|  |  |
| --- | --- |
|  | **FPT ACADEMY INTERNATIONAL**  **FPT – APTECH COMPUTER EDUCATION** |

**Centre Name: ACE-HCMC-2-FPT.**

**Address: 590 Cach Mang Thang 8, District 3, Ho Chi Minh City, Viet Nam.**

**MYSTIC JEWEL**

|  |  |  |
| --- | --- | --- |
| **Supervisor:** | Mr. Ngo Phuoc Nguyen | |
| **Semester:** | IV | |
| **Batch No:** | F2-13-1607-T1 | |
| **Group No:** | 2 | |
| **Order:** | **Full name** | **Roll No.** |
| 1. | Du Hue Phuong | Student948490 |
| 2. | Nguyen Khai Nguyen | Student935423 |
| 3. | Huynh Tan Phat | Student958915 |

April 2018

|  |  |
| --- | --- |
|  | **FPT ACADEMY INTERNATIONAL**  **FPT – APTECH COMPUTER EDUCATION** |

Table of Contents

[REVIEW 1 2](#_Toc512519615)

[1. INTRODUCTION 2](#_Toc512519616)

[2. EXISTING SCENARIO 2](#_Toc512519617)

[3. CUSTOMER REQUIREMENT SPECIFICATION 2](#_Toc512519618)

[4. FUNCTIONAL REQUIREMENT SPECIFICATION 2](#_Toc512519619)

[5. SYSTEM REQUIREMENTS 2](#_Toc512519639)

[6. DEVELOPMENT SOFTWARE 2](#_Toc512519642)

[7. TECHNOLOGY 2](#_Toc512519643)

[REVIEW 1 – TASK SHEET 2](#_Toc512519644)

[REVIEW 2 2](#_Toc512519645)

[1. Use Case and Sequence Diagram 2](#_Toc512519646)

[2. Entity Relationship Diagram (ERD) 2](#_Toc512519675)

[3. Entity Properties 2](#_Toc512519678)

[4. Table Design 2](#_Toc512519697)

[REVIEW 2 – TASK SHEET 2](#_Toc512519716)

[REVIEW 3 2](#_Toc512519717)

[1. Site Map 2](#_Toc512519718)

[2. GUI Design 2](#_Toc512519721)

[REVIEW 3 – TASK SHEET 2](#_Toc512519770)

REVIEW 1

# INTRODUCTION

**Mystic Jewel** is a company which operates in the fine jewellery industry, as the name may imply. They started off as a small, local business, specialized in jewellery making and selling. After a few years in the business, **Mystic Jewel** has become one of the most trusted name in the local area. They are committed to make every customer feel special with their high-quality precious gems and their original designs. Their mission has always been the customers’ trust and satisfaction with the value of the products and fair pricing.

However, due to the lack of customers, they began to look for new platform to bring their products to potential buyers. They tried printing flyers and paying for advertisement on the newspaper, but both methods proved to be too costly yet ineffective.

# EXISTING SCENARIO

The store mainly sells jewellery made by their experienced jewellers. Since they are a small business, they cannot afford TV and billboards advertisements. Leaflets and ads on local newspapers are also unable to reach new customers. In addition to their problems, customers are now more and more inclined to looking for what they need online, where things are much cheaper due to low upkeep (unlike in a physical store where the business owner must worry about maintenance cost and rent).

Realizing traditional marketing platforms is not going to aid the sale, **Mystic Jewel** decided to take the Internet approach as E-Commerce has become the fastest growing platform for businesses. An E-Commerce site lets their customers to access the store remotely, allowing them to reach out and expand their customer base. Not only does a website eliminate the need for opening multiple stores in different locations, it can also save money on leaflets and physical advertisements like bill boards and posters.

# CUSTOMER REQUIREMENT SPECIFICATION

* The website must allow customers who visit the site to browse and search for products using search terms or filtering products’ categories and attributes. They can compare products.
* If the customer wishes to buy a product, they must register an account. A customer with an existing account can login. They do not have the permission to change any information apart from their own.
* A registered customer can leave reviews or inquiries on products and rate products. They can also bookmark products they wish to visit later in the future.
* The website must allow the admins (shop owner and others who are appointed to the job) to view, update (when requirements are met) existing products, categories and jewellery collections, a product’s attributes, customers who have registered, orders which have been placed, customer’s feedbacks on products, discount vouchers.
* If an admin wants to change sensitive information (e.g. delete a product, ban customers, etc.), said admin must have the authority level of a “Super Admin”, otherwise they are not going to be able to access these functions.
* Aside from managing products, categories, orders, etc. an admin can also view reports on products, orders and customers.

# FUNCTIONAL REQUIREMENT SPECIFICATION

## **User Can Register**

* 1. Description

Customers must be able to register a new account. Once they have their account, they can login and access more features. A guest, who is not logged in, can only browse products.

* 1. Functional Requirement

User must provide the information below to create an account:

* Username (must be unique)
* Password
* First Name and Last Name
* Email (must be unique)
* Date Of Birth
* Telephone number

## **User Can Login**

* 1. Description

After registering an account, user can login. Once they have logged in, they can modify their personal information, including their password. They can also access more features: product review, order products.

* 1. Functional Requirement

User must already have an existing account and must not be a banned account. User must provide Username (or alternatively their Email) and Password to login.

## **User Can Buy Products**

* 1. Description

User can find products they want by searching or filtering results. User can buy products by adding chosen products to cart and proceeding to check out. They will be asked to provide billing and payment information.

* 1. Functional Requirement

User must log in to use this function. User can only buy products that are still in stock. User needs to provide the information below to complete their order placement:

* Name of the receiver
* Required shipping date
* Shipping address
* Payment method
* Valid payment information (card owner, card number, security code)

## **User Can Write Reviews on Products**

* 1. Description

User can leave their review and rating on products. Each user can only post one review on each product. Rating can be done by choosing on a scale from 1 to 5, in which 5 is the best score. This score will be calculated and serve as the overall rating of the products.

* 1. Functional Requirement

User must login to use the review function. They must provide:

* Review title
* Review content
* A score on a scale from 1 to 5

## **User Can Bookmark Products**

* 1. Description

User can bookmark, also known as “Add to Favourite” a product they like and wish to visit later. They can view a list of products they have bookmarked and remove products from the list.

* 1. Function Requirement

User must login to use this function. User can choose a product they like and click the heart icon to add that product into their bookmarked list.

## **Recommended Products, User’s History**

* 1. Description

User can see a list of products that are recommended based on what they have purchased and which products are popular. They can also see a list of products which they have viewed before.

* 1. Function Requirement
* User will see products that are recommended for them.
* User will see products they have viewed.

## **User Can Compare Products**

* 1. Description

User can choose a maximum of 5 products from to compare. The products’ price and attributes will be compared so it is easier for user to choose which product they prefer.

* 1. Function Requirement
* User can choose no more than 5 products. If they want to choose another product, they must remove at least one product from the compared list

## **Admin Can Manage Categories**

* 1. Description

The site must allow admins to interact with Products’ Categories. An admin can view a list of existing categories, insert a new category, modify category’s name. When an admin is a Super Admin, they can delete sub-categories. Categories chosen for deletion must not contain any existing products. A category with no products and no sub-categories is an empty category.

* 1. Function Requirement

To add or modify a category, admin must provide:

* Category’s Name
* Category’s parent ID (parent ID is the ID of a category)

Other requirements:

* Must login as an admin to add or modify categories
* Sub-categories cannot be a parent
* Names are unique
* Parent ID must exist
* Must be a Super Admin to delete categories
* Can only delete empty categories
* Admin can search for categories by names and

## **Admin Can Manage Collections**

* 1. Description

The site must allow admins to interact with Collections. An admin can view a list of existing collections, insert a new collection, they can also modify a collection’s name, description, banner, image, trending status and active status. When an admin is a Super Admin, they can delete collections. Collections chosen for deletion must not contain any existing products. A collection with no products is an empty collection.

* 1. Function Requirement

To add or modify a collection, admin must provide:

* Collection’s Name
* Collection’s Description
* Collection’s Banner. This banner will appear on the site’s carousel.
* Collection’s Image.
* Trending status (whether collection shows on carousel or not)
* Active status (whether user can see collection or not)

Other requirements:

* Must login as an admin to add or modify collections
* Names are unique
* Banner and image are stored as file names
* Must be a Super Admin to delete collections
* Can only delete empty collections
* When create a new collection, created date can be recorded
* When modify a collection, modified date can be recorded
* Admin can search collections by names

## **Admin Can Manage Attributes List**

* 1. Description

The site must allow admins to interact with Attribute Groups and Values. An admin can view a list of existing attributes group. Admin can add a new group and modify a group’s name. Admin can also add new values to a group and modify these values. When an admin is a Super Admin, they can delete attributes and values. Attribute groups chosen for deletion must not contain any values. Values chosen for deletion must not belong to any product.

* 1. Function Requirement

To add or modify attribute groups and values, admin must provide:

* Group’s Name
* Attribute’s Name (value)
* Group ID (which group that value belongs to)

Other requirements:

* Must login as an admin to add or modify attributes
* Groups’ names are unique
* Must be a Super Admin to delete attributes groups and values
* Can only delete group which do not contain any values
* Can only delete values which do not belong to any products
* Admin can search attributes by names and groups

## **Admin Can Manage Products,**

* 1. Description

The site must allow admins to interact with Products. An admin can view a list of existing products. Admin can add a new product and modify a product’s name, description, status, category, collection, discount value, start and end dates of discount. When an admin is a Super Admin, they can delete products. Products chosen for deletion must not belong to any wish lists and must contain no reviews. Admin can view a product’s details when click on the product.

* 1. Function Requirement

To add or modify a product, admin must provide:

* Product’s Name
* Product’s Description
* Active status (whether user can see product or not)
* Category which product belongs to
* Collection which product belongs to
* Discount value in percent
* Discount start date
* Discount end date

Other requirements:

* Must login as an admin to add or modify products
* Names are unique
* Category must exist
* Collection must exist
* Discount must be between 0 to 75
* A status of a product is always in active if it does not have at least one SKU
* Admin must add at least one SKU to set product’s status to active
* Must be a Super Admin to delete products
* Can only delete products which do not belong to any wish lists and have no reviews by customers
* When create a new product, created date can be recorded
* When modify a product, modified date can be recorded
* Admin can search products by names
* Admin can filter by categories, collections and attributes

## **Admin Can Manage Product’s Details (Images, Attributes, Reviews)**

* 1. Description

The site must allow admins to interact with Products’ Details. An admin can view a list of images and attributes of a product. Admin can add new images and modify existing images. Admin can modify a product’s attributes like adding and removing attributes. An admin can also see customers’ rating and reviews on a product and can response to customer’s review.

* 1. Function Requirement

To modify a product’s details, admin must provide:

* Specific product
* Image’s Link
* Attributes
* Attribute Groups
* Admin’s response to a review

Other requirements:

* Must login as an admin to add or modify products
* Can add multiple images
* Attributes must exist
* When modify a product’s details, modified date can be recorded

## **Admin Can Manage Stock Keeping Units (SKUs)**

* 1. Description

The site must allow admins to interact with Products’ SKUs. SKUs are for keeping track of stock and price as each product may have multiple size options. Sizes are linked to categories (e.g. ring size, necklace size). An admin can view a list of existing SKUs of a product. Admin can add new SKUs to a product and modify a SKU’s ID, Size of that SKU, Base Price, Unit Price, Units in Stock and Units on Order. When an admin is a Super Admin, they can delete SKUs. SKUs chosen for deletion must not belong to any orders.

* 1. Function Requirement

To add or modify SKUs, admin must provide:

* SKU ID (generated)
* Product ID (product which the SKU is a variant of)
* Base Price
* Unit Price
* Units in Stock
* Units on Order

Other requirements:

* Must login as an admin to add or modify SKUs
* Products must exist
* SKU ID is generated from a product’s category, size and product ID
* Size must be appropriate for product’s category
* Must be a Super Admin to delete SKUs
* Can only delete SKUs which do not belong to any orders
* If all SKUs of the same product are deleted, that product’s status will be automatically set to inactive
* When create a new SKU, created date can be recorded
* When modify a SKU, modified date can be recorded
* Admin can search SKUs by SKU ID
* Admin can filter SKUs by product ID, price range and stock
* New SKU can be added by adding a new product or update an existing product

## **Admin Can Manage Customers**

* 1. Description

The site must allow admins to interact with Customer’s Accounts. When a customer has registered an account, they will show up in customer list. An admin can view a list of existing registered customers. Admin can ban a customer who violates site’s rules by changing customer’s status.

* 1. Function Requirement
* Must login as an admin to ban customers
* Admin can search customers by Username and Emails
* Admin can filter customers by registered date and status
* Admin can ban customers
* Admin must provide: customer’s Active status

## **Admin Can Manage Orders**

* 1. Description

The site must allow admins to interact with Orders. When a customer places an order, it will show up in the list of orders and these orders’ details. An admin can view a list of existing orders and see their statuses. Admin can change order status. Orders are considered highly important data and cannot be deleted.

* 1. Function Requirement

An Order will contain:

* Customer ID
* Voucher ID if available
* Shipping fee
* Shipping Name
* Shipping address
* Shipping date
* Payment method
* Status: Pending, Confirmed, Completed, Cancelled

Other requirements:

* Must login as an admin to modify order’s status
* Order must show created date
* Order must show updated date
* Admin can search orders by customers
* Admin can filter customers by created date

## **Admin Can Manage Vouchers**

* 1. Description

The site must allow admins to interact with Voucher. An admin can view a list of existing Vouchers. Admin can add new vouchers and modify a voucher’s ID, Description, Value, Quantity and Valid Date. When an admin is a Super Admin, they can delete vouchers. Vouchers chosen for deletion must not belong to any orders.

* 1. Function Requirement

To add or modify Vouchers, admin must provide:

* Voucher ID
* Voucher’s Description
* Voucher’s Value (in percent)
* Quantity
* Valid time: start and end dates

Other requirements:

* Must login as an admin to add or modify vouchers
* Value must be between 0 to 75
* Can only delete vouchers which do not belong to any orders
* Admin can search vouchers by ID

## **Admin Can Manage Admin’s Information**

* 1. Description

The site must allow Super Admin to interact with Voucher. An authorized Super Admin can view a list of admins. A Super Admin can add new admins and modify existing admins. Admins cannot be deleted.

* 1. Function Requirement

To add or modify admins’ information, Super Admin must provide:

* Admin ID (for reference, cannot be changed)
* Username
* Password
* Role: Admin, Super Admin

Other requirements:

* Must login as a Super Admin to add or modify admins
* Usernames are unique
* When create an account, created date can be recorded
* When modify an account, modified date can be recorded
* Can search admins by id, usernames and roles

## **Admin Can View Admin’s Log**

* 1. Description

The site must allow admins to view admin’s activities log.

* 1. Function Requirement

Activity Log includes:

* Admin’s activity
* Admin who performs the activity
* Date performed

## **Admin Can Manage Reports**

* 1. Description

The site must allow admins to view reports of Products, Customers and Orders. Product report shows products by their sold quantity, the revenue of each individual product and allows admins to see which products need to be improved (low sale). Customer report shows customer’s activities such as the number of new customers, their orders and how much money a customer has spent. Order report shows the number of orders that have been placed and the revenues. Admin can export reports as a pdf file. Admin can also view the general report.

* 1. Function Requirement
* Must login as an admin to use report function
* Product: revenue for each product, revenue by category
* Order: total number of orders, total revenue
* Customer: total number of customer, customer’s activities

# SYSTEM REQUIREMENTS

## **Hardware**

|  |  |  |
| --- | --- | --- |
|  | SERVER | |
| Minimum | Recommended |
| Processor | 800 MHz Intel Pentium III | 2.6 GHz Intel Pentium IV |
| Memory | 2 GB | 4 GB |
| Disk space | 8 GB | 16 GB |
| Monitor | Super-VGA (800x600) or higher resolution monitor | |
| Internet | Internet access is required | |
|  | CLIENT | |
| Processor | 800 MHz Intel Pentium III | |
| Memory | 512 MB | |
| Monitor | Super-VGA (800x600) or higher resolution monitor | |
| Internet | Internet access is required | |

## **Software**

|  |  |
| --- | --- |
|  | SERVER |
| Operating System | Windows XP or later |
| Browser | Google Chrome 65.0 |
| Database | Microsoft SQL Server 2014 |
| Software | Java SDK 1.8 |
| GlassFish Server 4.0 |
|  | CLIENT |
| Operating System | Windows XP or later |
| Browser | Google Chrome 65.0 |

# DEVELOPMENT SOFTWARE

* Microsoft Windows 10
* Java SDK 1.8
* Netbeans IDE 7.4
* GlassFish Server 4.0
* Microsoft SQL Server 2014
* Google Chrome 65.0

# TECHNOLOGY

* J2EE
* JSP, Servlet
* EJB
* HTML/CSS
* JavaScript
* AJAX
* Jquery
* Apache Tiles Framework

REVIEW 1 – TASK SHEET

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Contents | Member | Activity Plan | Preparation | |
| Start Date | Actual Days | Status |
| 1 | Introduction | Du Hue Phuong  Nguyen Khai Nguyen  Huynh Tan Phat | 05/04/2018 | 3 | Completed |
| 2 | Existing Scenario |
| 3 | Customer Requirement Specification |
| 4 | Functional Requirement Specification |
| 5 | System Requirements |
| 6 | Development Software |
| 7 | Technology |

Plan prepared by: Group 2 April 10th, 2018  
 **Group Leader Faculty**

**Du Hue Phuong Mr. Ngo Phuoc Nguyen**

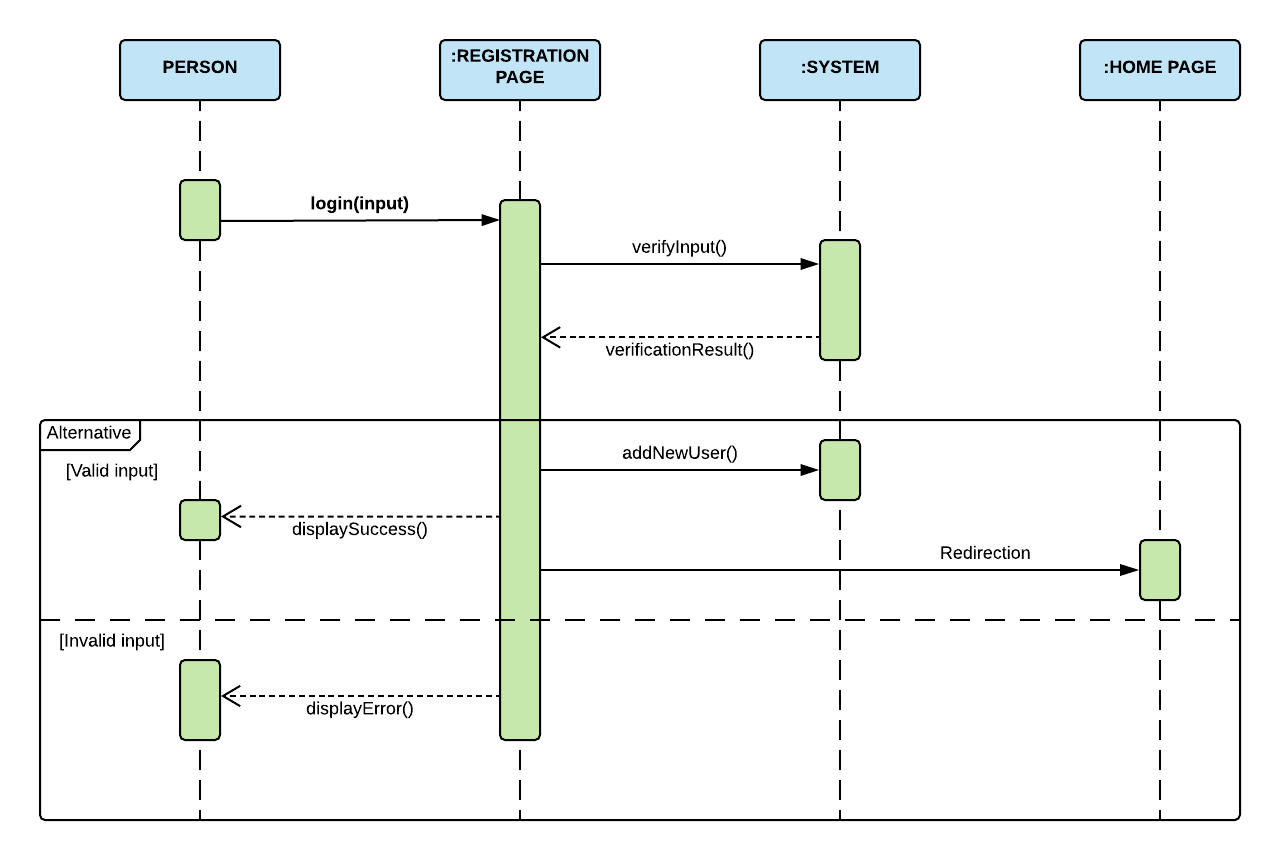
REVIEW 2

# Use Case and Sequence Diagram

## **User**

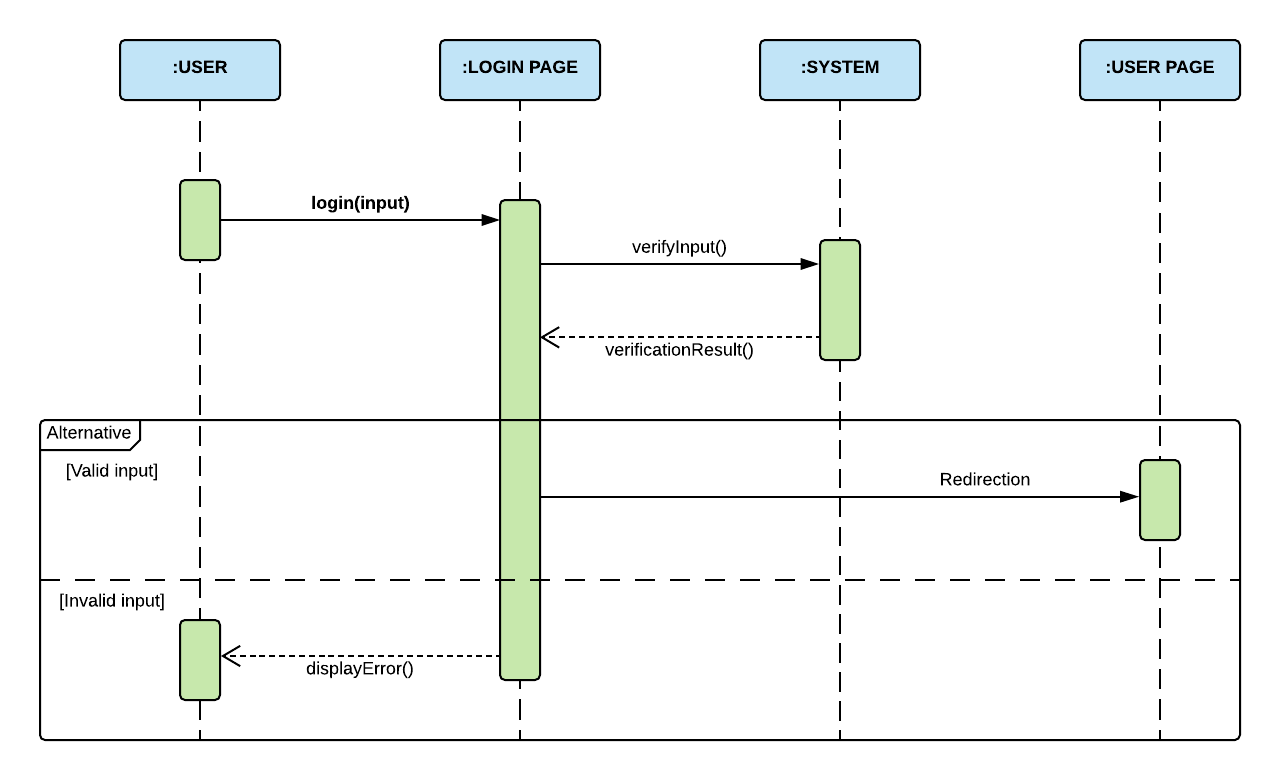
### Register as Customer

|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use Case Name** | Register as Customer | |
| **Actors** | Guest | |
| **Description** | Guest registers an account on the website and becomes a member (customer) | |
| **Requirements** | Guest provides their register information | |
| **Pre-conditions** | N/A | |
| **Post-conditions** | Success: Guest creates a Customer account in this website | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor clicks ‘Register’ link on Home Page.            3. Actor inputs register’s information and click ‘Register’ button. | 2. System redirects to Register page and displays Registration form with the following controls: - ‘Username’ text field - ‘Password’ text field - ‘Confirm Password’ text field - ‘First Name’ text field - ‘Last Name’ text field - ‘Email’ text field - ‘Phone1’ text field - ‘Register’ button - ‘Reset’ button 4. System validates the information. 5. System inserts the account into database  6. System show success message. |
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| - Actor left ‘Username’ text field blank. - Username has already existed. - Actor left ‘Password’ text field blank. - Actor left ‘Confirm Password’ text field blank. - Password and confirm password don’t match. - Actor left ‘Name’ text field blank. - Actor left ‘Birthday’ field blank. - Input age is less than 16. - Actor left ‘Phone’ text field blank. - Input phone is not numeric format or phone length is not between 8 and 14 characters. - Actor left ‘Email’ text field blank or input an invalid email address. | System displays message to actor: - ‘Username cannot be blank’. - ‘This Username has already existed’. - ‘Password cannot be blank’’. - ‘Confirm Password cannot be blank’. - ‘Confirm Password does not match password.’ - ‘Name cannot be blank’. - ‘Birthday is invalid’. - ‘Age must be over 16’. - ‘Phone is invalid’. - ‘The phone number should be numeric only and must be between 8 and 14 digits’. - ‘Email is invalid’. |



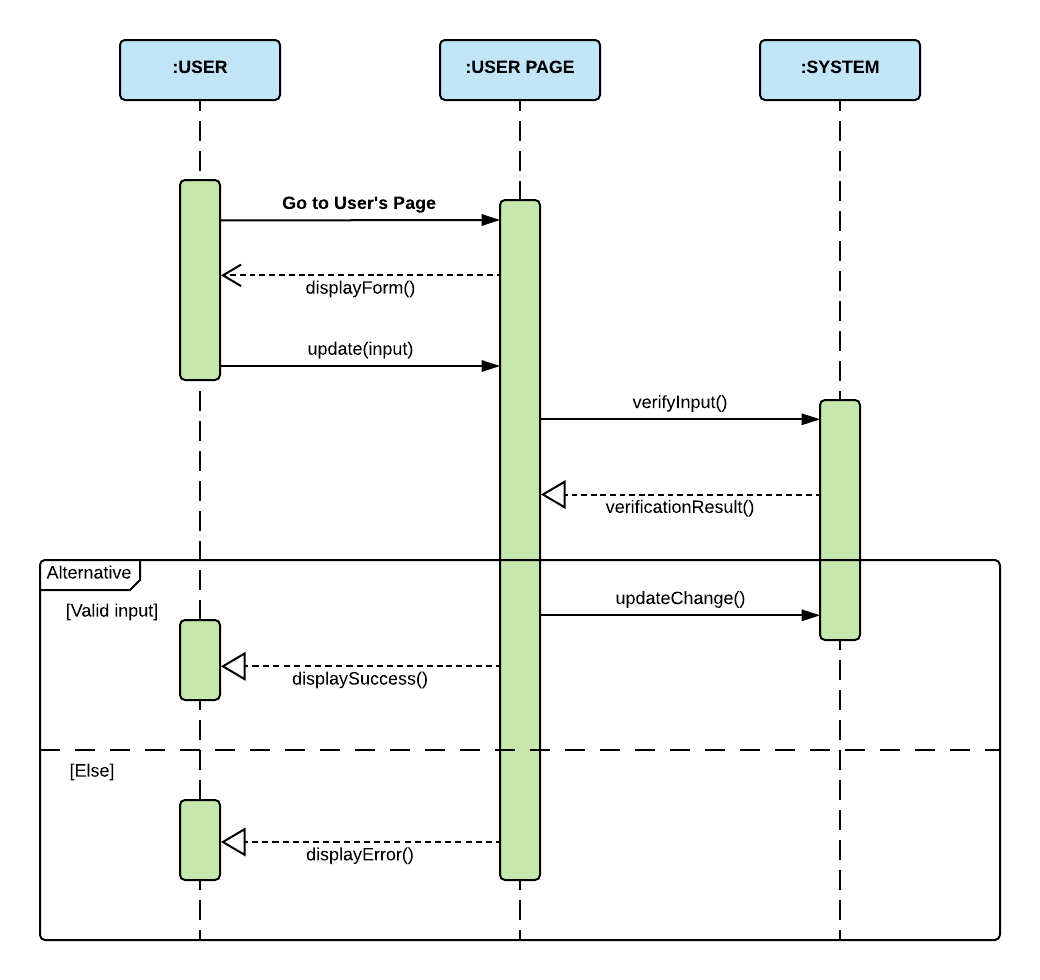
### Login as Customer

|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use Case Name** | Login | |
| **Actors** | Customer | |
| **Description** | Customer who has registered an account can login | |
| **Requirements** | Actor provides username and password | |
| **Pre-conditions** | Actor must register an account | |
| **Post-conditions** | Success: User is logged in to website | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor clicks on ‘Login’ button.     3. Actor inputs Username (or email) and Password, then click the ‘Login’ button. | 2. System redirects to Login Page with the following controls - 'Username' text field - 'Password' text field - 'Login' button 4. System checks the information. 5. System redirects to User’s dashboard. |
|
|
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor inputs invalid username (or email) and password. | System redirects to Login page with the following controls: - ‘Username’ text field - ‘Password’ text field - ‘Login’ button. System shows message: “Invalid Username and password”. |



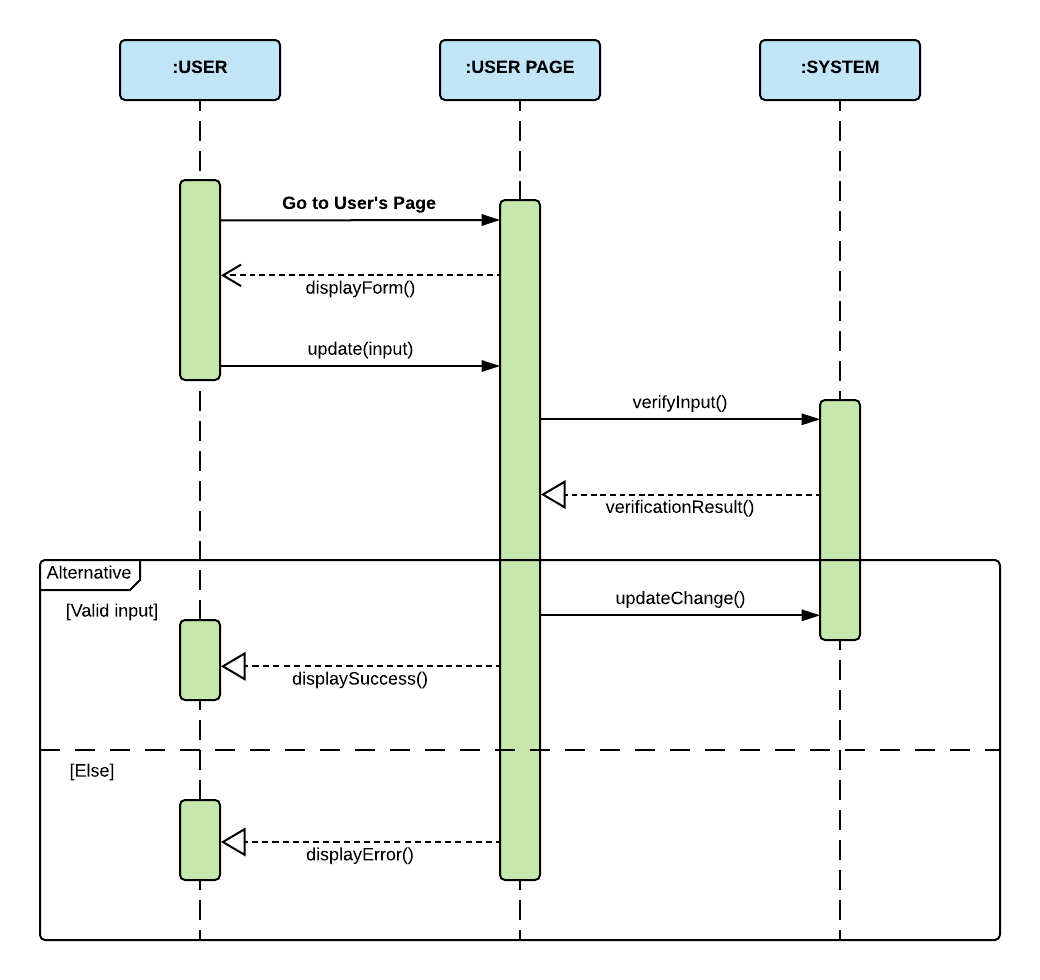
### Update Customer’s Profile

|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use Case Name** | Update Customer’s Profile | |
| **Actors** | Customer | |
| **Description** | Customer wants to Update their profile’s information. | |
| **Requirements** | Actor provides profile information | |
| **Pre-conditions** | Actor has logged in. | |
| **Post-conditions** | Success: Profile’s information of this user has been updated. | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor clicks ‘Profile’ button in Actor’s Home page menu.              3. Actor updates information and click ‘Save’ button. | 2. System redirects to Update Profile page and displays Update Profile form with the following controls :  - ‘Old Password’ text field - ‘New Password’ text field - ‘New Password Confirm’ text field - ‘First Name’ text field  - ‘Last Name’ text field - ‘Birthday’ field  - ‘Gender’ radio button  - ‘Address’ text field - ‘Phone’ text field (x2) - ‘Email’ text field - ‘Save’ button - ‘Cancel’ button 4. System checks the information. 5. System updates the account in database and then redirects to User’s Dashboard |
|
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘New Password’ text field blank. 2. Actor left ‘Confirm Password’ text field blank. 3. New Password and New Password Confirm do not match. 4. Actor left ‘Name’ text fields blank. 5. Actor left ‘Birthday’ field blank. 6. Input age in ‘Birthday’ field is less than 16. 7. Actor left ‘Address’ text field blank. 8. Actor left ‘Phone’ text fields blank. 9. Input phone is not in numeric or phone length is not in between 8 and 14 characters. 10. Actor left ‘Email’ text field blank or input an invalid email address. | System displays message to actor : - ‘Password cannot be blank’’. - ‘Confirm Password can’t be blank’. - ‘Confirm password does not match Password.’= - ‘Name cannot be blank’. - ‘Birthday is invalid’. - ‘Age must be over 16’. - ‘Address cannot be blank’. - ‘Phone is invalid’. - ‘The phone number should be numeric only and must be between 8 and 14 digits’. - ‘Email is invalid’. |

