# Multiple Client  Server Model

# PROJECT REPORT

## Internity Foundation


**Submitted By:**

Meenakshi Goyal

Nikita Narwal

Jyotiparsad Patil

Harshit Goel

**Supervised by:**

Mr. Anirudh M Aggarwal

Mr. Rahul Bansal

## ACKNOWLEDGEMENT

It has been rightly remarked, "Success is the satisfactory achievement of chosen and desired objective. It is the attainment of the major objects; you earnestly desired and worked for with burning enthusiasm and dedication."

Before we get into things, I would like to share a few heartfelt words with the people who were part of this project in numerous ways, people who gave unending from the beginning.

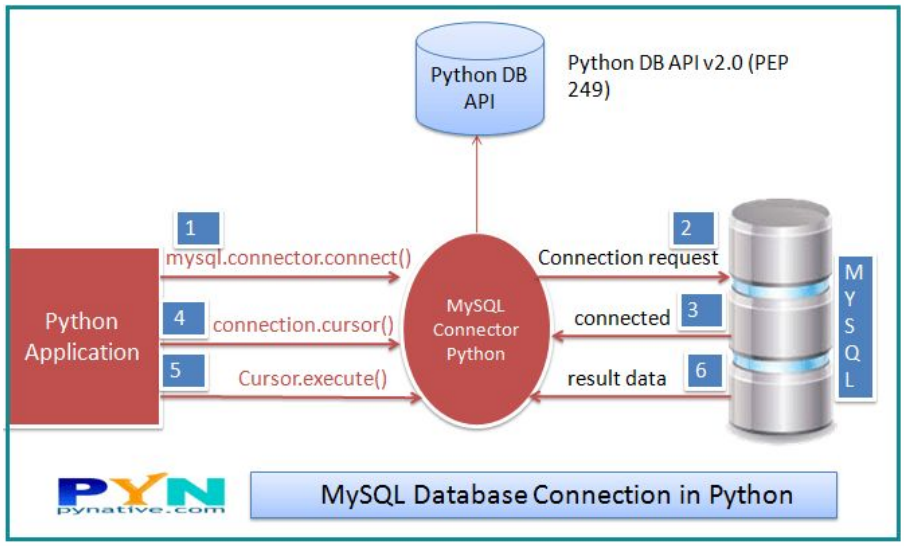
The successful completion of the project is a combined effort of a number of people, and all of them have their own importance in the achievement of objective.

I cannot miss this opportunity to thank **Mr. Rahul Bansal** and **Mr. Anirudh M Aggarwal** as a mentor for his timely support and valuable guidance throughout the project.

## Database Connectivity:

Username – i.e., the username that you use to work with MySQL Server. The default username for the MySQL database is a root

- **Password** – Password is given by the user at the time of installing the MySQL database. If you are using root then you won't need the password.
- **Host Name** – is the server name or Ip address on which MySQL is running. if you are running on localhost, then you can use localhost, or it's IP, i.e. 127.0.0.0
- **Database Name** – Database name to which you want to connect. Here we are using Database named 'Electronics' because we have already created this for our example.
- **Steps to connect MySQL database in Python using MySQL Connector Python**
  - Install MySQL Connector Python using pip.
  - Use the `mysql.connector.connect()` method of MySQL Connector Python with required parameters to connect MySQL.
  - Use the connection object returned by a `connect()` method to create a `cursor` object to perform Database Operations.
  - The `cursor.execute()` to execute SQL queries from Python.
  - Close the Cursor object using a `cursor.close()` and MySQL database connection using `connection.close()` after your work completes.
  - Catch Exception if any that may occur during this process.



```
import mysql.connector
```

- This line imports the MySQL Connector Python module in your program so you can use this module's API to connect MySQL.

```
from mysql.connector import Error
```

- mysql connector Error object is used to show us an error when we failed to connect Databases or if any other database error occurred while working with the database. Example ACCESS DENIED ERROR when username or password is wrong.

```
mysql.connector.connect()
```

- Using this method we can connect the MySQL Database, this method accepts four required parameters: Host, Database, User and Password that we already discussed.
- connect() method established a connection to the MySQL database from Python application and returned a MySQLConnection object. Then we can use MySQLConnection object to perform various operations on the MySQL Database.
- The `Connect()` method can throw an exception, i.e. Database error if one of the required parameters is wrong. For example, if you provide a database name that is not present in MySQL, then Python application throws an exception. So check the arguments that you are passing to this method.

```
connection.is_connected()
```

- is_connected() is the method of the MySQLConnection class through which we can verify is our python application connected to MySQL.

```
connection.cursor()
```

- This method returns a cursor object. Using a cursor object, we can execute SQL queries.
- The MySQLCursor class instantiates objects that can execute operations such as SQL statements.

