YALA *A Shopping List App*

Team Members: Evan Berryman (Leader), Sam Hobbs, Jake Allen

The problem of	forgetting items at the store
affects	people who shop
the impact of which is	wasted time
a successful solution	a list management application

Problem Statement: We need an automated way to keep track of lists and item inventory.

We will create a system that allows us to do this.

Our Solution

System Features / Functional Requirements

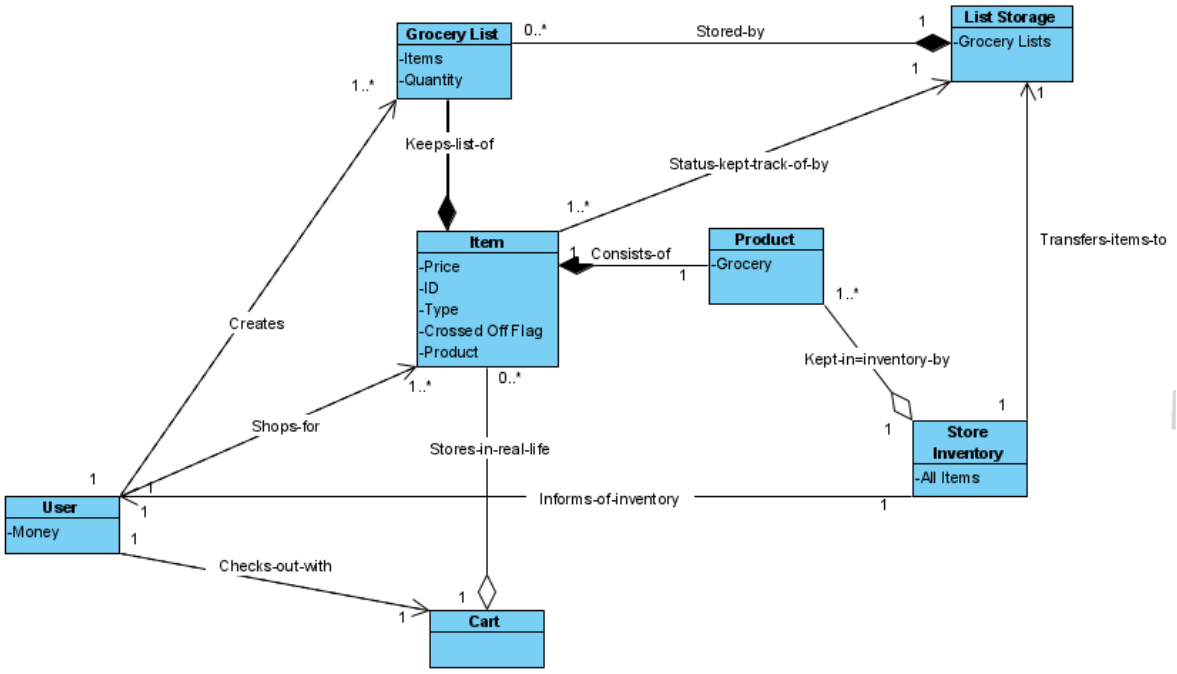
- Create List
- Search Items of Store Directories
- Add Item
- Crossing Off Item
- Uncross Item
- Removing Item
- Copy List
- Delete List
- Copy/Transfer an Item

