**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**



**ĐỒ ÁN**

**TỐT NGHIỆP ĐẠI HỌC**

**Đề tài:**

**THIẾT KẾ VÀ THỰC HIỆN GIAO DIỆN (FRONTEND) CHO NỀN TẢNG GIÁM SÁT THÀNH PHỐ THÔNG MINH**

Giảng viên hướng dẫn: PGS. GVCC Nguyễn Tài Hưng

Sinh viên thực hiện: Vũ Đức Tuấn

Lớp: Điện Tử 05 – K63

***Hà Nội, tháng 01 năm 2025***

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**



**ĐỒ ÁN**

**TỐT NGHIỆP ĐẠI HỌC**

**Đề tài:**

**THIẾT KẾ VÀ THỰC HIỆN GIAO DIỆN (FRONTEND) CHO NỀN TẢNG GIÁM SÁT THÀNH PHỐ THÔNG MINH**

Giảng viên hướng dẫn: PGS. GVCC Nguyễn Tài Hưng

Sinh viên thực hiện: Vũ Đức Tuấn

Lớp: Điện Tử 05 – K63

***Hà Nội, tháng 01 năm 2025***

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**

**(DÀNH CHO CÁN BỘ HƯỚNG DẪN)**

Tên đề tài: Thiết kế và thực hiện giao diện (frontend) cho nền tảng giám sát thành phố thông minh qua hình ảnh

Họ tên SV: Vũ Đức Tuấn MSSV: 20182869

Cán bộ hướng dẫn: PGS. GVCC Nguyễn Tài Hưng

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tiêu chí**  (Điểm tối đa) | **Hướng dẫn đánh giá tiêu chí** | **Điểm tiêu chí** |
| 1 | **Thái độ làm việc  (2,5 điểm)** | Nghiêm túc, tích cực và chủ động trong quá trình làm ĐATN |  |
| Hoàn thành đầy đủ và đúng tiến độ các nội dung được GVHD giao |
| 2 | **Kỹ năng viết quyển ĐATN (2 điểm)** | Trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v. |  |
| Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v. |  |
| 3 | **Nội dung và kết quả đạt được  (5 điểm)** | Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận. |  |
| Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan. |
| Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án. |
| 4 | **Điểm thành tích  (1 điểm)** | Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. **(1 điểm)** |  |
| Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. **(0,5 điểm)** |
|  | | **Điểm tổng các tiêu chí:** |  |
| **Điểm hướng dẫn:** |  |

***Nhận xét khác của cán bộ hướng dẫn***

|  |  |
| --- | --- |
|  | Ngày: … / … / 20…  **Cán bộ hướng dẫn**  (Ký và ghi rõ họ tên) |

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**

**(DÀNH CHO CÁN BỘ THÀNH VIÊN HỘI ĐỒNG)**

Hội đồng số: ……………………………………………………………………………………

Họ tên SV: Vũ Đức Tuấn MSSV: 20182869

Cán bộ thành viên HĐ: ……………………………………………………………………..…..

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tiêu chí** (Điểm tối đa) | **Hướng dẫn đánh giá tiêu chí** | **Điểm tiêu chí** |
| 1 | **Chất lượng slides/Bản vẽ kỹ thuật** (1,5 điểm) | Sử dụng các minh họa hỗ trợ: Hình ảnh, biểu đồ rõ nét và phù hợp, dễ hiểu |  |
| Không quá nhiều từ, biết sử dụng từ khoá; bố cục logic, có đánh số trang |
| 2 | **Kỹ năng thuyết trình** (1,5 điểm) | Tự tin, làm chủ nội dung trình bày, đúng thời gian quy định |  |
| Dễ hiểu, dễ theo dõi, lô-gic, lôi cuốn. |
| 3 | **Nội dung và kết quả đạt được** (4 điểm) | Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận. |  |
| Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan. |
| Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/ tính sáng tạo trong nội dung và kết quả đồ án. |
| 4 | **Trả lời câu hỏi** (2,5 điểm) | Trả lời ngắn gọn, chính xác, đi thẳng vào vấn đề của câu hỏi. |  |
| Nắm vững kiến thức cơ bản liên quan đến lĩnh vực nghiên cứu/ công việc của đồ án. |
| 5 | **Điểm thành tích** (1 điểm) | Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. **(1 điểm)** |  |
| Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. **(0,5 điểm)** |
|  |  | **Điểm tổng các tiêu chí:** |  |
|  |  | **Điểm bảo vệ:** |  |

***Nhận xét khác của cán bộ thành viên hội đồng***

|  |  |
| --- | --- |
|  | Ngày: … / … / 20…  **Cán bộ thành viên HĐ**  (Ký và ghi rõ họ tên) |

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**

**(DÀNH CHO CÁN BỘ PHẢN BIỆN)**

Tên đề tài: Thiết kế và thực hiện giao diện (frontend) cho nền tảng giám sát thành phố thông minh qua hình ảnh

Họ tên SV: Vũ Đức Tuấn MSSV: 20182869

Cán bộ phản biện:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tiêu chí** (Điểm tối đa) | **Hướng dẫn đánh giá tiêu chí** | **Điểm tiêu chí** |
| 1 | **Trình bày quyển ĐATN (4 điểm)** | Đồ án trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v. |  |
| Kỹ năng diễn đạt, phân tích, giải thích, lập luận: cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v. |
| 2 | **Nội dung và kết quả đạt được  (5,5 điểm)** | Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận. |  |
| Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan. |
| Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/ tính sáng tạo trong nội dung và kết quả đồ án. |
| 3 | **Điểm thành tích  (1 điểm)** | Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. **(1 điểm)** |  |
| Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. **(0,5 điểm)** |
|  | | **Điểm tổng các tiêu chí:** |  |
| **Điểm phản biện:** |  |

***Nhận xét khác của cán bộ phản biện***

|  |  |
| --- | --- |
|  | Ngày: … / … / 20…  **Cán bộ phản biện**  (Ký và ghi rõ họ tên) |

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ 4.0, việc quản lý và giám sát thông tin từ hệ thống camera giám sát đã trở thành yếu tố quan trọng, hỗ trợ các cơ quan chức năng nâng cao hiệu quả quản lý đô thị, bảo đảm an ninh trật tự, và tối ưu hóa nguồn lực. Thành phố Hà Nội - với vai trò là thủ đô và trung tâm chính trị, kinh tế, văn hóa của cả nước - đang không ngừng ứng dụng các giải pháp công nghệ tiên tiến nhằm xây dựng một thành phố thông minh.

Đề tài "Thiết kế và thực hiện giao diện (frontend) cho nền tảng giám sát thành phố thông minh qua hình ảnh" ra đời với mục tiêu xây dựng một nền tảng quản lý tập trung cho các luồng camera giám sát. VSM không chỉ cung cấp khả năng giám sát theo thời gian thực mà còn tích hợp các công cụ lọc, phân loại dữ liệu theo địa điểm và đối tượng quản lý. Qua đó, hệ thống hỗ trợ các đơn vị liên quan dễ dàng theo dõi, phân tích và đưa ra quyết định chính xác trong các tình huống quan trọng.

Đề tài tập trung vào việc áp dụng các công nghệ hiện đại như **Node.js** và **NestJS** để đảm bảo hệ thống có khả năng xử lý dữ liệu nhanh chóng và ổn định. Đồng thời, các chức năng như phân quyền người dùng, quản lý vị trí camera, và hỗ trợ đa luồng sẽ được triển khai nhằm đáp ứng các nhu cầu thực tiễn của thành phố Hà Nội.

Em tin rằng hệ thống VSM khi được đưa vào ứng dụng sẽ góp phần nâng cao năng lực quản lý, giảm tải công việc cho các đơn vị chức năng, đồng thời mang lại một bước tiến quan trọng trong lộ trình xây dựng đô thị thông minh của thủ đô.

Em xin chân thành cảm ơn sự hỗ trợ và đóng góp ý kiến từ các chuyên gia, giảng viên, và bạn bè đồng nghiệp trong quá trình thực hiện đề tài này.

Hà Nội, ngày 06 tháng 01 năm 2025

Sinh viên

Vũ Đức Tuấn

**TÓM TẮT BÁO CÁO**

Báo cáo chia làm 5 chương chính:

* Chương 1: Giới thiệu
  + Lý do chọn đề tài
  + Mục tiêu và phạm vi nghiên cứu
  + Phương pháp nghiên cứu
* Chương 2: Phân tích và thiết kế hệ thống
  + Phân tích yêu cầu
  + Thiết kế kiến trúc hệ thống
  + Thiết kế cơ sở dữ liệu
* Chương 3: Triển khai hệ thống
  + Công nghệ và công cụ sử dụng
  + Cài đặt môi trường phát triển
  + Xây dựng các thành phần chính
  + Tích hợp và kiểm thử
* Chương 4: Kết quả và đánh giá
  + Giao diện sản phẩm
  + Kết quả đạt được
  + Đánh giá kết quả
  + Đề xuất cải thiện
  + Đánh giá tổng quan
* Chương 5: Kết luận và hướng phát triển
  + Kết luận chung
  + Hướng phát triển trong tương lai

**LỜI CAM ĐOAN**

Em là Vũ Đức Tuấn, mã số sinh viên 20182869, sinh viên lớp Điện tử 05 khóa 63. Người hướng dẫn là PGS. GVCC Nguyễn Tài Hưng. Em xin cam đoan toàn bộ nội dung được trình bày trong đồ án “Thiết kế và thực hiện giao diện (frontend) cho nền tảng giám sát thành phố thông minh qua hình ảnh” là kết quả quá trình tìm hiểu và nghiên cứu của em. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ, các tài liệu tham khảo được liệt kê rõ ràng. Em xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH i](#_Toc129452420)

[DANH MỤC BẢNG BIỂU iii](#_Toc129452421)

[DANH MỤC KÍ HIỆU VÀ MỘT SỐ TỪ VIẾT TẮT iv](#_Toc129452422)

[BẢNG ĐỐI CHIẾU THUẬT NGỮ ANH-VIỆT v](#_Toc129452423)

[CHƯƠNG 1. MỞ ĐẦU 1](#_Toc129452424)

[1.1 Đặt vấn đề 1](#_Toc129452425)

[1.2 Mục đích đề tài 2](#_Toc129452426)

[1.3 Hướng tiếp cận và đề xuất cải tiến 2](#_Toc129452427)

[1.3.1 Hướng tiếp cận 2](#_Toc129452428)

[1.3.2 Đề xuất cải tiến 3](#_Toc129452429)

[1.4 Đánh giá, nhận xét kết quả 3](#_Toc129452430)

[1.5 Các công cụ sử dụng 4](#_Toc129452431)

[1.5.1 Các phần mềm hỗ trợ 4](#_Toc129452432)

[1.5.2 Các package được cài đặt 5](#_Toc129452433)

[CHƯƠNG 2. GIỚI THIỆU VỀ TRUY XUẤT HÌNH ẢNH 7](#_Toc129452434)

[2.1 Truy xuất hình ảnh 7](#_Toc129452435)

[2.1.1 Giới thiệu 7](#_Toc129452436)

[2.1.2 Truy xuất hình ảnh dựa trên văn bản 7](#_Toc129452437)

[2.1.3 Truy xuất hình ảnh dựa trên nội dung 8](#_Toc129452438)

[2.1.4 Truy xuất hình ảnh dựa trên ngữ nghĩa 10](#_Toc129452439)

[2.2 So sánh, đánh giá các kỹ thuật truy xuất hình ảnh 11](#_Toc129452440)

[2.2.1 Ưu/Nhược điểm của các kỹ thuật truy xuất hình ảnh 11](#_Toc129452441)

[2.2.2 So sánh, đánh giá các kỹ thuật 13](#_Toc129452442)

[2.3 Giới thiệu mô hình hình ảnh-ngôn ngữ cho bài toán truy xuất hình ảnh 14](#_Toc129452443)

[CHƯƠNG 3. MÔ HÌNH ĐÀO TẠO TRƯỚC HÌNH ẢNH-NGÔN NGỮ TƯƠNG PHẢN 16](#_Toc129452444)

[3.1 Mô hình CLIP 16](#_Toc129452445)

[3.1.1 Giới thiệu 17](#_Toc129452446)

[3.1.2 Phương thức hoạt động 18](#_Toc129452447)

[3.1.3 Đào tạo 21](#_Toc129452448)

[3.1.4 Đặc điểm nổi bật 22](#_Toc129452449)

[3.1.5 Hạn chế 24](#_Toc129452450)

[3.1.6 Tác động rộng hơn 25](#_Toc129452451)

[3.2 Kiến trúc mô hình 25](#_Toc129452452)

[3.3 Các ứng dụng của CLIP 29](#_Toc129452453)

[3.3.1 Phân loại zero-shot 29](#_Toc129452454)

[3.3.2 Tìm kiếm hình ảnh 30](#_Toc129452455)

[3.3.3 Tạo ảnh 31](#_Toc129452456)

[3.3.4 Chú thích hình ảnh 32](#_Toc129452457)

[3.3.5 Phát hiện vật thể 33](#_Toc129452458)

[3.4 Transformer 34](#_Toc129452459)

[3.4.1 Mô hình Sequence-to-Sequence sử dụng RNNs 35](#_Toc129452460)

[3.4.2 Long Short-Term Memory (LSTM) 36](#_Toc129452461)

[3.4.3 Transformer Model 37](#_Toc129452462)

[3.5 Vision Transformer 44](#_Toc129452463)

[3.5.1 Linear Projection and Flattend Patches 44](#_Toc129452464)

[3.5.2 Transformer Encoder 46](#_Toc129452465)

[3.6 CLIP đa ngôn ngữ (Multilingual-CLIP) 48](#_Toc129452466)

[CHƯƠNG 4. TRIỂN KHAI ỨNG DỤNG TRUY XUẤT DỰA TRÊN MÔ HÌNH CLIP ĐA NGÔN NGỮ 51](#_Toc129452467)

[4.1 Sơ đồ khối 51](#_Toc129452468)

[4.1.1 Khối đầu vào 51](#_Toc129452469)

[4.1.2 Khối mã hóa 51](#_Toc129452470)

[4.1.3 Khối xử lý 52](#_Toc129452471)

[4.1.4 Khối hiển thị 52](#_Toc129452472)

[4.2 Nguyên lý hoạt động của hệ thống 52](#_Toc129452473)

[4.3 Triển khai các đề xuất 53](#_Toc129452474)

[4.3.1 Phương pháp xử lý đa luồng trên CPU 53](#_Toc129452475)

[4.3.2 Tối ưu mô hình cho bộ mã hóa hình ảnh 54](#_Toc129452476)

[4.3.3 Tối ưu trong quá trình so sánh sử dụng phân cụm vector 56](#_Toc129452477)

[4.4 Giao diện sản phẩm 57](#_Toc129452478)

[4.5 Kết quả 58](#_Toc129452479)

[4.5.1 Kết quả triển khai các đề xuất cải tiến 58](#_Toc129452480)

[4.5.2 Kết quả chương trình 61](#_Toc129452481)

[CHƯƠNG 5. TỔNG KẾT 64](#_Toc129452482)

[5.1 Kết luận 64](#_Toc129452483)

[5.2 Hướng phát triển trong tương lai 64](#_Toc129452484)

[TÀI LIỆU THAM KHẢO 65](#_Toc129452485)

[PHỤ LỤC 67](#_Toc129452486)

# DANH MỤC HÌNH ẢNH

[**Hình 2.1 Sơ đồ Usecase của hệ thống 8**](#_Toc188365127)

[**Hình 2.2 Sơ đồ Sign up (Sequence) của hệ thống 9**](#_Toc188365128)

[**Hình 2.3 Sơ đồ Login (Sequence) của hệ thống 10**](#_Toc188365129)

[**Hình 2.4 Sơ đồ tạo luồng video stream (Sequence) của hệ thống 11**](#_Toc188365130)

