MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

iMuseum

Group 01	
Group members	Nguyễn Tấn Phát (Team Leader) Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng
Supervisor	Mr. Kiều Trọng Khánh
Ext. Supervisor	N/A
Capstone Project code	iBM

-Ho Chi Minh City, **09/05/2016**-

This page is intentionally left blank

Table of Contents

List of Tables

List of Figure

This page is intentionally left blank

Definitions, Acronyms, and Abbreviations

Name	Definition
iBM	iMuseum
BOM	Backend Office Management
BLE	Bluetooth Less Energy

A. Introduction

1. Project Information

- Project name: **iMuseum**
- Project Code: **iBM**
- Product Type: **Android Mobile application, BOM Website**
- Start Date: **09/05/2016**
- End Date:

2. Introduction

Museum is the place that we store tons of past memorable events, things, etc… for the next generations to research what happened in the past, what their seniors have done. However, it is hard for a normal person to explore the whole museum. There for a museum's employee or a tourist guide is required for every visit of visitors. In addition, because of the huge number of visitors, the museum's employee just can provide the overview information, and it cannot link related events. The information is sometimes missed.

We build a system, which help the visitors can acquire the fullest detail of information of the event. In the process of our research, we find out that iBeacon technology is the key to solve the problem. By using the Estimote Beacons, user's mobile device can listen signal from the beacons the physical world then the mobile application can understand the location of the beacons on a micro local scale, and deliver fully information content about the event to user base on location.

We all hope the system as so as our solution will help visitors can acquire historic information in the best way every time they visit the museum.

3. Current Situation

Currently, there is no official system that to support providing, linking fully information about the events, things, etc… which are demonstrated in the museum in Vietnam.

So far, the most effective way to provide information for the visitors needs a museum's employee or a tourist guide to provide the information.

Process of visiting the museum with a tourist guide:

- All visitors must follow the tourist guide.
- The tourist guide leads the way and provide the information about the event whose location they are currently standing.
- Victors can ask some question about the event.
- The tourist guide answers the question (in some cases, the question can't be answer due to the knowledge of the tourist guide)
- Visitor must depend a lot on the tourist guide and cannot spend more time for an event or freely explore the museum.

4. Problem Definition

Below are advantages / disadvantages of the current situation:

- Advantages:

- The comfortable and natural feeling when communicate between human and human.

-Disadvantages:

- The information is overview: Due to the huge number of the visitors and limited time, the tourist guide just can provide the information in a brief way.
- Lack/ wrong information: All the information that the visitor acquire is based on the tourist guide knowledge. Therefore, with a beginner or amateur tourist guide, the information may be lack or the tourist guide may provide wrong information.
- Visitors cannot freely explore the museum: museum is a great place to explore but if the visitors do that, they might miss the explanation of the tourist guide.
- Tourist guide cannot transfer all of information for visitors: there are a lot of information that the tourist guide need to transfer to visitors. However, because of time barrier, visitors cannot get all the information, which they want.

5. Proposed Solution

Our proposed solution is to build a system named “iBM”, which use a number of Estimote Beacons and an internet connected mobile device to help the visitors freely explore the museum and provide fully information to them without a tourist guide required. We also design the system to be scalable so we can deploy this system on other kind of instruction for traveling purpose in future plan.

iBM system includes a web application and android mobile application with following functions:

5.1 Feature functions

- ***Web application (for admin):***
  - *For admin:*
    - Manage historic item: Admin can take manage historic event, things.
    - Manage beacon: Admin can take manage beacon by manipulating beacon information.
    - Manage reports: Admin can take manage the visitors report by viewing and assigning them to expert.
    - View movements log: Admin can view the log information of beacon movements.
  - *For expert:*
    - Manage historic information: Expert can manipulate the historic information of historic event, things.

- ***Mobile application:***
  - *For visitors:*
    - Ticket code: Visitor can confirm the code after buy the ticket to use

the system. The code is printed on the ticket and available in a day.
- View museum map: Visitor can view the map of the museum.
- Provide information: Visitor can view fully information about the event, things that the beacon carries. Visitors can also know the related event, stories.
- Navigate: Visitor can view the navigations to know where the beacon that contain the related event to explore.
- Add favorite: Visitor can save the historic item and read them offline.

o *For staff:*
- Detect beacon location: if the beacon is move out of it proper location, the system will notify for the staff.

o *For admin:*
- Map item with beacon: Admin can map a historic item with a beacon.

5.2Advantages and disadvantages

The advantages and disadvantages of the proposed solution:

- **Advantages:**

- Information provide fully: Information about the historic events, stories will be provided fully, correctly. Not just the information about the event itself, all the related event will be suggested for visitors to explore.
- Explore freely: Visitors can freely explore the museum by themselves without a tourist guide.
- Visitors also know where they are in the museum with the navigation, so the visitors will not be lost or missed what they want to explore.
- In case, there is a crowd in front of the historic item, visitor can also get the information of historic item from a specific distance without come closer to the historic item.

- **Disadvantages:**

- This system is not currently support other OS.
- Smartphone must always have both Bluetooth (BLE) and internet connection to get data from beacon.

6. Functional Requirements

Function requirements of the system are listed as below:

- **Admin component:**

- *Manage historic item:* Admin can manipulate information of the historic item.
- *Manage beacon:* Admin can manipulate the information of the beacons.
- *Manage reports:* Admin can view and assign Visitor reports to Expert.
- *View log:* Admin can view the log of historic items movement.

- **Expert component:**

- *Manage historic information:* Expert can manipulate historic information of the historic item.

- **Staff component:**

- *Track Item:* staff will be notified which beacons move out of the proper location and see all moved-beacon in a list to checking again.

- **Visitor component:**

- *Confirm code:* Visitor can confirm the code on the ticket after buying to activate all the function of the system.
- *Detect historic item:* Visitor can view all the near by historic item with an overview description to decide what they are going to explore.
- *Provide information:* Visitors can view fully the information of the event with the beacon nearby. Visitors can also know the related event to explore
- *Navigate:* Visitor can view the navigations to know where the beacon that contain the related event to explore.
- Add favorite: Visitor can save the historic item and read them offline.

7. Role and Responsibility

No	Full Name	Role	Position	Contact
1	Kiều Trọng Khánh	Project Manager	Supervisor	khanhkt@fpt.edu.vn
2	Nguyễn Tấn Phát	Team Leader	Team Leader	phatntse61199@fpt.edu.vn
3	Trần Quang Tùng	Team Member	Team Member	tungtqse61279@fpt.edu.vn
4	Võ Thanh Hiếu	Team Member	Team Member	hieuvtse61201@fpt.edu.vn
5	Phan Quốc Hùng	Team Member	Team Member	hungpqse61169@fpt.edu.vn

Table 1. Roles and Responsibilities

B. Software Project Management Plan

1. Problem Definition

1.1 Name of this Capstone Project

-Official name: iMuseum
-Vietnamese name: Bảo tàng thông minh
-Abbreviation: iBM

1.2 Problem Abstract

To solve those problems, which mentioned above, we provide a system that provide fully information about the historic events, things, which are demonstrated in the museum for the visitors without a person as a tourist guide. The system also provides the related event. The system includes numbers of Estimote beacons, an internet connected mobile device and a web service. The system will play the tourist guide to provide information about everything in the museum. We also provide an information system as a BOM website for the admin to manipulate information of the beacons and tracking the location of the beacons.

However, at starting point, there is some problem for developing the system. The newest official SDK of Estimote Beacon is not stable. Sometimes the application crash because of the newest official SDK.

Following to Estimote Beacon document, ***"do not expect distance estimations from beacons."*** So the team must study the new way about identify the distance between smartphone and Estimote Beacon.

Following to Estimote Beacon document, it categorizes the beacons into four **proximity zones***: Immediate* (strong signal; usually up to a few centimeters), *near* (medium signal; usually up to a few meters), *far* (weak signal; more than a few meters), *unknown* ("hard to say", usually when the signal is very, very weak). They do not provide us the exactly proximity zones in number. Therefore, our team must test to determine the exactly number.

Estimote Beacon connects with smartphone via Bluetooth 4.0 BLE, our team should study about the connect protocol, which device support, which does not.

It is difficult for the museum to approach to new technology and management process.

Team has to set a plan to approach the Estimote Beacon:

- Test with available application to understand more about how to use the Estimote Beacon and how the Estimote Beacon work.
- Find out how to monitor and range the Estimote Beacon with the mobile device.
- Get data of Estimote Beacon of cloud and show it to mobile device.

1.3 Project Overview

1.3.1 Current Situation

Below there are the problems encountered in this project:

- **Disadvantages:**
  - Risk: Because the project uses Estimote Beacon, team must study new technology/ API to apply it.
  - Smartphone must always have Bluetooth (BLE) and internet connection to get data from beacon.
  - Because of unclear Estimote Beacon's documents, we must do some test to determine exactly the information of Estimote Beacon.
  - The museum must approach to new technology and new management process.

- **Advantages:**
  - Receive good support from Estimote Beacon development forum: Because of the development of Estimote community, it is easier for team to get support from Estimote community forum when raise a problem.
  - Estimote Beacon has the official SDK for Android.
  - Estimote Beacon development forum has detail demo code tutorial for us to study and understand how to monitor and range with beacon

1.3.2 The Proposed System

According to the technology researches, Estimote Beacon is the key to solve the current situation about providing information for the museum visitors. We can use the feature of Estimote Beacon to solve the problem. The basic idea is to use the Estimote Beacon to detect which historic items is nearby the visitors through Bluetooth 4.0 (BLE). When visitors move close the historic item, visitors can choose which historic item to read all of its information, and related historic items.

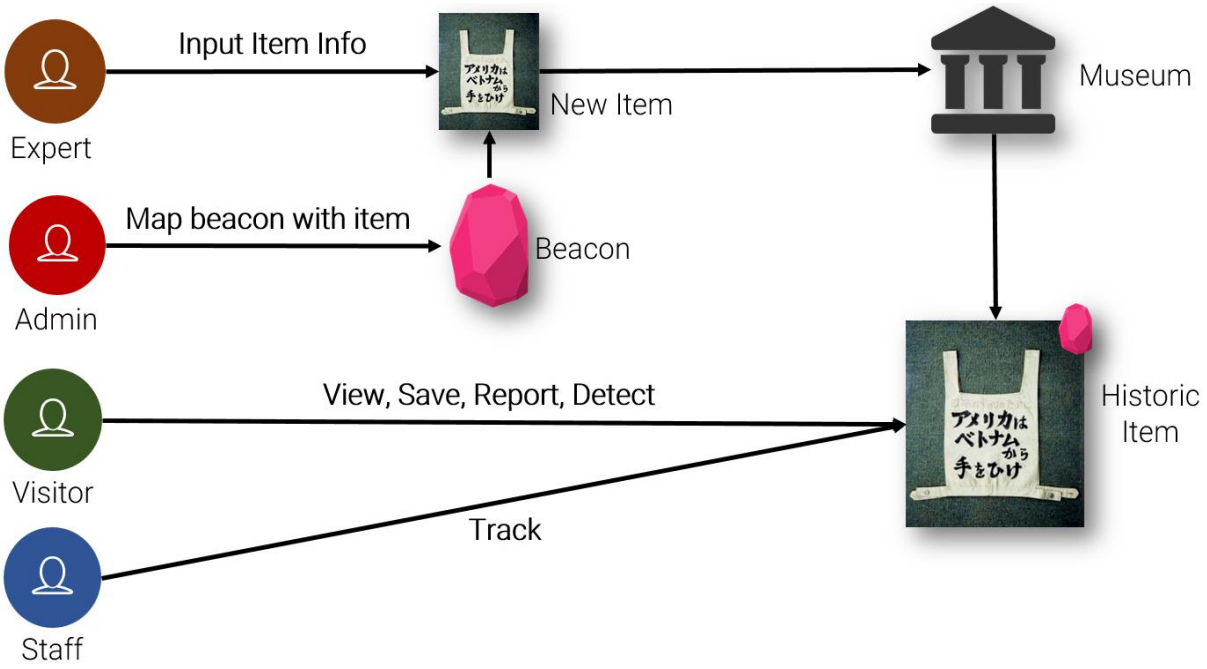


Figure 1. The proposed system

To deploy the system, the museum places the inputted information beacon at the proper event. The admin of museum will map beacon with historic item.

The visitors must use smartphone that support Bluetooth v4.0 BLE Less Energy with our application installed. To use the system, Visitors must confirm the code, which is printed on the museum ticket. While visitors explore the museum, they will pass by the events, things, which are demonstrated in the museum. The beacons will be display on the smartphone with overall description, to explore the information; the visitors have to choose the beacon on smartphone screen. The proper beacon will transfer the information to the smartphone including full information about the event and the related event.

Besides, the staff in the museum can check the historic item whether the item was moved or not.

1.3.2.1 BOM Website

BOM website is a place that admin can monitors, configures, manipulates the information in the system and assigns a Visitor's report to an expert. Expert can also use BOM website to manipulate historic item information, such as approving pending item and editing historic information.

- *For admin*
  - *Manage historic item*: Admin can take manage the historic item by manipulating the location, which beacon is connected and the size of the historic event, things.
  - *View item movement log*: Admin can view the log of historic item's movements.

- *Manage beacon*: Admin can take manage the beacon by manipulating the information of beacon.
- *Manage report of visitor*: Admin can receive and assign the report for expert to process the feedback of visitor.
- *Configure scheduler*: Admin can setup the time and the content in a website and let the auto handler get the information when hitting the time.

- *For Expert:*
  - *Manage historic information*: Expert can approve pending historic item information and manipulate those.
  - *View visitor's reports*: Expert can view visitor's reports, which are assigned by admin.

1.3.2.2 Web Services

Web service provide API for BOM Website and Mobile application to connect to server to perform function.

- *For visitor:*
  - Check the ticket code of visitor is valid or not to provide them the function get data of Estimote Beacon in the museum.
  - Get the historic event or things to mobile for visitor can read, save or report.
  - Show the direction from a historic event to another historic event.
  - Provide museum map.
- *For admin:*
  - Get the list of historic event or things in the museum.
  - Manipulate historic things.
  - Manipulate the link between beacons with historic things.
  - View report of visitors.
  - View Item movement log.
- *For expert:*
  - Get the list of historic event or things in the museum.
  - Manipulate historic things information.
  - View report of visitors.
- *For staff:*
  - Notify item's movement status for admin.

1.3.2.3 Mobile Application

- *For visitors*

This is the primary application, which provide to visitors with following functions:

- Confirm Ticket code: Visitor can confirm the code after buy the ticket to use the system. The code is printed on the ticket and available in a day.
- Get museum map: Visitor can view the map of the museum.

- Detect Beacon location: Visitors can locate exactly the location of what they want to explore with an overview description.
- Provide information: Visitors can view fully information about the event, things that the beacon carries. Visitors can also know the related event, stories.
- Speak Information: The application can speak the information about the events, things on the screen.
- Save information: Visitors can save information about events or things, so they could read them offline even from outside of the museum.
- Navigate: Visitors can view the navigations to know where the beacon that contain the related event to explore.

- *For staff*
  - Track historic item: if the beacon is move out of it proper location, the system will notify for the staff.
- *For admin*
  - Add beacon: Admin can add a new beacon to system to map it with a historic item.
  - Map beacon: Admin can map a beacon with a historic item.

1.3.3 Boundaries of the System

- A visitor who wants to use the functions of this system have to equip enough device includes:
  - A mobile device which has installed our application must support Bluetooth v4.0 BLE and internet connection.
- To do the job, a staff of the system must be equipped the following devices:
  - A mobile device which has installed our application must support Bluetooth v4.0 BLE and internet connection.

- The complete product includes:
  - A mobile application that allow:
    - Confirm ticket code (for Visitor)
    - View museum map (for Visitor)
    - Provide information (for Visitor)
    - Navigate (for Visitor)
    - Save historic item information for viewing offline (for Visitor)
    - Detect Item location (for Visitor)
    - Track historic item (for Staff)
    - Add a new beacon (for Admin)
    - Map a beacon with a historic item (for Admin)
  - A web application that allow:
    - Manage Beacons information (for Admin)
    - Manage historic item (for Admin / for Expert)
    - Process report of visitor (for Admin / for Expert)

- View historic item's movement log (for Admin)
- Setup scheduler (for Admin)

1.3.4 Future Plans

Currently, the system only deploys on a single platform: Android. We design the system to make it easily to scale to be a bigger model with more functions and run on more platform:

- Run on multiple platform on client side: iOS.
- The system can connect with many museums over the world to get information of historic events or things between museums.
- The system is design to easily scale for bigger travel model such as national park, city, etc…
- The mobile application can fully support visually impaired visitors by providing the speech-to-text function in order to search for specific historic events or things, guiding visitors to the items, and automatically read the items information for visitors to listen.
- Visitors can also answer yes/no question quickly by tapping the screen for "Yes" or the Back button of the device for "No".

1.3.5 Development Environment

1.3.5.1 Hardware requirements

- **For server/web development**

Windows	**Minimum Requirements**	**Recommended**
Internet Connection	Cable, Wi-Fi (4 Mbps)	Cable, Wi-Fi (8 Mbps)
Operating System	Ubuntu Server 12 LTS/ Windows Server 2008	Ubuntu Server 14.04.2 LTS/ Windows Server 2008
Computer Processor	Intel® CORE i3 Quad core 2.1 GHz	Intel® CORE i7 Quad core 2.4 GHz
Computer Memory	2GB RAM	4GB or more

Table 2. Hardware Requirement for server/ web development

- **For Estimote Beacon**

Identification (model number etc.)	Estimote model REV.D3.4 Radio Beacon.
Frequency range	2400 MHz to 2483.5 MHz
No. of preset switchable channels	40

No. of voice/data/TV channels	40 Data channels (including 3 advertising channels)
Tx-Rx channel separation	2 MHz
Adjacent channel separation	2 MHz
Frequency stability	<20ppm
2nd Harmonic radiation's	<25 dBuV
Mode of emission	not more than 20 DB
Bandwidth of emission	500 KHz
Type of modulation to be required	GFSK
Power output	4 dBm
Sensitivity	-93 dBm
CPU	32-bit ARM® Cortex M0
Flash memory	256 kB

Table 3. Hardware Requirement for Estimote Beacon

- **For mobile development**

Android	**Minimum Requirements**	**Recommended**
Internet Connection	Wi-Fi (4 Mbps)	Wi-Fi (8 Mbps)
Operating System	Android 4.4: Kitkat	Android 5.1.1: Lollipop
Processor	Snapdragon 400 1.7GHz Dual Core	Snapdragon 600 1.89GHz Quad Core or higher
Memory	512MB RAM	2GB
Bluetooth	Bluetooth 4.0 required	Bluetooth 4.0 required

Table 4. Hardware Requirement for client (Mobile device) development

1.3.5.2 Software requirements

	Name / Version	Description
Modeling tool	Star UML 5.0	Used to implement website and web service
IDE	Netbeans 8.0.2	Programming tools
DBMS	MS SQL Server 2008	Used to create & manage the database for system

Source control	SourceTree 1.8.3.0 or above	Used for source control
Web browser	Chrome 47 or above	Testing browser

Table 5. Software requirements for develop web site and web service

	Name / Version	Description
IDE	Android Studio 2.1.0	Programming tools
Source control	SourceTree 1.8.3.0 or above	Used for source control
Testing OS	Android 5.0.1: Lollipop	Testing Client Operation System

Table 6. Software requirements for develop client application

2. Project organization

2.1 Software Process Model

2.1.1 Overall Description

Agile development are methods allow the development team to focus on the software itself rather on design and documentation. Agile methods universally rely on an incremental approach to software specification, development, and delivery. They are best suited to application development where the system requirements usually change rapidly during the development process.

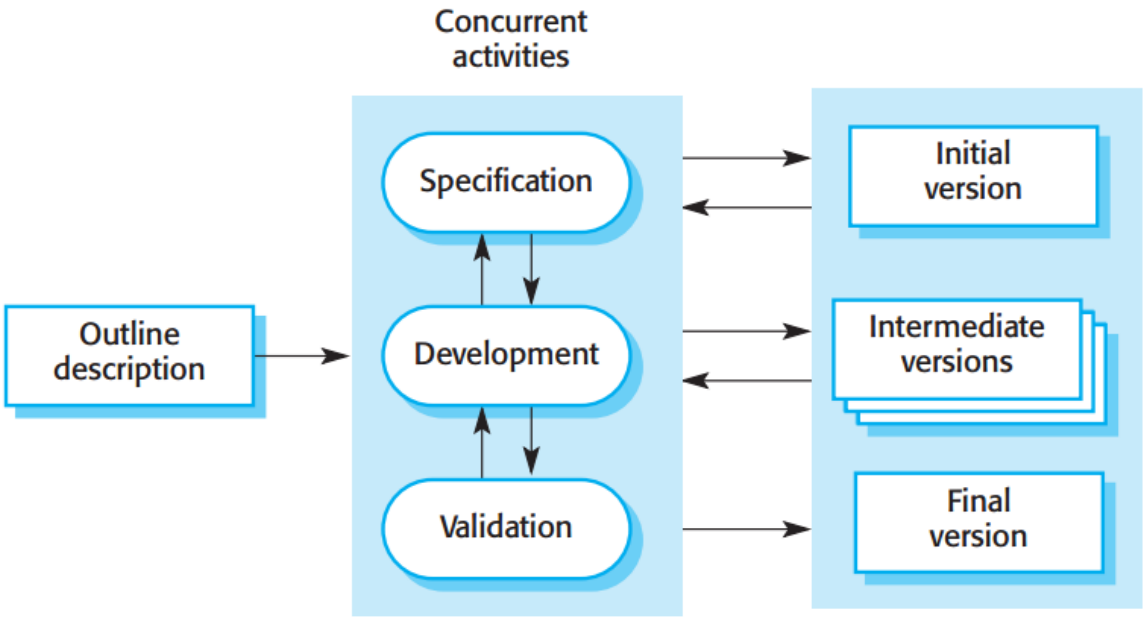
Evolutionary development model is one the models of Agile method. Evolutionary development is an iterative and incremental approach to software development. Instead of creating a comprehensive artifact, such as a SRS, that is reviewed and accepted before creating a comprehensive design model (and so on) developer instead evolve the critical development artifact over time in an iterative manner. Instead of building and then delivering the system is a single time release, developers deliver it incrementally over times.

References:

- Software Engineering, 9/E -Ian Sommerville. Chapter 3: Agile software development page 59
- http://www.agiledata.org/essays/evolutionaryDevelopment.html

2.1.2 Agile Development Method – Evolutionary

Figure 2. Evolutionary development Model



References:

Software Engineering, 9/E -Ian Sommerville. Chapter4 – Software processes Page 68

2.1.3 Reasons for Choosing

The project is developed under Evolutionary Development Model to capable with current situation of our team. We chose this model because of the following reasons:

- The project uses new technology – the Estimote Beacons, team does not sure about what the device can do so the solution and reality technology may mismatch.
- The Estimote Beacons API is not stable.
- Team must do a lot of Estimote Beacon test because of unclear specification.
- Team has no idea about the algorithms that solve the problem of the project.
- Team cannot predict what will happen during the time using the Estimote Beacon

For those reasons, the team must study Estimote Beacons to solve problems and implement project simultaneously.

2.2Roles and responsibilities

No	Full name	Role in Group	Responsibilities
1	Mr. Kiều Trọng Khánh	Product Owner – Technical Expert	• Specify user requirement • Specifying the business • Control the development process • Give advices on techniques, solutions and business analysis support
2	Nguyễn Tấn Phát	Team Leader, BA, DEV, Tester	• Managing process • Clarifying requirements • Researching solutions and techniques • Assigning task for members • Design architecture • Support team members • Reviewing the task result of members • Creating/ Editing documents and reports • Reviewing documents and reports • Designing Mobile application UI • Coding mobile application • Coding Web service • Creating test plan • Creating test case • Testing
3	Trần Quang Tùng	Team Member, BA, DEV, Tester	• Clarifying requirements • Researching solutions and techniques • Design architecture • Designing database • Reviewing documents and reports • Coding Web service • Coding BOM Website • Reviewing test plan • Reviewing test case • Testing

4	Võ Thanh Hiếu	Team Member, BA, DEV, Tester	• Clarifying requirements • Researching solutions and techniques • Design architecture • Designing database • Reviewing documents and reports • Designing Mobile application UI • Reviewing documents and reports • Coding Mobile • Reviewing test plan • Reviewing test case • Testing Coding • Testing
5	Phan Quốc Hùng	Team Member, BA, DEV, Tester	• Clarifying requirements • Researching solutions and techniques • Design architecture • Designing database • Reviewing documents and reports • Designing Mobile application UI • Reviewing documents and reports • Coding Web service • Coding BOM Website • Reviewing test plan • Reviewing test case • Testing Coding • Testing

Table 7. Roles and Responsibilities Details

2.3Tools and Techniques

Tool / Technique	Name /version
Front-end IDE	Android Studio 2.1.0
Back-end IDE	NetBean 8.0.2
Front-end technology	HTML5, CSS, JavaScript, JQuery, Ajax, Android
Back-end technology	MVC, JavaEE, Servlet, JSP
Managing database	SQLite 3, MS SQL Server 2008
Managing the project	SourceTree 1.8.3.0
Managing documents, reports, models and diagrams	Microsoft Office 2013

Table 8. Tools and Techniques

3. Project Management Plan

3.1Software development life cycle

Phase	Description	Deliverables	Resource needed	Dependencies and Constrains	Risks
Specification	- Identify and define system specification in general	-Introduction of proposed system. -General software requirement specification.	Estimate: - Init: 35 man – days - Intermediate: 15 man - days - Final: 10 man - days	N/A	- Lack of member share of understand - Lack of experience.
Development	- Design the current architecture - Choose technology - Implement module	-Task plan -Software design document -Technology notes - Actual software of each module	Estimate: - Init: 25 man – days - Intermediate: 55 man - days - Final: 25 man - days	Base on specification	- Lack of experience. - Code does not work.
Validation	- Integrate modules of system - Release the version - Create test case - Test the version - Note changes.	- Actual software of the whole system - Test case - Changes log / notes	Estimate: - Init: 10 man – days - Intermediate: 30 man - days - Final: 35 man - days	Depend on software of each module	- Modules can't connect with others - Test case doesn't cover all core functions

Phase	Description	Deliverables	Resource needed	Dependencies and Constrains	Risks
Initial Version	-The first version of the project (Implement core flow of project).	- Software specification - Software design - Functioned application	Estimate: 70 man-days	N/A	- Lack of member share of understand
Immediate Version	-Many versions of the project, being changed and updated iteratively	- Updated software specification - Updated software design - Updated application	Estimate: 100 man-days	Base on previous versions	- Lack of experience.
Final Version	- The final version of project	- Final software specification - Final software design - Final application	Estimate: 70 man-days	Base on previous versions	- Bug leakage

Table 9. Software Development Life Cycle Detail

If the result of current version in validation phrase is not satisfied, loop the process for the next version until result of the version is approved.

3.2Phase Detail

3.2.1 Specification

Task	Description	Author
1. Identify and define system specification in general.	Define which main functions system should provide.	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng

Table 10. Phase 1: Specification

3.2.2 Development

	Description	Author
1. Design the current architecture	Design the architecture for the current system bases on current definition of specification.	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng
2. Choose technology	Choose technology to implement the current system	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng
3. Implement modules	Implement modules base the designs and chosen technology	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng

Table 11. Phase 2: Implementation

3.2.3 Validation

Task	Description	Author
1. Integrate all modules of the system	Integrate all separate modules	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng
2. Release the version	Release a version after integrate all modules into a system	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng
3. Create test case	Create test case base current specification which was determined in Specification phrase	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng
4. Test the version	Execute the created test case	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng
5. Note changes	Note the changes in changes log for the next version.	Nguyễn Tấn Phát Trần Quang Tùng Võ Thanh Hiếu Phan Quốc Hùng

Table 12. Phase 3: Validation

3.3Task sheet

Place at folder “Task sheet” in BitBucket with the following URL:
_ https://hungpqse61169@bitbucket.org/tanphat199/imuseum.git (Security: Must be a member of BitBucket Repository)
_CD (./document/tasksheet)

3.4All Meeting Minutes

Place at folder “Meeting minute” in BitBucket with the following URL:
_ https://hungpqse61169@bitbucket.org/tanphat199/imuseum.git (Security: Must be a member of BitBucket Repository)
_CD (./document/meetingMinute)

4. Coding Convention

- Java: Using to develop website and web service.
- Android: Using to develop mobile application. Because team choose android native to develop the mobile application so the coding convention is based on Java.
  Summary:
  - *Naming Conventions:*

  -Variable name should be short yet meaningful. If the name is more than one word, it must be in mixed case, starting word with a lowercase.
  -Constants name should be all uppercase with words separated by underscores.
  -Methods name should be verbs, in mixed case with the first word lowercase, the first letter of each internal word capitalized.
  -Class name should be nouns, in mixed case with the first letter of each internal word capitalized.
  - *Package and import statements:*

  -Package statement is the first non-comment line.
  -Import statement is after package statement.
  - *Constants*

  -Numerical constants should not be coded directly.
  - *Variable Assignments:*

  -Avoid assigning several variables to the same value in a single statement.
  - *Comments:*

  -Using /* */ for block comments
  -Using // for line comments
  - *Return Statements:*

  -A return statement with a value should not use parentheses.

- Using Java coding convention from:
  http://www.oracle.com/technetwork/java/codeconvtoc-136057.html
- Using Android codding convention form:
  https://source.android.com/source/code-style.html

References:

Code Conventions for the Java ™ Programming Language
Revised April 20, 1999

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

C. Software Requirement Specification

1. User Requirement Specification

Actor	Definition
1. Guest	Person uses mobile application or website but not login into system
2. Authenticated User	Person who has logged in into the system, users can use some functions base on their role
3. Staff	Security, who tracks historic items in a specific area
4. Admin	Administrator, who monitors, configures, manipulates the information in the system and assigns a Visitor's report to an Expert.
5. Expert	History expert, who manipulates historic item information, such as approving pending item and editing historic information.
5. Visitor	Person uses mobile application and want to visit the museum
6. Auto Handler	Is a scheduler, a part of the system, do every automatic function according to set up.

Table 13. User Definition

1.1 Guest Requirement

- *Login:* guest uses username and password to login into the system to become staff or admin and use some function base on the role after login.

1.2 Staff Requirement

- *Track Historic Item:* staff uses mobile application for detect which historic item is moved.

1.3 Admin Requirement

- *Get log of item movements:* admin can get all log of item movements and its information.
- **Manipulate report information**
  - *Get Visitors reports:* admin can get visitors reports.
  - *Assign reports:* admin can assign report to an Expert.
- **Manipulate historic item information**
  - *Map historic item:* Map historic item to a Beacon, and set its location.

  - *Search historic item information:* admin can search the historic item information in the system for getting, updating or deleting that information.
  - *Delete historic item information:* permanently remove an existed beacon information out of system.
  - *Recover historic item information:* recover a historic item which had been sent to recycle bin.
- **Manipulate beacon information**
  - *Insert beacon information:* add a new beacon information to system. After adding a new beacon information, admin can map a beacon information with a historic item information.
  - *Map Beacon to Item:* map a beacon information to an existed historic item.
  - *Search beacon information:* admin can search the beacon information in the system for getting, updating, deleting that information or mapping beacon information with a specific historic item information.
  - *Delete beacon information:* remove an existed beacon information out of system.
- **Manipulate crawler configuration**
  - *Set crawler time:* Admin chooses a specific time and which days of week for the crawler to run.
  - *Add site:* add a museum website to system. The crawler will crawl historic item information from added sites.
  - *Remove site:* remove an existed site from system.

1.4 Expert Requirement

- *Get Visitors reports:* Expert can get visitors reports.
- **Manipulate historic item information**
  - *Search historic item information:* admin can search the historic item information in the system for getting, updating or deleting that information.
  - *Insert historic item information:* add a new historic item information to system.
  - *Update historic item information:* update an existed historic item information in system.
  - *Approve historic item information:* approve a pending historic item.
  - *Delete historic item information:* remove an existed beacon information out of system.
  - *Recover historic item information:* recover a historic item which had been sent to recycle bin.

1.5 Visitor Requirement

- *Get museum map:* visitor can get the entire museum map images.
- *Get historic item information:* visitor can use mobile application and get the detail information of a historic item. The historic item information includes historic item name, description, related items and near-by items.
- *Send report:* visitor can use the mobile application to feedback the admin about the historic item information.
- *Add favorite historic item information:* store the information of historic item in local mobile storage for offline viewing.
- *Get favorite historic item information:* visitor can use mobile application and get the detail information of a historic item, which visitor saved before.
- *Get related item location:* get the location of a specific historic item in the museum

1.6 Authenticated user Requirement

- *Logout:* when finish all activities at website staff and admin can log out of system.

1.7 Auto Handler Requirement

- *Simulator generate ticket code:* system generates ticket code for visitors.
- *Calculate historic item similarity:* system defines which historic item in the system relate to another.
- *Detect in-range historic item:* system will detect all in-range historic item when visitor move in the museum.
- *Detect moved historic item:* system will detect which historic item is moved.
- *Notify the report of visitor for admin:* system notify for admin know that visitor has just send a feedback about a historic item.
- *Speak historic item information:* system will read aloud the description of historic item information.
- *Crawl historic item information:* Periodically, the system will get all historic item information from another website and save to database.

2. System Requirement Specification

2.1 External Interface Requirement

2.1.1 User Interface

- The user interface use English and Vietnamese language in website and android mobile application.
- The user interface for website display best on 1024x768 pixels – screen size.
- The user interface for android application is designed base on material design and display best on 480x800 pixels – screen size

2.1.2 Hardware Interface

- Android smartphone supports Bluetooth 4.0 low energy
  - OS: Android (above 4.3)
  - Chipset: Snapdragon 400 1.7GHz Dual Core
  - RAM: 1GB
- Estimote beacons.
  - http://estimote.github.io/Android-SDK/JavaDocs

2.1.3 Software Interface

- Web application: work with Chrome (v47 or above), Internet Explorer (v10 or above), Firefox (v43 or above)
- Mobile application: Android operation system (v4.4 or above) integrate with Estimote beacon SDK (v0.9.4 or above).

2.1.4 Communication Protocol

- Use HTTP protocol 1.1 for communication between the web browser and the web server
- Use HTTP protocol 1.1 for communication between the mobile application and the web server
- Use Bluetooth 4.0 (BLE – Bluetooth Less Energy) protocol for communication between the estimate beacons with the mobile application

2.2System Overview Use Case

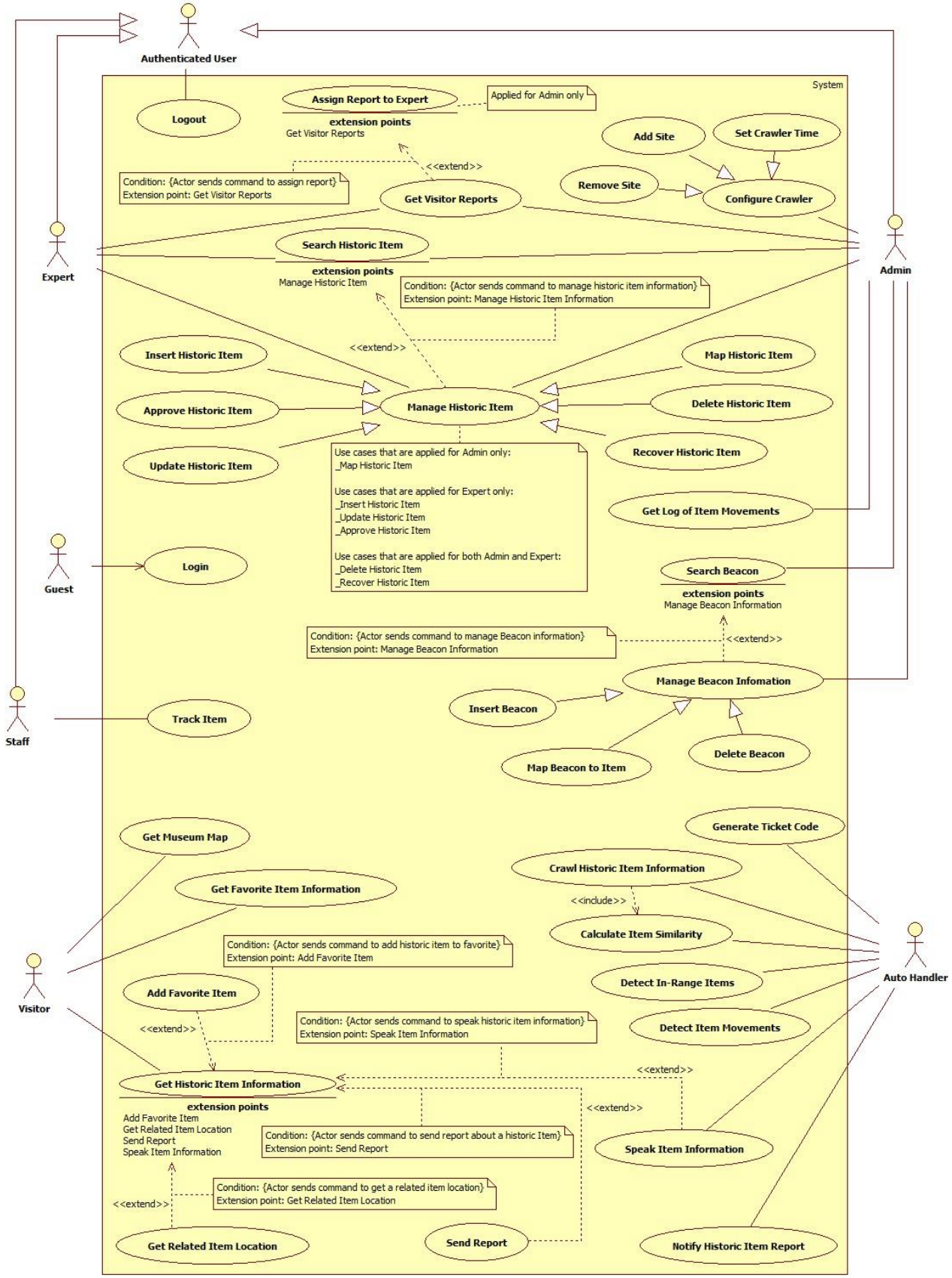


Figure 3. System Overview Use Case

2.3List of Use Case

2.3.1 <Visitor> Overview use case

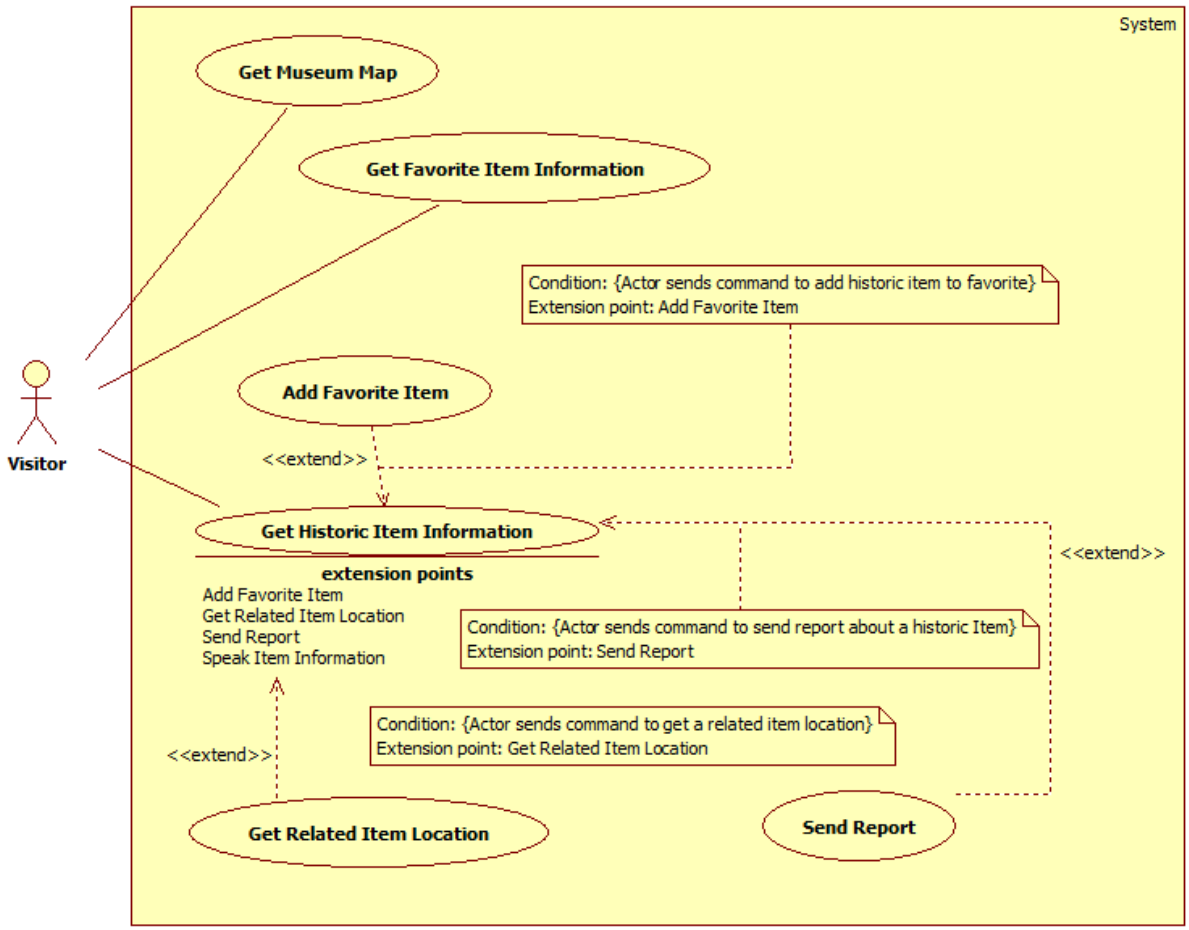


Figure 4. <User> Overview Use Case

2.3.1.1 <Visitor> Get Museum Map

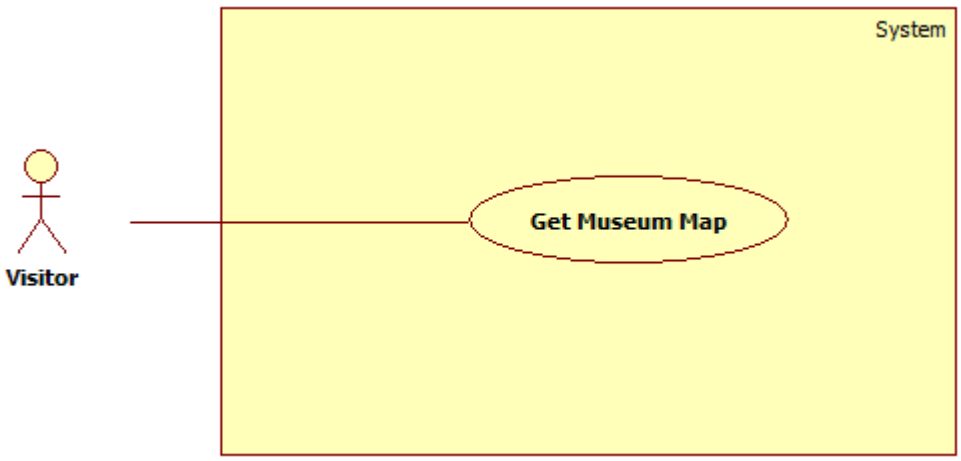


Figure 5. <Visitor> Get Museum Map

USE CASE – UC_V01			
Use Case No.	UC_V01	**Use Case Version**	1.0
Use Case Name	Get Museum Map		
Author	PhatNT		
Date	24/05/2016	**Priority**	Normal

Actor:

- Visitor

Summary:

- This use case allows user to get all overview map of the museum and name of all historic items in a specific floor.
- This use case applies for Android Mobile Application.

Goal:

- Visitor can get and see all the image map of all floor in the museum.
- Visitor can get and see name of all historic items of a specific floor in the museum.

Triggers:

- Visitor sends “Get Museum Map” command.

Preconditions:

- Actor has accessed the system under Visitor role with valid ticket code.

Post Conditions:

- **Success:**
  - If there are image maps in the system, image of all floors in the museum will be displayed.
  - If there is no image map in the system, message “The museum has no map” will be shown.
- **Fail:** fail message will be shown depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Visitor sends Get Museum Map command.	- System will get and show:

		• The list of floor images in the format. • Button (Command) to get all historic item's name in current floor. [Alternative 1] [Alternative 2]
2	Visitor sends Get Historic Item's Name in Current Floor command.	- System shows all historic item's name in format. [Alternative 3]

Alternative Scenario 1:

No.	Cause	System Response
1	Visitor sends Get Museum Map command.	- The message "There is no image map" will be shown.

Alternative Scenario 2:

No.	Cause	System Response
1	Visitor cancels Get Museum Map while system is processing.	- System stops get the list of floor images. - The message "You have cancelled get museum map" will be shown.

Alternative Scenario 3:

No.	Cause	System Response
1	Visitor sends Get Historic Item's Name in Current Floor command and there is no available historic item in selected floor.	- The message "Have no available items!" will be shown.

Exceptions: N/A

Relationships: N/A

Business Rules:

- Map are images of a floors which supplied by museum.
- Name of historic item will be show in format:
    - ***Vietnamese Name* | *English Name***

- System will download and save all museum map in mobile local storage.
- System will not download all museum map in the next time request if museum map is not changed. (Map changed: the image URLs of the museum map is different from the last time).
- Mobile application will scale down the quality of image before showing image to improve image load performance.

Table 14. Get Museum Map Specification

2.3.1.2 <Visitor> Get Historic Item Information

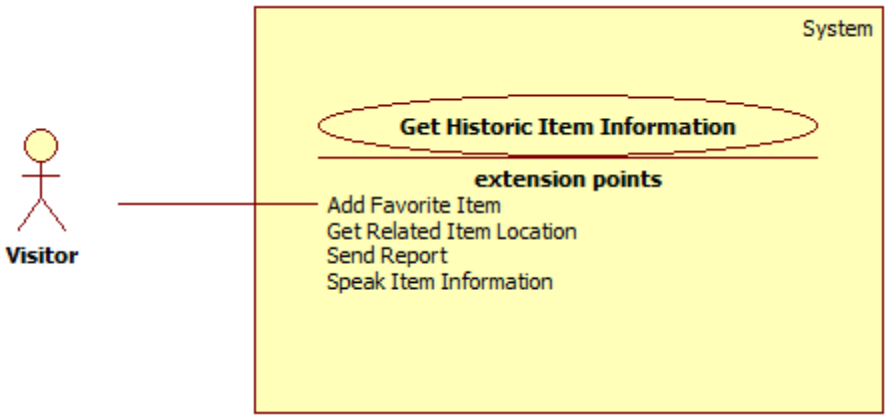


Figure 6. <Visitor> Get Historic Item Information

USE CASE – UC_V02			
Use Case No.	UC_V02	**Use Case Version**	1.0
Use Case Name	Get Item Information		
Author	PhatNT		
Date	24/05/2016	**Priority**	Normal

Actor:

- Visitor

Summary:

- This use case allows user to get all information of a specific historic item in the museum.
- This use case applies for Android Mobile Application.

Goal:

- Visitor can get and see all information of a specific historic item that they want.
- After getting all information of historic item, user can read, add to favorite or report information.

Triggers:

- Visitor sends “Get Historic Item Information” command.

Preconditions:

- Actor has accessed the system under Visitor role with valid ticket code.
- Actor must be in-range of a specific historic item.

Post Conditions:

- **Success:** all historic item information will be displayed on mobile screen.
- **Fail**: error message will be shown depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Visitor selects a historic item and send Get Historic Item Information command.	- System will get all historic item information and show them on the screen. - All historic item information include: - Historic item image - Historic item name in which language that visitor setup before. - Historic item description in which language that visitor setup before. - List of related Historic Item. [Alternative 1] [Alternative 2] [Alternative 3]

Alternative Scenario 1:

Step.	Cause	System Response
1	Visitor cancels Get Historic Item Information while system is processing.	- System stops get historic item information. - The message “You have cancelled get museum map” will be shown.

Alternative Scenario 2:

Step.	Cause	System Response
1	Visitor sends request to show English language.	

		- System show name and description of historic item in English.

Alternative Scenario 3:

Step.	Cause	System Response
1	Visitor sends request to show Vietnamese language.	- System show name and description of historic item in Vietnamese.

Exceptions: N/A

Relationships: Extension points of:

- Add favorite item
- Get related item location
- Send report
- Speak item information

Business Rules:

- The information of historic item includes:
  - *Historic item image:* system will scale down the quality of image before showing image to improve image load performance.
  - *Historic item name:* system will specify and show full name of historic item; name of a historic item will be shown in which language that visitor required.
  - *Historic item description:* system will specify and show full of historic item description; description of a historic item will be shown in which language that visitor required.
  - *List of related Historic Item*: system will get and show at most five rows of five related items of the chosen historic item. Each row is clickable to get the location of related historic item.

- When Visitor uses the application for the first time, the images of historic item will be downloaded into mobile local storage for showing in the later request.

Table 15. Get Item Information Specification

2.3.1.3 <Visitor> Report Item Information

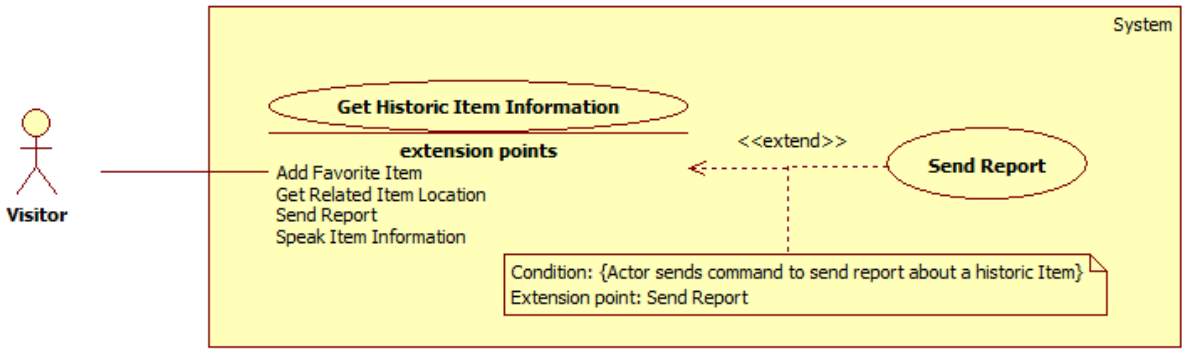


Figure 7. <Visitor> Report Item Information

USE CASE – UC_V03			
Use Case No.	UC_V03	**Use Case Version**	1.0
Use Case Name	Report Historic Item Information		
Author	PhatNT		
Date	24/05/2016	**Priority**	Normal

Actor:

- Visitor

Summary:

- This use case allows user to notify incorrect historic item information for admin.
- This use case applies for Android Mobile Application.

Goal:

- Visitor can send a feedback about the information of a historic item for admin to review and edit.

Triggers:

- Visitor sends “Report Historic Item Information” command.

Preconditions:

- Actor has accessed the system under Visitor role with valid ticket code.
- Actor got historic item information.

Post Conditions:

- **Success:**
  - Report detail will be saved.

- Message “Your report has been sent, thanks for your supporting information!” will be shown.

- **Fail:** Fail message will be shown depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Guest goes to report view.	System requires information: - *Message*: blank text input, required, length [1, 2000]
2	Guest inputs information	System show the input information in suitable text input.
3	Visitor sends Report Historic Item Information command. [Alternative 1] [Alternative 2]	- System will save report information. - Message “Your report has been sent, thanks for your supporting information!” will be shown.

Alternative Scenario 1:

Step.	Cause	System Response
1	Visitor leaves the message blank, or input more than 2000 characters	System does not allow Visitor to send “Send report” command.

Alternative Scenario 2:

Step.	Cause	System Response
1	Visitor cancels Report Historic Item Information.	

		System turns off report view and redirect to historic item information view.

Exceptions: N/A

Relationships:

Extend from get historic item information.

- Condition: actor send report historic item information.
- Extension point: Report Historic Item Information

Business Rules:

- Message of report is required when visitor want to report historic item information.
- System will save the report detail in local even though it is successful or fail.
- In case report failed to send, system will send all of them again in the next time visitor uses the application if Internet connection is available.
- The format of local data is:

ID	Date	Message	ItemId	Status

- ID is auto-increment field
- Date: system will get the current time of reporting and save in millisecond.
- Message: the message that visitor want to send to admin.
- ItemId will be specified by system.
- Status: there are two status type: “Pending” and “Success”
  - Initial status is “Pending”.
  - If visitor completed report process and received response from system, the status will change from “Pending” to “Success”.

Table 16. Report Item Information Specification

2.3.1.4 <Visitor> Add Favorite Item

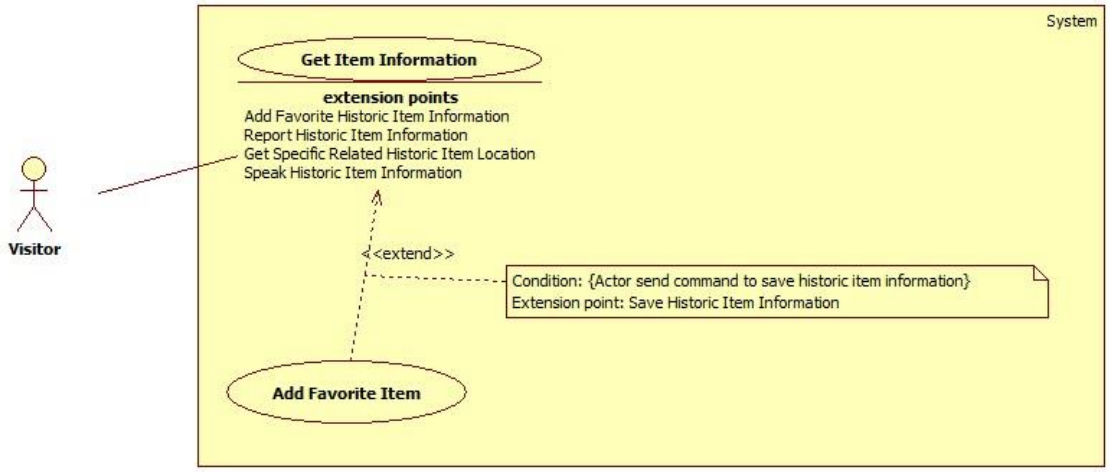


Figure 8. <Visitor> Add Favorite Item

USE CASE – UC_V04			
Use Case No.	UC_V04	**Use Case Version**	1.0
Use Case Name	Add Favorite Item		
Author	PhatNT		
Date	24/05/2016	**Priority**	Normal

Actor:

- Visitor

Summary:

- This use case allows user to add historic item information that they like and want to see offline.
- This use case applies for Android Mobile Application.

Goal:

- Visitor can store the historic item information for reading without the internet connection.

Triggers:

- Visitor sends “Add Favorite Item” command.

Preconditions:

- Actor has accessed the system under Visitor role with valid ticket code.
- Actor got historic item information.

Post Conditions:

- **Success:**
  - Historic item information will be saved into mobile local storage.
  - Message “Add Historic Item Information Successfully!” will be shown.
- **Fail:** Fail message will be shown depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Visitor sends Add Favorite Item command. [Alternative 1]	- System requests visitor chooses which category that they want to add.
2	Visitor chooses category and send “Add Favorite Item” request. [Alternative 2] [Alternative 3] [Alternative 4] [Alternative 5]	- System will save the historic item information into internal storage. - Message “Save Historic Item Information Successfully!” will be shown.

Alternative Scenario 1:

Step.	Cause	System Response
1	Historic item have already added to favorite.	System does not allow Visitor to send “Add favorite Item” command.

Alternative Scenario 2:

Step.	Cause	System Response
1	Visitor leaves the message blank, or input more than 200 characters	System does not allow Visitor to send “Add favorite Item” command.

Alternative Scenario 3:

Step.	Cause	System Response
1	Category name have already added	System does not allow Visitor to send “Create new category” command.

Alternative Scenario 4:

Step.	Cause	System Response
1	Visitor cancel Add Favorite Item.	- System turn off “Add Favorite Item” view and redirect to historic item information view.

Alternative Scenario 5:

Step.	Cause	System Response
1	Visitor sends “Create new category” request	System requires information: - *Name*: blank text input, required, length [1, 200]

2	Visitor sends “Add new category” request	System saves new category and return to “Add Favorite Item” for visitor to choose category and add item.

Exceptions: N/A

Relationships:

Extend from get historic item information.

- Condition: Visitor sends command to add favorite item.
- Extension point: Add Favorite Historic Item Information.

Business Rules:

- For each valid ticket code, Visitor can save at most three historic items.
- The historic item will be saved into mobile local storage in format:

ID	Name	Description	ImageURL	LocalImagePath	CategoryId

- ID is auto-increment field
- Name: system will specify name of historic item.
- Description: system will specify description of historic item.
- ImageUrl: system will specify the Image URL of historic item.
- LocalImagePath: image path after system download image base on ImageURL done.
- CategoryId: id of category after visitor add catrgory.

Table 17. Save Favorite Item Specification

2.3.1.5 <Visitor> Get Related Item Location

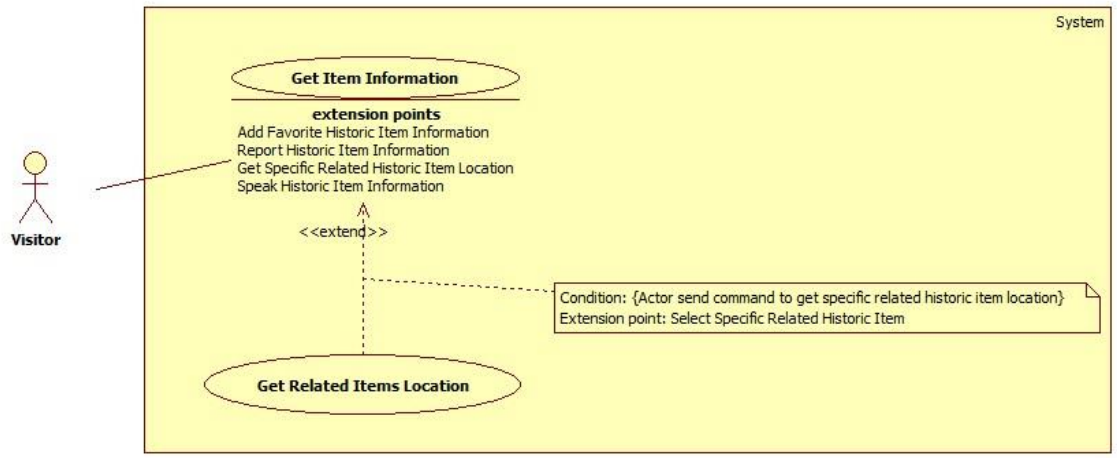


Figure 9. <Visitor> Get Related Item Location

USE CASE – UC_V06			
Use Case No.	UC_V06	**Use Case Version**	1.0
Use Case Name	Get Related Item Location		
Author	PhatNT		
Date	28/05/2016	**Priority**	Normal

Actor:

- Visitor

Summary:

- This use case allows user to get the location of a specific related historic item.
- This use case applies for Android Mobile Application.

Goal:

- Visitor can get and see the location detail (include floor, room, area and list nearby items) of a specific historic item that visitor want.

Triggers:

- Visitor send “Get Related Item Information” command.

Preconditions:

- Actor has accessed the system under Visitor role with valid ticket code.
- Actor got historic item information.

Post Conditions:

- **Success:** all location information about related historic item will be displayed.
- **Fail**: message will be shown depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Visitor selects a specific historic item and send “Get Related Item Location“ command.	- System will get all historic item information and show them on the screen. - All historic item location information include: - Floor location: label - Room location: label - Area location: label - Nearby historic item: label.
2	Visitor sends “Start Direction” command	- System will check all the historic item around and notify the visitor if the requested historic item is near. [Alternative 1]

Alternative Scenario 1:

Step.	Cause	System Response
1	Visitor sends request to cancel direction.	- System will stop checking all of historic item around the visitor.

		- System will redirect to historic item information which visitor requested before.

Exceptions: N/A

Relationships:

Extend from get historic item information.

- Condition: Actor send command to get related historic item location.
- Extension point: Get related historic item location.

Business Rules:

- System will specify the location of historic item and fill it into Floor, Room, Area and Nearby Historic Item.
    - If a field is not available, it will not be included in location message.
- When visitor sends “Start Direction” command:
    - System will continue range and check all the historic item information around the visitor.
    - If visitor come near to historic item, system will notify for visitor by vibrating the smartphone.
    - General information of historic item (image and name) will be show to visitor.

Table 18. Get Related Item Location Specification

2.3.2 <Admin> <Expert> Overview use case

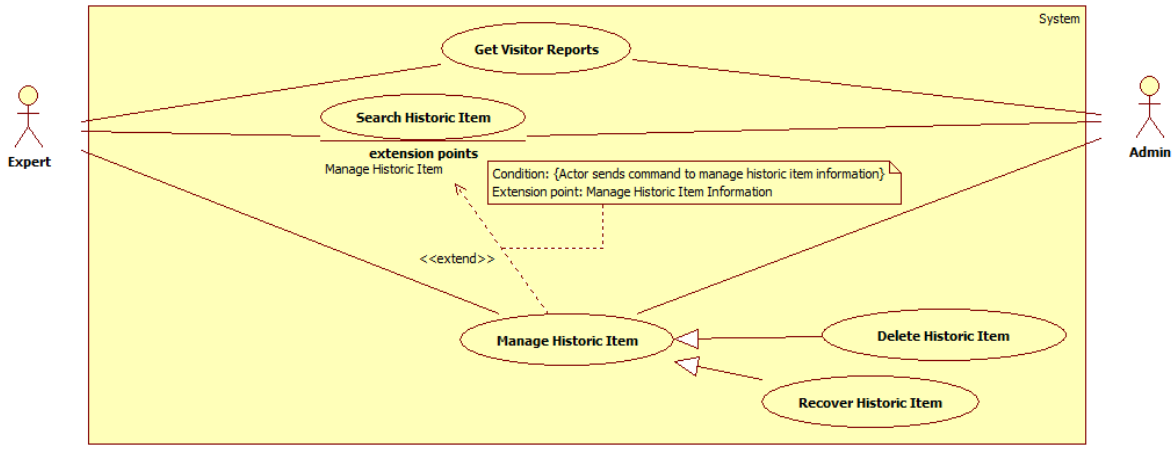


Figure 10. <Admin> <Expert> Overview use case

2.3.2.1 Get Visitor Reports

Figure 11. <Admin> <Expert> Get Visitor Reports

USE CASE – UC_AE01			
Use Case No.	UC_AE01	**Use Case Version**	1.2
Use Case Name	Get Visitor Reports		
Author	HungPQ		
Date	23/05/2016	**Priority**	Normal
Actor: - Admin - Expert **Summary:** - This use case allows Admin or Expert to get the list of Visitors Feedbacks.			

- This use case applies for Web application.

Goal:

- Actor can see the list of reports from visitors about the historic item information and review it.

Triggers:

- Actor sends “Get Visitors Reports” command.

Preconditions:

- Actor must login at Admin or Expert role.

Post Conditions:

- **Success:** The Visitors Feedbacks is successfully shown to Actor.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Actor goes to “Visitor Reports” view. [Alternative 1]	System will get and show all Visitor Reports. One record in list include (in order): - Number (label) - Item Name (label) - Message (label) - Time (label)

Alternative Scenario 1:

No	Actor Action	System Response
1	There is no Visitor feedback	System shows messages “There is no report from visitors!”

Exceptions: N/A
Relationships: N/A
Business Rules:
- The list of reports is sorted in descending order by time.
- The time format is: day/month/year hour:minute:second.
  o Example: 08/08/2016 12:21:00
- The list is divided into 2 sections: unseen reports and all reports.
  o Unseen reports: reports which haven't been clicked by Admin.
  o All reports: All reports including seen and unseen reports.
- Reports: While exploring the museum, Visitors can send feedbacks about a specific historic items. Reports could be about: Incorrect historic information, Application shows a different historic items, etc.
- Notifications: If there is a new feedback from visitors, System will notify immediately.

Table 19. Get Visitor Reports Specification

2.3.2.2 Search Historic Item

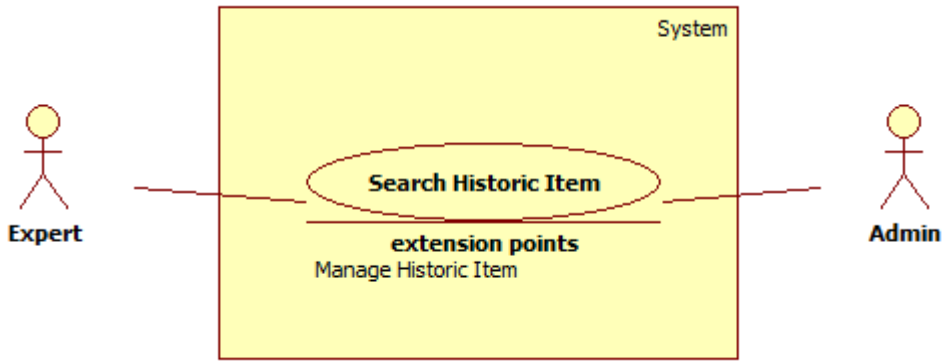


Figure 12. <Admin> <Expert> Search Historic Item

USE CASE – UC_AE02			
Use Case No.	UC_AE02	**Use Case Version**	1.2
Use Case Name	Search Historic Item Information		
Author	HungPQ		
Date	24/05/2016	**Priority**	Normal

Actor:

- Admin
- Expert

Summary:

- This use case allows Admin or Expert to search for historic items information.
- This use case applies for web application.

Goal:

- Actor sees the list of items whose name is similar to the entered keyword.

Triggers:

- Actor sends “Search Historic Items Information” command.

Preconditions:

- Actor logged in under Admin or Expert role.

