**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN HỌC**

**PBL4 : Dự án hệ điều hành và mạng máy tính**

**ĐỀ TÀI :**

**Xây dựng ứng dụng truy xuất và phân tích dữ liệu hình ảnh từ Camera qua mạng**

|  |  |  |
| --- | --- | --- |
| **Họ tên sinh viên** | **Mã sinh viên** | **Nhóm HP** |
| Nguyễn Đình Hiếu | 102230149 | 23.15A |
| Lê Hữu Tuấn Anh | 102230142 | 23.15A |

**CBHD : PGS.TS. Nguyễn Tấn Khôi**

*Đà nẵng, 12/2025*

**MỤC LỤC**

[**DANH SÁCH TỪ VIẾT TẮT 3**](#_heading=h.fowi008thoyj)

[**BẢNG PHÂN CÔNG CÔNG VIỆC 6**](#_heading=h.ge0cmhyco7zt)

[**MỞ ĐẦU 1**](#_heading=h.962rttnxihnb)

[**1. CƠ SỞ LÝ THUYẾT 2**](#_heading=h.p6vphv9yy0ba)

[1.1. Công nghệ xử lý Multimedia và Thư viện JavaCV 3](#_heading=h.czoywohkktjo)

[1.1.1. FFmpeg và JavaCV 3](#_heading=h.xa0srf6haka1)

[1.1.2. Pixel Format và Chuyển đổi không gian màu 3](#_heading=h.2s6lmbhfr60b)

[1.2. Kiến trúc hệ thống và giao tiếp mạng 3](#_heading=h.d32euwsyljj7)

[1.2.1. Các thành phần kiến trúc 3](#_heading=h.az94i9mxrlef)

[1.2.2. Luồng giao tiếp và Giao thức 4](#_heading=h.dnrhai5952cf)

[1.3. Lập trình đa luồng (Multithreading) và Quản lý tài nguyên 5](#_heading=h.6qq8jkg7f7yz)

[1.3.1. Đồng bộ hóa dữ liệu (Synchronization) 5](#_heading=h.4dy7whwp0trg)

[1.3.2. Mô hình Đa luồng (Multithreading) 5](#_heading=h.z5ke5h76q3xw)

[1.4. Kết chương 5](#_heading=h.j7mj7lakhtdv)

[**2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 5**](#_heading=h.k2pujt1bog66)

[2.1. PHÁT BIỂU BÀI TOÁN 6](#_heading=h.zc11p7dyhbnj)

[2.2. PHÂN TÍCH YÊU CẦU HỆ THỐNG 6](#_heading=h.3asze8ds5n8h)

[2.2.1. Yêu cầu chức năng 6](#_heading=h.4lu7v8fpmqhj)

[2.2.2. Yêu cầu phi chức năng 6](#_heading=h.1z0dzlb7f5zs)

[2.3. PHÂN TÍCH HIỆN TRẠNG 6](#_heading=h.jtloaeyxigtj)

[2.3.1. Các mô hình giám sát hiện có 6](#_heading=h.xshw6n8c5qhl)

[2.3.2. Đánh giá và so sánh 7](#_heading=h.qkph7b4ivytz)

[2.3.3. Vấn đề đặt ra (Problem Statement) 8](#_heading=h.w8oeylcc5l0l)

[2.4. PHÂN TÍCH CHỨC NĂNG 8](#_heading=h.fnomh93eb62f)

[2.4.1. Đối tượng sử dụng 8](#_heading=h.rtm1r65tj9lz)

[2.4.2. Các Chức năng Chính của Hệ thống 9](#_heading=h.u0q6iaz3y7dw)

[2.4.3. Các Thuật toán và Kỹ thuật Cốt lõi 10](#_heading=h.p5mrumaeac8j)

[2.4.4. Công nghệ sử dụng 12](#_heading=h.5u2wzy803lyf)

[2.5. THIẾT KẾ CƠ SỞ DỮ LIỆU 12](#_heading=h.y8k97qjaditr)

[2.5.1. Sơ đồ tổng thể 12](#_heading=h.7s4trvbrpyy)

[2.5.2. Table Users 12](#_heading=h.o1mjwupq30ny)

[2.5.3. Table Clients 13](#_heading=h.ndn15yxvd8kg)

[2.5.4. Table Cameras 14](#_heading=h.ybpldxldp7sp)

[2.5.5. Table Images 14](#_heading=h.m1h8iedn9n7u)

[2.6. XÂY DỰNG CHƯƠNG TRÌNH 16](#_heading=h.3ignh699n37)

[2.6.1. Tổ chức thư mục 16](#_heading=h.xypirg5o8tss)

[2.7. Kết chương 17](#_heading=h.t91i8ff35zty)

[**3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 17**](#_heading=h.i6j8np27uncr)

[3.1. MÔ HÌNH TRIỂN KHAI 18](#_heading=h.3bpoh92dhw1p)

[3.1.1. Môi trường triển khai 18](#_heading=h.9fqraspz50tw)

[3.1.2. Các công cụ sử dụng 18](#_heading=h.qtsm6o9u2frl)

[3.1.3. Cấu hình hệ thống 19](#_heading=h.w33p8yhizjry)

[3.2. KẾT QUẢ THỰC NGHIỆM 01 19](#_heading=h.780oo4jk33cz)

[3.2.1. Kịch bản 1: Chức năng đăng nhập hệ thống 19](#_heading=h.2xv1nk6l2jmy)

[3.2.2. Kịch bản 2: Chức năng: Đăng ký tài khoản (Register Account) 21](#_heading=h.hysuhpeoaevl)

[3.2.3. Kịch bản 3: Chức năng: Quản lý Camera của tôi (Manage My Cameras) 23](#_heading=h.p43ygjd2yf9w)

[3.2.4. Kịch bản 4: Chức năng: Quản lý người dùng (Manage Users - dành cho Admin) 24](#_heading=h.9ga74ch0x6an)

[3.2.5. Kịch bản 5: Chức năng: Xem danh sách ảnh Snapshot (View Snapshot List) 26](#_heading=h.9q2m9a342eaf)

[3.2.6. Kịch bản 6: Chức năng: Upload Snapshot (Upload Snapshot) 28](#_heading=h.9h9ucm4t1gd4)

[3.2.7. Kịch bản 7: Chức năng: Xóa ảnh Snapshot (Delete Snapshot) 28](#_heading=h.qrnbypqhvth3)

[3.2.8. Kịch bản 8: Chức năng:Xem Live Camera (trên Ứng dụng Client Desktop) 30](#_heading=h.3lunevpxt39j)

[3.2.9. Kịch bản 9: Chức năng: Quản lý Client của tôi (Manage My Clients) 31](#_heading=h.wca102p4i50f)

[3.3. NHẬN XÉT ĐÁNH GIÁ KẾT QUẢ 33](#_heading=)

[3.3.1. Những ưu điểm đạt được 33](#_heading=h.armo5hf9r3u1)

[3.3.2. Các hạn chế và tồn tại 33](#_heading=h.kwjj8ly3qtwm)

[3.3.3. Tổng hợp đánh giá hiệu năng 34](#_heading=h.gkdhm0guixiq)

[3.4. Kết chương 34](#_heading=h.hve7fzxioef0)

[**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 35**](#_heading=h.qukzffqomc58)

[1. KẾT QUẢ ĐẠT ĐƯỢC 36](#_heading=h.g9qj6ocuuw51)

[2. HẠN CHẾ VÀ TỒN TẠI 37](#_heading=h.26eymgky3ovz)

[3. KIẾN NGHỊ VÀ HƯỚNG PHÁT TRIỂN 37](#_heading=h.83qxmny3rks2)

**DANH SÁCH HÌNH ẢNH**

[*Hình 2.1 Biểu đồ chức năng*](#bookmark=id.dkcbtz20qlpn)

[*Hình 2.2: Sơ đồ tổng thể*](#bookmark=id.f62wh36muvjp)

[*Hình 3.1: Login thành công được kiểm thử trên postman*](#bookmark=id.kete5tn09enz)

[*Hình 3.2: Login thất bại do sai mật khẩu*](#bookmark=id.pf44d46birxf)

[*Hình 3.3: Bổ sung thông tin tạo tài khoản*](#bookmark=id.n4af3lb1rfo1)

[*Hình 3.4: Thông báo đăng ký thành công*](#bookmark=id.wus3xzhm4680)

[*Hình 3.5: Thêm camera*](#bookmark=id.8oax04yr0jv4)

[*Hình 3.6: Thêm camera thất bại*](#bookmark=id.ovu3i0o1vtw7)

[*Hình 3.7: Xác thực email*](#bookmark=id.lla21rb2zs2d)

[*Hình 3.8: Xem chi tiết người dùng*](#bookmark=id.qy0wof3338eo)

[*Hình 3.9: Chỉnh sửa thông tin người dùng*](#bookmark=id.sdeyekjp3i7c)

[*Hình 3.10: Thêm tài khoản mới (lựa chọn quyền cho tài khoản đó)*](#bookmark=id.h0rkx9vyrhdj)

[*Hình 3.11: Xem danh sách ảnh từ toàn bộ camera*](#bookmark=id.1swp6tmfjyeq)

[*Hình 3.12: Xem danh sách ảnh được lọc theo camera*](#bookmark=id.5xbur8a8miba)

[*Hình 3.13: Xóa một ảnh*](#bookmark=id.tljwaz99xdm)

[*Hình 3.14: Xóa nhiều ảnh*](#bookmark=id.gkpuztj34fhu)

[*Hình 3.15: Danh sách camera đang kết nối trực tiếp*](#bookmark=id.4hwj468jydqi)

[*Hình 3.16: Danh sách client của người dùng hiện tại*](#bookmark=id.50s3ctic8rur)

[*Hình 3.17: Danh sách camera của client*](#bookmark=id.656bjflr102u)

[*Hình 3.18: Xem cấu hình hiện tại của client và chỉnh sửa trực tiếp*](#bookmark=id.plkzj3ia9gfz)

# DANH SÁCH TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Giải thích (Tiếng Anh)** | **Diễn giải tiếng Việt** |
| **API** | Application Programming Interface | Giao diện lập trình ứng dụng |
| **CRUD** | Create, Read, Update, Delete | Bốn thao tác cơ bản của cơ sở dữ liệu (Tạo, Đọc, Sửa, Xóa) |
| **DTO** | Data Transfer Object | Đối tượng chuyển giao dữ liệu (Dùng để trao đổi giữa các lớp) |
| **HTTP** | Hypertext Transfer Protocol | Giao thức truyền tải siêu văn bản |
| **IP** | Internet Protocol | Giao thức liên mạng (Địa chỉ mạng) |
| **JDBC** | Java Database Connectivity | API kết nối cơ sở dữ liệu của Java |
| **JPA** | Java Persistence API | Chuẩn đặc tả quản lý dữ liệu quan hệ trong Java |
| **JSON** | JavaScript Object Notation | Định dạng hoán đổi dữ liệu dạng văn bản nhẹ (Ký hiệu đối tượng JavaScript) |
| **JWT** | JSON Web Token | Mã thông báo xác thực dạng JSON (Dùng để đăng nhập an toàn) |
| **LAN** | Local Area Network | Mạng cục bộ |
| **MD5** | Message-Digest Algorithm 5 | Giải thuật tóm tắt thông điệp 5 (Thuật toán băm) |
| **MJPEG** | Motion JPEG | Định dạng nén video (Chuỗi các ảnh JPEG liên tiếp) |
| **MVC** | Model-View-Controller | Mô hình kiến trúc phần mềm (Mô hình - Giao diện - Điều khiển) |
| **ONVIF** | Open Network Video Interface Forum | Chuẩn giao tiếp mở cho các thiết bị an ninh mạng (Camera IP) |
| **REST** | Representational State Transfer | Kiến trúc chuyển giao trạng thái đại diện (Dùng thiết kế API) |
| **RTSP** | Real Time Streaming Protocol | Giao thức truyền phát tin thời gian thực (Dùng lấy luồng video) |
| **SHA** | Secure Hash Algorithm | Giải thuật băm an toàn |
| **SQL** | Structured Query Language | Ngôn ngữ truy vấn mang tính cấu trúc (Dùng cho CSDL quan hệ) |
| **TCP** | Transmission Control Protocol | Giao thức điều khiển truyền vận (Đảm bảo tin cậy) |
| **URL** | Uniform Resource Locator | Định vị tài nguyên đồng nhất (Đường dẫn liên kết) |

# BẢNG PHÂN CÔNG CÔNG VIỆC

|  |  |  |
| --- | --- | --- |
| TT | Công việc | Người thực hiện |
| 1 | Thiết lập môi trường phát triển, GitHub Repository | Nguyễn Đình Hiếu |
| 2 | Thiết lập cơ sở dữ liệu (CSDL) ban đầu | Lê Hữu Tuấn Anh |
| 3 | Phân tích và thiết kế hệ thống (Sơ đồ ERD, thiết kế API, kiến trúc Client-Server) | Cả nhóm |
| 4 | [Backend] Xây dựng khung dự án Spring Boot, cấu hình kết nối Database và các Entity | Nguyễn Đình Hiếu |
| 5 | [Backend] Phát triển Module xác thực (Đăng ký, Đăng nhập, JWT Security) | Cả nhóm |
| 6 | [Backend] Phát triển các API quản lý (CRUD Camera, Client, User, Image) và API Upload ảnh | Cả nhóm |
| 7 | [Backend] Cấu hình và xử lý WebSocket Server để giao tiếp thời gian thực | Nguyễn Đình Hiếu |
| 8 | [Desktop App] Thiết kế giao diện (UI) JavaFX: Đăng nhập, Dashboard, Cấu hình | Lê Hữu Tuấn Anh |
| 9 | [Desktop App] Nghiên cứu và tích hợp thư viện JavaCV/FFmpeg xử lý luồng RTSP | Lê Hữu Tuấn Anh |
| 10 | [Desktop App] Lập trình logic: Kết nối Camera, tự động chụp ảnh và gửi về Server | Lê Hữu Tuấn Anh |
| 11 | [Web App] Thiết kế giao diện Web (HTML/CSS/JS): Dashboard, Thư viện ảnh, Admin | Nguyễn Đình Hiếu |
| 12 | [Web App] Lập trình logic Frontend: Gọi API lấy dữ liệu và nhận thông báo WebSocket | Nguyễn Đình Hiếu |
| 13 | Thực hiện kiểm thử tích hợp toàn hệ thống (Integration Testing) | Cả nhóm |
| 14 | Viết báo cáo tổng hợp (Chương 1, 2, 3) và soạn thảo Slide báo cáo | Cả nhóm |

# MỞ ĐẦU

**1.**  **Tổng quan về đề tài**

**1.1. Lý do chọn đề tài**

Trong kỷ nguyên Công nghiệp 4.0 và sự bùng nổ của Internet vạn vật (IoT), nhu cầu giám sát an ninh và thu thập dữ liệu hình ảnh trực quan đang trở nên thiết yếu trong mọi lĩnh vực, từ giao thông thông minh, nhà máy sản xuất đến an ninh hộ gia đình. Tuy nhiên, thực tế cho thấy các hệ thống camera giám sát truyền thống (CCTV/NVR) thường hoạt động theo mô hình khép kín, phụ thuộc nhiều vào phần cứng chuyên dụng đắt tiền (đầu ghi hình), khó mở rộng và hạn chế khả năng tích hợp với các hệ thống phân tích dữ liệu bên thứ ba.

Xuất phát từ thực tế đó, nhóm quyết định thực hiện đề tài "Xây dựng ứng dụng truy xuất và phân tích dữ liệu hình ảnh từ Camera qua mạng". Dự án hướng tới việc xây dựng một giải pháp phần mềm phân tán theo mô hình Client-Server, sử dụng các chuẩn giao tiếp mở (như RTSP, ONVIF) để kết nối, thu thập dữ liệu hình ảnh tập trung và cung cấp khả năng quản lý linh hoạt qua giao diện Web. Đây là giải pháp nhằm giảm thiểu chi phí phần cứng và tạo tiền đề cho các ứng dụng xử lý ảnh thông minh.

**1.2. Tính cấp thiết của đề tài**

**1.2.1. Thực tiễn học tập và nghiên cứu**

Đề tài là môi trường lý tưởng để sinh viên vận dụng tổng hợp các kiến thức cốt lõi như: Lập trình mạng (Network Programming), Hệ thống phân tán, Lập trình đa luồng (Multithreading) và Xử lý đa phương tiện (Multimedia Processing). Việc xây dựng hệ thống từ khâu thu thập tín hiệu video (Desktop Client), truyền tải qua mạng đến lưu trữ và hiển thị (Web/Server) giúp sinh viên nắm vững quy trình phát triển phần mềm phức tạp.

**1.2.2. Ứng dụng thực tế cao**

Hệ thống giải quyết bài toán quản lý tập trung nhiều camera từ các vị trí địa lý khác nhau mà không cần đầu tư hạ tầng VPN phức tạp hay đầu ghi hình cố định. Việc tách biệt module thu thập (Client) và module quản lý (Server) giúp hệ thống hoạt động linh hoạt, dễ dàng triển khai trên các máy tính phổ thông sẵn có.

**1.2.3. Khả năng mở rộng và phát triển**

Hệ thống được thiết kế với kiến trúc mở, là cơ sở hạ tầng quan trọng để tích hợp các module trí tuệ nhân tạo (AI) trong tương lai như: Nhận diện khuôn mặt, phát hiện xâm nhập trái phép, đếm lưu lượng người/xe hoặc cảnh báo cháy nổ dựa trên hình ảnh.

**1.3. Các vấn đề quan trọng cần giải quyết**

Để xây dựng thành công hệ thống, nhóm thực hiện cần tập trung giải quyết các bài toán kỹ thuật trọng tâm sau:

**1.3.1. Xử lý luồng video và đa luồng (Multimedia & Multithreading)**

Hệ thống yêu cầu khả năng kết nối ổn định tới Camera IP thông qua giao thức RTSP và trích xuất hình ảnh liên tục. Thách thức đặt ra là phải xử lý các tác vụ nặng này (giải mã video, chụp ảnh) mà không làm "treo" giao diện ứng dụng. Giải pháp là sử dụng thư viện JavaCV/FFmpeg kết hợp với kỹ thuật lập trình đa luồng để tối ưu hiệu năng.

**1.3.2. Truyền tải dữ liệu thời gian thực (Real-time Communication)**

Việc đồng bộ trạng thái thiết bị và hình ảnh từ Client về Server cần diễn ra tức thì. Vấn đề là làm sao để Server nhận biết ngay lập tức khi Camera mất kết nối hoặc có hình ảnh mới. Nhóm giải quyết bằng cách kết hợp giao thức HTTP (cho tải file) và WebSocket (cho tín hiệu điều khiển thời gian thực).

**1.3.3. Quản lý và lưu trữ dữ liệu tập trung**

Với số lượng camera lớn, khối lượng hình ảnh thu thập được là khổng lồ. Hệ thống cần cơ chế tổ chức lưu trữ khoa học trên Server (File System), đồng thời cung cấp API cho phép truy xuất, lọc và tìm kiếm lịch sử hình ảnh nhanh chóng theo thời gian thực.

**1.3.4. Bảo mật và xác thực (Authentication & Security)**

Đảm bảo chỉ người dùng được cấp quyền mới có thể xem hình ảnh và điều khiển thiết bị. Nhóm áp dụng chuẩn JSON Web Token (JWT) để quản lý phiên đăng nhập và bảo vệ các API quan trọng trước các truy cập trái phép.

**1.3.5. Đồng bộ cấu hình từ xa**

Hệ thống cần cho phép người quản trị thay đổi cấu hình (như chu kỳ chụp ảnh, trạng thái hoạt động) từ Web Server và đồng bộ xuống Desktop Client ngay lập tức mà không cần thao tác trực tiếp trên máy trạm.

**2. Mục đích và ý nghĩa của đề tài**

**2.1. Mục đích**

Hệ thống được xây dựng với mục tiêu cốt lõi là giải quyết bài toán phân mảnh trong giám sát an ninh, cụ thể hướng tới các mục đích sau:

Xây dựng nền tảng thu thập dữ liệu tập trung: Phát triển một cơ chế ổn định để kết nối và đồng bộ hóa tín hiệu hình ảnh từ đa dạng các chủng loại Camera IP (hỗ trợ chuẩn ONVIF/RTSP) về một máy chủ trung tâm. Hệ thống đóng vai trò như một lớp trung gian (Middleware), loại bỏ sự phụ thuộc vào các đầu ghi hình phần cứng (NVR/DVR) đắt tiền và khó nâng cấp.

Chuẩn hóa quy trình quản lý thiết bị: Cung cấp bộ công cụ toàn diện cho phép người quản trị thực hiện các thao tác từ xa như: thêm mới/xóa thiết bị, cấu hình chu kỳ trích xuất hình ảnh, và giám sát trạng thái hoạt động (Health check) của từng Camera trong thời gian thực.

Tối ưu hóa trải nghiệm truy xuất dữ liệu: Xây dựng giao diện Web Dashboard trực quan, thân thiện, cho phép người dùng dễ dàng tìm kiếm, lọc và xem lại lịch sử giám sát dựa trên các mốc thời gian cụ thể hoặc theo sự kiện. Đồng thời, hệ thống tích hợp cơ chế cảnh báo tức thì qua giao thức thời gian thực, giúp người dùng nắm bắt nhanh chóng các sự cố mất kết nối hoặc hành vi bất thường.

**2.2. Ý nghĩa**

Về mặt thực tiễn ứng dụng:

Giải pháp kinh tế và linh hoạt: Đề tài mang lại một giải pháp giám sát an ninh với chi phí triển khai thấp, tận dụng được hạ tầng máy tính và mạng có sẵn thay vì phải đầu tư hệ thống chuyên dụng khép kín. Khả năng hoạt động đa nền tảng (Desktop/Web) giúp việc giám sát trở nên linh động, không bị giới hạn bởi không gian địa lý.

Nền tảng cho chuyển đổi số trong an ninh: Hệ thống chuyển đổi dữ liệu video thô thành các tài nguyên số được lưu trữ có cấu trúc. Đây là tiền đề quan trọng để tích hợp các bài toán Trí tuệ nhân tạo (AI) và Học máy (Machine Learning) trong tương lai, như nhận diện khuôn mặt, phân tích hành vi đám đông, hay phát hiện xâm nhập trái phép, nâng tầm hệ thống từ "giám sát bị động" sang "cảnh báo chủ động".

