UTM

UNIVERSITI TEKNOLOGI MALAYSIA

NETWORK PROGRAMMING SEET4623
SECTION 1
SESSION 1 2022/2023

GROUP REPORT
GROUP 4
ASSIGNMENT 2: GROUP ASSIGNMENT

Lecturer's Name: Dr. Nurzal Effiyana Binti Ghazali

Name	Matrics Number
Ahmad Zidan Firmansyah	A19EE0442
Nur Ainaa Najwa Binti Razali	A19EE0383
Sivaaneshwar Murugan	A19EE0405

1. INTRODUCTION

Client-server communication, in its simplest definition, is a relationship between two programs in which one program will request a service or resource and the other program will provide it. The program that requests a service or resource is called a “client” while the program that provides the service is called a “server”. Usually, there will be one server with multiple clients such as in Figure 1.1 which shows the client-server communication model. The client can take the form of desktops, smartphones, tablets, laptops or other devices that will request a file or application from the server and the server will respond by providing the information that the client asks for. Examples of client-server applications are Email, World Wide Web and network printing.

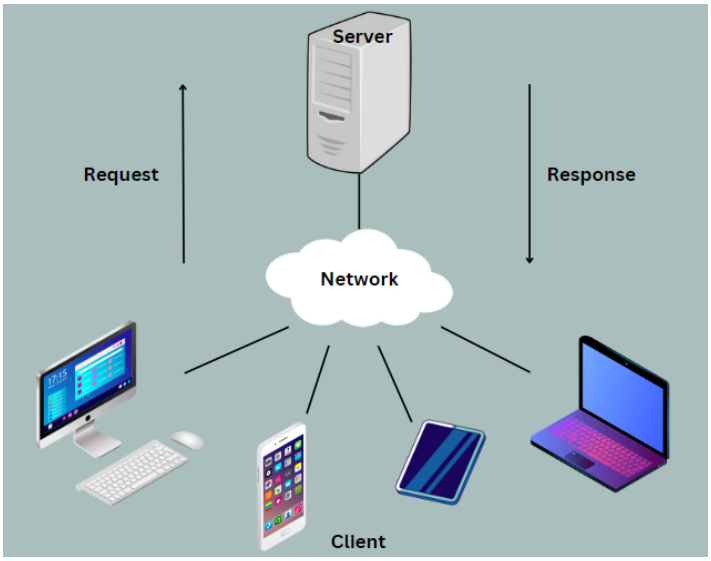


Figure 1.1: client-server communication model

The client-server network is very useful in data sharing for long distance communication. The advances in technology make it easier for the server to communicate with multiple clients at once with better speed as the technology grows. There are many advantages to this network, including centralization, scalability, easy management, accessibility and data security. Although there are a lot of advantages of the client-server network, there are still some disadvantages. The disadvantages of client-server networks are network traffic congestion, high cost, robustness, maintenance difficulty and unacquirable resources.

2. SCENARIO BACKGROUND

For the group assignment of Network Programming, the group was given a task to choose a scenario and create the coding for the chosen scenario using server-client communication. The scenario that the group chose is for the course registration for semester using a simple server-client communication. The application will allow students to register for courses offered by the university and administrators to view and manage the course registration data.

While the server is responsible for handling the registration data and serving it to the client, the client part will allow students to register for courses, view the list of available courses, and administrators to view the course registration data and manage it.

The following are the minimum requirements set by the group for the application:

1. Students should be able to register for courses.
2. Students should be able to view the list of available courses.
3. Administrators should be able to view the course registration data.
4. Administrators should be able to add, modify, and delete courses.
5. The application should validate user inputs to ensure that the data entered is valid.