File Management and Ticketing System

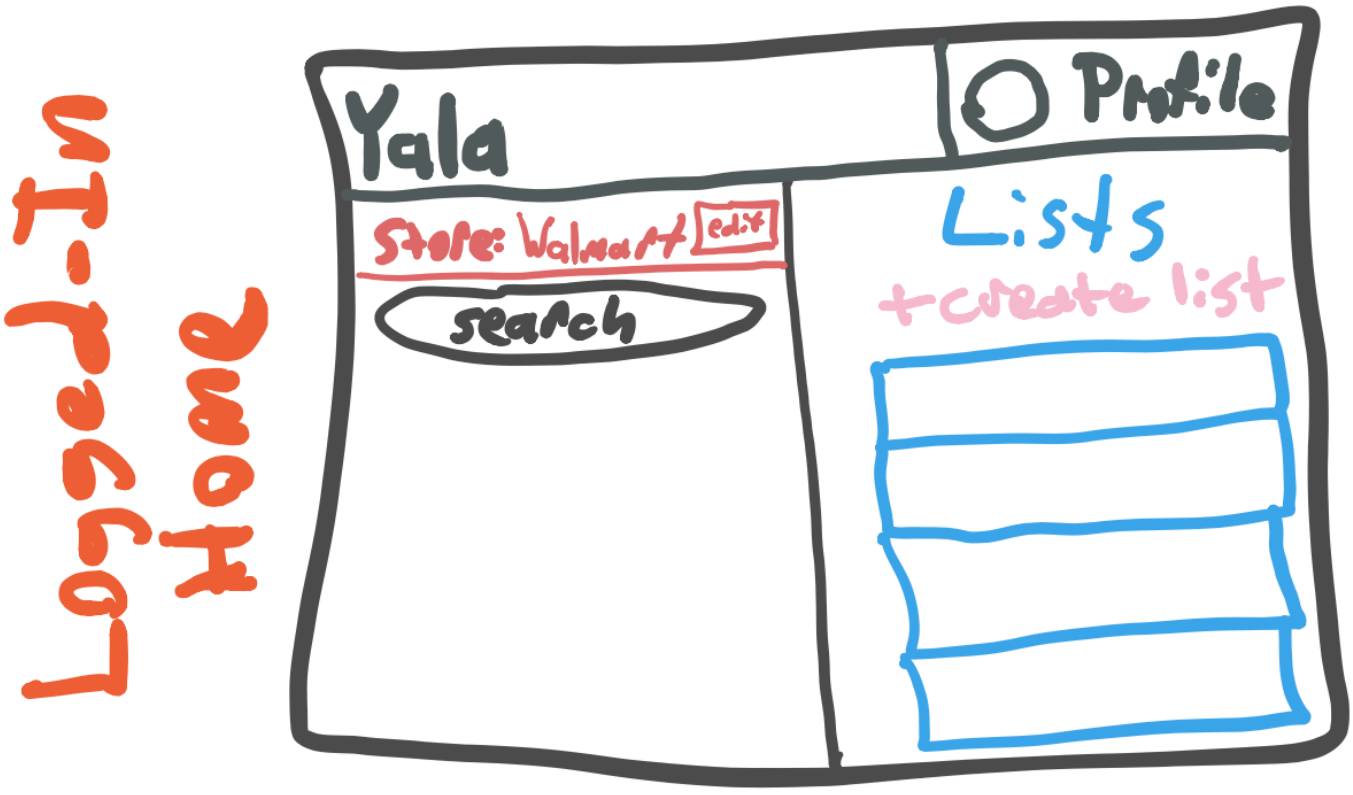
https://github.com/jake-allen/YALA

https://trello.com/b/8JTgqSMS/yala

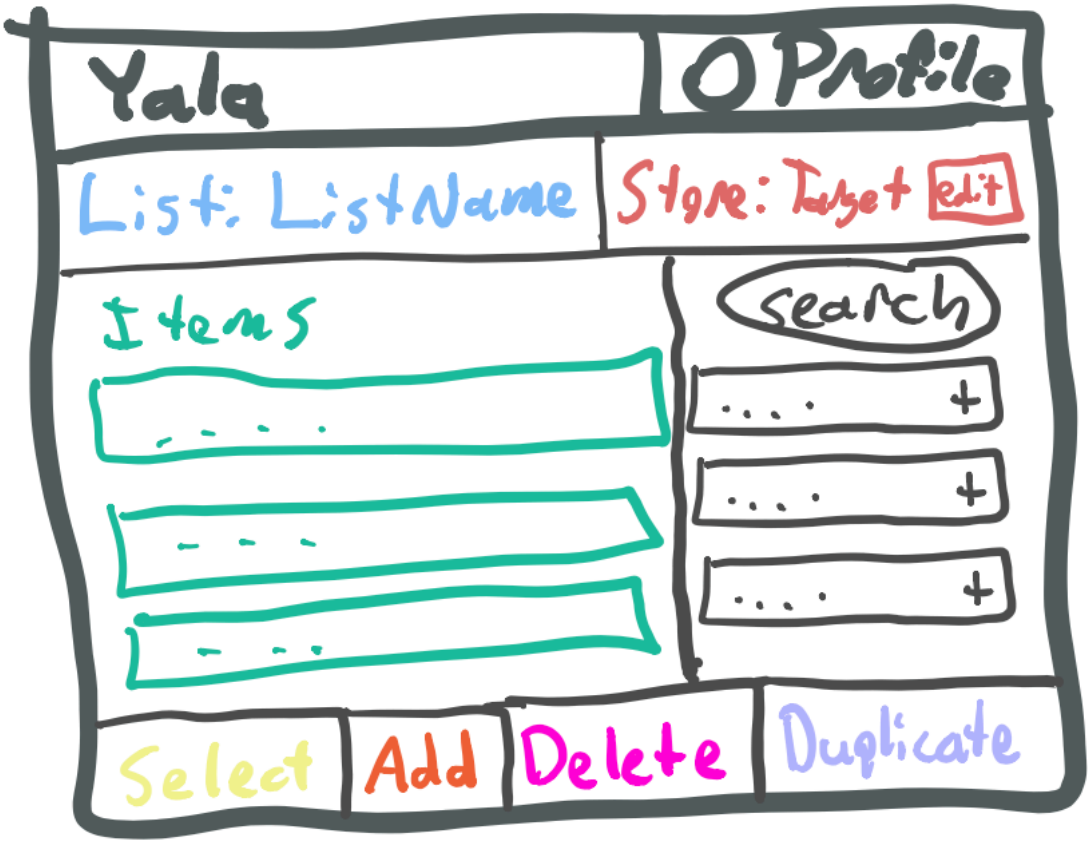
Classes Relationships Within Our System



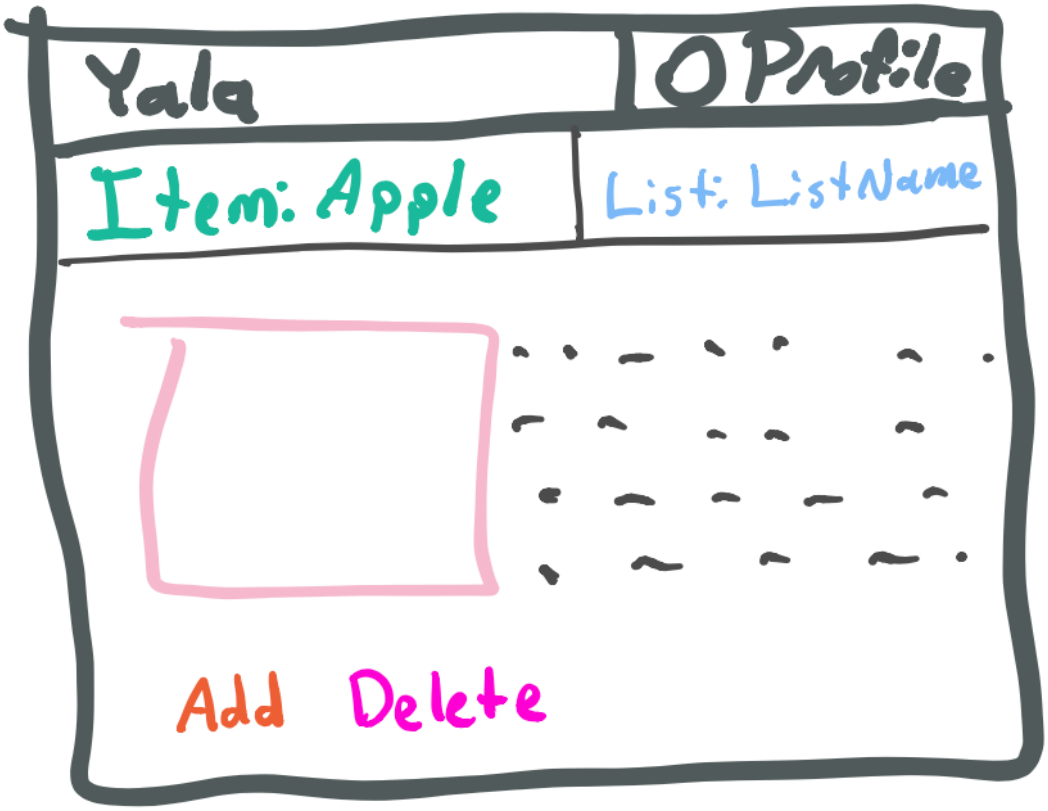
Rough Sketch of Functionality



List View

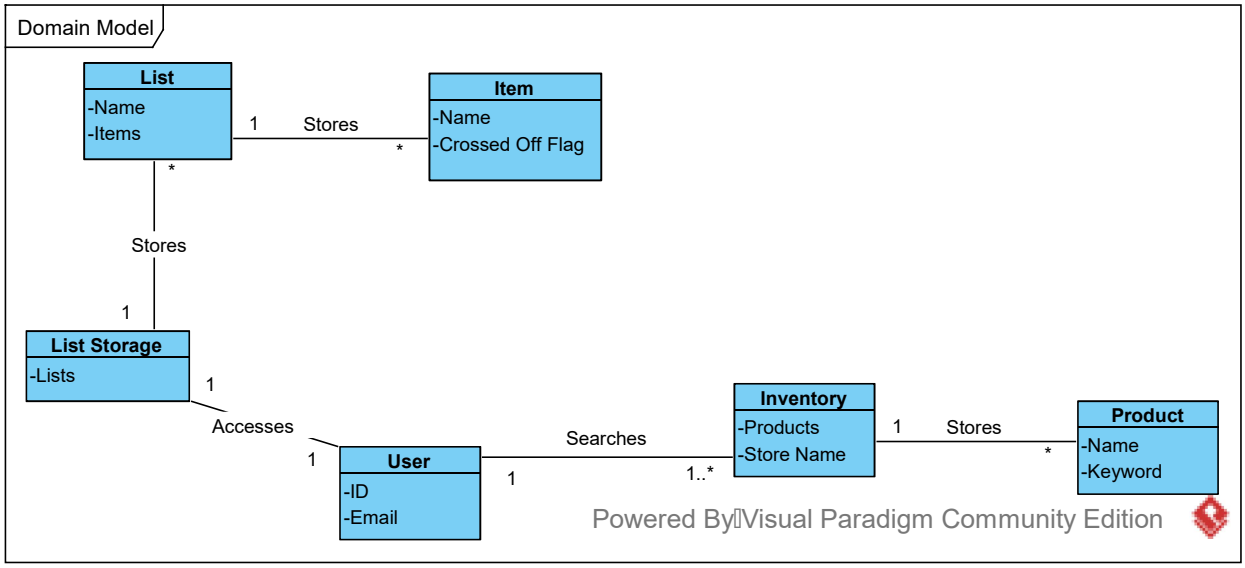


