Student Grade Management System

Objective:

The ***Student Grade Management System*** is a desktop application designed to allow students and administrators to manage and track student academic records efficiently. Its core functionalities include managing grades, course enrollment, semester tracking, and providing access based on user roles (admin or student). This program automates the calculation of student grades, allows students to view their progress, and enables administrators to maintain accurate records, ensuring transparency and accountability.

The **goal** is to streamline academic record management, reduce administrative overhead, and give students an easy way to track their academic performance.

Features

1. Login System

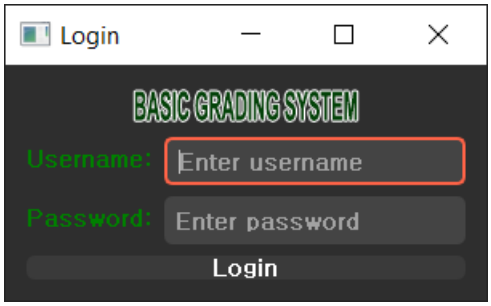
Explanation:

The login system ensures **role-based access control (RBAC)**, which guarantees that students can only view their own grades while administrators have access to all student data. The system uses simple authentication by checking the user's **username** and **password** against records stored in a **JSON file**.

- **Admin Users**: Have the ability to create, read, update, and delete student records. Admins can also export data to CSV and manage the system.
- **Student Users**: Can only view their own academic records and progress. They cannot modify any records.

Security Considerations:

- Currently, passwords are stored in **plain text** within the JSON file. This is a limitation that should be addressed in the future with **password hashing** (e.g., bcrypt or SHA256) to enhance security.
- **JSON storage** is lightweight but not scalable for larger systems; ideally, a database (e.g., SQLite, MySQL) should be used for better security and scalability.



---

2. Admin Features

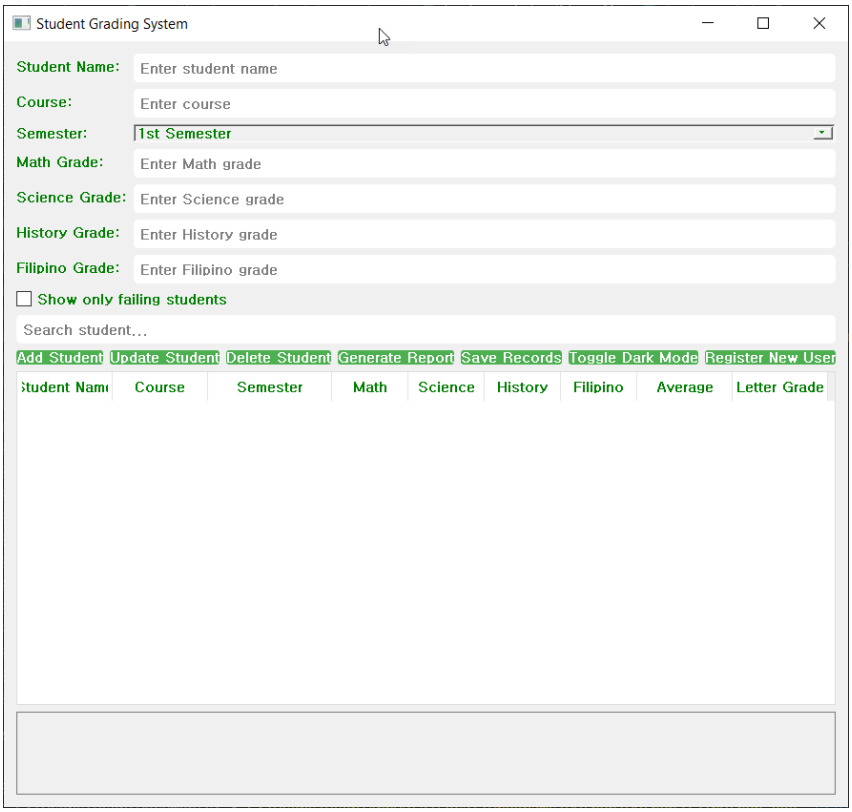
Admin users have control over the entire system and can perform the following actions:

- **Add/Edit/Delete Student Records**: Admins can manage students by adding or updating records, including student name, course, semester, and grades.

- **View All Student Records**: Admins can see a list of all student records with their grades and performance data. The records can be sorted by name, course, or semester.
- **Export Data to CSV**: Admins can export student records to a CSV file for reporting or external analysis.
- **Grade Calculation and Letter Assignment**: The system automatically calculates the average grade per student and assigns a letter grade (A, B, C, etc.) according to predefined thresholds.

Explanation:

This feature ensures that admins can easily manage student data while preventing unauthorized access to sensitive information.



