MULTITHREADED BANKING APPLICATION

PROJECT REPORT:

27/10/2022

Sunag R Kamath (200905006)
Gautham Prabhu M (200905124)
Swamiraju Satya Praveen Varma (200905044)
Yashwanth Paladugu(200905058)

Under the guidance of,

Mrs. Archana Praveen Kumar
Department of Computer Science Engineering
Manipal Institute of Technology

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

ACKNOWLEDGEMENT

Our team (Sunag R Kamath, Gautham Prabhu M, Swamiraju Satya Praveen Varma, Yashwanth) would like to thank the **Department of Computer Science Engineering, Manipal Institute of Technology** for providing us with the necessary infrastructure and support to carry out our project. We would also like to convey our deepest gratitude to **Mrs. Archana Praveen** for guiding us in our project, teaching us the fundamentals of how an Operating System works and supporting us all the way along.

We express our gratitude to **Mr. Praveen Pai** for guiding us during our lab hours.

Our deepest gratitude to the non-teaching staff of Manipal Institute of Technology.

We would also like to convey our deepest gratitude to our acquaintances for selflessly helping us.

INTRODUCTION:-

Our solution is an application for handling multiple accounts in a bank. In this project, we tried to show the working of a banking account system and cover the basic functionality of banking, particularly with a focus on achieving multi-client concurrency. This project is developed using C language.

Our project makes use of multiple important operating systems concepts which include but are not limited to multithreading, process synchronization, deadlock handling, mutex locks, concurrent client-server interactions, etc.

The core of our project stems from the fundamental idea of **multithreading**. Most attempts at a history of multithreaded processors start with the PPUs in the CDC 6600. However, the idea of sharing a single execution path among multiple threads of execution started at least a decade earlier. After its initial appearance in the 50s, and the subsequent investigation of IBM on the implementation of Simultaneous Multithreading(1968), multithreading has proven to be a powerful tool to handle multiple queries simultaneously from different clients.

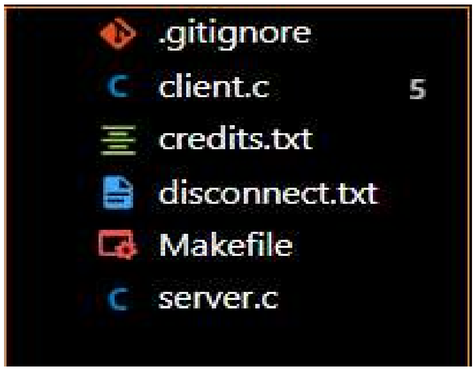
A significant issue faced by many real-world projects is that of different processes working parallelly without being synchronized appropriately. This can lead to major problems including but not limited to Deadlocks, inconsistency in data, the incorrect flow of data and instructions, etc. To combat the possible errors caused due to process synchronization we have utilized **semaphores, mutex locks,** etc.

Our prototype CLI application has the capacity to handle up to 20 concurrent clients in one session, the capacity can be ramped up as and when required. The code makes heavy use of threads in various aspects ranging from accepting sessions, handling client inputs, opening, and closing connections, etc.

The software has multiple functionalities, ranging from creating a user account during a session, possibly logging out and logging in, depositing and withdrawing money from an account to transferring money into another account. All these functionalities require background checks which range from checking the validity of customers, checking whether a given transaction can be processed, handling concurrent transactions, etc.

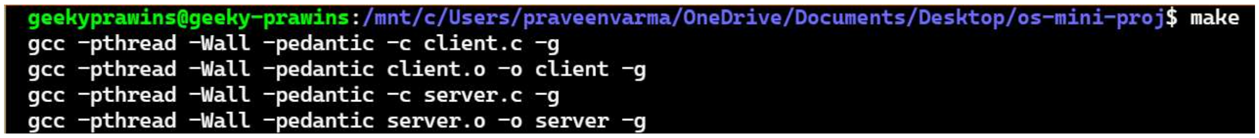
RESULTS AND OUTPUTS:-

Folder Structure -



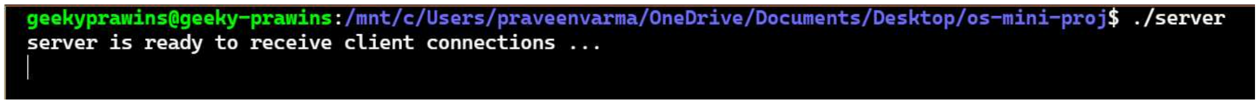
```
.gitignore
client.c        5
credits.txt
disconnect.txt
Makefile
server.c
```