Post Conditions:

- **Success:**
  - If system found historic items information base on keyword, list of historic items information will be displayed.
- **Fail:** N/A

Main Success Scenario:

Step	Actor Action	System Response
1	Actor enters keyword and sends “Search Historic Items Information” command	System will get and show items list based on keyword. One record in list include (in order): - Number (label) - Item name (label) - Item location (label) - Item status (label) [Alternative 1]

Alternative Scenario 1:

Step.	Cause	System Response
1	Actor enters keyword which do not match any items and send command	System shows message “No result is found!”

Exception: N/A

Relationships: Extension point of Manage Historic Item

Business Rules:

- System gets all of the historic item information whose name is similar to the keyword, ignoring cases and Vietnamese accent mark.
  - Example: “doc lap” could relate to “Bản Tuyên Ngôn Độc Lập”
- The historic item search result list is sorted in descending order by ItemID.
  - ItemID: the number that identifies the historic item, which is automatically generated by the system.
- System will specify the location of historic item, contains Area, Room and Floor.
- Item Status: show if Item is mapped to a Beacon or not.

Table 20. Search Historic Item Specification

2.3.2.3 Delete Historic Item

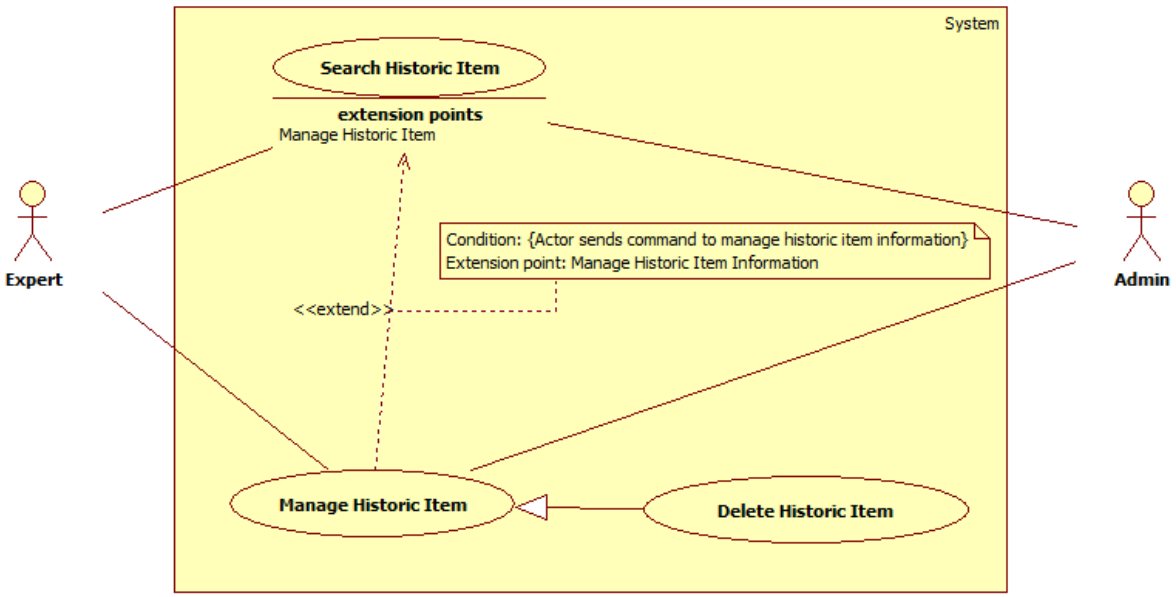


Figure 13. <Admin> <Expert> Delete Historic Item

USE CASE – UC_AE03			
Use Case No.	UC_AE03	**Use Case Version**	1.2
Use Case Name	Delete Historic Item		
Author	HungPQ		
Date	24/05/2016	**Priority**	High

Actor:

- Admin
- Expert

Summary:

- This use case allows Admin or Expert to delete a chosen historic item out of system.
- This use case applies for web application.

Goal:

- Actor can remove a specific historic item information out of the system.

Triggers:

- Actor sends “Delete historic item” command.

Preconditions:

- Actor logged in under Admin or Expert role.

Post Conditions:

- **Success:**
  - Item is successfully removed
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
2	Actor chooses a historic item and sends “Delete Historic Item” command.	System shows the dialog for admin to confirm the action.

3	Actor sends confirm command. [Alternative 1] [Alternative 2]	System deletes chosen historic item. System redirects to the previous view. [Exception 1]

Alternative Scenario 1:

No.	Actor Action	System Response
1	Admin cancels “Delete Historic Item” command.	System closes the dialog message. System will redirect to “Manage Historic Item” view.

Alternative Scenario 2:

No.	Actor Action	System Response
1	The chosen historic item is mapped with a Beacon	System will not allow Admin to send “Delete Historic Item command”.

Exception 1:

No.	Actor Action	System Response
1	The chosen historic item does not exist (Example: Have already deleted by someone else)	System show error message.

Relationships: Extend from “Search Historic Items” Use case

- Condition: Actor sends command to Delete Historic Item.
- Extension point: Manage Historic Item Information.

Business Rules:

- System will find and remove specific historic item information out of system based on its ID.
- If delete successfully:
  - The Beacon, which used to map with the removed historic item will change its status from “Mapped” to “Not Mapped” and another historic item can choose this Beacon.
  - The connection between the historic item and its related item will also be removed.
  - The connection between the historic item and its near item will also be removed.
- There are 2 types of delete:
  - ***Soft delete:*** If the Item is deleted for the first time, it will be disabled but still exist in the recycle bin and can be recovered (see at UC_AE04)
  - ***Hard delete:*** If the item is deleted while it was in the recycle bin, item will be removed permanently.

Table 21. Delete Historic Item Specification

2.3.2.4 Recover Historic Item

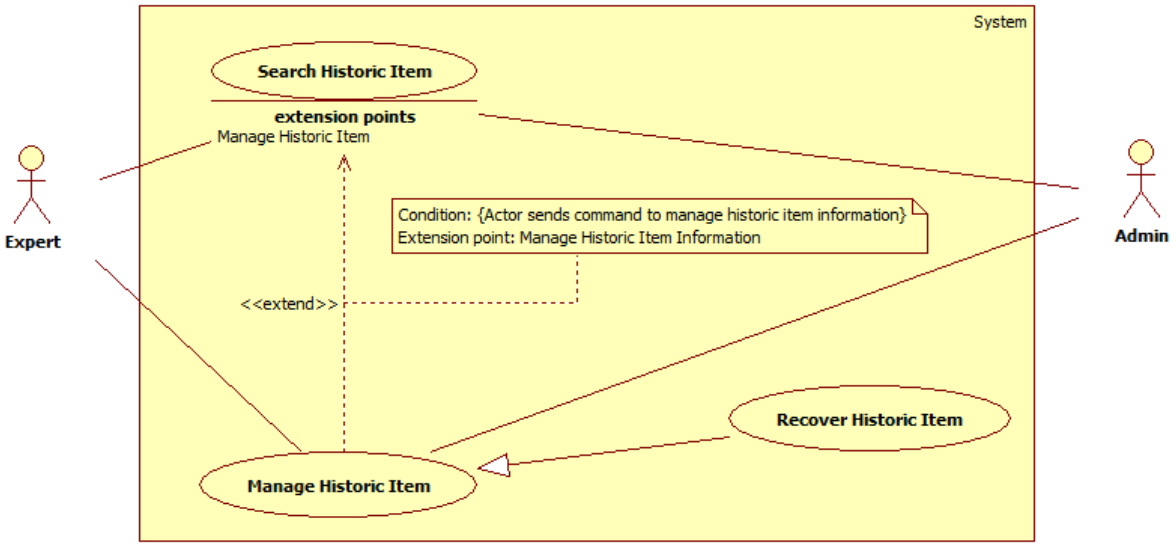


Figure 14. <Admin> <Expert> Recover Historic Item

USE CASE – UC_AE04			
Use Case No.	UC_AE04	**Use Case Version**	1.0
Use Case Name	Recover Historic Item		
Author	HungPQ		
Date	24/07/2016	**Priority**	Medium

Actor:

- Admin
- Expert

Summary:

- This use case allows admin or expert to recover a deleted historic item information.
- This use case applies for web application.

Goal:

- Actor can recover a historic item after it was deleted.

Triggers:

- Actor sends “Recover Historic Item” command.

Preconditions:

- Actor logged in under Admin or Expert role.

Post Conditions:

- **Success:**
  - Historic item information is recover.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Actor accesses “Item Recycle Bin” view	System shows a list of deleted historic items. One record in list include (in order): - Number (label) - Item name (label)
2	Actor chooses item and sends “Recover Historic Item” command.	System recovers item information.

		System redirects to “Item Recycle Bin” view. [Exception 1]

Alternative Scenario: N/A

Exception 1:

No.	Actor Action	System Response
1	The chosen historic item does not exist (Example: Have already deleted by someone else)	System show error message.

Relationships: Extend from “Search Historic Items” Use case

- Condition: Actor sends command to Recover Historic Item.
- Extension point: Manage Historic Item Information.

Business Rules:

- If the Item is deleted for the first time, it will be disabled but still exist in the recycle bin and can be recovered.
- The item in recycle bin has “Deleted” status. After recovered, its status will be changed into “Active”.

Table 22. Recover Historic Item Specification

2.3.3 <Admin> Overview use case

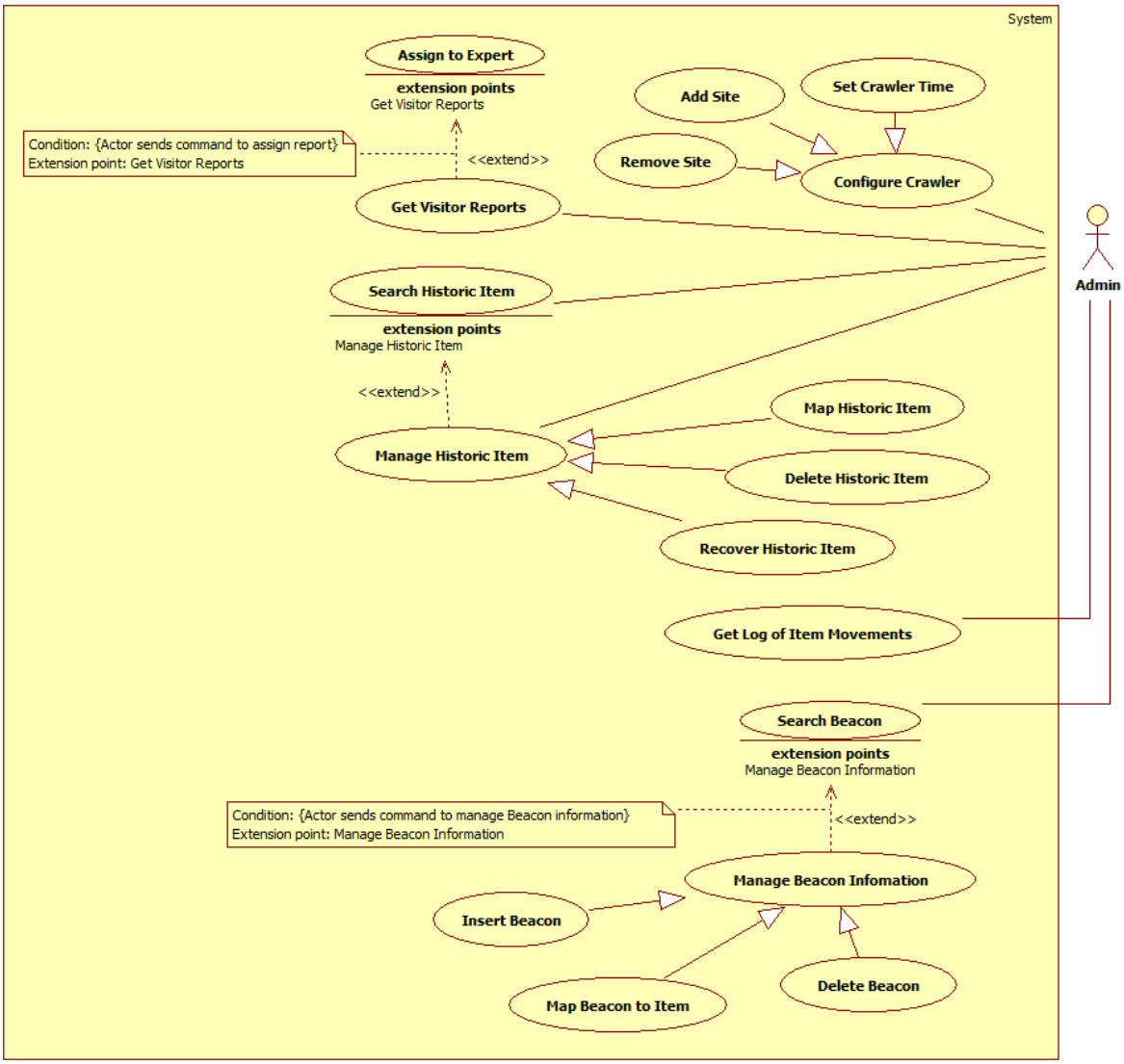


Figure 15. <Admin> Overview Use case

2.3.3.1 <Admin> Search Beacon

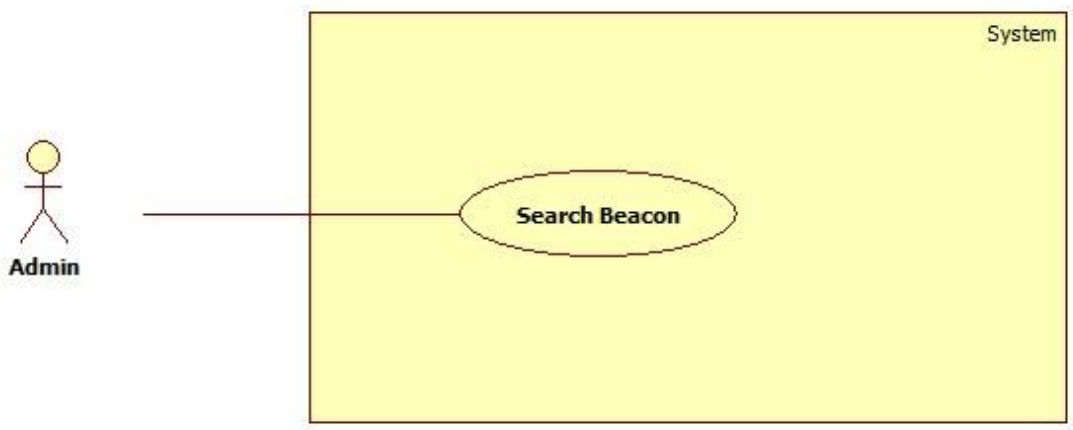


Figure 16. <Admin> Search Beacon

USE CASE – UC_A01			
Use Case No.	UC_A01	**Use Case Version**	2.0
Use Case Name	Search Beacon		
Author	HungPQ		
Date	24/05/2016	**Priority**	Normal

Actor:

- Admin

Summary:

- This use case allows user to search Beacon Information in system.
- This use case applies for web application.

Goal:

- Admin can see the list of Beacon information whose name is similar to the entered keyword.

Triggers:

- Admin enters a keyword and sends “Search Beacon Information” command.

Preconditions:

- Actor logged in under Admin role.

Post Conditions:

- **Success:** A list of Beacon Information that have MAC Address agreed with keyword will be displayed.
- **Fail:** N/A

Main Success Scenario:

Step	Actor Action	System Response
1	Admin sends “Search Beacon Information” command with non-empty keyword	System will get and show Beacon list based on keyword. One record in list include (in order): - Number: label - Beacon MAC Address: label

	[Alternative 1]	- Beacon status: label [Exception 1]

Alternative Scenario 1:

No.	Cause	System Response
1	Admin enters keyword which do not match any Beacon and send “Search Beacon Information” command	System shows message “No result is found!”

Exception: N/A

Relationships: N/A

Business Rules:

- System gets all of the Beacon information whose MAC Address is similar to the keyword into a list.
- The Beacon search result list is sorted in descending order by BeaconID.
  - BeaconID: the number that identifies the Beacon, which is automatically generated by the system.
- System will specify the information:
  - Beacon MAC Address are set default by Estimote Beacon. This information is used to distinct one Beacon with another.
  - Beacon status: determine if the Beacon have or have not yet been chosen by any historic item.
  - There are two type of beacon status: “Available” determine if the Beacon have been chosen by a specific historic item and “Not available” for otherwise

Table 23. Search Beacon specification

2.3.3.2 <Admin> Map Beacon

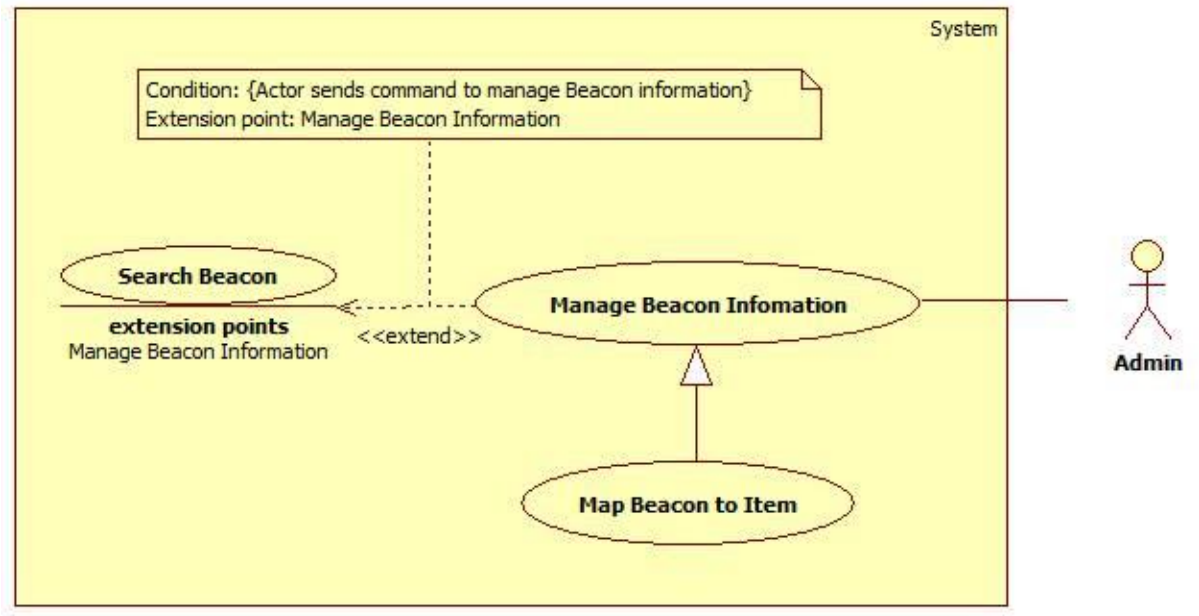


Figure 17. <Admin> Map Beacon

USE CASE – UC_A03			
Use Case No.	UC_A03	**Use Case Version**	1.2
Use Case Name	Map Beacon		
Author	PhatNT		
Date	24/05/2016	**Priority**	High

Actor:

- Admin

Summary:

- This use case allows admin to connect a historic item with a beacon.
- This use case applies for Android mobile and Web application.

Goal:

- Admin can specify which beacon will connect to historic item and location for that historic item.

Triggers:

- Admin selects “Un-Map Beacon”
- Admin sends “Map Beacon” command.

Preconditions:

- Actor logged in under Admin role.
- There must be existed historic items with location.

Post Conditions:

- **Success:**
  - Beacon and Historic Item connected successfully.
  - Location of Historic Item is specified.

- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Admin accesses “In-range Beacon List” [Alternative 1]	System shows the list of In-Range Beacons included: - New Beacon - Un-Map Beacon They are sorted in ascending order by distance.
2	Admin chooses a “Un-Map Beacon” in “In-range Beacon List” and accesses “Beacon Details” view [Alternative 2]	System lists out information of Beacon: - Beacon Mac Address: label. System required information: - *Historic item name*: auto complete text
3	Admin specify required information and sends “Map Beacon” command.	- System updates item information. - System shows message “Beacon has been mapped successfully!” - System redirects to “In-range Beacon List” page. - System updates status of that beacon to “Mapped Beacon”

Alternative Scenario 1:

No	Actor Action	System Response
1	There is no in-range Beacon	System shows messages “There is no Beacon around”

Alternative Scenario 2:

No	Actor Action	System Response
1	Admin chooses a “Mapped Beacon” in “In-range Beacon List” and accesses “Beacon Details” view	System shows messages “Beacon has mapped with an item!”

Exceptions 1:

No.	Cause	System Response
1	The Historic Item Name field is blank	System shows messages “Please specify historic item name!”

Relationships: Extend from “Search Beacon” Use case

- Condition: Admin sends command to Map Beacon.
- Extension point: Manage Beacon Information

Business Rules:

- System will specify the information:

○ Beacon MAC Address: is set default on Estimote Beacon Cloud. This information are used to distinct one Beacon with another. ○ Historic item name: name of an historic item, admin can type a few of character and system will suggest full name of historic item. ○ There are three type of beacon status: ▪ "New": beacon is not exist in system database. ▪ "Un-Map" beacon is existed in system database but not connect with any historic item. ▪ "Mapped" beacon is existed in system database and not connect with any historic item. ○ By default, the status of the new Beacon is "Available". - Historic Item Name: is unique and should present a historic item. - System will insert all the Beacon information.

Table 24. Map Beacon specification

2.3.3.3 <Admin> Delete Beacon

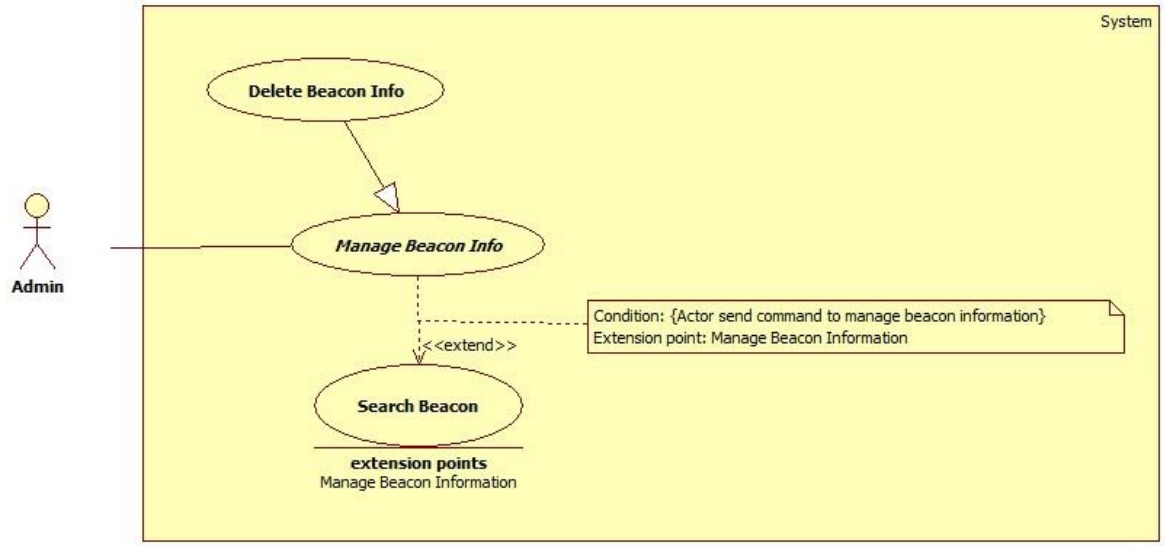


Figure 18. <Admin> Delete Beacon

USE CASE – UC_A05			
Use Case No.	UC_A05	**Use Case Version**	1.2
Use Case Name	Delete Beacon		
Author	HungPQ		
Date	24/05/2016	**Priority**	Normal

Actor:

- Admin

Summary:

- This use case allows admin to delete a chosen beacon from system.
- This use case applies for Web application.

Goal:

- Admin can remove an existed beacon out of the system.

Triggers:

- Admin sends “Delete Beacon” command.

Preconditions:

- Actor logged in under Admin role.
- Chosen beacon status must be “Not Mapped”.

Post Conditions:

- **Success:** Beacon is successfully removed.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Admin accesses “Beacon Management” View [Alternative 1]	System lists out information of Beacon: - Number: label - Beacon Mac Address: label - Beacon status: label
2	Admin selects a Beacon and sends “Delete Beacon” command.	System shows the alert for admin to confirm delete or not.
3	Admin sends confirm command. [Alternative 2]	System deletes beacon. System will redirect to “Beacon Management” view. [Exception 1]

Alternative Scenario 1:

No	Actor Action	System Response
1	There is no Beacon in list	System shows messages “You have no Beacon!”

Alternative Scenario 2:

No.	Actor Action	System Response
1	Admin sends cancel command.	System closes the dialog message. System will redirect to “Beacon Management” view.

Exception 1:

No.	Actor Action	System Response
1	The chosen beacon does not exist (Example: Have already deleted by someone else)	System show error message.

Relationships: Extend from “Search Beacon” Use case

- Condition: Admin sends command to Delete Beacon.
- Extension point: Mange Beacon Information.

Business Rules:

- System will find and remove specific beacon information out of system based on its ID.
- If delete successfully, the connection between the Beacon with the historic item will also be remove.
- Beacon status: There are 2 status of Beacon:
    - Available: Beacon is mapped to a historic Item.

○ Not Available: Beacon is not mapped to any historic item. - To be deleted, Beacon's status must be "Available". - There are 2 types of delete: ○ ***Soft delete:*** If the Beacon is deleted for the first time, it will be disabled but still exist in the recycle bin and can be recovered ○ ***Hard delete:*** If the Beacon is deleted while it was in the recycle bin, it will be removed permanently.

Table 25. Delete Beacon specification

2.3.3.4 <Admin> Get Item Movements Log

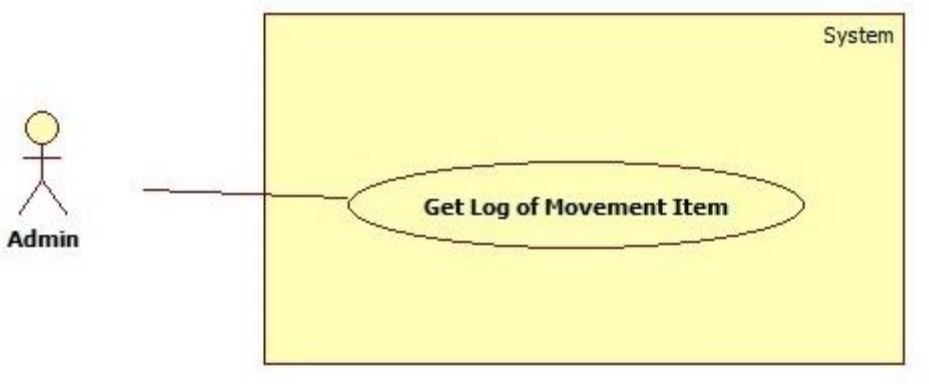


Figure 19. <Admin> Get Item Movements Log

USE CASE – UC_A06			
Use Case No.	UC_A06	**Use Case Version**	1.2
Use Case Name	Get Log of Item Movements		
Author	HungPQ		
Date	23/05/2016	**Priority**	High

Actor:
- Admin

Summary:
- This use case allows Admin to get the list of Item movements log.
- This use case applies for Web application.

Goal:
- Admin can see the list of Item movements log.

Triggers:

- Admin sends “Get Item Movements Log” command.

Preconditions:
- Actor must login at Admin role.

Post Conditions:
- **Success:** The historic item movements log is successfully shown to Admin.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Admin sends “Get Item Movements Log” command. [Alternative 1]	System will get and show all logs of items movements. One record in list include (in order): - Number (label) - Item Name (label) - Status - Notify time (label)

Alternative Scenario 1:

No	Actor Action	System Response
1	There is no record in Item Movements log	System shows messages “There is no movements records!”

Exceptions: N/A

Relationships: N/A

Business Rules:
- The list is sorted in descending order by time.
- The time format is: day/month/year hour:minute:second
- Movements:
  - Every Item movement will be logged by Staff mobile application.
  - See more: use case UC_S01

- Status: there're 2 types of report
  o Not Checked: The item was moved but there's no response from staff.
  o Checked: The item was checked by a staff.

Table 26. Get Item Movements Log specification

2.3.3.5 <Admin> Map Historic Item

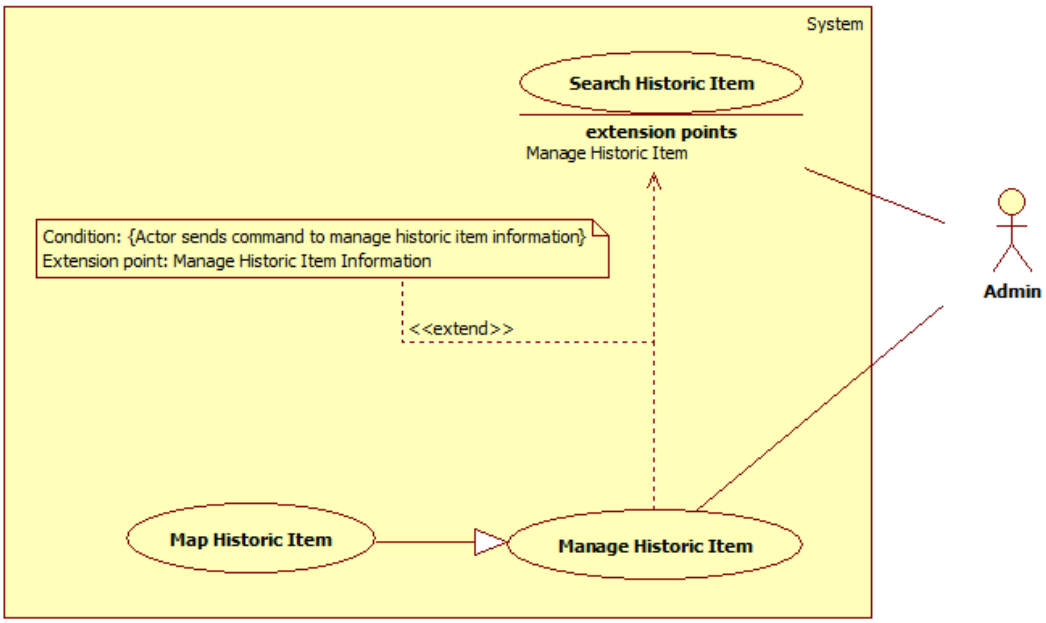


Figure 20. <Admin> Map Historic Item

USE CASE – UC_A07			
Use Case No.	UC_A07	**Use Case Version**	1.2
Use Case Name	Map Historic Item		
Author	HungPQ		
Date	24/05/2016	**Priority**	High
Actor: - Admin **Summary:** - This use case allows admin to map historic item information to Beacon. - This use case applies for Admin web application.			

Goal:

- Admin can map historic item information to a Beacon.

Triggers:

- Admin sends “Map historic Item” command.

Preconditions:

- Actor logged in under Admin role.

Post Conditions:

- **Success:** New item is successfully saved.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Admin accesses “Manage historic item” view.	System requires information of historic items: - *Item name*: label. - *Item English name*: label. - *Item floor*: drop-down list (show existed floor), required. - *Item room*: drop-down list (show existed room) required. - *Item area*: drop-down list (show existed area), required. - *Beacon MAC Address*: drop-down list (show available Beacons), required.
2	Admin enters information of items.	System shows the input information in suitable text input.
3		

	Admin sends “Map Historic Item” command. [Alternative 1]	System saves historic item information. System redirects to “Manage Historic Item” view. [Exception 1]

Alternative Scenario 1:

No	Actor Action	System Response
1	Admin sends cancel command.	System redirects to previous view and does nothing.

Exceptions 1:

No.	Cause	System Response
1	Admin ignores one of required field.	System shows messages “Please input all required fields”

Relationships: Extend from “Search Historic Items” Use case

- Condition: Admin sends command to Map Historic Item.
- Extension point: Manage Historic Item Information.

Business Rules:

- System will ensure that one historic item can only map with one beacon.
- If Admin maps Item to a Beacon successfully, that Beacon will change its status to “unavailable”. If the Beacon’s status is “unavailable”, it will not be allowed to choose anymore.
- If Admin selects a Beacon, location is required. However, if Admin chooses location only, Beacon is not required.

- "Item Location": is a model contains Floor, Room and Area.
  ○ A floor may contain rooms, and a room may contain areas. However, there are some areas which don't belong to any room.
  ○ System will ensure that when Admin choose a valid location.
  ○ If Actor chooses an existed Location, Item will take the its ID. If Actor chooses a new combination of Floor-Room-Area, a new Location will be created.
  ○ If Admin chooses a location and insert successfully, system will automatically re-define the Near relationship between items.

Table 27. Map Historic Item specification

2.3.3.6 <Admin> Assign Report to Expert

Figure 21. <Admin> Assign Report to Expert

USE CASE – UC_A10			
Use Case No.	UC_A10	**Use Case Version**	1.0
Use Case Name	Assign Report to Expert		
Author	HungPQ		
Date	08/08/2016	**Priority**	Normal
Actor: - Admin **Summary:** - This use case allows Admin to assign a Visitor's report to an Expert. - This use case applies for Web application. **Goal:** - Actor can assign a Visitor's report to an Expert.			

Triggers:

- Actor sends “Assign Report to Expert” command.

Preconditions:

- Actor must login at Admin role.

Post Conditions:

- **Success:** The Visitors Feedbacks is successfully shown to Actor.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Admin chooses a report to assign [Alternative 1]	System will get and show the list of all Expert One record in list include: - Expert name: drop-down list
2	Admin chooses an Expert, and send “Assign Report to Expert command”. [Alternative 2]	- Report status will be changed from “Unseen” to “Seen” - Report is assigned to Expert. - System redirects to “Visitor Reports” view.

Alternative Scenario 1:

No	Actor Action	System Response
1	There is no Visitor feedback	System shows messages “There is no report from visitors!”

Alternative Scenario 2:

No	Actor Action	System Response
1	There is no Expert in system.	System will not allow Admin to sends “Assign Report to Expert” command.

Exceptions: N/A

Relationships: N/A

Business Rules:

- Reports: While exploring the museum, Visitors can send feedbacks about a specific historic items. Reports could be about: Incorrect historic information, Application shows a different historic items, etc.
- System will ensure Admin must choose one and only Expert to assign report to.
- Notifications:
    - If there is a new feedback from visitors, system will notify Admin immediately.
    - When Admin assign a report to Expert, system will also notify Expert immediately.

2.3.4 <Expert> Overview use case

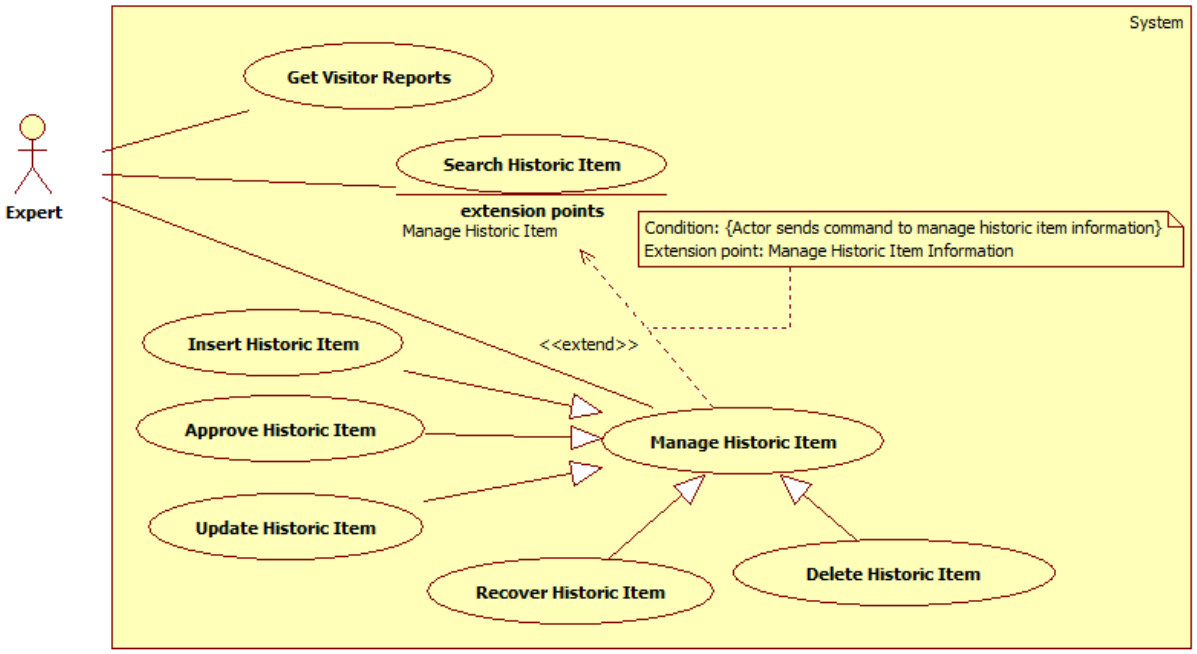


Figure 22. <Expert> Overview Use Case

2.3.4.1 <Expert> Insert Historic Item

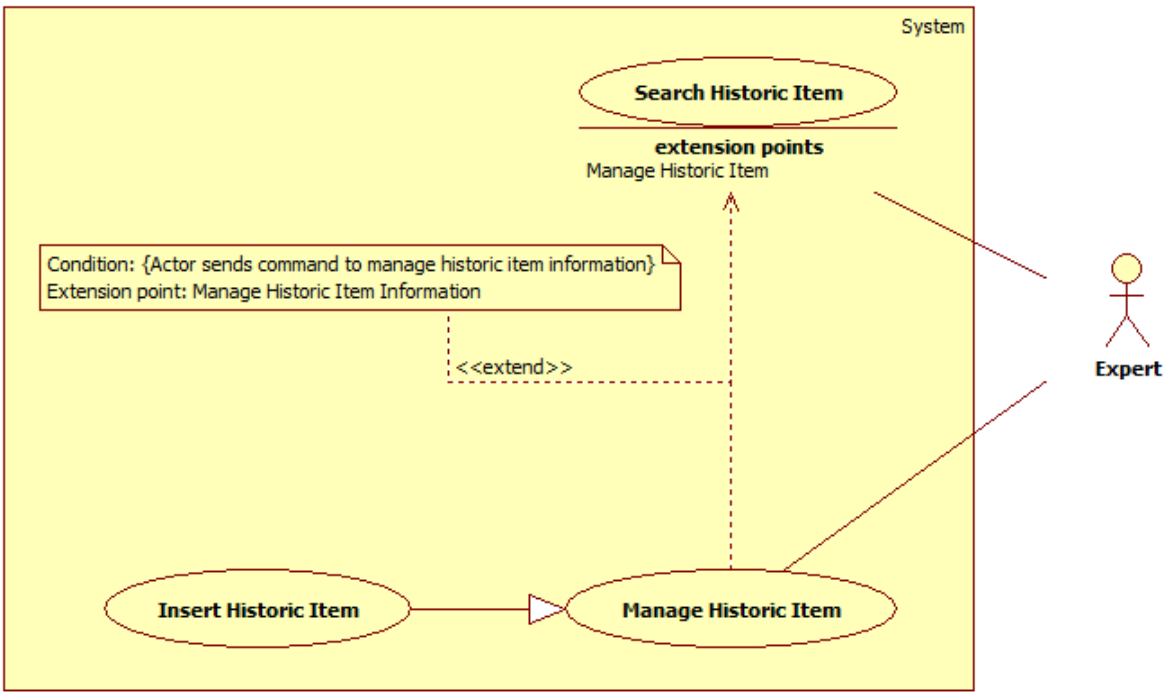


Figure 23. <Admin> Insert Historic Item

USE CASE – UC_E01			
Use Case No.	UC_E01	**Use Case Version**	2.0
Use Case Name	Insert Historic Item		
Author	HungPQ		
Date	24/07/2016	**Priority**	Medium

Actor:

- Expert

Summary:

- This use case allows expert to update a chosen historic item information.
- This use case applies for web application.

Goal:

- Expert can edit and save new information of chosen historic item.

Triggers:

- Expert sends “Insert Historic Item” command.

Preconditions:

- Actor logged in under Expert role.

Post Conditions:

- **Success:**
  - Item information is saved.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Expert accesses “Historic Item Details” view	System requires information of historic items: - *Item image URL*: input text, required, length [12, 2083]. - *Item name*: input text, required, length [1, 200]. - *Item description*: text area, required, length [1, 1000]. - *Item English name*: input text, length [1, 200].

		- *Item English description*: text area, length [1, 1000]. - *Item's Related Items*: check-boxes.
2	Expert enters information of items.	System shows the input information in suitable text input.
3	Expert sends "Update Historic Item" command. [Alternative 1]	System updates item information. System redirects to "Historic Item List" page. [Exception 1] [Exception 2]

Alternative Scenario 1:

No	Actor Action	System Response
1	Expert cancels "Insert Historic Item" command	System redirects to "Historic Item List" view.

Exceptions 1:

No.	Cause	System Response
1	Expert ignores one of required field.	System shows messages "Please input all required information"

Exceptions 2:

No.	Cause	System Response
1	The name of the item is duplicated.	System shows messages “This item is already existed!”

Relationships: Extend from “Search Historic Items” Use case

- Condition: Expert sends command to Insert Historic Item.
- Extension point: Manage Historic Item Information.

Business Rules:

- The name of the historic item should be unique.
- Image type: should be JPG or PNG.
- System will ensure that one historic item can only map with one beacon.
- If Admin maps current historic item information with new beacon, new beacon will change its status to “unavailable”. The previous beacon will change its status to “available”.
- Related Items:
  - When set the related item, the Expert will be suggested the similar items.
  - Auto-handler will calculate the similarity between items based on their Vietnamese name and description, using TF-IDF and Cosine Similarity algorithm.

Table 28. Insert Historic Item specification

2.3.4.2 <Expert> Approve Historic Item

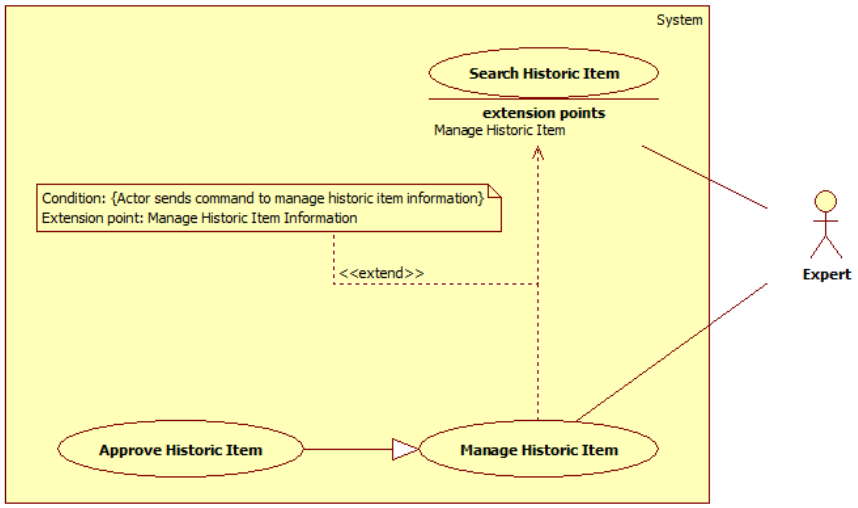


Figure 24. <Admin> Approve Historic Item

USE CASE – UC_E02			
Use Case No.	UC_E02	**Use Case Version**	1.0
Use Case Name	Approve Historic Item		
Author	HungPQ		
Date	24/07/2016	**Priority**	High

Actor:

- Expert

Summary:

- This use case allows expert to approve a pending historic item information.
- This use case applies for web application.

Goal:

- Expert can approve a historic item after it was crawled or updated.

Triggers:

- Admin sends “Approve Historic Item” command.

Preconditions:

- Actor logged in under Expert role.
- Chosen pending historic item must exist.

Post Conditions:

- **Success:**
  - Item information status is updated.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Expert accesses “Pending Historic Item Details” view	System shows information of pending historic items: - *Item image*: images. - *Item new name*: label. - *Item old name*: label. - *Item new description*: text-area. - *Item old description*: text-area.

		- *Item old English name*: label - *Item new English name*: label - *Item old English description*: text-area. - *Item new English description*: text-area.
2	Expert sends "Approve Pending Historic Item" command. [Alternative 1]	System updates item information. System redirects to "Pending Historic Item List" view.

Alternative Scenario 1:

No	Actor Action	System Response
1	Expert cancels "Approve Pending Historic Item" command	System redirects to "Pending Historic Item List" view.

Exceptions: N/A

Relationships: Extend from "Search Historic Items" Use case

- Condition: Expert sends command to Approve Historic Item.
- Extension point: Manage Historic Item Information.

Business Rules:

- Pending Historic Item: There're 2 types of pending item**:**
  - New pending item: Item haven't existed before.
  - Update pending item: Item have existed, but its information is changed.
- If Expert approves a "New" pending item: its status will be changed from "pending" to "active".
- If Expert approves an "Update" pending item: the existed "active" item information will be updated, and the chosen "pending" item will be deleted.

Table 29. Approve Historic Item specification

2.3.4.3 <Expert> Update Historic Item

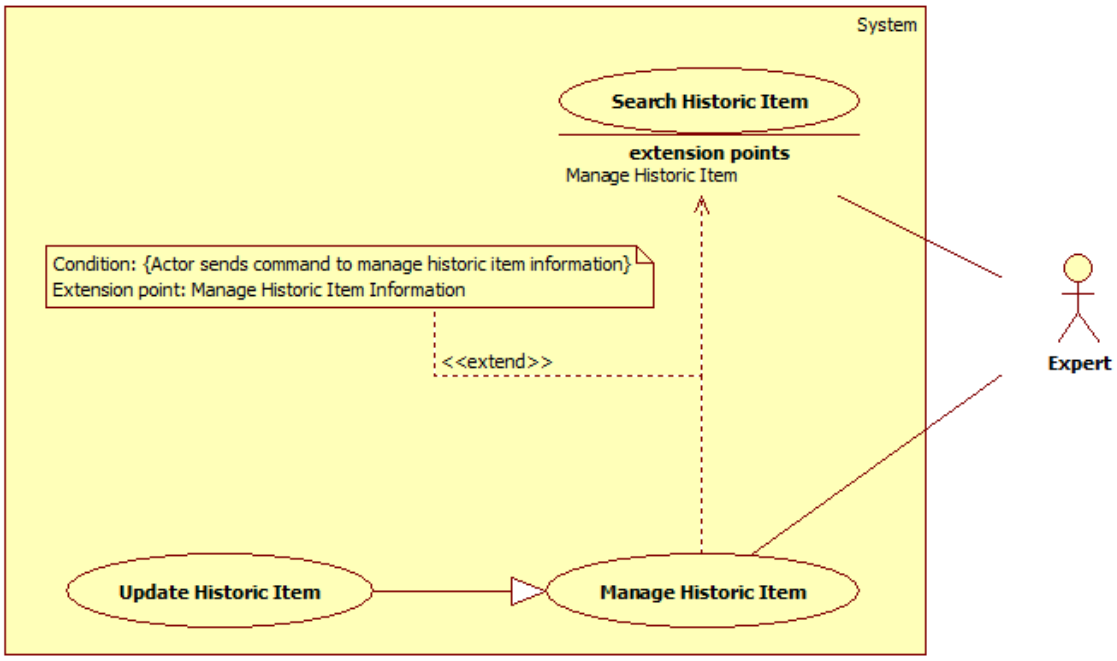


Figure 25. <Admin> Update Historic Item

USE CASE – UC_E03			
Use Case No.	UC_E03	**Use Case Version**	2.0
Use Case Name	Update Historic Item		
Author	HungPQ		
Date	24/07/2016	**Priority**	High

Actor:

- Expert

Summary:

- This use case allows expert to update a chosen historic item information.
- This use case applies for web application.

Goal:

- Expert can edit and save new information of chosen historic item.

Triggers:

- Expert sends “Update Historic Item” command.

Preconditions:

- Actor logged in under Expert role.
- Chosen historic item must exist.

Post Conditions:

- **Success:**
  - Item information is updated.

- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Expert accesses "Historic Item Details" view	System requires information of historic items: - *Item image URL*: input text, required, length [12, 2083]. - *Item name*: input text, required, length [1, 200]. - *Item description*: text area, required, length [1, 1000]. - *Item English name*: input text, length [1, 200]. - *Item English description*: text area, length [1, 1000]. - *Item's Related Items*: check-boxes.
2	Expert changes information of items.	System show the input information in suitable text input.
3	Expert sends "Update Historic Item" command. [Alternative 1]	System updates item information. System redirects to "Historic Item List" page. [Exception 1] [Exception 2]

Alternative Scenario 1:

No	Actor Action	System Response
1	Expert cancels “Update Historic Item” command	System redirects to “Historic Item List” view.

Exceptions 1:

No.	Cause	System Response
1	Expert ignores one of required field.	System shows messages “Please input all required information”

Exceptions 2:

No.	Cause	System Response
1	The name of the item is duplicated.	System shows messages “This item is already existed!”

Relationships: Extend from “Search Historic Items” Use case

- Condition: Expert sends command to Update Historic Item.
- Extension point: Manage Historic Item Information.

Business Rules:

- The name of the historic item should be unique.
- Image type: should be JPG or PNG.
- System will ensure that one historic item can only map with one beacon.
- If Admin maps current historic item information with new beacon, new beacon will change its status to “unavailable”. The previous beacon will change its status to “available”.
- Related Items:
    - When set the related item, the Expert will be suggested the similar items.

○ Auto-handler will calculate the similarity between items based on their Vietnamese name and description, using TF-IDF and Cosine Similarity algorithm.

Table 30. Update Historic Item specification

2.3.5 <Auto Handler> Overview use case

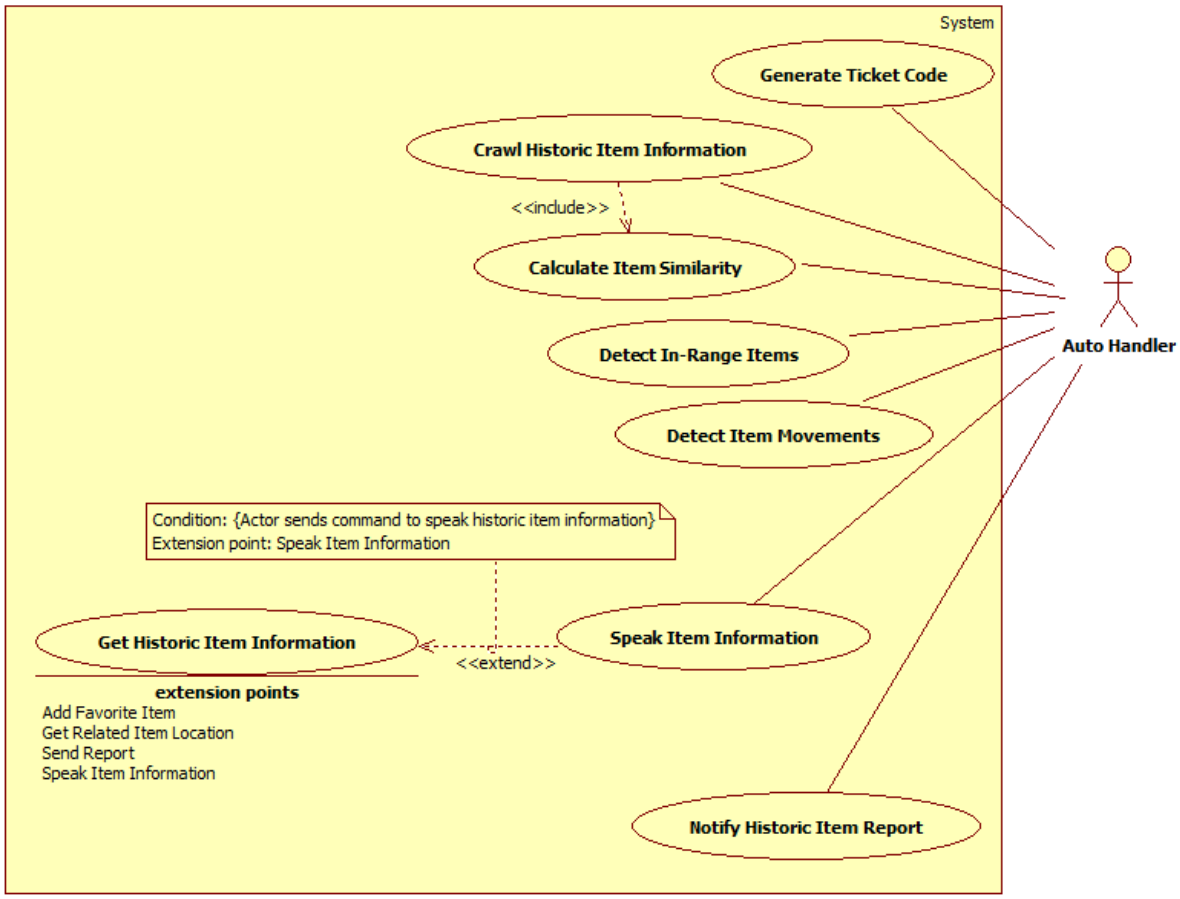


Figure 26. <Auto Handler> Overview Use Case

2.3.5.1 <Auto Handler> Calculate Item Similarity

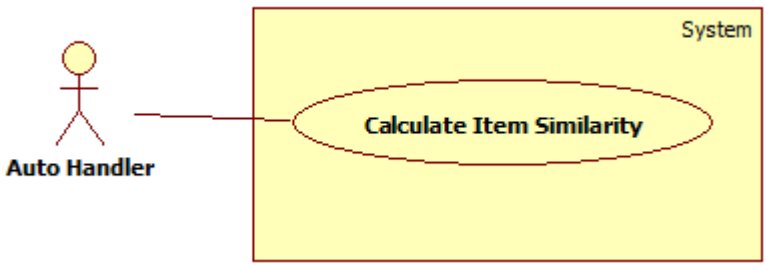


Figure 27. <Auto Handler> Calculate Item Similarity

USE CASE – UC_AH01			
Use Case No.	UC_AH01	**Use Case Version**	2.0
Use Case Name	Calculate Item Similarity		
Author	HungPQ		
Date	23/07/2016	**Priority**	Normal

Actor:

- Auto Handler

Summary:

- This use case allows auto handler to calculate the similarity rate between historic items.

Goal:

- Auto handler calculates the similarity between historic items.

Triggers:

- System hits the configured time or system finish crawling data.
- There are changes in historic items data since the last time it ran.

Preconditions:

- In storage, there must be historic items.
- All existed historic item must have full required information including name, and descriptions.

Post Conditions:

- **Success:**
  - Related relationship of items is updated.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Auto handler check the current time. If it hits the configure time, Auto handler sends “calculate items similarity” command. [Alternative 1]	System calculate items similarity base of their name and description. System updates new related item relationships. [Exception 1]

Alternative Scenario 1:

Step	Actor Action	System Response

1	When Auto handler finishes crawling data, Auto handler sends “calculate items similarity” command.	System calculate items similarity base of their name and description. System updates new related item relationships.

Alternative Scenario 2:

Step	Actor Action	System Response
1	There is nothing change in historic items data since the last time it ran.	System does nothing.

Exceptions 1:

No.	Cause	System Response
1	There is no item.	System does nothing.

Relationships: Included in “Crawl Data” use case.

Business Rules:

- “Calculate item similarity” process has 2 statues, which are “activated” and “not activate”. The status will be switch to “activated” when hit the configured time, and reset to “not activate” after the process finishes.
- Hits configured time: Auto handler can find configured time in the configuration file.
    - Current time hits the configured time.
    - Current time have passed the configured time, but “Calculate items similarity” process status is activated.
- Format of configuration file should be: * * * * * * * in order:
    1. Seconds
    2. Minutes
    3. Hours
    4. Day-of-Month
    5. Month
    6. Day-of-Week
    7. Year (optional field)
    - ***Example**: “0 0 12 ? * WED”* means “every Wednesday at 12:00:00 pm”
- System calculate historic items similarity base on their Vietnamese name and description, using TF-IDF and Cosine Similarity algorithm.

- The calculated results will be suggested to Expert when he set related item. Only related items set by the Expert will be shown to Visitor.

Table 31. Define Related Item specification

2.3.5.2 <Auto Handler> Notify Historic Item Report

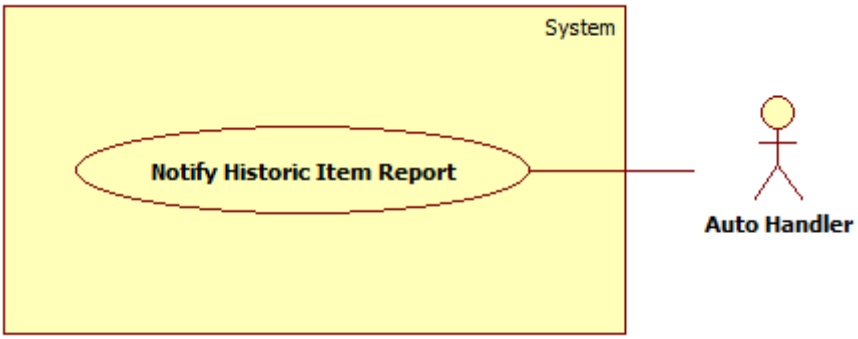


Figure 28. <Auto Handler> Notify Historic Item Report

USE CASE – UC_AH06			
Use Case No.	**UC_AH06**	**Use Case Version**	1.0
Use Case Name	Notify Historic Item Report		
Author	TungTQ		
Date	28/05/2016	**Priority**	Medium

Actor:
- Auto Handler

Summary:
- This use case allows system to notify historic item report for admin.

Goal:
- Auto handler sends notification about historic item report to admin automatically.

Triggers:
- Auto handler receives “Notify Historic Item Report” command.

Preconditions:
- Visitor sends report about a specific historic item.

Post Conditions:
- **Success:** System sends notification to admin on web application
- **Fail:** N/A

Main Success Scenario:

Step	Actor Action	System Response
1	Auto handler sends “notify historic item report” command.	System sends notification to admin.

Alternative Scenario: N/A

Exceptions: N/A

Relationships: Included by **"**Report Item Information" Use case.

Business Rules:

- Report: while exploring the museum, Visitors can send feedbacks about a specific historic item. Report could be include: incorrect historic information, application shows a different historic items, etc.
- Auto Handler will ensure Visitor's report is sent to Admin.
- System will update the number of new report in real-time.

Table 32. Notify Historic Item Report specification

2.3.5.3 <Auto Handler> Detect Moved Beacon

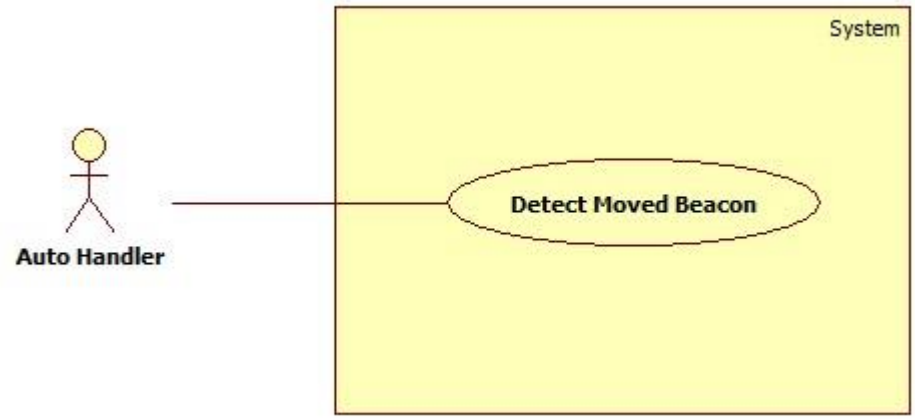


Figure 29. <Auto Handler> Detect Moved Beacon

USE CASE – UC_AH07			
Use Case No.	**UC_AH07**	**Use Case Version**	1.0
Use Case Name	Detect Moved Beacon		
Author	TungTQ		
Date	28/05/2016	**Priority**	High

Actor:

- Auto Handler

Summary:

- This use case allows system to detect moved of beacon.

Goal:

- Auto handler sends notification about movement of beacon to staff automatically.

Triggers:

- Auto handler receives “Detect item move" command.

Preconditions:

- Beacon is connected to Staff mobile device.

Post Conditions:

- **Success:**
  - System detect items movement
  - System sends notification to staff on mobile application
  - System saves log of Beacon movement.
- **Fail:** N/A

Main Success Scenario:

Step	Actor Action	System Response
1	Auto handler receives “items are moved" message.	System saves log of moved beacon. System sends notification to staff.

Alternative Scenario: N/A

Exceptions: N/A

Relationships: N/A

Business Rules:

- System will connect Beacons to mobile application. When done, mobile application will be notified Beacon movements using Estimote Beacon motion sensor through Bluetooth 4.0.
- By default, the status of the log is “Not Checked”. When Staff sends the confirm message, log’s status will be changed from “Not Checked” to “Checked”.

Table 33. Detect Moved Beacon specification

2.3.6 <Staff> Overview use case

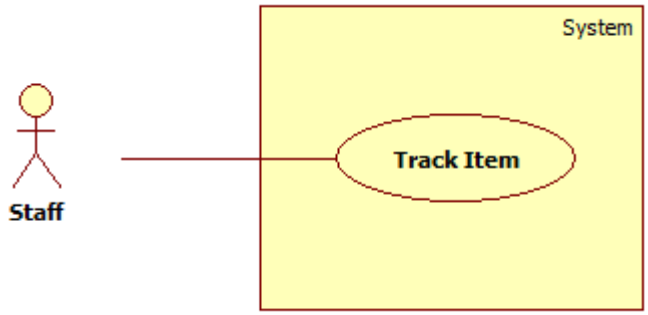


Figure 30. < Staff > Overview Use Case

2.3.6.1 < Staff> Track Item

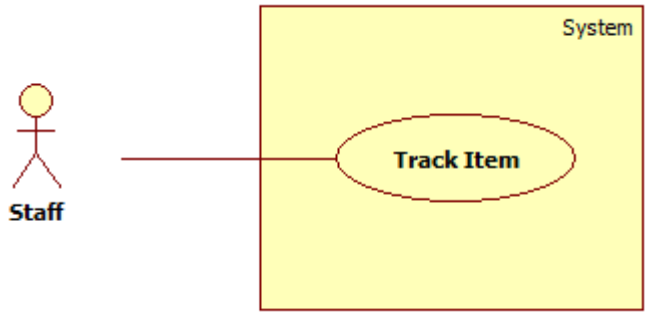


Figure 31. < Staff > Track Item

USE CASE – UC_S01			
Use Case No.	UC_S01	**Use Case Version**	2.0
Use Case Name	Track Item		
Author	HieuVT, PhatNT		
Date	25/05/2016	**Priority**	Normal

Actor:
- Staff.

Summary:
- This use case allows staff to track items wheather they are moved or not.
- This use case applies for Android Mobile Application.

Goal:
- Staff can tracking all historic item in range.
- Staff can see which item is moved
- Staff can check and confirm about movement of item.

Triggers:

- Staff sends “Track Item” command.

Preconditions:

- Actor has accessed the system under Staff role.
- Staff scanned all historic items, which need to be managed.
- Staff received at least one notification about movement of historic item from system.

Post Conditions:

- **Success:**
  - Historic item’s status is updated.
  - “Confirmation is successful” message will be shown.
- **Fail:** System shows error message depend on each specific exception case.

Main Success Scenario:

Step	Actor Action	System Response
1	Staff receives notification about movement of item	
2	Staff inputs a message and sends confirm command	System updates historic item’s status. System updates message. System updates time of confirmation. “Confirmation is successful” message will be shown.

Alternative Scenario: N/A

Exceptions: N/A

Relationships: extend scan item use case.

- Condition: Staff sends command to Track Historic Item.
- Extension point: Tracking Historic Item

Business Rules:

- Actor can track maximum 20 historic items.

- There are two type of status: “not checked” and “checked”.
- System will notify for Staff by vibrate their mobile if a historic item is in motion.
- When staff receives a new notification about movement of item, its default status is “not checked”.
- When staff completed track item process:
  o Historic item’s status is changed to “checked” and be updated.
  o Time of confirmation, which gets current time of confirmation, is updated.
  o Message which was inputted by staff will be saved

Table 34. Track Item specification

3. Software system attribute

3.1 Usability

3.1.1 Graphic user interface

- With BOM website, all the texts, labels, alerts and messages will be written in English.
- With android application, all the texts, labels, alerts and messages will be written in English or Vietnamese depend on the visitor.
- GUI of the BOM website is designed base on material design of Google with the horizontal menu contains all of main function.
- GUI of the mobile application is designed base on material design of Google with the navigation bar contains the main functions on the left of the screen.

3.1.2 Usability

- System provides user GUI with instruction step by step.
- Icon with function name aside will be nice and help user to recognize the feature easier.
- The system is easy to use. It will need about 1 hour for training admin to use the BOM website to take manage on the beacons.
- It will take about 30 minutes for training staff to use the application to track historic item in the museum.
- It will take about 30 minutes for user to get familiar with all function of the system for visitors.

3.1.3 Installation

User can follow installation and manual guide for installation. If there are any problems, user cans contacts developer for helping.

3.2 Reliability

- Mobile devices can approximate their distance to Estimote Beacons based on received signal strength through radio wave. However, distance readings are prone to fluctuation because they depend on a number of external factors. On average, the measurement error can be 20-30% of the actual distance.
- The system uses Estimote Beacon with high accuracy sensors (temperature and motion) the delay time for mobile application recognize the Estimote Beacon sensors can be 1-2s.
- Scheduler task run at 00:00 every day with 99% execution rate.

3.3 Availability

- The system can be adapted for a huge number of request.
- Server should have back-up method to make sure data can be restored easily if any problem happens.
- System is divided into modules, if a function is down, it will not influence others.

3.4 Security

- All input data should be validated before saving to database
- All privacy information, such as password, should be encrypted to ensure security.
- Roles permission should be specified clearly and user should be authenticated and authorized when accessing to the system.
- Mobile device will authenticate with Estimote Cloud before connecting with Estimote Beacon for using its motion sensor.

3.5 Maintainability

- System is divided into modules
- When a module of a function is down, it is easy to take it down to fix without impact other functions

3.6 Portability

- Admin can use the web application of the system on any OS that support web browser
- Staff and visitor can use mobile application on any Android smartphone that support Bluetooth 4.0 (BLE – Bluetooth Less Energy) and Android version 4.4 Kit Kat or above.

