TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Tài liệu đặc tả yêu cầu phần mềm

(Software Requirement Specification – SRS)

Phiên bản 1

Hệ thống quản lý yêu cầu của nhân viên trong doanh nghiệp

Môn: Phát triển phần mềm theo chuẩn ITSS

Mã HP: IT4945

Mã lớp: 126833

Nhóm 3

Trần Công Hoan - 20184105

Lê Ngọc Long - 20184137

Mai Thế Phong - 20184172

Đoàn Anh Tuấn - 20184218

*Hà Nội, ngày … tháng … năm …*

[**1.Bối cảnh**](#_heading=h.30j0zll) **5**

[**2.Tổng quan về dự án**](#_heading=h.1fob9te) **5**

[**3. Yêu cầu về chức năng**](#_heading=h.3znysh7) **5**

[Các chức năng phía CBNV:](#_heading=h.2et92p0) 5

[Các chức năng phía Admin (thành viên phòng HCNS):](#_heading=h.tyjcwt) 5

[Các chức năng phía Quản lý bộ phận:](#_heading=h.3dy6vkm) 6

[Other:](#_heading=h.1t3h5sf) 6

[**1. Xây dựng biểu đồ Usecase**](#_heading=h.2s8eyo1) **7**

[1.1 Sơ đồ Usecase tổng quan](#_heading=h.17dp8vu) 7

[1.2 Sơ đồ Use Case phân rã](#_heading=h.3rdcrjn) 8

[1.2.1 Phân nhóm chức năng thêm sửa xóa request](#_heading=h.26in1rg) 8

[1.2.2 Phân nhóm chức năng thêm sửa xóa department](#_heading=h.lnxbz9) 9

[1.2.3 Phân nhóm chức năng thêm sửa xóa category](#_heading=h.35nkun2) 10

[1.2.4 Phân nhóm chức năng thêm sửa xóa user](#_heading=h.1ksv4uv) 11

[**2. Đặc tả kịch bản sử dụng cho mỗi Usecase**](#_heading=h.z337ya) **11**

[2.1 Đăng nhập khi đã có tài khoản](#_heading=h.4i7ojhp) 12

[2.2 Đăng nhập bằng gmail](#_heading=h.2xcytpi) 13

[2.3 Tạo yêu cầu](#_heading=h.1ci93xb) 15

[2.4 Sửa yêu cầu](#_heading=h.2bn6wsx) 16

[2.5 Xóa yêu cầu](#_heading=h.qsh70q) 18

[2.6 Chấp nhận yêu cầu](#_heading=h.1pxezwc) 19

[2.7 Từ chối yêu cầu](#_heading=h.49x2ik5) 20

[2.8 Tạo người dùng mới](#_heading=h.2p2csry) 21

[2.9 Sửa người dùng](#_heading=h.3o7alnk) 22

[2.10 Xóa người dùng](#_heading=h.ihv636) 24

[2.11 Tạo department](#_heading=h.1hmsyys) 25

[2.12 Tạo category](#_heading=h.2grqrue) 26

[2.13 Bình luận](#_heading=h.3fwokq0) 28

[2.14 Xử lý request](#_heading=h.4f1mdlm) 29

[**1.Chức năng login**](#_heading=h.3tbugp1) **31**

[**2. Chức năng tạo user**](#_heading=h.28h4qwu) **32**

[**3. Chức năng sửa tài khoản user**](#_heading=h.nmf14n) **33**

[**4. Chức năng tạo request**](#_heading=h.37m2jsg) **34**

[**5. Chức năng sửa request**](#_heading=h.1mrcu09) **34**

[**6. Chức năng tạo category**](#_heading=h.46r0co2) **35**

[**7. Chức năng sửa category**](#_heading=h.2lwamvv) **36**

[**8. Chức năng tạo department**](#_heading=h.9s76zwrbyuha) **37**

[**4.1 Thiết kế kiến trúc**](#_heading=h.mjv7ptuaqq4t) **37**

[**4.1.1 Kiến trúc mô hình MVC**](#_heading=h.67bken5f9ip8) **37**

[**4.1.2 Design Pattern**](#_heading=h.2akx8tg0kdpt) **38**

[**4.1.3 Biểu đồ gói UML**](#_heading=h.y4c4rfw0ao8q) **39**

[**4.2 Thiết kế cơ sở dữ liệu**](#_heading=h.k9n1aa93ylb2) **40**

[**5.1 Thư viện và công cụ sử dụng**](#_heading=h.s56sc09u7u3q) **41**

[**5.1.1 Docker**](#_heading=h.d0hpl72feamk) **41**

[**0. Làm quen với Docker**](#_heading=h.lbfqxnx6ujsv) **41**

[**1. Tổng quan về Docker**](#_heading=h.u5vs1yjmxwn2) **41**

[**2. Lược sử của Docker**](#_heading=h.1mnahfcdhnjl) **42**

[**3. Kiến trúc của hệ thống**](#_heading=h.h78v61l9z6ty) **45**

[**4.1 Daemon Docker**](#_heading=h.8h1q05cgazun) **45**

[**4.2 Docker client**](#_heading=h.fc3qlf1lkpxg) **45**

[**4.3 Docker registries**](#_heading=h.os3r26pwu06z) **46**

[**4.4 Các đối tượng Docker**](#_heading=h.dc2lflpjl46) **46**

[**4.4.1 Images**](#_heading=h.g5bxq09oy4p9) **46**

[**4.4.2 Containers**](#_heading=h.7ysme9z0524a) **46**

[**4.4.3 Services**](#_heading=h.pkahw42n484s) **47**

[**5. Sử dụng docker trong dự án**](#_heading=h.x8xzy0eo3kwc) **47**

[**5.2 Hướng dẫn cài đặt và sử dụng chương trình**](#_heading=h.lolxxi2jxgmh) **49**

[**Kiến thức thu được**](#_heading=h.g590j41pk4ol) **50**

[**Hướng phát triển**](#_heading=h.m5m16xl5s9qj) **50**

Phân chia công việc

Lê Ngọc Long - Các chức năng liên quan đến Admin (tạo user, tạo department, tạo category...), vẽ biểu đồ trình tự liên quan đến các chức năng của Admin

Trần Công Hoan - Các chức năng liên quan đến User (tạo request, đổi password, đăng nhập) vẽ biểu đồ trình tự liên quan đến các chức năng của User

Đoàn Anh Tuấn - Giao diện người dùng của Employee,thiết kế cơ sở dữ liệu, tạo database.

Mai Thế Phong - Giao diện người dùng của Admin, gửi mail cho người dùng, cấu hình docker, thiết kế API

Phân chia công việc review

Mai Thế Phong reivew các chức năng của Admin của Lê Ngọc Long

Đoàn Anh Tuấn reivew các chức năng của User của Trần Công Hoan

Lê Ngọc Long và Trần Công Hoan kiểm tra sản phẩm khi đã hoàn thành.

Chương 1: Khảo sát bài toán

# 1.Bối cảnh

Hiện tại trong công ty khi có vấn đề từ lỗi kỹ thuật đến những vấn đề về dịch vụ xung quanh công ty thì đều phải thực hiện nhiệm vụ là lên từng phòng ban để báo cáo và sau đó họ sẽ xuống để sửa chữa. Tuy nhiên, nhân viên của từng phòng ban có hạn và nhiều khi họ không có trong văn phòng nên việc liên lạc sẽ rất phức tạp. Chính vì thế, website này sẽ giúp các nhân viên có thể gửi yêu cầu một cách trực tuyến, việc này sẽ giúp nhân viên trong công ty có thể tiết kiệm được thời gian cũng như cách để bố trí công việc để giải quyết một cách nhanh nhất.

# 2.Tổng quan về dự án

Request gate là cổng quản lý request của toàn bộ CBNV công ty.

Đối tượng sử dụng:

* CBNV (Employee).
* Admin (thành viên phòng HCNS).
* Quản lý bộ phận (Manager).

# 3. Yêu cầu về chức năng

Yêu cầu về chức năng:

Các chức năng chung:

* Đăng nhập.

#### Các chức năng phía CBNV:

* Tạo mới request:
  + Chọn category.
  + Chọn due date.
  + Chọn người chịu trách nhiệm.
* Sửa request đã tạo.
* Xoá request đã tạo.
* Theo dõi danh sách request đã tạo.
* Trong trường hợp HCNS đã tiếp nhận thì không thể chỉnh sửa hay xóa.

#### Các chức năng phía Admin (thành viên phòng HCNS):

* Nhóm chức năng request:
  + Xem danh sách request được gửi đến.
  + Cập nhật trạng thái của request.
  + Thay đổi trạng thái của request
  + Đánh độ ưu tiên cho request
* Nhóm chức năng quản lý user:
  + Tạo user
  + Thay đổi role của user.
  + Chỉnh sửa thông tin user.
* Nhóm chức năng quản lý request:
  + Thêm/sửa request category.
  + Enable/disable trạng thái của category.

#### Các chức năng phía Quản lý bộ phận:

* Xem danh sách request từ CBNV bộ phận mình quản lý.
* Approve hoặc reject request từ CBNV bộ phận mình quản lý.

#### Other:

* Request sẽ có cập nhật thay đổi. Admin có thể tiếp nhận request khi Quản lý bộ phận chưa approve.
* Quản lý bộ phận reject request sẽ chuyển thành Close.
* Các trạng thái của request: Open/In progress/Close.

