|  |
| --- |
| Software Documentation |
| Software Design and Development |
| Part 1 |

|  |
| --- |
| Robert Harries  25/2/2013 |

Contents

[Introduction 1](#_Toc349515616)

[Functionality Requirements 1](#_Toc349515617)

[Compatibility and Performance 1](#_Toc349515618)

[Research of Similar products 2](#_Toc349515619)

[inFlow Inventory 2](#_Toc349515620)

[iMagic Inventory 3](#_Toc349515621)

[Development Considerations 4](#_Toc349515622)

[Social and Ethical issues 5](#_Toc349515623)

[Acknowledgement of assistance and sources 5](#_Toc349515624)

[Privacy and security 5](#_Toc349515625)

[User Friendliness and inclusivity 5](#_Toc349515626)

[Gantt Chart 5](#_Toc349515627)

[Interface Design 6](#_Toc349515628)

[Sales window 6](#_Toc349515629)

[Add Item window 7](#_Toc349515630)

[Edit Item and Item Selection window 8](#_Toc349515631)

[System Modelling 8](#_Toc349515632)

[Data Dictionary 8](#_Toc349515633)

[Input Process Output charts 9](#_Toc349515634)

[Data Flow diagram 9](#_Toc349515635)

[Context diagram 10](#_Toc349515636)

[Structure chart 10](#_Toc349515637)

[Algorithm and Test Data 10](#_Toc349515638)

[Load Inventory File 10](#_Toc349515639)

[Add Item 10](#_Toc349515640)

[Edit Item 11](#_Toc349515641)

[Sales 11](#_Toc349515642)

[Desk Check 12](#_Toc349515643)

[Log Book 12](#_Toc349515644)

[Bibliography: 13](#_Toc349515645)

Software Design and Development Major Project

# Introduction

My Cataloguer is a program designed to control and handle all inventory systems for seller/business. Anyone who is selling items and/or keeping an inventory of items that needs to be managed can benefit from this program.

In the programs control of the inventory, it will keep track of values such as Prices of item, stock amounts in the inventory of the seller/business, capacity of the shelf space of the store, and most importantly, the minimum shelf levels of the store before a re-order is needed.

My Cataloguer will include features such as input for items and the ability to save values for each item, such as its price, its shelf capacity in the store/business, its current level of stock, and its minimum level before a new delivery is needed to prevent the item from running out of stock. It will also include a sales point, which will feature a checkout system, totalling the items bought by a single customer, and also tallying down the stock levels of the items bought. Inside the sales feature, will include an online sales/ordering system, along with electronic invoicing.

# Functionality Requirements

Functionality requirements describe what the software aims to do. These requirements allow the software development to have direction of where the planning is to lead, by adding measurable goals for the developers to aim for.

1. The program should be able to handle at least 150 store items at once

This will be tested by creating 150 items in the program and performing all the usual functions to see if the program will still run stably

1. The program should be able to save the inventory dictionary to a text document in under 10 seconds

This will be tested by creating a number of items, saving it to file and recording the time

1. The program should be able to read the inventory list from a file into the program in under 10 seconds

This will be tested by loading a saved inventory file into the program and recording the time taken

# Compatibility and Performance

Hardware requirements for this software will require and computer with an operating system capable of running Python. So basically, any desktop PC/Laptop running any operating system, Windows 2000/NT, XP, Vista, 7 or 8, Apple iOSX and Linux variants (Ubuntu, linux mint, debian, etc). Python and wxpython will be required to have been installed to execute the software.

