TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO
ĐỒ ÁN SOCKET

MÔN: MẠNG MÁY TÍNH

Tp. Hồ Chí Minh, tháng 12/2024

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO
ĐỒ ÁN SOCKET

MÔN: MẠNG MÁY TÍNH

LỚP: 23TNT1

GIÁO VIÊN HƯỚNG DẪN

Đỗ Hoàng Cường

Thành viên Nhóm 1

1. **Nguyễn Lâm Phú Quý - 23122048**
2. **Đào Sỹ Duy Minh - 23122041**
3. **Huỳnh Trung Kiệt - 23122039**

Tp. Hồ Chí Minh, tháng 12/2024

Lời cảm ơn

Lời đầu tiên, xin trân trọng cảm ơn thầy Đỗ Hoàng Cường đã tận tình hướng dẫn chúng em trong quá trình nghiên cứu cũng như hoàn thành báo cáo này để hỗ trợ cho đồ án.

Xin chân thành cảm ơn các Thầy, Cô thuộc khoa Công nghệ Thông tin nói riêng và toàn bộ Thầy, Cô ở trường Đại học Khoa học Tự Nhiên - ĐHQG TP.HCM nói chung đã tận tình giảng dạy cho chúng em trong suốt thời gian học tập.

Do giới hạn kiến thức và kĩ năng nên chúng em không tránh khỏi có những thiếu sót. Kính mong sự chỉ dẫn và đóng góp của các Thầy, Cô để bản báo cáo và sản phẩm đồ án của chúng em được hoàn thiện hơn. Chúng em xin chân thành cảm ơn!

Thành phố Hồ Chí Minh, ngày 6 tháng 12 năm 2024

Nhóm 1

Mục lục

Giới thiệu

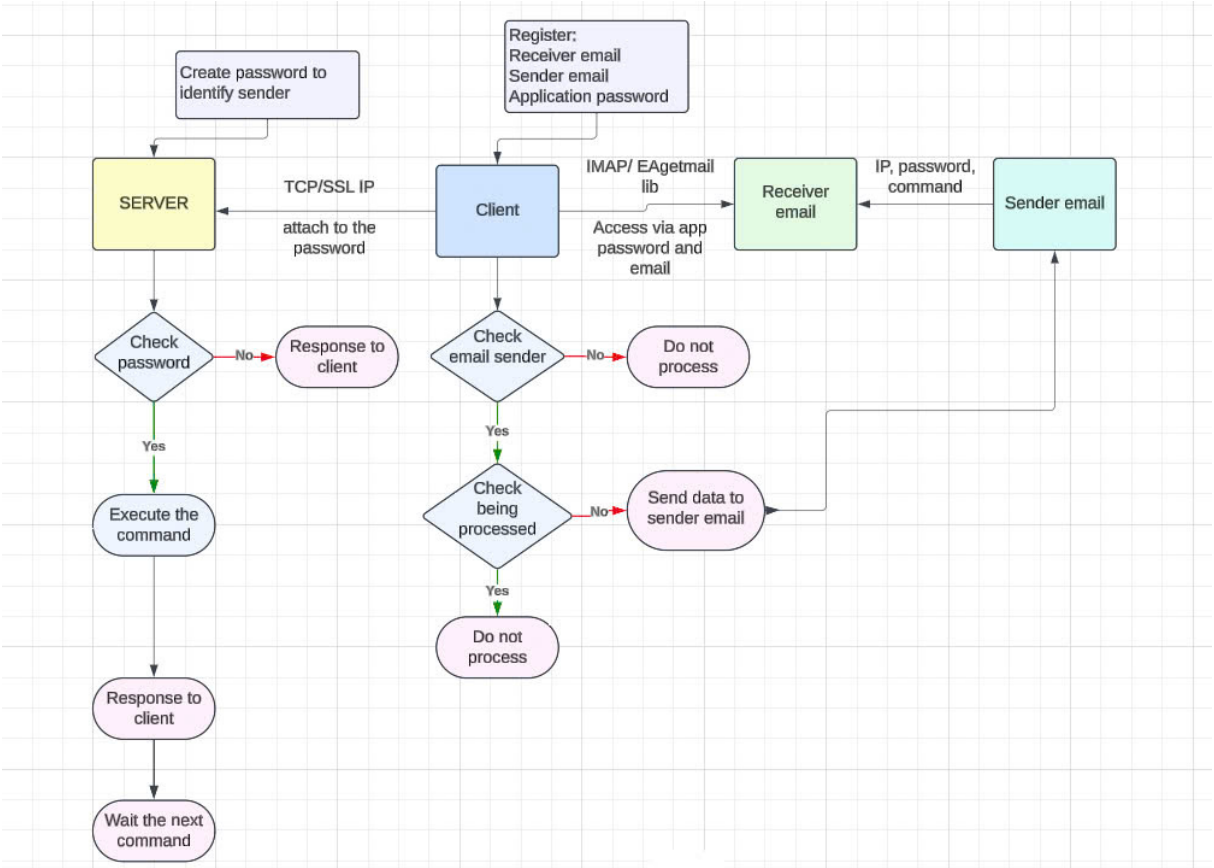
2.1 Thông tin thành viên nhóm

MSSV	Họ và Tên	Email
23122048	Nguyễn Lâm Phú Quý	23122048@student.hcmus.edu.vn
23122041	Đào Sỹ Duy Minh	23122041@student.hcmus.edu.vn
23122039	Huỳnh Trung Kiệt	23122039@student.hcmus.edu.vn

2.2 Phân công công việc

Tổng quan về đồ án socket

Đồ án socket yêu cầu viết một chương trình điều khiển máy tính bằng email (gmail). Cụ thể, đồ án có 3 thành phần chính là máy server, máy client và một email để trích xuất hòm thư.
Máy tính client sẽ trích xuất các mails từ hòm thư. Máy server sẽ nhận lệnh từ client, sau đó thực thi các lệnh gửi về cho máy client, cuối cùng, máy client sẽ gửi cho email đã ra lệnh.
Tổng quan về đồ án được miêu tả bởi sơ đồ sau:

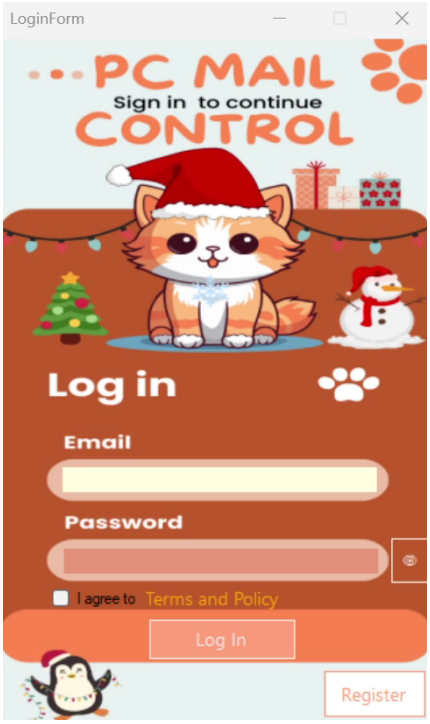


Hình 3.1: Flow Chart .