3.7 Performance

- Requests from web application are responded in less than 10 seconds at 8 Mbps bandwidth speed.
- Mobile application can return result of the information from the beacons through calling API after a rage from 0.3 to 0.5 second at 8 Mbps bandwidth speed.

4. Conceptual diagram

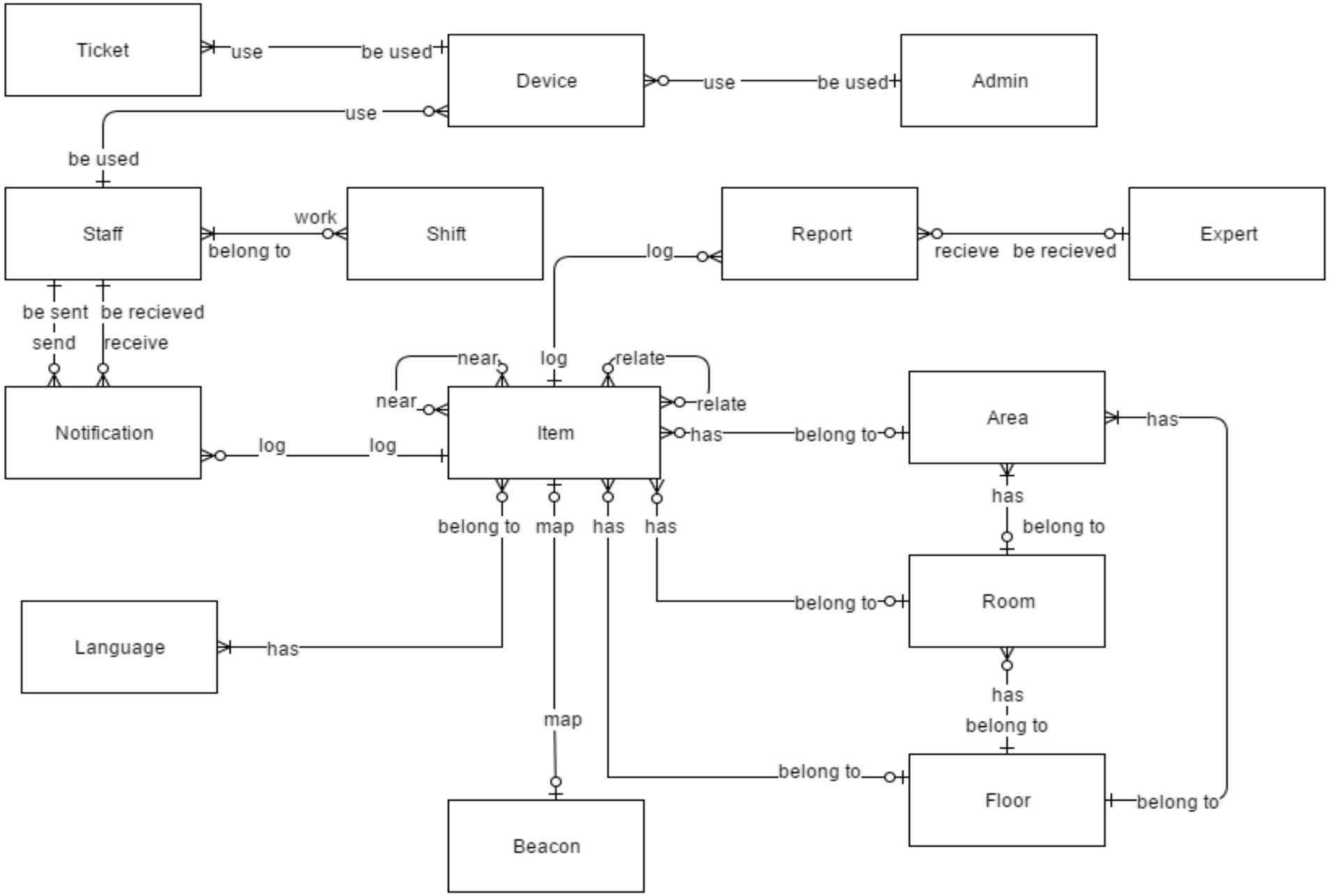


Figure 32. Conceptual diagram

Entity Data dictionary: describe all content of all entities			
Entity Name	**Mapping with Use Case**		**Description**
	Actor	**Behavior**	
Admin	Admin	N/A	Contain the admin information
Staff	Staff	N/A	Contain the staff information
Visitor	Visitor	N/A	Contain the visitor information
Shift	N/A	Track Item	Contain the shift information
Item	N/A	Manage Historic Item	Contain the historic item information
Languages	N/A	Manage Historic Item	Contain the language information
Beacon	N/A	Manage Beacon Information	Contain the beacon information
Notification	N/A	Get Log of Item Movements	Contain the notification information
Report	N/A	Get Visitor Reports, Send	Contain the visitor's report information
Area	N/A	Manage Historic Item	Contain the area information
Room	N/A	Manage Historic Item	Contain the room information
Floor	N/A	Manage Historic Item	Contain the floor information
Ticket	N/A	Generate Ticket Code	Contain the ticket information
Expert	Expert	N/A	Contain the expert information

Table 35. Conceptual diagram data dictionary

D. Software Design Description

1. Design Overview

This document describes the technical and user interface design of **iMuseum 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 screens design.
    - Section 6: describe a fully attributed ERD.
    - Section 7: describe algorithms**.**

2. System Architectural Design

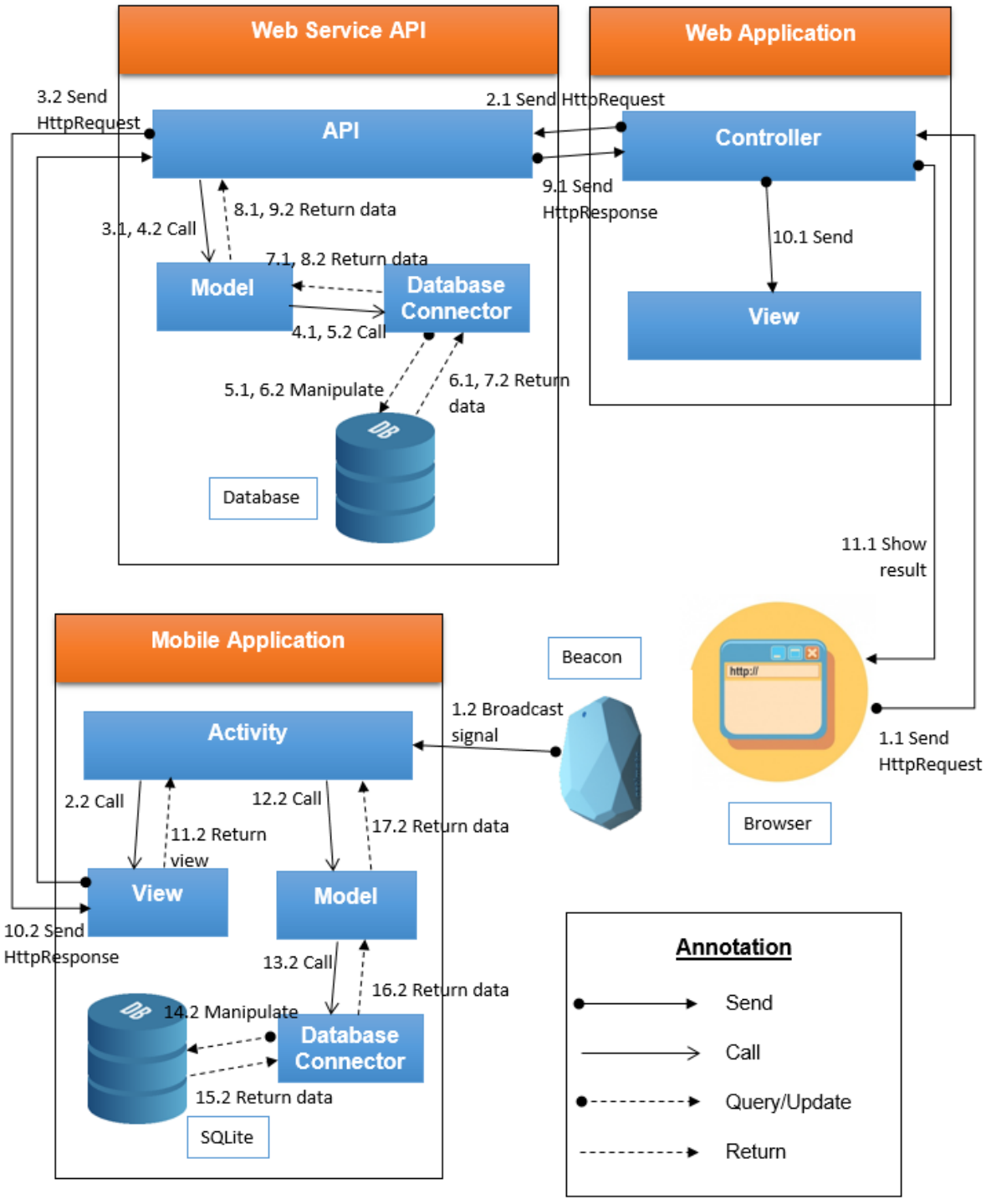


Figure 33. System Architectural Design

2.1 Web Application Architecture Description

We choose Restful Web Service model as main model because of following advantages:

- Our system uses two different platforms: Web and Android. In the future, our application may expand for other platforms as well. Web service provides reusable API that can be called by any platforms, so that our system could easily expand.
- Beacon is a newly developed technology; its SDK is changing frequently. When does, we only need to update mobile application without affecting other components.

Restful Web services provides Application Program Interface to user. Internal structure may be changed, so we choose MVC model to develop Web service because of following advantages:

- We can organize the code better for maintainability, extensibility, reusability so we can expand the scope for more function.
- MVC architecture make it easier to split the big project into small modules and make it easier to assign each module for members in our team.

Number	From	To	Type	Description
1.1	Browser	[Web Application] Controller	Send	Send HttpRequest
2.1	[Web Application] Controller	[Web Service] API	Send	Send HttpRequest
3.1	[Web Service] API	[Web Service] Model	Call	Process data
4.1	[Web Service] Model	[Web Service] Database Connector	Call	Connect database
5.1	[Web Service] Database Connector	Database	Query/Update	Manipulate data
6.1	Database	[Web Service] Database Connector	Return	Return data
7.1	Database	[Web Service] Model	Return	Return data
8.1	[Web Service] Model	[Web Service] API	Return	Return data
9.1	[Web Service] API	[Web Application] Controller	Send	Return data
10.1	[Web Application] Controller	[Web Application] View	Send	Create view
11.1	[Web Application] View	Browser	Send	Show result

Table 36. Executing flow of Web Application

2.2 Mobile Application Architecture Description

The application is developed as an Android native application. In general, the application architecture conforms to Android architecture.

> **Reference**: Android Developer Guider – Application Fundamentals
> http://developer.android.com/guide/components/fundamentals.html

Our team chooses Android because of following reasons:

- Android is currently the most common mobile platforms in the world, and it doesn't have any signs of slowing down. As we want to maximize our potential users, choosing Android will give us the highest popularity market.
- Developing other mobile platforms (such as iOS or Windows Phone) may require specific expensive devices and tools. On the other hand, Android development can be done on either Mac, Windows or Linux devices with free tools.
- Native Android applications are developed using Java programing language. Our team is strong at Java and chooses it as the main programing language.

Number	From	To	Type	Description
1.2	Beacon	[Mobile App] Activity	Send	Broadcast Signal
2.2	[Mobile App] Activity	[Mobile App] View	Call	Create view
3.2	[Mobile App] View	[Web Service] API	Send	Send HttpRequest
4.2	[Web Service] API	[Web Service] Model	Call	Process data
5.2	[Web Service] Model	[Web Service] Database Connector	Call	Manipulate data
6.2	[Web Service] Database Connector	Database	Query/Update	Connect Database
7.2	Database	[Web Service] Database Connector	Return	Return data
8.2	Database	[Web Service] Model	Return	Return data
9.2	[Web Service] Model	[Web Service] API	Return	Return data
10.2	[Web Service] API	[Mobile App] View	Send	Send HttpResponse

11.2	[Mobile App] View	[Mobile App] Activity	Return	Return view
12.2	[Mobile App] Activity	[Mobile App] Model	Call	Process data
13.2	[Mobile App] Model	Database	Query/Update	Manipulate local data
14.2	Database	[Mobile App] Model	Return	Return data
15.2	[Mobile App] Model	[Mobile App] Activity	Return	Return data

Table 37. Executing flow of Mobile Application

3. Component Diagram

3.1 Component Diagram for Server

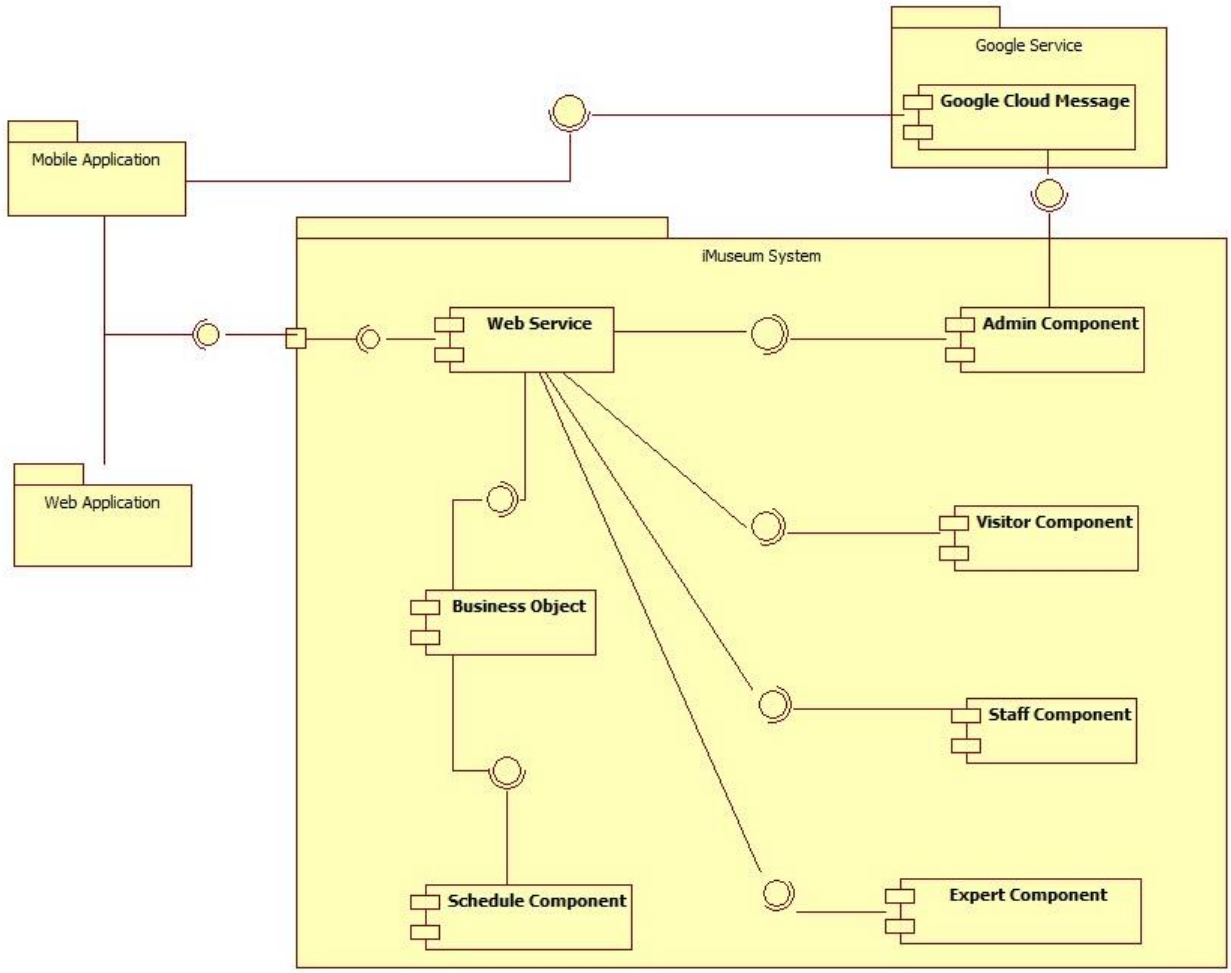


Figure 34. Component Diagram for Server

Component dictionary: describe component	
Component Name	**Description**
Web Services	Provide API for mobile application and web application to interact with the system.
Visitor Component	Component handles all of visitor activities in the system using API of web service such as get item information, report item….
Staff Component	Component handles all of staff activities in the system using API of web service such as track movement of item.
Admin Component	Component handles all of admin activities in the system using API of web service such as manage item, manage beacon, crawl data….
Expert Component	Component handles all of expert activities in the system using API of web service such as approve pending historic item, or edit
Scheduler Component	Component using interface from business object handles scheduler activities in the system.
Business Object	Common using API of web service object to handle domain business operations for each components
Google Cloud Services	It is an open service developed by Google. Our system uses it to handle push notifications. ***Reference:*** https://developers.google.com/cloud-messaging/

Table 38. Component Diagram for Server

3.2 Component Diagram for Mobile Application

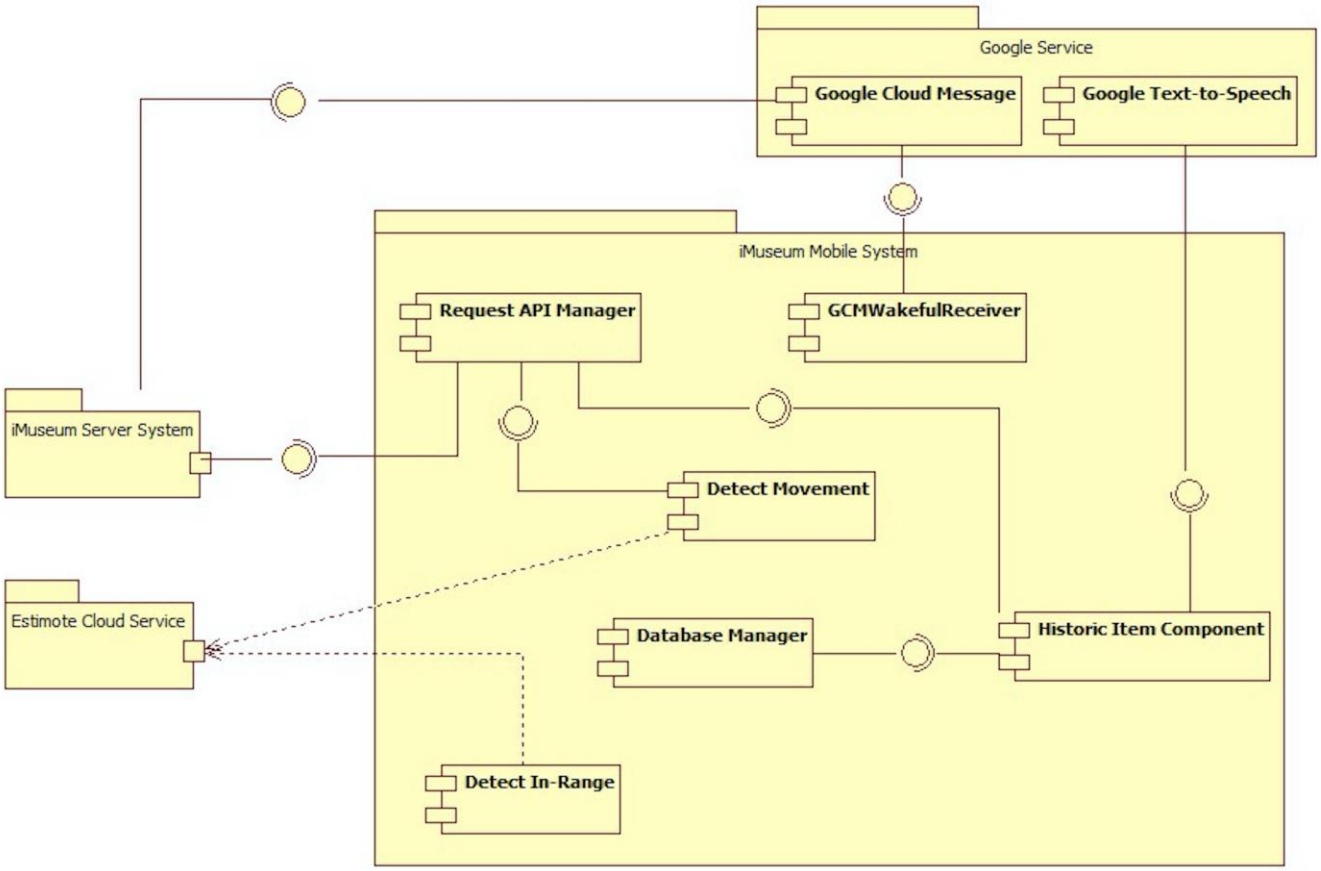


Figure 35. Component Diagrams for Mobile Application

Component dictionary: describe component	
Component Name	**Description**
Historic Item Component	Component handles all of historic items information in the mobile application using local database and API of iMuseum Server System.
Detect Movement	Component detects the movements of Beacons.
Detect In-Range	Component detects whether there is any Beacons around.
GCMWakefulReceiver	Component receives all of notifications from iMuseum Server System using Google Cloud Message service.
Request API Manager	Component manage all API that need to be called and handle all the received data from iMuseum Server System.
Database Manager	Component interacts with mobile local database. This component using for

	(uses for storing) store offline historic item information.
Google Text-to-Speech	Component provides the API to **transform (convert?)** historic items information from text to voice.
Google Cloud Message	It is an open service developed by Google. Our system uses it to handle push notifications. ***Reference:*** https://developers.google.com/cloud-messaging/

Table 39. Component Diagram for Mobile Application

4. Detailed Description

4.1 Class Diagram

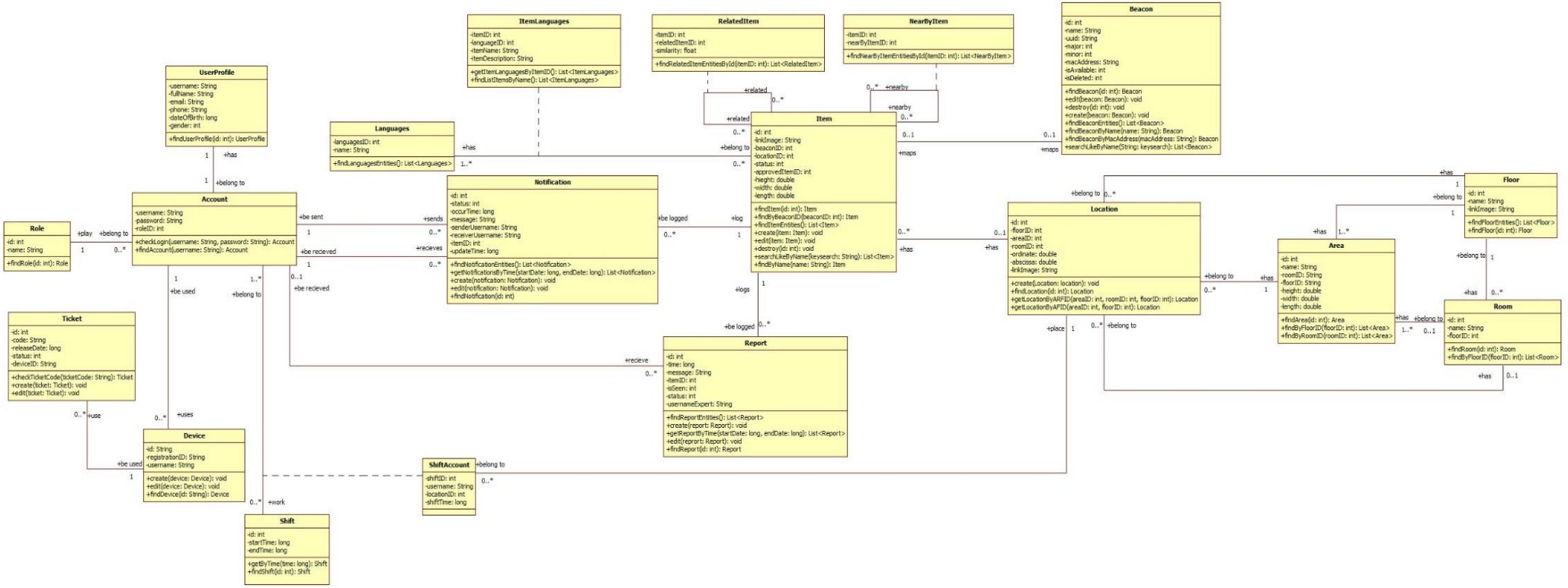


Figure 36. Class Diagram

Class dictionary: Describe Class			
Class Name	**Mapping column with Conceptual diagram**	**Mapping column with Entity diagram**	**Description**
Item	Item	Item	Contain the item information.
ItemLanguages	N/A	N/A	Contain information about the relationship of Item class and Languages class.
Languages	Languages	Languages	Contain the languages information.
Beacon	Beacon	Beacon	Contain the beacon information.
Report	Report	Report	Contain the report information.
Notification	Notification	Notification	Contain the notification information.
Shift	Shift	Shift	Contain the shift information.
Device	Device	Device	Contain the device information.
Area	Area	Area	Contain the area information.
Room	Room	Room	Contain the room information.
Floor	Floor	Floor	Contain the floor information.
Ticket	Ticket	Ticket	Contain the ticket information.
Account	N/A	Account	Contain the account information of museum staff
Role	N/A	Role	Contain the role information. Has 3 types: Admin and Staff.
UserProfile	N/A	UserProfile	Contain the user profile information.
RelatedItem	N/A	N/A	Contain information about the relationship of two Item classes that are relate together.
NearByItem	N/A	N/A	Contain information about the relationship of two Item classes that are nearby.
Location	N/A	Location	Contain the location information, which includes floor, room and area.
ShiftAccount	N/A	N/A	Contain information about the relationship of Account class and Shift class.

Table 40. Class Diagram Dictionary

4.2Class Diagram Explanation

4.2.1 Account

Attributes

Attribute	Type	Visibility	Description
username	String	private	Unique username of an account uses to login into specific system.
password	String	private	Password of an account uses to login into specific system.
roleID	int	private	Identifier of role that museum staff plays.
userProfileID	int	private	Identifier of userProfile that belong to account.

Table 41. Class Diagram Dictionary: Account Attributes

Methods

Method	Return Type	Visibility	Description
checkLogin	Account	public	Check username and password when guest login to system.
findAccount	Account	public	Get account by username.

Table 42. Class Diagram Dictionary: Account Methods

4.2.2 Role

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique identifier of a role.
name	String	private	Name of a role.

Table 43. Class Diagram Dictionary: Role Attributes

Methods

Method	Return Type	Visibility	Description
findRole	Role	public	Get role by id.

Table 44. Class Diagram Dictionary: Role Methods

4.2.3 UserProfile

Attributes

Attribute	Type	Visibility	Description
username	String	private	Unique username of an account that user_profile belong to.
fullName	String	private	Full name of a museum staff.
email	String	private	Email of a museum staff.
phone	String	private	Phone of a museum staff.
dateOfBirth	String	private	Date of Birth of a museum staff.
gender	int	private	Gender of a museum staff.

Table 45. Class Diagram Dictionary: UserProfile Attributes

Methods

Method	Return Type	Visibility	Description
findUserProfile	UserProfile	public	Get user profile by username.

Table 46. Class Diagram Dictionary: UserProfiles Methods

4.2.4 Device

Attributes

Attribute	Type	Visibility	Description
id	String	private	Unique identifier of a device.
registrationID	String	private	Registration ID is an identifier assigned by GCM to a single instance of a single application installed on an Android device.
username	String	private	Username of account that uses this device.

Table 47. Class Diagram Dictionary: Device Attributes

Methods

Method	Return Type	Visibility	Description
create	void	Public	Create a device.
edit	void	public	Update a device.
findDevice	Device	public	Get device by id.

Table 48. Class Diagram Dictionary: Device Methods

4.2.5 Shift

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique identifier of a device.
startTime	int	private	Hours of day that shift starts.
endTime	int	private	Hours of day that shift finishes.

Table 49. Class Diagram Dictionary: Shift Attributes

Methods

Method	Return Type	Visibility	Description
getByTime	Shift	public	Get shift by time that is between startTime and endTime.
findShift	Shift	public	Get shift by id.

Table 50. Class Diagram Dictionary: Shift Methods

4.2.6 ShiftAccount

Attributes

Attribute	Type	Visibility	Description
shiftID	int	private	Identifier of shift that museum staff works.
username	String	private	Unique Username of account that work in shift.
locationID	int	private	Identifier of location that where museum staff works in shift.
shiftTime	long	private	Time that museum staffs work in shift.

Table 51. Class Diagram Dictionary: ShiftAccount Attributes

4.2.7 Notification

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique identifier of a notification.
status	int	private	Status of notification have two statuses NOT_CHECK and CHECK. Default

			status is NOT_CHECK.
occurTime	long	private	Time that notification is created.
message	String	private	Message of notification.
senderUsername	String	private	Username of account that sends notification
receiverUsername	String	private	Username of museum staff that receives notification.
itemID	int	private	Identifier of moved item.
updateTime	int	private	Time that museum staff updates notification.

Table 52. Class Diagram Dictionary: Notification Attributes

Methods

Method	Return Type	Visibility	Description
findNotificationEntities	List<Notification>	public	Get all notifications.
getNotificationByTime	List<Notification>	public	Get list of notifications that have occurTime is between start time and end time.
create	void	public	Create notification.
edit	void	public	Update notification.
findNotification	Notification	public	Get notification by id.

Table 53. Class Diagram Dictionary: Notification Methods

4.2.8 Report

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique identifier of a report.
time	long	private	Time that report is created.
message	String	private	Message of report.
itemID	int	private	Identifier of feedback item.

isSeen	int	private	Status of report have two statuses NOT_SEEN and SEEN. When report is create, it is NOT_SEEN.
usernameExpert	String	private	Username of museum expert that is assigned visitor's reports by admin.

Table 54. Class Diagram Dictionary: Report Attributes

Methods

Method	Return Type	Visibility	Description
findReportEntities	List<Report>	public	Get all reports.
getReportByTime	List<Report>	public	Get list of report that have time is between start time and end time.
create	void	public	Create report.
edit	void	public	Update report.
findNotification	Notification	public	Get report by id.

Table 55. Class Diagram Dictionary: Report Methods

4.2.9 Languages

Attributes

Attribute	Type	Visibility	Description
languagesID	int	private	Unique identifier of a language.
name	String	private	The name of the language.

Table 56. Class Diagram Dictionary: Languages Attributes

Methods

Method	Return Type	Visibility	Description
findLanguagesEntities	List<Item>	public	Get all languages.

Table 57. Class Diagram Dictionary: Languages Methods

4.2.10 Item

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique identifier of an item.

linkImage	String	private	Link Image of item.
beaconID	int	private	Identifier of beacon that maps with this item.
locationID	int	private	Identifier of location that contains this item.
isDelete	int	private	Status of item have two status DELETED and UNDETED.
approvedItemID	Int	private	The identifier of the item which has same name but is existed in database before.
height	float	private	The height of the item
width	float	private	The width of the item
length	float	private	The length of the item

Table 58. Class Diagram Dictionary: Item Attributes

Methods

Method	Return Type	Visibility	Description
findItem	Item	public	Get item by id.
findByBeaconID	Item	public	Get item by identifier of beacon.
findItemEntities	List<Item>	public	Get all items.
create	void	public	Create item.
edit	Report	public	Update item.
destroy	void	public	Remove item by identifier of item.
searchLikeByName	List<Item>	public	Get list of item whose name like key search.
findByName	Item	public	Get item by name.
getNotDeletedItem	List<Item>	public	Get items, that have UNDELETED status
getDeletedItem	List<Item>	public	Get items, that have UNDELETED status

Table 59. Class Diagram Dictionary: Item Methods

4.2.11 ItemLanguages

Attributes

Attribute	Type	Visibility	Description
itemID	int	private	Identifier of an item.
languageID	Int	private	Identifier of a language.
itemName	String	private	The name of the item
itemDescription	String		The description of the item

Table 60. Class Diagram Dictionary: ItemLanguages Attributes

Methods

Method	Return Type	Visibility	Description
getItemLanguagesByItemID	List<ItemLanguages>	public	Get itemLanguages base on Item ID
findListItemByName	List<ItemLanguages>	public	Get itemLanguages base on Item name

Table 61. Class Diagram Dictionary: ItemLanguages Methods

4.2.12 RelatedItem

Attributes

Attribute	Type	Visibility	Description
itemID	int	private	Identifier of an item.
relatedItemID	int	private	Identifier of a related item.
similarity	float	private	Index of similarity of two item.

Table 62. Class Diagram Dictionary: RelatedItem Attributes

Methods

Method	Return Type	Visibility	Description
findRelatedItemEntitiesById	List<RelatedItem>	public	Get list related item by its identifier.

Table 63. Class Diagram Dictionary: RelatedItem Methods

4.2.13 NearByItem

Attributes

Attribute	Type	Visibility	Description
itemID	int	private	Identifier of an item.
nearByID	int	private	Identifier of a nearby item.

Table 64. Class Diagram Dictionary: NearByItem Attributes

Methods

Method	Return Type	Visibility	Description
findRelatedItemEntitiesById	List<NearByItem>	public	Get list nearby item by its identifier.

Table 65. Class Diagram Dictionary: NearByItem Methods

4.2.14 Beacon

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique identifier of a beacon
name	String	private	Name of beacon.
uuid	String	private	Universally unique identifier of beacon
major	int	private	Major of beacon
minor	int	private	Minor of beacon
macAddress	String	private	Mac address of beacon
isAvailable	int	private	Status of beacon is available or unavailable.
isDelete	Int	private	Status of beacon have two status DELETED and UNDETED.

Table 66. Class Diagram Dictionary: Beacon Attributes

Methods

Method	Return Type	Visibility	Description
findBeacon	Beacon	public	Get beacon by id
destroy	void	public	Remove beacon
create	void	public	Create beacon
edit	void	public	Update beacon
findBeaconEntities	List<Beacon>	public	Get all beacons
findBeaconByName	Beacon	public	Get beacon by name
findBeaconByMacAddress	Beacon	public	Get beacon by macaddress
searchLikeByName	List<Beacon>	public	Get list of beacon whose name like key search.
getNotDeletedBeacon			Get beacons, that have UNDELETED status.

getDeletedBeacon			Get beacons, that have DELETED status

Table 67. Class Diagram Dictionary: Beacon Methods

4.2.15 Location

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique identifier of each location
areaID	int	private	Identifier of area that belongs to location
roomID	int	private	Identifier of room that belongs to location
floorID	int	private	Identifier of floor that belongs to location
ordinate	double	private	Ordinate of museum location
abscissa	double	private	Abscissa of museum location
linkImage	String	private	Link Image of museum location

Table 68. Class Diagram Dictionary: Location Attributes

Methods

Method	Return Type	Visibility	Description
create	void	public	Create location
findLocation	Location	public	Get location by id
getLocationByARFID	Location	public	Get location by identifier of area, identifier of room, identifier of floor.
getLocationByAFID	Location	public	Get location by identifier of area, identifier of floor.

Table 69. Class Diagram Dictionary: Location Methods

4.2.16 Area

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique Identifier of each area
name	String	private	Name of area.
roomID	int	private	Identifier of room that this area belongs to

floorID	int	private	Identifier of floor that this area belongs to
height	float	private	The height of the area
width	float	private	The width of the area
length	float	private	The length of the area

Table 70. Class Diagram Dictionary: Area Attributes

Methods

Method	Return Type	Visibility	Description
findArea	Area	public	Get area by id
findByFloorID	List<Area>	public	Get list of area by identifier of floor.
findByRoomID	List<Area>	public	Get location by area by identifier room.

Table 71. Class Diagram Dictionary: Area Methods

4.2.17 Room

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique Identifier of each room.
name	String	private	Name of room.
floorID	int	private	Identifier of floor that this room belongs to.

Table 72. Class Diagram Dictionary: Room Attributes

Methods

Method	Return Type	Visibility	Description
findRoom	Room	public	Get room by identifier.
findByFloorID	List<Area>	public	Get list of room by identifier of floor.

Table 73. Class Diagram Dictionary: Room Methods

4.2.18 Floor

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique Identifier of each room.
name	String	private	Name of room.

linkImage	String	private	Link image of floor.

Table 74. Class Diagram Dictionary: Floor Attributes

Methods

Method	Return Type	Visibility	Description
findFloorEntities	List<Floor>	public	Get all floors.
findFloor	Floor	public	Get floor by identifier.

Table 75. Class Diagram Dictionary: Floor Methods

4.2.19 Ticket

Attributes

Attribute	Type	Visibility	Description
id	int	private	Unique Identifier of a ticket.
code	String	private	Code of ticket.
releaseDate	long	private	Time that ticket is created
status	int	private	Status of ticket. A ticket has two status is USED, NOT_USED.
deviceID	String	private	Identifier of device that is using this ticket.

Table 76. Class Diagram Dictionary: Ticket Attributes

Methods

Method	Return Type	Visibility	Description
checkTicketCode	Ticket	public	Check code of ticket
create	void	public	Create ticket
edit	void	public	Update ticket

Table 77. Class Diagram Dictionary: Ticket Methods

4.3 Interactive Diagram

4.3.1 Web Server

4.3.1.1 <Auto Handler>

4.3.1.1.1 Calculate Item Similarity

Summary: this diagram show process of calculate item similarity

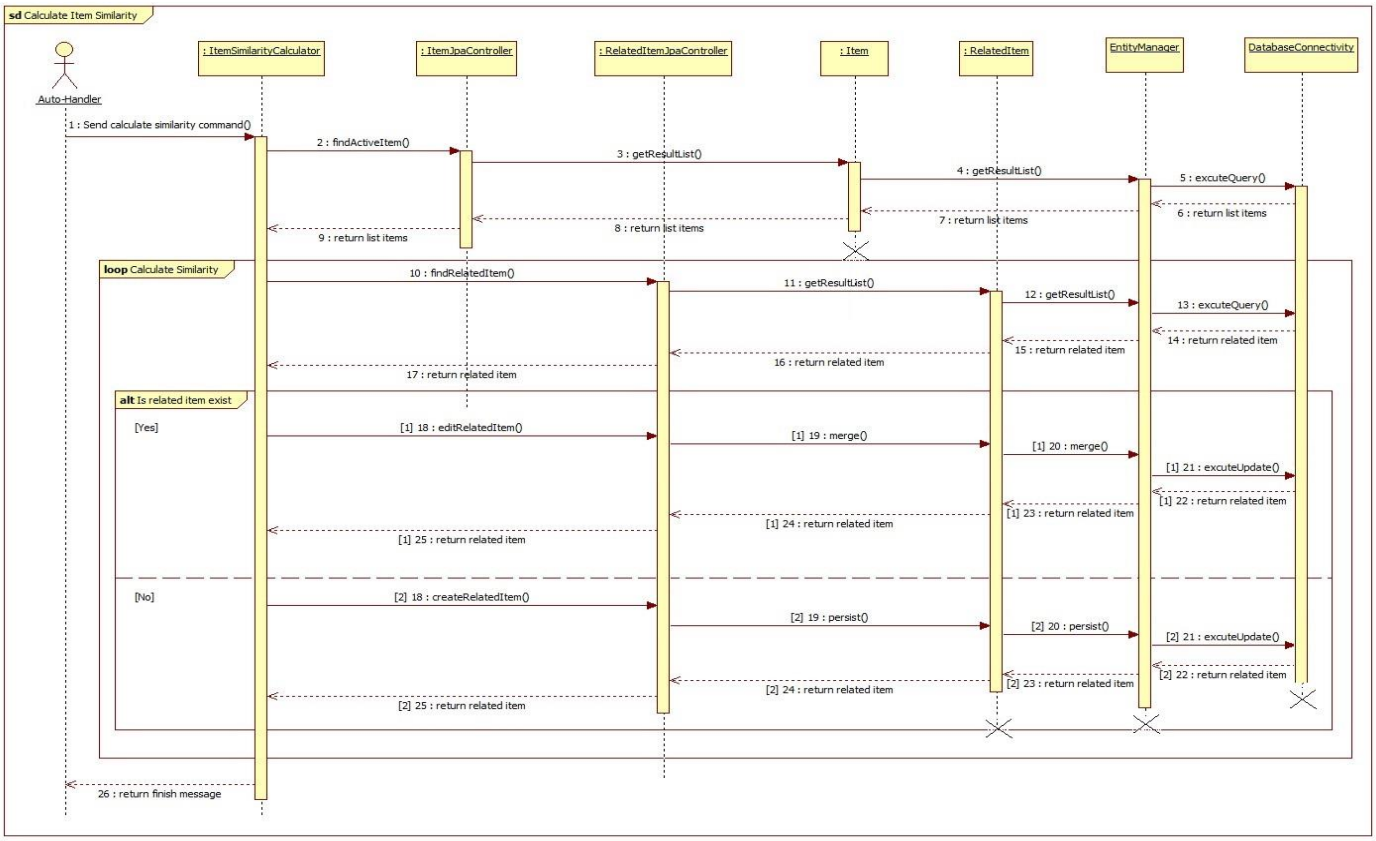


Figure 37. Calculate Item Similarity Sequence Diagram

4.3.2 Web Application

4.3.2.1 <Admin>

4.3.2.1.1 Get Reports

Summary: this diagram show process of get reports

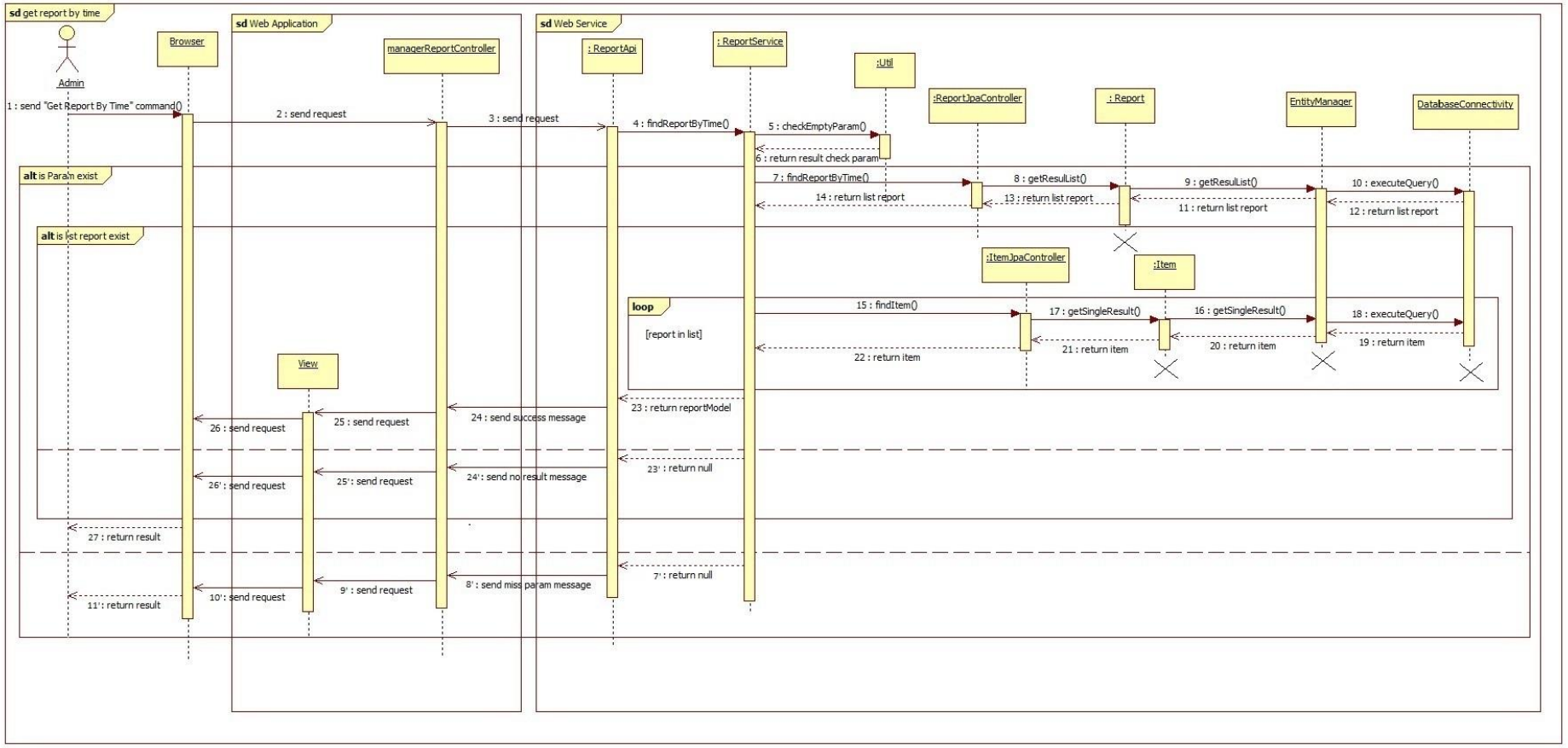


Figure 38. Get Reports Sequence Diagram

4.3.2.1.2 Load Deleted Historic Items

Summary: this diagram show process of load deleted historic items

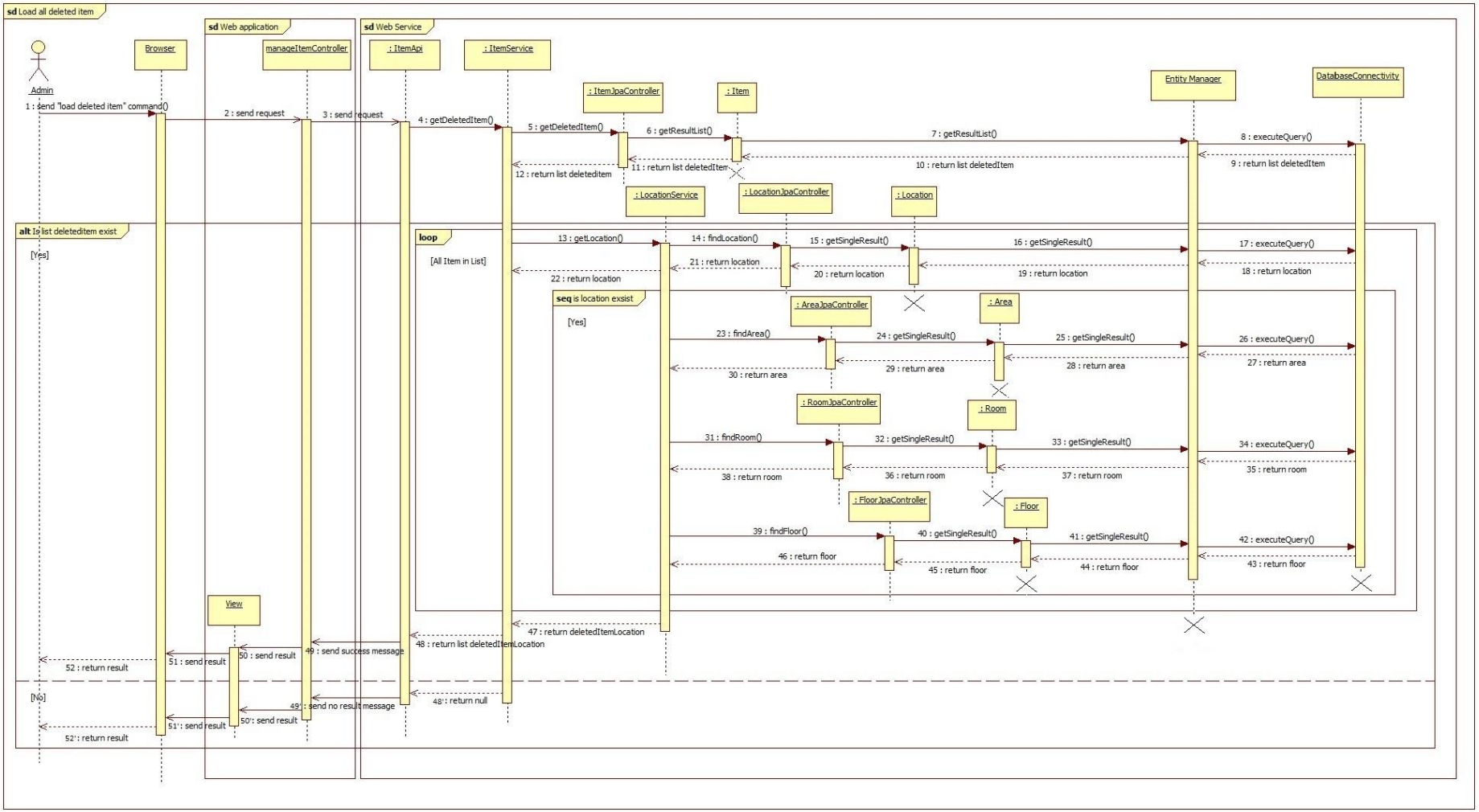


Figure 39. Load Deleted Items Sequence Diagram

4.3.2.1.1 Insert Historic Item

Summary: this diagram show process of insert historic item

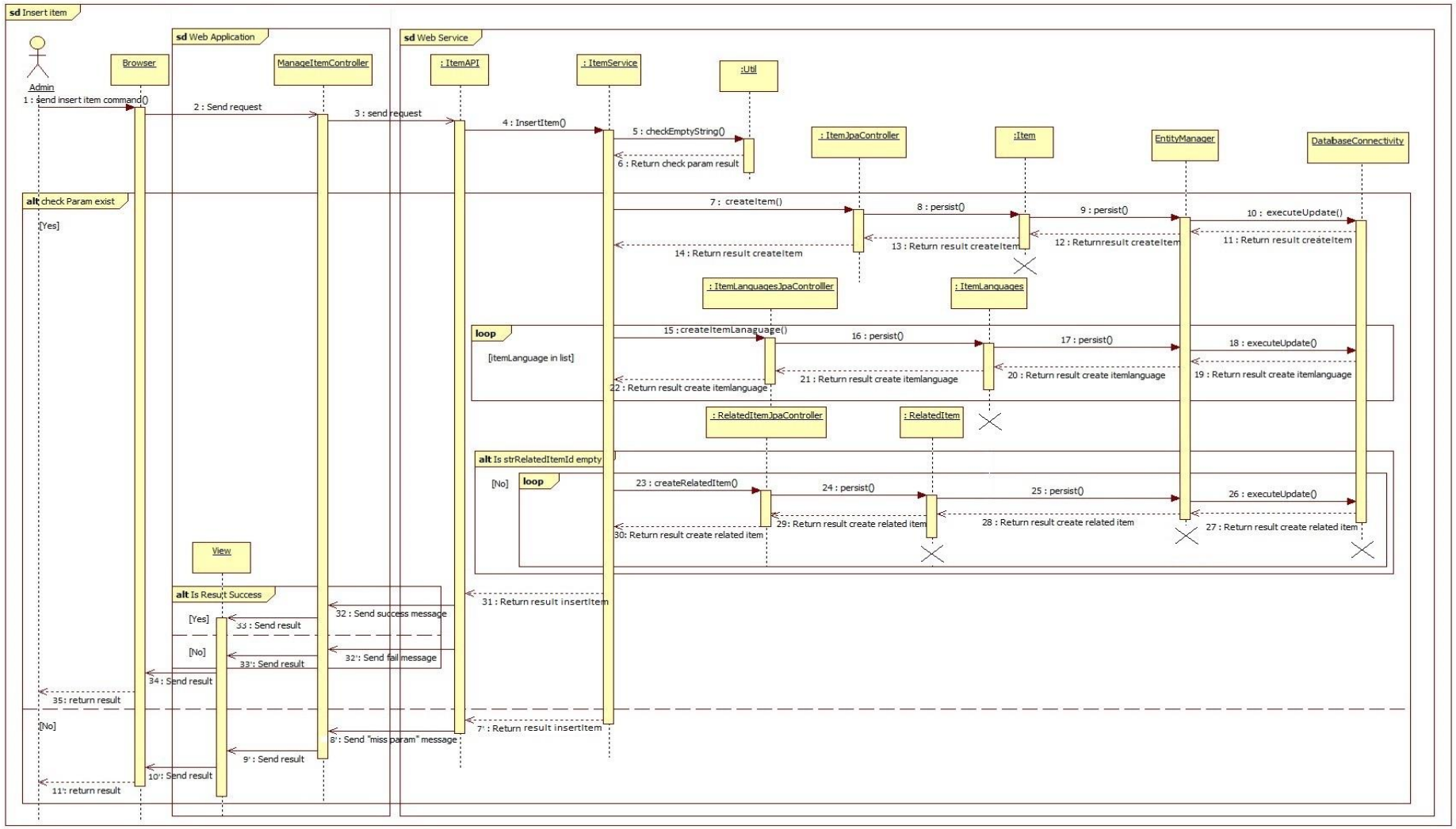


Figure 40. Insert Historic Item Sequence Diagram

4.3.2.1.2 Update Historic Item

Summary: this diagram show process of update historic item

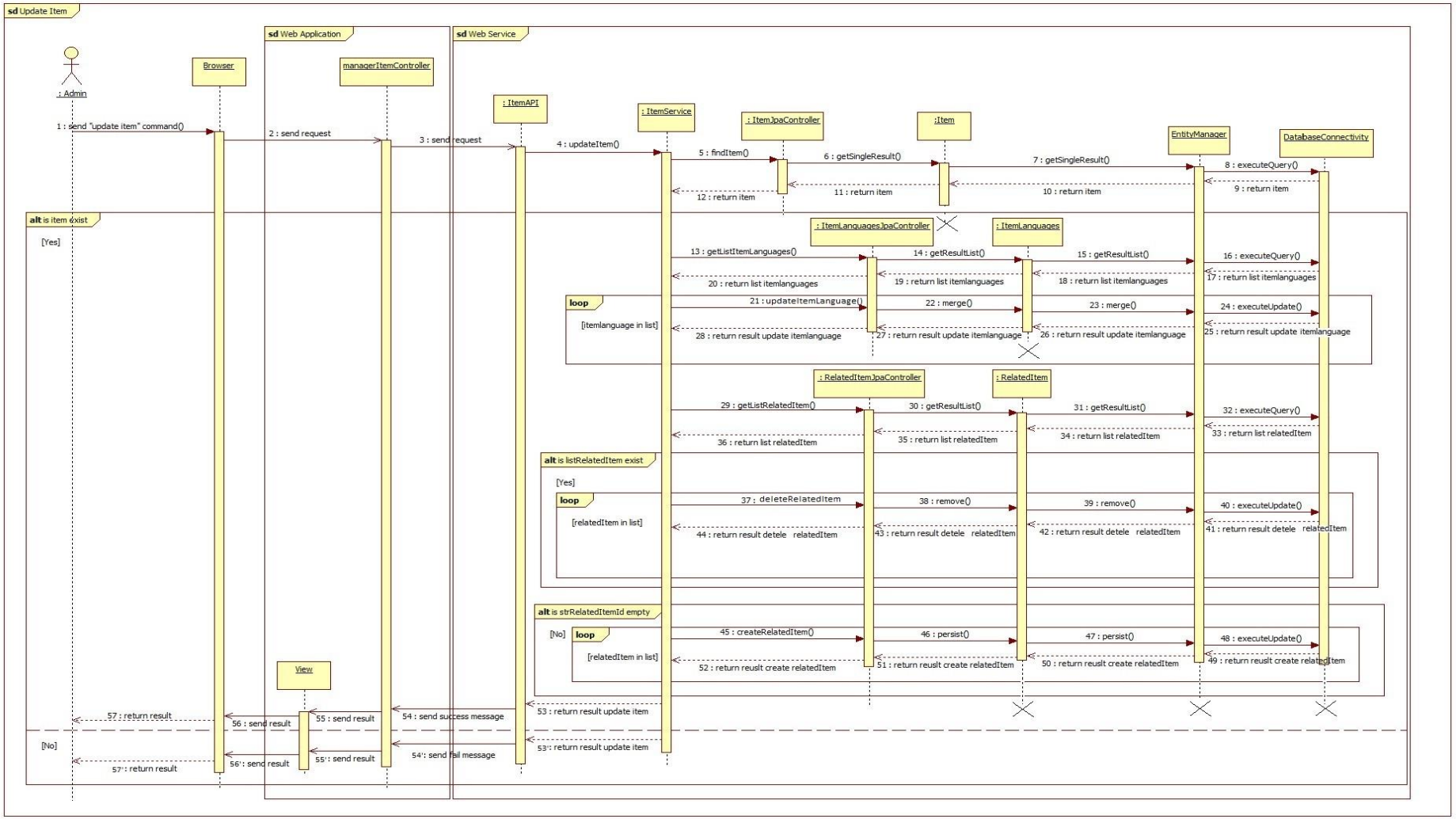


Figure 41. Update Historic Item Sequence Diagram

4.3.2.1.3 Soft Delete Historic Item

Summary: this diagram show process of soft delete historic item

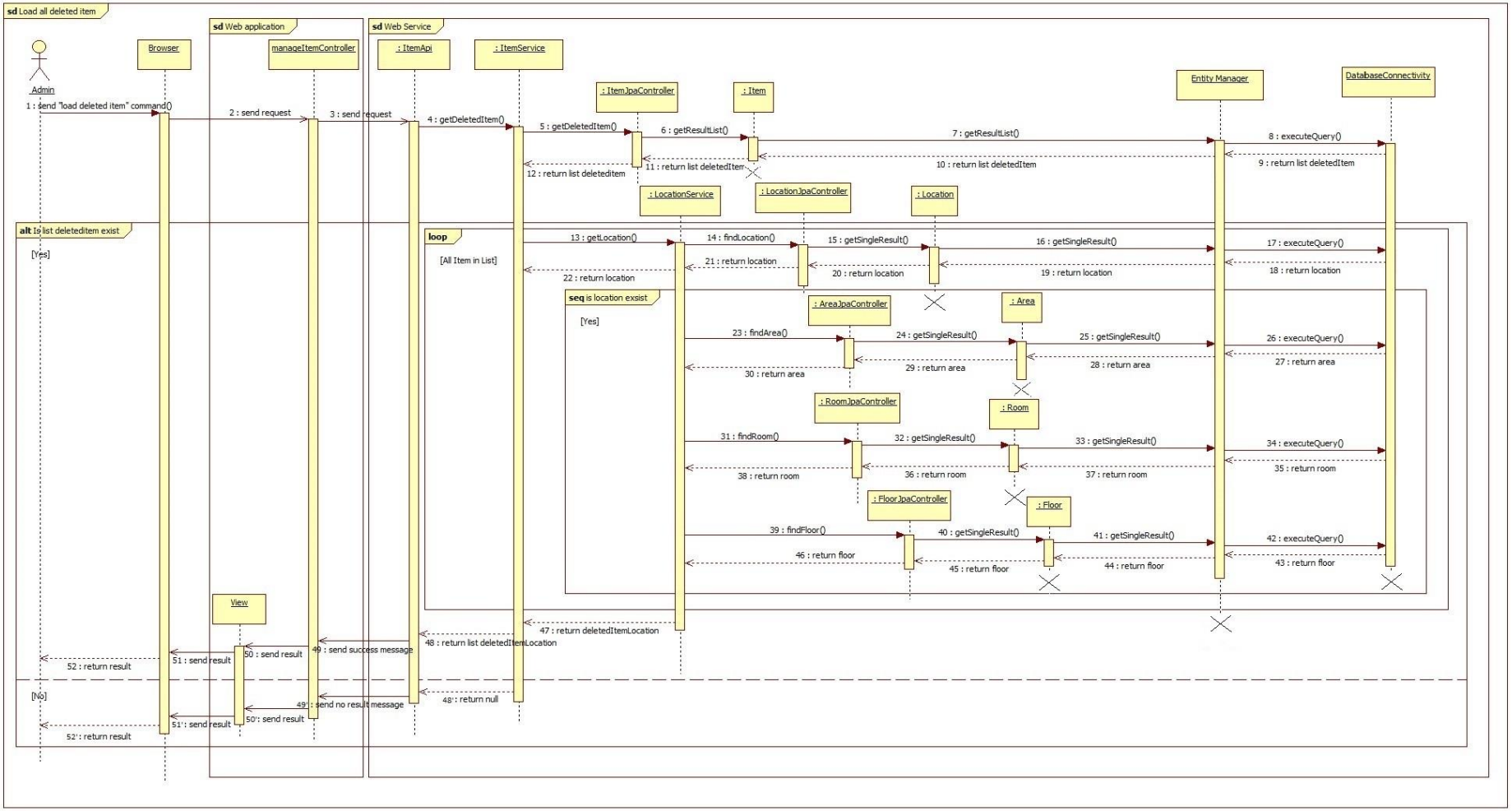


Figure 42. Soft Delete Historic Sequence Diagram

4.3.2.1.4 Load Deleted Beacons

Summary: this diagram show process of load deleted Beacons

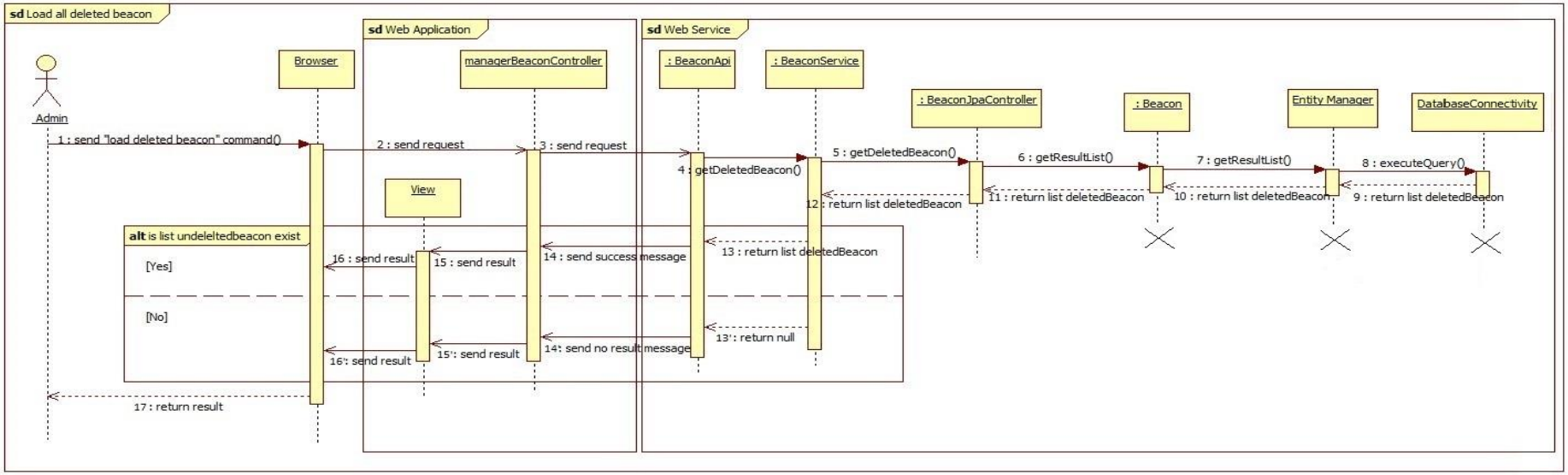


Figure 43. Load Deleted Beacon Sequence Diagram

4.3.2.1.5 Map Beacon

Summary: this diagram show process of map beacon

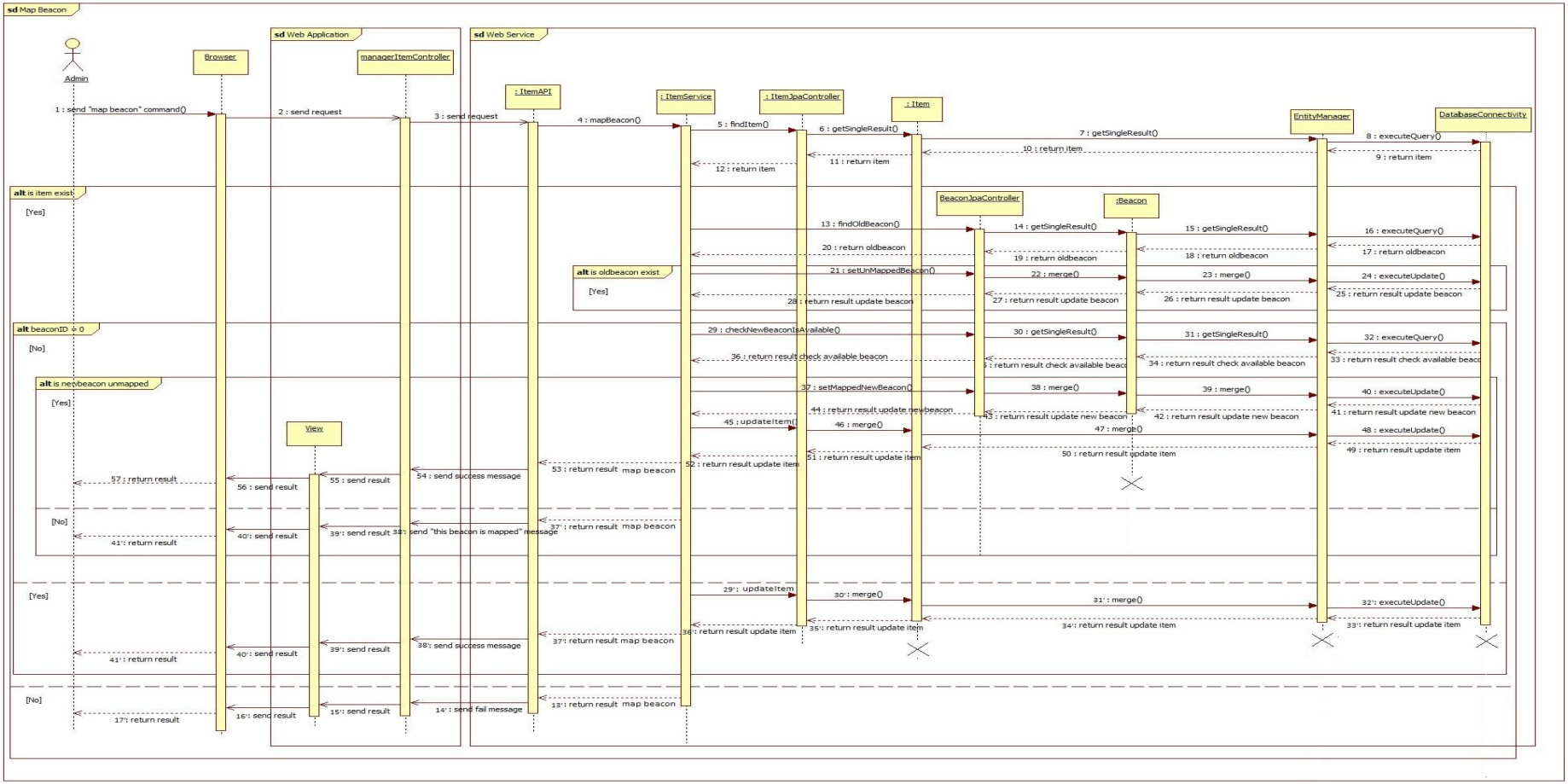


Figure 44. Map Beacon Sequence Diagram

4.3.2.1.6 Soft Delete Beacon

Summary: this diagram show process of soft delete Beacon.

