**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**PHẠM HUY HOÀNG - 17520518**

**KHÓA LUẬN TỐT NGHIỆP**

**PHÁT TRIỂN ỨNG DỤNG CHĂM SÓC SỨC KHỎE THÔNG MINH**

**DEVELOPMENT OF SMART EHEALTHCARE APPLICATION**

**KỸ SƯ NGÀNH TRUYỀN THÔNG VÀ MẠNG MÁY TÍNH**

**TP. HỒ CHÍ MINH, 2021**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**PHẠM HUY HOÀNG – 17520518**

**KHÓA LUẬN TỐT NGHIỆP**

**PHÁT TRIỂN ỨNG DỤNG CHĂM SÓC SỨC KHỎE THÔNG MINH**

**DEVELOPMENT OF SMART EHEALTHCARE APPLICATION**

**KỸ SƯ NGÀNH TRUYỀN THÔNG VÀ MẠNG MÁY TÍNH**

**GIẢNG VIÊN HƯỚNG DẪN**

**PGS.TS. LÊ TRUNG QUÂN**

**TP. HỒ CHÍ MINH, 2021**

THÔNG TIN HỘI ĐỒNG CHẤM KHÓA LUẬN TỐT NGHIỆP

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

LỜI CẢM ƠN

Đầu tiên, tôi xin gửi lời cảm ơn chân thành đến toàn thể quý Thầy/Cô Giảng viên trường Đại học Công nghệ thông tin - Đại học quốc gia TP. Hồ Chí Minh, đặc biệt là quý Thầy Cô của khoa Mạng máy tính và truyền thông - Thầy Cô đã trực tiếp giảng dạy, cung cấp những kiến thức, kỹ năng quý báu trong những năm qua tại giảng đường đại học.

Đặc biệt, để có thể hoàn thành khóa luận này, tôi xin tỏ lòng biết ơn sâu sắc đến Giảng viên hướng dẫn Thầy PGS.TS Lê Trung Quân. Trong suốt quá trình thực hiện đề tài này, dù gặp không ít trở ngại nhưng Thầy đã luôn quan tâm theo dõi và hỗ trợ kịp thời cho tôi. Xin chân thành cảm ơn Thầy. Ngoài ra, tôi xin phép gửi lời tri ân đến những người đã âm thầm ủng hộ theo dõi tôi. Xin cảm ơn Mẹ và những người thân đã luôn quan tâm, tạo điều kiện tốt nhất để tôi tập trung học tập trong khoảng thời gian ngồi trên giảng đường đại học.

Xin chân thành cảm ơn!

MỤC LỤC

[Chương 1. GIỚI THIỆU 12](#_Toc78535875)

[1.1. Lí do chọn đề tài và mục đích 12](#_Toc78535876)

[1.1.1. Lí do chọn đề tài 12](#_Toc78535877)

[1.1.2. Mục đích 12](#_Toc78535878)

[1.2. Đặt vấn đề 12](#_Toc78535879)

[1.3. Mục tiêu nghiên cứu của đề tài 13](#_Toc78535880)

[1.4. Phạm vi và đối tượng nghiên cứu 13](#_Toc78535881)

[1.4.1. Phạm vi nghiên cứu 13](#_Toc78535882)

[1.4.2. Đối tượng nghiên cứu 13](#_Toc78535883)

[Chương 2. CƠ SỞ LÝ THUYẾT 14](#_Toc78535884)

[2.1. Tổng quan về eHealth, IoT, Edge computing 14](#_Toc78535885)

[2.1.1. eHealth, IoT 14](#_Toc78535886)

[2.1.2. Edge computing 15](#_Toc78535887)

[2.2. Giới thiệu về MySignals 16](#_Toc78535888)

[2.3. Giới thiệu về Arduino, ESP8266 ESP-01, Raspberry Pi 17](#_Toc78535889)

[2.3.1. Arduino, Arduino IDE 17](#_Toc78535890)

[2.3.2. ESP8266 ESP-01 17](#_Toc78535891)

[2.3.3. Raspberry Pi, Raspbian 19](#_Toc78535892)

[2.4. Giới thiệu về Google Cloud Platform 19](#_Toc78535893)

[2.5. Công nghệ ảo hóa Docker, Kubernetes 22](#_Toc78535894)

[2.5.1. Docker, Docker Hub 22](#_Toc78535895)

[2.5.2. Kubernetes, K3S 24](#_Toc78535896)

[2.6. Công nghệ ReactJS, React-Native, NodeJS 27](#_Toc78535897)

[2.6.1. ReactJS 27](#_Toc78535898)

[2.6.2. React-Native 27](#_Toc78535899)

[2.6.3. NodeJS, Express 27](#_Toc78535900)

[Chương 3. HIỆN THỰC, TRIỂN KHAI HỆ THỐNG 30](#_Toc78535901)

[3.1. Tổng quan về mô hình sẽ tiến hành 30](#_Toc78535902)

[3.2. Triển khai cloud 31](#_Toc78535903)

[3.3. Triển khai thiết bị đầu cuối 33](#_Toc78535904)

[3.3.1. Lấy dữ liệu sức khỏe 33](#_Toc78535905)

[3.3.2. Patient App 36](#_Toc78535906)

[3.3.3. Admin web 42](#_Toc78535907)

[3.4. Triển khai tại Edge 45](#_Toc78535908)

[3.4.1. Server controller 45](#_Toc78535909)

[3.4.2. Emotion detect server 48](#_Toc78535910)

[3.4.2.1. Tạo AI emotion detect 48](#_Toc78535911)

[3.4.2.1. Tiến hành chạy và đóng gói 50](#_Toc78535912)

[3.4.3. Triển khai server trên Raspberry 54](#_Toc78535913)

[Chương 4. TỔNG KẾT 59](#_Toc78535914)

DANH MỤC HÌNH

[Hình 2. 1: Các cảm biến và MySignals Hardware 16](#_Toc78535915)

[Hình 2. 2: Module Thu Phát Wifi ESP8266 - 01 18](#_Toc78535916)

[Hình 2. 3: Các dịch vụ của Google Cloud Platform 20](#_Toc78535917)

[Hình 2. 4: Tóm tắt minh họa các phương pháp NAS 21](#_Toc78535918)

[Hình 2. 5: Kiến trúc TFX 22](#_Toc78535919)

[Hình 2. 6: Kiến trúc hoạt động của Docker 24](#_Toc78535920)

[Hình 2. 7: Kiến trúc hoạt động của k8s 25](#_Toc78535921)

[Hình 2. 8: Kiến trúc hoạt động của k3s 26](#_Toc78535922)

[Hình 3. 1: Mô hình hoạt động của hệ thống sẽ tiến hành 30](#_Toc78535923)

[Hình 3. 2: Màn hình project của Firebase 31](#_Toc78535924)

[Hình 3. 3: Dữ liệu được lưu trong Firebase Firestore 32](#_Toc78535925)

[Hình 3. 4: Cảm biến nhịp tim/oxy trong máu 33](#_Toc78535926)

[Hình 3. 5: Kiểm thử kết nối với server 34](#_Toc78535927)

[Hình 3. 6: Kiểm thử kết nối với server 34](#_Toc78535928)

[Hình 3. 7: Source code đọc dữ liệu từ sensor 35](#_Toc78535929)

[Hình 3. 8: Source code dữ liệu sẽ được truyền tới server 35](#_Toc78535930)

[Hình 3. 9: Màn hình điền thông tin bệnh nhân 37](#_Toc78535931)

[Hình 3. 10: Source code tạo socket kết nối tới server từ App bệnh nhân 38](#_Toc78535932)

[Hình 3. 11: Màn hình thông tin của bệnh nhân 39](#_Toc78535933)

[Hình 3. 12: Cảnh báo khi có số liệu bất thường 40](#_Toc78535934)

[Hình 3. 13: Source code xử lý dữ liệu nhân được 41](#_Toc78535935)

[Hình 3. 14: Source code cập nhật ảnh bệnh nhân 41](#_Toc78535936)

[Hình 3. 15: Màn hình đăng nhập admin 42](#_Toc78535937)

[Hình 3. 16: Màn hình đăng ký admin 42](#_Toc78535938)

[Hình 3. 17: Màn hình trang chủ của admin 43](#_Toc78535939)

[Hình 3. 18: Source code đọc dữ liệu từ Firebase 43](#_Toc78535940)

[Hình 3. 19: Màn hình thông tin từng bệnh nhân 44](#_Toc78535941)

[Hình 3. 20: Email được gửi tới chuyên viên y tế 44](#_Toc78535942)

[Hình 3. 21: Source code tạo server và kiểm thử kết nối 45](#_Toc78535943)

[Hình 3. 22: Source code socket nghe và truyền dữ liệu tới client 46](#_Toc78535944)

[Hình 3. 23: Source code route khi nhận dữ liệu 47](#_Toc78535945)

[Hình 3. 24: Source code route xử lý chuẩn đoán hình ảnh gương mặt 47](#_Toc78535946)

[Hình 3. 25: Tài nguyên sử dụng của server controller 48](#_Toc78535947)

[Hình 3. 26: Import file zip chứa ảnh được gán nhãn cảm xúc 48](#_Toc78535948)

[Hình 3. 27: Các image đọc được từ file và phân loại 49](#_Toc78535949)

[Hình 3. 28: Màn hình train model 49](#_Toc78535950)

[Hình 3. 29: Màn hình export model 50](#_Toc78535951)

[Hình 3. 30: Tải về model trong folder download\_dir 50](#_Toc78535952)

[Hình 3. 31: Cấu trúc file JSON được gửi 51](#_Toc78535953)

[Hình 3. 32: Kiểm thử với Postman 52](#_Toc78535954)

[Hình 3. 33: Source code Dockerfile của server emotion detect 52](#_Toc78535955)

[Hình 3. 34: Chạy lại server emotion detect 53](#_Toc78535956)

[Hình 3. 35: Kiểm lại với Postman 53](#_Toc78535957)

[Hình 3. 36: Tài nguyên sử dụng của emotion detect server 54](#_Toc78535958)

[Hình 3. 37: Dockerfile của controller server 55](#_Toc78535959)

[Hình 3. 38: File YAML của controller server 56](#_Toc78535960)

[Hình 3. 39: Ví dụ về các port khi tạo bằng file YAML 57](#_Toc78535961)

[Hình 3. 40: Lỗi không tương thích OS 57](#_Toc78535962)

[Hình 3. 41: Cài đặt và dùng buildx để thực hiện multi-arch 58](#_Toc78535963)

[Hình 3. 42: Các OS/Arch được tạo thành công trên Docker Hub 58](#_Toc78535964)

DANH MỤC BẢNG

[Bảng 2. 1: So sánh giữa NodeJS và ExpressJS 29](#_Toc76125891)

[Bảng 3. 1: Thông số nhịp tim và oxy trong máu theo độ tuổi theo WFSA[10] 32](#_Toc77424613)

[Bảng 3. 2: Bảng kết quả độ chính xác của model emotion detect 50](#_Toc77424614)

DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **TỪ** | **NỘI DUNG** | **DIỄN GIẢI** |
| **OS** | Operating system | Hệ điều hành |
| **Arch** | Architecture | Kiến trúc |
| **Multi-arch** | Multiple architecture | Đa kiến trúc |
| **TCP** | Transmission Control Protocol | Giao thức điều khiển truyền nhân |
| **UDP** | User Datagram Protocol | Giao thức dữ liệu người dùng |
| **HTTP** | HyperText Transfer Protocol | Giao thức truyền tải siêu văn bản |
| **AI** | Artificial Intelligence | Trí tuệ nhân tạo |
| **Method** |  | Phương thức |
| **Protocol** |  | Giao thức |
| **ML** | Machine learning | Máy học |

TÓM TẮT KHÓA LUẬN

Trong thời buổi dịch bệnh phức tạp như hiện nay, tác động của dịch bệnh đặc biệt là COVID-19 đã và đang có tác động mạnh mẽ đế hệ thống y tế trong và ngoài nước. Bệnh viện và các y bác sĩ phải đối mặt với tình trạng giường bệnh đầy ắp bệnh nhân hay thiếu người chăm sóc bệnh nhân. Thêm vào đó chính là sự bùng nổ của số lượng lớn thiết bị thuộc Internet of Things và sự phát triển của công nghệ 5G. Chính vì vậy, việc triển khai y tế thông minh là thực sự cần thiết để giảm áp lực đối với các y bác sĩ. Ở đề tài luận văn này, em sẽ phát triển tiếp mô hình tính toán cận biên (Edge-computing) với bộ dụng cụ của MySignals, đưa ra cảnh báo đồng thời kiểm tra mức độ hài lòng của bệnh nhân qua khuôn mặt.

# GIỚI THIỆU

## Lí do chọn đề tài và mục đích

### Lí do chọn đề tài

Đề tài được phát triển trong thời buổi dịch bệnh Covid-19 dẩn đến sự quá tải ở các bệnh viện, sự tiếp xúc giữa bác sĩ và bệnh nhân thì trở nên vô cùng phức tạp.

Với nhu cầu sức khỏe đang ngày càng được đặt lên hàng đầu thì việc chọn đề tài này sẽ có thể đáp ứng được nhu cầu hiện tại và phát triển thêm trong tương lai. Cụ thể đó là việc tự động thu thập dữ liệu sức khỏe bệnh nhân và đưa ra cảnh báo mà không cần các y bác sĩ.

### Mục đích

Đáp ứng nhu cầu cấp thiết liên quan đến sức khỏe con người ở thời điểm hiện tại cũng như có thể duy trì và phát triển được tiếp trong tương lai. Việc sử dụng IoT trong việc kiểm tra sức khỏe cũng giảm thiểu rủi ro liên quan đến Covid-19 cho các nhân viên y tế.

## Đặt vấn đề

Với tình hình thế giới hiện nay nói chung và Việt Nam nói riêng, khi mà dịch bệnh bùng phát với tốc độ rất nhanh cũng như thiếu hụt các y bác sĩ, sự quá tải ở bệnh viện thì việc phát triển eHealth là thực sự cần thiết.

Hệ sinh thái IoT eHealth lấy bệnh nhân làm trung tâm này cần có kiến ​​trúc nhiều lớp: thiết bị, điện toán sương mù và đám mây để trao quyền xử lý dữ liệu phức tạp về tính đa dạng, tốc độ và độ trễ[1].

Bên cạnh đó eHealth giúp giảm công việc về giấy tờ, giảm chi phí trùng lặp, giảm chi phí sức khỏe trong tương lai khi mà các quyết định lâm sàng mang tính đúng đắn hơn. Cùng với IoT ta có thể truy cập dữ liệu, kết nối với nhiều chương trình/đại lý (bệnh nhân, thiết bị, mạng, nhà cung cấp), hoạt động tự trị[2].

## Mục tiêu nghiên cứu của đề tài

Phát triển, đọc dữ liệu cảm biến dựa trên dụng cụ được cung cấp, lưu trữ dữ liệu thu được vào database cũng như hiển thị kết quả qua app, phát ra cảnh báo khi có nguy hiểm về sức khỏe. Đồng thời kiểm tra mức độ hài lòng của bệnh nhân qua khuôn mặt.

## Phạm vi và đối tượng nghiên cứu

### Phạm vi nghiên cứu

* Phát triển nhận biết sự hài lòng của bệnh nhân dữ liệu cảm biến dựa trên tập dữ liệu được thu thập từ cảm biến lưu vào cơ sở dữ liệu theo mô hình và phát triển thêm ML về nhận biết độ hài lòng qua khuôn mặt.

### Đối tượng nghiên cứu

* Bộ thiết bị MySignals.
* Docker, Kubernetes.
* Lưu trữ, đọc dữ liệu thu được trên cơ sở dữ liệu.
* Thuật toán máy học đánh giá sự hài lòng của bệnh nhân dựa trên khuôn mặt.

# CƠ SỞ LÝ THUYẾT

## Tổng quan về eHealth, IoT, Edge computing

### eHealth, IoT

Thuật ngữ “y tế điện tử” (e-Health) ngày càng được sử dụng nhiều trên thế giới, nhưng chỉ có ít người đưa ra được một định nghĩa rõ ràng cho từ vựng tương đối mới này. Bắt đầu được nhắc đền khoảng năm 1999, thuật ngữ dần dần trở nên khá thông dụng, để chỉ không những việc sử dụng Internet vào lĩnh vực chăm sóc sức khỏe, mà còn muốn đề cập đến 1 phạm vi rộng hơn, gần như là tất cả những gì liên quan đến tin học, máy tính để ứng dụng vào ngành y tế[3].

Các loại hình y tế điện tử có thể kể đến như:

* Dữ liệu y tế điện tử (Electronic health record).
* Điều trị từ xa (telemedicine).
* Thông tin điện tử về sức khỏe (health informatics).
* Cập nhận kiến thức y tế bằng.
* Đội ngũ chăm sóc “ảo” (Virtual healthcare team).
* m-health hay mobile health.
* Nghiên cứu y học thông qua hệ thống trao đổi dữ liệu toàn cầu.
* Hệ thống thông tin y tế (Healthcare Information System)

Internet of Things hay IoT đề cập đến hàng tỷ thiết bị vật lý trên khắp thế giới hiện được kết nối với internet, thu thập và chia sẻ dữ liệu. Nhờ bộ xử lý bên trong cùng mạng không dây, bạn có thể biến mọi thứ trở nên chủ động và thông minh hơn.

IoT được ứng dụng trong các lĩnh vực như thành phố thông minh, giám sát nhà và tự động hóa, chăm sóc sức khỏe, sản xuất, năng lượng và tiện ích, lưới điện thông minh, hệ thống giao thông thông minh và quản lý giao thông [4].

Mặc dù với sự tiếp diễn của dịch bệnh COVID-19, thị trường các thiết bị IoT vẫn phát triển. Thống kê từ IoT analytics cho thấy, năm 2020 có khoảng 11,7 tỷ kết nối thiết bị IoT, con số này cao hơn nhiều so với 9,5 tỷ vào năm 2019. Đến năm 2025, ước tính có khoảng 30 tỷ kết nối IoT, tức là trung bình gần 4 thiết bị IoT trên mỗi người trên toàn cầu[5].

Internet of Things đã nổi lên như một trong những những tiến bộ của công nghệ thông tin và truyền thông, và nó có tác động lớn khi tích hợp với dịch vụ sức khỏe, đặc biệt là eHealth. Việc sử dụng công nghệ IoT trong chăm sóc sức khỏe dịch vụ mang lại sự thoải mái cho bệnh nhân và bác sĩ vì các ứng dụng khác nhau của nó như giám sát thời gian thực, hệ thống quản lý thông tin bệnh nhân và sức khỏe hệ thống quản lý [6].

