Library Management System Documentation

{Developer: Reyes, Designer: Cueto, Tester: Espiritu}

Chosen Application Theme

The project's selected theme is library management system. The purpose of this system is to handle bookings and recently books that are returned within a library. The system manages recent changes via a stack. books that have been returned and a line to process reservations in a first-come, order of priority.

A library management system has been selected as the project's concept. This system is intended to handle recently returned books and reservations in a library. The system handles book reservations in a first-come, first-served order using a queue and manages recently returned books using a stack.

Rationale for Your Choice

The idea behind choosing this theme is to replicate actual library activities, such as patrons returning and reserving books. An effective and sensible method of managing these duties is to put in place a stack for returned books and a line for reservations. The system's main goal is to arrange and retrieve volumes according to current events, making sure that patron interactions with the library are controlled in a systematic and user-friendly way.

Explanation of the Implemented Application and its Features

The following are the main components of the Library Management System:

- Recently Returned Book Stack: A book is added to the stack upon return. This makes it possible to browse or remove the most recently returned books with ease.
- Book Reservation Queue: The system keeps track of a book reservation queue. Those who want to reserve a book are put in line, and when a book becomes available, it is served to the first person in line.
- CRUD Operations: The system enables the addition of new books to the collection, the updating of book information, the deletion of books, and the examination of the status of reservations and returns that are now in progress.
- GUI Interface: To facilitate user engagement, the system provides a GUI. Users can interact with buttons to perform stack and queue operations such as returning books, reserving books, and managing the lists of returned and reserved books.

- Remove Specific Book Operations: Users can choose which books in the QListWidget should be removed from the reserved queue or the stack of returned books (instead of the top or front book).

4. Test Cases for the Chosen Application

To make sure the system works as intended, the following three test cases are provided:

- **Test Case 1:** Add and View Books That Have Been Returned
  - **Test Input:** Give back a copy of "To Kill a Mockingbird."
  - **Anticipated Outcome:** The book is placed in the pile of recently returned literature. It ought to be prominently displayed at the top of the list.
  - **Steps:** Click "Add" after entering the book title and selecting "Return Book." Make that the book is in the stack that was just returned.

- **Test Case 2: Reserve a Book and View Reservation Queue**
  - **Test input:** Make a reservation for the book "1984."
  - **Anticipated Result:** Both the user and the book are added to the reservation queue.
  - **Steps:** Click "Add" after entering the book title and selecting "Reserve Book." If this is the first reservation, make sure the user is at the front and the book is in the line.
- **Test Case 3: Remove a Specific Returned Book**
  - **Test Input:** Take "The Great Gatsby" out of the stack that was just returned.
  - **Anticipated Result:** After being taken out of the stack, the book is no longer displayed in the stack display.
  - **Procedure:** Click "Remove" after selecting the book from the list of returned books. Make sure the book isn't visible anymore.

5. Challenges Faced During Development

• **Integrating Stack and Queue Operations:** It was difficult to implement stack and queue operations while making sure the system updated the GUI dynamically. Careful monitoring of the stack and queue's current states was necessary to guarantee that the appropriate book was taken out or served in response to user interaction.

• **GUI Design**: It was difficult to create an intuitive and user-friendly interface, particularly when it came to handling the lists' dynamic updates (recently returned books and reservations) and making sure users could quickly delete particular items from them.

• **Event Handling:** It was difficult to handle GUI events correctly for operations like button clicks and item selections. A seamless experience required that the system react to user activities in real-time.

6. Roles of Each Member and Their Contributions

• **Developer 1 (Reyes):** In charge of the application's general architecture, which includes designing the queue and stack data structures and integrating them with the graphical user interface. The basic logic for adding, deleting, and serving books was put into practice.

• **Designer (Cueto):** Designed the application's user interface and made sure it was easy to use. Qt was used to create the GUI's layout and incorporate the components required for user interaction (QPushButton, QListWidget, and QLineEdit).

• **Tester/Documentation (Espiritu):** in charge of creating the documentation, testing the application, and making sure the features that were developed worked as intended. Test cases were created and run to guarantee the system's dependability.

Source Code :

https://drive.google.com/drive/folders/1mmsgJ504DPe9keMkgyTBSt3xqxriWxXf?usp=sharing