To create a Library Management System:

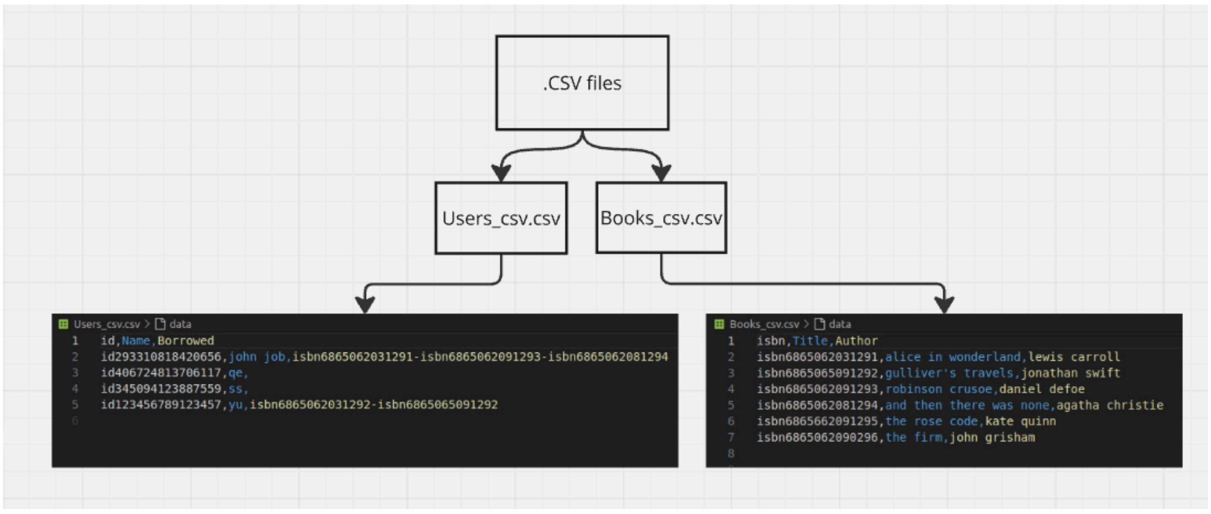
Some assumptions:

1. We are only checking which user has checked-in or checked-out which book and not keeping track of count of each copy of this book to reduce complexity.
2. I will be using panda library to perform operations on the dataset since pandas is great for handling tabular data.
3. I will be using 2 .csv files to store all data locally:
    a. Users.csv
    b. Books.csv

About the .csv files:

All addition, updates, deletion etc are reflected into these .csv files for persistent storage locally.

1. “User.csv”:
    - ID: unique system generated string of 17 digits with a prefix” ID” (example: id293310818420656)
    - Name: should have a max length of 255 chars, must be only alphabets
    - Borrowed: a list containing all the books a user has borrowed, with a maximum of 10 book. This is to incentivise the user to return some books to borrow further and provides us with a memory cap. The contents are ISBN ids that are strings which are concatenated together.
2. “Books.csv”:
    used to manage the following data about book:
    - ISBN: unique 17char string assigned to each book
    - Title: Title of the book
    - Author's name: Name of the author



```
id,Name,Borrowed
id293310818420656,john job,isbn6865062031291-isbn6865062091293-isbn6865062081294
id406724813706117,qe,
id345094123887559,ss,
id123456789123457,yu,isbn6865062031292-isbn6865065091292
```

```
isbn,Title,Author
isbn6865062031291,alice in wonderland,lewis carroll
isbn6865065091292,gulliver's travels,jonathan swift
isbn6865062091293,robinson crusoe,daniel defoe
isbn6865062081294,and then there was none,agatha christie
isbn6865662091295,the rose code,kate quinn
isbn6865062090296,the firm,john grisham
```

All Modules

Users.py

About:

1. Deals with the following attributes in the user.csv file:
    - ID: unique 17char string assigned to each book with prefix ID
    - Name: Name of the user

2. Format and validate any input to be used using functions defined

Common functions:

- Add a new row of details
- Delele a row:
    a. Only if the person does not have any borrowed books
- Update existing
- List all
- Search
- Readf_from_dataset
- Write_dataset_to_csv

Functionalities:

1. validate_id *:= check validity of user's unique id number*
2. search_among_user_attributes:*= searches based on attributes*
2.1. search_id *:= searches by id number*
2.2. search_name *:= search by name*
3. generate_unique_id *:= generates unique id for user*
4. update_an_existing_user_detail *:= update a cell in existing dataset*

5. list_all_user *:= print to screen all the users*
6. add_a_user := adds a new row with details of a new user
7. delete_a_user_based_on_id := delete a user data based on id number

Books.py:

About:

used to manage the following book details :

1. ISBN: unique 17char string assigned to each book
2. Title: Title of the book
3. Author's name: Name of the author

Common functions:

- Add a new row of details
- Delete a row
- Update existing book details
- List all rows
- Search for a specific elem

Functionalities in the code:

1. validate_isbn *:= checks validity of entered isbn*
2. validate_title *:= checks validity of validate_title*
3. search_among_book_attributes *:= searches for rows with matching string*
3.1 search_isbn *:= search by isbn value*
3.2 search_author *:= search by authors name*
3.3 searc_title *:= searches by title*
4. generate_unique_isbn *:= generate unique_isbn value*
5. update_an_existing_book_detail *:= update value in a row*
6. list_all_book *:= list all contents to CLI*
7. check_if_book_exists_using_isbn *:=validity of isbn*
8. check_if_book_exists_using_title_author *:= validity using title/author*
9. check_if_book_exists_using_isbn_title_author := *validity using isbn/title/author*
10. add_a_book *:= add a new row*
11. delete_a_book_based_on_isbn *:= deletes a record*
12. save_book_df_to_csv *:= save to .csv file*

Utilities.py:

About:

Contains methods used to format and validate the following:

1. For users dataset: userid, name, Borrowed
2. For books dataset: isbn,titles, authors

Functionalities:

1. format_string := removes trailing spaces, special chars etc
2. validate_name := validates the name according to preset rules
3. validate_choice_and_available_choices := used to check inputs from CLI and validate it
4. Exit := exits the program gracefully

Storage.py

About:

Class deals with:

1. checks if the user required .csv files are present in the same directory as one we are running the program from
2. if not found, a file is created, initialised and loaded into a dataframe to be used.

Functionalities:

1. __init__ := loads the dataframe from .csv
2. storage_operation := checks file existance, creates one if one doesn't exist, and initialises it
3. return_complete_file_path := used to get the directory path we are operating from.

Check.py

About

Used to show borrowing and returning of books in a library. This is based on the "Users.csv" dataframe.

Records books being checked out and checked in. This is reflected under the “Borrowed” feature in the users.cvs. We could have created a dedicated csv, but choose to add to the users.csv file to reduce complexity.

1. Borrowing:Ask which book the user wants to borrow.
    a. Search via isbn or title, enter full text in each case
    b. confirm it, then add it to “Borrowed” under the users.csv file.

2. Return: since there are only 10 books max that the person can borrow, print it to screen and ask to enter the corresponding integer, remove it from the “Borrowed”.
3. All changes to be reflected in .csv
4. Edge: returning book that user did not borrow

Functionalities:

1. __init__ *:= loads the users_df dataframe, making it available for rest of code*
2. all_borrowed_book *:= outputs list of all borrowed books*
3. borrow_a_book *:= handles everything related to borrowing of a book*
4. borrow_book_internal := *the internal code that is run when we borrow a book*
5. return_a_book *:= handles everything related to returning of a book*
6. return_book_internal *:= the internal code that is run when we return a book*

Storage.py

About:

Class checks if the user required .csv files are present in the same directory as one we are running the program from. if not found, a file is created, initialised and loaded into a dataframe to be used.

Functionalities:

- __init__ := loads the dataframe from .csv
- Storage_operation := checks file existance, creates one if one doesn't exist, and initialises it
- return_complete_file_path := used to get the directory path we are operating from.

Library_mangemeny.py

About:

Inherits from BooksManager, UsersManager, CheckManager, StorageManager classes.This class will be the point of contact for commands from the interface / CLI.

Functions:
Combines the functionalities of all the modules discussed above.

Main.py

Used to create the interface. This handles only the interactions, but all operations are performed by the Library manager class. Following is the primary interface.

```
"Library Management System"
----------------------------
"Main Menu"
"1. List Books"
"2. List users"
"3. Add a Books"
"4. Add a User"
"5. update a book's detail"
"6. update a user's detail"
"7. search for a book"
"8. Search for a user"
"9. Borrow a book"
"10. Return a book"
"11. Exit"
```

Design pattern:

Have used Singleton design pattern to maintain only a single instance of the LibraryManagement class.
This was chosen to:

1. Ensure that multiple instances are not accessing or writing to the same memory
2. We simplify the design by providing a central access to all the attributes and methods provided by all the differents classes.

How to run the program:

Run the python file **main.py**

Unit testing:

To have a feedback on how the code reflects on inputs and edge cases, unit testing is performed, files are:

1. Test_book_unit.py for testing book.py
2. Test_model.py for testing model.py
3. Test_user.py for testing user.py

For further information doctrine are provided for methods in every classes defined, please refer to it.
