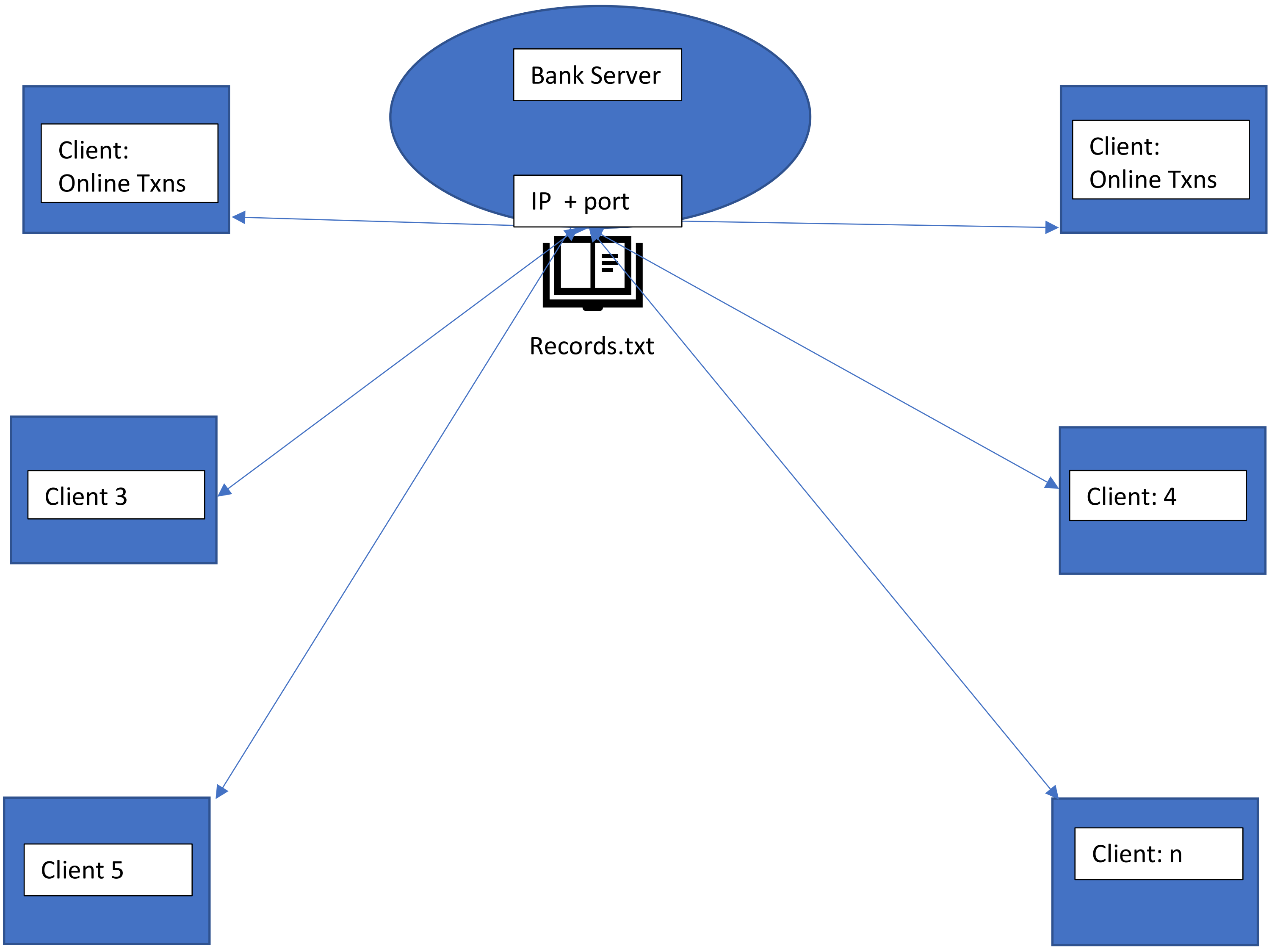
Centralized Multi-User Concurrent Bank Account Manager

Architecture:



Terminology:

Server – A server is a program that provides functionality for other programs or devices, called "clients"

In this project Sever Acts a Centralized Multi-User Bank account manager which performs concurrent transactions at a time safely. All the data is inserted into the bank using **Records.txt** file. As server is concurrent it can accept multiple simultaneous requests from multiple clients. Clients can perform Deposit or Withdraw transactions. Also, server before performing the transactions validates the amount (whether the bank account has enough funds for processing). Also, the server is multi-threaded. For every connection from client, a new thread will invoke and will perform the transaction. In addition, Server periodically adds the interest to every bank account.

Client - A client is a piece of hardware or software that accesses a service made available by a server.

In this project, client can be an ATM, or Online bank portal which will access the server for performing the transactions. Client will read the data from Transactions.txt file and will send the data to server using TCP protocol (Sockets). All the requests from the client are sent to server periodically based on the timestamp mentioned in the Transactions.txt file. Also, whenever client and server interact, the corresponding transaction output will be displayed on the console.