Cursor objects interact with the MySQL server using a MySQLConnection object.

```
cursor.close()
```

- Using the cursor's close method we can close the cursor object. Once we close the cursor object, we can not execute any SQL statement.

```
connection.close()
```

- At last, we are closing the MySQL database connection using a close() method of MySQLConnection class.

## Socket Server with Multiple Clients | Multithreading

## Socket Server Multithreading

Create a Server script first so that the client communicates with it. So for that, we need to import the socket library to establish a connection and thread library for multithreading.

```
import socket

import os

from _thread import *
```

Now let's create a socket connection using socket() of socket library. Along with this let's declare host and port on which we need to communicate with clients.

```
ServerSocket = socket.socket()

host = '127.0.0.1'

port = 1233

ThreadCount = 0
```

Now let’s bind the host and port to the socket server we created above in the program. So if it binds successfully then it starts waiting for the client otherwise it just returns the error that occurred while establishing a connection.

```
try:

    ServerSocket.bind((host, port))

except socket.error as e:

    print(str(e))

print('Waitiing for a Connection..')

ServerSocket.listen(5)
```

## Handle Multiple Clients Function

So what we need is that we need to support handle multiple clients or thread at the same time simultaneously.

So for that, we need to create a function that handles requests from the individual client by a thread.

So let’s defined a new function named threaded_client which connect to each individual client on the different address given by the server.

In this function, we are using recv() function to get data from each client independently and then we simply return the reply to the particular client with the same message with string concatenate “Server Says” in the beginning.

You can type your custom message or you can also provide run time messages typed to the client.

```
def threaded_client(connection):

    connection.send(str.encode('Welcome to the Server\n'))

    while True:

        data = connection.recv(2048)

        reply = 'Server Says: ' + data.decode('utf-8')

        if not data:

            break

        connection.sendall(str.encode(reply))

    connection.close()
```

## Accept Client Socket Connection

Now what we want is to run our Server all the time, which means we don't want to make that our Server get stopped.

So for that, we need to use a while loop to make it run Server endlessly until we manually stop the Server.

Now we are going to accept connection from client using accept() function of socket server.

It returns the type of client which has connected and along with the unique thread number or address provided to it.

Then we use start_new_thread() function of thread class which creates or assign a new thread to each client to handle them individually.

```
while True:

    Client, address = ServerSocket.accept()

    print('Connected to: ' + address[0] + ':' + str(address[1]))

    start_new_thread(threaded_client, (Client, ))

    ThreadCount += 1
```

```
    print('Thread Number: ' + str(ThreadCount))


ServerSocket.close()
```

## Socket Client Multithreading

Now, we have already implemented our Server Side which accepts or handles multiple clients connected simultaneously.

Multiple clients can connect to server and each time a client connects a corresponding thread is created for handling client requests

So, now we want to write the source code for the client-side so that the client can connect to the server we created.

So in this Client-Server, we need the same socket library to establish a connection with Server-Side.

Now, what we need is to assign same host and port number to the client as we defined in the Server otherwise it will not make the connection between them.

```
import socket

ClientSocket = socket.socket()

host = '127.0.0.1'
```

```
port = 1233
```

Now we want to set up a connection using connect() of the socket library which establishes a connection with the server using host and port we provided.

```
print('Waiting for connection')

try:

    ClientSocket.connect((host, port))

except socket.error as e:

    print(str(e))
```

Now what we want is to make sure that the Client keeps running as the Server is Running. So for that, we need to use a while loop for that.