Về mặt đào tạo và nghiên cứu khoa học:

Làm chủ công nghệ lõi: Quá trình thực hiện đề tài là cơ hội để sinh viên nghiên cứu sâu về các kỹ thuật xử lý đa phương tiện (Multimedia Processing) phức tạp, như giải mã luồng video bằng thư viện JavaCV/FFmpeg và xử lý hình ảnh chất lượng cao.

Vận dụng kiến trúc phần mềm hiện đại: Đề tài yêu cầu sinh viên thiết kế và triển khai hệ thống theo mô hình Client-Server phân tán, áp dụng kiến trúc Clean Architecture để đảm bảo tính dễ bảo trì và mở rộng. Đồng thời, sinh viên được rèn luyện kỹ năng giải quyết các bài toán khó về hiệu năng như: xử lý đa luồng (Multithreading) để tránh tắc nghẽn khi quản lý nhiều camera, tối ưu hóa băng thông mạng khi truyền tải dữ liệu lớn (Big Data), và triển khai cơ chế giao tiếp hai chiều thời gian thực (WebSocket).

Nâng cao tư duy bảo mật hệ thống: Thông qua việc triển khai cơ chế xác thực và phân quyền không trạng thái (Stateless Authentication) bằng JSON Web Token (JWT), sinh viên hiểu rõ hơn về các nguy cơ an ninh mạng và cách thức bảo vệ dữ liệu nhạy cảm trong các ứng dụng thực tế.

**3. Phương pháp thực hiện**

Quá trình thực hiện đề tài được triển khai tuần tự qua bốn giai đoạn chính: nghiên cứu tài liệu và phân tích yêu cầu, phân tích và thiết kế hệ thống, cài đặt và triển khai, cuối cùng là kiểm thử và đánh giá kết quả.

**3.1. Nghiên cứu tài liệu và phân tích yêu cầu**

Ở giai đoạn đầu, nhóm tiến hành tìm hiểu các kiến thức nền tảng liên quan đến xử lý đa phương tiện (Multimedia), giao thức truyền tải video thời gian thực (RTSP) và chuẩn giao tiếp thiết bị an ninh (ONVIF). Nhóm tập trung nghiên cứu thư viện JavaCV/FFmpeg để giải quyết bài toán trích xuất hình ảnh từ luồng video và kỹ thuật lập trình đa luồng (Multithreading) để tối ưu hiệu năng xử lý trên máy trạm.

Bên cạnh đó, nhóm cũng tìm hiểu kiến trúc Spring Boot để xây dựng Backend Server mạnh mẽ, kết hợp với cơ chế xác thực bảo mật bằng JSON Web Token (JWT). Các giao thức truyền thông mạng như HTTP (cho truyền tải file) và WebSocket (cho tín hiệu điều khiển tức thời) cũng được khảo sát kỹ lưỡng để lựa chọn giải pháp đồng bộ dữ liệu tối ưu.

**3.2. Phân tích và thiết kế hệ thống**

Sau khi làm rõ yêu cầu, nhóm tiến hành thiết kế kiến trúc tổng thể của hệ thống theo mô hình phân tán Client-Server. Trong đó, lớp thu thập dữ liệu (Desktop Client) được tách biệt hoàn toàn với lớp quản lý trung tâm (Server), đảm bảo tính linh hoạt và khả năng mở rộng.

Mô hình giao tiếp được thiết kế dựa trên sự kết hợp giữa RESTful API cho các tác vụ quản lý (CRUD) và WebSocket cho các tác vụ thời gian thực như cảnh báo hoặc điều khiển thiết bị.

Cơ sở dữ liệu quan hệ MySQL được thiết kế để quản lý thông tin người dùng, danh sách thiết bị camera, nhật ký hoạt động và metadata của hình ảnh. Từ đó, các module chức năng được đề xuất, bao gồm: module thu thập và xử lý ảnh tại biên (Edge Processing), module quản lý lưu trữ tập trung, module Dashboard giám sát trên Web và module bảo mật phân quyền.

**3.3. Cài đặt và triển khai**

Trên cơ sở thiết kế, hệ thống Backend được hiện thực bằng ngôn ngữ Java với framework Spring Boot, cung cấp các API chuẩn RESTful và dịch vụ WebSocket Server. Hệ thống lưu trữ hình ảnh được tổ chức khoa học trên File System của máy chủ, được lập chỉ mục qua cơ sở dữ liệu để truy xuất nhanh chóng.

Phần ứng dụng Client (Desktop) được xây dựng bằng JavaFX, tích hợp core xử lý ảnh JavaCV, cho phép kết nối đa luồng tới nhiều Camera IP đồng thời. Giao diện Web Dashboard được phát triển bằng HTML/JS (hoặc Thymeleaf), giúp người quản trị dễ dàng theo dõi hệ thống từ xa.

Trong quá trình hiện thực, các module (như module kết nối RTSP, module upload ảnh) được phát triển và kiểm thử đơn vị (Unit Test) riêng lẻ trước khi tích hợp vào hệ thống hoàn chỉnh.

**3.4. Kiểm thử và đánh giá**

Giai đoạn cuối cùng tập trung vào việc kiểm thử tích hợp toàn bộ hệ thống trong môi trường mạng thực tế (LAN/WiFi). Nhóm thực hiện các kịch bản kiểm thử chức năng (thêm/xóa camera, xem lịch sử), kiểm thử độ trễ (Latency) của tín hiệu cảnh báo và kiểm thử độ ổn định của luồng video khi hoạt động liên tục trong thời gian dài.

Nhóm cũng tiến hành đánh giá mức độ tiêu thụ tài nguyên (CPU/RAM) của ứng dụng Client khi xử lý nhiều luồng camera cùng lúc. Dựa trên kết quả thực nghiệm, hệ thống được tinh chỉnh các tham số cấu hình, rút ra các ưu điểm, hạn chế và đề xuất lộ trình nâng cấp tính năng thông minh (AI) trong tương lai.

**4. Bố cục của báo cáo**

Báo cáo được trình bày trong 3 chương chính:

Chương 1: Cơ sở lý thuyết: Trình bày tổng quan về công nghệ xử lý Multimedia (FFmpeg/JavaCV), kiến trúc hệ thống Client-Server và các kỹ thuật lập trình đa luồng, đồng bộ hóa dữ liệu.

Chương 2: Phân tích và Thiết kế hệ thống: Phân tích yêu cầu, hiện trạng bài toán, thiết kế kiến trúc tổng thể, cơ sở dữ liệu và chi tiết các chức năng của ứng dụng.

Chương 3: Triển khai và Đánh giá kết quả: Mô tả môi trường cài đặt, các công cụ phát triển, trình bày các kịch bản kiểm thử thực tế (Demo) và đánh giá hiệu năng, ưu nhược điểm của hệ thống.

Kết luận và Hướng phát triển: Tóm tắt kết quả đạt được và đề xuất hướng mở rộng.

# CƠ SỞ LÝ THUYẾT

## 1.1. MÔ HÌNH CLIENT – SERVER

### 1.1.1. Khái niệm và nguyên lý hoạt động.

Mô hình Client–Server (Khách – Chủ) là kiến trúc nền tảng của hầu hết các hệ thống phân tán hiện đại, trong đó các tác vụ và khối lượng công việc được phân chia giữa hai thực thể chính là bên cung cấp dịch vụ (Server) và bên yêu cầu dịch vụ (Client). Server là các máy tính có cấu hình mạnh, chạy các phần mềm chuyên dụng để quản lý tài nguyên, lưu trữ dữ liệu và xử lý các nghiệp vụ logic phức tạp. Ngược lại, Client là các thiết bị hoặc chương trình ứng dụng chạy trên máy trạm của người dùng cuối, có nhiệm vụ gửi các yêu cầu kết nối đến Server và hiển thị kết quả trả về thông qua giao diện người dùng.

Trong mô hình này, giao tiếp giữa hai bên tuân theo quy trình yêu cầu - phản hồi (Request-Response). Client khởi tạo phiên làm việc bằng cách gửi thông điệp yêu cầu đến một địa chỉ IP và cổng xác định của Server. Server luôn ở trạng thái lắng nghe, khi nhận được yêu cầu sẽ tiến hành xác thực, xử lý và gửi lại kết quả. Sự tách biệt rõ ràng này giúp hệ thống dễ dàng bảo trì, nâng cấp và bảo mật dữ liệu tập trung.

### Cấu trúc mô hình trong hệ thông Camera

Trong ngữ cảnh cụ thể của hệ thống quản lý camera giám sát, mô hình Client-Server được triển khai với vai trò đặc thù. Client ở đây không chỉ đơn thuần là trình duyệt web mà còn bao gồm ứng dụng Desktop (JavaFX) đóng vai trò là một thiết bị biên (Edge Device). Ứng dụng Desktop này chịu trách nhiệm kết nối trực tiếp với phần cứng Camera IP thông qua giao thức RTSP để lấy luồng dữ liệu thô. Tại đây, các tác vụ nặng như giải mã video (decoding) và trích xuất hình ảnh (snapshot) được thực hiện cục bộ trước khi gửi dữ liệu đã qua xử lý về Server.

Phía Server đóng vai trò là trung tâm điều phối và lưu trữ. Server tiếp nhận các hình ảnh và trạng thái thiết bị từ các Client gửi về, tổ chức lưu trữ chúng vào hệ thống tệp hoặc cơ sở dữ liệu có cấu trúc. Đồng thời, Server cung cấp các dịch vụ API để phục vụ cho các ứng dụng Web Dashboard, cho phép người quản trị giám sát và điều khiển toàn bộ hệ thống từ xa mà không cần tương tác trực tiếp với từng máy trạm.

### Ưu điểm và nhược điểm

Việc áp dụng mô hình Client-Server mang lại lợi thế lớn về khả năng tập trung hóa dữ liệu. Mọi thông tin về hình ảnh, nhật ký hoạt động và cấu hình người dùng đều được quản lý tại Server, giúp việc sao lưu, bảo mật và đồng bộ hóa trở nên dễ dàng và nhất quán. Bên cạnh đó, kiến trúc này giúp giảm tải đáng kể cho các thiết bị đầu cuối, khi các tác vụ lưu trữ lâu dài và phân tích dữ liệu lịch sử được chuyển về máy chủ trung tâm. Hệ thống cũng thể hiện khả năng mở rộng tốt, cho phép bổ sung thêm số lượng lớn Camera hoặc Client giám sát mà không làm gián đoạn cấu trúc mạng hiện có.

Tuy nhiên, mô hình này cũng tồn tại một số hạn chế cố hữu. Đầu tiên là vấn đề phụ thuộc vào băng thông mạng. Do luồng dữ liệu video và hình ảnh luôn chiếm dung lượng lớn, nếu đường truyền không ổn định sẽ dẫn đến hiện tượng trễ hình (latency) hoặc mất kết nối cục bộ. Thứ hai là rủi ro về "điểm chết duy nhất" (Single Point of Failure); nếu Server trung tâm gặp sự cố phần cứng hoặc phần mềm, toàn bộ chức năng lưu trữ và khả năng truy cập từ xa của hệ thống sẽ bị ngưng trệ hoàn toàn cho đến khi máy chủ được khôi phục.

## CÔNG NGHỆ XỬ LÝ ĐA PHƯƠNG TIỆN (MULTIMEDIA)

### 1.2.1. Giao thức RTSP (Real Time Streaming Protocol)

Giao thức RTSP (RFC 2326) là một chuẩn giao thức điều khiển lớp ứng dụng, được thiết kế chuyên biệt để thiết lập và kiểm soát các phiên truyền thông đa phương tiện như video và âm thanh. Khác với HTTP truyền tải toàn bộ tệp tin, RTSP hoạt động như một "chiếc điều khiển từ xa" cho các luồng dữ liệu thời gian thực. Nó không trực tiếp truyền tải dữ liệu video mà phối hợp với giao thức RTP (Real-time Transport Protocol) để vận chuyển các gói tin đa phương tiện.

Trong hệ thống này, RTSP đóng vai trò cốt lõi trong việc kết nối với các Camera IP chuẩn ONVIF. Quá trình giao tiếp bắt đầu bằng các lệnh như OPTIONS để kiểm tra tính năng, DESCRIBE để lấy thông tin về luồng video, và SETUP để thiết lập cơ chế truyền tải. Sau đó, lệnh PLAY được gửi đi để yêu cầu Camera bắt đầu gửi luồng dữ liệu về Client, và lệnh TEARDOWN dùng để kết thúc phiên làm việc, giải phóng tài nguyên mạng.

### Thư viện JavaCV và FFmpeg

Để xử lý được luồng tín hiệu RTSP phức tạp trong môi trường lập trình Java, hệ thống sử dụng JavaCV – một thư viện cầu nối (wrapper) giúp các ứng dụng Java có thể gọi trực tiếp các hàm API mạnh mẽ của FFmpeg. FFmpeg là bộ thư viện mã nguồn mở hàng đầu thế giới về xử lý multimedia, có khả năng giải mã (decode), mã hóa (encode) và chuyển đổi hầu hết các định dạng video/audio hiện hành.

Quy trình xử lý ảnh trong hệ thống diễn ra qua nhiều bước. Đầu tiên, JavaCV sử dụng FFmpeg để khởi tạo kết nối và "bắt" (grab) các gói tin video từ luồng RTSP. Các gói tin nén (thường là chuẩn H.264 hoặc H.265) sau đó được đưa qua bộ giải mã để chuyển thành các khung hình (Frame) thô chứa dữ liệu điểm ảnh (Raw Pixel). Cuối cùng, các khung hình này được chuyển đổi thành đối tượng hình ảnh tiêu chuẩn của Java (BufferedImage) để hiển thị lên giao diện người dùng hoặc nén lại dưới định dạng JPEG/PNG để gửi về Server lưu trữ.

## GIAO TIẾP THỜI GIAN THỰC (WEBSOCKET)

### 1.3.1. Khái niệm

WebSocket là một giao thức truyền thông tiên tiến cung cấp kênh giao tiếp song công toàn phần (Full-duplex) qua một kết nối TCP duy nhất. Điểm khác biệt cơ bản so với HTTP là WebSocket cho phép duy trì kết nối liên tục sau quá trình bắt tay ban đầu. Trong khi HTTP hoạt động theo cơ chế thụ động (Client phải gửi yêu cầu thì Server mới trả lời), WebSocket cho phép Server chủ động đẩy dữ liệu xuống Client ngay khi có sự kiện mới mà không cần chờ yêu cầu từ phía Client.

* + 1. **Ứng dụng trong hệ thống**

Hệ thống sử dụng WebSocket để giải quyết triệt để bài toán độ trễ trong giám sát an ninh. Khi xảy ra các sự kiện quan trọng như Camera mất kết nối đột ngột hoặc phát hiện chuyển động bất thường, Server ngay lập tức đẩy thông báo cảnh báo xuống giao diện Web Dashboard của người quản trị. Điều này đảm bảo tính tức thời của thông tin, giúp người dùng phản ứng kịp thời trước các sự cố. Ngoài ra, WebSocket còn được dùng để đồng bộ trạng thái thiết bị; khi một người dùng thay đổi cấu hình Camera, các Client khác đang cùng theo dõi cũng sẽ thấy trạng thái cập nhật ngay lập tức (Real-time) mà không cần tải lại trang.

## RESTFUL API VÀ KIẾN TRÚC HỆ THỐNG

### RESTful API

REST (Representational State Transfer) là kiểu kiến trúc phần mềm quy định các nguyên tắc thiết kế Web Service, coi mọi dữ liệu trong hệ thống (như Camera, Người dùng, Hình ảnh) là các Tài nguyên (Resource) được định danh bởi URI. Hệ thống sử dụng RESTful API để chuẩn hóa việc giao tiếp giữa Client và Server thông qua các phương thức HTTP chuẩn. Cụ thể, phương thức GET dùng để truy xuất thông tin, POST dùng để tạo mới dữ liệu, PUT để cập nhật và DELETE để xóa bỏ tài nguyên. Kiến trúc này giúp tách biệt hoàn toàn giữa Frontend và Backend, tạo điều kiện thuận lợi cho việc phát triển và mở rộng hệ thống độc lập.

### Cơ chế xác thực JWT (JSON Web Token)

Để đảm bảo an toàn bảo mật cho các API, hệ thống áp dụng cơ chế xác thực không trạng thái (Stateless Authentication) bằng chuẩn JSON Web Token (JWT). Một token JWT bao gồm ba phần chính: Header chứa thông tin thuật toán mã hóa, Payload chứa thông tin định danh người dùng và quyền hạn, và Signature là chữ ký số để đảm bảo tính toàn vẹn dữ liệu.

Quy trình xác thực diễn ra như sau: Khi người dùng đăng nhập thành công với tên tài khoản và mật khẩu đúng, Server sẽ tạo và trả về một chuỗi Token. Client có nhiệm vụ lưu trữ chuỗi này và đính kèm nó vào phần Header của mọi yêu cầu gửi lên Server sau đó. Server sẽ giải mã và kiểm tra chữ ký của Token để xác minh danh tính người dùng. Cơ chế này loại bỏ nhu cầu lưu trữ phiên làm việc (Session) trên bộ nhớ Server, giúp hệ thống hoạt động nhẹ nhàng và dễ dàng mở rộng theo chiều ngang (Scaling) khi lượng người dùng tăng cao.

### Các thành phần kiến trúc

Hệ thống được xây dựng theo mô hình Client-Server tập trung, bao gồm 3 thành phần chính tương tác với nhau: AppClient (Application Client), WebClient (Web Application) và Server (Backend API)

Camera Client (Desktop App): Một ứng dụng desktop tương tác với hệ thống camera, có thể để quản lý hoặc xem các luồng video, kết nối mạng LAN với camera thông qua giao thức RTSP để lấy luồng hình ảnh, kết nối internet với server thông qua giao thức http để gửi hình ảnh và dùng api RESTful cho các thao tác đăng nhập, quản lý camera…. Kết hợp giao tiếp WebSocket để cập nhật theo thời gian thực các thay đổi.  
 Web Frontend: Một giao diện người dùng web để người dùng có thể tương tác với hệ thống thông qua trình duyệt, kết nối với server thông qua giao thức http và các lệnh gọi api RESTful kết hợp giao tiếp WebSocket để cập nhật các ảnh mới lên theo thời gian thực.

Backend Server (Spring Boot): Đây là trung tâm của hệ thống, xử lý logic nghiệp vụ, lưu trữ dữ liệu, quản lý người dùng, camera, client, và có thể xử lý cả luồng video/hình ảnh. Nó cũng cung cấp API cho cả ứng dụng desktop client và web frontend,Chịu trách nhiệm xác thực (Authentication), lưu trữ dữ liệu vào Database và quản lý logic nghiệp vụ..

### Luồng giao tiếp và Giao thức

Hệ thống tập trung sử dụng các giao thức ở tầng ứng dụng để đảm bảo khả năng tương tác linh hoạt giữa các thành phần. Chi tiết các giao thức được mô tả trong Bảng

|  |  |  |
| --- | --- | --- |
| **Giao thức** | **Tầng (OSI)** | **Vai trò trong dự án** |
| HTTP | Ứng dụng (7) | - Giao tiếp chính cho các API RESTful (đăng nhập, quản lý client/camera/người dùng, upload ảnh snapshot).  - Phục vụ các tệp tĩnh (HTML, CSS, JS) cho web frontend.  - Gửi các frame live feed từ desktop client lên server. |
| WebSocket | Ứng dụng (7) | - Giao tiếp thời gian thực hai chiều giữa server và các client (desktop và web).  - Gửi thông báo cập nhật trạng thái client, snapshot mới. |
| TCP | Giao vận (4) | - Cung cấp kết nối đáng tin cậy, có thứ tự và kiểm soát luồng cho HTTP và WebSocket.  - Đảm bảo dữ liệu được gửi và nhận một cách toàn vẹn, không mất mát và đúng thứ tự. |
| IP | Mạng (3) | - Định tuyến các gói dữ liệu giữa các thiết bị trong mạng (client, server, database).  - Cho phép các thành phần giao tiếp với nhau qua địa chỉ IP. |
| TLS/SSL | Trình bày (6) / Phiên (5) (thường được coi là mở rộng của tầng Ứng dụng) | - Bảo vệ dữ liệu nhạy cảm (thông tin đăng nhập, JWT) khỏi bị nghe trộm và giả mạo. (Mặc dù dự án của bạn hiện sử dụng HTTP,production). |
| JDBC | Ứng dụng (7) | - API chuẩn của Java để kết nối và tương tác với cơ sở dữ liệu.  - Cho phép server thực hiện các thao tác CRUD với database (lưu trữ người dùng, client, camera, ảnh). |
| RTSP | Ứng dụng (7) | Appclient sử dụng để kết nối và nhận luồng video/hình ảnh gốc từ Camera IP. |

## Lập trình đa luồng (Multithreading) và Quản lý tài nguyên

### Đồng bộ hóa dữ liệu (Synchronization)

AtomicBoolean: Để quản lý trạng thái hoạt động của luồng (đang chạy hay đã dừng) một cách an toàn giữa các luồng khác nhau, đồ án sử dụng biến nguyên tử AtomicBoolean running. Điều này đảm bảo tính nhất quán của dữ liệu khi người dùng bấm nút "Stop" từ giao diện trong khi luồng đọc video vẫn đang chạy ngầm.

Volatile Keyword: Các biến cấu hình như imageWidth, imageHeight được khai báo volatile để đảm bảo khi người dùng thay đổi cấu hình từ UI, luồng xử lý video sẽ nhận được giá trị mới nhất ngay lập tức.

### Mô hình Đa luồng (Multithreading)

Khái niệm: Đa luồng cho phép một chương trình thực hiện nhiều tác vụ đồng thời.

ExecutorService: Trong Java, ExecutorService cung cấp cơ chế quản lý luồng (Thread Pool). Trong đồ án, việc kết nối Camera (vốn tốn thời gian và có thể bị chặn - blocking) được đẩy vào một luồng riêng biệt thông qua executor.submit(). Điều này giúp giao diện chính (UI Thread) không bị đơ khi đang chờ kết nối.

