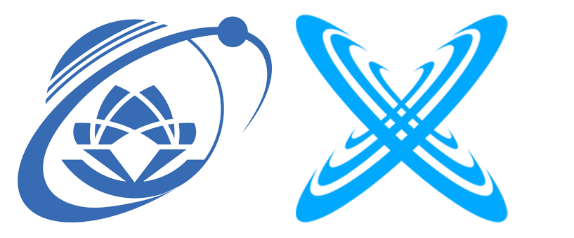
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

MÔN LẬP TRÌNH MẠNG CĂN BẢN



**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**Tên đồ án:** Dreaw - Ứng dụng vẽ chung qua mạng

**Link repository của đồ án:** [**Hac1es/NT106-DoAn: Dreaw - Ứng dụng vẽ chung qua mạng Internet**](https://github.com/Hac1es/NT106-DoAn)

**Nhóm thực hiện:** Nhóm 12

Nguyễn Viết Tùng 23521746 Trưởng nhóm

Ông Vũ Minh Thành 23521458 Thành viên

Lý Trung Thiện 23521479 Thành viên

Nguyễn Minh Thông 23521525 Thành viên

**Giảng viên hướng dẫn:** Ths. Đỗ Thị Hương Lan

**Thời gian thực hiện:** 9/2024 – 12/2024

*Thành phố Hồ Chí Minh, ngày 30 tháng 12 năm 2024*

**LỜI CẢM ƠN**

*Các thành viên trong nhóm 12 muốn gửi lời cảm ơn chân thành sâu sắc tới giảng viên hướng dẫn Ths. Đỗ Thị Hương Lan đã tận tình hỗ trợ nhóm để hoàn thành đồ án này. Qua đồ án này, nhóm đã tích lũy được nhiều kiến thức bổ ích về chuyên môn cũng như rèn luyện kĩ năng làm việc nhóm.*

*Mặc dù các thành viên trong nhóm đã cố gắng hoàn thành đồ án một cách hoàn thiện nhất nhưng do thiếu sót về kinh nghiệm cũng như một số hạn chế khác nên đồ án này cũng không thể tránh khỏi các sai sót. Nhóm rất mong nhận được sự cảm thông, chia sẻ và góp ý từ quý thầy cô cũng như các bạn sinh viên.*

*Nhóm 12 xin chân thành cảm ơn!* **TÓM TẮT**

**Tên đồ án:** Dreaw - Ứng dụng vẽ chung qua mạng

Đồ án này tập trung vào xây dựng một ứng dụng desktop có tên là Dreaw, nhằm giúp người sử dụng có thể làm việc nhóm, trao đổi ý tưởng và xây dựng những dự án chung theo thời gian thực trong môi trường lập trình mạng căn bản.

Ứng dụng hỗ trợ tính năng tạo tài khoản, đăng nhập và cấp lại mật khẩu qua email. Người dùng có thể tạo phòng vẽ chung, mời bạn bè vào hợp tác hay tham gia vào phòng vẽ của người khác. Ngoài ra, phòng vẽ sẽ được lưu trên cơ sở dữ liệu để người dùng có thể truy cập và tiếp tục dự án của mình.

Dreaw sẽ sử dụng các nguyên tắc và kỹ thuật lập trình mạng căn bản như giao thức TCP/IP, cơ chế đăng nhập, giao tiếp client-server và các yêu cầu bảo mật cơ bản để đảm bảo an toàn và bảo mật thông tin trong quá trình truyền tải dữ liệu.

Dreaw = Draw + Dream. Được trang bị các công cụ vẽ mạnh mẽ, giao diện thân thiện và tích hợp nhiều tính năng cộng tác trực tuyến, Dreaw là nơi người dùng cùng nhau hiện thực hóa giấc mơ của mình.

**Draw together. Dream together.**

Mục lục

[Chương 1: Tổng quan 5](#_Toc186314480)

[I. Giới thiệu đề tài 5](#_Toc186314481)

[II. Tổng quan các công nghệ được sử dụng 5](#_Toc186314482)

[Chương 2: Cơ sở dữ liệu 7](#_Toc186314483)

[Chương 3: Các vấn đề về đăng nhập/đăng ký 9](#_Toc186314484)

[I. Luồng của ứng dụng 9](#_Toc186314485)

[II. Logic tính năng: 10](#_Toc186314486)

[Chương 4: Trang cá nhân 20](#_Toc186314487)

[I. Tính năng tạo phòng 20](#_Toc186314488)

[II. Tính năng tham gia vào phòng 21](#_Toc186314489)

[III. Tham gia phòng từ danh sách phòng vẽ: 22](#_Toc186314490)

[IV. Bộ điều khiển API (RoomController) 23](#_Toc186314491)

[Chương 5: Tính năng vẽ chung thời gian thực 27](#_Toc186314492)

[I. Logic vẽ 27](#_Toc186314493)

[II. Tính năng vẽ: 29](#_Toc186314494)

[Chương 6: Giao tiếp và đồng bộ hóa dữ liệu thời gian thực sử dụng SignalR 34](#_Toc186314495)

[**I.** Công nghệ và các tính năng 34](#_Toc186314496)

[**II.** Logic tính năng 34](#_Toc186314497)

[Chương 7: Phân công công việc 40](#_Toc186314498)

[Chương 8: Ý tưởng về Multi-server 41](#_Toc186314500)

[Tài liệu tham khảo 42](#_Toc186314501)

# Chương 1: Tổng quan

## Giới thiệu đề tài

Dreaw là một ứng dụng desktop hỗ trợ người dùng trình bày ý tưởng thông qua các nét vẽ và tính năng trò chuyện (chat). Ứng dụng được thiết kế nhằm thúc đẩy sự hợp tác nhóm trong thời gian thực, giúp các thành viên dễ dàng chia sẻ và phát triển ý tưởng một cách trực quan và sáng tạo.

