**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

# **CAPSTONE PROJECT REGISTER**

(\*) Profession: <Software Engineer> Specialty: <ES> <IS> <JS>

x

(\*) Kinds of person make registers: Lecturer Students

x

## **Register information for supervisor (if have)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor 1 | Lâm Hữu Khánh Phương | 0915353001 | [phuonglhk@fpt.edu.vn](mailto:phuonglhk@fpt.edu.vn) | Mr. |

## **Register information for students (if have)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| Student 1 | Vũ Tấn Huy | SE62172 | 0909679977 | huyvtse62176@fpt.edu.vn | Leader |
| Student 2 | Đỗ Quốc Cường | SE62573 | 0904635563 | cuongdqse62573@fpt.edu.vn | Member |
| Student 3 | Nguyễn Quang Tuyến | SE62069 | 0969694629 | tuyennqse62069@fpt.edu.vn | Member |
| Student 4 | Đinh Hoàng Phúc | SE61768 | 0764038298 | phucdhse61768@fpt.edu.vn | Member |

## **Register content of Capstone Project**

### **Capstone Project name:**

* English: In-Store Customer Heatmap System
* Vietnamese: Xây dựng biểu đồ nhiệt khách hàng trong cửa hàng.
* Abbreviation: ICHS

### **Main proposal content (including result and product)**

1. Theory and practice (document):

* Student should apply the software development process and UML 2.0 in modelling system.
* The documents include User Requirement, Software Requirement Specification, Architecture Design, Detail Design, System Implementation and Testing Document, Installation Guide, sources code, and deployable software packages
* Server side technologies:
  + Server: .NET, Java or Javascript, Windows Azure…
  + Database Design: SQL Server or MySQL.
* Client side technologies:
  + Web Client: HTML5, CSS3, JavaScript.

1. Program:

Build a system with features:

* Feature 1: Cloud server for storing video data from store
* Feature 2: Evaluate and implement people counting algorithm on video
* Feature 3: Measure store traffic in real time and show on heat map (update data every 30s)
* Feature 4: Reporting data: number of passersby from the zone, average shopping time, Historical reports at hourly & daily intervals.

Hardware and software: camera support RTSP protocol, Windows or Linux server.

## **Other comment (propose all relative thing if have)**

There is no records kept about shopper‘s identity without their permissions, so no privacy concerns on this solutions.

# **INTRODUCTION**

1. **Project Information**

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

- Project Code: **ICHS**

- Project Type: **Web Application**

- Start Date: 03/05/19

- End Date: 27/08/19

1. **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.

1. **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.

1. **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.

1. **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

1. **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.

1. **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 PROCESS MODEL**

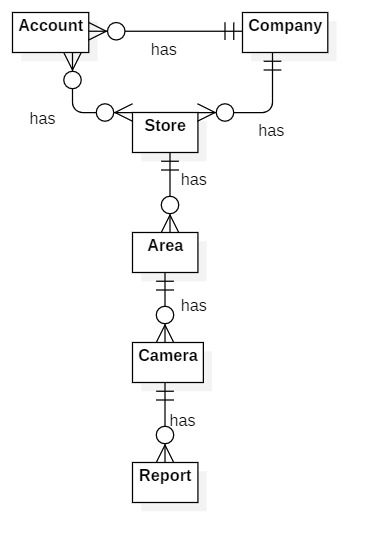
The software process model used in developing is based on the Waterfall model.

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.

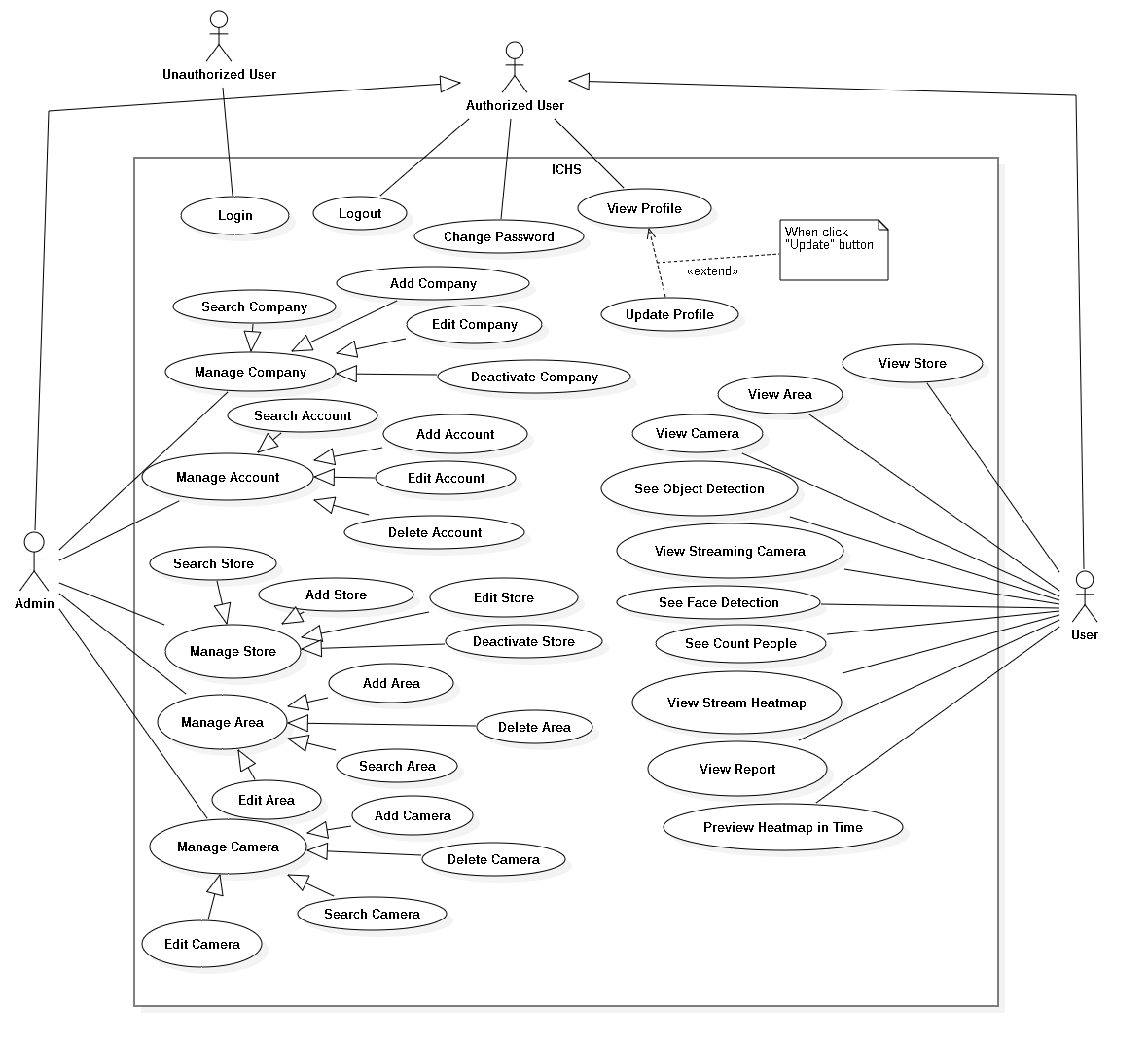


# **CONCEPTUAL DIAGRAM**



|  |  |
| --- | --- |
| **Entity Data dictionary: describe all content of all entities** | |
| **Entity Name** | **Description** |
| Account | Contains the account information. |
| Company | Contains the company information. |
| Store | Contains the store information. |
| Area | Contains the area information. |
| Camera | Contains the camera information. |
| Report | Contains the report information. |

# **USE CASE DIAGRAM**



# **USE CASE SPECIFICATION**

## **<Unauthorized User> Login**

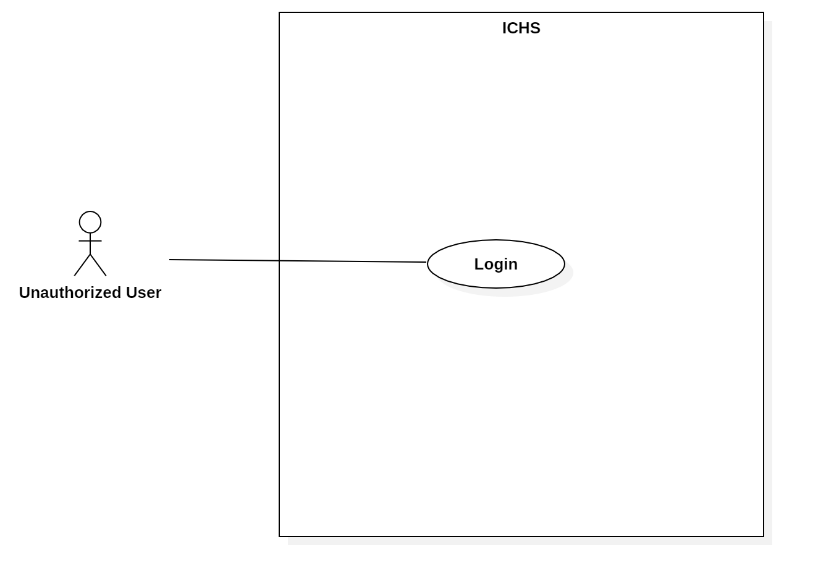


Figure : <Unauthorized User> Login

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_01** | | | |
| **Use Case No.** | 01 | **Use Case Version** | 1.0 |
| **Use Case Name** | Login | | |
| **Author** | HuyVT | | |
| **Date** | 08/08/2018 | **Priority** | Normal |
| **Actor:**   * Unauthorized user   **Summary:**   * This use case allows the unauthorized log in the system.   **Goal:**   * Unauthorized logged in the system.   **Triggers:**   * Unauthorized clicks “Đăng nhập” button. * Unauthorized sends the login command.   **Preconditions:**   * Unauthorized at “Đăng nhập” page.   **Post Conditions:**   * **Success:** Unauthorized login the system. * **Fail:** System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Unauthorized goes to Login page by clicking “Đăng nhập” button. | System requires information:   * Username: free text input. * Password: free text input. | | 2 | Unauthorized inputs information. |  | | 3 | Unauthorized clicks “Đăng nhập” button. | System validates the information and shows confirmation message.  [Exception 1,2,3] |   **Alternative:** N/A.  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows “Đăng nhập không thành công!” message. | | 2 | Username is blank. | System shows “Tên tài khoản rỗng” message. | | 3 | Password is blank. | System shows “Mật khẩu rỗng” message. |   **Relationships:** N/A.  **Business Rules:**   * After logged in, system redirect to main page. * A notification pop-up when user login success. * System must ensure has no duplicate user. * Password must be greater than 6. | | | |

Table : <Unauthorized User> Login

## **<Authorized User> Logout**



Figure : <Authorized User> Logout

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_02** | | | |
| **Use Case No.** | 02 | **Use Case Version** | 1.0 |
| **Use Case Name** | User Logout | | |
| **Author** | HuyVT | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the authorized user logout from application.   **Goal:**   * Authorized user logout from the system.   **Triggers:**   * Authorized user clicks Settings button. * Authorized user clicks “Log Out” button. * Authorized user sends the logout command.   **Preconditions:**   * Authorized login success.   **Post Conditions:**   * **Success:** Authorized logout from system. * **Fail:** System shows 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 : <Authorized User> Logout