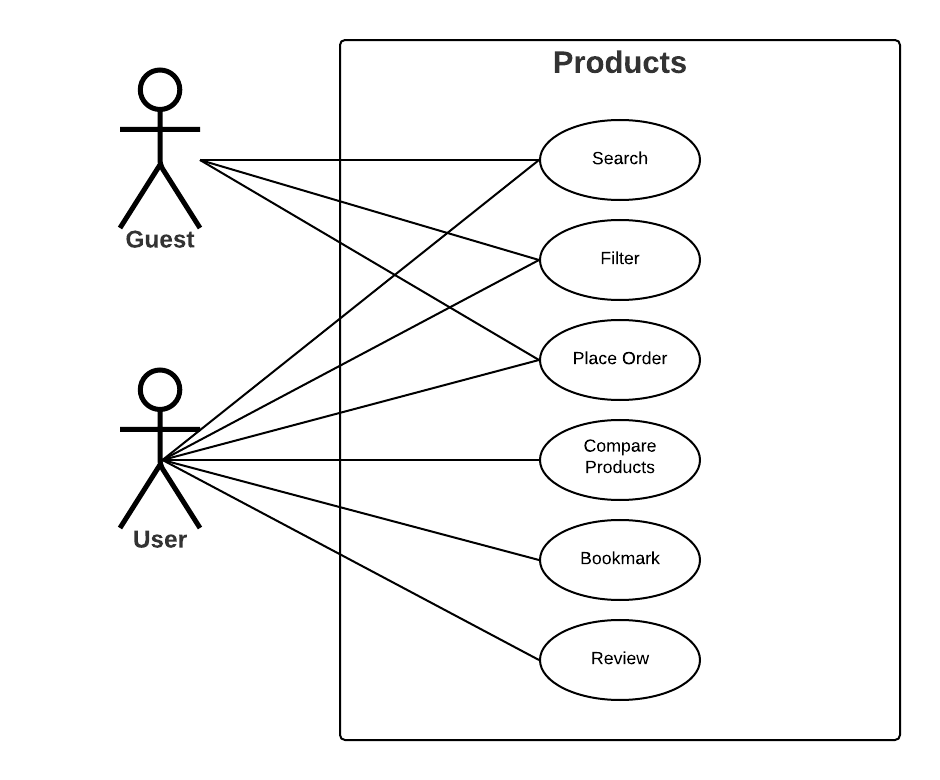


### Forgot Password

|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use Case Name** | Forgot Password | |
| **Actors** | Customer | |
| **Description** | Customer who has forgotten password can make a request to change password | |
| **Requirements** | Actor provides email address | |
| **Pre-conditions** | Actor must be logged on | |
| **Post-conditions** | Success: Actor can change their password | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor clicks on ‘Forgot Password’ button.   3. Actor inputs Email and clicks ‘Submit’.  5. Actor inputs reset token in Password Page and click ‘Submit’.  7. Actor inputs new password and click ‘Save’. | 2. System redirects to Password Page with the following controls: - 'Email' text field - 'Submit' button 4. System checks the information and sends a reset token to user. 6. System redirects to Password form with the following controls:  - ‘New Password’ text field  - ‘Confirm Password’ text field  - ‘Save’ button  8. System validates password then updates password in database.  9. System shows message: ‘Your password has been changed successfully’. |
|
|
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor inputs an email which does not exist in database.  2. Actor inputs authentication code incorrectly. | System shows messages:  - ‘Invalid email’.  - ‘Invalid authentication code’. |

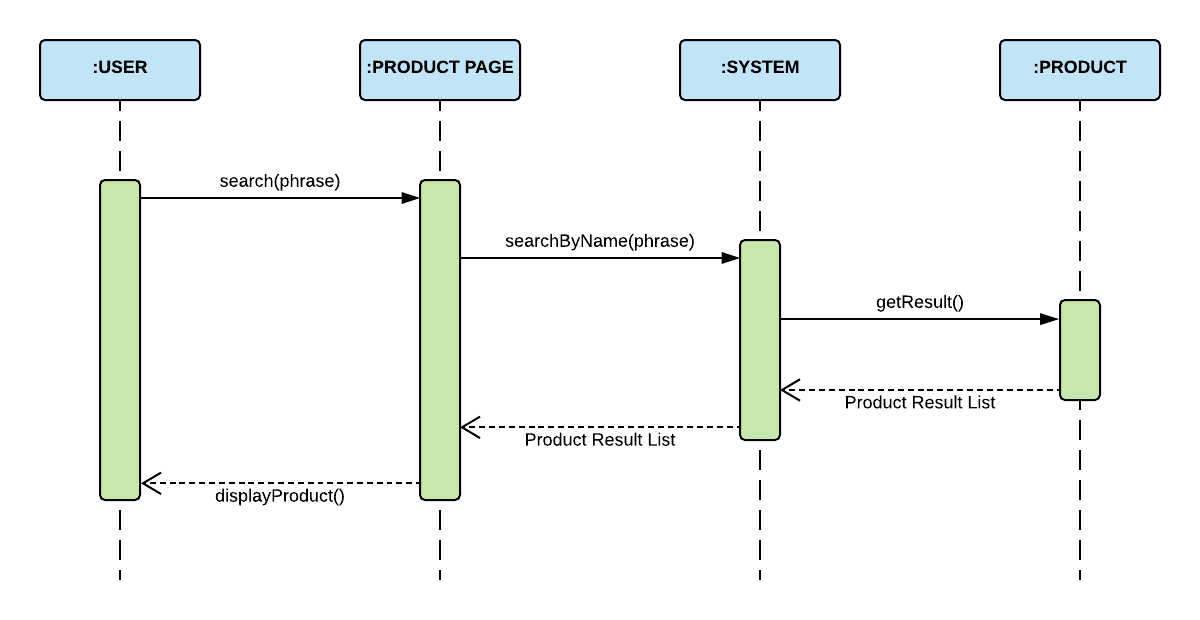


## **Product**



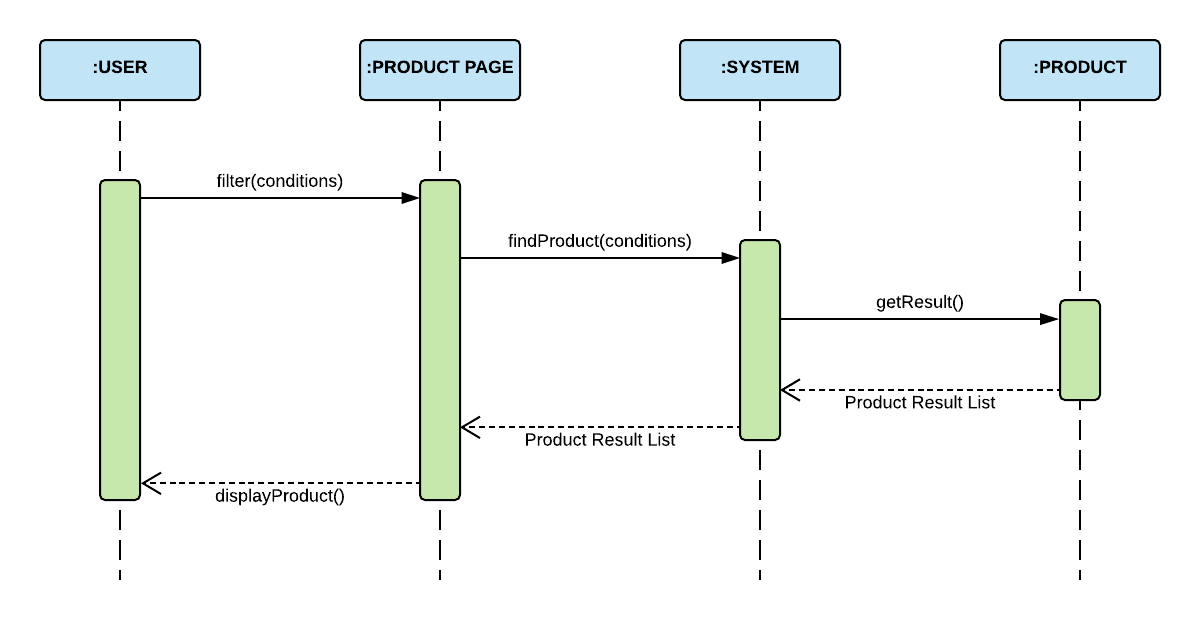
### Search Products

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use Case Name** | Search products by relative name | |
| **Actors** | Guest/Customer | |
| **Description** | Guest/Customer wants to search for products by name | |
| **Requirements** | Actor provides search phrase | |
| **Pre-conditions** | N/A | |
| **Post-conditions** | Success: Show products that matches search phrase | |
| Fail: No product is shown | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor inputs search phrase.  3. Actor clicks product’s name in list or view every product that matches search phrase. | 2. System show list of products that match the search phrase.  4. System allows actor to sort products by:  - Oldest  - Newest  - Lowest Price  - Highest Price |
|
|
|
|
|
|
|
|
|



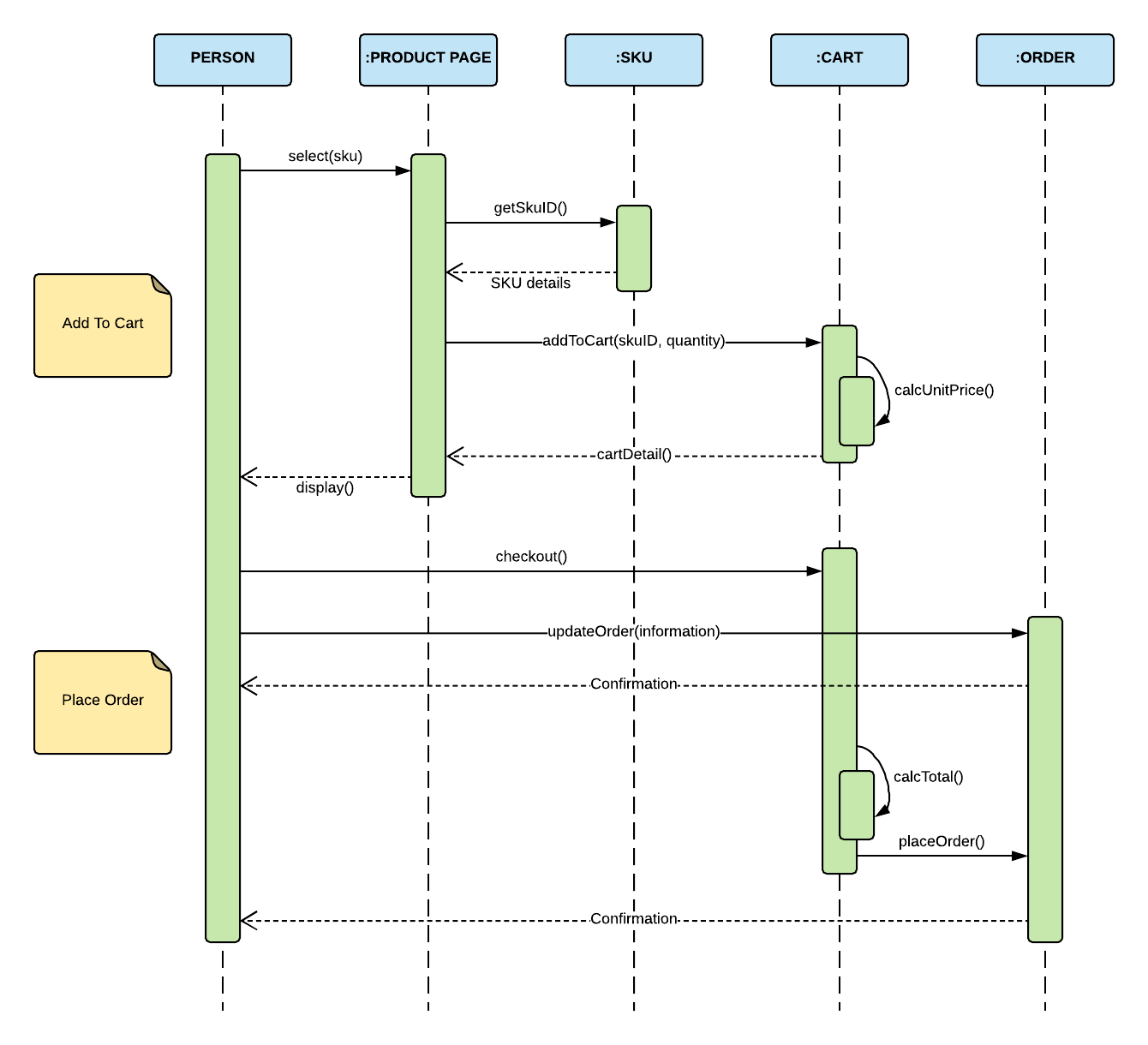
### Filter Products

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use Case Name** | Product Filter | |
| **Actors** | Guest/Customer | |
| **Description** | Guest/Customer wants to filter products by one or multiple conditions | |
| **Requirements** | Actor chooses conditions by ticking checkboxes | |
| **Pre-conditions** | N/A | |
| **Post-conditions** | Success: Products that matches condition(s) are shown | |
| Fail: No product is shown | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor ticks one or multiple of the following checkboxes:  - Categories  - Collections  - Attributes  - Price | 2. System show products that satisfy the condition(s) 3. System allows actor to sort products by:  - Oldest  - Newest  - Lowest Price  - Highest Price |



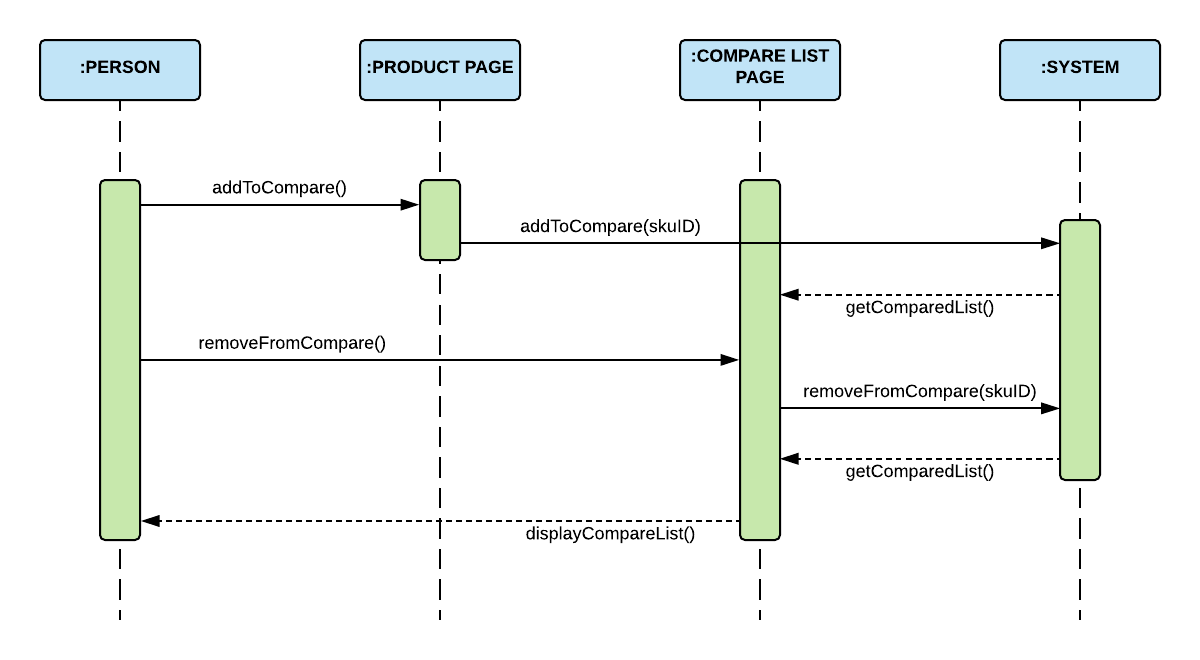
### Place Orders

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use Case Name** | Checkout and Place Order | |
| **Actors** | Guest/Customer | |
| **Description** | Guest/Customer wants to order products | |
| **Requirements** | Actor must choose at least one product to add to cart | |
| **Pre-conditions** | Cart has at least one product | |
| **Post-conditions** | Success: Actor order successfully | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses a product and click ‘Add to Cart’ button.  3. Actor clicks ‘Checkout’ button.  4. Actor inputs Voucher Code and clicks ‘Apply Voucher’ button.  8. Actor inputs order information, then click ‘Place Order’ button.  10. Actor inputs Card Owner’s Name, Card Number and Security Code.   13. Actor views order’s status in Order Page. | 2. System adds product to cart by Sku ID.  4. System shows Cart and added products. 5. System verifies voucher code.  6. System shows all products in cart and calculate unit price.  7. System redirects to Order Information page and show the following controls:  - ‘Billing Name’ text field  - ‘Address’ text field  - ‘Required date’  - ‘Payment’ checkboxes  8. Systems show Payment Page if user chooses Pay with Card method.  11. System validates information. 12. System redirects to Order page and shows order’s status. |
|
|
|
|
|
|
|
|
| **Alternative flow** | 1. Actor clicks ‘Go back to shopping’  3. Actor clicks ‘X’ button on each product. | 2. System redirects to product page.  4. System removes product from cart. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor inputs invalid voucher code.  2. Actor inputs more products than there are in stock.  3. Actor left ‘Billing Name’ text field blank.  4. Actor left ‘Address’ text field blank.  5. Actor left ‘Required date’ blank.  6. Actor left ‘Payment’ checkboxes blank.  7. Actor inputs invalid card information. | System shows messages:  - ‘Invalid voucher’  - ‘Product out of stock’  - ‘Billing name must not be blank’  - ‘Address must not be blank’  - ‘Must choose a required date’  - ‘Must choose a payment method’  - ‘You are using an invalid card’ |



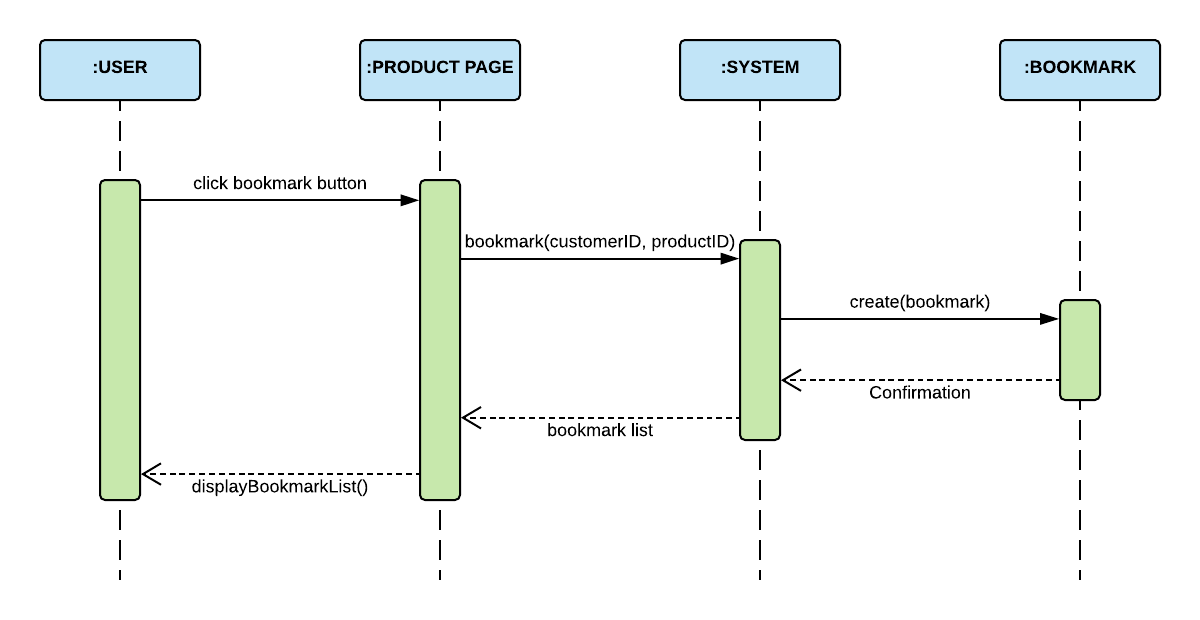
### Compare Products

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use Case Name** | Compare products | |
| **Actors** | Guest/Customer | |
| **Description** | Customer compares products | |
| **Requirements** | Customer choose products they want to compare | |
| **Pre-conditions** | N/A | |
| **Post-conditions** | Success: show a list of compared products | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor adds click ‘compare’ button on product.  3. Actor clicks ‘Compare List’ button in the menu. | 2. System adds product into compared list and shows the number of products which are added (default: 0).  4. System redirects to Compare List Page and shows all compared products. |
| **Alternative flow** | 1. Actor clicks ‘Remove this item’ button. | 2. System removes product form compared list. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor added more than 5 products. | 2. System does not add product and redirects to Compare List Page. |



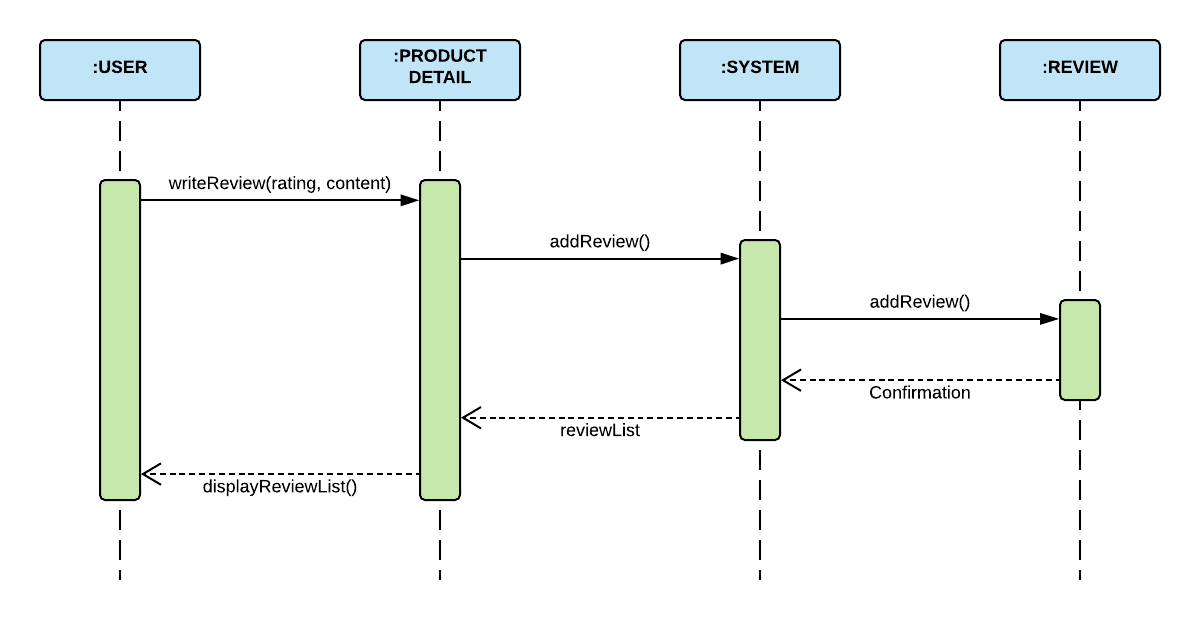
### Bookmark Products

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use Case Name** | Bookmark product | |
| **Actors** | Customer | |
| **Description** | Customer wants to bookmark products to view it later | |
| **Requirements** | Customer choose products they want to bookmark | |
| **Pre-conditions** | Actor is logged in | |
| **Post-conditions** | Success: show bookmarked products in customer's bookmark list | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor clicks ‘bookmark’ icon on each product in Product Page.  3. Actor clicks ‘Wish List’ link. | 2. System adds bookmark in database.  4. System shows list of bookmarked products. |

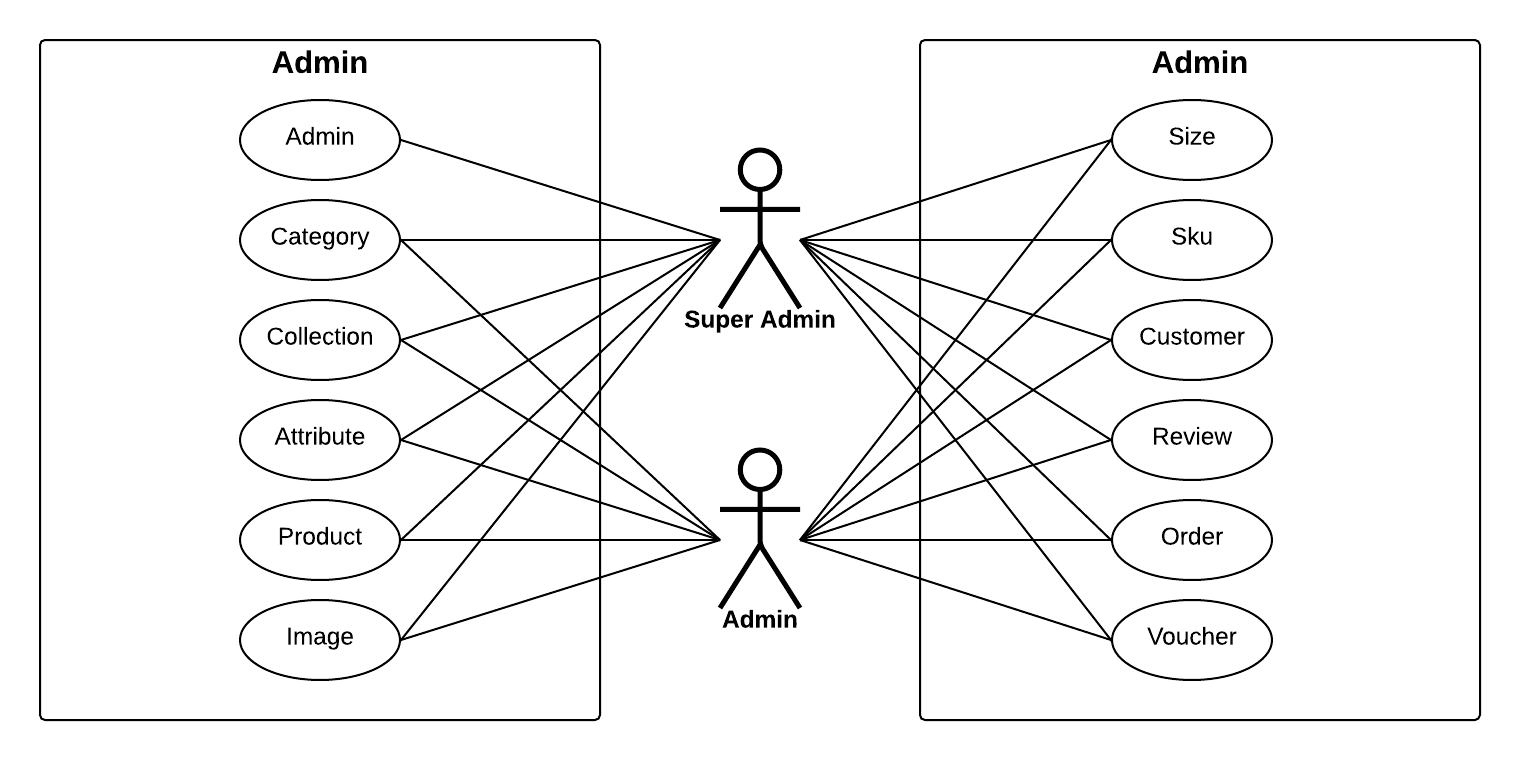


### Write Reviews for Products

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use Case Name** | Review product | |
| **Actors** | Customer | |
| **Description** | Customer wants to leave review and rating on products | |
| **Requirements** | Customer provides their rating and review | |
| **Pre-conditions** | Actor is logged in and has bought product | |
| **Post-conditions** | Success: show actor's review on that product | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor clicks on a product to see that product’s details.  3. Actor chooses rating score out of 5 stars and writes review content. | 2. System redirects to product’s details page.  4. System adds review to database and shows actor’s review in product’s detail page. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor does not rate product.  2. Actor does not write title.  3. Actor does not write content. | System displays messages:  - ‘You must rate before submit’  - ‘Review must have a title’  - ‘Content cannot be blank’ |



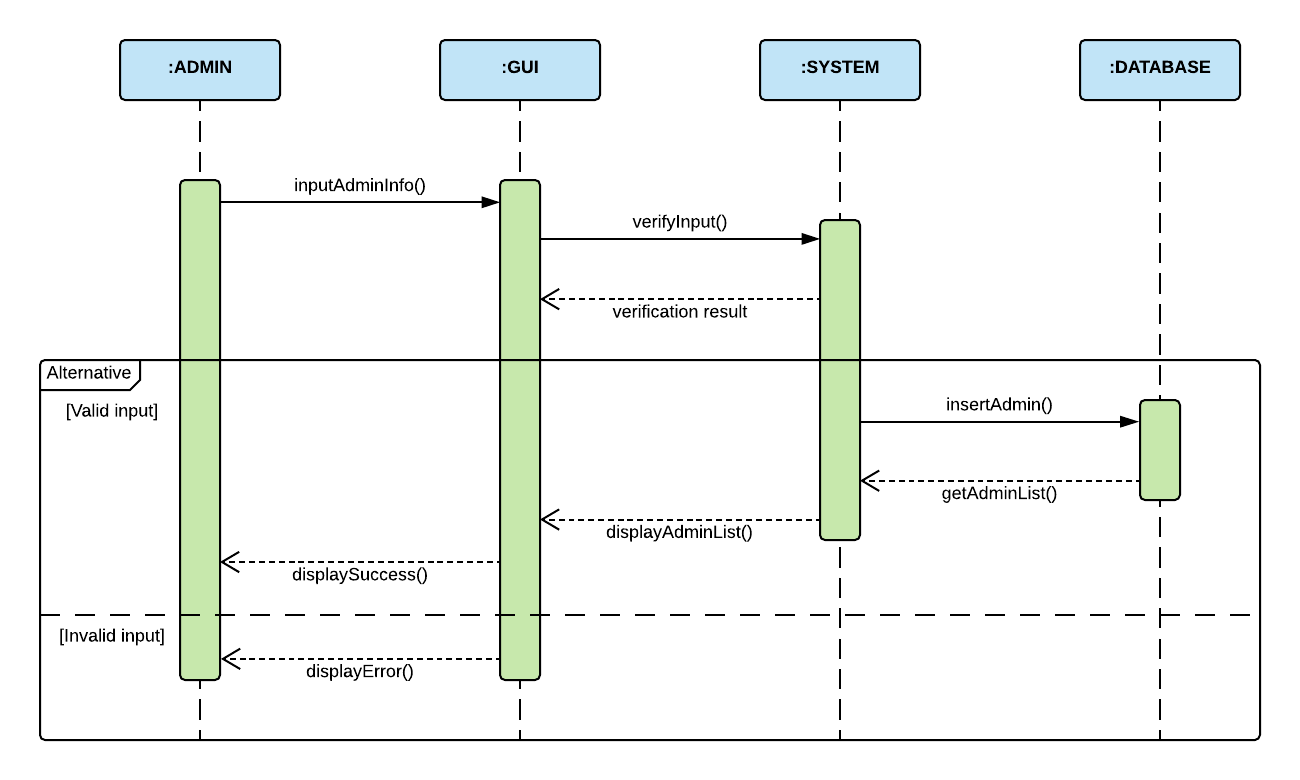
## **Admin**



### Admin’s Account

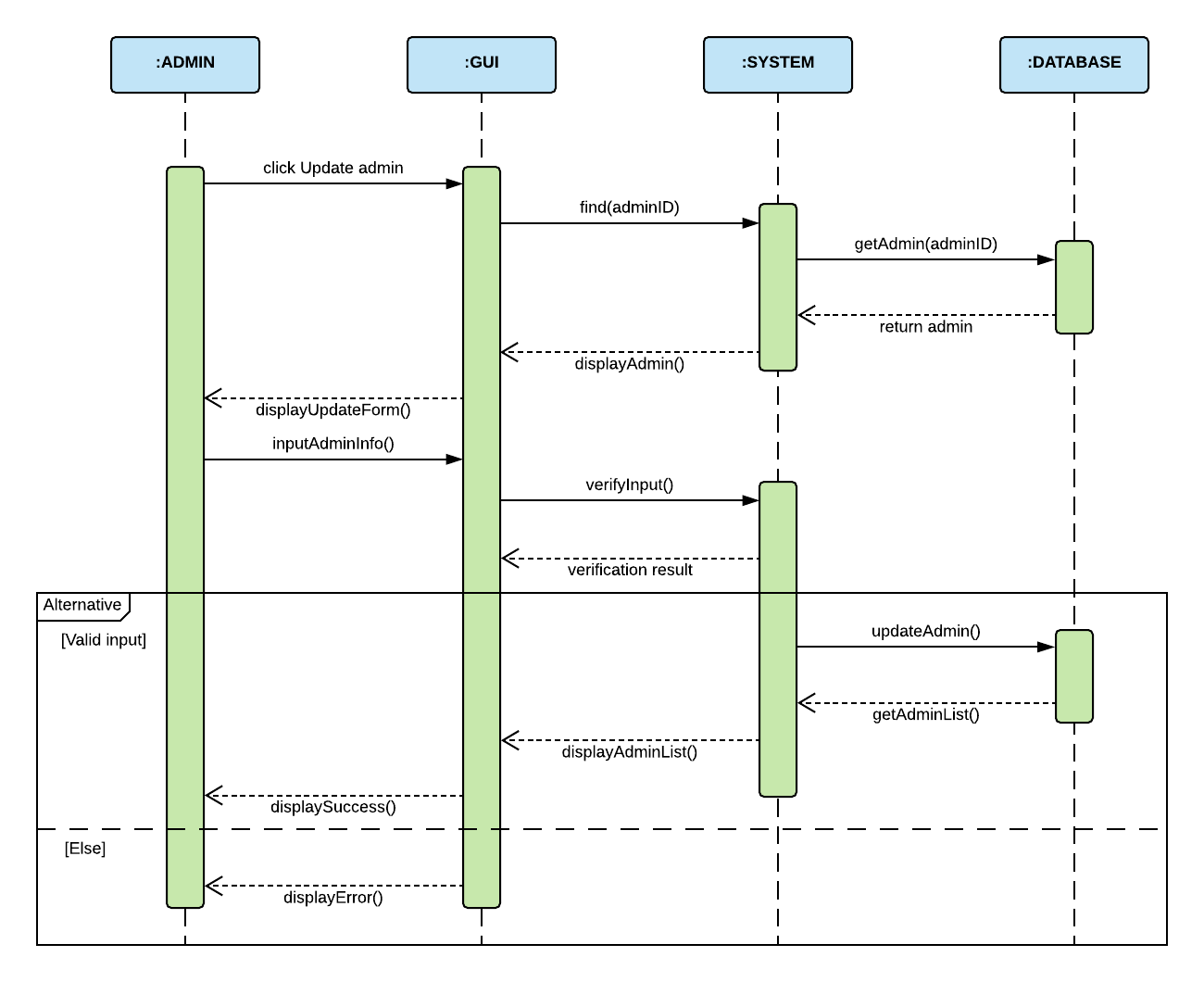
1. **Add Admin’s Account**

|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use case name** | Add admin’s accounts | |
| **Actors** | Super Administrator | |
| **Description** | When admin wants to add a new admin’s account in the database | |
| **Requirements** | Super Admin provides admin information | |
| **Pre-conditions** | Actor is logged in as a Super Admin | |
| **Post-conditions** | Success: a new admin’s account is created in the database | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Admin Tab on menu.  3. Actor chooses ‘Create an Admin Account’ button.  5. Actor inputs information to create admin. | 2. System redirects to 'Admin List' page and display a list of existing admin.  4. System displays admin form with the following fields. - Admin’s Username - Admin’s password  - Confirm password - Admin’s role  6. System validates inputs.  7. System adds new admin to database.  7. System displays confirmation message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Username’ field blank.  2. Actor inputs an existing Username.  2. Actor left ‘Password’ field blank.  3. ‘Confirm password’ does not match ‘Password’. | System displays messages: - ‘Username’ cannot be blank’.  - ‘Username already exists’ - ‘Password cannot be blank’.  - ‘Confirm password does not match password’ |
|
|



