**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**

**LÊ THỊ THU HUYỀN**

**PHAN ÁI LINH**

**KHÓA LUẬN TỐT NGHIỆP**

**NGHIÊN CỨU TÍCH HỢP CÔNG NGHỆ TÍNH TOÁN BIÊN CHO THIẾT BỊ BỤC GIẢNG TRONG LỚP HỌC HYBRID**

**Researching the intergration of edge computing technology for lectern devices in the hybrid classroom**

**KỸ SƯ NGÀNH KỸ THUẬT MÁY TÍNH**

**TP. HỒ CHÍ MINH, 2024**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**

**LÊ THỊ THU HUYỀN - 20521422**

**PHAN ÁI LINH – 20520614**

**KHÓA LUẬN TỐT NGHIỆP**

**NGHIÊN CỨU TÍCH HỢP CÔNG NGHỆ TÍNH TOÁN BIÊN CHO THIẾT BỊ BỤC GIẢNG TRONG LỚP HỌC HYBRID**

**Researching the intergration of edge computing technology for lectern devices in the hybrid classroom**

**KỸ SƯ NGÀNH KỸ THUẬT MÁY TÍNH**

**GIẢNG VIÊN HƯỚNG DẪN**

**TS. NGUYỄN MINH SƠN**

**TP. HỒ CHÍ MINH, 2024**

THÔNG TIN HỘI ĐỒNG CHẤM KHÓA LUẬN TỐT NGHIỆP

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số **728/QĐ-ĐHCNTT** ngày 20 tháng 07 năm 2023 của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

MỤC LỤC

[Chương 1. GIỚI THIỆU ĐỀ TÀI 3](#_Toc170461623)

[1.1. Tổng quan về đề tài 3](#_Toc170461624)

[1.1.1. Các nghiên cứu quốc tế 3](#_Toc170461625)

[1.1.2. Các nghiên cứu trong nước 4](#_Toc170461626)

[1.2. Mục tiêu của đề tài 5](#_Toc170461627)

[1.3. Giới hạn nghiên cứu của đề tài 5](#_Toc170461628)

[1.4. Kết quả mong đợi của đề tài 6](#_Toc170461629)

[Chương 2. TÌM HIỂU LÝ THUYẾT VÀ LỰA CHỌN CÔNG NGHỆ 7](#_Toc170461630)

[2.1. Công nghệ lập trình phần mềm nhúng 7](#_Toc170461631)

[2.2. Lựa chọn phần cứng 8](#_Toc170461632)

[2.3. Nhận diện khuôn mặt dựa trên nền tảng học sâu 10](#_Toc170461633)

[2.3.1. Mô hình MTCNN 10](#_Toc170461634)

[2.3.2. Mô hình rút trích đặc trưng khuôn mặt MobileFacenet 12](#_Toc170461635)

[2.3.3. Công nghệ chống giả mạo khuôn mặt 14](#_Toc170461636)

[2.3.3.1. Giải pháp Silent Face Anti Spoofing của Minivison 14](#_Toc170461637)

[2.4. Công nghệ và môi trường phát triển máy chủ 15](#_Toc170461638)

[2.4.1. Back-end: NodeJS – ExpressJS kết hợp cùng MongoDB 15](#_Toc170461639)

[2.4.2. Sream Video với RTSP 17](#_Toc170461640)

[2.4.3. Giao tiếp thời gian thực với SocketIO 18](#_Toc170461641)

[2.5. Triển khai máy chủ với Docker trên môi trường local và cloud AWS EC2 19](#_Toc170461642)

[Chương 3. PHÂN TÍCH YÊU CẦU VÀ ĐẶC TẢ HỆ THỐNG 22](#_Toc170461643)

[3.1. Phân tích yêu cầu từ thực tiễn 22](#_Toc170461644)

[3.1.1. Tổng quan về kết quả của đề tài 22](#_Toc170461645)

[3.1.2. Những chức năng cần có 22](#_Toc170461646)

[3.1.3. Ràng buộc tổng thể 22](#_Toc170461647)

[3.1.4. Phạm vi kết quả và tính bảo trì về sau 23](#_Toc170461648)

[3.2. Đặc tả các use-case cho hệ thống 23](#_Toc170461649)

[Chương 4. THIẾT KẾ SƠ ĐỒ PHẦN CỨNG 25](#_Toc170461650)

[4.1. Kiến trúc tổng quan hệ thống 25](#_Toc170461651)

[4.2. Biểu đồ thành phần hệ thống 26](#_Toc170461652)

[4.2.1. Biểu đồ thành phần ứng dụng Qt 26](#_Toc170461653)

[4.2.2. Biểu đồ thành phần server 27](#_Toc170461654)

[4.2.3. Thiết kế cơ sở dữ liệu 29](#_Toc170461655)

[Chương 5. TRIỂN KHAI HIỆN THỰC HỆ THỐNG 32](#_Toc170461656)

[5.1. Triển khai hiện thực máy chủ 32](#_Toc170461657)

[5.1.1. Máy chủ API 32](#_Toc170461658)

[5.1.2. Triển khai tính năng giao tiếp thời gian thực với SocketIO 36](#_Toc170461659)

[5.1.2.1. Triển khai phía máy chủ SocketIO 36](#_Toc170461660)

[5.1.2.2. Triển khai Client 37](#_Toc170461661)

[5.2. Triển khai giao diện ứng dụng nhúng ban đầu 38](#_Toc170461662)

[5.3. Triển khai module nhận diện khuôn mặt 41](#_Toc170461663)

[5.3.1. Chuẩn bị bộ dữ liệu 41](#_Toc170461664)

[5.3.2. Huấn luyện mô hình 46](#_Toc170461665)

[5.3.3. Kết quả huấn luyện 48](#_Toc170461666)

[5.3.4. Chuyển đổi mô hình xuống nền tảng nhúng 49](#_Toc170461667)

[5.3.5. Hiện thực trên C++ để tính hợp vào ứng dụng Qt 50](#_Toc170461668)

[5.3.6. Chuyển đổi mô hình Anti-Spoofing từ Python xuống C++ 53](#_Toc170461669)

[5.4. Triển khai stream video với RTSP 54](#_Toc170461670)

[5.5. Triển khai module Logger lưu lại nhật ký ứng dụng 57](#_Toc170461671)

[5.6. Triển khai ứng dụng xuống thiết bị phần cứng 58](#_Toc170461672)

[Chương 6. THỰC HIỆN KIỂM THỬ VÀ LÀM TÀI LIỆU 63](#_Toc170461673)

[6.1. Triển khai kiểm thử API với Postman 63](#_Toc170461674)

[6.2. Triển khai kiểm thử khả năng chịu tải máy chủ SocketIO 63](#_Toc170461675)

[6.3. Triển khai kiểm thử tính năng nhận diện khuôn mặt 67](#_Toc170461676)

[6.4. Thông số hoạt động của ứng dụng 71](#_Toc170461677)

[Chương 7. ĐÁNH GIÁ KẾT QUẢ VÀ KẾT LUẬN 73](#_Toc170461678)

[7.1. Đánh giá kết quả và so sánh với mục tiêu ban đầu 73](#_Toc170461679)

[7.2. Một số điểm hạn chế của đề tài 74](#_Toc170461680)

[7.3. Hướng phát triển tiếp theo 74](#_Toc170461681)

DANH MỤC HÌNH VẼ

[*Hình 1.1. Mô hình lớp học thông minh của đại học Quốc gia Thanh Hoa* 3](#_Toc170461565)

[*Hình 1.2. Lớp học thông minh mà nhóm tác giả ĐH CNTT đề xuất* 5](#_Toc170461566)

[*Hình 2.1. Hình ảnh về kiến trúc Qt* 8](#_Toc170461567)

[*Hình 2.2. Cấu hình của Jetson Nano* 8](#_Toc170461568)

[*Hình 2.3. Hình minh họa kit Jetson nano B01 làm bộ điều khiển trong Robot.* 9](#_Toc170461569)

[*Hình 2.4. Mô tả về hàm mất mát triplet loss* 12](#_Toc170461570)

[*Hình 2.5. Kiến trúc mô hình do Minivision phát triển mã nguồn mở* 14](#_Toc170461571)

[*Hình 2.6. Hình minh họa cách truyền nhận dữ liệu của giao thức RTSP* 18](#_Toc170461572)

[*Hình 3.1. Hình minh họa về thiết kế chức năng của thiết bị bục giảng* 24](#_Toc170461573)

[*Hình 4.1. Mô hình tổng thể của hệ thống* 25](#_Toc170461574)

[*Hình 4.2. Biểu đồ thành phần trong phần ứng dụng Qt* 27](#_Toc170461575)

[*Hình 4.3. Biểu đồ thành phần theo MVC* 28](#_Toc170461576)

[*Hình 4.4. Biểu đồ cơ sở dữ liệu* 31](#_Toc170461577)

[*Hình 5.1. Mô tả về cấu trúc thư mục của API server* 32](#_Toc170461578)

[*Hình 5.2. Hình mô tả luồng xử lý của máy chủ đối với 1 API* 34](#_Toc170461579)

[*Hình 5.3. Biểu đồ tuần tự để xác thực request* 35](#_Toc170461580)

[*Hình 5.4: Cấu hình nginx cân bằng tải* 36](#_Toc170461581)

[*Hình 5.5. Mô hình triển khai của máy chủ SocketIO* 37](#_Toc170461582)

[*Hình 5.6. Hình minh họa về bộ interface cho lớp SocketIO-client* 38](#_Toc170461583)

[*Hình 5.7. Hình minh họa về triển khai lắng nghe một sự kiện qua socket* 38](#_Toc170461584)

[*Hình 5.8. Giao diện trang đăng nhập và trang thông tin cá nhân* 39](#_Toc170461585)

[*Hình 5.9. Giao diện trang lịch học* 39](#_Toc170461586)

[*Hình 5.10. Giao diện trang lớp học và bài giảng* 39](#_Toc170461587)

[*Hình 5.11. Giao diện trang bài tập* 40](#_Toc170461588)

[*Hình 5.12. Giao diện trang danh sách học sinh và điểm danh* 40](#_Toc170461589)

[*Hình 5.13. Giao diện trang thông báo* 40](#_Toc170461590)

[*Hình 5.14. Giao diện trang xem stream video và mở bài giảng* 40](#_Toc170461591)

[*Hình 5.15. Một số mẫu dữ liệu trong tập CASIA-Webface* 44](#_Toc170461592)

[*Hình 5.16. Một số mẫu dữ liệu trong tập LFW(trái) và AgeDB-30 (phải)* 45](#_Toc170461593)

[*Hình 5.17. Một số mẫu về tập VN-Celeb* 45](#_Toc170461594)

[*Hình 5.18. Biểu đồ tuần tự thực hiện huấn luyện mô hình* 47](#_Toc170461595)

[*Hình 5.19. Kết quả kiểm thử độ chính xác sau khi tinh chỉnh trên bộ dữ liệu người VN* 48](#_Toc170461596)

[*Hình 5.20. Kết quả hàm mất mát giảm trong khi tinh chỉnh trên bộ dữ liệu người VN* 49](#_Toc170461597)

[*Hình 5.21. Các bước chuyển đổi định dạng pth qua param và bin* 50](#_Toc170461598)

[*Hình 5.22. Hình mô tả code interface cho class FaceService* 50](#_Toc170461599)

[*Hình 5.23. Biểu đồ tuần tự xử lý nhận diện khuôn mặt* 52](#_Toc170461600)

[*Hình 5.24. Kết quả nhận dạng khuôn mặt là thật* 53](#_Toc170461601)

[*Hình 5.25: Kết quả nhận dạng khuôn mặt là giả với hình ảnh trên điện thoại* 54](#_Toc170461602)

[*Hình 5.26. Hình minh họa về bộ interface cho lớp ScreenStreamer* 56](#_Toc170461603)

[*Hình 5.27. Triển khai module Logger* 58](#_Toc170461604)

[*Hình 5.28. Hình minh họa về nhật ký được lưu lại khi khởi động ứng dụng* 58](#_Toc170461605)

[*Hình 5.29. Script cài đặt NCNN* 59](#_Toc170461606)

[*Hình 5.30. Script cài đặt OpenCV 4.5.3* 60](#_Toc170461607)

[*Hình 5.31. Script cài đặt Qt 5.9.5* 61](#_Toc170461608)

[*Hình 5.32. Script biên dịch chương trình bằng CMake* 61](#_Toc170461609)

[*Hình 5.33. Hình ảnh ứng dụng Qt chạy trên thiết bị* 62](#_Toc170461610)

[*Hình 6.1. Hình mô tả quá trình thực hiện kiểm thử API trên Postman* 63](#_Toc170461611)

[*Hình 6.2: Tổng số yêu cầu trên 1 giây* 64](#_Toc170461612)

[*Hình 6.3: Thời gian phản hồi trung bình cho 1 request* 65](#_Toc170461613)

[*Hình 6.4: Tổng số người dùng truy cập* 65](#_Toc170461614)

[*Hình 6.6: Hiển thị người dùng trong trạng thái đã đăng nhập và vào lớp.* 66](#_Toc170461615)

[*Hình 6.7. Hình thông tin về RAM trước khi chạy ứng dụng* 72](#_Toc170461616)

[*Hình 6.8. Hình ảnh bộ nhớ RAM sau khi được chạy ứng dụng* 72](#_Toc170461617)

DANH MỤC BẢNG

[*Bảng 6‑1. Bảng thông số máy chủ SocketIO* 64](#_Toc170461618)

[*Bảng 6‑2: Bảng đánh giá chung cho việc kiểm tra tải.* 66](#_Toc170461619)

[*Bảng 6‑3. Bảng độ chính xác kiểm thử trên tập dữ liệu của mô hình nhận diện khuôn mặt* 68](#_Toc170461620)

[*Bảng 6‑4. Bảng kết quả thực nghiệm nhận diện khuôn mặt* 70](#_Toc170461621)

[*Bảng 7‑1. Bảng so sánh kết quả thu được và mục tiêu đề cương* 74](#_Toc170461622)

DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| Từ viết tắt | Ý nghĩa |
| DOM | Document Object Model |
| JSX | JavaScript XML |
| XHR Polling | XMLHTTPRequest Polling |
| JSONP Polling | JSON with Padding Polling |
| RTSP | Real-Time Streaming Protocol |
| EC2 | Elastic Compute Cloud |
| CLI | Command Line Interface |
| VPC | Virtual Private Cloud |
| CSDL | Cơ sở dữ liệu |
| API | Application Programming Interface |
| IoT | Internet of Things |
| MVC | Model-View-Controller |
| REST | Representational State Transfer |
| QML | Qt Meta-Object Language |
| NGINX | Engine-X (Web server software) |
| TLS | Transport Layer Security |
| SSL | Secure Sockets Layer |
| OS | Operating System |
| CPU | Central Processing Unit |
| CNN | Convolutional Neural Network |

LỜI CẢM ƠN

Trước tiên, nhóm em xin chân thành gửi lời cảm ơn đến thầy Nguyễn Minh Sơn, vì sự nhiệt tình hướng dẫn, những lời khuyên chân thành và sự hỗ trợ xuyên suốt quá trình thực hiện khóa luận này. Những góp ý và định hướng của thầy đã giúp nhóm tụi em hoàn thành luận văn và đã có được những kết quả đáng kể .

Nhóm em cũng xin bày tỏ sự biết ơn đến tất cả thầy cô giáo ở khoa Kỹ thuật máy tính và trường Đại học Công nghệ thông tin, những người đã truyền đạt kiến thức và kinh nghiệm quý báu trong suốt quá trình tham gia học tập tại trường. Sự hỗ trợ và khích lệ từ các thầy cô là động lực lớn lao giúp nhóm em vượt qua những rào cản trong suốt quá trình học tập và nghiên cứu.

Với hạn chế về thời gian nghiên cứu, cùng với sự hiểu biết và tài nguyên có hạn, khóa luận không thể tránh khỏi những sai sót. Nhóm em rất mong được sự góp ý của thầy, cô, cùng với những người quan tâm để bài khóa luận hoàn thiện hơn.

***Chúng em xin chân thành cảm ơn!***

Sinh viên   
  
 Lê Thị Thu Huyền

Phan Ái Linh

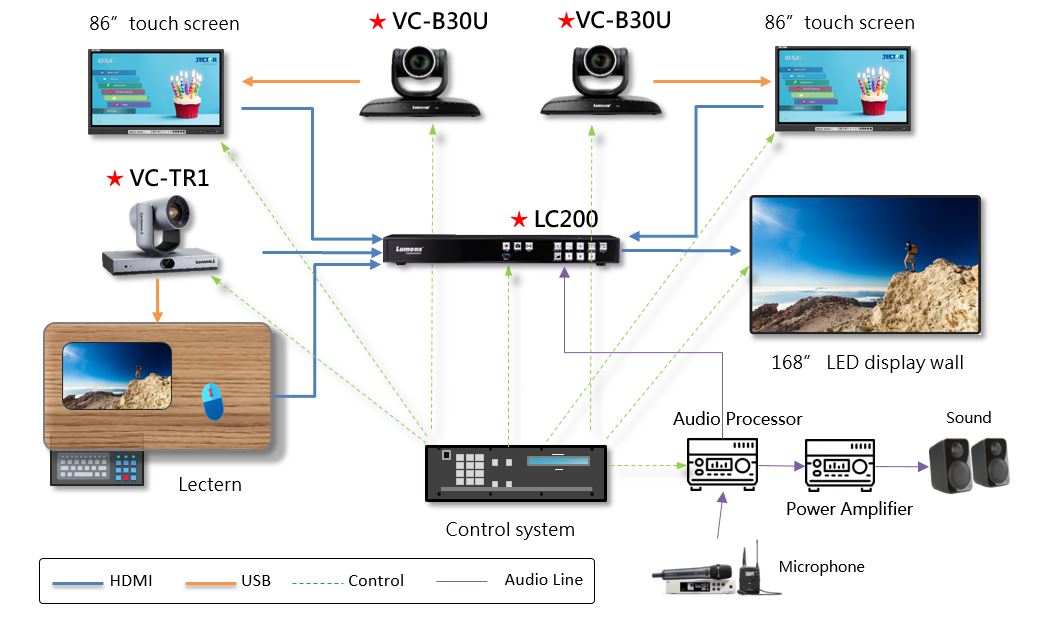
# GIỚI THIỆU ĐỀ TÀI

## Tổng quan về đề tài

Trước tình hình luôn thay đổi và nhu cầu thích ứng với hoàn cảnh thách thức, việc triển khai mô hình “lớp học Hybrid” là một giải pháp không còn xa lạ trong nền giáo dục. Mô hình này là sự kết hợp giữa cả cách học tập truyền thống và cả hiện đại với sự hỗ trợ từ các thiết bị thông minh, công nghệ hiện đại cùng với phương pháp học mới giúp việc học tập và dạy học trở nên linh hoạt, dễ dàng tiếp cận và đáp ứng đúng hoàn cảnh hơn như đợt dịch Covid vừa qua.

### Các nghiên cứu quốc tế

Mô hình lớp học thông minh kết hợp không phải mới ở các nước phát triển nhưng kể từ thời điểm dịch bệnh Covid-19 bùng phát thì thực sự bắt đầu được tập trung và đẩy mạnh nhiều hơn. Kể từ thời điểm này thì cũng có nhiều mô hình với các đặc điểm và chức năng khác nhau được giới thiệu như nghiên cứu lớp học kết học dành cho đại học của Đại học Quốc gia Thanh Hoa, Đài Loan.



*Hình 1.1. Mô hình lớp học thông minh của đại học Quốc gia Thanh Hoa*

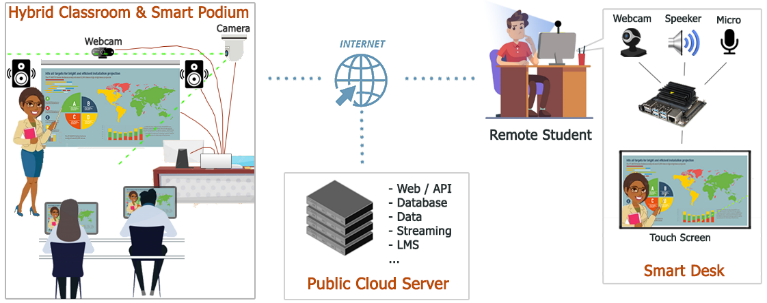
Như hình minh họa 1.1, Hệ thống Lớp học thông minh AUO bao gồm màn hình LED lớn 135 inch với màn hình cảm ứng 86 inch mỗi bên. Giáo viên có thể viết trực tiếp lên màn hình và kết nối với màn hình lớn. Ngoài ra, trong lớp học còn có 3 camera theo dõi tự động. Học sinh ở xa có thể nhìn và tương tác cùng với các bạn trong lớp thông qua hai camera còn lại và tương tác trong thời gian thực.

