|  |
| --- |
| Comp4 Project  **Food Management System** |
| Author  **Christopher Dunkling** |
| College  **Hills Road Sixth Form College** |
| Date Started  **1st September 2015** |

 ****

Contents

[Analysis 5](#_Toc445174556)

[Background Identification Of The Problem 5](#_Toc445174557)

[How A Computerised System Can Help With This Problem? 5](#_Toc445174558)

[Description of Current System 6](#_Toc445174559)

[Identification of Prospective User 6](#_Toc445174560)

[Would Anyone Be Interested In A Computerised System? 6](#_Toc445174561)

[Primary User 7](#_Toc445174562)

[Identification of User Requirements 8](#_Toc445174563)

[Answers to the Questionnaire – My mother 8](#_Toc445174564)

[Answers to the Questionnaire- My aunt’s family of six 9](#_Toc445174565)

[Answers to the Questionnaire- Friends family of four 9](#_Toc445174566)

[Answers to the Questionnaire - Grandparents 10](#_Toc445174567)

[Review of Similar Already on the Market 11](#_Toc445174568)

[Data Sources and Destinations 12](#_Toc445174569)

[Analysis Data Directory 13](#_Toc445174570)

[Data volumes 13](#_Toc445174571)

[Logical Data Flow Diagrams 14](#_Toc445174572)

[E-R model 17](#_Toc445174573)

[Identification of Objects 17](#_Toc445174574)

[Objectives 18](#_Toc445174575)

[Appraisal of Potential Solution 20](#_Toc445174576)

[Objectives and limitation 20](#_Toc445174577)

[Architecture 20](#_Toc445174578)

[Computer Platform 20](#_Toc445174579)

[Program Language 21](#_Toc445174580)

[Implementation 21](#_Toc445174581)

[Design 22](#_Toc445174582)

[Overall System Design 22](#_Toc445174583)

[Fundamental Implementation objectives 22](#_Toc445174584)

[The Prototype Stage Of Design 22](#_Toc445174585)

[Application Design Decisions 23](#_Toc445174586)

[IOPS 26](#_Toc445174587)

[Top Level System Flow Diagram 27](#_Toc445174588)

[Food Item Add Flow Diagram 28](#_Toc445174589)

[Food Item Delete Flow Diagram 29](#_Toc445174590)

[Descriptions Of Modular Structure Of System 30](#_Toc445174591)

[Process Diagram 30](#_Toc445174592)

[Database Design 31](#_Toc445174593)

[Object Orientated Planning 33](#_Toc445174594)

[Product Class 33](#_Toc445174595)

[Definition Of Data Requirements 34](#_Toc445174596)

[Stepwise Refinement 35](#_Toc445174597)

[User Interface Design 36](#_Toc445174598)

[Form Navigation 43](#_Toc445174599)

[Algorithm 44](#_Toc445174600)

[Class Design 49](#_Toc445174601)

[Testing 50](#_Toc445174602)

[Input and Output Testing 50](#_Toc445174603)

[Adding a product 50](#_Toc445174604)

[Deleting a product 52](#_Toc445174605)

[Search products 53](#_Toc445174606)

[Tracking Updates in the Stock and Product databace 53](#_Toc445174607)

[Maintenance 58](#_Toc445174608)

[Code for Outpan Request 58](#_Toc445174609)

[Code for Delete Request 59](#_Toc445174610)

[Forms in design view 61](#_Toc445174611)

[User Manual 66](#_Toc445174612)

[Appraisal 79](#_Toc445174621)

[Comparison of Project Performance Against Objectives 79](#_Toc445174622)

[Potential for Future Developments 82](#_Toc445174623)

[User Feedback 82](#_Toc445174624)

# Analysis

## Background Identification Of The Problem

According to the latest government reports *(Placeholder1) ,* the average UK family is wasting nearly £60 a month by throwing away food. That accumulates to 7 million tonnes of food every year of which 4.2 million tonnes is going straight from fridges and cupboards into the bin. This is wasting valuable resources and harming the environment.

Researchers have found:

With the modern busy life style most food we buy is based on what we think we need or may have run out of plus what’s on promotion. This means plenty of the food we buy we already have and conversely forget to buy food we have run out of.

The food is then put in the relevant cupboards, fridges and freezers, sometimes in an organised way but many cases almost random.

With the ever increasing amounts of food stored in the house, it becomes difficult to remember what food we do have, where it is stored and most importantly, when it will expire, so some food gets completely forgotten about or we forget how long we have had it for and it expires and has to be thrown out.

### How A Computerised System Can Help With This Problem?

We need a system to help us organise our food buying and consumption. If the details of the food we purchased and consumed could be tracked in a computer system, this could alert us about food that was about to expire, and food that is about to run out of, as well as allowing us to quickly view all the food we currently have.

This would at least help us more effectively manage buying and consuming food and reduce wasted food.

## Description of Current System

To observe how a household shops and consume food, I used our household as an initial representative of a typical household, observing the current system over the month of September 2015.

* With the modern busy life style our household, most food was purchased in a single shopping trip weekly or fortnightly
* The food was purchased on the basis of what we think we would like for the next week or two and to replenish food we think we have ran out of.
* This system inevitably resulted in buying some food we already have plenty of and forget to buy some food we ran out.
* The purchased food is stored in the relevant cupboards, fridges and freezers in a fairly organised way.
* With large amounts of food stored in the house, it was difficult to remember what food we do have, where it is stored and when it was going expire. So some food did get forgotten about and it expires and has to be thrown out.
* To try and be more efficient food in the cupboards and fridge and freezer were checked before going on each shopping trip, but this was both time consuming and prone to missing items and often not performed.
* A list of food that has run out or wanted was kept on the fridge door, but family members often forgot to add items to the list. That means we had to go without some food until the next shop as the food items were only added to the list when they ran out rather than the last few left.

## Identification of Prospective User

### Would Anyone Be Interested In A Computerised System?

A household computerised food management system sounds a good idea but would anyone else think it would be of benefit and more importantly, would be willing to use it.

I had preliminary discussions with a larger cross section of people from family, friends of the family and neighbours that were young to old and single to larger families.

I first ask them how they currently shopped, stored and consumed food, which helped me to check if our family was typical and it also got the people I was asking to think about how they currently do it.

I then ask them if a household computerised food management system was good idea and if they thought it would be of benefit and they would consider using such a system if it was available.

At this stage I just gave a very simple generic description of a household computerised food management system.

From the discussions I found the people fell into four groups:-

|  |  |
| --- | --- |
| **Group** | **Summary Of What People Said** |
| Young  Single | Their cupboards and fridge remained empty most of the time other than for drinks and instant frozen food as they ate out a lot of the time and did not think a computerised system would be of much benefit |
| Older Couples  (OAP) | They had plenty of time and enjoyed going shopping to get out and usually only brought small quantity of food at a time for the next few days and did not think a computerised system would be of much benefit especially only some had access to computers or willing to use them. |
| University  Students | A computerised system may be useful as they have a chaotic life and wanted a quick and easy way to keep track of food reserves, but they mainly only brought small quantity of food at a time for the next few days. They admitted most of the time they would either be too lazy or forget to add and remove food from the system unless it could be made automatic. They might embrace the system more if it was an app. |
| Families | They mainly shopped weekly or fortnightly and store a lot of food in the house which they found it hard to keep track of and did end up throwing food away as it had expired, so thought they would benefit from a computerised system if it helps to save cost of food bills and to cut food waste. They were enthusiastic about the idea even if it did involve a bit more time and effort to add and remove food from the system. |

On a small sample of only 11 people, I cannot say we have conclusive results. On a larger sample of people there would probably be more overlap across the grouping, but these preliminary discussions had shown there was an interest and willingness to use a computerised food management system, so this was a viable project to continue.

### Primary User

From the preliminary discussions it was families that were most enthusiastic about the idea and thought it would benefit them.

I would require a primary user to develop this system and the obvious choice was to use my own family as it was easy to get their opinion on the design, features, usability and feedback as this proposed computerised food management system was developed.

So I decided my mother would be a good primary user to target this project at. We are a fairy standard family of three with both mother and father working and my mother is computer literate but not an expert.

### Identification of User Requirements

The next stage was to refine the details of how people currently shopped, stored and consumed food and what they would like from a computerised food management system.

I created a questionnaire which I gave to my mother and three other families that had shown an interest

### Answers to the Questionnaire – My mother

|  |
| --- |
| **Computerised Food Management System Questionnaire** |
| 1. **How do you organise your items in the house after shopping?** |
| I put raw food in to the fridge or freezer. Foods with longer shelf life like packets of pulses and canned food as well as jars and bottles go into the larder cupboard. |
| 1. **What are the benefits of using the current system?** |
| It is a simple system. |
| 1. **What are the drawbacks of the current system?** |
| Sometimes I do find cans, jars or bottles of food that has passed expiry date. I also sometimes end up buying items that already have quite a few in the cupboard because I am relying on memory thinking that they have run out. |
| 1. **What features would you like to see in the proposed application?** |
| It would be useful to know which item relates to each barcode in the database. The ability to alert me on what items are running low and also items that are approaching expiry date. It would also be a plus if it could show the last price that the item was purchased at and which store it was bought at, in order to assist in my household budgeting to know if prices have gone up or down. |
| 1. **Which existing features do you find the most useful in your current system?** |
| I find the segregation of foods in the existing system sufficient so there is no need for any change. |

### Answers to the Questionnaire- My aunt’s family of six

|  |
| --- |
| **Computerised Food Management System Questionnaire** |
| 1. **How do you organise your items in the house after shopping?** |
| Not really organized, I just put things away where there is space. Separate cupboards for different types of food (jars and tins together, pasta, rice, and dried foods together etc.) and different shelves in fridge and freezer for meats, vegetables, fish etc. |
| 1. **What are the benefits of using the current system?** |
| Quick and easy |
| 1. **What are the drawbacks of the current system?** |
| Food at the back goes out of date if not checked. Not knowing what there is without going through everything. |
| 1. **What features would you like to see in the proposed application?** |
| Need to be quick and simple to use. To be able to look up a product and see the shop and price. To be able to set a priority list of products you use often that are important not to run out of. |
| 1. **Which existing features do you find the most useful in your current system?** |
| Shopping is put away quickly. Knowing where different types of food are stored. |

### Answers to the Questionnaire- Friends family of four

|  |
| --- |
| **Computerised Food Management System Questionnaire** |
| **1. How do you organise your items in the house after shopping?** |
| The goods are sorted according to snacks, frozen food, chilled food and other inedible products, etc. |
| **2. What are the benefits of using the current system?** |
| It clearly separates and categorises the food that needs to be refrigerated, or could be left on shelf. There has almost never been a time where we left something that needs to be kept in the fridge out in the open only to find it spoilt etc. |
| **3. What are the drawbacks of the current system?** |
| It is quite time consuming especially when there is a lot of items as we have to go through each and every one of them before deciding where to put in the cupboards |
| **4. What features would you like to see in the proposed application?** |
| Maybe you can make it to sort products according to their shelf life, chilled or frozen or can be left out in room temperature, as well as if the items can be placed together such as cooked food cannot be placed together with raw food etc. |
| **5. Which existing features do you find the most useful in your current system?** |
| The sorting of the items into their respective category and types is very useful as we will then know what items can be placed together and what cannot. |

### Answers to the Questionnaire - Grandparents

|  |
| --- |
| **Computerised Food Management System Questionnaire** |
| **1. How do you organise your items in the house after shopping?** |
| Separate cupboards for tins, jars, cereals, tea and coffee, bread, biscuits and savouries. Frozen foods go into freezers in kitchen and utility room. Perishables, milk, dairy products etc. in fridge, We check all use by dates keeping oldest at front. We always check for the best use by dates when buying perishable items and never buy more than we need. |
| **2. What are the benefits of using the current system?** |
| We never use our food waste recycling bin, any peelings outer leaves etc. are composted. We always know what food we have in stock and what we need on our next shop. |
| **3. What are the drawbacks of the current system?** |
| There are no drawbacks as it works well for us. |
| **4. What features would you like to see in the proposed application?** |
| A computerised system would not improve on our current set up therefore we have no suggestions as to what it should contain. |
| **5. Which existing features do you find the most useful in your current system?** |
| The fact that we never waste food is proof that the system works because it has been honed over many years of careful planning and it WORKS. |
| **Any other comments** |
| Being in our mid 80s we were brought up in the days of rationing and shortages and have learnt the hard way that wasting food is a crime. It does not take any extra effort to plan what and when to buy in the way of food supplies and a computerised system would not improve on what we are already doing. |

## Review of Similar Already on the Market

I looked at some stock systems to get a general feel of how they are organised.

There are many inventory stock systems that contain lots of features mainly aimed toward business systems but these do not have the features required or the ease of use for a Home Food Management system.

I could only find one system that was specifically aimed at Home Food Management and this was is an IPhone app called “The Home Food Storage app” on www.foodstorageapp.com.

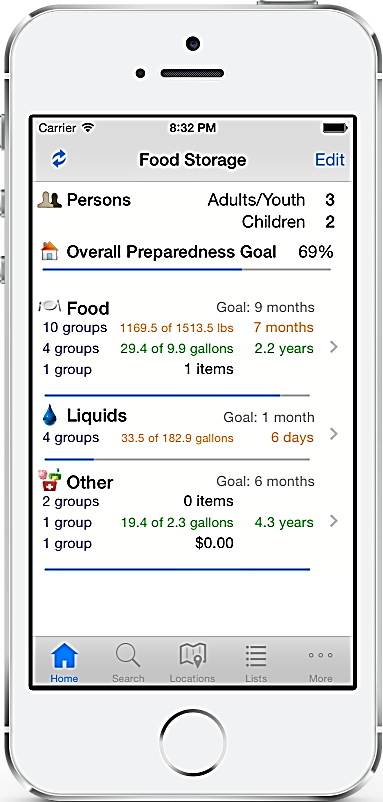
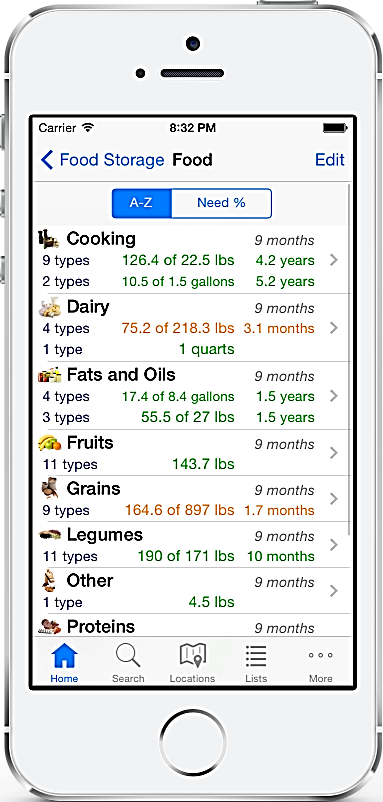
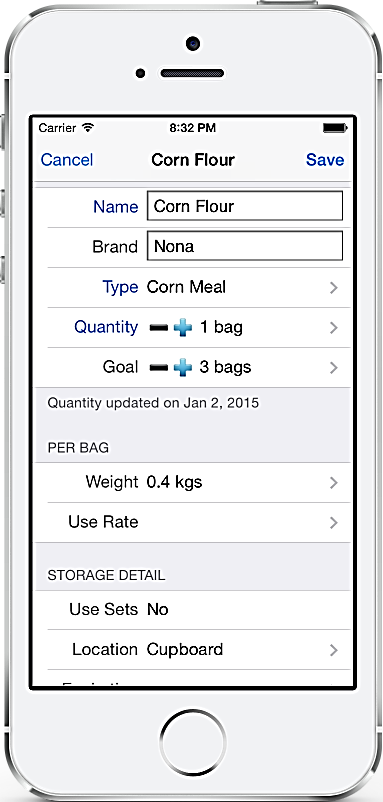
It looks like it has similar objectives as my Home Food Management project with the ability to:

* Add food items
* Remove food items
* View lists of expiring food and running out food.

There are also fields on the food entry to input weights, type of food and food category (e.g. dairy, meat, vegetable, etc.).

The app also has features to help people who are on a diet and are health conscious.

Overall, it is a nice app but it is limited to only later versions of the IPhones and it did depend on data connection to a central database of food product details that would need to be built up by the users.

## Data Sources and Destinations

We need to decide on some fundamental data type to be able to complete the Analysis

The central data for this proposed system is be a **Food Item** that contain details of each unique food type that system has ever

* The **Food Item** needs to contain a description of the food and a unique ID to uniquely differentiate this food item from other similar food items which could have very similar description
* As most food items will have a product barcode on the packaging, which the user can scan with a barcode reader,
  + This provides a quick and easy user entry system
  + The barcode numerical string that is unique for each different food item and so can be used for the food item unique ID.
* The users likely to quickly loose interest if they have to manually enter the description of every new food item on the system there this can be resolved by getting the system to read the food item description from an external web database, there are a number of external websites/databases that can be remotely queried with a barcodes to return a food item description. This will allow the food item description to be automatically entered.
* As food brought and entered on the system new food items will be created, these need to be stored locally so we will store the unique food items in a **Product Database** so they can be quickly retrieved next time
* it is likely we will have multiple instances of the same food item enter on the system but these may have different details such as expiry dates so each instance of a food item in stock will be store in a **Stock Database**
* Some of the extra attributes of a Food Item belong to the instance of the food item in the **Stock Database** such as
  + the date it was brought,
  + the date it expires,
  + the purchase price,
  + the store purchase from,
  + where it will be stored (cupboard, Fridge, Freezer …)
  + status flags relating to this instance of the food item (such as has this expired)
* Some of the extra attributes of a Food Item belong to all instance of the food item in the **Stock Database** such as
  + the barcode
  + the description ,
  + the quantity we currently have,
  + the minimum quantity we want to maintain,
  + is this one of our favourite foods

As these attributes are food item are not instance specific they can be stored with the food item in the **Product Database**

* If we sore the barcode with each instance of a food item in the **Stock Database** it provides a link back to the food Item in the **Product Database** and the food item barcode becomes our primary key

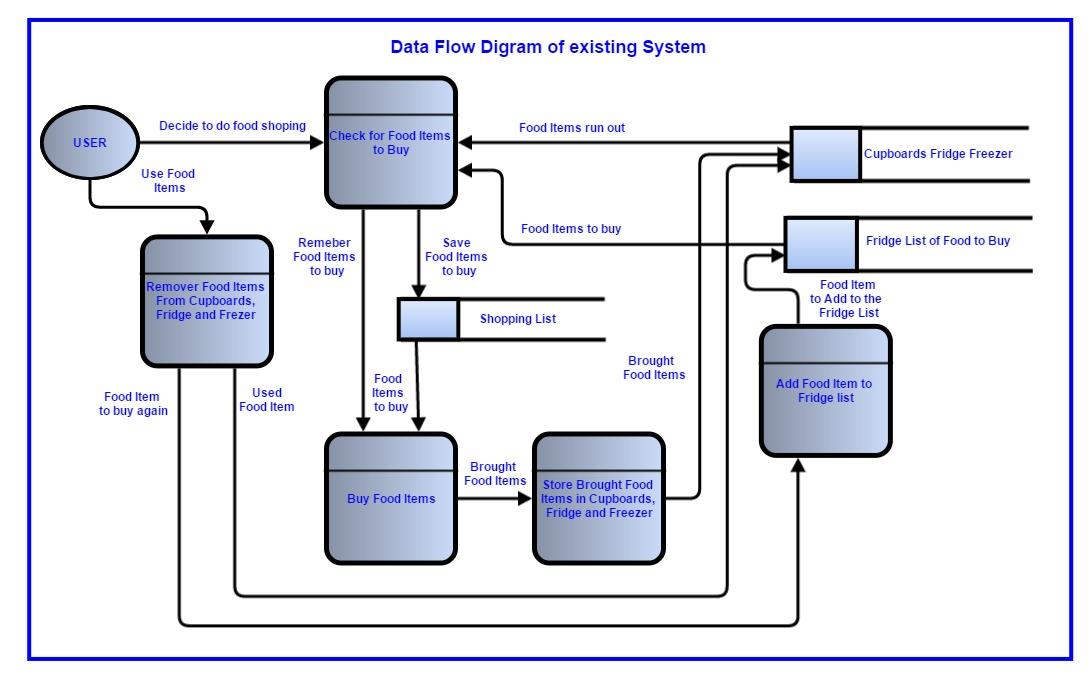
## Analysis Data Directory

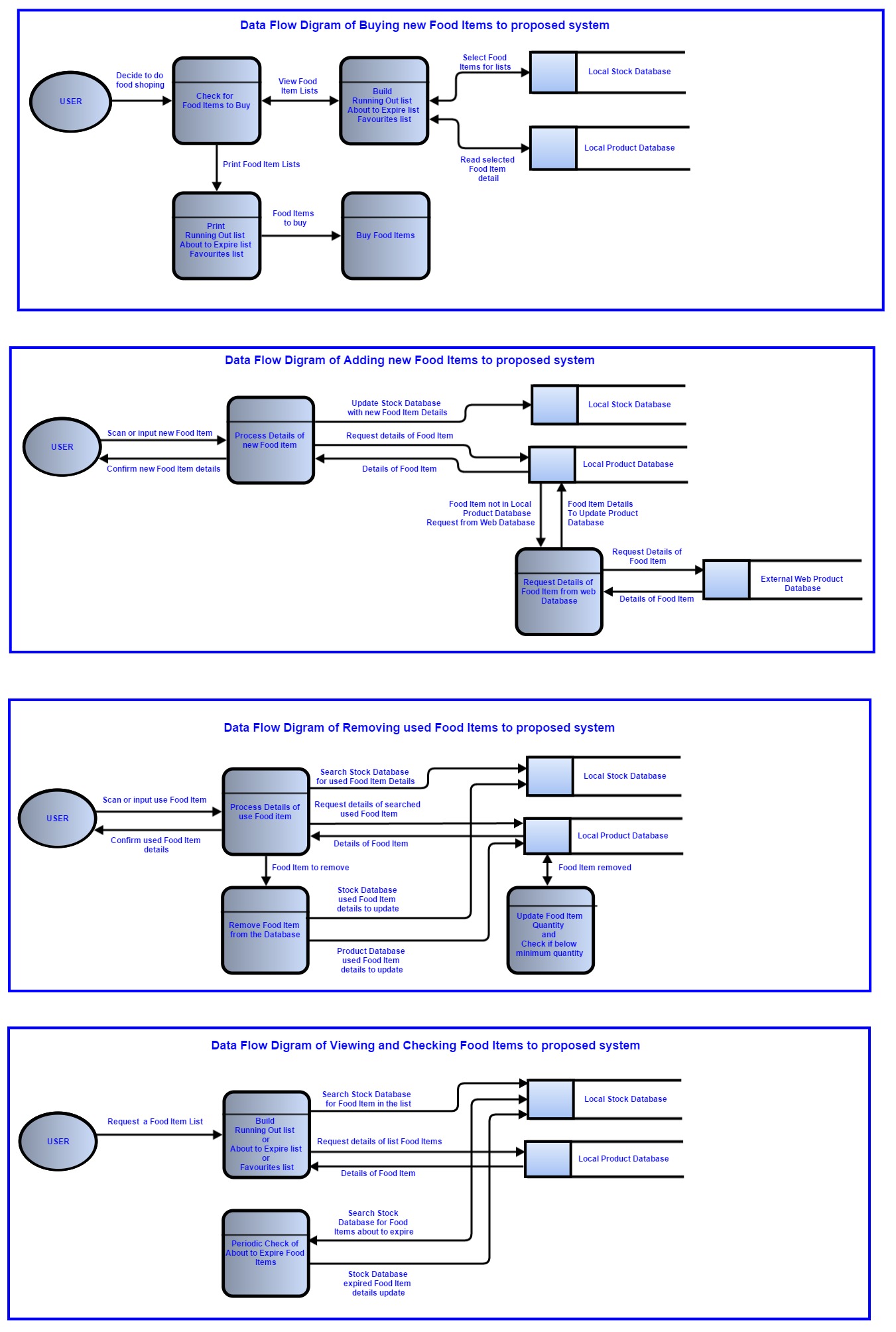
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Example** |
| Product name | String | “Rowntree’s fruit pastilles” |
| Barcode | String | “7613033796487” |
| Date bought | Date record | 17/11/2015 |
| Expiry Date | Date record | 24/11/2015 |
| Quantity | Integer | 5 |
| Minimum quantity | Integer | 5 |
| Location | String | “Cupboard” |
| Last bought at | String | “M&S” |
| Price | decimal | 2.55 |