## **<Authorized User> Change Password**

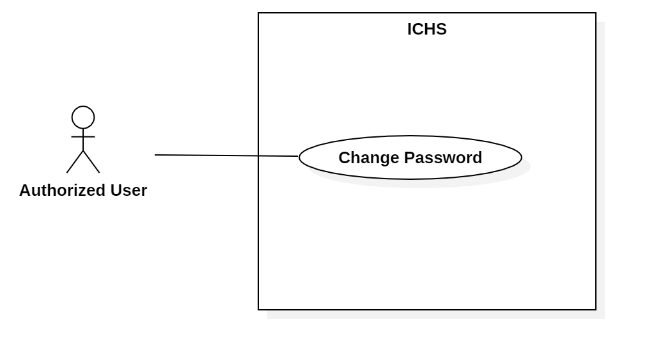


Figure : <Authorized User> Change Password

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_03** | | | |
| **Use Case No.** | 03 | **Use Case Version** | 1.0 |
| **Use Case Name** | Change password | | |
| **Author** | HuyVT | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the authorized user logout from application.   **Goal:**   * Authorized user logout from the system.   **Triggers:**   * Authorized user clicks “Change password” button. * Authorized user sends the logout command.   **Preconditions:**   * Authorized login success.   **Post Conditions:**   * **Success:** Authorized logout from system. * **Fail:** System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Change password” button. | System shows change password page with request information in form:  + Old password: text input  + Password: text input  + Confirm password: text input | | 2 | User inputs password that need to change. |  | | 3 | User clicks “Change” button. | System validates the information and shows result in the main page.  [Exception 1, 2, 3, 4, 5] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Network is problem. | System shows error message. | | 2 | Old password is blank | System shows “Old password is required!” message. | | 3 | Password is blank | System shows “Password is required!” message. | | 4 | Password length is not greater than 6 | System shows “Min length < 6” message. | | 5 | Confirm password are not equal with password | System shows “The passwords are not equal” message. |   **Relationships:** N/A.  **Business Rules:**   * New password must be greater than 6. * After change password, system shows “change password success” message and return dashboard page. * All field: old password, password, confirm password not blank. | | | |

Table : <Authorized User> Change Password

## **<Authorized User> View Profile**

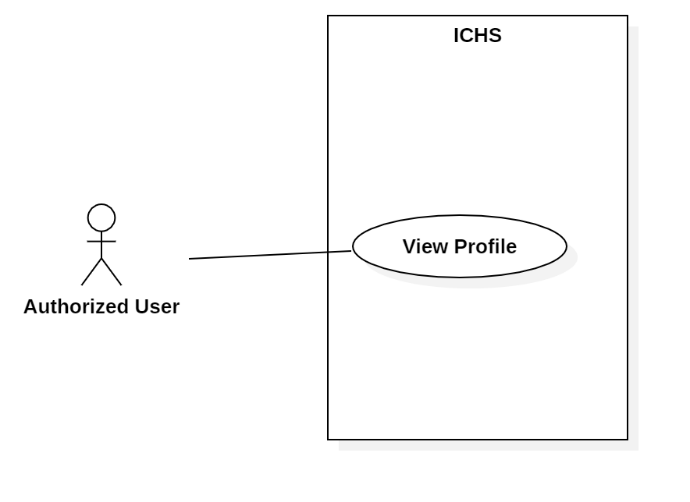


Figure : <Authorized User> View Profile

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_04** | | | |
| **Use Case No.** | 04 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Profile | | |
| **Author** | HuyVT | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the authorized user can view profile.   **Goal:**   * Authorized user check profile.   **Triggers:**   * Authorized user clicks “Edit profile” button. * Authorized user sends the logout command.   **Preconditions:**   * Authorized login success.   **Post Conditions:**   * **Success:** Authorized user can view profile. * **Fail:** System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Edit profile” button. | System displays profile page with request information in form:  + Username: text input (not edit)  + Fullname: text input  + Gender: male or female  + Email: text input  + Phone: text input  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Network has problem | System shows error message. |   **Relationships:** N/A.  **Business Rules: N/A.** | | | |

Table : <Authorized User> View Profile

## **<Authorized User> Update Profile**

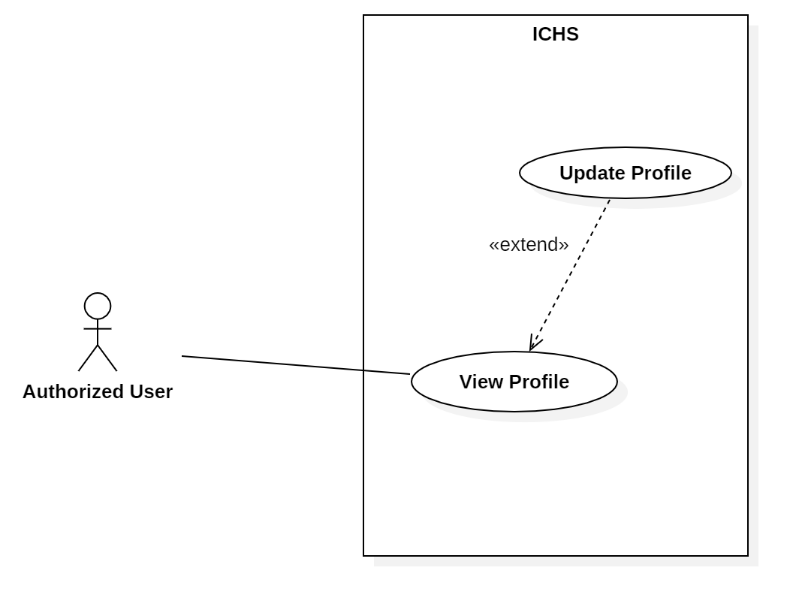


Figure : <Authorized User> Update Profile

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_05** | | | |
| **Use Case No.** | 05 | **Use Case Version** | 1.0 |
| **Use Case Name** | Update Profile | | |
| **Author** | HuyVT | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the authorized user can update information of the profile.   **Goal:**   * Authorized user update profile.   **Triggers:**   * Authorized user clicks “Edit profile” button. * Authorized user clicks “Update” button. * Authorized user sends the logout command.   **Preconditions:**   * Authorized login success.   **Post Conditions:**   * **Success:** Authorized user update profile success. * **Fail:** System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks “Edit profile” button. | System displays profile page with request information in form:  + Username: text input (not edit)  + Fullname: text input  + Gender: male or female  + Email: text input  + Phone: text input | | 2 | User inputs information that need to update. |  | | 3 | User clicks “Update” button. | System validates the information and shows confirmation message.  [Exception 1, 2, 3, 4] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Network has problem | System shows error message. | | 2 | Fullname is blank | System shows “Fullname is required!” message. | | 3 | Email is blank | System shows “Email is required!” message. | | 4 | Phone is blank | System shows “Phone is required!” message. |   **Relationships:** Extend from “View profile” use case.  **Business Rules: N/A.**   * After update, system return the dashboard page. * Phone must be 11 number. * Username not edit by user, only edit when admin ‘s role. | | | |

Table : <Authorized User> Update Profile

## **<User> View Store**

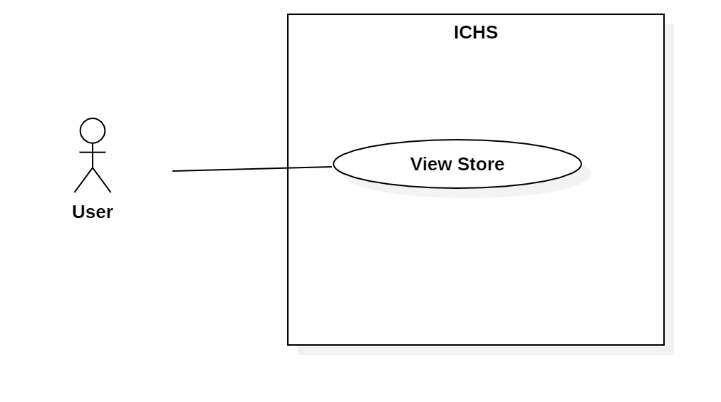


Figure : <User> View Store

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_26** | | | |
| **Use Case No.** | 26 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Store | | |
| **Author** | HuyVT | | |
| **Date** | 08/08/2018 | **Priority** | Normal |
| **Actor:**   * User   **Summary:**   * This use case allows the user view store of company.   **Goal:**   * User view all stores within the limits allowed by the admin.   **Triggers:**   * User clicks “Store” button.   **Preconditions:**   * User login successful with account provided by the admin.   **Post Conditions:**   * **Success:** show all store with account user. * **Fail:** System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User goes to dashboard page and click “store” button. | System show all store with account user.  [Exception 1] |   **Alternative:** N/A.  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A.  **Business Rules:** N/A. | | | |

Table : <User> View Store

## **<User> View Area**

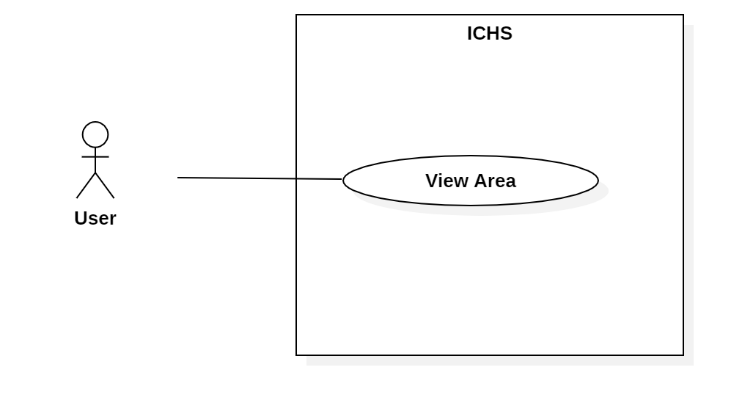


Figure : <User> View Area

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_27** | | | |
| **Use Case No.** | 27 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Area | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view all areas in one store.   **Goal:**   * User view all areas in one store.   **Triggers:**   * User clicks “Store” button. * User clicks “View Area” button. * User sends view area command.   **Preconditions:**   * User login successful with account provided by the admin. * User is at the chosen store page.   **Post Conditions:**   * **Success:** User view all areas. * **Fail:** System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User goes to dashboard page and click “store” button. | System show all store with account user. | | 2 | User clicks “Area” button in Store page. | System displays the “Area” page and shows areas.  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A.  **Business Rules:** N/A. | | | |

Table : <User> View Area

## **<User> View Camera**

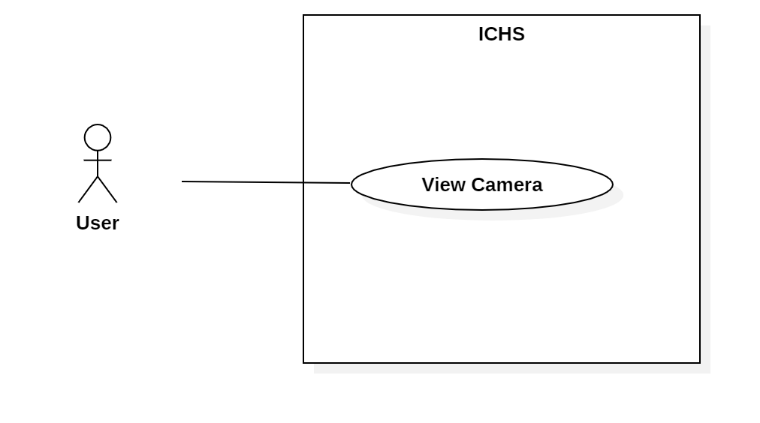


Figure : - <User> View Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_28** | | | |
| **Use Case No.** | 28 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Camera | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view all cameras in one area.   **Goal:**   * User view all cameras in one area that user choosen.   **Triggers:**   * User clicks “Store” button * User clicks “View area” button. * 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 shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User goes to dashboard page and click “store” button. | System show all store with account user. | | 2 | User clicks “Area” button in Store page. | System shows the “Area” page and shows areas. | | 3 | User clicks “Camera” button in Area page. | System shows the “Camera” page and shows cameras.  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A.  **Business Rules:** N/A. | | | |

