# Sense Boutique

# Inventory Management System

***Technical Documentation***

Name: Loh Hao Bin

Class: 1201PB1

Student ID: 1201A18902

# Table of Contents

[Table of Contents 2](#_Toc352261280)

[B) Design 5](#_Toc352261281)

[Aim 5](#_Toc352261282)

[System Objectives 6](#_Toc352261283)

[Requirement Specification 8](#_Toc352261284)

[Input Requirement 8](#_Toc352261285)

[Process Requirement 8](#_Toc352261286)

[Output Requirement 9](#_Toc352261287)

[Hardware and Software Requirements 10](#_Toc352261288)

[Hardware Requirements 10](#_Toc352261289)

[Software Requirements 12](#_Toc352261290)

[Context Diagram 13](#_Toc352261291)

[System Flowchart 18](#_Toc352261292)

[Gantt chart 19](#_Toc352261293)

[Jackson Structured Programming 20](#_Toc352261294)

[Entity-Relationship Diagram (ERD) 21](#_Toc352261295)

[Files, Records, and Data Structures 22](#_Toc352261296)

[Design of Forms 26](#_Toc352261297)

[Splash Screen 26](#_Toc352261298)

[Login Screen 27](#_Toc352261299)

[Main Screen and Menus 28](#_Toc352261300)

[Inventory Window 30](#_Toc352261301)

[Add New Inventory Items 32](#_Toc352261302)

[Edit Inventory Items 34](#_Toc352261303)

[Sale of Item 36](#_Toc352261304)

[Preview Window 37](#_Toc352261305)

[Suppliers Management 38](#_Toc352261306)

[Item Categories 40](#_Toc352261307)

[Transaction Report 41](#_Toc352261308)

[Users Management 42](#_Toc352261309)

[Creating New User 43](#_Toc352261310)

[Changing Passwords 44](#_Toc352261311)

[Backup 45](#_Toc352261312)

[Restore 46](#_Toc352261313)

[Quick Start Guide 47](#_Toc352261314)

[Help 48](#_Toc352261315)

[About 48](#_Toc352261316)

[Intended Benefits of the proposed system 49](#_Toc352261317)

[Test Plan 50](#_Toc352261318)

[Tests to be Carried Out 52](#_Toc352261319)

[Limitations 57](#_Toc352261320)

[File Size Calculations 58](#_Toc352261321)

[Program Listing (Coding) 60](#_Toc352261322)

[Base Module (Module1.bas) 60](#_Toc352261323)

[Splash Screen (FormSplash.frm) 61](#_Toc352261324)

[Login Screen (Form1.frm) 65](#_Toc352261325)

[Main Screen (MDIForm1.frm) 70](#_Toc352261326)

[Inventory List window (Inventory.frm) 75](#_Toc352261327)

[Add New Items (FormAddNew.frm) 88](#_Toc352261328)

[Edit Item… (FormEdit.frm) 101](#_Toc352261329)

[Sale of Item (FormSales.frm) 113](#_Toc352261330)

[Export/Print Preview (Preview.frm) 121](#_Toc352261331)

[Suppliers Management (FormSuppliers.frm) 133](#_Toc352261332)

[Transaction Journal/Report (FormJournal.frm) 143](#_Toc352261333)

[Categories Management (FormCategories.frm) 146](#_Toc352261334)

[Backup/Restore (FormBackup.frm) 150](#_Toc352261335)

[Users Management (FormUsers.frm) 158](#_Toc352261336)

[Add New Users (FormNewUser.frm) 163](#_Toc352261337)

[Changing Users Passwords (UserPassword.frm) 169](#_Toc352261338)

[Quick Start Guide (FormQuickStart.frm) 177](#_Toc352261339)

[User Help (FormHelp.frm) 178](#_Toc352261340)

[About (FormAbout.frm) 179](#_Toc352261341)

# B) Design

## Aim

To develop a user friendly inventory management system that allows for a more efficient process of keep tracking of inventory stock records, reduce input errors, employs more reliable data redundancy such as a backup system as well as better security.

The system will first require user credentials in order to grant access to the database, then the system should be able to provide an overview of the current stock on hand by listing them out in a neat list view which is easy to understand, provide methods to search through the list quickly, and provide methods to allow user to input various stock details easily.

Then as the business operates, the user should also be able to update stock levels, intakes and sales with ease, and the system should be able to keep track of these transactions to provide a report for managerial purposes and business decision. On the other hand, the system should also be able to keep records of various dealers to allow the user to access them whenever need arises. The user should also be able to print out reports and list of stocks on hand.

For the administrators, the system should also be easier to maintain, with automated database maintenance, reliable backup and restore methods, central user management as well as proper documentation for the users.

## System Objectives

**- To make the process of managing a large stock inventory more efficient**

Previously, staffs have to manually keep track a large amount of documents and records, at the same time, the process was tedious and error prone. One of the main objectives of creating the new system is to eliminate the hassle of manual labor work such as inventory records, sales records analysis and dealer’s list, etc. By employing methods that allow the user to one click add sales and purchases, this allows for higher business efficiency.

**- To increase productivity**

The system should also be much easier to use and reduces boredom for the staff, because the employees does not have to juggle with a large stack of paperwork or documents, hence translating to higher productivity. The new computerized system should be better and faster at handling a large amount of customers especially during peak periods, errors should also be reduced and the staff can be more productive. On the other hand, being able to search and input data easily also allows the staff to handle large and tedious amounts of work with far less effort, as the employees does not have to search through a lot of irrelevant information in order to look for the particular item they are looking for. Reports are also generated automatically and don’t require extra work.

**- To reduce input errors**

The computerized system will employ methods to check for simple input errors such as helping the user to prevent putting a non-numerical value in the price detail field. This allows for better validity of data, and makes the stock records less error prone. Time can also be saved because there would be a lower probability to waste time to look for accidental inputs.

**- To save time**

As stated in the previous points, the computerized system is more efficient at handling a large amount of data, while at the same time streamlining the business operation, making it much more efficient. The saved time can be translated to higher productivity and work can be finished faster as a result.

**- To have more reliable data redundancy**

This computerized system will be designed with a backup and restore system in mind, that allows the user to easily backup by creating a copy of the database elsewhere in a remote secure safe location, to help mitigate complete data destruction as well as allowing for data restoration in case of system failure, database corruption, missing data, accidental deletion of records, accidents, crashes, fire, floods or natural disasters.

**- To have better security measures**

This new system will incorporate a user accounts system that requires each user to login with a unique user ID and password in order to gain access to the database. This prevents unauthorized access to business data. The passwords are also stored in a secured encrypted format to prevent someone else to hack them easily. New users can be added and passwords can be changed easily by the administrator. On the other hand, there will also be levels of user privileges implemented so that only administrators can access certain features, such as backup/restore, viewing system reports and managing users.

**- To reduce costs and minimize risks**

By making the business operations more efficient, it is possible to reduce the cost of operations of business in the long run. Higher productivity can be translated to higher sales, hence higher profits as well. On the other hand, employment of better security measures as well as better backup/restore system allows for sensitive business data to be secured and losses can be minimized in event of any accidents or disasters. For example, if there is a flood, the data stored in the old system based on paper documents will all be destroyed, while the data in the new system may be secured due to a secure backup stored in a safe location.

**- To aid in making better business decisions**

Through the new computerized system, it is able to automatically generate transaction reports and calculate total costs as well as gross profits. This allows for more efficient management of the accounts and balance sheet. Hence, the business owner is able to easily make business decisions such as whether to buy more of these stocks in anticipation of future demand or higher profits.

## Requirement Specification

### Input Requirement

1. Security measure through password logins

* During login, the user inputs a User ID and password in order to login.

1. Inventory records

* To add a new record, the user would have to input a product ID, product name, quantity, cost, selling price, description, color, size, category and date.

1. New purchases incoming

* To add incoming new purchases, the user would have to input the quantity bought.

1. Sale of items outgoing

* To add a new sales transaction, the user would have to input quantity sold and selling price.

1. Suppliers details

* To add a new supplier, the user would have to input the name, IC, telephone number, email and address.

1. Manage categories types

* To add a new category, the user inputs a new category name.

1. Manage users

* To add a new user, the admin inputs a new username and password.

### Process Requirement

1. Storing of each inventory item details

* After inputting the item details (e.g. Product ID, Name, Color, Quantity, Cost etc), the system will check for input errors and then save it to the database.

1. Calculate gross profit for each sale transaction
2. Calculate the new stock level after purchases
3. Storing of supplier details
4. Backup and Restore of database
5. Encryption and decryption of passwords

### Output Requirement

1. Print Preview before print or export

* Displays a print preview window before allowing the user to print or export inventory list or transaction report to file.

1. Export of Inventory Items to Text file

* A list of inventory items will be saved as plain text format.

1. Printing of Inventory List to paper

* A list of inventory items will be printed in hardcopy format.

1. Export of transaction report to text file
   * A transaction report will be saved as plain text format.
2. Printing of transaction report to paper

* A transaction report will be printed in hardcopy format.

## Hardware and Software Requirements

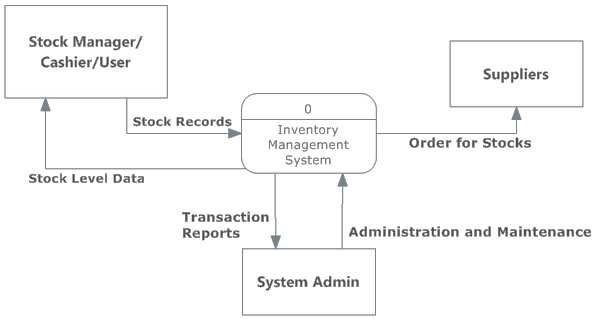
### Hardware Requirements

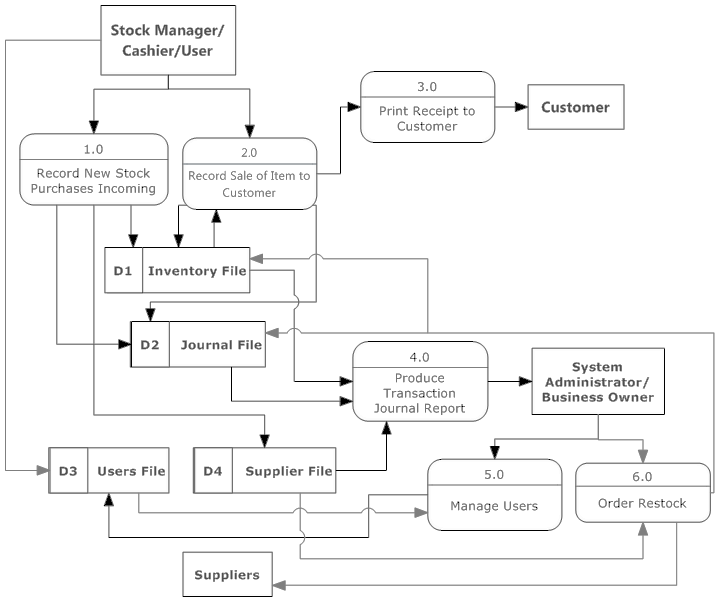
* **Monitor with minimum 1024x768 resolution (XGA)**
  + A monitor is needed to display the graphical user interface of the system. A XGA monitor (Extended Graphics Array) has a resolution of 1024x768. It is recommended that the monitor has a minimum resolution of 1024x768 to allow for viewing of more contents, as well as compatibility with certain websites.
* **QWERTY keyboard and mouse**
  + On a WIMP (Windows, Icons, Menus and Pointing devices) graphical user interface, it is recommended that the system employs a keyboard and mouse for easy navigation of GUI and allows for efficient data input.
* **A compatible CPU**
  + Intel Pentium/Celeron/Core series of processors, or AMD Athlon/Phenom, or other compatible processors
  + Minimum 1Ghz clock speed recommended for average
  + CPU is required to process the data and the faster the CPU, generally the speed of the system will be faster.
* **Minimum 512MB of RAM**
  + For running on Windows XP SP3, it is recommended to have a minimum of 512MB of RAM to ensure smooth operation.
  + For running on Windows Vista, Windows 7 or Windows 8, it is recommended to have a minimum of 2GB RAM to accommodate the more advanced graphics capabilities of these modern OS.
  + RAM are used to stored data temporarily while the system is running, a lower amount of RAM will cause the system to constantly swap data between the disk and the memory, leading to system slowdown.
* **Minimum 50GB hard disk drive**
  + For Windows XP, the OS and other medias takes up around 5GB-8GB of storage; meanwhile for Windows Vista, 7 and 8, the operating system files take up around 10-16GB depending on configuration.
  + The size of the program, the HTML documentations and the database itself takes around 20MB, depending on the size of the database and amounts of records stored in it.
* **CD/DVD Drive, or other external media such as USB flash drives**
  + These removable external storage media can be used to store backups, and then placed in another remote safe location for safekeeping. Then when there is a need for data to be restored, they can be retrieved from this storage easily to perform a restore operation.
* **Network Interface Card and Modem Routers**
  + The network interface card (NIC) allows the computer to connect to a network, while the modem router allows the computer to connect to the Internet.
  + An internet connection allows the user to connect to the internet for web browsing, download system software updates, backup onto the “cloud”, perform online businesses, emails and more.
* **Printer**
  + A printer is needed for the program to print out hardcopy versions of inventory lists or transaction reports.
  + Commonly used printer types are Inkjet printers and laser printers. Inkjet printers are cheaper but lower quality, while laser printers are fast and high quality, but they and their toners are more expensive.
  + Printers can also come with scanners or fax machines built in, which can be useful to the business as well.
* **UPS (Uninterruptible Power Supply)**
  + In a business environment, a temporary power loss can cause monetary damages to the business as well as causing loss in important data. Hence, an UPS can be used to provide backup power to continue running the system, providing enough time to save valuable data and safely shut down the computer in case of a power failure.

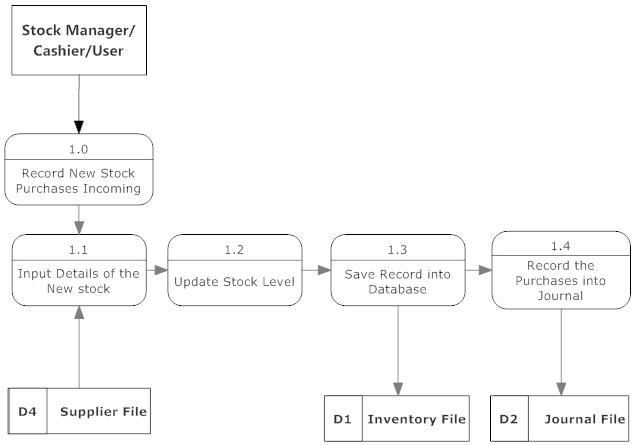
### Software Requirements

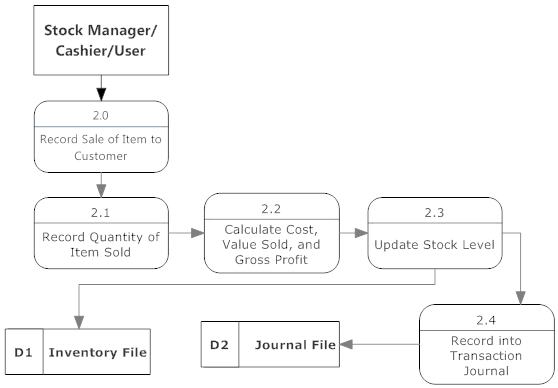
* **Microsoft Windows XP and above**
  + Windows is the most commonly used operating system in the market, and it is easy to obtain support as well as applications to run on this platform.
  + It is recommended to run newer versions of Windows, such as Windows XP, 7 and 8 to ensure a better experience, more secure environment, more stable and less bugs.
* **Internet Explorer 6.0 and above required** 
  + Internet Explorer 6 and above is required for the program to display HTML help documentation
  + However, it is not needed to be used for normal web browsing activities. Users are recommended to download alternative browsers such as Google Chrome, Safari, or Mozilla Firefox for such activities.
* **Antivirus software** 
  + It is recommended to install antivirus or antimalware programs if the system is going to access the internet, to protect the system from malware threats that can damage the system or cause data loss, or even stealing important personal information.
* **Microsoft Office suite**
  + Microsoft Office suite includes programs such as Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Microsoft Outlook and Microsoft Access, which are good for business activities such as writing formal documents, managing calendars and contacts, email, making presentation etc.
  + The program will be using the Microsoft Access OLEDB database framework.
* **Microsoft Visual Basic 6.0**
  + This will be the development tool that the system analyst is going to use to develop the new system. However, it is not required to run the developed system because the program will be compiled to native object code in the end. However, if there are needs to change the system to fix bugs or add new features, this will be needed.

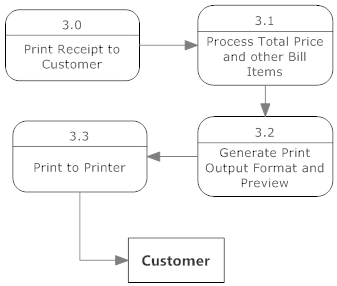
## Context Diagram

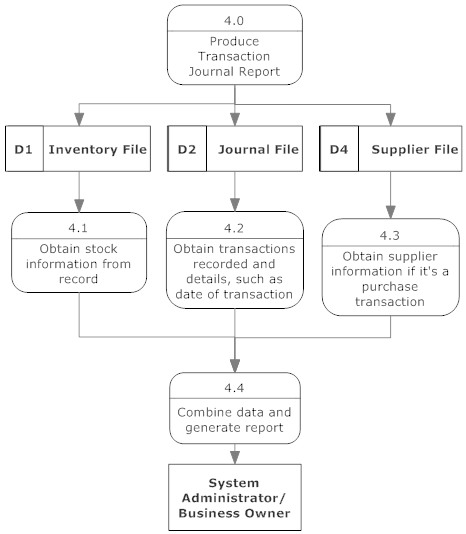


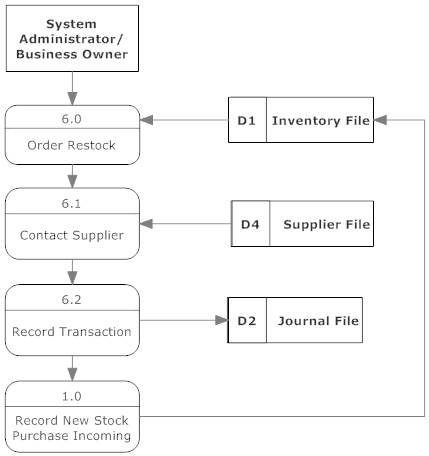
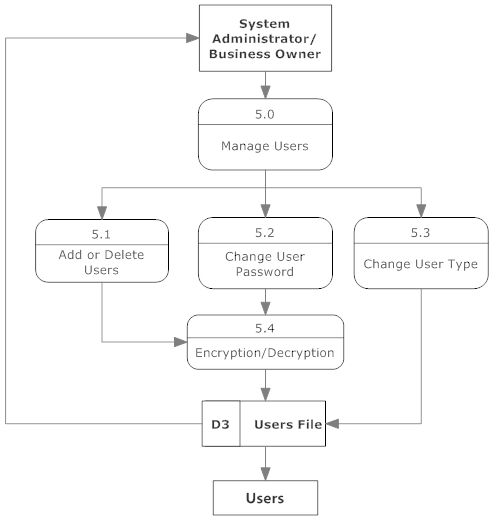