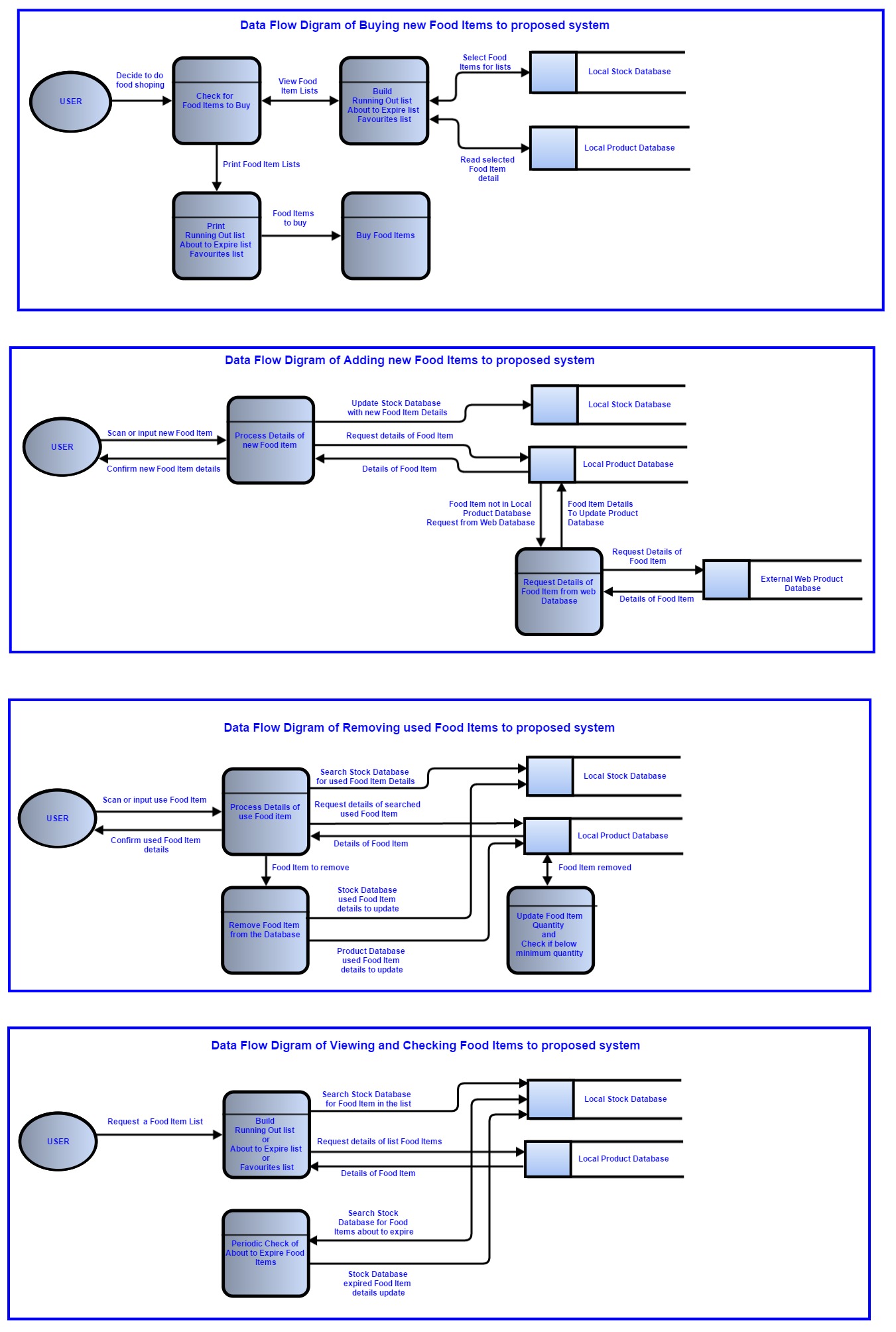
## Data volumes

* The main table will have 5 fields(name, barcode, date bought, expiry date and quantity, Location, Last bought at and price)
* The stock will have only the barcode, date added and expiry date
* The product will have barcode , name, quantity and min quantity

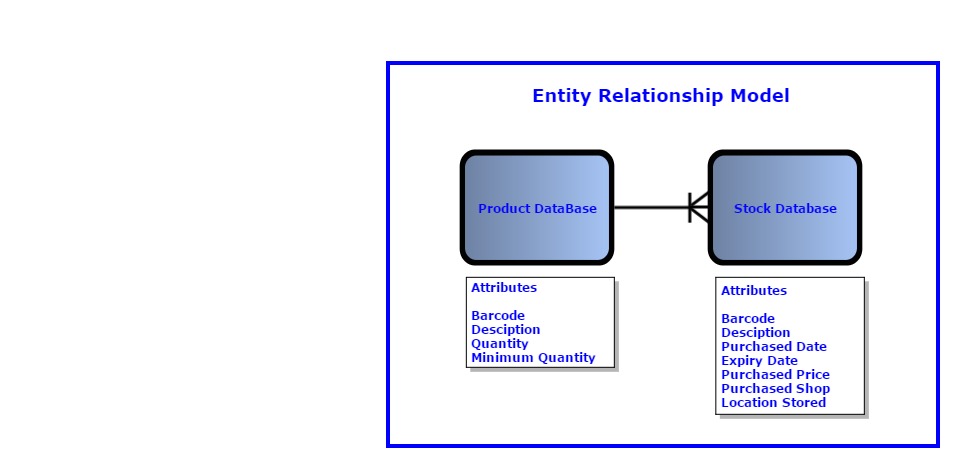
## Logical Data Flow Diagrams



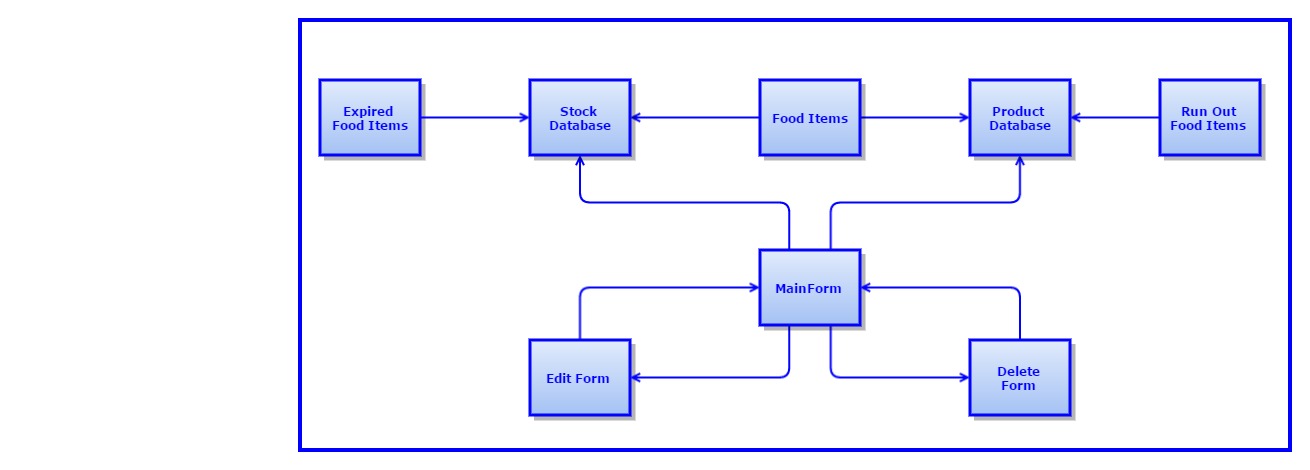




## E-R model



## Identification of Objects



## Objectives

From the feedback to the questionnaires, the primary objectives for this system are: -

1. The application must run on a desktop PC running Microsoft Windows (XP upwards)
2. The application where possible will use Microsoft Windows preferred interface controls to keep this application consistent with other applications and so easier and indurative for users use.
3. User input to add, remove and select food items will be form-based
4. A barcode reader will be connected to the PC and the application must be able to accept the barcodes read from food wrappers to:
   1. Add new food items
   2. Remove used food items
5. Where food items do not have a barcode an alternative entry to identify a food item must be provided.
6. The application will be able to query external web databases with the barcode to obtain food item descriptions
7. The application will always search for food items in local database before querying external web databases
8. The user must be able to simply and quickly enter the details of all food items purchased into the system.

These details will include:

* 1. The food item description (name and make and size)
  2. The date purchased
  3. The expiry date
  4. The purchased price
  5. The shop purchased from
  6. The minimum quantity to maintain
  7. The location stored
  8. Is on Favourites list

1. The application will attempt to auto-fill all entry fields with values from:
   1. External source like the external web database
   2. Historical data from previous purchase of the food Item
2. Entry fields for price, shop purchased, and location stored can be left blank if not required by the user
3. All entry fields can be edited and accepted changes will be reflected across all affected data and stored back into backup files
4. The user will be able to remove used food items by scanning the barcode or selecting the food item form any of the displayed views of food items
5. The user will be able view details of :
   1. All food items currently stored
   2. Food items that is close to or past expiring
   3. Food items that is close minimum quantity or ran out
   4. Favourites Food Items
   5. Search Food Items
6. The user will be able to print any of the displayed views of food items.
7. The application will load all stock and production data from file on start-up and store all stock and production data to file on close-down
8. The application will monitor expiry dates and minimum quantities to keep displayed views of Food items that is close to or past expiring and Food items that is close minimum quantity or ran out up to date.
9. The application should be able to process at least 5000 food items in the Product Database and 1000 food items Stock Database

## Appraisal of Potential Solution

There are several options for the implementation of this system

A top down approach was used to decide on the type and architecture of the system to be developed

### Objectives and limitation

The first step was to decide what the objectives of this development were

* The primary objective of this development is a proof of concept
* The system is initially being developed for Single user
* There is a lot of potential for enhance functionality, but within the time restrains of this project the development should concentrate on fundamental functionality

### Architecture

The project could be implemented using several different architectures

* Web Server based system, this would allow for remote access to system from server computer, tablets and mobile phones which would be nice but not essential. This would require a steep learning curve with my limited knowledge of Web Server and HTML development.
* Desktop distributed IOT (internet of things) system comprising of a central desktop computer and several wireless connected remote micros computer (such as the Raspberry Pi or Arduino) each with barcode scanners this would allow multiple barcode scanners each close to fridges, freezers and cupboards. This would be an interesting development but seemed a bit of an excessive solution especially as the remote units could easily be replaced with wireless barcode readers.
* Desktop computer application, this would be the simplest and quickest to both prototype and develop the full application with my current knowledge of software development and would provide all the current requirements.

### Computer Platform

This is a fairly straight forward decision most people including our household have desktop PC running a version of Microsoft windows, plus all the PC I need to develop and demonstrate this system on in college are running Microsoft windows

### Program Language

For development of a Microsoft windows application there is a large Choice of Languages

The Languages I currently have sufficient knowledge to develop this system without having to learn too much more are:

* Pascal

This is the language I have recently learnt at college, and I feel this may be a bit restrictive and not provide access to all the latest Microsoft window controls

* C++

This is a language I have learnt, but is not so well supported as C# by Microsoft and so would require more code and time to develop the application

* C#

This is a language I have learnt, and is well supported by Microsoft window as it is the native language for Windows applications development, with a large selection of classes and a good Window form editor

### Implementation

Using C# and Microsoft Visual Studios allows

* Quick creation of user interface window layout using the window form editor,
  + So a prototype user interface layouts to presented to the user for feedback early in the development,
  + It is simple to make refinement to the interface layouts throughout the project
* Microsoft C# provides a useful DataGridView classes for presenting and manipulating data in a table layout.

https://msdn.microsoft.com/en-gb/library/system.windows.forms.datagridview%28v=vs.110%29.aspx

* + This will speeds up the development with well tested third party code,
  + This was designed by Microsoft to provide their preferred interface layout and interaction that most people will be use to from other applications.
* For data Microsoft C# provides a DataTable class that provides all the functionality and memory management to add and delete and sort data records,

https://msdn.microsoft.com/en-us/library/system.data.datatable%28v=vs.110%29.aspx

* + This speeds up the development with well tested third party code.
  + Instances of the DataTable Class can be directly passed to the DataGridView Class to be displayed, which simplifies the processing
  + A List Data Instance has the methods to read and write to a database or a file, as the primary aim of this project is a proof of concept and is designed for a single user reading and writing to file is sufficient rather than adding unnecessary complexity of setting up a database, there will be no potential speed of response using files as the data lists are held in memory and only backup to file.

# Design

## Overall System Design

### Fundamental Implementation objectives

Analysis of the fundamental implementation objectives for the application are:

* It will be a proof of concept
* It is designed for a Single user
* It concentrates on the fundamental functionality requirement
* It runs on a Desktop computer running Microsoft Windows
* It is developed in C# using Microsoft Visual Studios IDE
* It will use Microsoft C# DataGridView class for presenting and manipulating data in a table layout
* It will use Microsoft C# with a DataTable Class to provide all the functionality and memory management to add, delete and sort data records
* It will store data to file instead of a database

### The Prototype Stage Of Design

* Testing the barcode reader
  + The barcode reader inputs character strings into the PC like a keyboard so when a form text entry box has the focus, it will accept character input from both the keyboard or barcode reader. This will allow the barcode to be manually entered when the barcode is not readable or not available.
* Locating a suitable external web database to query product descriptions form barcodes
  + There are a number of web sites that provide product lookup from a barcode, only some of which provide the facility to make a remote request.
  + Most of the better sites providing barcode lookup are commercial with sign-up charges and monthly subscription that may be worth considering if this is a commercial application with a large number of users. For the purpose of this project, we really wanted a free site, and I found a site called **Outpan** (<https://www.outpan.com>) that is a free service that allows user to add and share product information. It has rather limited contents and the product descriptions are a bit vague but it is sufficient for this project.
  + I implemented a simple test application to confirm the remote https queries worked and the response time was reasonable.
* Testing the Microsoft C# DataGridView class and DataTable class
  + Most of the data on the application is a list of data that are best displayed in a table format of a food item per row with the attributes in the columns. Microsoft has the DataGridView class to provide presentation and manipulation of data in a table layout and DataTable class to provide the memory management to add, delete and sort data in lists as well as reading and writing the data to file or database. An instance of DataTable can be passed directly to an instance of a DataGridView to be displayed in a form.
  + I read the Microsoft documentation on these two classes to confirm they had all the features I required and implemented a simple test application to confirm I understood how to use these classes in a program.
* Create prototype user interface layouts to presented to the user for feedback
  + The user interface can be quickly designed and modified using the Windows form editor
  + Adding the user interface to the simple application implemented to test the DataGridView and DataTable classes I was able to create example display layout of each part of the proposed application to show the user and get feedback.

### Application Design Decisions

With the main processing structure decided, we now need to look at the exception rules,

**Adding New Food Items**

* Most people would say a crucial requirement is that the system is simple and quick to use and entering food items purchased would require the most user input time. Anything that can be done to minimise this would make the users more likely to continue using the system.
  + Using the barcode reader to enter the food item identifier and **Outpan** product information server to obtain a product description from the barcode allows these entry fields to be quickly and automatically entered.
  + The purchase date can be assumed to be the entry date and so can be automatically entered.
  + The user will need to enter the expiry date as the system cannot read or even estimate this from previous food items of a similar type.
  + If a food item has been previously purchased, the last purchased price, shop, and location stored can be automatically entered and the user can update these if they have changed.
  + From the limited survey of user requirements, not all users were interested in the last purchased price, shop, and location stored. So there will be an option to leave these blank.
  + The minimum quantity will default to 1 unless previously set.
* The expiry date will be entered via the standard Microsoft calendar popup dialog box
  + This will always open with today’s date highlighted to make it easier to select the expiry date
  + As this is the only method for the user to enter the date it ensures the date entered is a valid date.
  + The user will be able to enter an expiry date that has already expired but the user will be prompted to confirm that this is correct.
  + Expiry dates do not contain time so time is not required to be entered, likewise there will be no time component in the purchase date.
* To speed up and simplify the shop and stored location entry fields
  + The shop entry field will have drop down lists of popular supermarkets.
  + The location stored entry field will have drop down lists of common places to store food (Fridge, Freezer, Cupboard 1, Cupboard 2 …).
  + For both these entry fields the user can still type in their own shop or stored location.
* The user will still have the choice to confirm or cancel the addition of food items at any time in the addition process.
* All the data entered will validated when the user confirms and if any data is missing or incorrect the user will be prompted to correct before the confirmation can be accepted.
* There will be food items that do not have barcodes so there must be an alternative to scanning a barcode
  + The simplest alternative for the user is to allow them to type in a food item name identifier that will be used as the unique identifier.
  + The user should select a check box to tell the application they intend to use a name identifier rather than a barcode so the application knows not to validate as a barcode value or send a request to the **Outpan** product information server for the description.
  + If a name is used for a new food item the name can be copied into the description field as default value, and the user can edit the description if required.
* The request is sent to **Outpan** for a product description from the barcode, often resulting in the barcode not being found or a poor returned description.
  + The description from **Outpan** (if any description available) is only a suggested description and can be edited by the user.
  + The description is only a request from **Outpan** for new food item and then the food item is stored in the product database and from then on the food item details from the product database will be used. Any user edition to the description will be retained and reflected in all views or the data files.

**Deleting Consumed Food Items**

* Although removing food items requires less user input, it is just as important for the system operation as the addition of the food items, so anything that can be done to minimise user input would make the users more likely to remember to remove a food item.
* If the product barcode on the packaging is still available, the barcode reader can be used to scan an item for removal.
* If the product barcode on the packaging is not available or the user finds it simpler, the food item can be selected for removal from the product list view of all food items in stock.
* In both cases, where there are more than one of the selected food item in stock the application will present the user with a list of the details of all the selected food item in stock with the oldest item selected for deletion, the user can select neither if this is not the one the user wants to delete.
* The user will have the choice to confirm or cancel the removal of the food item at any time in the process.
* The Run Out view will be updated if this is the last of the item and the Running Low view will be updated if this food item quantity has fallen below the minimum quantity. The quantity in the product list view will also be updated.
* When a food item is removed, it cannot be removed from the Stock list immediately as the user still needs to be able to view them in Run Out view and Running Out view.
* When a food item is removed, the quantity in the product list is decreased by one and the instance of the food item in the stock list is flagged as removed.
* Instance of food items in the stock list flagged as removed will be removed from the stock list 14 days after being flagged as removed. That should give sufficient time for the user to notice the food item has been consumed and go shopping to replenish.

**Viewing Food Items**

* The home page should be just for quick reference and contain a summary of important events
  + Two summary Table views
    - Latest ten Expired food items
    - Latest ten Run Out food items
* The Application also allows the user to view four more tables
  + The Product View, which contains the full list of all items in stock.

To keep the list easy to read, it only contains one instance of each food item stored and displays the quantity in stock.

* + The Running Out View, which contains all food item that the quantity has fallen below the minimum quantity, or already ran out in the last one week.
  + The Expiring View, which contains all food item that are going to expire in the next week or already have expired.
* The Search View, allows the user to enter a string that is compared against the stock food item descriptions, to produce a list of food item that match. The search ignores character case.
* The user is able to print any of the views.

**Data Storage**

* While the application is running, both the product data and stock data are held in memory. Even at the estimated full data size, this a would easily fit into all modern PC memory.
* The application uses three CSV (comma separated values) files to store the data while the application is not running and as a back-up.
  + There is a Product data, Stock data and Main Database CSV file that is held in the same directory as the application normally out of reach of the user to be safe.
  + As CSV files use commas to separate the variables in the file, we have to be careful to either remove the commas in the strings or place quotes around strings to prevent the data being misinterpreted.
* The data is loaded from the CSV files when the application starts and store back to the CSV files when the application is closed.

