Sense Boutique

Inventory Management System

Name: Loh Hao Bin
Class: 1201PB1
Student ID: 1201A18902

Table of Contents

Sense Boutique 1
Inventory Management System 1
Table of Contents 2
A) Definition, Investigation and Analysis 5
Nature of the Problem 5
Background of the Organisation 5
Nature of Business 8
Current methods used to manage the operations 9
Weaknesses of current system 10
Description of the Origin of Data 11
Investigation Method 15
Interview 16
Proposed System 20
Requirement Specification 22
Input Requirement 22
Process Requirement 22
Output Requirement 23
Hardware and Software Requirements 24
Hardware Requirements 24
Software Requirements 26
Evaluate alternative solutions 27
Final Verdict 28
B) Design 29
Aim 29
System Objectives 30
Context Diagram 32
System Flowchart 37
Gantt chart 38
Jackson Structured Programming 39
Entity-Relationship Diagram (ERD) 40

Files, Records, and Data Structures 41
Design of Forms 45
Splash Screen 45
Login Screen 46
Main Screen and Menus 47
Inventory Window 49
Add New Inventory Items 51
Edit Inventory Items 53
Sale of Item 55
Preview Window 56
Suppliers Management 57
Item Categories 59
Transaction Report 60
Users Management 61
Creating New User 62
Changing Passwords 63
Backup 64
Restore 65
Quick Start Guide 66
Help 67
About 67
Intended Benefits of the proposed system 68
Test Plan 69
Tests to be Carried Out 71
Limitations 76
File Size Calculations 77
C) Software Development, Programming, Testing and Installation 79
Program Listing (Coding) 79
Base Module (Module1.bas) 79
Splash Screen (FormSplash.frm) 80
Login Screen (Form1.frm) 84
Main Screen (MDIForm1.frm) 89
Inventory List window (Inventory.frm) 94

Add New Items (FormAddNew.frm) 107
Edit Item… (FormEdit.frm) 120
Sale of Item (FormSales.frm) 132
Export/Print Preview (Preview.frm) 140
Suppliers Management (FormSuppliers.frm) 152
Transaction Journal/Report (FormJournal.frm) 162
Categories Management (FormCategories.frm) 165
Backup/Restore (FormBackup.frm) 169
Users Management (FormUsers.frm) 177
Add New Users (FormNewUser.frm) 182
Changing Users Passwords (UserPassword.frm) 188
Quick Start Guide (FormQuickStart.frm) 196
User Help (FormHelp.frm) 197
About (FormAbout.frm) 198
Test Run 199
Data Verification 199
Validation Check 201
Black Box Testing 204
Alpha Testing 206
Implementations 234
System Changeover Method 235
System Changeover 236
Employee's Training 237
File Conversion Process 239
User Acceptance 240
E) Evaluation 241
Evaluation of objectives 241
Client's and User's Response 243

A) Definition, Investigation and Analysis

Nature of the Problem

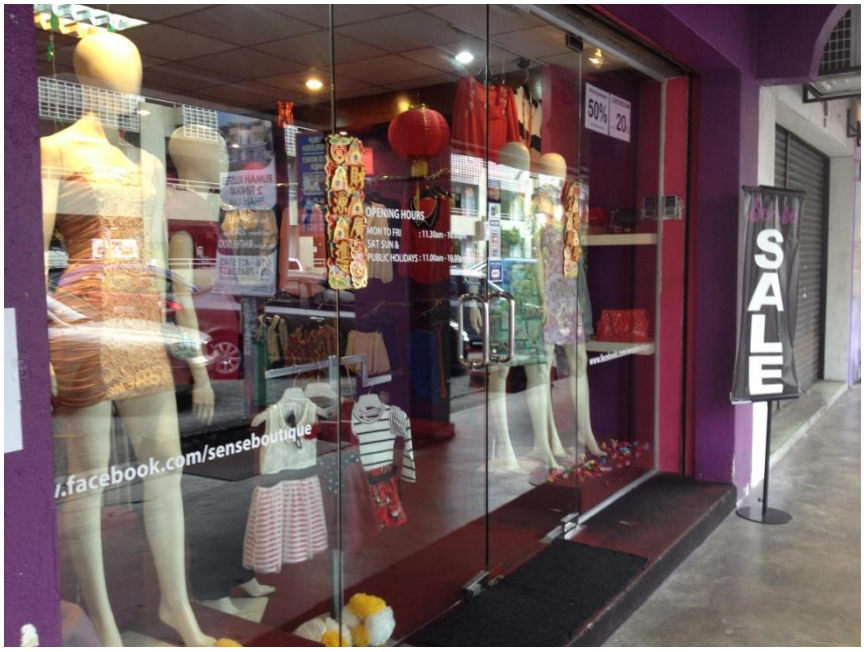
Background of the Organisation

Current Logo of the Business

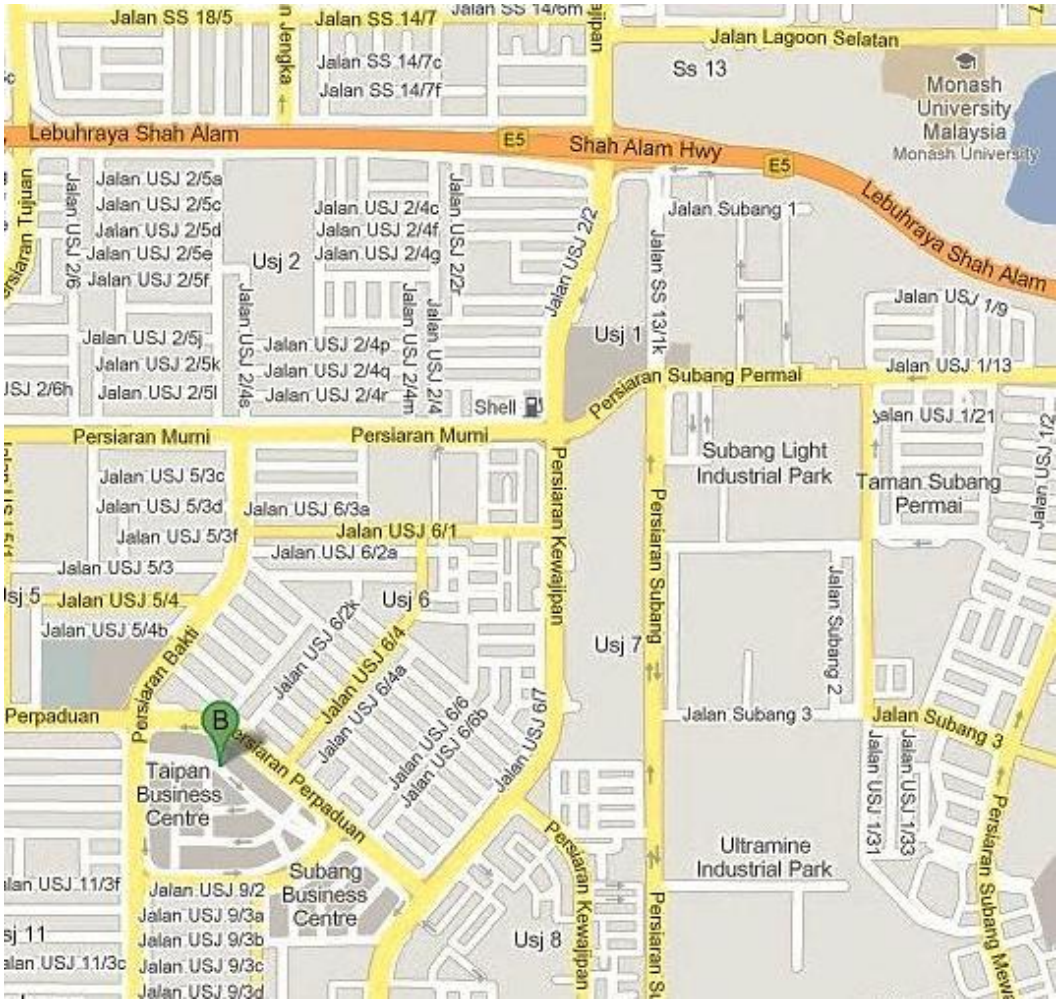
Sense Boutique is a clothing store located in the busy districts of the USJ10 Taipan Business Centre, and it was originally started in the year 2007. A large variety of latest fashionable items, ranging from clothings, accessories to handbags, can be spotted in this purple themed, fancily renovated storefront, all arranged neatly along the multitude of racks and shelves for easy perusal of customers. Currently there are three workers employed to help tend to the business.

Being situated in Taipan Business Centre, which is one of the busiest golden triangle business districts in the Subang Jaya region, Sense Boutique continues to attract a sizable amount of shoppers throughout the seasons. Opening hours are from 10.00 a.m. to 9.00 p.m. for the weekdays, while for weekends it's only opened from 10.00 a.m. to 3.00 p.m. They also provide online transaction services by allowing the customer to browse through their stock catalogue on their Facebook page, then ordering with delivery to customer's doorsteps for the convenience of the customers.

Their motto is to provide affordable, quality fashion goodies and good customer service for your satisfaction. The owner regularly scouts fashion outlets around the Klang Valley region for the latest fashion trends around the town, to bring the newest fashion items back to Subang Jaya residents.



Its new owner, Lim Ying Ying just recently took over the business in the early of year 2012 to fulfill her dreams of running her own clothing business. Backed by her extensive experience in the clothing retail business, she planned to bring about several major changes to the operations of the shop, including the incorporation of a computer system to improve business efficiency.



Location: No. 32, Jalan USJ 10/1B, Taipan Business Centre, 47620 UEP Subang Jaya, Selangor Darul Ehsan, Malaysia.
GPS Coordinate: 3.049497, 101.584745

Telephone: 03-5636 5600
Email: senseboutique@facebook.com
Facebook Page: https://www.facebook.com/SenseBoutique/

Opening Hours:
Mon - Fri: 11.30am – 10.00pm
Sat - Sun: 11.00am – 10.00pm

Nature of Business

Sense Boutique is a clothing store that offers over a thousand different selections of fashion items for both ladies and gentlemen ranging from clothing, accessories, eyewear, dresses, footwear and more. The owner periodically scouts various distributors across the Klang Valley region including the famed Kenanga Wholesale City for latest quality fashion items and updates her shop's products listing accordingly. She also got her stocks by ordering from various distributors around the Subang Jaya area, which then they will deliver the stocks to the shop.

Normal business operation consists of the customer choosing their desired product by browsing through the items around the shop; after they have finished selecting their desired clothing items, they will bring it to the cashier, and then pay for it at the payment counter. The customer can choose to pay by cash or by credit card. The business also offers a membership loyalty program that offers discounts, latest fashion updates and other benefits to recurring customers or registered customers.

On the other hand, the shop also provides online shopping services, in which the customers can choose a variety of goods by browsing from the pictures posted at the boutique's Facebook page, placing orders through private messages or by ordering through telephone calls, and then proceed to make payment through Paypal or through bank transfer, and have the desired goods delivered right to their doorstep.

There are currently three employees hired to maintain daily business operations, such as operating as cashier, tending to the customers, cleaning the storefront, arranging shelf items, managing stock records and check through the newly arrived items for defects. They will arrive at the shop about half an hour earlier before the opening hours to clean up the shop and prepare to open the doors.

Whenever a batch of new stock item arrives, the employees will also be helping the owner to go through all the bags (most of the time, the stocks will be arriving packed in huge plastic bags), and then tagging and labeling each of them, recording them in the stock list, then fold it or hang it on the shelves and racks. If there is a defective item, they will send a return outwards request to the supplier and ask for replacements.

The store also occasionally holds sales events in conjunction with certain festivals or celebrations, which include free items with purchases, discounts, vouchers or various benefits.

Current methods used to manage the operations

The current stock system used by Sense Boutique is a manual system that requires employees to keep track of stock levels, record stocktake and sale transactions, as well as manage new stocks delivered.

The data are stored in a paper spreadsheet, which the stock inventory data were manually recorded by hand. The paper documents are then filed in alphabetical order and according to categories, and then stored in a central cabinet in the office, which is locked using a conventional padlock.

However, there are no other security measures in place, which makes the data vulnerable to lockpicking. There is also no redundancy for the data (backup), hence the papers can easily be lost or damaged.

Whenever the employees need to access the record for stock check, for catalog enquiries, or end of day balance sheets, they would need to unlock the cabinet and search for the appropriate record by the corresponding categories or first alphabets. Also, when new stocks arrive, the employees would go through all the new stocks and record each of their serial numbers/IDs, prices and costs one by one onto respective spreadsheets.

Records of suppliers are also keep tracked in a similar folder that is stored at the same cabinet as well, with their contact information as well as records of stocks imported from them. Whenever the owner wants to reorder new stocks, she would need to search for the particular supplier's info through a stack of papers to obtain the phone number or other contact methods.

At the end of the month, the owner would also perform a monthly stock check, compare the amount of stocks on hand to the data recorded on the spreadsheet manually, and also calculate gross profit as well as other balance sheet items.

Weaknesses of current system

There are quite a few weaknesses that can be noticed from the current implementation of stock control:

1. Some effort required to update the records
   The current method of keeping inventory records requires the employees to juggle a huge amount of paper work and manually update them, which is time consuming, slow and tedious. Productivity may also fall due to boredom or human fatigue.

2. Unorganized data
   Although there is a basic categorizing filing system in place, most of the items are still largely unorganized. Inability to sort the records also made it harder for the manager to make decisions regarding inventory management. The amounts of records to be kept are also too large to be handled efficiently by only 3 employees, hence there may be errors or omissions, as well as outdated records remaining in the documents.

3. Poor backup, restore and maintenance system.
   The current system doesn't employ a backup methodology, which means that all data can be lost in case of accident, theft, arson or other natural disasters. The paper documents are also prone to normal daily wear and tear damages as well as water spillage.

4. Doesn't validate data entered
   The current recording method doesn't validate and validate the data entered; hence human error is possible in the records, such as erroneous entry and omissions of certain details. This may makes the documents inaccurate and unreliable.

5. Redundancy of Records
   The current system does not check for replicated entries (as it's entered manually by hand), hence there may be repeated entry for certain records, hence making stock levels inaccurate and may affect stocktaking decisions or wrong inventory information.

6. Insecure records
   The current cabinet used to store the paperwork is just secured only by a padlock, which all employees have the key to access to. That means everyone who has the key will be able to access the cabinet and read the records inside, and owner cannot choose to restrict who can view what records.

Description of the Origin of Data

Whenever a new product comes in, the employee would have to record the following info onto a form/spreadsheet that includes the following information:

Product:

- Product Name
- ProductID
- Cost
- Recommended Resale Price
- Quantity (Stock Level)
- Description
- Supplier
- Category
- Color
- Size
- Gender Suitability
- Date

If the owner has found a new supplier to conduct business with, she would record the particulars of the supplier in a separate form that includes the following information:

Supplier:

- Name
- IC Number
- Telephone Number
- Email Address
- Address

Whenever there is a sale that took place, the employee/cashier would write a receipt for the customer, at a same time, producing a copy for administration purposes:

Sale Transaction (Receipt):

- Product ID
- Quantity
- Discount Provided
- Final Selling Price
- Payment Method

Sense Boutique Inventory List

FOR CATEGORY: ___________________________

Start Date:	
End Date:	

Product ID	Produce Name/Description	Unit Cost	RRP	Quantity	Color	Size	Gender	Supplier ID/Name	Date Added
	Total								

Sense Boutique Enterprise

No. 32, Jalan USJ 10/1B, Taipan Business Centre, 47620 UEP Subang Jaya, Selangor Darul Ehsan, Malaysia.

TEL: 03-5636 5600 FB: https://www.facebook.com/SenseBoutique/

Supplier Form

Date: _____

Name: _____

IC Number:_____

Telephone:_____

Email:_____

Address:_____

Items from this supplier:

ID	NAME	QUANTITY	DISCOUNT	COST
			Subtotal	
			TOTAL	

Sense

Sense Boutique Enterprise

No. 32, Jalan USJ 10/1B, Taipan Business Centre, 47620 UEP Subang Jaya, Selangor Darul Ehsan, Malaysia.

TEL: 03-5636 5600 FB: https://www.facebook.com/SenseBoutique/

Date: _____

Invoice No.: _____

To:_____

Payment Method

:_____

ID	NAME/DESCRIPTION	QUANTITY	DISCOUNT	FINAL PRICE
			Subtotal	
			TOTAL	

For use by office only:

_____

Investigation Method

To investigate about the problems and possible solutions that can be undertaken to solve the client's problem, there are several methods that can be taken:

1. Face to face Interview

- The system developer will perform an interview session with the client or system administrator in the business. The system analyst is a systems development expert, however he doesn't know the company well enough to develop a system that can fits the client's needs. Hence, the system analyst will need to gain an understanding about the inner operations of the business by asking for further clarification of details from those who are the experts in how the company works, like the system admin or even the business owner. Through these communications, it can be made sure that the client's requirement and the analyst's understand matches each other.

2. Observation

- The system developer will stay around during business hours to observe how the business operates, in order to obtain firsthand knowledge on the daily operation of the company. The developer must take note of the major processes on how the business is run and then develop a suitable solution for these processes. However, this method depends on the understanding of the system analyst in order to be effective. Hence, it is best that this method be used alongside interviews to get the best of both worlds.

3. Research and Reference

- The system analyst will perform research based on past solutions that are employed on other similar typed or sized businesses from various data sources, and then devise a suitable solution for the business. This method allows for precise information and facts, but it can be time consuming.

4. Questionnaire

- Questionnaire is a method where the analyst devises a fixed set of questions and distributes it to a large sample of people to obtain results/opinions, then the results are analysed to derive a meaningful conclusion. However, it can be inaccurate if the participants doesn't take the questionnaire seriously or give biased answers due to demand characteristics.
- Due to the small scale of this business, this method is not suitable to be used in this project.

Interview

Before the initial proposal and design of the system can take place, an interview has to be conducted in order to communicate the client's requirement and developer's understanding better, to make sure the solution devised matches the client's needs.

By interviewing, it is possible to ask open ended questions that can lead to further clarification of how the business operates and provides more details on how should the system be developed. It is important to obtain the client's opinion and problems on the current implementation, which then I can use to develop a system that specifically targets as well as overcomes these problems and solve the objectives, in the end allowing a system that matches the client's expectation to be developed.

An interview was conducted with the owner of the Sense Boutique, Mrs Lim Ying Ying at her shop.

Draft of questions to be asked during the interview:

1. Background of the business
2. Nature of the business
3. Items sold in the shop
4. Handling of stock records
5. New records
6. Sale or transaction of items
7. Suppliers records
8. Dissatisfaction with the current system
9. System requirements
10. Extra requests

Interview Transcript:
Words in bold are by the system analyst (interviewer).
Words in normal formatting are by Mrs. Lim, owner of Sense Boutique.

Good afternoon Mrs. Lim, thank you for taking your time to conduct this interview with me. First and foremost, can you briefly clarify on the background of this business, like the history and what your business does?
Sense Boutique actually wasn't started by me, it was started around year 2007 by another woman; her husband supported her efforts and provided her capital to start her own business to sell clothing. Then as time went by, she gradually lost her interest in the fashion business and ventured into other businesses. That's when she decided to sell her shop to be taken over by another person. Around the end of 2011, I have decided to quit my desk job in the retail industry and take over her shop in order to pursue my lifelong

dream in starting my own clothing business. After some cleaning up and renovation, I reopened the shop in early 2012 with the same name. The business has been going pretty well for me since then, and I have hired a few more employees to help me tend the shop.

That was quite an amazing achievement for you to quit your stable job and venture into business. Okay, so can you elaborate more on the clothing items that you do sell in your store?
We sell various fashion items for both genders including T-shirts, dresses, pants, shoes and various accessories such as belts, sunglasses, braces, fashion necklaces, couple rings as well as handbags, and more. They are sourced from various dealers across Malaysia and Klang Valley, some items in the catalogue are handpicked by me from one shop to another shop, and some were even imported from countries such as Singapore, China and UK through the help of my friends there. I often look into fashion magazines for newest fashion trends so that I can update my catalogue as soon as possible.

Okay, so how exactly do you handle and store your stock records?
We store all the records into respective spreadsheets according to the categories they are in, and then sort them in alphabetical order into a file. Then we store the file inside a locked cabinet when it is not in use.

What do you normally do when a new item comes in?
When a new item is purchased and added to our inventory, we record down the particular product ID, name, cost, the recommended resale price based on our set profit margin, supplier, color, category, size as well as incoming date onto our inventory spreadsheet. Then my employees will sort the spreadsheet lists into the file in the cabinet.

What about when there is a sale?
When there is a sale transaction, we will write a receipt for the customer, while at the same time, producing a carbon copy of the receipt for our record and administration purposes. Things noted in the receipt include payment methods, date of sale, items, quantity and price.

How about the suppliers? How do you record them?
The suppliers are stored inside a separate form which records their particulars including name, contact information and address, as well as a brief list of items that we sourced from them. Then again, the forms are filed in alphabetical order and sorted into the file and placed inside the locked cabinet.

So what is it that you are not satisfied with the current implementation of your filing system? Do you think that the current system cannot satisfy the future demands of the expanding business?
There are quite a few shortfalls to this current filing method which was inherited from the previous owner. For example, it gets really troublesome to record each items manually, and then updating it again if there are any inflow or outflow of stocks. It was tedious, inefficient and takes up quite some effort to be able to carry it out. Especially during peak seasons such as Chinese New Year when there is a large influx of new customers seeking new clothing items and when I bring in a lot of new stocks, the current system just doesn't cope well, it takes up too much effort to sort through the items and record them manually. Once it took up so much time, that my employees and I were demotivated to continue recording the items and our productivity falls sharply as a result.

Plus sometimes when we are tired from a long day's work or due to employee's negligence, some items may be recorded incorrectly, causing even more problems. Once, one of my employees forgot to record a sold item, causing everyone to look around for the missing stock, only to find out from the receipts that it has been sold but hasn't been recorded yet.

So are there any exact requirements that you expect the new system to be able to do?
First of all, I would expect it to be able to show me all my inventory records at a glance, and allow me to organise and modify each record with ease. It should also allow me to search through all items easily. Retrieval of data should also be fast so to keep operations smooth and efficient. Adding and editing records should be done easily as well. Then it should also allow me to record sales or new stock takes easily.

Aside from that, it should also allow me to keep track of all incoming and outgoing stocks so that I can monitor sales performances. The system should also be able to print out a hardcopy of these lists for my perusal.

Because there are over a thousand varieties of clothing items in this shop, the system should also be able to handle a large amount of records, and allow me to keep check of them easily. The system should also be able to calculate some simple statistics like gross profit for me. The system should also be able to check my input for simple errors.

It should also store my supplier's records, and allow me to manage my list of suppliers easily.

Are there any extra features that you would like to have, such as security and encryption?
Well, for a start, yes, I would prefer that there is some kind of security measure that can safeguard my data from being accessed by anybody. Probably you may include some kind of password or login system? My current cabinet filing system seems too easily accessed by anybody. Even the key and lock may also be easily compromised by a skilled thief.

Aside from that, one thing I really worry about my stock records is that they are easily damaged. That day I nearly spilled my coffee on the stock list while I was updating the list, and that could've cost me so much time to try to recreate another list. So I would like some sort of backup system that I can use to make a copy of my data for safekeeping purposes. With the old system, it seems like it would be really easy to lose my papers, or got them damaged in accidents or disasters.

This marks the end of the interview. Thank you Mrs. Lim for allowing me your time!

Verified by,

_____
(Mrs. Lim Ying Ying)
Owner of Sense Boutique

Proposed System

As per the requirements set by the client, Mrs. Lim, the new computerised system should be coded using the Visual Basic programming language. It will be storing all inventory records in a Microsoft Access database in multiple relevant tables.

It will be able to show the user all inventory records at once through a list display, and the user will be able to search through the list easily and quickly on the fly. The user can also easily add, edit and remove records through a click of a button. There will also be separate buttons for recording a sale (outgoing) or purchases (incoming). It will also be able to store and display information of associated suppliers, and allow the user to add, edit or delete their information easily.

The new system will also be able to print a hardcopy or save the list of inventory records as well as the transaction journal report, for easy sharing of data, and to facilitate better decision making by the business owners.

The system will also be secured through the use of a password login system during startup. When the program loads, it will ask for a username and password in order to access its data. This helps keep unauthorised access to the data at bay. It is also achieved through implementation of different user groups such as administrator and normal users, with certain features, such as viewing transaction journal report and changing user passwords restricted to administrators only. The administrator will be able to add or remove users, change passwords as well as changing user types through a central user management console. This allows for future expansion of the business when more employees are hired or other scenarios.

There will also be a backup and restore system built in. It will allow the user to easily one click backup to any location, and then restores it automatically in event of any database damage or problems.

Aside from that, it will be easy to use and user friendly. Relevant help documentations, simple user manual and hints will be included as well.

Summary of features:

1. Inventory listing at a glance
2. One click adding, editing, deleting of records
3. One click record of sales and purchases
4. Fast searching of the database
5. Printing and exporting of the list of stock items

6. Recording suppliers
7. Managing categories
8. Transaction report journals
9. Users accounts login system with passwords
10. Backup/Restore system
11. Help documentation

Proposal agreed by,

_____
(Mrs. Lim Ying Ying)
Owner of Sense Boutique

Requirement Specification

Input Requirement

1. Security measure through password logins
   - During login, the user inputs a User ID and password in order to login.

2. 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.

3. New purchases incoming
   - To add incoming new purchases, the user would have to input the quantity bought.

4. Sale of items outgoing
   - To add a new sales transaction, the user would have to input quantity sold and selling price.

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

6. Manage categories types
   - To add a new category, the user inputs a new category name.

7. 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.
2. Calculate gross profit for each sale transaction
3. Calculate the new stock level after purchases
4. Storing of supplier details
5. Backup and Restore of database
6. 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.

2. Export of Inventory Items to Text file
   - A list of inventory items will be saved as plain text format.

3. Printing of Inventory List to paper
   - A list of inventory items will be printed in hardcopy format.

4. Export of transaction report to text file
   - A transaction report will be saved as plain text format.

5. 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.

Evaluate alternative solutions

Before any development of the new system is started, the system analyst has explored alternative solutions with the user in mind, in order to minimise cost and provide the greatest benefits. To deploy a system that matches the user's needs, there are a few methods that can be used.

- **Buying off the shelf software**
    - Refers to software that are readily made and can be purchased directly from a software vendor
    - Advantages
        - Less expensive
        - It is possible to speak to other users for reviews and evaluations before purchasing the software
        - Can be bought and installed straightaway
        - It is tried and tested, so may have less bugs
        - Well documented by its developers
        - Usually have general system requirements, hence can be used on a wide variety of systems
        - Can be found in most major software stores
    - Disadvantage
        - May not fit into the client's needs
        - Features are most of the time more general and less specific

- **Tailor made/custom made software**
    - Refers to software that are custom made based on the scope that is specified by the user
    - Advantages
        - Designed to do exactly what the user requires
        - Can be written to run on specific hardware
        - Can be written to integrate with existing programs
        - The perfect solution when there is not any suitable commercial programs available outside
        - Errors can be easily fixed, as the developer knows the inner workings of the program the most
        - Developer may be able to provide training and documentation
    - Disadvantages
        - May have a lot of bugs or unstable since it is not tested on a wide range of specifications or usages
        - Costliest option out of all three
        - Requires a lot of time and effort to develop a suitable solution.