Table : <User> View Camera

## **<User> View Streaming Camera**

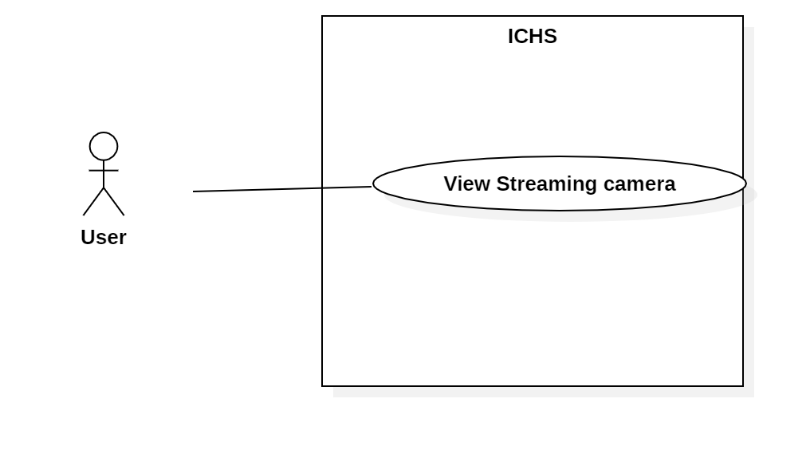


Figure : <User> View Streaming Camera

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_29** | | | |
| **Use Case No.** | 29 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Streaming Camera | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **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 on “View area” button. * User clicks on “View camera” button. * 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 on “View” button on store page. | System shows list all area in store. | | 2 | User clicks on “View” button on area page. | System shows list all camera in area. | | 3 | User click on “View” button on camera detail page. | System shows camera.  [Exception 1] |   **Alternative:** N/A.  **Exception:** N/A.   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A.  **Business Rules:** N/A. | | | |

Table : <User> View Streaming Camera

## **<User> See Object Detection**

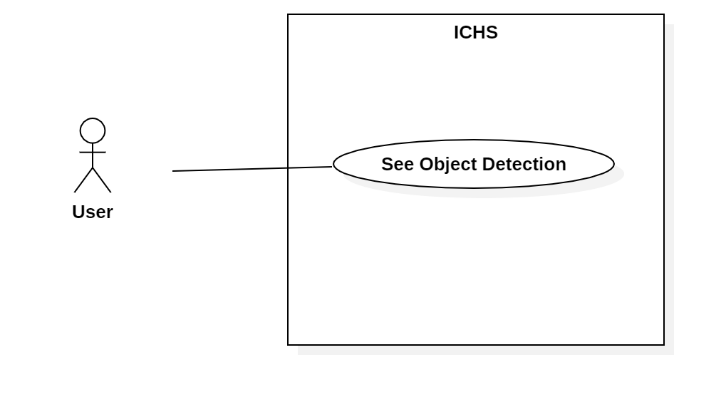


Figure : <User> See Object Detection

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_30** | | | |
| **Use Case No.** | 30 | **Use Case Version** | 1.0 |
| **Use Case Name** | See Object Detection | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **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 on “View area” button. * User clicks on “View camera” button. * 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 shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | User clicks on “View” button on store page. | System shows list all area in store. | | 2 | User clicks on “View” button on area page. | System shows list all camera in area. | | 3 | User click on “View” button on camera detail page. | System shows camera. | | 4 | User clicks “Object Detection” button. | System shows box in the video with count people.  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A  **Business Rules:** N/A. | | | |

Table : <User> See Object Detection

## **<User> See Count People**

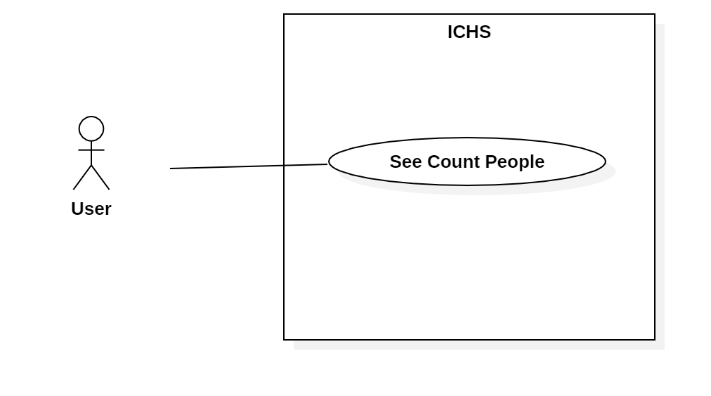


Figure : <User> See Count People

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_31** | | | |
| **Use Case No.** | 31 | **Use Case Version** | 1.0 |
| **Use Case Name** | See Count People | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **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 on “View area” button. * User clicks on “View camera” button * 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 on “View” button on store page. | System shows list all area in store. | | 2 | User clicks on “View” button on area page. | System shows list all camera in area. | | 3 | User click on “View” button on camera detail page. | System shows camera. | | 4 | User clicks “Object Detection” in Camera Detail page. | System displays the video with count people. | | 5 |  | System shows box on streaming camera and detect person by box and count person.  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A  **Business Rules:**   * System count people when people was detect. | | | |

Table : <User> See count people

## **<User> See Face Detection**

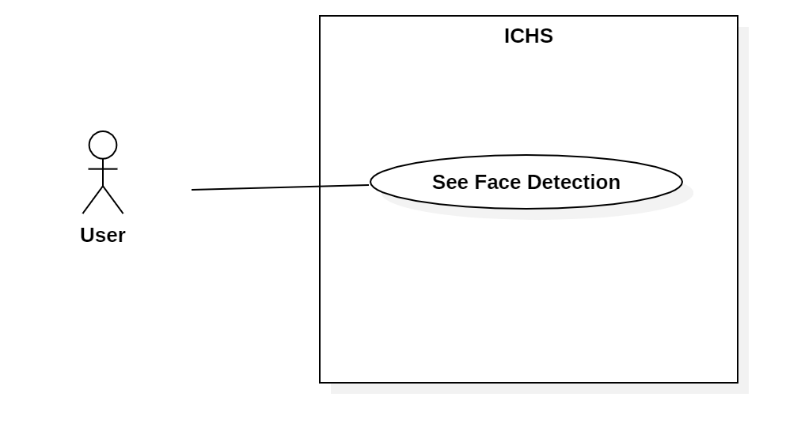


Figure : <User> See Face Detection

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_32** | | | |
| **Use Case No.** | 32 | **Use Case Version** | 1.0 |
| **Use Case Name** | See Face Detection | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **Priority** | Normal |
| **Actor:**   * User   **Summary:**   * This use case allows the user to see face detection in streaming video from the selected camera.   **Goal:**   * User see face detection in streaming video.   **Triggers:**   * User clicks on “View area” button. * User clicks on “View camera” button * 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 on “View” button on store page. | System shows list all area in store. | | 2 | User clicks on “View” button on area page. | System shows list all camera in area. | | 3 | User click on “View” button on camera detail page. | System shows camera. | | 4 |  | System shows box on streaming camera and detect face when face people near camera.  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A  **Business Rules:**   * System face detection when face is near camera. * Face detection suggest age and gender for people. | | | |

Table : <User> See Face Detection

## **<User> View Stream Heatmap**

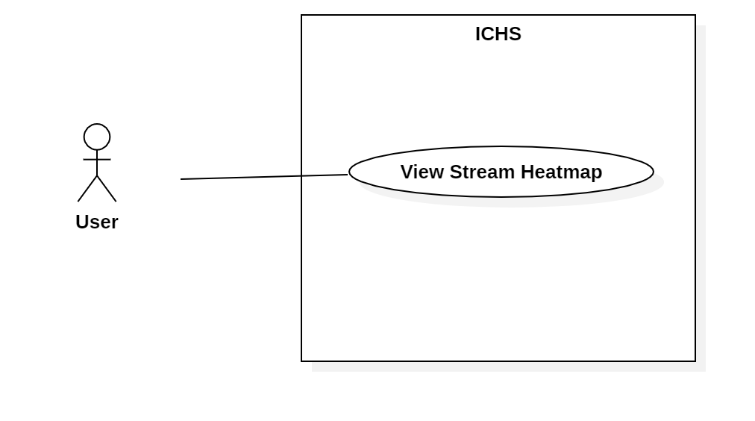


Figure : <User> View Stream Heatmap

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_33** | | | |
| **Use Case No.** | 33 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Streaming Heatmap | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view the streaming heatmap from the selected camera.   **Goal:**   * User view streaming heatmap from camera.   **Triggers:**   * User clicks on “Store” button. * User clicks on “view area” button. * User clicks on “view camera” button. * User clicks on “heatmap” 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 on “store” button. | System shows list of store of user. | | 2 | User clicks on “View area” button. | System shows list of area in this store that user clicks on. | | 3 | User clicks on “view camera” button. | System shows that camera user clicks on. | | 4 | User clicks “heatmap” button. | System shows heatmap on this camera.  [Exception 1] |   **Alternative:** N/A.  **Exception:** N/A.   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A.  **Business Rules:**   * Heatmap shows dots on streaming camera. | | | |

Table : <User> View Stream Heatmap

## **<User> View Report**

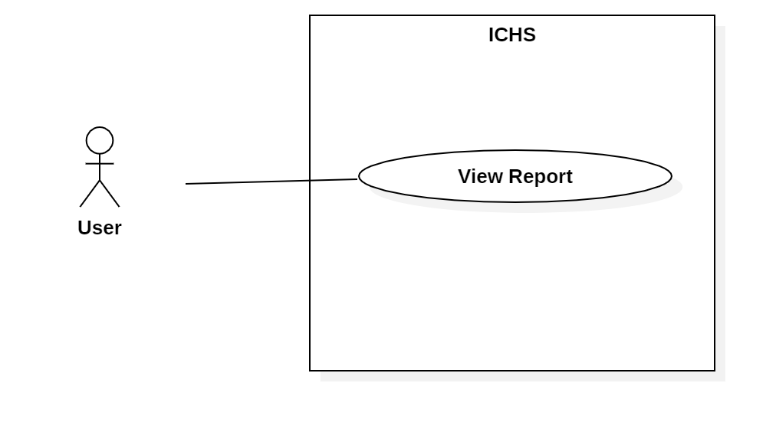


Figure : <User> View Report

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_34** | | | |
| **Use Case No.** | 34 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Report | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view report 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.  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A.  **Business Rules:**   * View report following store, area and camera. | | | |

