Map

Description automatically generated

Food Management

System Proposal

David Lockwood | IT252 | 4/14/2021

# Table of Contents

[Table of Contents 1](#_Toc71735547)

[Executive Summary 3](#_Toc71735548)

[System Request 4](#_Toc71735549)

[Work Plan 5](#_Toc71735550)

[Feasibility Analysis 6](#_Toc71735551)

[Technical Feasibility: 6](#_Toc71735552)

[Economic Feasibility: 6](#_Toc71735553)

[Organizational Feasibility: 6](#_Toc71735554)

[Requirements Definition 7](#_Toc71735555)

[Functional Requirements 7](#_Toc71735556)

[Nonfunctional Requirements 7](#_Toc71735557)

[Use Cases 8](#_Toc71735558)

[Use Case #1 8](#_Toc71735559)

[Use Case #2 9](#_Toc71735560)

[Use Case #3 10](#_Toc71735561)

[Use Case #4 11](#_Toc71735562)

[Use Case #5 12](#_Toc71735563)

[Use Case #6 13](#_Toc71735564)

[Process Models 14](#_Toc71735565)

[Log in sub-process 14](#_Toc71735566)

[Use Case #1: Add Item 15](#_Toc71735567)

[Use Case # 2: Create User-defined Item 16](#_Toc71735568)

[Use Case #3: Search Recipes 17](#_Toc71735569)

[Use Case #4: User-Defined Recipe 18](#_Toc71735570)

[Use Case #5: Add Items to Shopping List 19](#_Toc71735571)

[Use Case #6: Crossing Item Off Shopping List 20](#_Toc71735572)

[Data Model 21](#_Toc71735573)

[User Interface 22](#_Toc71735574)

[Home Page 22](#_Toc71735575)

[Inventory page 23](#_Toc71735576)

[Shopping List Page 24](#_Toc71735577)

[Structure Chart 25](#_Toc71735578)

[Physical Data Flow Diagram 26](#_Toc71735579)

[1.0 Navigation 26](#_Toc71735580)

[1.1 Inventory 27](#_Toc71735581)

[1.2 Shopping List 28](#_Toc71735582)

[1.3 Recipe 28](#_Toc71735583)

[Class Diagram 29](#_Toc71735584)

[Program Specification 30](#_Toc71735585)

[Module: 1.1.2 30](#_Toc71735586)

[Module: 1.1.3 31](#_Toc71735587)

[Module: 1.2.1 32](#_Toc71735588)

[Module: 1.2.2 33](#_Toc71735589)

[Module: 1.3.1 34](#_Toc71735590)

[Module: 1.3.2 35](#_Toc71735591)

[Appendices 36](#_Toc71735592)

# Executive Summary

Food Management is a system for organizing a household’s inventory of food items and tracking expiration dates of perishable items. Users can also view recipes recommended based on their inventory, add their own recipes, or create a shopping list. These features are intended to assist users who may not know what meals to prepare with the food they already have, and limit wasteful spending when shopping.

According to a study performed by William & Mary’s Department of Kinesiology and Health Sciences, the average American consumer spends roughly $1,300 per year on food that ends up being wasted[1]. The Centers for Disease Control and Prevention estimates that there are about 48 million cases of foodborne illness every year[2].

By providing a means to organize inventory and facilitating better shopping habits, Food Management will provide value to consumers that are becoming aware of how much money they are potentially wasting. By tracking expiration dates this system can inform users when items are about to perish. This will not only save them money, but also prevent any foodborne illnesses resulting from consuming expired food.

Once a large enough userbase is established, the system will accumulate a large amount of data on demographics and their shopping habits. This data is valuable to grocery stores to in turn help limit their waste from overstocking undesired items. By selling this data to grocery stores Food Management will generate revenue.

There is a growing need in America to address the amount of food wasted annually. News outlets are raising awareness and state governments are writing laws. It is my belief that it can also be addressed at the consumer level. Americans report that saving money is the most important motivator for reducing food waste. Providing a means to do so will greatly contribute to solving the problem.

# System Request

|  |
| --- |
| Food Management |
| Project Sponsor:  David Lockwood |
| Business Need:  Food Management will help anyone who grocery shops regularly by providing a means to track his or her inventory, shopping list, and browse various recipes at home on a desktop or on the go with a mobile device. |
| Business Requirements:  The requirements are a web-based application as well as a database to capture the state of the user’s inventory, store their favorite recipes and shopping list. It must be mobile-friendly, easily accessible when away from home if necessary. |
| Business Value:  Food Management will give value to users by helping them limit wasted groceries, saving them money, and getting more value for money spent. |
| Special Issues or Constraints:  The system will require manual input of inventory. A future enhancement will include barcode scanning with a smartphone’s camera. |

# Work Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Task Name | Start Date | End Date | Status |
| System Request | 3/11/2021 | 3/17/2021 | Complete |
| System Proposal | 4/1/2021 | 4/14/2021 | Complete |
| Work Plan | 4/7/2021 | TBD | In Progress |

The Work Plan is a work in progress and will be updated once more tasks are assigned.

# Feasibility Analysis

## Technical Feasibility:

* Food Management’s risk regarding familiarity with inventory management applications is high.
  + Developer has no experience creating an inventory management application.
  + Developer has knowledge regarding the subject matter of the application.
  + Resources are available to assist with development of the application.

## Economic Feasibility:

* Food Management will operate at a loss initially.
  + The application will be free to use.
* Once a large userbase is established, demographic data can be sold to grocery stores.

## Organizational Feasibility:

* Food Managements risk regarding the organization is low.
  + Developer’s interest in the project is strong.
* Provides value for users looking to save money on groceries.
* There is no existing system to replace.

# Requirements Definition

## Functional Requirements

1. Inventory
   * Display the user’s inventory.
   * Provide a means for adding items to their inventory.
   * Allow users to add, edit and delete user-defined items.
2. Recipes
   * Display recipes containing a picture, ingredients, and directions.
   * Suggest recipes based on items in the user’s inventory.
   * Provide a means to search for recipes.
   * Track user’s favorite recipes.
   * Provide a means for users to store their own recipes.
   * Allow users to add a recipe’s ingredients to their shopping list.
3. Shopping List
   * Display the user’s shopping list.
   * Allow user to add or remove items.
   * Store user’s previous shopping list and allow them to view it.
   * Allow the user to mark off items obtained.
   * Print the shopping list.

## Nonfunctional Requirements

1. Operational
   * The system should be accessible on the Web.
   * The system should be compatible with mobile devices.
2. Performance
   * The system should always be available.
   * The system should be able to process user’s input within 5 seconds.
3. Security
   * User’s login credentials should be secure.
4. Cultural & Political
   * The system should provide accommodations for users from diverse cultures and backgrounds.