And we also going to provide input option to the client so that it can send data back to the Server and along with this we also use the recv() function to receive data from Server Side.

```
Response = ClientSocket.recv(1024)

while True:

    Input = input('Say Something: ')
```

```
    ClientSocket.send(str.encode(Input))


    Response = ClientSocket.recv(1024)


    print(Response.decode('utf-8'))
```

**The SQL UPDATE**

```
print("Enter new data which you want to update. If you want tokeep
the columns same then enter the same data")

uname = input("Enter the username")

mycursor.execute("select userpass from netusers.connect where
username = %s", (uname,))

passresult = mycursor.fetchone()

mydb.commit()



if passresult is None:

   print("User Not exists")

else:

   pwd = passresult[0]

   userpass = input("Enter the UserPass")

   userpass = encrypt(userpass)

   if (userpass == pwd):
```

```
        # connect to the server on local computer

        s.connect(('127.0.0.1', port))

        usernamenew = input("Enter the new Username")

        Emailnew = input("Enter new E-mail")

        Phonenonew = input("Enter new Phoneno")

        # receive data from the server

        print(s.recv(1024))

        sql_update_query1 = """UPDATE netusers.connect SET username =
%s WHERE    userpass = %s"""

        inputData1 = (usernamenew, userpass)

        mycursor.execute(sql_update_query1, inputData1)

        sql_update_query2 = """UPDATE netusers.connect SET Email = %s
WHERE userpass = %s"""

        inputData2 = (Emailnew, userpass)

        mycursor.execute(sql_update_query2, inputData2)



        sql_update_query3 = """UPDATE netusers.connect SET Phoneno =
%s WHERE    userpass = %s"""

        inputData3 = (Phonenonew, userpass)

        mycursor.execute(sql_update_query3, inputData3)

        mydb.commit()
```

```
        # close the connection

        s.close()

        print("Updation successfull")



    else:

        print("Wrong Pass")
```

**The SQL Delete**

```
uname = input("Enter the username")

mycursor.execute("select userpass from netusers.connect where
username = %s", (uname,))

passresult = mycursor.fetchone()

mydb.commit()



if passresult is None:

    print("User Not exists")

else:

    pwd = passresult[0]

    userpass = input("Enter the UserPass")

    userpass = encrypt(userpass)
```

```
    if (userpass == pwd):

        # connect to the server on local computer

        s.connect(('127.0.0.1', port))



        # receive data from the server

        print(s.recv(1024))



        mycursor.execute("DELETE FROM netusers.connect WHERE username
= %s", (uname,))

        mydb.commit()




        # close the connection

        s.close()

        print("Deletion successfull")



    else:

        print("Wrong Pass")
```

## Server.py

```
import socket

import os

from _thread import *


s = socket.socket()

host = '127.0.0.1'

port = 1233

ThreadCount = 0

try:

    s.bind((host, port))

except socket.error as e:

    print(str(e))


print('Waitiing for a Connection..')

s.listen(5)




def threaded_client(connection):
```

```
    connection.send(str.encode('Welcome to the Server\n'))
    while True:
        data = connection.recv(2048)
        reply = 'Server Says: ' + data.decode('utf-8')
        if not data:
            break
        connection.sendall(str.encode(reply))
    connection.close()


while True:
    Client, address = s.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(threaded_client, (Client, ))
    ThreadCount += 1
    print('Thread Number: ' + str(ThreadCount))
ServerSocket.close()
```