Table : <User> View Report

## **<User> Preview heatmap in time**

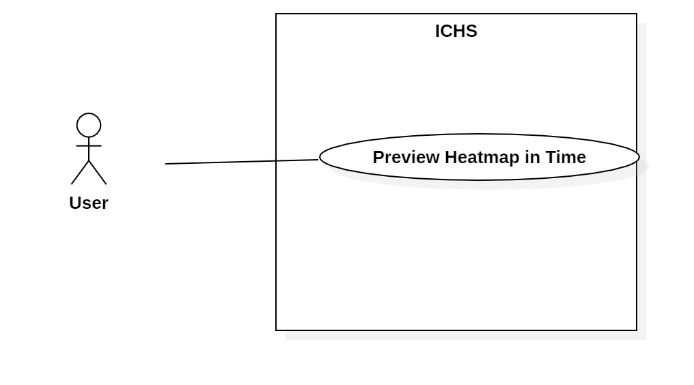
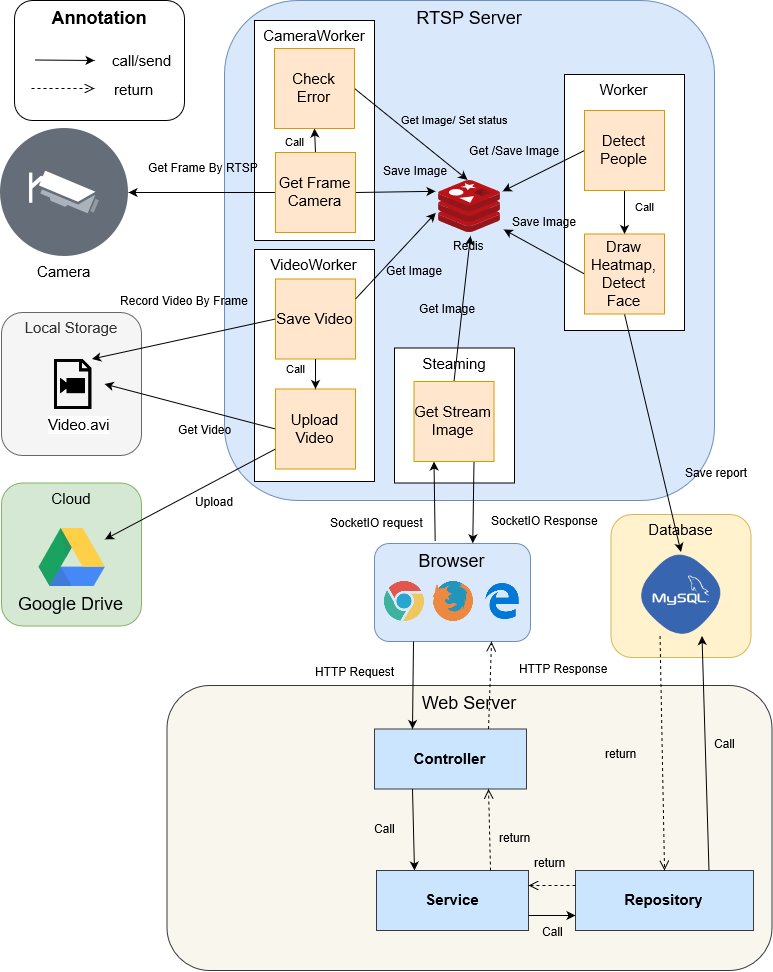


Figure : <User> View Heatmap

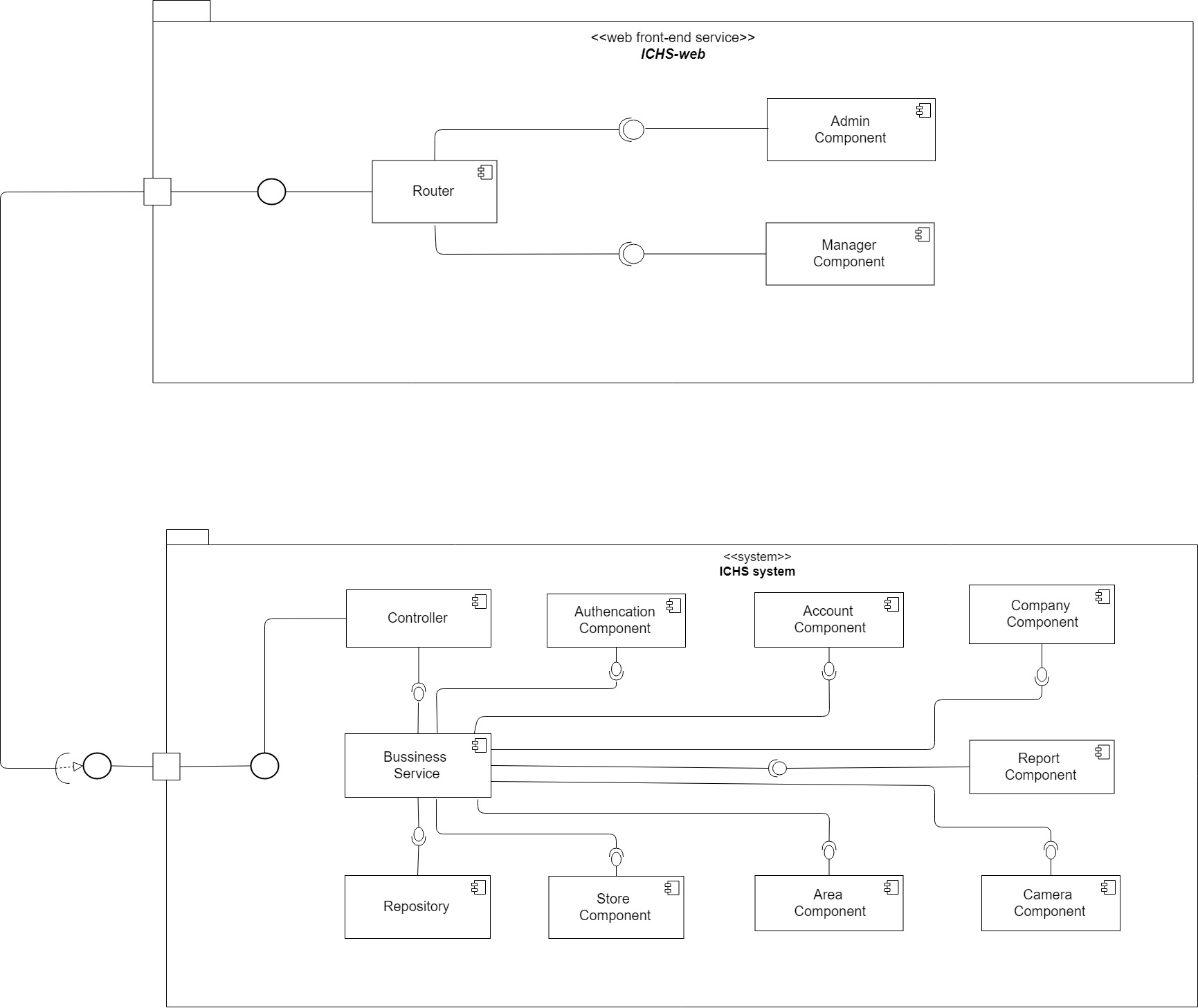
|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_35** | | | |
| **Use Case No.** | 35 | **Use Case Version** | 1.0 |
| **Use Case Name** | Preview Heatmap in Time | | |
| **Author** | TuyenNQ | | |
| **Date** | 08/08/2019 | **Priority** |  |
| **Actor:**   * User   **Summary:**   * This use case allows the user to view the streaming video from the selected camera and preview heatmap.   **Goal:**   * User view history heatmap.   **Triggers:**   * User clicks on “Store” button. * User clicks on “view area” button. * User clicks on “view camera” button. * User clicks on “preview heatmap” button. * 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 on “store” button. | System shows list of store of user. | | 2 | User clicks on “View area” button. | System shows list of area in this store that user clicks on. | | 3 | User clicks on “view camera” button. | System shows that camera user clicks on. | | 4 | User clicks on “preview heatmap” button. | System shows heatmap in history.  [Exception 1] |   **Alternative:** N/A.  **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Network problem. | System shows error message. |   **Relationships:** N/A  **Business Rules:**   * System must be show heatmap on per time. * Heatmap update every 1 minute. | | | |

Table : <User> preview heatmap in Time

# **ARCHITECTURAL DIAGRAM**

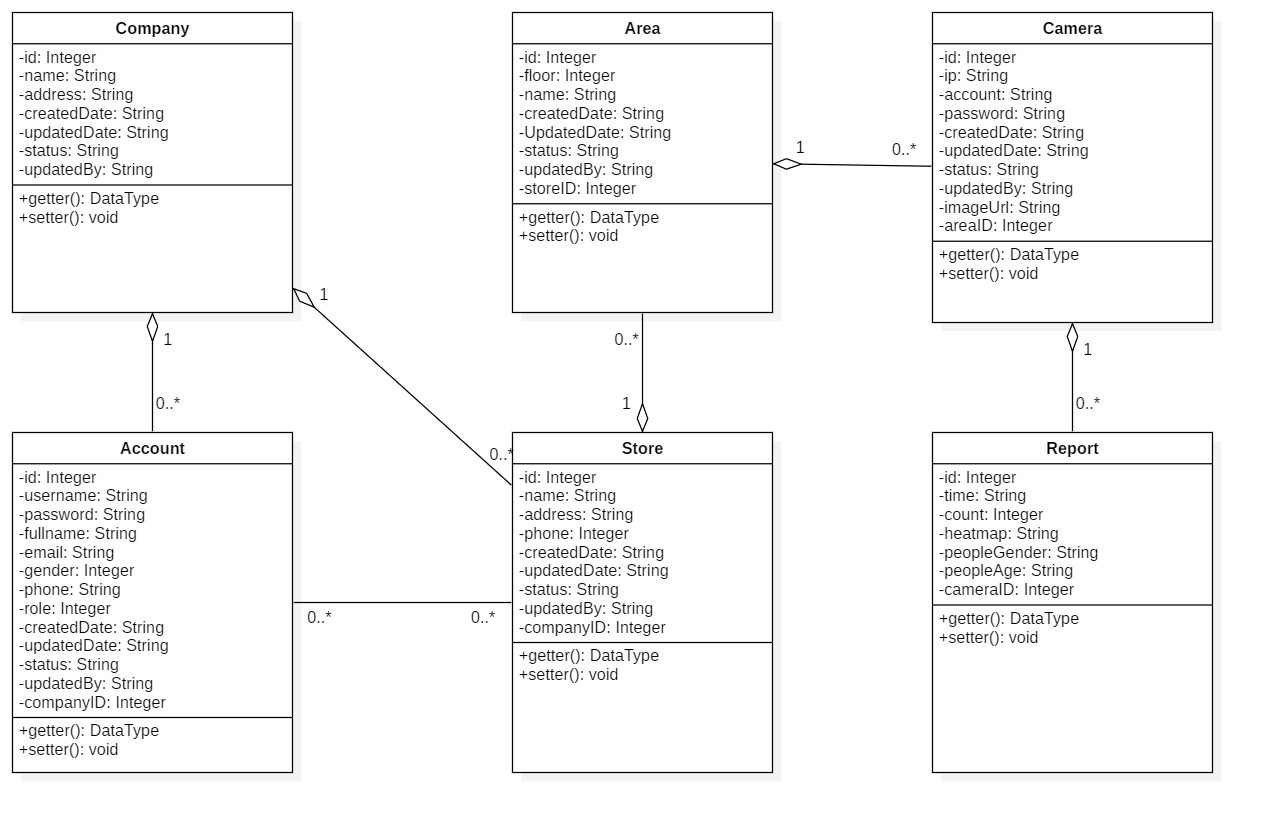


# **COMPONENT DIAGRAM**



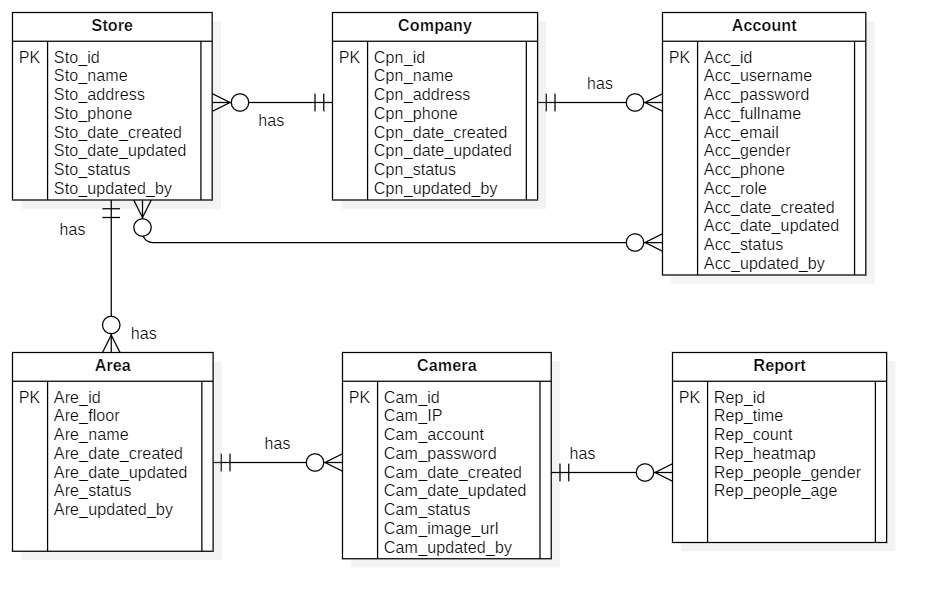
| **Components Dictionary: Describes components** | |
| --- | --- |
| Business Service | Component which handles business process for Web Sever. Include |
| Repository | Component which handles storing and retrieving data from database. |
| Controller | Component which handles request and response, accept 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 |