Các giai đoạn tiến hành đồ án

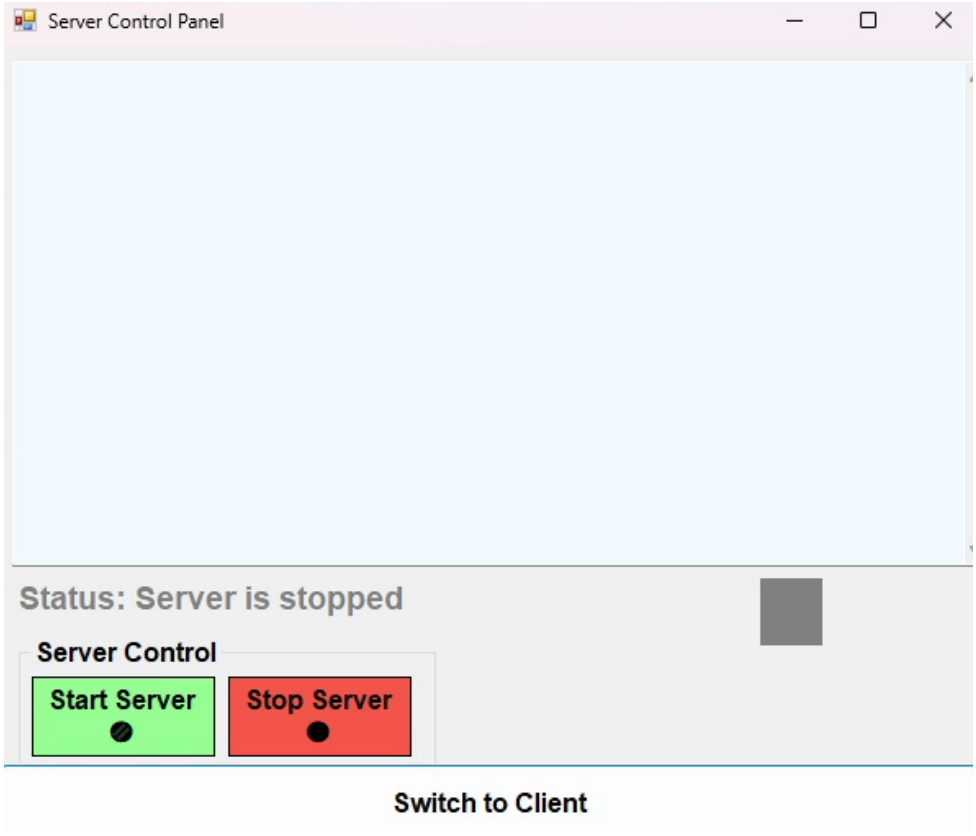
Giai đoạn	Nội dung công việc
Tuần 1, 2,	-Tìm hiểu giải pháp điều khiển máy tính qua Gmail -Nghiên cứu phương pháp giao tiếp qua Gmail -Xác định các tính năng cơ bản của ứng dụng
Tuần 3, 4 5, 6	-Xây dựng tích hợp chức năng điều khiển máy tính từ xa thông qua email. -Thử nghiệm các lệnh điều khiển cơ bản -Kiểm thử tính năng gửi lệnh và xử lý phản hồi. -Phát hiện và sửa lỗi trong quá trình kiểm thử
Tuần 5, 6	-Lên ý tưởng các chức năng mới -Thiết kế giao diện -Thiết kế database và sử dụng Microsoft Azure
Tuần 7, 8	-Tiến hành tinh chỉnh và tối ưu hóa code -Thực hiện các bước kiểm thử kỹ lưỡng trên môi trường máy ảo
Tuần 9, 10	Viết báo cáo và quay video demo ứng dụng

Giao diện ứng dụng

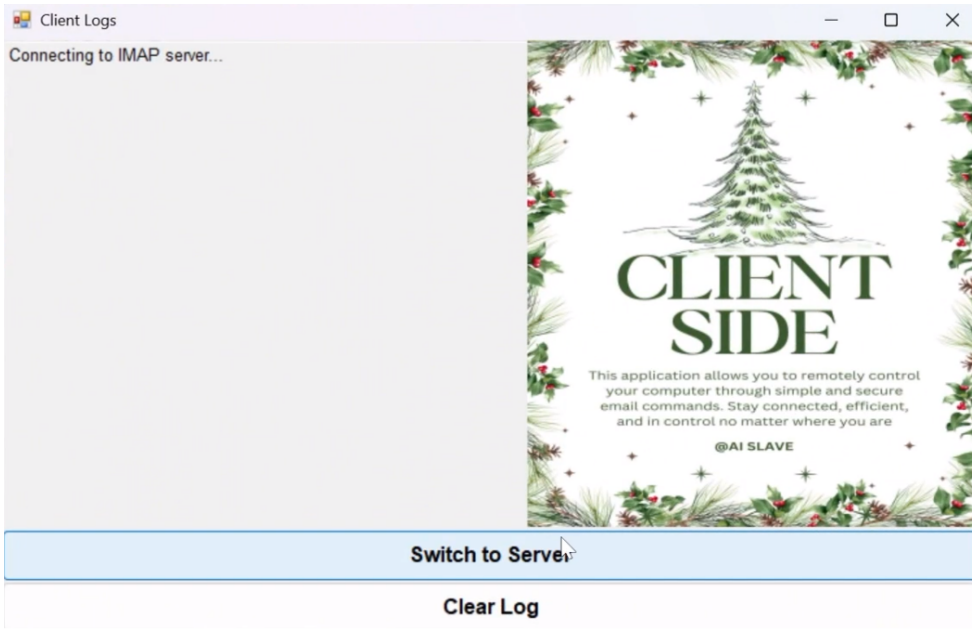


Hình 5.1: Giao diện Login .