### Edge computing

Edge Computing – Tính toán cận biên: là một phương pháp tối ưu hóa các hệ thống điện toán đám mây bằng cách xử lý dữ liệu ở rìa của mạng, gần nguồn dữ liệu. Điều này làm giảm băng thông truyền thông cần thiết giữa các bộ cảm biến và trung tâm dữ liệu.

Ngày nay, bệnh nhân đang cần một hệ thống chăm sóc sức khỏe mới hơn và tinh vi hơn, một hệ thống được cá nhân hóa hơn và phù hợp với tốc độ của cuộc sống hiện đại. Để đáp ứng các yêu cầu về độ trễ,hiệu quả và năng lượng cho việc thu thập và phân tích dữ liệu sức khỏe theo thời gian thực, môi trường điện toán biên chính là câu trả lời, kết hợp với tốc độ 5G và các kỹ thuật tính toán hiện đại. Ngay cả với những ưu điểm này, điện toán biên cũng có một số thách thức liên quan, bao gồm các yêu cầu về quyền riêng tư phức tạp và các phương pháp giảm thiểu dữ liệu để cho phép hiệu suất tương đương khi dựa vào Cloud, nhưng với độ phức tạp tính toán thấp hơn. Các hướng nghiên cứu trong tương lai về điện toán biên để chăm sóc sức khỏe đã được xác định để mang lại chất lượng cuộc sống cao hơn cho người dùng nếu được giải quyết[7].

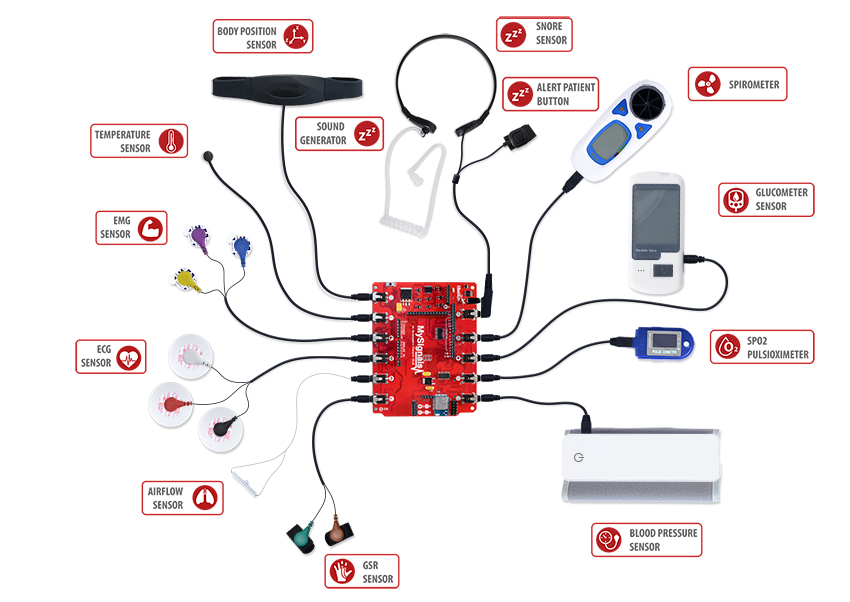
Ngoài Edge computing ta còn có Fog computing (Điện toán sương mù). Trong điện toán sương mù, chỉ có một thiết bị tính toán tập trung chịu trách nhiệm xử lý dữ liệu từ các end device khác nhau trong mạng. Mặt khác, trong tính toán biên, mọi mạng đều tham gia xử lý dữ liệu.

Điểm khác nhau chính của Edge computing và Fog computing đó là edge computing tính toán và đưa trực tiếp vào thiết bị điều khiển còn fog computing ngoài khả năng tính toán, nó còn đưa dữ liệu lên mạng LAN và chuyển đến Edge hay một fog node.

## Giới thiệu về MySignals

MySignals là một nền tảng phát triển cho các thiết bị y tế và các ứng dụng eHealth do Libelium phát triển. Bạn có thể sử dụng MySignals để phát triển các ứng dụng web eHealth của mình hoặc thậm chí thêm các cảm biến của riêng bạn.

MySignals là nền tảng phát triển eHealth hoàn chỉnh nhất trên thị trường, kết nối với hơn 15 cảm biến. Nó có thể coi như một bệnh viện di động, với kích thước của một cuốn sách và rẻ hơn 100 lần so với các giải pháp thị trường hiện tại



Hình 2. 1: Các cảm biến và MySignals Hardware

Các loại cảm biến có thể kể đến như:

* Snore (Cảm biến ngáy)
* Temperature
* Galvanic Skin Response (Hoạt động điện da)
* Electrocardiography (Điện tâm đồ)
* Electromyography (Điện cơ đồ)
* Spirometer (Phế dung kế)
* Bpm/Spo2
* Blood Pressure (Huyết áp)
* Glucometer (Máy đo đường huyết)
* Airflow (Tần số hô hấp)
* Body Position
* Scale
* Alarm Button

## Giới thiệu về Arduino, ESP8266 ESP-01, Raspberry Pi

### Arduino, Arduino IDE

Arduino là một board mạch vi xử lý, dùng để thu thập, xử lí tín hiệu từ các cảm biến và sau đó xuất ra các tín hiệu điều khiển các ngoại vi như đèn, động cơ, xy lanh....

Arduino IDE là một ứng dụng đa nền tảng được viết bằng các chức năng từ C và C ++, được sử dụng để viết và tải các chương trình lên các thiết bị được kết kết với board Arduino.

### ESP8266 ESP-01

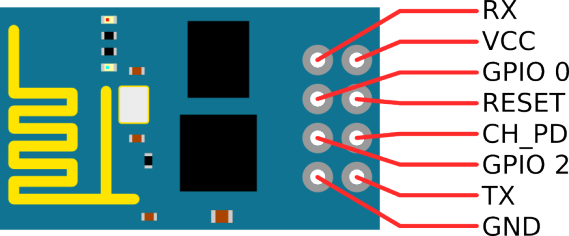
Module Thu Phát Wifi ESP8266 - 01 thường được dùng cho các ứng dụng IOT (Internet of Things). Module đã được nạp sẵn firmware giúp giao tiếp với wifi dễ dàng qua tập lệnh AT (một tập hợp các lệnh được sử dụng để điều khiển một modem) thông qua giao tiếp UART (baudrate mặc định 9600, một số mạch là 11520) quen thuộc.

Các thông số cơ bản:

* Hỗ trợ chuẩn 802.11 b/g/n,
* Wi-Fi 2.4 GHz, hỗ trợ WPA/WPA2,
* Chuẩn điện áp hoạt động: 3.3V.
* Chuẩn giao tiếp nối tiếp UART với tốc độ Baud lên đến 115200.
* Có 3 chế độ hoạt động: Client, Access Point, Both Client and Access Point.
* Hỗ trợ các chuẩn bảo mật như: OPEN, WEP, WPA\_PSK, WPA2\_PSK, WPA\_WPA2\_PSK.
* Hỗ trợ cả 2 giao tiếp TCP và UDP
* Làm việc như các máy chủ có thể kết nối với 5 máy con.

Là một sự lựa chọn khá tốt khi làm dự án IoT với giá thành tương đối thấp.

Sơ đồ chân:



Hình 2. 2: Module Thu Phát Wifi ESP8266 - 01

### Raspberry Pi, Raspbian

Raspberry Pi là một máy tính rất nhỏ gọn, kích thước hai cạnh như bằng khoảng một cái thẻ ATM và chạy hệ điều hành Linux. Raspberry Pi được phát triển bởi Raspberry Pi Foundation - một tổ chức phi lợi nhuận.

Với Raspberry Pi, bạn chỉ cần cài hệ điều hành, gắn chuột, bàn phím và màn hình là có thể sử dụng như một máy vi tính. Raspberry Pi không hoàn toàn có thể thay thế được máy tính để bàn hoặc laptop nhưng nó là một thiết bị đa năng có thể được sử dụng cho những hệ thống điện tử, thiết lập hệ thống tính toán, những dự án DIY… với chi phí rẻ. Vai trò của Raspberry Pi trong khóa luận này sẽ được sử dụng làm Edge device.

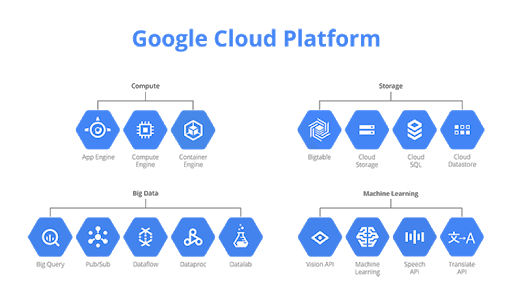
Raspberry Pi có rất nhiều hệ điều hành hỗ trợ, trong đó có Raspbian là hệ điều hành chính thức của Raspberry Pi Foundation. Raspbian là một hệ điều hành máy tính dựa trên Debian cho Raspberry Pi, mà Debian trong thế giới Linux lại vô cùng lớn và rất nhiều tài liệu phong phú. Chính vì vậy cho nên trong khóa luận này sẽ sử dụng Raspbian làm hệ điều hành chính cho các thiết bị Edge.

## Giới thiệu về Google Cloud Platform

Google Cloud Platform viết tắt là (GCP) là nền tảng điện toán đám mây của Google cho phép doanh nghiệp xây dựng phát triển các ứng dụng trên chính hệ thống mà Google đang sử dụng cho sản phẩm của họ như Google Search, G Suite, YouTube, Google Maps…[8]

Một số dịch vụ nổi bật của GCP có thể kể đến như:

* Compute: Compute Engine, App Engine, Google Kubernetes Engine (GKE), Cloud Pub/Sub.
* Storage and Databases: Cloud Storage, Cloud SQL, Cloud Bigtable.
* Big Data: BigQuery, Cloud Dataflow, Dataproc, Cloud Datalab.
* Networking: Google Cloud Virtual Network, Google Cloud Interconnect, Cloud DNS.
* Hybrid and Multi-cloud: Anthos, Cloud Run cho Anthos, Apigee, Istio.



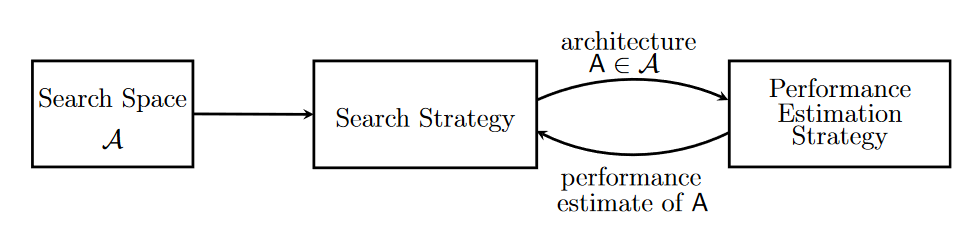
Hình 2. 3: Các dịch vụ của Google Cloud Platform

Bên cạnh những tiện ích về hệ thống quản lý, với hậu thuẫn là công ty tìm kiếm lớn nhất toàn cầu, Google còn đẩy mạnh các bộ công cụ về AI và Machine Learning vào Google Cloud Platform trong đó bao gồm các tính năng nâng cao như:

* Bộ công cụ AutoML với AutoML Natural Language, AutoML Tables, AutoML Translation, AutoML Video, AutoML Vision và Recommendations AI.
* Cloud Vision giúp các nhà phát triển hiểu nội dung của hình ảnh bằng cách đóng gói các mô hình học máy mạnh mẽ trong một API dễ sử dụng.
* Speech-to-Text được phát triển cho các nhà phát triển để chuyển đổi âm thanh thành văn bản bằng cách áp dụng các mô hình mạng nơ-ron mạnh mẽ trong một API dễ sử dụng.
* Cloud Translation – một API RESTful tự động dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác.

Ở khóa luận này, tôi sẽ sử dụng Firebase – một cloud storage của google để lưu trữ dữ liệu đồng thời sử dụng Cloud Vision để thực hiện đánh giá cảm xúc dựa trên khuôn mặt.

Về Cloud Vision, nó sử dụng kỹ thuật NAS (Neural Architecture Search), một trong những kỹ thuật nổi bật của lĩnh vực Thị giác máy tính (Computer Vision). Đây là phương pháp tự động hóa việc thiết kế kiến trúc mạng neural - một mô hình đang được sử dụng rộng rãi trong lĩnh vực máy học.



Hình 2. 4: Tóm tắt minh họa các phương pháp NAS

NAS gồm 3 thành phần chính:

* Search space (không gian tìm kiếm): tìm tất cả mô hình có thể có
* Search strategy (Chiến lược tìm kiếm): phương pháp để tạo ra từng mô hình
* Performance estimate stategy (Chiến lược ước tính hiệu suất được sử dụng): đánh giá hiệu suất từ mô hình mà không cần triển khai nó

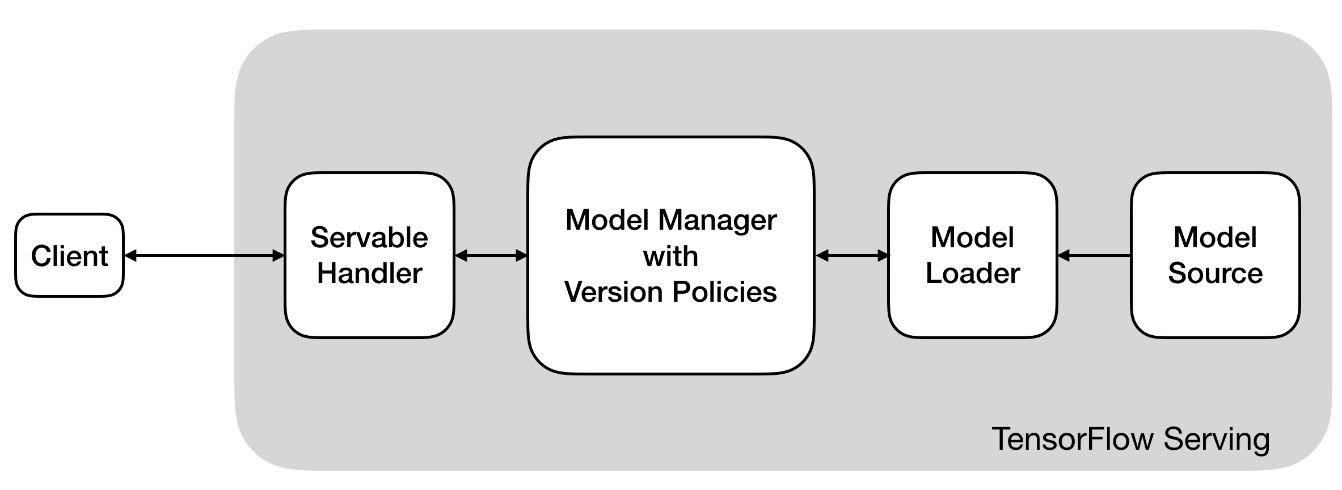
Sau đó, khi đã hoàn thành và đưa ra model, nó sẽ được vào TFX (Tensorflow Serving), bộ công cụ mã nguồn mở, dùng để triển khai (deploy) các mô hình được huấn luyện bởi tensorflow lên môi trường production với 4 phần chính:

Source: Kiểm soát model trên ổ đĩa, kiểm tra thay đổi và tạo đối tượng Loader với version thay đổi

Loader: chứa đầy đủ các thông tin meta-data của 1 model version: tag\_set, signature, model\_name, inputs, outputs

Manager: quản lý và thực hiên các tác vụ tương ứng với 3 thao tác chính: loading, unloading và serving model

Servable: tiếp nhận request từ client và yêu cầu tới Manager để gọi chính xác tới version cụ thể của model



Hình 2. 5: Kiến trúc TFX

Với dataset sử dụng là fer2013 với 28,709 hình ảnh trắng đen của khuôn mặt với kích cỡ 48x48 và các khuôn mặt đã được căn giữa và được phân loại theo cảm xúc (0 = Giận dữ, 1 = Chán ghét, 2 = Sợ hãi, 3 = Hạnh phúc, 4 = Buồn, 5 = Ngạc nhiên, 6 = Bình thường). Bộ dữ liệu này được chuẩn bị bởi Pierre-Luc Carrier và Aaron Courville, như một phần của dự án nghiên cứu đang diễn ra.

## Công nghệ ảo hóa Docker, Kubernetes

### Docker, Docker Hub

Docker là một công cụ giúp cho việc tạo ra và triển khai các container để phát triển, chạy ứng dụng được dễ dàng. Các container là môi trường, mà ở đó lập trình viên đưa vào các thành phần cần thiết để ứng dụng của họ chạy được, bằng cách đóng gói ứng dụng cùng với container như vậy, nó đảm bảo ứng dụng chạy được và giống nhau ở các máy khác nhau (Linux, Windows, Desktop, Server ...).