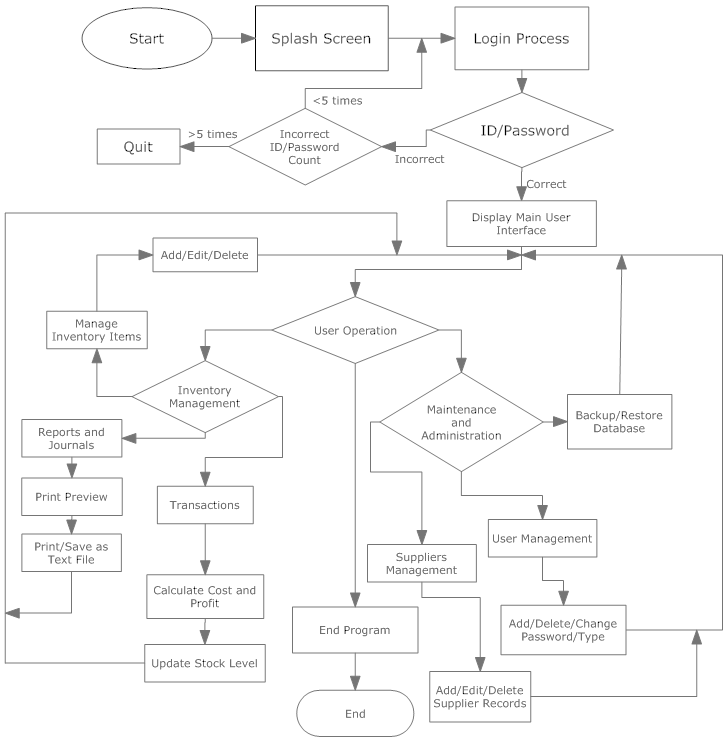




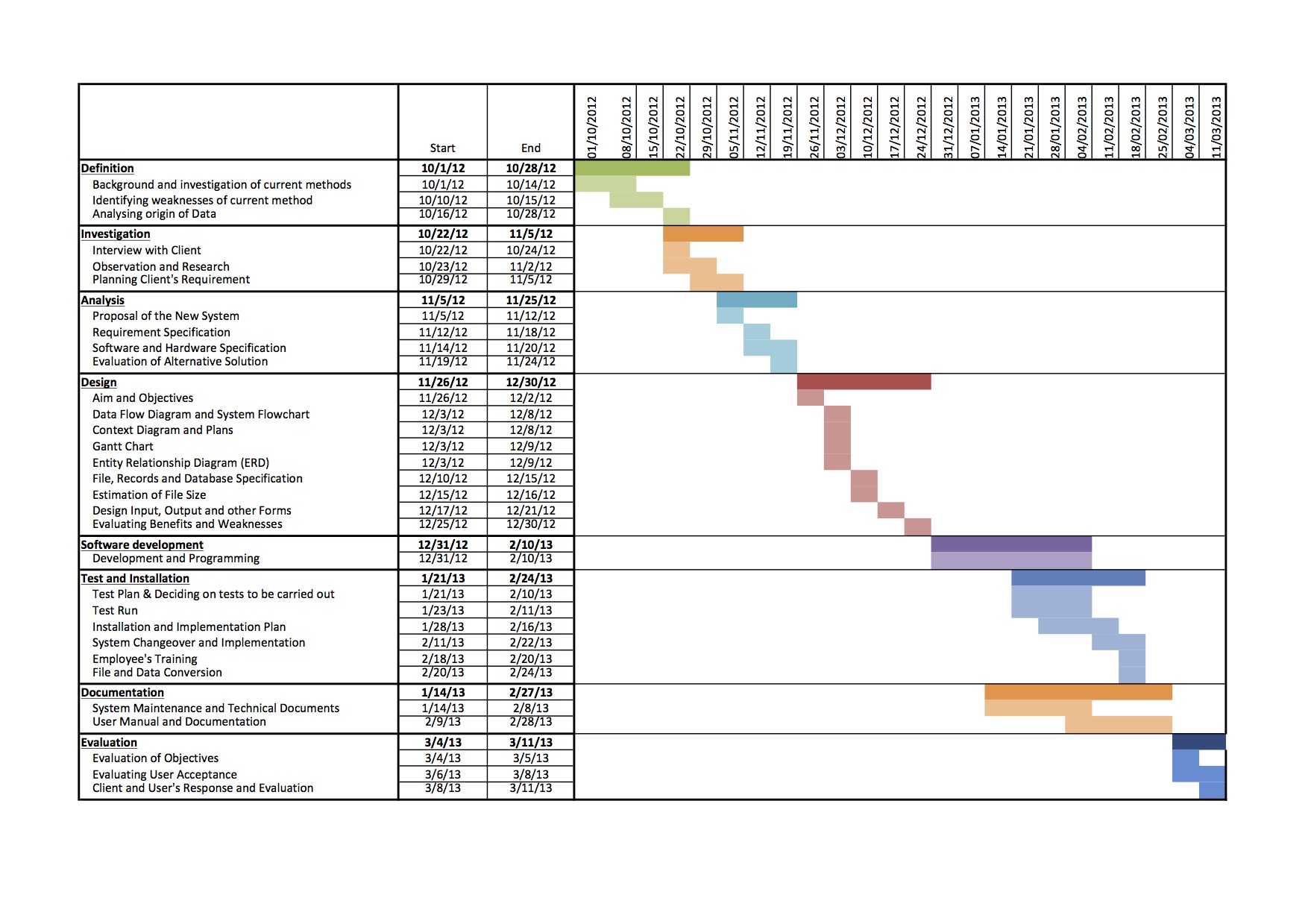




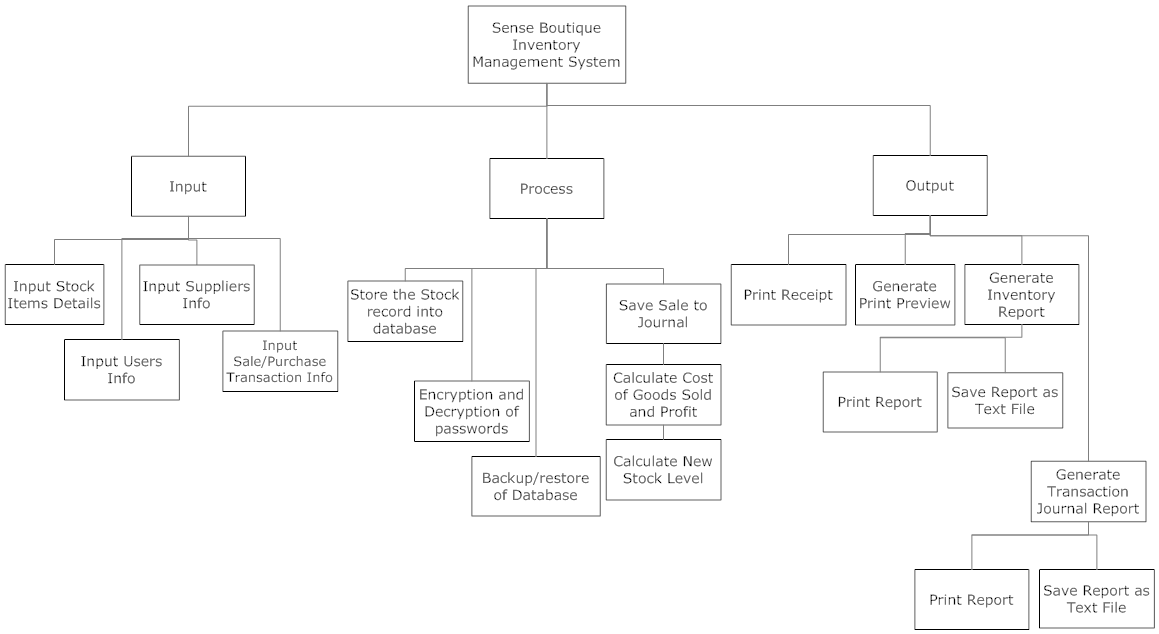
## System Flowchart



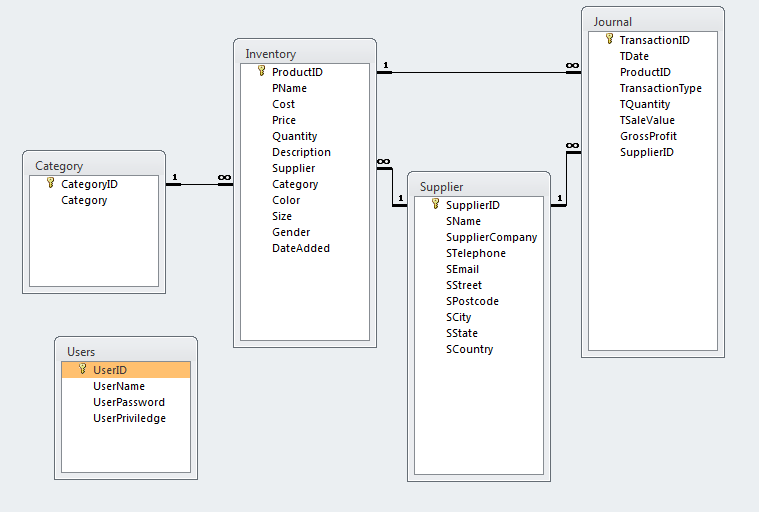
## Gantt chart



## Jackson Structured Programming



## Entity-Relationship Diagram (ERD)



## Files, Records, and Data Structures

To store the various records and data required by the system, the system employs a Microsoft Access database that is stored in the application directory on the local hard disk drive. The file name of the database is named Database.mdb. There are multiple tables used inside the database to store the various records, and they are linked to each other and are able to be processed at once.

***Note: All bolded field names marked with \* are primary keys.***

#### “Users” table

The “Users” table is used to store information about each user that is authorized to access the system, with their userID and the encrypted password hashes. It also contains the user group information. UserID will be the primary key identifying each unique user.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Description | Data Type | Size (Bytes) |
| UserID\* | The unique ID for each authorized user | Number | 4 |
| UserName | The user ID chosen | Text | 30 |
| UserPassword | The password associated with each user ID. It is encrypted and stored as a hash here. | Text | 30 |
| UserPriviledge | The user group each user belongs to: Admin or User | Text | 5 |

#### “Category” table

The “Category” table is used to store the category of items which will be displayed in a list view when adding and editing items. The administrator can edit and add new categories through a built in editor in the developed system. CategoryID will be the primary key used to identify each different added item categories.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Description | Data Type | Size (Bytes) |
| CategoryID\* | The ID for the specific category | Number | 2 |
| Category | The name of the item category/type | Text | 50 |

#### “Journal” table

The “Journal” table is used to store transaction records which are added after every sale or purchases. TransactionID will be the primary key used to identify each unique transaction.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Description | Data Type | Size (Bytes) |
| TransactionID\* | The specific ID of the transaction | Number | 4 |
| TDate | Date on which the transaction occurs | Date/Time | 8 |
| ProductID | The ID of product involved in the transaction | Number | 8 |
| TransactionType | The type of transaction: Sale or Purchase | Text | 10 |
| TQuantity | The quantity traded in this transaction | Number | 4 |
| TSaleValue | The value of transaction | Currency | 8 |
| GrossProfit | The gross profit of the transaction | Currency | 8 |
| SupplierID | The ID of the supplier if it’s a Purchase transaction. ‘0’ for Sales transaction. | Number | 4 |

#### “Inventory” table

The “Inventory” table is mainly used to store information and details about each product that is current on hand. The user is able to add, edit and remove each record through the system. The productID will be the primary key used to identify each unique product added.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Description | Data Type | Size (Bytes) |
| ProductID\* | The product ID of the specific item | Number | 8 |
| PName | The name of the item | Text | 50 |
| Cost | The cost of the item | Currency | 8 |
| Price | The recommended selling price of the item | Currency | 8 |
| Quantity | The quantity on hand/in stock | Number | 4 |
| Description | The description of the item | Text | 200 |
| Supplier | The ID of the supplier linked to the “Supplier” table | Number | 4 |
| Category | The category of the item | Text | 50 |
| Color | The color of the item | Text | 20 |
| Size | The size of the item | Text | 4 |
| Gender | Which gender is the item suitable for | Text | 1 |
| DateAdded | The date on which the item is added | Date/Time | 8 |

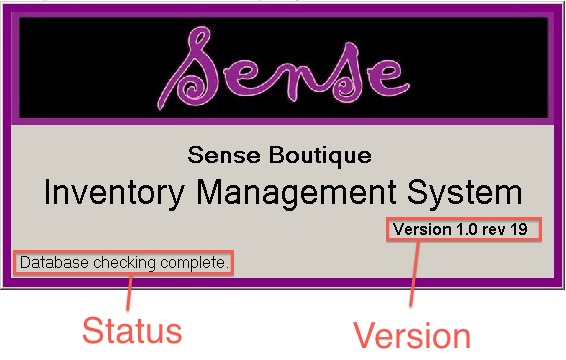
#### “Supplier” table

The “Supplier” table is used to store supplier records, including their contact information. SupplierID will be the primary key identifying each individual supplier.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Description | Data Type | Size (Bytes) |
| SupplierID\* | The unique ID of each individual supplier | Number | 4 |
| SName | The name of the supplier | Text | 100 |
| SupplierCompany | The company that the supplier belongs to | Text | 100 |
| STelephone | The telephone number of the supplier | Text | 15 |
| SEmail | The email of the supplier | Text | 100 |
| SStreet | The street name of the address | Text | 200 |
| SPostcode | The postcode of the address | Text | 10 |
| SCity | The city portion of the address | Text | 50 |
| SState | The state portion of the address | Text | 50 |
| SCountry | The country portion of the address | Text | 50 |

## Design of Forms

### Splash Screen

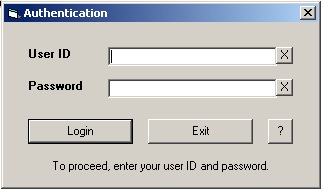


Whenever the user starts up the program, the user will be greeted with the splash screen, which will linger while the database is being loaded, displays status information and the current version number of the system.

**Label Controls**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| lblVersion | Displays the current version number. |
| LoadStatus | Displays the current program loading progress and status. |

### Login Screen



The login screen will appear after the splash screen if the database was loaded successfully. This screen allows the user to login with their unique user ID and password. It will reject users without a correct user ID and password, then after 5 times of unsuccessful login attempts, it will automatically quit the program to prevent password guessing from unauthorized users.

**TextBox Controls**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ID | Allows the user to input the user ID |
| PW | Allows the user to input the password |

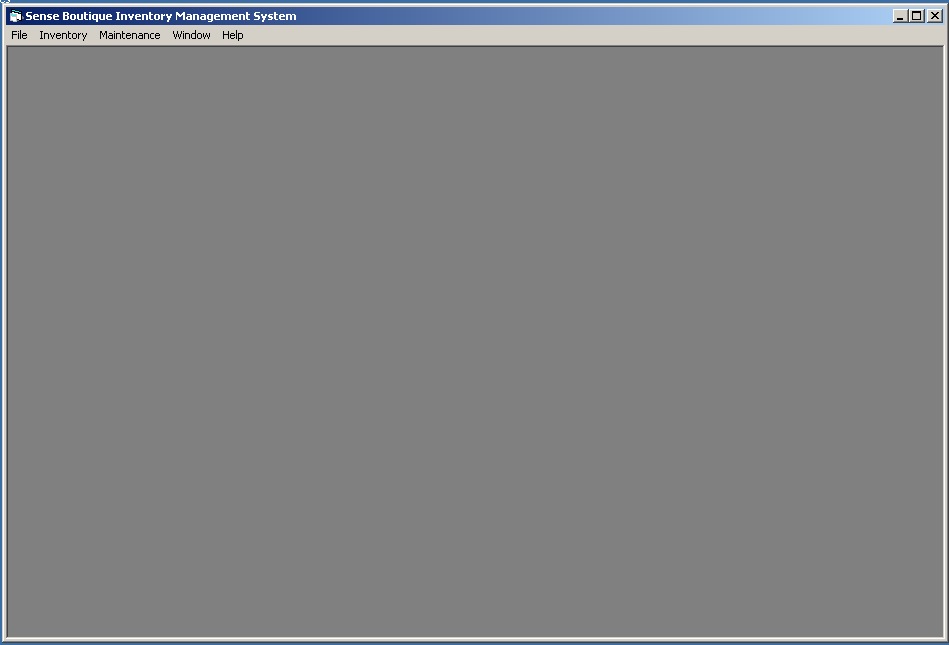
**Button Controls**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| LoginButton | Allows the user to initiate the login process |
| CancelButton | Allow the user to exit the login process |
| IDClear | Allows the user to clear the ID textbox immediately |
| PWClear | Allows the user to clear the password textbox immediately |

**Label Controls**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| LoginStatus | Displays login hints, login status and current remaining tries available (if login failed) |

### Main Screen and Menus

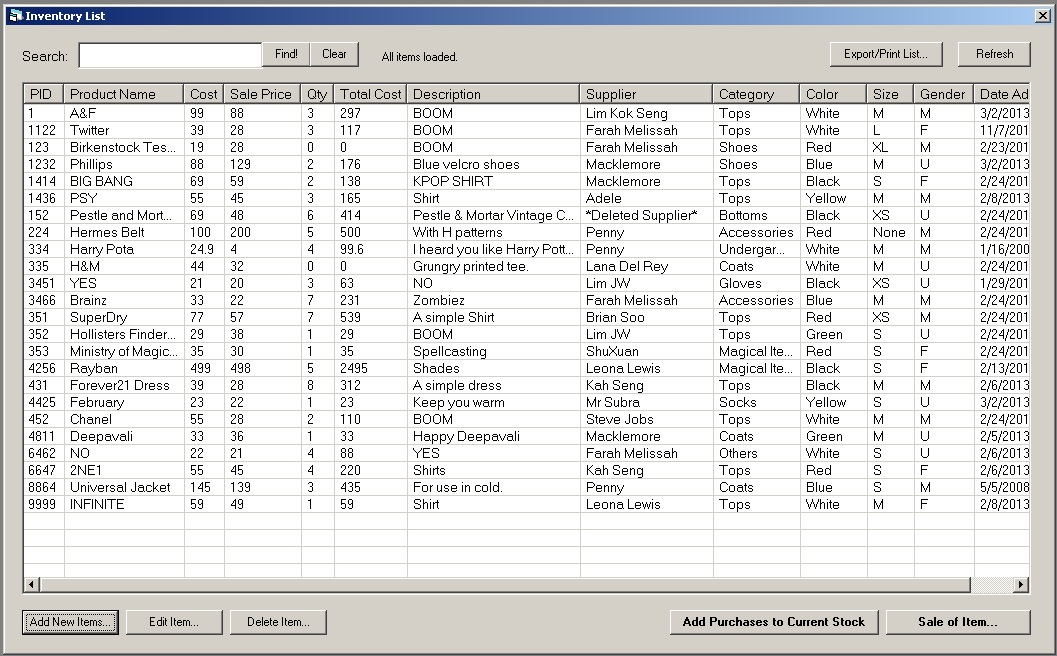


After the user entered a valid user ID and password, then successfully logged in, they will be shown this main screen with a menu bar on top. This main screen provides the user access to various features of the system and is an MDI (Multiple Document Interface) window that contains other ‘child’ windows while the system is running.

**Menus**

|  |  |  |  |
| --- | --- | --- | --- |
| **Menu Name** | | **Shortcut** | **Function** |
| File > | Logout… | Ctrl+F4 | Allows the user to logout of the system in order to let another user log in to the system. |
| Exit… | Ctrl+Q | Allows the user to quit the program. |
| Inventory > | Inventory List | Ctrl+I | Opens up the inventory list window |
| Search | F3 | Immediately focuses on the searchbox to allow the user to initiate search |
| Add New Item… | Ctrl+N | Opens up the Add New Item… window to allow the user to add new records |
| Sale of Item… | Ctrl+S | Opens up the Sale of Item window to allow the user to record a sale transaction |
| Suppliers… | - | Opens up the Suppliers Management window to allow the user to manage suppliers relationships |
| Maintenance > (Only visible to admin users) | Categories | - | Opens up the manage Categories window |
| Transaction Report | F7 | Displays the transaction journal report |
| Users Management | F8 | Shows the Users Management window to allow the admin to add, change or remove users |
| Backup/Restore | F9 | Opens up the Backup/Restore window to allow backup or restore processes to be carried out |
| Window > | - | - | This menu shows all the currently opened sub-windows in the Main window, and allows the user to switch between them easily. |
| Help > | Help Topics | F1 | Displays the HTML help documentation for explanations on various features on the program |
| Quick Start Guide | Ctrl+F1 | Displays a simple guide on how to access most frequently used functions. |
| About | **-** | Displays the current system version information and program name. |

### Inventory Window



The inventory window will be the main means through which the user will use to view all the records in the database and manage the records.

It should allow the user to select an item from the list and edit/view/delete the item easily.

It should also allow the user to add purchases or record sales data into the database.

It will also allow the user to search through the database easily.

**TextBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| SearchQuery | Allows the user to input search term to be found |

**Labels Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Status | Displays the load status of records from database, as well as displaying the search progress. |

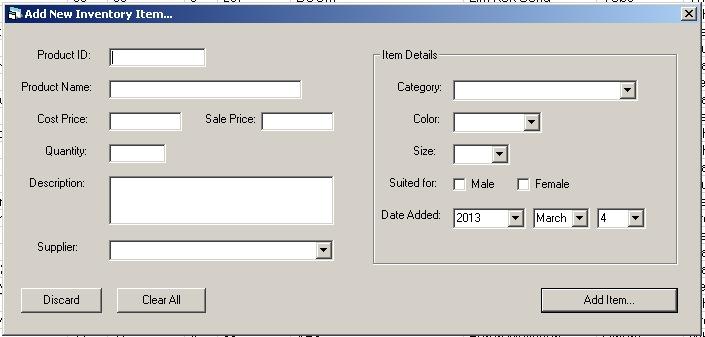
**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Search | Initiates the search process |
| ClearSearch | Clears the search box and resets the list view |
| ExportList | Opens up a preview window to allow the user to export or print all the current inventory records |
| RefreshList | Reloads all records from database |
| AddNew | Opens up the Add New Item window to allow the user to add another record to the database |
| EditItem | Opens up the Edit Item window to allow the user to edit the item currently selected in the list |
| DeleteItem | Allows the user to delete selected item from the list |
| AddStock | Allows the user to add purchases to the currently selected item (add stock level) |
| SalesButton | Allows the user to record a sales transaction on the currently selected item. |

**ListView Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ListView1 | Displays all records in a table view to allow information at a glance |

### Add New Inventory Items



This form will guide the user on how to add a new item record by filling each required items accordingly. Upon adding the item, it will also perform validation checks on each information entered to make sure they conform to the database rules and format specified.

**TextBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ProductID | To input the Product ID |
| ProductName | To input the name of the product |
| Cost | To input the cost of the product |
| SalePrice | To input the selling price |
| Quantity | To input the quantity currently in stock |
| Description | To input the description (if any) |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Close | Discard the current input and close the window |
| ClearAllButton | Clear all textboxes and inputs |
| AddButton | Add the current record into database |

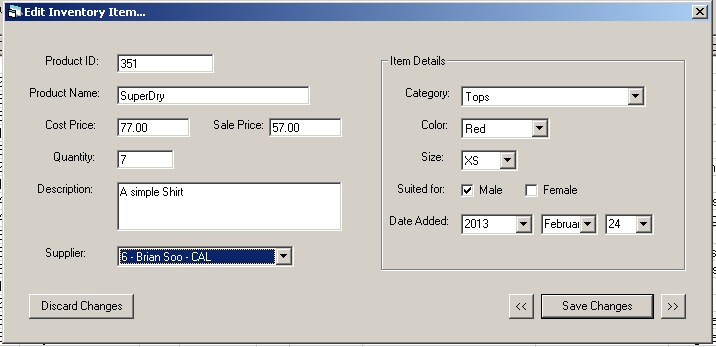
**ComboBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Supplier | To select the supplier from a list defined in Suppliers Management |
| Category | To select a category |
| Color | To select a color |
| Size | To select a size for the item |
| DateYear | To select the date added for the item |
| DateMonth |
| DateDay |

**CheckBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| GenderMale | To define the current item as for male, female or unisex (if both checked) |
| GenderFemale |

### Edit Inventory Items



This form will display all details about the selected item in Inventory List window on load, and it allows the user to edit each detail accordingly. There will be a previous and next button to allow quick navigation between items in the list.

**TextBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ProductID | To input the Product ID |
| ProductName | To input the name of the product |
| Cost | To input the cost of the product |
| SalePrice | To input the selling price |
| Quantity | To input the quantity currently in stock |
| Description | To input the description (if any) |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Close | Discard the current input and close the window |
| EditButton | Add the edits into database |
| PreviousRecord | Displays the previous record |
| NextRecord | Displays the next record |

**ComboBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Supplier | To select the supplier from a list defined in Suppliers Management |
| Category | To select a category |
| Color | To select a color |
| Size | To select a size for the item |
| DateYear | To select the date added for the item |
| DateMonth |
| DateDay |

**CheckBox Control**

|  |  |
| --- | --- |
| Control Name | Function of the Control |
| GenderMale | To define the current item as for male, female or unisex (if both checked) |
| GenderFemale |

### Sale of Item



The sale of Item form allows the user to record a sale into the database and reflect changes to the stock level accordingly. It will also output the total price and approximated gross profit.

**TextBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| PID | Disabled, only used for displaying the Product ID selected |
| ProductName | Disabled, only used for displaying the Product Name of the selected record |
| SalePrice | Allows the user to input the price in which the transaction is done |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Commit | Saves the sale transaction to database and update the stock level |
| Discard | Exits the process and lose all changes |

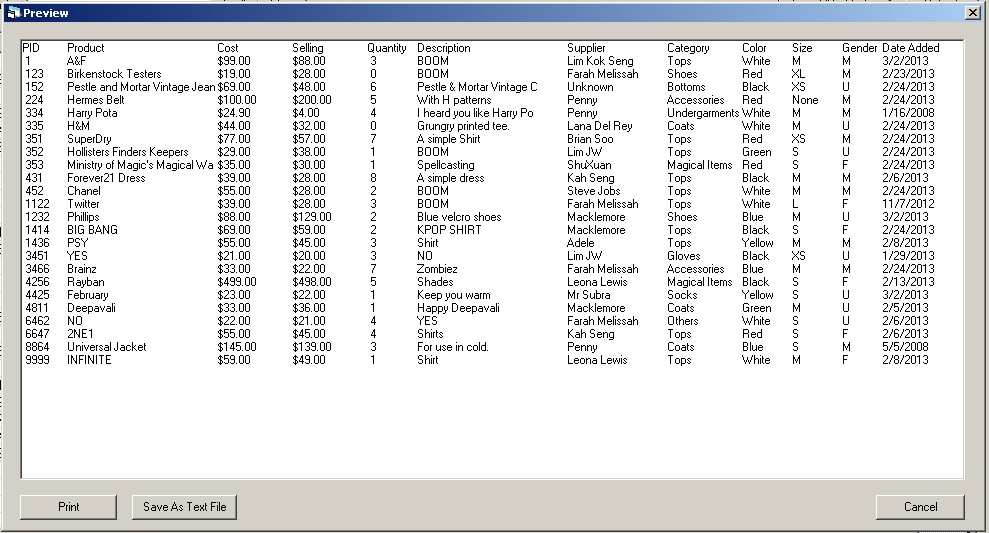
**ComboBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| QuantitySold | Allows the user to pick the amount of stock sold for this item |
| DateYear | Allows the user to pick the date (Year/Month/Day) on which the transaction has occured |
| DateMonth |
| DateDay |

**PictureBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Picture1 | Displays the Original Total Cost, Gross Profit before Tax and total value of transaction. |

### Preview Window



The preview window generates a preview of the output that is going to be outputted (either through printing to physical medium or saving to a text file), before allowing the user to finally confirm the print action or Save As action.

This window will generate preview for list of all stock items if it is called from the *Inventory List* window, or it will generate the transaction journal report if it is called from the *Transaction Report* window.

**Picturebox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Picture1 | Generates a preview of the output. |

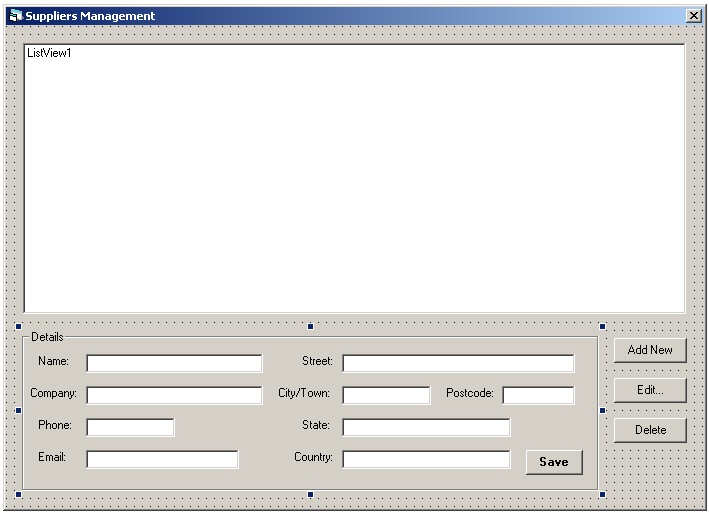
**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| PrintButton | Calls the print dialog to allow the user to print to a printer. |
| SaveAs | Allows the user to save the output in a text file. |
| CancelButton | Cancels the process and exits the current window. |

**Other Controls**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| CommonDialog1 | To provide a standard set of common dialog boxes for various operations such as Printing and Saving. |

### Suppliers Management



This form allows the user to view and manage the list of suppliers in the database. It will also allow the user to add, edit and delete supplier records easily.

**ListView control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ListView1 | To display a list of all suppliers information organised into columns |

**Frame control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Frame1 | To group the textboxes together so that they can be easily enabled and disabled. |

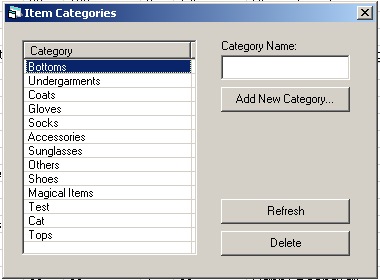
**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Add | To clear the textboxes and enable them to allow the user to add new entries |
| Edit | To enable the textboxes for editing current selected entry |
| Delete | To delete the currently selected entry |
| Save | To save the changes or new record to database. Hides when not in Add or Editing mode. |

**TextBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| SName | To allow the user to input the Supplier Name. |
| Scompany | To allow the user to input the Supplier Company |
| Sphone | To allow the user to input the Supplier’s phone number |
| Semail | To allow the user to input supplier’s email |
| Street | To guide the user to input the address of the supplier according to each sections. |
| City |
| Postcode |
| State |
| Country |

### Item Categories



This form allows the user to manage categories of items that may be chosen from the drop down combo box list in Add New or Edit Item windows.

**ListView Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ListView1 | To display a list of current categories set in the database |

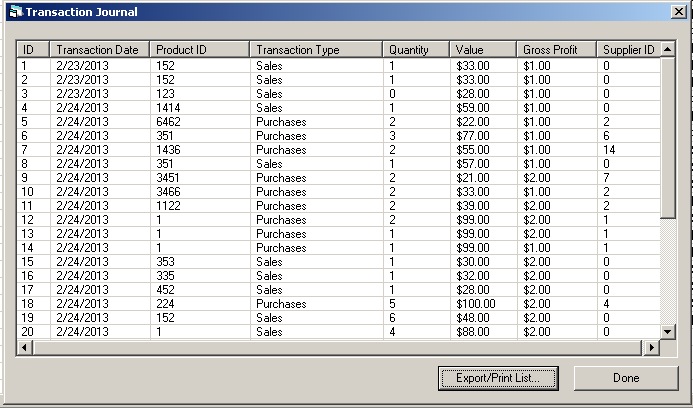
**TextBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| CatName | To allow the user to input the name for the new category |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| AddNew | To allow the user to add a new category |
| Refresh | To allow the user to refresh the list from database |
| Delete | To allow the user to delete a currently selected category |

### Transaction Report



**ListView control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ListView1 | To display a list of in and out transactions recorded in the database journal |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ExportPrint | Displays a preview window to allow the user to export or print the list |
| Done | Exits the current window |

### Users Management



This form allows the admin to easily manage a list of authorised users that are allowed to use the system and access the database. It will also allow the admin to add new users, remove existing users, change their passwords, and promote/demote the type of users.

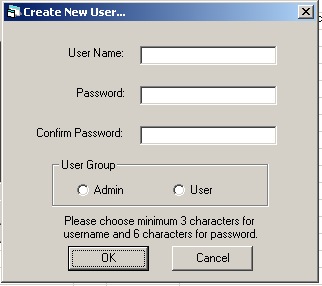
**ListView control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| ListView1 | To display a list of users and their type from the database |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| AddUser (+) | To allow the admin to add a new user |
| RemoveUser (-) | To allow the admin to remove a currently selected user |
| ChangePW | To allow the admin to change the password of the currently selected user |
| ChangeType | To toggle between Admin/User for a currently selected user in the list |

### Creating New User



This form allows the admin to add a new user and specify their user name, password and user type.

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| OK | To save the new user |
| Cancel | To cancel the operation and return to previous window |

**TextBox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| UID | To allow the admin to input the User ID |
| PW | To allow the admin to specify a password for the new user |
| ConfirmPW | To verify that the admin has inputted the correct password in the previous textbox |

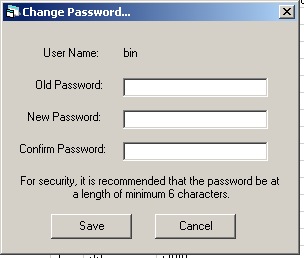
**Labels Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Status | To display helpful hints and error messages. |

**Radio Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| OptionAdmin | To set the new user as an admin. |
| OptionUser | To set the new user as a user. |

### Changing Passwords



This form allows the admin to change the password of an existing user.

**Labels Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| LabelUser | To display the user ID of the user selected to change password |
| Status | To display helpful hints and error messages (if any). |

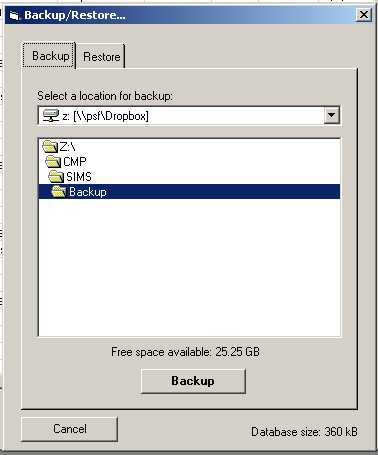
**Textbox Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| OldPW | To allow the admin to enter the old password. |
| NewPW | To allow the admin to specify a new password. |
| ConfirmPW | To confirm whether the new password specified is correct. |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| SavePW | To save the changes of password to database. |
| Cancel | To cancel the operation and return to previous window |

### Backup



This form allows the user to select a location to backup the current database.

**Tab Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| SSTab1 | To display backup and restore as separate pages accessible through tabs |

**Drive Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Drive1 | To allow the user to choose a drive to navigate to |

**Directory Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Dir1 | To allow the user to navigate to a directory on the drive selected |

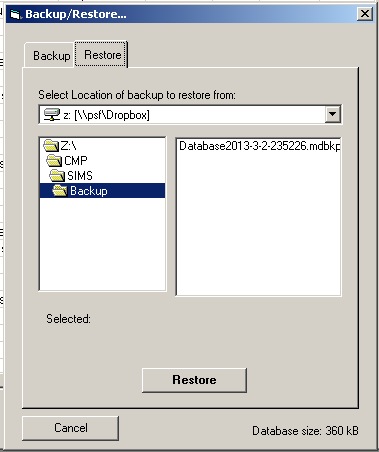
**Labels Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| FreeSpace | To display the amount of free space available |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Backup | To start the backup process |
| Cancel | To cancel the backup/restore operation |

### Restore



This form allows the user to select the backup file to restore database from.

**Labels Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| SelectedLabel | To display the currently selected backup file to the user |

**Drive Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Drive2 | To allow the user to navigate to a selected drive |

**Directory Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Dir2 | To allow the user to navigate through folders and directories on the selected drive |

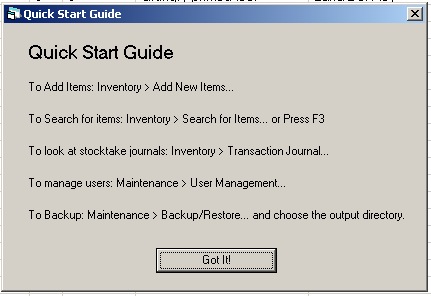
**File Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| File2 | To allow the user to select a backup file. It will only show \*.mdbkp files. |

**Button Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| Restore | To initiate the restore process. |

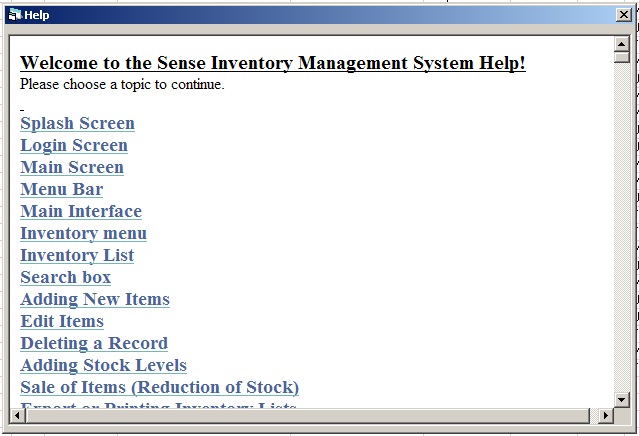
### Quick Start Guide



**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| OKButton | To dismiss the current dialog |

### Help

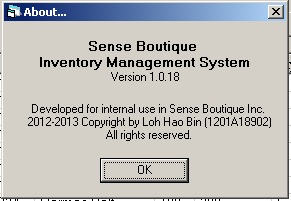


This form displays the HTML help file included with the system, to allow the user to get more information on how to use the system. It uses the Internet Explorer web browser controls that comes with Windows operating systems.

**WebBrowser Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| brwWebBrowser | To display the HTML help file |

### About



This form displays the current system version information.

**Labels Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| lblVersion | To display the current version |

**Buttons Control**

|  |  |
| --- | --- |
| **Control Name** | **Function of the Control** |
| OKButton | To dismiss the current dialog |

## Intended Benefits of the proposed system

The proposed system will be developed based on the user’s requirements, hence it will try to fulfill the user’s objectives and overcome several shortcomings of the previous system. It’s been predicted that the new system will be able to bring forth various potential benefits to multiple sides of the business, including the business owner, the system administrator, the employees, and the customer.

**Benefits to the business owner and system administrator:**

* The new system will be easier to manage
* The business owner is able to make better business decisions
* Data are now organized in a clear and neat order, hence the ability to see an overview of the inventory on hand
* Ability to print reports on list of items and all recorded transactions
* Can decide when to restock and whether to bring in more stock on a particular item because it is more popular lately
* Build better business relationships with the suppliers with the ability to store and access supplier data easily.
* Less unauthorized access to sensitive business information because of the login system implemented requiring user ID and password to access the system.
* Less physical storage space will be taken up because data are stored digitally on a storage medium, compared to the previous system where data are stored on stacks of paper documents which take up a lot of space.
* A proper backup and restore system, there will be less risk of data loss
* Less work needed to try to recover data in case of system failure
* A central method to manage users who can access the system, and ability to change their password easily
* Ability to add new users allows for more employees in the future due to future expansion of the business

**Benefits to the employee:**

* The new system should be user friendly, easy to learn and use
* Less time and effort needed to manage inventory records, hence more efficient, boredom is reduced, productivity increases, and the employees now can focus their efforts on other work such as tending to the customers and promote sales etc.
* Easier recording of sales and purchases
* Less input errors because of the built in error checking methods

**Benefits to the customer:**

* Better customer service

## Test Plan

In a business environment, the system must be able to run as stable as possible and bugs must be kept to a minimum in order for the system to be implemented and run effectively, leading to improved business operation. If a system is be riddled with bugs, causes problems and crashes easily, then it will only cause frustration to its users, cause data loss or financial damage, waste time, effort and resources.

Although tests can be conducted in order to detect problems with the system and possibly rectify them, it is nearly impossible to test every single possible combination of input to find out all bugs, as there are a multitude of hardware and software combinations out there and also various external factors that can cause problems. Hence, a test plan must be carefully constructed to relate the problems found on the system used by the client.

Hence, a comprehensive test plan has been devised in order to help detect errors and bugs that may be present in the system, and to make sure that the system is working as intended. There are several ways that the system can be tested:

1. **Validation Check**

* Validation check refers to the checking of input items to make sure that they are valid, whether they match the rules and requirement for the particular data. Validation checks include checks such as presence check, size check, range check, character check and format check.
* An invalid data should be rejected or they will cause the system to malfunction (e.g. crashes).
* In this system, the system will check for the data input on whether they are valid or not before allowing input. For example, a user will be notified if a non-numerical character is accidentally inputted into a numerical field (e.g. the quantity field)

1. **Data Verification**

* Data verification refers to the checking of the ‘correctness’ of the data, on whether it is accurate or not. However, it cannot be guaranteed that data can always be verified to be 100% correct.
* In this system, examples of data verification carried out include when an admin try to change the password of a user, and was asked to reconfirm the password. Aside from that, when a user tries to login or change password, the user will be asked for his/her password which will be checked against the password stored in the database to verify whether it is the correct password or not.

1. **Black box testing**

* Black box testing refers to testing the system with various different input values to test whether the program can cope with them and output the correct value. Data are feed into the program without thinking about the flow of the data inside the program, and only the end result is observed. Often the values used to test the system include normal data, extreme values and values that are simply not acceptable.
* In this system, black box testing will be carried out with various values in order to confirm that the system can handle a variety of inputs without issues.

1. **Alpha testing**

* Alpha testing refers to the testing of the program on whether it is working as intended, done by the developer or the software development house.
* In this case, alpha testing will be carried out by the developer after development has finished.

1. **Beta testing**

* Beta testing refers to the testing of the program by the client. Because clients generally are not computer savvy, so there might be problems encountered that are typically not encountered by the developers of the system.
* Beta testing will be carried out with the owner of the Sense Boutique and several employees in order to identify problems not encountered.