1. **Update Admin**

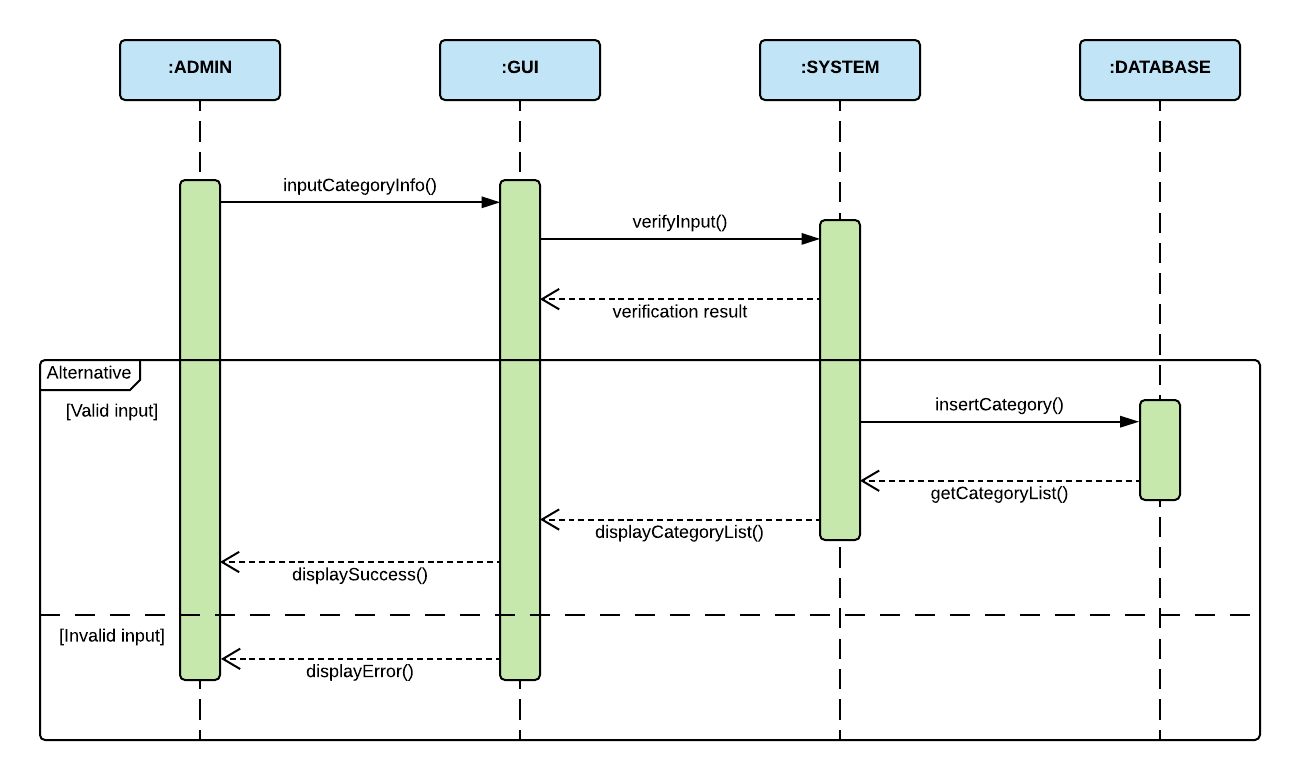
|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use case name** | Update admin’s accounts | |
| **Actors** | Super Administrator | |
| **Description** | When admin wants to update an admin’s account in the database | |
| **Requirements** | Super Admin provides admin information | |
| **Pre-conditions** | Actor is logged in as a Super Admin | |
| **Post-conditions** | Success: admin’s account is updated in the database | |
| Fail: Refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Admin Tab on menu.  3. Actor chooses Admin they want to update.  5. Actor inputs information to update admin. | 2. System redirects to 'Admin List' page and display a list of existing admin.  4. System displays admin form with the following fields. - Admin’s password  - Confirm password - Admin’s role  6. System validates inputs.  7. System updates admin in database.  8. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Username’ field blank.  2. Actor inputs an existing Username.  2. Actor left ‘Password’ field blank.  3. ‘Confirm password’ does not match ‘Password’. | System displays messages: - ‘Username’ cannot be blank’.  - ‘Username already exists’ - ‘Password cannot be blank’.  - ‘Confirm password does not match password’ |
|
|



### Category

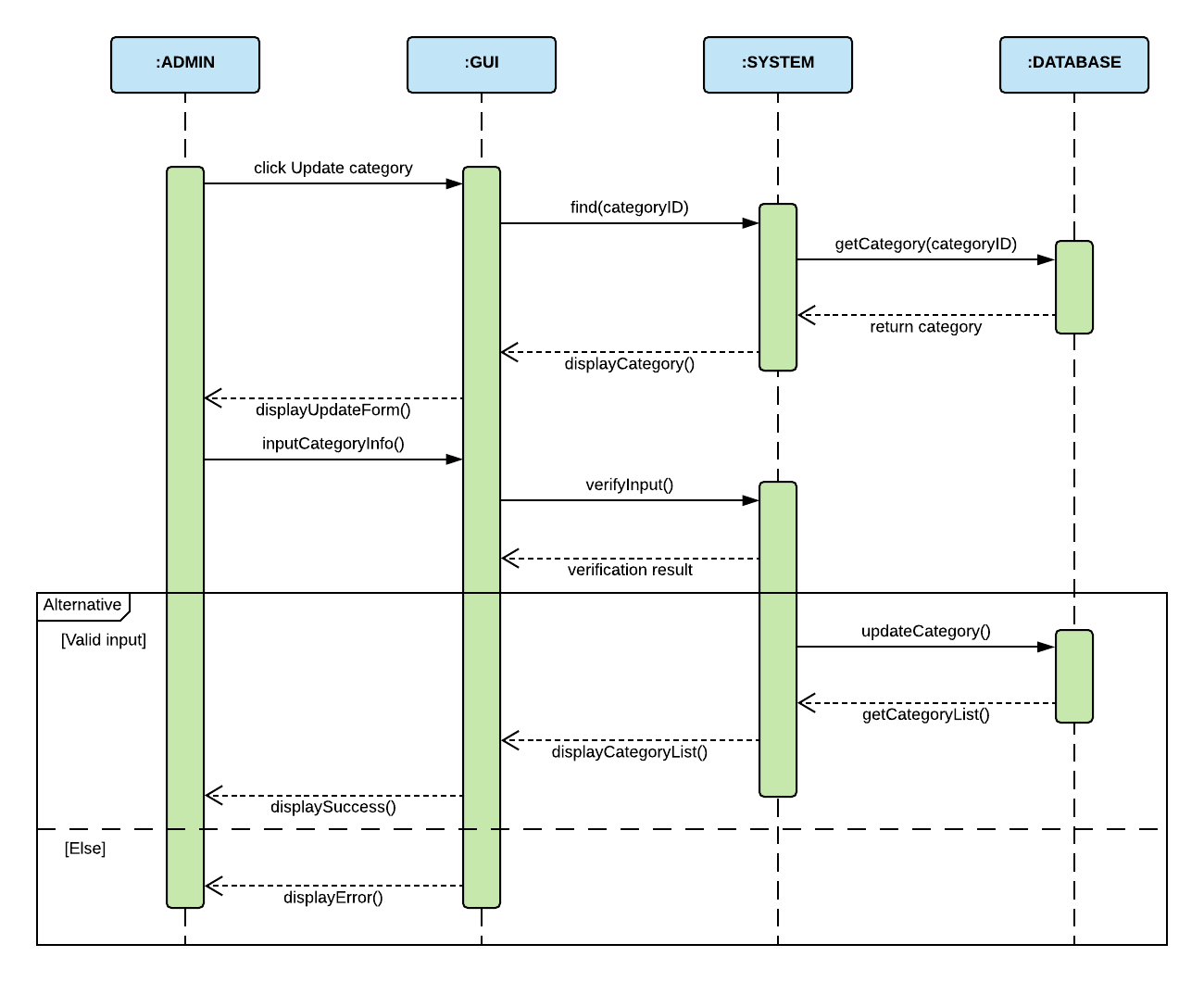
1. **Create new category**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Add product’s categories | |
| **Actors** | Administrator | |
| **Description** | When admin wants to add a new category in the database | |
| **Requirements** | Actor provides category’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: category is added in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Category Tab on menu.  3. Actor clicks ‘Add New Category’.  5. Actor inputs information to create category. | 2. System redirects to 'Category List' page and display a list of existing categories.  4. System displays a form with the following fields. - Category’s Name - Parent category  6. System validates inputs.  7. System add category into database.  8. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Category’s Name’ field blank.  2. Category’s Name existed in database. | System displays messages: - ‘Category’s Name cannot be blank’. - ‘Category’s Name must be unique’. |
|
|

****

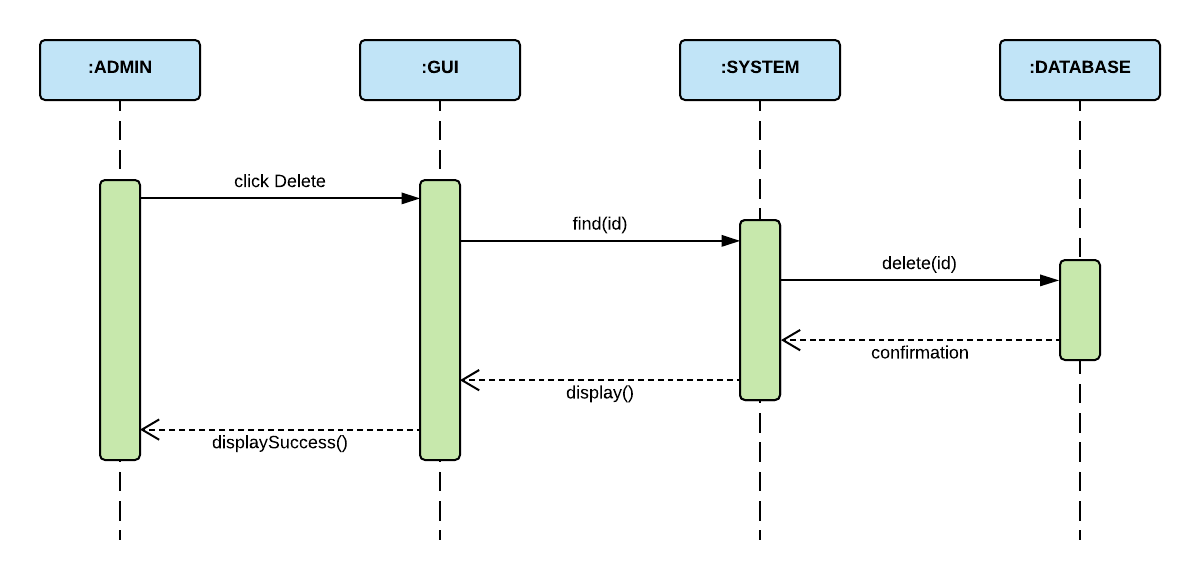
1. **Update category**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Update category | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update a category | |
| **Requirements** | Actor provides category’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: category is updated in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Category Tab on menu.  3. Actor clicks “Update” on Category list.  5. Actor inputs information update Category. | 2. System redirects to 'Category List' page and display a list of existing categories.  4. System displays a form with the following fields. - Category’s Name - Parent category (optional)  6. System validates inputs.  7. System updates category in database  8. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Category’s Name’ field blank.  2. Category’s Name existed in database. | System displays messages: - ‘Category’s Name cannot be blank’. - ‘Category’s Name must be unique’. |
|
|

****

1. **Delete category**

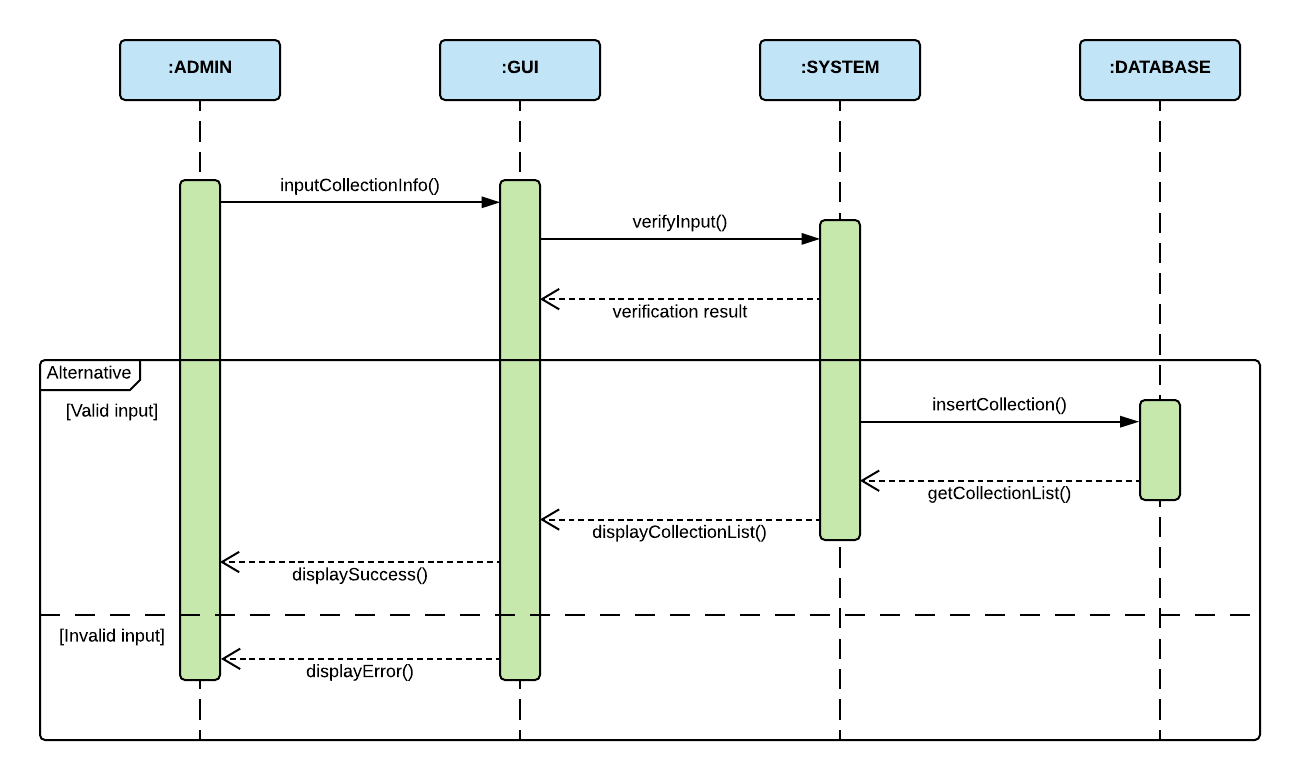
|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Delete category | |
| **Actors** | Super Admin | |
| **Description** | When a super admin wants to delete a category from the database | |
| **Requirements** | Actor provides category’s information | |
| **Pre-conditions** | Actor is logged in as super admin | |
| **Post-conditions** | Success: category is deleted from the database | |
| Fail: category is not deleted from the database | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Category Tab on menu.  3. Actor clicks on ‘Delete’ button and confirm. | 2. System redirects to 'Category List' page and display a list of existing categories.  4. System delete category from database.  5. System displays confirmation message or error message. |

****

### Product Collection

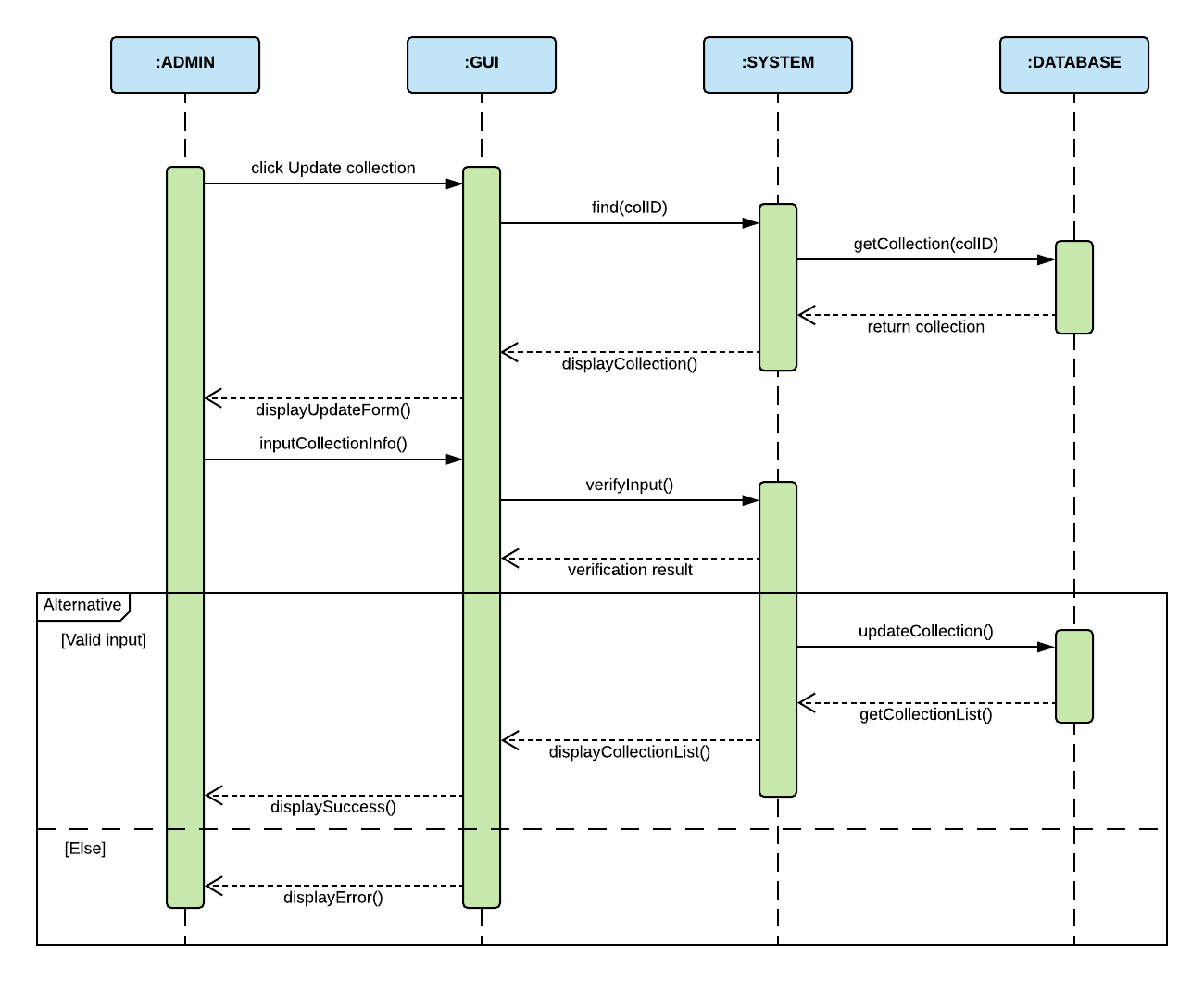
1. **Create new collection**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Create new collection | |
| **Actors** | Administrator | |
| **Description** | When admin wants to create a new collection | |
| **Requirements** | Actor provides collection’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: a new collection is added in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Collection Tab on menu.  3. Actor clicks ‘Add New Collection’ button.  5. Actor inputs information to create new collection. | 2. System redirects to 'Collection List' page and displays a list of existing collections.  4. System displays a form with the following fields: - Collection’s Name  - Collection’s Banner  - Collection’s Image  - Collection’s Description  6. System validates inputs.  7. System adds collection into the database.  7. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Collection’s Name’ field blank.  2. Actor left ‘Collection’s banner’ field blank.  3. Actor left ‘Collection’s image’ field blank.  4. Actor left ‘Description’ field blank.  5. Collection’s Name existed in database | System displays messages: - ‘Collection’s Name cannot be blank.’  - ‘Must choose a banner’  - ‘Must choose an image’  - ‘Description cannot be blank’  - ‘Collection’s Name’ must be unique |
|
|

****

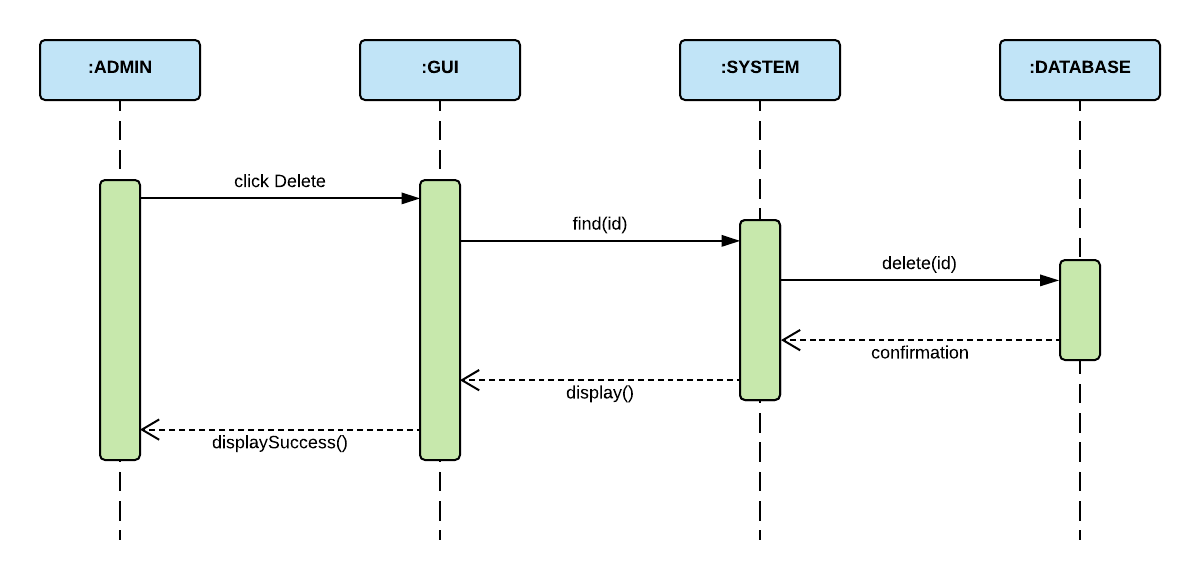
1. **Update collection**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Update product’s collection | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update product’s collection | |
| **Requirements** | Actor provides collection’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: collection is updated in database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Collection Tab on menu.  3. Actor clicks ‘Update’ button on a collection.  5. Actor inputs information to update collection. | 2. System redirects to 'Collection List' page and display a list of existing collections.  4. System displays a form with the following fields. - Collection’s Name  - Collection’s Banner  - Collection’s Image  - Collection’s Description  6. System validates inputs.  7. System updates collection in database.  8. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Collection’s Name’ field blank.  2. Actor left ‘Collection’s banner’ field blank.  3. Actor left ‘Collection’s image’ field blank.  4. Actor left ‘Description’ field blank. | System displays messages: - ‘Collection’s Name cannot be blank.’  - ‘Must choose a banner’  - ‘Must choose an image’  - ‘Description cannot be blank’ |
|
|

****

1. **Delete collection**

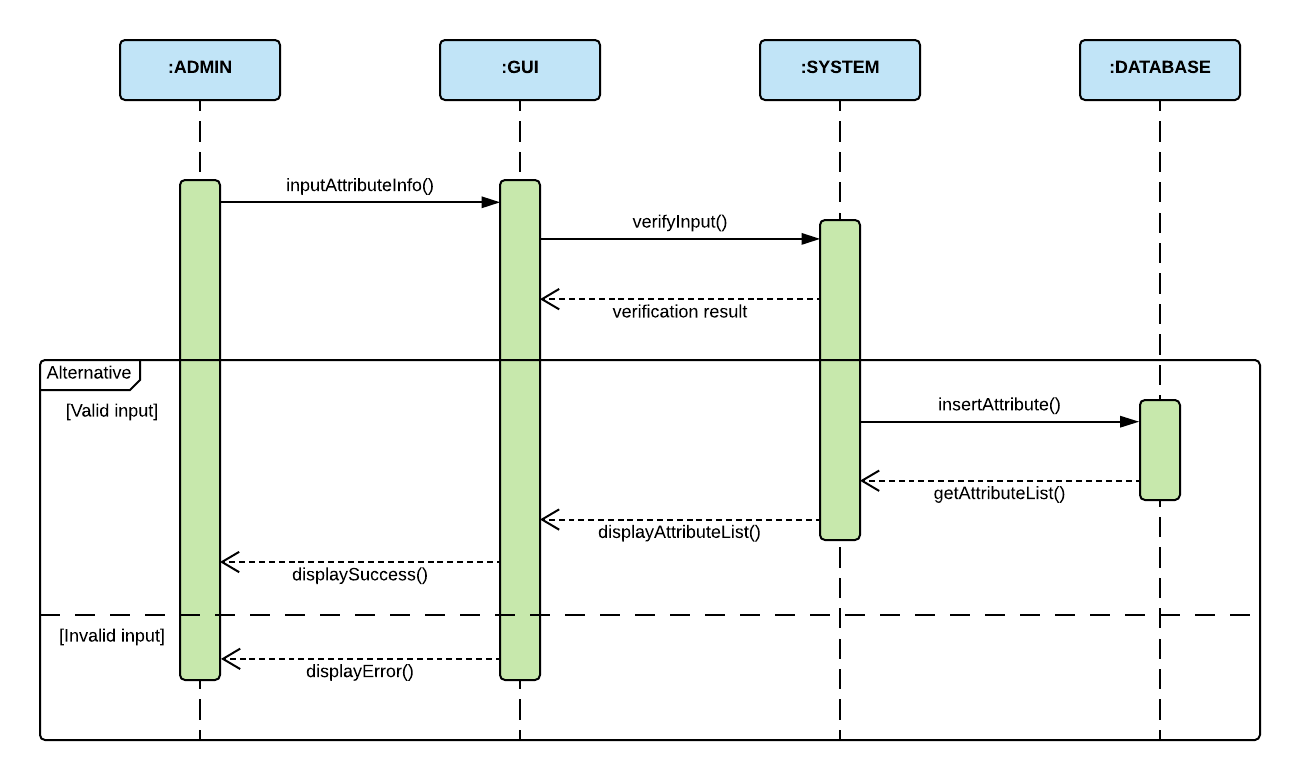
|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Delete Collection | |
| **Actors** | Super admin | |
| **Description** | When super admin wants to delete a collection from the database | |
| **Requirements** | Actor provides collection’s information | |
| **Pre-conditions** | Actor is logged in as super admin | |
| **Post-conditions** | Success: collection is deleted from the database | |
| Fail: collection is not deleted from the database | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Collection Tab on menu.  3. Actor clicks on ‘Delete’ button and confirm. | 2. System redirects to 'Collection List' page and displays a list of existing collections.  4. System deletes collection from database.  5. System displays confirmation message or error message. |



### Attribute

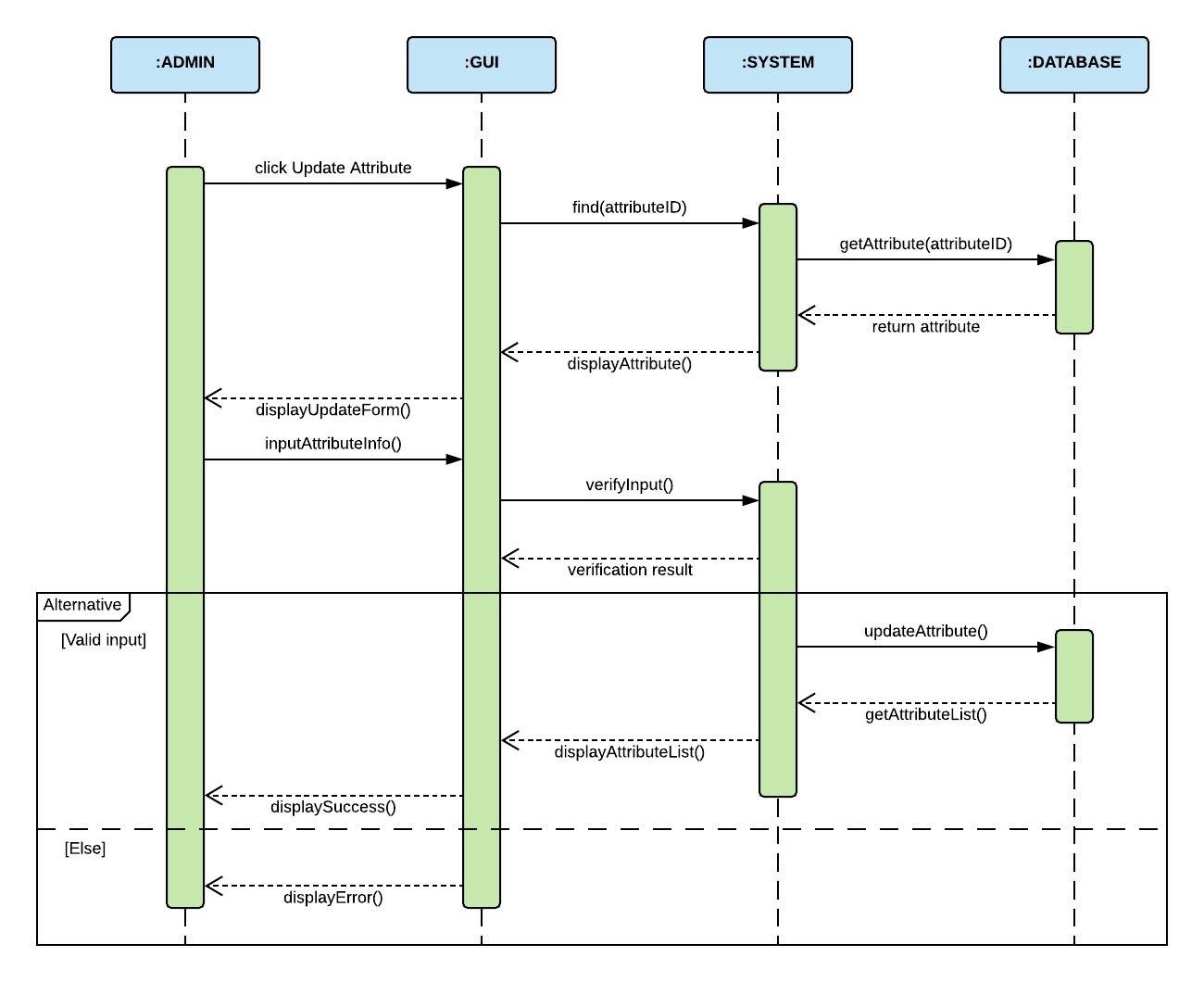
1. **Create New Attribute/Attribute Group**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Create new attribute/attribute group | |
| **Actors** | Administrator | |
| **Description** | When admin wants to add new attribute in the database | |
| **Requirements** | Actor provides attribute’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: attribute (and group) is added in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses ‘Attribute Tab’ in menu.  3. Actor clicks ‘Add New Attribute’ button.  5. Actor inputs attribute’s information to create new attribute. | 2. System redirects to 'Attribute List' page.  4. System displays list of all of Attributes with Attribute form containing the following controls:  - Attribute’s Name  - New Attribute Group’s Name  - Is New checkbox  - Attribute Group list  6. System validates inputs.  7. Systems adds attribute (and attribute group) into the database.  8. System display success or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left Attribute’s Name text field blank.  2. ‘Attribute’s Name’ existed in the database under the same group.  3. New Attribute Group’s Name existed in the database. | System displays messages: - ‘Attribute’s Name cannot be blank’.  - ‘Attribute’s Name already existed’.  - ‘New Group Name must be unique’. |
|
|



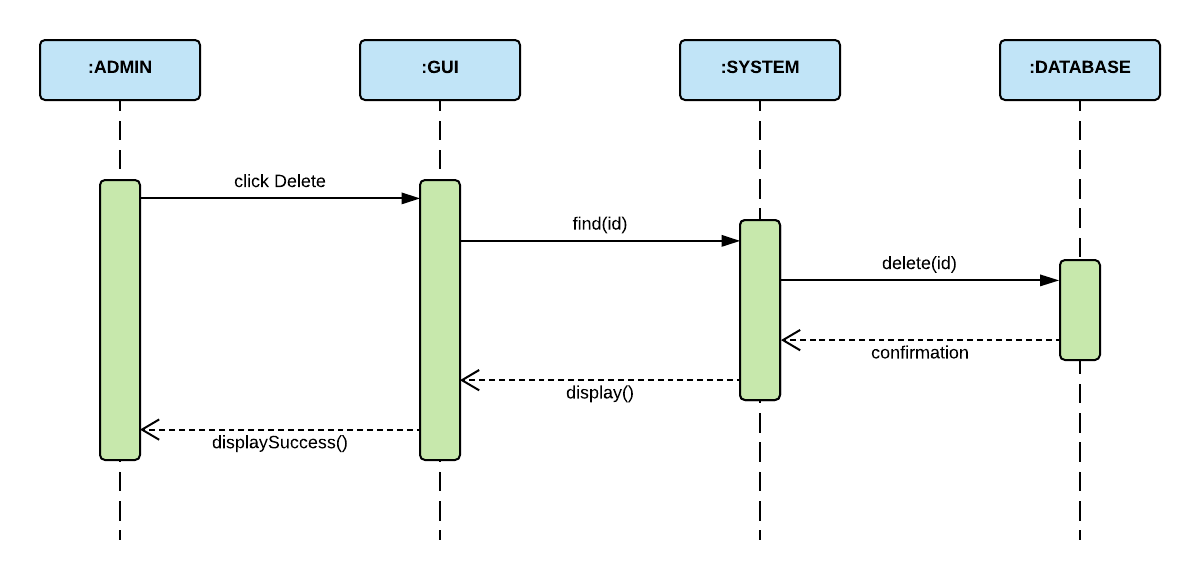
1. **Update Attribute**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Update attribute | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update an attribute in the database | |
| **Requirements** | Actor provides attribute’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: attribute is updated in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses ‘Attribute Tab’ in menu.  3. Actor clicks ‘Update’ button in an attribute.  5. Actor inputs attribute’s information to create new attribute. | 2. System redirects to 'Attribute List' page.  4. System displays list of all of Attributes with Attribute form containing the following controls:  - Attribute’s Name  - New Attribute Group’s Name  - Is New checkbox  - Attribute Group list  6. System validates inputs.  7. Systems updates into the database.  8. System display success or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left Attribute’s Name text field blank.  2. ‘Attribute’s Name’ existed in the database under the same group.  3. New Attribute Group’s Name existed in the database. | System displays messages: - ‘Attribute’s Name cannot be blank’.  - ‘Attribute’s Name already existed’.  - ‘New Group Name must be unique’. |
|
|