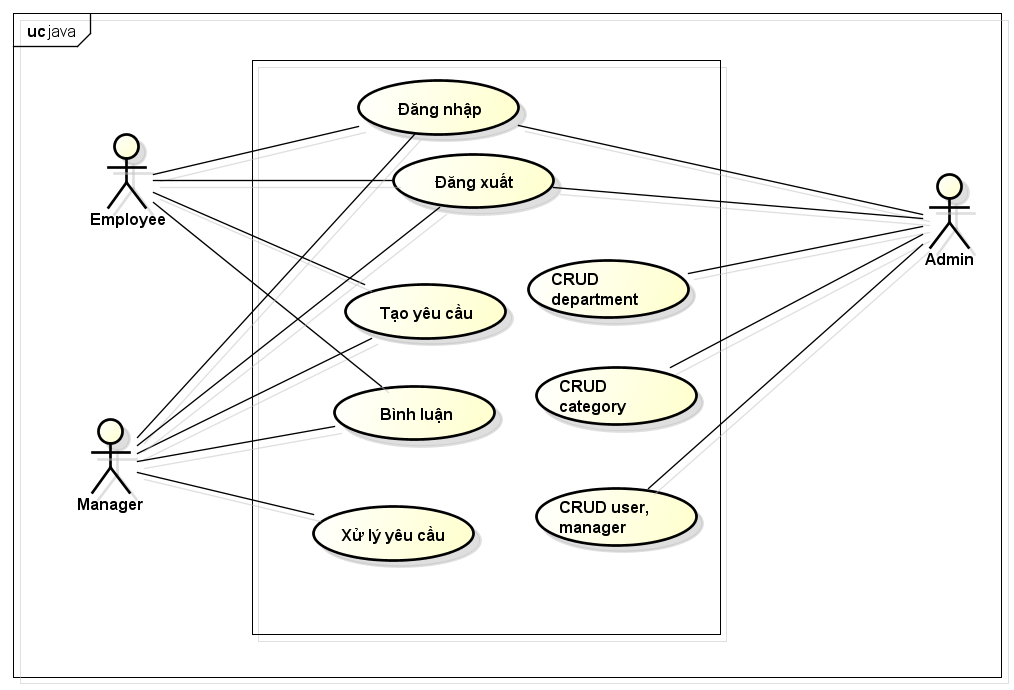
- Xác định các tác nhân (đối tượng người dùng, các hệ thống khác có tương tác,...) cho hệ thống website

- Xác định các chức năng cho mỗi đối tượng người dùng

Chương 2: Đặc tả yêu cầu bài toán

# 1. Xây dựng biểu đồ Usecase

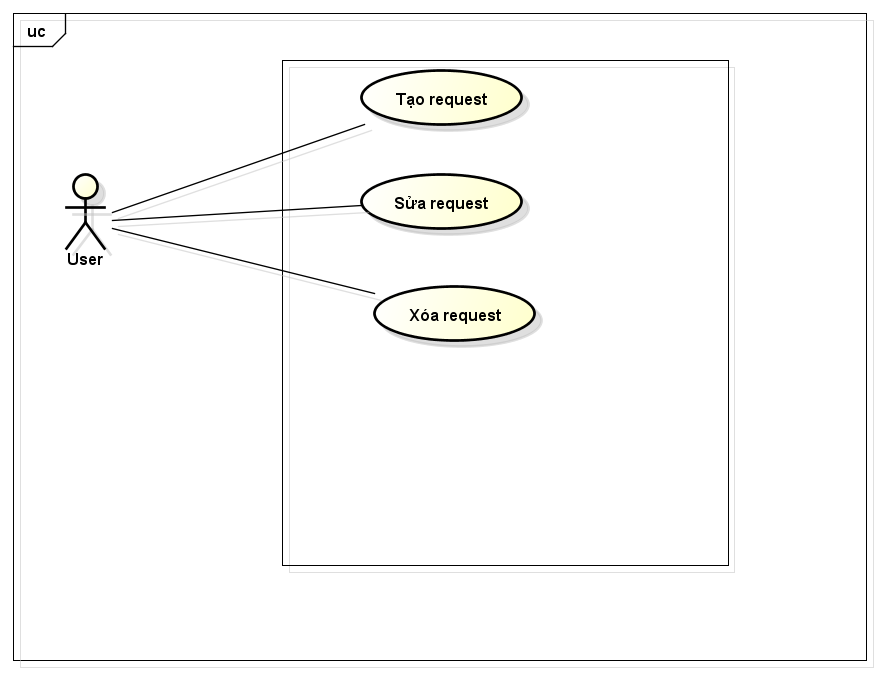
### 1.1 Sơ đồ Usecase tổng quan



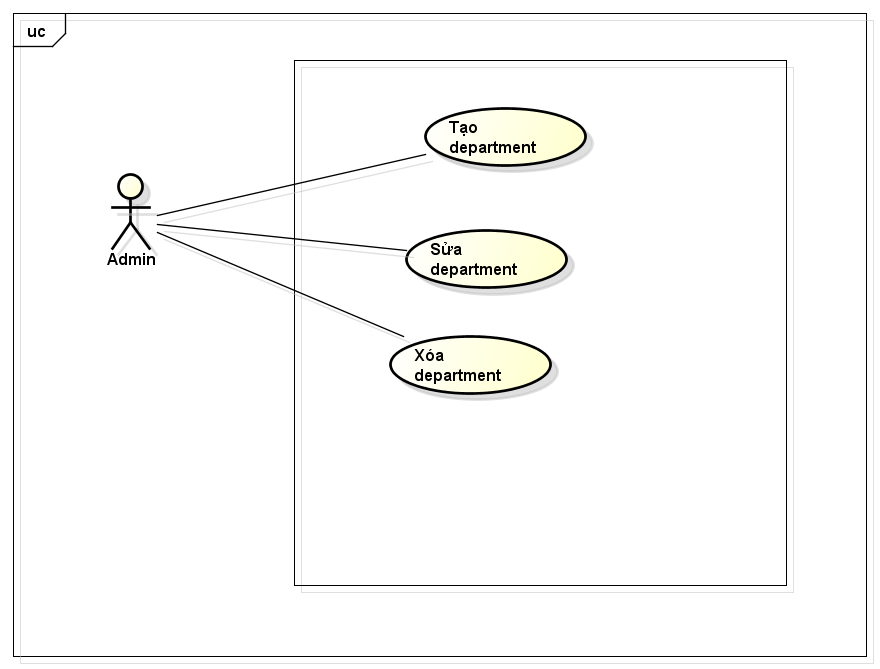
Tài khoản của người dùng phải được Admin (quản trị viên) tạo, hoặc có thể đăng nhập bằng tài khoản gmail. Khi user đăng nhập thành công, hệ thống sẽ trả về tương ứng với vai trò của người dùng trong hệ thống. Nếu là Employee những chức năng chính có trong hệ thống bao gồm: tạo yêu cầu, xóa yêu cầu, sửa yêu cầu. Đối với Manager sẽ có những chức năng chính bao gồm: tạo yêu cầu, xóa yêu cầu, sửa yêu cầu, tiếp nhận yêu cầu và từ chối yêu cầu. Cuối cùng admin sẽ có những chức năng tương tự như manager nhưng sẽ có thể những chức năng: tạo user, xóa user, sửa thông tin user, tạo phòng ban, xóa phòng ban, sửa thông tin phòng ban.

### 1.2 Sơ đồ Use Case phân rã

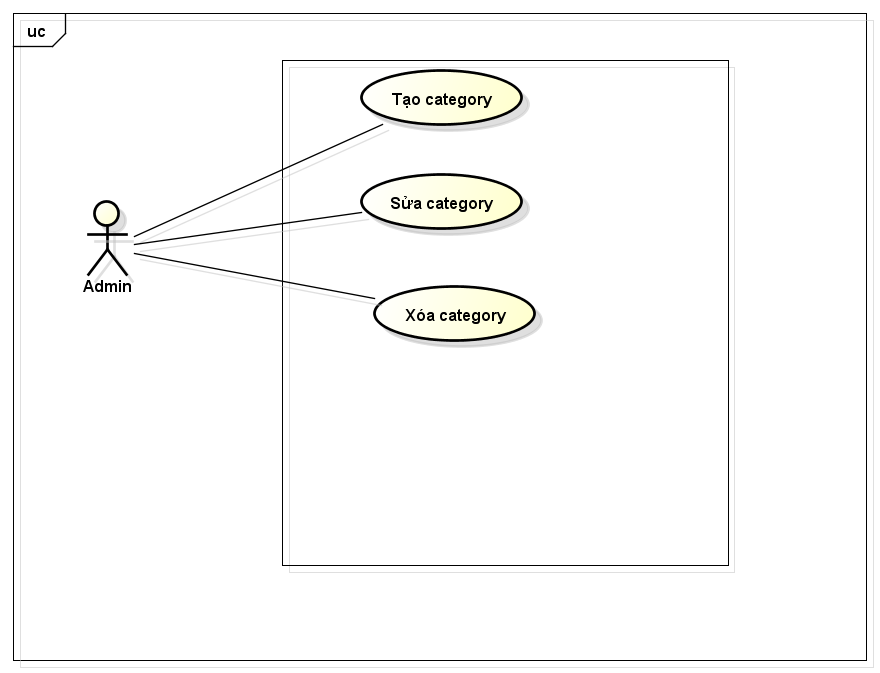
#### 1.2.1 Phân nhóm chức năng thêm sửa xóa request



#### 1.2.2 Phân nhóm chức năng thêm sửa xóa department



#### 1.2.3 Phân nhóm chức năng thêm sửa xóa category



#### 1.2.4 Phân nhóm chức năng thêm sửa xóa user

#### 

#### 

# 2. Đặc tả kịch bản sử dụng cho mỗi Usecase

## 

## 

## 2.1 Đăng nhập khi đã có tài khoản

| Mã Use case | UC001 | Tên Use case | Đăng nhập khi đã có tài khoản |
| --- | --- | --- | --- |
| Tác nhân | Employee, Manager, Admin | | |
| Tiền điều kiện | Không | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Employee, Manager, Admin | truy cập vào hệ thống | | 2. | Hệ thống | hiển thị giao diện đăng nhập | | 3. | Employee, Manager, Admin | nhập email và mật khẩu | | 4. | Employee, Manager, Admin | yêu cầu đăng nhập | | 5. | Hệ thống | kiểm tra xem khách đã nhập các trường bắt buộc nhập hay chưa | | 6. | Hệ thống | kiểm tra email và mật khẩu có hợp lệ do khách nhập trong hệ thống hay không | | 7. | Hệ thống | hiển thị trang chủ theo từng user | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 5a. | Hệ thống | thông báo lỗi: Cần nhập các trường bắt buộc nhập nếu khách nhập thiếu | | 6a. | Hệ thống | thông báo lỗi: Email và/hoặc mật khẩu chưa đúng nếu không tìm thấy email và mật khẩu trong hệ thống | | | |
| Hậu điều kiện | Không | | |

Dữ liệu đầu vào

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc?** | **Điều kiện hợp lệ** | **Ví dụ** |
| --- | --- | --- | --- | --- | --- |
| 1. | Email |  | Có |  | phong.mt184172@gmail.com |
| 2. | Mật khẩu |  | Có |  | phong1234567 |

## 2.2 Đăng nhập bằng gmail

| Mã Use case | UC002 | Tên Use case | Đăng nhập qua gmail |
| --- | --- | --- | --- |
| Tác nhân | Employee, Manager, Admin | | |
| Tiền điều kiện | Không | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Employee, Manager, Admin | truy cập vào hệ thống | | 2. | Hệ thống | hiển thị giao diện đăng nhập | | 3. | Employee, Manager, Admin | lựa chọn đăng nhập bằng gmail | | 4. | Hệ thống | yêu cầu nhập thông tin gmail và password | | 5. | Hệ thống | kiểm tra xem khách đã nhập các trường bắt buộc nhập hay chưa | | 6. | Hệ thống | kiểm tra email có tồn tại trong hệ thống | | 7. | Hệ thống | hiển thị trang chủ theo từng user | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 5a. | Hệ thống | thông báo lỗi: Cần nhập các trường bắt buộc nhập nếu khách nhập thiếu | | 6a. | Hệ thống | thông báo lỗi: Email và/hoặc mật khẩu chưa đúng | | | |
| Hậu điều kiện | Không | | |

