 **MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Just Walk Out Library**

|  |  |
| --- | --- |
| **Group 3** | |
| **Group members** | Võ Hồng Hà – Team Leader – SE61394  Nguyễn Tuấn Anh – Team Member – SE61476  Đặng Nhật Thiên – Team Member – SE61357 |
| **Supervisor** | Mr. Kiều Trọng Khánh |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | JWL |

- Ho Chi Minh City, 01/2017 -

**

**CAPSTONE PROJECT REGISTER**

Class: Duration time: from 02/01/2017…. To /2017…..

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

x

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

x

1. Register information for supervisor (if have)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor 1 | Kiều Trọng Khánh |  | khanhkt@fpt.edu.vn | Mr. |

2. Register information for students (if have)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| Student 1 | Võ Hồng Hà | SE61394 | 0949519894 | havhse61394@fpt.edu.vn | Leader |
| Student 2 | Nguyễn Tuấn Anh | SE61476 | 01692536559 | anhntse61476@fpt.edu.vn | Member |
| Student 3 | Đặng Nhật Thiên | SE61357 | 01678785551 | thiendnse61357@fpt.edu.vn | Member |

3. Register content of Capstone Project

(\*) 3.1. Capstone Project name:

English: Just Walk Out Library

Vietnamese: Thư viện Mượn Sách Tự động

Abbreviation:

- JWL

**- Context:**

+ How can you borrow books without having to queue and contact a librarian to check out in libraries?

+ NFC (Near Field Communication) is a form of contactless communication between devices like smartphones or tablets. Contactless communication allows a user to wave the smartphone over an NFC compatible device to send information without needing to touch the devices together or go through multiple steps setting up a connection. utilizes electromagnetic radio fields while technologies such as Bluetooth and Wi-Fi focus on radio transmissions instead (<http://nearfieldcommunication.org/> ). NFC is supported in card or the smartphone

+ QR code (abbreviated from Quick Response Code) is the trademark for a type of [matrix barcode](https://en.wikipedia.org/wiki/Matrix_barcode) (or two-dimensional [barcode](https://en.wikipedia.org/wiki/Barcode)) first designed for the [automotive industry in Japan](https://en.wikipedia.org/wiki/Automotive_industry_in_Japan). A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and [kanji](https://en.wikipedia.org/wiki/Kanji)) to efficiently store data; extensions may also be used

+ RFID stands for Radio-Frequency IDentification. The acronym refers to small electronic devices that consist of a small chip and an antenna. The chip typically is capable of carrying 2,000 bytes of data or less. The RFID device serves the same purpose as a bar code or a magnetic strip on the back of a credit card or ATM card; it provides a unique identifier for that object. And, just as a bar code or magnetic strip must be scanned to get the information, the RFID device must be scanned to retrieve the identifying information (<http://www.technovelgy.com/ct/technology-article.asp> )

**- Building the application provides following services**

* Combine smartphone devices and NFC, QR code, RFID, …
* Building the application that determines the borrowed books automatically when users go out the library
* ...

**- Simulator**

* The borrower uses the application on a smartphone, which supported NFC or QR Code, to emulator.
* The borrower takes any books to his/her bag
* The borrower uses the application to emulator at the exit entrance, the application will show a list of borrowed books
* …

(\*) 3.2. Main proposal content (including result and product)

1. Theory and practice (document):

* Student should apply the software development process and the UML
* Software artifacts include User Requirement, Software Requirement Specification, Architecture Design, Detail Design, System Implementation and Testing Document, Installation Guide, sources code, and deployable software packages
* 3 tiers should be applied
* Server side technique:
  + Database design, OOA, OOD, OOP, MVC, Java or .Net technology, Restful API, …
* Client side technique
  + HTML5, CSS, JavaScript, jQuery, Ajax, Android**, iOS...**
* Communication technique
  + Exchange information and transfer data in effective in networks, communicating protocol between mobile devices, ...
* Research
  + Algorithms
  + NFC, QR Code, RFID
  + …

1. Program:

* Main functions
  + The application can allow the user to manage their borrowed books
  + Web Application for users
  + Mobile Application for users
  + Management Web Application for admins
  + ...

1. Other products:

* All of management functions of the system must be implemented to support the operating system in best

4. Other comment (propose all relative thing if have)

N/A

|  |  |
| --- | --- |
| **Supervisor (If have)**  *(Sign and full name)* | HCM city, date 14/12/2016  **On behalf of Registers**  *(Sign and full name)* |

# Table of Contents

Table of Contents 5

List of Tables 7

A. Introduction 8

1. Project Overview 8

2. Introduction 8

3. Current Situation 8

4. Problem Definition 8

5. Proposed Solution 9

6. Functional Requirements 10

7. Roles and Responsibilities 10

B. Software Project Management Plan 11

1. Problem Definition 11

1.1 Name of this Capstone Project 11

1.2 Problem Abstract 11

1.3 Project Overview 11

2. Project organization 14

2.1 Software Project Model 14

2.2 Roles and Responsibilities 14

2.3 Tools and Techniques 15

3. Project Management Plan 17

3.1 Product Backlog 17

3.2 Sprint Backlog 17

C. Software Requirement Specifications 17

1. User Requirement Specification 17

1.1 Admin Requirement 17

1.2 Manager Requirement 17

1.3 Staff Requirement 17

1.4 User Requirement 18

1.5 Scheduler Requirement 18

1.6 Unauthorized User 18

1.7 Authorized User 18

2. Software Requirement Specification 19

2.1 External Interface Requirement 19

2.2 System Overview Use Case 20

2.3 List of Use Cases 21

3. Software Requirement Specification 31

4. Conceptual Diagram 32

D. Software Design Description 33

1. System Architectural Design 33

1.1 Web Application Architecture Description 33

1.2 Web Service Application Architecture Description 34

1.3 Mobile Application Architecture Description 35

2. Component Diagram 36

3. Detailed Description 37

3.1 Class Diagram 37

3.2 Class Diagram Explanation 38

3.3 Interaction Diagram 38

4. Database Design 46

4.1 Entity Relationship Diagram (ERD) 46

4.2 Data Dictionary 46

5. Algorithms 47

5.1 Training Product Algorithms 47

5.2 Timsort Algorithms 49

5.3 Color Distance Algorithms 49

5.4 Detect Best Color Algorithms 49

5.5 Search with Picture Algorithms 50

5. Task sheet 53

6. Appendix 59

# List of Tables

[Table 1: Roles and Responsibilities 10](#_Toc479753853)

[Table 2: Roles and Responsibilities 15](#_Toc479753854)

[Table 3: Tools and Techniques 15](#_Toc479753855)

[Table 4: USE CASE - <User> Search 21](#_Toc479753856)

[Table 5: USE CASE - <User> Make Order 23](#_Toc479753857)

[Table 39: USE CASE - UC\_S01 - <Staff> Create Product 25](#_Toc479753858)

[Table 41: USE CASE - UC\_S03 - <Staff> Train Product 26](#_Toc479753859)

[Table 43: USE CASE - UC\_S05 - <Staff> Add Concepts 27](#_Toc479753860)

[Table 45: USE CASE - UC\_S07 - <Staff> Add Options 29](#_Toc479753861)

[Table 6: Conceptual Diagram Data Dictionary 31](#_Toc479753862)

[Table 7: Component Dictionary 35](#_Toc479753863)

[Table 8: Class Diagram Dictionary 37](#_Toc479753864)

**List of Figures**

[Figure 1: Scrum Process 14](#_Toc479753865)

[Figure 2: System Overview Use Case 19](#_Toc479753866)

[Figure 3: <User> Search 20](#_Toc479753867)

[Figure 5: <User> Make Order 22](#_Toc479753868)

[Figure 25: <Staff> Create Product 23](#_Toc479753869)

[Figure 27: <Staff> Train Product 25](#_Toc479753870)

[Figure 29: <Staff> Add Concepts 26](#_Toc479753871)

[Figure 31: <Staff> Add Options 28](#_Toc479753872)

[Figure 6: Conceptual Diagram 31](#_Toc479753873)

[Figure 7: Web Application Architecture 32](#_Toc479753874)

[Figure 8: Backend Application Architecture 33](#_Toc479753875)

[Figure 9: Mobile Application Architecture 34](#_Toc479753876)

[Figure 10: Component Diagram 35](#_Toc479753877)

[Figure 11: Class Diagram 36](#_Toc479753878)

[Figure 12: Activity Diagram - <Unauthorized User> Login 37](#_Toc479753879)

[Figure 13: Activity Diagram - <User> Search with Picture 39](#_Toc479753880)

[Figure 14: Activity Diagram - <User> Check out 40](#_Toc479753881)

[Figure 19: Sequence Diagram - <Staff> Get log 41](#_Toc479753882)

[Figure 15: Sequence Diagram - <Staff> Create Product 42](#_Toc479753883)

[Figure 16: Sequence Diagram - <Staff> Train Product 43](#_Toc479753884)

[Figure 17: Sequence Diagram - <Staff> Add Concepts 43](#_Toc479753885)

[Figure 18: Sequence Diagram - <Staff> Add Option 44](#_Toc479753886)

[Figure 20: Entity Relationship Diagram (ERD) 45](#_Toc479753887)

# Introduction

## Project Overview

* Project name: **Just Walk Out Library**
* Abbreviation: **JWL**
* Product Type: **Web Application & Mobile App**
* Start Date: **January 5th, 2017**
* End Date: **April 14­th, 2017**

## Introduction

For a long time, library has been a reliable place for people to study and borrow books. However, almost every library system nowadays still uses a traditional way for their borrowers to obtain books: a borrower has to carry all the books that he/she wants to the librarian, so that the librarian can record which books that the borrower will borrow; after that, the borrower signs in each form to commit that he/she is borrowing those books. Only after these steps, the borrower can take back his/her library card and bring the books home. Because one librarian can only manage one borrower at a time, these traditional steps can be very painful if the library is currently crowded with borrowers.

Therefore, we build a system to help libraries and their borrowers to solve those problems. Applying this system, they will not only save a huge amount of time in book-borrowing process, but they will also have a better way to manage books.

## Current Situation

In order to have a more thorough view on our current situation, we will look into the book-borrowing procedure in the libraries of Ho Chi Minh City University of Social Sciences and Humanities, and FPT University.

### Ho Chi Minh City University of Social Sciences and Humanities library

1. The borrower gives the library’s card/student’s card to the librarian.
2. The borrower receives the locker’s key to put the bag in.
3. The borrower walks into the room.
4. The borrower finds wanted books.
5. The borrower takes wanted books.
6. The borrower fills in the book-borrowing form.
7. The borrower brings the books and the form to the librarian’s desk.
8. The librarian checks the form and records the books.
9. The librarian gives the books to the borrower.
10. The borrower re-checks the books, takes back the bag, returns the locker’s key, and brings the books home.
11. The librarian arranges the book-borrowing forms in a specific order.

Reference: [link](http://hcmussh.edu.vn/Resources/Docs/SubDomain/lib/TV.QT10.01%20-%20Quy%20trinh%20phuc%20vu%20muon%20tai%20lieu%20ve%20nha.pdf)

### FPT University library

1. The borrower gives the student card to the librarian.
2. The borrower receives the locker key to put the bag in.
3. The borrower walks into the reading room.
4. The borrower finds and takes wanted books.
5. The borrower goes to the librarian to check out.
6. With each book, the student has to fill in a form with the book’s code, and his/her signature.
7. The librarian checks the forms and books.
8. The librarian scans each book’s barcode to record the student’s borrow list. The alarm is turned off for each scanned book.
9. The librarian returns the student’s card to the student.
10. The process finishes. The student can take the books home.

