BCGL: Barcode Grocery List

Design Document 2
(Detailed Design Document)

Final Draft v2.0

March 29, 2021

RAID Inc.

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Rough Draft: Version 1.0	Bailey Martin, Andrew Moore, Kyle Hieb	*BCGLDesignDocument2.docx* provides detailed design information on four of RAID Inc.'s proposed Barcode Grocery List (BCGL) software components and modules.	3/22/21
Final Draft: Version 2.0	Bailey Martin	*BCGLDesignDocument2.docx* provides detailed design information on four of RAID Inc.'s proposed Barcode Grocery List (BCGL) software components and modules.	3/27/21

Contents

1 Introduction

1.1 Overview

This detailed design document will be used to outline the four major software modules that are found within the Barcode Grocery List (BCGL) software. This document discusses the classification and purpose of each module, as well as their constraints, interactions with other modules, method of processing, and data components of each item.

1.2 Modules to Be Discussed

The modules that will be discussed in this design document are:

- Scanning a barcode
- Searching for a SKU number
- Creating a shopping list
- Storing product information in the shopping list

2 Module 1: Scanning a Barcode

2.1 Classification and Purpose

This software module is classified as a function because it is something that is essential to the operation of the program. It serves a single purpose, which is to simplify the addition of items to a user-created shopping list.

2.2 Constraints and Interactions

This module will draw most of its computational function from the "searching a SKU number" function and will be constrained by the API that it interacts with. If a barcode or SKU is not contained in the API, then it will not perform the way we intend. When a UPC barcode is scanned, it will call the same process as the search function, which looks up the SKU number and product information. It does this by automatically inputting the SKU number into the search, rather than having the user do so.

2.3 Processes and Data Components (Pseudocode Outline)

```
/ * Function should call cameraImageInput functionality as a
    parameter
  *   The cameraImageInput type is a camera view that is
      embedded within the application. The picture taken
      (containing a product's barcode) will be named
      pictureInput and is passed into this function
  *
  * Function returns void
  * /
// Check to see that the barcode is found within the API and
will thus return productInfo
IF ( barcode is detected by device camera/pictureInput )
(cameraImageInput pictureInput) {
      runSearchFunction(); //NOTE: see runSearchFunction()
      pseudocode in Section 3
} //end of if
ELSE {
      Display ERROR message: "Barcode not detected"
} //end of else
```

3 Method 2: Searching for a SKU Number

3.1 Classification and Purpose

This module is classified as a function because it serves a single, basic purpose. The role of this function is to allow the user to manually search for a SKU number, given that they cannot use the barcode scanner. This manual search functionality would be useful in the event that a user's camera will not detect a product's barcode due to poor lighting conditions or if the user's smartphone or tablet is not equipped with a camera.

3.2 Constraints and Interactions

This class will be interacting with the barcode scan function, as this is the class that is called within that function. The barcode scan is constrained by the API, as a product missing from the API will not return any data for the output.

3.3 Processes and Data Components (Pseudocode Outline)

```
/ * Function has 2 parameters: SKU and API
  *   Function should call the product information API as a
      parameter
  *   Calling the API as a parameter will encapsulate the
          entire API database, so all products can be checked
          against the SKU string entered by the user
  *   The second parameter, SKU, will be a string and is the
          SKU/barcode number that was manually entered into
          the search bar by the user
  *   If the product is found in the API, then SKU =
          API.productSKU, return that specific product's
          information
  *
  * Function returns productInfo
  * /
SKU = manualSearchTextField.getText( );
IF ( SKU is found in the API ) {
      productInfo equals specified product data from the API
} //end of if
ELSE {
      productInfo equals "Product is not found"
} //end of else
```

```
return productInfo;
```

4 Module 3: Creating a List

4.1 Classification and Purpose

The list creation module is classified as a method. This module is designed to create a new shopping list for a user. It is important to note that users can create multiple shopping lists, so this function can be called at any point during program execution. An "Add Shopping List" or "+" button will trigger execution of this component. This is necessary for the program, as it allows users to create lists for multiple occasions, and to easily create a blank shopping list without overwriting an existing one.

4.2 Constraints and Interactions

This module will work closely with the store product information module, as it will be where that information is displayed. When the user accesses their list, they will see the data displayed there. Therefore, we will need to call the data to be displayed into the view. Also, it will be constrained by the store function, as it will only be allowed to display information that has been saved by the user.

4.3 Processes and Data Components (Pseudocode Outline)

```
/ * This method creates a new shopping list. The user will
enter the desired list name in a pop-up text box, and its text
will be passed as a parameter to this method.

  * addANewList() function returns void and accepts a
parameter that is the toString of addListPopupTextField

  * /

// Declare a list item here, which is a String

      Action ( when Add List Button is clicked ) {

            New List Name is obtained from the input in the
             toString();

      } //end of actionEvent

      //create a new list with title

      createNewList  ( newListName ) that was entered in the
      popup window;


/ * createNewList function also returns void, and accepts a
String parameter listName

      New String tempListName will be set equal to listName

      Create a new list of tempListNames
```

```
        Add this new list to the ListOfLists, which is where all
        of the shopping lists are stored.
} //end of createNewList()
```

5 Module 4: Storing Product Information in List

5.1 Classification and Purpose

This component of the BCGL system is identified as a function. This function is responsible for storing scanned (or searched) products in the user's selected shopping list. This functionality is key to the general design of the program, as it allows users to save their searched items into a shopping list. This functionality provides the foundation for stored shopping lists. Without this function, users would be unable to view multiple products simultaneously and would only have the ability to view product information immediately after scanning an item's barcode.

5.2 Constraints and Interactions

As discussed above, this module will be dealing with the list function, as it will be called by the list to display saved information. It will also interact with the search function, as it will save a link to the output of that function. We will not be storing product information locally. Therefore, when the user clicks on an item in their list, it will call the search function and create a new search request to the API. It will be constrained by the search because it can only store information that the search outputs.

5.3 Processes and Data Components (Pseudocode Outline)

```
/ * Function has 2 parameters: item and shoppingListName
  *   Function should be passed the item as a parameter
  *   The second parameter, shoppingListName, will be a string
        and is the name of the shopping list that we want
        to add or remove an item from
  *   If the product is found in the product API, then item is
        added or removed from the list shoppingListName
  *
  * Function returns void
  * /


When adding an item to a list:
      If (item is found and retrieved from search function) {
            Create link to item from API on the list
            Add item to shoppingListName
      }
When deleting an item from a list (or marking it as
completed/purchased):
```

```
If (an item link exists on a shopping list) {
      Prompt with confirmation to delete selected item.
      Remove link from the API to the item from the list
      Remove item from shoppingListName
}
```