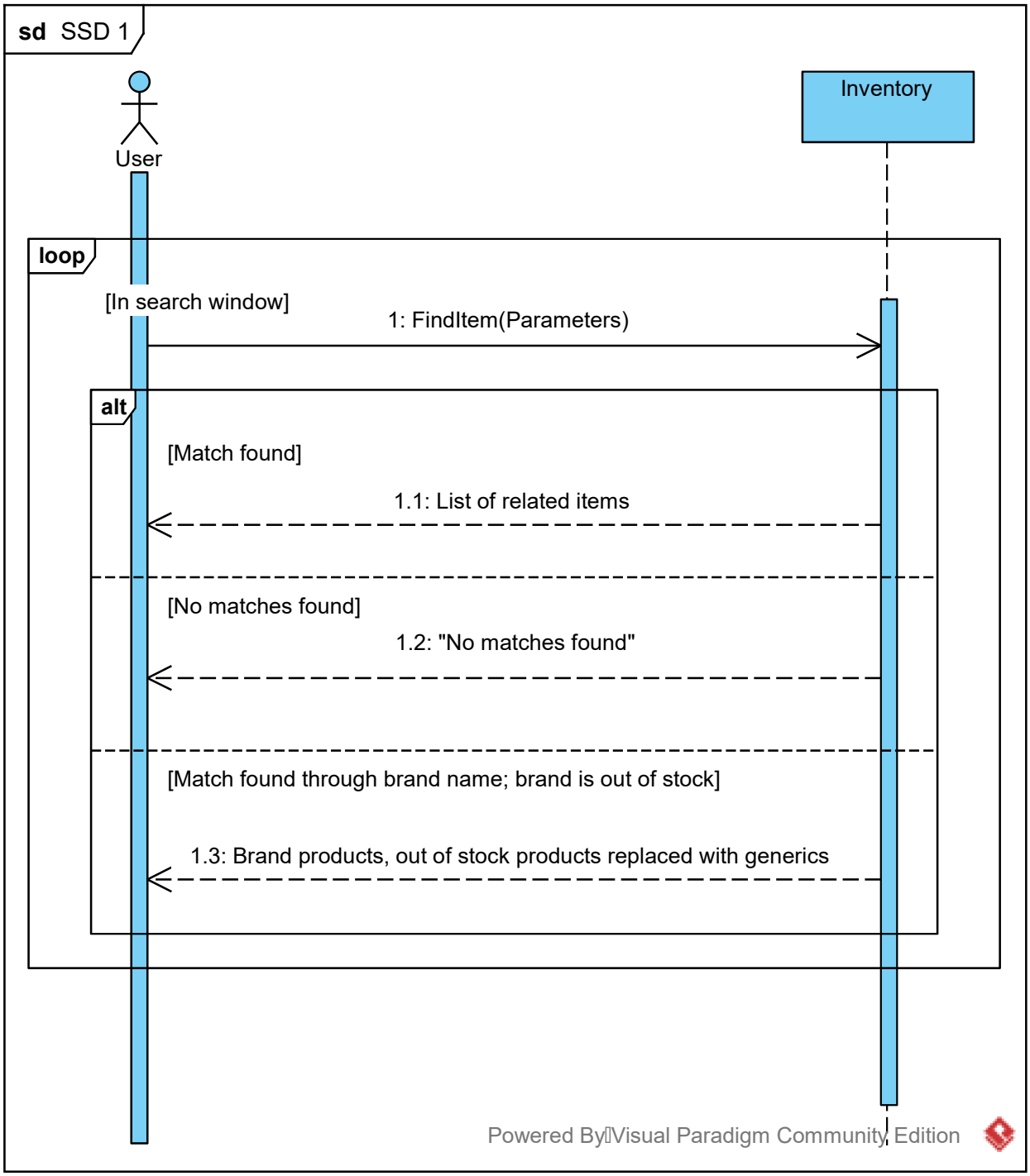
Item View

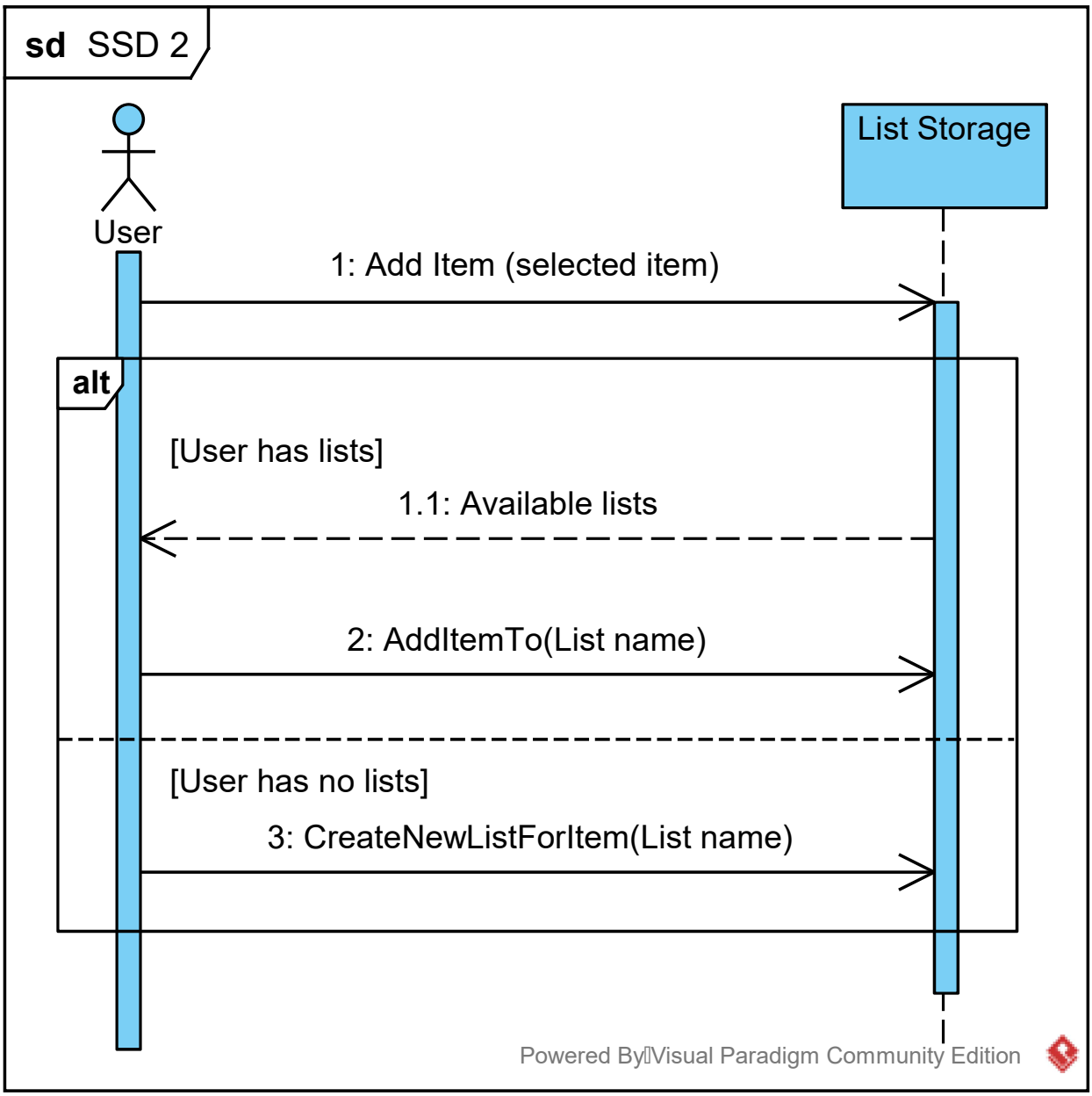


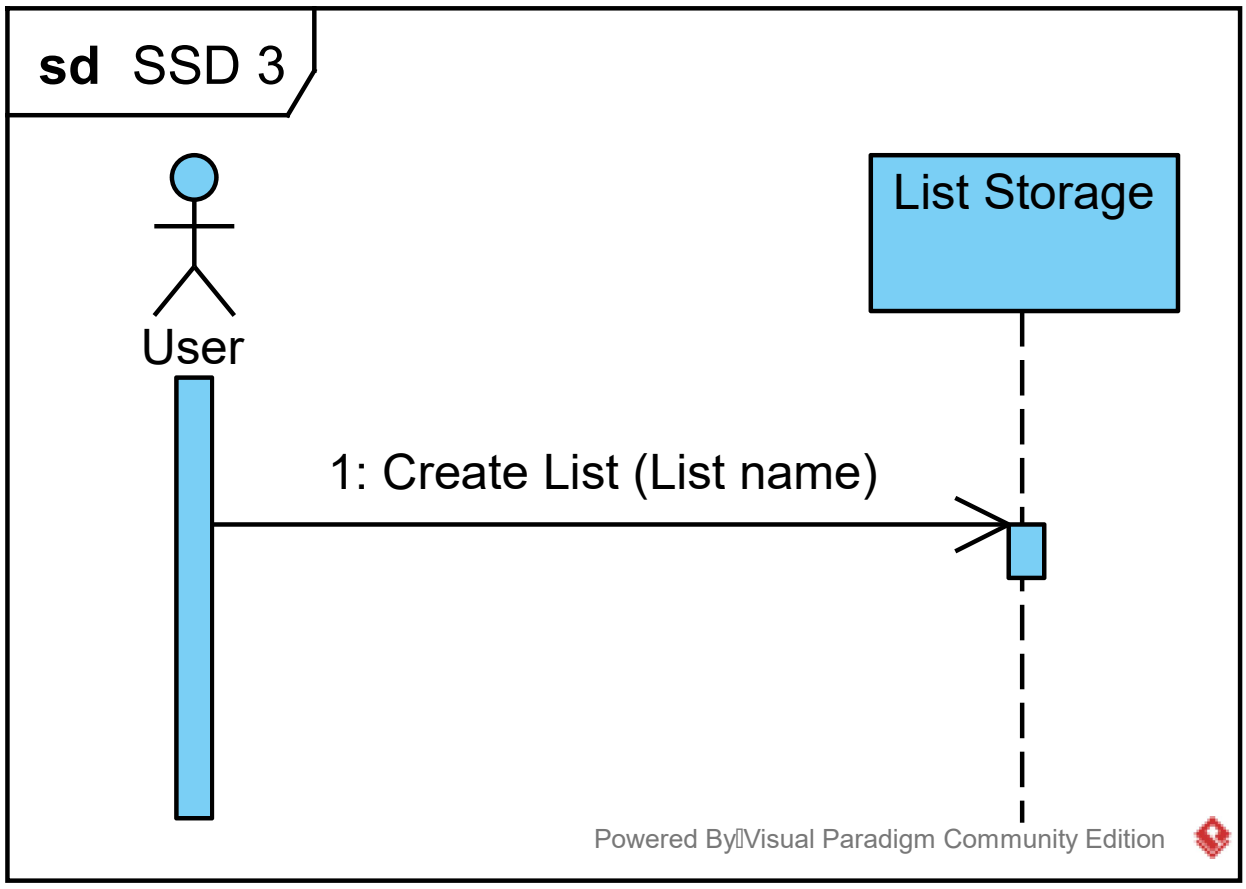
Definitions

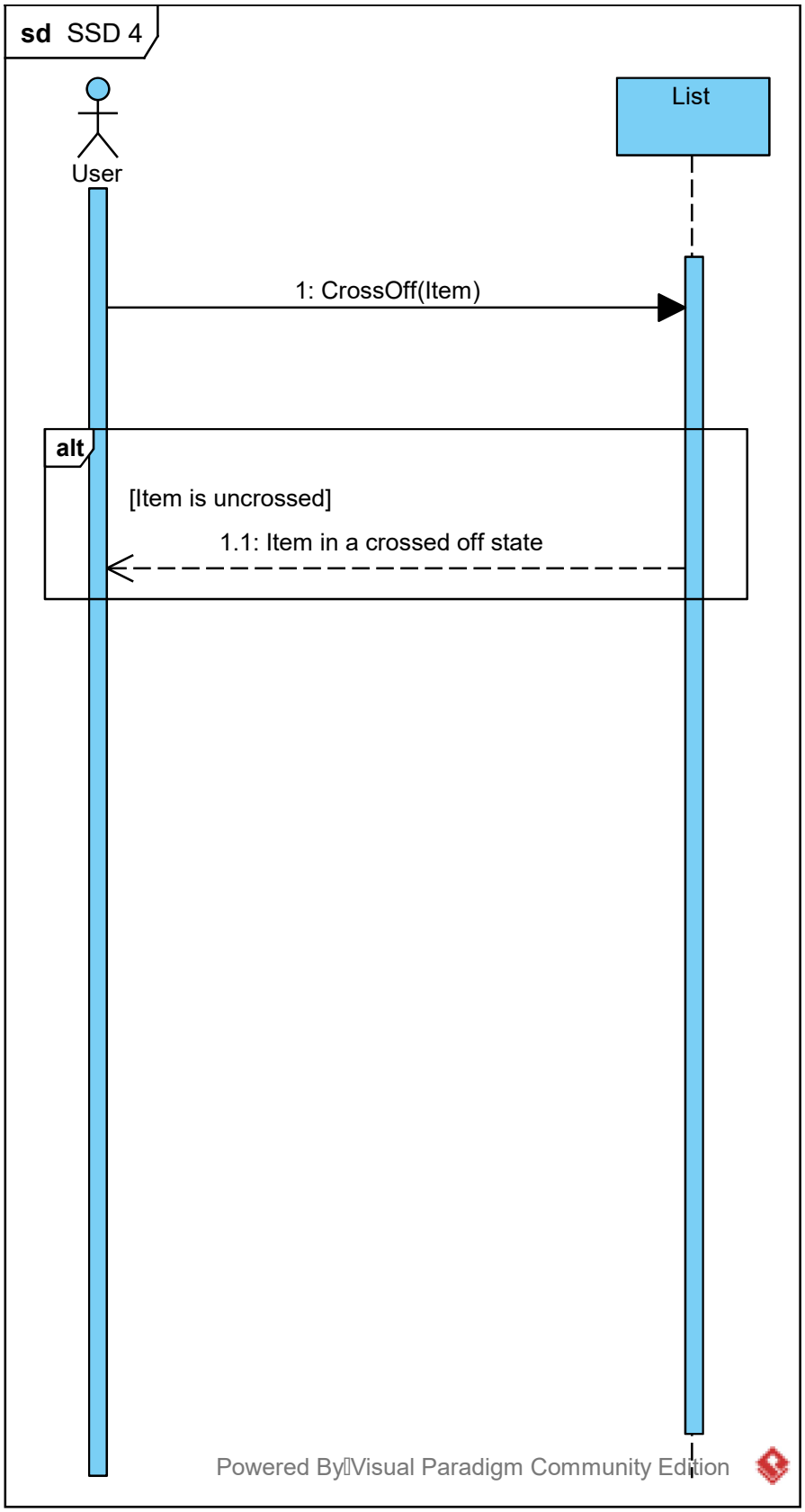
Name	Definition
System	Grocery Store Application
Item	The stored data on the grocery item – includes price, ID, name, etc.
Product	The physical manifestation of the item; placed