# Use Cases

## Use Case #1

|  |
| --- |
| Use Case Name:  Add Inventory Item |
| Actor:  User |
| Description:  This use case describes how the user adds an item to inventory. |
| Trigger:  External |
| Normal Course:   1. The user arrives on the home page. 2. The user clicks the Inventory navigation button. 3. The user locates the item in the inventory list. 4. The user updates the item’s quantity field to indicate one or more. 5. The user clicks the Submit button. |
| Postconditions:   1. The database is updated to reflect the change in the item’s quantity. 2. The user is informed of the success of the task. |
| Exceptions:   1. The system fails to update the database.    1. System will inform user of the failure. |

## Use Case #2

|  |
| --- |
| Use Case Name:  Create User-Defined Inventory Item |
| Actor:  User |
| Description:  This use case describes how the user creates a user-defined item. |
| Trigger:  External |
| Precondition:   1. Item is not currently defined by the system. |
| Normal Course:   1. The user arrives at the home page. 2. The user clicks the Inventory navigation button. 3. The user clicks the Create New Item button. 4. The user fills out the required fields: name and food group. 5. The user clicks the Submit button. |
| Exceptions:   1. The user does not include a required field when filling out the Create New Item form.    1. The system will indicate which fields are necessary for the user-defined item to be accepted. 2. The system fails to update the database.    1. System will inform user of the failure. |

## Use Case #3

|  |
| --- |
| Use Case Name:  Search Recipes |
| Actor:  User |
| Description:  This use case describes how the user searches for recipes. |
| Trigger:  External |
| Normal Course:   1. The user arrives at the home page. 2. The user clicks the Recipes navigation button. 3. The user clicks the Search button. 4. The user defines the parameters of the search as well as how to display the results. 5. The user clicks the Submit button. |
| Postcondition:   1. The user receives and views the results of their search. |
| Exceptions:   1. The system fails to retrieve recipes from the database.    1. User will be informed of the task’s failure and returned to the search page. |

## Use Case #4

|  |
| --- |
| Use Case Name:  User-Defined Recipe |
| Actor:  User |
| Description:  This use case describes how the user creates a user-defined recipe. |
| Trigger:  External |
| Normal Course:   1. The user arrives at the home page. 2. The user clicks the Recipes navigation button. 3. The user clicks the Create New Recipe button. 4. The user inputs the required fields: title, ingredients, and directions. 5. The user clicks the Save Recipe button. |
| Postcondition:   1. The user is informed of the task’s success. |
| Exceptions:   1. User does not input all the required fields.    1. User will be directed to the fields that are necessary for the system to accept the recipe. 2. System fails to update the database with the user-defined recipe.    1. User will be informed of the task’s failure and returned to the Create New Recipe page. |

## Use Case #5

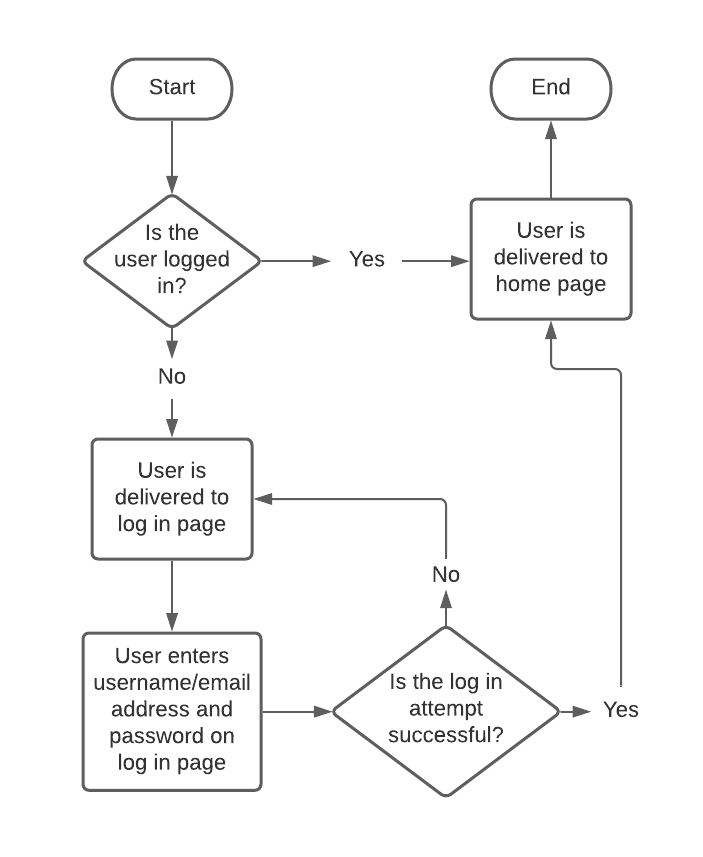
|  |
| --- |
| Use Case Name:  Add Multiple Items to Shopping List |
| Actor:  User |
| Description:  This use case describes how the user adds multiple items to their shopping list. |
| Trigger:  External |
| Normal Course:   1. The user arrives at the home page. 2. The user clicks the Shopping List navigation button. 3. The user clicks the Add Multiple Items button. 4. The user selects the items desired from the list displayed. 5. The user clicks the Add button. |
| Postconditions:   1. The database is updated with the items on the user’s shopping list. 2. The user is shown confirmation of the task’s success and returned to the shopping list page containing the updated list. |
| Exceptions:   1. The system fails to update the database.    1. The user will be informed of the task’s failure returned to the Add Multiple Items page. |

## Use Case #6

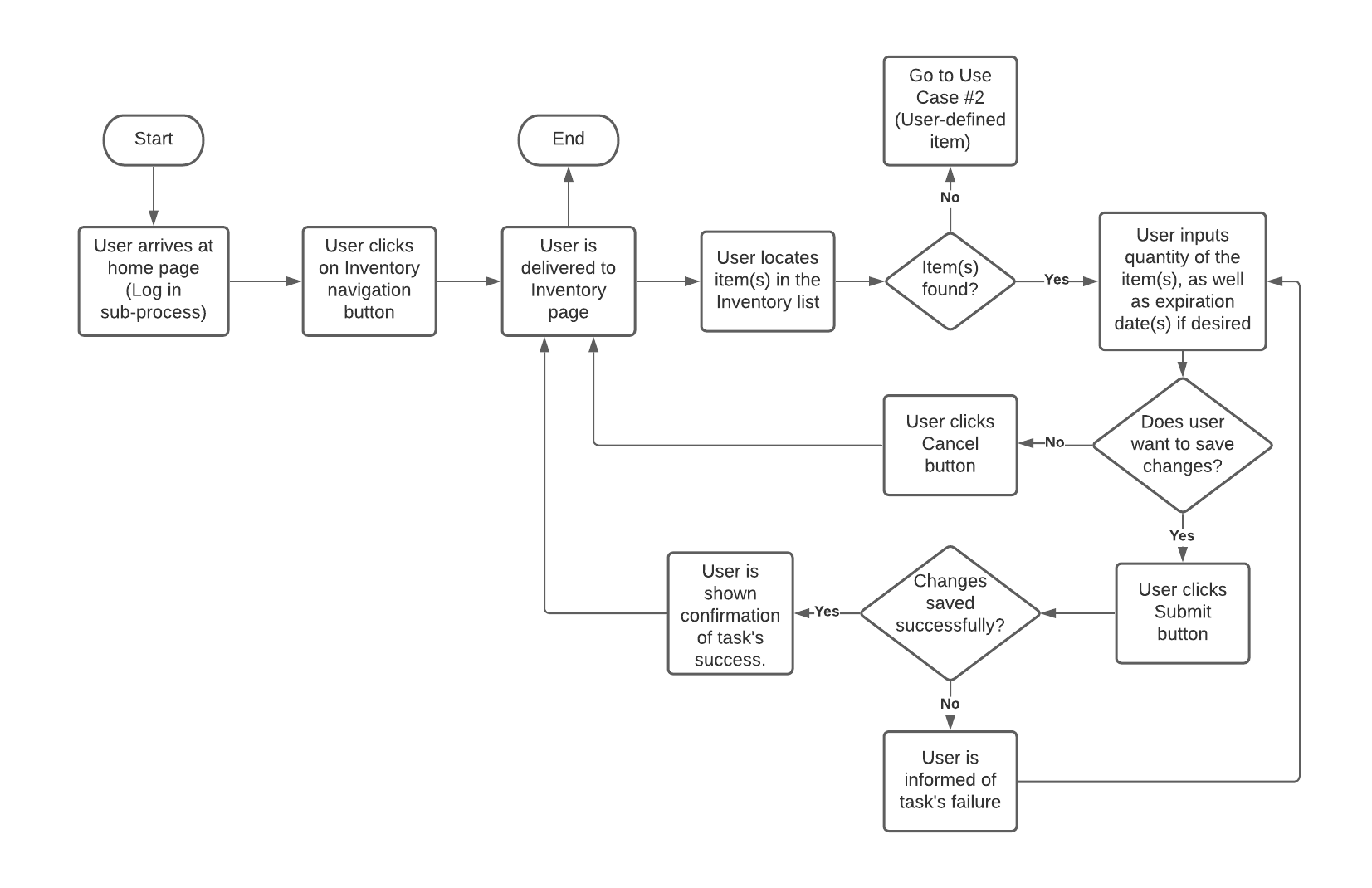
|  |
| --- |
| Use Case Name:  Crossing Item Off Shopping List |
| Actor:  User |
| Description:  This use case describes how the user indicates that an item has been obtained on their shopping list. |
| Precondition:   1. User has added an item or items to shopping list. |
| Normal Course:   1. The user arrives at the home page. 2. The user clicks the Shopping List navigation button. 3. The user obtains the item at the grocery store. 4. The user clicks the checkbox to the left of the item’s name to mark the item as obtained. |
| Postcondition:   1. The rest of the item’s fields, excluding the checkbox, will be greyed out and read-only to indicate the item as obtained. |

