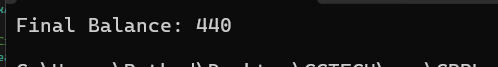
**Problem Statement**

**Bank Account Simulation with Multiple Users Accessing the Same Account:**

simulating multiple users accessing a shared bank account concurrently. Each user can either deposit or withdraw money from the account. To ensure data integrity, especially when multiple users (threads) access the account at the same time, proper synchronization must be applied.



### Key Concepts Learned:

1. **Multithreading in C++**: Using threads to run multiple tasks simultaneously.
2. **Race Conditions**: When two threads attempt to modify the same resource at the same time, leading to inconsistent data. This is avoided by using a mutex.
3. **Mutex**: A synchronization primitive that ensures only one thread accesses a critical section of code at a time.
4. **Thread-safe Functions**: By using std::lock\_guard, we ensured the critical sections in the deposit and withdraw methods were safe from concurrent access.