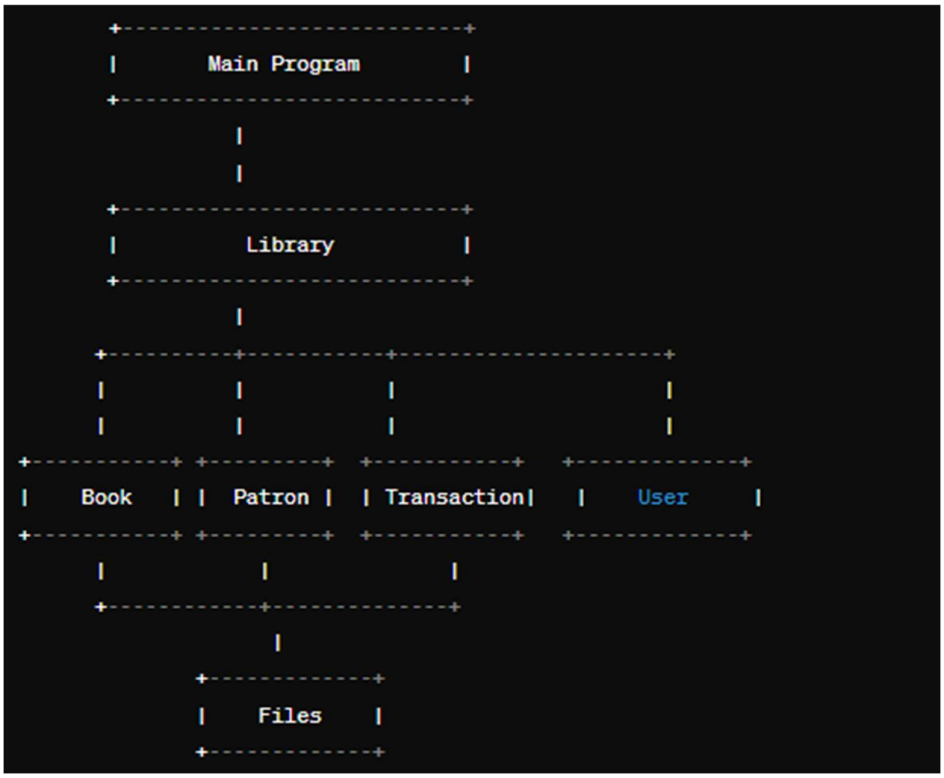
The Library Management System (LMS) is a Python-based application designed to help librarians efficiently manage books, patrons, and transactions within a library. It provides functionalities such as adding, updating, and removing books and patrons, searching for books and patrons, handling book checkouts and returns, generating reports, and managing users.

Code Structure:

The project consists of several classes: Library Management System, Library, Book, Patron, Transaction, and User.
Each class handles specific functionalities related to the library management system.
The code follows a modular and organized structure, with each class responsible for a specific aspect of the system.
1. Library Management System(main.py): This class handles the user interface and the main loop of the program.
2. Library: This class represents the library itself, including methods for managing books, patrons, transactions, and users.
3. Book: Represents a book entity, with methods for adding, updating, and deleting books.
4. Patron: Represents a patron entity, with methods for adding, updating, and deleting patrons.
5. Transaction: Represents a transaction entity, with methods for checking out and returning books, as well as calculating fines.
6. User: Represents a user entity, with methods for adding, deleting, and logging in users.



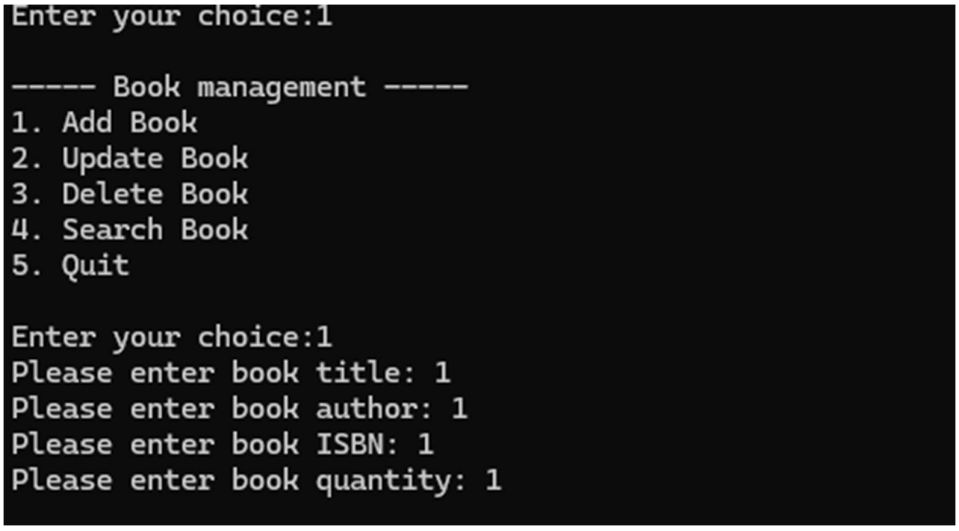
How to use:

1. Copy files to the runtime environment.
2. Run python main.py.
Super administrator account: Administrator, Password: Administrator. Ordinary administrator account: Account: root, Password: root123.
3. The database saves data files:
book.csv: Contains book data with two test entries.
patron.csv: Contains patron data with one test entry.
fines.csv: Contains overdue fines data.
transaction.csv: Contains transaction records.
user.csv: Contains administrator data.

Verification of the Sanity of the Code:

The code has been tested for basic functionality to ensure it runs without errors.

Scenario 1: Adding a Book
Choose option 1 (Book Management).
Choose option 1 (Add Book).
Enter book details (title, author, ISBN, quantity).



Scenario 2: Searching for a Book
Choose option 3 (Search Book).
Enter a keyword to search for a book.

```
Enter your choice:3
Please enter a keyword: 1

The number of query results is 4:
Title: title, Author: ww, ISBN: 1400021, Quantity: 16

Title: hello world, Author: www, ISBN: 1502, Quantity: 8

Title: 232, Author: 12, ISBN: 12, Quantity: 1

Title: 1, Author: 1, ISBN: 1, Quantity: 1
```

Scenario 3: Borrowing a Book
Choose option 5 (Borrow Book).
Enter book ISBN and patron ID.

```
Enter your choice:5
Please enter book ISBN: 1
Please enter patron Id: 1
Book with ISBN 1 updated successfully.
Book with ISBN 1 has been checked out by 1 with an expiry date of 2024-03-06.
```

Scenario 4: Generating a Report
Choose option 7 (Generate Report).

```
Enter your choice:7
***** Statistical Report ****
The number of Books: 4
the number of patrons: 1
The number of books currently on loan: 1
The total overdue fines: 6
```

Conclusion:

During the implementation of this project, several findings, challenges, and areas for improvement were encountered

Finding:

1. The project's structure, based on modular classes and encapsulation, facilitated maintainability and scalability.
2. Employing file-based (in this project I save data as csv. file to make sure my program can read it) data storage ensured the persistence of book, patron, and transaction information across system sessions.

Challenges:

1. During the programming process, it often happens that the program cannot retrieve the data of classes, and it takes a lot of time to ensure that there are no bugs.
2. I was not familiar with saving data into csv files, and I spent a lot of time going through the Internet to make sure I provided the correct code.

Limitations and Areas for Improvement:

1. It cannot be used by multiple users at the same time.

2. Using the interface without UI is very inconvenient for the user。