Computer Science Technical Report

Remote Cal-C

Jaspreet Singh
Dikshay Palta
Manpreet Singh Bhamra
Mehak Grover
Monika Monika
Group 10
{singh1i6, palta, mpbhamra, groverm, monikam}@uwindsor.ca

Advanced Systems Programming-656
March 29, 2017

Department of Computer Science, Windsor, ON, Canada

Contents

Abstract

As we all know that calculation is important part of our life. From school kids to adults everybody uses it for carrying out various tasks serving different purpose. we have picked up the simple concept but amplified its scope to span all user requirements. Our application can do simple operations to complex scientific computations.

We have used a very friendly user interface enabling the user to perform the calculation. One of these interfaces is designed in Android and other interface is accessible from the UNIX command line. Server and UNIX client is purely developed in C but the other client is developed in Android. Design is simple yet effective hiding the internal complexity of the programs. Further the communication between the two is achieved using sockets programming.

Android interface is very appealing for the non-technical users. However, the UNIX command line interface is more appealing to the computer literate people. Proximity to day to day life computations and good interface is the main goal of the project. A good cohesive design has really tailored this application for all users.

Further we are planning to incorporate finance and business features consolidating the flexibility of this application to fit diverse user needs.

Introduction

Cal-C is an acronym for the calculator that performs calculations from basic operations to complex mathematics. The primary idea is to facilitate the various range of users to perform real-time computations with ease and at a lightning fast pace to save time and reduce the human error and knowledge complexity. Since mathematics is a vast domain and it’s extensively used in STEM fields, so this automation of calculations, is useful for further sub domains i.e. probability and statistics. It is really a boon for novice users facilitating them to do real-time computations improving efficiency. The interactive user interface is friendly and hides the internal complexity of the program.

Primary insight of the project is to deliver an interactive application that spans from beginner to advanced learners. Generally, most of the high school kids are having trouble remembering algebraic identities, tons of formulas, but now with Cal-C, we have incorporated widely used formulas and other trigonometric functions so that they can easily do computations quick and easy. Now, keeping in mind the 40s, we have special functionalities that allow these people to calculate their mortgage estimates and other dues further consolidating investment and debt payoffs. Vision to serve users of all knowledge base and needs is the driving force behind this project.

We have used a simple concept but amplified its application in a pragmatic way. This distributed application is based on a client-server interaction through one of the protocols. A server process accepting multiple client processes is written in C and implements a remote connection providing a service to the client. The user can either access through the running C client program from the interactive UNIX shell as a command line utility or use the graphical user interface which is implemented using android. The communication between the client and server is achieved using sockets programming. The user can interact through either of the interfaces and perform the computations as per requirement.

Even though the application is extremely simple but accessibility through a shell client utility is really challenging for novice users. Furthermore, there is a need for a tutorial for the computer illiterate people so that they can be enlightened about proper usage and functionality and get over some of the knowledge constraints. Also, a calculator inhibits the learning of aspiring young lads affecting logical thinking and critical reasoning. Proximity to user’s needs along with impeccable design is the driving vision behind this concept.

Requirement Analysis

Basically, some of the requirements for this project are based on the client/server. The common requirement between these two is to establish a connection and then communicate among each other to transfer data. In this project, we are using the TCP/IP protocol to allow communication between client and server. In networking concept this process is also known as handshaking.

Basics steps required for connection at the server may include:

1. Socket Creation: - This is the first step to start a two-way communication between the server and the client. Basically, a socket act like a communication link between the client/server and is bound to a specific port which is known to the client so that it can interact with the server. The basic syntax for creating socket is: socket (family, type, protocol) and it is defined in the sys/socket.h header file.
2. Socket Binding: - After a socket has been created the next step is to bind the socket with particular address, so that client can connect with socket by providing that address and port number on which server is running. We can bind address with socket by using bind() function.
3. Listening: - Once the socket has been configured successfully, it is the time for server to listen any new connection from client on the socket. We can also specify the number of client a server can handle at a time by passing the parameters into listen function. The basic syntax for listen function is: listen(int sockfd, int num). Here sockfd is the socket name that we have created earlier and num is the number of connections server is going to accept.
4. Accept: - While listening to the connection on socket, if server founds any request from the client then it is going to accept it using accept() function. Once the request from client has been accepted by the server then it means a client can interact with the the server.
5. Transfer: - This is the last step in which a two-way communication link is established between the client/server and now they can send or receive data among each other by writing to or reading from socket. They can read or write data to the socket by using read() and write() function.

