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

MẠNG MÁY TÍNH

BÁO CÁO

XÂY DỰNG PHẦN MỀM ĐIỀU KHIỂN PC TỪ XA QUA WEBSITE

NHÓM 4

G.V Lý thuyết	ThS.	Đỗ Hoàng Cường
G.V Thực hành	ThS.	Huỳnh Thị Bảo Trân
Thành viên	24120001	Phan Tiến Dũng
	24120002	Đinh Đức Hiếu
	24120051	Ngô Thái Hòa

Thông tin chung

Thông tin thành viên:

MSSV	Họ & Tên	Email	Vai trò
24120001	Phan Tiến Dũng	24120001@student.hcmus.edu.vn	Lập trình Socket
24120002	Đinh Đức Hiếu	24120002@student.hcmus.edu.vn	Lập trình Socket
24120051	Ngô Thái Hòa	24120051@student.hcmus.edu.vn	Lập trình Socket

Bảng 1: Thông tin nhóm

Video trình bày sản phẩm

Bảng phân công công việc

STT	Nhiệm vụ	Phụ trách	Tiến độ
1	Lập trình bằng AI	Ngô Thái Hòa	100%
2	Viết báo cáo	Phan Tiến Dũng	100%
3	Chỉnh sửa video	Đinh Đức Hiếu	100%

Bảng 2: Bảng phân công

1 Tổng quan hệ thống

1.1 Giới thiệu và mục tiêu dự án

1.1.1 Mục tiêu chính của hệ thống

Hệ thống Remote Control được thiết kế như một giải pháp toàn diện cho việc điều khiển và giám sát máy tính từ xa thông qua giao diện web hiện đại. Hệ thống sử dụng kiến trúc client-server với công nghệ ASP.NET Core và SignalR, cung cấp khả năng giao tiếp thời gian thực giữa server điều khiển và các agent triển khai trên máy tính mục tiêu.

Mục tiêu chính của hệ thống là cung cấp một nền tảng quản lý tập trung cho phép người quản trị thực hiện các thao tác điều khiển từ xa, giám sát hoạt động hệ thống, và thu thập dữ liệu một cách an toàn và hiệu quả. Hệ thống được thiết kế với kiến trúc module hóa cao, cho phép dễ dàng mở rộng và tích hợp thêm tính năng mới.

1.1.2 Bảng So Sánh Chức Năng Hệ Thống

Nhóm Chức Năng	Tính Năng Cụ Thể	Trạng Thái Triển Khai
Quản lý tiến trình	Liệt kê tiến trình Dừng tiến trình theo PID Hiển thị thông tin chi tiết	**100% Hoàn Thành** - Real-time process listing - Kill process với xác nhận - Multi-process support
Điều khiển ứng dụng	Khởi chạy ứng dụng từ xa Quản lý ứng dụng đang chạy	**100% Hoàn Thành** - Shell execution - Path validation - Error handling
Chụp màn hình	Chụp toàn màn hình Nén ảnh tự động Tải ảnh về máy	**100% Hoàn Thành** - GDI capture - JPEG compression - Base64 streaming - Download capability
Keylogger	Ghi lại phím bấm thời gian thực Xử lý phím đặc biệt Lưu trữ và xuất dữ liệu	**100% Hoàn Thành** - Windows keyboard hook - Modifier key detection - Real-time console display - Export to text file
Webcam Streaming	Stream video thời gian thực Ghi lại video Điều chỉnh chất lượng	**100% Hoàn Thành** - DirectShow capture - Frame rate control (5fps) - Resolution scaling - Recording functionality
Điều khiển hệ thống	Shutdown từ xa Restart từ xa Thoát agent	**100% Hoàn Thành** - System command execution - Safety confirmation - Graceful termination
Quản lý Registry	Thay đổi registry values Hỗ trợ các root keys	**100% Hoàn Thành** - Multi-root support - Automatic type detection - Error handling
Thông tin hệ thống	Thu thập system info Hiển thị dashboard	**100% Hoàn Thành** - Multi-parameter collection - Real-time display - JSON serialization
Web Dashboard	Giao diện điều khiển web Real-time updates Responsive design	**100% Hoàn Thành** - Bootstrap 5 interface - SignalR communication - Mobile responsive

Bảng 3: Bảng tổng hợp các chức năng của hệ thống Remote Control

Hệ thống đã triển khai thành công tất cả các tính năng cốt lõi với độ ổn định cao. Kiến trúc module cho phép từng thành phần hoạt động độc lập, đồng thời tích hợp chặt chẽ thông qua SignalR Hub để đảm bảo trải nghiệm người dùng mượt mà.

1.2 Kiến trúc hệ thống

1.2.1 Kiến trúc tổng thể

Hệ thống được xây dựng theo mô hình client-server với kiến trúc ba tầng rõ ràng: tầng trình bày (Presentation Layer), tầng nghiệp vụ (Business Layer), và tầng dữ liệu (Data Layer). Mỗi tầng có nhiệm vụ riêng biệt và giao tiếp thông qua các giao thức chuẩn hóa.

Tầng trình bày bao gồm Web Dashboard được xây dựng bằng HTML5, Bootstrap 5 và JavaScript, cung cấp giao diện người dùng trực quan và dễ sử dụng. Dashboard tương tác với server thông qua SignalR để nhận và hiển thị dữ liệu thời gian thực.

Tầng nghiệp vụ được triển khai trên ASP.NET Core Server, xử lý tất cả logic nghiệp vụ, quản lý kết nối client, định tuyến lệnh và xử lý dữ liệu. SignalR Hub đóng vai trò trung tâm trong việc quản lý kết nối và truyền tải thông điệp.

Tầng dữ liệu bao gồm các agent client chạy trên máy tính mục tiêu, thực thi các lệnh từ server và thu thập dữ liệu hệ thống. Mỗi agent là một ứng dụng Windows độc lập, được thiết kế để hoạt động ổn định với khả năng tự kết nối lại khi mất kết nối.