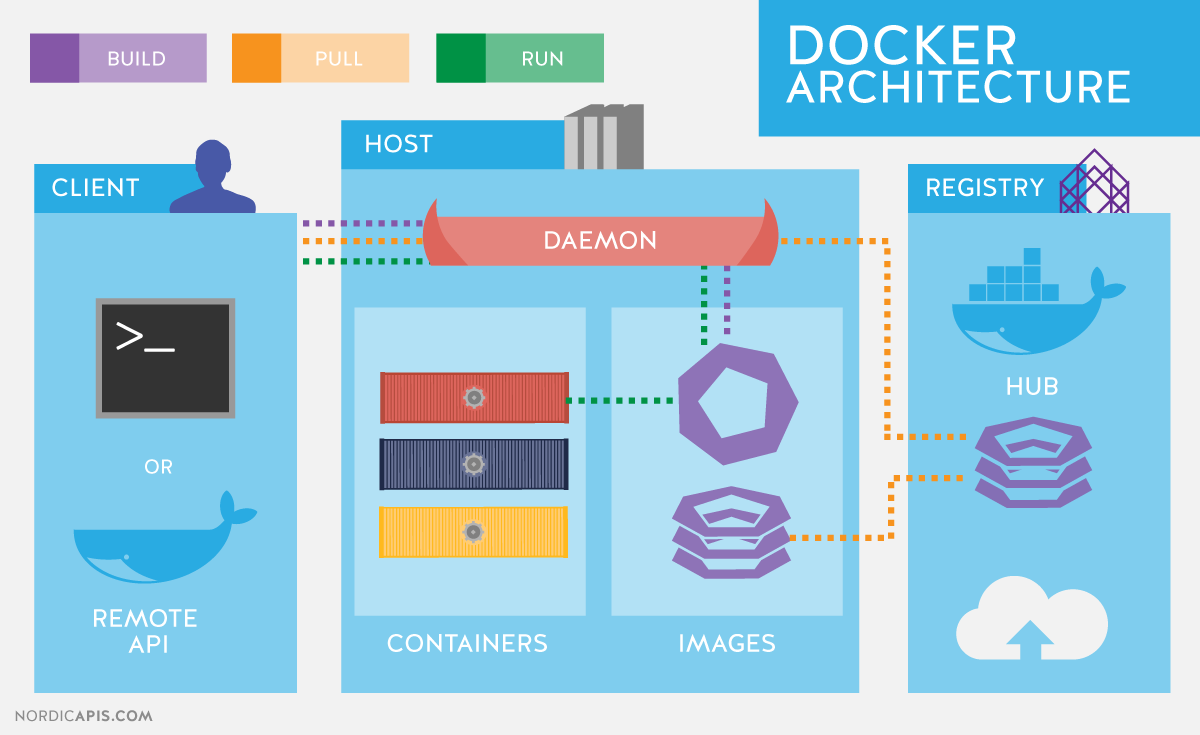
Docker có vẻ rất giống máy ảo (nhiều người từng tạo máy ảo với công cụ ảo hóa như Virtual Box, VMWare), nhưng có điểm khác với VM: thay vì tạo ra toàn bộ hệ thống (dù ảo hóa), Docker lại cho phép ứng dụng sử dụng nhân của hệ điều hành đang chạy Docker để chạy ứng dụng bằng cách bổ sung thêm các thành phần còn thiếu cung cấp bởi container. Cách này làm tăng hiệu xuất và giảm kích thước ứng dụng.

Docker mang lại lợi ích cho cả lập trình viên lẫn quản trị hệ thống, sử dụng Docker lập trình viên tập trung vào mà viết code chứ không lo lắng về việc triển khai, không lo lắng ở máy của lập trình viên chạy được, máy khác lại không chạy được ...

Docker Hub là một dịch vụ do Docker cung cấp, cho phép tìm kiếm và chia sẻ các container images. Các tính năng chính của Docker Hub:

* Repositories: Push và pull container images.
* Teams & Organizations: Quản lý quyền truy cập vào private repositories của container images.
* Official Images: Pull sử dụng container images của Docker.
* Publisher Images: Pull và sử dụng container images được cung cấp bởi vendors khác.
* Builds: Tự động tạo container images từ GitHub và Bitbucket. Push chúng lên Docker Hub.
* Webhooks: Kích hoạt các actions sau khi push thành công một repository lên Docker Hub với các dịch vụ khác.

Docker sử dụng kiến trúc client-server. Docker client sẽ giao tiếp với các Docker daemon, các Docker daemon sẽ thực hiện các tác vụ build, run và distributing các Docker container. Docker client và Docker daemon có thể chạy trên cùng 1 máy, hoặc có thể kết nối theo kiểu Docker client điều khiển các docker daemon như hình dưới. Docker client và daemon giao tiếp với nhau thông qua socket hoặc RESTful API.



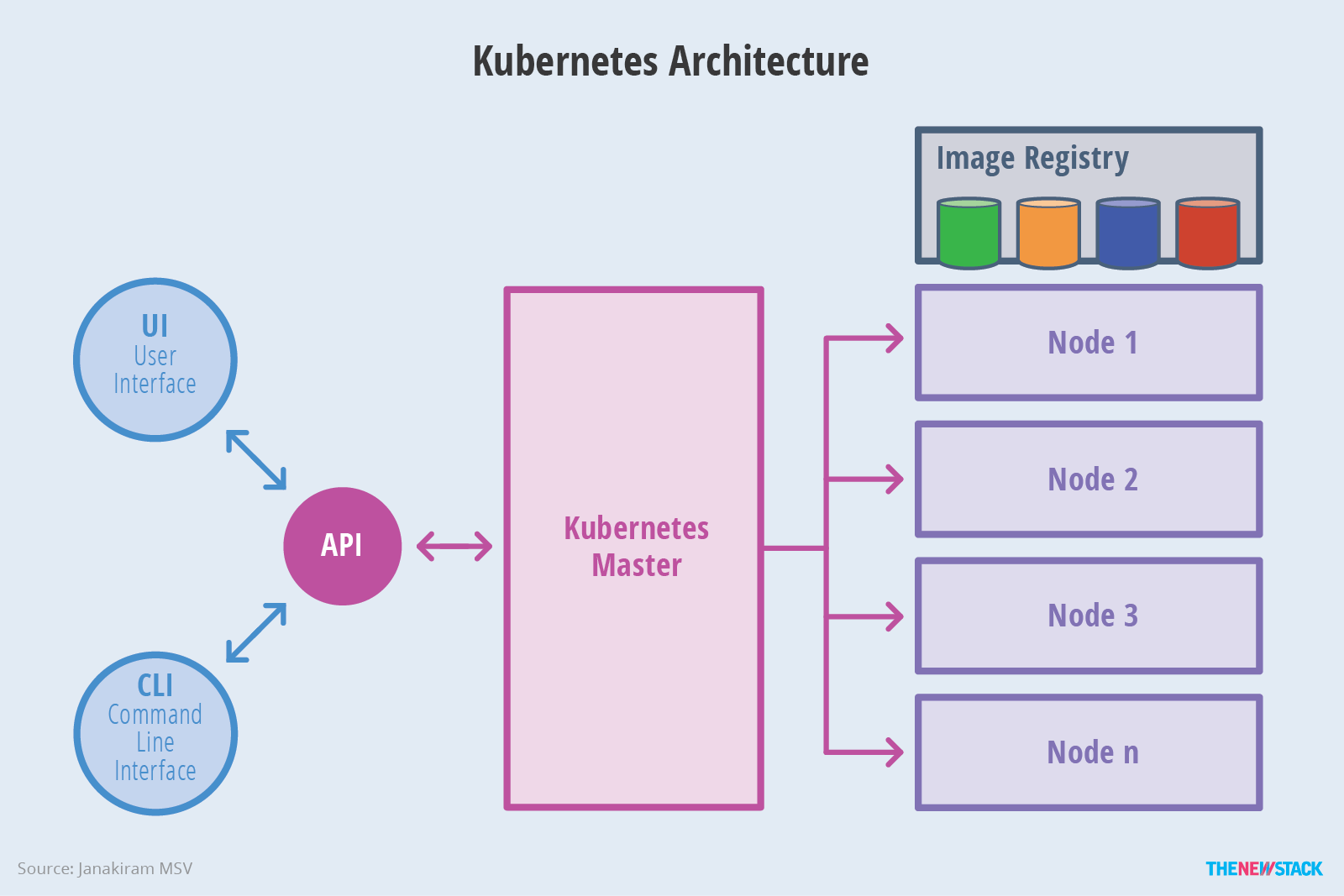
Hình 2. 6: Kiến trúc hoạt động của Docker

### Kubernetes, K3S

Kubernetes, hoặc k8s là một nền tảng mã nguồn mở tự động hoá việc quản lý, scaling và triển khai ứng dụng dưới dạng container hay còn gọi là Container orchestration engine. Nó loại bỏ rất nhiều các quy trình thủ công liên quan đến việc triển khai và mở rộng các containerized applications[9].

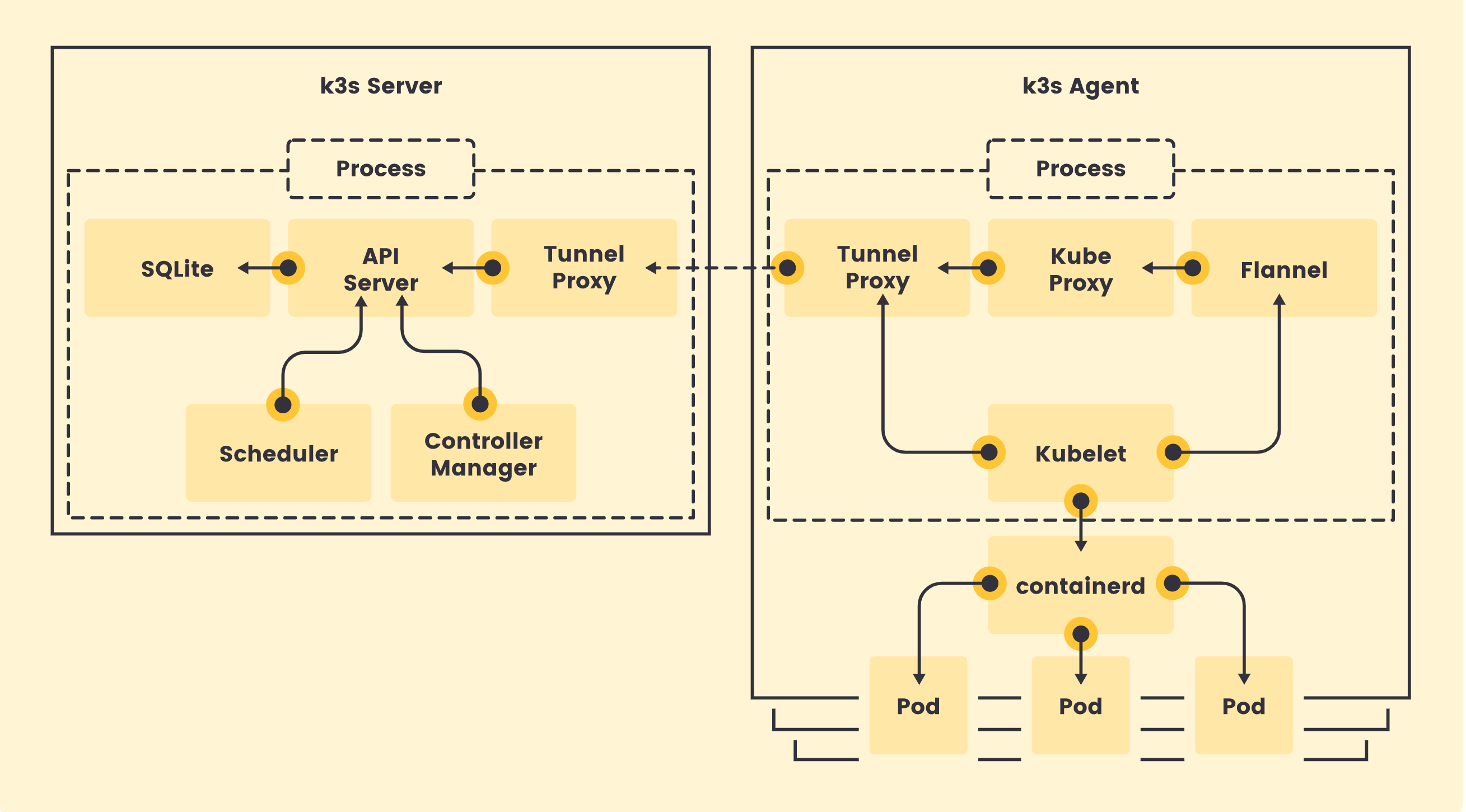
Kubernetes orchestration cho phép bạn xây dựng các dịch vụ ứng dụng mở rộng nhiều containers. Nó lên lịch các containers đó trên một cụm, mở rộng các containers và quản lý tình trạng của các containers theo thời gian.

Các ứng dụng production thực tế mở rộng nhiều containers. Các containers đó phải được triển khai trên nhiều server hosts. Kubernetes cung cấp khả năng phối hợp và quản lý cần thiết để triển khai các containers theo quy mô cho các workloads đó.



Hình 2. 7: Kiến trúc hoạt động của k8s

K3s là một giải pháp triển khai slim K8s. Dễ dàng cài đặt, dung lượng nhỏ và tất cả đều ở dạng nhị phân dưới 100MB.



Hình 2. 8: Kiến trúc hoạt động của k3s

So sánh K3s với K8s:

* Là phiên bản rút gọn của K8s, nhưng K3s cũng vẫn mạnh mẽ khi được đặt vào đúng môi trường. Thật sự, khi suy nghĩ kỹ, K3s là một bản phân phối mã nguồn mở Kubernetes nhỏ hơn, đã được kiểm chứng, tuân thủ đầy đủ quy chuẩn công nghệ và nó là ứng viên hoàn hảo phù hợp với cơ sở hạ tầng IoT, CI một cách tuyệt vời và nó siêu tập trung vào các môi trường hạn chế tài nguyên.
* Nó có kích thước bằng một phần tư bản Kubernetes vanilla, và nó có thể khởi động trong khoảng 30s cùng với các vùng chứa (container) ngay lập tức.
* K3s là 1 bản phân phối K8s tuần thủ đầy đủ quy chuẩn công nghệ, với các cải tiến và thay đổi.

K3S sẽ được triển khai trong khóa luận này trên các edge device (tức các thiết bị Raspberry pi).

## Công nghệ ReactJS, React-Native, NodeJS

### ReactJS

ReactJS là một thư viện Javascript mã nguồn mở hỗ trợ cả front-end lẫn server do Facebook phát triển. Nó còn được dùng để tạo giao diện tương tác cho điện thoại và website.

Những lí do sử dụng ReactJS thay vì những nền tảng phát triển khác bởi sự đơn giản, dễ sử dụng, tái sử dụng code, tính ràng buộc dữ liệu, cải thiện hiệu năng, dễ dàng kiểm thử,…

### React-Native

React Native là một framework được tạo bởi Facebook, cho phép các lập trình viên sử dụng JavaScript để làm mobile apps trên cả Android và iOS với có trải nghiệm và hiệu năng như native. React Native vượt trội ở chỗ chỉ cần viết một lần là có thể build ứng dụng cho cả iOS lẫn Android.

Khi đã làm quen với ReactJS thì việc sử dụng React Native sẽ trở nên hết sức dễ dàng. Thêm vào đó ở khóa luận này sử dụng React Native để thiết kế Patient App bởi đây là dự án nguồn mở với cộng đồng các nhà phát triển lớn, dễ dàng để tìm hiểu và sử dụng, hiệu suất tuyệt vời của các ứng dụng gốc React cùng giao diện đẹp và đơn giản

### NodeJS, Express

NodeJS là một môi trường chạy Javascript bên ngoài trình duyệt web, giúp xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng.

ExpressJS hay Express là một framework back-end cho NodeJS. Nó được thiết kế để xây dựng các ứng dụng web và các API. Express hỗ trợ các method HTTP và midleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.

Ở khóa luận này chọn Express cùng NodejS để xây dựng và phát triển server bởi:

ExpressJS hỗ trợ JavaScript, một ngôn ngữ được sử dụng rộng rãi, rất dễ học và cũng được hỗ trợ ở mọi nơi. Do đó, nếu đã biết JavaScript, thì bạn sẽ thực sự dễ dàng lập trình bằng Express.

Với sự trợ giúp của Express, bạn có thể dễ dàng xây dựng các loại ứng dụng web khác nhau trong một khoảng thời gian ngắn. Express cung cấp định tuyến đơn giản cho các yêu cầu của khách. Nó cũng cung cấp một phần mềm trung gian (middleware) chịu trách nhiệm đưa ra các quyết định để đưa ra các phản hồi chính xác cho các yêu cầu của khách.

Nếu không có Express, bạn phải viết đoạn mã của riêng mình để xây dựng thành phần định tuyến, đây là một công việc tốn thời gian và tẻ nhạt. Express cung cấp sự đơn giản, linh hoạt, hiệu quả, tối giản và khả năng mở rộng cho các lập trình viên. Nó cũng có lợi thế về hiệu suất mạnh mẽ vì nó là một framework của NodeJS.

NodeJS thực hiện tất cả các quá trình thực thi nhanh chóng với sự trợ giúp của Event loop để hoạt động hiệu quả. Hiệu suất mạnh mẽ của NodeJS và dễ dàng viết mã bằng ExpressJS là những tính năng phổ biến nhất được các nhà phát triển ứng dụng web yêu thích. Vì ExpressJS được viết bằng Javascript nên bạn có thể xây dựng các trang web, ứng dụng web hoặc thậm chí là ứng dụng di động bằng cách sử dụng nó.

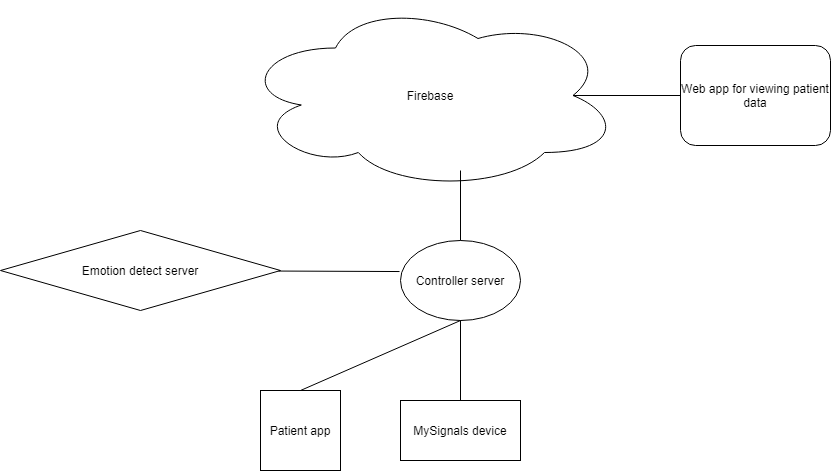
Đây là bảng so sánh giữa NodeJS và ExpressJS

|  |  |
| --- | --- |
| **Node.js** | **Express.js** |
| Được sử dụng để xây dựng cả frontend và backend của một ứng dụng web. | Là một framework của Node.js được sử dụng để xây dựng phần phụ trợ của một ứng dụng web. |
| Được viết bằng các ngôn ngữ lập trình khác nhau như JavaScript, C và C ++,… | Được viết bằng ngôn ngữ lập trình JavaScript. |
| Nodejs không phải là một framework | Là một framework |
| Các lập trình viên cần cài đặt NodeJS trên hệ thống máy tính của họ để sử dụng nó. | Các lập trình viên cần cài đặt Expressjs cùng với NodeJS để sử dụng Expressjs. |
| Được sử dụng để phát triển các ứng dụng mạng và phía máy chủ. | Được sử dụng để xây dựng các ứng dụng phía máy chủ trên NodeJS. |
| Nodejs phù hợp cho các dự án quy mô nhỏ. | Expressjs  phù hợp cho các dự án phụ nhỏ. |
| Nodejs có thể được sử dụng ở cả phía máy khách và phía máy chủ. | Expressjs lại chỉ được sử dụng ở phía máy chủ. |
| Nó tương thích với tất cả các loại hệ điều hành chính. | Expressjs tương thích với tất cả các loại hệ điều hành tương thích với NodeJS. |
| Nó cung cấp nhiều tính năng cho các nhà phát triển để xây dựng một ứng dụng web. | Nó cung cấp các thành phần định tuyến và hỗ trợ phần mềm trung gian để giúp phát triển ứng dụng web dễ dàng hơn. |
| Nodejs hỗ trợ các ngôn ngữ khác như TypeScript, CoffeeScript và Ruby. | Expressjs chỉ  hỗ trợ JavaScript. |
| Nó được sử dụng bởi PayPal, Walmart, LinkedIn, Uber,… | Nó được sử dụng bởi PayPal, IBM, Fox Sports,… |

Bảng 2. 1: So sánh giữa NodeJS và ExpressJS

# HIỆN THỰC, TRIỂN KHAI HỆ THỐNG

## Tổng quan về mô hình sẽ tiến hành



Hình 3. 1: Mô hình hoạt động của hệ thống sẽ tiến hành

Ngữ cảnh: Mỗi phòng sẽ có nhiều giường bệnh được phân biệt qua bedId, mỗi bedId sẽ có một bộ thiết bị ứng với bedId đó. Dữ liệu sẽ truyền tới edge server tương ứng với mỗi phòng.

