**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**In-store Customer Heatmap System**

|  |  |
| --- | --- |
| **Group 07** | |
| **Group members** | Vũ Tấn Huy – SE62172  Đỗ Quốc Cường – SE62573  Nguyễn Quang Tuyến – SE62069  Đinh Hoàng Phúc – SE61768 |
| **Supervisor** | Lâm Hữu Khánh Phương |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | ICHS |

Ho Chi Minh City, 3th May 2019

Table of Contents

[1. Project Information 1](#_Toc15377443)

[2. Introduction 1](#_Toc15377444)

[3. Current Situation 2](#_Toc15377445)

[4. Problem Definition 2](#_Toc15377446)

[5. Proposed Solution 2](#_Toc15377447)

[5.1. Feature functions 2](#_Toc15377448)

[5.2. Advantages and disadvantages 3](#_Toc15377449)

[6. Functional Requirement 3](#_Toc15377450)

[7. Role and Responsibilities 3](#_Toc15377451)

[A. Software Project Management Plan 3](#_Toc15377452)

[1. Problem Definition 3](#_Toc15377453)

[1.1. Name of This Capstone Project 3](#_Toc15377454)

[1.2. Problem Abstract 3](#_Toc15377455)

[1.3. Project Overview 4](#_Toc15377456)

[2. Project Organization 4](#_Toc15377457)

[2.1. Software Process Model 4](#_Toc15377458)

[2.2. Roles and Responsibilities 4](#_Toc15377459)

[2.3. Tools and Techniques 4](#_Toc15377460)

[3. Project Management Plan 4](#_Toc15377461)

[3.1. Software Development Life Cycle 4](#_Toc15377462)

[3.2. Phase Detail 4](#_Toc15377463)

[3.3. All Meeting Minutes 4](#_Toc15377464)

[4. Coding Convention 5](#_Toc15377465)

[B. Software Requirement Specification 5](#_Toc15377466)

[1. User Requirement Specification 5](#_Toc15377467)

[2. System Requirement Specification 6](#_Toc15377468)

[2.1. External Interface Requirement 6](#_Toc15377469)

[2.2. System Overview Use Case 6](#_Toc15377470)

[2.3. List of Use Case 7](#_Toc15377471)

[3. Software System Attribute 72](#_Toc15377472)

[3.1. Usability 72](#_Toc15377473)

[3.2. Reliability 72](#_Toc15377474)

[3.3. Availability 72](#_Toc15377475)

[3.4. Security 72](#_Toc15377476)

[3.5. Maintainability 72](#_Toc15377477)

[3.6. Portability 72](#_Toc15377478)

[3.7. Performance 72](#_Toc15377479)

[4. Conceptual Diagram 72](#_Toc15377480)

[C. Software Design Description 72](#_Toc15377481)

[1. Design Overview 72](#_Toc15377482)

[2. System Architecture Design 72](#_Toc15377483)

[3. Component Diagram 72](#_Toc15377484)

[4. Detailed Description 72](#_Toc15377485)

[5. User Interface Design 72](#_Toc15377486)

[6. Database Design 72](#_Toc15377487)

[7. Algorithms 72](#_Toc15377488)

[D. System Implementation & Test 73](#_Toc15377489)

[1. Introduction 73](#_Toc15377490)

[1.1. Overview 73](#_Toc15377491)

[1.2. Test Approach 73](#_Toc15377492)

[2. Database Relationship Diagram 73](#_Toc15377493)

[2.1. Physical Diagram 73](#_Toc15377494)

[2.2. Data Dictionary 73](#_Toc15377495)

[3. Test Plan 73](#_Toc15377496)

[3.1. Features to be Tested 73](#_Toc15377497)

[3.2. Features not to be Tested 73](#_Toc15377498)

[4. Testing Test Case 73](#_Toc15377499)

[4.1. Test Case 73](#_Toc15377500)

[4.2. .... 73](#_Toc15377501)

[E. Software User’s Manual 73](#_Toc15377502)

[1. Installation Guide 73](#_Toc15377503)

[1.1. Hardware Requirement 73](#_Toc15377504)

[1.2. Software Requirement 73](#_Toc15377505)

[2. User Guide 74](#_Toc15377506)

[F1.ソフトウェアユーザーマニュアル 74](#_Toc15377507)

[F. Appendix 74](#_Toc15377508)

# **1. Project Information**

- Project name: **In-Store Customer Heatmap System**

- Project Code: **ICHS**

- Project Type: **Web Application**

- Start Date:

- End Date:

# **2. Introduction**

In this project, we will introduce a solution in store data analysis. In the current era, information or data is always very important and useful in many fields, especially in analysis.

Our web is using technologies to make data’s analysis easier for users to manage or analyze, in which the camera supports RTSP protocol that supports converting video to the heatmap form shown on screen.

Through this web, along with the features it offers, the team wants to help managers or shop owners have a view of customer’s behavior that they can change their store and make it better.

# **3. Current Situation**

Currently, the use of shop's security cameras is simply used for monitoring purposes and when problems arise, the videos will be extracted. This inadvertently wastes the amazing effects that the camera offers. What the current camera offers is just quite difficult to exploit all that data effectively.

Nowadays, when information technology is very popular, AI (Artificial Intelligence) is also one of the things that are very interested. So, combining cameras with an AI-based application that helps users to make the most of the benefits of the camera is really a good idea.

# **4. Problem Definition**

* Applications included with the camera are usually only used for streaming and video playback.
* Streaming is mainly used for monitoring purposes.
* The extracted videos do not bring much value for analysis.
* Do not bring the most of the benefits that the camera offers.

# **5. Proposed Solution**

Our Proposed Solution is to build a system named In-Store Customer Heatmap System (ICHS), a Web application where users can view stream videos in the heatmap mode and people detection mode, can know the density of areas, count the number of people ... from there, based on reports, they are able to make assessments or reasonable changes.

## **5.1. Feature functions**

- Web application: Help user view streaming camera and see customer’s behavior.

* Video streaming: watch real-time streaming camera in website.
* See Detect people on video: see how to detect people in real-time streaming camera.
* See people counting on video: see count the result of people detection.
* See heatmap: see heatmap in real-time streaming video.
* See people analysis: see face analysis in real-time streaming video.
* Preview heatmap in time: see heatmap in each hour.
* Get report: report how many people in each time and how long people stay in each place.
* Web admin: Help admin to manage system.
* Manage company, account, store, area, camera.

- RTSP server: Get video steaming in camera and analyze it.

* Get camera streaming: use RTSP to take camera view.
* Detect people: detect people and where people are.
* Draw heatmap: base on result of detection and draw heatmap.
* Detect Face: detect face people and get their information.

## **5.2. Advantages and disadvantages**

Advantages

* System allows user see streaming camera in website.
* System can detect people and behavior to make a report for user.
* System allows user to see store/shop heatmap that can change product or style.

Disadvantages

* Don’t have many people know about heatmap

# **6. Functional Requirement**

Function requirements of the system are listed as below:

- User component:

* View Streaming camera.
* View result of detect, heatmap.
* Preview heatmap in time.
* View report.
* Admin component:
* Manage company, store, area, camera.
* RTSP server component:
* Get streaming camera.
* Detect people.
* Draw heatmap.
* Analyze face.
* Save video.
* Upload video in cloud.

# **7. Role and Responsibilities**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full Name | Role | Position | Contact |
| 1 | Mr. Lâm Hữu Khánh Phương | Project Manager | Supervisor | phuonglhk@fpt.edu.vn |
| 2 | Vũ Tấn Huy | Developer | Leader | huyvtse62176@fpt.edu.vn |
| 3 | Đỗ Quốc Cường | Developer | Member | cuongdqse62573@fpt.edu.vn |
| 4 | Nguyễn Quang Tuyến | Developer | Member | tuyennqse62069@fpt.edu.vn |
| 5 | Đinh Hoàng Phúc | Developer | Member | phucdhse61768@fpt.edu.vn |

# **Software Project Management Plan**

## **Problem Definition**

* 1. Name of This Capstone Project

**- Official name**: In-Store Customer Heatmap System

**- Vietnamese name**: Xây dựng biểu đồ nhiệt khách hàng trong cửa hàng

**- Abbreviation**: ICHS

* 1. Problem Abstract

Currently, the use of cameras in shops, stores, ... is extremely common and necessary. Cameras here are often used for observation, management or monitoring. But when technology is growing, we can fully utilize cameras to do more things. With that in mind, the develop team decided to develop a software to do it in the most optimal way.

Based on RTSP protocol, combined with camera and a stable internet connection. An web based application has been made, thanks to the combination of machine learning and algorithms, we can now count the number of people in the store through the video screen,measure store traffic in real time and show on heatmap, the owner/manager also can get some reports from the application ... thereby helping store management more proactive.

* 1. Project Overview

*1.3.1 Current Situation*

Problems encountered in this project:

* Face detection range is low.
* People detection is not 100% correct.
* Need a powerful server to load this system.
* Hard to test: base on time and brightness.

*1.3.2 The Proposed System*

We are focus to use camera as much as possible, make it have more benefit. User can access to view their camera every time everywhere and can see customer density.

For users:

* + - * Get store
      * Get area
      * Get camera
      * See streaming camera
      * See people detection
      * See people count
      * See face detection
      * See heatmap on camera
      * See heatmap in time
      * See report

For Admin:

* + - * Manage company
      * Manage account
      * Manage store
      * Manage area
      * Manage camera

For RTSP server:

* + - * Get streaming camera
      * Detect people
      * Count people
      * Detect face
      * Draw heatmap
      * Save video
      * Upload to cloud

*1.3.3 Future plans*

Divide face detection to one specific camera and use that camera to count people come in shop/store.

*1.3.4 Development Environment*

* **For server**

|  |  |  |
| --- | --- | --- |
| Windows | Minimum Requirements | Recommended |
| Internet Connection | Cable, Wi-Fi (8 Mbps) | Cable, Wi-Fi (16 Mbps) |
| Operating System | Window 7 | Window 10 |
| Computer Processor | Intel® Core i7 1.8GHz | Intel® Core i7 2.50 GHz |
| Computer Memory | 4GB RAM | 8GB RAM |

* For camera

|  |  |
| --- | --- |
| **Requirements** | |
| Lens | 2.8 mm @F2.0, horizontal field of view 105.8° |
| Lens Mount | M12 |
| Day& Night | IR cut filter with auto switch |
| Digital noise reduction | 3D DNR |
| IR Range | 30m |
| **Image** | |
| Max. Resolution | 1280 × 720 |
| Day/Night Switch | Support auto, scheduled |
| **Network** | |
| Network Storage | NAS (NFS, SMB/CIFS) |
| Protocol | RTSP |
| Alarm | Video tampering, network disconnected, IP address conflicted |
| Detections | Motion Detection |
| General Function | Anti-flicker, heartbeat, mirror, password protection, privacy mask, watermark, IP address filter |
| Communication Interface | 1 RJ45 10M/100M self-adaptive Ethernet port |

## **Project Organization**

* 1. Software Process Model

The software process model used in developing is based on the Waterfall model.

Reasons we choose this:

* RTSP server architecture need to be designed to make it fast as much as possible.
* The requirements are not often change.



* 1. Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| No | Full name | Role in Group | Responsibilities |
| 1 | Lâm Hữu Khánh Phương | Supervisor, Project Manager | * Specify user requirements * Control the development process * Give out technique and business analysis support |
| 2 | Vũ Tấn Huy | Team leader, B.A, Developer, Tester | * Managing process * Designing database * Clarifying requirements * Design architecture * GUI design * Create test plan * Coding * Testing |
| 3 | Đỗ Quốc Cường | Team member,  B.A, Developer,  Tester | * Clarifying requirements * Prepare documents * GUI design * Coding * Testing |
| 4 | Nguyễn Quang Tuyến | Team member,  B.A, Developer,  Tester | * Clarifying requirements * Prepare documents * GUI design * Coding * Testing |
| 5 | Đinh Hoàng Phúc | Team member,  B.A, Developer,  Tester | * Clarifying requirements * Prepare documents * GUI design * Coding * Testing |

* 1. Tools and Techniques

|  |  |
| --- | --- |
| **Tool** | **Name / Version** |
| Web server | Python 3.6.8 |
| Development tool | IntelliJ, IDEA, Visual Code |
| DBMS | MySQL |
| Source control | GitHub |
| Modeling tool | Star UML |
| Document tool | Microsoft Word 2016 |

Table: Tools List

|  |  |
| --- | --- |
| **Technique** | **Name / version** |
| Frontend | HTML5, CSS, TypeScript, Angular |
| Backend | Java, Spring Boot, Python 3.6.8, Flask-SocketIO, OpenCV, Redis |

Table: Techniques List

## **Project Management Plan**

* 1. Software Development Life Cycle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Description | Deliverables | Resource  needed | Dependencies  Constrains | Risks |
| Requirements Definition | Collect and analyze requirements | SRS | 14 man-days | N/A | Unclear requirements  Lack of experience |
| System and Software design | Design system structure | System architecture | 7 man-days | Depend on SRS and feature in it. | Can’t calculate how fast it can be. |
| Implementation and Unit Testing | Implement the system | Runnable system | 49 man-days | Requirement and system design | Always update system design to improve performance  At first, just use webcam to demo |
| System Testing  And Setup Camera | Test and fix bug  Setup real camera on real store | Test case  Safe system | 14 man-days | Runnable system | Hard to setup a place to set camera |
| Operation and Maintenance | Maintain system | The installation, migration, support and maintenance of complete system | 14 man-days | Camera with dynamic port |  |

* 1. Phase Detail

*3.2.1 Requirement Definition*

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1.Collect requirements | Find techniques and software which have similar feature | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |
| 2.Identify and clarify main functions | Define core main function and how to implement | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |
| 3.Requirements specification | Breakdown requirements in detail. | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |
| 4.Requirements validation | Validate requirement | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |

*3.2.2 System and Software design*

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1.Server Architecture Design | Design how server get camera and stream it into website | HuyVT |
| 2.Website Architecture Design | Design how API and website connect and technologies | CuongDQ,  TuyenNQ,  PhucDH |
| 3.Interface Design | Design UX, UI for website | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |
| 4.Database Design | Design system data structure and represent it in a database | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |

*3.2.3 Implementation and Unit testing*

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1.Implement server | Create server for get data from camera and use it to detect people and save it in database | HuyVT,  TuyenNQ |
| 2.Implement web and API | Make connection between API and website to get data | CuongDQ,  TuyenNQ,  PhucDH |
| 3.Implememt connection between server and website | Make connection between server and web to get stream camera | HuyVT,  CuongDQ |
| 4.Unit testing | Test output of each function and each case meet its design | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |

*3.2.4 System Testing and Setup Camera*

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1.Integration Testing | Test combine all of unit | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |
| 2.System Testing | Test whole system with main flow and all feature | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |
| 3.Setup Camera | Set camera in real shop/store | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |
| 4.Beta Testing | Run system with camera in real shop | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |

*3.2.5 Operation and Maintenance*

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1.Expanding system | Update new features | HuyVT,  TuyenNQ |
| 2.Maintain | Maintain system | HuyVT,  CuongDQ,  TuyenNQ,  PhucDH |

* 1. All Meeting Minutes

All Meeting minutes are contained in folder “Meeting Minutes”.

## **Coding Convention**

Python: Using to develop RTSP server

* Naming Conventions:
* Variable: using snake case, have meaningful and have separated by underscores.

Ex: current\_time, id\_camera

* Method: methods should be verb, name is using snake case, have meaningful and easily to understand.

Ex: get\_all\_camera

* Constants: constants should be uppercase all word and separated by underscores.

Ex: MODEL\_NAME

* Using Python code convention from:

<https://www.python.org/dev/peps/pep-0008/>

Java: Using to develop RESTful API server

* Naming Convention:
* Variable names should be meaningful and short, easy to understand and should not start with underscore \_ or dollar sign $ characters, even though both are allowed
* Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

Ex: run(), login(), create()

* Indentation:
* On the same line cannot put different types, variables and functions cannot declared on the same line
* Declaration Convention:
* Using Java Code Convention from:

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

# **Software Requirement Specification**

## **1.User Requirement Specification**

**1.1 Unauthorized User Requirement**

Unauthorized user is a person who just can login to use system. They can’t do anything without login. If they want to have an account to login, they must contact admin to be provided. Function Unauthorized user can use:

* Login

**1.2 Authorized User Requirement**

Authorized user is a person who can view profile and update profile. They can change their password if they don’t like old password. Function authorized user can use:

* Logout
* Change Password
* View Profile
* Update Profile

**1.3 User Requirement**

User is a person who use this web to see streaming video camera and see how it detect people and count people. By result that is detected by server, user can see report. Function user can use:

* View store
* View Area
* View Camera
* See People Detection
* View Streaming Camera
* See Face Detection
* See Count People
* View Stream Heatmap
* View Report
* Preview Heatmap in Time

**1.4 Admin Requirement**

Admin is a person who manage all data in database expect report from camera. Function admin can use:

* Manage Company
* Manage Account
* Manage Store
* Manage Area
* Manage Camera

**1.5 Server Requirement**

Server is automatic handler that use frame to detect people, face, count, draw heatmap, save video and upload to cloud. Server will automatic upload to cloud video in 00:00. Function server can use:

* Get Streaming By RTSP
* Detect People
* Detect Face
* Draw Heatmap
* Count People
* Draw Heatmap in Time
* Upload Cloud
* Save Video

## **2.System Requirement Specification**

2.1 External Interface Requirement

*2.1.1 User Interface*

* + - Website GUI should be simple, clear and use primary language is English.
    - Color is easily to watch in night or day.

*2.1.2 Hardware Requirement*

* + - Camera must be IP Camera and has supports RTSP.