## Client.py

```
#Here client can connect to the server we created in server.py file

import socket

import mysql.connector

s = socket.socket()

host = '127.0.0.1'#Here we will use the same host and port number to
connect to server

port = 1233

print('Waiting for connection')

mydb = mysql.connector.connect(host="localhost", user="root",
passwd="root")

mycursor = mydb.cursor()

def encrypt(text):

   result = ""

   s = 4

   # traverse text

   for i in range(len(text)):

       char = text[i]

       # Encrypt uppercase characters

       if (char.isupper()):

           result += chr((ord(char) + s - 65) % 26 + 65)
```

```
            # Encrypt lowercase characters
        else:
            result += chr((ord(char) + s - 97) % 26 + 97)
    return result
while(True):
    print("Enter 1 for login ")
    print("Enter 2 for register")
    print("Enter 3 for deletion")
    print("Enter 4 for updation")
    n = int(input())
    if (n == 1):
        uname = input("Enter the username")
        mycursor.execute("select userpass from netusers.connect where
username = %s", (uname,))
        passresult = mycursor.fetchone()
        mydb.commit()
        if passresult is None:
            print("User Not exists")
        else:
            pwd = passresult[0]
```

```
        userpass = input("Enter the UserPass")
        userpass = encrypt(userpass)
        if (userpass == pwd):
            # connect to the server on local computer
            s.connect(('127.0.0.1', port))
            # receive data from the server
            print(s.recv(1024))
            # close the connection
            s.close()
            print("Login successfully")
        else:
            print("Wrong Pass")
elif (n == 2):
    username = input("Enter username")
    password = input("Enter password")
    Email = input("Enter email address")
    phoneno = input("Enter phone number")
    password = encrypt(password)
    val = (username, password , Email,phoneno)
```

```
        mycursor.execute("insert into netusers.connect values(%s,%s
,%s,%s)", val)

        mydb.commit()

        print("Successfully registered")

    elif (n == 3):

        uname = input("Enter the username")

        mycursor.execute("select userpass from netusers.connect where
username = %s", (uname,))

        passresult = mycursor.fetchone()

        mydb.commit()

        if passresult is None:

            print("User Not exists")

        else:

            pwd = passresult[0]

            userpass = input("Enter the UserPass")

            userpass = encrypt(userpass)

            if (userpass == pwd):

                # connect to the server on local computer

                s.connect(('127.0.0.1', port))

                # receive data from the server

                print(s.recv(1024))
```

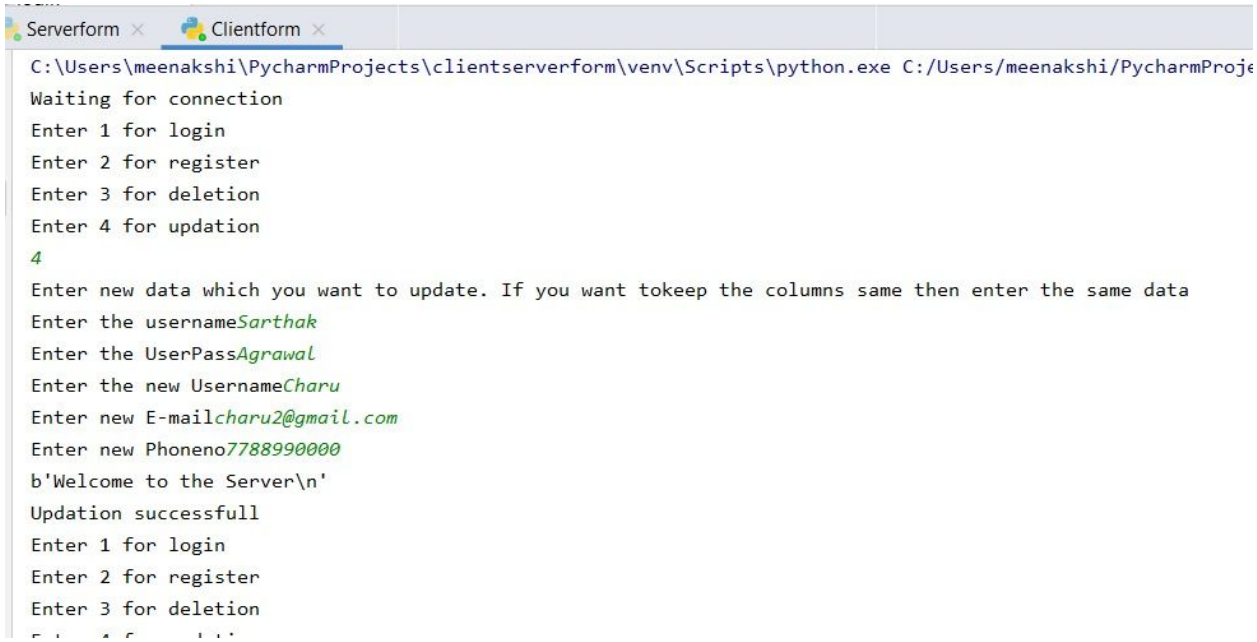
```
                mycursor.execute("DELETE FROM netusers.connect WHERE
username = %s", (uname,))

                mydb.commit()

                # close the connection

                s.close()

                print("Deletion successfull")

            else:

                print("Wrong Pass")

    elif (n == 4):

                print("Enter new data which you want to update. If you
want tokeep   the columns same then enter the same data")

                uname = input("Enter the username")

                mycursor.execute("select userpass from
netusers.connect where username = %s", (uname,))

                passresult = mycursor.fetchone()

                mydb.commit()

                if passresult is None:

                    print("User Not exists")

                else:

                    pwd = passresult[0]

                    userpass = input("Enter the UserPass")
```

```
                    userpass = encrypt(userpass)
                    if (userpass == pwd):
                        # connect to the server on local computer
                        s.connect(('127.0.0.1', port))
                        usernamenew = input("Enter the new Username")
                        Emailnew = input("Enter new E-mail")
                        Phonenonew = input("Enter new Phoneno")
                        # receive data from the server
                        print(s.recv(1024))


                        sql_update_query1 = """UPDATE netusers.connect
SET username = %s WHERE userpass = %s"""
                        inputData1 = (usernamenew, userpass)
                        mycursor.execute(sql_update_query1,
inputData1)
                        sql_update_query2 = """UPDATE netusers.connect
SET Email = %s WHERE userpass = %s"""
                        inputData2 = (Emailnew, userpass)
                        mycursor.execute(sql_update_query2,
inputData2)
                        sql_update_query3 = """UPDATE netusers.connect
SET Phoneno = %s WHERE userpass = %s"""
```