Connection between Client and Server using Sockets:

Server system calls

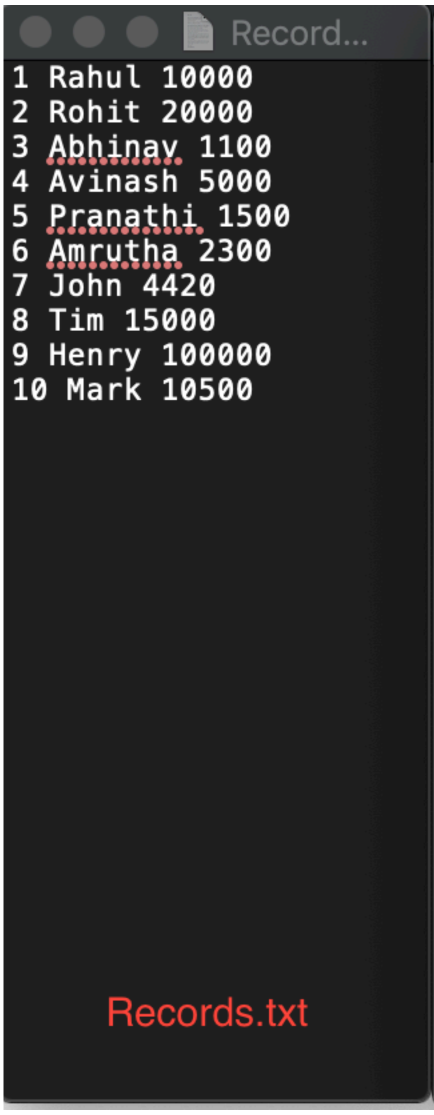
1. Socket
2. Bind
3. Listen
4. Accept
5. Send/Recv

Client System Calls

1. Socket
2. Bind
3. Connect
4. Send/Recv

Working: The server will initially create a socket and will be listening for clients continuously. Whenever a new client connects, Server accepts the client and will assign a new thread to the request sent by the client. Here, the communication between client and server uses stream sockets. Client reads the transactions from transaction.txt file line by line and then will send it to server. Server will process the transaction based on the type of transaction and after validating the bank account and amount.

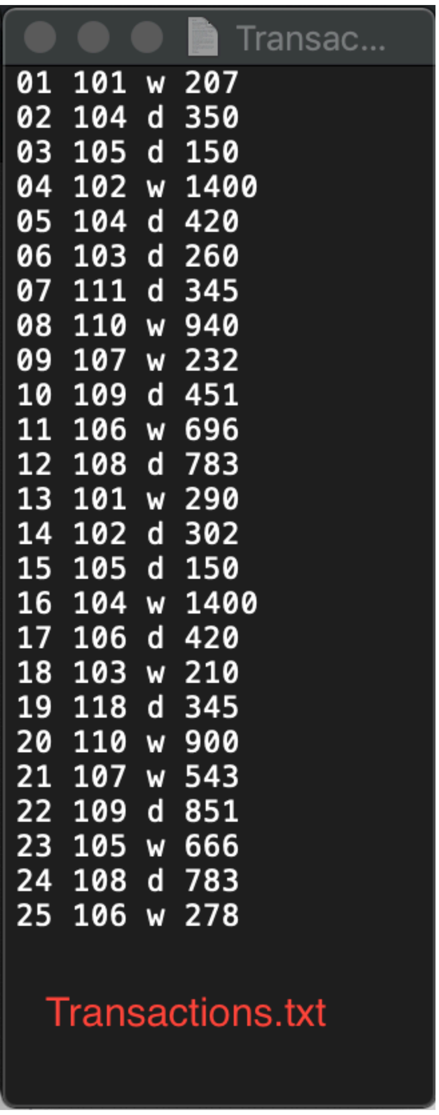
Records.txt



```
1 Rahul 10000
2 Rohit 20000
3 Abhinav 1100
4 Avinash 5000
5 Pranathi 1500
6 Amrutha 2300
7 John 4420
8 Tim 15000
9 Henry 100000
10 Mark 10500
```