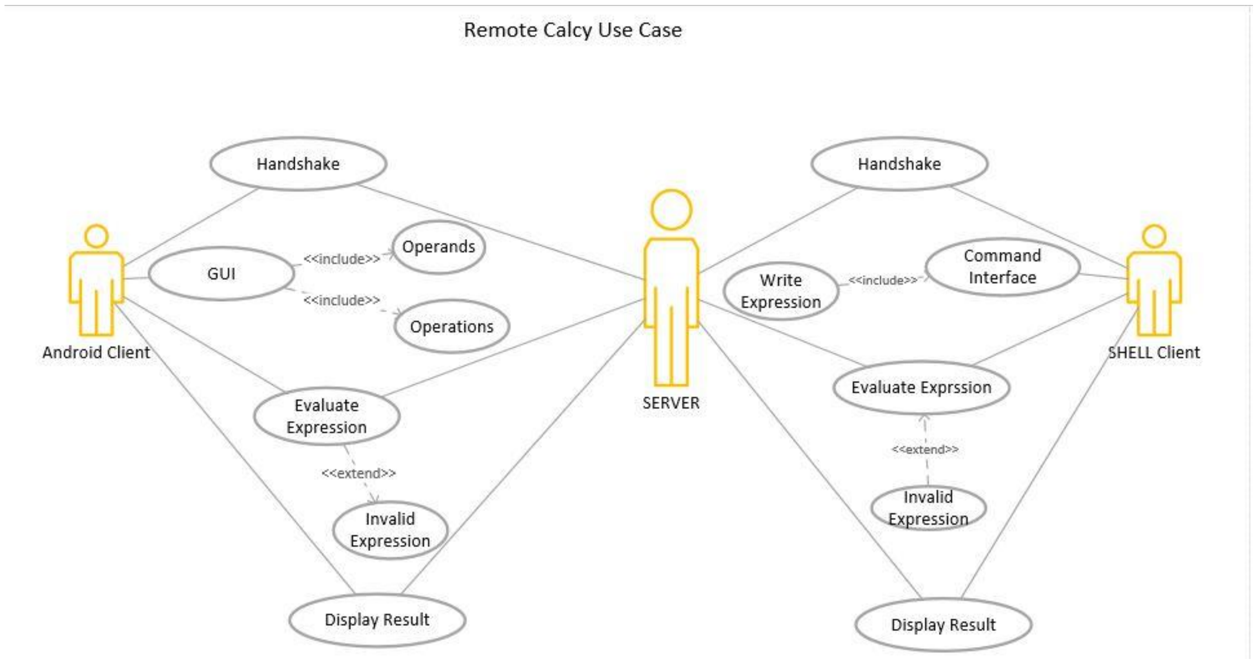
Basics steps required for connection at the Client may include:

1. Socket Creation: - To establish two-way communication we need to create socket on both the ends. We can use the same procedure as we used at the server side to create socket at the client side.

2. Connect to Socket: - Once the socket has been created successfully, the next step is to connect the host to the socket. We can use the same step for this also as we have used at the server side for socket connection.
3. Transfer: - This would be our last step at the client side, after the server accepted the client request, we can make them to start communicating among each other.

To make it more clear how does our project works we have developed a USE CASE diagram which depicts the sequential flow of our project.

Use Case Diagram:



As explained above handshake will cover all the steps till the connection has been established between two hosts. After they are connected to each other, it is the time for the client to use their respective interfaces and send the mathematical expression to the server that needs to be solved. So, in this case we have two clients and one server. We built a user-friendly android interface for the android client. So, it can send his request to the server by using that interface and correspondingly server will evaluate the result and give it back to the client.

On the other hand shell client, will use his command line interface to interact with the server and send expression to be evaluated by typing it on the interface and will get back the result from the server.

Whereas we also validation each expression at the server side.If the user enters the wrong expression then server will respond back by displaying a error message.

We have divided our whole project into three phase:

1. Phase I: - This phase is also known as an Inception phase. In this phase, we went through all the requirements that are to be needed for our project and decided whether it is feasible to make it or not. We also decided on which tools and technology we are going to work for our project development.
2. Phase II:- At this phase we analyze all the tools that are to be used in our project and get understanding of that tools, so as to used them effectively while developing project. As in our project we need to implement client/server architecture, so we spent our time on learning socket programming.
3. Phase III:- This is the final phase of our project. In this phase, all the functionality that have to be ad on are added to it and also test the whole project by using system testing to make sure that server is responding back to the client at fast pace. Other testing that we have done include interface and scalability testing.

Design

In this project, we are making a remote calculator in which server sends back the desired result to the user or client where the client can be a C client or Android client. It is a step by step approach that we have used the accomplish our goal for this project. Following are the steps that are followed:

1. Information Gathering:

Gathering the information is the most important thing that is required to design a prototype.
The basic motive is give the user the desired and accurate calculation which are used in day to day life. A client – server approach is required to fulfill the need.