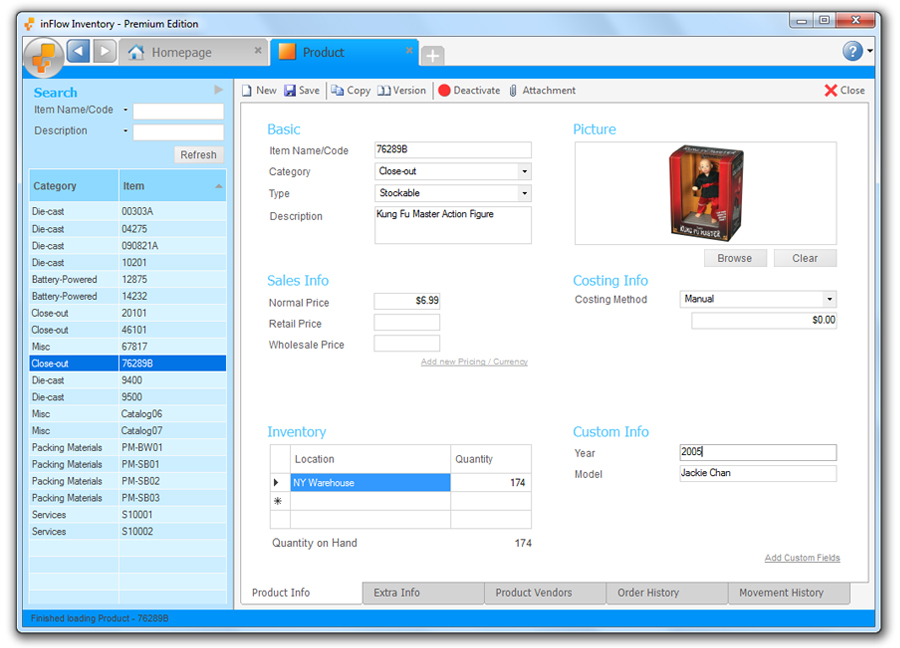
Performance issues that may arise in the software may include stability/speed reductions when using a large inventory lists. This will of course require testing, and possible refinement of the software to try and counteract these issues if they do present themselves. Possible changes to the code to try and fix these issues may include using multiple item dictionaries instead of just one.

# Research of Similar products

## inFlow Inventory

inFlow Inventory is an inventory system, which is either free (Max 100 products, with limited features), or can be bought for larger businesses ranging from $299 to $499. The main features of the software include a product list that can handle +10,000 products, barcode scanner input, locational information on each product, full movement of each products history, easy adjustment of item’s values such as prices and quantities, printing capability for invoices, an automatic ordering system and graphed statistics for items history. inFlow’s Premium package includes many features such as an unlimited product and customer database, 29 different reports, a multi-user mode, separate user accounts, user restriction options, and phone, email and forum support for the business.

Here is have a screenshot of inFlow’s item details page, one of the more used of the program



d.

a.

c.

e.

b.

As shown in this picture of inFlow’s layout of the main window, it has many more information values for each product than My Cataloguer. Though they have their similarities, such as price (a), stock on hand (b), there are many features included in inFlow that are not planned to be in My Cataloguer, such as warehouse information (c), description of the item (d), and pictures of the said item (e). Also, both pieces of software will have the capability of supporting sales, invoicing, auto-correcting values as items are sold and automatic re-ordering of stock of items sold.

## iMagic Inventory

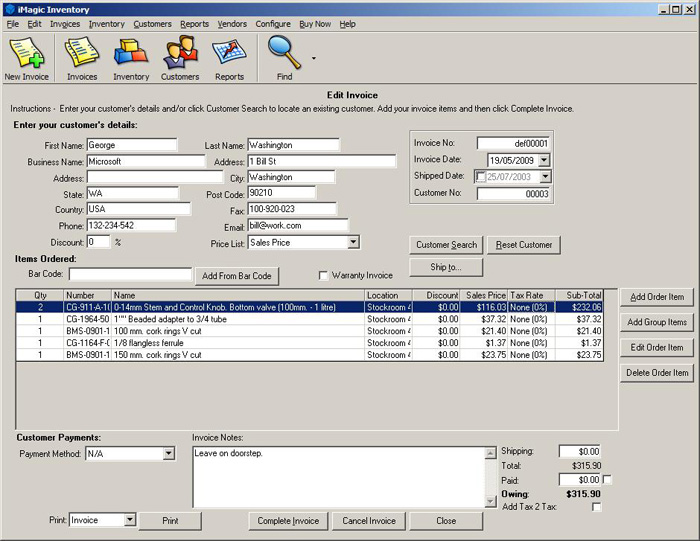
iMagic Inventory is also an inventory program, which ranges in price from $249 for single users to $749 for a 6 user licence. The main features of this program include a stock database, barcode support, multiple price lists for items, multiple warehouse compatibility, item tracking and automatic re-ordering for stock management, a customer database, billing address and information and payment tracking for orders for customer management, and lastly a database backup and importing and exporting features, with compatibility with Microsoft Office.