## 2.3 Tạo yêu cầu

## 

| Mã Use case | UC003 | Tên Use case | Tạo yêu cầu |
| --- | --- | --- | --- |
| Tác nhân | Employee, Manager, Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Employee, Manager, Admin | Chọn chức năng tạo yêu cầu | | 2. | Employee, Manager, Admin | Nhập các thông tin theo yêu cầu (tên request, mô tả request, assignee, date) | | 3. | Hệ thống | Kiểm tra những trường mà người dùng vào đã hợp lệ hay chưa | | 4. | Hệ thống | Hiển thị trang chủ theo từng user | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 3a. | Hệ thống | thông báo lỗi: Cần nhập các trường bắt buộc nhập nếu khách nhập thiếu | | 3b. | Hệ thống | thông báo lỗi: Nếu người dùng nhập sai các trường bắt buộc | | | |
| Hậu điều kiện | Không | | |

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc?** | **Điều kiện hợp lệ** | **Ví dụ** |
| --- | --- | --- | --- | --- | --- |
| 1. | Tên request |  | Có |  | Sửa máy tính |
| 2. | Nội dung |  | Có |  | Phòng IT bị lỗi 2 máy tính |
| 3 | Category |  | Có |  | Sửa về mặt kỹ thuật |
| 4 | Due date |  | Có |  | 10/12/2021 |
| 5 | Assign |  | Có |  | Manager1 |

## 2.4 Sửa yêu cầu

| Mã Use case | UC004 | Tên Use case | Sửa yêu cầu |
| --- | --- | --- | --- |
| Tác nhân | Employee, Manager, Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Employee, Manager, Admin | Chọn chức năng sửa yêu cầu | | 2. | Hệ thống | Kiểm tra tình trạng của yêu cầu (Open, In process, Close) | | 3. | Hệ thống | Nếu yêu cầu trong tình trạng là open sẽ dẫn tới trang sửa | | 4. | Employee, Manager, Admin | Sửa đổi thông tin theo yêu cầu | | 5 | Hệ thống | Kiểm tra thông tin người dùng nhập vào | | 6 | Hệ thống | Thông báo thành công và trả về trang chủ | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 3a. | Hệ thống | thông báo lỗi: Tình trạng không thể sửa | | 5a. | Hệ thống | thông báo lỗi: Nếu người dùng nhập sai các trường bắt buộc | | | |
| Hậu điều kiện | Không | | |

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc?** | **Điều kiện hợp lệ** | **Ví dụ** |
| --- | --- | --- | --- | --- | --- |
| 1. | Tên request |  | Không |  | Sửa máy tính |
| 2. | Nội dung |  | Không |  | Phòng IT bị lỗi 2 máy tính |
| 3 | Category |  | Không |  | Sửa về mặt kỹ thuật |
| 4 | Due date |  | Không |  | 10/12/2021 |
| 5 | Assign |  | Không |  | Manager1 |

## 2.5 Xóa yêu cầu

## 

| Mã Use case | UC005 | Tên Use case | Xóa yêu cầu |
| --- | --- | --- | --- |
| Tác nhân | Employee, Manager, Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Employee, Manager, Admin | Chọn chức năng xóa yêu cầu | | 2. | Hệ thống | Kiểm tra tình trạng của yêu cầu (Open, In process, Close) | | 3. | Hệ thống | Nếu yêu cầu trong tình trạng là open sẽ cho phép xóa | | 4 | Hệ thống | Thông báo thành công và trả về trang chủ | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 2a. | Hệ thống | thông báo lỗi: Tình trạng không thể xóa | | | |
| Hậu điều kiện | Không | | |

## 2.6 Chấp nhận yêu cầu

| Mã Use case | UC006 | Tên Use case | Chấp nhận yêu cầu |
| --- | --- | --- | --- |
| Tác nhân | Manager | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Manager | Chọn chức năng xem yêu cầu | | 2. | Hệ thống | Đưa ra danh sách những order đã trỏ tới manager tương ứng | | 3. | Manager | Chọn approve | | 4. | Hệ thống | Kiểm tra tình trạng của request | | 5 | Hệ thống | Thông báo thành công nếu trình trạng là open | | 6 | Hệ thống | Chuyển trạng thái của order sang in process | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 5a. | Hệ thống | thông báo lỗi: Nếu tình trạng của request khác open | | | |
| Hậu điều kiện | Không | | |

## 2.7 Từ chối yêu cầu

| Mã Use case | UC007 | Tên Use case | Từ chối yêu cầu |
| --- | --- | --- | --- |
| Tác nhân | Manager | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Manager | Chọn chức năng xem yêu cầu | | 2. | Hệ thống | Đưa ra danh sách những order đã trỏ tới manager tương ứng | | 3. | Manager | Chọn reject | | 4. | Hệ thống | Kiểm tra tình trạng của request | | 5 | Hệ thống | Thông báo thành công nếu trình trạng là open | | 6 | Hệ thống | Chuyển trạng thái của order sang close | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 5a. | Hệ thống | thông báo lỗi: Nếu tình trạng của request khác open | | | |
| Hậu điều kiện | Không | | |

## 2.8 Tạo người dùng mới

## 

| Mã Use case | UC008 | Tên Use case | Tạo người dùng |
| --- | --- | --- | --- |
| Tác nhân | Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Admin | Chọn chức năng tạo người dùng | | 2. | Admin | Nhập các thông tin theo yêu cầu (tên người dùng, username, password, department, role ) | | 3. | Admin | Click vào tạo user | | 4. | Hệ thống | Kiểm tra thông tin của admin nhập vào | | 5 | Hệ thống | Thông báo cho người dùng | | 5 | Hệ thống | Trả về trang chủ | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 4a. | Hệ thống | thông báo lỗi: Cần nhập các trường bắt buộc nhập nếu khách nhập thiếu hoặc sai | | 4b. | Hệ thống | thông báo thành công | | | |
| Hậu điều kiện | Không | | |

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc?** | **Điều kiện hợp lệ** | **Ví dụ** |
| --- | --- | --- | --- | --- | --- |
| 1. | Email |  | Có |  | phong.mt184172@gmail.com |
| 2. | Name |  | Có |  | Mai Phong |
| 3 | Department |  | Có |  | Finance |
| 4 | Role |  | Có |  | Admin |
| 5 | Status |  | Có |  | In Active |

## 2.9 Sửa người dùng

## 

| Mã Use case | UC009 | Tên Use case | Sửa người dùng |
| --- | --- | --- | --- |
| Tác nhân | Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1 | Admin | Chọn danh sách các user | | 2. | Admin | Chọn chức năng sửa người dùng | | 3. | Admin | Nhập các thông tin cần sửa (tên người dùng, username, password, department ) | | 4. | Admin | Click vào sửa user | | 5. | Hệ thống | Kiểm tra thông tin của admin nhập vào | | 6 | Hệ thống | Thông báo cho người dùng | | 7 | Hệ thống | Trả về trang chủ | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 5a. | Hệ thống | thông báo lỗi: Cần nhập các trường bắt buộc nhập nếu khách nhập thiếu hoặc sai | | 5b. | Hệ thống | thông báo thành công | | | |
| Hậu điều kiện | Không | | |

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc?** | **Điều kiện hợp lệ** | **Ví dụ** |
| --- | --- | --- | --- | --- | --- |
| 1. | Email |  | Không |  | phong.mt184172@gmail.com |
| 2. | Name |  | Không |  | Mai Phong |
| 3 | Department |  | Không |  | Finance |
| 4 | Role |  | Không |  | Admin |
| 5 | Status |  | Không |  | In Active |

## 2.10 Xóa người dùng

## 

| Mã Use case | UC0010 | Tên Use case | Xóa người dùng |
| --- | --- | --- | --- |
| Tác nhân | Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1 | Admin | Chọn danh sách các user | | 2. | Admin | Chọn người muốn xóa | | 3 | Hệ thống | Thông báo cho người dùng | | 4 | Hệ thống | Trả về trang chủ | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 3a. | Hệ thống | thông báo lỗi: nếu không thể xóa | | 3b. | Hệ thống | thông báo thành công | | | |
| Hậu điều kiện | Không | | |

## 2.11 Tạo department

## 

| Mã Use case | UC0011 | Tên Use case | Tạo department |
| --- | --- | --- | --- |
| Tác nhân | Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Admin | Chọn chức năng tạo department | | 2. | Admin | Nhập các thông tin theo yêu cầu (tên department, miêu tả ) | | 3. | Admin | Click vào tạo department | | 4. | Hệ thống | Kiểm tra thông tin của admin nhập vào | | 5 | Hệ thống | Thông báo cho người dùng | | 5 | Hệ thống | Trả về trang chủ | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 4a. | Hệ thống | thông báo lỗi: Cần nhập các trường bắt buộc nhập nếu khách nhập thiếu hoặc sai | | 4b. | Hệ thống | thông báo thành công | | | |
| Hậu điều kiện | Không | | |

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc?** | **Điều kiện hợp lệ** | **Ví dụ** |
| --- | --- | --- | --- | --- | --- |
| 1. | Name |  | Có |  | Marketing |
| 2. | Description |  | Có |  | Làm về quảng cáo |

## 2.12 Tạo category

## 

| Mã Use case | UC0012 | Tên Use case | Tạo category |
| --- | --- | --- | --- |
| Tác nhân | Admin | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | Admin | Chọn chức năng tạo category | | 2. | Admin | Nhập các thông tin theo yêu cầu (name, description, assignee, status) | | 3. | Admin | Click vào tạo category | | 4. | Hệ thống | Kiểm tra thông tin của admin nhập vào | | 5 | Hệ thống | Thông báo cho người dùng | | 5 | Hệ thống | Trả về trang chủ | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 4a. | Hệ thống | thông báo lỗi: Cần nhập các trường bắt buộc nhập nếu khách nhập thiếu hoặc sai | | 4b. | Hệ thống | thông báo thành công | | | |
| Hậu điều kiện | Không | | |

| **STT** | **Trường dữ liệu** | **Mô tả** | **Bắt buộc?** | **Điều kiện hợp lệ** | **Ví dụ** |
| --- | --- | --- | --- | --- | --- |
| 1. | Name |  | Có |  | Sửa kỹ thuật |
| 2. | Description |  | Có |  | Chuyên sửa chữa thiết bị kỹ thuật |
| 3 | Assignee |  | Có |  | Mai Phong |
| 4 | Status |  | Có |  | Enable/Disable |

