



# Group26 : GatherSphere

---

## I. Project Overview

### A. Project Introduction

GatherSphere is a seamless virtual communication platform that connects people in real-time. Presenters and audiences can create rooms, join discussions, and share ideas effortlessly. With UDP for video and audio and TCP for reliable text sharing, GatherSphere ensures a smooth and engaging experience. It's your gateway to interactive, community-driven engagement.

### B. Group member

- 111550105 郭宗信 : 負責撰寫 Client 端程式碼以及控制專題的進度以及報告撰寫。
- 111550050 方品仁 : 負責撰寫 Room Server 端程式碼以及專案的發想
- 111550079 李承晏 : 負責撰寫 central Server 端程式碼以及 User Interface 的美編

### C. Development Environment

## Operating System 作業系統

- Linux Ubuntu 22.04 LTS
  - Native installation
  - Virtual Machine (VMware/VirtualBox)

## Compiler Details

- C++ Version: C++11
- Optimization Level: O2
- Warning Level: -Wall (enable all warnings)

## D. System Requirements

### Required External Libraries

- OpenCV 4
- PortAudio
- pkg-config
- POSIX Socket Library



*These libraries need to load before compiled or execute.*

## II. Research Methodology & Design

### Room Server

1. 初始化階段
  - 創建多個socket

- 綁定不同埠號
- 準備接收用戶連線

## 2. 執行階段 ( `run()` )

- 啟動音訊串流 `thread`
- 啟動視訊串流 `thread`
- 進入用戶連線和訊息處理主迴圈

## 3. 用戶連線處理流程

- 監聽主要連線 `socket`
- 接收新用戶
- 驗證用戶身份(主持人/觀眾)
- 將用戶加入線上清單
- 更新文件描述符集合

## 4. 訊息處理流程

- 持續監聽所有已連線用戶
- 接收用戶傳送的文字訊息
- 廣播訊息給其他線上用戶
- 處理用戶斷線情況

## 5. 多媒體串流流程

- 音訊執行緒：接收並廣播音訊
- 視訊執行緒：接收並廣播視訊
- 將主持人的串流發送給所有觀眾

## 6. 結束流程

- 偵測主持人離線
- 通知所有用戶
- 關閉所有連線
- 結束執行緒

# Central Server

## 1. 初始化

- 創建socket
- 綁定IP和埠號
- 準備監聽客戶端連線

## 2. 主要運行迴圈

- 持續監聽並接受客戶端連線
- 每次接收到連線後，處理不同的命令類型

## 3. 支援的命令類型：

- 創建房間 (CREATE\_ROOM)
  - 產生新的房間資料
  - 傳回房間資訊給客戶端
  - 使用`fork()`創建房間伺服器
- 列出房間 (LIST\_ROOM)
  - 檢查當前所有房間的存活狀態
  - 篩選出仍然存活的房間
  - 將房間列表傳回給客戶端
- 加入房間 (JOIN\_ROOM)
  - 目前程式碼中尚未實作完全

## 4. 房間管理

- 使用`room_id_counter`追蹤房間編號
- 為每個房間分配獨立的埠號
- 使用`fork()`為每個房間創建獨立的伺服器進程

## 5. 錯誤處理

- 捕捉信號(SIGCHLD)以處理子進程
- 處理socket連線和通訊可能的錯誤情況

## Client

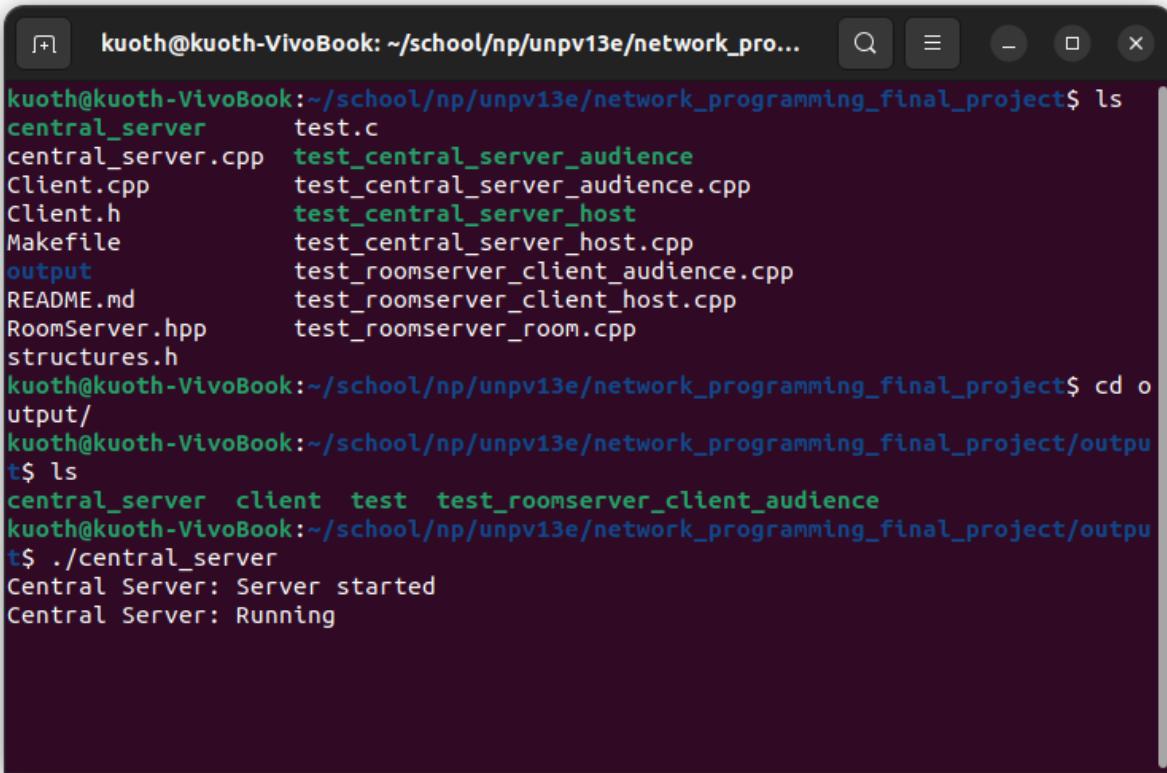
1. 程式啟動時，要求使用者輸入伺服器IP位址
  - 如果啟動時沒有直接提供IP，會要求手動輸入
2. 建立Client物件，連接到中央伺服器
  - 呼叫 `Connect_central_server()` 建立socket連線
3. 要求使用者輸入使用者名稱
  - 將名稱儲存在 `username` 變數中
4. 進入中央伺服器選單(`Central_loop()`)
  - 顯示選單選項：
    - A. 建立新房間
    - B. 列出所有房間
    - C. 加入現有房間
    - Q. 退出
5. 根據使用者選擇執行不同功能：
  - 建立房間：輸入房間名稱，發送創建請求
  - 列出房間：向伺服器請求房間列表並顯示
  - 加入房間：選擇要加入的房間編號
  - 退出：結束程式
6. 加入房間後
  - 連線切換到特定房間伺服器
  - 進入房間訊息迴圈(`Room_loop()`)
  - 可以發送和接收訊息
7. 訊息處理
  - 使用 `Handle_message()` 同時處理接收和發送訊息