- **Deployment of Integrated Application packages**
    - Refers to software that comes in a package and are integrated to work with each other.
    - E.g. Microsoft Office, Adobe Creative Suite
    - Advantages
        - Offers multiple products packaged together which can be cheaper than buying separately
        - Data are compatible between each application in the package
        - Menu design are consistent among the applications, hence easy to use and less confusion
        - Popularly used, hence can ask for reviews and opinions
        - Should have less bugs and more stable since they are well tested in various environments
    - Disadvantages
        - The package may come with extra unnecessary programs

Final Verdict

Through careful observation and evaluation of various options, a final decision has been made based on various considerations, that it is best to hire a system analyst to develop a custom database management system for the company.

By developing a custom made program, it can be tailor made to suit the company's exact needs and provides various other advantages as well. In a business environment, it is essential that programs chosen to be used in a system enjoys maximum stability and is able to work correctly, in order to prevent any data loss or financial losses, minimise risk and maximise resources usage. Due to the specific needs of the user, it is hard to find a suitable software package that does everything required by the user; hence a custom made solution is viable. On the other hand, it is believed that in the long run, a custom made solution will be cheaper compared to buying off the shelf programs.

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.

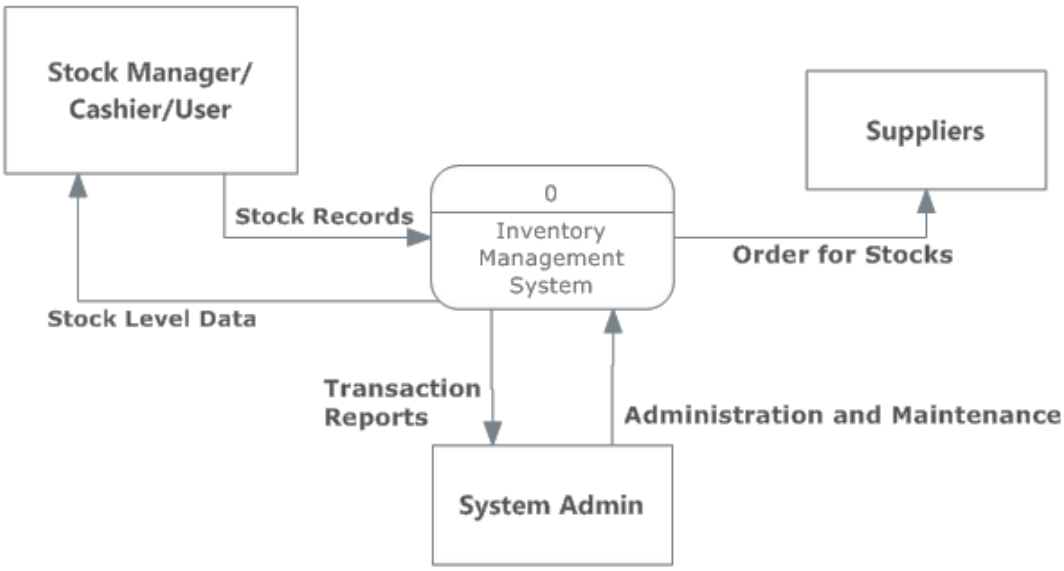
- To reduce costs and minimize risks

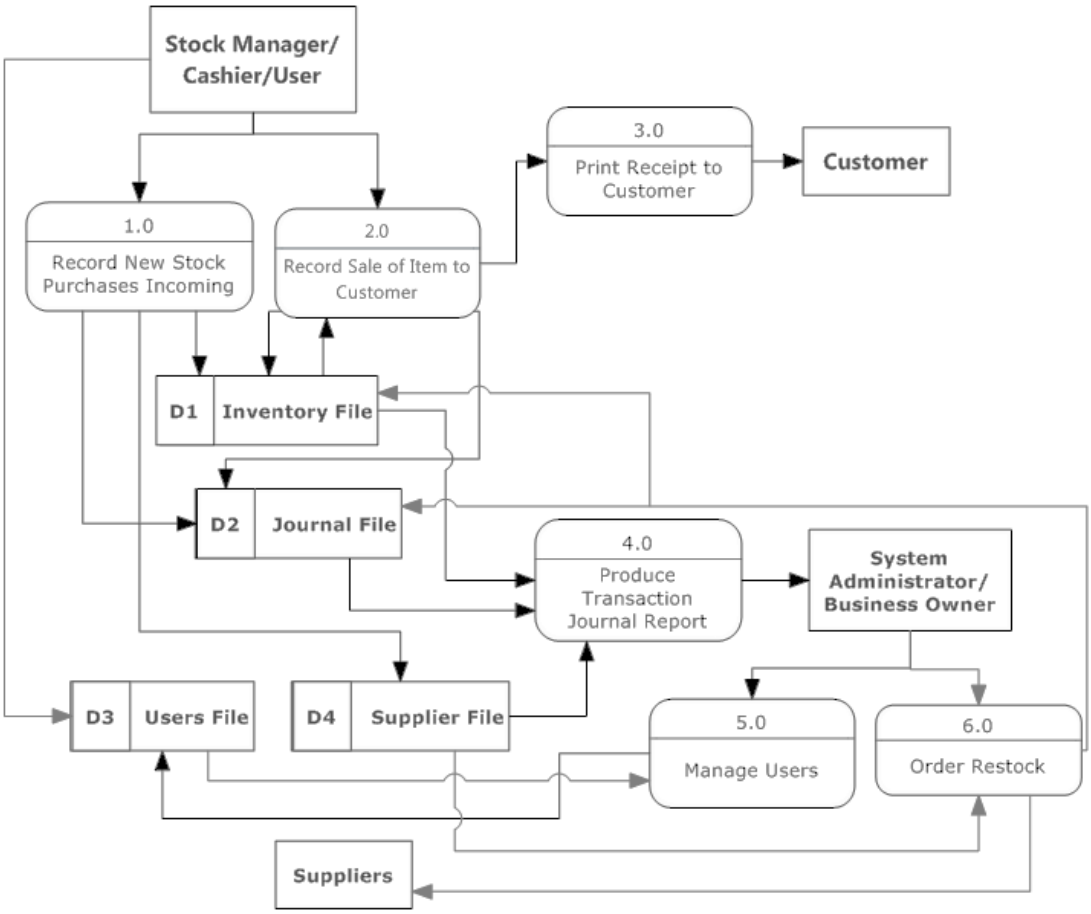
By making the business operations more efficient, it is possible to reduce the cost of operations of business in the long run. Higher productivity can be translated to higher sales, hence higher profits as well. On the other hand, employment of better security measures as well as better backup/restore 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.

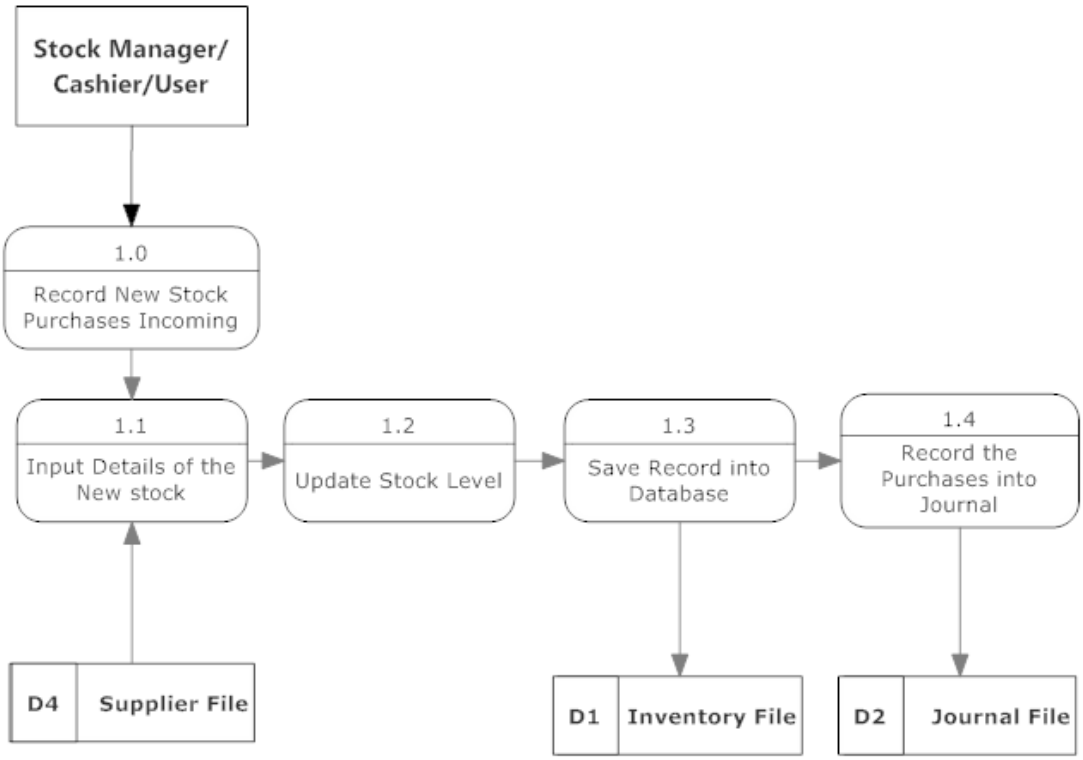
- To aid in making better business decisions

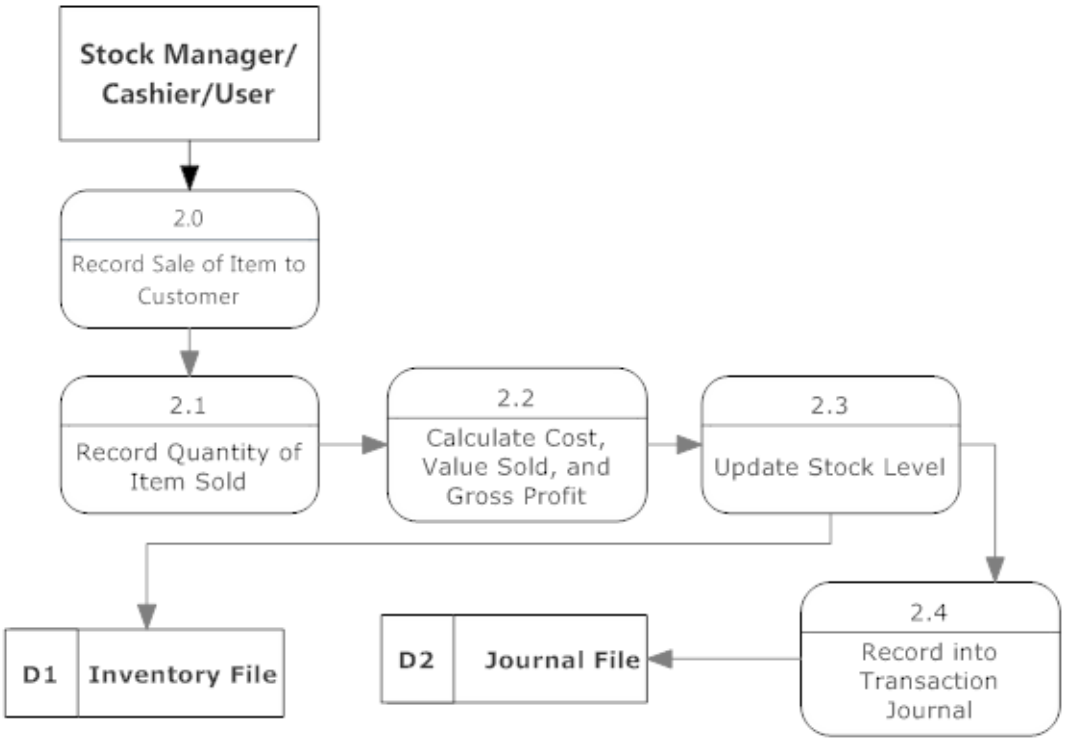
Through the new computerized system, it is able to automatically generate transaction reports and calculate total costs as well as gross profits. This allows for more efficient management of the accounts and balance sheet. Hence, the business owner is able to easily make business decisions such as whether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

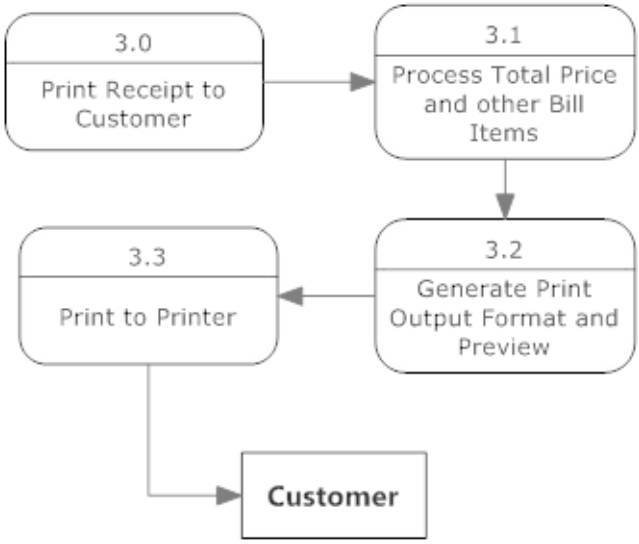
Context Diagram

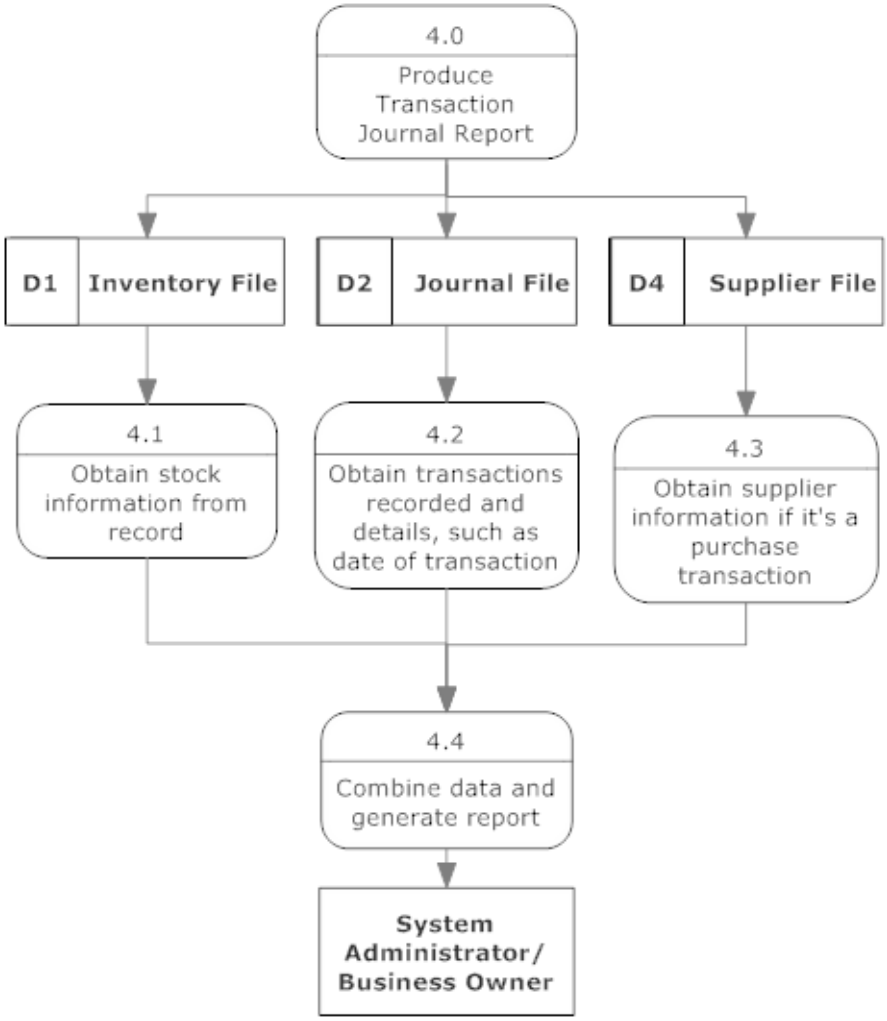


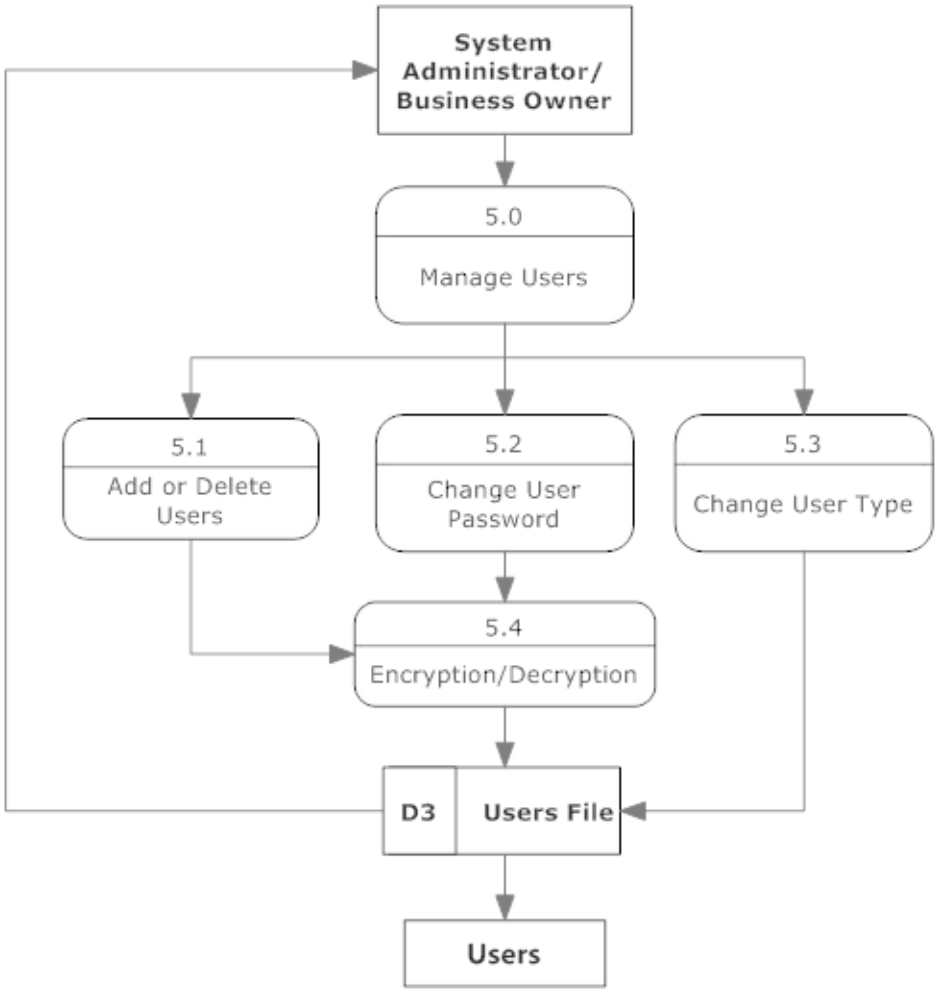


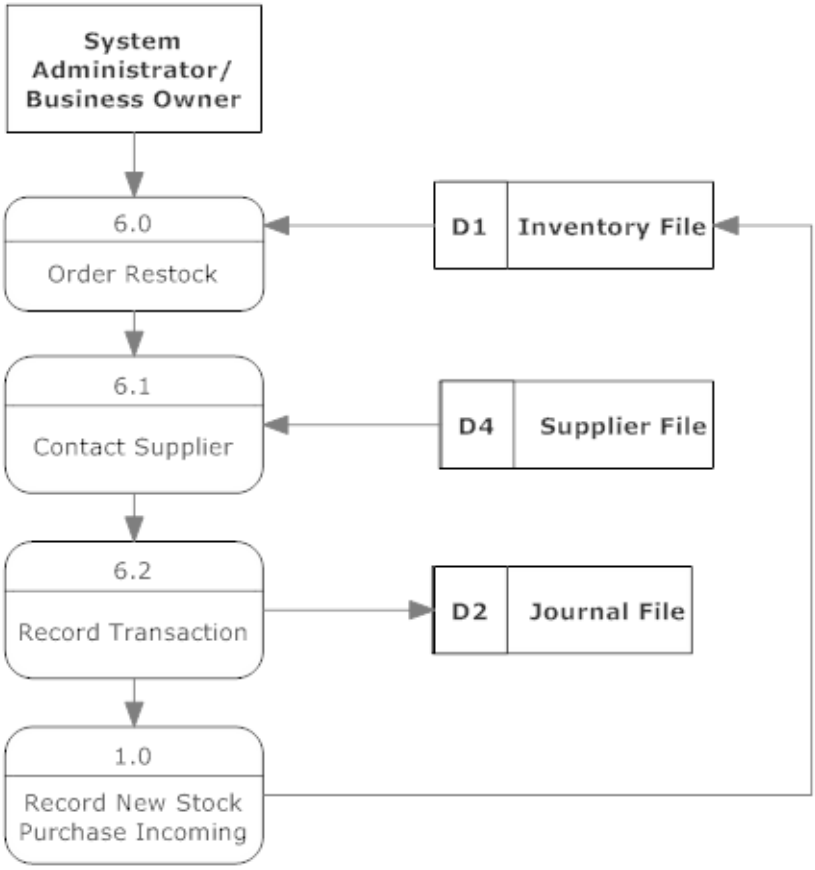

Stock Manager/ Cashier/User
1.0
Record New Stock Purchases Incoming
1.1
Input Details of the New stock
1.2
Update Stock Level
1.3
Save Record into Database
1.4
Record the Purchases into Journal
D4
Supplier File
D1
Inventory File
D2
Journal File


Stock Manager/ Cashier/User
2.0
Record Sale of Item to Customer
2.1
Record Quantity of Item Sold
2.2
Calculate Cost, Value Sold, and Gross Profit
2.3
Update Stock Level
2.4
Record into Transaction Journal
D1
Inventory File
D2
Journal File

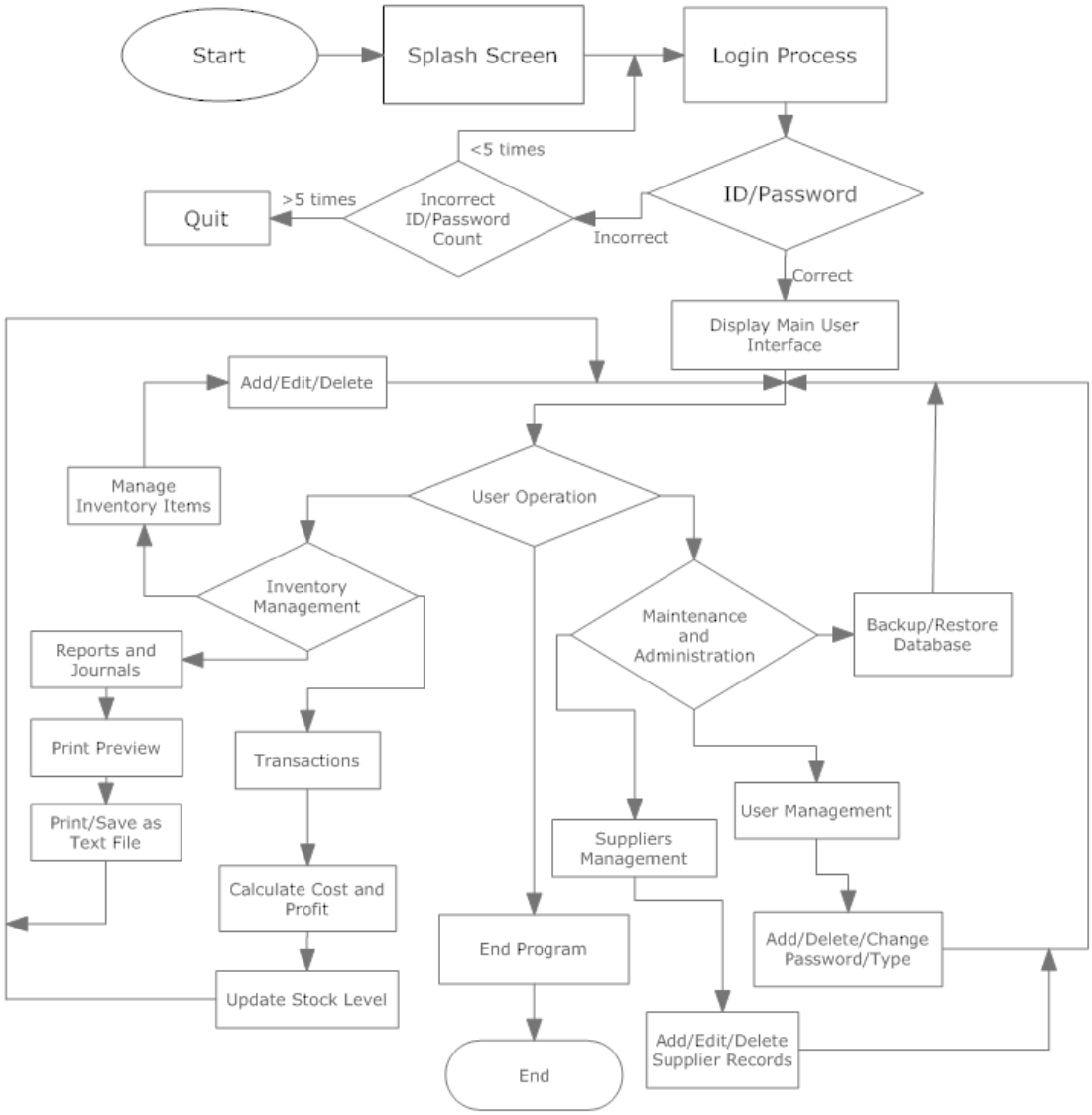

3.0
Print Receipt to Customer
3.1
Process Total Price and other Bill Items
3.2
Generate Print Output Format and Preview
3.3
Print to Printer
Customer


4.0
Produce Transaction Journal Report
D1
Inventory File
D2
Journal File
D4
Supplier File
4.1
Obtain stock information from record
4.2
Obtain transactions recorded and details, such as date of transaction
4.3
Obtain supplier information if it's a purchase transaction
4.4
Combine data and generate report
System Administrator/ Business Owner


System Administrator/ Business Owner
5.0
Manage Users
5.1
Add or Delete Users
5.2
Change User Password
5.3
Change User Type
5.4
Encryption/Decryption
D3
Users File
Users