2. Plan:

As per the information gathered the plan is to provide user a reliable remote calculator which can be used from anywhere while connected with the internet. This problem can be solved by a client interface where one client is graphical (Android) and other is Unix based. A C based server is used to send the accurate result. To meet the goal, we have decided to make a Server which will use the client’s expression and send back the result to the client. Technology used: C and Android.

3. Design Phase:

It concentrates on the how will the design looks like. It should be user interactive and easy to use. To accomplish this, we will be having a server will tells the user to enter the input when client gets connected. Similarly, a user interactive client is there, which tell the user to enter an input in C and Android based client is having graphical buttons which are very easy to use.

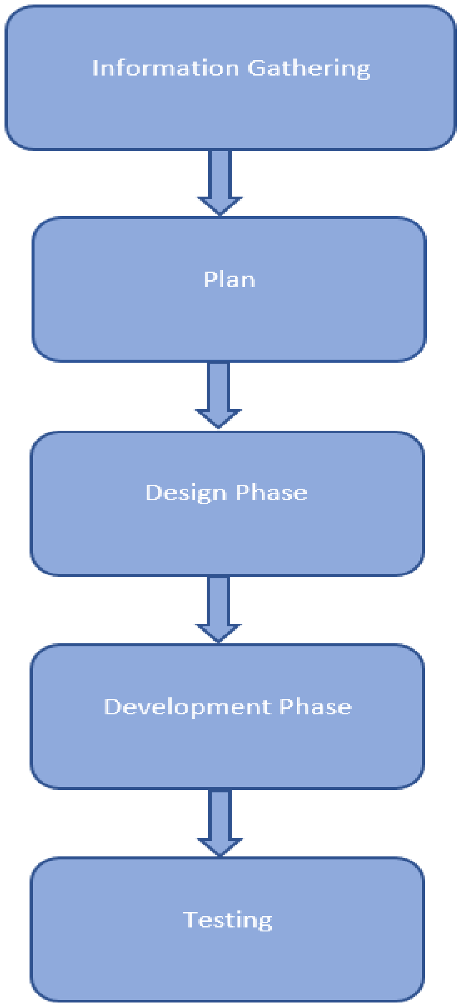
4. Development Phase:

In this step, the development of android client and C client was started. We started making the C server, C client and Android client. First the basic calculator was designed in the Android studio and sockets in C for both client and server.

5. Testing the Project:

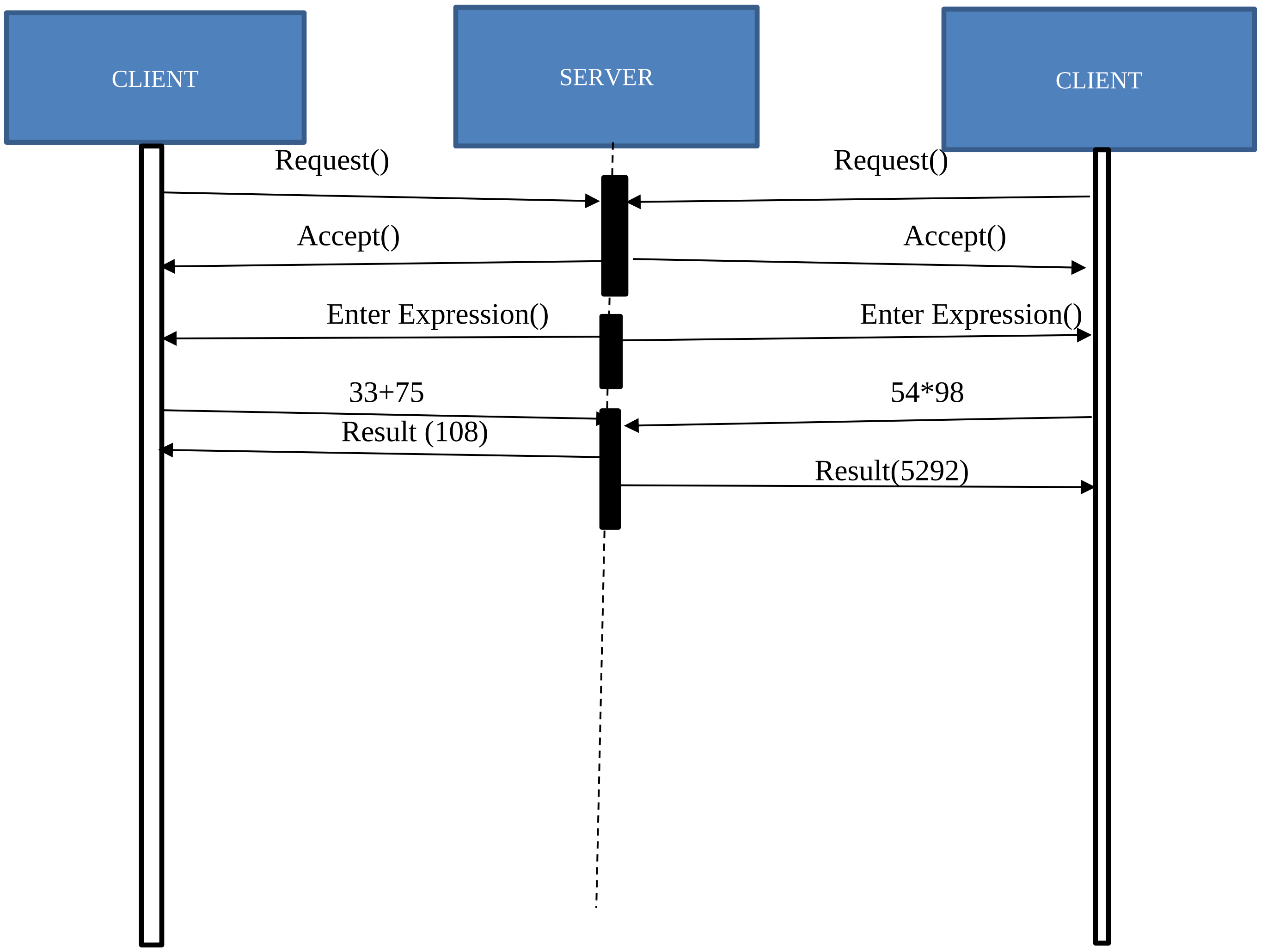
Testing is most important phase without this we cannot publish the project. The android app and C client & server are up and running, but we need to check whether they are giving the correct result to the user. In order to achieve this, we have to first check that the Server is able to connect with the client or not. If is connects then, checking that the client can now send the expressions to the server and the server is able to send the result. After checking all the integration among the client and server and removing the bugs the testing phase is completed.