Here is the invoicing page for iMagic

c.

f.

b.



a.

d.

e.

g.

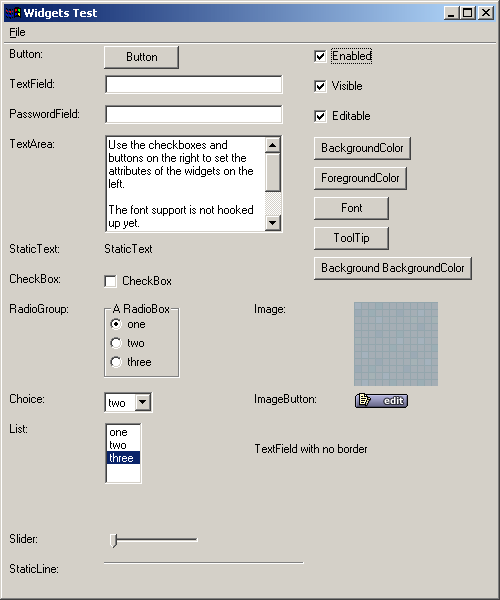
As shown in this picture, iMagic has an extensive invoicing system. Like My Cataloguer, basic features such as names, addresses, state, country, postcode, email and phone numbers are all included (a), along with subtotalling and totalling (b). Although, iMagic has extra features such as invoice numbering for archiving(c), a re-arrange able item listing (d), payment method selection (e), extra notes for inclusion on the invoice (f) and printability (g) are all not planned to included with My Cataloguer.

# Development Considerations

Python 2.7.3 is the chosen coding language, along with the GUI widget library of wxPython.

Python 2.7 has been chosen as the coding language because of its ease of use and its dictionary feature, which allows for lists, which include keys with values assigned to them, to be placed inside another key, which is part of a bigger list, making the inventory list easier to create, manage, call, and search than it may have been using other languages.

WxPython was chosen for its simple, neat, yet extensive list of widgets to use. These widgets include the basic windows and panels used to put other widgets in, its easy integration of pictures (a) and image buttons (b), static text (titles, labels) (c), label-able buttons (d), checkboxes (e), radio buttons (f), sliders (e.g. volume slider) (g), gauges (e.g. loading bar), spinners, static lines (h) for aesthetic purposes, and also customisable menu (i) and status bars.



i.

c.

f.

g.

h.

e.

d.

a.

b.

# Social and Ethical issues

### Acknowledgement of assistance and sources

Acknowledgement of assistance and sources will be included into the program in the form of a common way, usually labelled in a help menu called “about”. By clicking this button, a window is brought up listing resources and help used in the making of the software. Among the acknowledgements will be wxPython for the use of the libraries, Python for the IDLE and the code the software is built on.

### Privacy and security

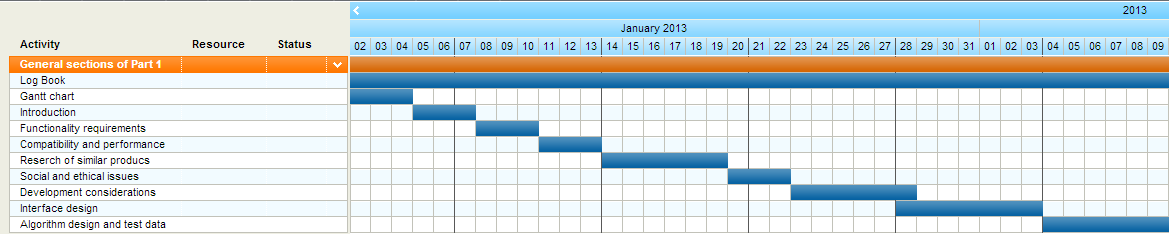
Privacy and security isn’t a major feature in My Cataloguer. The program is made for wide uses across a store, and therefore by possibly many staff. This makes pass-wording/security would become more of a hindrance than a help. Although the inventory saving system will allow for saving of the inventory into place, then remove it from the system, hence helping to secure the data, and because all save files are compatible with all versions of the software, it is quite easy to re-enter the file.