Making the file and compiling them -



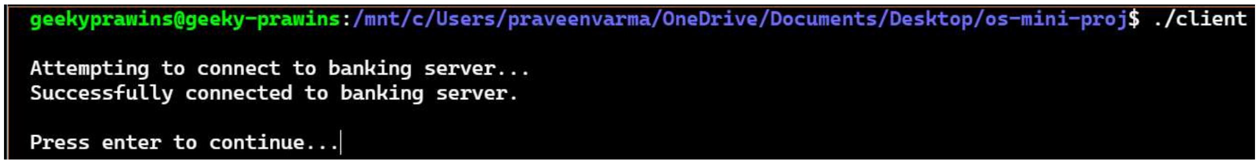
```
geekyprawins@geeky-prawins:/mnt/c/Users/praveenvarma/OneDrive/Documents/Desktop/os-mini-proj$ make
gcc -pthread -Wall -pedantic -c client.c -g
gcc -pthread -Wall -pedantic client.o -o client -g
gcc -pthread -Wall -pedantic -c server.c -g
gcc -pthread -Wall -pedantic server.o -o server -g
```

The server is ready to take in client connections -



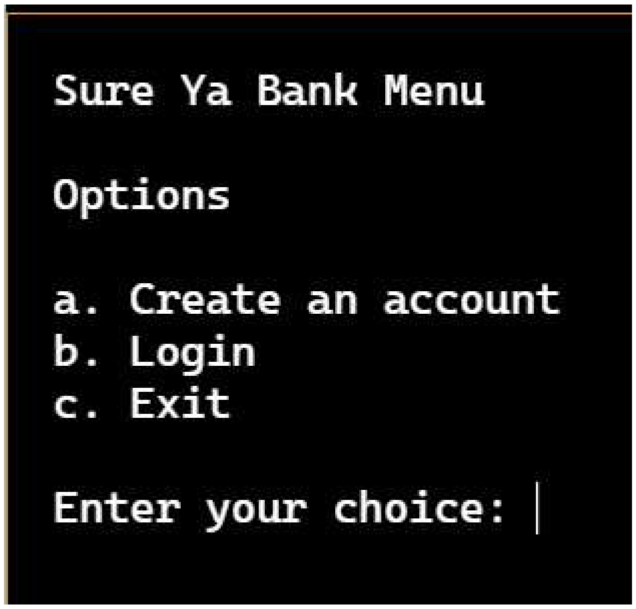
```
geekyprawins@geeky-prawins:/mnt/c/Users/praveenvarma/OneDrive/Documents/Desktop/os-mini-proj$ ./server
server is ready to receive client connections ...
```

A client attempting to connect to Server -



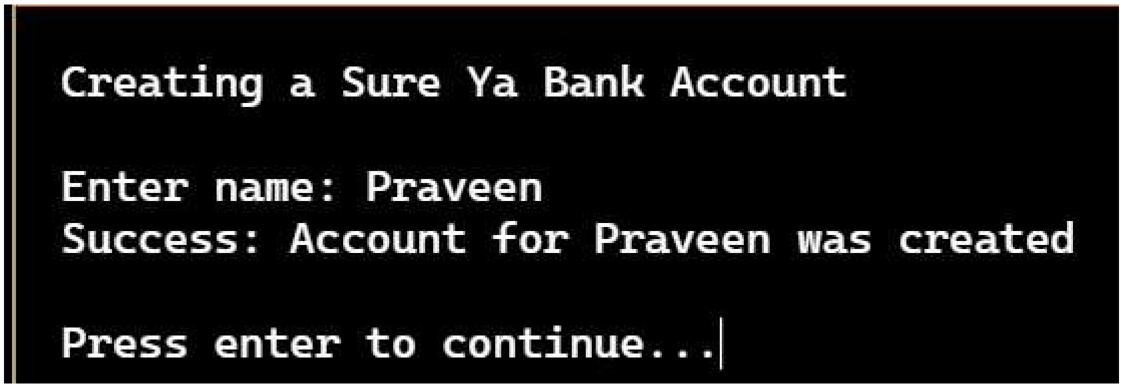
```
geekyprawins@geeky-prawins:/mnt/c/Users/praveenvarma/OneDrive/Documents/Desktop/os-mini-proj$ ./client

Attempting to connect to banking server...
Successfully connected to banking server.

Press enter to continue...
```