Future và Timeout: Sử dụng đối tượng Future để kiểm soát thời gian thực thi của một tác vụ. Nếu quá trình kết nối vượt quá thời gian quy định , hệ thống sẽ ném ra ngoại lệ TimeoutException và hủy tác vụ để giải phóng tài nguyên.Đồng bộ hóa dữ liệu (Synchronization)

## Kết chương

Chương 1 đã trình bày chi tiết và có hệ thống các cơ sở lý thuyết nền tảng để xây dựng ứng dụng quản lý Camera. Từ việc phân tích mô hình kiến trúc Client-Server phù hợp với bài toán phân tán, đến việc đi sâu vào các công nghệ xử lý luồng video chuyên dụng như RTSP và FFmpeg. Bên cạnh đó, các chuẩn giao tiếp mạng hiện đại như WebSocket và RESTful API cùng cơ chế bảo mật JWT cũng được làm rõ vai trò và cách thức hoạt động. Những kiến thức này đóng vai trò là nền tảng vững chắc để tiến hành phân tích, thiết kế và hiện thực hóa hệ thống trong các chương tiếp theo.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## PHÁT BIỂU BÀI TOÁN

* Xây dựng một hệ thống giám sát tập trung có khả năng kết nối đa dạng các loại Camera IP thông qua chuẩn ONVIF và giao thức RTSP/HTTP. Hệ thống bao gồm hai thành phần chính:
* Ứng dụng Client (Desktop App): Được đặt tại các điểm giám sát, chịu trách nhiệm kết nối trực tiếp với Camera, giải mã luồng video, hiển thị thời gian thực và thực hiện thuật toán chụp ảnh tự động để gửi về trung tâm.
* Ứng dụng Server (Web App): Đóng vai trò trung tâm lưu trữ, cung cấp giao diện quản trị để người dùng xem lại lịch sử hình ảnh và cấu hình tham số hoạt động cho các Client từ xa.

## PHÂN TÍCH YÊU CẦU HỆ THỐNG

### Yêu cầu chức năng

* Quản lý kết nối Camera: Cho phép thêm, sửa, xóa thông tin kết nối (IP, Username, Password) và tự động dò tìm Camera trong mạng LAN (ONVIF Discovery)
* Xem trực tiếp (Live View): Hiển thị luồng video từ Camera với độ trễ thấp nhất có thể.
* Tự động chụp và đồng bộ (Auto Capture & Sync): Định kỳ trích xuất khung hình từ luồng video (ví dụ: mỗi 10 giây), lưu trữ tạm thời và gửi lên Server qua RESTful API
* Cấu hình từ xa: Client có thể nhận các thay đổi cấu hình (tần suất chụp, chất lượng ảnh) từ Server mà không cần khởi động lại.

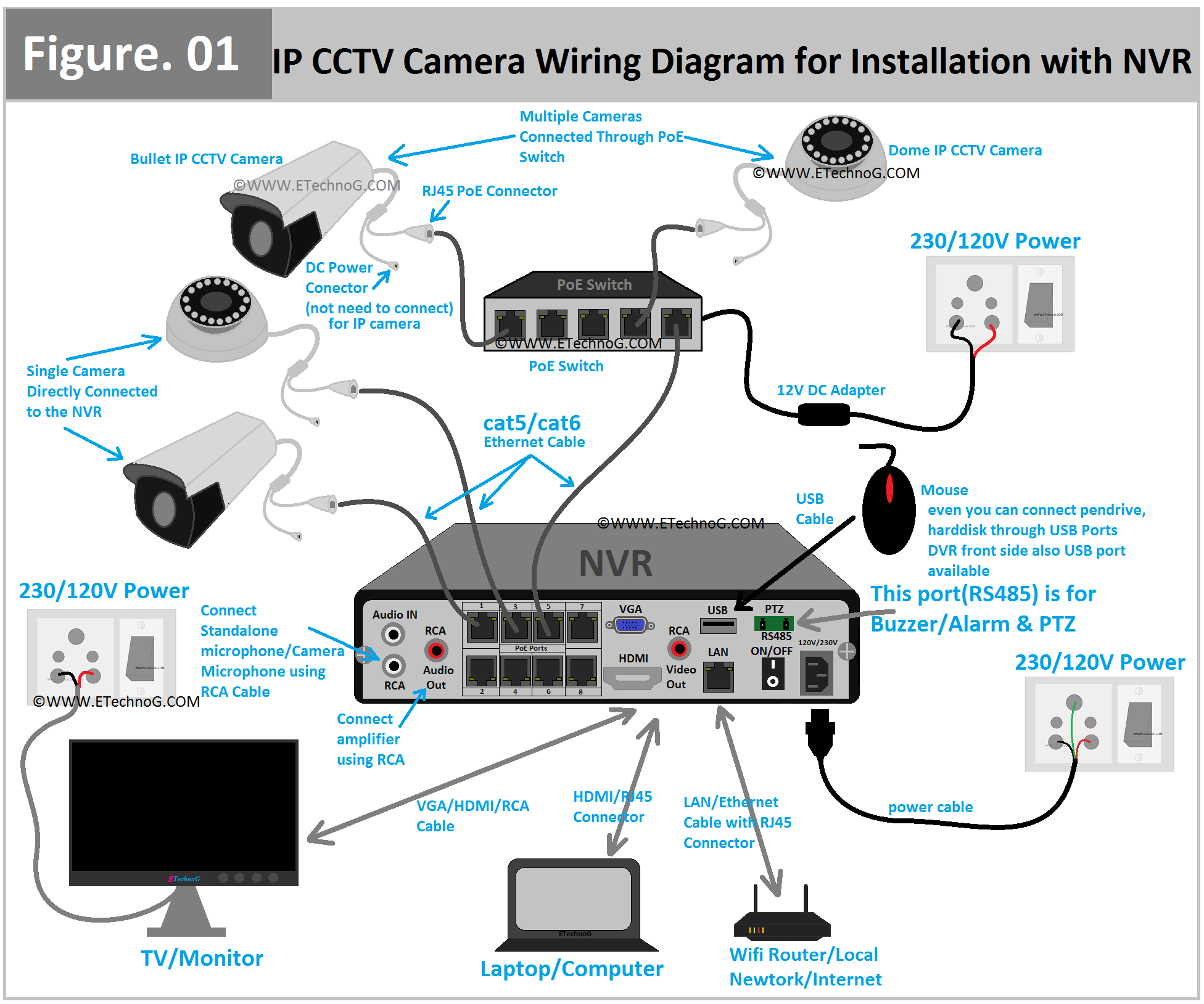
### Yêu cầu phi chức năng

* Tính ổn định: Hệ thống phải có cơ chế tự động xử lý lỗi (timeout, mất kết nối mạng) để không bị treo hoặc dừng hoạt động đột ngột.
* Hiệu năng: Sử dụng cơ chế đa luồng (Multithreading) để đảm bảo việc xử lý video và gửi dữ liệu mạng không làm đơ giao diện người dùng.
* Bảo mật : các dữ liệu lưu trên sever và dữ liệu hình ảnh truyền tảtải được mã hóa
* Trải nghiệm người dùng:giao diện hiện đại , phản hồi tức thì , khi xử lý ngầm đều hiển thị giao diện loading để người dùng biết hệ thống đang làm việc ,tránh cảm giác bị đơ
* Quản lý tài nguyên: Tính toán mức tối ưu tài nguyên của hệ thống để không làm ảnh hưởng tới các ứng dụng khác chạy song song

## PHÂN TÍCH HIỆN TRẠNG

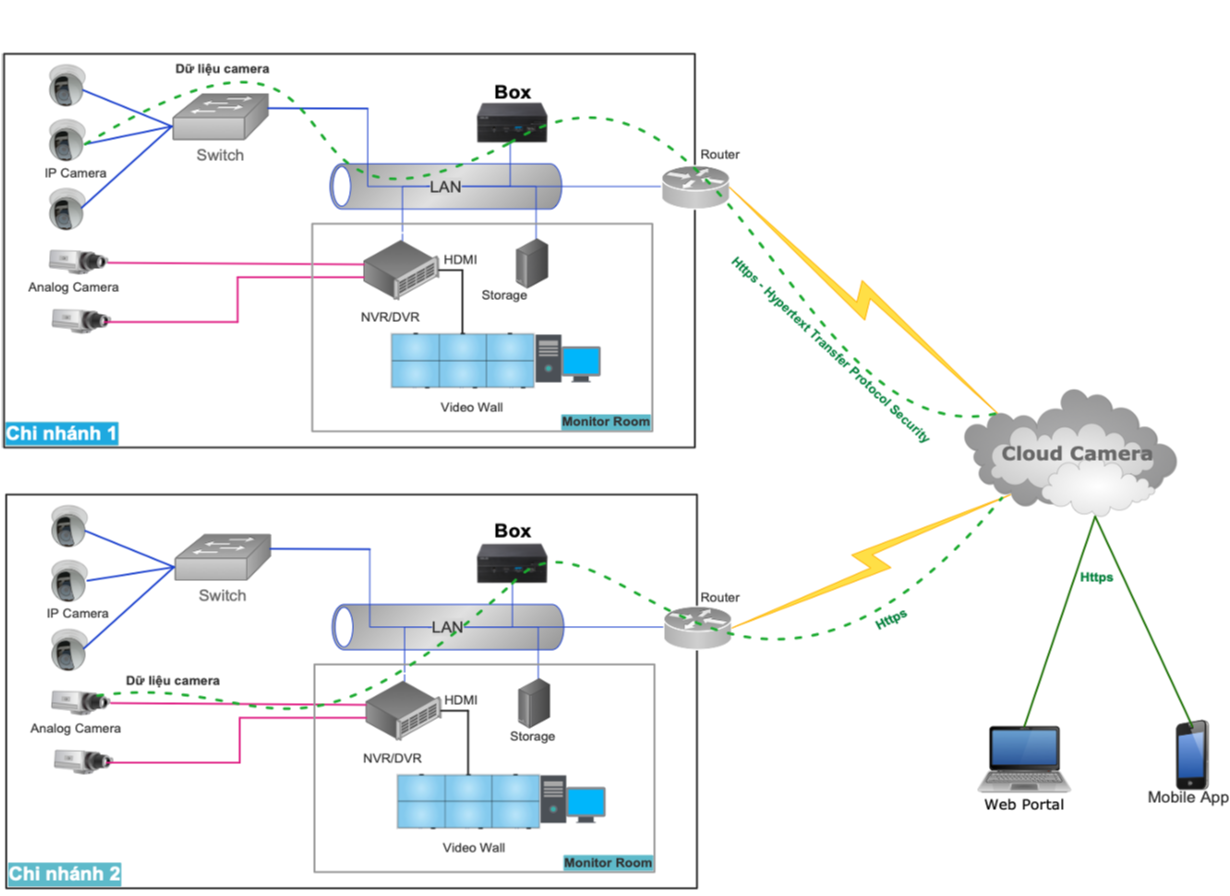
Hiện nay trên thị trường tồn tại hai giải pháp giám sát hình ảnh chủ đạo:

a. Mô hình Camera giám sát tập trung cục bộ (On-Premise CCTV/NVR)



* Kiến trúc & Cơ chế hoạt động: Đây là mô hình hoạt động trong mạng nội bộ (LAN). Các Camera (IP hoặc Analog) truyền tín hiệu video thô về một thiết bị quản lý trung tâm là đầu ghi hình (NVR/DVR).
* Luồng dữ liệu: Camera ➔ Switch/Cáp đồng trục ➔ Đầu ghi (NVR) ➔ Lưu trữ HDD.
* Giao thức: Thường sử dụng giao thức chuẩn ONVIF hoặc RTSP trong mạng LAN, nhưng giao tiếp ra ngoài Internet thường bị đóng gói bởi giao thức riêng của hãng sản xuất (Proprietary Protocol).
* Ưu điểm:
* Độc lập đường truyền: Hoạt động ổn định ngay cả khi mất Internet, do luồng video không đi ra ngoài mạng LAN.
* Chất lượng gốc: Giữ được chất lượng video cao (Bitrate cao) do không bị nén để tiết kiệm băng thông Internet.
* Nhược điểm (Vấn đề cần giải quyết):
* Tính "Đóng" (Silo dữ liệu): Dữ liệu video bị "nhốt" trong đầu ghi. Việc truy xuất luồng video để xử lý (ví dụ: đưa vào phần mềm Java để phân tích AI) rất khó khăn vì phụ thuộc vào SDK của từng hãng phần cứng cụ thể.
* Khả năng mở rộng kém: Khi số lượng Camera vượt quá số kênh của đầu ghi (ví dụ: đầu ghi 16 kênh muốn lắp thêm camera thứ 17), buộc phải thay thế phần cứng mới, gây lãng phí.

b. Mô hình Camera điện toán đám mây (Cloud-based SaaS)



* Kiến trúc & Cơ chế hoạt động: Mô hình này loại bỏ đầu ghi hình vật lý. Mỗi Camera hoạt động như một thiết bị IoT độc lập, kết nối trực tiếp tới máy chủ đám mây (Cloud Server) của nhà cung cấp dịch vụ thông qua Internet.
* Luồng dữ liệu: Camera ➔ Modem/Router ➔ Internet ➔ Cloud Server (Lưu trữ & Xử lý) ➔ Ứng dụng người dùng (Mobile/Web).

-Ưu điểm:

* Triển khai linh hoạt: Không cần cấu hình mạng phức tạp (NAT/Port Forwarding) hay lắp đặt đầu ghi cồng kềnh.
* Truy cập đa nền tảng: Dữ liệu được đồng bộ hóa trên Cloud, dễ dàng truy cập từ nhiều thiết bị khác nhau.

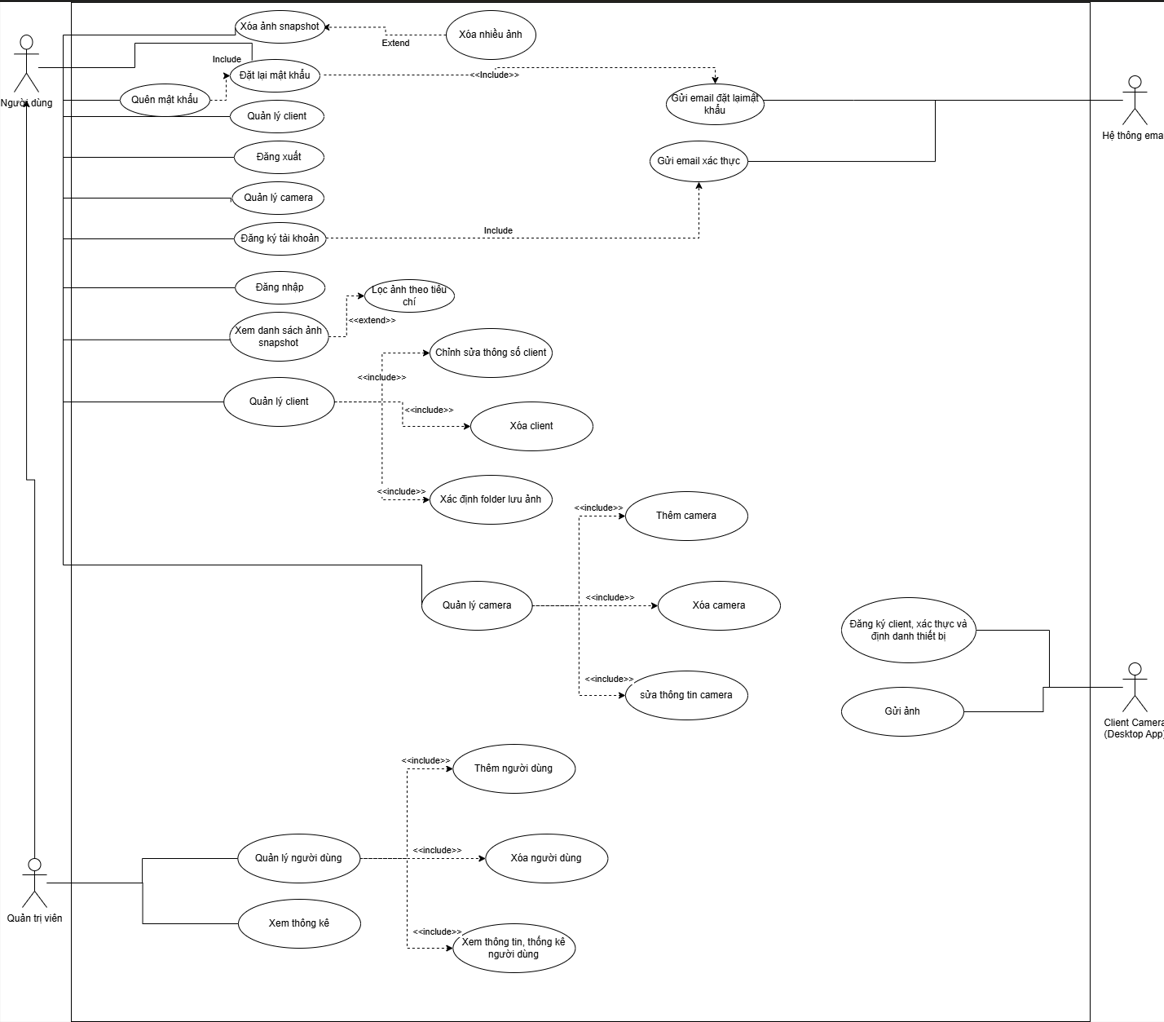
-Nhược điểm (Vấn đề cần giải quyết):

* Phụ thuộc hoàn toàn vào Internet: Nếu mạng chập chờn, dữ liệu video sẽ bị gián đoạn hoặc mất mát. Băng thông mạng bị chiếm dụng lớn do phải upload video liên tục 24/7.
* Độ trễ (Latency) & Khả năng điều khiển: Do tín hiệu phải đi vòng qua Server quốc tế/trong nước rồi mới về máy người dùng, độ trễ thường từ 3-10 giây. Điều này khiến việc ứng dụng vào các bài toán điều khiển thời gian thực (như điều khiển Robot, cảnh báo tức thời trên Desktop App) trở nên bất khả thi.
* Ràng buộc nhà cung cấp (Vendor Lock-in): Người dùng không thể chuyển dữ liệu sang hệ thống khác hoặc tự ý can thiệp thuật toán xử lý hình ảnh vì mã nguồn đóng kín trên Cloud.

## PHÂN TÍCH CHỨC NĂNG

### Đối tượng sử dụng

* Quản trị viên hệ thống (System Administrator):
* **Nền tảng:** Có thể thao tác trực tiếp trên Server hoặc qua giao diện Web Admin.
* **Chức năng:** Quản lý tài khoản người dùng, cấu hình tham số hệ thống, quản lý cơ sở dữ liệu.
* Người dùng quản trị/Giám sát từ xa (Web User):
* **Nền tảng:** Trình duyệt Web (Chrome, Firefox, v.v.).
* **Vai trò:** Người dùng muốn truy cập dữ liệu mọi lúc mọi nơi mà không cần cài đặt phần mềm.
* **Chức năng:** Đăng nhập hệ thống, xem danh sách các Camera đang hoạt động, xem hình ảnh mới nhất hoặc xem lại lịch sử, xem các báo cáo thống kê, thay đổi thông số client trực ti
* Người dùng tại điểm giám sát (Desktop Client User):
* **Nền tảng:** Ứng dụng Desktop (Java Swing/FlatLaf).
* **Vai trò:** Là các máy trạm kết nối trực tiếp với Camera.
* **Chức năng:** Tự động kết nối Camera, thu thập hình ảnh, sơ chế (nén/mã hóa) và gửi về Server qua Socket TCP.



*Hình 2.1 Biểu đồ chức năng*

### Các Chức năng Chính của Hệ thống

Chức năng Xác thực & Quản lý Tài khoản (Authentication & Account Management):

* Đăng nhập (Login): Xác thực người dùng bằng username và password.
* Đăng ký (Register): Cho phép người dùng tạo tài khoản mới.
* Xác thực Email (Email Verification): Gửi và xác nhận email để kích hoạt tài khoản.
* Quên mật khẩu (Forgot Password): Gửi link đặt lại mật khẩu qua email.
* Đặt lại mật khẩu (Reset Password): Cho phép người dùng đặt lại mật khẩu mới.
* Đăng xuất (Logout): Xóa phiên đăng nhập và token.
* Lấy thông tin người dùng hiện tại (/me): Truy xuất thông tin của người dùng đang đăng nhập.

Chức năng Quản lý Client (Client Management - Desktop Application):

* Đăng ký Client App (Register Client Application): Ứng dụng client desktop đăng ký với server (bằng machineId) để được định danh.
* Ghi nhận trạng thái Online/Offline: Server theo dõi trạng thái kết nối của client desktop qua WebSocket.
* Cập nhật thông tin Client: Xem, chỉnh sửa các thuộc tính của một client.
* Xóa Client: Loại bỏ một thiết bị client khỏi hệ thống.
* Lấy danh sách Client: Truy xuất danh sách các client thuộc sở hữu của người dùng.

Chức năng Quản lý Camera (Camera Management):

* Tạo Camera: Đăng ký một camera mới vào hệ thống.
* Xem thông tin Camera: Truy xuất chi tiết của một camera cụ thể.
* Cập nhật thông tin Camera: Chỉnh sửa các thuộc tính của camera.
* Xóa Camera: Loại bỏ một camera khỏi hệ thống.
* Lấy danh sách Camera: Truy xuất danh sách các camera (theo client hoặc tất cả).
* Cập nhật trạng thái Camera: Bật/tắt camera.

Chức năng Xử lý & Quản lý Hình ảnh/Video (Image/Video Processing & Management):

* Gửi Live Feed Frame: Ứng dụng client desktop gửi từng frame ảnh của luồng trực tiếp lên server.
* Phân phối Live Feed: Server nhận frame ảnh và phát tán (qua WebSocket) tới các web client đang xem.
* Upload Snapshot: Client tải lên các ảnh chụp tĩnh.
* Xem Snapshot: Truy xuất và hiển thị các ảnh đã tải lên.
* Lọc & Phân trang Snapshot: Tìm kiếm ảnh theo cameraId, khoảng thời gian và phân trang kết quả.
* Xóa Snapshot: Xóa một hoặc nhiều ảnh snapshot.
* Chức năng Thông báo Thời gian thực (Real-time Notifications):
* Thông báo Snapshot mới: Server thông báo đến web client khi có ảnh snapshot mới được tải lên.
* Thông báo Cập nhật Client: Server thông báo đến web client khi thông tin của một client được thay đổi.

Chức năng Quản lý Người dùng (User Management - dành cho Admin):

* Tạo Người dùng: Admin có thể tạo tài khoản người dùng mới.
* Xem Danh sách Người dùng: Admin xem tất cả tài khoản người dùng.
* Cập nhật Người dùng: Admin chỉnh sửa thông tin tài khoản người dùng.
* Xóa Người dùng: Admin xóa tài khoản người dùng.