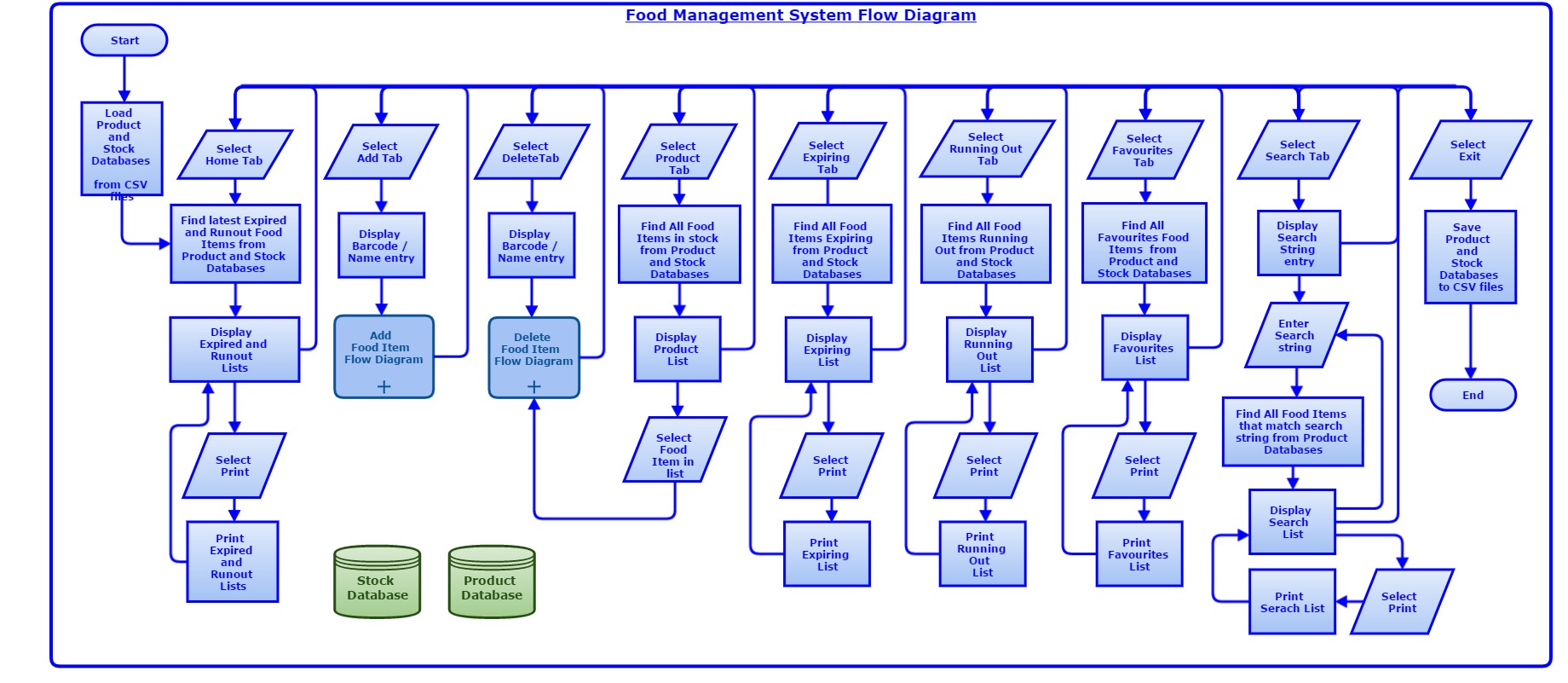
**Background Periodic Processing**

* The application needs to monitor when food items are expiring
  + This is check when the application starts
  + Since the expiry date is based on date, and not time, the application only needs to check again each new day.

### IOPS

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Process** | **Storage** |
| Barcode | Search results | Search for product | Product database |
| Name | Edit window | Delete products | Stock database |
| Location | CSV files for databases | Add products |  |
| Shop | Printed tables | Convert data table to csv file |  |
| Price |  | Status check on products |  |
| Purchase date |  |  |  |
| Expiry date |  |  |  |
| Min quantity |  |  |  |
| Position |  |  |  |

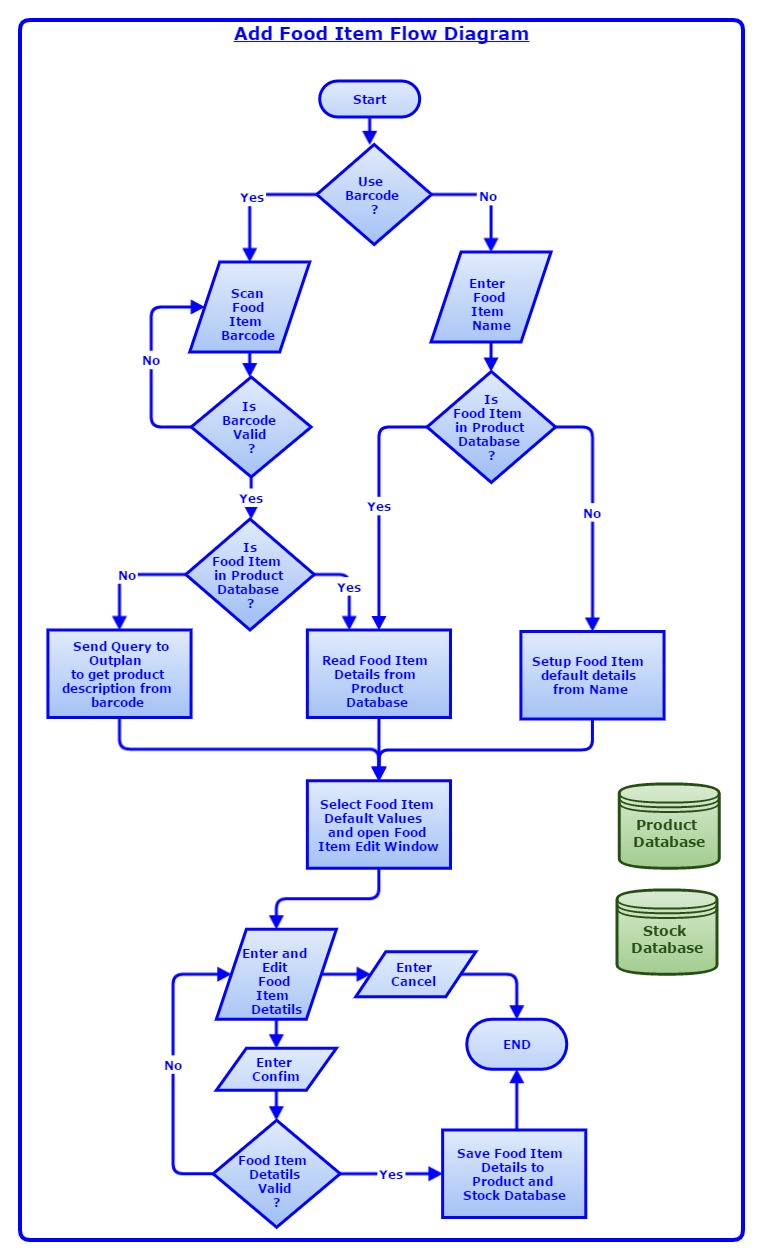
### Top Level System Flow Diagram



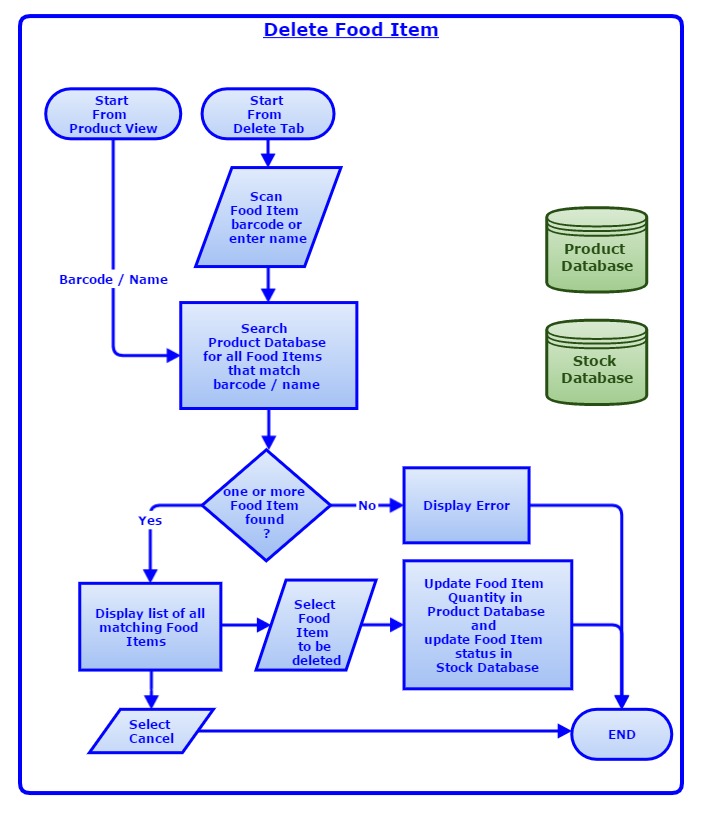
* This top level system diagram shows all the proposed high level processing of the food management system.
* All the top level processes (“Home”, “Add”, “Delete”, “Product”, “Expiring”, “Running Out”, “Favourites”, “Search”) will be selected from the Microsoft Service Tabs and so all these processes effectively run in parallel and the user can swap between Service Tabs at any time leaving the previously selected Service Tab process in current state until returned to.
* On start-up the system will open in the “Home” Tab window.
* The “Print” button will print the current selected Service Tabs window (if print is applicable to this process).

The following two Flow Diagrams show the proposed expanded Add and Delete processing of the food management system.

### Food Item Add Flow Diagram

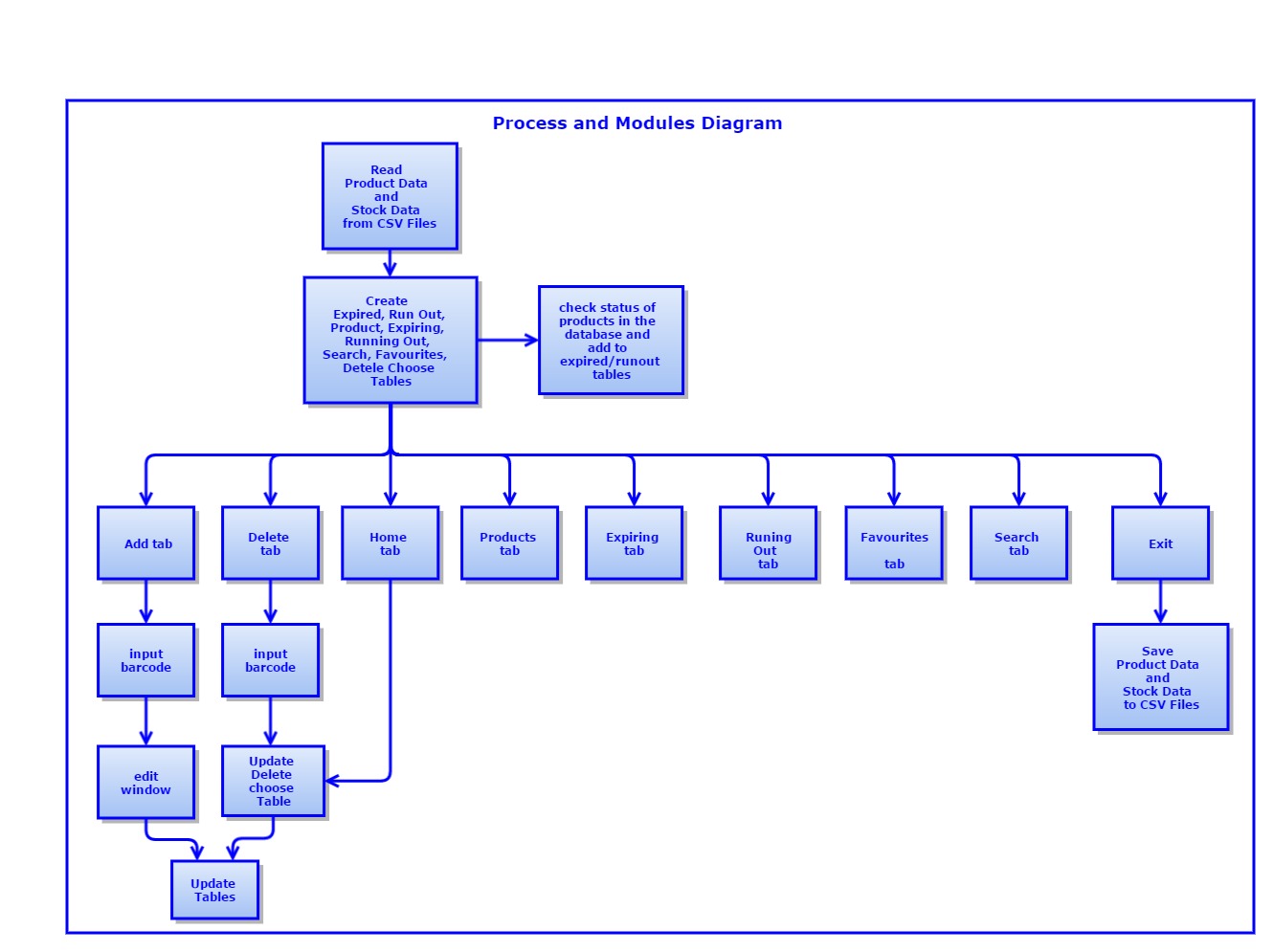


### Food Item Delete Flow Diagram



## Descriptions Of Modular Structure Of System

### Process Diagram



## Database Design

The **Product Table** will contain **Product Data records** for each unique food item entered into the system

|  |  |  |
| --- | --- | --- |
| **Product Data record** | | |
| **Field Name** | **Data Type** | **Data Size (bytes)** |
| Unique Barcode(Primary Key) | String | 16 |
| Description | String | 64 |
| Current Quantity in Stock | Integer | 4 |
| Minimum Quantity required | Enumerated | 2 |
| Favourites list flag | Boolean | 2 |

The **Stock Table** will contain **Stock Data records** for each instance of a food item currently stored in a cupboard, Fridge, Freezer …

|  |  |  |
| --- | --- | --- |
| **Stock Data record** | | |
| **Field Name** | **Data Type** | **Data Size (bytes)** |
| Barcode | String | 16 |
| Purchase Date | Day, Month, Year Record | 8 |
| Expiry Date | Day, Month, Year Record | 8 |
| Price | Decimal | 4 |
| Shop | String | 16 |
| Location Stored | String | 16 |
| Status Flag | Enumerated | 2 |
| Position(Primary Key) | Integer | 4 |

It is difficult to estimate the number of records that will be accumulated over time but totalling all the food items in our house and extrapolating up for a year is a reasonable estimate.

|  |  |  |
| --- | --- | --- |
| **Data Record** | **Size (bytes)** | **Size allowing for data alignment (bytes)** |
| Product Data | 88 | 96 |
| Stock Data | 70 | 84 |

|  |  |  |
| --- | --- | --- |
| **Data Record List** | **Records Stored** | **Total Size (Kbytes)** |
| Product Data List | 5000 | 468 |
| Stock Data List | 1000 | 78 |
|  |  | 546 |

* This is a very rough estimate but does show the data size is not too large to fit into most modern computer memory.
* Once the system settles down the Stock data list size should maintain a steady maximum level as old items are used and new items are purchased.
* The Product list will initially rapidly grow as new items are added but then should slow down as new items purchased will already be in the list and it will just be the few items that are different from the normal purchase or new items replacing obsolete items that need to be added.
* To prevent Product list getting too large, the system may need to run a “garbage collection” occasionally to remove old product items that have not been referenced for a long time.

## Object Orientated Planning

Most classes are auto generated by Microsoft Visual Studios when the display forms are created

### Product Class

The Product class is the only class not auto generated and contains the data from both the Product and Stock Database and the methods keep the data up to date.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product Class Data** | | | | |
| **Access Type** | **Field Name** | **Field Type** | **Initial Value** | **Description** |
| Public | Barcode | String | “” | It is the barcode of the new product |
| Public | Name | String | “” | It is the name of the product |
| public | Date added | DateTime | Today’s date | This is the date it is added |
| public | Expiry date | DateTime | “” | This is the date it expires |
| public | Quantity | Integer | 1 | The quantity of items in the database |
| public | Minimum Quantity | Integer | 1 | The quantity of items needed to not flag as runout |
| Public | Location | String | “” | The location in the house of the item |
| Public | Shop | String | “” | The shop where item is last purchased from |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product Class Methods** | | | | |
| **Access Type** | **Method Name** | **Parameters** | **return Type** | **Description** |
| Public | Statuscheck | void | void | This checks for expiring, running out and favourites products and called at start-up and after adding and deleting food items |
| Public | Updatetable | void | void | This takes data from the edit form and updates the details in the appropriate tables |

## Definition Of Data Requirements

The following table contains all the data that could be entered into the system by the user, and how I plan to validate it to ensure that all fields are correctly inputted and stored. The erroneous data for most fields is a blank field, so in cases where the validation check is only allowing the user to select correct fields from a list or directory, there is no way to test this.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Validation Check** | **Validation Description** |
| Product barcode | String | 13 characters | length | Allows the user to input 13 or less character strings to ensure the barcode is correct |
| Product name(for insertion) | String | Up to 50 characters | That it is a string | Allows user to only input strings for name |
| Product last shop | String | Up to 50 characters | That it is a string | Allows the user to select a shop from the list or input their own |
| Product Location | String | Up to 50 characters | That it is a string | Allows the user to specify a location where the product is stored |
| Product Price | Decimal | 4 digits. 2 either side of the decimal point | Any character that are not integer | Allows user to give a price to the product |
| Product name (for deletion) | String | Up to 50 characters | String exists in database | Allows only products in database to be selected for deletion |

|  |  |  |
| --- | --- | --- |
| **Field** | **Valid Data** | **Erroneous Data** |
| Product barcode | “12365” or “1111111111111” | “hi” or “111111111111111111111111111111111111111” |
| Product name (for insertion) | “Heinz Baked Beans” |  |
| Product Location | String with less than 50 characters | An integer or other non-string data type |
| Product Price | 4.45 | “Asda” |
| Product name (for deletion) | Product name in database | Product name not in database |

## Stepwise Refinement

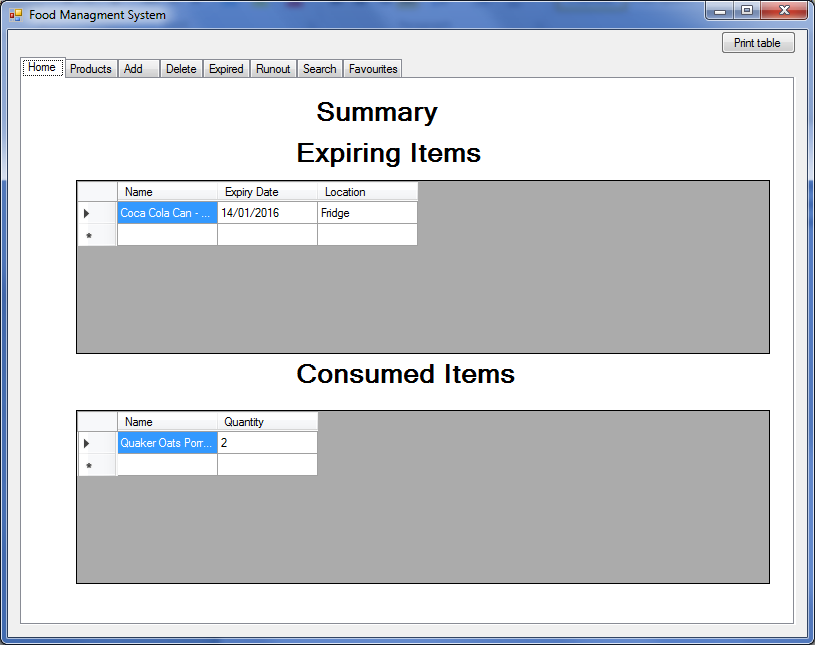
This is a more detailed list of the processes involved in the new system, mirroring the system flow chart. It allows me to break down each task into its simplest processes, and organise the data that has been inputted into the system.

1. **If searching EXISTING product** 
   1. **Select Search tab**
      1. Search by name
      2. List of items is created that contains the search string
2. **If ADDING NEW product(s)** 
   * 1. Select Add Tab
     2. Enter or scan product barcode
     3. Enter data into empty field or edit **Outpan** Data
     4. Choose if you want to add to favourites or not
     5. Select OK button
3. **If DELETING NEW product(s)** 
   1. **By DELETE TAB**
      1. Select Delete tab
      2. Enter or scan product barcode
      3. Choose product from list
      4. Select delete button
   2. **From PRODUCT TAB**
      1. Select Products tab
         1. Select row
         2. Press enter key
      2. Window with list of instances of the selected product appears
         1. Select row
         2. Select delete button
4. **If Printing Product table**
   1. **Select Tab of data which you want to print**
   2. **Select print button**

## User Interface Design

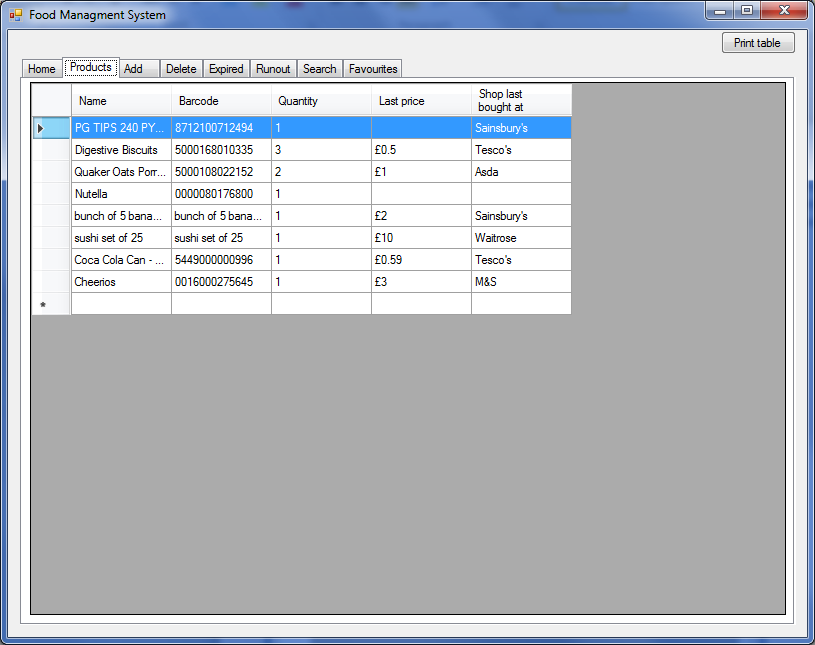
It was decided the user interface would use the Microsoft Service Tab to select the top level functionality.