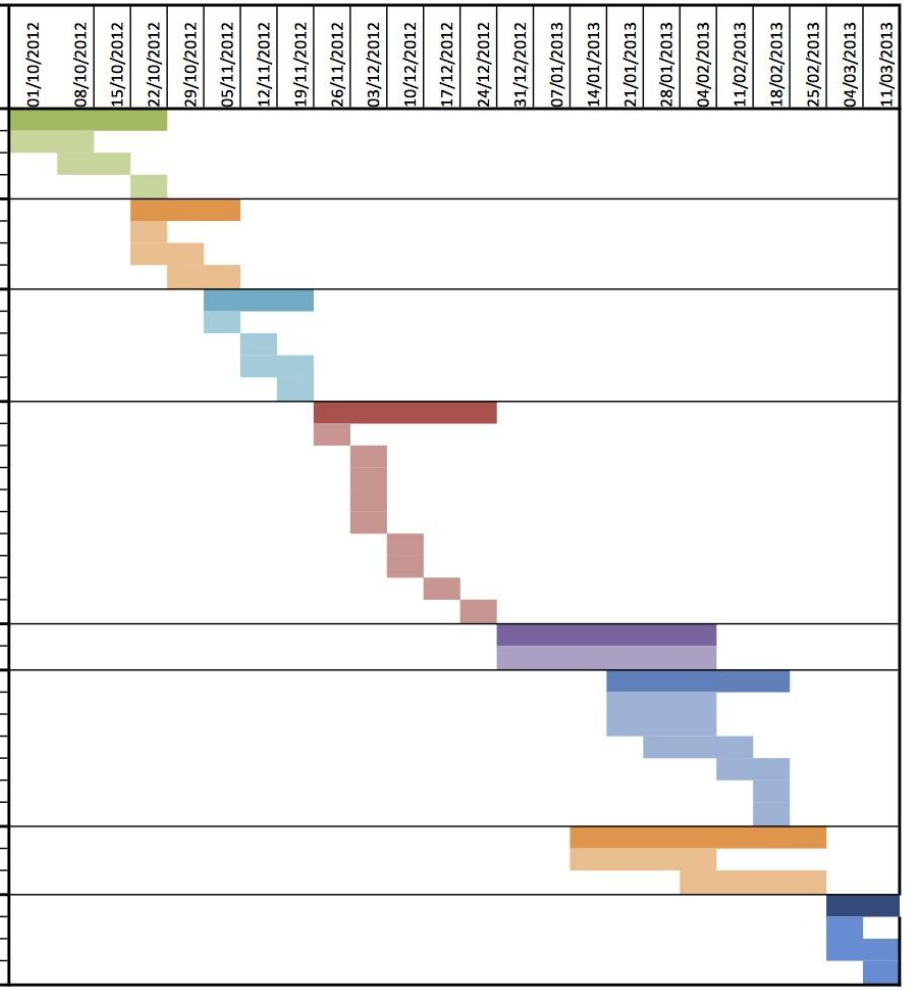

System Administrator/ Business Owner
6.0
Order Restock
D1
Inventory File
6.1
Contact Supplier
D4
Supplier File
6.2
Record Transaction
D2
Journal File
1.0
Record New Stock Purchase Incoming

System Flowchart

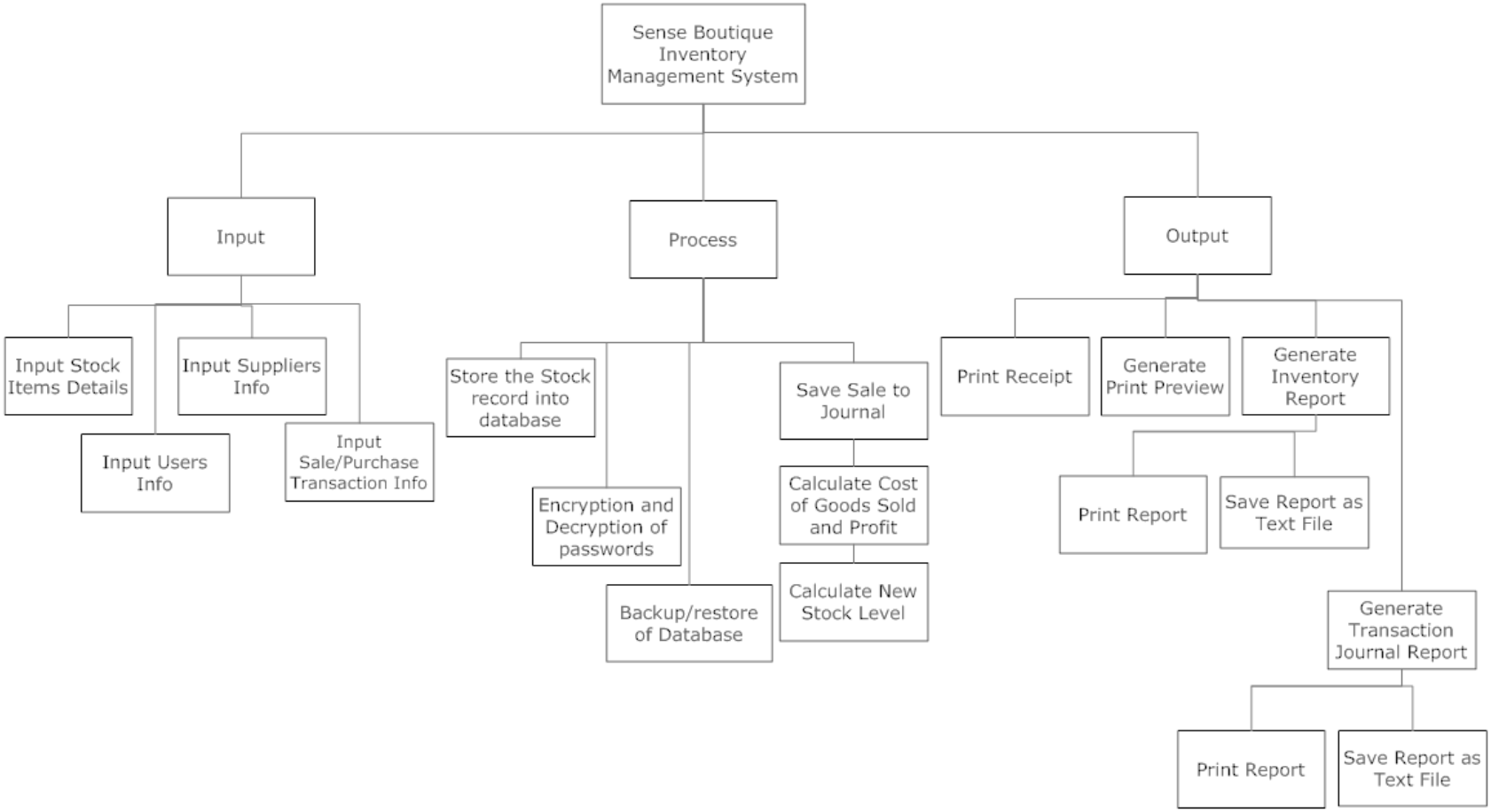


Gantt chart

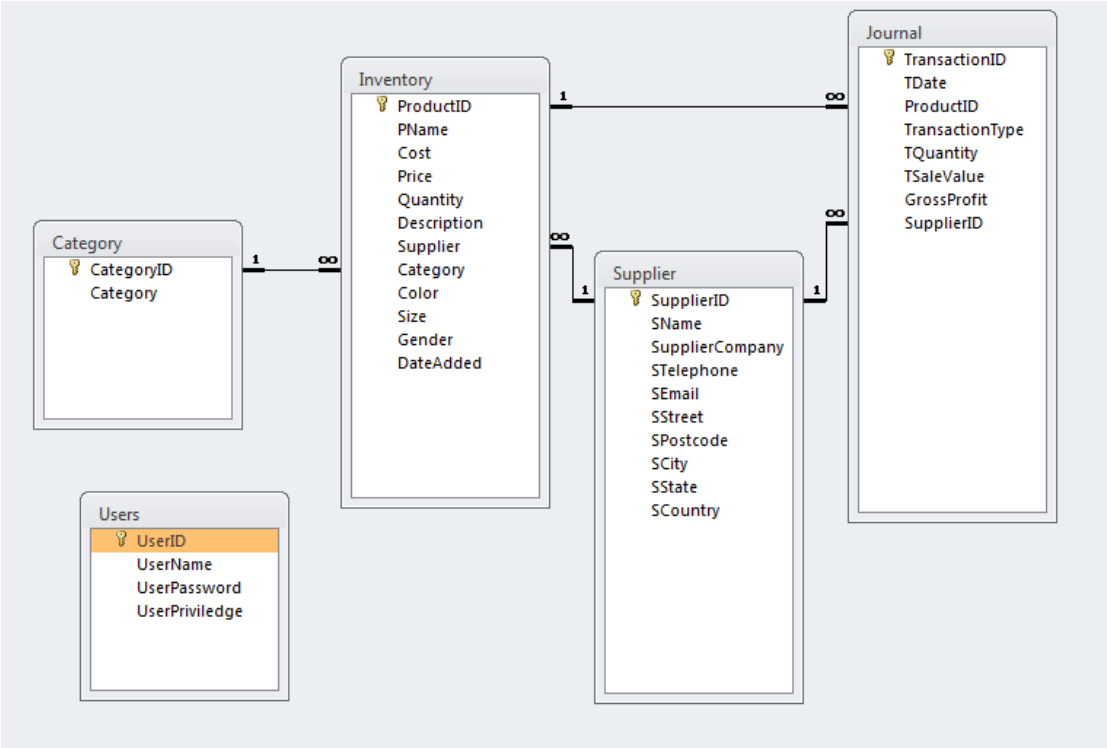
	Start	End
Definition	**10/1/12**	**10/28/12**
Background and investigation of current methods	10/1/12	10/14/12
Identifying weaknesses of current method	10/10/12	10/15/12
Analysing origin of Data	10/16/12	10/28/12
Investigation	**10/22/12**	**11/5/12**
Interview with Client	10/22/12	10/24/12
Observation and Research	10/23/12	11/2/12
Planning Client's Requirement	10/29/12	11/5/12
Analysis	**11/5/12**	**11/25/12**
Proposal of the New System	11/5/12	11/12/12
Requirement Specification	11/12/12	11/18/12
Software and Hardware Specification	11/14/12	11/20/12
Evaluation of Alternative Solution	11/19/12	11/24/12
Design	**11/26/12**	**12/30/12**
Aim and Objectives	11/26/12	12/2/12
Data Flow Diagram and System Flowchart	12/3/12	12/8/12
Context Diagram and Plans	12/3/12	12/8/12
Gantt Chart	12/3/12	12/9/12
Entity Relationship Diagram (ERD)	12/3/12	12/9/12
File, Records and Database Specification	12/10/12	12/15/12
Estimation of File Size	12/15/12	12/16/12
Design Input, Output and other Forms	12/17/12	12/21/12
Evaluating Benefits and Weaknesses	12/25/12	12/30/12
Software development	**12/31/12**	**2/10/13**
Development and Programming	12/31/12	2/10/13
Test and Installation	**1/21/13**	**2/24/13**
Test Plan & Deciding on tests to be carried out	1/21/13	2/10/13
Test Run	1/23/13	2/11/13
Installation and Implementation Plan	1/28/13	2/16/13
System Changeover and Implementation	2/11/13	2/22/13
Employee's Training	2/18/13	2/20/13
File and Data Conversion	2/20/13	2/24/13
Documentation	**1/14/13**	**2/27/13**
System Maintenance and Technical Documents	1/14/13	2/8/13
User Manual and Documentation	2/9/13	2/28/13
Evaluation	**3/4/13**	**3/11/13**
Evaluation of Objectives	3/4/13	3/5/13
Evaluating User Acceptance	3/6/13	3/8/13
Client and User's Response and Evaluation	3/8/13	3/11/13



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

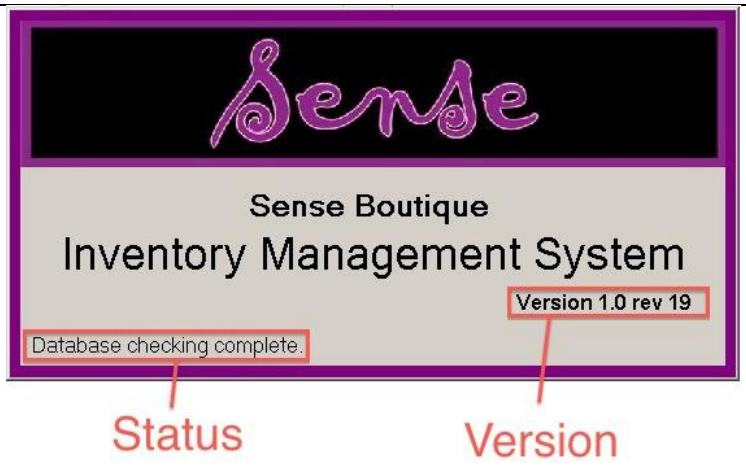
“Supplier” table

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

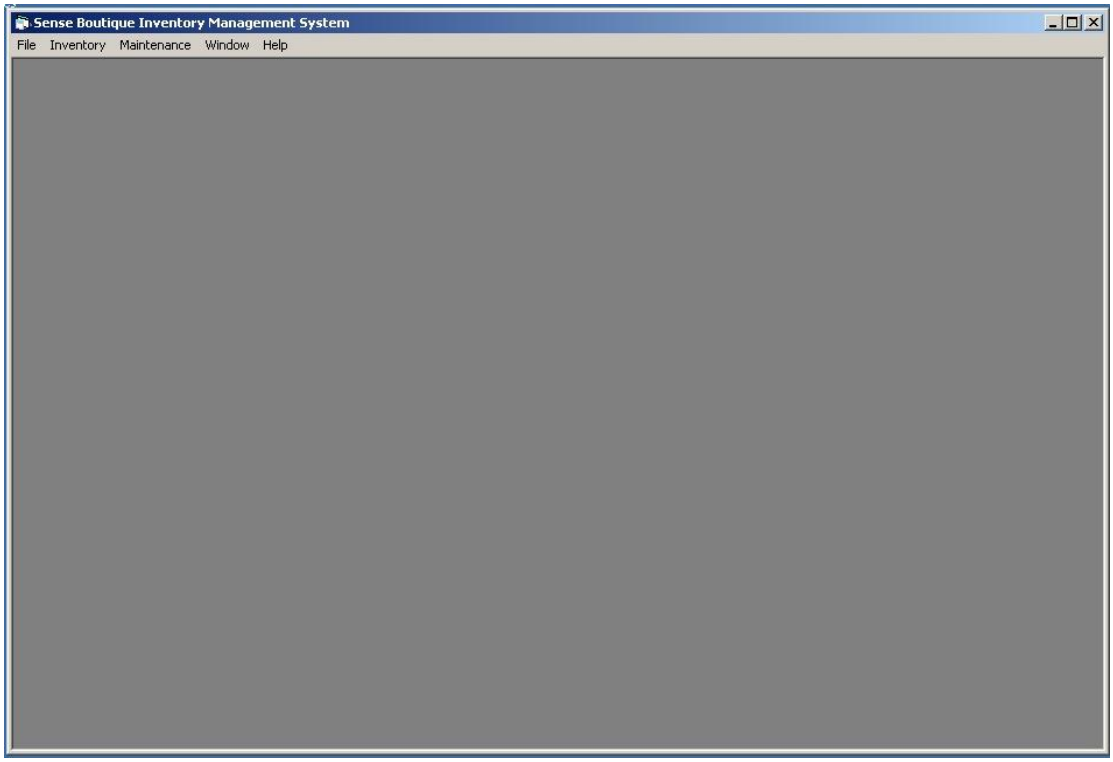
Button Controls

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



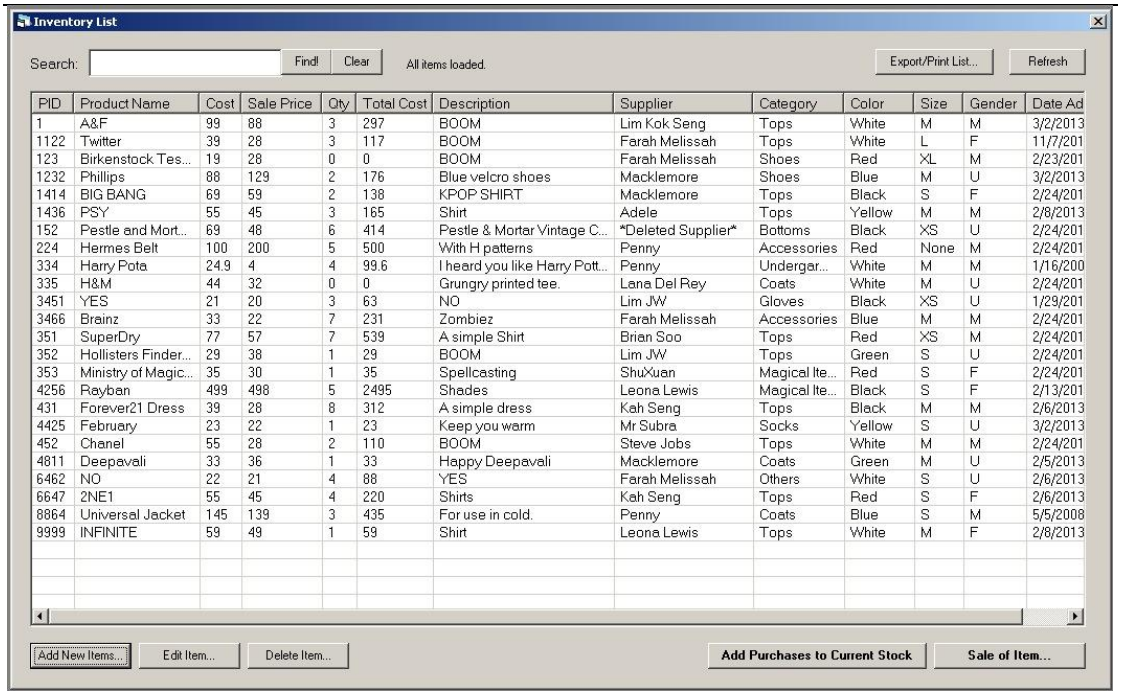
After the user entered a valid user ID and password, then successfully logged in, they will be shown this main screen with a menu bar on top. This main screen provides the user access to various features of the system and is an MDI (Multiple Document Interface) window that contains other 'child' windows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



PID	Product Name	Cost	Sale Price	Qty	Total Cost	Description	Supplier	Category	Color	Size	Gender	Date Ad
1	A&F	99	88	3	297	BOOM	Lim Kok Seng	Tops	White	M	M	3/2/2013
1122	Twitter	39	28	3	117	BOOM	Farah Melissah	Tops	White	L	F	11/7/201
123	Birkenstock Tes...	19	28	0	0	BOOM	Farah Melissah	Shoes	Red	XL	M	2/23/201
1232	Phillips	88	129	2	176	Blue velcro shoes	Macklemore	Shoes	Blue	M	U	3/2/2013
1414	BIG BANG	69	59	2	138	KPOP SHIRT	Macklemore	Tops	Black	S	F	2/24/201
1436	PSY	55	45	3	165	Shirt	Adele	Tops	Yellow	M	M	2/8/2013
152	Pestle and Mort...	69	48	6	414	Pestle & Mortar Vintage C...	*Deleted Supplier*	Bottoms	Black	XS	U	2/24/201
224	Hermes Belt	100	200	5	500	With H patterns	Penny	Accessories	Red	None	M	2/24/201
334	Harry Pota	24.9	4	4	99.6	I heard you like Harry Pott...	Penny	Undergar...	White	M	M	1/16/200
335	H&M	44	32	0	0	Grungry printed tee.	Lana Del Rey	Coats	White	M	U	2/24/201
3451	YES	21	20	3	63	NO	Lim JW	Gloves	Black	XS	U	1/29/201
3466	Brainz	33	22	7	231	Zombiez	Farah Melissah	Accessories	Blue	M	M	2/24/201
351	SuperDry	77	57	7	539	A simple Shirt	Brian Soo	Tops	Red	XS	M	2/24/201
352	Hollisters Finder...	29	38	1	29	BOOM	Lim JW	Tops	Green	S	U	2/24/201
353	Ministry of Magic...	35	30	1	35	Spellcasting	ShuXuan	Magical Ite...	Red	S	F	2/24/201
4256	Rayban	499	498	5	2495	Shades	Leona Lewis	Magical Ite...	Black	S	F	2/13/201
431	Forever21 Dress	39	28	8	312	A simple dress	Kah Seng	Tops	Black	M	M	2/6/2013
4425	February	23	22	1	23	Keep you warm	Mr Subra	Socks	Yellow	S	U	3/2/2013
452	Chanel	55	28	2	110	BOOM	Steve Jobs	Tops	White	M	M	2/24/201
4811	Deepavali	33	36	1	33	Happy Deepavali	Macklemore	Coats	Green	M	U	2/5/2013
6462	NO	22	21	4	88	YES	Farah Melissah	Others	White	S	U	2/6/2013
6647	2NE1	55	45	4	220	Shirts	Kah Seng	Tops	Red	S	F	2/6/2013
8864	Universal Jacket	145	139	3	435	For use in cold.	Penny	Coats	Blue	S	M	5/5/2008
9999	INFINITE	59	49	1	59	Shirt	Leona Lewis	Tops	White	M	F	2/8/2013

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

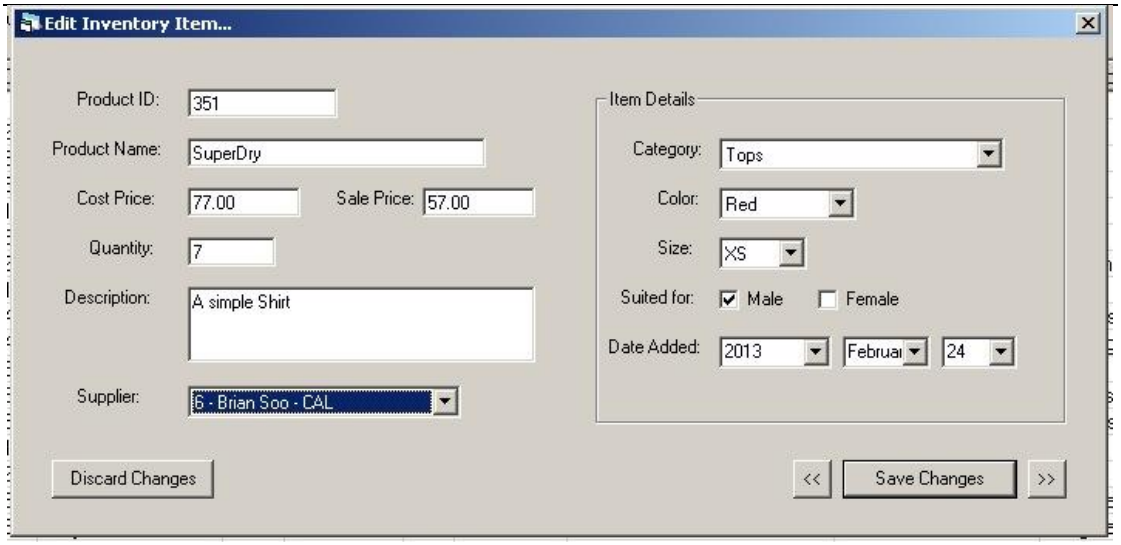
ComboBox Control

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

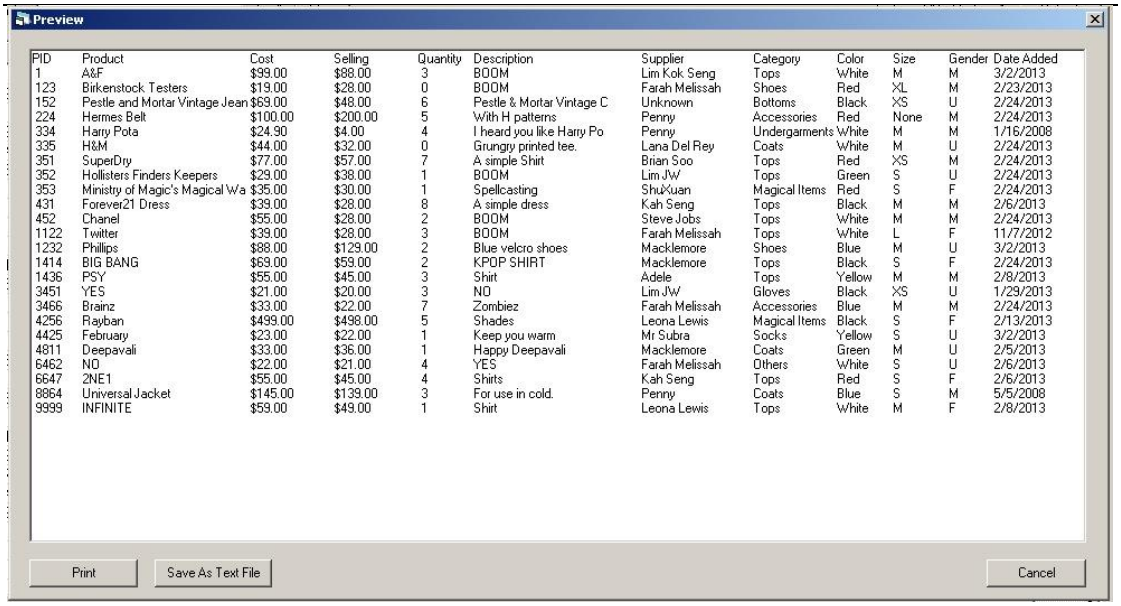
ComboBox Control

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.

