A Project Report

On

Client Server Model

INTERNITY

'TOGETHER WE CAN'

Created By:

Samarth Aggarwal

Mohit Bharti

Mohit Soni

ABSTRACT

This project is the detailed overview in developing a client server-based application using socket programming in a distributed computing environment. We developed a client -server based application for Login and signup of the user. CLI, socket programming, python modules,exceptions are also considered in the development stage. The communications between client server application processes using socket mechanism were mainly analyzed. The main objective of this project is to demonstrate the principles and concepts behind the socket programming as well as libraries available in python.

Socket Programming

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

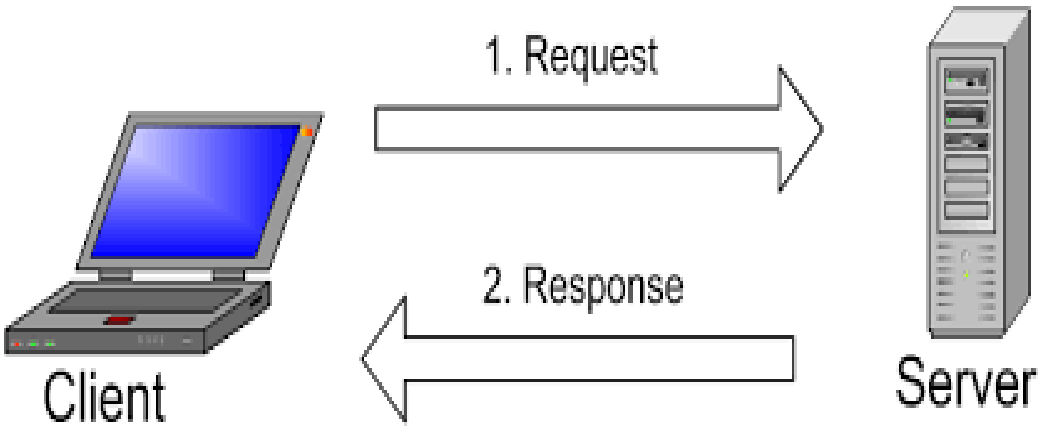
Socket programming is started by importing the socket library and making a simple socket.

import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

Here we made a socket instance and passed it two parameters. The first parameter is **AF_INET** and the second one is **SOCK_STREAM**. AF_INET refers to the address family ipv4. The SOCK_STREAM means connection oriented TCP protocol.

Client Server Communication



shown above describes the use of client server model based on its request – reply protocol. In this model, the client directly sends his request to the server asking for an appropriate service, and later server does the work and returns immediately the data or error code as a response back to the client. The server offers various services to the clients based on its client's request.

Client Request: The HTTP client sends a request message formatted according to the rules of the HTTP standard—an *HTTP Request or local request if run on local machine* . This message specifies the resource that the client wishes to retrieve, or includes information to be provided to the server.

Server Response: The server reads and interprets the request. It takes action relevant to the request and creates an *HTTP Response* message or *local request if run on local machine*, which it sends back to the client. The response message indicates whether the request was successful, and may also contain the content of the resource that the client requested, if appropriate.

Connectionless vs Connection Oriented Servers

If client and server communicate using UDP, the interaction is connectionless. If they use TCP, the interaction is connection-oriented. TCP provides the reliability to communicate across the Internet. E.g. retransmission of segments that do not arrive correctly at destination, using checksum to ensure data received is not corrupted during transmission, providing flow control etc.

UDP does not guarantee reliable delivery. UDP works well in a LAN environment but errors could arise in a WAN setup. Generally applications use TCP.

The **Application Program Interface** (API) to protocols like TCP/IP

- The TCP/IP software resides in the O/S. So application programs need to interact with O/S to use TCP/IP to communicate
- The routines provided by the O/S defines the API. The socket interface from UC Berkeley is one such interface for the BSD UNIX O/S
- Each socket is identified by a small integer, called socket descriptor. The function call, socket() creates a socket.
- TCP/IP protocols define a communication endpoint to consist of an IP address and a protocol port #

Other calls defined in the socket API include:

- bind() to specify the local endpoint address for a socket;
- connect() which is used by a client process to establish a connection to a server;
- listen() which is used by connection-oriented servers to indicate willingness to receive connections. This call places the socket of the server in a passive mode and make it ready to accept connections.
- accept() is called by a connection-oriented server to extract the next incoming request.

Proposed Method

The application was designed using socket programming in Python supported by TCP datagram. The client server-based application has many functions. The two programs (client and server) were simulated, demonstrated and analyzed.