*2.1.3 Software Interface*

* + - Web application: Chromes, Firefox.

*2.1.1 Communication Protocol*

* + - Use RTSP (Real Time Streaming Protocol) for communication between camera and RTSP server.
    - Use socket-IO for communication between web and RTSP server.
    - Use HTTP for communication between web and API.

2.2 System Overview Use Case

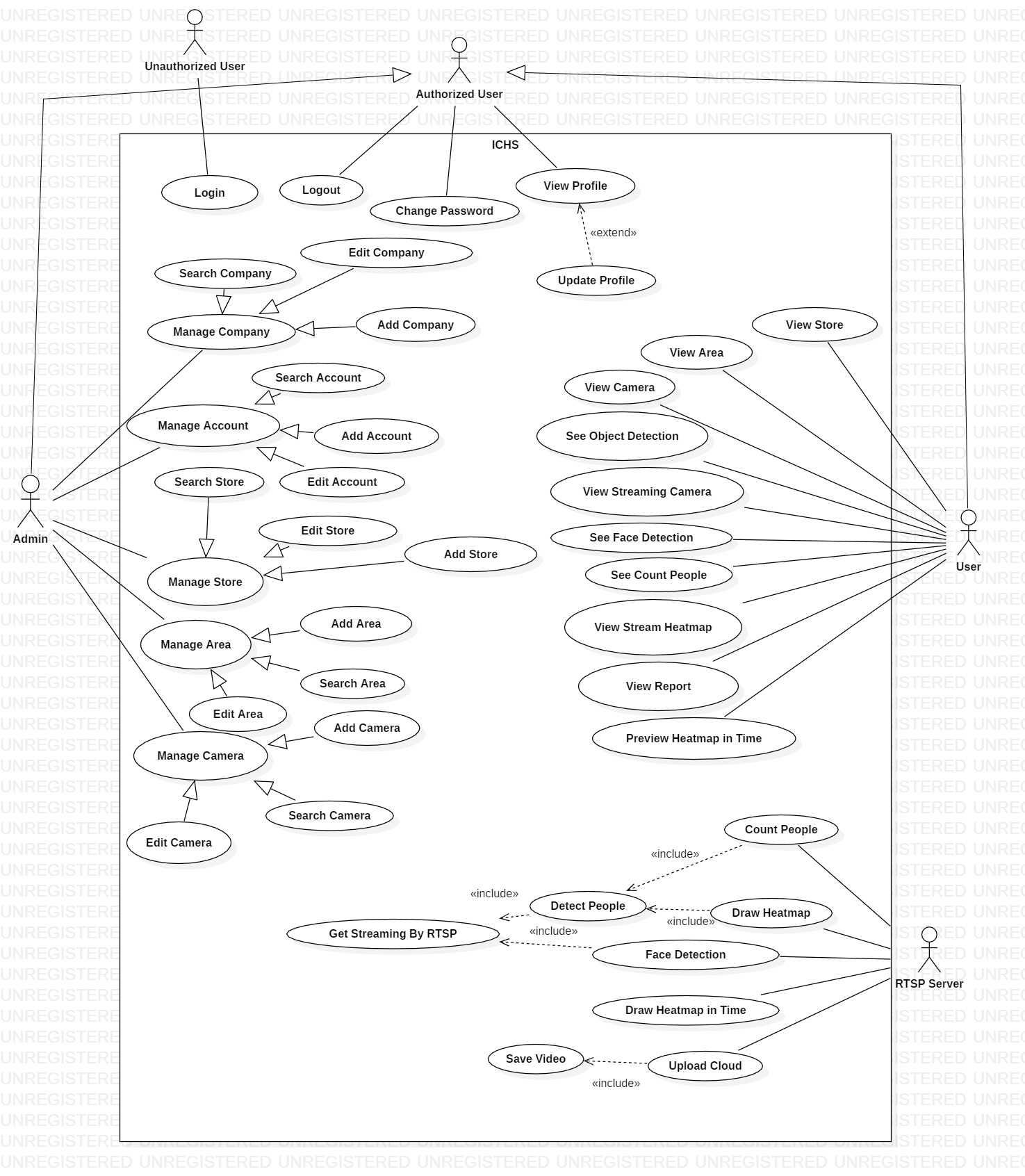


Figure 2- System Overview Use Case Diagram

* 1. List of Use Case

<Admin> Overview Use Case

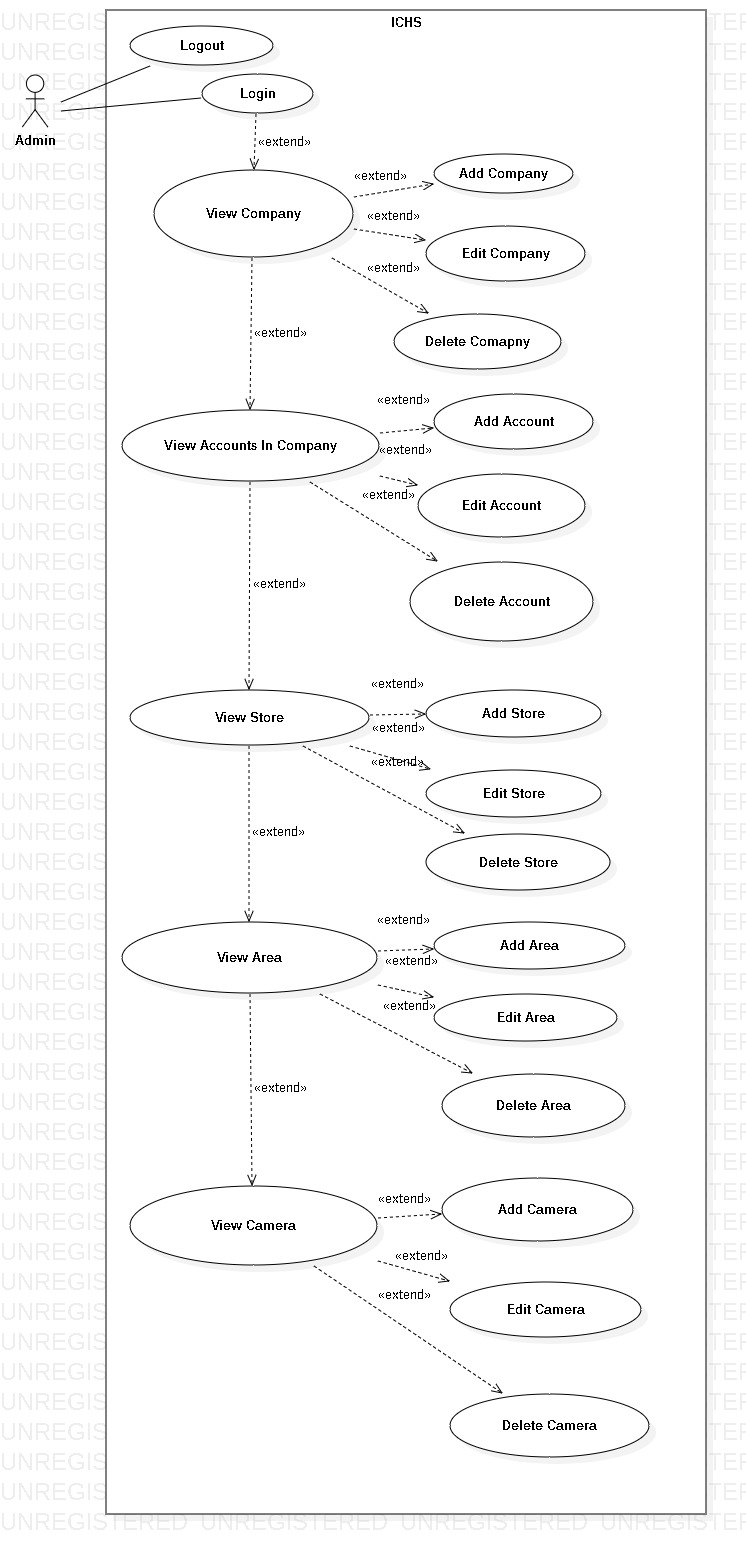


Figure 3- <Admin>Overview Use Case Diagram

##### <Admin> Admin Login



Figure 4- <Admin> Admin Login

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_01** | | | |
| **Use Case No.** | 01 | **Use Case Version** | 1.0 |
| **Use Case Name** | Admin Login | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin login to the system.   **Goal:**   * Admin logged in to the system.   **Triggers:**   * Admin clicks “Login” button. * Admin sends the login command.   **Preconditions:**   * Admin at “Login” page.   **Post Conditions:**   * **Success:** Admin login to the system. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin goes to Login page by clicking “Login” button. | System requires information:   * Username: text input. * Password: text input. | | 2 | Admin inputs information and clicks “Login” button. | System validates the information and displays confirmation message. |   **Alternative:** N/A.  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System displays “Login failed!” message. | | 2 | Username is blank. | System displays “Please input Username” message. | | 3 | Password is blank. | System displays “Please input Password” message. |   **Relationships:** N/A.  **Business Rules:**   * After logged in, system redirect to Admin page. * A notification pop-up when admin login success. * System must ensure has no duplicate user. * Password must be in rage of 6 – 30 characters. | | | |

Table - <Admin> Admin Login

##### <Admin> Admin Logout

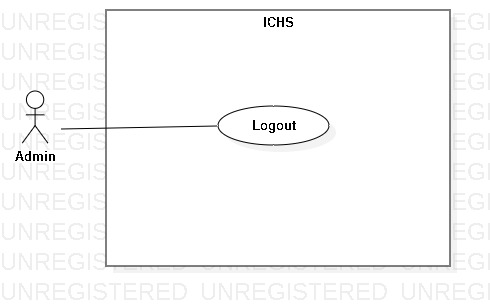


Figure 5- <Admin> Admin Logout

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_02** | | | |
| **Use Case No.** | 02 | **Use Case Version** | 1.0 |
| **Use Case Name** | Admin Logout | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin logout from application.   **Goal:**   * Admin logout from the system.   **Triggers:**   * Admin clicks Settings button. * Admin clicks “Log Out” button. * Admin sends the logout command.   **Preconditions:**   * Admin login success.   **Post Conditions:**   * **Success:** Admin logout from system. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Log Out” button. | System displays dialog with “You want to log out?” message. | | 2 | Admin clicks “Yes” button. | Admin logout from the system. |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Close the dialog. |   **Relationships:** N/A.  **Business Rules:**   * After logout from the system, system return Login page. * Admin can login with another account. | | | |

Table - <Admin> Admin Logout

##### <Admin> View Company

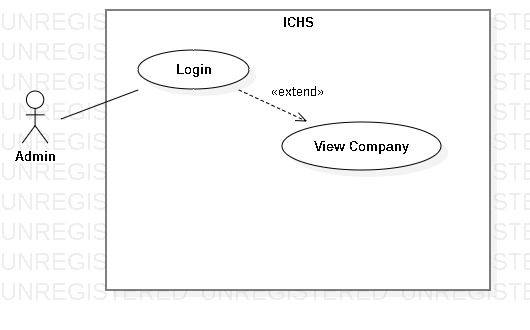


Figure 6- <Admin> View Company

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_03** | | | |
| **Use Case No.** | 03 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Company | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to view all companies in system.   **Goal:**   * Admin view all companies in the system.   **Triggers:**   * Admin clicks “Company” button. * Admin sends view company command.   **Preconditions:**   * Admin login success.   **Post Conditions:**   * **Success:** Admin view all companies in system. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Company” button in Home page. | System displays the “Company” page and shows companies. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “Admin Login” use case.  **Business Rules:** N/A. | | | |

Table - <Admin> View Company

##### <Admin> Add Company

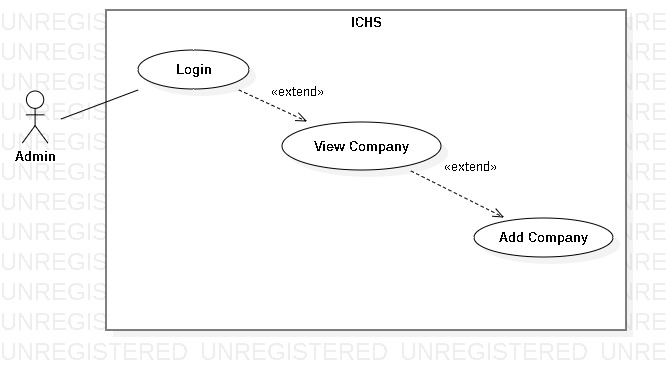


Figure 7- <Admin> Add Company

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_04** | | | |
| **Use Case No.** | 04 | **Use Case Version** | 1.0 |
| **Use Case Name** | Add Company | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to add company.   **Goal:**   * Admin add new company.   **Triggers:**   * Admin clicks “Add Company” button. * Admin sends add company command.   **Preconditions:**   * Admin login success. * Admin is at “Company” page.   **Post Conditions:**   * **Success:** Admin add company successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Add Company” button. | System displays “Add Company” page. Page contains 3 text fields and requires information :   * Company Name: text input * Company Address: text input * Company Phone Number: text input | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Add Company Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Company page. | | 2 | Company Name is blank. | System displays “Please input Company Name” message. | | 3 | Company Address is blank. | System displays “Please input Company Address” message. | | 4 | Company Phone Number is blank. | System displays “Please input Company Phone Number” message. |   **Relationships:** Extend from “View Company” use case.  **Business Rules:**   * After add new company to the system, admin return to Company page. | | | |

Table - <Admin> Add Company

##### <Admin> Edit Company



Figure 7- <Admin> Edit Company

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_05** | | | |
| **Use Case No.** | 05 | **Use Case Version** | 1.0 |
| **Use Case Name** | Edit Company | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to edit company.   **Goal:**   * Admin edit existing company.   **Triggers:**   * Admin clicks “Edit Company” button. * Admin sends edit company command.   **Preconditions:**   * Admin login success. * Admin is at “Company” page.   **Post Conditions:**   * **Success:** Admin edit company successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Edit Company” button. | System displays “Edit Company” page. Page contains 3 text fields and requires information:   * Company Name: text input * Company Address: text input * Company Phone Number: text input | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Edit Company Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Company page. | | 2 | Company Name is blank. | System displays “Please input Company Name” message. | | 3 | Company Address is blank. | System displays “Please input Company Address” message. | | 4 | Company Phone Number is blank. | System displays “Please input Company Phone Number” message. |   **Relationships:** Extend from “View Company” use case.  **Business Rules:**   * After edit company, admin return to Company page. | | | |

Table - <Admin> Edit Company

##### <Admin> Delete Company

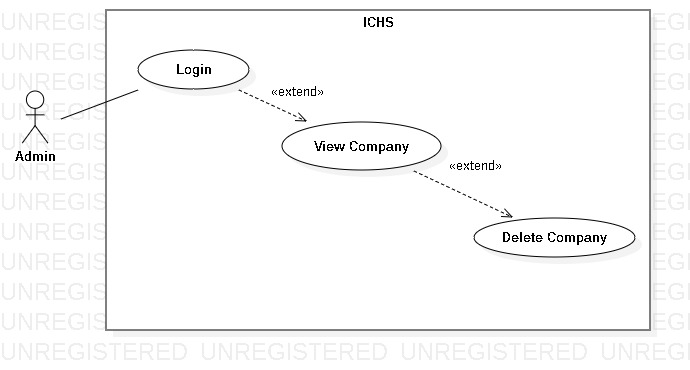


Figure 8- <Admin> Delete Company

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_06** | | | |
| **Use Case No.** | 06 | **Use Case Version** | 1.0 |
| **Use Case Name** | Delete Company | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to delete company.   **Goal:**   * Admin delete existing company.   **Triggers:**   * Admin clicks “Delete Company” button. * Admin sends delete company command.   **Preconditions:**   * Admin login success. * Admin is at “Company” page.   **Post Conditions:**   * **Success:** Admin delete company successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Delete Company” button. | System displays “Do you want to delete this company?” message. | | 2 | Admin clicks “Ok” button | System removes the company from Company page. |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Company page. |   **Relationships:** Extend from “View Company” use case.  **Business Rules:**   * After delete company, admin return to Company page. | | | |

Table - <Admin> Delete Company

##### <Admin> View Accounts In Company

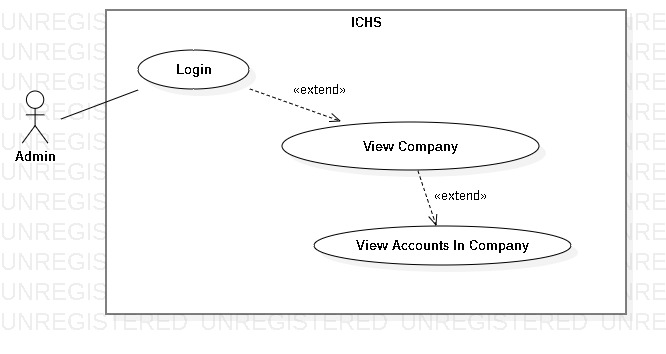


Figure 9- <Admin> View Accounts In Company

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_07** | | | |
| **Use Case No.** | 07 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Accounts In Company | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to view all accounts in one company.   **Goal:**   * Admin view all accounts in the chosen company.   **Triggers:**   * Admin clicks “View Account” button. * Admin sends view account command.   **Preconditions:**   * Admin login success. * Admin is at the chosen company page.   **Post Conditions:**   * **Success:** Admin view all accounts in chosen company. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “View Account” button in Home page. | System displays the “Account” page and shows all accounts in the chosen company. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Company” use case.  **Business Rules:** N/A. | | | |

