|  |  |
| --- | --- |
| **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  **KHOA CÔNG NGHỆ THÔNG TIN**  **BỘ MÔN CÔNG NGHỆ PHẦN MỀM**  **NGUYỄN QUANG TÍNH**  **NGUYỄN TIẾN TRIỂN**  **NGUYỄN CHÍ TRUNG**    **TÌM HIỂU VÀ XÂY DỰNG ỨNG DỤNG F.CLOUD CONSOLE**  **ĐỒ ÁN TỐT NGHIỆP CỬ NHÂN CNTT**  **TP. HCM, 2019** | |
| **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  **KHOA CÔNG NGHỆ THÔNG TIN**  **BỘ MÔN CÔNG NGHỆ PHẦN MỀM**  **NGUYỄN QUANG TÍNH – 1512576**  **NGUYỄN TIẾN TRIỂN – 1512599**  **NGUYỄN CHÍ TRUNG – 1512613**  **TÌM HIỂU VÀ XÂY DỰNG ỨNG DỤNG F.CLOUD CONSOLE**  **ĐỒ ÁN TỐT NGHIỆP CỬ NHÂN CNTT**  **GIÁO VIÊN HƯỚNG DẪN** TS. NGUYỄN TRẦN MINH THƯ **KHÓA 2015 - 2019** | |
| **NHẬN XÉT HƯỚNG DẪN ĐỒ ÁN TỐT NGHIỆP**  **Đề tài:** Tìm hiểu và xây dựng ứng dụng F.Cloud Console  **GVHD:** TS. Nguyễn Trần Minh Thư  **SVTH:** Nguyễn Quang Tính – 1512576  Nguyễn Tiến Triển – 1512599  Nguyễn Chí Trung – 1512613   1. Bối cảnh đề tài: 2. Bố cục và hình thức: 3. Nội dung: 4. Về lý thuyết: 5. Về chương trình: 6. Về thái độ 7. Trình độ: 8. Đánh giá chung: 9. Xếp loại:     TP.HCM, ngày 10 tháng 07 năm 2019  GVHD  Nguyễn Trần Minh Thư |
| **NHẬN XÉT PHẢN BIỆN KHÓA LUẬN – HỆ CHÍNH QUI**  **Đề tài:** Tìm hiểu và xây dựng ứng dụng F.Cloud Console  **Giảng viên hướng dẫn:** TS Nguyễn Trần Minh Thư,  **Sinh viên thực hiện:** Nguyễn Quang Tính, Nguyễn Tiến Triển,  Nguyễn Chí Trung  Những đóng góp của nhóm sinh viên cho chủ đề này:  Một số điểm cần cải thiện của báo cáo:  **Kết luận:**  **Xếp loại:**  Tp.HCM, ngày 10 tháng 07 năm 2019  Giảng viên phản biện |

LỜI CÁM ƠN

Chúng tôi xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên TP.HCM và công ty FUJINET SYSTEMS JSC đã tạo điều kiện tốt cho chúng tôi thực hiện đề tài này.

Chúng tôi xin chân thành cảm ơn TS. Nguyễn Trần Minh Thư đã tận tình hướng dẫn, chỉ bảo chúng tôi trong suốt thời gian thực hiện đề tài. Chúng tôi cũng xin cảm ơn anh Lê Văn Thân và anh Trương Duy Khánh (công ty Fujinet Systems JSC) đã có những trao đổi, những chỉ dẫn giúp chúng tôi giải quyết các vấn đề và hoàn thiện đề tài.

Chúng tôi cũng xin gửi lời cảm ơn sâu sắc đến quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng tôi những kiến thức quý báu trong những năm học vừa qua.

Chúng tôi xin gửi lòng biết ơn sâu sắc đến ba, mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng tôi trong những lúc khó khăn cũng như trong suốt thời gian học tập và nghiên cứu.

Mặc dù chúng tôi đã cố gắng hoàn thành đề tài trong phạm vi và khả năng cho phép, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, rất mong nhận được sự cảm thông và tận tình chỉ bảo của quý Thầy Cô, công ty và các bạn.

Tp.HCM, ngày 01 tháng 07 năm 2019

**Nhóm sinh viên thực hiện:**

Nguyễn Quang Tính

Nguyễn Tiến Triển

Nguyễn Chí Trung

ĐỀ CƯƠNG CHI TIẾT

|  |  |  |
| --- | --- | --- |
| **Tên Đề Tài**: Tìm hiểu và xây dựng ứng dụng F.Cloud Console | | |
| **Giáo viên hướng dẫn**: TS.Nguyễn Trần Minh Thư | | |
| **Người giám sát:** Lê Văn Thân, Trương Duy Khánh | | |
| **Thời gian thực hiện**: từ ngày 15/01/2019 đến ngày 30/06/2018 | | |
| **Loại đề tài:** Tìm hiểu công nghệ và xây dựng ứng dụng. | | |
| **Nội Dung Đề Tài**: | | |
| Client | * Tìm hiểu tổng quan HTML/CSS/Javascript * Tìm hiểu về framework Vuejs, Vuetify, Vuex * Tìm hiểu các Rest API * Tương tác với Server tạo thành ứng dụng hoàn chỉnh. | |
| Server | * Tìm hiểu ngôn ngữ lập trình Python với Framework Django để xây dựng Server. * Tìm hiểu về MySQL để xây dựng cơ sở dữ liệu. * Tìm hiểu Google Cloud Platform, NginX để xây dựng web server | |
| Other | * Tìm hiểu một số công cụ: Tortoise SVN, Swagger. | |
| **Kế Hoạch Thực Hiện:**   |  |  |  | | --- | --- | --- | | Thời gian | Nội dung công việc | Phân công | | Từ 15/01/2019 đến ngày 17/02/2019 | Làm quen và setup môi trường. | Cả 3 thành viên cùng thực hiện. | | Từ 18/02/2019 đến ngày 25/02/2019 | Nhận và phân tích đề tài. | Cả 3 thành viên cùng thực hiện. | | Từ 26/02/2019 đến ngày 12/03/2019 | Phân tích và thiết kế (Thiết kế sơ đồ Use Case, thiết kế giao diện Prototype, thiết kế Database,Class Diagram, thiết kế thông số Input, Output). | * Nguyễn Tiến Triển: Thiết kế sơ đồ Use Case, thiết kế giao diện Prototype, thiết kế thông số Input, Output. * Nguyễn Chí Trung: Thiết kế Database, Class Diagram, thiết kế giao diện Prototype, thiết kế thông số Input, Output. * Nguyễn Quang Tính: Thiết kế sơ đồ Use Case, thiết kế giao diện Prototype, thiết kế thông số Input, Output. | | Từ 13/03/2019 đến ngày 31/05/2019 | Tiến hành lập trình ứng dụng (gồm cả 2 phần: Client và Server) và kiểm thử ứng dụng. | * Nguyễn Tiến Triển: phụ trách lập trình phần Client. * Nguyễn Chí Trung: phụ trách lập trình phần Server. * Nguyễn Quang Tính: lập trình Client và Server. | | Từ 01/06/2019 đến ngày 14/06/2019 | Tạo tài liệu hướng dẫn Deployment và viết báo cáo đề tài. | * Nguyễn Tiến Triển: phụ trách viết báo cáo đề tài. * Nguyễn Chí Trung: deployment và viết báo cáo đề tài. * Nguyễn Quang Tính: deployment và viết báo cáo đề tài | | Từ 15/06/2019 đến ngày 21/06/2019 | Viết tài liệu User-Guide. | Cả 3 thành viên cùng thực hiện | | | |
| **Xác nhận của GVHD** | | **Ngày 01 tháng 07 năm 2019**  **Nhóm SV Thực hiện**  **Nguyễn Quang Tính**  **Nguyễn Tiến Triển**  **Nguyễn Chí Trung** |

MỤC LỤC

Table of Contents

[LỜI CÁM ƠN 2](#_Toc10118491)

[ĐỀ CƯƠNG CHI TIẾT 3](#_Toc10118492)

[MỤC LỤC 5](#_Toc10118493)

[TÓM TẮT ĐỒ ÁN 8](#_Toc10118494)

[**Chương 1** 9](#_Toc10118495)

[**Mở đầu** 9](#_Toc10118496)

[**1.1** **Giới thiệu chung** 9](#_Toc10118497)

[**1.2** **Mục tiêu đề tài** 9](#_Toc10118498)

[**1.3** **Yêu cầu cụ thể phía Công ty** 9](#_Toc10118499)

[**1.4** **Kết luận** 11](#_Toc10118500)

[**Chương 2** 12](#_Toc10118501)

[**Hiện trạng các ứng dụng tương tự hiện có** 12](#_Toc10118502)

[**2.1. Các ứng dụng hiện có trên thị trường** 12](#_Toc10118503)

[**2.1.1. Ứng dụng Google Cloud Platform** 12](#_Toc10118504)

[2.1.2. **Ứng dụng Amazon web services** 13](#_Toc10118505)

[**2.1.3. Một số ứng dụng khác và kết luận** 15](#_Toc10118506)

[**Chương 3** 17](#_Toc10118507)

[**Hệ thống F.Cloud Console** 17](#_Toc10118508)

[3.1. **Phát biểu phạm vi hệ thống** 17](#_Toc10118509)

[3.2. Phân tích, thiết kế các chức năng hệ thống: 18](#_Toc10118510)

[3.2.1. Sơ đồ User-case: 18](#_Toc10118511)

[3.2.2. Đặc tả cho từng chức năng: 19](#_Toc10118512)

[**Chương 4** 60](#_Toc10118513)

[**Công nghệ và môi trường triển khai** 60](#_Toc10118514)

[4.1. Môi trường triển khai: 60](#_Toc10118515)

[4.2. Giải pháp công nghệ sử dụng ở phía Client: 60](#_Toc10118516)

[4.2.1. Giới thiệu Vuejs: 60](#_Toc10118517)

[4.2.2. Giới thiệu Vuex: 61](#_Toc10118518)

[4.2.3. Giới thiệu Vuetify: 64](#_Toc10118519)

[4.2.4. Giới thiệu về NginX: 64](#_Toc10118520)

[4.2.5. Vấn đề HTTP client – Axios: 67](#_Toc10118521)

[4.3. Giải pháp công nghệ sử dụng ở phía server 70](#_Toc10118522)

[4.3.1. Ngôn ngữ Python 70](#_Toc10118523)

[4.3.2. Framework Django 71](#_Toc10118524)

[4.3.3. RESTful API web service: 72](#_Toc10118525)

[4.3.4. Giới thiệu hệ quản trị CSDL MySQL 72](#_Toc10118526)

[4.4. Kiến trúc mô hình 73](#_Toc10118527)

[4.4.1. Mô hình tổng quan về giao tiếp giữa Client – Server 73](#_Toc10118528)

[4.4.2. Giao tiếp và làm việc với MySQL 75](#_Toc10118529)

[4.4.3. Mô hình Authentication JWT: 75](#_Toc10118530)

[Chương 5 76](#_Toc10118531)

[Cài đặt minh họa ứng dụng 76](#_Toc10118532)

[5.1. Chức năng đăng nhập: 76](#_Toc10118533)

[5.2. 76](#_Toc10118534)

[Chương 6 77](#_Toc10118535)

[**Kết luận và hướng phát triển** 77](#_Toc10118536)

[**6.1** **Kết luận** 77](#_Toc10118537)

[***6.1.1*** ***Những điều đã thực hiện được*** 77](#_Toc10118538)

[***6.1.2*** ***Những điều cần cải thiện*** 78](#_Toc10118539)

[**6.2** **Hướng phát triển** 78](#_Toc10118540)

TÓM TẮT ĐỒ ÁN

Đồ án được thực hiện theo hướng sử dụng công nghệ và xây dựng ứng dụng và được thực hiện như một dự án thực tế thuộc Khối phát triển phần mềm 1 của công ty Fujinet. Mục tiêu của đề tài là xây dụng ứng dụng F.Cloud Console trên nền tảng web nhằm giới thiệu và là cổng giao tiếp của người dùng cuối với khó ứng dụng Cloud service của Fujinet (Face Detection, Face Recognition, OCR, Fire Detection…). Từ đó các công ty, doanh nghiệp nói chung và Fujinet nói riêng có thể đăng ký, thanh toán, sử dụng các ứng dụng một cách dễ dàng và thuận tiện. Khách hàng sẽ không còn phải khó khăn trong việc cài đặt hay thanh toán các ứng dụng, mọi thao tác sẽ dễ dàng được thực hiện trên website.Phía Client do sinh viên Nguyễn Tiến Triển và Nguyễn Quang Tính đảm nhận. Phía Server do sinh viên Nguyễn Chí Trung và Nguyễn Quang Tính đảm nhận.

Nội dung của quyển báo cáo cho đồ án này được trình bày gồm các phần như sau:

* + - Giới thiệu chung về đề tài.
    - Hiện trạng các ứng dụng tương tự đang có trên thị trường.
    - Công nghệ và môi trường triển khai.
    - Cài đặt minh họa của ứng dụng.
    - Kết luận và hướng phát triển của ứng dụng trong tương lai.

# **Chương 1**

# **Mở đầu**

*Nội dung giới thiệu chung về đề tài thực hiện, lý do chọn, mục tiêu mong muốn đạt được và yêu cầu của phía công ty.*

## **Giới thiệu chung**

Trong bối cảnh cuộc cách mạng lần thứ 4 (cách mạng 4.0) đang được đầu tư và phát triển rất mạnh như hiện nay, các phần mềm trí tuệ nhân tạo dường như không còn xa lạ với mọi người. Khi vào một số doanh nghiệp, công ty lớn thay vì điểm danh thủ công như trước đây, thì bây giờ đã có phần mềm điểm danh tự động thay thế. Các vấn đề về an ninh, an toàn cũng được dễ dàng kiểm soát bằng những phần mềm như nhận diện khuôn mặt, thông báo đám cháy. Nhận thấy sự quan trọng của các phần mềm trí tuệ nhân tạo, phía công ty Fujinet đã cho ra một số sản phẩm như F.Smart Door, F.Escape, F.Smart People Couting để giúp các công ty, doanh nghiệp nói chung và công ty Fujinet nói riêng dễ dàng quản lý và phát triển.

Tuy nhiên các ứng dụng này vẫn còn riêng lẻ, chưa liên kết lại với nhau do đó gây khó khăn trong việc sử dụng phía khách hàng cũng, như sự kém hiệu quả và thuận tiện. Nhằm giải quyết các như cầu trên cần có một phần mềm để quản lý các ứng dụng này nhằm giúp phía người dùng có thể dễ dàng thanh toán và sử dụng. Xuất phát từ những nhu cầu trên việc xây dựng và phát triển ứng dụng F.Cloud Console là một quyết định hợp lý.

## **Mục tiêu đề tài**

Thực hiện đề tài này, điều đầu tiên chúng tôi mong muốn là xây dựng một phần mềm phục vụ giới thiệu các sản phẩm của Fujinet như F.Smart Door, F.Escape, F.Smart People Couting, cho phép người dùng tương tác đăng ký sử dụng, quản lý các dịch vụ đã đăng ký một cách dễ dàng và thuận tiện. Trong nội dung tìm hiểu của đề tài này, chúng tôi đặt ra mục tiêu thực hiện những công việc sau:

* Tìm hiểu các công nghệ, kiến trúc mô hình như: Vuejs, Django Framework, Nginx để xây dựng ứng dụng.
* Thực hiện tổng hợp, so sánh, đánh giá các ứng dụng là cổng giao tiếp của người dùng cuối với kho ứng dụng để có thể nhận định và tìm ra hướng làm phù hợp.
* Phân tích thiết kế và cài đặt ứng dụng F.Cloud Console với các công nghệ được chỉ định.
* Đánh giá ưu, khuyết điểm của hệ thống và tìm các giải pháp để khắc phục các nhược điểm. Tìm hiểu các công nghệ hứa hẹn, lên lộ trình tiếp tục phát triển hệ thống.

Tiếp theo chúng tôi mong muốn được hỏi hỏi thêm nhiều kiến thức chuyên ngành mới. Đồng thời, thông qua quá trình thực hiện đề tài chúng tôi cũng mong muốn có thể học hỏi, trau dồi thêm kiến thức về quy trình phần mềm, kỹ năng làm việc nhóm, kỹ năng làm việc độc lập, tự tìm hiểu, nghiên cứu các công nghệ, nâng cao kỹ năng lập trình và kỹ năng giải quyết vấn đề.