### Các Thuật toán và Kỹ thuật Cốt lõi

1. Thuật toán Điều phối Kết nối Giới hạn

* Mục đích: Khi khởi động ứng dụng với N camera (ví dụ 20 cái), nếu khởi tạo tất cả cùng lúc, CPU và băng thông mạng sẽ bị quá tải , dẫn đến treo ứng dụng. Thuật toán này sử dụng Semaphore để giới hạn số lượng kết nối được thiết lập song song tại một thời điểm.
* Mã giả:

Input: List<Camera> cameras, int K (limit)

Output: Stream connections established

Initialize Semaphore permits = new Semaphore(K)

Initialize ThreadPool dispatcher

FUNCTION ConnectAllCameras(cameras):

    FOR EACH camera IN cameras:

        dispatcher.submit(TASK:

            TRY:

                // 1. Xin giấy phép. Nếu hết giấy, luồng sẽ CHỜ ở đây

                permits.acquire()

                // 2. Thực hiện kết nối nặng

                StartCameraStream(camera, ON\_COMPLETE callback)

            CATCH Error:

                permits.release() // Trả giấy phép nếu lỗi

        )

CALLBACK ON\_COMPLETE():

    // 3. Khi camera kết nối xong (thành công hoặc thất bại)

    // Trả giấy phép để camera tiếp theo trong hàng đợi được chạy

    permits.release()

* Độ phức tạp thời gian (Time Complexity):Thuật toán cân bằng giữa tốc độ và độ ổn định. Thời gian khởi động tăng tuyến tính theo số lượng camera, độ phức tạp O(
* Độ phức tạp không gian (Space Complexity): O(n) để lưu trữ danh sách các tác vụ (Tasks) trong hàng đợi của ThreadPool.

2. Kỹ thuật Vòng lặp Thu thập & Hồi phục

* Mục đích: Duy trì luồng video liên tục, xử lý hiện tượng mất gói tin hoặc camera tạm thời ngắt kết nối mà không làm chết luồng chạy nền.
* Mã giả

FUNCTION RunStream(url):

    Initialize Grabber(url)

    Start Grabber with Timeout T\_start

    WHILE (isRunning):

        TRY:

            // 1. Thu thập khung hình

            Frame f = grabber.grab()

            // 2. Xử lý Frame rỗng (Mất tín hiệu tạm thời)

            IF f IS NULL:

                Wait(BACKOFF\_TIME) // Nghỉ một chút để tránh spam CPU

                CONTINUE

            // 3. Xử lý ảnh (Convert to Image)

            Image img = Convert(f)

            // 4. Gửi sang UI (Bất đồng bộ)

            Platform.runLater(() -> UpdateUI(img))

            // 5. Điều tiết FPS (ngủ bù thời gian dư)

            Sleep(1000 / FPS - ProcessTime)

        CATCH Exception e:

            Log Error

            Wait(RETRY\_DELAY) // Chờ trước khi thử lại

            AttemptReconnection()

* Độ phức tạp thời gian: O (N\*W\*H)
* Độ phức tạp không gian:O(W\*H)

3. Thuật toán Kiểm tra Sớm

* Mục đích: Tránh lãng phí tài nguyên khởi tạo FFmpegFrameGrabber (rất nặng) cho các camera đã chết hoặc sai IP. Hệ thống dùng TCP Socket nhẹ để "thăm dò" trước.

-Mã giả:  
FUNCTION SmartConnect(ip, port, url):

    // Bước 1: Kiểm tra nhẹ (Lightweight Check)

    Start Timer

    TRY:

        Socket s = new Socket()

        // Thử kết nối TCP trong 1.5s

        s.connect(ip, port, timeout=1500ms)

        Close s

    CATCH:

        RETURN False // Camera Offline, hủy ngay lập tức

    // Bước 2: Nếu bước 1 qua, mới thực hiện kết nối nặng

    Initialize HeavyRTSPGrabber(url)

    RETURN True

* Độ phức tạp thời gian : O(1)

4.Thuật toán WS-Discovery

* Mục đích: Đây là kỹ thuật cốt lõi để tìm kiếm các Camera IP hỗ trợ chuẩn ONVIF (Open Network Video Interface Forum). Thay vì phải đoán IP của camera, phần mềm gửi một tin nhắn "Hỏi thăm" ra toàn mạng và chờ Camera tự trả lời.
* Mã giả:

CONST MULTICAST\_IP = "239.255.255.250"

CONST PORT = 3702

CONST TIMEOUT = 4000ms

FUNCTION DiscoverONVIF\_Cameras():

    // 1. Chuẩn bị socket

    Socket s = CreateMulticastSocket()

    Packet probe = CreateSOAPProbeMessage() // XML hỏi thăm

    // 2. Gửi tin nhắn Broadcast

    Send(s, probe, MULTICAST\_IP, PORT)

    // 3. Vòng lặp lắng nghe phản hồi

    StartTime = Now()

    List discovered = []

    WHILE (Now() - StartTime < TIMEOUT):

        TRY:

            Packet response = Receive(s, Timeout=RemainingTime)

            String xml = ConvertToString(response.Data)

            // Kỹ thuật Lightweight Parsing bằng Regex (Tối ưu hóa)

            // Tìm thẻ <XAddrs> chứa IP camera

            String ip = RegexExtract(xml, "<XAddrs>(.\*?)</XAddrs>")

            IF ip is Valid AND ip NOT in discovered:

                discovered.add(ip)

        CATCH Timeout:

            BREAK // Hết giờ chờ đợi

    RETURN discovered

* Độ phức tạp thời gian:O(T) với T là thời gian timeout
* Độ phức tạp không gian:O(1)

5. Thuật toán Quét mạng chủ động

* Mục đích: Trong mã nguồn có hàm getAllIPv4Addresses, điều này gợi ý kỹ thuật quét dải IP (Subnet Sweep). Kỹ thuật này dùng để tìm các camera đời cũ không hỗ trợ ONVIF hoặc bị tắt tính năng Discovery
* Mã giả:

FUNCTION ActiveScan(int targetPort):

    // 1. Xác định dải mạng

    String myIP = GetLocalIP() // VD: 192.168.1.10

    String subnet = GetSubnetPrefix(myIP) // VD: 192.168.1.

    List futures = []

    ThreadPool pool = CreateThreadPool(50) // 50 luồng chạy song song

    // 2. Duyệt qua 254 địa chỉ khả thi

    FOR i FROM 1 TO 254:

        String targetIP = subnet + i

        Task t = pool.submit(() -> {

            // Thử kết nối TCP (3-way handshake)

            Socket s = new Socket()

            TRY:

                s.connect(targetIP, targetPort, Timeout=200ms)

                RETURN targetIP // Tìm thấy thiết bị mở cổng này

            CATCH:

                RETURN NULL // Không có thiết bị

        })

        futures.add(t)

    // 3. Tổng hợp kết quả

    List foundCameras = []

    FOR f IN futures:

        Result res = f.get()

        IF res != NULL: foundCameras.add(res)

    RETURN foundCameras

* Độ phức tạp thời gian: O( với S là kích thước subnet,K là số luồng ,T là thời gian timeout
* Độ phức tạp không gian: O(S)

6. Lưu file ảnh, lưu metadata vào database, thông báo qua websocket

* Mục đích: nhận file ảnh từ client/app, lưu file vào cấu trúc thư mục có dạng {clientId}/{cameraId}/yyyy/MM/dd/{uniqueFilename}, lưu metadata vào bảng ImageEntity, cập nhật trạng thái client (ACTIVE) và gửi thông báo WebSocket cho user sở hữu camera.
* Mã giả

// Lấy camera; nếu không tồn tại thì ném lỗi

camera = cameraRepository.findById(cameraId) or throw

// Lấy clientId để lưu cấu trúc thư mục và cập nhật trạng thái client

clientId = camera.client != null ? camera.client.id : 0

updateClientStatusToActive(clientId) // mark client active

// Tạo tên file duy nhất để tránh trùng lặp

originalName = file.getOriginalFilename()

ext = extractExtension(originalName) // có thể rỗng

uniqueName = UUID() + ext

//Tạo đường dẫn theo ngày để tổ chức file (clientId/cameraId/yyyy/MM/dd/filename)

datePath = format(capturedAt, "yyyy/MM/dd")

relativePath = join(clientId, cameraId, datePath, uniqueName) // dùng / cho URL

// Chuẩn bị thư mục đích trên filesystem

targetDir = rootUploadDir.resolve(join(clientId, cameraId, datePath))

createDirectoriesIfNotExist(targetDir) // đảm bảo folder tồn tại

// Ghi file (stream copy) - chi phí thời gian phụ thuộc kích thước file

copy(file.inputStream, targetDir/uniqueName, REPLACE\_EXISTING)

// Tạo ImageEntity và lưu metadata vào DB

imageEntity.relativePath = relativePath

imageEntity.fileSizeKb = file.size / 1024.0

imageEntity.capturedAt = capturedAt.toLocalDateTime()

imageEntity.uploadedAt = nowTimestamp()

imageEntity.camera = camera

saved = imageRepository.save(imageEntity) // INSERT

// Thông báo WebSocket tới chủ sở hữu (không block quá nặng nếu lỗi)

try:

    dto = toDto(saved)

    dto.filePath = buildFileUrl(dto.filePath) // map sang URL xem ảnh

    wsMsg = { type: "NEW\_IMAGE", data: dto }

    webSocketHandler.sendMessageToUser(ownerUsername, toJson(wsMsg))

catch Exception:

    log("WS send failed") // không rollback lưu file/DB nếu WS lỗi

return toDto(saved)

* Độ phức tạp:
* Thời gian: O(K + DB). Với K là kích thước dữ liệu, DB là thời gian truy suất dữ liệu trong database
* Không gian: O(K)

7. Lấy danh sách ảnh có phân trang và lọc theo camera / ngày

* Mục đích: cung cấp API phân trang ảnh cho user, với tuỳ chọn lọc theo camera hoặc khoảng ngày.
* Mã giả

// chuẩn bị khoảng thời gian nếu null -> dùng giá trị cực lớn/nhỏ

if start == null: start = very\_early\_date

if end == null: end = now + 1 day

if cameraId != null:

    if hasDateFilter:

        page = imageRepository.findByCameraIdAndCapturedAtBetweenOrderByCapturedAtDesc(cameraId, start, end, pageable)

    else:

        page = imageRepository.findByCameraIdOrderByCapturedAtDesc(cameraId, pageable)

else:

    if hasDateFilter:

        page = imageRepository.findByCameraClientUserIdAndCapturedAtBetweenOrderByCapturedAtDesc(userId, start, end, pageable)

    else:

        page = imageRepository.findByCameraClientUserIdOrderByCapturedAtDesc(userId, pageable)

// Chuyển Page<ImageEntity> -> Page<Image DTO> một cách lazy bằng map

return page.map(entity -> toDto(entity))

* Độ phức tạp:
* Thời gian: O(P + DB). Với P là kích thước trang, DB là thời gian truy vấn trong cơ sở dữ liệu
* Không gian: O(P). Vơi P là kích thước trang

8. Quản lý websocket session

* Mục đích: quản lý kết nối WebSocket cho cả App (client devices) và Web (user dashboards). Lưu/thu hồi session theo username, đánh dấu client online/offline, gửi thông điệp tới tất cả session của 1 user.
* Mã giả:

// Khi nhận AUTH message:

if tokenProvider.validateToken(token):

    username = tokenProvider.getUsernameFromJWT(token)

    // Lưu session vào tập các session của user (multi-session)

    userSessions = sessionsByUsername.computeIfAbsent(username, newConcurrentSet)

    userSessions.add(session)

    userBySessionId.put(session.id, username)

    // Nếu client là APP: xác thực machineId -> set client status ACTIVE

    if clientType == "APP":

        machineId = msg.machineId or closeSession("machineId required")

        userEntity = userRepository.findByUsername(username) or close

        client = clientRepository.findByMachineIdAndUser(machineId, userEntity)

        if client present:

            client.setStatus("ACTIVE"); clientRepository.save(client)

            clientIdBySessionId.put(session.id, client.id)

// Khi connection đóng:

clientId = clientIdBySessionId.remove(session.id)

if clientId != null:

    client = clientRepository.findById(clientId)

    client.setStatus("OFFLINE"); clientRepository.save(client)

    cameraRepository.updateAllByClientId(client.id, false) // tắt tất cả camera của client

username = userBySessionId.remove(session.id)

if username != null:

    sessions = sessionsByUsername.get(username)

    sessions.remove(session)

    if sessions empty: sessionsByUsername.remove(username)

// Gửi tin đến user:

sendMessageToUser(username, json):

    sessions = sessionsByUsername.get(username)

    if sessions == null or empty: return // user offline

    for s in sessions:

        if s.isOpen():

            s.sendMessage(json) // có I/O; nếu nhiều session, lặp theo S sessions

* Độ phức tạp:
* Thời gian: O(S) với S là số session của một username
* Không gian: O(U + S) với U là số lượng user đang có phiên hoạt động, S là số session của mỗi user đó

### Công nghệ sử dụng

|  |  |  |
| --- | --- | --- |
| STT | Hạng mục | Công nghệ / Thư viện cụ thể |
| 1 | Ngôn ngữ | Java (JDK 11+) |
| 2 | Giao diện (UI) | JavaFX, FXML, CSS |
| 3 | Video Engine | JavaCV, FFmpeg (FFmpegFrameGrabber) |
| 4 | Network Client | OkHttp 4.x, Java WebSocket |
| 5 | Data Format | JSON (Jackson Library), XML (Regex Parsing) |
| 6 | Protocols | RTSP, HTTP/HTTPS, UDP Multicast (ONVIF) |
| 7 | Concurrency | ExecutorService, Semaphore, JavaFX Task |
| 8 | Framework (Backend) | Spring Boot (Spring Web, Spring MVC) |
| 9 | Cơ sở dữ liệu | MySQL (mysql-connector-j) |
| 10 | Bảo mật (Server) | Spring Security, JWT (io.jsonwebtoken) |
| 11 | Real-time Server | Spring Boot WebSocket (Starter WebSocket) |
| 12 | Tiện ích mở rộng | Spring Boot Mail, Java NIO (File I/O) |
| 13 | Quản lý dự án | Apache Maven (pom.xml) |
| 14 | Kiểm thử (Testing) | Postman, Spring Boot Test |

## THIẾT KẾ CƠ SỞ DỮ LIỆU

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.2: Sơ đồ tổng thể

## XÂY DỰNG CHƯƠNG TRÌNH

Workspace (PBL4)

├── camera-common

│ ├── src/main/java

│ └── pbl4.common

│ └── model

│

├── camera-client

│ ├── pom.xml

│ ├── src/main/java

│ │ └── com.pbl4.cameraclient

│ │ ├── main

│ │ ├── model

│ │ ├── network

│ │ ├── service

│ │ └── ui

│ │ └── controller

│ └── src/main/resources

│ └── com/pbl4/cameraclient

│ └── ui

│ └── view

│ ├── config.properties

│ └── logback.xml

│

└── server

├── pom.xml

├── src/main/java

│ └── com.pbl4.server

│ ├── config

│ ├── controller

│ ├── dto

│ ├── entity

│ ├── repository

│ ├── security

│ ├── service

│ └── websocket

│ └── ServerApplication.java

└── src/main/resources

├── application.properties

├── static/

│ ├── admin

│ ├── assets

│ │ ├── css

│ │ ├── images

│ │ └── js

│ └── <various\_html\_files>

└── templates/

└── uploads/

└── <userID>/

└── <clientID>/

└── <year>/

└── <month>/

└── <day>/

## Kết chương

Trong chương này, nhóm thực hiện đã tập trung phân tích và thiết kế chi tiết cho hệ thống "Ứng dụng truy xuất và phân tích dữ liệu hình ảnh từ Camera qua mạng". Các nội dung cốt lõi đã đạt được bao gồm:

* Về mặt kiến trúc: Đã xác định và xây dựng mô hình hệ thống dựa trên kiến trúc Client-Server mở rộng với 3 thành phần chính tương tác chặt chẽ: AppClient (Thu thập), Server (Xử lý trung tâm) và WebClient (Quản trị). Mô hình này đảm bảo tính linh hoạt, khả năng mở rộng và phù hợp với bài toán quản lý thiết bị phân tán.
* Về mặt dữ liệu: Đã thiết kế hoàn chỉnh sơ đồ Cơ sở dữ liệu (ERD) với các bảng lưu trữ tối ưu cho việc quản lý người dùng, thiết bị và lịch sử hình ảnh.
* Về mặt giải thuật: Đã xây dựng được các lưu đồ thuật toán và quy trình xử lý nghiệp vụ quan trọng như: cơ chế tự động định danh thiết bị (Provisioning), quy trình thu thập - nén - gửi ảnh từ AppClient và quy trình xác thực - lưu trữ tại Server.

Các kết quả phân tích và thiết kế trong chương này là cơ sở lý thuyết và kỹ thuật vững chắc để tiến hành xây dựng ứng dụng thực tế. Trong chương tiếp theo (Chương 3: Triển khai và Đánh giá kết quả), nhóm sẽ trình bày chi tiết về môi trường cài đặt, các thư viện lập trình được sử dụng và các kịch bản kiểm thử để chứng minh tính đúng đắn của hệ thống đã thiết kế.

# TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

## MÔ HÌNH TRIỂN KHAI

### Môi trường triển khai

Hệ thống được triển khai trên mô hình mạng cục bộ (LAN) tuân thủ theo yêu cầu của đề tài, bao gồm các thành phần vật lý và logic sau:

Thiết bị đầu cuối (Hardware):

* Máy tính (Workstation): Đóng vai trò là nơi chạy ứng dụng Client (thu thập ảnh) và Server (Web quản lý).
* Camera IP: Sử dụng Camera hỗ trợ chuẩn ONVIF và cung cấp luồng dữ liệu qua giao thức RTSP/HTTP. Thiết bị được cấp nguồn và kết nối vào mạng qua cáp Ethernet.
* Thiết bị mạng (Hub/Switch & Router):
* Modem Router đóng vai trò Gateway, kết nối Internet và chạy dịch vụ DHCP để cấp phát địa chỉ IP động cho các thiết bị trong mạng.
* Hub/Switch dùng để mở rộng cổng mạng, kết nối Camera và Máy tính.

Mô hình kết nối mạng:

* Tất cả thiết bị (Camera, Máy tính) nằm trong cùng một lớp mạng (Subnet), ví dụ: 192.168.1.x.
* Máy tính có thể kết nối TCP Socket với Camera.

### Cấu hình hệ thống

Cấu hình kết nối Camera (Client) Các tham số kỹ thuật được thiết lập để đảm bảo tính ổn định và chất lượng dữ liệu:

* Giao thức truyền tải (RTSP Transport): Sử dụng TCP (rtsp\_transport = tcp) thay vì UDP để tránh mất gói tin khi đường truyền không ổn định.
* Định dạng Pixel: Ép kiểu về BGR24 (avutil.AV\_PIX\_FMT\_BGR24) để tương thích tốt nhất với bộ xử lý ảnh của Java và giảm lỗi cảnh báo từ FFmpeg.
* Cơ chế Timeout (Chống treo hệ thống):
* Socket Timeout (stimeout): 5.000.000 µs (5 giây) - Thời gian tối đa chờ gói tin mạng.
* Read Timeout (timeout): 10.000 ms (10 giây) - Thời gian tối đa chờ đọc frame tiếp theo.
* Start Timeout: 30 giây - Thời gian tối đa chờ khởi tạo kết nối ban đầu.