in the user's cart in real life, contains all attributes of item

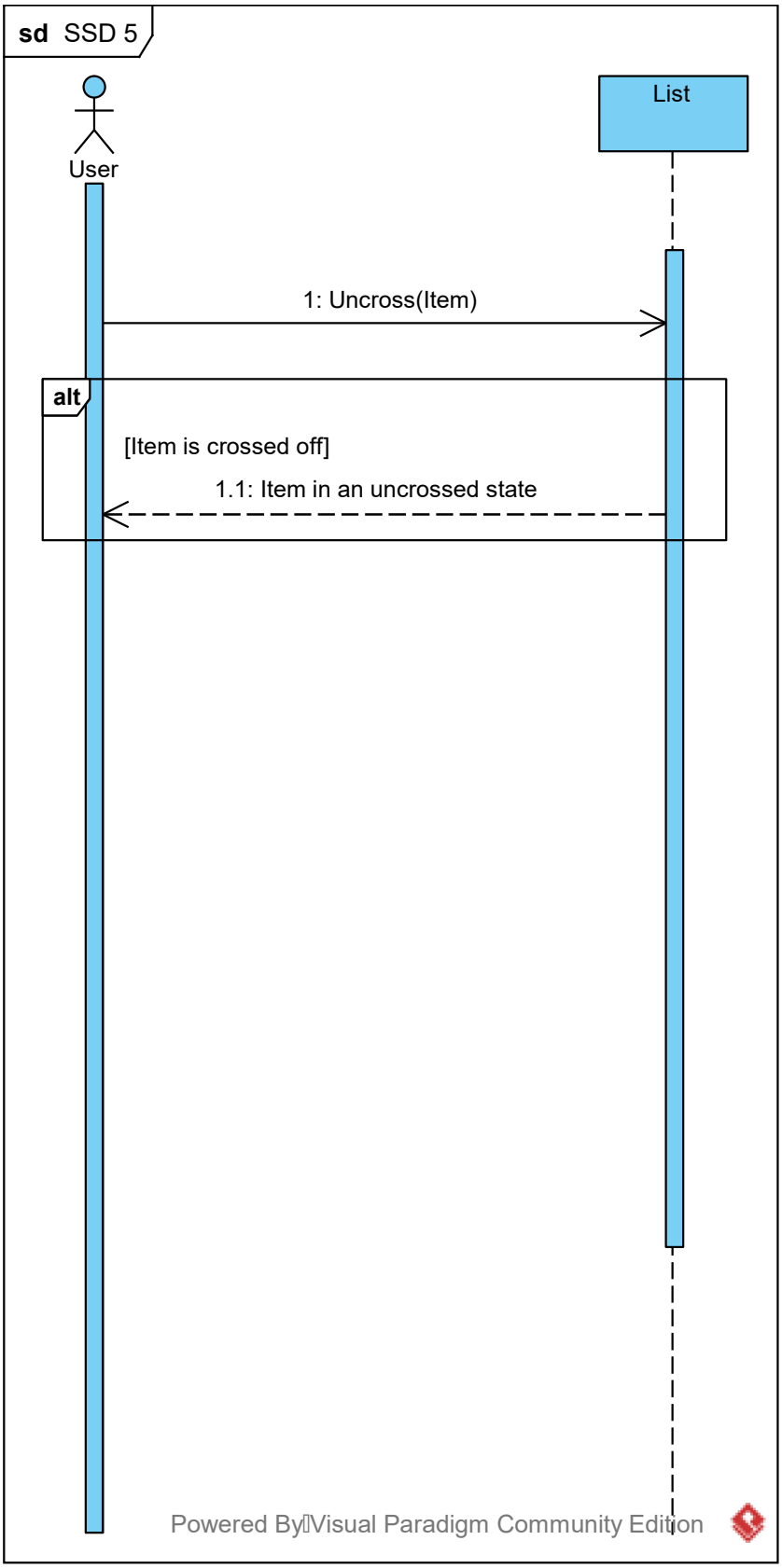

Domain Model
List
-Name
-Items
1
Stores
*
Item
-Name
-Crossed Off Flag
*
Stores
1
List Storage
-Lists
1
Accesses
1
User
-ID
-Email
Searches
1
1..*
Inventory
-Products
-Store Name
1
Stores
*
Product
-Name
-Keyword
Powered By Visual Paradigm Community Edition