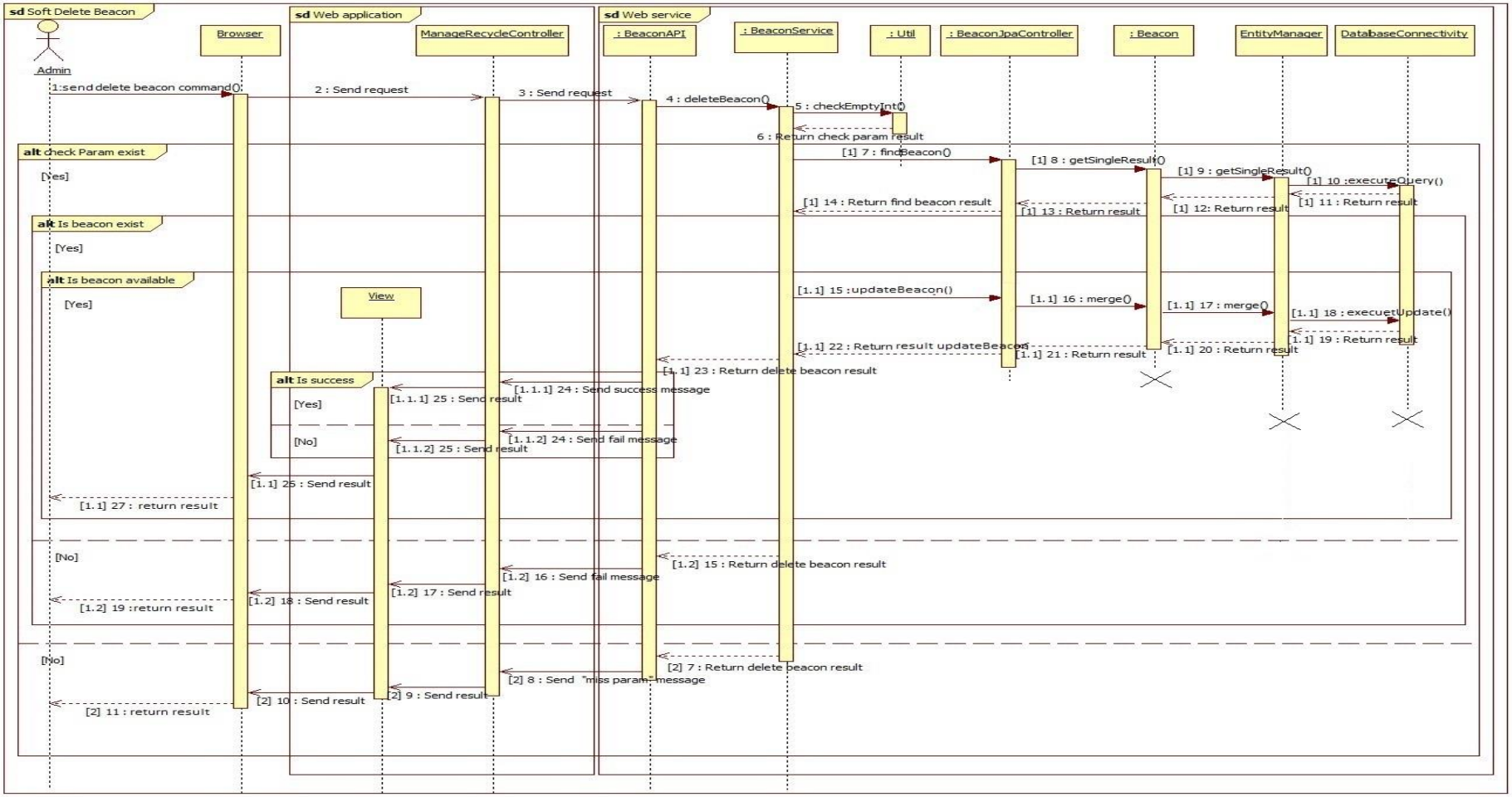


Figure 45. Soft Delete Beacon Sequence Diagram

4.3.2.1.7 Get Item Movements Log

Summary: this diagram show process of get item movements log

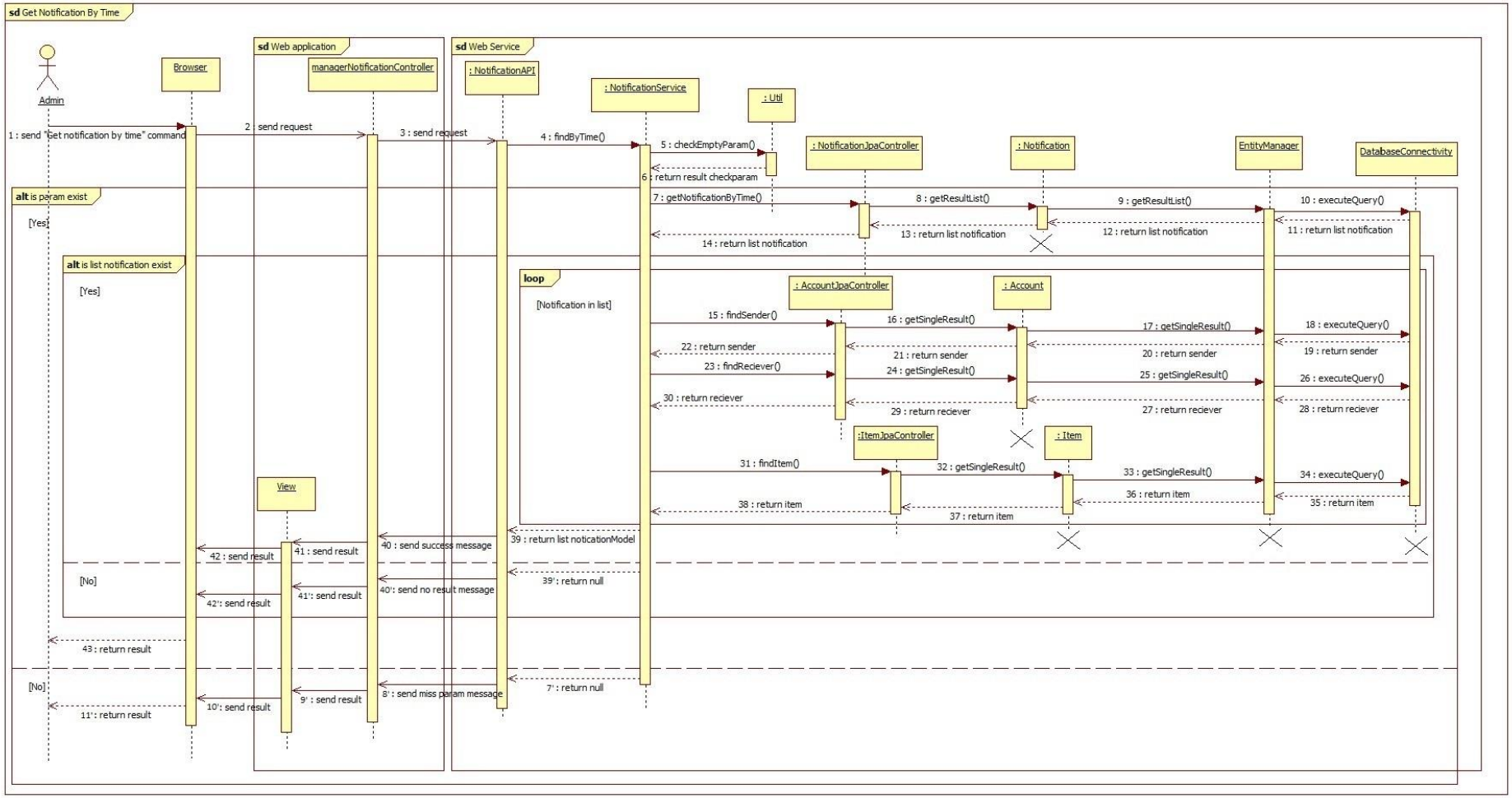


Figure 46. Get Item Movement Log Sequence Diagram

4.3.3 Mobile Application

4.3.3.1 <Visitor>

4.3.3.1.1 Get Item Information

Summary: this diagram show process of get item information

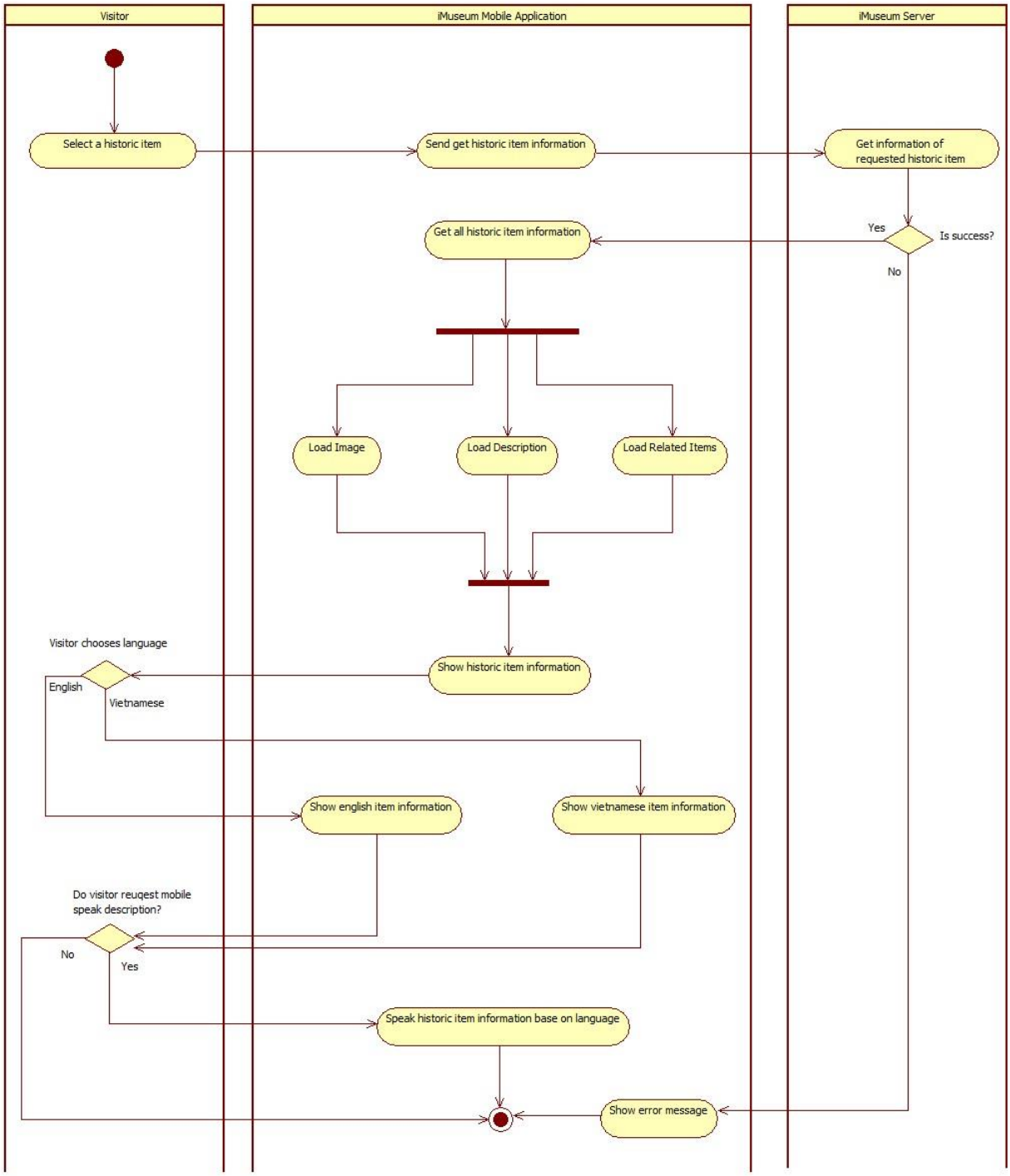


Figure 47. Get Item Information Activity Diagram

4.3.2.1.3 Report Item Information

Summary: this diagram show process of report item information

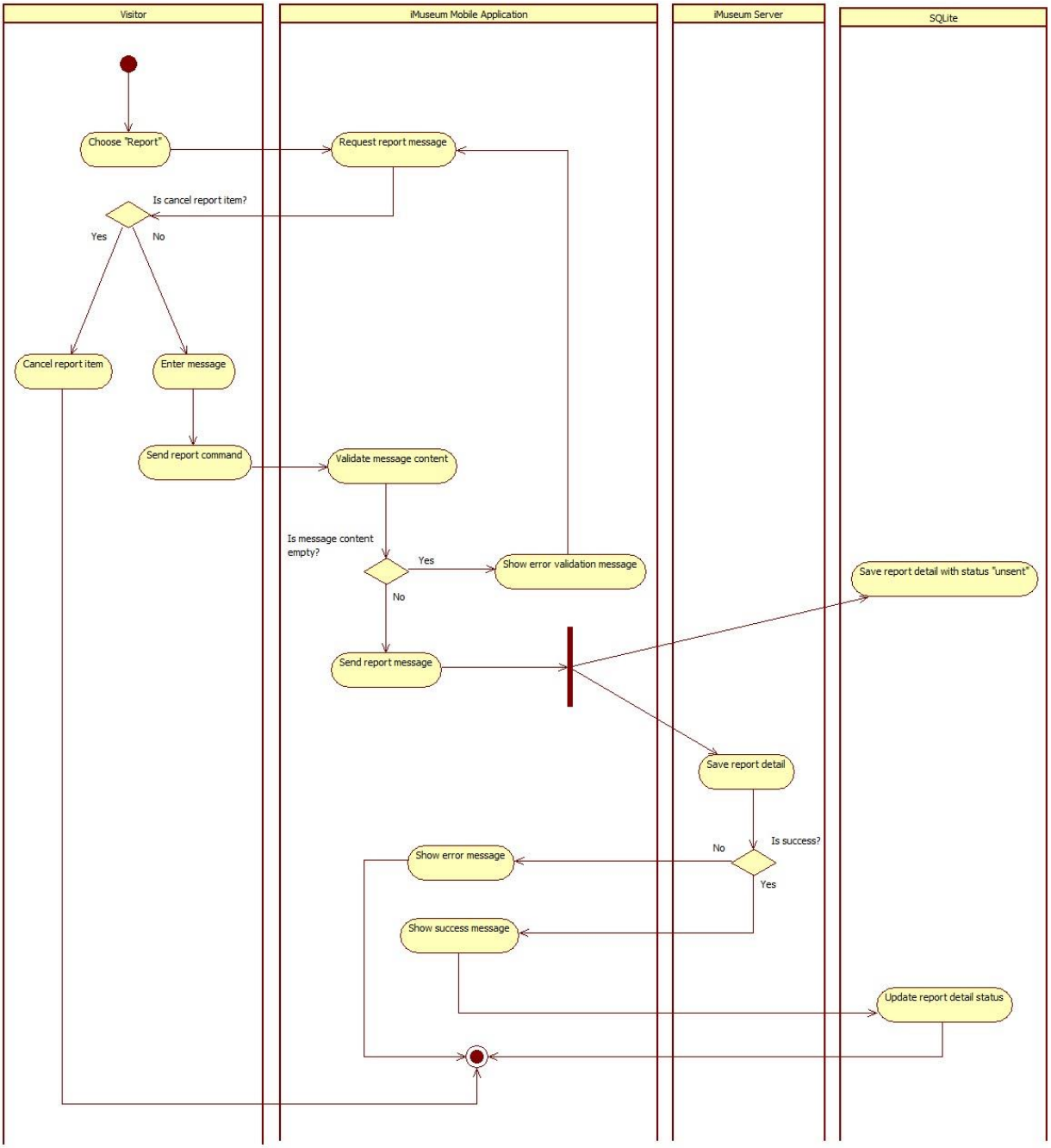


Figure 48. Report Item Information Activity Diagram

4.3.2.1.4 Get Related Item Location

Summary: this diagram show process of get related item location

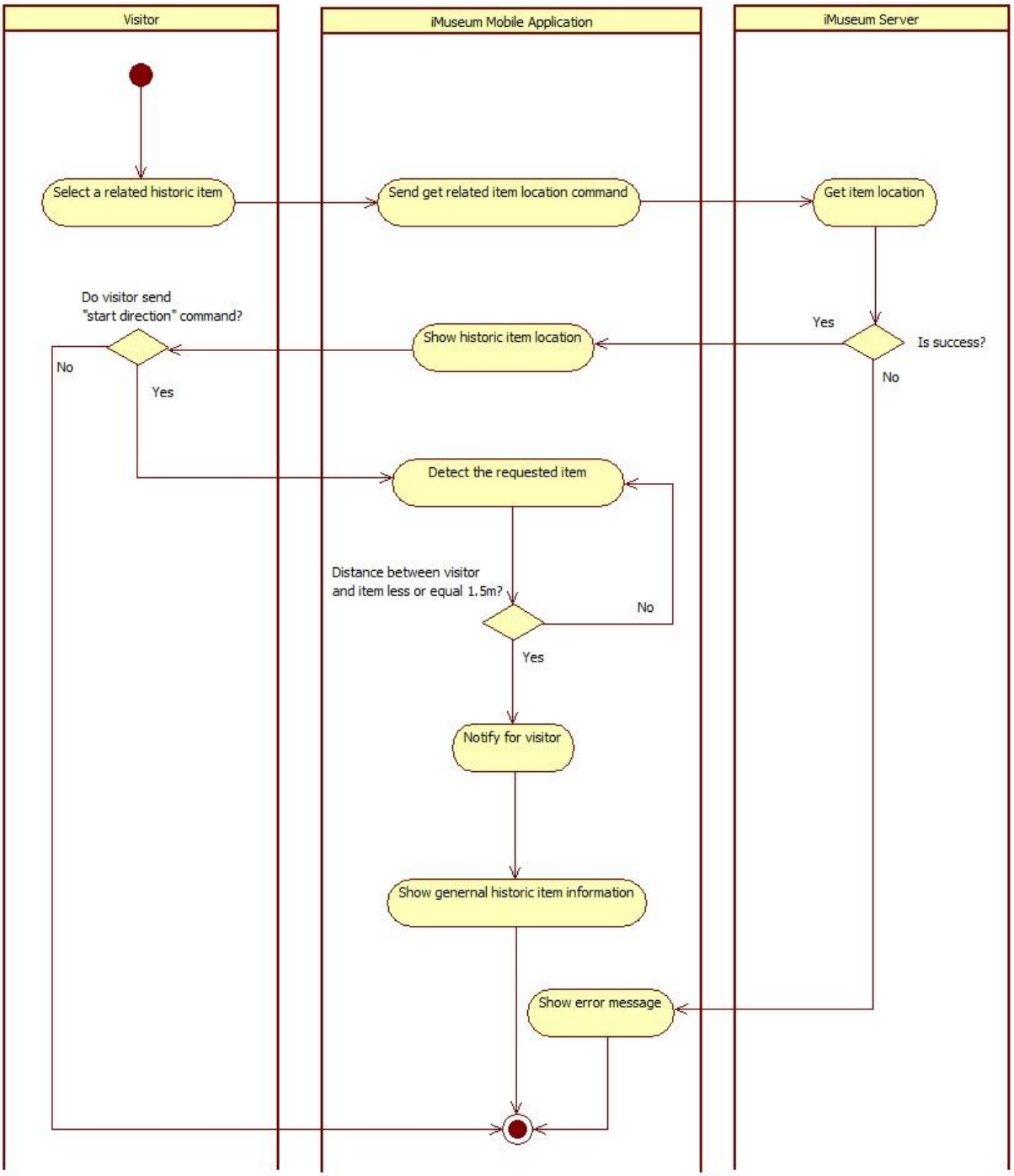


Figure 49. Get Related Item Location Activity Diagram

4.3.2.1.5 Add Favorite Item

Summary: this diagram show process of add favorite item

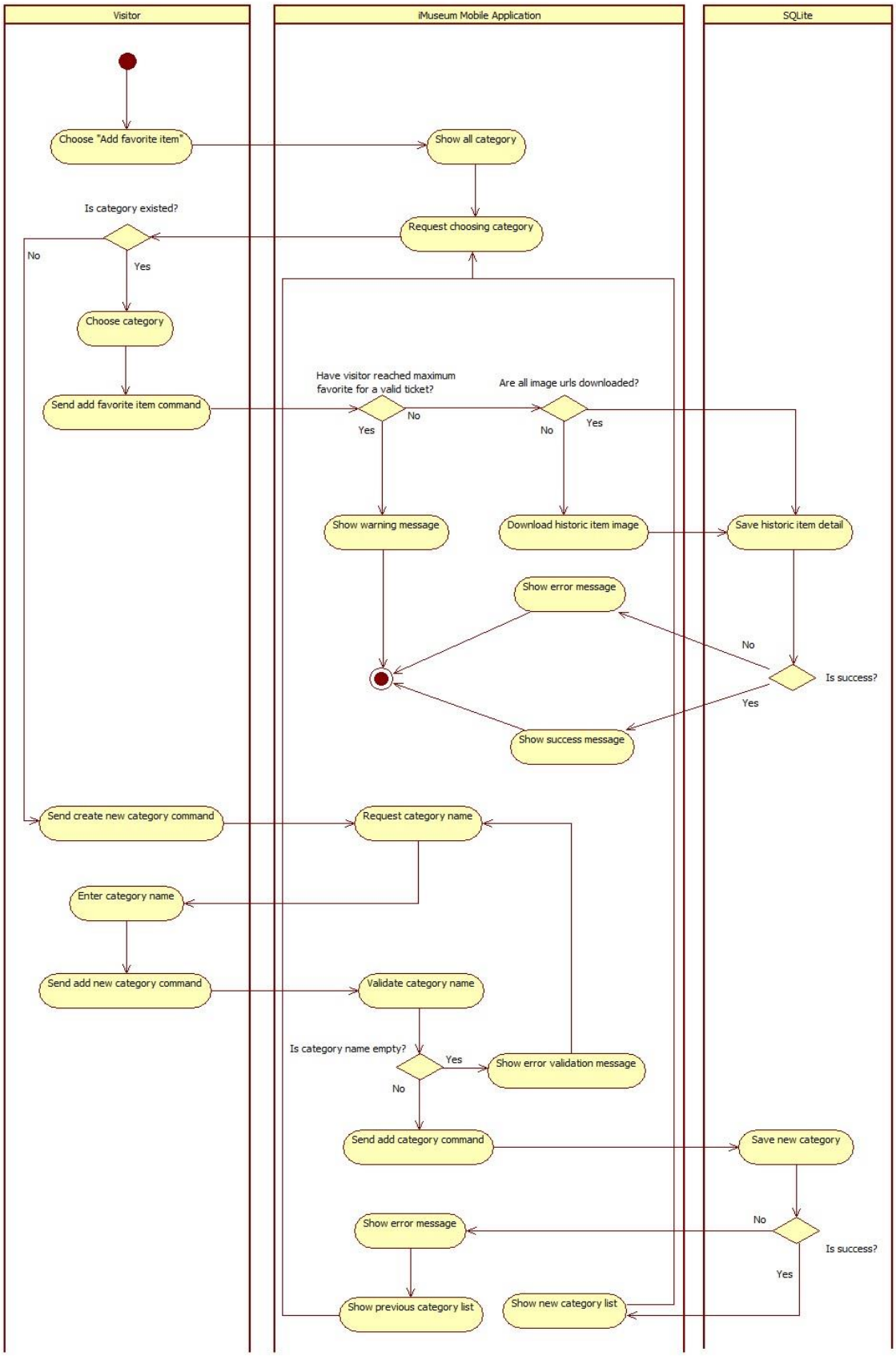


Figure 50. Add Favorite Item Activity Diagram

4.3.2.1.6 Get Museum Map

Summary: this diagram show process of get museum map

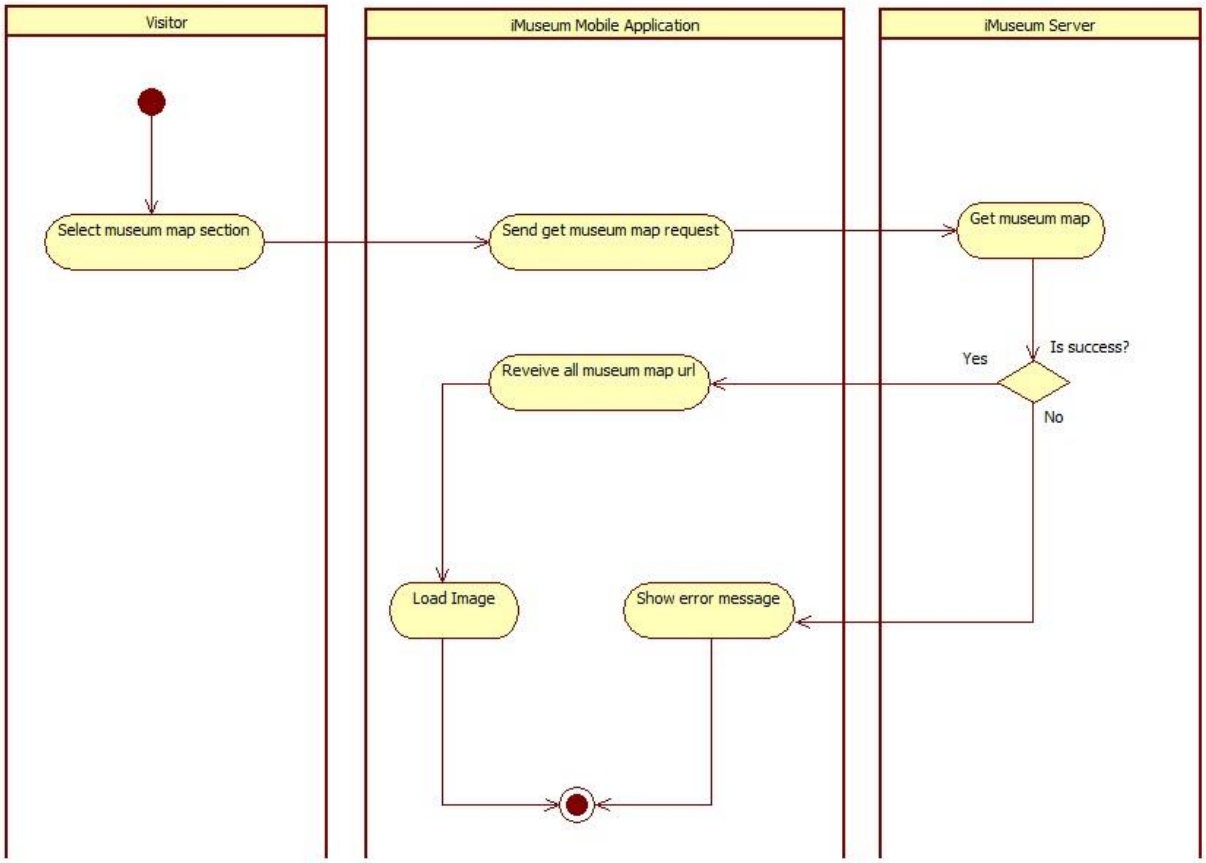


Figure 51. Get Museum Map Activity Diagram

4.3.3.2 <Staff>

4.3.2.2.1 Track item

Summary: this diagram show process of track item

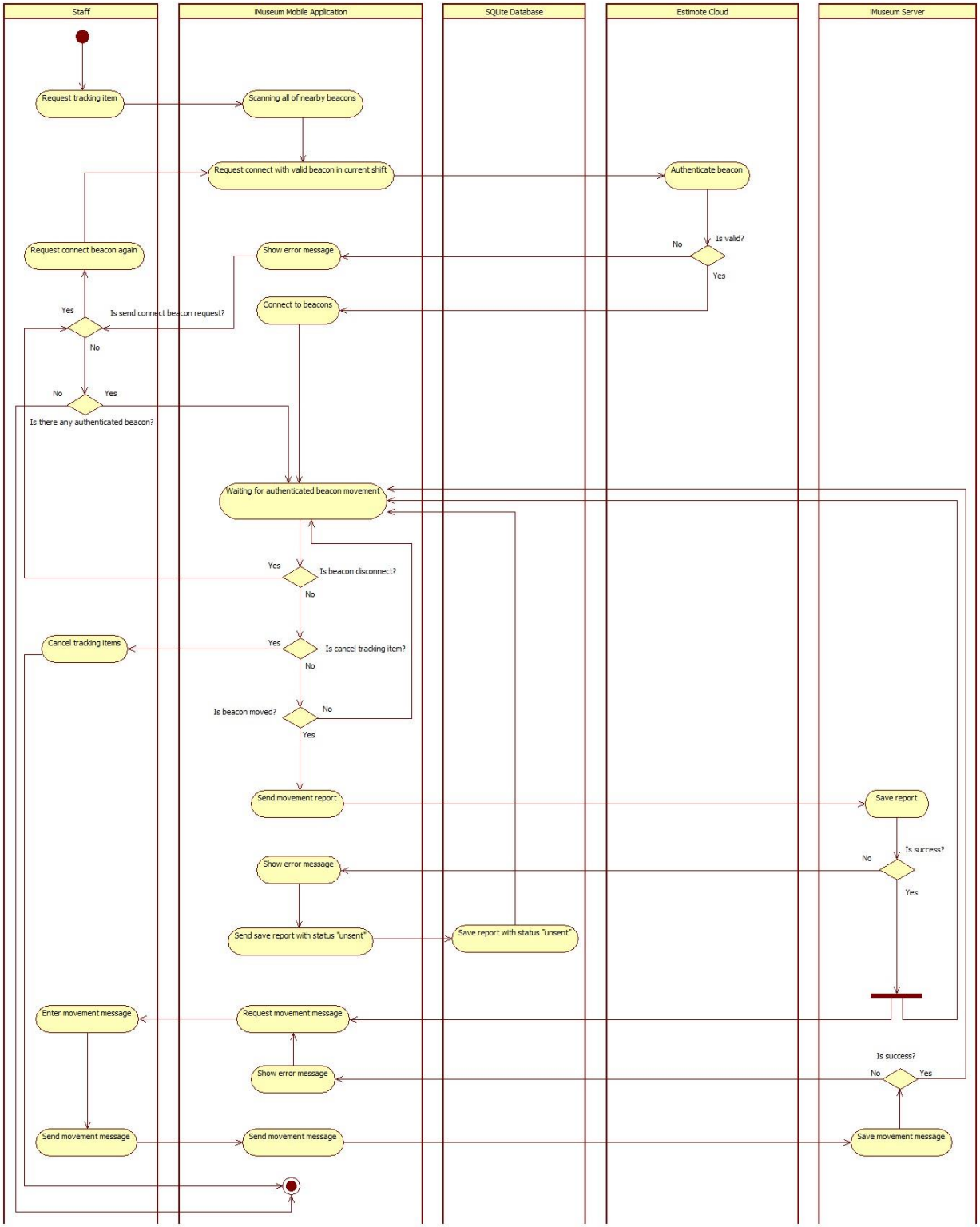


Figure 52. Track Item Activity Diagram

4.3.3.3 <Admin>

4.3.2.3.2 Map Beacon

Summary: this diagram show process of map beacon

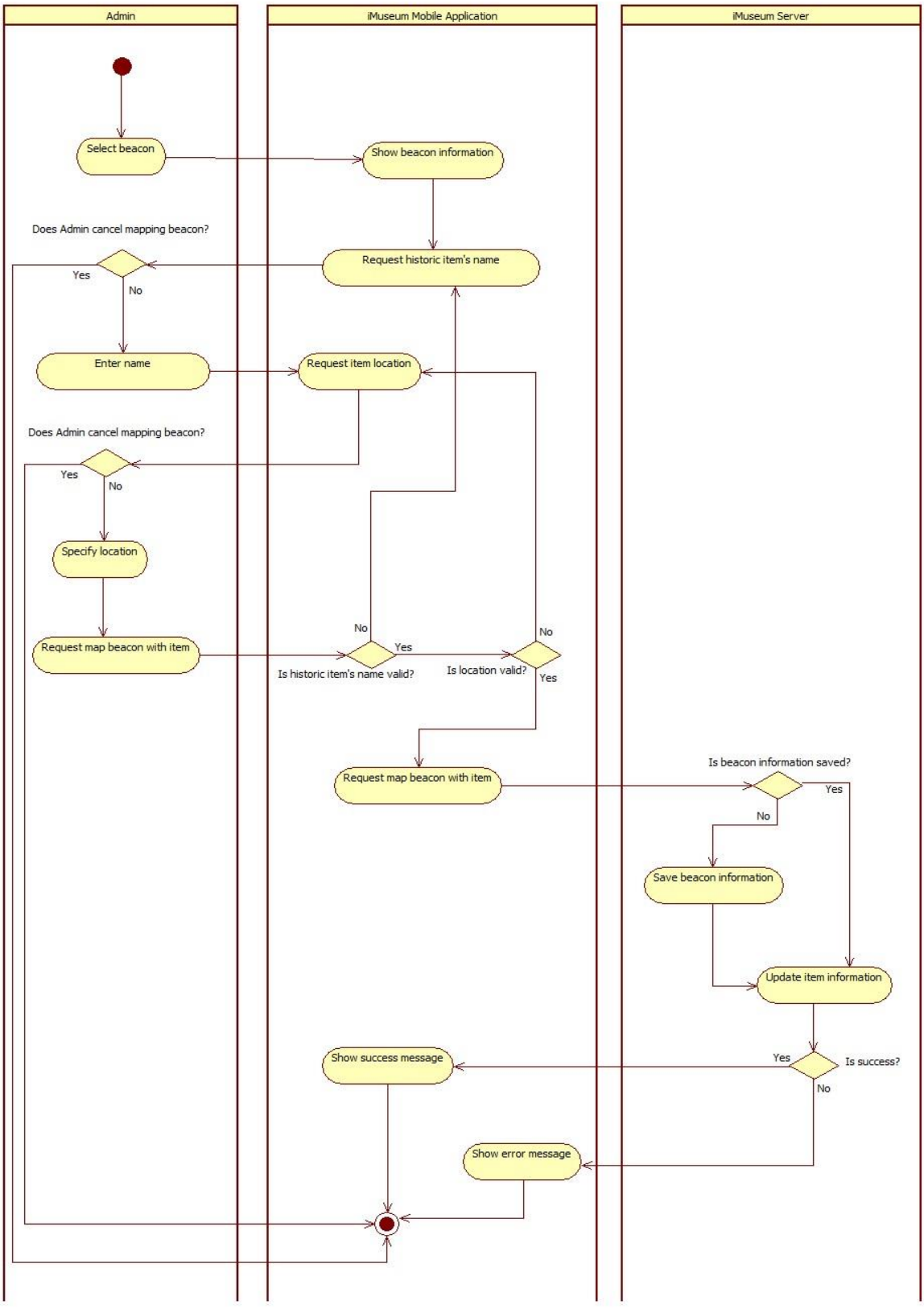


Figure 53. Map Beacon Activity Diagram

5. Interface

5.1 Component Interface

Component Interface are Web Service API which will be used by both Web Application and Mobile Application.

Signature	Description	Input	Output	Output Format	Exception
public String checkLogin (@Formparam(“username”) String username, @Formparam(“password”) String password)	Check username and password when admin login.	username : String password : String	String response	String	N/A
public String checkLoginMobile (@Queryparam(“username”) String username, @ Queryparam(“password”) String password)	Check username and password when staff login	username : String password : String	String response	String	N/A
public String updateBeacon (@FormParam("name") String name, @FormParam("id") int id)	Update beacon name by beacon id	name : String id : int	String response	String	N/A
public String removeNotDeletedBeacon (@FormParam("id") int id)	Change status of beacon is “deleted”.	id : int	String response	String	N/A
public String removeDeletedBeacon (@FormParam("id") int id)	Remove beacon when status of beacon is deleted	id : int	String response	String	N/A
public String getNotDeletedBeacon()	Get all beacon that status of beacon is “Not_deleted”	N/A	String response	String	N/A
public String getDeletedBeacon()	Get all beacon that status of beacon is “Deleted”	N/A	String response	String	N/A
public String getBeaconByName (@QueryParam("name") String name)	Get beacon by name	name : String	String response	String	N/A
public String getBeaconByMacAddress (@QueryParam("macAddress") String macAddress)	Get beacon by mac_address	macAddress:String	String response	String	N/A

public String insertBeacon (@QueryParam("name") String name, @QueryParam("uuid") String uuid, @QueryParam("major") int major, @QueryParam("minor") int minor, @QueryParam("macAddress") String macAddress)	Add new beacon with information include: name, uuid, major, minor, macAddress to storage.	name : String uuid : String major : int minor : int macAddress : String	String response	String	N/A
public String recoverNotDeletedBeacon (@FormParam("id") int id)	Update status of beacon is “Not_Deleted”	id : int	String response	String	N/A
public String insertDevice (@QueryParam("id") String id, @QueryParam("registrationID") String registrationID)	Add a device with information include : id, registrationID to storage	id: String registrationID : String	String response	String	N/A
public String updateDevice (@QueryParam("accountID") int accountID, @QueryParam("id") String id)	Assign device to museum staff.	accountID : int, id : String	String response	String	N/A
public String getAllFloor()	Get all floor	N/A	String response	String	N/A
public String filterByFloorID (@QueryParam("floorID") int floorID)	Get all room and area by identifier floor	florID: int	String response	String	N/A
public String getItemInfoByID (@QueryParam("id") int id)	Get Item by identifier	id: int	String response	String	N/A
public String findByBeaconID (@QueryParam("beaconID") int beaconID)	Get Item by identifier beacon	beaconID : int	String response	String	N/A
public String getItemDeleted()	Get all Item that has status is “Deleted”	N/A	String response	String	N/A
public String getItemNotDeleted()	Get all Item that has status is “Not_deleted”	N/A	String response	String	N/A
public String getItemInfo (@QueryParam("macAddress") String macAddress)	Get Item by macAddress of beacon	macAddress: String	String response	String	N/A

public String insertItem (@FormParam("name") String name, @FormParam("linkImage") String linkImage, @FormParam("description") String description, @FormParam("beaconID") int beaconID, @FormParam("areaID") int areaID, @FormParam("roomID") int roomID, @FormParam("floorID") int floorID)	Add item with information include: name, linkImage, description, beaconID, areaID, roomID,floorID	name : String linkImage: String description: String beaconID: int areaID: int roomID: int floorID: int	String response	String	N/A
public String updateItem (@FormParam("name") String name, @FormParam("linkImage") String linkImage, @FormParam("description") String description, @FormParam("beaconID") int beaconID, @FormParam("areaID") int areaID, @FormParam("roomID") int roomID, @FormParam("floorID") int floorID, @FormParam("id") int id)	Update item information by item identifier	name : String linkImage: String description: String beaconID: int areaID: int roomID: int floorID: int id : int	String response	String	N/A
public String removeNotDeletedItem (@FormParam("id") int id)	Change item status is “Deleted”	id : int	String response	String	N/A
public String removeDeletedItem (@FormParam("id") int id)	Remove item when item status is “Deleted”	id : int	String response	String	N/A
public String findLocationOfItem (@QueryParam("id") int id)	Get item location by item identifier	id: int	String response	String	N/A
public String getListNearByItem (@QueryParam("id") int id)	Get list of item that nearby specific item by identifier item.	id : int	String response	String	N/A
public String getItemBeacon()	Get list item that is mapped beacon	N/A	String response	String	N/A

public String recoverNotDeletedBeacon (@FormParam("id") int id)	Change item status is “Not_deleted”	id: int	String response	String	N/A
public String findAll()	Get all notification	N/A	String response	String	N/A
public String findByTime (@QueryParam("startDate") long startDate, @QueryParam("endDate") long endDate)	Get all notification that occur time is between startDate and endDate	startDate : long endDate : long	String response	String	N/A
public String notifyMovedItem (@QueryParam("macAddress") String macAddress, @QueryParam("username") String username)	Add a new notification when item is moved	macAddress: String username: String	String response	String	N/A
public String updateNotifyMovedItem (@QueryParam("id") int id, @QueryParam("username") String username, @QueryParam("message") String message)	Update information include new message and new sender of notification that has added	id: int username: String message: String	String response	String	N/A
public String insertReport (@QueryParam ("itemID") int itemID, @QueryParam("date") long lDate, @QueryParam("message") String message)	Add a new report when visitor report to item	itemID: int message: String lDate: long	String response	String	N/A
public String findAll()	Get all reports	N/A	String response	String	N/A
public String getNumberOfUnSeenReport ()	Get all reports that is not seen by museum staff	N/A	String response	String	N/A
public String getReportByTime (@QueryParam("startDate")long startDate,@QueryParam("endDate") long endDate)	Get reports that time occur is between startDate and endDate	startDate: long endDate: long	String response	String	N/A
public String getRoomModel (@QueryParam("roomID") int roomID)	Get room by identifier	roomID: int	String response	String	N/A
public String insertTicket()	Release a new ticket when need	N/A	String response	String	N/A

public String checkTicketCode (@QueryParam("ticketcode") String ticketCode,@QueryParam("deviceID") String deviceID)	Check ticket code.	ticketCode: String deviceID: String	String response	String	N/A

Table 78. Component Interface

5.1User Interface Design

5.1.1 Web app

5.1.1.1 Login

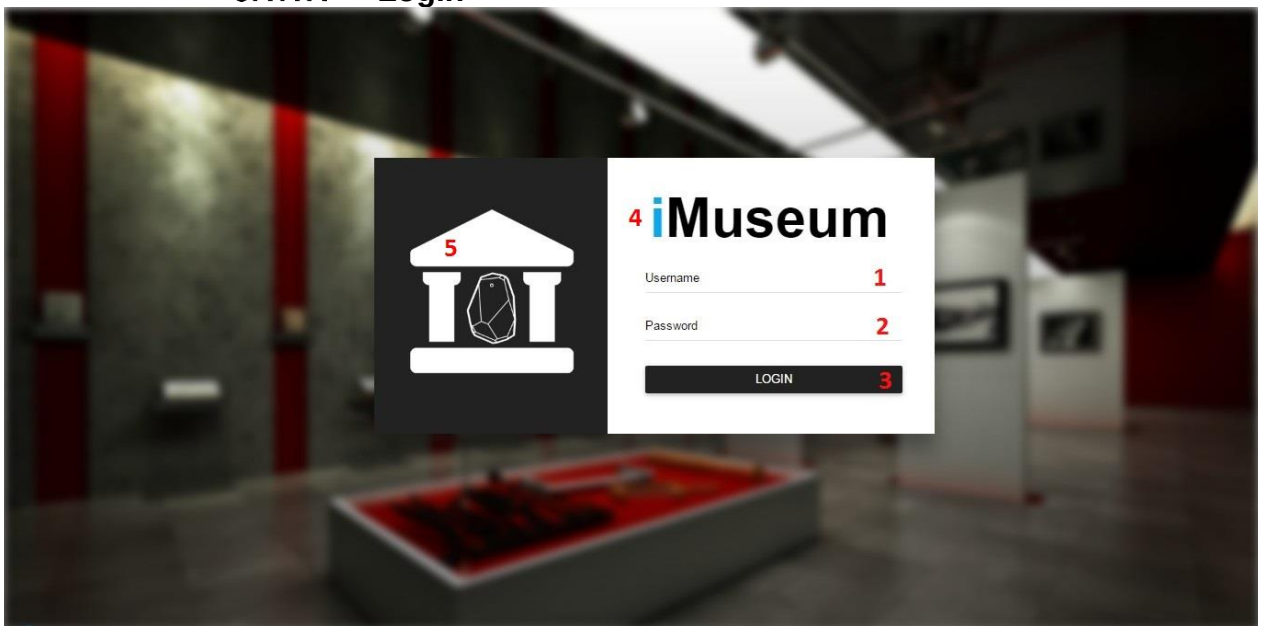


Figure 54. Login

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Username	Fill Username	No	Yes	Textbox	String
2	Password	Fill Password	No	Yes	Password	String
4	NameWeb	Name of web	Yes	No	H1	String
5	Logo	Logo of web	Yes	No	Image	N/A

Table 79. <Web Apps> Login Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
3	Login	Login into system	All fields required before summit	Redirect to movement log page

Table 80. <Web Apps> Login Button/Hyperlinks

5.1.1.2 Expert

5.1.1.2.1 Items Management

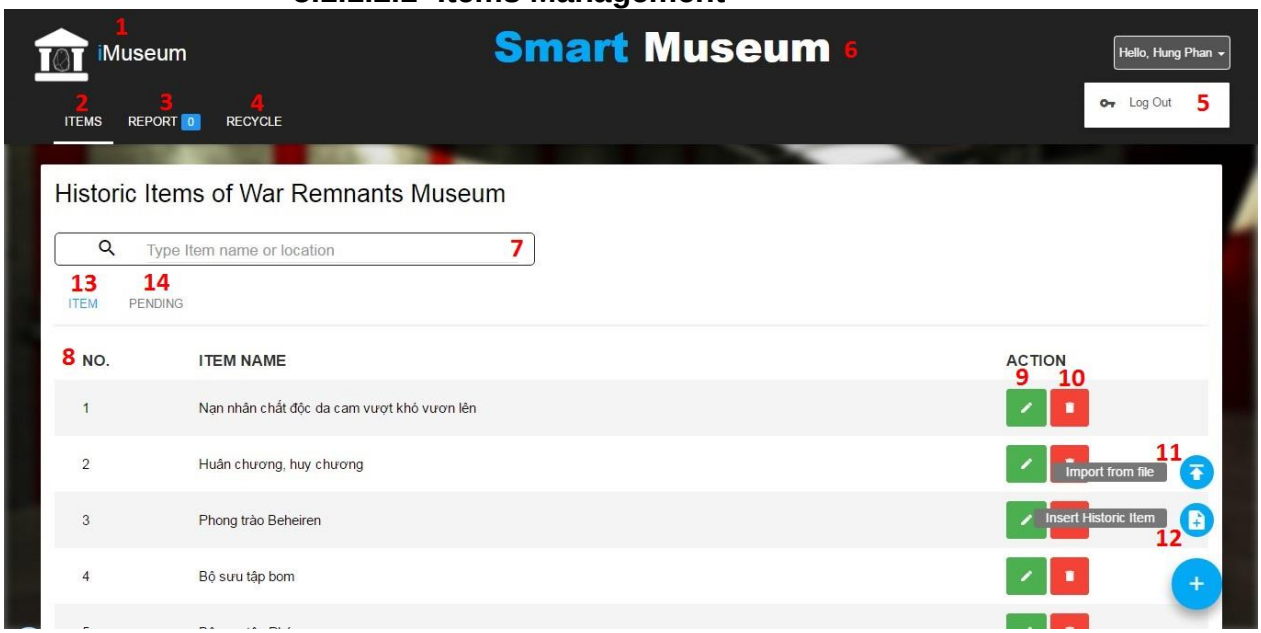


Figure 55. Items management

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
6	NameMuseum	Name of museum	Yes	No	Text	String
7	SearchItem	Search item by name or location	No	No	Textbox	String
8	itemData	Information of historic items	Yes	No	Table	String

Table 81. <Web apps> Item Management Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	Home	Load home page	N/A	Redirect to items management page
2	Item	Load historic items management page	N/A	Redirect to items management page
3	Report	Load report management page	N/A	Redirect to report management page
4	Recycle	Load recycle bin page	N/A	Redirect to recycle bin page
5	Logout	Log out account	N/A	Redirect to login page

9	Update	Update historic item with beacon	N/A	Show modal update item
10	Delete	Delete a historic item	N/A	Show modal delete item
11	Import	Import data in file (.txt) into create modal	N/A	Show modal create item
12	Create	Create a historic item	N/A	Show modal create item
13	itemTab	Tab items management	N/A	Show tab items management
14	pendingTab	Tab pending items management	N/A	Show tab pending items management

Table 82. <Web apps> Items Management Buttons/Hyperlinks

5.1.1.2.2 Create/Update item

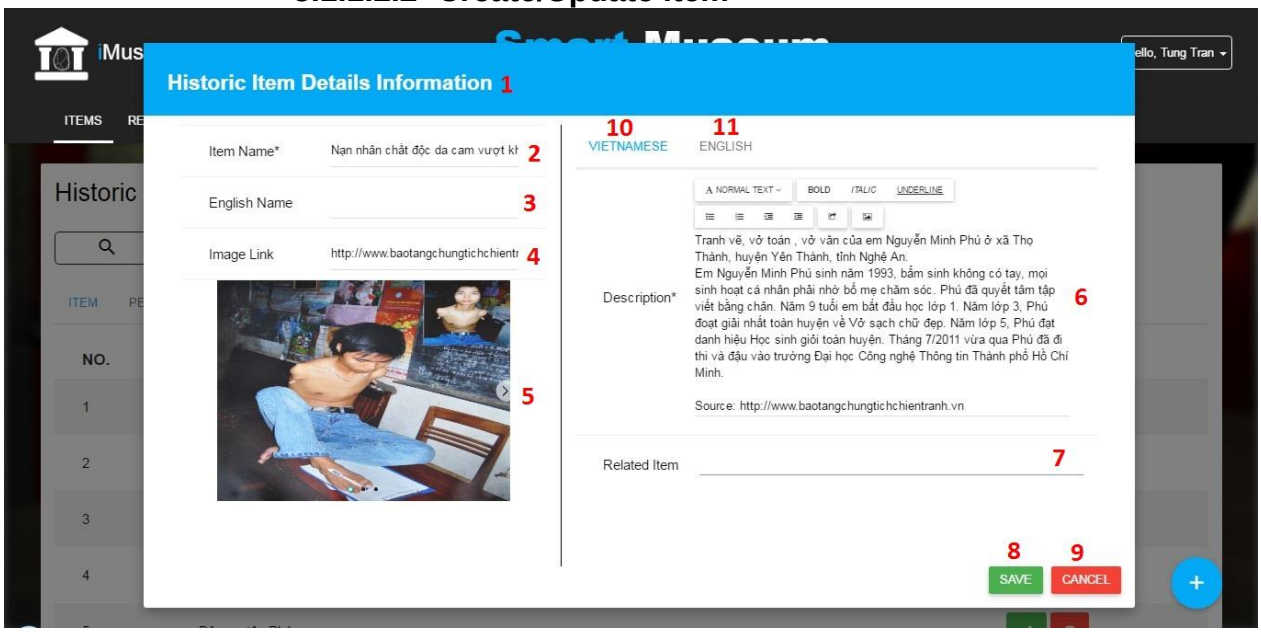


Figure 56. Create/Update item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Name Modal	Name of modal	Yes	No	H3	String
2	Item Name	Name of item	No	Yes	Textbox	String
3	English Name	English name of item	No	Yes	Textbox	String
4	Image Link	Link image of item	No	No	Textbox	String

5	Image Item	Image of item	Yes	No	Image	N/A
6	Description	Description of item	No	Yes	Text editor	String
7	Related item	Related item of item	No	No	Textbox	String

Table 83. <Web apps> Create/Update item Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
8	Save	Send create/update item command	Validation all required fields before summit	Summit create/update item
9	Cancel	Close create/update item modal	N/A	Close create/update item modal
10	Vietnamese	Tab Vietnamese description	N/A	Show tab Vietnamese description
11	English	Tab English description	N/A	Show tab English description

Table 84. <Web apps> Create/Update item Buttons/Hyperlinks

5.1.1.2.3 Delete Item

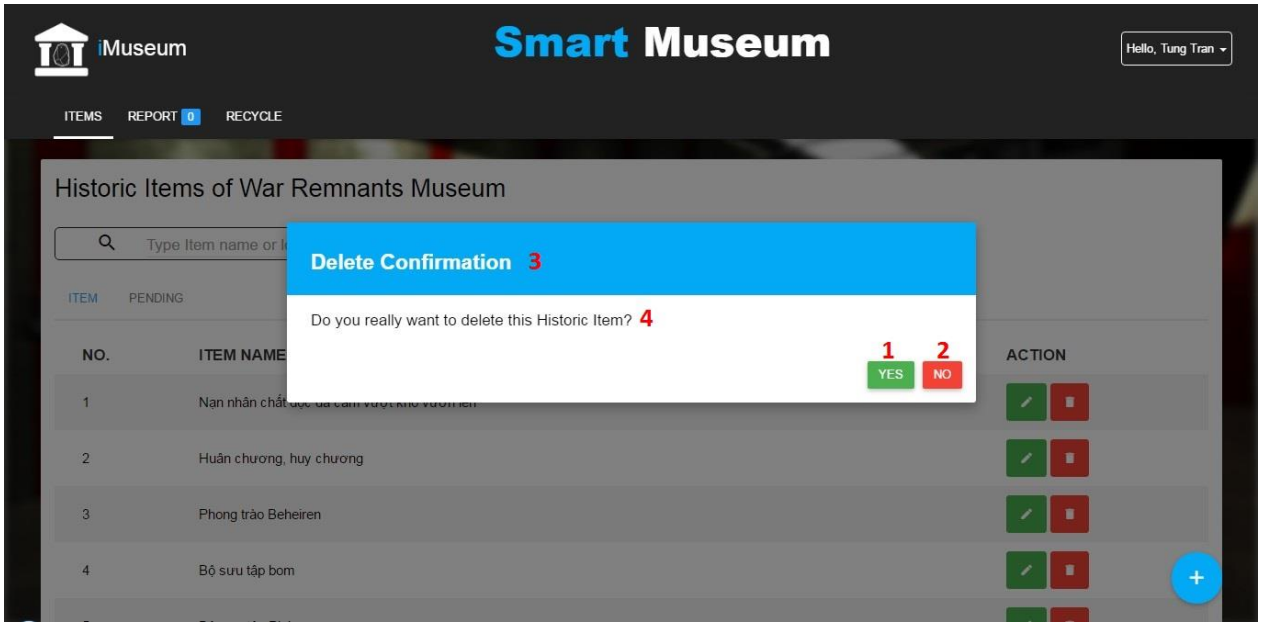


Figure 57. Delete item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
3	Name Modal	Name of modal	Yes	No	H3	String

4	ConfirmDelete	Text confirm action delete item	Yes	No	H4	String

Table 85. <Web apps> Delete Item Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	Yes	Send delete item command	N/A	Summit delete item
2	No	Close delete item modal	N/A	Close delete item modal

Table 86. <Web apps> Delete Item Buttons/Hyperlinks

5.1.1.2.4 Approved Pending Item

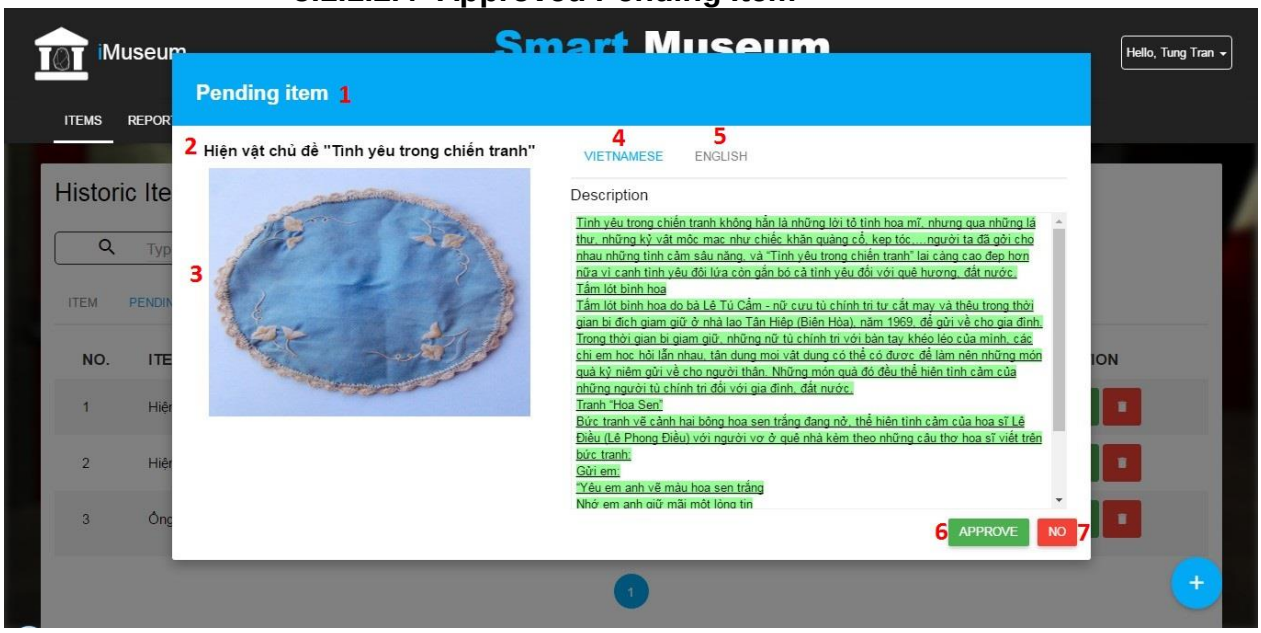


Figure 58. Approved Pending Item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Name Modal	Name of modal	Yes	No	H3	String
2	Item Name	Name of item	Yes	Yes	Textview	String
3	Image Item	Image of item	Yes	No	Image	N/A

Table 87. <Web apps>Approved Pending ItemFields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome

4	Vietnamese	Tab Vietnamese description	N/A	Show tab Vietnamese description
5	English	Tab English description	N/A	Show tab English description
6	Approve	Send approved pending item command	N/A	Summit approved pending item
7	No	Close approved item modal	N/A	Close approved item modal

Table 88. <Web apps> Approved Pending Items Buttons/Hyperlinks

5.1.1.2.5 Report management

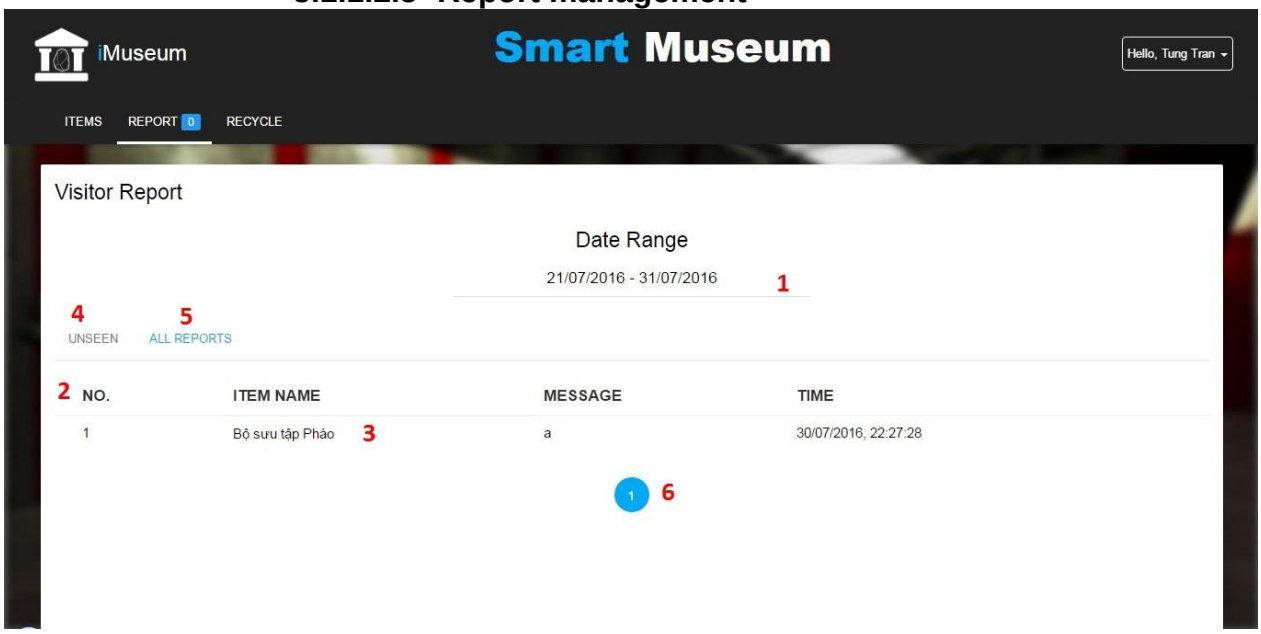


Figure 59. Report management

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	DateRange	Fill date	No	Yes	DateRangePicker	String
2	reportData	Information of report	Yes	No	Table	String

Table 89. <Web apps> Report Management Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
3	checkReport	Check report by update item	N/A	Show update item modal
4	unseen	Tab view unseen report	N/A	Show tab view unseen report

5	allReport	Tab view all report	N/A	Show tab view all report
6	pageNumber	Paging of list report item	N/A	Switch to other report pages

Table 90. <Web apps> Report Management Buttons/Hyperlinks

5.1.1.2.6 Update item in report page

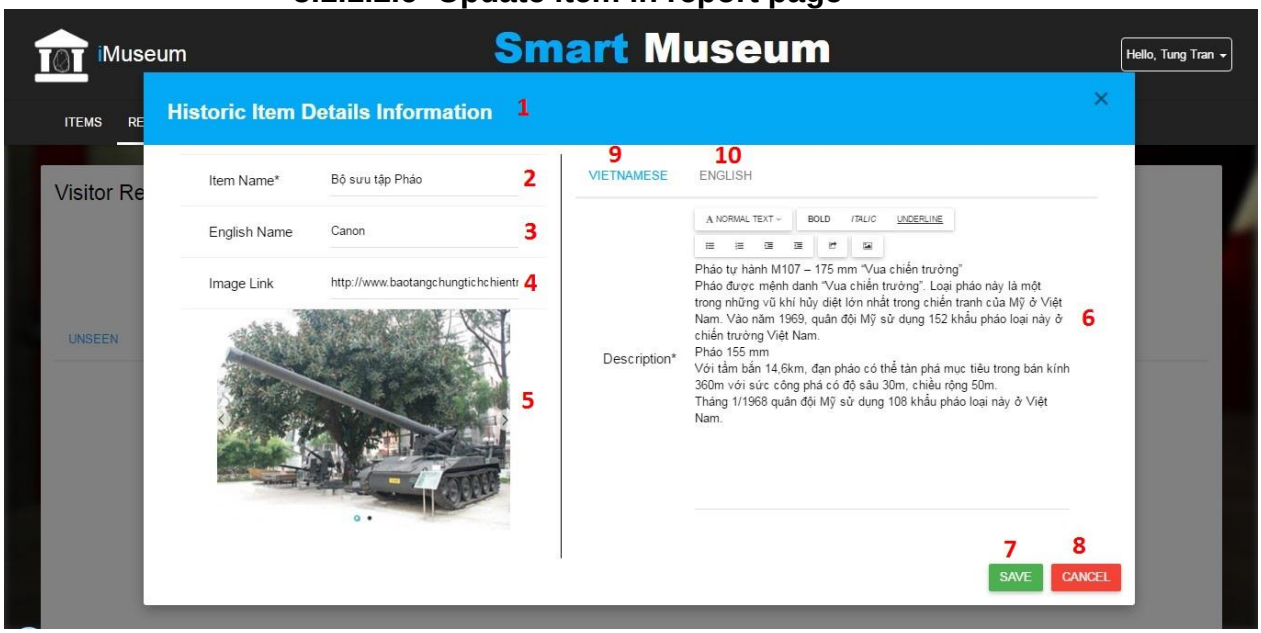


Figure 60. Update item in report page

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Name Modal	Name of modal	Yes	No	H3	String
2	Item Name	Name of item	No	Yes	Textbox	String
3	English Name	English name of item	No	Yes	Textbox	String
4	Image Link	Link image of item	No	No	Textbox	String
5	Image Item	Image of item	Yes	No	Image	N/A

6	Description	Description of item	No	Yes	Text editor	String

Table 91. <Web apps> Create/Update item Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
7	Save	Send create/update item command	Validation all required fields before summit	Summit create/update item
8	Cancel	Close create/update item modal	N/A	Close create/update item modal
9	Vietnamese	Tab Vietnamese description	N/A	Show tab Vietnamese description
10	English	Tab English description	N/A	Show tab English description

Table 92. <Web apps> Create/Update item Buttons/Hyperlinks

5.1.1.3 Admin

5.1.1.3.1 Movement Log

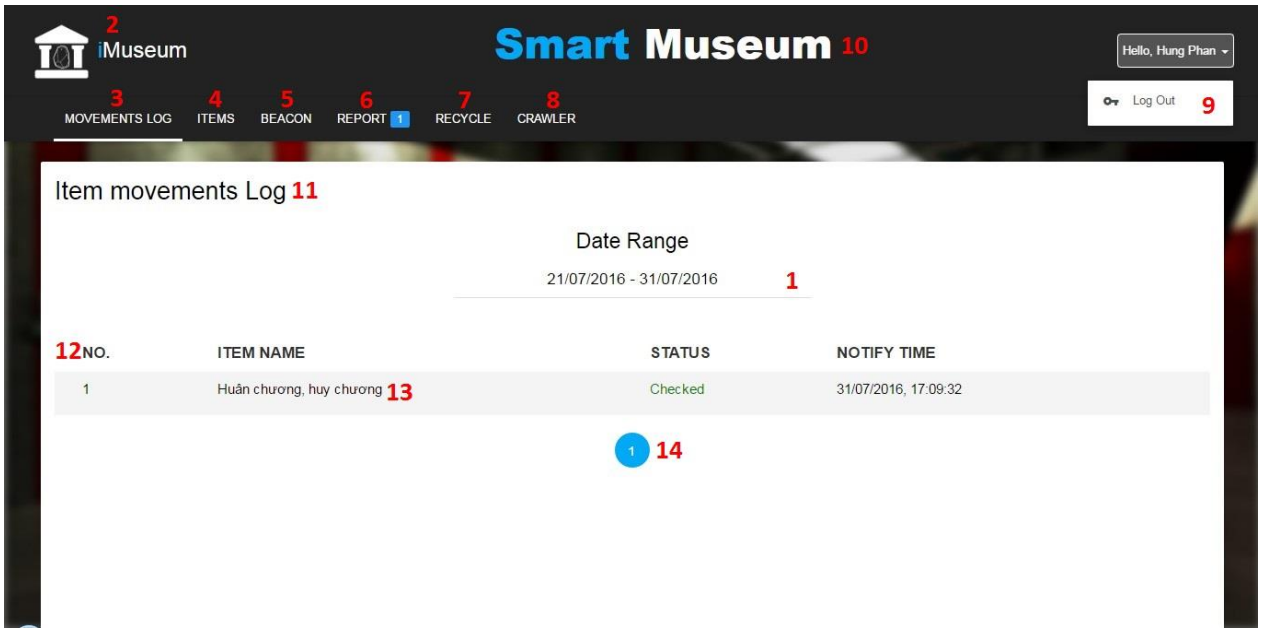


Figure 61. Movement log

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	DateRange	Fill date	No	Yes	DateRangePicker	String
10	NameMuseum	Name of museum	Yes	No	Text	String