- 使用 `select()` 監聽socket和標準輸入
- 

## III. Results

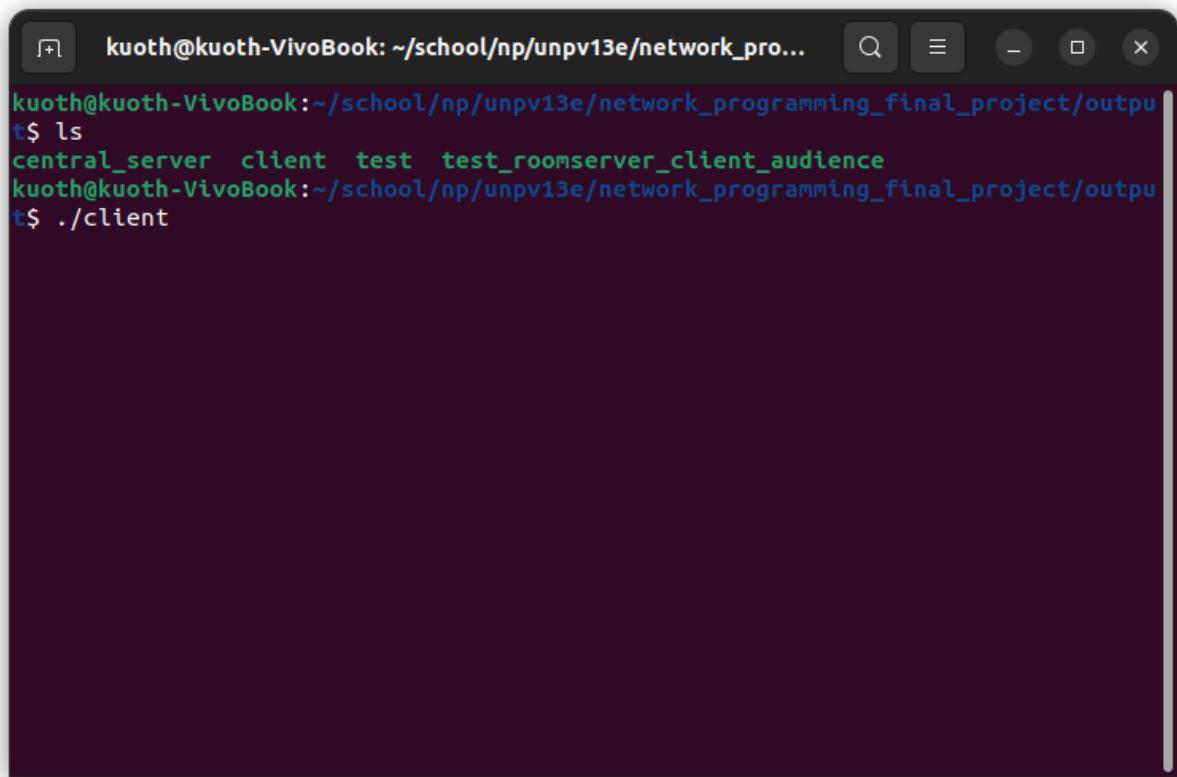
最後成果的主要功能與特色。

1. *Central Server Room Server* 架設好後client 可以藉由輸入 ‘./client *IP addr*’ 輸入 name，若沒有預先輸入 *IP addr* 可以在進入程式之後再輸入即可。
- 建立 Central Server 後 Room Server 會被 Central Server 建立，並且上面會有相關 狀態訊息



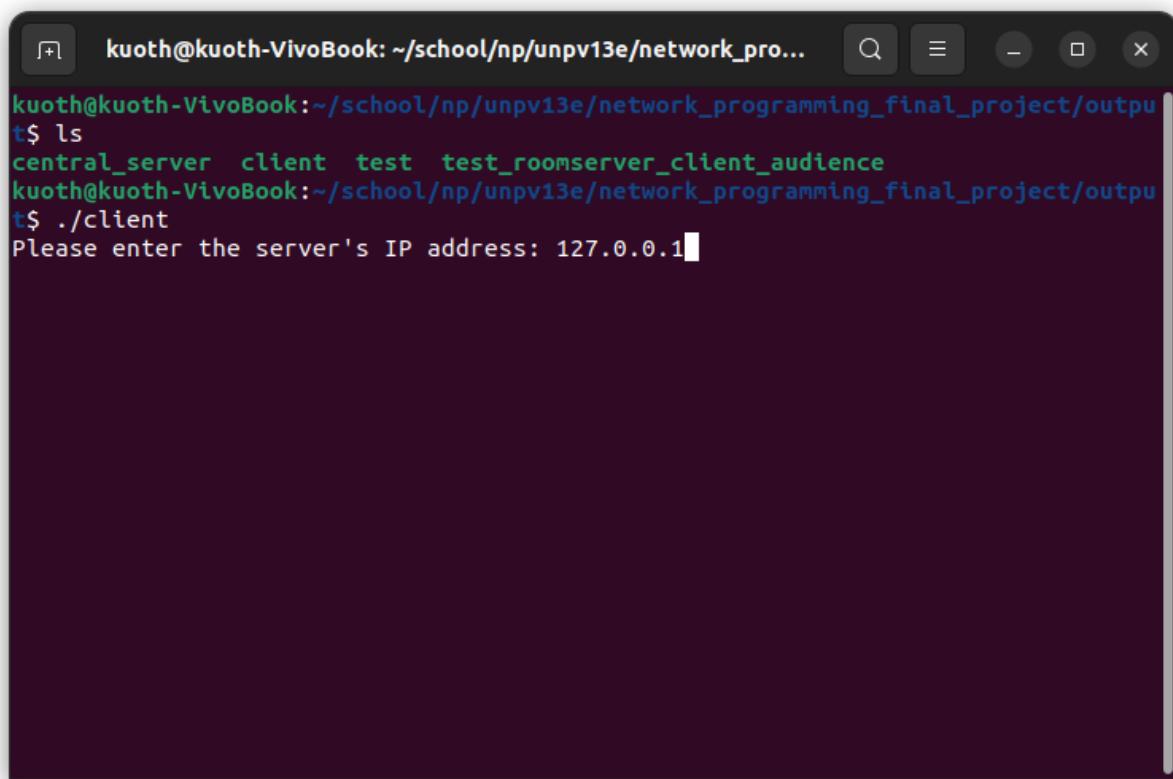
```
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project$ ls
central_server      test.c
central_server.cpp  test_central_server_audience
Client.cpp          test_central_server_audience.cpp
Client.h            test_central_server_host
Makefile            test_central_server_host.cpp
output              test_roomserver_client_audience.cpp
README.md           test_roomserver_client_host.cpp
RoomServer.hpp      test_roomserver_room.cpp
structures.h
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project$ cd output/
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project/output$ ls
central_server  client  test  test_roomserver_client_audience
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project/output$ ./central_server
Central Server: Server started
Central Server: Running
```