## 2.13 Bình luận

## 

| Mã Use case | UC0013 | Tên Use case | Bình luận |
| --- | --- | --- | --- |
| Tác nhân | Admin, Manager, User | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | User | User nhập nội dung comment | | 2. | User | Bấm Ctrl Enter để xác nhận | | 3. | Hệ thống | Lưu vào trong Database | | 4. | Hệ thống | Thông báo cho người dùng | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 3a. | Hệ thống | thông báo lỗi: Nếu không thể nhập vào trong database | | | |
| Hậu điều kiện | Không | | |

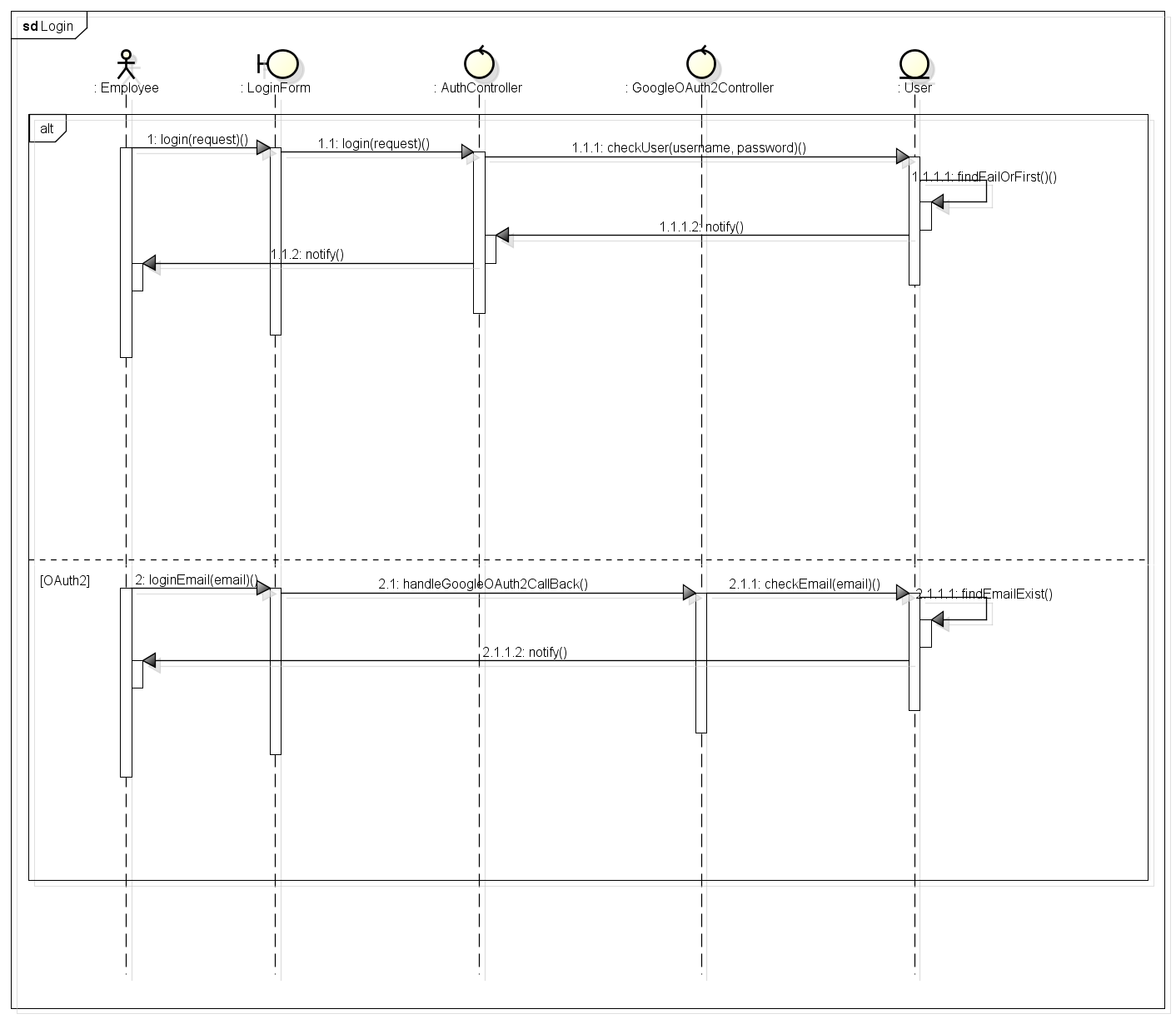
## 2.14 Xử lý request

## 

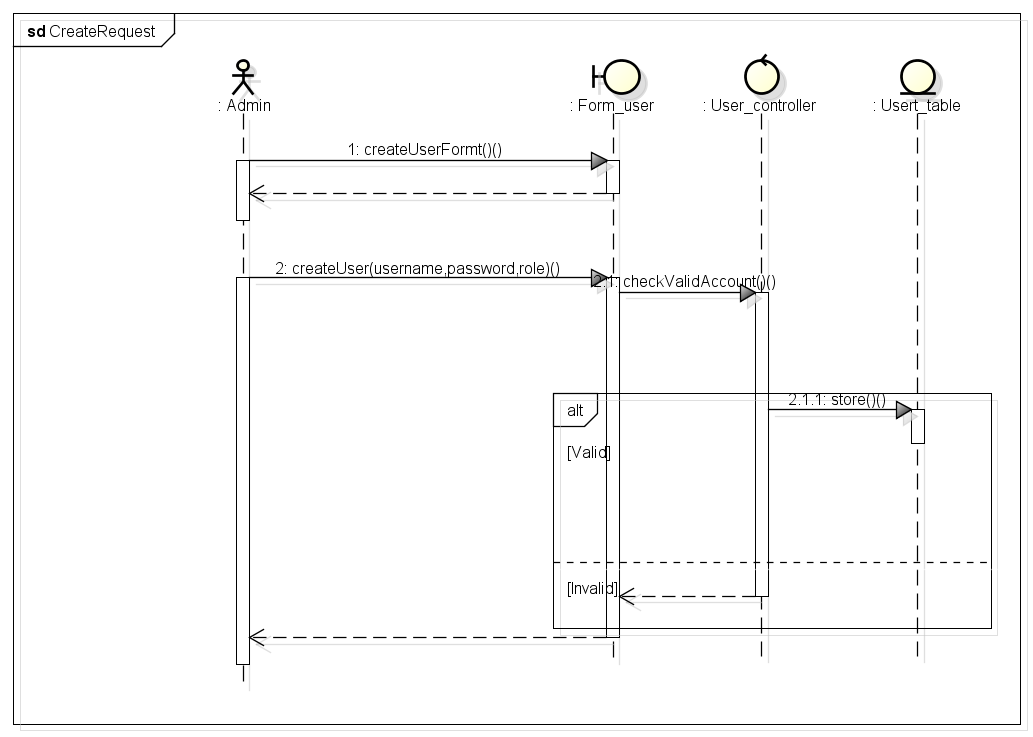
| Mã Use case | UC0014 | Tên Use case | Xử lý request |
| --- | --- | --- | --- |
| Tác nhân | Manager | | |
| Tiền điều kiện | Đã đăng nhập vào hệ thống | | |
| Luồng sự kiện chính  (Thành công) | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 1. | User | Đã tạo request | | 2. | Manager | Chuyển trạng thái Open -> In progress | | 3. | Manager | Bình luận | | | |
| Luồng sự kiện thay thế | | STT | Thực hiện bởi | Hành động | | --- | --- | --- | | 2.1 | Manager | Kết thúc request | |  |  |  | | | |
| Hậu điều kiện | Không | | |

