Name: Atharva Muley
Campus Id: GJ20497

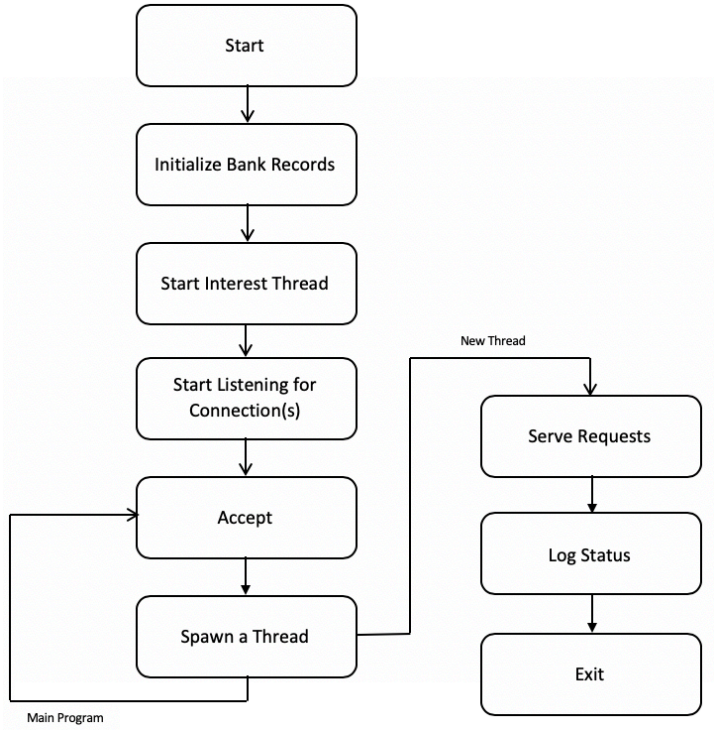
AOS Project 1

Server

Execution Flow:

1. The server reads Records.txt and initializes Bank Accounts it its memory.
2. The server then starts interestDeamon which posts interest into all of the client accounts.
3. Server then creates a socket, bind an address and start listening on the socket.
4. Server waits for clients to connect.
5. Once the client is connected then the server spawns a Thread and passes the clientfd as a parameter to the thread.
6. The server then continues to listen for other connections.
7. Meanwhile, threads created manages the transaction posted by the client and serve to those requests.
8. The thread then checks the transaction type and performs function accordingly.
9. The thread then replies back with the transaction status.
10. Server logs the transaction status on the console.

Activity Diagram



Client

Execution Flow:

1. The client reads the transaction from the Transactions.txt file, line by line.
2. For each transaction the client tries to connect with the sever and sends the transaction and waits for the server's reply.
3. On receiving the server's reply the client logs the data into the Logs-SingleThread.txt and on the console as well.

Client Multithread

Execution Flow:

1. The client reads the transactions from the Transactions.txt file, line by line.
2. For each line client spawns a Thread and tries to connect to the server.
3. Upon successful connection, the thread sends the transaction to the server and waits for the server's reply.
4. On receiving the server's reply the client logs the data into the Logs-Multithread.txt and on the console as well.

How to compile and run the code.

Steps:

1. To Compile the code, navigate into **src/** directory
2. Type **make** or **make compile** to compile the code
3. To run the server type **./server**
4. To run the client open another terminal and type .**/client** for single threaded client or type **./client-multithread** for multithreaded client.
5. Alternatingly, you can also run **./client-spawn.sh** to spawn multiple single thread clients at the same time. Clients here are single threaded.

Design Tradeoffs made

- There are two types of Client programs available i.e., Client and Client-Multithread. The Client program sends one transaction at a time and doesn't provide simultaneous connection simultaneously, whereas the Client-Multithread sends concurrent request to the server.
- The server uses List data structure to store the account information of the customers. List data structure provides flexibility but if the list size grows the traversals can be bit slow.

Multiple Threads and Synchronization

Every client is spawned a Thread. The thread performs the functionality mentioned in the **worker** function in the sever. Every account has its own mutex and conditional variable associated with it. Whenever a thread wants to operate on an account it needs to acquire lock of that specific account before performing the operations else the thread waits till the lock is acquired. Providing mutex for each account make concurrency possible.