- 建立 Client 可以先輸入 *IP addr* 或在啟動程式後才輸入，連線後即可輸入所要用的 ID name



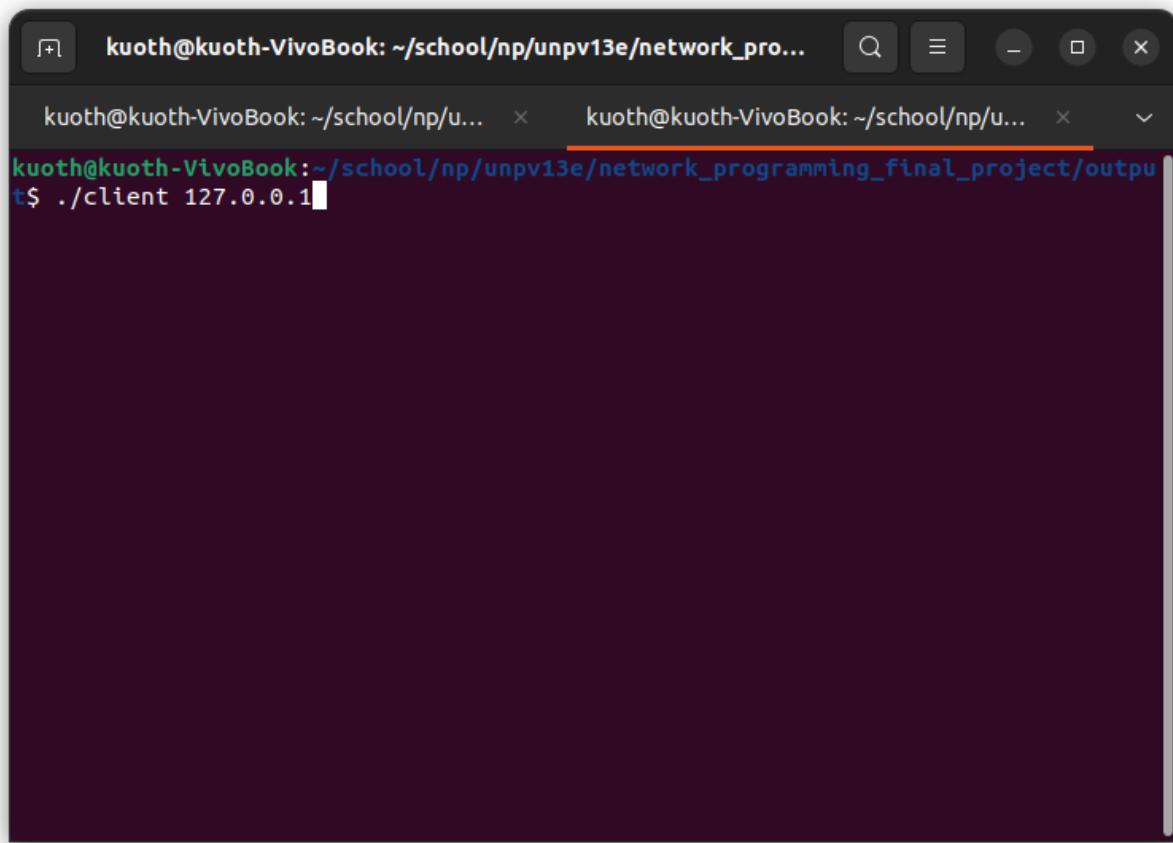
A screenshot of a terminal window titled "kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network\_programming\_final\_project/output". The window shows the following command-line session:

```
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project/output
$ ls
central_server client test test_roomserver_client_audience
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project/output
$ ./client
```