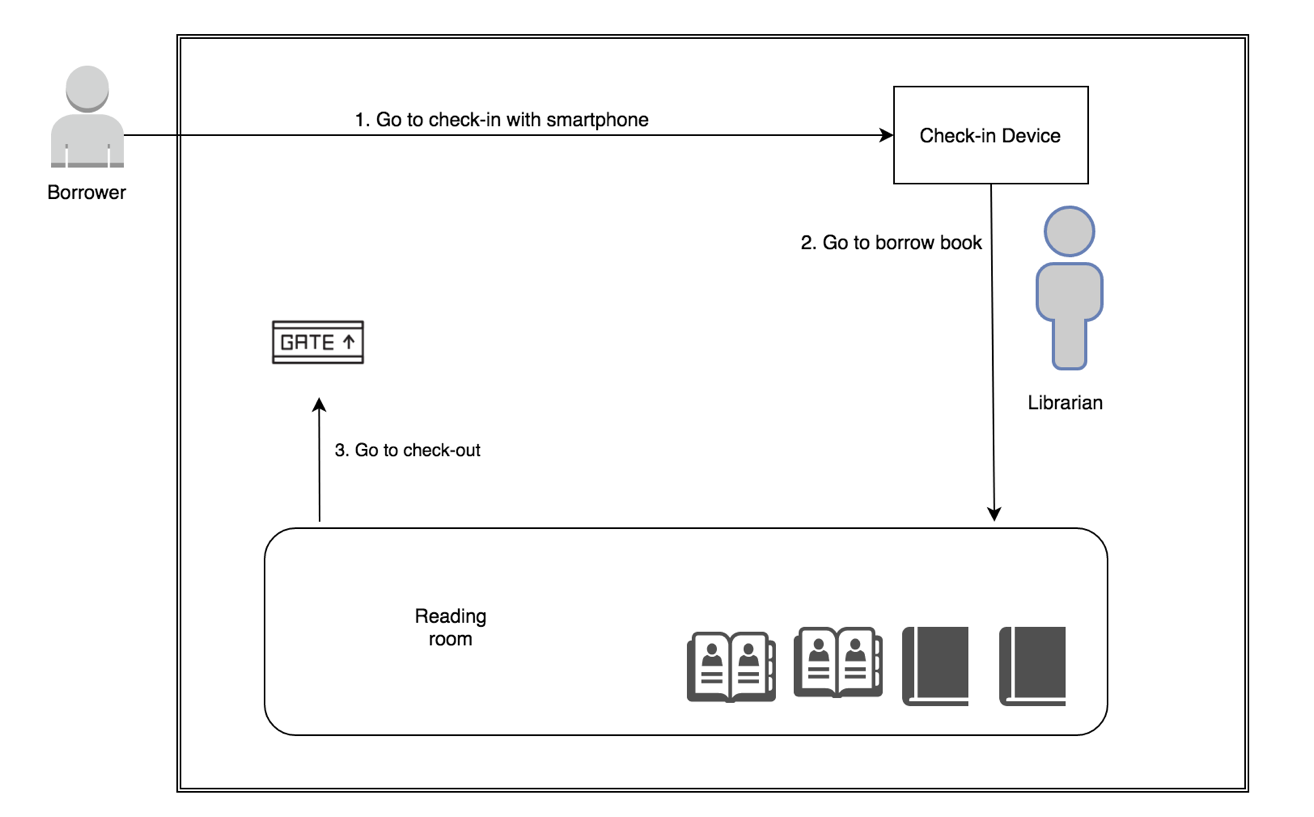
## Problem Definition

Because of current situation, we found that the traditional process has many advantages and disadvantages below:

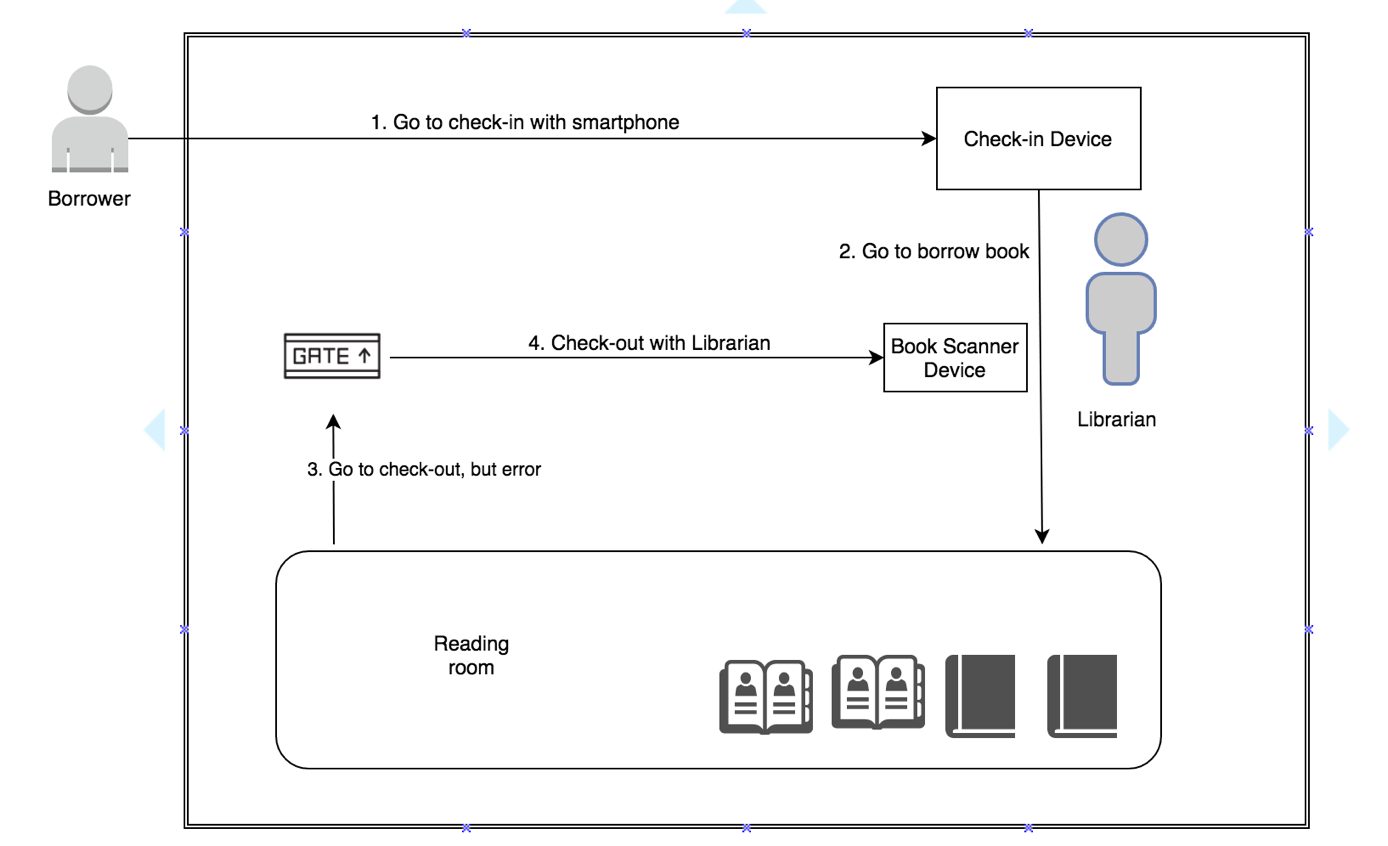
* Advantages:
* No technical skills needed.
* Disadvantages:
* Borrowing books need to go through complicated steps, thus during the rush hour, it might take too much time to wait for checking out for both borrowers and librarians.
* The librarian should work very hard when having many borrowers.
* Paper records are hard to manage and easy to be damaged.
* Librarians may forget to remind borrowers about their deadline.
* Borrowers have to memorize the deadline of their borrowing books.
* It is hard for borrowers to know when a wanted book is available.
* All activities that relate to borrowing and returning books are manual.

## Proposed Solution

Our proposed solution is to build a system named Just Walk out Library (JWL), which uses NFC, QR Code, RFID technologies to create a more easily and efficiently book - borrowing process: Borrowers only need to use smartphones to check in the library and borrow books, the system will automatically match and save the right books to the right borrower. Therefore, our solution helps to reduce librarian’s work in book – borrowing process, let them have more time for other tasks. Furthermore, librarians and borrowers can manage the borrowing books better.



*Figure 1: Borrower goes to borrow book*



*Figure 2: Borrower borrows book with Librarian*

* 1. Featured functions

Web application:

* Provides the ability to train the system with new products for Staff.
* An UI for Staff to manage all unknown products that have been searched by customers.
* An UI for Staff to manage all popular unknown products that have been searched by customers.

Mobile application:

* Provides customers with the ability to upload images to search.
* Customers can also add the measurements they want for auto size suggestion.
  1. Possible drawbacks
* Inability to yield results because of the poor image quality provided by customers.
* Inability to yield best results because the the product has not yet available in our system.
* Staffs have to spend significant time training the system due to a great number of products demanded by store managers.

## Functional Requirements

The functional requirements are listed as below:

* Customer component:
  + Upload an image: customers can search product by uploading images on their devices or take one.
  + Add products to wish list: customers can add products to their wish list so that they can get them later when the condition allowed.
  + Add products to cart: customers can add products they are about to buy to their cart.
* Admin component:
  + Admin will have the ability to manage stores and its managers, concepts and categories
* Manager component:
  + Manager can manage his/her store information and the store’s staffs
* Staff component:
  + Training: staffs will create new products and train the system to recognize them.
  + Receive notification: staffs will be notified with unknown products search result.
* System component:
  + Learn images: the system will learn to recognize each product using images provided by staffs.
  + Analyze image: the system will analyze images uploaded by customers.
  + Search image: the system will search the similar products based on the analysis results.
  + Notify: the system will write a log every time a product is searched without returning a result and notify staffs the popular unknown ones.

## Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| 1 | Kiều Trọng Khánh | Project Manager | Supervisor | khanhkt@fpt.edu.vn |
| 2 | Vũ Huy Quân | Developer | Leader | quanvhse61457@fpt.edu.vn |
| 3 | Nguyễn Minh Khôi | Developer | Member | khoinmse61471@fpt.edu.vn |
| 4 | Tạ Đức Lộc | Developer | Member | loctdse61622@fpt.edu.vn |
| *Table 1: Roles and Responsibilities* | | | | |

# B. Software Project Management Plan

## Problem Definition

### Name of this Capstone Project

* Official name: Shopping Clothes with Pictures
* Vietnamese name: Mua quần áo online thông qua hình ảnh
* Abbreviation: SCP

### Problem Abstract

* AI is a new technology in the software field and our team is not familiar with it so it might require time and effort for the whole team to be able to understand and use it.
* Moreover, when our team finally understands how it works, we need to figure out the correct steps to input data that helps create the best results.
* Our team also does not have UI, UX design background so our application and web design may not be easy to use or the main features may not attract enough attention from customers.

### Project Overview

#### Current diagnostic diseases

Below are the problems encountered in this project:

* **New with developing Clarifai:** Most our teams do not have experience with Clarifai. Therefore, our team need significant amount of time to research.
* **Lack of knowledge about data training flow:** Our project need team member to do the data training process for staffs. How these operations work affect the success of the project.
* **Lack of UI, UX knowledge:** Our team members all study Information System and no one have good understanding about UI and UX of a standard E-Commerce application. That is the reason why spending time to study popular websites UI, UX is quite important.
* **Need time to setup micro-service:** Our team members decided to separate the implementation of the applications (frontend, backend and data science) into different servers. Doing this bring flexibility, decoupling and performance to the whole system but need time and knowledge to do DevOps.

#### The Proposed System

* To understand and apply Clarifai, our team will divide to research and study about their API including:
* How to detect some common properties of a picture (color, material, type)
* How to train the system to understand new product given its pictures
* How to ask the system to predict the new picture
* The system will have three sub-systems:
* A web application for admins, staffs and managers
* A mobile application for customers to take pictures and do online shopping
* An API application to serve API for web, mobile and communicate with Clarifai

##### Web Application

* For admin:
* Manage stores, concepts, categories, managers
* Configure system
* For manager:
* Manage his/her store, staffs
* For staff:
* Manage products
* Provide training for new product
* For customer:
* Search products, make orders
* Manage profile

##### Mobile Application

The mobile application mostly for customers to:

* Search products
* Choose clothes sizes and make orders
* Manage profile

##### API Application

Main functions of backend site include

* Serve formatted data for both web and mobile
* Communicate with Clarifai
* Communicate with third party. For example: purchase services, storage services (S3 Amazon)

#### Boundaries of the System

The system should do:

* Allow customers to do shopping online
* Allow customers to search products by images
* Allow customers to input their clothing sizes
* Allow customers to input information about their body and suggest suitable size
* Suggest similar products to customers in case the system can not find the product
* Allow staffs to train product images
* Notify staffs when a product is not found many times

The system should not do:

Our system is not a store, therefore, we do not handle any store-related tasks

* + Do not manage inventory
  + Do not manage promotions
  + Do not manage human resources

#### Future plans

* System use machine learning to recommend products for customers
* Mobile and web application can support customers to build their own Virtue Cloth Model so that so customers can try clothes online before doing purchase

#### Development Environment

##### Hardware requirement

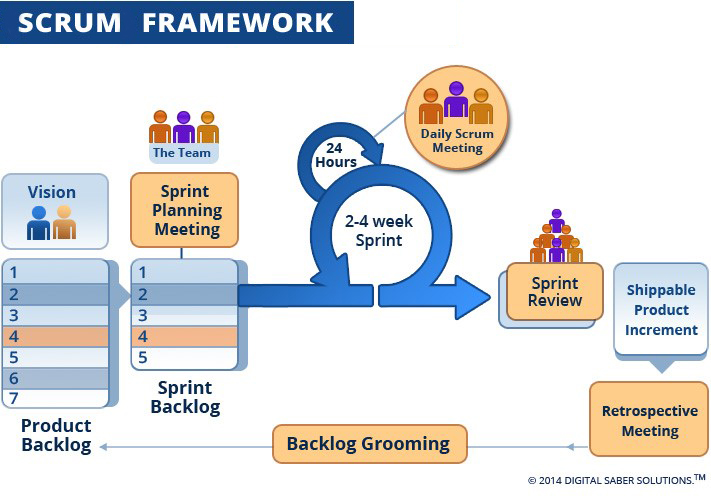
(*References to main document, Section B - 1.3.5)*

## Project organization

### Software Project Model

This project is developed using Scrum model – part of an agile framework for Software development project. Our team choose Scrum model because of the following reasons:

* Clarifai AI is a new concept to us, we need time to research and apply it gradually to our project.
* Prototypes are delivered frequently for evaluation, usually weekly, rather than months.
* Take fewer risks when there is a change in requirement.
* All members must work together in order to avoid misunderstanding or miscommunication.
* Able to study new skills or knowledge at the same time as developing.



