**Efficient Library Book Borrowing System: Utilizing Queues and Stacks for Student ID, Email, and 3-Day Return Management**

**Proponents:**

Junio Layba

Jhuro Reyes

Nathan Rodgriguez

Neo Medrano

**Chosen Application Theme**

A Library Management System that allows users (students) to borrow and return books, with tracking features based on their student ID and email.

**Rationale for your choice**

Rationale for your choice

The proponents have chosen the Library Management System as their main theme and implementation because it effectively demonstrates and utilizes stacks and queues in a practical, real-world scenario. Initially, the team planned to develop an Ordering System as the first project. However, due to time constraints and the complexity involved—especially the need for more widgets, button navigation, and significantly more lines of code—we decided to abandon it. The Library Management System was chosen as a more manageable alternative, while still offering a valuable opportunity to showcase the functionality of stacks and queues. This application is a common use case in educational institutions, making it familiar to users. The tasks of borrowing books, tracking due dates, and managing borrowed books can be efficiently handled using stacks and queues, providing a simpler yet functional way to visualize and apply these concepts.

**Explanation of the Implemented Application and Its Features**

Our application simulates a Library Management System, which allows users (students) to borrow and return books with the following core features**:**

1. **Stack Operations:**
   * **Push:** Adds a recently borrowed book to the stack (for tracking the history of borrowed books).
   * **Pop:** Removes the most recently borrowed book from the stack (for return tracking).
   * **Peek:** Views the most recently borrowed book without removing it from the stack.
2. **Queue Operations:**
   * **Enqueue:** Adds a book to the waiting list for borrowing.
   * **Dequeue:** Removes the first book from the waiting list and marks it as borrowed.
   * **Peek:** Views the next book in the queue without removing it.
3. **Graphical User Interface (GUI):**
   * Buttons for Each Operation: Users can interact with the application through buttons for push, pop, enqueue, and dequeue operations.
   * Display Current States: The GUI shows the current states of the stack (borrowed books) and the queue (books in the borrowing list), allowing users to see all borrowed books and upcoming borrowed books in an organized manner.

**2. Queue Operations for Library Management System:**

1. **Enqueue (Add to Queue):**
   * Adds a **borrow request** (e.g., a book borrowed by a student) to the **queue**.
   * Ensures the borrow request is processed in **First-In-First-Out (FIFO)** order.

**Example Scenario:**

* + A student borrows a book, and their request is added to the queue.
  + The queue ensures that the first request added will be processed first.

1. **Dequeue (Process Borrow Request):**
   * Removes the next **borrow request** from the queue and processes it.
   * Updates the system to show that the book is borrowed by the corresponding student.

**Example Scenario:**

* + The first borrow request in the queue is processed (e.g., the book is marked as borrowed, and the student's due date is updated).
  + The processed request is removed from the queue.

1. **Peek (View Next Borrow Request):**
   * Views the next borrow request in the queue without removing it.
   * Allows the librarian to see which request will be processed next.

**Example Scenario:**

* + The librarian checks the next borrow request to prepare for the transaction (e.g., fetching the book).

3. Graphical User Interface (GUI):

- Buttons for Each Operation: Users can interact with the application using buttons for push, pop, enqueue, and dequeue operations.

- Display Current States: The GUI displays the current states of the stack and queue, allowing users to see the songs in the stack and the playlist queue.

**Three Test Cases for the Chosen Application**

**Challenges Faced During Development**

**Development:**

1. **Overdue Notifications:** The development initially considered implementing a feature where if the 3-day borrowing period expires, an email notification would be sent to the student's email, reminding them of the overdue book. This feature was abandoned due to challenges with integrating emailnotification systems within the QT environment, as well as tracking the history of overdue books effectively.
2. **Handling Null Pointers**: Ensuring that library data, such as books and users, were not null before accessing them to prevent crashes.
3. **Thread Safety:** Ensuring that interactions with the GUI were done safely on the main thread to avoid issues related to concurrency and GUI updates.
4. **Debugging:** Debugging issues related to the queue and stack management, such as ensuring books are added and removed correctly and efficiently.

**Design:**

1. **Color Palette:** The decision on a color palette was rushed due to project time constraints. As a result, the color choices were made quickly to meet the deadline, without fully exploring options for a more professional and user-friendly design.
2. **QT CSS Override:** Initially, the design was intended to be done through the drag-and-drop UI menu, but to meet design requirements, most elements were custom-coded in mainwindow.cpp.
3. **Static Built-in Buttons:** The static nature of QT buttons limited some design choices. This was addressed by opting for a simple and effective UI layout using built-in styles.

more libraries being access liked qpaint and qpixmap, but it was decided to avoid such methos for the sake of time efficiency.

**Roles of Each Member and Their Contributions**

Developer: Neo Medrano/Junio Layba

-Implemented the core functionality of the application, including stack and queue operations, and integrated the GUI components.

Designer: Nathan Rodgriguez

-Designed the graphical user interface, ensuring that it was user-friendly and visually appealing.

Tester: Jhuro Reyes

Conducted thorough testing of the application, identified bugs, and provided feedback for improvements.