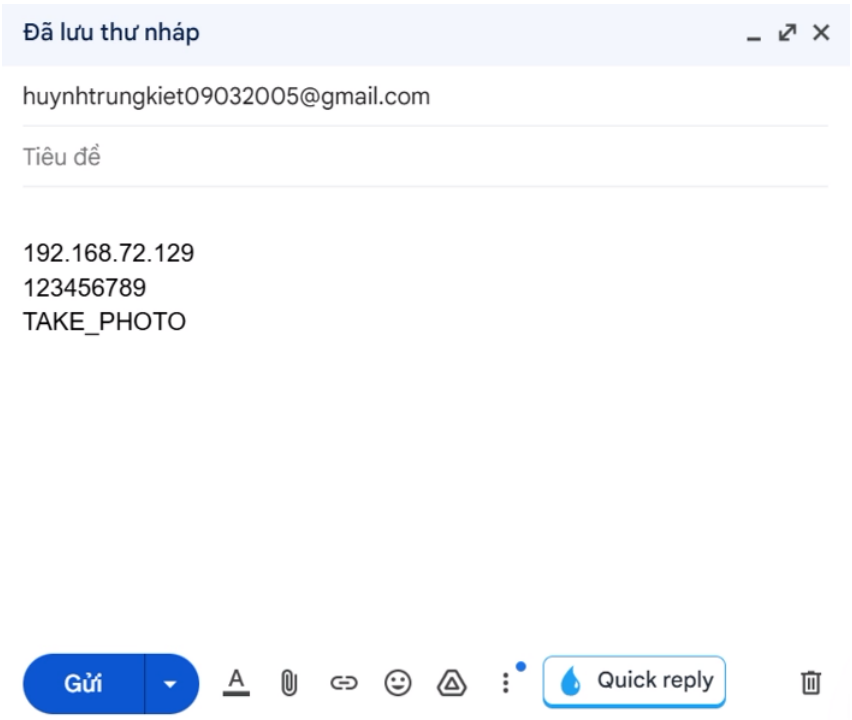
Hình 5.2: Giao diện Sign Up .



Hình 5.3: Giao diện Server.



Hình 5.4: Giao diện Client.



Hình 5.5: Gửi thư Gmail.

- Nội dung của thư gửi:
  - Dòng thứ nhất: IP của máy server
  - Dòng thứ hai: Mật khẩu ứng dụng
  - Dòng thứ ba: câu lệnh
- Các câu lệnh có thể sử dụng:
  - LIST_APPS
  - LIST_SERVICES
  - START_APP đường dẫn tuyệt đối
  - STOP_APP đường dẫn tuyệt đối
  - GET_FILE đường dẫn tuyệt đối
  - DELETE_FILE đường dẫn tuyệt đối
  - TAKE_SCREENSHOT
  - TAKE_PHOTO
  - SEND_VIDEO số giây
  - START_SERVICE tên service
  - STOP_SERVICE tên service
  - SHUTDOWN

Chi tiết về mã nguồn trong đồ án

BackEnd

6.2 Server-side

Ở phần này, hàm ExecuteCommand(Stringˆ command, ServerFormˆ form, NetworkStreamˆ stream) có chức năng đọc lệnh trong tham số command, rồi lần lượt thực thi các lệnh tương ứng

6.2.1 Chức năng list các apps đang chạy trên máy server

- Server sẽ thực hiện việc lấy các tiến trình đang chạy bằng cách khởi tạo biến processList có kiểu Process: Sau đó, bằng cách thực hiện lệnh "tasklist" (một lệnh nội bộ của windows), ta có thể liệt kê các tiến trình đang chạy trên hệ thống.
- Sau đó, ta chuyển dữ liệu lấy được thành một mảng byte, rồi ghi dữ liệu trong mảng byte này vào luồng mạng.

6.2.2 Chức năng list các services đang chạy trên máy server

- Tương tự như chức năng listing apps, một biến serviceList cũng được khởi tạo như một đối tượng process để lưu giữ các service đang chạy cũng như trạng thái của service đó trên máy tính.
- Bằng cách sử dụng công cụ "sc" (Service controller) và truyền tham số "query" (Liệt kê các dịch vụ đang chạy và trạng thái của nó), ta có thể yêu cầu liệt kê ra tất cả dịch vụ và trạng thái của hệ thống.
- Sau đó, ta chỉ cần gửi dữ liệu đọc được cho client thông qua cách tương tự như chức năng listing apps.

6.2.3 Chức năng shutdown

- Khi nhận được lệnh shutdown, server sẽ gửi một phản hồi rằng nó đã nhận được lệnh bằng câu "Server is shutting down..."
- Tiếp theo, bằng cách sử dụng hàm Start("shutdown", "/s /t 10") trong lớp Process, server sẽ đợi 10 giây (chờ cho thông điệp thông báo đã tắt máy đến client) rồi tắt.

6.2.4 Chức năng chụp màn hình

- Ban đầu, ta khởi tạo đối tượng hình ảnh (Bitmap) có tên là screenshot có kích thước bằng với màn hình chính của máy tính, đối tượng này cho phép thực hiện các thao tác trên pixel.
- Tiếp theo, Tạo đối tượng Graphics từ Bitmap để thực hiện thao tác vẽ trên bitmap này.
- Sau đó, ta chụp màn hình bằng lệnh CopyFromScreen, toàn bộ hình ảnh màn hình sẽ được lưu vào đối tượng screenshot được khởi tạo trước đó.
- Bằng việc khởi tạo đối tượng MemoryStream, ta có thể ghi lại hình ảnh đã chụp được dưới dạng file ảnh .png. Bước cuối cùng, ta chuyển đổi dữ liệu trong memory stream thành mảng byte rồi gửi đến client.
- Sau khi thực hiện gửi dữ liệu xong, ta cần giải phóng đối tượng graphic, đối tượng screenshot và đóng luồng bộ nhớ memory stream.

6.2.5 Chức năng chụp bằng webcam

- Ta cũng tạo một đối tượng Processˆ có tên là cameraCapturePRocess để chạy tiến trình bên ngoài, sau đó sử dụng công cụ chụp ảnh bằng webcam có tên là "CommandCam" để chụp ảnh, bước đọc dữ liệu và ghi vào luồng tương tự như khi chụp ảnh màn hình

6.2.6 Chức năng quay video bằng webcam

- Khi nhận lệnh có chứa SEND_VIDEO, hàm sẽ vào cả số giây đính kèm trong email để quay video theo thời lượng được chỉ định, nếu không đính kèm số giây hoặc số giây không hợp lệ (lớn hơn 10 giây hoặc bé hơn 1 giây), chương trình sẽ mặc định quay video trong 10 giây.
- Ta dùng lệnh FFMpeg để liệt kê các thiết bị video có sẵn như webcam, sau đó dùng webcam để quay video trong 5 giây và lưu lại dưới dạng tệp output.mp4
- Cụ thể, ta đặt kích thước khung hình video là 640x480 pixel và tốc độ khung hình là 30fps, sử dụng thiết bị có tên HD Webcam làm nguồn video.
- Sau khi hoàn tất việc ghi video vào file .mp4, ta cũng đọc dữ liệu vào mảng Byte rồi ghi nó vào luồng, gửi dữ liệu sang cho client.