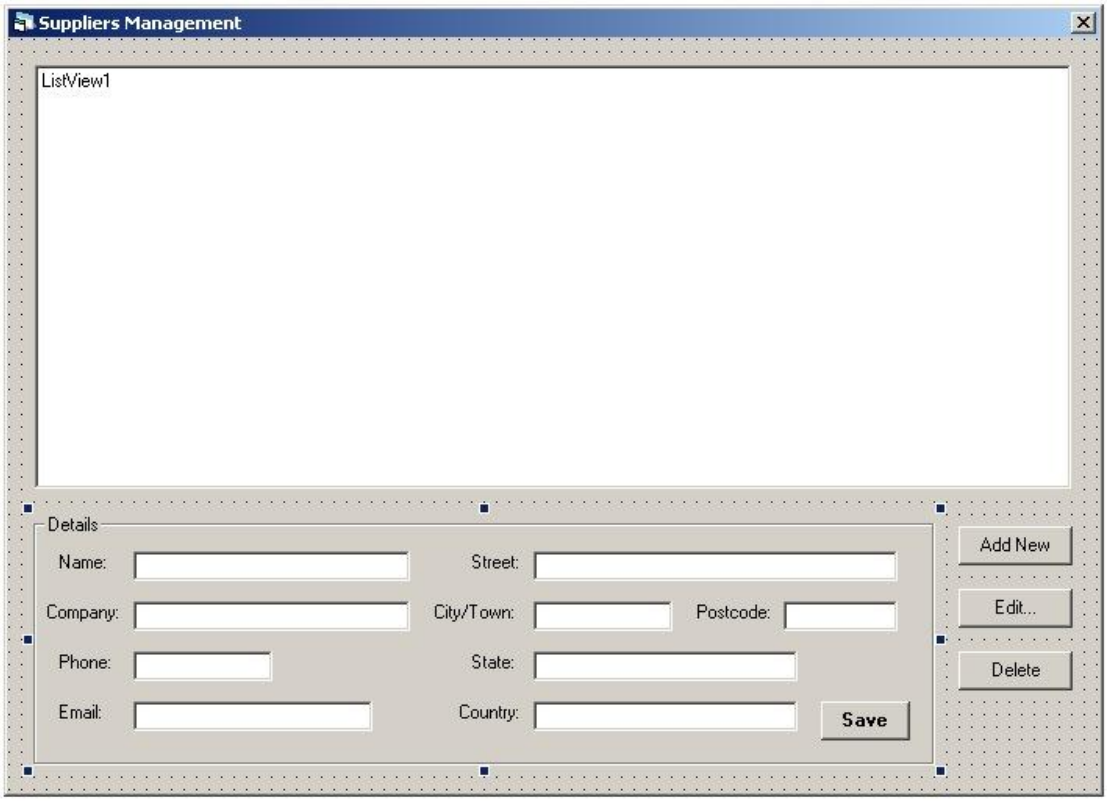
Buttons Control

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

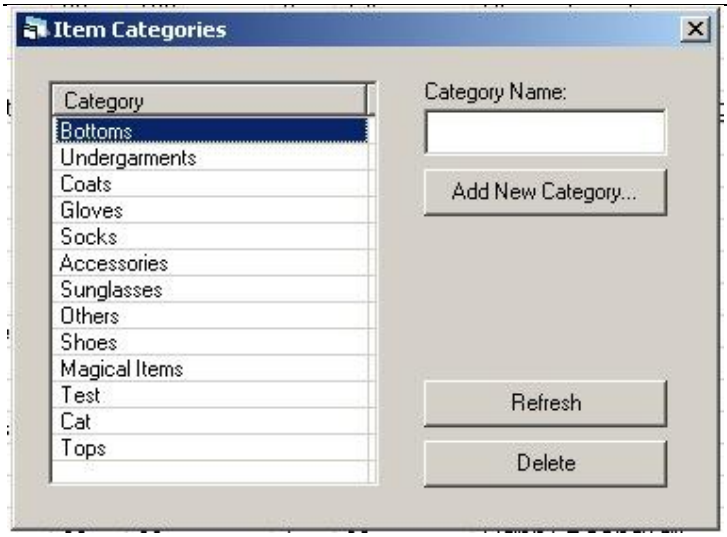
Control Name	Function of the Control
Add	To clear the textboxes and enable them to allow the user to add new

	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

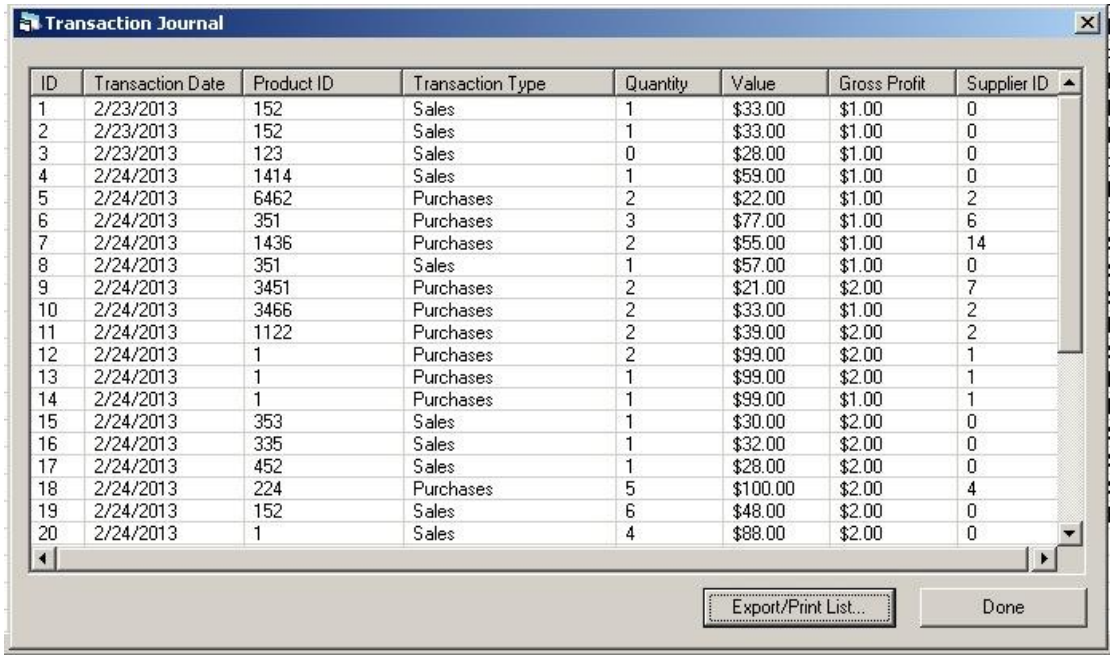
TextBox Control

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).

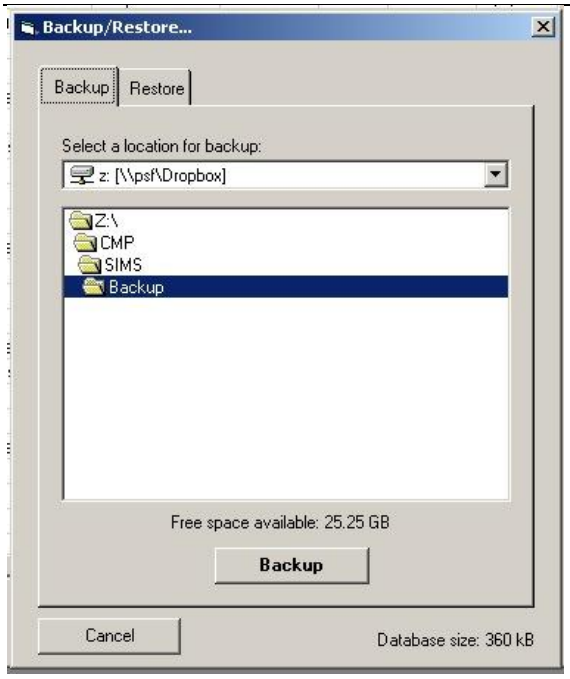
Textbox Control

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

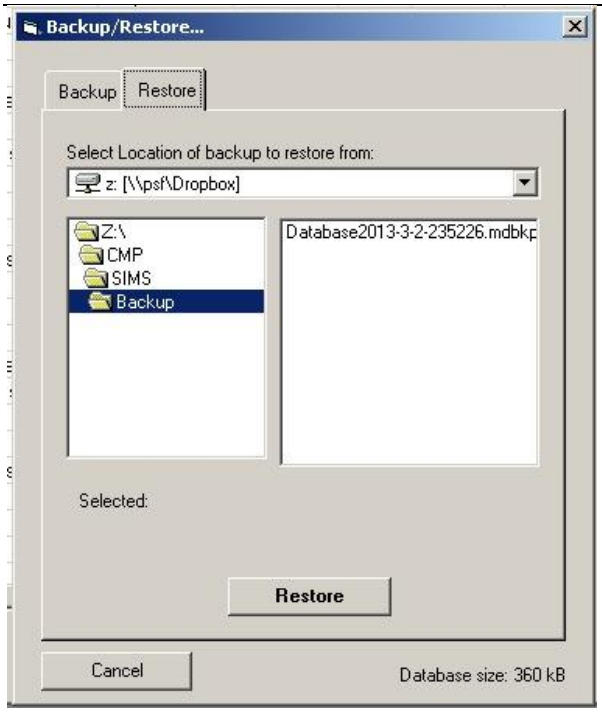
Labels Control

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

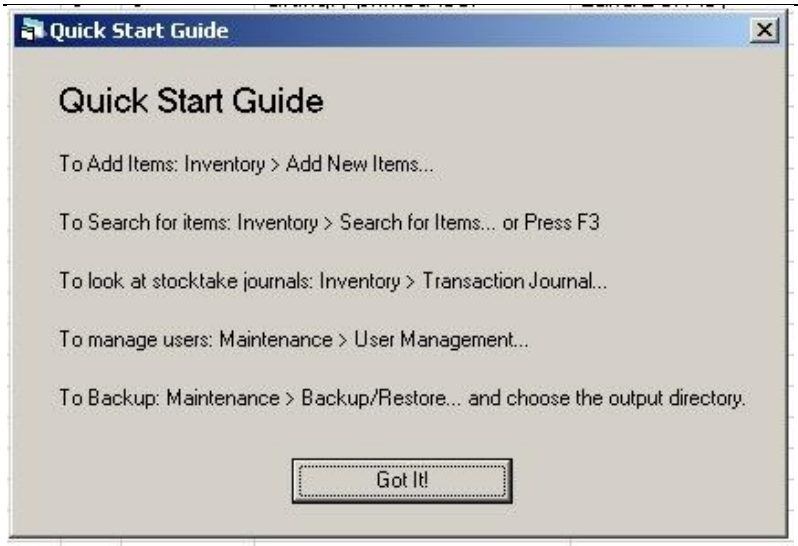
File Control

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.

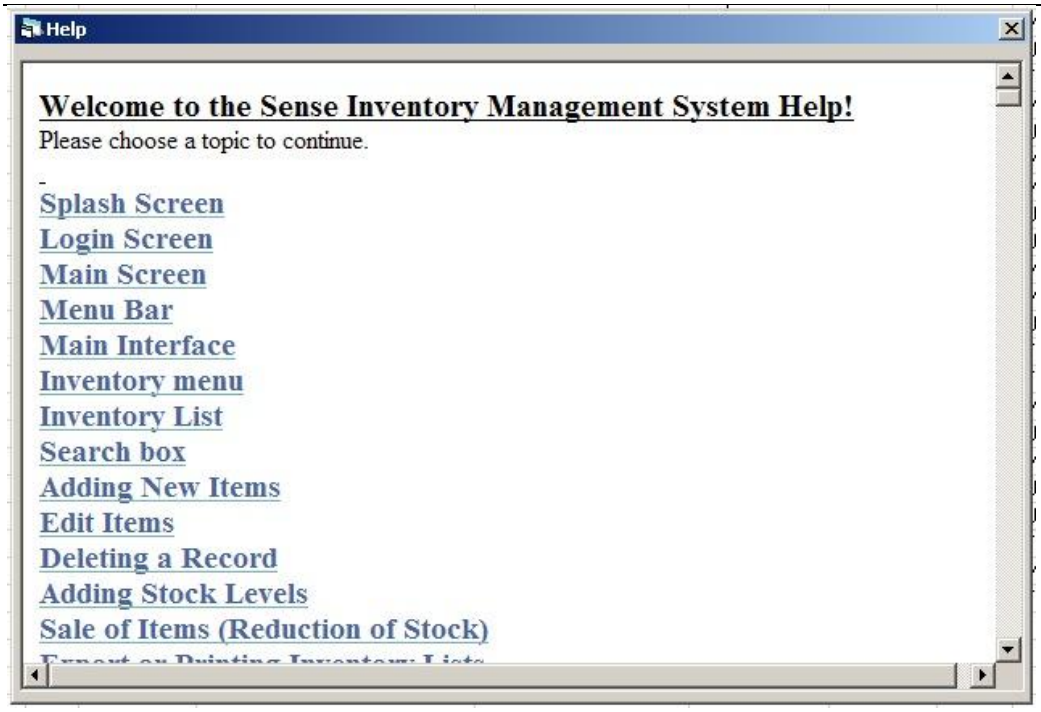
Quick Start Guide



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)

2. **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.

3. **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.

4. **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.

5. **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

A. 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.

B. 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.

C. 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.

D. 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

2. 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

3. 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

4. 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

5. 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)

C) Software Development, Programming, Testing and Installation

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 " & Ap
p.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. Woul
d 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 & "/D
atabase.mdb"
            MsgBox "Successfully created a new database!", vbOKOnly, "Restore
Complete"
            MsgBox "Default username: bin, password: 000", vbInformation, "Def
ault 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 with
out 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 & "\Dat
abase2.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.", vbOK
Only, "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 wil
l 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, "Canc
el")
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, "Confirmati
on")
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.Sele
ctedItem & " 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 Inve
ntory.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.Pro
ductID 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.Supp
lierID = " & 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 mis
sing 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 Inv
entory.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 an
other 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, "Er

ror")

   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.Tex
t & " 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 Inve
ntory.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 " & NewQu
antity & ".", 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?", vbOKCan
cel, "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 Inv
entory.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, DateAddedValu
e, "/", 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 Inventor
y.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, "Er
ror")
    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.P
roductID = " & 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, "Er
ror"
   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 & "/" & Date
Year
    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 no
t 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); "S
elling"; 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); FormatCurr
ency(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(15
5); 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(4
3); "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)); Ta
b(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); FormatCur
rency(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); "Se
lling"; 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.SupplierI
D = " & 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); FormatCurre
ncy(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); Ta
b(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 save
d 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"; T
ab(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(8
3); 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 + cdlOF
NOverwritePrompt
```