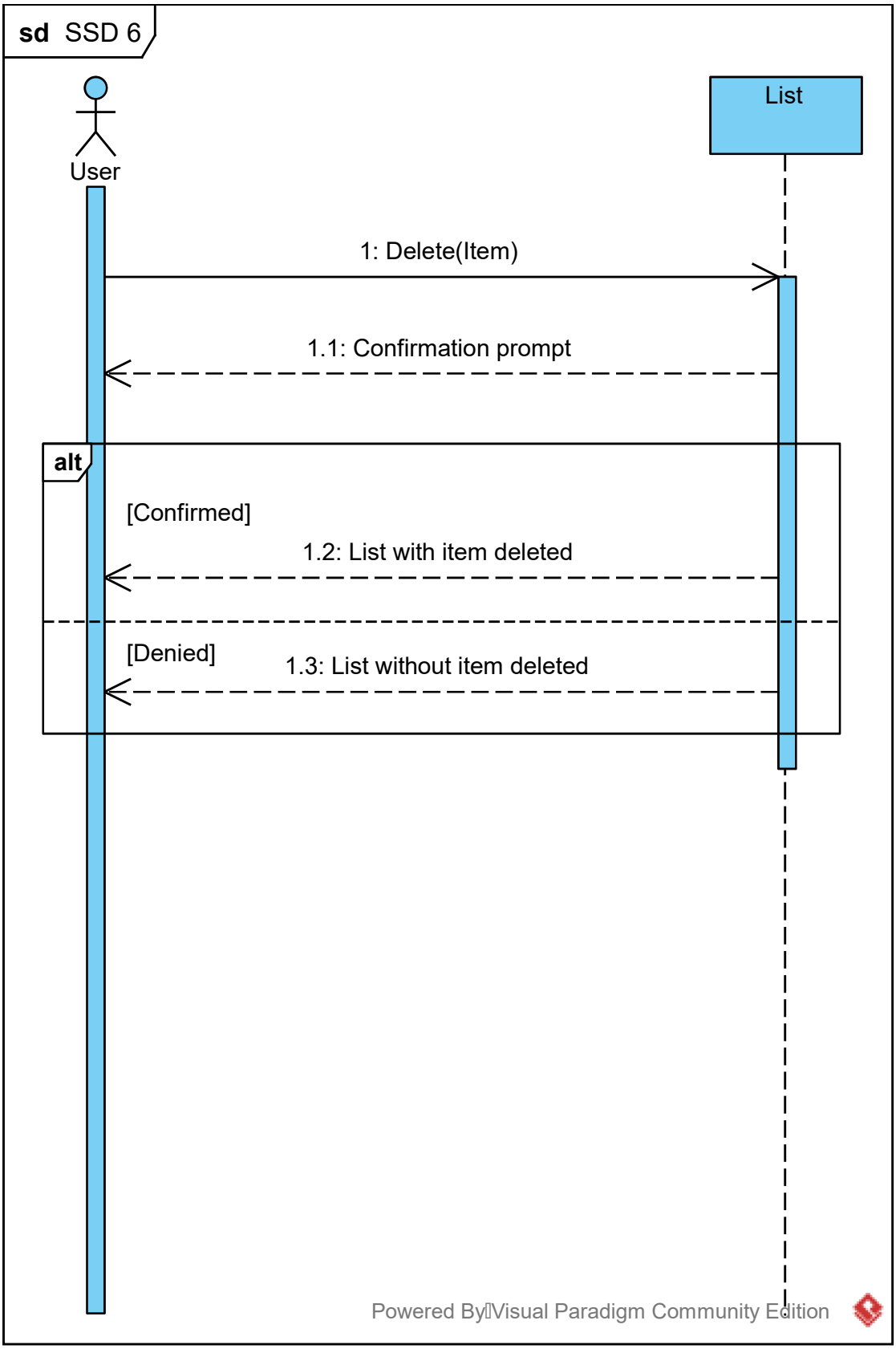

sd SSD 1
User
Inventory
loop
[In search window]
1: FindItem(Parameters)
alt
[Match found]
1.1: List of related items
[No matches found]
1.2: "No matches found"
[Match found through brand name; brand is out of stock]
1.3: Brand products, out of stock products replaced with generics
Powered By Visual Paradigm Community Edition

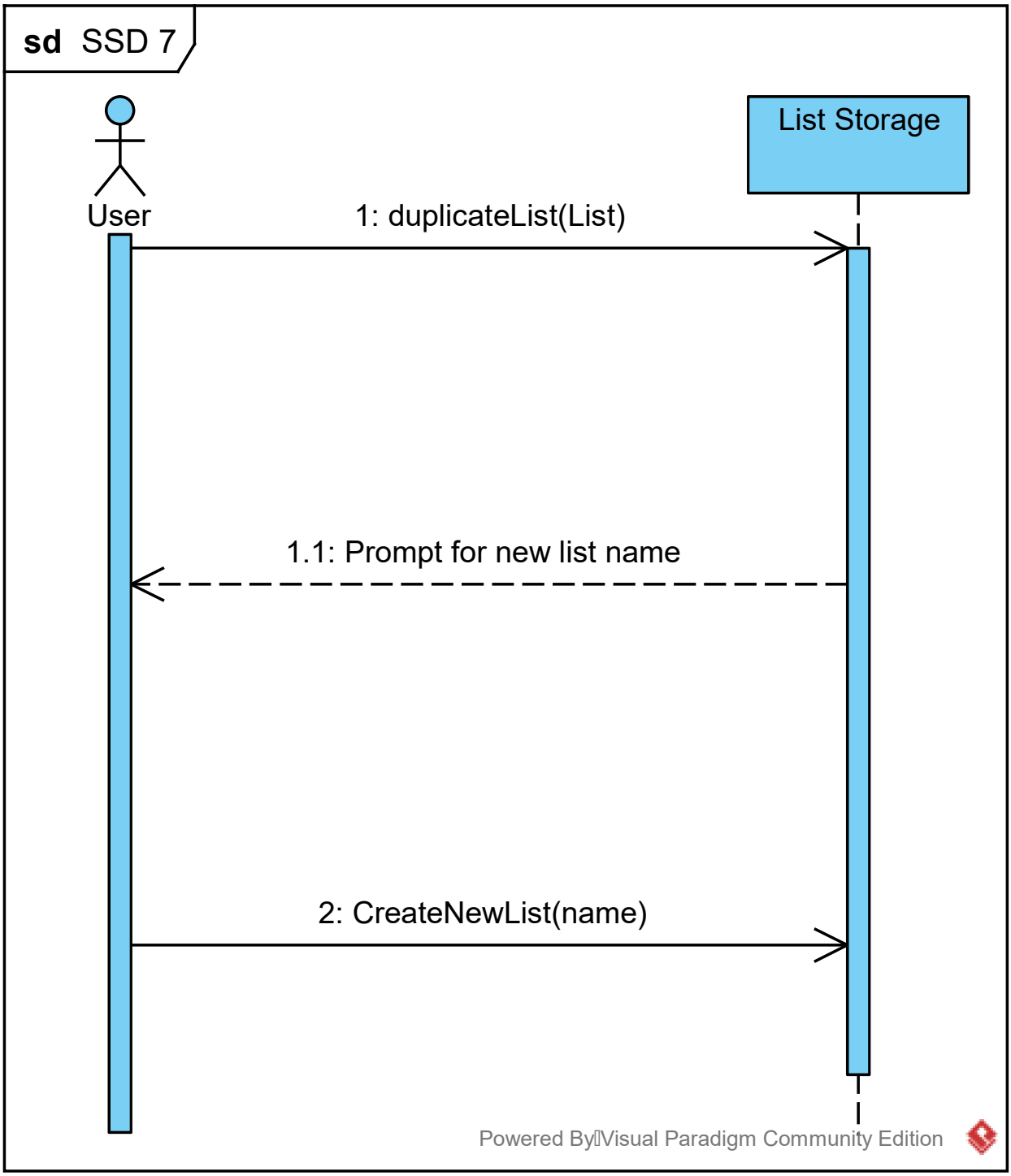

sd SSD 2
User
List Storage
1: Add Item (selected item)
alt
[User has lists]
1.1: Available lists
2: AddItemTo(List name)
[User has no lists]
3: CreateNewListForItem(List name)
Powered By Visual Paradigm Community Edition