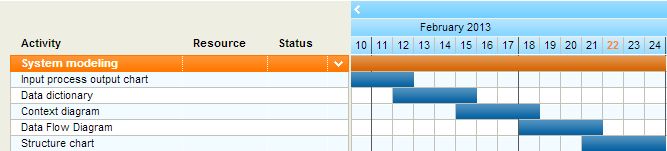
### User Friendliness and inclusivity

In terms of user friendliness, the GUI in My Cataloguer is kept as simple as possible. All menu’s and submenu’s are in the most logical and common places that they are found. Instructions on how each process is done are clearly and simply explained when prompted.

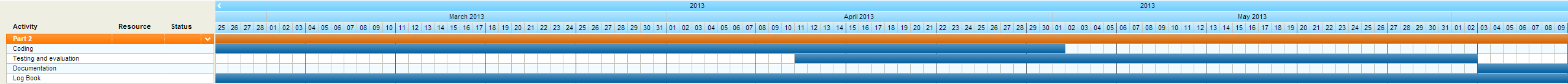
With this simplicity, comes the ability for the software to widely inclusive. People with low level computer skills will be able to operate this software with the instructions. The software will also be freeware. The software will also be usable in other countries because of the currency used not being accounted for, it’s just a number. Although, languages other than English are not to be included, this does not help to include non-English speaking countries to be able to use this software.

# Gantt Chart

Part 1:

Higher resolution download of image <http://imgur.com/download/xQnIDW1>

Higher resolution download of image <http://imgur.com/download/JpfsV7r>

Part 2:

Higher resolution download of image <http://imgur.com/download/X1GM8Pp>

# Interface Design

## Sales window

This window, the sales window, will be the main window for the program. Starting from the top, the menu bar is shown on this window. From here, File > Load will allow for the loading of saved inventory files, and File > Quit will allow for closing the program along with the “x” in the top corner. In edit are the selections of the “add item” and “edit item” options.

The layout of this window is simple. There is an item list to the left, where an item will be chosen by clicking it, then pressing the select button. This will then bring up the price and stock level in the “Item details” section, as clearly labelled. The product will be displayed in the “item” field in the “sale details” section, where the user can then enter a quantity (Default is 1). An alternate price can then be entered, in the price field, but if left blank, will use the default price. The Current total is then updated to include those items upon clicking the “enter” button. Then right down the bottom, there is a status bar, which when mousing over menus and sub-menu’s, will provide a brief explanation of its purpose.

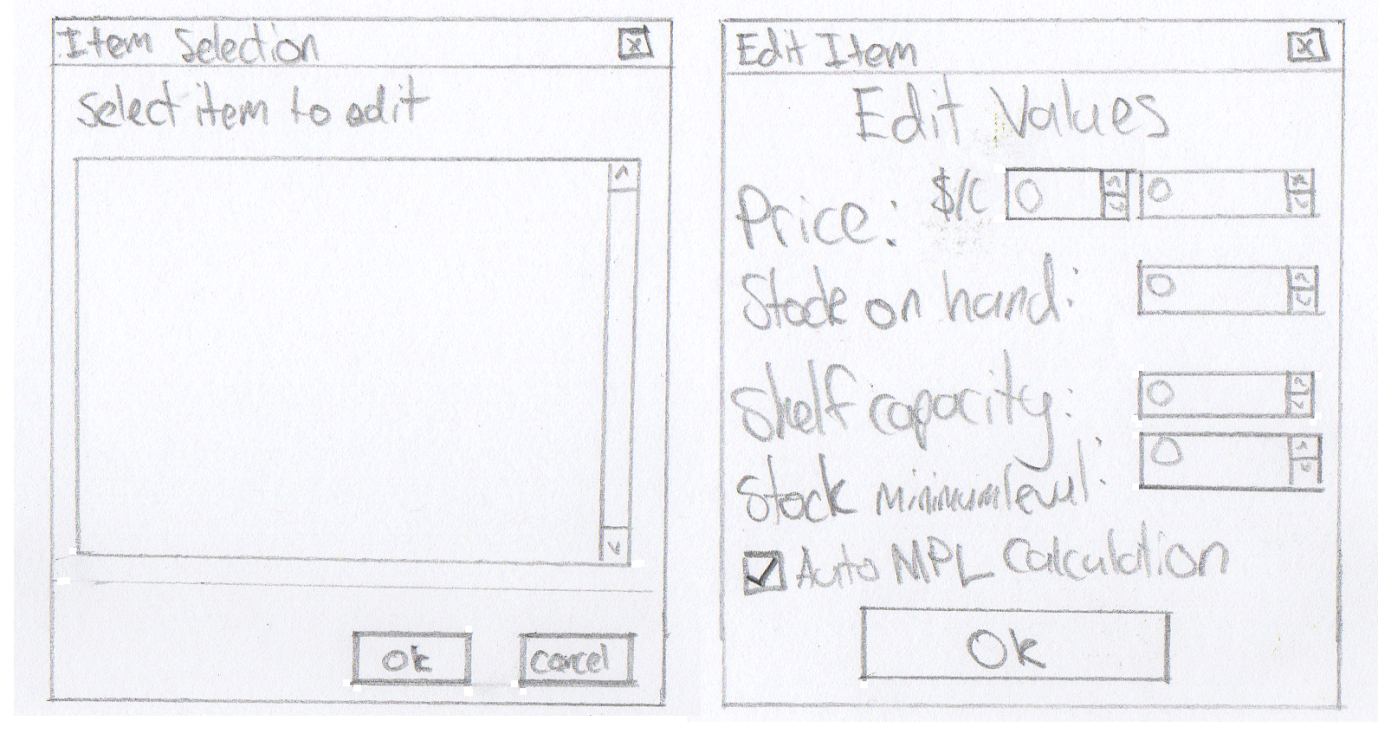
The arrangement of this window is very ordered. There are the menu options in one place, then the item selection, details, and sale details in order, making the sale a simple process rather than scouring a page full of information to find the right field. The status bar also helps improve useability by explaining the options.