Flow Diagram of the design is as follows:



IMPLEMENTATION

Implementation gives an analysis how our research on project Remote Cal-C was carried on. There are numerous stages that result to the creation or development of a software product. A short time period of 3 weeks has been contributed to it. Each iteration like requirement gathering, Design, Implementation, testing, deployment has a purpose to result a real business value at an early phase. The implementation of each and every component was done independently and were amalgamated and integrated when the development of components was completed. The goal of this project was to perform calculations from basic operations. This application is based on Client Server Interaction through the protocols.



Sequence Diagram

Requests by multiple clients is handled by server application and is built on a single machine. It handles several requests, stores the detail and processes the request and send the response back to the client (example -enter expression). The client again sends the request to the server(example-33+75,54*98), the server processes on the request and send back the result(eg-108,5292) related to the request

System Requirements:

Server side:

Hardware components: PC, CPU, Memory, NIC etc.
Software components
Operating System
Communication middleware e.g. IPX
Language: C
Development tools: Code Blocks, Putty, No Machine
Repository : GitHub, Redmine Uwindsor

Client Side:

Hardware
Operating system
Graphical/Command Line Interface
Communication Layer
Language: C, Java
Development Tools: Code Blocks, Putty, No Machine, Android Studio 2.3, Redmine Uwindsor

PROGRAMMING LANGUAGE: C

C language is handy and the programs written in one computer could be compiled or run on another computer. It has collection of data types and operators. It is a high-level language. It can also handle low level activities.

Advantages:

- It is a procedure oriented language which is easy to learn, it also follows algorithm to develop a program to convert it into procedure or functions.
- High speed of compilation-as compared to other language compilers c compiler produces code instantly. It also optimizes the code.

- You can take this language in your pen drives, floppy drives and install it and operate. Its output file can be executed in any computer. Its portable.

TESTING

Software testing is useful to find the errors or bugs in the applications or programs by executing them and by accessing the functionalities of the program. In order, to make our (Remote_calcy) simple calculator application free from errors, bugs and to ensure about the connectivity between the server and two clients we go through Unit testings.

