Project Title: SmartLibrary - A Digital Library Management System

Project Overview

SmartLibrary is an object-oriented Python program that manages a digital library. It enables users to register, borrow, return books, and track borrowing history while implementing real-world OOP principles.

Development Timeline & Key Concepts

Day 1: Creating the Core Model – Books and Users

Scenario:
A library needs to store information about books and users. Each book has a title, author, ISBN, and availability status. Users can register in the library and borrow books.

Implementation:

- Define a `Book` class with attributes such as `title`, `author`, `isbn`, and `is_available`.
- Create a `User` class that maintains `user_id`, `name`, and `borrowed_books`.

Key Learning:
Encapsulation and class attributes.

Day 2: Implementing Borrowing and Returning System

Scenario:
Users should be able to borrow books if they are available and return them after reading.

Implementation:

- Add a `borrow_book(book)` method in the `User` class that checks availability before allowing borrowing.
- Add a `return_book(book)` method to update the availability status and remove the book from the user's borrowed list.

Key Learning:
Method interactions between classes.

Day 3: Library Class – Managing Multiple Books and Users

Scenario:
A library needs to keep track of multiple books and users while handling user registrations and book inventory.

Implementation:

- Introduce a `Library` class to store `books` and `users` collections.
- Methods:
  - `add_book(book)`: Adds a new book to the catalog.
  - `register_user(user)`: Registers a new user.
  - `search_book(title/author)`: Searches for books by title or author.
  - `borrow_book(user, book)`: Manages borrowing through the `User` class.
  - `return_book(user, book)`: Handles book returns.

Key Learning:
Composition and managing object relationships.

Day 4: Implementing Book Tracking and Borrowing History

Scenario:
Each user should have a borrowing history to track their past activities.

Implementation:

- Modify `User` class to include `borrowing_history`, which stores previously borrowed books with timestamps.
- Create a `Transaction` class that logs borrowing and returning activities.

Key Learning:
Polymorphism and data persistence.

Day 5: Bringing Everything Together – Full System Integration

Scenario:
The library system should provide a user-friendly interface for interacting with books and users.

Implementation:

- Develop a command-line interface that lets users:
  - View available books.

- Register themselves.
- Borrow/return books.
- Check their borrowing history.