Tổng quan thành phần:

-Firebase: là nơi ta sẽ lưu trữ dữ liệu dưới dạng cơ sở dữ liệu là NoSql

-Admin web app: Là nơi giúp những chuyên gia y tế xem tình trạng sức khỏe cũng như độ hài lòng các bệnh nhân

-Controller server: nơi trung gian tiếp nhận dữ liệu sức khỏe từ MySignals cũng như hình ảnh để phán đoán mức độ hài lòng từ patient app. Kết nối và lưu trữ dữ liệu nhận được vào database

-Emotion detect server(Edge): Server sẽ thực hiện đọc dữ liệu được gửi để giải quyết từ controller, sau đó thực hiện phán đoán cảm xúc và gửi trả lại controller

-Patient app: Là app điện thoại dành cho bệnh nhân để theo dõi trực tiếp tình trạng sức khỏe của mình

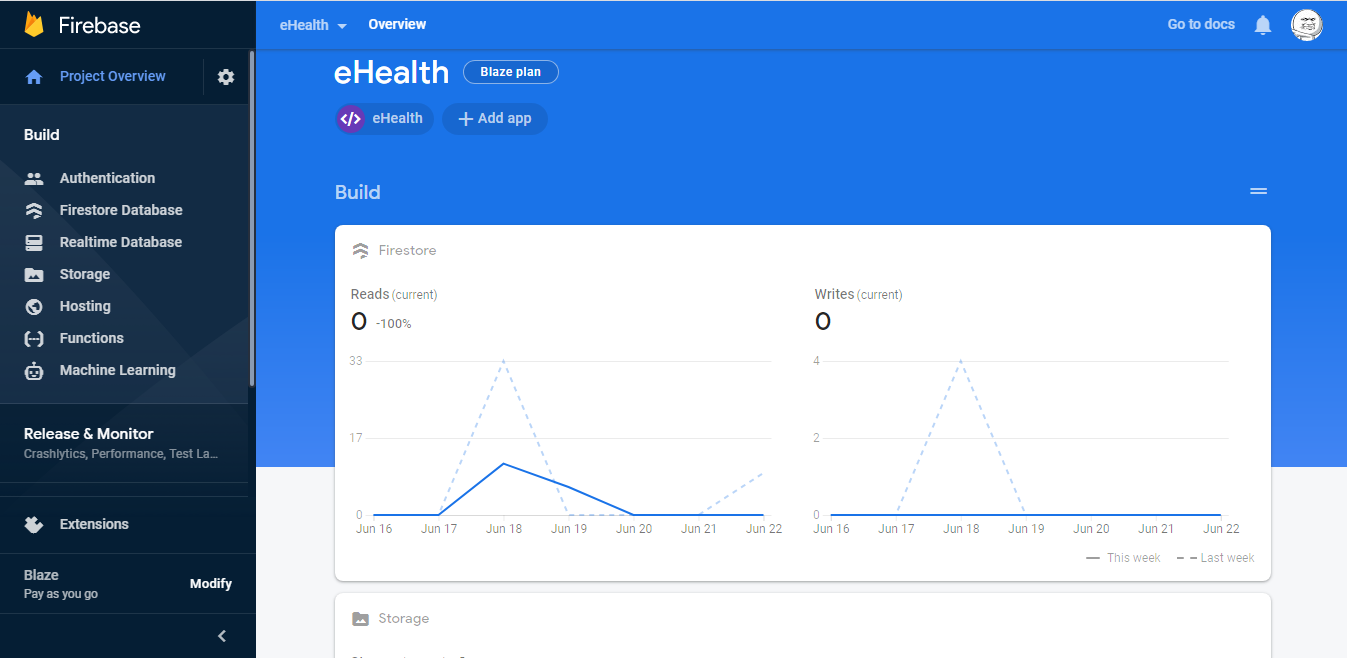
-MySignals: bộ kit của MySignals để đọc dữ liệu sức khỏe.

Như theo mô hình, ở đây ta sẽ phân mô hình làm 3 phần:

* Thiết bị đầu cuối: Patient app, MySignals device, Web app for viewing patient data (hay admin app)
* Edge device: Controller server, Emotion detect server
* Cloud: Firebase

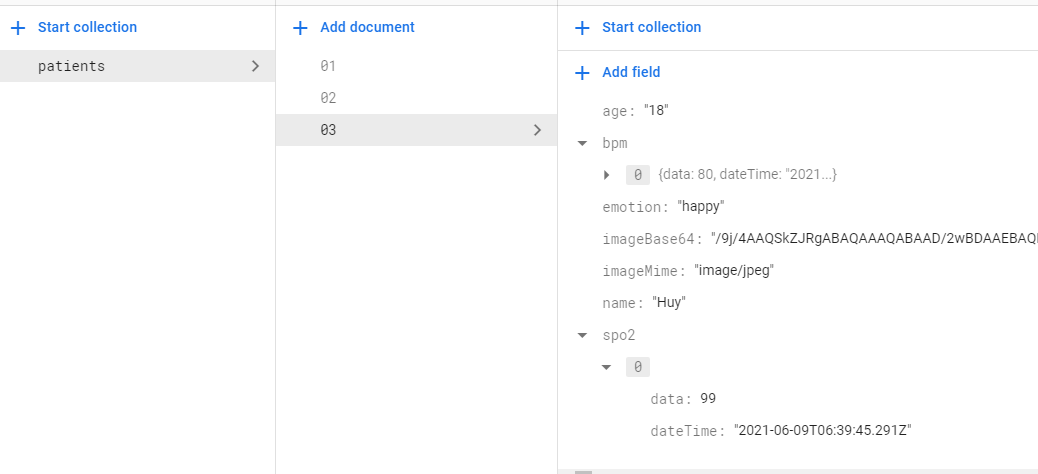
## Triển khai cloud

Ta thực hiện đăng nhập và tạo tài khoản tại Firebase



Hình 3. 2: Màn hình project của Firebase

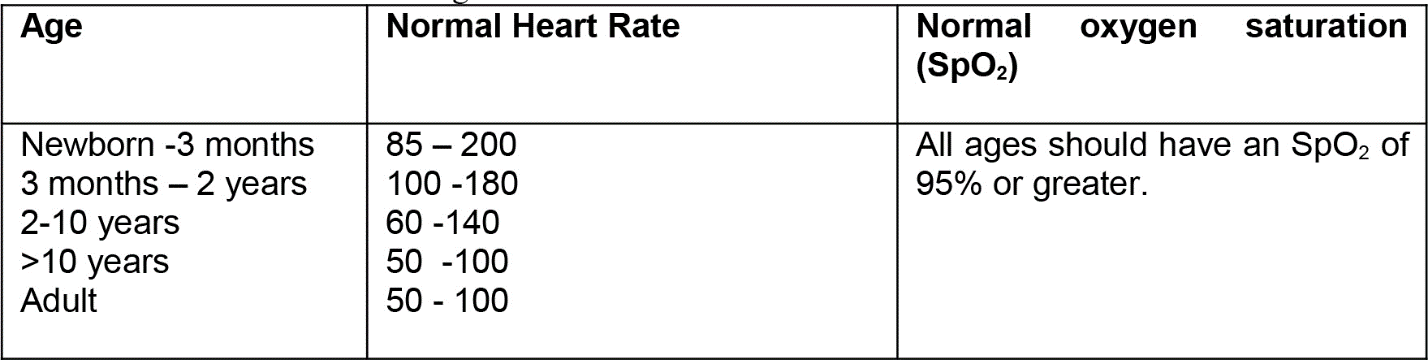
Ta sẽ sử dụng Firestore Database để lưu trữ hình ảnh khuôn mặt dưới dạng base64 cũng như các giá trị khác của bệnh nhân. Dữ liệu sẽ được triển khai như sau:



Hình 3. 3: Dữ liệu được lưu trong Firebase Firestore

Danh sách các bệnh nhân sẽ được lưu dưới dạng id (01,02,…) với các trường:

* bpm và spo2: được lưu trữ dưới dạng array các object gồm data tức dữ liệu về thông số đo được tương ứng và ngày cập nhật giá trị này. Những thông số này sẽ làm cơ sở để các chuyên gia y tế có đưa ra cách giải quyết một cách rõ rang hơn.
* age: thông số vể tuổi của bệnh nhân để có thể dựa vào để đưa ra cảnh báo vể tình trạng bất thường của bệnh nhân theo bảng.



Bảng 3. 1: Thông số nhịp tim và oxy trong máu theo độ tuổi theo WFSA[10]

* emotion: cảm xúc của bệnh nhân sau khi phân tích ảnh.
* imageBase64 và imageMime: dữ liệu và thuộc tính hình ảnh theo base64 để lưu trữ dưới dạng string cũng như có khả năng chuyển đổi lại với imageMime tương ứng tại server.
* name: tên của bệnh nhân.

## Triển khai thiết bị đầu cuối

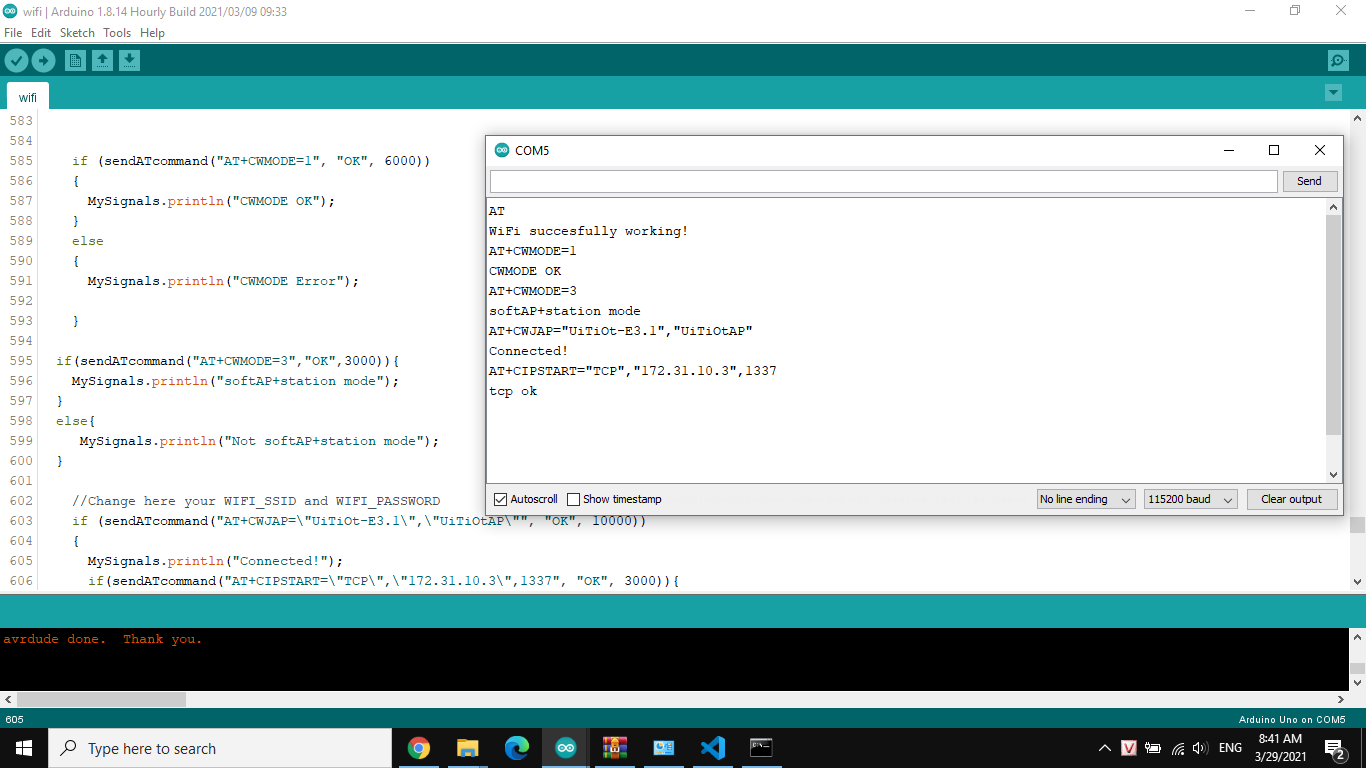
### Lấy dữ liệu sức khỏe

Đầu tiên ta kết nối Arduino board cùng với MySignals hardware cùng ESP-01. Sau đó dùng Arduino IDE thực hiện viết chương trình



Hình 3. 4: Cảm biến nhịp tim/oxy trong máu

Ta thực hiện kiểm tra cũng như kết nối bằng AT command để đảm bảo module hoạt động bình thường để có thể truyền nhận dữ liệu như hình



Hình 3. 5: Kiểm thử kết nối với server



Hình 3. 6: Kiểm thử kết nối với server

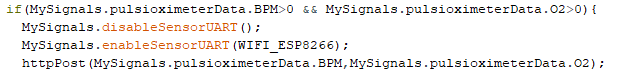
Trong đó một số lệnh quan trọng để có thể giao tiếp với module ESP – 01:

AT: kiểm tra module có đang hoạt động hay không

AT+CIPSTART: Khởi tạo kết nối TCP hoặc UDP

AT+CIPSEND: chuyển tiếp dữ liệu

Ta sử dụng thư viện của MySignals để có thể đọc được giá trị của sensor từ đó chuyển dữ liệu tới server tại edge.



Hình 3. 7: Source code đọc dữ liệu từ sensor

Cuối cùng, sau khi lấy giá trị từ sensor thành công ta sẽ chuyển tiếp dữ liệu này tới server controller với body gói tin có dạng

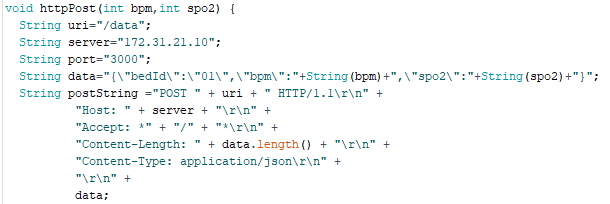
*{*

*"bedId":"01",*

*"spo2":"55",*

*"bpm":"100"*

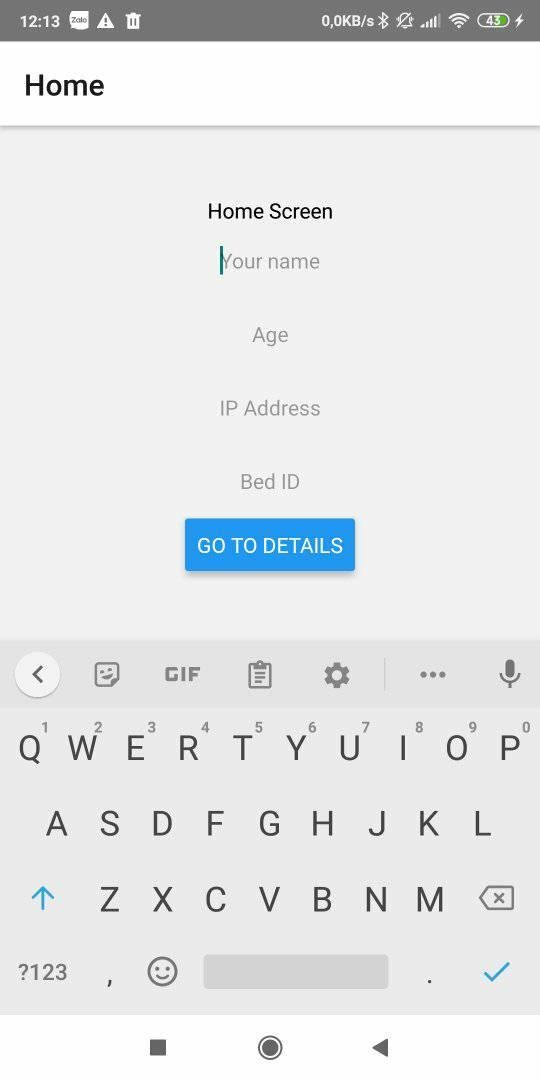
*}*



Hình 3. 8: Source code dữ liệu sẽ được truyền tới server

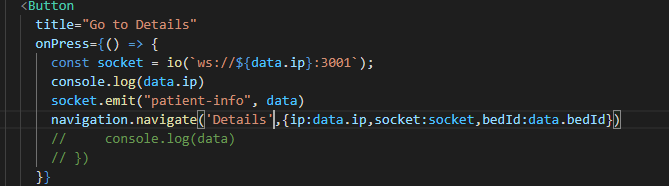
### Patient App

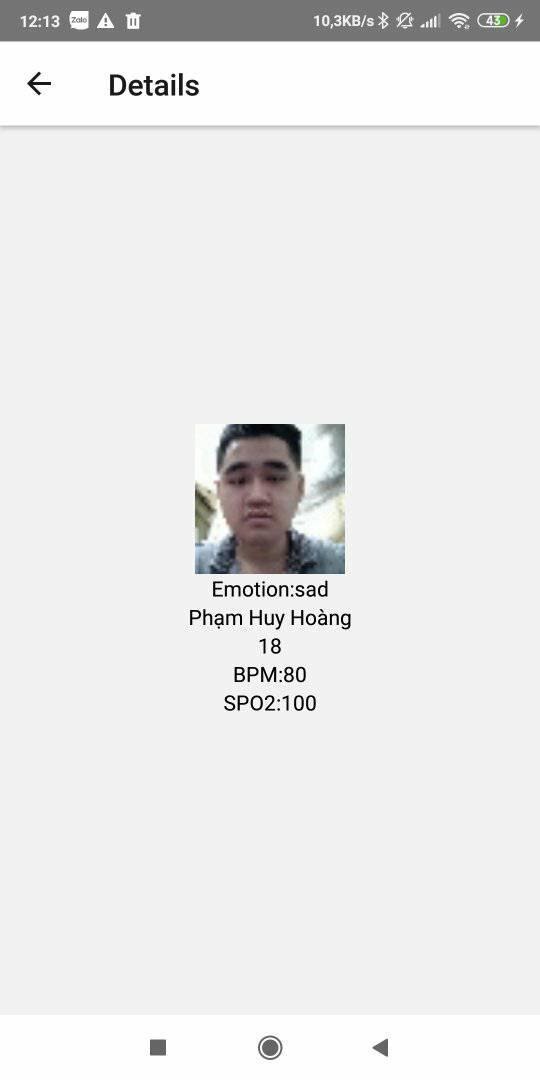
Ta sử dụng framework React Native để tạo ra 2 màn hình chính trên app bệnh nhân gồm màn hình chính để bệnh nhân có thể nhập thông tin cần thiết cũng như thiết lập kết nối tới controller server tại edge.