## Add Item window

The Add Item window is the prompt that is brought up after clicking the “Add Item” option in the “Edit” menu on the Sales window. This is a basic text entry box, whereby the user enters a name of a new item to add to the item dictionary, then presses ok.

The instructions for this window are clearly labelled, and the window is clear of any unnecessary information.

## Edit Item and Item Selection window

The Item selection window is a simple one, with an instruction, a field for selection, and an ok and cancel button. The Item list will be loaded into the selection field, so all the user has to do is select one and press ok.

The Edit Item window has labels for each value assigned to an item, each with their corresponding adjuster, along with a checkbox near the bottom to select automatic MPL calculation, and below that an ok button.

Both of these windows are again, set out in a neat, ordered fashion. The Edit window is laid out in a way that the user can just go through each field one at a time, and not miss anything. The item selection is simple, and clearly instructed.

# System Modelling

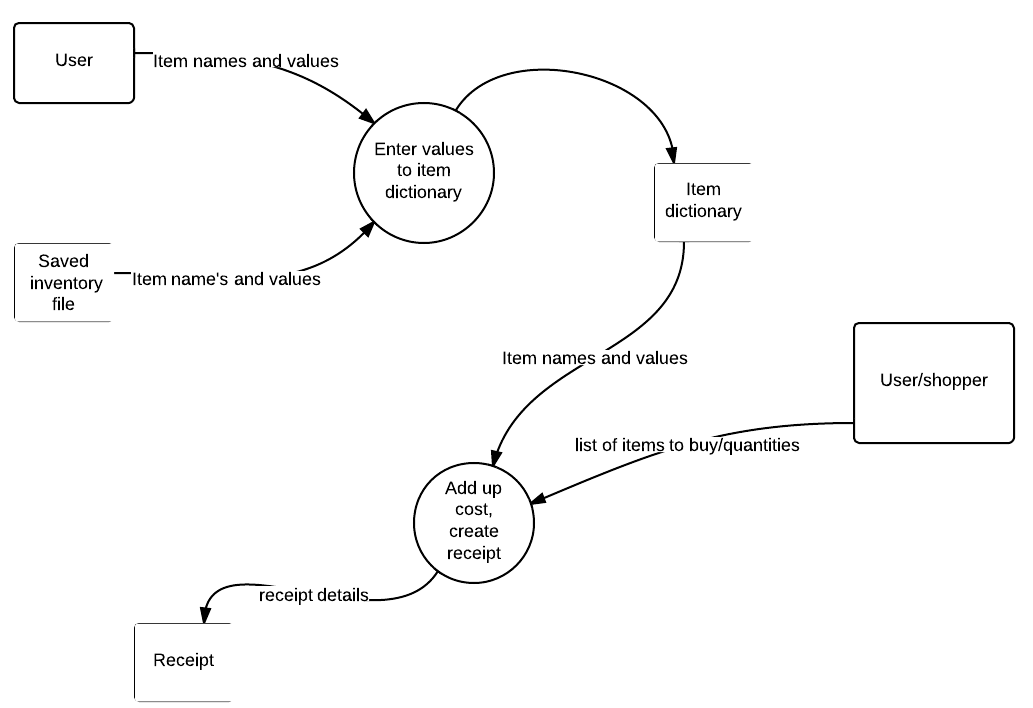
## Data Dictionary

|  |  |  |
| --- | --- | --- |