Table - <Admin> View Accounts In Company

##### <Admin> Add Account

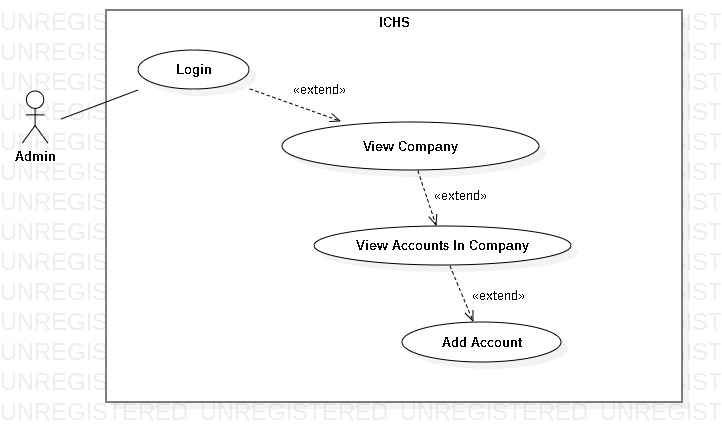


Figure 10- <Admin> Add Account

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_08** | | | |
| **Use Case No.** | 08 | **Use Case Version** | 1.0 |
| **Use Case Name** | Add Account | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to add new account to company.   **Goal:**   * Admin add new account.   **Triggers:**   * Admin clicks “Add Account” button. * Admin sends add account command.   **Preconditions:**   * Admin login success. * Admin is at “Account” page.   **Post Conditions:**   * **Success:** Admin add account successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Add Account” button. | System displays “Add Account” page. Page contains 5 text fields, 2 checkboxes and requires information:   * User Name: text input * Password: text input * Full Name: text input * Email: text input * Gender: checkbox * Phone Number: text input * Role: checkbox | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Add Account Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Account page. | | 2 | Username is blank. | System displays “Please input Username” message. | | 3 | Password is blank. | System displays “Please input Password” message. | | 4 | Fullname is blank. | System displays “Please input Fullname” message. | | 5 | Email is blank. | System displays “Please input Email” message. | | 6 | Phone Number is blank. | System displays “Please input Phone Number” message. | | 7 | Username hase been used. | System displays “This username has been used” message. |   **Relationships:** Extend from “View Accounts In Company” use case.  **Business Rules:**   * After add new account to the company, admin return to Account page. | | | |

Table - <Admin> Add Account

##### <Admin> Edit Account

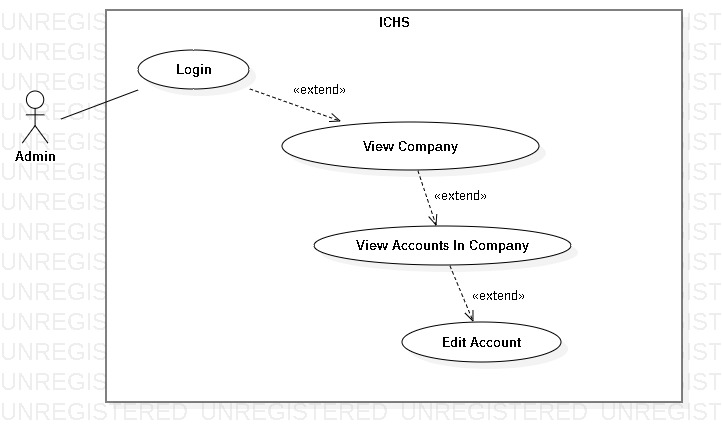


Figure 11- <Admin> Edit Account

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_09** | | | |
| **Use Case No.** | 09 | **Use Case Version** | 1.0 |
| **Use Case Name** | Edit Account | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to edit account.   **Goal:**   * Admin edit existing account.   **Triggers:**   * Admin clicks “Edit Account” button. * Admin sends edit account command.   **Preconditions:**   * Admin login success. * Admin is at “Account” page.   **Post Conditions:**   * **Success:** Admin edit account successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Edit Account” button. | System displays “Edit Account” page. Page contains 5 text fields, 2 checkboxes and requires information:   * User Name: text input * Password: text input * Full Name: text input * Email: text input * Gender: checkbox * Phone Number: text input * Role: checkbox | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Edit Account Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Account page. | | 2 | Username is blank. | System displays “Please input Username” message. | | 3 | Password is blank. | System displays “Please input Password” message. | | 4 | Fullname is blank. | System displays “Please input Fullname” message. | | 5 | Email is blank. | System displays “Please input Email” message. | | 6 | Phone Number is blank. | System displays “Please input Phone Number” message. | | 7 | Username has been used. | System displays “This username has been used” message. |   **Relationships:** Extend from “View Accounts In Company” use case.  **Business Rules:**   * After edit account, admin return to Account page. | | | |

Table - <Admin> Edit Account

##### <Admin> Delete Account

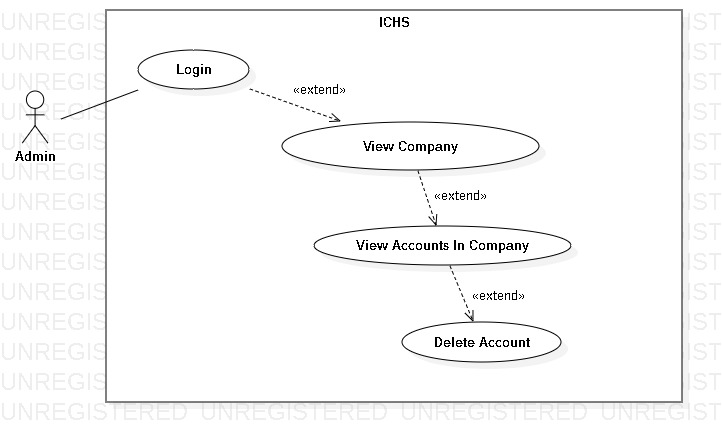


Figure 12- <Admin> Delete Account

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_10** | | | |
| **Use Case No.** | 10 | **Use Case Version** | 1.0 |
| **Use Case Name** | Delete Account | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to delete account.   **Goal:**   * Admin delete existing account.   **Triggers:**   * Admin clicks “Delete Account” button. * Admin sends delete account command.   **Preconditions:**   * Admin login success. * Admin is at “Account” page.   **Post Conditions:**   * **Success:** Admin delete account successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Delete Account” button. | System displays “Do you want to delete this account ?” message. | | 2 | Admin clicks “Ok” button | System removes the account from Account page. |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Account page. |   **Relationships:** Extend from “View Accounts in Company” use case.  **Business Rules:**   * After delete account, admin return to Account page. | | | |

Table - <Admin> Delete Account

##### <Admin> View Store

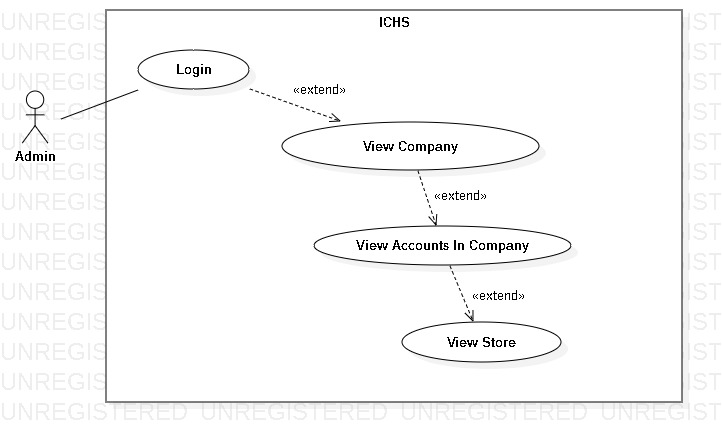


Figure 13- <Admin> View Store

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_11** | | | |
| **Use Case No.** | 11 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Store | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to view all store in company that the chosen account is allowed to view.   **Goal:**   * Admin view all stores of user within the limits allowed by the admin.   **Triggers:**   * Admin clicks “Store” button. * Admin sends view store command.   **Preconditions:**   * Admin login success. * Admin is at the chosen account page.   **Post Conditions:**   * **Success:** Admin view all stores. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Store” button in Account page. | System displays the “Store” page and shows stores. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Accounts In Company” use case.  **Business Rules:** N/A. | | | |

Table - <Admin> View Store

##### <Admin> Add Store

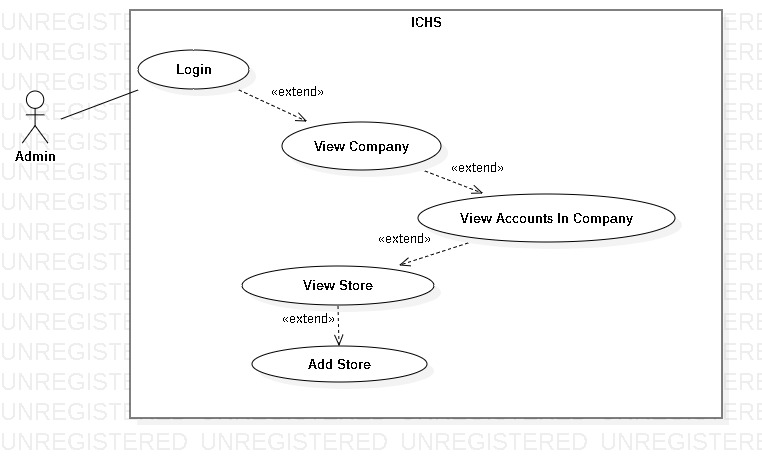


Figure 14- <Admin> Add Store

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_12** | | | |
| **Use Case No.** | 12 | **Use Case Version** | 1.0 |
| **Use Case Name** | Add Store | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to add store.   **Goal:**   * Admin add new store.   **Triggers:**   * Admin clicks “Add Store” button. * Admin sends add store command.   **Preconditions:**   * Admin login success. * Admin is at “Store” page.   **Post Conditions:**   * **Success:** Admin add store successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Add Store” button. | System displays “Add Store” page. Page contains 3 text fields and requires information:   * Store Name: text input * Store Address: text input * Store Phone Number: text input | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Add Store Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Store page. | | 2 | Store Name is blank. | System displays “Please input Store Name” message. | | 3 | Store Address is blank. | System displays “Please input Store Address” message. | | 4 | Store Phone Number is blank. | System displays “Please input Store Phone Number” message. |   **Relationships:** Extend from “View Store” use case.  **Business Rules:**   * After add new store, admin return to Store page. | | | |

Table - <Admin> Add Store

##### <Admin> Edit Store

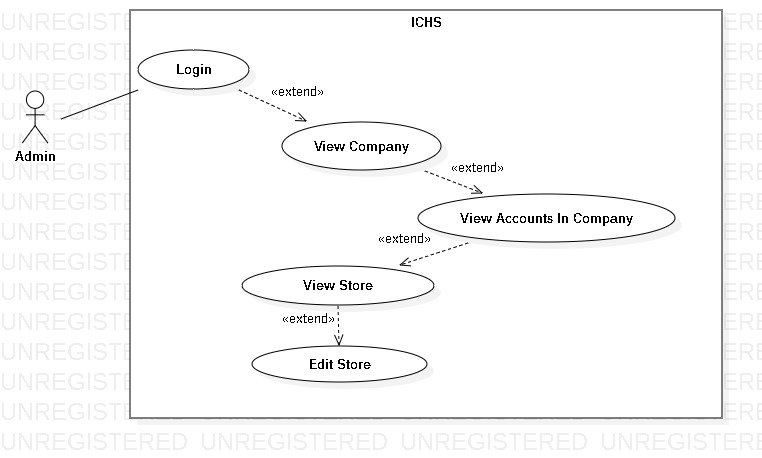


Figure 15- <Admin> Edit Store

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_13** | | | |
| **Use Case No.** | 13 | **Use Case Version** | 1.0 |
| **Use Case Name** | Edit Store | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to edit store.   **Goal:**   * Admin edit existing store.   **Triggers:**   * Admin clicks “Edit Store” button. * Admin sends edit store command.   **Preconditions:**   * Admin login success. * Admin is at “Store” page.   **Post Conditions:**   * **Success:** Admin edit store successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Edit Store” button. | System displays “Edit Store” page. Page contains 3 text fields and requires information:   * Store Name: text input * Store Address: text input * Store Phone Number: text input | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Edit Store Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Store page. | | 2 | Store Name is blank. | System displays “Please input Store Name” message. | | 3 | Store Address is blank. | System displays “Please input Store Address” message. | | 4 | Store Phone Number is blank. | System displays “Please input Store Phone Number” message. |   **Relationships:** Extend from “View Store” use case.  **Business Rules:**   * After edit store , admin return to Store page. | | | |

Table - <Admin> Edit Store

##### <Admin> Delete Store

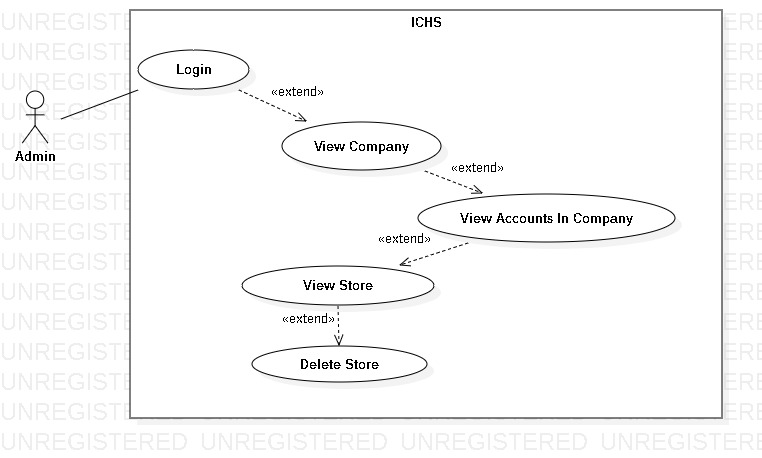


Figure 16- <Admin> Delete Store

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_14** | | | |
| **Use Case No.** | 14 | **Use Case Version** | 1.0 |
| **Use Case Name** | Delete Store | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to delete store.   **Goal:**   * Admin delete existing store.   **Triggers:**   * Admin clicks “Delete Store” button. * Admin sends delete store command.   **Preconditions:**   * Admin login success. * Admin is at “Store” page.   **Post Conditions:**   * **Success:** Admin delete store successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Delete Store” button. | System displays “Do you want to delete this store?” message. | | 2 | Admin clicks “Ok” button | System removes the store from Store page. |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Store page. |   **Relationships:** Extend from “View Store” use case.  **Business Rules:**   * After delete store, admin return to Store page. | | | |

Table - <Admin> Delete Store

##### <Admin> View Area

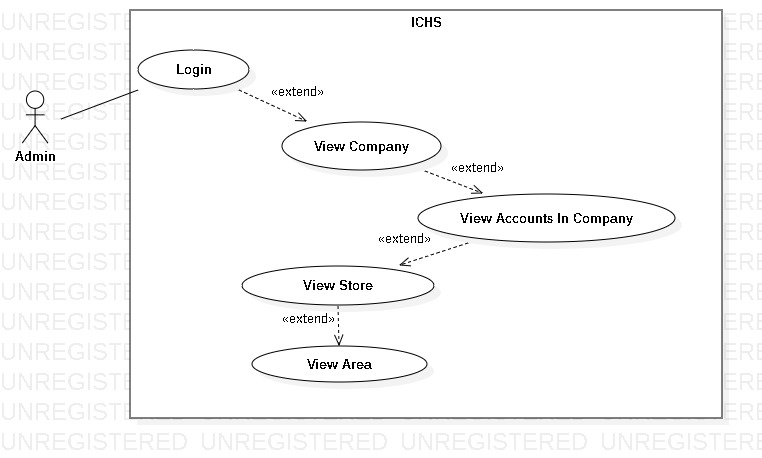


Figure 17- <Admin> View Area

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_15** | | | |
| **Use Case No.** | 15 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Area | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to view all areas in one store.   **Goal:**   * Admin view all areas in the chosen store.   **Triggers:**   * Admin clicks “Area” button. * Admin sends view area command.   **Preconditions:**   * Admin login success. * Admin is at the chosen store page.   **Post Conditions:**   * **Success:** Admin view all areas. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Area” button in Store page. | System displays the “Area” page and shows areas. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Store” use case.  **Business Rules:** N/A. | | | |

Table - <Admin> View Area

##### <Admin> Add Area

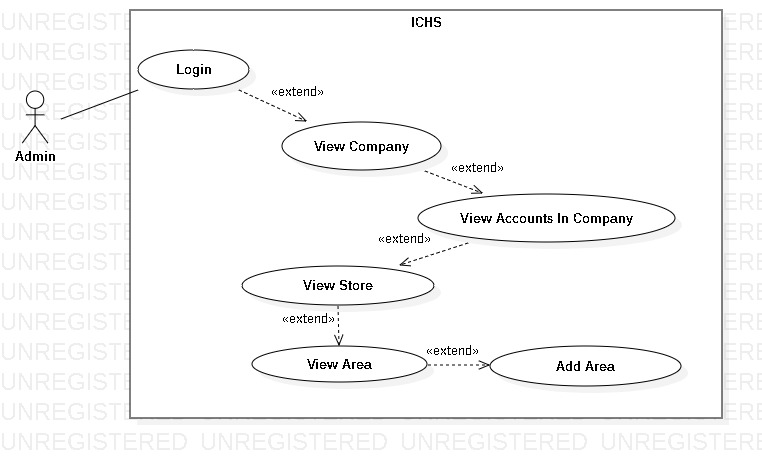


Figure 18- <Admin> Add Area

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_16** | | | |
| **Use Case No.** | 16 | **Use Case Version** | 1.0 |
| **Use Case Name** | Add Area | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to add area to store.   **Goal:**   * Admin add new area to store.   **Triggers:**   * Admin clicks “Add Area” button. * Admin sends add area command.   **Preconditions:**   * Admin login success. * Admin is at “Area” page.   **Post Conditions:**   * **Success:** Admin add area successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Add Area” button. | System displays “Add Area” page. Page contains 2 text fields and requires information:   * Area Floor: text input * Area Name: text input | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Add Area Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Area page. | | 2 | Area Floor is blank. | System displays “Please input Area Floor” message. | | 3 | Area Name is blank. | System displays “Please input Area Name” message. |   **Relationships:** Extend from “View Area” use case.  **Business Rules:**   * After add new area, admin return to Area page. | | | |

