PROJECT REPORT

Student Grade Management System

Pranshu Gupta
Dr. Sandeep Monga
22 November 2025
Registration Number-25BAI11057

1. Introduction

This code implements a **console based student management system**This system is designed to perform **CRUD (Create, Read, Update, Delete)** operations on student records, including tracking their subject-wise marks, calculating total marks, percentages, and assigning a final letter grade.

Key Features and Components

The system primarily utilizes **JSON** to persist (save) the student data to a file named `students.json`, allowing the data to remain available even after the program closes.**Data Persistence:** The `load_data()` and `save_data()` functions handle reading student records from, and writing them back to, the `students.json` file.

- **Grade Calculation:** The `calculate_grade()` function defines the grading logic:
  - 90% and above→'A'
  - 80–89%→'B'
  - 70–79%→'C'
  - 60–69%→'D'
  - Below 60%→'F'
- **Student Operations:**
  - `add_student()`: Gathers student details (ID, name, class) and subject marks, calculates initial metrics, and saves the new record. It includes validation to prevent duplicate IDs and ensure marks are within the 0-100 range.
  - `view_all_students()`: Displays a formatted list of all registered students, showing their ID, name, class, percentage, and grade.

    - `search_student()`: Finds and displays the complete JSON record for a student using their ID.
    - `update_marks()`: Allows an admin to select a student by ID, choose a subject, and update the mark, automatically recalculating the total, percentage, and grade.
    - `delete_student()`: Removes a student record based on their ID.
- **User Interface:** The `menu()` function provides a simple text-based interface for users to select the desired action.
- **Formatting Utility:** The script attempts to use the **`tabulate`** library for professional table formatting but includes a **fallback function** if the library is not installed, ensuring the system remains functional.

Overall, the code is a **modular and complete solution** for basic student record management, demonstrating practical use of Python's I/O and data structure handling.

2. Problem Statement

Schools and universities require efficient tools to record marks and generate grades automatically. Manual calculations are time consuming and error prone. This project solves these issues by automating the grading workflow.

3. Functional Requirements

- Add student details and marks
- Store marks (0–100) for multiple subjects • Calculate total, percentage, and grade
- View all student records
- Search and update existing records
- Delete student records

4. Non-Functional Requirements

- Usability: Simple menu-driven CLI
- Reliability: Input validation prevents incorrect entries • Maintainability: Code structured using functions
- Data Persistence: JSON file storage

5. System Architecture

The system follows a modular function▪based architecture using: • Python as main programming language

- JSON file as lightweight database
- Menu-driven user interface

6. Design Diagram

High-level architecture:

User → Menu → Functions → JSON Storage → Output

7. Use Case Diagram

Actors: User

Use Cases: Add Student, Update Marks, Delete Student, View Students, Search Student

8. Sequence Diagram

1. User selects option

2. System loads JSON data

3. Performs requested operation 4. Updates JSON

5. Displays result

9. Workflow Diagram

Start → Show Menu → Take Input → Execute Function → Save/ Display Output → Loop/Exit

10. Class Diagram

The system is function-oriented; implicit data model: Student

- id
- name
- class
- subjects{}
- total
- percentage • grade

11. ER Diagram

Entity: Student

Attributes: ID, Name, Class, Subjects, Total, Percentage, Grade

12. Design Decisions & Rationale

- Backend: Python chosen for simplicity and readability
- Algorithm: Rule-based grading, no ML required
- Database: JSON used for lightweight storage
- Architecture: Modular procedural design

13. Implementation Details

The system uses input-based data collection, dictionary storage of subjects, and JSON serialization. The calculate_grade() function assigns grades based on percentage.

14. Testing Approach

- Input validation tests
- Boundary value testing (0 and 100 marks) • Function testing for add/update/delete

15. Challenges Faced

- Handling invalid numeric inputs
- Maintaining clean JSON formatting • Ensuring robust menu flow

16. Learning & Key Takeaways

- Improved understanding of file I/O
- Practical experience with structured program design • Enhanced skills in error handling and validation

17. Future Enhancements

- GUI-based interface
- Cloud database support
- Report card download feature • Web dashboard for analytics