### Tests to be Carried Out

1. **Data Verification**

|  |  |
| --- | --- |
| **Test 1** | **Login Check** |
| **Valid Data:** | Input correct User ID and password |
| Expected Result: | The main window will be loaded allowing access to the system. |
| **Invalid Data:** | Incorrect UserID or password is inputted |
| Expected Result: | The system will alert the user of incorrect ID or password, and deny access to the system. |

|  |  |
| --- | --- |
| **Test 2** | **Confirm Password Check** |
| **Valid Data:** | Matching Input in both Password and Confirm Password field, when creating a new user |
| Expected Result: | A new user will be created successfully. |
| **Invalid Data:** | The input in both password and confirm password fields do not match. |
| Expected Result: | The system will display an error of both fields not matching and refuse to create a new user. |

1. **Validation Check**

|  |  |
| --- | --- |
| **Test 3** | **Presence Check** |
| **Valid Data:** | All item details are filled in, and there were no blanks. |
| Expected Result: | The entry will be added successfully, with a dialog box notifying the user of successful entry. |
| **Invalid Data:** | Some item details are deliberately left blank. |
| Expected Result: | Error dialogs appear requesting the user to fill in the blank inputs. |

|  |  |
| --- | --- |
| **Test 4** | **Character Check** |
| **Valid Data:** | Quantity, Cost and Sale Price fields are entered with valid numerical currency values. |
| Expected Result: | The entry will be added successfully, with a dialog box notifying the user of successful entry. |
| **Invalid Data:** | An alphabet is entered into these fields. |
| Expected Result: | Error dialog appear notifying the user of the non-numerical value. |

|  |  |
| --- | --- |
| **Test 5** | **Format Check – During creation of new users** |
| **Valid Data:** | Username of length greater than 3. |
| Expected Result: | User created successfully and a dialog box appears to notify the user. |
| **Invalid Data:** | Username of length smaller than 3. |
| Expected Result: | A system message will be displayed requesting the user to choose a username longer than 3 characters. |
| **Extreme Data:** | User name of length 100 |
| Expected Result: | The system will notify the user that the username chosen is too long. |

1. **Black Box Testing**

|  |  |
| --- | --- |
| **Test 6** | **Total Cost and Gross Profit Calculation** |
| **Valid Data:** | An acceptable sale price is inputted, e.g $99 |
| Expected Result: | Total cost and gross profit calculations are correctly calculated and displayed in the window. |
| **Invalid Data:** | Alphabets or symbols are inputted for selling price, e.g “A” |
| Expected Result: | The system will notify the user of an invalid selling price. |
| **Extreme Data:** | Extremely high values, e.g. $999999999999 |
| Expected Result: | The system should be able to function as normal without crashing. |

1. **Alpha Testing**

|  |  |
| --- | --- |
| **Test 7** | **Adding New Records** |
| **Valid Data:** | All items are filled in properly. |
| Expected Result: | Record added successfully. |
| **Invalid Data:** | Some items are missing or invalid. |
| Expected Result: | Dialog box appears informing the user of the error, and record was not added to the database. |

|  |  |
| --- | --- |
| **Test 8** | **Editing Records** |
| **Valid Data:** | The changes in data are filled in properly and correct. |
| Expected Result: | Changes saved successfully in the database. |
| **Invalid Data:** | Incorrect or missing data are inputted. |
| Expected Result: | Dialog box appears informing the user of the error, and record was not added to the database. |

|  |  |
| --- | --- |
| **Test 9** | **Deleting Records** |
| **Operation:** | An item is selected from the list to be deleted. |
| Expected Result: | After confirming deletion, the item should be deleted from the database. |

|  |  |
| --- | --- |
| **Test 10** | **Search Records** |
| **Valid Data:** | Search terms are entered properly, with the search term existing in the database. |
| Expected Result: | Results matching the search term should appear in the list. |
| **Invalid Data:** | Search terms that do not exist at all are entered into the search field. |
| Expected Result: | Status will report that no matching items found. |

|  |  |
| --- | --- |
| **Test 11** | **Adding Purchases** |
| **Valid Data:** | A valid number is inputted. |
| Expected Result: | The new stock level is updated. |
| **Invalid Data:** | A non numerical value is inputted, e.g “A”, “D” |
| Expected Result: | Error message appears informing the user of the invalid input. |

|  |  |
| --- | --- |
| **Test 12** | **Adding Sales Record** |
| **Valid Data:** | A valid quantity sold and sale price is inputted. |
| Expected Result: | Sale successfully recorded with a dialog box informing the user. |

|  |  |
| --- | --- |
| **Test 13** | **Print and Export Inventory List** |
| **Operation:** | Clicking the Print button. |
| Expected Result: | A printer common dialog should appear, and then the output is successfully sent to the printer. |
| **Operation:** | Clicking the Save to Text File button. |
| Expected Result: | A file browser dialog should appear, and save a text file at the selected location containing the output. |

|  |  |
| --- | --- |
| **Test 14** | **Print and Export Transaction Journal Report** |
| **Operation:** | Clicking the Print button. |
| Expected Result: | A printer common dialog should appear, and then the output is successfully sent to the printer. |
| **Operation:** | Clicking the Save to Text File button. |
| Expected Result: | A file browser dialog should appear, and save a text file at the selected location containing the output. |

|  |  |
| --- | --- |
| **Test 15** | **Adding, Editing and Deleting Suppliers** |
| **Operation:** | Adding a supplier, with all the inputs boxes properly filled. |
| Expected Result: | The new supplier is successfully added to the database and now appears in the list above. |
| **Operation:** | A record is selected, and the Edit button is pressed. |
| Expected Result: | Save button appears and the input boxes becomes editable, allowing the user to edit the details. |
| **Operation:** | A record is selected and confirmed for deletion. |
| Expected Result: | The entry is deleted from the list and the database. |

|  |  |
| --- | --- |
| **Test 16** | **Backup and Restore** |
| **Operation:** | Backup: A valid writable location is chosen, and the backup process is initiated. |
| Expected Result: | The backup process completes and backup file appears at the destination with name “Database[Date+Time].mdbkp” |
| **Operation:** | Restore: A backup file is chosen from the location, and restore process is started. |
| Expected Result: | The restore process completes, and a dialog box appears informing the user that the system will be restarted. |

|  |  |
| --- | --- |
| **Test 17** | **Adding, Changing and Removing Users** |
| **Operation:** | Adding a new user with a username longer than 3 characters, and matching passwords longer than 6 characters. |
| Expected Result: | User successfully added. |
| **Operation:** | Deleting a user selected from the list. |
| Expected Result: | User successfully removed. |
| **Operation:** | Selecting a user from the list, and change the user type. |
| Expected Result: | The user changes type successfully. |

|  |  |
| --- | --- |
| **Test 18** | **Changing User Passwords** |
| **Valid Data:** | Correct old password and matching new passwords (with length >6) are entered. |
| Expected Result: | The password is successfully changed. |
| **Invalid Data:** | Incorrect old password. |
| Expected Result: | System displays an error and password was not changed. |
| **Invalid Data:** | New Password does not match the Confirm Password |
| Expected Result: | System displays an error and password was not changed. |
| **Invalid Data:** | New password is less than 6 characters long. |
| Expected Result: | System displays an error and password was not changed. |
| **Invalid Data:** | New password is the same as the old password |
| Expected Result: | System displays an error and password was not changed. |

|  |  |
| --- | --- |
| **Test 19** | **Managing Categories** |
| **Valid Data:** | A valid category name with length less than 30 is entered. |
| Expected Result: | New category successfully added into the database with a prompt notification. |
| **Extreme/Invalid Data:** | A category name with length 100 is added. |
| Expected Result: | An error message appears and the category was not added. |
| **Operation:** | A category is selected and deleted. |
| Expected Result: | The category should be deleted from the database and removed from the list. |

## Limitations

Despite the various benefits offered by the new system, there are also several limitations and disadvantages to the new system.

**- Software and system limitations**

Aside from that, there are also several obvious limitations to the new system. First of all, it cannot make business decisions for the owner because it is not intelligent enough to do so, for example analyse the popularity of a particular item and order more to fulfill the demand. Then, the new system is unable to automatically import the previous records from the paper documents; hence some effort is still needed to add the old records to the new system. Thirdly, the new system still requires some level of user engagement in order to work effectively, i.e it is not fully automated. The new system developed can only work on Windows operating system, which is currently the most used operating system in the world, so this is not much of a problem for now.

Lastly, there may still be bugs in the new system because it is custom developed and haven’t been thoroughly tested on a large variety of environment and machines.

**- Resources, budget constraints and financial availability**

The new computerized system requires the purchase of a computer (old or new), and various hardware peripheral devices such as a printer, barcode scanner and USB flash drives in order to work to its maximum potential. A network or internet connection may also need to be setup if the client wishes to connect multiple systems together or to connect to the Internet. On the other hand, the purchase and licensing of genuine software including Windows, Microsoft Office, and Microsoft Visual Basic can also incur a tremendous amount of cost to the client. Hence, the plan may be not viable for a firm that is financially tight or doesn’t have the budget to implement it, unless it is confident that it may be able to recover the incurred costs in the long run.

Compared with the previous paper based filing system, the new system requires the usage of electricity, which may cause higher operating costs due to the extra electricity fees expenses incurred.

**- Time constraints**

The client has requested that the system should be up and running by the April of 2013, hence due to the short timeframe in which the development is allowed, several non-essential features have to be removed and only key aspects of the system have to be developed. This ensures that the developer is able to meet the client’s requirements without violating the deadline.

### File Size Calculations

The database file used by the new system comprises a total of five tables that stores different data relating to the operation of the system.

1. Category table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Size (Bytes) |
| CategoryID | Number | 2 |
| Category | Text | 50 |
| Total Size Per Record (Bytes) | | 52 |
| Number of Records | | 25 |
| Total Size of Table (Bytes) | | 1300 |

1. Inventory table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Size (Bytes) |
| ProductID | Number | 8 |
| PName | Text | 50 |
| Cost | Currency | 8 |
| Price | Currency | 8 |
| Quantity | Number | 4 |
| Description | Text | 200 |
| Supplier | Number | 4 |
| Category | Text | 50 |
| Color | Text | 20 |
| Size | Text | 4 |
| Gender | Text | 1 |
| DateAdded | Date/Time | 8 |
| Total Size Per Record (Bytes) | | 365 |
| Number of Records | | 600 |
| Total Size of Table (Bytes) | | 219000 |

1. Journal table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Size (Bytes) |
| TransactionID | Number | 4 |
| TDate | Date/Time | 8 |
| ProductID | Number | 8 |
| TransactionType | Text | 10 |
| TQuantity | Number | 4 |
| TSaleValue | Currency | 8 |
| GrossProfit | Currency | 8 |
| SupplierID | Number | 4 |
| Total Size Per Record (Bytes) | | 54 |
| Number of Records | | 1000 |
| Total Size of Table (Bytes) | | 54000 |

1. Supplier table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Size (Bytes) |
| SupplierID | Number | 4 |
| SName | Text | 100 |
| SupplierCompany | Text | 100 |
| STelephone | Text | 15 |
| SEmail | Text | 100 |
| SStreet | Text | 200 |
| SPostcode | Text | 10 |
| SCity | Text | 50 |
| SState | Text | 50 |
| SCountry | Text | 50 |
| Total Size Per Record (Bytes) | | 679 |
| Number of Records | | 50 |
| Total Size of Table (Bytes) | | 33950 |

1. Users table

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Size (Bytes) |
| UserID | Number | 4 |
| UserName | Text | 30 |
| UserPassword | Text | 30 |
| UserPriviledge | Text | 5 |
| Total Size Per Record (Bytes) | | 69 |
| Number of Records | | 6 |
| Total Size of Table (Bytes) | | 414 |

**Total Estimated File Size of Database**

= All Total Size of Table + 10% Overhead

= (1300 + 219000 + 54000 + 33950 + 414) \* 110%

= 308664 \* 110%

**= 339530.4 bytes**

= 339530.4/1024 (Convert from bytes to kB)

**= 331.57 kilobytes (kB)**

= 331.57/1024 (Convert from kB to MB)

**= 0.324 Megabytes (MB)**

## Program Listing (Coding)

### Base Module (Module1.bas)

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'\* SENSE INVENTORY MANAGEMENT SYSTEM \*

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'Program Title : Sense Inventory Management System

'Developer : Loh Hao Bin (1201A18902)

'Version Number : 1.0

'Date : 03/03/2013

'Language : Microsoft Visual Basic 6.0

'Dependencies : Microsoft Common Dialog Control 6.0 (SP3)

' Microsoft ADO Data Control 6.0 (OLEDB)

' Microsoft Internet Controls

' Microsoft Tabbed Dialog Controls 6.0

' Microsoft Windows Common Controls 6.0 (SP6)

' Microsoft DAO 3.6 Object Library

' Microsoft Scripting Runtime

'

'Name of Client : Sense Boutique Inc.

'Main Features : Keeping records of stock

' Add, Edit and remove inventory records

' Search through records

' Manage supplier’s records

' Produce a stock inflow and outflow journal report

'

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Public SessionUserLevel As Integer

'To store the current session user priviledge: Admin or user?

Public PreviewFlag As String

'To store the current session flag for Preview window, in order to determine whether to generate print preview for Inventory or Transaction Journal

### Splash Screen (FormSplash.frm)

Private Sub Form\_Load()  
On Error GoTo FileError  
  
'Display the current version  
lblVersion.Caption = "Version " & App.Major & "." & App.Minor & " rev " & App.Revision  
  
'Check database and do startup maintenance  
Call CheckDatabase  
Call RepairDB  
  
'After finish checking, enable the timer to countdown load  
LoadStatus.Caption = "Database checking complete. "  
Timer1.Enabled = True  
Exit Sub  
  
FileError:  
    'If there is somekind of problem with the database  
    'Ask the user whether they want to restore from backup  
    AskIfRestore = MsgBox("Database.mdb cannot be found or corrupted. Would you like to restore from a backup?", vbQuestion + vbYesNo, "Error")  
    If AskIfRestore = vbYes Then  
        'If yes, then load limited FormBackup wiht only Restore visible  
        Load FormBackup  
        FormBackup.Show  
        FormBackup.SetFocus  
        FormBackup.SSTab1.Tab = 1  
        FormBackup.SSTab1.TabVisible(0) = False  
    Else  
        'If no, then ask if the user would like to create a new one.  
        'Then copy the Default database to the application directory  
        AskIfReset = MsgBox("The program cannot continue without a database. Would you like to create a new one?", vbExclamation + vbYesNo, "Reset")  
        If AskIfReset = vbYes Then  
                FileCopy App.Path & "/Backup/Default.MDBDefault", App.Path & "/Database.mdb"  
                MsgBox "Successfully created a new database!", vbOKOnly, "Restore Complete"  
                MsgBox "Default username: bin, password: 000", vbInformation, "Default Login"  
                Call Timer1\_Timer  
        Else  
            'If the user chose No to both, the program will exit.  
            MsgBox "No database can be found, the program cannot continue without a database. The program will now exit.", vbCritical + vbOKOnly, "Error"  
        End If  
    End If  
Unload Me  
End Sub

Private Sub CheckDatabase()  
Set db = OpenDatabase(App.Path & "\Database.MDB") 'Check if Database can be accessed  
  
LoadStatus.Caption = "Checking database..."  
'Check if each required tables are available  
Set rst = db.OpenRecordset("Inventory")  
Set rst = db.OpenRecordset("Supplier")  
Set rst = db.OpenRecordset("Category")  
Set rst = db.OpenRecordset("Users")  
Set rst = db.OpenRecordset("Journal")  
Set rst = Nothing  
db.Close  
End Sub

Private Sub RepairDB()  
LoadStatus.Caption = "Compacting Database..."  
  
'Calling the CompactDatabase command to Compact and Repair database  
'Removing redundant entries and reduce file size  
DBEngine.CompactDatabase App.Path & "\Database.MDB", App.Path & "\Database2.MDB"  
'Delete the old database (before compaction)  
Kill (App.Path & "\Database.MDB")  
'Renaming the newly compacted database to become the new database  
Name App.Path & "\Database2.MDB" As App.Path & "\Database.MDB"  
End Sub

Private Sub Timer1\_Timer()  
'Wait for 1 second before showing the login form  
Load FormLogin  
FormLogin.Show  
Unload Me  
End Sub

### Login Screen (Form1.frm)

'This form is a security feature that allows the user to login to the system  
  
Dim AQCount As Integer  
'AQCount to keep track of number of times the user has tried to login but failed in the current session

Private Sub Form\_Load()  
'Initialise AQCount  
AQCount = 0  
'Initialise status to provide hints to user to login  
Label3.Caption = "To proceed, enter your user ID and password."  
End Sub

Private Sub IDClear\_Click()  
'Clear UserID  
ID.Text = ""  
End Sub

Private Sub PWClear\_Click()  
'Clear Password field  
PW.Text = ""  
End Sub

Private Sub LoginButton\_Click()  
'Check if the user has attempted login more than 5 times  
If AQCount < 5 Then  
    Call CheckLogin  
End If  
End Sub

Private Sub IncorrectLogin()  
'If incorrect login  
'Clear forms  
        ID.Text = ""  
        PW.Text = ""  
        Message = MsgBox("Incorrect login ID/Password, please try again.", vbOKOnly, "Login Error")  
        ID.SetFocus     'Refocus on UserID  
        AQCount = AQCount + 1   'Increment AQCount  
        Label3.Caption = "Remaining logins: " & (5 - AQCount) 'Change status to display remaining trials available  
        If AQCount >= 5 Then  
            'If tried more than 5 times, quit the application  
            Message = MsgBox("You have tried more than 5 times, the program will now close.", vbOKOnly, "Login Failure")  
            Close  
            Unload Me  
        End If  
End Sub

Private Sub CheckLogin()  
Dim ToBeDecrypted As String  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form an SQL expression to find if user exists  
TargetUser = "SELECT Users.\* " & "FROM Users " & "WHERE Users.UserName = '" & ID.Text & "'"  
'Open the result set based on the query  
Set rst = db.OpenRecordset(TargetUser)  
'If the result set is empty, user does not exist and call Incorrectlogin  
If rst.EOF And rst.BOF Then  
            Call IncorrectLogin  
Else  
'User exists, check password now  
    ToBeDecrypted = rst(2)  
    'Load the password hash from database  
    'Pass to decrypting function  
    Call DecryptPassword(ToBeDecrypted)  
        If PW.Text = ToBeDecrypted Then  
            'If the decrypted hash in database matches the password inputted by user  
            'Load the main window and unload the login form  
            Main1.Show  
            Unload Me  
            'Check if the current user is admin and set priviledge accordingly  
            If rst(3) = "Admin" Then  
                    SessionUserLevel = 0  
            Else  
                    SessionUserLevel = 1  
            End If  
        Else  
        'If decrypted hash doesn't match the password inputted, call IncorrectLogin  
            Call IncorrectLogin  
        End If  
End If  
'Unload everything  
rst.Close  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub

Private Sub DecryptPassword(ByRef PW As String)  
'Decrypt the encrypted hash from database  
Dim Decrypted As String  
'Initialise Decrypted variable  
Let Decrypted = ""  
  
'For each of the characters in the hash,  
'Take it  
'Convert the taken character to its ASCII code equivalent  
'Then take the first letter of the username and convert it to ASCII code equivalent  
'Subtract the ASCII code of username from the ASCII code of the taken character, then add 17  
'Final value is converted back to character and placed into Decrypted variable  
'Reiterate  
'Finally pass back the decrypted password to the login dialog for comparison  
For i = 1 To Len(PW)  
        Let Char = Mid(PW, i, 1)  
        Let Value = Asc(Char) - Asc(Mid(ID.Text, 1, 1)) + 17  
        Let Decrypted = Decrypted & Chr(Value)  
Next i  
Let PW = Decrypted  
End Sub

Private Sub CancelButton\_Click()  
'Ask user if they want to cancel login  
Dim Response As Integer  
Response = MsgBox("Are you sure you want to cancel login?", vbYesNo, "Cancel")  
If Response = vbYes Then  
    'If yes, then unload form  
    Close  
    Unload Me  
Else  
    'Do nothing  
End If  
End Sub

### Main Screen (MDIForm1.frm)

Private Sub MDIForm\_Activate()  
'Check user priviledge  
Call UserCheck  
  
'Show the Inventory list window automatically  
Inventory.Show  
End Sub

Private Sub UserCheck()  
If SessionUserLevel = 0 Then  
    'If admin user, show the Maintenance menu  
    Maintenance\_Menu.Enabled = True  
    Maintenance\_Menu.Visible = True  
ElseIf SessionUserLevel = 1 Then  
    'If ordinary user, hide the Maintenance menu  
    Main1.Maintenance\_Menu.Enabled = False  
    Main1.Maintenance\_Menu.Visible = False  
End If  
  
End Sub

Private Sub File\_Exit\_Click()  
'Confirm exit with user  
Message = MsgBox("Are you sure you want to quit?", vbYesNo, "Confirmation")  
If Message = vbYes Then  
    'Unload everything  
    Unload Me  
    Unload FormBackup  
    Else  
    'Do nothing  
End If  
End Sub

Private Sub File\_Logout\_Click()  
'Confirm logout with user  
Message = MsgBox("Are you sure you want to log out?", vbYesNo, "Confirmation")  
If Message = vbYes Then  
    'Unload everything  
    Unload Me  
    Unload FormBackup  
    'Show the login dialog  
    FormLogin.Show  
    Else  
    'Do nothing  
End If  
End Sub

Private Sub Help\_About\_Click()  
'Show the About form  
FormAbout.Show  
End Sub

Private Sub Help\_HelpTopics\_Click()  
'Show the Help form  
FormHelp.Show  
End Sub

Private Sub Help\_Quick\_Click()  
'Show the Quick Start Guide  
FormQuickStart.Show  
End Sub  
  
Private Sub Inventory\_Add\_Click()  
'Show the add new items form  
FormAddNew.Show  
End Sub

Private Sub Inventory\_Categories\_Click()  
'Show the Category management form  
FormCategories.Show  
End Sub

Private Sub Inventory\_List\_Click()  
'Show the main Inventory list  
Inventory.Show  
End Sub

Private Sub Inventory\_Search\_Click()  
'Focus on the Search box  
Inventory.SearchQuery.SetFocus  
End Sub

Private Sub Maintenance\_BR\_Click()  
'Show the backup form  
FormBackup.Show  
End Sub

Private Sub Maintenance\_Users\_Click()  
'Show the Users Management form  
If SessionUserLevel = 0 Then  
    FormUsers.Show  
End If  
End Sub

Private Sub Inventory\_Suppliers\_Click()  
'Show the list of suppliers  
FormSuppliers.Show  
End Sub

Private Sub Transaction\_Report\_Click()  
'Show the Transaction Journal  
FormJournal.Show  
End Sub

Private Sub Transaction\_Sales\_Click()  
'Show the Sale of Items form  
FormSales.Show  
End Sub

### Inventory List window (Inventory.frm)

'This is the main inventory list window that lists out all the current stock records in database.  
'It shows automatically during startup.  
  