Client Menu program as soon as the program start -



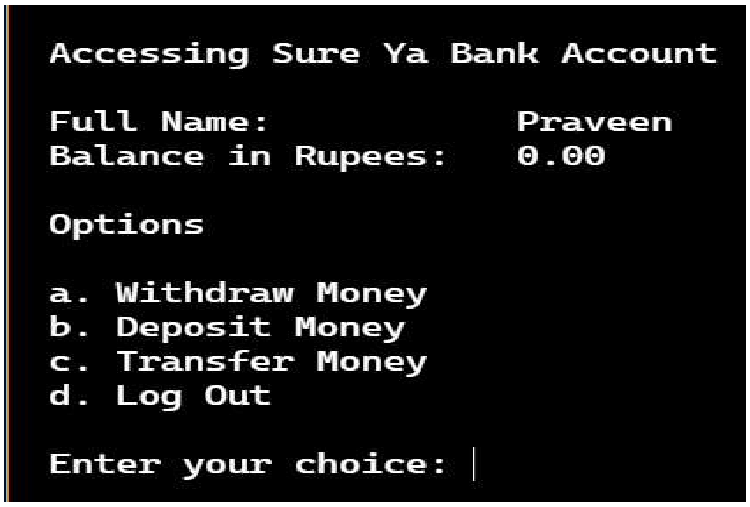
```
Sure Ya Bank Menu

Options

a. Create an account
b. Login
c. Exit

Enter your choice:
```

Creating an Account Successfully -



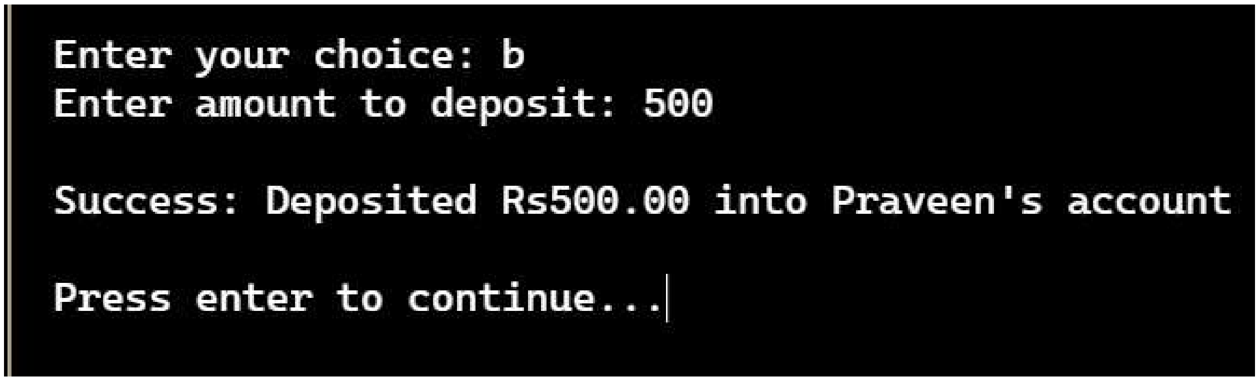
```
Creating a Sure Ya Bank Account

Enter name: Praveen
Success: Account for Praveen was created

Press enter to continue...
```

Login After Successful creation of account -



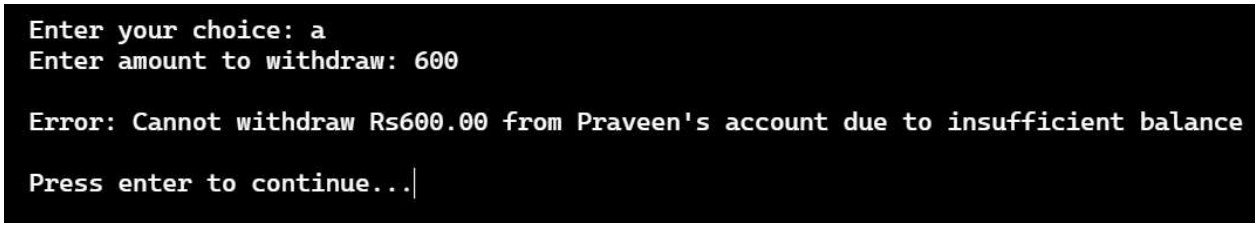
```
Accessing Sure Ya Bank Account

Full Name:           Praveen
Balance in Rupees:   0.00

Options

a. Withdraw Money
b. Deposit Money
c. Transfer Money
d. Log Out

Enter your choice:
```

Deposit Amount functionality:



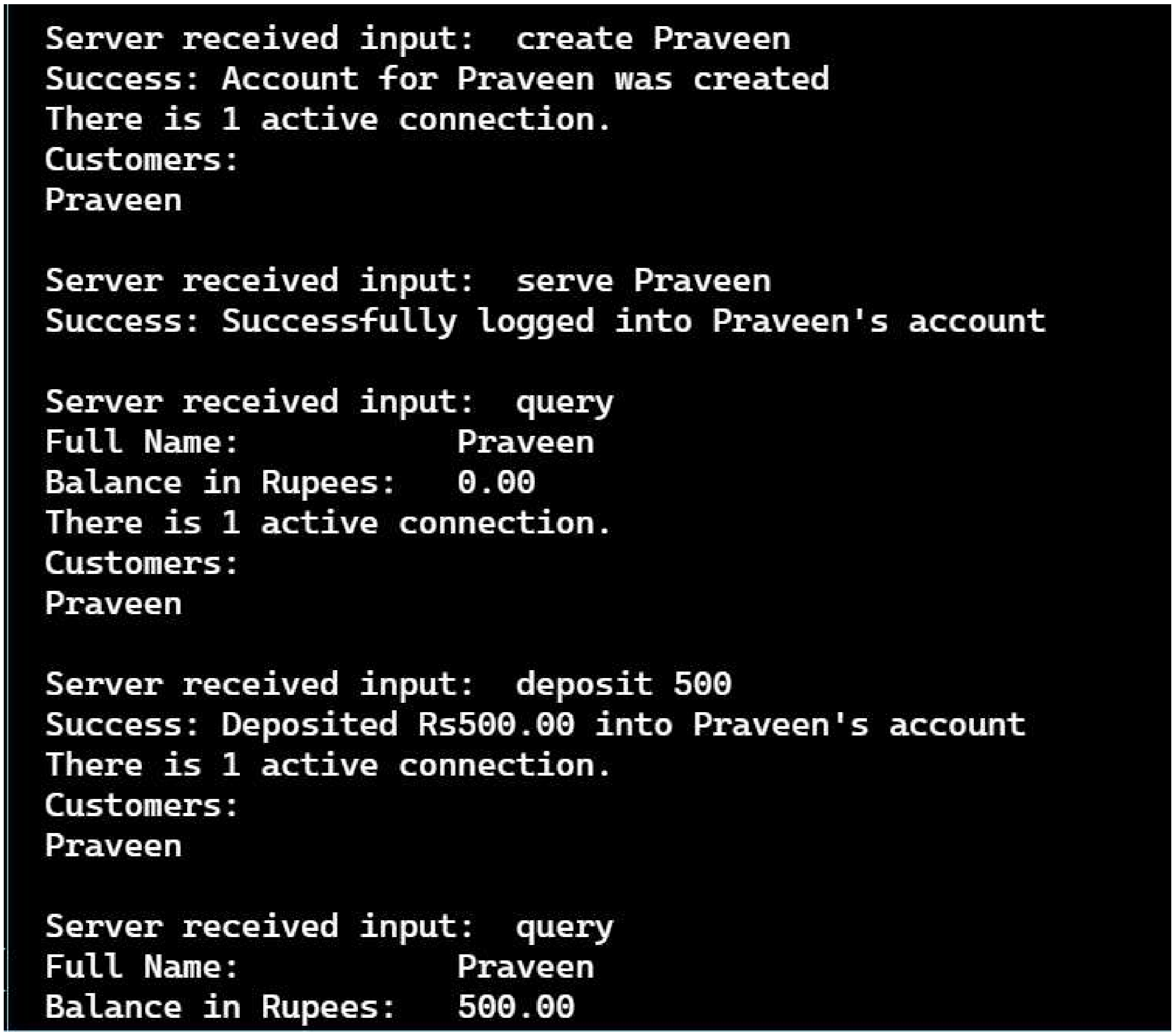
```
Enter your choice: b
Enter amount to deposit: 500

Success: Deposited Rs500.00 into Praveen's account

Press enter to continue...
```

Withdraw Amount functionality:



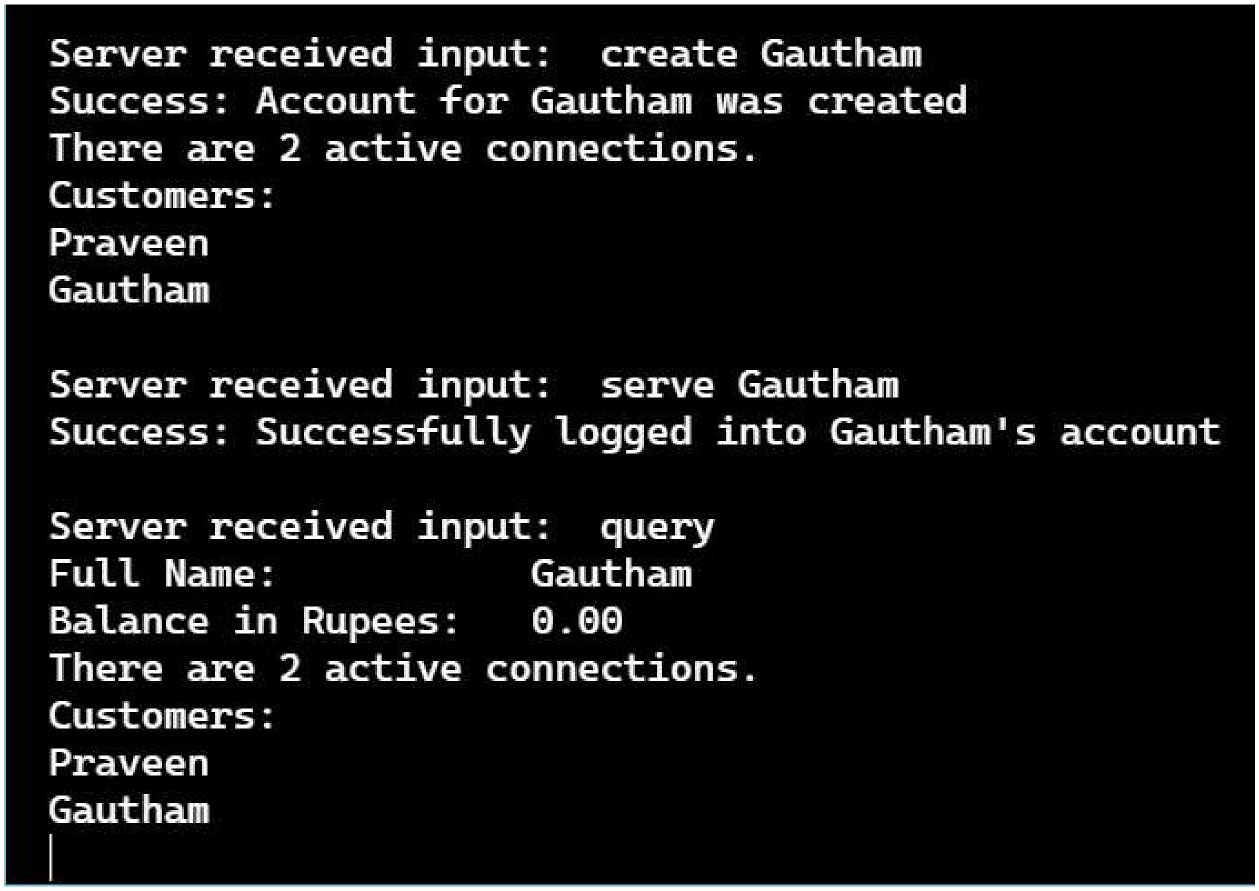
```
Enter your choice: a
Enter amount to withdraw: 600

Error: Cannot withdraw Rs600.00 from Praveen's account due to insufficient balance

Press enter to continue...
```

Meanwhile, let’s have a look at what server logs on the console:



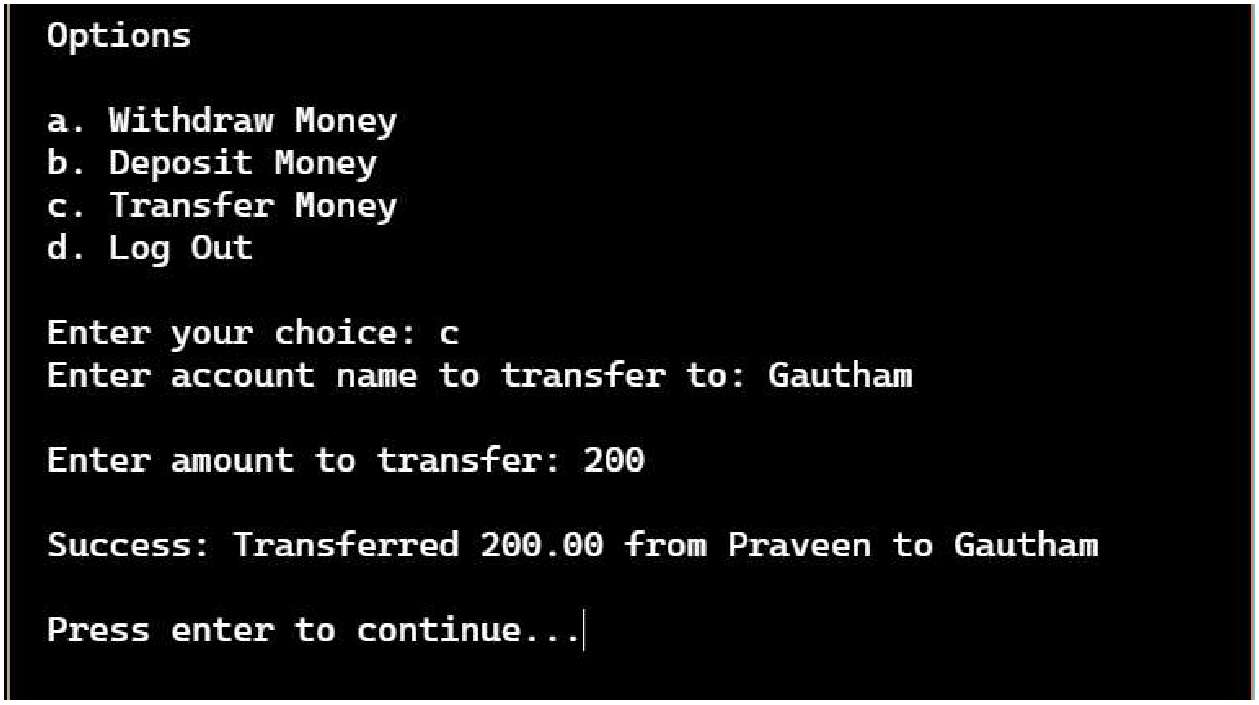
```
Server received input:  create Praveen
Success: Account for Praveen was created
There is 1 active connection.
Customers:
Praveen

Server received input:  serve Praveen
Success: Successfully logged into Praveen's account

Server received input:  query
Full Name:           Praveen
Balance in Rupees:   0.00
There is 1 active connection.
Customers:
Praveen

Server received input:  deposit 500
Success: Deposited Rs500.00 into Praveen's account
There is 1 active connection.
Customers:
Praveen

Server received input:  query
Full Name:           Praveen
Balance in Rupees:   500.00
```