1. **Delete Attribute**

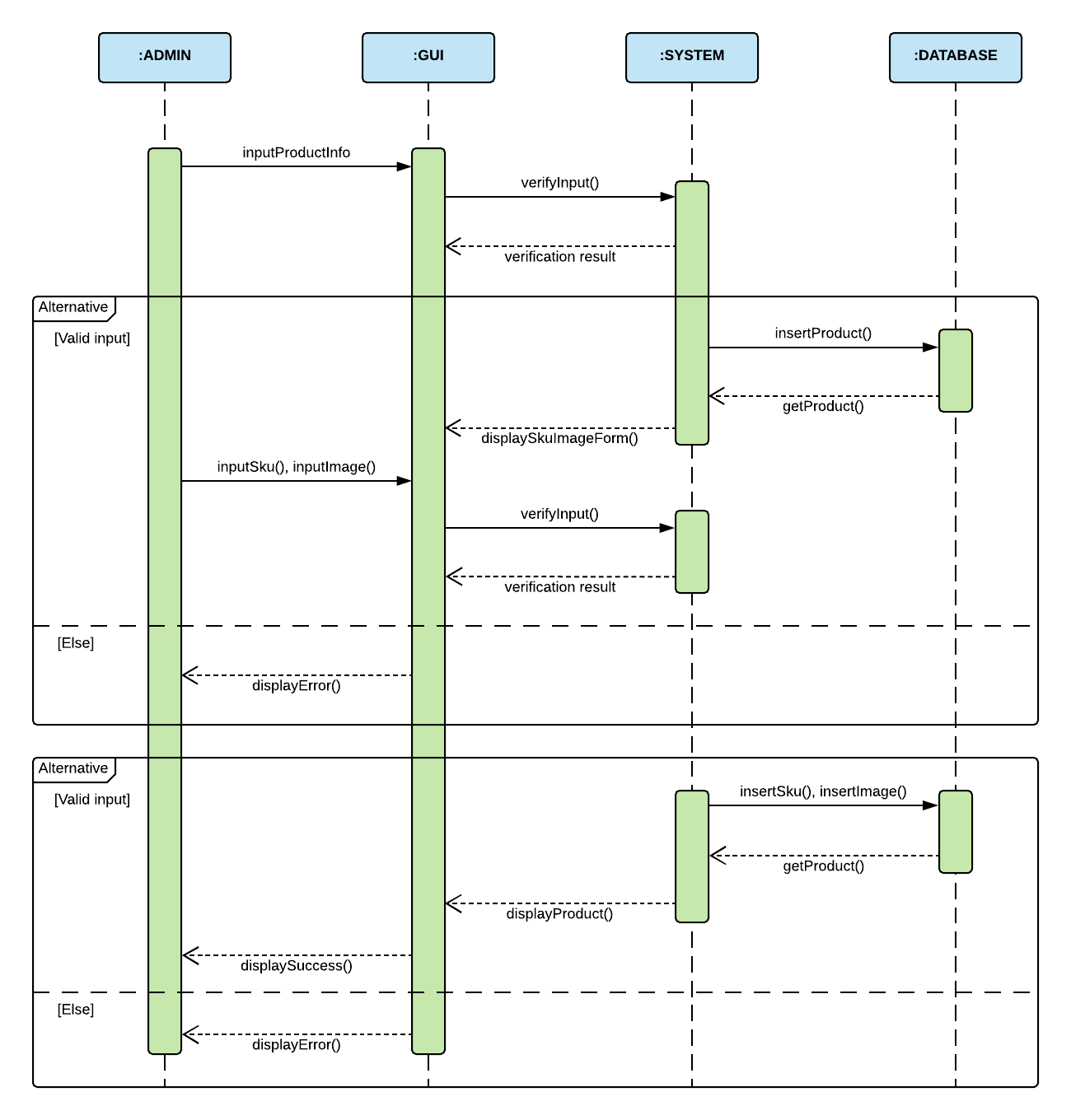
|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Delete Attribute | |
| **Actors** | Super admin | |
| **Description** | When super admin wants to delete an attribute from the database | |
| **Requirements** | Actor provides attribute’s information | |
| **Pre-conditions** | Actor is logged in as super admin | |
| **Post-conditions** | Success: attribute is deleted from the database | |
| Fail: attribute is not deleted from the database | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses ‘Attribute Tab’ on menu.  3. Actor clicks on ‘Delete’ button and confirm. | 2. System redirects to 'Attribute List' page and displays a list of existing attributes.  4. System deletes attribute from database.  5. System displays confirmation message or error message. |



### Product

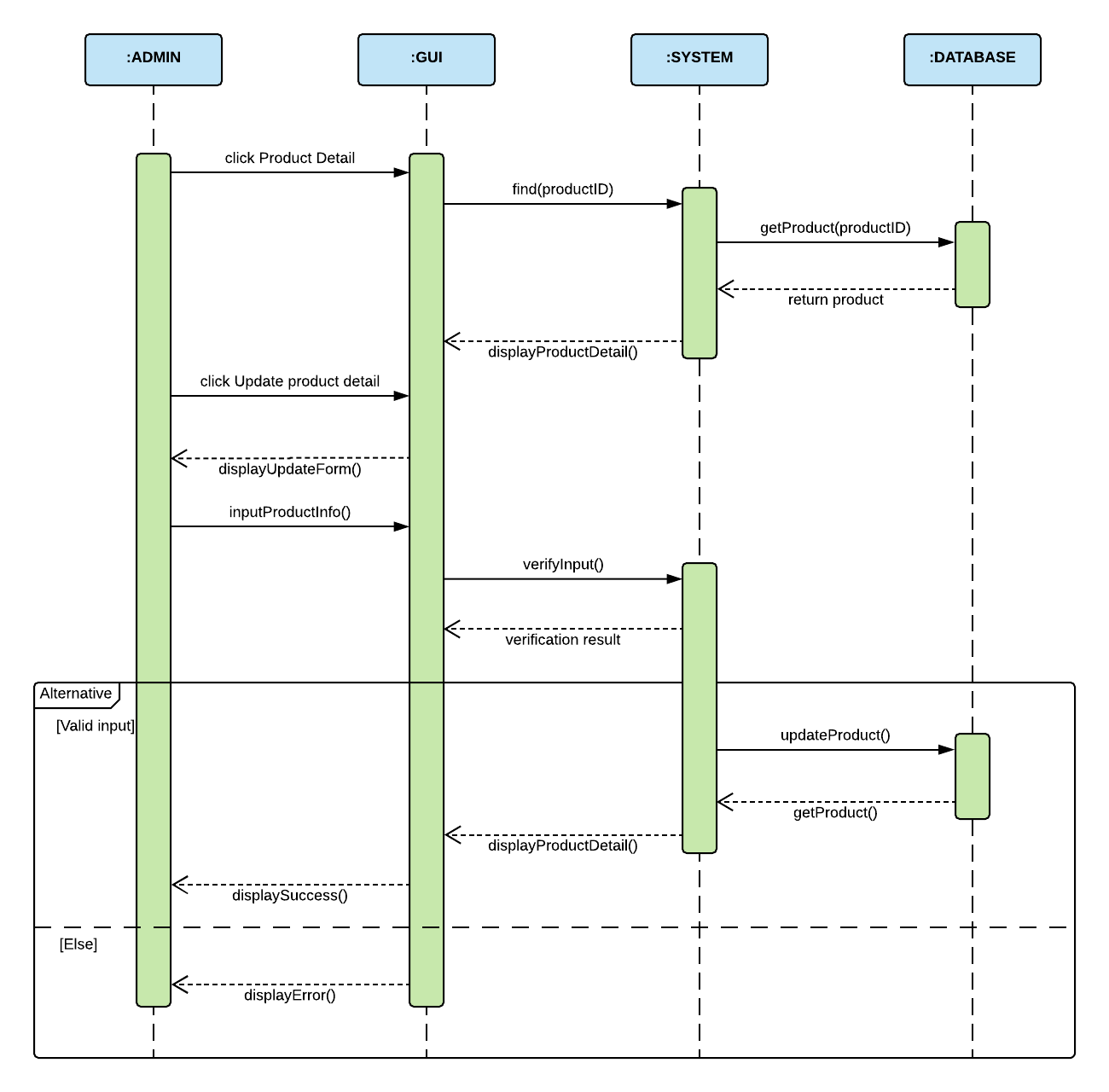
1. **Create new Product (with SKUs and Images)**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Create New Product (with SKUs and Images) | |
| **Actors** | Administrator | |
| **Description** | When admin wants to create a new product | |
| **Requirements** | Actor provides product’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: new product with SKUs and images is added into the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Product Tab on menu.  3. Actor clicks ‘Add New Product’ button.  5. Actor inputs information to create product. | 2. System redirects to ‘Product’ page and display a list of existing products.  4. System displays a form with the following fields. - Product’s Name  - Product’s Description  - Collection  - Category  - Discount Value  - Discount Valid dates  - List of attributes  6. System validates inputs.  7. System adds product into database.  8. System displays a form for adding SKUs and Images for this product.  9. System displays confirmation message or error message. |
|
|
|
|
|
|
|
|
|
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Product Name’ blank.  2. Actor left ‘Product Description’ blank.  3. Actor does not choose a Category.  4. Actor choose invalid Discount Dates  5. Actor does not add any Sku  6. Actor does not add any Image  7. Sku ID existed in database | System displays messages: - ‘Product Name cannot be blank’  - ‘Description cannot be blank’  - ‘Must choose a Category’  - ‘Must choose valid Discount Date’  - ‘Product must have Sku and Image’  - ‘Sku ID must be unique’ |
|
|



1. **Update product**

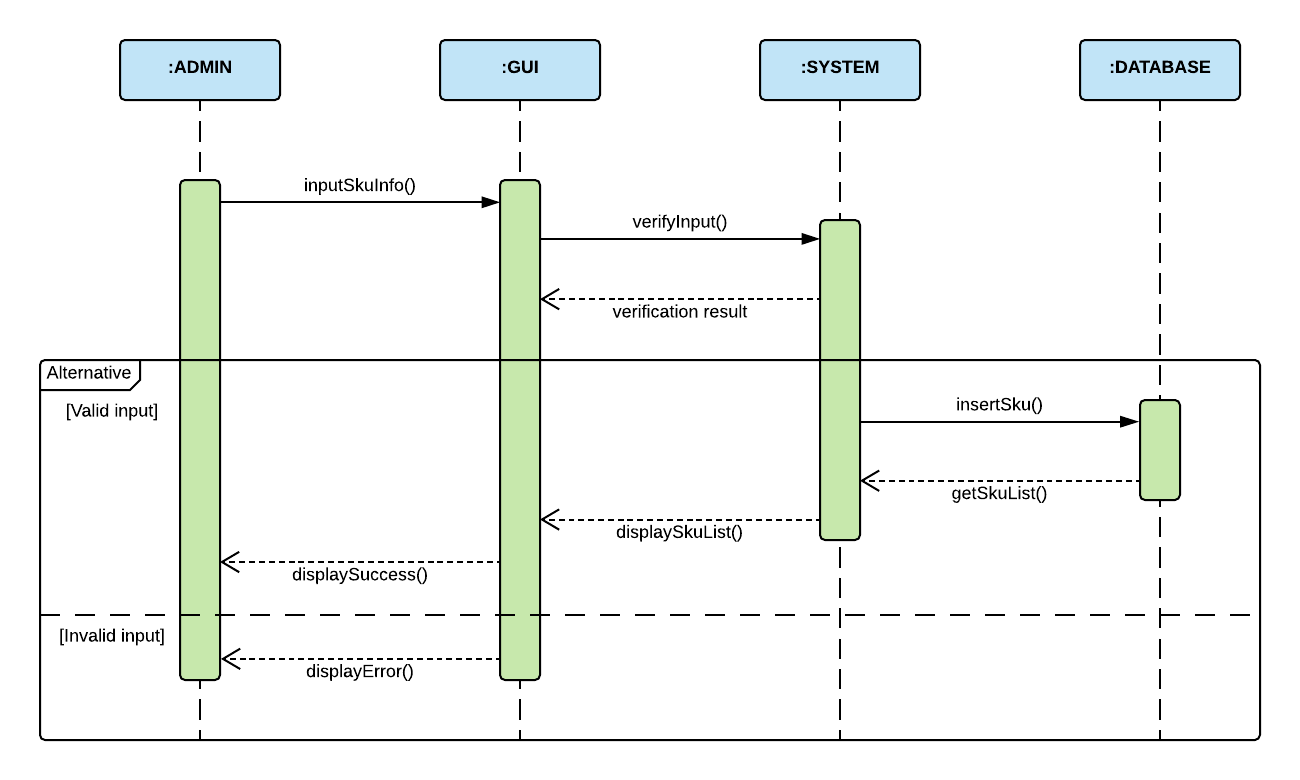
|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Update product’s details | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update product’s details | |
| **Requirements** | Actor provides product’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: product is updated in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Product Tab on menu.  3. Actor clicks on ‘View Details’ on a product.  5. Actor chooses the criteria they want to update and input information. | 2. System redirects to ‘Product Details’ page and displays a list of existing products.  4. System displays product’s detail where actor can update the following information: - Product’s Name  - Product’s Description  - Collection  - Category  - Discount Value  - Discount Valid dates  - List of attributes  - Product Status  - Add/Delete Sku  - Add/Delete Images  6. System validates inputs.  7. System updates product in the database.  8. System displays confirmation message or error message.  9. System redirects to product’s detail page. |
|
|
|
|
|
|
|
|
|
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Product Name’ blank.  2. Actor left ‘Product Description’ blank.  3. Actor does not choose a Category.  4. Actor choose invalid Discount Dates  5. Sku ID existed in database | System displays messages: - ‘Product Name cannot be blank’  - ‘Description cannot be blank’  - ‘Must choose a Category’  - ‘Must choose valid Discount Date’  - ‘Product must have Sku and Image’  - ‘Sku ID must be unique’ |
|
|



### SKU

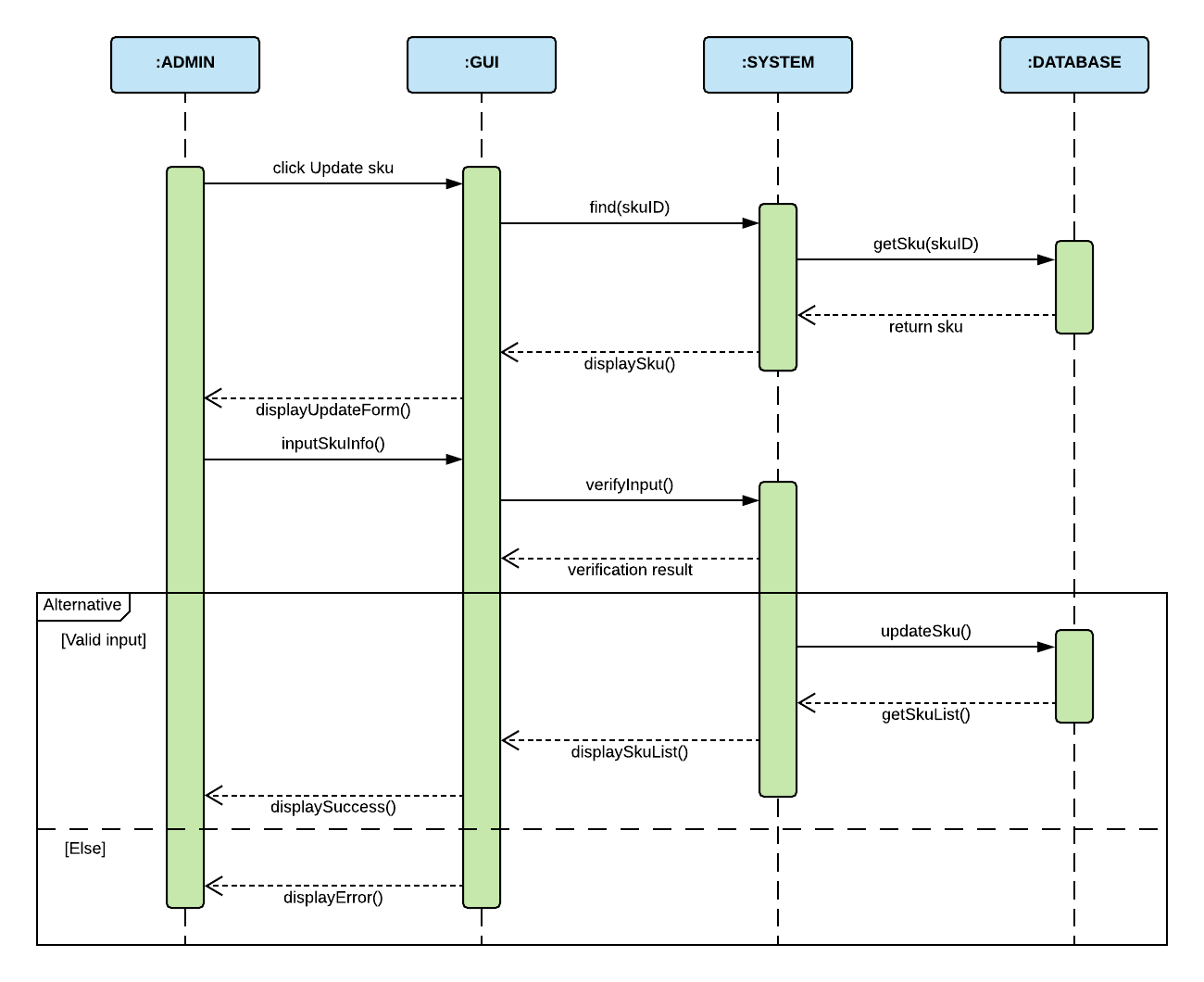
1. **Create new SKU for product**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Create new sku | |
| **Actors** | Administrator | |
| **Description** | When admin wants to create a new sku for a product | |
| **Requirements** | Actor provides sku’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: a new sku is added in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Product Tab on menu.  3. Actor clicks ‘Product Detail’ button.  5. Actor clicks ‘Add New Sku’ in Sku List.  7. Actor inputs information to create new sku. | 2. System redirects to 'Product List' page and displays a list of existing products.  4. System displays Product’s detail page.  6. System displays a form with the following controls:  - Sku ID (partly auto-generated)  - Sku Size,  - Base Price  - Unit Price  - Quantity in Stock  8. System validates inputs.  9. System adds sku in the database.  10. System display confirmation or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Sku ID’ blank.  2. Sku ID existed in the database  3. Sku with the same size existed in the same product  4. Base Price is below 10  5. Unit Price is lower than Base Price or higher than 100.000.000  6. Quantity in Stock is less than 1 or greater than 10000. | System displays messages: - ‘Sku ID cannot be blank’  - ‘Sku with the same size already existed’  - ‘Base Price must be greater than $10’  - ‘Unit Price cannot be less than Base Price’  - ‘Unit Price cannot be higher than $100.000.000’  - Quantity in Stock cannot be less than 1’ |
|
|



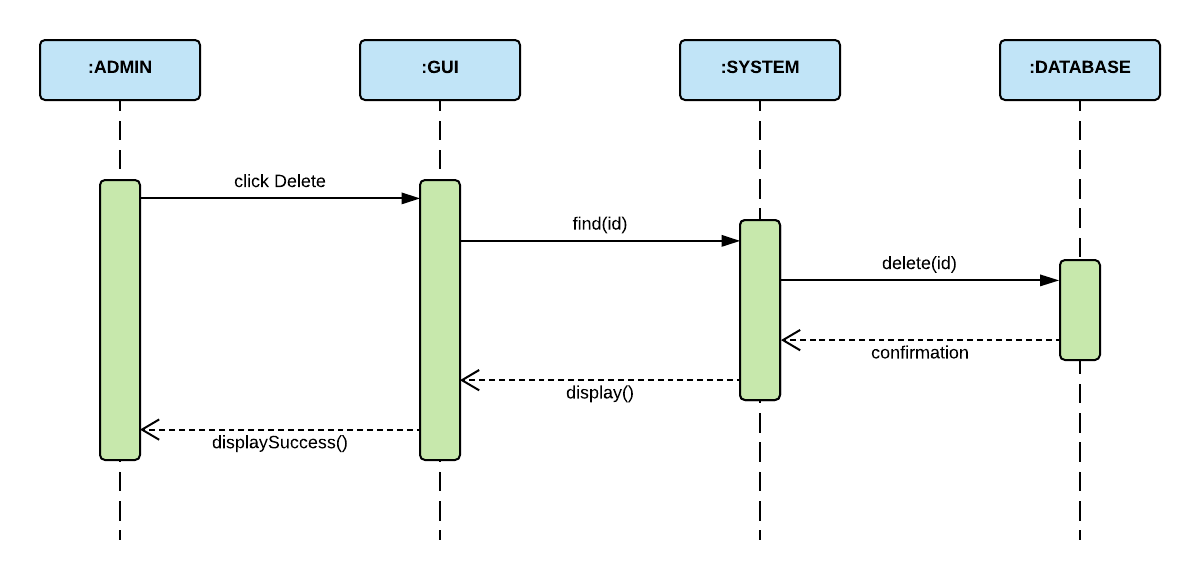
1. **Update SKU**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Update product’s sku | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update sku’s information | |
| **Requirements** | Actor provides sku’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: sku is updated in database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Product Tab on menu.  3. Actor clicks ‘Product Detail’ button.  5. Actor clicks ‘Update’ in Sku List.  7. Actor inputs information to create update sku. | 2. System redirects to 'Product List' page and displays a list of existing products.  4. System displays Product’s detail page.  6. System displays a form with the following controls:  - Sku Size,  - Base Price  - Unit Price  - Quantity in Stock  8. System validates inputs.  9. System updates sku in the database.  10. System display confirmation or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Sku with the same size existed in the same product  2. Base Price is below 10  3. Unit Price is lower than Base Price or higher than 100.000.000  4. Quantity in Stock is less than 1 or greater than 10000. | System displays messages: - ‘Sku with the same size already existed’  - ‘Base Price must be greater than $10’  - ‘Unit Price cannot be less than Base Price’  - ‘Unit Price cannot be higher than $100.000.000’  - Quantity in Stock cannot be less than 1’ |
|
|



1. **Delete SKU**

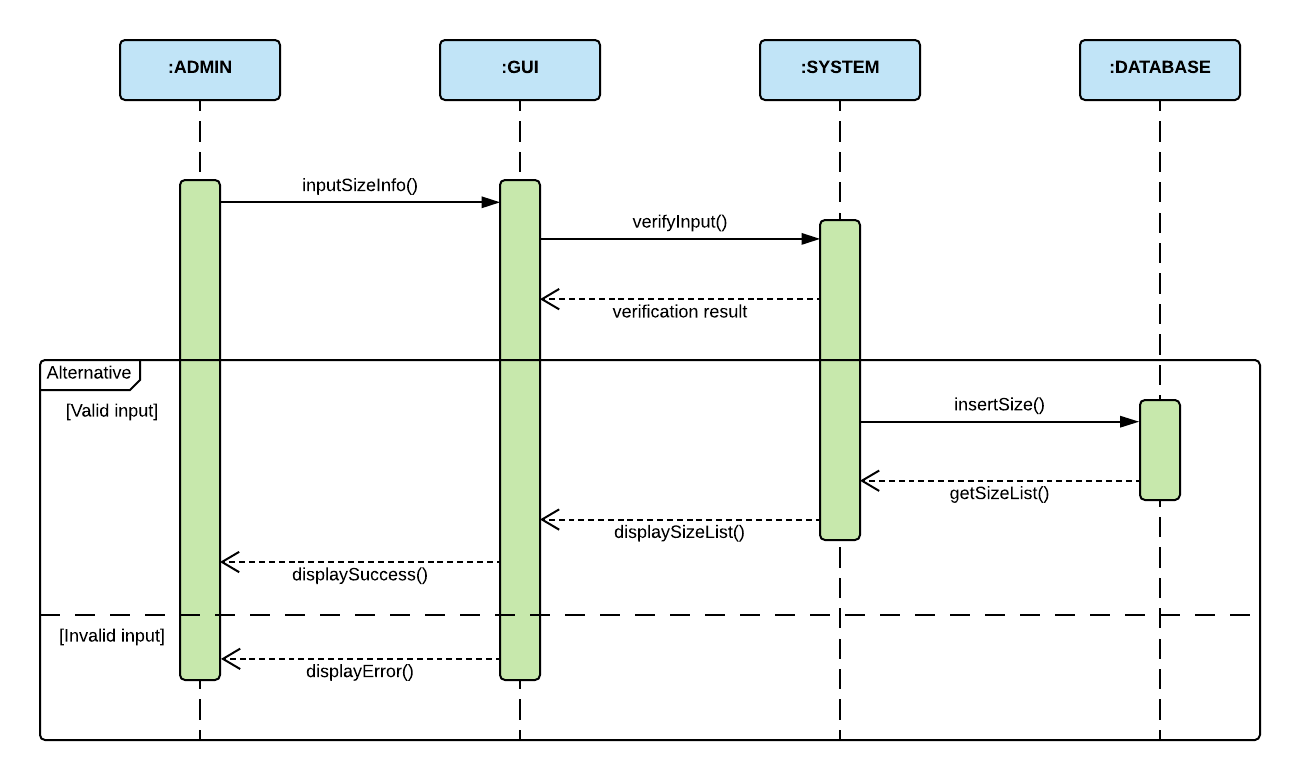
|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Delete Sku | |
| **Actors** | Super admin | |
| **Description** | When super admin wants to delete a Sku from the database | |
| **Requirements** | Actor provides sku’s information | |
| **Pre-conditions** | Actor is logged in as super admin | |
| **Post-conditions** | Success: sku is deleted from the database | |
| Fail: sku is not deleted from the database | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Product Tab on menu.  3. Actor clicks ‘Product Detail’ button.  5. Actor clicks ‘Delete’ in Sku List and clicks confirm. | 2. System redirects to 'Product List' page and displays a list of existing products.  4. System displays Product’s detail page.  8. System validates inputs.  9. System deletes sku in the database.  10. System displays confirmation or error. |



### Size

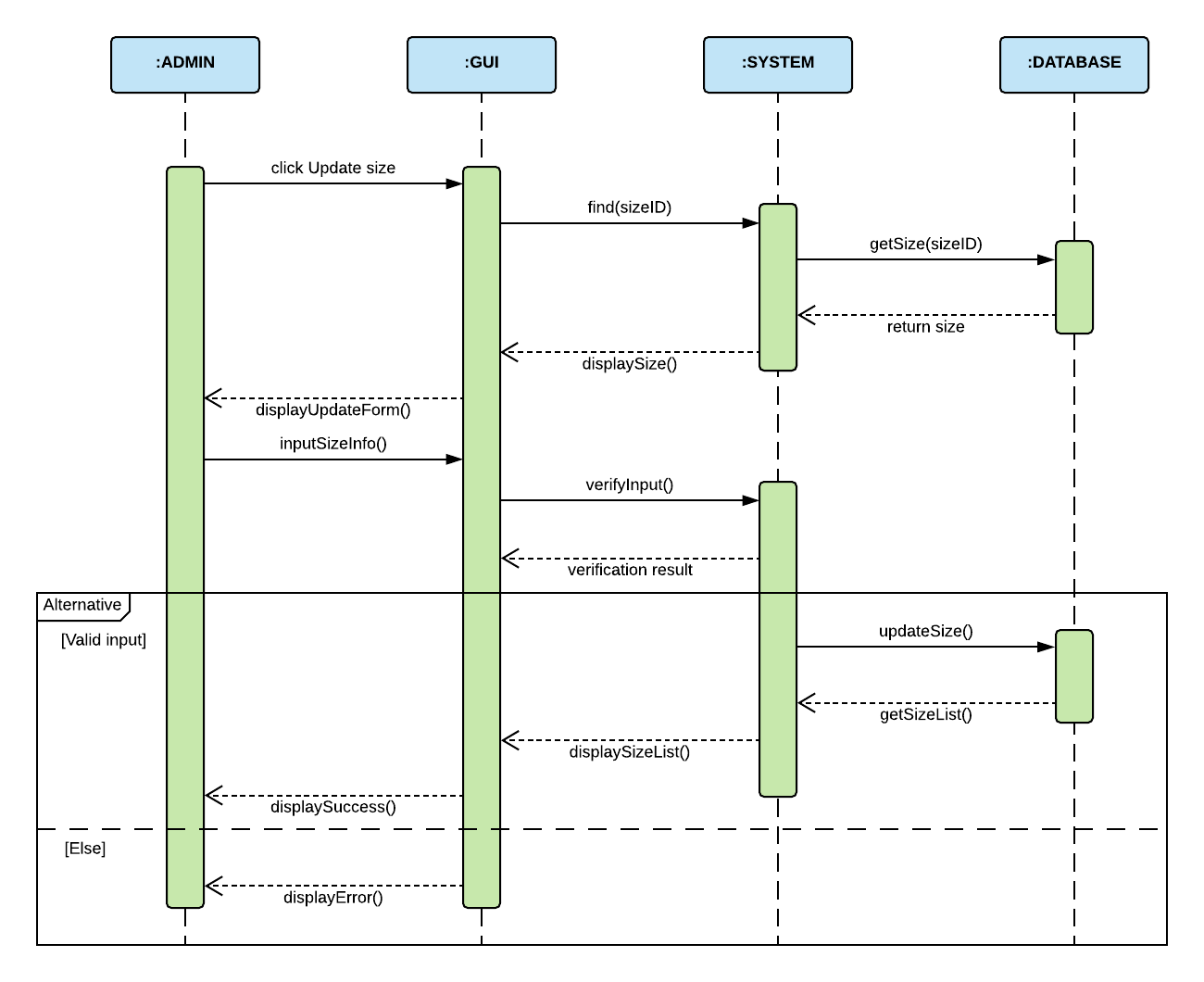
1. **Create new Size**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Create new size | |
| **Actors** | Administrator | |
| **Description** | When admin wants to create a new size for product | |
| **Requirements** | Actor provides collection’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: a new size is added in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Size Tab on menu.  3. Actor clicks ‘Add New Size’ button.  5. Actor inputs information to create new size. | 2. System redirects to 'Size List' page and displays a list of existing sizes.  4. System displays a form with the following fields: - Size’s Value  - Category  6. System validates inputs.  7. System adds size into the database.  7. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Size Value’ blank.  5. Size value matches an existing size in the same Category. | System displays messages: - ‘Size value must not be blank’  - ‘Size already existed in this Category’ |
|
|



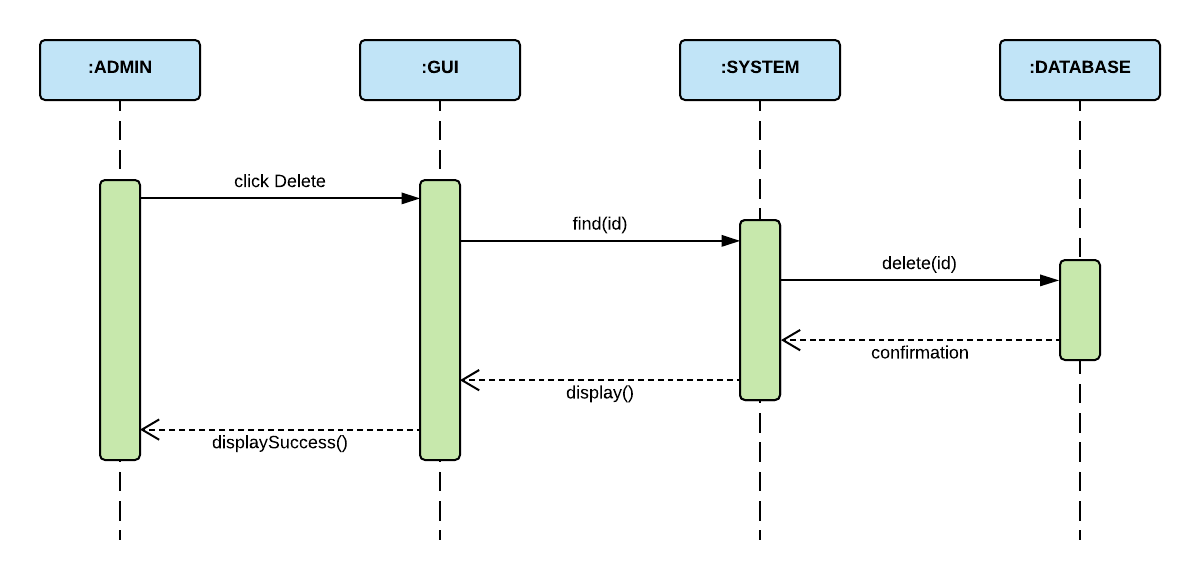
1. **Update Size**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Update size | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update size | |
| **Requirements** | Actor provides collection’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: size is updated in database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Size Tab on menu.  3. Actor clicks ‘Update’ button.  5. Actor inputs information to update new size. | 2. System redirects to 'Size List' page and displays a list of existing sizes.  4. System displays a form with the following fields: - Size’s Value  - Category  6. System validates inputs.  7. System adds size into the database.  7. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Size Value’ blank.  5. Size value matches an existing size in the same Category. | System displays messages: - ‘Size value must not be blank’  - ‘Size already existed in this Category’ |
|
|



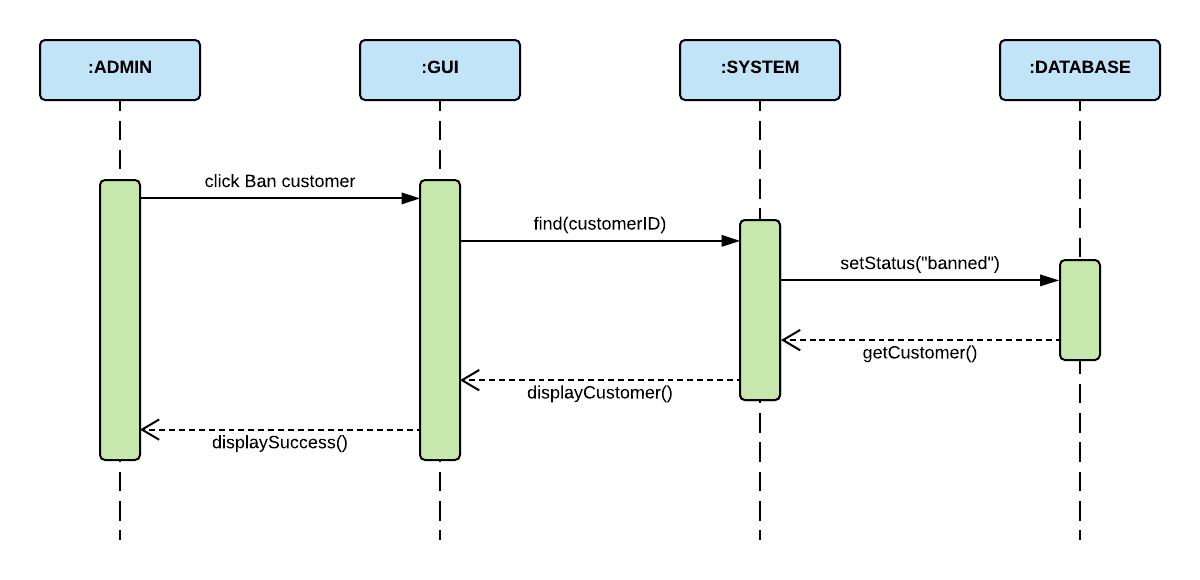
1. **Delete Size**

|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Delete Size | |
| **Actors** | Super admin | |
| **Description** | When super admin wants to delete a collection from the database | |
| **Requirements** | Actor provides size’s information | |
| **Pre-conditions** | Actor is logged in as super admin | |
| **Post-conditions** | Success: size is deleted from the database | |
| Fail: size is not deleted from the database | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Size Tab on menu.  3. Actor clicks on ‘Delete’ button and confirm. | 2. System redirects to 'Size List' page and displays a list of existing sizes.  4. System deletes size from database.  5. System displays confirmation message or error message. |