1- Application Concept: This research study used a specific scenario in designing the client server application in order to demonstrate the proper implementation of socket programming. The design and implement a client server-based application to enter the user detail of signup and login. This application implements communication between a client and a server. Further, the application involves multiple client and one server; with the following requirements:

   a. The client sends enquiries to the server;

   b. The server executes the commands and displays whatever the server sends back to the client;

   c. The application must be able to handle multi-users access on the client side;

   d. The server manages all the requests coming from the client and maintains user's account information in the SQL database; and

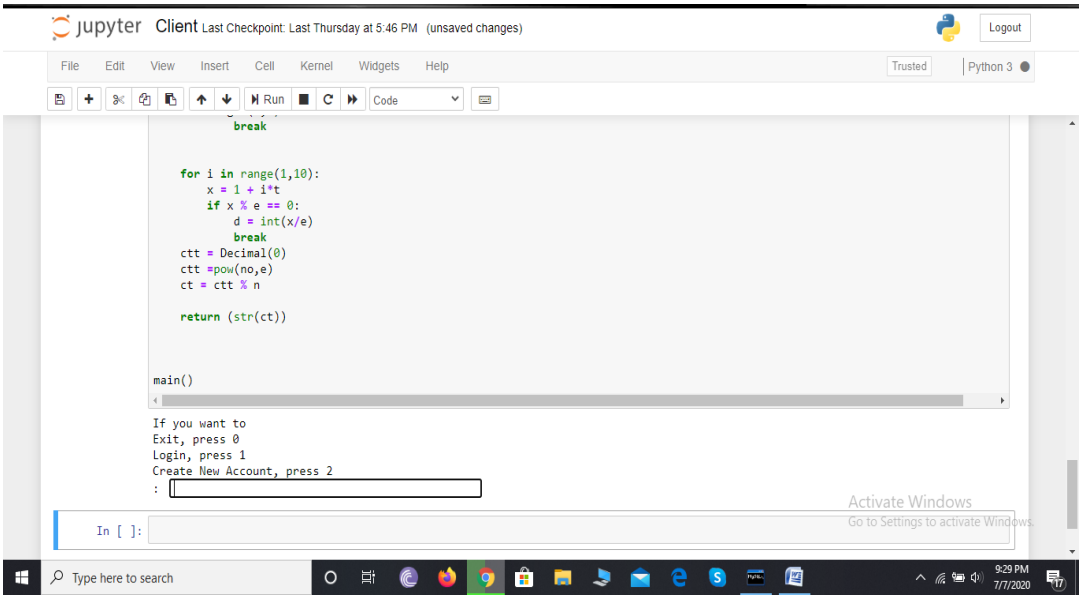
   e. The application requires log-in and password for security purposes.

   f-After establishing connection user will have the choice to Register or Login.

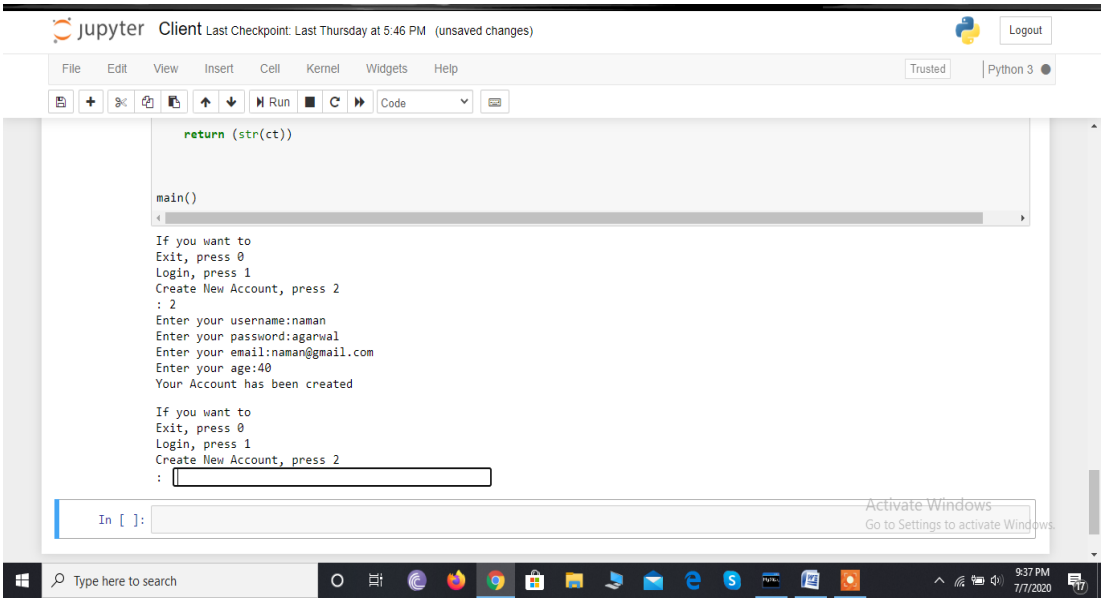
   g-If the user register then the entry will be saved to the database.

   h-If the user enters for the login, then various operation can be performed by the user accordingly.