*Figure 4: Scrum Process*

(*References*: <http://www.digitalsaber.com/process/agile-scrum-framework/>)

### Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full Name** | **Role in Group** | **Responsibilities** |
| 1 | Kiều Trọng Khánh | Product owner | * Specify scope and user requirement. * Supervise the development progress. * Provide professional techniques and business analysis support. |
| 2 | Vũ Huy Quân | Scrum master | * Managing process * Designing database * Clarifying requirements * Prepare documents * GUI design * Create test plan * Coding * Testing |
| 3 | Nguyễn Minh Khôi | Team member | * Designing database * Clarifying requirements * Prepare documents * GUI design * Create test plan * Coding   - Testing |
| 4 | Tạ Đức Lộc | Team member | * Designing database * Clarifying requirements * Prepare documents * GUI design * Create test plan * Coding * Testing |
| *Table 2: Roles and Responsibilities* | | | |

### Tools and Techniques

|  |  |  |
| --- | --- | --- |
|  | **Tools** | **Techniques** |
| Front-end | WebStorm | * HTML5 * CSS3 * ES6 * NodeJS * ReactJS |
| Back-end | RubyMine | * Rails * RESTFul * JSON * Ruby * PostgresSQL |
| Web server | Puma, nginx | N/A |
| Mobile Application | Android studio | * Java 8 * Android SDK |
| Database management system | PgAdmin | N/A |
| *Table 3: Tools and Techniques* | | |

## Project Management Plan

### Product Backlog

*(References to main document, Section B – 3.1)*

### Sprint Backlog

*(References to main document, Section B – 3.2)*

##### 

# Software Requirement Specifications

## User Requirement Specification

### Admin Requirement

Admin is the person who manages all the stores in the system. Admin can do the following functions:

* Manage stores:
* Add new store
* Edit store
* Get store
* Add concepts
* Manage category:
* Add category
* Edit category

### Manager Requirement

Manager is the person who manages staffs of the store which they are responsible for. Manager can do the following functions:

* Edit store
* Manage Staff:
* Add staff
* Edit staff
* Delete staff

### Staff Requirement

Staff is the person who responsible for training the system and managing products. Staff can do the following functions:

* Train Product
* Edit profile
* Manage Product:
* Edit Product
* Edit Product Image
* Create Product
* Manage Options:
* Add Options
* Edit Options
* Delete Options
* Manage Concepts for Images:
* Add concepts to Product
* Modify concepts of Product

### User Requirement

User can do the following functions:

* Get Product by Category
* Get Product Detail
* Get Order History
* Search

### Scheduler Requirement

* Notify Staff about Unknown Product Search

### Unauthorized User

* Sign Up
* Sign In

### Authorized User

* Make Order
* Cancel Order
* Check Out
* Sign Out
* Rate Product
* Manage Wish List:
* Add Wish List
* Remove Wish List

## Software Requirement Specification

### External Interface Requirement

#### User Interface

* The user interface uses Vietnamese as main language for users and English for Staff, Manager and Admin on both Mobile and Web application.
* The user interface displays best on 1024x768 and above screen size

#### Hardware Interface

* Android Smartphone: Nexus 4 or later.

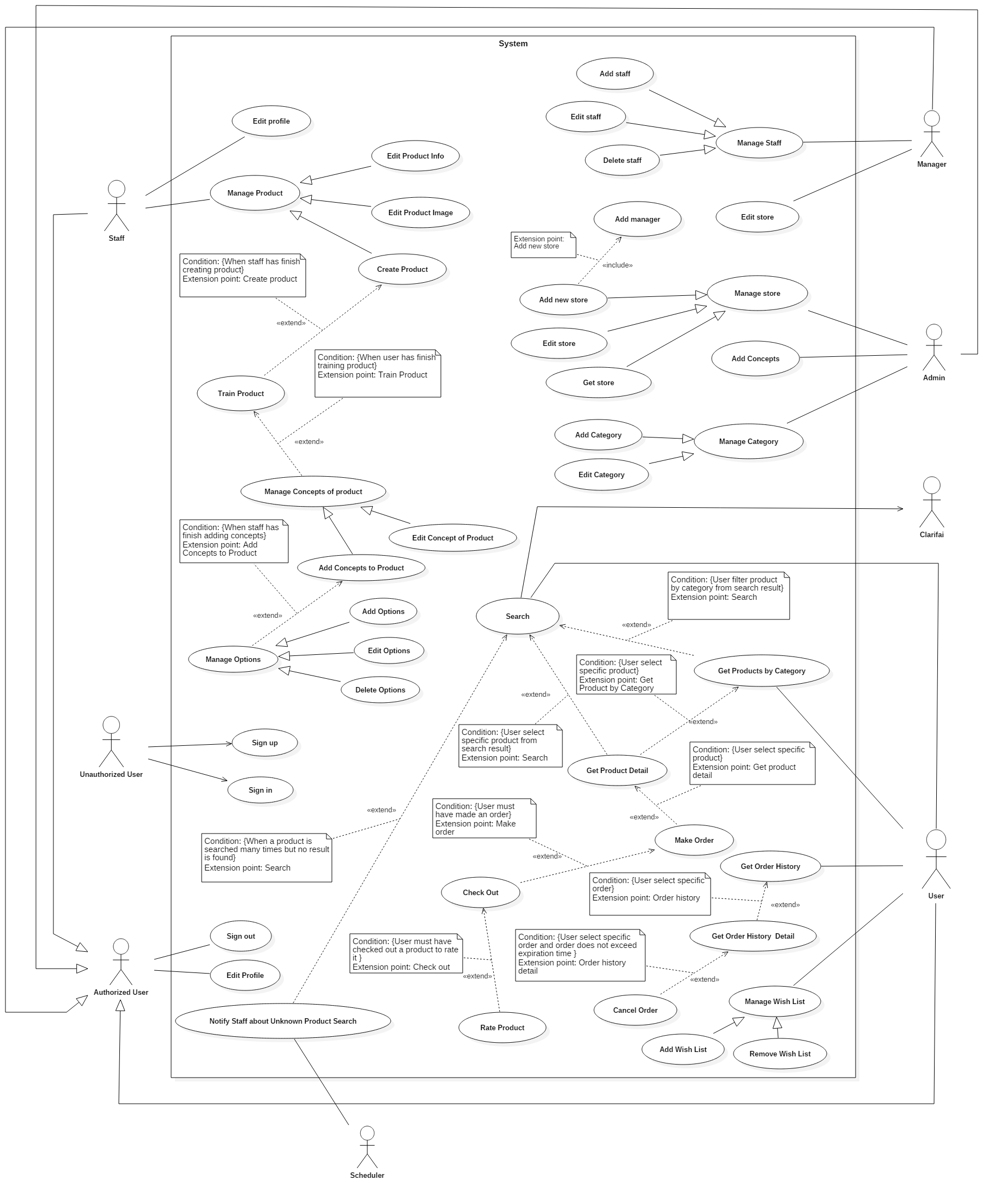
#### Software Interface

* Web application: work with Firefox (v30 or above), Chromes (v25 or above)
* Mobile Application: Android 4 (or later)

#### 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 service.
* Use HTTP protocol 1.1 for communication between the server and the Microsoft service.

