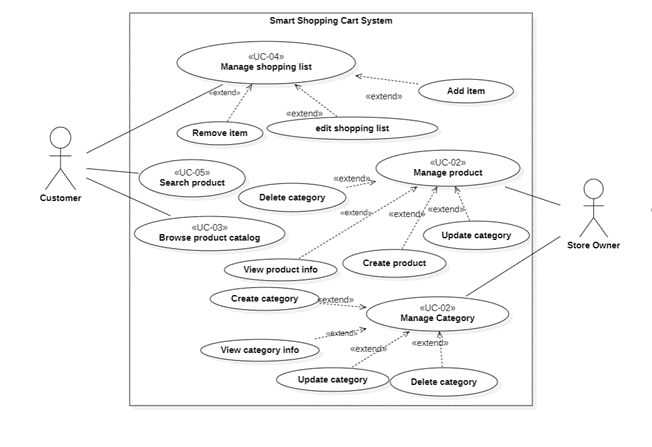
# Sprint Backlog:

|  |  |  |
| --- | --- | --- |
| Req-Id | Title | Type |
| Req-01 | The system must allow store owner to add products | Functional |
| Req-02 | The system must allow store owner to edit products info | Functional |
| Req-03 | The system must allow store owner to remove products | Functional |
| Req-04 | The system must allow store owner to view products | Functional |
| Req-05 | The system must allow store owner to view information about a specific product | Functional |
| Req-06 | The system shall allow store owner to search about product by name | Functional |
| Req-07 | The system shall allow store owner to filter products by category, price, name | Functional |
| Req-08 | The system shall allow store owner to filter categories by name | Functional |
| Req-09 | The system shall support pagination system | Functional |
| Req-10 | The system must allow store owner to add category | Functional |
| Req-11 | The system must allow store owner to edit category | Functional |
| Req-12 | The system must allow store owner to remove category | Functional |
| Req-13 | The system must allow store owner to view categories | Functional |
| Req-14 | The system must allow store owner to view information about a specific category | Functional |
| Req-15 | The system must allow customer to browse products in shop | Functional |
| Req-16 | The system must allow customer to view product info | Functional |
| Req-17 | The system must allow customer to add product to shopping list | Functional |
| Req-18 | The system must allow customer to remove product from shopping list | Functional |
| Req-19 | The system must allow customer to update quantity for any product in list | Functional |
| Req-20 | The system shall calculate the total price of each items in the shopping list | Functional |
| Req-21 | The system shall calculate the total cost of shopping list | Functional |
| Req-22 | The system shall allow customer track picked-up items to prevent forgetting products | Functional |
| Req-23 | The system must save shopping list for customer automatically | Functional |
| Req-24 | The system shall allow customer to search about product by name, description | Functional |
| Req-25 | The system shall allow customer to view his shopping list | Functional |
| Req-26 | The system must handle multiple stores with unique products, categories, and layouts. | Functional |

