Student Grade Calculator Project

1. Project Overview and Objectives

Project Overview

The Student Grade Calculator is Python-based application used to collect student's details, validates marks and assigns grade with encouraging messages. It demonstrates core programming concepts such as input handling, conditional statements and loops.

Objectives

- To understand how to take and validate user input in Python
- To apply control structures such as loops and conditional statements
- To design a modular program using functions
- To present output in a clear, user-friendly format

2. Setup and Installation Instructions

Prerequisites

- Python 3.x installed on the system
- Any code editor (VS Code)

Installation Steps

1. Download or clone the project repository from GitHub
2. Navigate to the project directory
3. Ensure Python is installed by running:

```
python --version
```

4. Run the application using:

```
python grade_calculator.py
```

No external libraries are required for this project.

3. Code Structure Explanation

```
Student-Grade-Calculator/
├──grade_calculator.py
├──README.md
├──test_cases.txt
└──screenshots/
    ├──output1.png
    └── output2.png … output7.png
```

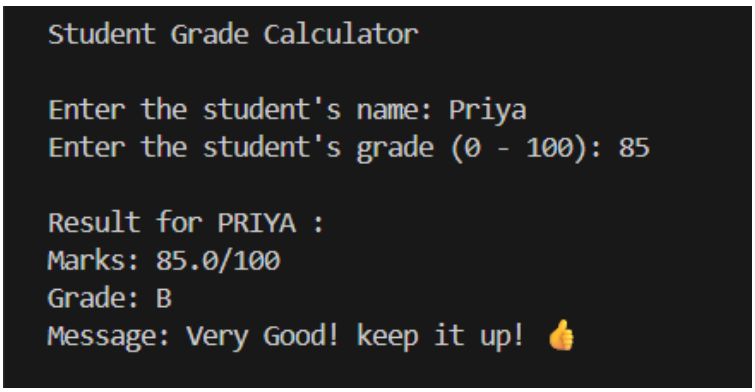
File Descriptions

- grade_calculator.py
  Main Python script responsible for:
  - Accepting user inputs
  - Validating data where required
  - Displaying formatted output
- README.md
  Contains:
  - Project description
  - Features
  - Instructions to run the program
- test_cases.txt
  Contains a list of sample inputs and expected outputs used to verify that the program works correctly for different scenarios, including valid marks and invalid input cases.
- screenshot.png
  A screenshot demonstrating the successful execution of the application and sample output.

4. Screenshots of Working Application

The screenshot.png file included in the project directory shows:

- User input prompts
- Entered values
- Final formatted output displayed in the terminal



This visual proof confirms that the application runs correctly and meets functional requirements.

5. Technical Requirements Fulfillment Explanation

Category	Technical Requirement	How It Was Implemented
Algorithm	Input–processing–output flow	Sequential logic was used to accept student name and marks, process grading logic, and display results.
Algorithm	Conditional grading logic	if-elif-else statements were used to assign grades based on marks ranges.
Algorithm	Input validation	Conditional checks ensure marks are within the valid range (0–100).
Algorithm	Repetition control	A while loop repeatedly prompts the user until valid marks are entered.
Data Structure	Variable storage	Variables store student name, marks, grade, and feedback message.

Category	Technical Requirement	How It Was Implemented
Architecture	Modular design	Functions such as valid_marks() and calculate_grade() were created to separate concerns and improve readability.
Architecture	Single-file execution	All program logic is contained in grade_calculator.py for simplicity and ease of execution.

6. Learning Outcomes

Through this project, the following learning outcomes were achieved:

- Gained hands-on experience with **core Python syntax and program execution** by building a complete, functional application.
- Learned to **validate user input effectively** using while loops, conditional statements, and exception handling to ensure marks were entered within the valid range (0–100).
- Understood the importance of **modular programming** by implementing functions for grade calculation and input validation, improving code readability and reusability.
- Enhanced **logical thinking and problem-solving skills** by designing and implementing grading rules using if-elif-else decision structures.

7. Conclusion

This program successfully demonstrates the use of Python control structures, functions, and input validation to calculate student grades accurately while providing meaningful feedback in a user-friendly manner.