UNIT TESTING:

We tested the programs of Server and Clients individually to ensure about their execution.
Then after building the connection between them we ensure that:

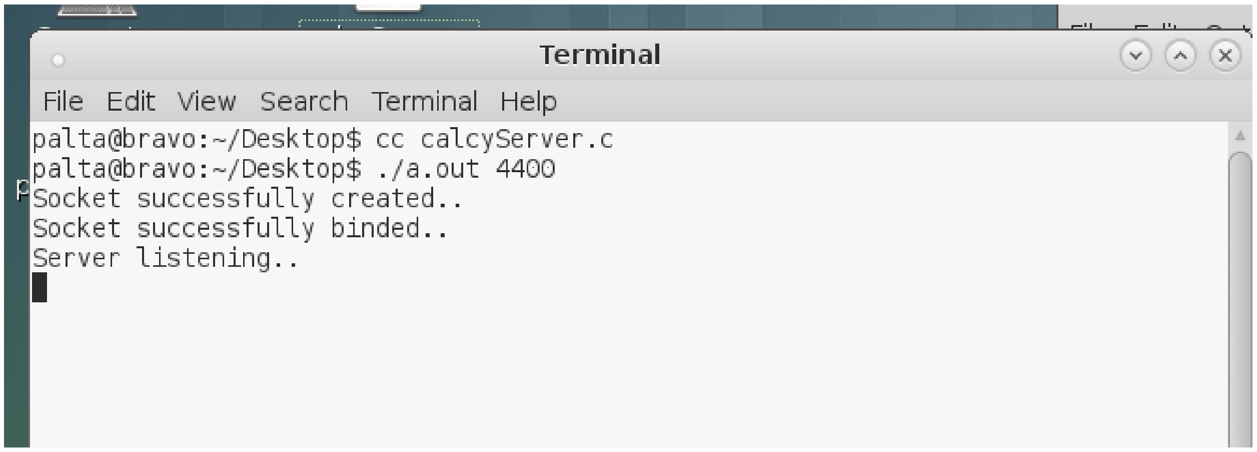
- After importing connectivity code, we rechecked both Server and Clients individually.
- Whether the connection is built between the Server and the Client 1.
- Does the Server able to read the instructions of the client.
- Vice versa to make sure that if the client is getting reply from the Server.

Several issues were generated while integrating our code. Some of them are as follows:
Firstly, we got stuck in the connectivity of the Android client and the Server.
Secondly, Server was not able to read the android client expression correctly.
Apart from these issues we have faced several challenges during the design of the product as well. These issues include:

- Design issues
- Connectivity issues
- Stack overflow.
- Security issues.
- Integration issues.

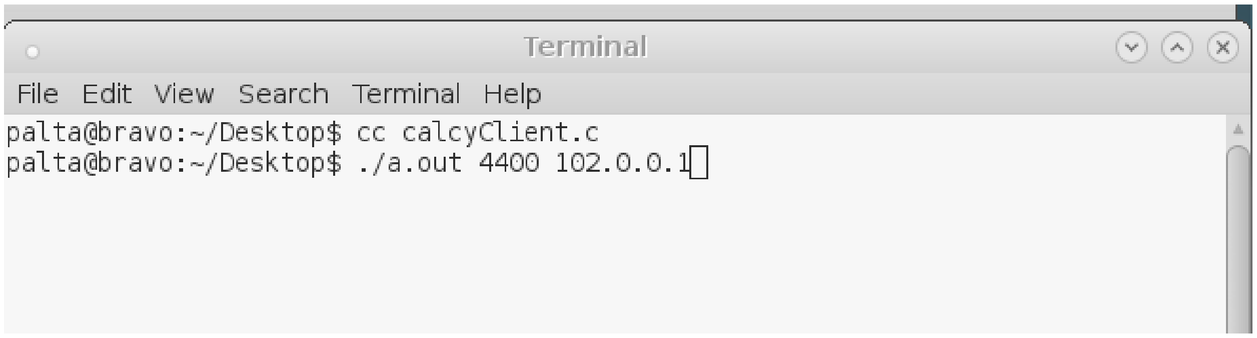
After careful debugging, all those issues were sorted and fixed. We have also stress tested our product for the maximum no of clients it can handle. We achieved the integration using the sockets programming and tested our code several times before making our product live.

PROTOTYPE



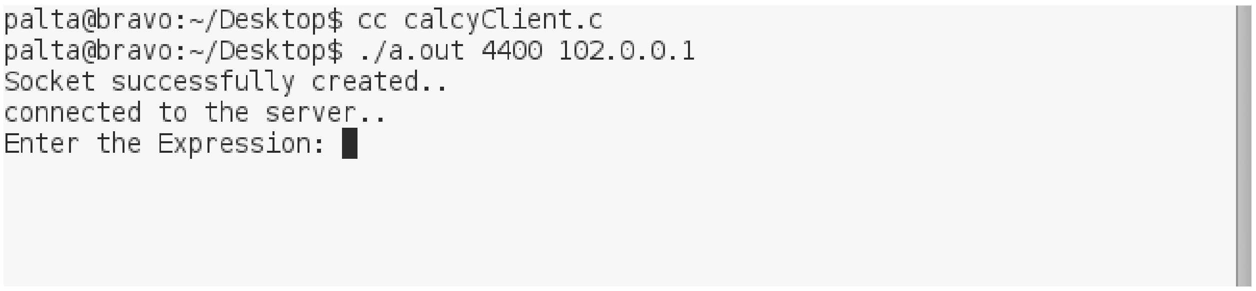
```
palta@bravo:~/Desktop$ cc calcyServer.c
palta@bravo:~/Desktop$ ./a.out 4400
Socket successfully created..
Socket successfully binded..
Server listening..
```