6.2.7 Chức năng lấy file

- Khi nhận được lệnh GET_FILE, hàm cũng sẽ kiểm tra rằng fileName có phải giá trị nullptr hay không, nếu có sẽ không giải quyết.
- Hàm cũng đồng thời kiểm tra fileName(đường dẫn) có tồn tại hay không, nếu không thì sẽ gửi thông báo lỗi cho client.
- Khi đã đảm bảo file tồn tại trong máy server, hàm sẽ tính toán kích thước file và gửi đi trước, sau đó đến tên file và cuối cùng là nội dung file.

6.2.8 Chức năng xóa file

- Khi nhận được lệnh "DELETE_FILE", hàm cũng sẽ kiểm tra đường dẫn trong thông điệp có nhận giá trị null hay không. Nếu không, đường dẫn tới file sẽ được kiểm tra liệu có tồn tại trong máy server hay không. Nếu có, hàm sẽ thực hiện xóa file và gửi thông điệp chứa byte 1 ra hiệu rằng đã xóa file thành công cho client, ngược lại gửi byte 0 báo lệnh xóa file bị lỗi cho client.

6.2.9 Chức năng mở app

- Khi nhận được lệnh "START_APP", hàm cũng sẽ kiểm tra đường dẫn đến file exe của app đượng chỉ định xem có phải giá trị null không, nếu là null thì sẽ không thực thi
- Ngược lại, hàm sẽ tạo một đối tượng ProcessStartInfoˆ startInfo, rồi gán cho thuộc tính FileName của đối tượng mới tạo này là đường dẫn tới file exe. Sau đó, hàm gọi phương thức Start trong lớp Processs để mở app. Khi mở app thành công, hàm gửi về cho client một thông điệp phản hồi chứa byte 1 ra hiệu đã mở app thành công. Ngược lại hàm gửi về byte 0 cho biết rằng việc mở app đã thất bại.

6.2.10 Chức năng tắt app

- Khi nhận lệnh "STOP_APP", hàm cũng thực hiện việc kiểm tra các đường dẫn như các chức năng trên. Khác ở chỗ, nếu hàm đọc vào một đường dẫn tuyệt đối, nó sẽ chỉ lấy tên file thực thi bằng cách dùng hàm GetFileName trong lớp Path, vì việc dừng một tiến trình trong hàm này chỉ yêu cầu tên file thực thi.
- Sau khi tắt app, hàm đọc kết quả và nếu có lỗi, nó gửi thông điệp chứa byte 0 cho client để ra hiệu, ngược lại gửi byte 1 để báo đã tắt app thành công

6.2.11 Chức năng bật và tắt service

- Chức năng này cũng tương tự như chức năng start_app và stop_app.

6.2.12 Hàm khởi chạy server

- Tạo một đối tượng TcpListener để lắng nghe các kết nối đến trên cổng 12345. IPAddress::Any có nghĩa là server sẽ lắng nghe trên tất cả các địa chỉ IP khả dụng trên máy chủ.
- listener->Start(): Bắt đầu quá trình lắng nghe kết nối từ các client.
- Tiếp theo, ta khởi tạo vòng lặp while true, trong đó server sẽ chấp nhận kết nối từ mọi client, đồng thời đọc địa chỉ IP và cổng mà client đang kết nối vào một đối tượng có kiểu IPEndPoint^ là clientEndPoint.
- Tiếp theo, ta khởi tạo đối tượng có kiểu NetworkStream^ là stream để lấy stream giao tiếp với client thông qua phương thức client->GetStream(). Bước đầu tiên là phải xác nhận mật khẩu với client (với mật khẩu xác thực đã được lưu trong database và gán vào biến toàn cục User::Password) khi chương trình chạy. Nếu password không đúng thì ta sẽ gửi lại phản hồi ra hiệu mật khẩu sai cho client bằng cách gửi byte 0 cho client và đóng kết nối. Ngược lại ta sẽ gửi thông điệp xác nhận đúng mật khẩu bằng byte 1 rồi bắt đầu đọc lệnh từ client gửi qua. Cứ mỗi khi xử lý xong lệnh và gửi phản hồi cho client, hàm sẽ đóng kết nối TCP và quay lại vòng lặp, đồng ý kết nối với các client muốn gửi thông điệp.

6.3 Client-side

6.3.1 Hàm tìm thư mục inbox - FindInboxFolder

- Hàm này nhận tham số đầu vào là một mảng kiểu Imap4Folder^ đại diện cho thư mục trong giao thức IMAP.
- Tiếp theo, hàm sẽ chạy một vòng lặp để duyệt qua hết tất cả các thư mục có trong hệ thống, nếu tìm thấy thư mục nào có tên "INBOX" thì trả về thư mục, ngược lại trả về nullptr.

6.3.2 Hàm lưu lại email đã xử lý - SaveProcessedEmail

- Hàm này nhận vào hai tham số đầu vào là String^ messageId và String^ processedEmailsFile, là ID của email vừa xử lý đường dẫn tới tệp văn bản nơi lưu trữ danh sách các email đã xử lý.
- Phương thức tĩnh File::AppendAllText dùng để mở chế độ ghi (append). Nếu tệp không tồn tại, phương thức sẽ tự động tạo tệp mới, và messageId sẽ được ghi ở một dòng mới trong file processedEmailsFile. Nhờ vậy, mỗi khi cần kiểm tra một email đã được xử lý hay chưa, ta chỉ cần duyệt qua file này và kiểm tra ID của mail có nằm trong đó hay không.

6.3.3 Hàm kiểm tra một email đã được process hay chưa - hàm IsEmailProcessed

- Hàm này nhận vào messageID của email cần kiểm tra và thư mục chứa các mail là processedEmailsFile.
- Nếu không tồn tại thư mục processedEmailsFile nghĩa là email này chưa được xử lý. Ngược lại, kiểm tra messageID có nằm trong thư mục processedEmailsFile không, nếu không thì đây là email chưa được xử lý, nếu có thì ta không xử lý email này nữa.