```
'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 = " & List
View1.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, "Erro
r")
   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 = " & List
View1.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 Suppli
er.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.SelectedIt
em & "? ", 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.Categ
ory = '" & 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 m
ore 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.AvailableSpa
ce / (1024 ^ 4), "###.00") & " TB"
    ElseIf Drive.AvailableSpace > 1000000000 Then
       FreeSpace.Caption = "Free space available: " & Format(Drive.AvailableSpa
ce / (1024 ^ 3), "###.00") & " GB"
    ElseIf Drive.AvailableSpace > 1000000 Then
       FreeSpace.Caption = "Free space available: " & Format(Drive.AvailableSpa
ce / (1024 ^ 2), "###.00") & " MB"
    ElseIf Drive.AvailableSpace > 1000 Then
       FreeSpace.Caption = "Free space available: " & Format(Drive.AvailableSpa
ce / (1024 ^ 1), "###.00") & " kB"
    Else
       FreeSpace.Caption = "Free space available: " & Drive.AvailableSpace & " b
ytes"
    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 & "\Dat
abase2.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 + v
bOKCancel, "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/BackupBefore
Restore.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.", vbI
nformation + 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:" & ListView
1.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 ch
aracters 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 character
s."
    Else
        If Len(PW) < 6 Then
            Status.Caption = "Please choose a password with longer than 6 charact
ers."
        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 anothe
r 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 le
ngth 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   'Chec
k 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 Sys
tem" '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
```

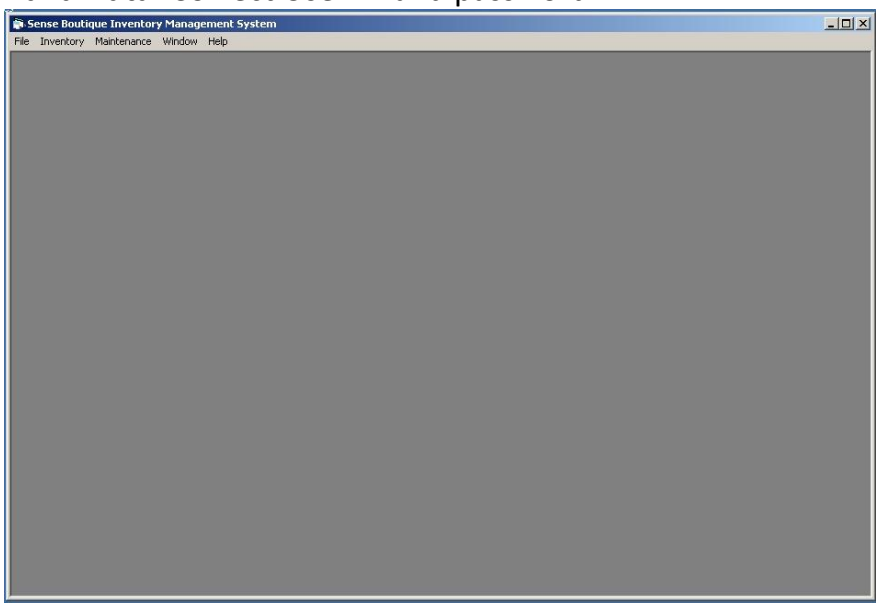
Test Run

Data Verification

Login Check

This test checks whether the login system verifies the user ID and password before allowing the user to access the system.

Valid Data: Correct User ID and password



Result: It will unload the login dialog and show the main window.

Invalid Data: Incorrect user ID or incorrect password

Result: A dialog box will appear to notify the user ot incorrect ID or password inputted, and the system is unable to log them in.

Confirm Password check

This test checks whether the password inputted is correct as intended by comparing the password field with the confirm password field.

Valid Data: Matching input in both Password and Confirm Password field

Result: It will notify the user of successful user creation.

Invalid Data: The input in both password and confirm password do not match.

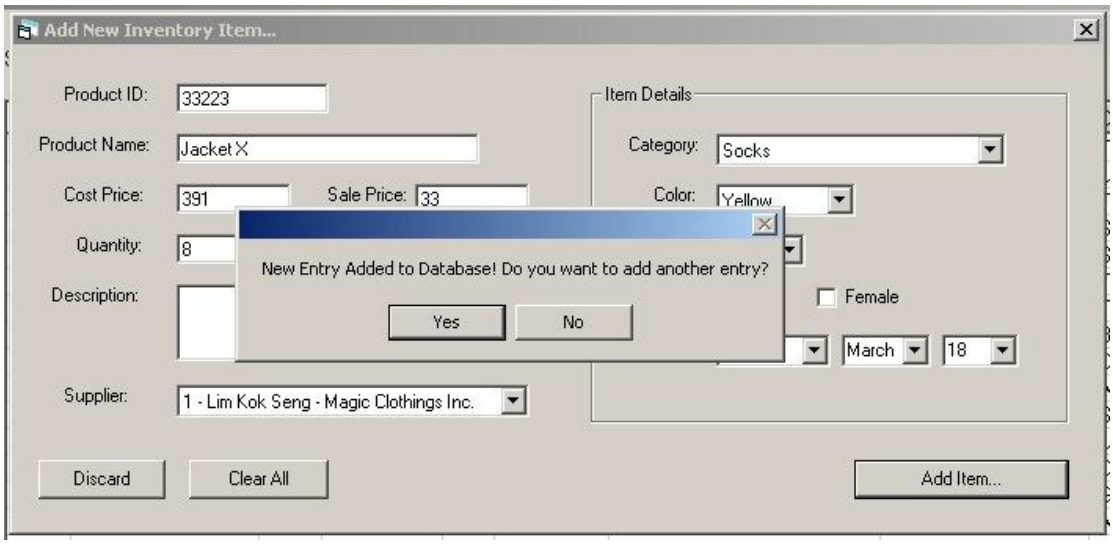
Result: The system will notify the user of not matching inputs in both fields.

Validation Check

Presence Check

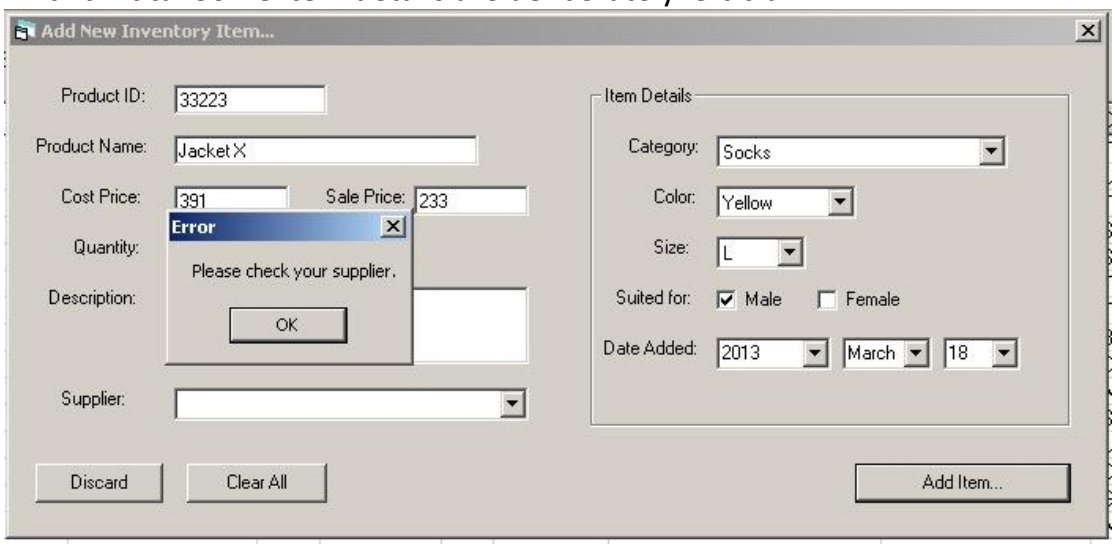
This test checks whether all required item details are filled in when adding new items.

Valid Data: All item details are filled in.



Result: A dialog box will appear to notify the user of successful entry.

Invalid Data: Some item details are deliberately left blank.

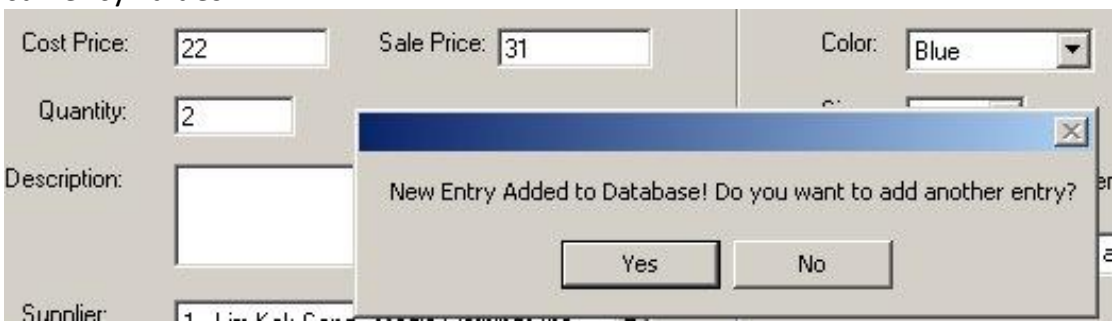


Result: An error dialog will appear asking the user to fill in the missing information.

Character Check

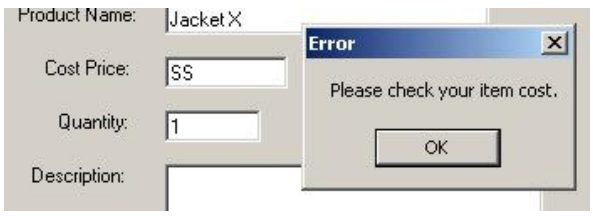
This test checks whether the input are conforming to the rules of the individual fields, e.g entering only numerical values on numerical fields.

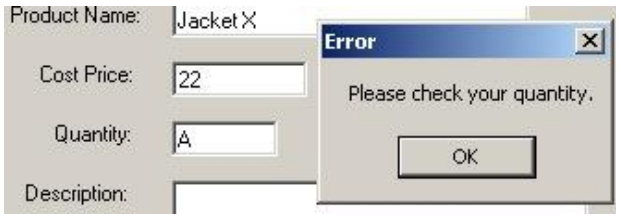
Valid Data: Quantity, Cost and Sale Price fields are filled with numerical currency values.



Result: A dialog box will appear to notify the user of successful entry.

Invalid Data: Quantity, Cost or Sale Price fields are inputted with alphabets.





Result: An error dialog will appear to notify the user of invalid entry in the specified fields.

Format Check

This test checks if the input is correct in format.

Valid Data: Username of length greater than 3.

Result: User created successfully and a dialog box appears to notify the user.

Invalid Data: Username of length smaller than 3.

Result: The system will notify the user to choose a longer username.

Extreme Data: Username of length 100

Result: The system will notify the user that the length of the username is too long.

Black Box Testing

Gross Profit and Total Cost Calculation

This test will test whether the gross profit and total cost calculation in the Sales window is correct.

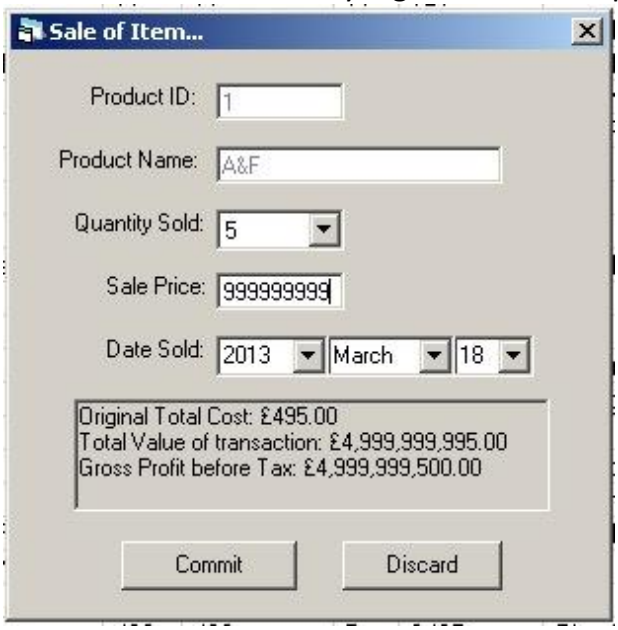
Valid Data: A typical number for selling price is inputted.

Results: The total cost and gross profit calculations are correctly displayed.

Invalid Data: Alphabets or symbols are inputted for selling price.

Results: The system will notify the user of an invalid selling price.

Extreme Data: Extremely high values are inputted into the field.



Results: The system functions as normal - total cost, gross profits etc are calculated and displayed.

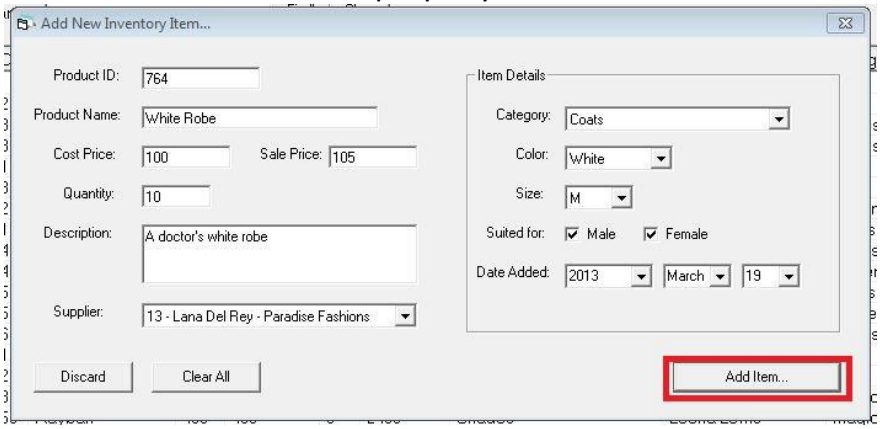
Alpha Testing

Adding New Records

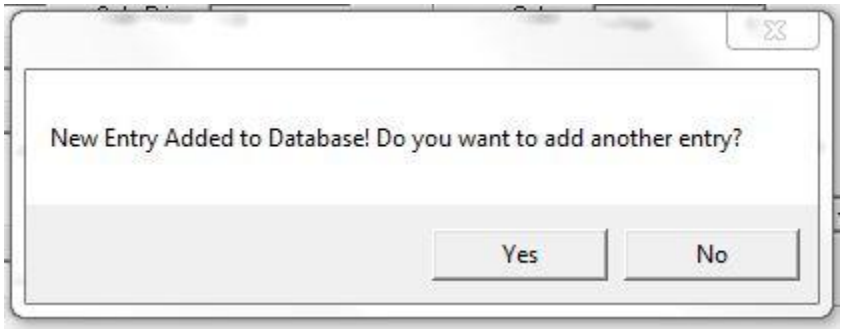
This test checks whether Add new record functionality is working as it should and is able to save the new record to database as well as displaying it in the list.

In the *Inventory List* window, the Add New Items… button is clicked, and when the New Item form comes up, all fields are filled in properly.

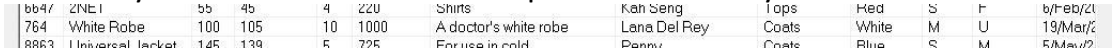
After all items are filled in properly, the *Add Item* button is clicked.



A message dialog will appear to inform the user of successful addition to the database.



The newly added item should show up in the inventory list view.

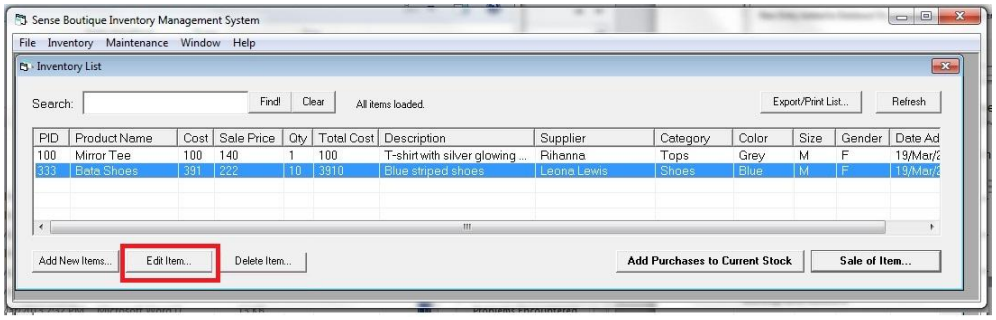


Looking inside the database.mdb file will confirm that the new record has been added.

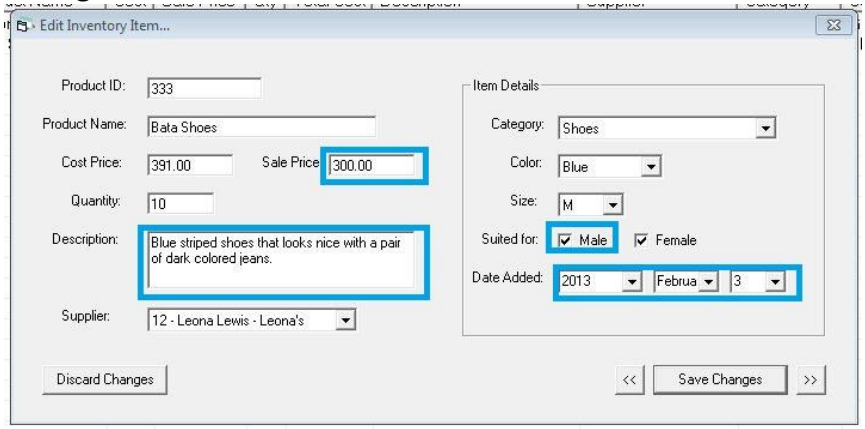
Editing Records

This test checks whether the editing record function is working, and is able to save any changes to a particular item record to the database.

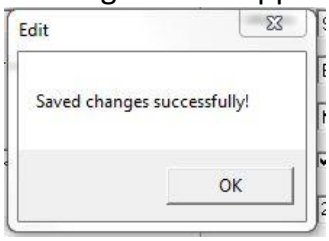
An item is selected in the list, and the *Edit Item…* button is clicked.



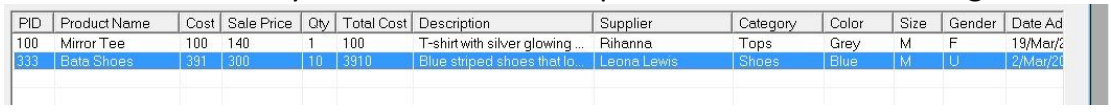
Sale price, description, gender and date added are changed. Then the Save Changes button is clicked.



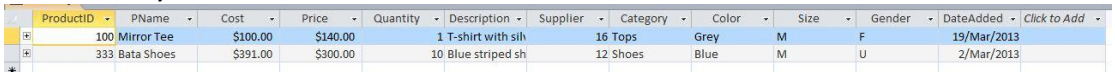
A dialog box will appear to inform of successful saving of changes.



The item in inventory list window will be updated to reflect the changes.



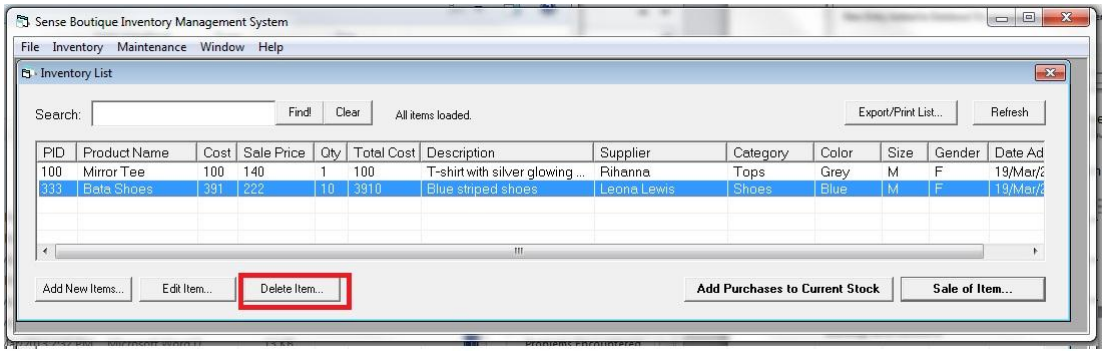
Looking inside Database.mdb, the record will appear to be changed successfully.



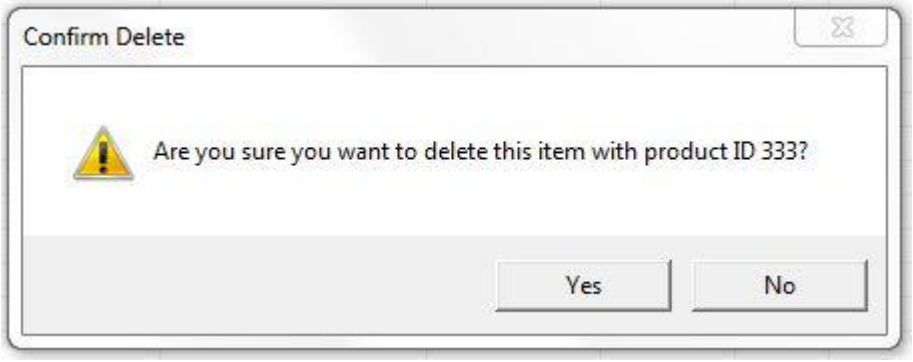
Deleting Records

This test checks whether the deleting item function is working as it should and is able to delete a record from database.

An item is selected from the list, and the Delete Item button is clicked.

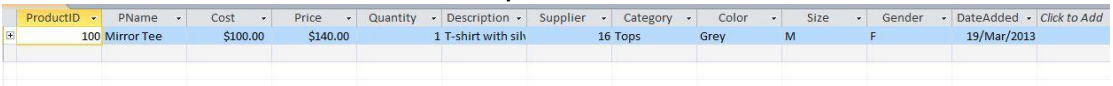


A confirmation dialog will appear to ask whether to confirm delete.



Click the Yes button and the item will be deleted. The inventory list will no longer display the item.

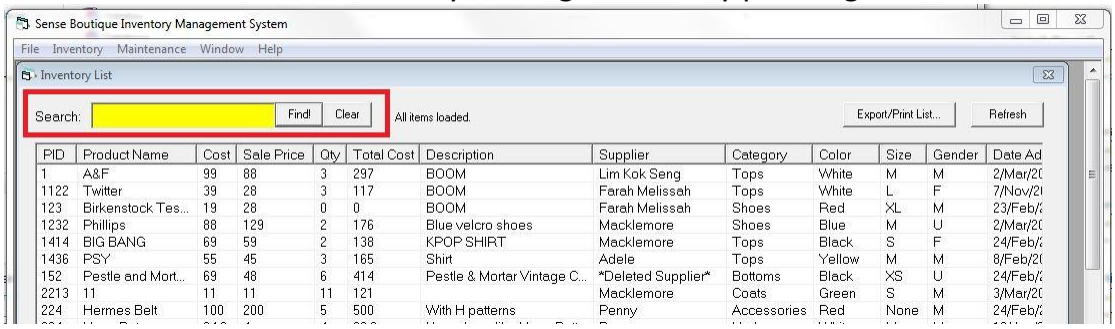
The record will also be deleted from the database.mdb file, showing that the record has been deleted successfully.



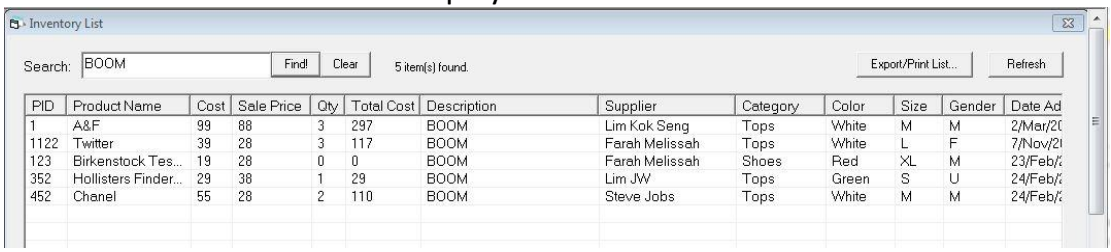
Search Records

This test checks whether the searching record feature is working, and is able to retrieve the record that the user is asking for.

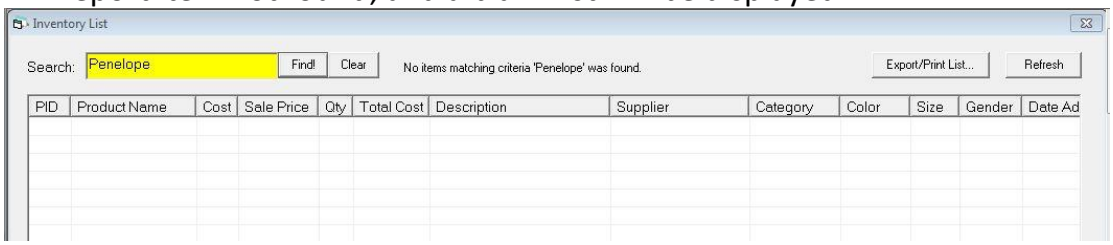
The Search textbox is focused by clicking on it, or by pressing the F3 button.



As the search term “BOOM” is typed in, items matching or containing the search term are filtered and displayed.



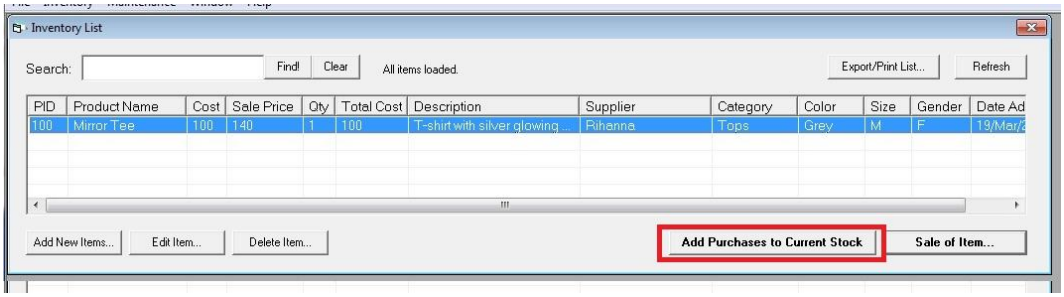
If a non-existing or invalid search term (e.g “Penelope”) is typed in, the system will report item not found, and a blank list will be displayed.



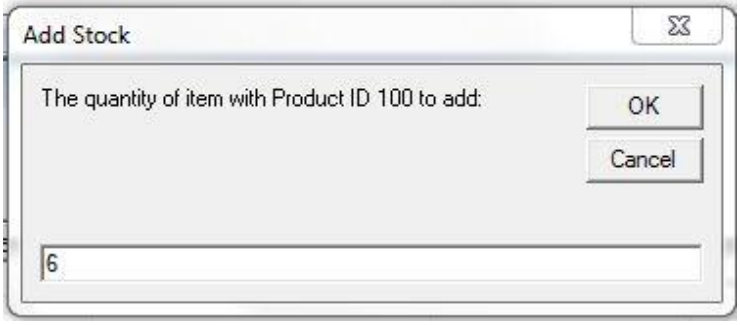
Adding Purchases

This test checks whether the adding purchases feature successfully updates the new stock level and records it in the transaction journal as a Purchases.

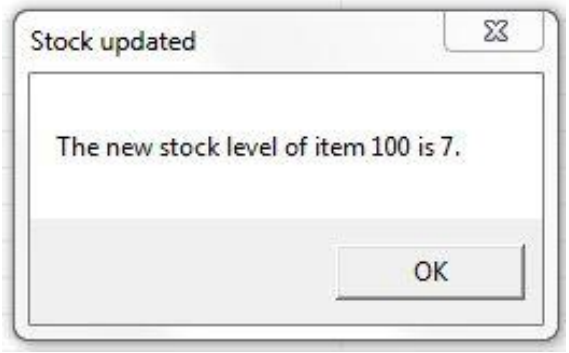
An item is selected from the list (with current stock quantity: 1). The “Add Purchases to Current Stock” button is clicked.



A number is inputted into the subsequent dialog box (e.g 6).



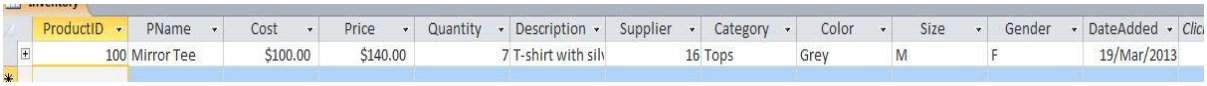
A dialog box will appear informing the updated stock level.



The new stock level will be reflected in the list.

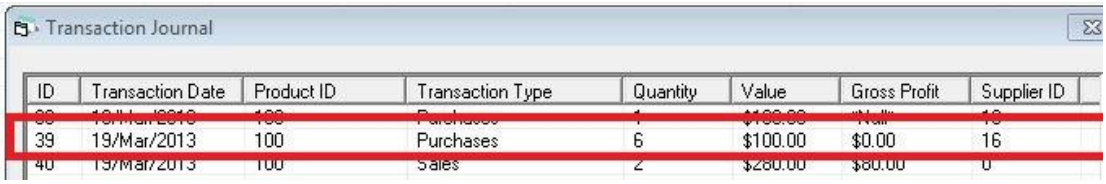
PID	Product Name	Cost	Sale Price	Qty	Total Cost	Descriptio
100	Mirror Tee	100	140	7	700	T-shirt with

Record in Database.mdb

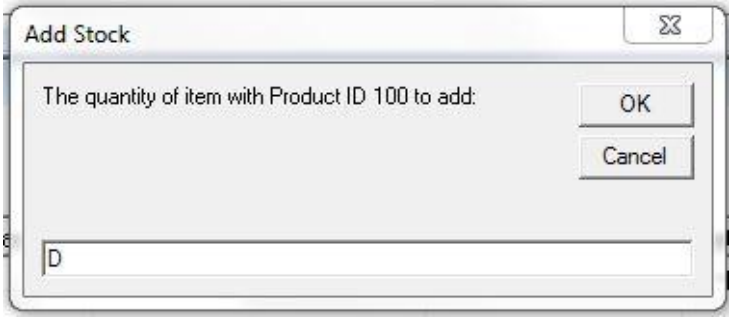


ProductID	PName	Cost	Price	Quantity	Description	Supplier	Category	Color	Size	Gender	DateAdded
100	Mirror Tee	$100.00	$140.00	7	T-shirt with silv	16	Tops	Grey	M	F	19/Mar/2013

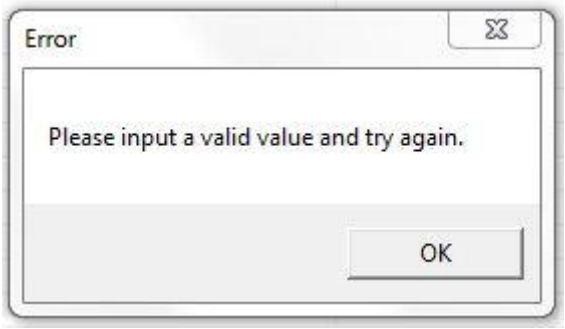
An entry will appear in the transaction journal showing the Purchases transaction.



If a non-numerical value is inputted (e.g Alphabets),



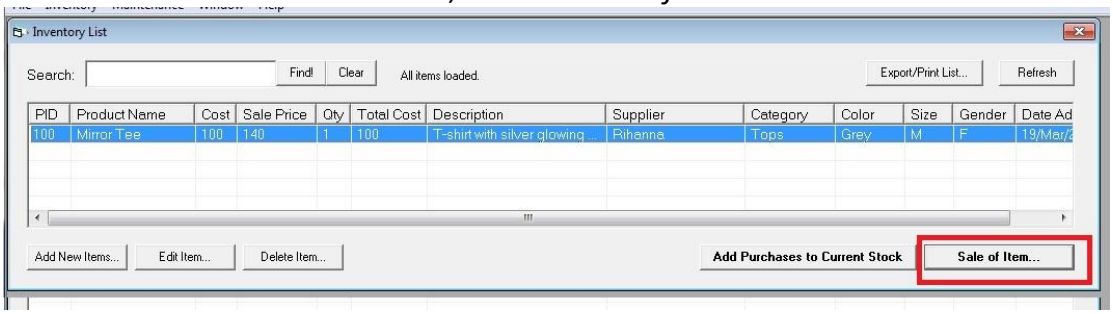
An error message will appear to inform the user of the invalid input.



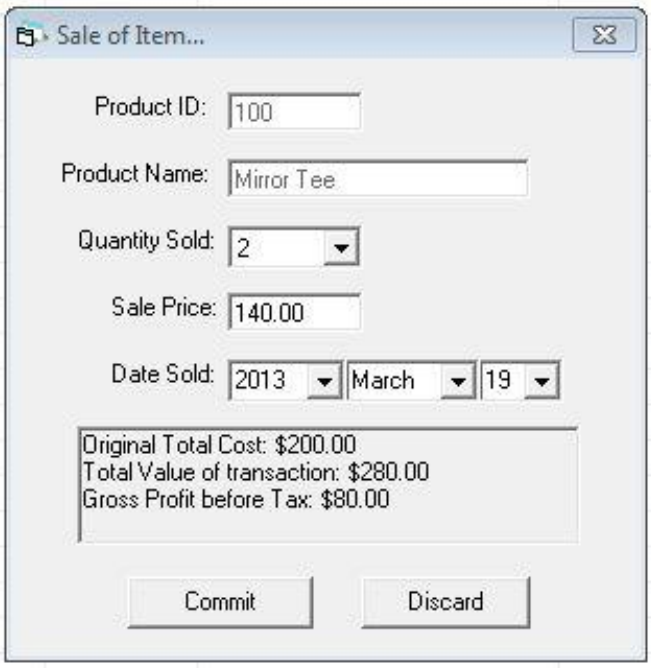
Adding Sales records

This test checks whether the recording sales transactions feature successfully updates the stock level and records it in the transaction journal as a Sale.

An item is selected from the list, and the *Sale of Item…* button is clicked.



The sale of item window will appear. A value is selected from the dropdown list for Quantity Sold (e.g. 2), the picture box below will update to display the total cost of goods, total value of transaction, and gross profit estimated.



Upon clicking the commit button, a dialog box will appear informing the user of successful record entry.

The sale will be reflected in the Inventory List and also the database.mdb file.

PID	Product Name	Cost	Sale Price	Qty	T
100	Mirror Tee	100	140	5	5

ProductID	PName	Cost	Price	Quantity	Description	Supplier	Cat
100	Mirror Tee	$100.00	$140.00	5	T-shirt with silv	16	Tops

A new transaction record will be added to the transaction journal report, showing the new sale transaction.

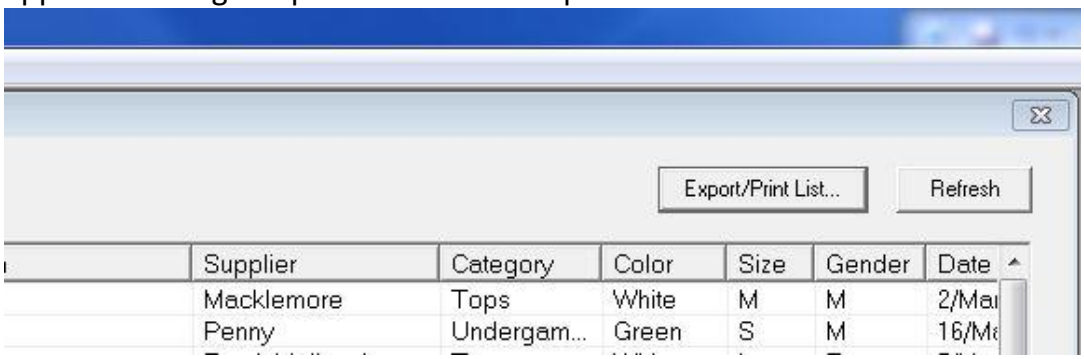
Transaction Journal

ID	Transaction Date	Product ID	Transaction Type	Quantity	Value	Gross Profit	Supplier ID
38	19/Mar/2013	100	Purchases	1	$100.00	*Null*	16
40	19/Mar/2013	100	Sales	2	$280.00	$80.00	0

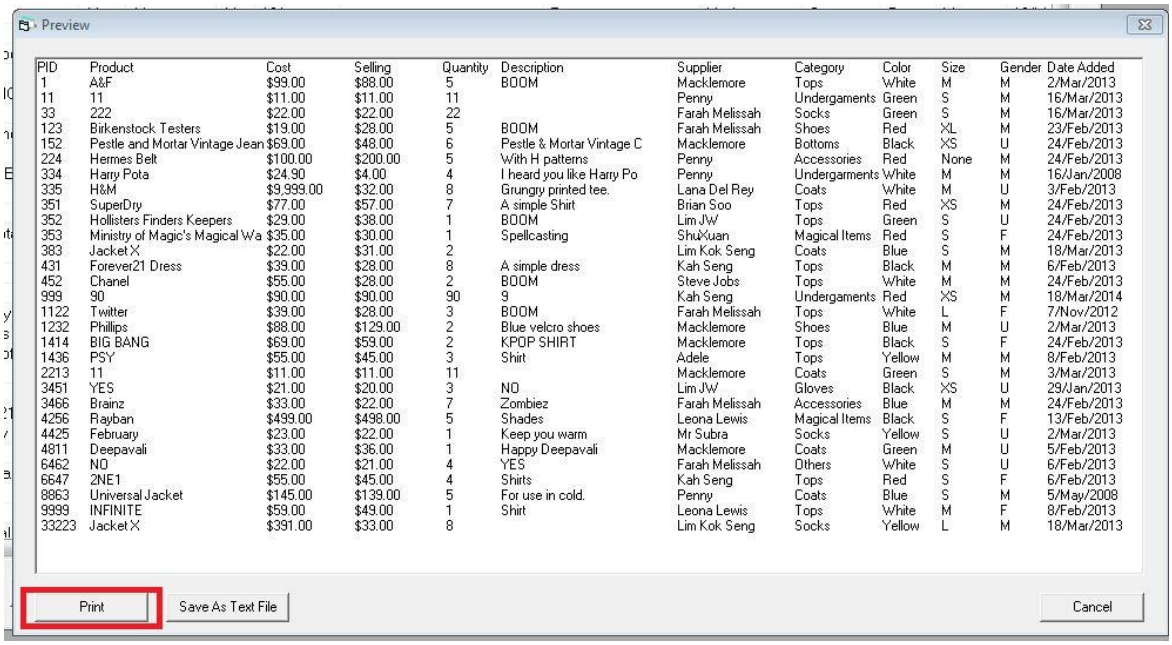
Print and Export list of items

This test checks whether the printing and exporting inventory list as text file options are working.

When the *Export/Print List..* button is clicked, a print preview window will appear showing the preview for the output.

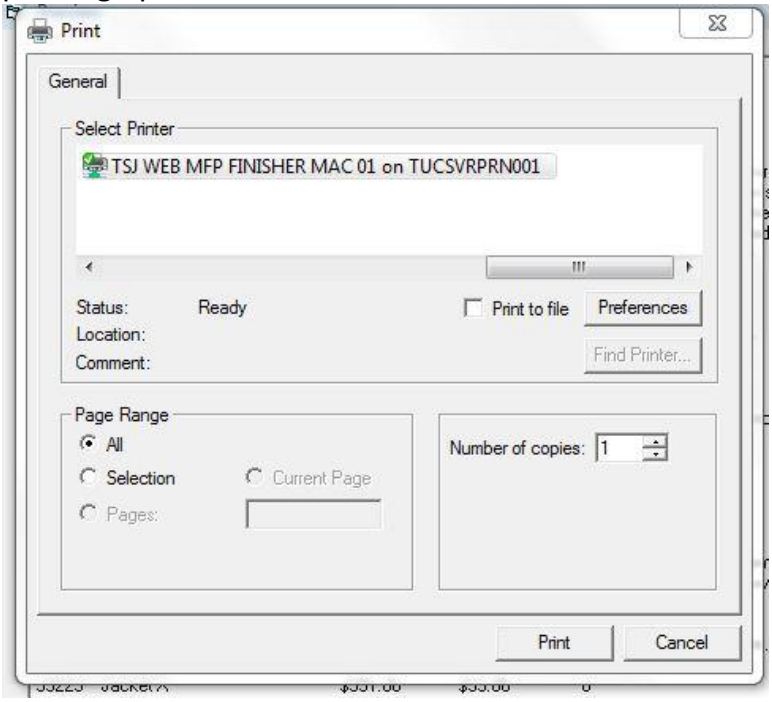


For printing, press the Print button at the bottom left of the Preview window.

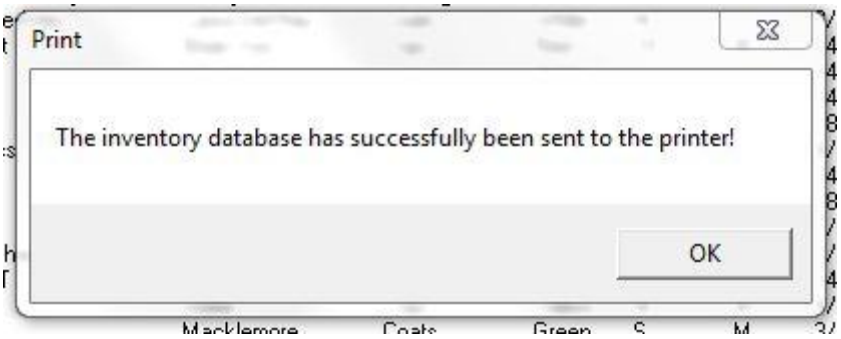


PID	Product	Cost	Selling	Quantity	Description	Supplier	Category	Color	Size	Gender	Date Added
1	A&F	$99.00	$88.00	5	BOOM	Macklemore	Tops	White	M	M	2/Mar/2013
11	11	$11.00	$11.00	11		Penny	Undergaments	Green	S	M	16/Mar/2013
33	222	$22.00	$22.00	22		Farah Melissah	Socks	Green	S	M	16/Mar/2013
123	Birkenstock Testers	$19.00	$28.00	5	BOOM	Farah Melissah	Shoes	Red	XL	M	23/Feb/2013
152	Pestle and Mortar Vintage Jean	$69.00	$48.00	6	Pestle & Mortar Vintage C	Macklemore	Bottoms	Black	XS	U	24/Feb/2013
224	Hermes Belt	$100.00	$200.00	5	With H patterns	Penny	Accessories	Red	None	M	24/Feb/2013
334	Harry Pota	$24.90	$4.00	4	I heard you like Harry Po	Penny	Undergarments	White	M	M	16/Jan/2008