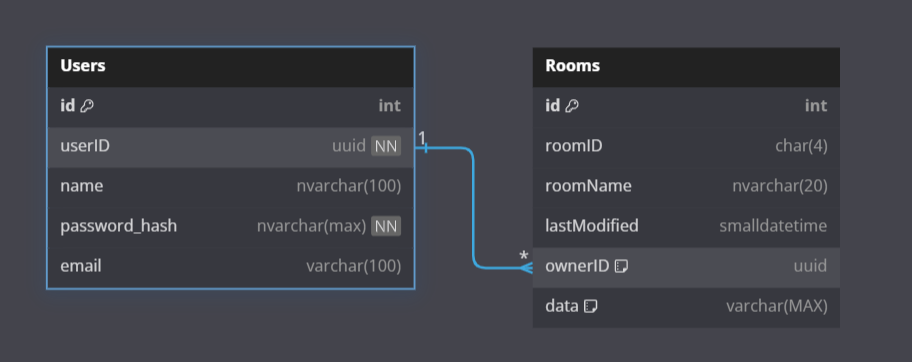
## Tổng quan các công nghệ được sử dụng

* **C# .NET (công nghệ chính):** Một ngôn ngữ lập trình mạnh mẽ do Microsoft phát triển, cung cấp thư viện phong phú để làm việc với giao thức TCP/IP, quản lý socket, và xử lý đa luồng hiệu quả.
* **Winform:** Framework được sử dụng để xây dựng giao diện người dùng desktop.
* **ASP.NET:** Đóng vai trò backend, xử lý các chức năng quan trọng như đăng nhập, tạo tài khoản, cấp lại mật khẩu qua email, quản lý thông tin người dùng và phòng vẽ.

**ASP.NET SignalR:** Thư viện hỗ trợ giao tiếp thời gian thực trong ASP.NET, đảm bảo tính đồng bộ và cập nhật liên tục giữa client và server.

* **Python:** Một ngôn ngữ lập trình đa năng với hệ sinh thái thư viện mạnh mẽ, đặc biệt nổi bật trong các ứng dụng liên quan đến học máy và xử lý dữ liệu.
* **Flask:** Framework backend của Python, được sử dụng để xây dựng API hỗ trợ tính năng OCR (nhận dạng ký tự từ hình ảnh).
* **Uvicorn:** Đóng vai trò máy chủ ASGI để triển khai FlaskAPI, hỗ trợ hiệu quả xử lý nhiều yêu cầu đồng thời và tối ưu hiệu năng nhờ khả năng lập trình bất đồng bộ.
* **PaddleOCR:** Thư viện phân tích hình ảnh mạnh mẽ của Baidu, được tích hợp vào Flask để phục vụ tính năng chuyển đổi nét vẽ thành nội dung văn bản.
* **Skia:** Thư viện đồ họa 2D mạnh mẽ đến từ Google, cung cấp các công cụ để vẽ hình ảnh, văn bản, và xử lý đồ họa 2D với hiệu suất cao. Skia được sử dụng rộng rãi trong nhiều ứng dụng và trình duyệt hiện đại.
* **SkiaSharp:** Một wrapper .NET của Skia, là thư viện vẽ của Dreaw, đảm bảo hiệu suất trong môi trường thời gian thực.
* **SQL Server:** Hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ do Microsoft phát triển, được thiết kế để xử lý khối lượng dữ liệu lớn với hiệu suất cao và tính sẵn sàng đáng tin cậy.

# Chương 2: Cơ sở dữ liệu



*Sơ đồ quan hệ cơ sở dữ liệu*

1. **Bảng Users dùng để quản lý thông tin người dùng, bao gồm 05 thuộc tính:**

* **id**: Là khóa chính, chứa mã định danh duy nhất cho mỗi người dùng.
* **userID**: Chứa mã định danh duy nhất của người dùng (Unique Identifier).
* **name**: Chứa tên của người dùng
* **password\_hash**: Lưu trữ mật khẩu của người dùng dưới dạng mã băm (hash) để đảm bảo bảo mật.
* **email**: Chứa địa chỉ email của người dùng.

→ Bảng Users lưu thông tin cơ bản của người dùng bao gồm tài khoản, mật khẩu, và trạng thái hiện tại. Mỗi người dùng sẽ có một mã định danh duy nhất giúp dễ dàng truy vấn thông tin.

1. **Bảng Rooms dùng để lưu phòng vẽ, bao gồm 05 thuộc tính:**

* **id**: Là khóa chính, chứa mã định danh duy nhất cho mỗi phòng.
* **roomID**: Chứa mã định danh duy nhất của phòng (Unique Identifier).
* **roomName**: Chứa tên của phòng.
* **ownerID**: Là khóa ngoại, tham chiếu đến userID trong bảng Users. Thể hiện chủ sở hữu phòng.
* **data**: Chứa dữ liệu liên quan đến phòng, có thể là thông tin trạng thái hoặc dữ liệu vẽ trong phòng.

→ Bảng Rooms chứa thông tin về các phòng, bao gồm tên, chủ sở hữu, và dữ liệu phòng. Khóa ngoại ownerID đảm bảo mỗi phòng đều thuộc về một người dùng duy nhất.

**Mối quan hệ giữa các bảng:**

* Quan hệ giữa bảng Users và Rooms là 1 - n (một người dùng có thể sở hữu nhiều phòng).
* ownerID trong bảng Rooms là khóa ngoại tham chiếu đến userID trong bảng Users.

# Chương 3: Các vấn đề về đăng nhập/đăng ký

## Luồng của ứng dụng

**Luồng đăng nhập:**

**A diagram of a system

Description automatically generated**

*Sơ đồ luồng đăng nhập*

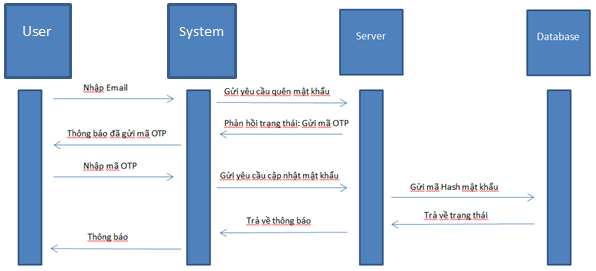
**Luồng đăng ký:**

**A diagram of a system

Description automatically generated**

*Sơ đồ luồng đăng ký*

**Luồng gửi yêu cầu quên mật khẩu:**

****

*Sơ đồ luồng "Quên mật khẩu"*

## Logic tính năng:

* **Form đăng ký:**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A white background with black text

Description automatically generated

* Kiểm tra trạng thái isSending: Ngăn việc gửi yêu cầu mới nếu yêu cầu trước chưa xử lý xong.