1.2.2 Công nghệ và môi trường phát triển

Hệ thống được phát triển bằng ngôn ngữ C# với .NET 10.0, tận dụng các tính năng hiện đại của nền tảng .NET để xây dựng ứng dụng cross-platform. Server sử dụng ASP.NET Core để xây dựng web API và SignalR Hub, trong khi client sử dụng Windows Forms cho các thao tác hệ thống cấp thấp.

Các thư viện và framework chính bao gồm:

- **ASP.NET Core 10.0**: Web server và SignalR framework
- **SignalR 6.0**: Real-time communication
- **AForge.Video 2.2.5**: Webcam capture và xử lý video
- **Bootstrap 5.3.0**: Frontend framework cho dashboard
- **System.Drawing.Common**: Xử lý ảnh và đồ họa

Môi trường phát triển chính là Visual Studio 2022 với .NET 10.0 SDK. Hệ thống hỗ trợ triển khai trên cả Windows Server và các nền tảng .NET Core khác. Build process sử dụng MSBuild với cấu hình cho cả Debug và Release modes.

1.2.3 Các module chính của hệ thống

SignalR Communication Hub

ControlHub là trái tim của hệ thống, quản lý tất cả kết nối giữa server và clients. Hub sử dụng ConcurrentDictionary để lưu trữ thông tin các agent đang kết nối, cung cấp các phương thức hai chiều cho việc gửi lệnh và nhận dữ liệu.

Hub triển khai hai nhóm phương thức chính:

- **Server-to-Client**: Các lệnh điều khiển gửi từ dashboard đến agent

- **Client-to-Server**: Dữ liệu phản hồi từ agent gửi về dashboard

Mỗi kết nối được gán một ConnectionID duy nhất và lưu trữ thông tin agent bao gồm machine name, user name, thời gian kết nối. Hub hỗ trợ broadcast đến tất cả clients hoặc gửi đến client cụ thể dựa trên ConnectionID.

Webcam Service Module

WebcamService sử dụng AForge.Video.DirectShow để truy cập và điều khiển webcam. Module được thiết kế với các tính năng tối ưu hóa hiệu suất:

- **Auto-resolution selection**: Tự động chọn độ phân giải thấp nhất (<320x240) để giảm bandwidth
- **Frame rate control**: Giới hạn 5fps để giảm tải CPU và network
- **Image compression**: Nén ảnh JPEG với chất lượng 40%
- **Memory management**: Sử dụng using statement và proper disposal để tránh memory leak

Service implement IDisposable pattern để đảm bảo giải phóng tài nguyên camera đúng cách. Cơ chế timeout được áp dụng khi dừng stream để tránh treo ứng dụng.

Keylogger Service Module

KeyLoggerService sử dụng Windows Hook API (WH_KEYBOARD_LL) để bắt sự kiện bàn phím ở mức độ hệ thống. Module chạy trong thread riêng với ApartmentState.STA để đảm bảo hoạt động ổn định của Windows message loop.

Các tính năng chính của keylogger:

- **Modifier key detection**: Phát hiện Shift, Ctrl, Alt, Caps Lock **Special key handling**: Xử lý các phím đặc biệt (Enter, Tab, Arrow keys)
- **Character conversion**: Chuyển đổi key code thành ký tự hiển thị được
- **Thread safety**: Sử dụng lock và volatile flags để đảm bảo thread safety

Service được thiết kế để hoạt động ổn định ngay cả khi có exception, với cơ chế cleanup đảm bảo unhook keyboard khi dừng service.

Registry Management Module

RegistryHelper cung cấp interface đơn giản để thao tác với Windows Registry từ xa. Module hỗ trợ tất cả các root keys chính (HKCU, HKLM, HKCR, HKU, HKCC) và tự động detect kiểu dữ liệu của giá trị.

Tính năng tự động type detection:

- **Integer**: Tự động parse string thành integer
- **Boolean**: Chuyển đổi "true"/"false"thành 1/0
- **Hexadecimal**: Hỗ trợ giá trị hex (0x prefix)
- **String**: Giữ nguyên giá trị string

Module xử lý exception toàn diện và trả về message lỗi chi tiết để debug. Tất cả registry keys được mở với using statement để đảm bảo đóng đúng cách.

Web Dashboard Interface

Dashboard được xây dựng như một Single Page Application (SPA) với các thành phần chính:

- **Real-time Screen Viewer**: Hiển thị màn hình remote với khả năng capture và download
- **Webcam Stream Panel**: Hiển thị video stream từ webcam với controls ghi hình
- **Process Manager Table**: Hiển thị danh sách process với action kill
- **System Control Panel**: Các nút điều khiển hệ thống (shutdown, restart, launch app)
- **Keylogger Console**: Hiển thị keystrokes thời gian thực
- **System Log Panel**: Hiển thị log hệ thống với màu sắc phân loại
- **Statistics Cards**: Thống kê hoạt động hệ thống

Dashboard sử dụng JavaScript SignalR client để kết nối và nhận dữ liệu thời gian thực. Giao diện được thiết kế responsive với Bootstrap 5, hoạt động tốt trên cả desktop và mobile.

1.2.4 Luồng dữ liệu và giao thức

Hệ thống sử dụng SignalR làm giao thức giao tiếp chính, hỗ trợ cả WebSocket và long-polling fallback. Các loại dữ liệu được truyền tải bao gồm:

Command Data Flow

1. User thực hiện action trên dashboard
2. JavaScript gọi sendCommand() với command name và data
3. SignalR client gửi message đến server
4. ControlHub nhận message và gửi đến tất cả agents
5. Agent nhận command và thực thi thông qua Program.cs
6. Kết quả được gửi ngược lại qua Hub đến dashboard

Image/Video Data Flow

1. Agent capture ảnh/video từ hệ thống
2. Dữ liệu được convert sang base64 string
3. Base64 string được gửi qua SignalR Hub
4. Dashboard nhận base64 và hiển thị bằng thẻ <img> hoặc <video>
5. Dữ liệu có thể được lưu cục bộ thông qua download