| Field name | Data type | Description |
| itemDict | Dictionary{string: {string: integer} | Holds item names and their values |
| nameList | Array[string] | Is a list of the names of all the items in alphabetical order |
| selectedItem | String | Hold the name of the selected item for editing, enabling it to be used in another class (Global) |
| totalPrice | Integer | Total cost of a sale |
| Quantity | Integer | Amount of the current chosen item |

## Input Process Output charts

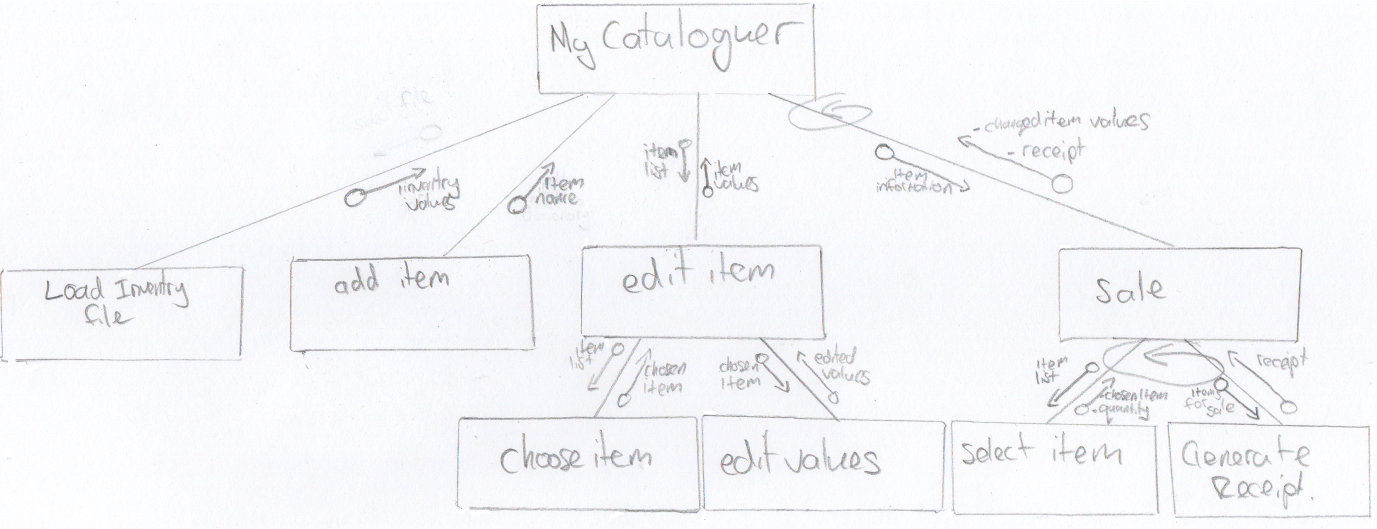
|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Process | Output |
| Inventory list | | -Select file to load into program -Read contents of file into item dictionary | - |
| Item name | | -Select the add item option  -Enter item name  -Add item name to item dictionary along with respective values | - |
| Values of the item (SOH, Price, Capacity, MPL) | | -Select item  -Get edited values  -Append dictionary values for chosen item | **-** |
|  | | -Sort the item names into an array | **Name list** |
| Sale details | | -Select item -Enter quantity  -Add to receipt  -Add price to subtotal | **Receipt, Total price, Quantities** |
|  | -Write item dictionary to file | **Inventory list** | |