Cấu hình vận hành (Operational Config)

* Tần suất chụp ảnh (Capture Interval): Mặc định 5 giây/lần (có thể cấu hình nóng từ Server).
* Bộ đệm video (Video Buffer):
* Server API Endpoint:

## KẾT QUẢ THỰC NGHIỆM

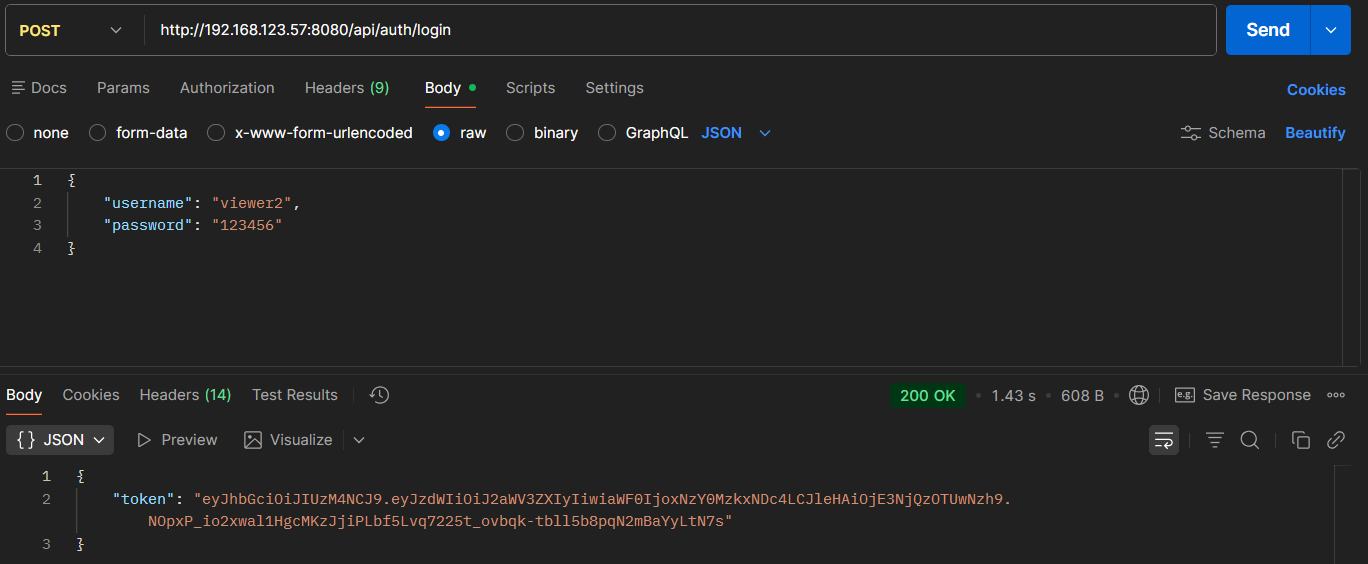
### Kịch bản 1: Chức năng đăng nhập hệ thống

* Mô tả: Người dùng cung cấp tên đăng nhập (username) và mật khẩu (password) để xác thực danh tính và truy cập vào các tính năng được bảo vệ của hệ thống. Đây là cánh cổng đầu tiên vào ứng dụng web và desktop client.
* Đầu vào:

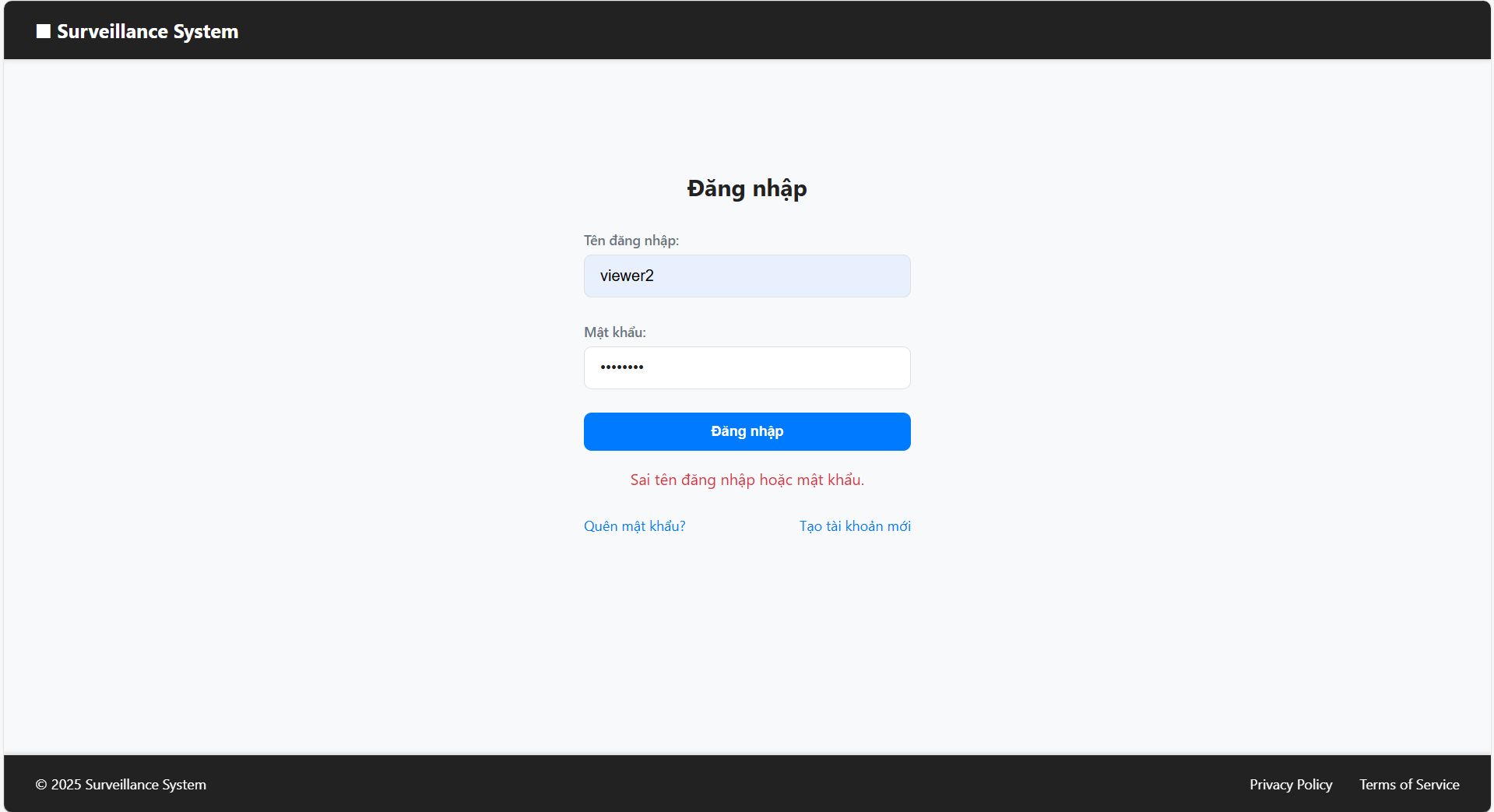
username (chuỗi): Tên người dùng.

password (chuỗi): Mật khẩu người dùng.

* Kết quả:
* Thành công: Trả về một JWT (JSON Web Token). Token này sẽ được client lưu trữ và gửi kèm trong header Authorization của tất cả các request API tiếp theo yêu cầu xác thực. Đồng thời, client cũng nhận được thông tin cơ bản của người dùng (ID, username, role) để phục vụ việc phân quyền và hiển thị giao diện.

  
*Hình 3.1: Login thành công được kiểm thử trên postman*

* Thất bại: Trả về thông báo lỗi (ví dụ: "Tên đăng nhập hoặc mật khẩu không đúng", "Tài khoản chưa được kích hoạt") và mã lỗi HTTP (401 Unauthorized, 400 Bad Request).

  
*Hình 3.2: Login thất bại do sai mật khẩu*

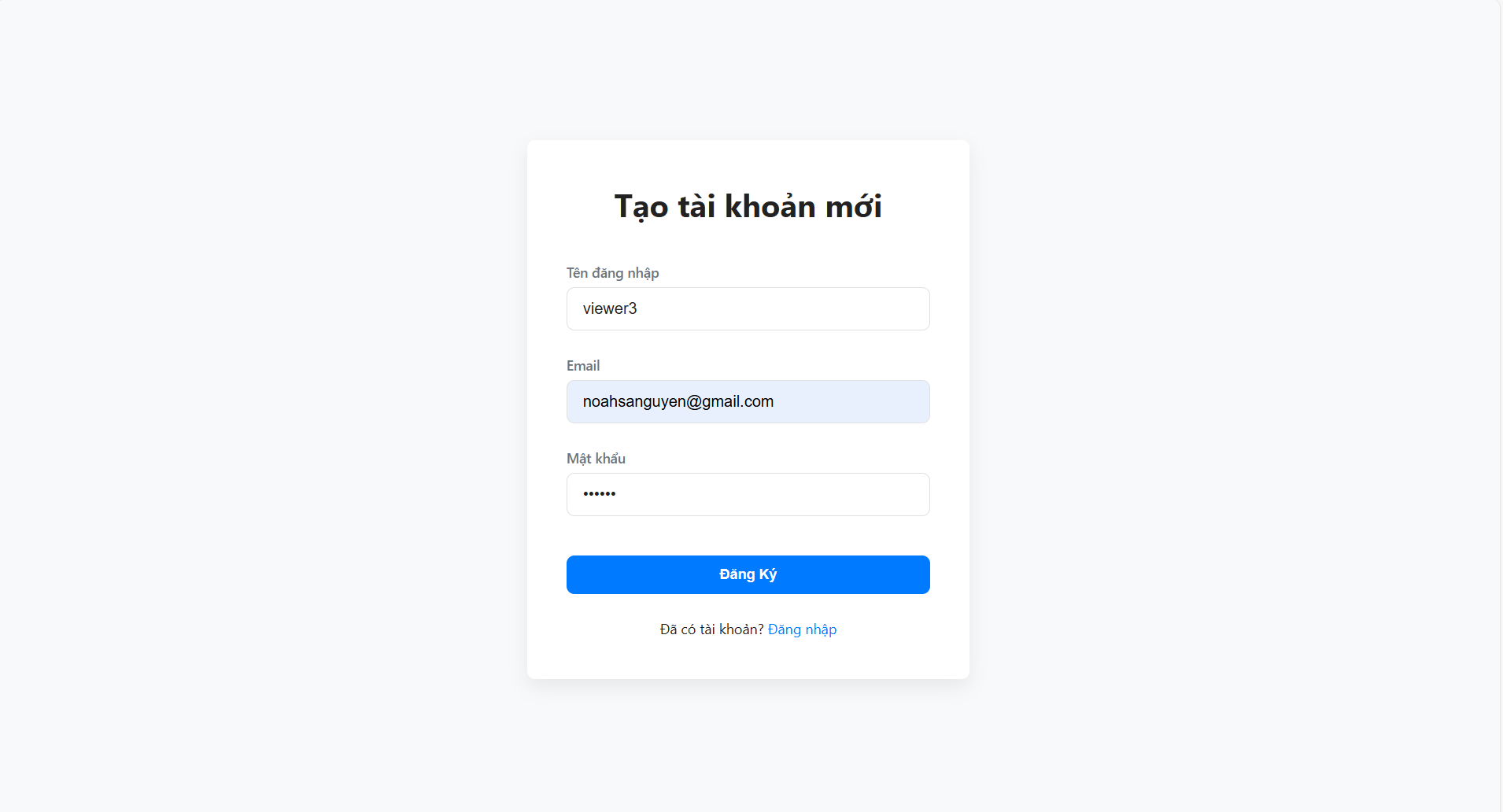
* Nhận xét:
* Quan trọng: Đây là chức năng cốt lõi cho bảo mật. Việc sử dụng JWT là một phương pháp phổ biến và hiệu quả cho xác thực stateless.
* Mã hóa mật khẩu: Mật khẩu được gửi qua kết nối an toàn (hy vọng là HTTPS trong môi trường production) và được hash (băm) trên server trước khi lưu trữ, không bao giờ lưu dưới dạng văn bản thuần.
* Tích hợp: Chức năng này được sử dụng bởi cả camera-client (desktop app) và pbl4-web (web frontend), chứng tỏ tính nhất quán của API xác thực.
* Phân quyền: Vai trò (role) của người dùng được lấy sau khi đăng nhập thành công và được sử dụng để chuyển hướng người dùng đến các trang/tính năng phù hợp (ví dụ: Admin Dashboard vs. Live Feed).

### Kịch bản 2: Chức năng: Đăng ký tài khoản (Register Account)

* Mô tả: Cho phép người dùng mới tạo một tài khoản trong hệ thống bằng cách cung cấp các thông tin cần thiết.
* Đầu vào:
* username (chuỗi): Tên đăng nhập mong muốn.
* email (chuỗi): Địa chỉ email của người dùng.
* password (chuỗi): Mật khẩu.

(Có thể bao gồm các trường khác như fullName, v.v. tùy thuộc vào model User)

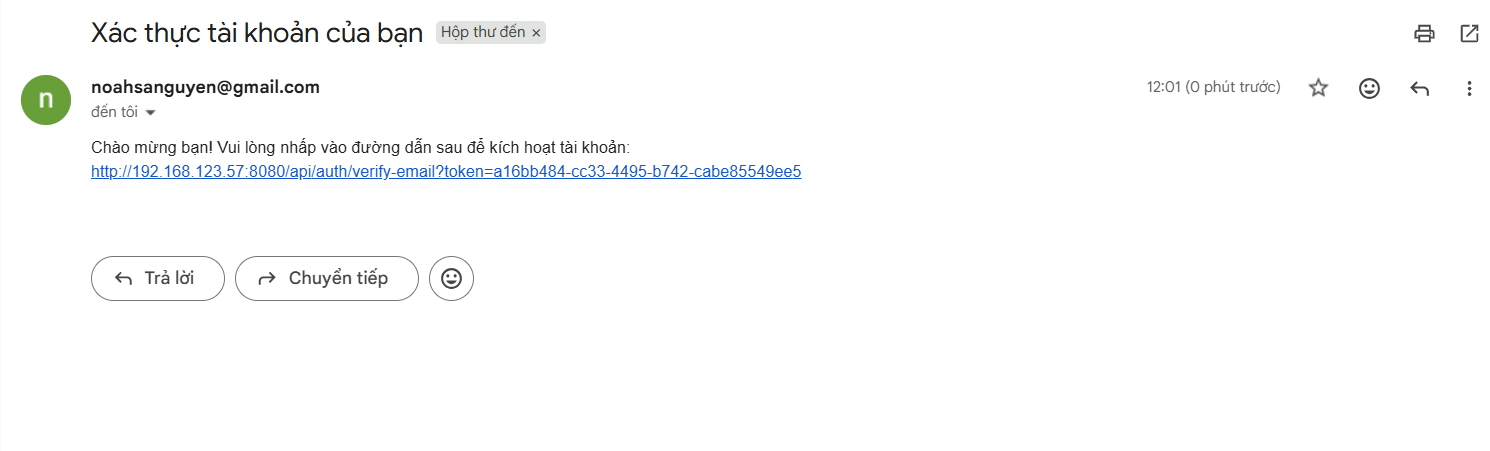
* Kết quả:
* Thành công: Trả về thông báo xác nhận rằng tài khoản đã được tạo và một email xác thực đã được gửi. Mã HTTP 201 Created.



*Hình 3.3: Bổ sung thông tin tạo tài khoản*

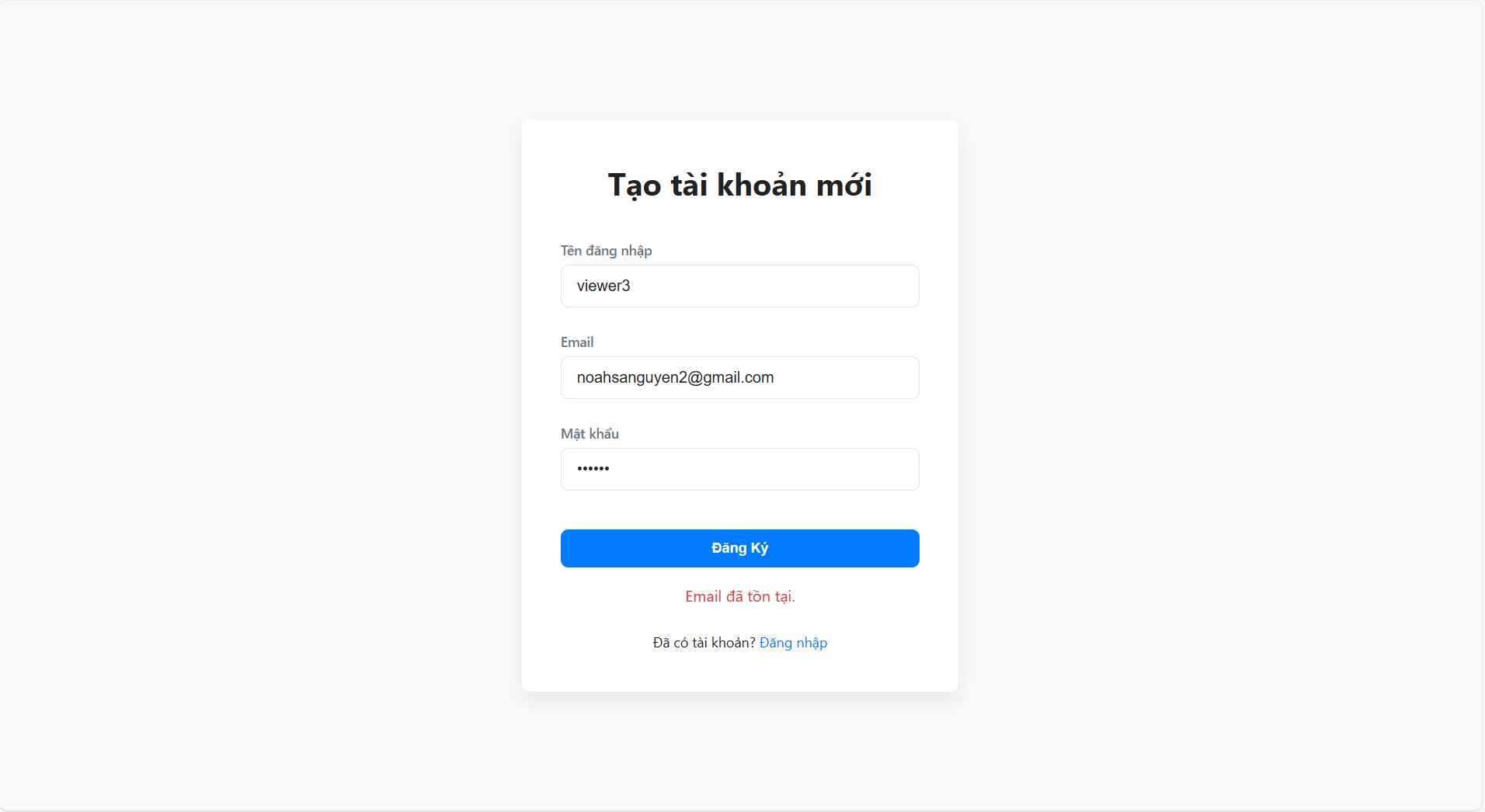


*Hình 3.4: Thông báo đăng ký thành công*



*Hình 3.5: Xác thực email*

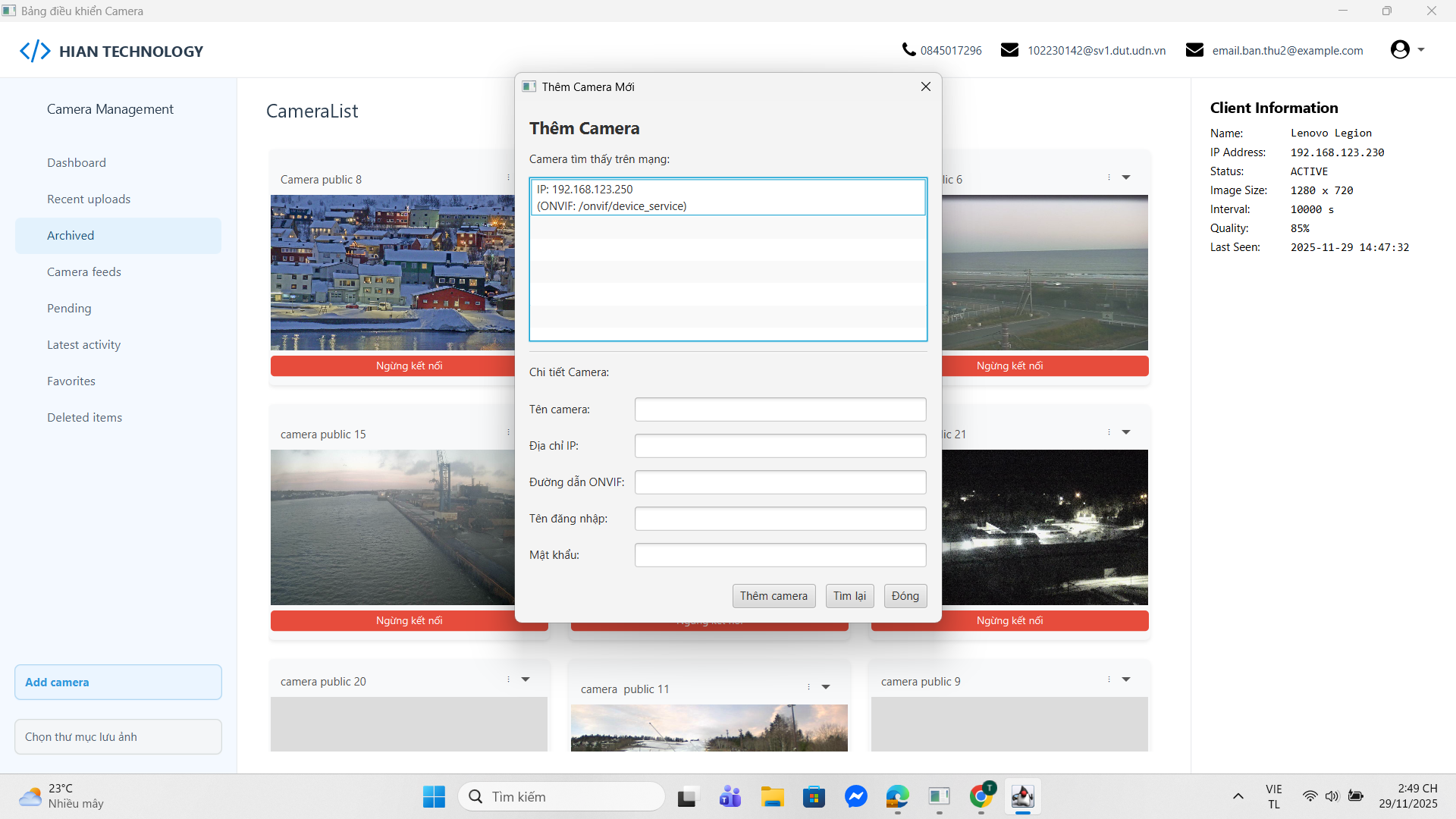
* Thất bại: Trả về thông báo lỗi (ví dụ: "Tên đăng nhập đã tồn tại", "Email đã tồn tại", "Mật khẩu không đủ mạnh") và mã lỗi HTTP (400 Bad Request, 500 Internal Server Error).

  
Hình 3.4: Lỗi email đã tồn tại khi đăng ký tài khoản

* Nhận xét:
* Email xác thực: Hệ thống yêu cầu xác thực email để đảm bảo tính hợp lệ của địa chỉ email và tăng cường bảo mật.
* Public API: Đây là một public API (không yêu cầu JWT để gọi) để người dùng chưa đăng nhập có thể tạo tài khoản.
* Validation: Cần có logic kiểm tra dữ liệu đầu vào (validation) để đảm bảo dữ liệu hợp lệ trước khi lưu trữ.

### Kịch bản 3: Chức năng: Quản lý Camera của tôi (Manage My Cameras)

* Mô tả: Người dùng có thể xem danh sách các camera của họ, xem chi tiết từng camera, cập nhật thông tin camera (ví dụ: tên, mô tả, bật/tắt), và xóa camera.
* Đầu vào:
* Xem danh sách: Có thể lọc theo clientId (tùy chọn), ngoài JWT của người dùng.
* Xem chi tiết: id của camera (trong URL path).
* Cập nhật: id của camera (trong URL path), và JSON payload chứa các trường cập nhật (ví dụ: cameraName, status).
* Xóa: id của camera (trong URL path).
* Yêu cầu JWT để xác thực người dùng.
* Kết quả:
* Thành công (Xem): Trả về danh sách/chi tiết đối tượng Camera dưới dạng JSON. Mã HTTP 200 OK.



*Hình 3.5: Thêm camera*

* Thành công (Cập nhật): Trả về đối tượng Camera đã cập nhật dưới dạng JSON. Mã HTTP 200 OK.
* Thành công (Xóa): Mã HTTP 204 No Content.
* Thất bại: Trả về thông báo lỗi (ví dụ: "Camera không tìm thấy", "Không có quyền") và mã lỗi HTTP (404 Not Found, 401 Unauthorized, 403 Forbidden).