Tuy mô hình này đã đáp ứng được những nhu cầu trong giáo dục và học tập, nhưng chưa khai thác được tiềm năng của trí tuệ nhân tạo (AI) và vạn vật internet (IoT). Với sự phát triển nhanh chóng của công nghệ, việc áp dụng AI và IoT trong xây dựng một giải pháp toàn diện cho lớp học thông minh được coi là cần thiết để đạt được sự đột phá và mang lại giá trị cao nhằm phục vụ cho học sinh, giáo viên, phụ huynh và nhà trường.

### Các nghiên cứu trong nước

Ở Việt Nam, đã có những nghiên cứu liên quan mô hình lớp học kết hợp như trong đề tài *[1] "Smart Desk in Hybrid Classroom: Detecting student's lack of concentration when studying"*, do các tác giả Manh Hung LE, Thien Minh DOAN, Duy Dieu NGUYEN và Minh Son NGUYEN thực hiện và được công bố tại hội nghị ACOMPA 2022.

Trong nghiên cứu này, các tác giả đưa ra đề xuất một mô hình lớp học kết hợp, gồm bục giảng thông minh, bàn học thông minh cùng với các ứng dụng phần mềm chạy cùng với thiết bị. Những phần mềm này giúp học sinh có thể tham gia lớp học trực tuyến hoặc học trực tiếp, đồng thời cũng giúp cho giáo viên quản lý lớp học.



*Hình 1.2. Lớp học thông minh mà nhóm tác giả ĐH CNTT đề xuất*

Đặc biệt là trong nghiên cứ này thì nhóm tác giả còn đề cập đến việc tích hợp thêm trí tuệ nhân tạo (AI) nhằm giám sát sự tập trung học của học sinh và ứng dụng Internet of Things (IoT) trong quản lý lớp học.

Mô hình lớp học được đề cập ở nghiên cứu này đã đánh dấu sự tiến bộ trong việc ứng dụng công nghệ vào giáo dục, tạo ra một môi trường học tập tiên tiến và tăng tương tác. Việc ứng dụng thêm trí tuệ nhân tạo và Internet of Things cho lớp học không chỉ giúp cải thiện quá trình học tập, mà còn mở ra những tiềm năng phát triển hơn cho nền giáo dục.

## Mục tiêu của đề tài

Thiết kế và triển khai một Miniserver trong phạm vi trường học để thiết bị bục giảng và các thiết bị IoT khác như bàn học thông minh trong mô hình lớp học Hybrid có thể giao tiếp theo thời gian thực. Thêm vào đó, để đáp ứng chức năng giảng dạy online của lớp học Hybrid, server này cũng được sử dụng cho việc stream video và đồng bộ dữ liệu với Cloud server. Ngoài ra, cũng xây dựng một ứng dụng Qt với các chức năng như tự động nhận diện khuôn mặt giáo viên khi đăng nhập, điểm danh tự động khi học sinh vào lớp,… dùng để kiểm tra các chức năng đã trình bày của Miniserver.

## Giới hạn nghiên cứu của đề tài

* Ứng dụng Qt C++ chạy được trên thiết bị nhúng cụ thể là Jetson Nano
* Hiện thực được Miniserver đáp ứng nhu cầu giao tiếp thời gian thực giữa thiết bị bục giảng và thiết bị bàn học với khả năng chịu tải tối đa 1000 người dùng truy cập
* Giải thuật nhận diện khuôn mặt có độ chính xác với tập dữ liệu kiểm tra đạt trên 90% và tốc độ xử lý ở trên thiết bị nhúng nhỏ hơn 150ms trên 1 khung hình
* Tích hợp được chức năng stream video trên chính Miniserver đó phục vụ cho việc mở lớp học online trên ứng dụng Qt.

## Kết quả mong đợi của đề tài

* Ứng dụng Qt có các chức năng đã đề cập ở trên có thể chạy ổn định không bị đứt quãng khi đang sử dụng trên thiết bị nhúng Jetson Nano.
* Giải thuật nhận diện khuôn mặt có độ chính xác với tập dữ liệu kiểm tra đạt trên 95% và tốc độ xử lý trên thiết bị nhúng nhỏ hơn 100ms trên 1 khung hình
* Tốc độ giao tiếp thời gian thực giữa thiết bị bục giảng và bàn học trong lớp học có độ trễ nhỏ hơn 2 giây và có thể chịu tải tối đa 1000 người dùng truy cập.

# TÌM HIỂU LÝ THUYẾT VÀ LỰA CHỌN CÔNG NGHỆ

## Công nghệ lập trình phần mềm nhúng

Qt là Framework hỗ trợ máy tính phát triển trên các ứng dụng đa nền tảng. Nó được phát triển bởi công ty The Qt Company và là công ty con thuộc sở hữu của Digia Plc. Dưới đây là một số điểm nổi trội:

* Ngôn ngữ lập trình: Qt nó không là ngôn ngữ lập trình riêng. Đó là một framework và dùng ngôn ngữ C++. Có bộ tiền xử lý MOC (Meta-Object Compiler) được sử dụng để mở rộng ngôn ngữ với các tính năng như signals và slots.
* Nền tảng hỗ trợ: Linux, OS X, Windows, Andorid, VxWorks, QNX, Ios, BlackBerry và những nền tảng khác.
* Xây dựng hệ thống: mặc dù bất kỳ hệ thống nào cũng có thể được sử dụng với Qt, nhưng Qt có QMake của riêng nó. Nó là giao diện đa nền tảng người dùng dành cho các hệ thống xây dựng dựa trên một số nền tảng như GNU Make, Visual Studio và Xcode. Thêm vào đó, CMake cũng là một giải pháp thay thế phổ biến để xây dựng dự án Qt.
* Các thành phần và thư viện: Các thành phần chính gồm Qt Core (quản lý tài nguyên và quản lý sự kiện), Qt GUI (đồ họa và đa phương tiện), Qt Widgets (giao diện người dùng truyền thống), Qt QML (giao diện người dùng động và linh hoạt) và Qt Network (mạng).
* Công cụ phát triển tích hợp: Qt đi kèm với Môi trường phát triển tích hợp (IDE) của riêng nó, được đặt tên là Qt Creator. Nó chạy trên Linux, OS X và Windows. Cung cấp tính năng hoàn thiện mã thông minh, tích hợp hệ thống trợ giúp, trình gỡ lỗi và trình biên dịch cũng như tất cả các hệ thống kiểm soát phiên bản chính.
* Quốc tế hóa và bản địa hóa: Qt có tính năng hỗ trợ hầu hết các ngôn ngữ và hệ thống chữ viết ngày nay.
* Hai thành phần của Qt là GUI Design và C++. Qt UI Design có 2 lựa chọn là Qt Quick và Qt Widgets. Tuy nhiên, việc sử dụng Qt Quick mang lại vẻ đẹp và sự hiện đại hơn cho các thiết bị hay hệ điều hành.

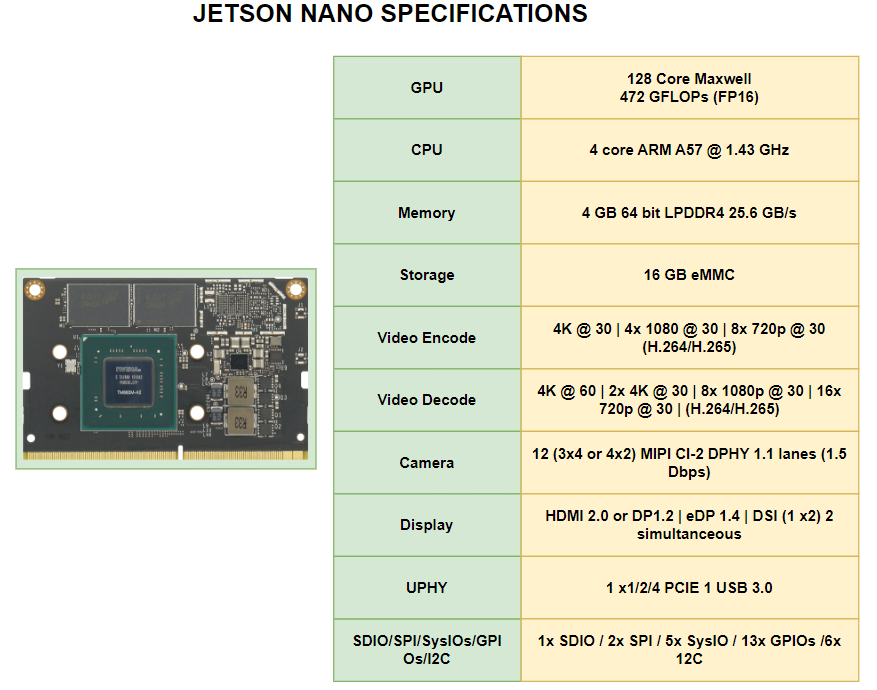


*Hình 2.1. Hình ảnh về kiến trúc Qt*

QML (Meta-Object Language) là ngôn ngữ cho phép xây dựng các thành phần giao diện tĩnh dựa trên JavaScript. Nó hỗ trợ xử lý các sự kiện, rang buộc dữ liệu và tạo hiệu ứng đồ họa. Bên cạnh đó, nó cũng cung cấp khả năng kết hợp với mã C++ thông qua các liên kết C++/QML để cho phép bạn mở rộng khả năng ứng dụng.

## Lựa chọn phần cứng

NVIDIA Jetson Nano Developer Kit B01 là một máy tính AI cỡ nhỏ được tích hợp có kích thước của một bảng mạch - một bộ phát triển nhúng từ NVIDIA. Dưới đây là một số thông số kỹ thuật quan trọng:



*Hình 2.2. Cấu hình của Jetson Nano*

* Kết nối: Jetson Nano B01 có các cổng kết nối sau:
  + Một cổng HDMI cho đầu ra video.
  + Một cổng Gigabit Ethernet được kết nối mạng có dây.
  + Bốn cổng USB 3.0 dsnh cho các thiết bị ngoại vi.
  + Một khe cắm thẻ nhớ microSD để lưu trữ dữ liệu và hệ điều hành.
  + Cổng cấp nguồn 5V/4A để cung cấp điện cho Jetson Nano.
* GPIO: Jetson Nano B01 có bộ chân GPIO 40 chân, cho phép kết nối và tương tác với các linh kiện và thiết bị ngoại vi khác nhau.

Jetson Nano Developer Kit B01 là một công cụ để phát triển một cách mạnh mẽ. Nó cho phép kết nối với màn hình, bàn phím, chuột để sử dụng, Nó được ứng dụng làm bộ điều khiển chính cho Robot, làm các dự án lập trình nhúng và đặc biệt là các dự án nghiên cứu về lĩnh vực AI, ...



*Hình 2.3. Hình minh họa kit Jetson nano B01 làm bộ điều khiển trong Robot.*

## Nhận diện khuôn mặt dựa trên nền tảng học sâu

### Mô hình MTCNN

MTCNN (Multi-task Cascaded Convolutional Networks) là một trong những công nghệ đã cách mạng hóa lĩnh vực phát hiện và nhận dạng khuôn mặt. Được phát triển vào năm 2016, thuật toán MTCNN sử dụng một chuỗi mạng lưới thần kinh xếp tầng để phát hiện, căn chỉnh và trích xuất các đặc điểm khuôn mặt từ hình ảnh kỹ thuật số với độ chính xác và tốc độ cao. Kiến trúc của MTCNN bao gồm ba cấp độ: P-Net, R-Net và O-Net. Chi tiết như sau:

* P-Net (Proposal Network):
  + P-Net chịu trách nhiệm quét ảnh đầu vào và tạo ra các đề xuất khung chứa khuôn mặt (bounding box proposals) cùng với điểm số tin cậy cho mỗi khung.
  + P-Net là một mạng CNN nhỏ gọn với các lớp tích chập và lớp pooling. Nó quét ảnh đầu vào ở nhiều tỷ lệ để phát hiện khuôn mặt ở các kích thước khác nhau.
  + P-Net cho đầu ra với khung chứa khuôn mặt, điểm số tin cậy cũng như các điểm mốc sơ bộ.
* R-Net (Refine Network):
  + R-Net nhận các đề xuất khung từ P-Net và lọc chúng, loại bỏ các khung chứa khuôn mặt giả và tinh chỉnh các khung chứa khuôn mặt còn lại để chính xác hơn.
  + R-Net là một mạng CNN phức tạp hơn P-Net, với các lớp tích chập và fully connected layer.
  + R-Net cho đầu ra gồm khung chứa khuôn mặt, điểm số tin cậy và các điểm mốc khuôn mặt sau khi được tin chỉnh.
* O-Net (Output Network):
  + O-Net là bước cuối cùng trong MTCNN và có nhiệm vụ chính xác hóa các bounding box và xác định danh tính của khuôn mặt.
  + O-Net là mạng lớn nhất trong ba mạng, với nhiều lớp tích chập và fully connected layer hơn.
  + O-Net cho đầu ra sẽ là khung chứa khuôn mặt, điểm số tin cậy và điểm mốc khuôn mặt chính xác.

MTCNN sử dụng một loại hàm mất mát là Triplet Loss. Nó là một phương pháp huấn luyện mạng neural network sử dụng trong MTCNN để học các đặc trưng khuôn mặt. Nó giúp tạo ra một không gian biểu diễn (embedding space) sao cho các đặc trưng của cùng một người có khoảng cách gần nhau hơn so với các người khác.

Triplet Loss được tính toán bằng cách chọn ba mẫu ảnh từ đợt này: một mốc đại diện cho một ảnh khuôn mặt (anchor), một ảnh khuôn mặt cùng người với mốc (positive) và một ảnh khuôn mặt không cùng người với mốc (negative). Mục tiêu là đảm bảo rằng khoảng cách giữa mốc và ảnh khuôn mặt cùng người nhỏ hơn khoảng cách giữa mốc và ảnh khuôn mặt không cùng người với mốc một lượng nhất định.

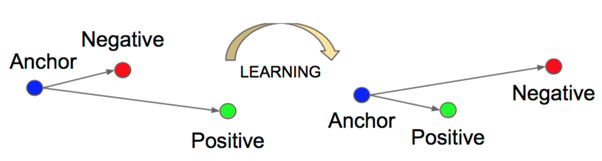
Công thức tính Triplet Loss như sau:

Trong đó:

* f(A), f(P) và f(N) là các biểu diễn (embeddings) của mốc, positive và negative trong không gian đặc trưng.
* Margin là một giá trị biên dương để đảm bảo rằng các điểm của lớp khác khau phải cách xa nhau ít nhất.

Quy trình hóc với Triplet Loss như sau:

* Lựa chọn các bộ ba (triplets): Việc chọn các bộ ba mẫu A, P, N rất quan trọng. Các bộ ba này cần phải được chọn sao cho hữu ích cho quá trình học. Thường thì sẽ có các chiến lược chọn bộ ba như:
* Hard Negative Mining: Chọn các bộ ba mà khoảng cách giữa Anchor và Negative nhỏ hơn hoặc bằng khoảng cách giữa Anchor và Positive.
* Semi-Hard Negative Mining: Chọn các bộ ba mà khoảng cách giữa Anchor và Negative lớn hơn khoảng cách giữa Anchor và Positive nhưng nhỏ hơn khoảng cách giữa Anchor và Positive cộng với margin
* Tính toán khoảng cách: Thường sử dụng khoảng cách Euclidean hoặc cosine để đo khoảng cách giữa các mẫu trong không gian nhúng.
* Cập nhật trọng số: Sử dụng hàm mất mát Triplet Loss để cập nhật trọng số của mạng nơ-ron thông qua quá trình lan truyền ngược (backpropagation).Minh họa cho lý thuyết trên được trình bày trực quan hơn ở hình mô tả 2.10 bên dưới.



*Hình 2.4. Mô tả về hàm mất mát triplet loss*

Để đảm bảo tiến độ cũng như chất lượng kết quả đã được nghiên cứu từ các nhóm trước đó, nhóm quyết định tái sử dụng bộ tham số MTCNN đã được huấn luyện bởi Google.

### Mô hình rút trích đặc trưng khuôn mặt MobileFacenet

Trong bài báo *[21] “MobileFaceNets: Efficient CNNs for Accurate RealTime Face Verification on Mobile Devices”* nhóm tác giả đến từ Đại học Giao Thông Bắc Kinh đã trình bày về mô hình nhận diện khuôn mặt thời gian thực. Nó được tối ưu hóa cho việc triển khai trên thiết bị có tài nguyên hạn chế bằng cách sử dụng một số kỹ thuật quan trọng. Một số điểm nhấn trong mô hình đó như sau:

* *Depthwise Separable Convolution*: Là một mạng nơ-ron tích chập (CNN) nhằm giảm số lượng các tham số và giảm chi phí tính toán trong suốt quá trình huấn luyện. Depthwise Separable Convolution bao gồm hai bước chính là Depthwise Convolution và Pointwise Convolution. Depthwise Convolution thực hiện convolution cho mỗi kênh đầu vào riêng biệt. Áp dụng một convolution 1x1 lên các feature map thu được từ Depthwise Convolution.
* *Batch Normalization*: Thực hiện chuẩn hóa (normalize) đầu vào của mỗi lớp bằng cách điều chỉnh và biến đổi lại thành các giá trị có giá trị kỳ vọng (mean) gần bằng 0 và phương sai (variance) gần bằng 1. Điều này giúp đảm bảo tính ổn định và hiệu quả trong quá trình huấn luyện. Giảm thiểu hiện tượng covariate shift.
* *Residual Learning*: Hay còn gọi là residual networks (ResNets), là một mô hình kiến trúc neural network đặc biệt đã đạt được thành công đáng kể trong việc huấn luyện các mạng sâu sử dụng deep learning. Ý tưởng chính của Residual Learning xuất phát từ việc giảm vấn đề của việc huấn luyện mạng sâu (vanishing gradient problem) bằng cách sử dụng các "skip connections" hoặc "shortcut connections".
* *Global Average Pooling*: Là một kỹ thuật quan trọng trong các mạng CNN, thường được sử dụng để chuyển đổi các feature map cuối cùng thành các vector đặc trưng có kích thước cố định, từ đó cải thiện hiệu suất và tính tổng quát hóa của mô hình.
* *Linear Bottleneck*: Là một khái niệm thiết kế trong các mạng neural network nhằm tối ưu hóa hiệu suất và hiệu quả tính toán, đặc biệt là trong các mô hình như EfficientNet. Bằng cách sử dụng các convolution 1x1 và linear activation, nó giúp giảm chi phí tính toán và số lượng tham số, đồng thời duy trì độ chính xác của mô hình.

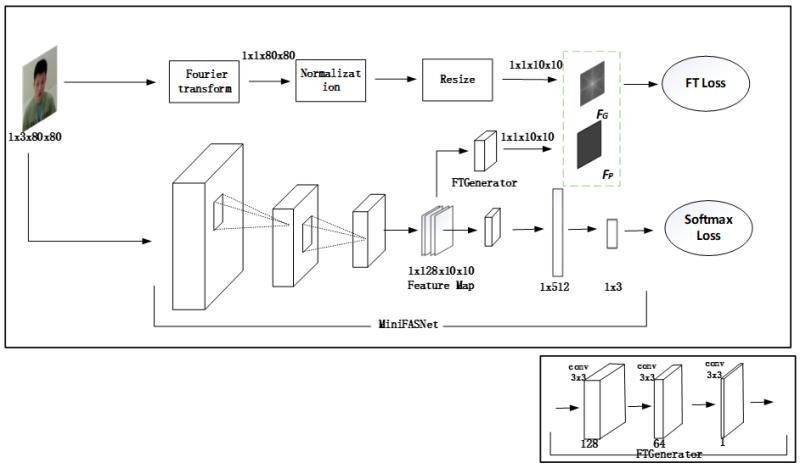
Kết quả đạt được của bài báo trên là:

* Thiết kế một lớp CNN cực kỳ hiệu quả cho xác minh khuôn mặt trên nền tảng các thiết bị di động và nhúng.
* Sự vượt trội của global depthwise convolution layer so với các phương pháp truyền thống như global average pooling.
* Cung cấp các kết quả thực nghiệm chi tiết và so sánh với các mô hình hiện đại khác, chứng minh MobileFaceNets là một giải pháp hiệu quả cho các ứng dụng di động.