11	NamePage	Name of page	Yes	No	Div	String
12	logData	Information of movement log	Yes	No	Table	String

Table 93. <Web apps> Movement Log

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
2	Home	Load home page	N/A	Redirect to movement log page
3	MovementLog	Load beacon movement log page	N/A	Redirect to movement log page
4	ItemManagement	Load historic items management page	N/A	Redirect to items management page
5	BeaconManagement	Load beacon management page	N/A	Redirect to beacon management page
6	VisitorReport	Load report management page	N/A	Redirect to report management page
7	RecycleBin	Load recycle bin page	N/A	Redirect to recycle bin page
8	Crawler	Load crawler configuration page	N/A	Redirect to crawler configuration page
9	Logout	Log out account	N/A	Redirect to login page
13	viewDetail	View detail of log	N/A	Show view detail movement log modal
14	pageNumber	Paging of list movement log	N/A	Switch to other movement log pages

Table 94. <Web apps> Movement Log Buttons/Hyperlinks

5.1.1.3.2 Beacon Management

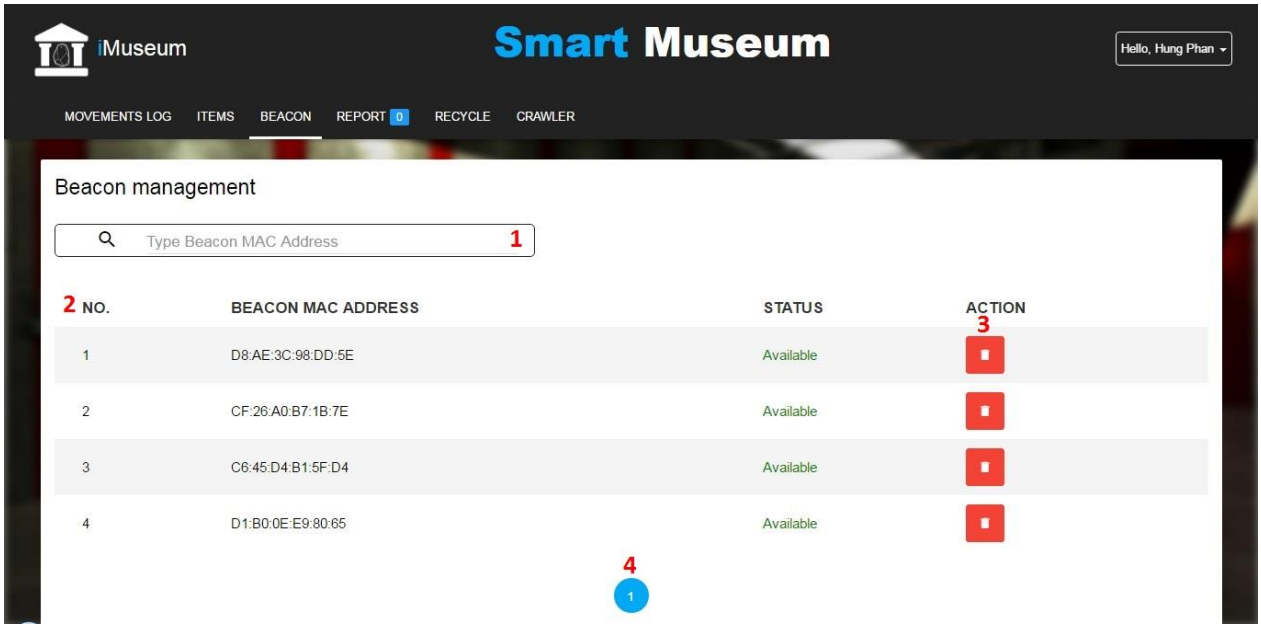


Figure 62. Beacon management

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	SearchBeacon	Search beacon by mac address	No	No	Textbox	String
2	beaconData	Information of beacon	No	No	Table	String

Table 95. <Web apps> Beacon Management Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
3	Delete	Delete item	N/A	Show modal delete item
4	pageNumber	Paging of list beacons	N/A	Switch to other beacon pages

Table 96. <Web apps> Beacon Management Buttons/Hyperlinks

5.1.1.3.3 Assign report

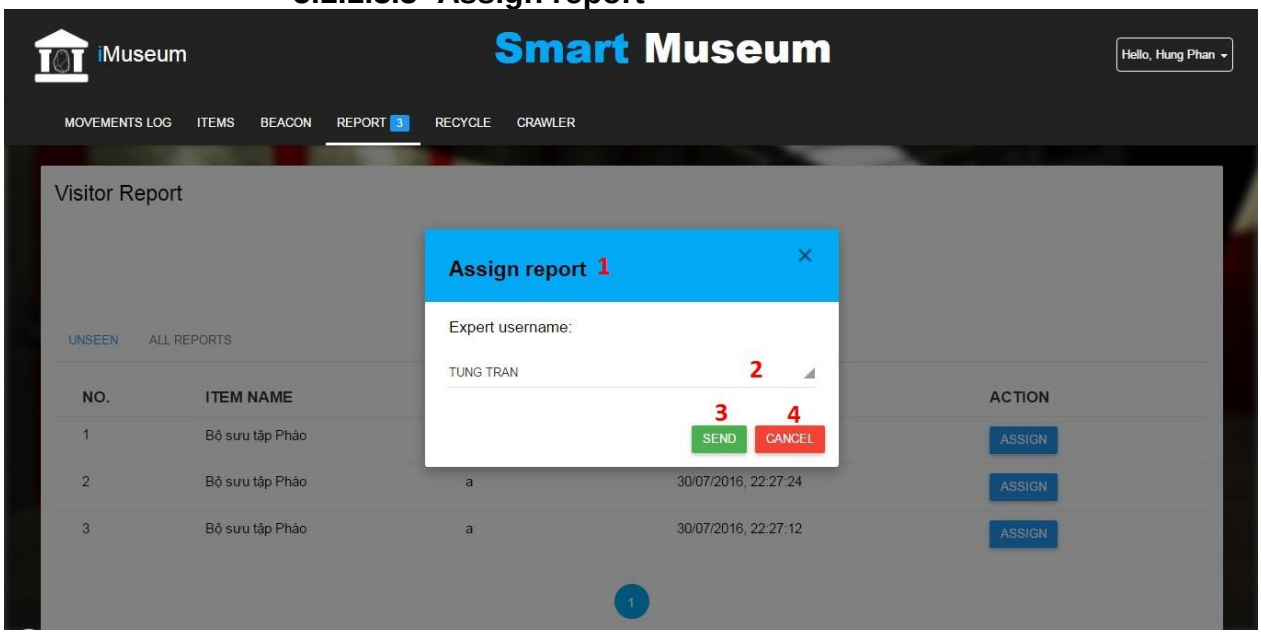


Figure 63. Assign report

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Name Modal	Name of modal	Yes	No	H3	String
2	Expert name	Name of expert	No	Yes	Dropdown list	String

Table 97. <Web apps> Assign report Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
3	Send	Send assign report command	N/A	Summit assign report
4	Cancel	Close assign report modal	N/A	Close assign report modal

Table 98. <Web apps> Assign report Buttons/Hyperlinks

5.1.1.3.4 Recycle Bin Item

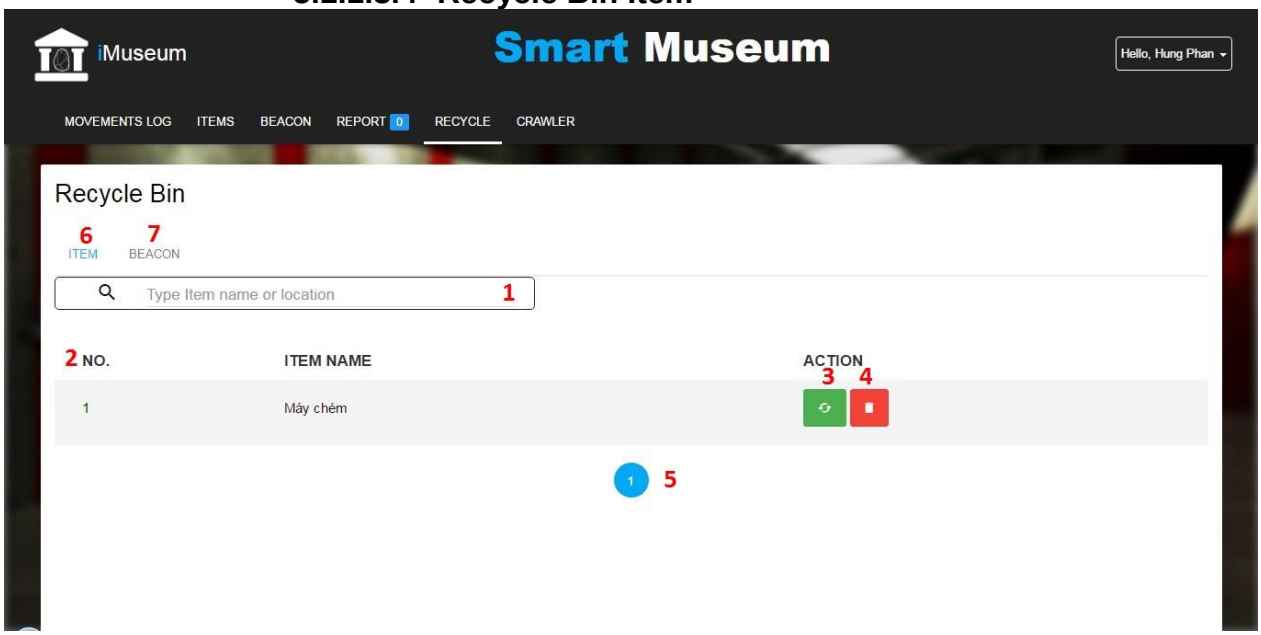


Figure 64. Recycle bin item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	SearchItem	Search item by name	No	No	Textbox	String
2	ItemData	Information of item	No	No	Table	String

Table 99. <Web app> Recycle Bin Item Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
3	Recover	Send recover item command	N/A	Summit recover item
4	Delete	Delete item	N/A	Show modal delete item
5	pageNumber	Paging of list recycle bin item	N/A	Switch to other recycle bin item pages
6	itemTab	Tab view recycle bin item	N/A	Show tab view recycle bin item
7	beaconTab	Tab view recycle bin beacon	N/A	Show tab view recycle bin beacon

Table 100. <Web app> Recycle Bin Items Buttons/Hyperlinks

5.1.1.3.5 Recycle Bin Beacon

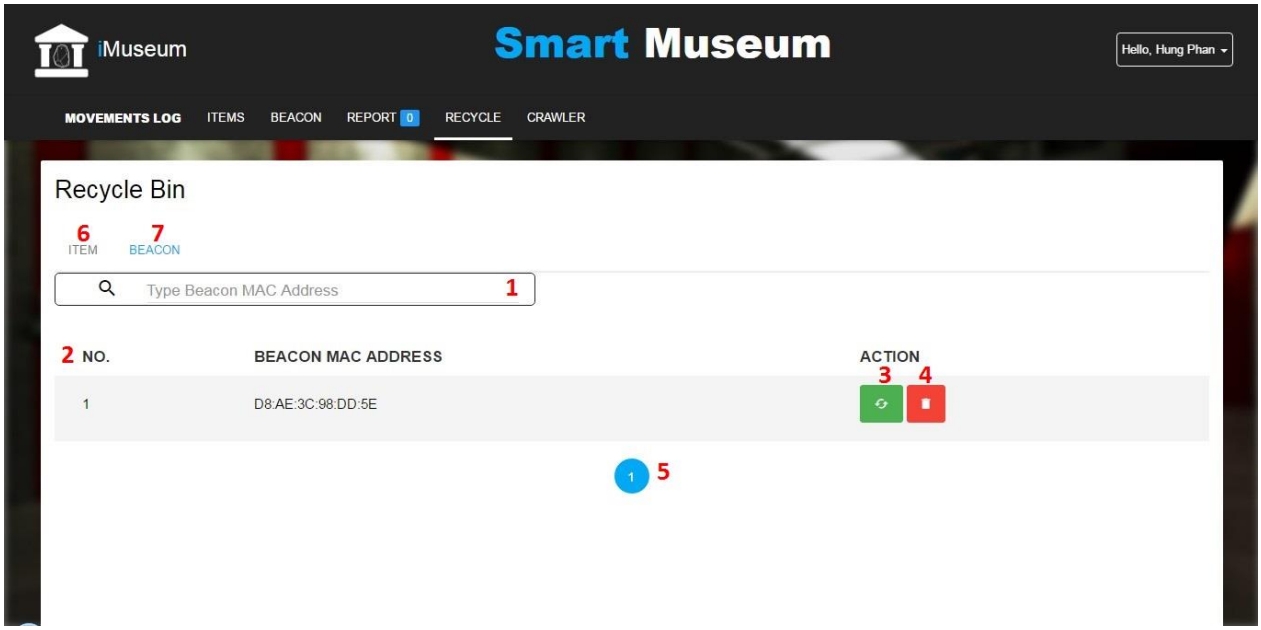


Figure 65. Recycle bin beacon

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	SearchBeacon	Search beacon by mac address	No	No	Textbox	String
2	beaconData	Information of beacon	No	No	Table	String

Table 101. <Web app> Delete Beacon in Recycle Bin Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
3	Recover	Send recover beacon command	N/A	Summit recover beacon
4	Delete	Delete beacon	N/A	Show modal delete beacon
5	pageNumber	Paging of list recycle bin beacon	N/A	Switch to other recycle bin beacon pages
6	itemTab	Tab view recycle bin item	N/A	Show tab view recycle bin item
7	beaconTab	Tab view recycle bin beacon	N/A	Show tab view recycle bin beacon

Table 102. <Web app> Delete Beacon in Recycle Bin Buttons/Hyperlinks

5.1.1 Mobile app

5.1.1.1 Staff/Admin Login

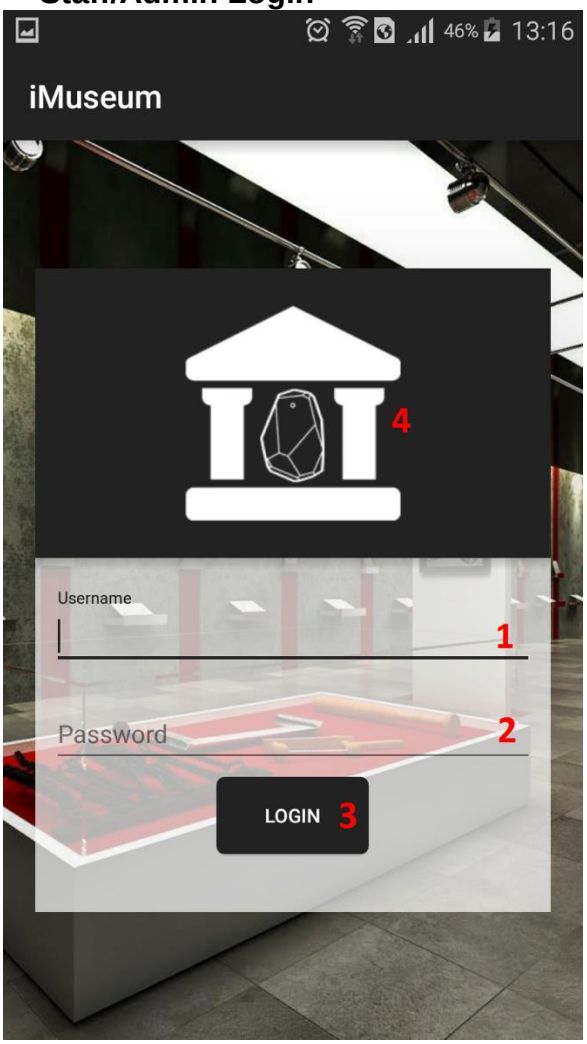


Figure 66. Staff login

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Username	Fill Username	No	Yes	EditText	String
2	Password	Fill Password	No	Yes	EditText	String
4	Logo	Logo	Yes	No	ImageView	N/A

Table 103. <Mobile app> Login Fields

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
3	Login	Login into staff app	Validation required fields before summit	Transfer to staff app

Table 104. <Mobile app> Login Buttons/Hyperlinks

5.1.1.2 Admin

5.1.1.2.1 Map beacon

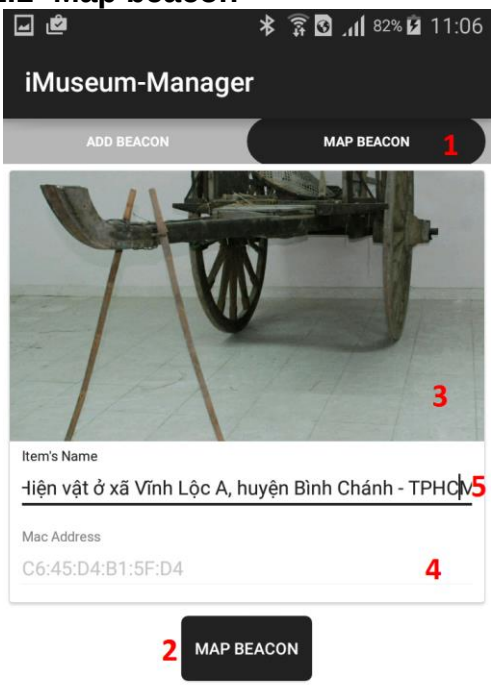


Figure 67. map beacon

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
3	itemImage	Image of item	Yes	No	ImageView	N/A
4	macAddress	Mac Address of beacon	Yes	No	TextView	String
5	itemName	Name of item	No	Yes	EditText	String

Table 105. **<Mobile app> Map Beacon Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	Manage beacon menu	All functions in managing beacon	N/A	Show function tabs
2	mapbeacon	Send map beacon command	N/A	Summit map beacon

Table 106. **<Mobile app> Map Beacon Buttons/Hyperlinks**

5.1.1.3 Staff

5.1.1.3.1 Track item

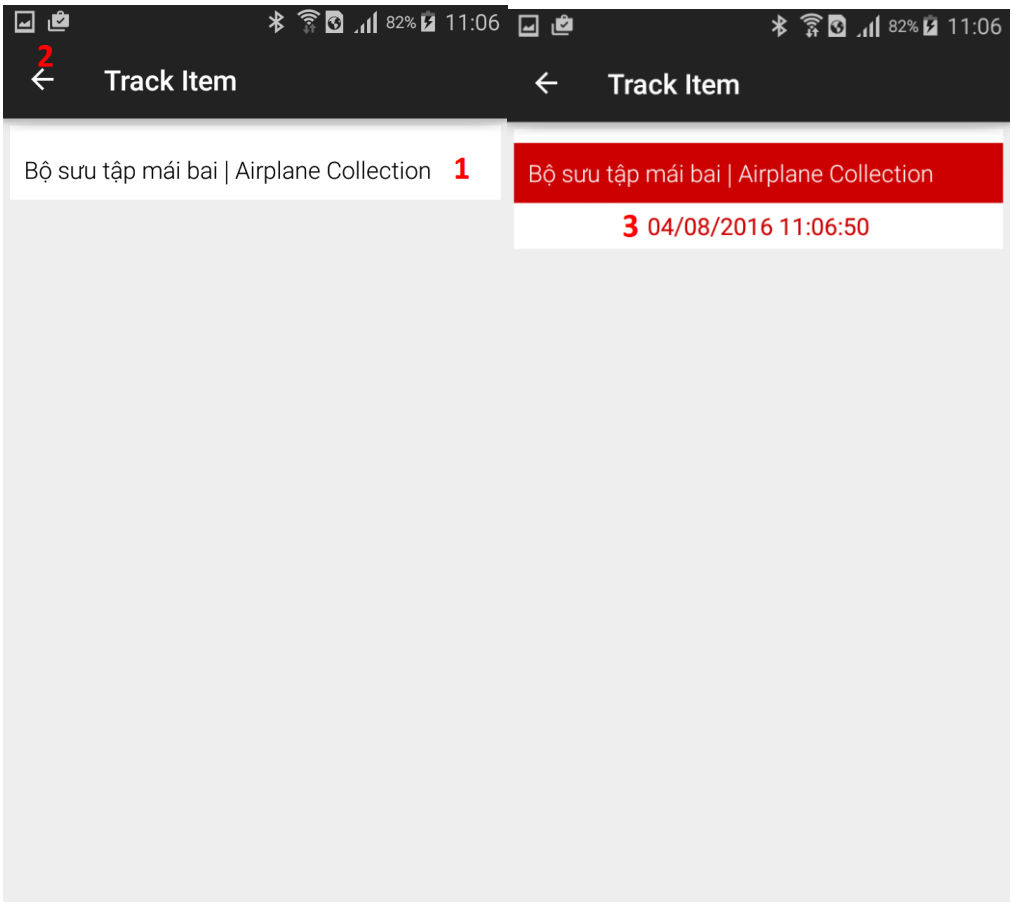


Figure 68. Track Item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
3	timeAlert	Time of alert	Yes	No	TextView	String

Table 107. **<Mobile app> Track Item Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	locationCheck	Show location of tracked beacon	N/A	Show location of tracked beacon
2	back	Back previous activity	N/A	Back to login activity

Table 108. **<Mobile app> Track Item Buttons/Hyperlinks**

5.1.1.4 Visitor

5.1.1.4.1 Navigation View

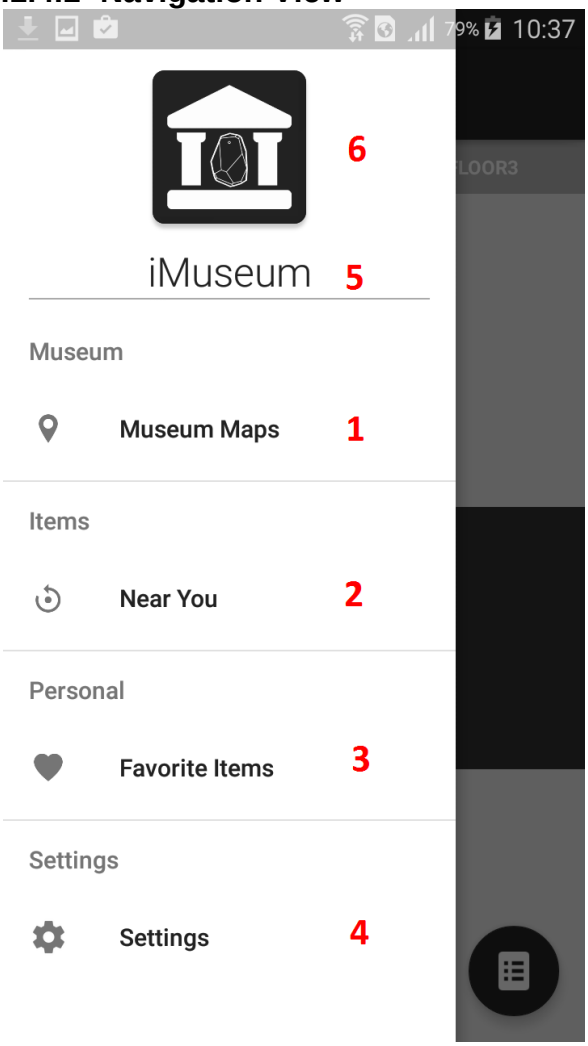


Figure 69. Navigation view

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
5	Logo	Logo	Yes	No	ImageView	N/A
6	Name	Name app	Yes	No	TextView	String

Table 109. **<Mobile app> Navigation view Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	getMuseumMaps	Load map of museum	N/A	Transfer to map activity
2	getNearByBeacon	Load nearby historic items	N/A	Transfer to near item activity
3	getFavoriteItems	Load favorite items	N/A	Transfer to favorite item activity

4	Settings	Load system	N/A	Transfer to setting activity

Table 110. **<Mobile app> Navigation view Buttons/Hyperlinks**

5.1.1.4.2 Museum map

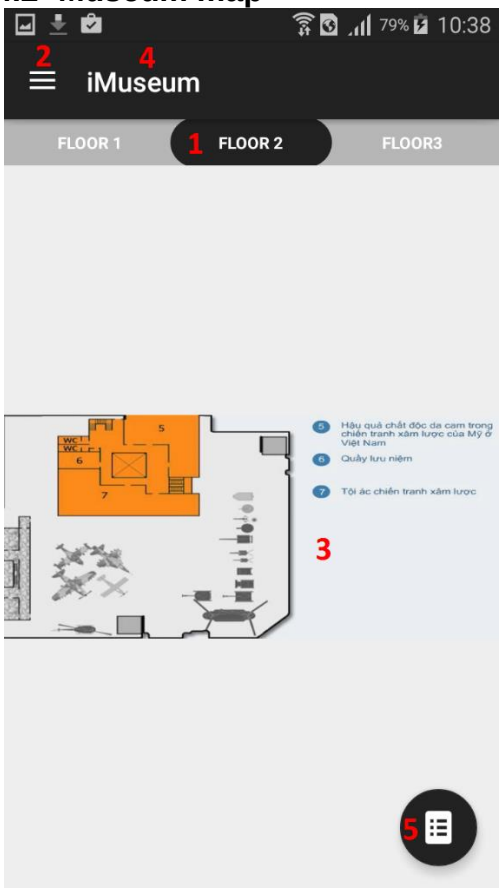


Figure 70. Museum map

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
3	floorImage	Image of floor	Yes	No	ImageView	N/A
4	nameActivity	Name of activity	Yes	No	TextView	String

Table 111. **<Mobile app> Museum Map Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	getAllFloor	All floor map of museum	N/A	Switch other floor tabs

2	showNavigation	Show navigation view	N/A	Show navigation view
5	loadItemInFloor	Show list item in floor popup	N/A	Show list item in floor popup

Table 112. **<Mobile app> Museum Map Buttons/Hyperlinks**

5.1.1.4.3 List Item

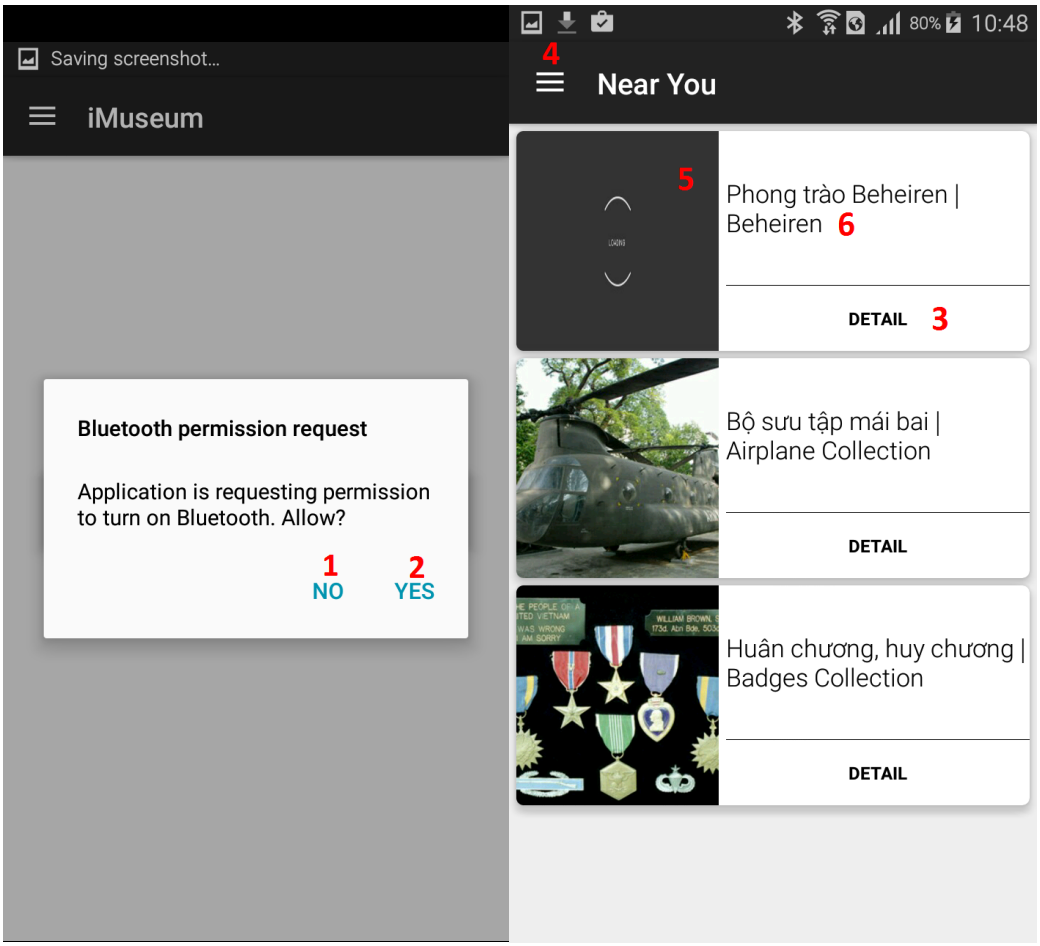


Figure 71. List item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
5	itemImage	Image of item	Yes	No	ImageView	N/A
6	itemName	Name of item	Yes	No	TextView	String

Table 113. **<Mobile app> List Item Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	No	Close turn on Bluetooth dialog	N/A	Close turn on Bluetooth dialog

2	Yes	Turn on Bluetooth	N/A	Turn on Bluetooth
3	showHistoricItem	Contain historic item image and name	N/A	Redirect to historic item details view
4	showNavigation	Show navigation view	N/A	Show navigation view

Table 114. **<Mobile app> List Item Buttons/Hyperlinks**

5.1.1.4.4 Item Details

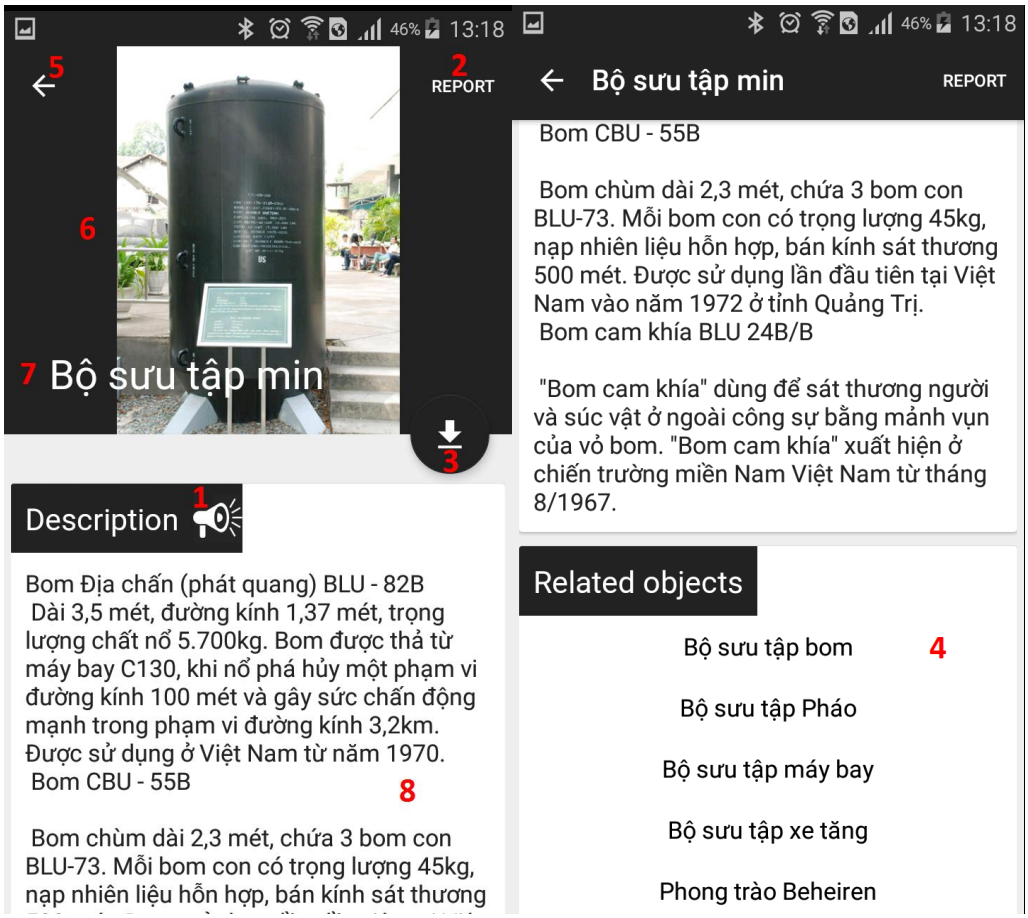


Figure 72. Item details

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
6	imageItem	Image of item	Yes	No	ImageView	N/A
7	nameItem	Name of item	Yes	No	TextView	String
8	Description	Description of item	Yes	No	TextView	String

Table 115. **<Mobile app> Item Details Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	speakItemInformation	Send speak description command	N/A	Speak description
2	reportItemInformation	Show report popup	N/A	Show report popup
3	addFavoriteItem	Send add favorite command	N/A	Summit add favorite command
4	relatedItems	Show related item popup	N/A	Show related item popup
5	back	Back previous activity	N/A	Back to list item activity

Table 116. **<Mobile app> Item Details Buttons/Hyperlinks**

5.1.1.4.5 Report Item

Figure 73. Report item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Message	Message report	No	Yes	Textbox	String
4	nameItem	Name of item	Yes	No	TextView	String

Table 117. **<Mobile app> Report Item Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
2	send	Send report item command	N/A	Summit report item command
3	cancel	Close report item popup	N/A	Close report item popup

Table 118. **<Mobile app>Report Item Buttons/Hyperlinks**

5.1.1.4.6 Add favorite item

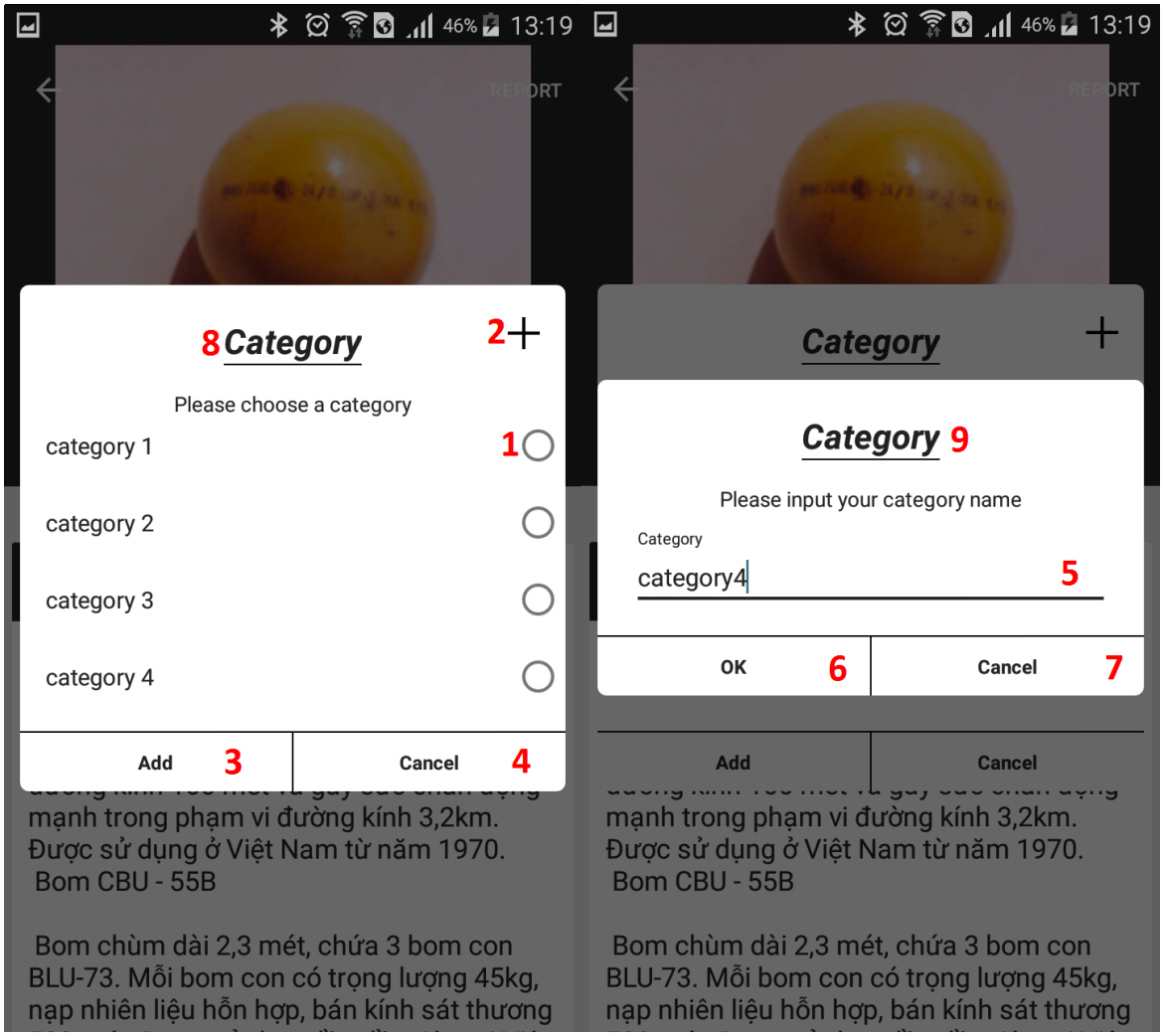


Figure 74. Add favorite item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Category	Choose category	No	Yes	Combo box	N/A
5	Category Name	Name of category	No	Yes	Textbox	String
8	Popup Name	Name of popup category	Yes	No	TextView	String

9	Popup Create Name	Name of popup create category	Yes	No	TextView	String

Table 119. **<Mobile app> Add favorite Item Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
2	createCategory	Create new category	N/A	Show create new category popup
3	addFavoriteItem	Send add favorite item command	Validation required fields before summit	Summit add favorite item command
4	cancelCategory	Close category popup	N/A	Close category popup
6	confirmCategory	Send create new category command	Validation required fields before summit	Summit create new category command
7	cancelCreate	Close create new category popup	N/A	Close create new category popup

Table 120. <Mobile app> Add favorite Item Buttons/Hyperlinks

5.1.1.4.7 Find related item

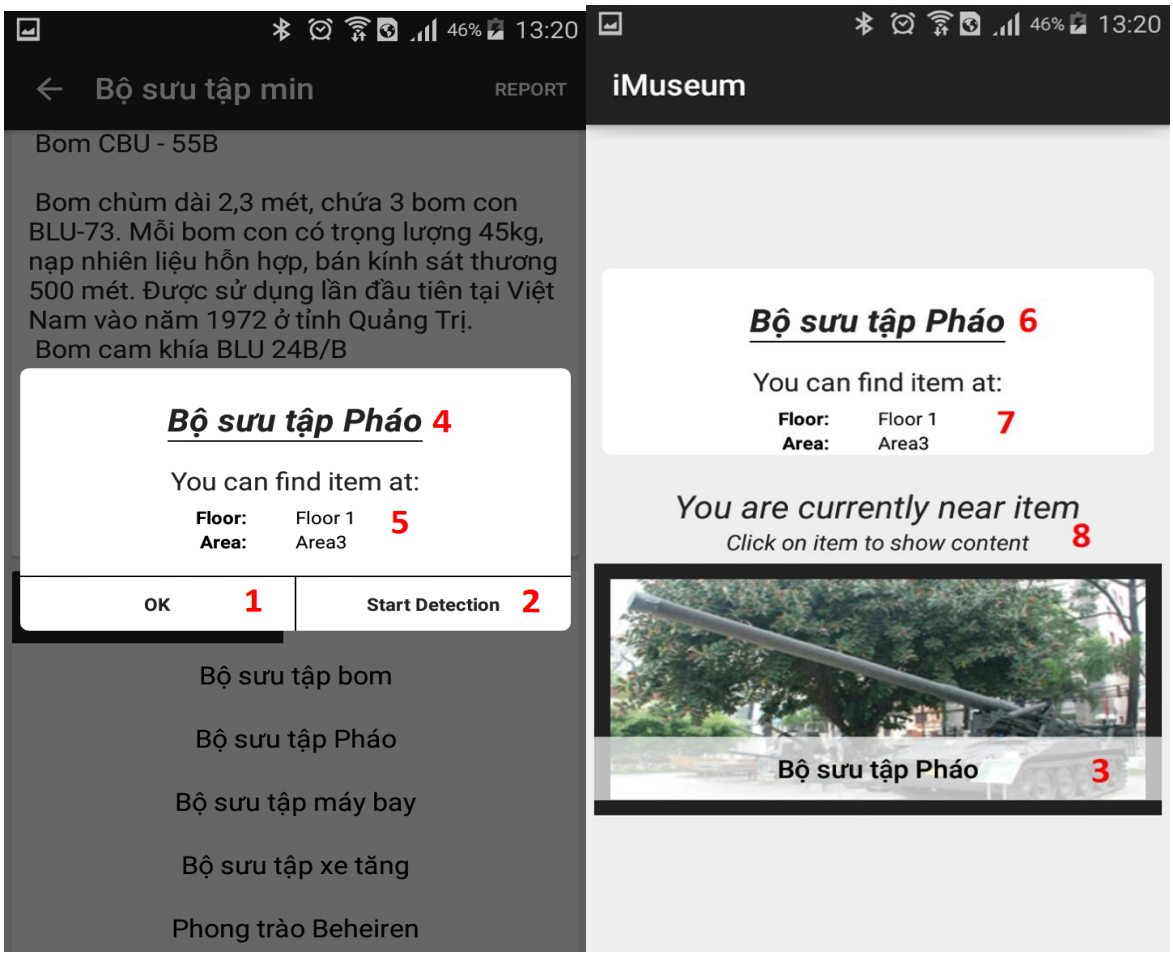


Figure 75. Find related item

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
4	relateName	Name of related item	Yes	No	TextView	String
5	relateLocation	Location of related item	Yes	No	TextView	String
6	relateName	Name of related item	Yes	No	TextView	String
7	relateLocation	Location of related item	Yes	No	TextView	String
8	nearItem	Show you near related item	Yes	No	TextView	String

Table 121. **<Mobile app> Find Related Item Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
1	ok	Close related item popup	N/A	Close related item popup
2	startDetection	Show detect related item activity	N/A	Transfer to detect related item activity
3	Historic item	Show related item detail	N/A	Transfer to item details activity

Table 122. **<Mobile app>Find Related Item Buttons/Hyperlinks**

5.1.1.4.8 Favorite items

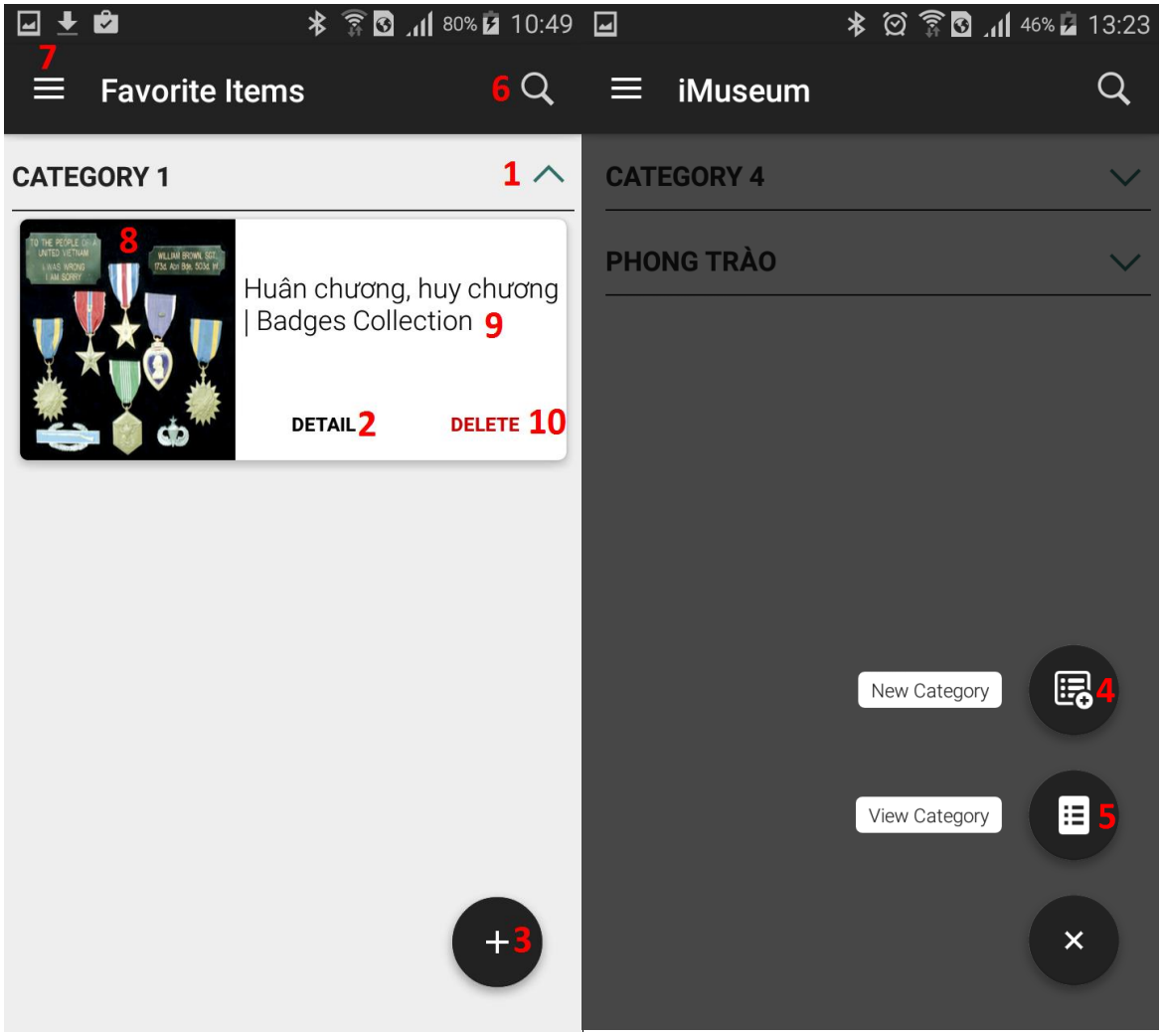


Figure 76. Favorite items

Fields

No	Field Names	Description	Read Only	Mandatory	Control Type	Data Type
1	Category name	category	No	No	Dropdown list	N/A
8	itemImage	Image of item	Yes	No	ImageView	N/A
9	itemName	Name of item	Yes	No	TextView	String

Table 123. **<Mobile app> Favorite Items Fields**

Buttons/Hyperlinks

No	Function	Description	Validation	Outcome
2	detail	Show favorite item detail	N/A	Transfer to item details activity

3	manageCategory	Show button to manage category	N/A	Show button to manage category
4	newCategory	Show create new category popup	N/A	Show create new category popup
5	viewCategory	Show manage category activity	N/A	Show manage category activity
6	search	Search item	N/A	Show item match
7	navigation	Show navigation view	N/A	Show navigation view
10	delete	Delete favorite item	N/A	Show delete favorite item popup

Table 124. **<Mobile app> Favorite Items Buttons/Hyperlinks**

6. Database Design

6.1 Entity Relationship Diagram (ERD)

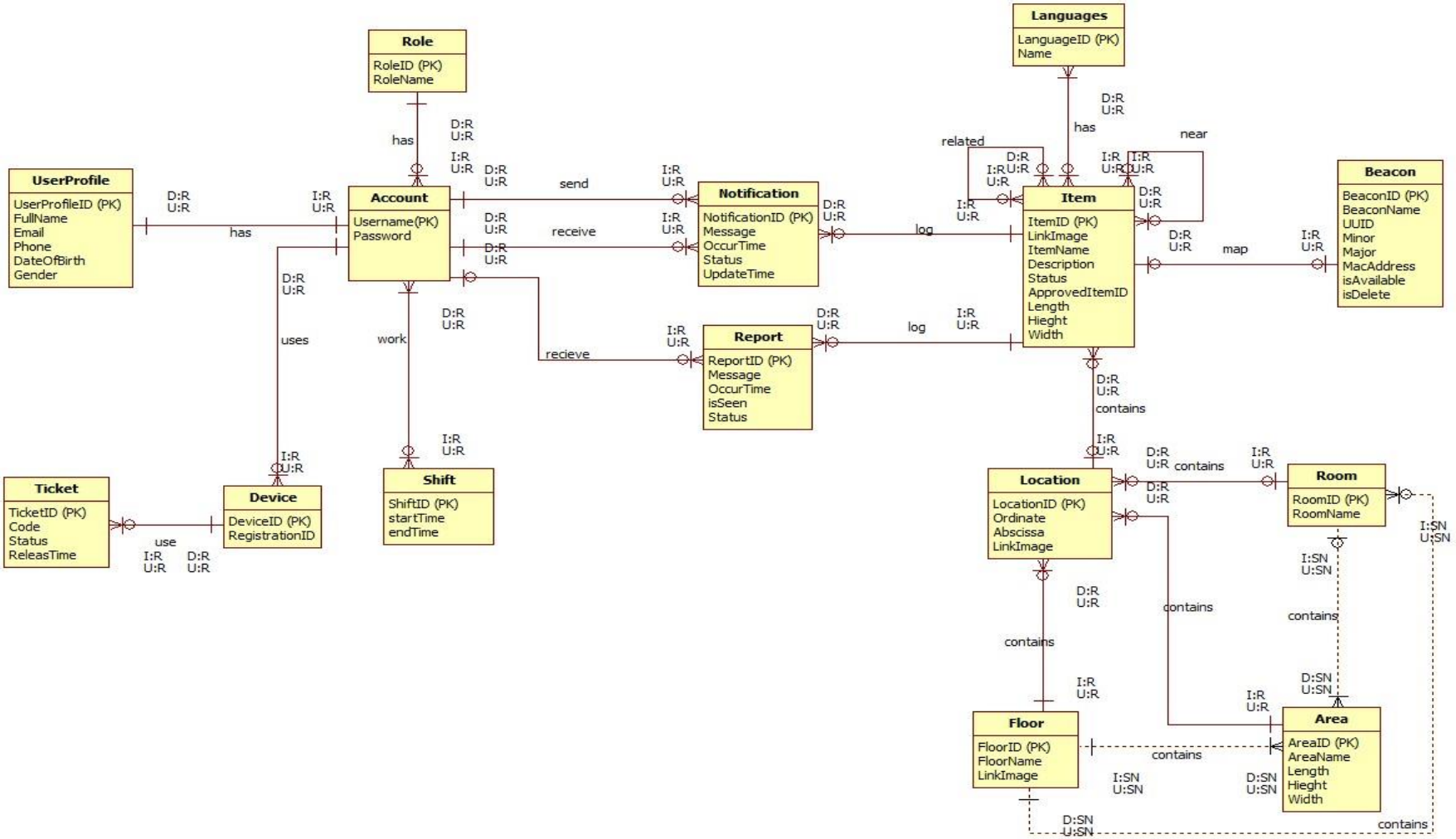


Figure 77. ERD Diagram

6.2 Entity Relationship Data Dictionary

Entity Name	Mapping column with Conceptual diagram	Description
Role	N/A	Contain the role information. Has 3 types: Admin, Staff and Expert.
Account	N/A	Contain the account information.
UserProfile	N/A	Contain the user profile information.
Shift	Shift	Contain the information of Staff's shift.
Device	Device	Contain the information of the devices used by Staff.
Item	Item	Contain the item information.
Language	Language	Contain the language information.
Notification	Notification	Contain the log of item movements.
Report	Report	Contain the visitor report information.
Beacon	Beacon	Contain the Beacon information
Location	N/A	Contain the location information, which includes floor, room and area.
Floor	Floor	Contain the floor information.
Room	Room	Contain the room information.
Area	Area	Contain the area information.
Ticket	Ticket	Contain the ticket information.

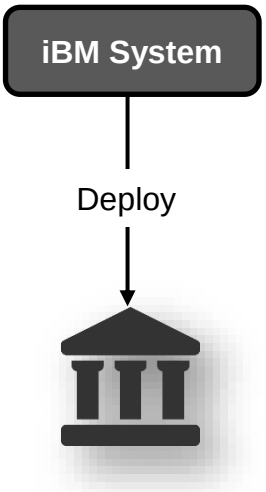
Table 125. ERD Dictionary

7. Strategy for future expand plan

iBM system is currently an application to deploy to a single museum. Our vision for the future plan is to deploy iBM system to multiple museum not only the museums in the country but also foreign museums

All of the museum will be connected with each other. With this connection, visitors not only can know the information of historic item in the current museum, which visitors are discovering, but also the information of the other historic item in the different museum.

Base on the original requirement, iBM System will support manage historic item and beacon in a specific museum. The iBM System will be deploy to a museum which provide a single service.



Single Service Deployment Model

Figure 78. Single Service Deployment Model

As a part of the implement process, to obtain the scalability of the system, this is what we designed the components:

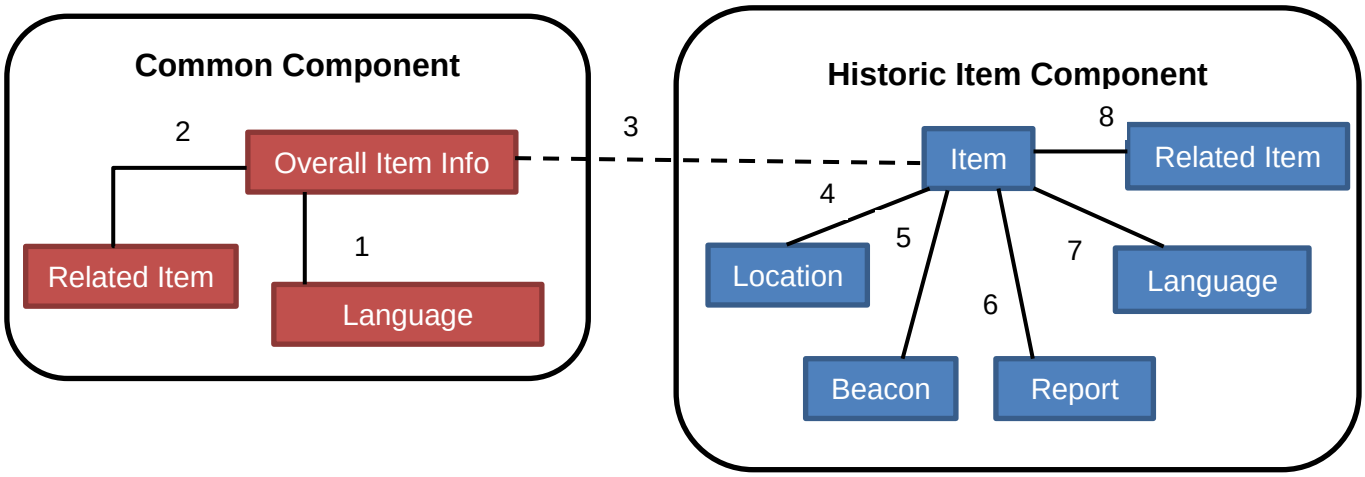
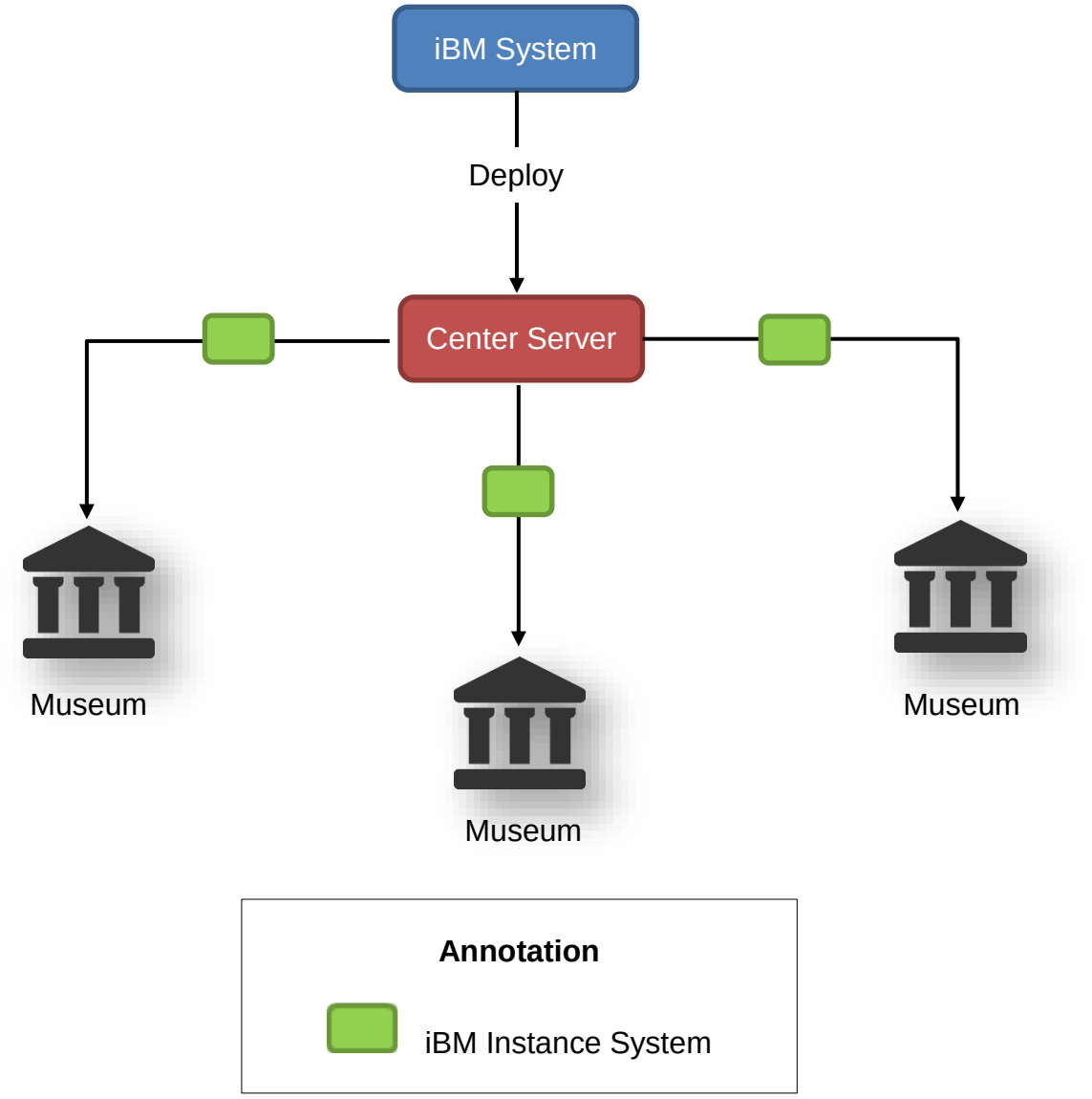


Figure 79. Entities group by component

Next, we will show the process to expand this system to the new system that supports multiple service.



Distributed Multiple Service

Figure 80. Distributed Multiple Service Deployment Model

This is the final model of iBM System form our vision. In this model, we deploy iBM System in a Center Server which operates multiple insurance of museums.

All of the overall information of historic item will be store, and visitor can get the overall information of historic item in the different museum with just only one click in mobile application. With this model, visitor is not only discovering one museum at the current time, but also the different museum in current country or even though in foreign country.

And this is the expected deployment component we estimate:

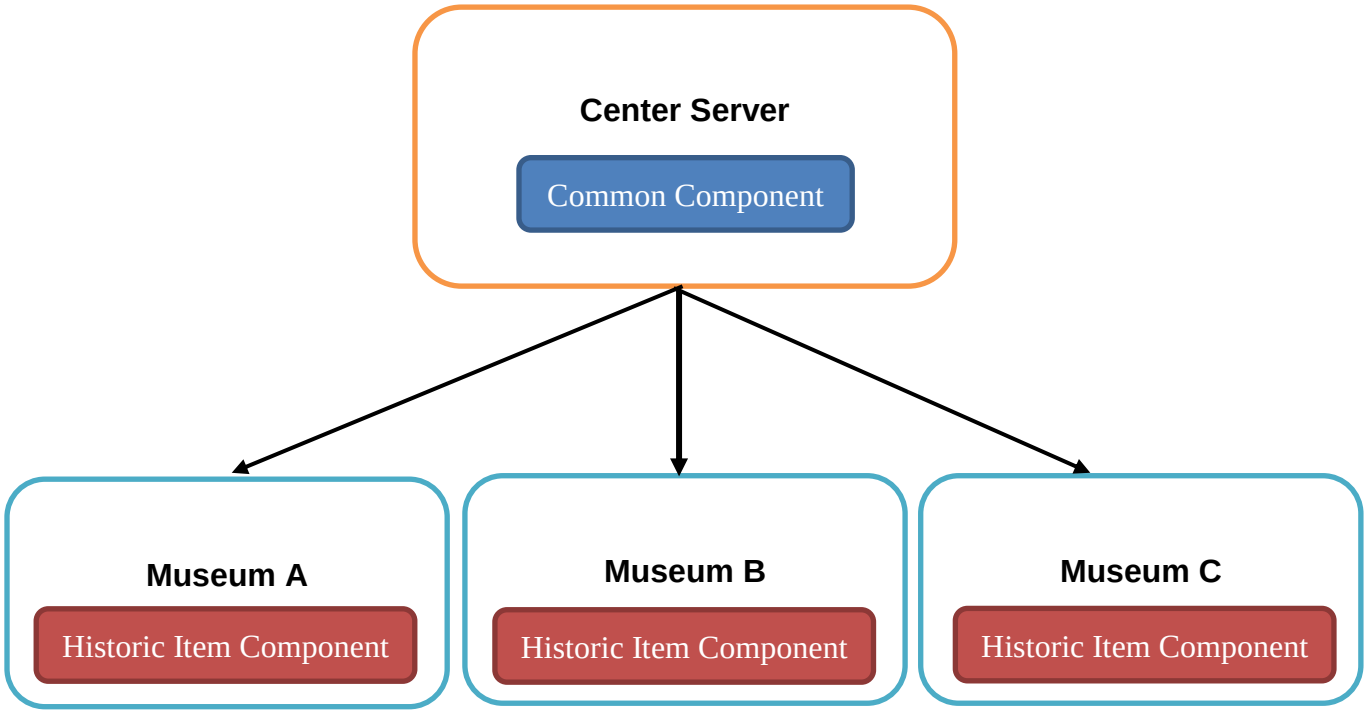


Figure 81. Components group by container (Distributed Multiple Service)

The Distributed Multiple Service plan is still need to be investigate a lot to clarify the components and relationships between entities as well as the architecture design to maintain the best performance and less impact to the current system, we also need to identify the arise problems when changing to distributed model.

8. Algorithms

8.1 Crawl Data from website

8.1.1 Definition

Automatically crawl historic items information from the website of a museum.

8.1.2 Define Problem

If Admin have to add historic item information one by one, it will take a huge time and effort.
On the other hand, historic items information is usually stored in the website of the museum, we can reuse those historic items information without adding them one by one.