6.3.4 Chức năng trích xuất chi tiết email - hàm ExtractEmailDetails

- Hàm này nhận vào tham số là nội dung email có kiểu là Stringvà đã được xử lý dấu xuống dòng ở cuối văn bản.
- Đầu tiên, nó kiểm tra email có hợp lệ hay không (kiểm tra xem có rỗng hay không đủ thông tin địa chỉ IP, mật khẩu server và các lệnh), nếu không hợp lệ, hàm trả về con trỏ NULL
- Ngược lại, hàm bắt đầu trích xuất các dòng của email để lấy thông tin, mỗi lần trích xuất nó sẽ loại bỏ các dấu xuống dòng bằng phương thức Trim() để tránh gây nhiễu. Mỗi dòng của email sẽ là một phần tử kiểu chuỗi được lưu trong một mảng. Sau khi trích xuất xong, hàm trả về mảng đó

6.3.5 Chức năng xử lý xác nhận mật khẩu với server - hàm HandleVerification

- Hàm này trả về kiểu boolean để nhận phản hồi với server rằng mật khẩu có đúng hay không bằng việc nhận vào tham số NetworkStreamstream là luồng dữ liệu cần đọc.
- Trong trường hợp không thể đọc được dữ liệu từ luồng, luồng rỗng hoặc byte đọc được là 0 thì trả về false, ngược lại trả về true xác nhận rằng mật khẩu chính xác.

6.3.6 Chức năng gửi email phản hồi - Hàm SendEmail

- Hàm số này nhận vào các tham số là String^ recipient, String^ Credent, String^ app_password, String^ subject, String^ body, String^ attachmentPath, lần lượt là email nhận phản hồi, email được ủy quyền để gửi mail, mật khẩu ứng dụng, tiêu đề email, nội dung email và đường dẫn đến tệp đính kèm.
- Hàm này có chức năng gửi lại phản hồi cho người dùng đã ra lệnh thông qua giao thức SMTP.
- Cụ thể, hàm sẽ tạo cấu hình SMTP với SMTP server là gmail với địa chỉ máy chủ là smtp.gmail.com và cổng nhận là 587
- Để thực hiện việc gửi email, ta cần cho phép chương trình truy cập vào quyền gửi email của email được chỉ định nào đó (Mà cụ thể là email đã được đăng ký với mật khẩu ứng dụng khi chương trình được khởi động), dòng smtpClient->EnableSsl = true cũng đồng ghời thiết lập kết nối bảo mật để gửi email.
- Tiếp theo, hàm sẽ tạo email để gửi bằng cách khởi tạo một đối tượng MailMessage() mới, thiết lập bên gửi là email đã được đăng ký mật khẩu ứng dụng và bên nhận là email ra lệnh, sau đó nó gán tham số mailMessage->Body là tham số body được truyền vào.
- Nếu chuỗi attachmentPath khác NULL, nghĩa là hàm gửi cần đính kèm tệp trong email, nó sẽ đọc attachmentPath rồi gửi kèm với email bằng phương thức Attachment.
- Sau khi hoàn thành việc chuẩn bị nội dung email, ta sẽ gọi hàm smtmpClient->send(mailMessage) để gửi mail đi và in ra trên màn hình của máy client rằng email đã được gửi đi thành công

6.3.7 Chức năng xử lý phản hồi từ server - Hàm handleResponse

- Hàm handleResponse nhận vào các tham số là String^ command, NetworkStream^ stream và String^ recipientEmail lần lượt đại diện cho câu lệnh đã được nhận, luồng mà nó lấy dữ diệu, và Email phản hồi
- Trong từng trường hợp khác nhau mà hàm handleResponse sẽ xử lý từng loại dữ liệu khác nhau trước khi đóng gói lại và gửi cho email.
- Với trường hợp dữ liệu trong luồng là ảnh chụp màn hình, ảnh chụp từ webcam, một biến bytesRead có kiểu int sẽ được tạo để đọc 4 bytes đầu tiên nhằm xác định kích thước của dữ liệu. Nếu không nhận đủ 4 bytes thì việc nhận dữ liệu sẽ bị hủy. Ngược lại, ta tiến hành đọc các bytes trong luồng để tạo thành ảnh bằng vòng while (Đọc đến khi tổng lượng dữ liệu bằng dataSize). Sau khi việc

đọc dữ liệu từ luồng hoàn tất, dữ liệu sẽ được ghi vào tệp tại máy client với định dạng tương ứng (png hay .mp4) với tên là ngày giờ được ghi dữ liệu. Bằng hàm sendEmail đã được định nghĩa ở trên, ảnh hoặc video sẽ được gửi tới user thông qua gmail.

- Đối với lệnh GET_FILE, dữ liệu nhận được cũng được ghi lại tại máy client, tuy nhiên, do lúc yêu cầu lệnh người nhận nhập vào đường dẫn tuyệt đối của file tại máy server, do đó, sau khi lấy file, để lưu tại máy client, ta cần biến đổi đường dẫn file này một chút để nó không gây lỗi khi lưu tại máy client. Việc này có thể được thực hiện nhờ hàm GetFileName trong lớp Path, ta chỉ cần lấy tên file và lưu lại file trong thư mục của chương trình. Sau khi lưu xong, file sẽ được đính kèm vào email và gửi cho người dùng.
- Đối với các lệnh như DELETE_FILE, START_APP, STOP_APP, START_SERVICE, STOP_SERVICE, SHUDTDOWN thông điệp trả về từ server là một byte 1 (thực hiện thành công) hoặc byte 0 (thực hiện không thành công). Khi nhận được thông điệp từ server, client sẽ gửi thông báo cho user rằng đã thực hiện lệnh thành công hay chưa.