Chương 3: Phân tích yêu cầu

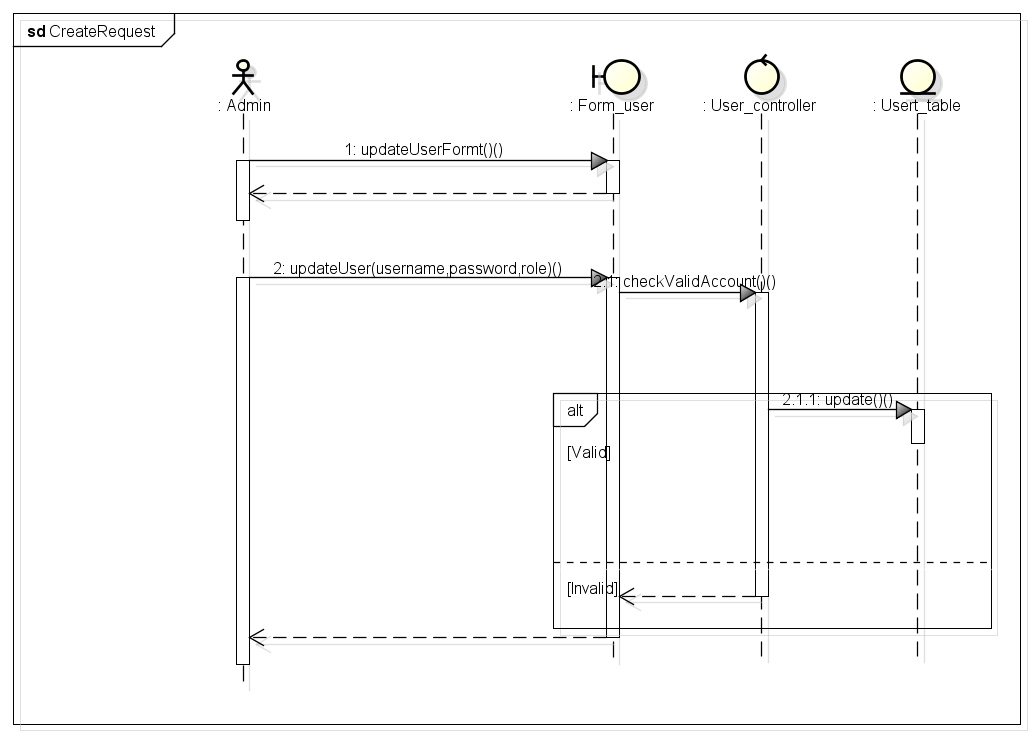
# 1.Chức năng login



# 2. Chức năng tạo user

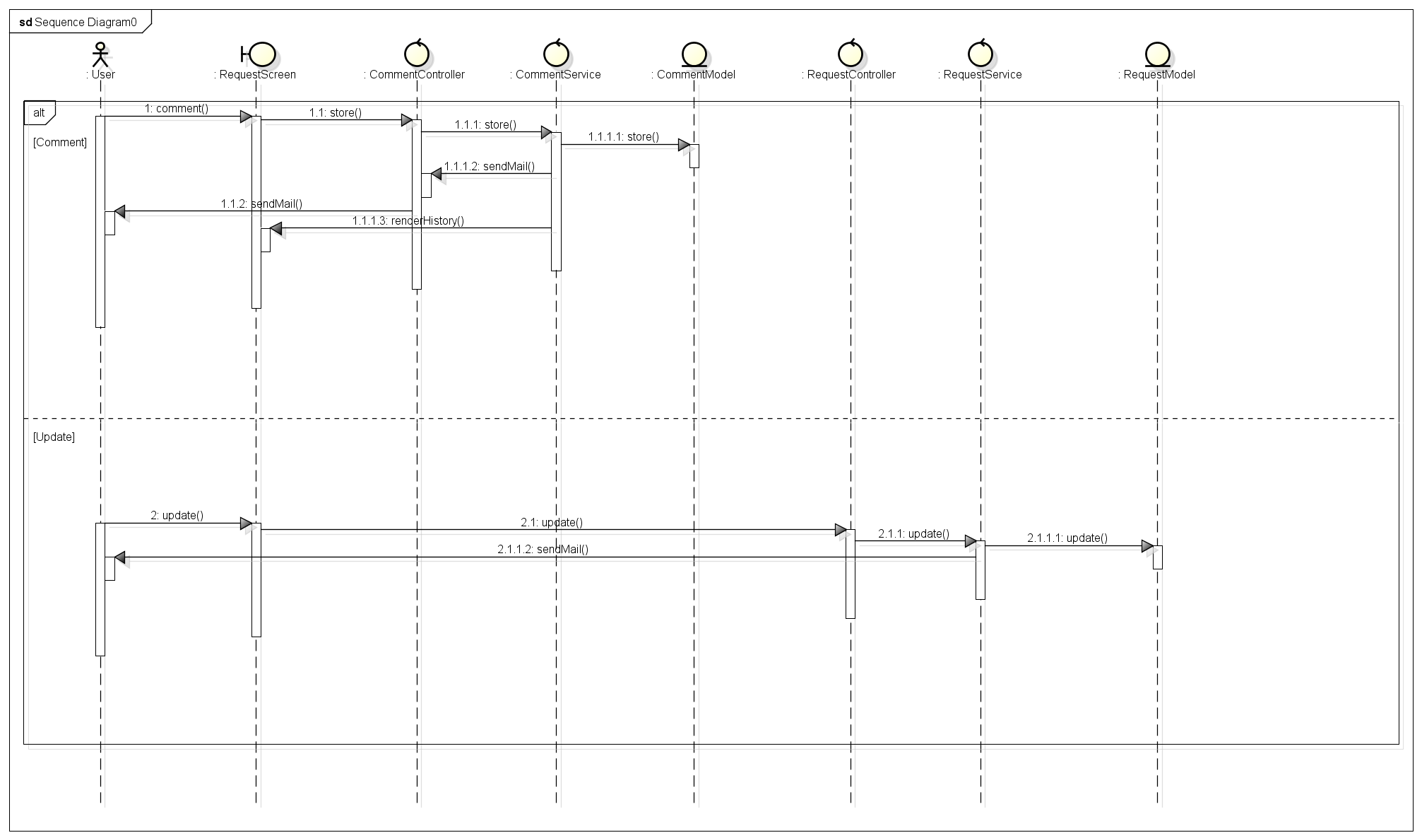


# 3. Chức năng sửa tài khoản user

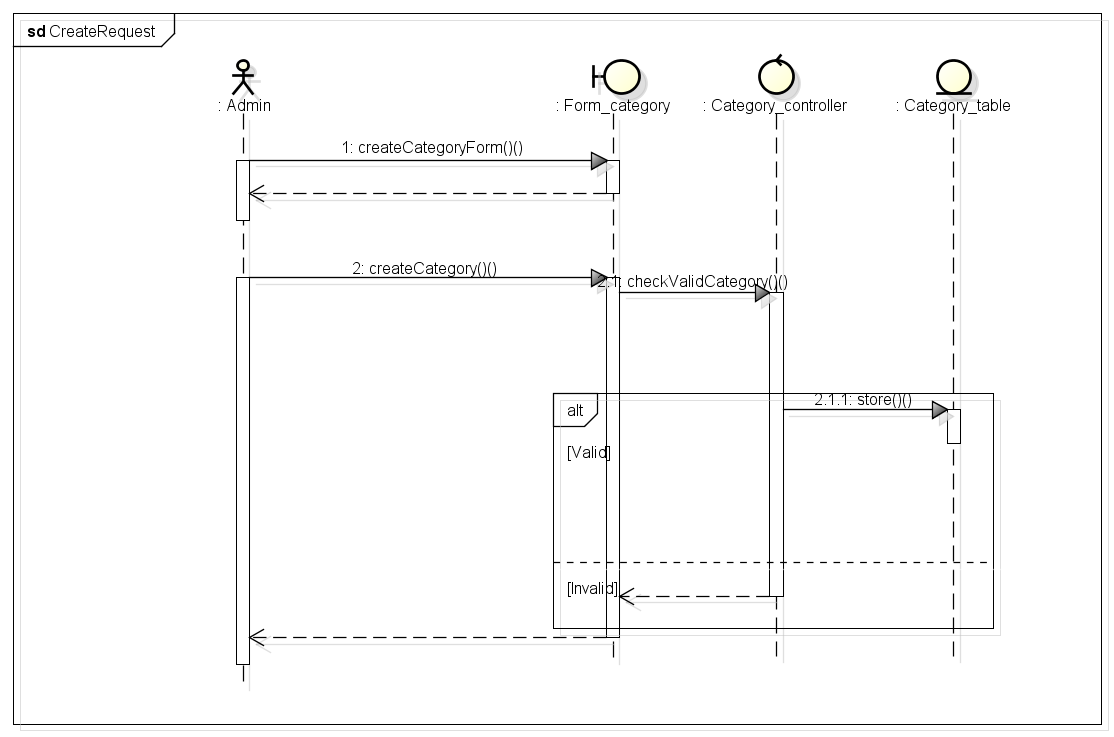


# 4. Chức năng tạo request

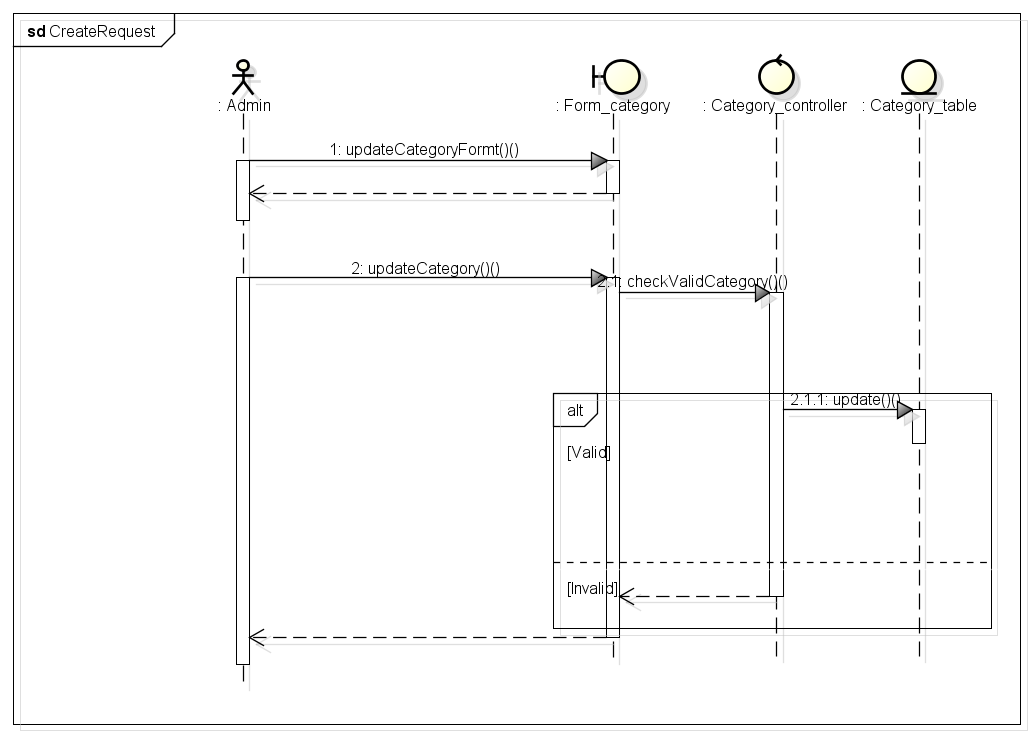
# 5. Chức năng sửa request



# 6. Chức năng tạo category



# 7. Chức năng sửa category



# 8. Chức năng tạo department

Chương 4: Thiết kế chương trình

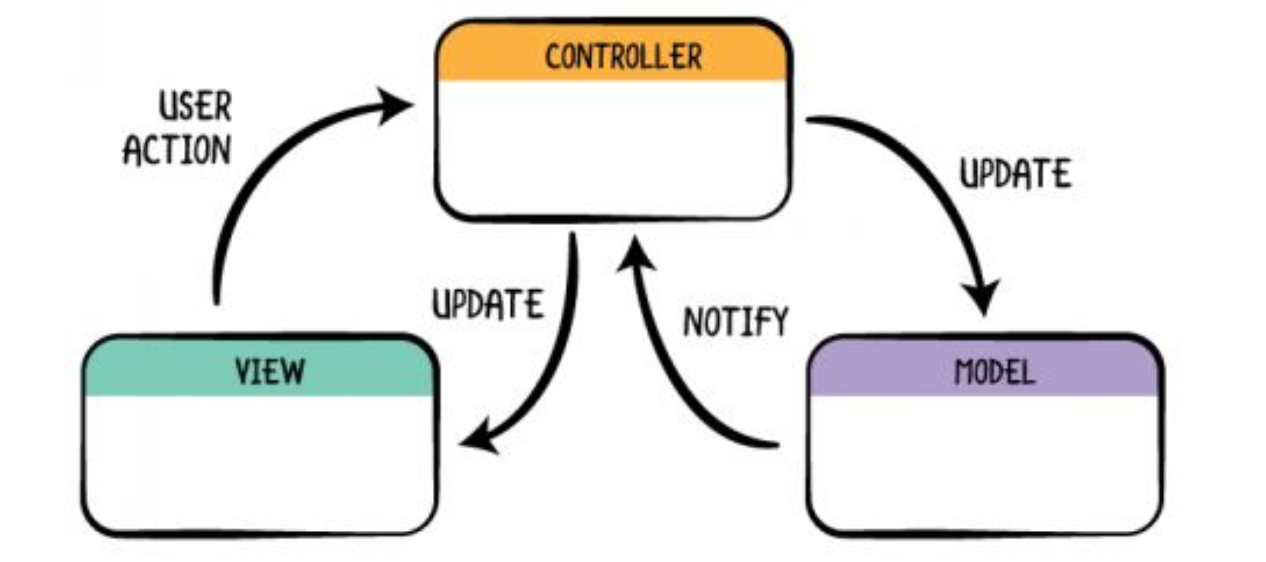
# 4.1 Thiết kế kiến trúc

## 4.1.1 Kiến trúc mô hình MVC

MVC là viết tắt của Model-View-Controller. Trong đó, Model xử lý dữ liệu.