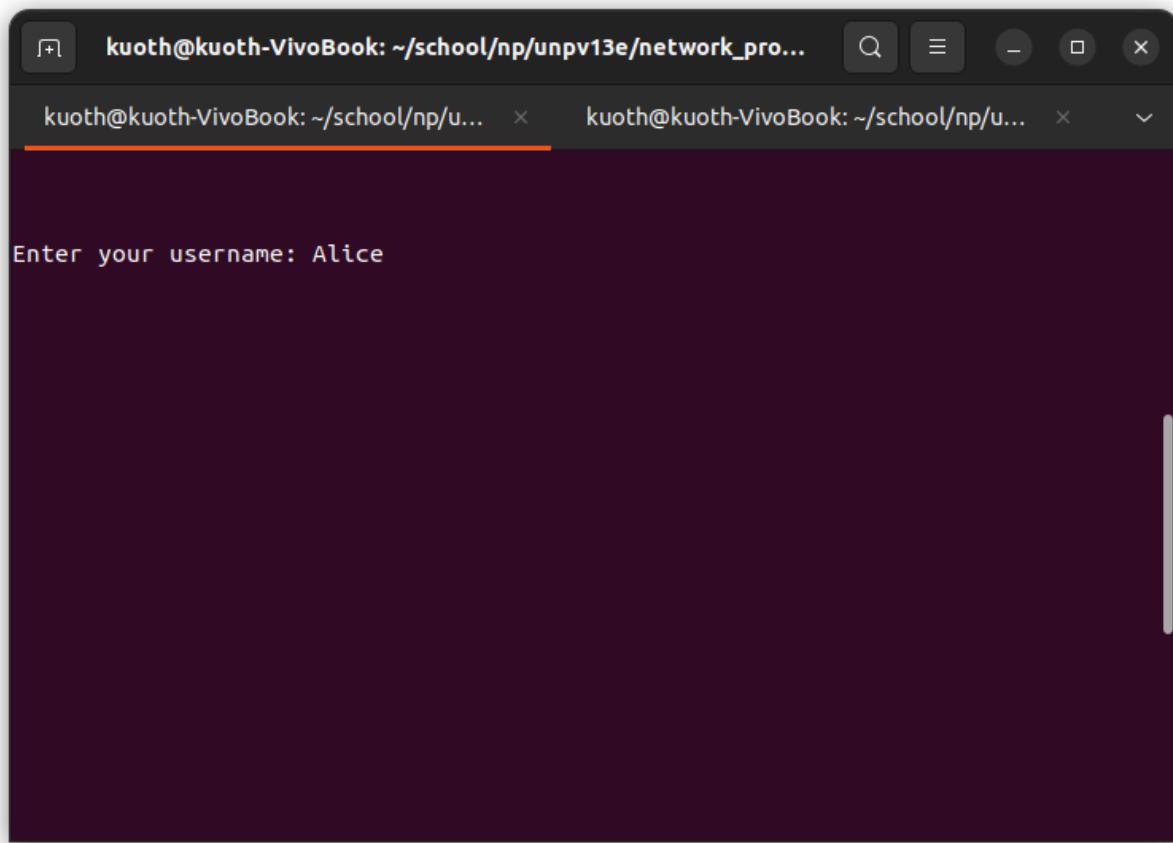


kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network\_programming\_final\_project/outputs

```
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project/outputs
t$ ls
central_server  client  test  test_roomserver_client_audience
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project/outputs
t$ ./client
Please enter the server's IP address: 127.0.0.1
```

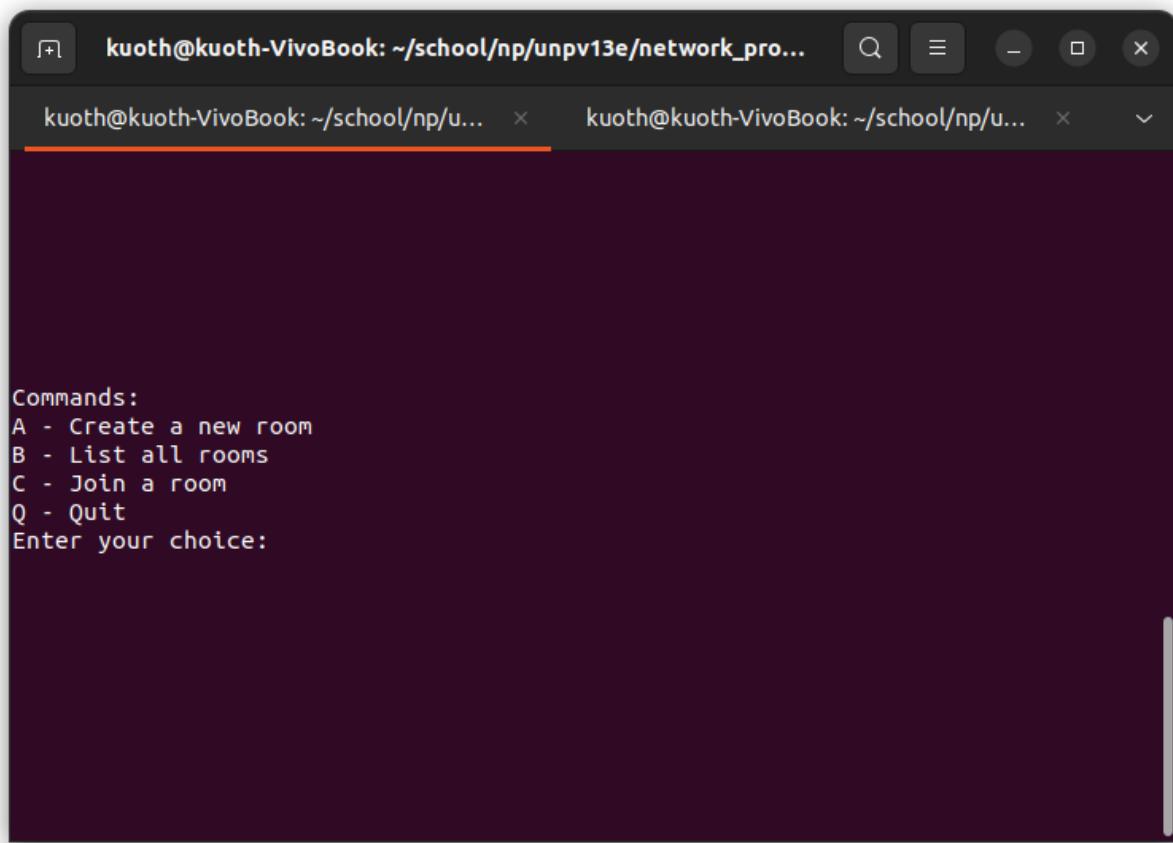


A screenshot of a terminal window with a dark background and light-colored text. The terminal is titled "kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network\_pro...". It contains two tabs, both titled "kuoth@kuoth-VivoBook: ~/school/np/u...". The active tab shows the command "kuoth@kuoth-VivoBook:~/school/np/unpv13e/network\_programming\_final\_project/output\$ ./client 127.0.0.1" being typed into the terminal. The command is partially visible at the bottom of the screen.



A screenshot of a terminal window titled "kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network\_pro...". The window contains two tabs, both showing the command "kuoth@kuoth-VivoBook: ~/school/np/u...". The terminal is dark-themed. The text "Enter your username: Alice" is displayed on the screen.