8.1.3 Solution

Using the Web Application, Admin can setup and add museum website. The added website and its configuration will be stored in an XML file; whose format is like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configuration-sites>
    <site>
        <category-link></category-link>   1
        <category-xpath></category-xpath> 2
        <main-xpath></main-xpath>  3
        <crawl-type></crawl-type>  4
        <xpaths>
            <name></name>  5
            <image></image>  6
            <description></description>  7
        </xpaths>
    </site>
</configuration-sites>
```

Number	Element name	Meaning
1	category-link	The URL of the museum website (can be blank if crawl-type=1)
2	category-xpath	XPath of *site's pagination* (can be blank if crawl-type=1)
3	main-xpath	XPath of *all articles container* (can be blank if crawl-type=1)
4	crawl-type	The type of the site • crawl-type=1: crawl single item • crawl-type=2: crawl all item in site
5	name	XPath of *Item Name* (required)
6	image	XPath of *Item Image* (required)
7	description	XPath of *Item Description* (required)

Admin can also configure the time and day for the crawler to run.

When hit the configured time, the crawler will follow these steps to get historic item data from stored websites:

- Crawler will read the XML configuration file to get all website URL and their elements.
- For each website, crawler will go to every item article base on the XPath of *page's pagination* and *all articles container*.
- For each item article, crawler will get Item's name, image and Description base on XPath.
- If the item is not existed in system database, the item will be created as "New" item. Otherwise, the item will be created as "Update" if its description or images are different from the existed item. Crawled item will be save with "Pending" status.

After crawled, Expert need to approve pending items before using them.

8.1.4 Flow chart

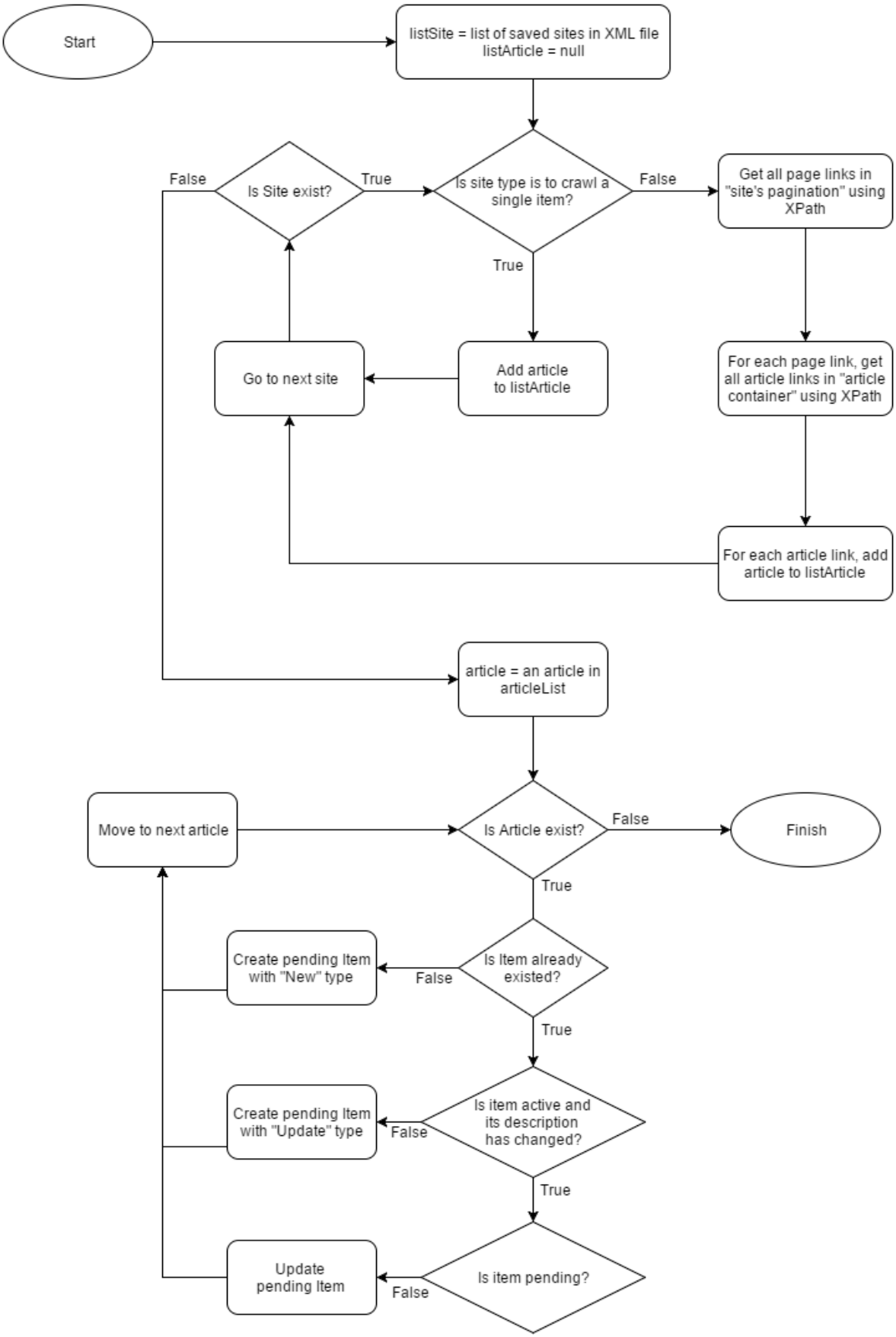


Figure 82. Crawl Item Flowchart

8.1.5 Complexity

The complexity of this algorithm is: $\boldsymbol{O}(\boldsymbol{n^2})$

8.2 Search data contains Vietnamese

8.2.1 Definition

Ignoring case sensitive and Vietnamese accent marks while searching data.
Example: “suu tap” should be equivalent to “Sưu tập”.

8.2.2 Define Problem

Vietnamese is a complicated language, which each term usually contains at least one accent mark.
In most cases, a specific software is required in order to type Vietnamese properly. Even with the software installed, it will take user more time and effort to search for a historic item with accent mark.

8.2.3 Solution

We simply replace any characters contain accent mark with a normal Latin character.

- Firstly, string will be converted to lower case.
- Then, characters contain accent mark will be replaced:
  - à, á, ạ, ả, ã, â, ầ, ấ, ậ, ẩ, ẫ, ă, ằ, ắ, ặ, ẳ, ẵ → a
  - è, é, ẹ, ẻ, ẽ, ê, ề, ế, ệ, ể, ễ → e
  - ì, í, ị, ỉ, ĩ → i
  - ò, ó, ọ, ỏ, õ, ô, ồ, ố, ộ, ổ, ỗ, ơ, ờ, ớ, ợ, ở, ỡ → o
  - ù, ú, ụ, ủ, ũ, ư, ừ, ứ, ự, ử, ữ → u
  - ỳ, ý, ỵ, ỷ, ỹ → y
  - đ → d
- With strings have its accent mark removed, we just call search function normally.

Example: The input word is “Bộ sưu tập”. The word will go through the above process to have its accent mark removed:

- Bộ sưu tập → bộ sưu tập
- bộ sưu tập → bộ sưu tap → bo sưu tap → bo suu tap

8.2.4 Complexity

The complexity of this algorithm is: $\boldsymbol{O}(\boldsymbol{n})$

8.3Get keywords from documents

8.3.1 Definition

Using a Vietnamese dictionary, this algorithm will get all compound words, proper nouns and event years within a document.
Example:

- The input document: “Cuộc kháng chiến chống Mỹ kết thúc vào năm 1975.”
- The output: “cuộc kháng chiến”, “kháng chiến”, “Mỹ”, “kết thúc”, “1975”.

8.3.2 Define Problem

In order to calculate similarity between items, we need to get all keywords within item's name and description.
Because historic articles usually content proper nouns and event years, this algorithm can cover those cases.

8.3.3 Solution

For each document, we follow these steps to get its keywords.

- We will separate punctuation mark to word.
  Example: "Năm 1975, cuộc chiến kết thúc." → "Năm 1975 , cuộc chiến kết thúc ."
- We will check every word of the document
  - Check and get event year: Using regular expression, we can check if word is format as DD/MM/YYYY, MM/YYYY or YYYY. If yes, we will split and take only the year part of the word.
    Example: 22/02/1994 → 1994, 8/1975→1975.
  - Check for proper nouns:
    - We have a String variable properNoun, empty by default.
    - If word is capital and the previous word is not a punctuation: properNoun += word.
    - If word is not capital, and properNoun is not empty. properNoun will be added to keywords list, and reset to an empty string.
    - Example: We are checking term "Nhật" in sentence "Người dân Nhật Bản kiên cường"
      - word = Nhật, is a capital ~> properNoun = Nhật
      - word = Bản, is a capital ~> properNoun = Nhật Bản
      - word = kiên, is not a capital ~> properNoun = Nhật Bản is not empty ~> "Nhật Bản" is added, properNoun reset to empty.
      - …
  - Check for combound words:
    - We have a String variable term.
      By default, term = the previous word + the current word.
    - We will check if term is a valid Vietnamese 2-word term, by comparing it to a Vietnamese dictionary. If yes, term will be added to keyword list.
    - term += the next word, then we check if term is a valid Vietnamese 3-word term. If yes, term will be added to keyword list.
    - term += the next word, then we check if term is a valid Vietnamese 4-word term. If yes, term will be added to keyword list.
    - Then, move on the next word.
    - Example: We are checking the word "tự" in sentence "vũ khí bán tự động của Mỹ"
      - term = "bán tự" → skip
      - term = "bán tự động" → add
      - term = "bán tự động của" → skip (Move on next word: "động")
      - term = "tự động" → add

- term = “tự động của” → skip
- term = “tự động của Mỹ” → skip
- …

8.3.1 Flow chart

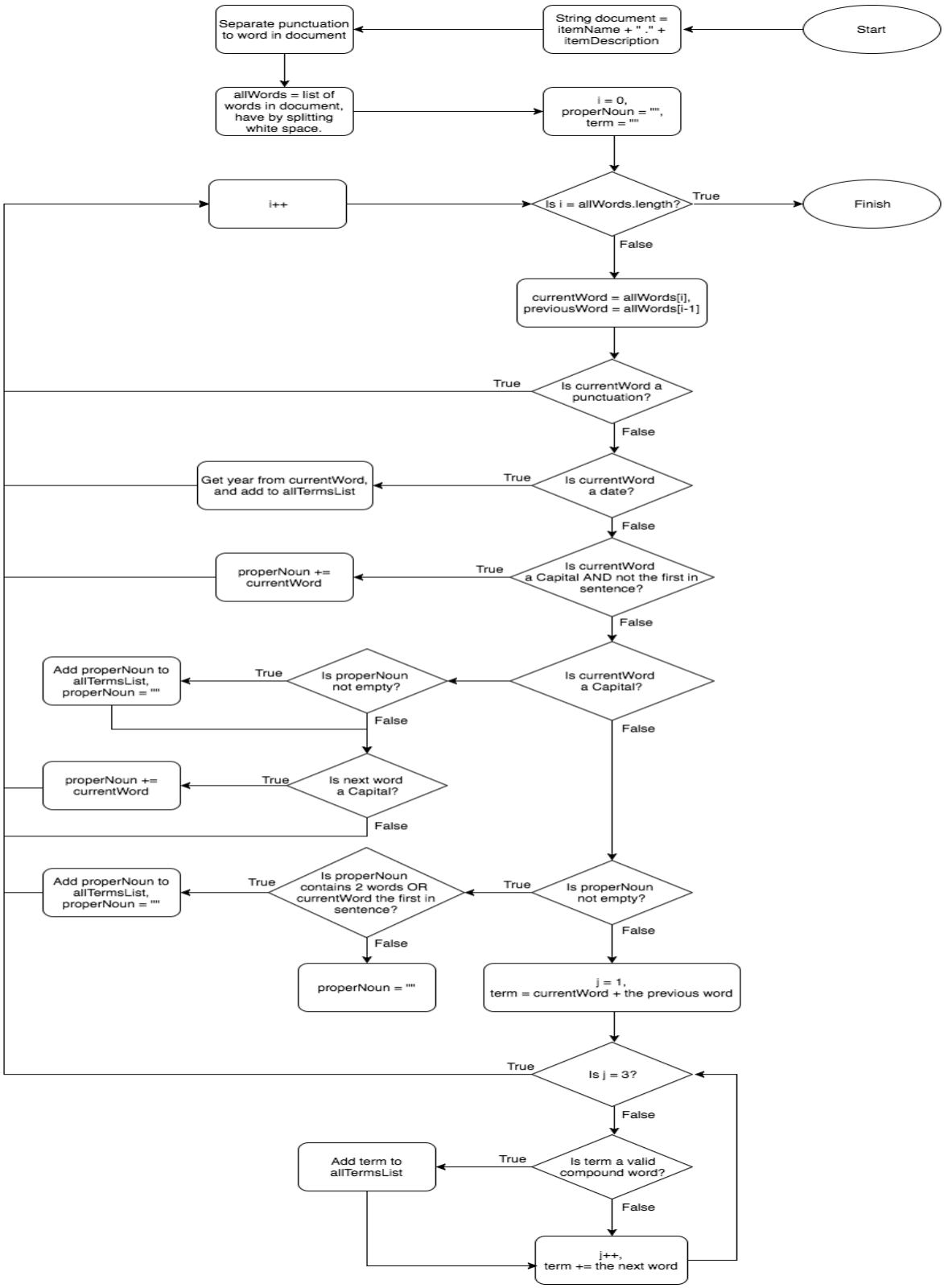


Figure 83. Get keywords from document Flowchart

8.3.1 Complexity

The complexity of this algorithm is: $\boldsymbol{O}(\boldsymbol{n^2})$

8.4Calculate similarity between Items

8.4.1 Definition

Calculating the similarity between items base on their name and description. This algorithm is used to check whether a historic item is related to another.

8.4.2 Define Problem

After viewing a historic item information, visitor may want to view historic item that relate to it. Therefore, the related items should be defined.
However, in a large amount of information, which are updated frequently, finding which items is related to another, and then choosing five most related one to show to the visitors is impossible for a normal person.

8.4.3 Solution

Every day at a configured time, the scheduler will check if there are any changes during the day. If yes, the system will re-calculate the similarity between items. The calculation will follow these steps:

- Step 1: Get all keywords from documents. (Using algorithm 8.3)

- Step 2: Use TF-IDF methods to calculate the weight of these keywords on each document.
  - *TF-IDF* is a numerical statistic that is intended to reflect how important a word is to a document. The smaller TF-IDF of a word, the less important that word is in document.
  - TF (Term Frequency): The number of time a keyword occurs in *one document.*

$$tf(t) = \frac{number\ of\ times\ t\ occur\ in\ document}{total\ number\ of\ term\ in\ document}$$

  - IDF (Inverse Term Frequency): Measure how much information a word provides.

$$idf(t) = 1 + \log_e \frac{total\ number\ of\ documents}{number\ of\ documents\ contain\ term\ t}$$

  - Finally, $TFIDF(t) = tf(t)\ .\ idf(t)$

  When done, an n-dimensions Vector is created for each document containing TF-IDF of all keywords, where n is the number of keywords.

- Step 3: Calculate the Cosine Similarity of these Vectors
  - *Cosine similarity* is a measure of similarity between two vectors using the cosine of the angle between them. Two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a

similarity of 0, and two vectors diametrically opposed have a similarity of -1.

- In our case, Cosine Similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].
- Cosine Similarity formula:

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} . \vec{b}}{|\vec{a}| . |\vec{b}|}$$

- The higher the calculated number, the more similar two historic items are

After calculation progress finishes, the similarity of 2 historic items will be saved to system database. Base on the calculated similarity, system will suggest the Expert when setting Related item.

Example:

Item1: Bộ sưu tập vũ khí của lính Mỹ.
Item2: Bộ sưu tập xe tăng. Xe tăng là một vũ khí nguy hiểm.
Item3: Huân chương của lính Mỹ.

- Step 1: We get all the keywords: “sưu tập”, “vũ khí”, “Mỹ”, “xe tăng”, “nguy hiểm”, “huân chương”

- Step 2: s
  - Calculate TF:

	sưu tập	vũ khí	Mỹ	xe tăng	nguy hiểm	huân chương
Item1	0.333333	0.333333	0.333333	0	0	0
Item2	0.25	0.25	0	0.5	0.25	0
Item3	0	0	0.5	0	0	0.5

  - Calculate IDF:

	sưu tập	vũ khí	Mỹ	xe tăng	nguy hiểm	huân chương
IDF	1.405465	1.405465	1.405465	2.09861	2.09861	2.09861

  - Calculate TF-IDF:

	sưu tập	vũ khí	Mỹ	xe tăng	nguy hiểm	huân chương
Item1	0.468488	0.468488	0.468488	0	0	0
Item2	0.351366	0.351366	0	1.049305	0.5246525	0
Item3	0	0	0.702733	0	0	1.049305

- Step 3:

$$Cos(\overrightarrow{item1}, \overrightarrow{item2}) = \frac{0.47 * 0.35 + 0.47 * 0.35 + 0 + 0 + 0 + 0}{\sqrt{0.47^2 + 0.47^2 + 0.47^2}.\sqrt{0.35^2 + 0.35^2 + 1.05^2 + 0.5^2}}$$

$$= 0.3197$$

$$Cos(\overrightarrow{item1}, \overrightarrow{item3}) = \frac{0 + 0 + 0.47 * 0.7 + 0 + 0 + 0}{\sqrt{0.47^2 + 0.47^2 + 0.47^2}.\sqrt{0.7^2 + 1.05^2}}$$

$$= 0.32026$$

$$Cos(\overrightarrow{item2}, \overrightarrow{item3}) = \frac{0 + 0 + 0 + 0 + 0 + 0}{\sqrt{0.35^2 + 0.35^2 + 1.05^2 + 0.5^2}.\sqrt{0.7^2 + 1.05^2}}$$

$$= 0$$

Therefore, the similarity is:

- Item1 and item2 is 0.3197
- Item1 and item3 is 0.32026
- Item2 and item3 is 0.

Which means Item1 and Item3 is the most similar, and Item2 and item3 have nothing in common.

8.4.4 Flow chart

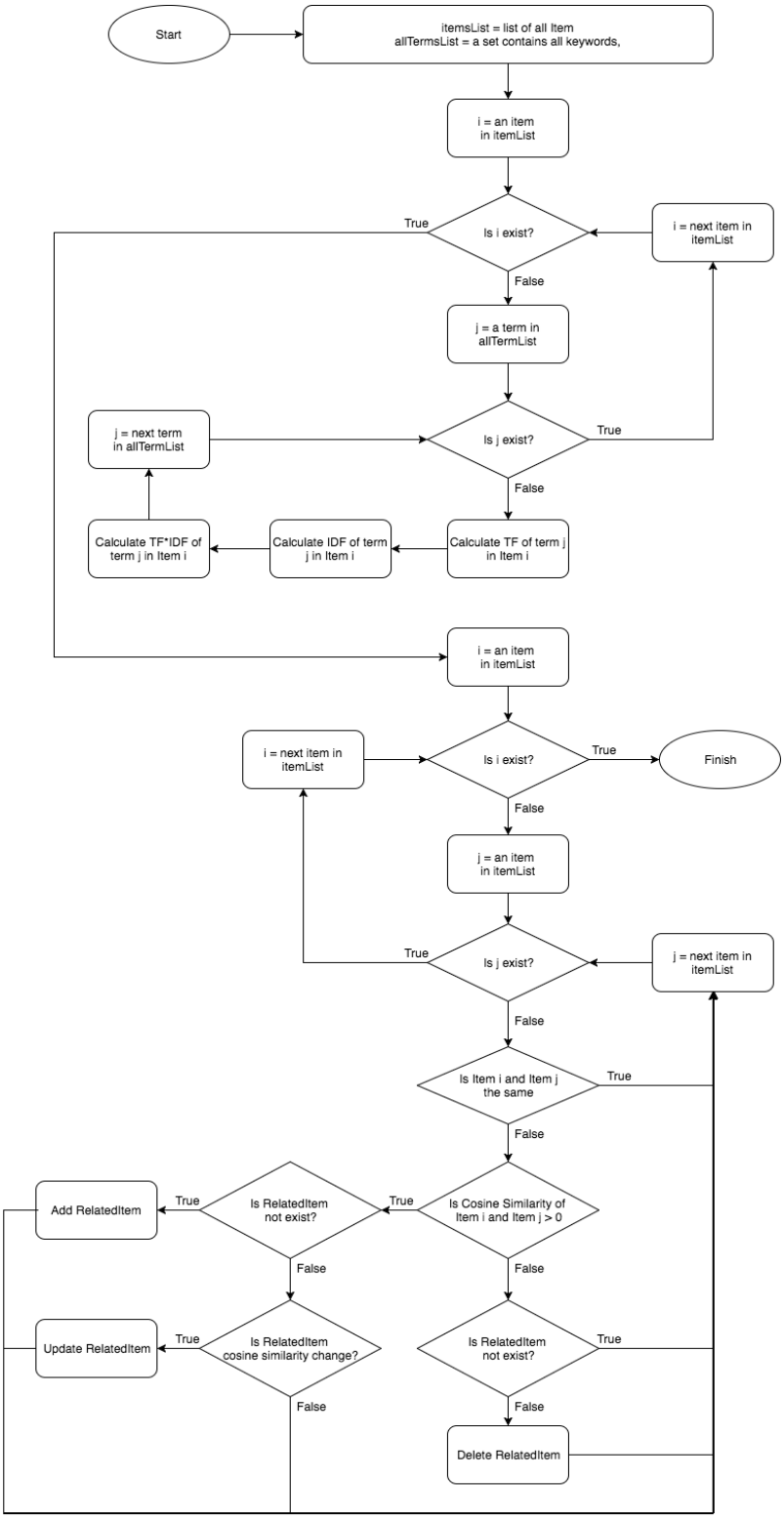


Figure 84. Calculate Item Similarity Flowchart

8.4.5 Complexity

The complexity of this algorithm is: $\boldsymbol{O(n^2)}$

8.5Check if item is fit the area

8.5.1 Definition

Check if the historic item can be placed in an existed area.
This algorithm is used to suggest available areas to Admin.

8.5.2 Define Problem

Placing a historic item to an area have follow conditions:

- Each area has different width, length height and so does each historic item.
- An area may have already contained some historic items.
- The distance between each historic item must be bigger than 2 meters.

8.5.3 Solution

We have:

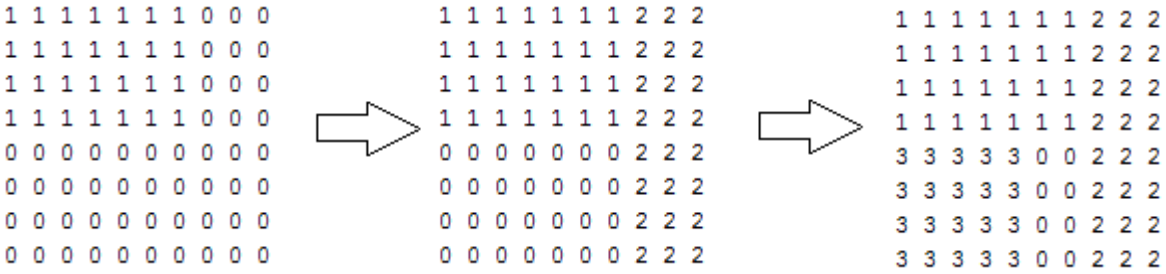
- The historic item that we need to check.
- The area that we need to check.
- The list of the historic items that already be placed in the area.

We'll try to arrange all items in the area to check if it is possible:

- Firstly, if the height of one historic item is bigger than the height of the area, item is not fit.
- Because the minimum distance between each historic item is 2 meters, items width and length will be expanded by 1 meters before calculation.
- The area will be converted to a 2-dimension array; each cell default value is 0.

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

- We will find the cell where its value is 0, and try to place the item there with different rotation. After the placement, cells value will be changed.



- If success, the algorithm will return true and stop. Else, the algorithm will try another permutation of historic items order. After trying all possible permutation, the algorithm will return false.

8.5.4 Flow chart

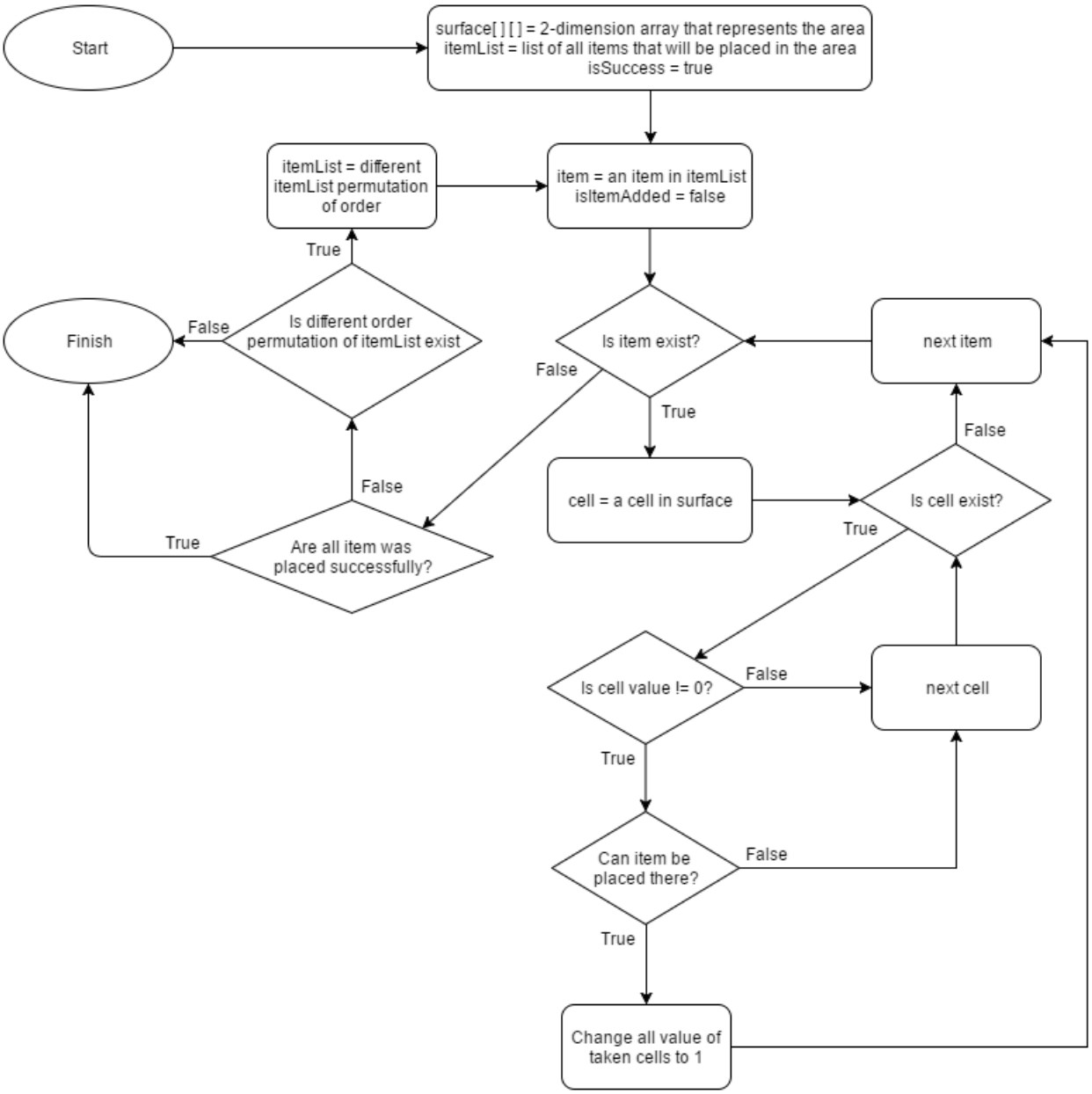


Figure 85. Check If item is fit area

8.5.5 Complexity

The complexity of this algorithm is: $\boldsymbol{O}(\boldsymbol{n^6})$

E. System Implementation & Test (SIT)

1. Introduction

1.1 System Overview

This section describes the approach and methodologies used by group to plan, organize and manage the testing of iMuseum system. It provides in the detail of 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 as much as possible cases that we can.

1.2 Test Approach

- Goal: Test core features in the iMuseum system based on the core flow.
- Method: black-box Testing.
- Size: System Component.
- Technique: Check list

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

- System testing is focused on assessing the system’s reliability. This process is concerned with finding errors that result from unanticipated interaction between components and component interface problems.

2. Database relationship diagram

2.1 Physical diagram

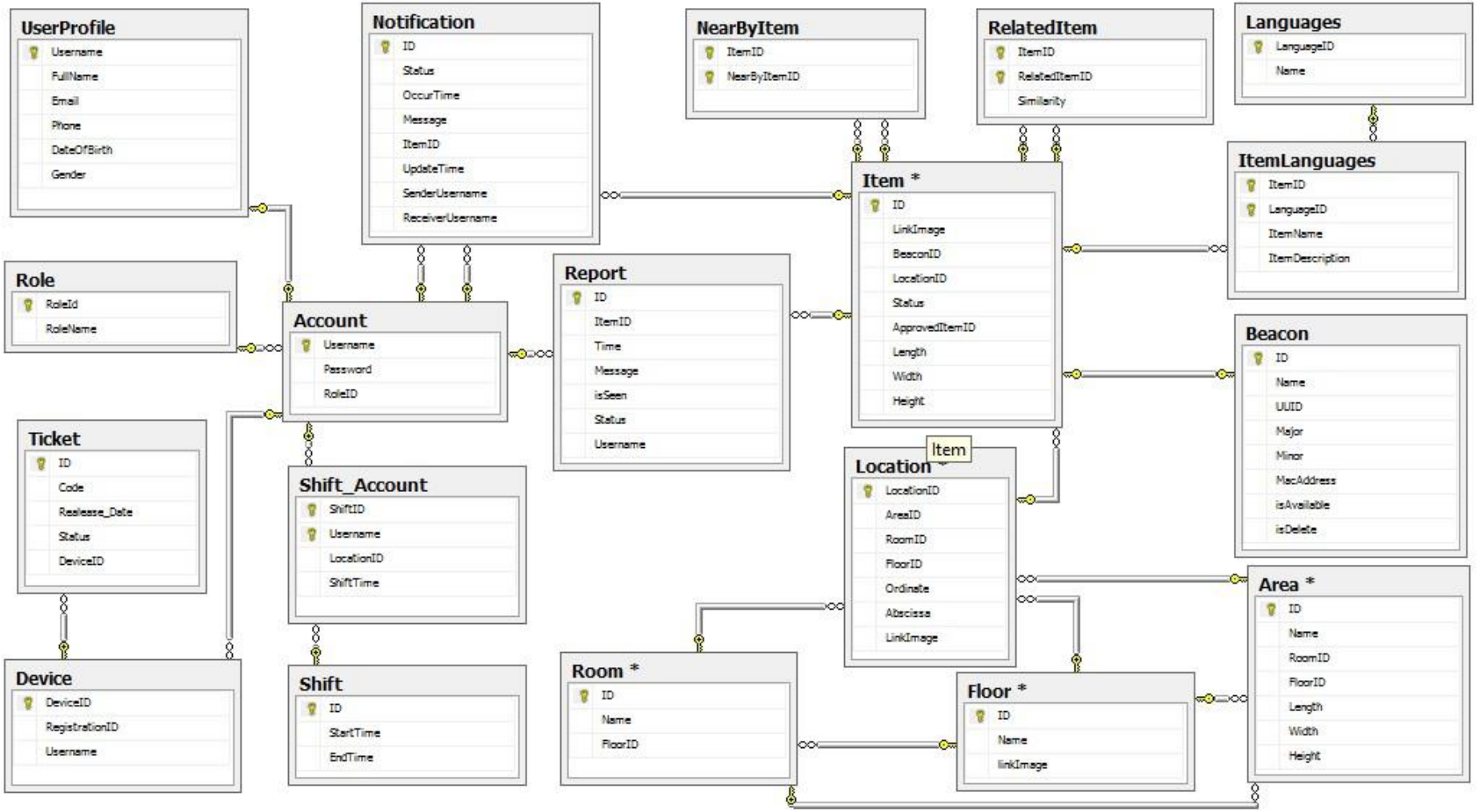


Figure 86. Physical Diagram

2.2 Data dictionary

Data dictionary: describe content of all tables			
Table Name	**Mapping with Conceptual diagram**	**Mapping with Entity diagram**	**Description**
Item	Item	Item	Contain information of historic items.
Beacon	Beacon	Beacon	Contain information of beacons, which can map with item.
Report	Report	Report	Contain information of reports which are report about historic items such as: wrong historic item name or wrong historic item description…
Notification	Notification	Notification	Contain information of notifications, which are notification of movement historic items.
Account	N/A	Account	Contain accounts of museum staffs to login to system.
Role	N/A	Role	Contain all role of museum staffs in system.
UserProfile	N/A	UserProfile	Contain information of museum staff include: full name, phone number, email, gender and date of birth.
Shift	Shift	Shift	Contain information of shift include start time, end time.
ShiftAccount	N/A	N/A	Contain information of shift include location of shift, staff who works in shift.
Ticket	Ticket	Ticket	Contain information of ticket.
Device	Device	Device	Contain information device which access to system.
Location	N/A	Location	Contain information of location.
Area	Area	Area	Contain information of museum area.
Room	Room	Room	Contain information of museum room.
Floor	Floor	Floor	Contain information of museum floor.

RelatedItem	N/A	N/A	Contain information of related historic item.
NearByItem	N/A	N/A	Contain information of nearby historic item.
Languages	Languages	Languages	Contain information of system languages.
ItemLanguages	N/A	N/A	Contain information of item include name and description that have multi languages.

Table 126. Data table dictionary

Table Name	Attributes	Description	Domain	Null
Account	Username (PK)	Username of museum staff account to login to system.	nvarchar(100)	No
	Password	Password of museum staff account to login to system.	nvarchar(100)	No
	RoleID	Identifier of role that this museum staff plays.	int	Yes
Role	RoleID	Identifier of role.	int	No
	RoleName	Name of role	nvarchar(50)	No
UserProfile	Username (PK)	Username of museum staff account to login to system.	nvarchar(100)	No
	FullName	Full name of museum staff.	nvarchar(MAX)	Yes
	Phone	Phone of museum staff.	nvarchar(15)	Yes
	DateOfBirth	Date of birth of museum staff.	bigint	Yes
	Email	Email of museum staff.	nvarchar(MAX)	Yes
	Gender	Gender of museum staff.	int	Yes
Ticket	ID (PK)	Identifier of ticket	int	No
	Code	Code of ticket that visitor use to access to system.	nchar(10)	No
	Release_Date	Time that ticket is created.	bigint	No

	Status	Status of ticket include 2 status: USED,NOT_USED	int	No
	DeviceID	Identifier of device that access to system by this ticket.	nvarchar(25)	Yes
Device	DeviceID (PK)	Identifier of device	nvarchar(25)	No
	RegistrationID	Registration ID is an identifier assigned by GCM to a single instance of a single application installed on an Android device.	nvarchar(300)	Yes
	Username	Username of museum staff who login to system.	nvarchar(100)	Yes
Shift	ID (PK)	Identifier of shift.	int	No
	Start time	Time that shift start.	bigint	No
	End time	Time that shift finish.	bigint	No
ShiftAccount	ShiftID	Identifier of shift.	int	No
	Username	Username of museum staff who work in this shift.	nvarchar(100)	No
	LocationID	Identifier of location of shift.	int	Yes
	ShiftTime	Time that museum staff work in shift	bigint	Yes
Location	LocationID	Identifier of location	int	No
	AreaID	Identifier of museum area	int	No
	RoomID	Identifier of museum room	int	Yes
	FloorID	Identifier of museum floor	int	No
Area	ID	Identifier of museum area	int	No
	Name	Name of museum area	nvarchar(20)	No
	RoomID	Identifier of museum room	int	Yes
	FloorID	Identifier of museum floor	int	No

	Height	The height of the area	Double	Yes
	Width	The width of the area	Double	Yes
	Length	The length of the area	Double	Yes
Room	ID	Identifier of museum room	int	No
	Name	Name of museum room	nvarchar(20)	No
	FloorID	Identifier of museum floor	int	Yes
Floor	ID	Identifier of museum floor	int	No
	Name	Name of museum floor	nvarchar(20)	No
	LinkImage	Link image of floor	nvarchar(MAX)	Yes
Report	ID	Identifier of report	int	No
	ItemID	Identifier of historic item that is reported.	int	No
	Time	Time that visitor send report of item.	bigint	No
	Message	Message of report	nvarchar(MAX)	Yes
	isSeen	Status of report have two status NOT_SEEN and SEEN. When report is create, it is NOT_SEEN.	int	No
	Status	Status of report have three status UNASSIGN, PROCESSING and DONE.	int	No
Notification	ID	Identifier of a notification.	int	No
	Status	Status of notification have two status NOT_CHECK and CHECK. Default status is NOT_CHECK.	int	No
	OccurTime	Time that notification is created.	bigint	No

	Message	Message of notification.	nvarchar(MAX)	Yes
	UpdateTime	Time that museum staff updates notification.	bigint	No
	SenderUsername	Username of account that sends notification	nvarchar(100)	Yes
	RecieverUsername	Username of museum staff that receives notification.	nvarchar(100)	No
	ItemID	Identifier of moved item.	int	No
Beacon	ID	Identifier of a beacon	int	No
	Name	Name of beacon	nvarchar(MAX)	Yes
	UUID	Universally unique identifier of beacon	nchar(37)	No
	Major	Major of beacon	int	No
	Minor	Minor of beacon	int	No
	MacAddress	Mac address of beacon	nchar(20)	No
	isAvailable	Status of beacon is available or unavailable.	int	No
	isDeleted	Status of beacon have two status DELETED and UNDETED.	int	No
RelatedItem	ItemID	Identifier of an item.	int	No
	RelatedItemID	Identifier of a related item.	int	No
	Similarity	Index of similarity of two item.	float	Yes
NearByItem	ItemID	Identifier of an item.	int	No
	NearByItemID	Identifier of a nearby item.	int	No
Languages	LanguagesID	Identifier of a nearby language.	int	No
	Name	Name of language	nvarchar(MAX)	No
ItemLanguage	ItemID	Identifier of item	int	No
	LanguagesID	Identifier of language	int	No
	ItemName	Name of item	nvarchar(MAX)	Yes
	ItemDescription	Description of item	nvarchar(MAX)	Yes

Item	ID (PK)	Identifier of historic item	int	No
	LinkImage	Link image of historic item	nvarchar(MAX)	Yes
	BeaconID	Identifier of beacon that map with this historic item.	int	Yes
	LocationID	Identifier of location that contain this this item.	int	Yes
	Status	Status of item	int	Yes
	ApprovedItemID	Identifier of item which item name is the same new item name but their description is not the same.	int	Yes
	Height	The height of the item	Double	Yes
	Width	The width of the item	Double	Yes
	Length	The length of the item	Double	Yes

Table 127. Data table dictionary

3. System Architectural Implementation

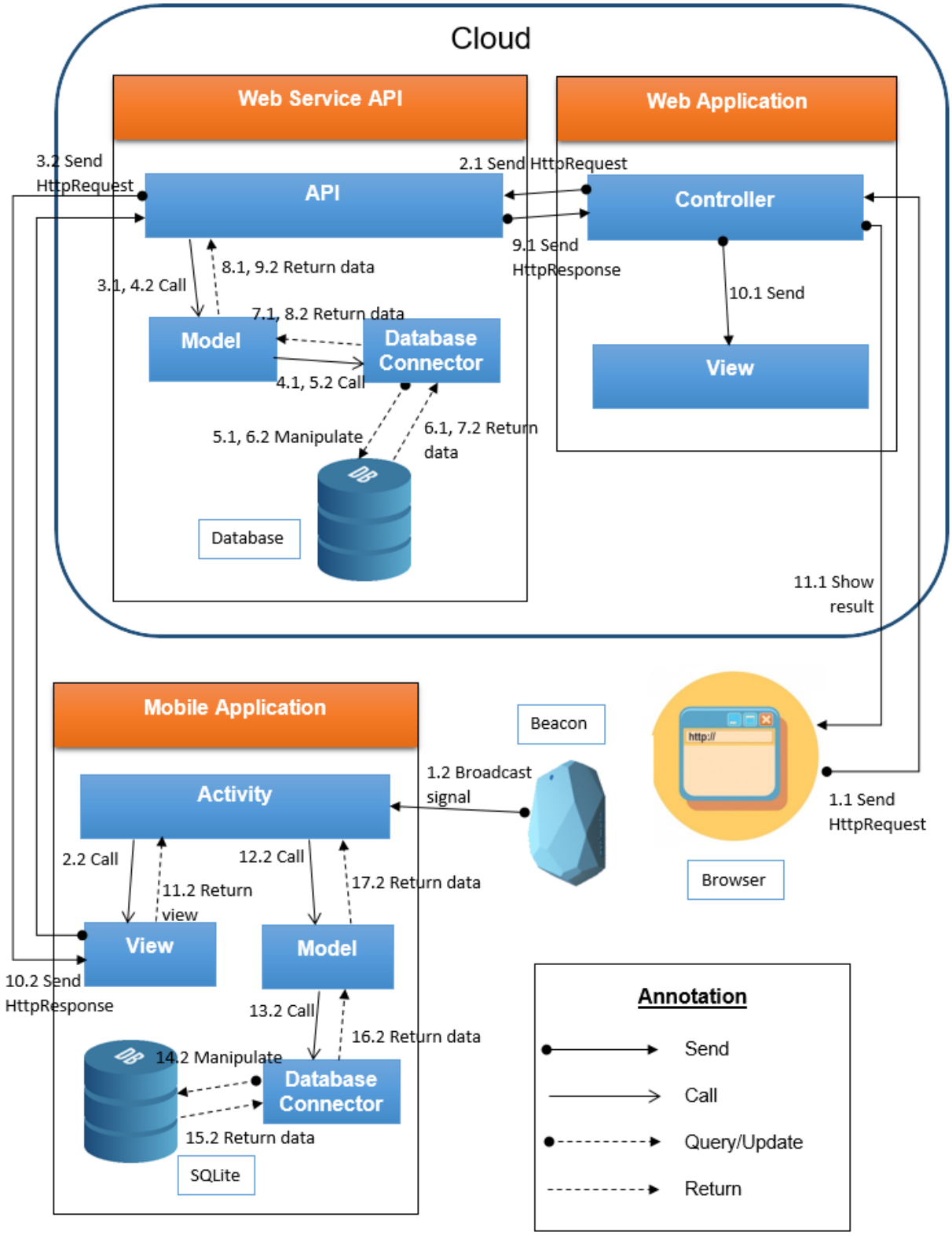


Figure 87. System Architectural Implementation

Number	From	To	Type	Description
1.1	Browser	[Web Application] Controller	Send	Send HttpRequest
2.1	[Web Application] Controller	[Web Service] API	Send	Send HttpRequest
3.1	[Web Service] API	[Web Service] Model	Call	Process data
4.1	[Web Service] Model	Database	Query/Update	Manipulate data