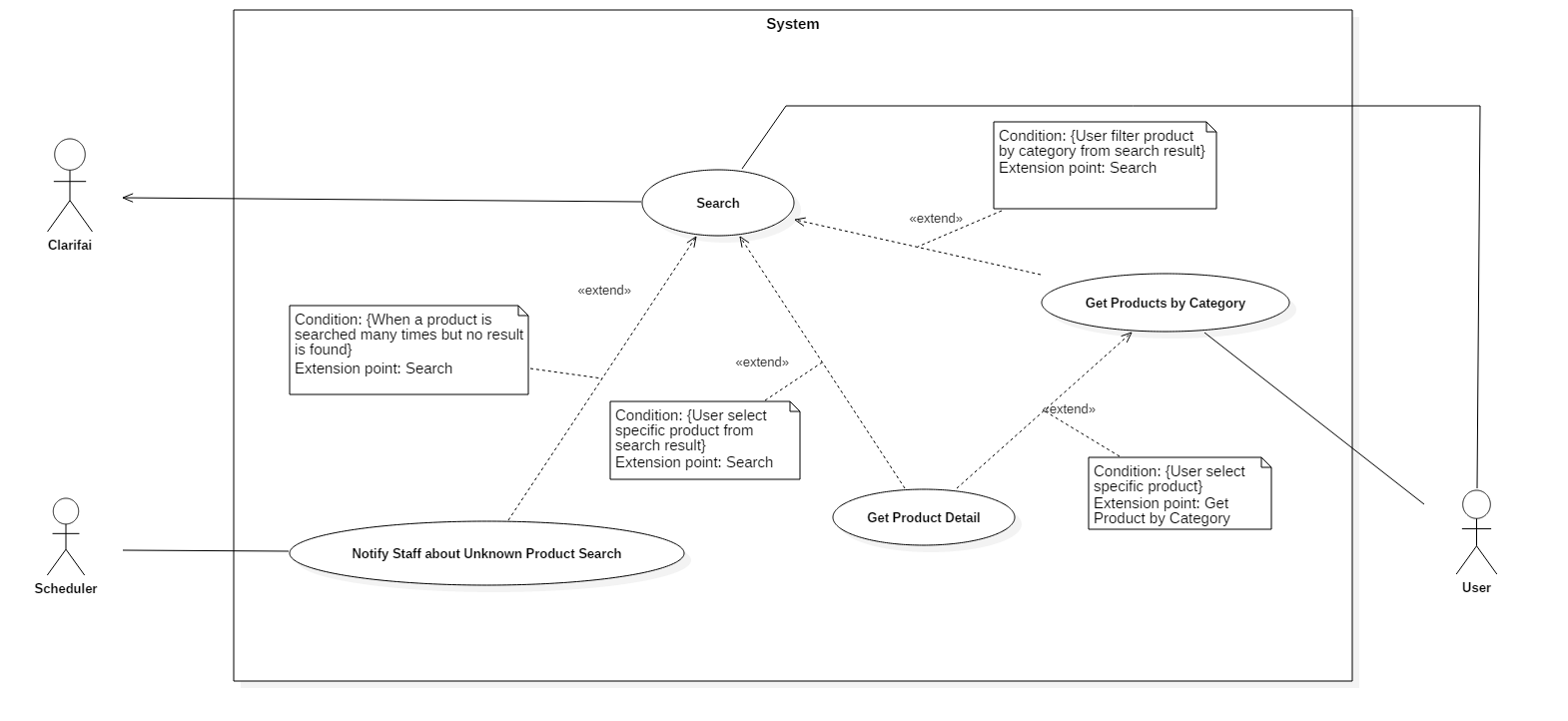
### System Overview Use Case



*Figure 5: System Overview Use Case*

### List of Use Cases

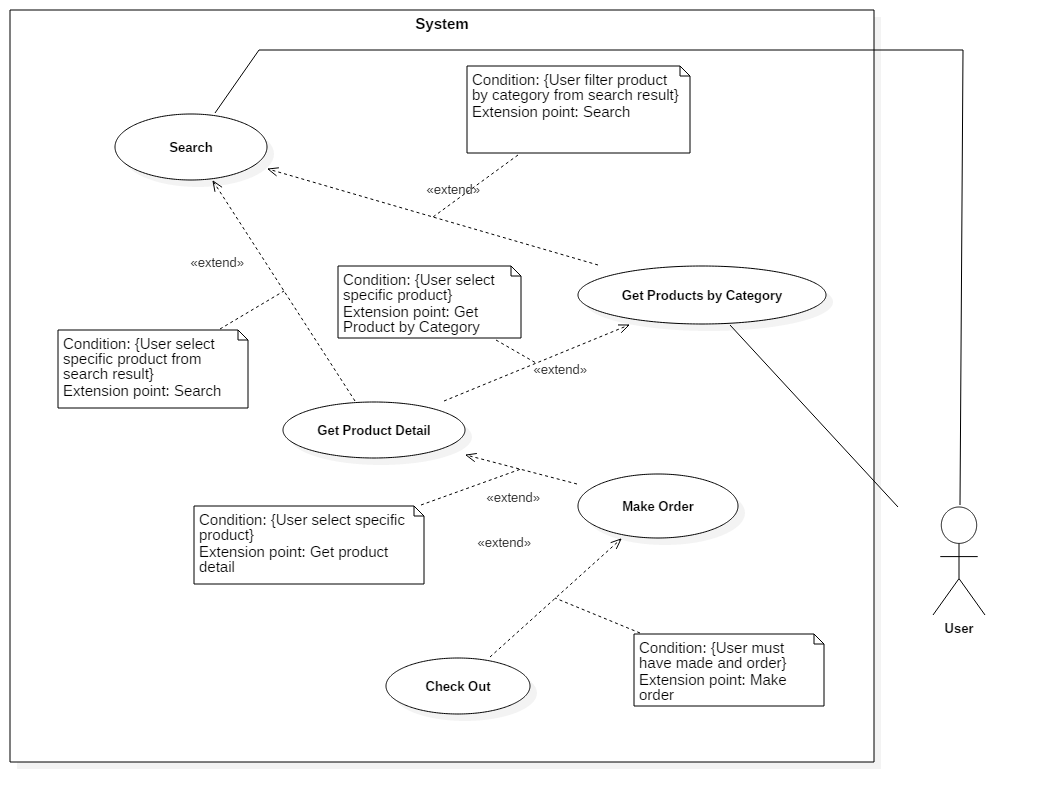
#### <User> Search



*Figure 6: <User> Search*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_U01** | | | |
| **Use Case No.** | UC\_U01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Search | | |
| **Author** | LocTD | | |
| **Date** | 13/01/2017 | **Priority** | Normal |
| **Actor:**   * User   **Summary:**   * This use case allows actor to search for products via keywords or pictures.   **Goal:**   * Display result for searched products.   **Triggers:**   * Actor sends command to search.   **Preconditions:**   * Actor must login as User role.   **Post Conditions:**   * **Success:** Show results of possible products that match user input. * **Fail:** System will display error message below.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor input keywords to search textbox. |  | | 2 | Actor sends command to search.  [Alternative 1, 2] | System process keywords and return search results. |   **Alternative Scenario:**  *[Alternative 1]*   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User choose to search with pictures | System open Camera. | | 2 | User capture new picture or select from devices. |  | | 3 | User sends command to upload. | System upload picture to server.  [Exception 1, 2] | | 4 |  | System requires user to select Category. | | 5 | User select category. | System calls web service. | | 6 |  | System returns possible search result in a grid view, 2 columns, paging 20.  [Exception 3] |   *[Alternative 2]*   |  |  |  | | --- | --- | --- | | Step | Cause | System Response | | 1 | User receives no result. | System writes log and notify staff. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Invalid picture format. | System shows message: “Uploaded picture must be in jpg/jpeg/png format and must have at least 600x600px and less than 160Kb.” | | 2 | Unable to upload pictures. | System shows message: “System timeout, please try again later.” | | 3 | No search results. | System shows message: “System timeout, please try again later”. |   **Relationships:**   * Extend from Notify staff about Unknown product search: System will notify staff when a product is searched but no result is found. * Extend from Get product detail: When user selects a specific product from search results. * Extend from Get Product Category: When user filter product by category from search results.   **Business Rules:**   * Log record structure in database:  |  |  |  |  |  | | --- | --- | --- | --- | --- | | No. | Product | Message | Created at | User Id | |  |  |  |  |  | | | | |
| *Table 4: USE CASE - <User> Search* | | | |

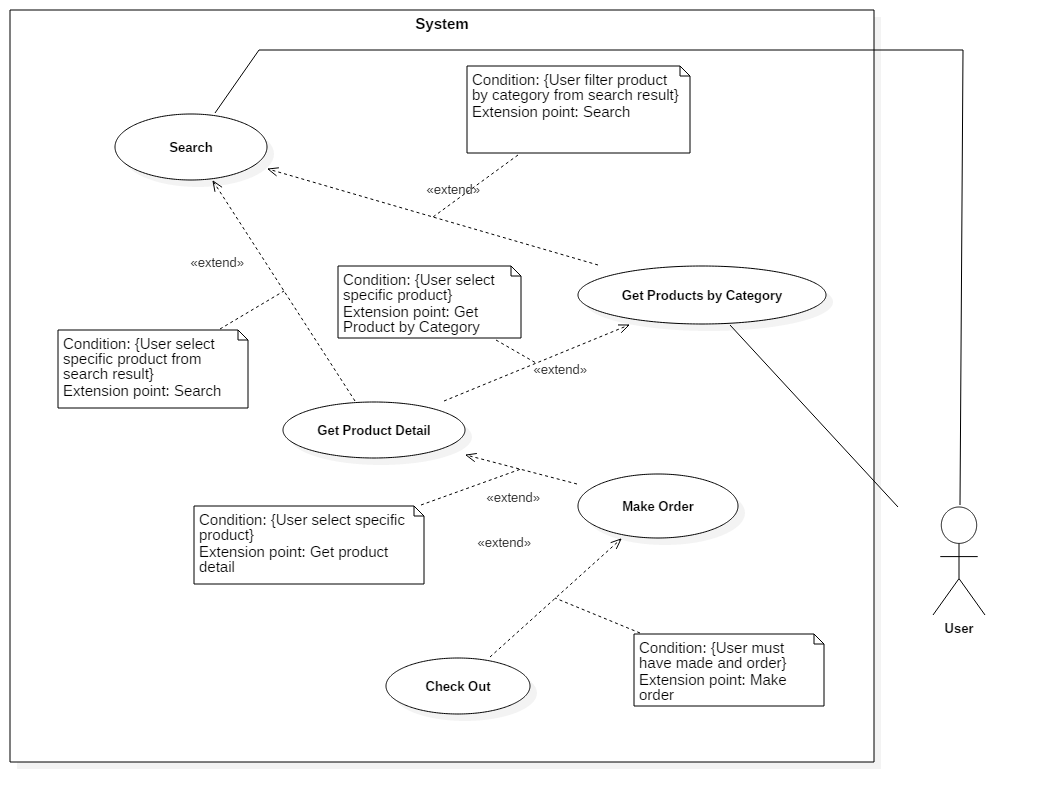
#### <User> Make order



*Figure 7: <User> Make Order*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_U04** | | | |
| **Use Case No.** | UC\_U04 | **Use Case Version** | 2.0 |
| **Use Case Name** | Make Order | | |
| **Author** | LocTD | | |
| **Date** | 13/01/2017 | **Priority** | Normal |
| **Actor:**   * User   **Summary:**   * This use case allows actor to make an order for a specific product.   **Goal:**   * Add selected products to cart.   **Triggers:**   * Actor sends command to make order.   **Preconditions:**   * Actor must login as User role. * User must select a specific product.   **Post Conditions:**   * **Success:** Add selected product to cart. * **Fail:** N/A.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor goes to product detail view. | System display detailed product information. | | 2 | Actor selects “Thêm vào giỏ” button | System require actor to select product option:   * **Color** * **Size** * **Quantity** | | 3 | Actor selects prefered product options |  | | 4 | Actor sends command to add to cart. | System add selected item to cart. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:**   * Extend to Get Product Detail: When user selects a specific product. * Extend from Check Out: When user has already made an order.   **Business Rules:**   * System will create a cart for actor in database. * If the selected product is already existed in the cart, increase the quantity by the newly added number. * If the selected product is already existed in the cart with different options, system will create a new instance for the selected product with corresponding options. | | | |
| *Table 24: USE CASE - UC\_U04 - <User> Make Order* | | | |

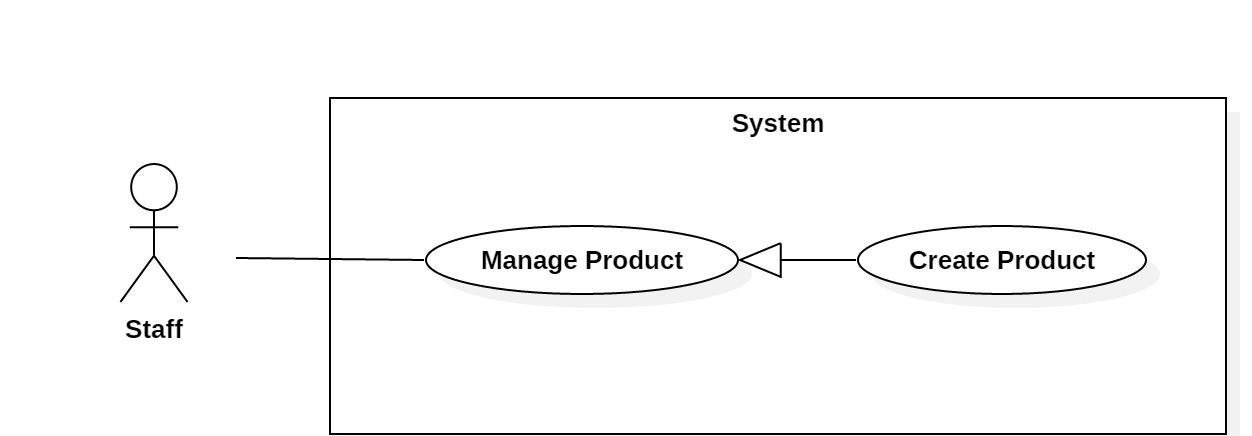
#### <User> Check out



*Figure 8: <User> Check Out*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_U05** | | | |
| **Use Case No.** | UC\_U05 | **Use Case Version** | 2.0 |
| **Use Case Name** | Check Out | | |
| **Author** | LocTD | | |
| **Date** | 13/01/2017 | **Priority** | Normal |
| **Actor:**   * User   **Summary:**   * This use case allows actor to check out orders using their prefered payment.   **Goal:**   * To finish buying products.   **Triggers:**   * Actor sends command to check out order.   **Preconditions:**   * Actor must login as User role. * Actor must have added items to cart.   **Post Conditions:**   * **Success:** Notify user by email. * **Fail:** System will display error message below.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor goes to check out view. | System display cart with added items. | | 2 | Actor sends command to check out. | System request user to choose payment method. | | 3 | Actor selects payment method. | System request user confirmation. | | 3 | Actors select “Finish”.  [Alternative 1]  [Exception 1] | System will notify actor via email and redirect to actor’s home page. |   **Alternative Scenario:**  *[Alternative 1]*   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor selects “Cancel”. | System returns to Check Out page. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 |  | System shows message: “System is busy” when the internet is lost. |   **Relationships:**   * Extend to Make Order: Actor must have made an order. * Extend from Rate Product: Actor must have checked out a product to rate it.   **Business Rules:**   * System will take actor’s address, phone, email to create order. * Each product in cart will have 1 order detail. * On checkout, order status will be marked as “Created”. * All products will be removed from cart once finished checking out. | | | |
| *Table 25: USE CASE - UC\_U05 - <User> Check Out* | | | |

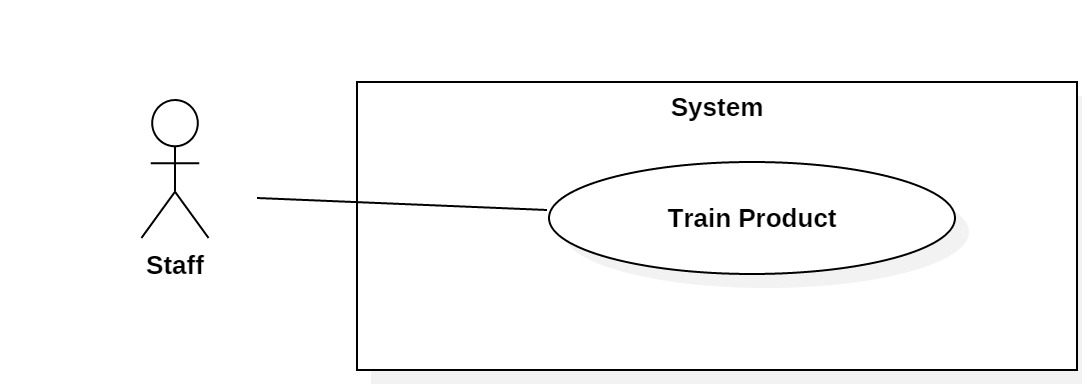
#### <Staff> Create Product



*Figure 9: <Staff> Create Product*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_S01** | | | |
| **Use Case No.** | UC\_S01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Product | | |
| **Author** | KhoiNM | | |
| **Date** | 18/01/2016 | **Priority** | High |
| **Actor:**   * Staff   **Summary:**   * This use case helps Actor create a new product in the system when a new physical product is available.   **Goal:**   * A new product is added to the system.   **Triggers:**   * Actor sends create product command.   **Preconditions:**   * Actor must login as a Staff role.   **Post Conditions:**   * **Success:** New product is added to the system. * **Fail:** Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor goes to create product view. | System requires information from actor: - **Product image**: hidden free text input, required, length 8 - 40 - **Product name**: free text input, required, length 8 - 40  - **Price**: free text input, required, float  - **Description**: free text input, required, length 8 - 500  - **Category**: select one of the options in list, required | | 2 | Actor inputs required information |  | | 3 | Actor sends create product command | System validates inputted information  [Exception 1, 2, 3, 4, 5] | | 4 |  | System creates a new product and redirects to train product view |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Cause | System Response | | 1 | Invalid image | System shows error message: “Image cannot be blank and must be smaller than 772 x 960 ”. | | 2 | Invalid product name format | System shows error message: “Product name cannot be blank and must be 8-40 characters long”. | | 3 | Invalid price format | System shows error message: “Price cannot be blank or contain alphabetical characters and must be 1-9 digits, ”. | | 4 | Invalid description format | System shows error message: “Description name cannot be blank and must be 8-500 characters long”. | | 5 | Category is not selected | System shows error message: “Category must be selected”. |   **Relationships:** N/A  **Business Rules:**   * Categories are loaded from system. * Main image will be compressed before uploaded into S3. * S3 will return the image urls then the system will store these urls into database * Main image will be use to represent the product when displaying in UI * System will change status of product to Info | | | |
| *Table 39: USE CASE - UC\_S01 - <Staff> Create Product* | | | |