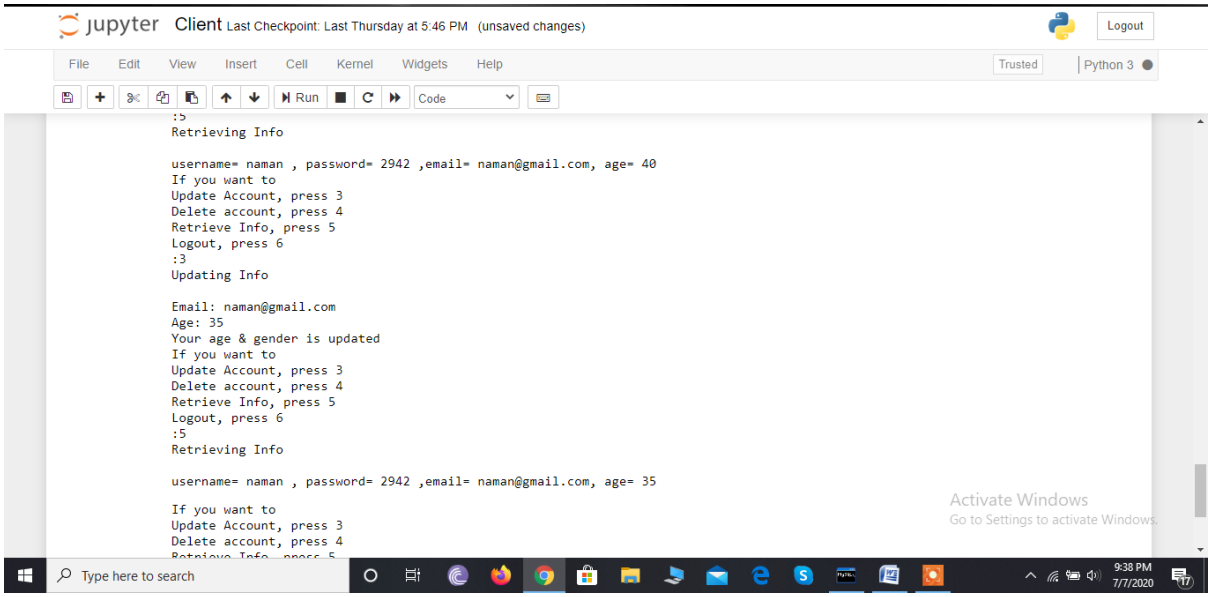
Interface



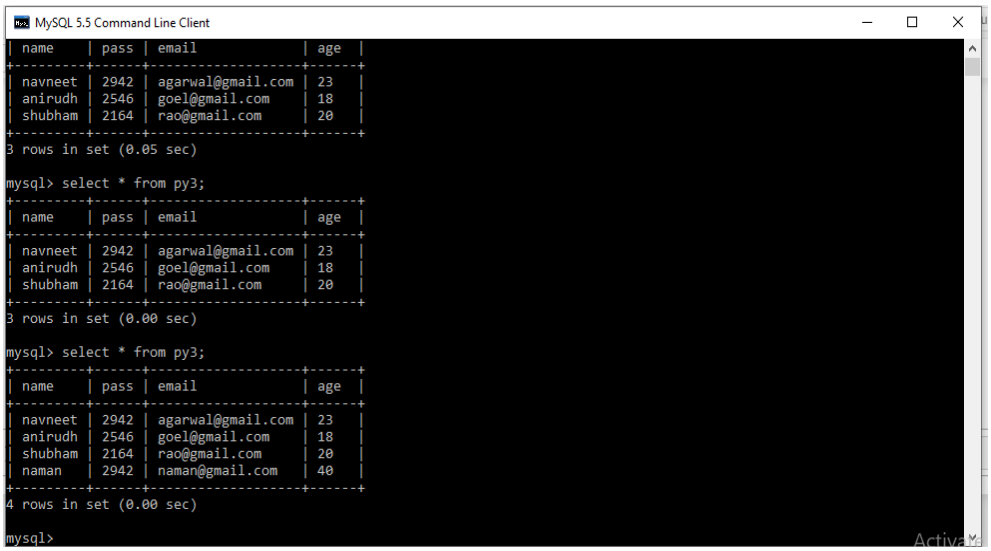
After Successfully executing the program the terminal requests for an user input, as shown in the above image in which to exit uh can press 0 and press 1 to login, if you have already registered you can directly logged in. press 2 if you are a new user you can create your new account.



In the above image we have created a new account, where for creating the account you have to enter all the necessary details required to create an account, after successfully creating account the program rerun and asked for another input from the user.



In the above image we are retrieving the user's information, first it shows the user details and asked for the input to update and delete account and to logout.



Database Created using Mysql

REQUIREMENTS

Hardware Requirements:

Device: Laptop or Desktop computer

Processor: Intel core i5 8th Gen

Ram: 2 GB

Hard Disk: 512GB

Software Requirements:

Platforms: MySQL, Jupyter Notebook

Language: Python, SQL

Technological requirements:

Python-Socket Programming, Multithreading, Python-database connectivity

References

- https://www.tutorialspoint.com/python/python_networking.htm#:~:text=A%20Simple%20Client,to%20hostname%20on%20the%20port.
- https://www.unf.edu/~sahuja/cis6302/client-server.PDF
- http://www.tcpipguide.com/free/t_HTTPOperationalModelandClientServerCommunication.htm
- https://www.geeksforgeeks.org/socket-programming-python/#:~:text=Socket%20programming%20is%20a%20way,reaches%20out%20to%20the%20server.