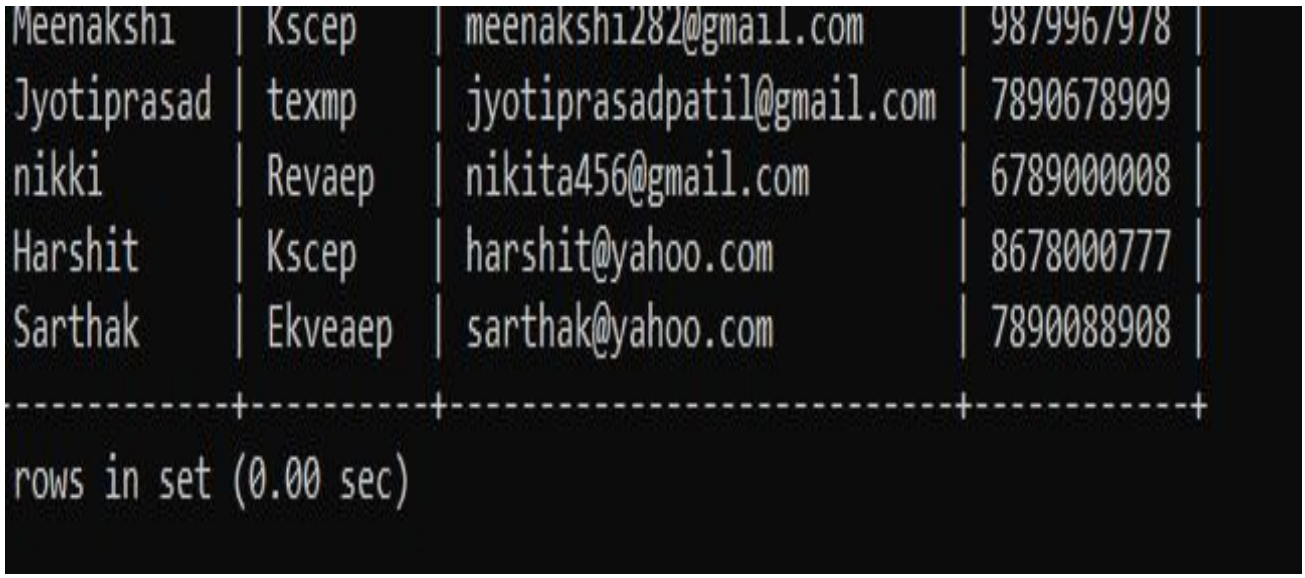
```
                    inputData3 = (Phonenonew, userpass)
                    mycursor.execute(sql_update_query3,
inputData3)
                    mydb.commit()


                    # close the connection
                    s.close()
                    print("Updation successfull")
                else:
                    print("Wrong Pass")
```