Table - <Admin> Add Area

##### <Admin> Edit Area

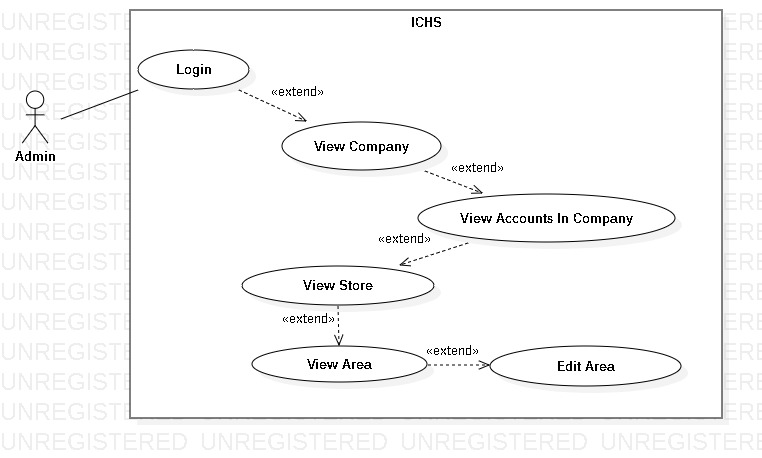


Figure 19- <Admin> Edit Area

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_17** | | | |
| **Use Case No.** | 17 | **Use Case Version** | 1.0 |
| **Use Case Name** | Edit Area | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to edit area.   **Goal:**   * Admin edit existing area.   **Triggers:**   * Admin clicks “Edit Area” button. * Admin sends edit area command.   **Preconditions:**   * Admin login success. * Admin is at “Area” page.   **Post Conditions:**   * **Success:** Admin edit area successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Edit Area” button. | System displays “Edit Area” page. Page contains 2 text fields and requires information:   * Area Floor: text input * Area Name: text input | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Edit Area Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Area page. | | 2 | Area Floor is blank. | System displays “Please input Area Floor” message. | | 3 | Area Name is blank. | System displays “Please input Area Name” message. |   **Relationships:** Extend from “View Area” use case.  **Business Rules:**   * After edit area , admin return to Area page. | | | |

Table - <Admin> Edit Area

##### <Admin> Delete Area

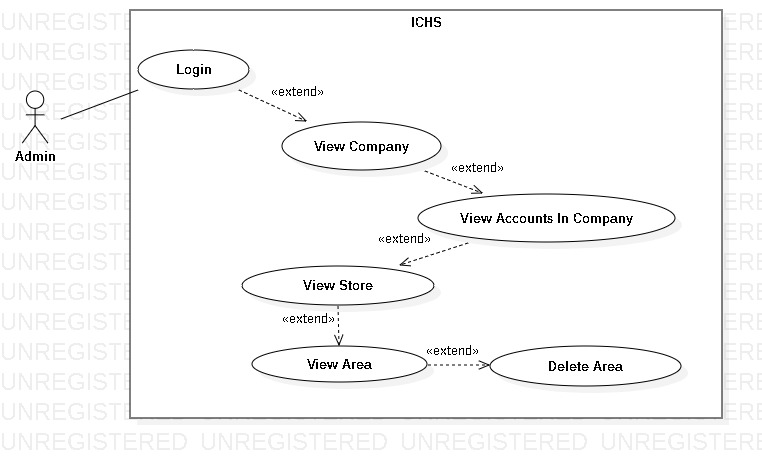


Figure 20- <Admin> Delete Area

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_18** | | | |
| **Use Case No.** | 18 | **Use Case Version** | 1.0 |
| **Use Case Name** | Delete Area | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to delete one area.   **Goal:**   * Admin delete existing area.   **Triggers:**   * Admin clicks “Delete Area” button. * Admin sends delete area command.   **Preconditions:**   * Admin login success. * Admin is at “Area” page.   **Post Conditions:**   * **Success:** Admin delete area successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Delete Area” button. | System displays “Do you want to delete this area?” message. | | 2 | Admin clicks “Ok” button | System removes the area from Area page. |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Area page. |   **Relationships:** Extend from “View Area” use case.  **Business Rules:**   * After delete area, admin return to Area page. | | | |

Table - <Admin> Delete Area

##### <Admin> View Camera

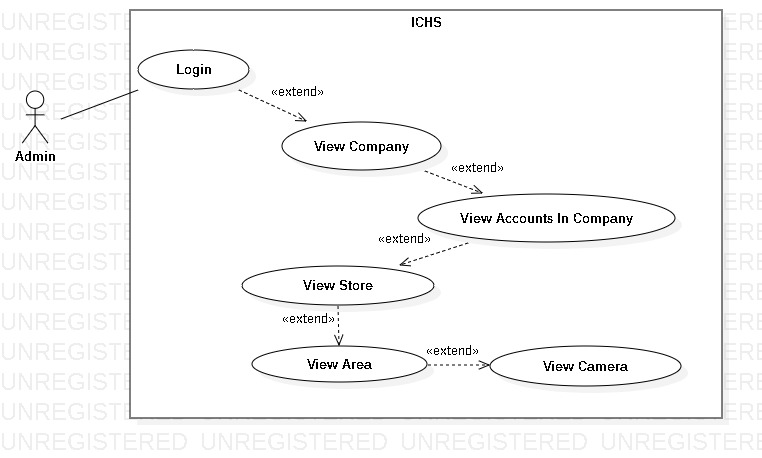


Figure 21- <Admin> View Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_19** | | | |
| **Use Case No.** | 19 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Camera | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to view all cameras in one area.   **Goal:**   * Admin view all cameras in the chosen area.   **Triggers:**   * Admin clicks “Camera” button. * Admin sends view camera command.   **Preconditions:**   * Admin login success. * Admin is at the chosen area page.   **Post Conditions:**   * **Success:** Admin view all cameras. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Camera” button in Area page. | System displays the “Camera” page and shows cameras. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Area” use case.  **Business Rules:** N/A. | | | |

Table - <Admin> View Camera

##### <Admin> Add Camera

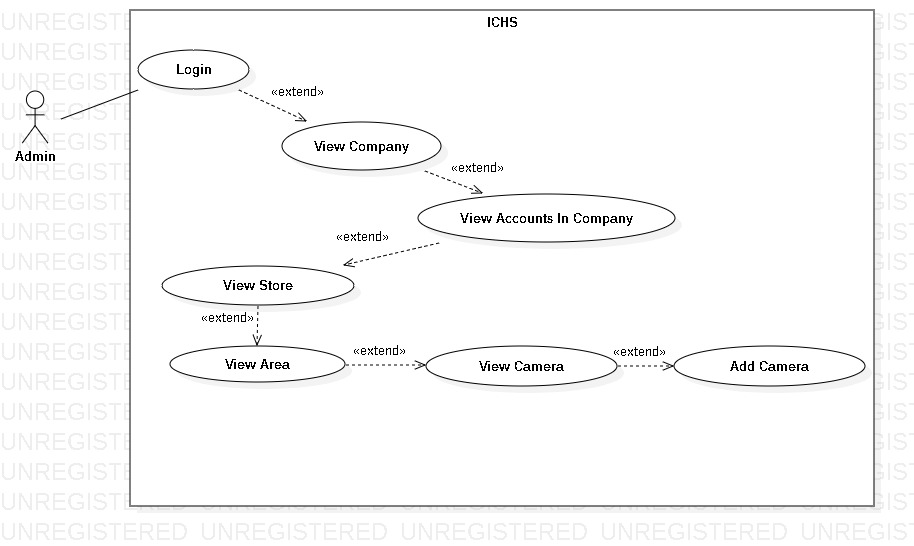


Figure 22- <Admin> Add Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_20** | | | |
| **Use Case No.** | 20 | **Use Case Version** | 1.0 |
| **Use Case Name** | Add Camera | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to add camera to area.   **Goal:**   * Admin add new camera to area.   **Triggers:**   * Admin clicks “Add Camera” button. * Admin sends add camera command.   **Preconditions:**   * Admin login success. * Admin is at “Camera” page.   **Post Conditions:**   * **Success:** Admin add camera successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Add Camera” button. | System displays “Add Camera” page. Page contains 3text fields and requires information:   * Camera IP: text input * Camera Account: text input * Camera Password: text input | | 2 | Admin inputs infomation and clicks “Ok” button | System displays message “Add Camera Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Camera page. | | 2 | Camera IP is blank. | System displays “Please input Camera IP” message. | | 3 | Camera Account is blank. | System displays “Please input Camera Account” message. | | 4 | Camera Password is blank. | System displays “Please input Camera Password” message. |   **Relationships:** Extend from “View Camera” use case.  **Business Rules:**   * After add new camera, admin return to camera page. | | | |

Table - <Admin> Add Camera

##### <Admin> Edit Camera

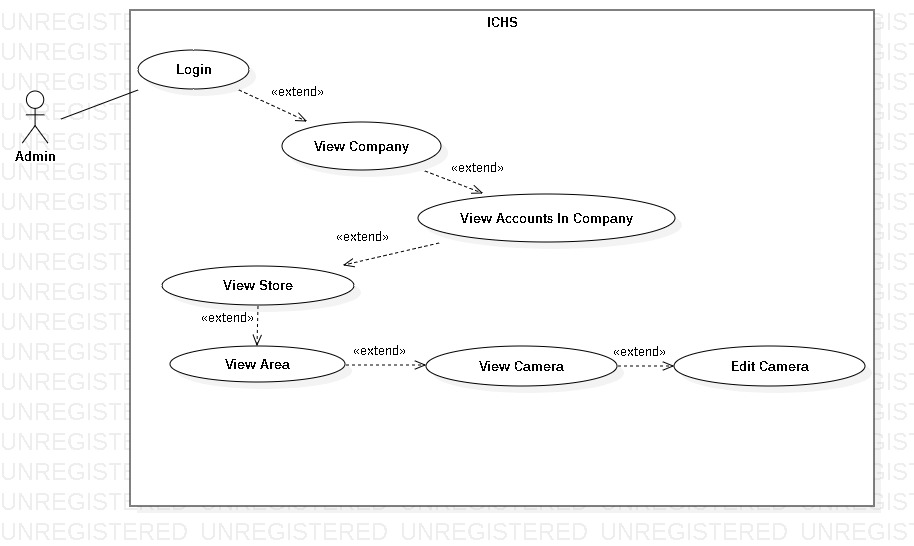


Figure 23- <Admin> Edit Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_21** | | | |
| **Use Case No.** | 21 | **Use Case Version** | 1.0 |
| **Use Case Name** | Edit Camera | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to edit camera.   **Goal:**   * Admin edit existing camera.   **Triggers:**   * Admin clicks “Edit Camera” button. * Admin sends edit camera command.   **Preconditions:**   * Admin login success. * Admin is at “Camera” page.   **Post Conditions:**   * **Success:** Admin edit camera successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Edit Camera” button. | System displays “Edit Camera” page. Page contains 3text fields and requires information:   * Camera IP: text input * Camera Account: text input * Camera Password: text input | | 2 | Admin inputs information and clicks “Ok” button | System displays message “Edit Camera Successfully !” |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Camera page. | | 2 | Camera IP is blank. | System displays “Please input Camera IP” message. | | 3 | Camera Account is blank. | System displays “Please input Camera Account” message. | | 4 | Camera Password is blank. | System displays “Please input Camera Password” message. |   **Relationships:** Extend from “View Area” use case.  **Business Rules:**   * After edit camera, admin return to Camera page. | | | |

Table - <Admin> Edit Camera

##### <Admin> Delete Camera

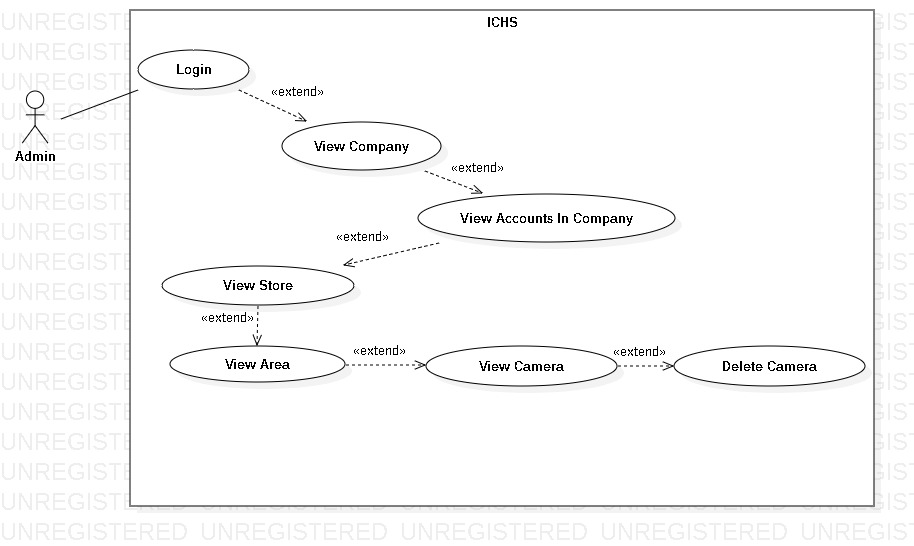


Figure 24- <Admin> Delete Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_22** | | | |
| **Use Case No.** | 22 | **Use Case Version** | 1.0 |
| **Use Case Name** | Delete Camera | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * Admin   **Summary:**   * This use case allows the admin to delete one camera.   **Goal:**   * Admin delete existing camera.   **Triggers:**   * Admin clicks “Delete Camera” button. * Admin sends delete area command.   **Preconditions:**   * Admin login success. * Admin is at “Camera” page.   **Post Conditions:**   * **Success:** Admin delete camera successfully. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin clicks “Delete Camera” button. | System displays “Do you want to delete this camera?” message. | | 2 | Admin clicks “Ok” button | System removes the camera from Camera page. |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin clicks “Cancel” button. | Back to Camera page. |   **Relationships:** Extend from “View Camera” use case.  **Business Rules:**   * After delete camera, admin return to Camera page. | | | |

Table - <Admin> Delete Camera

<User> Overview Use Case

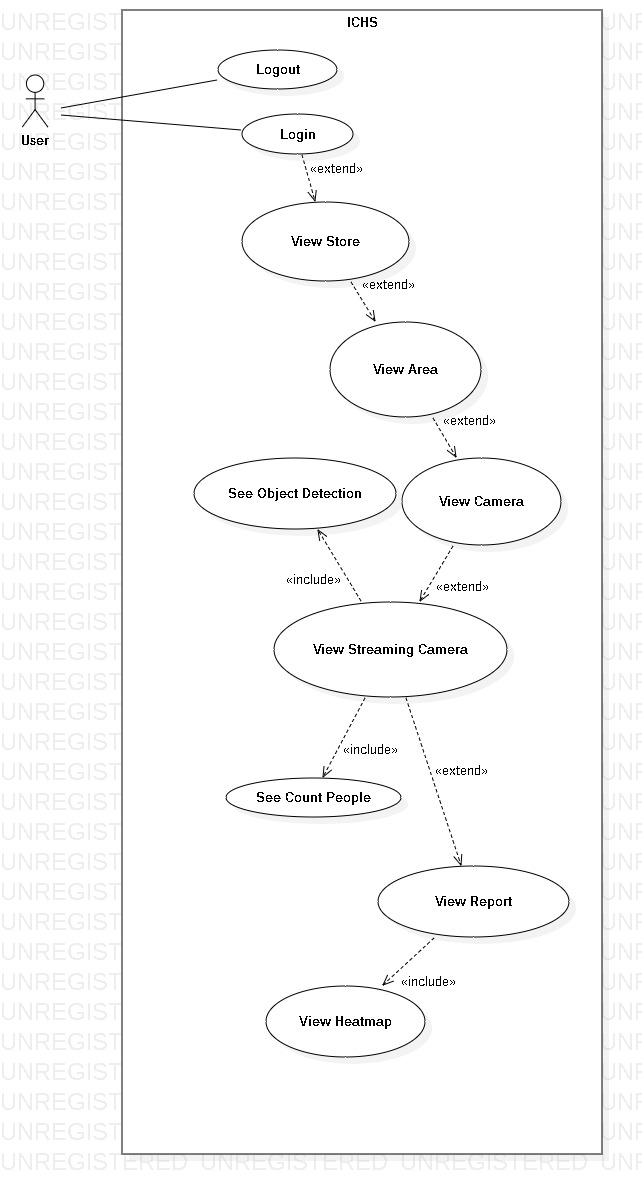


Figure 25- <User> Overview Use Case Diagram

##### <User> User Login

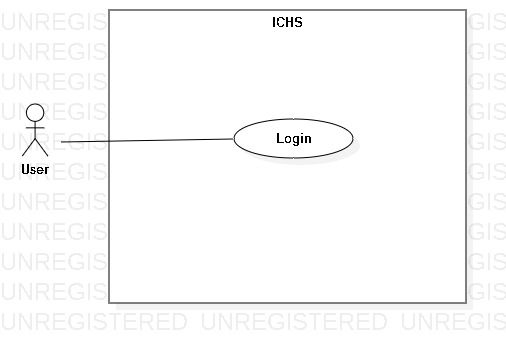


Figure 26- <User> User Login

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_23** | | | |
| **Use Case No.** | 23 | **Use Case Version** | 1.0 |
| **Use Case Name** | User Login | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user login to the system.   **Goal:**   * User logged in to the system.   **Triggers:**   * User clicks “Login” button. * User sends the login command.   **Preconditions:**   * User at “Login” page.   **Post Conditions:**   * **Success:** User login to the system. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User goes to Login page by clicking “Login” button. | System requires information:   * Username: text input. * Password: text input. | | 2 | User inputs information and clicks “Login” button. | System validates the information and displays confirmation message. |   **Alternative:** N/A.  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System displays “Login failed!” message. | | 2 | Username is blank. | System displays “Please input Username” message. | | 3 | Password is blank. | System displays “Please input Password” message. |   **Relationships:** N/A.  **Business Rules:**   * After logged in, system redirect to Home page. * A notification pop-up when user login success. * System must ensure has no duplicate user. * Password must be in rage of 6 – 30 characters. | | | |