* User are now familiar with the Microsoft Service Tabs to navigate between top level functions as they are now used in a majority of modern applications rather than menus.
* The Microsoft Service Tabs allow a tidy way to have multiple windows processing in parallel.
* The Home tab shown below is the start-up window displaying the tables of recently expired and consumed (run-out) products.
* All the other selectable process Tabs are shown in a row above the window making all choices easy to see and select.
* The print button is on the right hand corner of the window and is context sensitive to the selected tab and so it will print the data displayed in the current tab window.

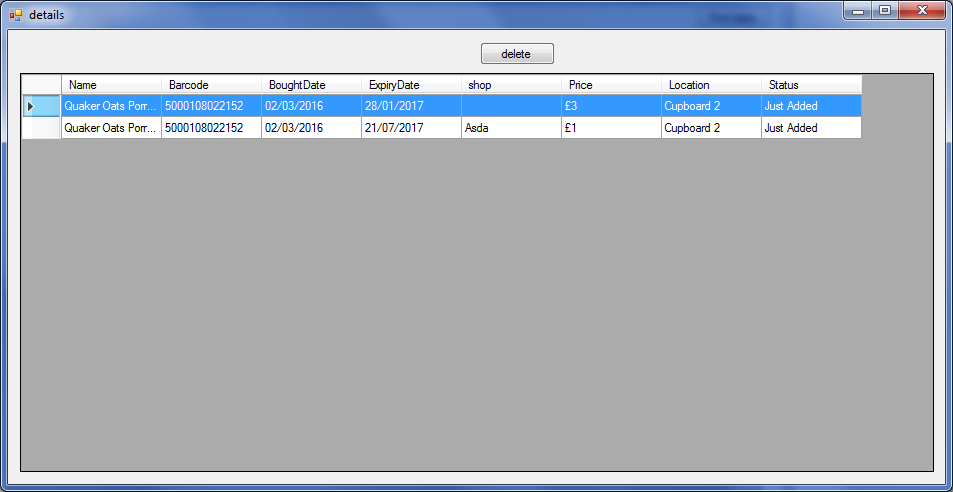


The Microsoft TableGridViews are used to display the data as shown below for the Product tab window

* The TableGridView display allows the table data to be sorted by any column in increasing or decreasing order and the column width to be increased or decreased or auto adjust to best view.
* The Microsoft TableGridView class is responsible for all the user interaction to manipulate the table data and so use standard controls used by many other applications and will be familiar to Users.



* Selecting a product from the table opens a window providing details of all these products in stock and allows one of them to be deleted if required.

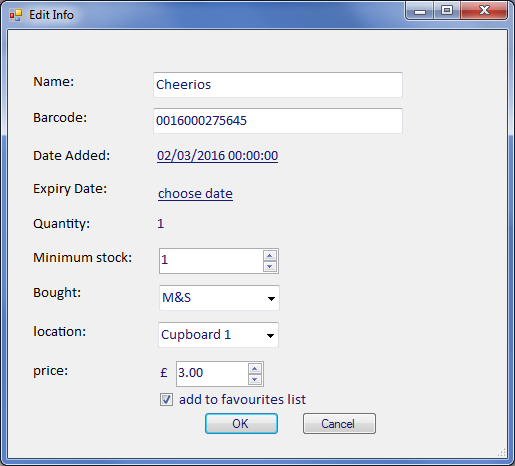


To Add food items the user selects the Add tab which displays scan/ input barcode window as shown below

* The user can either scan the barcode in using the barcode reader or type the barcode in if the barcode reader fails to read a damaged barcode.
* For food items that have no barcode, the user can also select the tick box “Has no barcode” and enter a name for the food item

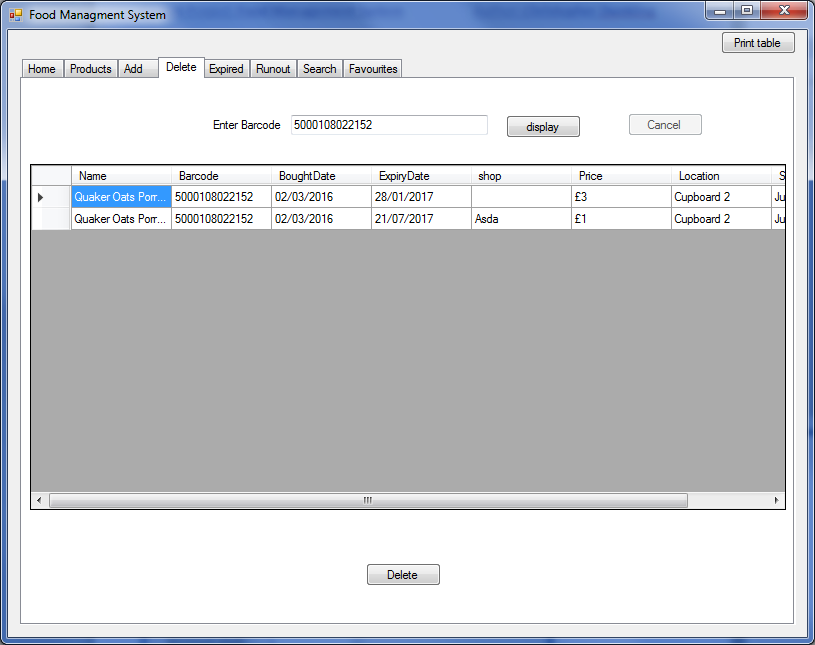


* When OK is selected on the Add tab scan/ input barcode window the Edit info window below will be displayed with as many of the input box auto filled with data
* There are some fields such as the expiry date that user must enter a valid date but the user interface can still help by displaying the stand Microsoft calendar window to select the date that makes it quicker to enter and will always be a valid date entered
* The user can edit any of the auto filled fields that are incorrect or could be improved
* Some fields such as Brought, Location and Price can be left blank if not required
* When the user selects OK the data fields entered are validated and if any fields are not valid an error window will be displayed and the user can’t preceded until all fields are valid
* At any time the user can select Cancel to cancel adding the food item

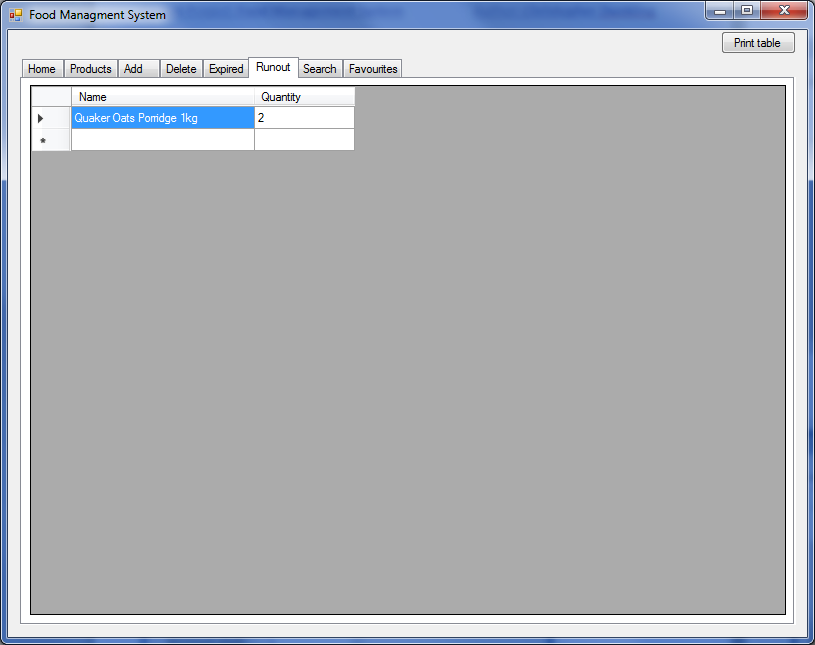


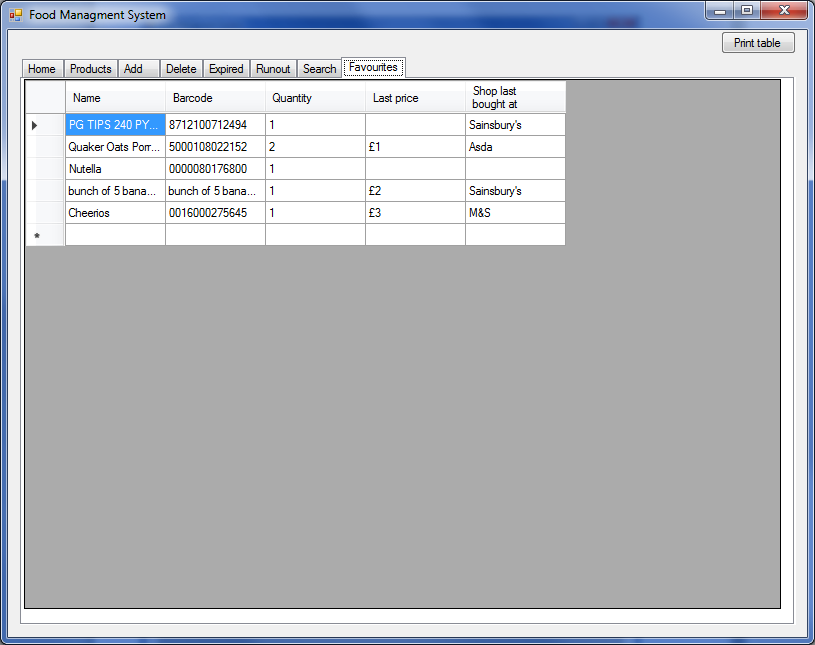
To delete a food item the user can either select the food item from the product table as shown previously or the user can select the Delete tab which displays scan/ input barcode entry box

* The user can either scan the barcode of the food item to remove using the barcode reader or type the barcode in if the barcode reader fails to read a damaged barcode.
* To remove food items that have no barcode, the user can enter the Add name for the food item
* Selecting the display button displays a table providing details of all these products in stock
* The oldest of the food items in stock will automatically initially be selected
* The user can selecting one of the food items in the table to delete and selecting the delete button will delete the food item
* At any time the user can select Cancel to cancel deleting the food item



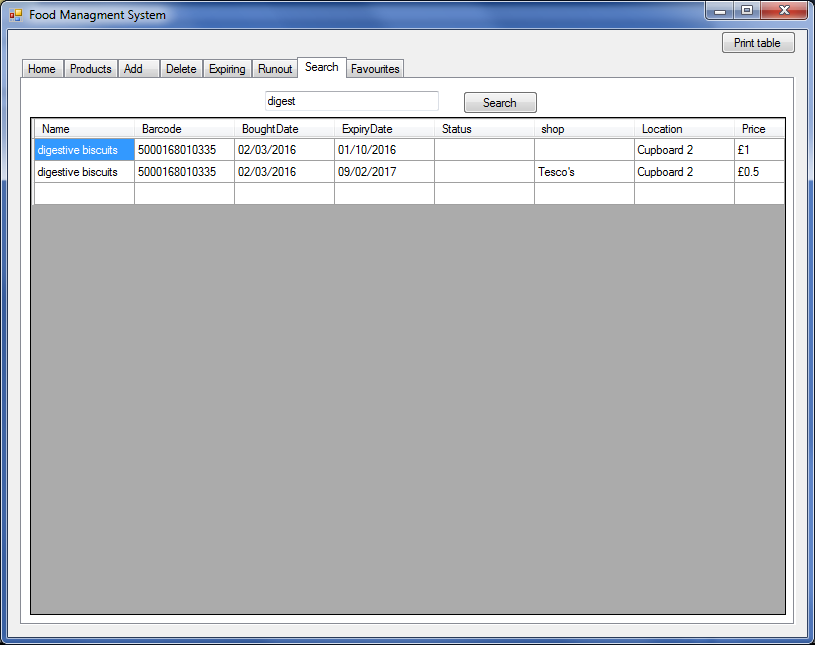
To view food items running out, expiring and favourites the user selects the relevant tab which will display the relevant tables of food item details.





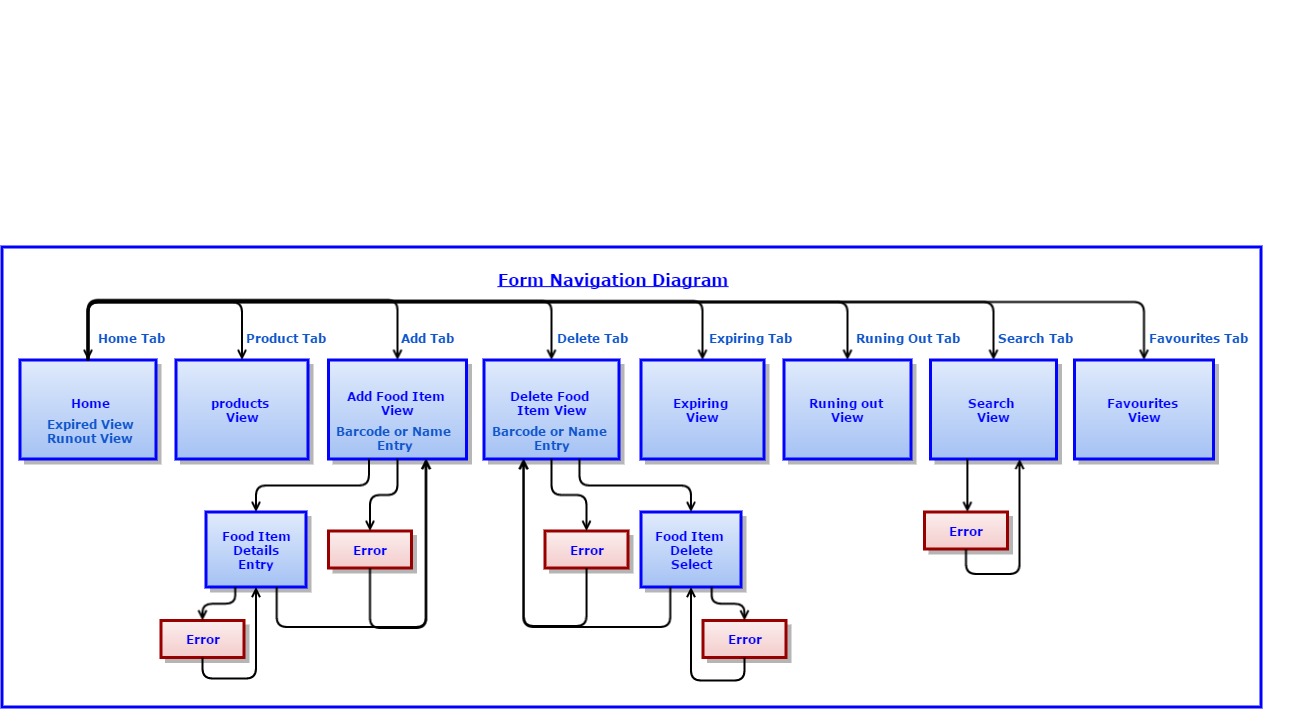
To Search food item the selects search tab which displays search string entry box

* Entering the search string and selecting the search button displays a table providing details of all those food items in stock with a description that matches the search string



### Form Navigation

This diagram shows the Food Management system windows navigation



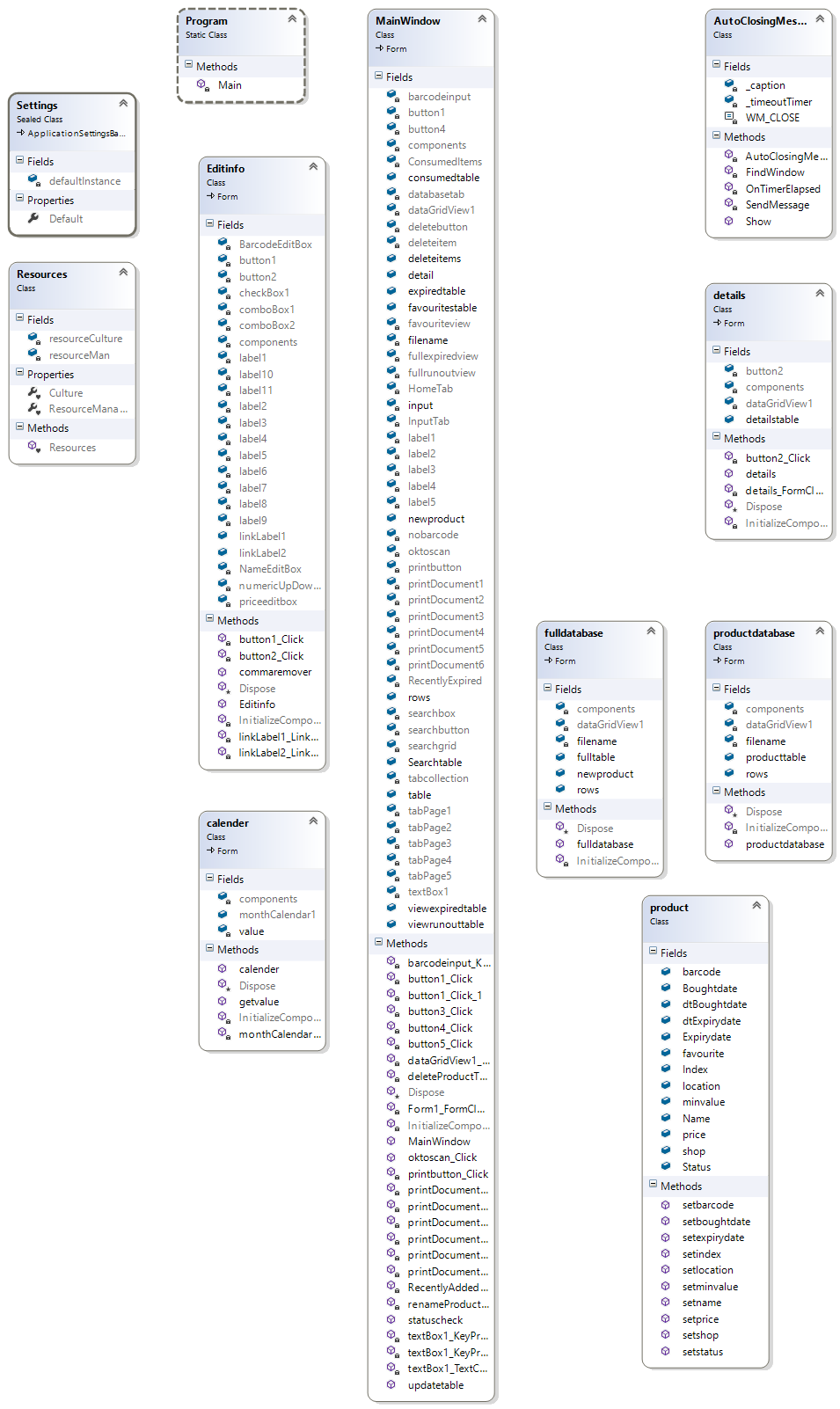
## Algorithm

|  |
| --- |
| **Title** Product Description request from **Outpan** |
| **Description**  If the entered barcode is a valid format and “Has no barcode” Tick box is not checked and the barcode was not found in the local product database  The Product Description (name) is request from **Outpan product information server** |
| **Pseudo code** |
| //This process is only called when  // There is a valid format barcode entered  // “Has no barcode” Tick box is not checked  // The barcode was not found in the local product database  Barcode 🡨 BarcodeInputBox data  **IF** Barcode Length < 13  **FOR** Barcode Length to 13  **Add** ’0’ to Start of Barcode  **ENDFOR**  **ENDIF**  **SEND HTTP** Barcode request to Outpan  **GET** **HTTP** Response from Outpan    //split Response String to extract just name  Name 🡨 NameFromResponseString Response    **IF** Name Starts with "tin"  //does not have a name in Outpan for this product barcode  NameEditBox 🡨 ""  **ELSE**  NameEditBox 🡨 Name  **ENDIF**  //The Barcode and Bought Date cannot be change by the user in the Edit window so Labels  BarcodeEditLable 🡨 Barcode  BroughtDateLable 🡨 todays date |