Hình 3. 9: Màn hình điền thông tin bệnh nhân

Sau khi nhập đầy đủ thông tin và nhấn nút “Go to details” ta sẽ chuyển tới màn hình tiếp theo đó là màn hình thông tin bệnh nhân



Hình 3. 10: Source code tạo socket kết nối tới server từ App bệnh nhân

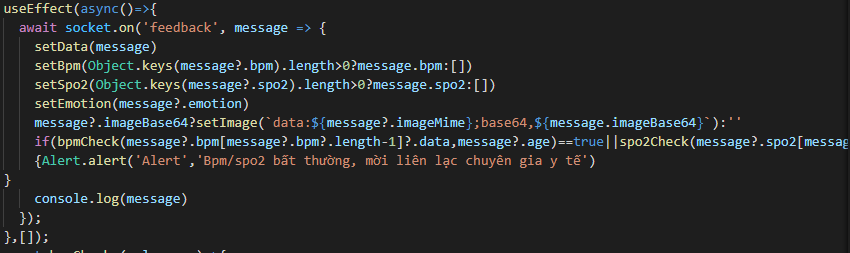
Hình 3. 11: Màn hình thông tin của bệnh nhân

Tại màn hình này ta thực hiện việc đọc dữ liệu được trả về sao khi gửi request thông qua socket đế server controller tại edge với những thông tin như hình ảnh bệnh nhân, cảm xúc, dữ liệu sức khỏe và tuổi.

Đồng thời phát ra cảnh báo ngay khi phát hiện dữ liệu sức khỏe không bình thường

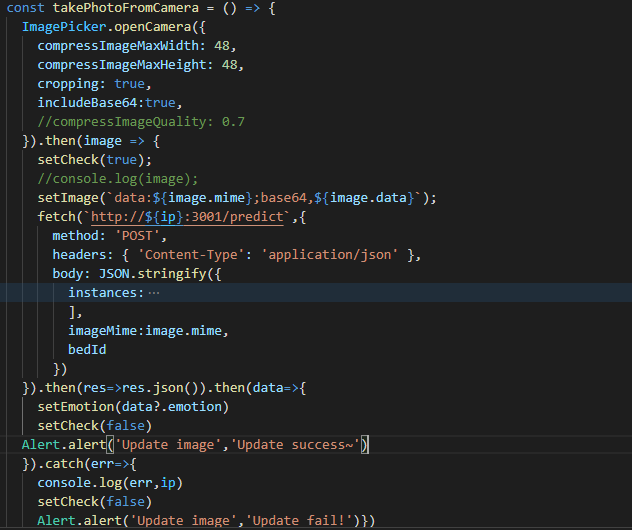


Hình 3. 12: Cảnh báo khi có số liệu bất thường



Hình 3. 13: Source code xử lý dữ liệu nhân được

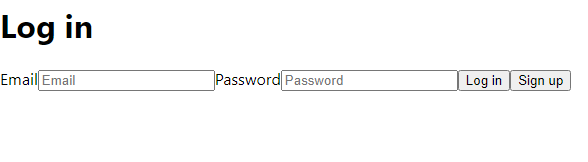
Với hình ảnh bệnh nhân, ta thực hiện viết code để có thể chụp ảnh bệnh nhân rồi chuyển đổi hình ảnh thu được thành dạng base64 và image mime và thực hiện HTTP post đến controller server



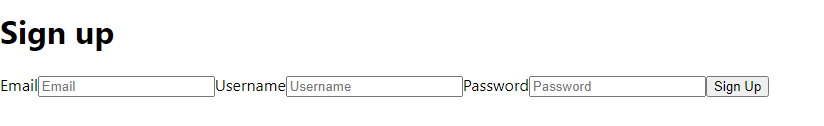
Hình 3. 14: Source code cập nhật ảnh bệnh nhân

### Admin web

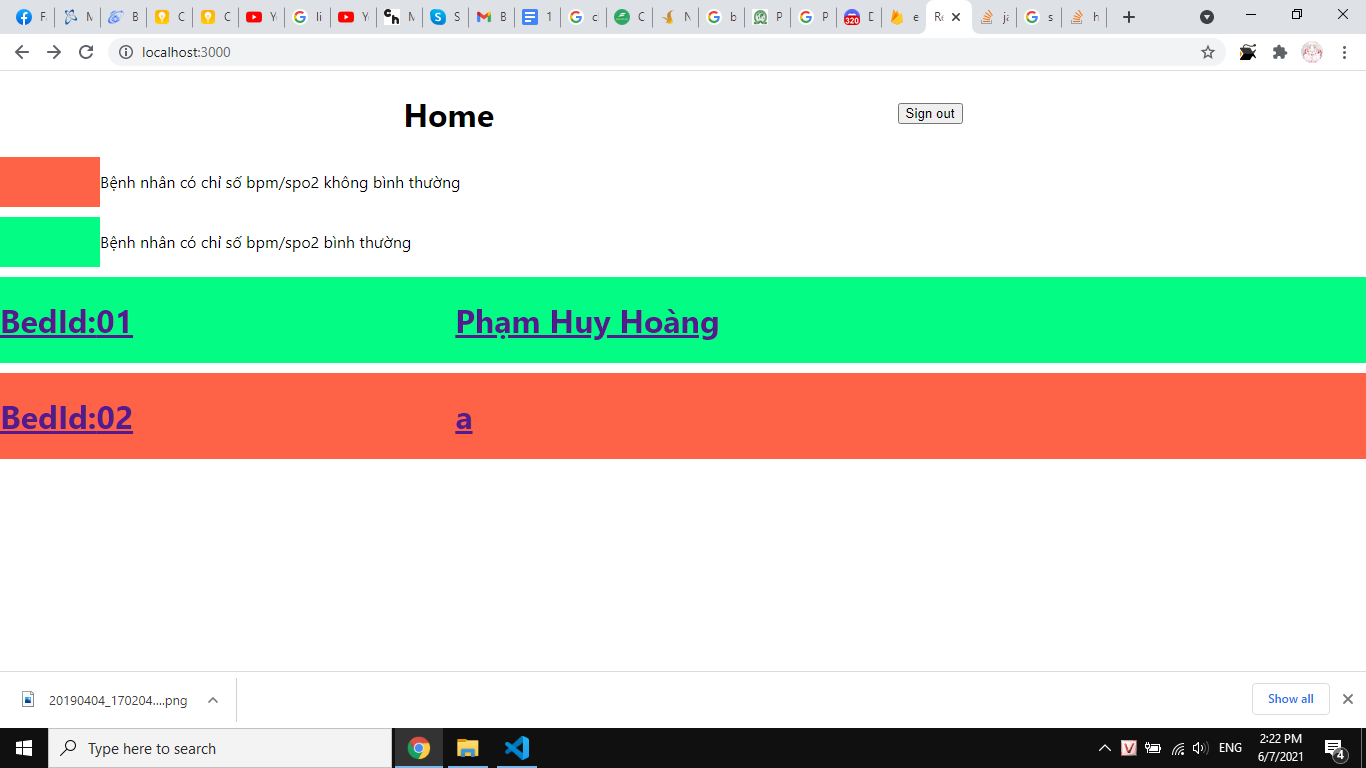
Đầu tiên là màn hình đăng nhập và đăng ký cho người quản lý



Hình 3. 15: Màn hình đăng nhập admin

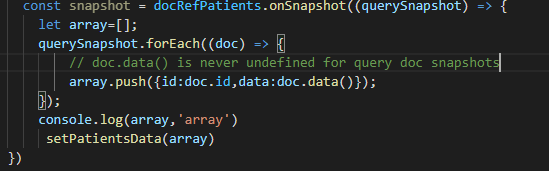


Hình 3. 16: Màn hình đăng ký admin

Màn hình trang chủ theo dõi đầy đủ danh sách các bệnh nhân gồm 2 trường là bedId và tên bệnh nhân, cùng đó là màu trạng thái với màu đỏ chỉ ra những bệnh nhân có dữ liệu sức khỏe không ổn định và màu xanh đại diện cho những bệnh nhân có dữ liệu sức khỏe bình thường. 

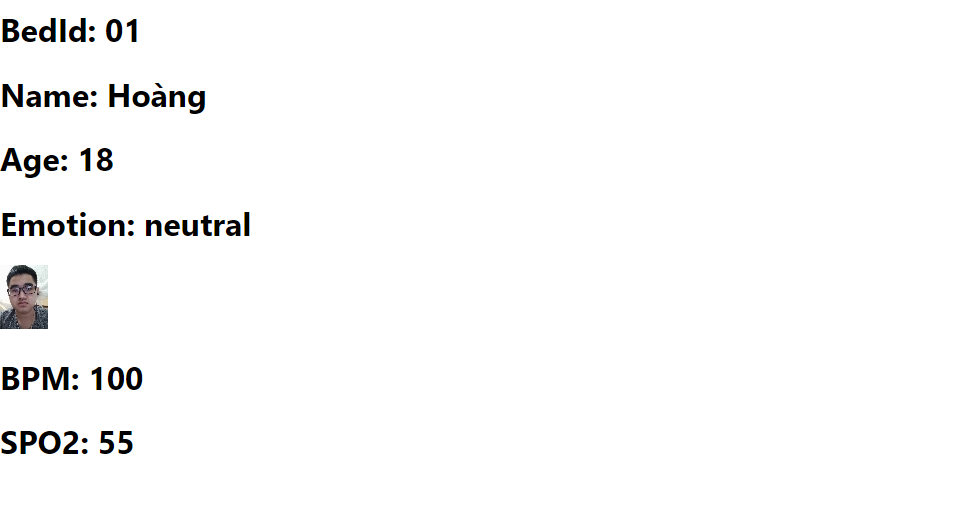
Hình 3. 17: Màn hình trang chủ của admin

Các dữ liệu về bệnh nhân và trạng thái sẽ luôn được cập nhật theo thời gian thật.



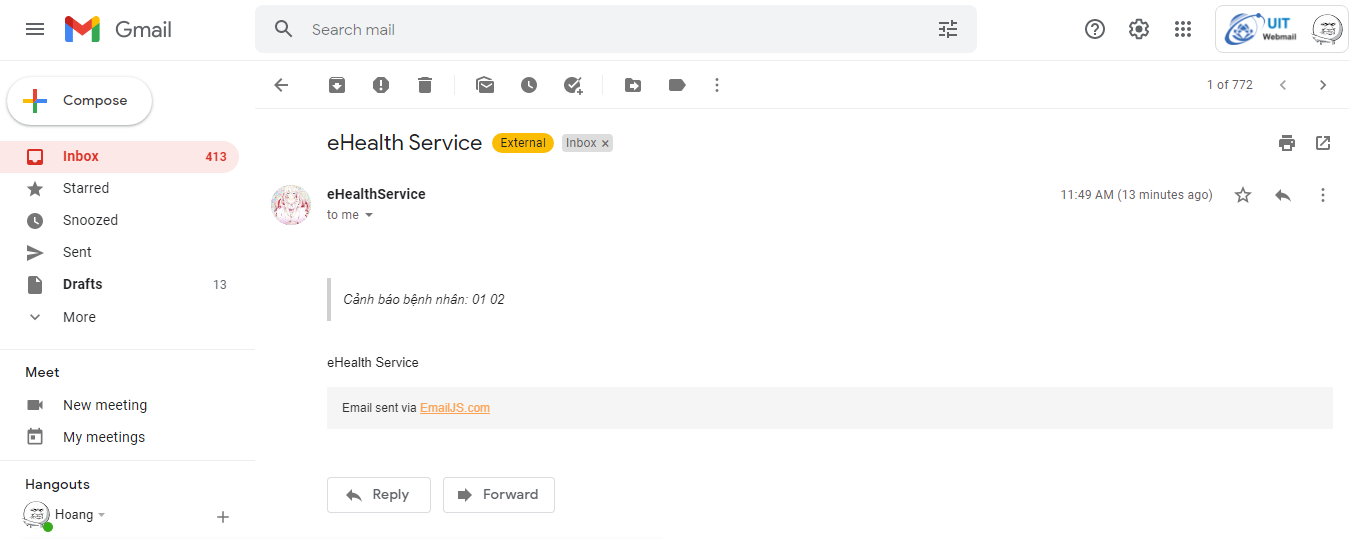
Hình 3. 18: Source code đọc dữ liệu từ Firebase

Ở màn hình từng bệnh nhân, ta sẽ thấy đầy đủ những dữ liệu về bệnh nhân như: tên, tuổi, cảm xúc, các dữ liệu sức khỏe,… cũng luôn được cập nhật theo theo thời gian thật.



Hình 3. 19: Màn hình thông tin từng bệnh nhân

Với những bệnh nhân có trạng thái đỏ, tức dữ liệu sức khỏe không ổn định thì server sẽ tự động gửi mail cho các bác sĩ, chuyên viên y tế.



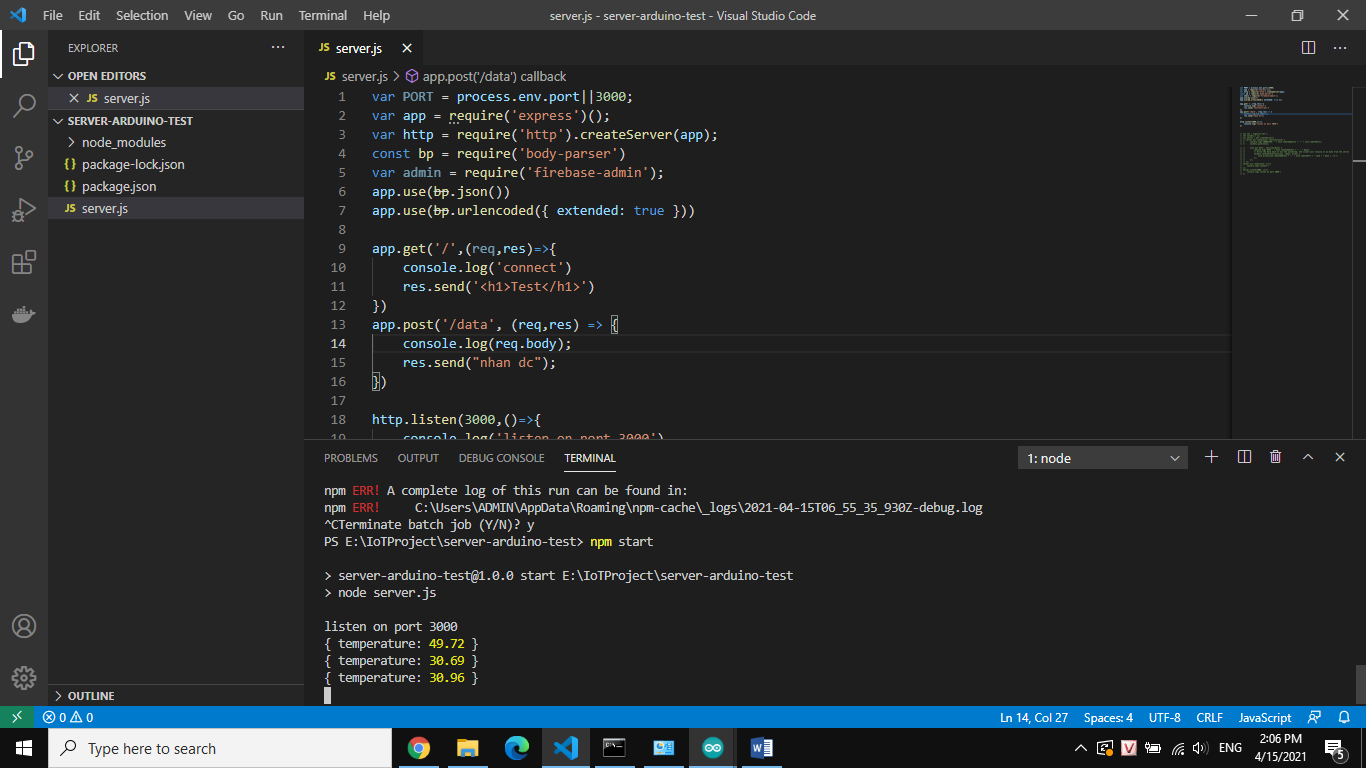
Hình 3. 20: Email được gửi tới chuyên viên y tế

## Triển khai tại Edge

### Server controller

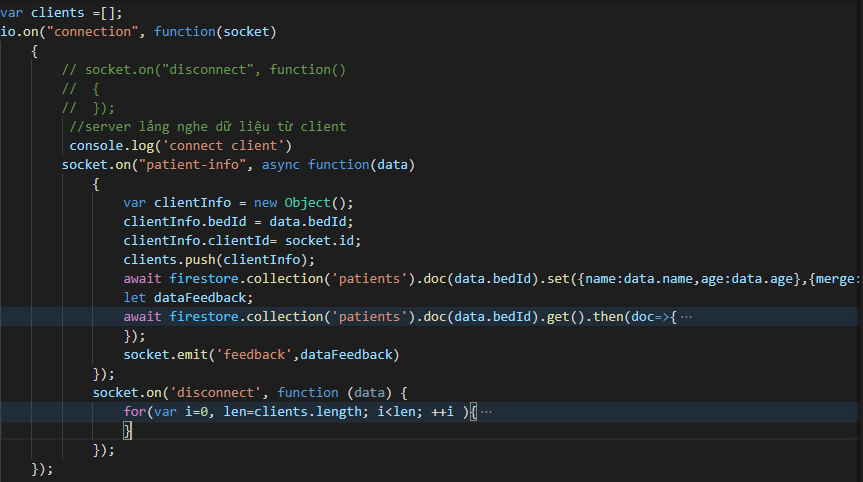
Ta thực hiện server controller với chức năng đó là đọc nhận dữ liệu từ các thiết bị đầu cuối hay cụ thể ở đây chính là đọc dữ liệu khuôn mặt, các thông số cơ bản từ app bệnh nhân gửi tới và dữ liệu sức khỏe từ MySignals. Đồng thời controller còn chuyển tiếp dữ liệu khuôn mặt tới Emotion detect server và đưa các dữ liệu đọc và xử lý tại đây tới Firebase.