Dim db As Database  
Dim rst As Recordset  
Dim srst As Recordset

Private Sub Form\_Load()  
On Error Resume Next  
  
'Populating the ListView column headers  
With ListView1.ColumnHeaders  
    .Add 1, , "PID", 600  
    .Add 2, , "Product Name", 1800  
    .Add 3, , "Cost", 600  
    .Add 4, , "Sale Price", 1150  
    .Add 5, , "Qty", 500  
    .Add 6, , "Total Cost", 1100  
    .Add 7, , "Description", 2600  
    .Add 8, , "Supplier", 2000  
    .Add 9, , "Category", 1300  
    .Add 10, , "Color", 1000  
    .Add 11, , "Size", 700  
    .Add 12, , "Gender", 900  
    .Add 13, , "Date Added", 1500  
End With  
  
'Load the listview with stock records  
Call PopulateListView  
End Sub

Private Sub ClearSearch\_Click()  
'Clear the search box  
SearchQuery.Text = ""  
  
'Reload the listview  
Call PopulateListView  
End Sub

Private Sub AddStock\_Click()  
    AddQty = InputBox("The quantity of item with Product ID " & ListView1.SelectedItem & " to add: ", "Add Stock")  
    If AddQty = 0 Or AddQty = "" Or IsNumeric(AddQty) = False Then  
        MsgBox "Please input a valid value and try again.", vbOKOnly, "Error"  
    Else  
        Set db = OpenDatabase(App.Path & "/Database.MDB")  
        EditedEntry = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID = " & ListView1.SelectedItem  
        Set rst = db.OpenRecordset(EditedEntry)  
        PriceforJournal = rst!Cost  
        SIDforJournal = rst!Supplier  
        rst.Edit  
        NewQuantity = rst!Quantity + AddQty  
        rst!Quantity = NewQuantity  
        rst.Update  
        rst.Close  
          
        Set rst = db.OpenRecordset("Journal")  
            rst.AddNew  
            rst!TDate = DateValue(Now)  
            rst!ProductID = ListView1.SelectedItem  
            rst!TransactionType = "Purchases"  
            rst!TQuantity = AddQty  
            rst!TSaleValue = PriceforJournal  
            rst!SupplierID = SIDforJournal  
            rst!GrossProfit = 0  
            MsgBox "The new stock level of item " & ListView1.SelectedItem & " is " & NewQuantity & ".", vbOKOnly, "Stock updated"  
            rst.Update  
            rst.Close  
            Set rst = Nothing  
            Set db = Nothing  
    End If  
Call PopulateListView  
End Sub

Private Sub Form\_Activate()  
'Reload the listview  
Call PopulateListView  
End Sub

Private Sub SalesButton\_Click()  
'Load the Sale of Items form  
Load FormSales  
FormSales.Show  
End Sub

Private Sub SQLSearch()  
On Error GoTo ErrorHandling

'Get search query from search box  
Y = SearchQuery.Text  
'Initialise FoundFlag  
Let FoundFlag = 0  
'Clear Listview items  
ListView1.ListItems.Clear  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items matching the selected criteria from database  
Query = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID LIKE '\*" & Y & "\*' or Inventory.PName LIKE '\*" & Y & "\*' or Inventory.Description LIKE '\*" & Y & "\*' or Inventory.Category LIKE '\*" & Y & "\*' or Inventory.Color LIKE '\*" & Y & "\*' or Inventory.Supplier LIKE '\*" & Y & "\*'"  
'MsgBox "Debug message: " & Query, vbOKOnly '\*\*\*For debug purposes only  
  
'Open the result set as defined by the query above  
Set BLA = db.OpenRecordset(Query)  
      
    With BLA  
        Do While Not .EOF  
            'Load the first column of list view with first column of current pointed record  
            Set StockList = ListView1.ListItems.Add(, , BLA(0))  
              
            'For columns 1 to 12  
            For i = 1 To 12  
                    'For columns 1 to 4, load from database as usual  
                    If i = 1 Or i = 2 Or i = 3 Or i = 4 Then  
                        StockList.SubItems(i) = BLA(i)  
                      
                    'For column 5, perform multiplication on the fly to calculate Total Cost  
                    ElseIf i = 5 Then  
                        StockList.SubItems(5) = BLA(2) \* BLA(4)  
                          
                    'For column 7 (Supplier), get supplier name from [Supplier]table based on the SupplierID in [Inventory] table  
                    ElseIf i = 7 Then  
                        'Form SQL expression to query items matching the selected criteria from database  
                        SQL = "Select Supplier.Sname from Supplier where Supplier.SupplierID = " & BLA(6)  
                        'Open the result set as defined by the query above  
                        Set srst = db.OpenRecordset(SQL)  
                          
                        If srst.RecordCount = 0 Then  
                            'If no record found, then just display Deleted Supplier  
                            StockList.SubItems(7) = "\*Deleted Supplier\*"  
                        Else  
                            'Else, just load as normal  
                            StockList.SubItems(7) = srst(0)  
                        End If  
                          
                    Else  
                        'Otherwise, just load like normal from the records  
                        StockList.SubItems(i) = BLA(i - 1)  
                    End If  
            Next i  
            'Move pointer to next record  
            .MoveNext  
            'Set FoundFlag to 1 to indicate item found  
            Let FoundFlag = "1"  
        Loop  
    End With  
  
'Unload recordset and database  
Set BLA = Nothing  
db.Close  
Set db = Nothing  
  
If FoundFlag = "0" Then  
    'If nothing was found, update status  
    Status.Caption = "No items matching criteria '" & Y & "' was found."  
ElseIf FoundFlag = "1" Then  
    'If found something, update number of items found  
    Status.Caption = Inventory.ListView1.ListItems.Count & " item(s) found."  
End If

Exit Sub

ErrorHandling:

Select Case Err.Number

Case 93

Status.Caption = "Invalid search string."

Case Else

Status.Caption = "Unknown error occurred."

End Select  
End Sub

Private Sub PopulateListView()  
On Error GoTo ErrorHandler  
  
'Initialise ErrorFlag  
ErrorFlag = 0  
  
'Populate the list from database  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open the Inventory table  
Set rst = db.OpenRecordset("Inventory")  
'Clear the ListView  
ListView1.ListItems.Clear  
  
'Repeat until end of Inventory table  
Do Until rst.EOF  
    'Load the first column of list view with first column of current pointed record  
    Set StockList = ListView1.ListItems.Add(, , rst(0))  
      
    'For columns 1 to 12, repeat  
    For i = 1 To 12  
      
        'For columns 1-4, just load from database  
        If i = 1 Or i = 2 Or i = 3 Or i = 4 Then  
            StockList.SubItems(i) = rst(i)  
              
        ElseIf i = 5 Then  
        'For column 5, perform on the fly multiplication for Total Cost  
            StockList.SubItems(5) = rst(2) \* rst(4)  
              
        ElseIf i = 7 Then  
        'For supplier column (7), load the supplier name from [Supplier]table based on the SupplierID in [Inventory] table  
              
            'Form SQL expression to query items matching the selected criteria from database  
            SQL = "Select Supplier.Sname from Supplier where Supplier.SupplierID = " & rst(6)  
            'Open the result set as defined by the query above  
            Set srst = db.OpenRecordset(SQL)  
              
            'Check if supplier still exists  
            If srst.RecordCount = 0 Then  
                'If not, then just show Deleted Supplier  
                StockList.SubItems(7) = "\*Deleted Supplier\*"  
            Else  
                'Else, loads his/her name.  
                StockList.SubItems(7) = srst(0)  
            End If  
              
        Else  
            'Otherwise, just load like normal from the records  
            StockList.SubItems(i) = rst(i - 1)  
        End If  
          
    Next i  
    'Move the pointer to next record  
    rst.MoveNext  
Loop  
  
'Close the recordset and database.  
Set rst = Nothing  
db.Close  
Set db = Nothing  
  
If ErrorFlag = 0 Then  
    'If no errors, then update status.  
    Status.Caption = "All items loaded."  
End If  
Exit Sub  
  
ErrorHandler:  
ErrorFlag = 1   'To update status on errors  
    Select Case Err.Number  
        Case 13 'Empty records or type mismatch  
            Status.Caption = "Records loaded but there were some information missing from your database. "  
            Resume Next  
        Case Else   'Other errors  
            Status.Caption = "There were some problems loading records from the database. Please contact the administrator for more info."  
            Resume Next  
    End Select  
End Sub

Private Sub RefreshList\_Click()  
'Reloads the items list from database.  
Call PopulateListView  
End Sub

Private Sub AddNew\_Click()  
'Loads the Add New Item window...  
FormAddNew.Show  
End Sub

Private Sub ExportList\_Click()  
'Show the preview window, and set the PreviewFlag as Inv  
Load Preview  
Preview.Show  
PreviewFlag = "Inv"  
End Sub

Private Sub DeleteItem\_Click()  
'Confirm with user on deleting the item  
Message = MsgBox("Are you sure you want to delete this item with product ID " & ListView1.SelectedItem & "?", vbExclamation + vbYesNo, "Confirm Delete")  
If Message = vbYes Then  
    Call DeleteRecord  
Else  
    'Do nothing  
End If  
End Sub

Private Sub DeleteRecord()  
'Delete the selected item from Listview  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items to be deleted matching the selected criteria from database  
    SelectedDelete = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID = " & ListView1.SelectedItem & ""  
    'Open the result set as defined by the query above  
    Set MarkedForDeletion = db.OpenRecordset(SelectedDelete)  
      
    'Delete items  
    With MarkedForDeletion  
        Do While Not .EOF  
        .Delete  
        .MoveNext  
        Loop  
    End With  
      
    'Close the database  
    Set MarkedForDeletion = Nothing  
    db.Close  
      
    'Remove from listview  
    ListView1.ListItems.Remove ListView1.SelectedItem.Index  
End Sub

Private Sub EditItem\_Click()  
'Show the Edit item form to edit the selected item  
FormEdit.Show  
End Sub

Private Sub Search\_Click()  
'When search button is clicked  
'Check if search box is empty  
If SearchQuery.Text <> "" Then  
    Call SQLSearch  
End If  
End Sub

Private Sub SearchQuery\_Change()  
'If the searchbox is empty, load all records, else Search based on the query  
If SearchQuery.Text = "" Then  
    Call PopulateListView  
Else  
    Call SQLSearch  
End If  
End Sub

Private Sub SearchQuery\_GotFocus()  
'When the searchbox is focused, change its color  
Inventory.SearchQuery.BackColor = vbYellow  
  
'Set the search button as the default button  
Search.Default = True  
End Sub

Private Sub SearchQuery\_LostFocus()  
'Reset the background color of Search box  
SearchQuery.BackColor = vbWhite  
'Removes the default button status of the search button  
Search.Default = False  
'Clears the load status  
Status.Caption = ""  
End Sub

### Add New Items (FormAddNew.frm)

Dim db As Database  
Dim rst As Recordset  
Dim GenderVariable As String

'This form allows the user to add new records to the database.  
'The user will input the required information into each field, then the program will check the data for basic input errors  
'then add it into the Inventory table as well as recording it in the transaction journal.

Private Sub Form\_Load()  
Call PopulateCategory   'Call procedure to load categories list from database  
Call DateList   'Adding dates to the date combobox list  
End Sub  
Private Sub AddButton\_Click()  
    Call EmptyEntriesCheck 'Perform Error Checking before calling procedure InsertRecord  
End Sub

Private Sub InsertRecord()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
  
'Adding items to Inventory table  
'Open recordset from Inventory table from database  
Set rst = db.OpenRecordset("Inventory")  
        rst.AddNew 'Open Empty Record  
          
        'Adding each field values to the record  
        rst!ProductID = ProductID.Text  
        rst!PName = ProductName.Text  
        rst!Cost = Cost.Text  
        rst!Price = SalePrice.Text  
        rst!Description = Description.Text  
        rst!Quantity = Quantity.Text  
        rst!Category = Category.Text  
        rst!Color = Color.Text  
        rst!Size = Size.Text  
          
        'Take the numbers on the left as Supplier ID as defined in the Supplier table  
        rst!Supplier = Left(Supplier.Text, InStr(1, Supplier.Text, " ", 1) - 1)  
  
        'Check if it's male  
        If GenderMale.Value = 1 And GenderFemale.Value = 0 Then  
            rst!Gender = "M"  
        'Check if it's female  
        ElseIf GenderMale.Value = 0 And GenderFemale.Value = 1 Then  
            rst!Gender = "F"  
        'Both selected or deselected means it's suitable for both gender  
        Else  
            rst!Gender = "U"  
        End If  
          
        'Concatenating the Month, Day, and Year into a single string, then add to database  
        DateAddedValue = Month(CDate("1 " & DateMonth)) & "/" & DateDay & "/" & DateYear  
        rst!DateAdded = DateAddedValue  
     
        rst.Update 'Save changes to record  
        rst.Close  'Closes the opened recordset  
          
'Record the new incoming item to Journal  
'Open the Journal table from database  
Set rst = db.OpenRecordset("Journal")  
        rst.AddNew 'Open new empty record  
           'Add each field values to the corresponding record fields  
           rst!TDate = DateAddedValue  
           rst!ProductID = ProductID.Text  
           rst!TransactionType = "Purchases"  
           rst!TQuantity = Quantity.Text  
           rst!TSaleValue = Cost.Text  
           'Taking the supplier ID from the left of selected Supplier string  
           rst!SupplierID = Left(Supplier.Text, InStr(1, Supplier.Text, " ", 1) - 1)  
        rst.Update 'Save changes to the record  
        rst.Close   'Closes the opened recordset  
      
    'Ask user if wanted to add another entry  
    NextEntry = MsgBox("New Entry Added to Database! Do you want to add another entry?", vbYesNo, Successful)  
    If NextEntry = vbYes Then  
        Call ClearAll   'If Yes, clear all the textboxes for the user to input another new entry  
    Else  
        'If No, unload the database and the form  
        Set rst = Nothing  
        db.Close  
        Set db = Nothing  
        Unload Me  
    End If  
      
End Sub

Private Sub ClearAllButton\_Click()  
Call ClearAll   'Call the ClearAll procedure to clear all input boxes  
End Sub

Private Sub ClearAll()  
Dim Ctrl As Control  
For Each Ctrl In Me.Controls    'For all controls in the current form  
    If TypeOf Ctrl Is TextBox Then  
        Ctrl.Text = ""          'If the control is a textbox, set it to empty string  
    End If  
    If TypeOf Ctrl Is ComboBox Then  
        Ctrl.ListIndex = -1     'If the control is a combobox, set the currently selected index to default (-1)  
    End If  
    If TypeOf Ctrl Is CheckBox Then  
        Ctrl.Value = 0          'If the control is a checkbox, unset the checkbox.  
    End If  
Next  
End Sub

Private Sub Close\_Click()  
'Ask user to confirm discarding all item  
Message = MsgBox("Are you sure you want to discard this item?", vbOKCancel, "Confirm Cancel")  
  
'If Yes, then unload the current form  
If Message = vbOK Then  
    Unload Me  
Else  
    'Do nothing  
End If  
  
End Sub

Private Sub DateList()  
'Populating the dates combobox  
  
For i = 1 To 31  
    DateDay.AddItem i   'Adding the day to Day combobox  
Next  
  
For i = 1 To 12  
    DateMonth.AddItem MonthName(i)  'Adding the months to Month combobox  
Next  
  
  
For i = 1 To 30  
    DateYear.AddItem (Year(Now) - i)    'Adding years to Years combobox for the past 30 years up to the current year  
Next  
  
'Set the default value for dates to the current date  
DateYear = Year(Now)  
DateMonth = MonthName(Month(Now))  
DateDay = Day(Now)  
  
End Sub

Private Sub PopulateCategory()  
'Populate Categories and Supplier List at runtime  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open up the category table  
Set rst = db.OpenRecordset("Category")  
  
'If it has not reached the end of the table, add the item into Category combobox, then rst.MoveNext will move the pointer to the next item and reiterate  
    Do While Not rst.EOF  
            Category.AddItem rst!Category  
            rst.MoveNext  
    Loop  
rst.Close   'Close the Category Table  
  
'Open up the Supplier Table  
Set rst = db.OpenRecordset("Supplier")  
'If it has not reached the end of Supplier table, it will concatenate the Supplier ID, Supplier Name and Supplier Company, then add the string into Supplier combobox  
    Do While Not rst.EOF  
        Supplier.AddItem rst!SupplierID & " - " & rst!SName & " - " & rst!SupplierCompany  
        rst.MoveNext    'After adding the string, move the pointer to the next record and reiterate  
    Loop  
Set rst = Nothing  
db.Close  
Set db = Nothing  
'Closes the recordset and database.  
  
'Populate Colors and Size  
Call AddColor  
Call AddSize  
End Sub

Private Sub AddColor()  
'Adding each color item into the Color combobox selection  
With Color  
    .AddItem "Red"  
    .AddItem "Green"  
    .AddItem ("Blue")  
    .AddItem ("White")  
    .AddItem ("Yellow")  
    .AddItem ("Pink")  
    .AddItem ("Grey")  
    .AddItem ("Black")  
End With  
End Sub

Private Sub AddSize()  
'Adding each size item into the Size combobox selection  
With Size  
    .AddItem ("XS")  
    .AddItem ("S")  
    .AddItem ("M")  
    .AddItem ("L")  
    .AddItem ("XL")  
    .AddItem ("None")  
End With  
  
End Sub

Private Sub EmptyEntriesCheck()  
'Input validation  
If ProductID = "" Or IsNumeric(ProductID) = False Then  
    'If ProductID is empty or not numerical, then stop input.  
    Message = MsgBox("Please check the product ID.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf ProductName = "" Then  
    'If product name is empty, then stop input  
    Message = MsgBox("Please fill in the product name.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Cost = "" Or IsNumeric(Cost) = False Then  
    'Check if cost is empty or the user has inputted a non-numerical value for cost  
    Message = MsgBox("Please check your item cost.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf SalePrice = "" Or IsNumeric(SalePrice) = False Then  
    'Check if price is empty or the user has inputted a non-numerical value for price  
    Message = MsgBox("Please check your price.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Quantity = "" Or IsNumeric(Quantity) = False Then  
    'Check if quantity is empty or if the user has inputted a non-numerical value for quantity  
    Message = MsgBox("Please check your quantity.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Supplier.ListIndex = -1 Then  
    'Check if user has selected any category, if not, then stop input  
    Message = MsgBox("Please choose supplier for this item.", vbOKOnly, "Error")  
    Exit Sub

ElseIf Category.ListIndex = -1 Then  
    'Check if user has selected any category, if not, then stop input  
    Message = MsgBox("Please choose a category for this item.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Color.ListIndex = -1 Then  
    'Check if user has selected any color, if not, then stop input  
    Message = MsgBox("Please choose a color for this item.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Size.ListIndex = -1 Then  
    'Check if user has selected any size, if not, then stop input  
    Message = MsgBox("Please choose a size for this item.", vbOKOnly, "Error")  
    Exit Sub  
Else  
'If no input error, check for repeat product IDs  
Call RepeatIDCheck  
End If  
  
End Sub

Private Sub RepeatIDCheck()  
'Check if there are existing product IDs in the database  
'If yes, the program will offer to add new purchases to existing entries.  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
Set rst = db.OpenRecordset("Inventory")  
'Open the Inventory table  
  
'Initialise FoundFlag  
Let FoundFlag = 0  
  
'Repeat until the end of recordset, if there are any existing product ID that match the new input  
Do Until rst.EOF  
    If rst(0) = ProductID.Text Then  
            Let FoundFlag = 1  
            'If found, set FoundFlag  
    End If  
    'Move to next record  
    rst.MoveNext  
Loop  
  
If FoundFlag = 0 Then  
    'If there aren't any existing records with the same ID, move on to InsertRecord  
    Call InsertRecord  
Else  
    'If there are already existing record with the same ID, ask the user whether to add new purchases instead  
    AskIfIncrease = MsgBox("There are already an item with the same ID, would you like to add to the quantity of that item instead?", vbYesNo, "Duplicate ID Found")  
    'If user say yes, call IncreaseStock  
    If AskIfIncrease = vbYes Then  
        Call IncreaseStock  
    End If  
End If  
End Sub

Private Sub IncreaseStock()  
'Add new purchases to existing entries  
  
    AddQty = InputBox("The quantity of item with Product ID " & ProductID.Text & " to add: ", "Add Stock")  
    'Call up an input box to receive input  
      
    'Check for valid input  
    If AddQty = 0 Or AddQty = "" Or IsNumeric(AddQty) = False Then  
        MsgBox "Please input a valid value and try again.", vbOKOnly, "Error"  
        Call IncreaseStock  
    Else  
        'If input is valid, update the new amount  
        'Set path of database  
        Set db = OpenDatabase(App.Path & "/Database.MDB")  
        'Form an SQL expression to select those records that match the requested productID  
        EditedEntry = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID = " & ProductID.Text  
        'Create a Result Set that matches the SQL query  
        Set rst = db.OpenRecordset(EditedEntry)  
              
            'Load data for journaling purposes  
            PriceforJournal = rst!Cost  
            SIDforJournal = rst!Supplier  
              
            rst.Edit 'Edit existing record  
            NewQuantity = rst!Quantity + AddQty  
            'Adding the user inputted quantity to existing records  
                rst!Quantity = NewQuantity  
            rst.Update  
            rst.Close  
              
        'Open the Journal table  
        Set rst = db.OpenRecordset("Journal")  
            rst.AddNew  
            'Open a new record and add the following items to the journal  
                rst!TDate = DateValue(Now)  
                rst!ProductID = ProductID.Text  
                rst!TransactionType = "Purchases"  
                rst!TQuantity = AddQty  
                rst!TSaleValue = PriceforJournal  
                rst!SupplierID = SIDforJournal  
        'Notify the user of successful entry  
        MsgBox "The new stock level of item " & ProductID.Text & " is " & NewQuantity & ".", vbOKOnly, "Stock updated"  
          
        'Unload everything  
        rst.Update  
        rst.Close  
        Set rst = Nothing  
        Set db = Nothing  
        Unload Me  
    End If  
End Sub

### Edit Item… (FormEdit.frm)

'This form allows the user to edit existing records selected in the Inventory form ListView  
  
Dim db As Database  
Dim rst As Recordset  
Dim srst As Recordset

Private Sub Close\_Click()  
Message = MsgBox("Are you sure you want to discard all changes?", vbOKCancel, "Confirm Cancel")  
If Message = vbOK Then  
    Unload Me  
Else  
    'Do nothing  
End If  
End Sub

Private Sub EditButton\_Click()  
Call EmptyEntriesCheck  
End Sub

Private Sub Form\_Load()  
'Populate the comboboxes and datelist  
Call PopulateCategory  
Call DateList  
  
'Set date comboboxes to current date  
DateYear = Year(Now)  
DateMonth = Month(Now)  
DateDay = Day(Now)  
  
'Load record from database  
Call LoadRecord  
End Sub

Private Sub DateList()  
'Populating the dates combobox  
  
For i = 1 To 31  
    DateDay.AddItem i   'Adding the day to Day combobox  
Next  
  
For i = 1 To 12  
    DateMonth.AddItem MonthName(i)  'Adding the months to Month combobox  
Next  
  
  
For i = 1 To 30  
    DateYear.AddItem (Year(Now) - i)    'Adding years to Years combobox for the past 30 years up to the current year  
Next  
  
End Sub

Private Sub PopulateCategory()  
'Populate Categories List'Populate Categories and Supplier List at runtime  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open the Category Table  
Set rst = db.OpenRecordset("Category")  
  
  
    'If it has not reached the end of the table, add the item into Category combobox, then rst.MoveNext will move the pointer to the next item and reiterate  
    Do While Not rst.EOF  
            Category.AddItem rst!Category  
            rst.MoveNext  
    Loop  
    rst.Close   'Close the Category Table  
    Set rst = Nothing  
  
  
    'Open up Supplier Table  
    Set rst = db.OpenRecordset("Supplier")  
    'If it has not reached the end of Supplier table, it will concatenate the Supplier ID, Supplier Name and Supplier Company, then add the string into Supplier combobox  
    Do While Not rst.EOF  
        Supplier.AddItem rst!SupplierID & " - " & rst!SName & " - " & rst!SupplierCompany  
        rst.MoveNext  
    Loop  
    Set rst = Nothing  
    db.Close  
    Set db = Nothing  
    'Closes the recordset and database.  
  