# Use Case for Sprint:



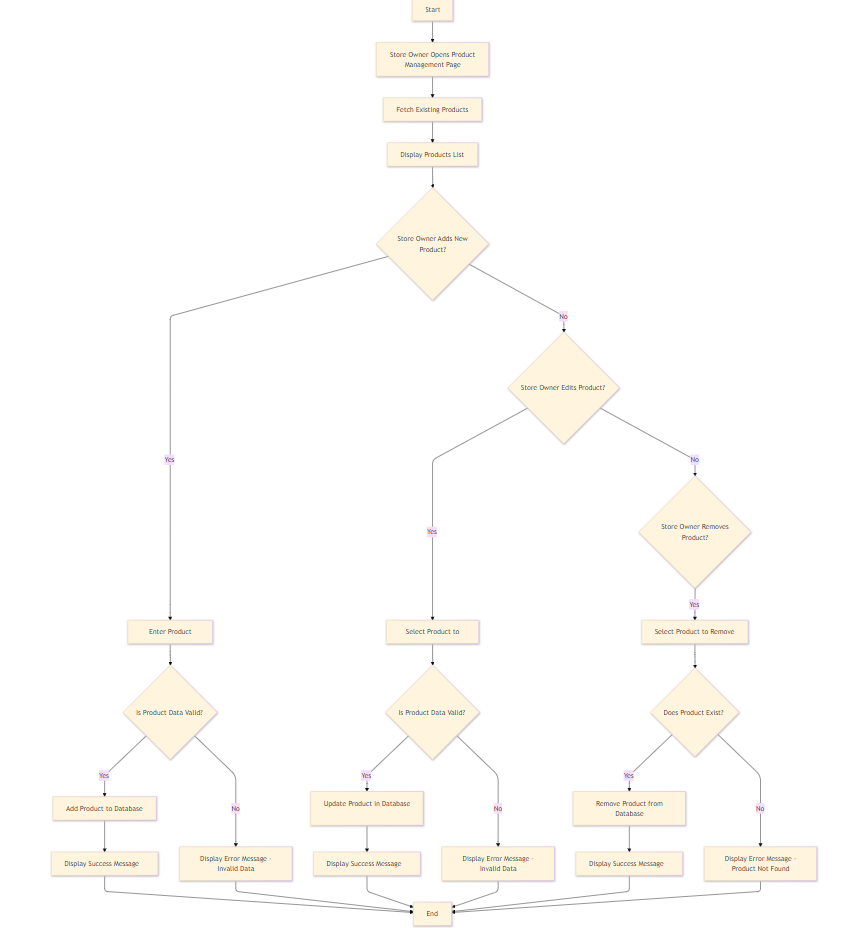
# Use Case Specification

1. Product Management

|  |  |
| --- | --- |
| Use case Id | UC-01 |
| Use case title | Product Management |
| Description | This use case describes how a store owner can manage the products within their store by adding, editing, and deleting products in the system. |
| Actors | Store Owner |
| Precondition | The store owner must be logged into the system. |
| The flow of events | 1. The store owner navigates to the product management page. 2. The system displays a form to add new products and a list of existing products. 3. The store owner fills in the product details and submits the form to add a new product. 4. The system validates the input data. 5. The system saves the new product in the database and confirms success to the store owner. 6. If the store owner chooses to update a product, they click on the product in the list, update the relevant fields, and submit the changes. 7. The system validates the updates and applies the changes in the database. 8. If the store owner chooses to delete a product, they click the delete button. 9. The system confirms deletion and removes the product from the database. |
| Alternative scenarios | **A1: Product Already Exists**   1. The system detects that the product name already exists. 2. The system prompts the store owner to either modify the product details or use a different product name.   **A2: Optional Fields Omitted**   1. The system allows the store owner to proceed without filling in optional fields. 2. The product is added with default or blank values for those fields. |
| Error scenarios | **E1: Invalid Data**   1. The system detects invalid input and displays an error message. 2. The store owner is prompted to correct the data and resubmit.   **E2: Database Error**   1. The system displays an error message indicating the failure. 2. The store owner is advised to retry the action or contact support if the issue persists. |
| Postcondition | The product is successfully added, updated, or deleted from the catalog. |

## Sequence diagram

## Activity diagram



1. Category Management

|  |  |
| --- | --- |
| Use case Id | UC-02 |
| Use case title | Category Management |
| Description | This use case describes how a store owner can manage the categories within their store by adding, editing, and deleting categories in the system. |
| Actors | Store Owner |
| Precondition | The store owner must be logged into the system. |
| The flow of events | * + 1. The store owner navigates to the category management page.     2. The system displays a form to add new categories and a list of existing categories.  1. The store owner fills in the category details and submits the form to add a new category. 2. The system validates the input data. 3. The system saves the new category in the database and confirms success to the store owner. 4. If the store owner chooses to update a category, they click on the category in the list, update the relevant fields, and submit the changes. 5. The system validates the updates and applies the changes in the database. 6. If the store owner chooses to delete a category, they click the delete button. 7. The system confirms deletion and removes the category from the database. |
| Alternative scenarios | **A1: Product Already Exists**   1. The system detects that the category name already exists. 2. The system prompts the store owner to either modify the category details or use a different category name. |
| Error scenarios | **E1: Invalid Data**   * + 1. The system detects invalid input and displays an error message.     2. The store owner is prompted to correct the data and resubmit.   **E2: Database Error**   * + 1. The system displays an error message indicating the failure.     2. The store owner is advised to retry the action or contact support if the issue persists. |
| Postcondition | The category is successfully added, updated, or deleted from the catalog. |

