**LẬP TRÌNH MẠNG – NHÓM 1: CHỦ ĐỀ: LẬP TRÌNH ỨNG DỤNG CHÁT**

A screenshot of a computer

AI-generated content may be incorrect.

# **A. KHO LƯU TRỮ DỰ ÁN NHÓM**

[**https://github.com/omniBaodzai/LTM-Chat-App-Python.git**](https://github.com/omniBaodzai/LTM-Chat-App-Python.git)

# **B. BẢNG PHÂN CÔNG CÔNG VIỆC**

[**https://docs.google.com/spreadsheets/d/1h8pPIczWtqtbiracHqnIhuI\_Dm6yTaLmnSB7GHPUR3U/edit?hl=vi&pli=1&gid=1924277268#gid=1924277268**](https://docs.google.com/spreadsheets/d/1h8pPIczWtqtbiracHqnIhuI_Dm6yTaLmnSB7GHPUR3U/edit?hl=vi&pli=1&gid=1924277268#gid=1924277268)

# **C. NỘI DUNG HỌP NHÓM**

## **I. TÌM HIỂU VỀ LẬP TRÌNH ỨNG DỤNG CHAT (1 – 1, 1 – N) VỚI NNLT PYTHON**

### 1. HIỂU SƠ LƯỢC: CHAT 1 – 1 VÀ CHAT 1 – N LÀ GÌ?

* **Chat 1 – 1**: Giống như bạn gửi tin nhắn riêng cho một người bạn trên Facebook hoặc Zalo.
* **Chat 1 – N**: Là khi một người gửi tin nhắn cho nhiều người cùng lúc, giống như trong nhóm Zalo, Messenger hay một thông báo gửi đến nhiều người.

### 2. PYTHON DÙNG ĐỂ LÀM GÌ TRONG ỨNG DỤNG CHAT?

Python có thể được dùng để:

* Tạo **giao diện người dùng đơn giản** (nơi bạn gõ và xem tin nhắn).
* Viết **server và client** để gửi và nhận tin nhắn.
* Quản lý **nhiều người dùng**, kết nối, và giao tiếp giữa họ.

### 3. NHỮNG CÔNG NGHỆ PYTHON ĐƠN GIẢN THƯỜNG DÙNG ĐỂ LÀM CHAT APP

|  |  |
| --- | --- |
| Thành phần | Công cụ Python sử dụng |
| Giao diện người dùng (UI) | tkinter hoặc PyQt (nếu cần có cửa sổ hiển thị) |
| Giao tiếp mạng | socket (thư viện có sẵn trong Python) |
| Quản lý đa người dùng | threading (để chạy nhiều kết nối song song) |
| Server lưu trữ tạm tin nhắn | Có thể dùng tạm bộ nhớ hoặc lưu tệp (nâng cao hơn có thể dùng cơ sở dữ liệu như SQLite hoặc MongoDB) |

### 4. ỨNG DỤNG CHAT CƠ BẢN HOẠT ĐỘNG THẾ NÀO?

**1. Phía người dùng (client)**

* Gửi tin nhắn đến **server**.
* Nhận và hiển thị tin nhắn từ **server**.

**2. Phía máy chủ (server)**

* Nhận tin nhắn từ người dùng A.
* Gửi tin nhắn đó đến người dùng B (trong 1-1) hoặc cả nhóm (trong 1-N).

### 5. VÍ DỤ: CHAT ĐƠN GIẢN 1 – 1 BẰNG PYTHON

Bạn có thể tưởng tượng như sau:

* Một người chạy **server**.
* Hai người còn lại chạy **client** để kết nối và trò chuyện với nhau qua server.

**🧱 Cấu trúc chương trình**

Chúng ta sẽ có **2 phần**:

1. **Server (máy chủ)** – dùng để trung gian truyền tin nhắn.
2. **Client (máy khách)** – 2 người dùng chạy chương trình client để kết nối với server và gửi/nhận tin nhắn.