Đầu tiên ta sẽ sử dụng NodeJS và Express framework để tiến hành tạo server nhận kết nối từ thiết bị đầu cuối tại port 3001.



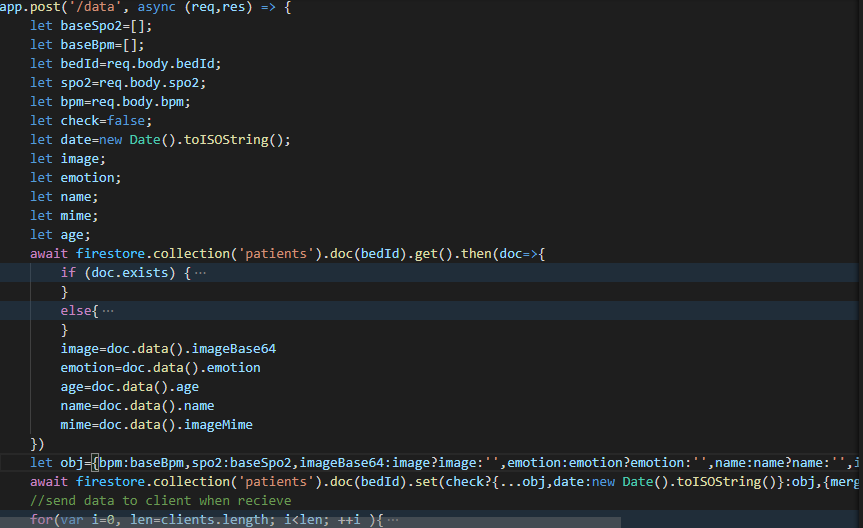
Hình 3. 21: Source code tạo server và kiểm thử kết nối

Tạo socket để lưu client tới server để có thể đọc những yêu cầu tới và trả lại những phản hồi tương ứng.



Hình 3. 22: Source code socket nghe và truyền dữ liệu tới client

Tạo route với HTTP POST request tới server với chức năng lưu giá trị sức khỏe vào bedId tương ứng trên database, đồng thời gửi thông báo thay đổi cho client qua socket tại bedId đó.



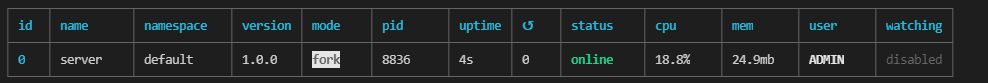
Hình 3. 23: Source code route khi nhận dữ liệu

Cuối cùng, ta tạo route khi có yêu cầu chuẩn đoán cảm xúc qua khuôn mặt, sau đó, lưu giá trị chuyển đoán được cùng hình ảnh thu được dưới dạng base64 vào database và gửi trả kết quả chuẩn đoán được cho client



Hình 3. 24: Source code route xử lý chuẩn đoán hình ảnh gương mặt

Ta sử dụng gói PM2 để kiểm tra server controller



Hình 3. 25: Tài nguyên sử dụng của server controller

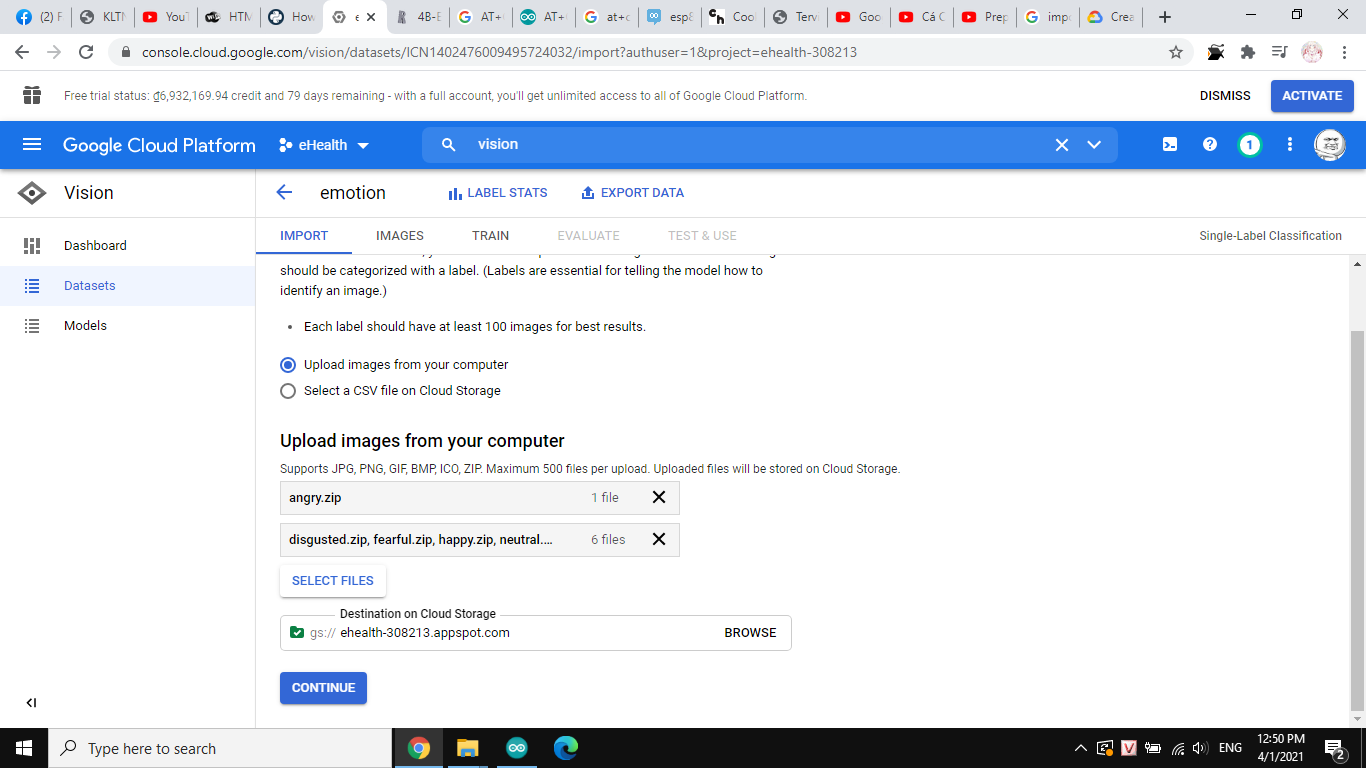
Server controller sử dụng 24.9MB nên hoàn toàn có thể triển khai được trên thiết bị Raspberry Pi.

### Emotion detect server

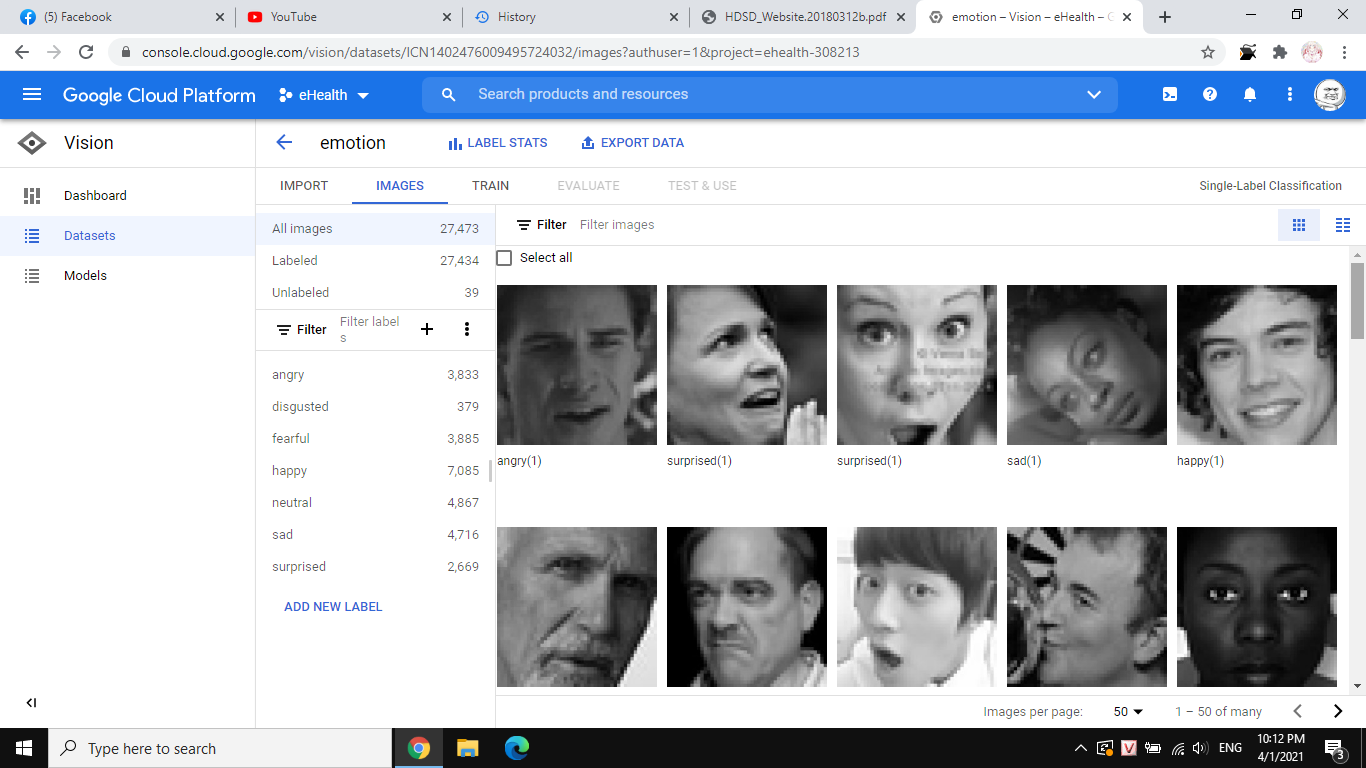
#### Tạo AI emotion detect

Ta sẽ tiến hành training AI dựa vào Tensorflow Serving- dùng để triển khai (deploy) các mô hình được huấn luyện bởi tensorflow lên môi trường production dược Google Vision cung cấp.

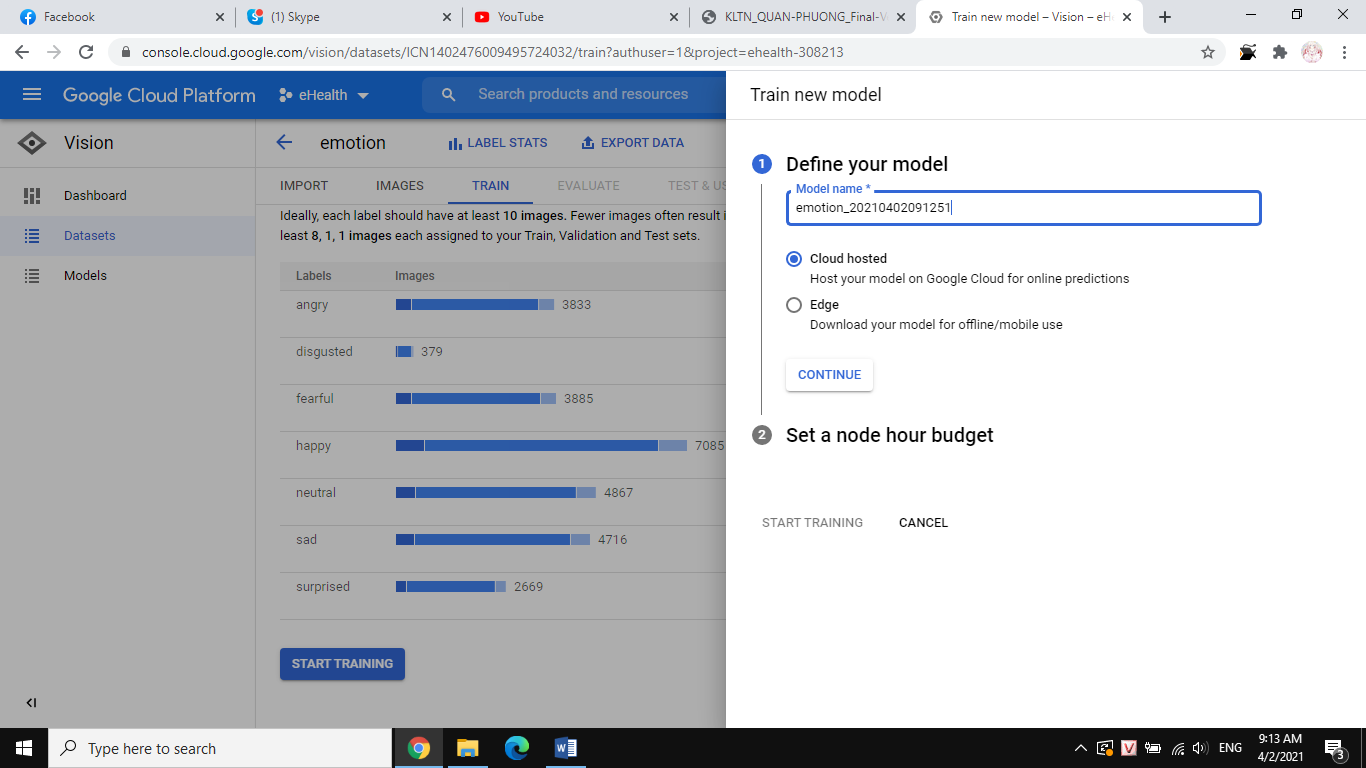
Import các image được dáng nhãn tương ứng với cảm xúc được nén lại vào file zip vào Google Vision.



Hình 3. 26: Import file zip chứa ảnh được gán nhãn cảm xúc



Hình 3. 27: Các image đọc được từ file và phân loại

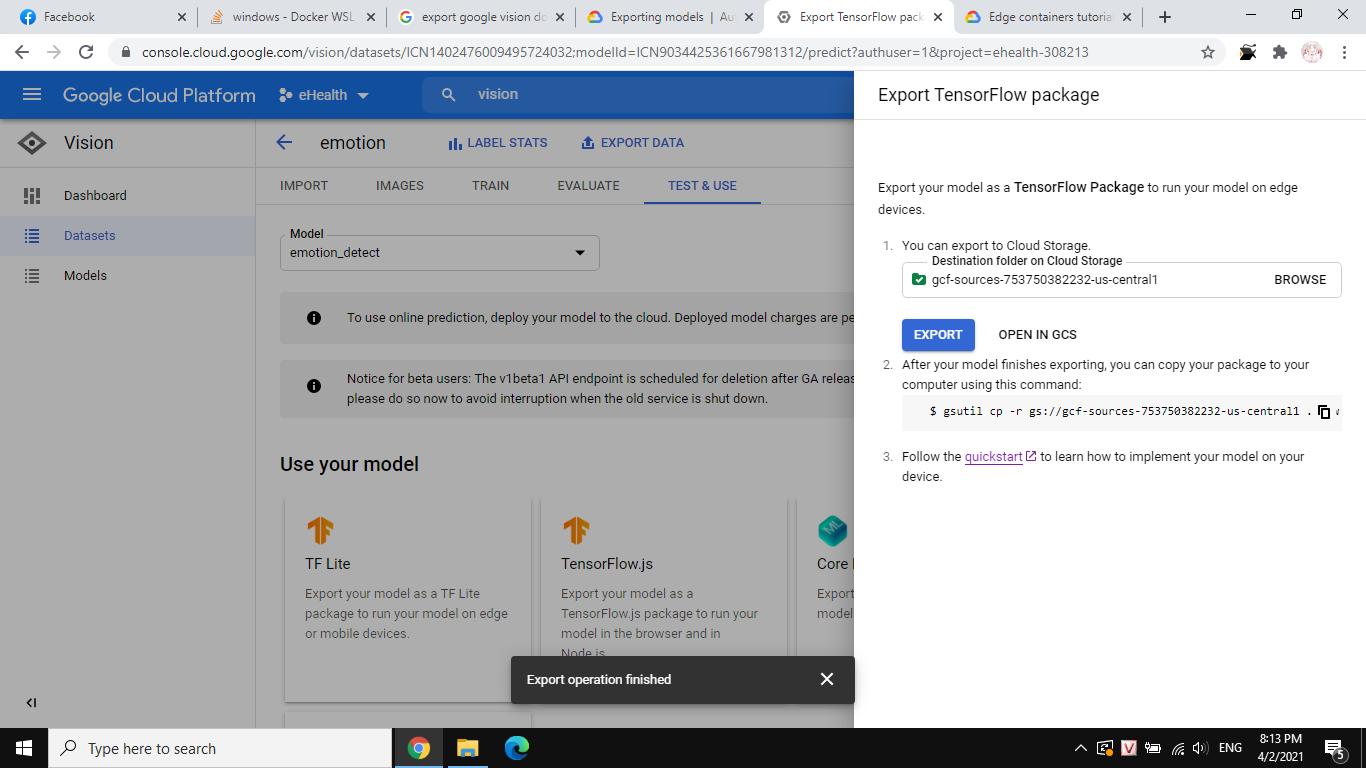


Hình 3. 28: Màn hình train model

Tiếp tục chọn next và tiến hành training model. Khi nhận mail train model thành công, ta sẽ có đánh giá sơ lược về model như bảng 3.2, sau đó ta tiến hành export model.