Records.txt

Transactions.txt



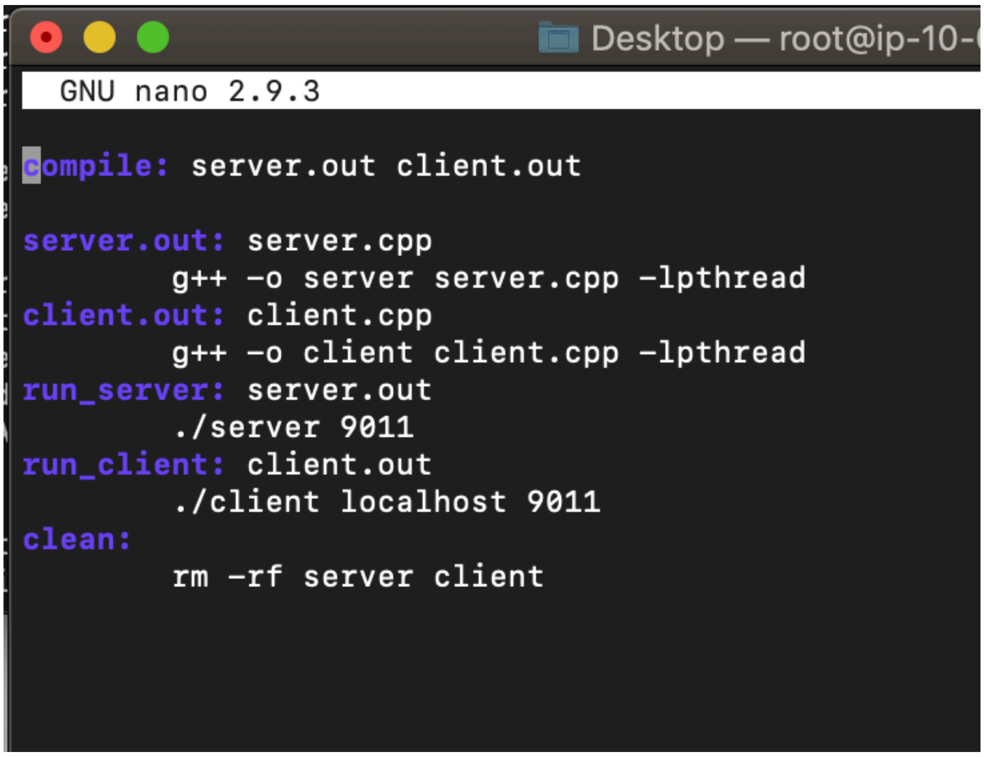
```
01 101 w 207
02 104 d 350
03 105 d 150
04 102 w 1400
05 104 d 420
06 103 d 260
07 111 d 345
08 110 w 940
09 107 w 232
10 109 d 451
11 106 w 696
12 108 d 783
13 101 w 290
14 102 d 302
15 105 d 150
16 104 w 1400
17 106 d 420
18 103 w 210
19 118 d 345
20 110 w 900
21 107 w 543
22 109 d 851
23 105 w 666
24 108 d 783
25 106 w 278
```

Transactions.txt

Execution of the Project:

Make Utility: using Make utility, we can run and compile programs more efficiently.

Makefile is used in this project with which we can compile, run and clean the programs.



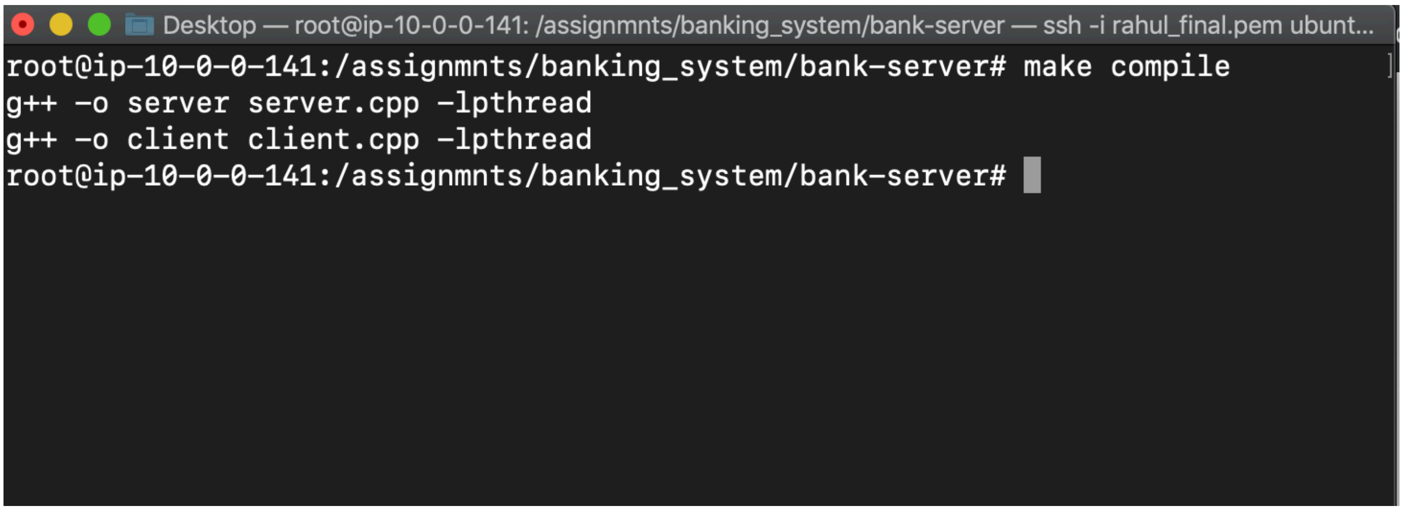
```
GNU nano 2.9.3

compile: server.out client.out

server.out: server.cpp
        g++ -o server server.cpp -lpthread
client.out: client.cpp
        g++ -o client client.cpp -lpthread
run_server: server.out
        ./server 9011
run_client: client.out
        ./client localhost 9011
clean:
        rm -rf server client
```