6.3.8 Hàm khởi chạy client

- Hàm này nhận vào các tham số là email để nhận lệnh, mật khẩu ứng dụng, email được cấp quyền gửi mail ra lệnh cho máy tính và giao thức Imap4.
- Khi hàm này được gọi, nó bắt đầu tính giờ được khởi chạy bằng cách sử dụng một biến startTime có kiểu DateTime. Sau đó, nó cũng khởi tạo một biến kiểu chuỗi là processedEmailsFile để lấy đường dẫn của tệp chứa các id của các email đã được xử lý.
- Tiếp theo, chương trình sẽ khởi tạo một đối tượng mail server có kiểu là MailServer^ có địa chỉ là imap.gmail.com, truy cập vào hòm thư của email được chỉ định để nhận mail thông qua mật khẩu ứng dụng bằng giao thức Imap4. Kết nối này được thiết lập thông qua cổng 993 có sử dụng SSL.
- Tiếp theo, chương trình sẽ liên tục đọc các mail mới được gửi tới bằng cách sử dụng vòng lặp while vô hạn (while (true)). Trong vòng lặp này, nó sẽ liên tục truy cập vào thư mục inbox bằng hàm FindInboxFolder đã được định nghĩa từ trước. Sau đó, nó sẽ lấy các email trong thư mục inbox rồi xử lý từng email.
- Các email đã được xử lý sẽ được lưu id vào tệp processed_emails.txt, nó sẽ lần lượt kiểm tra các email đọc được đã được xử lý hay chưa, nếu rồi thì bỏ qua, nếu chưa thì thực hiện các bước tiếp theo.
- Khi email đọc được có thời gian nhận là trước khi chương trình client được khởi chạy, nó sẽ dừng việc kiểm tra tiếp tục các email và chờ email mới được gửi đến.
- Khi chương trình kiểm tra được một email hợp lệ, nghĩa là email đó được gửi bởi email được ủy quyền và có thời gian nhận là sau khi chương trình được chạy, chương trình sẽ tiến hành trích xuất các nội dung trong email đó bằng hàm ExtractEmailDetails đã được định nghĩa trước đó. Hàm này sẽ trả về các chuỗi chứa địa chỉ ip của máy server, mật khẩu xác nhận với server và lệnh cần thực thi. Sau khi trích xuất xong, nó sẽ đánh dấu email đã được xử xử lý bằng cách lưu id của mail vào thư mục processed_email.
- Trong trường hợp các nội dung của email bị rỗng, hàm sẽ gửi thông báo lỗi cho user yêu cầu gửi lại.
- Khi các nội dung trong mail được đảm bảo đúng cú pháp, chương trình sẽ khởi tạo một đối tượng client có kiểu là TcpClient^ và thiết lập một kết nối qua cổng 12345 tới địa chỉ ip của server, đồng thời cũng khởi tạo một luồng kiểu NetworkStream^ để lấy thông điệp từ luồng. Sau đó, nó gửi vào luồng mật khẩu xác nhận với server rồi chờ phản hồi. Nếu phản hồi từ server là byte 0, nghiax là mật khẩu sai, chương trình sẽ gửi lại cho người dùng thông điệp báo rằng mật khẩu sai và yêu cầu gửi lại mail khác, rồi tiến đến xử lý mail tiếp theo trong hòm thư, ngược lại nếu mật khẩu đúng thnó sẽ gửi các lệnh cho server.

- Bằng việc sử dụng hàm handleResponse được định nghĩa ở trên, ta có thể nhận phản hồi từ server và trả kết quả cho người dùng. Sau khi đã thực hiện xong, kết nối TCP được ngắt.
- Nếu có bất cứ lỗi gì trong quá trình trên, chương trình sẽ gửi email cho người dùng báo lỗi không thể kết nối.
- Sau khi xử lý xong mail, mail sẽ được lưu vào processed_mail.txt để đánh dấu đã đọc mail, chương trình sẽ tiếp tục kiểm tra các email khác trong hòm thư.

6.4 Hàm Main

Trong phần này, chúng ta sẽ giải thích hàm `main`, điểm khởi đầu của ứng dụng, nơi mà form chính được khởi tạo và chạy.

6.4.1 Hàm `main`

- Hàm `main` được đánh dấu với thuộc tính `[STAThreadAttribute]`, cho phép ứng dụng sử dụng các thành phần giao diện người dùng trong một luồng đơn.
- Hàm này bắt đầu bằng việc kích hoạt các kiểu hiển thị hình ảnh cho ứng dụng bằng cách gọi `Application::EnableVisualStyles()`.
- Tiếp theo, `Application::SetCompatibleTextRenderingDefault(false)` được gọi để thiết lập chế độ kết xuất văn bản cho các điều khiển.

6.4.2 Khởi tạo form

- Một đối tượng `MyForm` được khởi tạo từ namespace `LoginForm`. Đây là form chính của ứng dụng mà người dùng sẽ tương tác.
- Biến `form` sẽ lưu trữ thể hiện của lớp `MyForm`.

6.4.3 Chạy ứng dụng

- Cuối cùng, hàm `Application::Run(%form)` được gọi để bắt đầu chạy ứng dụng với form đã khởi tạo.
- Điều này sẽ mở form và bắt đầu vòng lặp xử lý sự kiện cho ứng dụng, cho phép người dùng tương tác với nó.

FrontEnd

6.6 LoginForm

Trong phần này, chúng ta sẽ giải thích các hàm chính trong lớp `LoginForm1`, được sử dụng để tạo một form đăng nhập người dùng.

6.6.1 Giải thích các hàm trong `MyForm`

- **Hàm `button1_Click`:**

  Hàm này được gọi khi người dùng nhấn vào nút "Log In".

  1. Kiểm tra xem các trường `textBox1` (Email) và `textBox2` (Password) có được điền hay chưa.
  2. Kiểm tra trạng thái checkbox `checkBox1` (đồng ý với điều khoản).
  3. Kết nối với cơ sở dữ liệu để xác minh thông tin đăng nhập:

  - Nếu thông tin chính xác, tải dữ liệu người dùng (ID, Email gửi/nhận, Application Key) và chuyển sang form `SelectionForm`.
  - Nếu thông tin sai, hiển thị thông báo lỗi.

- **Hàm `buttonRegister_Click`:**

  Hiển thị form đăng ký người dùng mới (`RegisterForm1`).

  - Khi form đăng ký đóng lại, `MyForm` sẽ được hiển thị lại.

- **Hàm `buttonShowPassword_Click`:**

  Hiển thị hoặc ẩn mật khẩu trong trường `textBox2` bằng cách thay đổi thuộc tính `PasswordChar`.

  - Biểu tượng nút thay đổi từ `" "` (hiển thị) sang `"***"` (ẩn).

- **Hàm `MyForm_Load`:**

  Được gọi khi form `MyForm` tải:

  - Tăng độ mờ (`Opacity`) từ 0 lên 1 để tạo hiệu ứng chuyển đổi mềm mại.
  - Cấu hình các cài đặt mặc định cho form.

- **Hàm xử lý sự kiện `textBox1_Enter` và `textBox1_Leave`:**

  Đổi màu nền của `textBox1` để báo hiệu người dùng khi trường được chọn hoặc bỏ chọn.

  - Màu nền đổi sang `LightYellow` khi được chọn.
  - Màu nền trở lại `DarkSalmon` khi bỏ chọn.

- **Hàm xử lý sự kiện `textBox2_Enter` và `textBox2_Leave`:**

  Tương tự với `textBox1`, nhưng áp dụng cho trường mật khẩu `textBox2`.