### Customer (Ban Customer)

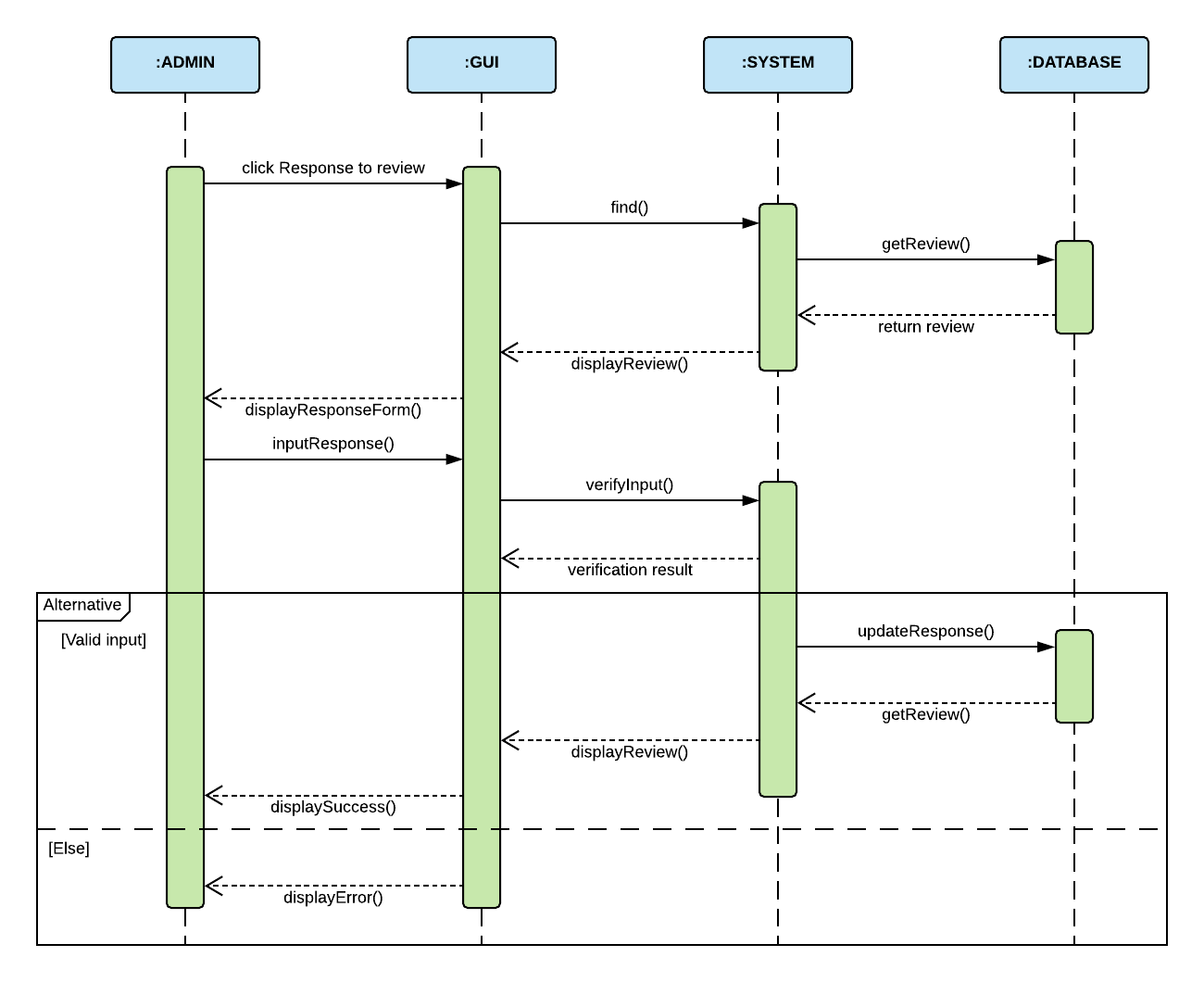
|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use case name** | Ban a customer | |
| **Actors** | Administrator | |
| **Description** | When admin wants to ban a customer | |
| **Requirements** | Actor provides customer’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: set customer’s status into “Banned” and updated in database | |
| Fail: No changes are made | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Customer row on Customer list and click “Ban” button. | 2. System updates customer’s status in database.  3. System displays confirmation message or error message. |



### Review

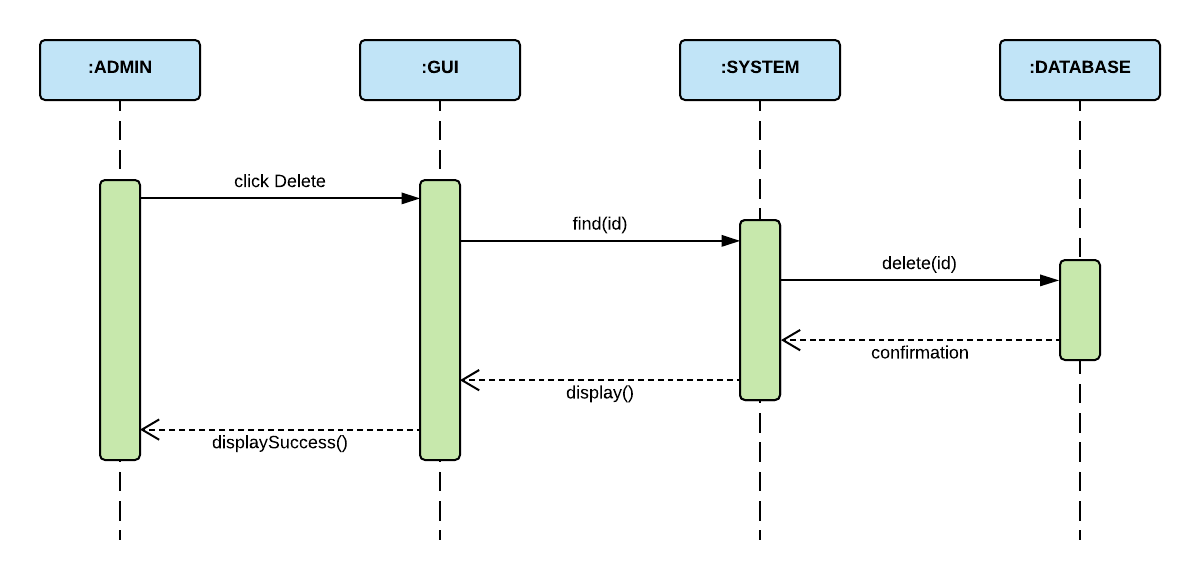
1. **Response to Review**

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use case name** | Response review of customer | |
| **Actors** | Administrator | |
| **Description** | When admin wants to response review of customer | |
| **Requirements** | Actor provides customer’s review information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: response content is updated in database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses review row on review list and click “Response” button.  4. Actor input content into “admin response” text field and click “Save” button. | 2. System redirects to 'Response review' page and displays review of customer.  3. System displays a form with the following fields. - Customer ID (read only)  - Product ID (read only)  - Title (read only)  - Content (read only)  - Rating (read only)  - Admin response  5. System validates inputs.  6. System updates review in database.  7. System redirects to 'Review list' page. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Admin response’ field blank. | System displays messages: - ‘Response cannot be blank.’ |
|



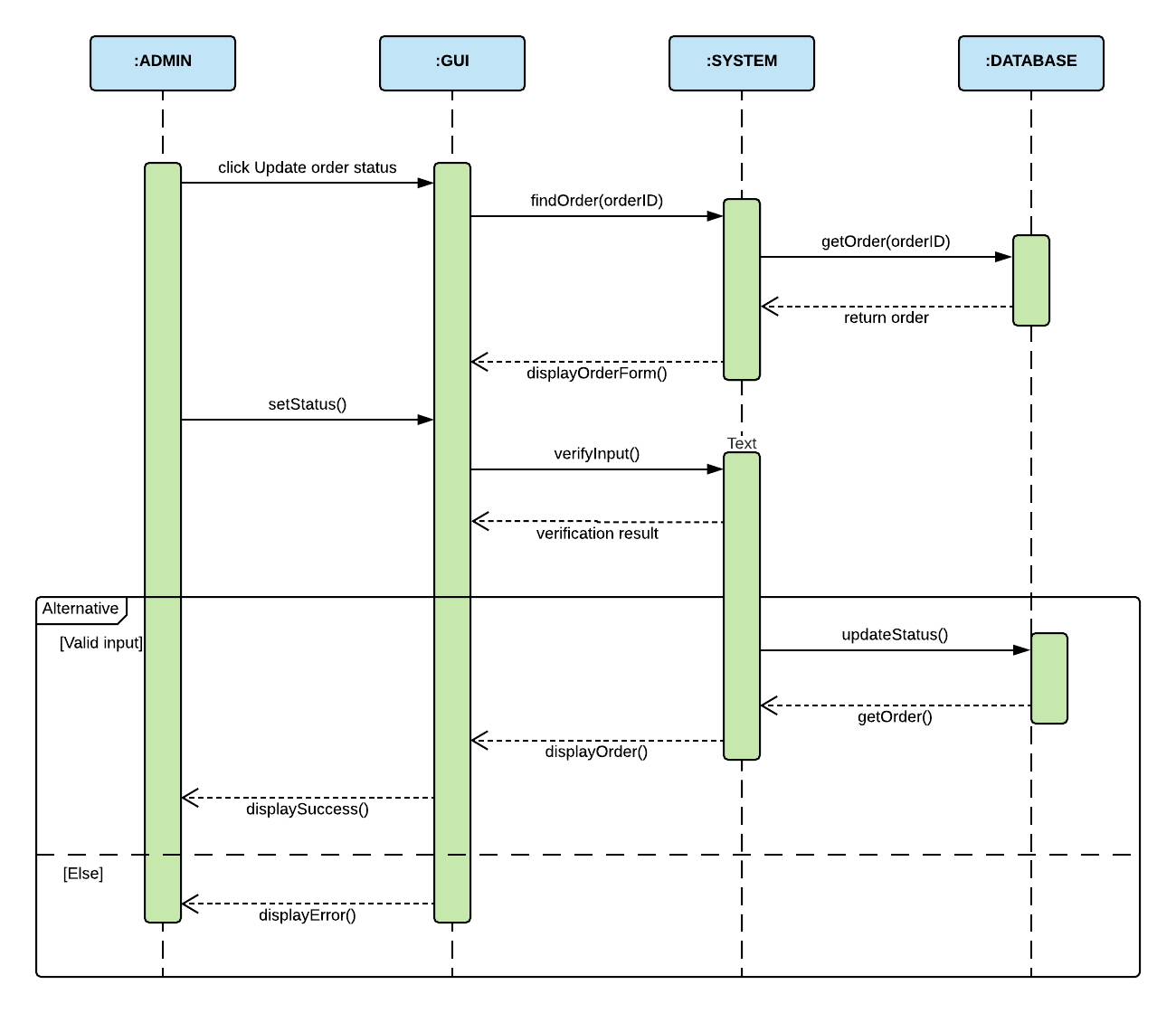
1. **Delete Review**

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use case name** | Delete Review | |
| **Actors** | Super admin | |
| **Description** | When super admin wants to delete a review of customer from the database | |
| **Requirements** | Actor provides customer’s review information | |
| **Pre-conditions** | Actor is logged in as super admin | |
| **Post-conditions** | Success: review is deleted from the database | |
| Fail: review is not deleted from the database | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses review row on review list and click “Delete” button. | 2. System deletes review from database.  3. System displays confirmation message. |



### Order (Update Order’s Status)

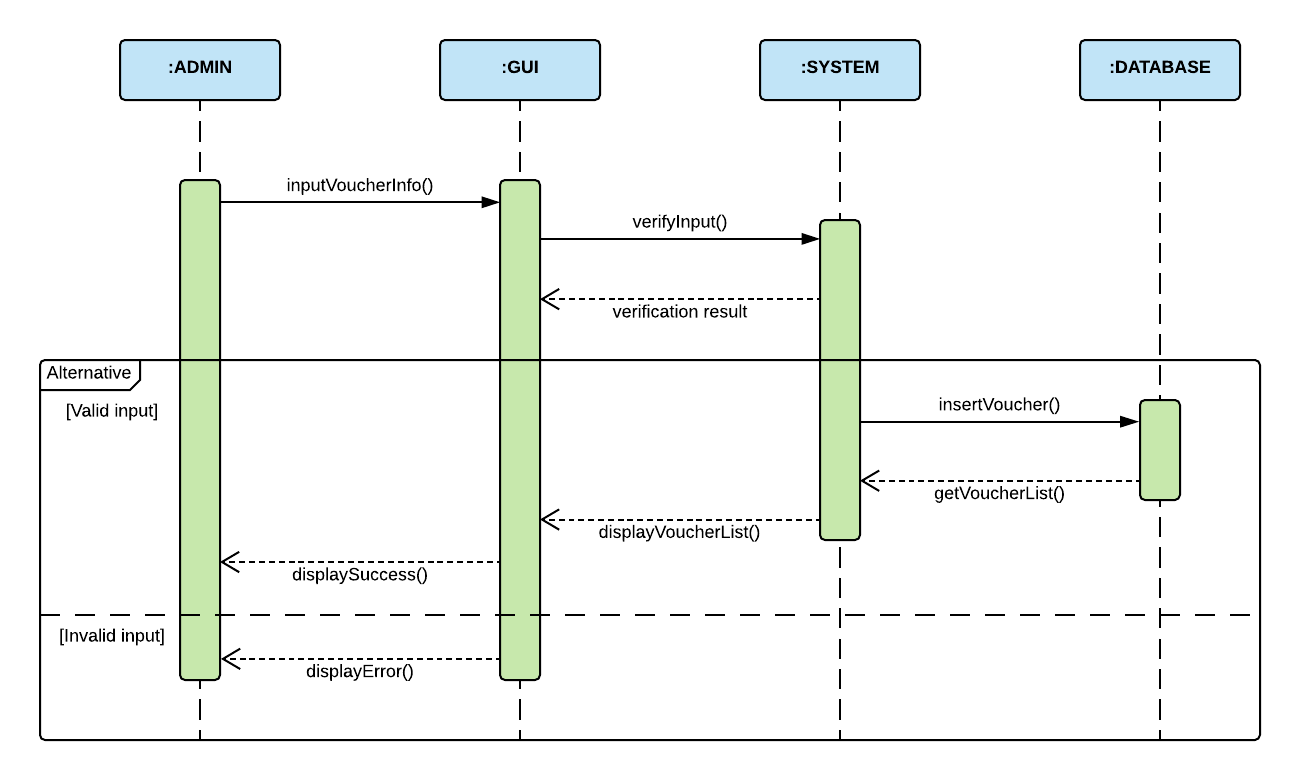
|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | Update Order’s Status | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update order’s status | |
| **Requirements** | Actor provides order’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: order’s status is updated in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Order Tab on menu.  3. Actor clicks ‘Update Status’ button on an order.  5. Actor chooses the status they want to update. | 2. System redirects to 'Order List' page and display a list of existing orders.  4. System displays a list of statuses: Pending, Confirmed, Canceled, Completed, On Delivery.  6. System updates collection in database.  7. System displays confirmation message. |



### Voucher

1. **Create new Voucher**

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use case name** | Create new voucher | |
| **Actors** | Administrator | |
| **Description** | When admin wants to create a new voucher | |
| **Requirements** | Actor provides voucher’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: a new voucher is added in the database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Add Voucher Tab on menu.  4. Actor inputs information to create new voucher and click “Save” button. | 2. System redirects to 'Add voucher' page.  3. System displays a form with the following fields: - Voucher’s ID  - Voucher’s discount valid date (date valid from and date valid to)  - Voucher’s Description  - Voucher’s Discount value  - Voucher’s Quantity  5. System validates inputs.  6. System adds voucher into the database.  7. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Voucher ID’ field blank.  2. Actor left ‘Description’ field blank.  3. Actor left ‘Discount value’ field blank.  4. Actor left ‘Quantity’ field blank.  5. Voucher’s ID existed in database  6. Valid date in the past | System displays messages: - ‘Voucher’s ID cannot be blank.’  - ‘Voucher’s Description cannot be blank’  - ‘Voucher’s Discount value cannot be blank’  - ‘Voucher’s Quantity cannot be blank’  - ‘Voucher’s ID’ must be unique  - ‘Valid date cannot be in the past’ |
|
|



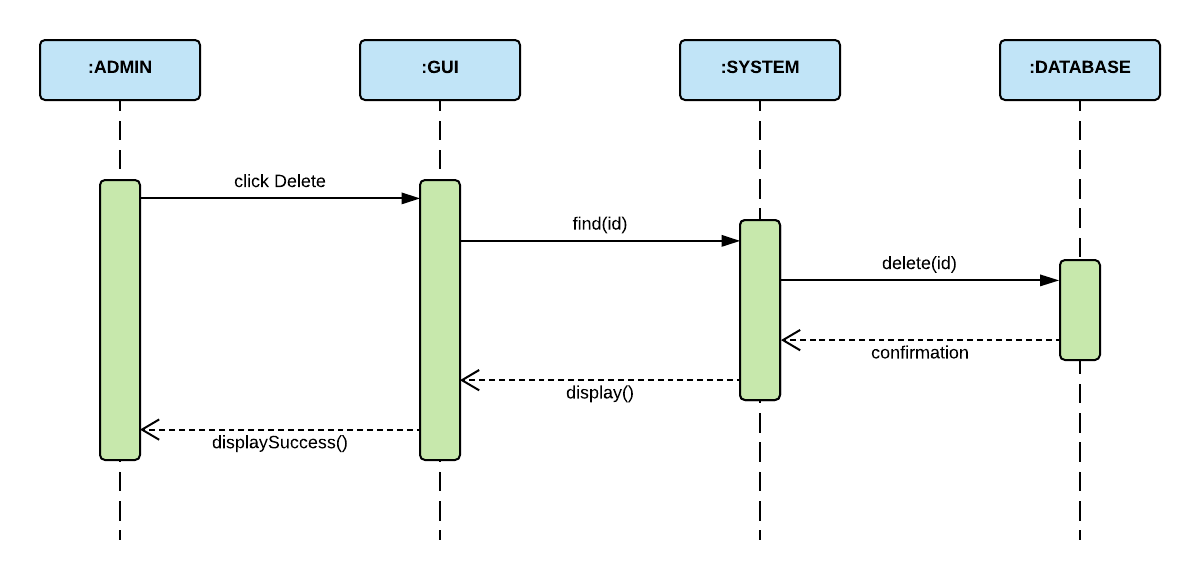
1. **Update Voucher**

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use case name** | Update product’s voucher | |
| **Actors** | Administrator | |
| **Description** | When admin wants to update product’s voucher | |
| **Requirements** | Actor provides collection’s information | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: voucher is updated in database | |
| Fail: refill information | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Voucher row on voucher list and click “Update” button.  4. Actor inputs information to update voucher and click “Save” button. | 2. System redirects to 'Update voucher' page.  3. System displays a form with the following fields. - Voucher’s ID (read only)  - Voucher’s discount valid date (date valid from and date valid to)  - Voucher’s Description  - Voucher’s Discount value  - Voucher’s Quantity  6. System validates inputs.  7. System updates collection in database.  8. System displays confirmation message or error message. |
| **Exceptions** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor left ‘Description’ field blank.  2. Actor left ‘Discount value’ field blank.  3. Actor left ‘Quantity’ field blank.  4. Valid date in the past | System displays messages: - ‘Voucher’s Description cannot be blank’  - ‘Voucher’s Discount value cannot be blank’  - ‘Voucher’s Quantity cannot be blank’  - ‘Valid date cannot be in the past’ |
|
|



1. **Delete Voucher**

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use case name** | Delete Voucher | |
| **Actors** | Super admin | |
| **Description** | When super admin wants to delete a Voucher from the database | |
| **Requirements** | Actor provides voucher’s information | |
| **Pre-conditions** | Actor is logged in as super admin | |
| **Post-conditions** | Success: voucher is deleted from the database | |
| Fail: voucher is not deleted from the database | |
| **Basic flow** | **Actor’s actions:** | **System’s responses:** |
| 1. Actor chooses Voucher row on voucher list and click “Delete” button. | 2. System deletes voucher from database.  3. System displays confirmation message. |



## **Report**

### View Product Report

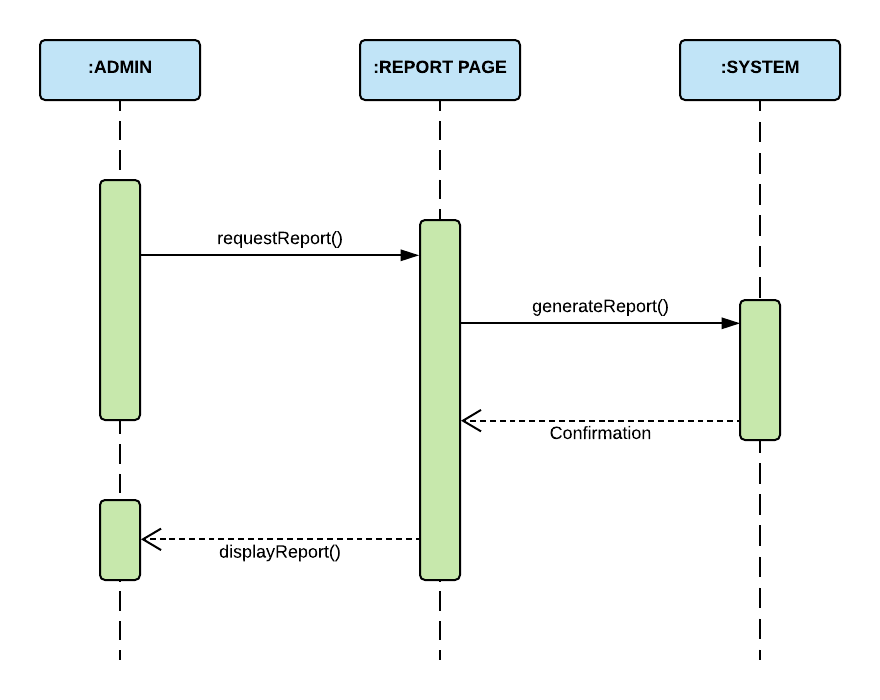
|  |  |  |
| --- | --- | --- |
| **Author** | Du Hue Phuong | |
| **Use case name** | View Product Report | |
| **Actors** | Administrator | |
| **Description** | When admin wants to view report on products | |
| **Requirements** | N/A | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: Display a report on products | |
| **Basic flow** | **Actor’s actions:** | **System’s responses** |
| 1. Actor chooses ‘Product Report’ in menu. | 2. System redirects to ‘Product Report’ page. |

### View Order Report

|  |  |  |
| --- | --- | --- |
| **Author** | Huynh Tan Phat | |
| **Use case name** | View Order Report | |
| **Actors** | Administrator | |
| **Description** | When admin wants to view report on orders | |
| **Requirements** | N/A | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: Display a report on orders | |
| **Basic flow** | **Actor’s actions:** | **System’s responses** |
| 1. Actor chooses ‘Order Report’ in menu. | 2. System redirects to ‘Order Report’ page. |

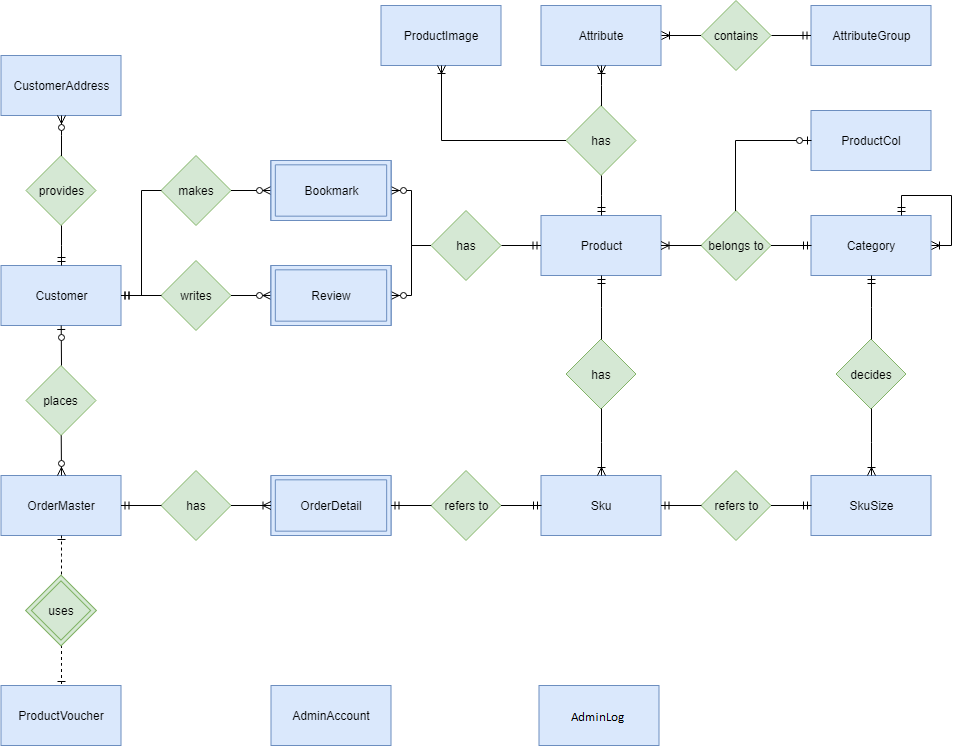
### View Customer Report

|  |  |  |
| --- | --- | --- |
| **Author** | Nguyen Khai Nguyen | |
| **Use case name** | View Customer Report | |
| **Actors** | Administrator | |
| **Description** | When admin wants to view report on customers | |
| **Requirements** | N/A | |
| **Pre-conditions** | Actor is logged in as admin or super admin | |
| **Post-conditions** | Success: Display a report on customers | |
| **Basic flow** | **Actor’s actions:** | **System’s responses** |
| 1. Actor chooses ‘Customer Report’ in menu. | 2. System redirects to ‘Customer Report’ page. |

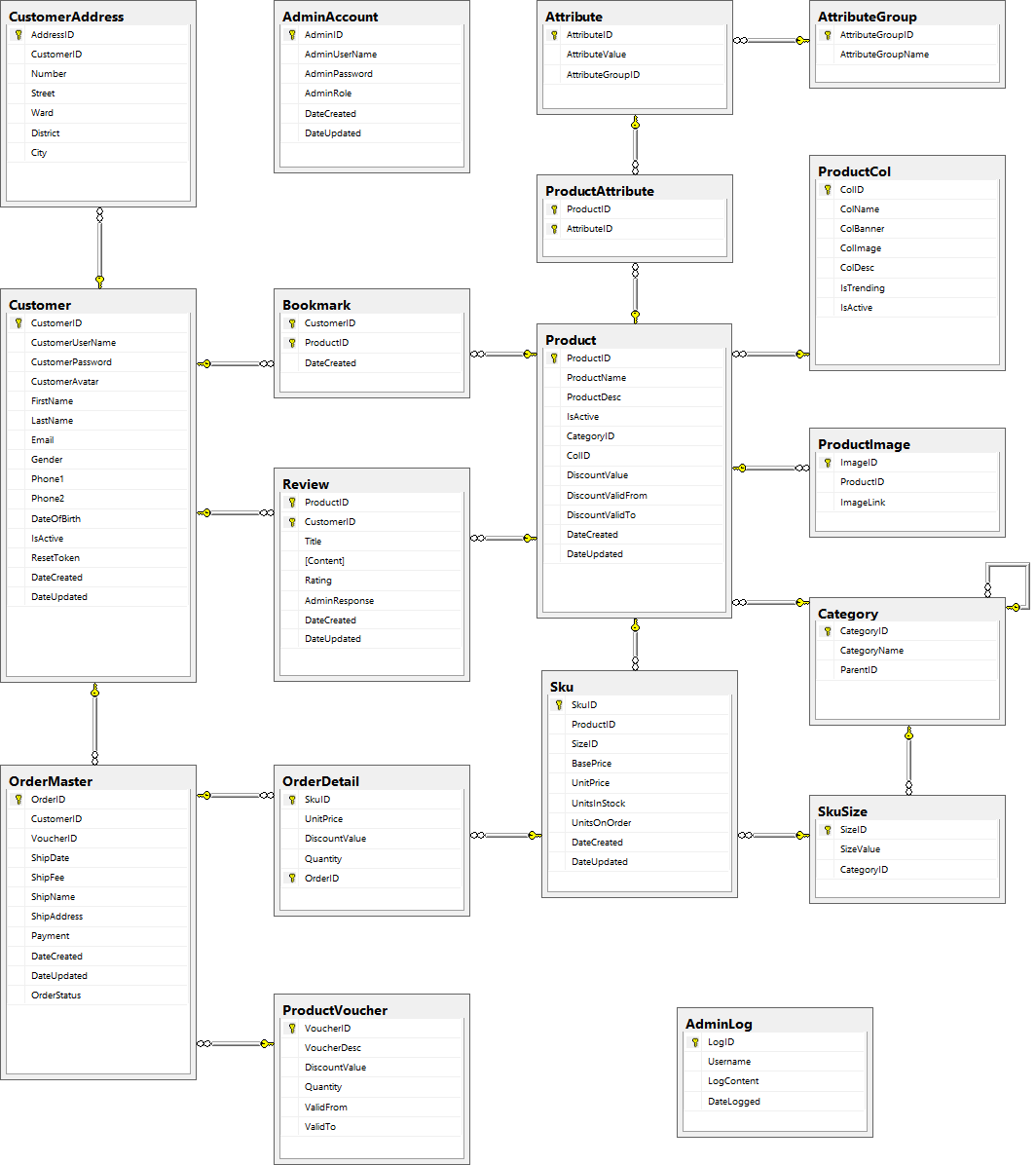


# Entity Relationship Diagram (ERD)

## **2.1 Logical ERD**



## **2.2 Physical ERD**



# Entity Properties

## **3.1 AdminAccount**



## **3.2 Customer**



## **3.3 CustomerAddress**



## **3.4 Category**



## **3.5 ProductCol**



## **3.6 AttributeGroup**



## **3.7 Attribute**



## **3.8 Product**



## **3.9 ProductImage**



## **3.10 ProductAttribute (join table)**



## **3.11 SkuSize**



## **3.12 Sku**



## **3.13 Bookmark**



## **3.14 Review**



## **3.15 Voucher**



## **3.16 OrderMaster**



## **3.17 OrderDetail**



## **3.18 AdminLog**



# Table Design

## **4.1 AdminAccount**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: AdminAccount** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | AdminID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of admin |
| 2 | AdminUserName | VARCHAR(20) | NOT NULL UNIQUE | Stores the Username of admin |
| 3 | AdminPassword | VARCHAR(20) | NOT NULL | Stores the phone of admin |
| 4 | AdminRole | VARCHAR(20) | NOT NULL | Stores the role of admin |
| 5 | DateCreated | DATETIME | NOT NULL | Stores the date at created |
| 6 | DateUpdated | DATETIME | NOT NULL | Stores the date at updated |

## **4.2 Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Customer** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | CustomerID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of customer |
| 2 | CustomerUserName | VARCHAR(20) | NOT NULL UNIQUE | Stores the username of customer |
| 3 | CustomerPassword | VARCHAR(20) | NOT NULL | Stores the password of customer |
| 4 | CustomerAvatar | VARCHAR(300) | NULL | Stores the avatar image of customer |
| 5 | FirstName | VARCHAR(50) | NOT NULL | Stores the first name of customer |
| 6 | LastName | VARCHAR(50) | NOT NULL | Stores the last name of customer |
| 7 | Email | VARCHAR(50) | NOT NULL UNIQUE | Stores the email of customer |
| 8 | Gender | VARCHAR(10) | NOT NULL | Stores the gender of customer |
| 9 | Phone1 | VARCHAR(20) | NOT NULL | Stores the first phone number of customer |
| 10 | Phone2 | VARCHAR(20) | NULL | Stores the second phone number of customer |
| 11 | DateOfBirth | DATE | NOT NULL | Stores the birthday of customer |
| 12 | IsActive | BIT | NOT NULL DEFAULT (1) | Stores the active status of customer |
| 13 | ResetToken | VARCHAR(50) | NULL | Store reset token |
| 14 | DateCreated | DATETIME | NOT NULL | Stores the date at created |
| 15 | DateUpdated | DATETIME | NOT NULL | Stores the date at updated |

## **4.3 Customer’s Address**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: CustomerAddress** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | AddressID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of customer's address |
| 2 | CustomerID | INT | NOT NULL  FOREIGN KEY | Stores the ID of customer |
| 3 | Number | VARCHAR(20) | NOT NULL | Stores the number of customer's address |
| 4 | Street | VARCHAR(100) | NOT NULL | Stores the street of customer's address |
| 5 | Ward | VARCHAR(20) | NOT NULL | Stores the ward of customer's address |
| 6 | District | VARCHAR(20) | NOT NULL | Stores the district of customer's address |
| 7 | City | VARCHAR(20) | NOT NULL | Stores the city of customer's address |

## **4.4 Category**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Category** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | CategoryID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of category |
| 2 | CategoryName | VARCHAR(100) | NOT NULL UNIQUE | Stores the name of category |
| 3 | ParentID | INT | NULL  FOREIGN KEY | Stores the ID of parent category |

## **4.5 Product Collection**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: ProductCol** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | ColID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of collection |
| 2 | ColName | VARCHAR(100) | NOT NULL UNIQUE | Stores the name of collection |
| 3 | ColBanner | VARCHAR(300) | NOT NULL | Stores the banner of collection |
| 4 | ColImage | VARCHAR(300) | NOT NULL | Stores the image of collection |
| 5 | ColDesc | TEXT | NOT NULL | Stores the description of collection |
| 6 | IsTrending | BIT | NOT NULL | Stores the trending status of collection |
| 7 | IsActive | BIT | NOT NULL | Stores the active status of collection |

## **4.6 Attribute Group**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: AttributeGroup** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | AttributeGroupID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of attribute group |
| 2 | AttributeGroupName | VARCHAR(50) | NOT NULL UNIQUE | Stores the name of attribute group |

## **4.7 Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Attribute** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | AttributeID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of attribute |
| 2 | AttributeValue | VARCHAR(50) | NOT NULL | Stores the value of attribute |
| 3 | AttributeGroupID | INT | NOT NULL  FOREIGN KEY | Stores the ID of attribute group |

## **4.8 Product**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Product** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | ProductID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of product |
| 2 | ProductName | VARCHAR(100) | NOT NULL | Stores the name of product |
| 3 | ProductDesc | TEXT | NOT NULL | Stores the description of product |
| 4 | IsActive | BIT | NOT NULL DEFAULT (1) | Stores the active status of product |
| 5 | CategoryID | INT | NOT NULL  FOREIGN KEY | Stores the ID category of product |
| 6 | ColID | INT | NULL  FOREIGN KEY | Stores the ID collection of product |
| 7 | DiscountValue | INT | NULL DEFAULT (0) | Stores the value of discount |
| 8 | DiscountValidFrom | DATETIME | NULL | Stores the begin date of discount |
| 9 | DiscountValidTo | DATETIME | NULL | Stores the end date of discount |
| 10 | DateCreated | DATETIME | NOT NULL | Stores the date at created |
| 11 | DateUpdated | DATETIME | NOT NULL | Stores the date at updated |

## **4.9 ProductImage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: ProductImage** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | ImageID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of image |
| 2 | ProductID | INT | NOT NULL  FOREIGN KEY | Stores the ID of product |
| 3 | ImageLink | VARCHAR(300) | NOT NULL | Stores the link of image |

## **4.10 ProductAttribute (join table)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: ProductAttribute** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | ProductID | INT | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of product |
| 2 | AttributeID | INT | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of attribute |

## **4.11 Sku Size**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: SkuSize** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | SizeID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of SkuSize |
| 2 | SizeValue | VARCHAR(5) | NOT NULL | Stores the value of size |
| 3 | CategoryID | INT | NOT NULL  FOREIGN KEY | Stores the ID of category |

## **4.12 Sku**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Sku** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | SkuID | VARCHAR(20) | NOT NULL PRIMARY KEY | Stores the ID of Sku |
| 2 | ProductID | INT | NOT NULL  FOREIGN KEY | Stores the ID of product |
| 3 | SizeID | INT | NOT NULL  FOREIGN KEY | Stores the ID of SkuSize |
| 4 | BasePrice | INT | NOT NULL | Stores the base price of Sku |
| 5 | UnitPrice | INT | NOT NULL | Stores the unit price of Sku |
| 6 | UnitsInStock | INT | NOT NULL | Stores the units in stock of Sku |
| 7 | UnitsOnOrder | INT | NOT NULL  DEFAULT(0) | Stores the units on order of Sku |
| 8 | DateCreated | DATETIME | NOT NULL | Stores the date at created |
| 9 | DateUpdated | DATETIME | NOT NULL | Stores the date at updated |

## **4.13 Voucher**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: ProductVoucher** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | VoucherID | VARCHAR(20) | NOT NULL PRIMARY KEY | Stores the ID of voucher |
| 2 | VoucherDesc | TEXT | NOT NULL | Stores the description of voucher |
| 3 | DiscountValue | INT | NOT NULL | Stores the discount value of voucher |
| 4 | Quantity | INT | NOT NULL | Stores the quantity of voucher |
| 5 | ValidFrom | DATETIME | NOT NULL | Stores the begin date of voucher |
| 6 | ValidTo | DATETIME | NOT NULL | Stores the end date of voucher |