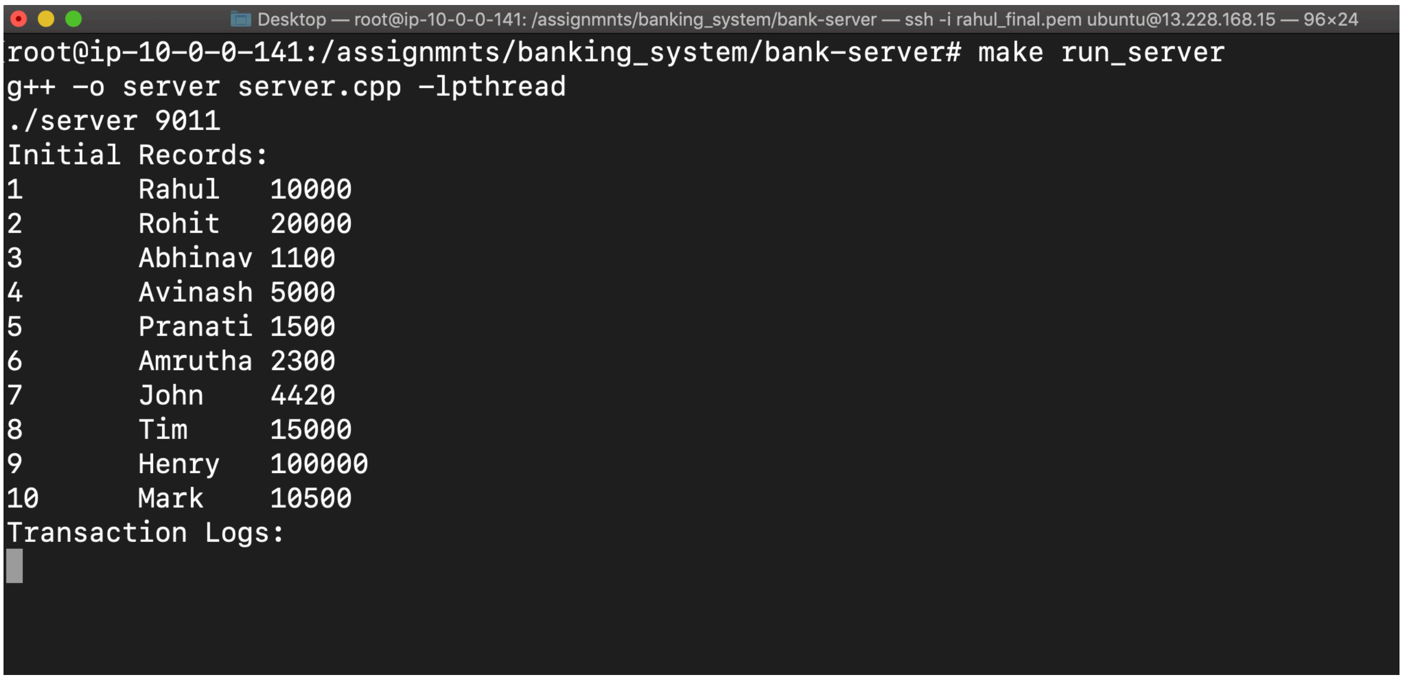
Go to the project directory and run

make compile: This will compile two files server.cpp and client.cpp which will generate two executables client and server.


Desktop — root@ip-10-0-0-141: /assignmnts/banking_system/bank-server — ssh -i rahul_final.pem ubunt...

```
root@ip-10-0-0-141:/assignmnts/banking_system/bank-server# make compile
g++ -o server server.cpp -lpthread
g++ -o client client.cpp -lpthread
root@ip-10-0-0-141:/assignmnts/banking_system/bank-server# 
```