**📥 Bước 1: Tạo server**

**📄 server.py**

**import socket**

**HOST = '127.0.0.1'**

**PORT = 12345**

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

**server\_socket.bind((HOST, PORT))**

**server\_socket.listen()**

**print(f"🟢 Server đang chạy tại {HOST}:{PORT}, chờ kết nối...")**

**conn, addr = server\_socket.accept()**

**print(f"✅ Kết nối từ: {addr}")**

**while True:**

**data = conn.recv(1024).decode()**

**if not data:**

**break**

**print(f" Client: {data}")**

**msg = input("👨‍💻 Bạn: ")**

**conn.send(msg.encode())**

**conn.close()**

**server\_socket.close()**

**📤 Bước 2: Tạo client**

**📄 client.py**

**import socket**

**HOST = '127.0.0.1'**

**PORT = 12345**

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

**client\_socket.connect((HOST, PORT))**

**print("📲 Đã kết nối đến server.")**

**while True:**

**msg = input("👨‍💻 Bạn: ")**

**client\_socket.send(msg.encode())**

**data = client\_socket.recv(1024).decode()**

**print(f" Server: {data}")**

**client\_socket.close()**

**🚀 Cách chạy chương trình**

1. **Mở 1 cửa sổ Terminal, di chuyển đến ổ đĩa chứa file,**

- Giả sử đang ở ổ C:

A black screen with white text

AI-generated content may be incorrect.

- Muốn chuyển sang ổ D (nơi chứa 2 tệp server.py và client.py) thì gõ: D: rồi nhất enter

A screenshot of a computer

AI-generated content may be incorrect.

- Nếu nó nằm ở thư mục nào đó thì gõ: cd + đường dẫn rồi nhấn enter

A black screen with white text

AI-generated content may be incorrect.

1. **chạy server.py:**

**python server.py**

1. **Mở 2 cửa sổ Terminal khác, mỗi cửa sổ chạy client.py:**

**python client.py**

1. Giờ bạn có thể chat giữa 2 cửa sổ client!

**✅ Ghi chú**

* Đây là mô hình **chat qua LAN (nội bộ)**.
* Bạn có thể mở rộng thêm để:
  + Thêm tên người dùng.
  + Lưu lịch sử chat.
  + Làm giao diện đẹp hơn.
* **thêm giao diện đơn giản (GUI) và chuyển sang gửi qua mạng thật (Internet)**

## **II. CẤU HÌNH GIT**

**git config --global user.name "omniBaodzai"**

**git config --global user.email "email\_cua\_ban@gmail.com"**

Mục đích là để **Git biết bạn là ai** khi bạn thực hiện các hành động như **commit** (lưu lại thay đổi) trong các dự án. Nói đơn giản:

* user.name: là tên sẽ xuất hiện trên mỗi lần bạn commit.
* user.email: là email liên kết với tài khoản GitHub của bạn, để khi bạn push code lên GitHub, nó sẽ gắn bạn là người đã làm điều đó.

**Muốn kiểm tra lại thì gõ:**

**git config --global user.name**

**git config --global user.email**

**Muốn xóa thông tin Git đã cấu hình thì dễ lắm luôn á, cậu chỉ cần dùng lệnh sau để xóa từng cấu hình:**

**🧹 Xóa từng mục:**

**git config --global --unset user.name**

**git config --global --unset user.email**

Sau khi xóa, Git sẽ **không còn biết bạn là ai** mỗi lần bạn commit nữa — nó sẽ yêu cầu bạn nhập lại hoặc báo lỗi nếu thiếu thông tin.