## **Yêu cầu cụ thể phía Công ty**

Công ty Fujinet System JSC và giáo viên hướng dẫn Nguyễn Trần Minh Thư, cũng là trưởng phòng R&D tại Khối phát triển phần mềm số 1 của công ty đưa ra yêu cầu xây dựng một hệ thống F.Cloud Console để quản lý các ứng dựng F.Smart Door, F.Escape, F.Smart People Couting với bối cảnh tại công ty Fujinet System JSC. Sau quá trình tìm hiểu đề tài và đề xuất ý tưởng với bên công ty và giáo viên hướng dẫn thì hệ thống F.Cloude Console được xây dựng với phạm vi yêu cầu của hệ thống được điều chỉnh lại như sau:

Ứng dụng gồm 2 phần:

**+ Server:**

- Quản lý tài khoản người dùng

- Nhận yêu cầu từ Client (Web Client) và gửi phản hồi lại..

- Truy xuất cở sở dữ liệu.

- Công nghệ chỉ định: Django Framework.

+ **Client:**

- Gửi yêu cầu cụ thể tới phía Server.

- Hiển thị giao diện quản lý person: thêm person, thêm group person, chỉnh sửa person…

- Hiển thị giao diện quản lý camera: thêm camera, thêm group camera, chỉnh sửa camera…

- Hiển thị giao diện quản lý process: thêm process, khởi động process, chỉnh sửa process…

- Hiển thị giao diện quản lý project: thêm mới project, mua function…

- Nhận phản hồi từ phía Server và hiển thị cho người dùng.

- Thực hiện các chức năng quản lý tài khoản: đăng nhập/đăng xuất, thay đổi mật khẩu...

- Công nghệ, framework chỉ định: VueJS, Vuetify

## **Kết luận**

Trên đây là phần giới thiệu nội dung đề tài và mục tiêu nhóm chúng tôi mong muốn đạt được sau khi thực hiện đề tài. Bên cạnh đó, nhóm cũng đưa ra các phạm vi yêu cầu của hệ thống mà giáo viên hướng dẫn và công ty Fujinet JSC thống nhất đề ra để nhóm bám sát thực hiện đề tài. Tiếp sau đây, trước khi tiến hành tìm hiểu công nghệ và xây dựng ứng dụng, nhóm chúng tôi sẽ khảo sát các ứng dụng tương tự hiện có trên thế giới để tham khảo các chức năng, cùng với ưu điểm, khuyết điểm của những ứng dụng này. Từ đó áp dụng, cải thiện chức năng cho ứng dụng nhóm sẽ thực hiện.

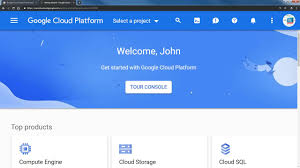
# **Chương 2**

# **Hiện trạng các ứng dụng tương tự hiện có**

* Nội dung Chương này giới thiệu các ứng dụng tương tự đã có trên thị trường cũng như ưu khuyết điểm của những ứng dụng này.

## **Các ứng dụng hiện có trên thị trường**

### **Ứng dụng Google Cloud Platform**

****

**Hình 2.1 Ứng dụng Google Platform**

[Google Cloud Platform](http://vicoders.com/google-cloud-platform-la-gi-va-chuc-nang-cua-google-cloud-platform/)**(GCP)** là nền tảng điện toán đám mây cho phép doanh nghiệp, tổ chức xây dựng và chạy các ứng dụng của mình trên chính hệ thống mà Google xây dựng. Các ứng dụng phổ biến của Google sử dụng Google Cloud Platform đang rất được ưa chuộng hiện nay đó là: Google Search, Google Maps, Google Apps, Chrome, Youtube, …**GCP** mang đến tất các các dịch vụ thiết yếu, bao gồm **Big Data, Storage, Compute Engine, Networking, Management, Developer Tools, Mobile.** Do đó, doanh nghiệp chỉ cần tập trung vào những công việc thật sự quan trọng với họ mà không cần phải quan tâm đến hệ thống bên dưới. **GCP** không chỉ mang đến cho doanh nghiệp những dịch vụ quen thuộc, điểm khác biệt còn nằm ở chỗ những dịch vụ đó được đặt tại **Google** – nơi có một hệ thống **Datacenter** với mức độ bảo mật và an toàn dữ liệu, đáp ứng các tiêu chuẩn khắt khe nhất đối với một hệ thống điện toán đám mây.

Các sản phẩm chính mà Google Cloud Platform cung cấp bao gồm:

* **Compute** – App Engine, Compute Engine, Container Engine
* **Storage** – Cloud Storage, Cloud Datastore, Cloud SQL, Cloud Bigtable
* **Big Data** – BigQuery, Cloud Dataflow, Cloud Dataproc, Cloud Pub/Sub
* **Services** – Cloud Endpoints, Translate API, Prediction API

**Google Cloud Platform** cung cấp các dịch vụ tích hợp và phát triển ứng dụng.

* **Nhận xét**: Số lượng rất lớn người dùng trên thế giới cùng với những đánh giá tích cực đã nói lên được chất lượng của ứng dụng. Bộ dịch vụ Google Cloud Platform luôn phát triển và Google thường xuyên giới thiệu. Các dịch vụ thay đổi hoặc ngừng cung cấp dịch vụ dựa trên nhu cầu của người dùng hoặc áp lực cạnh tranh.Qua quá trình sử dụng thử, nhóm chúng tôi thấy ứng dụng đáp ứng nhu cầu người dùng rất tốt, có thể xem đây là ứng dụng đang dẫn đầu trong danh mục các ứng dụng tương tự.

2.1.2. **Ứng dụng Amazon web services**

****

**Hình 2.2 Ứng dụng Amazon Web Services**

Amazon web services là một nền tảng điện toán đám mây phát triển toàn diện được cung cấp bởi [Amazon.com](http://amazon.com/). Dịch vụ Web đôi khi được gọi là dịch vụ đám mây hoặc các dịch vụ điện toán từ xa. Các dịch vụ AWS đầu tiên đã được đưa ra vào năm 2006 để cung cấp các dịch vụ trực tuyến cho các trang web và các ứng dụng phía máy khách.

 Bộ sưu tập AWS đang phát triển cung cấp hơn ba chục dịch vụ đa dạng bao gồm:

* CloudDrive cho phép người dùng tải và truy cập nhạc, video, tài liệu và ảnh từ các thiết bị kết nối Web. Dịch vụ này cũng cho phép người dùng phát nhạc trực tuyến tới thiết bị của họ
* CloudSearch, một dịch vụ tìm kiếm có thể mở rộng được sử dụng để tích hợp khả năng tìm kiếm tùy chỉnh vào các ứng dụng khác.
* Cơ sở dữ liệu Dynamo (còn gọi là DynamoDB hoặc DDB), một dịch vụ cơ sở dữ liệu NoSQL được quản lý đầy đủ được biết đến với độ trễ và khả năng mở rộng thấp
* Elastic Compute Cloud, cho phép các thuê bao kinh doanh chạy ứng dụng
* ElastiCache, dịch vụ bộ nhớ đệm được quản lý đầy đủ, tương thích với Memcached, một hệ thống lưu trữ bộ nhớ đối tượng được phân phối, mã nguồn mở, hiệu năng cao để tăng tốc các ứng dụng Web động bằng cách giảm tải cơ sở dữ liệu.
* Mechanical Turk, một giao diện chương trình ứng dụng (API) cho phép các nhà phát triển tích hợp trí thông minh con người vào các cuộc gọi thủ tục từ xa (RPC) sử dụng mạng con người để thực hiện các tác vụ mà máy tính không thích hợp.
* RedShift, dịch vụ kho dữ liệu quy mô petabyte được thiết kế cho các khối lượng công việc phân tích, kết nối với các khách hàng dựa trên SQL chuẩn và các công cụ thông minh kinh doanh.
* Dịch vụ Lưu trữ Đơn giản (S3), một dịch vụ có tốc độ cao, tốc độ thấp được thiết kế để sao lưu và lưu trữ trực tuyến các dữ liệu và các chương trình ứng dụng.
* **Nhận xét**: Amazon web services là một trong những đối thủ cạnh tranh với Google cloud console. Với số lượng người dùng lớn cùng những tính năng đáp ứng được nhu cầu người dùng, có thể thấy Amazon web services cũng là một ứng dụng nổi bật trong danh mục các ứng dụng tương tự. Tuy nhiên so với Google cloud console thì Amazon web services có vẻ ít được tin tưởng hơn do đã từng xảy ra sự cố, thuê lại đường truyền để kết nối các trung tâm dữ liệu và hiệu suất thấp hơn.

**2.1.3. Một số ứng dụng khác và kết luận**

Bên cạnh các ứng dụng có tên tuổi trong và ngoài nước như trên vẫn có một số ứng dụng khác cũng giải quyết các vấn đề về cổng giao tiếp của người dùng cuối với kho ứng dụng như: Oracle Cloud Console: cung cấp một số dịch vụ như Software-as-a-service (Doanh nghiệp trả một mức phí hàng tháng để sử dụng ứng dụng qua internet). Platform-as-a-service (Doanh nghiệp trả tiền để các nhà cung cấp thứ 3 host ứng dụng của họ, thường là những ứng dụng tự tạo theo nhu cầu của mỗi doanh nghiệp) và cuối cùng là Infrastructure-as-a-service(IaaS) (doanh nghiệp thuê tất cả phần cứng , tự mình set-up, và có thể host bất kỳ ứng dụng gì mà mình muốn)

* **Kết luận:** Trên đây là khảo sát của nhóm chúng tôi về các ứng dụng tương tự hiện có trên thị trường. Qua đó chúng tôi có được cái nhìn tổng quát về các ứng dụng có chức năng làm cổng giao tiếp của người dùng cuối với kho ứng dụng và có thêm cở sở để xây dựng các chức năng cho ứng dụng của đề tài. Tiếp sau đây, chúng tôi sẽ trình bày thiết kế hệ thống cho ứng dụng.

# **Chương 3**

# **Hệ thống F.Cloud Console**

* Nội dung chương này trình bày về phạm vi của hệ thống, chi tiết thiết kế các chức năng cho ứng dụng.
  1. **Phát biểu phạm vi hệ thống**

Từ những yêu cầu của phía công ty Fujinet được nêu trong phần 1.3, sau khi phân tích, chúng tôi xác định phạm vi của đề tài như sau:

***Về chức năng trên toàn hệ thống:***

* + - Quản lý resource ứng dụng F.Smart Door:
* Khởi tạo và quản lý nhân sự và các nhóm nhân sự.
* Khởi tạo và quản lý camera.
* Khởi tạo và quản lý các tiến trình sử dụng camera.

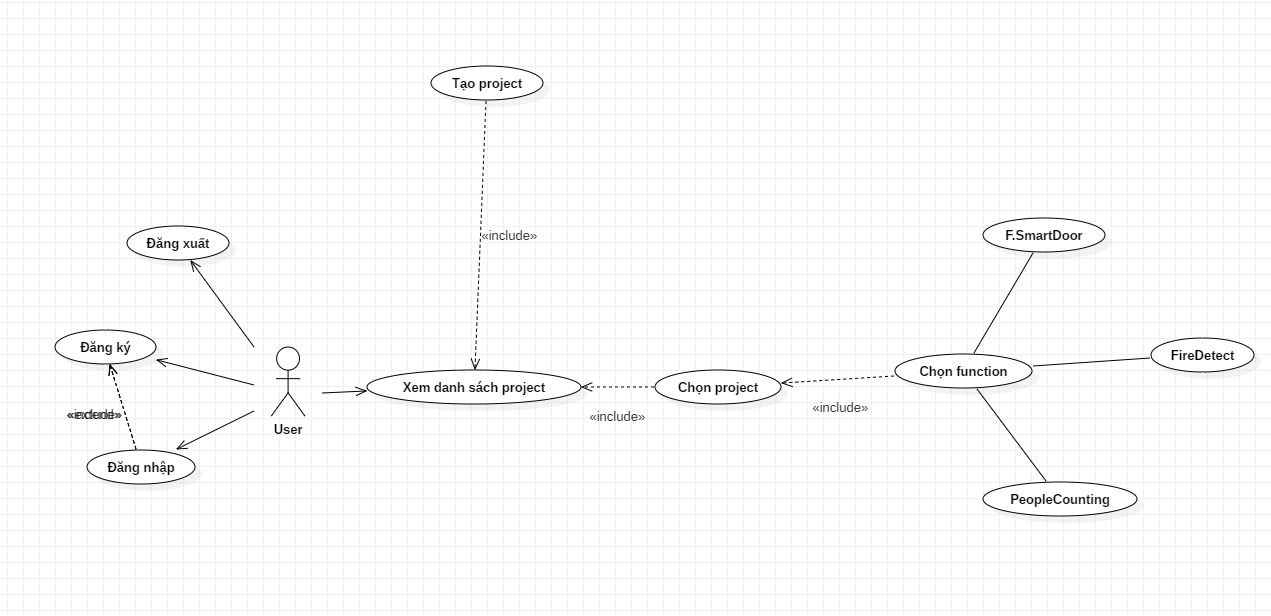
***Về các công nghệ sử dụng trên từng phần của ứng dụng:***

**Client:**

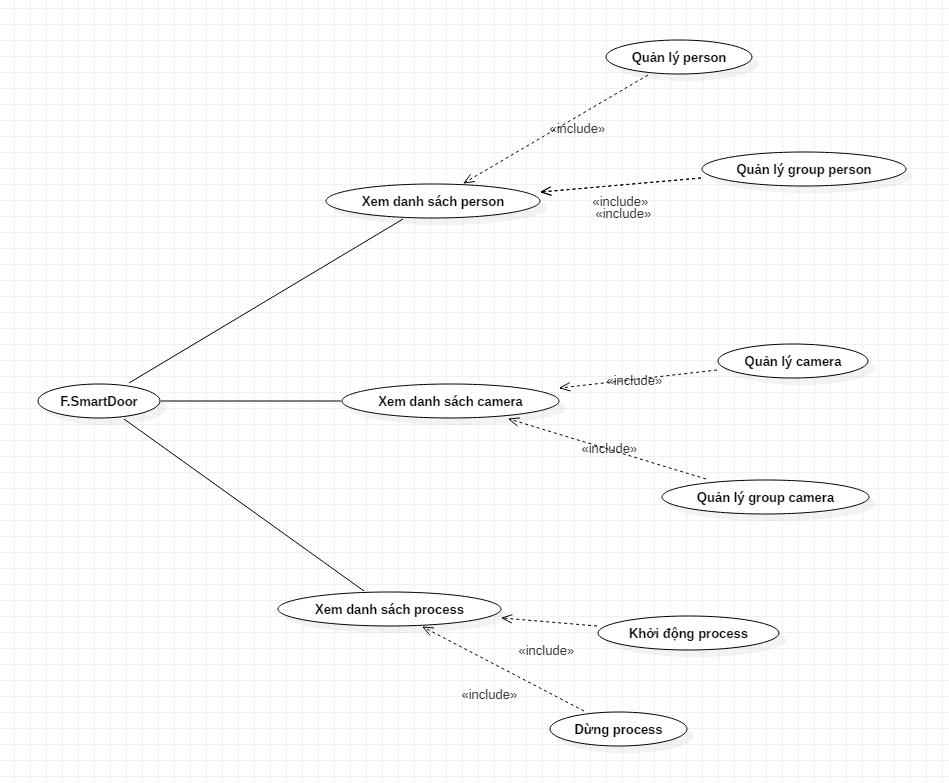
* + - Các framework Vuejs, Vuetify, Vuex
    - Tìm hiểu công nghệ hỗ trợ giao tiếp với Server.
    - Các thư viện hỗ trợ làm giao diện.