### Công nghệ chống giả mạo khuôn mặt

#### Giải pháp Silent Face Anti Spoofing của Minivison

[4] Năm 2020, Minivision technology một công ty tư nhân về giải pháp công nghệ của Trung Quốc có trụ sở tại Nam Kinh đã phát triển mã nguồn mở cho mô hình chống giả mạo khuôn mặt nhanh, nhỏ nhẹ và có độ chính xác cao. Mô hình có kiến trúc như hình bên dưới:

**

*Hình 2.5. Kiến trúc mô hình do Minivision phát triển mã nguồn mở*

MiniFASNetV1 được thiết kế với kích thước tinh gọn và độ chính xác cao, giúp nó phù hợp cho việc triển khai trên các thiết bị mà có sự hạn chế về tài nguyên như điện thoại di động hay hệ thống nhúng.

* Cấu trúc MiniFASNetV1:
  + Convolutional Layers: MiniFASNetV1 sử dụng các lớp convolution để rút trích đặc trưng từ khuôn mặt của ảnh đầu vào.
  + Depthwise Separable Convolution: Đây là một loại convolution được sử dụng để giảm thiểu lượng tham số cần thiết trong mô hình, đồng thời tăng tốc độ tính toán và giảm độ phức tạp. MiniFASNetV1 thường áp dụng Depthwise Separable Convolution để đảm bảo mô hình nhỏ gọn và phù hợp với các điều kiện có hạn về tài nguyên tính toán như trong các thiết bị di động.
  + Batch Normalization: Đây là kỹ thuật được sử dụng để cân bằng lại đầu ra của các lớp trước khi đưa vào hàm kích hoạt (activation function), giúp cải thiện tốc độ hội tụ và ổn định mô hình.
  + Activation Functions: MiniFASNetV1 thường sử dụng các hàm kích hoạt như ReLU (Rectified Linear Unit) nhằm giúp học được các đặc trưng phi tuyến tính cho mô hình.
  + Pooling Layers: Để giảm kích thước của tensor và giảm số lượng tham số, MiniFASNetV1 sử dụng các lớp pooling như MaxPooling.
  + Fully Connected Layers: Sau khi rút trích đặc trưng từ các lớp convolution, thông thường mô hình sẽ kết hợp các lớp fully connected để phân loại đối tượng (ví dụ như xác định khuôn mặt thật hay giả).

Với kết quả đạt được từ nghiên cứu đó nhóm đã quyết định sử dụng lại kết quả nghiên cứu từ mã nguồn mở đã có sẵn để tiếp tục triển khai.

## Công nghệ và môi trường phát triển máy chủ

### Back-end: NodeJS – ExpressJS kết hợp cùng MongoDB

Node.js là một nền tảng phát triển dựa trên nền tảng JavaScript và sử dụng runtime V8 của Google Chrome. Các điểm nổi bật của Node.js bao gồm:

* Non-blocking và asynchronous: Node.js sử dụng các mô hình xử lý không đồng bộ (non-blocking) giúp đáp ứng các ứng dụng có nhiều thao tác I/O như ứng dụng web.
* JavaScript: Sử dụng JavaScript cho cả frontend và backend giúp cho việc phát triển và duy trì dễ dàng hơn, đồng thời giảm thiểu ngưỡng học của nhà phát triển.
* Mô-đun và cộng đồng mở rộng: Node.js có một hệ sinh thái mô-đun phong phú trên npm (Node Package Manager), cung cấp nhiều module hữu ích để phát triển nhanh chóng và mở rộng.

Express.js là một framework web cho Node.js, cung cấp một cách tiếp cận tối giản hóa để xây dựng các ứng dụng web. Các đặc điểm quan trọng của Express.js bao gồm:

* Lightweight và unopinionated: Express.js là một framework nhẹ, linh hoạt và không ép buộc một cách tiếp cận cụ thể nào. Điều này cho phép nhà phát triển có thể tự do chọn các công nghệ và cấu trúc để phù hợp với dự án của họ.
* Middleware: Express.js sử dụng middleware để xử lý yêu cầu HTTP và hỗ trợ việc quản lý session, route, error handling, và nhiều nội dung khác một cách hiệu quả.

MongoDB là một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL), sử dụng JSON-syle documents để lưu trữ dữ liệu. Những điểm mạnh của MongoDB bao gồm:

* Schema-free: MongoDB không yêu cầu schema cứng nhắc, cho phép dữ liệu linh hoạt và thay đổi theo thời gian.
* Tích hợp tốt với JavaScript: Với MongoDB, nhà phát triển có thể lưu trữ và truy xuất dữ liệu một cách tự nhiên bằng JavaScript.
* Horizontal scalability: MongoDB hỗ trợ mở rộng theo chiều ngang (horizontal scaling), giúp cho việc mở rộng và quản lý dữ liệu trở nên dễ dàng và hiệu quả hơn.

Kết hợp Node.js, Express.js và MongoDB mang lại nhiều lợi ích trong việc phát triển ứng dụng:

* Tích hợp dễ dàng: Node.js và Express.js cùng với MongoDB hoạt động tốt với nhau, cung cấp một cấu trúc phát triển linh hoạt.
* Phát triển nhanh chóng: Nhờ vào JavaScript trên cả frontend và backend, việc phát triển và thử nghiệm ứng dụng được đơn giản hóa.
* Hiệu suất cao: Sử dụng Node.js với Express.js giúp cho việc xử lý yêu cầu và phản hồi nhanh chóng, kết hợp với MongoDB giúp tối ưu hóa lưu trữ và truy xuất dữ liệu.
* Mở rộng dễ dàng: Cả Node.js và MongoDB đều hỗ trợ mở rộng theo chiều ngang, cho phép ứng dụng mở rộng khi cần thiết.

### Sream Video với RTSP

RTSP, viết tắt của Real Time Streaming Protocol, nó là một giao thức mạng phục vụ điều khiển phát trực tuyến âm thanh hoặc video. Nó cung cấp một tập các phương pháp điều khiển cho các ứng dụng phát trực tuyến đa phương tiện, cho phép người dùng tạm dừng, tua trở lại, v.v. các dữ liệu đa phương tiện được phát trực tuyến. Nằm ở lớp ứng dụng của mô hình OSI. Nó được sử dụng để thiết lập và kiểm soát các phiên đa phương tiện giữa các người dùng cuối. RTSP giúp làm việc với nhiều giao thức truyền thông dữ liệu như RTP (Real-time Transport Protocol) và RDP (Reliable Data Protocol).

A picture containing text, screenshot, diagram, number

Description automatically generated

*Hình 2.6. Hình minh họa cách truyền nhận dữ liệu của giao thức RTSP*

RTSP chủ yếu được sử dụng trong các ứng dụng yêu cầu phát trực tuyến trực tiếp, ví dụ: trong các hệ thống video theo dõi, hệ thống giảng dạy trực tuyến và nhiều hơn nữa.

### Giao tiếp thời gian thực với SocketIO

Socket.IO là một thư viện quan trọng và phổ biến trong các ứng dụng web đòi hỏi thời gian thực. Nó hỗ trợ việc truyền tải dữ liệu một cách nhanh chóng và liên tục giữa server và client, Socket.IO là lựa chọn hàng đầu cho các ứng dụng yêu cầu cập nhật liên tục như chat, thông báo thời gian thực, và trò chơi trực tuyến.

Socket.IO hoạt động dựa trên giao thức WebSocket, cho phép truyền tải dữ liệu hai chiều giữa client và server qua một kết nối duy nhất. Nếu môi trường hoặc trình duyệt không hỗ trợ WebSocket, Socket.IO cũng tự động chuyển sang các phương thức thay thế như AJAX long-polling, giúp đảm bảo sự ổn định và độ tương thích cao.

Thư viện cung cấp các API đơn giản và dễ dàng sử dụng cho cả server và client. Trên phía server, Socket.IO thường được tích hợp với Node.js để quản lý kết nối và giao tiếp với client. Trên phía client, thư viện này hỗ trợ những ứng dụng để kết nối và giao tiếp với server một cách hiệu quả.

Trường hợp sử dụng thư viện SocketIO-CPP-Client cho ứng dụng Qt là một quyết định thông minh. Thư viện này mang lại nền tảng ổn định và linh hoạt để kết nối và giao tiếp với máy chủ Socket.IO bằng ngôn ngữ C++, giúp tiết kiệm thời gian và sức lực trong triển khai các tính năng yêu cầu giao tiếp thời gian thực.

Việc sử dụng lại thư viện mã nguồn mở như SocketIO-CPP-Client giúp giảm bớt lượng công việc và tăng tính khả dụng và sự tin cậy của ứng dụng.

## Triển khai máy chủ với Docker trên môi trường local và cloud AWS EC2

Docker là một nền tảng cho phép chúng ta đóng gói và hiện thực ứng dụng cùng với các phụ thuộc của nó vào các container độc lập nhau. Các container này được đóng gói một cách chặt chẽ và có thể di chuyển giữa các môi trường máy chủ một cách dễ dàng mà không cần thay đổi code hay cài đặt lại phần mềm. Docker có một số lợi ích sau:

* Đóng gói độc lập và di động: Docker cho phép đóng gói các ứng dụng cùng với các phụ thuộc của nó vào các container độc lập. Điều này giúp bảo đảm rằng ứng dụng sẽ hoạt động giống nhau trên mọi môi trường mà không phụ thuộc vào cài đặt môi trường máy chủ. Và ta có thể khởi chạy container trên mỗi hệ thống mà ta muốn.
* Tính nhẹ và hiệu suất: Docker sử dụng các container để chạy ứng dụng, giúp tiết kiệm tài nguyên và hiệu suất.
* Tính linh hoạt và dễ dàng mở rộng: Docker cho phép dễ dàng mở rộng ứng dụng bằng cách triển khai nhiều container trên nhiều máy chủ. Việc điều chỉnh số lượng container có thể tự động hoặc thủ công tùy vào nhu cầu.
* Quản lý phụ thuộc và phiên bản: Dockerfile cho phép xác định rõ ràng các phụ thuộc của ứng dụng và cách cài đặt chúng, giúp dễ dàng quản lý các phiên bản và cập nhật của ứng dụng.

Với việc triển khai mô hình tính toán biên (Egde Computing) là mô hình tính toán phân tán bằng cách triển khai gần với người dùng cuối that thì triển khai trên cloud thì nhóm đã lựa chọn sử dụng một máy tính mà xem nó là một MiniServer. Việc áp dụng triển khai mô hình này mang lại một số lợi ích sau:

* Triển khai gần nguồn dữ liệu: Điện toán biên cho phép dữ liệu được xử lý và tính toán được thực hiện ngay tại chỗ, gần với nơi thu thập dữ liệu. Điều này giúp giảm độ trễ và tăng tính phản hồi của các ứng dụng, đặc biệt là trong các ngữ cảnh yêu cầu xử lý nhanh chóng và theo thời gian thực.
* Giảm tải cho mạng và trung tâm dữ liệu: Việc xử lý dữ liệu trực tiếp tại các điểm biên giảm tải cho mạng truyền thông và cả trung tâm dữ liệu, do không cần truyền dữ liệu lớn qua lại giữa các thiết bị và đám mây.

Nhưng với mục tiêu mở rộng hệ thống, nhóm đã triển khai thêm trên môi trường cloud sử dụng Amazon EC2.

Amazon Elastic Compute Cloud (Amazon EC2) là một dịch vụ cung cấp khả năng tính toán đám mây mạnh mẽ của Amazon Web Services (AWS). Dịch vụ này cho phép chúng ta thuê máy chủ ảo và quản lý dịch vụ máy chủ đám mây theo nhu cầu mà không cần phải mua và duy trì phần cứng máy chủ riêng biệt. Một đặc điểm chính của Amazon EC2 trở nên phổ biến: Khả năng mở rộng linh hoạt, môi trường an toàn để phát triển các ứng dụng, phổ biến hầu hết các vùng lãnh thổ.

Vì còn nhiều hạn chế về mặt tài chính nên ở thời điểm hiện tại nhóm quyết định thử nghiệm trên cấu hình cloud với máy ảo EC2 của Amazon và local như sau:

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Cấu hình | |
| AWS EC2 | Local |
| Hệ điều hành | Ubuntu, 20.04 LTS focal image  build on 2023-05-17 | Windows 10 Pro 64-bit (10.0, Build 19045) |
| Kiến trúc (Loại) | AMD-64 (t2.micro) | 11th Gen Intel® Core™ i7-1165G7 (8 CPUs), ~1.7GHz |
| RAM | 1Gb | 16381MB |
| SSD | 32Gb | 512GB |
| Nơi đặt máy chủ | Singapore |  |

# PHÂN TÍCH YÊU CẦU VÀ ĐẶC TẢ HỆ THỐNG

## Phân tích yêu cầu từ thực tiễn

### Tổng quan về kết quả của đề tài

Kết quả của đề tài là đã hiện thực được một thiết bị bục giảng có các chức năng công nghệ như có khả năng tự động đăng nhập bằng nhận dạng khuôn mặt, nhận thông tin điểm danh từ thiết bị bàn học thông minh theo thời gian thực. Giáo viên cũng có thể stream video màn hình ứng dụng cùng lúc với video từ camera được cài đặt chung với thiết bị, cũng như giao bài tập, các bài kiểm tra cho học sinh. Ngoài thiết bị bục giảng với các chức năng trên thì đề tài cũng nghiên cứu thiết bị một Miniserver dùng cho mô hình giao tiếp thời gian thực giữa 2 thiết bị bục giảng và bàn học thông minh trong lớp học Hybrid.

### Những chức năng cần có

* Ứng dụng Qt cho thiết bị bục giảng:
  + Kết nối theo thời gian thực cho các thiết bị bàn học thông minh.
  + Mở được các phòng học trực tuyến để các thiết bị của học sinh có thể tham gia vào lớp học.
  + Giao bài tập tới các thiết bị bàn học thông minh theo thời gian thực.
  + Chia sẻ màn hình bài giảng tới các thiết bị bàn học thông minh
* Miniserver:
  + Cung cấp các API để các thiết bị để chúng có thể trao đổi dữ liệu.
  + Cân bằng tải, thực hiện các tác vụ xử lý một cách đồng thời, tăng tính phản hồi cho các yêu cầu từ client.
  + Xử lý các sự kiện theo thời gian thực.

### Ràng buộc tổng thể

Với tính chất của đề tài thì cần có rằng buộc về tổng thể một số tiêu chí như sau:

* Cần đáp ứng được yêu cầu về đăng nhập tự động bằng nhận dạng khuôn mặt.
* Cần đảm bảo kết nối thời gian thực với các thiết bị bàn học.
* Cần cung cấp giao diện hiển thị người dùng dễ sử dụng
* Cần phải tuân thủ những quy tắc về bảo mật, không làm mất mát dữ liệu ra bên ngoài.

### Phạm vi kết quả và tính bảo trì về sau

Phạm vi kết quả của nghiên cứu là xây dựng một thiết bị bục giảng có các chức năng đã được đề cập và một miniserver cung cấp các dịch vụ cho việc giao tiếp, giao đổi dữ liệu giữa các thiết bị trong mô hình lớp học Hybrid. Tính bảo trì về sau yêu cầu đảm bảo rằng thiết bị hoạt động ổn định và có thể cập nhật các công nghệ mới để có thể giải quyết hoặc cải thiện những vấn đề phát sinh.

## Đặc tả các use-case cho hệ thống

Nhằm đáp ứng những nhu cầu cụ thể từ giáo viên và từ học sinh trong quá trình tương tác của hai thiết bị bục giảng thông minh và bàn học thông minh, trong đề tài nghiên cứu này nhóm đã quyết đinh phát triển tập trung các chức năng sau trước tiên, bao gồm:

**Thiết bị bục giảng:**

* *Đăng nhập*: Giáo viên có thể đăng nhập bằng khuôn mặt hoặc tùy chọn đăng nhập bằng tên đăng nhập và mật khẩu.
* *Quản lý lớp học trực tuyến*: Giáo viên có thể mở lớp học trực tuyến và học sinh có thể truy cập vào lớp học trên đa nền tảng một cách dễ dàng để tương tác và học tập. Quá trình này được đảm bảo tính riêng tư và bảo mật cho lớp học.
* *Thông tin về trạng thái học sinh*: Giáo viên có thể kiểm tra liệu học sinh đã điểm danh hay chưa hoặc cũng có thể sử dụng tính năng mở camera để xem hình ảnh của học sinh trong lớp trong lúc làm bài kiểm tra
* *Giao bài tập*: Giáo viên có thể thực hiện giao bài tập, bài kiểm tra ngay trong lớp theo thời gian thực.

A diagram with text and circles

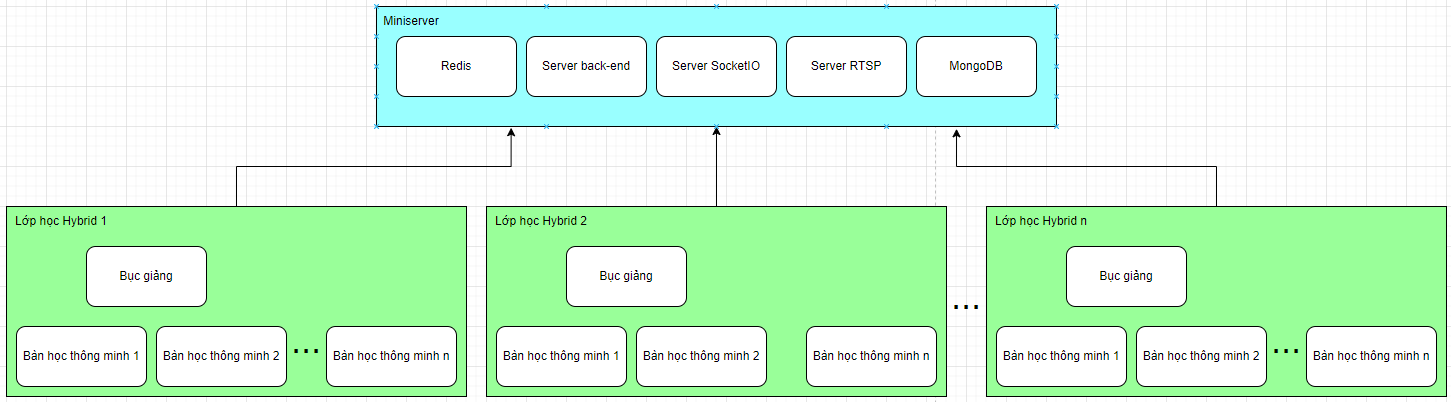
Description automatically generated with medium confidence

*Hình 3.1. Hình minh họa về thiết kế chức năng của thiết bị bục giảng*

# THIẾT KẾ SƠ ĐỒ PHẦN CỨNG

## Kiến trúc tổng quan hệ thống

Sau khi xác định các chức năng cần có và phân tích các use case, nhóm tiến vào giai đoạn thiết kế mô hình tổng quan của hệ thống



*Hình 4.1. Mô hình tổng thể của hệ thống*

Chi tiết về các thành phần trong thiết kế trên như sau:

* *Nhóm ứng dụng*: Bao gồm các phần mềm nhúng được chạy trên các thiết bị bục giảng, bàn học thông minh trong lớp học Hybrid. Đây là phần giao diện mà người dùng sẽ tương tác trực tiếp để truy cập và sử dụng các tính năng của hệ thống lớp học.
* *Nhóm máy chủ*: Đây là các máy chủ được triển khai trên nền cả tảng đám mây cũng như nền tảng local để hỗ trợ các chức năng và xử lý dữ liệu của hệ thống. Các thành phần trong nhóm máy chủ bao gồm:
* *Redis*: Được sử dụng để đồng bộ dữ liệu giữa các máy chủ trong hệ thống cân bằng tải cũng như cung cấp bộ nhớ tạm để lưu trữ dữ liệu tạm thời.
* *Máy chủ SocketIO*: Được sử dụng để tạo và quản lý kết nối thời gian thực giữa ứng dụng và các thiết bị khác nhau, đảm bảo việc truyền thông dữ liệu nhanh chóng và tin cậy.
* *Máy chủ RTSP*: Được sử dụng để truyền luồng video thời gian thực từ các thiết bị ghi hình hoặc camera đến ứng dụng và các thiết bị khác.
* *Máy chủ Back-end*: đảm nhiệm các chức năng quản lý dữ liệu người dùng, lớp học, tài liệu và các hoạt động xử lý dữ liệu khác. Nó cung cấp các API để cho phép các ứng dụng khác tương tác và trao đổi dữ liệu với hệ thống.
* *Cụm cơ sở dữ liệu MongoDB*: Đây là cơ sở dữ liệu đám mây được sử dụng để lưu trữ và quản lý dữ liệu của hệ thống lớp học. MongoDB cung cấp tính năng linh hoạt, phù hợp với mô hình dữ liệu linh hoạt trong hệ thống này.