Table - <User> User Login

##### <User> User Logout

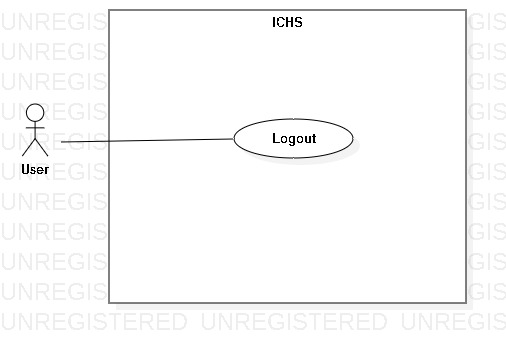


Figure 27- <User> User Logout

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_24** | | | |
| **Use Case No.** | 24 | **Use Case Version** | 1.0 |
| **Use Case Name** | User Logout | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user logout from application.   **Goal:**   * User logout from the system.   **Triggers:**   * User clicks Settings button. * User clicks “Log Out” button. * User sends the logout command.   **Preconditions:**   * User login success.   **Post Conditions:**   * **Success:** User logout from system. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Log Out” button. | System displays dialog with “You want to log out?” message. | | 2 | User clicks “Yes” button. | User logout from the system. |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | User clicks “Cancel” button. | Close the dialog. |   **Relationships:** N/A.  **Business Rules:**   * After logout from the system, system return Login page. * User can login with another account. | | | |

Table - <User> User Logout

##### <User> View Store

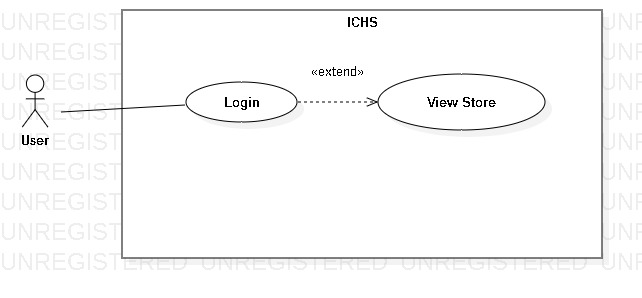


Figure 28- <User> View Store

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_25** | | | |
| **Use Case No.** | 25 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Store | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view all store in company that user is allowed to view.   **Goal:**   * User view all stores within the limits allowed by the admin.   **Triggers:**   * User clicks “Store” button. * User sends view store command.   **Preconditions:**   * User login success.   **Post Conditions:**   * **Success:** User view all stores. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Store” button in Home page. | System displays the “Store” page and shows stores. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “User Login” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Store

##### <User> View Area

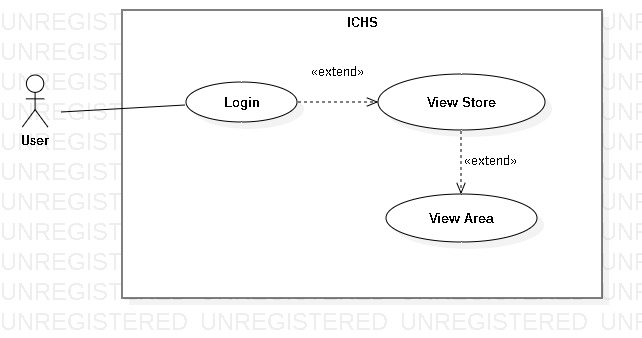


Figure 29- <User> View Area

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_26** | | | |
| **Use Case No.** | 26 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Area | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view all areas in one store.   **Goal:**   * User view all areas in the chosen store.   **Triggers:**   * User clicks “Area” button. * User sends view area command.   **Preconditions:**   * User login success. * User is at the chosen store page.   **Post Conditions:**   * **Success:** User view all areas. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Area” button in Store page. | System displays the “Area” page and shows areas. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Store” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Area

##### <User> View Camera

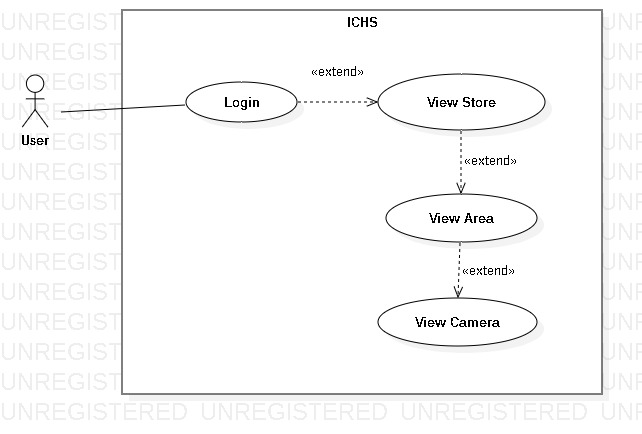


Figure 30- <User> View Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_27** | | | |
| **Use Case No.** | 27 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Camera | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view all cameras in one area.   **Goal:**   * User view all cameras in the chosen area.   **Triggers:**   * User clicks “Camera” button. * User sends view camera command.   **Preconditions:**   * User login success. * User is at the chosen area page.   **Post Conditions:**   * **Success:** User view all cameras. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Camera” button in Area page. | System displays the “Camera” page and shows cameras. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Area” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Camera

##### <User> View Streaming Camera

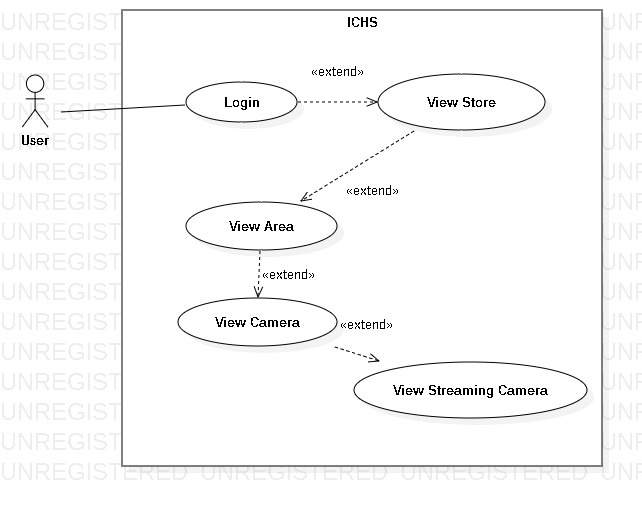


Figure 31- <User> View Streaming Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_28** | | | |
| **Use Case No.** | 28 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Streaming Camera | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view the streaming video from the selected camera.   **Goal:**   * User view streaming video from camera.   **Triggers:**   * User clicks the “Detail” button of the camera. * User sends view streaming camera command.   **Preconditions:**   * User login success. * User is at “Camera” page.   **Post Conditions:**   * **Success:** User view streaming camera. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Detail” button of the camera in the Camera page. | System displays the “Camera Detail” page and shows streaming video. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Camera” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Streaming Camera

##### <User> See Object Detection

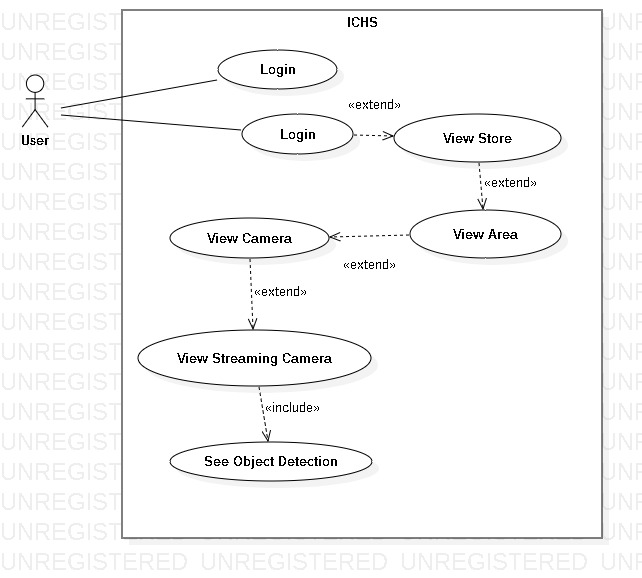


Figure 32- <User> See Object Detection

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_29** | | | |
| **Use Case No.** | 29 | **Use Case Version** | 1.0 |
| **Use Case Name** | See Object Detection | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to see the object detection in streaming video from the selected camera.   **Goal:**   * User see object detection in streaming video.   **Triggers:**   * User clicks the switch button in streaming video page. * User sends see object detection command.   **Preconditions:**   * User login success. * User is at “Camera Detail” page.   **Post Conditions:**   * **Success:** User see count people. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Object Detection” in Camera Detail page. | System displays the video with count people. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Camera” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Streaming Camera

##### <User> See Count People

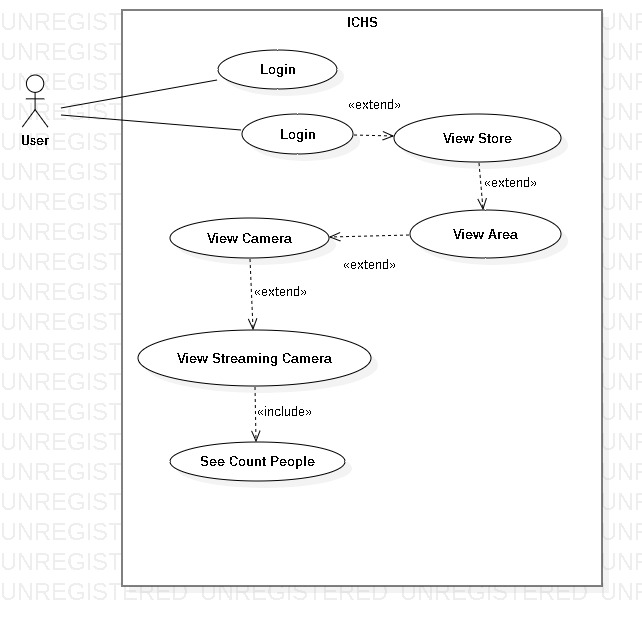


Figure 33- <User> See Count People

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_30** | | | |
| **Use Case No.** | 30 | **Use Case Version** | 1.0 |
| **Use Case Name** | See Count People | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to see the people counting in streaming video from the selected camera.   **Goal:**   * User see count people in streaming video.   **Triggers:**   * User clicks the switch button in streaming video page. * User sends see count people command.   **Preconditions:**   * User login success. * User is at ”Camera” page.   **Post Conditions:**   * **Success:** User view streaming camera. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Object Detection” in Camera Detail page. | System displays the video with count people. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Camera” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Streaming Camera

##### <User> View Report

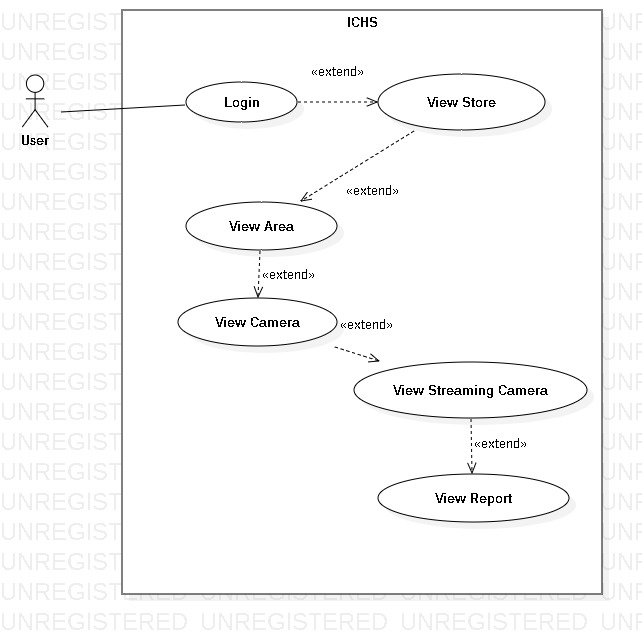


Figure 34- <User> View Report

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_31** | | | |
| **Use Case No.** | 31 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Report | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view the streaming video from the selected camera.   **Goal:**   * User view streaming video from camera.   **Triggers:**   * User clicks the thumbnail of the camera. * User sends view streaming camera command.   **Preconditions:**   * User login success. * User is at ”Camera” page.   **Post Conditions:**   * **Success:** User view streaming camera. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks thumbnail of the selected camera in Camera page. | System displays the selected camera page and shows streaming video. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Camera” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Streaming Camera

##### <User> View Heatmap

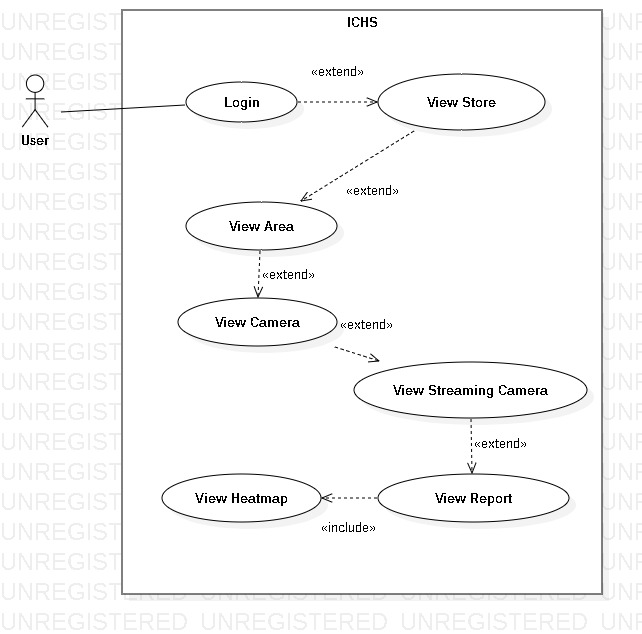


Figure 35- <User> View Heatmap

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_32** | | | |
| **Use Case No.** | 32 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Heatmap | | |
| **Author** | PhucDH | | |
| **Date** | 25/06/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view the streaming video from the selected camera.   **Goal:**   * User view streaming video from camera.   **Triggers:**   * User clicks the thumbnail of the camera. * User sends view streaming camera command.   **Preconditions:**   * User login success. * User is at “Camera” page.   **Post Conditions:**   * **Success:** User view streaming camera. * **Fail:** System displays error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks thumbnail of the selected camera in Camera page. | System displays the selected camera page and shows streaming video. |   **Alternative:** N/A.  **Exception:** N/A.  **Relationships:** Extend from “View Camera” use case.  **Business Rules:** N/A. | | | |

Table - <User> View Streaming Camera

## **3.****Software System Attribute**

* 1. Usability

- User Interface is simple and easy to understand.

- Provide a convenient way to interact with system.

- Only takes about 1 hour to learn how to use.

- User only need to click without input anything else.

* 1. Reliability

- The accuracy of counting people is about 75%.

- The accuracy of heatmap is about 80%.

* 1. Security

- User connects to internet via browser to use the Web Application.

- Web Application uses the internet to push data to server and save on cloud.

- The system upload to cloud every day at 12:00 PM.

* 1. Maintainability

- The code is easy to manage, maintain or upgrade.

- The system will be maintained on the last day of each month.

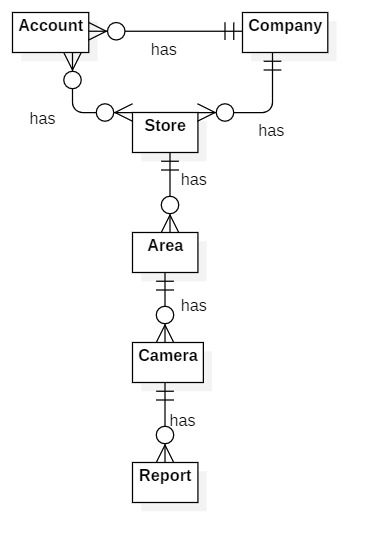
- There will be immediate support when an incident occurs.

* 1. Portability

- The system is running on Windows 7 or above.