[**Hình 2.5 Sơ đồ lưu trữ video (Sequence) của hệ thống 12**](#_Toc188365131)

[**Hình 3.1 Phần mềm VS Code 16**](#_Toc188365132)

[**Hình 3.2 Phần mềm DBeaver 17**](#_Toc188365133)

[**Hình 3.3 Phần mềm Postman 18**](#_Toc188365134)

[**Hình 3.4 Phần mềm OBS Studio 19**](#_Toc188365135)

[**Hình 3.5 Hệ thống phần mềm Node.js 20**](#_Toc188365136)

[**Hình 3.6 Thư viện React 21**](#_Toc188365137)

[**Hình 3.7 Framework NestJS 22**](#_Toc188365138)

[**Hình 3.8 Kiểm tra version Node và npm 24**](#_Toc188365139)

[**Hình 3.9 Cài đặt & triển khai backend NestJS 24**](#_Toc188365140)

[**Hình 3.10 Cấu hình server API và server RTMP 25**](#_Toc188365141)

[**Hình 3.11 Cây thư mục hoàn chỉnh của backend 26**](#_Toc188365142)

[**Hình 3.12 Cài đặt & triển khai frontend ReactJS 26**](#_Toc188365143)

[**Hình 3.13 Cây thư mục hoàn chỉnh của frontend 27**](#_Toc188365144)

[**Hình 3.14 Khởi tạo RTMP Server 28**](#_Toc188365145)

[**Hình 4.1 Giao diện màn Login 30**](#_Toc188365146)

[**Hình 4.2 Giao diện màn Signup 30**](#_Toc188365147)

[**Hình 4.3 Giao diện mặc định khi chưa đăng nhập 31**](#_Toc188365148)

[**Hình 4.4 Giao diện mặc định khi đã đăng nhập 31**](#_Toc188365149)

[**Hình 4.5 Màn hình setting camera 32**](#_Toc188365150)

[**Hình 4.6 Màn hình quản lý camera 32**](#_Toc188365151)

[**Hình 4.7 Test tải API với Apache JMeter 33**](#_Toc188365152)

# DANH MỤC KÍ HIỆU VÀ MỘT SỐ TỪ VIẾT TẮT

|  |  |
| --- | --- |
| VSM | Video Stream Manager |
| OBS | Open Broadcaster Software |
| VSC | Visual Studio Code |
| RTMP | Real-Time Messaging Protocol |
| FLV | Flash Video |
| HLS | HTTP Live Streaming |
|  |  |
|  |  |
|  |  |

# MỞ ĐẦU

## Lý do chọn đề tài

Trong bối cảnh đô thị hóa và phát triển công nghệ nhanh chóng, các thành phố lớn như Hà Nội đang đối mặt với nhiều thách thức trong quản lý an ninh, giao thông và môi trường sống. Để giải quyết những vấn đề này, việc xây dựng các thành phố thông minh với hệ thống giám sát và quản lý hiệu quả là một xu hướng tất yếu. Trong đó, hệ thống camera giám sát đóng vai trò quan trọng, giúp cải thiện công tác quản lý đô thị và đảm bảo an ninh trật tự. Tuy nhiên, việc vận hành và tích hợp các nguồn dữ liệu camera từ nhiều khu vực khác nhau lại gặp nhiều khó khăn, như:

Sự phân mảnh của hệ thống hiện tại: Hiện nay, các thành phố thông minh có hàng nghìn camera giám sát được triển khai tại các quận, huyện, tuy nhiên chúng thường hoạt động độc lập, không được tích hợp vào một hệ thống quản lý tập trung. Điều này gây khó khăn trong việc theo dõi và xử lý tình huống một cách hiệu quả.

Khó khăn trong quản lý và tìm kiếm thông tin: Khi cần truy xuất dữ liệu từ hệ thống camera, các đơn vị chức năng phải mất nhiều thời gian và công sức để lọc tìm thông tin từ các nguồn khác nhau. Điều này làm giảm hiệu suất trong việc xử lý các sự cố khẩn cấp.

Nhu cầu tối ưu hóa việc giám sát theo thời gian thực: Trong các tình huống khẩn cấp như tai nạn giao thông, mất an ninh trật tự, hoặc thiên tai, việc theo dõi trực tiếp và chính xác thông tin từ camera là yếu tố quyết định. Một hệ thống tập trung và tối ưu hóa có thể giúp đưa ra quyết định nhanh chóng hơn.

Ứng dụng công nghệ để xây dựng thành phố thông minh: Đề tài này phù hợp với xu hướng phát triển công nghệ và chuyển đổi số tại Việt Nam. Hệ thống Video Stream Manager không chỉ đáp ứng nhu cầu quản lý dữ liệu video mà còn có khả năng mở rộng, tích hợp với các giải pháp khác, như trí tuệ nhân tạo (AI) để phân tích dữ liệu, phát hiện bất thường, hoặc dự đoán các tình huống nguy hiểm.

Góp phần hiện đại hóa cơ sở hạ tầng công nghệ của thành phố thông minh: Việc triển khai hệ thống VSM sẽ giúp tăng cường năng lực quản lý và hỗ trợ xây dựng hình ảnh một đô thị hiện đại, an toàn và đáng sống.

Với những lý do trên, việc nghiên cứu và triển khai hệ thống Video Stream Manager không chỉ giải quyết các vấn đề cấp bách hiện nay mà còn đóng góp vào sự phát triển bền vững của thành phố, đồng thời mở ra cơ hội ứng dụng rộng rãi cho các địa phương khác trên cả nước.

## Mục tiêu nghiên cứu

### Xây dựng hệ thống quản lý video stream tập trung:

Nâng cao hiệu quả giám sát và quản lý đô thị: Hệ thống cần hỗ trợ các cơ quan chức năng theo dõi tình hình an ninh, giao thông và các sự cố trong thời gian thực, giúp tăng tốc độ phản ứng và giảm thiểu thời gian xử lý tình huống khẩn cấp.

### Ứng dụng công nghệ hiện đại để tối ưu hóa hiệu suất:

Tích hợp các công nghệ tiên tiến như Node.js và NestJS để đảm bảo hệ thống hoạt động ổn định, xử lý dữ liệu nhanh chóng và dễ dàng mở rộng trong tương lai. Cung cấp các công cụ lọc dữ liệu và tìm kiếm nâng cao nhằm cải thiện trải nghiệm người dùng và tối ưu hóa quy trình làm việc.

### Hỗ trợ phân quyền và bảo mật dữ liệu:

Thiết kế hệ thống phân quyền chặt chẽ, đảm bảo mỗi người dùng chỉ được truy cập thông tin và chức năng phù hợp với vai trò của họ. Đồng thời, đảm bảo dữ liệu video được bảo mật, tránh các rủi ro về an ninh mạng.

### Định hướng phát triển lâu dài:

Đề xuất các giải pháp mở rộng hệ thống, như tích hợp trí tuệ nhân tạo (AI) để phân tích video, phát hiện bất thường và cung cấp cảnh báo sớm.

## Phạm vi ứng dụng

### Địa bàn áp dụng

Hệ thống sẽ được áp dụng trên phạm vi toàn thành phố thông minh.

Các khu vực trọng điểm như khu hành chính, khu dân cư đông đúc, trường học, bệnh viện và các nút giao thông chính sẽ được ưu tiên triển khai và giám sát.

### Đối tượng sử dụng

Tất cả người dùng đăng kí hệ thống đều có thể tham gia sử dụng hệ thống.

Admin hệ thống: Có thể quản lý những user tham gia hệ thống.

## Phương pháp nghiên cứu

Để đảm bảo tính khả thi và hiệu quả của đề tài "Thiết kế và thực hiện giao diện (frontend) cho nền tảng giám sát thành phố thông minh qua hình ảnh" các phương pháp nghiên cứu được áp dụng bao gồm:

### Phương pháp thu thập thông tin.

Tìm hiểu các tài liệu, báo cáo liên quan đến quản lý video stream trong thành phố thông minh.

Tham khảo các hệ thống giám sát hiện có như EarthCam và các nền tảng mã nguồn mở.

### Phương pháp phân tích và thiết kế.

Xác định các yêu cầu chức năng (quản lý camera, xem video trực tiếp, lọc dữ liệu) và phi chức năng (bảo mật, hiệu suất, khả năng mở rộng).

Sử dụng các công cụ như UML (Unified Modeling Language) để xây dựng biểu đồ luồng dữ liệu (DFD), biểu đồ hoạt động (Activity Diagram) và biểu đồ trình tự (Sequence Diagram).

### Phương pháp phát triển và triển khai.

Sử dụng công nghệ hiện đại như Node.js cho backend, NestJS để tổ chức code theo module và RTMP để truyền dữ liệu video thời gian thực.

Triển khai hệ thống thử nghiệm tại một số khu vực trọng điểm ở Hà Nội, thu thập phản hồi từ người dùng để tối ưu hóa hệ thống

### Phương pháp đánh giá và tổng kết.

Đánh giá tính ổn định, tốc độ xử lý và mức độ đáp ứng của hệ thống trong các tình huống thực tế.

Đưa ra kế hoạch phát triển hệ thống trong tương lai, hướng tới tích hợp AI và mở rộng quy mô.

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Chương 2 của báo cáo này sẽ giới thiệu về các phần như: Phân tích yêu cầu, thiết kế kiến trúc hệ thống, thiết kế cơ sở dữ liệu và quy trình hoạt động của hệ thống.

## Phân tích yêu cầu

### Yêu cầu chức năng

Yêu cầu chức năng của hệ thống Video Stream Manager bao gồm các tính năng cần thiết mà hệ thống phải thực hiện để đáp ứng nhu cầu của người dùng và các bên liên quan. Các tính năng này cần phải được triển khai để giải quyết các vấn đề đã nêu trên:

1. **Quản lý luồng video**
   * **Tạo luồng video mới**: Người dùng có thể tạo và bắt đầu một luồng video mới từ các nguồn khác nhau (camera giám sát, sự kiện livestream, v.v.).
   * **Dừng hoặc ngừng luồng video**: Người dùng có thể dừng hoặc ngừng một luồng video bất kỳ trong quá trình phát.
   * **Quản lý luồng video đang hoạt động**: Hệ thống sẽ cung cấp thông tin về trạng thái của các luồng video (đang phát, tạm dừng, lỗi, v.v.), giúp người dùng dễ dàng theo dõi và quản lý.
   * **Chuyển đổi định dạng video**: Hệ thống phải hỗ trợ chuyển đổi giữa các định dạng video để tối ưu hóa việc phát trên các thiết bị và nền tảng khác nhau (ví dụ: từ RTMP sang HLS hoặc WebRTC).
   * **Quản lý nguồn video**: Hệ thống phải có khả năng tích hợp nhiều nguồn video từ các camera khác nhau hoặc các hệ thống khác.
2. **Quản lý người dùng và phân quyền**
   * **Xác thực người dùng**: Người dùng cần phải đăng nhập vào hệ thống để truy cập các tính năng quản lý video. Hệ thống sử dụng Firebase Authentication để xác thực và quản lý người dùng.
   * **Phân quyền người dùng**: Có các loại người dùng khác nhau với các quyền truy cập khác nhau, ví dụ:
     + Người quản trị có thể quản lý tất cả các luồng video và người dùng.
     + Người dùng thông thường chỉ có thể xem các luồng video đã được chia sẻ công khai hoặc theo phân quyền của họ.
   * **Quản lý nhóm người dùng**: Tạo và quản lý các nhóm người dùng để dễ dàng phân quyền truy cập cho các loại video khác nhau.
3. **Lưu trữ và phân tích video**
   * **Lưu trữ video**: Các luồng video được ghi lại và lưu trữ trong hệ thống để phục vụ việc xem lại sau này. Hệ thống phải có khả năng lưu trữ video lâu dài và dễ dàng truy xuất.
   * **Tìm kiếm video**: Hệ thống phải hỗ trợ tính năng tìm kiếm và lọc video theo các tiêu chí như thời gian, vị trí, loại sự kiện, v.v.
4. **Quản lý và phát video theo yêu cầu**
   * **Xem video trực tiếp**: Người dùng có thể xem các luồng video trực tiếp từ các nguồn khác nhau qua giao diện web hoặc ứng dụng di động.
   * **Quản lý phát lại video**: Hệ thống cung cấp tính năng phát lại các video đã ghi lại với khả năng tìm kiếm, tạm dừng, tua đi, tua lại.
5. **Bảo mật và quyền truy cập**
   * **Xác thực qua token**: Sử dụng các mã thông báo (token) để xác thực quyền truy cập video đối với người dùng, giúp bảo mật và kiểm soát người xem.

### Yêu cầu phi chức năng

Yêu cầu phi chức năng đề cập đến các đặc tính không liên quan trực tiếp đến chức năng chính của hệ thống nhưng có ảnh hưởng quan trọng đến hiệu suất, bảo mật và khả năng mở rộng của hệ thống.

1. **Hiệu suất và khả năng mở rộng**
   * **Khả năng mở rộng**: Hệ thống cần phải có khả năng mở rộng để xử lý hàng nghìn luồng video đồng thời. Điều này bao gồm khả năng tăng cường tài nguyên phần cứng và phần mềm khi số lượng người dùng và luồng video tăng lên.
   * **Hiệu suất phát video**: Các luồng video phải được phát với độ trễ thấp và chất lượng cao. Hệ thống cần tối ưu hóa băng thông và tài nguyên để đảm bảo việc phát video liên tục mà không gián đoạn.
2. **Độ tin cậy**
   * **Khả năng phục hồi sau sự cố**: Hệ thống phải đảm bảo độ tin cậy cao, có khả năng tự động khôi phục khi gặp sự cố hoặc lỗi hệ thống. Các video cần được bảo vệ tránh bị mất mát khi gặp sự cố.
   * **Cảnh báo và thông báo**: Hệ thống phải gửi cảnh báo ngay khi có sự cố xảy ra, ví dụ như khi một luồng video gặp lỗi, hoặc khi hệ thống phát hiện sự cố trong quá trình lưu trữ.
3. **Tính tương thích**
   * **Tương thích với các thiết bị**: Hệ thống phải tương thích với nhiều loại thiết bị khác nhau, bao gồm máy tính để bàn, điện thoại di động, máy tính bảng, và các thiết bị phát video khác.
   * **Tương thích với các trình duyệt**: Hệ thống cần đảm bảo rằng video có thể phát trên các trình duyệt phổ biến như Chrome, Firefox, Safari và Edge.
4. **Bảo mật**
   * **Mã hóa và bảo vệ dữ liệu**: Mọi dữ liệu truyền tải qua hệ thống (bao gồm video và thông tin người dùng) phải được mã hóa để đảm bảo an toàn.
   * **Xác thực người dùng và bảo mật hệ thống**: Sử dụng các cơ chế bảo mật mạnh mẽ như OAuth, JWT, và các phương thức xác thực khác để bảo vệ hệ thống khỏi các mối đe dọa.
5. **Dễ sử dụng**
   * **Giao diện người dùng trực quan**: Giao diện quản lý video và người dùng cần phải dễ sử dụng, dễ hiểu và dễ dàng thao tác ngay cả với những người không có kinh nghiệm công nghệ.
   * **Tối ưu hóa trải nghiệm người dùng**: Hệ thống cần cung cấp trải nghiệm người dùng mượt mà với khả năng truy cập nhanh chóng vào các video và tính năng quản lý.