3. RESULT AND DISCUSSION

a. Server Programs

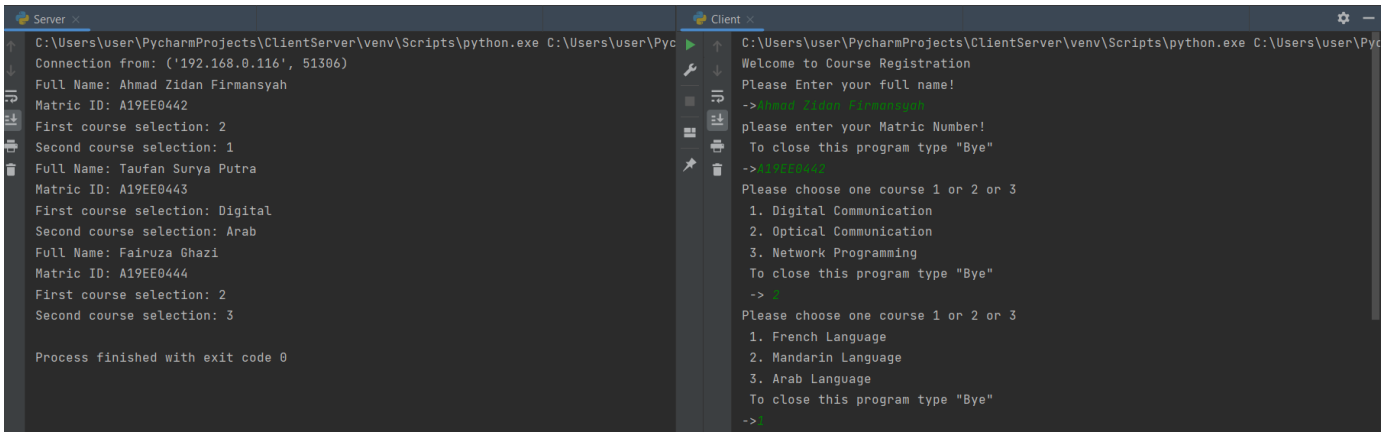
```
import socket
import sqlite3
def server_program():
    # get the hostname
    host = socket.gethostname()
    port = 5000  # initiate port no above 1024

    server_socket = socket.socket()  # get instance
    # look closely. The bind() function takes tuple as argument
    server_socket.bind((host, port))  # bind host address and port together

    # configure how many client the server can listen simultaneously
    server_socket.listen(2)
    conn, address = server_socket.accept()  # accept new connection
    print("Connection from: " + str(address))
    while True:
        # receive data stream. it won't accept data packet greater than 1024 bytes
        database = sqlite3.connect('course.db')
        cur = database.cursor()
        data = conn.recv(1024).decode()
        if not data:
            # if data is not received break
            break
        print("Full Name: " + str(data))
        Full_Name = str(data)
        data1 = conn.recv(1024).decode()
        print('Matric ID: ' + str(data1))
        data2 = conn.recv(1024).decode()
        print("First course selection: " + str(data2))
        data3 = conn.recv(1024).decode()
```

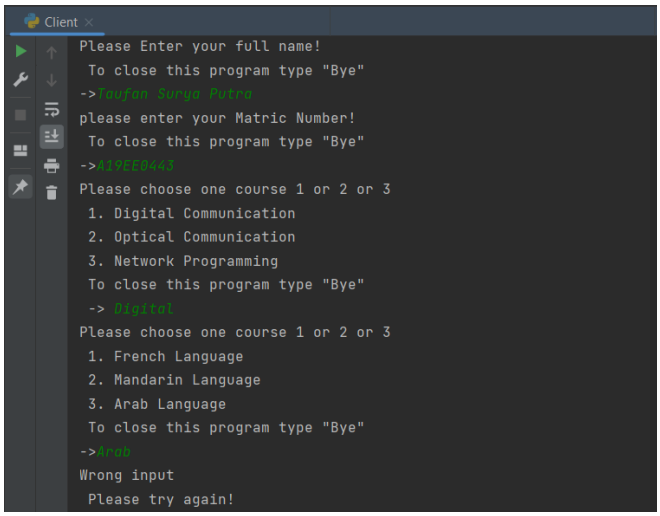
```
        print("Second course selection: " + str(data3))
        Matric = str(data1)
        First_Course = str(data2)
        Second_Course = str(data3)
        if (First_Course == '1'or First_Course =='2'or First_Course =='3') and (Second_Course == '1' or Second_Course =='2'or Second_Course =='3'):
            data = 'Correct'
            conn.send(data.encode())
            cur.execute("CREATE TABLE IF NOT EXISTS course(Full_Name text, Matric text, First_Course text, Second_Course text)");
            cur.execute("INSERT INTO course VALUES ({!r},{!r},{!r},{!r})".format(Full_Name, Matric, First_Course, Second_Course))
            database.commit()
            cur.execute("SELECT * FROM course")
        else:
            Wrong1 = 'Wrong input'
            conn.send(Wrong1.encode())
        database.close()
    conn.close()
if __name__ == '__main__':
    server_program()
```

