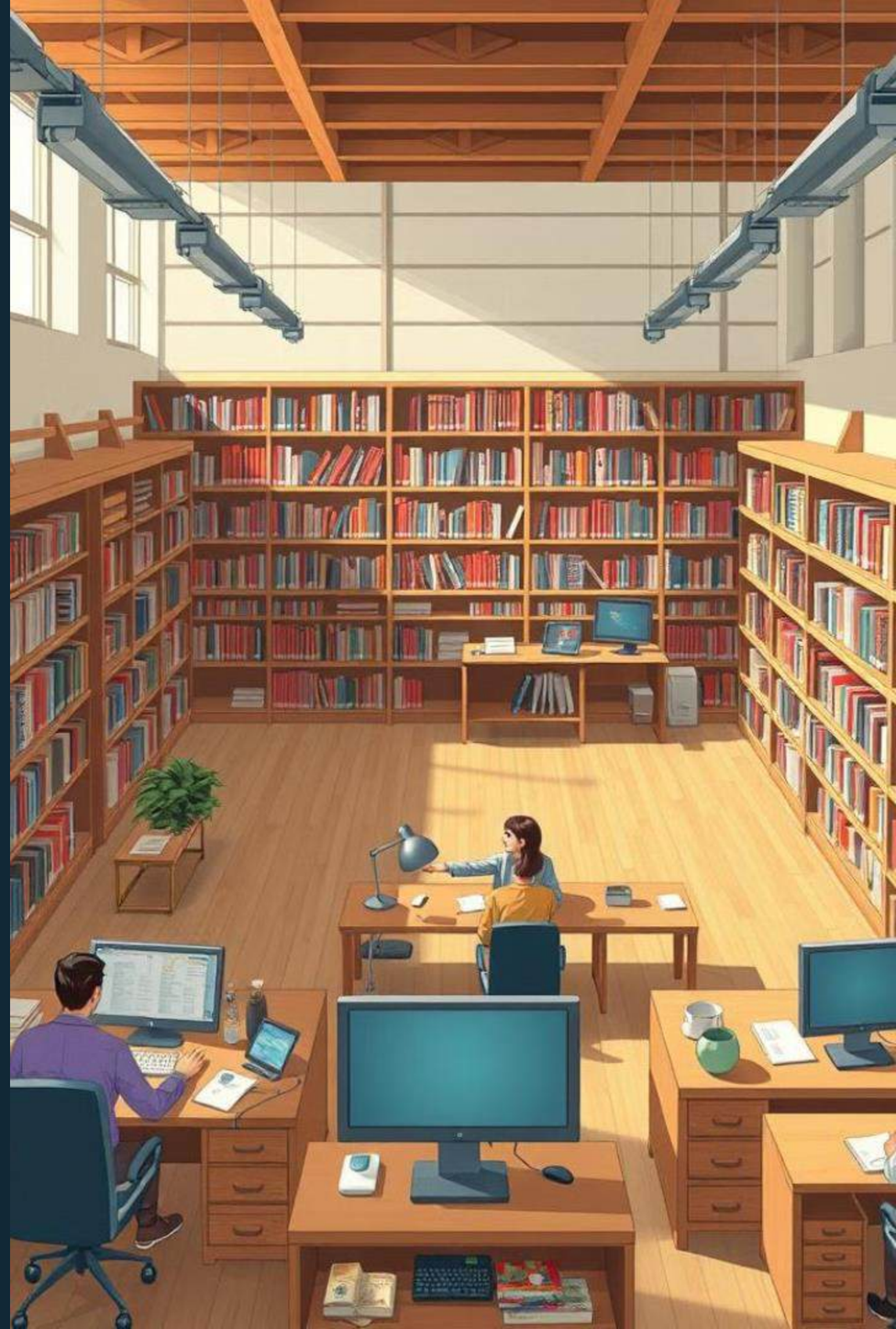
LIBRARY MANAGEMENT SYSTEM IN C

This project outlines the development of a simple Library Management System implemented in the C programming language. The project aims to provide a basic framework for managing data of the books issued ,returned and available in a library and also manages the accounts of multiple users.



TECHNOLOGIES USED

The project was developed using :

- C programming language
- Standard C library (stdlib.h , time.h, string.h)
- Text file handling to store data
- A simple interface for easy user interaction

PROJECT DESIGN

The **Library Management System** manages the day-to-day operations of a library. It allows users to add, view, issue, and return books, while maintaining a record of all data of the user which can be accessed by entering the password to the account. The system utilizes file handling to store and retrieve book and user data, ensuring that information is persistent across sessions. It supports sorting of books based on genres, and tracks the availability of each book. Users can create and access their accounts to check borrowed books and manage their details. The project is implemented in C, using concepts such as structures, file handling, and arrays, providing an easy-to-use interface for both library administrators and users. The goal is to automate the management of library operations, making it more efficient and user-friendly.