Creating multiple users:

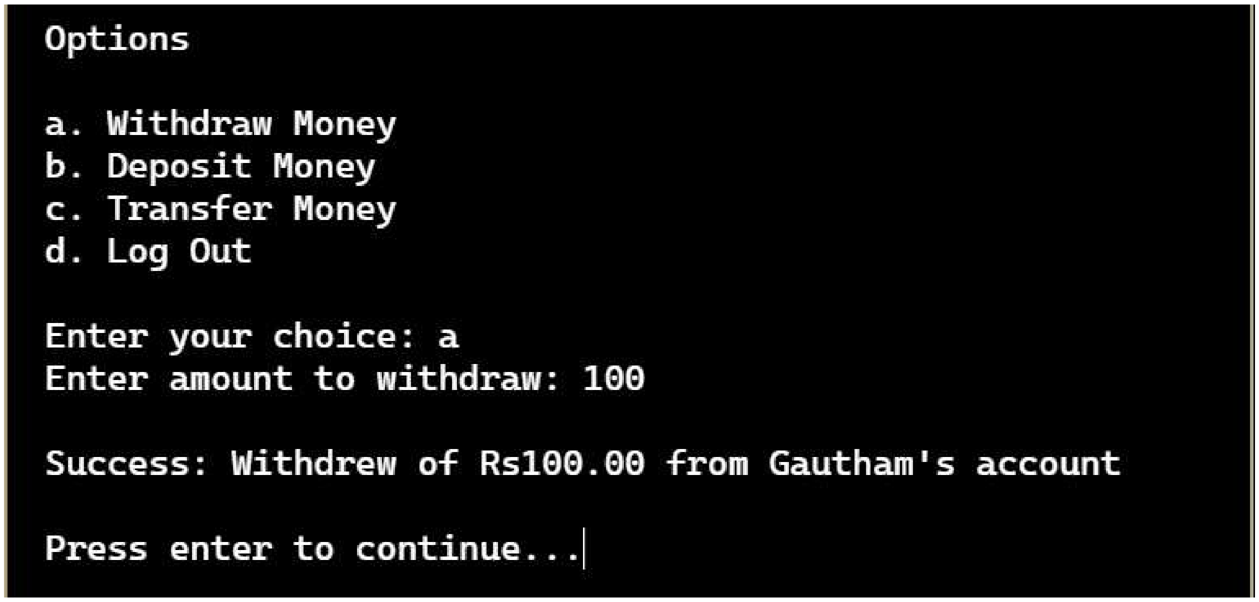


```
Server received input:  create Gautham
Success: Account for Gautham was created
There are 2 active connections.
Customers:
Praveen
Gautham

Server received input:  serve Gautham
Success: Successfully logged into Gautham's account

Server received input:  query
Full Name:           Gautham
Balance in Rupees:   0.00
There are 2 active connections.
Customers:
Praveen
Gautham
```

Transfer between accounts:

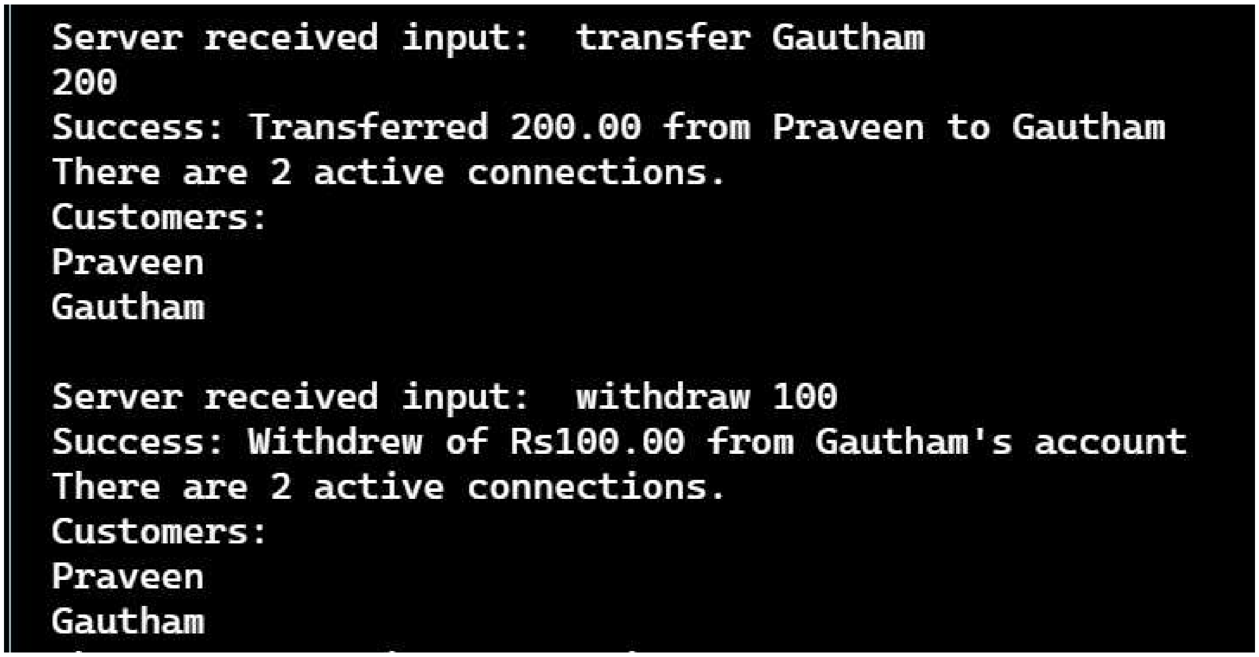


```
Options

a. Withdraw Money
b. Deposit Money
c. Transfer Money
d. Log Out

Enter your choice: c
Enter account name to transfer to: Gautham

Enter amount to transfer: 200

Success: Transferred 200.00 from Praveen to Gautham

Press enter to continue...
```



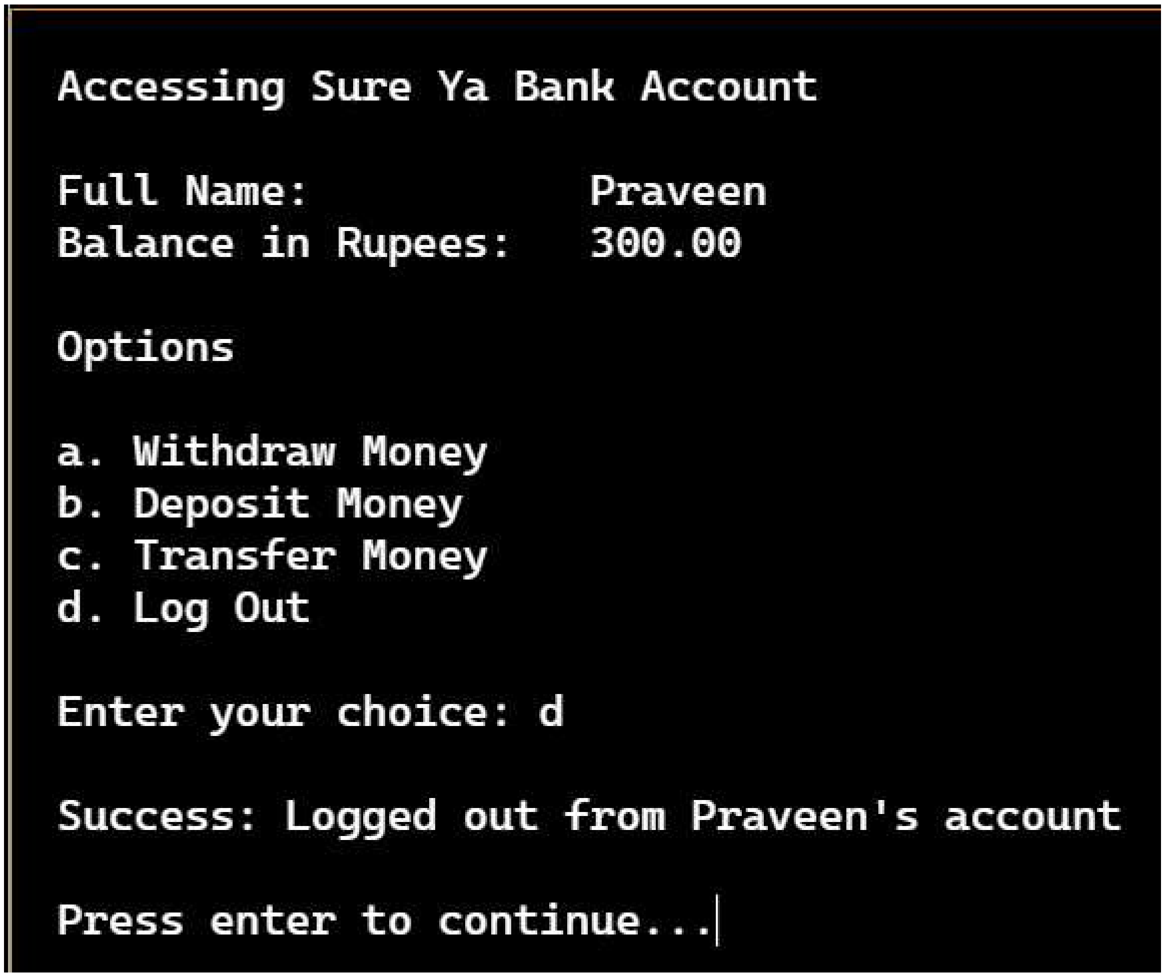
```
Options

a. Withdraw Money
b. Deposit Money
c. Transfer Money
d. Log Out

Enter your choice: a
Enter amount to withdraw: 100

Success: Withdrew of Rs100.00 from Gautham's account

Press enter to continue...
```