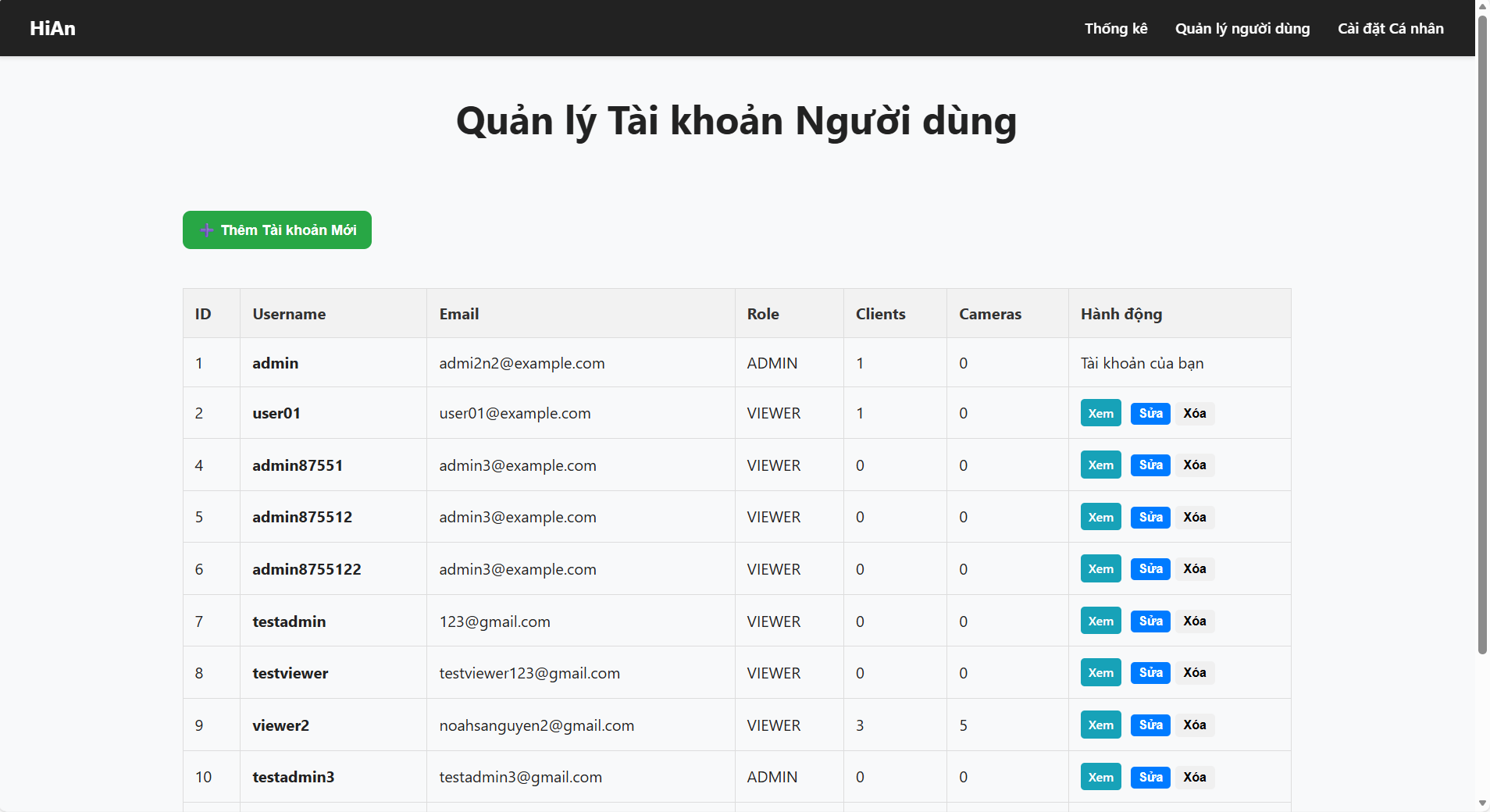


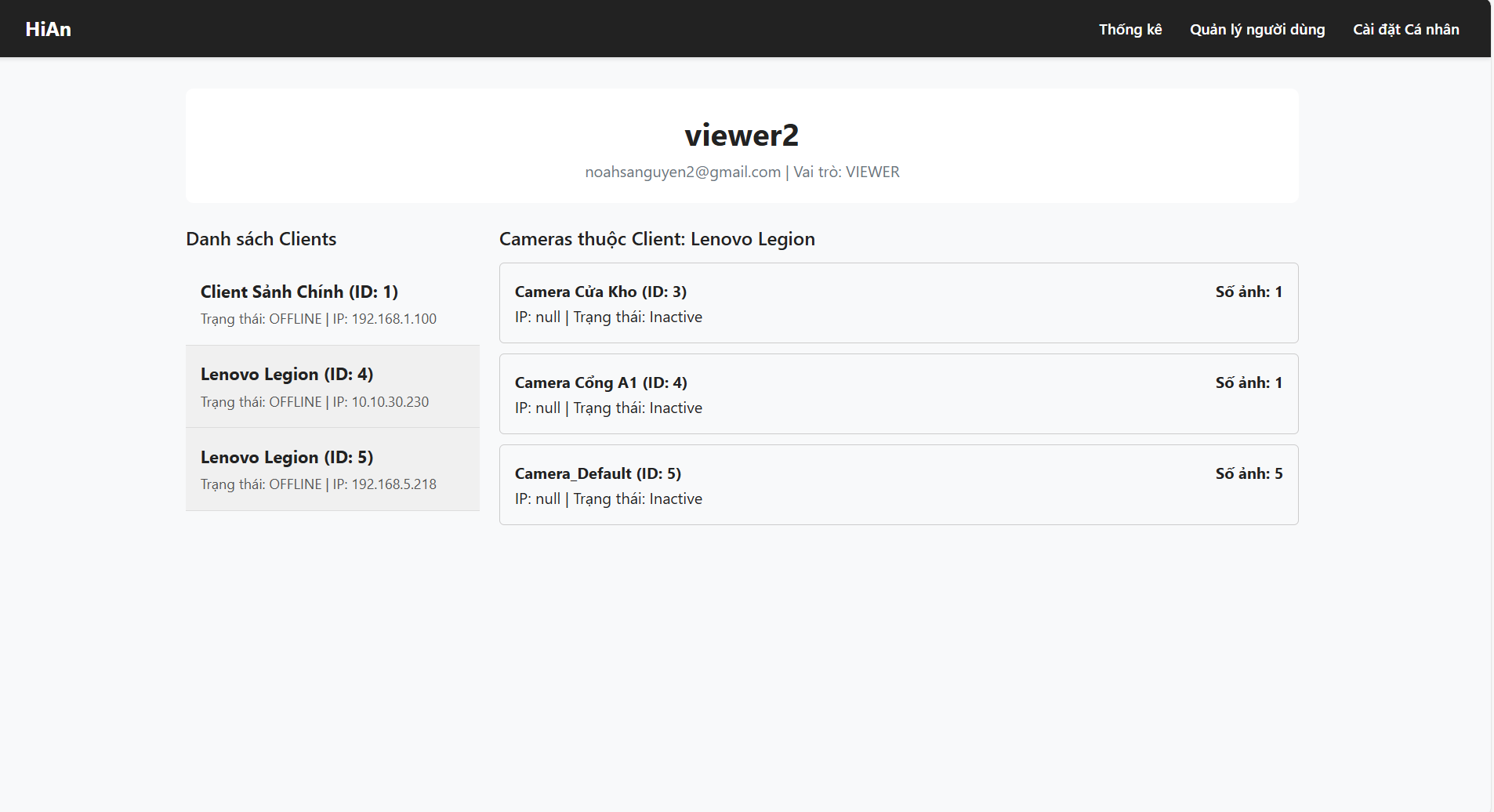
*Hình 3.6: Thêm camera thất bại*

* Nhận xét:
* Kiểm tra quyền: Server phải đảm bảo người dùng chỉ có thể quản lý các camera thuộc sở hữu của họ.
* Tích hợp với Client: Việc cập nhật trạng thái on/off của camera trên server có thể cần một cơ chế để thông báo cho Client Camera liên quan để thực hiện hành động tương ứng.

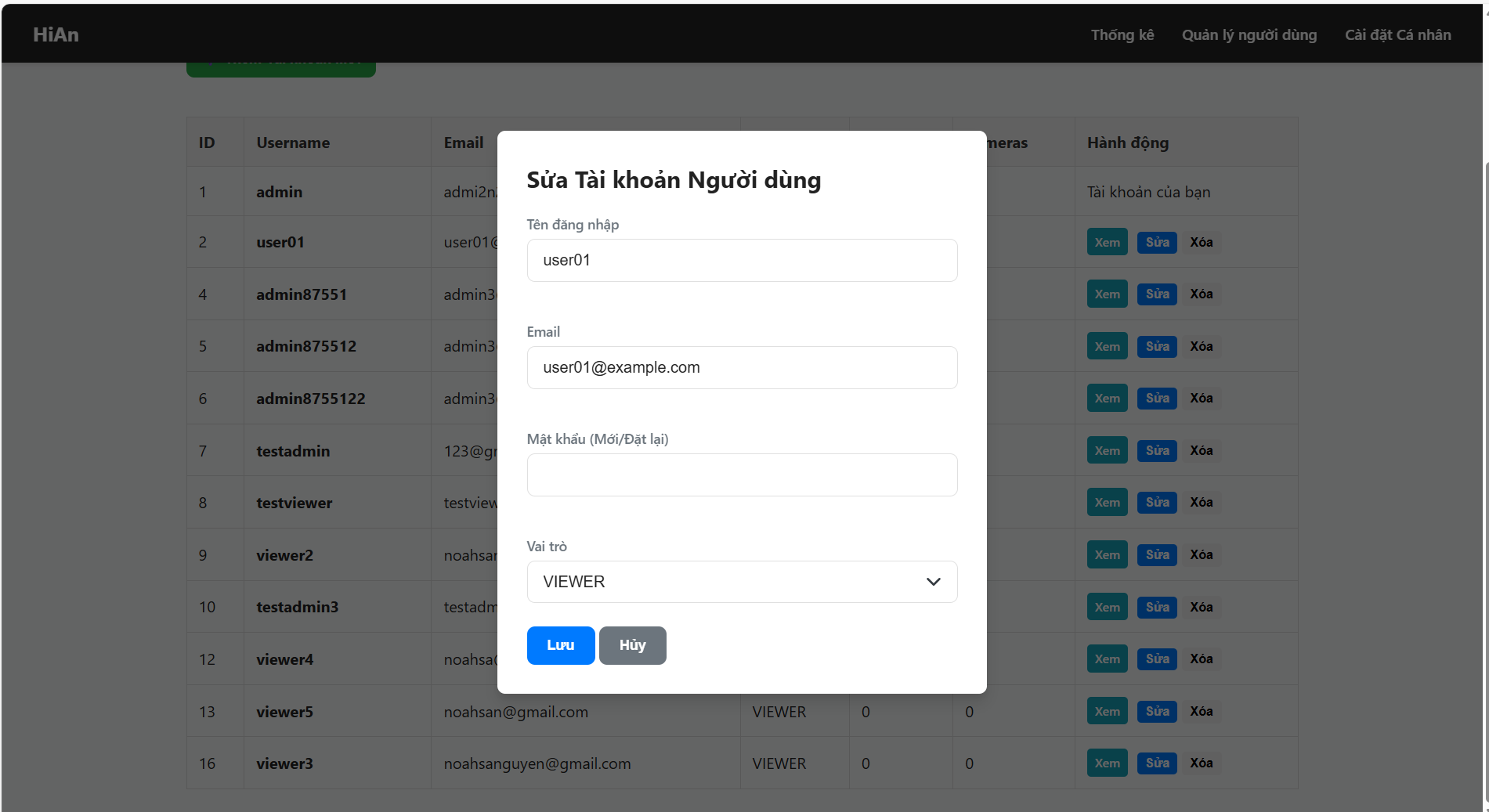
### Kịch bản 4: Chức năng: Quản lý người dùng (Manage Users - dành cho Admin)

* Mô tả: Quản trị viên có quyền tạo, xem, cập nhật và xóa các tài khoản người dùng trong hệ thống.
* Đầu vào:
* Xem danh sách: Không có đầu vào đặc biệt ngoài JWT của Admin.
* Tạo: JSON payload chứa thông tin người dùng mới (username, email, password, role).
* Xem chi tiết: id của người dùng (trong URL path).
* Cập nhật: id của người dùng (trong URL path), JSON payload chứa các trường cập nhật.
* Xóa: id của người dùng (trong URL path).
* Yêu cầu JWT của Admin để xác thực.
* Kết quả:
* Thành công: Trả về dữ liệu người dùng (khi tạo, xem, cập nhật) hoặc 204 No Content (khi xóa). Mã HTTP 200 OK / 201 Created / 204 No Content.

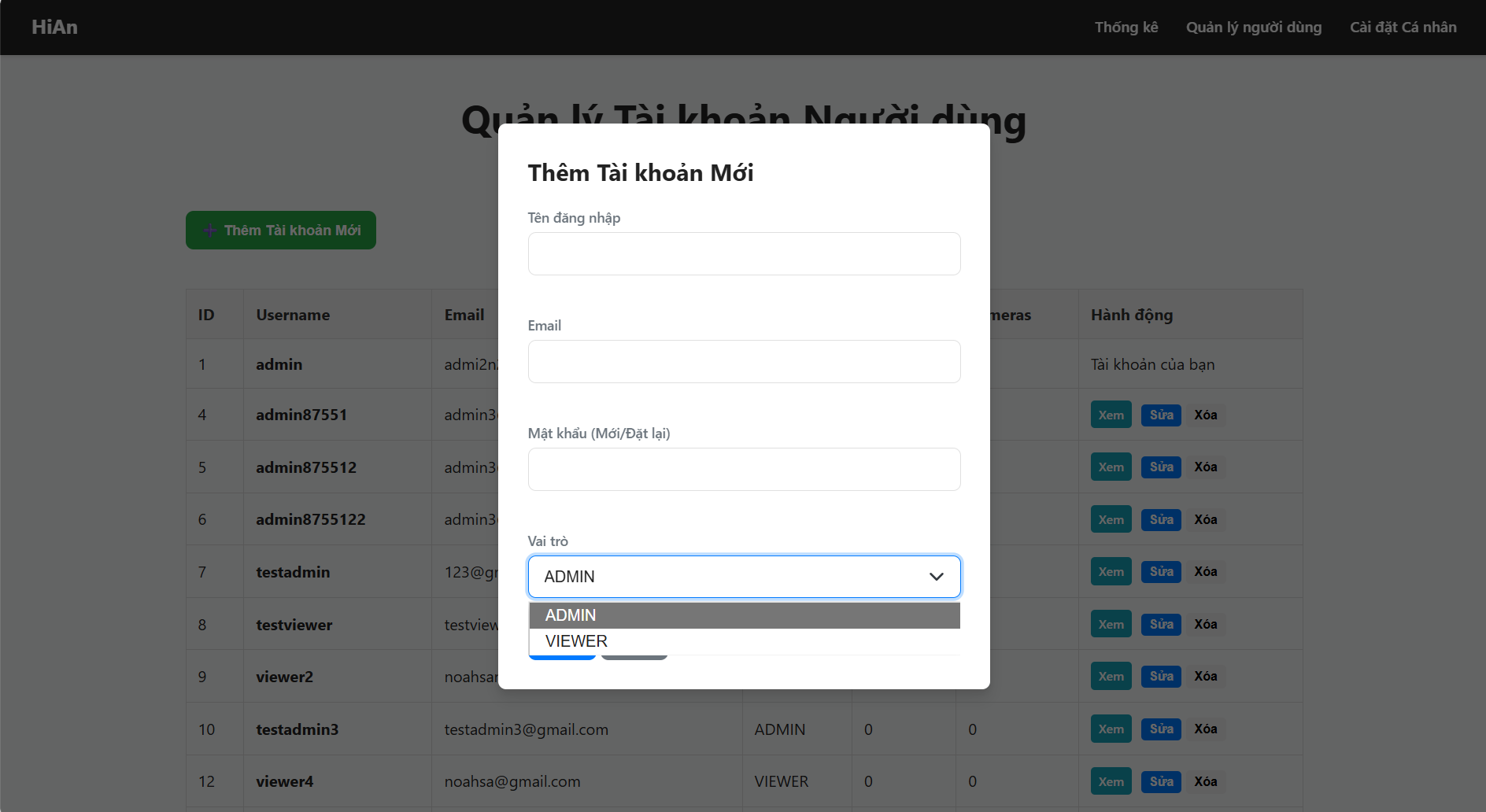
  
*Hình 3.7: Xem danh sách người dùng*

**

*Hình 3.8: Xem chi tiết người dùng*

**

*Hình 3.9: Chỉnh sửa thông tin người dùng*

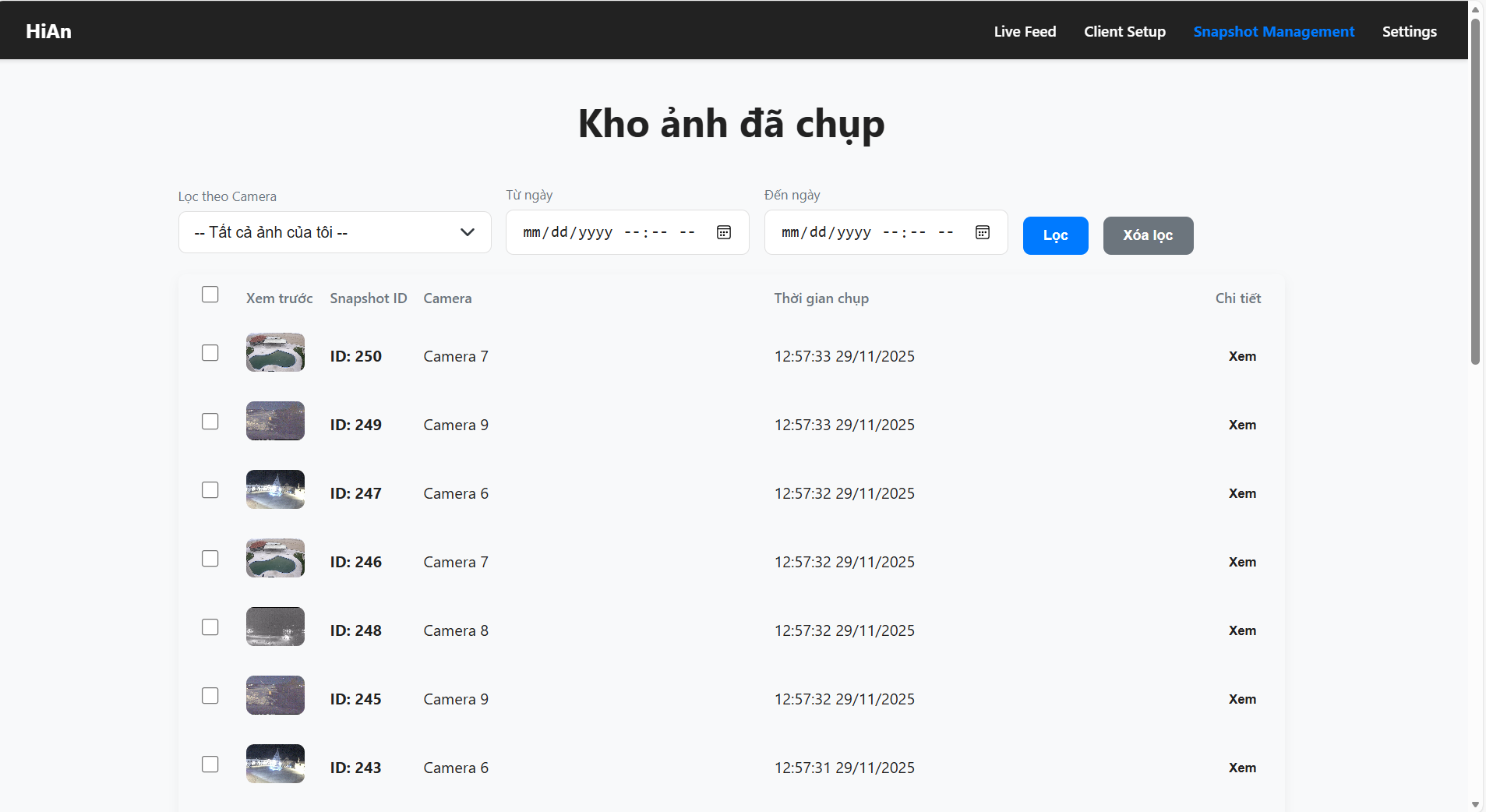
**

*Hình 3.10: Thêm tài khoản mới (lựa chọn quyền cho tài khoản đó*

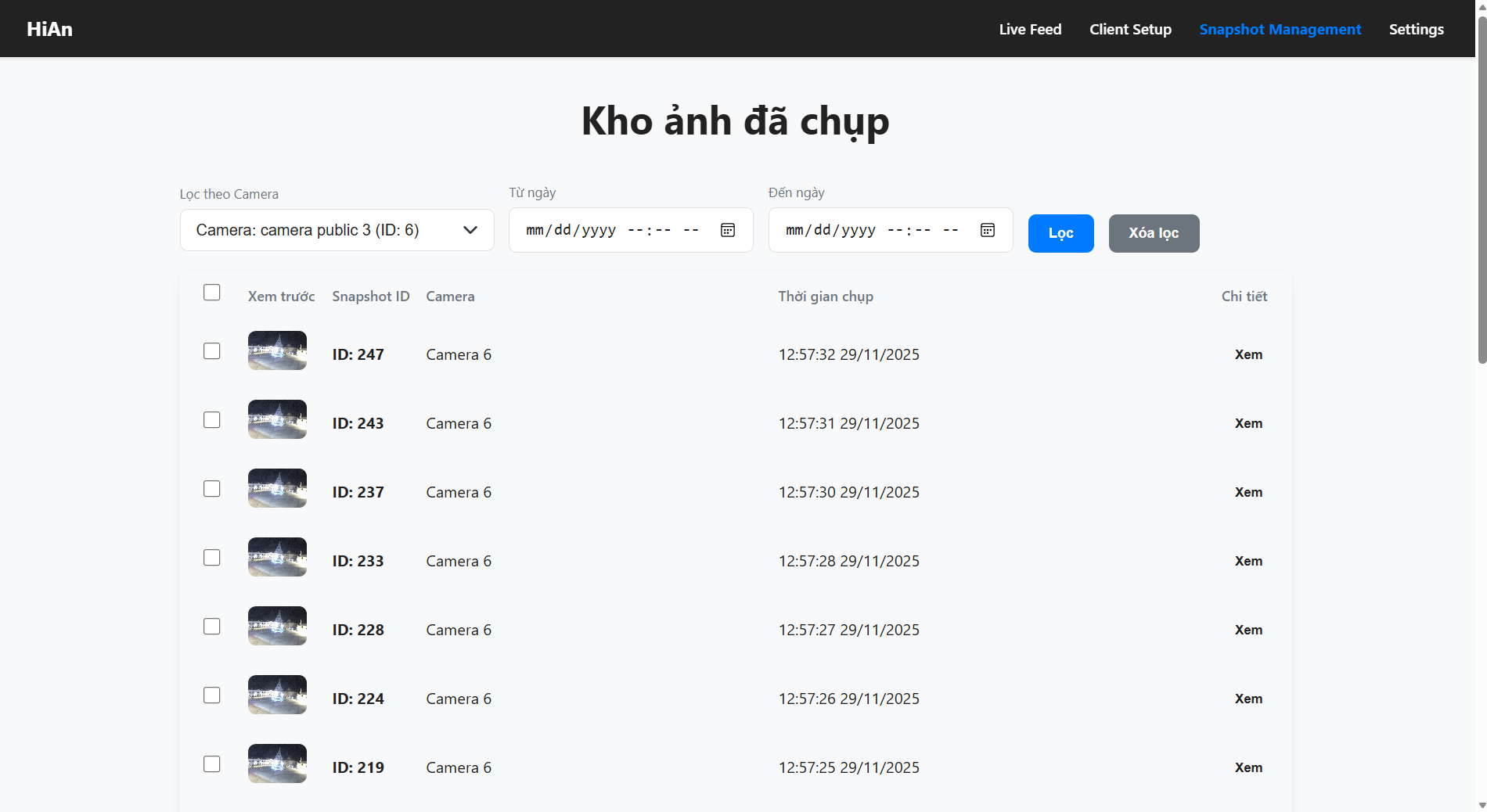
* Thất bại: Trả về thông báo lỗi (ví dụ: "Người dùng không tìm thấy", "Không có quyền Admin") và mã lỗi HTTP (404 Not Found, 403 Forbidden, 500 Internal Server Error).
* Nhận xét:
* Quyền hạn cao: Đây là các chức năng yêu cầu quyền hạn cao nhất (role ADMIN) và được bảo vệ nghiêm ngặt.
* Đảm bảo tính toàn vẹn: Khi xóa người dùng, cần đảm bảo tất cả các client và camera liên quan cũng được xử lý (ví dụ: chuyển giao quyền sở hữu, xóa bỏ).