|  |
| --- |
| **Title** Adding product to database |
| **Description**  This is processed after the User has entered / updated the product details in the Product Add Edit window and selects “OK”  The Pseudo code reads the product the user has entered in the Product Add Edit window to create a product class instance to add into the databases |
| **Pseudo code** |
| // The User has entered / updated the product details in the Product Add Edit window and selects “OK”  // The product Data may have been auto generated for previous instance of this product or from request to Outpan  **IF** ExpiryDateEditBox < todays date  **Display Message Box** with message "Item has already expired! Still add it?" with yes and no buttons  **IF** result = "NO"  **close** dialog  **RETURN** //Wait for user to enter valid expiry\_date  **ENDIF**  //Must be "YES"  **ENDIF**    NewProduct 🡨 **NEW** instance of Product Class  NewProduct.Barcode 🡨 BarcodeEditLabel (removed commas)  NewProduct.Name 🡨 NameEditBox (removed commas)  NewProduct.BoughtDate 🡨 BroughtDateEditBox  NewProduct.ExpiryDate 🡨 ExpiryDateEditBox  NewProduct.MinQuantity 🡨 MinQuantiyEditBox  NewProduct.Price 🡨 PriceEditBox  NewProduct.Location 🡨 LiocationEditBox  NewProduct.Shop 🡨 ShopEditBox  **IF** FavouriteTickBox checked  NewProduct. Favourite 🡨 true  **ELSE**  NewProduct. Favourite 🡨 false  **ENDIF**  UpdateTables with NewProduct    **Display Message Box** with message "Product has been added"    **CLOSE** EditWindow |
| **Title** Updating table after add |
| **Description**  This is processed after a product has been successfully Added to update the Product and Stock Database as well as the Product and Favourites Tables. The other Tables are updated during the add product to database process in UpdateTables |
| **Pseudo code** |
| **IF** NewProduct.Price = 0  NewProduct.Price 🡨 "Blank"  **ENDIF**  Count 🡨 1  **FOREACH** Product in StockDatabase  **IF** NewProduct.Barcode = Product.Barcode  Count 🡨 Count + 1  **ENDIF**  **ENDFOREACH**  **IF** Count = 1  **ADD** NewProduct to StockDatabase containing(NewProduct.Barcode, NewProduct.BoughtDate, NewProduct.ExpiredDate,  NewProduct.Shop, NewProduct.Location, NewProduct.Price)  **ADD** NewProduct to ProductDatabase containing(NewProduct.Name, NewProduct.Barcode, Count, NewProduct.MinQuanty)  **ADD** NewProduct to ProductTable containing (NewProduct.Name, NewProduct.barcode,Count, NewProduct.Price,  NewProduct.Shop, NewProduct.Favourites)  **IF** NewProduct.Favourites = true  **ADD** NewProduct to FavoritesTable containing(NewProduct.Name, NewProduct.barcode,Count, NewProduct.Price,  NewProduct.Shop)  **ENDIF**  **ELSE**  **ADD** NewProduct to StockDatabase containing(NewProduct.Barcode, NewProduct.BoughtDate, NewProduct.ExpiredDate,  NewProduct.Shop, NewProduct.Location, NewProduct.Price)  **ADD** NewProduct to ProductDatabase containing(NewProduct.Name, NewProduct.Barcode, Count, NewProduct.MinQuanty)  **FOREACH** Product in ProductDatabase  **IF** NewProduct.Barcode = Product.Barcode  Product.Quantity 🡨 Count  Product.Name 🡨 NewProduct.Name  Product.MinQuantity 🡨 NewProduct.MinQuantity  **ENDIF**  **ENDFOREACH**    **FOREACH** Product in ProductTable  **IF** NewProduct.Barcode = Product.Barcode  Product.Quantity 🡨 Count  Product.LastPrice 🡨 NewProduct.Name  Product.LastShop 🡨 NewProduct.MinQuantity  Product.Favourites 🡨 NewProduct.Favourites  **ENDIF**  **ENDFOREACH**  **FOREACH** Product in FavouritesTablable  **IF** NewProduct.Barcode = Product.Barcode  Product.Quantity 🡨 Count  Product.LastPrice 🡨 NewProduct.Name  Product.LastShop 🡨 NewProduct.MinQuantity  Product.LastPrice 🡨 NewProduct.Price  **ENDIF**  **ENDFOREACH**  **ENDIF** |

|  |
| --- |
| **Title** Delete |
| **Description**  Item is removed from the stock database and in stock for that item is decreased by one in all other databases |
| **Pseudo code** |
| Count 🡨 0  **FOREACH** Product in StockDatabase  **IF** Barcode = Product.Barcode  Count 🡨 Count + 1  **ADD** Product to DeleteChooseTable containing(Product.Barcode, Product.BoughtDate, Product.ExpiredDate,  Product.Shop, Product.Location, Product.Price)  **ENDIF**  **ENDFOREACH**  **IF** **Count** = 0  **Display Message Box** with message "The barcode returned no results."  **RETURN**  **ELSE**  **SET** Delete Button Enable  **ENDIF**  // user selects Product to be deleted from DeleteChooseTable and selects Delete Button    **DETETE** selected Product form StockDatabase  **FOREACH** Product in ProductDatabase  **IF** Barcode = Product.Barcode  Product.Quantity 🡨 Count - 1  **ENDIF**  **ENDFOREACH**  **FOREACH** Product in ProductTable  **IF** Barcode = Product.Barcode  Product.Quantity 🡨 Count - 1  **ENDIF**  **ENDFOREACH**    **CLEAR** DeleteChooseTable  UpdateTables |

## Class Design



# Testing

Test the system works correctly and can handle all user inputs (both correct and incorrect)

## Input and Output Testing

The following table tests the expected outcome against the actual outcome for user input to system output for typical (correct and expected) data, and erroneous data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 1** | Adding a product | | | |
|  | **Test Data** | **Expected Result** | **Test Result** | **Comments** |
|  | Enter full valid data | Product is added and window closes  All Table Views Updated Correctly | Pass |  |
|  | Name empty  with all other data valid | Message box reporting the name is empty | Pass |  |
|  | Expiry Date not selected  with all other data valid | Message box reporting the expiry date is invalid | Pass |  |
|  | Location empty  with all other data valid | Product is added and window closes  All Table Views Updated Correctly except the With the Product Location shown as blank | Pass |  |
|  | Shop listed as “none”  with all other data valid | Product is added and window closes  All Table Views Updated Correctly except the With the Product Shop shown as blank | Pass |  |
|  | Price left at "0.00"  with all other data valid | Product is added and window closes  All Table Views Updated Correctly except the With the Product Price shown as blank | Pass |  |
|  | Location empty  Shop listed as “none”  Price left at "0.00"  with all other data valid | Product is added and window closes  All Table Views Updated Correctly except the With the Product Shop, Location and Price shown as blank | Pass |  |
|  | Enter Name “pasta, by Tesco’s”  with all other data valid | Product is added and window closes  All Table Views Updated Correctly except the With the Product Name show without commas | Pass | CSV files use commas to split the values, commas in description would confuse the system and so had to be omitted. (See figure 1) |
|  | Enter a Expiry Date which has already passed the current date | Message box reporting the product has already expired and provides the option to cancel or to continue and add the product | Pass | Some products have overly pessimistic expiry dates and people can keep the products up to a week after the expiry date. Other people buy expiring products as they are normally on discount.(See figure 2) |
| Select “Cancel” and can correct the Expiry Date |
| Select “OK” and the Expiry Date is accepted and the product is immediately displayed on the Expired Table |
|  | Enter barcode "0123456789asdf" | Message box reporting this is not a valid barcode. | Pass |  |
|  | Enter barcode "123456789" | The barcode should be extended to 13 digits with ‘0’ s at the start  "0000123456789"  As this is not a valid barcode **Outpan** wont find this barcode and the product name will be blank  When the product name is entered with the rest of the data this product can be added | Pass | **Outpan** contains quite a few invalid barcodes, this could be due to **Outpan** being open source. Allowing anyone to entered false information |
|  | Select favourite list box  with all Product data valid | Product is added and window closes  All Table Views Updated Correctly including product being displayed on favourite table | Pass | (See figure 3) |
|  | Select “Has no barcode” and enter a name instead of barcode  with all Product data valid | Product is added and window closes  All Table Views Updated | Pass |  |
|  | Select “Has no barcode” and enter the same name in different case  with all Product data valid | Product is added and window closes  All Table Views Updated and show there is a quantity of two for this product | Pass |  |

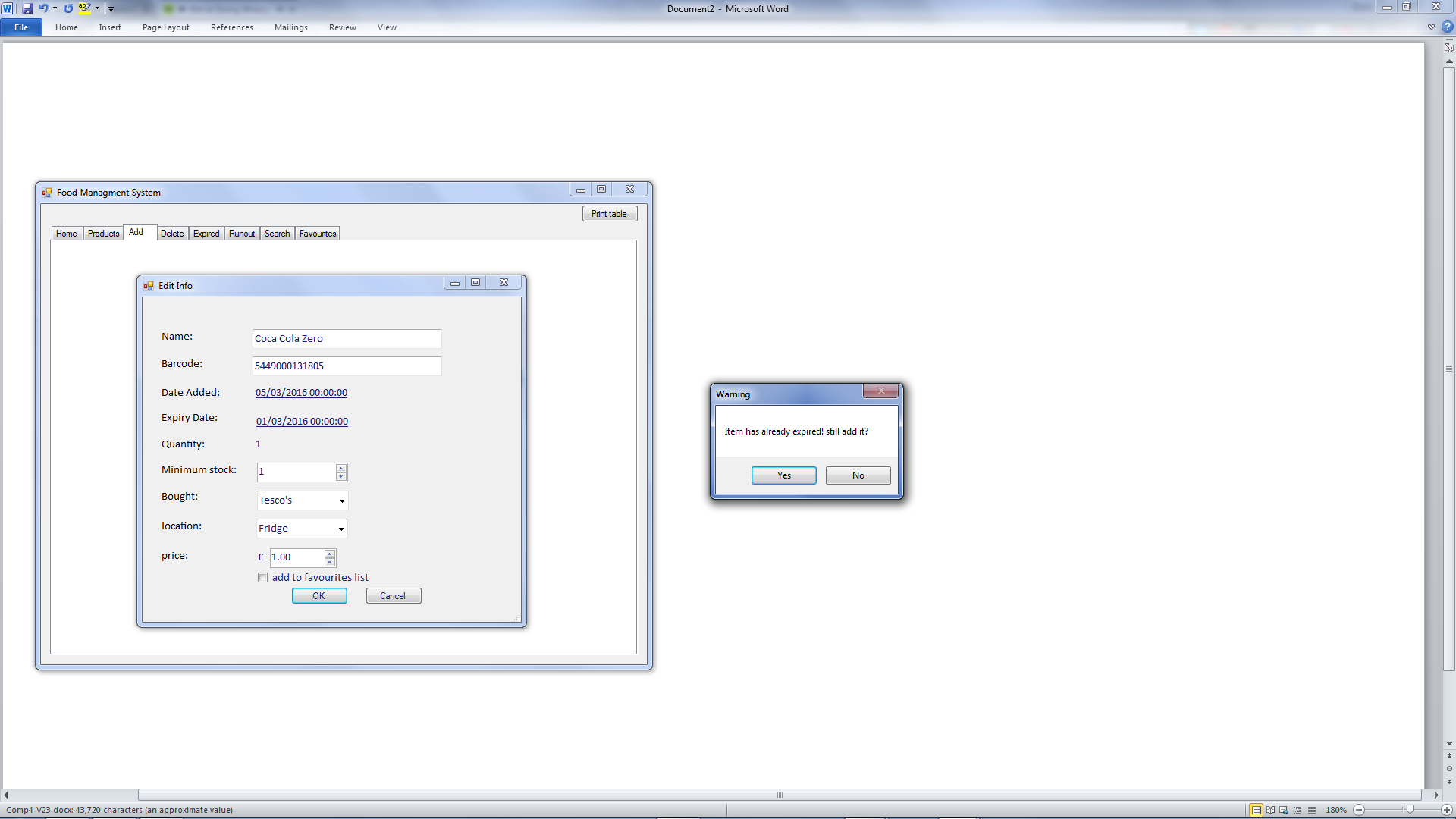
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 2** | Deleting a product | | | |
|  | **Test Data** | **Expected Result** | **Actual Result** | **Comments** |
|  | Enter a product barcode for a product that currently has a quantity two or more for the product | list of products with that barcode are displayed | Pass |  |
|  | Enter a product barcode for a product that currently has a quantity one for the product |  | Pass |  |
|  | Enter a product barcode for a product is currently not on the system | Message box reporting product not found | Pass |  |
|  | Enter a product Name for a product that was added using a name |  | Pass |  |
|  | Enter a empty product barcode | nothing | Pass |  |
|  | From the Product Table select a product to delete |  | Pass | (See figure 4) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test 3** | Search products | | | |
|  | **Test Data** | **Expected Result** | **Actual Result** | **Comments** |
|  | Enter a search string which is a sub-string of more than one product currently in stock | list of products containing the search term are displayed | Pass | (See figure 5) |
|  | Enter the same search string in a different case |  | Pass |  |
|  | Enter a empty search string | nothing | Pass |  |
|  | Enter a search string which is NOT a sub-string any product currently in stock | Message box reporting product not found | Pass |  |

## Tracking Updates in the Stock and Product databace

The following table tests the expected outcome against the actual outcome for user input to system output for typical (correct and expected) data, and erroneous data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Searching a product  Test No 4 | | | | |
| Test Data | Expected Result | Actual Result | Comments |
| enter search term which is one of the items in database | list of products containing the search term are displayed | As expected | (See figure 5) |
| nothing entered | nothing | As expected |  |
| non existing product barcode entered | Message box saying product not found | As expected |  |



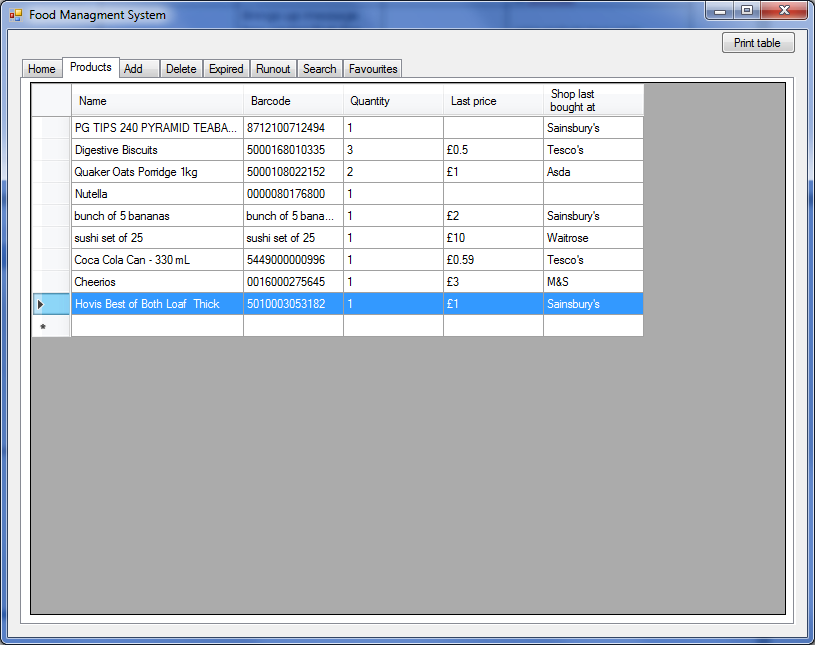
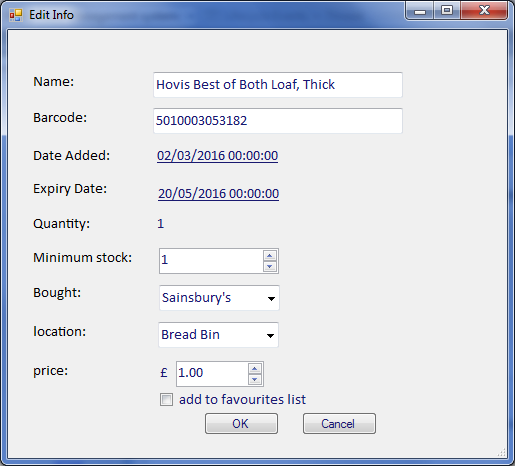


Figure 1

Figure 2

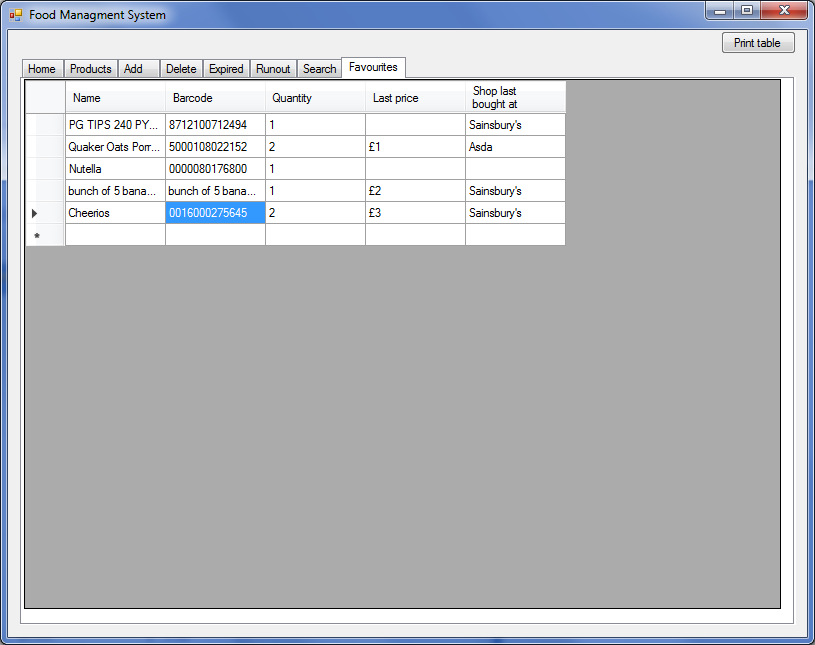
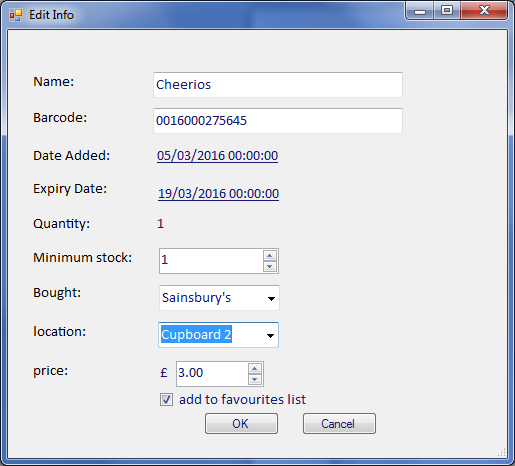


Figure 3

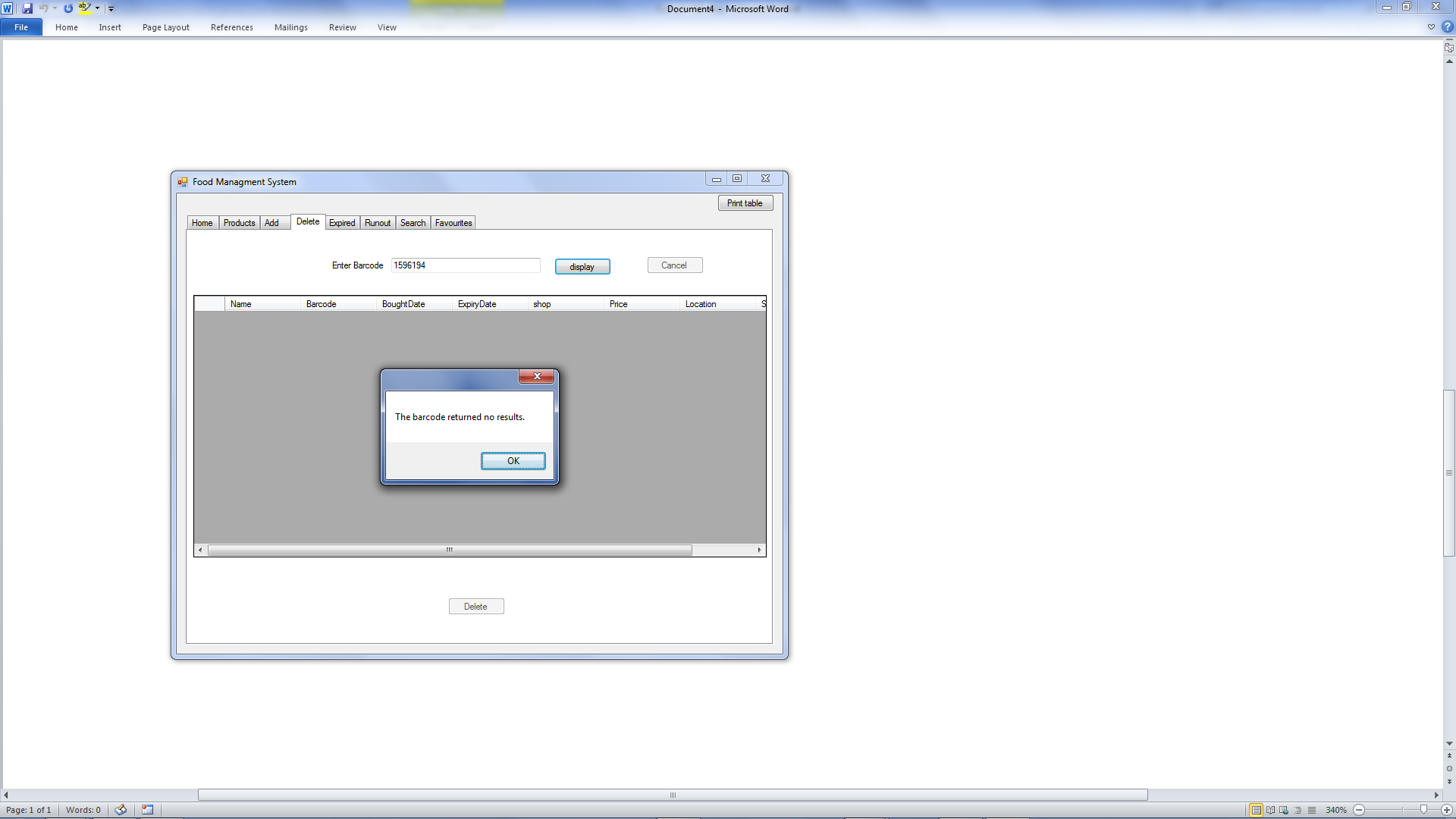
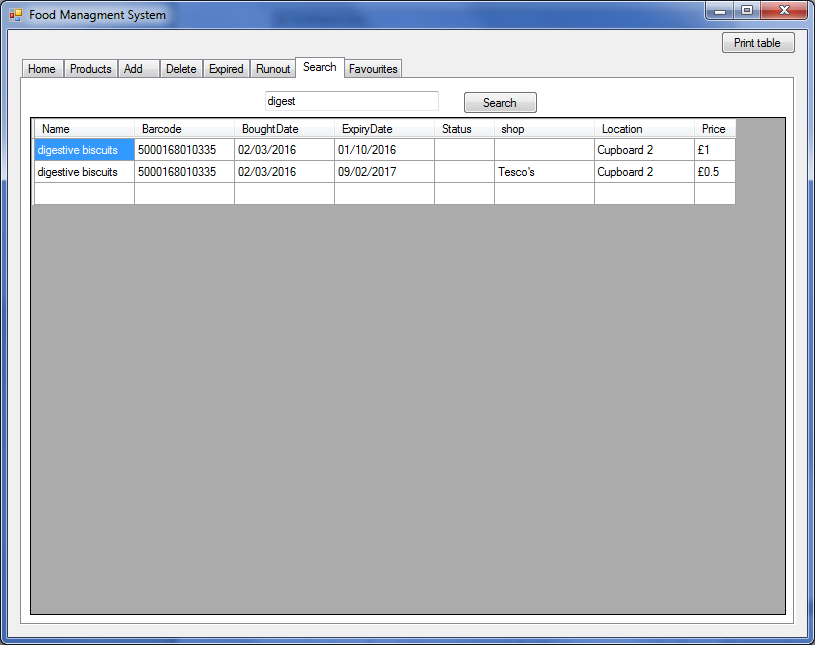


