Title Page

DEVELOPING A BUSINESS MANAGEMENT APPLICATION USING OBJECT ORIENTED PROGRAMMING TECHNIQUES

Case Study: Homewares Lighting Systems Ltd.

By

NZEIH CHUKWUEMEKA
PGD/MCS/99/2000/934

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS/COMPUTER SCIENCE
FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA.

IN PARTIAL FUFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE POST GRADUATE DIPLOMA IN COMPUTER SCIENCE

SEPTEMBER, 2001.

CERTIFICATION

This project has been read and approved as meeting the requirements of the award of Post Graduate Diploma in Computer Science, Department of Mathematics and Computer Science, Federal University of Technology, Minna.

Mr. L.N. Ezeakor Supervisor	Date
Dr. S. A. Reju Head of Department.	Date
External Examiner	Date

DEDICATION

This work is dedicated to God Almighty and all my friends for their well wishes.

ACKNOWLEDGEMENT

I wish to acknowledge the efforts of all my lecturers at the Federal University of Technology, Minna for their sincere effort in seeing that I get the required knowledge in Computer Science and Information technology at its highest level.

I am particularly grateful to my Supervisor Mr. L.N. Ezeakor for his guidance.

Finally, I wish to thank my friends and course mates for their company and encouragements.

TABLE OF CONTENTS

ABSTRACT

One of the major areas of Information Technology deployment is business management. Proper application of Information Technology tools has helped so many businesses in enhancing their management and operational activities.

Internal and External constraints exist that acts as impediments to effective operations of business organizations transcends both small and large businesses. In resolving these constraints, I.T. tool and techniques have been applied with positive results. I.T. solutions to Business Management needs the appropriate software. Thus developing the required software is a major step in achieving the objective of offering solutions to Business needs using Information technology. Microsoft Visual Basic is one of the most flexible and powerful Rapid Application Development tool available today. Its use for Software development usually results in Software with the Windows feel and look.

Therefore, in developing a business management application for transaction processing, Microsoft Visual Basic would help in developing a software that meet the need for which it was designed.

Chapter One

1.1 Introduction

Modern Business Management recognizes the presence of internal and external constrains that needs to be properly attended to in order to achieve progress. These constraints comes in the form of record management, transaction processing, customer relations management and the coordination of other business activities. One way these constraints have been tackled head-on is by office automation (Office System). Office Systems refers to equipment used to create, store, process, or communicate information in a business environment. In today's world the fulcrum of an office system is the PC (microcomputer) and its associated software. It is well known that hardwares do not function in isolation. In a business environment the benefit of computing and its applications cannot be appreciated except there are software's specifically tailored to offer business solutions. The major tools needed to develop modern business application are object oriented programming (OOP) tools. OOP tools dictate the present trend in computer programming. This is because they generate software's that are interactive, user friendly and efficient.

Apart from the perceived gains of computer usage in business management, other intrinsic benefits are derivable from using business software as a management tool; it enhances productivity, promotes business process reengineering and supports and provides a viable information system.

Programming is part of an esoteric world where logic is sacred. Even if you understand exactly why a program works, there is still a magical element involved. Things appear and disappear. Objects materialize, and then dematerialize. They do so according to strictly defined logical rules; but still, there is the fact that things appear and disappear right before our eyes.

To be a good programmer, one has to be an insider to program development tool. A programmer to study arcane material, sit up over it and ponder its meaning, seeking to understand its mysteries. Many people never understand the subtleties of programming. They don't ever penetrate to the inner mysteries of this challenging field.

Some products seem to be effective at capturing the essence of the beautiful, mysterious logic that underlies the world of programming. Products such as C++, Visual Basic, Visual J++ and other Rapid Application Development tool have effectively captured and simplified the steps in development effective programs by providing a powerful sets of programming tools.

1.2 Statement of Problem

As we traverse through the years it would be notice that computing needs of business changes. So also the type of software and computing requirement that would be needed to solve these needs. Most business requires effective computer software's to acts as a management tool. These software usually would be expected to incorporate a database that would serve as repository of information into which large volume of data would be stored and retrieved as required.

Therefore database management systems are being called on to provide a higher level of database management. No longer will databases manage data; they must manage information and be the knowledge centers of the enterprise. To accomplish this, the database must be extended to;

- Provide a higher level of information integration.
- Stores and retrieve all types of data.

Applications that require database support are quickly extending beyond traditional data processing into sophisticated office automation software. These applications have complex data structuring needs, significantly different data accessing patterns and special performance requirements. *Conventional programming methodologies are not necessarily appropriate for these applications and conventional data management system may not be appropriate for managing their data.*

Business management Applications deals with a hierarchical structure of information organization. Database access for these applications is typically a directed graph structure rather than an ad hoc query. In trying to manipulate such complex data, a programmer writes code to handle these. The Object Oriented Programming tool is the therefore the best bet in solving the complexities for handling such data management requirement.

1.3 Objectives of Study

- To highlight the benefits of object oriented programming tools in developing modern applications.
- Illustrate the Software development process using a particular Object Oriented Programming Tool.

- To develop a business management software using Ms Visual Basic.

1.4 Scope of Work

Object Oriented Programming and its principles would be discussed in detail, the actual software development process would be done using Microsoft Visual Basic. Hence, the study would be restricted further to Visual Basic components and their applications in software development. A business management software would be developed to handle the information processing required of a organization; Homeware Lightening Systems Ltd.

1.5 Method of Data Collection

The data collection methods used in this study were by Interviews and Studying procedural Manual. In designing the software for the system, the personnel that uses the manual system were interviewed, the forms they used studies. This is to enable the design

1.5 Definition of Terms

ActiveX: Microsoft's brand name for the technologies that enable interoperability using the Component Object Model (COM).

API: Application programming interface. The set of commands that an application uses to request and carry out lower-level services performed by a computer's operating system.

AGP: Accelerated Graphic Port; a new kind of Video RAM standard.

Polymorphism: In an object-oriented programming language, the ability to redefine a routine in a derived class (a class that inherited its data structures and routines from another class). Polymorphism allows the programmer to define a base class that includes routines that perform standard operations on groups of related objects, without regard to the exact type of each object. The programmer can redefine the routines, taking into account the type of the object, in the derived classes for each of the types.

Index: In Visual Basic, a number that identifies an element in an array, control array, or collection. In data access, a dynamic cross-reference of one or more table data fields (columns) that permits faster retrieval of specific records (rows) from a table. As records are added, changed, or deleted, the database management system automatically updates the index to reflect the changes.

Method A procedure that acts on an object.

Module A set of declarations followed by procedures.

Object: A combination of code and data that can be treated as a unit, for example, a control, form, or application component. Each object is defined by a class.

Object-Oriented Programming: In contrast with procedural programming, involves the use of both object-oriented design and an object-oriented programming language. Instead of consisting of sets of data loosely coupled to many different procedures, object-oriented programs consist of software modules called objects that encapsulate both data and processing while hiding their inner complexities from programmers and hence from other objects.

ODBC (Open Database Connectivity): A standard protocol that permits applications to connect to a variety of external database servers or files. ODBC drivers used by the ODBC driver manager permit access to SQL Server and several other data sources, including text files and Microsoft Excel spreadsheets. The ODBC application programming interface (API) may also be used to access ODBC drivers and the databases they connect to without using the Microsoft Jet database engine.

Binding: The process of putting an object into the running state so that operations supplied by the object's application (such as edit or play) can be invoked. The type of binding determines the speed with which an object's methods are accessed using the object variable.

Procedure: A named sequence of statements executed as a unit. For example, Function, Property, and Sub are types of procedures. A procedure name is always defined at module level. All executable code must be contained in a procedure. Procedures can't be nested within other procedures.