The first step to make our project in working condition is to compile and run the server. Once the server is running it is the time to accept new client request. So, we can do that by compiling and running the client code as shown below



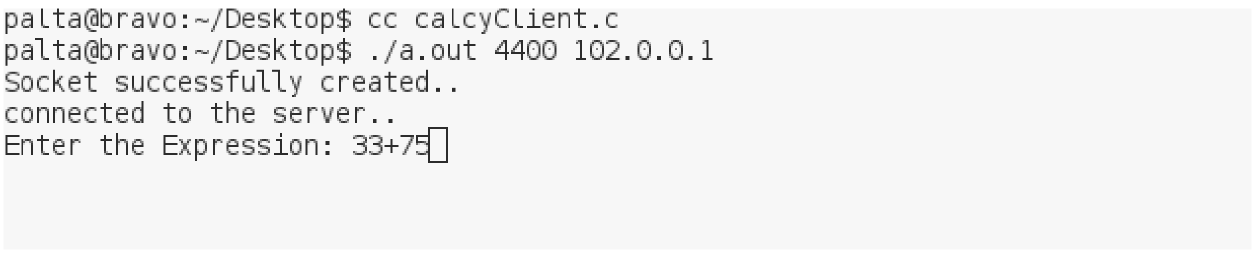
```
palta@bravo:~/Desktop$ cc calcyClient.c
palta@bravo:~/Desktop$ ./a.out 4400 102.0.0.1
```

Client will be allowed to enter the port number and the IP address to connect it with the server in order to allow communication between them.



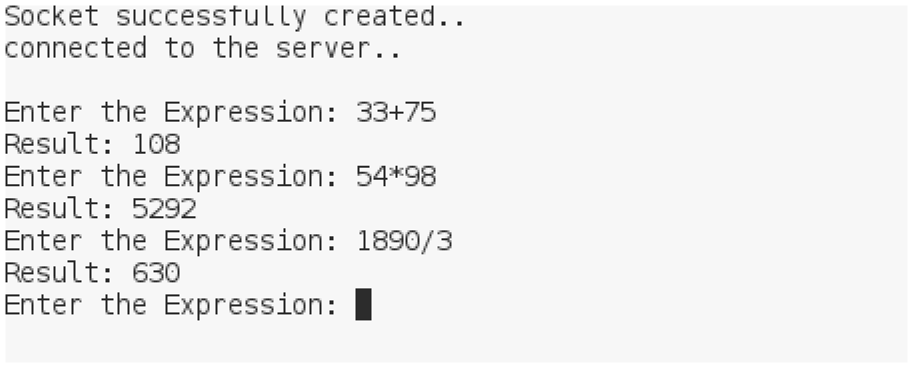
```
palta@bravo:~/Desktop$ cc calcyClient.c
palta@bravo:~/Desktop$ ./a.out 4400 102.0.0.1
Socket successfully created..
connected to the server..
Enter the Expression:
```

After the connection, has successfully established between the client and server, server will ask the client to enter the expression.



