
Student Grade Calculator – Documentation

---

1. Overview

Project Name: Student Grade Calculator
Language: Python 3.x
GUI Library: Tkinter
Purpose:
A desktop application to calculate student grades based on marks, provide encouraging messages, save results in a cumulative Excel-like CSV file, and export/print results as PDF.

Key Features:

- Dark mode GUI with professional layout
- Splash screen and app icon
- Input validation (marks 0–100)
- Grade calculation with encouraging messages
- Reset button
- Download result as PDF (text-based)
- Print PDF through default viewer
- Save and update student name and marks cumulatively in CSV
- Automatically capitalizes the first letter of student names

---

2. Requirements

- **Python 3.x** (tested on 3.13)
- **No external libraries** needed; uses built-in libraries:

tkinter, os, csv, tempfile, webbrowser

---

3. Project Files

```
student-grade-calculator/
|
├── grade_calculator_gui.py      # Main application script
├── icon.ico                     # Application icon
├── splash.png                   # Splash screen image
├── Student_Marks.csv            # Excel-like CSV file (auto-created)
├── screenshot.png               # Screenshot for README/documentation
├── README.md                    # GitHub project README
└── requirements.txt             # Project requirements info
```

---

4. GUI Layout

The GUI consists of:

1. **Splash Screen**
    - Shows at startup for 2.5 seconds
    - Displays an icon and/or splash image
2. **Main Window**
    - Dark-themed window
    - Title: *Student Grade Calculator*
3. **Input Fields**
    - Student Name (text entry)
    - Marks (numeric entry, 0–100)
4. **Buttons**
    - **Calculate:** Calculates grade and displays message
    - **Reset:** Clears all fields
    - **Download PDF:** Saves the displayed result as a text-based PDF
    - **Print PDF:** Opens the PDF in the default viewer for printing

- **Save to Excel:** Adds/updates student name and marks in cumulative CSV

5. **Result Label**
    - Displays student name, marks, grade, and encouraging message

---

5. Functionality

5.1 Grade Calculation Logic

Grade	Marks Range	Message
A	90–100	Outstanding performance. Keep it up.
B	80–89	Great job. You are close to excellence.
C	70–79	Good effort. Keep improving.
D	60–69	You passed. Try harder next time.
F	0–59	Do not give up. Learn and try again.

Code Implementation:

```
def calculate_grade(marks):
    if 90 <= marks <= 100:
        return "A", "Outstanding performance. Keep it up."
    elif 80 <= marks <= 89:
        return "B", "Great job. You are close to excellence."
    elif 70 <= marks <= 79:
        return "C", "Good effort. Keep improving."
    elif 60 <= marks <= 69:
        return "D", "You passed. Try harder next time."
    else:
        return "F", "Do not give up. Learn and try again."
```

5.2 Input Validation

- Student name cannot be empty
- Marks must be an integer between 0 and 100
- Automatically capitalizes first letter of the student name

Code Snippet:

```
name = name_entry.get().strip()
name = name.capitalize()
marks = int(marks_entry.get().strip())
if not (0 <= marks <= 100):
    raise ValueError
```

5.3 Reset Button

Clears the input fields and result label:

```
def reset_fields():
    name_entry.delete(0, tk.END)
    marks_entry.delete(0, tk.END)
    result_label.config(text="")
```

5.4 PDF Download & Print

- **Download PDF:** Saves result as text-based PDF (no external library needed)
- **Print PDF:** Opens the saved PDF in default viewer for printing

Download PDF Snippet:

```
def download_pdf():
    text = result_label.cget("text")
    file_path = filedialog.asksaveasfilename(defaultextension=".pdf")
    temp_txt = file_path.replace(".pdf", ".txt")
```

```
    with open(temp_txt, "w", encoding="utf-8") as f:
        f.write(text)
```

5.5 Save/Update Student Data in Excel-like CSV

- CSV file name: Student_Marks.csv
- Columns: Student Name | Marks
- Appends each new entry; cumulative storage

```
def save_to_excel():
    name = name_entry.get().strip().capitalize()
    marks = int(marks_entry.get().strip())
    existing_data = []
    if os.path.isfile(EXCEL_FILE):
        with open(EXCEL_FILE, mode='r', newline='', encoding='utf-8') as file:
            reader = csv.reader(file)
            existing_data = list(reader)
    if not existing_data:
        existing_data.append(["Student Name", "Marks"])
    existing_data.append([name, marks])
    with open(EXCEL_FILE, mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerows(existing_data)
```

---

6. How to Run

1. Ensure **Python 3.x** is installed.
2. Place all files in the same folder: grade_calculator_gui.py, icon.ico, splash.png.
3. Run the application:

python grade_calculator_gui.py

4. Enter student name and marks, then click:

- **Calculate** → displays grade & message
- **Reset** → clears fields
- **Download PDF** → save result
- **Print PDF** → print result
- **Save to Excel** → store/update data

5. CSV file Student_Marks.csv is updated cumulatively.

---

7. GitHub Folder Structure

```
student-grade-calculator/
|
 ├── grade_calculator_gui.py
 ├── icon.ico
 ├── splash.png
 ├── Student_Marks.csv  # auto-created
 ├── screenshot.png
 ├── README.md
 └── requirements.txt
```

---

8. Notes

- Compatible with **Windows**
- No external libraries required
- PDF export is text-based for maximum compatibility
- Excel-like CSV can be opened in Excel or LibreOffice
- Supports multiple student entries, cumulative storage