## **4.14 Review**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Review** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | ProductID | INT | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of Produtct |
| 2 | CustomerID | INT | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of Customer |
| 3 | Title | VARCHAR(100) | NOT NULL | Stores the title of review |
| 4 | Content | TEXT | NOT NULL | Stores the content of review |
| 5 | Rating | FLOAT | NOT NULL | Stores the rating of review |
| 6 | AdminResponse | TEXT | NULL | Stores the response of admin |
| 7 | DateCreated | DATETIME | NOT NULL | Stores the date at created |
| 8 | DateCreated | DATETIME | NOT NULL | Stores the date at updated |

## **4.15 Bookmark**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Bookmark** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | CustomerID | INT | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of customer |
| 2 | ProductID | INT | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of product |
| 3 | DateCreated | DATETIME | NOT NULL | Stores the date at created |

## **4.16 Order Master**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: OrderMaster** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | OrderID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of order |
| 2 | CustomerID | INT | NULL  FOREIGN KEY | Stores the ID of customer |
| 3 | VoucherID | VARCHAR(20) | NULL | Stores the ID of voucher |
| 4 | ShipDate | DATE | NOT NULL | Stores the date shipment of order |
| 5 | ShipFee | INT | NOT NULL | Stores the fee shipment of order |
| 6 | ShipName | VARCHAR(100) | NOT NULL | Stores the name shipment of order |
| 7 | ShipAddress | TEXT | NOT NULL | Stores the shipping address of order |
| 8 | DateCreated | DATETIME | NOT NULL | Stores the date at created |
| 9 | DateUpdated | DATETIME | NOT NULL | Stores the date at updated |
| 10 | OrderStatus | VARCHAR(20) | NOT NULL | Stores the status of order |
| 11 | Payment | VARCHAR(10) | NOT NULL | Stores the payment method |

## **4.17 Order Detail**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: OrderDetail** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | SkuID | VARCHAR(20) | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of Sku |
| 2 | OrderID | INT | NOT NULL PRIMARY KEY  FOREIGN KEY | Stores the ID of order master |
| 3 | UnitPrice | INT | NOT NULL | Stores the unit price of Sku |
| 4 | DiscountValue | INT | NULL | Stores the value of discount |
| 5 | Quantity | INT | NOT NULL | Stores the quantity of product |

## **4.18 Admin Log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: OrderDetail** | | | | |
| **No.** | **Column Name** | **Data Type** | **Constraint** | **Description** |
| 1 | LogID | INT  IDENTITY(1,1) | NOT NULL PRIMARY KEY | Stores the ID of log |
| 2 | Username | INT | NOT NULL | Stores the Username of the admin who performs the activity |
| 3 | LogContent | INT | NOT NULL | Stores the activity |
| 4 | DateLogged | DATE | NOT NULL | Stores the date on which the activity is performed |

REVIEW 2 – TASK SHEET

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Contents | Member | Activity Plan | Preparation | |
| Start Date | Actual Days | Status |
| 1 | Use Case and Sequence Diagram | Du Hue Phuong  Nguyen Khai Nguyen  Huynh Tan Phat | 010/04/2018 | 3 | Completed |
| 2 | Entity Relationship Diagram (ERD) |
| 3 | Entity Properties |
| 4 | Table Design |

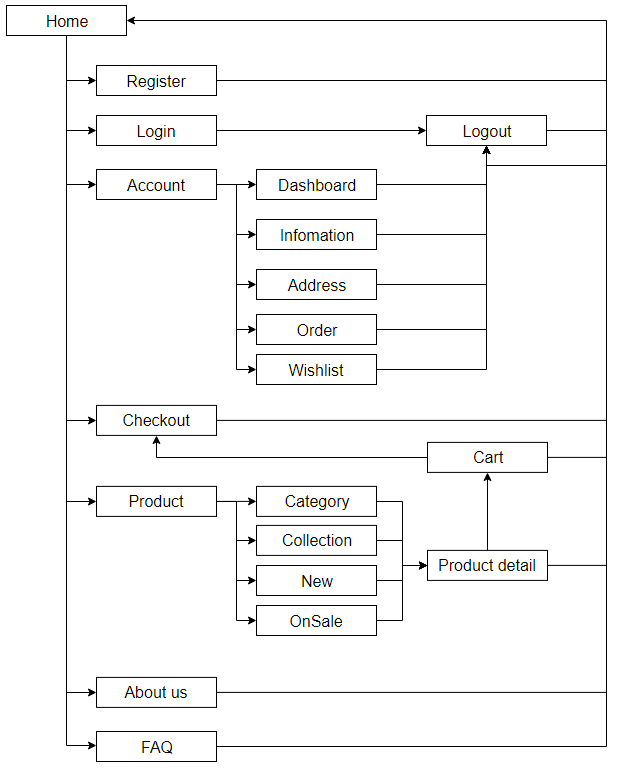
Plan prepared by: Group 2 April 21st, 2018  
 **Group Leader Faculty**

**Du Hue Phuong Mr. Ngo Phuoc Nguyen**

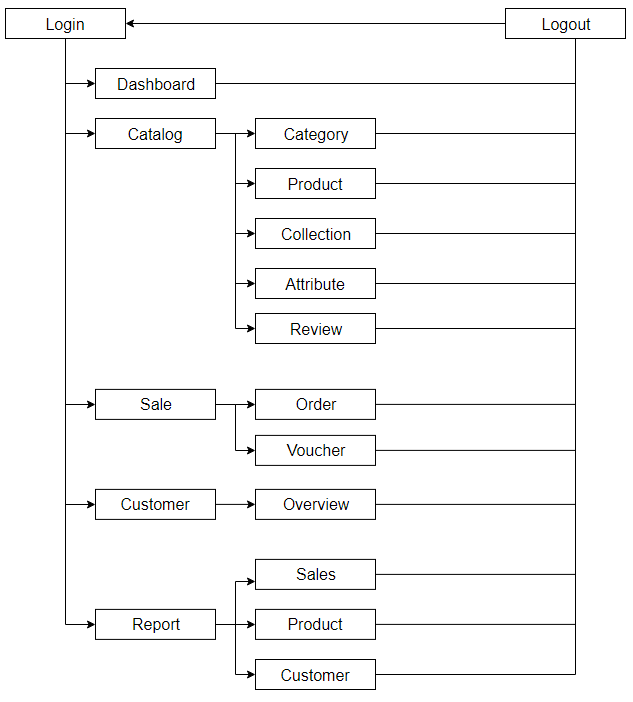
REVIEW 3

# Site Map

## **Client’s Site Map**



## **Admin’s Site Map**



# GUI Design

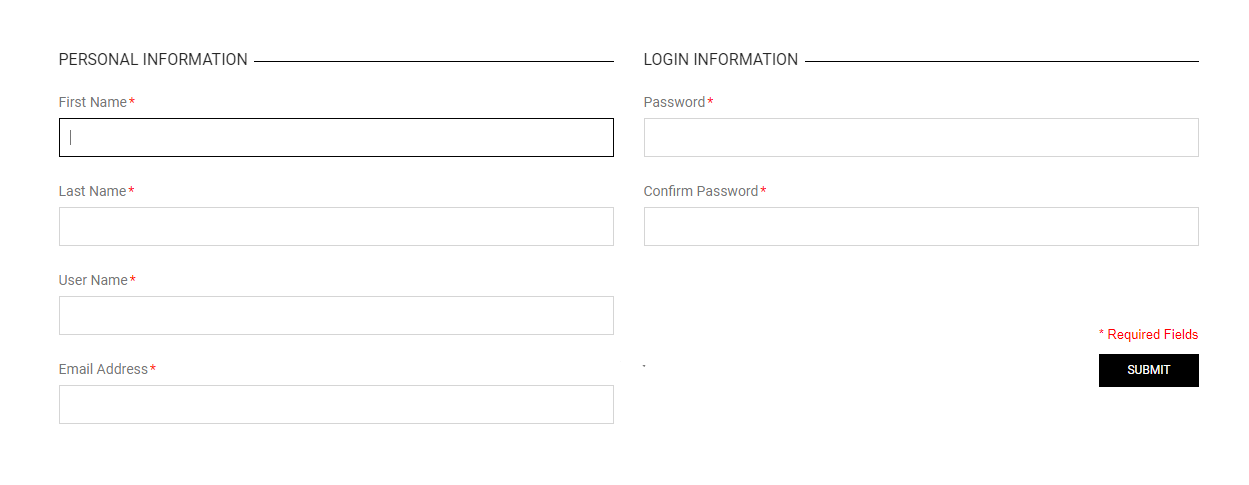
## **Client’s Side**

### Home



|  |  |
| --- | --- |
| Description | Home page |
| Connect From | N/A |
| Connect To | Login page, Product page, Profile, Cart, Checkout, Compare |
| Validation | N/A |
| Initial Values | N/A |
| Events | - Navigate the website by clicking on menu items or products. |

### Register



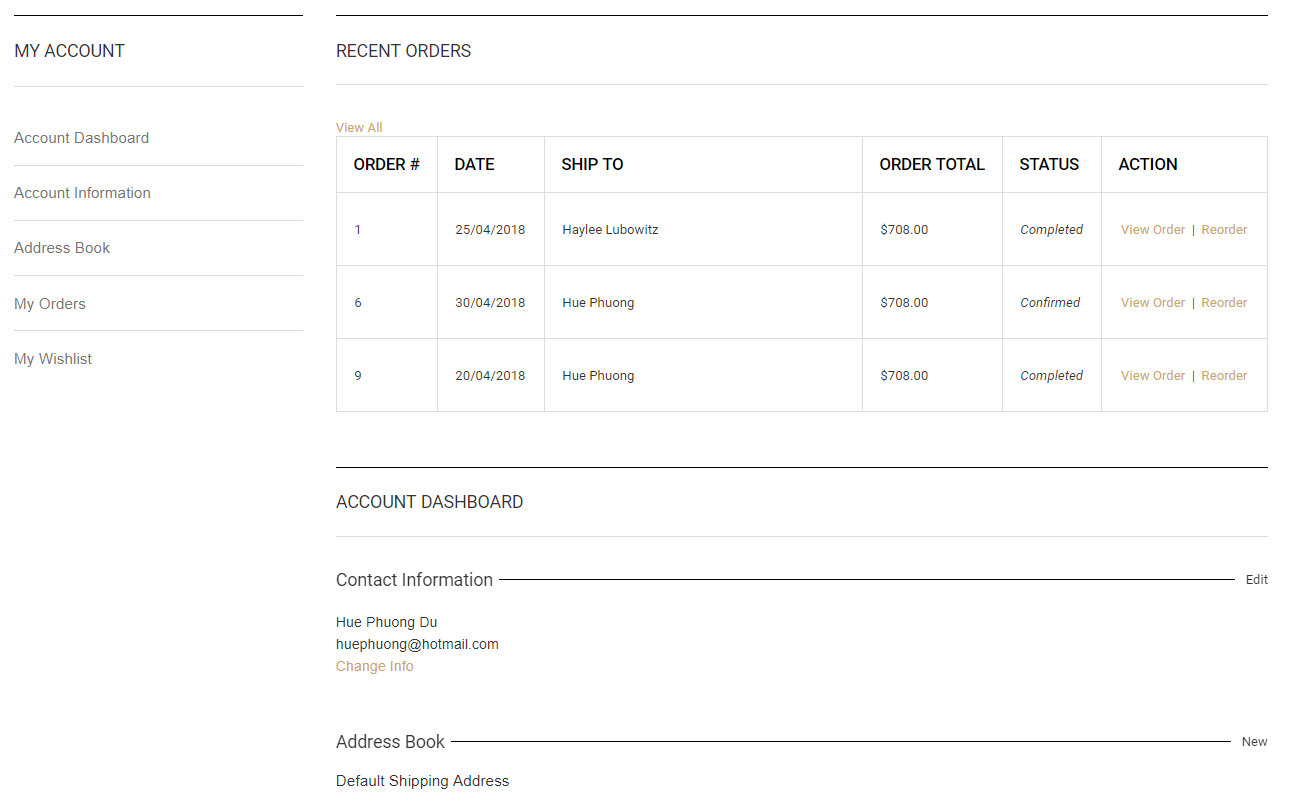
|  |  |
| --- | --- |
| Description | Guest signs up for new account |
| Connect From | Home |
| Connect To | Login page, Home |
| Validation | - All fields: not blank  - First name & last name: must be not contains special characters - Username: must be unique - Email: must be valid email format  - Password: must be between 6 to 20 characters - Confirm password: must be the same as password |
| Initial Values | N/A |
| Events | - When click button "Submit", check email and password. If success, go to login page |

### Login



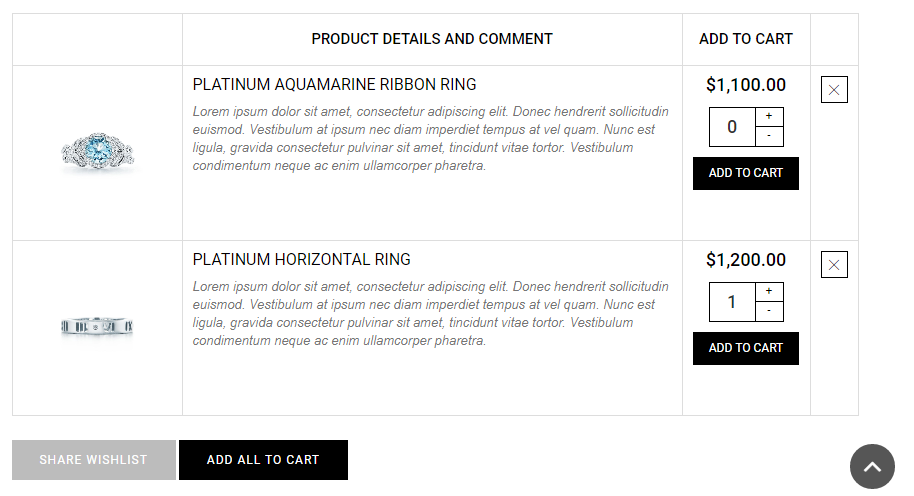
|  |  |
| --- | --- |
| Description | Login with email and password |
| Connect From | Home |
| Connect To | Customer’s profile, Forget password |
| Validation | All fields: not blank - Email: must be valid email format - Password: must be more than 6 characters  - Email & Password must be the same with database |
| Initial Values | N/A |
| Events | - When click button "Login", check email and password, then login. Login success, go to profile page - When click "Forgot Your Password?" link, go to Forgot password page |

### Customer’s Dashboard



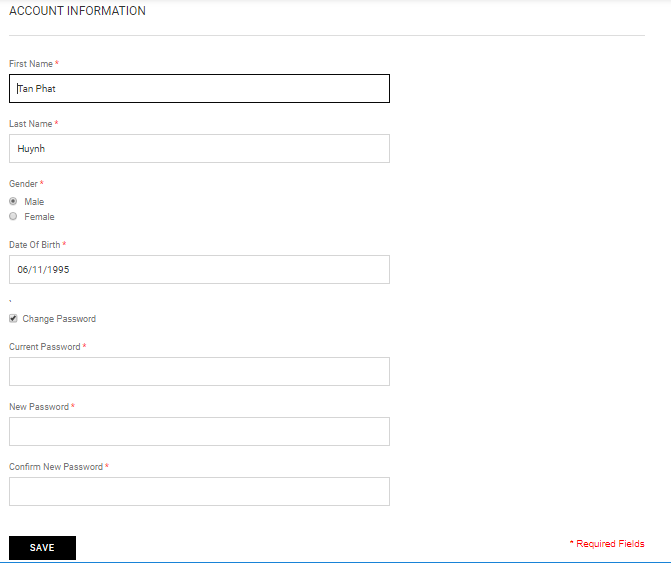
|  |  |
| --- | --- |
| Description | Customer’s dashboard (profile) |
| Connect From | Home, Login page |
| Connect To | Update profile, Update Address, Wish list, Order details |
| Validation | Must log in to view |
| Initial Values | N/A |
| Events | - When click tab “Account Information”, view customer’s profile  - When click tab “Address Book”, view all addresses  - When click “View Order”, view Order details  - When click “Change info”, view Update profile page |

### Customer’s Wish list



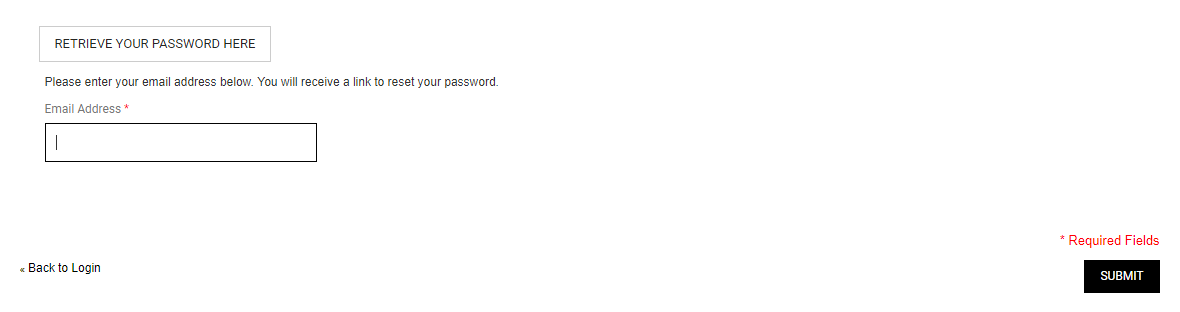
|  |  |
| --- | --- |
| Description | Show customer’s wish list (bookmarks) |
| Connect From | Customer’s dashboard |
| Connect To | Cart, share wish list |
| Validation | Customer is logged |
| Initial Values | List of bookmarked products |
| Events | - When click button "Add to cart", add selected product to shopping cart - When click button "Add all to cart", add all products to shopping cart - When click button "Share", share all product to customer's friends via email. |

### Update Customer’s Profile



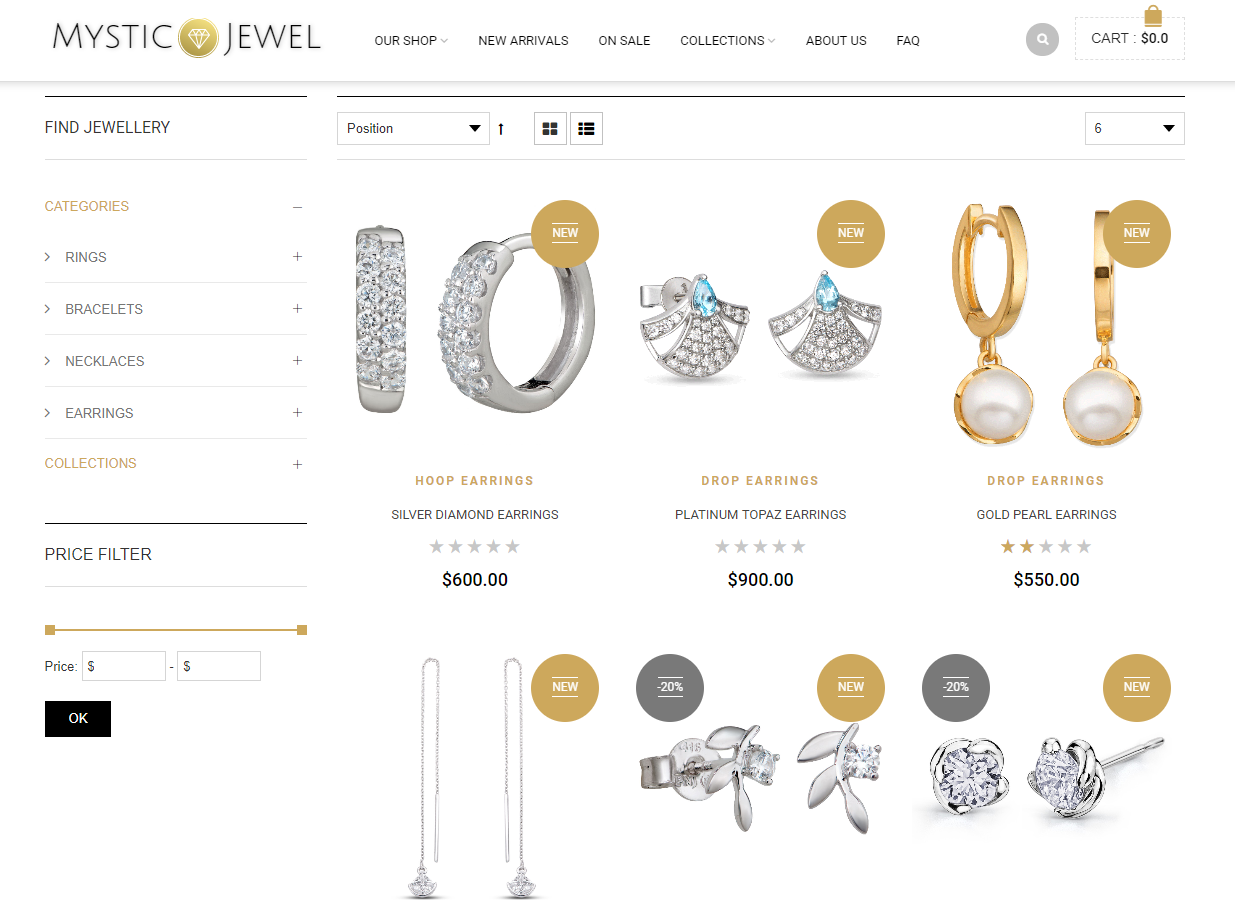
|  |  |
| --- | --- |
| Description | Customer’s profile update form |
| Connect From | Customer’s dashboard |
| Connect To | Customer’s dashboard |
| Validation | - All fields cannot be blank  - First Name is no more than 50 characters  - Last Name is not more than 50 characters  - Current password must be valid  - New Password must be between 6 – 24 characters  - Confirm New Password must match New Password |
| Initial Values | - First Name  - Last Name  - Gender  - Date of Birth |
| Events | - When click button “Save”, update customer’s information in database  - When check ‘Change password’, allow customer to change their password. |

### Reset Customer’s Password



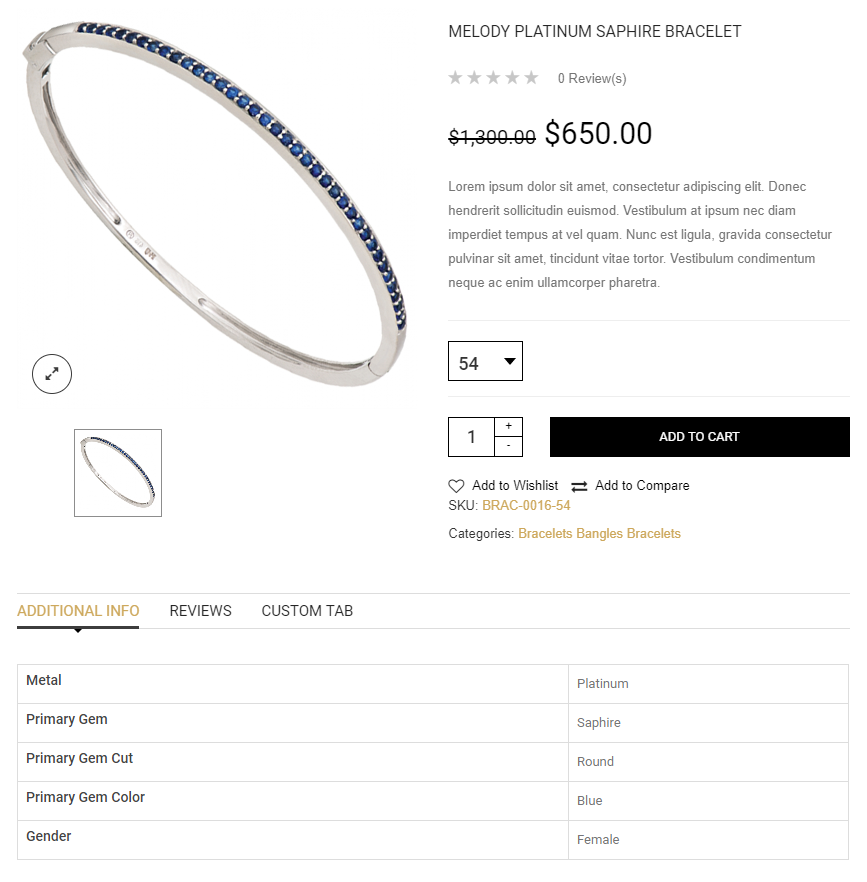
|  |  |
| --- | --- |
| Description | Retrieve password |
| Connect From | Login page |
| Connect To | Login page |
| Validation | - Email must not be blank and must be correct email format  - Email exists in the database |
| Initial Values | N/A |
| Events | - When click button "Submit", check email and send Reset Token to customer via email  - When click button “Back to Login”, redirect to Login page |

### Product



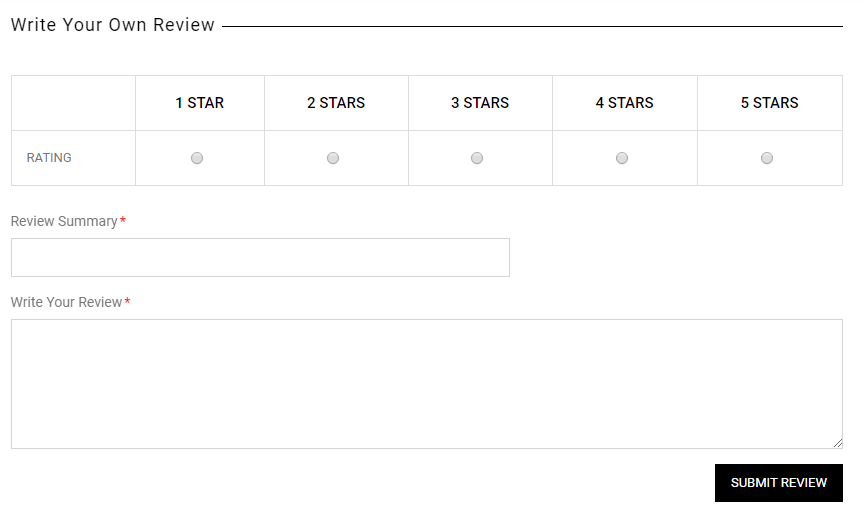
|  |  |
| --- | --- |
| Description | Show list of products |
| Connect From | Home, top Navigation |
| Connect To | Product detail, cart, compare list |
| Validation | N/A |
| Initial Values | List of products following given criteria (category, collection, attributes, price, etc.) |
| Events | - When click on product, redirect to product detail  - When click on “Quick View”, open Product quick view  - When click on “Compare” icon, add Product to compare list  - When choose “Sort from Newest to Oldest”, sort products from newest to oldest, and vice versa  - When choose “Sort from Lowest Price to Highest Price, sort products from lowest price to highest price, and vice versa |

### Product Details



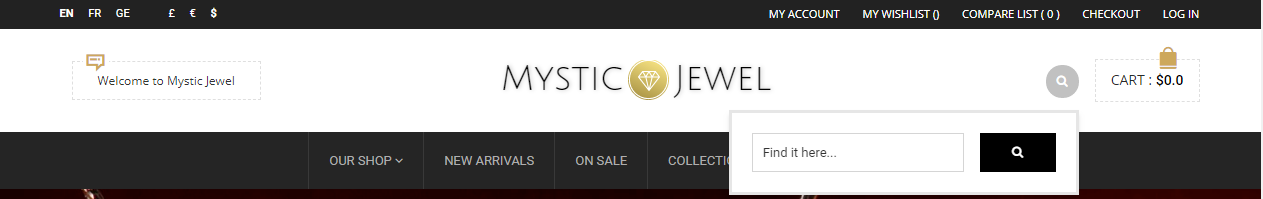
|  |  |
| --- | --- |
| Description | Show product’s detail |
| Connect From | Home, product page, cart, compare list |
| Connect To | Cart, Review |
| Validation | N/A |
| Initial Values | Product’s detail: name, price, size, attribute, discount value, description |
| Events | - When click button “Add To Cart”, add product to cart  - When choose a different size, reload price. |

### Product Review



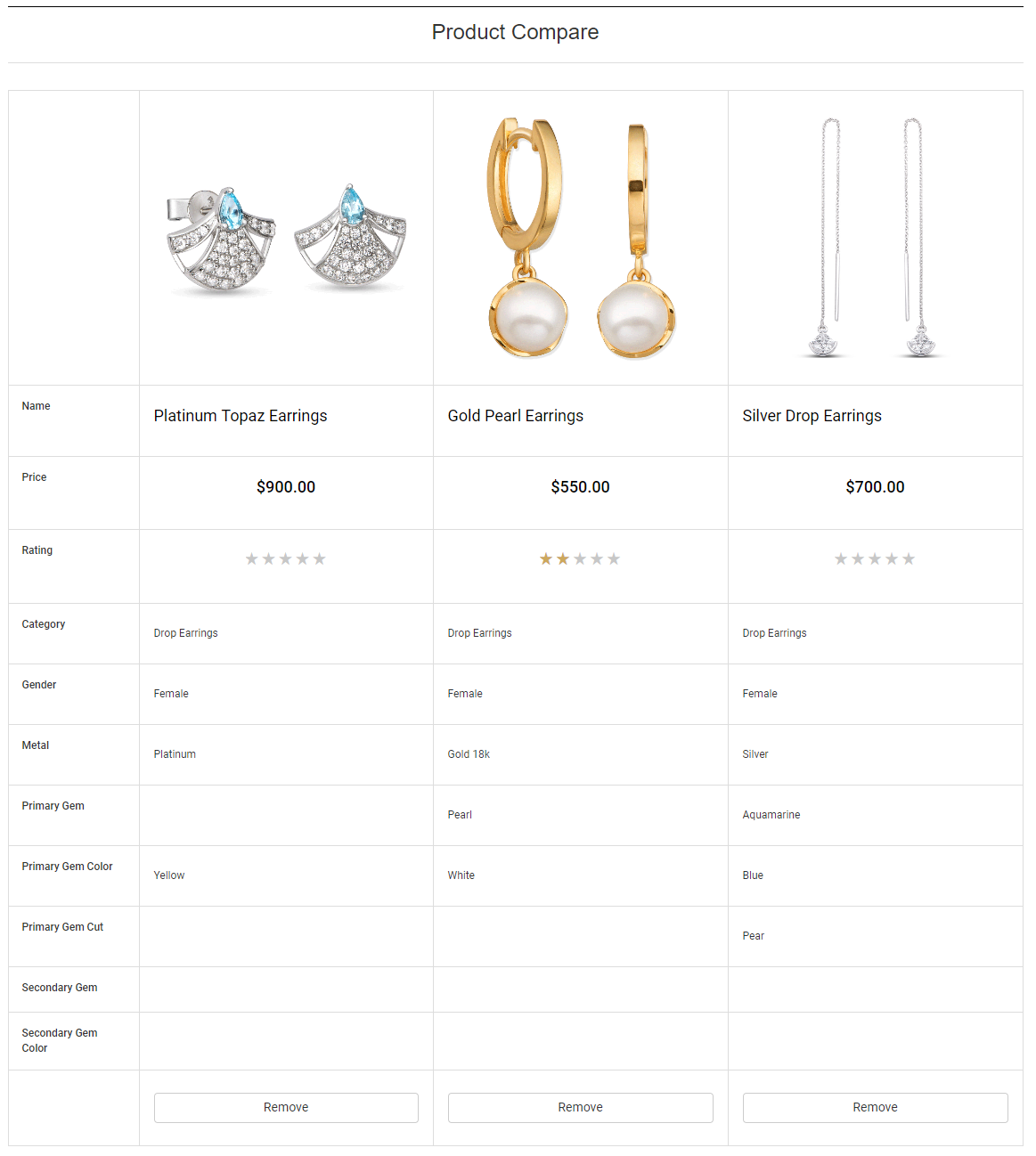
|  |  |
| --- | --- |
| Description | Write review for product |
| Connect From | Product’s detail |
| Connect To | Product’s detail |
| Validation | - Customer is logged in - Customer has bought product - All fields: not blank  - Review Summary: 5 – 30 characters  - Write your review: 50 – 1000 characters |
| Initial Values | N/A |
| Events | - When click button "Submit review", save customer's review to database. |

### Search Product



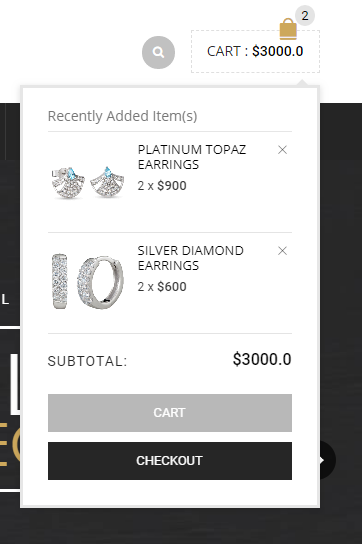
|  |  |
| --- | --- |
| Description | Search Product |
| Connect From | Home, anywhere |
| Connect To | Product page |
| Validation | Must not be blank |
| Initial Values | N/A |
| Events | - When click search icon, show a list of matching product |

### Compare Product



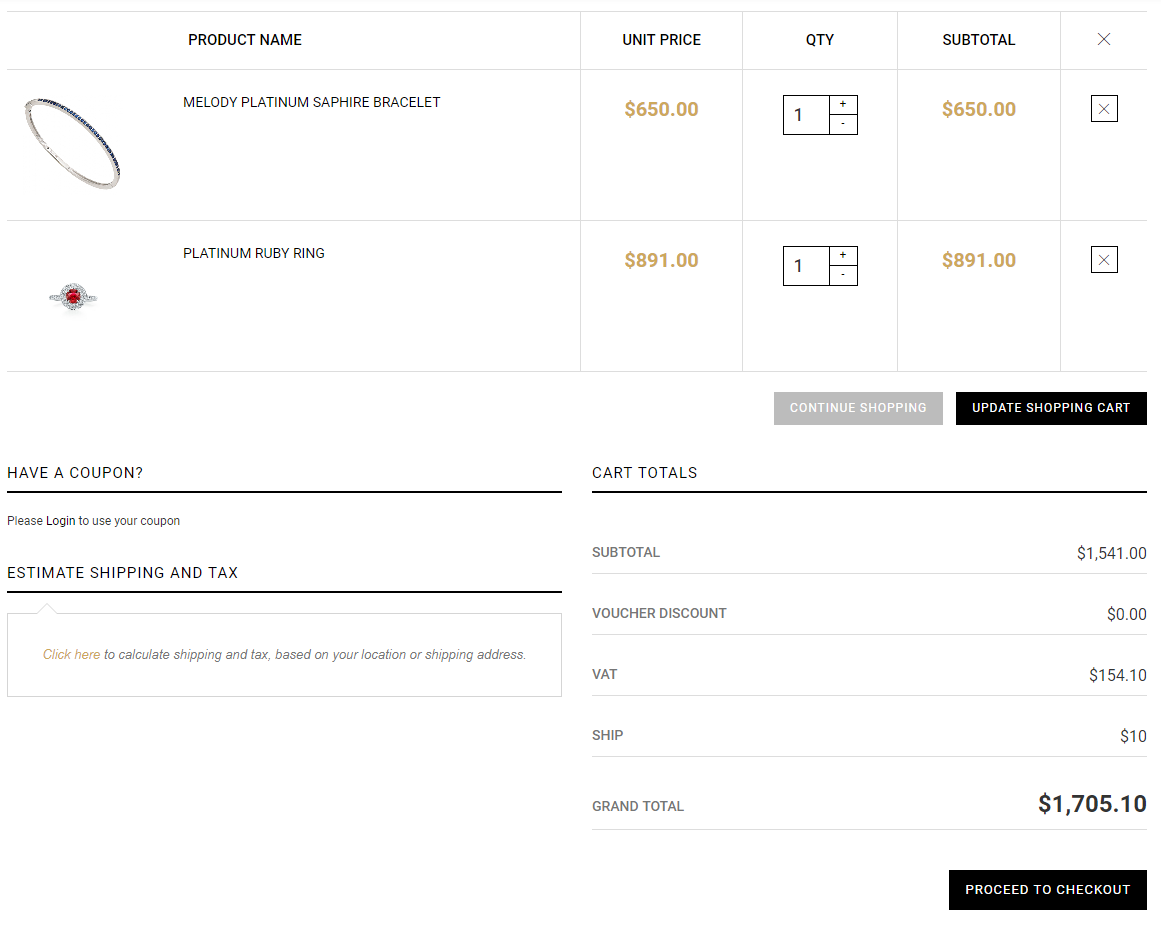
|  |  |
| --- | --- |
| Description | Compare products |
| Connect From | Home, Product page, Product’s detail |
| Connect To | Product’s detail, Cart |
| Validation | Must add at least one product to compare |
| Initial Values | List of products being compared |
| Events | - When click Product Name or Product Image, redirect to Product’s detail page  - When click on “Remove”, remove product from compare list |

