Inter-Thread Communication

Project Requirement Analysis

- **Introduction**

In a multi-threaded application, it is important to ensure that threads communicate and synchronize their actions to prevent data races, deadlocks, and other synchronization-related problems. In this project, we will implement a banking system that uses inter-thread communication to synchronize the actions of multiple threads.

- **Project Scope**

The scope of the project includes the implementation of a banking system that consists of three types of accounts: savings, current, and NRI (Non-Resident Indian) accounts. Each account can perform deposit and withdrawal operations, and the system should enforce certain constraints such as overdraft limits, negative balance limits, and interest rates. The system should use inter-thread communication to ensure that the operations on each account are synchronized and safe.

- **Requirements**

Functional Requirements

1. The system should support three types of accounts: savings, current, and NRI accounts.
2. Each account should have the following operations: deposit and withdrawal.
3. The system should enforce an overdraft limit of 500 for savings accounts.
4. The system should enforce a negative balance limit of -500 for current accounts.
5. The system should allow NRI accounts to have negative balances.
6. The system should allow the interest rate for NRI accounts to be set by the user.
7. Support Currency Exchanger
8. The system should use inter-thread communication to ensure that the operations on each account are synchronized and safe.
9. The system should generate notifications for each transaction (deposit and withdrawal) on each account.

Non-functional Requirements

1. The system should be implemented in C++.
2. The system should use mutexes and condition variables for inter-thread communication.
3. The system should be thread-safe and free from data races, deadlocks, and other synchronization-related problems.

4. The system should use object-oriented design principles and good programming practices.
5. The system should have a user-friendly command-line interface.

Use Cases

1. Deposit money into a savings account.
2. Option's for Currency Exchange.
3. Withdraw money from a savings account.
4. Deposit money into a current account.
5. Withdraw money from a current account.
6. Deposit money into an NRI account.
7. Withdraw money from an NRI account.
8. Set the interest rate for an NRI account.
9. Constraints and Assumptions
10. The system assumes that each account belongs to a single user.
11. The system assumes that there are no network delays or failures during inter-thread communication.
12. The system assumes that there are no hardware failures during the execution of the program.

- **Error Handling**

The system should be equipped with error handling mechanisms to ensure that errors in communication between threads are detected and resolved in a timely manner.

- **Conclusion**

The implementation of a banking system that uses inter-thread communication is essential to ensure the safe and synchronized execution of multiple threads in a multi-threaded application. The project requirements analysis provides a clear understanding of the functional and non-functional requirements, use cases, constraints, and assumptions that will guide the implementation of the banking system.