Figure 5

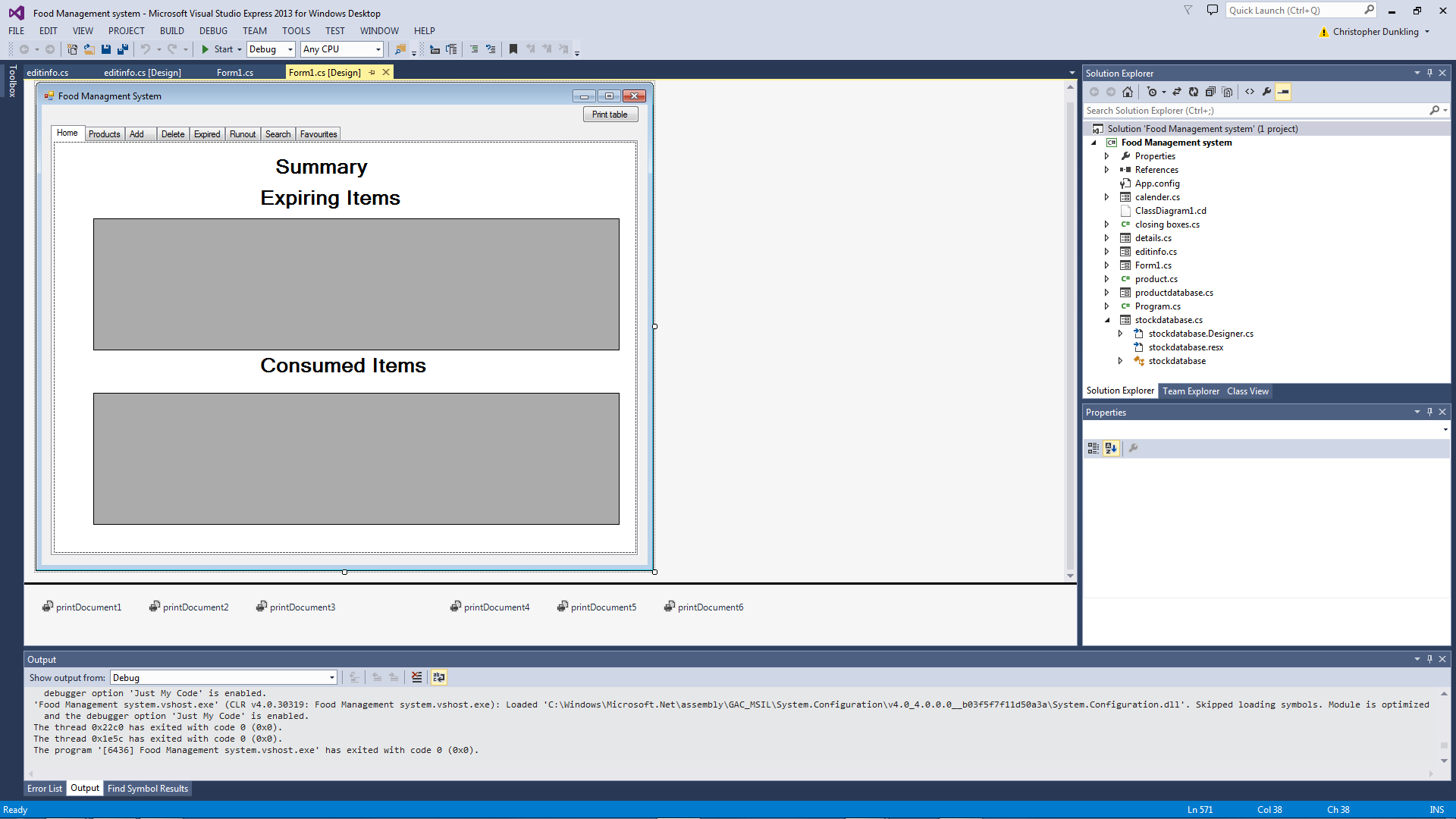
Figure 4

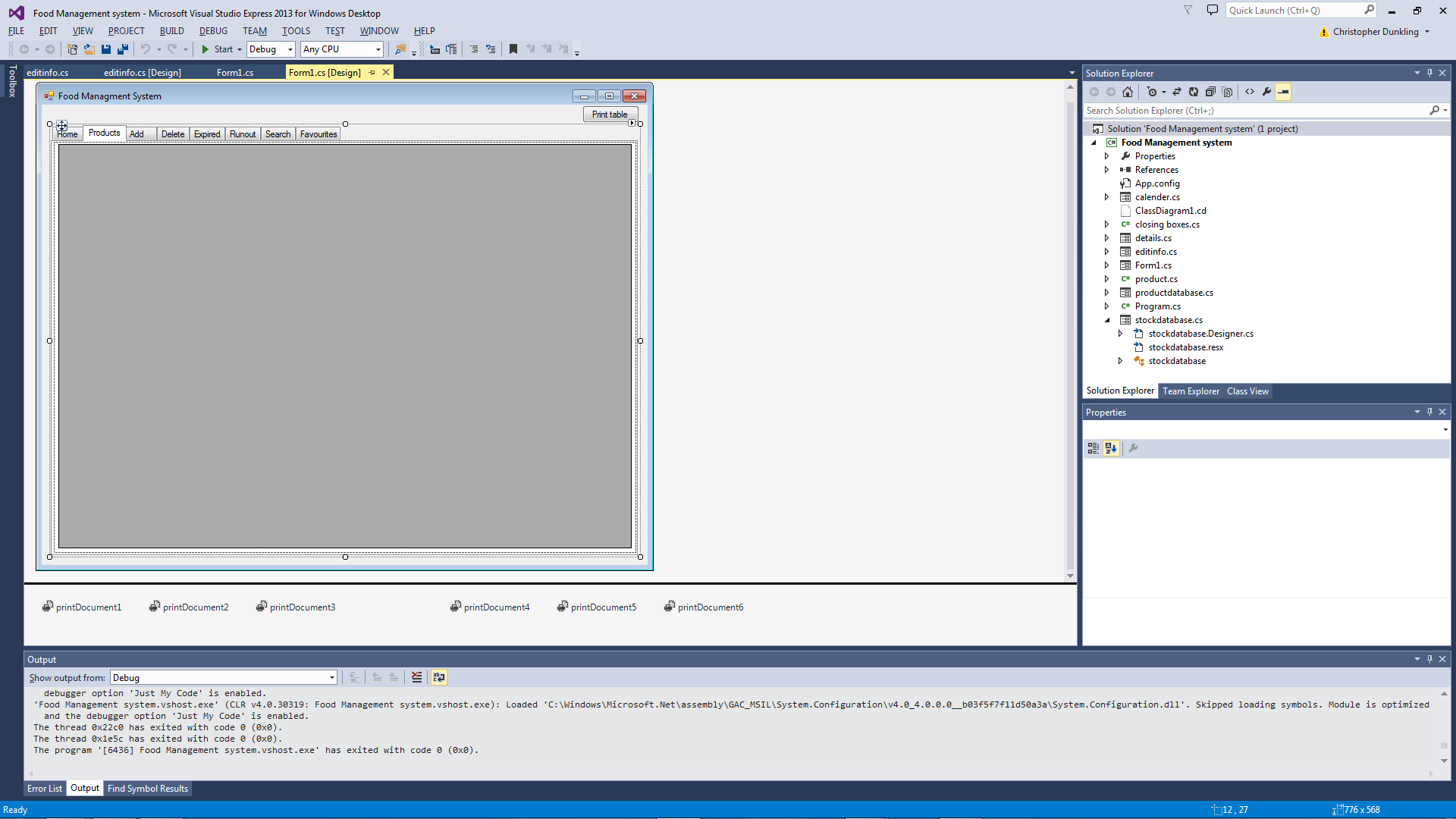
# Maintenance

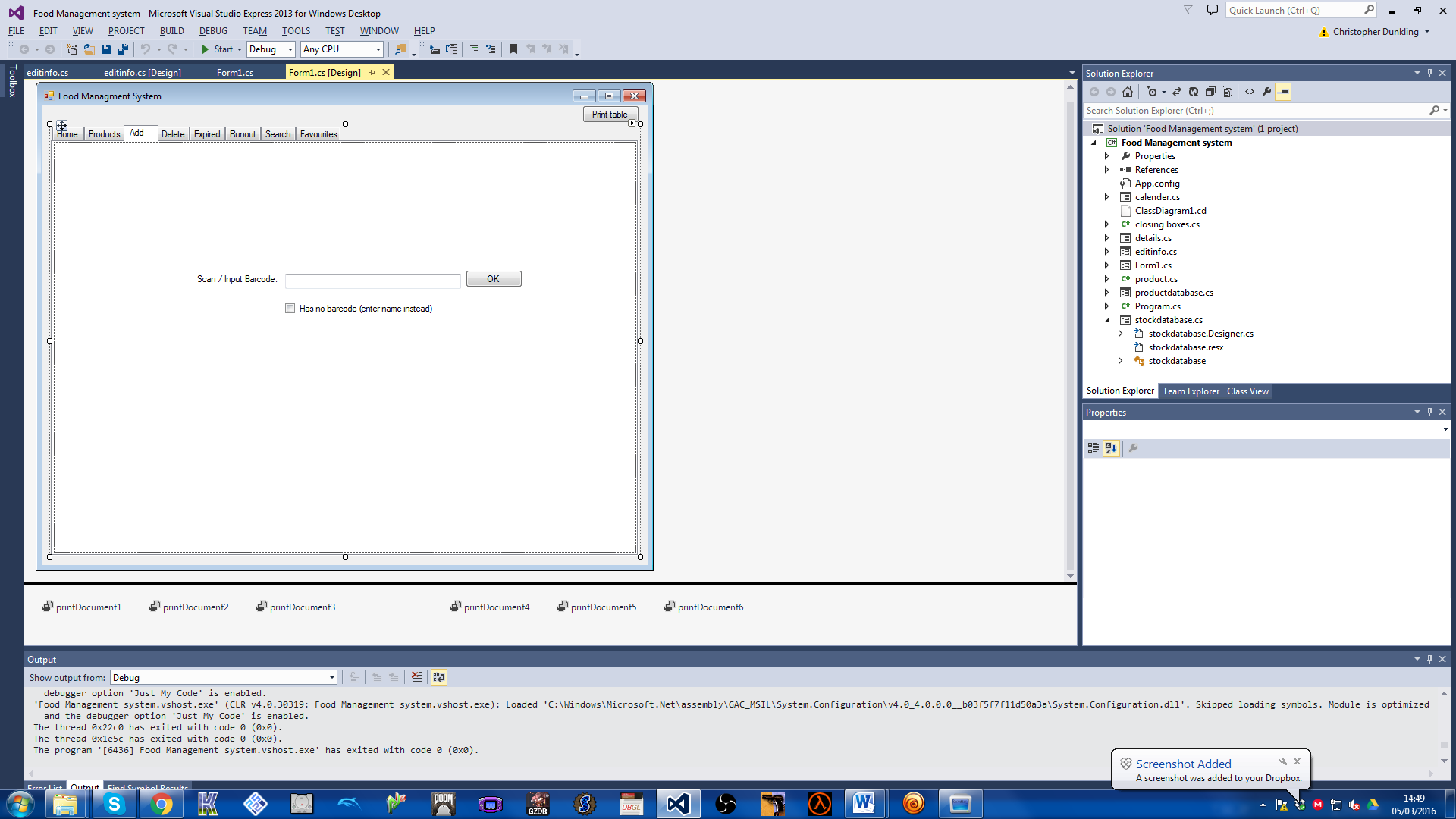
|  |
| --- |
| Code for Outpan Request |
| if (barcodeinput.Text != "")  {  input = barcodeinput.Text;  //this makes sure that the barcode has 13 digits by heading 0s to fill  if (input.Length < 13)  {  for (int x = input.Length; x < 13; x++)  {  input = "0" + input;  }  }  //Http request using the outpan api to scan their database for information on barcode  var request = HttpWebRequest.Create("https://api.outpan.com/v1/products/" + input);  //outpan require the user details this encoded string is my unique code to access the system  var encodedString =  Convert.ToBase64String(Encoding.Default.GetBytes("c96c011b6746445555dd5614d1572114:"));  request.Headers["Authorization"] = "Basic " + encodedString;  var response = request.GetResponse();  //response is requested on a data stream  Stream dataStream = response.GetResponseStream();  StreamReader reader = new StreamReader(dataStream);  string responseFromServer = reader.ReadToEnd();  //The response is received in the form of fields and their “response”.  //This means that it needs to be split up so that the name can be extracted  int first = responseFromServer.IndexOf("\"name\": \"") + "\"name\": \"".Length;  string str2 = responseFromServer.Substring(first);  int last = str2.IndexOf("\",\n");  string str3 = str2.Substring(0, last);  //initialize auto fill data for edit window  //if the string starts with "tin"  //this indicates that there is no details in outpans database for this barcode.  if (str3.StartsWith("tin"))  {  str3 = ""; //no name  newproduct.setbarcode(input);  newproduct.setname(str3);  newproduct.setboughtdate(DateTime.Now);  newproduct.setstatus("");  reader.Close();  dataStream.Close();  response.Close();  barcodeinput.Clear();  }  else  {  newproduct.setbarcode(input);  newproduct.setname(str3); //name from Outpan  newproduct.setboughtdate(DateTime.Now);  newproduct.setstatus("");  barcodeinput.Clear();  }  }  //Open Edit window  Editinfo frm = new Editinfo();  frm.ShowDialog(); |

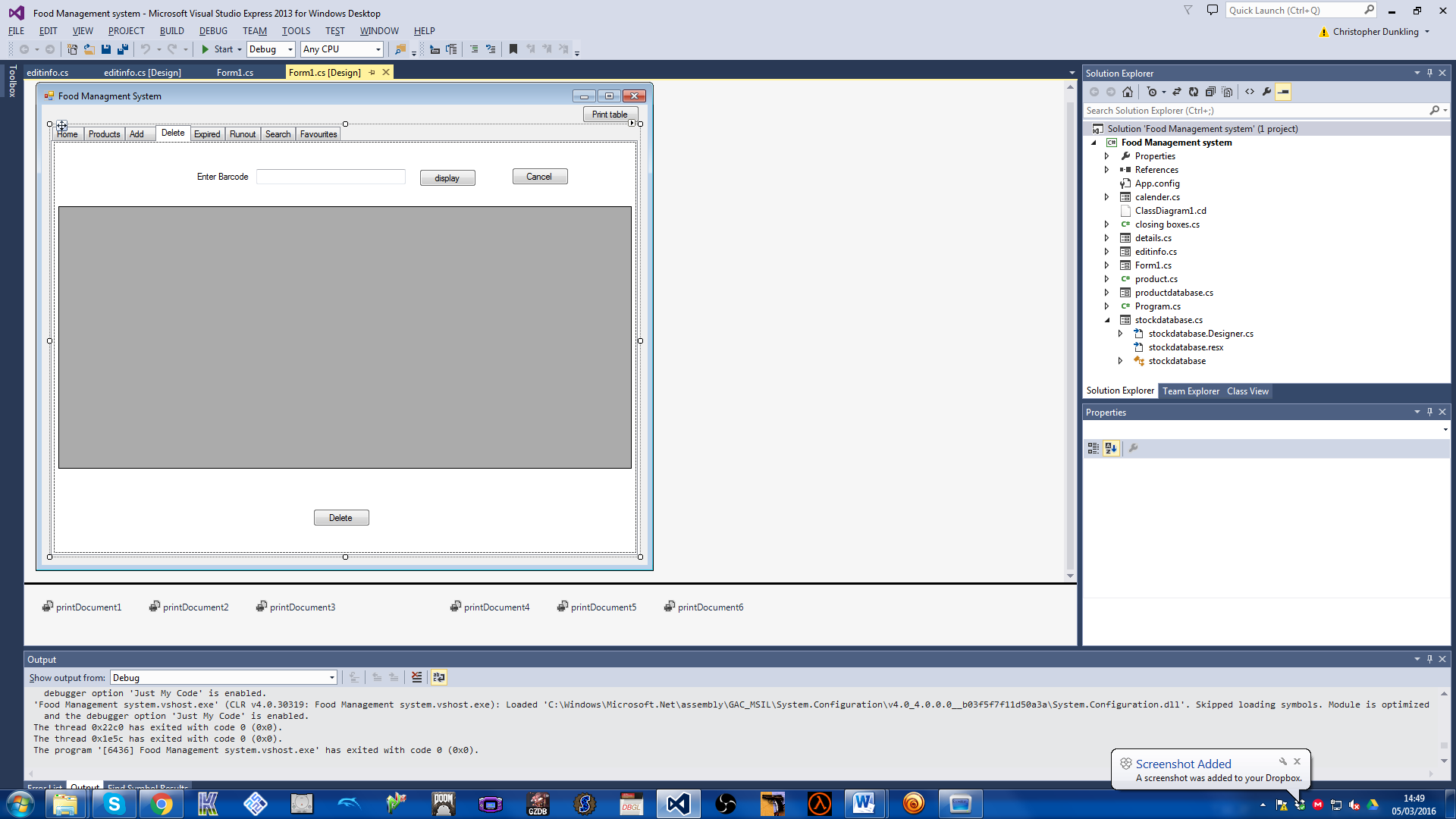
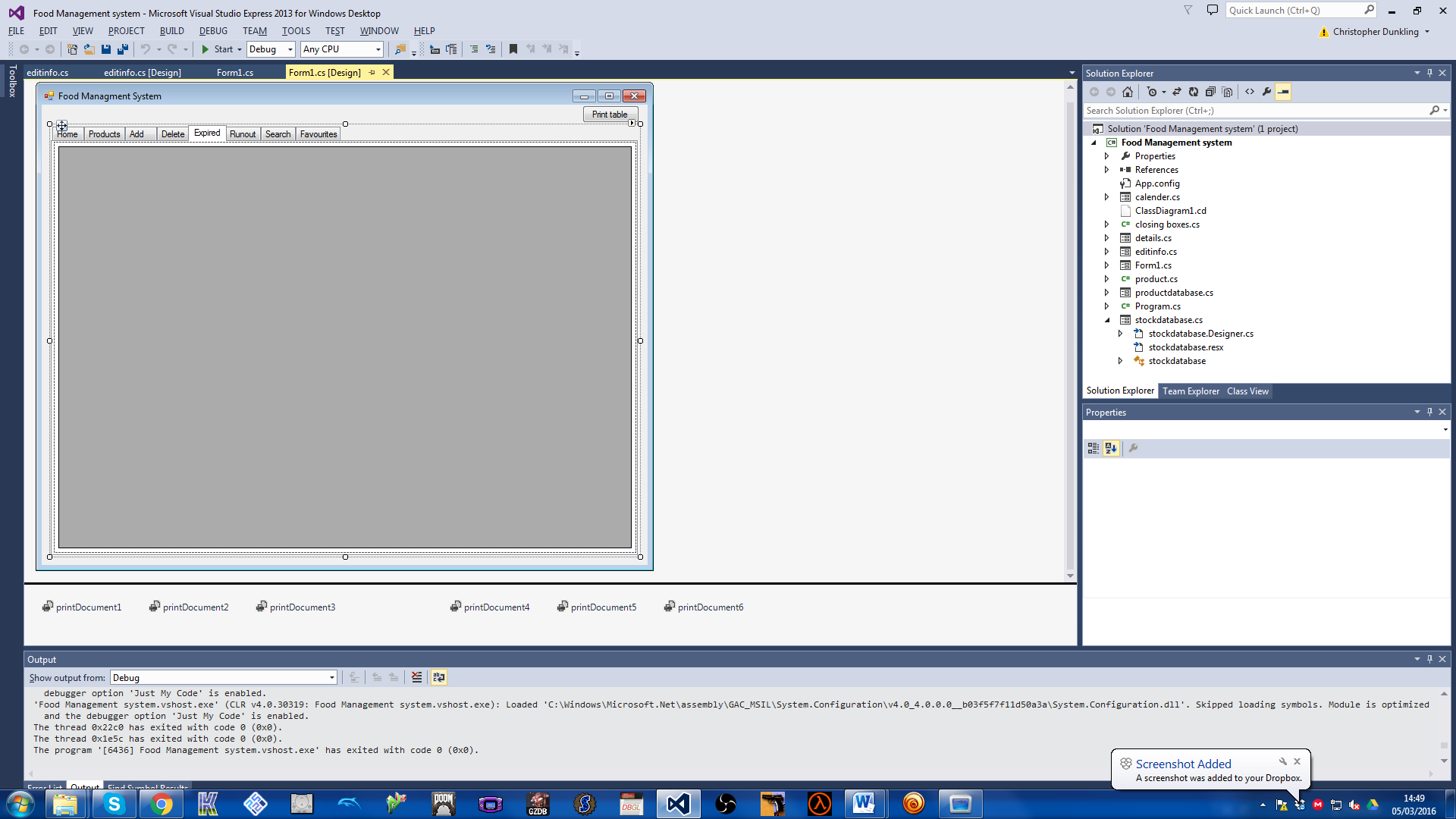
|  |
| --- |
| Code for Delete Request |
| // find products that matches the barcode and adds to delete table  private void button4\_Click(object sender, EventArgs e)  {  input = textBox1.Text;  if (input.Length < 13)  {  for (int x = input.Length; x < 13; x++)  {  input = "0" + input;  }  }  foreach (DataRow dr in stockdatabase.stocktable.Rows)  {  string rowValue = dr["Barcode"].ToString();  if (rowValue == input)  {  string[] fullarray = new string[9];  foreach (DataRow dx in productdatabase.producttable.Rows)  {  string rowValue2 = dx["Barcode"].ToString();  if (rowValue2 == input)  {  fullarray[0] = dx["Name"].ToString();  break;  }  }  fullarray[1] = textBox1.Text;  fullarray[2] = dr["BoughtDate"].ToString();  fullarray[3] = dr["expirydate"].ToString();  fullarray[4] = dr["shop"].ToString();  fullarray[5] = dr["Price"].ToString();  fullarray[6] = dr["Location"].ToString();  fullarray[7] = dr["Status"].ToString();  fullarray[8] = dr["position"].ToString();  deleteitems.Rows.Add(fullarray);  }  }  if (deleteitems.Rows.Count == 0)  {  MessageBox.Show("The barcode returned no results.");  }  else  {  deletebutton.Enabled = true;  }  }  // uses details from selected row to decrease the quantity  private void button3\_Click(object sender, EventArgs e)  {  string index = deleteitem.SelectedRows[0].Cells["position"].Value.ToString();  int indexnum = Int32.Parse(index);  string barcode = deleteitem.SelectedRows[0].Cells["Barcode"].Value.ToString();  int value;  foreach (DataRow dr in MainWindow.table.Rows)  {  if (barcode == dr["Barcode"].ToString())  {  value = Convert.ToInt32(dr["Quantity"]);  value -= 1;  stockdatabase.stocktable.Rows.RemoveAt(indexnum);  MainWindow.table.Columns["Quantity"].ReadOnly = false;  dr["Quantity"] = value;  foreach (DataRow dx in productdatabase.producttable.Rows)  {  if (barcode == dx["Barcode"].ToString())  {  dr["Quantity"] = value;  }  }  MainWindow.table.Columns["Quantity"].ReadOnly = true;  MainWindow.statuscheck();  deleteitems.Clear();  break;  }  }  } |

