

Food Storage Inventory Manager

Executive Summary

The Problem

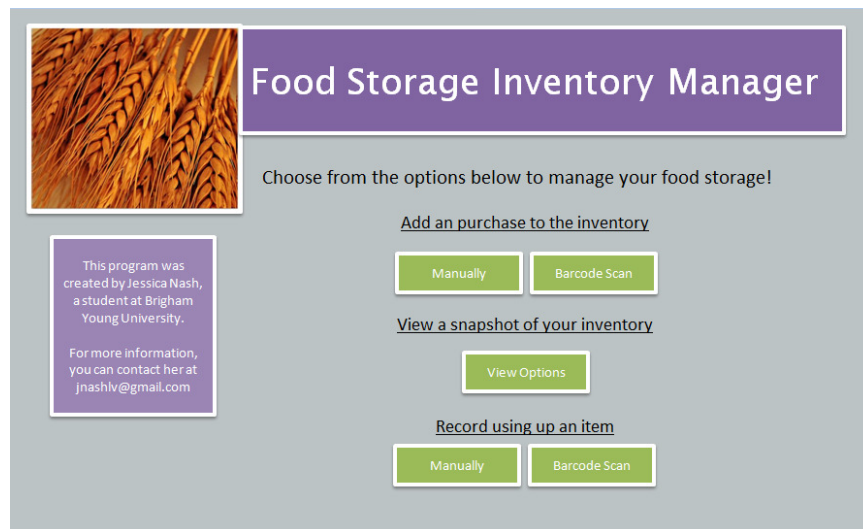
Whether it's a one year supply of food or just a well stocked pantry, it can be difficult to keep track of the expiration dates of stored food. Letting food expire is not only frustrating, it wastes both time and money. Keeping track of expiration dates by hand can be repetitive and boring—meaning it just won't get done.

The Solution

The Food Storage Inventory Manager is a simple and easy to use tool for keeping track of your food storage. A collection of user forms allows even the least computer savvy user to add purchases, sort the inventory by either item type or expiration date, and record when an item is used up. The inventory manager can be updated either manually or with a barcode scanner (CueCat), which means the novelty of the system just might last longer than that clipboard.

Implementation

The user begins at the Welcome page, where they can choose between three actions: add a purchase, view an inventory snapshot, or record the use of an item. The entire program is designed to be highly user-friendly so that anyone without experience with Excel can use it.



Add a purchase to the inventory

The user begins by entering purchases into the system. The user can choose to do this manually, or by using a CueCat barcode scanner. The programming for both methods was pretty similar, the differences with the CueCat method will be described later.

New purchases can be recorded by using the form shown below. In order to maintain comparability with multiple purchases of the same product, this inventory system uses an Item

Key as a master list of possible items. If the user is entering the purchase of a new kind of item, they can select the “Add new item” button (circled).

Add Purchase

Select the item you added to your food storage from the drop down box.

[Dropdown Menu]

Date canned/purchased [Text Field]

Number of units [Text Field]

Cost per unit [Text Field]

Expiration date [Text Field]

Notes [Text Field]

[OK] [Add new item] [Cancel]

This brings up the “New Item” form. A combo box is populated from the item types column in the item key sheet, and the user can fill in the remaining information. Selecting “OK” adds the item to the Item Key (shown below the form) and returns the user to the Add Purchase userform.

New Item

Type of item [Store bought]

Item Name [Text Field]

Expected Life [Text Field]

Units [Text Field]

Notes [Text Field]

ex. 16 oz cans

[OK] [Cancel]

Type	Bar code #	Item Name	Expected Life	Notes	Units	Types
Store bought		Bread Flour	5 years		10 lb bags	White 45 lbs Bucket
Store bought		C&H Sugar	5 years	Stored in bins	10 lb bags	#10 Cannery Can
#10 Cannery Can		Chopped Dried Onion	10 years		Cans	Store bought
Store bought		Honey	3 years		3 lb bottles	Scanned
#10 Cannery Can		Hot Cocoa Mix	10 years		Cans	
#10 Cannery Can		Macaroni	30+ years		Cans	
Store bought		Peanut Butter	10 months		48 oz jars	
#10 Cannery Can		Rollod Oats	30+ years		Cans	
Store bought		Spaghetti Sauce	1 year		jars	
White 45 lbs Bucket		Wheat	30+ years		Bucket	
#10 Cannery Can		White Rice	30+ years	Buy less next time	Cans	
#10 Cannery Can		White Sugar	30+ years		Cans	
Scanned	19900003202	Clabber Girl Baking Powder		10 OZ. (284g)		

The user begins to add an item by selecting the item name from the combo box (circled below). If the user first adds a new item, it will already be available in the combo box. After filling out

all the information, selecting “OK” puts the values input by the user into the inventory sheet, and the inventory sheet is activated with the new addition selected.