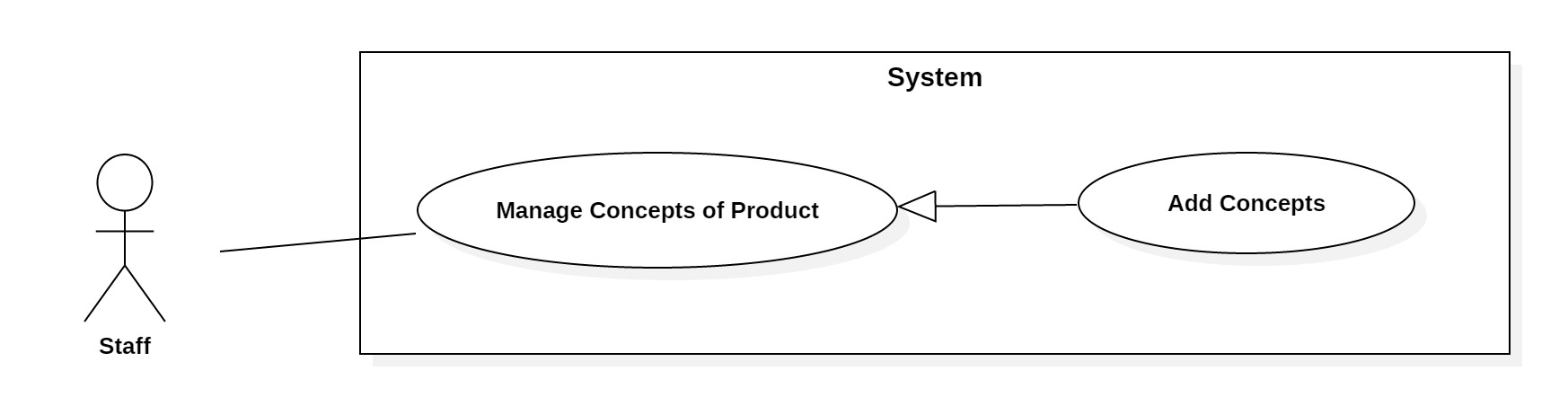
#### <Staff> Train Product



*Figure 10: <Staff> Train Product*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_S01** | | | |
| **Use Case No.** | UC\_S03 | **Use Case Version** | 2.0 |
| **Use Case Name** | Train Product | | |
| **Author** | KhoiNM | | |
| **Date** | 18/01/2016 | **Priority** | High |
| **Actor:**   * Staff   **Summary:**   * This use case helps Staff upload similar images of the product into the system.   **Goal:**   * The systems can recognize the product or similar products after being trained.   **Triggers:**   * Staff sends train command.   **Preconditions:**   * Users must be logged in with Staff role. * Product must be created successfully.   **Post Conditions:**   * **Success:** Product is trained successfully. * **Fail:** Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Staff goes to train product view | System show training view and requires:  - **Images**: hidden free text input, required, length 8 - 40 | | 2 | Staff uploads images | System validates inputted information  [Exception 1] | | 3 |  | Systems saved the images |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Cause | System Response | | 1 | Invalid image | System shows error message: “Image cannot be blank and must be smaller than 600 x 960 ”. |   **Relationships:** Extends to Create Product  **Business Rules:**   * Images will be compressed before being uploaded into S3. * S3 will return the image urls then the system will store these urls into database * System will change status of product to Image | | | |
| *Table 41: USE CASE - UC\_S03 - <Staff> Train Product* | | | |

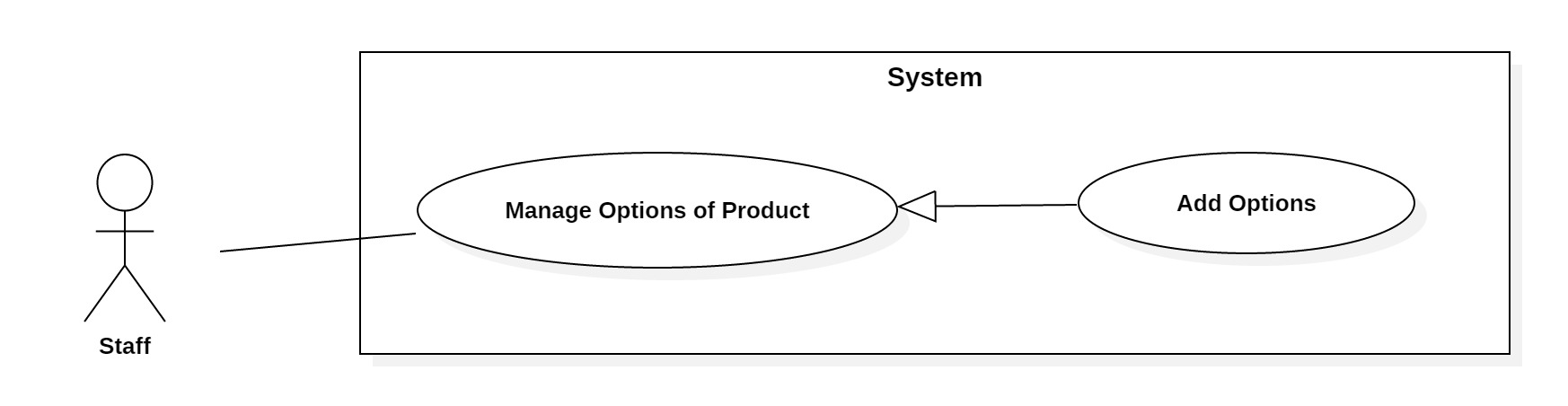
#### <Staff> Add Concepts



*Figure 11: <Staff> Add Concepts*

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_S05** | | | |
| **Use Case No.** | UC\_S05 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Concepts | | |
| **Author** | KhoiNM | | |
| **Date** | 18/01/2016 | **Priority** | High |
| **Actor:**   * Staff   **Summary:**   * This use case helps actor add concepts that describe correctly the product to improve the system’s recognition accuracy.   **Goal:**   * Concepts are added to a product.   **Triggers:**   * Actor sends add concepts command.   **Preconditions:**   * Users must be logged in with Staff role * Train product step must be completed.   **Post Conditions:**   * **Success:** Concepts are added successfully * **Fail:** System will display error message below   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor goes to add concepts view | System requires information from actor:  - **Concepts**: free text input, required, length: 1 - 40 | | 2 | Actor inputs the information |  | | 3 | Actor sends add concepts command | System validates inputted information  [Exception 1] | | 4 |  | System adds inputted concepts to product |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Cause | System Response | | 1 | Invalid concepts | System shows error message: “Concepts cannot be blank or duplicated”. |   **Relationships:** Extends to Train Product  **Business Rules:**   * Concepts for the product will be stored into database * System will change status of product to Concept | | | |
| *Table 43: USE CASE - UC\_S05 - <Staff> Add Concepts* | | | |

#### <Staff> Add Options



*Figure 12: <Staff> Add Options*

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case No.** | UC\_S07 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Options | | |
| **Author** | KhoiNM | | |
| **Date** | 18/01/2016 | **Priority** | High |
| **Actor:**   * Staff   **Summary:**   * This use case helps actor add options to a product (Option includes color, size and quantity).   **Goal:**   * Colors, sizes and quantity of the product are available to choose in customer’s UI.   **Triggers:**   * Actor sends add options command.   **Preconditions:**   * Users must be logged in with Staff role. * Product must be created.   **Post Conditions:**   * **Success:** New options are added successfully * **Fail:** Show error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor goes to add options view. | System requires information from actor:  - **Size**: select one of the options in list, required.  - **Color**: select one of the options in list, required.  - **Quantity**: free text input, Integer, required. | | 2 | Actor inputs information. |  | | 3 | Actor sends add option command | System validates inputted information  [Exception 1] | | 4 |  | System saves options to product. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Cause | System Response | | 1 | Size is not selected | System shows error message: “Required”. | | 2 | Color is not selected | System shows error message: “Required”. | | 3 | Quantity price format | System shows error message: “Quantity cannot be blank or contain alphabetical characters and must be 1-9 digits, ”. |   **Relationships:** Extends to Add Concepts  **Business Rules:**   * A product has many options * System will check for duplicated options before adding * System will change status of product to Option | | | |
| *Table 45: USE CASE - UC\_S07 - <Staff> Add Options* | | | |

## Software Requirement Specification

* 1. **Usability**

- Admin and Staff interface are written in English including all dialogs and messages.

- User interface is written in Vietnamese including all dialogs and messages.

- The system needs no more than 2 days of training for staffs.

* 1. **Reliability**

- The system can detect pictures with highly accurate result (more than 80%).

- Each product is trained with more than hundreds of pictures to ensure accuracy.

- The scheduler will notify staff in case there was an unknown product search for several times. User will be notified whenever the product is available.

* 1. **Availability**

- Server has back up method to protect and restore necessary data as soon as possible whenever there is a problem.

* 1. **Security**

- Each role of user has a specific permission to interact with the system.

- System always checks for authorization and authentication before doing anything.

- Only Admin can grant permission to other roles.

* 1. **Maintainability**

- The system is divided into separated modules.

* 1. **Portability**

- User can use the mobile application on devices running Android 4 or later.

- Web application can be run on Chrome browser version 42 or later.

* 1. **Performance**

- System detects and returns results in 4 seconds or less under 4Mbps bandwidth.

- Pictures are uploaded to S3 storage in less than 1 second under 4Mbps bandwidth.

## Conceptual Diagram

Conceptual(1)

*Figure 13: Conceptual Diagram*

**Data Dictionary**

|  |  |
| --- | --- |
| **Entity Name** | **Description** |
| Admin | Person who manages the system. |
| Staff | Person who trains the system. |
| User | Contains information of all users in the system. |
| Wishlist | Contains a list of products created by users. |
| Order | Contains a list of orders made by users. |
| Payment | Contains all payment methods. |
| Product | Contains information of all products in the system. |
| Concept | Contains a list of concepts created by staff. |
| Image | Contains all images of products. |
| Category | Contains all categories of products. |
| Attr Product | Contains all properties of products. |
| Size | Contains all sizes of products. |
| Color | Contains all colors of products. |
| *Table 6: Conceptual Diagram Data Dictionary* | |

# Software Design Description

## System Architectural Design

### Web Application Architecture Description

Web Architecture (1)

*Figure 14: Web Application Architecture*

The web application uses Flux (<https://facebook.github.io/flux/>) as an architecture. We choose this architecture for a couple of reasons:

* To maximize the render performance of web client. Our application may need most of the time to do training
* Easy to develop, this architecture separates frontend out of backend. It helps developers can work independently with each other
* The architecture divides the application into components. This increases reusability and testability
* Reduce side effects in application

### Web Service Application Architecture Description

Backend Architecture

*Figure 15: Backend Application Architecture*

The Web Service Application Architecture will base on MVC architecture. We choose this architect because of following advantages:

* The application will be divided into three clear part
* Testing each component in the architecture become easier
* The existing of Model or Business Layer leverage testability and decoupling to the View
* Active Record is a pattern to abstract the data layer and manipulate the database; it also acts as an ORM. It combines both data and domain logic into object so that user can easily know how to update the data

### Mobile Application Architecture Description



*Figure 16: Mobile Application Architecture*

Mobile application architecture will use MVVM. There is couple of reasons choosing this architect:

* It minimizes the number of dependency, helps increase loose coupling between application
* The View is passive and will be notified to update data
* The View Model will contain complex view logic; it also has no information about View. This help increase reusability of View Model
* The Model is where business logic sit in; it also performs API requests to backend application
* Develop Mobile Application using MVVM have increase productivity and reduce side effects

## Component Diagram



*Figure 17: Component Diagram*

|  |  |
| --- | --- |
| **Name** | **Description** |
| SCP System | Shopping Clothes with Picture system |
| Web Service | Component is used to provide API for mobile application to interact with the system. |
| Business Logic | Common services are used to handle system’s business operations |
| Controller | Component to handle HTTP request |
| Management | Component to manage system data |
| Order | Component to handle making order and interact with payment system |
| Payment System | External component for handle purchasing |
| Image Recognizer | External component for doing image recognizer |
| Training | Component to train new product to be recognized by system |
| Search | Component to handle search products |
| Active Record | Abstract data layer to interact with database system |
| Database System | Where application data is manipulated and stored |
| Scheduler | Run background job and send message to other system |
| Mail System | External Component to send email |
| Cloud Message | External Component to send notification to user |
| *Table 7: Component Dictionary* | |

## Detailed Description

### Class Diagram



*Figure 18: Class Diagram*

|  |  |  |
| --- | --- | --- |
| **Class name** | **Mapping column with conceptual diagram** | **Description** |
| User Entity | User | Contains the information of users in the system. |
| Order Entity | Order | Contains the detailed information of order. |
| Payment Entity | Payment | Contains the information of payment method. |
| Wishlist Entity | Wishlist | Contains the information of user’s wishlist. |
| Product Entity | Product | Contains the informtaion of all products in the systems. |
| Attr Product Entity | Attr Product | Contains the detailed information of all products in the systems. |
| Category Entity | Category | Contains the information of all categories in the system. |
| Image Entity | Image | Contains all images of products. |
| Concept Entity | Concept | Contains all concepts of products. |
| Size Entity | Size | Contains all sizes of products. |
| Color Entity | Color | Contains all colors of products. |
| *Table 8: Class Diagram Dictionary* | | |