# Process Models

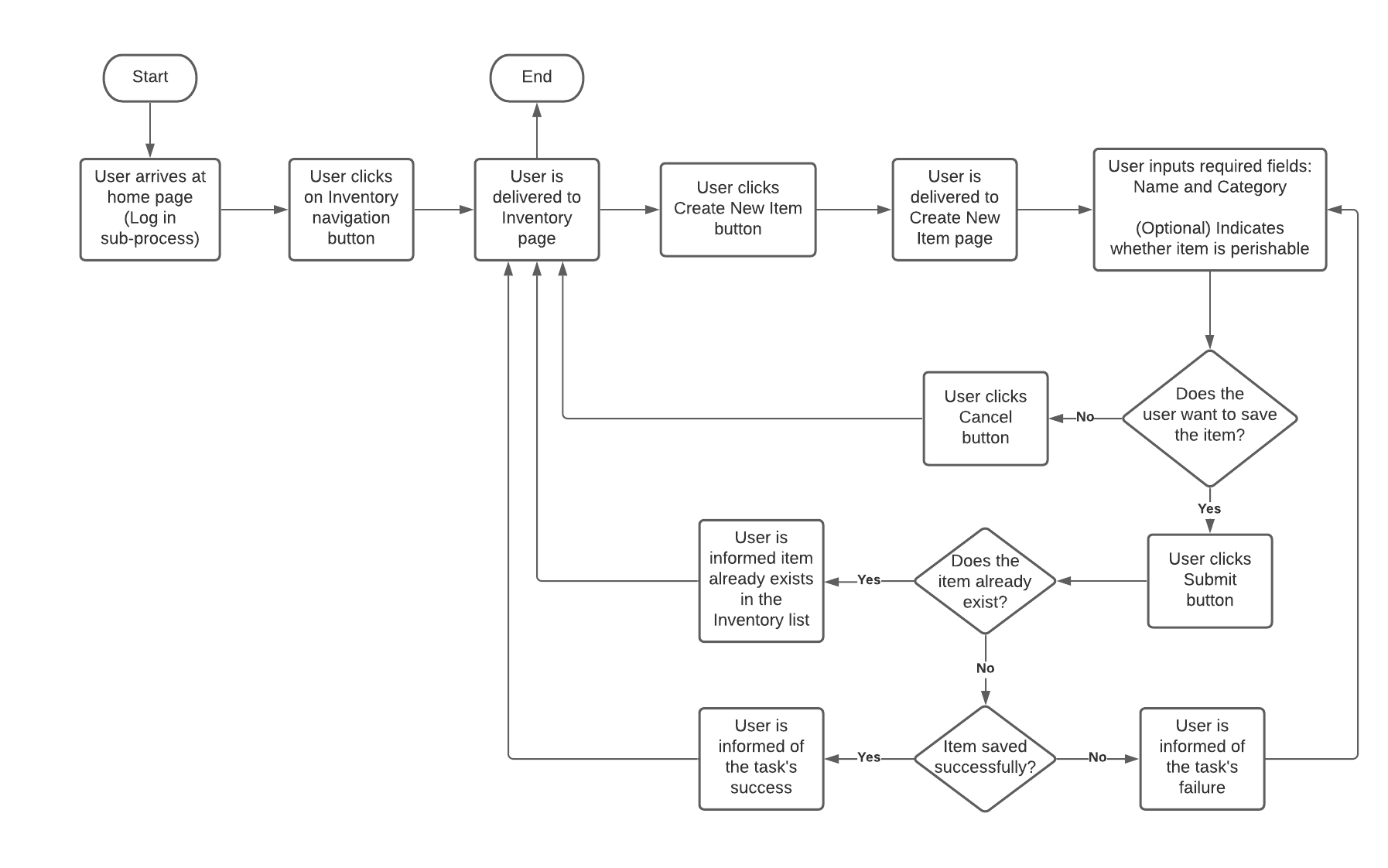
## Log in sub-process



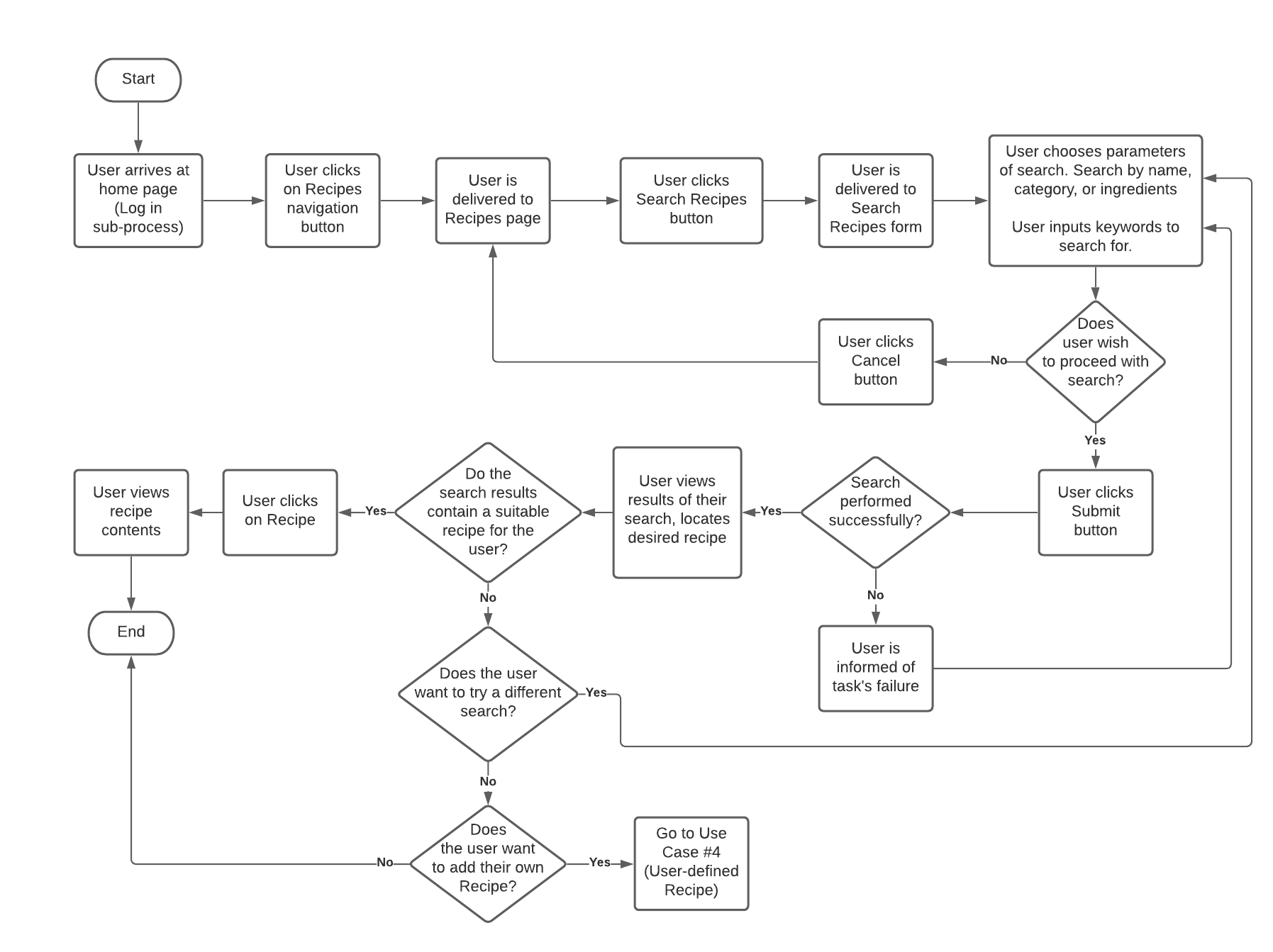