Lấy và kiểm tra dữ liệu:

* Kiểm tra các trường nhập liệu không rỗng.
* Kiểm tra định dạng email bằng IsValidEmail().
* Xử lý phản hồi:

200 OK:

* Trích xuất mã OTP từ phản hồi JSON.
* Mở form mới (code) và truyền OTP cùng thông tin đăng ký.
* Hiển thị thông báo thành công.

409 Conflict:

* Thông báo email đã tồn tại và yêu cầu người dùng đăng nhập.

Các lỗi khác: Thông báo lỗi server.

* **Form đăng nhập:**
* Kiểm tra trạng thái isSending: Ngăn việc gửi yêu cầu mới nếu yêu cầu trước chưa xử lý xong.
* Kiểm tra dữ liệu người dùng nhập
* Kiểm tra các trường nhập liệu không rỗng.
* Kiểm tra định dạng email bằng IsValidEmail().
* Gửi dữ liệu đăng ký đến server
* Tạo một object chứa trường Email.
* Chuyển object thành chuỗi JSON và gửi đi qua phương thức HTTP POST.
* Xử lý phản hồi từ server
* Nếu thành công:
* Hiển thị thông báo "Sign in successful".
* Trích xuất thông tin từ phản hồi JSON (name, userID).
* Gửi yêu cầu khác đến endpoint /api/room/getroomlist để lấy danh sách phòng vẽ của người dùng.
* Tạo danh sách các phòng vẽ (userRooms) và chuyển người dùng đến form world (Trang cá nhân).
* Nếu thất bại, hiện thị thông bảo lỗi.
* 401 Unauthorized: Hiển thị thông báo "Invalid Email or Password".

A screenshot of a computer screen

Description automatically generated

* **Lấy danh sách phòng vẽ**
* Endpoint: /api/room/getroomlist
* Gửi yêu cầu POST đến server với nội dung là userID (ID của người dùng).
* Server phản hồi danh sách phòng vẽ dưới dạng JSON.
* Parse JSON phản hồi để trích xuất các thông tin:
* roomName: Tên phòng.
* lastModified: Thời điểm chỉnh sửa lần cuối.
* roomID: ID của phòng.
* Tạo danh sách các phòng vẽ (userRooms) và chuyển sang form world.

A screenshot of a computer screen

Description automatically generated

* **Form quên mật khẩu:**
* Thực hiện gửi email để xác nhận quên mật khẩu.
* Xử lý phản hồi từ server.
* Hoạt động chi tiết:
* Kiểm tra trạng thái isSending
* Kiểm tra dữ liệu người dùng nhập
* Gửi yêu cầu quên mật khẩu đến server
* Xử lý phản hồi từ server
* Thành công (200 OK) => Mở form Code
* Thay đổi trạng thái sau khi hoàn tất

A screenshot of a computer program

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

* **Form Code (Nhập mã OTP): Là nơi người dùng được chuyển tới sau khi đăng ký/quên mật khẩu**
* So sánh mã người dùng nhập với mã OTP nhận được từ server.
* Nếu sai, hiển thị thông báo lỗi.
* Gửi yêu cầu hoàn tất đăng ký tới server qua API /api/finishsignup.
* Nếu thành công, mở form đăng nhập (Loginform).
* Nếu thất bại, hiển thị thông báo lỗi.
* Hiển thị thông báo xác nhận OTP thành công.
* Nếu được tạo ra để hoàn tất đăng ký: Gửi yêu cầu tới endpoint api/finishsignup, yêu cầu ghi dữ liệu người dùng vào CSDL -> Mở form đăng ký(login)
* Nếu được tạo ra để hoàn tất việc đặt mật khẩu mới: Mở form đặt lại mật khẩu (newpw).

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

* Tính năng gửi lại mã OTP:

+ API /api/resendotp được gọi để tạo OTP mới.

+ Nếu thành công:

* Nhận OTP mới từ server.
* Bắt đầu lại thời gian đếm ngược (3 phút).
* Hiển thị thông báo thành công.

+ Nếu thất bại, trả về thông báo lỗi.

A computer screen shot of a program

Description automatically generated

A computer screen with text

Description automatically generated

* Bộ đếm 3 phút, thời gian để gửi yêu cầu gửi lại OTP kế tiếp:

+ Mỗi lần sự kiện Tick của Timer được kích hoạt:

+ Kiểm tra nếu tổng thời gian đã đạt 3 phút (180 giây):

* Dừng bộ đếm (waittoResend.Stop()).
* Đặt trạng thái \_isTicking về false.

+ Nếu chưa, tăng biến timer theo khoảng thời gian của Timer.

A screen shot of a computer

Description automatically generated

* **Form cập nhật mật khẩu:**
* Kiểm tra trạng thái gửi yêu cầu
* Kiểm tra đầu vào
* Chuẩn bị dữ liệu gửi
* Gửi yêu cầu HTTP POST:
* Gửi yêu cầu đến endpoint /api/updatepw trên server.
* Nếu thành công:
* Hiển thị thông báo "Password updated successfully".
* Mở form đăng nhập (Loginform).
* Đóng form hiện tại.
* Nếu thất bại:
* Hiển thị thông báo lỗi.

**A computer screen shot of a program

Description automatically generated**

**A screenshot of a computer code

Description automatically generated**

* **Hàm kiểm tra Email:**
* [^@\s]+: Một hoặc nhiều ký tự không phải @ hoặc khoảng trắng.
* @: Ký tự bắt buộc.
* [^@\s]+: Một hoặc nhiều ký tự không phải @ hoặc khoảng trắng (phần tên miền).
* \.: Dấu chấm (phần mở rộng tên miền).
* [^@\s]+: Một hoặc nhiều ký tự không phải @ hoặc khoảng trắng (phần mở rộng tên miền).