**Server:**

* + - Sử dụng một ngôn ngữ ngôn ngữ lập trình Python với Framework Django để xây dựng một Web Service đáp ứng request từ Client, xử lý và trả về kết quả cho Client..
    - Tìm hiểu công nghệ để deploy ứng dụng.
  1. **Phân tích, thiết kế các chức năng hệ thống:**
     1. **Sơ đồ Use-case:**



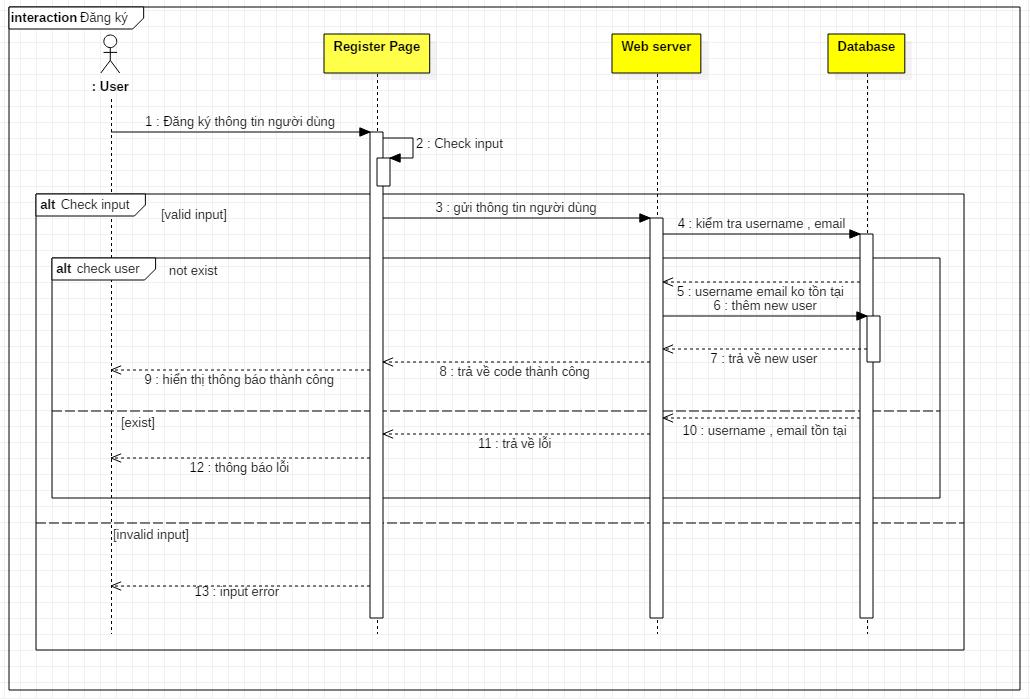
**Hình 3.1 Sơ đồ Use-case**



**Hình 3.2 Sơ đồ Use-case chi tiết**

* + 1. Đặc tả cho từng chức năng:
       1. Đăng ký

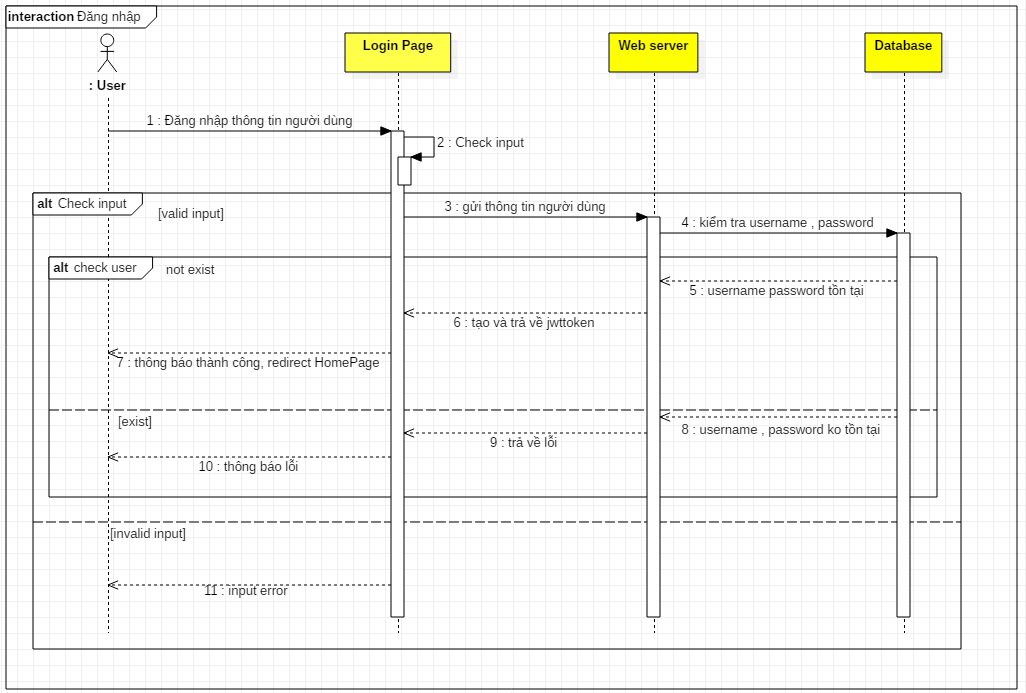
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC001 | |
| **Tên Use Case** | Đăng ký tài khoản người dùng. | |
| **Tóm tắt** | Use case cho phép người dùng tạo một tài khoản để đăng nhập vào ứng dụng. | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** |  | |
| **Kết quả** | Thông tin đăng ký của người dùng được lưu vào hệ thống để sử dụng cho việc đăng nhập sau này. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Nhập thông tin đăng ký tài khoản và nhấn nút Đăng ký |  |
|  | Kiểm tra thông tin về mặt vật lý (mật khẩu và mật khẩu xác nhận phải giống nhau, mật khẩu có đủ độ dài, …).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (thông tin tài khoản đã tồn tại chưa?).  Thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | - Khi thông tin về email, username bị trùng thì sẽ báo lỗi về cho người dùng.  - Khi các thông tin người dùng nhập bị lỗi thì thông báo lỗi sẽ trả về cho người dùng | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình 2.2.1 Sơ đồ tuần tự - Đăng ký***

* + - 1. Đăng nhập

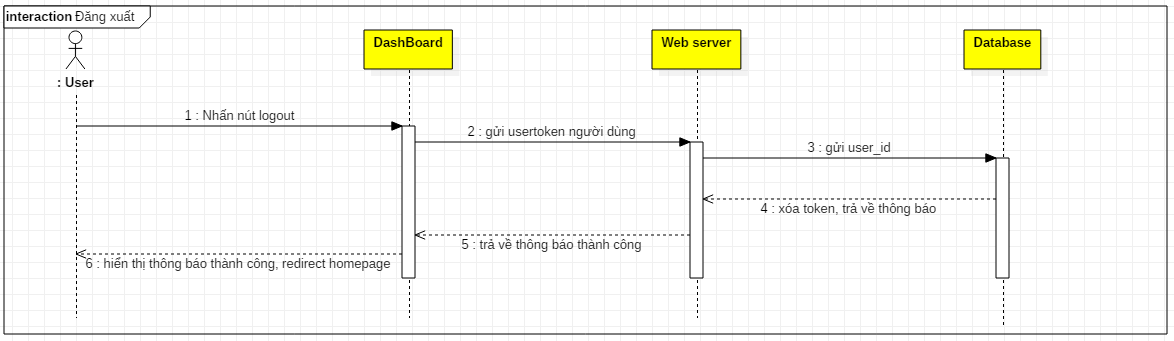
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC002 | |
| **Tên Use Case** | Đăng nhập tài khoản người dùng. | |
| **Tóm tắt** | Use case cho phép người dùng đăng nhập vào hệ thống | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã có tài khoản | |
| **Kết quả** | Người dùng đăng nhập thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Nhập thông tin đăng nhập tài khoản và nhấn nút Đăng nhập |  |
|  | Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin tài khoản(username tương ứng với password hay không)  Trả về trang Dashboard |
| **Kịch bản phụ** | - Khi thông tin về username hay password bị sai , thông báo lỗi sẽ trả về cho người dùng | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình 2.2.2 – Sơ đồ tuần tự - Đăng nhập***

* + - 1. Đăng xuất

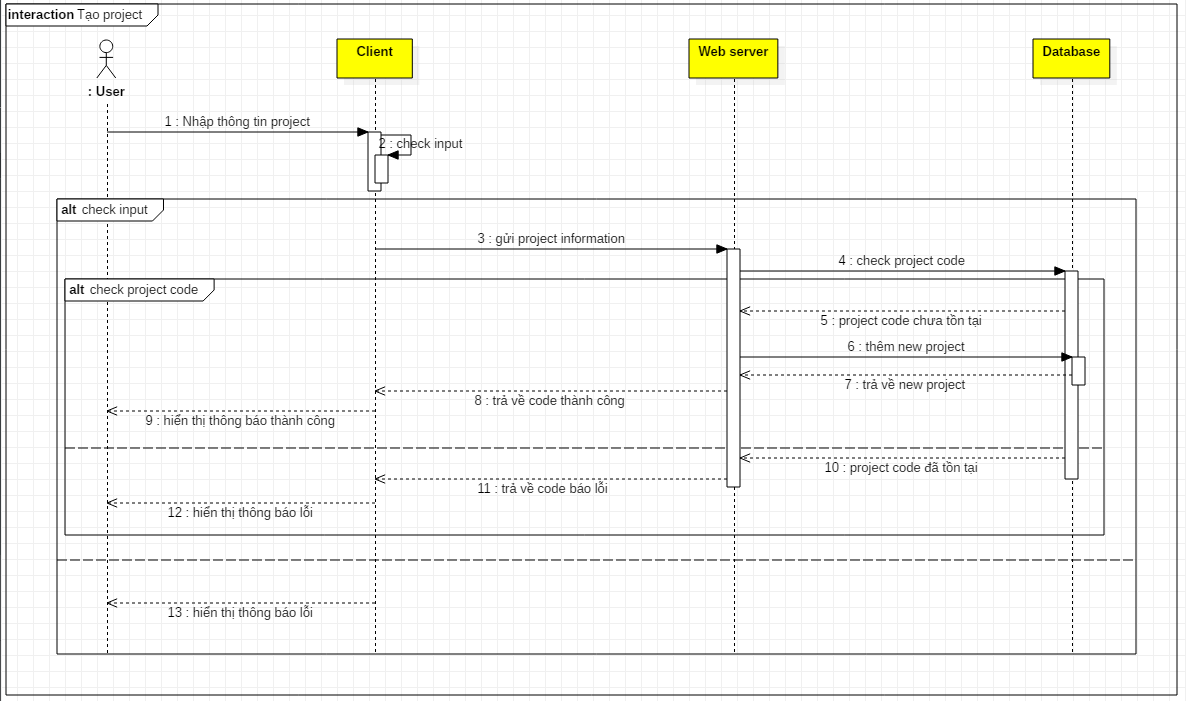
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC003 | |
| **Tên Use Case** | Đăng xuất tài khoản người dùng. | |
| **Tóm tắt** | Use case cho phép người dùng đăng xuất khỏi hệ thống | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã có tài khoản  Người dùng đã đăng nhập vào hệ thống | |
| **Kết quả** | Người dùng đăng xuất thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Người dùng nhấn nút đăng xuất trên Dashboard menu |  |
|  | Thông tin người dùng được gửi lên server. |
|  | Server kiểm tra trạng thái tài khoản của người dùng và tiến hành đăng xuất tài khoản |
| **Kịch bản phụ** |  | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình 2.2.3 Sơ đồ tuần tự - Đăng xuất***

* + - 1. Tạo project

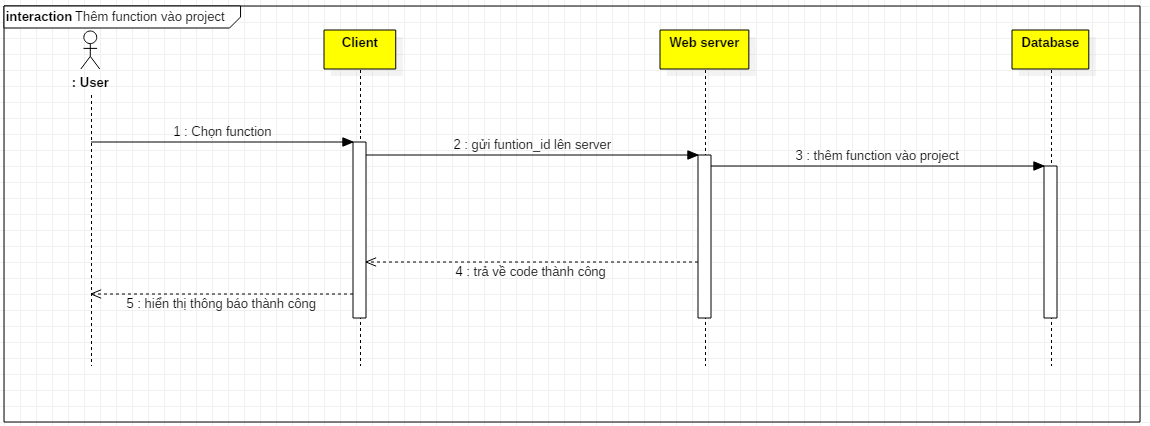
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC004 | |
| **Tên Use Case** | Tạo project mới | |
| **Tóm tắt** | Use case cho phép người dùng tạo một project mới | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng phải đăng nhập vào hệ thống | |
| **Kết quả** | Thông tin của project mới được lưu vào trong hệ thống | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Người dùng nhấn nút Chọn project ở trên thanh toolbar  - Người dùng nhấn nút Tạo project mới ở góc phải dialog mới xuất hiện  - Người dùng nhập Project code và project name và nhấn nút Khởi tạo |  |
|  | - Hệ thống kiểm tra thông tin về mặt login (project code có trùng hay không) |
|  | - Hệ thống tiền hành thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | - Khi thông tin về project\_code bị trùng thì sẽ báo lỗi về cho người dùng. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình 2.2.4 Sơ đồ tuần tự - Tạo project mới***

* + - 1. Thêm function

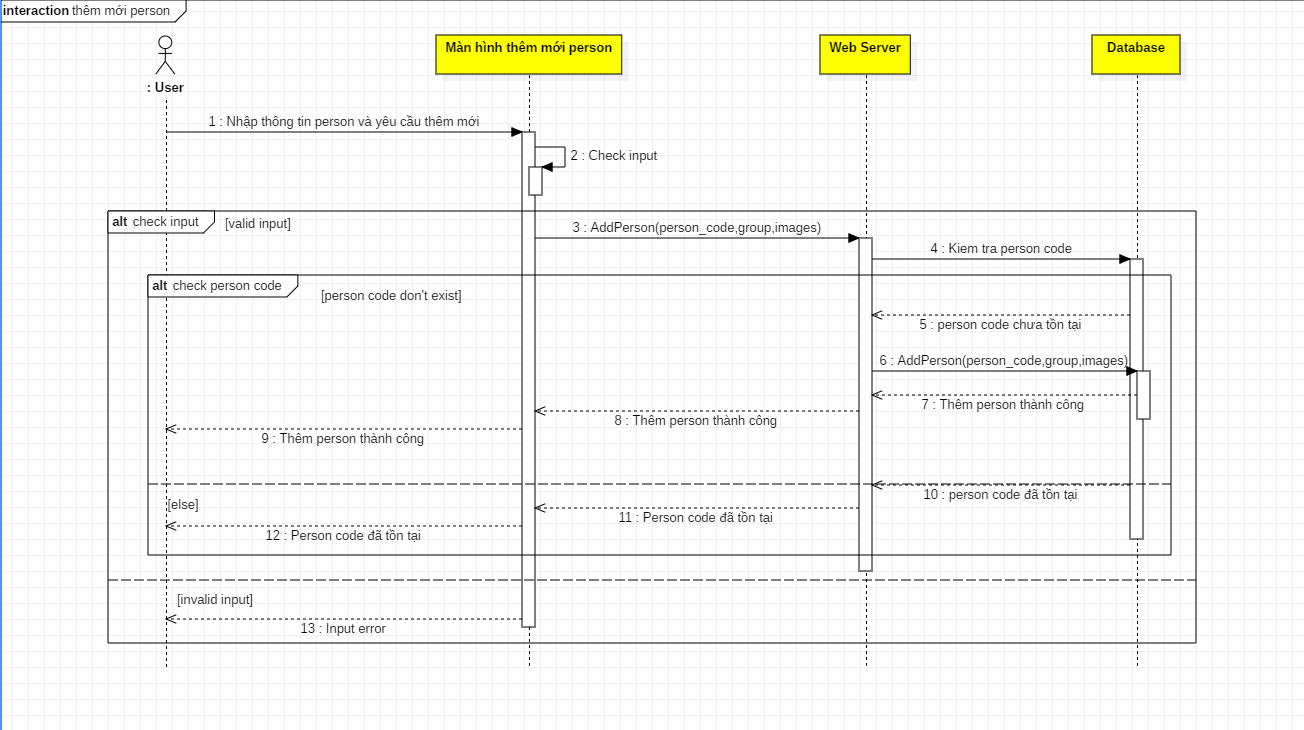
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC006 | |
| **Tên Use Case** | Thêm function vào project | |
| **Tóm tắt** | Use case cho phép người dùng thêm một function vào project | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | - Người dùng phải đăng nhập vào hệ thống  - Project được chọn chưa được thêm function | |
| **Kết quả** | - Người dùng thêm function thành công vào project | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Người dùng nhấn nút Menu ở trên thanh toolbar  - Người dùng nhấn chuột vào Function muốn thêm vào  - Người dùng nhấn nút Buy now |  |
|  | - Hệ thống kiểm tra thông tin của project và function  - Hệ thống thêm thông tin function vào dữ liệu |
| ­ | - Hệ thống thông báo thành công và trả về trang Function |
| **Kịch bản phụ** |  | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Thêm function vào project***