335	H&M	$9,999.00	$32.00	8	Grungry printed tee.	Lana Del Rey	Coats	White	M	U	3/Feb/2013
351	SuperDry	$77.00	$57.00	7	A simple Shirt	Brian Soo	Tops	Red	XS	M	24/Feb/2013
352	Hollisters Finders Keepers	$29.00	$38.00	1	BOOM	Lim JW	Tops	Green	S	U	24/Feb/2013
353	Ministry of Magic's Magical Wa	$35.00	$30.00	1	Spellcasting	ShuXuan	Magical Items	Red	S	F	24/Feb/2013
383	Jacket X	$22.00	$31.00	2		Lim Kok Seng	Coats	Blue	S	M	18/Mar/2013
431	Forever21 Dress	$39.00	$28.00	8	A simple dress	Kah Seng	Tops	Black	M	M	6/Feb/2013
452	Chanel	$55.00	$28.00	2	BOOM	Steve Jobs	Tops	White	M	M	24/Feb/2013
999	90	$90.00	$90.00	90	9	Kah Seng	Undergaments	Red	XS	M	18/Mar/2014
1122	Twitter	$39.00	$28.00	3	BOOM	Farah Melissah	Tops	White	L	F	7/Nov/2012
1232	Phillips	$88.00	$129.00	2	Blue velcro shoes	Macklemore	Shoes	Blue	M	U	2/Mar/2013
1414	BIG BANG	$69.00	$59.00	2	KPOP SHIRT	Macklemore	Tops	Black	S	F	24/Feb/2013
1436	PSY	$55.00	$45.00	3	Shirt	Adele	Tops	Yellow	M	M	8/Feb/2013
2213	11	$11.00	$11.00	11		Macklemore	Coats	Green	S	M	3/Mar/2013
3451	YES	$21.00	$20.00	3	NO	Lim JW	Gloves	Black	XS	U	29/Jan/2013
3466	Brainz	$33.00	$22.00	7	Zombiez	Farah Melissah	Accessories	Blue	M	M	24/Feb/2013
4256	Rayban	$499.00	$498.00	5	Shades	Leona Lewis	Magical Items	Black	S	F	13/Feb/2013
4425	February	$23.00	$22.00	1	Keep you warm	Mr Subra	Socks	Yellow	S	U	2/Mar/2013
4811	Deepavali	$33.00	$36.00	1	Happy Deepavali	Macklemore	Coats	Green	M	U	5/Feb/2013
6462	NO	$22.00	$21.00	4	YES	Farah Melissah	Others	White	S	U	6/Feb/2013
6647	2NE1	$55.00	$45.00	4	Shirts	Kah Seng	Tops	Red	S	F	6/Feb/2013
8863	Universal Jacket	$145.00	$139.00	5	For use in cold.	Penny	Coats	Blue	S	M	5/May/2008
9999	INFINITE	$59.00	$49.00	1	Shirt	Leona Lewis	Tops	White	M	F	8/Feb/2013
33223	Jacket X	$391.00	$33.00	8		Lim Kok Seng	Socks	Yellow	L	M	18/Mar/2013

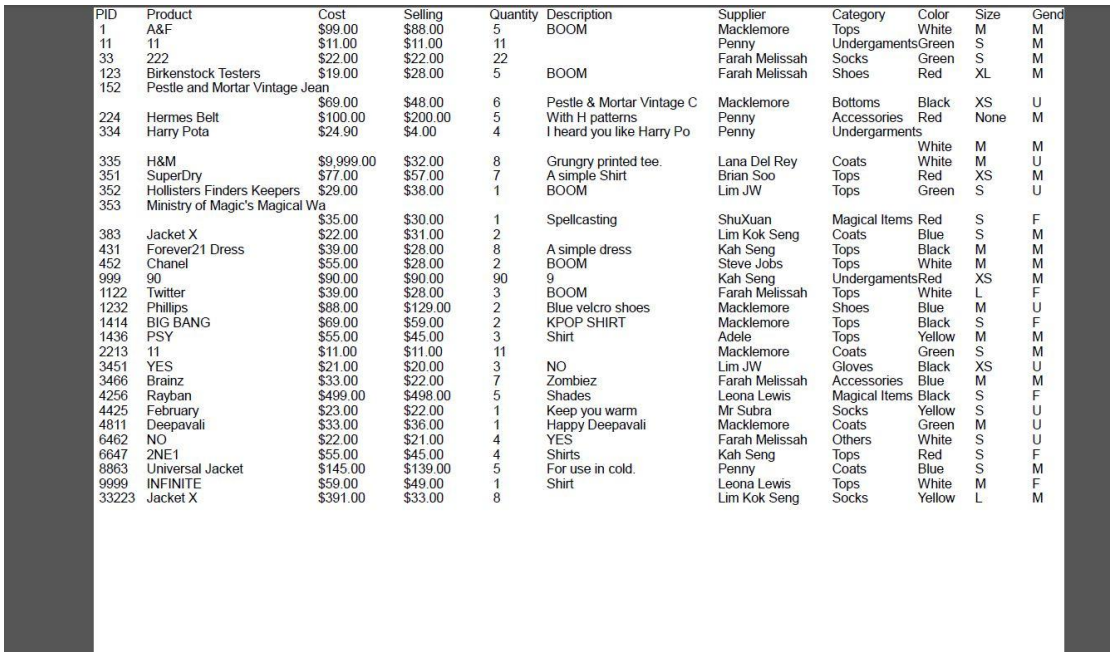
A print common dialog appears to allow the user to select printer or other printing options.



Press the print button, and a message will appear to inform the user that the data has been successfully been sent to the printer.

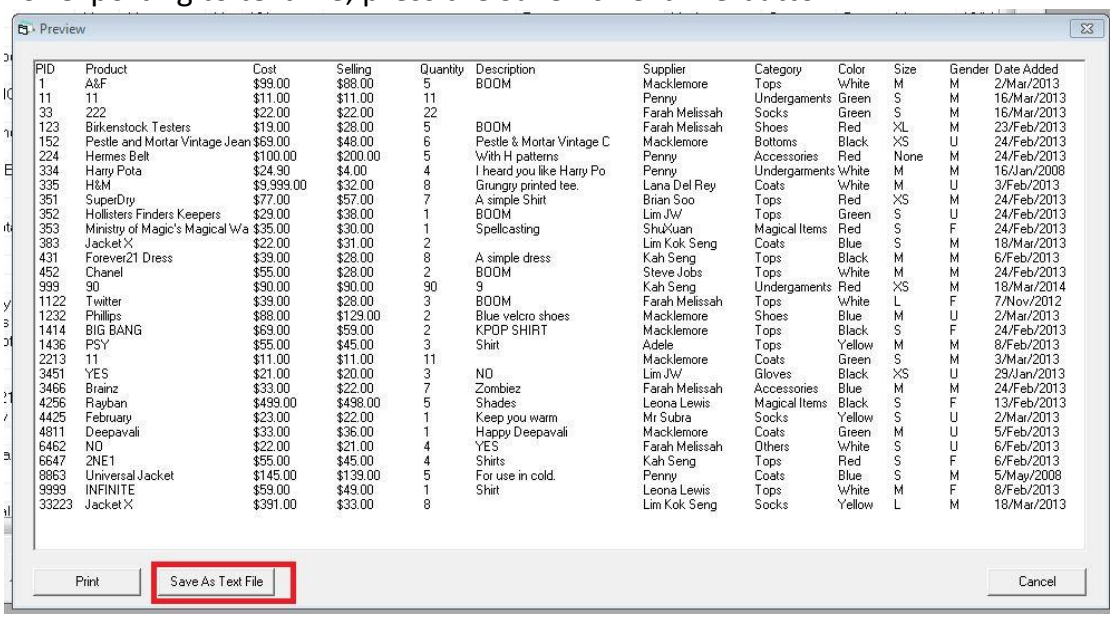


This is the output in physical paper form.



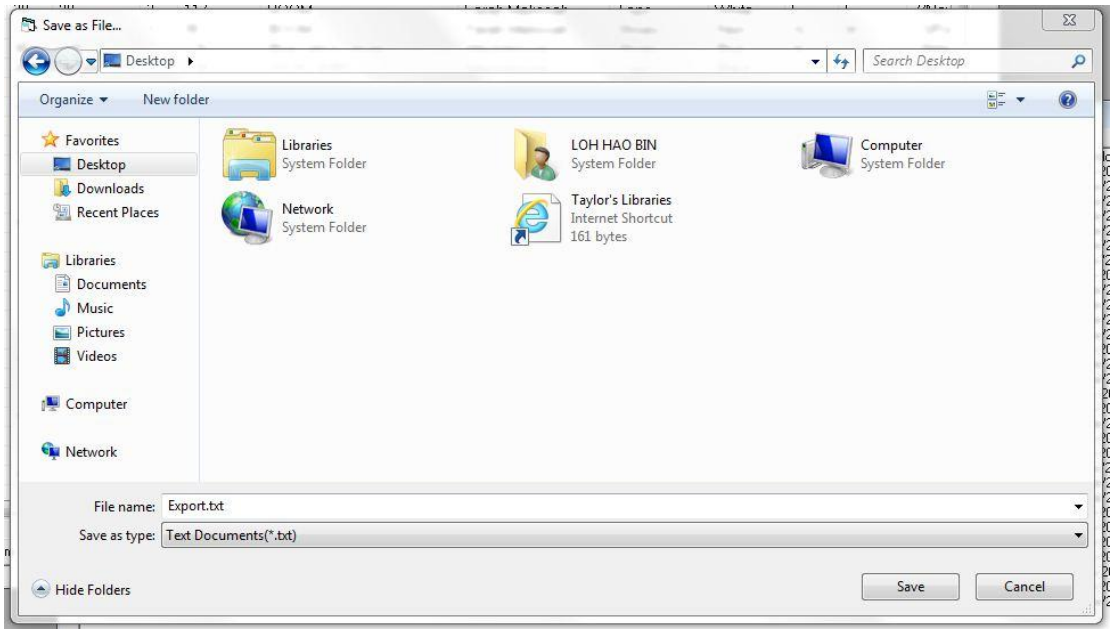
PID	Product	Cost	Selling	Quantity	Description	Supplier	Category	Color	Size	Gend
1	A&F	$99.00	$88.00	5	BOOM	Macklemore	Tops	White	M	M
11	11	$11.00	$11.00	11		Penny	Undergaments	Green	S	M
33	222	$22.00	$22.00	22		Farah Melissah	Socks	Green	S	M
123	Birkenstock Testers	$19.00	$28.00	5	BOOM	Farah Melissah	Shoes	Red	XL	M
152	Pestle and Mortar Vintage Jean									
		$69.00	$48.00	6	Pestle & Mortar Vintage C	Macklemore	Bottoms	Black	XS	U
224	Hermes Belt	$100.00	$200.00	5	With H patterns	Penny	Accessories	Red	None	M
334	Harry Pota	$24.90	$4.00	4	I heard you like Harry Po	Penny	Undergarments			
								White	M	M
335	H&M	$9,999.00	$32.00	8	Grungry printed tee.	Lana Del Rey	Coats	White	M	U
351	SuperDry	$77.00	$57.00	7	A simple Shirt	Brian Soo	Tops	Red	XS	M
352	Hollisters Finders Keepers	$29.00	$38.00	1	BOOM	Lim JW	Tops	Green	S	U
353	Ministry of Magic's Magical Wa									
		$35.00	$30.00	1	Spellcasting	ShuXuan	Magical Items	Red	S	F
383	Jacket X	$22.00	$31.00	2		Lim Kok Seng	Coats	Blue	S	M
431	Forever21 Dress	$39.00	$28.00	8	A simple dress	Kah Seng	Tops	Black	M	M
452	Chanel	$55.00	$28.00	2	BOOM	Steve Jobs	Tops	White	M	M
999	90	$90.00	$90.00	90	9	Kah Seng	Undergaments	Red	XS	M
1122	Twitter	$39.00	$28.00	3	BOOM	Farah Melissah	Tops	White	L	F
1232	Phillips	$88.00	$129.00	2	Blue velcro shoes	Macklemore	Shoes	Blue	M	U
1414	BIG BANG	$69.00	$59.00	2	KPOP SHIRT	Macklemore	Tops	Black	S	F
1436	PSY	$55.00	$45.00	3	Shirt	Adele	Tops	Yellow	M	M
2213	11	$11.00	$11.00	11		Macklemore	Coats	Green	S	M
3451	YES	$21.00	$20.00	3	NO	Lim JW	Gloves	Black	XS	U
3466	Brainz	$33.00	$22.00	7	Zombiez	Farah Melissah	Accessories	Blue	M	M
4256	Rayban	$499.00	$498.00	5	Shades	Leona Lewis	Magical Items	Black	S	F
4425	February	$23.00	$22.00	1	Keep you warm	Mr Subra	Socks	Yellow	S	U
4811	Deepavali	$33.00	$36.00	1	Happy Deepavali	Macklemore	Coats	Green	M	U
6462	NO	$22.00	$21.00	4	YES	Farah Melissah	Others	White	S	U
6647	2NE1	$55.00	$45.00	4	Shirts	Kah Seng	Tops	Red	S	F
8863	Universal Jacket	$145.00	$139.00	5	For use in cold.	Penny	Coats	Blue	S	M
9999	INFINITE	$59.00	$49.00	1	Shirt	Leona Lewis	Tops	White	M	F
33223	Jacket X	$391.00	$33.00	8		Lim Kok Seng	Socks	Yellow	L	M

For exporting to text file, press the *Save As Text File* button.

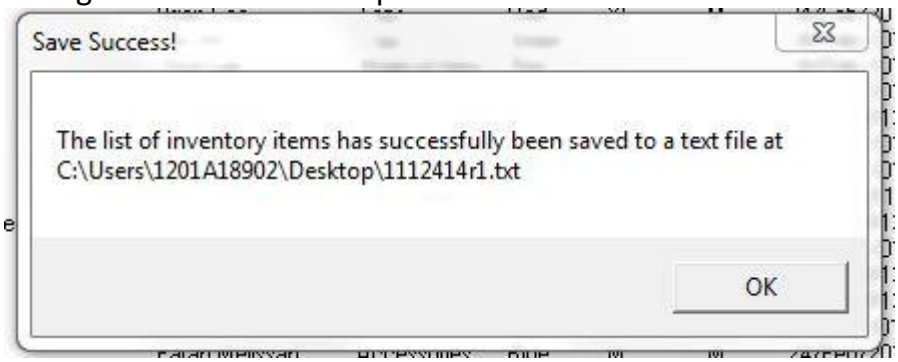


PID	Product	Cost	Selling	Quantity	Description	Supplier	Category	Color	Size	Gender	Date Added
1	A&F	$99.00	$88.00	5	BOOM	Macklemore	Tops	White	M	M	2/Mar/2013
11	11	$11.00	$11.00	11		Penny	Undergaments	Green	S	M	16/Mar/2013
33	222	$22.00	$22.00	22		Farah Melissah	Socks	Green	S	M	16/Mar/2013
123	Birkenstock Testers	$19.00	$28.00	5	BOOM	Farah Melissah	Shoes	Red	XL	M	23/Feb/2013
152	Pestle and Mortar Vintage Jean	$69.00	$48.00	6	Pestle & Mortar Vintage C	Macklemore	Bottoms	Black	XS	U	24/Feb/2013
224	Hermes Belt	$100.00	$200.00	5	With H patterns	Penny	Accessories	Red	None	M	24/Feb/2013
334	Harry Pota	$24.90	$4.00	4	I heard you like Harry Po	Penny	Undergarments	White	M	M	16/Jan/2008
335	H&M	$9,999.00	$32.00	8	Grungry printed tee.	Lana Del Rey	Coats	White	M	U	3/Feb/2013
351	SuperDry	$77.00	$57.00	7	A simple Shirt	Brian Soo	Tops	Red	XS	M	24/Feb/2013
352	Hollisters Finders Keepers	$29.00	$38.00	1	BOOM	Lim JW	Tops	Green	S	U	24/Feb/2013
353	Ministry of Magic's Magical Wa	$35.00	$30.00	1	Spellcasting	ShuXuan	Magical Items	Red	S	F	24/Feb/2013
383	Jacket X	$22.00	$31.00	2		Lim Kok Seng	Coats	Blue	S	M	18/Mar/2013
431	Forever21 Dress	$39.00	$28.00	8	A simple dress	Kah Seng	Tops	Black	M	M	6/Feb/2013
452	Chanel	$55.00	$28.00	2	BOOM	Steve Jobs	Tops	White	M	M	24/Feb/2013
999	90	$90.00	$90.00	90	9	Kah Seng	Undergaments	Red	XS	M	18/Mar/2014
1122	Twitter	$39.00	$28.00	3	BOOM	Farah Melissah	Tops	White	L	F	7/Nov/2012
1232	Phillips	$88.00	$129.00	2	Blue velcro shoes	Macklemore	Shoes	Blue	M	U	2/Mar/2013
1414	BIG BANG	$69.00	$59.00	2	KPOP SHIRT	Macklemore	Tops	Black	S	F	24/Feb/2013
1436	PSY	$55.00	$45.00	3	Shirt	Adele	Tops	Yellow	M	M	8/Feb/2013
2213	11	$11.00	$11.00	11		Macklemore	Coats	Green	S	M	3/Mar/2013
3451	YES	$21.00	$20.00	3	NO	Lim JW	Gloves	Black	XS	U	29/Jan/2013
3466	Brainz	$33.00	$22.00	7	Zombiez	Farah Melissah	Accessories	Blue	M	M	24/Feb/2013
4256	Rayban	$499.00	$498.00	5	Shades	Leona Lewis	Magical Items	Black	S	F	13/Feb/2013
4425	February	$23.00	$22.00	1	Keep you warm	Mr Subra	Socks	Yellow	S	U	2/Mar/2013
4811	Deepavali	$33.00	$36.00	1	Happy Deepavali	Macklemore	Coats	Green	M	U	5/Feb/2013
6462	NO	$22.00	$21.00	4	YES	Farah Melissah	Others	White	S	U	6/Feb/2013
6647	2NE1	$55.00	$45.00	4	Shirts	Kah Seng	Tops	Red	S	F	6/Feb/2013
8863	Universal Jacket	$145.00	$139.00	5	For use in cold.	Penny	Coats	Blue	S	M	5/May/2008
9999	INFINITE	$59.00	$49.00	1	Shirt	Leona Lewis	Tops	White	M	F	8/Feb/2013
33223	Jacket X	$391.00	$33.00	8		Lim Kok Seng	Socks	Yellow	L	M	18/Mar/2013

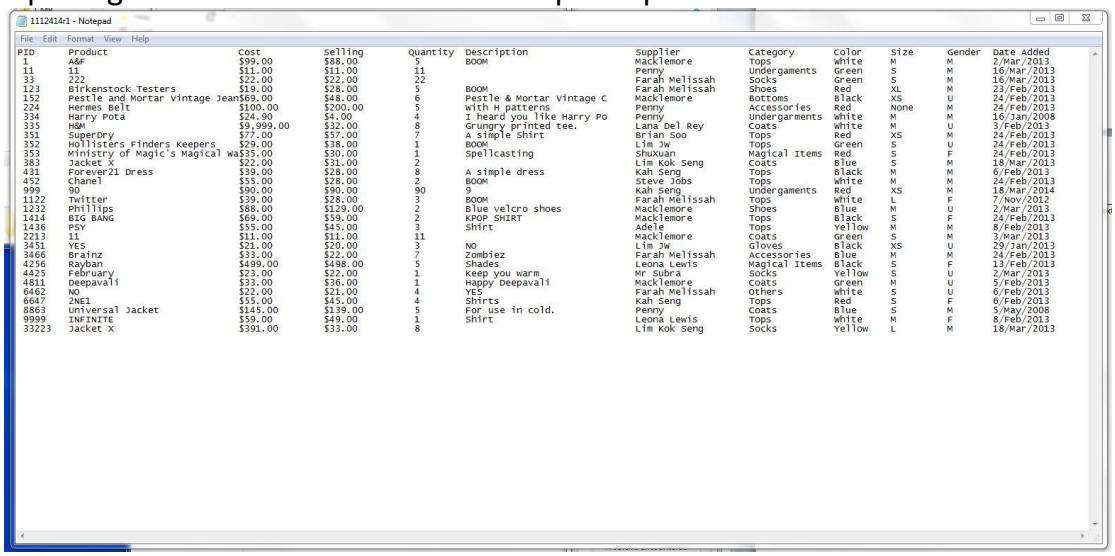
A file browser common dialog appears allowing the user to choose the location to save the file.



Press save, and a dialog box will appear informing the user of successful saving the text file to the specified location.



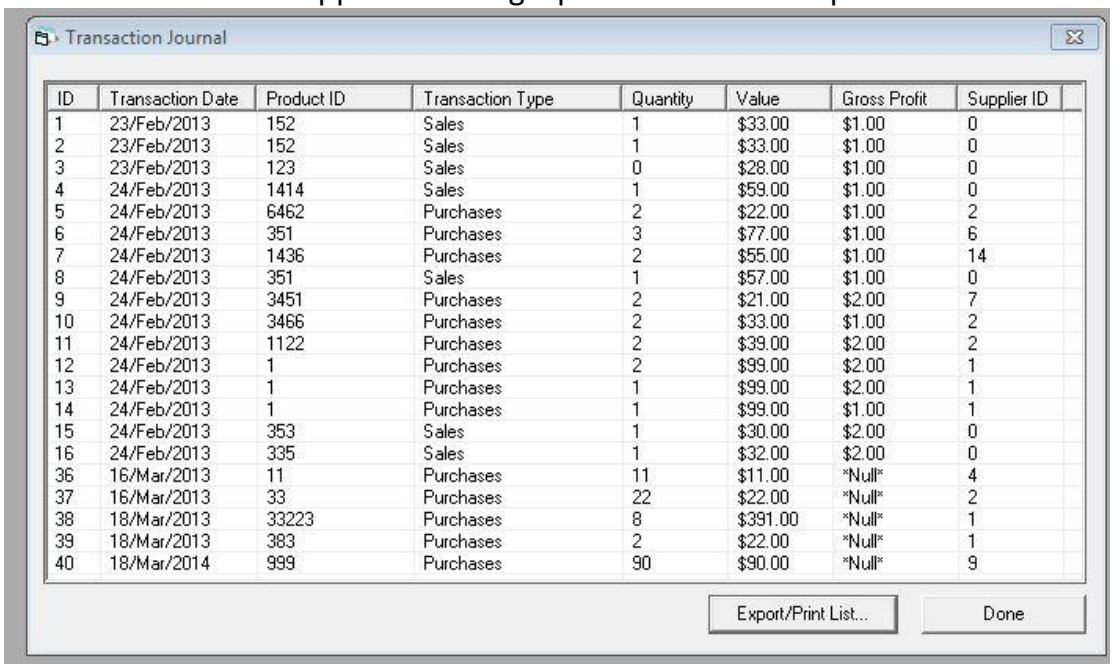
Opening the text file will reveal the output exported.



Print and Export journal reports

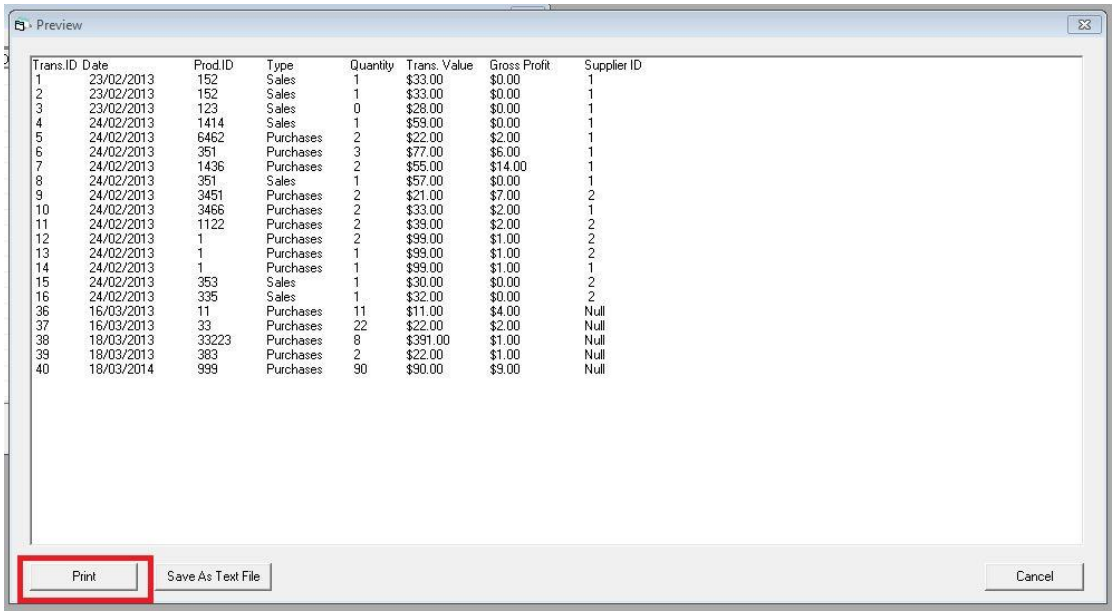
This test checks whether the printing and exporting transaction journal as text file options are working.

In the transaction journal, when the *Export/Print List…* button is pressed, a Preview window will appear showing a preview for the output.

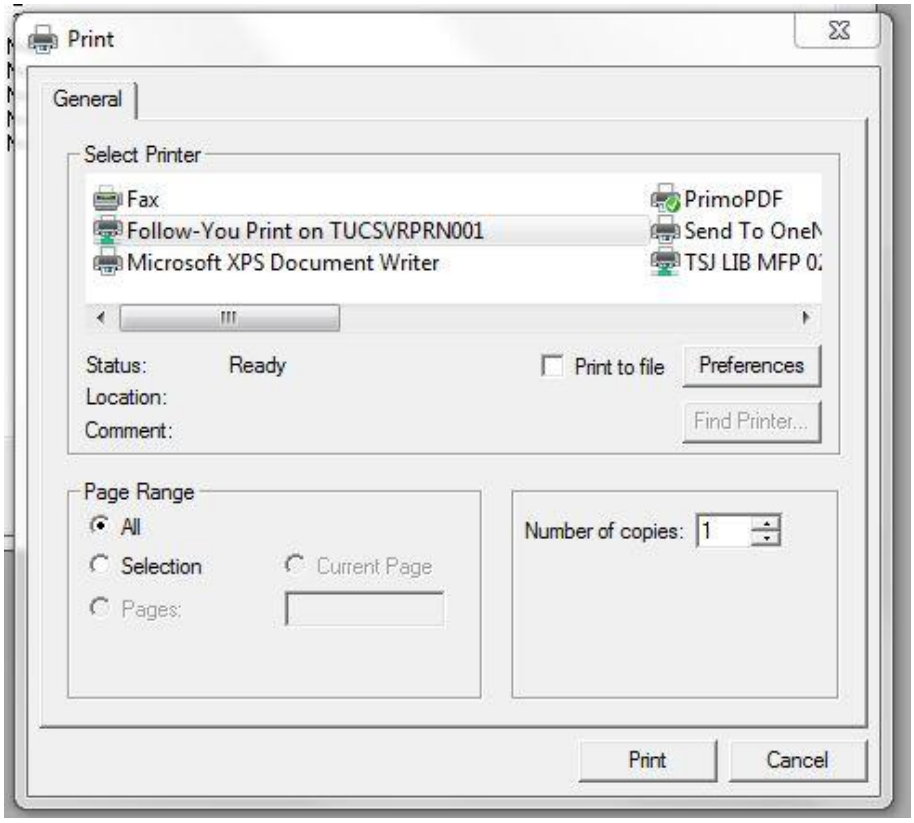


ID	Transaction Date	Product ID	Transaction Type	Quantity	Value	Gross Profit	Supplier ID
1	23/Feb/2013	152	Sales	1	$33.00	$1.00	0
2	23/Feb/2013	152	Sales	1	$33.00	$1.00	0
3	23/Feb/2013	123	Sales	0	$28.00	$1.00	0
4	24/Feb/2013	1414	Sales	1	$59.00	$1.00	0
5	24/Feb/2013	6462	Purchases	2	$22.00	$1.00	2
6	24/Feb/2013	351	Purchases	3	$77.00	$1.00	6
7	24/Feb/2013	1436	Purchases	2	$55.00	$1.00	14
8	24/Feb/2013	351	Sales	1	$57.00	$1.00	0
9	24/Feb/2013	3451	Purchases	2	$21.00	$2.00	7
10	24/Feb/2013	3466	Purchases	2	$33.00	$1.00	2
11	24/Feb/2013	1122	Purchases	2	$39.00	$2.00	2
12	24/Feb/2013	1	Purchases	2	$99.00	$2.00	1
13	24/Feb/2013	1	Purchases	1	$99.00	$2.00	1
14	24/Feb/2013	1	Purchases	1	$99.00	$1.00	1
15	24/Feb/2013	353	Sales	1	$30.00	$2.00	0
16	24/Feb/2013	335	Sales	1	$32.00	$2.00	0
36	16/Mar/2013	11	Purchases	11	$11.00	*Null*	4
37	16/Mar/2013	33	Purchases	22	$22.00	*Null*	2
38	18/Mar/2013	33223	Purchases	8	$391.00	*Null*	1
39	18/Mar/2013	383	Purchases	2	$22.00	*Null*	1
40	18/Mar/2014	999	Purchases	90	$90.00	*Null*	9

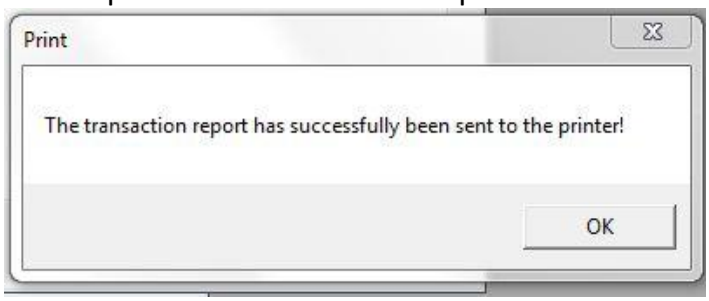
When the print button is pressed, a print dialog box will appear displaying options for the user to choose printer and various printing options.



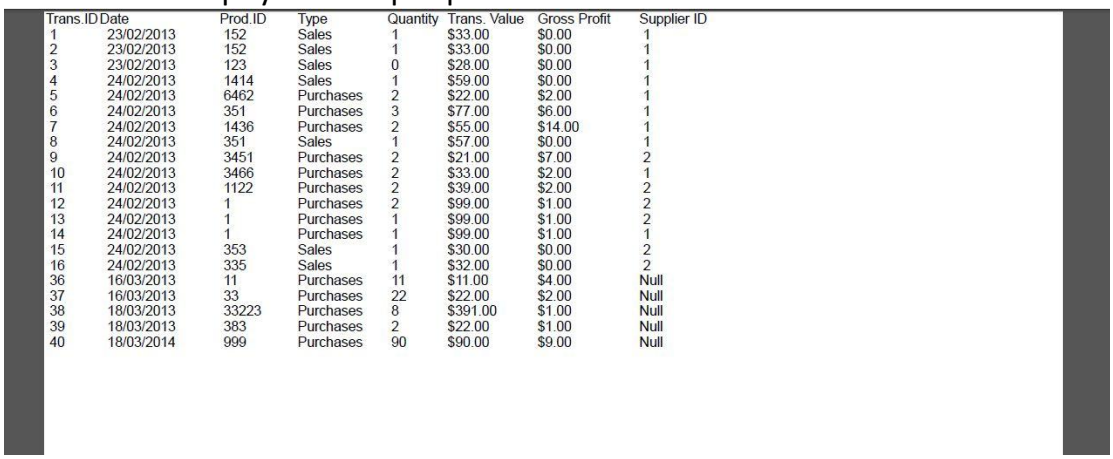
Trans.ID	Date	Prod.ID	Type	Quantity	Trans. Value	Gross Profit	Supplier ID
1	23/02/2013	152	Sales	1	$33.00	$0.00	1
2	23/02/2013	152	Sales	1	$33.00	$0.00	1
3	23/02/2013	123	Sales	0	$28.00	$0.00	1
4	24/02/2013	1414	Sales	1	$59.00	$0.00	1
5	24/02/2013	6462	Purchases	2	$22.00	$2.00	1
6	24/02/2013	351	Purchases	3	$77.00	$6.00	1
7	24/02/2013	1436	Purchases	2	$55.00	$14.00	1
8	24/02/2013	351	Sales	1	$57.00	$0.00	1
9	24/02/2013	3451	Purchases	2	$21.00	$7.00	2
10	24/02/2013	3466	Purchases	2	$33.00	$2.00	1
11	24/02/2013	1122	Purchases	2	$39.00	$2.00	2
12	24/02/2013	1	Purchases	2	$99.00	$1.00	2
13	24/02/2013	1	Purchases	1	$99.00	$1.00	2
14	24/02/2013	1	Purchases	1	$99.00	$1.00	1
15	24/02/2013	353	Sales	1	$30.00	$0.00	2
16	24/02/2013	335	Sales	1	$32.00	$0.00	2
36	16/03/2013	11	Purchases	11	$11.00	$4.00	Null
37	16/03/2013	33	Purchases	22	$22.00	$2.00	Null
38	18/03/2013	33223	Purchases	8	$391.00	$1.00	Null
39	18/03/2013	383	Purchases	2	$22.00	$1.00	Null
40	18/03/2014	999	Purchases	90	$90.00	$9.00	Null



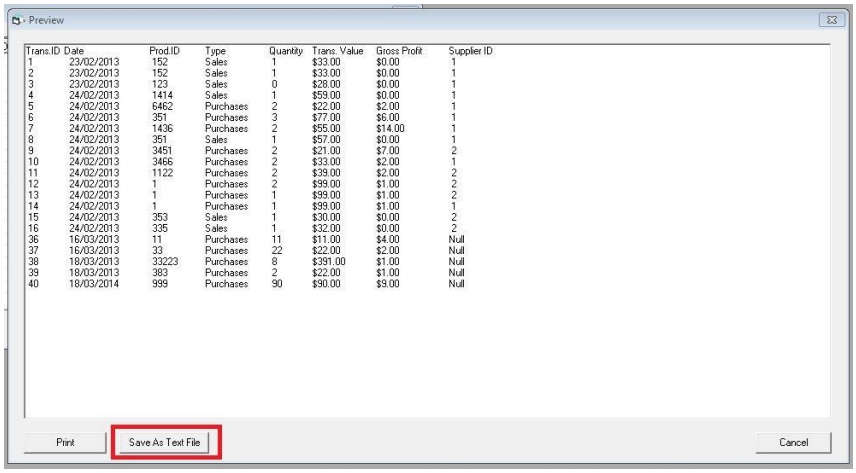
The print button is pressed, and a dialog box appears informing the user that the output has been sent to the printer.



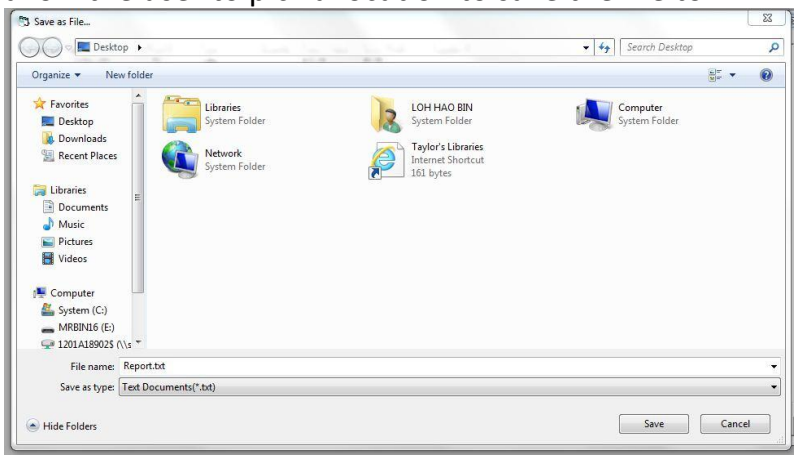
This is how the physical output printed look like:



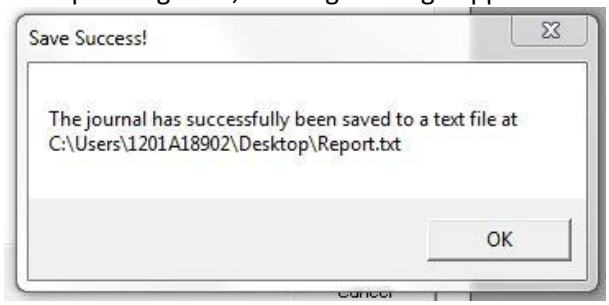
Trans.ID	Date	Prod.ID	Type	Quantity	Trans. Value	Gross Profit	Supplier ID
1	23/02/2013	152	Sales	1	$33.00	$0.00	1
2	23/02/2013	152	Sales	1	$33.00	$0.00	1
3	23/02/2013	123	Sales	0	$28.00	$0.00	1
4	24/02/2013	1414	Sales	1	$59.00	$0.00	1
5	24/02/2013	6462	Purchases	2	$22.00	$2.00	1
6	24/02/2013	351	Purchases	3	$77.00	$6.00	1
7	24/02/2013	1436	Purchases	2	$55.00	$14.00	1
8	24/02/2013	351	Sales	1	$57.00	$0.00	1
9	24/02/2013	3451	Purchases	2	$21.00	$7.00	2
10	24/02/2013	3466	Purchases	2	$33.00	$2.00	1
11	24/02/2013	1122	Purchases	2	$39.00	$2.00	2
12	24/02/2013	1	Purchases	2	$99.00	$1.00	2
13	24/02/2013	1	Purchases	1	$99.00	$1.00	2
14	24/02/2013	1	Purchases	1	$99.00	$1.00	1
15	24/02/2013	353	Sales	1	$30.00	$0.00	2
16	24/02/2013	335	Sales	1	$32.00	$0.00	2
36	16/03/2013	11	Purchases	11	$11.00	$4.00	Null
37	16/03/2013	33	Purchases	22	$22.00	$2.00	Null
38	18/03/2013	33223	Purchases	8	$391.00	$1.00	Null
39	18/03/2013	383	Purchases	2	$22.00	$1.00	Null
40	18/03/2014	999	Purchases	90	$90.00	$9.00	Null



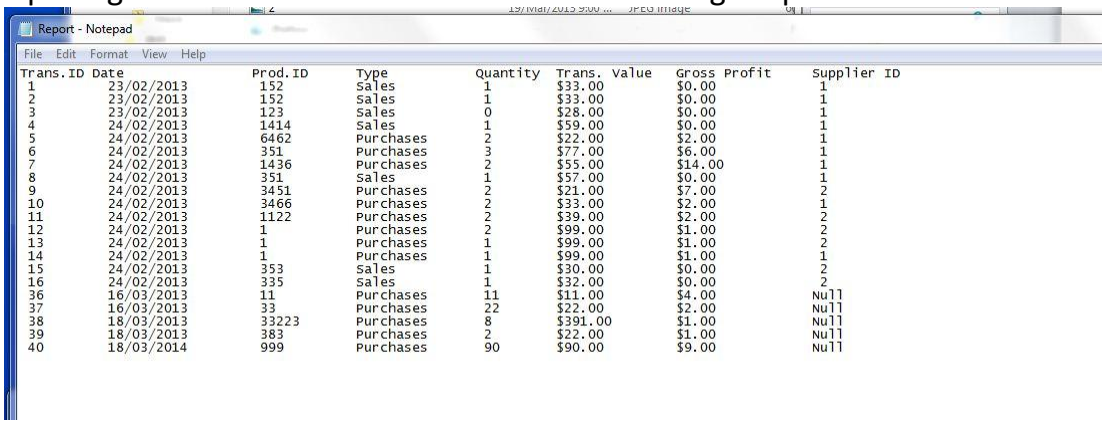
The Save As Text File button is pressed, and a file browser dialog appears to allow the user to pick a location to save the file to.



After pressing Save, a dialog message appears to inform the user of successful saving.



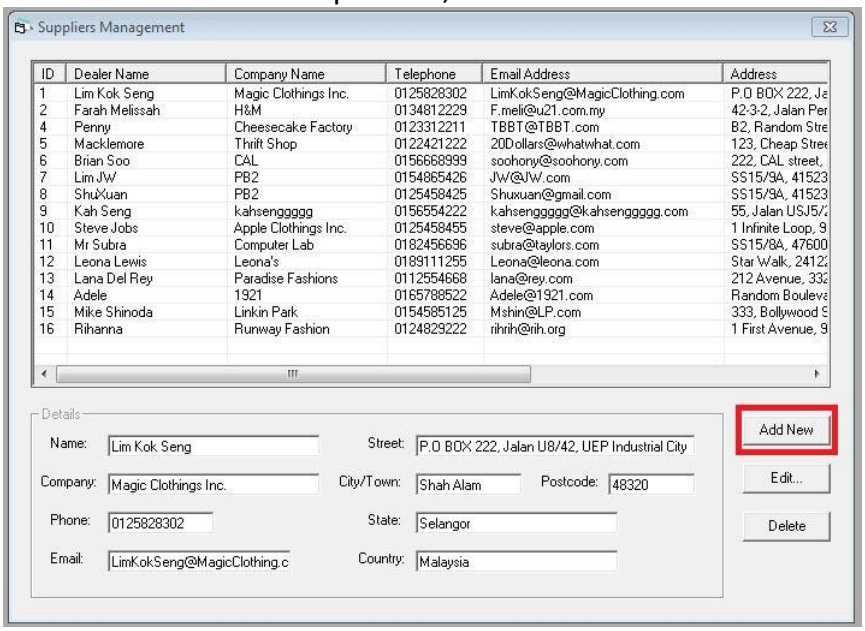
Opening the saved text file will reveal the following output:

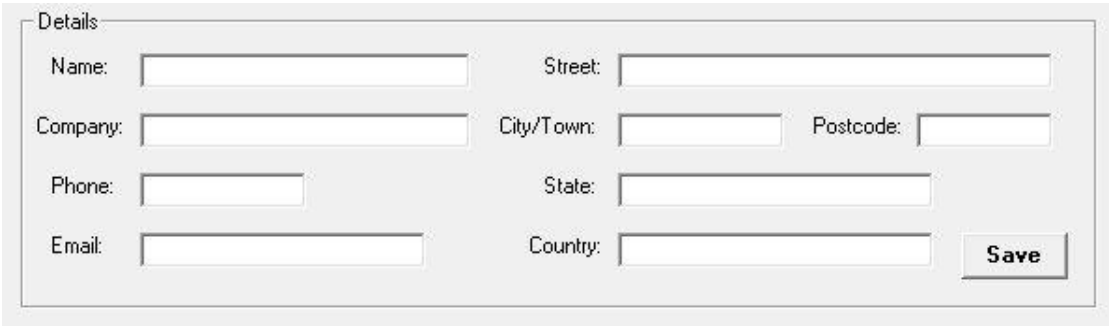


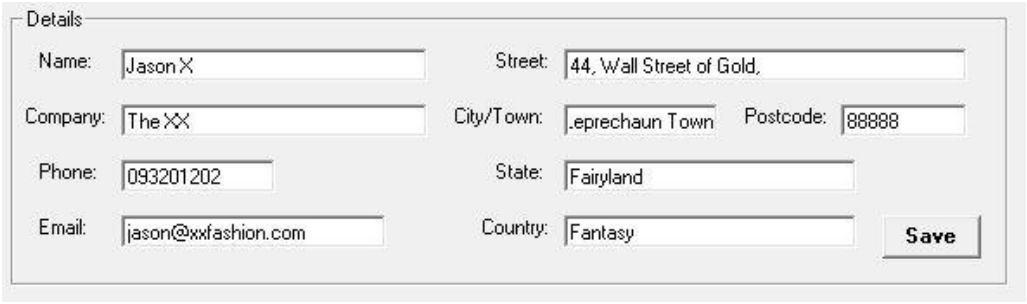
Adding, Editing and Deleting Suppliers

