Library Management System:

The Library Management System is designed to manage books and users in a library. It provides functionality to add, update, delete, and search for books and users. Additionally, it handles the check-in and check-out of books, late-fees, maintaining logs for these transactions. The system ensures that only authorized managers can access and manipulate the data.

late-fees of each book per day charge is 10 rupees.

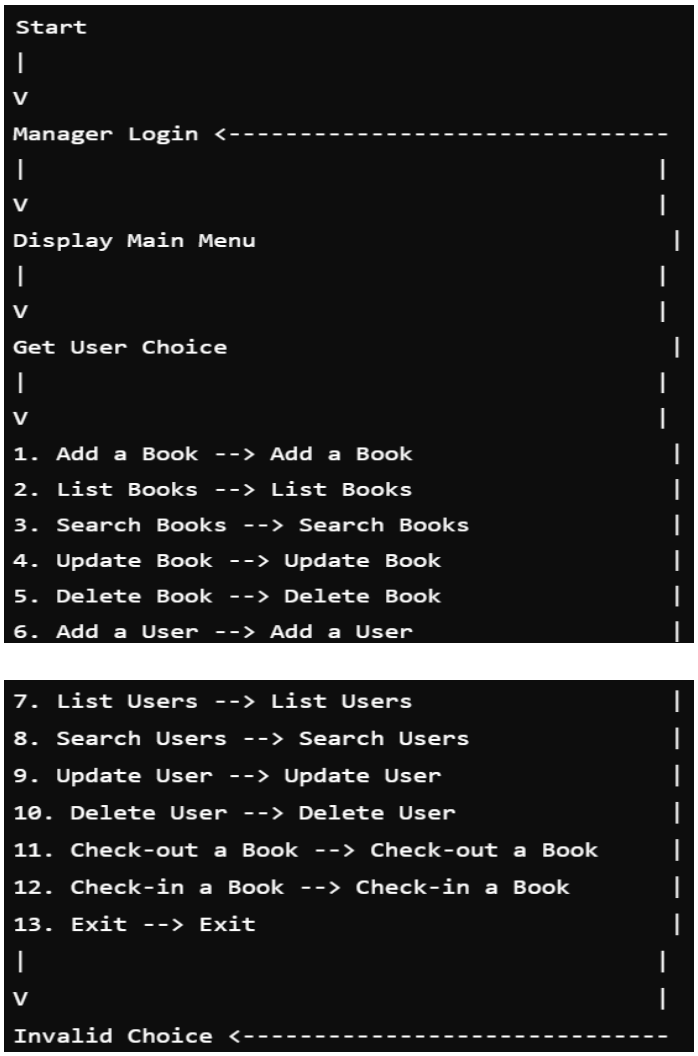
Design Decisions:

1. **Separation of Concerns**:
    - Classes and methods are organized based on functionality. Book Manager and User Manager handle book and user data respectively, while Storage integrates these with file operations.
2. **Data Persistence**:
    - Data is stored in JSON files (books.json, users.json, and book_in_out.json) to ensure persistence across sessions.
3. **Security**:
    - Manager credentials are secured with hashed passwords using SHA-256.
4. **User Interface**:
    - A text-based menu-driven interface is provided for interaction.

Architecture:

- **Book-Manager**: Manages book data including adding, updating, deleting, and searching books.
- **User-Manager**: Manages user data including adding, updating, deleting, and searching users.
- **Storage**: Integrates Book-Manager and User-Manager with file operations and manages book check-in/check-out logs.
- **main.py**: Entry point of the application, providing a menu-driven interface for the manager to interact with the system.

Flowchart for Library Management System:



1. **Start**

1. **Manager Login**
    a. Prompt for username
    b. Prompt for password
    c. Hash the password
    d. Compare with stored credentials
    e. If valid, proceed; else, display "Invalid credentials" and retry.