* + - 1. Thêm mới person

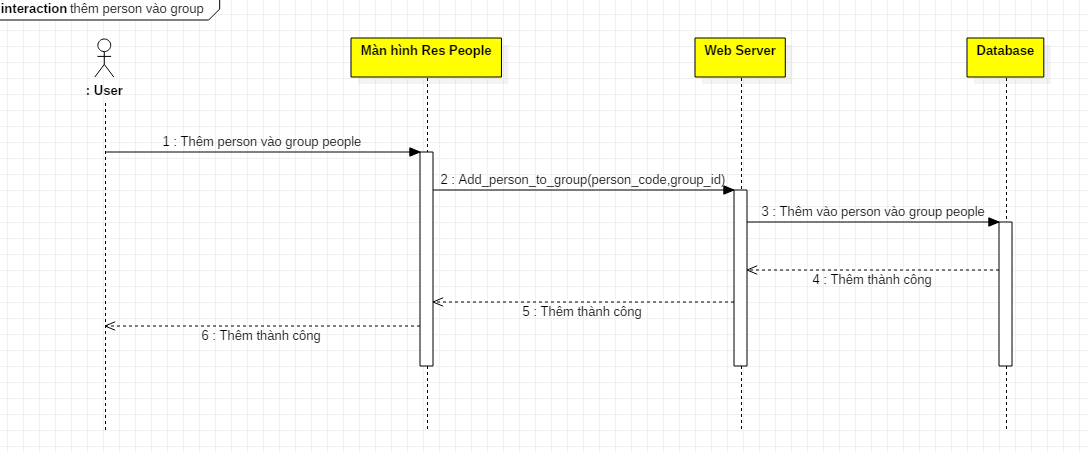
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC001 | |
| **Tên Use Case** | Thêm mới một person | |
| **Tóm tắt** | Use case cho phép người dùng tạo thêm mới một person trong res people | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. | |
| **Kết quả** | Person được thêm vào hệ thống, một số thông tin cơ bản lập tức được hiển thị lên giao diện. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện thêm mới person nhập đầy đủ thông tin và chọn hình ảnh, group people để thêm person.  Nhấn OK để thêm person. |  |
|  | Kiểm tra thông tin về mặt vật lý (person code phải đủ độ dài, không chứ ký tự đặc biệt…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Person đã tồn tại hay chưa?).  Thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu Person đã tồn tại thì không thêm vào cơ sở dữ liệu, thông báo lỗi về cho Client. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Thêm mới person***

* + - 1. Thêm person vào group people

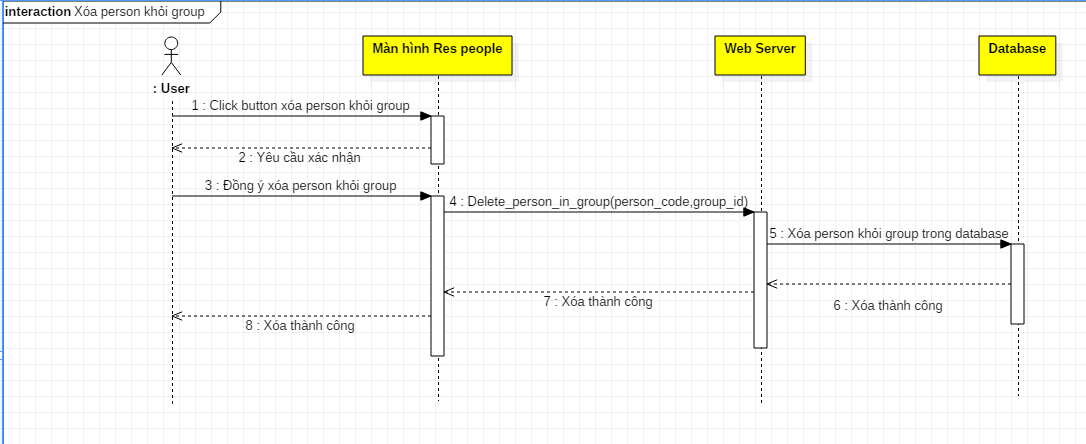
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC002 | |
| **Tên Use Case** | Thêm person vào group people | |
| **Tóm tắt** | Use Case này cho phép người dùng thêm person vào một hoặc nhiều group people nào đó | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. Person và group people đã được tạo trước đó. | |
| **Kết quả** | Person được thêm vào group people được chọn, giao diện hiển thị các group people mà person có trong đó. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Chọn các group people muốn thêm person vào.  Nhấn OK để thêm vào. |  |
|  | Thêm person vào các group people được chọn. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Thêm person vào group***

* + - 1. Xóa person khỏi group people.

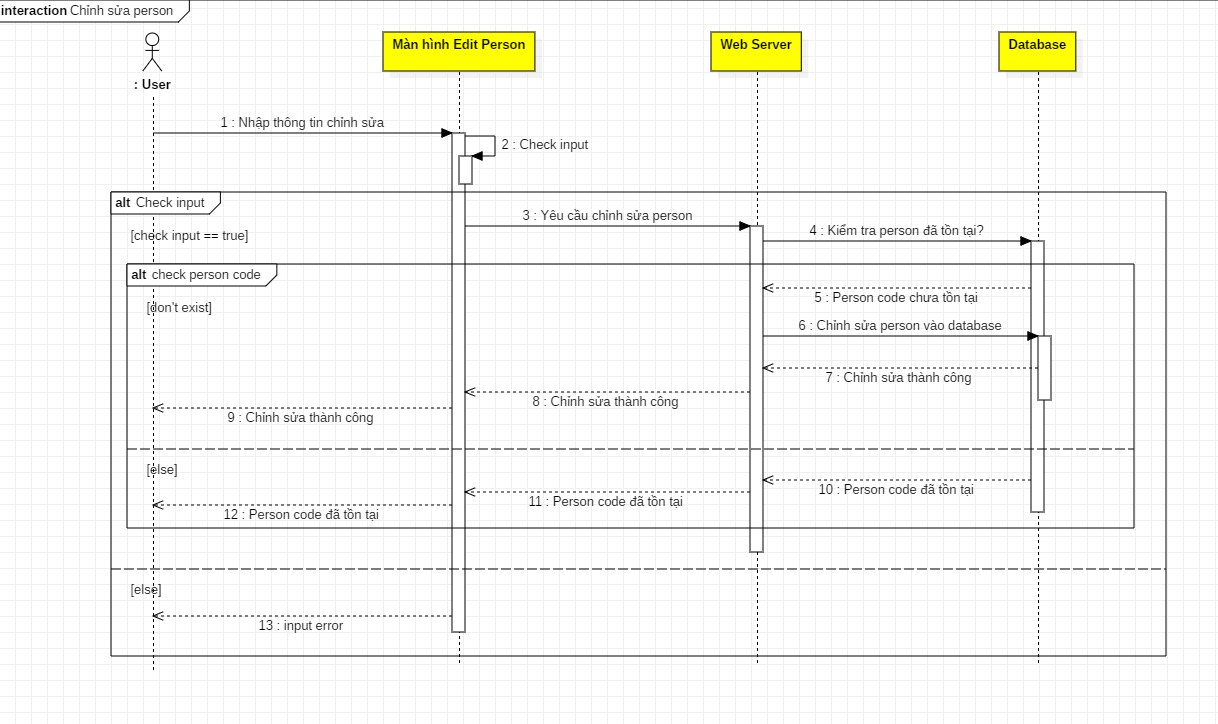
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC003 | |
| **Tên Use Case** | Xóa person khỏi group people | |
| **Tóm tắt** | Use Case này cho phép người dùng xóa person khỏi group people | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function và person có tồn tại trong group people. | |
| **Kết quả** | Person được xóa khỏi group, danh sách group của person được cập nhật. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện hiển thị danh sách các person, chọn dấu X tại group muốn xóa.  Thông báo hiện hiện lên, chọn Accept để xóa. |  |
|  | Serser lấy thông tin person và group được gủi đi, sau đó xóa person khỏi group. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Xóa person khỏi group***

* + - 1. Chỉnh sửa person

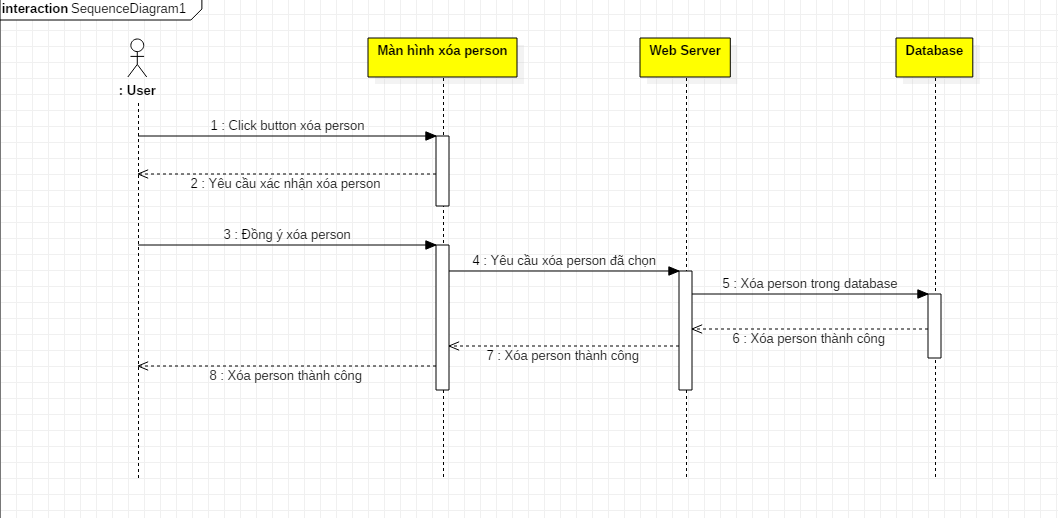
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC004 | |
| **Tên Use Case** | Chỉnh sửa person | |
| **Tóm tắt** | Use Case này cho phép người dùng chỉnh sửa thông tin, hình ảnh person. | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. Person đã được thêm vào trước khi thực hiện Use case này. | |
| **Kết quả** | Person được chỉnh sửa và cập nhật. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện chỉnh sửa person, điền đầy đủ các thông tin muốn thay đổi, chọn hình ảnh chỉnh sửa.  Nhấp OK để chỉnh sửa person |  |
|  | Kiểm tra thông tin chỉnh sửa về mặt vật lý (person code phải đủ độ dài, không chứa ký tự đặc biệt…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Person code thay đổi có bị trùng hay không?).  Thông tin được cập nhật vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu Person code mới đã tồn tại thì không chỉnh sửa vào cơ sở dữ liệu, thông báo lỗi về cho Client | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Chỉnh sửa person***

* + - 1. Xóa Person

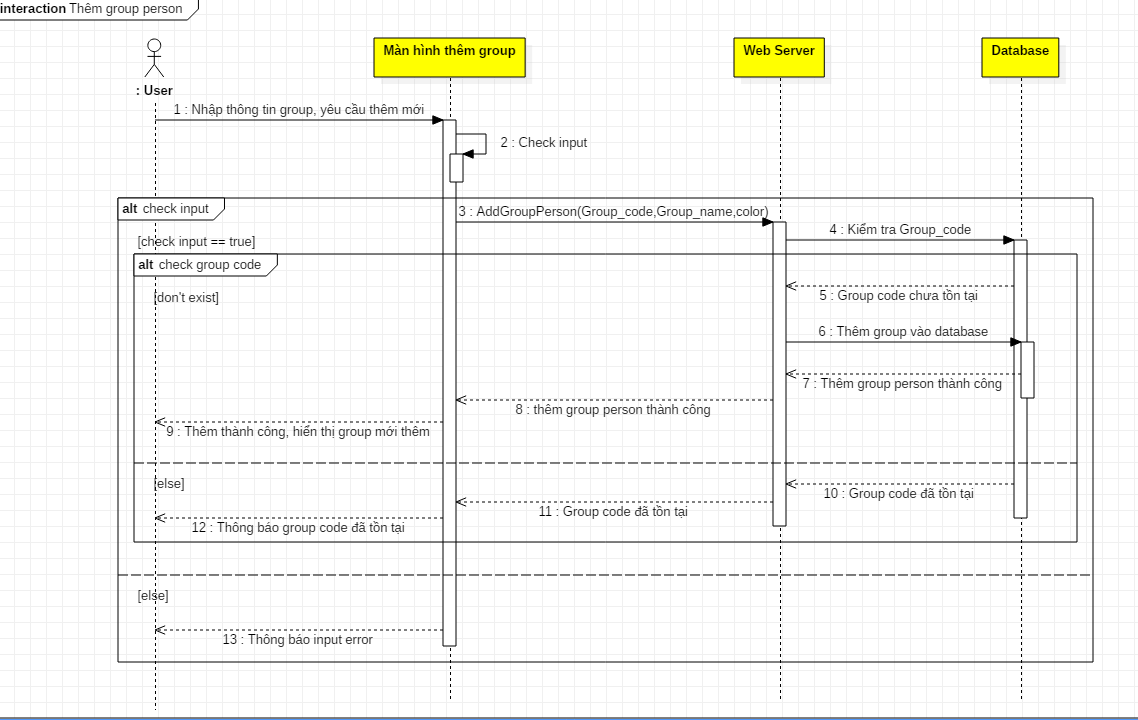
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC005 | |
| **Tên Use Case** | Xóa person. | |
| **Tóm tắt** | Use Case này cho phép người dùng xóa person. | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. Person đã được thêm trước khi thực hiện User case này. | |
| **Kết quả** | Person được xóa, giao diện danh sách person được cập nhật. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện hiển thị danh sách các person, chọn dấu X tại person muốn xóa.  Thông báo hiện hiện lên, chọn Accept để xóa. |  |
|  | Serser lấy thông tin person được gửi đi, sau đó xóa person. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Xóa person***

* + - 1. Thêm group person

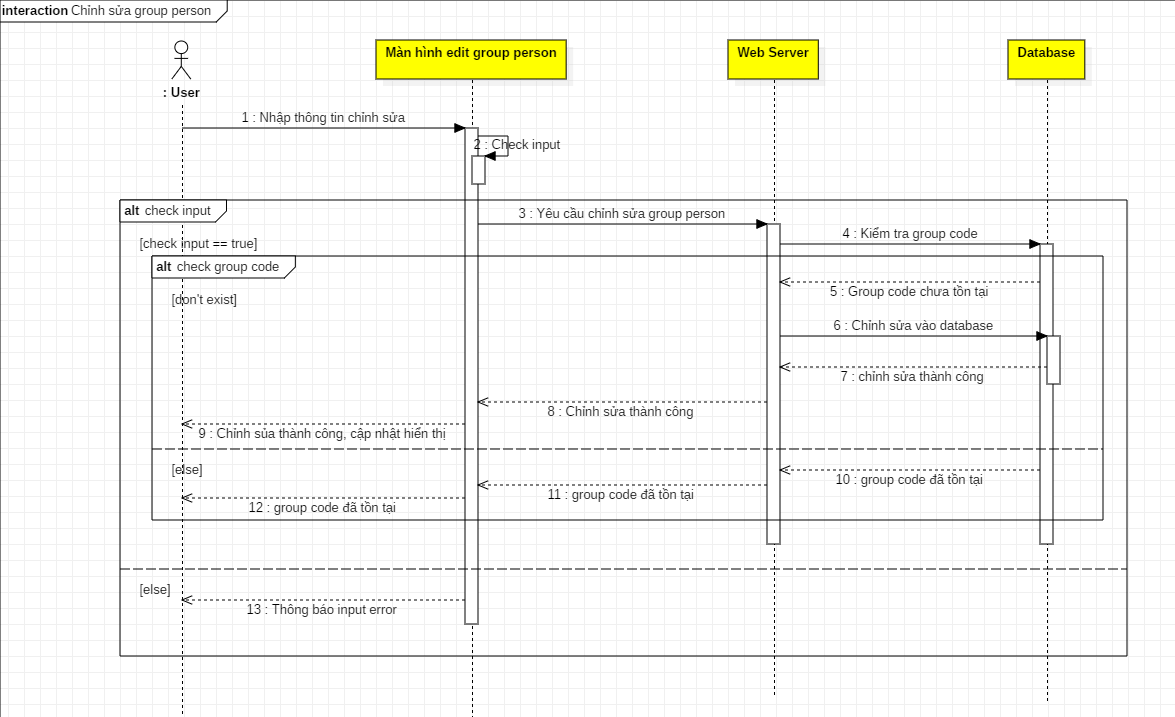
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC007 | |
| **Tên Use Case** | Thêm group person | |
| **Tóm tắt** | Use Case này cho phép người dùng thêm mới một group person | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. | |
| **Kết quả** | Group person được thêm mới và hiển thị trên giao diện danh sách group person. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện thêm group person, điền đầy đủ thông tin và chọn màu cho group.  Nhấn OK để tạo mới group person |  |
|  | Kiểm tra thông tin về mặt vật lý (group person code phải đủ độ dài, không chứa ký tự đặc biệt…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (group person code thêm mới có bị trùng hay không?).  Thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu Group person đã tồn tại thì không thêm vào cơ sở dữ liệu, thông báo lỗi về cho Client. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Thêm group person***

* + - 1. Chỉnh sửa group person.