Chapter Two

2.1 The Basics of Programming Languages

Programming Language, in computer science is an artificial language used to write a sequence of instructions (a computer program) that can be run by a computer. Similar to natural languages, such as English, programming languages have a vocabulary, grammar, and syntax. However, natural languages are not suited for programming computers because they are ambiguous, meaning that their vocabulary and grammatical structure may be interpreted in multiple ways. The languages used to program computers must have simple logical structures, and the rules for their grammar, spelling, and punctuation must be precise.

Programming languages vary greatly in their sophistication and in their degree of versatility. Some programming languages are written to address a particular kind of computing problem or for use on a particular model of computer system. For instance, programming languages such as FORTRAN and COBOL were written to solve certain general types of programming problems—FORTRAN for scientific applications, and COBOL for business applications. Although these languages were designed to address specific categories of computer problems, they are highly portable, meaning that they may be used to program many types of computers. Other languages, such as machine languages, are designed to be used by one specific model of computer system, or even by one specific computer in certain research applications. The most commonly used programming languages are highly portable and can be used to effectively solve diverse types of computing problems. Languages like C, PASCAL, and BASIC fall into this category.

High-level languages are commonly classified as

- procedure-oriented,
- functional,
- object-oriented, or logic languages.

The most common high-level languages today are procedure-oriented languages. In these languages, one or more related blocks of statements that perform some complete function are grouped together into a program module, or procedure, and given a name such as "procedure A." If the same sequence of operations is needed elsewhere in the program, a simple statement can be used to refer back to the procedure. In essence, a procedure is just a mini-program. A large program can be constructed by grouping together procedures that perform different tasks. Procedural languages allow programs to be shorter and easier for the computer to read, but they require the programmer to design each procedure to be general enough to be used in different situations.

Functional languages treat procedures like mathematical functions and allow them to be processed like any other data in a program. This allows a much higher and more rigorous level of program construction. Functional languages also allow variables—symbols for data that can be specified and changed by the user as the program is running—to be given values only once. This simplifies programming by reducing the need to be concerned with the exact order of statement execution, since a variable does not have to be redeclared, or restated, each time it is used in a program statement. Many of the ideas from functional languages have become key parts of many modern procedural languages.

Object-oriented languages are outgrowths of functional languages. In object-oriented languages, the code used to write the program and the data processed by the program are grouped together into units called objects. Objects are further grouped into classes, which define the attributes objects must have. A simple example of a class is the class Book. Objects within this class might be Novel and Short Story. Objects also have certain functions associated with them, called methods. The computer accesses an object through the use of one of the object's methods. The method performs some action to the data in the object and returns this value to the computer. Classes of objects can also be further grouped into hierarchies, in which objects of one class can inherit methods from another class. The structure provided in object-oriented languages makes them very useful for complicated programming tasks.

Logic languages use logic as their mathematical base. A logic program consists of sets of facts and if-then rules, which specify how one set of facts may be deduced from others, for example:

If the statement X is true, then the statement Y is false.

In the execution of such a program, an input statement can be logically deduced from other statements in the program. Many artificial intelligence programs are written in such languages.

2.2 OOP Principles

OOP is a disciplined programming style that incorporates three characteristics; *encapsulation, inheritance and dynamic binding.* These characteristics differentiate OOP from traditional programming models in which data has a type and structure, distinct from the program code and is processed sequentially. OOP builds on the concepts of reuse through the development and maintenance of class libraries of objects available for use and marinating applications.

- Encapsulation joins procedures and data to create an object, so that only the procedures are visible to the user, data is hidden from view. The purpose of encapsulation is to mask the complexity of the data and the internal working of the object. Only the procedures (methods) are visible to the outside world.
- Inheritance passes attributes to dependent objects, called descendants or receives attributes from objects called ancestors on which the object depends.
- Dynamic binding is the process whereby linking occurs at program execution time. All objects are program execution. For example in a stock management application, the function called program trading can sell or buy, depending on a large range of economic variables that define the current state. These variables are transparent to the user who invokes the trade process.
- Class library is mature, tested reusable codes that provides application enabling code such as help management, error recovery, function key support, navigation logic and cursor management. The class library concept is inherent to the OOP concept and in combination with the standards and training fundamentals - is inherent to the productivity and error reduction encountered in project in which OOP tools are used.

Object Oriented programming is most effective when reusable components can be cut and pasted to create a skeleton application. Into this skeleton the custom business logic for this function is embedded. It is essential that the standard components use dynamic binding so that changes can be made and applied to all applications in the environment. This provides one of the major maintenance productivity advantages.

- Objects
- Encapsulation and message passing
- Classes
- Libraries
- Inheritance
- Access modifiers

Objects

The fundamental unit in object-oriented programming is the object. Languages that follow object-oriented concepts describe the interaction among objects. All objects have a state and a behavior. The state of an object pertains to data elements and their associated values. Everything the object knows about these elements and values describes the state of the object. Data elements associated with objects are called *instance variables*.

The behavior of an object depends on the actions the object can perform on the instance variables defined within the object. In procedural programming, such a construct would be called a *function*. In object-oriented terminology, this construct is called a *method*. A method belongs to the class it is a member of, and you use a method when you need to perform a specific action more than once.

Thus, the state of an object depends on the things the object knows, and the behavior of the object depends on the actions the object can perform. If a software object that models a television is created, the object would have variables describing the television's current state, such as it is on, the current channel setting is 8, the current volume setting is 23, and there is no input coming from the remote control. The object would also have methods that describe the permissible actions, such as turn the television on or off, change the channel, change the volume, and accept input from the remote control.

Encapsulation and Message Passing

Objects encapsulate instance variables and related methods into a single, identifiable unit. Therefore, objects are easy to reuse, update, and maintain. A programmer can quickly and easily do the following:

- Pinpoint the necessary input to the object and the output from the object
- Find variable dependencies
- Isolate the effects of changes
- Make updates as necessary
- Create subclasses based on the original object

Objects are as dynamic as you make them. An object can invoke one or more methods to accomplish a task. A user initiates a method by passing a message to an object. A message must contain the name of the object you are sending the message to, the names of the methods to perform, and the values needed by those methods. The object receiving the message uses this information to invoke the appropriate methods with the specified values.

The benefit of encapsulation of instance variables and methods is that the programmer can send messages to any object without having to know how the object works. All he needs to know is what values a method will accept. Therefore, the software object describing the television could be extremely complex, but all that needs to be done is for the programmer or the end user have to know to use the television is how to press the appropriate buttons on the remote control. The press of a button on the remote control sends a message to the television's software object, telling it which method to perform and the new input values for the method.

Classes

Classes encapsulate objects. A single class can be used to instantiate multiple objects. This means that you can have many active objects or instances of a class. The object describing the functions of your television is an instance of a class of objects called television. Keep in mind that each object within a class retains its own states and behaviors. By encapsulating objects within a class structure, a programmer can group sets of objects by type.

Libraries

In C++ and other programming languages, a collection of related classes or functions is called a *library*. Java puts a twist on the concept of libraries by using the term *package* to describe a collection of related classes. Just as classes encapsulate objects, packages encapsulate classes in Java.

Inheritance

Inheritance is a powerful aspect of object-oriented programming that allows codes to be reused. and extend the functionality of existing classes. If a class is created to draw a shaded rectangle on the screen, you could extend the class to move the rectangle to specific locations on the screen without having to rewrite the original class. A programmer could also extend the class for the shaded rectangle to display a series of user-selectable rectangles. In either case, the new class would inherit the methods that created the shaded rectangle and then extend the methods to perform the appropriate action.

Using this aspect of object-oriented programming, a new class can be created that inherits the functionality of an existing class. Then functions can be extended to form part of the old class in ways that suit your current needs. The television class could have subclasses for black-and-white televisions, color televisions, and home-theater-style televisions. The new television subclass is not limited by the instance variables or methods of the superclass and can include instance variables and methods not defined in the superclass. The new subclass can also override inherited methods.