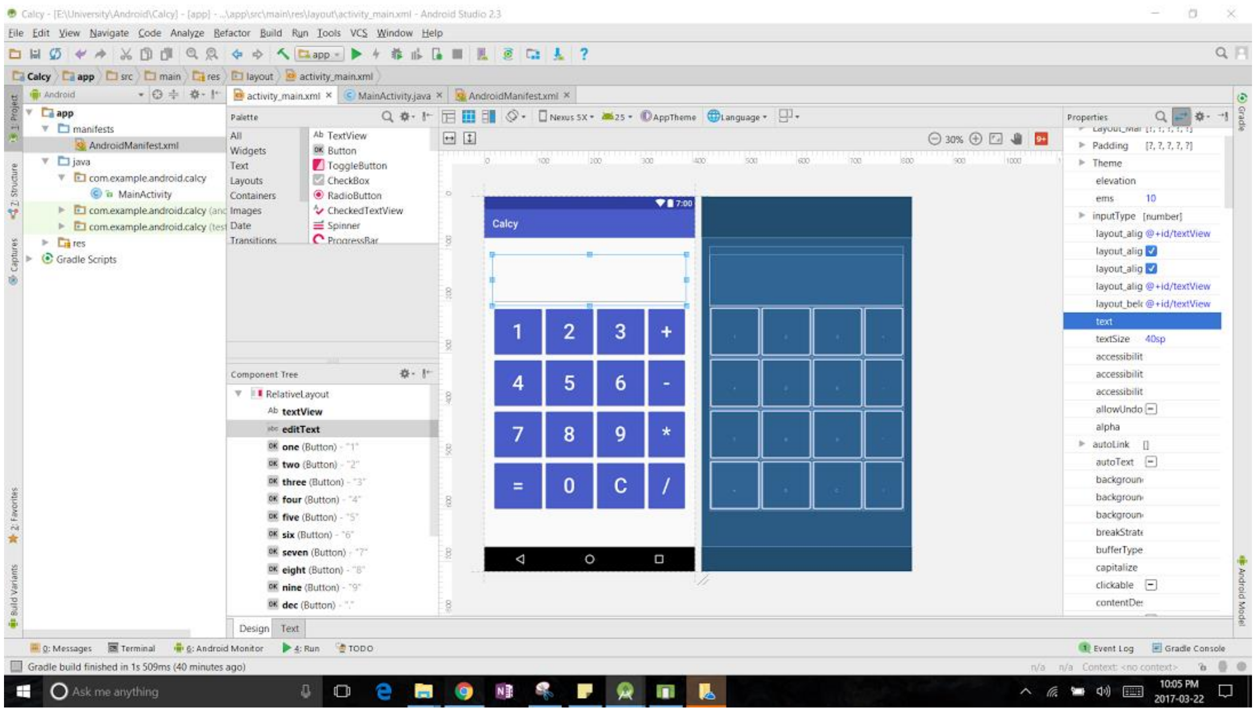
```
palta@bravo:~/Desktop$ cc calcyClient.c
palta@bravo:~/Desktop$ ./a.out 4400 102.0.0.1
Socket successfully created..
connected to the server..
Enter the Expression: 33+75
```

After entering, the expression is sent to the server and the server will evaluate it and respond it back to the client with appropriate result.



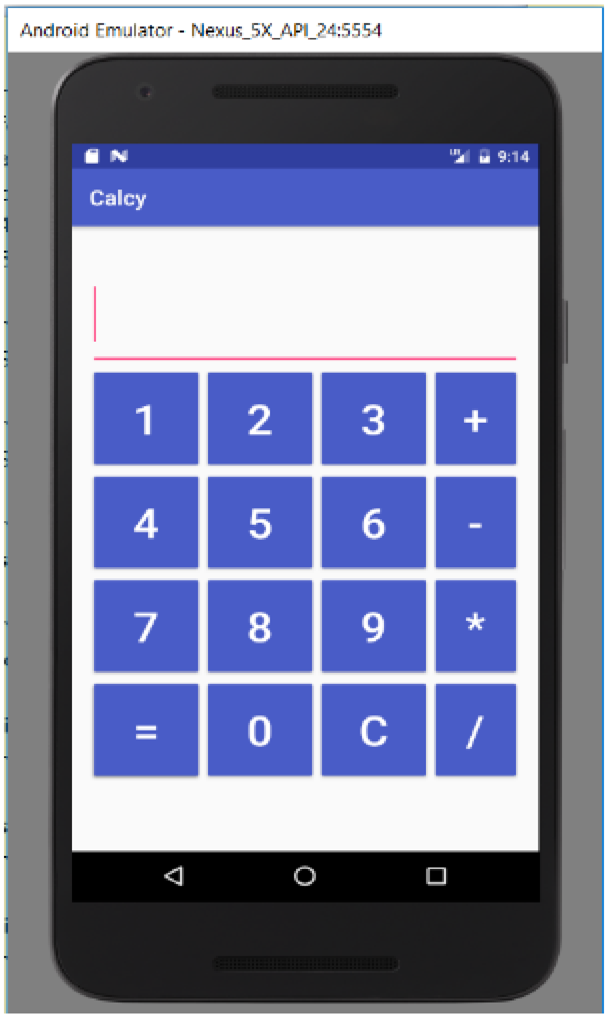
```
Socket successfully created..
connected to the server..

Enter the Expression: 33+75
Result: 108
Enter the Expression: 54*98
Result: 5292
Enter the Expression: 1890/3
Result: 630
Enter the Expression: 
```

There may be a chance that user will enter some inappropriate expression, so in that case server will check whether the expression entered by user is wrong or right if it is wrong then it will respond back to the client by displaying message 'wrong expression please try again'.
Whereas our second client interface looks like this:



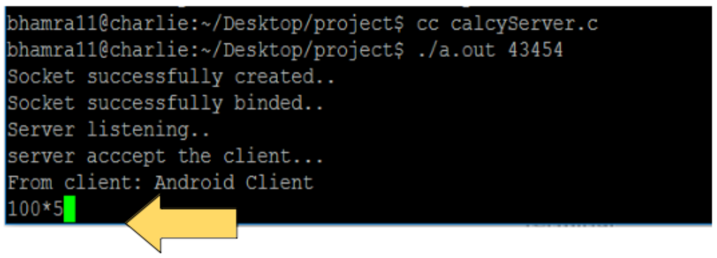
Calcy - [E:\University\Android\Calcy] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 2.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
activity_main.xml
MainActivity.java
AndroidManifest.xml
Palette
Component Tree
Properties
Calcy
1 2 3 +
4 5 6 -
7 8 9 *
= 0 C /
Design Text
Gradle build finished in 1s 509ms (40 minutes ago)
Ask me anything
10:05 PM 2017-03-22

User is allowed to perform any basics mathematical operations by entering the value into the textbox through the use of graphical interface provided to it.

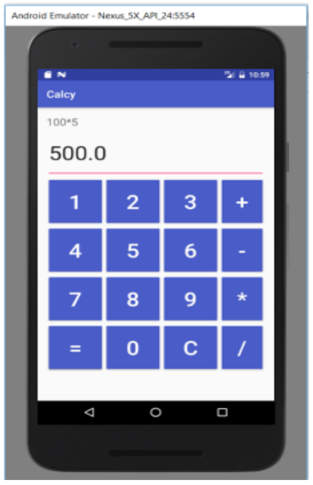


So as we can see that in this case user is performing the multiplication operation i.e 500*5. We have also added a textbox above the result textbox of android interface to make clear to the user about the expression they are performing.

Continue...



```
bhamra11@charlie:~/Desktop/project$ cc calcyServer.c
bhamra11@charlie:~/Desktop/project$ ./a.out 43454
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: Android Client
100*5
```



CONCLUSION

The Remote Cal-C application currently inculcates the basic operation of a calculator. The application does everything what it is intended to do. We built the server in C language and two clients, one in C language and one in android.

The connectivity between the server and clients is done by sockets.

Product Quality Assessment:

Unit testing is done twice to ensure the quality of the application. The application worked properly after testing and the design of the application is developed as well.

Product Dependability Assessment:

Currently the product is dependent on the input by the client and the configuration of the server as well.

Product Security Assessment:

Supports secure transferring of data between clients and server.

Platform and Technical Tools Used:

Both Client and Server were coded in C language using code blocks, putty and No machine.
Android Client in Android Studio 2.3, for Repository we used Github and Redmine Uwindsor.

Our team work was divided as follows: -
Mehak Grover and Manpreet Bhamra – Designed android client and ensured their connectivity and prepared report.

Dikshay Palta- Designed Unix client and Managed everything.

Monika and Jaspreet Singh- Worked on Server and Prepared Presentation and Report.

In future, we are planning to make it as a dynamic calculator enabling people to manage their finances and designing a dashboard and notification centre.

ACKNOWLEDGEMENT

We are delighted to express our amiable and sincere acknowledgment and indebtedness to Dr. Ziad Kobti, Professor and Director of School of Computer Science at University Of Windsor for his constant supervision and encouragement to complete this project on time as well as providing crucial information regarding the project.
We also express our abyss gratitude to all Lead Teaching Assistants for their help and support during the working stage of the project.
Copious Thanks to all the Team Members whose cooperation and teamwork made this application possible.

REFERENCES

[1]. "Developing Android Apps | Udacity", Udacity.com
URL: https://www.udacity.com/course/new-android-fundamentals--ud851

[2]. "C Socket Programming for Linux with a Server and Client Example Code",Thegeekstuff.comURL:http://www.thegeekstuff.com/2011/12/c-socket-programming

[3]. "This kind of socket isn't for your wall - learn about network sockets", Lifewire
URL:https://www.lifewire.com/socket-programming-for-computer-networking-4056385

[4]. "C Socket Programming for Linux with a Server and Client Example Code", Thegeekstuff.com
URL:http://www.thegeekstuff.com/2011/12/c-socket-programming/?utm_source=feedburner

[5]. "Computer Science <| Projects |>", Redmine.cs.uwindsor.ca
URL: https://redmine.cs.uwindsor.ca/projects/sockets1

[6]. "Download Android Studio and SDK Tools | Android Studio", Developer Std.
URL: https://developer.android.com/studio/index.html

Appendix

1. The Client-Server code is attached as two files calcyServer.c which is the Server code in C language and calcyClient.c which is the Client code in C language.

2. The Android client code is attached as calculator.zip, It consists of all .xml and .java files used for coding android calculator app.

3 The link to the Git repository used for this code is https://redmine.cs.uwindsor.ca/projects/sockets1