## Sequence diagram

## Activity diagram

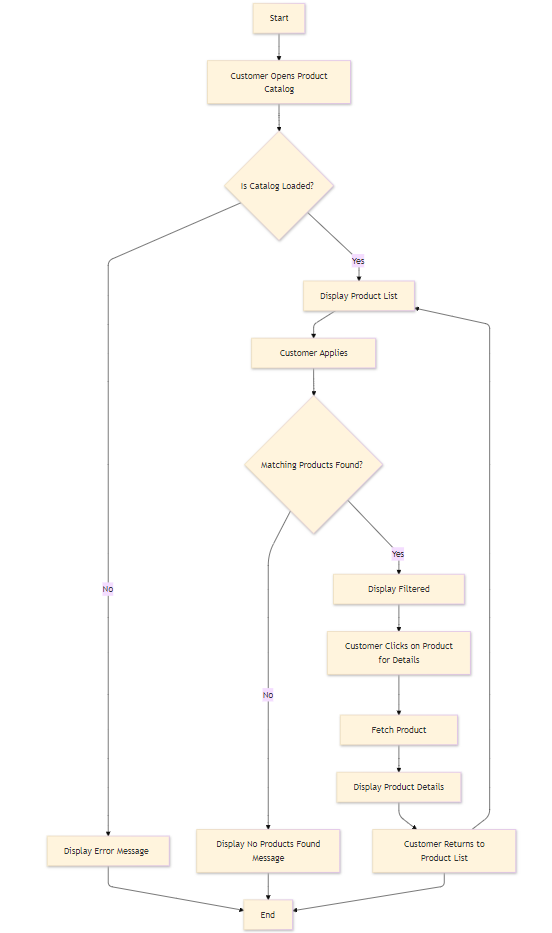
# Use Case Specification

## Product Catalog

|  |  |
| --- | --- |
| Use case Id | UC-03 |
| Use case title | Product Catalog |
| Description | This use case describes how customers can browse through available products and view detailed information about specific products in the store. |
| Actors | Customer |
| Precondition | The customer is on the product catalog page. |
| The flow of events | 1. The customer navigates to the product catalog page. 2. The system displays all available products in a grid or list format. 3. The customer scrolls through the list and clicks on a product to view more details. 4. The system displays detailed information about the selected product. 5. The customer can return to the catalog list to continue browsing. |
| Alternative scenarios | A1: Filter Products   1. The customer selects the filter criteria. 2. The system updates the product list to show only products matching the filter criteria. |
| Error scenarios | E1: System Failure   1. The system displays an error message indicating the catalog is unavailable. 2. The customer is advised to try again later. |
| Postcondition | The customer can successfully browse the products and view detailed information about any product. |