A computer screen shot of a computer code

Description automatically generated

# Chương 4: Trang cá nhân

# **Tính năng tạo phòng**

* + 1. Các tính năng:
* Cho phép người dùng nhập tên phòng vào một hộp văn bản (Textbox).
* Tự động kiểm tra và thông báo nếu tên phòng không hợp lệ (quá dài hoặc để trống).
* Tạo một mã phòng ngẫu nhiên (4 chữ số) đảm bảo tính duy nhất thông qua xác thực với một server từ xa.
* Đảm bảo thông tin được lưu lại (name, ID) và gửi thông báo đến server.
  + 1. Các thành phần chính:

A computer code on a black background

Description automatically generated

* name: Tên phòng nhập bởi người dùng, lưu vào sau khi kiểm tra hợp lệ.
* ID: Mã phòng hợp lệ sau khi server xác thực.
* Cấu hình API: serverAdd chứa địa chỉ server để gửi yêu cầu POST và xác thực mã phòng.
  + 1. Kiểm tra dữ liệu nhập:
* Nếu trống: Hiển thị thông báo yêu cầu nhập tên phòng và ẩn nút "Tạo phòng".
* Nếu vượt quá 20 ký tự: Hiển thị lỗi màu đỏ
* Hợp lệ: Hiển thị nút "Tạo phòng" (button1), ẩn thông báo lỗi.
  + 1. Sinh mã phòng ngẫu nhiên và xác thực:
* Sinh ngẫu nhiên mã phòng (4 chữ số).
* Xác thực mã với server:
* Mã đã tồn tại: Thử lại với mã khác.
* Mã không tồn tại: Chấp nhận mã và kết thúc vòng lặp.
  + 1. Tạo phòng:
* Tạo mã phòng: Gọi hàm GenerateRoom (bất đồng bộ) để lấy mã phòng duy nhất.
* Lưu tên phòng: Lấy từ hộp văn bản richTextBox1 và lưu lại.
* Đóng Form: Sau khi hoàn tất.

## Tính năng tham gia vào phòng

* + 1. Các tính năng:
* Kiểm tra trạng thái của việc tham gia phòng.
* Hiển thị form nhập mã phòng (enterCode).
* Xử lý mã phòng hợp lệ: kiểm tra nếu phòng đã thuộc danh sách phòng.
* Thực hiện kết nối với máy chủ và hiển thị giao diện vẽ cho phòng.
  + 1. Quy trình thực hiện:
* Kiểm tra trạng thái is\_join\_a\_room
* Nhập mã phòng qua enterCode
* Xử lý mã phòng hợp lệ:
* Trường hợp 1: Mã phòng đã có trong danh sách phòng
* Sử dụng danh sách userRooms để kiểm tra mã phòng có tồn tại không
* Nếu có, thực hiện lấy bitmap hiện tại của phòng thông qua GetBitmap.
* Kết nối với server bằng hàm ConnectServer.
* Hiển thị màn hình vẽ (DoAnPaint.Form1) với bitmap nếu dữ liệu không rỗng.
* Trường hợp 2: Mã phòng không có trong danh sách phòng
* Gửi mã phòng tới API /api/room/getname để lấy tên phòng.
* Nếu API trả về mã trạng thái thành công (IsSuccessStatusCode), tên phòng được lấy từ nội dung phản hồi.
* Lấy bitmap và hiển thị màn hình vẽ như ở trường hợp 1.
* Xử lý mã phòng không hợp lệ:
* Hiển thị thông báo lỗi, đặt lại con trỏ chuột và trạng thái tham gia.

## Tham gia phòng từ danh sách phòng vẽ:

* + 1. Các tính năng:
* Hiển thị danh sách các phòng vẽ hiện có mà người dùng sở hữu hoặc đã tham gia.
* Cho phép người dùng chọn phòng vẽ từ danh sách để kết nối và tiếp tục sử dụng giao diện vẽ.
* Xử lý trạng thái tham gia phòng để ngăn chặn việc gửi yêu cầu trùng lặp.
  + 1. Quy trình thực hiện:
* Hiển thị danh sách phòng:
  + Sử dụng danh sách userRooms chứa các thông tin về phòng (ID, tên phòng, và các thông tin liên quan).
  + Hiển thị từng phòng dưới dạng các control trong danh sách userRoomList.
* Xử lý sự kiện chọn phòng:
  + Khi người dùng chọn một phòng, tất cả các phòng khác sẽ được đặt lại màu nền mặc định.
  + Phòng được chọn sẽ được tô sáng (Color.LightYellow) và cập nhật thông tin selectedRoom (ID phòng) và roomname (tên phòng).
* Kết nối tới phòng:
  + Khi người dùng nhấn nút xác nhận tham gia phòng (sự kiện pictureBox4\_Click):
    - Kiểm tra trạng thái is\_join\_a\_room:
      * Nếu người dùng đang gửi một yêu cầu tham gia khác, hiển thị thông báo và ngăn thực hiện thêm yêu cầu.
    - Thực hiện kết nối:
      * Lấy dữ liệu bitmap hiện tại của phòng bằng phương thức bất đồng bộ GetBitmap.
      * Kết nối tới server thông qua ConnectServer với thông tin người dùng và phòng.
      * Hiển thị giao diện vẽ (Form DoAnPaint.Form1) với bitmap hiện tại nếu có, hoặc tạo một canvas trống nếu không có dữ liệu.
    1. Các thành phần chính:
* Danh sách phòng (userRooms):
  + Chứa thông tin về các phòng người dùng sở hữu hoặc tham gia.
* Phương thức chính:
  + pictureBox4\_Click: Xử lý sự kiện tham gia phòng.
  + GetBitmap: Lấy dữ liệu bitmap của phòng từ server.
  + ConnectServer: Kết nối tới server với các thông tin cần thiết.
    1. Quy trình chi tiết:
* Hiển thị danh sách phòng trong userRoomList.
* Xử lý sự kiện chọn phòng, cập nhật thông tin phòng được chọn.
* Khi nhấn nút tham gia:
  + Kiểm tra trạng thái is\_join\_a\_room.
  + Kết nối với server và lấy dữ liệu bitmap.
  + Hiển thị giao diện vẽ với dữ liệu phù hợp.