## Use Case #1: Add Item



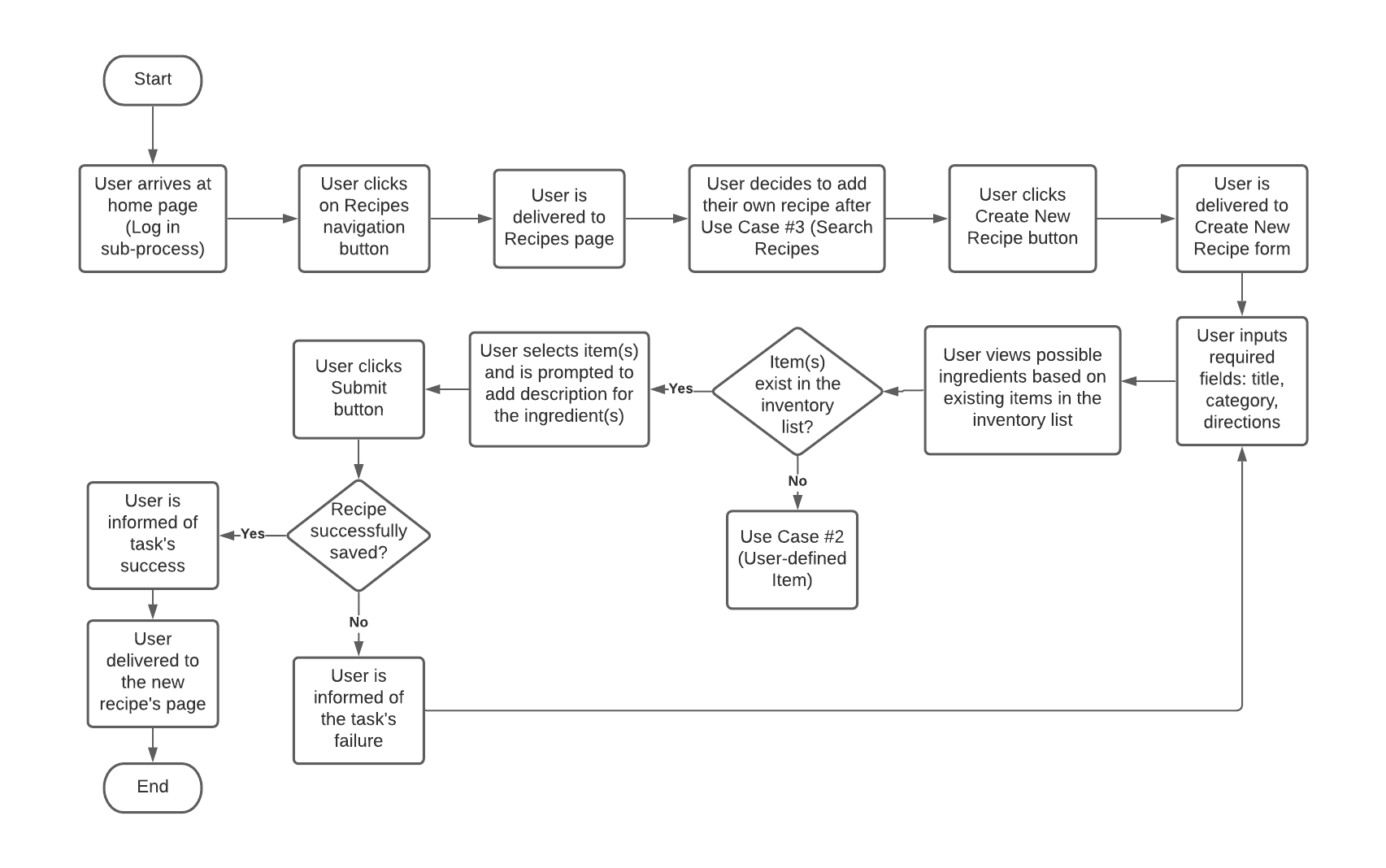
## Use Case # 2: Create User-defined Item