# **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 |

# **ENTITY RELATIONSHIP DIAGRAM**



|  |  |  |
| --- | --- | --- |
| **Entity dictionary** | | |
| **Entity Name** | **Mapping column with Conceptual diagram** | **Description** |
| **Account** | Account | Contain the account information. |
| **Company** | Company | Contain the company information. |
| **Store** | Store | Contain the store information |
| **Area** | Area | Contain the area information. |
| **Camera** | Camera | Contain the camera information. |
| **Report** | Report | Contain the report information. |

# **INTERACTION DIAGRAM**

## **Login**

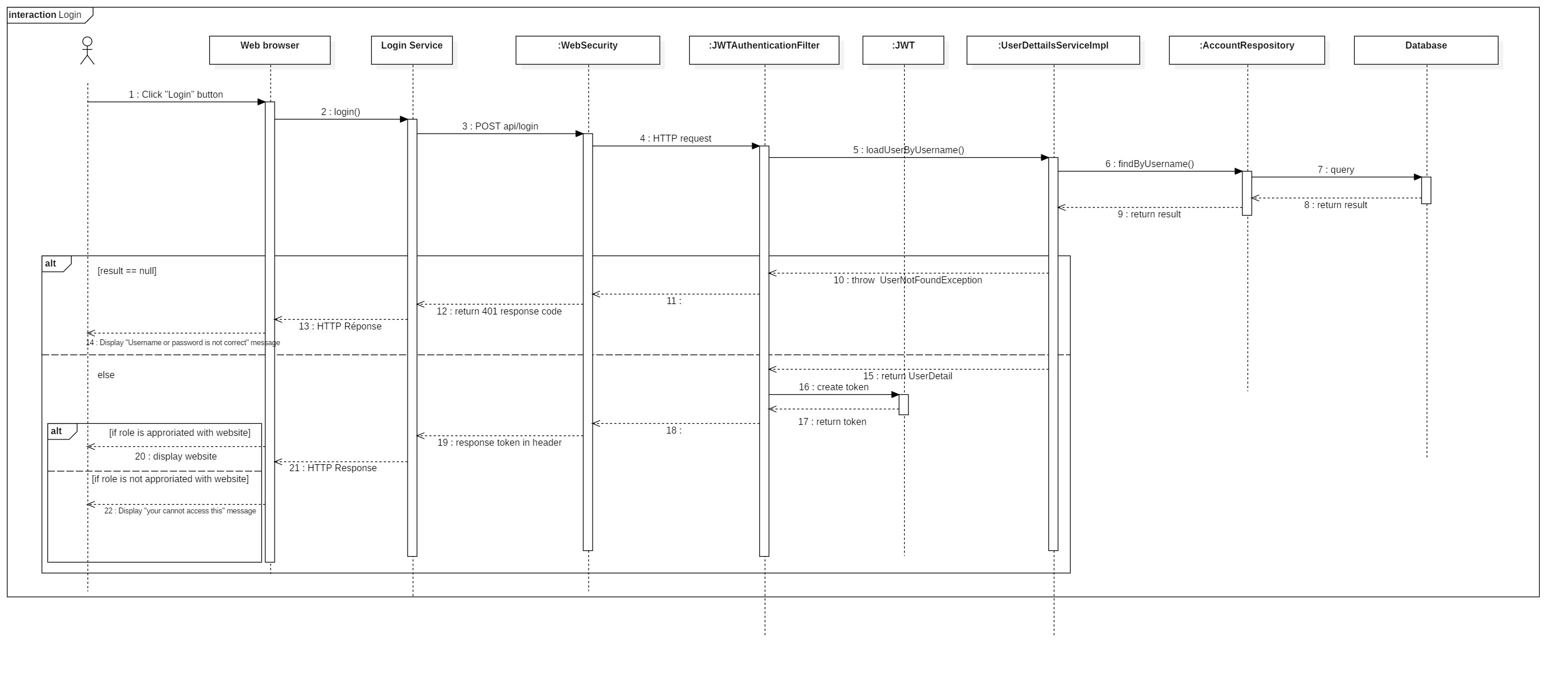


Figure : Sequence Diagram <Login>

## **Create Company**

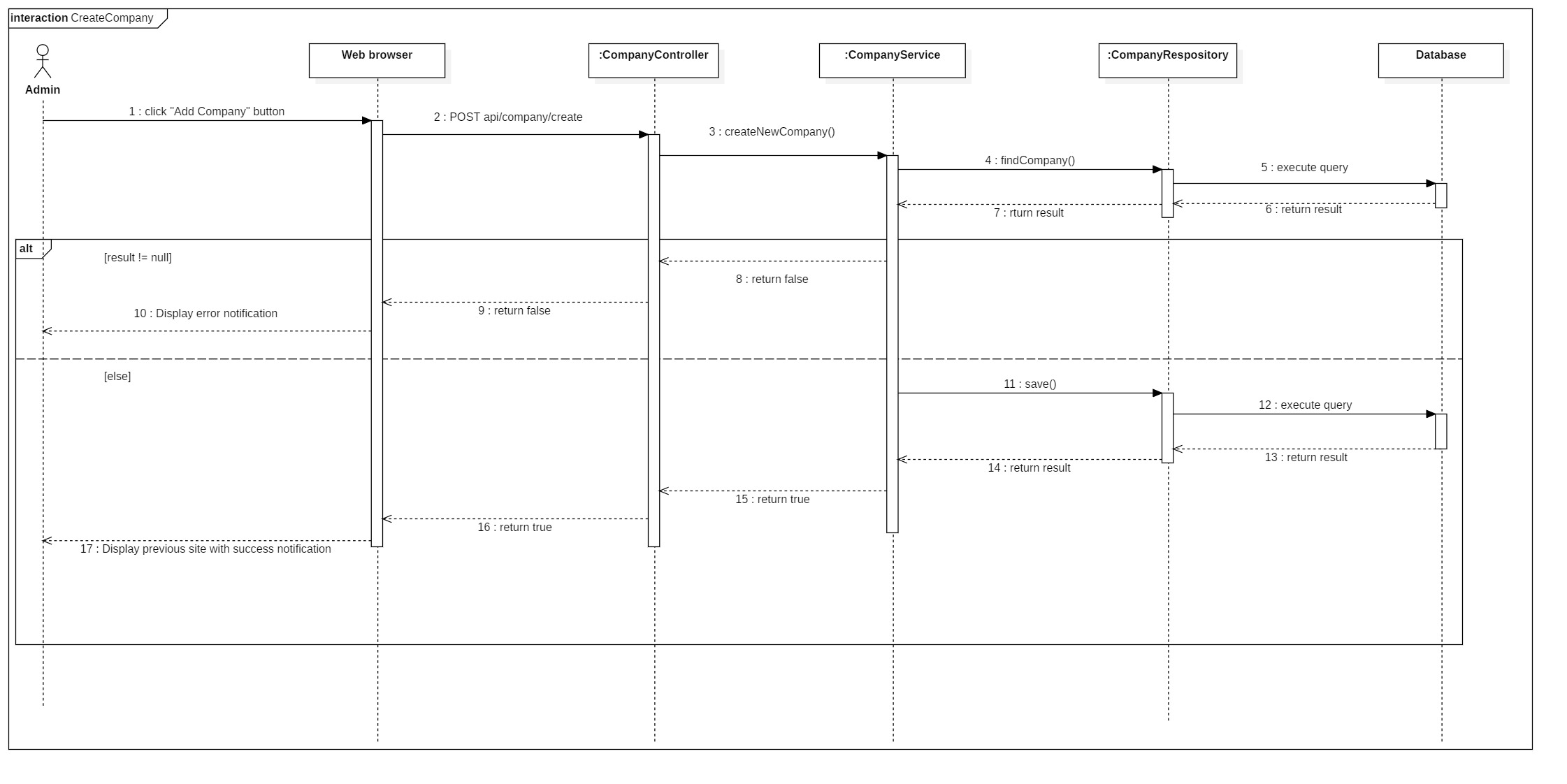


Figure : Sequence Diagram <Create Company>

## **Update Company**

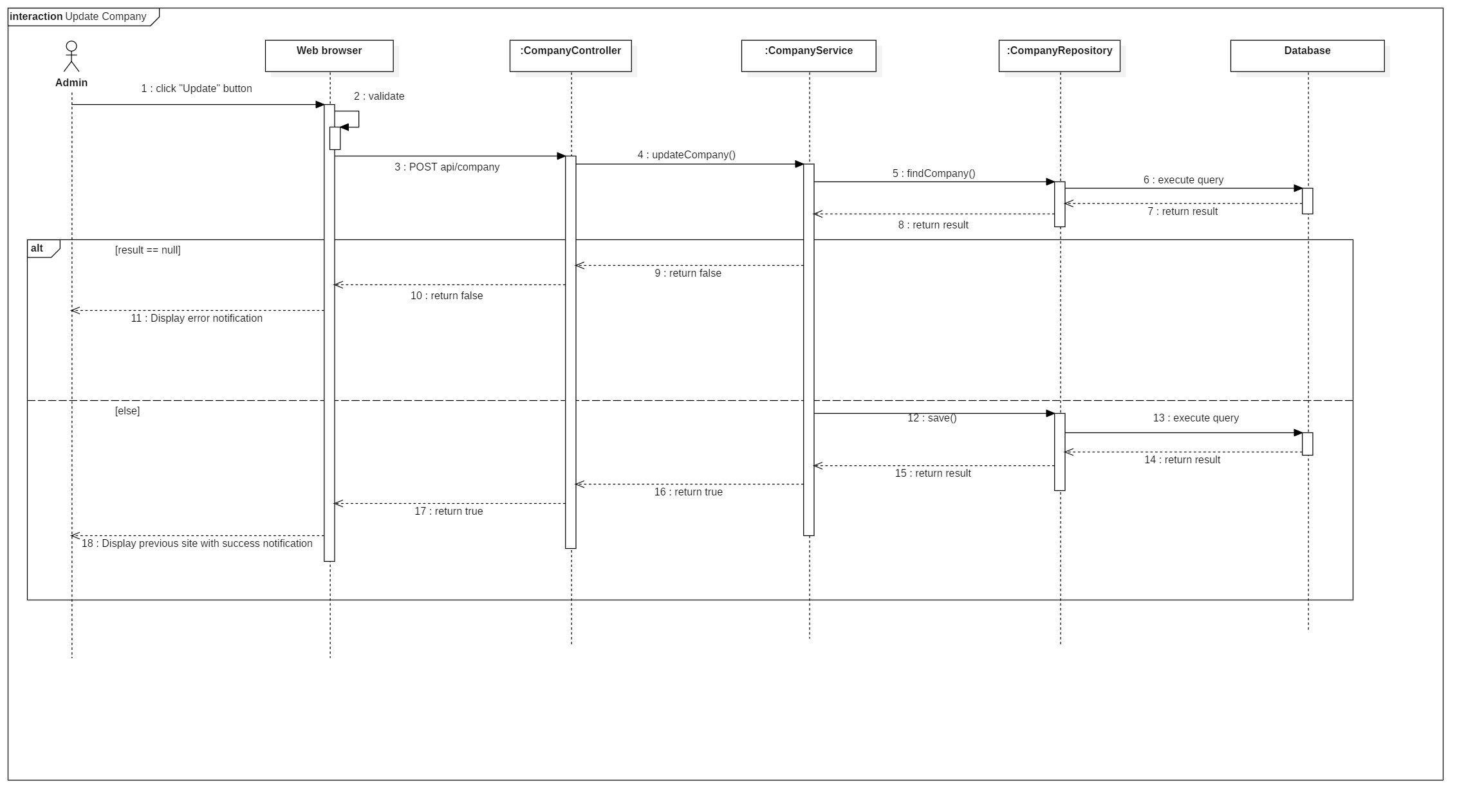


Figure : Sequence Diagram <Update Company>

## **Activate Company**

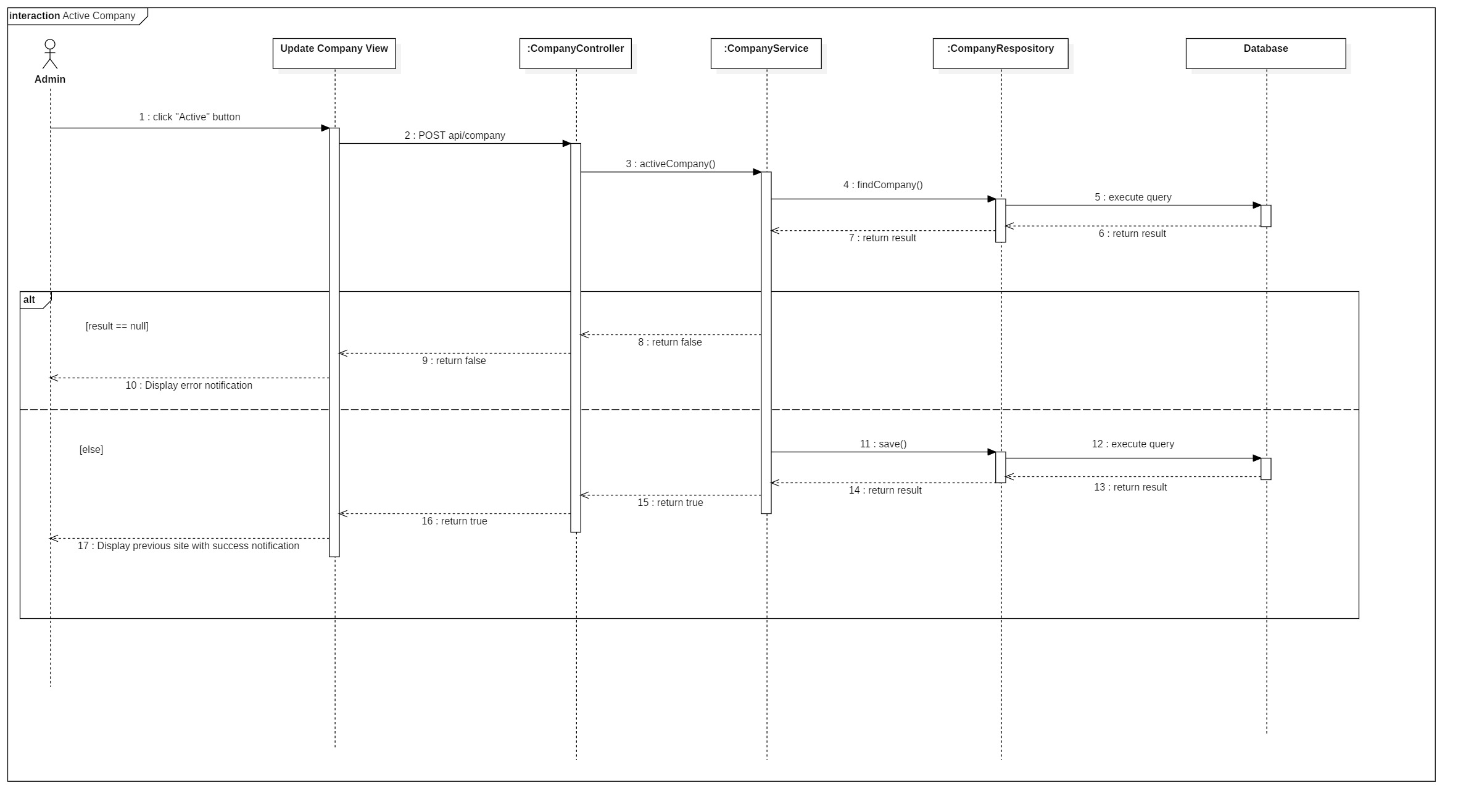