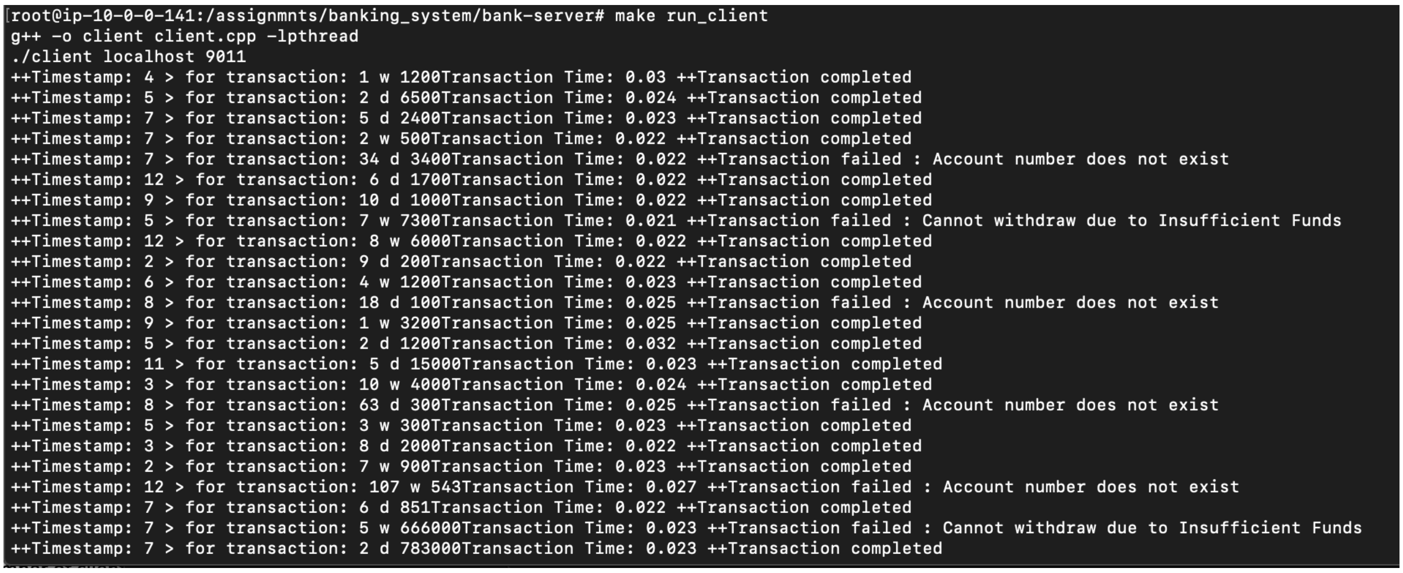
make run_server: This will start the server and begins execution



Desktop — root@ip-10-0-0-141: /assignmnts/banking_system/bank-server — ssh -i rahul_final.pem ubuntu@13.228.168.15 — 96×24

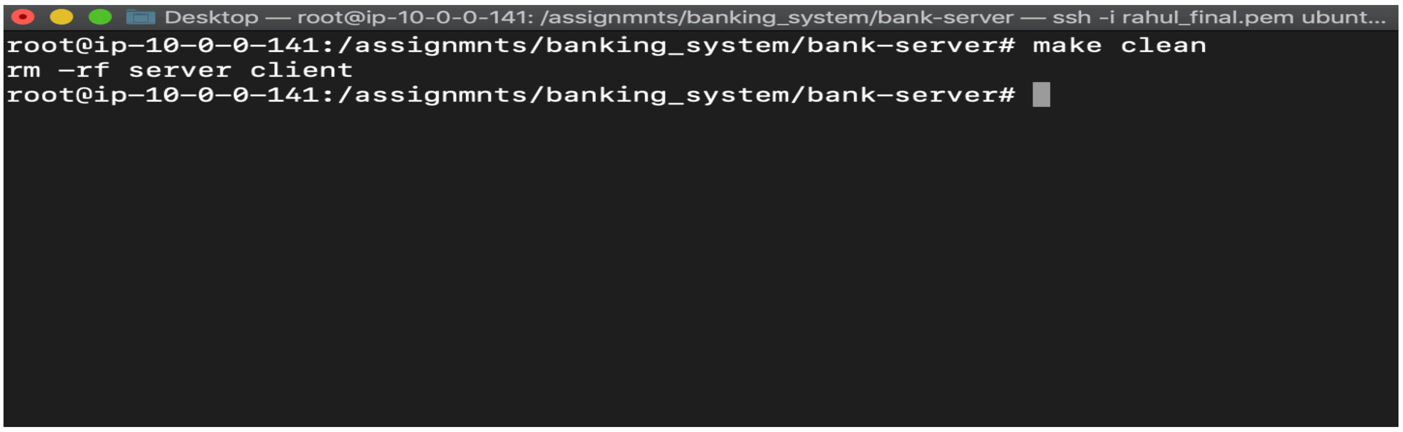
```
root@ip-10-0-0-141:/assignmnts/banking_system/bank-server# make run_server
g++ -o server server.cpp -lpthread
./server 9011
Initial Records:
1       Rahul   10000
2       Rohit   20000
3       Abhinav 1100
4       Avinash 5000
5       Pranati 1500
6       Amrutha 2300
7       John    4420
8       Tim     15000
9       Henry   100000
10      Mark    10500
Transaction Logs:
```