## Thiết kế kiến trúc hệ thống

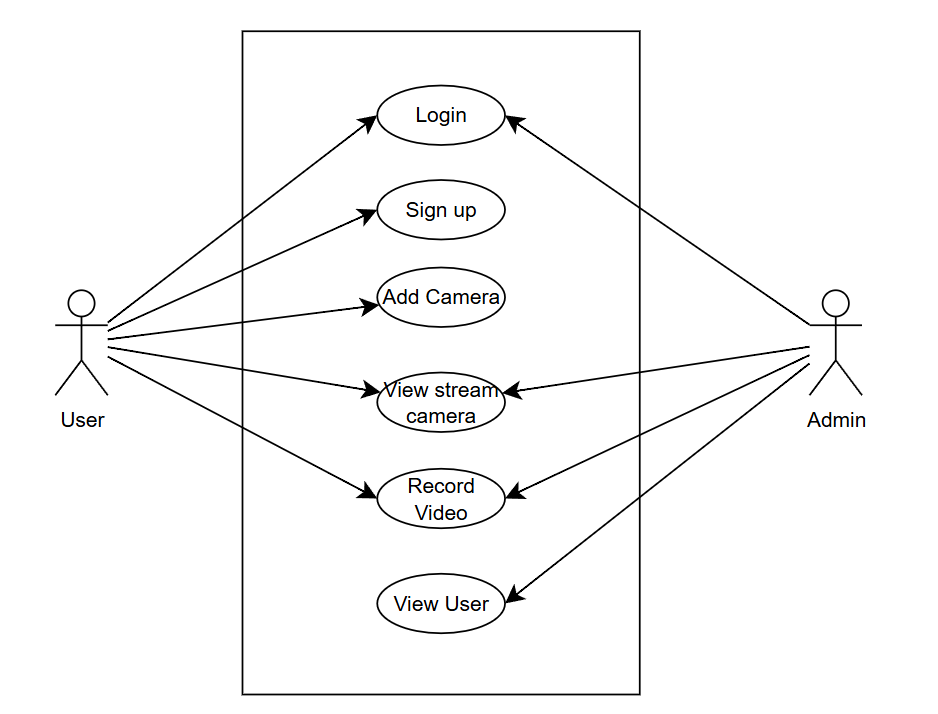
### Kiến trúc tổng thể

Kiến trúc hệ thống Video Stream Manager (VSM) sẽ được xây dựng dựa trên một mô hình kiến trúc Microservices kết hợp với các thành phần phân tán, cho phép hệ thống dễ dàng mở rộng và bảo trì. Hệ thống sẽ bao gồm các lớp chính như sau:

* **Frontend (Giao diện người dùng)**:
  + Giao diện người dùng sẽ được phát triển bằng **ReactJS** cung cấp các chức năng quản lý luồng video, tìm kiếm video, xem video trực tiếp và quản lý người dùng.
  + Giao diện sẽ kết nối với các dịch vụ backend qua API.
* **Backend (Dịch vụ xử lý và quản lý video)**:
  + **Node.js** sẽ được sử dụng làm nền tảng cho các dịch vụ backend, với thư viện hỗ trợ NestJS để xây dựng API và xử lý các yêu cầu từ frontend.
  + **NestJS** có thể được sử dụng để xây dựng các dịch vụ backend, với khả năng tổ chức mã nguồn sạch sẽ và dễ mở rộng.
  + Dịch vụ backend bao gồm các thành phần chính sau:
    - **Video Stream Management Service**: Xử lý việc quản lý, phát và dừng các luồng video, cũng như chuyển đổi giữa các định dạng video.
    - **User Authentication Service**: Sử dụng JWT Authentication để quản lý người dùng và phân quyền truy cập.
    - **Storage Service**: Lưu trữ các video đã ghi lại và các metadata liên quan bằng Firebase.
* **Video Stream Servers**:
  + **Node-Media-Server** để quản lý các luồng video trực tiếp (live streaming) thông qua các giao thức như RTMP, HLS và FLV.
  + Video stream servers sẽ tiếp nhận các luồng video từ các camera hoặc thiết bị phát video và phân phối chúng đến người dùng cuối.
* **Database (Cơ sở dữ liệu)**:
  + **PostgreSQL** sẽ là cơ sở dữ liệu quan hệ chính để lưu trữ thông tin người dùng, video metadata (ví dụ: thông tin video, thời gian bắt đầu, kết thúc, trạng thái), các quyền truy cập và phân quyền.

### Quy trình hoạt động của hệ thống

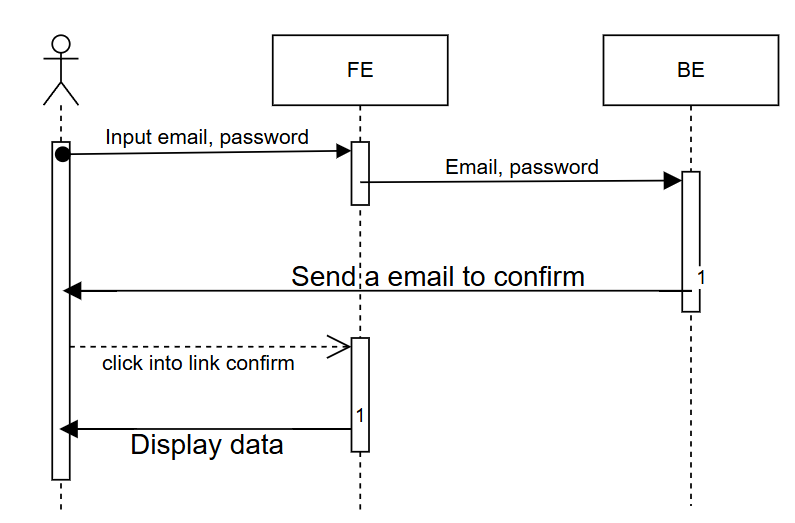
Đây là sơ đồ Usecase của hệ thống thể hiện các chức năng mà mỗi role có thể thực hiện được.



Hình . Sơ đồ Usecase của hệ thống

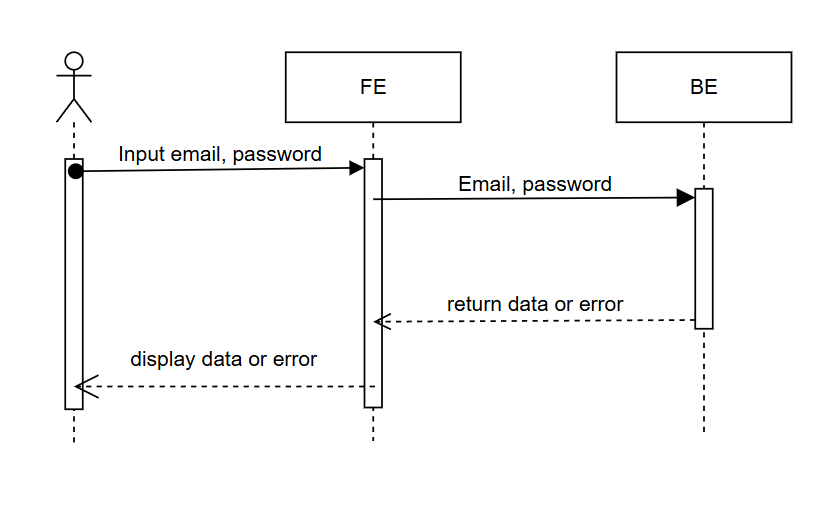
Còn dưới đây là chi tiết về các quá trình của hệ thống:

1. **Quá trình đăng kí**
   1. Người dùng nhập email, password để đăng kí tài khoản
   2. Server sẽ gửi 1 email xác thực tài khoản về email đăng kí của người dùng
   3. Người dùng click vào link trong email để hoàn tất xác thực và đưa đến trang đăng nhập

****

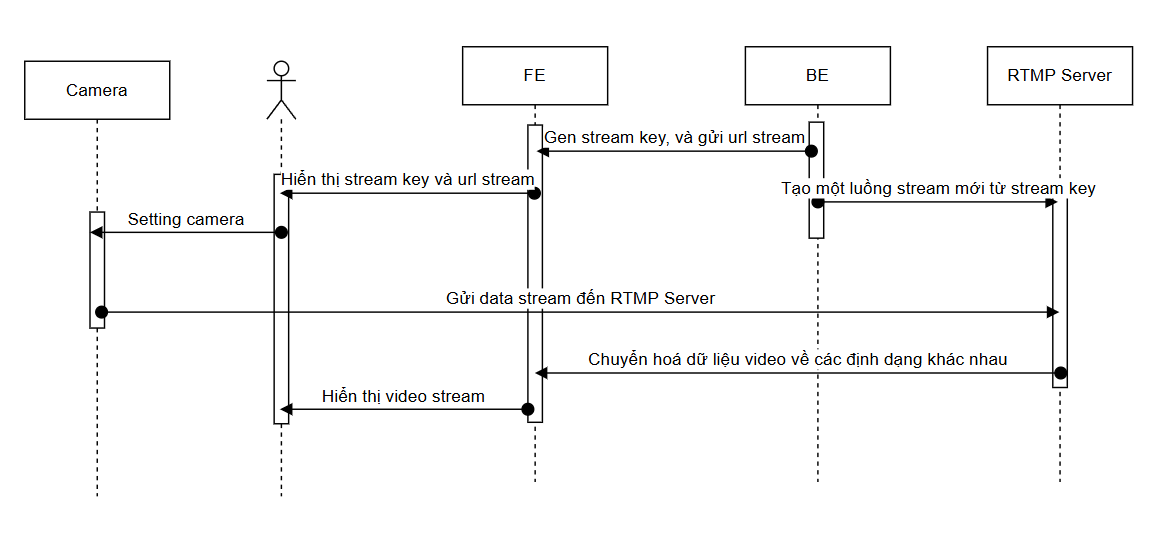
Hình . Sơ đồ Sign up (Sequence) của hệ thống

1. **Quá trình đăng nhập**
   1. Người dùng nhập email và password để đăng nhập
   2. FE gửi dữ liệu người dùng vừa nhập lên server
   3. Server trả về dữ liệu cho FE và hiển thi cho người dùng thấy kết quả



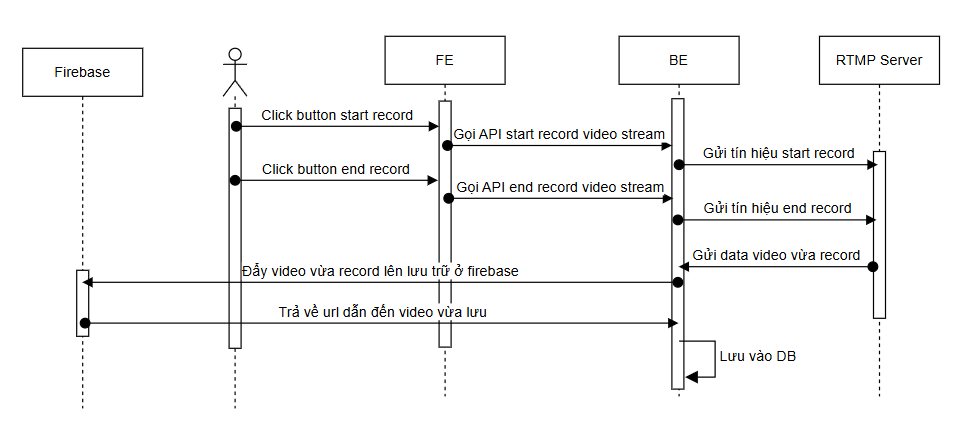
Hình . Sơ đồ Login (Sequence) của hệ thống

1. **Quá trình tạo luồng video stream (Add camera)**:
   1. Người dùng sẽ được cung cấp 1 url server và 1 stream key để setup camera của người dùng thông qua ứng dụng OBS. Người dùng setting vị trí để camera được hiển thị trên bản đồ của hệ thống.
   2. Server stream (sử dụng FLV) sẽ bắt đầu phát luồng video.
   3. Đồng thời, thông tin luồng video có thể lưu vào cơ sở dữ liệu để người dùng có thể tìm kiếm và quản lý sau này.



Hình . Sơ đồ tạo luồng video stream (Sequence) của hệ thống

1. **Quá trình phát video trực tiếp**:
   1. Khi người dùng muốn xem video trực tiếp, thao tác trực tiếp trên bản đồ hệ thống (click vào icon camera trên bản đồ để xem chi tiết). Hoặc người dùng truy cập vào tab VSM để xem toàn bộ camera đang được cho phép truy cập.
   2. **Stream server luôn giữ kết nối và phát video stream từ các camera mà người dùng đã cung cấp thông qua giao thức FLV**.
2. **Quá trình lưu trữ video**:
   1. Người dùng có thể lưu trữ video bằng chắc năng record của server. Bằng cách click vào button Start Recording và Stop Recording.
   2. **Video sau khi được hoàn tất sẽ được đẩy lên lưu trữ ở storage của Firebase và lưu url dẫn đến video này vào trong database của hệ thống thông qua service của server**



Hình . Sơ đồ lưu trữ video (Sequence) của hệ thống

1. **Quá trình quản lý người dùng**:
   1. Khi người dùng đăng kí, hệ thống sẽ gửi 1 email về tài khoản mà người dùng đã đăng kí để xác thực và đăng nhập vào hệ thống.
   2. Admin có thể quản lý tất cả các người dùng khác trong hệ thống.

## Thiết kế cơ sở dữ liệu

### Phân tích yêu cầu cơ sở dữ liệu

Cơ sở dữ liệu cần lưu trữ các thông tin sau:

1. **Thông tin người dùng**: Để xác thực và phân quyền.
2. **Thông tin video stream**: Bao gồm trạng thái, thời gian, camera nguồn và metadata.
3. **Thông tin camera**: Vị trí, trạng thái hoạt động, liên kết với video streams.
4. **Phân quyền truy cập**: Xác định quyền của từng người dùng.
5. **Lưu trữ video và metadata**: Thông tin video đã lưu trữ để phục vụ cho việc tìm kiếm, phát lại và phân tích.

### Thiết kế bảng dữ liệu

#### ****Bảng users****

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | Mã định danh duy nhất cho mỗi người dùng. |
| email | VARCHAR(255) | UNIQUE,NOT NULL | Email dùng để đăng nhập |
| password | VARCHAR(255) | NOT NULL | Mật khẩu (được mã hóa). |
| avatar | VARCHAR(255) |  | Avatar của người dùng |
| isEmailConfirmed | BOOLEAN | DEFAULT 'false' | Field kiểm tra xác thực người dùng đã xác thực email hay chưa |
| confirmationToken | VARCHAR(255) |  | Dùng để xác thực email. |
| role | ENUM('admin', 'user') | DEFAULT 'user' | Vai trò của người dùng. |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Thời gian tạo tài khoản. |
| updated\_at | DATETIME | ONUPDATE CURRENT\_TIMESTAMP | Thời gian cập nhật gần nhất. |

#### Bảng cameras

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | Mã định danh của camera. |
| userId | INT | FOREIGN KEY | Khoá phụ liên kết tới bảng Users |
| streamKey | VARCHAR(255) |  | Key để định danh kết nối RTMP |
| url | VARCHAR(255) | NOT NULL | Đường dẫn stream |
| lat | VARCHAR(255) |  | Vĩ độ của camera |
| lng | VARCHAR(255) |  | Kinh độ của camera |
| country | VARCHAR(255) |  | Quốc gia của camera |
| city | VARCHAR(255) |  | Thành phố của camera |
| district | VARCHAR(255) |  | Quận đặt camera |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Thời gian tạo tài khoản. |
| updated\_at | DATETIME | ONUPDATE CURRENT\_TIMESTAMP | Thời gian cập nhật gần nhất. |

#### Bảng records

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Constraints** | **Description** |
| id | INT | PRIMARY KEY | Mã định danh của record. |
| userId | INT | FOREIGN KEY | Khoá phụ liên kết tới bảng Users |
| url | VARCHAR(255) | NOT NULL | Đường dẫn tới video đã lưu |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Thời gian tạo tài khoản. |
| updated\_at | DATETIME | ONUPDATE CURRENT\_TIMESTAMP | Thời gian cập nhật gần nhất. |

#### Quan hệ giữa các bảng

users – cameras: Quan hệ 1 – N, một user có thể setting nhiều camera.

users – records: Quan hệ 1 – N, một user có thể có nhiều record

#### Tối ưu hóa và mở rộng

 **Indexing**:

* Tạo chỉ mục cho các trường như email, camera\_id, và video\_id để tăng tốc độ truy vấn.

