Library Management System (Pair Project)

Goal

You and your partner will build a **Library Management System** using Python.
This project will give you practice with:

- Writing classes in Python (OOP)
- Organizing your code
- Reading and writing files (JSON and/or CSV)
- Working with Git (branches, commits, merging)
- Basic teamwork (splitting tasks, planning, reviewing)

Team Roles

- **Developer:**
    - Focuses mainly on writing the code.
    - Implements the classes and functions.
- **Project Manager + Developer:**
    - Helps with coding **and** manages planning.
    - Keeps track of tasks in a simple TODO file.
    - Manages the Git repository (branches, merging, pull/push).

Both roles still write code, but the "manager" also handles organization.

Requirements

Classes (OOP)

You must create at least these 3 classes:

1. Book

    - Attributes: title, author, ISBN, is_available (True/False)
    - Methods: maybe __str__ to print book info

2. User

    - Attributes: name, id, borrowed_books (list of Book objects or book IDs)

3. Library

    - Attributes: list of books, list of users
    - Methods:
        - add_book(book)
        - add_user(user)
        - borrow_book(user_id, book_isbn)
        - return_book(user_id, book_isbn)
        - list_available_books()
        - search_book(title or author)

Features

- Add new books and new users.
- Borrow a book (book becomes unavailable).
- Return a book (book becomes available again).
- Show all available books.
- Search for a book by title or author.

File Handling (Persistence)

- Save the list of books and users into a **JSON file**.
- Load the data from the file when the program starts.
- **Optiona**l **(bonus):** additionally, export the data to **CSV format**.
  (Hint: Will you need only one CSV file? Think about it).

SOLID Principles (Intro)

Don't worry if you don't fully remember SOLID yet.
Just try to follow these **simple beginner rules**:

- **Single Responsibility Principle (SRP):** Each class should do *one main thing*. Example: Book only holds book data, it does not manage borrowing.
- **Open/Closed Principle (OCP):** Your code should be easy to **add to** without changing everything. Example: If you want to add another search method (by ISBN), you should not break old code. *(Tip: If you need new behavior, try adding a new method or subclass instead of changing existing ones)*
-

If you can apply at least **one or both of these ideas**, that's enough.

Git Workflow

- One person creates the repo and invites the partner.
- Each person works on a **separate branch** (e.g., book-class, library-class).
- Commit small changes often (don't wait until the end).
- Write **meaningful commit messages** that explain *what* you changed in a few words.
  I.e.:
  - **YES** → `Add Library class and basic borrow() method`
  - **NO** → `fix stuff` or `update`
- Merge branches using pull requests (or simple merges if time is short).

Deliverables

- **Python code** with classes
- **JSON file** with saved data
- **README.md** with:
  - Features you implemented
  - Team members and roles
  - How to run the program

Suggested Workflow

1. Decide class design together (draw a quick diagram on paper).
2. Write a TODO list with tasks and assign them.
3. Create the repo and make sure both can commit.
4. Each person starts with a simple class (Book/User).
5. Combine into the Library class.
6. Add file saving/loading (JSON).
7. Test borrowing/returning/search.
8. Write README.

Hint: Building the Menu in main.py

To interact with the library, you can create a simple menu using a while loop:

1. Display options to the user using print().
2. Get user input with input() to know which action to perform.
3. Use if/elif statements to call the correct Library methods (e.g., add_book(), borrow_book()).
4. Repeat the loop until the user chooses an exit option.
5. Provide simple feedback after each action (e.g., "Book added!" or "Alice borrowed '1984'").

Example skeleton:

```
choice = None
while choice != "7":
    print("1. Add Book\n2. Add User\n3. Borrow Book\n7. Save & Exit")
    choice = input("Enter your choice: ")

    if choice == "1":
        # ask for book info and add book
        pass
    elif choice == "2":
        # ask for user info and add user
        pass

    elif choice == "7":
        # save data and exit
        break
    else:
        print("Invalid choice, try again.")
```

Grading Criteria

- Clear OOP design (3 classes minimum)
- JSON file works (data persists)
- Git used by both partners
- SOLID idea demonstrated (SRP or OCP at least)
- README present and understandable