18. References

- Python Documentation
- ReportLab Library Docs
- JSON Standards

19. My code

```
import json
import os

try:
    from tabulate import tabulate
    tabulate_available = True
except ImportError:
    tabulate_available = False


    def tabulate(data, headers, tablefmt):

        header_line = " | ".join(headers)
        separator = "-" * len(header_line)
        rows = "\n".join([" | ".join(map(str, row)) for row in data])
        return f"\n{header_line}\n{separator}\n{rows}"

    print(" WARNING: 'tabulate' library not found. Install it using 'pip
install tabulate' for better formatting.")

data_file = 'students.json'

def load_data():

    if not os.path.exists(data_file):
        return []
    try:
        with open(data_file, 'r') as f:
            return json.load(f)
    except json.JSONDecodeError:

        print(f"Warning: Data file '{data_file}' is empty or invalid.
Starting with no records.")
        return []

def save_data(data):

    try:
        with open(data_file, 'w') as f:
            json.dump(data, f, indent=4)
    except IOError as e:
        print(f"Error: Could not save data to file {data_file}. Details:
{e}")

def calculate_grade(percentage):

    if percentage >= 90:
        return 'A'
    elif percentage >= 80:
        return 'B'
    elif percentage >= 70:
        return 'C'
    elif percentage >= 60:
        return 'D'
    else:
        return 'F'

def add_student():

    data = load_data()
```

```
    student_id = input("Enter Student ID: ").strip()

    for s in data:
        if s['id'] == student_id:
            print("Error: Student ID already exists!")
            return

    name = input("Enter Student Name: ").strip()
    class_name = input("Enter Class/Section: ").strip()

    subjects = {}
    try:
        n_subjects_input = input("How many subjects? ")
        if not n_subjects_input.isdigit():
            print("Invalid input for number of subjects. Operation
cancelled.")
            return
        n_subjects = int(n_subjects_input)
    except ValueError:

        print("Invalid number of subjects. Operation cancelled.")
        return

    for i in range(n_subjects):
        while True:
            sub = input(f"Subject {i+1} name: ").strip()
            if not sub:
                print("Subject name cannot be empty.")
                continue

            try:
                marks_input = input(f"Marks for {sub} (0-100): ")
                marks = int(marks_input)
                if 0 <= marks <= 100:
                    subjects[sub] = marks
                    break
                else:
                    print("Marks must be between 0 and 100.")
            except ValueError:
                print("Invalid input. Please enter a number for marks.")

    if not subjects:
        print("No subjects added. Record saved with no marks.")
        total = 0
        percentage = 0.0
    else:
        total = sum(subjects.values())
        percentage = total / len(subjects)

    grade = calculate_grade(percentage)

    student = {
        'id': student_id,
        'name': name,
        'class': class_name,
        'subjects': subjects,
        'total': total,
        'percentage': percentage,
        'grade': grade
    }
```

```
        data.append(student)
        save_data(data)
        print("Student added successfully!\n")

    def view_all_students():

        data = load_data()
        if not data:
            print("No student records found!")
            return

        table = []
        for s in data:

            percent = s.get('percentage', 0.0)
            grade = s.get('grade', 'N/A')
            table.append([
                s['id'], s['name'], s['class'], f"{percent:.2f}%", grade
            ])

        print("\n--- All Students ---")

        table_format = "grid" if tabulate_available else "simple"
        print(tabulate(table, headers=["ID", "Name", "Class", "Percent",
    "Grade"], tablefmt=table_format))
        print()

    def search_student():

        data = load_data()
        sid = input("Enter Student ID to search: ").strip()

        for s in data:
            if s['id'] == sid:
                print("\n--- Student Record ---")
                print(json.dumps(s, indent=4))
                print()
                return

        print("Student not found!\n")

    def update_marks():

        data = load_data()
        sid = input("Enter Student ID to update: ").strip()
        student_found = False

        for s in data:
            if s['id'] == sid:
                student_found = True
                print(f"Updating marks for: {s['name']}")

                if not s.get('subjects'):
                    print("No subjects found for this student. Cannot update
    marks.")
                    return

                print("Current subjects:")
                for sub, mark in s['subjects'].items():
                    print(f"  {sub}: {mark}")
```

```
                subject = input("Enter subject name to update: ").strip()
                if subject not in s['subjects']:
                    print("Subject not found for this student!\n")
                    return

                while True:
                    try:
                        new_mark_input = input("Enter new marks (0-100): ")
                        new_mark = int(new_mark_input)
                        if 0 <= new_mark <= 100:
                            break
                        else:
                            print("Marks must be between 0 and 100.")
                    except ValueError:
                        print("Invalid input. Please enter a number for
    marks.")

                s['subjects'][subject] = new_mark


                s['total'] = sum(s['subjects'].values())

                if s['subjects']:
                    s['percentage'] = s['total'] / len(s['subjects'])
                    p = s['percentage']
                    s['grade'] = calculate_grade(p)
                else:
                     s['percentage'] = 0.0
                     s['grade'] = 'F'
                save_data(data)
                print("Marks updated successfully!\n")
                return

        if not student_found:
            print("Student not found!\n")

    def delete_student():

        data = load_data()
        sid = input("Enter Student ID to delete: ").strip()

        new_data = [s for s in data if s['id'] != sid]

        if len(new_data) == len(data):
            print("Student not found!\n")
        else:
            save_data(new_data)
            print("Record deleted successfully!\n")

    def menu():

        while True:
            print("\n===== Student Grade Management System =====")
            print("1. Add Student")
            print("2. View All Students")
            print("3. Search Student")
            print("4. Update Marks")
            print("5. Delete Student")
            print("6. Exit")

            try:
```

```
                choice = input("Enter your choice: ").strip()
            except EOFError:
                print("\n Input stream closed (EOF). Exiting non-
    interactively.")
                break

            if choice == '1':
                add_student()
            elif choice == '2':
                view_all_students()
            elif choice == '3':
                search_student()
            elif choice == '4':
                update_marks()
            elif choice == '5':
                delete_student()
            elif choice == '6':
                print("Exiting... Goodbye!")
                break
            else:
                print("Invalid choice! Try again.\n")

    if __name__ == "__main__":
        menu()
```