'Populate Colors and Size  
Call AddColor  
Call AddSize  
End Sub

Private Sub AddColor()  
'Adding each color item into the Color combobox selection  
With Color  
    .AddItem "Red"  
    .AddItem "Green"  
    .AddItem ("Blue")  
    .AddItem ("White")  
    .AddItem ("Yellow")  
    .AddItem ("Pink")  
    .AddItem ("Grey")  
    .AddItem ("Black")  
End With  
End Sub

Private Sub AddSize()  
'Adding each size item into the Size combobox selection  
With Size  
    .AddItem ("XS")  
    .AddItem ("S")  
    .AddItem ("M")  
    .AddItem ("L")  
    .AddItem ("XL")  
    .AddItem ("None")  
End With  
  
End Sub

Private Sub LoadRecord()  
On Error Resume Next  
  
    'Set path of database  
    Set db = OpenDatabase(App.Path & "/Database.MDB")  
    'Form SQL query to select records that matches the selected item from ListView  
    SelectedRecord = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID = " & Inventory.ListView1.SelectedItem & ""  
    'Open result set that matches the query  
    Set rst = db.OpenRecordset(SelectedRecord)  
        'Start loading items from record into respective fields  
        ProductID.Text = rst!ProductID  
        ProductName.Text = rst!PName  
        'Apply formatting to the cost and price to 2 decimal places  
        Cost.Text = FormatNumber(rst!Cost, 2)  
        SalePrice.Text = FormatNumber(rst!Price, 2)  
        Description.Text = rst!Description  
        Quantity.Text = rst!Quantity  
        Category.Text = rst!Category  
        Color.Text = rst!Color  
        Size.Text = rst!Size  
          
        'Form SQL Query to select supplier names from Supplier table based on supplierID stored in Inventory table  
        SQL = "Select Supplier.Sname, Supplier.SupplierCompany FROM Supplier WHERE Supplier.SupplierID = " & rst!Supplier  
        'Open the result set that matches the query above  
        Set srst = db.OpenRecordset(SQL)  
        'Set supplier combobox to default  
        Supplier.ListIndex = -1  
        'If there are no records in the result set, leave the combobox default/blank  
        If srst.RecordCount < 0 Then  
            Supplier.ListIndex = -1  
        Else  
        'Else if there are records, combobox set selection according to the SupplierID  
            Supplier.Text = rst!Supplier & " - " & srst(0) & " - " & srst(1)  
        End If  
        Set srst = Nothing  
          
          
        If rst!Gender = "M" Then 'Check if it's male  
            GenderMale.Value = 1  
            GenderFemale.Value = 0  
        ElseIf rst!Gender = "F" Then  'Check if it's female  
            GenderMale.Value = 0  
            GenderFemale.Value = 1  
        ElseIf rst!Gender = "U" Then 'Both selected or deselected means it's suitable for both gender  
            GenderMale.Value = 1  
            GenderFemale.Value = 1  
        End If  
          
        'Extracting the date stored in the database to separate components of Year, Month, Day  
        DateAddedValue = rst!DateAdded  
        DateYear = Right(DateAddedValue, 4)  
        DateMonth = MonthName(Left(DateAddedValue, InStr(1, DateAddedValue, "/", 1) - 1))  
        DateExtract = Left(DateAddedValue, Len(DateAddedValue) - 5)  
        DateDay = Right(DateExtract, Len(DateExtract) - InStr(1, DateExtract, "/", 1))  
          
        'Finish, unload database and recordset  
        Set rst = Nothing  
        db.Close  
        Set db = Nothing  
End Sub

Private Sub ChangeRecord()  
'The user clicks Commit Changes  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL query to request for the record that matches the product ID  
EditedEntry = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID = " & Inventory.ListView1.SelectedItem & ""  
'Load the record that matches the query  
Set rst = db.OpenRecordset(EditedEntry)  
        'Open the record to edit  
        rst.Edit  
        'Start to load each fields into each corresponding fields in database  
        rst!ProductID = ProductID.Text  
        rst!PName = ProductName.Text  
        rst!Cost = Cost.Text  
        rst!Price = SalePrice.Text  
        rst!Description = Description.Text  
        rst!Quantity = Quantity.Text  
        rst!Supplier = Left(Supplier.Text, InStr(1, Supplier.Text, " ") - 1)  
        rst!Category = Category.Text  
        rst!Color = Color.Text  
        rst!Size = Size.Text  
          
        'Check if it's male  
        If GenderMale.Value = 1 And GenderFemale.Value = 0 Then  
            rst!Gender = "M"  
        'Check if it's female  
        ElseIf GenderMale.Value = 0 And GenderFemale.Value = 1 Then  
            rst!Gender = "F"  
        'Both selected or deselected means it's suitable for both gender  
        Else  
            rst!Gender = "U"  
        End If  
          
        'Recombining the date into the database  
        DateAddedValue = Month(CDate("1 " & DateMonth)) & "/" & DateDay & "/" & DateYear  
        rst!DateAdded = DateAddedValue  
      
    'Save changes and close the database  
    rst.Update  
    rst.Close  
      
Set rst = Nothing  
db.Close  
Set db = Nothing  
'Notify the user of successful changes.  
MsgBox "Saved changes successfully!", vbOKOnly, "Edit"  
      
End Sub

Private Sub EmptyEntriesCheck()  
'Input validation  
If ProductID = "" Or IsNumeric(ProductID) = False Then  
    'If ProductID is empty or not numerical, then stop input.  
    Message = MsgBox("Please check the product ID.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf ProductName = "" Then  
    'If product name is empty, then stop input  
    Message = MsgBox("Please fill in the product name.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Cost = "" Or IsNumeric(Cost) = False Then  
    'Check if cost is empty or the user has inputted a non-numerical value for cost  
    Message = MsgBox("Please check your item cost.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf SalePrice = "" Or IsNumeric(SalePrice) = False Then  
    'Check if price is empty or the user has inputted a non-numerical value for price  
    Message = MsgBox("Please check your price.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Quantity = "" Or IsNumeric(Quantity) = False Then  
    'Check if quantity is empty or if the user has inputted a non-numerical value for quantity  
    Message = MsgBox("Please check your quantity.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Category.ListIndex = -1 Then  
    'Check if user has selected any category, if not, then stop input  
    Message = MsgBox("Please choose a category for this item.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Color.ListIndex = -1 Then  
    'Check if user has selected any color, if not, then stop input  
    Message = MsgBox("Please choose a color for this item.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Size.ListIndex = -1 Then  
    'Check if user has selected any size, if not, then stop input  
    Message = MsgBox("Please choose a size for this item.", vbOKOnly, "Error")  
    Exit Sub  
Else  
'If no input error, check for repeat product IDs  
Call ChangeRecord  
End If  
End Sub

Private Sub NextRecord\_Click()  
'When the user click the >> button  
  
'Increment the Selected item index in listview by 1  
X = Inventory.ListView1.SelectedItem.Index + 1  
  
'Check if end of listview  
'If not, select the next record and load the next record  
'Else, report end of list  
If X < Inventory.ListView1.ListItems.Count Then  
    Inventory.ListView1.ListItems(X).Selected = True  
    Call LoadRecord  
Else  
    MsgBox "You have reached the end of the list.", vbOKOnly, "Info"  
    Exit Sub  
End If  
End Sub

Private Sub PreviousRecord\_Click()  
'When the user click the << button  
  
'Decrease the Selected item index in listview by 1  
X = Inventory.ListView1.SelectedItem.Index - 1  
  
'Check if reached beginning of listview  
'If not, select the previous record and load the previous record  
'Else, report start of list  
If X > 0 Then  
    Inventory.ListView1.ListItems(X).Selected = True  
    Call LoadRecord  
Else  
    MsgBox "You have reached the start of the list.", vbOKOnly, "Info"  
    Exit Sub  
End If  
End Sub

### Sale of Item (FormSales.frm)

Dim OriginalCost As Currency  
Dim GrossProfit As Currency  
Dim TransValue As Currency

Private Sub Discard\_Click()  
'Unload the current form if Discard is clicked  
Unload Me  
End Sub

Private Sub UpdateDisplay()  
' On Error GoTo ErrorHandling   
When the user selects the quantity to be sold, or changes the sale price  
'Update the mini display picturebox below to show  
'Original total cost, total value of current transaction, and the gross profit before tax  
'Calculations  
TransValue = QuantitySold \* SalePrice.Text  
OriginalTotalCost = OriginalCost \* QuantitySold  
GrossProfit = TransValue - OriginalTotalCost  
  
'Clear the picturebox  
Picture1.Cls  
'Display the items  
Picture1.Print "Original Total Cost: " & FormatCurrency(OriginalTotalCost)  
Picture1.Print "Total Value of transaction: " & FormatCurrency(TransValue)  
Picture1.Print "Gross Profit before Tax: " & FormatCurrency(GrossProfit)  
Exit Sub

ErrorHandling:

Select Case Err.Number

Case 13

Picture1.Cls

Picture1.Print "Please input a valid selling price."

Case Else

Picture1.Print "Unknown error occurred."

End Select

End Sub

Private Sub Form\_Load()  
'Initialisation  
TransValue = 0  
OriginalCost = 0  
GrossProfit = 0  
  
'Populate the comboboxes and load certain stock info from database  
Call DateList  
Call LoadStock  
End Sub

Private Sub DateList()  
'Populating the dates combobox  
  
For i = 1 To 31  
    DateDay.AddItem i   'Adding the day to Day combobox  
Next  
  
For i = 1 To 12  
    DateMonth.AddItem MonthName(i)  'Adding the months to Month combobox  
Next  
  
  
For i = 1 To 30  
    DateYear.AddItem (Year(Now) - i)    'Adding years to Years combobox for the past 30 years up to the current year  
Next  
  
'Set date comboboxes to current date  
DateYear = Year(Now)  
DateMonth = MonthName(Month(Now))  
DateDay = Day(Now)  
End Sub

Private Sub LoadStock()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL query expression to search for items record that were selected in InventoryListView  
SoldItem = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID = " & Inventory.ListView1.SelectedItem  
'Open the result set as defined by the query above  
Set rst = db.OpenRecordset(SoldItem)  
  
'Load items  
Let PID.Text = Inventory.ListView1.SelectedItem  
ProductName.Text = rst(1)  
SalePrice.Text = FormatNumber(rst(3), 2)  
  
'Populate the selectable quantity sold to combobox  
For i = 0 To rst(4)  
    QuantitySold.AddItem i  
Next i  
  
'Initialise QuantitySold  
Let QuantitySold = 0  
Let OriginalCost = rst(2)  
  
'If the quantity recorded in database is 0  
'Report to user out of stock and stop sales  
If rst(4) = 0 Then  
    MsgBox "No sales can be committed because there isn't any stock left for this item (PID: " & PID.Text & "). Please contact your supplier to restock.", vbOKOnly, "Out of stock"  
    QuantitySold.Enabled = False  
    SalePrice.Enabled = False  
    DateDay.Enabled = False  
    DateMonth.Enabled = False  
    DateYear.Enabled = False  
    Commit.Enabled = False  
End If  
  
'Unload database and record  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub

Private Sub Commit\_Click()  
'When the user clicks Commit, check for entry validity  
Call CheckEntry  
End Sub

Private Sub CheckEntry()  
'Check if quantity sold is valid  
If QuantitySold <> 0 Then  
    'Check if sale price is valid  
    If SalePrice.Text <> "" Or IsNumeric(SalePrice.Text) Then  
        Call WriteSales  
    Else  
        MsgBox "Please input the price at which the good is sold.", vbOKOnly, "Error"  
    End If  
Else  
    MsgBox "Please select the quantity sold.", vbOKOnly, "Error"  
End If  
End Sub

Private Sub WriteSales()  
On Error GoTo ErrHandling  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query records that matches the criteria selected  
SalesEntry = "SELECT Inventory.\* " & "FROM Inventory " & "WHERE Inventory.ProductID = " & Inventory.ListView1.SelectedItem & ""  
'Open the result set as defined by the query above  
Set rst = db.OpenRecordset(SalesEntry)  
        'Open the record for edit  
        rst.Edit  
        'Saving the field datas into record  
        rst!Quantity = rst!Quantity - QuantitySold  
        rst!DateAdded = Month(CDate("1 " & DateMonth)) & "/" & DateDay & "/" & DateYear  
        'Save changes  
        rst.Update  
        'Close the record  
        rst.Close  
  
'Open the Journal table to record this transaction  
Set rst = db.OpenRecordset("Journal")  
    'Open a new record  
    rst.AddNew  
    'Adding items to this new record  
    rst!TDate = Month(CDate("1 " & DateMonth)) & "/" & DateDay & "/" & DateYear  
    rst!ProductID = Inventory.ListView1.SelectedItem  
    rst!TransactionType = "Sales"  
    rst!SupplierID = "0"  
    rst!TSaleValue = TransValue  
    rst!TQuantity = QuantitySold  
    rst!GrossProfit = GrossProfit  
    'Save Changes  
    rst.Update  
    'Close the record  
    rst.Close  
      
    Set rst = Nothing  
    db.Close  
    Set db = Nothing  
      
'Notify the user of successful entry  
MsgBox "Sales successfully recorded!"  
Unload Me  
Exit Sub  
  
ErrHandling:  
    Select Case Err.Number  
        Case 13 'Type mismatch  
            MsgBox "Invalid value entered.", vbOKOnly, "Error"  
        Case Else  
            MsgBox "Unknown error occurred.", vbOKOnly, "Error"  
    End Select  
End Sub

Private Sub QuantitySold\_Click()  
'Check if the user has selected valid quantity and sale prices  
'Then call UpdateDisplay  
If QuantitySold.ListIndex <> -1 And SalePrice.Text <> "" Then  
    Call UpdateDisplay  
Else  
    Picture1.Cls  
    Picture1.Print "Please choose a valid quantity and sale price."  
End If  
End Sub

Private Sub SalePrice\_Change()  
'Check if the user has selected valid quantity and sale prices  
'Then call UpdateDisplay  
If QuantitySold.ListIndex <> -1 And SalePrice.Text <> "" Then  
    Call UpdateDisplay  
Else  
    Picture1.Cls  
    Picture1.Print "Please choose a valid quantity and sale price."  
End If  
End Sub

### Export/Print Preview (Preview.frm)

'This form generates a print preview before exporting or printing  
  
Dim SaveDir As String  
Dim db As Database  
Dim rst As Recordset  
Dim srst As Recordset  
  
Private Sub Form\_Activate()  
On Error GoTo ErrorHandler  
  
'Check for the previewflag set as they are passed from other forms  
If PreviewFlag = "Inv" Or PreviewFlag = "Jrn" Then  
    Call GeneratePreview  
Else  
    'If no flag are set or wrong previewflags are set, then output error  
    Picture1.Print "Error generating preview: invalid preview flag set."  
    'Disable buttons  
    SaveAs.Enabled = False  
    PrintButton.Enabled = False  
End If  
Exit Sub  
  
ErrorHandler:  
Call CommonErrorHandler  
End Sub

Private Sub CommonErrorHandler()  
 Select Case Err.Number  
    Case 482    'Printer error  
        MsgBox "Error 482: The print spooler service is not started or printer is not configured properly.", vbOKOnly, "Error"  
    Case 32755  
        'User cancelled operation, so do nothing  
    Case Else  
        MsgBox "Unknown error occurred, please contact the developer for more info.", vbOKOnly, "Error"  
    End Select  
End Sub

Private Sub CancelButton\_Click()  
'If the user pressed the cancel button  
Unload Me  
PreviewFlag = ""  
End Sub

Private Sub GeneratePreview()  
'Generate print previews  
  
If PreviewFlag = "Inv" Then  
    'Set path of database  
    Set db = OpenDatabase(App.Path & "/Database.MDB")  
    'Open the Inventory table  
    Set rst = db.OpenRecordset("Inventory")  
      
    'Print the columnn headers  
    Picture1.Print Tab(0); "PID"; Tab(10); "Product"; Tab(40); "Cost"; Tab(55); "Selling"; Tab(70); "Quantity"; Tab(80); "Description"; Tab(110); \_  
                "Supplier"; Tab(130); "Category"; Tab(145); "Color"; Tab(155); "Size"; Tab(165); "Gender"; Tab(173); "Date Added"  
                  
    'Get the real supplier name from the [Supplier] table based on supplier ID from [Inventory] table  
    Do While Not rst.EOF  
        'Check if there are missing supplier information  
        'Form SQL expression to query items matching the selected criteria from database  
        SQL = "Select Supplier.Sname from Supplier where Supplier.SupplierID = " & rst(6)  
        'Open the result set as defined by the query above  
        Set srst = db.OpenRecordset(SQL)  
            If srst.RecordCount = 0 Then  
                'If the supplier record has already been deleted, set unknown  
                SupplierName = "Unknown"  
            Else  
                SupplierName = srst(0)  
            End If  
          
        'Print each record  
        Picture1.Print Tab(0); rst(0); Tab(10); Left(rst(1), 30); Tab(40); FormatCurrency(rst(2)); Tab(55); FormatCurrency(rst(3)); Tab(70); rst(4); Tab(80); Left(rst(5), 25); Tab(110); \_  
        Left(SupplierName, 20); Tab(130); Left(rst(7), 14); Tab(145); rst(8); Tab(155); rst(9); Tab(165); rst(10); Tab(173); rst(11)  
        'Move pointer to next record  
        rst.MoveNext  
    Loop  
End If  
  
If PreviewFlag = "Jrn" Then  
    'Set path of database  
    Set db = OpenDatabase(App.Path & "/Database.MDB")  
    'Open the Journal table  
    Set rst = db.OpenRecordset("Journal")  
      
    'Print the columnn headers  
    Picture1.Print Tab(0); "Trans.ID"; Tab(10); "Date"; Tab(30); "Prod.ID"; Tab(43); "Type"; Tab(58); "Quantity"; Tab(68); "Trans. Value"; Tab(83); "Gross Profit"; Tab(100); "Supplier ID"  
    Do While Not rst.EOF  
        'Print each record  
        Picture1.Print Tab(0); rst(0); Tab(10); Format(rst(1), "  dd/mm/yyyy"); Tab(30); rst(2); Tab(43); rst(3); Tab(58); rst(4); Tab(68); FormatCurrency(rst(5)); Tab(83); ; FormatCurrency(rst(7)); Tab(100); rst(6)  
        'Move pointer to next record  
        rst.MoveNext  
    Loop  
End If  
  