Figure : Sequence Diagram <Activate Company>

## **Deactivate Company**

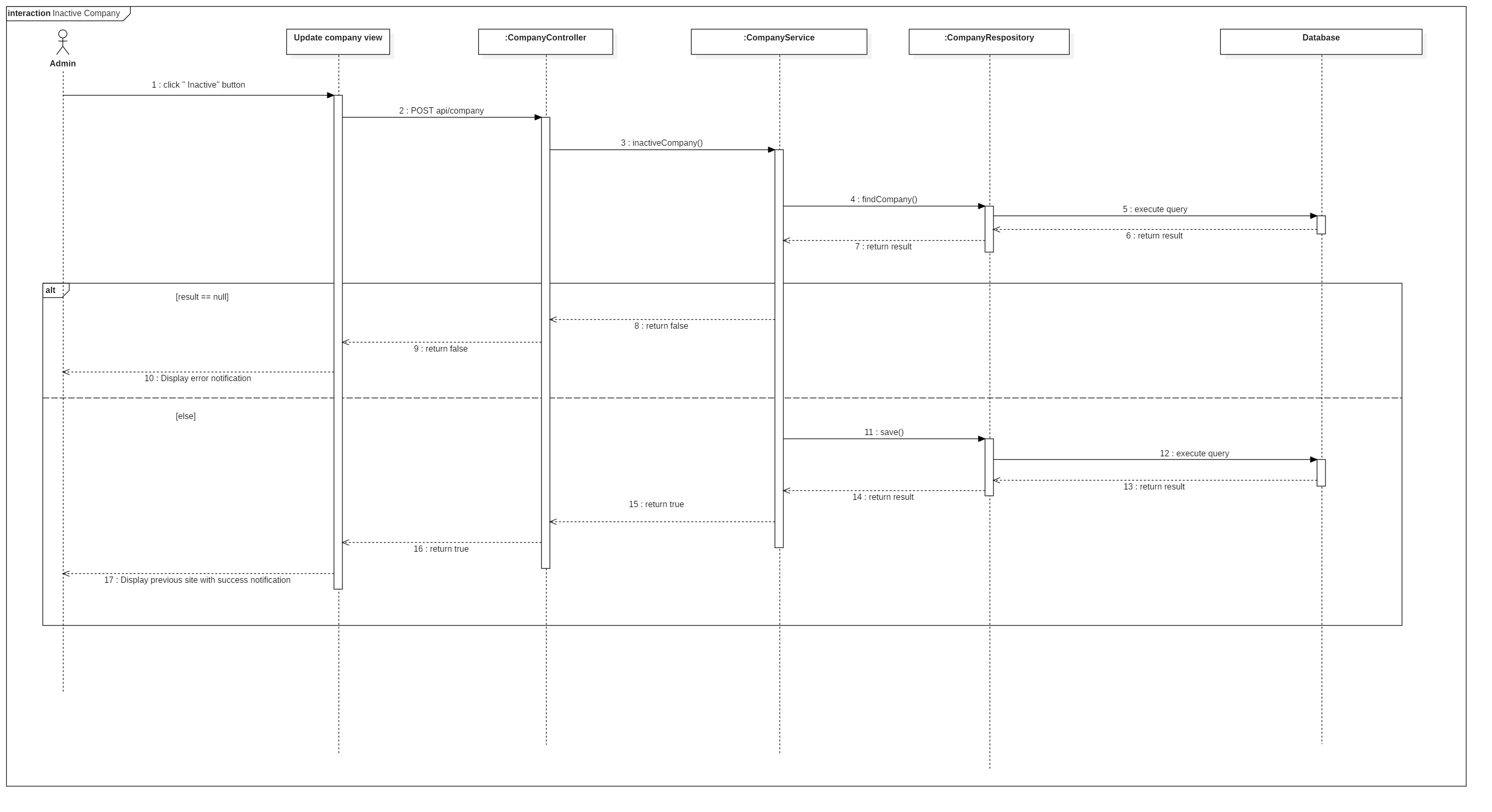


Figure : Sequence Diagram <Deactivate Company>

## **Search Company**

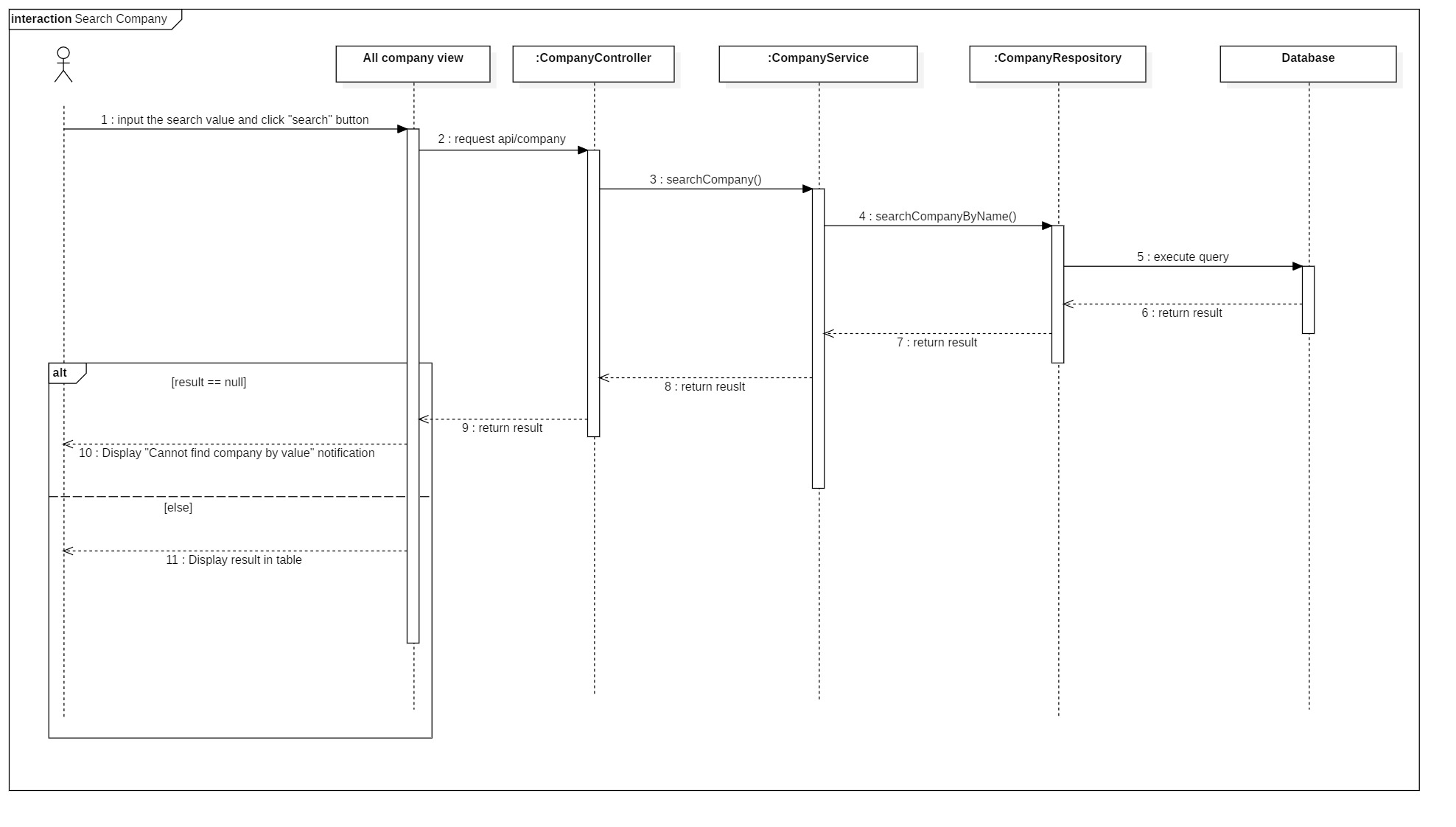


Figure : Sequence Diagram <Search Company>

## **Get report camera by time**

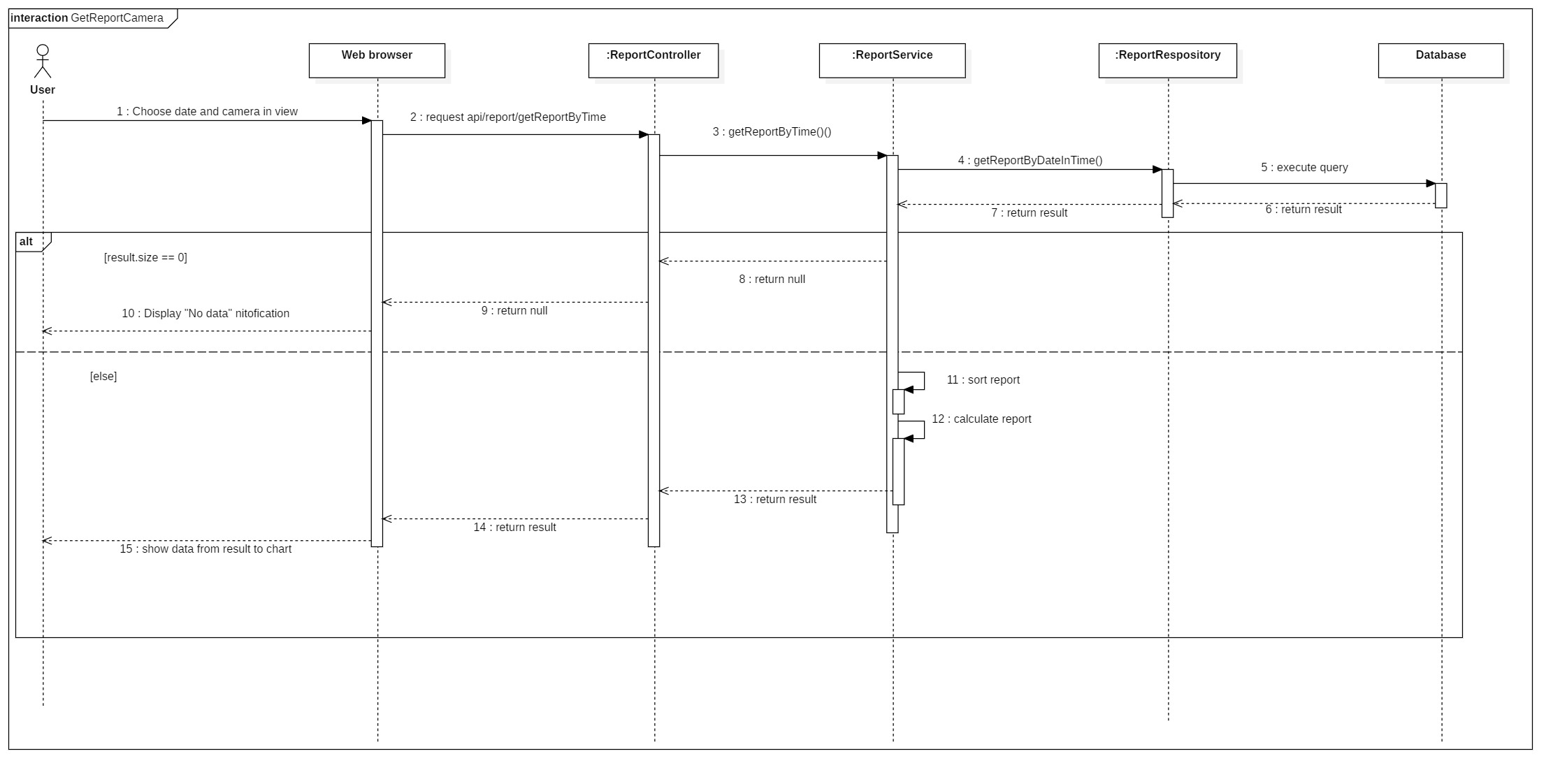


Figure : Sequence Diagram <Get report camera by time>

## **Get report area by time**

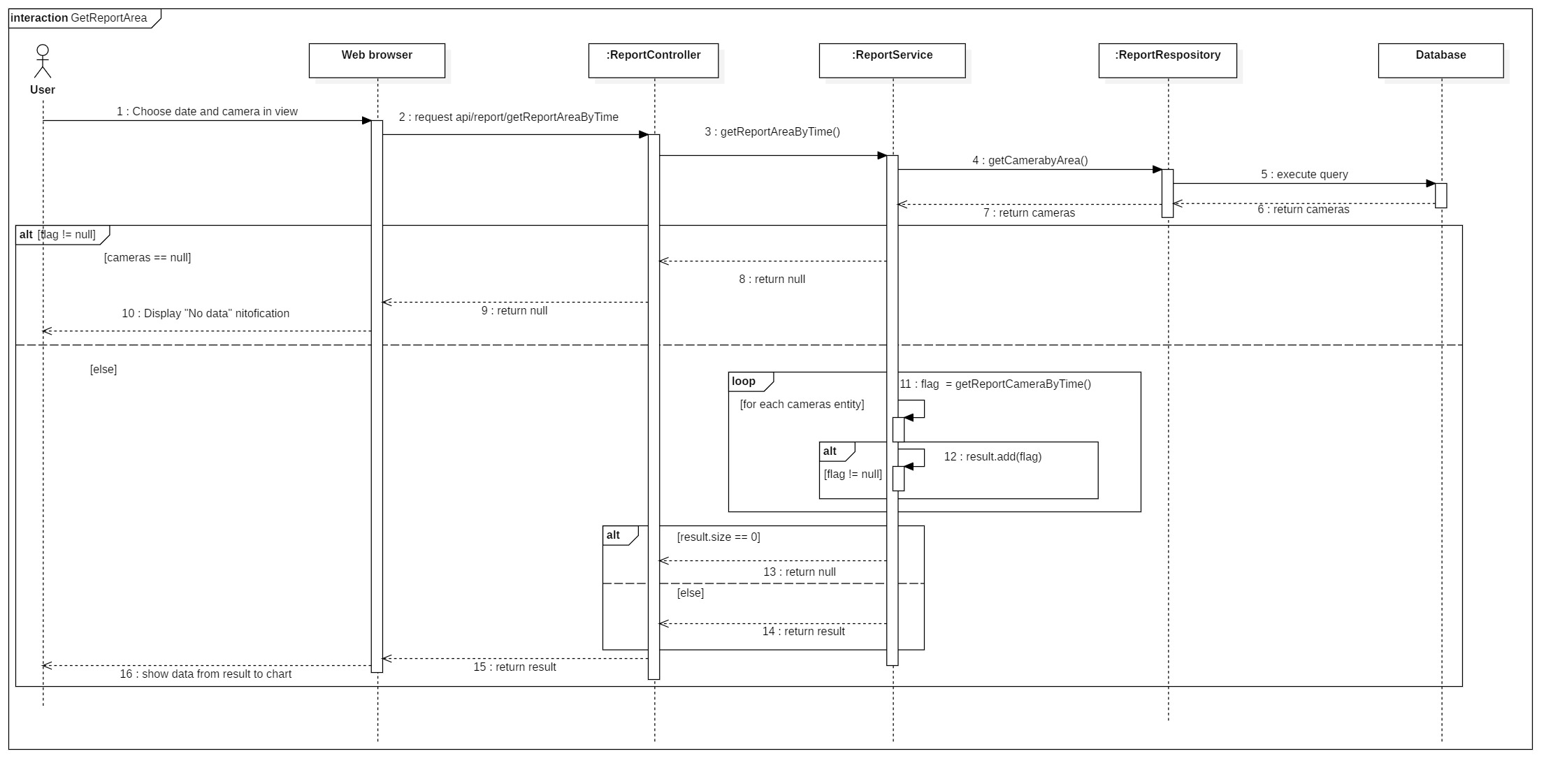


Figure : Sequence Diagram <Get report area by time>

## **Get report store by time**

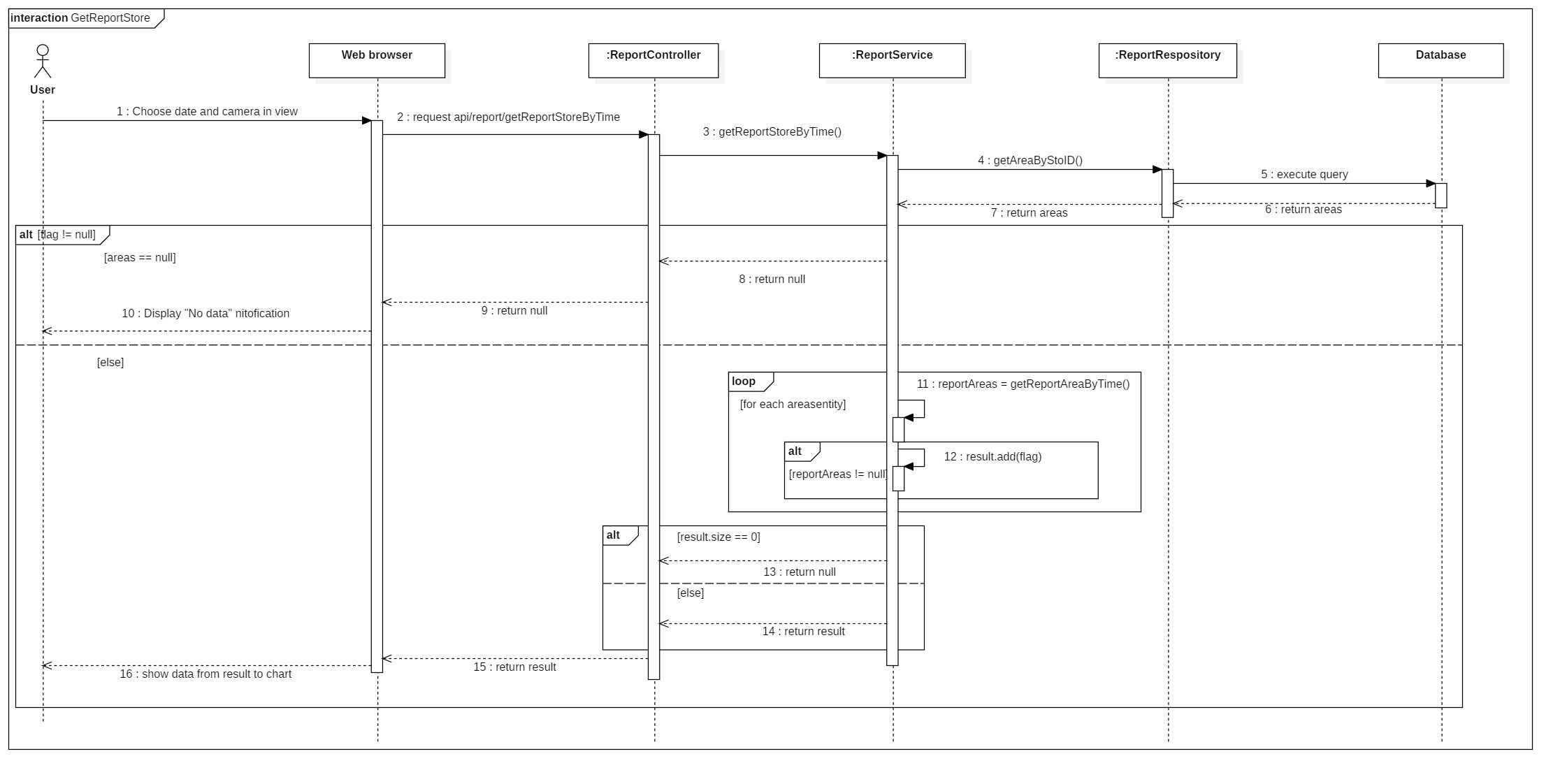
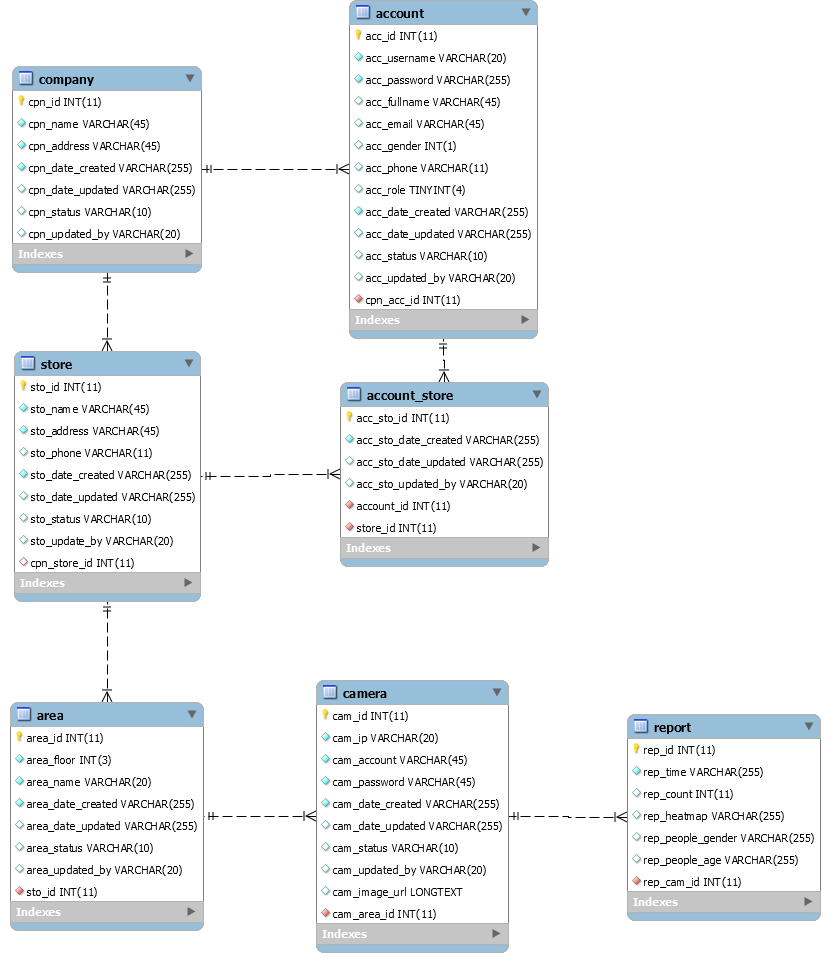


Figure : Sequence Diagram <Get report store by time>

# **PHYSICAL DIAGRAM**

,s,s,s,,s,s,s,s,s,s,s,s,s,s

|  |  |
| --- | --- |
| 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. |

# **ARCHITECTURE DIAGRAM**

# **ALGORITHMS**

## **Faster RCNN**

### **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.

### **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.

### **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.





Figure : Faster RCNN 1