---

3. Student Features

The **student interface** is designed to be simple and easy to navigate, focusing on grade transparency and personal data access:

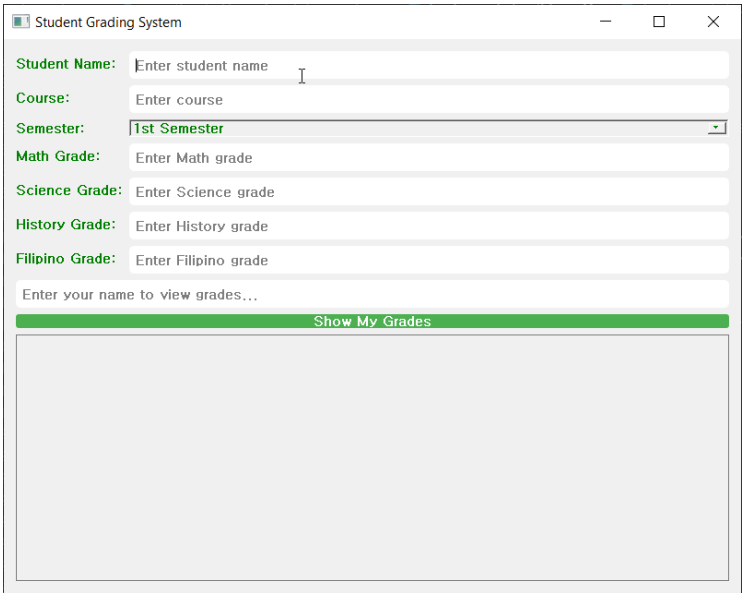
- **View Grades**: Students can view their grades for each subject, semester, and course. The system displays the letter grade based on average scores.
- **View Overall Performance**: The student can see an overall average across semesters and a cumulative letter grade.
- **Search and Filter Records**: Students can search for records by their name or by semester to quickly access relevant data.

  Letter Grades and Average Calculation: Based on the grades entered by the admin, the system calculates and displays the student's average for the semester. This average is then converted into a letter grade based on a predefined scale:

- A (Excellent): 90% or higher.
- A-: 87%–89%.
- B+: 83%–86%.
- B: 80%–82%.
- B-: 77%–79%.
- C+: 73%–76%.
- C: 70%–72%.
- C-: 67%–69%.
- D+: 63%–66%.
- D: 60%–62%.
- F (Failing): Below 60%.

Explanation:

This functionality provides students with the necessary tools to keep track of their academic progress in a user-friendly environment. Students can make more informed decisions regarding their academic trajectory based on performance data.



---

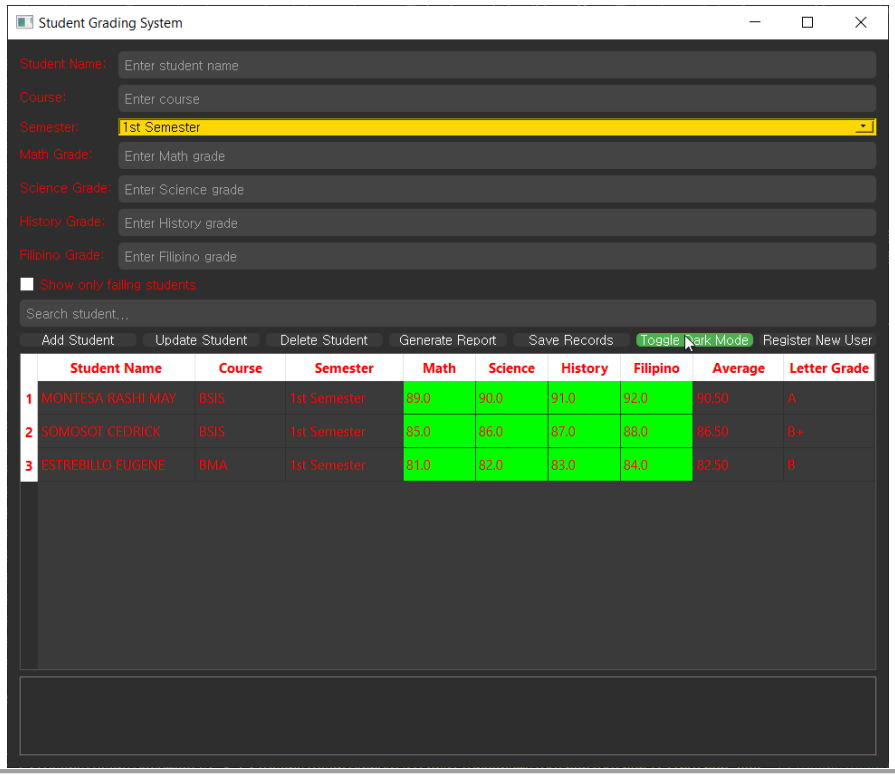
4. User Interface (UI)

The user interface is designed with usability in mind:

- **Responsive Design**: The UI layout automatically adjusts to different screen sizes.
- **Dark Mode & Light Mode**: Users can switch between dark and light themes, which enhances visual comfort and accommodates different preferences.
- **Error Handling**: The system provides clear and informative error messages when the user enters invalid data (e.g., non-numeric grades or missing required fields).

Explanation:

A clean and responsive UI ensures that the system is accessible to all users, whether they are students or administrators. The dark mode and light mode settings provide customization options to enhance the user experience.



Components Used

1. Python Concepts and Libraries

Object-Oriented Programming (OOP):

The program is built using **object-oriented principles**, with distinct **classes** representing entities like students, courses, and grades. This approach ensures that the system is modular, easily extensible, and maintainable.