End Sub

Private Sub PrintButton\_Click()  
On Error GoTo ErrorHandler  
  
'Show the print dialog  
CommonDialog1.ShowPrinter  
  
'Check whether did user cancel the print operation  
If Err.Number <> 32755 Then  
  
    If PreviewFlag = "Inv" Then 'If the user came from the Inventory form  
        'Set path of database  
        Set db = OpenDatabase(App.Path & "/Database.MDB")  
        'Open Inventory table  
        Set rst = db.OpenRecordset("Inventory")  
          
        'Print column headers  
        Printer.Print Tab(0); "PID"; Tab(10); "Product"; Tab(40); "Cost"; Tab(55); "Selling"; Tab(70); "Quantity"; Tab(80); "Description"; Tab(110); \_  
                "Supplier"; Tab(130); "Category"; Tab(145); "Color"; Tab(155); "Size"; Tab(165); "Gender"; Tab(173); "Date Added"  
                  
        Do While Not rst.EOF  
            'Check if there are missing supplier information  
            'Form SQL expression to query items matching the selected criteria from database  
            SQL = "Select Supplier.Sname from Supplier where Supplier.SupplierID = " & rst(6)  
            'Open the result set as defined by the query above  
            Set srst = db.OpenRecordset(SQL)  
                  
                If srst.RecordCount = 0 Then  
                    'If the supplier record has already been deleted, set unknown  
                    SupplierName = "Unknown"  
                Else  
                    SupplierName = srst(0)  
                End If  
                  
            'Print each record  
            Printer.Print Tab(0); rst(0); Tab(10); Left(rst(1), 30); Tab(40); FormatCurrency(rst(2)); Tab(55); FormatCurrency(rst(3)); Tab(70); rst(4); Tab(80); Left(rst(5), 25); Tab(110); \_  
            Left(SupplierName, 20); Tab(130); Left(rst(7), 14); Tab(145); rst(8); Tab(155); rst(9); Tab(165); rst(10); Tab(173); rst(11)  
            'Move pointer to next record  
            rst.MoveNext  
              
        Loop  
        'Ending the print operation  
        Printer.EndDoc  
        'Notify the user of successfully sending the document to printer  
        MsgBox "The inventory database has successfully been sent to the printer!", vbOKOnly, "Print"  
    End If  
  
    If PreviewFlag = "Jrn" Then 'If the user came from the Journal form  
        'Set path of database  
        Set db = OpenDatabase(App.Path & "/Database.MDB")  
        'Open journal table  
        Set rst = db.OpenRecordset("Journal")  
          
        'Print column headers  
        Printer.Print Tab(0); "Trans.ID"; Tab(10); "Date"; Tab(30); "Prod.ID"; Tab(43); "Type"; Tab(58); "Quantity"; Tab(68); "Trans. Value"; Tab(83); "Gross Profit"; Tab(100); "Supplier ID"  
          
        'Print each record  
        Do While Not rst.EOF  
            Printer.Print Tab(0); rst(0); Tab(10); Format(rst(1), "  dd/mm/yyyy"); Tab(30); rst(2); Tab(43); rst(3); Tab(58); rst(4); Tab(68); FormatCurrency(rst(5)); Tab(83); FormatCurrency(rst(7)); Tab(100); rst(6)  
            'Move pointer to next record  
            rst.MoveNext  
        Loop  
          
        'Ending the print operation  
        Printer.EndDoc  
        'Notify the user of successfully sending the document to printer  
        MsgBox "The transaction report has successfully been sent to the printer!", vbOKOnly, "Print"  
    End If  
End If  
  
'Unset the previewflag  
PreviewFlag = ""  
  
'Unload the current form  
Unload Me  
Exit Sub  
  
ErrorHandler:  
Call CommonErrorHandler  
End Sub

Private Sub SaveAs\_Click()  
On Error GoTo ErrorHandler  
  
Call SaveDialog  
'Load the save dialog  
  
'Check if the user cancelled the save dialog or not  
If SaveDir <> "" Then  
    If PreviewFlag = "Inv" Then     'If the user came from the Inventory form  
        Open SaveDir For Append As #1  
        'Open the new file for appending  
            'Set path of database  
            Set db = OpenDatabase(App.Path & "/Database.MDB")  
            'Opening the Inventory table  
            Set rst = db.OpenRecordset("Inventory")  
            'Print column headers  
            Print #1, Tab(0); "PID"; Tab(10); "Product"; Tab(40); "Cost"; Tab(55); "Selling"; Tab(70); "Quantity"; Tab(80); "Description"; Tab(110); \_  
                "Supplier"; Tab(130); "Category"; Tab(145); "Color"; Tab(155); "Size"; Tab(165); "Gender"; Tab(173); "Date Added"  
            Do While Not rst.EOF  
                'Check if there are missing supplier information  
                'Form SQL expression to query items matching the selected criteria from database  
                SQL = "Select Supplier.Sname from Supplier where Supplier.SupplierID = " & rst(6)  
                'Open the result set as defined by the query above  
                Set srst = db.OpenRecordset(SQL)  
                    If srst.RecordCount = 0 Then  
                        'If the supplier record has already been deleted, set unknown  
                        SupplierName = "Unknown"  
                    Else  
                        SupplierName = srst(0)  
                    End If  
                'Print each entry  
                Print #1, Tab(0); rst(0); Tab(10); Left(rst(1), 30); Tab(40); FormatCurrency(rst(2)); Tab(55); FormatCurrency(rst(3)); Tab(70); rst(4); Tab(80); Left(rst(5), 25); Tab(110); \_  
                Left(SupplierName, 20); Tab(130); Left(rst(7), 14); Tab(145); rst(8); Tab(155); rst(9); Tab(165); rst(10); Tab(173); rst(11)  
                'Move pointer to next record  
                rst.MoveNext  
            Loop  
        'Closing the file  
        Close #1  
        'Notify the user of successful export  
        Message = MsgBox("The list of inventory items has successfully been saved to a text file at " & SaveDir, vbOKOnly, "Save Success!")  
    End If  
  
    If PreviewFlag = "Jrn" Then     'If the user came from the Journal form  
        Open SaveDir For Append As #1  
        'Open the new file for appending  
          
        'Set path of database  
        Set db = OpenDatabase(App.Path & "/Database.MDB")  
        'Opening the Journal table  
        Set rst = db.OpenRecordset("Journal")  
          
        'Print column headers  
        Print #1, Tab(0); "Trans.ID"; Tab(10); "Date"; Tab(30); "Prod.ID"; Tab(43); "Type"; Tab(58); "Quantity"; Tab(68); "Trans. Value"; Tab(83); "Gross Profit"; Tab(100); "Supplier ID"  
        Do While Not rst.EOF  
            'Print each entry  
            Print #1, Tab(0); rst(0); Tab(10); Format(rst(1), "  dd/mm/yyyy"); Tab(30); rst(2); Tab(43); rst(3); Tab(58); rst(4); Tab(68); FormatCurrency(rst(5)); Tab(83); FormatCurrency(rst(7)); Tab(100); rst(6)  
            'Move pointer to next record  
            rst.MoveNext  
        Loop  
        'Closing the file  
        Close #1  
        'Notify the user of successful export  
        Message = MsgBox("The journal has successfully been saved to a text file at " & SaveDir, vbOKOnly, "Save Success!")  
    End If  
End If  
  
'Unset the previewflag  
PreviewFlag = ""  
  
'Unload the current form  
Unload Me  
Exit Sub  
  
ErrorHandler:  
Call CommonErrorHandler  
End Sub

Private Sub SaveDialog()  
'Call the Save Common Dialog  
'Define the save dialog title  
CommonDialog1.DialogTitle = "Save as File..."  
  
'Define flags for the commondialog  
'cdlOFNHideReadOnly - Hides the read only checkbox  
'cdlOFNPathMustExist - The user can only input valid paths  
'cdlOFNOverwritePrompt - The user will be notified if there is an existing file with same name, and ask if to overwrite it  
CommonDialog1.Flags = cdlOFNHideReadOnly + cdlOFNPathMustExist + cdlOFNOverwritePrompt  
  
'Define the file type  
CommonDialog1.Filter = "Text Documents(\*.txt)|\*.txt"  
  
'Display the Save common dialog  
CommonDialog1.ShowSave  
  
'After the save dialog is closed, set SaveDir to the selected file path  
SaveDir = CommonDialog1.FileName  
Exit Sub  
End Sub

### Suppliers Management (FormSuppliers.frm)

Dim NewOrEdit As Integer  
'To define whether the user is editing or adding a new record, Edit:0, New:1  
  
Private Sub Form\_Load()  
'Load the column headers  
With ListView1.ColumnHeaders  
    .Add 1, , "ID", 400  
    .Add 2, , "Dealer Name", 2000  
    .Add 3, , "Company Name", 2000  
    .Add 4, , "Telephone", 1200  
    .Add 5, , "Email Address", 3000  
    .Add 6, , "Address", 7000  
End With  
  
'Populate the listview  
Call PopulateListView  
Call LoadDetails  
End Sub

Private Sub PopulateListView()  
'Populate the list from database  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open the supplier table  
Set rst = db.OpenRecordset("Supplier")  
'Clear the ListView  
ListView1.ListItems.Clear  
  
  
Do Until rst.EOF  
    Set StockList = ListView1.ListItems.Add(, , rst(0))  
    'Load each column into listview  
    For i = 1 To 4  
        StockList.SubItems(i) = rst(i)  
    Next  
    'For address, concatenate the separate fields into one  
    StockList.SubItems(5) = rst(5) & ", " & rst(6) & " " & rst(7) & ", " & rst(8) & ", " & rst(9) & "."  
    'Go to next record  
    rst.MoveNext  
Loop  
  
'Close the database and record  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub

Private Sub LoadDetails()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query Supplier records that matches the selected criteria  
SQL = "SELECT Supplier.\* FROM Supplier WHERE Supplier.SupplierID = " & ListView1.SelectedItem  
'Open the recordset as defined from the query  
Set rst = db.OpenRecordset(SQL)  
    'Start to load items  
    SName.Text = rst!SName  
    SCompany.Text = rst!SupplierCompany  
    STelephone.Text = rst!STelephone  
    SEmail.Text = rst!SEmail  
    Street.Text = rst!SStreet  
    City.Text = rst!SCity  
    Postcode.Text = rst!SPostcode  
    State.Text = rst!SState  
    Country.Text = rst!SCountry  
Set rst = Nothing  
db.Close  
Set db = Nothing  
  
'Disable the frame from being edited  
'Hide the save button  
Frame1.Enabled = False  
Save.Visible = False  
End Sub

Private Sub ListView1\_Click()  
'Whenever the user click on an entry, automatically loads the details  
Call LoadDetails  
End Sub

Private Sub Add\_Click()  
'The user presses Add New button  
  
Let NewOrEdit = 1  
'Make the save button visible  
Save.Visible = True  
'Allow editing in the form below  
Frame1.Enabled = True  
  
'Clear all textboxes in the form  
Dim Ctrl As Control  
For Each Ctrl In Me.Controls  
    If TypeOf Ctrl Is TextBox Then  
        Ctrl.Text = ""  
    End If  
Next  
End Sub

Private Sub Edit\_Click()  
'The user presses Edit button  
  
Let NewOrEdit = 0  
'Make the save button visible  
Save.Visible = True  
'Allow editing in the form below  
Frame1.Enabled = True  
End Sub

Private Sub Save\_Click()  
'Check for entry validity  
Call EntryCheck  
End Sub

Private Sub EntryCheck()  
'Input validation  
If SName = "" Then  
    'Check if Name is empty  
    Message = MsgBox("Please fill a valid name.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf SCompany = "" Then  
    'Check if Company Name is empty  
    Message = MsgBox("Please fill a valid company name.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf STelephone = "" Or IsNumeric(STelephone) = False Then  
    'Check if Telephone is empty or Numerical  
    Message = MsgBox("Please fill a valid telephone number.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf SEmail = "" Then  
    'Check if Email is empty  
    Message = MsgBox("Please fill a valid email.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Street = "" Then  
    'Check if Street is empty  
    Message = MsgBox("Please fill a valid street address", vbOKOnly, "Error")  
    Exit Sub  
ElseIf City = "" Then  
    'Check if City is empty  
    Message = MsgBox("Please fill a valid city.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Postcode = "" Or IsNumeric(Postcode) = False Then  
    'Check if Postcode is empty or numerical  
    Message = MsgBox("Please fill a valid postcode.", vbOKOnly, "Error")  
    Exit Sub  
ElseIf State = "" Then  
    'Check if State is empty  
    Message = MsgBox("Please fill a valid state", vbOKOnly, "Error")  
    Exit Sub  
ElseIf Country = "" Then  
    'Check if Country is empty  
    Message = MsgBox("Please fill a valid country.", vbOKOnly, "Error")  
    Exit Sub  
Else  
'No input error  
Call SaveCalls  
End If  
End Sub

Private Sub SaveCalls()  
'The user presses the save button  
  
If NewOrEdit = 0 Then 'If the NewOrEdit flag is 0, then edit record  
    Call EditRecord  
ElseIf NewOrEdit = 1 Then 'If the NewOrEdit flag is 1, then add new record  
    Call AddRecord  
Else  
    'Do nothing  
End If  
    'Reload the listview and disable editing for now  
    Call PopulateListView  
    Frame1.Enabled = False  
    Save.Visible = False  
End Sub

Private Sub AddRecord()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open the supplier table  
Set rst = db.OpenRecordset("Supplier")  
    'Open a new record  
    rst.AddNew  
    'Start to add items to database  
    rst!SName = SName.Text  
    rst!SupplierCompany = SCompany.Text  
    rst!STelephone = STelephone.Text  
    rst!SEmail = SEmail.Text  
    rst!SStreet = Street.Text  
    rst!SCity = City.Text  
    rst!SPostcode = Postcode.Text  
    rst!SState = State.Text  
    rst!SCountry = Country.Text  
    'Save changes  
    rst.Update  
    rst.Close  
'Close database and recordset  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub

Private Sub EditRecord()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items matching the criteria from database  
SQL = "SELECT Supplier.\* FROM Supplier WHERE Supplier.SupplierID = " & ListView1.SelectedItem  
'Open the result set as defined by the query above  
Set rst = db.OpenRecordset(SQL)  
    'Open the record for edit  
    rst.Edit  
    'Saving each fields into the database  
    rst!SName = SName.Text  
    rst!SupplierCompany = SCompany.Text  
    rst!STelephone = STelephone.Text  
    rst!SEmail = SEmail.Text  
    rst!SStreet = Street.Text  
    rst!SCity = City.Text  
    rst!SPostcode = Postcode.Text  
    rst!SState = State.Text  
    rst!SCountry = Country.Text  
    'Save changes  
    rst.Update  
    rst.Close  
'Close database and recordset  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub

Private Sub Delete\_Click()  
'Confirm with user on whether to delete or not  
Message = MsgBox("Are you sure you want to delete this supplier?", vbYesNo, "Confirm Delete")  
If Message = vbYes Then  
    Call DeleteRecord  
Else  
    'Do nothing  
End If  
End Sub

Private Sub DeleteRecord()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items matching the criteria from database  
    SelectedDelete = "SELECT Supplier.\* " & "FROM Supplier " & "WHERE Supplier.SupplierID = " & ListView1.SelectedItem & ""  
    'Open the result set as defined by the query above  
    Set MarkedForDeletion = db.OpenRecordset(SelectedDelete)  
    'Delete records  
    With MarkedForDeletion  
        Do While Not .EOF  
        .Delete  
        .MoveNext  
        Loop  
    End With  
    Set MarkedForDeletion = Nothing  
    db.Close  
    'Removing it from listview  
    ListView1.ListItems.Remove ListView1.SelectedItem.Index  
End Sub

### Transaction Journal/Report (FormJournal.frm)

'This form displays the list of inventory inflow/outflow journal records from database  
'and allows the user to print/export the list.

Private Sub Done\_Click()  
'If user clicks Cancel, unload the form  
Unload Me  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Private Sub ExportPrint\_Click()  
'When the user clicks the Export/Print Button, show preview window  
Preview.Show  
'Set the global preview flag to Jrn, to activate modules in FormPreview relating to Journals  
PreviewFlag = "Jrn"  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Private Sub Form\_Load()  
'Populate the column headers  
With ListView1.ColumnHeaders  
    .Add 1, , "ID", 500  
    .Add 2, , "Transaction Date", 1500  
    .Add 3, , "Product ID", 1500  
    .Add 4, , "Transaction Type", 2000  
    .Add 5, , "Quantity", 1000  
    .Add 6, , "Value", 1000  
    .Add 7, , "Gross Profit", 1200  
    .Add 8, , "Supplier ID", 1000  
End With  
'Load the journal entries from database  
Call LoadLog  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub LoadLog()  
'Clear the listview  
ListView1.ListItems.Clear  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open the Journal table  
Set rst = db.OpenRecordset("Journal")  
  
'Start to populate the journal entries  
Do Until rst.EOF  
    'Add each items to listview  
    Set JournalList = ListView1.ListItems.Add(, , rst(0))  
            For i = 1 To 7  
                If rst(i) <> "" Then  
                    If i = 5 Or i = 6 Then  
                        'If the column is Cost or Gross Profit, format as currency  
                        JournalList.SubItems(i) = FormatCurrency(rst(i))  
                    Else    'Else just load directly as normal  
                        JournalList.SubItems(i) = rst(i)  
                    End If  
                Else  
                    'If there is missing data  
                    JournalList.SubItems(i) = "\*Null\*"  
                End If  
            Next i  
    'Move to next record  
    rst.MoveNext  
Loop  
  
'Close the database and recordset after finish loading  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub

### Categories Management (FormCategories.frm)

'This form allows the user to manage categories.  
  
Private Sub Form\_Load()  
'Adding each column header to the ListView  
With ListView1.ColumnHeaders  
    .Add 1, , "Category", 2500  
End With  
  
'After adding column headers, load categories from database  
Call RefreshList  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub RefreshList()  
'Clear all items from listview  
ListView1.ListItems.Clear  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
Set rst = db.OpenRecordset("Category")  
'Load the Category recordset table  
  
'Load all category names from database until end of record  
Do Until rst.EOF  
    Set CategoryList = ListView1.ListItems.Add(, , rst(1))  
    'Move the pointer to the next record  
    rst.MoveNext  
Loop  
  
'Unload the recordset and database  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Private Sub Refresh\_Click()  
'Reload categories from database  
Call RefreshList  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub AddNew\_Click()

On Error goto ErrorHandling  
'Adding new categories to the database  
  
'Check if category name inputted is empty  
If CatName <> "" Then  
    'Set path of database  
    Set db = OpenDatabase(App.Path & "/Database.MDB")  
    'Open the category table  
    Set rst = db.OpenRecordset("Category")  
    'Open a new record to be added  
    rst.AddNew  
    'Add to record  
    rst!Category = CatName  
    'Save changes to record  
    rst.Update  
    'Close the record  
    rst.Close  
'Reload the categories from database  
Call RefreshList  
Else  
'If it's empty, notify the user to input a valid name for category  
    MsgBox "Please type in a valid category name."  
End If

Exit Sub

ErrorHandling:

Select Case Err.Number

Case 3163

MsgBox "Please limit the new category name to less than 30 characters.", vbOKOnly, "Error"

Exit Sub

Case Else

MsgBox "Error occurred:" & Err.Number & ". Please report the error to the developer.", vbOKOnly, "Error"

End Select  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Delete\_Click()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
  
'Ask if user want to delete the selected category  
Confirm = MsgBox("Are you sure you want to remove " & ListView1.SelectedItem & "? ", vbYesNo, "Confirm Delete")  
  
If Confirm = vbYes Then  
    'If yes, form an SQL query to look for the selected item from Category table  
    SQL = "SELECT Category.\* " & "FROM Category " & "WHERE Category.Category = '" & ListView1.SelectedItem & "'"  
    'Load the result set based on the query  
    Set MarkedForDeletion = db.OpenRecordset(SQL)  
      
    'Delete all records that match the queried criteria in the result set  
    With MarkedForDeletion  
        Do While Not .EOF  
        .Delete  
        .MoveNext  
        Loop  
    End With  
    'Unload database and resultset  
    Set MarkedForDeletion = Nothing  
    db.Close  
      
    'Removing that item from listview  
    ListView1.ListItems.Remove ListView1.SelectedItem.Index  
Else  
    'Do nothing  
End If  
  
'Reload items from database  
Call RefreshList  
End Sub

### Backup/Restore (FormBackup.frm)

'This form allows the user to backup and restore database.  
'It will also be loaded during startup if there is a need for restoring database  
  
Private Sub Form\_Load()  
On Error GoTo ErrorHandling  
  
'Initialise the initial path selected in the directory view controls  
Dir1.Path = App.Path & "\Backup\"  
Dir2.Path = App.Path & "\Backup\"  
  
'Call functions to check free space and database size  
Call FreeSpaceCheck  
Call DatabaseSize  
Exit Sub  
  