Controller xử lý logic. Còn View là phần hiển thị và tiếp nhận request từ phía user (người

dùng).



Trong MVC, Controller đóng vai trò cầu nối giữa Model và View. Giữa

Controller-View và Controller-Model đều là tương tác 2 chiều. Mỗi phần trong đó sẽ bao gồm các đoạn code xử lý độc lập theo vai trò của mình. Mục đích chính của mô hình này nhằm chia nhỏ code để dễ phát triển và bảo trì.

Khi có một action từ phía user, ví dụ như submit form, action đó sẽ đi qua một

Controller chính. Controller này gọi đến các Controller phụ và các Model cần thiết để xử lý.

Sau đó, nó sẽ quyết định gọi đến phần View nào cần hiển thị và cập nhật lại. Đặc tính này

cũng nâng cao tính tái sử dụng của code. Trong đó, View và Model không cần phải quan tâm mình được gọi như thế nào và ở đâu

## 4.1.2 Design Pattern

Design Pattern nhóm sử dụng

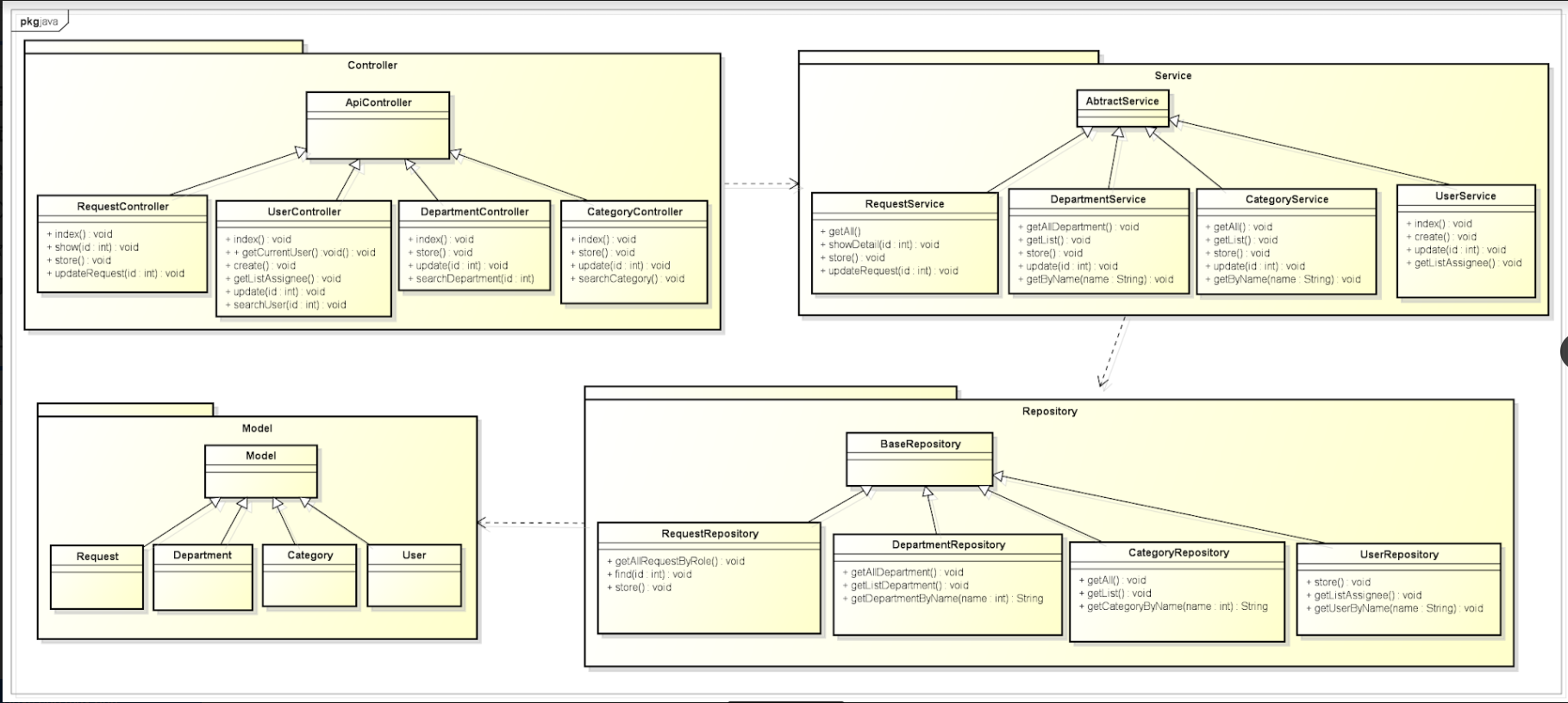
Repository thường là nơi các bạn viết các câu truy vấn database. Trong một ứng dụng, ta thường phải xử lý dữ liệu trước khi lưu vào database hoặc trước khi trả về. Thông thường ta viết liền các đoạn xử lý đó ở trong controller dẫn đến tình trạng controller bị phình to ra và rất khó đọc nếu ta có quá nhiều đoạn xử lý logic đó. Ta có một giải pháp đó là sử dụng Service Layer.

"Service Layer" nằm ở giữa phần UI và phần backend lưu dữ liệu. Nó phụ trách việc biến đổi và thông dịch dữ liệu giữa 2 tầng với nhau

về cơ bản thì, Service Layer nằm ở giữa tầng trình diễn và tầng dữ liệu. Vậy nên, đây là nơi bạn viết logic của ứng dụng. Trong bài này, ứng dụng của chúng ta sẽ có 5 tầng:

UI > Controller > Service > Repository > Model/Database

## 4.1.3 Biểu đồ gói UML



Cấu trúc Package bao gồm các phần như sau:

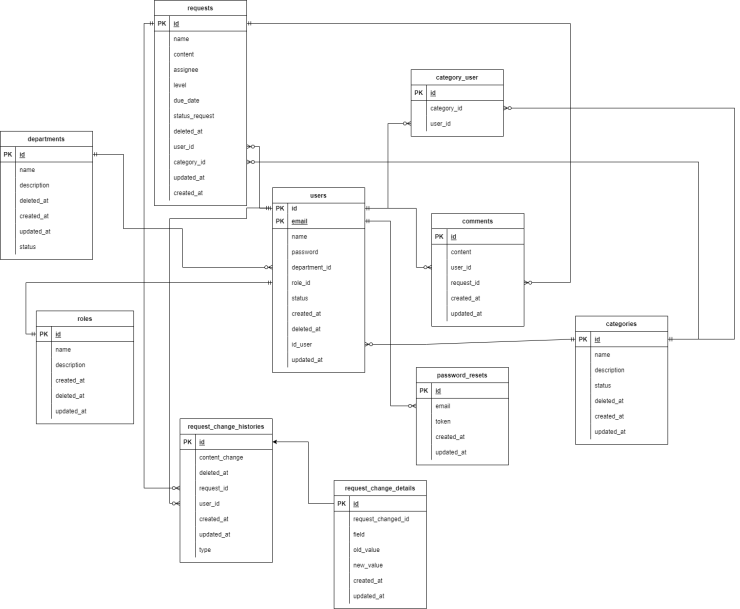
- Model: Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Bộ phận này là một cầu nối giữa 2 thành phần bên dưới là View và Controller.

- Controller: Bộ phận có nhiệm vụ trả dữ liệu đến cho front end dưới dạng JSON

- Repository: Bộ phận đảm nhiệm việc truy xuất cơ sở dữ liệu

- Service: Bộ phận sẽ thực hiện những nghiệp vụ logic và sẽ cung cấp những dịch vụ cho bên Controller

# 4.2 Thiết kế cơ sở dữ liệu



Chương 5: Xây dựng chương trình minh họa

# 5.1 Thư viện và công cụ sử dụng

Nhóm 3 sử dụng Laravel để phát triển back-end, ReactJS để phát triển front-end, thêm vào đó, nhóm cũng sử dụng MySql và IDE là Visual Studio Code. Nhóm cũng sử dụng docker để quản lý version của phần mềm của mình.

## 5.1.1 Docker

## 0. Làm quen với Docker

Trong thời đại cách mạng công nghiệp 4.0, thời đại phát triển như vũ bão về lĩnh vực công nghệ, đặc biệt là công nghệ thông tin, có vô số các công nghệ mới được sinh ra nhằm đáp ứng các nhu cầu của khách hàng và cả lập trình viên. Chính vì thế mà nhu cầu của con người ngày càng cao, ngày càng có nhiều phần mềm được ra đời, phục vụ nhu cầu của con người trong đời sống hàng ngày. Tuy nhiên, việc setup và deploy ứng dụng lên một hoặc nhiều server mất rất nhiều thời gian từ việc phải cài đặt các công cụ, môi trường cần cho ứng dụng đến việc chạy được ứng dụng, chưa kể việc không đồng nhất giữa các môi trường trên nhiều server khác nhau. Chính vì lý do đó Docker được ra đời để giải quyết vấn đề này.

Docker cho phép các nhà phát triển tách biệt ứng dụng của họ với môi trường, giải quyết vấn đề đau đầu: “It works on my machines - Ứng dụng hoạt động trên máy của tôi”. Đối với hàng triệu nhà phát triển ngày nay, Docker là tiêu chuẩn thực tế để xây dựng và chia sẻ các ứng dụng được chứa trong container.

Trong phạm vi bài viết này, mình sẽ trình bày về các khái niệm cơ bản và kiến trúc tổng quát của Docker.

## **1. Tổng quan về Docker**

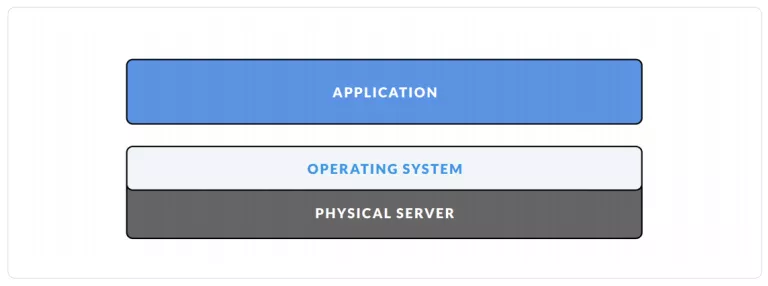
Docker là một nền tảng mở cho phát triển, vận chuyển và chạy ứng dụng, cho phép bạn tách các ứng dụng ra khỏi cơ sở hạ tầng của mình để có thể cung cấp phần mềm một cách nhanh chóng. Với Docker, bạn có thể quản lý cơ sở hạ tầng theo cùng cách quản lý ứng dụng của mình. Bằng cách tận dụng các phương pháp của Docker để vận chuyển, thử nghiệm và triển khai mã một cách nhanh chóng, bạn có thể làm giảm đáng kể sự chậm trễ giữa việc viết mã và chạy nó trong sản xuất.