- **Student Class**: Manages student details, including grades and semester information.
- **Grade Class**: Handles grade calculation and categorization.
- **Admin Class**: Manages administrative functionalities, such as adding or updating student records.

File Handling:

- **CSV** is used for storing student records, allowing the data to be exported easily and read back into the system.
- **JSON** is used for authentication storage (username and password), enabling quick, lightweight storage and retrieval of user credentials.

Event Handling:

Event-driven programming is used to handle user actions (button clicks, data input) in real time. This approach keeps the UI responsive and interactive.

Libraries Used:

- **PyQt5**: For building the graphical user interface (GUI). It provides a wide range of widgets and layouts that help in creating an interactive and visually appealing UI.
- **CSV**: For exporting and importing student records in a readable format.
- **re (Regular Expressions)**: For validating user input, such as ensuring the student name contains only letters and spaces.

Explanation:

The use of **OOP** ensures that each aspect of the program is modular and maintainable. By separating concerns into different classes, future updates and changes can be applied without significant changes to the core logic. The choice of **PyQt5** for the GUI ensures compatibility across different operating systems (Windows, macOS, Linux).

User Guide

Step-by-Step Instructions:

Installation:

1. **Install Python:**
    - **Ensure Python 3.6+ is installed on your machine.**
    - **Download from python.org.**
2. **Install Required Libraries: Install the required libraries using pip:**

```
PS C:\Users\user\Documents> pip install pyqt5
```

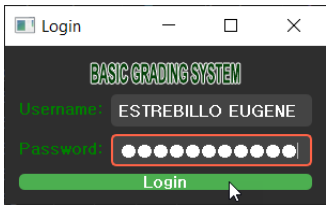
- Clone or Download the Code:
- Clone the repository or download the files from your source.
- Navigate to the project folder.
- Run the Program:
- Open a terminal or command prompt, navigate to the project folder, and run:
- Open a terminal or command prompt.
- Navigate to the folder containing the program and run the following command:

```
PS C:\Users\user\Documents> python main.py
```

- This will launch the graphical user interface (GUI).

2. Login:

- Enter your **username** and **password**.
- Admins will be redirected to the **admin panel** where they can manage student records.
- Students will be redirected to their **personal dashboard**, where they can view their grades.



3. For Admin Users:

- Admins can add, update, or delete student records from the admin interface.
- Admins can export records to **CSV** for reporting.

4. For Student Users:

- Students can view their grades and overall performance on the dashboard.
- If a student's grades are not available, they can contact the admin.

5. Switching Themes:

- Users can toggle between **light mode** and **dark mode** using the provided toggle button on the interface.

Student Record Management Application - User Guide

Welcome to the **Student Record Management Application**! This tool allows you to manage student records, including personal details, course information, and grades. Below are detailed instructions on how to use the application for the first time.

1. Installation and Setup

Before you begin using the application, ensure that the following prerequisites are met:

- **Python 3.x** should be installed on your system.
- **PyQt5** (for the graphical interface) should be installed. If it's not installed, you can install it via pip:

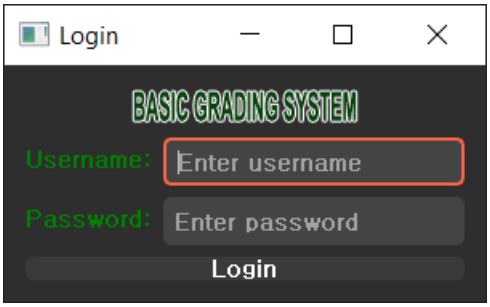
```
PS C:\Users\user> pip install PyQt5
```

- **CSV File**: The application stores student records in a CSV file named student_records.csv. Ensure that this file is in the same directory as the application.

Once the prerequisites are set up, launch the application by running the script.

2. Logging In

Upon launching the application, you will see a **login screen** and then proceed to the **main interface** where you can manage student data.



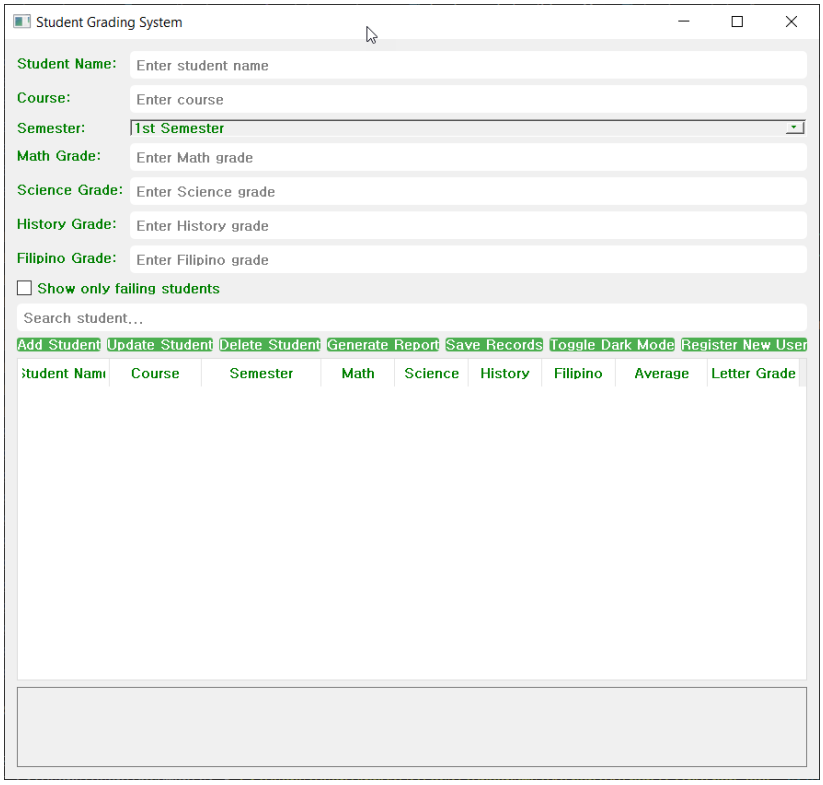
- **Admin Access: If you are an admin, enter your credentials (username and password) and click Login.**
- **Student Access: If you are a student, enter your name and click Login.**