- **Hàm xử lý sự kiện chuột với `button1`:**

  Các hàm thay đổi màu sắc của nút "Log In" dựa trên hành động của chuột:

  - `button1_MouseEnter`: Đổi màu nền thành `DodgerBlue` khi chuột di chuyển vào nút.
  - `button1_MouseLeave`: Trả màu nền về `LightCyan` khi chuột rời nút.
  - `button1_MouseDown`: Đổi màu nền thành `RoyalBlue` khi nút được nhấn.
  - `button1_MouseUp`: Trả màu nền về `LightCyan` khi nút được thả.

- **Hàm `label5_Click`:**

  Xử lý khi người dùng nhấn vào dòng "Terms and Policy". Hàm hiện tại chưa được triển khai chi tiết.

6.7 RegisterForm

Trong phần này, chúng ta sẽ giải thích các hàm chính trong lớp `RegisterForm1`, được sử dụng để tạo một form đăng ký người dùng.

6.7.1 Hàm khởi tạo - RegisterForm1

- Hàm này được gọi khi một đối tượng `RegisterForm1` được khởi tạo. Nó thực hiện việc khởi tạo các thành phần của form và thiết lập vị trí bắt đầu của form ở giữa màn hình.
- Timer được khởi tạo để theo dõi độ mạnh của mật khẩu và thiết lập thời gian tick cho timer.

6.7.2 Hàm `InitializeComponent`

- Hàm này chịu trách nhiệm khởi tạo và cấu hình tất cả các thành phần giao diện người dùng của form.
- Nó bao gồm việc thiết lập các thuộc tính cho các điều khiển như `Button`, `TextBox`, `Label`, và `ProgressBar`.
- Các sự kiện như `Click` cho các nút và `TextChanged` cho các ô nhập liệu cũng được đăng ký tại đây.

6.7.3 Hàm `showPasswordCheckBox_CheckedChanged`

- Hàm này được gọi khi người dùng thay đổi trạng thái của checkbox để hiển thị mật khẩu.
- Nếu checkbox được chọn, ký tự mật khẩu sẽ được hiển thị, ngược lại, nó sẽ được ẩn đi bằng ký tự `*`.

6.7.4 Hàm `UpdatePasswordStrength`

- Hàm này được gọi mỗi khi nội dung của ô nhập mật khẩu thay đổi. Nó tính toán độ mạnh của mật khẩu dựa trên các tiêu chí như độ dài, sự xuất hiện của chữ hoa, chữ thường và số.
- Độ mạnh được biểu thị bằng một giá trị từ 0 đến 100 và cập nhật vào thanh tiến trình `passwordStrengthBar`.
- Nếu timer chưa chạy, nó sẽ bắt đầu timer để cập nhật thanh tiến trình một cách mượt mà.

6.7.5 Hàm `OnPasswordStrengthTimerTick`

- Hàm này được gọi mỗi khi timer tick. Nó điều chỉnh giá trị của thanh tiến trình `passwordStrengthBar` để đồng bộ với độ mạnh của mật khẩu.
- Nếu giá trị của thanh tiến trình đạt đến độ mạnh mục tiêu, timer sẽ dừng lại.

6.7.6 Hàm `ValidateEmail`

- Hàm này nhận vào một chuỗi email và kiểm tra xem nó có hợp lệ hay không bằng cách sử dụng biểu thức chính quy.
- Nếu email hợp lệ, hàm trả về `true`, ngược lại trả về `false`.

6.7.7 Hàm `ValidateInputFields`

- Hàm này kiểm tra tất cả các trường nhập liệu trong form để đảm bảo rằng chúng hợp lệ trước khi thực hiện đăng ký.
- Nếu một trường không hợp lệ, màu nền của trường đó sẽ được thay đổi để phản ánh tình trạng không hợp lệ.

6.7.8 Hàm `registerButton_Click`

- Hàm này được gọi khi người dùng nhấn nút đăng ký. Nó thực hiện kiểm tra tính hợp lệ của các trường nhập liệu và thông báo cho người dùng nếu có lỗi.
- Nếu tất cả các trường hợp lệ, nó sẽ thực hiện chèn dữ liệu vào cơ sở dữ liệu bằng cách sử dụng `SqlConnection` và `SqlCommand`.
- Kết quả của thao tác chèn sẽ được thông báo cho người dùng.

6.7.9 Hàm `cancelButton_Click`

- Hàm này được gọi khi người dùng nhấn nút hủy. Nó hiển thị một hộp thoại xác nhận để người dùng có chắc chắn muốn thoát không.
- Nếu người dùng xác nhận, form sẽ đóng lại.

6.7.10 Hàm `RegisterForm1_Load`

- Hàm này được gọi khi form được tải. Nó thiết lập độ mờ ban đầu của form và khởi động một timer để tạo hiệu ứng fade-in cho form.

6.7.11 Hàm `FadeInTimer_Tick`

- Hàm này được gọi bởi timer để tăng độ mờ của form cho đến khi đạt được giá trị tối đa là 1.0.
- Khi độ mờ đạt giá trị tối đa, timer sẽ dừng lại.

6.7.12 Hàm `AppendLog`

- Hàm này được sử dụng để thêm một thông điệp vào hộp văn bản ghi log (`logTextBox`).
- Nếu phương thức này được gọi từ một luồng khác (không phải luồng giao diện người dùng), nó sẽ sử dụng `Invoke` để đảm bảo rằng việc cập nhật giao diện người dùng diễn ra trên luồng chính.
- Nếu không, thông điệp sẽ được thêm trực tiếp vào `logTextBox` cùng với một dòng mới.

6.7.13 Hàm `OnDoWork`

- Hàm này được gọi khi `BackgroundWorker` bắt đầu làm việc. Nó sẽ thực hiện nhiệm vụ chính của ứng dụng là xử lý email.
- Phương thức này gọi hàm `RunClient` với các tham số đã được khởi tạo trong hàm khởi tạo của lớp `Client`.

6.8 Giao Diện Client

6.8.1 Hàm `OnRunWorkerCompleted`

- Hàm này được gọi khi `BackgroundWorker` hoàn thành công việc của nó.
- Nó sẽ gọi hàm `AppendLog` để ghi lại thông báo rằng quá trình xử lý email đã hoàn tất.

6.8.2 Hàm `clearLogButton_Click`

- Hàm này được gọi khi người dùng nhấn nút "Clear Log".
- Nó sẽ xóa toàn bộ nội dung trong hộp văn bản ghi log (`logTextBox`).