Real-time Data Flow

- **Keystrokes**: Mỗi phím bấm được gửi ngay lập tức khi được ghi nhận
- **Webcam frames**: Frame mới nhất được gửi mỗi 200ms (5fps)
- **System logs**: Log messages được gửi ngay khi phát sinh
- **Connection status**: Status updates được push khi có thay đổi

1.2.5 Quản lý kết nối và xử lý lỗi

Hệ thống triển khai cơ chế quản lý kết nối mạnh mẽ với các tính năng:

Auto-reconnection Mechanism

- **Client-side reconnection**: SignalR client tự động reconnect với exponential backoff
- **Server-side cleanup**: Hub tự động xóa disconnected clients khỏi danh sách
- **Connection state tracking**: Theo dõi trạng thái kết nối với visual indicators
- **Heartbeat monitoring**: Ping/pong mechanism để phát hiện mất kết nối

Error Handling Strategy

- **Try-catch comprehensive**: Tất cả operations được wrap trong try-catch
- **Graceful degradation**: Hệ thống tiếp tục hoạt động khi một component fail
- **User-friendly error messages**: Hiển thị lỗi dễ hiểu trên dashboard
- **Logging hệ thống**: Ghi log chi tiết cho mọi exception

Resource Management

- **IDisposable pattern**: Tất cả resource-intensive objects implement IDisposable
- **Using statements**: Đảm bảo resource được giải phóng đúng cách
- **Memory optimization**: Giới hạn frame rate và image quality
- **Connection pooling**: SignalR quản lý connection pool tự động

1.2.6 Bảo mật hệ thống

Hệ thống hiện tại được thiết kế cho môi trường trusted network với các tính năng bảo mật cơ bản:

Current Security Implementation

- **Input validation**: Validate tất cả user inputs trên dashboard
- **Path sanitization**: Kiểm tra và sanitize file paths trước khi thực thi
- **Process authorization**: Kiểm tra quyền trước khi kill process
- **Error message sanitization**: Không expose thông tin nhạy cảm trong error messages

Security Recommendations for Production

- **HTTPS enforcement**: Sử dụng SSL/TLS cho tất cả communications
- **Authentication system**: Triển khai JWT hoặc Windows Authentication
- **Authorization framework**: Phân quyền user based trên roles
- **Rate limiting**: Giới hạn request rate để ngăn DoS attacks
- **Audit logging**: Ghi log tất cả actions cho mục đích auditing
- **IP whitelisting**: Chỉ cho phép kết nối từ IP addresses được authorized

Data Privacy Considerations

- **Data encryption**: Mã hóa sensitive data trong transit và at rest
- **Data retention policy**: Xóa dữ liệu cũ tự động theo policy
- **User consent**: Yêu cầu consent trước khi thu thập dữ liệu
- **Compliance**: Tuân thủ các regulations như GDPR, CCPA nếu applicable

1.2.7 Kiến trúc mở rộng và scalability

Hệ thống được thiết kế với khả năng mở rộng cho các deployment quy mô lớn:

Horizontal Scaling Strategies

- **Multiple server instances**: Triển khai nhiều server instances behind load balancer
- **SignalR backplane**: Sử dụng Redis hoặc Azure SignalR Service cho distributed messaging
- **Database persistence**: Lưu trữ connection state và historical data trong database
- **Microservices architecture**: Tách các components thành microservices độc lập

Performance Optimization

- **Image compression algorithms**: Tối ưu compression ratio và quality
- **Binary protocols**: Chuyển từ base64 sang binary protocol cho image/video data
- **CDN integration**: Sử dụng CDN cho static assets của dashboard
- **Caching mechanisms**: Cache frequently accessed data để giảm load server

Monitoring và Maintenance

- **Health checks**: REST endpoint để kiểm tra tình trạng hệ thống
- **Metrics collection**: Thu thập performance metrics (CPU, memory, network)
- **Alerting system**: Gửi alerts khi phát hiện issues
- **Automated deployment**: CI/CD pipeline cho automated testing và deployment

1.2.8 Triển khai thực tế và use cases

Hệ thống có thể được triển khai trong nhiều môi trường khác nhau với các use cases cụ thể:

Enterprise IT Management

- **Remote support**: Hỗ trợ kỹ thuật từ xa cho nhân viên
- **Software deployment**: Cài đặt và cập nhật ứng dụng từ xa
- **System monitoring**: Giám sát tình trạng hệ thống mạng nội bộ
- **Compliance auditing**: Kiểm tra cấu hình hệ thống cho compliance

Educational Environments

- **Classroom management**: Quản lý phòng máy tính trong trường học
- **Student monitoring**: Giám sát hoạt động của học sinh trong giờ học
- **Remote instruction**: Hướng dẫn thực hành từ xa
- **Lab management**: Quản lý phòng lab máy tính

Other Use Cases

- **Digital forensics**: Thu thập evidence với proper legal authorization
- **Penetration testing**: Security testing trong môi trường được kiểm soát
- **Parental control**: Giám sát máy tính gia đình (với consent)
- **Remote administration**: Quản trị hệ thống từ xa cho các chi nhánh

Deployment Scenarios

- **Local network**: Server chạy trong mạng nội bộ, agents kết nối qua local IP
- **Cloud deployment**: Server triển khai trên cloud (Azure, AWS), agents kết nối qua internet
- **Hybrid model**: Kết hợp cả local và cloud deployment
- **Air-gapped networks**: Triển khai trong môi trường không có internet

Hệ thống Remote Control đã chứng minh tính hiệu quả trong việc cung cấp giải pháp điều khiển và giám sát từ xa toàn diện. Với kiến trúc module hóa và khả năng mở rộng, hệ thống có thể đáp ứng được nhu cầu của nhiều môi trường triển khai khác nhau.