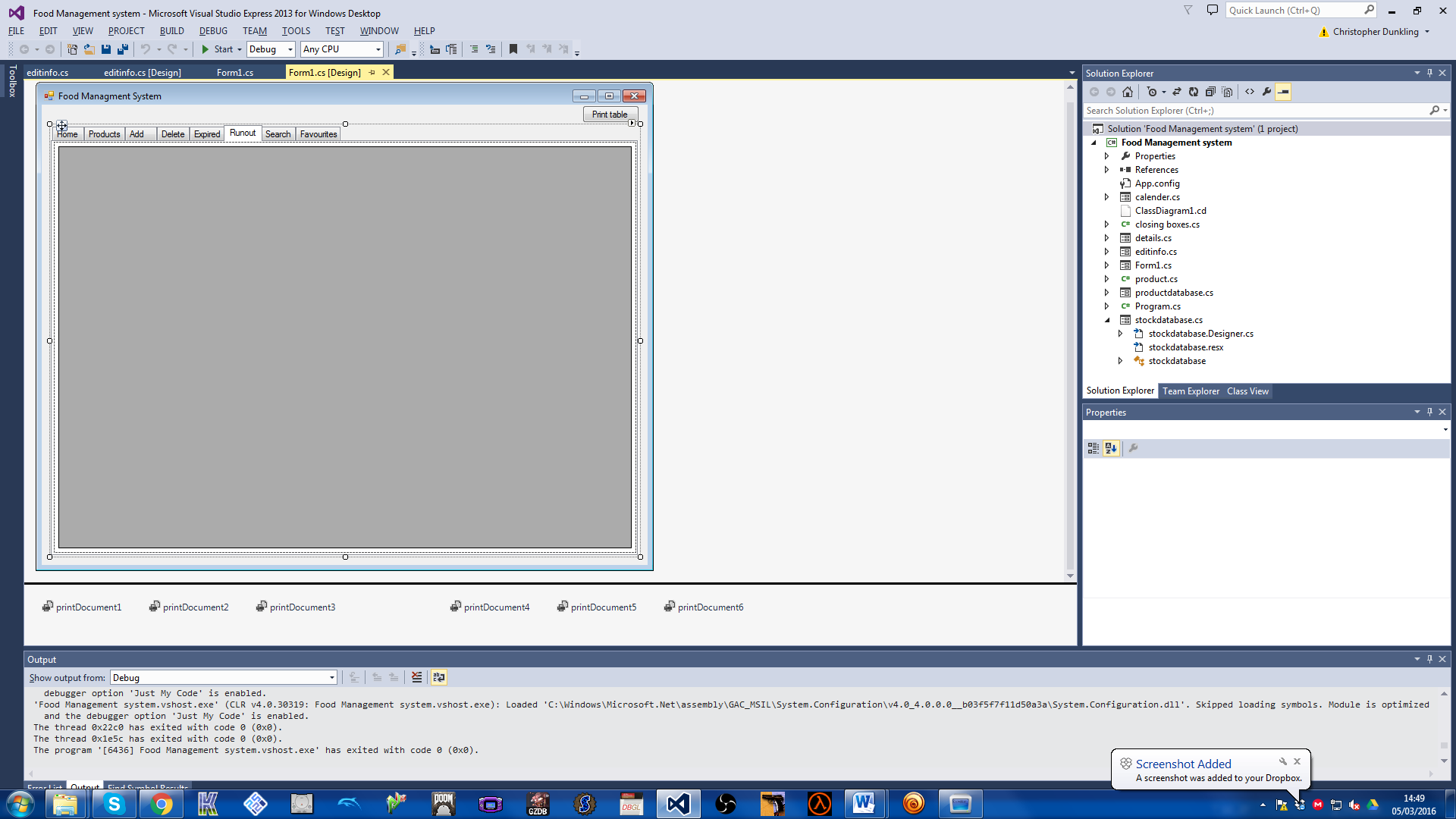
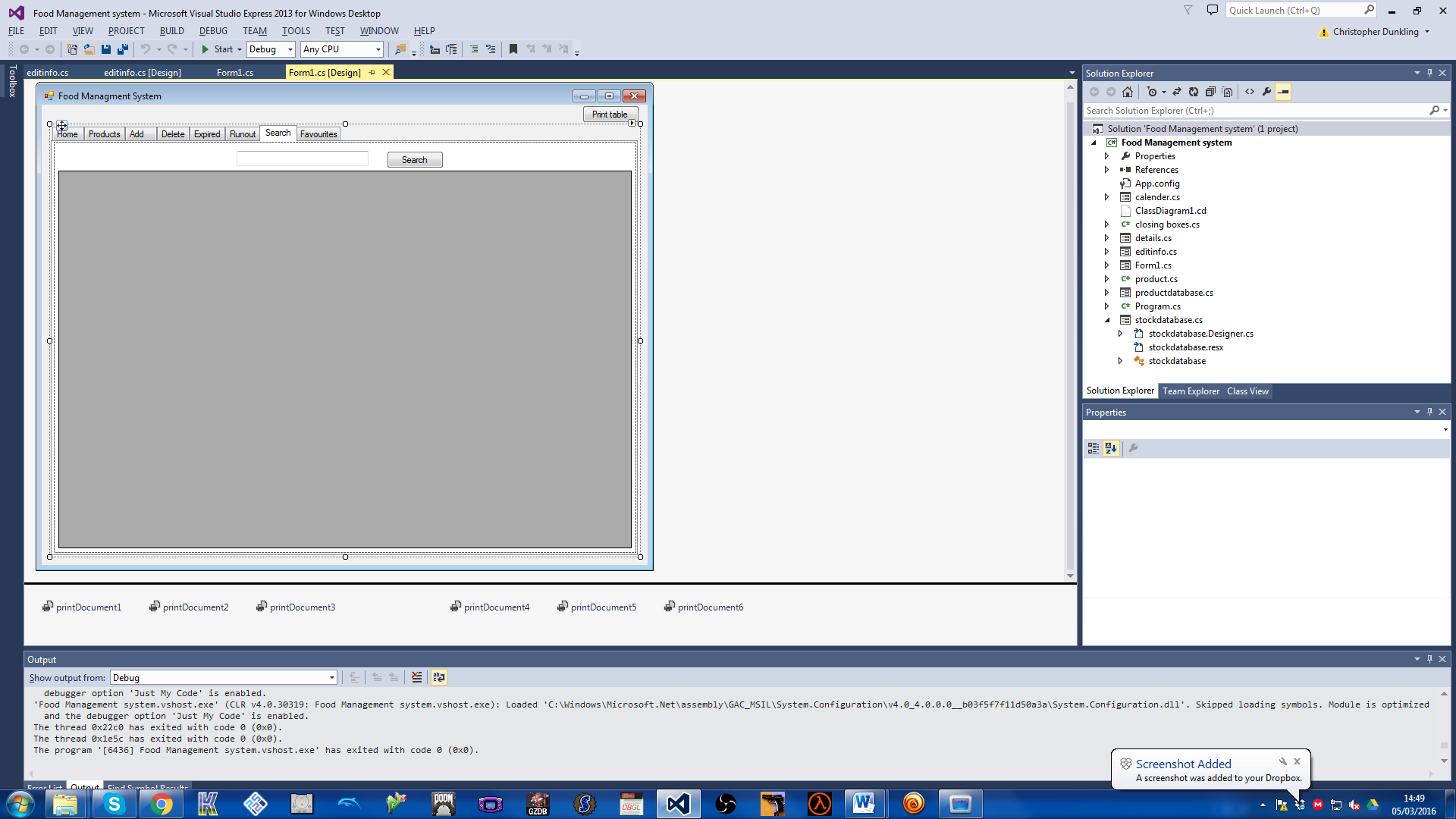
### Forms in design view

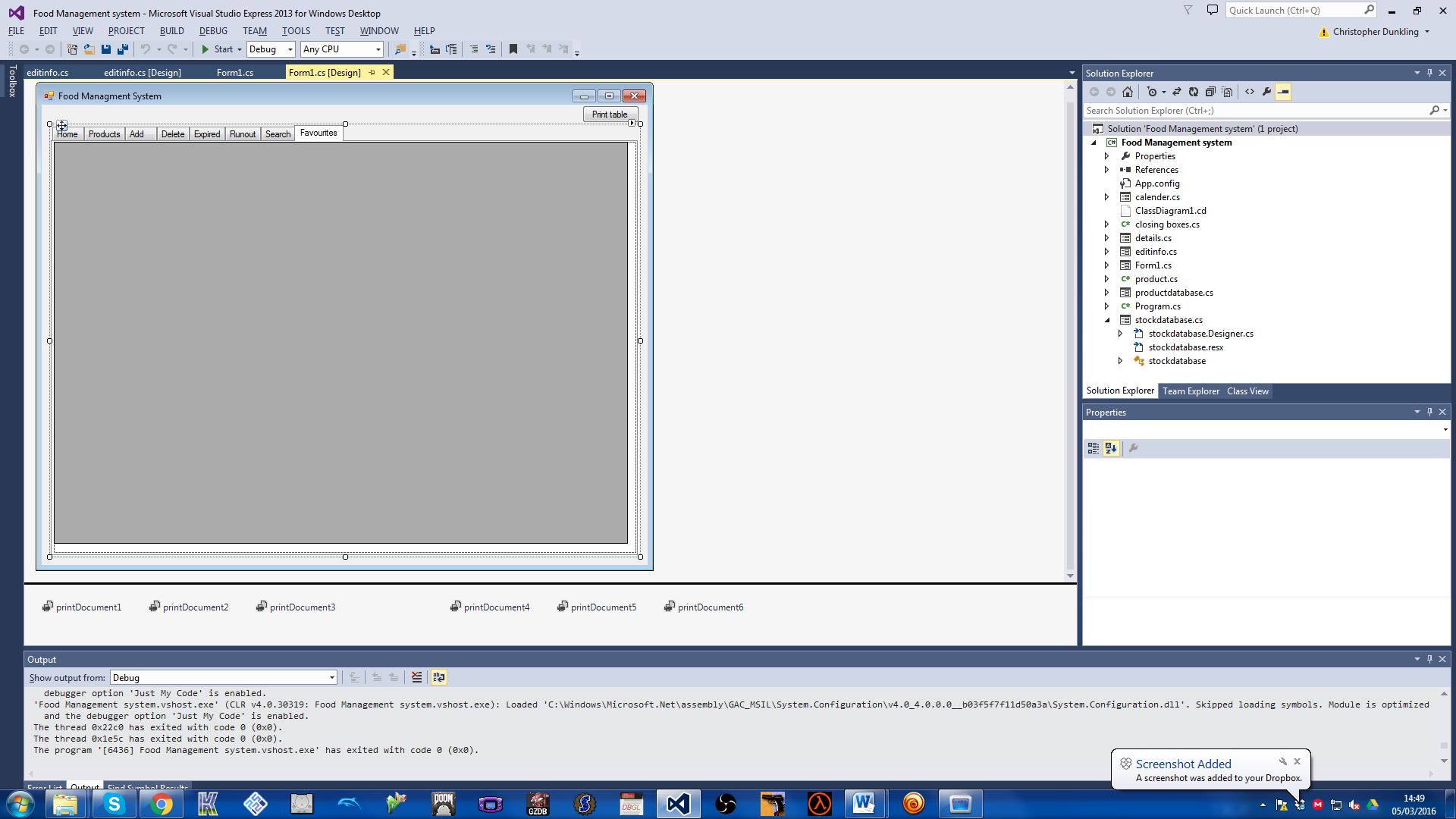


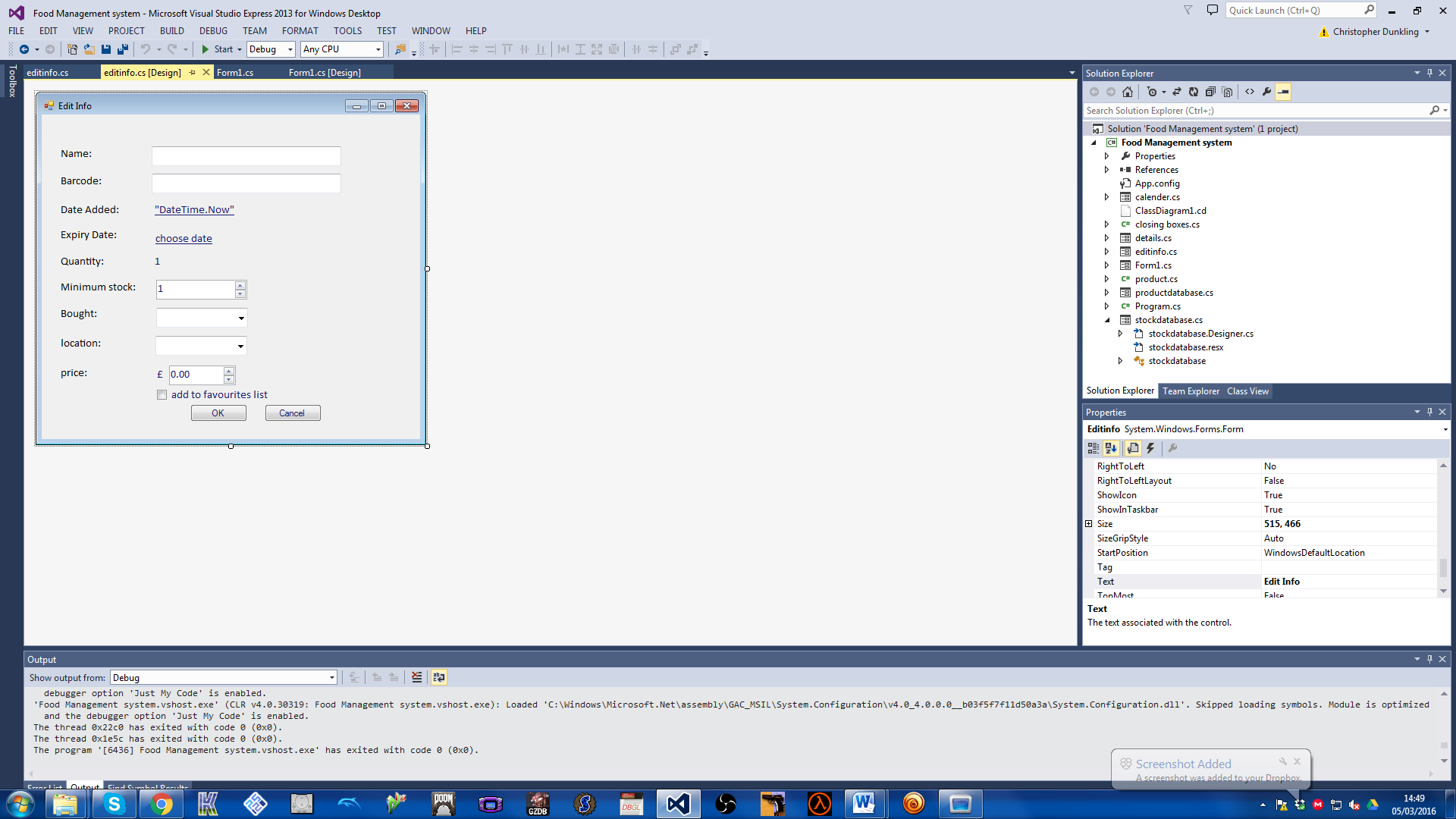












# User Manual

|  |
| --- |
| Food Management System  User Manual |
| By Christopher Dunkling |

Contents

[Analysis 3](#_Toc445174481)

[Background Identification Of The Problem 3](#_Toc445174482)

[How A Computerised System Can Help With This Problem? 3](#_Toc445174483)

[Description of Current System 4](#_Toc445174484)

[Identification of Prospective User 4](#_Toc445174485)

[Would Anyone Be Interested In A Computerised System? 4](#_Toc445174486)

[Primary User 5](#_Toc445174487)

[Identification of User Requirements 6](#_Toc445174488)

[Answers to the Questionnaire – My mother 6](#_Toc445174489)

[Answers to the Questionnaire- My aunt’s family of six 7](#_Toc445174490)

[Answers to the Questionnaire- Friends family of four 7](#_Toc445174491)

[Answers to the Questionnaire - Grandparents 8](#_Toc445174492)

[Review of Similar Already on the Market 9](#_Toc445174493)

[Data Sources and Destinations 10](#_Toc445174494)

[Analysis Data Directory 11](#_Toc445174495)

[Data volumes 11](#_Toc445174496)

[Logical Data Flow Diagrams 12](#_Toc445174497)

[E-R model 15](#_Toc445174498)

[Identification of Objects 15](#_Toc445174499)

[Objectives 16](#_Toc445174500)

[Appraisal of Potential Solution 18](#_Toc445174501)

[Objectives and limitation 18](#_Toc445174502)

[Architecture 18](#_Toc445174503)

[Computer Platform 18](#_Toc445174504)

[Program Language 19](#_Toc445174505)

[Implementation 19](#_Toc445174506)

[Design 20](#_Toc445174507)

[Overall System Design 20](#_Toc445174508)

[Fundamental Implementation objectives 20](#_Toc445174509)

[The Prototype Stage Of Design 20](#_Toc445174510)

[Application Design Decisions 21](#_Toc445174511)

[IOPS 24](#_Toc445174512)

[Top Level System Flow Diagram 25](#_Toc445174513)

[Food Item Add Flow Diagram 26](#_Toc445174514)

[Food Item Delete Flow Diagram 27](#_Toc445174515)

[Descriptions Of Modular Structure Of System 28](#_Toc445174516)

[Process Diagram 28](#_Toc445174517)

[Database Design 29](#_Toc445174518)

[Object Orientated Planning 31](#_Toc445174519)

[Product Class 31](#_Toc445174520)

[Definition Of Data Requirements 32](#_Toc445174521)

[Stepwise Refinement 33](#_Toc445174522)

[User Interface Design 34](#_Toc445174523)

[Form Navigation 41](#_Toc445174524)

[Algorithm 42](#_Toc445174525)

[Class Design 47](#_Toc445174526)

[Testing 48](#_Toc445174527)

[Input and Output Testing 48](#_Toc445174528)

[Adding a product 48](#_Toc445174529)

[Deleting a product 50](#_Toc445174530)

[Search products 51](#_Toc445174531)

[Tracking Updates in the Stock and Product databace 51](#_Toc445174532)

[Maintenance 56](#_Toc445174533)

[Code for Outpan Request 56](#_Toc445174534)

[Code for Delete Request 57](#_Toc445174535)

[Forms in design view 59](#_Toc445174536)

[User Manual 64](#_Toc445174537)

[Using the program 69](#_Toc445174538)

[Getting Started 69](#_Toc445174539)

[Functions 73](#_Toc445174540)

[**Adding items (which have a barcode)** 73](#_Toc445174541)

[**Adding items (which do not have a barcode)** 74](#_Toc445174542)

[**Deleting items** 75](#_Toc445174543)

[**Searching for items** 76](#_Toc445174544)

[**Details on product screen** 76](#_Toc445174545)

[Appraisal 77](#_Toc445174546)

[Comparison of Project Performance Against Objectives 77](#_Toc445174547)

[Potential for Future Developments 80](#_Toc445174548)

[User Feedback 80](#_Toc445174549)

**Introduction**

According to latest government reports *(Placeholder1)* the average UK family is wasting nearly £60 a month by throwing away food, that accumulates to 7 million tonnes of food in the every year of which 4.2 million tonnes is going straight from fridges and cupboards into the bin. This is wasting a valuable resources and harming the environment

Researchers have found:

With the modern busy life style most food we buy is based on what we think we need or may have run out of plus what’s on promotion, this means a lot of food is buy we already have and conversely forget to buy food we have run out of.