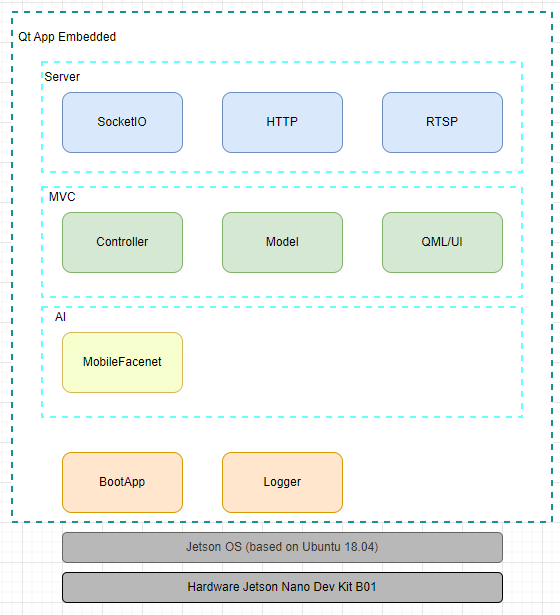
## Biểu đồ thành phần hệ thống

### Biểu đồ thành phần ứng dụng Qt

Dựa vào thiết kế tổng quan, để có thể dễ dàng quản lý và xây dựng các tính năng cho ứng dụng Qt nhóm đã tách riêng các chức năng cần thiết cho ứng dụng thành từng module riêng lẻ. Nhóm sẽ chia thành 3 phần chủ yếu gồm:

* Server: sẽ có các module như RTSP, SocketIO, HTTP dùng để ứng dụng có thể giao tiếp và truyền tải dữ liệu với Miniserver.
* MVC: đối với các chức năng phục vụ cho việc giao tiếp với người dùng, nhóm quyết định áp dụng kiến trúc phần mềm MVC (Model-View-Controller). Một mô hình thiết kế phần mềm chia ứng dụng thành 3 phần chính là Model – quản lý dữ liệu, View – hiển thị dữ liệu và giao diện người dùng, Controller – xử lý các yêu cầu từ người dùng, cập nhật Model và làm mới View. Sự tách biệt này giúp cho việc mở rộng phát triển và bảo trì ứng dụng trở nên dễ dàng hơn.
* AI: module này sẽ được sử dụng để thực hiện chức năng nhận diện khuôn mặt để có thể đăng nhập tự động

Ngoài ra, nhóm cũng triển khai riêng thêm 2 module là BootApp và Logger để tiện cho việc debug trong quá trình xây dựng phần ứng dụng Qt

**

*Hình 4.2. Biểu đồ thành phần trong phần ứng dụng Qt*

### Biểu đồ thành phần server

Chi tiết của việc thiết kế theo mô hình MVC (Model-View-Controller) được diễn tả như Hình 4.3

A diagram of a person with a diagram

Description automatically generated

*Hình 4.3. Biểu đồ thành phần theo MVC*

Chi tiết về các khối chức năng chính trong biểu đồ thành phần trên như sau:

* *Controller*: ở đây là các *Service* đại diện cho các logic xử lý và quy trình nghiệp vụ trong hệ thống. Service được sử dụng để hiện thực các chức năng cụ thể của ứng dụng. Trong thiết kế của ứng dụng đã triển khai các Service bao gồm:
  + *ClassService*: Đảm nhiệm việc quản lý thông tin của lớp học, bao gồm tạo mới lớp học, cập nhật thông tin, xóa lớp học và các chức năng khác.
  + *SubjectService*: Xử lý các việc liên quan đến môn học, bao gồm quản lý danh sách môn học, thêm mới môn học, chỉnh sửa thông tin, xóa môn học và các tác vụ khác.
  + *UserService*: Quản lý thông tin người dùng( gồm giáo viên và học sinh ), bao gồm đăng ký, đăng nhập, cập nhật thông tin cá nhân, xóa tài khoản và các chức năng khác liên quan đến người dùng trong hệ thống.
  + *DocsService*: Xử lý việc quản lý tài liệu, bao gồm tải lên, tải xuống, xóa tài liệu và các chức năng khác.
  + *AssignService*: Thực hiện các công việc liên quan đến phân công công việc, bao gồm tạo nhiệm vụ, gán nhiệm vụ cho người dùng, cập nhật thông tin và xóa nhiệm vụ đã được giao.
  + *QuestionsService*: Xử lý các việc liên quan đến câu hỏi, bao gồm tạo câu hỏi, quản lý danh sách câu hỏi, chỉnh sửa, xóa câu hỏi và các chức năng khác liên quan.
  + *DeviceService*: Đảm nhận việc quản lý các thiết bị trong hệ thống, bao gồm thêm mới thiết bị, cập nhật thông tin, xóa thiết bị và các chức năng khác liên quan đến quản lý thiết bị.
  + *AttendanceService*: Xử lý việc quản lý điểm danh, bao gồm ghi nhận điểm danh, xác nhận, cập nhật thông tin và các chức năng khác liên quan đến quản lý điểm danh trong hệ thống.
* *Model*: Đại diện cho các đối tượng và cấu trúc dữ liệu trong ứng dụng. Nó bao gồm các lớp và phương thức để định nghĩa các đối tượng, quản lý và truy xuất dữ liệu. Các Model phổ biến trong thiết kế kiến trúc web theo mô hình MVC bao gồm các lớp như *Classroom*, *Subject*, *User*, *Docs*, *Assignment*, *Question*, *Device* và *Attendance*. Các lớp Model này sẽ có các thuộc tính và phương thức để thực hiện các thao tác liên quan đến đối tượng tương ứng trong hệ thống.

Mô hình MVC (Model-View-Controller) mang đến một phương pháp tổ chức và phát triển hệ thống web rõ ràng, linh hoạt và dễ bảo trì. Việc phân chia các thành phần thành Controller, Service và Model giúp đơn giản hóa quy trình phát triển và tạo ra các module độc lập, dễ tái sử dụng.

Sự tách biệt giữa các thành phần này không chỉ giúp quy trình làm việc trở nên rõ ràng hơn mà còn tăng cường khả năng kiểm thử và quản lý mã nguồn. Mỗi phần của ứng dụng có thể được kiểm thử một cách độc lập, đảm bảo chất lượng cao hơn và giảm thiểu lỗi.

Thêm vào đó, mô hình MVC cho phép các thành viên trong nhóm phát triển làm việc độc lập trên các phần khác nhau của ứng dụng. Điều này không chỉ tăng cường hiệu suất làm việc mà còn giảm thiểu xung đột trong quá trình phát triển, tạo ra một môi trường làm việc hiệu quả và hài hòa.

### Thiết kế cơ sở dữ liệu

Nhóm đã sử dụng cơ sở dữ liệu MongoDB, một hệ quản trị cơ sở dữ liệu phi quan hệ, để không ràng buộc cấu trúc dữ liệu. Do đó, nhóm đã tập trung vào việc thiết kế các collection để tối ưu truy vấn và tiết kiệm bộ nhớ. Dưới đây là danh sách các bộ dữ liệu mà nhóm đã xác định và mô tả chúng:

* *Documents*: chứa thông tin về các tài liệu, bài giảng của giáo viên
* *Class*: chứa thông tin về các lớp học, như mã lớp, tên lớp
* *Classroom*: chứa thông tin về các phòng học chung, ví dụ như số lượng thiết bị trong phòng học.
* *Assign*: chứa cấu trúc dữ liệu về bài tập, bài kiểm tra.
* *Attendance*: chứa cấu trúc dữ liệu về việc điểm danh.
* *Question*: Bộ dữ liệu con của Assign, chứa cấu trúc dữ liệu về bộ câu hỏi trắc nghiệm, tự luận.
* *Users*: chứa thông tin về giáo viên.
* *Schedule*: chứa thông tin về lịch giảng dạy của giáo viên, lịch học của học sinh
* *Subjects*: chứa thông tin về quản lý các môn học.
* *Device*: chứa thông tin về các thiết bị được trang bị trong phòng học.
* *Noti*: chứa thông tin về các thông báo.
* *Student*: chứa thông tin về học sinh
* *Log*: chứa thông tin về thời điểm khi học sinh đăng nhập, tham gia vào lớp học.
* *Parent:* chứa thông tin của phụ huynh học sinh

Chi tiết hơn về các thuộc tính sẽ được trình bày ở hình bên dưới:

A screenshot of a computer

Description automatically generated

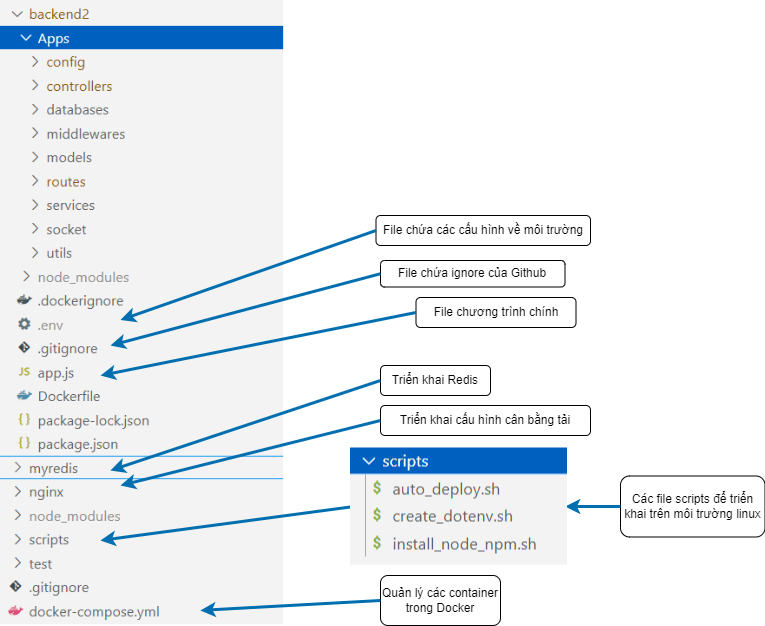
*Hình 4.4. Biểu đồ cơ sở dữ liệu*

# TRIỂN KHAI HIỆN THỰC HỆ THỐNG

## Triển khai hiện thực máy chủ

### Máy chủ API

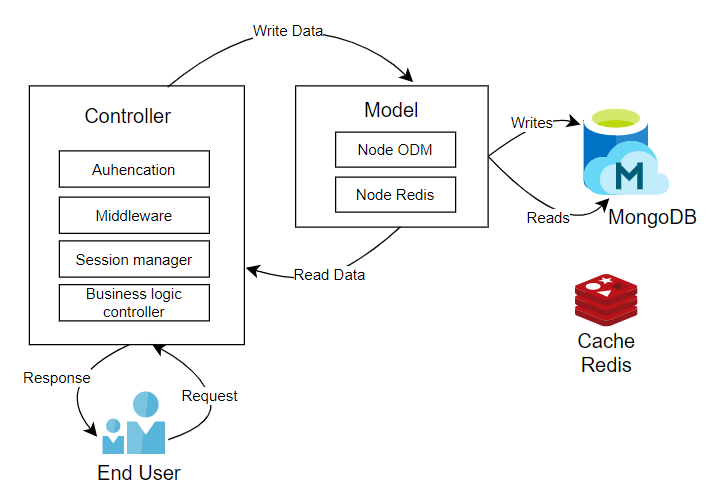
Việc đầu tiên để bắt đầu viết mã là tổ chức cấu trúc thư mục một cách chi tiết. Dưới đây là mô tả tổ chức cấu trúc thư mục được sử dụng, và mô tả chi tiết về các thư mục như sau:

**

*Hình 5.1. Mô tả về cấu trúc thư mục của API server*

* *Apps*: Đây là nơi chứa các ứng dụng chính của dự án. Mỗi ứng dụng có thể có cấu trúc thư mục riêng bên trong.
* *Config*: Thư mục này chứa các tệp cấu hình cho dự án. Đây là nơi để định nghĩa các biến môi trường, cấu hình hệ thống và các tùy chọn khác.
* *Controller*: Đây là nơi chứa các tệp điều khiển (controllers). Controllers chịu trách nhiệm xử lý các yêu cầu và gửi phản hồi cho ứng dụng.
* *Models*: Thư mục này chứa các tệp mô hình (models). Models đại diện cho các đối tượng trong dự án và định nghĩa các phương thức để tương tác với cơ sở dữ liệu hoặc thực hiện các thao tác liên quan đến dữ liệu.
* *Routes*: Thư mục này chứa các tệp định tuyến (routes). Routes xác định các tuyến đường URL và liên kết chúng với các controllers tương ứng để xử lý yêu cầu từ người dùng.
* *Tests*: Thư mục này chứa các tệp kiểm thử (tests). Tests được sử dụng để kiểm tra tính đúng đắn và hiệu suất của mã.
* *Utils*: Đây là nơi chứa các tệp tiện ích (utilities). Thư mục này thường chứa các hàm tiện ích và công cụ được sử dụng trong toàn bộ dự án.

Chi tiết về luồng xử lý khi nhận được yêu cầu từ người dùng được mô tả như hình dưới đây:



*Hình 5.2. Hình mô tả luồng xử lý của máy chủ đối với 1 API*

Quy trình xử lý yêu cầu trên máy chủ theo như hình 5.2 mô tả bao gồm các bước sau:

* *Xác thực yêu cầu*: Mỗi khi máy chủ nhận được một yêu cầu, hệ thống bắt đầu bằng việc xác thực yêu cầu để kiểm tra tính hợp lệ của nó. Nếu yêu cầu không hợp lệ, hệ thống sẽ ngay lập tức trả về kết quả thất bại và dừng quá trình xử lý.
* *Kiểm tra phân quyền*: Nếu yêu cầu đã được xác thực thành công, hệ thống tiếp tục bước kiểm tra phân quyền. Trong bước này, hệ thống kiểm tra xem người dùng đã được cấp phép truy cập vào tài nguyên hệ thống mà yêu cầu liên quan thông qua API tương ứng hay không. Nếu người dùng không có quyền truy cập, hệ thống sẽ từ chối yêu cầu và trả về kết quả thích hợp.
* *Lưu trữ phiên làm việc*: Nếu yêu cầu đã vượt qua bước kiểm tra phân quyền, hệ thống tiếp tục bằng việc lưu trữ phiên làm việc. Điều này đảm bảo rằng thông tin và trạng thái của yêu cầu được theo dõi và duy trì trong suốt quá trình xử lý.
* *Truy cập và truy vấn cơ sở dữ liệu*: Sau khi phiên làm việc được lưu trữ, controller bắt đầu truy cập và truy vấn cơ sở dữ liệu. Hệ thống sử dụng mô hình (Model) để tương tác với cơ sở dữ liệu và truy xuất thông tin cần thiết để xử lý yêu cầu. Quá trình này có thể bao gồm truy vấn, cập nhật hoặc xóa dữ liệu.
* *Trả về kết quả*: Sau khi truy vấn cơ sở dữ liệu hoàn thành, hệ thống trả về kết quả cho người dùng. Kết quả có thể là dữ liệu được truy vấn từ cơ sở dữ liệu, thông báo thành công hoặc các mã lỗi tương ứng.

Để minh họa rõ hơn chi tiết cho các bước ở trên thì đọc giả có thể tham khảo qua hai biểu đồ tuần tự ở hình 5.3 và hình 5.4 bên dưới đây.

A picture containing text, diagram, screenshot, parallel

Description automatically generated

*Hình 5.3. Biểu đồ tuần tự để xác thực request*

Nội dung ở trên đã trình bày quy trình xử lý yêu cầu trên máy chủ thông qua các bước xác thực, kiểm tra phân quyền, lưu trữ phiên làm việc, truy cập và truy vấn cơ sở dữ liệu, và trả về kết quả cho người dùng. Qua đó, hệ thống có thể xử lý các yêu cầu từ người dùng một cách an toàn và hiệu quả, đảm bảo đầy đủ các tính chất về mặt bảo mật và an toàn thông tin.

### Triển khai tính năng giao tiếp thời gian thực với SocketIO

#### Triển khai phía máy chủ SocketIO

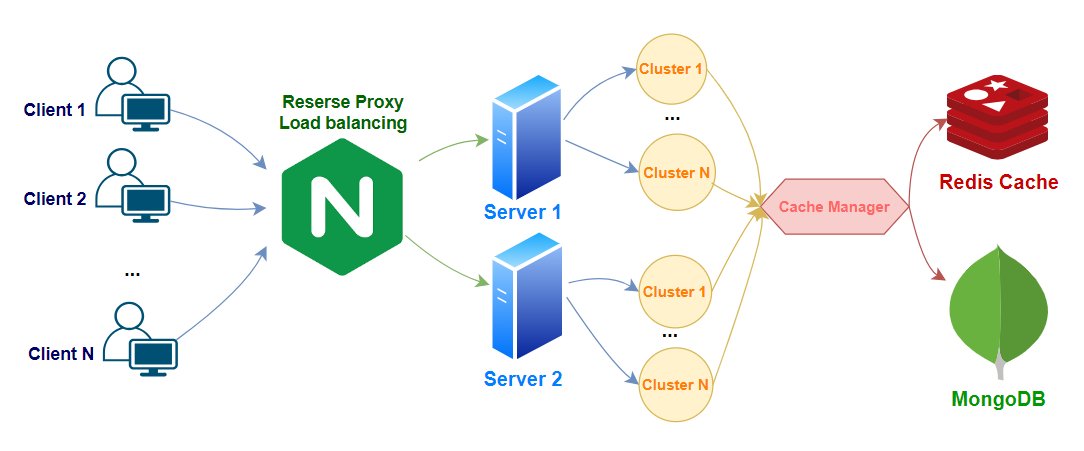
Để xử lý lượng kết nối lớn và đảm bảo khả năng mở rộng tự động khi nhu cầu tăng, nhóm nghiên cứu đã triển khai một loạt giải pháp và công nghệ tiên tiến. Đầu tiên, nhóm đã chọn Nginx, một web server mã nguồn mở, để làm máy chủ proxy. Với khả năng cân bằng tải, Nginx phân phối lưu lượng kết nối đồng đều giữa các máy chủ ứng dụng SocketIO, từ đó tăng cường hiệu suất hệ thống.



*Hình 5.4: Cấu hình nginx cân bằng tải*

Nhằm tối ưu hóa khả năng xử lý đồng thời, nhóm đã áp dụng kỹ thuật thiết lập nhiều cluster. Bằng cách này, ứng dụng có thể chạy nhiều tiến trình NodeJS độc lập trên cùng một máy chủ, mỗi tiến trình sẽ xử lý một phần lưu lượng kết nối. Kỹ thuật này không chỉ giúp chia sẻ tải công việc mà còn nâng cao khả năng đáp ứng của hệ thống.

Thêm vào đó, Redis đã được tích hợp như một bộ nhớ đệm để tăng tốc độ phản hồi. Redis lưu trữ các dữ liệu tạm thời như phiên đăng nhập, thông tin người dùng và trạng thái kết nối, giúp giảm tải cho cơ sở dữ liệu chính và cung cấp truy cập nhanh hơn vào dữ liệu cần thiết trong quá trình giao tiếp với SocketIO.

Về tổng thể, hệ thống được xây dựng dựa trên ExpressJS để hiện thực máy chủ SocketIO, kết hợp với Nginx để cân bằng tải và Redis để tăng tốc độ phản hồi. Thiết kế này không chỉ đáp ứng lượng kết nối lớn mà còn có khả năng mở rộng linh hoạt. Kỹ thuật thiết lập nhiều cluster giúp xử lý đồng thời và phân chia tải công việc, nâng cao hiệu suất ứng dụng. Hình dưới đây minh họa kiến trúc tổng quan của hệ thống máy chủ SocketIO.