FEATURES AND FUNCTIONALITY

- **User Account Management**: Allows users to create, view, and update their accounts, providing personalized access to borrowed books.
- **Book Management**: Enables adding, viewing, editing, and deleting books in the system available. Books can be sorted by genre for easy navigation.
- **Book Issue and Return**: Tracks the issuance and return of books, updating the availability status to ensure books are properly managed.
- **File Handling for Data Persistence**: Stores user and book data in files, ensuring that the system's information is preserved even after the program is closed.
- **Genre-Based Sorting**: Books can be categorized and sorted by genre, making it easier for users to search and find books based on their interests.

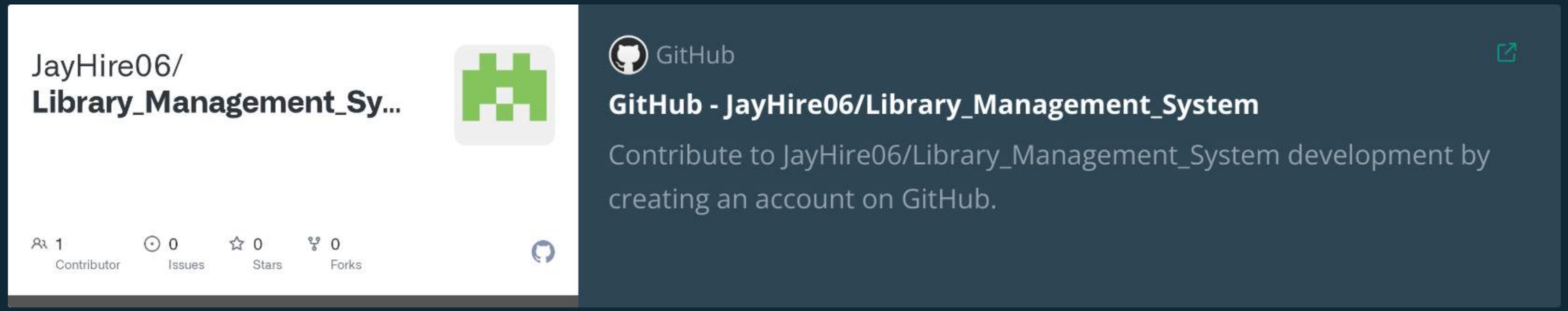
TESTING AND RESULTS

The system was thoroughly tested with various test cases, including:

- **Account Creation Test**: Ensuring the system successfully creates accounts with valid inputs and rejects invalid or duplicate inputs, displaying appropriate error messages.
- **Book Issue and Return Test**: Verify that books can be issued and returned correctly, updating availability and user borrowing records. Test with both available and unavailable books.
- **Book Search and Sorting**: Test the search functionality for finding books by title and ensure books are sorted correctly by genre.
- **File Handling Test**: Check that user and book data is correctly saved and loaded from files, ensuring data persistence between program sessions.

The testing process ensured that the system operates as intended and handles potential errors gracefully.

CODE LINK FOR THE PROJECT



CONCLUSION AND FUTURE WORK

The **Library Management System** successfully automates the basic operations of a library, such as book management and user account handling. By implementing file handling, the system ensures data persistence across sessions. The project demonstrates a practical application of C programming concepts, including structures, file Input and Output, and basic algorithms. Future enhancements could include adding advanced features like user authentication and online catalog browsing. Implementing a graphical user interface (GUI) for a more user-friendly experience.

CHALLENGES FACED

Several challenges were encountered during development, including:

- **File Handling**: Managing consistent data read/write operations while maintaining file integrity was challenging.
- **Data Validation**: Ensuring correct input formats and preventing duplicate entries required careful validation.
- **Sorting and Searching**: Implementing efficient sorting by genre and accurate search functionality as data grew.
- **Memory Management**: Properly managing memory allocation and deallocation for dynamic data structures was crucial.
- **User Interface Design**: Creating an intuitive, text-based interface that was both functional and user-friendly was a challenge.

These challenges were addressed through careful code design, thorough testing, and incorporating error handling mechanisms.

ACKNOWLEDGEMENTS

This project would not have been possible without the contributions of the following individuals and organizations:

- Course instructors, Mr. Ravi Nahta, for his guidance and support throughout the course.
- The online available blogs and other resources for C language.

GROUP MEMBERS

Member 1: Jay Hire (202452323)

Member 2: Rashi Lavekar (202451142)

Member 3: Tanmay Joshi (202452334)

Course Instructor: Mr. Ravi Nahta (IT 101)