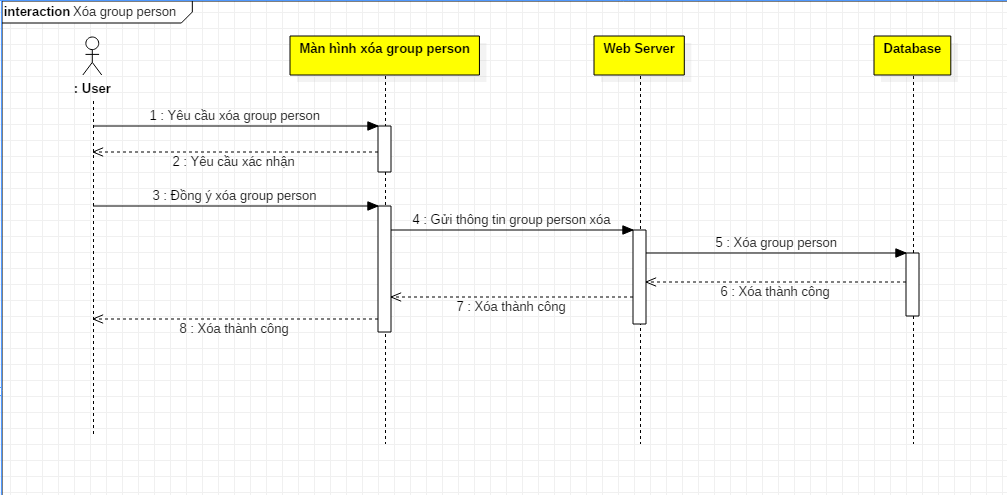
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC008 | |
| **Tên Use Case** | Chỉnh sửa group person | |
| **Tóm tắt** | Use Case này cho phép người dùng chỉnh sửa group person | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. Group person đã được thêm trước đó. | |
| **Kết quả** | Group person được chỉnh sửa và cập nhật. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện chỉnh sửa group person, điền đầy đủ thông tin và màu mới cho group.  Nhấn OK để chỉnh sửa group person. |  |
|  | Kiểm tra thông tin chỉnh sửa về mặt vật lý (group person code phải đủ độ dài, không chứa ký tự đặc biệt…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (group person code chỉnh sửa có bị trùng hay không?).  Cập nhật thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu Group person đã tồn tại thì không chỉnh sửa vào cơ sở dữ liệu, thông báo lỗi về cho Client. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Chỉnh sửa group person***

* + - 1. Xóa group person.

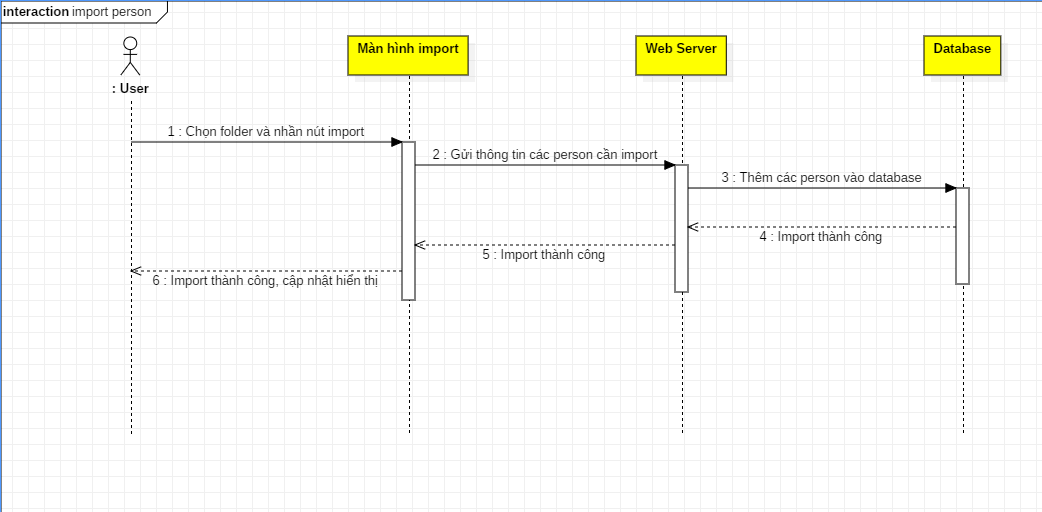
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC009 | |
| **Tên Use Case** | Xóa group person. | |
| **Tóm tắt** | Use Case này cho phép người dùng xóa group person. | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. Group person đã được thêm trước đó. | |
| **Kết quả** | Group Person được xóa, giao diện danh sách person được cập nhật. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện hiển thị danh sách các group person, chuột phải vào group person muốn xóa.  Chọn delete person.  Thông báo hiện hiện lên, chọn Accept để xóa. |  |
|  | Serser lấy thông tin group person được gửi đi, sau đó xóa person. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Xóa group person***

* + - 1. Import person

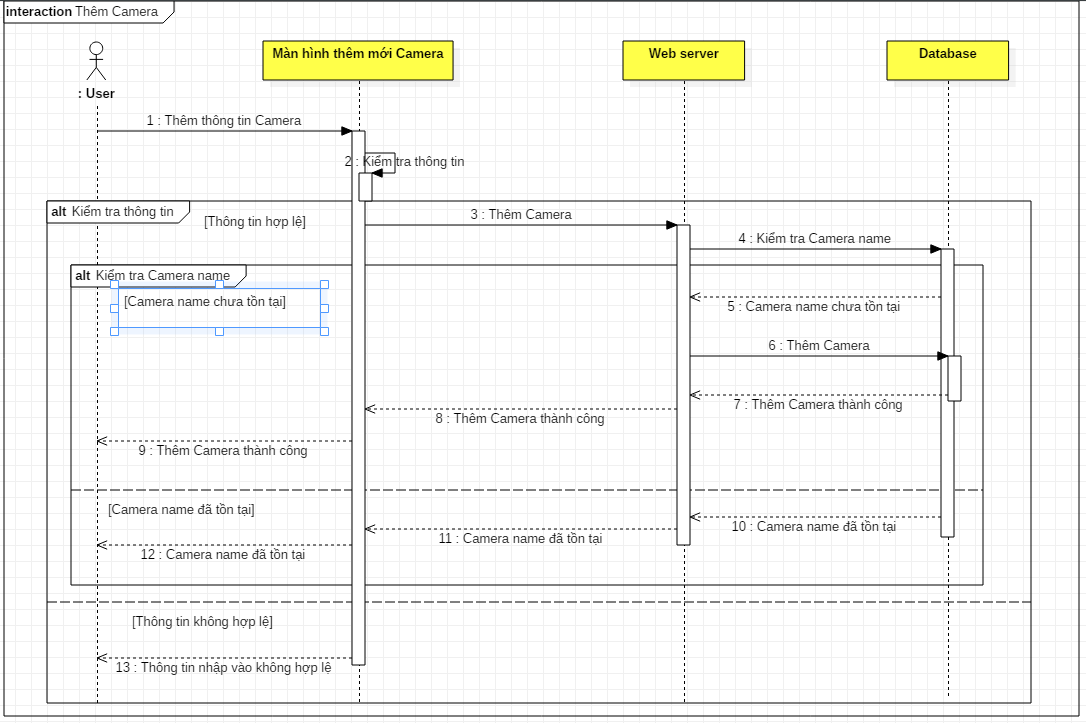
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC0010 | |
| **Tên Use Case** | Thêm nhiều person ( import person) | |
| **Tóm tắt** | Use Case này cho phép người dùng thêm nhiều person cùng lúc | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. Thư mục import đúng định dạng. | |
| **Kết quả** | Các person được thêm vào và danh sách person được cập nhật. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện import person, chọn thử mục theo đúng định dạng.  Chọn submit để gửi danh sách person thêm mới |  |
|  | Serser lấy danh sách person và thêm vào cơ sở dữ liệu. |
| **Kịch bản phụ** | Thử mục không đúng theo định dạng, thông báo lỗi cho người dùng. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Import person***

* + - 1. Thêm Camera:

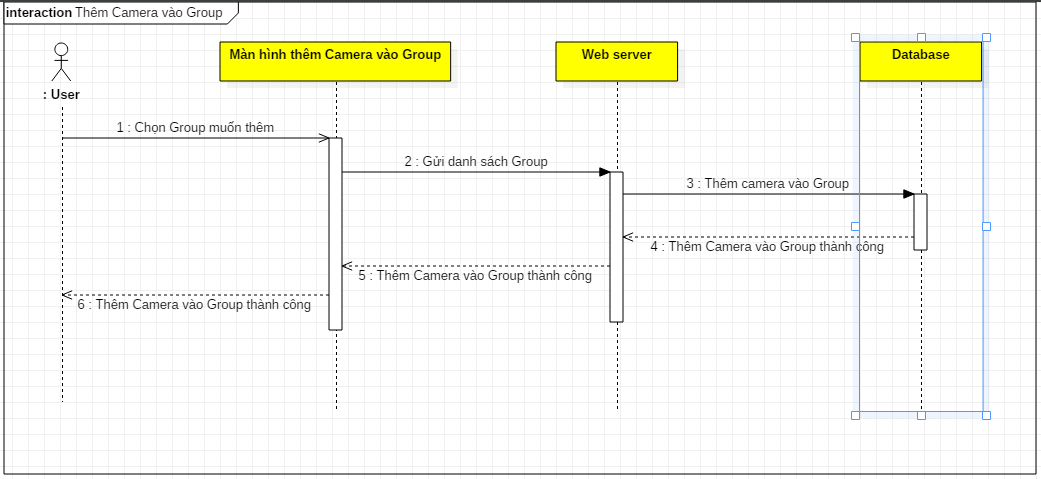
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC002 | |
| **Tên Use Case** | Thêm Camera | |
| **Tóm tắt** | Use case cho phép người dùng thêm camera | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function khi thực hiện Use Case này. | |
| **Kết quả** | Camera được thêm vào danh sách Camera | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Nhấn vào button + ở cuối sanh sách Camera, nhập thông tin Camera name, Stream url,… Bấm nút OK |  |
|  | Kiểm tra thông tin về mặt vật lý (Camera name, Stream url không được bỏ trống,…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Camera name đã tồn tại chưa?...).  Thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu thông tin người dùng đã tồn tại thì không thêm vào cơ sở dữ liệu, thông báo lỗi về cho Client. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Thêm Camera***

* + - 1. Thêm Camera vào Group:

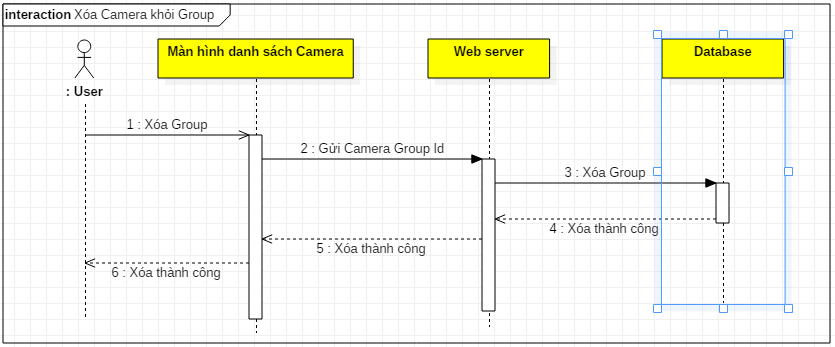
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC003 | |
| **Tên Use Case** | Thêm Camera vào Group | |
| **Tóm tắt** | Use case cho phép người dùng thêm camera vào Group đã có | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function, đã thêm Camera và thêm Group khi thực hiện Use Case này. | |
| **Kết quả** | Camera được thêm vào danh sách Group | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Nhấn vào button + ở cột Group, bấm Chọn Group. Bấm nút OK |  |
|  | Gửi thông tin lên Server. |
|  | Thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng muốn chọn 1 Group không có trong danh sách Group đã tạo trước thì có thể bấm New Group để tạo Group mới như mong muốn | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Thêm Camera vào Group***

* + - 1. Xóa Camera khỏi Group:

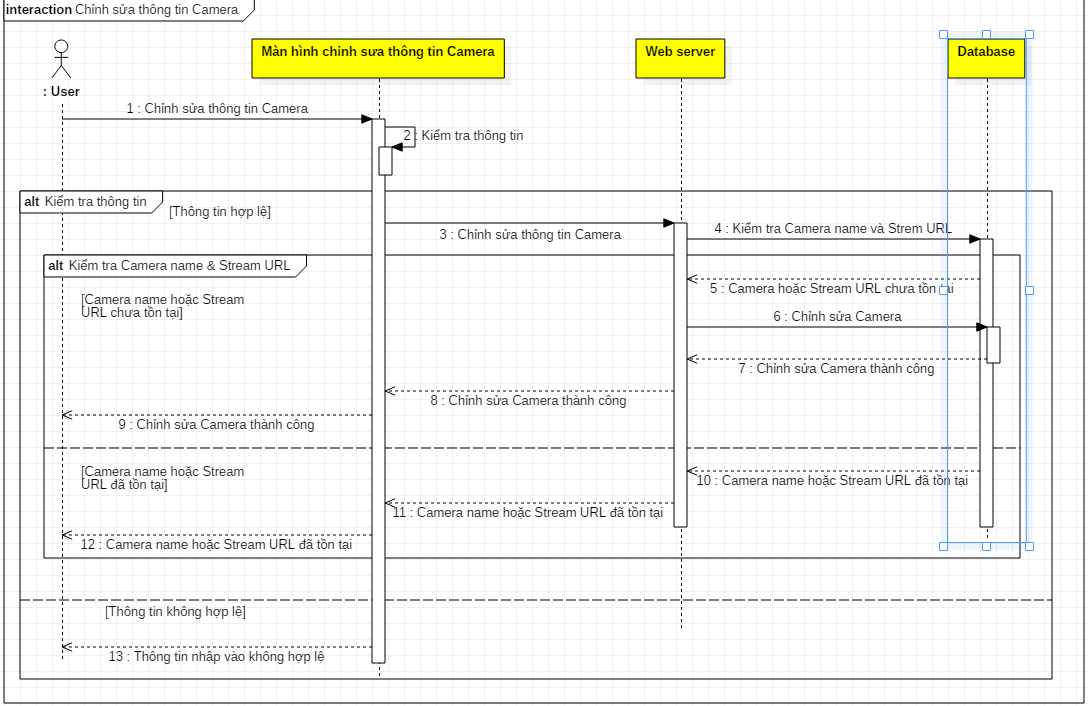
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC004 | |
| **Tên Use Case** | Xóa Camera khỏi Group | |
| **Tóm tắt** | Use case cho phép người xóa Camera khỏi Group đã có | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function, đã thêm Camera và thêm Group khi thực hiện Use Case này. | |
| **Kết quả** | Camera được xóa khỏi danh sách Group | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Ở cột Group bấm vào dấu X vào Group muốn xóa Camera.Bấm xác nhận |  |
|  | Gửi thông tin lên Server. |
|  | Xóa thông tin khỏi cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Xóa Camera khỏi Group***