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)





Figure : Faster RCNN 2

## **Draw Heatmap**

### **Definition**

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

### **Define problem**

Color must be changed based on long people stay in camera

### **Solution**

Use dot image to draw on grayscale image.

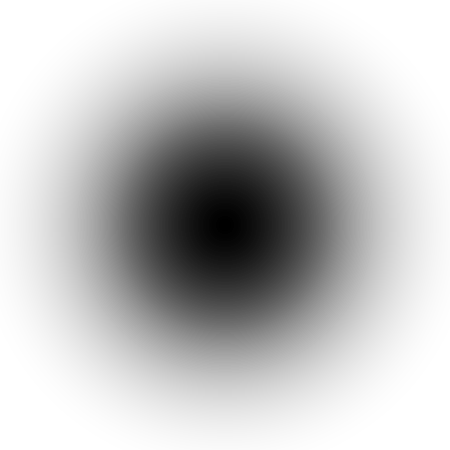


Figure : Draw Heatmap <Dot Image>

And create a heatmap color image in local storage.



Figure : Draw Heatmap <Color Range>

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.

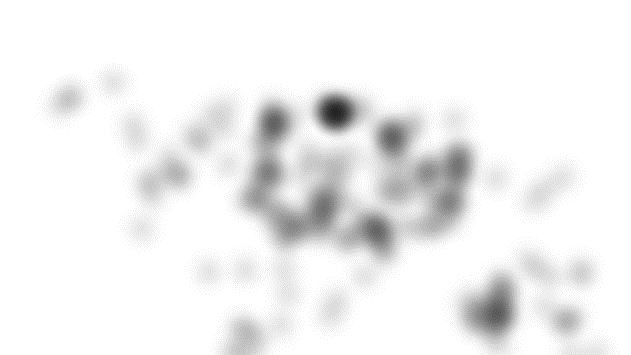
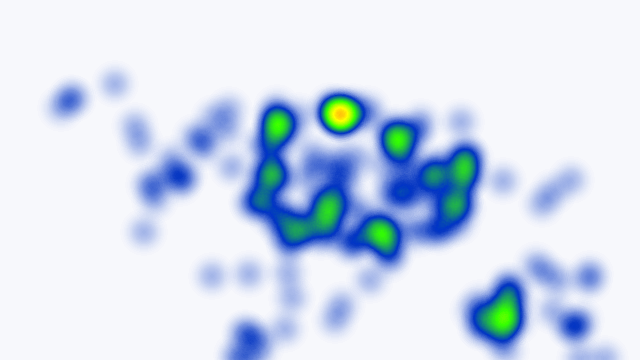
 

Figure : Draw Heatmap <Heatmap Color Image>

## **Face Detection**

### **Definition**

The Haar-like algorithm is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, centre detection for detecting eyes, nose, mouth, etc. in the picture.

### **Define problem**



Figure : Face Detection