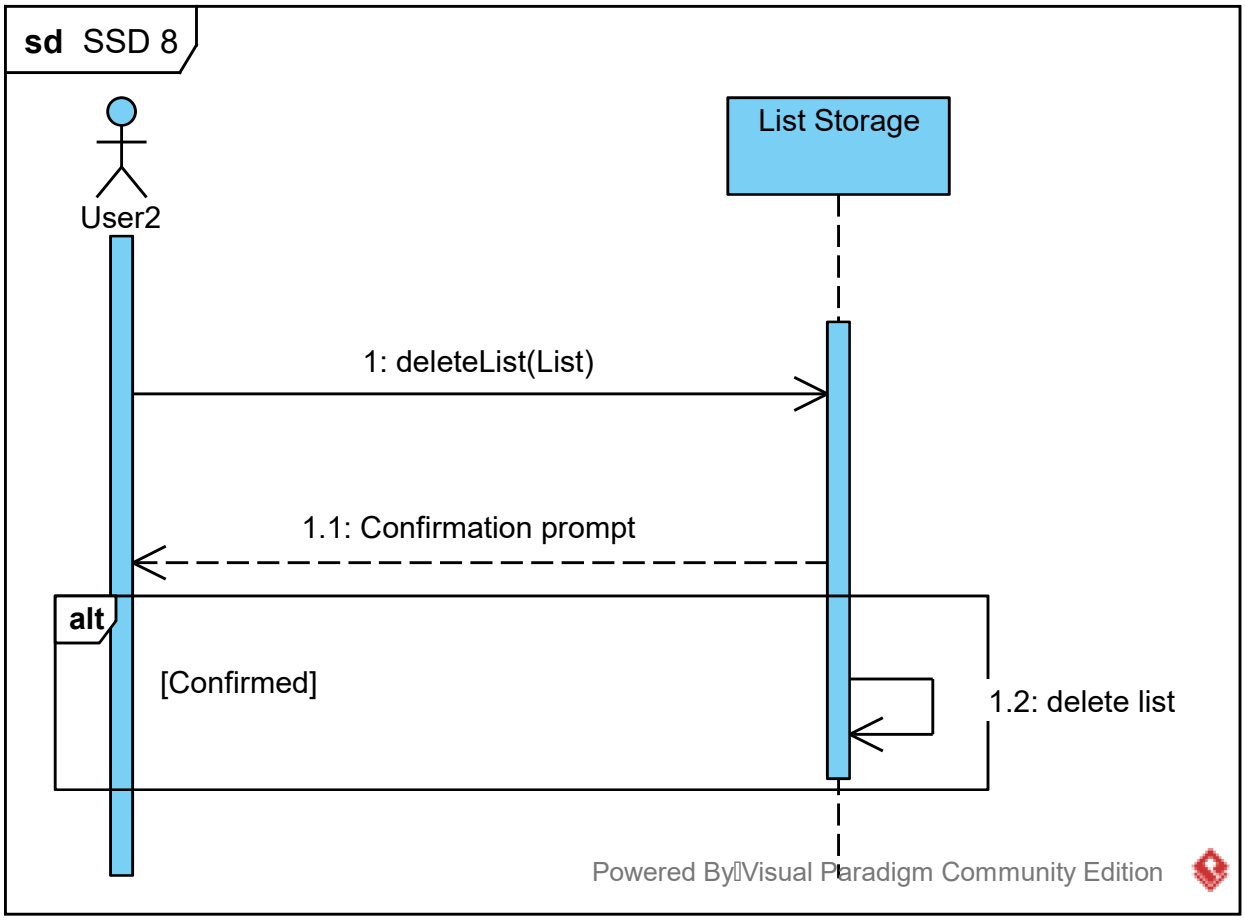

sd SSD 3
User
List Storage
1: Create List (List name)
Powered By Visual Paradigm Community Edition

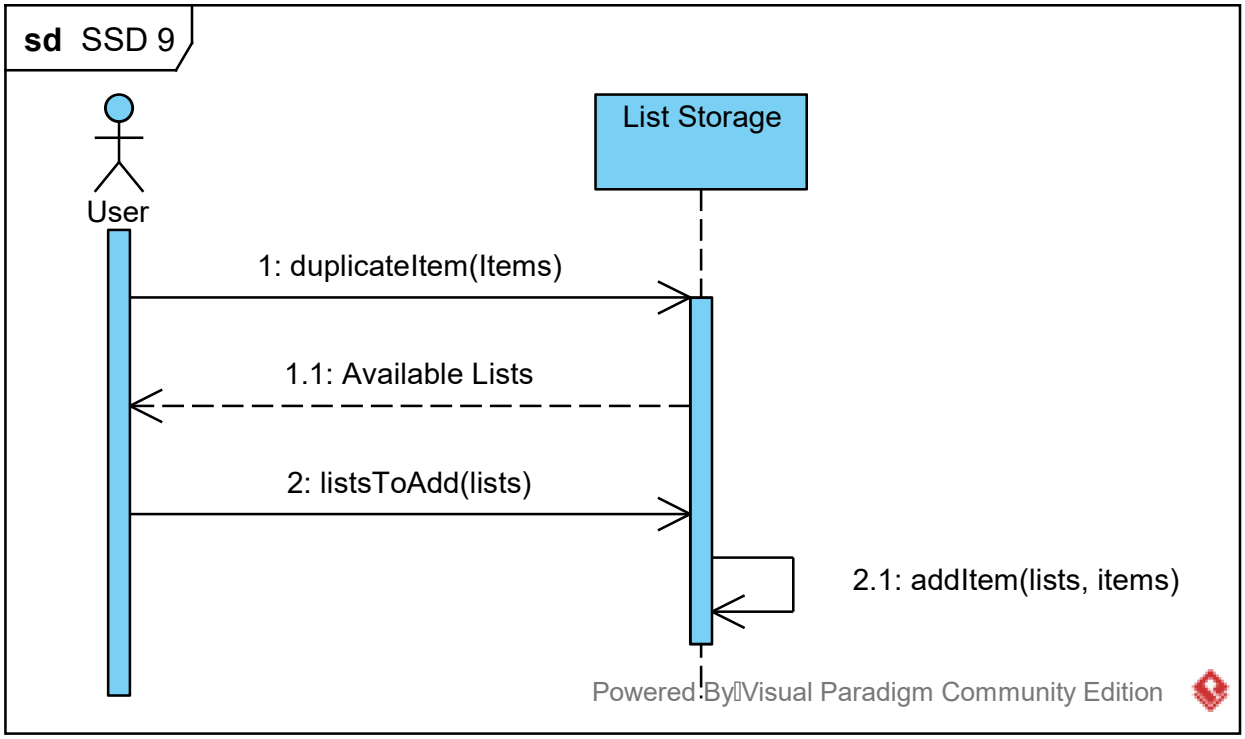

sd SSD 4
User
List
1: CrossOff(Item)
alt
[Item is uncrossed]
1.1: Item in a crossed off state
Powered By Visual Paradigm Community Edition


sd SSD 5
User
List
1: Uncross(Item)
alt
[Item is crossed off]
1.1: Item in an uncrossed state
Powered By Visual Paradigm Community Edition


sd SSD 6
User
List
1: Delete(Item)
1.1: Confirmation prompt
alt
[Confirmed]
1.2: List with item deleted
[Denied]
1.3: List without item deleted
Powered By Visual Paradigm Community Edition


sd SSD 7
User
List Storage
1: duplicateList(List)
1.1: Prompt for new list name
2: CreateNewList(name)
Powered By Visual Paradigm Community Edition


sd SSD 8
User2
List Storage
1: deleteList(List)
1.1: Confirmation prompt
alt
[Confirmed]
1.2: delete list
Powered By Visual Paradigm Community Edition


sd SSD 9
User
List Storage
1: duplicateItem(Items)
1.1: Available Lists
2: listsToAdd(lists)
2.1: addItem(lists, items)
Powered By Visual Paradigm Community Edition

OC 1

Problem Statement

Operation: FindItem(Parameters)

Cross Reference: Search Item

Precondition: The user is in the search window

Postconditions:

- A product list was created (instance creation)
- The product list was populated with products that match the search parameters (attribute modification)

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

OC 2

Problem Statement

Operation: AddItem(Item)

Cross Reference: Add Item

Precondition: There is an available product list to choose from

Postconidtions:

- A new item is created based on the selected product (instance creation)
- The product list is deleted (instance deletion)

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

OC 3

Problem Statement

Operation: AddItemTo(ListName)

Cross References: Add Item

Preconditions:

- A new item is ready to be added to a list
- The user already has lists

Postconditions:

- A list is selected to add the item to (association formed)

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

OC 4

Problem Statement

Operation: CreateNewListForItem(List name)

Cross References: Add Item

Preconditions:

- A new item is ready to be added to a list
- The user has no lists

Postconditions:

- A new list is created with the inputted name (instance creation)
- The list is added to List Storage (association formed)
- The item is added to the list (association formed)

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

OC 5

Problem Statement

Operation: CreateList(ListName)

Cross References: Create New List

Preconditions: The user has selected to create a new list

Postconditions:

- A new list is created with the inputted name (instance creation)
- The list is added to List Storage (association formed)

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

OC 6

Problem Statement

Operation: selectList(List)
References: Use Case 1: Copying List, Use Case 2: Deleting List
Preconditions: list menu is displayed and list is selected by the user
Postconditions: the selected list is passed and the list's information is
then requested to display it back

Operation: duplicateList(List)
References: Use Case 1: Copying List
Preconditions: a list is selected and displayed
Postconditions: the system prompts for a name for the list and then
duplicates it with a new name

Operation: duplicateList(String)
References: Use Case 1: Copying List
Preconditions: a list is selected and duplicate was pressed and the user entered the new name for the list
Postconditions: the list is duplicated

Operation: deleteList(List)
References: Use Case 2: DELETING List
Preconditions: a list is selected and delete list was pressed
Postconditions: the system confirms that the user would like to delete the list, the list and its items are deleted

Operation: confirmDeletion(bool)
References: Use Case 2: DELETING List

Preconditions: a list is selected and delete list was pressed
Postconditions: the list is deleted

Operation: selectItem(Item)
References: Use Case 3: Moving Items
Preconditions: the items were displayed and the user selected an item
Postconditions: the item is shown as selected

Operation: duplicateItem()
References: Use Case 3: Moving Items
Preconditions: the option to select more items is shown and the list of lists to select/deselect is shown

Operation: selectAnotherItem(Item)
References: Use Case 3: Moving Items
Preconditions: the items were displayed and the user selected at least two

items

Postconditions: the items are shown as selected

Operation: selectLists(bool[])

References: Use Case 3: Moving Items

Preconditions: the user has selected items to duplicate and has selected or deselected a list or multiple lists