 **Sharding và Replication**:

* Với lượng dữ liệu lớn, có thể sử dụng sharding để chia nhỏ dữ liệu theo camera hoặc người dùng.
* Sử dụng replication để đảm bảo dữ liệu an toàn và sẵn sàng cao.

 **Lưu trữ video**:

* Video được lưu trữ trên dịch vụ Firebase và chỉ lưu metadata trong cơ sở dữ liệu.

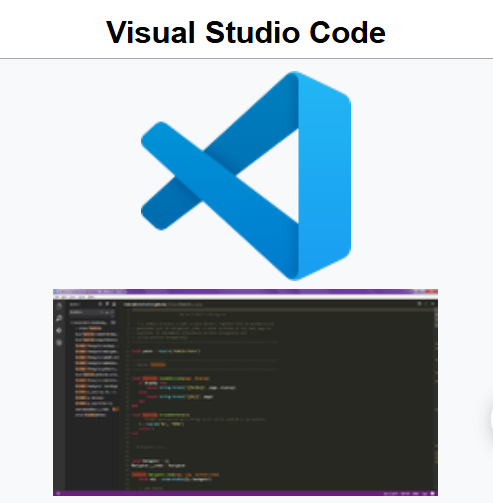
# TRIỂN KHAI HỆ THỐNG

Chương 3 của báo cáo này em sẽ tập trung vào phần triển khai hệ thống.

## Các công nghệ và công cụ sử dụng

### Các phần mềm hỗ trợ

1. **Visual Studio Code**

****

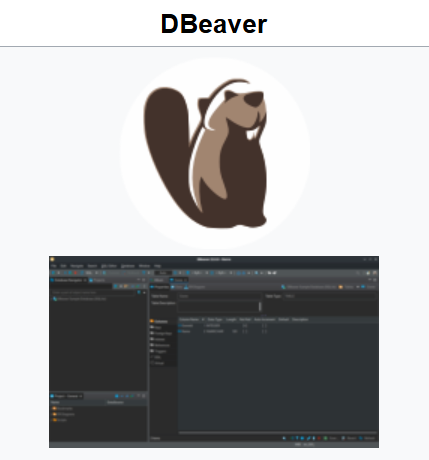
Hình . Phần mềm VS Code

Visual Studio Code là một trình soạn thảo mã nguồn được phát triển bởi Microsoft dành cho Windows, Linux và macOS. Nó hỗ trợ chức năng debug, đi kèm với Git, có chức năng nổi bật cú pháp (syntax highlighting), tự hoàn thành mã thông minh, snippets và cải tiến mã nguồn. Nó cũng cho phép tùy chỉnh, do đó, người dùng có thể thay đổi theme, phím tắt và các tùy chọn khác. Nó miễn phí và là phần mềm mã nguồn mở theo giấy phép MIT, mặc dù bản phát hành của Microsoft là theo giấy phép phần mềm miễn phí.

Visual Studio Code được dựa trên Electron, một nền tảng được sử dụng để triển khai các ứng dụng Node.js máy tính cá nhân chạy trên động cơ bố trí Blink. Mặc dù nó sử dụng nền tảng Electron nhưng phần mềm này không phải là một bản khác của Atom, nó thực ra được dựa trên trình biên tập của Visual Studio Online (tên mã là "Monaco").

Phần mềm này hỗ trợ rất nhiều ngôn ngữ lập trình đặc biệt là Javascript – ngôn ngữ cơ sở lập trình Node.js và React.js. Em sử dụng phần mềm này để thuận tiện trong việc triển khai và phát triển ứng dụng của mình.

1. **DBeaver**

****

Hình . Phần mềm DBeaver

DBeaver là một ứng dụng phần mềm máy khách SQL và là một công cụ quản trị cơ sở dữ liệu. Đối với cơ sở dữ liệu quan hệ, nó sử dụng giao diện lập trình ứng dụng JDBC để tương tác với cơ sở dữ liệu thông qua trình điều khiển JDBC. Đối với các cơ sở dữ liệu khác, nó sử dụng trình điều khiển cơ sở dữ liệu độc quyền.

Em sử dụng DBeaver để quản lý cơ sở dữ liệu Postgres, giúp trực quan hoá dữ liệu xử lý và dễ dàng quản lý dữ liệu.

1. **Postman**

****

Hình . Phần mềm Postman

**Postman** hiện là một trong những công cụ phổ biến nhất được sử dụng trong thử nghiệm các API.

Như ta đã biết, API chịu trách nhiệm kết nối các ứng dụng với nhau, có Postman sẽ giúp cho thao tác với API này trở nên dễ dàng hơn. Thông thường, Postman sẽ được dùng cho API kiểu REST. Với Postman, ta có thể gọi Rest API mà không cần viết dòng code nào.

Postman hỗ trợ tất cả các phương thức HTTP (GET, POST, PUT, PATCH, DELETE, …). Bên cạnh đó, nó còn cho phép lưu lại lịch sử các lần request, rất tiện cho việc sử dụng lại khi cần.

Em sử dụng Postman để dễ dàng kiểm thử các API khi viết code Backend.

1. **Open Broadcaster Software(OBS)**

****

Hình . Phần mềm OBS Studio

OBS Studio là một phần mềm mã nguồn mở và miễn phí được sử dụng để ghi và stream video trực tiếp trên máy tính. OBS là viết tắt của "Open Broadcaster Software". Nó cung cấp cho người dùng các công cụ mạnh mẽ để quay video, chụp ảnh, ghi âm và stream nội dung trực tiếp lên các nền tảng như Twitch, YouTube, Facebook và nhiều nền tảng khác.

OBS Studio cho phép bạn tùy chỉnh các cài đặt và hiệu chỉnh màn hình, âm thanh, video và các thiết lập khác để tạo ra các sản phẩm video chất lượng cao. Bạn có thể thêm các nguồn video từ webcam, màn hình máy tính, file video đã có sẵn và thậm chí các nguồn âm thanh bổ sung. Nó cũng cung cấp các tính năng phức tạp như xử lý video trực tiếp, thiết lập tạo mục, overlay và chuyển đổi giữa các scene để tạo ra những trình diễn đa phương tiện chuyên nghiệp.

OBS Studio phổ biến trong cộng đồng streamer, nhà làm phim và các nhà sản xuất nội dung trực tuyến. Với giao diện đơn giản và khả năng tùy chỉnh cao, OBS Studio là một công cụ mạnh mẽ để tạo ra và chia sẻ nội dung video chất lượng cao trực tiếp trên Internet.

Em sử dụng OBS để thực hiện cài đặt và kết nối các camera stream.

### Cơ sở công nghệ

#### Video Streaming

Video Streaming (truyền phát video trực tuyến) là công nghệ cho phép người dùng xem video trên internet mà không cần tải xuống toàn bộ nội dung. Công nghệ này bao gồm nhiều thành phần và giao thức phối hợp để đảm bảo truyền tải dữ liệu hiệu quả, mượt mà, và tối ưu hóa trải nghiệm người dùng.

##### Thành phần cơ bản của Video Streaming

**a. Máy chủ (Server)**

**Streaming Server**:

Lưu trữ và cung cấp nội dung video.

Các công cụ phổ biến: Node-Media-Server, Wowza Streaming Engine, nginx-rtmp-module.

**Content Delivery Network (CDN)**:

Hệ thống máy chủ phân phối nội dung (như Akamai, Cloudflare, AWS CloudFront), giảm tải máy chủ chính và tăng tốc độ truyền tải.

**b. Máy khách (Client)**

Thiết bị của người dùng như máy tính, điện thoại di động, TV thông minh.

Máy khách nhận dữ liệu qua mạng, giải mã và hiển thị nội dung video.

**c. Giao thức truyền phát**

Định nghĩa cách dữ liệu video và âm thanh được truyền qua mạng.

Các giao thức phổ biến:

**RTMP** (Real-Time Messaging Protocol)

**HLS** (HTTP Live Streaming)

**DASH** (Dynamic Adaptive Streaming over HTTP)

**WebRTC** (Web Real-Time Communication)

##### Các định dạng video và âm thanh

**a. Định dạng container**

Lưu trữ video, âm thanh và metadata.

Các định dạng phổ biến:

**MP4**: Hỗ trợ mã hóa H.264, phù hợp cho hầu hết các thiết bị.

**FLV**: Dành cho RTMP streaming, không còn phổ biến do sự suy giảm của Flash.

**MKV**, **MOV**: Các định dạng có tính linh hoạt cao.

**b. Codec**

Bộ mã hóa và giải mã video/âm thanh.

Codec phổ biến:

**Video**: H.264 (AVC), H.265 (HEVC), VP9, AV1.

**Âm thanh**: AAC, MP3, Opus.

##### Giao thức truyền tải dữ liệu

**a. RTMP (Real-Time Messaging Protocol)**

Giao thức thời gian thực để truyền dữ liệu âm thanh, video và metadata.

Được sử dụng rộng rãi trong live streaming.

**b. HLS (HTTP Live Streaming)**

Giao thức chia nhỏ video thành các phân đoạn nhỏ để truyền qua HTTP.

Hỗ trợ adaptive bitrate (tự động thay đổi chất lượng theo băng thông).

**c. DASH (Dynamic Adaptive Streaming over HTTP)**

Giao thức dựa trên HTTP tương tự HLS nhưng không phụ thuộc vào nền tảng.

**d. WebRTC (Web Real-Time Communication)**

Giao thức thời gian thực cho các ứng dụng giao tiếp trực tuyến như họp video, gọi thoại.

##### Công nghệ hỗ trợ Video Streaming

**a. Mã hóa và nén video**

Sử dụng codec để giảm kích thước tệp video trong khi giữ chất lượng.

H.264 và H.265 là chuẩn phổ biến.

**b. Adaptive Bitrate Streaming**

Điều chỉnh chất lượng video dựa trên tốc độ mạng của người dùng.

Giúp tránh hiện tượng giật hoặc dừng khi phát video.

**c. CDN (Content Delivery Network)**

Phân phối nội dung qua nhiều máy chủ trên toàn thế giới.

Tăng tốc độ truy cập và giảm tải máy chủ chính.

**d. Công nghệ bảo mật**

Mã hóa AES-128 hoặc DRM (Digital Rights Management) bảo vệ nội dung khỏi việc sao chép trái phép.

##### Mô hình hoạt động của Video Streaming

**a.** **Tải nội dung**: Video được tải lên máy chủ streaming.

**b.** **Mã hóa và phân đoạn**:

Video được mã hóa thành định dạng tối ưu (H.264, H.265).

Chia thành các phân đoạn nhỏ (chunks).

**c.** **Phân phối qua mạng**:

Sử dụng CDN và các giao thức truyền tải (HLS, DASH).

**d. Phát lại tại máy khách**:

Thiết bị của người dùng nhận và giải mã video.

Trình phát video hiển thị nội dung dựa trên khả năng mạng.

##### Ứng dụng của Video Streaming

**Live Streaming**: Sự kiện trực tiếp, thể thao, giáo dục trực tuyến.

**Video on Demand (VOD)**: Nền tảng Netflix, YouTube, Amazon Prime.

**Giám sát an ninh**: Camera IP truyền video thời gian thực.

**Truyền thông trực tuyến**: Họp video, hội nghị trực tuyến.

#### Video Storage

Video Storage là quá trình lưu trữ dữ liệu video trên các hệ thống hoặc nền tảng lưu trữ, cho phép truy xuất và quản lý nội dung một cách hiệu quả. Với nhu cầu ngày càng cao về phát trực tuyến, video theo yêu cầu (VOD), và lưu trữ dài hạn, công nghệ video storage đã phát triển để đáp ứng các yêu cầu về dung lượng lớn, tốc độ truy xuất, và khả năng mở rộng.

##### Các yêu cầu đối với Video Storage

**a. Dung lượng lớn:**

Video thường là dữ liệu dung lượng cao, đặc biệt với các định dạng chất lượng cao như 4K, 8K hoặc HDR.

**b. Truy xuất nhanh:**

Yêu cầu truy cập tức thì, đặc biệt cho các ứng dụng phát trực tuyến hoặc phát lại theo yêu cầu.

**c. Khả năng mở rộng:**

Hệ thống cần mở rộng dễ dàng để đáp ứng nhu cầu tăng trưởng nội dung và số lượng người dùng.

**d. Bảo mật:**

Đảm bảo dữ liệu video được bảo vệ khỏi truy cập trái phép.

**e. Độ bền và dự phòng:**

Dữ liệu cần được sao lưu và có khả năng phục hồi sau lỗi.

##### Phương pháp lưu trữ video

**a. Lưu trữ cục bộ (On-Premises Storage)**

**Hệ thống lưu trữ**:

Dùng các máy chủ, ổ cứng (HDD/SSD) hoặc NAS (Network-Attached Storage) để lưu trữ dữ liệu video.

**Ưu điểm**:

Toàn quyền kiểm soát dữ liệu.

Đáp ứng tốt với các nhu cầu nội bộ.

**Nhược điểm**:

Tốn chi phí đầu tư ban đầu và bảo trì.

Khó mở rộng khi dữ liệu tăng nhanh.

**b. Lưu trữ đám mây (Cloud Storage)**

**Dịch vụ lưu trữ đám mây phổ biến**:

AWS S3, Google Cloud Storage, Microsoft Azure Blob Storage, Backblaze B2.

**Ưu điểm**:

Mở rộng linh hoạt.

Khả năng truy cập từ mọi nơi.

Tích hợp các dịch vụ khác như CDN.

**Nhược điểm**:

Phụ thuộc vào nhà cung cấp dịch vụ.

Chi phí có thể tăng cao với lượng dữ liệu lớn.

**c. Lưu trữ kết hợp (Hybrid Storage)**

Kết hợp lưu trữ cục bộ và đám mây để tận dụng lợi thế của cả hai phương pháp.

Dữ liệu quan trọng được lưu trữ cục bộ, còn dữ liệu ít truy cập được lưu trên đám mây.

##### Công nghệ lưu trữ video

**a. Hệ thống tệp (File Systems)**

**NFS (Network File System)**:

Phù hợp cho các hệ thống lưu trữ cục bộ với nhiều người dùng truy cập.

**HDFS (Hadoop Distributed File System)**:

Tối ưu cho việc lưu trữ và xử lý video trên quy mô lớn.

**b. Cơ sở dữ liệu đối tượng (Object Storage)**

Lưu trữ dữ liệu video dưới dạng các đối tượng với metadata đi kèm.

AWS S3, Google Cloud Storage, Firebase và Azure Blob Storage là các ví dụ tiêu biểu.

**Ưu điểm**:

Linh hoạt, dễ mở rộng.

Hỗ trợ tốt cho phát trực tuyến và VOD.

**c. Hệ thống lưu trữ khối (Block Storage)**

Lưu trữ dữ liệu dưới dạng các khối, thường sử dụng cho các cơ sở dữ liệu hoặc hệ thống ảo hóa.

Ví dụ: Amazon EBS, Azure Managed Disks, Firebase.

**d. Lưu trữ SSD và HDD**

**HDD (Hard Disk Drive)**:

Chi phí thấp, phù hợp cho lưu trữ video dung lượng lớn.

**SSD (Solid State Drive)**:

Tốc độ truy xuất nhanh, phù hợp cho các ứng dụng yêu cầu hiệu năng cao.

##### Các công nghệ tối ưu hóa lưu trữ video

**a. Nén video**

Sử dụng các codec nén như H.264, H.265 (HEVC), VP9, AV1 để giảm kích thước video mà không làm mất nhiều chất lượng.