Access Modifiers

In object-oriented programming, access to methods and variables is controlled through access modifiers. The Java programming language defines four levels of access controls:

- Private methods and variables
- Protected methods and variables
- Friendly methods and variables
- Public methods and variables

Private Methods and Variables

Methods and variables that are controlled by an associated object and are not accessible to objects of different classes are generally considered to be *private*. The advantage of this is that only objects in a particular class can access the methods or variables without limitation. Java's private methods and variables are likewise accessible only by objects within the same class.

Protected Methods and Variables

Methods and variables that are controlled by an associated object and are accessible to objects in the current class or a subclass of the current class are generally considered to be *protected*. The advantage of this is that only objects in specific classes can access the variables without limitation. Java's protected methods and variables are likewise accessible only by methods in the same class or subclass.

Friendly Methods and Variables

Methods and variables that are accessible to other objects in most circumstances are considered to be *friendly*. By default, methods and variables you declare in Java are assumed to be friendly and are accessible by any class and objects in the same package. The advantage of this is that objects in a particular package (generally a set of related classes) can access each other without limitation.

Public Methods and Variables

Methods and variables that are accessible to all objects, even those outside the current class and package, are considered to be *public*. Java's public methods and variables are accessible by any object or class. Therefore, public methods and variables can be accessed without limitation.

2.3 Visual Tools and Object Programming

Modern Object Oriented Programming has been enhanced by the use of Visual Development Tools. The use of these tool allows application to be created with reduced amount of programming. That is, a good percentage of the job to be done is handled by using Visual designs and graphics to which underlying codes are attached.

Visual Tools have made programming quite simple. Task that hitherto requires a great deal of programming to be achieved can be created by first creating the graphical interface by using the tools provided the compiler and then associating program codes to them.

Visual tools allows the creation of application that has the *Windows feel and look*. That is, applications created with Visual tools are fully compatible with the Microsoft Windows Operating System and they also share common controls, interfaces and dialog boxes with the Windows O/S. Thus the use of Visual Tools is also referred to as Windows Programming. Popular examples of Visual Programming Tools are Microsoft Visual Studio that incorporates Visual Basic, Visual C++ Visual FoxPro and Visual Interdev.

2.4 Event Driven Models and Interactive Development

In traditional or "procedural" applications, the application itself controls which portions of code execute and in what sequence. Execution starts with the first line of code and follows a predefined path through the application, calling procedures as needed.

In an event-driven application, the code doesn't follow a predetermined path — it executes different code sections in response to events. Events can be triggered by the user's actions, by messages from the system or other applications, or even from the application itself. The sequence of these events determines the sequence in which the code executes, thus the path through the application's code differs each time the program runs.

Because the sequence of events cannot be predicted,, the code must make certain assumptions about the "state of the world" when it executes. When you make assumptions (for example, that an entry field must contain a value before running a procedure to process that value), you should structure your application in such a way as to make sure that the assumption will always be valid (for example, disabling the command button that starts the procedure until the entry field contains a value).

Your code can also trigger events during execution. For example, programmatically changing the text in a text box cause the text box's Change event to occur. This would cause the code (if any) contained in the Change event to execute. If it is assumed that this event would only be triggered by user interaction, you might see unexpected results. It is for this reason that it is important to understand the event-driven model and keep it in mind when designing your application.

The traditional application development process can be broken into three distinct steps: writing, compiling, and testing code. Unlike traditional languages, Visual Basic uses an interactive approach to development, blurring the distinction between the three steps.

With most languages, if a mistake is made in writing your code, the error is caught by the compiler when you start to compile your application. The programmer must then find and fix the error and begin the compile cycle again, repeating the process for each error found. Visual Basic interprets your code as you enter it, catching and highlighting most syntax or spelling errors on the fly. It's almost like having an expert watching over the programmers shoulder as he enters the code.

In addition to catching errors on the fly, Visual Basic also partially compiles the code as it is entered. When you are ready to run and test your application, there is only a brief delay to finish compiling. If the compiler finds an error, it is highlighted in the code. The error can be fixed and continue compiling without having to start over.

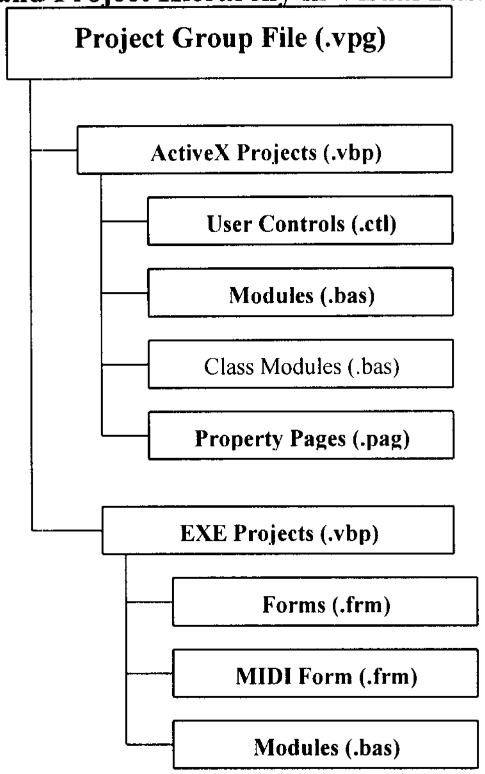
2.5 Using Visual Basic for OOP

The Basic programming language began as a *procedural* language, based on variables, function calls, and statements. It is evolving toward an *object-oriented* language, based on objects, properties, methods, and collections.

Visual Basic has always had support for objects, even though you couldn't always create objects in VB. The ability to create objects from classes was added in VB 4.0, while the ability to create an ActiveX object is new in VB 5.0. and VB 6.0

While purists would argue that Visual Basic isn't a fully object-oriented language (mainly because it doesn't have all the object-oriented features of C++, , it has become more object oriented over time.

2.5 File and Project Hierarchy in Visual Basic



2.6 Designing Applications using Visual Basic

There are three main steps in creating application in Visual basic.

1. Create the interface.
2. Set Properties
3. Write codes.

2.6.1 Creating Interface.

Forms are the foundation for creating the interface of an application. Forms are used to add windows and dialog box to an application. They are also used as containers for items that are not a visible part of the application interface. For example, a form in an application can serve as a container for graphics that is to be displayed on other forms.

The first steps in building an application are to create the forms that will be the basis for the application's interface. Then objects that make up the interface are added to the forms. Essentially, the Integrated Development Environment (IDE) of Visual Basic is build around forms. Most programming feature of Visual Basics are built into forms and the appropriate code assigned to such controls.

2.6.2 Setting Properties

The next step is to set the properties for the object that is created. The properties Windows are used to do this in Visual Basic. The property of a form or controls determines the way the form , control or object appears, how it is displayed and the action it performs when a particular condition is satisfied.

2.6.3 Writing Code

The Code Editor window is where the Visual basic Codes are written for an application. Codes consist of language statements, constants and declarations. The code window can be used to view and edit any of the code in an application.

Codes in Visual basic is divided into smaller blocks called procedures. An event procedure such as those contains code that is executed when an event occurs (such as when a user clicks a button). An event procedure for a control combines the control's actual name (specified in the Name property), an underscore (_) and the event name. For example, if a programmer wants a command button name Command1 to invoke an event procedure when it is clicked, use the procedure Command1_Click.

```
    Private Sub enter_Click()
            Unload Form1
          Form2.Show
    End Sub
```

The event procedure is used to remove a form (Form1) from the screen and display another Form (Form2). The event procedure is invoked by clicking of enter command on a menu. The control name is *enter* while the event is a click event.