Once logged in, you will have access to the following features based on your role:

- **Admin: Can view, add, update, and delete any student's record.**
- **Student: Can only view their own record and grades.**

Main Sections of the Application:

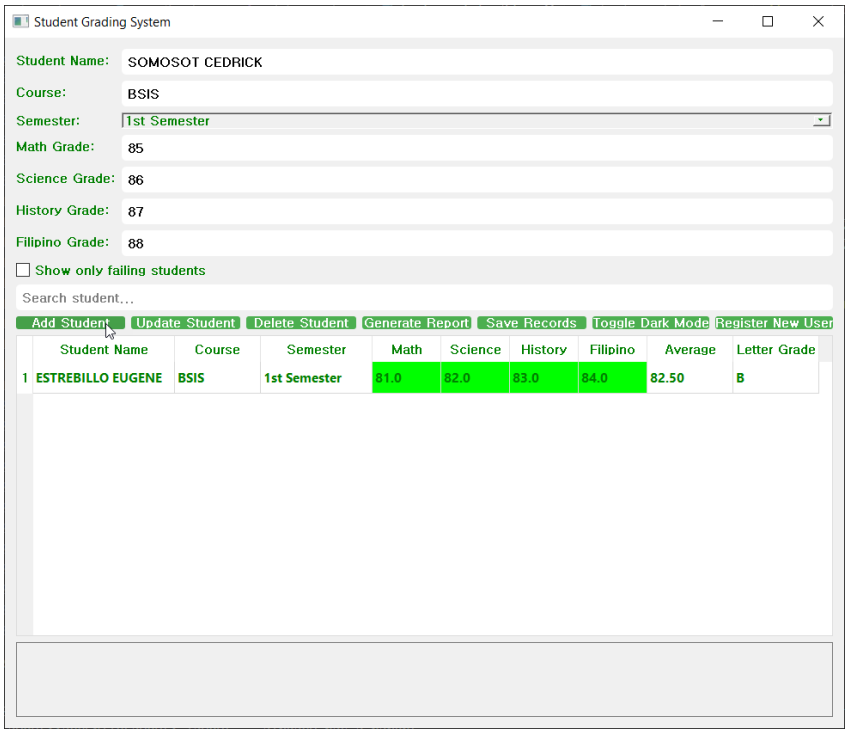
1. **Student Records Table**: Displays a table of student records including their name, course, semester, grades, and letter grade.
2. **Student Form**: A form where you can add or update student records (name, course, grades).
3. **Search Bar**: Allows you to search for student records by name.
4. **Mode Toggle (Light/Dark)**: You can switch between light and dark themes for the interface.



3. Managing Student Records

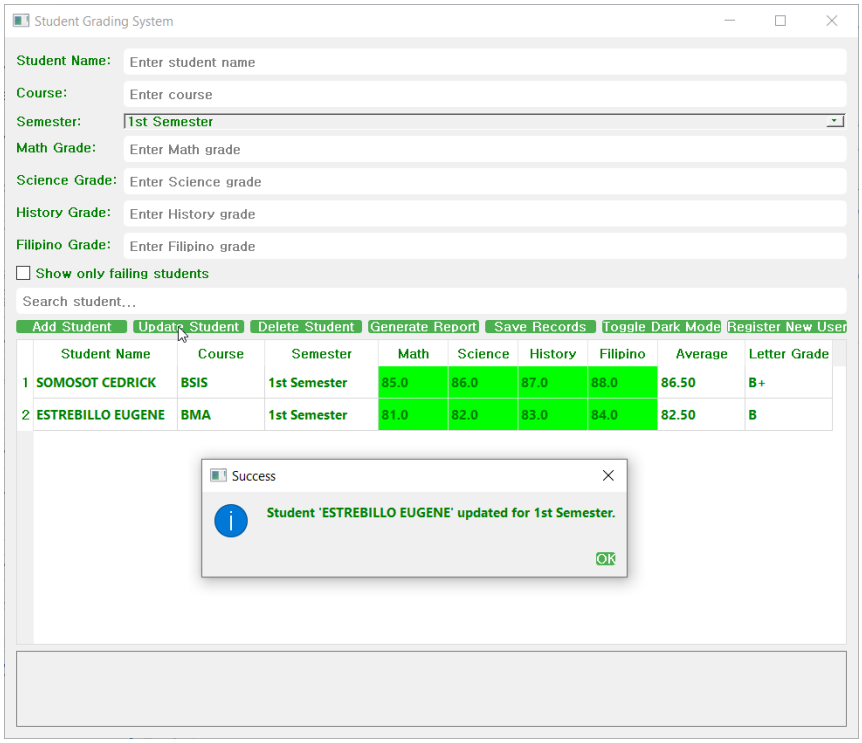
Adding a New Student Record:

1. **Navigate to the Student Form section.**
2. **Fill in the following fields:**
   - **Student Name: Enter the student's full name (only letters and spaces are allowed).**
   - **Course: Enter the course the student is enrolled in.**
   - **Semester: Select the semester (e.g., "Fall 2024") from the dropdown list.**
   - **Grades: Enter grades for the following subjects (math, science, history, Filipino) in the corresponding fields. Ensure that you enter valid numerical grades (e.g., 85.5).**
3. **Once the form is completed, click the Add Student button to save the record.**
4. **If successful, the new record will appear in the Student Records Table.**



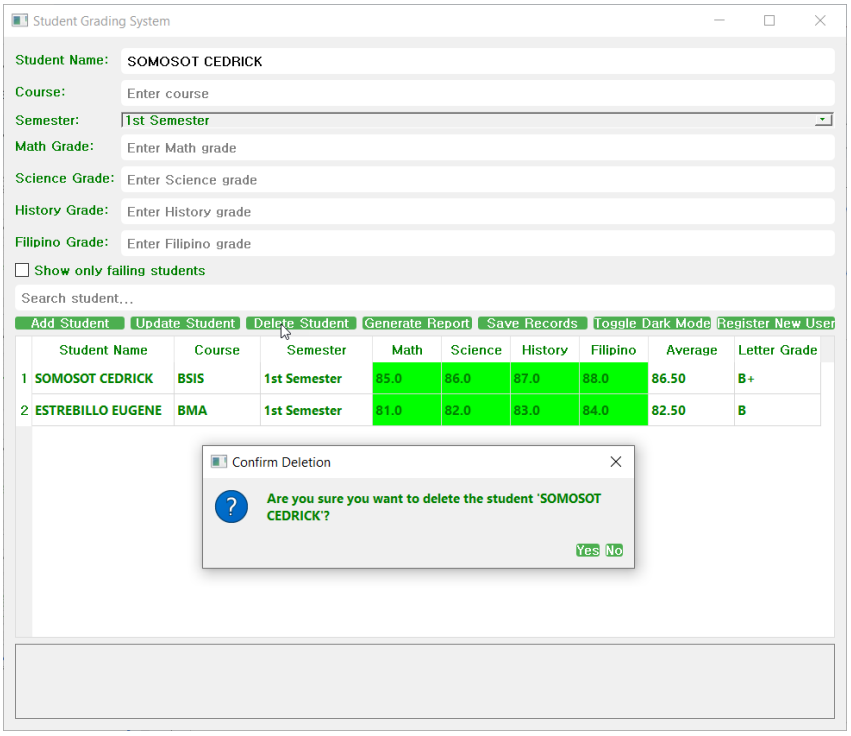
Updating an Existing Student Record:

1. **To update an existing record, search for the student's name in the search bar.**
2. **Select the student's name and semester you wish to update.**
3. **Modify the course or grades as needed.**
4. **Click the Update Student button to save the changes.**
5. **The updated record will be reflected in the Student Records Table.**



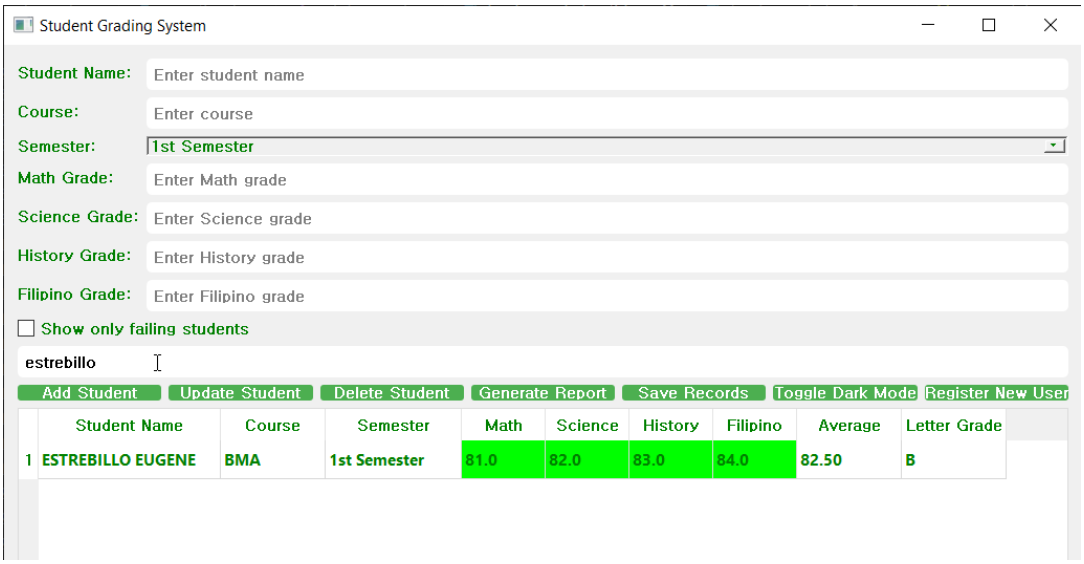
Deleting a Student Record:

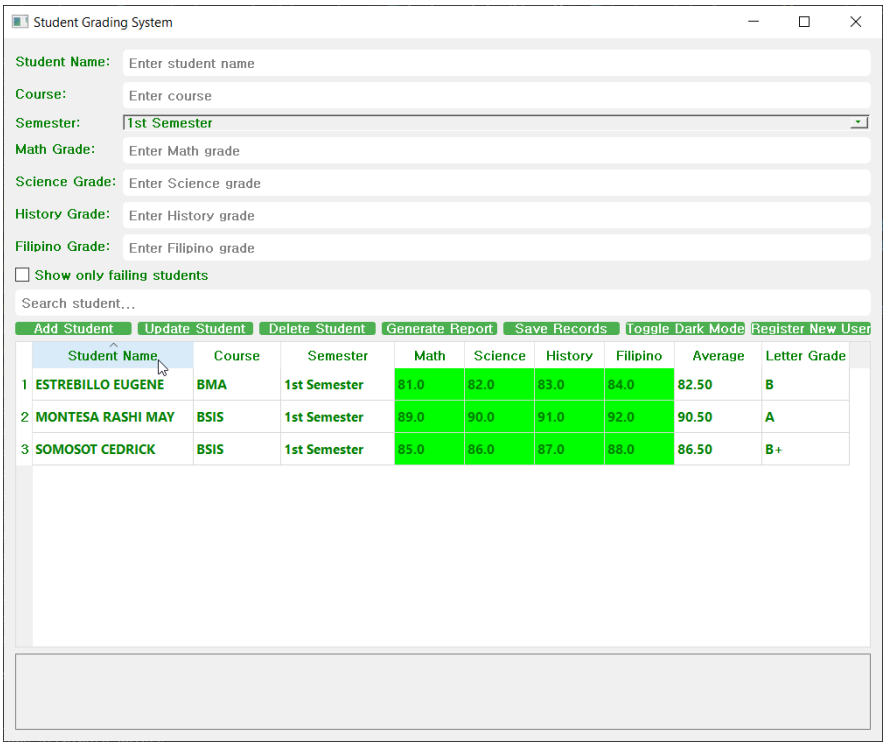
1. **To delete a student record, search for the student's name in the search bar.**
2. **Select the student's record you want to delete.**
3. **Click the Delete Student button. You will be prompted to confirm the deletion.**
4. **Confirm the deletion by clicking Yes. The record will be removed from the system.**



5. Viewing and Sorting Student Records

- **Search Function: Use the search bar to filter student records by name. Admin users can search for any student's record, while students can only search for their own name.**
- **Sorting by Grades: Student records are sorted by average grade in descending order, so students with the highest grades appear at the top of the list.**

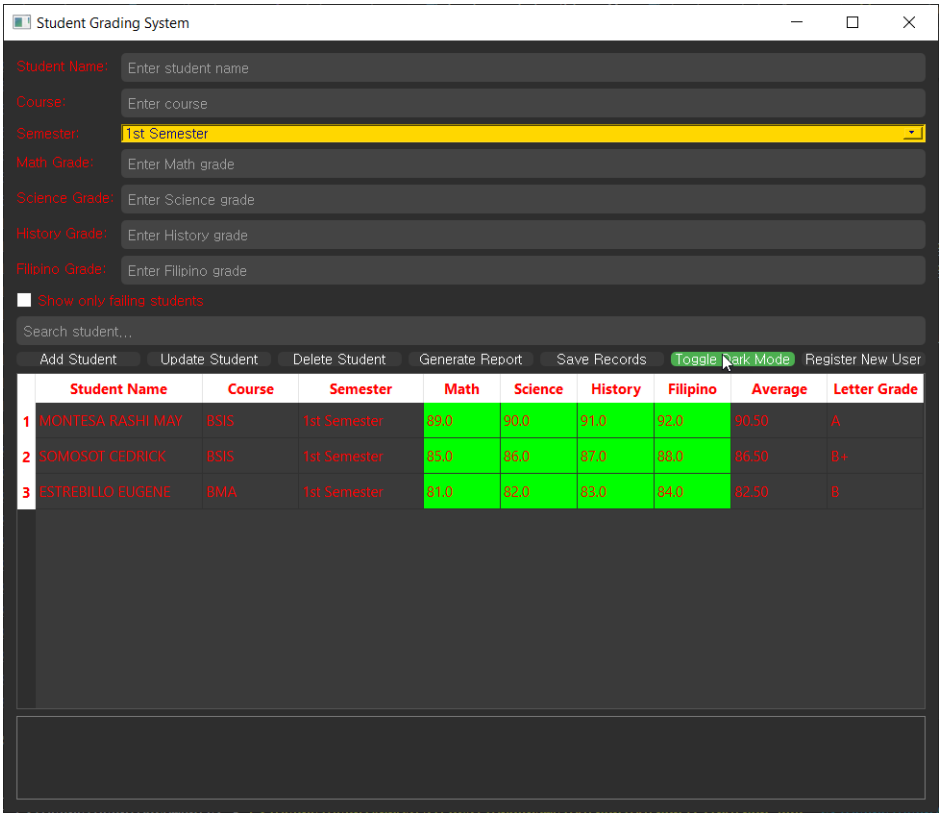




6. Light and Dark Modes

The application supports both Light Mode and Dark Mode to suit your preference.