## Bộ điều khiển API (RoomController)

* + 1. Chức năng chính và mục tiêu:
* Chức năng chính và mục tiêu.
* Tìm tên phòng bằng mã phòng.
* Lấy dữ liệu hiện tại của phòng (canvas data).
* Truy xuất danh sách phòng thuộc sở hữu của người dùng.
* Tích hợp **SignalR** để gửi thông báo và truyền dữ liệu tới các nhóm kết nối trong thời gian thực.
  + 1. Các thành phần chính:
* Thuộc tính và Constructor:
* IHubContext: Đối tượng SignalR cho phép giao tiếp với các nhóm kết nối hoặc các client cụ thể.
* Constructor nhận IHubContext<DreawHub> qua Dependency Injection, cho phép sử dụng SignalR trong các hành động của controller.
* Endpoint kiểm tra mã phòng (RoomExists):
* API POST /api/room/exists: Kiểm tra sự tồn tại của mã phòng.
* Đọc mã phòng từ nội dung HTTP request (Request.Body).
* Gọi phương thức CheckRoomExists để kiểm tra trong cơ sở dữ liệu:
* **Mã phòng tồn tại:** Trả về mã HTTP **200 OK**.
* **Mã phòng không tồn tại:** Trả về mã **409 Conflict**.
  + 1. Endpoint tìm tên phòng (IsWorking):
* API POST /api/room/getname: Lấy tên phòng từ mã phòng.
* Đọc mã phòng từ nội dung HTTP request.
* Gọi phương thức FindnameByID để tìm tên phòng trong cơ sở dữ liệu:
* **Tìm thấy tên phòng**: Trả về tên phòng với mã **HTTP 200 OK.**
* **Không tìm thấy: Trả về mã HTTP 409 Conflict.**
  + 1. Endpoint lấy dữ liệu hiện tại của phòng (GetBitmap):
* API POST /api/room/getcurrentbmp:
* Kiểm tra xem phòng có đang tồn tại trong RoomUserMapping (danh sách kết nối SignalR).
* Nếu có:
* Gửi lệnh **StopConsumer** đến tất cả các kết nối trong phòng.
* Gọi **GetUserBitmap** để lấy dữ liệu bitmap của phòng từ kết nối hiện tại.
* Gửi lệnh **StartConsumer** để khởi động lại quá trình tiêu dùng dữ liệu.
* Nếu không:
* Gọi phương thức GetRoomData để lấy dữ liệu canvas từ cơ sở dữ liệu:
* **Dữ liệu tồn tại:** Trả về dữ liệu với mã **200 OK**.
* **Dữ liệu không tồn tại:** Trả về mã **404 Not Found**.
  + 1. Các phương thức hỗ trợ:
* Kiểm tra mã phòng (CheckRoomExists)
* Gửi truy vấn SQL để kiểm tra mã phòng trong bảng Rooms
* Kết quả: Nếu tồn tại, trả về true; nếu không, trả về false.
* Tìm tên phòng (FindnameByID)
* Gửi truy vấn SQL để lấy tên phòng từ mã phòng:
* Kết quả: Trả về tên phòng (chuỗi) nếu tìm thấy, null nếu không.
* Lấy dữ liệu canvas của phòng (GetRoomData):
* Gửi truy vấn SQL để lấy canvasData (dữ liệu canvas) của phòng từ mã phòng.
* Kết quả: Trả về dữ liệu canvas nếu tìm thấy, null nếu không.
* Lấy danh sách phòng của người dùng (GetDataList)
* Gửi truy vấn SQL để lấy danh sách phòng thuộc sở hữu của một người dùng.
* Trả về danh sách dưới dạng JSON, với các trường:
* roomName - Tên phòng.
* lastModified - Thời gian chỉnh sửa lần cuối.
* roomID - Mã phòng.
  + 1. Quy trình hoạt động:
* Người dùng gửi yêu cầu đến API với hành động cụ thể (kiểm tra phòng, lấy tên phòng...).
* Controller xử lý yêu cầu:
* Đọc dữ liệu từ HTTP request body.
* Gửi truy vấn SQL để lấy dữ liệu cần thiết hoặc xác nhận với SignalR.
* Trả về dữ liệu hoặc trạng thái HTTP tương ứng.

# Chương 5: Tính năng vẽ chung thời gian thực

Vẽ trên SKBitmap, render bằng SKCanvas và hiện thị bằng SKControl

## Logic vẽ

-Để giải quyết vấn đề tranh chấp tài nguyên giữa luồng UI và luồng dữ liệu từ Server, nhóm em sử dụng mô hình Producer-Consumer để quản lý dữ liệu:

* Luồng Producer (Nhà sản xuất):

+Khi người dùng thực hiện các thao tác vẽ hoặc chỉnh sửa trên canvas, dữ liệu tương ứng (toạ độ, màu sắc, kiểu hình vẽ) được đưa vào hàng đợi (**BOTQueue**).

+Nếu có tín hiệu từ server, dữ liệu này cũng được đưa vào hàng đợi để xử lý.

* Luồng Consumer (Người tiêu thụ):

+Luồng này liên tục theo dõi hàng đợi **BOTQueue** để nhận dữ liệu mới.

+Dữ liệu vẽ được áp dụng lên canvas, bao gồm vẽ hình, tô màu, hoặc cập nhật preview theo thời gian thực.

* Tóm lại: Chỉ luồng Consumer được sử dụng Bitmap. Giống như việc thay vì có hai đứa trẻ tranh nhau tấm bảng vẽ thì giờ Producer-Consumer sẽ đóng vai trò như bảo mẫu, nghe xem hai đứa trẻ muốn vẽ gì và vẽ lên bảng vẽ => Hiệu suất được cải thiện rất đáng kể.

-Sơ đồ hoạt động:

* Luồng Input-Output: Với Input là dữ liệu từ việc người dùng tương tác với giao diện, và Output là dữ liệu hiện thị lên giao diện:

A diagram of a server

Description automatically generated

* Khối xử lý dữ liệu:  
  A diagram of a process

  Description automatically generated

*\*Queue sử dụng BlockingCollection()  
\*Nét Preview không được gửi lên Server*

-Cách xử lý dữ liệu:

Luồng Consumer hoạt động bằng cách:

* Dùng foreach để duyệt từng phần dữ liệu trong hàng đợi.
* Tách dữ liệu thành 3 phần: command (enum), data (DrawingData) và preview (bool), truyền vào 3 biến toàn cục (Do sự kiện PaintSurface không hỗ trợ truyền tham số, nên em phải làm vậy)
* Gọi sự kiện PaintSurface():
* Tạo ra render\_canvas, là SKCanvas tạo ra từ bề mặt của SKControl, nơi hiện thị. Xóa trắng render\_canvas.
* Preview là false hay true?
* Nếu false, truyền SKCanvas graphic (SKCanvas tạo ra từ SKBitmap chính) cùng dữ liệu (được gắn vào biến toàn cục ở trên) vào HandleDrawData() để hàm này vẽ lên SKBitmap chính. Sau đó dùng render\_canvas render SKBitmap chính lên giao diện.
* Nếu true, render SKBitmap chính lên giao diện trước. Sau đó truyền render\_canvas cùng dữ liệu vào HandleDrawData() để render thẳng lên giao diện.

