY:

- **get_grade_point:** Converts letter grades to numerical points based on a predefined scale.
- **calculate_gpa:** Computes GPA by weighting the grade points with credit hours and dividing by total credits.
- **process_csv:** Handles reading from the CSV file and passing data to the GPA calculation function.

USER INTERACTION:

The program interacts with users by processing input CSV files containing grade and credit data and outputting the computed GPA. Users need to provide the CSV file as input for the program to work.

7. Testing and Validation

TESTING APPROACH:

The program was tested using a combination of unit tests and manual tests to ensure that all functions work as intended and produce accurate results.

TEST CASES:

Valid Input: Test with a correctly formatted CSV file containing grades and credits.

Invalid Input: Test with missing or incorrectly formatted data to ensure robust error handling.

RESULTS:

Testing confirmed that the GPA calculation was accurate for valid inputs. Any issues related to incorrect input formats were handled effectively by the program.

ERROR HANDLING:

The program includes basic error handling to manage cases such as missing data or incorrect file formats, ensuring that users are notified of issues without crashing the program.

8. Challenges and Solutions

CHALLENGES ENCOUNTERED:

- Handling various CSV formats and ensuring data integrity.
- Implementing accurate error handling for unexpected inputs.

SOLUTIONS IMPLEMENTED:

- Used standard CSV handling techniques to ensure proper data parsing.
- Incorporated error handling mechanisms to manage invalid data and provide meaningful error messages.

9. Final Output

PROGRAM FUNCTIONALITY:

The final version of the program calculates GPA based on grades and credits from a CSV file and displays the result. It handles input errors gracefully and provides accurate calculations.

FILE STORAGE:

The program does not alter the original CSV file but processes it to generate the GPA. The output GPA value is displayed or returned to the user, and additional storage is not implemented.

10. Conclusion

PROJECT SUCCESS:

The project successfully achieved its goals of creating a functional GPA calculator. It meets the needs for an accurate, easy-to-use tool for GPA calculation.

LEARNING OUTCOMES:

The project provided insights into handling data files, implementing mathematical calculations, and ensuring robust error handling. It also enhanced understanding of practical applications of Python programming.

FUTURE RECOMMENDATIONS:

Potential improvements include adding features for handling larger datasets, integrating with academic systems for automated data import, or extending the functionality to include additional grading scales or GPA calculation methods.