5.1	Database	[Web Service] Model	Return	Return data
6.1	[Web Service] Model	[Web Service] API	Return	Return data
7.1	[Web Service] API	[Web Application] Controller	Send	Return data
8.1	[Web Application] Controller	[Web Application] View	Send	Create view
9.1	[Web Application] View	Browser	Send	Show result

Table 128. Executing flow of Web Application

Number	From	To	Type	Description
1.2	Beacon	[Mobile App] Activity	Send	Broadcast Signal
2.2	[Mobile App] Activity	[Mobile App] View	Call	Create view
3.2	[Mobile App] View	[Web Service] API	Send	Send HttpRequest
4.2	[Web Service] API	[Web Service] Model	Call	Process data
5.2	[Web Service] Model	Database	Query/Update	Manipulate data
6.2	Database	[Web Service] Model	Return	Return data
7.2	[Web Service] Model	[Web Service] API	Return	Return data
8.2	[Web Service] API	[Mobile App] View	Send	Send HttpResponse
9.2	[Mobile App] View	[Mobile App] Activity	Return	Return view
10.2	[Mobile App] Activity	[Mobile App] Model	Call	Process data

11.2	[Mobile App] Model	Database	Query/Update	Manipulate local data
12.2	Database	[Mobile App] Model	Return	Return data
13.2	[Mobile App] Model	[Mobile App] Activity	Return	Return data

Table 129. Executing flow of Mobile Application

4. Performance measures

4.1 Mobile Application API load speed

4.1.1 Definition

This sections tests the load speed of mobile app when connect to server through API.

4.1.2 Test Environment

Server

Operation System	Windows Server 2012 64 bit
RAM	8GB
Storage	100GB
Processor	Intel@ CORE i5 Quad core 2.4 GHz
Network	Cable 1Mbps

Table 130. Server Test Environment

Mobile

Internet connection	Wi-Fi 4Mbps
Operation System	Android 5.0.2: Lollipop
Processor	Qualcomm Snapdragon 800 2.27 GHz
RAM	2GB
Bluetooth	Bluetooth 4.0 BLE (Bluetooth Less Energy)

Table 131. Server Test Environment

4.1.3 Test case

Using mobile application to send entire API request to Web Server API

List of APIs:

No.	API	Description
1	checkTicketCode	Check ticket code of visitor is valid or not
2	getItemInfo	Get information of a historic item
3	findLocationOfItem	Find location of an historic item (included: floor, room, area and nearByItem)

Table 132. Mobile API load speed test case

4.1.4 Test result

4.1.4.1 API checkTicketCode

The test is run 20 times, each time we run all the test cases listed above and log down the average API response time

No.	Average page load time (second)	Execute date
1	0.7	25 July 2016
2	1.1	25 July 2016
3	1.3	25 July 2016
4	0.63	25 July 2016
5	0.51	25 July 2016
6	0.82	25 July 2016
7	0.81	25 July 2016
8	0.72	25 July 2016
9	1.0	25 July 2016
10	1.34	25 July 2016
11	1.22	25 July 2016
12	1.51	25 July 2016
13	2.0	25 July 2016
14	0.51	25 July 2016
15	0.92	25 July 2016
16	0.53	25 July 2016
17	0.71	25 July 2016
18	0.69	25 July 2016
19	0.71	25 July 2016
20	0.86	25 July 2016
	Average: 0.9295s	

Table 133. API checkTicketCode load speed test result

4.1.4.2 API getItemInfo

The test is run 40 times, each time we run all the test cases listed above and log down the average API response time

No.	Average page load time (second)	Execute date
1	0.7	25 July 2016
2	1.1	25 July 2016
3	1.3	25 July 2016
4	1.53	25 July 2016
5	0.86	25 July 2016
6	0.82	25 July 2016
7	0.96	25 July 2016
8	1.6	25 July 2016
9	0.97	25 July 2016
10	0.89	25 July 2016
11	0.98	25 July 2016
12	0.86	25 July 2016
13	0.89	25 July 2016
14	0.87	25 July 2016
15	0.92	25 July 2016
16	0.59	25 July 2016
17	0.71	25 July 2016
18	0.66	25 July 2016
19	0.71	25 July 2016
20	1.87	25 July 2016
	Average: 0.9895s	

Table 134. API getItemInfo load speed test result

4.1.4.3 API findLocationOfItem

The test is run 40 times, each time we run all the test cases listed above and log down the average API response time

No.	Average page load time (second)	Execute date
1	0.92	25 July 2016
2	0.59	25 July 2016
3	0.71	25 July 2016
4	0.66	25 July 2016
5	0.71	25 July 2016
6	1.87	25 July 2016
7	0.92	25 July 2016
8	0.59	25 July 2016
9	1.51	25 July 2016
10	0.51	25 July 2016
11	0.92	25 July 2016
12	0.71	25 July 2016
13	0.89	25 July 2016
14	0.99	25 July 2016
15	0.97	25 July 2016
16	0.53	25 July 2016
17	0.71	25 July 2016
18	0.69	25 July 2016
19	0.97	25 July 2016
20	1.16	25 July 2016
	Average: 0.8765s	

Table 135. API findLocationOfItem load speed test result

5. Test Plan

The overall purpose of testing is to ensure iMuseum system meets its entire technical, functional and business requirement. The purpose of this document is to describe the overall test plan and strategy for testing the core flow of iMuseum system.

The approach described in this document provides the framework for all testing related to this application. Each test cases will be written for each version of the application that is released. This document will also be updated as required for each release.

5.1 Features to be tested

We will carry out test based on core workflow of system. All main functions will be tested carefully and clearly following phases.

- Admin: insert item, update item, insert beacon.
- Expert: approve item.
- Staff: track item.
- Visitor: view item information, save item, report item, detect item.

5.2 Features not to be tested

- Guest: login, logout
- User: search
- Scheduler: crawl data
- Admin: update beacon

6. System testing test case

6.1 Communication diagram

Test case are created from below communication diagram

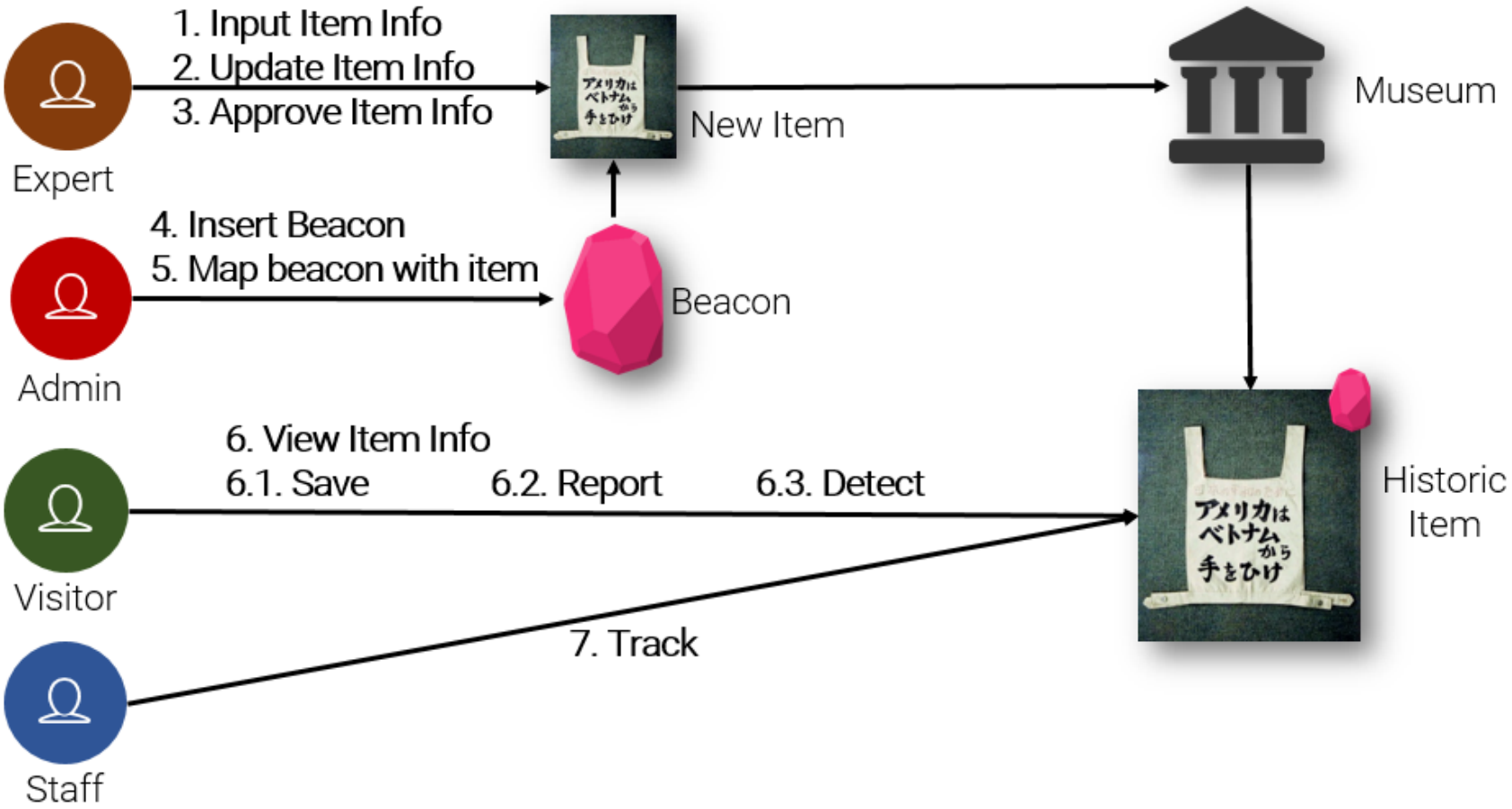


Figure 88. Web application communication diagram

6.2Test case

6.2.1 <Admin> Insert Beacon

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
IB1	Test Admin add a new beacon first time	1. Database for containing beacon is blank. 2. Bluetooth of mobile is turned on. 3. Having internet connection.	1. Admin chooses a beacon 2. All beacon information will be shown. 3. Admin send add request to insert new beacon.	1. System shows “Add beacon successfully”. 2. New beacon is inserted into system database.	N/A	Pass	26 July 2016
IB2	Test Admin add a new beacon second time	1. Database contains 1 beacon. 2. Bluetooth of mobile is turned on. 3. Having internet connection.	1. Admin chooses a beacon 2. All beacon information will be shown. 3. Admin send add request to insert new beacon.	1. System shows “Add beacon successfully”. 2. New beacon is inserted into system database.	N/A	Pass	26 July 2016
IB3	Test Admin add a new beacon third time	1. Database contains 2 beacons.	1. Admin chooses a beacon	1. System shows “Add beacon successfully”.	N/A	Pass	26 July 2016

		2. Bluetooth of mobile is turned on. 3. Having internet connection.	2. All beacon information will be shown. 3. Admin send add request to insert new beacon.	2. New beacon is inserted into system database.			
IB4	Test Admin add a new beacon forth time with an Added-Beacon	1. Bluetooth of mobile is turned on.	1. Admin chooses an Added-Beacon	1. System redirects to Map Beacon page.	N/A	Pass	26 July 2016
IB5	Test Admin add a new beacon fifth time with an Added-Beacon	1. Bluetooth of mobile is turned on.	1. Admin chooses an Added-Beacon	1. System redirects to Map Beacon page.	N/A	Pass	26 July 2016
IB6	Test Admin add a new beacon sixth time and have no internet connection	1. Bluetooth of mobile is turned on. 2. No internet connection.	1. Admin chooses a beacon 2. All beacon information will be shown. 3. Admin send add request to insert new beacon.	System shows “Please check your internet connection”.	N/A	Pass	26 July 2016
IB7	Test Admin add a new beacon seventh time	1. Database contains 3 beacons.	1. Admin chooses a beacon	System shows “Add beacon fail, Try again later!”.	N/A	Pass	26 July 2016

	with server is error	2. Bluetooth of mobile is turned on. 3. Having internet connection.	2. All beacon information will be shown. 3. Admin send add request to insert new beacon.				
IB8	Test Admin add a new beacon eighth time	1. Database contains 3 beacons. 2. Bluetooth of mobile is turned on. 3. Having internet connection.	1. Admin chooses a beacon 2. All beacon information will be shown. 3. Admin send add request to insert new beacon.	1. System shows “Add beacon successfully”. 2. New beacon is inserted into system database.	N/A	Pass	26 July 2016

Table 136. Insert Beacon Test Case

6.2.2 <Admin> Map Beacon with Item

Mobile

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
MB1	Test Admin maps an existed beacon with an existed item first time	1. Database contains 4 Unmapped-Beacons. 2. Bluetooth of mobile is turned on.	1. Admin chooses an available beacon 2. Map beacon page will be shown. 3. Admin chooses a valid historic item name. 4. Admin choose a valid location. 5. Admin send request to map a valid beacon with an item	1. System shows “Map beacon successfully”. 2. There are 3 available beacons in system database.	N/A	Pass	26 July 2016
MB2	Test Admin maps an existed beacon with an existed item second time	1. Database contains 3 Unmapped-Beacons. 2. Bluetooth of mobile is turned on.	1. Admin chooses an available beacon 2. Map beacon page will be shown.	1. System shows “Map beacon successfully”. 2. There are two available beacon in system database	N/A	Pass	26 July 2016

			3. Admin chooses a valid historic item name. 4. Admin choose a valid location. 5. Admin send request to map a valid beacon with an item				
MB3	Test Admin maps an existed beacon with an existed item third time	1. Database contains 2 available beacons. 2. Bluetooth of mobile is turned on.	1. Admin chooses an available beacon 2. Map beacon page will be shown. 3. Admin chooses a valid historic item name. 4. Admin choose a valid location. 5. Admin send request to map a valid beacon with an item	1. System shows “Map beacon successfully”. 2. There are one available beacon in system database	N/A	Pass	26 July 2016
MB4	Test Admin maps an existed beacon with an invalid item.	1. Database contains 1 available beacon.	1. Admin chooses an available beacon 2. Map beacon page will be shown.	1. System shows “Please specify a valid historic item name”.	N/A	Pass	26 July 2016

		2. Bluetooth of mobile is turned on.	3. Admin chooses an invalid historic item name. 4. Admin choose a valid location. 5. Admin send request to map a valid beacon with an item				
MB5	Test Admin maps an existed beacon with a valid item but invalid location.	1. Database contains 1 available beacon. 2. Bluetooth of mobile is turned on.	1. Admin chooses an available beacon 2. Map beacon page will be shown. 3. Admin chooses an invalid historic item name. 4. Admin choose an invalid location. 5. Admin send request to map a valid beacon with an item	1. System shows “Please specify a valid location”.	N/A	Pass	26 July 2016
MB6	Test Admin maps an unavailable beacon.	1. Database contains 1 available beacon.	1. Admin chooses an unavailable beacon	1. System shows “Beacon is mapped with item …”.	N/A	Pass	26 July 2016

		2. Bluetooth of mobile is turned on.					
MB7	Test Admin maps an existed beacon with a valid item with no internet connection.	1. Database contains 1 available beacon. 2. Bluetooth of mobile is turned on. 3. No internet connection	1. Admin chooses an available beacon 2. Map beacon page will be shown. 3. Admin chooses a valid historic item name. 4. Admin choose a valid location. 5. Admin send request to map a valid beacon with an item	System shows “Please check your internet connection”.	N/A	Pass	26 July 2016
MB8	Test Admin maps an existed beacon with a valid item with server error.	1. Database contains 1 available beacon. 2. Bluetooth of mobile is turned on.	1. Admin chooses an available beacon 2. Map beacon page will be shown. 3. Admin chooses a valid historic item name. 4. Admin choose a valid location.	System shows “Cannot map beacon with item, please check again later!”	N/A	Pass	26 July 2016

			5. Admin send request to map a valid beacon with an item				
MB9	Test Admin maps an existed beacon with an existed item third time	1. Database contains 1 available beacons. 2. Bluetooth of mobile is turned on.	1. Admin chooses an available beacon 2. Map beacon page will be shown. 3. Admin chooses a valid historic item name. 4. Admin choose a valid location. 5. Admin send request to map a valid beacon with an item	1. System shows “Map beacon successfully”. 2. There is no available beacon in system database	N/A	Pass	26 July 2016

Web Application

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
MB10	Test Admin maps an existed beacon with an existed item first time	Database contains 4 Unmapped-Beacons.	1. Admin chooses a valid historic item name. 2. Admin choose a valid location. 3. Admin choose a valid beacon 5. Admin send request to map a valid beacon with an item	1. System shows “Map beacon successfully”. 2. There are 3 available beacons in system database.	N/A	Pass	26 July 2016
MB11	Test Admin maps an existed beacon with an existed item second time	Database contains 3 Unmapped-Beacons.	1. Admin chooses a valid historic item name. 2. Admin choose a valid location. 3. Admin choose a valid beacon 5. Admin send request to map a valid beacon with an item	1. System shows “Map beacon successfully”. 2. There are 2 available beacons in system database.	N/A	Pass	26 July 2016

MB12	Test Admin maps an existed beacon with an existed item third time	Database contains 2 Unmapped-Beacons.	1. Admin chooses a valid historic item name. 2. Admin choose a valid location. 3. Admin choose a valid beacon 5. Admin send request to map a valid beacon with an item	1. System shows “Map beacon successfully”. 2. There are 1 available beacons in system database.	N/A	Pass	26 July 2016
MB13	Test Admin maps an existed beacon with an existed item with server error	Database contains 1 Unmapped-Beacons.	1. Admin chooses a valid historic item name. 2. Admin choose a valid location. 3. Admin choose a valid beacon 5. Admin send request to map a valid beacon with an item	1. System shows “Cannot map beacon with item, please check again later!” 2. There are 1 available beacons in system database.	N/A	Pass	26 July 2016
MB14	Test Admin maps an existed beacon with an existed item but invalid location	Database contains 1 Unmapped-Beacons.	1. Admin chooses a valid historic item name. 2. Admin choose an invalid location.	1. System shows “Please specify valid location”	N/A	Pass	26 July 2016

			3. Admin choose a valid beacon 5. Admin send request to map a valid beacon with an item	2. There are 1 available beacons in system database.			
MB15	Test Admin maps an existed beacon with an existed item forth time	Database contains 1 Unmapped-Beacons.	1. Admin chooses a valid historic item name. 2. Admin choose a valid location. 3. Admin choose a valid beacon 5. Admin send request to map a valid beacon with an item	1. System shows “Map beacon successfully”. 2. There is no available beacons in system database.	N/A	Pass	26 July 2016
MB16	Test Admin maps an existed item but have no available beacon	Database contains 0 Unmapped-Beacons.	1. Admin chooses a valid historic item name. 2. Admin choose a valid location. 3. Admin choose a beacon	System is not show any beacon for admin to choose	N/A	Pass	26 July 2016

			5. Admin send request to map a valid beacon with an item				

Table 137. Insert Beacon Test Case

6.2.3 <Visitor> Get Historic Item Information

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
GII01	Test Visitor gets information of a historic item with no related item.	1. Database contains valid historic item and mapped with a beacon. 2. Having internet connection 3. Bluetooth is turned on. 4. Mobile is in-range of beacon.	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	1. Load all historic image 2. Show all description of historic item.	N/A	Pass	26 July 2016
GII02	Test Visitor gets information of a historic item with 1 related item.	1. Database contains valid historic item and mapped with a beacon.	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	1. Load all historic image 2. Show all description of historic item.	N/A	Pass	26 July 2016

		2. Having internet connection 3. Bluetooth is turned on. 4. Mobile is in-range of beacon.		3. Show 1 related item.			
GII03	Test Visitor gets information of a historic item with 2 related items.	1. Database contains valid historic item and mapped with a beacon. 2. Having internet connection 3. Bluetooth is turned on. 4. Mobile is in-range of beacon.	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	1. Load all historic image 2. Show all description of historic item. 3. Show 2 related items.	N/A	Pass	26 July 2016
GII04	Test Visitor gets information of a historic item with 3 related items.	1. Database contains valid historic item and mapped with a beacon. 2. Having internet connection	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	1. Load all historic image 2. Show all description of historic item. 3. Show 3 related items.	N/A	Pass	26 July 2016

		3. Bluetooth is turned on. 4. Mobile is in-range of beacon.					
GII05	Test Visitor gets information of a historic item with 4 related items.	1. Database contains valid historic item and mapped with a beacon. 2. Having internet connection 3. Bluetooth is turned on. 4. Mobile is in-range of beacon.	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	1. Load all historic image 2. Show all description of historic item. 3. Show 4 related items.	N/A	Pass	26 July 2016
GII06	Test Visitor gets information of a historic item with 5 related items.	1. Database contains valid historic item and mapped with a beacon. 2. Having internet connection 3. Bluetooth is turned on.	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	1. Load all historic image 2. Show all description of historic item. 3. Show 5 related items.	N/A	Pass	26 July 2016

		4. Mobile is in-range of beacon.					
GII07	Test Visitor gets information of a historic item with no internet connection.	1. Database contains valid historic item and mapped with a beacon. 2. Bluetooth is turned on. 3. Mobile is in-range of beacon.	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	System shows “Please check your internet connection!”	N/A	Pass	26 July 2016
GII08	Test Visitor gets information of a historic item with server error.	1. Database contains valid historic item and mapped with a beacon. 2. Having internet connection 3. Bluetooth is turned on. 4. Mobile is in-range of beacon.	1. Visitor chooses a historic item. 2. Visitor sends request to get information of an item.	System shows “Cannot get historic item information!”	N/A	Pass	26 July 2016

Table 138. Get Historic Item Test Case

6.2.4 <Visitor> Add Item to Favorite

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
AFI01	Test Visitor add item to favorite first time with existed category	Visitor got historic item information	1. Visitor requests add historic item to favorite 2. Visitor chooses category 3. Visitor sends request to add information of an item.	1. Item images are downloaded into mobile local storage. 2. Historic item information is saved into local database	N/A	Pass	26 July 2016
AFI02	Test Visitor add item to favorite second time with existed category	Visitor got historic item information	1. Visitor requests add historic item to favorite 2. Visitor chooses category 3. Visitor sends request to add information of an item.	1. Item images are downloaded into mobile local storage. 2. Historic item information is saved into local database	N/A	Pass	26 July 2016
AFI03	Test Visitor add item to favorite with not existed category	Visitor got historic item information	1. Visitor requests add historic item to favorite 2. Visitor creates valid category	1. Item images are downloaded into mobile local storage. 2. Historic item information is saved into local database	N/A	Pass	26 July 2016

			3. Visitor sends request to add information of an item.				
AFI04	Test Visitor add item to favorite with not existed category	Visitor got historic item information	1. Visitor requests add historic item to favorite 2. Visitor creates invalid category 3. Visitor sends request to add information of an item.	System shows “Please check category name again!”	N/A	Pass	26 July 2016
