CS-500 Instructor Gradebook Project

Program Execution Instructions

1. Installation Requirements:
    - Confirm that Python is installed on your system.
2. Running the Program:
    - Navigate to the directory containing **gradebookApp.py**.
    - Execute the command **python gradebookApp.py** in the terminal to launch the program.
    - Utilize the on-screen prompts to engage with the application's features.
3. Data Files:
    - Verify that all necessary data files (such as JSON files containing grading policies and Grades for students) are present in the same folder.
    - Upon program initiation, it will automatically import this data.

In summary:

- The starting point of the program is the **gradebookApp.py** file. Running this script is the primary method for utilizing and evaluating the program's functionalities.
- On program start-up, it reads various JSON files from policy.dat and Grades.dat files, which contain essential information like student details, grading policies, and grade data, to set up the environment for gradebook management.
- When saving any data, it's crucial to include the filename with the appropriate extension to ensure the integrity of the data.
- The program tracks the grades of students as they are entered and updated. It checks against the grading policy to ensure that grades are within permissible bounds before accepting them.
- Once grading for a course is completed, a final grade can be calculated and stored in the database and in the student record.
- The grade output contains students' details, displaying all attributes and both scores and equivalent letter grades.
- The generated reports serve various stakeholders, with one tailored for student distribution, showcasing individual performance, and another designed for administrative purposes, summarizing class performance.

Test Cases and Results Documentation

Test Case Structure:

Each test case is meticulously documented to include the following sections:

Description: This segment provides a succinct summary of the test case's objective, outlining what aspect of the gradebook program it is designed to evaluate.

Expected Outcome: This details the anticipated result or behavior of the application if it operates according to its specifications upon the test case execution.

Actual Outcome: This records the genuine response or output observed when the test case is carried out, facilitating comparison against the expected outcome.

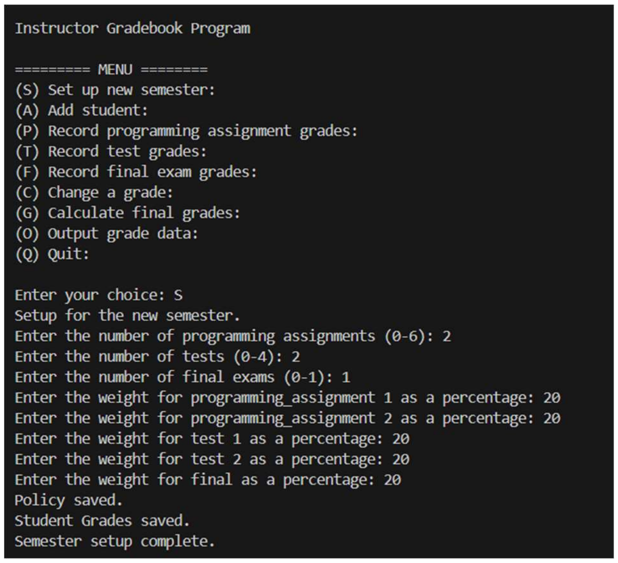
Screenshots: Visual evidence displaying the test case execution, which will include images of the application's response or results as shown on the screen.

Documenting Test Cases:

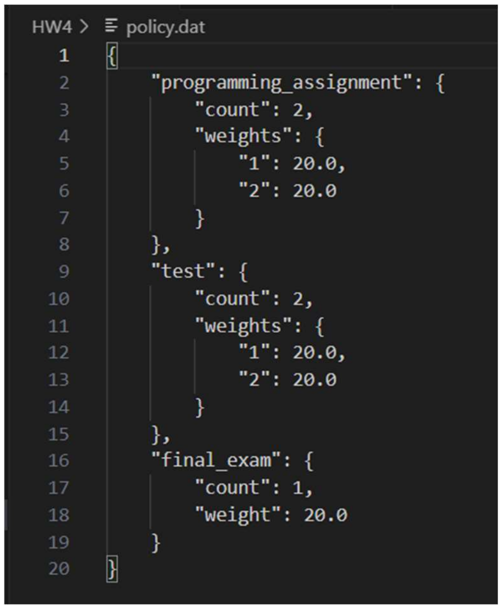
To ensure comprehensive coverage, a suite of test cases has been created for the core functionalities of the application. These test cases can be replicated as delineated below, and results should be chronicled accordingly.