2 Cài đặt và khởi chạy

2.1 Yêu cầu hệ thống

2.1.1 Yêu cầu phần cứng

- **CPU**: Intel/AMD x64 hoặc x86 architecture
- **RAM**: Tối thiểu 2GB, khuyến nghị 4GB+
- **Ổ cứng**: 500MB dung lượng trống
- **Webcam**: Hỗ trợ DirectShow (cho tính năng webcam streaming)
- **Mạng**: Ethernet/WiFi connection
- **Màn hình**: Độ phân giải tối thiểu 1024×768

2.1.2 Yêu cầu phần mềm

- **Hệ điều hành**: Windows 10/11 (64-bit hoặc 32-bit)
- **.NET Runtime**: .NET 10.0 Runtime hoặc SDK
- **Visual Studio 2022** (cho development) hoặc dotnet CLI
- **Trình duyệt web**: Chrome/Edge/Firefox phiên bản mới nhất
- **Quyền administrator**: Cho một số tính năng như keylogger, registry

2.2 Hướng dẫn cài đặt

2.2.1 Cài đặt .NET 10.0 Runtime

1. **Tải .NET Runtime**: Truy cập https://dotnet.microsoft.com/download/dotnet/10.0
2. **Chọn phiên bản**: Tải .NET 10.0 Desktop Runtime cho Windows
3. **Cài đặt**: Chạy installer và làm theo hướng dẫn
4. **Xác nhận cài đặt**:

```
dotnet --version
# Kết quả: 10.0.x
```

2.2.2 Clone source code từ repository

1. **Clone repository**:

```
git clone
   https://github.com/phantiendung-fr/RemoteControlSystem.git
cd RemoteControlSystem
```

2. **Cấu trúc thư mục**:

- `RemoteControl.Server/`: Mã nguồn server
- `RemoteControl.Client/`: Mã nguồn client
- `README.md`: Hướng dẫn sử dụng
- `LICENSE`: Giấy phép

2.2.3 Build bằng Visual Studio 2022

1. **Mở solution**:

```
# Mở file solution
RemoteControlSystem.sln
```

2. **Restore packages**:

   - Click chuột phải vào solution → "Restore NuGet Packages"
   - Hoặc sử dụng Package Manager Console:

```
dotnet restore
```

3. **Build solution**:

   - Chọn Configuration: "Release"
   - Chọn Platform: "Any CPU"hoặc "x64"
   - Build → Build Solution (Ctrl+Shift+B)

4. **Output files**:

   - Server: `RemoteControl.Server/bin/Release/net10.0/`
   - Client: `RemoteControl.Client/bin/Release/net10.0-windows/`

2.2.4 Build bằng dotnet CLI

1. **Build server**:

```
cd RemoteControl.Server
dotnet publish -c Release -o ./publish
```

2. **Build client**:

```
cd RemoteControl.Client
dotnet publish -c Release -r win10-x64 -o ./publish
```

3. **Kiểm tra build**:

```
# Server
dir RemoteControl.Server\publish\*.exe

# Client
dir RemoteControl.Client\publish\*.exe
```

2.2.5 Publish tự động với script

1. **Chạy build script**:

```
# Windows
build.bat

# Linux/macOS
chmod +x build.sh
./build.sh
```

2. **Script sẽ thực hiện**:

   - Clean build directories
   - Restore NuGet packages
   - Build server và client
   - Copy dependencies
   - Tạo zip file cho distribution

2.3 Hướng dẫn sử dụng

2.3.1 Khởi động Server

1. **Cách 1: Chạy từ Visual Studio**:

   - Set `RemoteControl.Server` làm startup project
   - Nhấn F5 hoặc Debug → Start Debugging
   - Server sẽ tự động mở trình duyệt tại http://localhost:5000

2. **Cách 2: Chạy từ executable**:

```
cd RemoteControl.Server\publish
.\RemoteControl.Server.exe
```

3. **Cách 3: Chạy với dotnet CLI**:

```
cd RemoteControl.Server
dotnet run
```

4. **Console output**:

```
================================================
    REMOTE CONTROL SERVER
================================================
Server URL: http://localhost:5000
SignalR Hub: http://localhost:5000/controlHub
Dashboard: http://localhost:5000
Health Check: http://localhost:5000/health
================================================
Press Ctrl+C to stop the server
================================================
```

2.3.2 Khởi động Client (Agent)

1. **Cài đặt agent trên máy mục tiêu**:
   - Copy thư mục `RemoteControl.Client/publish` đến máy mục tiêu
   - Chạy file `RemoteControl.Client.exe`
   - Agent sẽ tự động kết nối đến server
2. **Chạy agent từ command line**:

```
.\RemoteControl.Client.exe --server http://your-server:5000
```

3. **Console output khi kết nối thành công**:

```
Đang kết nối tới: http://your-server:5000/controlHub...
--> Đã kết nối thành công!
Máy DESKTOP-ABC123 (Admin) đã online.
```

4. **Chạy agent như Windows Service** (tùy chọn):

```
# Cài đặt service
sc create RemoteControlAgent binPath=
   "C:\Path\To\RemoteControl.Client.exe"

# Khởi động service
sc start RemoteControlAgent

# Dừng service
sc stop RemoteControlAgent
```

2.3.3 Sử dụng Web Dashboard

1. **Truy cập dashboard**:
   - Mở trình duyệt web
   - Truy cập http://localhost:5000 (hoặc server IP)
   - Dashboard sẽ tự động kết nối đến SignalR Hub
2. **Interface components**:
   - **Connection Status**: Hiển thị trạng thái kết nối agent
   - **Remote Screen**: Panel hiển thị màn hình từ xa
   - **Webcam Stream**: Panel hiển thị webcam từ xa
   - **Process Manager**: Quản lý tiến trình
   - **System Controls**: Điều khiển hệ thống
   - **Keylogger**: Console hiển thị keystrokes
   - **System Logs**: Panel hiển thị logs hệ thống
   - **Statistics**: Thống kê hoạt động