* + - 1. Chỉnh sửa thông tin Camera:

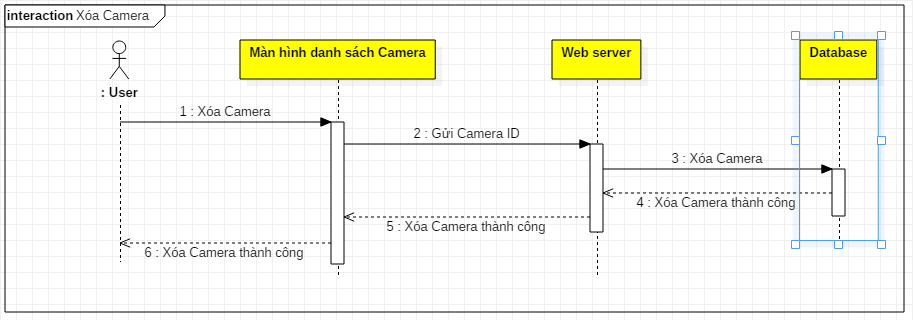
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC005 | |
| **Tên Use Case** | Chỉnh sửa thông tin Camera | |
| **Tóm tắt** | Use case cho phép người dùng chỉnh sửa thông tin Camera đã có | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function, đã thêm Camera khi thực hiện Use Case này. | |
| **Kết quả** | Thông tin Camera được chỉnh sửa thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Nhấn vào biểu tượng Edit ở cột Action. |  |
|  | Kiểm tra thông tin về mặt vật lý (Camera name, Stream url không được bỏ trống,…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Camera name đã tồn tại chưa?...).  Chỉnh sửa thông tin trong cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu thông tin người dùng đã tồn tại thì không thêm vào cơ sở dữ liệu, thông báo lỗi về cho Client. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Chỉnh sửa thông tin Camera***

* + - 1. Xóa Camera:

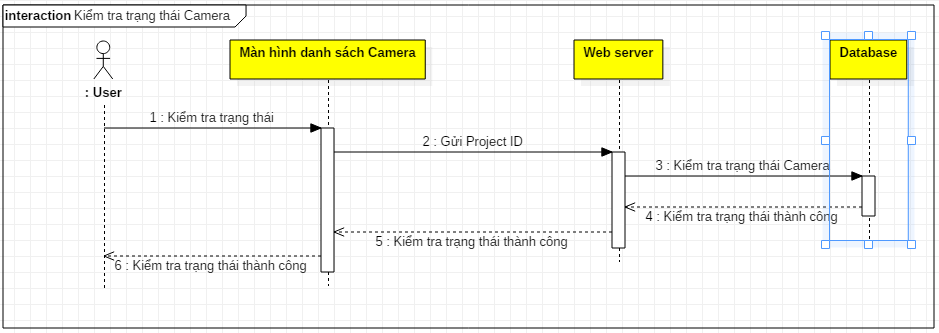
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC006 | |
| **Tên Use Case** | Xóa Camera | |
| **Tóm tắt** | Use case cho phép người dùng xóa Camera đã có | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function, đã thêm Camera khi thực hiện Use Case này. | |
| **Kết quả** | Xóa Camera thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Nhấn vào biểu tượng Delete ở cột Action. Bấm xác nhận |  |
|  | Gửi thông tin lên Server. |
|  | Chỉnh sửa thông tin trong cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Xóa Camera***

* + - 1. Kiểm tra trạng thái Camera:

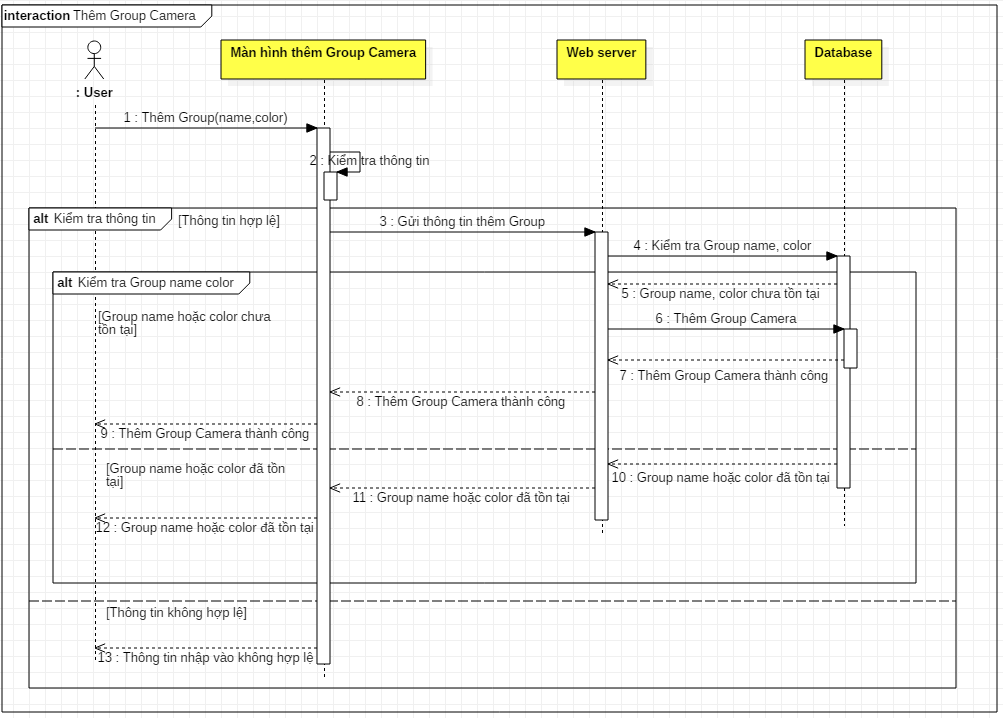
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC007 | |
| **Tên Use Case** | Kiểm tra trạng thái Camera | |
| **Tóm tắt** | Use case cho phép người dùng kiểm tra trạng thái Camera đã có | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function, đã thêm Camera khi thực hiện Use Case này. | |
| **Kết quả** | Kiểm tra trạng thái Camera thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Nhấn vào biểu tượng Delete ở cột Action. Bấm xác nhận |  |
|  | Gửi thông tin lên Server. |
|  | Chỉnh sửa thông tin trong cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Kiểm tra trạng thái Camera***

* + - 1. Thêm Group Camera:

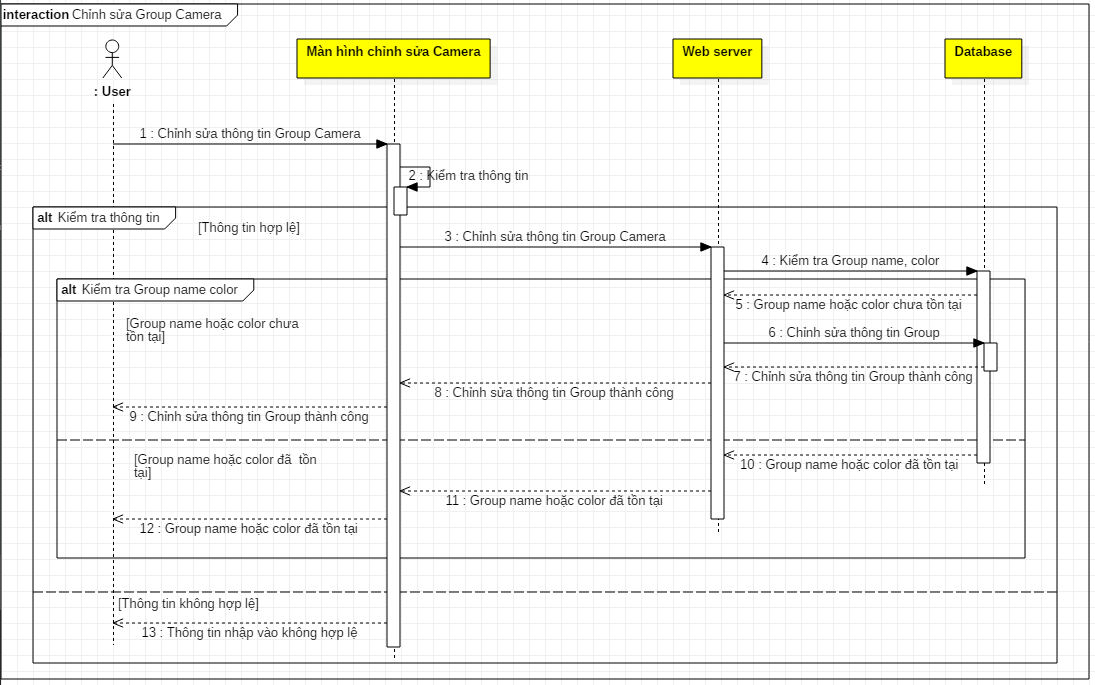
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC007 | |
| **Tên Use Case** | Thêm Group Camera | |
| **Tóm tắt** | Use case cho phép người thêm Group Camera mới | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function khi thực hiện Use Case này. | |
| **Kết quả** | Thêm Group Camera thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Nhấn vào biểu tượng Add ở phần Res Group. Nhập Group name và Group color. Bấm xác nhận |  |
|  | Kiểm tra thông tin về mặt vật lý (Group name, không được bỏ trống,…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Group name đã tồn tại chưa?...).  Thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu thông tin người dùng đã tồn tại thì không thêm vào cơ sở dữ liệu, thông báo lỗi về cho Client. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Thêm Group Camera***

* + - 1. Chỉnh sửa Group Camera:

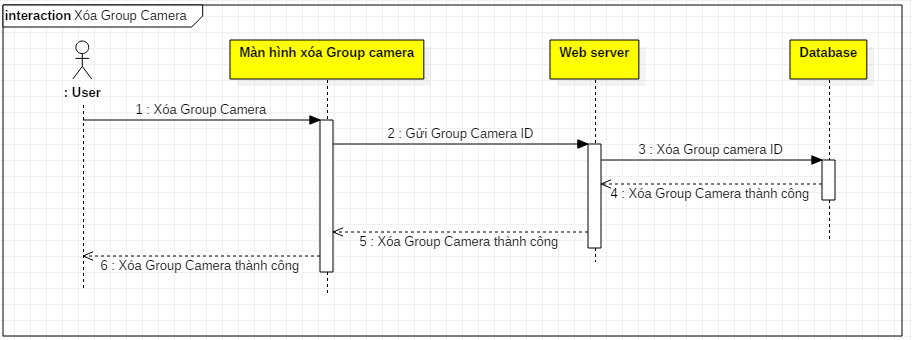
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC007 | |
| **Tên Use Case** | Chỉnh sửa Group Camera | |
| **Tóm tắt** | Use case cho phép người chỉnh sửa Group Camera | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function và đã thêm Group Camera khi thực hiện Use Case này. | |
| **Kết quả** | Chỉnh sửa Group Camera thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Chuột phải vào Group chọn chỉnh sửa Group. Nhập Group name và Group Camera. Bấm xác nhận. |  |
|  |  | Kiểm tra thông tin về mặt vật lý (Group name, không được bỏ trống,…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Group name đã tồn tại chưa?...).  Chỉnh sửa thông tin cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Nếu người dùng nhập thiếu trường dữ liệu hoặc nhập sai định dạng thì thông báo lỗi cho người dùng và yêu cầu người dùng nhập lại.  Nếu thông tin người dùng đã tồn tại thì không thêm vào cơ sở dữ liệu, thông báo lỗi về cho Client. | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Chỉnh sửa Group Camera***

* + - 1. Xóa Group Camera”

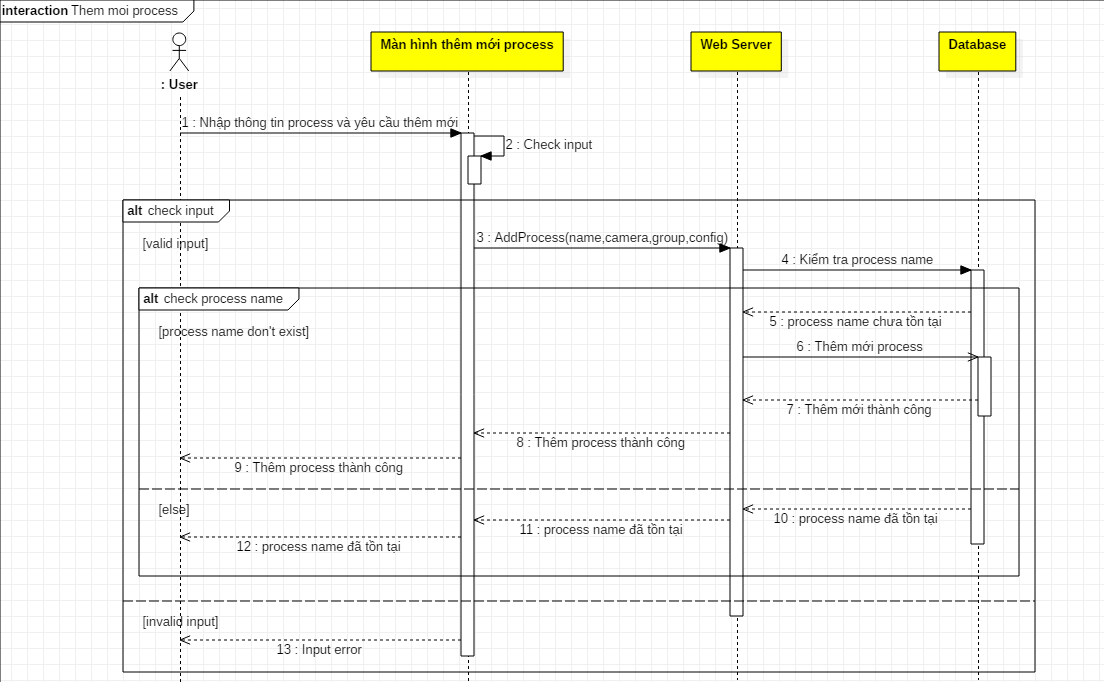
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC006 | |
| **Tên Use Case** | Xóa Group Camera | |
| **Tóm tắt** | Use case cho phép người dùng xóa Group Camera đã có | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function, đã thêmGroup Camera khi thực hiện Use Case này. | |
| **Kết quả** | Xóa Group Camera thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện Dashboard, nhấn vào biểu tượng thanh Menu. Nhấn vào F.Smart Door. Chọn vào Res Camera. Chuột phải vào Group chọn xóa Group. Bấm xác nhận |  |
|  | Gửi thông tin lên Server. |
|  | Chỉnh sửa thông tin trong cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |



***Hình xx Sơ đồ tuần tự - Xóa Group Camera***

Thêm mới process

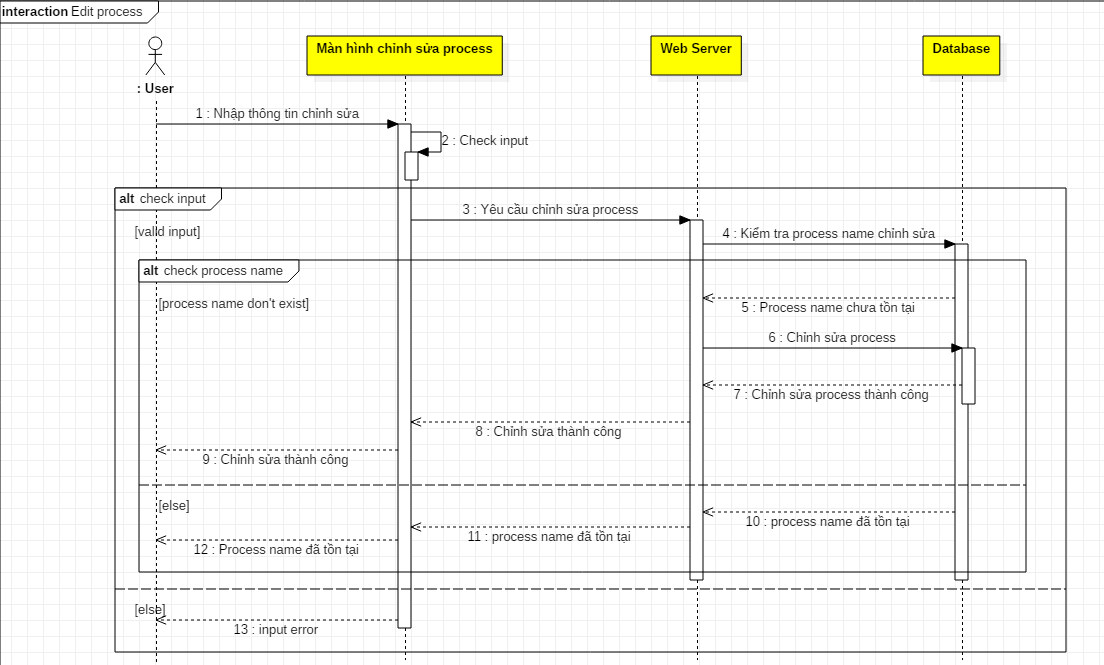
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC | |
| **Tên Use Case** | Thêm mới process | |
| **Tóm tắt** | Use case cho phép người dùng thêm mới process | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function khi thực hiện Use Case này. | |
| **Kết quả** | Thêm process thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tại giao diện thêm mới person, nhập đầy đủ thông tin và click button Ok để thêm process. |  |
|  | Kiểm tra thông tin về mặt vật lý (process name phải đủ độ dài, không chứ ký tự đặc biệt…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Process đã tồn tại hay chưa?).  Thêm thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Sai input hoặc process đã tồn tại, trả về thông báo cho người dùng. | |
| **Ràng buộc phi chức năng** | Không có. | |



Hình xxx.