2. 可以提供選單給 client 決定要做什麼動作(此部分與central server 互動)



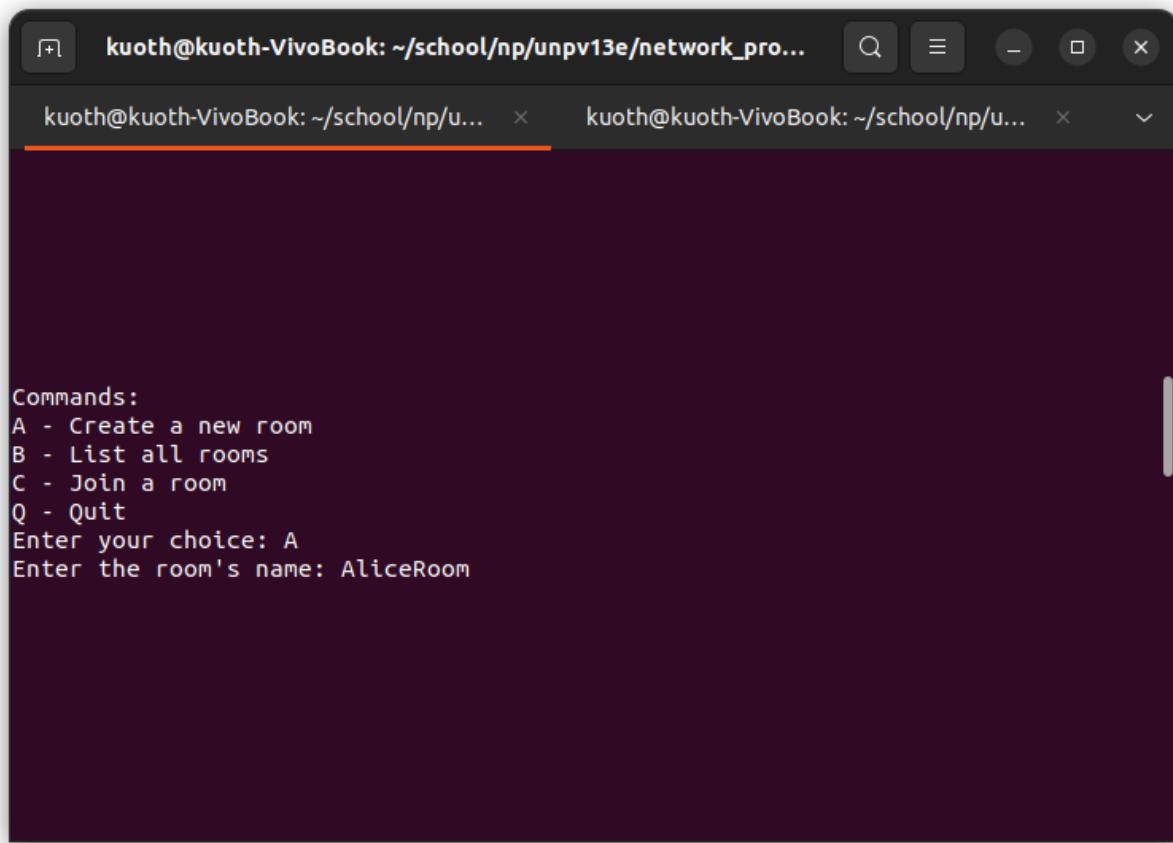
```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro...
kuoth@kuoth-VivoBook: ~/school/np/u... x kuoth@kuoth-VivoBook: ~/school/np/u... x v

Commands:
A - Create a new room
B - List all rooms
C - Join a room
Q - Quit
Enter your choice:
```

- 顯示選單選項：

- A. 建立新房間
- B. 列出所有房間
- C. 加入現有房間
- Q. 退出

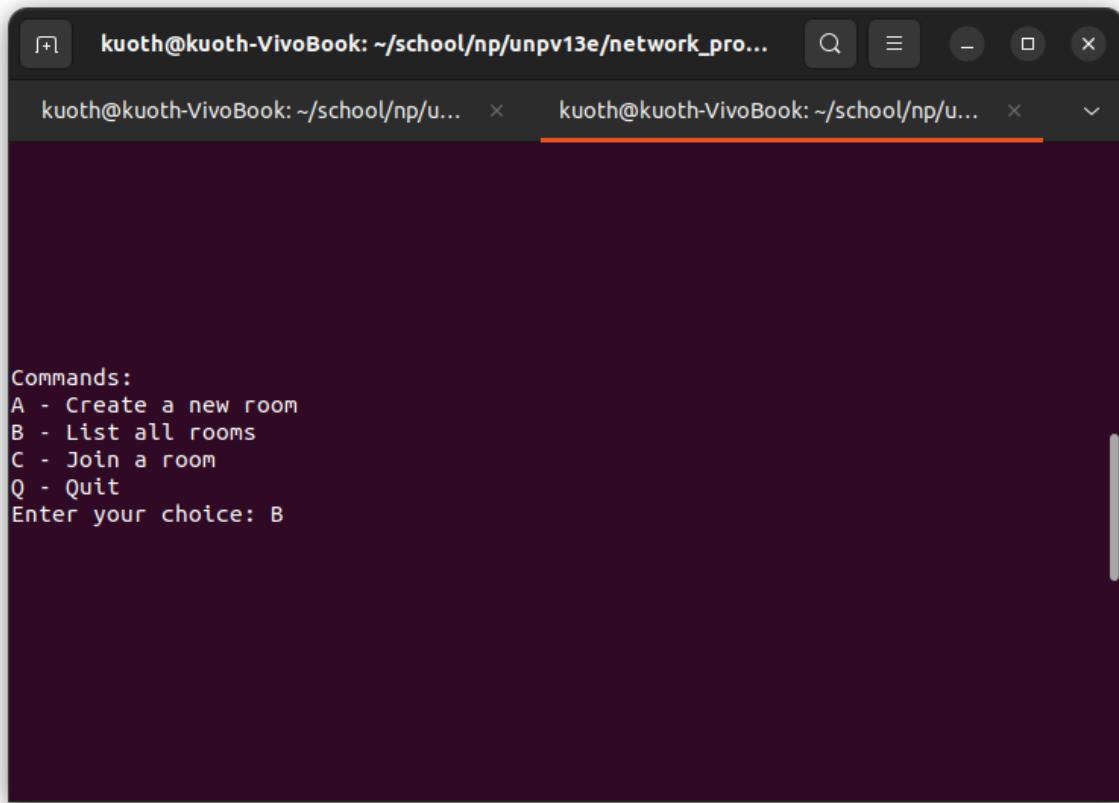
### 3. 可以自己建立新的 Room



```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro...
kuoth@kuoth-VivoBook: ~/school/np/u... x kuoth@kuoth-VivoBook: ~/school/np/u... x

Commands:
A - Create a new room
B - List all rooms
C - Join a room
Q - Quit
Enter your choice: A
Enter the room's name: AliceRoom
```