*Hình 5.5. Mô hình triển khai của máy chủ SocketIO*

#### Triển khai Client

Việc triển khai lớp SocketIoClient thông qua C++ cho phép tương tác với các client khác thông qua giao thức SocketIO. Dưới đây là một ví dụ về giao diện để triển khai lớp SocketIoClient trong C++:

A screenshot of a computer code

Description automatically generated with medium confidence

*Hình 5.6. Hình minh họa về bộ interface cho lớp SocketIO-client*

A picture containing text, screenshot, font, number

Description automatically generated

*Hình 5.7. Hình minh họa về triển khai lắng nghe một sự kiện qua socket*

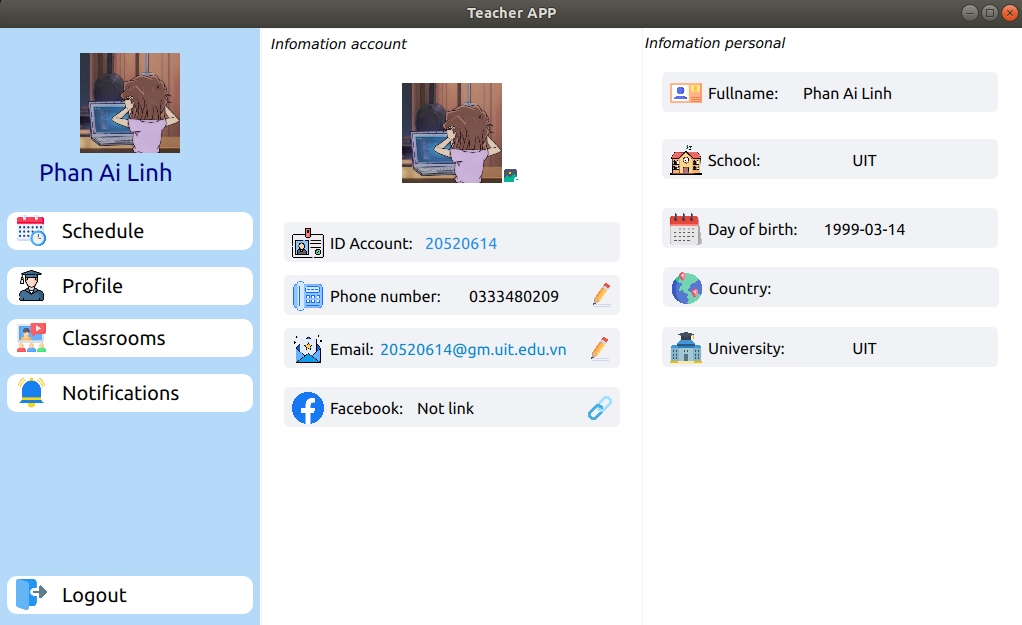
## Triển khai giao diện ứng dụng nhúng ban đầu

Trước tiên, để có thể triển khai các chức năng thì ta cần xây dựng phần giao diện người dùng sau đó sẽ xác định những chức năng cần có để phục vụ cho việc tương tác của người dùng với ứng dụng.

Nhóm sử dụng C++ là ngôn ngữ lập trình chính để viết mã logic phức tạp và xử lý dữ liệu và kết hợp với QML để xây dựng UI. C++ có ưu điểm về hiệu suất và tối ưu hóa tốt, cho phép nhóm thực hiện các tác vụ phức tạp một cách hiệu quả và tích hợp linh hoạt với các thành phần hệ thống khác.

Dưới đây là một số hình ảnh giao diện ứng nhóm đã hiện thực:

A screenshot of a computer

Description automatically generated 

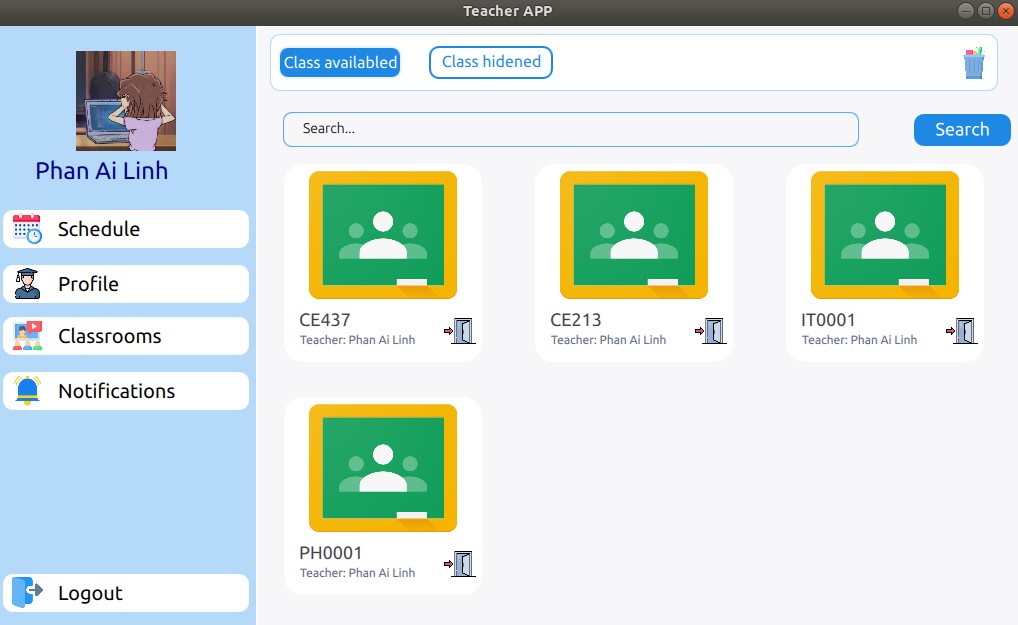
*Hình 5.8. Giao diện trang đăng nhập và trang thông tin cá nhân*

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

*Hình 5.9. Giao diện trang lịch học*

 A screenshot of a computer

Description automatically generated

*Hình 5.10. Giao diện trang lớp học và bài giảng*

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

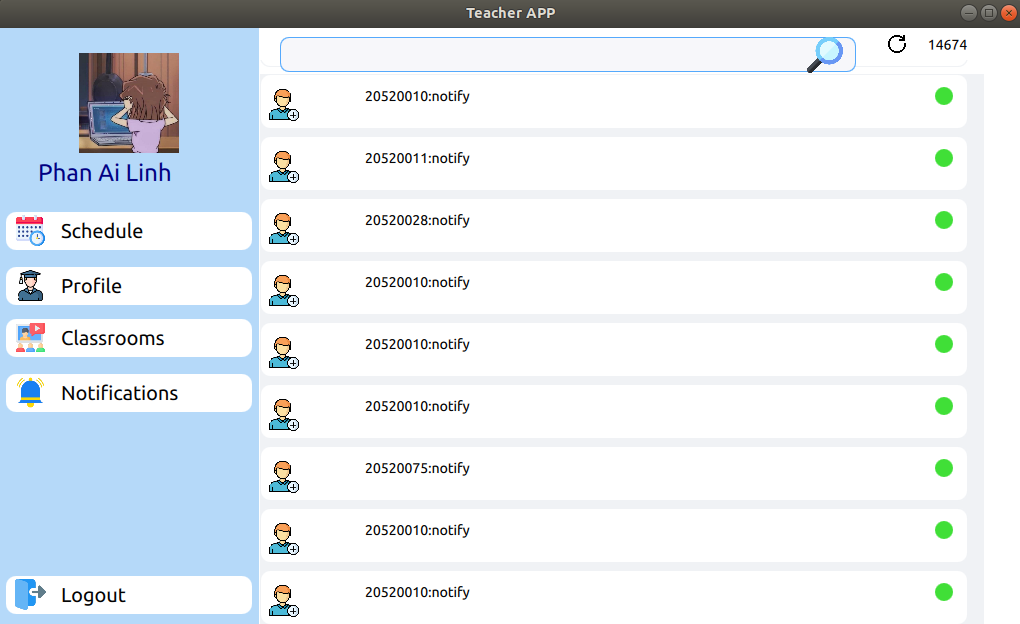
*Hình 5.11. Giao diện trang bài tập*

A screenshot of a computer

Description automatically generated A screenshot of a computer

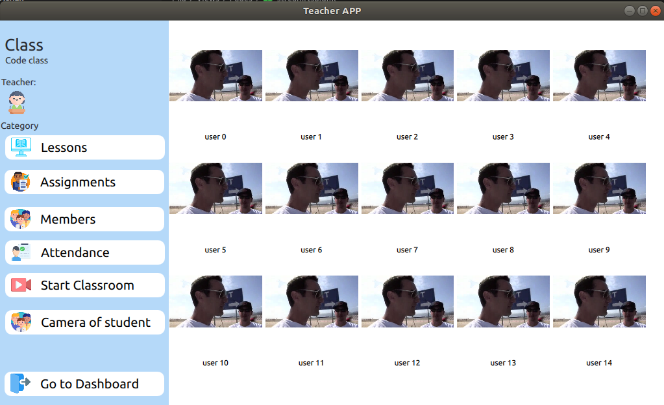
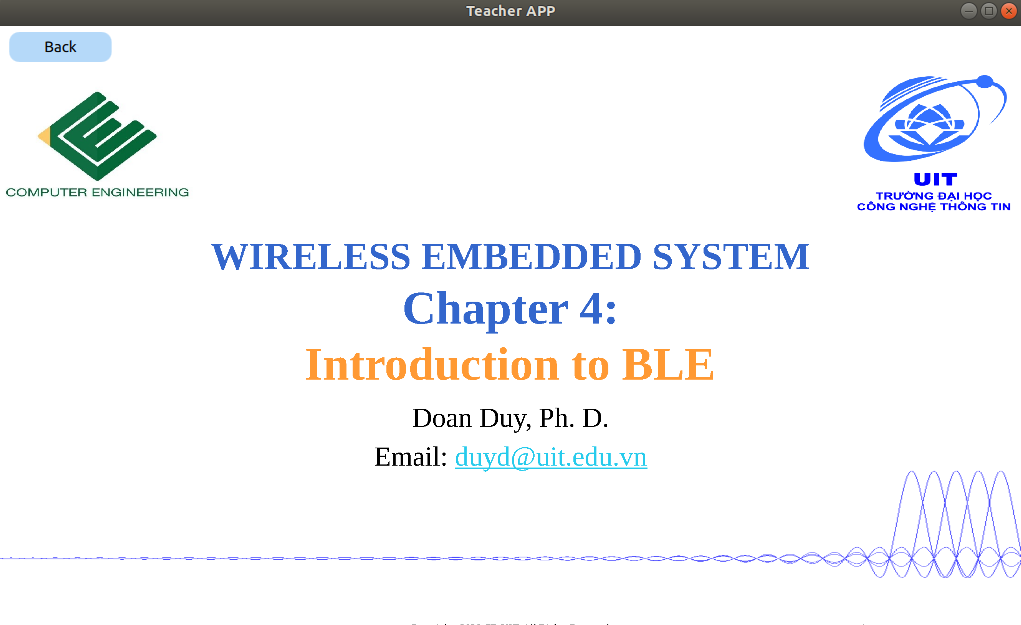
Description automatically generated

*Hình 5.12. Giao diện trang danh sách học sinh và điểm danh*

 A screenshot of a computer

Description automatically generated with medium confidence

*Hình 5.13. Giao diện trang thông báo*

*Hình 5.14. Giao diện trang xem stream video và mở bài giảng*

## Triển khai module nhận diện khuôn mặt

### Chuẩn bị bộ dữ liệu

Việc chuẩn bị bộ dữ liệu cho việc nhận diện khuôn mặt là rất quan trọng vì một số lý do sau đây:

* Đảm bảo độ đa dạng: Bộ dữ liệu cần phải bao gồm đủ nhiều các tình huống và điều kiện khác nhau mà mô hình có thể gặp phải trong thực tế, như ánh sáng yếu, góc chụp khác nhau, và các biến thể về diện mạo của khuôn mặt.
* Độ phủ mẫu (Sample Coverage): Bộ dữ liệu cần phải đủ lớn để bao quát đủ các đặc trưng của khuôn mặt, từ các nhóm tuổi, giới tính, đến sắc tộc và đặc điểm cá nhân khác.
* Chất lượng và sự chuẩn xác: Việc gán nhãn và chuẩn bị dữ liệu phải được thực hiện một cách cẩn thận để đảm bảo sự chính xác và độ tin cậy cao. Mô hình chỉ có thể học được những gì mà dữ liệu huấn luyện cung cấp, vì vậy dữ liệu không chính xác hoặc không đủ đa dạng có thể dẫn đến mô hình không hoạt động đúng cách.
* Tăng cường dữ liệu (Data Augmentation): Quá trình tiền xử lý dữ liệu và tăng cường dữ liệu làm giàu bộ dữ liệu bằng cách áp dụng các biến đổi như xoay, phóng to, thay đổi màu sắc, giúp mô hình học được khả năng tổng quát hóa tốt hơn đối với các điều kiện khác nhau.
* Đánh giá hiệu suất và cải tiến mô hình: Bộ dữ liệu chất lượng cao là cơ sở để đánh giá hiệu suất của mô hình và điều chỉnh các tham số để cải thiện độ chính xác và độ phân biệt của mô hình nhận diện khuôn mặt.

Với những lý do trên thì nhóm đã tìm kiếm và thu thập được những bộ dữ liệu từ Internet như sau:

|  |  |  |
| --- | --- | --- |
| **Bộ dữ liệu** | **Thông số** | **Ghi chú** |
| FaceNet (Google) | ~200,000,000 ảnh  8,000,000 định danh | Công khai |
| MS-Celeb-1M | 10,000,000 ảnh  100,000 định danh | Công khai |
| MegaFace | 4,700,000 ảnh  672,000 định danh | Công khai |
| Sicial Face Classification (Facebook) | 4,400,000 ảnh  4,030 định danh | Công khai |
| **CASIA-WebFace** | 494,414 ảnh  10,575 định danh | Công khai |
| **LFW** | 13,233 ảnh  5,749 định danh | Công khai |
| **Age-DB 30** | 16,488 ảnh  568 định danh | Công khai |
| **VN-Celeb** | ~23,000 ảnh 1000 định danh | Bộ dữ liệu khuôn mặt người Việt |

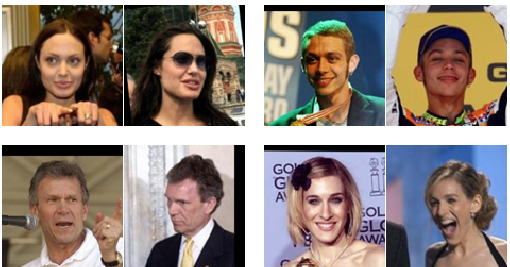
Sau khi thống kê và cân nhắc tài nguyên, nhóm đã lựa chọn các bộ dữ liệu với mục đích sử dụng như sau:

* *CASIA-WebFace*: Được sử dụng để huấn luyện mô hình. Bộ dữ liệu này được xây dựng từ các hình ảnh trang web.



*Hình 5.15. Một số mẫu dữ liệu trong tập CASIA-Webface*

* *LFW*: Được sử dụng để xác thực mô hình trong quá trình huấn luyện. Việc xác minh khuôn mặt với bộ dữ liệu này bằng cách đặt từng cặp ảnh và dựa vào việc phân loại nhị phân để dự đoán xem hai ảnh đó có cùng một người hay không.



* *Age-DB 30*: Được sử dụng để kiểm thử mô hình. Bộ dữ liệu này được thu thập với nhiều độ tuổi khác nhau.
* 

*Hình 5.16. Một số mẫu dữ liệu trong tập LFW(trái) và AgeDB-30 (phải)*

* *VN-Celeb*: Được sử dụng để tinh chỉnh (fine-tuning) mô hình đã được huấn luyện sử dụng bộ dữ liệu CASIA-WebFace.



*Hình 5.17. Một số mẫu về tập VN-Celeb*

Điều này giúp đảm bảo rằng mô hình được huấn luyện với dữ liệu đa dạng và đủ chất lượng, và sau đó được kiểm tra và điều chỉnh để đạt hiệu suất tốt hơn trên các bộ dữ liệu khác nhau.

### Huấn luyện mô hình

Sau khi hoàn tất việc chuẩn bị bốn bộ dữ liệu, nhóm bắt đầu quá trình huấn luyện, chia thành hai giai đoạn:

* Giai đoạn 1: Huấn luyện trên bộ dữ liệu lớn CASIA-WebFace, xác thực qua bộ dữ liệu LFW và kiểm thử trên bộ AgeDB-30.
* Giai đoạn 2: Dựa trên kết quả của giai đoạn 1, tinh chỉnh mô hình bằng bộ dữ liệu khuôn mặt Việt Nam.

Trong cả hai giai đoạn, sự khác biệt chính nằm ở bộ dữ liệu đầu vào, còn về quy trình huấn luyện, cả hai đều khá tương đồng. Chi tiết quy trình huấn luyện như sau:

* Chuẩn bị và tiền xử lý dữ liệu:
* Nhóm sử dụng module torch.utils.data.DataLoader để viết class Dataloader cho ba bộ dữ liệu đã chuẩn bị.
* Ảnh được load bằng OpenCV và có cài đặt shuffle trong Dataloader để đảm bảo dữ liệu được xáo trộn ngẫu nhiên.
* Quá trình huấn luyện diễn ra trên nền tảng Google Colab (GPU Tesla K80) với bộ nhớ GPU 12GB, nhóm cài đặt batch size là 128 để phù hợp với bộ nhớ.
* Huấn luyện mô hình:
* Bắt đầu vòng lặp huấn luyện qua các bước lan truyền xuôi, tính giá trị hàm mất mát, sau đó lan truyền ngược.
* Nhóm sử dụng hàm mất mát ArcFace và hàm tối ưu SGD (Stochastic Gradient Descent) với momentum là 0.9, learning rate là 0.1, chia đều tại các iter 36,000, 52,000 và 58,000.
* Đánh giá mô hình:
* Đánh giá trên bộ dữ liệu LFW bằng phương pháp Ma trận nhầm lẫn (Confusion Matrix).
* Quá trình này diễn ra sau mỗi 1000 iter để theo dõi tiến trình huấn luyện.
* Sau khi hoàn thành huấn luyện, mô hình cuối cùng được kiểm thử trên tập dữ liệu AgeDB-30 để kiểm chứng độ chính xác.

Qua hai giai đoạn huấn luyện, nhóm đã tuân thủ đầy đủ các bước trong quy trình phát triển một mô hình học máy cơ bản. Biểu đồ tuần tự chi tiết được trình bày ở hình 42 bên dưới, và kết quả thu được sau hai giai đoạn sẽ được trình bày trong mục tiếp theo.