Giảm dung lượng lưu trữ và băng thông truyền tải.

**b. Tích hợp CDN (Content Delivery Network)**

Tăng tốc truy cập video bằng cách lưu trữ bản sao tại các máy chủ gần người dùng.

Phù hợp với các nền tảng phát trực tuyến quy mô lớn.

**c. Chuyển mã (Transcoding)**

Tạo ra nhiều phiên bản video với các độ phân giải và bitrate khác nhau để phục vụ các thiết bị và mạng khác nhau.

**d. Sao lưu và dự phòng**

**Replication**: Sao chép dữ liệu video giữa các trung tâm dữ liệu.

**RAID**: Bảo vệ dữ liệu trong các hệ thống lưu trữ cục bộ.

##### Bảo mật và quản lý video storage

**a. Mã hóa dữ liệu**

Sử dụng AES-256 để mã hóa dữ liệu video trong quá trình lưu trữ.

**b. Quản lý truy cập**

Dùng IAM (Identity and Access Management) để giới hạn quyền truy cập.

**c. Giám sát và kiểm tra**

Theo dõi các hoạt động truy cập và sử dụng dữ liệu video để phát hiện bất thường.

##### Ứng dụng của công nghệ lưu trữ video

**a. Nền tảng VOD (Netflix, YouTube, etc.)**:

Yêu cầu lưu trữ khối lượng lớn nội dung video với khả năng truy xuất nhanh.

**b. Giám sát an ninh**:

Lưu trữ video từ camera an ninh cho mục đích theo dõi và điều tra.

**c. Phát trực tuyến**:

Lưu trữ video phục vụ cho các sự kiện trực tiếp.

**d. Lưu trữ dài hạn**:

Lưu trữ nội dung phim, tư liệu để sử dụng trong tương lai.

#### API (Application Programming Interface)

API (Application Programming Interface) là một tập hợp các giao thức, quy tắc, và công cụ cho phép các ứng dụng hoặc dịch vụ giao tiếp với nhau. Đây là nền tảng quan trọng trong việc xây dựng hệ thống phần mềm hiện đại, đặc biệt trong các kiến trúc phân tán và microservices.

##### Tổng quan về API

**a. Khái niệm API**:

API là cầu nối giữa các hệ thống, giúp các ứng dụng truy cập vào dữ liệu hoặc chức năng của một hệ thống khác mà không cần biết chi tiết triển khai bên trong.

**b. Phân loại API**:

**Public API (Open API)**: Được công khai và cho phép mọi nhà phát triển sử dụng.

**Private API**: Chỉ sử dụng trong nội bộ tổ chức.

**Partner API**: Chia sẻ với các đối tác kinh doanh cụ thể.

**Composite API**: Kết hợp nhiều API để thực hiện các thao tác phức tạp.

##### Các loại API phổ biến

**a. REST API (Representational State Transfer)**

Dựa trên giao thức HTTP, REST là loại API phổ biến nhất nhờ tính đơn giản và hiệu quả.

**Đặc điểm**:

Stateless: Không lưu trạng thái giữa các yêu cầu.

Dựa trên các phương thức HTTP như GET, POST, PUT, DELETE.

Trả về dữ liệu ở định dạng JSON hoặc XML.

**Ứng dụng**:

Giao tiếp giữa các dịch vụ web.

Tích hợp ứng dụng di động và web.

**b. SOAP API (Simple Object Access Protocol)**

Giao thức dựa trên XML để truyền tải dữ liệu.

**Đặc điểm**:

Cấu trúc chặt chẽ với các quy tắc nghiêm ngặt.

Hỗ trợ bảo mật và giao dịch phức tạp.

**Ứng dụng**:

Các hệ thống ngân hàng, tài chính yêu cầu bảo mật cao.

**c. GraphQL API**

Ngôn ngữ truy vấn dữ liệu được phát triển bởi Facebook.

**Đặc điểm**:

Linh hoạt trong việc truy xuất dữ liệu (lấy đúng những gì cần thiết).

Một endpoint duy nhất cho tất cả các truy vấn.

**Ứng dụng**:

Các ứng dụng có cấu trúc dữ liệu phức tạp.

**d. gRPC (Google Remote Procedure Call)**

Một framework mã nguồn mở của Google dựa trên giao thức RPC (Remote Procedure Call).

**Đặc điểm**:

Sử dụng giao thức HTTP/2.

Truyền dữ liệu nhị phân, tốc độ cao.

Hỗ trợ đa ngôn ngữ.

**Ứng dụng**:

Các hệ thống yêu cầu hiệu năng cao và giao tiếp thời gian thực.

**e. WebSocket API**

Giao thức giao tiếp hai chiều giữa client và server.

**Đặc điểm**:

Duy trì kết nối liên tục.

Phù hợp với các ứng dụng thời gian thực như chat, game, hay streaming.

**Ứng dụng**:

Ứng dụng chat, game online, và thông báo thời gian thực.

##### Công nghệ và công cụ phát triển API

**a. Frameworks và thư viện phổ biến**

**Node.js**: Express.js, NestJS.

**Python**: Flask, Django, FastAPI.

**Java**: Spring Boot, Micronaut.

**Ruby**: Ruby on Rails.

**PHP**: Laravel, Slim.

**Go**: Gin, Echo.

**b. Hệ sinh thái hỗ trợ**

**Postman**: Công cụ kiểm thử và phát triển API.

**Swagger/OpenAPI**:

Tiêu chuẩn mô tả và tài liệu hóa API.

Giúp tự động tạo tài liệu và kiểm thử API.

**GraphQL Playground**:

Giao diện đồ họa để tương tác với API GraphQL.

**c. Các công nghệ liên quan**

**OAuth 2.0**:

Tiêu chuẩn xác thực cho phép các ứng dụng truy cập dữ liệu của người dùng mà không cần tiết lộ mật khẩu.

**JWT (JSON Web Token)**:

Phương pháp xác thực và trao đổi thông tin giữa các bên.

**Rate Limiting**:

Hạn chế số lượng yêu cầu API để bảo vệ hệ thống khỏi quá tải.

##### Xu hướng công nghệ API hiện đại

**a. Serverless API**

Xây dựng API mà không cần quản lý máy chủ (serverless).

Dịch vụ phổ biến: AWS Lambda, Google Cloud Functions.

**b. API Gateway**

Công cụ trung gian để quản lý API.

Ví dụ: AWS API Gateway, Kong, Apigee.

**c. API First Development**

Thiết kế API trước, sau đó phát triển ứng dụng dựa trên API.

**d. Event-Driven API**

API dựa trên các sự kiện, phù hợp với các ứng dụng thời gian thực.

**e. Edge Computing APIs**

API xử lý dữ liệu ở biên (edge) để giảm độ trễ và tăng tốc độ phản hồi.

##### Lợi ích của API trong công nghệ

**a. Kết nối dịch vụ**:

API cho phép các ứng dụng hoặc dịch vụ giao tiếp và trao đổi dữ liệu dễ dàng.

**b. Tái sử dụng**:

Cung cấp khả năng tái sử dụng các chức năng hiện có mà không cần phát triển lại.

**c. Mở rộng hệ thống**:

API giúp mở rộng hệ thống một cách dễ dàng, cho phép tích hợp với nhiều dịch vụ khác.

**d. Tăng cường bảo mật**:

API cung cấp các cơ chế xác thực và phân quyền để bảo vệ dữ liệu.

##### Ứng dụng của API

**a. Phát triển ứng dụng**:

Tích hợp API từ Google Maps, Facebook, hoặc Twitter để mở rộng tính năng.

**b. IoT**:

API kết nối các thiết bị IoT với máy chủ và ứng dụng.

**c. Thương mại điện tử**:

API thanh toán như PayPal, Stripe.

**d. Truyền thông**:

API cung cấp dịch vụ nhắn tin, gọi video, hoặc streaming.

#### Frontend Programming

Frontend programming là một lĩnh vực của phát triển web và ứng dụng, tập trung vào giao diện người dùng (UI) và trải nghiệm người dùng (UX). Đây là phần mà người dùng trực tiếp tương tác, bao gồm các trang web, ứng dụng di động, và giao diện phần mềm.

##### Tổng quan về Frontend Programming

**a. Khái niệm**:

Là quá trình sử dụng các công nghệ và công cụ để tạo ra giao diện người dùng.

Mục tiêu là cung cấp trải nghiệm trực quan, nhanh chóng, và hiệu quả.

**b. Các thành phần chính**:

**Cấu trúc**: Được xây dựng bằng HTML.

**Thiết kế**: Sử dụng CSS để định dạng giao diện.

**Tương tác**: Sử dụng JavaScript và các thư viện/framework để xử lý hành động của người dùng.

##### Các công nghệ cốt lõi trong Frontend Programming

**a. HTML (HyperText Markup Language)**

**Vai trò**:

Ngôn ngữ đánh dấu để xây dựng cấu trúc cơ bản của các trang web.

**Các tính năng chính**:

Thẻ và thuộc tính dùng để tạo ra các thành phần như tiêu đề, đoạn văn, liên kết, bảng, và biểu mẫu.

Hỗ trợ các tiêu chuẩn mới như HTML5 với các thành phần đa phương tiện như video, audio, canvas.

**b. CSS (Cascading Style Sheets)**

**Vai trò**:

Cung cấp định dạng và thiết kế cho các thành phần HTML.

**Các tính năng chính**:

Tạo kiểu (màu sắc, font chữ, kích thước).

Bố cục (flexbox, grid).

Hiệu ứng động (animation, transition).

**Công nghệ nâng cao**:

**CSS Preprocessors**: SASS, LESS.

**CSS Frameworks**: Bootstrap, TailwindCSS.

**c. JavaScript (JS)**

**Vai trò**:

Thêm tính tương tác vào giao diện người dùng.

**Các tính năng chính**:

DOM manipulation: Thay đổi nội dung, kiểu dáng, hoặc cấu trúc của trang.

Sự kiện: Xử lý click, hover, kéo thả.

Giao tiếp với backend qua API (AJAX, Fetch).

**Frameworks và Libraries**:

**React.js**: Xây dựng giao diện người dùng bằng cách sử dụng component.

**Vue.js**: Framework nhẹ, đơn giản và linh hoạt.

**Angular**: Framework mạnh mẽ cho các ứng dụng phức tạp.

##### Các công cụ và công nghệ hỗ trợ

**a. Build Tools**

**Webpack**: Bundler phổ biến, gom và tối ưu mã nguồn.

**Vite**: Công cụ build nhanh và tối ưu cho các dự án hiện đại.

**Parcel**: Bundler zero-config đơn giản.

**b. Task Runners**

**Gulp**: Tự động hóa các tác vụ như minify, compile.

**npm Scripts**: Chạy các lệnh thông qua tệp package.json.

**c. Version Control Systems**

**Git**: Hệ thống quản lý phiên bản phổ biến.

**GitHub/GitLab/Bitbucket**: Lưu trữ mã nguồn và hợp tác.

**d. Package Managers**

**npm (Node Package Manager)**: Cài đặt và quản lý thư viện JavaScript.

**yarn**: Thay thế npm với hiệu suất cao hơn.

**pnpm**: Quản lý hiệu quả không gian lưu trữ.

**e. Testing Tools**

**Jest**: Testing framework phổ biến cho JavaScript.

**Cypress**: Kiểm tra giao diện và chức năng của ứng dụng.

**Storybook**: Phát triển và kiểm tra các component UI.

##### Kiến trúc Frontend hiện đại

**a. Component-Based Architecture**

Tập trung vào việc xây dựng ứng dụng từ các thành phần (component) tái sử dụng được.

Framework/Library tiêu biểu: React, Vue, Angular.

**b. Single Page Application (SPA)**

Ứng dụng tải một lần và cập nhật nội dung động thông qua JavaScript.

**Ưu điểm**: Nhanh chóng, trải nghiệm người dùng tốt.

**Nhược điểm**: SEO khó hơn so với SSR.

**c. Server-Side Rendering (SSR)**

Render HTML trên máy chủ và gửi về trình duyệt.

**Framework tiêu biểu**: Next.js, Nuxt.js.

**Lợi ích**: Cải thiện SEO và thời gian tải trang.

**d. Progressive Web Apps (PWA)**

Ứng dụng web có tính năng như ứng dụng di động.

Hỗ trợ offline, push notifications, và cài đặt trên thiết bị.

##### Xu hướng công nghệ Frontend hiện đại

**a. Micro-Frontend**

Phân tách ứng dụng lớn thành các phần nhỏ độc lập.

Dễ quản lý và triển khai.

**b. Static Site Generators (SSG)**

Tạo trang tĩnh từ trước và phân phối qua CDN.

Công cụ phổ biến: Gatsby, Next.js, Hugo.

**c. State Management**

Quản lý trạng thái ứng dụng với Redux, MobX, Recoil.

**d. Responsive Design**

Thiết kế giao diện tương thích với mọi kích thước màn hình.

Công cụ hỗ trợ: Media Queries, Flexbox, CSS Grid.

**e. Headless CMS**

Tích hợp nội dung qua API mà không ràng buộc với frontend cụ thể.

Ví dụ: Strapi, Contentful, Sanity.

##### Thách thức trong Frontend Programming

**a. Hiệu năng**:

Giảm tải tài nguyên không cần thiết (lazy load).

Tối ưu hóa mã nguồn (minify, tree-shaking).

**b. Tương thích trình duyệt**:

Xử lý khác biệt giữa các trình duyệt (polyfills).

**c. Bảo mật**:

Ngăn chặn tấn công XSS, CSRF.

**d. Khả năng mở rộng**:

Thiết kế hệ thống dễ dàng mở rộng khi ứng dụng phát triển.

##### Ứng dụng thực tế

**a. Thương mại điện tử**:

Shopee, Tiki sử dụng React, Angular.

**b. Mạng xã hội**:

Facebook (React), Twitter (Node.js, React).

**c. Hệ thống quản lý nội dung (CMS)**:

WordPress, Ghost.

**d. Truyền thông & Streaming**:

Netflix, YouTube sử dụng PWA và SSR

### Các công nghệ sử dụng

#### Node.js



**Hình 3.5** **Hệ thống phần mềm Node.js**

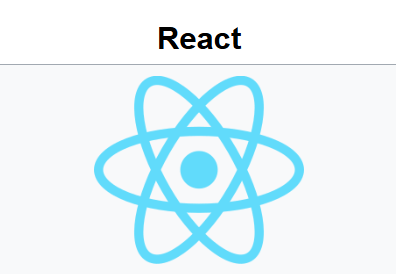
Node.js là một hệ thống phần mềm được thiết kế để viết các ứng dụng internet có khả năng mở rộng, đặc biệt là máy chủ web. Chương trình được viết bằng JavaScript, sử dụng kỹ thuật điều khiển theo sự kiện, nhập/xuất không đồng bộ để tối thiểu tổng chi phí và tối đa khả năng mở rộng. Node.js bao gồm có V8 JavaScript engine của Google, libUV, và vài thư viện khác.

Node.js được tạo bởi Ryan Dahl từ năm 2009, và phát triển dưới sự bảo trợ của Joyent.

Mục tiêu ban đầu của Dahl là làm cho trang web có khả năng push như trong một số ứng dụng web như Gmail. Sau khi thử với vài ngôn ngữ Dahl chọn Javascript vì một API Nhập/Xuất không đầy đủ. Điều này cho phép anh có thể định nghĩa một quy ước Nhập/Xuất điểu khiển theo sự kiện, non-blocking.

Vài môi trường tương tự được viết trong các ngôn ngữ khác bao gồm Twisted cho Python, Perl Object Environment cho Perl, libevent cho C và EventMachine cho Ruby. Khác với hầu hết các chương trình Javascript, Nodejs không chạy trên một trình duyệt mà chạy trên Server. Node.js sử dụng nhiều chi tiết kỹ thuật của CommonJS. Nó cung cấp một môi trường REPL cho kiểm thử tương tác.