A	B	C	D	E	F	G	H	I	J
Type	Item Name	Bar Code	Number Units	Notes	# of units	Date Canned/purchased	Cost/Unit	Expiration Date	
#10 Cannery Can	Macaroni		Cans		4	2/5/2010	\$1.03	2/5/2015	
Store bought	Peanut Butter		48 oz jars		10	3/1/2009	\$7.00	5/1/2010	
White 45 lbs Bucket	Wheat		Bucket		5	2/3/2004	\$6.80	2/3/2034	
#10 Cannery Can	White Rice		Cans	Don't buy more for awhile	4	3/1/2001	\$3.50	3/1/2031	
Scanned	Clabber Girl Baking Powder	19900003202	10 OZ. (284g)		3	4/1/2008	\$2.80	5/2/2011	
#10 Cannery Can	Chopped Dried Onion		Cans		4	3/1/2008	\$2.50	5/1/2012	

The code for this action is fairly simple, first the next available row in the inventory is located:

```
Private Sub cmdOK_Click()
    Dim x As Integer

    'find the next available row
    x = 2
    Do Until Sheets("Inventory").Cells(x, 2).Value = ""
        x = x + 1
    Loop
```

Then the item type and units is looked up on the item key page:

```
'find the type
Sheets("Item Key").Activate
Cells.Find(What:=cboItem.Value, after:=Range("a1"), LookIn:=xlFormulas, _
    LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
    MatchCase:=False, SearchFormat:=False).Activate

ActiveCell.Offset(0, -2).Select
Selection.Copy
Sheets("Inventory").Activate
Cells(x, 1).Select
ActiveSheet.Paste

'find the units
Sheets("Item Key").Activate
Cells.Find(What:=cboItem.Value, after:=Range("a1"), LookIn:=xlFormulas, _
    LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
    MatchCase:=False, SearchFormat:=False).Activate

ActiveCell.Offset(0, 3).Select
Selection.Copy
Sheets("Inventory").Activate
Cells(x, 4).Select
ActiveSheet.Paste
Application.CutCopyMode = False
```

Finally, the fields entered by the user are copied into the form and the new row is selected:

```
'copy in from form
Sheets("Inventory").Cells(x, 2).Value = cboItem.Value
Sheets("Inventory").Cells(x, 7).Value = txtPurchdate.Value
Sheets("Inventory").Cells(x, 6).Value = txtNumunits.Value
Sheets("Inventory").Cells(x, 8).Value = txtUnitcost.Value
Sheets("Inventory").Cells(x, 9).Value = txtExpiration.Value
Sheets("Inventory").Cells(x, 5).Value = txtNotes.Value

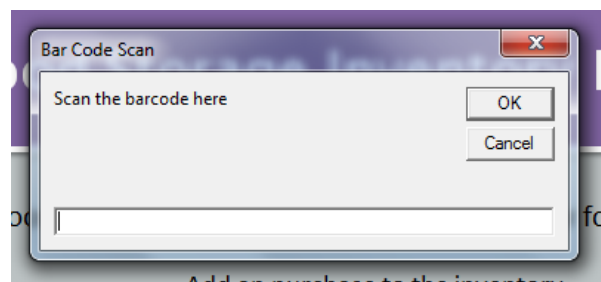
Sheets("Inventory").Activate
Rows(x).Select

Unload Me

End Sub
```

Add a purchase using the CueCat

To use the CueCat, the user is first prompted to scan the barcode into an input box:



Once the scan is completed, the scan is converted into a barcode by using the cuecat function given in class. Then a web query is run to look up the item name and unit information at upcdatabase.com. This is done by creating a “lookUpitem” subprocedure, shown below.

```

Sub lookUpitem(barcode As String)

ActiveWorkbook.Worksheets.Add
    With ActiveSheet.QueryTables.Add(Connection:= _
        "URL;http://www.upcdatabase.com/item/0" & barcode, Destination:=Range("A1"))
        .FieldNames = True
        .RowNumbers = False
        .FillAdjacentFormulas = False
        .PreserveFormatting = True
        .RefreshOnFileOpen = False
        .BackgroundQuery = True
        .RefreshStyle = xlInsertDeleteCells
        .SavePassword = False
        .SaveData = True
        .AdjustColumnWidth = True
        .RefreshPeriod = 0
        .WebSelectionType = xlAllTables
        .WebFormatting = xlWebFormattingNone
        .WebPreFormattedTextToColumns = True
        .WebConsecutiveDelimitersAsOne = True
        .WebSingleBlockTextImport = False
        .WebDisableDateRecognition = False
        .WebDisableRedirections = False
        .Refresh BackgroundQuery:=False
    End With

    Cells.Find(What:="Description", after:=ActiveCell, LookIn:=xlFormulas, _
        LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
        MatchCase:=False, SearchFormat:=False).Activate
    ActiveCell.Offset(0, 2).Activate
    itemName = ActiveCell.Value
    Cells.Find(What:="size", after:=ActiveCell, LookIn:=xlFormulas, _
        LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
        MatchCase:=False, SearchFormat:=False).Activate
    ActiveCell.Offset(0, 2).Activate
    itemSize = ActiveCell.Value
End Sub

```

The sub is passed the barcode number by the userform_initialize subprocedure, shown below. Although creating the web query was not too difficult, it was tricky to determine where it should go so as to not confuse the user. This subprocedure determines the barcode number, turns off screen updates, runs the web query (which also finds the itemName and itemSize variables), deletes the webquery sheet, and copies the name, units, and barcode into the form.

```

Private Sub UserForm_Initialize()

barcode = cueCat(scan, 3)

Application.ScreenUpdating = False

lookUpitem (barcode)

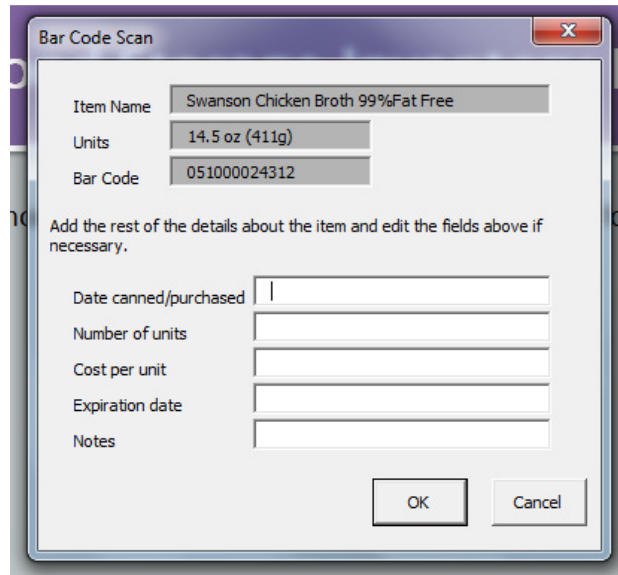
'delete the query page
Application.DisplayAlerts = False
ActiveSheet.Delete
Application.DisplayAlerts = True
Application.ScreenUpdating = True

txtName.Value = itemName
txtUnits.Value = itemSize
txtBarcode.Value = barcode

End Sub

```

The resulting userform is very similar to the manual method user form for entering a purchase, except the Item Name, Units, and Bar Code boxes are already populated.

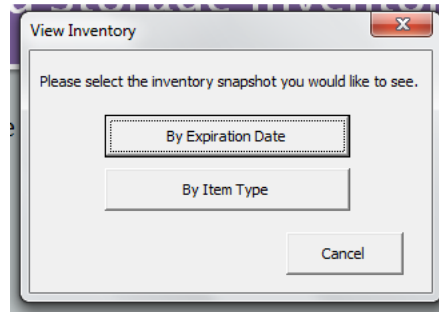


The user can fill in the remaining fields, and edit the already populated fields as necessary. Selecting “OK” does the same as manual method, with the exception of searching the item key to see if this is a new type of item, and adding it to the item key list if it is. This is done by the following code:

```
Private Sub cmdOK_Click()  
Dim y As Integer  
Dim found As String  
Dim nextKeyline As Integer  
  
'add it to the item key if unique  
found = "no"  
y = 2  
Do Until Sheets("Item Key").Cells(y, 3) = ""  
    If Sheets("item Key").Cells(y, 3).Value = itemName Then  
        found = "yes"  
        y = y + 1  
    Else  
        y = y + 1  
    End If  
Loop  
  
'if not already there, add to the item key  
If found = "no" Then  
  
    nextKeyline = 2  
    Do Until Sheets("item key").Cells(nextKeyline, 3) = ""  
        nextKeyline = nextKeyline + 1  
    Loop  
  
    Sheets("Item key").Cells(nextKeyline, 1).Value = "Scanned"  
    Sheets("Item key").Cells(nextKeyline, 2).Value = barcode  
    Sheets("Item key").Cells(nextKeyline, 3).Value = itemName  
    Sheets("Item key").Cells(nextKeyline, 6).Value = itemSize  
End If
```