Chapter Three

Systems Analysis And Design

3.1 Description of Existing System.

The System that would be studied is the Business Management Procedure of Homewares Lightening System Limited; an establishment involved in the indoor and outdoor lightening, artistic painting and corporate gifts. Their major operations are;

- ❖ Stocking and Retailing of good to customers.
- ❖ Maintaining of a list of all customers.
- ❖ Transaction processing for all transactions carried out.

Maintaining manual logs, bin cards, inventory form, and invoices carries out these activities and the entire operations is coordinated by a production Supervisor.

The main aim of setting up the existing system is on commencement of the business operations of *Homewares Lightening Limited;* there arose the need for a record keeping method to be devised to take care of the business record keeping activities of the business. The existing system is strictly manual. That is all record keeping are done manual, transcribed to forms and other documents manually.

3.2. Problems of Existing System.

- ➢ No definite order and pattern for processing transaction records (information).
- ➢ Administrative and Record Keeping problems resulting from the absence of Data Processing Standards and Procedures.

- The absence of an existing MIS structure. Hence there is no laid out pattern for information sharing and communication among the different levels of managers.
- Records are difficult to trace, since the clerk has to flip through piles of cards and patter in sequential order to get to the record to be located.
- There is no logical links between the different types of records
- No effective method of protecting data integrity and protection from accidental loss, destruction or corruption.

The present problems occurred because of the following reasons;

- No existing information processing system and no set standard for data processing within the system. Therefore the Staff relies on arbitrary methods for handling their data.
- If there was an existing information processing system, information would be easily communicated to management. Hence the business management process would be a lot easier.
- The absence of trained and proficient staff that can handle and operate a Computer based information system.

3.3 Feasibility Study

Designing and implementing a new system Homewares Lightening Systems Limited is feasible. Each alternatives have been realistically viewed against the current market trends and existing situations. Though each alternative would help see to the solution of the problem at hand; they all have their peculiar merits. The available feasible alternatives are;

1. ***Office Automation Approach***: This involves the establishment of an Office System. An Office Systems is a set of equipment used to create, store, process, or communicate information in a business environment. This information can be manually, electrically, or electronically produced, duplicated, and transmitted. Most modern office equipment—including typewriters, dictation equipment, facsimile machines, photocopiers, calculators, word processors and telephone systems.

   - An office automation system though would be effective in providing a solution to the problems at hand it would serve for the short run only.
   - Using an office system the only means of sending printed reports and other recorded data to head office is by fax. This is more expensive and tedious as many copies of reports would be continuously faxed on a routine basis.
   - Using an Office System we would have no choice than sending reports and other data through mail or courier service. In the long run with envisaged expansion in business operations (growth), market trends and increasing competition, there would be the need for a more sophisticated information processing system; that would be flexible enough to cope with increasing business needs and the changing market environment. Therefore the Office System Approach would not offer a comprehensive solution to the problem at hand.

2. ***Information System Approach:*** An information processing system refers to the entire resources for handling the information requirement of an establishment. It incorporates both the human resources requirement, computers and automated devices involved. The major component of an information system is the data processing system within it.

The information processing approach involves the setting up information and data processing system. Data processing is the analysis and organization of data by the repeated use of one or more computer programs. Data processing is used extensively in business, engineering, and science and to an increasing extent in nearly all areas in which computers are used. Businesses use data processing for such tasks as payroll preparation, accounting, record keeping, inventory control, sales analysis, and the processing of bank and credit card/value card account statements.

The Data Processing System for the Homewares would be designed to operate as a complete integrated set of interrelated system. This approach would provide information that satisfies the following attributes, viz; *provide Information that is timely, accurate and relevant.*

The information system approach would also used a customized business management software to manage the entire transaction/information processing function of the organization.

The Information System Approach is the most feasible alternative to solving the problems of Homewares Lightening Systems.

The Information System Approach was then viewed from three perspectives;

- Technical Feasibility.
- Operational Feasibility.
- Economic Feasibility.

Technical feasibility is the extent to which it is possible for Computer Systems to be applied as a replacement for business activities that were hitherto done manually. The was found to be quite feasible. Since business management software would be designed, it would serve as the new means of data entry and storage hence eliminating the need for data to be recorded on paper. The PC (microcomputer) is a versatile tool whose application cuts across diverse fields,.

Operational feasibility: the new system would be feasible operationally since new system would be designed to simulate the manual system and also eliminate the shortcomings of the manual system. Personnel would be trained on the inner workings and operations of the system and adequate documentation provided. Hence it possible for the system to be operated successfully.

Economic Feasibility: This is the cost consideration involved in implementing anew system. In recent times, there has been a considerable decrease in the cost of Microcomputer Systems. Hence Information processing system that are dependent on one or two PC are too cost intensive to implement. The cost implications of developing a new system are made up for by the benefits obtainable from the new system.

3.4 Strengths of New System

The alternative method would offer the following benefits;

i. Ensure effective processing of transaction, administration and operational data.

ii. Proper record keeping and effective information documentation both in form of printed reports and as digital files on secondary storage devices.

iii. The new system would provide a management view of the entire organization, since all business information can be accessed from a single location.

iv. Information can be easily accessed and retrieved when need.

v. Multiple reports can be created for different transaction records.

3.1 Design Strategies For Implementing a Business Management System

- Identification of current system requirements
- Selection and Organization of the data that needs to be stored
- Planning for system Efficiency and Reliability
- Economic Cost comparison: Determination of the cost of implementing the proposed system

3.5.1 Planning System Requirement

- Determination of the information or data to be keep track off.
- Organization strategy of the required information
- Determination of the user of the proposed system and the capability of such user.
- Projection of future database needs.

3.5.2 Potential Hurdles

- Apathy of users to computerized systems, leading to the exhibition of system under-utilization.
- Over dependency leading to the assumptions that the computerized system is fail-prove.
- Unauthorized access to stored data and information.
- Data lost due to user lack of maintenance culture
- System malfunctioning.

- User adaptation to old system.

3.5.3 Solving Potential Problems

- Adequate training of operators, users and all personnel concerned with the use of the new system.
- Periodic/Routine backup of stored information.
- Access control and user verification methods should be built into the software.
- Management Control: Management should always summarize information, observe trends and performance of variances.

3.6 Features Of Used Programming Language (Visual Basic 6.0)

The proposed system would be developed with modern Object Oriented Programming Tools (Visual Basic) and would very much help in enhancing any standard operations.
The Visual Basic has a number of features, which in turn are incorporated into the proposed system, this feature includes:

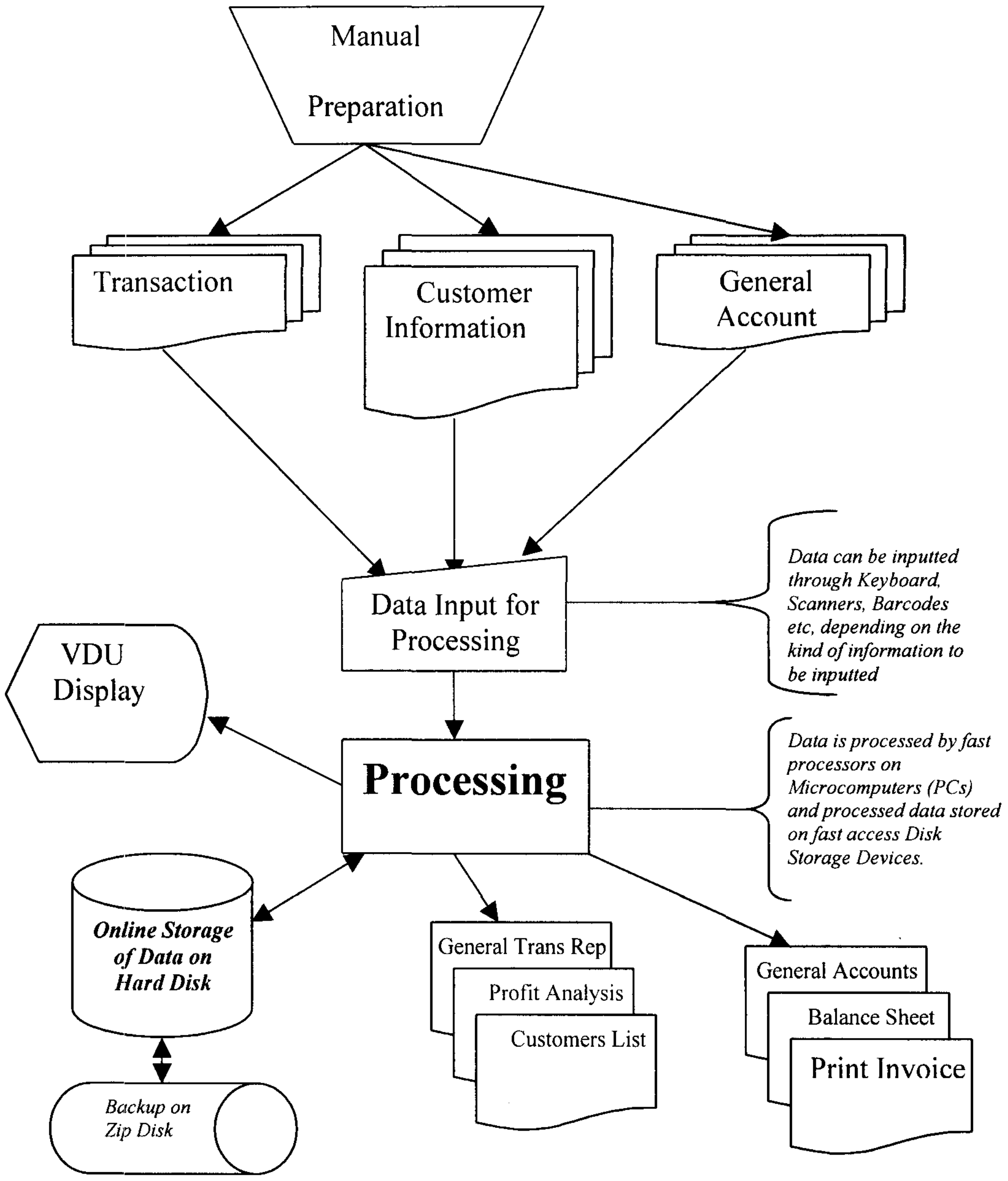
FEATURES

- ***Full Windows Compatibility:*** the software is designed to run in the Ms Windows Operating and any compatible environment, which is the standard for any modern software.
- ***Mouse Support :*** apart from keyboard invocations of commands, the software we develop supports the use of the mouse, hence, it is easier to navigate through the software environment and tasks can be executed at the click of a button.
- ***Menu and Event Driven:*** the software incorporate Pull Down menus, windows compatible dialogue boxes, prompts and command buttons. This has the added advantage of making the software more user-friendly and interactive.
- ***Connectivity with other Windows based application:*** the software has the ability for Dynamic Data Exchange (DDE) and Dynamic Link Library (DLL) for Shared files, with other Ms Windows based software. As such data can be exported or imported to

and from it (i.e. communication with other window based software is totally effective).

- ***Improved Multiple Database Structure***: the database structure of any new software developed by us uses the Microsoft Open Database Connectivity (ODBC) style that supports the creation of multiple tables in a single database. This allows for better handling of large volume of related data items while still compact.
- ***Security:*** the software is designed such that only valid and authorized users can only gain access; thereby protecting the integrity of your data and the source code of your program being tampered with, which is one of the problems you faced with uncompileable program. Also the databases in the software could encrypted such that they can only be manipulated by the software alone.
- ***Output Control:*** the software controls the output to Screen, Printer or directly to a file in the format of any Ms Windows base standard software. (E.g. Access, Excel, Ms Word)
- ***Practical WYSIWYG (What-You-See-Is-What-You-Get) Report Display:*** the display screen shows you exactly what your final report will look like when your print
- ***Multiple Size Page Views:*** View as many report pages at Actual, Double or Half size.
- ***Pictorial Data Representation:*** data charting in our software gives view enhancement.
- ***Multi-Media Effects:*** Visual and Audio multi-media are available optionally, also they use all your current Operating System Settings like Sound, Screen Color, Date & Time e.t.c.
- ***Utility Features:*** pop-up on screen Calculator & Calendar.

3.7 System Flowchart


Manual Preparation
Transaction
Customer Information
General Account
Data Input for Processing
Data can be inputted through Keyboard, Scanners, Barcodes etc, depending on the kind of information to be inputted
VDU Display
Processing
Data is processed by fast processors on Microcomputers (PCs) and processed data stored on fast access Disk Storage Devices.
Online Storage of Data on Hard Disk
Backup on Zip Disk
General Trans Rep
Profit Analysis
Customers List
General Accounts
Balance Sheet
Print Invoice

3.8 Input Specification

The database used supports Open Database Connectivity (ODBC) techniques that allows the creation of multiple tables in a single database. Hence the database is assigned an extension

*.mdb (*multiple database). The tables contained in the Homewares.mdb database and their structure are;

- **Customer :** for storing customers details

Fields Name	DataType	Field Size	Field Meaning
Name	Text	50	Customer Name
Cuscode	Text	12	Customer Code
OfficeTel	Text	15	Office Telephone
HomeTel	Text	15	Home Telephone
Address	Text	150	Customer Address
MobileTel	Text	15	Mobile Phone

This table is indexed on the customer code field (cuscode).

- **Category:** for storing assigned categories

Fields Name	DataType	Field Size	Field Meaning
Name	Text	50	Item Category Name
Code	Text	50	Category Code

- **Balsheet:** this table generates a balance sheet from the available transaction.

Fields Name	DataType	Field Size	Field Meaning
Desc	Text	200	Description
CRAmt	Currency	8	Credit Amount
DRAmt	Currency	8	Debit Amount

- **Pass :** storing registered password and user access levels.

Fields Name	DataType	Field Size	Field Meaning
Username	Text	60	Username
Password	Text	30	Pass Word
AccessLevel	Text	30	Access Level for user

➢ **PayDetails :** Recording payment and invoicing details.

Fields Name	DataType	Field Size	Field Meaning
Cuscode	Text	18	Customer Code
OrderNo	Text	15	Order Number
VAT	Currency	8	VAT Charge
Paid	Currency	8	Amount Paid
PayType	Text	10	Payment Type (Cash or Cheque)
Bal	Currency	8	Outstanding balance
BankName	Text	50	Name of Bank
ChequeNo	Text	20	Cheque Number
AccountNo	Text	20	Account Number
Date	Date/Time	8	Date of Transaction

This table is indexed on the CusCode and OrderNo

➢ **Settings :** This table is used to assigned settings for the Company that would be making use of the software.

Fields Name	DataType	Field Size	Field Meaning
CoyName	Text	50	Company Name
Admin	Text	50	Administrator's Name
VAT	Double	8	% charged on VAT
CoyAdd	Memo		Company Address

➢ **Transaction** : for registering new order and generating invoice.

Fields Name	DataType	Field Size	Field Meaning
Icode	Text	50	Item Code
Cashier	Text	50	Cashier Name
OrderNo	Double	15	Order Number
Quantity	Text	8	Quantity of Item
CusCode	Text	15	Customer Code
TranDate	Date/Time	8	Date of Trans
TotalAmount	Currency	8	Total Amount

This table is indexed on the CusCode, Icode OrderNo and TransDate.