#### React.js



Hình . Thư viện React

React (hay còn được gọi là React.js hoặc ReactJS) là một thư viện JavaScript front-end mã nguồn mở và miễn phí để xây dựng giao diện người dùng dựa trên các thành phần UI riêng lẻ.

Nó được phát triển và duy trì bởi Meta (trước đây là Facebook) và cộng đồng các nhà phát triển và công ty cá nhân. React có thể được sử dụng làm cơ sở để phát triển các ứng dụng SPA (Single page application), thiết bị di động hoặc ứng dụng được kết xuất bằng máy chủ với các thư viện khác như Next.js.

Tuy nhiên, React chỉ hướng tới việc quản lý trạng thái và hiển thị trạng thái đó cho DOM, vì vậy việc tạo ứng dụng bằng React thường yêu cầu sử dụng thêm các thư viện bổ sung để thực hiện định tuyến trang, cũng như thêm một số chức năng ở phía máy khách.

#### NestJS



**Hình 3.7 Framework NestJS**

Nest (NestJS) là một khuôn khổ để xây dựng các ứng dụng Node.js các ứng dụng phía máy chủ. Nó sử dụng JavaScript lũy tiến, được xây dựng và hỗ trợ đầy đủ TypeScript (nhưng vẫn cho phép các nhà phát triển viết mã bằng JavaScript thuần túy) và kết hợp các yếu tố của OOP (Lập trình hướng đối tượng), FP (Lập trình chức năng) và FRP (Lập trình phản ứng chức năng).

Về cơ bản, Nest sử dụng các khung Máy chủ HTTP mạnh mẽ như Express (mặc định) và có thể được cấu hình tùy chọn để sử dụng Fastify!

Nest cung cấp mức độ trừu tượng trên các framework phổ biến của Node.js (Express/Fastify), nhưng cũng hiển thị API của họ trực tiếp cho nhà phát triển. Điều này cho phép các nhà phát triển tự do sử dụng vô số module của bên thứ ba có sẵn cho nền tảng cơ bản.

### Các package quan trọng được cài đặt

**a**. **node-media-server**

**node-media-server (NMS)** là một thư viện mã nguồn mở được xây dựng bằng Node.js, cung cấp giải pháp đơn giản và mạnh mẽ để thiết lập và quản lý các dịch vụ phát trực tuyến (streaming). NMS hỗ trợ các giao thức phổ biến như **RTMP, HLS và** **FLV** phù hợp cho các ứng dụng livestream, giám sát video và truyền phát nội dung thời gian thực.

**b. firebase-admin**

**firebase-admin** là thư viện chính thức của Firebase được phát triển dành cho môi trường máy chủ (server-side), giúp bạn quản lý và tương tác với các dịch vụ Firebase trong các ứng dụng Node.js. Nó cung cấp các API mạnh mẽ để thực hiện các tác vụ quản trị Firebase như xác thực người dùng, quản lý cơ sở dữ liệu, gửi thông báo đẩy và nhiều tính năng khác.

**c. nodemailer**

Nodemailer là một thư viện Node.js giúp gửi email từ ứng dụng web. Nó hỗ trợ nhiều phương thức gửi email khác nhau, chẳng hạn như sử dụng SMTP, gửi qua các dịch vụ email như Gmail, hoặc dịch vụ email tùy chỉnh. Nodemailer rất phổ biến trong việc tích hợp gửi email tự động, ví dụ như thông báo xác nhận đăng ký, thay đổi mật khẩu, hoặc thông báo hệ thống.

**d. pg**

Thư viện Node.js hỗ trợ kết nối và tương tác với database PostgreSQL.

**e. passport-jwt**

Thư viện hỗ trợ xác thực user.

**f. prisma**

Prisma là một ORM (Object-Relational Mapping) hiện đại dành cho Node.js và TypeScript, giúp bạn làm việc với cơ sở dữ liệu dễ dàng và hiệu quả. Prisma hỗ trợ nhiều loại cơ sở dữ liệu, bao gồm PostgreSQL, MySQL, SQLite, SQL Server và MongoDB. Prisma giúp bạn tương tác với cơ sở dữ liệu thông qua các API mạnh mẽ và kiểu dữ liệu an toàn của TypeScript, thay vì phải viết các câu lệnh SQL thủ công.

**g. React**

React là một thư viện JavaScript mã nguồn mở, được phát triển và duy trì bởi Facebook (nay là Meta), dùng để xây dựng các giao diện người dùng (UI) cho các ứng dụng web, đặc biệt là các ứng dụng một trang (SPA - Single Page Applications). React giúp việc phát triển giao diện người dùng trở nên dễ dàng hơn thông qua việc phân chia giao diện thành các thành phần (components) độc lập và tái sử dụng.

**h. flv.js**

**FLV.js** là một thư viện JavaScript mã nguồn mở, cho phép trình duyệt web phát video FLV (Flash Video) mà không cần sử dụng plugin Flash Player. FLV.js sử dụng HTML5 <video> element để phát video, giúp người dùng xem các video FLV ngay trong trình duyệt mà không cần cài đặt thêm phần mềm.

**i. hls.js**

**HLS.js** là một thư viện JavaScript mã nguồn mở, giúp phát video HLS (HTTP Live Streaming) trên các trình duyệt web hỗ trợ HTML5. HLS là một giao thức streaming phổ biến do Apple phát triển, cho phép phát video theo dạng phân đoạn nhỏ và chuyển tiếp mượt mà trong các điều kiện băng thông thay đổi. Tuy nhiên, không phải tất cả trình duyệt hỗ trợ HLS natively, đặc biệt là trên các nền tảng không phải của Apple. HLS.js giải quyết vấn đề này bằng cách cung cấp một giải pháp thay thế cho việc phát video HLS trên các trình duyệt không hỗ trợ trực tiếp.

**j. Axios**

**Axios** là một thư viện JavaScript dùng để thực hiện các yêu cầu HTTP từ trình duyệt hoặc Node.js. Axios giúp bạn giao tiếp với API, tải dữ liệu từ máy chủ, hoặc gửi dữ liệu đến máy chủ một cách dễ dàng. Nó hỗ trợ các phương thức HTTP như GET, POST, PUT, DELETE v.v. và cung cấp các tính năng như quản lý timeout, xử lý dữ liệu JSON tự động, hoặc cấu hình các request một cách linh hoạt.

**i. Leaflet**

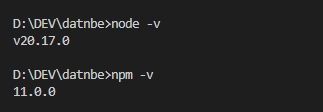
**Leaflet** là một thư viện JavaScript mã nguồn mở dùng để xây dựng bản đồ tương tác trên web. Đây là một công cụ gọn nhẹ, dễ sử dụng, và rất phổ biến trong việc tích hợp bản đồ vào các ứng dụng web. Leaflet hỗ trợ các tính năng cơ bản và nâng cao của bản đồ, đồng thời có thể được mở rộng với nhiều plugin để bổ sung chức năng.

## Cài đặt môi trường phát triển

### Cài đặt cơ bản

**Node.js**: Tải và cài đặt Node.js. Phiên bản khuyến nghị: LTS.

Kiểm tra cài đặt:

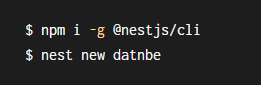


Hình 3.8 Kiểm tra version Node và npm

### Khởi tạo dự án

#### Backend

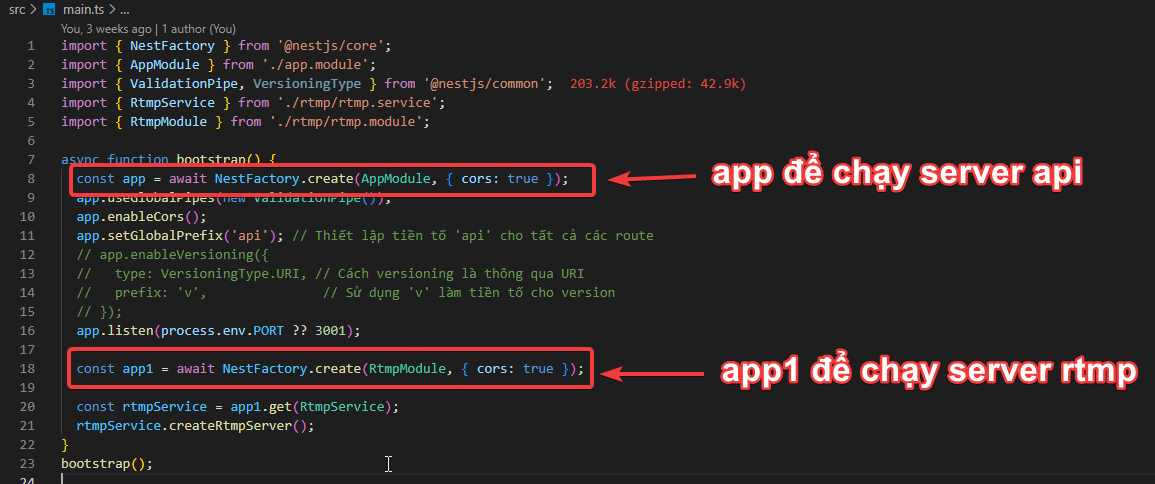
Khởi tạo project backend với CLI của framework NestJS



Hình . Cài đặt & triển khai backend NestJS

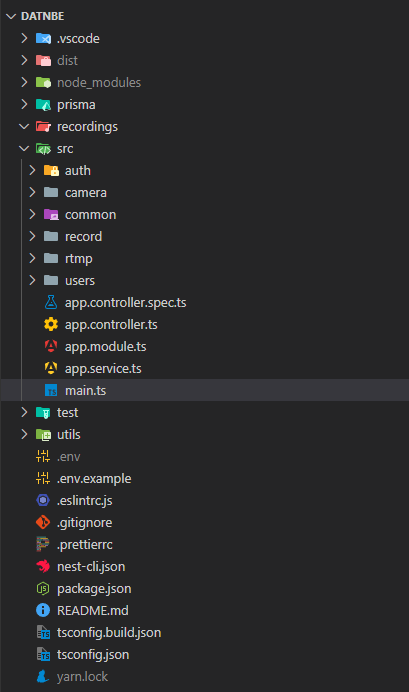
Phần Backend được chia làm 2 phần lớn:

* Server Backend (cung cấp các API phục vụ việc thao tác với database)
* Server RTMP (phục vụ truyền tải các video stream)



**Hình 3.10 Cấu hình server API và server RTMP**

Project backend hoàn chỉnh:



Hình . Cây thư mục hoàn chỉnh của backend

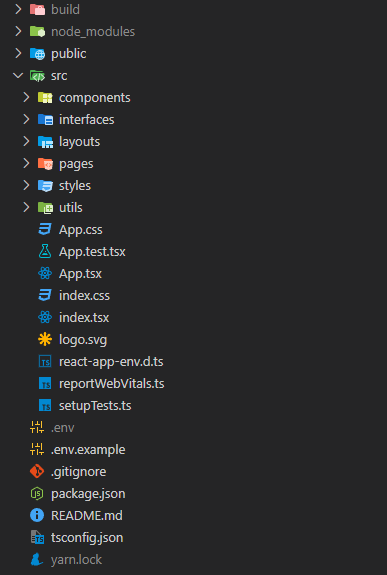
#### Frontend

Tạo project frontend với thư viện React.js sử dụng Typescript:



**Hình 3.12 Cài đặt & triển khai frontend ReactJS**

Thư mục hoàn chỉnh của project:

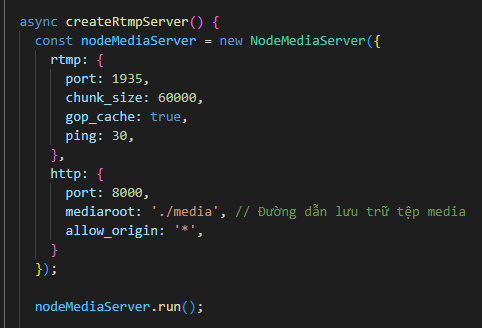


**Hình 3.13 Cây thư mục hoàn chỉnh của frontend**

## Xây dựng các thành phần chính

### RTMP Server

Khởi tạo server:



Hình . Khởi tạo RTMP Server

Node-Media-Server được sử dụng để quản lý các luồng video. Cấu hình cho phép:

* Phát video qua giao thức RTMP, HLS và FLV.
* Lưu trữ video cục bộ hoặc trên dịch vụ đám mây.

### API Server

#### Xây dựng các module chính

* UserModule
* RecordModule
* CameraModule

#### Xây dựng phân quyền và bảo mật

* Phân quyền Admin và User
* Bảo mật bằng xác thực thông qua JSON Web Token (JWT)
* Mã hoá dữ liệu với bcrypt

#### Lưu trữ video với Firebase

Video được lưu trữ trên Firebase. Metadata được lưu trong PostgreSQL.

### Frontend

#### Xây dựng các components

* Auth
* Header
* LiveStreamPlayer
* LocationMarker
* Popup
* Sidebar

#### Xây dựng layouts

* Default layout
* Landing layout

#### Xây dựng các pages

* AdminDashboard
* Camera
* LoginPage
* Map
* ProfilePage
* SignupPage
* VSM
* RecordPage

## Tích hợp và kiểm thử

### Tích hợp

Hệ thống tích hợp chạy song song Backend(NestJS) và Frontend(ReactJS)

Hiện tại hệ thống đang chạy ở phía local.

Trong tương lai có thể triển khai hệ thống này lên Hosting nào đó

Dùng docker để đóng gói và thuận tiện cho việc triển khai

### Kiểm thử

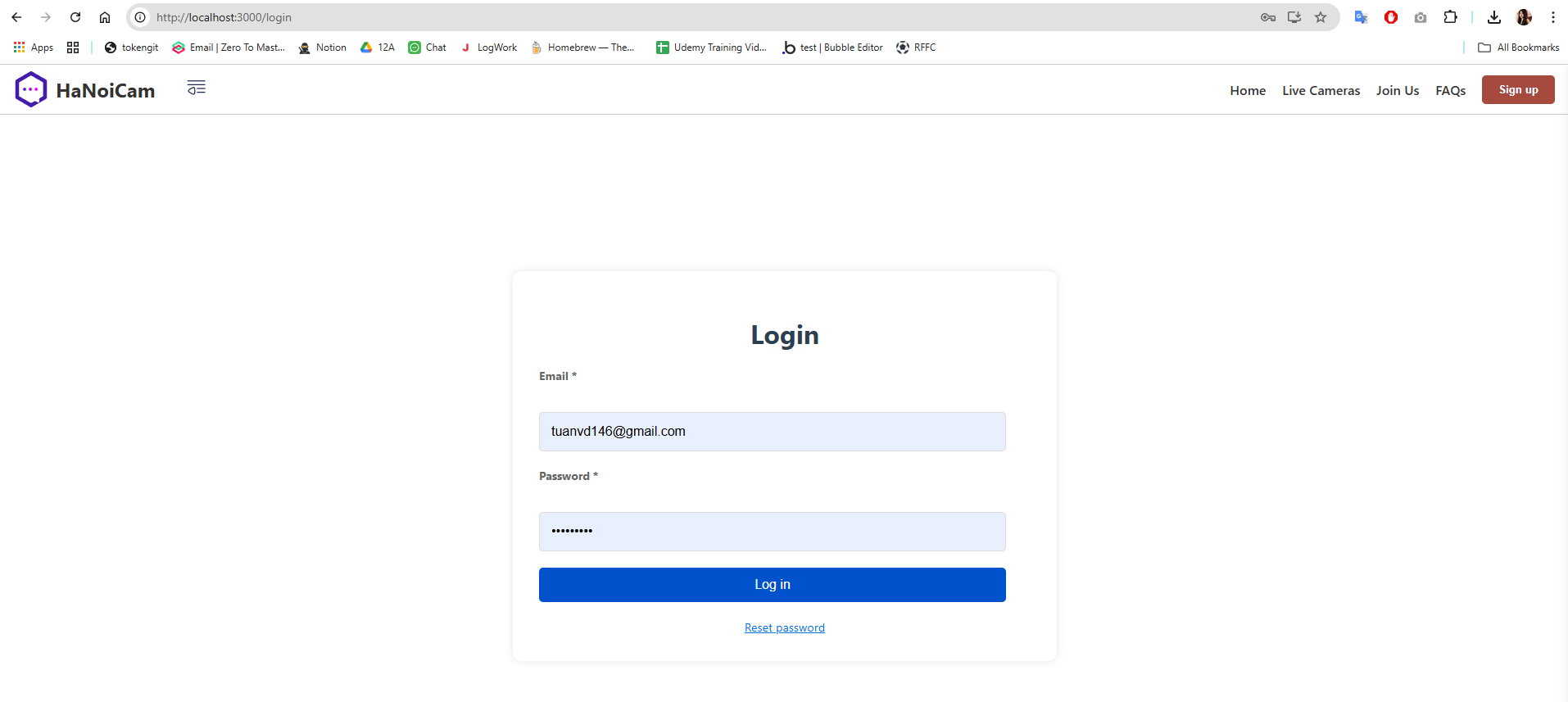
* Kiểm thử chức năng API bằng Postman.
* Kiểm thử giao diện bằng Cypress hoặc Jest.