Nhóm Lệnh	Chức Năng và Cú Pháp
Process Management	`GetProcesses`: Liệt kê tất cả tiến trình đang chạy `KillProcess <pid>`: Dừng tiến trình theo PID
Application Control	`StartApplication <path>`: Khởi chạy ứng dụng từ đường dẫn `Shutdown`: Tắt máy tính từ xa `Restart`: Khởi động lại máy tính từ xa
Screen Capture	`RequestScreenshot`: Chụp màn hình máy tính từ xa `Download`: Tải ảnh screenshot về máy
Webcam Control	`StartWebcam`: Bắt đầu stream từ webcam `StopWebcam`: Dừng stream từ webcam `StartRecording`: Bắt đầu ghi video `StopRecording`: Dừng ghi video
Keylogger	`StartKeylogger`: Bắt đầu ghi lại keystrokes `StopKeylogger`: Dừng ghi keystrokes `Download Keylog`: Tải file keylog về máy
Registry Control	`SetRegistry <root> <path> <key> <value>`: Thay đổi registry value
System Information	`GetSystemInfo`: Lấy thông tin hệ thống `Ping`: Kiểm tra kết nối agent
Agent Control	`ExitAgent`: Thoát agent từ xa

Bảng 4: Các lệnh điều khiển chính của hệ thống Remote Control

2.3.4 Các tình huống sử dụng phổ biến

Giám sát màn hình từ xa

1. Truy cập dashboard tại `http://server:5000`
2. Click nút "Capture Screen"để chụp màn hình
3. Ảnh sẽ hiển thị trong panel "Remote Screen"
4. Click "Download"để tải ảnh về máy

Stream webcam từ xa

1. Đảm bảo agent đã được cài đặt trên máy có webcam
2. Từ dashboard, click "Start Webcam"
3. Video stream sẽ hiển thị trong panel "Webcam Stream"
4. Click "Start Recording"để bắt đầu ghi video
5. Click "Stop Recording"và "Save Video"để lưu video

Quản lý tiến trình từ xa

1. Từ dashboard, click "Refresh"trong panel "Process Manager"
2. Danh sách tiến trình sẽ hiển thị trong bảng
3. Để dừng tiến trình, click nút "Kill"tương ứng
4. Xác nhận hành động trong hộp thoại

Ghi lại keystrokes

1. Click "Start"trong panel "Key Logger"
2. Keystrokes sẽ hiển thị real-time trong console
3. Click "Stop"để dừng ghi
4. Click "Save"để tải file keylog về máy

2.4 Cấu hình nâng cao

2.4.1 Cấu hình Server

1. **Chỉnh sửa appsettings.json**:

```
{
  "SignalR": {
    "MaxReceiveMessageSize": 52428800, // 50MB
    "ClientTimeoutInterval": 30, // gi y
    "KeepAliveInterval": 15, // gi y
    "HandshakeTimeout": 15 // gi y
  },
  "Server": {
    "Port": 5000,
    "MaxConnections": 100,
    "EnableCompression": true
  }
}
```

2. **Thay đổi port**:
   - Mở file `Program.cs`
   - Tìm dòng: `app.Urls.Add("http://localhost:5000");`
   - Thay đổi port theo nhu cầu
   - Lưu ý: Cần mở port trên firewall

2.4.2 Cấu hình Client

1. **Thay đổi server URL**:
   - Mở file `Program.cs` trong project Client
   - Tìm dòng: `private static string HUB_URL = "http://localhost:5000/controlHub";`

- Thay đổi URL cho phù hợp
- Rebuild client

2. **Cấu hình webcam**:

- Trong file `WebcamService.cs`:

```
        // Thay đổi FPS limit
        if ((DateTime.Now - _lastFrameTime).TotalMilliseconds
            < 200) // 5fps

        // Thay đổi resolution
        int newWidth = (int)(original.Width * ratio);
        int newHeight = (int)(original.Height * ratio);

        // Thay đổi chất lượng nh
        image.Save(ms, encoder, encoderParams); // quality:
            40L
```

2.4.3 Triển khai production

1. **Reverse Proxy với Nginx**:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }
}
```

2. **SSL/HTTPS configuration**:

- Sử dụng Let's Encrypt cho SSL certificate
- Cấu hình trong `Program.cs`:

```
        app.Urls.Add("https://0.0.0.0:5001");
```

3. **Windows Firewall configuration**:

```
# Mở port 5000
netsh advfirewall firewall add rule ^
name="RemoteControl Server" ^
dir=in action=allow protocol=TCP localport=5000

# Mở port cho SignalR WebSocket
netsh advfirewall firewall add rule ^
name="SignalR WebSocket" ^
dir=in action=allow protocol=TCP localport=5001
```

2.5 Xử lý sự cố thường gặp

2.5.1 Client không kết nối được đến Server

- **Kiểm tra kết nối mạng**:

```
ping server-ip
telnet server-ip 5000
```

- **Kiểm tra firewall**:
  - Đảm bảo port 5000 được mở trên server
  - Tạm thời tắt firewall để test
- **Kiểm tra server status**:

```
curl http://server-ip:5000/health
# Kết quả: {"status":"healthy","timestamp":"..."}
```

2.5.2 Webcam không hoạt động

- **Kiểm tra webcam**:
  - Đảm bảo webcam được kết nối và enabled
  - Kiểm tra trong Device Manager
  - Test với ứng dụng khác (Camera app)
- **Kiểm tra permissions**:
  - Đảm bảo ứng dụng có quyền truy cập webcam
  - Trong Windows Settings: Privacy → Camera
- **Kiểm tra drivers**:
  - Update webcam drivers
  - Cài đặt lại drivers nếu cần

2.5.3 Keylogger không ghi được keystrokes

- **Kiểm tra permissions**:
  - Chạy agent với quyền Administrator
  - Đảm bảo không có ứng dụng antivirus chặn hook
- **Kiểm tra Windows Hook**:
  - Keylogger sử dụng low-level keyboard hook
  - Một số ứng dụng có thể chặn hook
  - Thử tạm thời tắt antivirus
- **Kiểm tra thread state**:
  - Đảm bảo hook thread đang chạy
  - Kiểm tra console output của agent

2.5.4 Dashboard không hiển thị dữ liệu real-time

- **Kiểm tra SignalR connection**:
  - Mở Developer Tools (F12)
  - Chọn tab Network → WebSocket
  - Kiểm tra kết nối WebSocket
- **Kiểm tra CORS configuration**:
  - Đảm bảo CORS được cấu hình đúng
  - Kiểm tra console log trong browser
- **Kiểm tra browser compatibility**:
  - Đảm bảo sử dụng browser hỗ trợ WebSocket
  - Thử với Chrome/Edge/Firefox mới nhất

2.6 Bảo mật và an toàn

2.6.1 Các biện pháp bảo mật cơ bản

- **Thay đổi port mặc định**: Không sử dụng port 5000 mặc định
- **Sử dụng HTTPS**: Luôn sử dụng SSL/TLS trong production
- **Firewall configuration**: Chỉ mở port cần thiết
- **Authentication**: Thêm authentication cho dashboard (chưa có sẵn)
- **Logging**: Theo dõi logs để phát hiện hoạt động bất thường

2.6.2 Hardening cho production

1. **Thêm authentication**:

```
// Trong Program.cs
builder.Services.AddAuthentication();
builder.Services.AddAuthorization();
```

2. **Rate limiting**:

```
// Giới hạn request rate
builder.Services.AddRateLimiter(options => {
    options.GlobalLimiter =
       PartitionedRateLimiter.Create<HttpContext, string>(
        httpContext =>
           RateLimitPartition.GetFixedWindowLimiter(
            partitionKey: httpContext.User.Identity?.Name ??
            httpContext.Request.Headers.Host.ToString(),
            factory: partition => new
               FixedWindowRateLimiterOptions {
                AutoReplenishment = true,
                PermitLimit = 100,
```

```
                    QueueLimit = 0,
                    Window = TimeSpan.FromMinutes(1)
                }));
    });
```

3. **Audit logging**:
    - Log tất cả commands được thực thi
    - Log IP address của người dùng
    - Log thời gian và kết quả của mỗi command

2.6.3 Legal considerations

- **Consent**: Luôn có sự đồng ý của người dùng trước khi cài đặt agent
- **Transparency**: Thông báo cho người dùng về việc giám sát
- **Data retention**: Xóa dữ liệu thu thập được khi không cần thiết
- **Compliance**: Tuân thủ các luật về quyền riêng tư (GDPR, CCPA, etc.)

Hệ thống Remote Control được thiết kế để dễ dàng cài đặt và sử dụng, với khả năng mở rộng cho các môi trường production. Luôn đảm bảo tuân thủ các best practices về bảo mật và pháp lý khi triển khai trong môi trường thực tế.

3 Phân tích chi tiết các lệnh

3.1 GetProcesses

1. **Mục đích**: Liệt kê tất cả các tiến trình đang chạy trên máy tính từ xa
2. **Đặc điểm kỹ thuật**:
   - Sử dụng .NET Process API: `Process.GetProcesses()`
   - Lọc các tiến trình hợp lệ (không rỗng tên)
   - Format dữ liệu JSON với cấu trúc: `{ id: PID, name: "ProcessName"}`
   - Sắp xếp theo tên tiến trình (alphabetical order)
   - Truyền qua SignalR với method: `ReceiveProcesses(object processes)`
   - Hiển thị dạng bảng trên dashboard với cột: PID, Tên tiến trình, Actions
3. **Performance considerations**:
   - Asynchronous execution để không block main thread
   - Memory optimization: Chỉ lấy thông tin cần thiết (PID và Name)
   - Error handling: Try-catch bao bọc toàn bộ operation
   - Response timeout: 30 giây

3.2 KillProcess

1. **Mục đích**: Dừng tiến trình từ xa theo Process ID (PID)
2. **Đặc điểm kỹ thuật**:
   - Sử dụng .NET Process API: `Process.GetProcessById(pid)`
   - Phương thức: `process.Kill()` với force termination
   - Validation: Kiểm tra PID hợp lệ và tiến trình tồn tại
   - Logging: Ghi lại tên tiến trình và PID đã bị kill
   - Error handling: Xử lý các exception như Access Denied, Process not found
3. **Security considerations**:
   - Không thể kill system critical processes (cần elevated privileges)
   - User confirmation trước khi thực thi từ dashboard
   - Logging đầy đủ cho audit trail

3.3 StartApplication

1. **Mục đích**: Khởi chạy ứng dụng từ xa trên máy tính mục tiêu
2. **Đặc điểm kỹ thuật**:
   - Sử dụng .NET Process.Start() với ProcessStartInfo
   - Cấu hình: `UseShellExecute = true` để mở file thông thường

- Window style: `ProcessWindowStyle.Normal`
- Path validation: Kiểm tra đường dẫn hợp lệ
- Error handling: Xử lý FileNotFoundException, Win32Exception

3. **Supported formats**:
   - Executable files (.exe, .bat, .cmd)
   - Document files (.docx, .pdf, .xlsx) - mở với ứng dụng mặc định
   - URL và web links
   - Folder paths (mở Explorer)

3.4 RequestScreenshot

1. **Mục đích**: Chụp màn hình máy tính từ xa và gửi ảnh về dashboard
2. **Đặc điểm kỹ thuật**:
   - Sử dụng Windows Forms API: `Screen.PrimaryScreen`
   - Capture method: `Graphics.CopyFromScreen()`
   - Image format: JPEG với chất lượng tự động (20-40%)
   - Resolution: Full screen với auto-scaling
   - Compression: Base64 encoding cho truyền qua SignalR
   - Transfer method: `ReceiveScreenshot(string base64)`
3. **Performance optimization**:
   - Tự động tính chất lượng dựa trên độ phân giải
   - Memory management: Sử dụng using statement cho Bitmap và Graphics
   - Async processing: Không block UI thread
   - Size optimization: Tự động resize cho màn hình lớn

3.5 StartKeylogger / StopKeylogger

1. **Mục đích**: Ghi lại và gửi các phím bấm từ máy tính từ xa
2. **Đặc điểm kỹ thuật**:
   - Sử dụng Windows Hook API: `SetWindowsHookEx(WH_KEYBOARD_LL)`
   - Hook type: Low-level keyboard hook (system-wide)
   - Thread architecture: Chạy trong STA thread riêng với message loop
   - Key processing: Xử lý các phím đặc biệt, modifier keys
   - Real-time transfer: Gửi từng phím qua SignalR method `ReceiveKey`
3. **Key processing logic**:
   - Detection: Caps Lock và Shift state
   - Special keys: Enter, Tab, Backspace, Arrow keys, Function keys

- Modifier keys: Ctrl, Alt, Windows key
- Character conversion: Chuyển đổi key codes thành readable text

4. **Security and privacy**:
   - Cần elevated privileges (Administrator)
   - Clear indication khi keylogger đang chạy
   - Optional: Chỉ ghi log khi có sự đồng ý

3.6 Shutdown / Restart

1. **Mục đích**: Tắt hoặc khởi động lại máy tính từ xa