➢ **Expenditure** : stores data for expenditure account

Fields Name	DataType	Field Size	Field Meaning
TransType	Text	50	Transaction Type
TransAmount	Currency	8	Transaction Amount
TransDate	Date/Time	8	Transaction Date
TransBy	Text	50	Staff Authorizing Transaction
TransTo	Text	50	Customer to which Transaction is made
Receipt_no	Text	20	Receipt Number
TransUnit	Double	8	No of Items Transacted

➢ **Income** : stores data for income account

Fields Name	DataType	Field Size	Field Meaning
TransType	Text	50	Field Meaning
TransAmount	Currency	8	Transaction Amount
TransDate	Date/Time	8	Transaction Date
TransBy	Text	50	Staff carrying out trans
TransTo	Text	50	Organization involved in transaction
Receipt_no	Text	20	Receipt Number
TransUnit	Double	8	No of items transacted.

➢ **ItemInvent** : is used to update the available inventory after a transaction has been processes

Fields Name	DataType	Field Size	Field Meaning
Desc	Text	50	Description
UnitInStock	Double	8	Unit in Stock
Category	Text	50	Item Category
ReOrder	Double	8	Reorder Level
UnitSellPrice	Currency	8	Unit Selling Price
UnitCostPrice	Currency	8	Unit Cost Price

This table is indexed on the Desc, Icode

➢ **HomeInvent** : is used to add a new item to available inventory at hand.

Fields Name	DataType	Field Size	Field Meaning
Desc	Text	50	Description
UnitInStock	Double	8	Unit in Stock
ReOrder	Double	8	Reorder Level

The tables illustrated above are the component units of the *Homwares.mdb* database. The database structured in the Microsoft Access Database format.

3.9 OUTPUT SPECIFICATION

For the output of processing to be view (on Paper or Screen) they have to be first written to a file (Database Table) before they are sent to any output medium (Printer- for Hardcopy or Screen – for Softcopy). It should be noted that most (in fact all) of this Table are generic, with the sole aim of 'dumping' Report data/information and are deleted afterwards. It is also note-worthy that a generated table could be used for/by many Reports.

- **TrashGAcct**

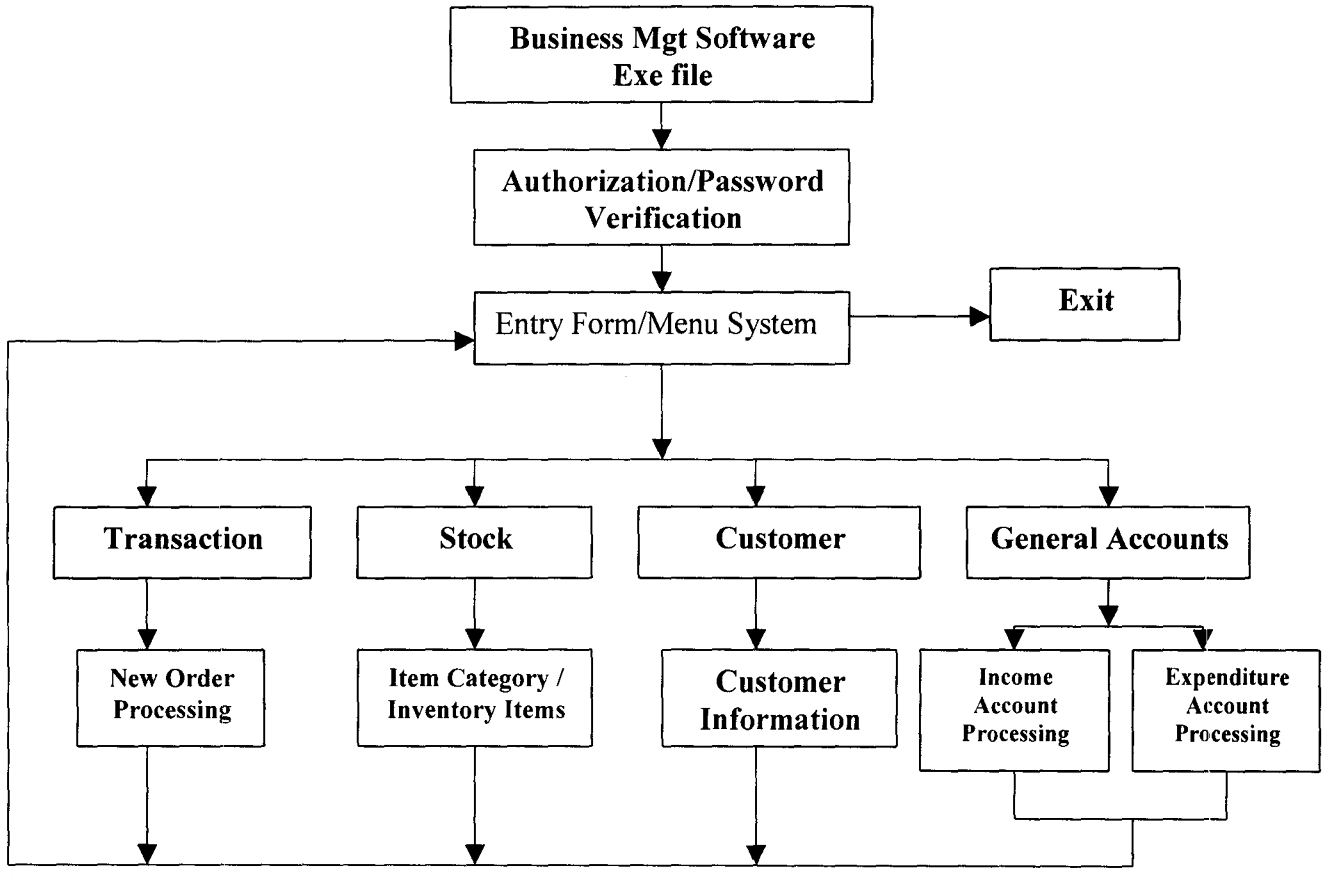
Fields Name	DataType	Field Size	Field Meaning
TransType	Text	50	Transaction Type
TransAmount	Currency	8	Amount
TransDate	Date/Time	8	Transaction date
TransTo	Text	50	Transaction total
Receipt-No	Text	20	Receipt No. Issued
TransUnit	Double	8	

- **PayDTrash**

Fields Name	Data Type	Field Size	Field Meaning
Cuscode	Text	15	Customer Code
OrderNo	Text	15	Order Number
VAT	Currency	8	Charge for VAT
Paid	Currency	8	Amount paid
PayType	Text	10	Payment Type (Cash or Cheque)
Bal	Currency	8	Balance
BankName	Text	60	Bank Name
ChequeNo	Text	20	Cheque Number
AccountNo	Text	20	Account Number
Date	Date/Time	8	Date of Payment

This table is indexed on the Cuscode and OrderNo fields.