Chỉnh sửa process

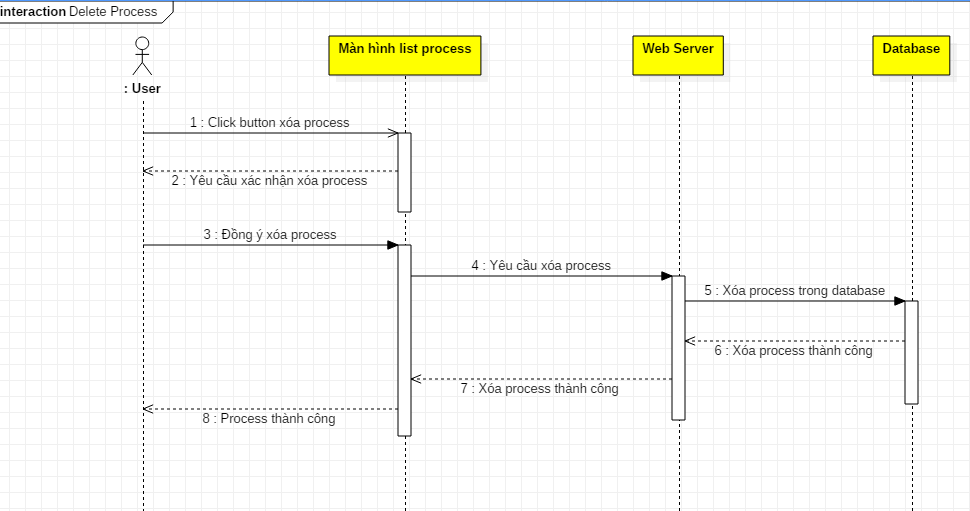
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC | |
| **Tên Use Case** | Chỉnh sửa process | |
| **Tóm tắt** | Use case cho phép người dùng chỉnh sửa process | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project, mua function và đã thêm mới process khi thực hiện Use Case này. | |
| **Kết quả** | Thêm process thành công | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện chỉnh sửa process, điền đầy đủ các thông tin muốn thay đổi.  Nhấp OK để chỉnh sửa person |  |
|  | Kiểm tra thông tin về mặt vật lý (process name phải đủ độ dài, không chứ ký tự đặc biệt…).  Gửi thông tin lên Server. |
|  | Server kiểm tra thông tin về mặt logic (Process name mới đã tồn tại hay chưa?).  Chỉnh sửa thông tin vào cơ sở dữ liệu. Trả kết quả xử lý về cho Client. |
| **Kịch bản phụ** | Sai input hoặc process name mới đã tồn tại, trả về thông báo cho người dùng. | |
| **Ràng buộc phi chức năng** | Không có. | |



Hình xxxx

Xóa process

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC00 | |
| **Tên Use Case** | Xóa process | |
| **Tóm tắt** | Use Case này cho phép người dùng xóa process. | |
| **Tác nhân** | User. | |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập vào hệ thống, tạo project và mua function. Process đã được thêm trước khi thực hiện User case này. | |
| **Kết quả** | Process được xóa, giao diện danh sách process được cập nhật. | |
| **Kịch bản chính** | **Hành động của tác nhân** | **Phản ứng của hệ thống** |
| Tai giao diện hiển thị danh sách các process, chọn button delete tại person muốn xóa.  Thông báo hiện hiện lên, chọn Accept để xóa. |  |
|  | Serser lấy thông tin process được gửi đi, sau đó xóa process. |
| **Kịch bản phụ** | Không có | |
| **Ràng buộc phi chức năng** | Không có. | |

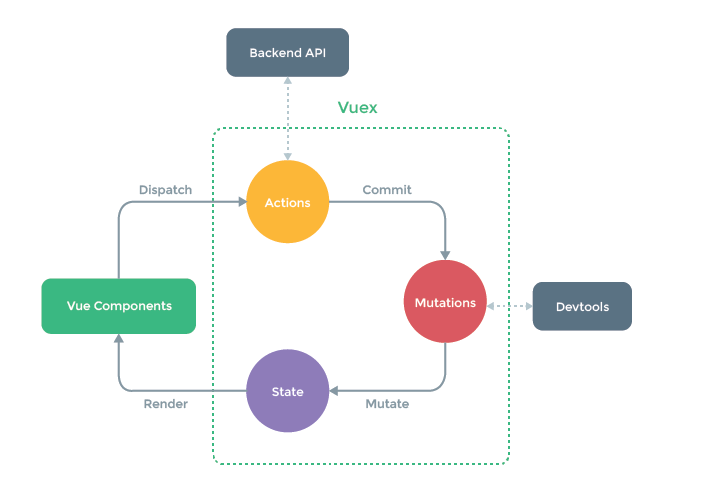


Hình xxx

**Chương 4**

# **Công nghệ và môi trường triển khai**

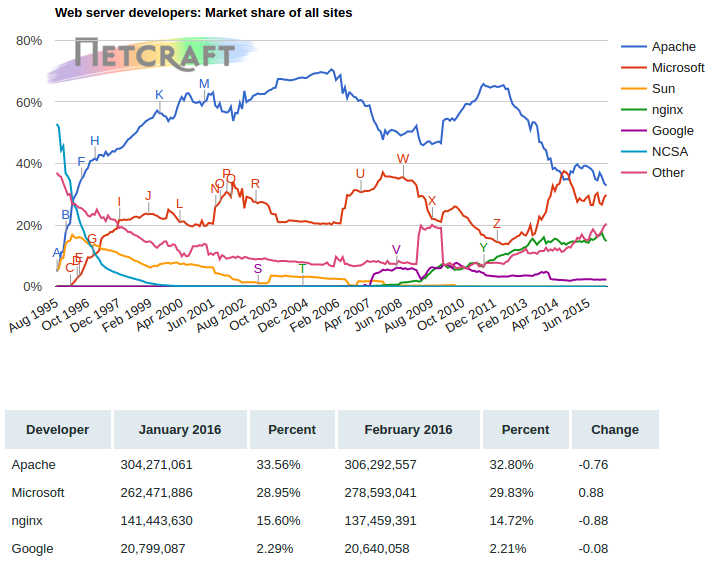
* Nội dung chương này giới thiệu công nghệ và môi trường triển khai cho ứng dụng phía Client và Server (Vuejs, Python, MySQL, NginX, Google Cloud Platform,).
  1. **Môi trường triển khai:**
  2. **Giải pháp công nghệ sử dụng ở phía Client:**
     1. **Giới thiệu Vuex:**
* Vuex là thư viện giúp quản lý trạng thái các component trong Vue.js, nó là nơi lưu trữ tập trung cho tất cả các component trong một ứng dụng, với nguyên tắc trạng thái chỉ có thể được thay đổi theo kiểu có thể dự đoán. Vuex nhìn thấy tại sao không đưa các trạng thái được chia sẻ của các component ra và quản lý chúng trong một bộ máy toàn cục, và đó chính là lý do cho sự ra đời của Vuex. Trong đó, các component trở thành các view và các component có thể truy xuất trạng thái hoặc trigger các hành động. Với cách thức này, mã nguồn có cấu trúc và dễ dàng duy trì.



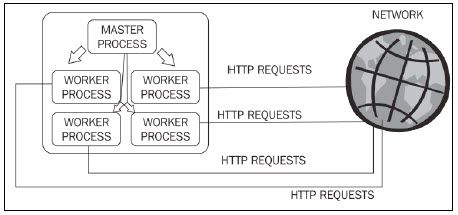
* Vuex hoạt động theo mô hình "Luồng dữ liệu một chiều" với các thành phần sau:
* State: Vuex sử dụng một cây trạng thái duy nhất, đối tượng này sẽ chứa tất các trạng thái của ứng dụng, như vậy bạn chỉ có duy nhất một kho lưu trữ cho mỗi ứng dụng, điều này làm cho việc xác định các trạng thái là dễ dàng và cũng đơn giản trong việc tạo ra các ảnh chụp trạng thái (snapshot) của ứng dụng hiện tại.
* Action: Là nơi chứa các hành động như lấy dữ liệu api từ serve hay một hành động làm thay đổi dữ liệu trong CSDL( thêm, sửa, xóa) và hơn thế nữa action commit các thay đổi state ở mutation. Action có thể chứa các hoạt động không đồng bộ.
* Mutation: Trạng thái không thể thay đổi trực tiếp mà chỉ được thay đổi thông qua commit và mỗi commit này là một hành động được thực hiện tại action. Vuex mutation tương tự như sự kiện, mỗi mutation có một kiểu string và một handel.
* Getters: Đôi khi chúng ta cần lấy các trạng thái dựa vào việc tính toán, lọc bỏ các trạng thái được cung cấp bởi kho lưu trữ.
* Module: Vuex sử dụng cây trạng thái duy nhất, tất cả các trạng thái của ứng dụng được đưa vào một đối tượng, như vậy khi ứng dụng phát triển lên, store có thể phình lên rất nhiều. Vuex cho phép chia nhỏ store thành các module nhỏ hơn, mỗi module cũng có state, mutation, action, getter và thậm chí còn cho phép các module lồng nhau.
* Vuex giúp cho việc quản lý dự án trở nên hiệu quả. Nó cân bằng giữa tốc độ và hiệu năng của dự án. Nếu chưa phân tích kỹ yêu cầu của dự án mà lao ngay vào code vuex, sẽ cảm thấy code vuex khá là dài dòng, phức tạp và đương nhiên không mang lại hiệu quả gì. Khi có ý định code một **SPA** (**Single-Page App**) tầm trung đến lớn, vuex có lẽ là sự lựa chọn hoàn toàn sáng suốt.



* + 1. **Giới thiệu về NginX:**
       1. Tổng quan:
* Nginx là 1 máy chủ reverse proxy mã nguồn mở cho các giao thức HTTP, HTTPS, SMTP, POP3 và IMAP, cũng như là 1 máy chủ cân bằng tải (load balancer), HTTP cache và web. Dự án Nginx được bắt đầu với việc tập trung vào tính đồng thời cao, hiệu năng cao và sử dụng tài nguyên thấp và được phát triển bởi Igor Sysoev vào nằm 2002, được phân phối ra công chúng lần đầu vào nằm 2004.
* Không giống với các máy chủ web truyền thống, Nginx không dựa trên luồn (thread) để xử lý yêu cầu. Thay vào đó, nó sử dụng 1 kiến trúc bất đồng bộ hướng sự kiện linh hoạt . Kiến trúc này sử dụng ít, nhưng quan trọng hợn, là lượng bộ nhớ có thể dự đoán khi hoạt động. Đây chính là điểm mấu chốt khiến Nginx là 1 trong số ít những máy chủ được viết để giải quyết vấn đề C10K
* Nói đến đây một số bạn sẽ tò mò là vấn đề C10K là gì ? Hiểu đơn giản thì do các máy chủ web truyền thống xử lý các yêu cầu dựa trên luồn (thread), tức là mỗi khi máy chủ web nhận được 1 yêu cầu mới, nó sẽ tạo ra 1 luồng mới để xử lý cho yêu cầu này, và cứ thế khi số lượng các yêu cầu gửi đến máy chủ web ngày càng nhiều thì số lượng các luồn xử lý này trong máy chủ sẽ ngày càng tăng. Và điều này dẫn đến việc thiếu hụt tài nguyên cấp cho các luồn xử lý trên
* Hiện nay, có khoảng 14,72 % (hơn 137 triệu) các website trên Internet đang sử dụng Nginx là máy chủ web.



* + - 1. **Kiến trúc của Nginx**
* Khi được khởi chạy service, nginx khởi tạo mọt tiến trình chủ và cũng là tiến trình duy nhất tồn tại trong bộ nhớ Master Process. Tiến trình này không chịu trách nhiệm tự xử lý bất kỳ request nào từ phía client mà thay vào đó nó sinh ra các tiến trình con gọi là Worker Process để xử lý các request này.



* Để định nghĩa cho các Worker Process này, chúng ta cần sử dụng tệp tin cấu hình để xác định số tiến trình, số lượng kết nối , tài khoản và nhóm tài khoản mà mỗi Worker Process chạy
  + 1. **Vấn đề HTTP client – Axios:**
* Có nhiều cách gọi API từ front-end để kết nối và lấy dữ liệu từ một Web Service. Một số thư viện để kết nối với Internet và nhận dữ liệu từ Server như: Fetch, Axios, Request-promise, Superagent,…
* Chính vì có nhiều thư viện hỗ trợ việc liên lạc giữa front-end (Client) với Server như vậy, nhóm chúng tôi sẽ tiến hành tìm hiểu, so sánh để đưa ra lựa chọn thích hợp.
* Về mức độ phổ biến thì chúng tôi sẽ dựa theo đánh giá số sao trên Github tương ứng với từng thư viện: Axios (59608), Fetch (161), Request-promise (3918),… [[1]](#footnote-1)).
* Với mức độ phổ biến ở Việt Nam chúng tôi sẽ tiến hành lập bảng so sánh 2 thư viện là Axios và Fetch
* Bảng so sánh:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí so sánh** | **Axios** | **Fetch** | **Nhận xét** |
| **Thực hiện việc gửi Request** | Cả hai thư viện đều hỗ trợ thực thi Request | |  |
| **Response** | - Tự động trả về kiểu dữ liệu JSON.  - Có thể hủy Request và đặc tính thời gian chờ phản hồi | - Không tự động trả về kiểu dữ liệu JSON.  - Không cung cấp API để đặt thời gian chờ phản hồi.. |  |
| **Cookies** | - Tự động gửi trả Cookies về Server khi tạo 1 request | - Không tự động gửi trả Cookies về Server khi tạo 1 request |  |
| **Cập nhật tiến độ Uploads/ Downloads** | - Axios được xây dựng dựa trên các XHR API cũ hơn, có thể khai báo các hàm callback cho onUploadProgress và onDownloadProgress để hiện thị phần trăm thành công tại giao diện cho app | - Không hỗ trợ |  |
| **HTTP error code** | - Axios sẽ reject tất cả các promise của request nếu một trong các mã lỗi trên được trả về | - Khi server trả về các mã lỗi 4xx hay 5xx, thì hàm catch() sẽ không được gọi đến và người lập trình viên phải tự kiểm tra trạng thái của mã trả về để xác định xem liệu request đó có thành công hay không |  |
| **Browser** | - Hỗ trợ tất cả browser hiện có như Firefox, Chrome, Edge, IE, Safari, Opera,… | - Hỗ trợ một số trình duyệt phổ biến hiện nay mà thôi |  |

* **Kết luận:**
* Cả hai thư viện trên đều đáp ứng đủ các nhu cầu cho dự án hiện tại của nhóm tôi. Tuy nhiên, theo khảo sát thì tài liệu của Axios đầy đủ hơn so với Fetch, cú pháp nhìn rõ ràng hơn và có nhiều lựa chọn để tùy chỉnh (customize), cộng đồng người sử dụng cũng đông hơn nên nhóm tôi quyết định chọn Axios làm thư viện thực hiện gửi nhận dữ liệu giữa Client và Server để có thể có được sự hỗ trợ tốt nhất.
  1. **Giải pháp công nghệ sử dụng ở phía server**
     1. **Ngôn ngữ Python**

Python là một ngôn ngữ đa nền tảng có thể thực thi trên nhiều hệ điều hành khác nhau : Unix, MS-DOS ,Windows,Mac OS, Linux… Bạn có thể sử dụng nó trong các lĩnh vực như phát triển phần mềm, phát triển web, thiết kế web,v.v… Ngoài ra , Python còn có các đặc điểm nổi bật như :

***+ Đơn giản*** : Cú pháp đơn giản giúp cho người lập trình dễ dàng đọc và tìm hiểu.

+ Tốc độ**: Python** có tốc độ xử lý nhanh hơn so với ngôn ngữ PHP

***+ Tương tác*** : Chế độ tương tác cho phép người lập trình thử nghiệm tương tác sửa lỗi của các đoạn mã.

***+ Chất lượng*** :Thư viện có tiêu chuẩn cao, **Python** có khối cơ sở dữ liệu khá lớn nhằm cung cấp giao diện cho tất cả các CSDL thương mại lớn.