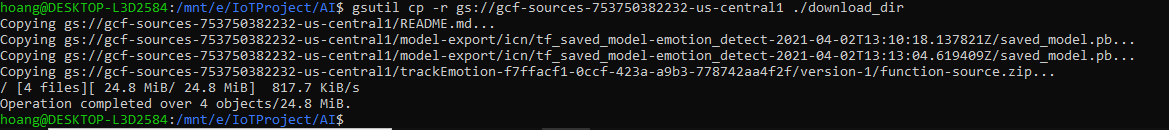
|  |  |
| --- | --- |
| Avarage precision | 0.738 |
| Precision | 78.32% |
| Recall | 54.67% |
| Train cost | 3.187 node hours |

Bảng 3. 2: Bảng kết quả độ chính xác của model emotion detect



Hình 3. 29: Màn hình export model

Tiếp theo ta tiến hành cài đặt google cloud sdk. Trong quá trình cài đặt dể tránh bị lỗi do thư mục có chưa ký tự ta tiến hành cài đặt WSL2. Sau khi cài đặt thành công ta export model



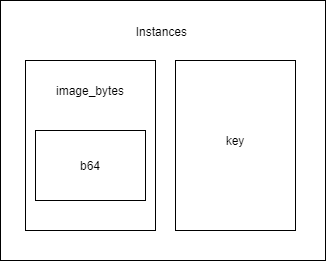
Hình 3. 30: Tải về model trong folder download\_dir

#### Tiến hành chạy và đóng gói

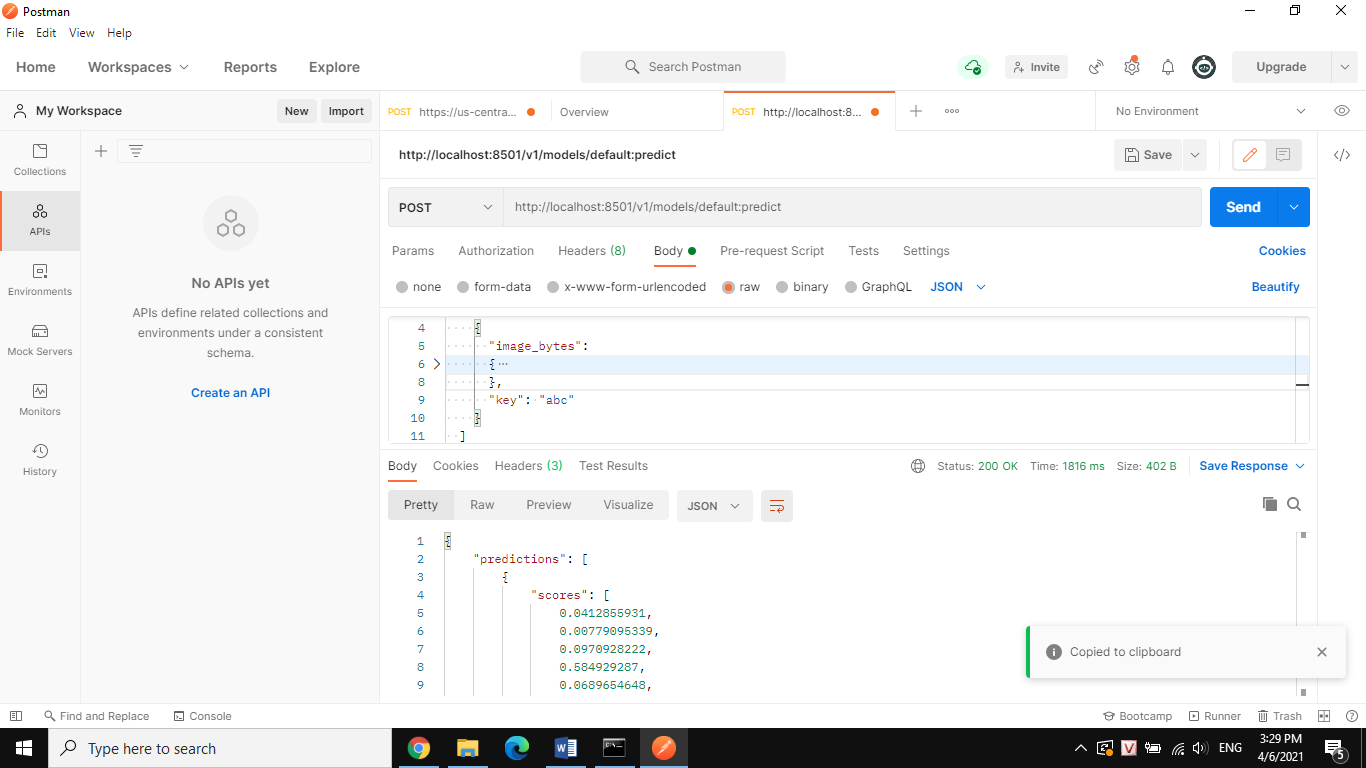
Lệnh dưới đây để chạy server emotion detect

*docker run --rm --name automl\_emotion -p 8501:8501 -v E:\IoTProject\AI\download\_dir\gcf-sources-753750382232-us-central1\model-export\icn:/tmp/mounted\_model/0001 -t gcr.io/cloud-devrel-public-resources/gcloud-container-1.14.0:latest*

Ta sử dụng công cụ Postman để gửi và nhận gói tin HTTP với body có dạng JSON sau:

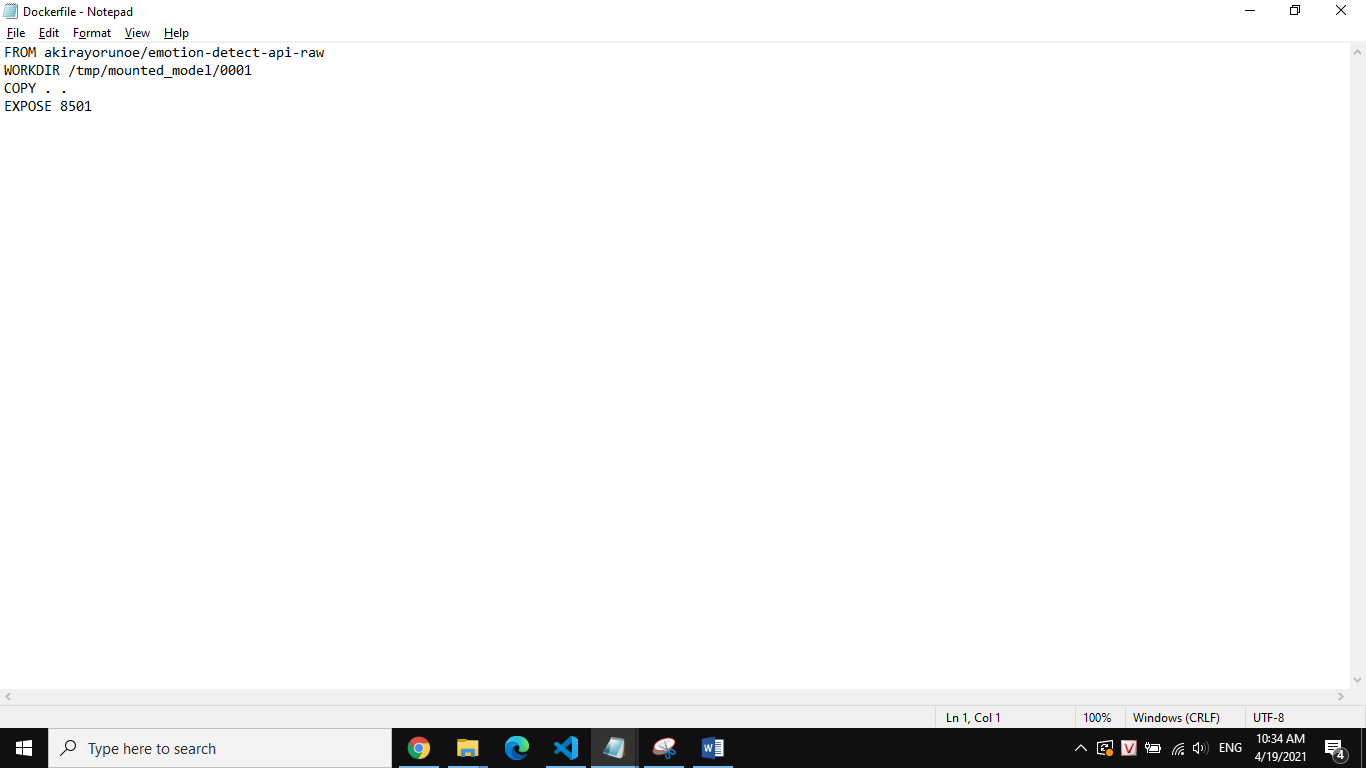


Hình 3. 31: Cấu trúc file JSON được gửi



Hình 3. 32: Kiểm thử với Postman

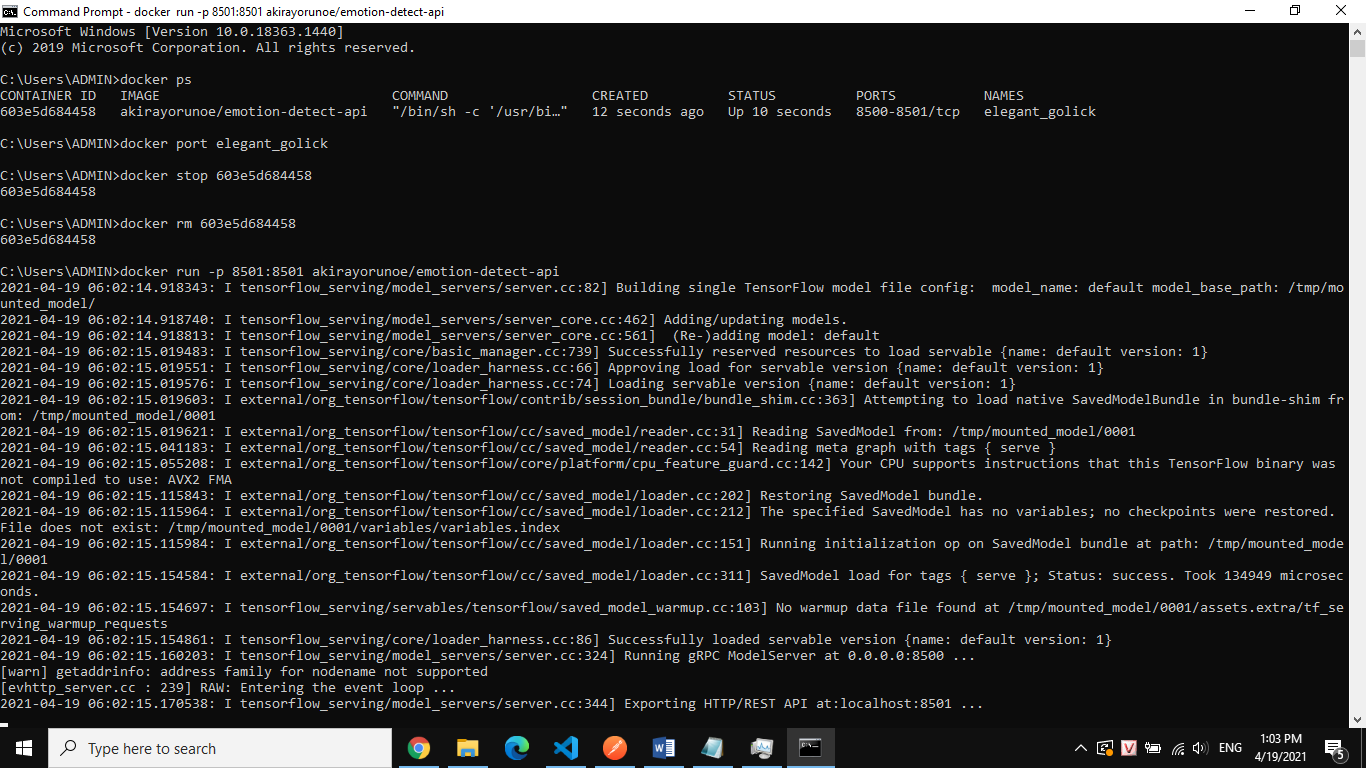
Sau khi chạy thành công, ta tiến hành sau đổi tên cho image với image “akirayorunoe/emotion-detect-api-raw” là tag name mà ta đặt khi container được tạo với lệnh chạy server emotion detect trên. Sau đó ta push container này lên Docker Hub. Nhưng nếu chỉ vậy thì khi pull container về ta sẽ bị lỗi do không tìm thấy model. Chính vì vậy ta tạo 1 file Dockerfile như sau để đưa model vào container rồi push lên Docker Hub



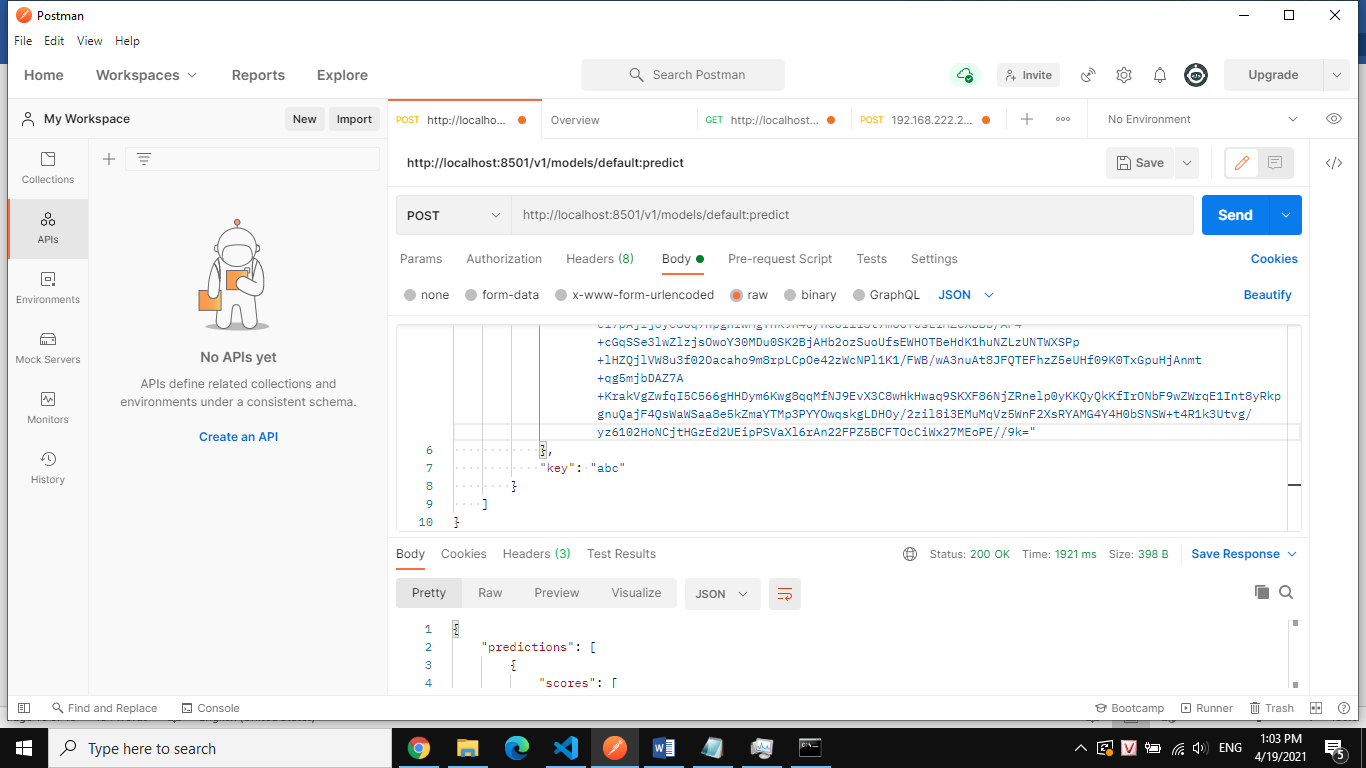
Hình 3. 33: Source code Dockerfile của server emotion detect

Ta test thử lại với container đã được deploy lên Docker Hub với lệnh

*docker run -p 8501:8501 akirayorunoe/emotion-detect-api*



Hình 3. 34: Chạy lại server emotion detect



Hình 3. 35: Kiểm lại với Postman

Ta sử dụng lệnh *docker stats* để kiểm tra việc sử dụng tài nguyên của server khi chạy trên Docker container

Capturee

Hình 3. 36: Tài nguyên sử dụng của emotion detect server

Như vậy, server emotion detect chỉ sử dụng khoảng 56MB nên hoàn toàn có thể triển khai trên thiết bị Raspberry Pi.

### Triển khai server trên Raspberry

Trên Raspberry Pi được sử dụng làm master ta thực hiện lệnh sau để cài k3s:

*curl -sfL https://get.k3s.io | sh –*

Sau đó ta dùng lệnh sau để lấy token: *cat /var/lib/rancher/k3s/server/node-token*

Token này được sử dụng cho các agent có thể kết nối với master.