This test checks whether supplier management features are working as intended.

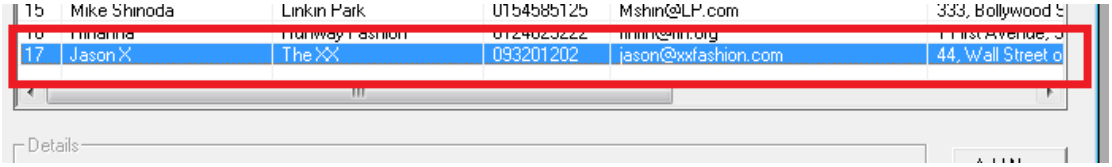
The Add New button is pressed, then the various textboxes are emptied.







Valid supplier data and values are filled into each of the textboxes, and then the Save button is pressed.



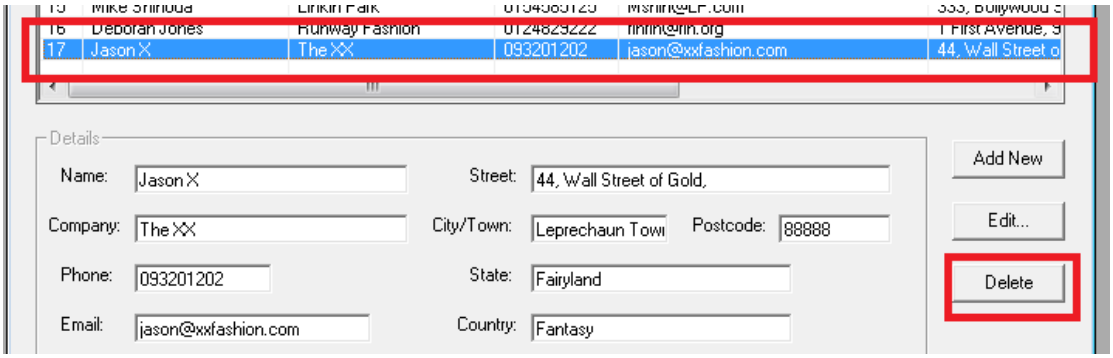
A new entry appeared in the supplier's list above, showing that the new record has been added.

Then, another entry is selected from the list, and the Edit… button is pressed. The Details fields became editable and a Save button appears.

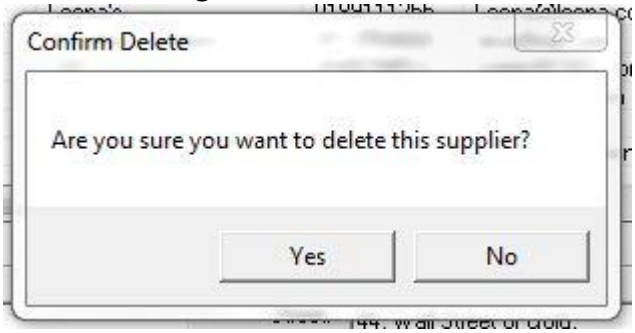
Some of the details (e.g. Name) are changed, and the Save button is pressed.

The changes are immediately reflected on the list above and database.mdb file, showing the new changes have been saved.

Then, an entry is selected from the list, and the Delete button is pressed.



A dialog box will appear to confirm to the user that the selected supplier record is being deleted.

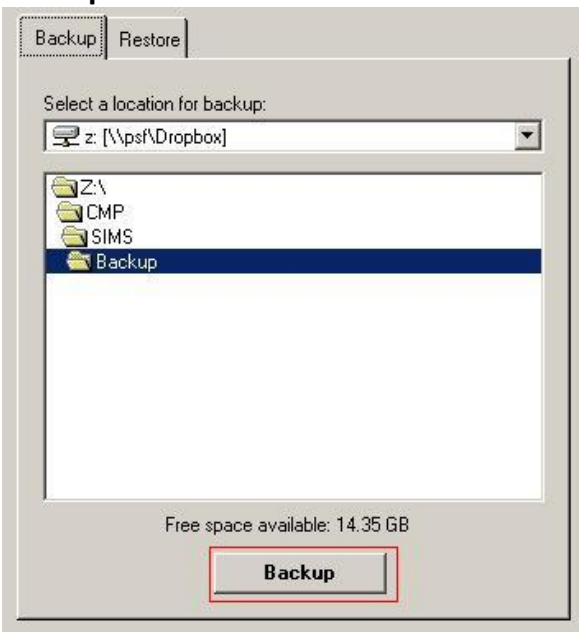


The deleted entry will no longer appear on the list after being deleted.

Backup and Restore

These tests checks whether backup and restore features are working correctly.

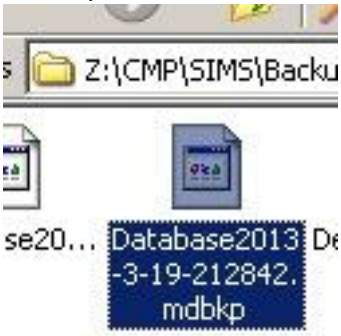
Backup:



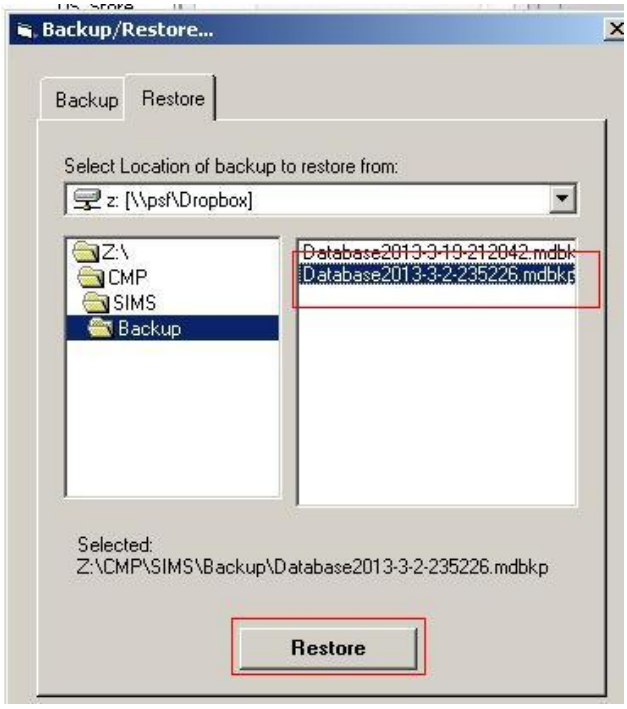
A location is selected, and the backup button is pressed. A confirmation dialog will appear asking the user to confirm the backup process.

After the process has been finished, a message box appears informing the user of the successful backup.

The backup file is now found at the selected location (In this case, /Backup folder) named with the backup date and timestamp.



Restore:

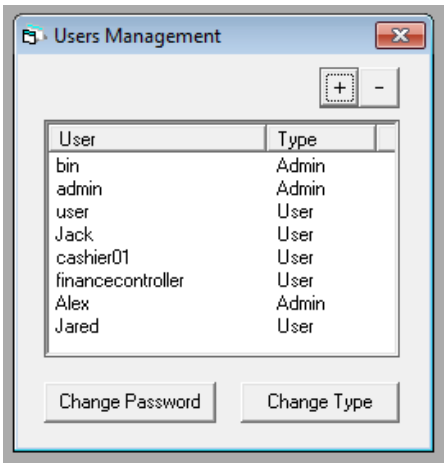


A backup file is selected from the backup location, and the Restore button is clicked. A confirmation dialog appears asking if the user wants to restore.

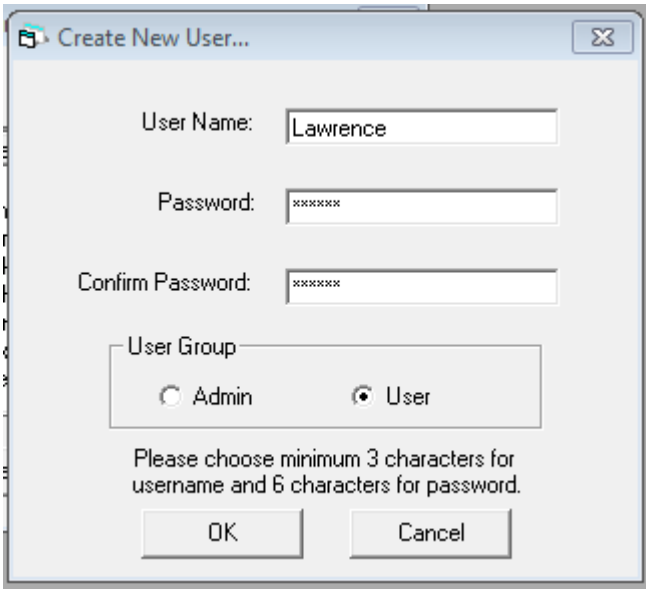
After the restoration process completes, the program prompts the user that the system is restarting.

Adding, Changing and Removing Users

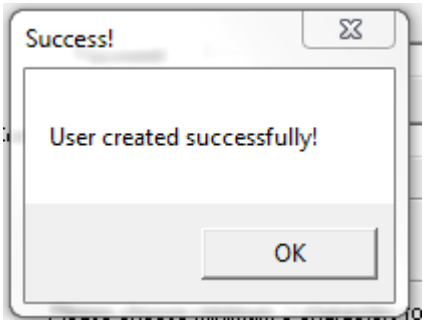
This test checks whether users can be added, changed and removed correctly.



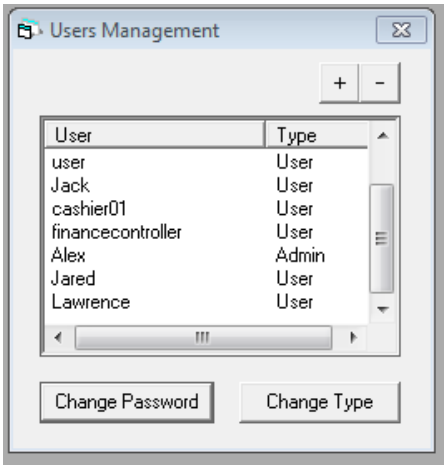
When the "+" button is pressed, a Create New User form appears.



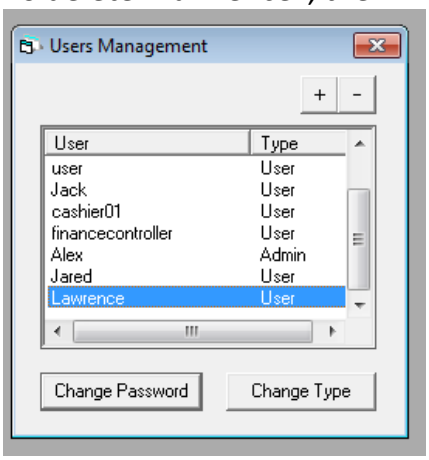
After filling in a valid username and password, the OK button is pressed and a dialog box will appear to inform the user of successful creation of the new user.



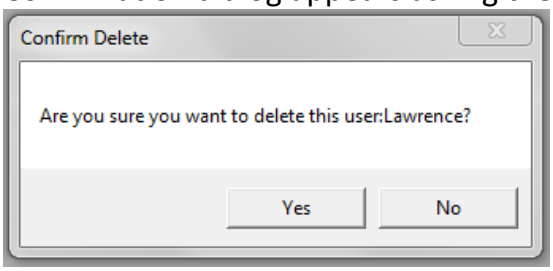
After: “Lawrence” is now in the list of users.



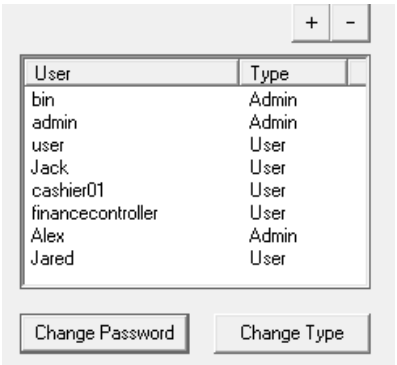
To delete “Lawrence”, the “-“ button at top right is pressed.



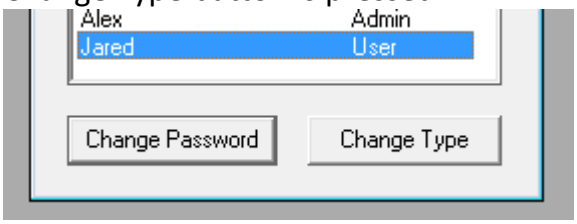
Confirmation dialog appears asking the user to confirm delete of “Lawrence”.



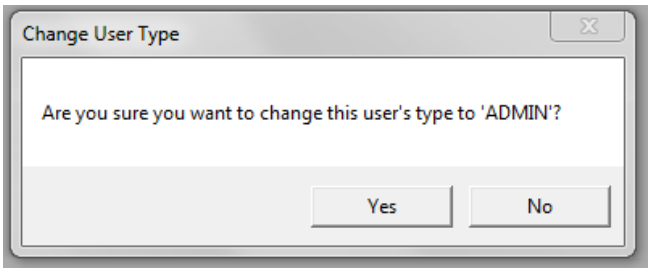
After: “Lawrence” has been deleted successfully from the database.



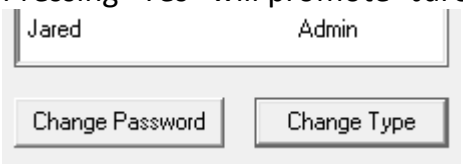
To change the type of “Jared”, “Jared” is selected from the list, and the Change Type button is pressed.



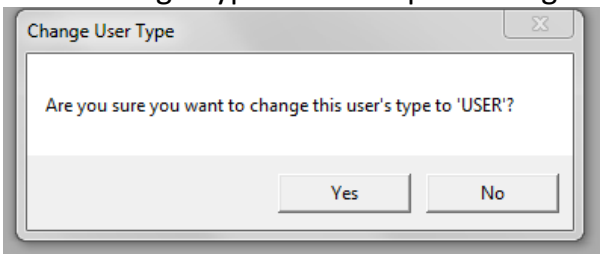
A confirmation dialog appears confirming the elevation of “Jared” to an administrator.



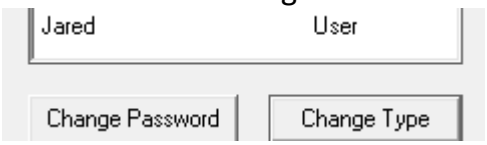
Pressing “Yes” will promote “Jared” to an admin as shown below.



If the Change Type button is pressed again for “Jared” selected,



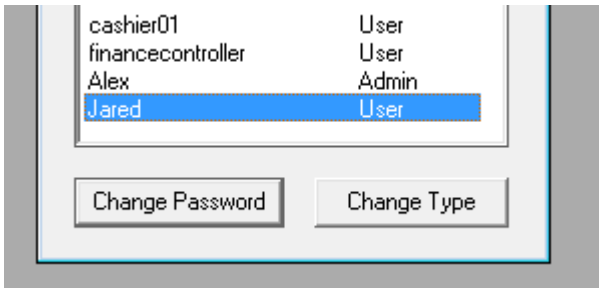
“Jared” will be changed back to a normal user.



Change user password

This test checks whether user passwords can be changed properly.

"Jared" is selected from the user list, and the *Change Password* button is pressed.



A change password dialog appears.

Valid data: Correct old password and matching new passwords (with length >6) are entered.

A message box appears informing the admin that the password for "Jared" has been changed.

Invalid Data: Incorrect old password

The status will display “Incorrect old password” to the user.

Invalid Data: New Password does not match the Confirm Password

The status will display “The confirm password does not match the first one.”

Invalid Data: New password is less than 6 characters long.

The status will display “The new password must be more than 6 characters.”

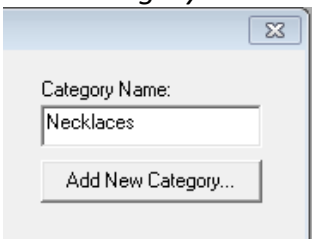
Invalid Data: New password is the same as the old password

The status will display “New password cannot be the same as the old one.”

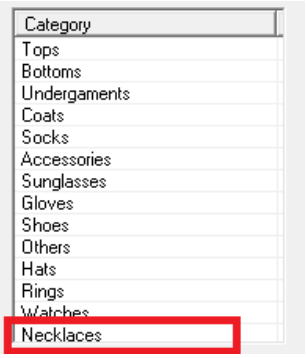
Managing categories

This test checks whether categories can be added and removed as intended.

Valid Data: A valid category name (e.g. “Necklaces”) are inputted, and the *Add New Category…* button is pressed.

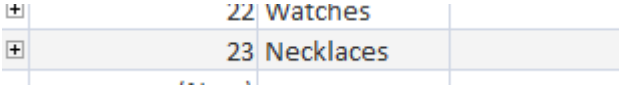


If the category is successfully added, it will appear in the list.



A message will also be shown to inform the user of successful entry.

It will also be saved to Database.mdb file.

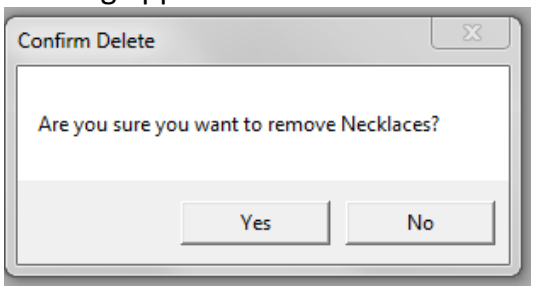


Invalid Data: A category name with length 100 is inputted.

An error message will appear asking the user to limit the category name to less than 30 characters length.

Deleting a category: The entry “Necklaces” is selected from the list, then press the Delete button.

A dialog appears to confirm that “Necklace” will be deleted.



After: “Necklace” has been deleted.

Implementations

After all the tests that has been carried out and it's been decided to utilize the new system, a plan of implementation must be drawn out in order to transition the business from using the old system to the new system. The few ways that system handover can be carried out has been evaluated during the discussion between the client and the system analyst, including:

1. **Parallel conversion**
   This method of conversion runs both the old system and the new system side by side, performing the same operation every time for a period of time (typically a few months), until it is confirmed that the new system is free from faults and there are no major problems. This method is the safest, because if there is indeed a fault with the new system, no data are lost from the moment the new system is implemented and business operations can still carry on as usual. Staff training can also be carried out during this period. But the major caveat to this method is that it is very expensive to carry out, because there is a need to keep two systems running at the same time, and it is also tedious because every operation needs to be performed twice.
2. **Pilot conversion**
   In this method, the new system is only implemented in a smaller group/department/area of the organization, while running alongside the old system until the new system is confirmed to be free from major problems. This method allows for only a small part of the organization to be affected if there is indeed a problem with the new system. However, it takes a long time to transition over the whole organization.
3. **Direct changeover**
   In direct changeover, the old system is straightaway converted to the new system and all data are transitioned over directly and immediately. The new system completely takes over the old system in operation. This method is the cheapest, quickest and easiest to implement; however it is very risky as there may still be major problems with the system that may be unforeseen, all data may be lost in case of failure.
4. **Phase transition**
   Only a few key parts of the new system are implemented, with the remaining components using the old system. This method means only a small part of the system will fail if there is a problem with the new system. Staff training can also be carried out part by part this way. However, the system must be able to support being split into smaller separate parts in order to work. Aside from that, the system in smaller parts may also not perform the same as when performed together as a whole systems package.

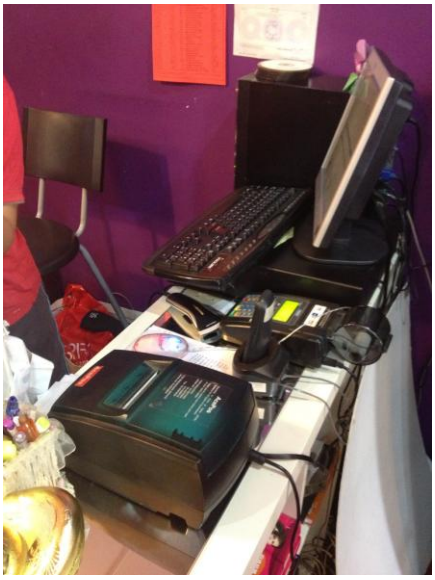
System Changeover Method

To select the most appropriate system changeover method, a discussion has been carried out with Mrs Lim, the owner of Sense Boutique. Various aspects have been taken into consideration including cost, time taken, staff training and safety of the data.

Firstly, we considered direct changeover, but since the new system has not been tested thoroughly in real world usage, there are still risks to be taken to implement such a drastic change to how the business operates and stock records may be lost if there are problems. Then we evaluated pilot conversion, but since it is only effective for large organisations with a lot of departments and teams, this idea is abandoned. Thirdly, we talked about phased conversion, but the old and new systems are different enough and cannot be broken down into smaller modules to be partly replaced.

Lastly, it's been decided that a parallel conversion should be carried out. The old method of manual filing shall be carried out simultaneously alongside the new computerized system for three months, while observing the performance of the new system. Although this is tedious, it does not incur much extra cost because the old system is manual labour based (to file records in paper documents), which means it does not consume extra electricity expenditure to keep both systems running. Aside from that, Mrs. Lim has stated that she doesn't mind paying a little more for the first few months if it pays off in the long run.

If the new system does not show any significant problem and is able to meet the business demands, then the old system shall be deprecated in favour of fully transitioning to the new one.



Why parallel conversion is chosen for this system:

1. **Minimising risk and possible data loss**
   Because new stock records are input into both the old manual system and the new computerized system at once, in event of the new system failing to perform as expected, there is no loss of data. Aside from that, business operations would also not be interrupted if there is a problem with the new system, because the employees can just reuse the old system and the new system can be disabled while debugging process are being carried out.

2. **Providing a period where staff can slowly get used to the new system.**
   It is important to not rush the conversion process, as employees may still require time to get used to the new system, or they may require time for training in order to get to know about how to use the new system. The employees must be comfortable with the new system before operations can be carried out efficiently.

3. **Ensure that the system is stable before it fully takes over the operations.**
   As mentioned previously, the new system has not been tested "in the wild", i.e. tested in normal daily business operations to test whether the system can perform adequately under load. Hence, there might still be major or minor bugs that can cause problems for the business. Therefore, a parallel conversion can facilitate testing and observe how the system performs over a period of time. If the system still performs well after a period of time, then it should be fit to replace the old system.

System Changeover

The installation of the system consists of several procedures. First of all, a ready built desktop computer will be purchased from a manufacturer (or self-built using custom components sourced from various suppliers), and peripherals such as barcode scanner, printer, modem, routers etc are purchased from various computer hardware dealers. Then the hardware were set up and placed in correct order, and an internet connection was subscribed from a local ISP and configured accordingly. The analyst will then configure the software used by the system, such as installing Windows operating system, configuring various settings suitable for the system, installs various programs such as Microsoft Office, and finally the inventory management system software is installed into the computer along with the documentation files. Then employee training will be carried out and old data will be converted into the new format.