### Kịch bản 5: Chức năng: Xem danh sách ảnh Snapshot (View Snapshot List)

* Mô tả: Người dùng truy cập để xem danh sách các ảnh chụp (snapshots) đã được lưu trữ trong hệ thống, có thể áp dụng các bộ lọc.
* Đầu vào:
* Yêu cầu JWT để xác thực người dùng.
* page (số nguyên, tùy chọn): Số trang để phân trang.
* size (số nguyên, tùy chọn): Số lượng ảnh trên mỗi trang.
* cameraId (số nguyên, tùy chọn): Lọc ảnh theo ID camera cụ thể.
* start (datetime, tùy chọn): Lọc ảnh từ thời điểm bắt đầu.
* end (datetime, tùy chọn): Lọc ảnh đến thời điểm kết thúc.
* Kết quả:
* Thành công: Trả về một đối tượng Page<Image> dưới dạng JSON, bao gồm danh sách ảnh, thông tin phân trang (tổng số trang, tổng số phần tử), và URL để xem từng ảnh. Mã HTTP 200 OK.



*Hình 3.11: Xem danh sách ảnh từ toàn bộ camera*

**

*Hình 3.12: Xem danh sách ảnh được lọc theo camera*

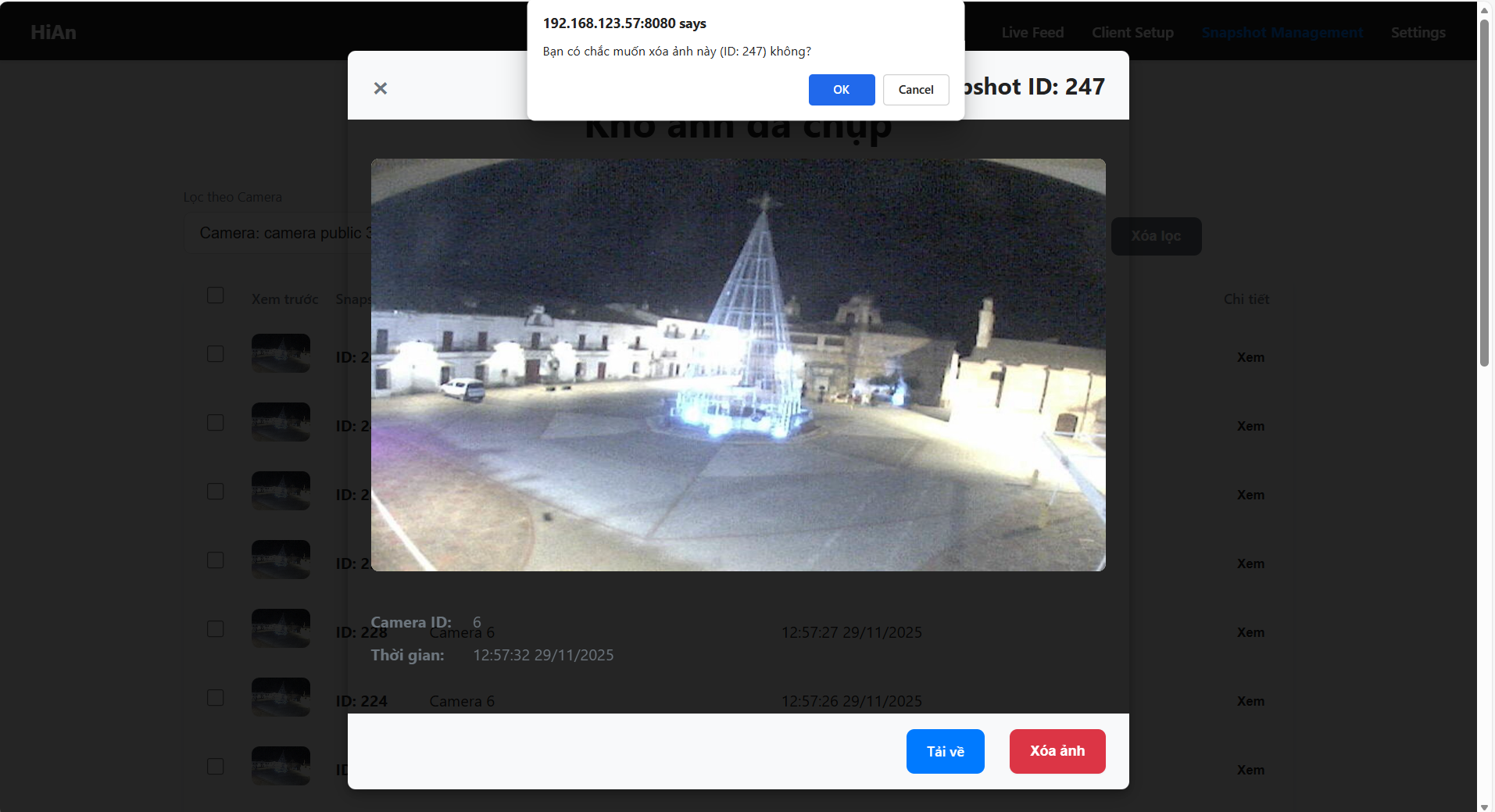
* Thất bại: Trả về thông báo lỗi (ví dụ: "Không có quyền", "Tham số không hợp lệ") và mã lỗi HTTP (401 Unauthorized, 400 Bad Request).
* Nhận xét:
* Phân trang & Lọc: Các tính năng này rất quan trọng để quản lý và duyệt qua một lượng lớn ảnh hiệu quả.
* Authorization: Server đảm bảo người dùng chỉ có thể xem ảnh của các camera mà họ có quyền truy cập.

### Kịch bản 6: Chức năng: Upload Snapshot (Upload Snapshot)

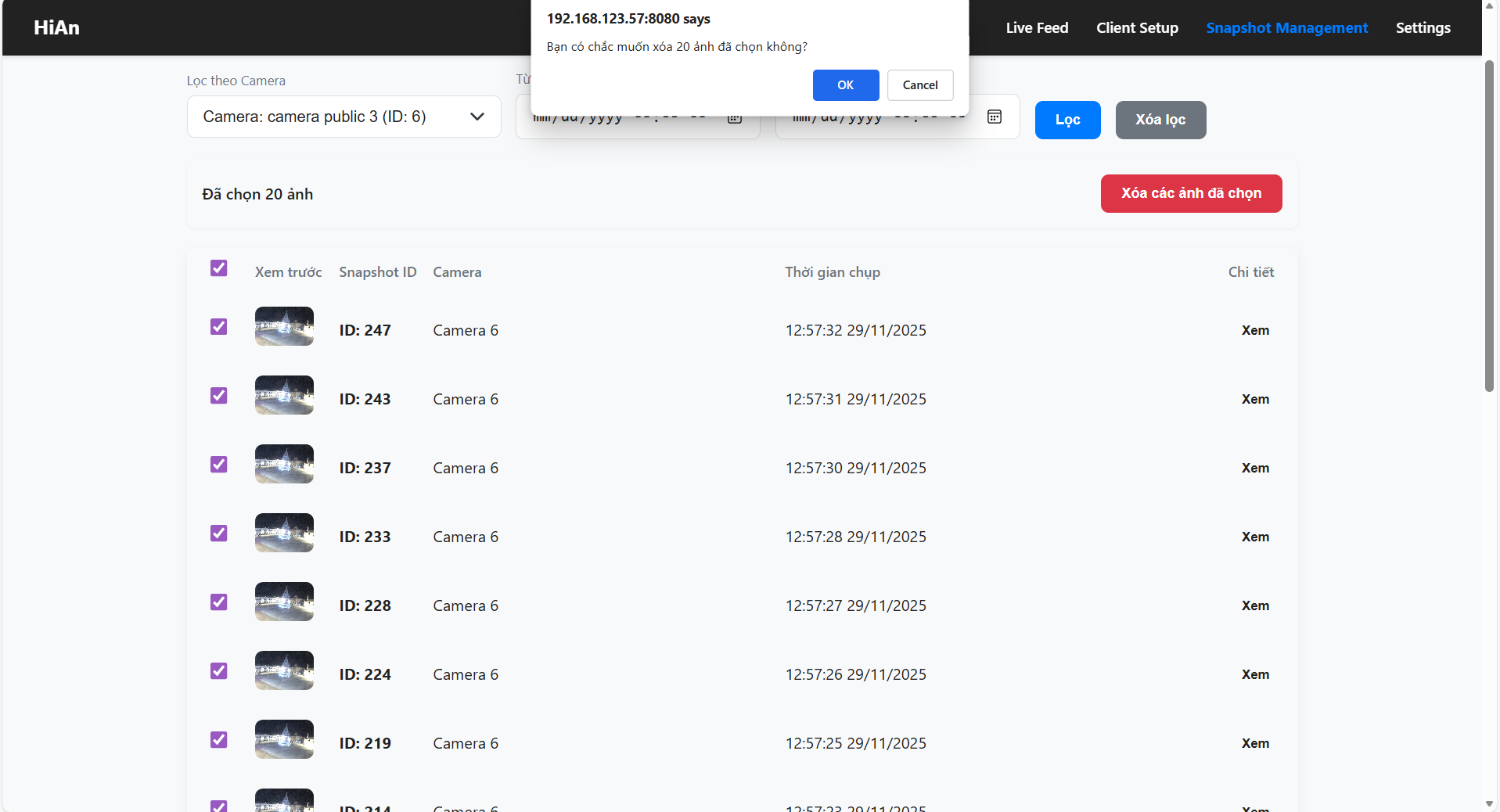
* Mô tả: Client (desktop hoặc web) tải lên một ảnh chụp tĩnh (snapshot) của camera lên server để lưu trữ vĩnh viễn.
* Đầu vào:
* file (multipart file): Tệp hình ảnh của snapshot.
* cameraId (số nguyên): ID của camera mà snapshot này thuộc về.
* capturedAt (timestamp): Thời điểm ảnh được chụp.
* Yêu cầu JWT để xác thực người dùng tải lên.
* Kết quả:
* Thành công: Server lưu tệp ảnh, lưu thông tin ảnh vào database và trả về đối tượng Image đã lưu (bao gồm URL để xem ảnh). Đồng thời, server gửi tin nhắn WebSocket (NEW\_SNAPSHOT) đến các web client của người dùng đã tải lên. Mã HTTP 201 Created.
* Thất bại: Trả về thông báo lỗi (ví dụ: "File không được rỗng", "Camera không tồn tại", "Lỗi upload") và mã lỗi HTTP (400 Bad Request, 404 Not Found, 500 Internal Server Error).
* Nhận xét:
* Lưu trữ vĩnh viễn: Không giống như live feed (chỉ là luồng tạm thời), snapshot được lưu trữ trên server.
* Thông báo thời gian thực: WebSockets được sử dụng để thông báo cho người dùng về các ảnh mới, giúp giao diện người dùng hiển thị tức thì mà không cần làm mới thủ công.
* URL động: Server tạo URL để truy cập ảnh đã tải lên, giúp client dễ dàng hiển thị.

### Kịch bản 7: Chức năng: Xóa ảnh Snapshot (Delete Snapshot)

* Mô tả: Người dùng yêu cầu xóa một hoặc nhiều ảnh chụp tĩnh đã lưu khỏi hệ thống.
* Đầu vào:
* Xóa một ảnh: id của ảnh (trong URL path).
* Xóa hàng loạt: JSON payload chứa danh sách các id của ảnh cần xóa (photoIds).
* Yêu cầu JWT để xác thực người dùng.
* Kết quả:
* Thành công (xóa một): Mã HTTP 204 No Content.



*Hình 3.13: Xóa một ảnh*

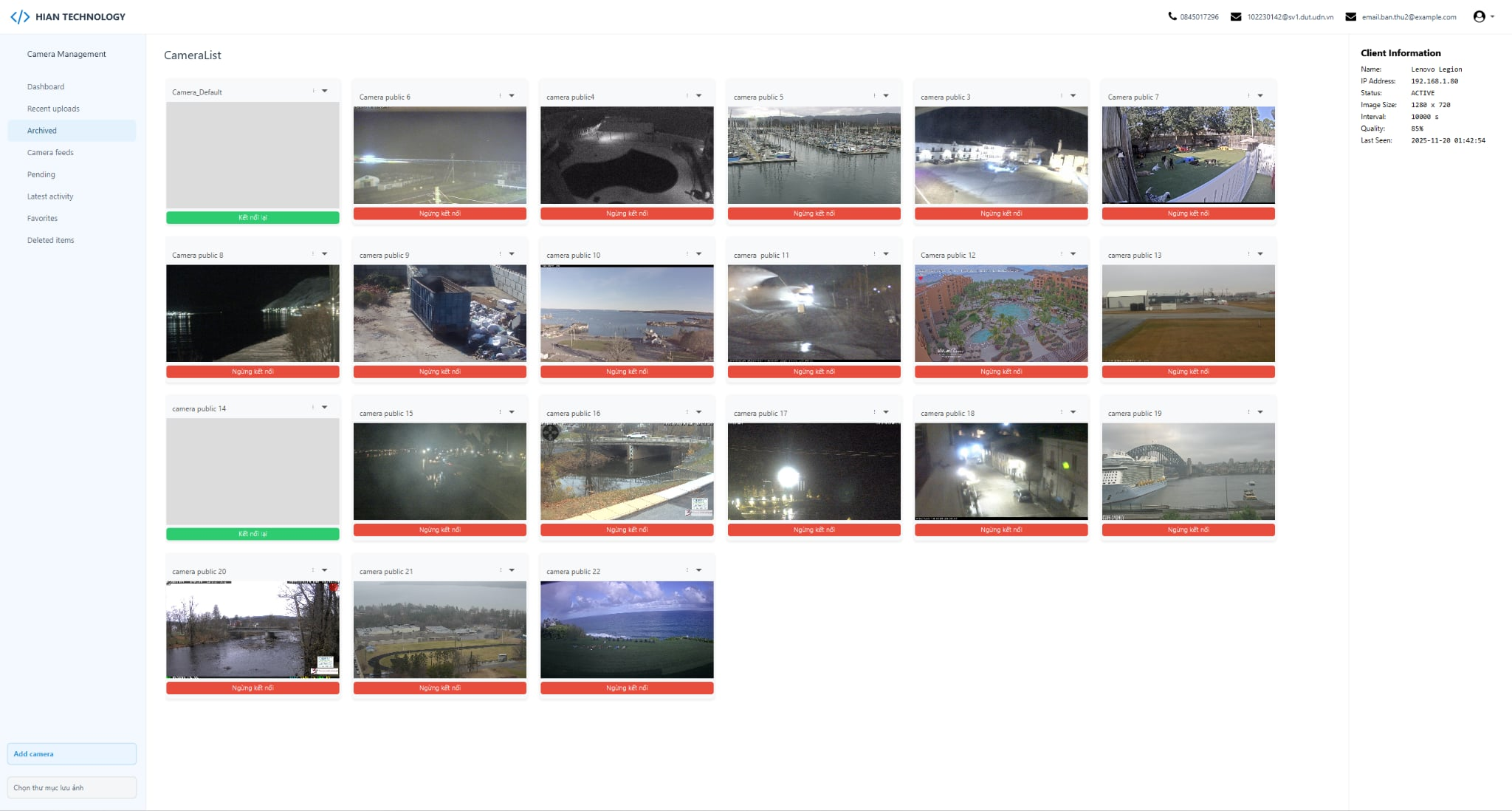
**

*Hình 3.14: Xóa nhiều ảnh*

* Thành công (xóa hàng loạt): Trả về JSON thông báo "Đã xóa thành công X ảnh." Mã HTTP 200 OK.
* Thất bại: Trả về thông báo lỗi (ví dụ: "Ảnh không tìm thấy", "Không có quyền xóa") và mã lỗi HTTP (404 Not Found, 403 Forbidden, 500 Internal Server Error).
* Nhận xét:
* Kiểm tra quyền: Server phải kiểm tra xem người dùng có quyền xóa ảnh đó hay không.
* Xóa tệp vật lý: Khi xóa ảnh khỏi database, server cũng cần xóa tệp ảnh vật lý khỏi hệ thống lưu trữ để giải phóng dung lượng.

### Kịch bản 8: Chức năng:Xem Live Camera (trên Ứng dụng Client Desktop)

* Mô tả: Ứng dụng camera-client (desktop) có khả năng trực tiếp truy cập và hiển thị luồng video trực tiếp từ một camera cục bộ hoặc một camera trên mạng nội bộ mà nó có quyền truy cập, mà không cần thông qua backend server để phân phối luồng.
* Đầu vào:
* cameraId (số nguyên): ID của camera muốn xem live feed. (Sẽ cần cơ chế để client xác định camera này, có thể từ danh sách camera được quản lý bởi server).
* Yêu cầu JWT (từ ứng dụng client) để xác thực người dùng và quyền truy cập camera nếu cần.
* Kết quả:
* Thành công: Ứng dụng desktop hiển thị luồng video trực tiếp từ camera.

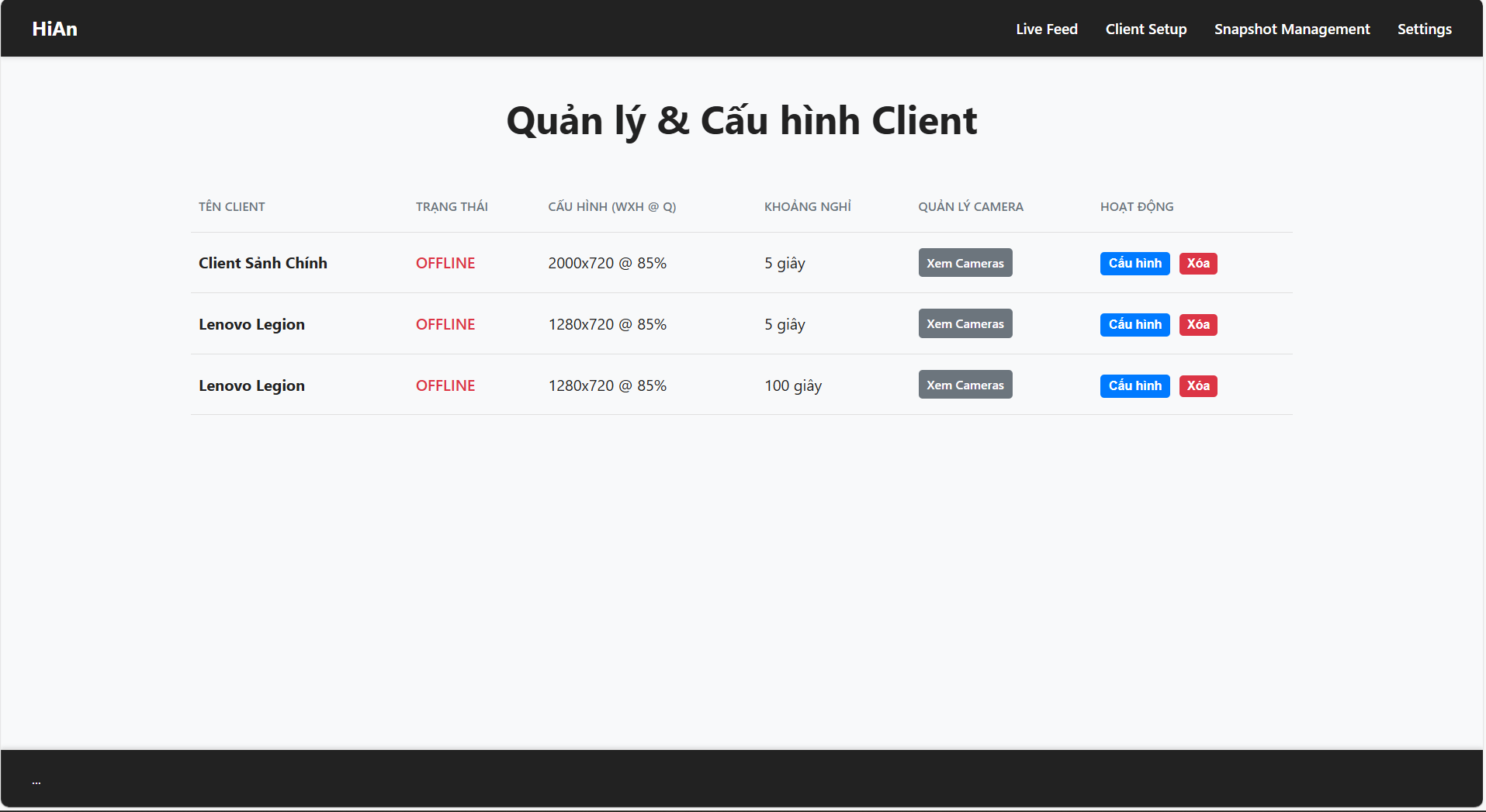


*Hình 3.15: Danh sách camera đang kết nối trực tiếp*

* Thất bại: Ứng dụng không thể kết nối đến camera, không có quyền truy cập, hoặc lỗi xử lý video cục bộ. Thông báo lỗi hiển thị trên giao diện người dùng của ứng dụng desktop.
* Nhận xét:
* Giao tiếp trực tiếp: Không giống như luồng live feed hiện tại (client desktop -> server -> web client), chức năng này ngụ ý một kênh giao tiếp trực tiếp hơn giữa ứng dụng desktop và camera. Điều này có thể bao gồm:
* Truy cập Camera cục bộ: Nếu camera được kết nối trực tiếp với máy tính chạy ứng dụng client.
* Truy cập Camera qua mạng LAN: Nếu camera có địa chỉ IP cục bộ và ứng dụng client có thể kết nối trực tiếp qua các giao thức như RTSP (Real-Time Streaming Protocol), MJPEG, hoặc các API camera chuyên biệt.
* Không qua Server: Ưu điểm là giảm tải cho server và độ trễ thấp hơn, nhưng yêu cầu ứng dụng client phải có khả năng xử lý luồng video trực tiếp và quản lý kết nối với camera.
* Phân quyền: Vẫn cần cơ chế xác thực và ủy quyền để đảm bảo người dùng chỉ xem được các camera mà họ có quyền. Điều này có thể được thực hiện bằng cách client desktop gọi API của server để lấy danh sách camera được phép xem và thông tin kết nối (địa chỉ IP, cổng, thông tin đăng nhập camera).
* Xử lý video: Ứng dụng desktop sẽ cần các thư viện xử lý video (ví dụ: OpenCV, JavaCV, FFmpeg) để thu nhận, giải mã và hiển thị luồng video.