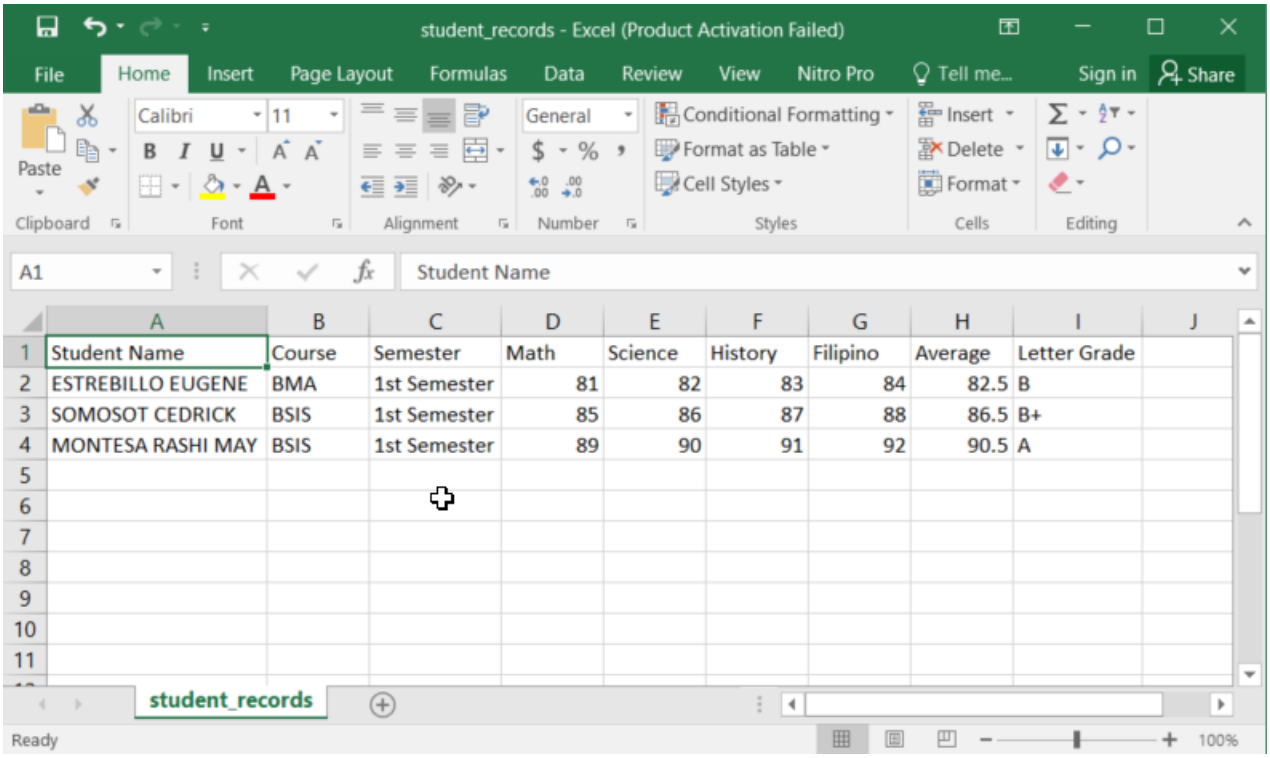
- **To switch between modes, click the Mode Toggle button on the main interface.**
- **When in Light Mode, the background will be white, and text will be black.**
- **In Dark Mode, the background will be dark, and text will be light.**



7. Saving and Loading Records

- **Saving: The application automatically saves all student records to a CSV file (student_records.csv). You can see the saved data even after restarting the application.**
- **Loading: On application startup, all records saved in student_records.csv will be loaded into the system automatically.**

If there is an error while saving or loading the records, the application will show a message indicating the issue (e.g., file not found or permission denied).