AFI05	Test Visitor requests add an added historic item	Visitor got historic item information	1. Visitor requests add historic item to favorite	System shows “Item is already added!”	N/A	Pass	26 July 2016

Table 139. Add Item to Favorite Test Case

6.2.5 <Visitor> Report Item

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
RI01	Test Visitor sends report item first time	Visitor got historic item information	1. Visitor requests report an historic item	1. System shows “Report successfully!”	N/A	Pass	26 July 2016

			2. Visitor inputs message 3. Visitor sends report.	2. Report is saved into system database			
RI02	Test Visitor sends report item second time	Visitor got historic item information	1. Visitor requests report an historic item 2. Visitor inputs message 3. Visitor sends report.	1. System shows “Report successfully!” 2. Report is saved into system database	N/A	Pass	26 July 2016
RI03	Test Visitor sends report item with no message	Visitor got historic item information	1. Visitor requests report an historic item 2. Visitor sends report.	1. System shows “Please input message!”	N/A	Pass	26 July 2016
RI04	Test Visitor sends report item with no internet connection	Visitor got historic item information	1. Visitor requests report an historic item 2. Visitor inputs message 3. Visitor sends report.	1. System shows “Please check your internet connection!”	N/A	Pass	26 July 2016
RI05	Test Visitor sends report item with server error	Visitor got historic item information	1. Visitor requests report an historic item 2. Visitor inputs message 3. Visitor sends report.	1. System shows “Cannot send report! Report will be sent next time!” 2. Report will be save into mobile local	N/A	Pass	26 July 2016

				database with status ‘unsent’			

Table 140. Report Item Test Case

6.2.6 <Visitor> Detect Item

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
DI01	Test Visitor start direction of an item first time	Visitor got historic item information	1. Visitor requests detect a historic item 2. Visitor moves following the direction. 3. The distance between visitor and item less than or equal 1.5m	1. Historic item name and image will be show	N/A	Pass	26 July 2016
DI02	Test Visitor start direction of an item second time	Visitor got historic item information	1. Visitor requests detect a historic item 2. Visitor moves following the direction. 3. The distance between visitor and	1. Mobile will be vibrate. 2. Historic item name and image will be show.	N/A	Pass	26 July 2016

			item less than or equal 1.5m				
DI03	Test Visitor start direction of an item.	Visitor got historic item information	1. Visitor requests detect a historic item 2. Visitor moves following the direction. 3. The distance between visitor and item greater than 1.5m	1. System continues detect historic item by scanning near by beacon.	N/A	Pass	26 July 2016

Table 141. Detect Item Test Case

6.2.7 <Staff> Track Item

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
TI01	Test Staff start tracking 2 “not-in-motion” items	1. Login as staff 2. Having internet connection	1. Staff requests tracking a historic item 2. Mobile authenticates beacon with estimate beacon.	1. Status of each item is “not in motion”	N/A	Pass	26 July 2016
TI02	Test Staff start tracking 3 “not-in-motion” items	1. Login as staff 2. Having internet connection	1. Staff requests tracking a historic item	1. Status of each item is “not in motion”	N/A	Pass	26 July 2016

			2. Mobile authenticates beacon with estimate beacon.				
TI03	Test Staff start tracking 4 “not-in-motion” items	1. Login as staff 2. Having internet connection	1. Staff requests tracking a historic item 2. Mobile authenticates beacon with estimate beacon.	1. Status of each item is “not in motion”	N/A	Pass	26 July 2016
TI04	Test Staff start tracking 0 “not-in-motion” items	1. Login as staff 2. Having internet connection	1. Staff requests tracking a historic item	System shows “No historic item in current shift need to be track!”	N/A	Pass	26 July 2016
TI05	Test Staff start tracking 2 “In-motion” items	1. Login as staff 2. Having internet connection	1. Staff requests tracking a historic item 2. Mobile authenticates beacon with estimate beacon.	1. Status of each item is “in motion” 2. Mobile device vibrates. 3. Mobile device sounds 4. Movement log is saved into system database.	N/A	Pass	26 July 2016
TI06	Test Staff start tracking 2 items with first item in	1. Login as staff	1. Staff requests tracking a historic item	1. Status of first item is “Not in motion” and	N/A	Pass	26 July 2016

	“Not in motion” status and second item in “In Motion” status	2. Having internet connection	2. Mobile authenticates beacon with estimate beacon.	second item is “in motion” 2. Mobile device vibrates. 3. Mobile device sounds 4. Movement log is saved into system database.			
TI07	Test Staff start tracking items with no internet connection	1. Login as staff	1. Staff requests tracking a historic item 2. Mobile authenticates beacon with estimate beacon.	1. System shows “No internet connection!” 2. Reconnect button of each items will be shown	N/A	Pass	26 July 2016
TI08	Test Staff start tracking 2 items and disconnect with first item.	1. Login as staff 2. Having internet connection	1. Staff requests tracking a historic item 2. Mobile authenticates beacon with estimate beacon. 3. Staff tracks items.	1. Reconnect button of first item will be shown	N/A	Pass	26 July 2016

Table 142. Track Item Test Case

6.2.8 <Expert> Approve Item

ID	Test Case Description	Precondition	Test Case Procedure	Expected Output	Inter-test Case dependence	Result	Test Date
AI1	Test Admin approves a “New” historic item for the first time.	System database contains at least 1 pending item. Its status is “New”. System database contains no active item.	1. Admin chooses a valid pending item 2. Admin sends request to approve historic item.	1. Historic item status is updated in system database 2. Historic item information is updated on “Manage Item View” in advance.	N/A	Pass	26 July 2016
AI2	Test Admin approves a “New” historic item for the second time.	System database contains at least 1 pending item. Its status is “New”. System database contains at least 1 active item.	1. Admin chooses a valid pending item 2. Admin sends request to approve historic item.	1. Historic item status is updated in system database 2. Historic item information is updated on “Manage Item View” in advance.	N/A	Pass	26 July 2016
AI3	Test Admin approves an “Updated” historic item	System database contains at least 1 pending item. Its status is “Updated”. System database contains at least 1 active item. Its	1. Admin chooses a valid pending item 2. Admin sends request to approve historic item.	1. Pending historic item status is removed in system database 2. New item information is applied	N/A	Pass	26 July 2016

		name is the same as the pending item mentioned above.		to the existed active item. 3. Historic item information is updated on “Manage Item View” in advance.			
AI4	Test Admin approves a historic item with server error or No Internet connection.	System database contains at least 1 pending item.	1. Admin chooses a valid pending item 2. Admin sends request to approve historic item.	1. System shows “Server Error! Historic item has failed to update!”	N/A	Pass	26 July 2016

Table 143. Approve Item Test Case

6.3 Test case results and statistics

Test case	Runs	Pass rate (%)
IB01	20	100
IB02	20	100
IB03	20	100
IB04	20	100
IB05	20	100
IB06	20	100
IB07	20	100
IB08	20	100
MB01	30	100
MB02	30	100
MB03	40	80
MB04	40	80
MB05	30	80
MB06	30	80
MB07	30	80
MB08	30	80
MB09	30	100
MB10	30	100
MB11	30	100
MB12	30	100
MB13	30	100
MB14	30	100
MB15	30	100
MB16	30	100

GII01	35	100
GII02	35	100
GII03	35	100
GII04	35	100
GII05	35	100
GII06	35	100
GII07	35	100
GII08	35	100
AFI01	20	100
AFI02	20	100
AFI03	20	100
AFI04	20	100
AFI05	20	100
RI01	20	100
RI02	20	100
RI03	20	100
RI04	20	100
RI05	20	100
DI01	20	90
DI02	20	90
DI03	20	90
TI01	20	100
TI02	20	100
TI03	20	100
TI04	20	90
TI05	20	90

TI06	20	90
TI07	20	100
TI08	20	80
AI01	20	100
AI02	20	100
AI03	20	100
AI04	20	100

Table 144. Test case result and statistic

F. System User's Manual

1. Installation Guide

1.1Setting up environment at server side

Bellows are requirements for hardware and software environment to run iMuseum system in 10 years. The specifications are based on the dependencies requirements and performance test result from previous section of this document.

1.1.1 Hardware requirements

Hardware	Specification
Internet Connection	8 Mbps
Computer Processor	Intel® CORE i7 Quad core 2.4 GHz
Computer Memory	4GB of RAM or more
Hard Disk Drive	50GB or more

Table 145. Hardware requirements

1.1.2 Software requirements

Software	Application name / version
Operating System	Ubuntu Server 14.04.2 LTS
Java	1.7.0_79
Web Server	Apache Tomcat 8.0.3
Database	SQL Server 2008

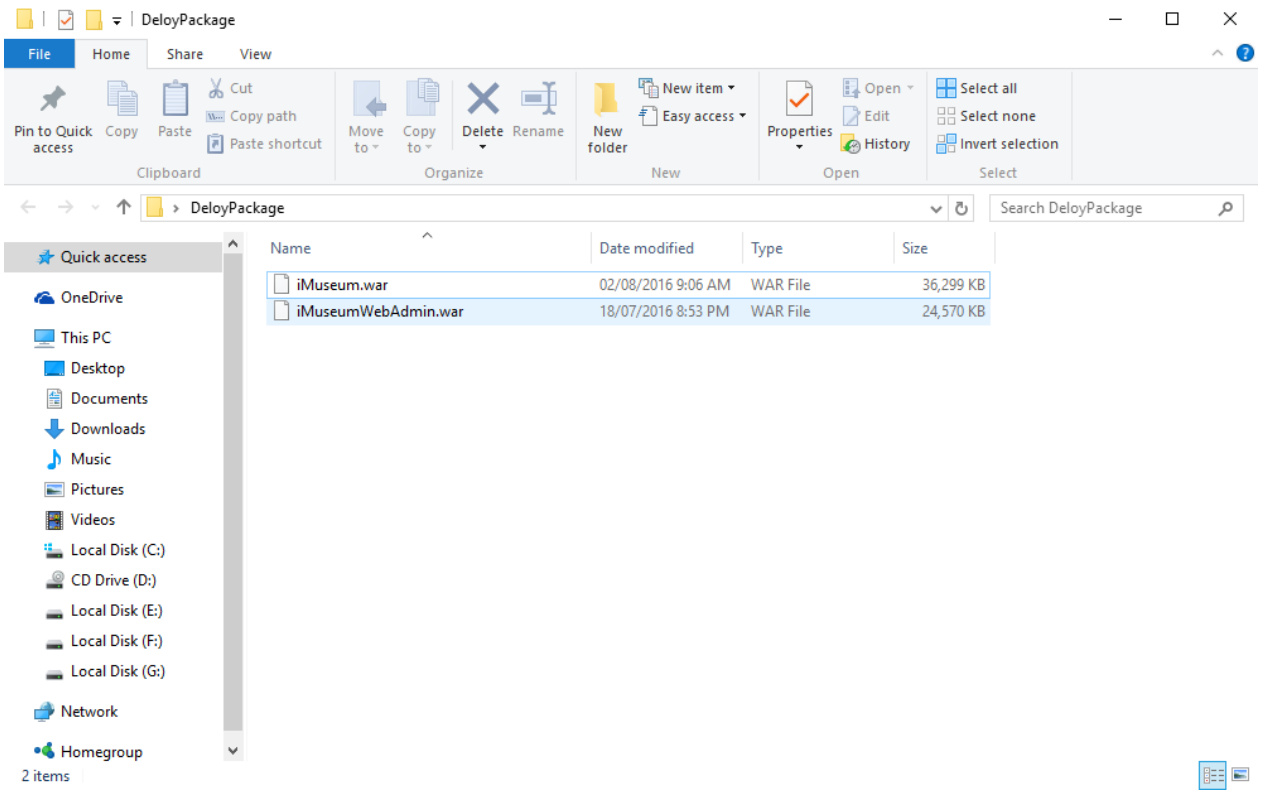
Table 146. Software requirements

1.2 Deployment of Web Service / Web Application

Register Azure account in https://portal.azure.com.

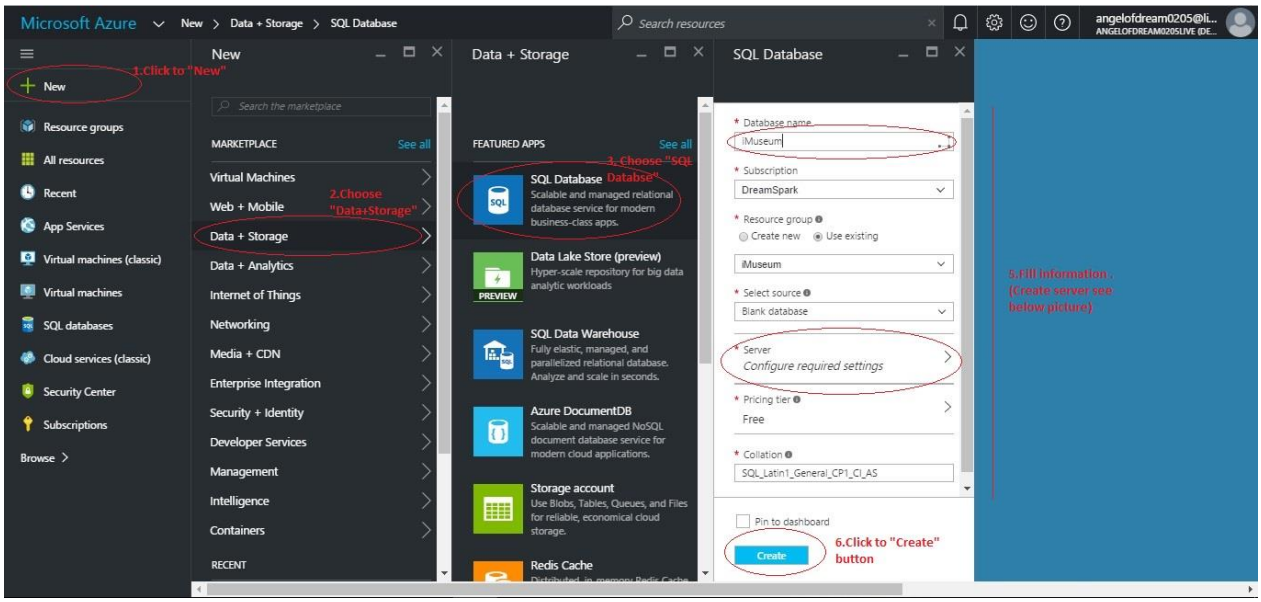
1.2.1 Prepare deployment package

- Create a new folder to contains deployment package, for example: Desktop\DeloyPackage
- Copy iMuseum.war file and iMuseumWebAdmin.war file to that folder.

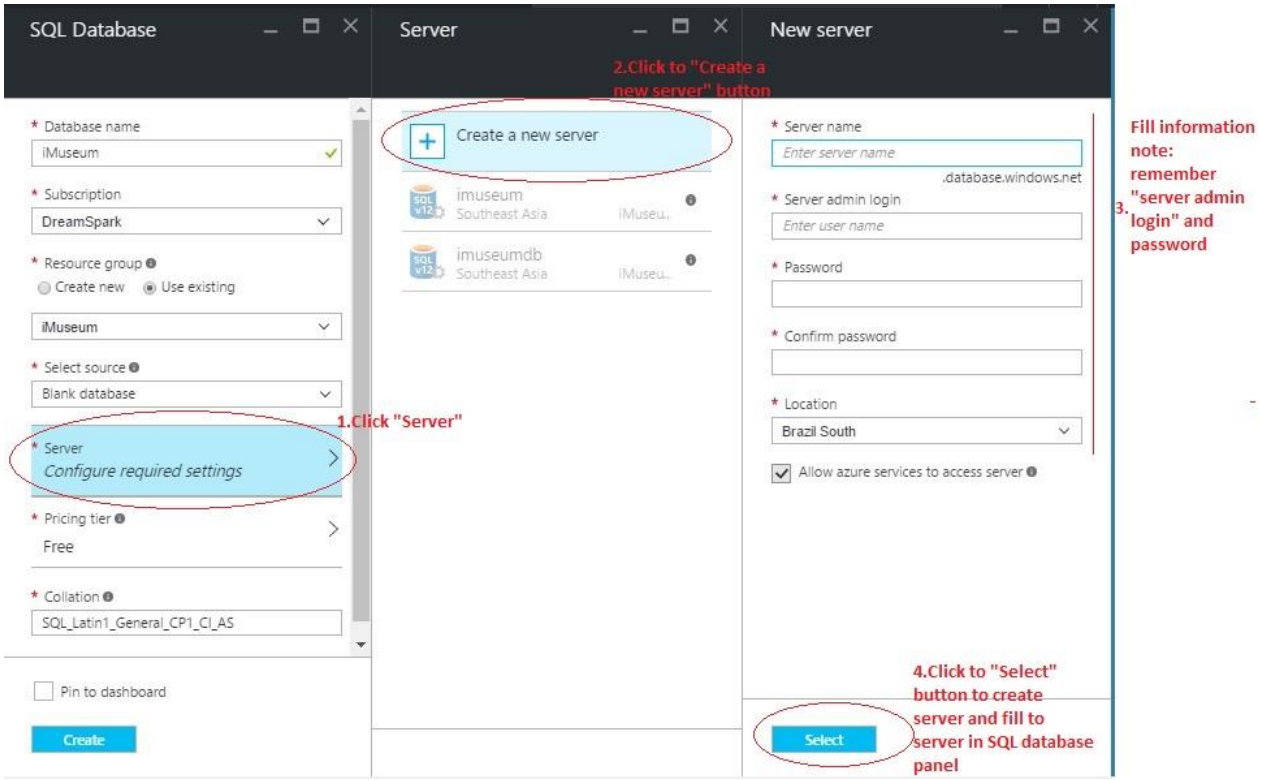


1.2.2 Database Deployment

Create SQL database.



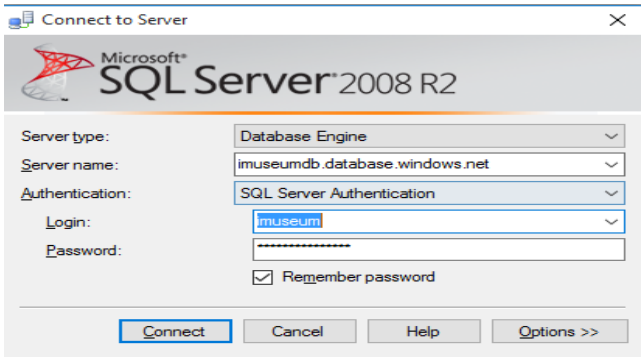
Select SQL server.



Open SQLAzureMW tool (link download: https://sqlazuremw.codeplex.com/) and do follow link: https://www.mssqltips.com/sqlservertip/3035/using-the-deploy-database-to-sql-azure-wizard-in-sql-server-management-studio-to-move-to-the-cloud/ to deploy database local to azure server.

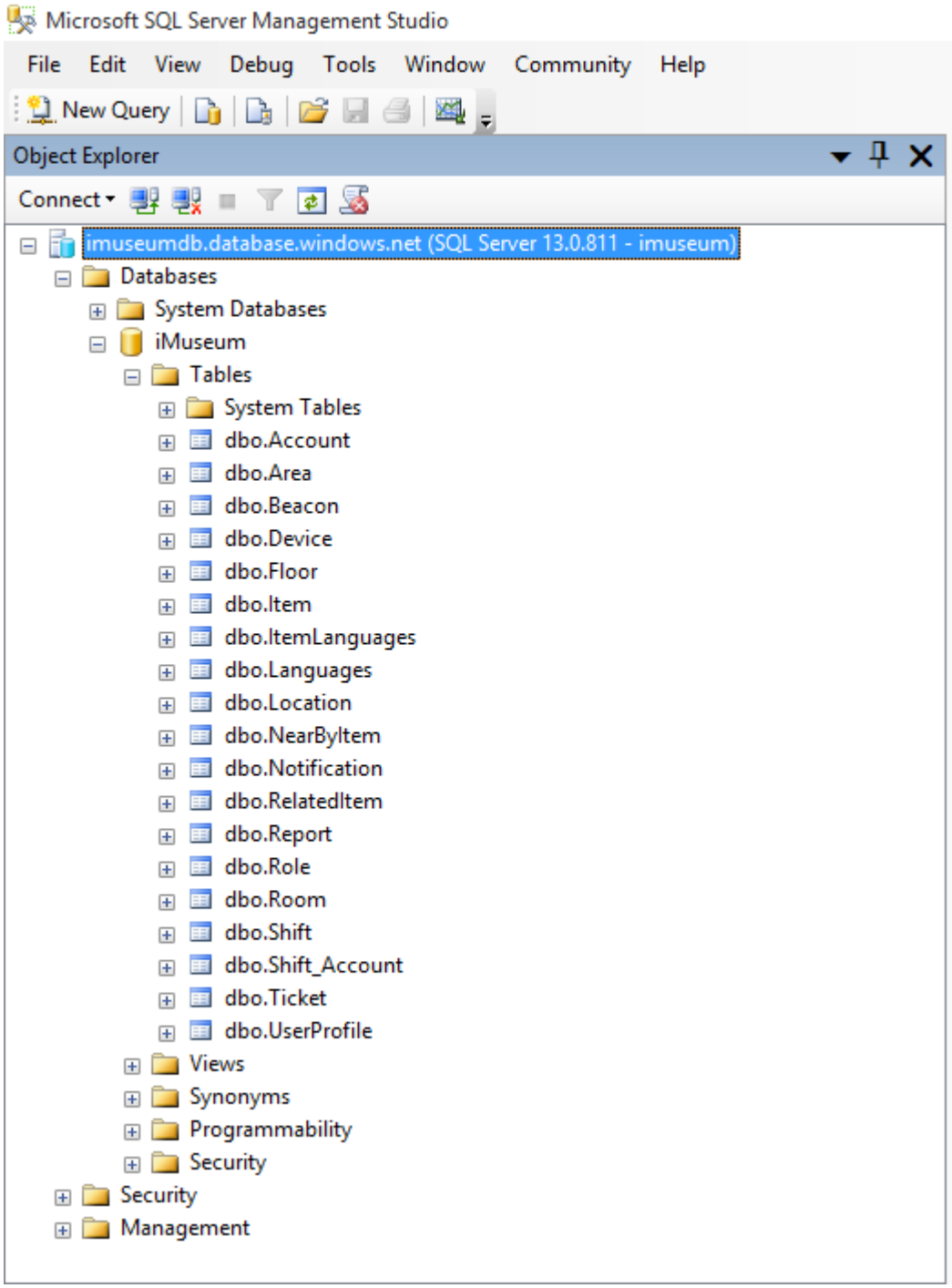
Open Microsoft SQL Server Management Studio. Connect to your SQL Server by entering:

- Server name: Server name of SQL server in azure.
- Authentication: Choose appropriated authentication method that meets your system specification.
- Login: Server admin login.
- Password: password of the above username.



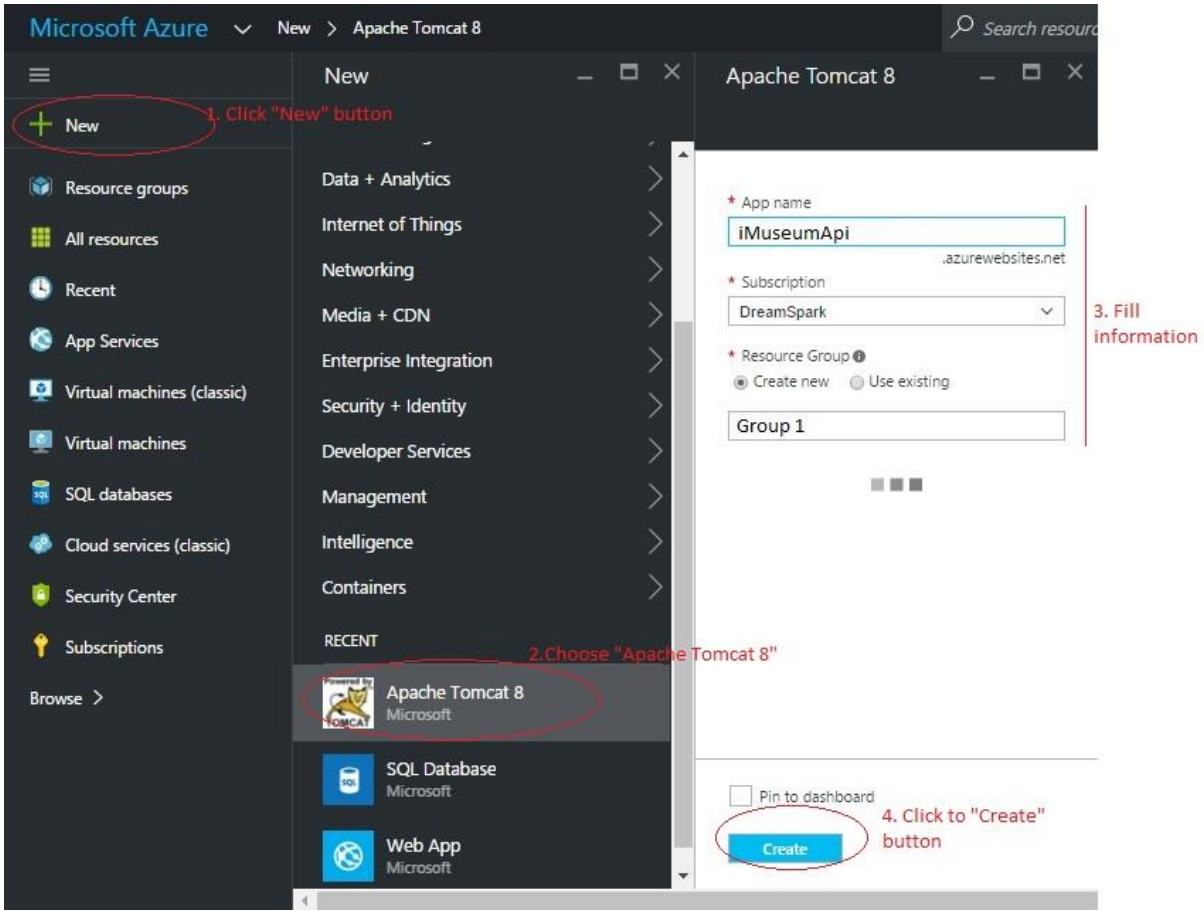
- Click connect and wait for operation to be completed.

Check if iMuseum database (19 tables) is exist successfully

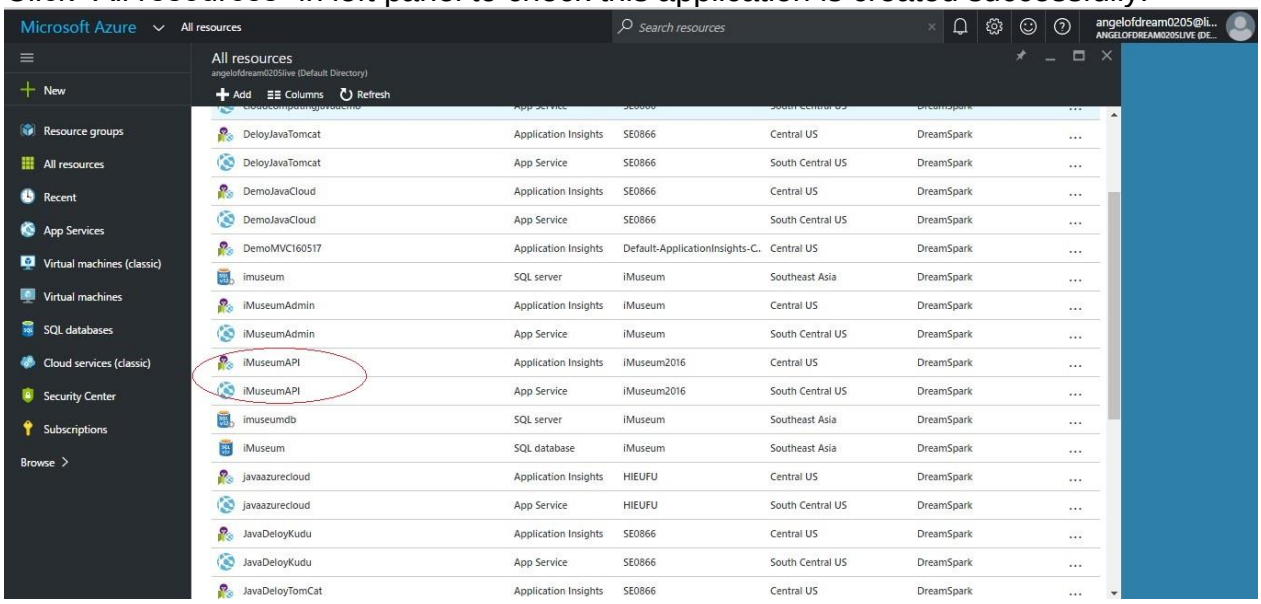

Microsoft SQL Server Management Studio
File Edit View Debug Tools Window Community Help
New Query
Object Explorer
Connect
imuseumdb.database.windows.net (SQL Server 13.0.811 - imuseum)
Databases
System Databases
iMuseum
Tables
System Tables
dbo.Account
dbo.Area
dbo.Beacon
dbo.Device
dbo.Floor
dbo.Item
dbo.ItemLanguages
dbo.Languages
dbo.Location
dbo.NearByItem
dbo.Notification
dbo.RelatedItem
dbo.Report
dbo.Role
dbo.Room
dbo.Shift
dbo.Shift_Account
dbo.Ticket
dbo.UserProfile
Views
Synonyms
Programmability
Security
Security
Management

1.2.3 Server configuration

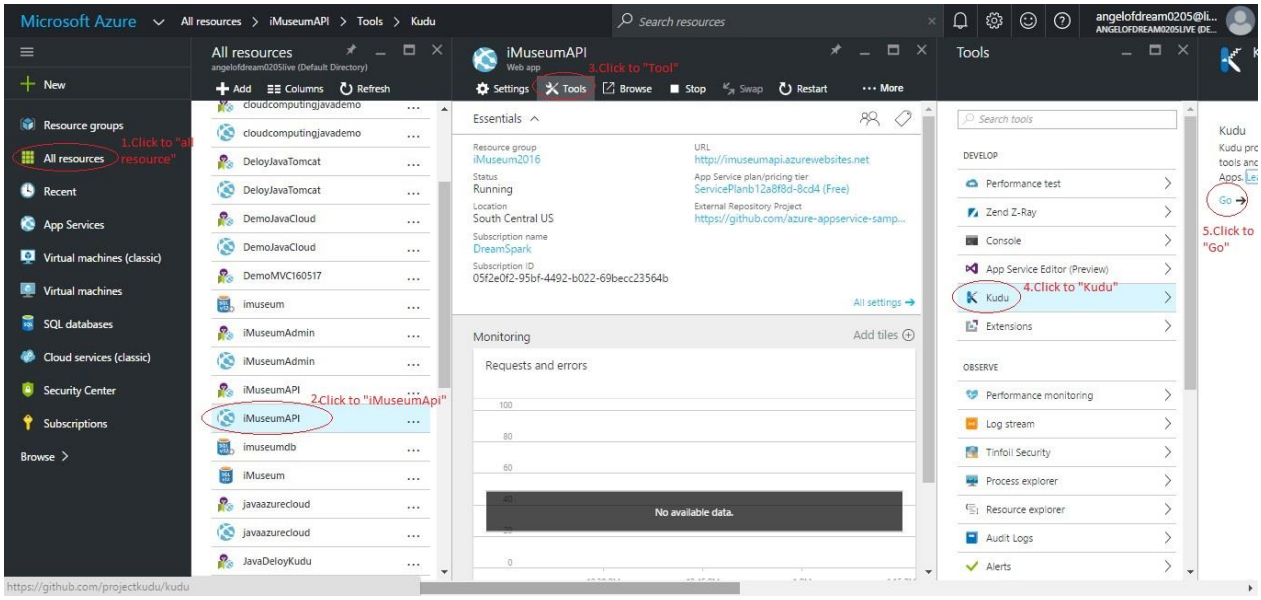
Create Tomcat server.



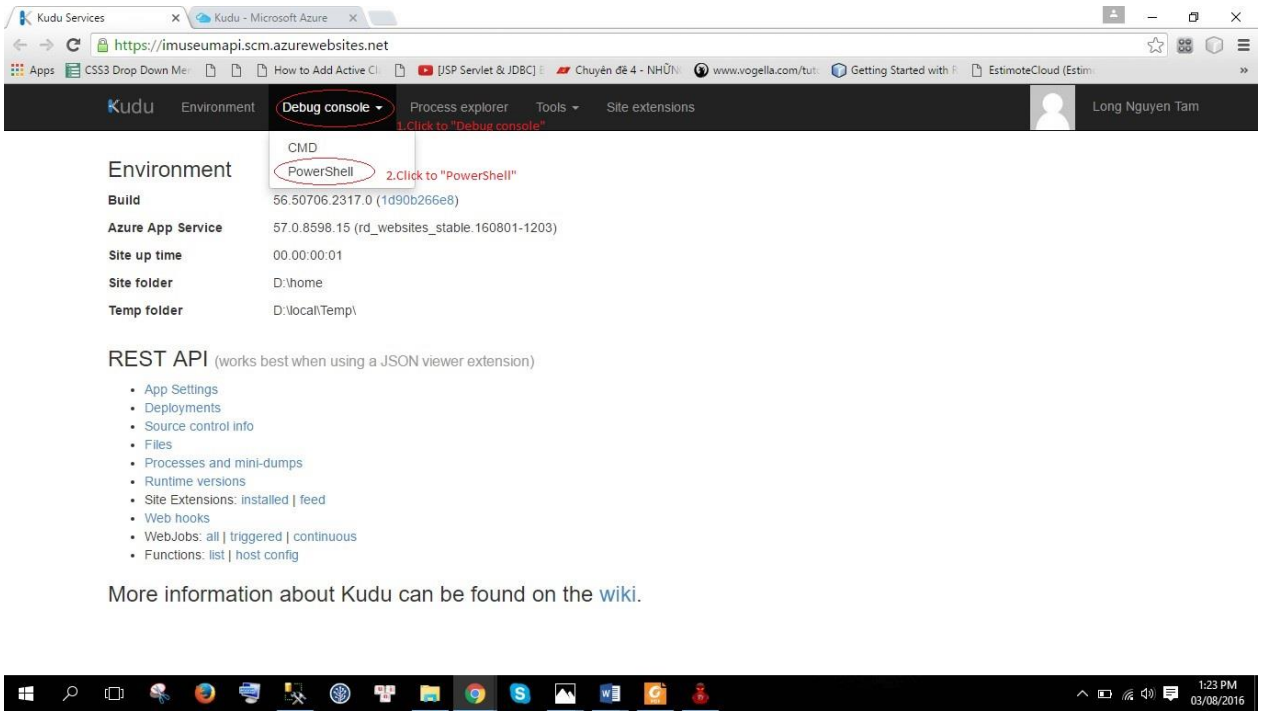
Click “All resources” in left panel to check this application is created successfully.



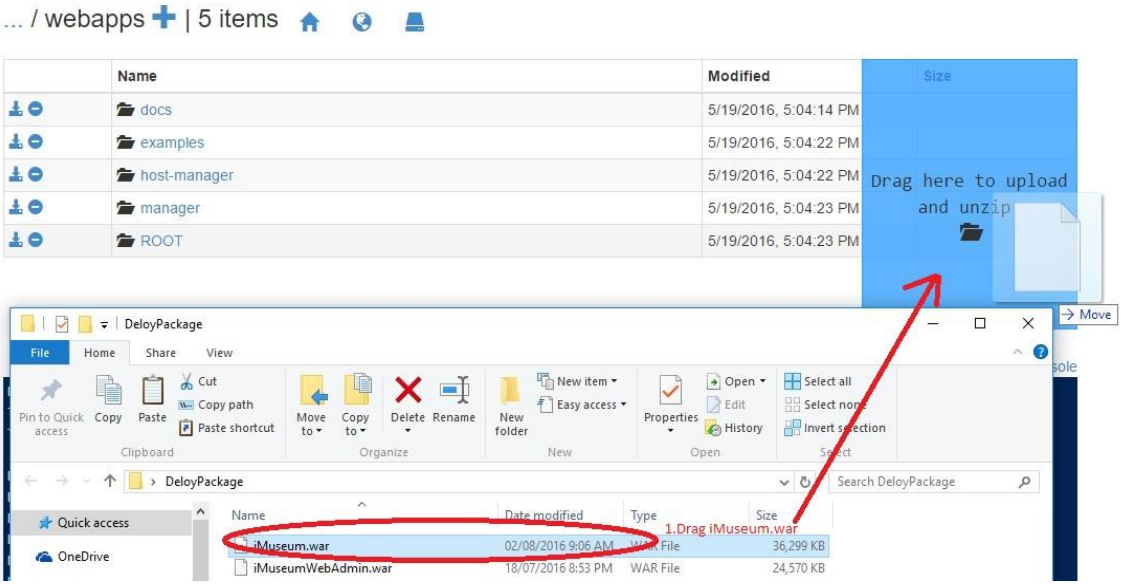
1.2.4 Deploy



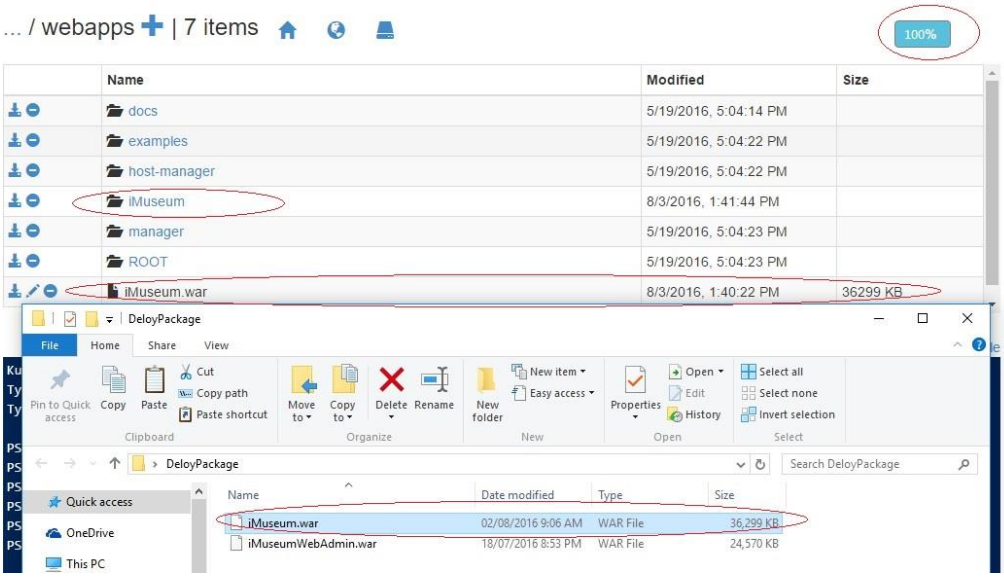
Kudu window.



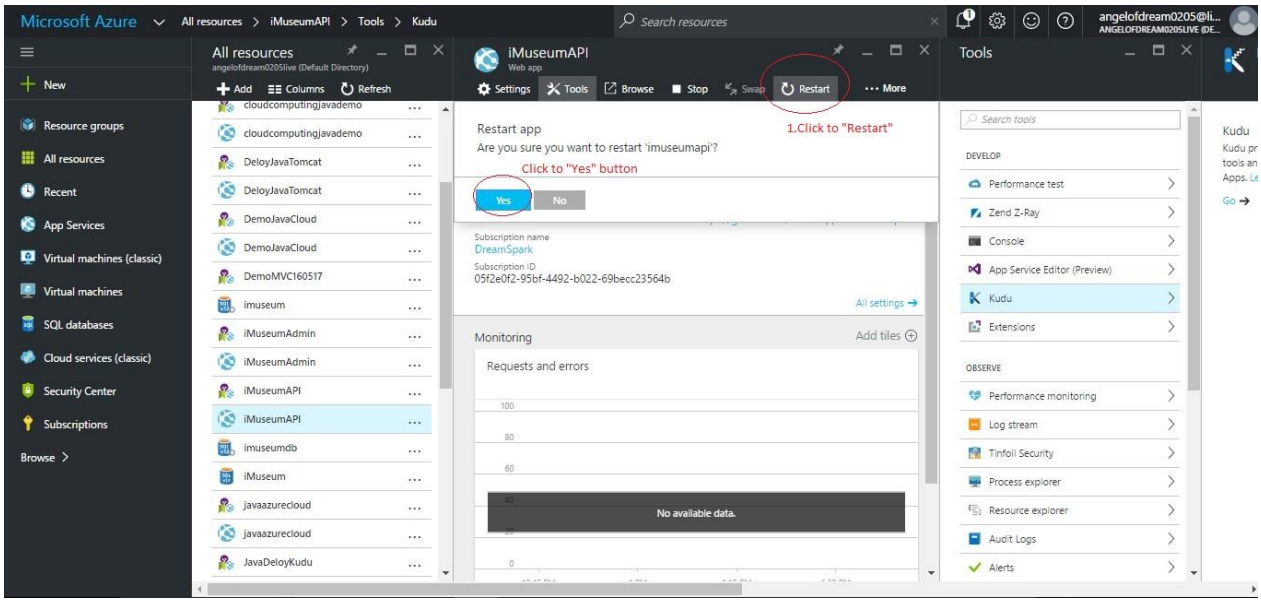
Navigate to *home\site\wwwroot\bin\apache-tomcat-8.0.33\webapps.*



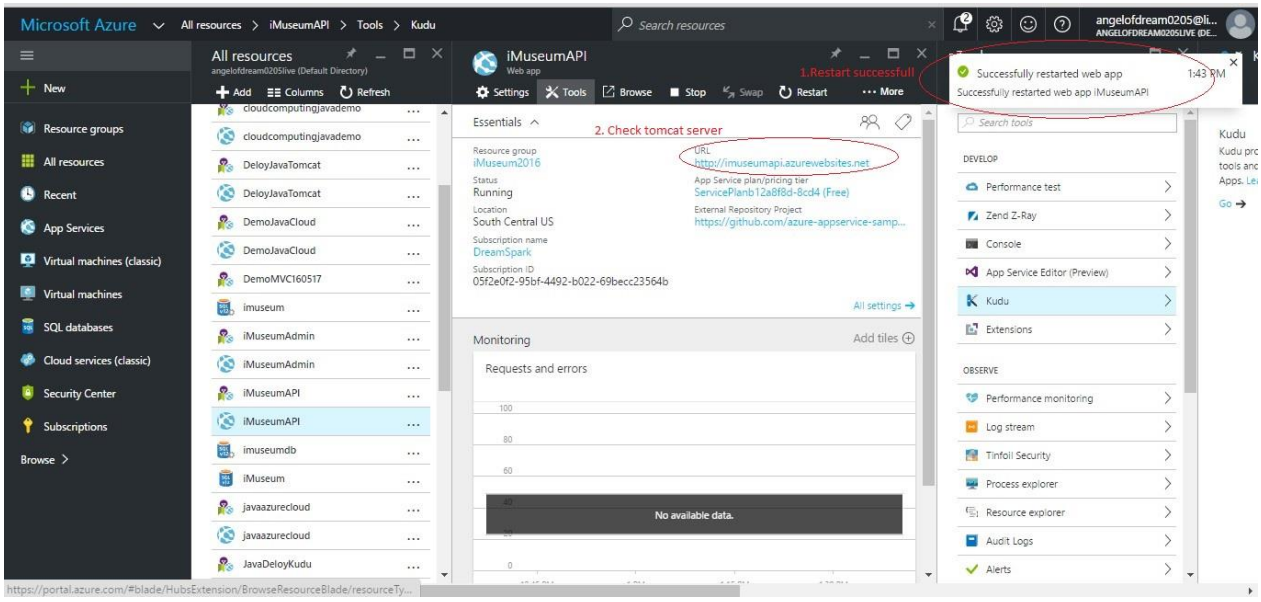
Wait execute



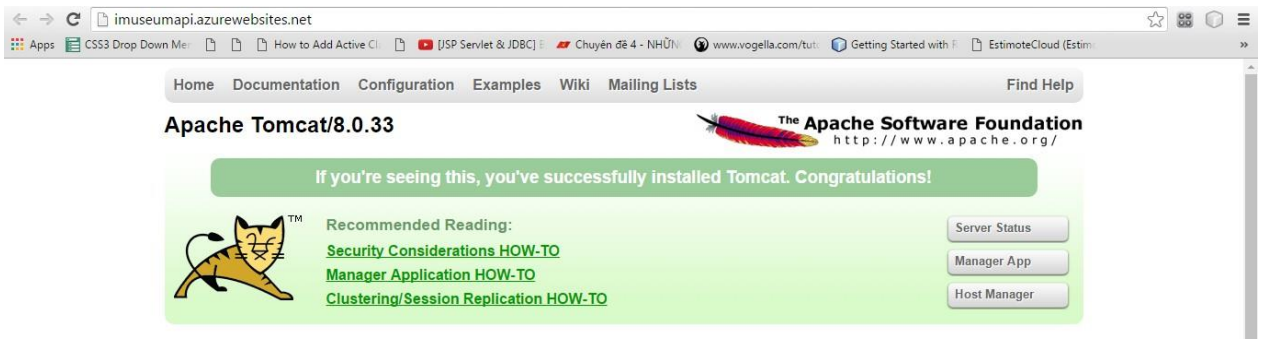
Restart server



Check Tomcat Server



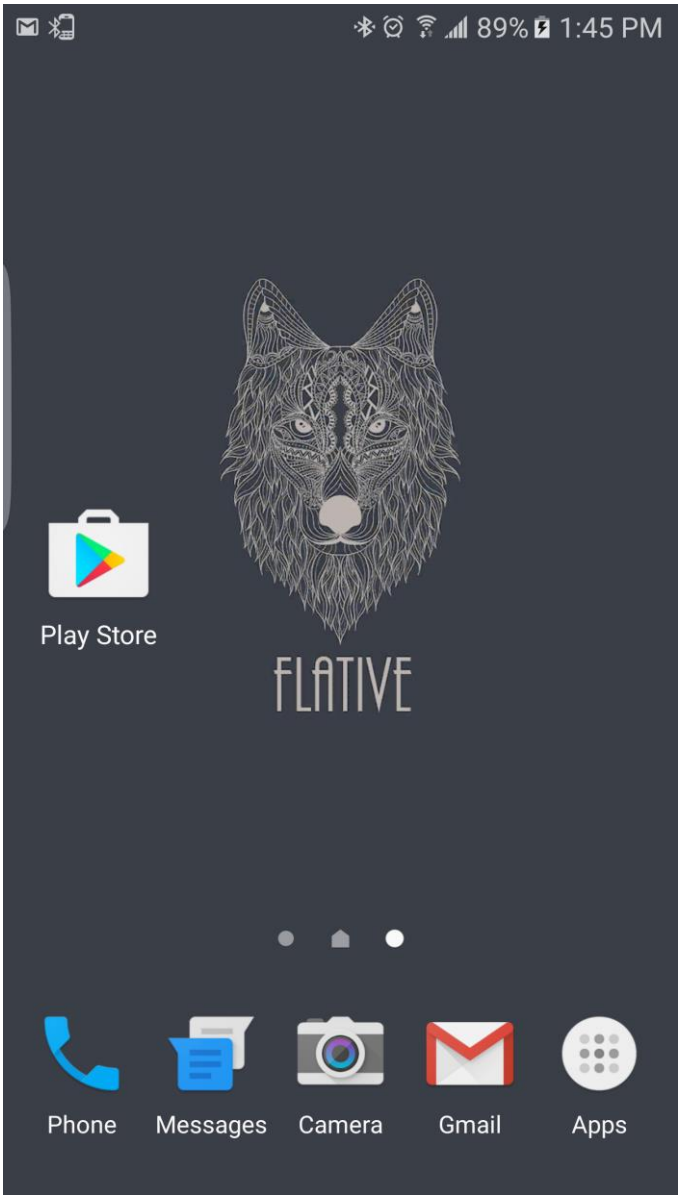
Tomcat Server Successful



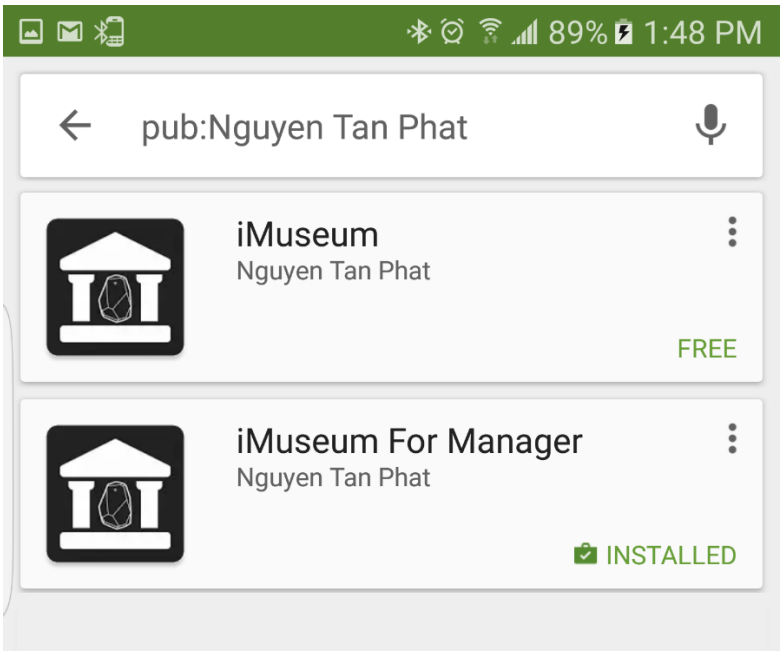
Application deploy successful

1.3Deployment of Mobile Application

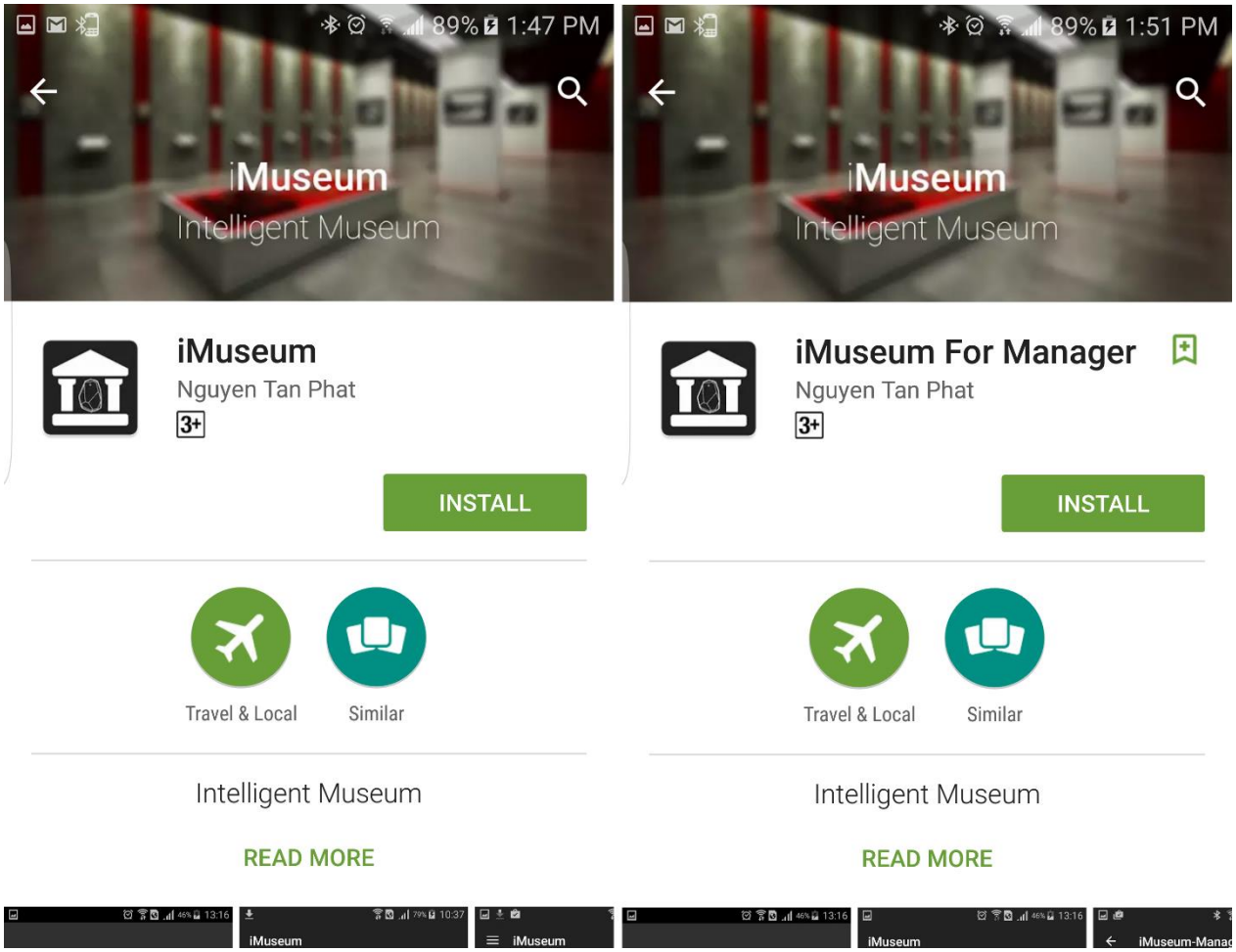
On mobile device, open *Google Play Store* application.



Search ***"pub:Nguyen Tan Phat"***



Install *"iMuseum"* (for Visitor) or *"iMuseum For Manager"* (for Admin/Expert/Staff).



2. User's Guide

2.1Web application

2.1.1 Admin – Expert Login

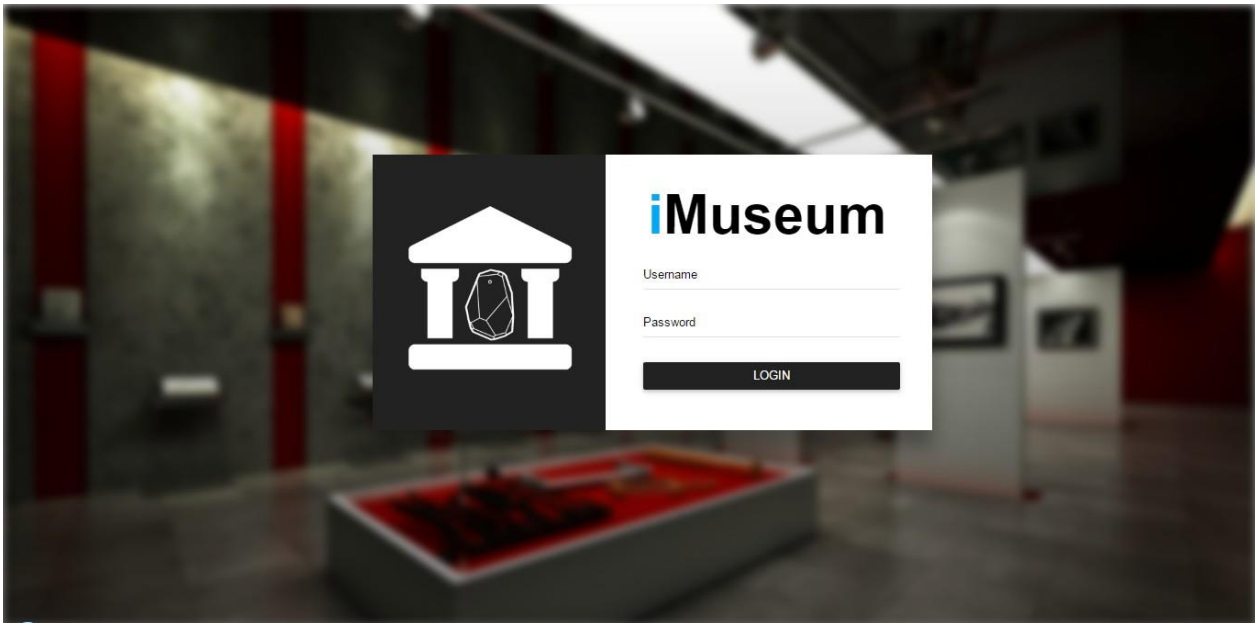


Figure 89. Admin - Staff Login Page

Step	Description
1	Enter "Username" và "Password" (E.g: Username: admin. Password: 12345678)
2	Click "Login"

Table 147. Login Step

2.1.2 Expert

2.1.2.1 View list of historic items

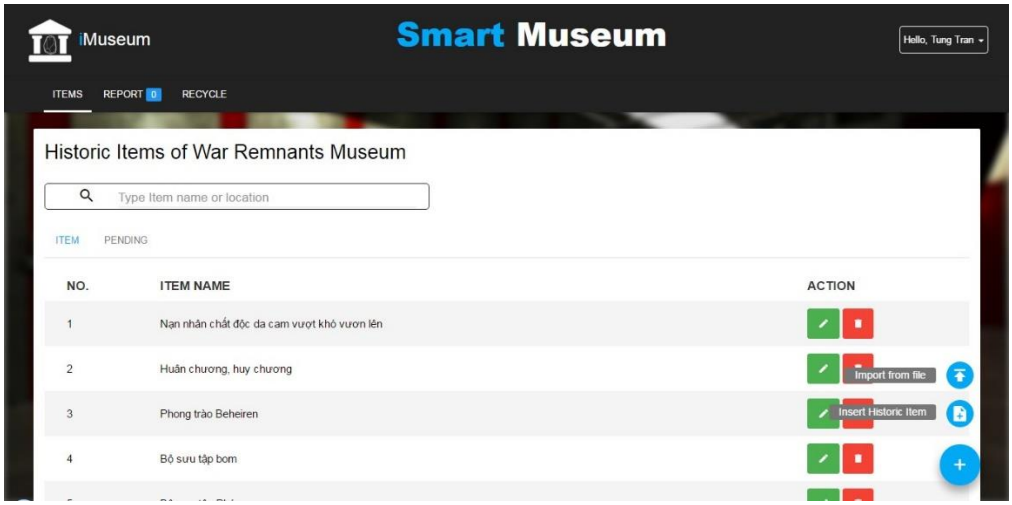


Figure 90. View list of historic items

Step	Description
1	Click “ITEMS” on menu
2	Click “ITEM” tab
3	Show list historic items

Table 148. View list historic items

2.1.2.2 Update details of historic item

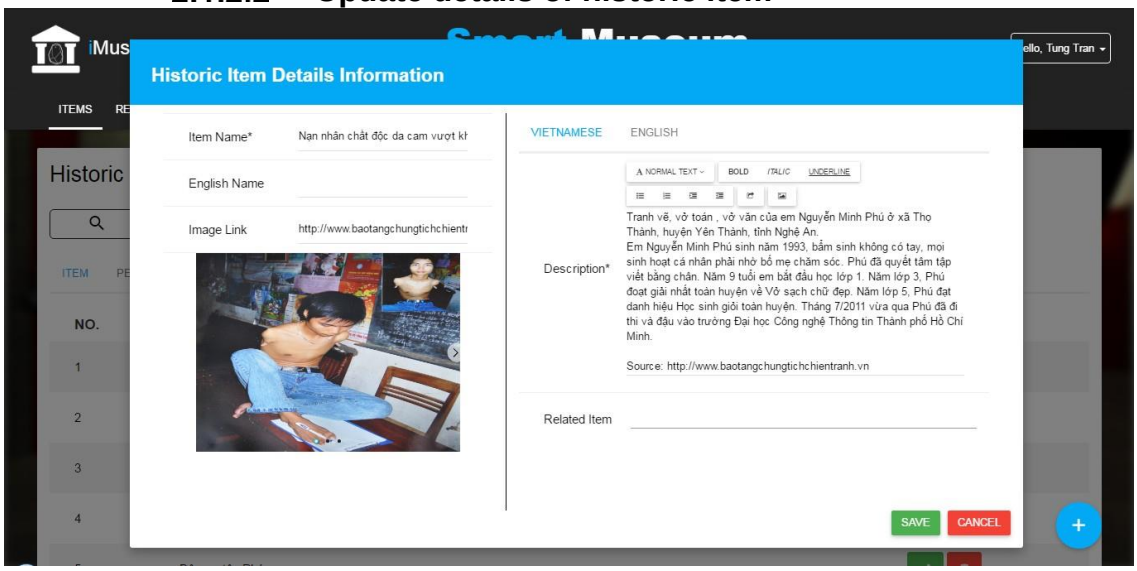


Figure 91. Update details of historic item

Step	Description
1	Click button “ ” on row
2	update details historic item
3	Click button “SAVE”

Table 149. Update details of historic items

2.1.2.3 Insert historic item

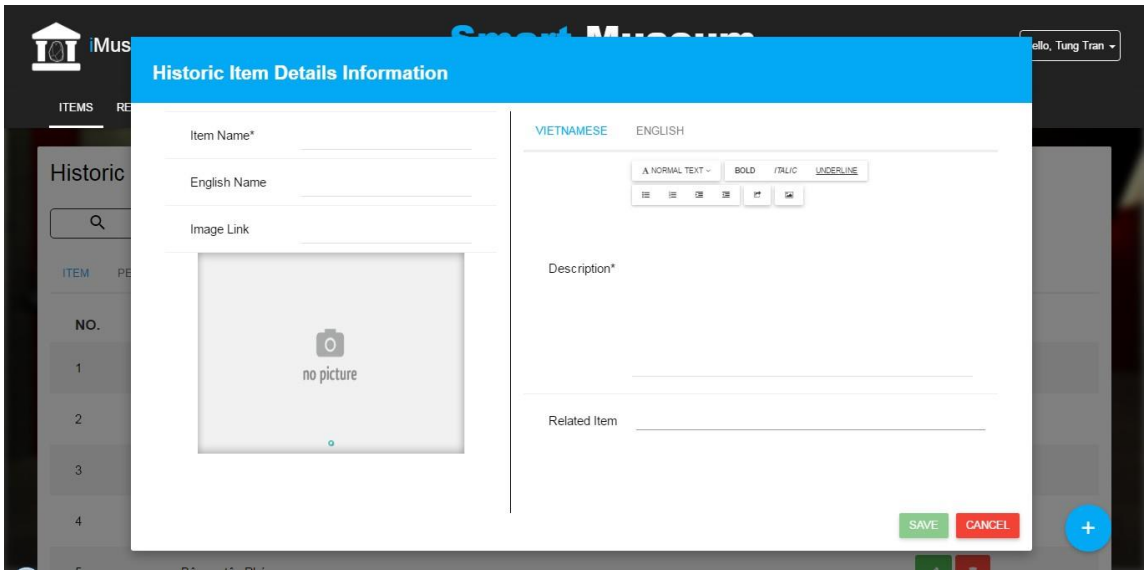


Figure 92. Update details of historic item

Step	Description
1	Click button “Insert Historic Item”
2	Enter details historic item
3	Click button “SAVE”

Table 150. Update details of historic items

2.1.2.4 Delete historic item

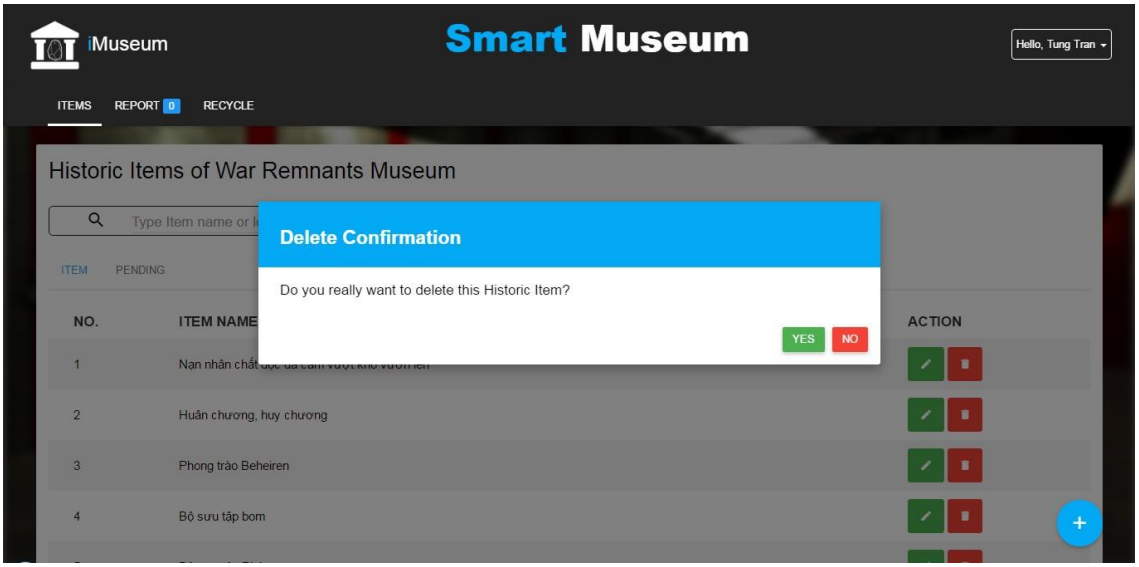


Figure 93. Delete historic item

Step	Description
1	Click button “” on row
2	Click button “YES”

Table 151. Delete historic items

2.1.2.5 Approved historic items

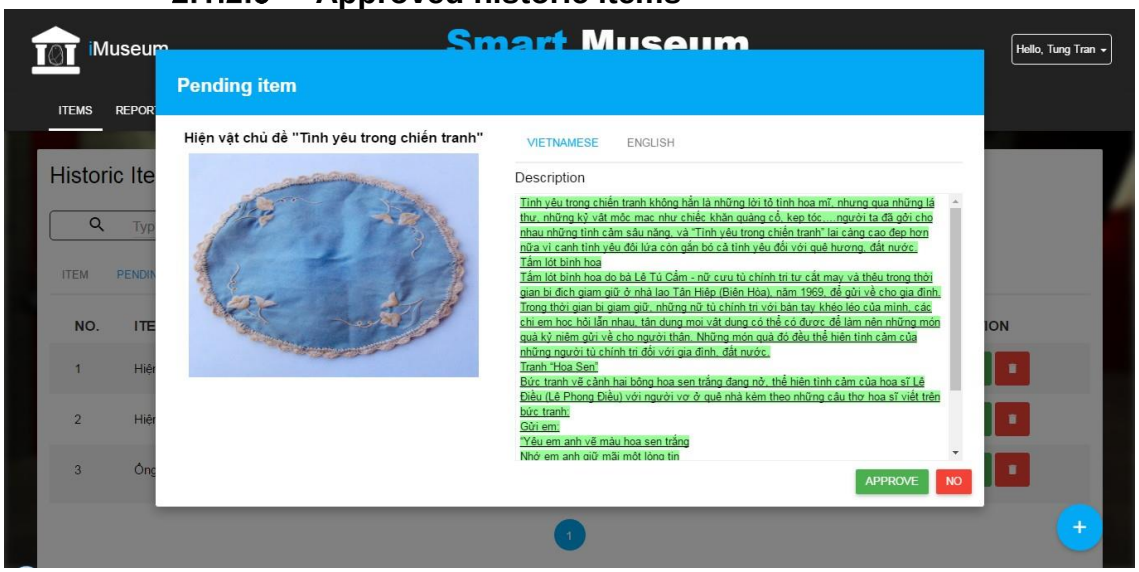


Figure 94. Approved pending item

Step	Description
1	Click button “ ” on row
2	Click button “APPROVE”

Table 152. Approved pending item

2.1.2.6 View list of reports

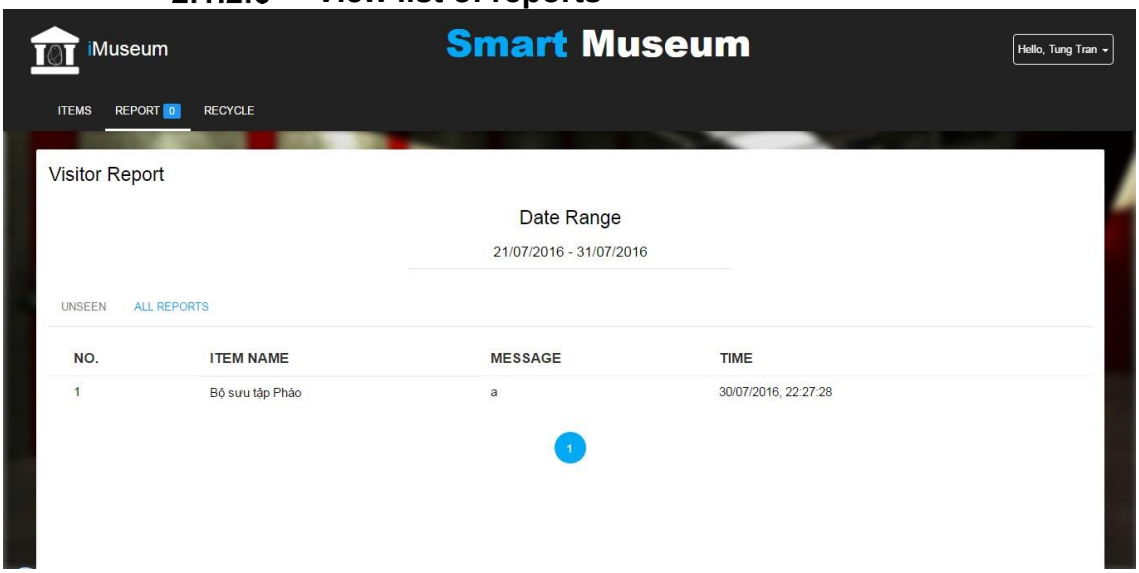


Figure 95. View list of reports

Step	Description
1	Click “REPORT” on menu
2	Choose date range
3	Show list reports

Table 153. View list reports

2.1.2.7 Check report

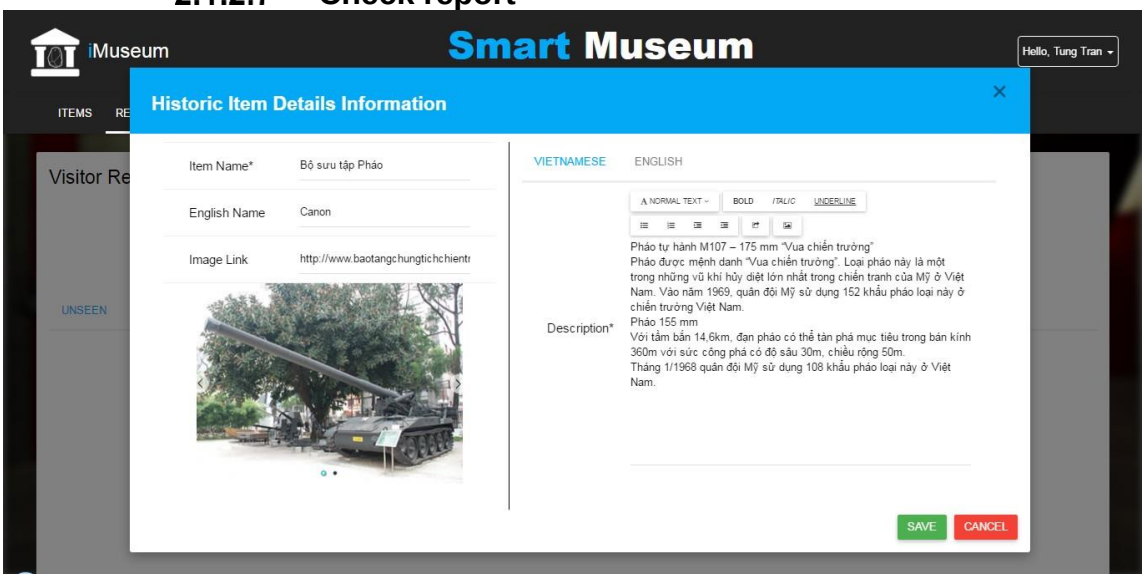


Figure 96. Check report

Step	Description
1	Click name of report
2	update details historic item
3	Click button “SAVE”

Table 154. Check report

2.1.2.8 Recover item in recycle bin

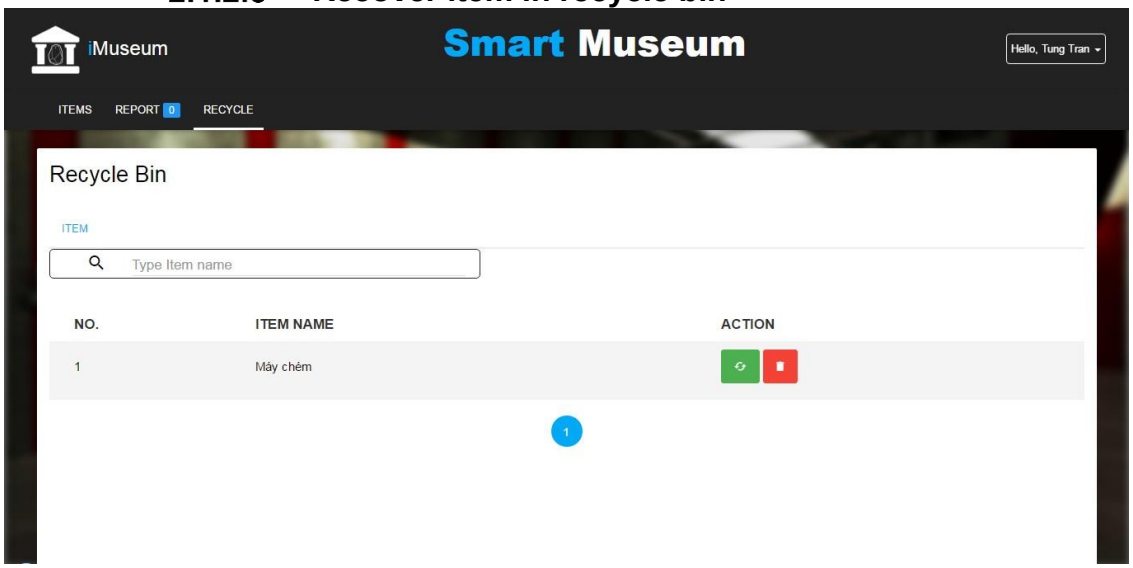


Figure 97. Recover item in recycle bin

Step	Description
1	Click button “ ” on row

Table 155. Recover item in recycle bin

2.1.3 Admin

2.1.3.1 View list of movement log

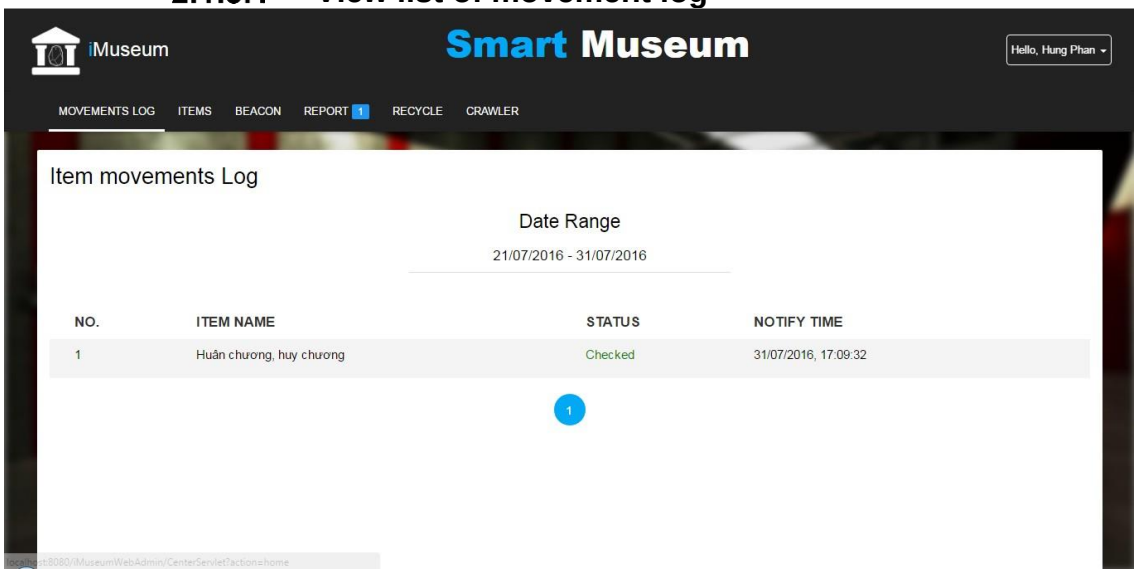


Figure 98. View list of movement log

Step	Description
1	Click “MOVEMENTS LOG” on menu
2	Choose date range
3	Show list movement log

Table 156. View list of movement log

2.1.3.2 View list of historic items

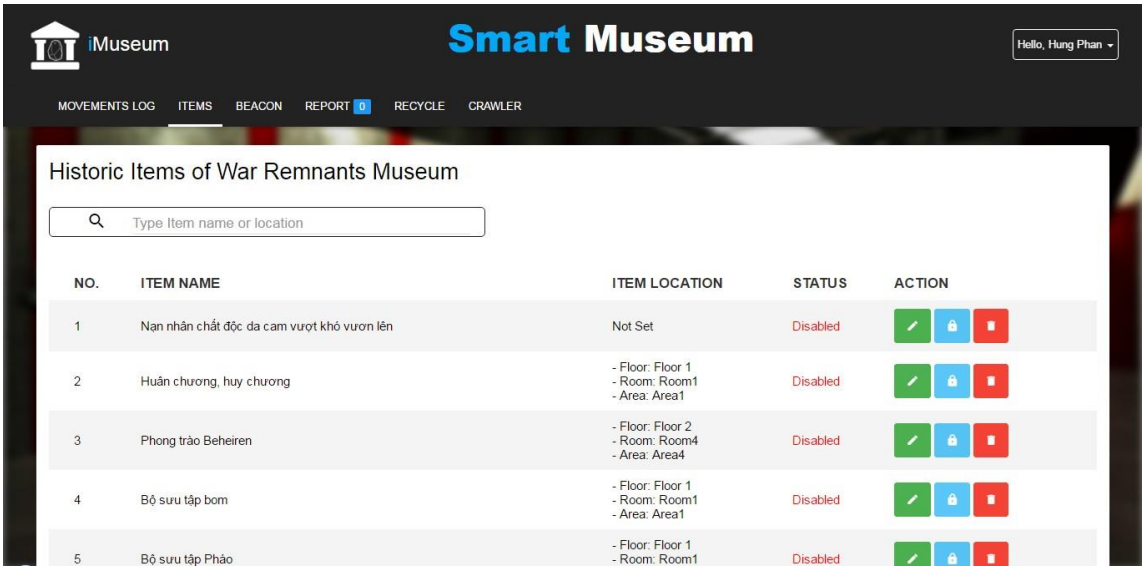


Figure 99. View list of historic items

Step	Description
1	Click “ITEMS” on menu
2	Show list historic items

Table 157. View list historic items

2.1.3.3 Delete historic item

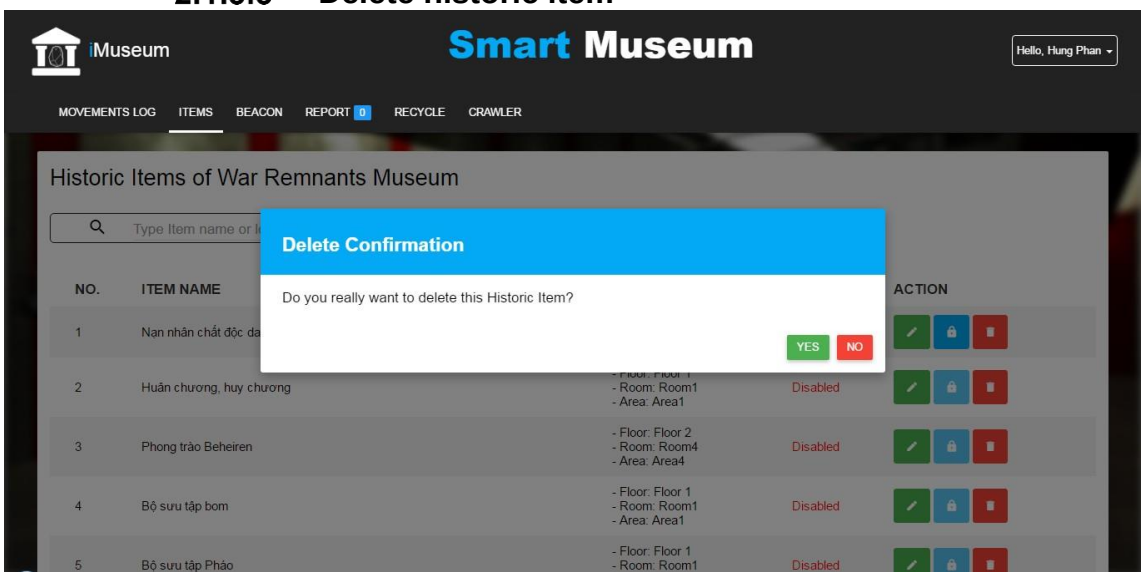


Figure 100. Delete historic item

Step	Description
1	Click button “ ” on row
2	Click button “YES”

Table 158. Delete historic item

2.1.3.4 View list of beacons

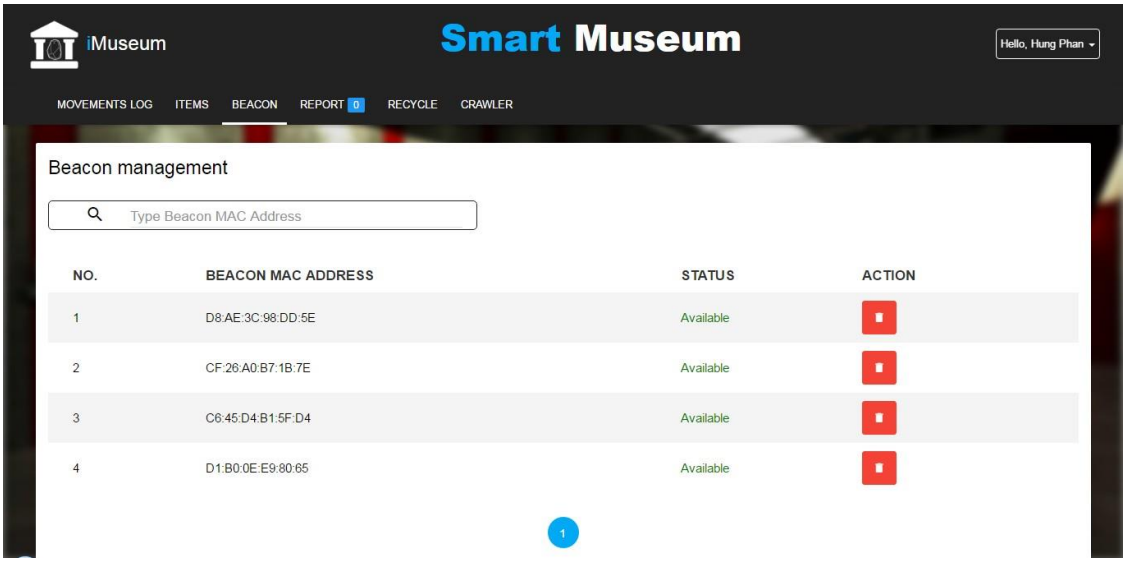


Figure 101. View list of beacons

Step	Description
1	Click “BEACON” on menu
2	Show list historic items

Table 159. View list beacons

2.1.3.5 Delete beacon

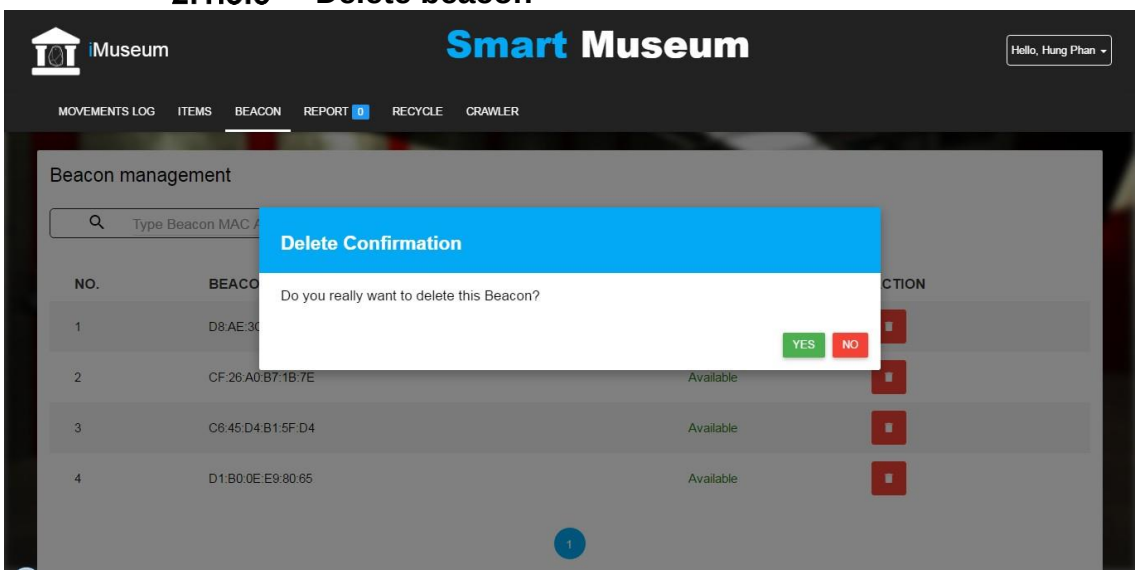


Figure 102. Delete beacon

Step	Description
1	Click button “ ” on row
2	Click button “YES”

Table 160. Delete beacon

2.1.3.6 View list of reports

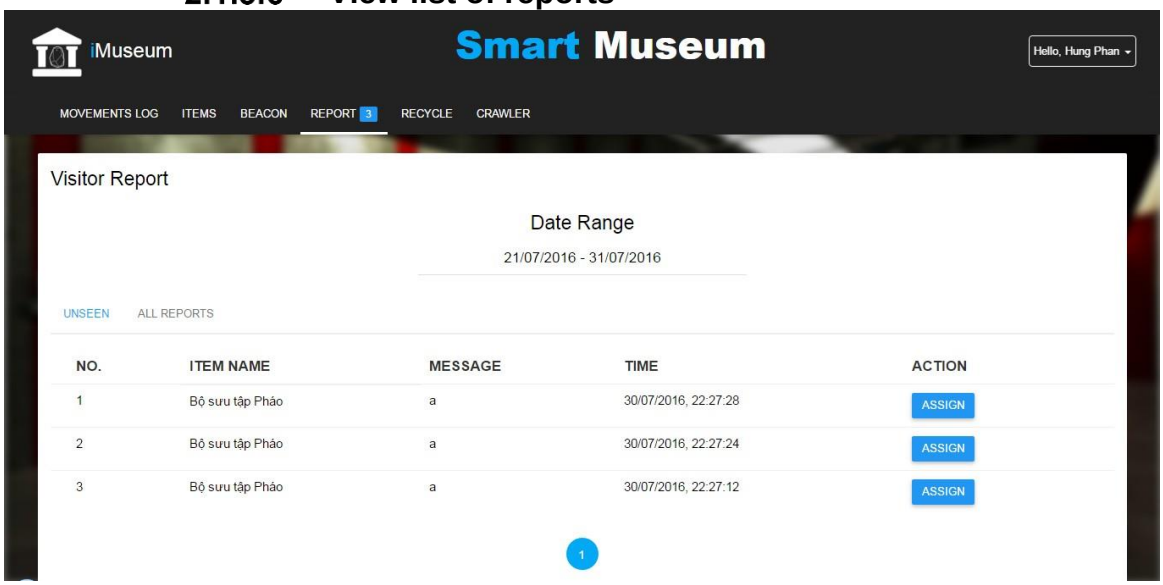


Figure 103. View list of reports

Step	Description
1	Click “REPORT” on menu
2	Choose date range
3	Show list reports

Table 161. View list reports

2.1.3.7 Check report

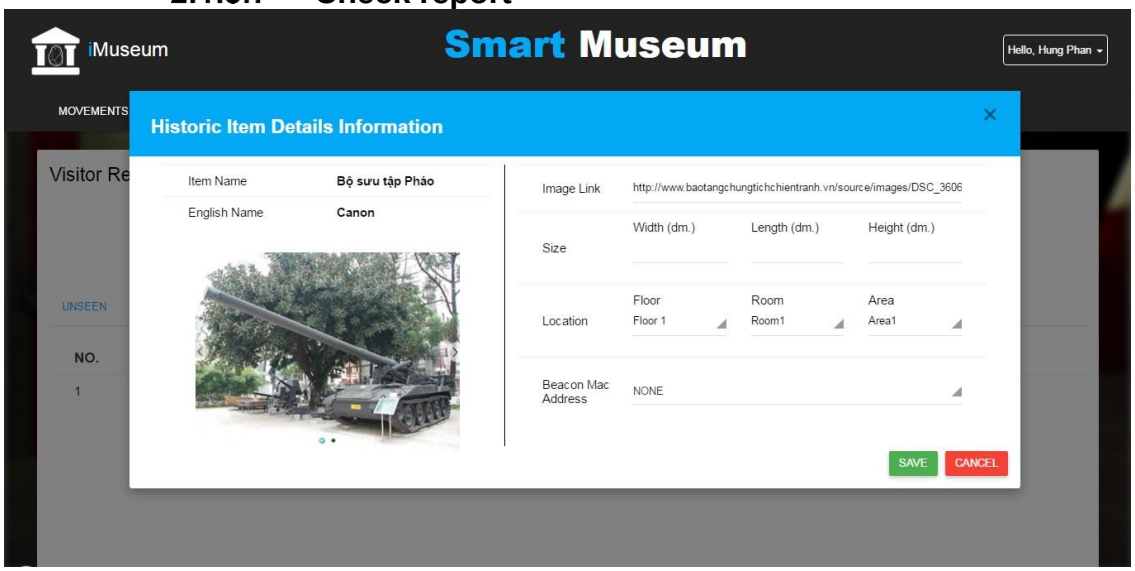


Figure 104. Check report

Step	Description
1	Click name of report
2	Update beacon and location
3	Click button “SAVE”

Table 162. Check report

2.1.3.8 Assign report

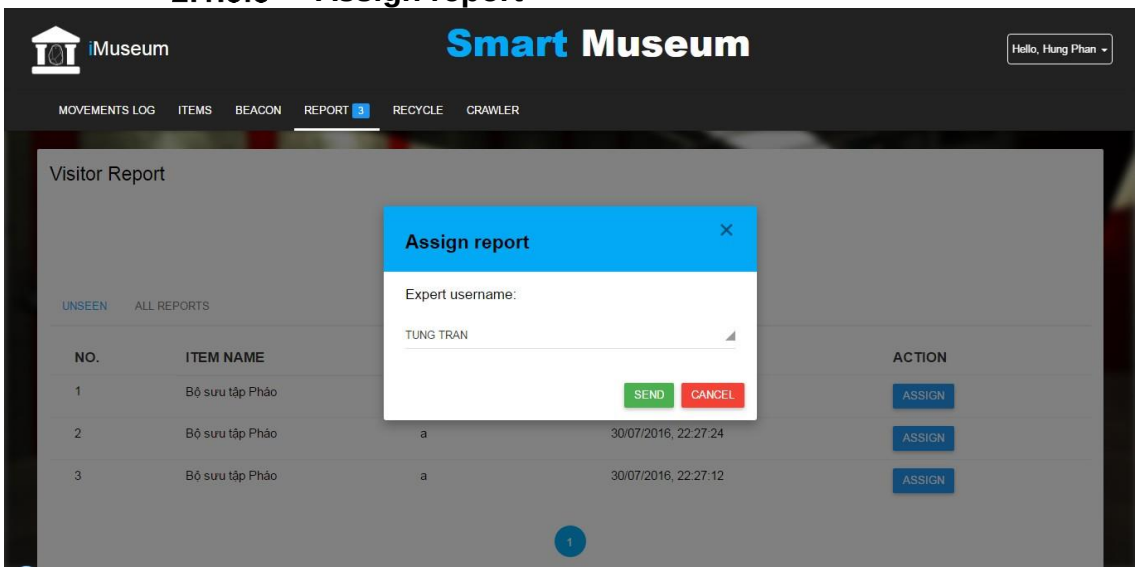


Figure 105. Assign report

Step	Description
1	Click button “ASSIGN”
2	Choose name expert
3	Click button “SEND”

Table 163. Assign report

2.1.3.9 Recover item in recycle bin

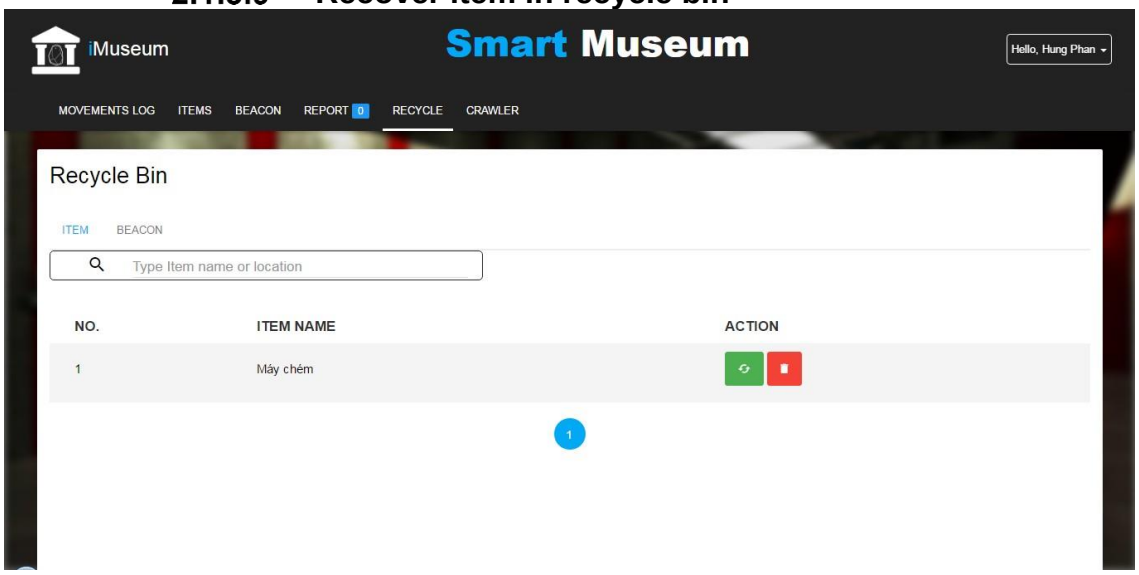


Figure 106. Recover item in recycle bin

Step	Description
1	Click button “ ” on row

Table 164. Recover **item in recycle bin**

2.1.3.10 Recover beacon in recycle bin

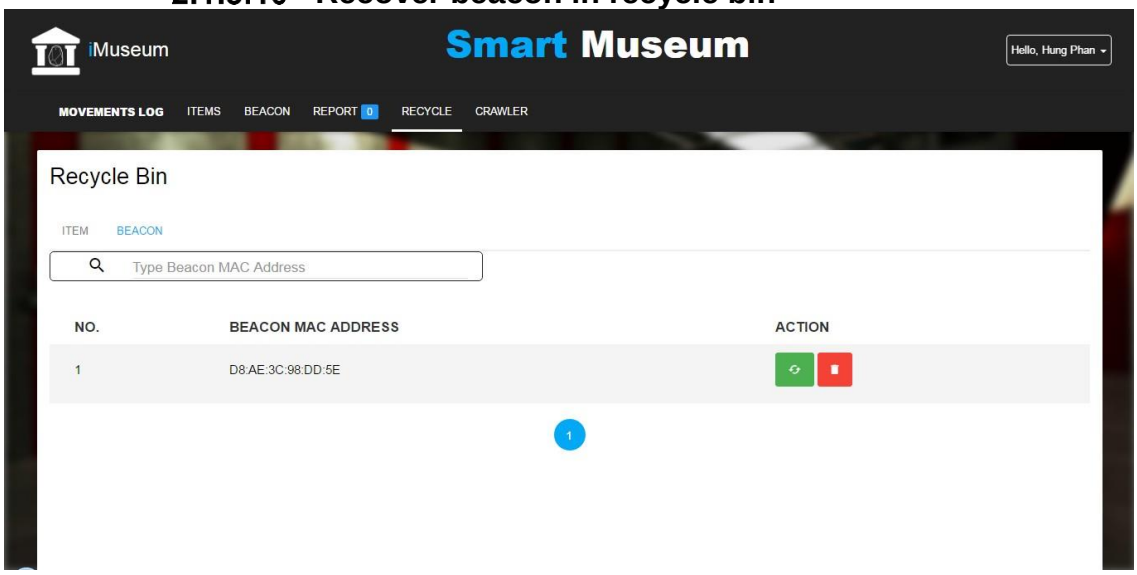


Figure 107. **Recover beacon in recycle bin**

Step	Description
1	Click button “ ” on row

Table 165. Recover **beacon in recycle bin**

2.2Mobile application

2.2.1 Admin – Staff Login

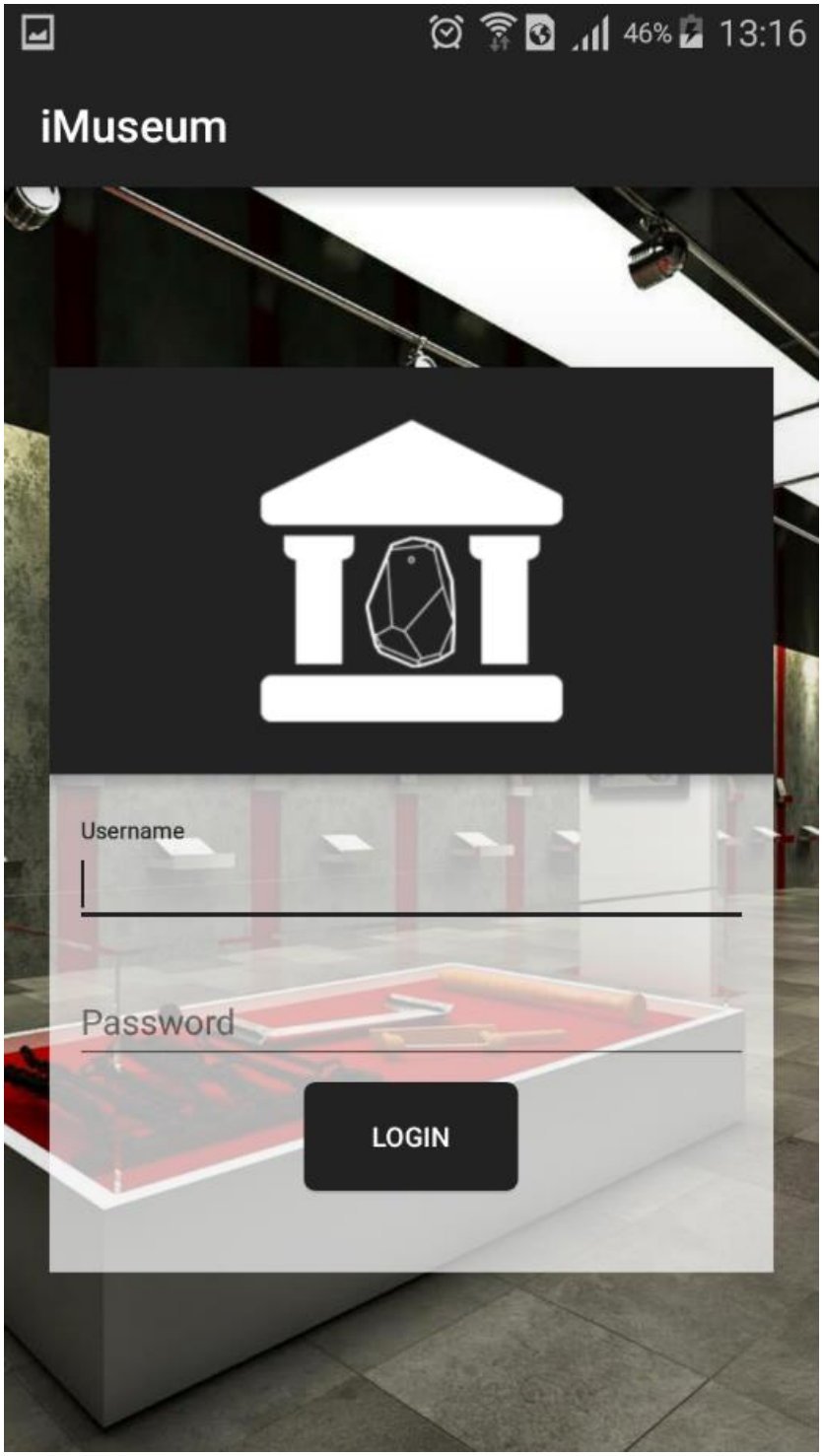


Figure 108. Admin - Staff Login

Step	Description
1	Enter “Username” và “Password” (E.g: Username: admin. Password: 12345678)
2	Click button “Login”

Table 166. Login Step

2.2.2 Admin

2.2.2.1 Map beacon

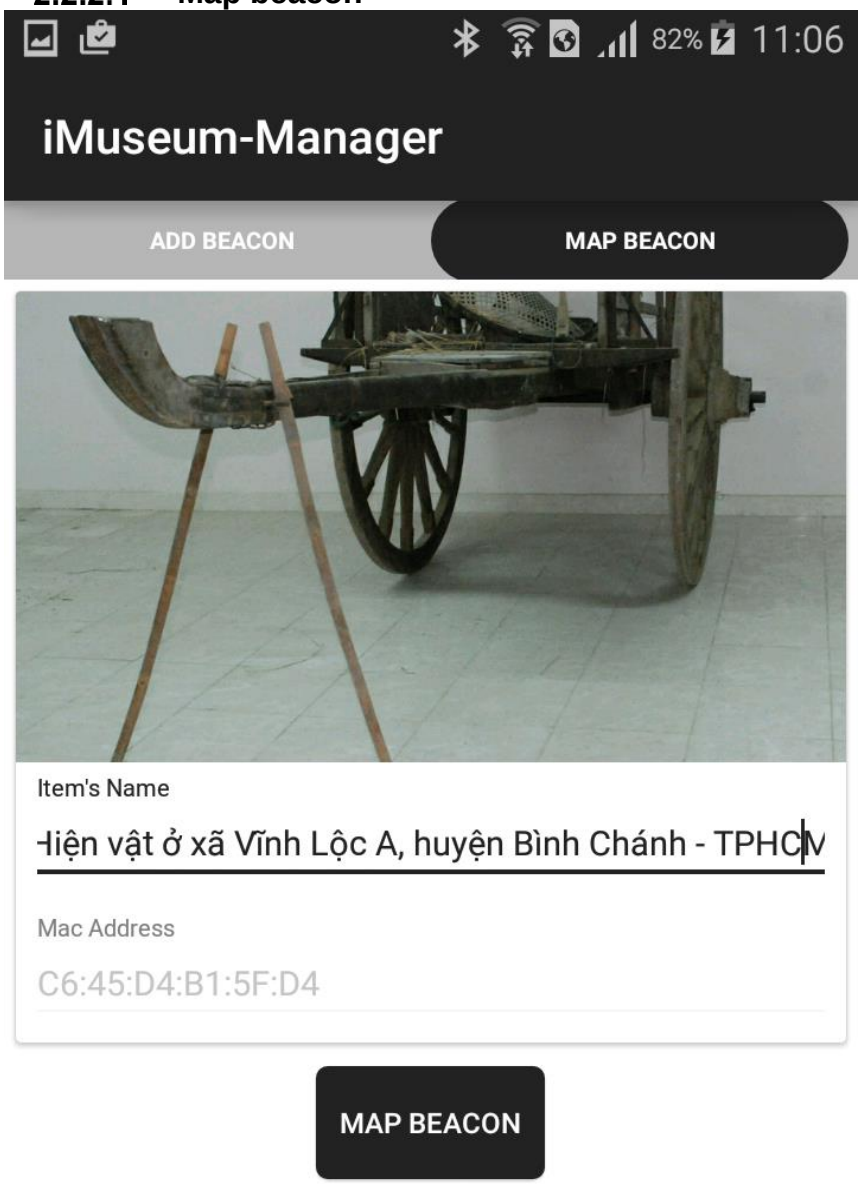


Figure 109. Map beacon

Step	Description
1	Click un-map beacon
2	Enter name historic item
3	Click button “Map beacon”

Table 167. Map beacon

2.2.3 Staff

2.2.3.1 View location item move

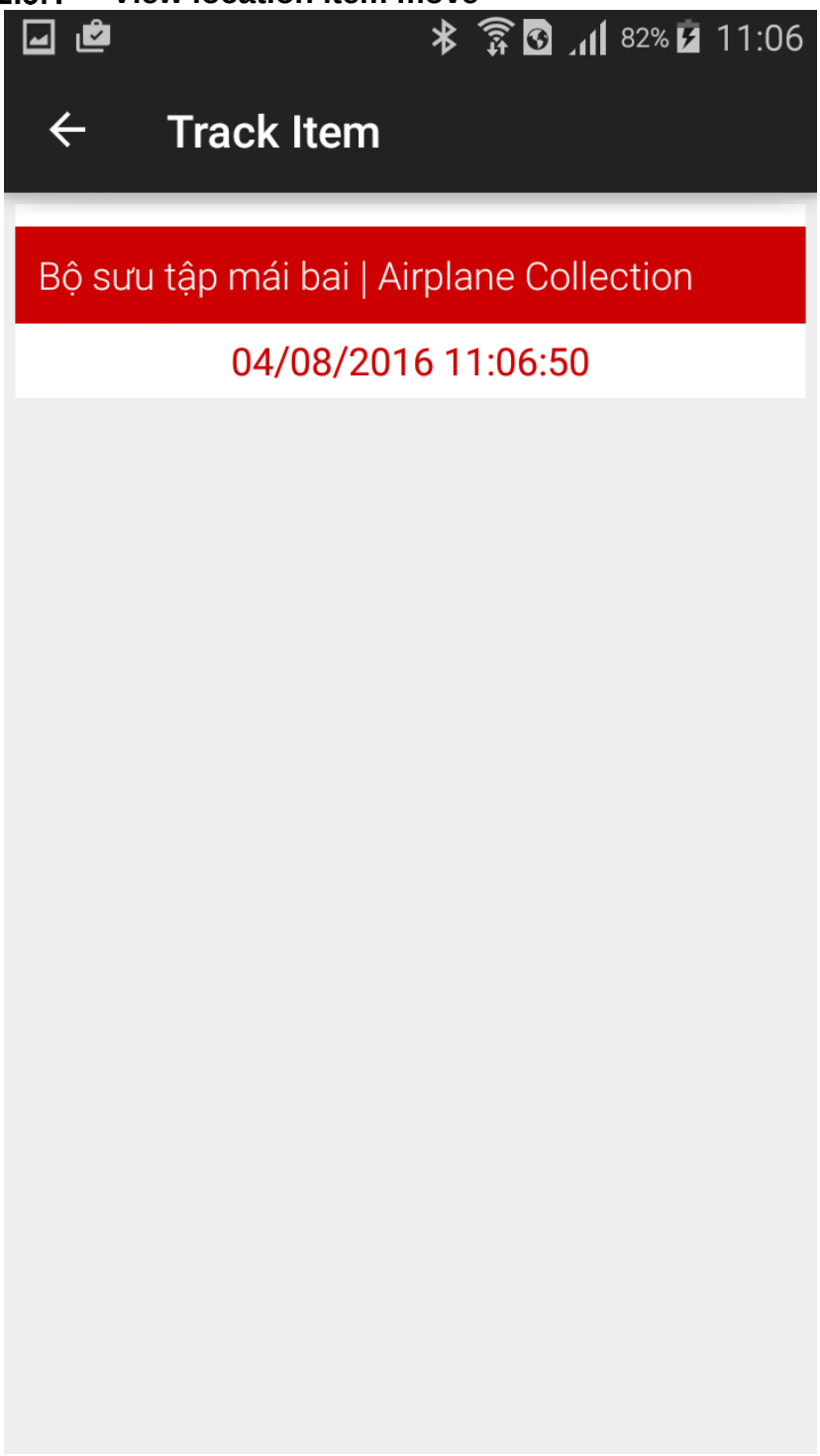


Figure 110. View location item move

Step	Description
1	Click name item
2	Show location item move

Table 168. View location item move

2.2.4 Visitor

2.2.4.1 View list item in floor

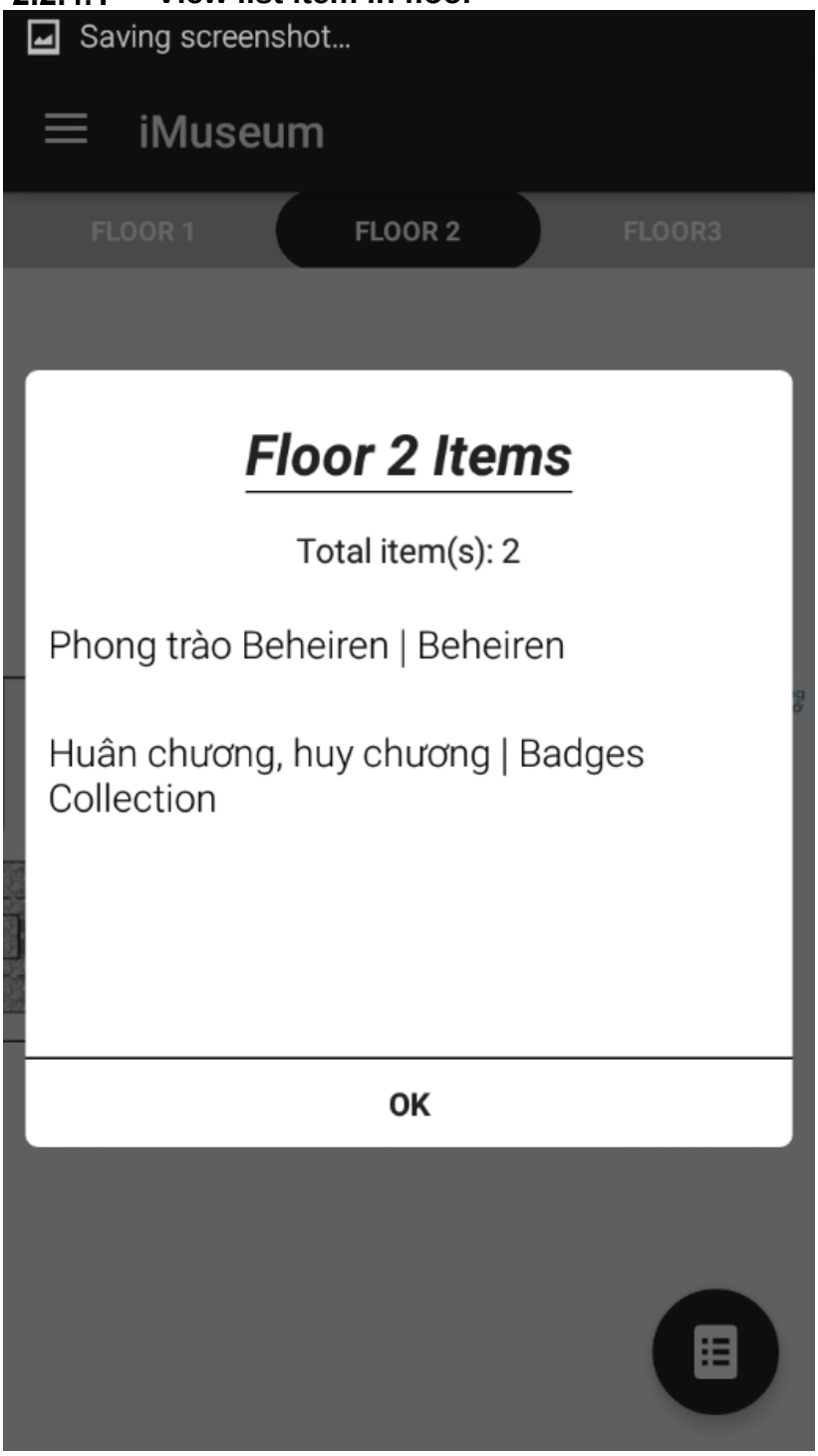


Figure 111. View list item in floor

Step	Description
1	Click button “museum map” on navigation view
2	Choose floor tap
3	Click button “”
4	Show list item in floor

Table 169. View list item in floor

2.2.4.2 View list item near you

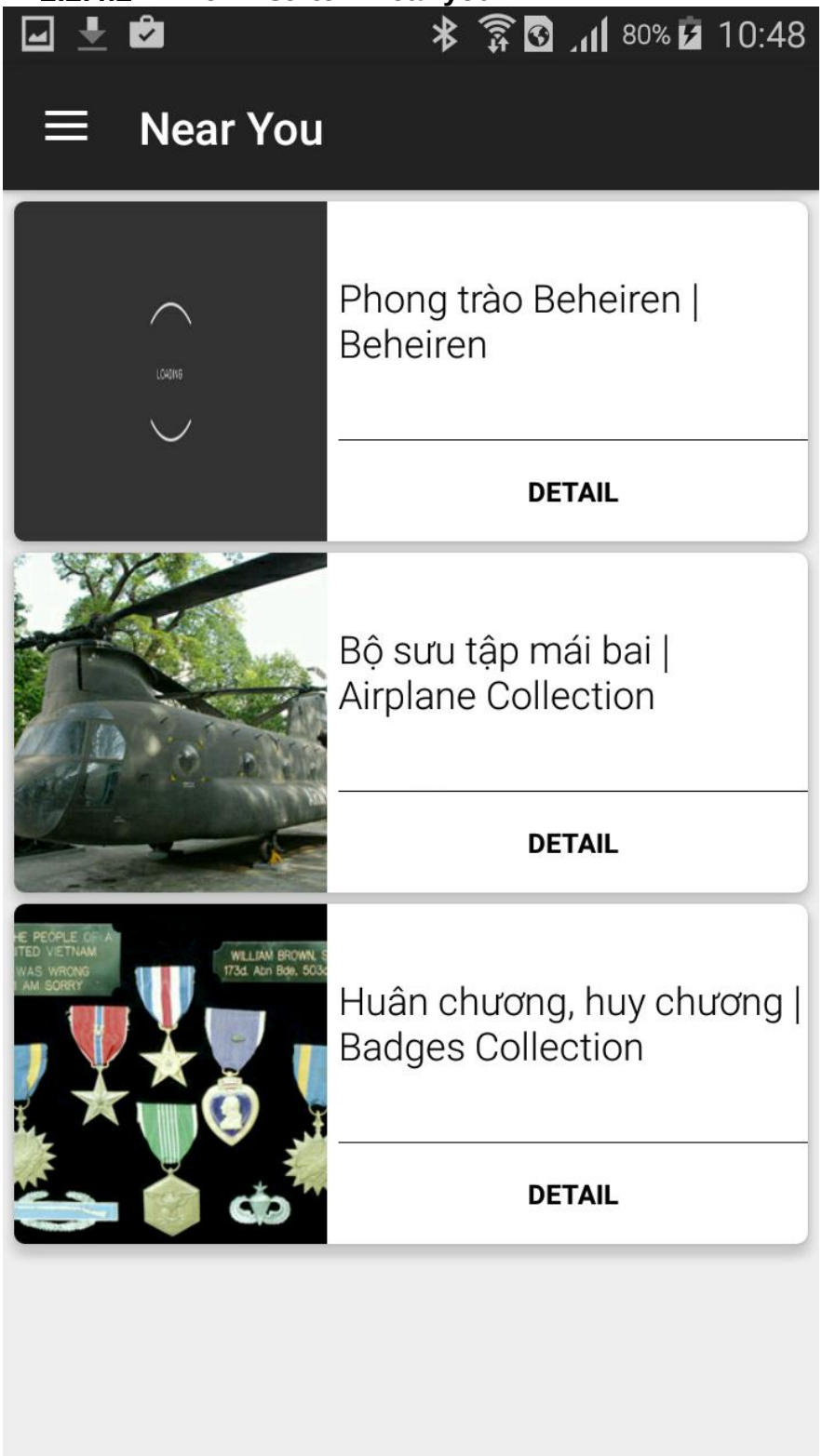


Figure 112. View list item near you

Step	Description
1	Click button “near you” on navigation view
2	Show list item near you

Table 170. View list item near you

2.2.4.3 View item details

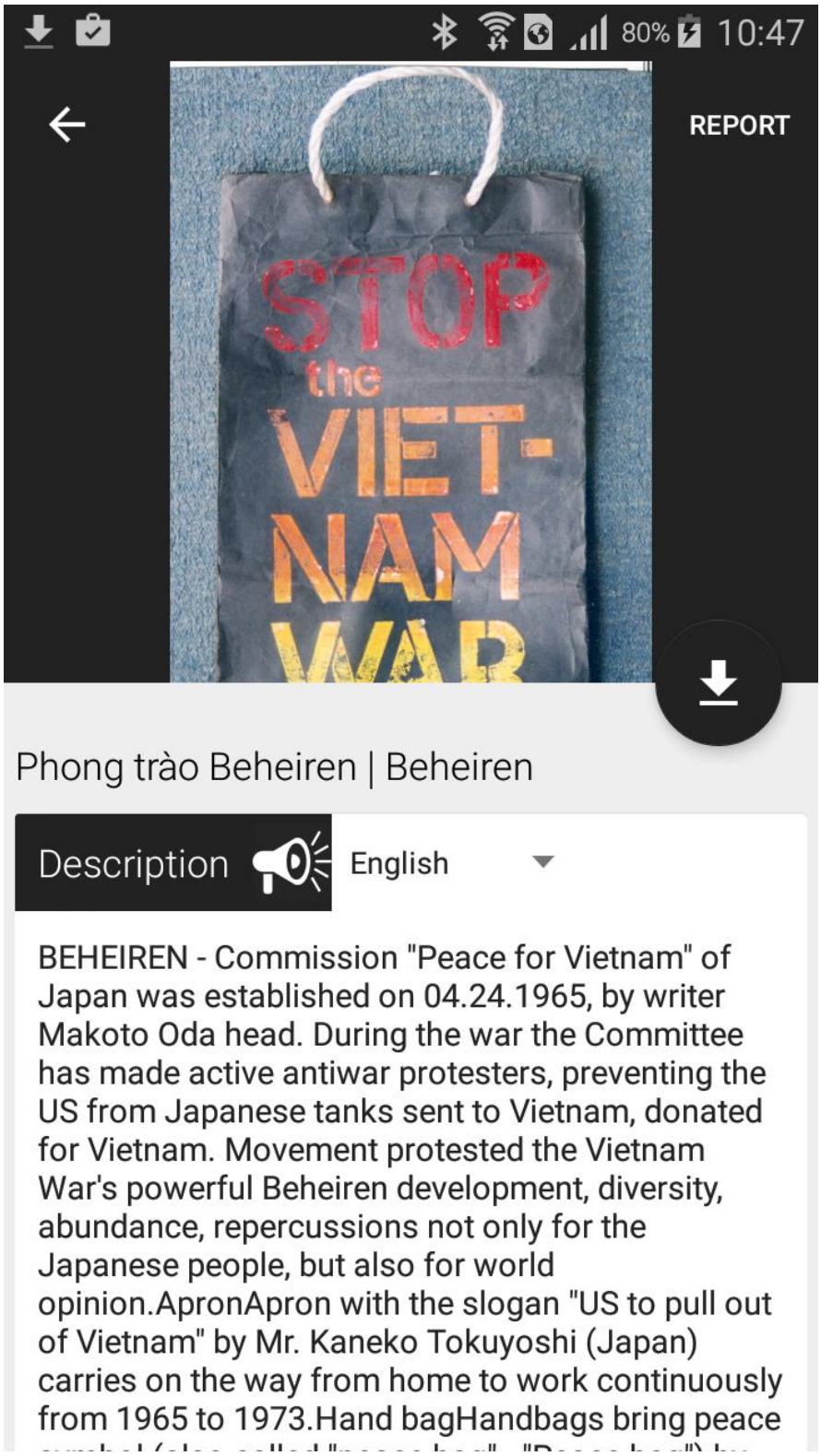


Figure 113. View item details

Step	Description
1	Click button “Details” on item
2	Show item details

Table 171. View item details

2.2.4.4 Report Item

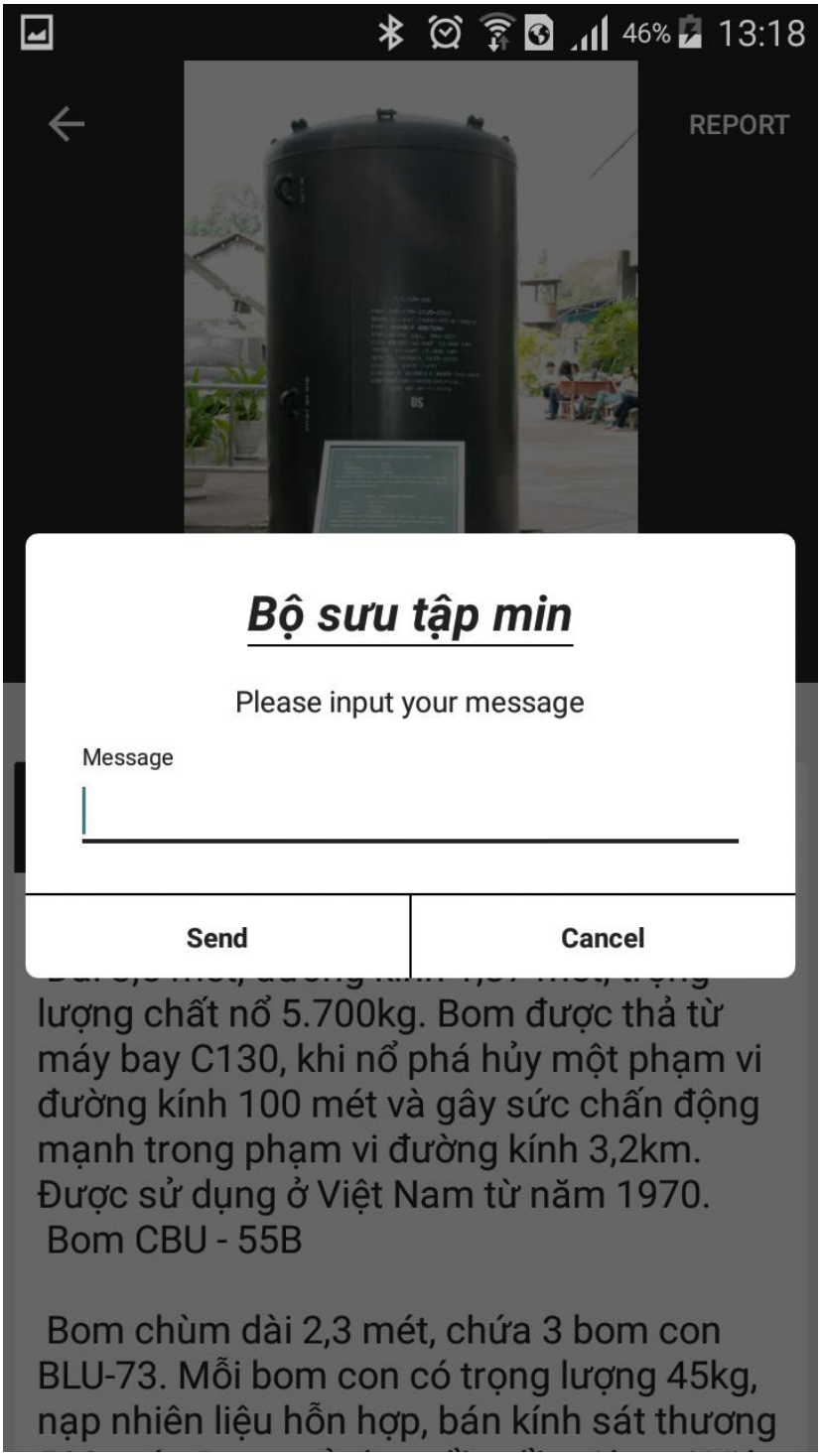


Figure 114. Report item

Step	Description
1	Click button “Details” on item
2	Click button “Report”
3	Enter message
4	Click button “Send”

Table 172. Report item

2.2.4.5 Add favorite item

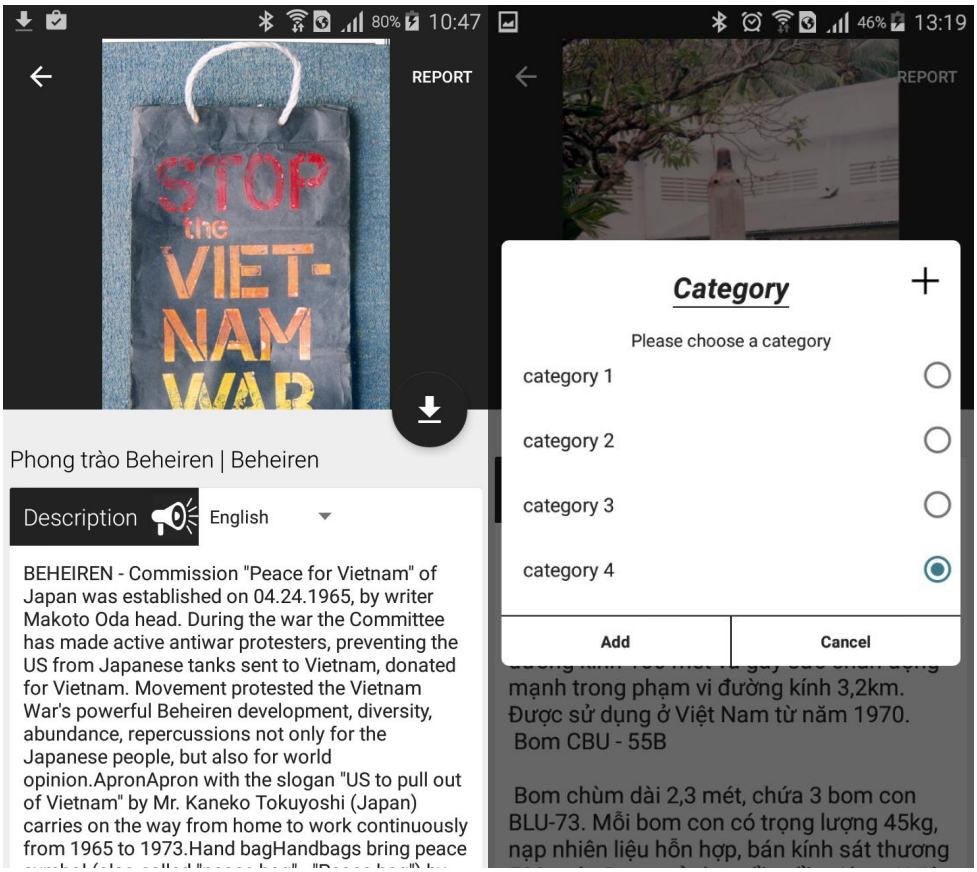


Figure 115. Add favorite item

Step	Description
1	Click button “Details” on item
2	Click button “Report”
3	Click button “”
4	Choose category
5	Click button “Add”

Table 173. Add favorite item

2.2.4.6 Read item's description

Phong trào Beheiren | Beheiren

Description English

BEHEIREN - Commission "Peace for Vietnam" of Japan was established on 04.24.1965, by writer Makoto Oda head. During the war the Committee has made active antiwar protesters, preventing the US from Japanese tanks sent to Vietnam, donated for Vietnam. Movement protested the Vietnam War's powerful Beheiren development, diversity, abundance, repercussions not only for the Japanese people, but also for world opinion.ApronApron with the slogan "US to pull out of Vietnam" by Mr. Kaneko Tokuyoshi (Japan) carries on the way from home to work continuously from 1965 to 1973.Hand bagHandbags bring peace

Figure 116. Read item's description

Step	Description
1	Click button "Details" on item
2	Click button "Report"
3	Click button ""

Table 174. Read item's description

2.2.4.7 Detect related item location

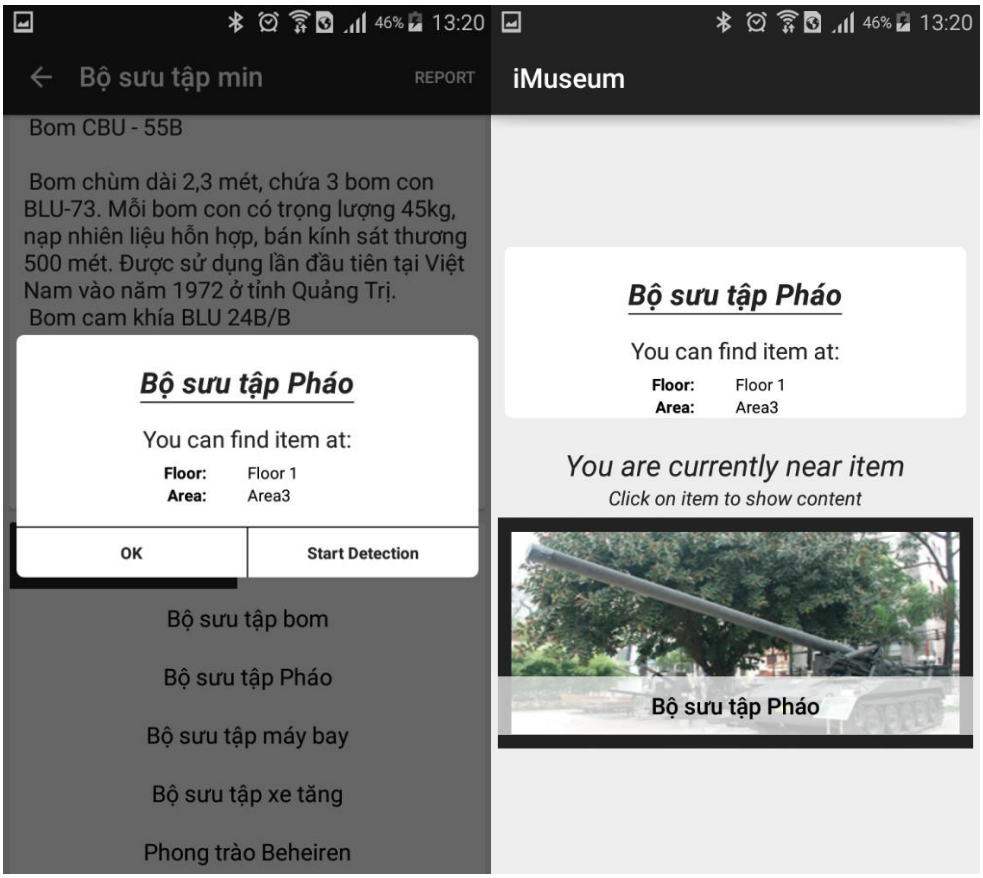


Figure 117. Detect related item location

Step	Description
1	Click button "Details" on item
2	Click button "Report"
3	Click name of related item
4	Click button "Start Detection"
5	Detect related item

Table 175. Detect related item location

2.2.4.8 View list favorite item

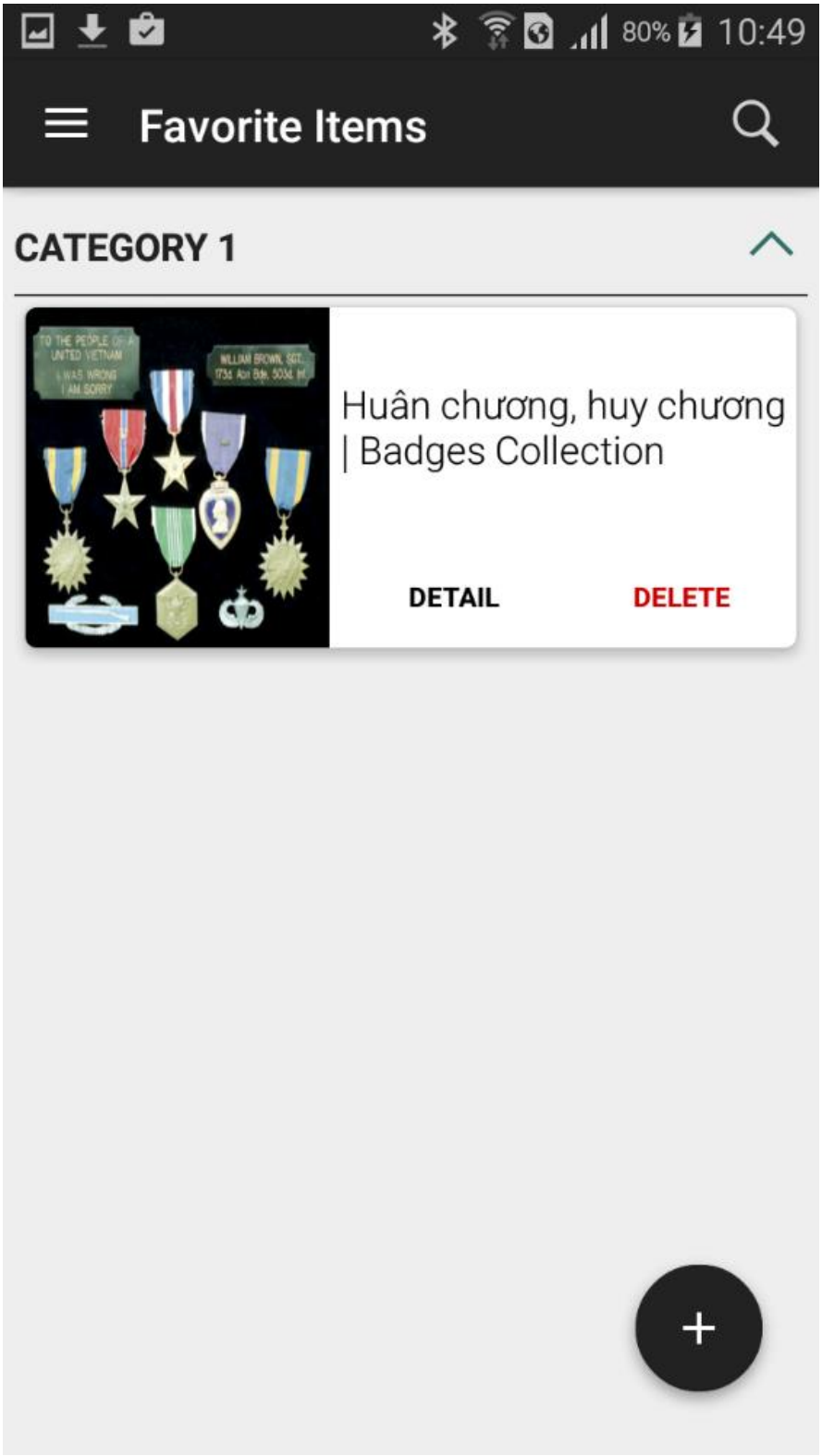


Figure 118. View list favorite item

Step	Description
1	Click button “Favorite item” on navigation view
2	Show list favorite item

Table 176. View list favorite item

G. Appendix

1. SOFTWARE ENGINEERING 8th Edition, by Ian Sommerville
2. SOFTWARE ENGINEERING 9th Edition, by Ian Sommerville
3. Code Conventions for the Java TM Programming Language, by Sun Microsystems, rev April 20, 1999.
4. Android Developer Guide – Application Fundamentals http://developer.android.com/guide/components/fundamentals.html
5. http://www.agiledata.org/essays/evoluationaryDevelopment.html
6. Estimote Beacon Developer http://developer.estimote.com/