* 1. Performance

## **Conceptual Diagram**



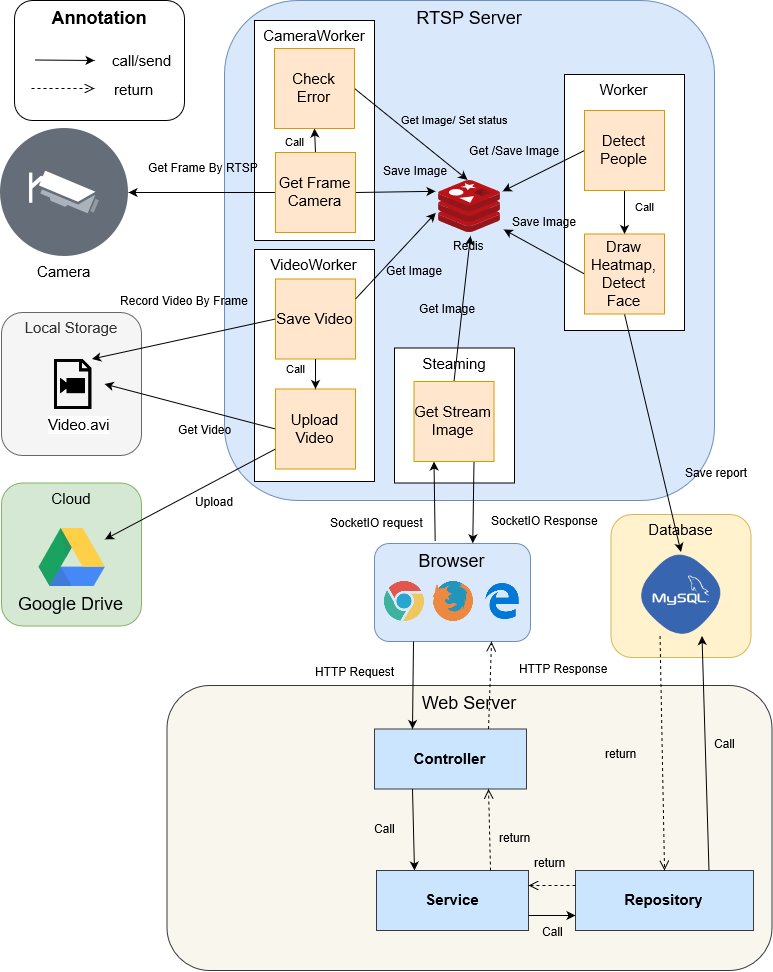
# **Software Design Description**

## **Design Overview**

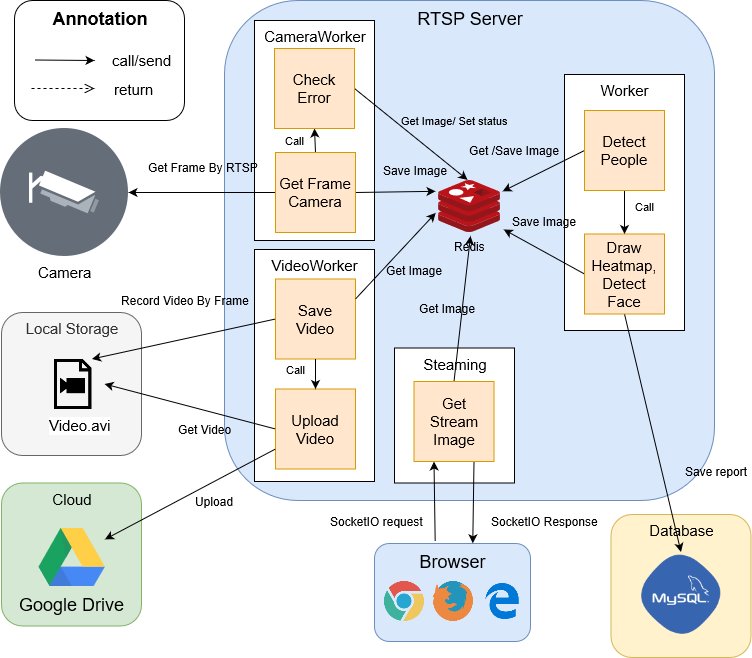
* This document describes the technical and user interface design of ICHS system. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
* Section 2: gives an overall description of the system architecture design.
* Section 3: gives component diagrams that describe the connection and integration of the system.
* Section 4: gives the detail design description, which includes class diagram, class explanation, and sequence diagram to details the application functions.
* Section 5: describe a fully attributed Entity Relationship Diagram.

## **System Architecture Design**

### **Overview**



### **Server Architecture Design**



### **Server Architecture Description**

This Server is multi-thread server because Server always gets image from camera by RTSP, detect people and send it to web application when it receives request at the same time.

We split RTSP server into 3 group with separate features and we call it is Worker:

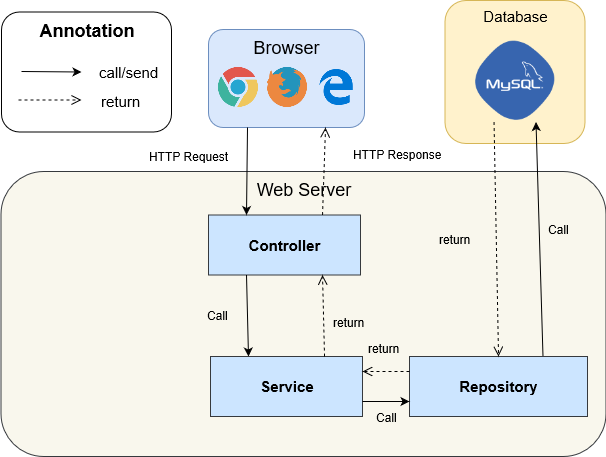
* CameraWorker: It is a group of features that interact with camera, it gets image from camera and check error when camera has incident.
* VideoWorker: It is a group of features that interact with video, it records video in local storage and upload it when new day is come.
* Worker: It if a group of features that use TensorFlow API to detect people, draw heatmap, detect face.

CameraWorker get image from camera and save it in redis by base64. When error is appeared, the latest image will save in error image in redis and check if current image is different with error image. If correct, it will set flag in redis and stop VideoWorker and Worker.

Worker will take image in redis and detect people in it and save detected image into redis. After 1 minute, it will call heatmap and face detection and send bounding boxes to draw get result face detection and save it in database.

VideoWorker get image from camera and record it to video and call upload video process at 00:00.

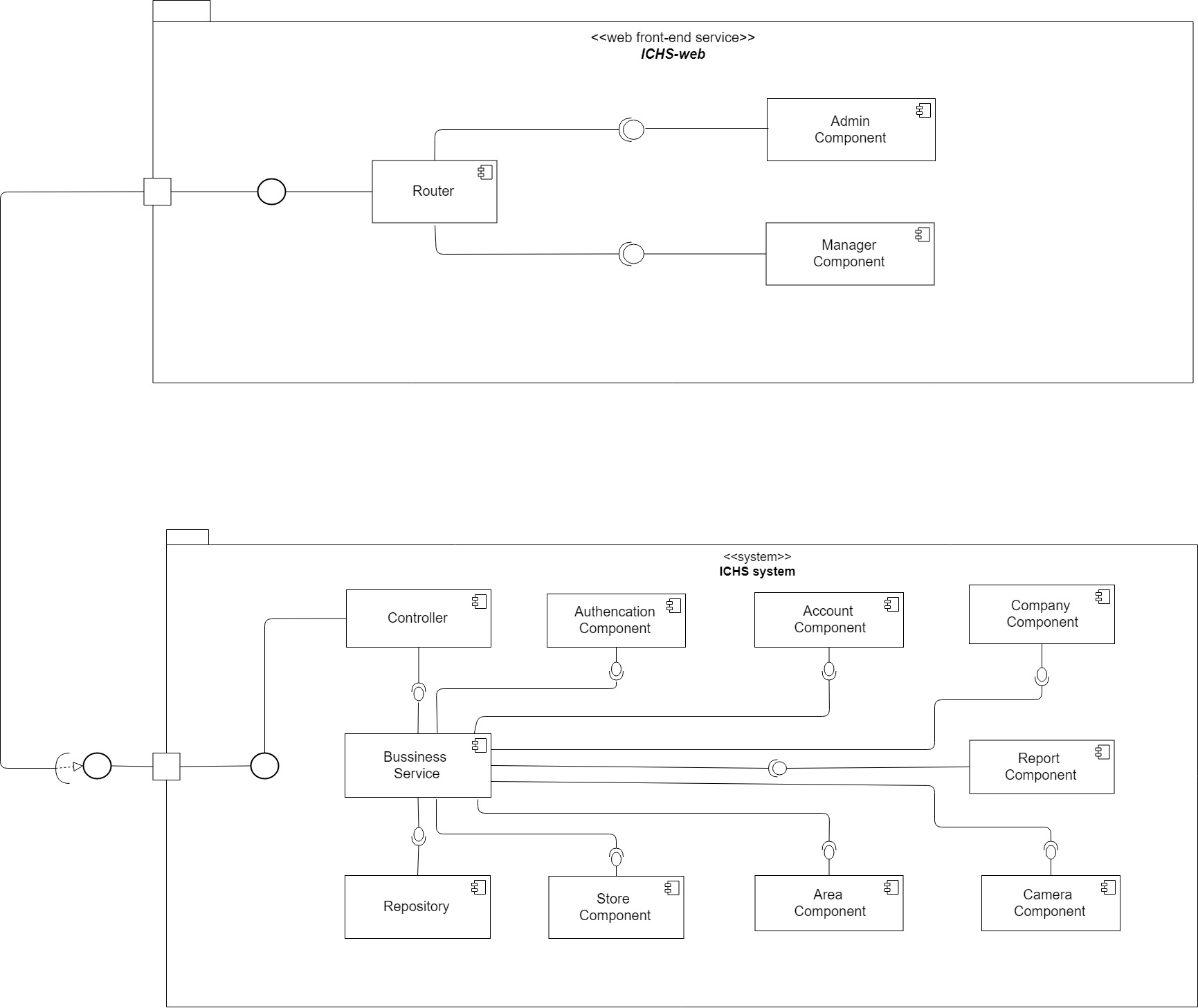
### **Web Application Architecture Design**



### **Web Application Architecture Description**

* API Web Server (Spring boot): the server handles all requests from front-end to query data from database and validate it then send http response to front-end.
* Database (MySQL Workbench 8.0 CE): this database store all data of system.
* Angular has component-based architecture that provides a higher quality of code. Encapsulation also ensures that new developers – who’ve been recently onboarded to a project – can read code better.

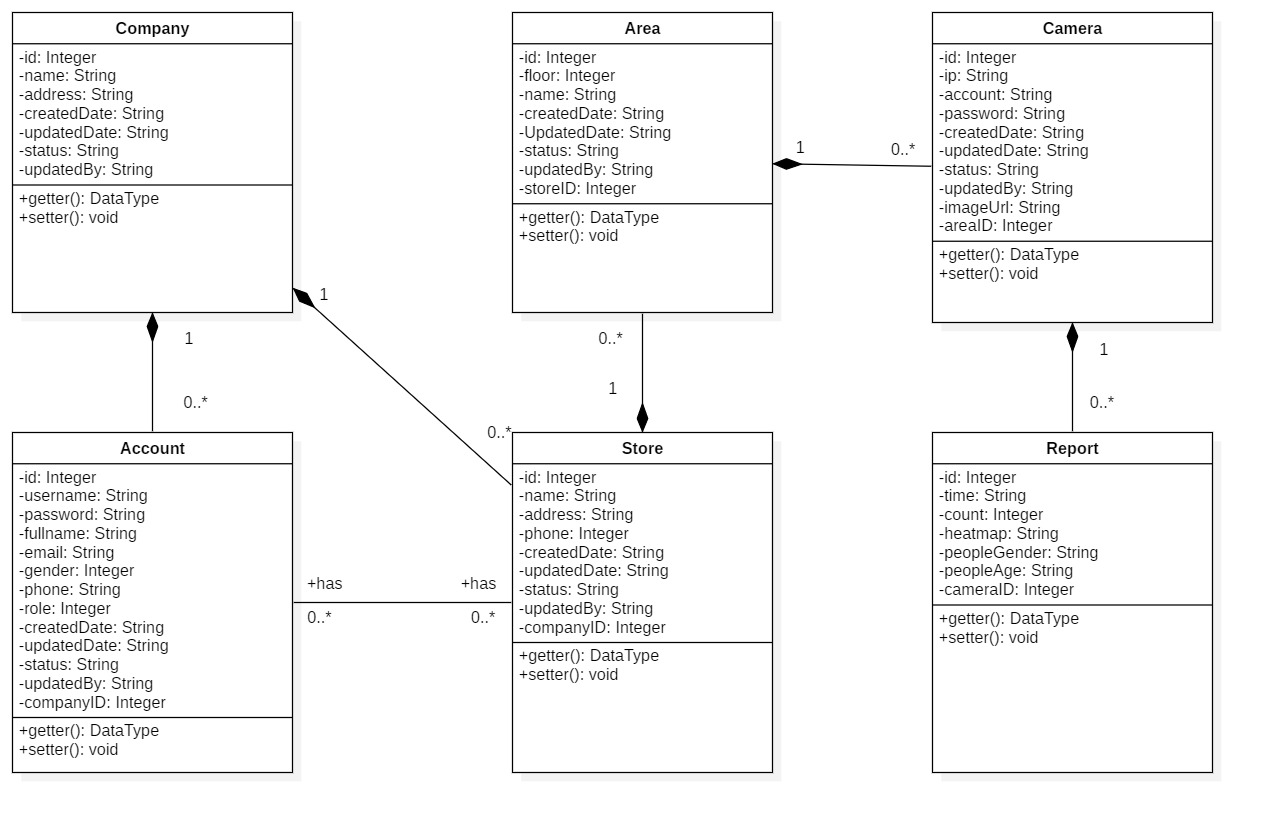
## **Component Diagram**



| **Components Dictionary: Describes components** | |
| --- | --- |
| Business Service | Component which handles business process for Web Sever. Include |
| Repository | Component which handles storing and retrieving data from database. |
| Controller | Component which handles request and response, accept input, convert it to commands for model and view. |
| Router | Handle request and response, accept input and convert it to commands for back-end |
| Admin Component | Handle admin’s activities in the system |
| Manager Component | Handle manager’s activities in the system |

## **Detailed Description**

* 1. Class Diagram



|  |  |  |
| --- | --- | --- |
| **CLASS DICTIONARY: DESCRIBE CLASS** | | |
| **Class Name** | **Mapping column with Conceptual diagram** | **Description** |
| **Company** | Company | Contains the information of Company |
| **Account** | Account | Contains the information of Account |
| **Store** | Store | Contains the information of Store |
| **Area** | Area | Contains the information of Area |
| **Camera** | Camera | Contains the information of Camera |
| **Report** | Report | Contains the information of Report |

* 1. Class Diagram Explanation
     1. Account

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | Integer | Private | Unique identifier of an account |
| Username | String | Private | Username of Account |
| Password | String | Private | Password of Account |
| Email | String | Private | Email of Account |
| Gender | Integer | Private | Gender of Account |
| Phone | Integer | Private | Phone number of Account |
| Role | Integer | Private | Role of Account |
| CreatedDate | String | Private | Date when account is created |
| UpdatedDate | String | Private | Date when account is updated |
| Status | String | Private | Status of account |
| UpdatedBy | String | Private | Which username updates account |
| CompanyID | Integer | Private | Contain information of the mentioned attribute |
| Method | **Return Type** | **Visibility** | **Description** |
| getter | DataType | Public | Get value for attribute |
| Setter | Void | Public | Set value for attribute |

* + 1. Company

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | Integer | Private | Unique identifier of a company |
| Name | String | Private | Name of company |
| Address | String | Private | Address of company |
| CreatedDate | String | Private | Date when company is created |
| UpdatedDate | String | Private | Date when company is updated |
| Status | String | Private | Status of company |
| UpdatedBy | String | Private | Which username updates company |
| Method | **Return Type** | **Visibility** | **Description** |
| getter | DataType | Public | Get value for attribute |
| Setter | Void | Public | Set value for attribute |

* + 1. Store

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | Integer | Private | Unique identifier of a store |
| Name | String | Private | Name of store |
| Address | String | Private | Address of store |
| Phone | Integer | Private | Phone of store |
| CreatedDate | String | Private | Date when store is created |
| UpdatedDate | String | Private | Date when store is updated |
| Status | String | Private | Status of store |
| UpdatedBy | String | Private | Which username updates store |
| CompanyID | Integer | Private | Contain information of the mentioned attribute |
| Method | **Return Type** | **Visibility** | **Description** |
| getter | DataType | Public | Get value for attribute |
| Setter | Void | Public | Set value for attribute |

* + 1. Area

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | Integer | Private | Unique identifier of an area |
| Floor | Integer | Private | Floor of area |
| Name | String | Private | Name of area |
| CreatedDate | String | Private | Date when area is created |
| UpdatedDate | String | Private | Date when area is updated |
| Status | String | Private | Status of area |
| UpdatedBy | String | Private | Which username updates area |
| StoreID | String | Private | Contain information of the mentioned attribute |
| Method | **Return Type** | **Visibility** | **Description** |
| getter | DataType | Public | Get value for attribute |
| setter | Void | Public | Set value for attribute |