Với Raspberry Pi được sử sụng làm agent ta thực hiện lệnh sau:

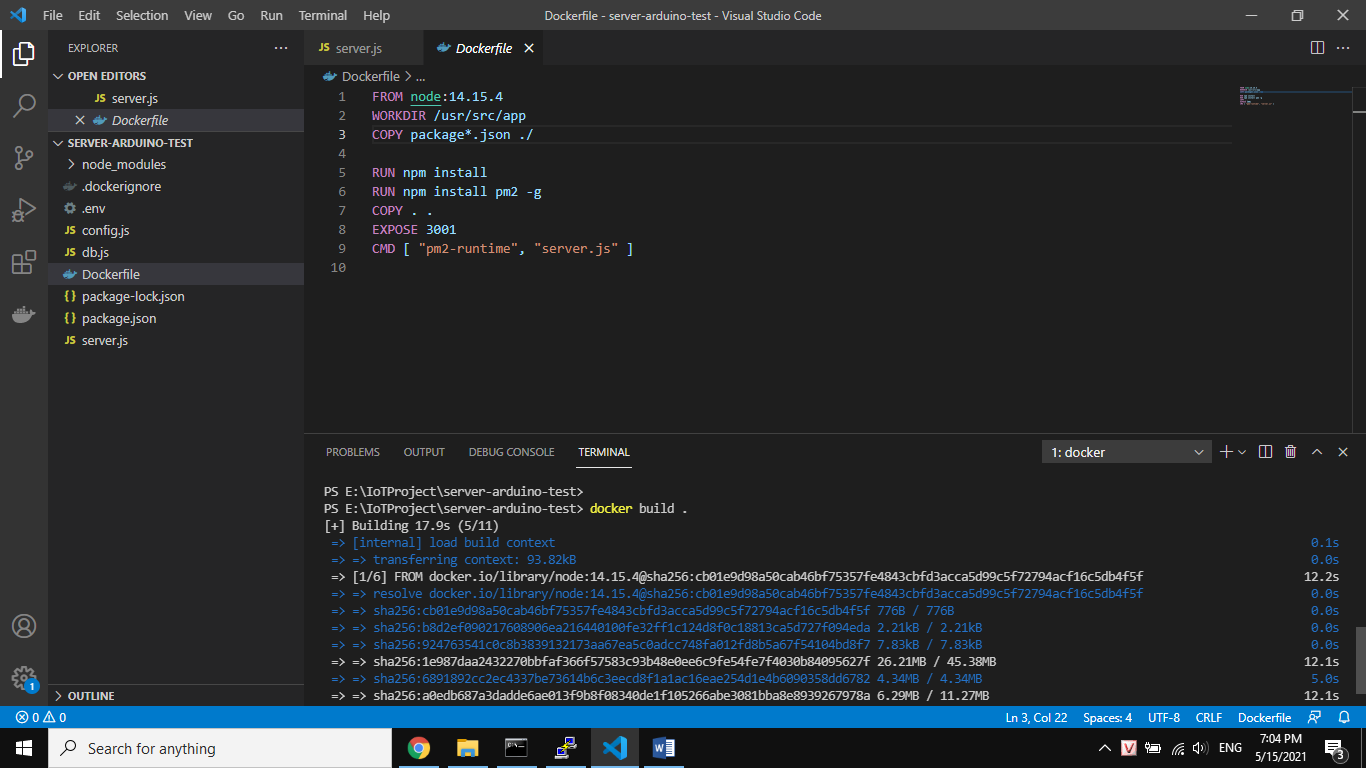
*export K3S\_URL="{ip của máy master}:6443"*

*export K3S\_TOKEN="{Token được lấy từ master}"*

*curl -sfL https://get.k3s.io | sh –*

Sau đó ta dùng lệnh *kubectl get nodes* để kiểm tra kết nối agent-master giữa các node

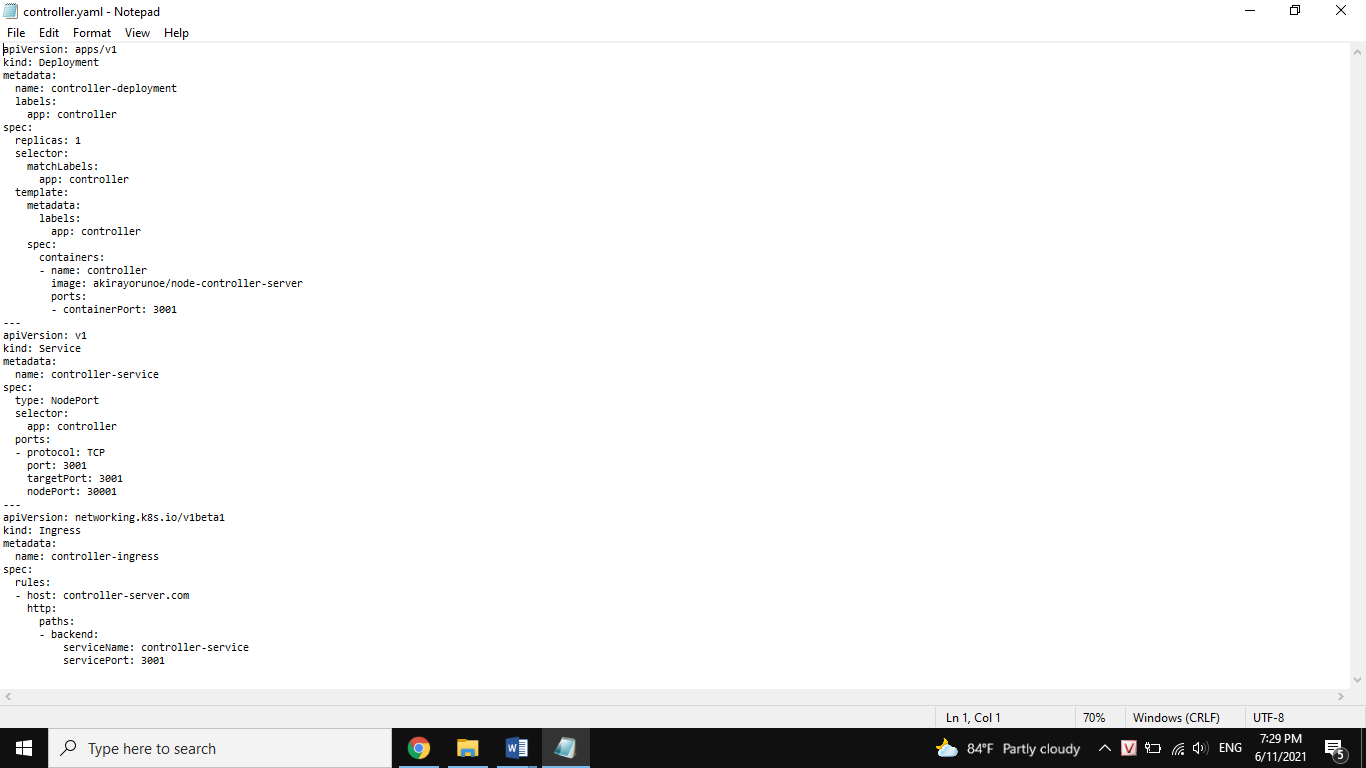
Ta có file Dockerfile sau để containerized server controller tại Edge



Hình 3. 37: Dockerfile của controller server

Sau khi build thành công ta push lên Docker Hub với tag name là “akirayorunoe/node-controller-server”

Tiếp theo, ta tạo file yaml với các loại (Deployment, Service) được dùng trong k8s.



Hình 3. 38: File YAML của controller server

Chi tiết như sau:

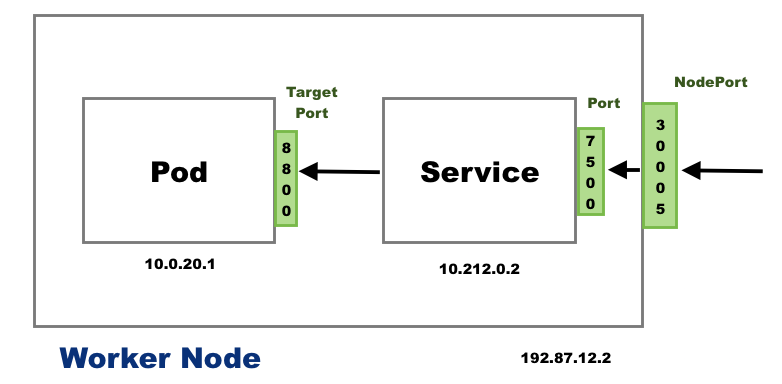
-Development: là 1 blueprint để có thể tạo ra pod (đơn vị nhỏ nhất trong kubernetes) từ đó tạo các container.

-Service: Dùng để thiết lập ip giao tiếp giữa các pod trong cluster, đưa traffic tới pod.

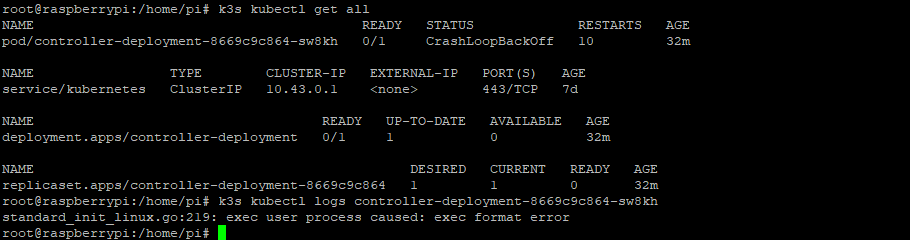
1. Port: Port của service, các pod khác trong cùng cluster có thể giao tiếp qua pod này.

2. TargetPort: Port mà service sẽ gửi request tới pod tương ứng, hay cũng port mà application chạy trong container.

3. NodePort: Nơi tiếp nhận request từ bên ngoài để chuyển tới TargetPort hay Port



Hình 3. 39: Ví dụ về các port khi tạo bằng file YAML

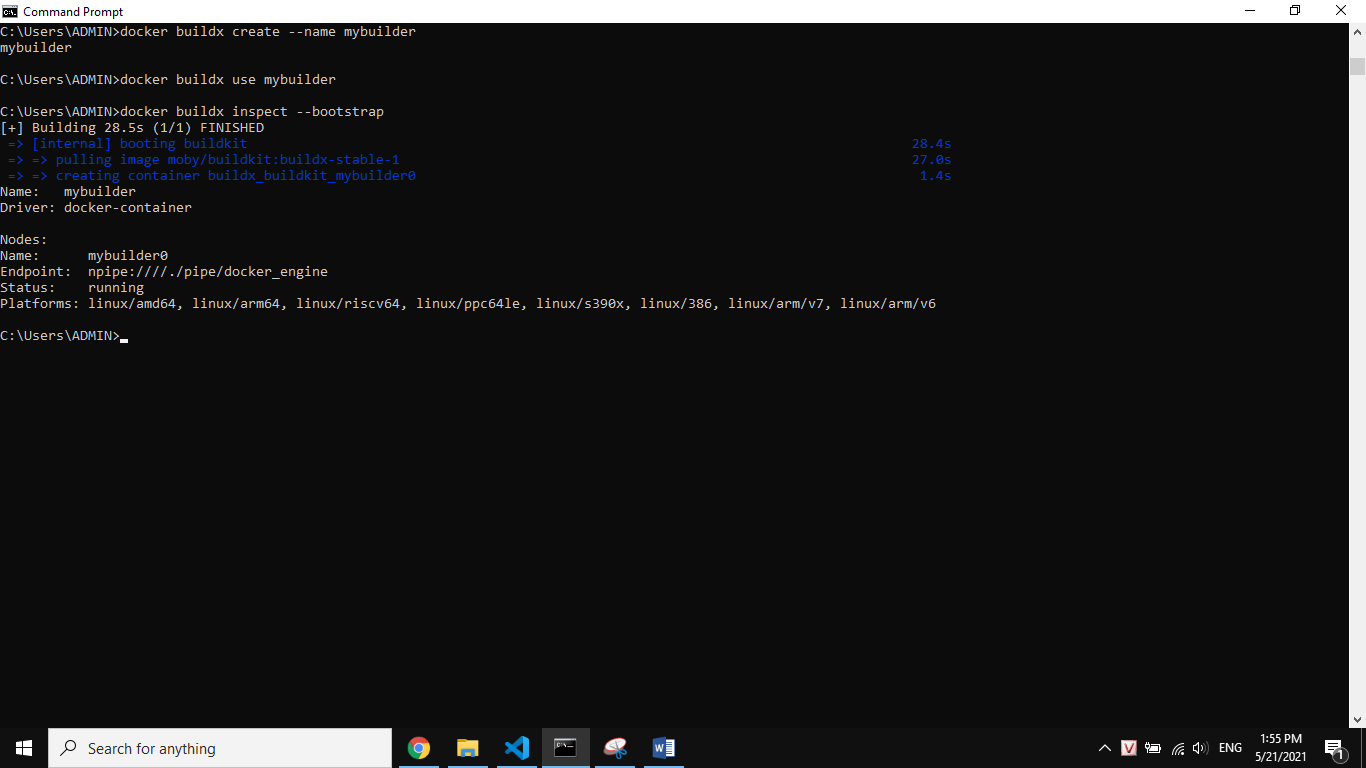


Hình 3. 40: Lỗi không tương thích OS

Khi chạy deployment trên Raspberry Pi ta thấy xuất hiện lỗi không tương thích OS

Để giải quyết vấn đề này, ta thực hiện multi-arch khi build container.

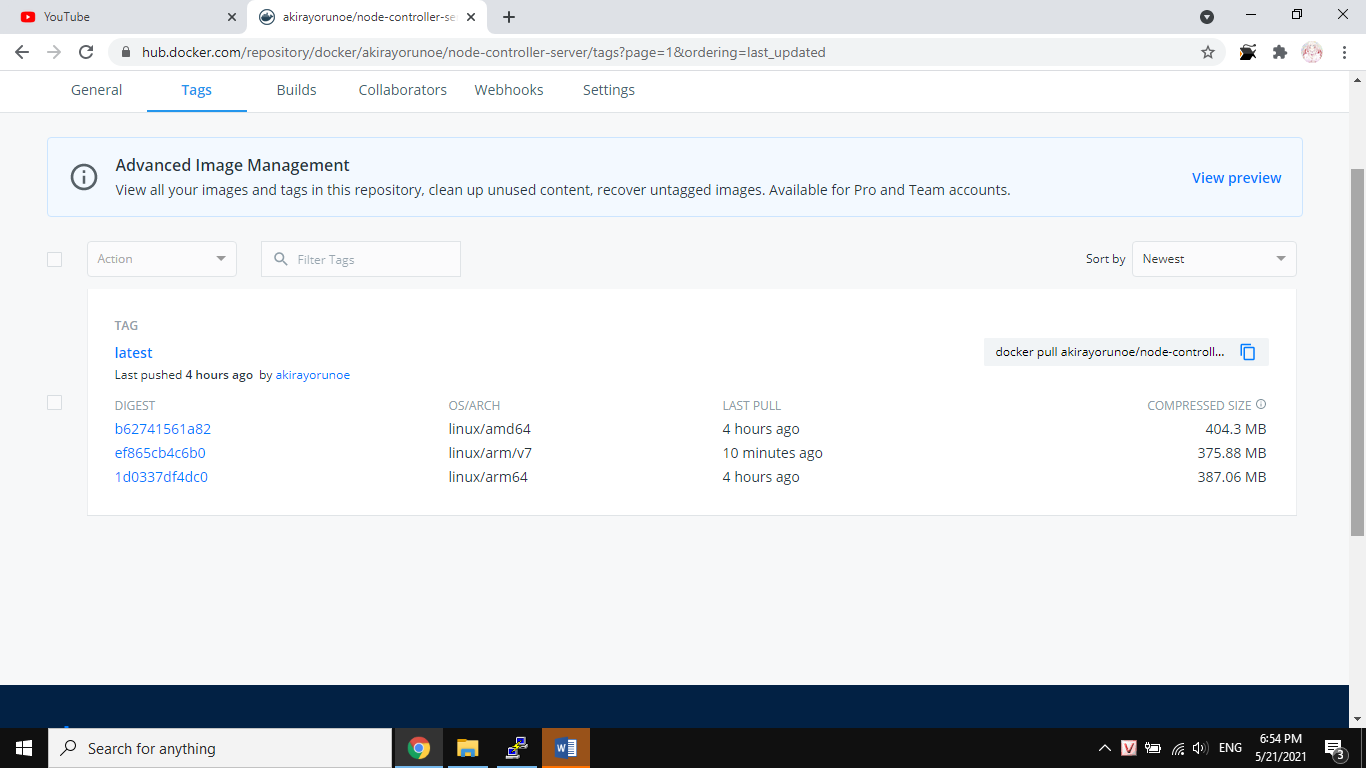
Ta thực hiện cài đặt và sử dụng buildx



Hình 3. 41: Cài đặt và dùng buildx để thực hiện multi-arch

Sau đó, ta thực hiện lệnh sau để thực hiện multi-arch và push lên Docker Hub

*docker buildx build --platform linux/amd64,linux/arm64,linux/arm/v7 -t akirayorunoe/node-controller-server:latest --push .tiến hành build multi-arch*



Hình 3. 42: Các OS/Arch được tạo thành công trên Docker Hub

Với điều này, sau khai báo image từ file yaml. Nó sẽ tự động lấy file thích hợp với OS và cài vào.

# TỔNG KẾT

Qua việc triển khai hệ thống theo mô hình, ta đã thực hiện thành công việc truyển nhận dữ liệu từ thiết bị MySignals cũng như dữ liệu từ app bệnh nhân trên di động từ đó đưa tới Edge để xử lý dữ liệu cùng với dự đoán cảm xúc bệnh nhân qua khuôn mặt từ đó lưu trữ giá trị cần thiết tới Cloud tại Firebase.

Với việc xử lý tại Edge giúp cho dữ liệu nhận được giảm độ trễ cũng như xử lý thông tin một cách nhanh chóng hơn.

TÀI LIỆU THAM KHẢO

[1] Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare [Online] *https://www.sciencedirect.com/science/article/abs/pii/S0167739X17307677*

[2] Y tế thông minh và vạn vật kết nối: Cơ hội và thách thức [Online] *https://ehealth.gov.vn/?action=News&newsId=45023*

[3] Y tế điện tử (e-health) là gì ?[Online] http://spiroupharmablog.blogspot.com/2010/08/y-te-ien-tu-e-health-la-gi.html

[4] V. M. Rohokale, N. R. Prasad and R. Prasad “A Cooperative Internet of Things (IoT) for Rural Healthcare Monitoring and Control.” in *Proc. 2011 2nd Int. Conf. Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology*, Chennai, 2011, pp. 1–6.

[5] IoT analytics [Online] [*https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/*](https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/)

[6] D. He and S. Zeadally, “An Analysis of RFID Authentication Schemes for Internet of Things in Healthcare Environment Using Elliptic Curve Cryptography,” *IEEE Internet of Things Journal*, vol.2, issue. 1, pp. 72–83, Feb. 2015.

[7] Edge computing in smart health care systems: Review, challenges, and research directions [Online] *https://onlinelibrary.wiley.com/doi/10.1002/ett.3710*

[8] GOOGLE CLOUD PLATFORM LÀ GÌ? [Online] *https://gcloudvn.com/kienthuc/google-cloud-platform-la-gi/*

[9] Kubernetes là gì? Cùng tìm hiểu cách hoạt động [Online] *https://topdev.vn/blog/kubernetes-la-gi/*

[10] World Federation of Societies of Anaesthesiologists [Online] *https://resources.wfsahq.org/atotw/pulse-oximetry-part-1/*