A picture containing text, diagram, technical drawing, plan

Description automatically generated

*Hình 5.18. Biểu đồ tuần tự thực hiện huấn luyện mô hình*

### Kết quả huấn luyện

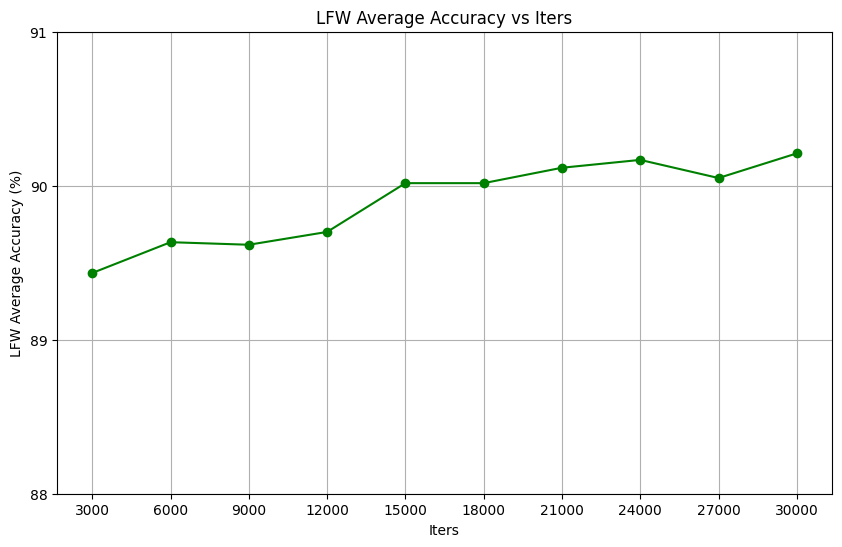
Sau quá trình huấn luyện với hai giai đoạn như trên thì nhóm có kết quả như sau:

* Kết quả sau khi kết thúc giai đoạn một:

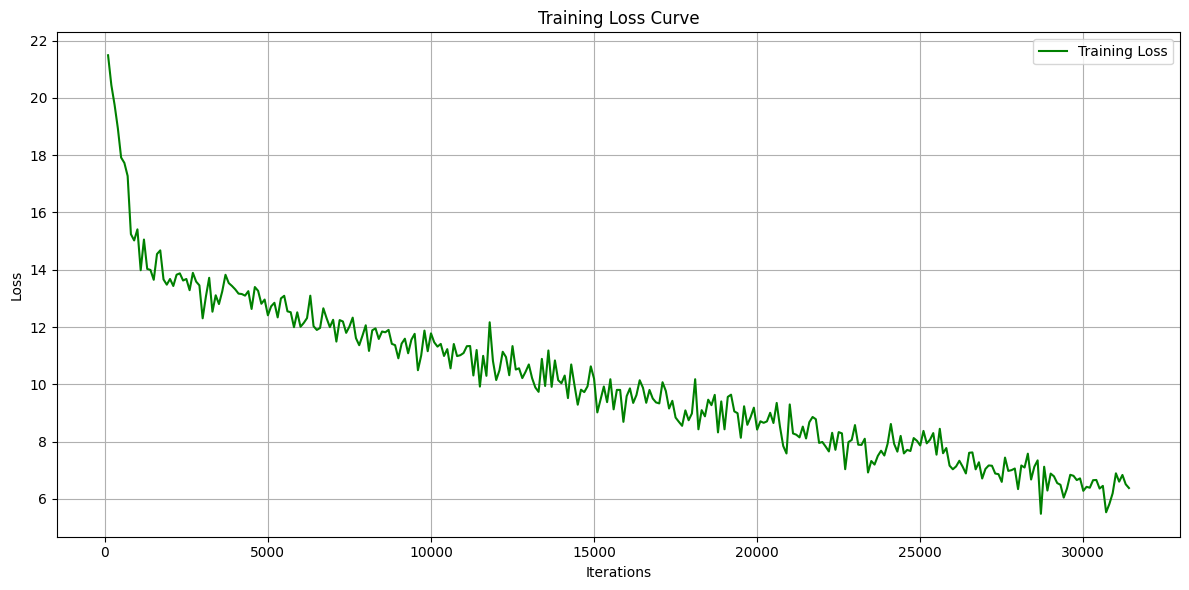
|  |  |
| --- | --- |
| Tập dữ liệu kiểm tra | Độ chính xác nhận diện |
| LFW | 98.91% |
| AgeDB-30 | 89.50% |

* Kết quả sau khi tinh chỉnh mô hình bằng bộ dữ liệu mặt người Việt Nam:

|  |  |
| --- | --- |
| Tập dữ liệu kiểm tra | Độ chính xác |
| 20% bộ VN-Celeb | 90.21% |



*Hình 5.19. Kết quả kiểm thử độ chính xác sau khi tinh chỉnh trên bộ dữ liệu người VN*

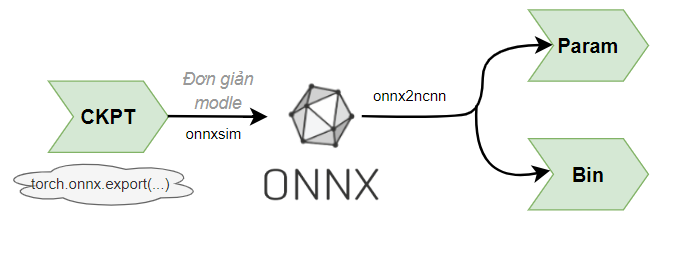


*Hình 5.20. Kết quả hàm mất mát giảm trong khi tinh chỉnh trên bộ dữ liệu người VN*

### Chuyển đổi mô hình xuống nền tảng nhúng

Để hiện thực hóa giải thuật của mô hình xuống dưới C++, chúng ta cần chuyển đổi định dạng từ PyTorch sang nền tảng NCNN. Quy trình chuyển đổi này bao gồm các bước sau:

* Xuất mô hình từ PyTorch sang ONNX:
* Từ PyTorch, chúng ta sẽ xuất mô hình dưới dạng ONNX.
* Chuyển đổi từ ONNX sang NCNN:
* ONNX là định dạng trung gian, từ đây chúng ta tiếp tục chuyển đổi sang định dạng của NCNN.
* Do NCNN chưa hỗ trợ các toán tử dư thừa như Shape, Gather hay Unsqueeze, chúng ta cần sử dụng thư viện "onnx-simplifier" để thay thế các toán tử này bằng các giá trị không đổi tương ứng.
* Sử dụng công cụ onnx2ncnn:
* Khi đã có mô hình ONNX được chuẩn hóa cho NCNN, chúng ta sử dụng bộ công cụ "onnx2ncnn" do nền tảng NCNN cung cấp để chuyển đổi mô hình sang định dạng weights của NCNN.
* Kết quả chuyển đổi sẽ là hai file có định dạng .param và .bin.

Bộ công cụ "onnx2ncnn" này được build cùng với source của NCNN, giúp chúng ta dễ dàng chuyển đổi và tích hợp mô hình vào ứng dụng C++. 

*Hình 5.21. Các bước chuyển đổi định dạng pth qua param và bin*

### Hiện thực trên C++ để tính hợp vào ứng dụng Qt

Trong quá trình triển khai việc coding thì nhóm đã tách nhận diện khuôn mặt ra làm một mô đun riêng và đoạn mã ở bên dưới cho chúng ta thấy interface cho một lớp *FaceCheckService* trong hệ thống nhận dạng khuôn mặt. Lớp này cung cấp các phương thức để khởi tạo, tải tài nguyên, đăng nhập, nhận dạng và tạo mẫu cho việc nhận dạng khuôn mặt.

A screenshot of a computer program

Description automatically generated with medium confidence

*Hình 5.22. Hình mô tả code interface cho class FaceService*

Như minh họa trong hình 5.31 ở trên thì bộ interface được thiết kế với các phương thức như *init()*, *getResources()*, *loginWithFace*() và *recognize()* để tương tác với dữ liệu và thực hiện các công việc liên quan đến nhận dạng khuôn mặt. Đối tượng của lớp được tạo dưới dạng Singleton, đảm bảo rằng chỉ có một phiên bản của FaceCheckService được tạo ra. Các biến thành viên *asf*, *mtcnn* và *mbfn* được lưu trữ bằng con trỏ thông minh, giúp quản lý tự động vòng đời của các đối tượng. Điều này cho phép tải và sử dụng các mô hình nhận dạng khuôn mặt tùy chỉnh. Các vector nameFaces và fc1 lưu trữ tên và nhúng của các khuôn mặt đã được đăng ký trong cơ sở dữ liệu.

A picture containing text, diagram, plan, parallel

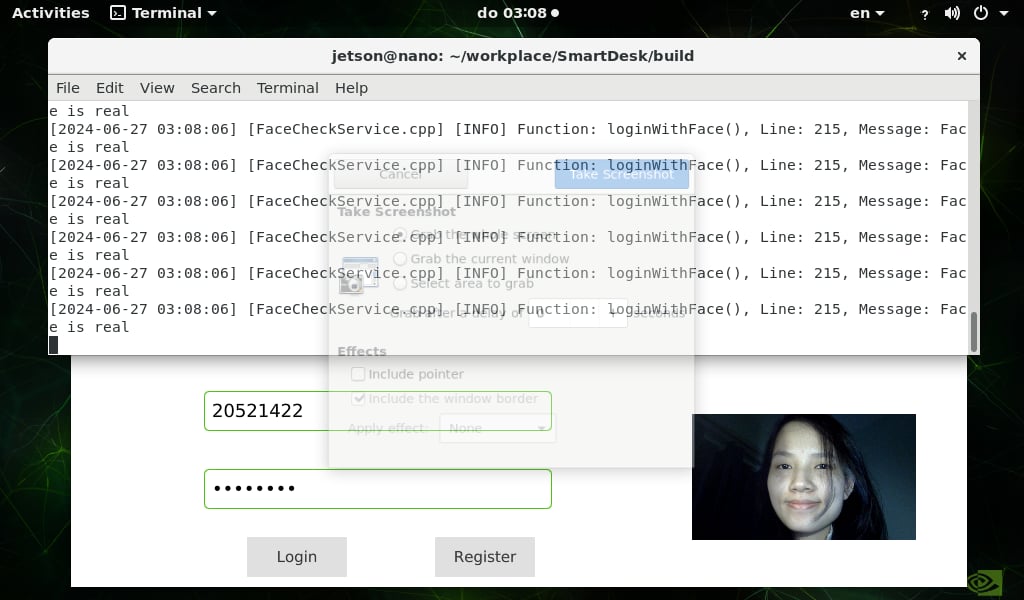
Description automatically generated

*Hình 5.23. Biểu đồ tuần tự xử lý nhận diện khuôn mặt*

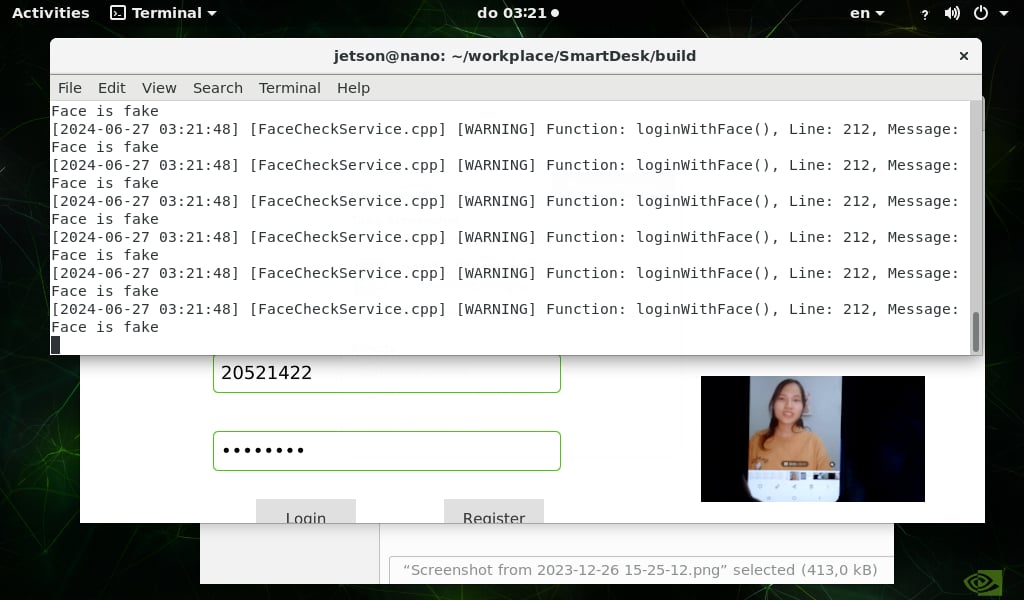
### Chuyển đổi mô hình Anti-Spoofing từ Python xuống C++

Dựa trên quy trình chuyển đổi từ các nền tảng Python sang nền tảng C++, nhóm đã sử dụng bộ tham số mã nguồn mở là Minivision-AI về Anti-Spoofing. Thông qua quá trình chuyển đổi như trên, nhóm đã đạt được các kết quả sau:

* **Kết quả:**



*Hình 5.24. Kết quả nhận dạng khuôn mặt là thật*



*Hình 5.25: Kết quả nhận dạng khuôn mặt là giả với hình ảnh trên điện thoại*

## Triển khai stream video với RTSP

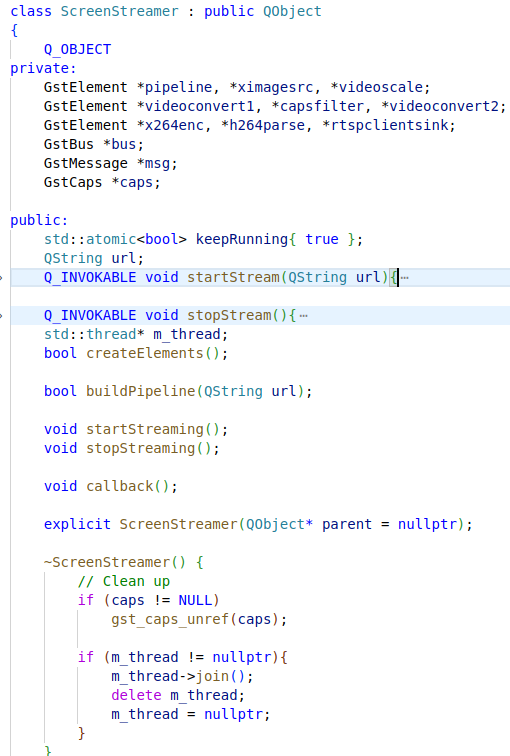
Việc học trực tuyến và triển khai một lớp học hybrid đòi hỏi sự linh hoạt và khả năng tương tác giữa giảng viên và học viên. Để xây dựng một server RTSP từ đầu có thể mất rất nhiều thời gian và công sức, vậy nên nhóm đã quyết định sử dụng MediaMTX, một server mã nguồn mở.

MediaMTX (trước đây là rtsp-simple-server) là một server truyền thông thời gian thực và proxy media cho phép publish, read, proxy, record và playback các luồng video và âm thanh.

MediaMTX được thiết kế như một "media router" để định tuyến các luồng media từ một đầu đến một đầu khác. Các tính năng nổi bật của MediaMTX bao gồm:

* Publish live streams: Cho phép publish các luồng trực tiếp lên server.
* Read live streams: Cho phép đọc các luồng trực tiếp từ server.
* Protocol conversion: Tự động chuyển đổi giữa các giao thức.
* Multiple streams: Hỗ trợ nhiều luồng cùng lúc trên các đường dẫn riêng biệt.
* Record streams: Ghi lại các luồng vào đĩa.
* Playback recorded streams: Phát lại các luồng đã ghi.
* User authentication: Xác thực người dùng.
* Load balancing: Chuyển hướng người đọc đến các server RTSP khác.
* Control API: Điều khiển server thông qua API.
* Hot reloading: Tải lại cấu hình mà không ngắt kết nối các client hiện tại.
* Prometheus-compatible metrics: Đọc các số liệu tương thích với Prometheus.
* Hooks: Chạy các lệnh bên ngoài khi client kết nối, ngắt kết nối, đọc hoặc publish các luồng.
* Cross-platform compatibility: Tương thích với Linux, Windows và macOS, không yêu cầu bất kỳ thư viện hoặc trình thông dịch nào, chỉ cần một tệp thực thi duy nhất.

Về phía ứng dụng Qt, để có thể stream video lên server thì nhóm đã sử dụng GStreamer, một thư viện phổ biến để xử lý các chức năng liên quan đến video cũng như audio ở thiết bị nhúng. Dưới đây là một ví dụ để triển khai lớp ScreenStreamer, một lớp dùng để stream video màn hình của ứng dụng bằng GStreamer trong C++:



*Hình 5.26. Hình minh họa về bộ interface cho lớp ScreenStreamer*

Thành phần của lớp ScreenStreamer như sau:

* Thành phần dữ liệu:
  + **pipeline**: Pipeline GStreamer chính.
  + **ximagesrc, videoscale, videoconvert1, capsfilter, videoconvert2, x264enc, h264parse, rtspclientsink**: Các phần tử của pipeline dùng để lấy màn hình, chuyển đổi định dạng, mã hóa và gửi qua RTSP.
  + **bus, msg:** Để xử lý các thông điệp GStreamer (ví dụ: lỗi, EOS).
  + **caps**: Định dạng và giới hạn cho dữ liệu video.
  + **keepRunning**: Biến điều khiển để giữ cho luồng callback chạy.
  + **url**: Địa chỉ RTSP để stream.
  + **m\_thread**: Con trỏ đến luồng thực hiện việc stream.
* Các phương thức hỗ trợ cho việc stream video:
  + **createElements()**: Tạo các phần tử GStreamer cần thiết cho pipeline. Nếu không thể tạo bất kỳ phần tử nào, sẽ ghi log lỗi và trả về false.
  + **buildPipeline(QString url)**: Thêm các phần tử vào pipeline và liên kết chúng lại với nhau. Thiết lập các thuộc tính mặc định cho các phần tử. Đặt URL cho phần tử *rtspclientsink* để gửi stream đến đó.
  + **startStreaming()**: Đặt trạng thái của pipeline thành *GST\_STATE\_PLAYING.* Chờ cho đến khi nhận được thông điệp lỗi hoặc EOS (End of Stream). Giải phóng các tài nguyên khi kết thúc.
  + **stopStreaming()**: Gửi sự kiện EOS (End of Stream) đến pipeline để kết thúc stream. Ghi log thông tin nếu thành công hoặc thất bại.
  + **callback()**: Vòng lặp liên tục tạo các phần tử, xây dựng pipeline và bắt đầu stream cho đến khi *keepRunning* được đặt thành *false*.

## Triển khai module Logger lưu lại nhật ký ứng dụng

Trong quá trình xây dựng ứng dụng thì việc gỡ lỗi có thể gặp khó khăn, do chương trình có thể bị dừng đột ngột và không được cung cấp đủ thông tin chi tiết về nguyên nhân, tạo ra nhiều khó khăn trong quá trình gỡ lỗi. Để giải quyết vấn đề này, nhóm đã triển khai một module Logger để ghi lại log hoạt động của chương trình, giúp dễ dàng theo dõi và phân tích lỗi trong quá trình phát triển. Dưới đây là bản triển khai của module Logger:



*Hình 5.27. Triển khai module Logger*

A screenshot of a computer screen

Description automatically generated

*Hình 5.28. Hình minh họa về nhật ký được lưu lại khi khởi động ứng dụng*

## Triển khai ứng dụng xuống thiết bị phần cứng

Trong quá trình phát triển ứng dụng nhúng, việc chuyển từ môi trường máy tính sang môi trường phần cứng nhúng luôn gặp khó khăn và tiềm ẩn nhiều rủi ro. Để giảm thiểu rủi ro, nhóm đã thực hiện những cải tiến quan trọng.