## Tính năng vẽ:

* + 1. Các tính năng:
* Ứng dụng hỗ trợ nhiều tính năng:
  + Pencil: Vẽ các đường thẳng liền nét.
  + Crayon: Vẽ các đường nét với texture sáp màu.
  + Eraser: Xóa các nét vẽ bằng cách vẽ màu trắng.
  + Line, Rectangle, Ellipse: Vẽ các hình cơ bản.
  + Polygon: Vẽ đa giác bằng cách xác định nhiều điểm.
  + Bezier: Vẽ đường cong mượt mà.
  + Fill: Tô màu vào vùng được chọn.
  + OCR: Nhận diện và hiển thị văn bản từ vùng chọn.
  + Cursor: Con trỏ gốc để chọn vùng.
* Mỗi tính năng có một command riêng, được quản lý thông qua thuộc tính **Cmd**. Khi tính năng thay đổi, giao diện UI sẽ cập nhật màu sắc và kiểu con trỏ chuột tương ứng (event-driven).
  + 1. Dữ liệu vẽ
* Dữ liệu được biểu diễn qua đối tượng DrawingData, bao gồm:
* Màu sắc (color), độ dày (width)
* Tọa độ (PointX, PointY, startX, startY, endX, endY)
* List các điểm (hỗ trợ Polygon và Curve)
* Bitmap (hỗ trợ Fill và tính năng đồng bộ hóa)
* Dữ liệu này được tạo ra từ Input, và chuyển lên server dưới dạng JSON.
  + 1. Nhóm Pen(Pencil, Crayon, Eraser):
* Logic: Vẽ các đường liền nét
* Khi người dùng ghì chuột, ứng dụng ghi nhận điểm bắt đầu.
* Khi di chuyển chuột:

-Tính toán tọa độ mới (pointY).

-Tạo dữ liệu vẽ DrawingData chứa màu, độ dày, và tọa độ hai đầu đoạn thẳng.

-Thêm dữ liệu vào hàng đợi vẽ.

-Gọi hàm SendData(bất đồng bộ) và cập nhật điểm bắt đầu mới (pointX = pointY).

* + 1. Nhóm Shape(Rectangle, Ellipse, Polygon, Curved):
* Logic: Vẽ hình dạng theo một tập hợp các điểm:

-Rectangle: Điểm bắt đầu, chiều rộng, chiều cao

-Ellipse: Điểm bắt đầu, kích thước 2 trục elip

-Polygon: Vẽ đa giác là một đường kín nối qua từng điểm theo thứ tự

-Curved: Vẽ một đường cong mượt mà nối qua từng điểm theo thứ tự

* Shape hỗ trợ tính năng Preview: Khi di chuyển chuột, sẽ vẽ hình tương ứng với chế độ đang chọn dựa vào vị trí chuột người dùng lên thẳng UI(không vẽ vào Bitmap).
* Bắt đầu vẽ: Nhấn chuột xuống, ghi nhận điểm bắt đầu.
* Điều chỉnh nét vẽ:  
  -Nếu là Rectangle/Ellipse: Yêu cầu ghì chuột. Kích thước được tính toán liên tục theo tọa độ chuột.  
  -Nếu là Polygon/Curved: Chỉ cần di chuyển chuột. Ấn chuột trái để thêm điểm vào Points(List lưu trữ để vẽ điểm). Di chuyển chuột sẽ cập nhật vị trí chuột vào TempPoints(List lưu điểm tạm, TempPoints bằng Points + tọa độ chuột hiện tại).

-Tạo dữ liệu vẽ DrawingData chứa màu, độ dày nét bút, tọa độ bắt đầu và kích thước hiện tại/TempPoints, tùy chế độ.

-Thêm dữ liệu vào hàng đợi vẽ, cờ Preview = true.

* Kết thúc:  
   -Nếu là Rectangle/Ellipse: Thả chuột. Tính toán kích thước cuối cùng.

-Nếu là Polygon/Curved: Ấn chuột phải sẽ thêm điểm vừa ấn vào Points.

-Tạo dữ liệu vẽ DrawingData chứa màu, độ dày nét bút, tọa độ bắt đầu và kích thước cuối cùng/Points, tùy chế độ.

-Thêm dữ liệu vào hàng đợi vẽ, cờ Preview = false.

-Gọi hàm SendData(bất đồng bộ).

* + 1. Fill:
* Tô toàn bộ một vùng kín được chọn.
* Logic: Để tránh block luồng chính, em sử dụng logic bất đồng bộ:

-Khi người dùng nhấn chuột vào canvas trong chế độ Fill, chương trình ghi nhận toạ độ pixel tại vị trí chuột; màu sắc của pixel tại điểm nhấn chuột được gọi là màu gốc.

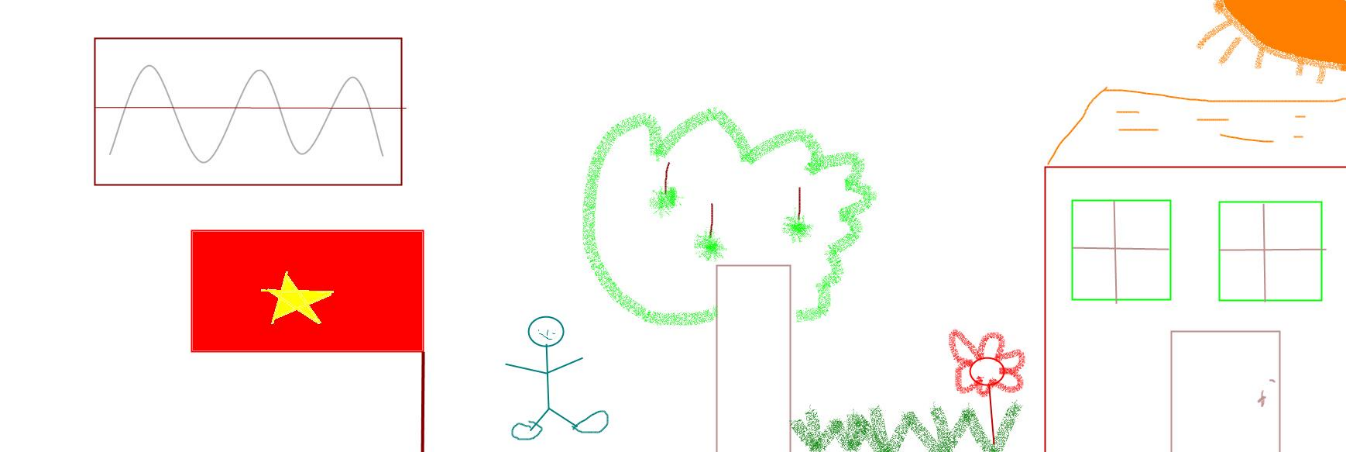
-Thuật toán tô màu: Sử dụng thuật toán DFS (Depth-First Search) để duyệt qua tất cả các pixel liền kề có màu trùng với màu gốc. Tại mỗi pixel được duyệt, màu của pixel đó sẽ được thay thế bằng màu mới do người dùng chọn.