### Class Diagram Explanation

*(References to main document, Section D – 4.1)*

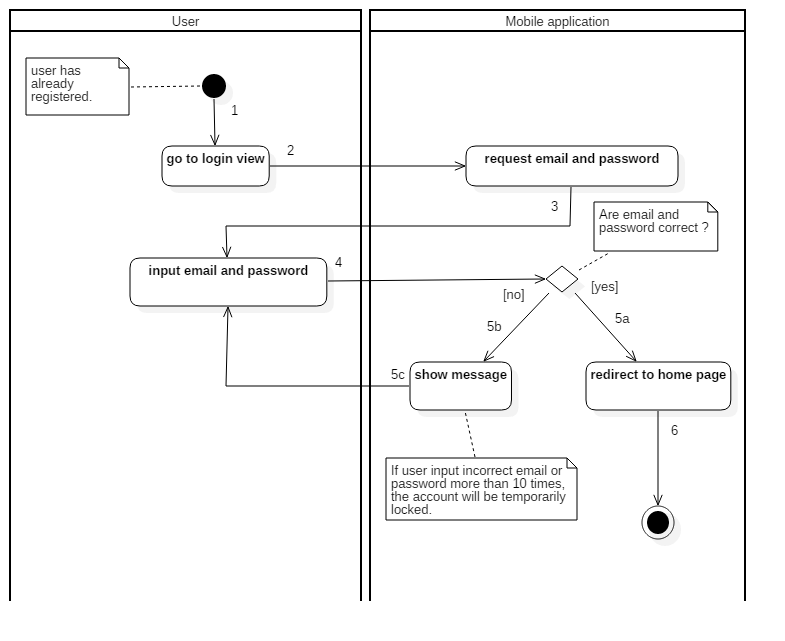
### Interaction Diagram

#### Mobile Application

##### Unauthorized User

###### **<Unauthorized User> Login**

Summary: these diagrams show the process of unauthorized user login into the system using mobile application



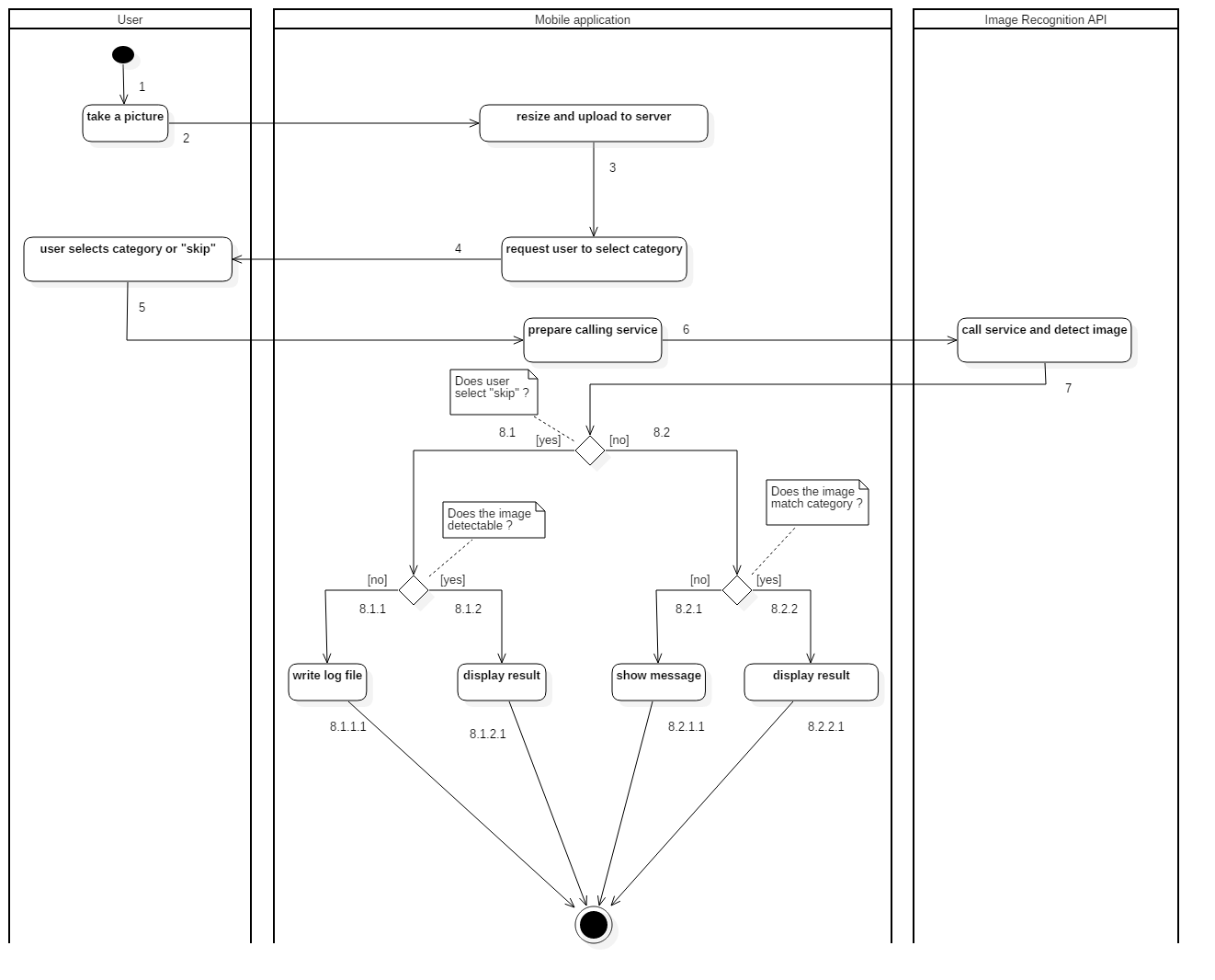
*Figure 19: Activity Diagram - <Unauthorized User> Login*

|  |  |  |
| --- | --- | --- |
| 1. Start | 2. Go to Login view | 3. Request email and password |
| 4. Input email and password | 5a. Correct email and password | 5b. Incorrect email and password |
| 5c. Input email and password again | 6. Redirect to home page and finish |  |

##### User

###### **<User> Search with Picture**

Summary: This diagram shows the process of user searching for products with picture using mobile application

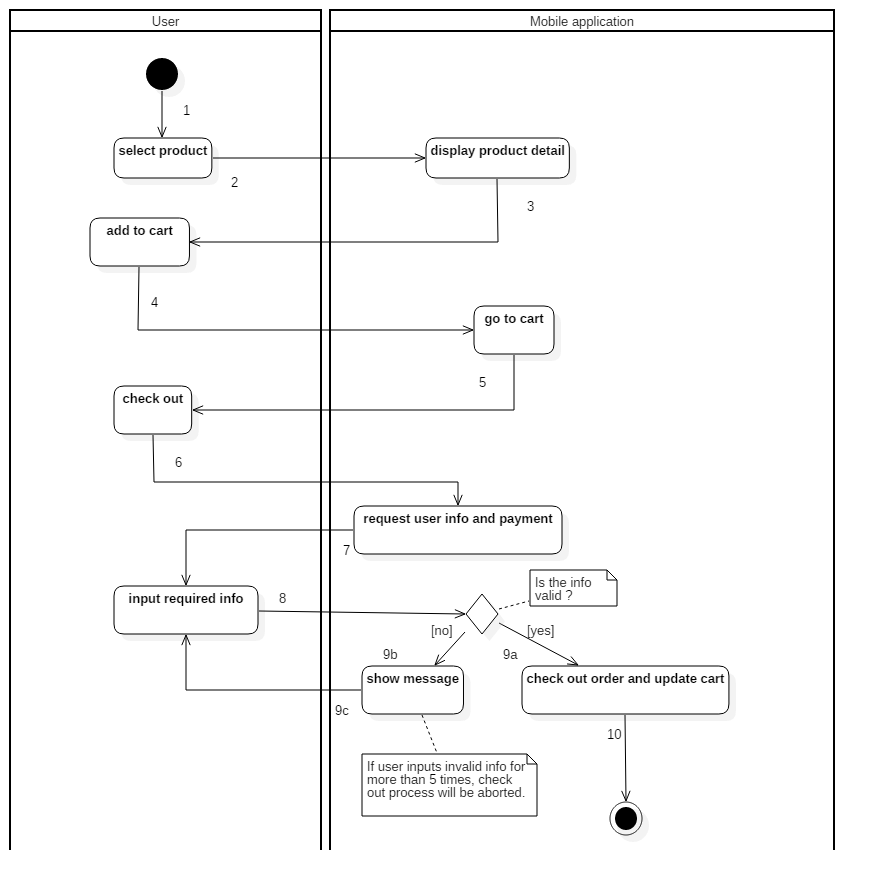


*Figure 20: Activity Diagram - <User> Search with Picture*

|  |  |  |
| --- | --- | --- |
| 1. Start | 2. Take a picture | 3. Resize and upload to server |
| 4. Request user to select category | 5. User selects category or “skip” | 6. Prepare calling service |
| 7. Call service and detect image | 8.1. User selects “skip” | 8.1.1. Unable to detect image |
| 8.1.1.1. Write log file and finish | 8.1.2. Able to detect image | 8.1.2.1. Display results and finish |
| 8.2. User selects a category | 8.2.1. Image does not match the category | 8.2.1.1. Show message and finish |
| 8.2.2. Image matches the category | 8.2.2.1. Display result and finish. |  |

###### **<User> Check out**

Summary: This diagram shows the process of user checking out order using mobile application.



*Figure 21: Activity Diagram - <User> Check out*

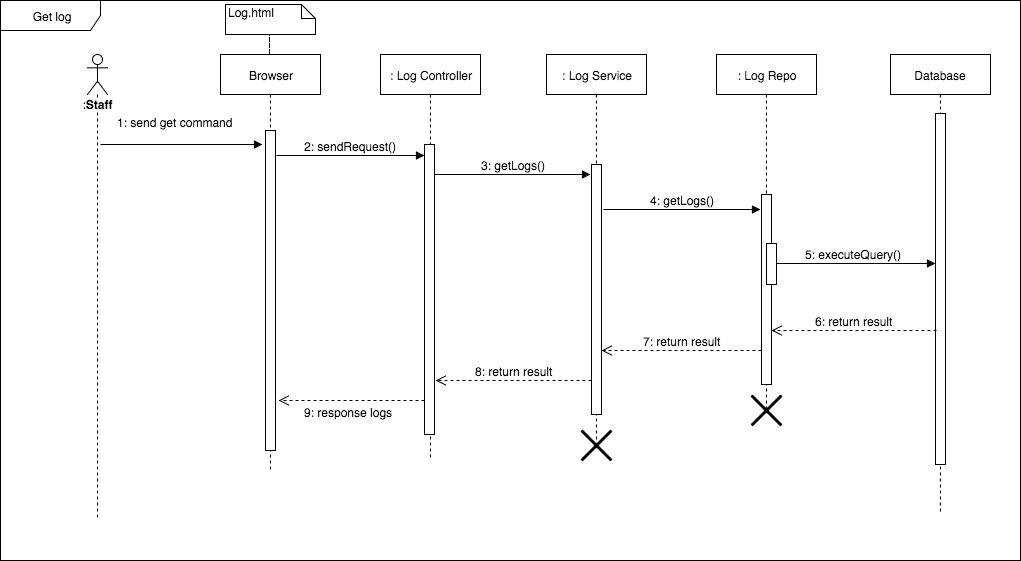
|  |  |  |
| --- | --- | --- |
| 1. Start | 2. Select product | 3. Display product detail |
| 4. Add to cart | 5. Go to cart | 6. Check out |
| 7. Request user info and payment | 8. Input required info | 9a. Valid info |
| 9b. Invalid info | 9c. Input required info again | 10. Checkout order and update cart then finish. |

#### Web Application

##### Staff

###### ***<Staff> Get log***

Summary: This diagram show the process of staff getting logs using the web application



*Figure 22: Sequence Diagram - <Staff> Get log*

###### **<Staff> Create product**

Summary: This diagram show the process of staff creating a product using the web application