## **III. TẠO THƯ MỤC CHỨA PROJECT (bỏ qua mục III)**

- Trên ổ D:

+) Tạo một thư mục mới

+) Đặt tên: LTM-ChatApp-Python

## **IV. KÉO REPOSITORY (KHO LƯU TRỮ DỰ ÁN) VỀ THƯ MỤC VỪA TẠO TRONG MÁY TÍNH**

### 💻 BƯỚC 1: MỖI THÀNH VIÊN CLONE REPO VỀ MÁY

**# Clone repo vào ổ D, hoặc bất cứ đâu trên máy**

**cd D:**

**git clone https://github.com/omniBaodzai/LTM-Chat-App-Python.git**

→ Git sẽ tạo thư mục LTM-Chat-App-Python chứa toàn bộ mã nguồn từ GitHub (hiện tại mới có README.md).

### 🚶 BƯỚC 2: DI CHUYỂN VÀO THƯ MỤC DỰ ÁN

**cd LTM-Chat-App-Python**

### 🌿 BƯỚC 3: TẠO NHÁNH RIÊNG CHO TỪNG THÀNH VIÊN

**🎯 Ví dụ nhóm 5 người:**

|  |  |  |
| --- | --- | --- |
| Thành viên | Chức năng | Tên nhánh |
| Bảo | Server | server-bao |
| Nguyên | Server | server-nguyen |
| Băng | Client | client-bang |
| Uyên | Client | client-uyen |
| Vũ | Client | client-uyen |

**git checkout -b ten-nhanh-cua-ban**

Ví dụ:

**git checkout -b server-trung**

### ✅ BƯỚC 4: GHI LẠI THAY ĐỔI

**git add .**

**git commit -m "Thêm code xử lý server cơ bản"**

- Nhớ là có khoảng trắng rồi mới gõ dấu chấm nha.

- Trong cặp ngoặc kép: Ai làm xong chức năng hay nội dung gì thì ghi rõ theo cú pháp:

**“ngay/thang/nam - Hoàn thành - nội dung mà mình đã hoàn thành”**

Ví dụ:

**“06/05/2025 – Cập nhật lần 1 - Giao diện trò chuyện”**

**“07/05/2025 – Cập nhật lần 2 - Giao diện trò chuyện”**

**“08/05/2025 – Hoàn thành - Giao diện trò chuyện”**

### ☁️ BƯỚC 5: PUSH NHÁNH LÊN GITHUB

**git push -u origin ten-nhanh-cua-ban**

Ví dụ:

**git push -u origin server-trung**

→ Trên GitHub sẽ xuất hiện nhánh mới chứa code riêng của bạn.

### 👀 BƯỚC 6: KIỂM TRA CÁC NHÁNH TRÊN GITHUB

Vào đây nè:  
👉 <https://github.com/omniBaodzai/LTM-Chat-App-Python/branches>

### 🧪 BƯỚC 7: SAU KHI HOÀN THÀNH CHỨC NĂNG – TẠO PULL REQUEST (PR)

1. Vào GitHub repo.
2. Chọn nhánh của bạn.
3. Nhấn nút **“Compare & Pull Request”**.
4. Nhập nội dung PR, rồi nhấn **“Create Pull Request”**.

→ Cả nhóm hoặc admin xem, kiểm tra và merge vào main.

**🎁 Gợi ý phân chia thư mục**

Cậu có thể gợi ý cả nhóm làm cấu trúc như sau:

A screenshot of a computer

AI-generated content may be incorrect.

### LƯU Ý KHI ĐỔI TÊN NHÁNH

Nếu cậu hoặc một bạn nào đó **lỡ đặt sai tên nhánh** và muốn **đổi lại tên**, thì Git hỗ trợ làm điều đó rất dễ dàng nha! Mình hướng dẫn cụ thể cả **trên máy (local)** và **trên GitHub (remote)** luôn nè:

**✅ Trường hợp 1: Đổi tên nhánh hiện tại đang đứng trên máy**

**🌱 Giả sử bạn đang ở nhánh server-trugn (bị sai chính tả), muốn đổi thành server-trung:**

# 1. Đổi tên nhánh hiện tại

**git branch -m server-trung**

**-m là viết tắt của --move, dùng để đổi tên.**

**✅ Trường hợp 2: Đã push nhánh sai tên lên GitHub rồi, giờ muốn sửa luôn trên GitHub**

**🪜 Làm theo 4 bước:**

**# 1. Đổi tên nhánh trên local**

**git branch -m server-trung**

**# 2. Xóa nhánh cũ trên GitHub (vì nó sai)**

**git push origin --delete server-trugn**

**# 3. Đẩy nhánh mới lên GitHub**

**git push origin server-trung**

**# 4. Thiết lập nhánh mới thành nhánh theo dõi từ xa (tracking)**

**git push --set-upstream origin server-trung**

**📌 Ghi nhớ:**

|  |  |
| --- | --- |
| Lệnh | Tác dụng |
| git branch -m TênMới | Đổi tên nhánh hiện tại |
| git push origin --delete ... | Xóa nhánh trên GitHub |
| git push origin TênMới | Đẩy nhánh mới lên GitHub |
| git push --set-upstream ... | Liên kết nhánh local với GitHub |

**⚠️ Lưu ý quan trọng:**

* Nếu **chưa push nhánh sai tên lên GitHub** thì chỉ cần git branch -m là đủ!
* Nếu **đã push rồi** thì làm thêm bước xóa nhánh cũ trên GitHub và đẩy nhánh mới.

### TÌNH HUỐNG:

* ✅ **Bạn A** hôm nay đã code xong, rồi:

**git add .**

**git commit -m "Thêm giao diện chat"**

**git push origin gui-linh**

* ❌ **Bạn B** hôm nay nghỉ.
* ✅ **Ngày mai**, bạn B mở máy → cần **lấy code mới nhất** từ GitHub repo về để tiếp tục.

**✅ Vậy bạn B cần làm gì?**

Tùy theo bạn B đang làm việc ở **nhánh nào**, sẽ có 2 trường hợp:

**▶️ Trường hợp 1: Bạn B muốn cập nhật code từ main hoặc một nhánh có sẵn trên GitHub**

**👉 Các bước:**

**# 1. Di chuyển vào thư mục repo**

**cd LTM-Chat-App-Python**

**# 2. Lấy toàn bộ thông tin mới nhất từ GitHub**

**git fetch**

**# 3. Cập nhật nhánh mình đang đứng (ví dụ đang ở nhánh 'main')**

**git pull origin main**

✅ Nếu bạn B đang ở nhánh client-hoa, thì sửa main thành client-hoa

**▶️ Trường hợp 2: Bạn B muốn xem/cập nhật 1 nhánh mới do bạn A vừa tạo**

Giả sử bạn A hôm qua tạo nhánh gui-linh, bạn B chưa từng thấy.

**👉 Làm như sau:**

**# 1. Fetch tất cả thông tin về**

**git fetch**

**# 2. Kiểm tra danh sách nhánh có trên GitHub**

**git branch -r**

**# 3. Tạo nhánh local từ nhánh trên GitHub**

**git checkout -b gui-linh origin/gui-linh**

Bây giờ bạn B đã có thể xem hoặc làm tiếp trên nhánh gui-linh.

**🧊 Tóm gọn cho nhóm:**

|  |  |
| --- | --- |
| Mục đích | Câu lệnh |
| Lấy mọi thứ mới nhất từ GitHub | git fetch |
| Cập nhật nhánh hiện tại | git pull origin <ten-nhanh> |
| Xem các nhánh trên GitHub | git branch -r |
| Lấy nhánh mới do người khác tạo | git checkout -b <nhánh-local> origin/<nhánh-trên-GitHub> |