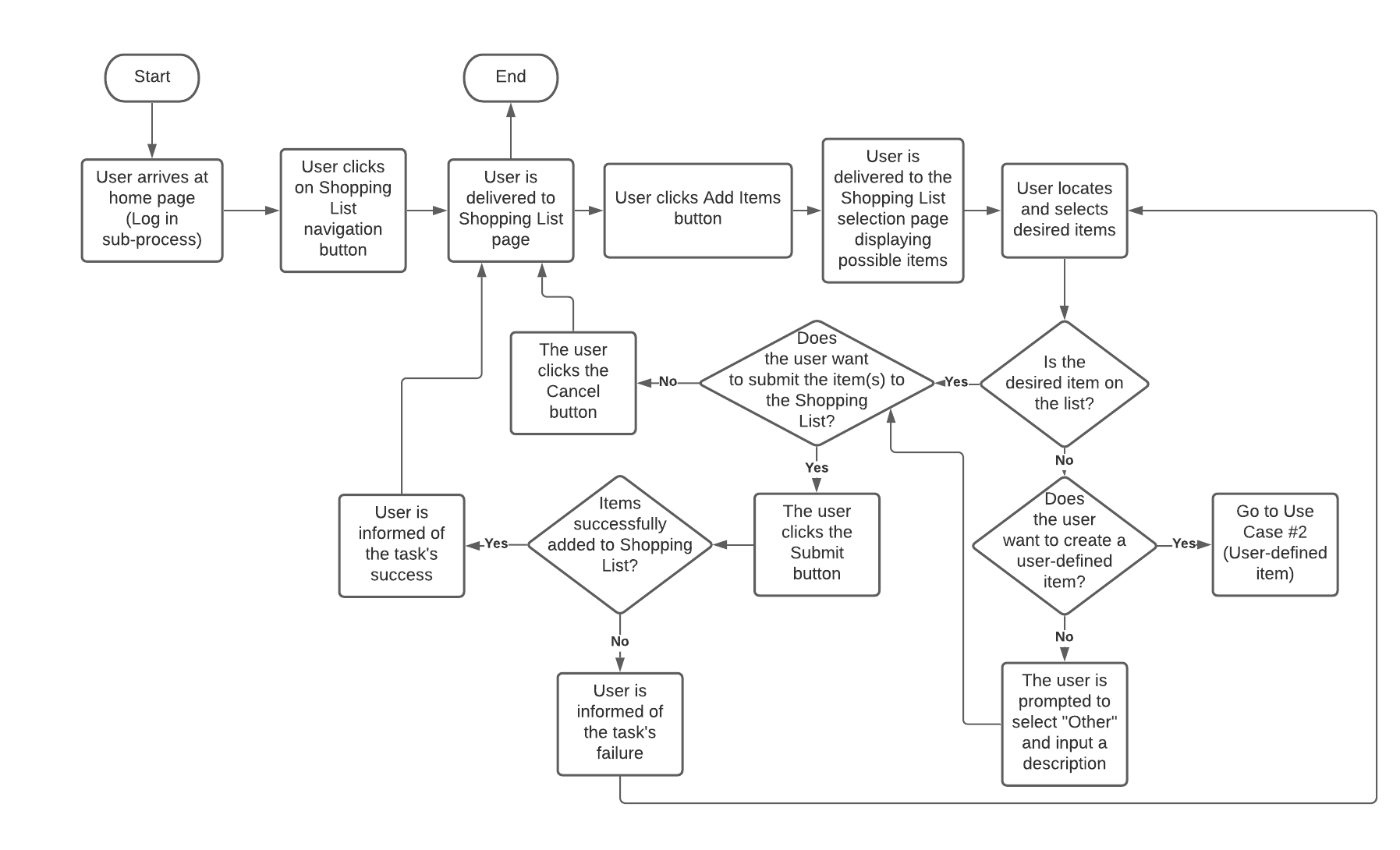
## Use Case #3: Search Recipes



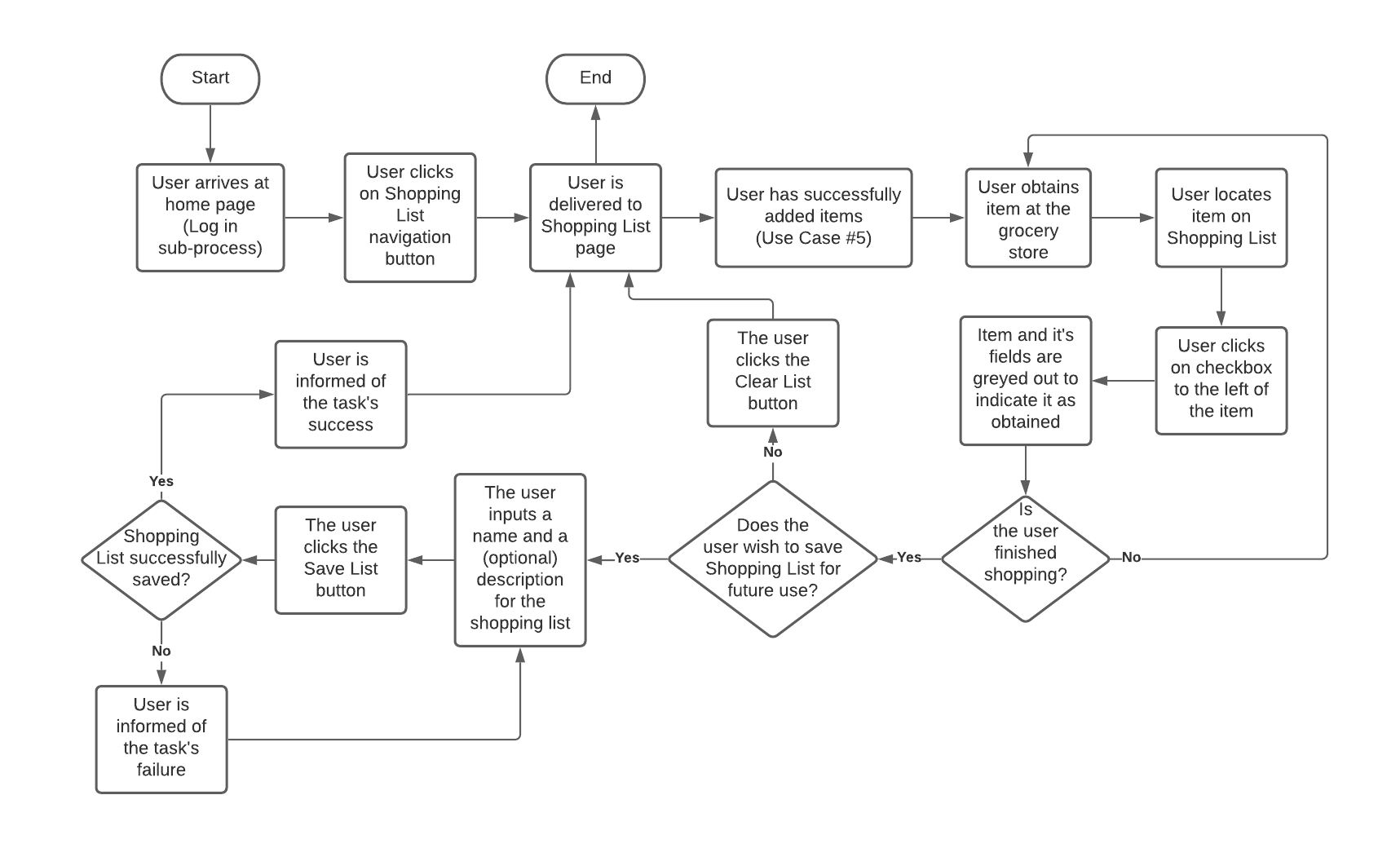
## Use Case #4: User-Defined Recipe



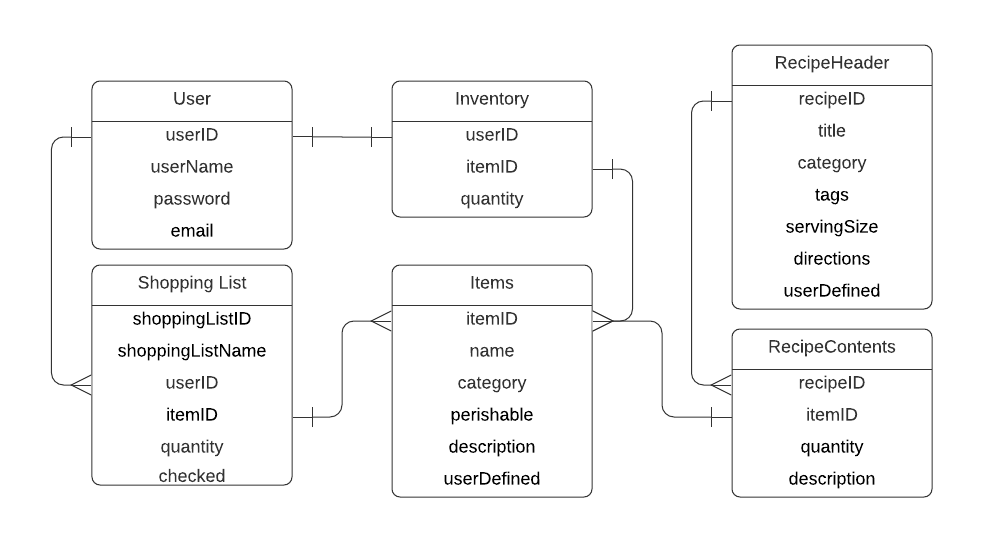
## Use Case #5: Add Items to Shopping List