20. Output

```
/usr/local/bin/python3 "/Users/pranshugupta/Documents/cpp codes/Python codes/student_grade_system.py"
pranshugupta@Pranshus-MacBook-Air cpp codes % /usr/local/bin/python3 "/Users/pranshugupta/Documents/cpp codes/Python codes/student_grade_s
ystem.py"

===== Student Grade Management System =====
1. Add Student
2. View All Students
3. Search Student
4. Update Marks
5. Delete Student
6. Exit
Enter your choice: 1
Enter Student ID: 074
Enter Student Name: Pranshu
Enter Class/Section: 12A
How many subjects? 2
Subject 1 name: Maths
Marks for Maths (0-100): 90
Subject 2 name: Physics
Marks for Physics (0-100): 91
Student added successfully!


===== Student Grade Management System =====
1. Add Student
2. View All Students
3. Search Student
4. Update Marks
5. Delete Student
6. Exit
Enter your choice: 2

--- All Students ---
+------+---------+---------+-----------+---------+
|   ID | Name    | Class   | Percent   | Grade   |
+======+=========+=========+===========+=========+
|  023 | Pranhsu | 3A      | 97.50%    | A       |
+------+---------+---------+-----------+---------+
|  037 | pranshu | 3A      | 92.00%    | A       |
+------+---------+---------+-----------+---------+
|  087 | Shivam  | 7A      | 88.00%    | B       |
+------+---------+---------+-----------+---------+
|  074 | Pranshu | 12A     | 90.50%    | A       |
+------+---------+---------+-----------+---------+
```

```
===== Student Grade Management System =====
1. Add Student
2. View All Students
3. Search Student
4. Update Marks
5. Delete Student
6. Exit
Enter your choice: 3
Enter Student ID to search: 023

--- Student Record ---
{
    "id": "023",
    "name": "Pranhsu",
    "class": "3A",
    "subjects": {
        "MATHS": 98,
        "PHYSICS": 97
    },
    "total": 195,
    "percentage": 97.5,
    "grade": "A"
}


===== Student Grade Management System =====
1. Add Student
2. View All Students
3. Search Student
4. Update Marks
5. Delete Student
6. Exit
Enter your choice: 4
Enter Student ID to update: 023
Updating marks for: Pranhsu
Current subjects:
  MATHS: 98
  PHYSICS: 97
Enter subject name to update: Maths
Subject not found for this student!
```

```
===== Student Grade Management System =====
1. Add Student
2. View All Students
3. Search Student
4. Update Marks
5. Delete Student
6. Exit
Enter your choice: 5
Enter Student ID to delete: 023
Record deleted successfully!


===== Student Grade Management System =====
1. Add Student
2. View All Students
3. Search Student
4. Update Marks
5. Delete Student
6. Exit
Enter your choice: 6
Exiting... Goodbye!
pranshugupta@Pranshus-MacBook-Air cpp codes %
```