Tối t2, t4,t6

Muốn xóa nội dung vừa đẩy lên github thì làm sao nhỉ? btech@MyLoveFamily MINGW64 / $ cd D btech@MyLoveFamily MINGW64 /D $ cd D:\LTM-Chat-App-Python btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (main) $ git checkout -b client-bao Switched to a new branch 'client-bao' btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (client-bao) $ git add client.py btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (client-bao) $ git commit -m "11-05-2025 - Hoàn thành - Client" [client-bao 6b48dba] 11-05-2025 - Hoàn thành - Client 1 file changed, 17 insertions(+) create mode 100644 client.py btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (client-bao) $ git push -u origin client-bao Enumerating objects: 4, done. Counting objects: 100% (4/4), done. Delta compression using up to 8 threads Compressing objects: 100% (3/3), done. Writing objects: 100% (3/3), 573 bytes | 286.00 KiB/s, done. Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0) remote: Resolving deltas: 100% (1/1), completed with 1 local object. remote: remote: Create a pull request for 'client-bao' on GitHub by visiting: remote: https://github.com/omniBaodzai/LTM-Chat-App-Python/pull/new/client-bao remote: To https://github.com/omniBaodzai/LTM-Chat-App-Python.git \* [new branch] client-bao -> client-bao branch 'client-bao' set up to track 'origin/client-bao'. btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (client-bao) $

**ChatGPT đã nói:**

Để **xóa nội dung vừa đẩy lên GitHub** (cụ thể là file client.py và commit mới trên nhánh client-bao), bạn có một số lựa chọn tùy theo mục đích:

**🔁 1. Nếu bạn chỉ muốn xóa file khỏi GitHub nhưng giữ nhánh:**

Bạn có thể xóa file client.py, commit lại và đẩy thay đổi:

bash

Sao chépChỉnh sửa

git rm client.py

git commit -m "Xóa file client.py"

git push origin client-bao

**🧹 2. Nếu bạn muốn xóa hoàn toàn commit vừa rồi khỏi GitHub (quay lại trước khi push):**

⚠️ Cách này sẽ **xóa lịch sử commit** đã đẩy lên GitHub, không nên dùng nếu người khác đã pull hoặc làm việc trên nhánh này.

bash

Sao chépChỉnh sửa

git reset --hard HEAD~1

git push origin client-bao --force

HEAD~1 tức là quay lại commit trước đó. --force cần thiết vì bạn đang ghi đè lịch sử trên GitHub.

**🔥 3. Nếu bạn muốn xóa luôn nhánh client-bao khỏi GitHub (xóa cả code và nhánh):**

**Trên Git:**

bash

Sao chépChỉnh sửa

git push origin --delete client-bao

**Nếu muốn xóa cả nhánh local:**

bash

Sao chépChỉnh sửa

git branch -D client-bao

Bạn muốn giữ lại nhánh nhưng xóa file, hay xóa luôn nhánh và toàn bộ nội dung đã đẩy lên?

**Bạn đã nói:**

như này nghĩa là sao? btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (client-bao) $ git push origin --delete client-bao To https://github.com/omniBaodzai/LTM-Chat-App-Python.git - [deleted] client-bao btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (client-bao) $ git branch -D client-bao error: cannot delete branch 'client-bao' used by worktree at 'D:/LTM-Chat-App-Python'

**ChatGPT đã nói:**

Câu lệnh và kết quả của bạn có nghĩa như sau:

**✅ git push origin --delete client-bao:**

Bạn đã **xóa thành công nhánh client-bao khỏi GitHub** (tức là không còn tồn tại trên GitHub nữa).

**❌ git branch -D client-bao và lỗi:**

go

Sao chépChỉnh sửa

error: cannot delete branch 'client-bao' used by worktree at 'D:/LTM-Chat-App-Python'

Lỗi này xảy ra vì **bạn đang đứng trên chính nhánh client-bao**, nên Git không cho phép xóa nhánh đó trong local.