## Sequence diagram

## Activity diagram

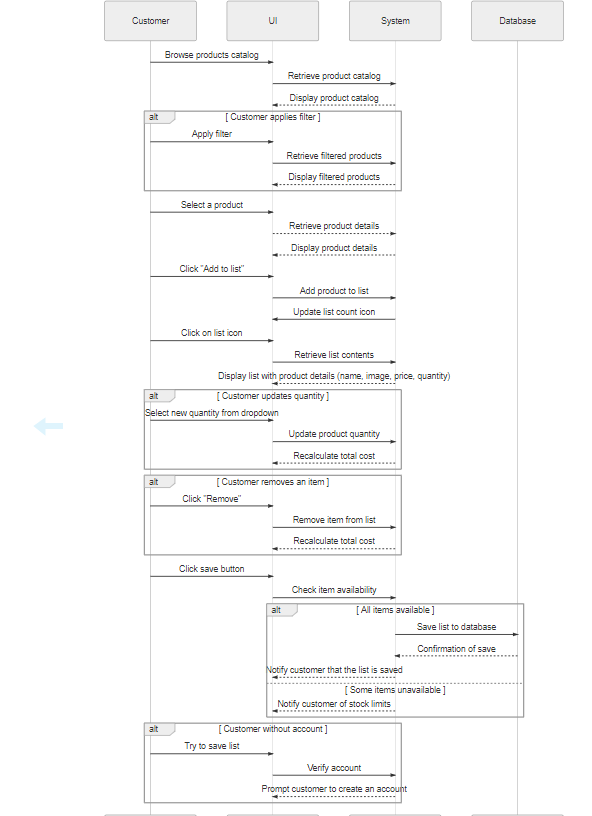


# Use Case Specification

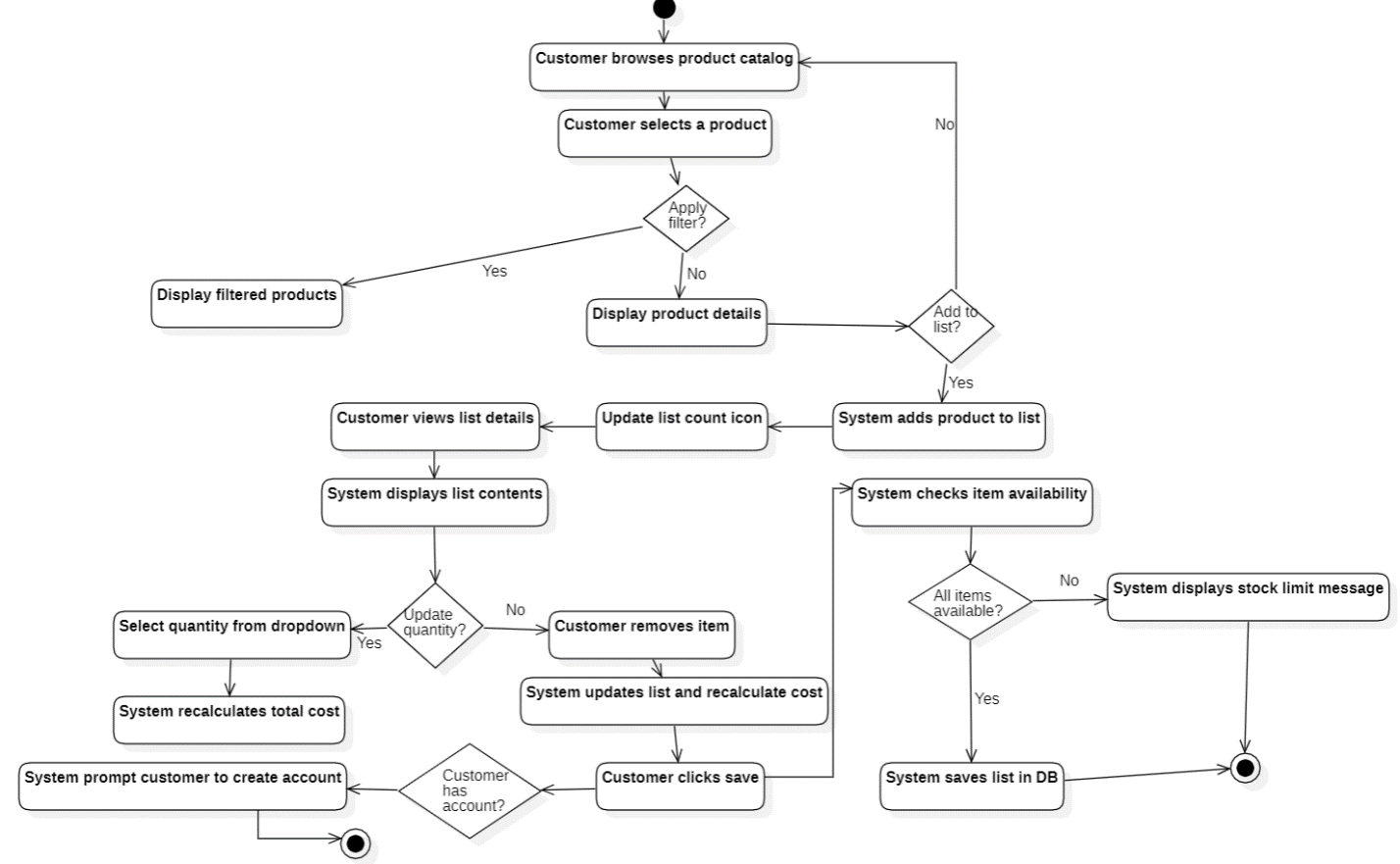
## Shopping List Management

|  |  |
| --- | --- |
| Use case Id | UC-03 |
| Use case title | Shopping List Management |
| Description | This use case allows customers to manage their shopping list. The customer can add Items to the list, view the list contents, update item quantities, remove items. The list should dynamically update in response to customer actions. |
| Actors | Customer |
| Precondition | * Customer has an account * Customer must be logged in |
| The flow of events | 1. The customer browses products catalog page and selects a product to view. 2. If the customer wants to “Add to list”: 3. The customer clicks the "Add to list" button. 4. The system adds the selected product to the customer's list and updates the list count icon in the UI. 5. If The customer wants to view list details: 6. The customer clicks on the list icon to view their list. 7. The system displays the list contents with product details (name, image, price, quantity, etc.). 8. If the customer wants to update list: 9. The customer updates the quantity of an Item by either: 10. Selecting a new quantity from a dropdown. 11. Typing the quantity directly. 12. The system recalculates the total cost based on the updated quantities. 13. The customer removes an item by clicking the "Remove" or "Delete" button. 14. The system updates the list and recalculates the total cost. 15. The customer clicks on save button 16. The system checks items availability 17. If all items in list are available: 18. The system saves the list in database |
| Alternative scenarios | A1: start at step 1 in main scenario if customer applies filter to narrow down the product list:   1. The system refreshes the catalog page to display filtered results.   And the flow will return to step 1 in main flow |