Postconditions: the selected items are duplicated into the selected lists and if the current list was deselected the selected items are removed from it

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

Problem Statement

Operation: checkOff(Items)
References: Crossing Use Case
Preconditions: A list is open with items that the user wants crossed off
Postconditions: The selected item in the list is crossed off

Operation: uncross(Items)
References: Uncrossing Use Case
Preconditions: A list is open with items crossed off that the user wants to uncrossed
Postconditions: The specified items in the list are uncrossed

Operation: delete(Items)
References: Delete Use Case
Preconditions: User has confirmed an item for deletion

Postconditions: Specified items are deleted from list

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 1

Problem Statement

ID: Use Case 01 (Search for Item)

Scope: Product searcher

Stakeholders:

User- wants to easily find items to add to their shopping list

Store- wants to quickly provide the user with relevant products

Precondition: The user has the search window open

Postcondition: The user has a list of results that match their query

Main Success Scenario:

1. The user puts search parameters into the search bar
2. The store's inventory is scanned for

matching words

3. Search results are returned that match the parameters

Extensions:

3. a. No matches are found

1. The list says no matches were found and asks the user to try again

3. b. A specified item is out of stock

1. If a matching item is out of stock, it will notify the user and be replaced by a generic alternative

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 2

Problem Statement

ID: UC 02 (Add item)

Scope: List editing

Stakeholders:

User- wants to quickly add items from the search onto their lists

Precondition: An item search has been successfully completed

Postcondition: An item is added to the user's list.

Main Success Scenario:

1. The user selects an item
2. The system asks which list the user wants to add the item to, or if they want to make a new one
3. The user selects the list

4. The items is added to the selected list

Extensions:

2. a. The user has no lists
    1. The system begins list creation and

asks the user for a list name

2. The user provides a name

3. The list is created and the item added

2. b. The user creates a new list

1. The system begins list creation and asks the user for a list name

2. The user provides a name

3. The list is created and the item added

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 3

Problem Statement

ID: UC 03 (Create list)

Scope: List storage

Stakeholders:

User- wants to quickly create a new list

Preconditions: The user has selected to create a new list

Postconditions: A new list is created

Main Success Scenario:

1. The user selects to create a new list
2. The system asks for the name of the list
3. The user inputs the list name
4. The list is created with that name

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 4

Problem Statement

ID: Use Case 4 (Cross out item)

Scope: Item management

Stakeholders:

User - seeks to cross off items from list so they know if they have it in cart

Precondition: There is an uncrossed item on the list

Post Condition: Item is crossed off

Main Success Scenario:

1. User touches item
2. Item is crossed off

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 5

Problem Statement

ID: Use Case 5 (Uncross item)

Scope: Item management

Stakeholders:

User - seeks to uncross items from list

Precondition: The list has a crossed out item

Post Condition: Item is uncrossed

Main Success Scenario:

1. User touches a crossed out item
2. Item is uncrossed

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 6

Problem Statement

ID: Use Case 6 (Delete item)

Scope: Item management

Stakeholders:

User - seeks to permanently delete an item from the list

Precondition: An item is on the list

Post Condition: Item is deleted from list

Main Success Scenario:

1. User selects items to access delete menu
2. User confirms the item will be deleted
3. Item is deleted from the list

Extensions:

2. a. The user no longer wants to delete the item
    1. User closes the delete menu

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 7

Problem Statement

ID: Use Case 7 (Duplicate list)

Scope: List management

Stakeholders:

User - seeks to duplicate a list

Precondition: User is viewing the list menu

Post Condition: A new list is created with the same items as the selected list.

Main Success Scenario:

1. User selects a list
2. User clicks "Duplicate List"
3. System asks for a new name for the list
4. User gives a new name for the duplicate
5. The new list is created and the items are copied to the new list.

Extensions:

4. a. The user leaves the new name blank
    1. System automatically names the new list [Original List Name]_copy

Data Dictionary

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 8

Problem Statement

ID: Use Case 8 (Delete list)

Scope: List management

Stakeholders:

User - seeks to delete a list

Precondition: User is viewing the list menu

Post Condition: A certain list is deleted from emeory

Main Success Scenario:

1. User selects a list
2. User clicks "Delete List"
3. System prompts asking the user if they are sure
4. The user responds yes
5. The list (along with all of its items) is deleted

Extensions:

4. a. User responds no
    1. Prompt goes away and the list is not deleted

Data Dictionary

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.

UC 9

Problem Statement

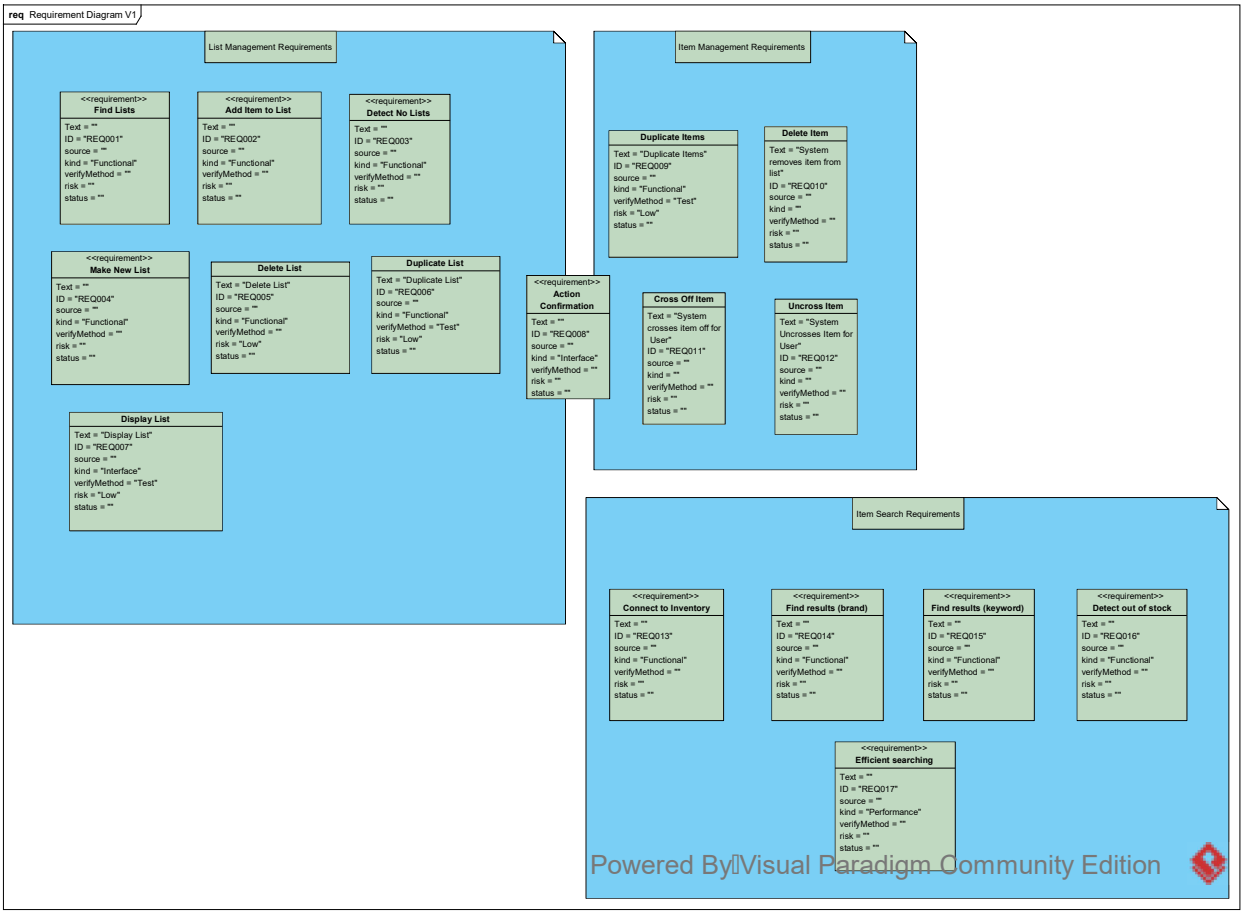
ID: Use Case 9 (Copy item to other lists)
Scope: Item management
Stakeholders:
User - seeks to move one or more items from one list to another
Precondition: User is viewing a list
Post Condition: An item or multiple items from one list are now contained in another and may be removed from the previous list in which they were contained
Main Success Scenario:
1. User selects an item or multiple items
2. User clicks "Duplicate Item(s)"
3. Prompt shows a list of the user's other lists
4. User selects which of these lists they would

like to have the items copied to
5. The lists the user selected now contain the item(s) the user selected and if the current list was unselected the item(s) are removed from

the current list

Data Dictionary

No.	Candidate Class	Extracted Text	Type	Description	Occur.