2. **Đặc điểm kỹ thuật**:
   - Sử dụng Windows shutdown command với parameters:
     - Shutdown: `shutdown /s /t 5 /c "message"`
     - Restart: `shutdown /r /t 5 /c "message"`
   - Delay: 5 giây trước khi thực hiện
   - User notification: Hiển thị message cho người dùng
   - Process execution: Sử dụng `Process.Start()`
3. **Safety features**:
   - User confirmation trước khi thực thi
   - Delay để cho phép hủy bỏ (thông qua `shutdown /a`)
   - Logging đầy đủ cho audit purposes
   - Chỉ thực thi khi có đủ privileges

3.7 SetRegistry

1. **Mục đích**: Thay đổi giá trị Windows Registry từ xa
2. **Đặc điểm kỹ thuật**:
   - Sử dụng .NET Registry API: `Microsoft.Win32.Registry`
   - Supported root keys: HKCU, HKLM, HKCR, HKU, HKCC
   - Auto type detection: String, Integer, Boolean, Hexadecimal
   - Path handling: Tự động tạo subkey nếu không tồn tại
   - Error handling: Xử lý access denied, invalid path
3. **Registry value type detection**:
   - Integer: Parse string thành int
   - Boolean: "true"/"false"→ 1/0
   - Hexadecimal: String bắt đầu với "0x"
   - String: Mặc định

4. **Security considerations**:
    - Limited access: Một số registry keys cần elevated privileges
    - Validation: Kiểm tra path hợp lệ trước khi thực thi
    - Backup: Khuyến nghị backup registry trước khi thay đổi
    - Logging: Ghi lại tất cả thay đổi registry

3.8 StartWebcam / StopWebcam

1. **Mục đích**: Bật/tắt stream video từ webcam của máy tính từ xa
2. **Đặc điểm kỹ thuật**:
    - Sử dụng AForge.Video.DirectShow library
    - Camera discovery: `FilterInfoCollection(FilterCategory.VideoInputDevice)`
    - Resolution selection: Tự động chọn resolution 320×240
    - Frame rate control: Giới hạn 5 FPS (200ms interval)
    - Image processing: Resize, JPEG compression (40% quality)
    - Real-time transfer: Base64 encoding qua `ReceiveWebcam`
3. **Performance optimization**:
    - Low resolution: Giảm bandwidth sử dụng
    - Frame rate limiting: Giảm CPU usage
    - JPEG compression: Giảm kích thước ảnh
    - Memory management: Proper disposal của Bitmap objects
4. **Recording features**:
    - Video recording: Capture và lưu frames
    - Frame storage: Lưu tạm trong memory buffer
    - Video creation: Tạo video từ captured frames
    - Download: Tải video về máy local

3.9 GetSystemInfo

1. **Mục đích**: Thu thập và gửi thông tin hệ thống từ máy tính từ xa
2. **Thông tin thu thập**:
    - Machine name: `Environment.MachineName`
    - User name: `Environment.UserName`
    - OS version: `Environment.OSVersion`
    - Processor count: `Environment.ProcessorCount`
    - Working set: `Environment.WorkingSet` (converted to MB)
    - System directories: Current, System

- Uptime: `Environment.TickCount`
- Current time: `DateTime.Now`

3. **Data format**:
   - JSON object với các properties
   - Human-readable formatting
   - Structured để hiển thị trên dashboard

4 SignalR Communication Architecture

4.1 ControlHub Implementation

1. **Connection Management**:
   - `ConcurrentDictionary<string, AgentInfo>`: Lưu trữ connected agents
   - `OnConnectedAsync()`: Xử lý khi agent kết nối
   - `OnDisconnectedAsync()`: Xử lý khi agent ngắt kết nối
   - Agent info bao gồm: ConnectionId, MachineName, UserName, ConnectedAt
2. **Message Routing**:
   - Server-to-Client: `Clients.All.SendAsync(command, data)`
   - Client-to-Server: Các method `ReceiveX()` (ReceiveScreenshot, ReceiveKey, etc.)
   - Broadcast: Gửi đến tất cả connected clients
   - Specific client: `Clients.Client(connectionId).SendAsync()`

4.2 Real-time Data Transfer

1. **Đặc điểm kỹ thuật**:
   - Image capture: Sử dụng Bitmap và Graphics APIs
   - JPEG compression: Chất lượng 20-40% tùy độ phân giải
   - MemoryStream: Convert Bitmap thành byte array
   - Base64 encoding: `Convert.ToBase64String()`
   - SignalR transfer: String message qua WebSocket
   - Client-side decoding: `[data]`
2. **Performance considerations**:
   - Message size limit: 50MB (cấu hình trong HubOptions)
   - Compression trade-off: Chất lượng vs kích thước
   - Bandwidth optimization: Frame rate limiting, resolution scaling

4.3 Client Connection Management

1. **Implementation details**:
   - SignalR built-in reconnection: `WithAutomaticReconnect()`
   - Exponential backoff: Tăng dần thời gian chờ giữa các lần retry
   - Event handlers: `OnReconnecting`, `OnReconnected`, `OnClosed`
   - Connection state tracking: Visual indicators trên dashboard
2. **Error handling**:
   - Network failures: Tự động retry với backoff strategy
   - Server unavailable: Liên tục thử kết nối lại
   - Connection state: Hiển thị trạng thái real-time
   - User notification: Thông báo khi mất kết nối/kết nối lại

5 Web Dashboard Architecture

5.1 Real-time UI Updates

1. **Event Handling**:
   - Connection events: `onreconnecting`, `onreconnected`, `onclose`
   - Data events: `on("ReceiveScreenshot")`, `on("ReceiveKey")`, etc.
   - Command execution: `connection.invoke(command, data)`
   - Error handling: Try-catch với user-friendly messages