# KẾT QUẢ VÀ ĐÁNH GIÁ

Phần này trình bày các kết quả đạt được trong quá trình triển khai bao gồm đánh giá hiệu quả, mức độ đáp ứng yêu cầu, và những điểm cần cải thiện.

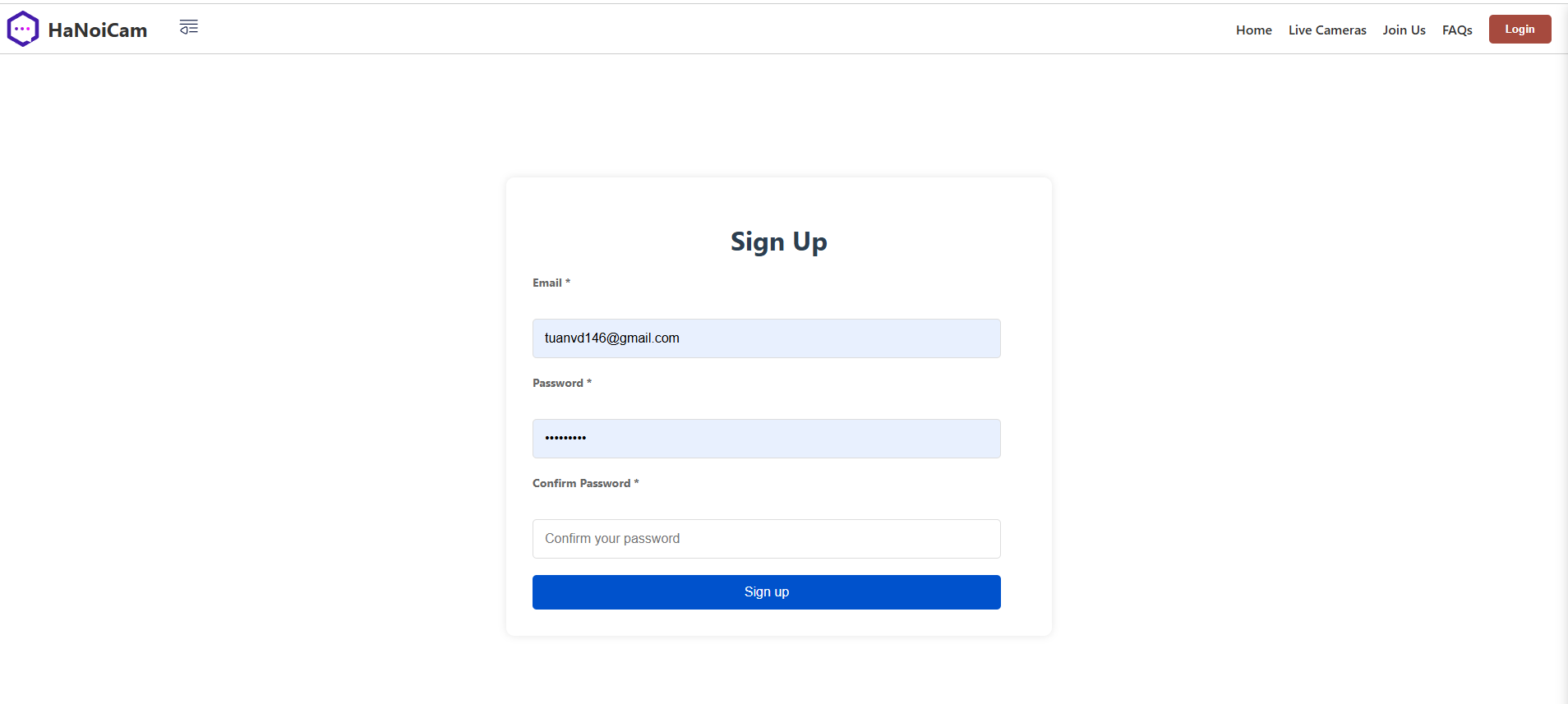
## Giao diện sản phẩm

a. Màn hình đăng nhập



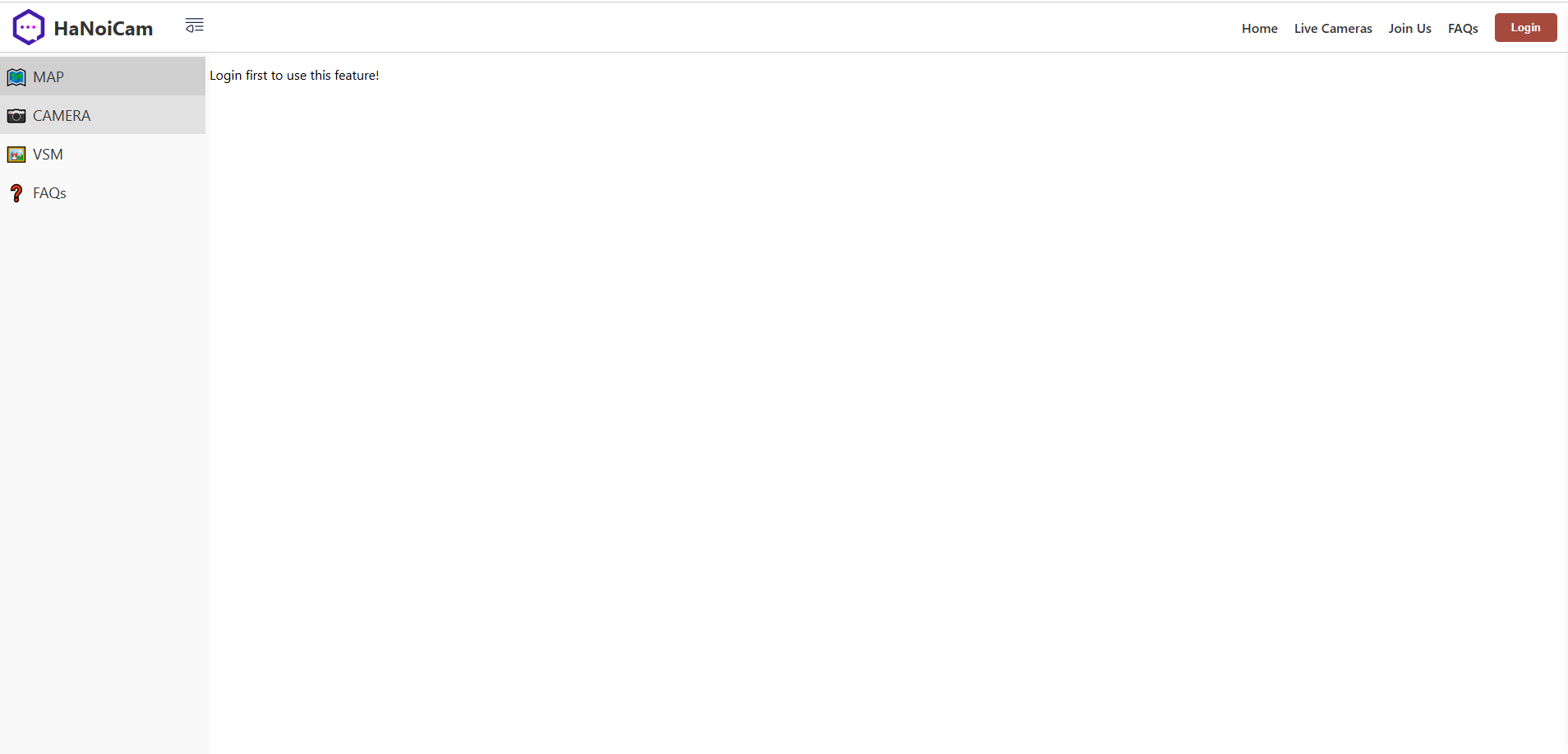
Hình . Giao diện màn Login

b. Màn hình đăng kí



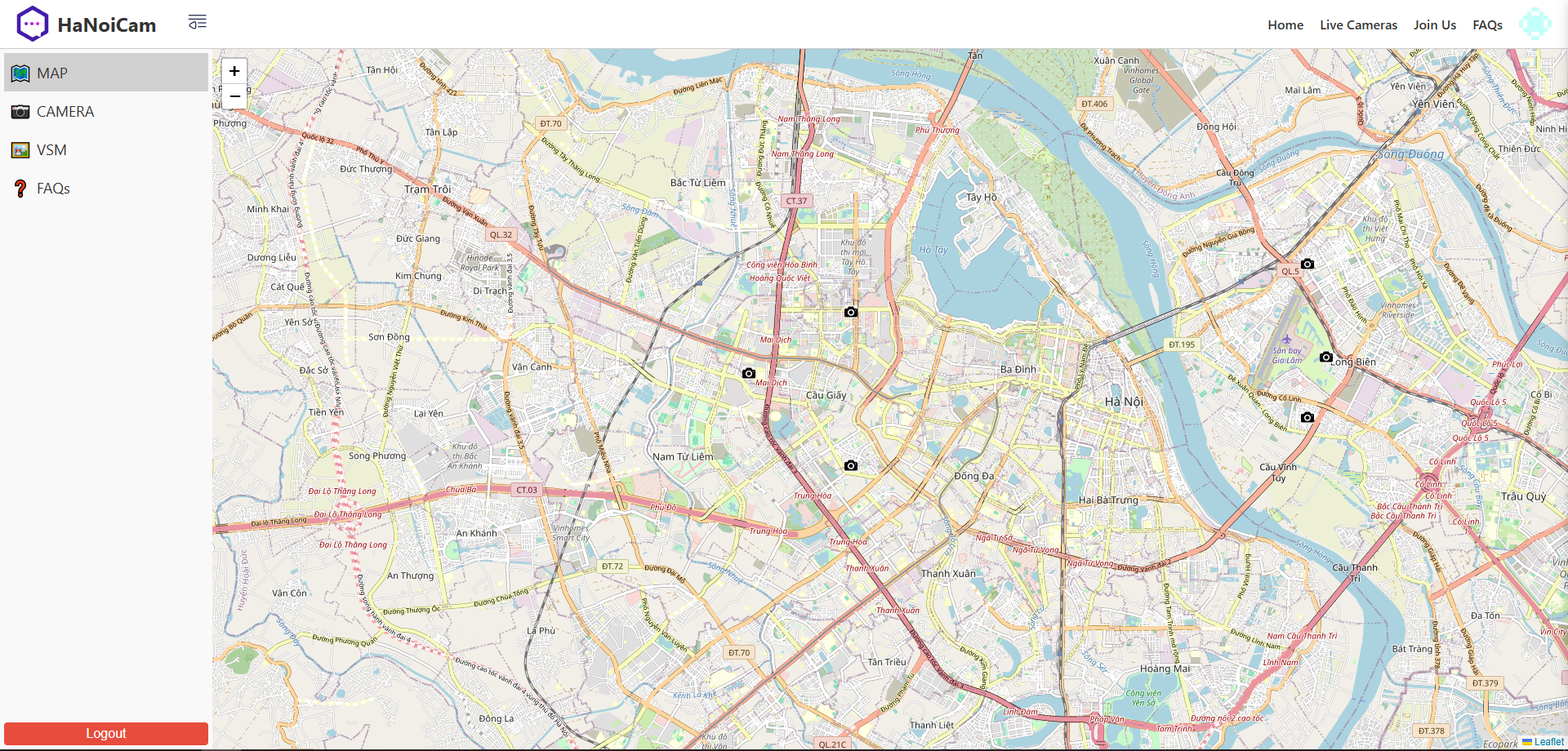
Hình . Giao diện màn Signup

c. Màn hình mặc định khi chưa đăng nhập:



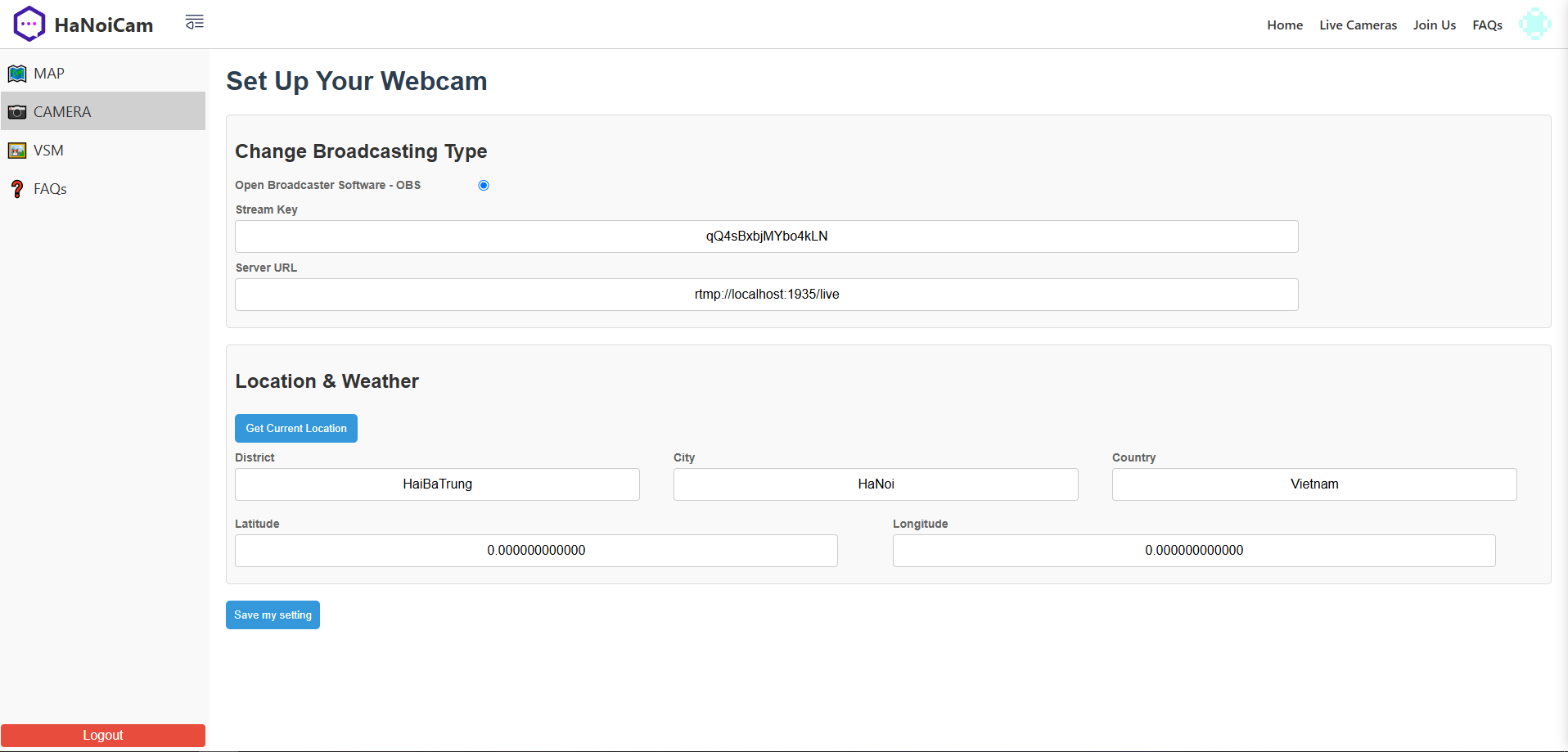
Hình . Giao diện mặc định khi chưa đăng nhập