View the inventory snapshot

At any time the user can view in the inventory currently entered in the system. After choosing “view options” the user can choose the type of view they would like to see. Choosing either of these options performs a sort of the inventory and takes the user to the inventory page.



For example, choosing to view by item type sorts the inventory list first by item type, then by expiration date. Below is the code that accomplishes this. The sort code for sorting by expiration date is very similar.

```
Private Sub cmdItemtype_Click()

Unload Me

Sheets("Inventory").Select
Range("b1").Select
ActiveWorkbook.Worksheets("Inventory").Sort.SortFields.Clear
ActiveWorkbook.Worksheets("Inventory").Sort.SortFields.Add Key:=Range(ActiveCell, ActiveCell.End(xlDown)),
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
Range("i1").Select
ActiveWorkbook.Worksheets("Inventory").Sort.SortFields.Add Key:=Range(ActiveCell, ActiveCell.End(xlDown)),
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
Range("a1").Select
With ActiveWorkbook.Worksheets("Inventory").Sort
    'make sure below refers to the last column in the table
    .SetRange Range(ActiveCell, Range("i1").End(xlDown))
    .Header = xlYes
    .MatchCase = False
    .Orientation = xlTopToBottom
    .SortMethod = xlPinYin
    .Apply
End With

Range("b2").Select

End Sub
```

Here is a screenshot of the inventory sorted by item type (and then by expiration date).

Type	Item Name	Bar Code	Number	Units	Notes	# of units	Date Canned/purchased	Cost/Unit	Expiration Date
Scanned	Campbell's Condensed Cream of Mushroom Soup	51000012616	10	3/4 OZ		5	4/6/2006	\$3.50	2/5/2025
#10 Cannery Can	Chopped Dried Onion			Cans		4	3/1/2008	\$2.50	5/1/2012
#10 Cannery Can	Chopped Dried Onion			Cans		8	6/1/1999	\$6.00	5/8/2020
Scanned	Clabber Girl Baking Powder	19900003202	10	OZ. (284g)		3	4/1/2008	\$2.80	5/2/2011
#10 Cannery Can	Macaroni			Cans		4	2/5/2010	\$1.03	2/5/2015
Store bought	Peanut Butter			48 oz jars		9	3/1/2009	\$7.00	5/1/2010
Scanned	Swanson Chicken Broth 99%Fat Free	51000024312	14.5	oz (411g)		50	5/2/1999	\$4.50	5/2/2003
Scanned	Swanson Chicken Broth 99%Fat Free	51000024312	14.5	oz (411g)		5	12/7/2010	\$0.99	12/7/2012
Store bought	Water Bottles			17 oz bottle		5	2/5/2010	\$3.00	2/5/2011
White 45 lbs Bucket	Wheat			Bucket		5	2/3/2004	\$6.80	2/3/2034
#10 Cannery Can	White Rice			Cans	Don't buy more for awhile	4	3/1/2001	\$3.50	3/1/2031

Return to
homepage

Recording using an item

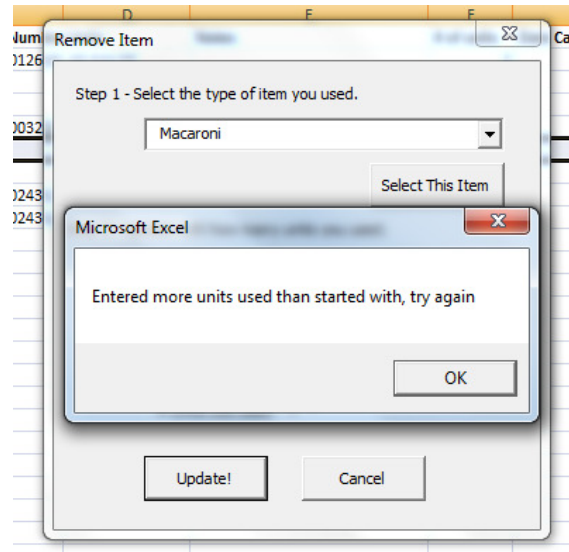
Similar to adding an item, recording the use of an item can be done either manually or by bar code scanning. After selecting the option to remove an item manually, the inventory page is activated, and the following userform is opened. First the user selects the item type taken out from the combo box populated from the item key. If the item selected is not currently in the inventory, a message box appears to alert the user.