3.10 PROCEDURE CHART

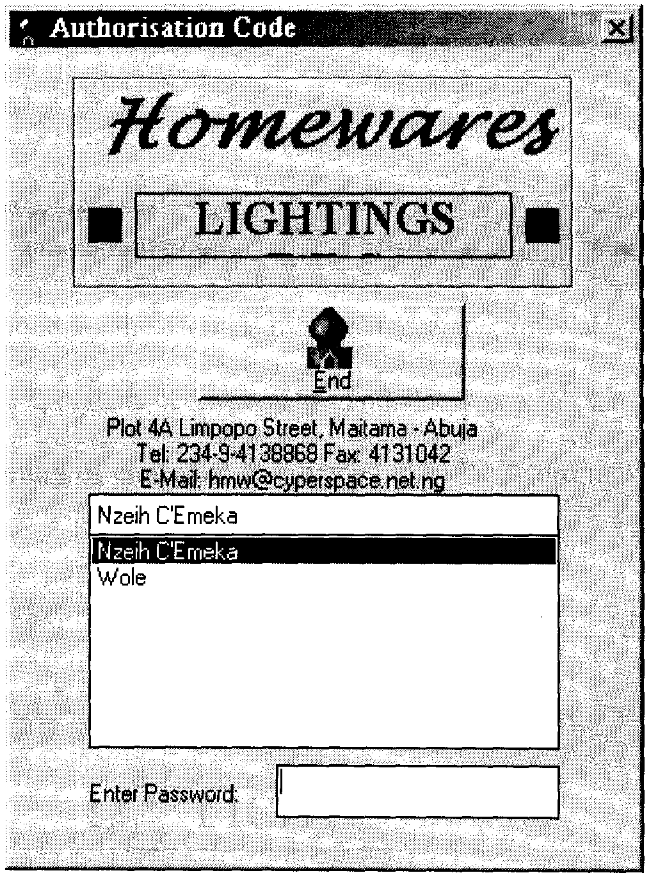

Business Mgt Software Exe file
Authorization/Password Verification
Entry Form/Menu System
Exit
Transaction
Stock
Customer
General Accounts
New Order Processing
Item Category / Inventory Items
Customer Information
Income Account Processing
Expenditure Account Processing

3.11 INPUT DESIGN

The Design of Screens (FORMS as called in Visual Basic) is important in any system development process, because it is through this Forms (interface) that the User actually communicates with the program, thus, the efficiency or robustness of a program is firstly determined by the User interface (Forms). It is the user friendliness of the Forms in a program that determines whether is Software is good or not. With this taken into consideration, the following input designs are used.

FormAccess (Access.frm)

Screen Purpose: This Form doubly serves as the Welcoming Screen as well as the authorization check-point of the Homewares Management Softwares. Here it is expected that the user should supply his/her Username and Password to gain access to the system proper. An incorrect entry of either the Username or the Password makes the user an invalid user.



Plot 4A Limpopo Street, Maitama - Abuja
Tel: 234-9-4138868 Fax: 4131042
E-Mail: hmw@cyperspace.net.ng

Nzeih C'Emeka

Nzeih C'Emeka
Wole

Enter Password:

FrmMainMenu.Frm

Screen Purpose: This is the Main menu Form that contains the different activities the system can do. Different operational options are available on the main menu form for users to choose. The main menu form displays two types of menu items, the horizontal pull down menu and the vertical pop-up menus;

The functions performed by the horizontal menu items are;

REPORTS

- *General Transaction Reports:* This option displays the general transaction reports on a monthly, daily, weekly, quarterly or yearly basis based on the option selected. This option also displays the expected profit Analysis, Credit List, Debtor List and can be used to Print Invoices.
- *General Income:* This is used to invoke the report that displays the income breakdown from all preceeding transactions.
- *General Expenditure:* This is used to invoke the report that displays all expenditure (expense) carried out the organization.
- *Balance Sheet:* This option is used to prepare a trial balance from all transactions. It used a time frame to query the tables containing transaction information.
- *Inventory Analysis:* This option is used to display a listing of, List of Stock Items, Item Category Listing and Customer List.

TOOLS

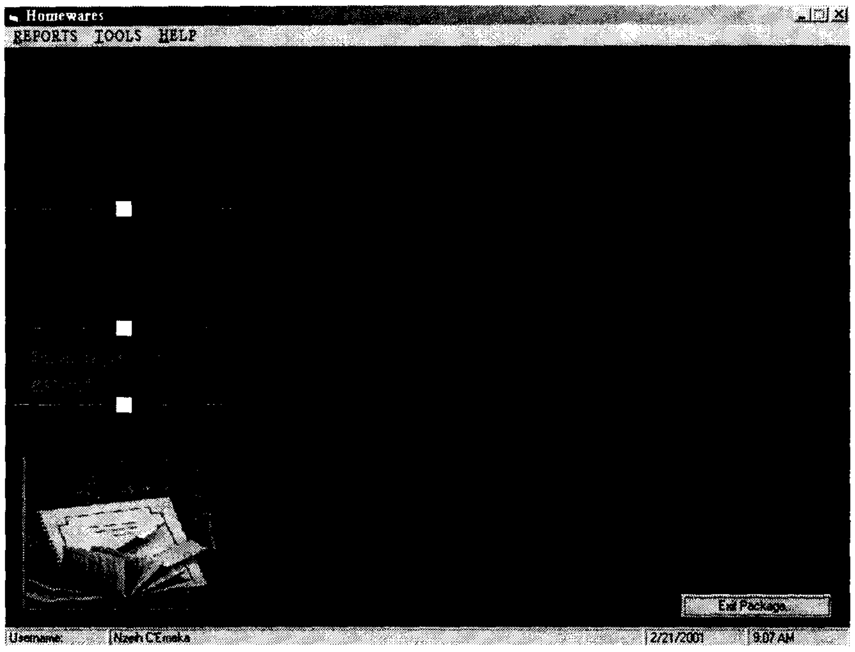
- *Password Administration*: This is used to set the list of Authorized User Name and their password. The name and password are input when Frmaccess (Access.Frm) is display when the software starts.
- *Software Settings:* This is used to set company particulars (Name and Address) of the organization that would be making used of the software.

The Vertical Pop-Up menu displays a set of command buttons on the right hand side of the screen when they receive focus (when the mouse pointer rests on any of the menu item). The vertical menu items are used mainly to enter data into an underlying table or process a transaction. When any of the vertical menu item receives focus a set of command buttons appears on the right and the user can select by clicking any of the options that corresponds with the intended operation.

TRANSACTION

- *Transaction:* This displays the main transaction entry form, general invoice, find and updates and edits an existing order and performs general transaction process.
- *Stock Details:* This option is used to add or view Item category.
- *Customer Details:* This is displays the form that is used to send customer information to the Customer table.
- *House Ware Inventory:* is used to add and view item inventory and make inventory requisition.
- *House General Accounts:* This is used to view and update income and expenditure account information.
- *End:* This option is used to exit the application. Wne it is clicked it displays a message box that prompt the user to continue or exit the application environment.

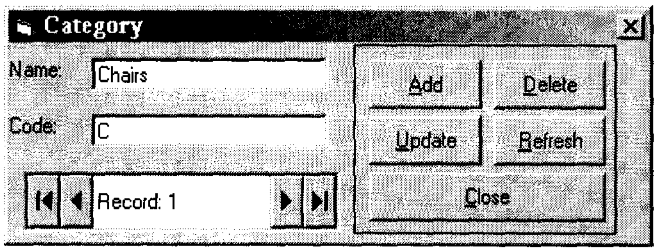
The FrmMainMenu also display a status bar at the bottom of the screen that displays the name of the current user, time and time.



Username: Nzeh C Emeka 2/21/2001 9:07 AM

frmCategory.frm

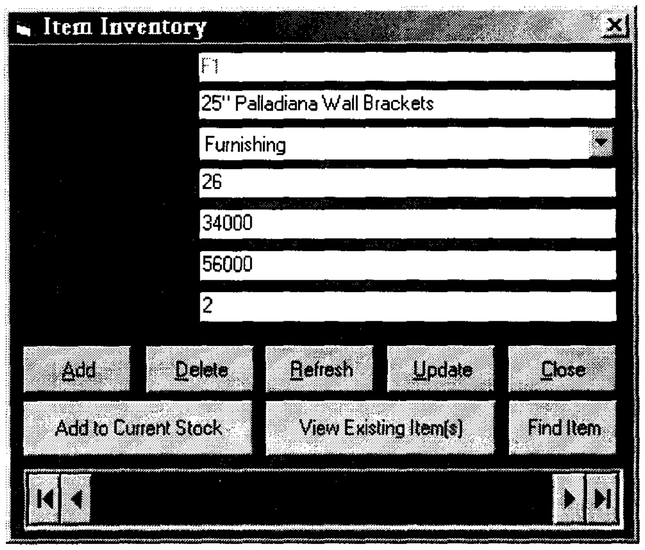
Screen Purpose: This is the input design that is used to add or delete item categories.