-Tuy nhiên, thuật toán không tô thẳng lên Bitmap chính. Khi người dùng click chuột, chương trình sao chép một Bitmap phụ, chạy thuật toán tô màu lên đó rồi trả lại một Bitmap đã được tô màu. Bitmap này sau đó sẽ được đóng gói vào DrawingData, đẩy vào hàng đợi vẽ và gửi lên Server. Điều này cho phép Fill chạy bất đồng bộ.

* + 1. OCR:
* Đọc chữ từ vùng được chọn.
* Logic:
* Phía Client:  
  -Đầu tiên người dùng phải chọn một vùng trên canvas ở chế độ Cursor. Sau đó, khi người dùng ấn nút “Read text”, chương trình cắt ra Bitmap tương đương với hình chữ nhật được chọn bởi người dùng, đóng gói vào một MultipartFormDataContent và gửi tới Server bằng HttpPost.
* Phía Server:  
  -Server ASP.NET Core proxy sang cho server Flask (chạy trên Uvicorn để xử lý được nhiều yêu cầu đồng thời).

-Flask kiểm tra, đảm bảo file ảnh không bị lỗi hoặc thiếu thông tin. Sau đó, đọc file ảnh dùng OpenCV, dùng PaddleOCR để xử lý ảnh, và trả về một string là một ma trận tái hiện vị trí của chữ trên ảnh.  
-ASP.NET Core proxy ngược về Client.

* Phía Client:  
  -Nếu nhận được phản hồi 200 OK, tạo ra DrawingData và đẩy vào hàng đợi vẽ + gửi lên Server. Khi vẽ, đầu tiên tô trắng toàn bộ vùng chọn của người dùng, sau đó dùng DrawText của SkiaSharp vẽ đè lên.  
  -Nếu không, xử lý tùy lỗi và return.
  + 1. Tính năng khác:
* Color Picker: Nhấn 2 lần để chọn màu là màu tại vị trí chuột.
* Clear: Khi chọn vùng trước, ấn nút Clear để xóa vùng đó. Nếu không chọn, Clear sẽ xóa trắng cả Canvas.
* Chat: Tin nhắn từ Chat cũng được xử lý theo Producer-Consumer, nhưng Chat có BlockingCollection riêng và Consumer riêng.
* Lưu ảnh về máy.



*Một bức vẽ sử dụng ứng dụng*

# Chương 6: Giao tiếp và đồng bộ hóa dữ liệu thời gian thực sử dụng SignalR

**I.** Công nghệ và các tính năng:

- DreawData.cs là tệp nhằm hỗ trợ giao tiếp thời gian thực, được viết dựa trên thư viện SignalR, cung cấp cơ chế quản lý và đồng bộ dữ liệu giữa các người dùng.

- SignalR, một thư viện của ASP.NET dùng để xây dựng các ứng dụng giao tiếp thời gian thực, cho phép server và client trao đổi dữ liệu ngay lập tức. SignalR tự chọn công nghệ giao tiếp (WebSockets, Server-Sent Events, Long Polling).

- Các tính năng chính:

* **Giao tiếp thời gian thực**: Cho phép gửi, nhận thông điệp hoặc dữ liệu vẽ giữa các thành viên trong cùng một phòng.
* **Đồng bộ hóa dữ liệu**: Đảm bảo mọi người dùng trong phòng đều có cùng một trạng thái canvas.
* **Lưu trữ dữ liệu**: Hỗ trợ lưu trữ dữ liệu vẽ vào cơ sở dữ liệu (SQL Server) tại máy host.
* **Và các logic liên quan đến vào/ra phòng khác…**

**II.** Logic tính năng:

* **Giao tiếp thời gian thực**:
* Phương thức BroadcastDraw chuẩn bị 1 tuple dữ liệu của client dựa trên ConnectionID, chọn trong số ấy mã phòng. Sau đó thực hiện đẩy dữ liệu client đã vẽ mới đến các client khác hiện có mặt chung phòng.

=> Thông báo client hiện tại thực hiện mới thao tác vẽ

* Phương thức BroadcastMessage hoạt động tương tự với dữ liệu là thông điệp.

=> Thông báo client hiện tại có thông điệp cần lan tỏa.

A computer screen with many colorful text

Description automatically generated

*Phương thức gửi, nhận thông điệp hoặc dữ liệu vẽ*

* **Đồng bộ hóa dữ liệu**:

+ Khi client được xác thực vào phòng, server sẽ gửi lại bitmap mới nhất đã vẽ từ các client khác đến client vừa vào.

A screen shot of a computer program

Description automatically generated

*Phương thức đồng bộ dữ liệu lần đầu*

* Tạo 1 TaskCompletionSource để đợi kết quả bitmap từ client (bất đồng bộ), sau khi nhận thì gửi ngay yêu cầu đồng bộ tới các client khác thông qua signalR hub, chờ kết quả.
* Phương thức được thực hiện liên tục đảm bảo tính thời gian thực, đồng bộ dữ liệu tức thời.

A computer screen shot of a program

Description automatically generated

*Phương thức đồng bộ dữ liệu*

* **Lưu trữ dữ liệu**:

+ Phương thức hỗ trợ cho chức năng lưu dữ liệu vào database (Chương 4). thực hiện các kiểm tra và các câu lệnh SQL tương tác với database để thêm/sửa.