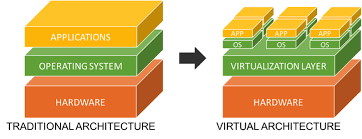
Docker cung cấp khả năng đóng gói và chạy một ứng dụng trong một môi trường tách biệt lỏng lẻo gọi là container. Nó giúp cách ly và bảo mật, cho phép bạn chạy nhiều containers đồng thời trên một máy chủ nhất định. Các container là lightweight và chúng không cần tải thêm của một hypervisor, chạy trực tiếp trên kernel của máy chủ. Điều này có nghĩa là bạn có thể chạy nhiều container hơn trên một kết hợp phần cứng nhất định hơn là nếu bạn đang sử dụng các máy ảo.

Về cơ bản, docker đóng gói một bộ file hệ thống (filesystem) hoàn chỉnh vào trong 1 container, trên container đó có thể thao tác và cài đặt mọi thư viện cần thiết để phát triển ứng dụng giống như một hệ điều hành thu nhỏ. Container này chạy trên filesystem độc lập với filesystem trên máy hiện tại (host computer), nên mọi thay đổi trên container không ảnh hưởng đến filesystem của host computer.

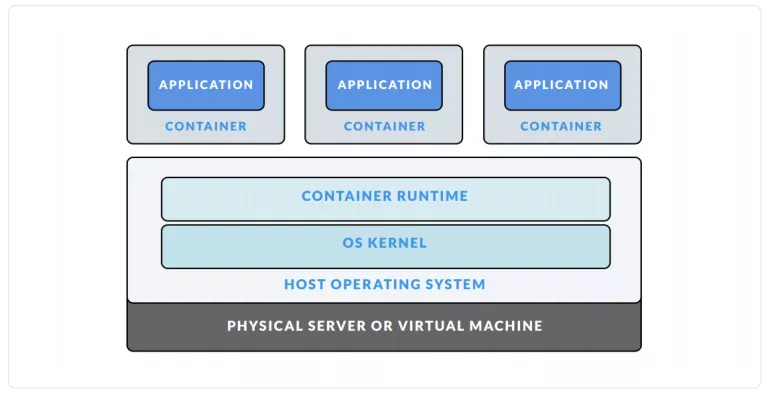
Ngoài ra do container có thể chạy giống nhau trên bất kỳ hệ điều hành nào và trên bất kỳ phần cứng nào nên developer hay sysadmin không cần quan tâm nhiều đến việc tạo môi trường để deploy ứng dụng, theo đúng “khẩu quyết” của docker là “build one, run anywhere”.

## **2. Lược sử của Docker**

Mô hình trước đây thường là 1 máy chủ vật lý + 1 hệ điều hành (OS) + 1 ứng dụng:

Khi ứng dụng phát triển lên, mô hình này sẽ nảy sinh các vấn đề lãng phí tài nguyên, khó khăn trong việc mở rộng hệ thống,.. Lúc này công nghệ ảo hóa Virtualization ra đời. Với công nghệ này, trên cùng 1 máy chủ vật lý có thể tạo ra nhiều hệ điều hành (OS), có nghĩa là ta có thể sử dụng được nhiều ứng dụng. Ưu điểm của nó là tận dụng được tốt hơn tài nguyên của máy tính, tránh được vấn đề lãng phí tài nguyên. Tuy nhiên, các vấn đề mới phát sinh mà các nhà phát triển phải đối mặt như:

* Thứ nhất, vấn đề ngốn tài nguyên khi chứa nhiều máy ảo trên một máy vật lý. Mỗi máy ảo sẽ chiếm một phần tài nguyên được cung cấp cứng - nghĩa là phần tài nguyên thực tế sẽ luôn phải cung cấp cho máy ảo mặc dù chưa thực hiện hành động nào trên đó).
* Thứ hai là vấn đề về thời gian thực thi máy ảo: khởi động hoặc tắt máy ảo tốn một khoảng thời gian rất lâu. Và việc chạy nhiều máy ảo một lúc như vậy khiến tải của server phải xử lý rất cồng kềnh, không thể chạy hết hiệu suất được.

Vì thế, các nhà phát triển bắt đầu tìm kiếm một giải pháp tốt hơn, và Containerization ra đời. Công nghệ này sinh ra nhiều máy con trên một máy chủ (giống Virtualization), tuy nhiên nó không chiếm cứng một tài nguyên nhất định nào của máy chủ như Virtualization mà nó sử dụng máy chủ như một tài nguyên dùng chung (ví dụ như RAM). Vì thế mà tài nguyên được tận dụng tối ưu hơn.Điểm nổi bật nhất của Containerization là sử dụng các container. Container là một giải pháp để chuyển giao phần mềm một cách đáng tin cậy giữa các môi trường máy tính khác nhau bằng cách:

* Tạo ra môi trường chứa mọi thứ mà phần mềm cần để có thể chạy được.
* Không bị các yếu tố liên quan đến môi trường hệ thống làm ảnh hưởng tới.
* Và không làm ảnh hưởng tới các phần còn lại của hệ thống.

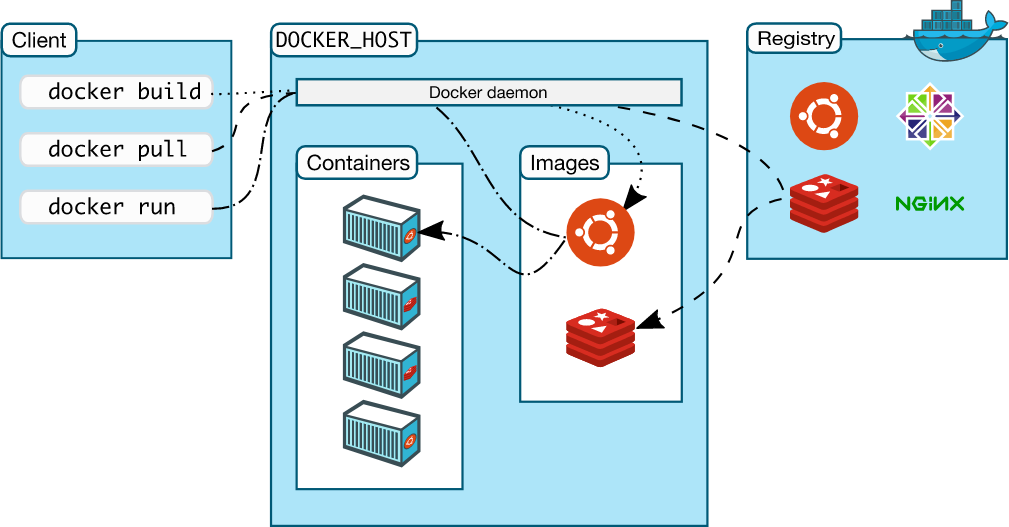
Các ưu điểm nổi bật của container là: linh động - có thể triển khai ở bất cứ đâu; nhanh - do chia sẻ tài nguyên máy tính nên container có thể được tạo gần như tức thì; nhẹ - việc chia sẻ tài nguyên giúp các container có thể dùng chung các images nên không tốn quá nhiều bộ nhớ; đồng nhất - việc phát triển cùng một dự án sẽ không bị gặp vấn đề cài đặt sai môi trường; đóng gói - container gói tất cả mọi thứ (môi trường, app,..) để chạy được ứng dụng vào một cái thùng, giúp việc quản lý dễ dàng hơn.

Nhược điểm của container này chủ yếu về tính an toàn trong bảo mật vì các container dùng chung tài nguyên của máy chủ, do đó nếu máy chủ bị hack, tất cả các container bên trong sẽ bị ảnh hưởng.

Vậy Docker có mối liên hệ gì với các công nghệ trên? Docker là một dự án mã nguồn mở được phát triển dựa trên chính các đặc tính của container trong công nghệ Containerization. Cha đẻ của Docker là Solomon Hykes, ông bắt đầu tạo ra Docker khi làm việc ở Pháp, trong một dự án nội bộ của dotCloud, ban đầu có thêm sự đóng góp của các kỹ sư dotCloud.

Docker được phát hành dạng mã nguồn mở trong tháng 3 năm 2013. Tới tháng 10 năm 2015, dự án Docker đã có hơn 25,600 sao trên GitHub (trở thành top 20 dự án có số sao cao nhất trên GitHub), có hơn 6,800 fork, và gần 1.100 lập trình viên tham gia đóng góp. Năm 2016, phân tích cho thấy các tổ chức sau có đóng góp chính cho việc phát triển Docker: nhóm Docker, Cisco, Google, Huawei, IBM, Microsoft, và Redhat.

## **3. Kiến trúc của hệ thống**

Docker sử dụng kiến trúc máy khách - máy chủ. Docker client nói chuyện với Docker daemon, trình chạy nền này thực hiện công việc building, running, và phân phối các Docker containers của mình. Docker client và daemon có thể chạy trên cùng một hệ thống hoặc có thể kết nối docker client với docker daemon từ xa. Docker client và daemon giao tiếp bằng API REST, qua ổ cắm unix hoặc giao diện mạng.

## **4.1 Daemon Docker**

Trong đa nhiệm máy tính hệ điều hành , một daemon là một chương trình máy tính mà chạy như một tiến trình nền, chứ không phải là dưới sự kiểm soát trực tiếp của một người sử dụng tương tác.

Daemon Docker (dockerd) lắng nghe các yêu cầu API Docker và quản lý các đối tượng Docker như hình ảnh, vùng chứa, mạng và khối lượng. Một daemon cũng có thể giao tiếp với các daemon khác để quản lý các dịch vụ Docker.

Dockerd là quá trình liên tục quản lý các vùng chứa. Docker sử dụng các tệp nhị phân khác nhau cho daemon và máy khách. Để chạy daemon, sử dụng lệnh dockerd. Để chạy daemon với đầu ra gỡ lỗi, sử dụng dockerd -D hoặc thêm "debug": true vào file daemon.json.

## **4.2 Docker client**

Docker client (docker) là cách chính mà nhiều người dùng Docker tương tác với Docker. Khi bạn sử dụng các lệnh như docker run, máy khách sẽ gửi các lệnh này đến dockerd, nơi thực thi chúng. Lệnh docker sử dụng Docker API. Docker client có thể giao tiếp với nhiều hơn một daemon.