make run_client (on a new terminal): This will start the client and begins execution



```
root@ip-10-0-0-141:/assignmnts/banking_system/bank-server# make run_client
g++ -o client client.cpp -lpthread
./client localhost 9011
++Timestamp: 4 > for transaction: 1 w 1200Transaction Time: 0.03 ++Transaction completed
++Timestamp: 5 > for transaction: 2 d 6500Transaction Time: 0.024 ++Transaction completed
++Timestamp: 7 > for transaction: 5 d 2400Transaction Time: 0.023 ++Transaction completed
++Timestamp: 7 > for transaction: 2 w 500Transaction Time: 0.022 ++Transaction completed
++Timestamp: 7 > for transaction: 34 d 3400Transaction Time: 0.022 ++Transaction failed : Account number does not exist
++Timestamp: 12 > for transaction: 6 d 1700Transaction Time: 0.022 ++Transaction completed
++Timestamp: 9 > for transaction: 10 d 1000Transaction Time: 0.022 ++Transaction completed
++Timestamp: 5 > for transaction: 7 w 7300Transaction Time: 0.021 ++Transaction failed : Cannot withdraw due to Insufficient Funds
++Timestamp: 12 > for transaction: 8 w 6000Transaction Time: 0.022 ++Transaction completed
++Timestamp: 2 > for transaction: 9 d 200Transaction Time: 0.022 ++Transaction completed
++Timestamp: 6 > for transaction: 4 w 1200Transaction Time: 0.023 ++Transaction completed
++Timestamp: 8 > for transaction: 18 d 100Transaction Time: 0.025 ++Transaction failed : Account number does not exist
++Timestamp: 9 > for transaction: 1 w 3200Transaction Time: 0.025 ++Transaction completed
++Timestamp: 5 > for transaction: 2 d 1200Transaction Time: 0.032 ++Transaction completed
++Timestamp: 11 > for transaction: 5 d 15000Transaction Time: 0.023 ++Transaction completed
++Timestamp: 3 > for transaction: 10 w 4000Transaction Time: 0.024 ++Transaction completed
++Timestamp: 8 > for transaction: 63 d 300Transaction Time: 0.025 ++Transaction failed : Account number does not exist
++Timestamp: 5 > for transaction: 3 w 300Transaction Time: 0.023 ++Transaction completed
++Timestamp: 3 > for transaction: 8 d 2000Transaction Time: 0.022 ++Transaction completed
++Timestamp: 2 > for transaction: 7 w 900Transaction Time: 0.023 ++Transaction completed
++Timestamp: 12 > for transaction: 107 w 543Transaction Time: 0.027 ++Transaction failed : Account number does not exist
++Timestamp: 7 > for transaction: 6 d 851Transaction Time: 0.022 ++Transaction completed
++Timestamp: 7 > for transaction: 5 w 666000Transaction Time: 0.023 ++Transaction failed : Cannot withdraw due to Insufficient Funds
++Timestamp: 7 > for transaction: 2 d 783000Transaction Time: 0.023 ++Transaction completed
```

make clean: This will delete the client and server executables



Desktop — root@ip-10-0-0-141: /assignmnts/banking_system/bank-server — ssh -i rahul_final.pem ubunt...

```
root@ip-10-0-0-141:/assignmnts/banking_system/bank-server# make clean
rm -rf server client
root@ip-10-0-0-141:/assignmnts/banking_system/bank-server#
```

Design Considerations:

This project is implemented using multi-threading to handle concurrent transactions. Server will run in an infinite loop checking for client connections and for every connection it will create a new thread. Also, Mutex locks are implemented in this project. So, every time only one thread will be reading/writing the transactions file. In addition, stream socket was used for communication between client and server to take advantage of TCP protocol.

Tradeoffs:

Bank accounts should be added in Records.txt file only.

No Authentication Mechanism for clients.

Correctness: case – 1

The server allows to perform the transactions after validating the account number and current balance. Following cases are handled in the project

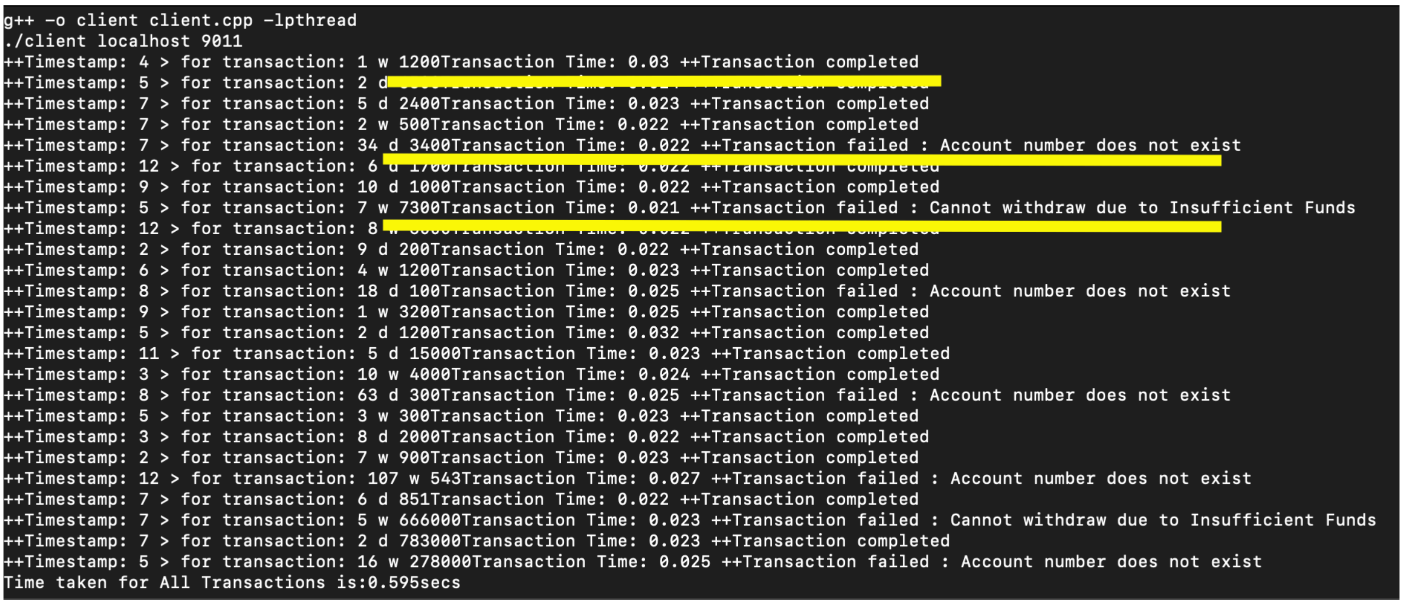
1: Valid Account Number and has enough Balance: Successful Transaction