A computer screen shot of a program

Description automatically generated

*Phương thức lưu trữ dữ liệu vào database của host*

+ Phương thức lưu trữ bitmap dựa trên phương thức SavetoDB, thực hiện tìm thông tin client dựa trên ConnectionId, dừng mọi hoạt động của các client khác, sau đó lưu trữ bitmap hiện tại vào database, hoàn tất thì cho phép các client hoạt động tiếp.

A computer screen shot of text

Description automatically generated

*Phương thức lưu trữ bản vẽ vào database*

* **Các logic vào/ra phòng:**

+ Các phương thức phụ hỗ trợ việc thêm/xóa client khỏi room (ngoài ra còn thông báo vào/ra khỏi phòng).

A computer code on a black background

Description automatically generated

*Các phương thức hỗ trợ việc thêm/xóa client khỏi room*

+ Khi client kết nối đến server, hub SignalR sẽ lấy các thông tin cần thiết, thêm thông tin client vào room, cập nhật roomId của client và đẩy client vào room đích.

A computer screen shot of a program

Description automatically generated

*Phương thức kết nối client đến server*

+ Khi client yêu cầu ngắt kết nối, Hub SignalR sẽ tìm client đó dựa trên ConnectionId, loại client khỏi room, xóa hoàn toàn dữ liệu client.

A screen shot of a computer program

Description automatically generated

*Phương thức hủy kết nối*

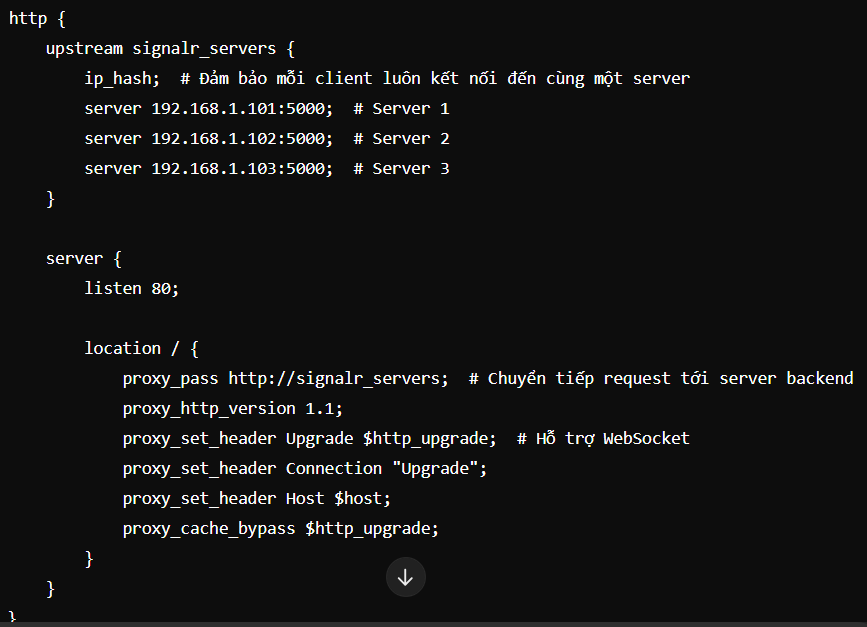
# Chương 7: Phân công công việc

# Bảng phân công công việc chi tiết

|  |  |  |  |
| --- | --- | --- | --- |
| STT | MSSV | Tên | Nội dung công việc phụ trách |
| 1 | 23521746 | Nguyễn Viết Tùng | - Thiết kế logic ứng dụng - Phân công công việc, giám sát tiến độ - Phát triển tính năng vẽ, backend Flask  - Phát triển logic đóng gói và xử lý dữ liệu vẽ  - Hỗ trợ phát triển backend ASP.NET Core - Soạn báo cáo cuối kỳ |
| 2 | 23521458 | Ông Vũ Minh Thành | - Thiết kế logic xác thực người dùng - Phát triển tính năng đăng nhập, đăng ký, quên mật khẩu  - Triển khai endpoint API cho tính năng đăng nhập, đăng ký, quên mật khẩu trên backend  - Soạn báo cáo cuối kỳ |
| 3 | 23521479 | Lý Trung Thiện | - Thiết kế cơ sở dữ liệu - Thiết kế giao diện người dùng - Phát triển form trang cá nhân  - Triển khai các endpoint hỗ trợ form trang cá nhân trên backend  - Soạn báo cáo cuối kỳ |
| 4 | 23521525 | Nguyễn Minh Thông | - Làm slide báo cáo - Triển khai phần giao tiếp và đồng bộ hóa thời gian thực trên backend  - Triển khai việc gửi dữ liệu dưới frontend  - Hỗ trợ triển khai tính năng OCR  - Triển khai endpoint đồng bộ hóa trên backend - Soạn báo cáo cuối kỳ |

Chương 8: Ý tưởng về Multi-serverĐây là phần nhóm em chưa kịp triển khai, nhưng chúng em có ý tưởng về cách triển khai Multi-server và Load Balancing:

* Sử dụng Nginx làm Load Balancer.
* Vì SignalR yêu cầu Sticky Session (mọi kết nối từ một địa chỉ IP phải tới một server), em sẽ kích hoạt tùy chọn ip\_hash trong Nginx để đảm bảo điều đó.
* Để đảm bảo việc đồng bộ hóa trạng thái giữa các server, em sẽ sử dụng Redis Backplane để giúp các server có thể liên kết và chia sẻ trạng thái với nhau, đảm bảo khi một node hư hỏng thì server vẫn sẽ có thể hoạt động.
* Ví dụ triển khai Nginx:



* Ví dụ triển khai Redis Backplane:



# Tài liệu tham khảo

[1] ASP.NET Core: [Get started with ASP.NET Core | Microsoft Learn](https://learn.microsoft.com/vi-vn/aspnet/core/getting-started/?view=aspnetcore-9.0)

[2] SignalR: [Overview of ASP.NET Core SignalR | Microsoft Learn](https://learn.microsoft.com/vi-vn/aspnet/core/signalr/introduction?view=aspnetcore-9.0)

[3] SkiaSharp: [SkiaSharp Namespace | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/api/skiasharp?view=skiasharp-2.88)

[4] PaddleOCR: [Home - PaddleOCR Documentation](https://paddlepaddle.github.io/PaddleOCR/main/en/index.html)

[5] Stack Overflow: [Stack Overflow - Where Developers Learn, Share, & Build Careers](https://stackoverflow.com/)