## **4.3 Docker registries**

Một sổ đăng ký Docker lưu trữ các Docker images. Docker Hub là cơ quan đăng ký công khai mà bất kỳ ai cũng có thể sử dụng và Docker được định cấu hình để tìm kiếm images trong Docker Hub theo mặc định. Bạn thậm chí có thể chạy sổ đăng ký private của riêng mình.

Khi bạn sử dụng lệnh docker pull hoặc docker run, các images bắt buộc sẽ được lấy từ sổ đăng ký đã cấu hình của bạn. Khi bạn sử dụng lệnh docker push, images của bạn được đẩy vào sổ đăng ký đã định cấu hình của bạn.

## **4.4 Các đối tượng Docker**

Khi bạn sử dụng Docker, bạn đang tạo và sử dụng images, containers, mạng, ổ đĩa, plugin và các đối tượng khác. Dưới đây là tổng quan về một số đối tượng trong Docker.

### **4.4.1 Images**

Một image là một mẫu chỉ đọc (read only) với hướng dẫn để tạo ra một container Docker. Thông thường, một image dựa trên một image khác, với một số tùy chỉnh bổ sung. Ví dụ: bạn có thể xây dựng một image dựa trên image ubuntu, nhưng cài đặt máy chủ Web Apache và ứng dụng của bạn. cũng như các chi tiết cấu hình cần thiết để chạy ứng dụng của bạn.

Bạn có thể tạo image của riêng mình hoặc có thể sử dụng những image do người khác tạo và xuất bản trong sổ đăng ký. Để xây dựng một image của mình, bạn tạo một Dockerfile với cú pháp đơn giản để xác định các bước cần thiết để tạo image và chạy nó. Mỗi lệnh trong Dockerfile tạo một lớp trong image. Khi bạn thay đổi Dockerfile và xây dựng lại image, chỉ những lớp thay đổi mới được xây dựng lại. Đây là một phần lý do khiến image trở nên nhẹ, nhỏ và nhanh, khi so sánh với các công nghệ ảo hóa khác.

### **4.4.2 Containers**

Container là một thể hiện có thể chạy được của một image. Bạn có thể tạo (create), bắt đầu (start), dừng (stop), di chuyển (move), hoặc xóa (delete) một container bằng API Docker hoặc CLI. Bạn có thể kết nối một vùng chứa tới một hoặc nhiều mạng, đính kèm bộ nhớ vào nó hoặc thậm chí tạo image mới dựa trên trạng thái hiện tại của nó. Theo mặc định, một container được cách ly tương đối tốt với các container khác và máy chủ của nó. Bạn có thể kiểm soát mức độ cô lập của mạng, bộ nhớ hoặc các hệ thống con cơ bản khác của container với các container khác hoặc với máy chủ.

Container được xác định bởi image của nó cũng như bất kỳ tùy chọn cấu hình nào bạn cung cấp cho nó khi bạn khởi tạo hoặc khởi động nó. Khi một container bị xóa, mọi thay đổi đối với trạng thái của nó mà không được lưu trữ trong bộ nhớ liên tục sẽ biến mất.

### **4.4.3 Services**

Các Services cho phép ta có thể mở rộng các container trên nhiều daemon Docker, tất cả đều hoạt động cùng nhau như một nhóm với nhiều managers và workers. Mỗi thành viên của một nhóm là một daemon Docker và tất cả các daemon giao tiếp bằng API Docker. Một service cho phép bạn xác định trạng thái mong muốn, chẳng hạn như số lượng bản sao của service phải có sẵn tại mọi thời điểm. Theo mặc định, service được cân bằng tải trên tất cả các nút worker. Đối với người tiêu dùng, Docker service như là một ứng dụng duy nhất. Docker Engine hỗ trợ chế độ swarm (bầy đàn) từ Docker 1.12 trở lên.

## **5. Sử dụng docker trong dự án**

Trong phần này, chúng tôi sẽ đề cập chủ yếu vào việc ứng dụng docker khi làm project này. Nhờ có docker mà việc cài đặt môi trường khi làm việc nhóm đã trở nên rất dễ dàng, đặc biệt docker-compose sẽ thiết lập môi trường chỉ cần vài dòng lệnh.

docker-compose.yml

version: '3'

services:

php:

build:

context: .

dockerfile: Dockerfile

restart: unless-stopped

tty: true

ports:

- "9000:9000"

volumes:

- .:/var/www

networks:

- app-network

nginx:

image: nginx:alpine

restart: unless-stopped

tty: true

ports:

- "8088:80"

- "443:443"

volumes:

- .:/var/www

- ./nginx.default.conf:/etc/nginx/conf.d/default.conf

networks:

- app-network

mysql:

image: mysql:5.7

restart: unless-stopped

ports:

- "3307:3306"

volumes:

- .data/mysql:/var/lib/mysql

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABASE=request\_gate

- MYSQL\_USER=request\_gate

- MYSQL\_PASSWORD=request\_gate

networks:

- app-network

phpmyadmin:

image: phpmyadmin/phpmyadmin

environment:

- PMA\_ARBITRARY=0

- PMA\_HOST=mysql

- PMA\_PORT=3306

- PMA\_USER=request\_gate

- PMA\_PASSWORD=request\_gate

restart: unless-stopped

ports:

- "8080:80"

volumes:

- /sessions

networks:

- app-network

networks:

app-network:

driver: bridge

Trong file docker-compose.yml này chúng ta sẽ cài đặt 3 container đó là PHP, Nginx và Mysql có thêm cả phần Phpmyadmin nhưng chúng ta sẽ không sử dụng nó trong project.

* Đầu tiên là container php trái tim của project. Vì chúng ta sẽ phải cài đặt một số thứ như git, vim hay curl chính vì thế ta sẽ viết hết những câu lệnh để cài đặt những thứ đó ở trong dockerfile và đơn giản khi container php được build thì sẽ tự động chạy file dockerfile tức là sẽ cài git, curl hay vim ở trong container đó mà ta không cần vào container để cài. Điều quan trọng tiếp theo trong container này là ta phải cấu hình cổng cho nó. Ở đây tôi đang để số hiệu cổng của container này là 9000. Tiếp theo đó là volumes nơi mà chúng ta sẽ lưu trữ source code ở trong thư mục /var/www/. Cuối cùng để các container có thể giao tiếp được với nhau thì chúng sẽ phải giao tiếp thông qua một mạng mà ở trong docker gọi là bridge. Ở đây tất cả các container sẽ giao tiếp với nhau thông qua mạng có tên là app-network.
* Tiếp theo chúng ta sẽ cùng đi đến cài đặt webserver ở đây chúng ta sẽ sử dụng nginx. Ở đây sẽ khác với container php kia đó chính là chúng ta phải lên docker hub để pull được image về. Ở container php chúng ta cài đặt php thông qua dockerfile còn ở đây và container mysql tiếp theo chúng ta sẽ cài đặt bằng việc lấy các images trên docker hub về để chạy. Tiếp theo đó là việc chúng ta phải thiết lập cho nó một số hiệu cổng và thêm app-network để nó có thể giao tiếp với các container còn lại.
* Tương tự như việc chúng ta cài đặt webserver thì đối với mysql chúng ta sẽ thêm phần environment cho mysql của chúng ta, với tài khoản và database, ở đây chúng ta sẽ để mật khẩu của tài khoản root và tạo DB request\_gate. Cuối cùng vẫn như 2 container phía trên chúng ta sẽ thêm network app-network để có thể giao tiếp với các container còn lại.

Đến đây việc config môi trường đã hoàn thành, có thể nói việc sử dụng docker giúp chúng ta cài đặt môi trường khá đơn giản. Bây giờ chúng ta sẽ chỉ cần chạy lệnh docker-compose up -d để docker chạy file docker-compose.yml như chúng ta đã việc ở trên.

# 5.2 Hướng dẫn cài đặt và sử dụng chương trình

Truy cập vào đường link để tải source code: <https://drive.google.com/drive/folders/1RrQznzsT99xbi5lLRNodJpDWUV9L54UA>

Tiếp theo, cần cài đặt docker hoặc có thể sử dụng XAMPP

Hướng dẫn với docker:

<https://docs.docker.com/engine/install/ubuntu/>

Tiếp theo, làm theo hướng dẫn trong file Readme.md trong source code.

Kết luận và hướng phát triển

# Kiến thức thu được

- Nâng cao kỹ năng làm việc nhóm: tất cả thành viên trong nhóm đã cùng nhau góp sức, xây dựng ý tưởng và tìm hiểu tài liệu. Trong quá trình làm việc, có sự giúp đỡ rất nhiệt tình của cả đội. Ngoài ra việc phân công công việc cho các thành viên và thời hạn nộp kết quả các thành viên đều thực hiện rất xuất sắc.

- Kỹ năng viết báo cáo: đã biết được cấu trúc của một bài báo cáo tiêu chuẩn, cả nhóm đã dùng nhau kiểm tra những lỗi chính tả, cấu trúc báo cáo.

- Kiến thức về Design Pattern: nhóm đã học được những design pattern được sử dụng để viết trong Laravel. Mẫu Design Pattern nhóm sử dụng cũng đã được sử dụng rất phổ biến trong những dự án Laravel.

- Kiến thức xây dựng một sản phẩm phần mềm: từ những phần phân tích yêu cầu người dùng đến phân tích các lớp.

- Nắm được kiến thức cơ bản về quy trình xây dựng UML trong quy trình xây dựng sản phẩm.

# Hướng phát triển

Sản phẩm đã trong giai đoạn hoàn thiện được tất cả những chức năng mà nhóm đặt ra. Tuy nhiên trong tương lai nhóm mong muốn phát triển một số phần như sau:

* Giao diện của phần mềm cần chỉnh sửa để tăng thêm mức độ thích ứng với người dùng.
* Tạo ra những bộ test case chính xác hơn để kiểm tra tính chính xác của phần mềm.
* Thiết kế thêm API documentation để có thể phát triển sau này.
* Sử dụng cloud để có thể deploy sản phẩm tới người dùng.
* Tạo thêm những chức năng thống kế về các yêu cầu của người dùng để quản lý có thể có cái nhìn trực quan hơn về các yêu cầu.

Tài liệu tham khảo

1. Nội dung các bài học của học phần Phát triển phần mềm theo chuẩn ITSS - thầy Nguyễn Mạnh Tuấn
2. Tài liệu về Docker <https://docs.docker.com/>
3. Tài liệu về Laravel 5.7 <https://laravel.com/docs/5.7>
4. Tài liệu về ReactJS <https://reactjs.org/docs/getting-started.html>
5. UML Sequence Diagram <https://www.uml-diagrams.org/sequence-diagrams.html>