6.8.3 Hàm `switchToServerButton_Click`

- Hàm này được gọi khi người dùng nhấn nút "Switch to Server".
- Chức năng cụ thể của hàm này chưa được mô tả trong đoạn mã, nhưng nó thường sẽ chứa logic để chuyển đổi giao diện hoặc thực hiện một hành động liên quan đến server.

6.9 Giao Diện Server

6.9.1 Hàm `UpdateCommunicationLog`

- Hàm này được sử dụng để cập nhật log giao tiếp trong hộp văn bản `textBoxServerOutput`.
- Nếu hàm được gọi từ một luồng khác (không phải luồng giao diện người dùng), nó sẽ sử dụng `Invoke` để đảm bảo rằng việc cập nhật giao diện người dùng diễn ra trên luồng chính.
- Nếu không, thông điệp sẽ được thêm trực tiếp vào `textBoxServerOutput` cùng với một dòng mới.

6.9.2 Hàm `StartServer`

- Hàm này khởi động một `TcpListener` để lắng nghe các kết nối đến từ client.
- Khi một client kết nối, nó sẽ chấp nhận kết nối và đọc lệnh từ luồng mạng.
- Sau khi nhận được lệnh, hàm sẽ gọi hàm `ExecuteCommand` để xử lý lệnh và gửi phản hồi về cho client.
- Quá trình này sẽ lặp lại cho đến khi server dừng lại.

6.9.3 Hàm `buttonStartServer_Click`

- Hàm này được gọi khi người dùng nhấn nút "Start Server".
- Nó sẽ cập nhật trạng thái của server và khởi động một luồng mới để chạy hàm `StartServer`.
- Trạng thái của server sẽ được hiển thị trên giao diện người dùng.

6.9.4 Hàm `buttonStopServer_Click`

- Hàm này được gọi khi người dùng nhấn nút "Stop Server".
- Nó sẽ cập nhật trạng thái của server và thông báo rằng server đã dừng hoạt động.

6.9.5 Hàm `checkBoxDarkMode_CheckedChanged`

- Hàm này được gọi khi người dùng thay đổi trạng thái của checkbox "Dark Mode".
- Nếu checkbox được chọn, giao diện sẽ chuyển sang chế độ tối; nếu không, giao diện sẽ trở về chế độ sáng.
- Các thuộc tính màu sắc của các thành phần giao diện sẽ được cập nhật tương ứng.

6.9.6 Hàm `switchToClientButton_Click`

- Hàm này được gọi khi người dùng nhấn nút "Switch to Client".
- Nó sẽ ẩn form server và khởi tạo một instance của lớp `Client` với các thông tin người dùng đã lưu.
- Cuối cùng, nó sẽ hiển thị form client cho người dùng.

6.10 Database

6.10.1 Giới thiệu

Cơ sở dữ liệu là một thành phần thiết yếu trong ứng dụng của chúng em, cho phép lưu trữ và quản lý thông tin người dùng một cách hiệu quả. Chúng em đã chọn sử dụng Microsoft SQL Server trên nền tảng Azure để tận dụng khả năng mở rộng, độ tin cậy và bảo mật mà dịch vụ này cung cấp. Cơ sở dữ liệu không chỉ giúp chúng em duy trì thông tin người dùng mà còn hỗ trợ các tính năng như xác thực và phân quyền truy cập, đảm bảo rằng dữ liệu được bảo vệ và chỉ có thể truy cập bởi những người dùng hợp lệ.

6.10.2 Kiến trúc cơ sở dữ liệu

Cơ sở dữ liệu của chúng em được thiết kế với cấu trúc đơn giản nhưng hiệu quả, bao gồm các bảng chính như sau:

- **Users**: Bảng này lưu trữ thông tin người dùng với các trường như `Id`, `Sender_Email`, `Receiver_Email`, `Application_Key`, và `Password`.
- **Relationships**: Mối quan hệ giữa các bảng sẽ được thiết lập thông qua khóa chính và khóa ngoại, mặc dù trong kiến trúc hiện tại, chúng em chỉ sử dụng một bảng duy nhất cho người dùng.
- **Kiểu dữ liệu**: Các kiểu dữ liệu sử dụng trong bảng `Users` bao gồm `VARCHAR` cho email và khóa ứng dụng, và `NVARCHAR` cho mật khẩu, cho phép lưu trữ ký tự Unicode.

6.10.3 Tính năng chính

Cơ sở dữ liệu của chúng em cung cấp nhiều tính năng quan trọng:

- **Truy vấn dữ liệu**: Chúng em sử dụng các truy vấn SQL để xác thực người dùng và lưu trữ thông tin một cách hiệu quả. Ví dụ, khi người dùng đăng nhập, một truy vấn được thực hiện để kiểm tra xem thông tin xác thực có khớp với dữ liệu trong cơ sở dữ liệu hay không.
- **Lưu trữ dữ liệu**: Tất cả thông tin người dùng được lưu trữ an toàn trong cơ sở dữ liệu, đảm bảo rằng dữ liệu có thể được truy cập và quản lý một cách dễ dàng.
- **Bảo mật**: Chúng em đã thực hiện các biện pháp bảo mật như mã hóa mật khẩu và sử dụng tham số trong truy vấn SQL để ngăn chặn các cuộc tấn công SQL Injection.

6.10.4 Vấn đề và giải pháp

Trong quá trình phát triển, chúng em đã gặp một số thách thức liên quan đến việc quản lý kết nối cơ sở dữ liệu và xử lý lỗi. Một số vấn đề bao gồm:

- **Kết nối không thành công**: Đôi khi, kết nối đến cơ sở dữ liệu Azure có thể thất bại do cấu hình sai hoặc sự cố mạng. Để giải quyết vấn đề này, chúng em đã triển khai các thông báo lỗi chi tiết để người dùng biết được nguyên nhân và hướng dẫn họ kiểm tra lại thông tin.
- **Xử lý lỗi**: Chúng em đã sử dụng khối `try-catch` để bắt và xử lý các ngoại lệ phát sinh trong quá trình thực hiện truy vấn, giúp ứng dụng không bị treo và cung cấp phản hồi thích hợp cho người dùng.

6.10.5 Kết luận

Cơ sở dữ liệu đóng vai trò quan trọng trong ứng dụng của chúng em, cung cấp nền tảng vững chắc cho việc lưu trữ và quản lý thông tin người dùng. Với kế hoạch phát triển trong tương lai, chúng em dự định mở rộng cơ sở dữ liệu để bao gồm nhiều bảng hơn, hỗ trợ các tính năng phức tạp hơn như báo cáo và phân tích dữ liệu. Điều này sẽ giúp cải thiện trải nghiệm người dùng và tối ưu hóa hiệu suất của ứng dụng.