2. **UI Components**:
   - Screen viewer: Hiển thị ảnh screenshot từ base64
   - Webcam viewer: Real-time video stream từ webcam frames
   - Keylogger console: Append keystrokes với timestamp
   - Process table: Dynamic updates khi có process changes
   - System logs: Color-coded log entries

5.2 Data Processing and Storage

1. **Client-side Data Processing**:
   - Image display: `<img src="[data]">`
   - Video recording: Capture webcam frames vào array
   - Keylog storage: Append vào textarea với scroll auto
   - Process data: Render HTML table từ JSON data
   - Statistics: Update counters real-time
2. **File Download Operations**:
   - Screenshot download: Tạo <a> element với base64 href

- Keylog export: Tạo Blob từ text content
- Video save: Tạo video từ captured frames
- Automatic cleanup: Remove temporary elements

6 Bảo mật và xử lý lỗi

6.1 Security Implementation

1. **Current Security Measures**:
   - Input validation: Kiểm tra tất cả user inputs
   - Path sanitization: Ngăn chặn path traversal attacks
   - Process authorization: Kiểm tra trước khi kill process
   - Error message sanitization: Không expose sensitive information
   - Connection validation: Chỉ chấp nhận connections hợp lệ
2. **Security Limitations**:
   - No authentication: Bất kỳ ai cũng có thể kết nối
   - No encryption: Dữ liệu truyền qua HTTP (không HTTPS)
   - CORS AllowAll: Chấp nhận connections từ bất kỳ origin
   - No rate limiting: Không giới hạn request rate
3. **Recommended Security Improvements**:
   - HTTPS implementation: SSL/TLS encryption
   - Authentication system: JWT hoặc Windows Authentication
   - Authorization framework: Role-based access control
   - IP whitelisting: Chỉ cho phép IP addresses được authorized
   - Rate limiting: Ngăn chặn DoS attacks
   - Audit logging: Log tất cả actions cho security auditing

6.2 Error Handling Strategy

1. **Error Categories**:
   - Network errors: Connection failures, timeouts
   - System errors: Access denied, file not found, process not found
   - Resource errors: Out of memory, camera unavailable
   - User errors: Invalid input, insufficient permissions
2. **Error Recovery Mechanisms**:
   - Auto-retry: Network operations với exponential backoff
   - Graceful degradation: Continue operation khi component fails

- Resource cleanup: Proper disposal trong finally blocks
  Connection recovery: Tự động reconnect khi mất kết nối

3. **Logging System**:
   - Structured logging: Timestamp, log level, message
   - Console output: Real-time logging trên server và client
   - Dashboard display: Color-coded log entries
     File logging: Optional log file rotation

6.3 Resource Management

1. **Memory Management**:
   - IDisposable pattern: Tất cả resource-intensive classes implement IDisposable
   - Using statements: Đảm bảo proper disposal
   - Garbage collection: Manual GC.Collect() khi cần thiết
   - Memory limits: Giới hạn buffer sizes để tránh memory leaks
2. **Thread Management**:
   - Thread safety: Sử dụng lock, SemaphoreSlim cho synchronization
   - Thread termination: Proper thread cleanup khi ứng dụng kết thúc
   - Background threads: Sử dụng background threads cho long-running operations
   - Thread pooling: .NET ThreadPool cho các tác vụ ngắn
3. **Network Resource Management**:
   - Connection pooling: SignalR quản lý connection pool
   - Bandwidth throttling: Giới hạn frame rate và image quality
   - Timeout handling: Connection và operation timeouts
   - Buffer management: Proper buffer allocation và cleanup

7 Kết luận

Hệ thống Remote Control này thể hiện một kiến trúc client-server hiện đại với:

- **Real-time Communication**: Sử dụng SignalR cho low-latency, bidirectional communication
- **Modular Design**: Tách biệt các thành phần (WebcamService, KeyLoggerService, etc.)
- **Web-based Interface**: Dashboard hiện đại với responsive design
- **Comprehensive Feature Set**: Đầy đủ các tính năng điều khiển và giám sát từ xa
- **Cross-platform Support**: .NET 10.0 cho cả server và client

7.1 Ưu điểm chính

- **Dễ sử dụng**: Web dashboard với giao diện trực quan
- **Hiệu suất cao**: Real-time updates với tối ưu hóa bandwidth
- **Đáng tin cậy**: Auto-reconnection và comprehensive error handling
- **Dễ mở rộng**: Kiến trúc module cho phép thêm tính năng mới
- **Deployment linh hoạt**: Có thể triển khai trên LAN hoặc Internet

7.2 Hạn chế và hướng phát triển

- **Bảo mật**: Cần thêm authentication, authorization, và encryption
- **Scalability**: Cần distributed architecture cho nhiều agents
- **Tính năng bổ sung**: Remote shell, file transfer, session recording
- **Multi-platform**: Hỗ trợ thêm Linux và macOS
- **Mobile App**: Ứng dụng di động cho remote monitoring

7.3 Ứng dụng thực tế

Hệ thống có thể được sử dụng trong các tình huống:

- IT Help Desk: Hỗ trợ kỹ thuật từ xa
- System Administration: Quản lý nhiều máy tính trong mạng
- Classroom Management: Giám sát phòng máy tính
- Parental Control: Giám sát máy tính gia đình
- Digital Forensics: Thu thập evidence (với proper authorization)