## Use Case #6: Crossing Item Off Shopping List

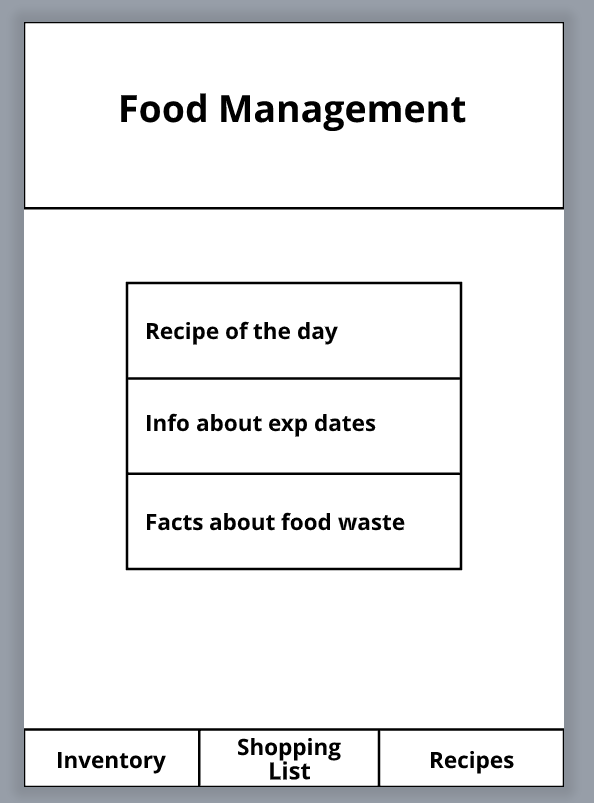


# Data Model

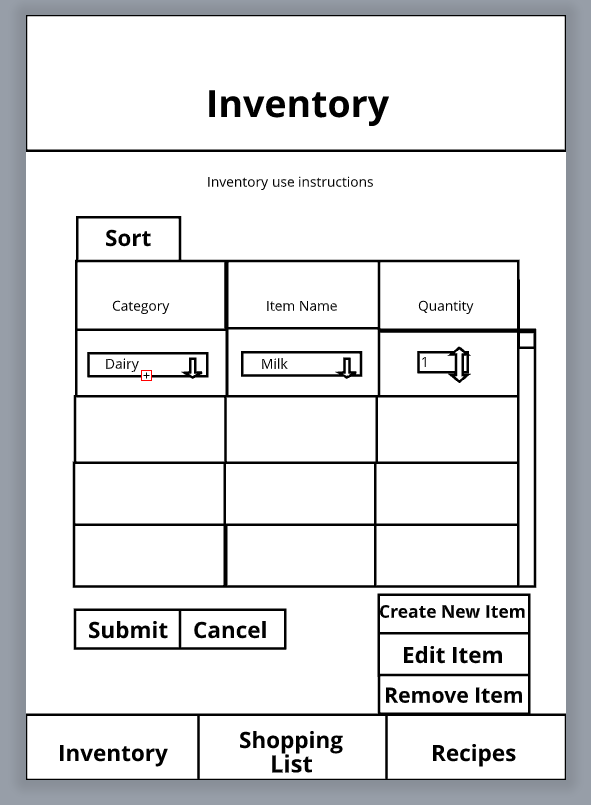


# User Interface

## Home Page



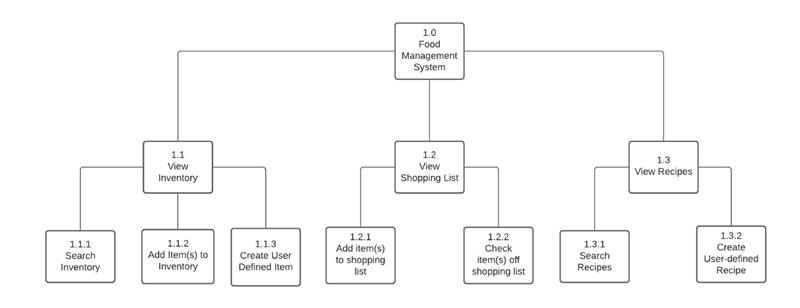
## Inventory page



## Shopping List Page

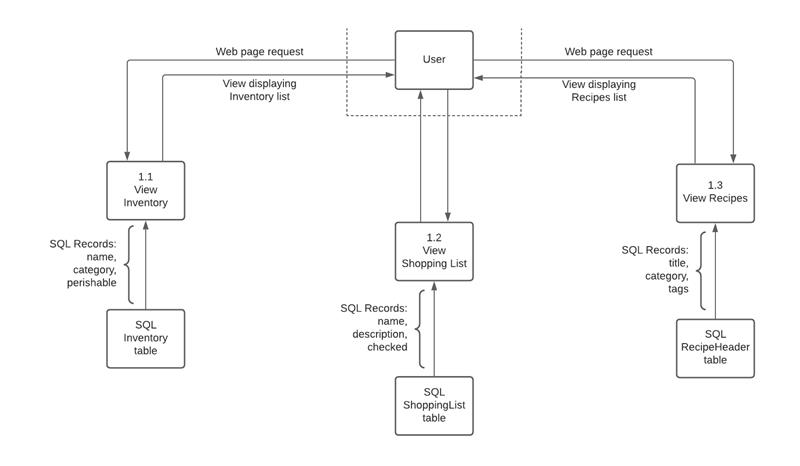


# Structure Chart

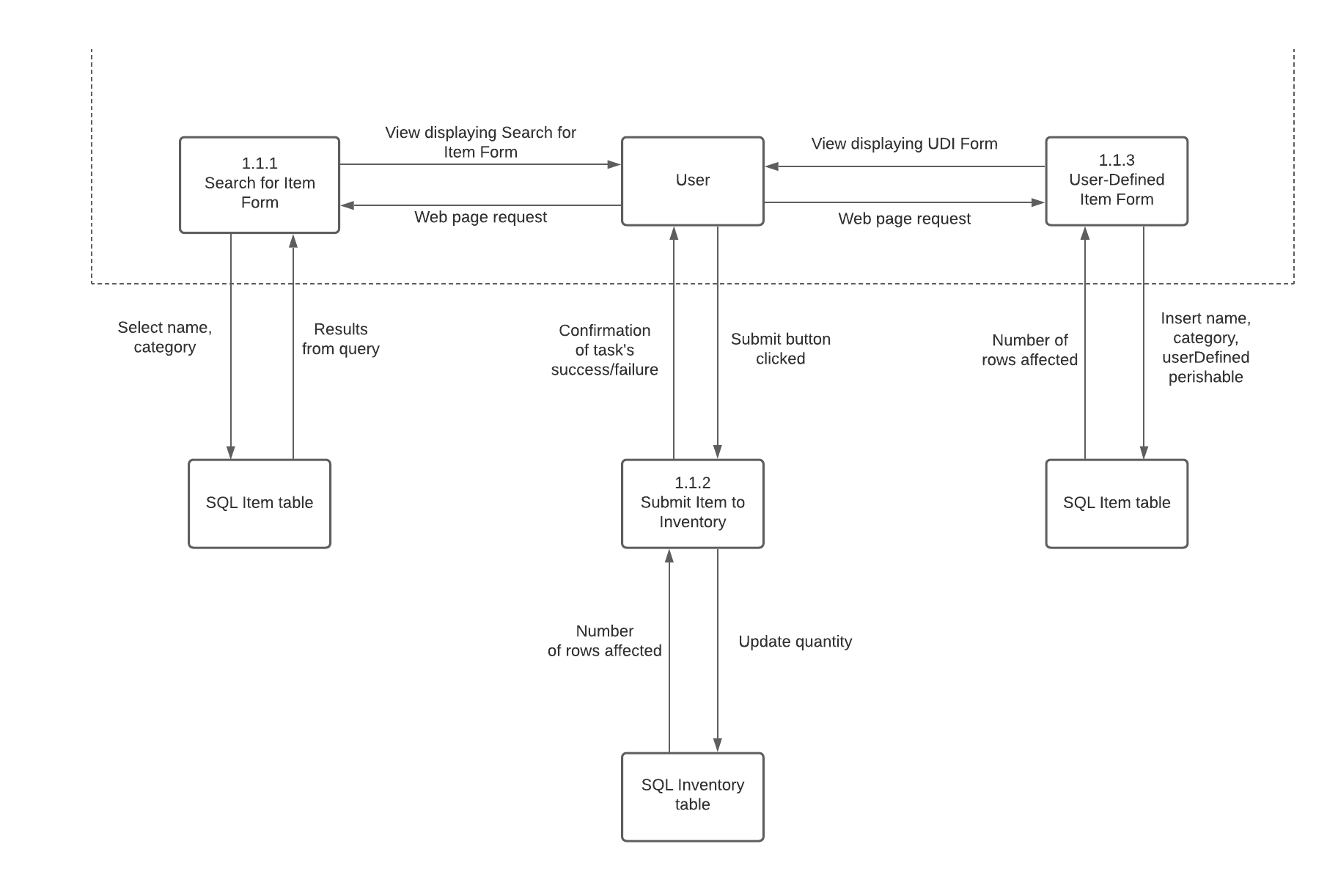


# Physical Data Flow Diagram

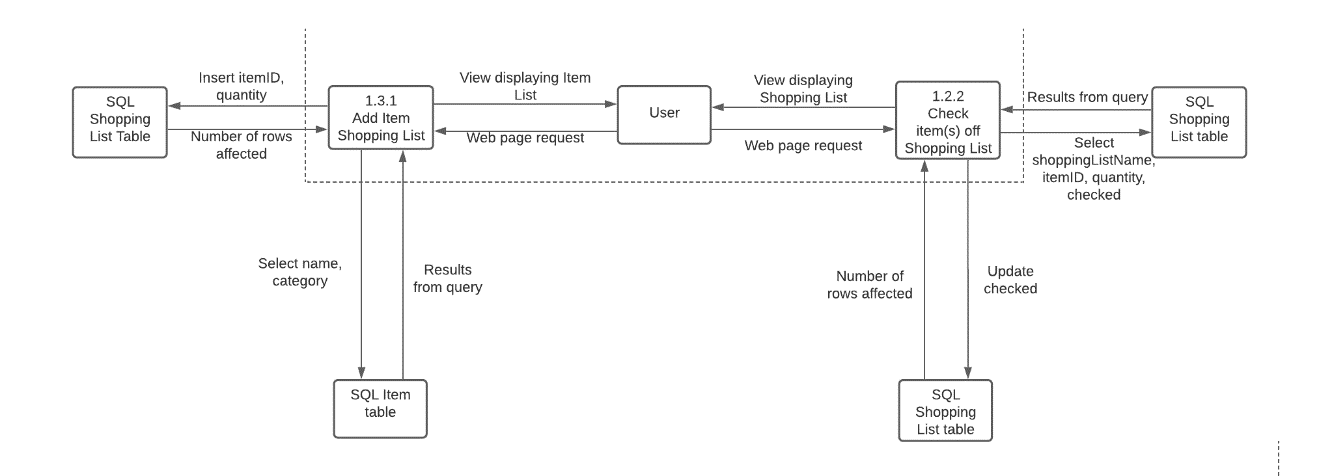
## 1.0 Navigation



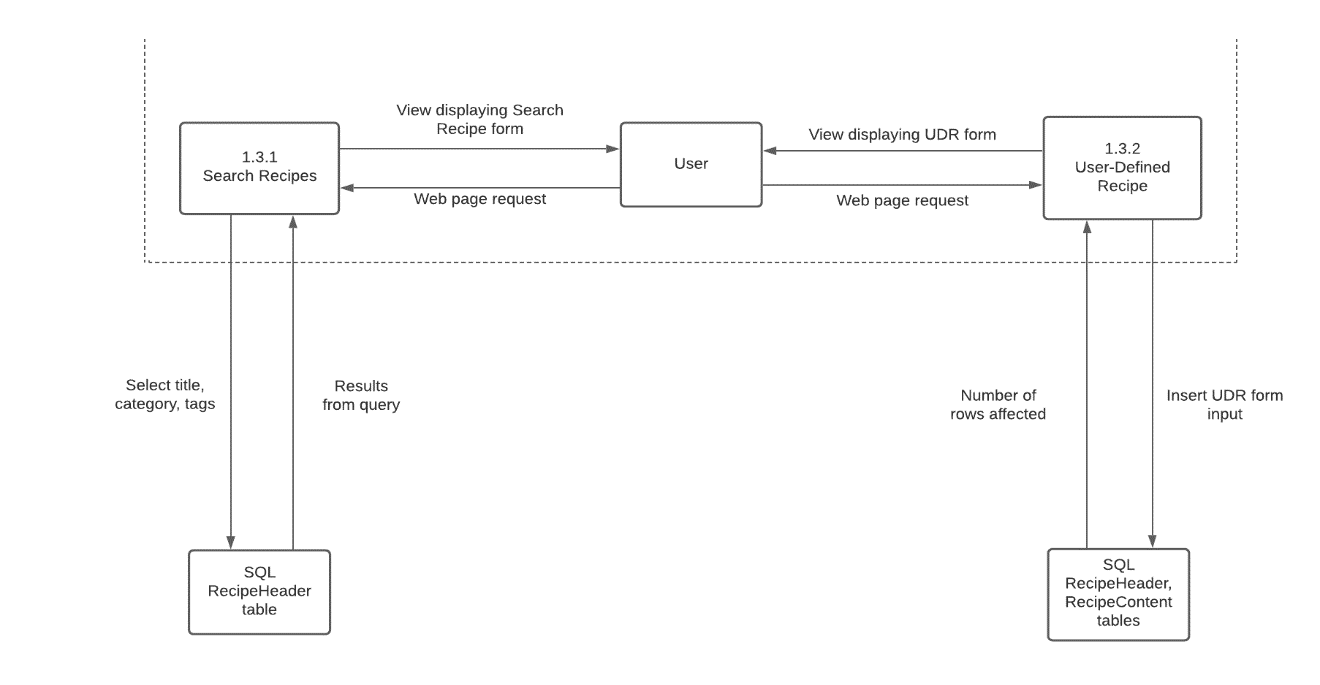
## 1.1 Inventory



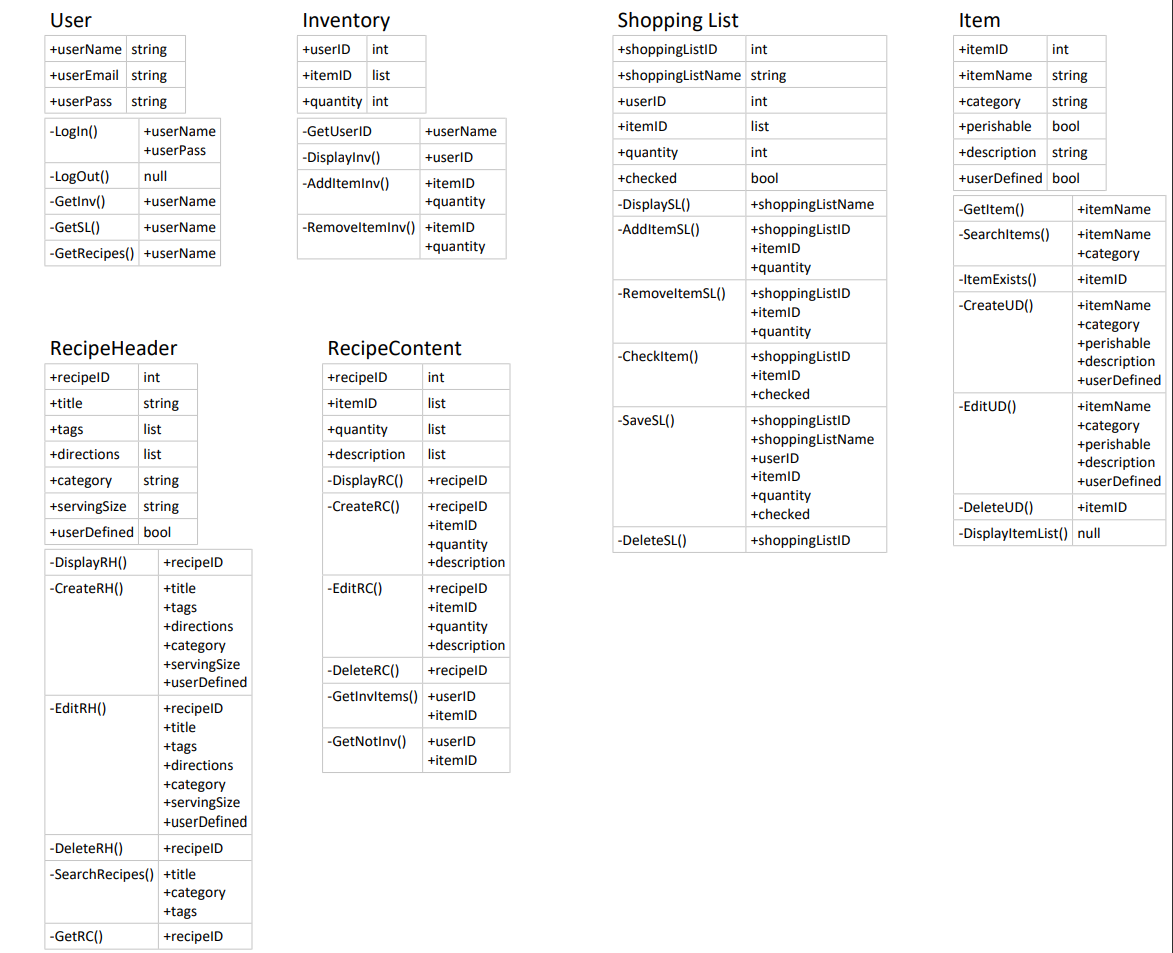
## 1.2 Shopping List



## 1.3 Recipe



# Class Diagram



# Program Specification

## Module: 1.1.2

**Name:** Add Item

**Purpose:** Update the user’s inventory and determine if the task was successful or not.

**Events:** User clicks on the “Submit” button on the Inventory page.

|  |  |  |
| --- | --- | --- |
| **Input name:** | **Type:** | **Notes:** |
| name | string |  |
| quantity | int |  |

|  |  |  |
| --- | --- | --- |
| **Output name:** | **Type:** | **Notes:** |
| saveSuccess | bool | Indicates whether the changes were saved successfully. |

**Pseudocode:**

saveSuccess = false

For each record in Inventory

If user’s Inventory name = Inventory name in database

Update quantity with the amount input by user

If query executed successfully

saveSuccess = true

End If

End If

End For

Return

## Module: 1.1.3

**Name:** Add User-Defined Item

**Purpose:** Update the user’s inventory with an item defined by the user and determine if the task was successful or not.

**Events:** User clicks on the “Submit” button on the Create New Item page.

|  |  |  |
| --- | --- | --- |
| **Input name:** | **Type:** | **Notes:** |
| name | string |  |
| category | string |  |
| userDefined | bool | Always true |
| perishable | bool | Optional input |

|  |  |  |
| --- | --- | --- |
| **Output name:** | **Type:** | **Notes:** |
| itemExists | bool | Indicates if the item already exists or not. |
| saveSuccess | bool | Indicates whether the changes were saved successfully. |

**Pseudocode:**

itemExists = true

saveSuccess = false

Select from Items where name is equal to input name