Test Case 1:

- **Description**: Setting up the grading policy for a new semester.
- **Preconditions**: The set up new semester should be executed at the start of every semester, and when executed existing grading policy data and grades will be ignored.
- **Expected Outcome**: The system allows entry of the number and weights of assignments, tests, and the final exam, and confirms policy setup for the semester.
- **Actual Outcome**: A new policy is set up and saved in the gradebook and "policy.dat" JSON file. The existing (if students grades were saved previously) students in Grades.dat file are ignored and deleted.
- **Screenshots**:

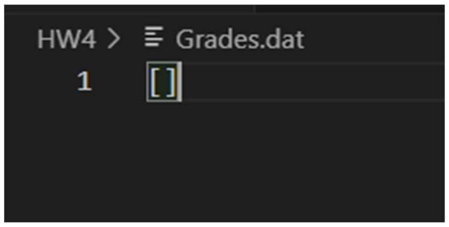


```
Instructor Gradebook Program

========= MENU ========
(S) Set up new semester:
(A) Add student:
(P) Record programming assignment grades:
(T) Record test grades:
(F) Record final exam grades:
(C) Change a grade:
(G) Calculate final grades:
(O) Output grade data:
(Q) Quit:

Enter your choice: S
Setup for the new semester.
Enter the number of programming assignments (0-6): 2
Enter the number of tests (0-4): 2
Enter the number of final exams (0-1): 1
Enter the weight for programming_assignment 1 as a percentage: 20
Enter the weight for programming_assignment 2 as a percentage: 20
Enter the weight for test 1 as a percentage: 20
Enter the weight for test 2 as a percentage: 20
Enter the weight for final as a percentage: 20
Policy saved.
Student Grades saved.
Semester setup complete.
```



HW4 > policy.dat

```
{
    "programming_assignment": {
        "count": 2,
        "weights": {
            "1": 20.0,
            "2": 20.0
        }
    },
    "test": {
        "count": 2,
        "weights": {
            "1": 20.0,
            "2": 20.0
        }
    },
    "final_exam": {
        "count": 1,
        "weight": 20.0
    }
}
```



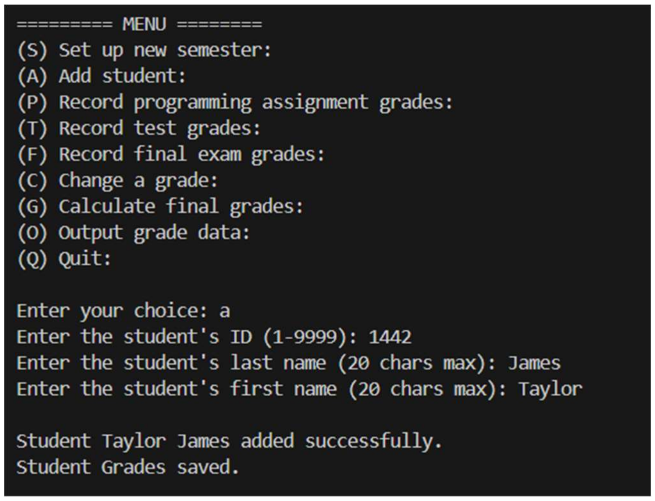
HW4 > Grades.dat

```
[]
```

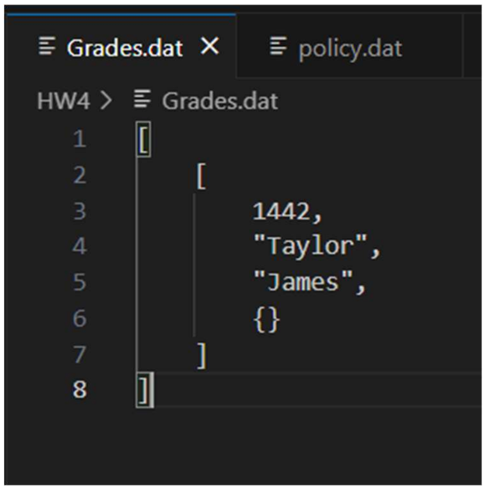
Test Case 2:

- **Description**: Adding a new student to the gradebook.
- **Preconditions**: To add a new student to the gradebook, a policy must be present first when semester is setup, otherwise program will not allow adding a student.
- **Expected Outcome**: If The student is successfully added, and the student is added to the gradebook list of students and saved in "Grades.dat" JSON file..