Việc thay đổi môi trường chạy ứng dụng từ máy tính sang thiết bị phần cứng thường gặp khó khăn và tiềm ẩn rủi ro do có nhiều thành phần sẽ không tương thích. Để giảm thiểu những rủi ro này, nhóm đã thực hiện như sau:

* Đầu tiên, nhóm đã chọn hệ điều hành trên máy tính cho việc phát triển ứng dụng là Ubuntu 18.04, tương thích với phiên bản hệ điều hành mặc địch của Nvidia cho Jetson Nano, và lựa chọn phiên bản của Qt, OpenCV là mặc định được biện dịch kèm theo hệ điều hành trong Jetson Nano.
* Để tiết kiệm thời gian cho việc cài đặt các thư viện cần thiết để chạy ứng dụng thì nhóm đã tạo ra các tệp thực thi tự động để cài đặt cho Jetson Nano hoặc trên một máy tính khác. Những tệp thực thi này hỗ trợ rất nhiều khi cần cài đặt lại môi trường ban đầu cho ứng dụng trên nhiều thiết bị khác nhau. Ví dụ về tập thực thi tự động được minh họa ở các hình sau:

A screenshot of a computer program

Description automatically generated

*Hình 5.29. Script cài đặt NCNN*

A screenshot of a computer

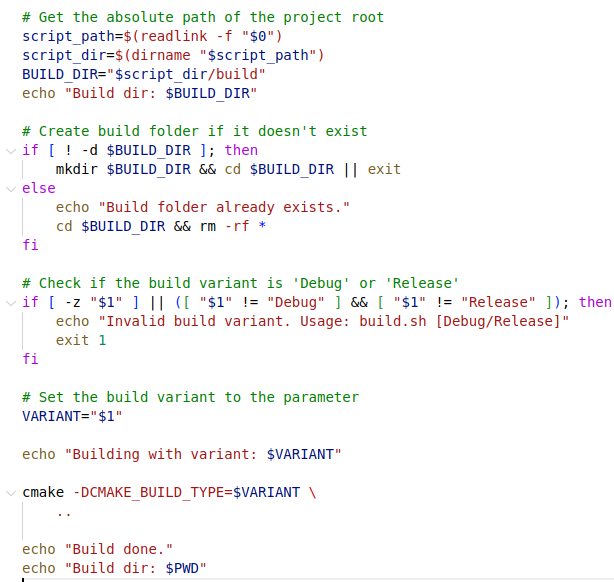
Description automatically generated

*Hình 5.30. Script cài đặt OpenCV 4.5.3*

A close-up of a white background

Description automatically generated

*Hình 5.31. Script cài đặt Qt 5.9.5*



*Hình 5.32. Script biên dịch chương trình bằng CMake*

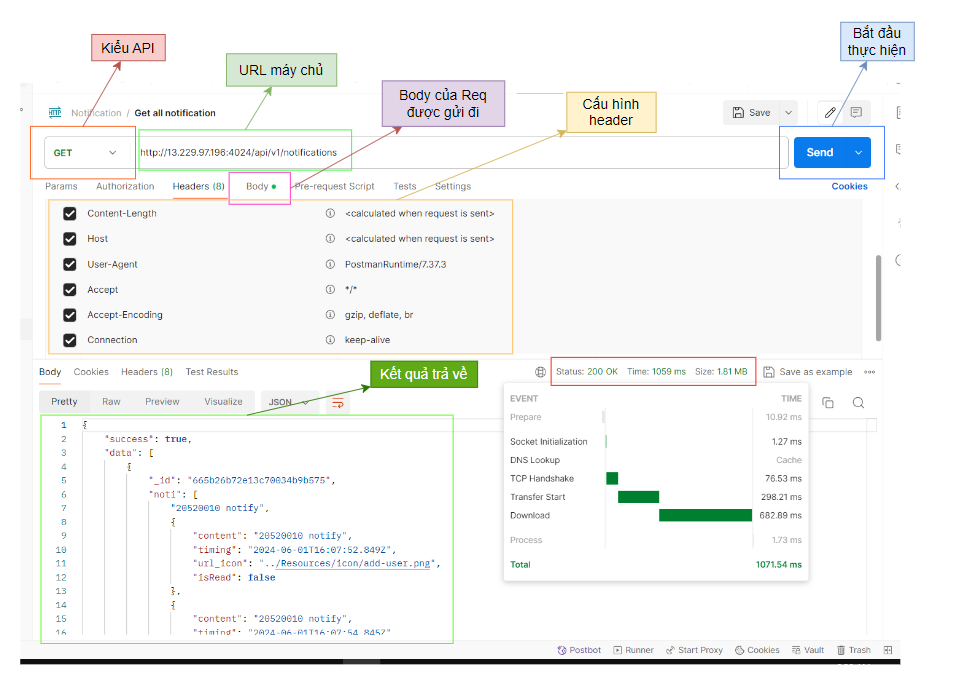
A screen shot of a computer

Description automatically generated with medium confidence

*Hình 5.33. Hình ảnh ứng dụng Qt chạy trên thiết bị*

# THỰC HIỆN KIỂM THỬ VÀ LÀM TÀI LIỆU

## Triển khai kiểm thử API với Postman

Đối với kịch bản kiểm thử thì vì thời gian có hạn nên nhóm chỉ kiểm thử riêng biệt từng API mà không tiến hành kiểm thử tích hợp. Mục tiêu nhóm đặt ra là API hoạt động đúng logic và trả về dữ liệu đúng đắn. Hình ảnh về việc kiểm thử cho 1 API trên phần mềm được biểu diễn như hình dưới đây:  


*Hình 6.1. Hình mô tả quá trình thực hiện kiểm thử API trên Postman*

Kết quả như trên cho thấy rằng API hoạt động đúng đắn và ổn định. Đảm bảo được dữ liệu cũng như tính logic của cơ sở dữ liệu muốn truy vấn.

## Triển khai kiểm thử khả năng chịu tải máy chủ SocketIO

Trong phần này, nhóm đã tiến hành đánh giá khả năng chịu tải của máy chủ Socket.io. Mục tiêu của chúng ta là tạo ra một môi trường kiểm thử mô phỏng tải cao, với hàng loạt kết nối đồng thời đến máy chủ để đo lường hiệu suất của nó.

Kiểm thử khả năng chịu tải là một nhiệm vụ đầy thách thức, đòi hỏi một hệ thống kiểm thử mạnh mẽ và đáng tin cậy. Chẳng hạn, việc mô phỏng hàng ngàn người dùng ảo gửi sự kiện đến máy chủ cùng một lúc cần một cấu hình hệ thống mạnh mẽ. Tuy nhiên, do hạn chế về tài nguyên tại thời điểm thực hiện, nhóm chỉ tiến hành kiểm thử với ba tiêu chí chính:

* Kết Nối Đồng Thời: Đánh giá khả năng xử lý của máy chủ khi phải duy trì nhiều kết nối cùng lúc.
* Xử Lý Sự Kiện: Kiểm tra hiệu suất của máy chủ khi phải xử lý một lượng lớn sự kiện từ các kết nối.
* Tải Nặng: Đánh giá khả năng của máy chủ khi phải chịu tải nặng liên tục trong thời gian dài.

Bằng cách tập trung vào những khía cạnh này, chúng ta sẽ có cái nhìn rõ ràng về khả năng chịu tải và hiệu suất của máy chủ Socket.io dưới áp lực cao.Về môi trường kiểm thử thì nhóm thực hiện trên cấu hình máy chủ như sau:

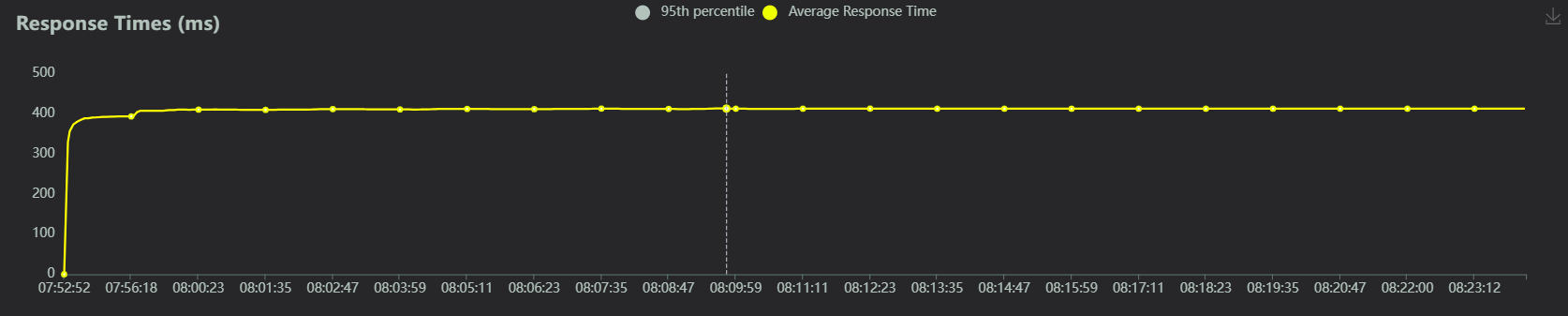
|  |  |
| --- | --- |
| Thuộc tính | Cấu hình |
| Hệ điều hành | Windows 10 Pro 64-bit ( 10.0, Build 19045 ) |
| CPU & RAM | 11th Gen Intel® Core™ i7-1165G7 (8 CPUs), ~1.7GHz với RAM 16381MB |

*Bảng 6‑1. Bảng thông số máy chủ SocketIO*

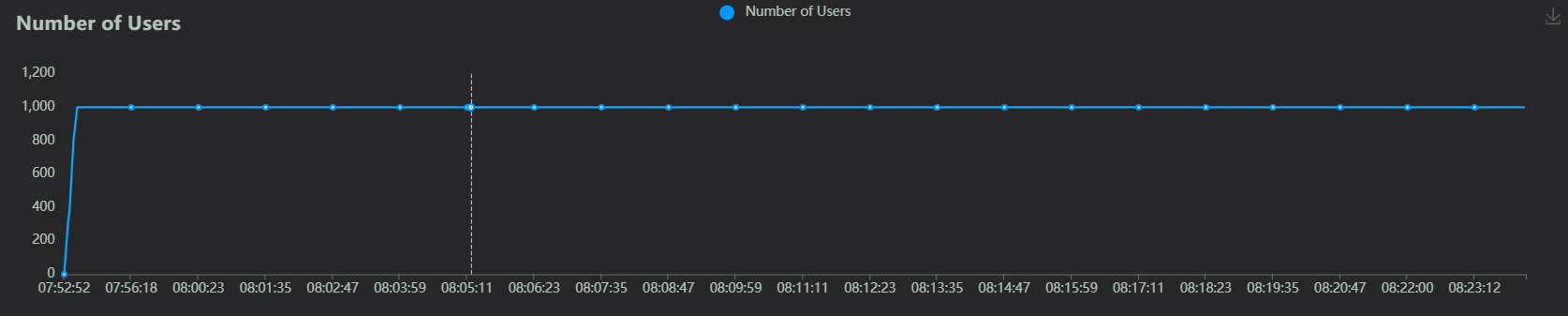
Chi tiết về kết quả đo đạc được được trình bày qua các biểu đồ ở hình 6.8 bên dưới.



*Hình 6.2: Tổng số yêu cầu trên 1 giây*



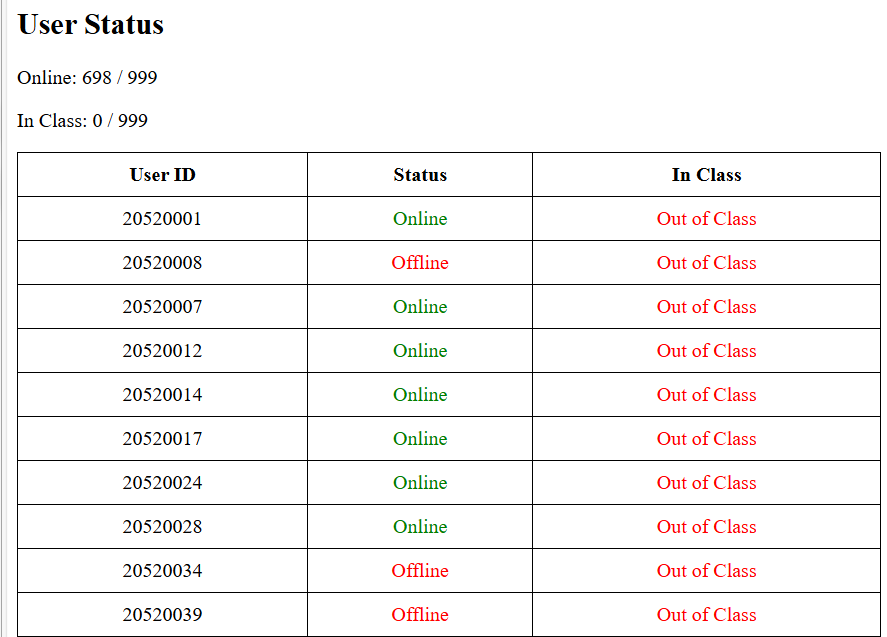
*Hình 6.3: Thời gian phản hồi trung bình cho 1 request*



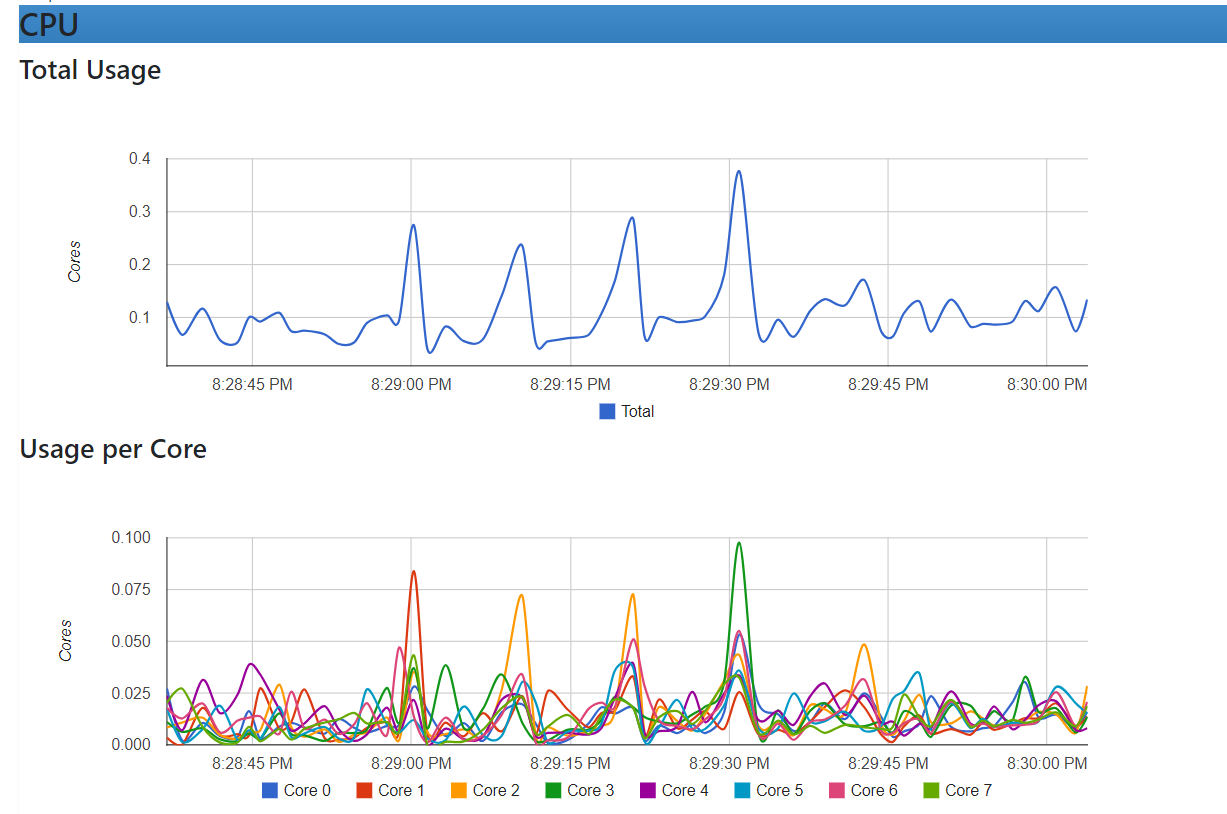
*Hình 6.4: Tổng số người dùng truy cập*

*Bảng 6‑2: Bảng đánh giá chung cho việc kiểm tra tải.*

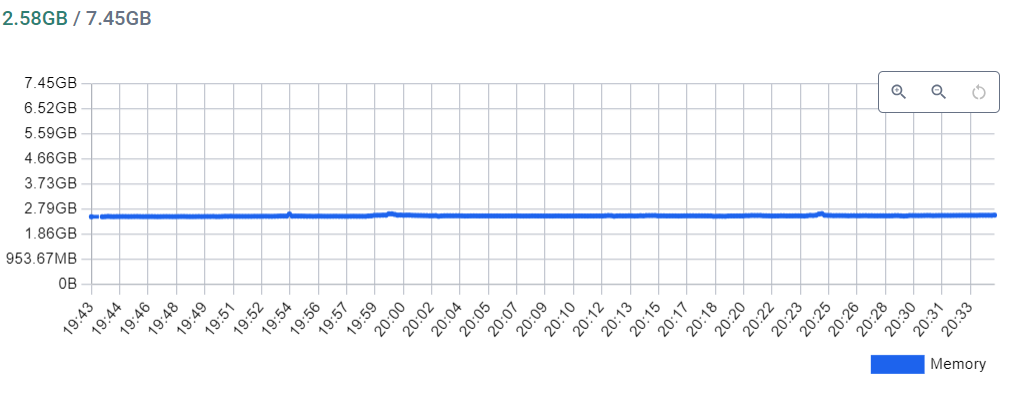
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Type | Name | Requests | Fails | Average (ms) | Min (ms) | Max  (ms) | Average size (bytes) |
| Websocket | Socket | 1395565 | 0 | 413.09 | 7 | 5504 | 29.33 |



*Hình 6.5: Hiển thị người dùng trong trạng thái đã đăng nhập và vào lớp.*



*Hình 6.5: Lượng tiêu thụ CPU khi server socket chịu tải nặng.*



*Hình 6.6:* *Lượng tiêu thụ bộ nhớ server socket chịu tải nặng.*

Các số liệu ở các biểu đồ trên đây là kết quả của kịch bản kiểm tra tải nặng cho các yêu cầu giao tiếp thời gian thực với số lượng user tối đa là 1000, ram up (số lượng người dùng tăng lên trên một giây) là 100 và tiến hành kiểm tra liên tục trong vòng 30 phút.

Dựa vào số liệu từ các biểu đồ và bảng thống kê ở trên, ta có thể nhận thấy rằng máy chủ SocketIO mặc dù có hạn chế về cấu hình vật lý nhưng vẫn có khả năng đáp ứng với điều kiện tải nặng, số lượng request lớn cùng với số lượng người dùng lớn. Kết quả cho thấy máy chủ có thể đáp ứng cho đến 1000 người dùng với thời gian phản hồi trung bình là 413,09ms cho một request.

## Triển khai kiểm thử tính năng nhận diện khuôn mặt

Trước khi triển khai thực nghiệm thì nhóm đã tiến hành kiểm thử qua mô hình nhận dạng khuôn mặt qua các bộ dữ liệu mặt người công khai và cho ra kết quả đều đạt tỷ lệ chính xác cao (hơn 90%) và tiệm cận với các nghiên cứu liên quan. Đặc biệt là có kiểm thử trên bộ dữ liệu mặt người Việt Nam (VN-Celeb). Chi tiết về các kết quả này được trình bày ở bảng dưới đây.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | LFW | AgeDB-30 | CFP\_FP | VN-Celeb |
| Độ chính xác | 98.91% | 89.50% | 92.35% | 90.26% |

*Bảng 6‑3. Bảng độ chính xác kiểm thử trên tập dữ liệu của   
mô hình nhận diện khuôn mặt*

Để thực hiện kiểm tra hiệu suất của mô hình nhận diện khuôn mặt trên thiết bị phần cứng (Jetson NanoTM Development Kit 4GB) thì nhóm sẽ thực nghiệm ở trên năm người thực nghiệm khác nhau (Huyền, Linh, Nhung, Đức Anh và Dương) và trong ba điều kiện sáng khác nhau (Thiếu sáng, Bình thường và Sáng nhiều). Các kịch bản kiểm thử này giúp đánh giá tính hiệu quả và độ tin cậy của mô hình trong các tình huống thực tế khác nhau. Chi tiết về các yêu cầu đặt ra và tiêu chí đánh giá như sau:

* Người thực nghiệm: Gồm năm người tham gia trong kiểm thử (Huyền, Linh, Nhung, Đức Anh và Dương). Đối với mỗi người thực nghiệm, nhóm tiến hành thử nghiệm trong các điều kiện sáng và góc khuôn mặt khác nhau.
* Điều kiện sáng: Được chia thành ba loại gồm Thiếu sáng (< 350 lux), Bình thường (350-550 lux) và Sáng nhiều (> 550 lux). Các điều kiện sáng này giúp đánh giá hiệu suất của mô hình nhận diện khuôn mặt trong các tình huống ánh sáng khác nhau.
* Lượng dữ liệu trong CSDL: Dữ liệu được sử dụng trong kiểm thử là 10 ng/100 ảnh cho mỗi người thực nghiệm. Số lượng dữ liệu này cung cấp một mẫu đủ lớn để đảm bảo tính đại diện và đáng tin cậy trong việc đánh giá hiệu suất của mô hình.
* Góc khuôn mặt (°): Mô hình được kiểm tra trong các góc khuôn mặt khác nhau, bao gồm thẳng 0°, nghiêng 15° và nghiêng 30°. Điều này giúp đánh giá khả năng nhận diện và định danh khuôn mặt của mô hình trong các góc đa dạng.
* Kết quả phát hiện khuôn mặt: Kết quả được thể hiện qua số lần phát hiện đúng, số lần phát hiện sai và tỷ lệ chính xác (%) của mô hình trong việc phát hiện khuôn mặt trên từng người thực nghiệm và điều kiện sáng.
* Kết quả định danh khuôn mặt: Kết quả định danh được thể hiện qua số lần định danh đúng, số lần định danh sai và tỷ lệ chính xác (%) của mô hình trong việc xác định danh tính khuôn mặt trên từng người thực nghiệm và điều kiện sáng.
* Tốc độ xử lý trung bình (ms): Kết quả này thể hiện tốc độ xử lý trung bình của mô hình nhận diện khuôn mặt trên mỗi người thực nghiệm và điều kiện sáng.