| Exception scenarios | E1: If the customer tries to save list and he doesn’t have account:   1. The system notifies customer to create an account and use case failed   E2: If the product the customer selects is unavailable (out of stock or discontinued):   1. The system displays an out-of-stock message.   E3: If the product cannot be added to the list due to stock limitations (e.g., only a few items are left).   1. The system displays a message informing the customer of stock limits and adjusts the quantity accordingly. |
| Postcondition | * The list reflects the latest changes made by the customer. * The list has been save in database |

## Sequence diagram



## Activity diagram

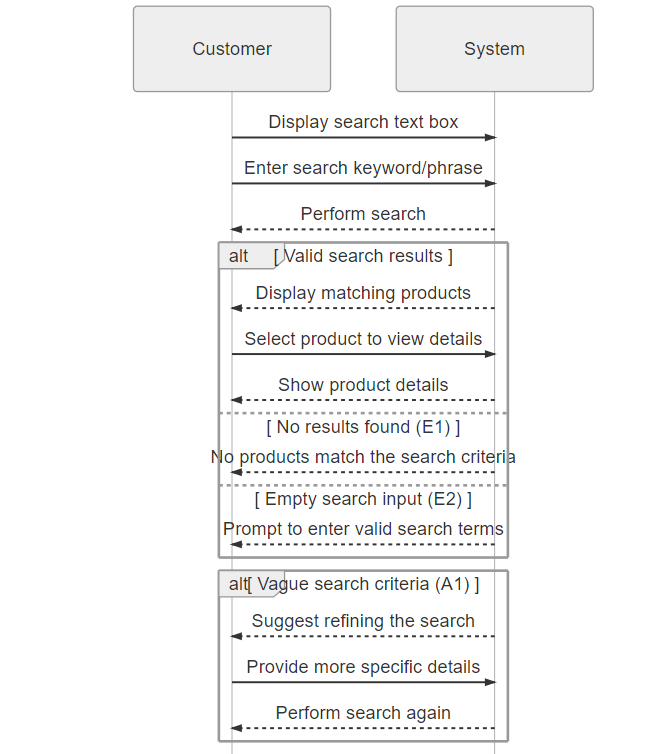


# Use Case Specification

## Product Search

|  |  |
| --- | --- |
| Use case Id | UC-04 |
| Use case title | Search Product |
| Description | This use case allows customers to search for products within the store |
| Actors | Customer |
| Precondition | The customer must be logged into the system |
| The flow of events | 1. The system displays a search text box for entering search criteria. 2. The customer enters a keyword or phrase in the search text box (e.g., product name, category, or other details) 3. The system performs search using the input data, matching it against product names, categories, or other relevant fields. 4. The system displays a list of products that match the search criteria 5. The customer selects a product from the search results to view more details. |
| Alternative scenarios | A1: start at step 2 if customer entered vague search criteria:   1. The system suggests refining the search 2. The customer presents more specific details   And flow will return to step 3 in the main flow |
| Exception scenarios | E1: If no results found for the search input:   1. The system informs the customer that no products match the search criteria   E2: If search input left empty:   1. The system prompts the customer to enter valid search terms before proceedings with the search |
| Postcondition | The customer is presented with a list of products that match the search criteria. |

## Sequence diagram



## Activity diagram