- **Actual Outcome**: Student successfully added.
- **Screenshots**:



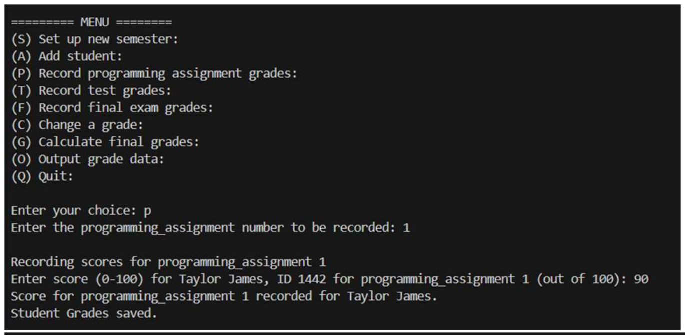
```
========= MENU ========
(S) Set up new semester:
(A) Add student:
(P) Record programming assignment grades:
(T) Record test grades:
(F) Record final exam grades:
(C) Change a grade:
(G) Calculate final grades:
(O) Output grade data:
(Q) Quit:

Enter your choice: a
Enter the student's ID (1-9999): 1442
Enter the student's last name (20 chars max): James
Enter the student's first name (20 chars max): Taylor

Student Taylor James added successfully.
Student Grades saved.
```



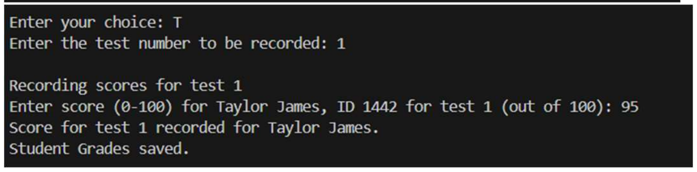
```
[
    [
        1442,
        "Taylor",
        "James",
        {}
    ]
]
```

Test Case 3:

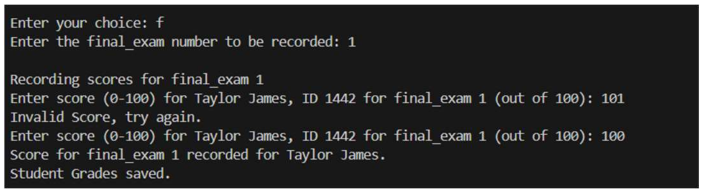
- **Description**: Recording grades for an assignment.
- **Precondition**: The gradebook system is initialized, and at least one student is already added to the system with the semester set up.
- **Expected Outcome**: Input grades are accepted within the specified range and correctly reflected in the student's record according to the relative weights.
- **Actual Outcome**: Grades added successfully for all students.
- **Screenshots**:



```
========= MENU ========
(S) Set up new semester:
(A) Add student:
(P) Record programming assignment grades:
(T) Record test grades:
(F) Record final exam grades:
(C) Change a grade:
(G) Calculate final grades:
(O) Output grade data:
(Q) Quit:

Enter your choice: p
Enter the programming_assignment number to be recorded: 1

Recording scores for programming_assignment 1
Enter score (0-100) for Taylor James, ID 1442 for programming_assignment 1 (out of 100): 90
Score for programming_assignment 1 recorded for Taylor James.
Student Grades saved.
```



```
Enter your choice: T
Enter the test number to be recorded: 1

Recording scores for test 1
Enter score (0-100) for Taylor James, ID 1442 for test 1 (out of 100): 95
Score for test 1 recorded for Taylor James.
Student Grades saved.
```

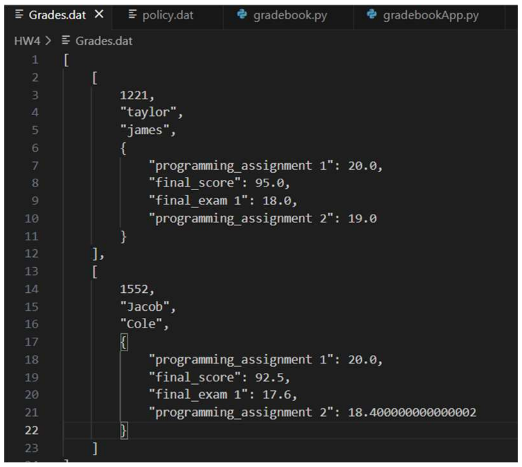


```
Enter your choice: f
Enter the final_exam number to be recorded: 1

Recording scores for final_exam 1
Enter score (0-100) for Taylor James, ID 1442 for final_exam 1 (out of 100): 101
Invalid Score, try again.
Enter score (0-100) for Taylor James, ID 1442 for final_exam 1 (out of 100): 100
Score for final_exam 1 recorded for Taylor James.
Student Grades saved.
```