**✅ Cách xử lý:**

1. **Chuyển sang nhánh khác** (ví dụ main):

bash

Sao chépChỉnh sửa

git checkout main

1. **Sau đó mới xóa nhánh client-bao trong local**:

bash

Sao chépChỉnh sửa

git branch -D client-bao

Bạn có muốn tạo lại nhánh mới sạch sẽ sau khi xóa không?

**Bạn đã nói:**

có

**ChatGPT đã nói:**

Tuyệt! Để tạo lại một nhánh mới hoàn toàn sạch sẽ (không có những commit cũ), bạn làm theo các bước sau:

**✅ Bước 1: Chuyển về nhánh chính (main)**

bash

Sao chépChỉnh sửa

git checkout main

**✅ Bước 2: Tạo nhánh mới từ main**

Ví dụ, đặt tên nhánh là client-bao như cũ:

bash

Sao chépChỉnh sửa

git checkout -b client-bao

Lúc này nhánh client-bao hoàn toàn sạch, giống hệt main tại thời điểm hiện tại.

**✅ Bước 3: (Tuỳ chọn) Làm lại từ đầu**

Thêm lại file hoặc commit mới nếu bạn muốn:

bash

Sao chépChỉnh sửa

git add client.py

git commit -m "Thêm lại client.py sau khi làm sạch nhánh"

git push -u origin client-bao

Nếu bạn muốn chắc chắn là nhánh này **không còn gì từ commit cũ**, bạn có thể kiểm tra bằng:

bash

Sao chépChỉnh sửa

git log

Bạn cần mình hỗ trợ kiểm tra lại nội dung commit hoặc file không?

Lỗi gì đây nhỉ? btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (main) $ git add . btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (main) $ git commit -m "12-05-2025 - Cập nhật lần 01 - Tài liệu họp nhóm" [main b98cb56] 12-05-2025 - Cập nhật lần 01 - Tài liệu họp nhóm 3 files changed, 0 insertions(+), 0 deletions(-) delete mode 100644 "~$i dung h\341\273\215p nh\303\263m LTM-ChatApp-Python.docx" btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (main) $ git push -u origin main To https://github.com/omniBaodzai/LTM-Chat-App-Python.git ! [rejected] main -> main (fetch first) error: failed to push some refs to 'https://github.com/omniBaodzai/LTM-Chat-App-Python.git' hint: Updates were rejected because the remote contains work that you do not hint: have locally. This is usually caused by another repository pushing to hint: the same ref. If you want to integrate the remote changes, use hint: 'git pull' before pushing again. hint: See the 'Note about fast-forwards' in 'git push --help' for details. btech@MyLoveFamily MINGW64 /d/LTM-Chat-App-Python (main)

**ChatGPT said:**

Lỗi bạn gặp là do nhánh main trên GitHub đã có **cập nhật mới** mà bạn **chưa có trong máy local**, vì vậy khi bạn git push, Git từ chối vì **nó không muốn bạn ghi đè lên lịch sử hiện có trên remote**.

**Tóm tắt lỗi:**

pgsql

CopyEdit

Updates were rejected because the remote contains work that you do not have locally.

**Cách khắc phục:**

**✅ Cách an toàn (nên dùng): Kéo về trước rồi đẩy lên**

bash

CopyEdit

git pull --rebase origin main

git push -u origin main

* git pull --rebase: đồng bộ và đặt commit của bạn "trên cùng" của thay đổi từ remote (giữ lịch sử sạch hơn).
* Sau đó bạn push lại là được.

**❗Trường hợp bạn *chắc chắn* không cần những thay đổi trên GitHub (không khuyến khích):**

bash

CopyEdit

git push -f origin main

* ⚠️ Lưu ý: **-f (force push)** sẽ **ghi đè toàn bộ** lịch sử trên GitHub bằng phiên bản local của bạn — chỉ dùng nếu bạn chắc chắn bạn không cần các commit từ remote!

Bạn có muốn mình kiểm tra giúp có gì thay đổi trên GitHub chưa được pull về không?