Light, masked, special emotion, looks like face ...

### **Solution**

Featured of Haar-like

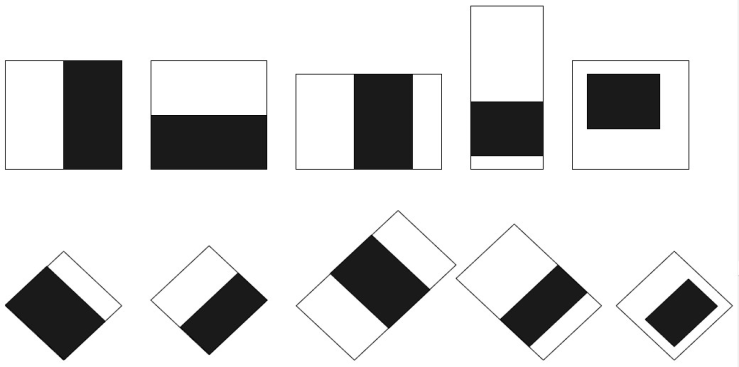
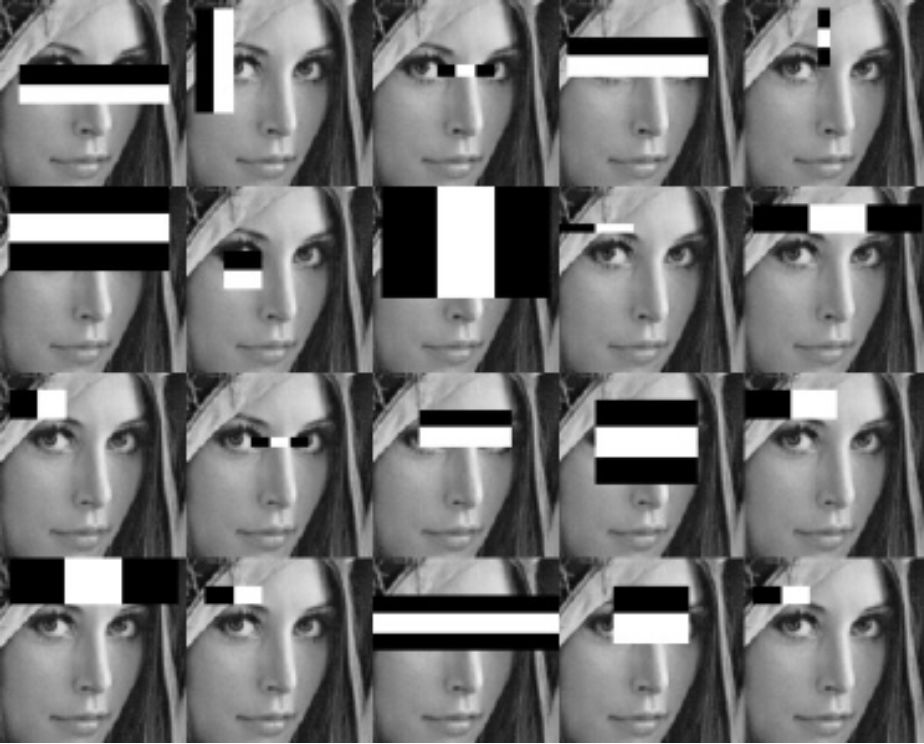


Figure : Face Detection <Haar-Like> 1

Haar-like algorithm give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face.



# **FUTURE PLAN**

# **DIAGRAM EXPLANATION**

## **CLASS DIAGRAM EXPLANATION**

### **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 |

Table : Clas Diagram Explanation <Account>

### **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 |

Table : Clas Diagram Explanation <Company>

### **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 |

Table : Clas Diagram Explanation <Store>

### **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 |

Table : Clas Diagram Explanation <Area>

### **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 |

Table : Clas Diagram Explanation <Camera>

### **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 |

Table : Clas Diagram Explanation <Report>