The screenshot shows a 'Remove Item' dialog box with two steps. Step 1 is 'Select the type of item you used.' with a dropdown menu showing 'Spaghetti'. A button labeled 'Select This Item' is highlighted. Step 2 is 'Update with how many units you used.' with fields for 'Item', 'Expiration Date', 'Amount', and '# Units you used'. The 'Update!' and 'Cancel' buttons are at the bottom.

After choosing to select the item (if it is in the inventory), the item name, expiration date, and amount of units is populated into the form. The program automatically chooses the item entry with the earliest expiration date, assuming that is what gets used up first. All the user needs to do is enter how many units they used.

The screenshot shows the 'Remove Item' dialog box with 'Macaroni' selected in the dropdown menu. The 'Select This Item' button is highlighted. In Step 2, the fields are populated: 'Item' is 'Macaroni', 'Expiration Date' is '2/5/2015', 'Amount' is '4', and '# Units you used' is '5'. The 'Update!' and 'Cancel' buttons are at the bottom.

If the user inputs more units than are in the record, a message box alerts them to change the amount, otherwise the record is updated. If the same number of units are entered as used as are in the system, the whole row is deleted, otherwise the difference between what was there and what was used is put into the number of units column.



The code below brings in the item searched for at the top of the form, and populated the information in the bottom of the form. The data is sorted by item type when the form initializes, so the first matching item name is also the one with the earliest expiration date. The program pulls the expiration date and number of items information into the form.

```
Private Sub cmdSelect_Click()
    Dim found As String

    'find the item
    found = "no"
    x = 2
    Do Until Sheets("Inventory").Cells(x, 2) = "" Or found = "yes"
        If Sheets("Inventory").Cells(x, 2) = cboItem.Value Then
            found = "yes"
            x = x + 1
        Else
            x = x + 1
        End If
    Loop

    'put the item info in the form
    If found = "no" Then MsgBox "Item not found, please try again"
    If found = "yes" Then
        'need to know what row the item is on...
        x = x - 1
        txtItem.Value = Sheets("inventory").Cells(x, 2).Value
        txtExp.Value = Sheets("inventory").Cells(x, 9).Value
        txtAmount.Value = Sheets("inventory").Cells(x, 6).Value
        Rows(x).Select
    End If
End Sub
```

The code to update the number of units first checks to make sure that the number fits, then either deletes the row or updates the number of units.

```
Private Sub cmdUpdate_Click()  
  
If unitsUsed > txtAmount.Value Then  
    MsgBox "Entered more units used than started with, try again"  
    txtAmountused.Value = ""  
    Exit Sub  
End If  
  
If unitsUsed <= txtAmount.Value Then  
    unitsLeft = txtAmount.Value - unitsUsed  
    'MsgBox unitsLeft  
  
    If unitsLeft = 0 Then Rows(x).EntireRow.Delete  
    If unitsLeft > 0 Then  
        Sheets("Inventory").Cells(x, 6).Value = unitsLeft  
    End If  
End If  
  
Sheets("Inventory").Range("a1").Select  
  
Unload Me  
  
End Sub  
Private Sub txtAmountused_AfterUpdate()  
  
unitsUsed = txtAmountused.Value  
  
End Sub
```

Challenges and Learning

One of the biggest challenges for me was creating a project that could be easily used by someone without much knowledge of Excel. I carefully tried to avoid the user getting stuck on a page that they didn't know what to do with, or how to navigate away from. I also wanted to make a very practical program that could really be used effectively. One thing I decided from the beginning is that an item key would keep things clean. When I added the CueCat capability, I wasn't sure initially how to make it work with the item key, but then I added the step to check for new items when something is scanned in.

This project incorporated a lot of what we learned in class; web queries, userforms, loops, and barcode scanning. I learned a lot from the experience of putting all those concepts together and building something from scratch. I didn't learn a lot of new topics, but I did gain more understanding of what I had learned in class. I also found that the basic nomenclature has come a lot easier since working on this project, as before I would have to look up almost any line of code I wanted to write.