Student Grading System

Objective

The **Student Grading System** is a desktop application designed to help educators or administrators manage student records and their academic performance. The system allows users to input and modify student grades across various subjects, generate reports of student performance, and store the data in a CSV file. It also includes basic features like theme switching between light and dark modes, and the ability to filter student records by name.

The primary goal of this program is to simplify the task of grade tracking, reporting, and management in an educational environment.

Features

1. **Login System**:
    - Basic Authentication:
        - The login system currently relies on hardcoded credentials (admin / password123), which is fine for a demo or proof of concept, but in a real-world application, you'd want to implement more secure methods like password hashing (e.g., using bcrypt or hashlib).
        - For added security, you could also allow user roles (admin, teacher, students) and implement more advanced access control.
2. **Student Record Management**:
    - Add, Update, Delete Records
        - Add new student records. This is a core functionality that allows an admin or educator to maintain a dynamic database of student information, including grades.
        - Update existing student records, including grades and course information. It can further enhance this by adding validation to ensure data integrity (e.g., checking for duplicate records, handling missing data).
        - Delete student records. It could also implement more sophisticated error handling—e.g., prompting the user with a warning if they try to delete a record with existing dependencies (e.g., semester or report data).
3. **Grade Input and Management**:
    - **Grades per Subject**:
        - The system supports the input of grades for multiple subjects like Math, Science, History, and Filipino. One idea for future improvement could be to allow customizable subjects or course types in case the system needs to be adapted for different educational contexts (e.g., arts, physical education, etc.).
        - Update grades for any student in a particular semester.
        - **Validation**: Ensure that grades fall within a valid range (0-100), which could easily be added as a check before submitting the grade.

4. **Student Report Generation**:
    - **Detailed Reports**
        - Generate and display a detailed report of student performance, including grades per subject, semester, average grade, and letter grade.
        - The report can be viewed in a read-only QTextEdit widget within the application.
5. **Search Functionality**:
    - **Filtering Records by Name**:
        - Filter the student records by name using a search bar.
        - This feature would be useful for quickly retrieving student data, especially if there are many students in the system. It could be enhanced by adding filters for other criteria like grade range or subject.
6. **Theme Switching (Dark Mode/Light Mode)**:
    - **User Interface Customization:**
        - Toggle between light and dark themes to suit user preferences.
        - The ability to toggle between dark and light themes is a nice touch for improving user comfort, especially for long sessions of data entry or review.
7. **Data Persistence (CSV Support)**:
    - **Saving and Loading Student Records**:
        - Save and load student records from a CSV file.
        - Automatically load records upon program startup and save them before program exit.

Components Used

Python Concepts:

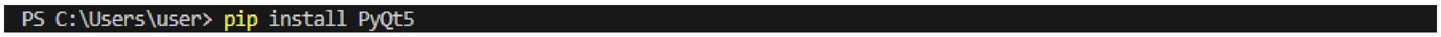
- **Object-Oriented Programming (OOP)**:
    - The program is structured around classes, where the core classes are LoginWindow (for the login interface) and StudentGradingSystem (for the main student management interface).
- **Event Handling**:
    - The application uses PyQt5's signal-slot mechanism for event handling, such as when a button is clicked or a text field is modified.
- **File I/O (CSV Handling)**:
    - Python's built-in csv module is used to read from and write to CSV files, allowing the system to save and load student records.
- **Layouts and Widgets**:
    - PyQt5's layouts (QVBoxLayout, QFormLayout, QHBoxLayout) and widgets (QPushButton, QLineEdit, QTableWidget, QComboBox, etc.) are used to build the graphical user interface.

- **Stylesheets**:
    - Custom Qt stylesheets are applied to create the dark and light mode themes for the UI.
- **Mathematical Calculations**:
    - Average grades are calculated based on inputted grades, and letter grades are derived from these averages.

User Guide

Installation Steps:

1. **Install Python**:
    - Ensure you have Python 3.x installed on your system.
2. **Install Required Libraries**:
    - You will need the PyQt5 package, which can be installed using pip:



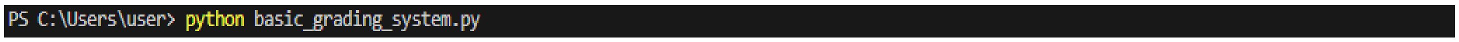
```
PS C:\Users\user> pip install PyQt5
```

3. **Prepare the Project**:

- Save the Python script (basic_grading_system.py) in a directory of your choice.
- Ensure the directory contains a logo.png image for the login screen (or update the path in the code).

4. **Run the Program**:

- Navigate to the directory containing the script in your terminal or command prompt.
- Execute the program using:



```
PS C:\Users\user> python basic_grading_system.py
```

How to Use the Application:

1. **Login**:
    - Open the program, and you will be presented with a login screen.
    - Use the default credentials:
        - Username: admin
        - Password: password123
2. **Student Grading System Interface**:
    - After logging in, you will be directed to the main interface where you can:
        - **Add a Student**: Enter the student's name, course, semester, and grades for each subject, then click "Add Student".

        - **Update Student**: Modify a student's existing record by selecting their name, updating the grade fields, and clicking "Update Student".
        - **Delete Student**: Remove a student’s record from the system by entering their name and clicking "Delete Student".
        - **Generate Report**: Click "Generate Report" to view a detailed report of all students, their grades, average scores, and letter grades.
        - **Toggle Dark Mode**: Switch between light and dark themes by clicking the "Toggle Dark Mode" button.

3. **Search Function**:
    - You can search for a student by typing their name in the "Search student..." field. The system will display matching student records.
4. **Saving and Loading Records**:
    - The program automatically loads student records from a student_records.csv file upon startup and saves any changes made during the session when the program is closed.

Limitations and Future Improvements

Limitations:

1. **Basic Authentication**:
    - The login system is very basic, relying on hardcoded credentials with no encryption or security. This is only a placeholder for testing purposes.
2. **Data Validation**:
    - Currently, there are no checks to ensure that grade inputs are within a valid range (0 to 100). The system should validate input before accepting it.
3. **UI Design**:
    - While functional, the user interface could be further refined for better user experience and visual appeal.
4. **Single User**:
    - The program is designed for a single user. There is no multi-user support or role management (e.g., admins vs. regular users).
5. **Error Handling**:
    - Though basic error handling is included, more specific error messages and handling for edge cases (e.g., invalid CSV format, missing data) would improve the user experience.

Future Improvements:

1. **Enhanced Login System**:
    - Implement password hashing and add multi-user support for more secure and flexible access control.
2. **Advanced Data Validation**:

    - Validate user inputs to ensure grades are between 0 and 100 and that required fields are filled before submission.

3. **User Interface Improvements**:
    - Enhance the UI with more intuitive navigation, form validation feedback, and perhaps use icons or tooltips for clarity.
4. **Advanced Reporting**:
    - Include the ability to generate more detailed reports, such as semester-wise performance trends, graphical representation of grades, and comparative analysis.
5. **Backup and Restore**:
    - Implement a backup and restore functionality to avoid data loss in case of file corruption or accidental deletion.
6. **Web-based Version**:
    - Port the application to a web framework (e.g., Django, Flask) to allow access from different devices and provide a more scalable solution.

Conclusion:

This Student Grading System serves as an excellent starting point for educational institutions looking to digitize student record management and grade tracking. With its current functionality and the outlined future improvements, it could evolve into a highly effective tool for managing student performance and supporting educators in their day-to-day tasks.