### Understanding the Assignment

The assignment is about creating a **client-server application** using **TCP socket connections** to manage data for a school’s central server. The setup consists of three departments, each having a **central server and central IT administration**. The server stores data on students and employees, including:

Department number

First name

Last name

Phone number

Email address

#### Client Requests:

**Request Email Address:**

Send **first and last name** or **last name and department number** to get the **email address**.

**Request Phone Number:**

Similar approach as the email request.

**Request Employees/Students List:**

Send **department number** to get a **list of all members** in that department.

### Implementation Steps

#### 1. Setting Up the Server:

Use **TCP sockets** to listen for client connections.

Implement **multithreading** to handle multiple client requests simultaneously.

Store data in a **simple database** (like a dictionary or a small file).

Handle errors for **non-existent records**.

#### 2. Setting Up the Client:

Establish a **TCP connection** to the server.

Present a **menu** to the user with the options:

Request Email

Request Phone Number

Request List of Employees/Students

Send the appropriate request based on user input and receive the response.

#### 3. Data Handling:

Server retrieves the requested data from the central database.

If data is not found, send an **error message**.

### Tools to Help with Implementation

**Python:**

**Socket Library:** For TCP connections.

**Threading:** To manage multiple client connections.

**SQLite3 or CSV Files:** For storing the student and employee data.

**Java (alternative):**

**Java Sockets:** For TCP connections.

**Java RMI (Remote Method Invocation):** For handling distributed data requests.

**JDBC (Java Database Connectivity):** To connect to a database.

### Basic Python Implementation (Server):

import socket

import threading

# Data storage (Example)

data = {

("John", "Doe", 1): ("john.doe@example.com", "1234567890"),

("Jane", "Smith", 2): ("jane.smith@example.com", "9876543210")

}

def handle\_client(client\_socket):

try:

request = client\_socket.recv(1024).decode()

print(f"Received: {request}")

# Process request and send response

client\_socket.send(f"Processed: {request}".encode())

finally:

client\_socket.close()

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(("0.0.0.0", 9999))

server.listen(5)

print("Server listening on port 9999")

while True:

client, addr = server.accept()

print(f"Connection from {addr}")

client\_handler = threading.Thread(target=handle\_client, args=(client,))

client\_handler.start()

### Basic Python Implementation (Client):

import socket

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect(("127.0.0.1", 9999))

request = input("Enter your request: ")

client.send(request.encode())

response = client.recv(4096).decode()

print(f"Server Response: {response}")

client.close()

### Documentation

#### Abstract

This project implements a client-server application using **TCP socket programming** to retrieve student and employee data. The system supports multiple client connections simultaneously through multithreading.

#### Introduction

The school’s central server manages data for three departments. Students and employees can request data like email, phone numbers, or a list of department members.

#### Implementation

The server is implemented using **Python sockets** and supports multithreading for handling multiple connections. Data is stored in a simple data structure (like a dictionary or database).

#### Discussion

**Advantages of Sockets + Threads:**

Real-time communication between client and server.

Ability to handle multiple connections without blocking.

Scalability for multiple users.

#### Conclusion

The client-server application efficiently manages data retrieval with real-time responses. The use of TCP sockets and threading makes it suitable for a multi-client environment.

Let me know if you need further assistance with the code or documentation!  
For a better experience and more solutions, you can also check out [HIX.AI](https://hix.ai/chat), the best ChatGPT alternative.