### Cart



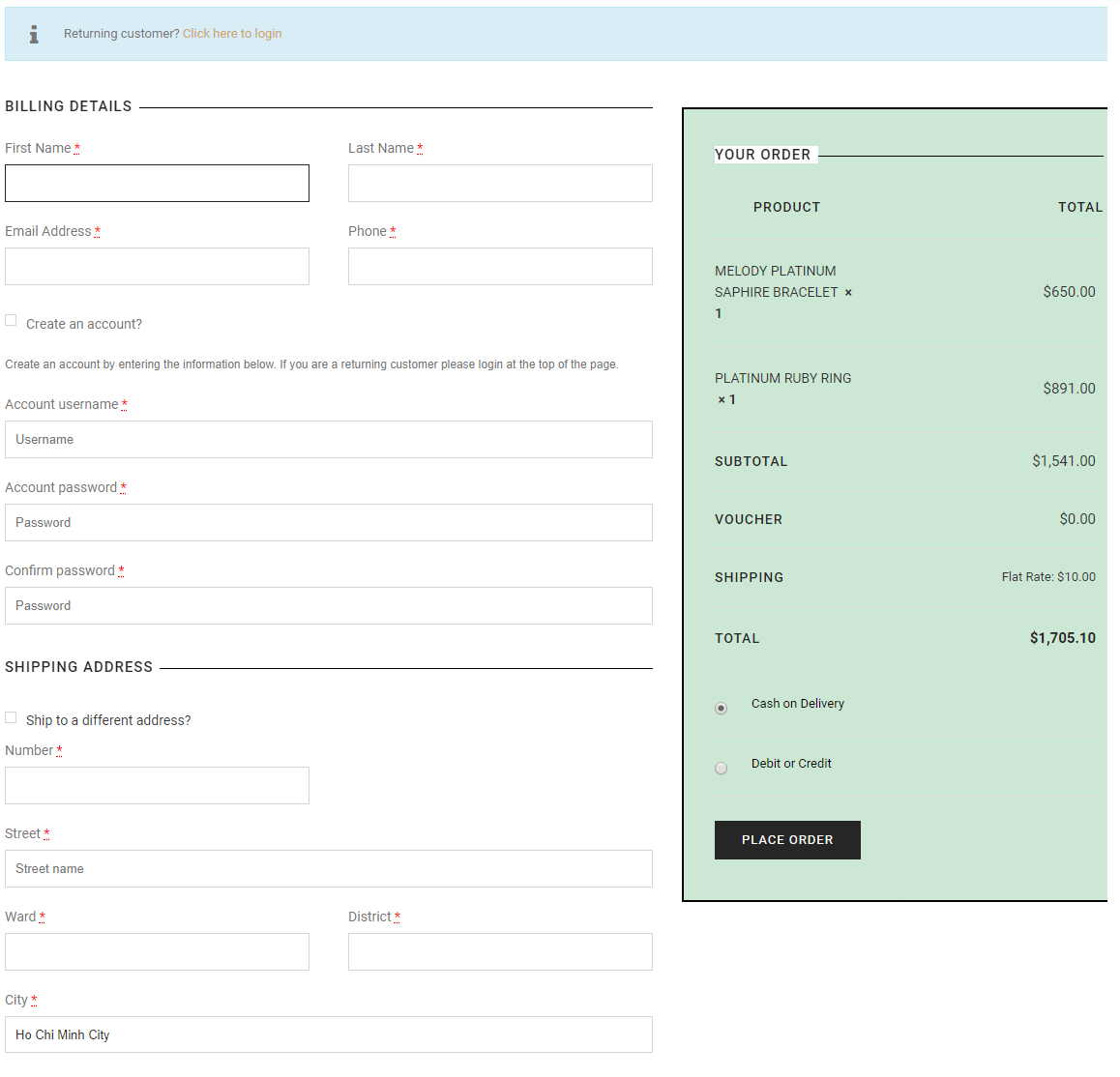
|  |  |
| --- | --- |
| Description | Show a list of products which have been added to cart |
| Connect From | Home, Product page, Product’s detail |
| Connect To | Checkout page |
| Validation | N/A |
| Initial Values | List of products added to cart |
| Events | - When click button "Cart", go to Cart page - When click button "Checkout", go to order page  - When click “x” icon, remove item from cart |

### Cart Page



|  |  |
| --- | --- |
| Description | Cart page |
| Connect From | Top Navigation |
| Connect To | Place Order page |
| Validation | Cart is not empty |
| Initial Values | List of products in cart |
| Events | - If customer is logged in, they can submit a voucher for discount  - Customer can estimate shipping fee by submit their address  - When click “Proceed To Checkout” button, redirect to Place Order page  - When adjust quantity, update quantity and subtotal  - When click “x” icon, remove item from cart |

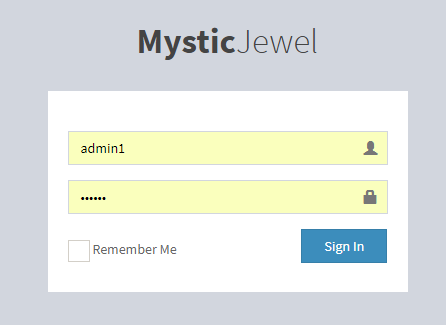
### Place Order Page



|  |  |
| --- | --- |
| Description | Show Order page |
| Connect From | Cart |
| Connect To | Home, Payment page |
| Validation | Cart is not empty  - All fields: not blank  - Email: valid email address  - Phone: contains only digits, between 7 – 12 digits |
| Initial Values | List of products in cart  Optional: Customer’s First Name, Last Name, Email, Phone, Address |
| Events | - When check “Create an account”, validate information and create a new account  - When check “Ship to different address”, save new address in order instead of default address  - When check “Credit/Debit” method, redirect to Payment page after Place Order |

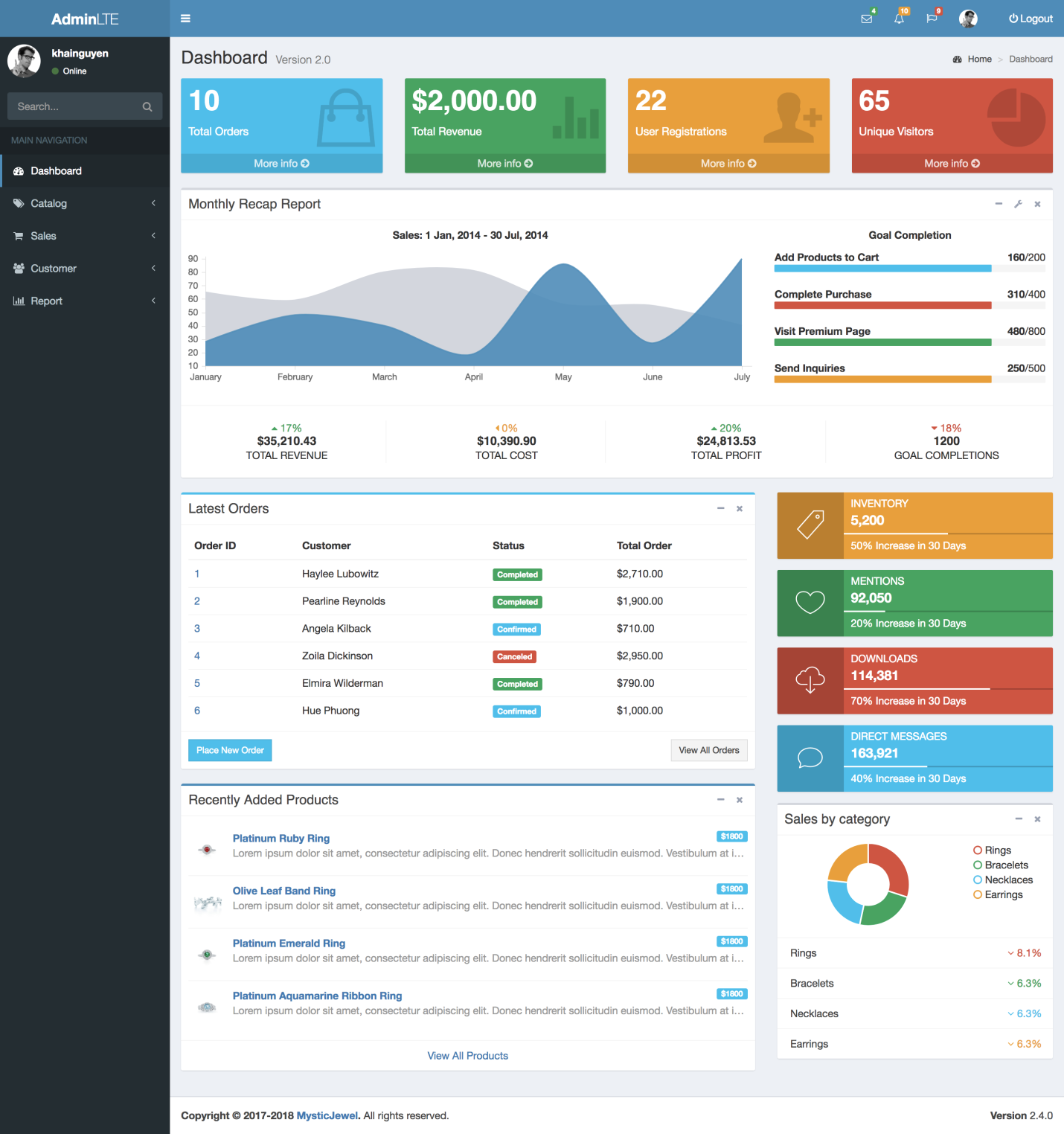
## **Admin’s Side**

### Admin Login



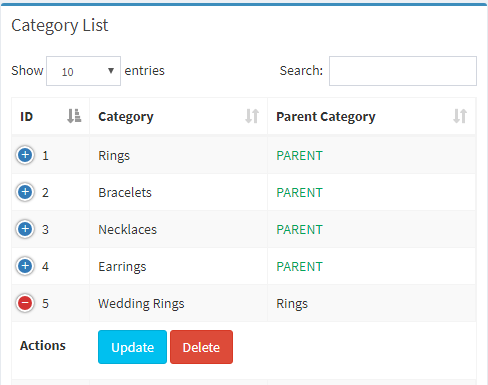
|  |  |
| --- | --- |
| Description | Login as admin or super admin |
| Connect From | N/A |
| Connect To | Admin’s dashboard |
| Validation | - All fields: not blank - username must be more than 5 character - Password must be more than 6 character |
| Initial Values | N/A |
| Events | - When click button "Sign in", check username and password, then login - When click checkbox "Remember", save username and password on browser's cookie |

### Admin’s Dashboard



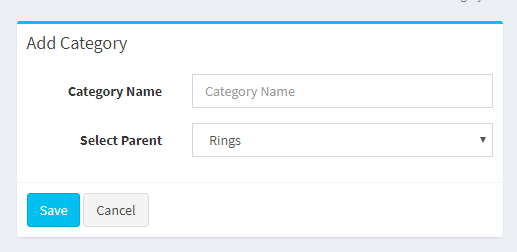
|  |  |
| --- | --- |
| Description | Admin Dashboard |
| Connect From | Login page |
| Connect To | All |
| Validation | N/A |
| Initial Values | N/A |
| Events | - When click on menu tabs, redirect to respective pages |

### Category List



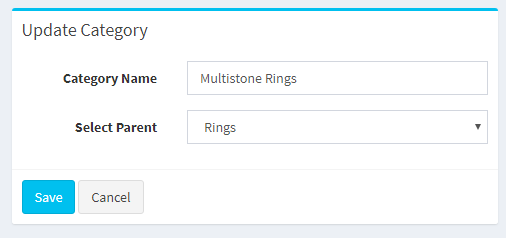
|  |  |
| --- | --- |
| Description | Category overview |
| Connect From | Dashboard |
| Connect To | Category: update, delete |
| Validation | N/A |
| Initial Values | Category list |
| Events | - when select "Show" option, show result with amount record selected  - when input text field search, show result Category match keyword  - when click "Update" button, go to category-update page - when click "Delete" button, check and delete category in database |

### Category Add



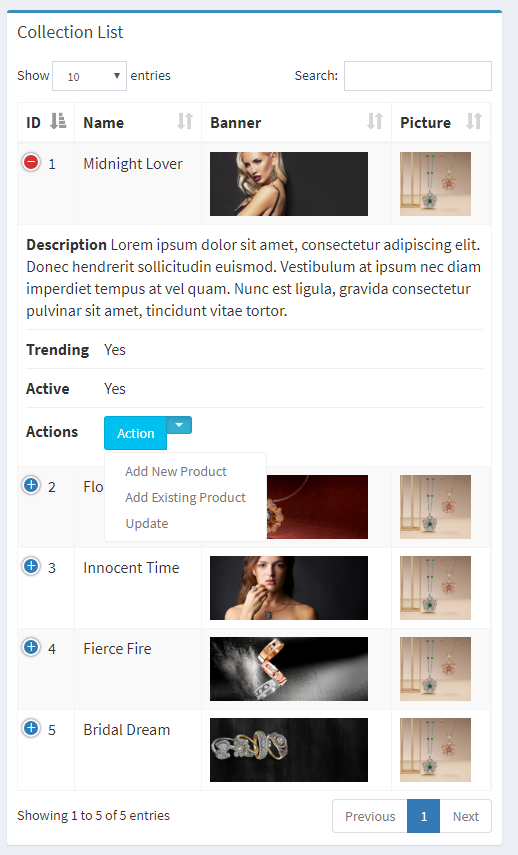
|  |  |
| --- | --- |
| Description | Insert new category |
| Connect From | Category-overview page |
| Connect To | category-overview page |
| Validation | - Not Blank: category name - category name is unique |
| Initial Values | Category parent list |
| Events | - when click "Save" button, save new category to database and go to category-overview page - when click "Cancel" button, reset all fields to blank |

### Category Update



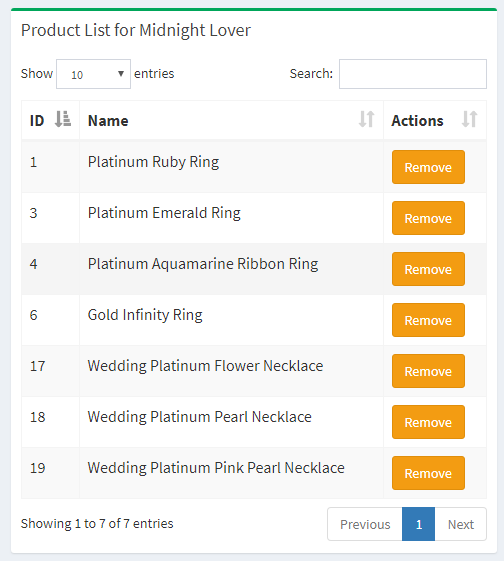
|  |  |
| --- | --- |
| Description | Update category |
| Connect From | Category-overview page |
| Connect To | category-overview page |
| Validation | - Not Blank: category name - category name is unique |
| Initial Values | Category parent list |
| Events | - when click "Save" button, update category to database and go to category-overview page - when click "Cancel" button, reset all fields to blank |

### Collection list

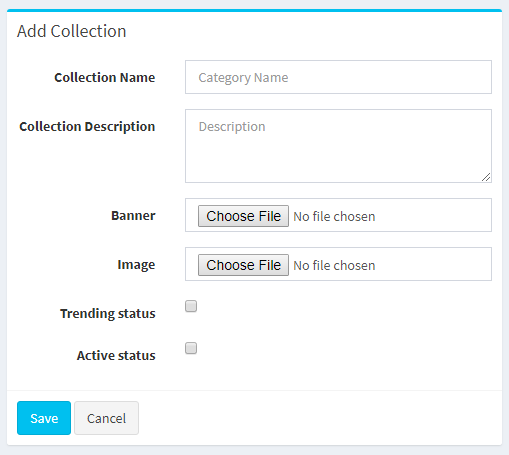


|  |  |
| --- | --- |
| Description | show list collection |
| Connect From | admin-page |
| Connect To | product-add page, collection-update |
| Validation | N/A |
| Initial Values | Collection list |
| Events | - when select "Show" option, show result with amount record selected  - when input text field search, show result collection match keyword  - when click "Add new product" button, go to product-add page - when click "Add existing product" button, go to product-update page  - when click "Update" button, go to collection-update page |

### Collection’s product list

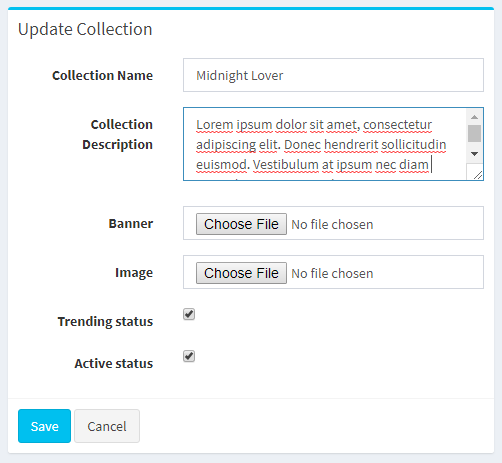


### Collection Add



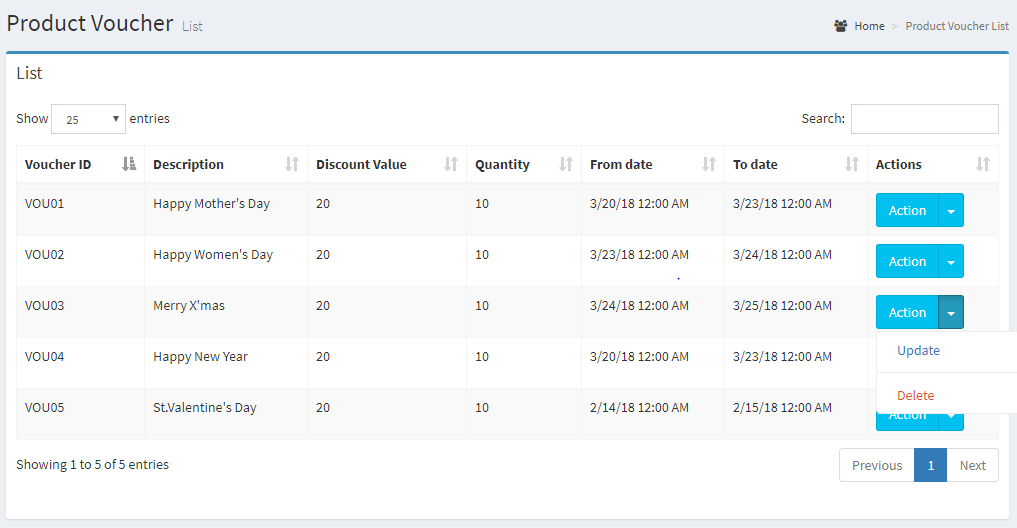
|  |  |
| --- | --- |
| Description | Insert new collection |
| Connect From | admin-page |
| Connect To | collection-list page |
| Validation | - Not Blank: Collection name, Collection Description, Banner, Image. - Collection name is unique and must be more than 6 characters  - Collection description must be more than 50 characters  - Banner & Image: file upload must has extension is .png or .jpg |
| Initial Values | N/A |
| Events | - when click "Save" button, save new collection to database and go to collection-list page - when click "Cancel" button, reset all fields to blank |

### Collection Update



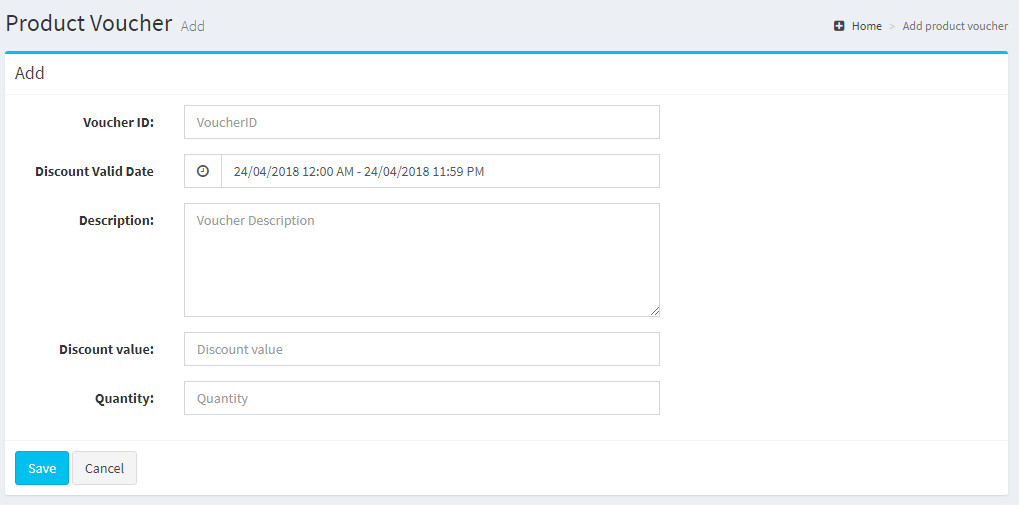
|  |  |
| --- | --- |
| Description | Edit collection |
| Connect From | collection-list page |
| Connect To | collection-list page |
| Validation | - Not Blank: Collection name, Collection Description, Banner, Image. - Collection name is unique and must be more than 6 characters  - Collection description must be more than 50  Characters  - Banner & Image: file upload must has extension is .png, .jpg |
| Initial Values | All fields of collection is edited |
| Events | - when click "Save" button, edit and save collection to database and go to collection-list page - when click "Cancel" button, reset all fields to blank |

### Voucher list



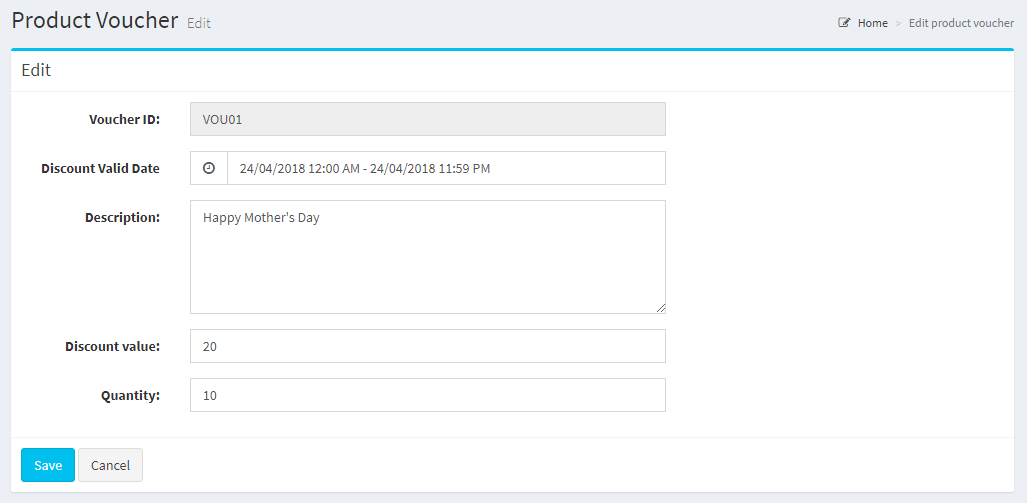
|  |  |
| --- | --- |
| Description | show list voucher |
| Connect From | admin-page |
| Connect To | voucher-update page |
| Validation | N/A |
| Initial Values | Voucher List |
| Events | - when select "Show" option, show result with amount record selected  - when input text field search, show result voucher match keyword - when click "update" button, go to voucher-update page - when click "delete" button, delete voucher in this row |

### Voucher Add



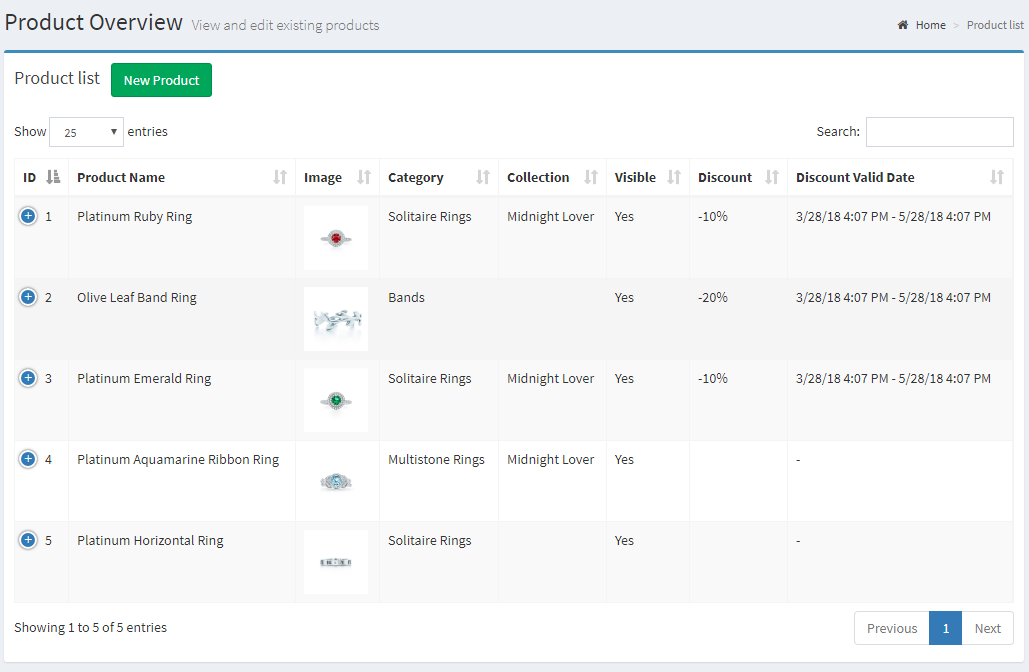
|  |  |
| --- | --- |
| Description | Insert new voucher |
| Connect From | admin-page |
| Connect To | voucher-list page |
| Validation | - All fields: Not blank - Voucher ID is unique and must be follow format VOUxxx  - Discount valid date: date valid from must be before date valid to  - Description must be more than 10 character - Discount value must be more than 5% and less than 75% - Quantity must be more than 1 |
| Initial Values |  |
| Events | - when click "Save" button, save new voucher to database and go to voucher-list page - when click "Cancel" button, reset all fields to blank |

### Voucher Update



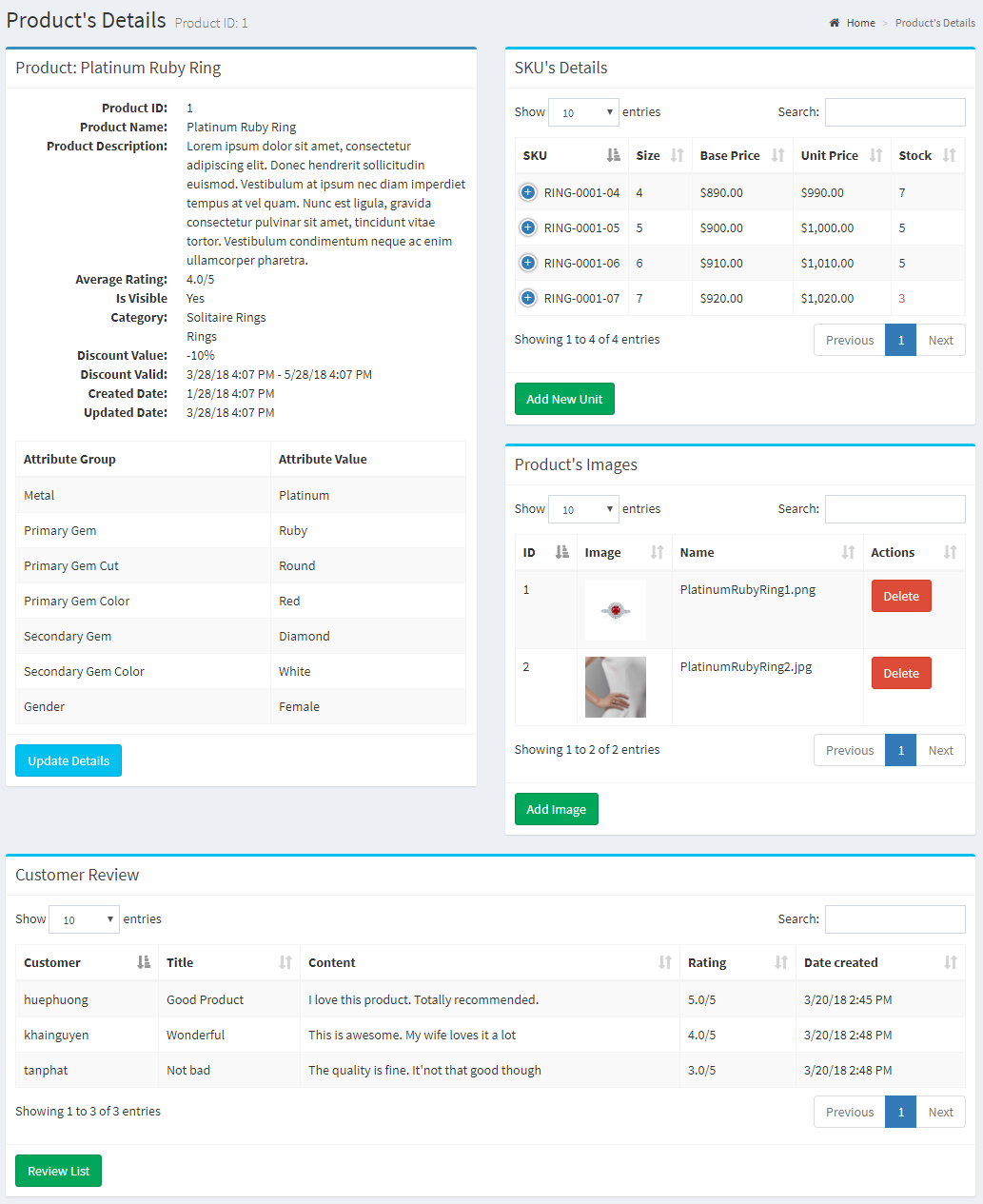
|  |  |
| --- | --- |
| Description | update voucher |
| Connect From | voucher-list page |
| Connect To |  |
| Validation | - Discount valid date: date valid from must be before date valid to  - Description must be more than 10 character - Discount value must be more than 5% and less than 75% - Quantity must be more than 1 |
| Initial Values | All fields of voucher is edited |
| Events | - when click "Save" button, edit and save this voucher to database and go to voucher-list page - when click "Cancel" button, reset all fields to blank |

### Product List



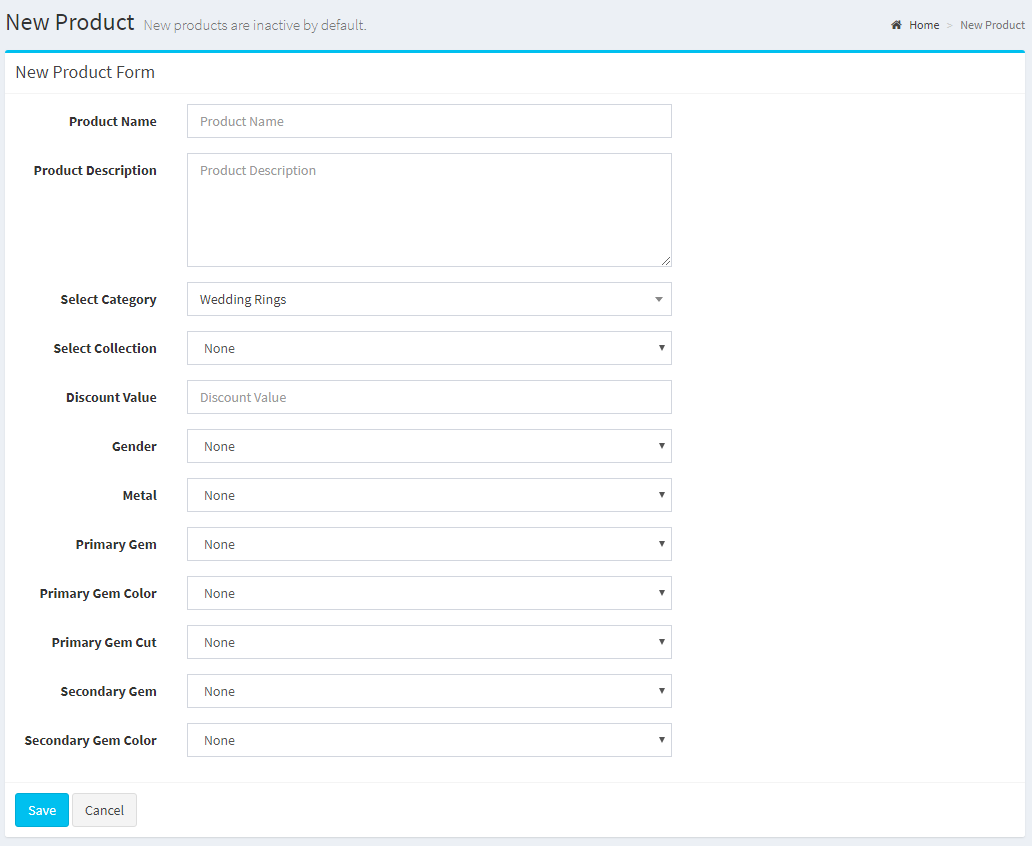
|  |  |
| --- | --- |
| Description | Product list |
| Connect From | Dashboard |
| Connect To | Product detail, Product update |
| Validation | N/A |
| Initial Values | Product List |
| Events | - When click “Detail”, redirect to Product Detail  - When click “Update Detail”, redirect to Product update |