The food is then put in the relevant in cupboards, fringes and freezers sometimes in an organised way but many cases almost random

With the ever increasing amounts of food stored in the house, it becomes difficult to remember what food we do have, were it is stored and most importantly when it will expire, so some food gets completely forgotten about or we forget how long we have had it for and it expires and has to be thrown out.

User Manual Page 2

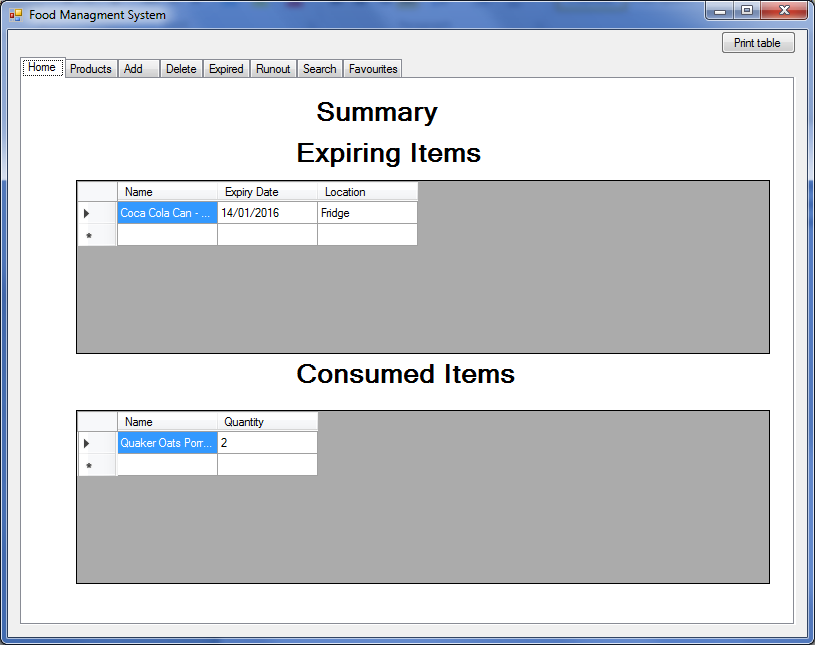
**Installation**

1. Run the exe file
2. Select where you want to store the program
3. Start the shortcut on desktop.

User Manual Page 3

**Using the program**

**Getting Started**

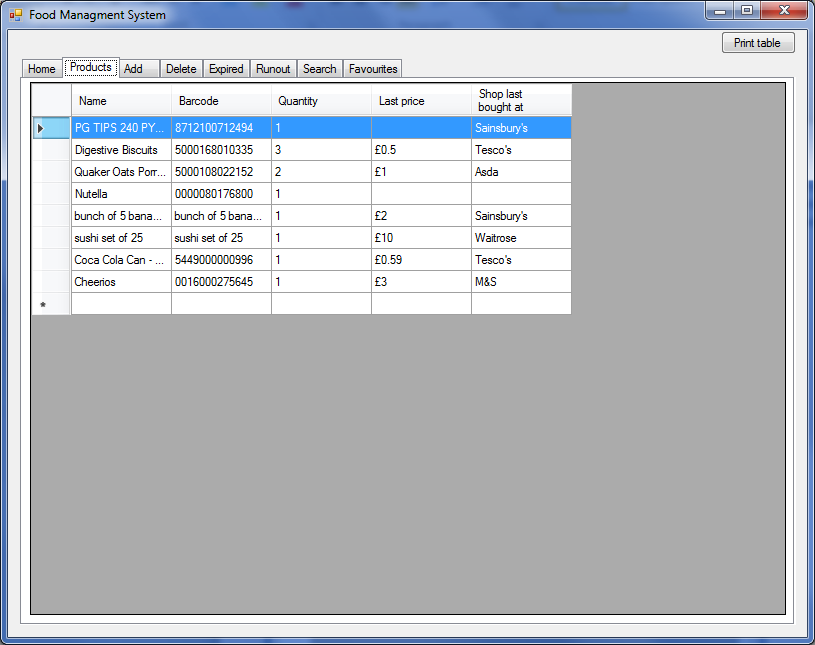
When you first boot up the program you will be shown this screen:

You have seven tabs: Home, Products, Input, Delete, Expiring, Runout and search

**Home** (above)

This is the main tab that you see every time you start the program and gives a summary on the expiring items and consumed items.

**Products** (below)

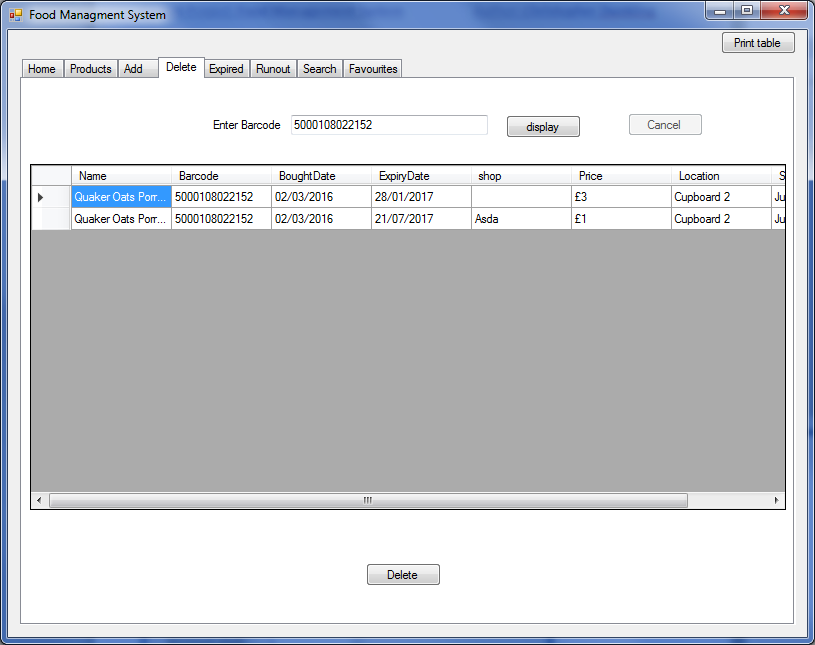
This shows all the products you have and their quantities. It will also display the last price and last shop bought from.

User Manual Page 4

**Add** (below)

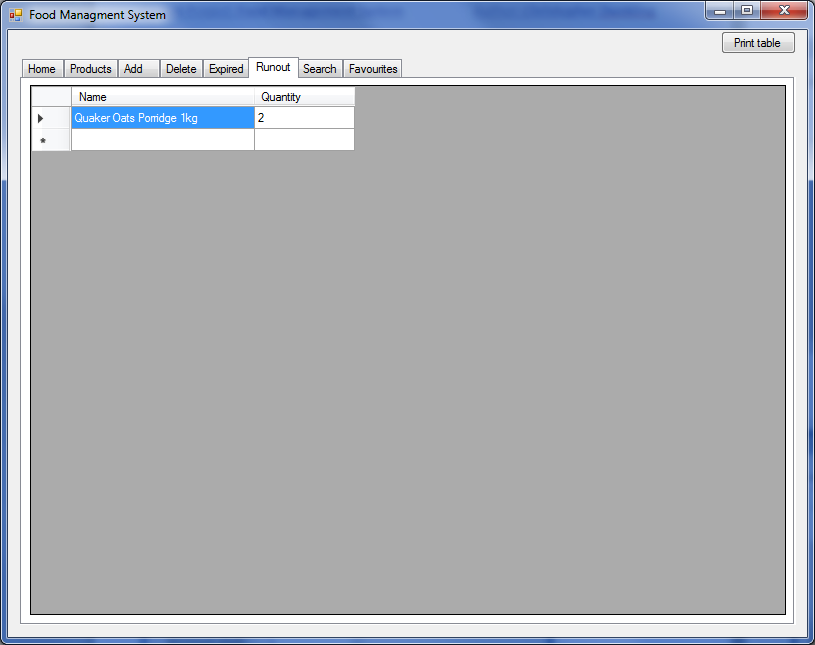
This is the tab which you use to add products to the database which can be added via scanning/manual input

**Delete** (below)

This is the tab which you use to delete stock. This will decrease the quantity and remove the selected item.

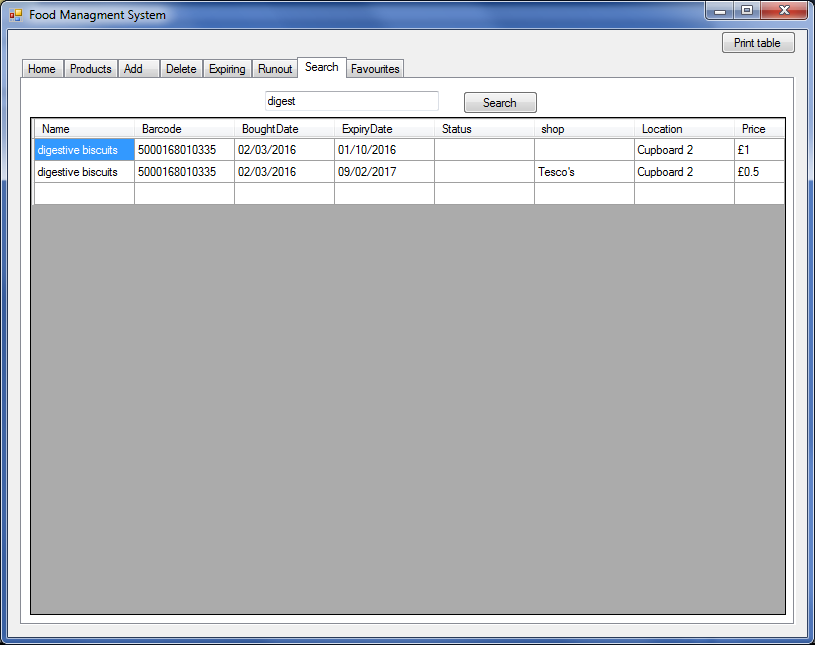
User Manual Page 5

**Expiring and Runout** (Below)

These tabs warn about items that are low in quantity or are reaching/reached their expiry date

**Search** (below)

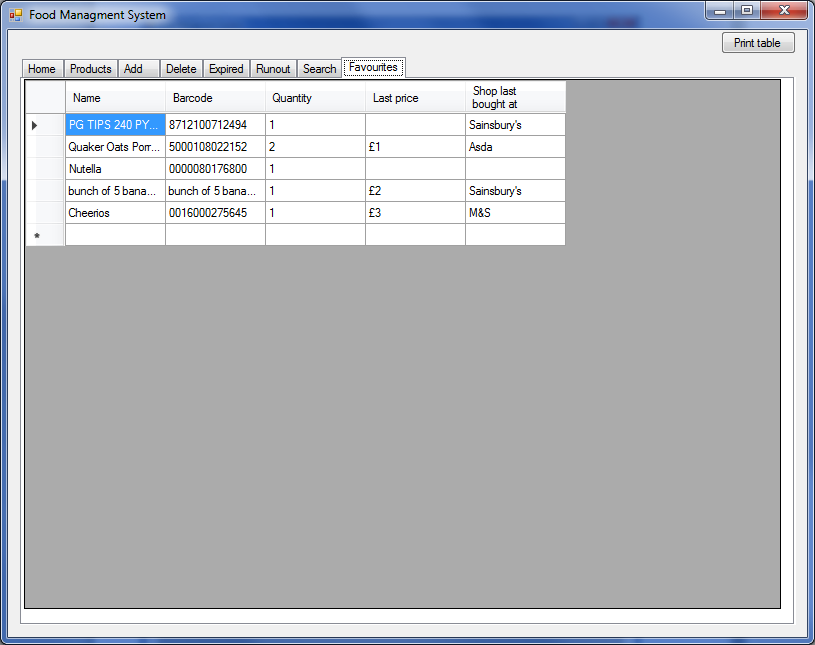
This tab allows the user to search for products



User Manual Page 6

**Favourites** (below)

Contains the products that have been marked as favourite so they can be tracked easier



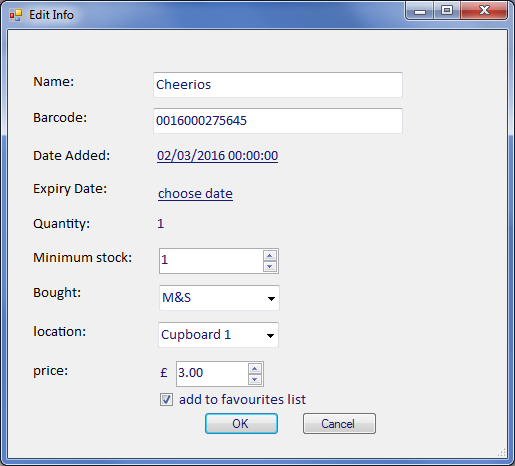
User Manual Page 7

**Functions**

**Adding items (which have a barcode)**



1. Navigate to input tab
2. Scan/input barcode
3. You will then be presented with this screen:



User Manual Page 8

1. You need to fill in:

|  |  |
| --- | --- |
| **Name** | anything can be inputted but commas will be removed |
| **Barcode** | automatically filled in from the scanned barcode |
| **Date added** | date is selected by using the popup calendar. Defaults to today’s date |
| **Expiry date** | works the same as date added. It will ask when expiry date is before today’s date |
| **Minimum stock** | this is the amount where if the quantity goes under this number it will be issued as running out |
| **Bought** | This is the shop where the item was bought from. Select from the list or input your own. It can also be left blank |
| **Location** | This is where the item is stored in the house. Select from the list or input your own. It can also be left blank |
| **Price** | This is the price of the item. Use the up and down buttons to alter the number or input a number. Putting the number as 0 will indicate that do not know/care about the price and when in database will be blank. |
| **Add to favourites list** | This allows you to add the product to favourites as an item which used regularly. |

1. Press ok to add the product to the database. It should then be visible in the products tab

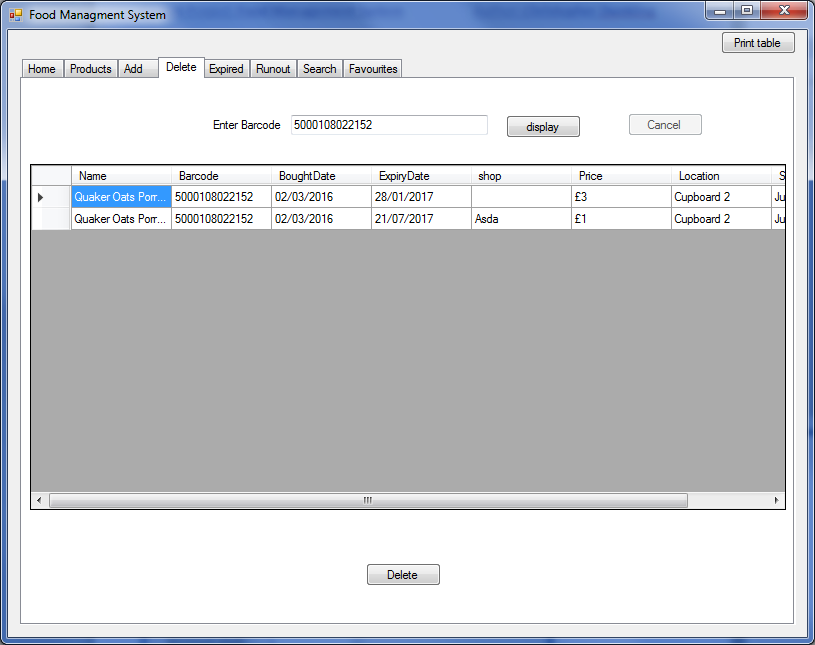
Note that the process can be cancel at any time by pressing cancel or closing the window.

**Adding items (which do not have a barcode)**

1. Navigate to input tab
2. Tick the “has no barcode” box
3. Instead of inputting a barcode, input the name
4. Continue as normal

User Manual Page 9

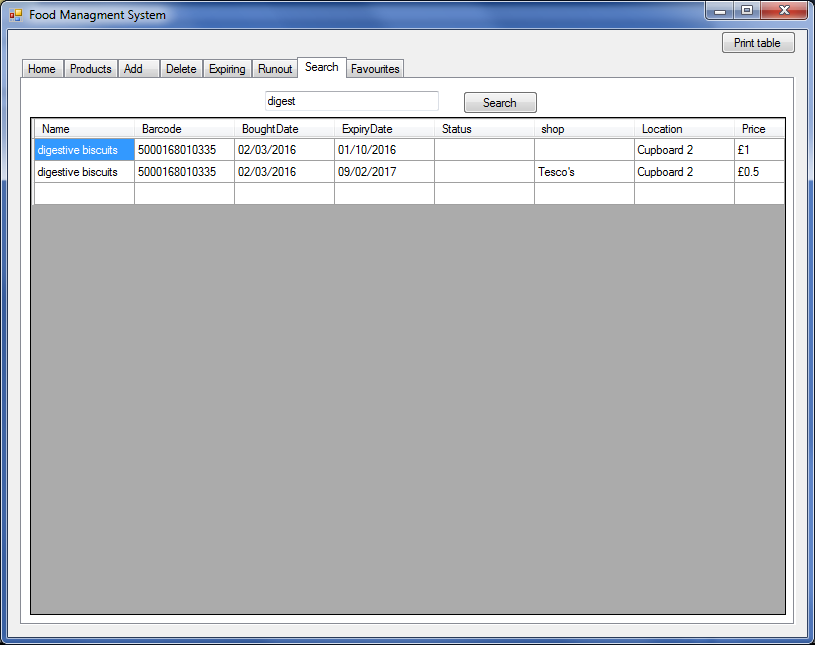
**Deleting items**



1. Navigate to delete tab
2. Input/scan barcode
3. Press display
4. This will list results relating to barcode and enable the delete button.
5. Select a row and press delete button
6. The quantity for that item should have decreased and the instance you deleted gone

User Manual Page 10

**Searching for items**



1. Navigate to search tab
2. Enter the phrase or word relating to the product you are trying to find
3. Press search

**Details on product screen**

1. Navigate to product tab
2. Select row
3. Press space
4. This will show you all the stock items for that product and gives you the ability to delete them as well

User Manual Page 11

# Appraisal

## Comparison of Project Performance Against Objectives

|  |  |
| --- | --- |
| 1. The application must run on a desktop PC running Microsoft Windows (XP upwards) | This was automatically achieved as any application made using visual studios 2012/13 is compatible with any Windows PC after XP |
| 1. The application where possible will use Microsoft Windows preferred interface controls to keep this application consistent with other applications and so easier and indurative for users use. | All of the forms were done with standard windows controls and is tab based to keep up to modern conventions |
| 1. User input to add, remove and select food items will be form-based | Add and delete tabs handle the two function and different events like running out and expiring |
| 1. A barcode reader will be connected to the PC and the application must be able to accept the barcodes read from food wrappers to:    1. Add new food items    2. Remove used food items | As the project is only a proof of concept the barcode is used to enter barcodes but could easily have been done via keyboard but made it less convenient |
| 1. Where food items do not have a barcode an alternative entry to identify a food item must be provided. | An item without a barcode can be entered by ticking the no barcode box and this substitutes the product name for the barcode |
| 1. The application will be able to query external web databases with the barcode to obtain food item descriptions | The application is linked to Outpan which is open source database. It has a lot of missing products but if I was to ever make this into a commercial product I would pay a subscription to one the big commercial database |
| 1. The application will always search for food items in local database before querying external web databases | The application looks to see if the product is already in the product database before consulting Outpan |
| 1. The user must be able to simply and quickly enter the details of all food items purchased into the system.   These details will include:  The food item description (name and make and size)   * 1. The food item description (name and make and size)   2. The date purchased   3. The expiry date   4. The purchased price   5. The shop purchased from   6. The minimum quantity to maintain   7. The location stored   8. Is on Favourites list | All fields have been included but some the questionnaire results suggest that other categories could be included but do make too big of a difference. |
| 1. The application will attempt to auto-fill all entry fields with values from:    1. External source like the external web database    2. Historical data from previous purchase of the food Item | As per objective 7. The application looks at the data to see what can be filled in for the user from historical data or Outpan |
| 1. Entry fields for price, shop purchased, and location stored can be left blank if not required by the user | All three can be blank as it is considered that these fields are not critical to be filled in. |
| 1. All entry fields can be edited and accepted changes will be reflected across all affected data and stored back into backup files | Changes to pre-existing data will affect across the board and be remembered for the next entry |
| 1. The user will be able to remove used food items by scanning the barcode or selecting the food item form any of the displayed views of food items | The user can remove via the delete tab or product tab after pressing tab to view the details window. However you cannot delete on every tab |
| 1. The user will be able view details of :    1. All food items currently stored    2. Food items that is close to or past expiring    3. Food items that is close minimum quantity or ran out    4. Favourites Food Items    5. Search Food Items | These all have their own separate tab to make easy for the user to check |
| 1. The user will be able to print any of the displayed views of food items. | A print button is in the top right hand corner at all times which allows the user to print any of the tables excluding delete table as it thought that not many people will want to print this. |
| 1. The application will load all stock and production data from file on start-up and store all stock and production data to file on close-down | All data is stored in CSV files. |
| 1. The application will monitor expiry dates and minimum quantities to keep displayed views of Food items that is close to or past expiring and Food items that is close minimum quantity or ran out up to date. | The data for running out and expiring are in their own tabs and also in the home tab in summary of ten items per table |
| 1. The application should be able to process at least 5000 food items in the Product Database and 1000 food items Stock Database | This is unknown as I have never entered this many items |

## Potential for Future Developments

## User Feedback

|  |  |
| --- | --- |
| **Adding Products** | The system is really easy to use. The only slight improvement is for the calendar to close automatically once the date is chosen instead of having to close the calendar window. |
| **Deleting Products** | User friendly to use |
| **Expiring Products** | Easy to look this up before shopping trip to know what item is expiring and whether to replenish it. |
| **Runout tab** | It is an efficient way to quickly check if an item is out of stock. |
| **Search function** | Excellent function, able to find items without spelling the full name. It makes looking up the stock level of items much easier. |
| **Favourites tab** | It allows you to add staple items into this list because it will be items most frequently need to be replenished. |
| **Overall** | It is a system that is easy to use and provides useful information for shopping |