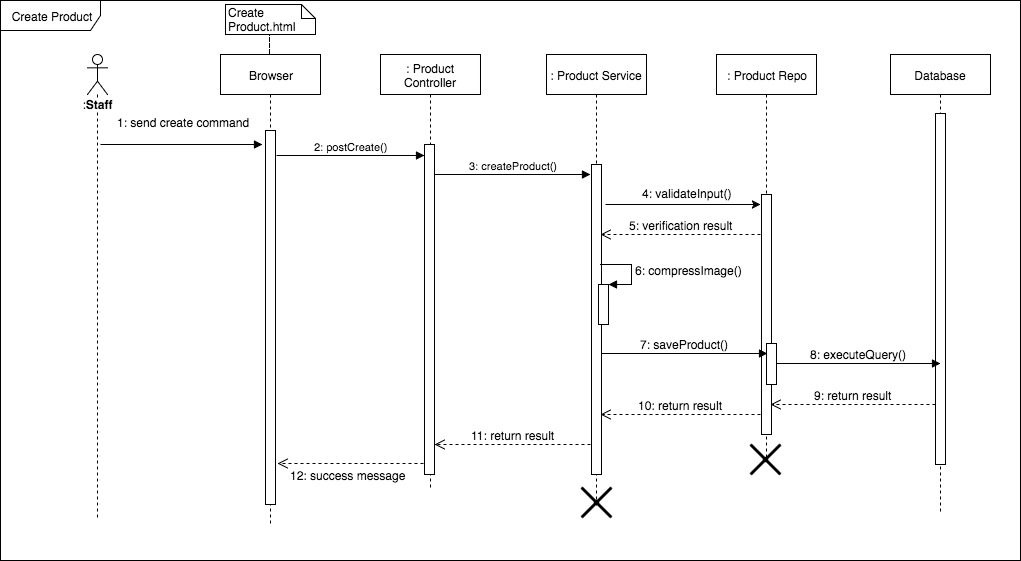
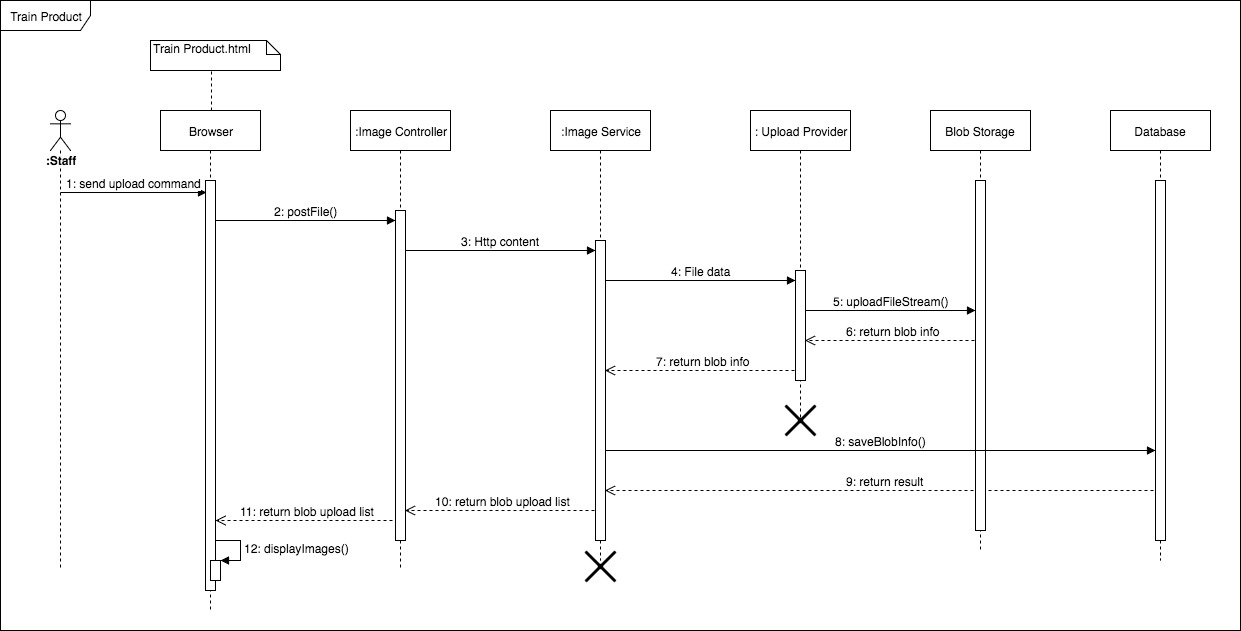


Figure *23: Sequence Diagram - <Staff> Create Product*

###### **<Staff> Train product**

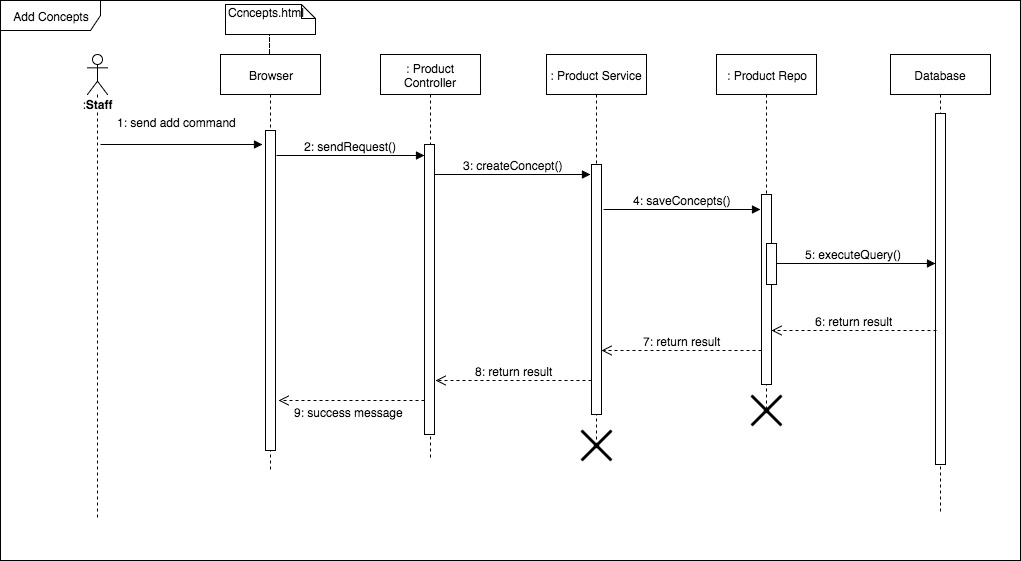
Summary: This diagram show the process of staff uploading images to train a product using the web application



*Figure 24: Sequence Diagram - <Staff> Train Product*

###### **<Staff> Add concepts**

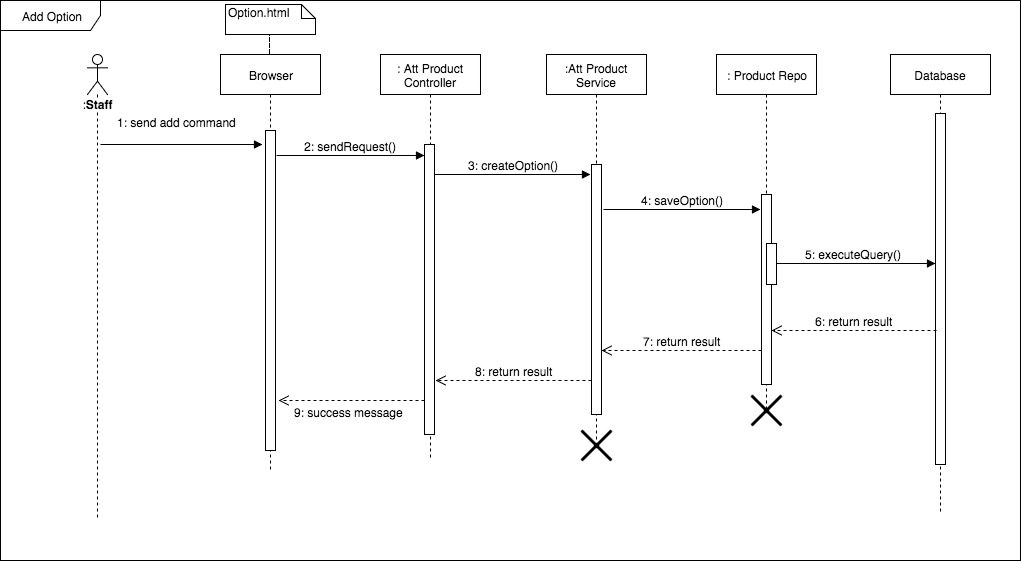
Summary: This diagram show the process of staff uploading images to train a product using the web application



*Figure 25: Sequence Diagram - <Staff> Add Concepts*

###### ***<Staff> Add Option***

Summary: This diagram show the process of staff adding available options to a product using the web application



*Figure 26: Sequence Diagram - <Staff> Add Option*

## Database Design

### Entity Relationship Diagram (ERD)



*Figure 27: Entity Relationship Diagram (ERD)*

### Data Dictionary

|  |  |
| --- | --- |
| **Entity Data Dictionary: describe contents of all entities** | |
| **Entity Name** | **Description** |
| Admin | Contains information of all admins in the system. |
| Staff | Contains information of all staff in the system. |
| User | Contains information of all users in the system. |
| Wishlist | Contains a list of products created by users. |
| Order | Contains a list of orders made by users. |
| Payment | Contains all payment methods. |
| Product | Contains information of all products in the system. |
| Concept | Contains a list of concepts created by staff. |
| Image | Contains all images of products. |
| Category | Contains all categories of products. |
| AttrProduct | Contains all properties of products. |
| Size | Contains all sizes of products. |
| Color | Contains all colors of products. |

## Algorithms

### Training Product Algorithms

#### Definition

* The algorithms propose method to collect new product images and tags to make the system aware of the product and increase accuracy of image recognition.
* Train has two meanings:

1. Train a completely new product.

2. Train an existing product to improve system’s accuracy.

#### Define the problem

* Users are able to search products by image so we need algorithms to train new products to help the system be able to recognize them. Algorithms can analyze the image and improve the predictive models.

#### Image Recognizer Concepts

* **Concept:** is a definition that be attached to an image. It could be understood as tags of an image. A concept belongs to a model.
* **Non-Concept:** is a definition that be attached to an image. It tells that the image does not have this tag. This is useful to increase accuracy of Image Recognizer.
* **Model:** is a set of concepts (or tags), it could be customized to predict a given image. An output of a model is percent of each concept in that model. For example: a food model contains: pizza, noodle and hamburger can have an output like: 20% pizza, 10% noodle and 1% of hamburger.
* **Input:** is an image used to feed a model. It could be an url or a raw file

#### Solution

To make Image Recognizer to be useful we decided to create two custom models.

* **Category:** used to detect a category of an image. For example: detect if a cloth is t-shirt, skirt or dress
* **Concept:** used to detect the definition of design of a cloth. For example: detect if a cloth is stripe, plain, modern, classic or having superman

When creating a new product, staff will add images, concepts and non-concepts to make the system aware of the product. The training process includes:

* Step 1: Staff creates a new product with basic information includes: an image, name, price, description and category
* Step 2: Staff inputs multiple images that best describes the new product
* Step 3: Staff then inputs concepts that best describes the product. He or she also input non-concepts to tell that those concepts do not belong to this product
* Step 4: Staff then adds options to the product. For example, a cloth can have many color and variety of sizes
* Step 5: Staff publishes the product. The system then wraps all necessary data (images, concepts and non-concepts) and submits them to Image Recognizer System in a separate background thread. The system will start training process
* Step 6: After successful training the system then notify users about new product

#### Complexity

In total, the complexity of this algorithm based on the image recognizer. They do not publish summary algorithms or any descriptions. As the result, we spend time testing the algorithm complexity.

|  |  |
| --- | --- |
| **Network** | **Time (s)** |
| Wi-Fi (Download: 35.91M, Upload: 22.08M, Ping: 2ms) | 17.066 |
| 3G (Download: 24.44M, Upload: 2.55M, Ping: 20ms) | 18.882 |

#### Accuracy

* From testing result image recognition rate is more than 80%. The testing process:
* Step 1: train a product with multiple images
* Step 2: send those images one by one to the predict API
* Step 3: collect data and analyze
* From the result to optimize the training we process we should input at least 5 images. From the document adding more images we increase the accuracy of training process. We recommend input 10 images for each product

### Timsort Algorithms

#### Definition

- The algorithms used to sort a given array

#### Define the problem

- The system can sort the search result base on color difference. As the result, we need an algorithm to solve it

#### References

- References to algorithm complexity and document:

<https://en.wikipedia.org/wiki/Timsort>

### Color Distance Algorithms

#### Definition

- The algorithms used to determine the similarity of two colors

#### Define the problem

- The system can find the best color match the image. It can also sort the clothes based on color. This lead to the need of calculate a score between a detected color and a color of the clothes

#### Solution

- We will use CIEDE2000 algorithm for calculating the distance between two given colors

#### References

- References to algorithm complexity and document:

<https://en.wikipedia.org/wiki/Color_difference>

### Detect Best Color Algorithms

#### Definition

- The algorithms used to find the best color available in the system the match the color results from Image Recognizer

#### Define the problem

- Users would like to have search results sorted by color. As the result, we need an algorithm to find the best color in the system that match the input image

#### Solution

- Step 1: User input an image

- Step 2: System send image to Image Recognizer to detect colors

- Step 3: Sort the result from Image Recognizer and take only the first half of them. The formula is:

(n is the number of colors return by Image Recognizer, a is hex color of a color return from Image Recognizer, b is the percent of a from the input image, D is the max color difference, and c is a color in the system

- Step 4: System iterates through all colors available and calculates the score between them and finds the best result

#### Complexity

- Given N is number of colors return from Image Recognizer

- The sort algorithm takes 

- Calculate the score takes 

- Given M is number of colors in the system (M is less then 20)

🡪 The complexity of the algorithm is 

#### Comparing to other methods

<https://drive.google.com/open?id=0B67tFOgiGX6YUzNBaE01dXBsU2s>

### Search with Picture Algorithms

#### Definition

- The algorithms used to get products from the system that match the picture taken by users

#### Define the problem

- Users can search for products via pictures, so we need an algorithm that can make our system able to detect the pictures and return the best possible result. It can also detect pictures based on users’ choice of categories.