### Product Details



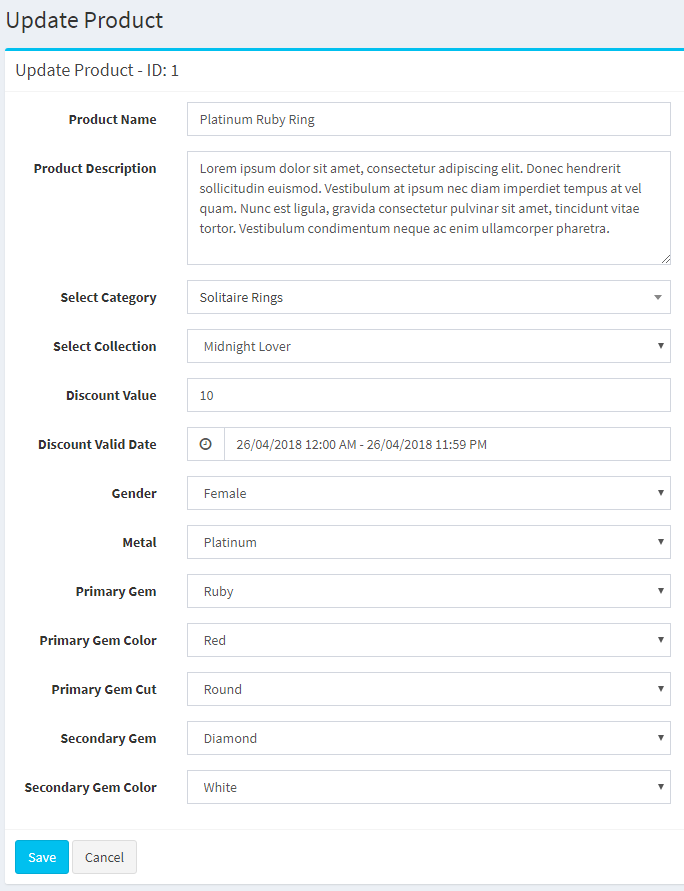
|  |  |
| --- | --- |
| Description | Product Detail |
| Connect From | Product Page |
| Connect To | Product Update, Sku list, Image list, Review list |
| Validation | N/A |
| Initial Values | Product Details, Sku list, Image list, Review list |
| Events | - when click “Update Detail”, redirect to Update form  - When click Delete on Sku List, delete Sku if acceptable  - When click Delete on Image List, delete Image if acceptable  - When click “Add New SKU”, redirect to add SKU form  - When click “Add new Image”, redirect to add Image form |

### Product Add



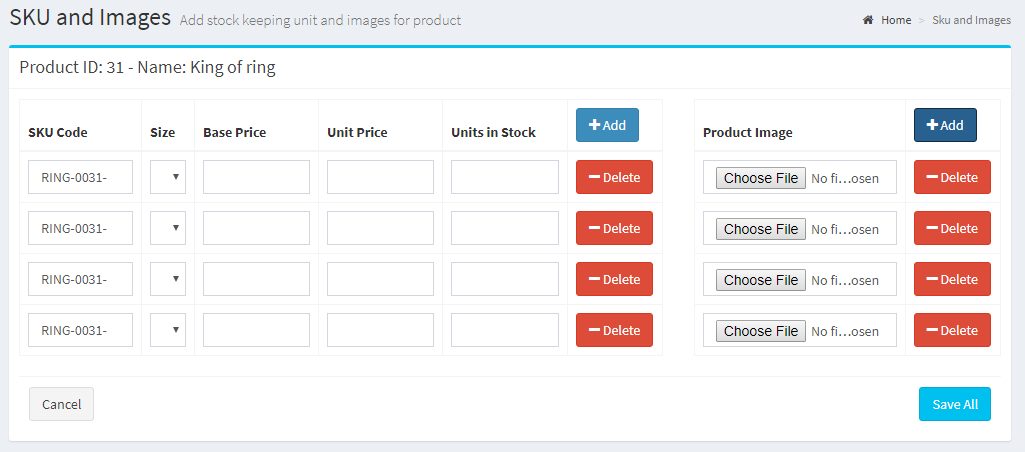
|  |  |
| --- | --- |
| Description | Product Update |
| Connect From | Product Detail, Product Page |
| Connect To | Product Detail |
| Validation | - All fields: not blank  - Category: not null  - Discount value between: 5% - 75% |
| Initial Values | N/A |
| Events | - When click “Save”, add product in database  - When click “Cancel”, reset all fields |

### Product Update



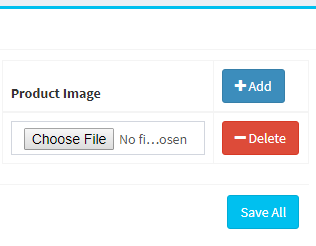
|  |  |
| --- | --- |
| Description | Product Update |
| Connect From | Product Detail, Product Page |
| Connect To | Product Detail |
| Validation | - All fields: not blank  - Category: not null  - Discount value between: 5% - 75% |
| Initial Values | - Product Detail |
| Events | - When click “Save”, update product in database  - When click “Cancel”, reset all fields |

### Product-SKU-Image Add



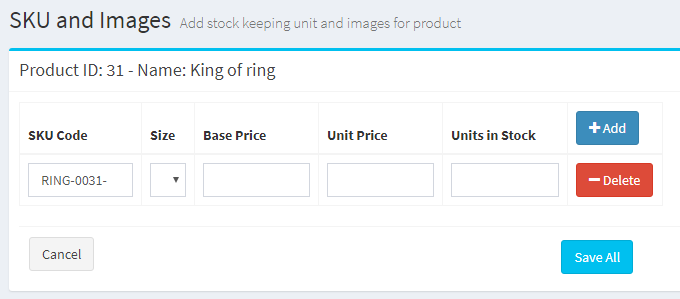
|  |  |
| --- | --- |
| Description | Add SKU, Images for product |
| Connect From | Product: Add New |
| Connect To | Product Detail |
| Validation | - All fields: not blank  - Must have at least one SKU and one Image  - SKU ID must be unique  - Unit In Stock must be greater than 0  - Unit Price must be greater than Base Price  - Base Price must be greater than $10 |
| Initial Values | - SKU ID: generated from Category and Product ID |
| Events | - When click “Save”, add SKU and Images to database  - When click “Cancel”, reset all fields  - When click “Add”, add another form group  - When click “Delete”, remove form group |

### Product-Image Update



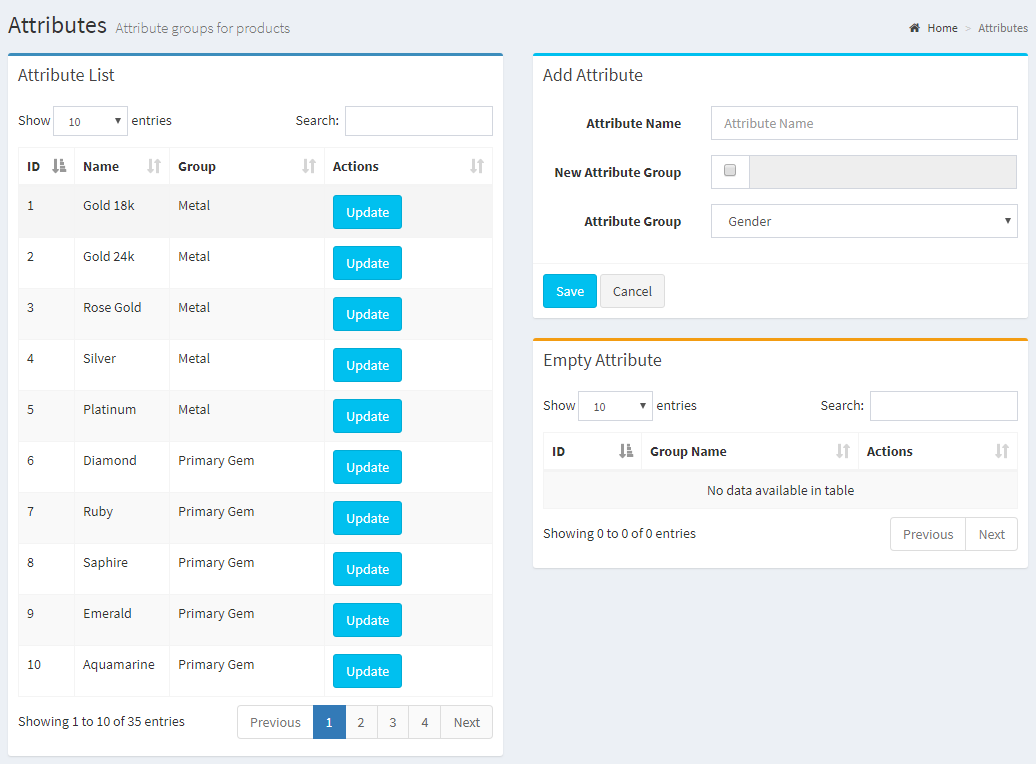
|  |  |
| --- | --- |
| Description | Update Image for Product |
| Connect From | Product Detail |
| Connect To | Product Detail |
| Validation | Image field must not be blank |
| Initial Values | N/A |
| Events | - When click “Save”, add Image in database  - When click “Cancel”, reset all fields |

### Product-SKU Update



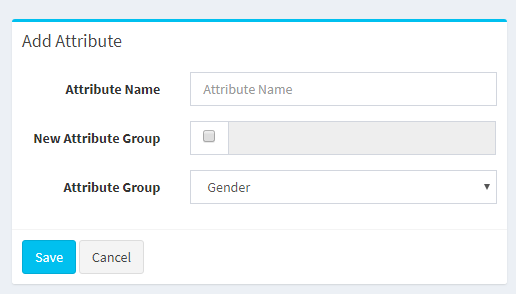
|  |  |
| --- | --- |
| Description | Add SKU, Images for product |
| Connect From | Product: Add New |
| Connect To | Update SKU |
| Validation | - All fields: not blank  - SKU ID must be unique  - Unit In Stock must be greater than 0  - Unit Price must be greater than Base Price  - Base Price must be greater than $10 |
| Initial Values | - SKU ID: generated from Category and Product ID |
| Events | - When click “Save”, add SKU and Images to database  - When click “Cancel”, reset all fields  - When click “Add”, add another form group  - When click “Delete”, remove form group |

### Attribute Overview



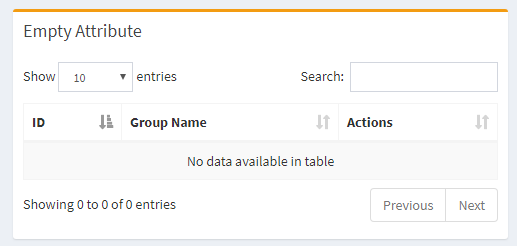
|  |  |
| --- | --- |
| Description | Attribute Overview |
| Connect From | Dashboard |
| Connect To | Attribute: Add, Update |
| Validation | N/A |
| Initial Values | Attribute List |
| Events | - When there are empty Attribute Group, they will show up in Empty Group List  - When the form is “Add”, admin can add New Attribute  - When click on Update, admin can update respective Attribute |

### Attribute Add



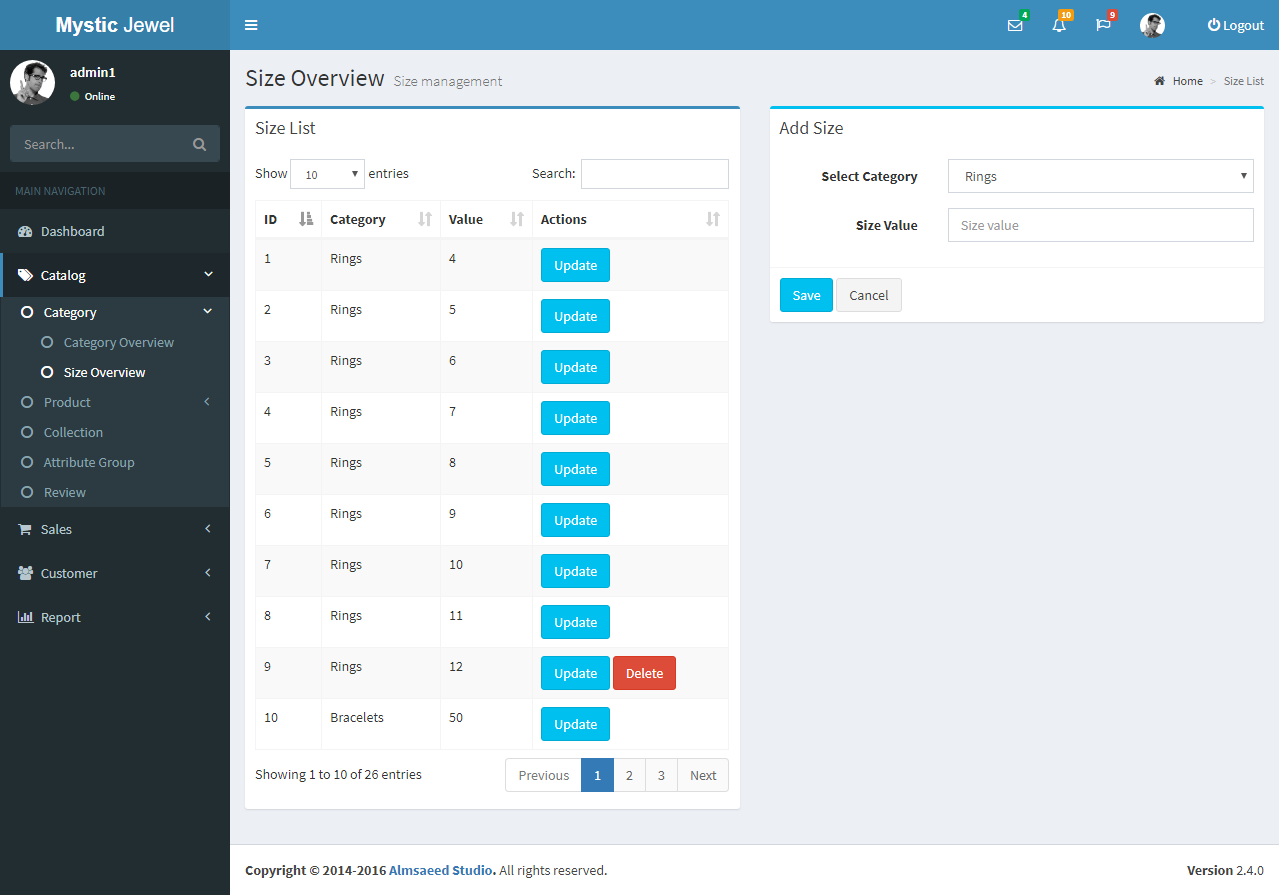
|  |  |
| --- | --- |
| Description | Add New Attribute |
| Connect From | Attribute Overview |
| Connect To | Attribute Overview |
| Validation | - All fields: not blank  - Attribute Group name is unique |
| Initial Values | - Attribute Group list |
| Events | - When click “Save”, add Attribute (and Group) to database  - When click “Cancel”, reset all fields  - When check New Attribute Group: create new Attribute Group |

### Attribute Empty



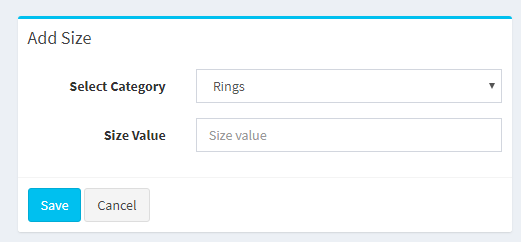
|  |  |
| --- | --- |
| Description | Empty Attribute Group List |
| Connect From | Attribute Overview |
| Connect To | N/A |
| Validation | N/A |
| Initial Values | Empty Attribute Group List |
| Events | - When click “Delete”, group will be deleted from database |

### Size Overview



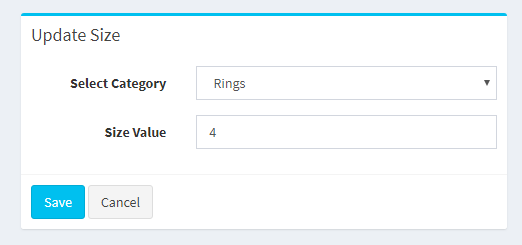
|  |  |
| --- | --- |
| Description | Size Overview |
| Connect From | Dashboard |
| Connect To | Size: Add, Update |
| Validation | N/A |
| Initial Values | Size List |
| Events | - When click “Update”, redirect to update form  - When click “Delete”, size will be deleted from database |

### Size Add



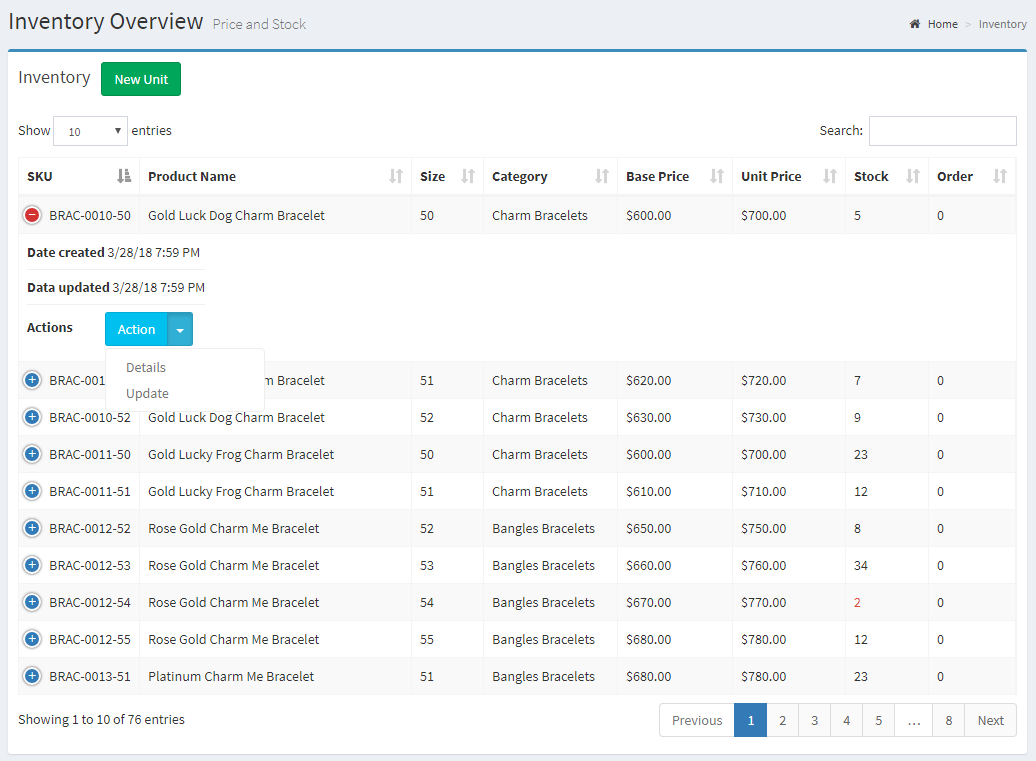
|  |  |
| --- | --- |
| Description | Size Add |
| Connect From | Size Overview |
| Connect To | Size Overview |
| Validation | - Size is Unique within the same Category |
| Initial Values | - Main Category List |
| Events | - When click “Save”, add Size to database  - When click “Cancel”, reset all fields |

### Size Update



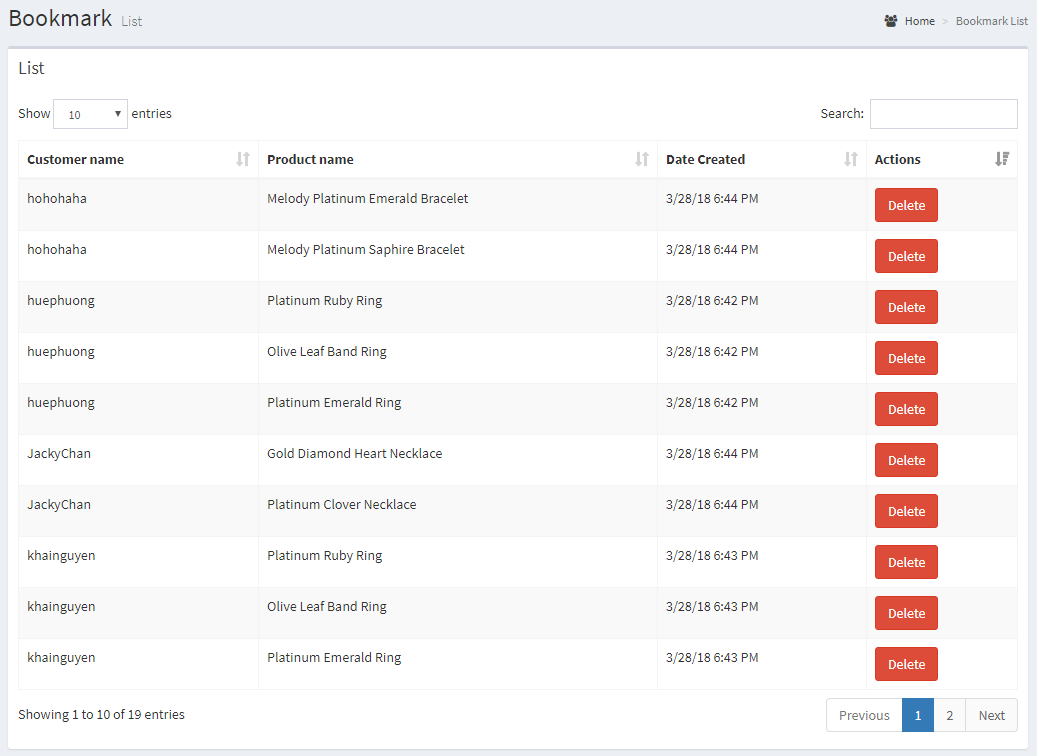
|  |  |
| --- | --- |
| Description | Size Update |
| Connect From | Size Overview |
| Connect To | Size Overview |
| Validation | - Size is Unique within the same Category |
| Initial Values | - Main Category List |
| Events | - When click “Save”, update Size to database  - When click “Cancel”, reset all fields |

### Inventory



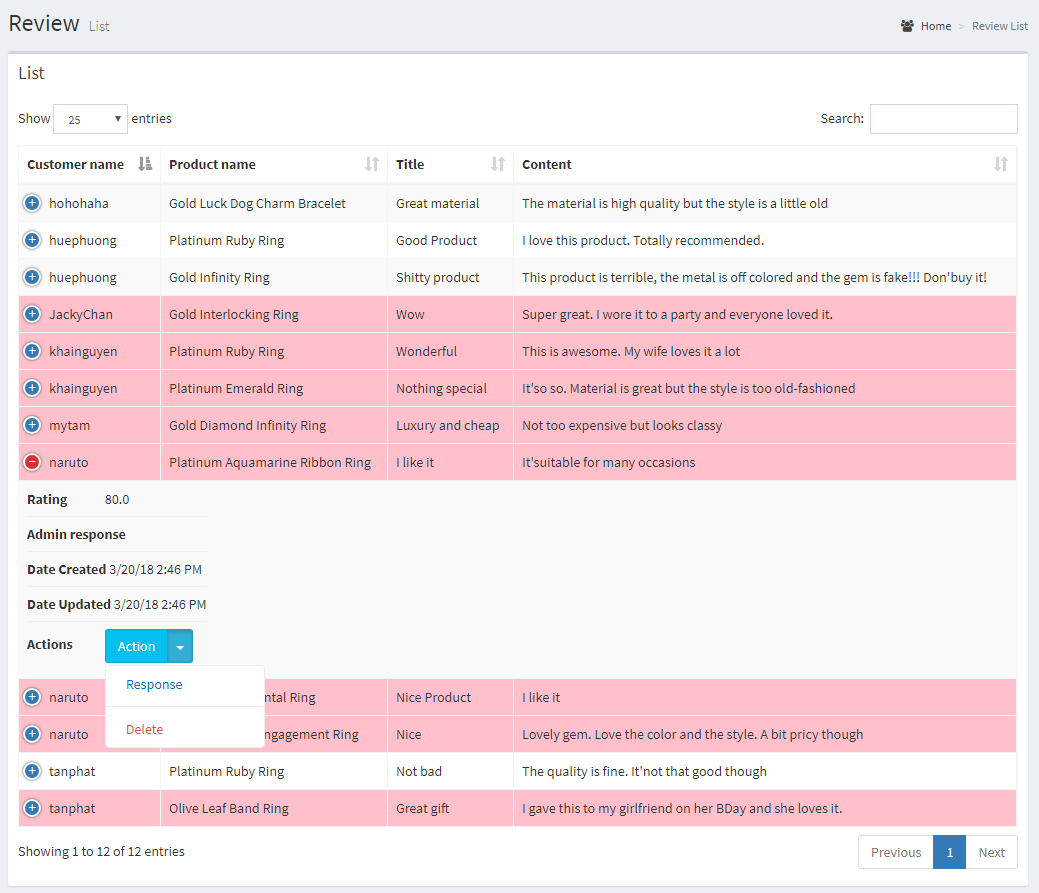
|  |  |
| --- | --- |
| Description | View Inventory (SKU list) |
| Connect From | Dashboard |
| Connect To | N/A |
| Validation | N/A |
| Initial Values | SKU List |
| Events | - When click Update, redirect to update form  - When click New Unit, redirect to Add New SKU form |

### Bookmark list



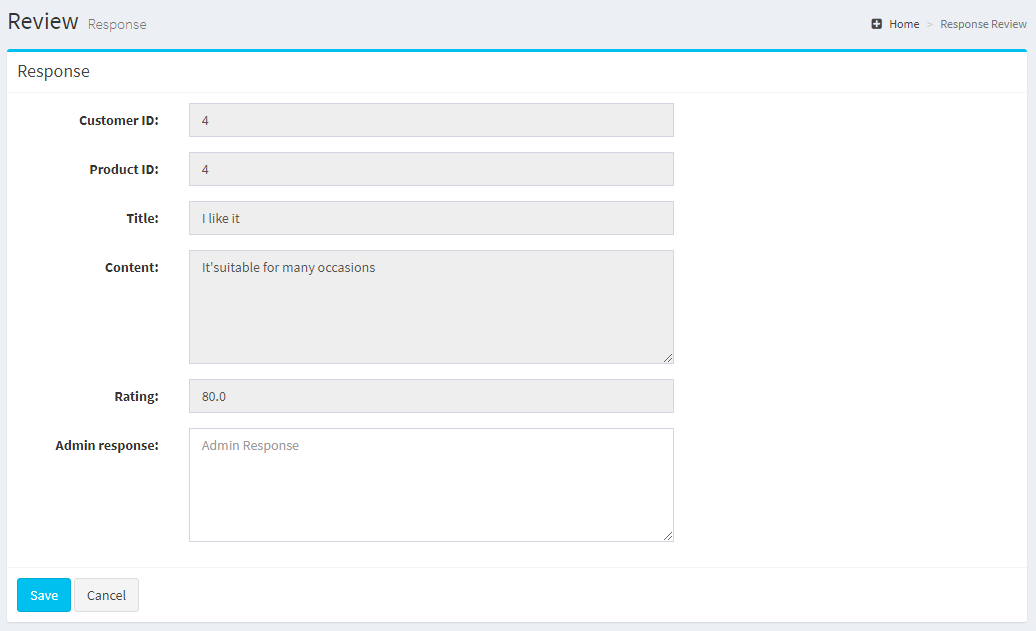
|  |  |
| --- | --- |
| Description | show bookmark list of customers |
| Connect From | Dashboard |
| Connect To | N/A |
| Validation | N/A |
| Initial Values | Bookmark List |
| Events | - when select "Show" option, show result with amount record selected  - when input text field search, show result bookmark match keyword |

### Review List



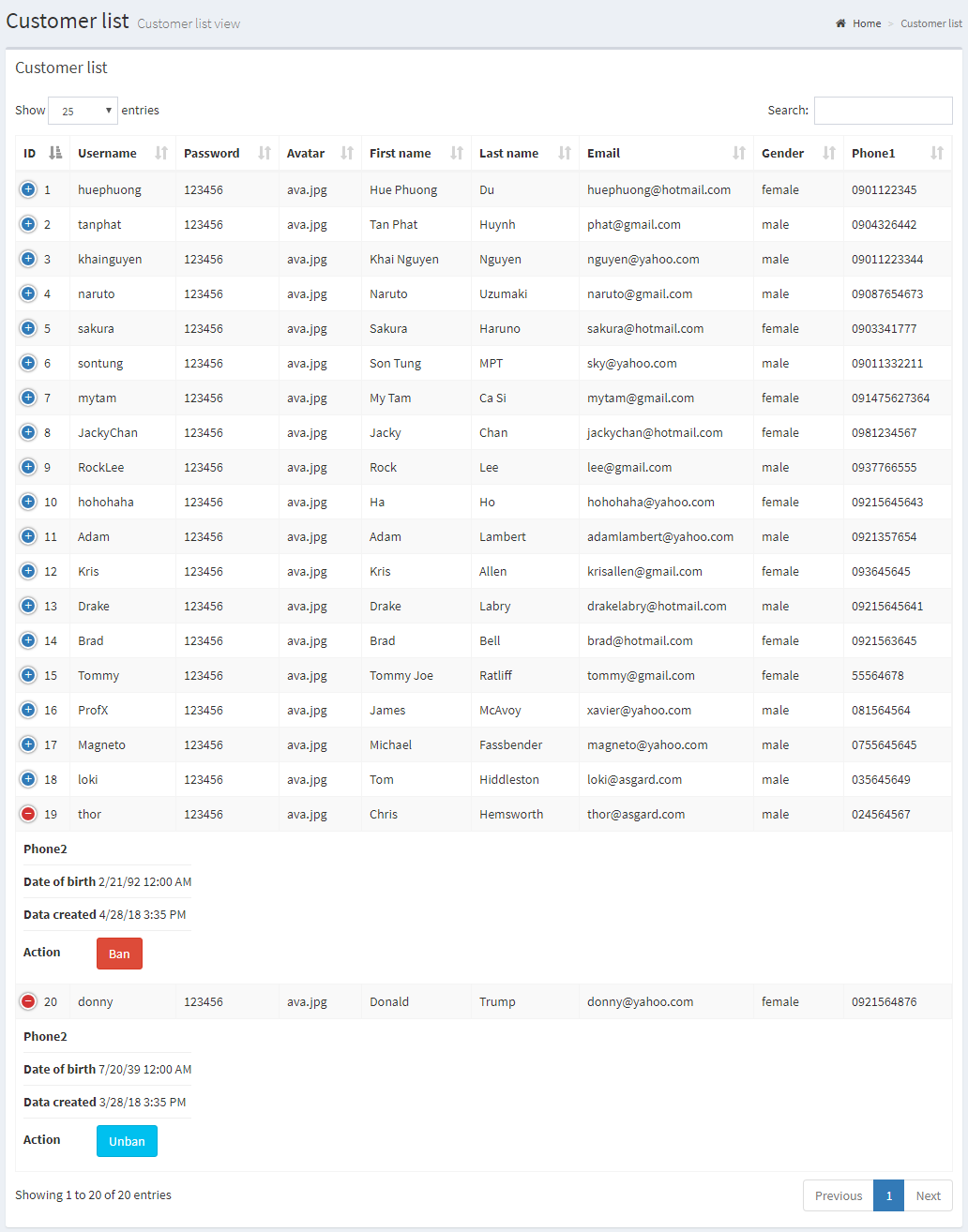
|  |  |
| --- | --- |
| Description | show list review of customers |
| Connect From | admin-page |
| Connect To | review-response page |
| Validation | If review does not has admin response, background color of that review row is pink |
| Initial Values | - Review List |
| Events | - when click "Response" button, go to review-response page to response that review - when click "Delete" button, delete review is selected |

### Review response



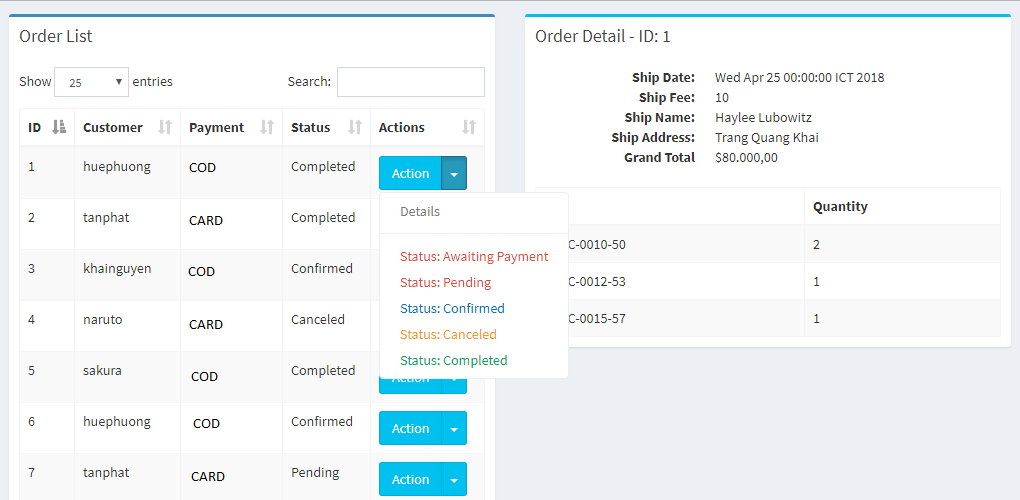
|  |  |
| --- | --- |
| Description | Give response to customer's review |
| Connect From | review-list page |
| Connect To | review-list page |
| Validation | Admin response: not blank and must be more than 10 characters |
| Initial Values | - Customer’s review details |
| Events | - when click "Save" button, give response to customer and save to database and go to review-list page - when click "Cancel" button, reset all fields to blank |

### Customer Overview



|  |  |
| --- | --- |
| Description | Show Customer List |
| Connect From | Dashboard |
| Connect To | Ban Customer |
| Validation | N/A |
| Initial Values | Customer List |
| Events | - when select "Show" option, show result with amount record selected  - when input text field search, show result customer match keyword  - when click "Ban" button, Ban customer's behavior - when click "Unban" button, Unban customer's behavior |

### Order Overview



|  |  |
| --- | --- |
| Description | Show Order Overview |
| Connect From | Dashboard |
| Connect To | Order Detail |
| Validation | N/A |
| Initial Values | Order List |
| Events | - When click “Details”, show Order detail  - When click “Status”, change order’s status in database. |

REVIEW 3 – TASK SHEET

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Contents** | **Member** | **Activity Plan** | **Preparation** | |
| **Start Date** | **Actual Days** | **Status** |
| **Client** | | Du Hue Phuong | 10/04/2018 | 15 | Completed |
| 1 | Homepage |
| 2 | Product: new, hot, on sale |
| 3 | Product: view by Category, Collection |
| 4 | Product’s detail |
| 5 | Recommend products |
| 6 | Compare products |
| 7 | Place orders |
| 8 | Payment |
| **Admin** | |
| 9 | Category: view, add, update, delete |
| 10 | Collection: view, add, update, delete |
| 11 | Size: view, add, update, delete |
| 12 | Attribute/AttributeGroup: view, add, update, delete |
| 13 | Product: view, add, update |
| 14 | Sku: view, add, update, delete |
| 15 | ProductImage: view, add, update, delete |
| 16 | Order: view, update status |
| 17 | Report: Product’s revenue |
|  | | | | | |
| **Client** | | Nguyen Khai Nguyen | 10/04/2018 | 15 | Completed |
| 1 | GUI Design |
| 2 | Login/Logout/Session |
| 3 | Customer register |
| 4 | Forgot password/Reset password |
| 5 | Customer’s profile: update |
| 6 | Customer’s address book: add, update, delete |
| 7 | Cart: add, update, remove |
| 8 | Send mail to customer: share wish list, send password reset token |
| 9 | Calculate shipping fee with Google API |
| **Admin** | |
| 10 | GUI Design |
| 11 | Login/Logout/Session |
| 12 | Admin permission |
| 13 | AdminAccount: view, add, update |
| 14 | Customer: view, ban (update) |
| 15 | CustomerAddress: view |
| 16 | Report: Customer |
|  | | | | | |
| **Client** | | Huynh Tan Phat | 10/04/2018 | 15 | Completed |
| 1 | Wish List: view, add, delete |
| 2 | Review/Rating on product |
| 3 | Search products by relative name |
| 4 | Filter products by: collection, category, attributes, price |
| 5 | Sort by: newest/oldest, lowest/highest price |
| 6 | Validate product discount dates |
| 7 | Validate voucher |
| **Admin** | |
| 8 | Review: view, response (update), delete |
| 9 | Bookmark: view |
| 10 | Voucher: view, add, update, delete |
| 11 | Report: Order |

Plan prepared by: Group 2 April 25th, 2018  
 **Group Leader Faculty**

**Du Hue Phuong Mr. Ngo Phuoc Nguyen**