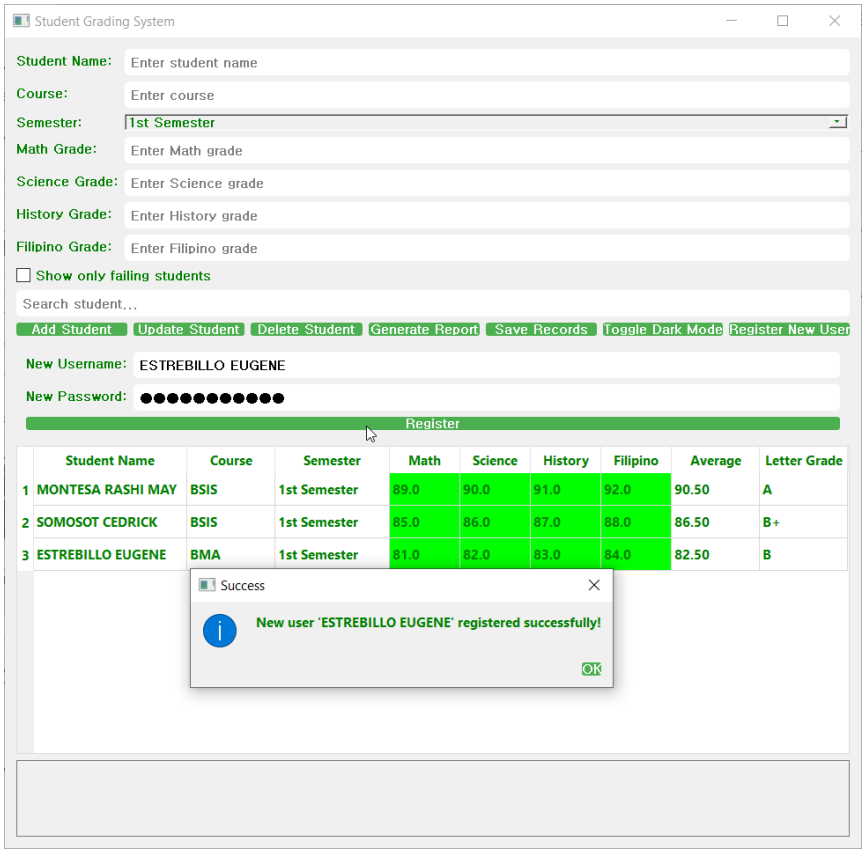


	A	B	C	D	E	F	G	H	I	J
1	Student Name	Course	Semester	Math	Science	History	Filipino	Average	Letter Grade	
2	ESTREBILLO EUGENE	BMA	1st Semester	81	82	83	84	82.5	B	
3	SOMOSOT CEDRICK	BSIS	1st Semester	85	86	87	88	86.5	B+	
4	MONTESA RASHI MAY	BSIS	1st Semester	89	90	91	92	90.5	A	
5										
6										
7										
8										
9										
10										
11										

8. Access the Registration Form:

- **Only Admin can register new users. Navigate to the Register New User section in the admin panel.**
- **Fill Out the Form:**
- **Username: Enter a unique username.**
- **Password: Enter a password for the new user.**
- **Click Register to add the new user. If any field is empty, an error message will prompt you to fill them in.**
- **The new user will be saved with the role of student.**
- **A confirmation message will appear: "New user 'username' registered successfully!".**
- **The form will reset and hide after registration.**

This feature helps Admins register users with basic details (username, password) for system access.



And the user data will automatically save in json file

Limitations and Future Improvements

Limitations:

1. ***Security****:*
    - **Password Storage**: The current implementation stores passwords in plain text, which is a **security risk**. A more secure approach would involve password **hashing** (e.g., using SHA256 or bcrypt).
    - **Authentication**: The authentication system is based on a local JSON file. It is vulnerable to manipulation and should be replaced with a **database**-based authentication system for enhanced security.
2. ***Scalability****:*
    - The system uses **CSV** and **JSON files** for data storage, which may not scale well for large datasets. A more scalable solution would be to use a **database system** such as **SQLite** or **MySQL**.
3. ***Validation****:*
    - Input validation is relatively basic (e.g., checking if grades are numbers), but it could be enhanced with **range checks** (e.g., grades should be between 0 and 100) and other validations like **email format** or **missing fields**.
4. ***No Multi-User Support****:*
    - The system is built for a single-user experience per session. Implementing multi-user support could allow different users to log in simultaneously, which is especially useful for large institutions.

Future Improvements:

1. ***Enhanced Security****:*
    - Implement **password hashing** for better security.
    - Transition to a **database-backed authentication** and storage system (e.g., SQLite, PostgreSQL).
2. ***Database Integration****:*
    - Switch from using JSON and CSV to a **relational database system (e.g., MySQL, SQLite)**, which would offer better data integrity, concurrency support, and scalability.
3. ***Advanced Search and Filtering****:*
    - Implement more sophisticated filtering and sorting for admin users, such as searching by semester, course, or grade range.
    - Provide **data analytics** for students, such as trend analysis (performance over time) or suggestions for improvement.
4. **Web/Mobile Application**:
    - Transition to a **web application** using a framework like **Flask** or **Django**, enabling access from any device with a browser.
    - Alternatively, create a **mobile app** for more convenient access.

5. ***User Interface Enhancements**:*
    - Improve the UI to make it more visually appealing and user-friendly, possibly by adding **charts** or **graphs** to visualize performance trends.

Conclusion

The ***Student Grade Management System*** serves as a practical solution for managing student academic records in educational institutions. The program is user-friendly, with role-based access for students and administrators, and it automates important tasks such as grade calculation and reporting.

Though the current implementation meets basic requirements, there are significant areas for improvement, especially in security, scalability, and multi-user support. Future enhancements, including the adoption of a more robust database solution, improved security practices, and the introduction of advanced features, will elevate the system to a more comprehensive educational tool.

This project demonstrates the potential for developing a fully-fledged student information system that can be extended to meet the needs of modern educational institutions.