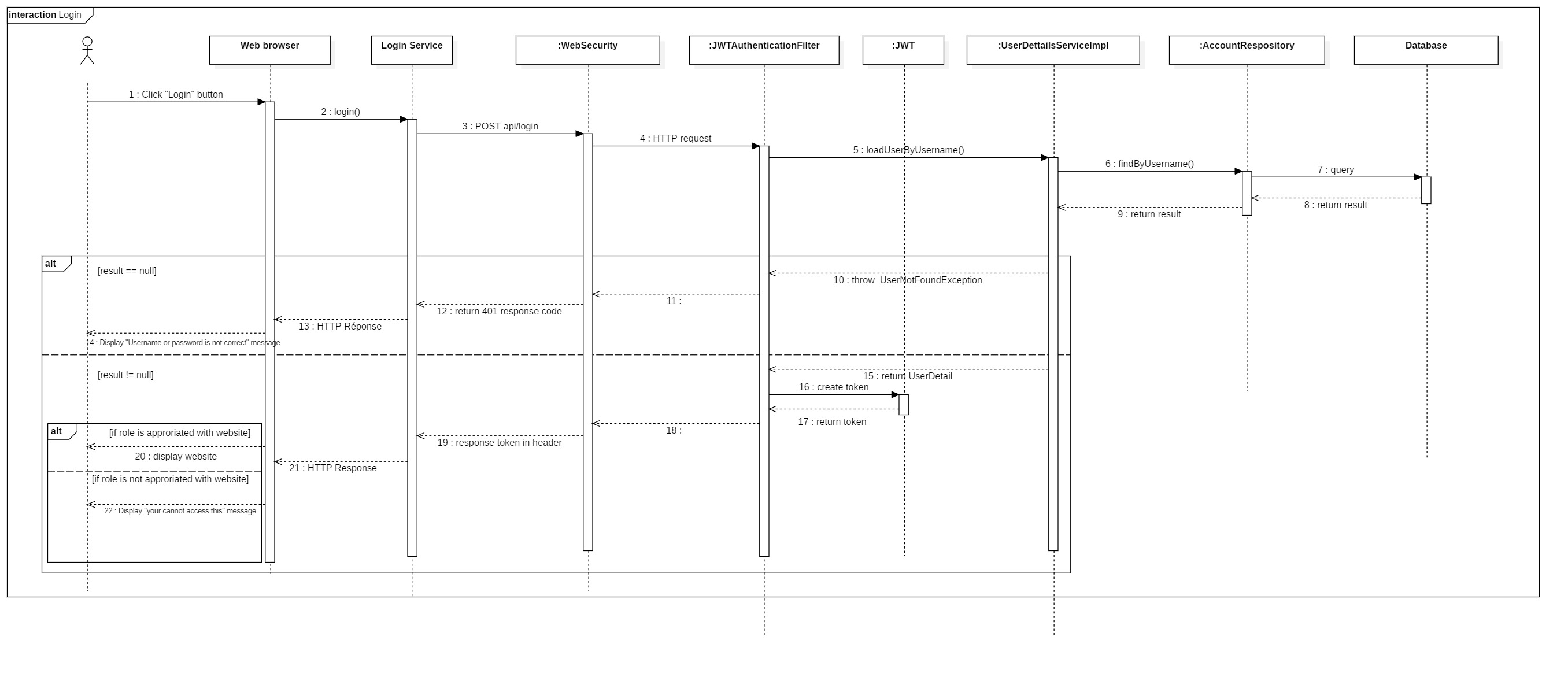
* + 1. Camera

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | Integer | Private | Unique identifier of a camera |
| Ip | String | Private | Ip of camera |
| Account | String | Private | Account of camera |
| Passsword | String | Private | Password of camera |
| CreatedDate | String | Private | Date when camera is created |
| UpdatedDate | String | Private | Date when camera is updated |
| Status | String | Private | Status of camera |
| ImageUrl | String | Private | Image url of camera |
| UpdatedBy | String | Private | Which username updates camera |
| AreaID | Integer | Private | Contain information of the mentioned attribute |
| Method | **Return Type** | **Visibility** | **Description** |
| getter | DataType | Public | Get value for attribute |
| Setter | Void | Public | Set value for attribute |

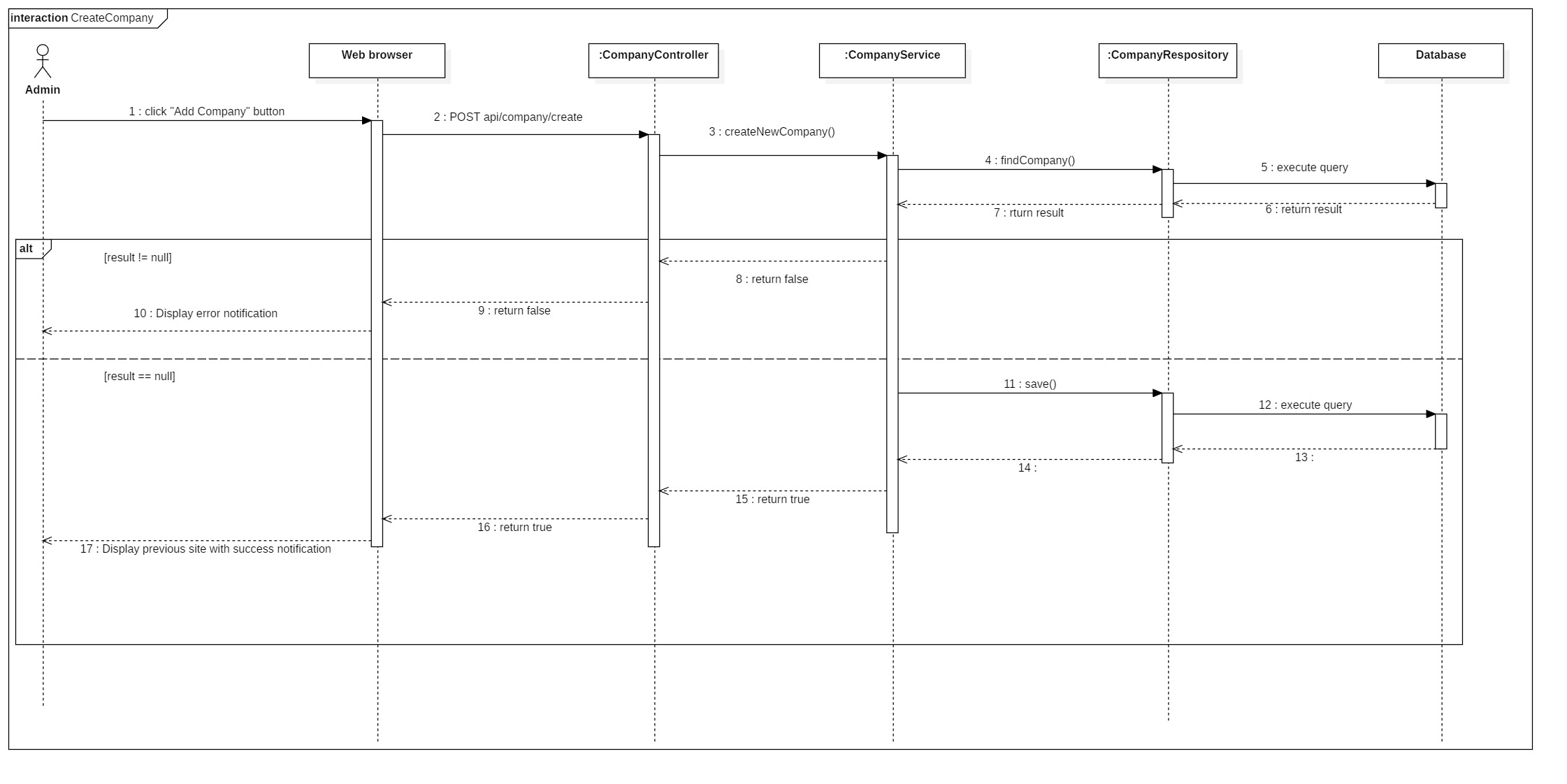
* + 1. Report

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| Id | Integer | Private | Unique identifier of a report |
| Time | String | Private | Time of report |
| Count | String | Private | Number of counted people |
| Heatmap | String | Private | The matrix heatmap of people |
| PeopleGender | String | Private | Number gender of people |
| PeopleAge | String | Private | Number age of people |
| CameraID | Integer | Private | Contain information of the mentioned attribute |
| Method | **Return Type** | **Visibility** | **Description** |
| getter | DataType | Public | Get value for attribute |
| Setter | Void | Public | Set value for attribute |