Server logs:



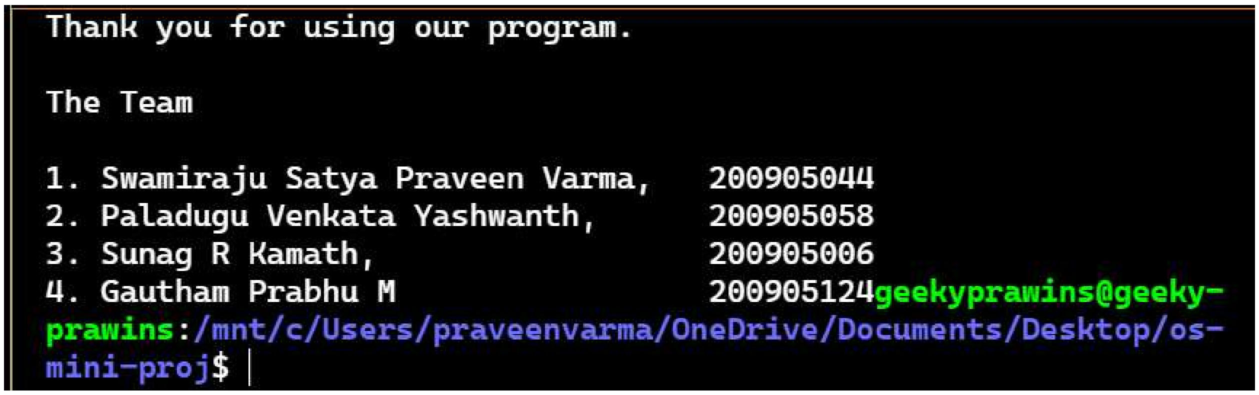
```
Server received input:  transfer Gautham
200
Success: Transferred 200.00 from Praveen to Gautham
There are 2 active connections.
Customers:
Praveen
Gautham

Server received input:  withdraw 100
Success: Withdrew of Rs100.00 from Gautham's account
There are 2 active connections.
Customers:
Praveen
Gautham
```

Logout from account:



```
Accessing Sure Ya Bank Account

Full Name:            Praveen
Balance in Rupees:    300.00

Options

a. Withdraw Money
b. Deposit Money
c. Transfer Money
d. Log Out

Enter your choice: d

Success: Logged out from Praveen's account

Press enter to continue...
```

Terminate connection with server:



```
Thank you for using our program.

The Team

1. Swamiraju Satya Praveen Varma,   200905044
2. Paladugu Venkata Yashwanth,      200905058
3. Sunag R Kamath,                  200905006
4. Gautham Prabhu M                 200905124geekyprawins@geeky-
prawins:/mnt/c/Users/praveenvarma/OneDrive/Documents/Desktop/os-
mini-proj$
```

CONCLUSION:-

We have created a robust system using **Threads** and **Mutex locks** in our Linux-based multi-threaded banking system. Since the threads are interconnected, our menu-driven application makes it simple to know some crucial facts, such as transaction and account details. In our project, thread synchronization is effectively managed using Mutex locks to ensure that two or more threads do not execute important sections concurrently. We used multithreading to decrease banking complexity through thread parallelism. Our project will walk you through financial processes including opening accounts, withdrawing money, transferring money to different accounts, etc. with the use of straightforward solutions that make it easy to resolve any pressing banking concerns. The evaluated code produced satisfactory results in a variety of real-life banking circumstances.