+ Thuận tiện**: Python** được biên dịch và chạy trên tất cả các nền tảng lớn hiện nay.

***+ Mở rộng***: Với tính năng này, Python cho phép người lập trình có thể thêm hoặc tùy chỉnh các công cụ nhằm tối đa hiệu quả có thể đạt được trong công việc.

***+ GUI Programming***: Giúp cho việc thực hiện ảnh minh hoạ di động một cách tự nhiên và sống động.

* *Kết luận: Với những tính năng nổi bật nói trên cộng với rất nhiều những lợi thế khác, chúng ta có thể thấy Python dường như có thể xử lý mọi vấn đề liên quan lập trình cho chúng ta – và xây dựng một ứng dụng là web service cũng không phải là ngoại lệ.*
  + 1. **Framework Django**

Django là một web framework miễn phí mã nguồn mở được viết bằng Python. Django sử dụng mô hình Model-View-Control (MVC). Mục tiêu chính của Django là đơn giản hóa việc tạo các website phức tạp có sử dụng cơ sở dữ liệu. Django tập trung vào tính năng “có thể tái sử dụng” và “có thể tự chạy” của các component, tính năng phát triển nhanh, không làm lại những gì đã làm. Một số website phổ biến được xây dựng từ Django là Pinterest, Instagram, Mozilla, và Bitbucket. Django có các ưu điểm nổi bật như :

* Có đầy đủ các thư viện/module cần thiết: Django có sẵn các thư viện về user authentication, content admin, site maps, RSS feed v.v...
* Đảm bảo về tính bảo mật: Không còn các nỗi lo về các lỗi bảo mật thông thường như SQL Injection, cross-site scripting, cross-site request forgery hay [clickjacking](https://junookyo.blogspot.com/2013/11/clickjacking-qua-khu-hien-tai-va-tuong-lai.html) nữa. Django cũng cung cấp cả phương pháp để lưu mật khẩu an toàn nữa!
* Khả năng mở rộng tốt: Django có thể đáp ứng lượng traffic lớn, nghĩa là bạn không cần phải lo lắng về khả năng scale sản phẩm của mình nữa.
* Ngoài ra Django còn hỗ trợ ORM , dễ dàng chuyển đổi qua lại giữa các database engine một cách dễ dàng

*Sơ đồ tổng quan cách áp dụng Django để xây dựng Fcloud Console:*

**Browser**

DJANGO

**URL Routing**

**Controller/ Middleware**

**Model/ Django ORM**

**FUSER Database**

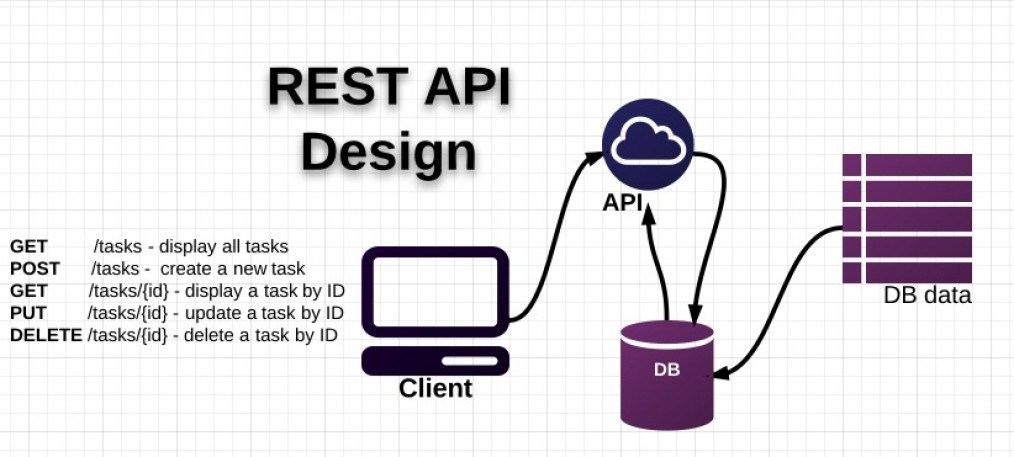
**FAI Database**

HTTP Request

JSON Response

* *Kết luận: Chính vì những đặc điểm nổi bật trên của Django nên nhóm quyết định xây dựng web service trên framework này.*
  + 1. **RESTful API web service:**

1. **Giới thiệu về Rest API**



**REST** (**RE**presentational **S**tate **T**ransfer) là một dạng chuyển đổi cấu trúc dữ liệu, là một phong cách kiến ​​trúc cho việc thiết kế các ứng dụng có kết nối. Nó sử dụng HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu

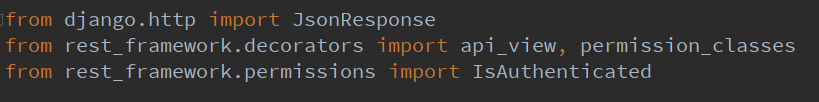
**API** (**A**pplication **P**rogramming **I**nterface) là giao diện lập trình ứng dụng giúp tạo ra các phương thức kết nối với các thư viện và ứng dụng khác nhau.

**REST API** là một ứng dụng chuyển đổi cấu trúc dữ liệu có các phương thức để kết nối với các thư viện và ứng dụng khác. REST API không được xem là một công nghệ, nó là một giải pháp để tạo ra các ứng dụng web services thay thế cho các kiểu khác như SOAP, WSDL (Web Service Definition Language),…

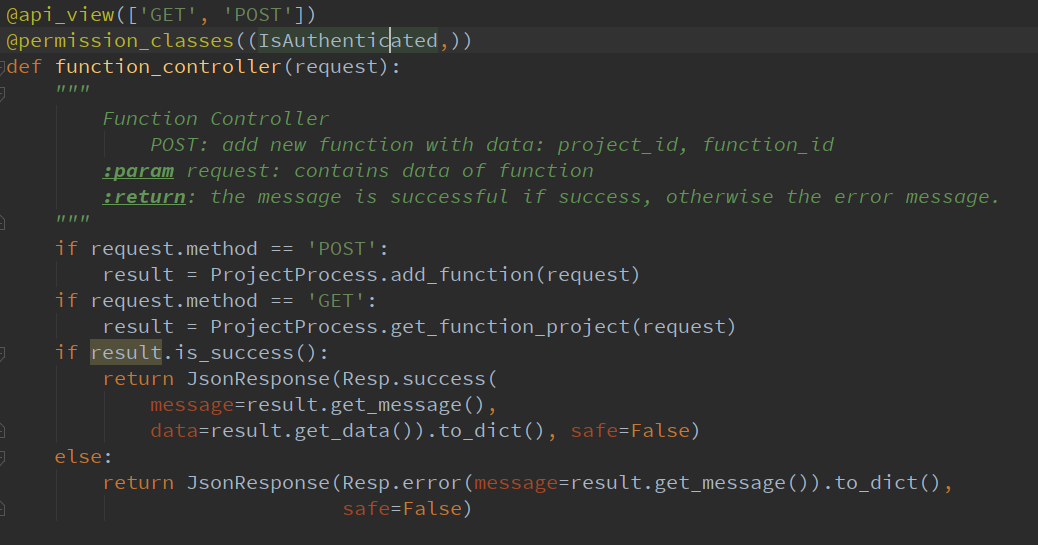
1. **Áp dụng RestAPI vào Fcloud Console**

- Framework Django hỗ trợ thư viện Django REST Framework để giúp xây dựng một hệ thống Rest API hoàn chỉnh.

- Để xây dựng được 1 giao thức HTTP , đầu tiên chúng ta cần phải import tất cả các thư viện cần thiết vào :



- Xây dựng một Controller trong RestAPI gồm :



@api\_view(['GET', 'POST'])

-> Nhận tất cả các phương thức HTTP mà server phải phản hồi

@permission\_classes((IsAuthenticated,))

-> Khai báo class dùng để xác thực

if request.method == 'POST':

-> Điều kiện của phương thức HTTP

result = ProjectProcess.add\_function(request)

-> Gọi hàm xử lý request

if result.is\_success():  
 return JsonResponse(Resp.success(  
 message=result.get\_message(),  
 data=result.get\_data()).to\_dict(), safe=False)  
else:  
 return JsonResponse(Resp.error(message=result.get\_message()).to\_dict(),  
 safe=False)

-> Trả về kết quả Json cho client

1. Kết quả và đánh giá kết quả xây dựng RestAPI

- Cài đặt được các API phục vụ đầy đủ các nhu cầu request từ Client.

- Xây dựng được hệ thống giao tiếp Client – Server.

- Hiểu được cách làm việc của JWT. Tự cài đặt được chức năng tạo token để cung cấp cho Client, cũng như xác thực được tính hợp lệ của request từ Client thông qua token này.

- Đánh giá mức độ hoàn thành: 100%

* + 1. **Giới thiệu hệ quản trị CSDL MySQL**

MySQL là hệ quản trị cơ sở dữ liệu quan hệ sử dụng ngôn ngữ truy vấn có cấu trúc (SQL). MySQL là có tốc độ cao, ổn định và dễ sử dụng, có tính khả chuyển, hoạt động trên nhiều hệ điều hành cung cấp một hệ thống lớn các hàm tiện ích rất mạnh. MySQL có nhiều phiên bản cho các hệ điều hành khác nhau: Windows, Linux, Mac OS X, Unix, FreeBSD, NetBSD, Novell NetWare, SGI Irix, Solaris, SunOS,…

* + 1. **Giao tiếp và làm việc với MySQL**

#### - Vai trò của module:

Đối tượng model của API Web Services sẽ là thành phần giao tiếp giữa Web Services và cơ sở dữ liệu. Django và MySQL sẽ giao tiếp với nhau thông qua Django ORM.

#### - Chi phí về thời gian

Django ORM cung cấp khả năng xử lý SQL khá nhanh với những hệ thống vừa và nhỏ

* + - Đánh giá mức độ hoàn thành: 100%
    1. Mô hình Authentication JWT:

5. Kiểm tra JWT token. Giải mã thông tin user từ JWT token

6. Gửi response về cho người dùng

1. Gửi JWT token lên Authorization Header

3. Trả về JWT token vừa khởi tạo cho Browser

1. POST user/login với username và password

2. Tạo 1 JWT token với Secret key

Server

Browser

**Chương 5**

**Cài đặt minh họa ứng dụng**

Link tham khảo:

<https://movan.vn/rest-api-gioi-thieu-rest-api/>



**Chương 6**

**Kết luận và hướng phát triển**

* Nội dung chương này trình bày những kết luận, những điều đã thực hiện và chưa thực hiện được. Những điều cần cải thiện cũng như hướng phát triển cho ứng dụng trong tương lai.

1. **Kết luận**

Về cơ bản, các chức năng do phía công ty yêu cầu đều đã hoàn thành. Ngoài ra chúng tôi cũng tìm hiểu thực hiện chức năng như quản lý Fire Detect, quản lý People Couting. Các chức năng này hiện vẫn chưa hoàn thành nhưng đã xây dựng được tiền đề để tiếp tục hoàn thiện trong tương lai.

1. ***Những điều đã thực hiện được***

Sau quá trình hoàn thành đề tài, nhóm chúng tôi đã tổng hợp lại những gì đã thực hiện được như sau:

* **Về kĩ thuật:**
  + - Xây dựng đầy đủ các chức năng công ty yêu cầu.

|  |  |  |
| --- | --- | --- |
| STT | Chức năng | Hoàn thành |
| 1 | Quản lý tài khoản người dùng | ✓ |
| 2 | Quản lý person( thêm, xóa, chỉnh sửa) | ✓ |
| 3 | Quản lý group person(thêm, xóa, chỉnh sửa) | ✓ |
| 4 | Quản lý camera(thêm, xóa, chỉnh sửa) | ✓ |
| 5 | Quản lý group camera(thêm, xóa, chỉnh sửa) | ✓ |
| 6 | Quản lý process(thêm, xóa, chỉnh sửa) | ✓ |

* **Về kỹ năng và giao tiếp nhóm:**
  + - Áp dụng quy trình thác nước vào phát triển dự án phần mềm.
    - Sử dụng các công cụ hỗ trợ làm việc và giao tiếp với nhóm (Tortoise SVN).
    - Nâng cao kỹ năng tìm hiểu công nghệ (vuejs, django framework).
    - Nâng cao kỹ năng làm việc theo nhóm

1. ***Những điều cần cải thiện***

Quá trình thực hiện phát triển phần mềm, ứng dụng luôn không thể tránh khỏi những thiếu sót, những lỗi lầm, những khó khăn, vv… Vì thế nhóm chúng tôi đã tổng hợp lại những điều cần cải thiện như sau:

* **Về kỹ thuật:**

|  |  |  |
| --- | --- | --- |
| STT | Chức năng | Hoàn thành |
| 1 | Thanh toán trưc tuyến | 50% |
| 2 | Quản lý trạng thái camera | 70% |

* **Về kỹ năng và giao tiếp nhóm:**
  + - Điều quan trọng nhất là phải luôn có tinh thần trách nhiệm với mục tiêu của nhóm và các thành viên trong nhóm, không vì mục đích, lợi ích của cá nhân mà làm ảnh hưởng đến tập thể, ảnh hưởng đến nhóm phát triển.
    - Cải thiện hơn về khả năng tư duy logic, tìm hiểu công nghệ, kĩ thuật để làm ứng dụng hoạt động tốt hơn, áp dụng ứng dụng vào thực tế cuộc sống tốt hơn.
    - Tập trung vào phát triển ứng dụng phục vụ nhu cầu, lợi ích, giải quyết các vấn đề người dùng hơn. Tránh tập trung quá nhiều vào công nghệ và kĩ thuật dẫn đến làm giảm đi tính áp dụng thực tiễn và thời gian phát triển ứng dụng.

1. **Hướng phát triển**

Về hướng phát triển, trước tiền nhóm sẽ hoàn thiện và cải thiện các chức năng hiện có. Từ đó sẽ phát triển thêm các chức năng như quản lý Fire Detect, quản lý People Counting.

Hệ thống sẽ có thêm các chức năng như thanh toán sản phẩm online, hỗ trợ người dùng.

Khi hoàn thiện các chức năng hệ thống sẽ được deploy và sử dụng trong thực tế.

**-- Hết --**

TÀI LIỆU THAM KHẢO

1. Số liệu được lấy vào ngày 22-05-2019 [↑](#footnote-ref-1)