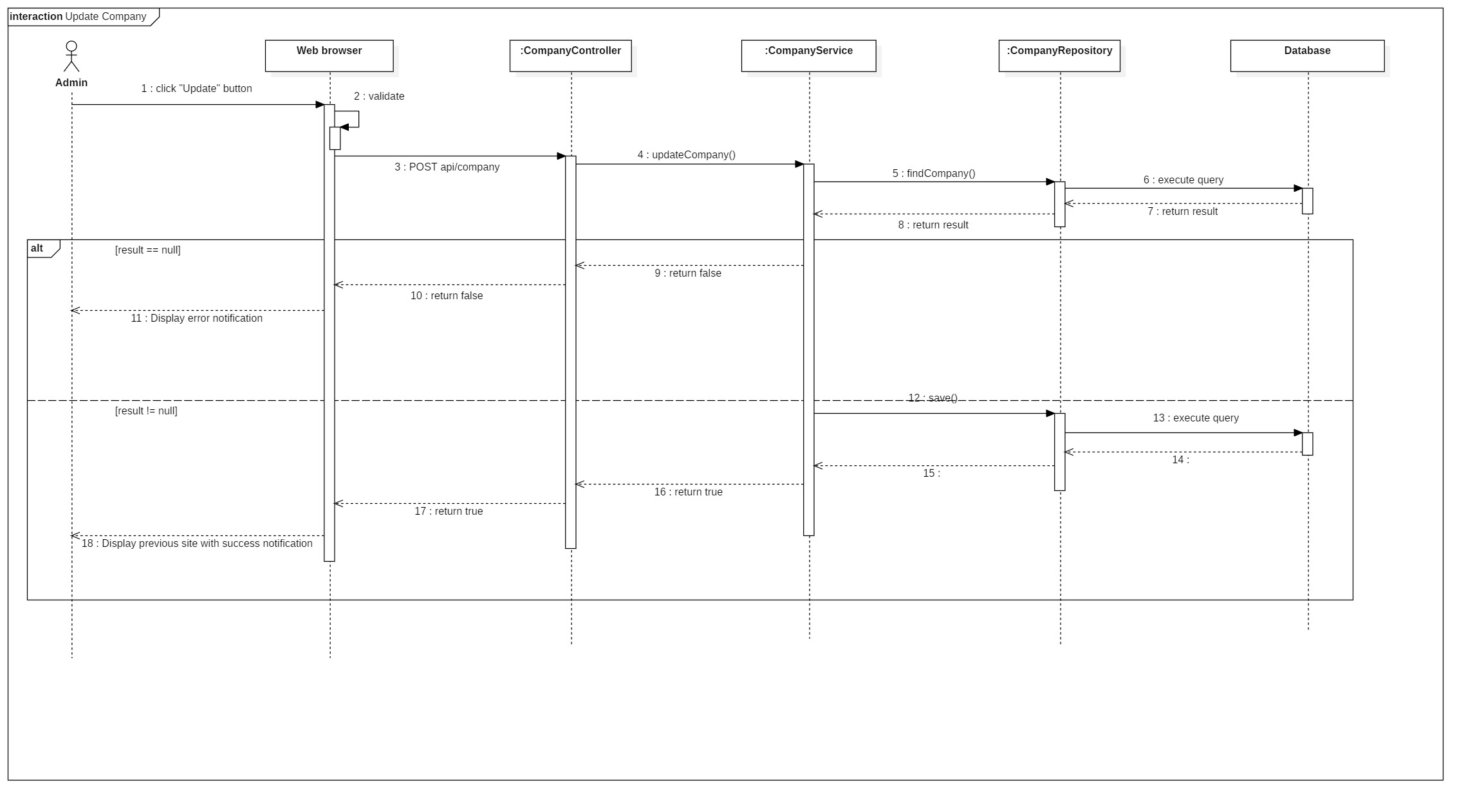
* 1. Interaction Diagram
     1. Login



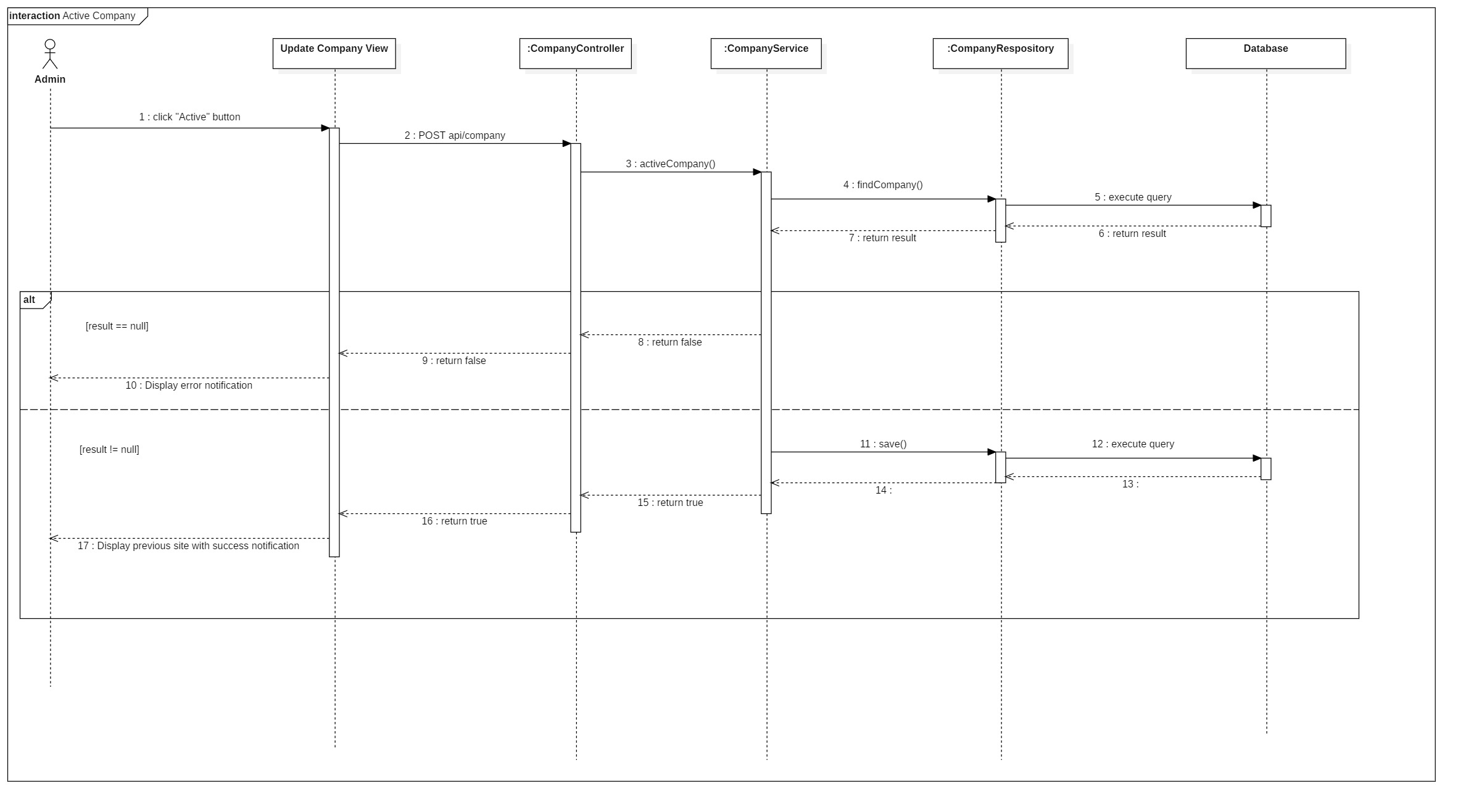
* + 1. Create Company



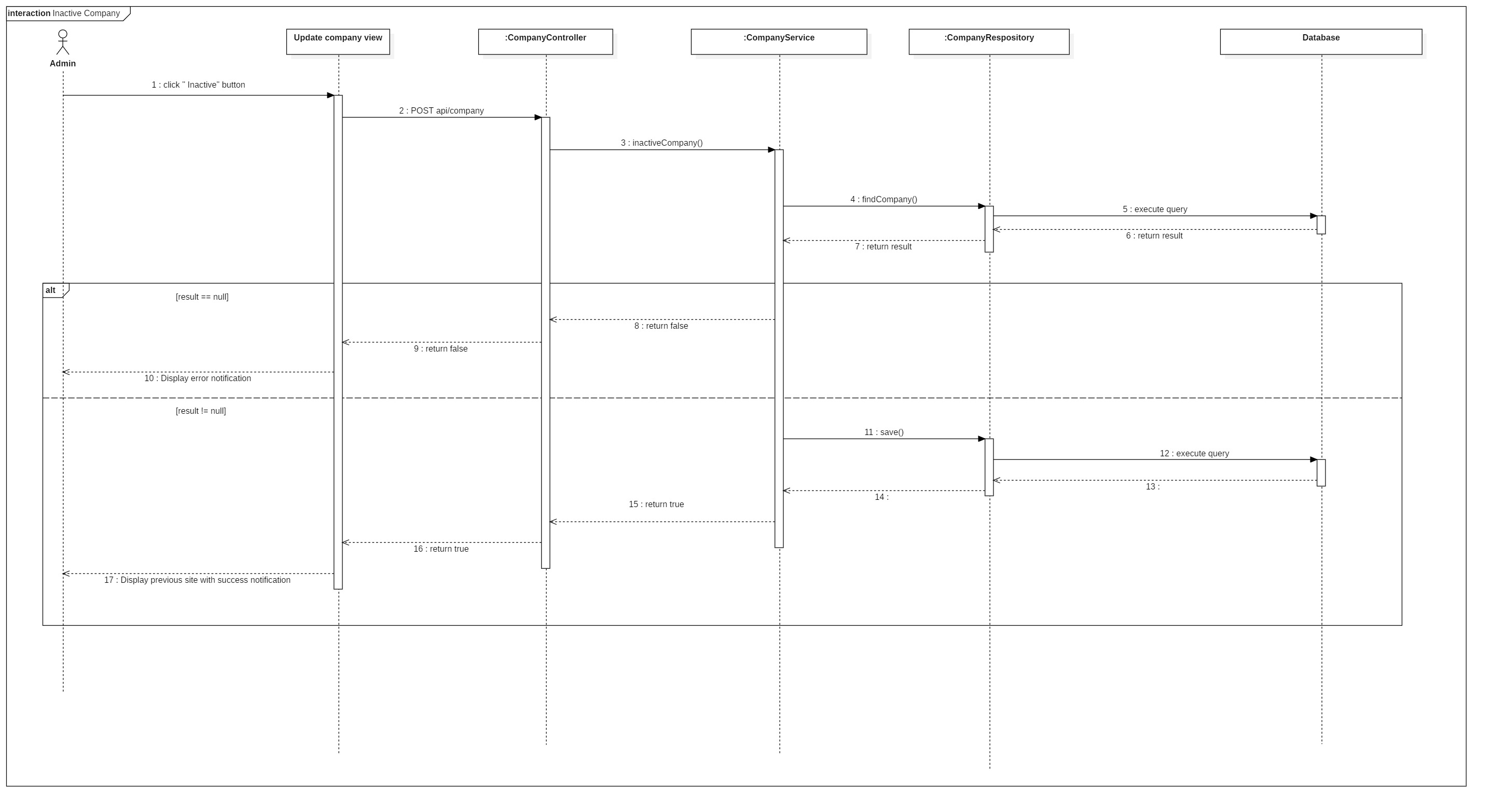
* + 1. Update Company



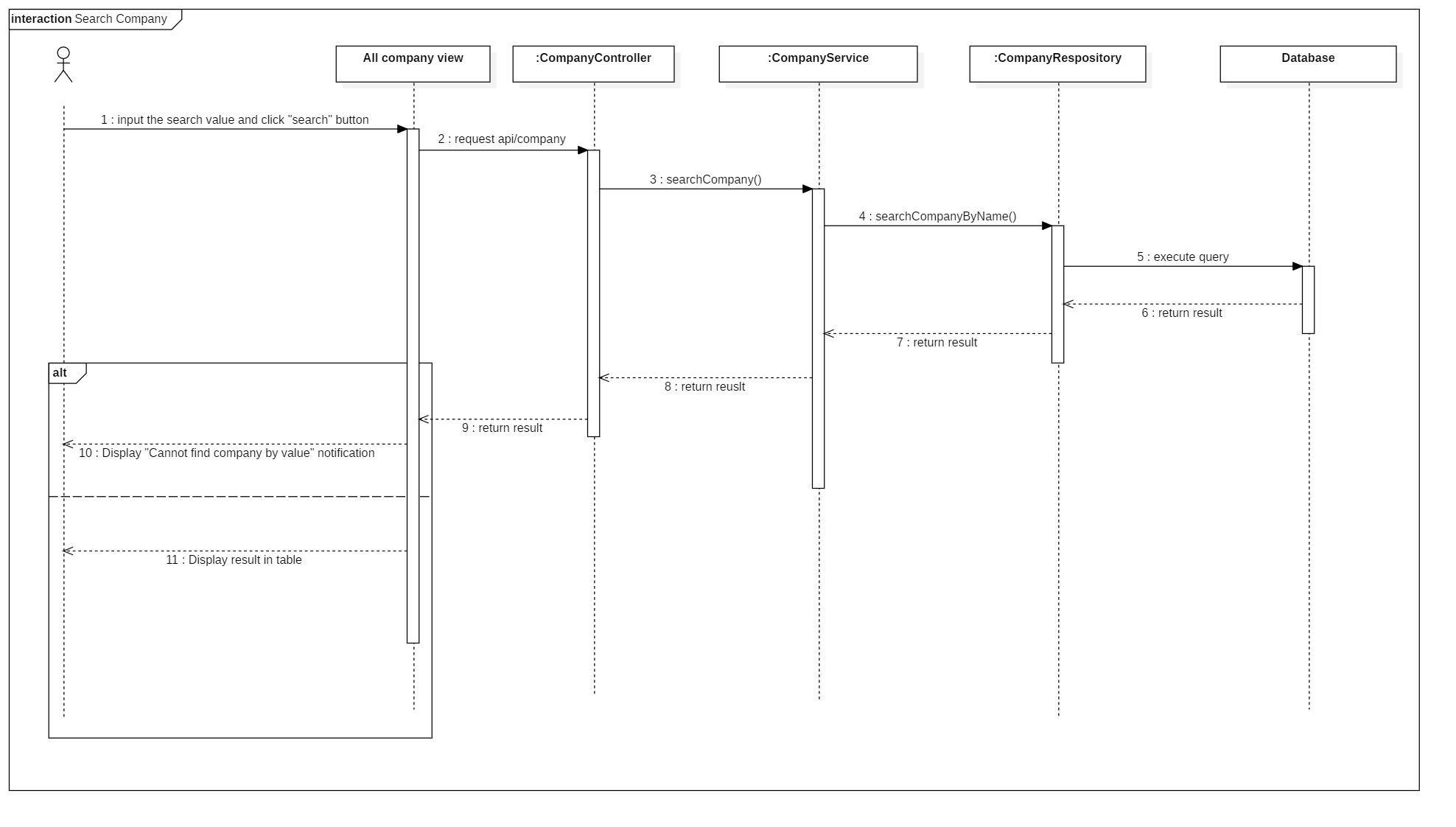
* + 1. Active Company



* + 1. Inactive Company



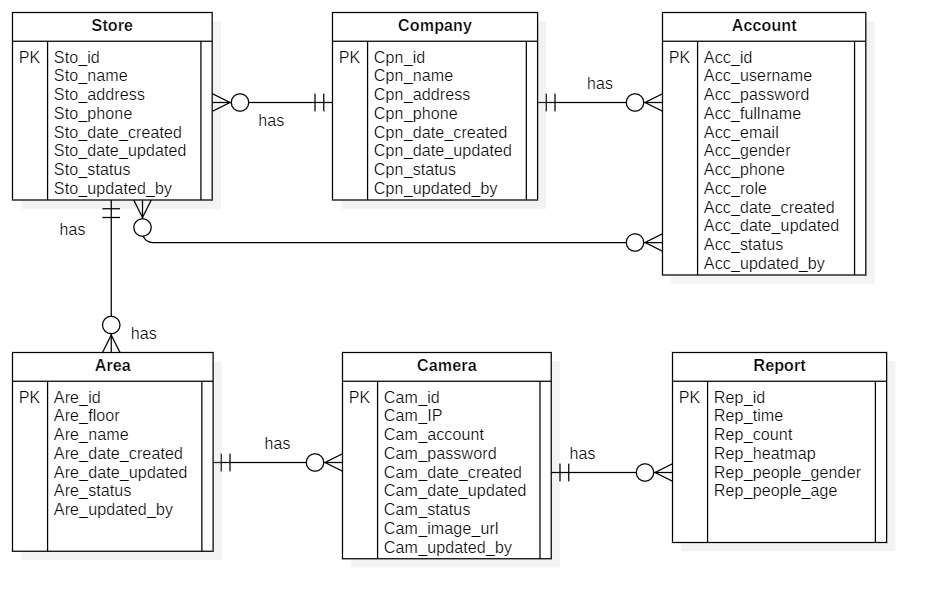
* + 1. Search Company



## **User Interface Design**

## **Database Design**

* 1. Entity Relationship Diagram



* 1. Entity Dictionary

## **Algorithms**

### **Faster RCNN**

* + 1. ***Definition***

Faster RCNN is one of the most well-known objects detection networks which have 3 neural networks (Feature Network, Region Proposal Network, Detection Network). It is very useful for detecting object.

* + 1. ***Define problems***

We don’t have experience of training AI for detecting people. Because people is object that needed detect so It need a big data set to train for many posture. So we decide to use TensorFlow API that use this algorithms.

* + 1. ***Solution***

we decide to use TensorFlow API that use this algorithm.

Faster RCNN is Algorithms that TensorFlow API is using for detect object in image.





Steps:

1. Take image to Region Proposal Network (RPN) to get Region Proposal which has ability that contain object.
2. By RPN, we can define which box have object and by ROI Pooling to convert image to fixed size image.
3. After get fixed image, classify object in image and return 2 result: class which is type of object and bounding boxes.

* **Region Proposal Network**

Input is image and output are Region Proposal which is rectangle. Instead of using (x center, y center, width, height), RPN detect Region Proposal by using Anchor and create anchor box with 4 parameters (x min, y min, x max, y max)





### **Draw Heatmap**

* + 1. ***Definition***

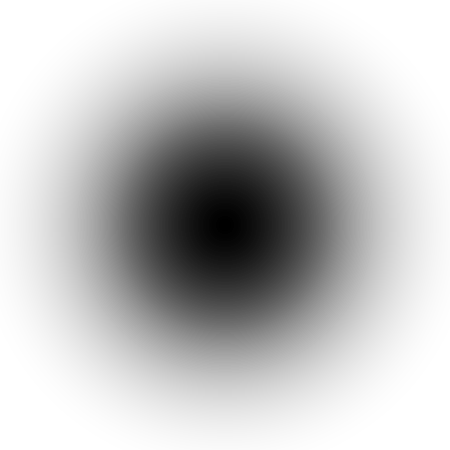
Using LinearSegmentedColormap to create color map and use it to define what color based on grayscale image.

* + 1. ***Define problem***

Color must be changed based on long people stay in camera

* + 1. ***Solution***

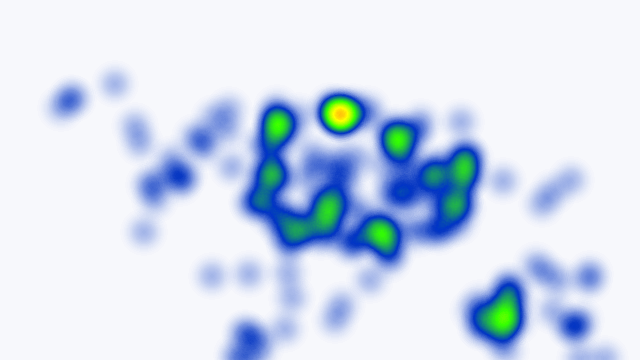
Use dot image to draw on grayscale image.



And create a heatmap color image in local storage.



We will set list of dots in a white image that it will become grayscale image and change its color based on how black it is.

# **System Implementation & Test**

## **Introduction**

* 1. Overview

This section describes the approach and methodologies used by group to plan, organize and manage the testing of ICHS. It provides in detail all necessary information about the implementation and testing procedure of the system included test plans, test cases, test result, test environments, pass/fail criteria and risks estimations as well as a checklist to cover all possible cases.

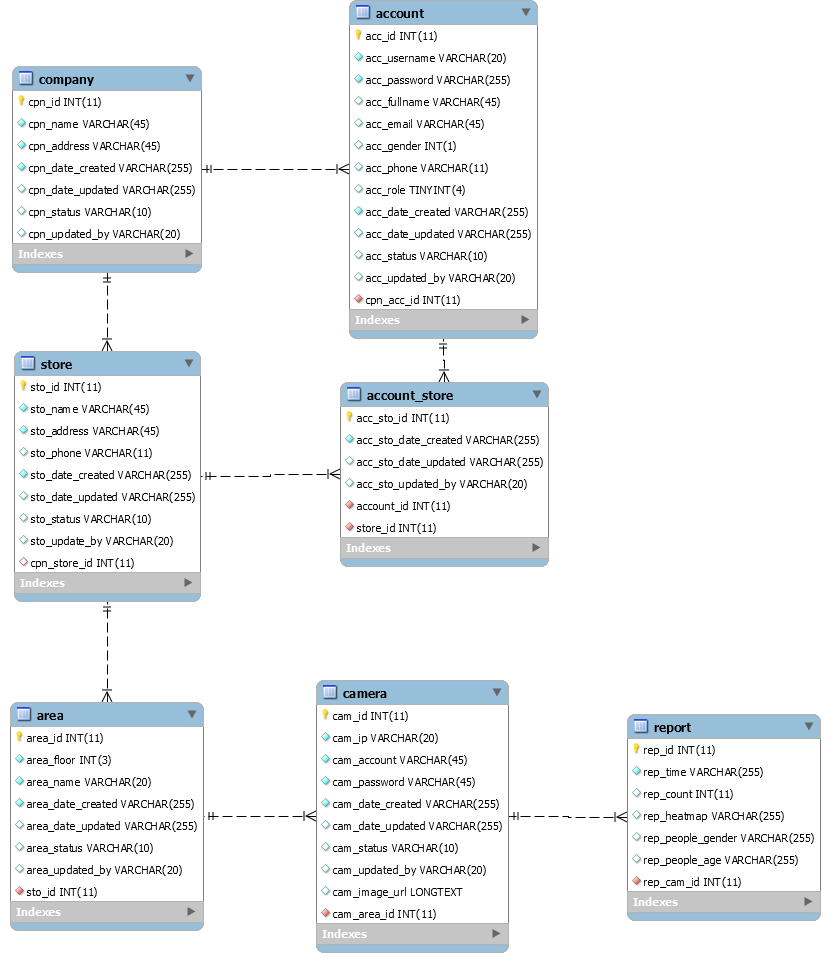
* 1. Test Approach
* **Goal**: Test all features in the whole VHT application based on the core flow.
* **Method**: Black-box testing.
* **Technique**: Check list.

The testing for this project consists of Integration System test level. Testing the program which was integrated and as a complete system to ensure that the software requirements have been met.

* Integration testing is performed by all member of team and approved by team leader.
* System testing is focused on assessing the system’s reliability. This process is concerned with finding errors that result from unanticipated interactions between components and component interface problems.

## **Database Relationship Diagram**

* 1. Physical Diagram



* 1. Data Dictionary

|  |  |
| --- | --- |
| DATA DICTIONARY: DESCRIBE CONTENT OF ALL TABLES | |
| Table name | **Description** |
| Company | Contains the company information. |
| Account | Contains the account information. |
| Store | Contains the store information. |
| Area | Contains the area information. |
| Camera | Contains the camera information. |
| Report | Contains the report information. |
| Account\_Store | Contains relationship of account and store. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | Attributes | Description | Domain | Nulls |
| Account | acc\_id | ID of account | int(11) | No |
| acc\_username | Username of account | varchar(20) | No |
| acc\_password | Password of account | varchar(255) | No |
| acc\_fullname | Fullname of account | varchar(45) | Yes |
| acc\_email | Email of account | varchar(45) | Yes |
| acc\_gender | Gender of account | int(1) | Yes |
| acc\_phone | Phone of account | varchar(11) | Yes |
| acc\_role | real name of user | tinyint(4) | Yes |
| acc\_date\_created | Date when account is created | varchar(255) | No |
| acc\_date\_updated | Date when account is updated | varchar(255) | Yes |
| acc\_status | Status of account | varchar(10) | Yes |
| acc\_updated\_by | Username updated account | varchar(20) | Yes |
| cpn\_acc\_id | ID of company | int(11) | No |
| Company | cpn\_id | ID of admin | int(11) | No |
| cpn\_name | Name of company | varchar(45) | No |
| cpn\_address | Address of company | varchar(45) | No |
| cpn\_date\_created | Date when company is created | varchar(255) | No |
| cpn\_date\_updated | Date when company is updated | varchar(255) | Yes |
| cpn\_status | Status of company | varchar(10) | Yes |
| cpn\_updated\_by | Username updated company | varchar(20) | Yes |
| Store | sto\_id | ID of store | int(11) | No |
| sto\_name | Name of store | varchar(45) | No |
| sto\_address | Address of store | varchar(45) | No |
| sto\_phone | Phone of store | varchar(11) | Yes |
| sto\_date\_created | Date when store is created | varchar(255) | No |
| Sto\_date\_updated | Date when store is updated | varchar(255) | Yes |
| sto\_status | Status of store | varchar(10) | Yes |
| sto\_updated\_by | Username updated store | varchar(20) | Yes |
| cpn\_store\_id | ID of store | int(11) | Yes |
| Area | area\_id | ID of area | int(11) | No |
| area\_floor | Floor of area | int(3) | No |
| area\_name | Name of area | varchar(36) | No |
| area\_date\_created | Date when area is created | varchar(255) | No |
| area\_date\_updated | Date when area is created | varchar(255) | Yes |
| area\_status | Status of area | varchar(10) | Yes |
| area\_updated\_by | Username updated area | varchar(20) | Yes |
| sto\_id | ID of store | int(11) | No |
| Camera | cam\_id | ID of camera | int(11) | No |
| cam\_ip | Ip of camera | varchar(20) | No |
| cam\_account | Account of camera | varchar(45) | No |
| cam\_password | Password of camera | varchar(45) | No |
| cam\_date\_created | Date when camera is created | varchar(255) | No |
| cam\_date\_updated | Date when camera is updated | varchar(255) | Yes |
| cam\_status | Status of camera | varchar(10) | Yes |
| cam\_updated\_by | Username updated camera | varchar(20) | Yes |
| cam\_image\_url | Image url of camera | text | Yes |
| cam\_area\_id | ID of area | int(11) | No |
| Report | rep\_id | ID of report | int(11) | No |
| rep\_time | Time of report | varchar(255) | No |
| rep\_count | Count number of report | int(11) | Yes |
| rep\_heatmap | Heatmap matrix of report | varchar(255) | Yes |
| rep\_people\_gender | Gender of people of report | varchar(255) | Yes |
| rep\_people\_age | Age of people of report | varchar(255) | Yes |
| rep\_cam\_id | ID of camera | int(11) | No |
| Account\_Store | acc\_sto\_id | ID of account\_store | int(11) | No |
| acc\_sto\_date\_created | Date when the account\_store is created | varchar(255) | No |
| acc\_sto\_date\_updated | Date when the account\_store is updated | varchar(255) | Yes |
| acc\_sto\_updated\_by | Username updated account\_store | varchar(20) | Yes |
| account\_id | ID of account | int(11) | No |
| store\_id | ID of store | int(11) | No |

## **Test Plan**

* 1. Features to be Tested
  2. Features not to be Tested

## **Testing Test Case**

* 1. Test Case
  2. ....

# **Software User’s Manual**

## **Installation Guide**

* 1. Hardware Requirement
  2. Software Requirement

## **User Guide**

# **F1.ソフトウェアユーザーマニュアル**

# **Appendix**