Student Grade Calculator – Documentation Guideline

Project Overview:

Project Goals: The goal of this project is to build a simple, interactive Python program that:

- Collects a student's name and marks.
- Validates the marks to ensure they are between 0 and 100.
- Uses `if-elif-else` statements to assign a grade (A–F).
- Displays an encouraging message alongside the grade.
- Provides a user-friendly experience with clear prompts and formatted output.

Objectives:

- **Learn Conditional Logic:** Practice using `if-elif-else` statements to implement grading rules.
- **Input Validation:** Ensure user inputs are correct using a `while` loop.
- **Function Design:** Encapsulate grading logic inside a reusable function.
- **Friendly Output:** Display results with motivational messages to encourage students.
- **Documentation & Testing:** Provide clear README, test cases, and screenshots to demonstrate functionality.

Setup Instructions:

- **Install Python**
  - Download and install Python 3.8 or higher from [python.org](https://www.python.org/downloads/).
  - Verify installation:
    Bash python –version
- **Install VS Code**
  - Download and install [Visual Studio Code](https://code.visualstudio.com/).
  - Open VS Code after installation.
- **Create a Project Folder**
  Make a new folder for your project. For example: grade_calculator_project
- **Open Folder in VS Code**
  In VS Code, go to File → Open Folder and select grade_calculator_project.
- **Add Files**
  Inside the folder, create the following files:
  - grade_calculator.py→ main Python program
  - README.md → documentation
  - test_cases.txt` → sample inputs and expected outputs
  - screenshot.png → screenshot of program output

- **Run the program**

```
python grade_calculator.py
```

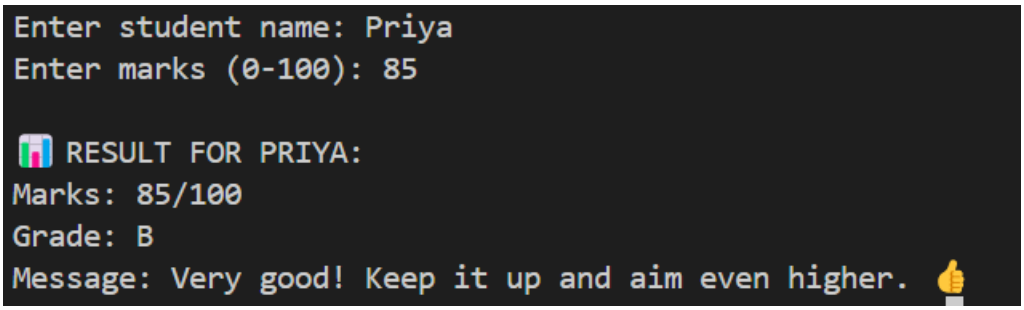
- **Test & Verify**
  Give the input in the terminal and check that the welcome message displays correctly.

Code Structure:

- **grade_calculator.py**
  - get_grade_and_message(marks) → returns (grade, message).
  - get_valid_marks() → validates input using a while loop.
  - main() → orchestrates input, grading, and output.
- **test_cases.txt** → sample inputs/outputs for validation.
- **screenshots** → images showing how the program runs.

Visual Documentation:

Include a screenshot showing:



Example with invalid input followed by valid:

```
Enter student name: Aarav
Enter marks (0-100): 110
Marks must be between 0 and 100. Try again.
Enter marks (0-100): eighty
Please enter a whole number between 0 and 100.
Enter marks (0-100): 72

📊 RESULT FOR AARAV:
Marks: 72/100
Grade: C
Message: Good effort—review a few topics and you'll level up. 💪
```

Technical Details:

- **Algorithm:** Validate marks → map to grade via thresholds → print grade with message.
- **Data Structures:** Primitive integers and strings.
- **Architecture:** Single-file script; functions for grading and validation; procedural main().

Testing Evidence:

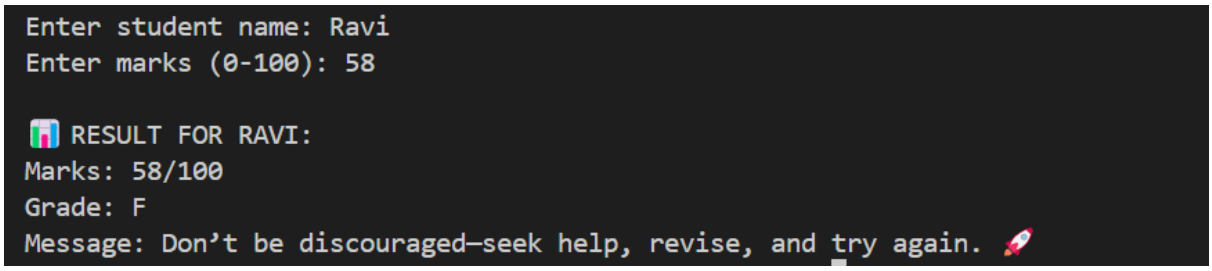
- **Test Case 1:**

```
Enter student name: Priya
Enter marks (0-100): 85

📊 RESULT FOR PRIYA:
Marks: 85/100
Grade: B
Message: Very good! Keep it up and aim even higher. 👍
```

- **Test Case 2:**

```
Enter student name: Neha
Enter marks (0-100): 63

📊 RESULT FOR NEHA:
Marks: 63/100
Grade: D
Message: You're close—focus on weak areas and practice more. 📚
```

- **Test Case 3:**



- **Test Case 4:**