2: Not a valid account number: Failure: Invalid Account Number

3: Valid Account Number and not enough balance: Failure: Insufficient Funds

All the above cases are highlighted in the screenshot below

for below screenshot records.txt and transactions.txt have been used along with same values which are displayed above.

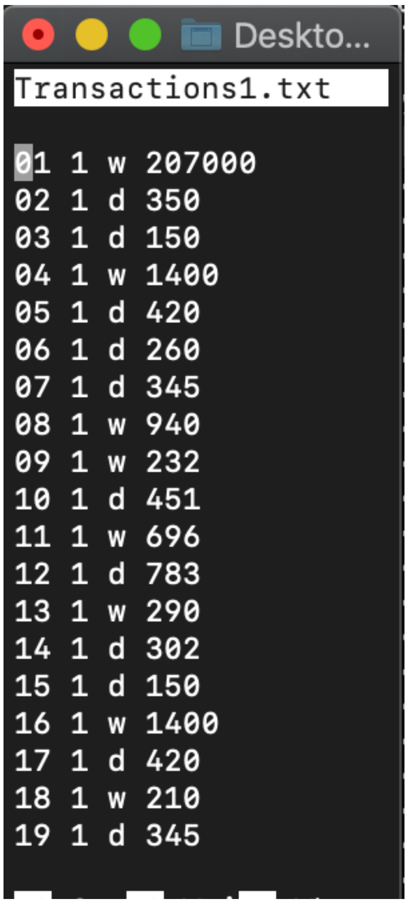


```
g++ -o client client.cpp -lpthread
./client localhost 9011
++Timestamp: 4 > for transaction: 1 w 1200Transaction Time: 0.03 ++Transaction completed
++Timestamp: 5 > for transaction: 2 d [illegible]Transaction Time: 0.02[illegible] ++Transaction completed
++Timestamp: 7 > for transaction: 5 d 2400Transaction Time: 0.023 ++Transaction completed
++Timestamp: 7 > for transaction: 2 w 500Transaction Time: 0.022 ++Transaction completed
++Timestamp: 7 > for transaction: 34 d 3400Transaction Time: 0.022 ++Transaction failed : Account number does not exist
++Timestamp: 12 > for transaction: 6 d 1700Transaction Time: 0.022 ++Transaction completed
++Timestamp: 9 > for transaction: 10 d 1000Transaction Time: 0.022 ++Transaction completed
++Timestamp: 5 > for transaction: 7 w 7300Transaction Time: 0.021 ++Transaction failed : Cannot withdraw due to Insufficient Funds
++Timestamp: 12 > for transaction: 8 w [illegible]Transaction Time: 0.022 ++Transaction completed
++Timestamp: 2 > for transaction: 9 d 200Transaction Time: 0.022 ++Transaction completed
++Timestamp: 6 > for transaction: 4 w 1200Transaction Time: 0.023 ++Transaction completed
++Timestamp: 8 > for transaction: 18 d 100Transaction Time: 0.025 ++Transaction failed : Account number does not exist
++Timestamp: 9 > for transaction: 1 w 3200Transaction Time: 0.025 ++Transaction completed
++Timestamp: 5 > for transaction: 2 d 1200Transaction Time: 0.032 ++Transaction completed
++Timestamp: 11 > for transaction: 5 d 15000Transaction Time: 0.023 ++Transaction completed
++Timestamp: 3 > for transaction: 10 w 4000Transaction Time: 0.024 ++Transaction completed
++Timestamp: 8 > for transaction: 63 d 300Transaction Time: 0.025 ++Transaction failed : Account number does not exist
++Timestamp: 5 > for transaction: 3 w 300Transaction Time: 0.023 ++Transaction completed
++Timestamp: 3 > for transaction: 8 d 2000Transaction Time: 0.022 ++Transaction completed
++Timestamp: 2 > for transaction: 7 w 900Transaction Time: 0.023 ++Transaction completed
++Timestamp: 12 > for transaction: 107 w 543Transaction Time: 0.027 ++Transaction failed : Account number does not exist
++Timestamp: 7 > for transaction: 6 d 851Transaction Time: 0.022 ++Transaction completed
++Timestamp: 7 > for transaction: 5 w 666000Transaction Time: 0.023 ++Transaction failed : Cannot withdraw due to Insufficient Funds
++Timestamp: 7 > for transaction: 2 d 783000Transaction Time: 0.023 ++Transaction completed
++Timestamp: 5 > for transaction: 16 w 278000Transaction Time: 0.025 ++Transaction failed : Account number does not exist
Time taken for All Transactions is:0.595secs
```