If a row is not returned

itemExists = false

Insert Item name, category, userDefined and perishable value

Select name where name equals input name

If query returns a row

saveSuccess = true

End If

End If

Return

## Module: 1.2.1

**Name:** Add item to Shopping List

**Purpose:** Update the Shopping List with items added by the user.

**Events:** User clicks “Submit” on the Shopping List item selection page.

|  |  |  |
| --- | --- | --- |
| **Input name:** | **Type:** | **Notes:** |
| itemID | int |  |
| shoppingListID | int |  |
| quantity | int |  |

|  |  |  |
| --- | --- | --- |
| **Output name:** | **Type:** | **Notes:** |
| saveSuccess | bool | Indicates whether the changes were saved successfully. |

**Pseudocode:**

saveSuccess = false

If itemID And shoppingListID Exist

Update itemID list with input itemID, quantity with input quantity

Where shoppingListID = input shoppingListID

If query returns a row changed

saveSuccess = true

End If

End If

Return

## Module: 1.2.2

**Name:** Check item(s) off shopping list

**Purpose:** Indicate item has been procured by the user while shopping

**Events:** User clicks check box next to item’s name on the shopping list

|  |  |  |
| --- | --- | --- |
| **Input name:** | **Type:** | **Notes:** |
| shoppingListID | int |  |
| itemID | int |  |
| checked | bool |  |

|  |  |  |
| --- | --- | --- |
| **Output name:** | **Type:** | **Notes:** |
| checked | bool |  |

**Pseudocode:**

checked = false

if itemID and shoppingListID exist

checked = true

Update checked where itemID and shoppingListID = input itemID and input shoppingListID

If query returns a row changed

Row becomes read-only

End if

End if

Return

## Module: 1.3.1

**Name:** Search Recipes

**Purpose:** Search for recipes based on title, category, and tags

**Events:** User clicks “Submit” button on Search Recipes form

|  |  |  |
| --- | --- | --- |
| **Input name:** | **Type:** | **Notes:** |
| title | string |  |
| category | string | Optional input |
| tags | string | Optional input |

|  |  |  |
| --- | --- | --- |
| **Output name:** | **Type:** | **Notes:** |
