Please note that the given codebase is not complete and require improvements and implementation of extra features and functionalities:

**Required improvement to the codebase:**

|  |  |
| --- | --- |
| **Required Improvement Num.** | **Description** |
| **R1** | Adding code comments to explain the code. |
| **R2** | Better organize the code into different files/classes and folders (make sure your solution is a proper Object Oriented solution including the use of inheritance, encapsulation and any other applicable OOP principles). |
| **R3** | Implement a well-designed database using your proposed ERD including Students, Teachers, Courses and Registrations entities, as well as adding necessary attributes with accurate datatypes. Also, your database should include constraints such as Primary Keys, Foreign keys, referential integrity constraints, Unique, Index, Default, and Check Constraints if applicable. Your ERD should be normalised to the third normal form, at minimum. |
| **R4** | CRUD functionalities must be fully implemented to manage students, teachers, and courses records, as well as managing student registrations for different courses. Which means, all functionalities (add, update, delete, and view all records) must be working properly within your application so the data in your database can be easily manipulated. |
| **R5** | Use sorting and searching algorithms to filter and sort loaded data. (for example your application should allow search for a particular record (e.g student record) by id or name...etc) |
| **R6** | Error and Exception handling (try, except) |
| **R7** | Implementing input validation to ensure your program receives valid input. |
| **R8** | Follow PEP 8 Standards |
| **R9** | Implementing loops, so the program should run until the user decides to quit the program. (the program should not close after executing the first operation, also invalid input should promote the user to input a valid choice until the user choose to exit the program). |

**Optional improvement to the codebase:**

|  |  |
| --- | --- |
| **Optional improvement Num.** | **Description** |
| **O1** | Implementing a login feature to validate the username and password for the administrator before displaying the main menu options. (Authentication) |
| **O2** | Adding additional entities/tables within your database going beyond the minimum required entities specified in R3.   * For example, you could create 'Attendance' table for recording daily attendance details. Other examples might include creating ‘Rooms’, ‘Buildings’, or ‘Grades’ tables. |
| **O3** | Adding additional features/functionalities beyond the minimum required functionalities covered in R4.   * For example, you program could generate reports covering data across multiple tables which might require the use of SQL Join Queries to join tables and retrieve data from multiple tables. * Other examples might include functionalities within your application to manage students’ attendance, grades…etc |
| **O4** | Improving the output (usability of the system): Below are some examples:   * 1. Use the rich table to improve the output and print in a table format   from rich.table import Table   * 1. Use rich print so the output can be in different colours and styles   from rich import print as rprint |
| **O5** | Any general code refactoring/improvements (for example use of match case rather than if else when applicable) |
| **O6** | Implementing different access levels for your application with different permissions and operations to perform on each level (Authorization) |
| **O7** | Create GUIs for your application |