#### Solution

When searching for products, users can either select pictures from device or take a new one. The process includes:

* Step 1: Users input picture
* Step 2: System will scale and save picture
* Step 3: Users continue to select category from the list provided by the system (ex: Shirts, T-Shirt, Jeans or Shorts) or chooses to skip
* Step 4: System send image url to Image Recognizer
* Step 5: If Image Recognizer unable to detect the image (result concept and category must be >= 50%), system will write a log file to record it
* Step 6: Otherwise return data for user

#### Complexity

The complexity of this algorithm is depended on the image recognition service for which they do not provide any summary or description; therefore, we spend time testing the algorithm complexity.

|  |  |
| --- | --- |
| **Network** | **Time (s)** |
| Wi-Fi (Download: 35.91M, Upload: 22.08M, Ping: 2ms) | 4.039 |
| 3G (Download: 24.44M, Upload: 2.55M, Ping: 20ms) | 5.633 |

#### Accuracy

* From testing, result rate returned from Image Recognition service is more than 80%. (Result from 524 out of 561 cases for Category and 454 out of 561 cases for Concept)
* Test result:
* Category: <https://drive.google.com/open?id=0B0EgBSrhB9kfUFA3RVVoSnoyRlU>
* Concept:

<https://drive.google.com/open?id=0B0EgBSrhB9kfSzNqMzBNMldILXM>

* The testing process:
* Step 1: Search for new products that do not already exist in the system
* Step 2: Start training for the new products
* Step 3: Search for those products again and collect data

# Task sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Product Deliverables** | **Task** | **QuanVH** | **KhoiNM** | **LocTD** | **Size** |
| 1 | Report1 - Introduction | Project Overview |  | **O** |  |  |
| Introduction |  | **O** |  |  |
| Current Situation |  | **O** |  |  |
| Problem Definition |  |  | **O** |  |
| Proposed Solution |  |  | **O** |  |
| Functional Requirements | **O** |  |  |  |
| Roles and Responsibilities | **O** |  |  |  |
| 2 | Report2- Software Project Management Plan | Problem Definition |  |  |  |  |
| Name of this Capstone Project |  | **O** |  |  |
| Problem Abstract |  | **O** |  |  |
| Project Overview |  | **O** |  |  |
| Project organization |  |  |  |  |
| Software Project Model |  |  | **O** |  |
| Roles and Responsibilities |  |  | **O** |  |
| Tools and Techniques |  |  | **O** |  |
| Project Management Plan |  |  |  |  |
| Product Backlog | **O** |  |  |  |
| Sprint Backlog | **O** |  |  |  |
| Deliverables | **O** |  |  |  |
| All Meeting Minutes |  | **O** |  |  |
| **Coding Convention** | **O** |  |  |  |
| 3 | Report 3- Software Requirement Specification | [User Requirement Specification](#_Toc479512100) |  |  |  |  |
| [Admin Requirement](#_Toc479512101) | **O** |  |  |  |
| [Manager Requirement](#_Toc479512102) | **O** |  |  |  |
| [Staff Requirement](#_Toc479512103) | **O** |  |  |  |
| [User Requirement](#_Toc479512104) | **O** |  |  |  |
| [Scheduler Requirement](#_Toc479512105) | **O** |  |  |  |
| [Unauthorized User](#_Toc479512106) | **O** |  |  |  |
| [Authorized User](#_Toc479512107) | **O** |  |  |  |
| [Software Requirement Specification](#_Toc479512108) |  |  |  |  |
| [External Interface Requirement](#_Toc479512109) | **O** |  |  |  |
| [System Overview Use Case](#_Toc479512110) | **O** |  |  |  |
| **List of usecase** |  |  |  |  |
| [<Admin> Add new store](#_Toc479512179) |  | **O** |  |  |
| [<Admin> Edit store](#_Toc479512180) |  | **O** |  |  |
| [<Admin> Get store](#_Toc479512181) |  | **O** |  |  |
| [<Admin> Add Category](#_Toc479512182) |  | **O** |  |  |
| [<Admin> Edit Category](#_Toc479512183) |  | **O** |  |  |
| [<Admin> Add Concepts](#_Toc479512184) |  | **O** |  |  |
| [<User> Search](#_Toc479512185) |  |  | **O** |  |
| [<User> Get Products by Category](#_Toc479512186) |  |  | **O** |  |
| [<User> Get Product Detail](#_Toc479512187) |  |  | **O** |  |
| [<User> Make Order](#_Toc479512188) |  |  | **O** |  |
| [<User> Check Out](#_Toc479512189) |  |  | **O** |  |
| [<User> Rate Product](#_Toc479512190) |  |  | **O** |  |
| [<User> Get Order History](#_Toc479512191) |  |  | **O** |  |
| [<User> Get Order History Detail](#_Toc479512192) |  |  | **O** |  |
| [<User> Cancel Order](#_Toc479512193) |  |  | **O** |  |
| [<User> Add Wish List](#_Toc479512194) |  |  | **O** |  |
| [<Manager> Add staff](#_Toc479512196) |  | **O** |  |  |
| [<Manager> Edit staff](#_Toc479512198) |  | **O** |  |  |
| [<Manager> Delete staff](#_Toc479512199) |  | **O** |  |  |
| [<Manager> Update store](#_Toc479512201) |  | **O** |  |  |
| [<Staff> Create Product](#_Toc479512203) |  | **O** |  |  |
| [<Staff> Edit Product Info](#_Toc479512204) |  | **O** |  |  |
| [<Staff> Train Product](#_Toc479512205) |  | **O** |  |  |
| [<Staff> Edit Trained Product](#_Toc479512206) |  | **O** |  |  |
| [<Staff> Add Concepts](#_Toc479512207) |  | **O** |  |  |
| [<Staff> Edit Concepts](#_Toc479512208) |  | **O** |  |  |
| [<Staff> Add Options](#_Toc479512209) |  | **O** |  |  |
| [<Staff> Add Options](#_Toc479512210) |  | **O** |  |  |
| [<Staff> Edit Options](#_Toc479512211) |  | **O** |  |  |
| [<Authorized User> Edit Profile](#_Toc479512212) |  | **O** |  |  |
| [<Authorized User> Sign out](#_Toc479512213) |  | **O** |  |  |
| [<Unauthorized User> Sign up](#_Toc479512214) |  | **O** |  |  |
| [<Unauthorized User> Sign In](#_Toc479512215) |  | **O** |  |  |
| [Software Requirement Specification](#_Toc479512112) | **O** |  |  |  |
| [Conceptual Diagram](#_Toc479512113) | **O** |  |  |  |
| 4 | Report 4- Software Design Description | [Design Overview](#_Toc479512115) |  | **O** |  |  |
| [System Architectural Design](#_Toc479512116) |  |  |  |  |
| [Web Application Architecture Description](#_Toc479512117) | **O** |  |  |  |
| [Web Service Application Architecture Description](#_Toc479512118) | **O** |  |  |  |
| [Mobile Application Architecture Description](#_Toc479512119) | **O** |  |  |  |
| [Component Diagram](#_Toc479512120) |  |  | **O** |  |
| [Detailed Description](#_Toc479512121) |  |  |  |  |
| [Class Diagram](#_Toc479512122) |  |  | **O** |  |
| [Class Diagram Explanation](#_Toc479512123) |  |  | **O** |  |
| [Interaction Diagram](#_Toc479512124) |  |  | **O** |  |
| [Interface](#_Toc479512125) |  |  |  |  |
| [Component interface](#_Toc479512126) |  |  | **O** |  |
| [User Interface Design](#_Toc479512127) |  |  | **O** |  |
| [Database Design](#_Toc479512128) |  |  |  |  |
| [Entity Relationship Diagram (ERD)](#_Toc479512129) | **O** |  |  |  |
| [Data Dictionary](#_Toc479512130) | **O** |  |  |  |
| [Algorithms](#_Toc479512131) |  |  |  |  |
| [Training Product Algorithms](#_Toc479512132) | **O** |  |  |  |
| [Timsort Algorithms](#_Toc479512133) | **O** |  |  |  |
| [Color Distance Algorithms](#_Toc479512134) | **O** |  |  |  |
| [Detect Best Color Algorithms](#_Toc479512135) |  |  | **O** |  |
| [Search with Picture Algorithms](#_Toc479512136) |  |  | **O** |  |
| [Suggest T-Shirt Size Algorithms](#_Toc479512137) |  | **O** |  |  |
| [Suggest Shirt Size Algorithms](#_Toc479512138) |  | **O** |  |  |
| 5 | Report 5 - Software Implementation and Test Document | [Introduction](#_Toc479512140) |  |  |  |  |
| [Overview](#_Toc479512141) |  | **O** |  |  |
| [Test Approach](#_Toc479512142) |  | **O** |  |  |
| [Database Relationship Diagram](#_Toc479512143) |  |  |  |  |
| [Physical Diagram](#_Toc479512144) | **O** |  |  |  |
| [Data Dictionary](#_Toc479512145) | **O** |  |  |  |
| [Implementation](#_Toc479512146) |  |  |  |  |
| [Backend Architecture Implementation](#_Toc479512147) |  |  | **O** |  |
| [Mobile Application Architecture Implementation](#_Toc479512148) | **O** |  |  |  |
| [Web Architecture Implementation](#_Toc479512149) |  | **O** |  |  |
| [Test Driven Development](#_Toc479512150) | **O** |  |  |  |
| [Test Plan](#_Toc479512151) |  |  |  |  |
| [Communication Diagram](#_Toc479512152) |  |  | **O** |  |
| [System Testing Test Case](#_Toc479512153) |  |  |  |  |
| [Web Application Test Cases](#_Toc479512154) |  | **O** |  |  |
| [Mobile Application Test Cases](#_Toc479512155) |  | **O** |  |  |
|  | Database | Create database schema |  |  | **O** | 3 |
| Insert default data schema |  |  | **O** | 1 |
|  | Framework | Design server code structure |  |  | **O** | 5 |
| Design mobile code structure | **O** |  |  | 5 |
| Design web code structure |  | **O** |  | 5 |
|  | **Algorithm** | **Implement search algorithm** |  |  | **O** | **5** |
| **Implement train algorithm** |  |  | **O** | **5** |
| **Implement detect best color algorithm** | **O** |  |  | **5** |
| **Implement sort color algorithm** | **O** |  |  | **5** |
| **Implement suggest T-shirt size algorithm** |  | **O** |  | **5** |
| **Implement suggest Shirt size algorithm** |  | **O** |  | **5** |
|  | Web | Implement manage accounts |  | **O** |  | 1 |
| Implement manage categories |  | **O** |  | 1 |
| Implement manage concepts |  | **O** |  | 1 |
| Implement train product |  | **O** |  | 3 |
| Implement manage stores |  | **O** |  | 1 |
| Implement manage logs |  | **O** |  | 2 |
| Implement manage staffs |  | **O** |  | 1 |
|  | Server | Implement manage component |  |  | **O** | 1 |
| Implement search component | **O** |  |  | 3 |
| Implement train component |  |  | **O** | 3 |
| Implement order component |  |  | **O** | 2 |
| Implement scheduler component | **O** |  |  | 3 |
|  | Mobile | Implement taking picture | **O** |  |  | 1 |
| Implement resize picture | **O** |  |  | 2 |
| Implement search | **O** |  |  | 2 |
| Implement add to cart | **O** |  |  | 1 |
| Implement choose option | **O** |  |  | 1 |
| Implement checkout | **O** |  |  | 1 |
| Implement add to wishlist | **O** |  |  | 1 |
| Implement notification | **O** |  |  | 2 |
| 6 | Report 6 - Software User's Manual | [Installation Guide](#_Toc479512157) |  |  |  |  |
| [Setting up environment at server side](#_Toc479512158) |  |  | **O** |  |
| [Web Services Application Deployment Process](#_Toc479512159) |  | **O** |  |  |
| [Mobile Application Deployment Process](#_Toc479512160) | **O** |  |  |  |
| [User Guide](#_Toc479512161) |  |  |  |  |
| [Mobile application](#_Toc479512162) |  |  | **O** |  |
| [Web application](#_Toc479512163) |  | **O** |  |  |

# Appendix

1. UML standard diagrams:

<https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm>

1. Component diagram notation:

<https://www.ibm.com/developerworks/rational/library/dec04/bell/>

1. Timsort algorithm:

<https://en.wikipedia.org/wiki/Timsort>

1. UML Diagrams:

<http://www.uml-diagrams.org/>

1. Clarifai documents:

<https://developer.clarifai.com/docs/>

1. Color difference:

<https://en.wikipedia.org/wiki/Color_difference>

1. Amazon documents:

<https://aws.amazon.com/documentation/s3/>