| recipeResults | object | Contains results from query |

**Pseudocode:**

recipeResults = null

If title field has text

If category field has text

If tags field has text

Select all recipeID from RecipeHeader that contain title, category, and tags input values

If rows returned

recipeResults = query results

End If

End If

Select all recipeID from RecipeHeader that contain title and category input values

If rows returned

recipeResults = query results

End If

End If

Select all recipeID from RecipeHeader that contain title input value

If rows returned

recipeResults = query results

End If

End If

Return

## Module: 1.3.2

**Name:** Create User-defined Recipe

**Purpose:** Update the RecipeHeader and RecipeContent database with Recipe defined by the user.

**Events:** User clicks on the “Submit” button on the Create Recipe form

|  |  |  |
| --- | --- | --- |
| **Input name:** | **Type:** | **Notes:** |
| title | string | Header |
| tags | list | Header |
| directions | list | Header |
| category | string | Header |
| servingSize | string | Header |
| userDefined | bool | Header - Always true |
| itemID | list | Content |
| quantity | list | Content |
| description | list | Content |

|  |  |  |
| --- | --- | --- |
| **Output name:** | **Type:** | **Notes:** |
| saveSuccess | bool | Indicates whether the database is successfully updated |

**Pseudocode:**

saveSuccess = false

If title is not equal to any RecipeHeader title in the database

Insert all Header input into RecipeHeader table

Select recipeID from RecipeHeader where title is equal to input title

If a row is returned

recipeID = query result

Insert recipeID and all Content input into RecipeContent table

Select recipeID from RecipeContent where recipeID is equal to recipeID

If a row is returned

saveSuccess = true

End If

End If

End If

Return

# Appendices

[1]Food Waste Study by William & Mary’s Department of Kinesiology & Health Sciences

* <https://nutritionj.biomedcentral.com/articles/10.1186/s12937-020-00552-w>

[2]FDA How to Cut Food Waste and Maintain Food Safety

* <https://www.fda.gov/food/consumers/how-cut-food-waste-and-maintain-food-safety>