ErrorHandling:  
Select Case Err.Number  
    Case 53  
        'Silencing the error if database is not found, to allow Restore Database dialog to load during startup  
        DBSize.Caption = ""  
    Case Else  
        MsgBox "Unknown error occurred. Please contact the administrator for more info."  
End Select  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub DatabaseSize()  
'Check for the database file size  
FileSize = FileLen(App.Path & "/Database.MDB")  
  
'Convert obtained database file size to other more readable equivalent units by dividing 1024  
If FileSize < 1000 Then  
    DBSize.Caption = "Database size: " & (FileSize) & " bytes"  
ElseIf FileSize > 1000 Then  
    DBSize.Caption = "Database size: " & (FileSize / 1024) & " kB"  
ElseIf FileSize > 1000000 Then  
    DBSize.Caption = "Database size: " & (FileSize / 1024 ^ 2) & " MB"  
ElseIf FileSize > 1000000000 Then  
    DBSize.Caption = "Database size: " & (FileSize / 1024 ^ 3) & " GB"  
End If  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Private Sub FreeSpaceCheck()  
'Setting the variables for file system scripting operations  
Dim FileSystem As Scripting.FileSystemObject  
Dim DriveList As Drives  
Dim Drive As Drive  
  
Set FileSystem = New Scripting.FileSystemObject  
Set DriveList = FileSystem.Drives  
'Refreshing the list of drives currently connected to the system  
  
'Taking the first character of the drive selected by user in the list, to obtain the Drive Letter  
DriveLetter = Left(Drive1.Drive, 1)  
  
'Looking through the list of drives connected to the system as defined by DriveList  
'If the name of drive matches the one selected by user  
'Display available space  
'Convert bytes to equivalent more readable units  
For Each Drive In DriveList  
  If Drive.DriveLetter = UCase(DriveLetter) Then  
    If Drive.AvailableSpace > 1000000000000# Then  
        FreeSpace.Caption = "Free space available: " & Format(Drive.AvailableSpace / (1024 ^ 4), "###.00") & " TB"  
    ElseIf Drive.AvailableSpace > 1000000000 Then  
        FreeSpace.Caption = "Free space available: " & Format(Drive.AvailableSpace / (1024 ^ 3), "###.00") & " GB"  
    ElseIf Drive.AvailableSpace > 1000000 Then  
        FreeSpace.Caption = "Free space available: " & Format(Drive.AvailableSpace / (1024 ^ 2), "###.00") & " MB"  
    ElseIf Drive.AvailableSpace > 1000 Then  
        FreeSpace.Caption = "Free space available: " & Format(Drive.AvailableSpace / (1024 ^ 1), "###.00") & " kB"  
    Else  
        FreeSpace.Caption = "Free space available: " & Drive.AvailableSpace & " bytes"  
    End If  
  End If  
Next  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Private Sub Drive1\_Change()  
'When the user selects a different drive from the list  
On Error GoTo ErrorHandling  
  
'Set path to the drive selected  
Dir1.Path = Drive1.Drive  
'Check for free space available on the drive  
Call FreeSpaceCheck  
Exit Sub  
  
ErrorHandling:  
    Select Case Err.Number  
        Case 68  
            'If the drive is not ready, not responding, or not connected  
            FreeSpace.Caption = "Drive not connected or not available."  
        Case Else  
            Resume Next  
    End Select  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Private Sub Backup\_Click()  
Dim DBKPCount As Integer, BKName As String  
'Set path of database  
Set Dbase = OpenDatabase(App.Path & "\Database.MDB")  
'Check if database can be accessed.  
Dbase.Close  
  
'Calling the CompactDatabase command to Compact and Repair database  
'Removing redundant entries and reduce file size  
DBEngine.CompactDatabase App.Path & "\Database.MDB", App.Path & "\Database2.MDB"  
  
'Delete the old database (before compaction)  
Kill (App.Path & "\Database.MDB")  
  
'Renaming the newly compacted database to become the new database  
Name App.Path & "\Database2.MDB" As App.Path & "\Database.MDB"  
  
'Ask for user confirmation on database restore  
ConfirmBackup = MsgBox("This will start the backup process.", vbQuestion + vbOKCancel, "Confirmation")  
  
'If yes, start the backup process  
'The name of the backup is automatically generated from the current date and time  
'Copy the database to the user selected directory and rename it to the \*Name of Backup\*  
'Display a dialog box to confirm that backup operation has completed  
If ConfirmBackup = vbOK Then  
        BKName = "\Database" & Year(Now) & "-" & Month(Now) & "-" & Day(Now) & "-" & Hour(Now) & Minute(Now) & Second(Now) & ".mdbkp"  
        FileCopy App.Path & "\Database.MDB", Dir1.Path & BKName  
        MsgBox "Backup completed!", vbInformation, "Backup"  
End If  
      
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Drive2\_Change()  
'Set path displayed in Dir2  
Dir2.Path = Drive2.Drive  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Dir2\_Change()  
'Set file paths to be displayed in File2  
File2.Path = Dir2.Path  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub File2\_Click()  
'Obtain filepath and name from currently selected file  
FileName = File2.Path & "\" & File2.FileName  
  
'Show the currently selected file  
Label5.Caption = "Selected: " & FileName  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
  
Private Sub Restore\_Click()  
On Error Resume Next  
  
ConfirmRestore = MsgBox("Are you sure you want to restore this database?", vbQuestion + vbYesNo, "Confirmation")  
'Ask for user confirmation on database restore  
  
'If yes, start restoration process  
  
If ConfirmRestore = vbYes Then  
    'Backing up the current database  
    FileCopy App.Path & "/Database.MDB", App.Path & "/Backup/BackupBeforeRestore.mdbkp"  
    'Copy user selected backup to main location  
    FileCopy FileName, App.Path & "/Database.MDB"  
      
    'Notify the user of successful completion  
    MsgBox "Restoration completed!", vbInformation, "Restore Complete"  
    MsgBox "Database have been restored, the program is now restarting.", vbInformation + vbOKOnly, "Restore Complete"  
      
    'Unload everything and restarts the program  
    Unload Me  
    Unload Main1  
    Load FormSplash  
    FormSplash.Show  
    Unload Me  
End If  
  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Private Sub Command2\_Click()  
'Unload the current form  
Unload Me  
End Sub

### Users Management (FormUsers.frm)

Dim db As Database  
Dim rst As Recordset  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub AddUser\_Click()  
'When the user clicks the + button, call the NewUser form  
Load FormNewUser  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub ChangePW\_Click()  
'When the user clicks change password  
UserPassword.Show  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub ChangeType\_Click()  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items matching the selected criteria from database  
TargetUser = "SELECT Users.\* " & "FROM Users " & "WHERE Users.UserName = '" & ListView1.SelectedItem & "'"  
'Open the result set as defined by the query above  
Set rst = db.OpenRecordset(TargetUser)  
'Open the record for edit  
rst.Edit  
  
If rst!UserPriviledge = "Admin" Then  
    'if the user is already an admin, change to user  
    X = MsgBox("Are you sure you want to change this user's type to 'USER'?", vbYesNo, "Change User Type")  
    If X = vbYes Then  
        rst!UserPriviledge = "User"  
    End If  
ElseIf rst!UserPriviledge = "User" Then  
    'If the user is already a user, change to admin  
    X = MsgBox("Are you sure you want to change this user's type to 'ADMIN'?", vbYesNo, "Change User Type")  
    If X = vbYes Then  
        rst!UserPriviledge = "Admin"  
    End If  
End If  
  
'Save changes, close the database and record  
rst.Update  
rst.Close  
Set rst = Nothing  
db.Close  
Set db = Nothing  
  
'Reload the users list  
Call LoadUsers  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Form\_Activate()  
Call LoadUsers  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Form\_Load()  
'Populate the listview column headers  
With ListView1.ColumnHeaders  
    .Add 1, , "User", 2000  
    .Add 2, , "Type", 1000  
End With  
'Populate user lists  
Call LoadUsers  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub LoadUsers()  
'Clear the listview  
ListView1.ListItems.Clear  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open the users table  
Set rst = db.OpenRecordset("Users")  
  
'Add each column and row into the listview  
Do Until rst.EOF  
    Set UserList = ListView1.ListItems.Add(, , rst(1))  
    UserList.SubItems(1) = rst(3)  
    'Move the pointer to next record  
    rst.MoveNext  
Loop  
  
'Close the recordset and database  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub RemoveUser\_Click()  
'Confirm delete with user  
DeleteCheck = MsgBox("Are you sure you want to delete this user:" & ListView1.SelectedItem & "?", vbYesNo, "Confirm Delete")  
    If DeleteCheck = vbYes Then  
        'If yes, set path of database  
        Set db = OpenDatabase(App.Path & "/Database.MDB")  
        'Form SQL expression to query items matching the selected criteria from database  
        SQL = "SELECT Users.\* " & "FROM Users " & "WHERE Users.UserName = '" & ListView1.SelectedItem & "'"  
        'Open the result set as defined by the query above  
        Set MarkedForDeletion = db.OpenRecordset(SQL)  
          
        'Delete all records in the result set  
        With MarkedForDeletion  
            Do While Not .EOF  
                .Delete  
                .MoveNext  
            Loop  
        End With  
        Set MarkedForDeletion = Nothing  
        'Close database  
        db.Close  
        'Remove from listview  
        ListView1.ListItems.Remove ListView1.SelectedItem.Index  
    End If  
End Sub

### Add New Users (FormNewUser.frm)

'This form will be called when the admin wants to add a new user to the system.  
  
Dim UserPriviledge As String  
  
Private Sub Cancel\_Click()  
'If the user clicks cancel, dismiss the form  
Unload Me  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Form\_Load()  
'Provide simple hints to the user on the minimum character requirement  
Status.Caption = "Please choose minimum 3 characters for username and 6 characters for password."  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub OK\_Click()  
'The user clicks OK  
  
'Check if Password or UserID is empty  
If PW <> "" Or UID <> "" Then  
    'If the user group selected is Admin  
    If OptionAdmin.Value = True Then  
        UserPriviledge = "Admin"  
        Call UserNameCheck  
    'If the user group selected is user  
    ElseIf OptionUser.Value = True Then  
        UserPriviledge = "User"  
        Call UserNameCheck  
    'If no user group is specified  
    Else  
        Status.Caption = "Please choose a user group above."  
    End If  
Else  
'If password or userID is empty  
    Status.Caption = "Please fill in the blanks."  
End If  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub UserNameCheck()  
'Check if username already exists in database  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Open the Users table  
Set rst = db.OpenRecordset("Users")  
  
'Initialise FoundFlag  
Let FoundFlag = 0  
  
'Compare records with the user inputted string  
'If match, set FoundFlag to 1  
Do Until rst.EOF  
    If rst(1) = UID.Text Then  
            Let FoundFlag = 1  
    End If  
    rst.MoveNext  
Loop  
Set rst = Nothing  
db.Close  
Set db = Nothing  
  
'If FoundFlag is still 0,  
'then check if username is more than 3 characters  
'or check if the password is more than 6 characters  
'or check if the password and confirmed password match  
If FoundFlag = 0 Then  
    If Len(UID) < 3 Then  
        Status.Caption = "Please choose a username with longer than 3 characters."  
    Else  
        If Len(PW) < 6 Then  
            Status.Caption = "Please choose a password with longer than 6 characters."  
        Else  
            If PW <> ConfirmPW Then  
                Status.Caption = "The passwords do not match. Please try again."  
            Else  
                'Passes all tests and ready to be saved to database  
                Call SaveUser  
            End If  
        End If  
    End If  
Else  
'If FoundFlag is set to 1 then there are already existing users with the same name  
Status.Caption = "This username has already been used. Please choose another username."  
End If  
          
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub EncryptPassword(ByRef PW As String)  
'Password is passed from SaveUser procedure to be encrypted  
'Encrypt the password into a hash  
  
Dim Encrypted As String  
'Initialise Encrypted variable  
Let Encrypted = ""  
  
'For each of the characters in the hash,  
'Take it  
'Convert the taken character to its ASCII code equivalent  
'Then take the first letter of the username and convert it to ASCII code equivalent  
'Add the ASCII code of username to the ASCII code of the taken character, then subtract 17  
'Final value is converted back to character and placed into Encrypted variable  
'Reiterate  
'Finally pass back the encrypted password to the SaveUser to be stored as an encrypted hash  
For i = 1 To Len(PW)  
        Let Char = Mid(PW, i, 1)  
        Let Value = Asc(Char) + Asc(Mid(UID.Text, 1, 1)) - 17  
        Let Encrypted = Encrypted & Chr(Value)  
Next i  
Let PW = Encrypted  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub SaveUser()  
Dim PasswordTemp As String  
PasswordTemp = PW.Text  
'Pass the password to EncryptPassword to be encrypted  
Call EncryptPassword(PasswordTemp)  
  
        'Encrypted hash is now being written to database  
        'Set path of database  
        Set db = OpenDatabase(App.Path & "/Database.MDB")  
        'Open Users table  
        Set rst = db.OpenRecordset("Users")  
        rst.AddNew 'Open Empty Record  
        rst!UserName = UID.Text  
        rst!UserPassword = PasswordTemp  
        rst!UserPriviledge = UserPriviledge  
          
        'Save changes and close the database/record  
        rst.Update  
        rst.Close  
        Set rst = Nothing  
        db.Close  
        Set db = Nothing  
  
'Notify the user of successful creation of new user  
'Unload the form  
MsgBox "User created successfully!", vbOKOnly, "Success!"  
Load FormUsers  
Unload Me  
End Sub

### Changing Users Passwords (UserPassword.frm)

'This form allows the user to change password, and is accessed from the Users Management window  
Dim SelectedUser As String  
Dim TargetedUserID As Integer  
  
Private Sub Form\_Load()  
'Set selected user based on the selected user in FormUser's listview  
SelectedUser = FormUsers.ListView1.SelectedItem  
'Display selected user  
LabelUser.Caption = "User Name:        " & SelectedUser  
'Provide hint to user  
Status.Caption = "For security, it is recommended that the password be at a length of minimum 6 characters."  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items matching the selected criteria from database  
TargetUser = "SELECT Users.\* " & "FROM Users " & "WHERE Users.UserName = '" & SelectedUser & "'"  
'Open the result set as defined by the query above  
Set rst = db.OpenRecordset(TargetUser)  
'Get TargetedUserID from database  
TargetedUserID = rst(0)  
  
'Close database  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Cancel\_Click()  
'The user presses cancel  
Unload Me  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub SavePW\_Click()  
'The user presses save  
Call PWCheck  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub PWCheck()  
'Check for password validity  
  
If Len(NewPW.Text) >= 6 Then    'Check if 6 characters or more  
    If OldPW.Text = "" Or NewPW.Text = "" Or ConfirmPW.Text = "" Then   'Check if blank  
        Status.Caption = "Please fill in the blank textboxes before continuing."  
            'Highlight respective boxes if they are blank  
            If OldPW.Text = "" Then  
                OldPW.BackColor = vbYellow  
            End If  
            If NewPW.Text = "" Then  
                NewPW.BackColor = vbYellow  
            End If  
            If ConfirmPW.Text = "" Then  
                ConfirmPW.BackColor = vbYellow  
            End If  
        Else  
            If NewPW.Text = OldPW.Text Then 'Check if same as old password  
                Status.Caption = "New password cannot be the same as the old one."  
                Else  
                    If NewPW.Text <> ConfirmPW.Text Then    'Check if password is the same as confirm password  
                        Status.Caption = "The confirm password does not match the first one."  
                    Else    'Pass all tests, now verifying old password  
                        Call CheckOldPassword  
                    End If  
            End If  
        End If  
    Else  
        Status.Caption = "The new password must be more than 6 characters."  
End If  
  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub CheckOldPassword()  
Dim ToBeDecrypted As String  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items matching the selected criteria from database  
TargetUser = "SELECT Users.\* " & "FROM Users " & "WHERE Users.UserID = " & TargetedUserID & ""  
'Open the result set as defined by the query above  
Set rst = db.OpenRecordset(TargetUser)  
  
'Load old password from database  
ToBeDecrypted = rst(2)  
'Decrypt old password  
Call DecryptPassword(ToBeDecrypted)  
'Compare new password with old password  
'If they matches, then can proceed to change password  
If OldPW.Text = ToBeDecrypted Then  
    Call ChangePassword  
Else  
    'Else, block the password changing attempt.  
    Status.Caption = "Incorrect old password."  
End If  
  
'Close the record and database  
rst.Close  
Set rst = Nothing  
db.Close  
Set db = Nothing  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
  
Private Sub DecryptPassword(ByRef PW As String)  
'Decrypt the encrypted hash from database  
Dim Decrypted As String  
'Initialise Decrypted variable  
Let Decrypted = ""  
  
'For each of the characters in the hash,  
'Take it  
'Convert the taken character to its ASCII code equivalent  
'Then take the first letter of the username and convert it to ASCII code equivalent  
'Subtract the ASCII code of username from the ASCII code of the taken character, then add 17  
'Final value is converted back to character and placed into Decrypted variable  
'Reiterate  
'Finally pass back the decrypted password to the login dialog for comparison  
For i = 1 To Len(PW)  
        Let Char = Mid(PW, i, 1)  
        Let Value = Asc(Char) - Asc(Mid(SelectedUser, 1, 1)) + 17  
        Let Decrypted = Decrypted & Chr(Value)  
Next i  
Let PW = Decrypted  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Private Sub EncryptPassword(ByRef PW As String)  
'Encrypt the password into a hash  
Dim Encrypted As String  
'Initialise Encrypted variable  
Let Encrypted = ""  
  
'For each of the characters in the hash,  
'Take it  
'Convert the taken character to its ASCII code equivalent  
'Then take the first letter of the username and convert it to ASCII code equivalent  
'Add the ASCII code of username to the ASCII code of the taken character, then subtract 17  
'Final value is converted back to character and placed into Encrypted variable  
'Reiterate  
'Finally pass back the encrypted password to the SaveUser to be stored as an encrypted hash  
For i = 1 To Len(PW)  
        Let Char = Mid(PW, i, 1)  
        Let Value = Asc(Char) + Asc(Mid(SelectedUser, 1, 1)) - 17  
        Let Encrypted = Encrypted & Chr(Value)  
Next i  
Let PW = Encrypted  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub ChangePassword()  
Dim PasswordTemp As String  
PasswordTemp = NewPW.Text  
  
'Call encryption module to encrypte the password  
Call EncryptPassword(PasswordTemp)  
  
'Set path of database  
Set db = OpenDatabase(App.Path & "/Database.MDB")  
'Form SQL expression to query items matching the selected criteria from database  
TargetUser = "SELECT Users.\* " & "FROM Users " & "WHERE Users.UserID = " & TargetedUserID & ""  
'Open the result set as defined by the query above  
Set rst = db.OpenRecordset(TargetUser)  
    'Open the record for edit  
    rst.Edit  
    'Saving the hashed password to database  
    rst!UserPassword = PasswordTemp  
    rst.Update  
    rst.Close  
    MsgBox "Password successfully changed for " & SelectedUser & "!", vbOKOnly, "Success"  
'Closing the database, recordset and unload the current form  
Set rst = Nothing  
db.Close  
Set db = Nothing  
PasswordTemp = ""  
Unload Me  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Private Sub OldPW\_Click()  
'Reset background color  
OldPW.BackColor = vbWhite  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub NewPW\_Click()  
'Reset background color  
NewPW.BackColor = vbWhite  
End Sub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub ConfirmPW\_Click()  
'Reset background color  
ConfirmPW.BackColor = vbWhite  
End Sub

### Quick Start Guide (FormQuickStart.frm)

'This form is a simple quick start guide that provides a brief guide to how to use the system.  
  
Private Sub Command1\_Click()  
'Unload the current form  
Unload Me  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Form\_Load()  
'Populate each of the labels with these texts  
Label2.Caption = "To Add Items: Inventory > Add New Items..."  
Label3.Caption = "To Search for items: Inventory > Search for Items... or Press F3"  
Label4.Caption = "To look at stocktake journals: Inventory > Transaction Journal..."  
Label5.Caption = "To manage users: Maintenance > User Management..."  
Label6.Caption = "To Backup: Maintenance > Backup/Restore... and choose the output directory."  
End Sub

### User Help (FormHelp.frm)

'This form calls for a browser control that displays the HTML help file included with the system  
Public StartingAddress As String  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Private Sub Form\_Load()  
    On Error Resume Next  
    'Set startingaddress to the path of the help file  
    StartingAddress = App.Path & "\Help\Index.html"  
    'Call the webbrowser plugin to load the HTML help file  
    brwWebBrowser.Navigate StartingAddress  
End Sub

### About (FormAbout.frm)

'This form displays information about the current system  
  
Private Sub Form\_Load()  
    Me.Caption = "About... " 'Set dialog title bar  
    lblTitle.Caption = "Sense Boutique " & vbCrLf & "Inventory Management System" 'Set name of system  
    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision  'Display version numbers  
    lblDescription.Caption = "Developed for internal use in Sense Boutique Inc. " & vbCrLf & "2012-2013 Copyright by Loh Hao Bin (1201A18902)" \_  
    & vbCrLf & "All rights reserved."   'Display disclaimers  
  End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Private Sub OKButton\_Click()  
Unload Me 'Unload the current form  
End Sub  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_