Chi tiết về số liệu thực nghiệm được sẽ được trình bày rõ trong bảng 6.4 ngay ở bên dưới đây:



*Bảng 6‑4. Bảng kết quả thực nghiệm nhận diện khuôn mặt*

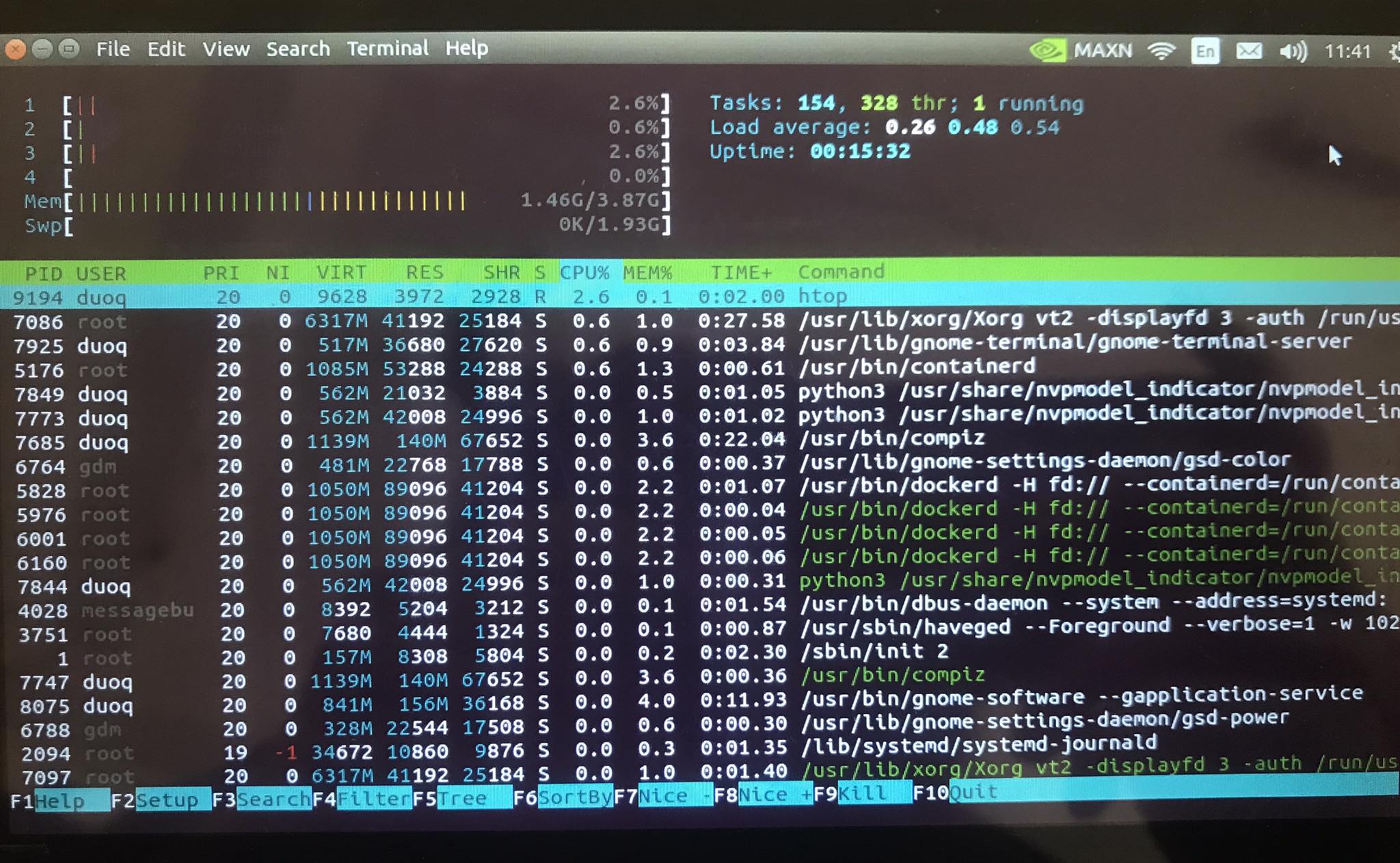
Qua các kịch bản trên chúng ta có thể thấy độ chính xác của tính năng phát hiện khuôn mặt trung bình đạt trên 99% và định danh khuôn mặt tương đối đáng tin cậy ở mức trung bình trên 94% đi kèm cùng với tốc độ xử lý đạt được trên phần cứng của Jetson Nano cũng là rất khả quan với tốc độ trung bình 66.86 ms cho 1 khung hình. Qua việc kiểm thử trên cho thấy mô hình đạt được hiệu suất cao và đáng tin cậy trong việc nhận diện và định danh khuôn mặt trên nền tảng Jetson Nano.

## Thông số hoạt động của ứng dụng

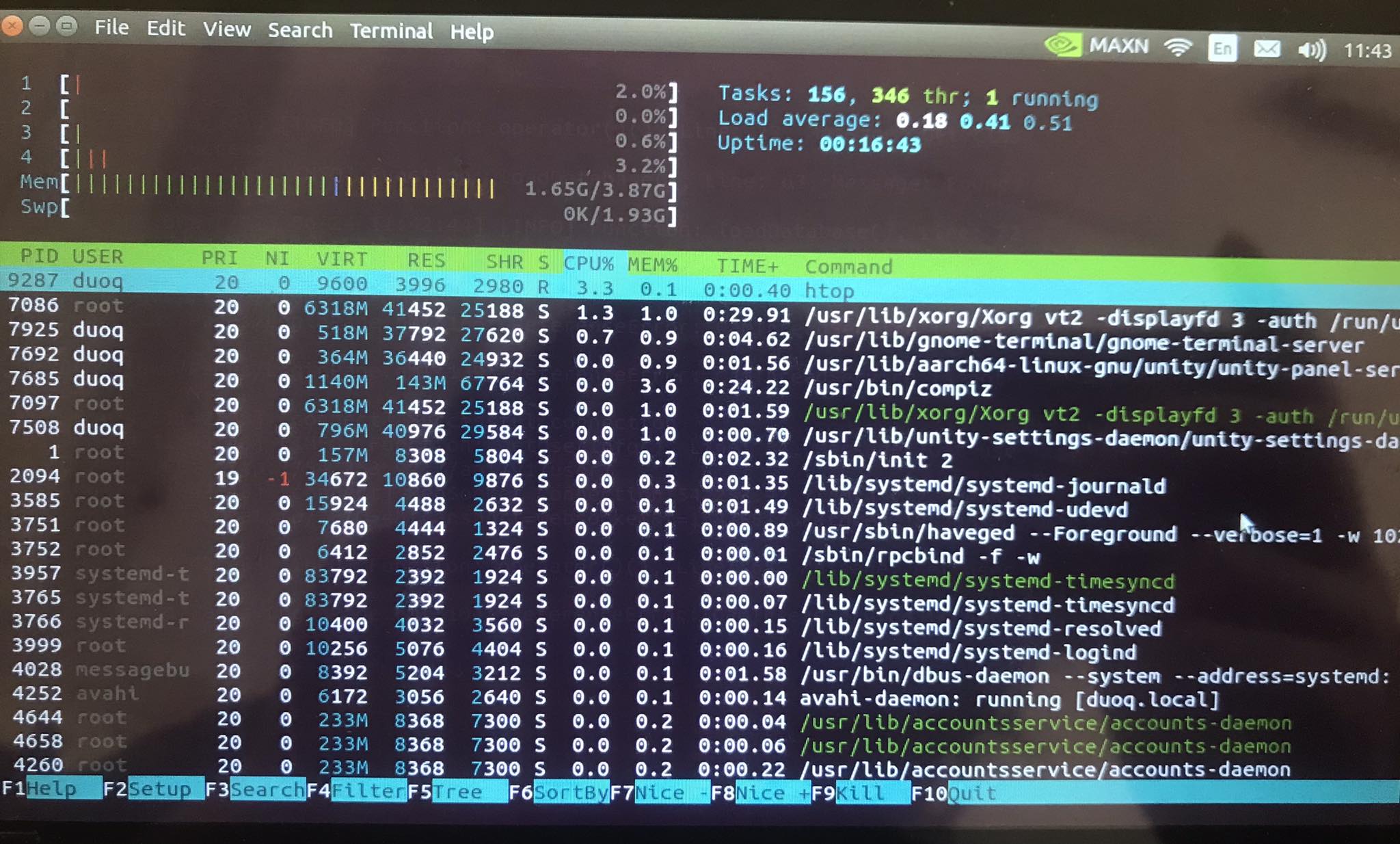
Hiệu suất là một yếu tố cực kỳ quan trọng khi triển khai ứng dụng xuống thiết bị phần cứng. Qua việc chạy thử trên Jetson Nano thì nhóm thấy ứng dụng không chiếm quá nhiều bộ nhớ RAM.

Ban đầu trước khi chạy ứng dụng thì OS của Jetson sẽ chiếm khoảng 1,4GB RAM, và sau khi chạy ứng dụng thì bộ nhớ chiếm khoảng 1,65GB, mức chênh lệch là khoảng 200MB. Một ứng dụng có chức năng về AI cũng như stream video mà chỉ chiếm khoảng 200MB khi triển khai dưới thiết bị phần cứng thì đã được đánh giá là tương đối tốt.

Tuy nhiên, về mặt CPU thì khi ứng dụng không dùng các chức năng AI hay stream video thì phần trăm xử lý của CPU chỉ khoảng 5 – 15%, nhưng khi những chức năng đó được sử dụng thì mức sử dụng CPU luôn nằm trong khoảng hơn 85%. Đây là một ngưỡng sử dụng tài nguyên CPU cao nhưng ta cần chấp nhận đánh đổi để có thể thực thi được các chức năng đó.



*Hình 6.7. Hình thông tin về RAM trước khi chạy ứng dụng*



*Hình 6.8. Hình ảnh bộ nhớ RAM sau khi được chạy ứng dụng*

# ĐÁNH GIÁ KẾT QUẢ VÀ KẾT LUẬN

## Đánh giá kết quả và so sánh với mục tiêu ban đầu

Trong khóa luận này, với sự kế thừa từ đồ án trước, cùng với thời gian thực hiện đề tài, nhóm đã đạt được một số kết quả khả quan. Dưới đây là tổng kết về các kết quả và so sánh với mục tiêu đề ra trong đề cương:

* *Giao diện người dùng thân thiện:* Ứng dụng Qt đã được thiết kế với một giao diện thân thiện với người dùng và dễ sử dụng.
* *Nhận diện khuôn mặt*: Hệ thống nhận diện khuôn mặt đã được tinh chỉnh lại phù hợp với người dùng Việt Nam và đạt kết quả khả quan.
* *Giao tiếp và trao đổi dữ liệu với máy chủ*: Ứng dụng có thể kết nối và truyền nhận dữ liệu một cách liên tục với server.
* *Kết nối thời gian thực tới đa thiết bị*: Hệ thống hỗ trợ kết nối thời gian thực với nhiều thiết bị khác nhau
* *Stream video cho việc mở lớp học trực tuyến:* Ứng dụng để tích hợp được chức năng có thể stream video màn hình và camera để phục vụ cho việc mở lớp học trực tuyến.

Tổng kết lại, kết quả kiểm thử cho thấy ứng dụng Qt và phần Miniserver đã đạt được những mục tiêu đề ra trong đề cương. Các tính năng quan trọng đã được cải thiện và các mục tiêu trong đề cương đã được hoàn thàn. Dưới đây là bảng mô tả chi tiết về các tính năng và số liệu kiểm thử:

|  |  |  |
| --- | --- | --- |
| Mục tiêu | Kết quả | Mục tiêu đề cương |
| Ứng dụng chạy được trên thiết bị nhúng Jetson Nano | Đã có thể chạy được trên phần cứng Jetson Nano | Có thể chạy trên phần cứng Jetson Nano |
| Nhận diện khuôn mặt | - Đạt 98.91 % độ chính xác cao nhất trên tập dữ liệu kiểm tra LFW, 89.50% trên AgeDB-30, 92.35% trên CFP\_FP và 90.26 % trên VN\_Celeb - Tốc độ xử lý trung bình 66.86 ms cho 1 khung hình | - Đạt trên 90.00% độ chính xác trên tập dữ liệu kiểm tra  - Tốc độ xử lý trung bình cho 1 khung hình nhỏ hơn 100ms |
| Tích hợp được chức năng stream video trên ứng dụng Qt | Đã tích hợp được | Tích hợp được |
| Độ trễ khi giao tiếp giữa các thiết bị | Trung bình 413ms với điều kiện tải nặng với 1000 users. | 2000ms |

*Bảng 7‑1. Bảng so sánh kết quả thu được và mục tiêu đề cương*

## Một số điểm hạn chế của đề tài

Mặc dù ứng dụng Qt và cả Miniserver đã đạt được nhiều kết quả khả quan. Tuy nhiên, thời gian thực hiện đề tài là tương đối hạn chế so với khối lượng công việc nhóm đề ra vì vậy vẫn còn tồn đọng một số điểm hạn chế cần được cải thiện như sau:

* *Độ chính xác của nhận diện khuôn mặt*: Mặc dù hệ thống nhận diện khuôn mặt đã được triển khai thành công, tuy nhiên thì đôi khi gặp phải vấn đề trong việc nhận diện người dùng.
* *Độ tương tích của giao diện*: Hiện tại giao diện người dùng đang được thiết kế với kích thước của màn hình 7inch trên Jetson Nano, vẫn còn hạn chế trong việc tương thích với các kích thước màn hình khác.
* *Hỗ trợ nền tảng phần cứng giới hạn*: Hiện tại, ứng dụng chỉ hỗ trợ thiết bị phần cứng và nền tảng cụ thể.
* *Tính ổn định và độ tương thích*: Ứng dụng có thể gặp phải các vấn đề liên quan đến tính ổn định và tương thích với một số hệ điều hành, trình duyệt hoặc phiên bản thiết bị cụ thể.

## Hướng phát triển tiếp theo

Dựa trên những kết quả đã đạt được và những điểm hạn chế, nhóm đề xuất một số hướng phát triển để nâng cao chất lượng và khả năng sử dụng của hệ thống trong tương lai như sau:

* *Cải thiện độ chính xác của nhận diện*: Để đảm bảo độ chính xác cao hơn trong việc nhận diện khuôn mặt, nhóm có thể tiếp tục nghiên cứu và phát triển các thuật toán nhận diện tiên tiến. Tiếp tục chuẩn bị các bộ dữ liệu khuôn mặt để tinh chỉnh mô hình nhằm tăng cường độ chính xác.
* *Mở rộng mô hình*: Với ý tưởng thực hiện một mô hình cho nhiều trường học thì trong tương lai có thể cải tiến hơn Miniserer và kết nối các Miniserver vào chung một Cloud Server để phục vụ cho một hệ thống lớn.
* *Nâng cao khả năng tương tác ngôn ngữ*: Về mặt chính thì ứng dụng mới chỉ tích hợp các tính năng AI về xử lý ảnh, nhưng về tương lai hoàn toàn có thể tích hợp công nghệ xử lý ngôn ngữ tự nhiên (NLP) và xây dựng một hệ thống giao tiếp tự động thông minh sẽ nâng cao trải nghiệm người dùng.
* *Mở rộng hỗ trợ phần cứng* Để đáp ứng nhu cầu sử dụng đa dạng của người dùng, nhóm có thể mở rộng khả năng hỗ trợ thiết bị bằng cách phát triển ứng dụng trên nhiều hệ điều hành và nền tảng khác nhau. Điều này sẽ tạo điều kiện cho người dùng sử dụng có thể dễ tiếp cận hơn.
* *Tăng cường tính ổn định và tương thích*: Để cung cấp trải nghiệm ổn định và tương thích cho người dùng, nhóm có thể tiếp tục nghiên cứu và kiểm tra phần mềm trên nhiều hệ điều hành, trình duyệt và phiên bản thiết bị khác nhau. Việc sửa lỗi, tối ưu và cải thiện tính tương thích sẽ giúp đảm bảo hoạt động mượt mà và hiệu quả của phần mềm.

TÀI LIỆU THAM KHẢO

[1] Manh Hung LE, Thien Minh DOAN, Duy Dieu NGUYEN và Minh Son NGUYEN, “Smart Desk in Hybrid Classroom: Detecting student's lack of concentration when studying” in ACOMPA 2022.

[1] N. Duy Dieu, L. Manh Hung, N. Xuan Huy, N. Hieu Truong và N. Minh Son, “Smart Desk in Hybrid Classroom: Automatic Attendance System based on Face Recognition using MTCNN and ARCFACE,” 2022.

[2] Nvidia, https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[3] M. A. Cicekci và F. Sadik, “Teachers' and Students' Opinions about Students' Attention Problems during the Lesson,” 2019.

[4] M. A. Kamran, M. M. N. Mannan và M. Y. Jeong, “Drowsiness, Fatigue and Poor Sleep’s Causes and Detection: A Comprehensive Study,” 2019.

[5] M. A. Tanveer, M. J. Khan, M. J. Qureshi, N. Naseer và K.-S. Hong, “Enhanced Drowsiness Detection Using Deep Learning: An fNIRS Study,” 2019.

[6] K. Narender and D. N. Barwar, “Analysis of Real Time Driver Fatigue Detection Based on Eye and Yawning,” 2014.

[7] P. Sharma, M. Esengönül, S. R. Khanal, T. T. Khanal and V. F. &. M. J. C. S. Reis, “Student Concentration Evaluation Index in an E-learning Context Using Facial Emotion Analysis,” 2019.

[8] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran and M. Grundmann, “BlazeFace:,” Sub-millisecond Neural Face Detection on Mobile GPUs, 2019.

[9] A. Ablavatski, A. Vakunov, I. Grishchenko, K. Raveendran and M. Zhdanovich, Real-time Pupil Tracking from Monocular Video for Digital Puppetry, 2020.

[10] L. Manh Hung và N. Duy Dieu, “Survey on teachers' assessment of student's status in the classroom”, 2022. https://forms.gle/Se6mRpbGSgB2usJa6.

[11] “Bentley Hybrid Learning” https://www.bentley.edu/atc/hybrid-classroom-project [Truy cập vào 1 tháng 6 năm 2023].

[12] Johnson, Jeff, and Joe Conway. "Exploring the performance benefits of Qt QML." In Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity, pp. 253-262. ACM, 2015.

[13] IEEE Standards Association. "IEEE Standard for Software and System Test Documentation." IEEE Std 829-2008 (Revision of IEEE Std 829-1998), 2008.

[14] ncnn GitHub Repository by Tencent [Truy cập vào 1 tháng 6 năm 2023]. https://github.com/Tencent/ncnn

[15] Express.js GitHub Repository by ExpressJS [Truy cập vào 1 tháng 6 năm 2023]. https://github.com/expressjs/express

[16] React.js GitHub Repository by Facebook [Truy cập vào 1 tháng 6 năm 2023]. https://github.com/facebook/react

[17] Qt QML GitHub Repository. [Truy cập vào 1 tháng 6 năm 2023]. https://github.com/qt/qtdeclarative

[18] GitLab CI/CD GitHub Repository. [Truy cập vào 1 tháng 6 năm 2023]. https://github.com/gitlabhq/gitlab-ci-yaml

[19] Johnson, Jeff, and Joe Conway. "Exploring the performance benefits of Qt QML." In Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity, pp. 253-262. ACM, 2015.

[20] Richardson, Leonard, Sam Ruby, and David Heinemeier Hansson. "Agile web development with rails 6." Pearson Education, 2019.

[21] Sheng Chen, Yang Liu, Xiang Gao, Zhen Han “MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices”

[22] Liu, W., et al., “SphereFace: Deep Hypersphere Embedding for Face Recognition”. arXiv:1704.08063[cs:CV] 26 April 2017.

[23] Wang, H., et al., “CosFace: Large Margin Cosine Loss for Deep Face Recognition”. arXiv:1801.09414 [cs:CV] 29 Jan 2018.

[24] Jiankang Deng, Jia Guo, Niannan Xue, Stefanos Zafeiriou “Arcface: Additive Angular Margin Loss for Deep Face Recognition” in arXiv:1801.07698 [cs.CV] 9 Feb 2019.

[25] Xiang Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices” in arXiv:1707.01083 [cs.CV] 4 July 2017.