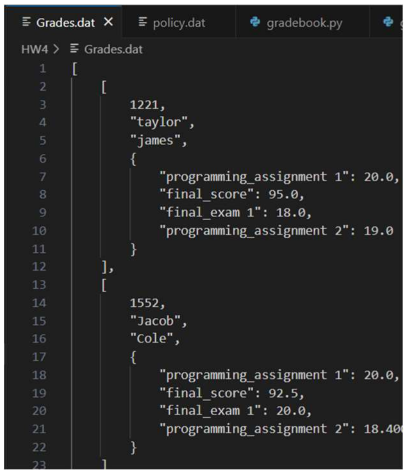
Test Case 4:

- **Description**: Changing a student's grade and verifying update.
- **Precondition**: A student's grade record exists, and a valid grade and type of score to be recorded is given.
- **Expected Outcome**: The specified grade is changed, and the update is reflected in the student's record and the database.
- **Actual Outcome**: Student grade modified successfully.
- **Screenshots**:

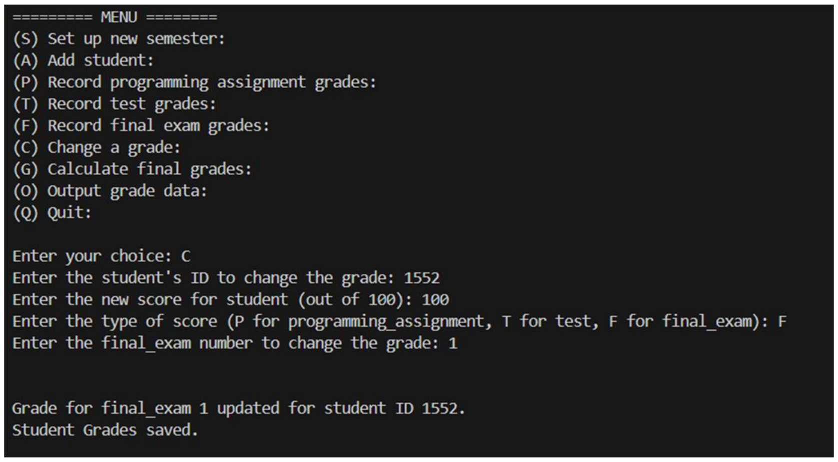
Before: **After**:



```
[
    [
        1221,
        "taylor",
        "james",
        {
            "programming_assignment 1": 20.0,
            "final_score": 95.0,
            "final_exam 1": 18.0,
            "programming_assignment 2": 19.0
        }
    ],
    [
        1552,
        "Jacob",
        "Cole",
        {
            "programming_assignment 1": 20.0,
            "final_score": 92.5,
            "final_exam 1": 17.6,
            "programming_assignment 2": 18.400000000000002
        }
    ]
```



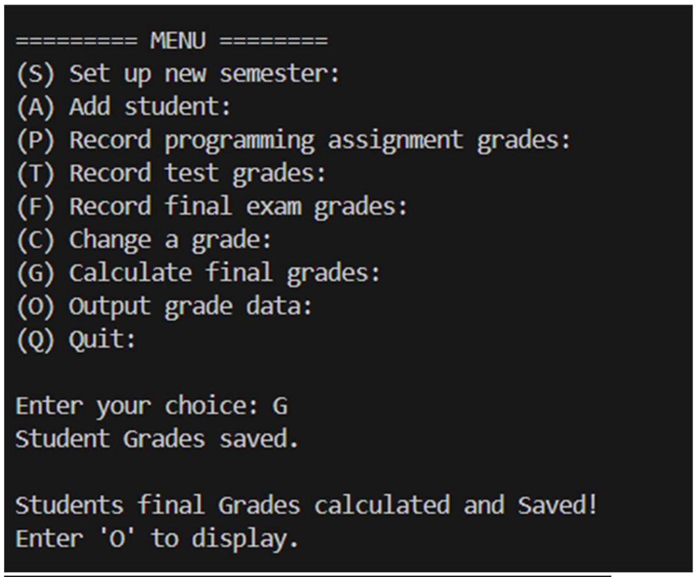
```
[
    [
        1221,
        "taylor",
        "james",
        {
            "programming_assignment 1": 20.0,
            "final_score": 95.0,
            "final_exam 1": 18.0,
            "programming_assignment 2": 19.0
        }
    ],
    [
        1552,
        "Jacob",
        "Cole",
        {
            "programming_assignment 1": 20.0,
            "final_score": 92.5,
            "final_exam 1": 20.0,
            "programming_assignment 2": 18.40
        }
    ]
```