## Data Flow diagram



## Context diagram

## Structure chart



Higher resolution: <http://imgur.com/download/6mbCl2Q>

# Algorithm and Test Data

## Load Inventory File

BEGIN Load Inventory File  
 Get File location  
 Get File Contents  
 Set itemDict = File Contents  
END Load Inventory File

## Add Item

BEGIN Add Item  
 Get EnteredName  
 Set itemDict{Entered name: {Capacity: 0, Price: 0, SOH: 0, MPL: 0}}  
END Add Item

## Edit Item

BEGIN Edit Item  
 Get SelectedItem  
 Get EnteredPrice  
 Get EnteredSOH  
 Get EnteredCapacity  
 IF AutoMPL = TRUE  
 Set MPLValue = (EnteredCapacity/5)+1  
 ELSE  
 Set MPLValue = Entered MPL  
 Set itemDict{SelectedItem: { Capacity: EnteredCapacity, Price: EnteredPrice, SOH: EnteredSOH, MPL: MPLValue}}  
 ENDIF  
END Edit Item

## Sales

BEGIN Sales  
 WHILE FinishShopping = false:  
 Get NameList  
 Display Namelist  
 IF SelectedItemButton = true  
 Get SelectedItem  
 Get SelectedItem[Price]  
 Get SelectedItem[SOH]  
 Display SelectedItem  
 Display SelectedItem[Price]  
 Display SelectedItem[SOH]  
 ENDIF  
 IF EnterButton = true  
 Get EnteredQuantity  
 IF EnteredPrice =/= 0  
 Set FinPrice = EnteredPrice  
 ELSE  
 Set FinPrice = SelectedItem[Price]  
 ENDIF  
 Set CurrentTotal = CurrentTotal + (FinPrice\*EnteredQuantity)  
 ENDIF  
 ENDWHILE  
END Sales

## Desk Check

For Edit Item:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Inputs: | Price | Capacity | SOH | MPL |
| Test 1 | 5.99 | 1 | 0 | Auto |
| Test 2 | 6.00 | 2 | 1 | Auto |
| Test 3 | 87.75 | 65 | 34 | Auto |
| Test 4 | 34.98 | 345 | 76 | 70 |
| Test 5 | 23.90 | 21 | 456 | 10 |
| Test 6 | 0.99 | 14 | 123 | 5 |

|  |  |
| --- | --- |
| Test Output: | Output (itemDict) |
| Test 1 | itemDict{Test1: { Capacity: 1, Price: 5.99, SOH: 0, MPL: 1}} |
| Test 2 | itemDict{Test2: { Capacity: 2, Price: 6, SOH: 1, MPL: 1}} |
| Test 3 | itemDict{Test3: { Capacity: 65, Price: 87.75, SOH: 34, MPL: 14}} |
| Test 4 | itemDict{Test4: { Capacity: 345, Price: 34.98, SOH: 76, MPL: 70}} |
| Test 5 | itemDict{Test5: { Capacity: 21, Price: 23.9, SOH: 456, MPL: 10}} |
| Test 6 | itemDict{Test6: { Capacity: 14, Price: 0.99, SOH: 123, MPL: 5}} |

# Log Book

#### Jan 2nd:

Log book started

#### Jan 4th:

Gantt Chart completed

#### Jan 7th:

Introduction complete

#### Jan 10th:

Functionality requirements complete

#### Jan 13th:

Compatibility and performance issues complete

#### Jan 19th:

Research of similar product completed

#### Jan 22nd:

Social and Ethical issues completed

#### Jan 27th:

Development considerations completed

#### Feb 03th:

Interface design completed

#### Feb 09th:

Algorithm design and testing completed

#### Feb 12th:

Input process Output chart complete

#### Feb 15th:

Data dictionary complete

#### Feb 18th:

Context diagram complete

#### Feb 21th:

Data Flow Diagram complete

#### Feb 22nd:

Help gained from Helen Dizura for Input process Output charts and structure charts

#### Feb 23rd:

Structure chat complete,

#### Feb 24th:

Part 1 complete

# Bibliography:

<http://www.inflowinventory.com/>

<http://www.imagicinventorysoftware.com/>

<http://www.tomsplanner.com/>

<https://www.lucidchart.com/>

<http://www.wxpython.org/>

Software design and development Davis textbook