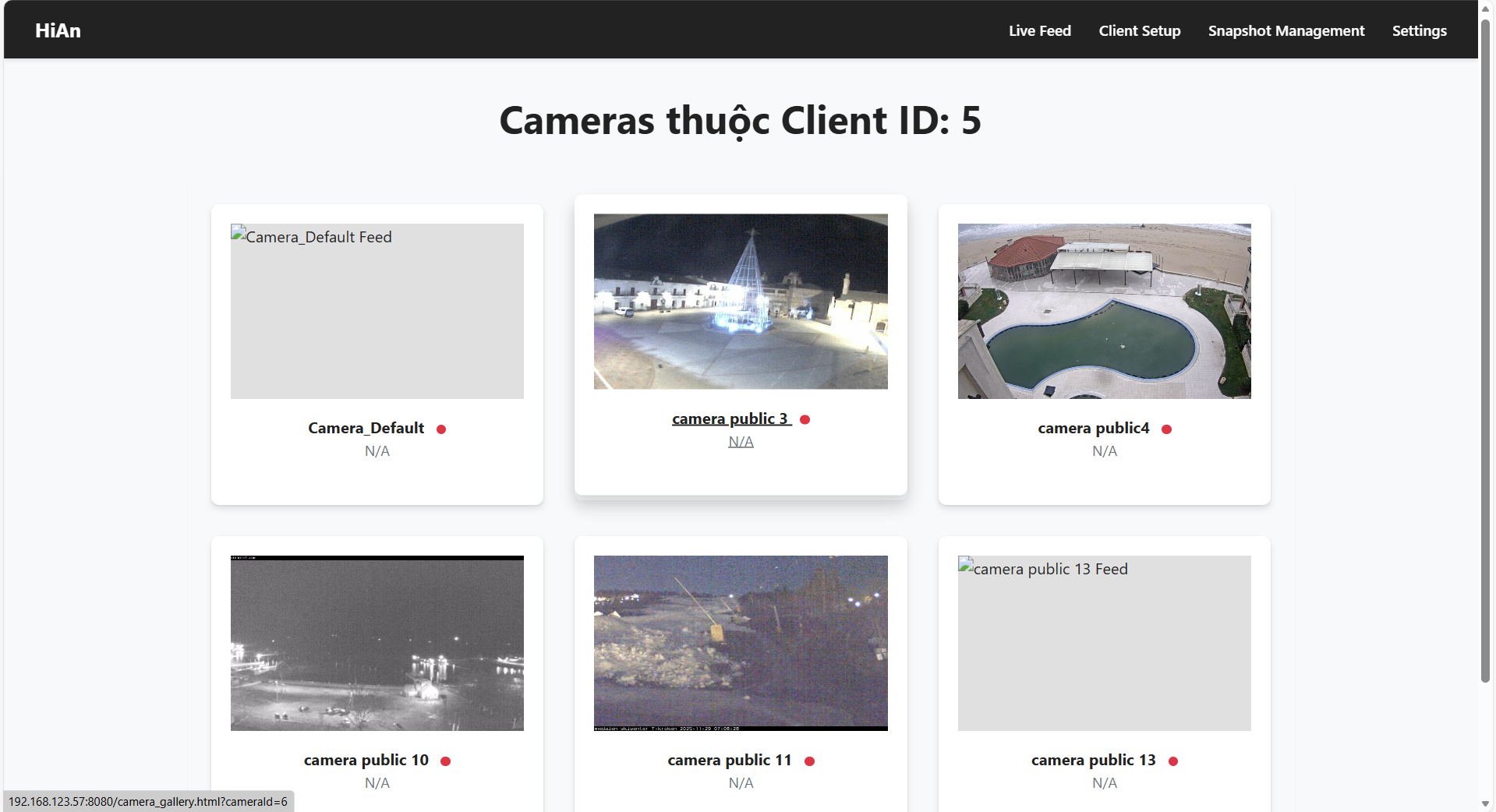
### Kịch bản 9: Chức năng: Quản lý Client của tôi (Manage My Clients)

* Mô tả: Người dùng có thể xem danh sách các thiết bị client mà họ sở hữu, xem chi tiết từng client, cập nhật thông tin client (ví dụ: tên client), và xóa client.
* Đầu vào:
* Xem danh sách: Không có đầu vào đặc biệt ngoài JWT của người dùng.
* Xem chi tiết: id của client (trong URL path).
* Cập nhật: id của client (trong URL path), và JSON payload chứa các trường cập nhật (ví dụ: clientName).
* Xóa: id của client (trong URL path).
* Kết quả:
* Thành công (Xem danh sách): Trả về danh sách các đối tượng Client dưới dạng JSON.



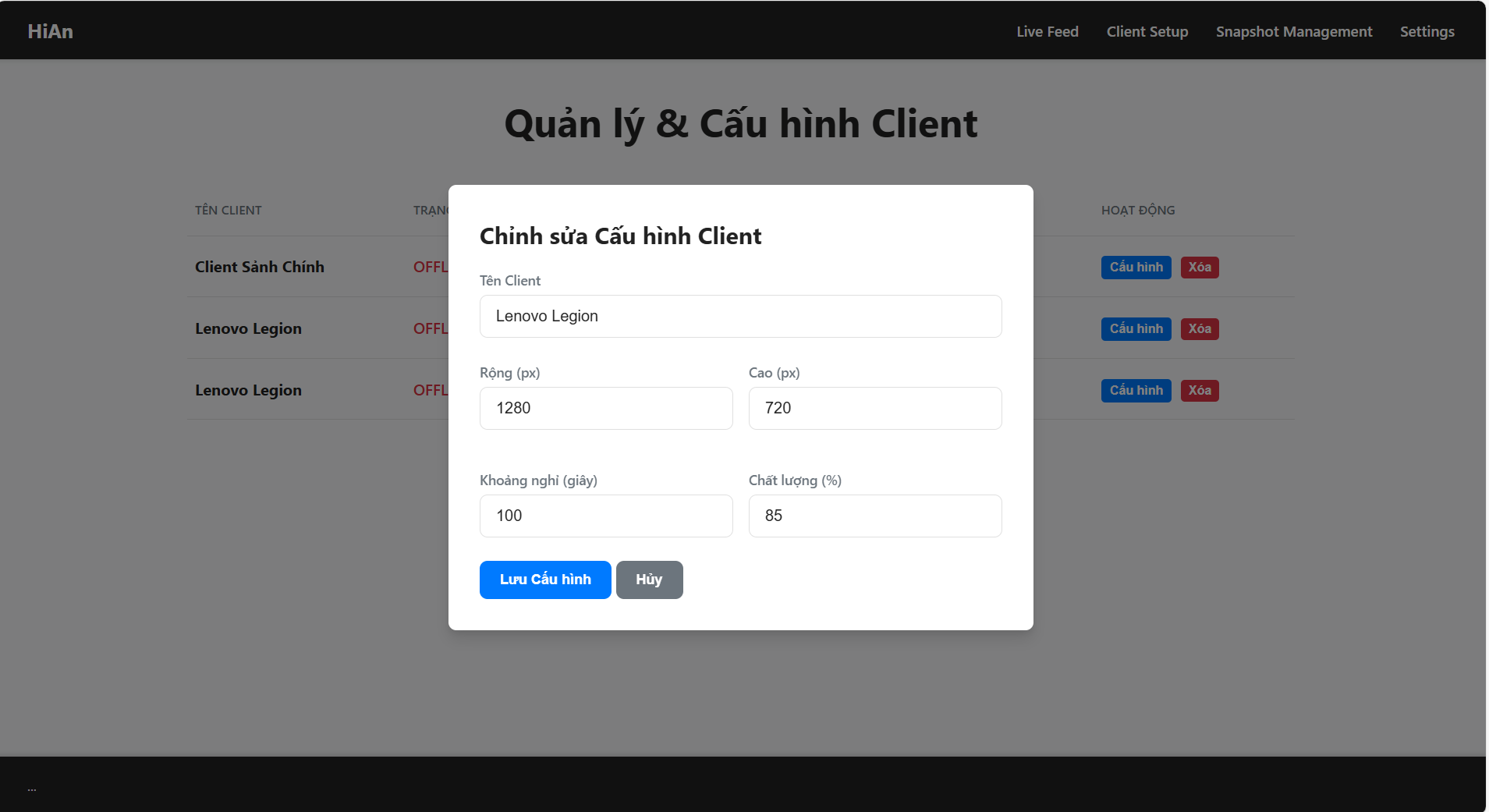
*Hình 3.16: Danh sách client của người dùng hiện tại*

* Thành công (Xem chi tiết): Trả về đối tượng Client dưới dạng JSON.



*Hình 3.17: Danh sách camera của client*

* Thành công (Cập nhật): Trả về đối tượng Client đã cập nhật dưới dạng JSON. Đặc biệt: Server cũng sẽ gửi một tin nhắn WebSocket (CONFIG\_UPDATE) đến các web client của người dùng sở hữu client đó.

**

*Hình 3.18: Xem cấu hình hiện tại của client và chỉnh sửa trực tiếp*

* + Thành công (Xóa): Trả về trạng thái HTTP 204 No Content.
* Thất bại: Trả về thông báo lỗi (ví dụ: "Client không tìm thấy", "Không có quyền") và mã lỗi HTTP (404 Not Found, 401 Unauthorized, 403 Forbidden).
* Nhận xét:
* Yêu cầu xác thực: Tất cả các thao tác này đều yêu cầu người dùng đã đăng nhập (JWT hợp lệ) và phải có quyền sở hữu hoặc quyền quản trị đối với client đang được thao tác.
* Cập nhật thời gian thực: Việc sử dụng WebSocket để thông báo cập nhật client (CONFIG\_UPDATE) là một tính năng thông minh, giúp các giao diện người dùng tự động phản ánh các thay đổi mà không cần làm mới trang.
* Model chung: Việc sử dụng pbl4.common.model.Client đảm bảo tính nhất quán của cấu trúc dữ liệu client giữa client và server.

## NHẬN XÉT ĐÁNH GIÁ KẾT QUẢ

Sau quá trình triển khai thực nghiệm và kiểm thử hệ thống , kết quả thu được được phân tích và đánh giá chi tiết qua các ưu điểm, hạn chế và hiệu năng tổng thể như sau:

### Những ưu điểm đạt được

Hệ thống đã giải quyết tốt các yêu cầu phi chức năng về tính ổn định và trải nghiệm người dùng hiện đại, cụ thể:

* Khả năng kết nối 1 lúc nhiều camera : được dùng các thuật toán để tối ưu thời gian , giúp người dùng ít phải chờ đợi , tận dụng tối đa tài nguyên hệ thống cho phép
* Khả năng tự động hóa và phát hiện thiết bị camera : giúp người dùng quét và tìm kiếm các camera đang hoạt động trong mạng LAN
* Trải ngiệm người dùng tối ưu, mượt mà , hiện đại: giao diện đẹp ,tách biệt hoàn toàn luồng xử lý và luồng giao diện
* Khả năng lưu trữ và quản lý nhiều ảnh, nhiều camera và client: Dùng thuật toán để tối ưu thời gian, tối ưu câu lệnh truy vấn, giảm thiểu thời gian chờ cho người dung
* Chức năng tự động cập nhật: cập nhật liên tục theo thời gian thực, giảm thiểu tối đa độ trễ, quản lý và cập nhật liên tục client.

### Các hạn chế và tồn tại

* Phụ thuộc lớn vào tài nguyên phần cứng (CPU / RAM)
* Chưa tối ưu hóa tăng tốc phần cứng
* Giới hạn về bằng thông mạng
* Khi lượng người dung tang thì sẽ dẫn tới tình trạng tắc nghẽn do quản lý websocket tốn nhiều tài nguyên hơn

### Tổng hợp đánh giá hiệu năng

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Kết quả đo đạc / Quan sát | Đánh giá |
| Thời gian khởi động AppClient | < 5 giây (Bao gồm kết nối Server và kết nối 1 Camera mạng LAN)  < 30 giây(Bao gồm kết nối Server và kết nối 1 Camera mạng WAN)  > 60 giây(Bao gồm kết nối Server và kết nối nhiều Camera mạng WAN | Tốt |
| Độ trễ tín hiệu điều khiển | < 500ms (Từ lúc bấm nút trên Web đến lúc Client nhận lệnh) | Xuất sắc (Nhờ WebSocket) |
| Tỉ lệ gửi ảnh thành công | > 98% (Trong điều kiện mạng LAN ổn định) | Tốt |
| Tài nguyên tiêu thụ (Server) | RAM: ~400-600MB (Khi chạy Spring Boot + MySQL) | Trung bình |
| Tài nguyên tiêu thụ (Client) | CPU: 15-20% (Khi xử lý 1 luồng HD 720p) | Tốt |
| Giới hạn camera kết nối cùng lúc | 20 Camera mạng WAN (Bị giới hạn đường truyền mạng) | Tốt |

## Kết chương

Chương 3 đã trình bày chi tiết quy trình triển khai hệ thống trên môi trường thực tế, từ việc cấu hình phần cứng, cài đặt phần mềm đến việc thực hiện kiểm thử qua các kịch bản chức năng cụ thể. Kết quả thực nghiệm cho thấy hệ thống hoạt động đúng theo thiết kế đề ra trong Chương 2, đáp ứng được các yêu cầu cơ bản về giám sát và quản lý tập trung, mặc dù vẫn còn một số hạn chế nhỏ về độ trễ và khả năng phân tích sâu cần được cải thiện trong tương lai.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 1. KẾT QUẢ ĐẠT ĐƯỢC

Sau quá trình nghiên cứu và triển khai đề tài **"Xây dựng ứng dụng truy xuất và phân tích dữ liệu hình ảnh từ Camera qua mạng"**, nhóm thực hiện đã hoàn thành các mục tiêu đề ra ban đầu, đạt được những kết quả cụ thể cả về mặt lý thuyết và thực tiễn:

**1.1. Về mặt lý thuyết**

* **Nắm vững kiến trúc Client-Server:** Hiểu và vận dụng thành công mô hình kiến trúc phân tán, tách biệt rõ ràng giữa lớp thu thập dữ liệu (AppClient), lớp xử lý trung tâm (Server) và lớp trình bày (WebClient).
* **Làm chủ các giao thức truyền thông:** Đã nghiên cứu và ứng dụng hiệu quả các giao thức mạng quan trọng như **RTSP** (lấy luồng video), **HTTP/RESTful** (giao tiếp API), **WebSocket** (giao tiếp thời gian thực) và **TCP** (truyền tải tin cậy).
* **Hiểu sâu về xử lý đa luồng (Multithreading):** Áp dụng thành công kỹ thuật lập trình đa luồng trong Java để xử lý song song việc thu nhận hình ảnh và gửi dữ liệu mạng mà không làm treo giao diện ứng dụng.
* **Bảo mật hệ thống:** Nắm được quy trình xác thực và phân quyền không trạng thái (Stateless) sử dụng chuẩn **JWT (JSON Web Token)**.

**1.2. Về mặt thực tiễn ứng dụng**

Nhóm đã xây dựng thành công một hệ thống giám sát hoàn chỉnh với các chức năng:

* **Xây dựng AppClient (Desktop):** Ứng dụng chạy ổn định trên máy trạm, tự động kết nối Camera IP qua chuẩn ONVIF/RTSP, thực hiện trích xuất hình ảnh (snapshot) và đồng bộ về Server.
* **Xây dựng Backend Server:** Hệ thống máy chủ mạnh mẽ sử dụng **Spring Boot**, cung cấp đầy đủ các API quản lý người dùng, thiết bị và lưu trữ dữ liệu tập trung.
* **Xây dựng WebClient (Dashboard):** Giao diện quản trị trực quan, cho phép người dùng xem danh sách Camera, xem lại lịch sử hình ảnh với bộ lọc thời gian và nhận thông báo cảnh báo tức thì qua WebSocket.
* **Triển khai thành công trên mạng LAN:** Hệ thống đã được kiểm thử và hoạt động ổn định trong môi trường mạng nội bộ với độ trễ thấp và độ tin cậy cao.

## 2. KIẾN NGHỊ VÀ HƯỚNG PHÁT TRIỂN

Để hoàn thiện và nâng cao tính ứng dụng của đề tài trong tương lai, nhóm đề xuất các hướng phát triển sau:

**3.1. Nâng cấp khả năng "Phân tích dữ liệu" (Trọng tâm)**

* **Tích hợp AI/Computer Vision:** Nhúng các model Deep Learning (như YOLO, SSD) vào AppClient hoặc Server để thực hiện:
* Phát hiện xâm nhập (Human Detection).
* Nhận diện biển số xe hoặc khuôn mặt.
* Chỉ gửi cảnh báo và hình ảnh khi phát hiện sự kiện bất thường để tiết kiệm băng thông.

**3.2. Tối ưu hóa kiến trúc và Hiệu năng**

* **Chuyển đổi giao thức Streaming:** Nghiên cứu sử dụng WebRTC hoặc HLS (HTTP Live Streaming) để truyền tải video mượt mà hơn với độ trễ thấp hơn thay vì gửi ảnh snapshot liên tục.
* **Lưu trữ đám mây (Cloud Storage):** Tích hợp các dịch vụ lưu trữ đối tượng (Object Storage) như AWS S3 hoặc MinIO để giải quyết bài toán lưu trữ lớn và sao lưu dữ liệu an toàn.

**3.3. Mở rộng nền tảng**

* **Phát triển Mobile App:** Xây dựng ứng dụng trên Android/iOS để người dùng có thể nhận thông báo cảnh báo (Push Notification) và xem camera mọi lúc mọi nơi.
* **Bảo mật nâng cao:** Triển khai giao thức HTTPS (SSL/TLS) cho toàn bộ hệ thống để mã hóa dữ liệu truyền tải trên môi trường Internet công cộng.

**TÀI LIỆU THAM KHẢO**

**Tài liệu Tiếng Việt (Giáo trình ĐH)**

*[1]* ***Nguyễn Thúc Hải (2009)****, Mạng máy tính và các hệ thống mở, Nhà xuất bản Giáo dục. (Dùng cho chương lý thuyết mạng, mô hình OSI/TCP-IP).*

*[2]* ***Lương Chi Mai (2007)****, Nhập môn xử lý ảnh số, Nhà xuất bản Khoa học và Kỹ thuật. (Dùng cho phần lý thuyết về điểm ảnh, nén ảnh JPEG/MJPEG).*

*[3]* ***Phạm Hữu Khang (2012)****, Xây dựng ứng dụng với Spring Framework, Nhà xuất bản Phương Đông. (Dùng tham khảo kiến trúc Spring).*

#### **Tài liệu trực tuyến (Documentation)**

*[4]* ***JavaCV Team****, JavaCV - Java interface to OpenCV, FFmpeg, and more. [Online]. Available:* [*https://github.com/bytedeco/javacv*](https://github.com/bytedeco/javacv)*.*

*[5]* ***Spring IO****, Spring Boot Reference Documentation. [Online]. Available:* [*https://docs.spring.io/spring-boot/docs/current/reference/html/*](https://www.google.com/search?q=https://docs.spring.io/spring-boot/docs/current/reference/html/)*.*

*[6]* ***FFmpeg****, FFmpeg Documentation. [Online]. Available:* [*https://ffmpeg.org/documentation.html*](https://ffmpeg.org/documentation.html)*.*

**Các tiêu chuẩn kỹ thuật (Standards & RFCs)**

*[7]* ***IETF RFC 2326 (1998)****, Real Time Streaming Protocol (RTSP). Available:* [*https://www.ietf.org/rfc/rfc2326.txt*](https://www.ietf.org/rfc/rfc2326.txt)*. (Chuẩn giao thức RTSP em dùng để kết nối Camera).*

*[8]* ***IETF RFC 7519 (2015)****, JSON Web Token (JWT). Available:* [*https://tools.ietf.org/html/rfc7519*](https://tools.ietf.org/html/rfc7519)*. (Chuẩn bảo mật em dùng để đăng nhập).*

*[9]* ***ONVIF Core Specification (2020)****, Open Network Video Interface Forum. Available:* [*https://www.onvif.org/profiles/*](https://www.onvif.org/profiles/)*. (Chuẩn giao tiếp Camera IP).*

**PHỤ LỤC**

Gihub: https://github.com/Noahsa-HiNg/PBL4.git