3. 可以列出所有已經存在的 Room 以及舉辦者的名子並且可以輸入 id 後加入



```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro...  kuoth@kuoth-VivoBook: ~/school/np/u...  kuoth@kuoth-VivoBook: ~/school/np/u...
Commands:
A - Create a new room
B - List all rooms
C - Join a room
Q - Quit
Enter your choice: B
```

```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro...  kuoth@kuoth-VivoBook: ~/school/np/u...  kuoth@kuoth-VivoBook: ~/school/np/u...  ↴
```

```
[Client][Debug] Request_room_list(): Received response
room_id: 1
room_name: AliceRoom
running_port:10008
host_user:
name: Alice
identity: 2

#      Room Name      Host
-----
 0      AliceRoom      Alice

Commands:
A - Create a new room
B - List all rooms
C - Join a room
Q - Quit
Enter your choice:
```

```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro...  kuoth@kuoth-VivoBook: ~/school/np/u...  kuoth@kuoth-VivoBook: ~/school/np/u...  ↴

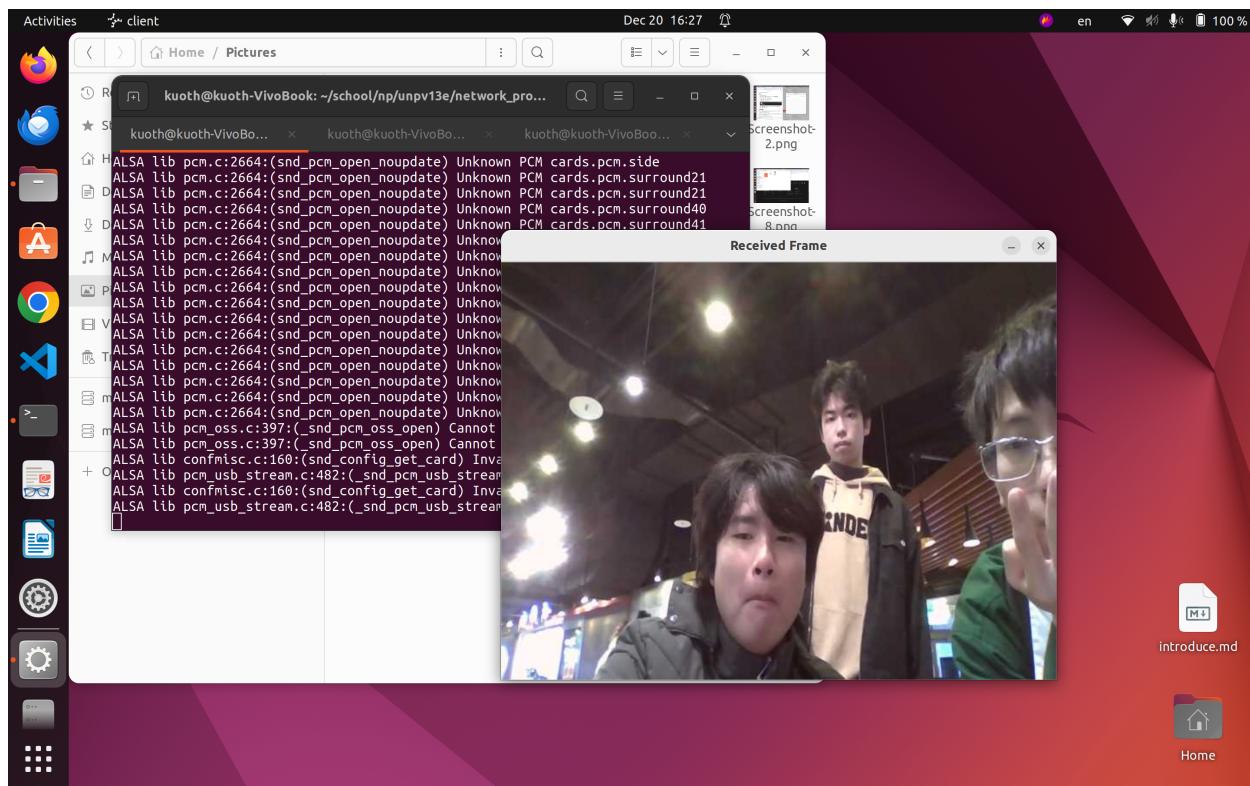
Commands:
A - Create a new room
B - List all rooms
C - Join a room
Q - Quit
Enter your choice: C
[Client][Debug] Request_room_list(): Received response
room_id: 1
room_name: AliceRoom
running_port:10008
host_user:
name: Alice
identity: 2

#      Room Name      Host
-----
0      AliceRoom      Alice

Please enter the room number you want to join:0
```