Main Menu

3. **Display Main Menu**

a. Options:

i. Add a Book
ii. List Books
iii. Search Books
iv. Update Book
v. Delete Book
vi. Add a User
vii. List Users
viii. Search Users
ix. Update User
x. Delete User
xi. Check-out a Book
xii. Check-in a Book
xiii. Exit

Handle Choices

4. **Get User Choice**
    1. If choice is 1, go to **Add a Book**
    2. If choice is 2, go to **List Books**
    3. If choice is 3, go to **Search Books**
    4. If choice is 4, go to **Update Book**
    5. If choice is 5, go to **Delete Book**
    6. If choice is 6, go to **Add a User**
    7. If choice is 7, go to **List Users**
    8. If choice is 8, go to **Search Users**
    9. If choice is 9, go to **Update User**
    10. If choice is 10, go to **Delete User**
    11. If choice is 11, go to **Check-out a Book**
    12. If choice is 12, go to **Check-in a Book**
    13. If choice is 13, go to **Exit**

Add Book

5. **Add a Book**
    1. Prompt for title, author, ISBN, and number of copies available
    2. Call storage.add_book()
    3. Display "Book added"
    4. Return to **Main Menu**

List Books

6. **List Books**
    1. Call storage.list_books()
    2. Display list of books
    3. Return to **Main Menu**

Search Books

7. **Search Books**
    1. Prompt for search keyword and search by (title/author/ISBN)
    2. Call storage.search_books()
    3. Display search results
    4. Return to **Main Menu**

Update Book

8. **Update Book**
    1. Prompt for ISBN of the book to update
    2. Prompt for new title, author, and number of copies (allow blanks to keep current values)
    3. Call storage.update_book()
    4. Display "Book updated"
    5. Return to **Main Menu**

Delete Book

9. **Delete Book**
    1. rompt for ISBN of the book to delete
    2. Call storage.delete_book()
    3. Display "Book deleted"
    4. Return to **Main Menu**

Add a User

10. **Add a User**
    1. Prompt for name, user ID, and optional mobile number
    2. Call storage.add_user()
    3. Display "User added"
    4. Return to **Main Menu**

List Users

11. **List Users**
    1. Call storage.list_users()
    2. Display list of users
    3. Return to **Main Menu**

Search Users

12. **Search Users**
    1. Prompt for search keyword and search by (name/user ID)
    2. Call storage.search_users()
    3. Display search results
    4. Return to **Main Menu**

Update User

13. **Update User**
    1. Prompt for user ID to update

2. Prompt for new name and optional mobile number (allow blanks to keep current values)
3. Call storage.update_user()
4. Display "User updated"
5. Return to **Main Menu**

Delete User

14. **Delete User**
    1. Prompt for user ID to delete
    2. Call storage.delete_user()
    3. Display "User deleted"
    4. Return to **Main Menu**

Check-out a Book

15. **Check-out a Book**
    1. Prompt for user ID, user name, book ISBN, and book title
    2. Call storage.log_book_check_in()
    3. Display "Book check-in successful"
    4. Return to **Main Menu**

Check-in a Book

16. **Check-in a Book**
    1. Prompt for In/Out ID
    2. Call storage.log_book_check_out()
    3. Display result of check-out
    4. Return to **Main Menu**

Exit

17. **Exit**

    Terminate the program

Invalid Choice

18. **Invalid Choice**
    1. Display "Invalid choice. Please try again."
    2. Return to **Main Menu**

Classes and Methods:

Book.py

Book: Represents a book entity with title, author, ISBN, and availability.

Book-Manager: Handles book-related operations. It maintains a list of books and provides methods to add, list, search, update, and delete books.

user.py

- **User**: Represents a user entity with name, user ID, and optional mobile number.
- **User-Manager**: Handles user-related operations. It maintains a list of users and provides methods to add, list, search, update, and delete users.

Storage.py

Storage: Integrates BookManager and UserManager with file operations. Manages book and user data persistence, and logs book check-in/check-out transactions.

- load_books, save_books, load_users, save_users, load_book_in_out, save_book_in_out: Handle file I/O operations for books, users, and check-in/check-out logs.
- log_book_check_in, log_book_check_out: Manage logging of book check-in and check-out transactions.

Main.py

login(): Handles manager login by verifying the username and hashed password.

main(): Displays a menu and handles user choices. It interacts with the Storage class to perform various operations on books and users, as well as logging book check-in/check-out transactions. The loop continues until the manager chooses to exit.