Correctness: case – 2

Server will not allow simultaneous access to the same account from multiple clients at same time. It’s mainly because of implementing locks for every thread while it’s accessing a bank account

For this use case I've considered two clients with same transactions.txt file as displayed below



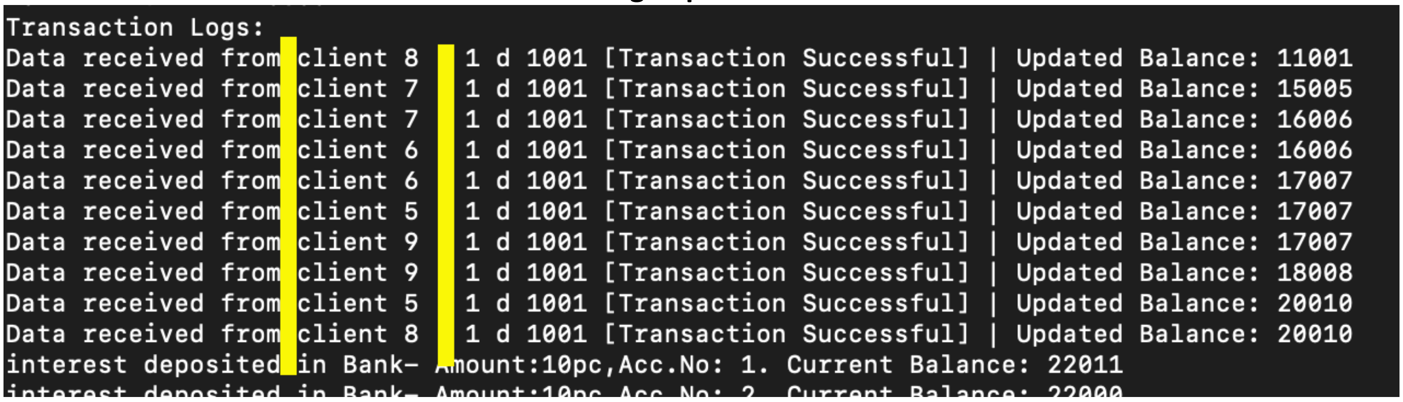
```
Deskto...
Transactions1.txt

01 1 w 207000
02 1 d 350
03 1 d 150
04 1 w 1400
05 1 d 420
06 1 d 260
07 1 d 345
08 1 w 940
09 1 w 232
10 1 d 451
11 1 w 696
12 1 d 783
13 1 w 290
14 1 d 302
15 1 d 150
16 1 w 1400
17 1 d 420
18 1 w 210
19 1 d 345
```

Performed all the transactions on same bank account number which is 1. Tested the program with 2 clients.

Result: All the transactions are performed without corrupting the data when connected to multiple clients as we can see in the below diagram.

Data Received from clients 5,6,7,8 and 9 logs are being displayed. In addition, on the bottom of screenshot we can see that Interest is also being deposited.ß



```
Transaction Logs:
Data received from client 8   1 d 1001 [Transaction Successful] | Updated Balance: 11001
Data received from client 7   1 d 1001 [Transaction Successful] | Updated Balance: 15005
Data received from client 7   1 d 1001 [Transaction Successful] | Updated Balance: 16006
Data received from client 6   1 d 1001 [Transaction Successful] | Updated Balance: 16006
Data received from client 6   1 d 1001 [Transaction Successful] | Updated Balance: 17007
Data received from client 5   1 d 1001 [Transaction Successful] | Updated Balance: 17007
Data received from client 9   1 d 1001 [Transaction Successful] | Updated Balance: 17007
Data received from client 9   1 d 1001 [Transaction Successful] | Updated Balance: 18008
Data received from client 5   1 d 1001 [Transaction Successful] | Updated Balance: 20010
Data received from client 8   1 d 1001 [Transaction Successful] | Updated Balance: 20010
interest deposited in Bank- Amount:10pc,Acc.No: 1. Current Balance: 22011
interest deposited in Bank- Amount:10pc,Acc.No: 2. Current Balance: 22000
```

***** END OF THE DOCUMENT*****