# Screenshots

## Client.py




## Database before Deletion



## Database after Deletion

```
mysql> select * from netusers.connect;
+-------------+----------+-----------------------------+------------+
| username    | userpass | Email                       | Phoneno    |
+-------------+----------+-----------------------------+------------+
| Jyotiprasad | texmp    | jyotiprasadpatil@gmail.com | 7890678909 |
| nikki       | Revaep   | nikita456@gmail.com         | 6789000008 |
| Harshit     | Kscep    | harshit@yahoo.com           | 8678000777 |
| Sarthak     | Ekveaep  | sarthak@yahoo.com           | 7890088908 |
+-------------+----------+-----------------------------+------------+
4 rows in set (0.00 sec)
```

## NewUser:

```
sql> select * from netusers.connect;
------------+----------+-----------------------------+------------+
username    | userpass | Email                       | Phoneno    |
------------+----------+-----------------------------+------------+
Meenakshi   | Kscep    | meenakshi282@gmail.com      | 9879967978 |
Jyotiprasad | texmp    | jyotiprasadpatil@gmail.com | 7890678909 |
nikki       | Revaep   | nikita456@gmail.com         | 6789000008 |
Harshit     | Kscep    | harshit@yahoo.com           | 8678000777 |
Sarthak     | Ekveaep  | sarthak@yahoo.com           | 7890088908 |
------------+----------+-----------------------------+------------+
rows in set (0.00 sec)
```

## Updation:

```
+-------------+----------+---------------------------+------------+
| username    | userpass | Email                     | Phoneno    |
+-------------+----------+---------------------------+------------+
| Jyotiprasad | texmp    | jyotiprasadpatil@gmail.com | 7890678909 |
| nikki       | Revaep   | nikita456@gmail.com       | 6789000008 |
| Harshit     | Kscep    | harshit@yahoo.com         | 8678000777 |
| Sarthak     | Ekveaep  | sarthak@yahoo.com         | 7890088908 |
+-------------+----------+---------------------------+------------+
 rows in set (0.00 sec)

ysql> select * from netusers.connect;
+-------------+----------+---------------------------+------------+
| username    | userpass | Email                     | Phoneno    |
+-------------+----------+---------------------------+------------+
| Jyotiprasad | texmp    | jyotiprasadpatil@gmail.com | 7890678909 |
| nikki       | Revaep   | nikita456@gmail.com       | 6789000008 |
| Harshit     | Kscep    | harshit@yahoo.com         | 8678000777 |
| Charu       | Ekveaep  | charu2@gmail.com          | 7788990000 |
+-------------+----------+---------------------------+------------+
 rows in set (0.00 sec)

ysql>
```

# References/Bibliography

Author:


- Belanger, F., Hiller, J. S. and Smith, W. J., (2002). Trustworthiness in electronic commerce: The role of privacy, security, and site attributes. Journal of Strategic Information Systems, 11, 245–270.
- Çelik, H., (2011). Influence of social norms, perceived playfulness and online shopping anxiety on customers' adoption of online retail shopping: An empirical study in the Turkish context. International Journal of Retail & Distribution Management, 39 (6), 390–413.
- Chang, H. H. and Chen, S. W., (2009). Consumer perception of interface quality, security, and loyalty in electronic commerce. Information & Management, 46, 411–417.
- Cheema, A. and Papatla, P., (2010). Relative importance of online versus offline information for Internet purchases: Product category and Internet experience effects. Journal of Business Research, 63, 979–985.


**Web Links:**


- www.google.com
- www.wikipedia.com
- https://codezup.com/socket-server-with-multiple-clients-model-multithreading-python/
- https://codezup.com/socket-server-with-multiple-clients-model-multithreading-python/
- https://www.w3schools.com/sql/sql_update.asp
- https://www.w3schools.com/sql/sql_update.asp