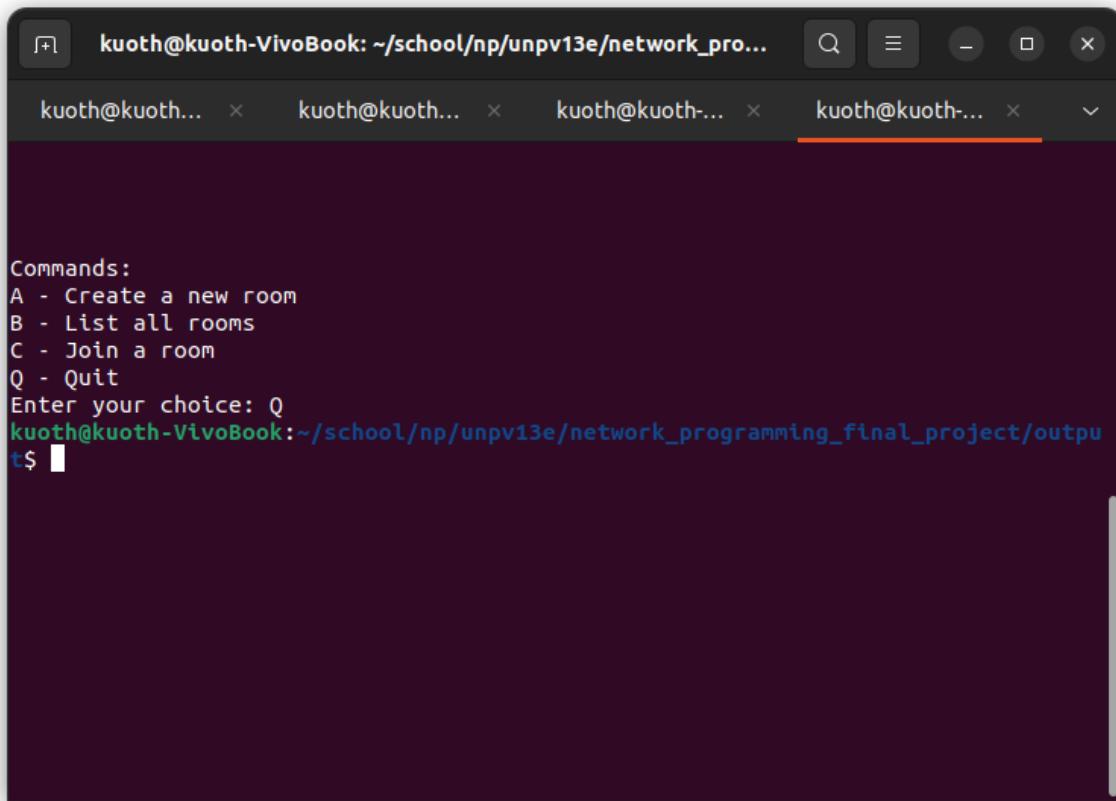
4. 當選擇加入的的room 後即可開始進行視訊與文字的交流

```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro...  kuoth@kuoth-VivoBo...  kuoth@kuoth-VivoBo...
kuoth@kuoth-VivoBo...  kuoth@kuoth-VivoBo...  kuoth@kuoth-VivoBo...
(Alice)
(Alice)
(Alice)
(Alice)
(Alice) hi
(CC)
(CC)
(Alice) hello
(Alice) hi
(Alice) my name is bob
no you are alice
```



[https://prod-files-secure.s3.us-west-2.amazonaws.com/99afef0-48af-4055-98f4-9e29c90f7b05/912c2ce6-cc27-4799-9fd6-7e6a5746c46e/Screen cast\\_from\\_01-01-2025\\_110218\\_PM.webm](https://prod-files-secure.s3.us-west-2.amazonaws.com/99afef0-48af-4055-98f4-9e29c90f7b05/912c2ce6-cc27-4799-9fd6-7e6a5746c46e/Screen cast_from_01-01-2025_110218_PM.webm)

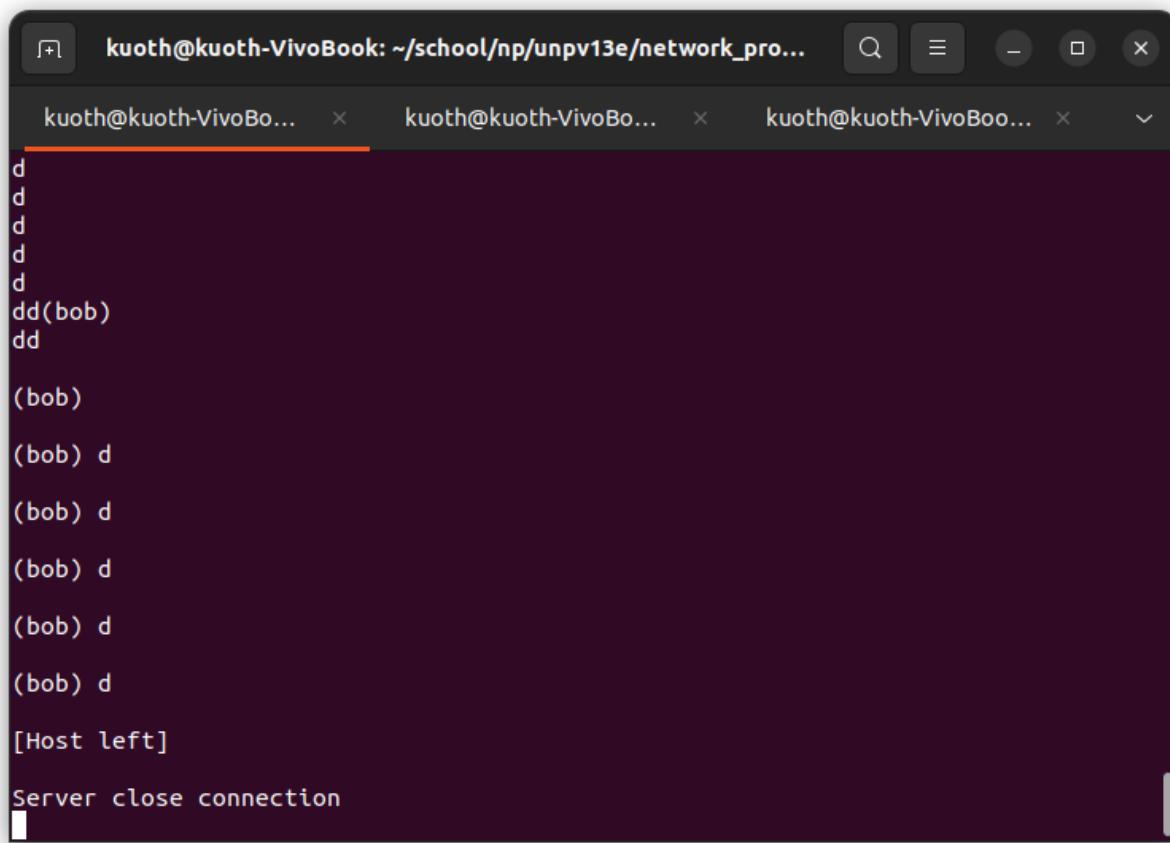
## 5. 可以選擇離開退出離開主程式（主選單）



```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro...
kuoth@kuoth... x  kuoth@kuoth... x  kuoth@kuoth... x  kuoth@kuoth-... x  ▾

Commands:
A - Create a new room
B - List all rooms
C - Join a room
Q - Quit
Enter your choice: Q
kuoth@kuoth-VivoBook:~/school/np/unpv13e/network_programming_final_project/output
t$
```

## 6. 當直播提供者離開後房間內的其他人可以知道，此時直播室就會關閉



```
kuoth@kuoth-VivoBook: ~/school/np/unpv13e/network_pro... d
d
d
d
d
d
dd(bob)
dd

(bob)
(bob) d
(bob) d
(bob) d
(bob) d
(bob) d
(bob) d

[Host left]

Server close connection
```