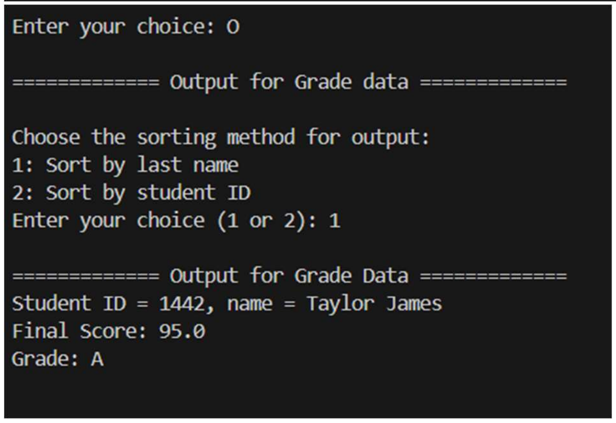
```
========= MENU ========
(S) Set up new semester:
(A) Add student:
(P) Record programming assignment grades:
(T) Record test grades:
(F) Record final exam grades:
(C) Change a grade:
(G) Calculate final grades:
(O) Output grade data:
(Q) Quit:

Enter your choice: C
Enter the student's ID to change the grade: 1552
Enter the new score for student (out of 100): 100
Enter the type of score (P for programming_assignment, T for test, F for final_exam): F
Enter the final_exam number to change the grade: 1


Grade for final_exam 1 updated for student ID 1552.
Student Grades saved.
```

Test Case 5:

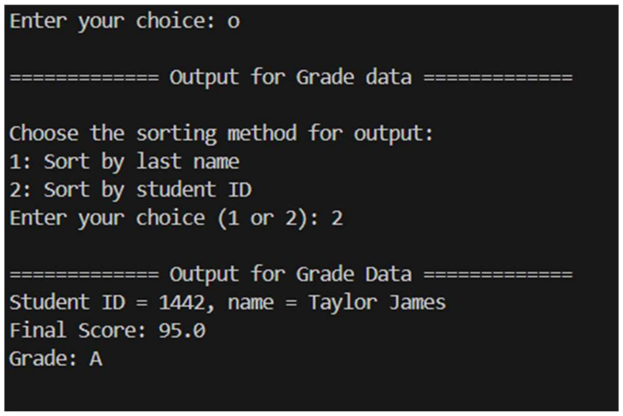
- **Description**: Calculating and Output of final grades after all scores are entered.
- **Precondition**: All or any assignment and test scores for the students have been entered into the system.
- **Expected Outcome**: The program calculates final scores based on the predefined weights and outputs them correctly sorted by either student ID or last name.
- **Actual Outcome**: Final grade calculated and added to record successfully.
- **Screenshots**: [Attach screenshot here]



```
========= MENU ========
(S) Set up new semester:
(A) Add student:
(P) Record programming assignment grades:
(T) Record test grades:
(F) Record final exam grades:
(C) Change a grade:
(G) Calculate final grades:
(O) Output grade data:
(Q) Quit:

Enter your choice: G
Student Grades saved.

Students final Grades calculated and Saved!
Enter 'O' to display.
```



```
Enter your choice: O

============ Output for Grade data ============

Choose the sorting method for output:
1: Sort by last name
2: Sort by student ID
Enter your choice (1 or 2): 1

============ Output for Grade Data ============
Student ID = 1442, name = Taylor James
Final Score: 95.0
Grade: A
```



```
Enter your choice: o

============ Output for Grade data ============

Choose the sorting method for output:
1: Sort by last name
2: Sort by student ID
Enter your choice (1 or 2): 2

============ Output for Grade Data ============
Student ID = 1442, name = Taylor James
Final Score: 95.0
Grade: A
```