Employee's Training

After the system has been installed successfully, the employees must undergo training in order to familiarize themselves with the system, get used to the new system and know their way around the various features of the new system, in order to maximize the potential benefits of the new system. Aside from that, training will also provide them an avenue to ask questions or seek clarifications regarding the operations of the new system.

The training class will be mainly about the operation of the new system and should offer the employees an idea on how to utilize the new system effectively. On the other hand, the training will also offer a briefing on how should the employees help in the initial conversion of the data from the old system to the new system.

Because of the small scale and to avoid taking up business operating hours, the training will be conducted on 3 selected business days for about an hour each, from 10.30am to 11.30 am, right before the shop opens, for a total of 3 hours of training.

The first session and third session will involve all the employees as well as the business owner, and will offer lessons in the general operations of the system. Meanwhile, the second session will be exclusively for the business owner only, offering lessons on how to use administrative functions offered by the system.

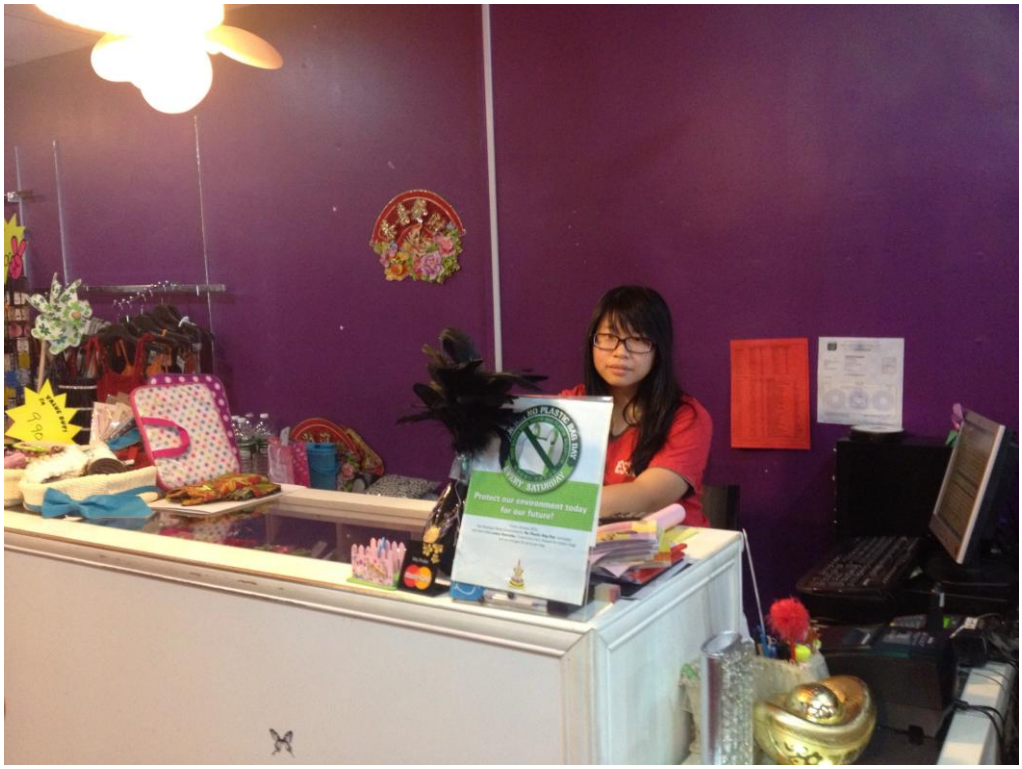
The class will be conducted interactively, with the system analyst demonstrating each feature and explaining their usage, and then the employees are free to ask questions, as well as hands on trying it out.

- 1st Day
    - Briefing on information about the new system
    - Splash screen and its error messages
    - How to login, logout and exit
    - Guide to the main GUI, menus and the interface
    - Inventory List and its buttons
        - Adding, editing and removing records
        - Searching through the records
        - Checking stock levels
        - Export and print the list
    - Recording sales and purchases
    - Suppliers record management
        - Adding, editing, removing suppliers
        - Accessing supplier records
    - Common error messages

- 2nd Day (Owner/admin only)
    - Administrative features
    - Managing categories
    - Viewing the transaction journal
        - Print or export the transaction report
    - Users management
        - Adding, removing users
        - Changing their password
        - Change user types
    - Backup and restore

- 3rd Day
    - Brief revision on the 1st class lessons
    - Help and support documentations built in
    - Briefing on the conversion process of old data to new data
    - Test run

At the end of the training sessions, the employees should be able to know where to access for each feature, as well as various aspects of the new system, while the shop owner should be able to understand how to administer and manage this system effectively.



File Conversion Process

Since the older data were filed in the previous manual based filing system, in order to utilize the new system, it is a must to complete a one-time manual input of all previous data into the new computerized system in order to change them into digital format.

Although this might also take some effort to be completed, the business can soon reap the benefits after the initial setup is completed. The following items would have to be inputted into the database:

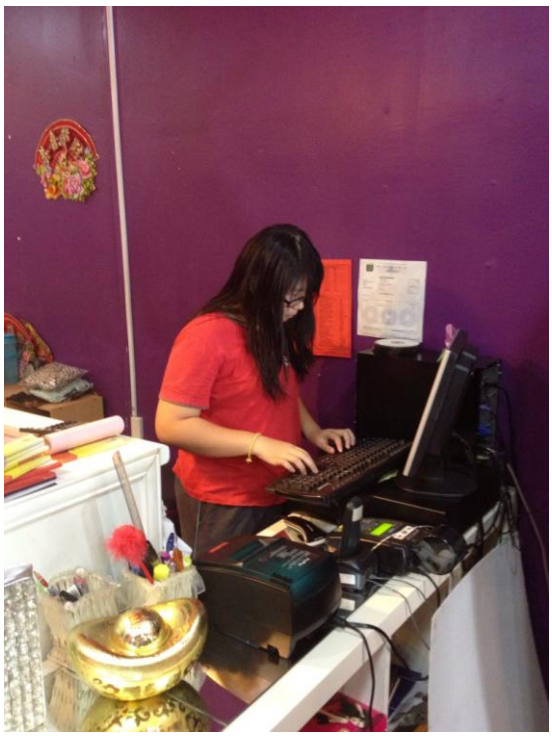
- Stock records
    - Through the use of the *Add New Items* dialog
    - Product ID, Name, Quantity on hand, cost, price, color, size, description, category, date added

- Suppliers
    - Through the *Suppliers Management* dialog
    - Name, Company, Email, Telephone, Address

- Sale transaction and stock purchases
    - Through the use of *Add Sales* and *Add Purchases* dialog

User Acceptance

After the detailed design, specification and a working sample of the system program has been completed, Mrs. Lim and her workers were asked to provide feedback and evaluation on the draft before it is finalized and agreed to implementation. They were also asked to test the working sample of the system program, to collect more user feedback and as a form of beta testing from the users.



The user is testing the new system.

Their responses has been mostly positive aside from a few minor changes that they would like to see implemented, such as change of some words used in the system (e.g from Add Stock to Add Purchases), and a few minor questions about the implementation strategy (Parallel vs. Direct). After further discussion and amendments to the implementation strategy, Mrs. Lim has approved the implementation strategy and the system can now be implemented. The workers were also very accepting of the new system and are willing to learn how to use it as long as it boosts their productivity and makes their work more efficient.

A copy of the letter of acceptance can be found in the next page.

E) Evaluation

Evaluation of objectives

After the system is implemented successfully, it was evaluated and compared against the original objectives to identify the degree of which the original objectives were achieved. The following original objectives were compared:

- To make the process of managing a large stock inventory more efficient
  Yes, this objective is met. The system is able to handle a large amount of records and lay them out in a neat and organised list for easy at-a-glance view by the user. The user is now also able to easily add, edit and remove stock records. In contrast to the old paper based document system, this electronic system is much more efficient.

- To increase productivity
  It is not certain that this objective is met, although currently the user is able to manage stock records and inventory more efficiently, it is not certain that in the long run, the system may become slower or the workers may be distracted by the internet usage. On the other hand, the parallel conversion method also introduces extra work to the workers in the short term.

- To reduce input errors
  Yes, this objective is met. The system is able to detect incorrect format inputs or accidental input of inappropriate data such as an alphabet into a numerical field, and notify the user of such error. Also, with the clear layout of the system and various input forms, user errors are less likely to happen.

- To save time
  Yes, this objective is met. The user is able to locate a certain record easily by simply typing in the search criteria, and is able to manage inventory easily as well. Hence, the time saved can be better used to serve customers.

- To have more reliable data redundancy
  Yes, this objective is met. In this system, a backup and restore system is implemented, and database can be backed up to any safe location or external media easily. This allows the even better data recovery in case of accidents. The system will also automatically detect database errors and offer to restore data easily.

- To have better security measures
  Yes, this objective is met. A login security system is implemented on this system using user names and passwords, and there are separate tiers of user privileges such as Admin and Users. This discourages any unauthorized access to the system and keeps confidential business data such as suppliers' information safe.

- To reduce costs and minimise risks
  It is not certain that this objective is met, because in terms of monetary savings, it still remains to be seen that the new system is able to successfully save costs for the business. The new system incurs extra costs such as purchasing a new computer, internet subscription and extra electricity bills. However, it is possible that in the long term, the extra efficiency brought by the new system is able to help the business recover these costs and even bring in more revenue as a result. Risks are minimized due to the use of better backup mechanism and security measures.

- To aid in making better business decisions
  Yes, this objective is met. The new system is able to automatically generate and keep track of transaction reports, as well as calculate total costs/gross profits easily. The owner is able to take note of such financial figures and should be able to help make business decisions.

Out of the eight original objectives, six are met successfully with the new system, while effectiveness of the other two still remains to be seen in the long term.

Client's and User's Response

After the system is successfully implemented, the business resumed operation using the new system. After about one week of usage, the client and users were surveyed and asked how the system worked for them so far. The response received has been pretty positive overall, and users are currently adapting well to the methods of the new system.

According to the users, the new system is quite easy to learn and use, and the overall interface is quite user friendly, has an easy learning curve, with helpful hints placed all over various system user interfaces, as well as a detailed user documentation accessible through a shortcut. The interface is also designed and layout neatly. They also reported the system is running stable and no major quirks so far have been discovered.

With the new system, the users also find that there are less hassle in dealing with various stock records and searching for them to update an item. And they now have more free time to serve and communicate with the customers to meet their needs. They also noted that the customers also noticed their new system and were impressed about it. They think that the business reputation is now better than before.

Lastly, they also find that their office is now less messy and more organized than before, due to a significant reduction in paper documents as well as stock records.

The client also reports that the system has actually made their lives easier because of the easier input methods and the quick searching of the records. She also reports that the new system's backup and restore system allowed her to keep her data safe and is easy to use at the same time. She also commented on the supplier's management tool allowing her to keep track of suppliers, and then get their addresses or phone numbers in order to restock easily, being a huge timesaver. She also finds that she is able to decide whether to import more stock of this item more easily due to the transaction report allowing her to keep track of which item has been sold.

Below is a table of the results of the survey, showing the mean score (average score) rated for each of the questions by the client and the three employees.

No	Question	Mean Rating
1	Do you think this system makes your work more efficient?	4.5
2	Do you think this system has helped to prevent input mistakes?	3.75
3	Do you think this system is secure and safe enough to prevent unauthorized access?	4.75
4	Do you think this system is easy enough to use and user friendly?	4.25
5	Do you think this system has a good backup system in place?	5
6	Do you think this system will help save costs?	4
7	What do you think about the stability of the system? Does it crash often?	4.5
8	What do you think about the features of the system?	4
9	Do you think this system is able to meet the business's future expansion and needs?	4
10	What would you, as a user, rate the overall system as a whole?	4.5

Average Score: 4.325/5

A sample of the survey form can be found in the next page.