## IV. Conclusion

### 專題製作心得

在本次報告中，我們花費了許多時間進行協調、討論與合作，透過彼此的努力，才得以在期限內完成最終的成果。特別是在段考週期間，除了準備考試，還要抽出時間進行專題製作，確實是一項困難且充滿挑戰的任務。

我們這次選擇的主題是一個通訊/社交性質的專題，目標是讓多人能夠同時連線並進行互動。當初選擇這個主題，是因為聽過老師的講解後，發現過去尚未有人做過類似的專題，再加上我們運用了電腦上的攝影機與麥克風，這樣的技術整合讓我們覺得非常酷炫且具有挑戰性。雖然網路上已經有許多成熟的應用，例如 Google Meet 等，但我們希望能將人

與人面對面的即時互動感，轉換為透過電腦連線也能達成的效果，這樣的想法激發了我們的創意。

在專題的設計上，我們將各項工作進行了分層與抽象化，並透過兩個 Server 來處理訊息交換的邏輯。在這個過程中，我們深刻體會到流程規劃的重要性。此外，我們也接觸到許多課堂上未曾學習的新技術，例如如何使用攝影機和麥克風、如何透過 OpenCV 處理影像，以及使用 PortAudio 解決音訊處理的問題。

然而，在使用這些新技術時，我們發現除了老師教過的基本功能（如 Socket 連接與訊息傳遞）外，還有一些更複雜的部分需要整合，例如多執行緒（Thread）的同步處理，才能順利同步影像與聲音。此外，在觀念與設計上，我們也整合了這學期所學的作業系統知識，並運用過去學習的物件導向程式設計 (OOP) 概念。同時，透過網路查詢資料以及撰寫報告的過程，讓我們對相關函式庫有了更深入的理解，也讓我們體會到 C++ 擁有如此強大的功能與彈性。

透過本次專題，我們結合了所學知識、創意，以及專題中獲得的新技術，成功完成了最終的成果——**GatherSphere**。這次經驗不僅加深了我們對程式設計的理解，也提升了我們解決問題的能力，是一次非常有收穫的學習旅程。

## 遭遇困難及解決經過

在本次專題中，由於 Server 和 Client 的程式碼是由不同成員分工負責，因此在對接過程中，我們面臨了許多挑戰，最主要的困難在於 **訊息傳遞的順序與格式**。這樣的情況使我們不僅需要專注於撰寫自己負責的程式碼，還必須投入大量時間來檢查彼此之間的訊息傳送內容與邏輯，並透過溝通和測試來確保雙方的程式能夠正確對接。

在初期開發階段，即便針對單獨的 **Client** 或 **Server** 進行了測試，並未發現明顯的問題，但當真正將兩者對接時，卻頻繁地出現 **segmentation fault**。這類問題往往難以直接定位，經過反覆測試後，我們發現原因主要是 **Client 傳遞的資訊格式** 與 **Server 預期的格式** 不一致，導致伺服器在解析接收到的資料時出錯，進而引發異常。為了排查這些錯誤，我們對 Client 端傳送的資訊進行了詳細的分析，逐一檢查每個傳遞的封包內容與格式，最終找到了格式錯誤的部分，並進行了修正。

此外，由於我們的系統需要處理大量的訊息交流，這也引發了另一個問題：**資源競爭**。在多執行緒環境下，特定共享變數（例如 `ready_to_end`）在多個執行緒同時存取時可能會產生競爭狀況，導致系統行為不穩定或非預期的結果。為了解決這個問題，我們引入了

**mutex lock**，將關鍵變數的存取操作加以保護，確保只有一個執行緒能夠修改該變數，從而避免資源競爭的發生。

在實作的過程中，我們還發現 `recvfrom` 函式會導致執行緒被阻塞的問題。由於 `recvfrom` 是一個 **阻塞式函式**，當沒有接收到資料時，執行緒會卡在該函式呼叫中，這導致我們的 **Room Server** 在某些情況下無法正常關閉，因為執行緒無法繼續往下執行。為了解決這個問題，我們為 `recvfrom` 設置了 **timeout (超時機制)**，讓它在一定時間內沒有接收到資料時會自動返回，並繼續執行下一個迴圈。這樣一來，即使在沒有傳入資料的情況下，執行緒仍然可以進行其他任務，確保伺服器能夠正常關閉或執行其他操作。也不會有資源競爭的情況。

## 成果未來改進或延伸方向

未來如果要上線，我們希望可以在 central server 連結一個資料庫來管理帳號相關的事宜，例如提供註冊、登入以及第三方登入功能。這部分可以使用 SQL 資料庫如 MySQL 或 PostgreSQL，並透過加密技術來保障帳號資訊的安全性。同時，介面設計方面也希望能夠有更好的視覺呈現，目前的 UI 風格過於簡陋，未來可以設計成更友好的圖形介面，例如使用 C++ 的 Qt 庫來實現更美觀的 UI。此外，這次的專題僅能讓單一使用者輸出影像畫面，其他使用者只能觀看，未來希望能夠讓所有使用者都可以同時傳送自己的鏡頭畫面與聲音，實現真正的多人雙向影音通訊。

另外，我們目前並未測試整體對網路頻寬的需求，未來可以透過網路分析工具如 Wireshark 來監測網路流量，並根據結果進行優化，例如透過 H.264 壓縮視訊與 Opus 音訊編碼來降低頻寬消耗。透過壓縮與動態調整畫質，讓系統能支援更多使用者同時在線。若使用人數大量增加，系統架構也需要進行擴展，可以將中央伺服器拆分成多個子伺服器，並引入負載平衡機制來分散流量，確保系統運行的穩定性。

此外，我們也可以加入一些額外功能，例如文字聊天室、主持人權限管理、螢幕分享以及視訊錄製功能，讓使用者有更多元的互動體驗。這次的專題讓我們看到了系統設計與實現的潛力，未來可以透過這些優化與擴展，將其發展成一個更完整的多人通訊平台。

最後是一些小地方可以做優化，像是可以增加更多可能發生的狀況，例如直播者突然離開可能會導致觀眾還會卡在直播間，之後就可以設計再回到主選單中，另外就是如果觀眾想要跳離也可以有方法跳離直播間。

# V. References And Appendices

Git hub link : [GitHub\\_link](#)