System Features

RMI Implementation

The system must be able to establish communication between it's server and it's clients. Java's native JRMP (RMI) API is a valid possibility as it is natively supported and fulfills the requirements of this project.

The best option would be to create two separate service managers, one for the server and one for the clients. These would serve as the central access points of available services and also manage their connections. This would remove the need to clutter the code with a lot of copy paste code each time a service is required.

Authentication Roles

For the outer layer of the multi user implementation, there should be some sort of a permission system. One employee may have access to more features than others, for example a rental terminal that is used by people renting books, should not have access to any part of the management part of the system. Also an administrator may have more access than a regular employee.

Examples of permission types.

- **Roles/Groups**: Create different role types like 'delete', 'add', 'review' etc. Each user can then be added to one or more of these roles depending on how much access they should have.
- **Level Based**: Create different levels of access like 'administrator', 'employee', 'rental' etc. In this case a user is assigned a single level and depending on how high that level is, decides how much access the user should have.

Roles are a bit more complex, but allows much more flexibility and is the preferred option in most systems.

Authentication also needs to be library based. In theory, an employee might work at two libraries with different access while administrators have access to all of them.

Multi User

The system requires two layers of multi user implementation. There is the outer layer that would be used to login a single client machine, like a work station for a library employee or a terminal for book rentals. Then there is the inner layer which would be used to validate a user renting a book, which is done on a terminal that would require a connection to the server via the outer layer of authentication.

Possible Solution

1. A database table containing a user name/id and a user token.

2. Any time some sort of user needs to input user name/id and their password, token is created and checked via a authentication service on the server.

3. The authentication roles can be used to check whether a user is allowed to login to the system, or is only allowed to rent books from it.

This solution requires only one layer of user control that can be used by both users renting books and by employees that needs to login to the management part of the system. It also allows employees to rent books themselves without requiring an additional account to be created for it