req Requirement Diagram V1
List Management Requirements
<<requirement>>
Find Lists
Text = ""
ID = "REQ001"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
<<requirement>>
Add Item to List
Text = ""
ID = "REQ002"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
<<requirement>>
Detect No Lists
Text = ""
ID = "REQ003"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
<<requirement>>
Make New List
Text = ""
ID = "REQ004"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
Delete List
Text = "Delete List"
ID = "REQ005"
source = ""
kind = "Functional"
verifyMethod = ""
risk = "Low"
status = ""
Duplicate List
Text = "Duplicate List"
ID = "REQ006"
source = ""
kind = "Functional"
verifyMethod = "Test"
risk = "Low"
status = ""
<<requirement>>
Action Confirmation
Text = ""
ID = "REQ008"
source = ""
kind = "Interface"
verifyMethod = ""
risk = ""
status = ""
Display List
Text = "Display List"
ID = "REQ007"
source = ""
kind = "Interface"
verifyMethod = "Test"
risk = "Low"
status = ""
Item Management Requirements
Duplicate Items
Text = "Duplicate Items"
ID = "REQ009"
source = ""
kind = "Functional"
verifyMethod = "Test"
risk = "Low"
status = ""
Delete Item
Text = "System removes item from list"
ID = "REQ010"
source = ""
kind = ""
verifyMethod = ""
risk = ""
status = ""
Cross Off Item
Text = "System crosses item off for User"
ID = "REQ011"
source = ""
kind = ""
verifyMethod = ""
risk = ""
status = ""
Uncross Item
Text = "System Uncrosses Item for User"
ID = "REQ012"
source = ""
kind = ""
verifyMethod = ""
risk = ""
status = ""
Item Search Requirements
<<requirement>>
Connect to Inventory
Text = ""
ID = "REQ013"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
<<requirement>>
Find results (brand)
Text = ""
ID = "REQ014"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
<<requirement>>
Find results (keyword)
Text = ""
ID = "REQ015"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
<<requirement>>
Detect out of stock
Text = ""
ID = "REQ016"
source = ""
kind = "Functional"
verifyMethod = ""
risk = ""
status = ""
<<requirement>>
Efficient searching
Text = ""
ID = "REQ017"
source = ""
kind = "Performance"
verifyMethod = ""
risk = ""
status = ""
Powered By Visual Paradigm Community Edition

Requirement Identifiers (REQ…)	1	2	3	4	5	6	7	8	9	10	11	12	13	15	16	17
UC1: Search Item													✓	✓	✓	✓
UC2: Add Item	✓	✓	✓										✓	✓	✓	✓
UC3: Cross Off Item	✓										✓					
UC4: Delete Item	✓									✓						
UC5: Uncross Item	✓											✓				
UC6: Create New List	✓	✓	✓	✓			✓									
UC7: Delete List	✓		✓		✓		✓	✓		✓						
UC8: Copy List	✓			✓		✓	✓									
UC9: Move Item to Different List	✓								✓							

YALA_Gantt_Diagram

Sep 28, 2020

https://github.com/jake-allen/YALA

Software I - Jake Allen, Evan Berryman, Sam Hobbs

Project manager	
Project dates	Sep 7, 2020 - Nov 28, 2020
Completion	18%
Tasks	38
Resources	3

Tasks

Name	Begin date	End date
Inception	9/7/20	9/25/20
Vision of the Project	9/7/20	9/25/20
Project Plan	9/7/20	9/11/20
Use Cases & Scenarios	9/11/20	9/23/20
Relevant Diagrams	9/11/20	9/25/20
Requirement Analysis	9/11/20	9/25/20
Glossary	9/23/20	9/25/20
Iteration I Presentation	9/25/20	9/28/20
Finalize Presentation	9/25/20	9/25/20
Present	9/28/20	9/28/20
Elaboration	9/28/20	10/13/20
Analytical Model	9/28/20	10/1/20
Software Architecture Description	10/5/20	10/6/20
Architecture Prototype	9/28/20	10/7/20
Refine Architecture Prototype	9/28/20	10/9/20
Design Model	10/9/20	10/13/20
Iteration II Presentation	10/8/20	10/12/20
Prepare Presentation	10/8/20	10/9/20
Present	10/12/20	10/12/20
Construction	10/6/20	11/4/20
Design Model	10/6/20	10/16/20
Software Components and Subsystems	10/8/20	10/26/20
Integration	10/14/20	10/30/20
Test Planning	10/20/20	11/4/20
Documentation	10/30/20	11/4/20

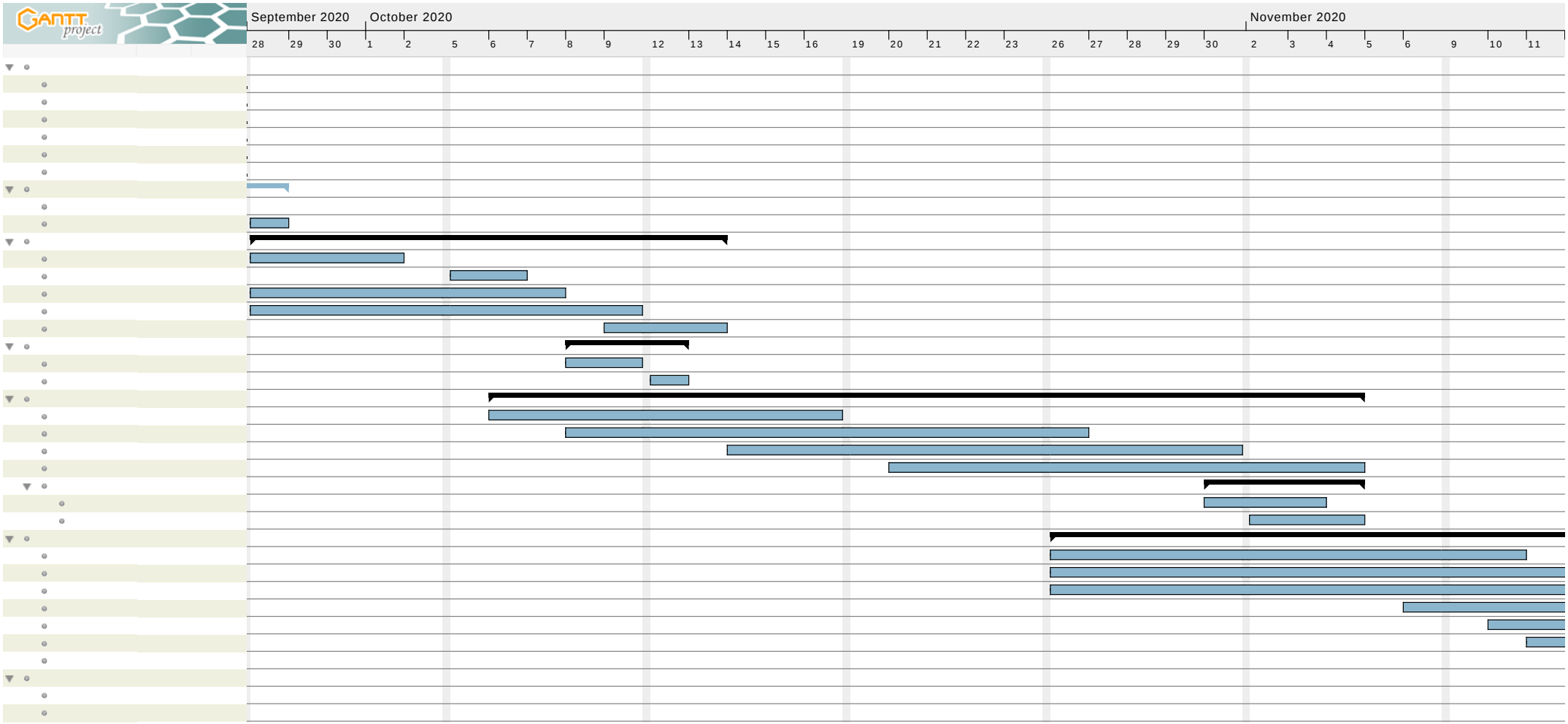
Tasks

Name	Begin date	End date
User Manual	10/30/20	11/3/20
Installation Guide	11/2/20	11/4/20
Transition	10/26/20	11/25/20
Software Increment	10/26/20	11/10/20
Software Maintenance	10/26/20	11/19/20
Test Reporting	10/26/20	11/17/20
Statistics/User Monitoring	11/6/20	11/17/20
User Feedback	11/10/20	11/17/20
Performance Analysis	11/11/20	11/18/20
Adaptations	11/17/20	11/25/20
Iteration III Presentation	11/26/20	11/27/20
Prepare Presentation	11/26/20	11/26/20
Present	11/27/20	11/27/20

Resources

Name	Default role
Evan Berryman	Leader / Engineer
Jake Allen	Project Manager / Engineer
Sam Hobbs	Engineer

Gantt Chart


GANTT project
September 2020
October 2020
November 2020
28 29 30 1 2 5 6 7 8 9 12 13 14 15 16 19 20 21 22 23 26 27 28 29 30 2 3 4 5 6 9 10 11

Resources Chart