d. Màn hình mặc định khi đã đăng nhập



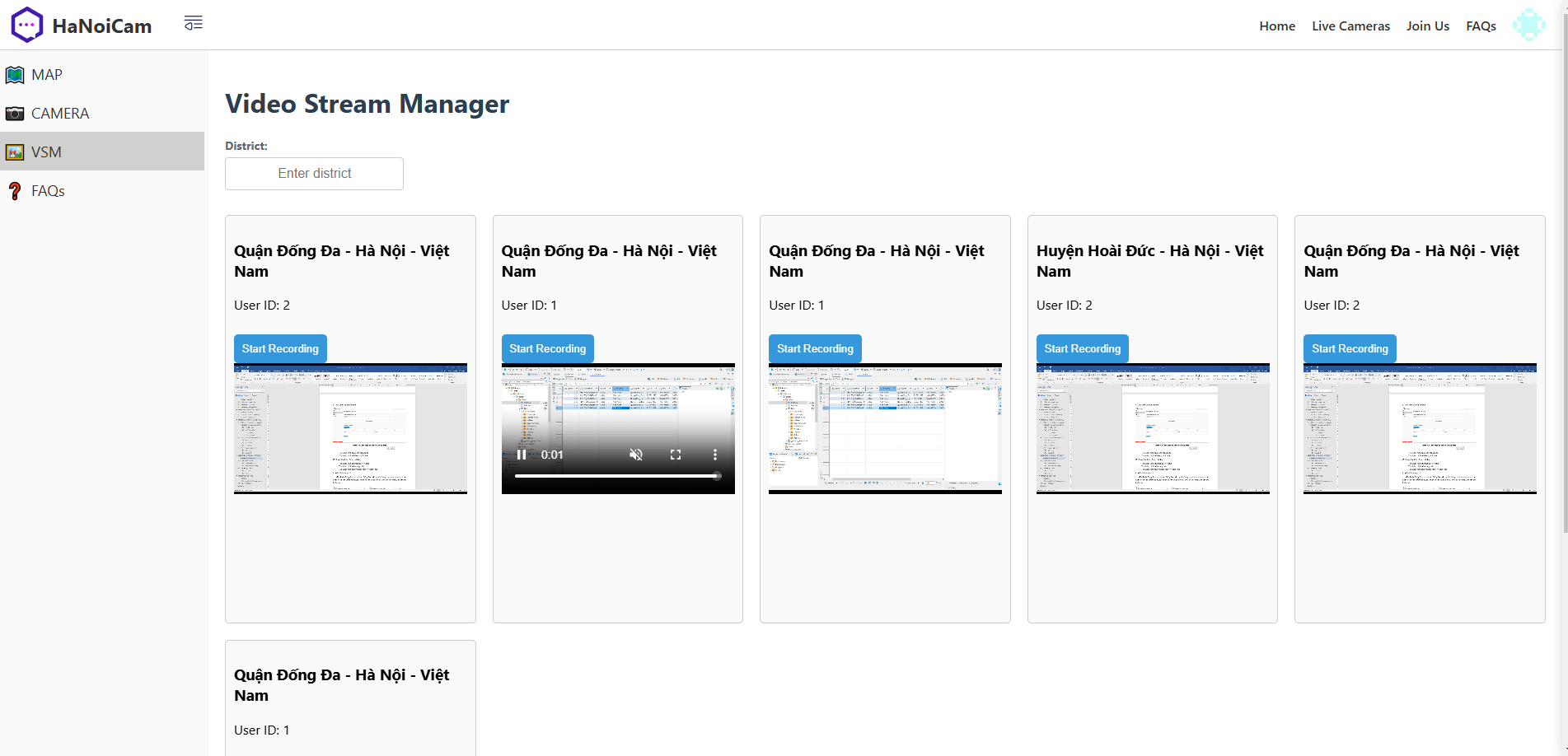
Hình . Giao diện mặc định khi đã đăng nhập

e. Màn hình setting camera



Hình . Màn hình setting camera

f. Màn hình quản lý Camera



Hình . Màn hình quản lý camera

## Kết quả đạt được

### Tính năng chính

 **Quản lý người dùng**:

* Người dùng có thể đăng nhập và đăng xuất với hệ thống xác thực an toàn.
* Phân quyền người dùng (quản trị viên và người dùng thông thường) hoạt động ổn định.

 **Quản lý camera**:

* Thành công trong việc thêm, sửa, xóa và xem danh sách các camera.
* Cung cấp thông tin chi tiết về từng camera như vị trí, trạng thái kết nối.

 **Quản lý video stream**:

* Phát trực tiếp (livestream) thành công từ các camera thông qua giao thức RTMP (FLV).
* Cho phép dừng và bắt đầu luồng video từ giao diện người dùng hoặc backend.
* Hỗ trợ nhiều camera phát cùng lúc với chất lượng ổn định.

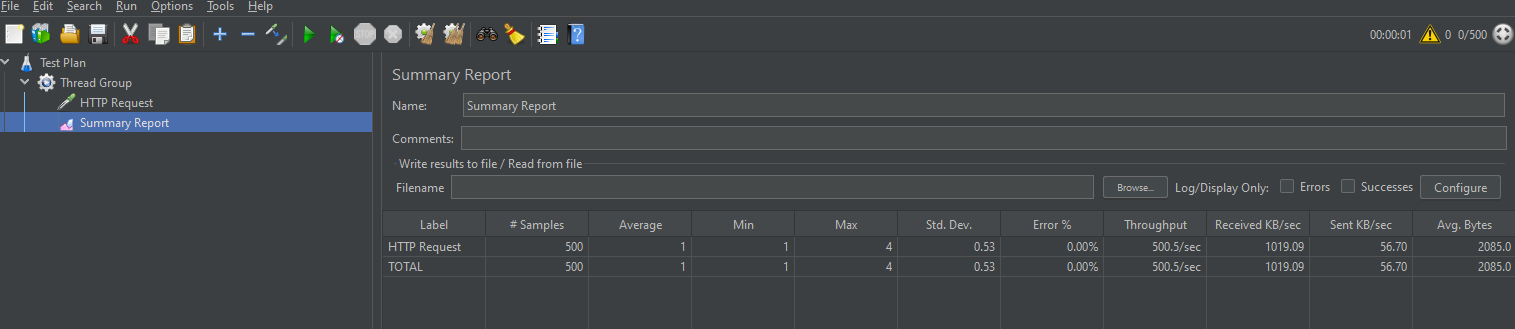
 **Tích hợp bản đồ**:

* Hiển thị vị trí các camera trên bản đồ tương tác.
* Người dùng có thể tìm kiếm và lọc camera theo vị trí địa lý.

### Hiệu suất hệ thống

 **Tải hệ thống**:

* Xử lý đồng thời 6 luồng video từ các camera với mức độ trễ không đáng kể.
* Đáp ứng tốt 500 yêu cầu API mỗi giây với mức tải trung bình (sử dụng công cụ kiểm thử hiệu năng Apache JMeter).



Hình . Test tải API với Apache JMeter

 **Độ ổn định**:

* Hệ thống hoạt động liên tục trong 72 giờ mà không xảy ra lỗi lớn.

## Đánh giá kết quả

### Ưu điểm

 **Đáp ứng yêu cầu**:

* Hệ thống đã đáp ứng được hầu hết các yêu cầu đề ra, từ quản lý camera đến phát trực tiếp.
* Giao diện người dùng trực quan và dễ sử dụng.

 **Tính mở rộng**:

* Thiết kế hệ thống theo hướng module hóa giúp dễ dàng mở rộng trong tương lai (ví dụ: hỗ trợ thêm các giao thức truyền tải video khác như WebRTC).

 **Công nghệ hiện đại**:

* Sử dụng Node.js, React, và Node-Media-Server đảm bảo hiệu suất cao và dễ bảo trì.
* Tích hợp bản đồ giúp tăng trải nghiệm người dùng.

### Hạn chế

 **Hiệu năng**:

* Khi số lượng camera vượt quá 6, hệ thống bắt đầu xuất hiện độ trễ trong xử lý video.
* Khi số lượng request API quá lớn (> 1000 req/s) sẽ khiến Server bị đơ lag
* Không thể load được hết dữ liệu từ các camera.

 **Tính năng còn thiếu**:

* Chưa hỗ trợ điều chỉnh tốc độ các luồng video đã phát.
* Chưa tích hợp thông báo real-time khi có sự cố xảy ra với camera.

 **Bảo mật**:

* Cần cải thiện thêm cơ chế bảo mật, như mã hóa dữ liệu khi truyền tải giữa frontend và backend.
* Hệ thống xác thực người dùng cần tích hợp xác thực hai yếu tố (2FA) để tăng cường bảo mật.

## Đề xuất cải thiện

### ****Nâng cao hiệu năng****

* Tối ưu hóa truy vấn cơ sở dữ liệu để xử lý khối lượng lớn dữ liệu nhanh hơn.
* Triển khai cơ chế cân bằng tải (load balancing) để hỗ trợ nhiều luồng video cùng lúc.

### ****Bổ sung tính năng****

* Hỗ trợ lưu trữ và phát lại luồng video.
* Tích hợp cảnh báo tự động khi hệ thống phát hiện sự cố từ camera (offline, mất kết nối).
* Xây dựng ứng dụng di động để người dùng tiện theo dõi camera mọi lúc, mọi nơi.

### ****Cải thiện bảo mật****

* Sử dụng HTTPS và chứng chỉ SSL để bảo mật dữ liệu truyền tải.
* Áp dụng cơ chế xác thực OAuth2.0 hoặc OpenID Connect cho hệ thống đăng nhập.

## Đánh giá tổng quan

 **Hiệu quả ứng dụng thực tế**: Hệ thống đã chứng minh được khả năng quản lý hiệu quả các camera trong một thành phố lớn như Hà Nội. Việc tích hợp bản đồ và khả năng phát trực tiếp giúp hệ thống trở thành một công cụ đắc lực trong việc giám sát và quản lý an ninh đô thị.

 **Khả năng mở rộng**: Hệ thống được thiết kế với kiến trúc linh hoạt, dễ dàng nâng cấp và tích hợp thêm các chức năng mới.

 **Tính khả thi**: Với những điều chỉnh và cải tiến nhỏ, hệ thống có tiềm năng triển khai thực tế để quản lý hàng nghìn camera tại các thành phố lớn.

# TỔNG KẾT

## Kết luận

Sau khoảng thời gian tích cực tham gia nghiên cứu với đề tài “Thiết kế và thực hiện giao diện (frontend) cho nền tảng giám sát thành phố thông minh qua hình ảnh”, em đã đạt được một số kết quả như yêu cầu đã đặt ra từ ban đầu. Phần mềm đã có thể đăng kí, đăng nhập, thêm camera, quản lý camera, record video, phát trực tiếp luồng video từ camera và tích hợp bản đồ tương tác.

Hệ thống được xây dựng dựa trên các công nghệ hiện đại như **Node.js, React, Node-Media-Server** giúp tối ưu hóa hiệu suất và tăng tính linh hoạt. Các chức năng chính của hệ thống đã hoạt động ổn định, đáp ứng được khoảng 80% các yêu cầu trong điều kiện thực nghiệm

Về hiệu quả ứng dụng, hệ thống đã chứng minh tiềm năng trong việc giám sát giao thông, an ninh đô thị và quản lý cơ sở hạ tầng thông qua việc quản lý và giám sát trực tiếp các camera tại các vị trí quan trọng.

Hệ thống hiện tại không chỉ phục vụ cho mục đích giám sát mà còn có thể mở rộng để tích hợp các tính năng nâng cao như phân tích hình ảnh, nhận diện sự cố và cảnh báo tự động.

Em hy vọng hệ thống này có thể góp phần nâng cao hiệu quả quản lý trong các thành phố lớn, hỗ trợ các cơ quan chức năng và các tổ chức nghiên cứu liên quan.

## Hướng phát triển trong tương lai

Dựa trên nền tảng hiện tại, hệ thống Video Stream Manager có thể được phát triển thêm các tính năng và cải tiến sau:

1. **Tích hợp trí tuệ nhân tạo (AI)**:
   * Ứng dụng AI để nhận diện khuôn mặt, phương tiện, hoặc các tình huống bất thường trong luồng video.
   * Phân tích dữ liệu real-time để đưa ra cảnh báo tự động khi phát hiện các sự cố hoặc hành vi đáng ngờ.
2. **Nâng cao tính bảo mật**:
   * Tích hợp xác thực hai yếu tố (2FA) và mã hóa dữ liệu để tăng cường an toàn thông tin.
   * Đưa vào cơ chế phát hiện và ngăn chặn các tấn công mạng phổ biến như DDoS.
3. **Tăng cường khả năng mở rộng**:
   * Tối ưu hóa hệ thống để hỗ trợ hơn 1000 camera hoạt động đồng thời.
   * Triển khai cơ chế cân bằng tải (load balancing) để đảm bảo hiệu suất ổn định khi hệ thống mở rộng.
4. **Phát triển ứng dụng di động**:
   * Cung cấp ứng dụng dành cho Android và iOS để người dùng có thể theo dõi luồng video mọi lúc, mọi nơi.
   * Hỗ trợ thông báo đẩy (push notification) để cảnh báo real-time.
5. **Nghiên cứu và phát triển thêm tính năng bản địa hóa**:
   * Hỗ trợ ngôn ngữ tiếng Việt trong giao diện và các mô hình phân tích, đảm bảo hệ thống phù hợp với nhu cầu sử dụng tại Việt Nam.

Một lần nữa, em xin gửi lời cảm ơn sâu sắc tới **PGS. GVCC Nguyễn Tài Hưng**, cùng các thầy cô trong khoa và bạn bè đã nhiệt tình hỗ trợ em trong suốt quá trình nghiên cứu và triển khai đề tài này. Những kinh nghiệm quý báu từ quá trình thực hiện sẽ là hành trang quan trọng để em tiếp tục phát triển các dự án công nghệ trong tương lai.

# TÀI LIỆU THAM KHẢO

[1] <https://docs.nestjs.com>

[2] <https://react.dev/learn>

[3] <https://viblo.asia/p/jmeter-phan-1-apache-jmeter-la-gi-huong-dan-cai-dat-apache-jmeter-Az45bbew5xY>

[4] <https://www.earthcam.com/myearthcam>

[5] <https://vi.wikipedia.org/wiki/Visual_Studio_Code>

[6] <https://en.wikipedia.org/wiki/DBeaver>

[7] <https://en.wikipedia.org/wiki/Postman_(software)>

[8] <https://vi.wikipedia.org/wiki/Node.js>

[9] <https://vi.wikipedia.org/wiki/OBS_Studio>

[10] [https://en.wikipedia.org/wiki/Firebase](https://en.wikipedia.org/wiki/Firebase%20)

# PHỤ LỤC

Về source code của project em đã tổng hợp trên github:

<https://github.com/kurovu146/DATN2024>

<https://github.com/kurovu146/BEDATN2024>