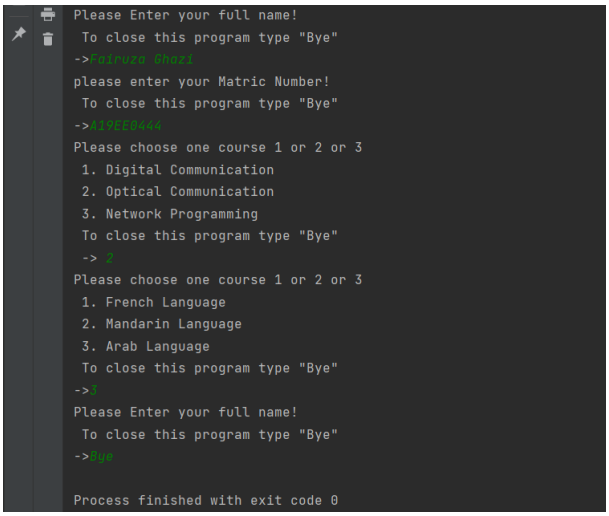
b. Client Programs

```python
import socket
def client_program():
    host = socket.gethostname()  # as both code is running on same pc
    port = 5000  # socket server port number

    client_socket = socket.socket()  # instantiate
    client_socket.connect((host, port))  # connect to the server

    print('Welcome to Course Registration')
    print('Please Enter your full name!')
    message = input("->")  # take input

    while message != 'Bye':
        client_socket.send(message.encode())  # send message
        print('please enter your Matric Number! \n To close this program type "Bye"')
        Matric = input('->')
        client_socket.send((Matric.encode()))
        print('Please choose one course 1 or 2 or 3 \n 1. Digital Communication \n 2. Optical Communication \n 3. Network Programming \n To close this program type "Bye"')
        First_Course = input(" -> ")
        client_socket.send((First_Course.encode()))
        print('Please choose one course 1 or 2 or 3 \n 1. French Language \n 2. Mandarin Language \n 3. Arab Language \n To close this program type "Bye"')
        Second_Course = input('->')
        client_socket.send((Second_Course.encode()))
        data = client_socket.recv(1024).decode()
        data = str(data)
        if data == 'Wrong input':
            print('Wrong input \n Please try again!')
            print('Please Enter your full name! \n To close this program type "Bye"')
            message = input("->")
        else:
            print('Please Enter your full name! \n To close this program type "Bye"')
            message = input("->")
    client_socket.close()  # close the connection
if __name__ == '__main__':
    client_program()
```

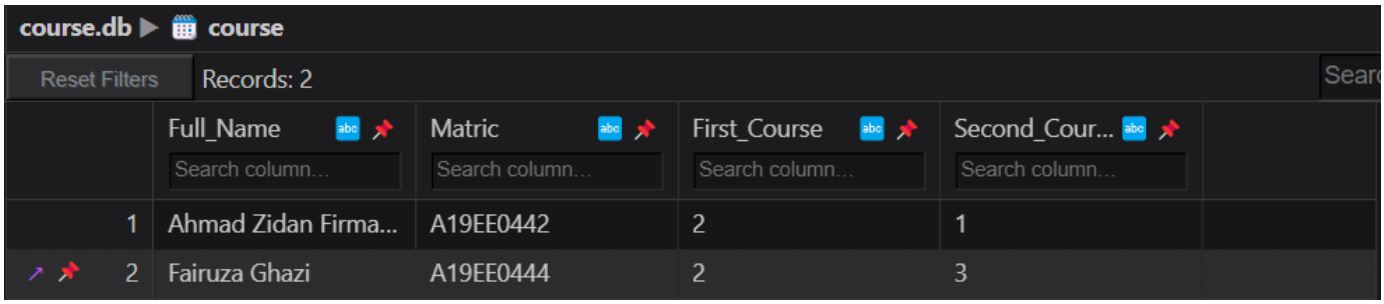
c. Output







d. Database result



As can be seen from the results above, the codes satisfy the minimum requirements provided in the scenario. By using *sqlite3.connect* function we built a database of course registration and stored it in to course.db. The wrong input also did not registered in the database, only the correct input.

4. CONCLUSION

In conclusion, we manage to create client-server communication that allows students to register for their course for that semester. The students can register for the course that they want, and the administrator can view the student application. The registration data is tabulated as in the database result (3d). The server code is created to collect the data and tabulate it while the client code is created to list down the courses offered by the university to the student. The client-server communication is working as per the scenario created by our group and it gives the output that we want. This concludes that this client-server communication is successful.

5. REFERENCES

[1] What is the Client-Server Model? - Definition from WhatIs.com. (n.d.). SearchNetworking. https://www.techtarget.com/searchnetworking/definition/client-server#:~:text=Client%2Dserver%20is%20a%20relationship

[2] How IT Works: Client Server Model | Mindsight. (2016, November 22). Https://Gomindsight.com/. https://gomindsight.com/insights/blog/works-client-server-model/#:~:text=At%2DA%2DGlance%3A%20The%20Client%20Server%20Model&text=Clients%2C%20taking%20the%20form%20of

[3] What is Client-Server? Definition and FAQs | HEAVY.AI. (n.d.). Www.heavy.ai. Retrieved February 3, 2023, from https://www.heavy.ai/technical-glossary/client-server#:~:text=Popular%20client%2Dserver%20applications%20include

[4] What is Client-Server Networking? Definition, Advantages, and Disadvantages - sunnyvalley.io. (n.d.). Www.sunnyvalley.io. https://www.sunnyvalley.io/docs/network-basics/what-is-client-server-network

[5] *Python Conditions*. (n.d.). https://www.w3schools.com/python/python_conditions.asp