Name: Chairs
Code: C
Record: 1

frmItemInvent.frm

Screen Purpose: This is used to input or edit data for item inventory.


Item Inventory

F1

25" Palladiana Wall Brackets

Furnishing

26

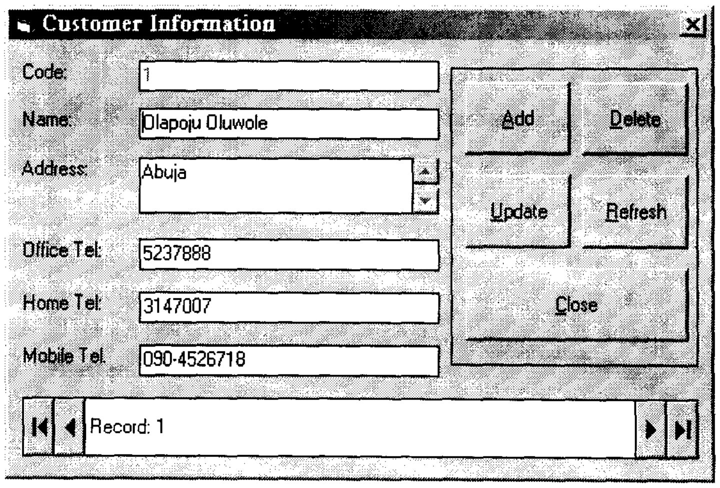
34000

56000

2

frmCustDetails.frm

Screen Purpose: this from allows for inputting, editing and deleting of custonmers information.


Customer Information

Code: 1

Name: Olapoju Oluwole

Address: Abuja

Office Tel: 5237888

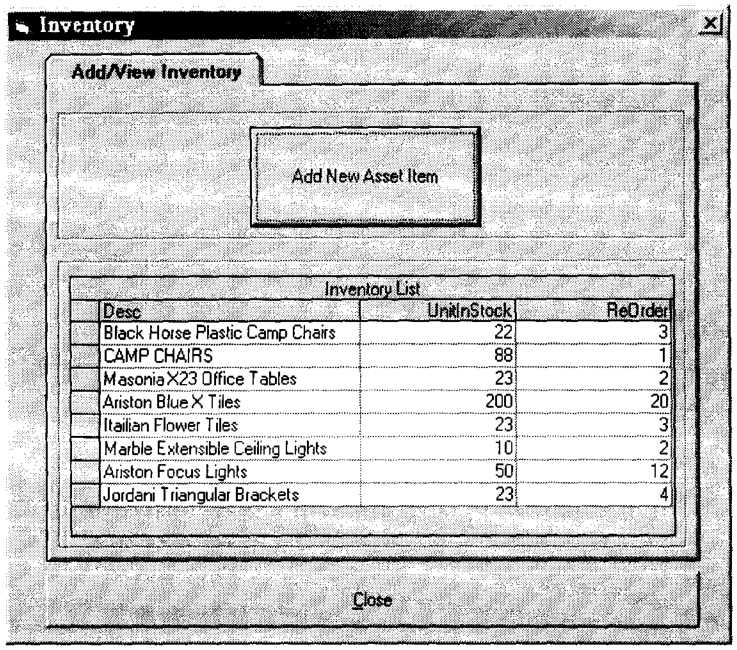
Home Tel: 3147007

Mobile Tel: 090-4526718

Record: 1

frmInvent.frm

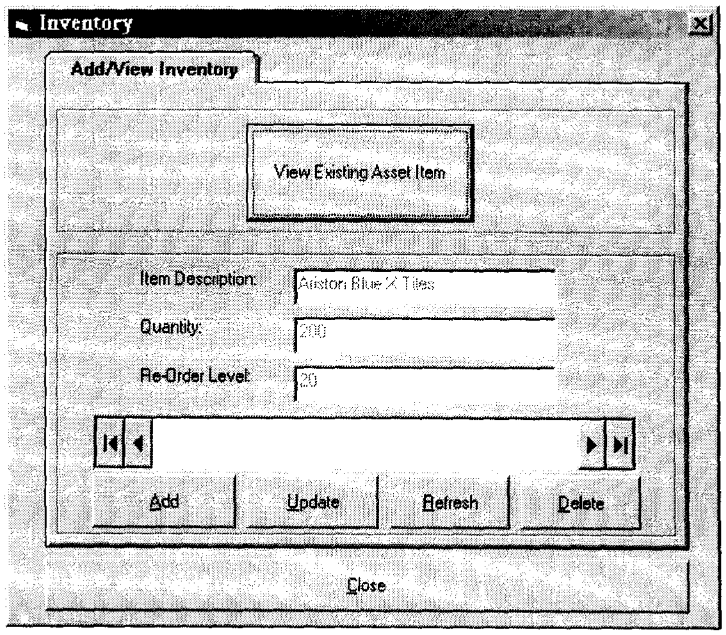
Screen Purpose: This form displays the Inventory list of available items in stock and number of such asset in stock and its reorder level..



Inventory List

Desc	UnitInStock	ReOrder
Black Horse Plastic Camp Chairs	22	3
CAMP CHAIRS	88	1
MasoniaX23 Office Tables	23	2
Ariston Blue X Tiles	200	20
Itailian Flower Tiles	23	3
Marble Extensible Ceiling Lights	10	2
Ariston Focus Lights	50	12
Jordani Triangular Brackets	23	4

When the Add New Asset button is clicked, the form is modified to allow for new assets to be registered.

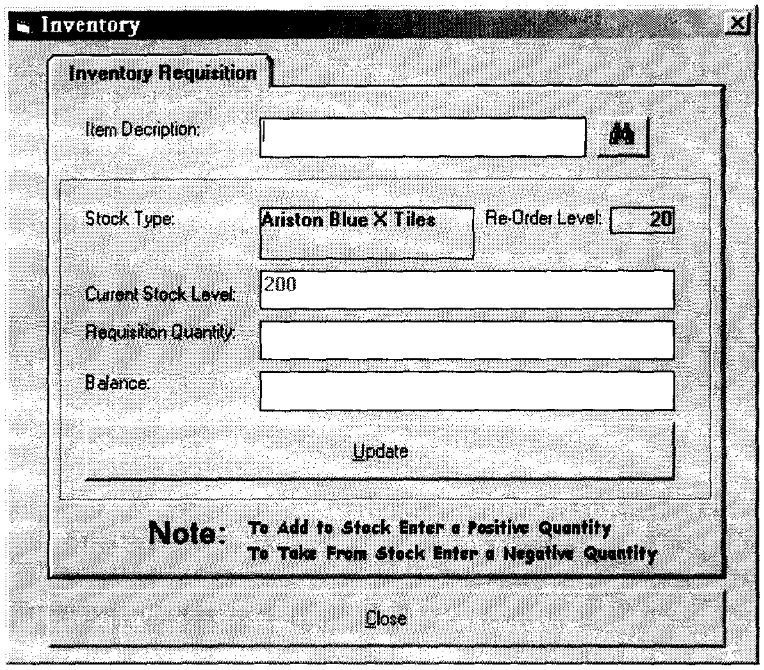


Item Description: Ariston Blue X Tiles

Quantity: 200

Re-Order Level: 20

The **frmInvent.frm** can also be modified to display requisition when an inventory requisition is to be made.



Item Decription:

Stock Type: Ariston Blue X Tiles

Re-Order Level: 20

Current Stock Level: 200

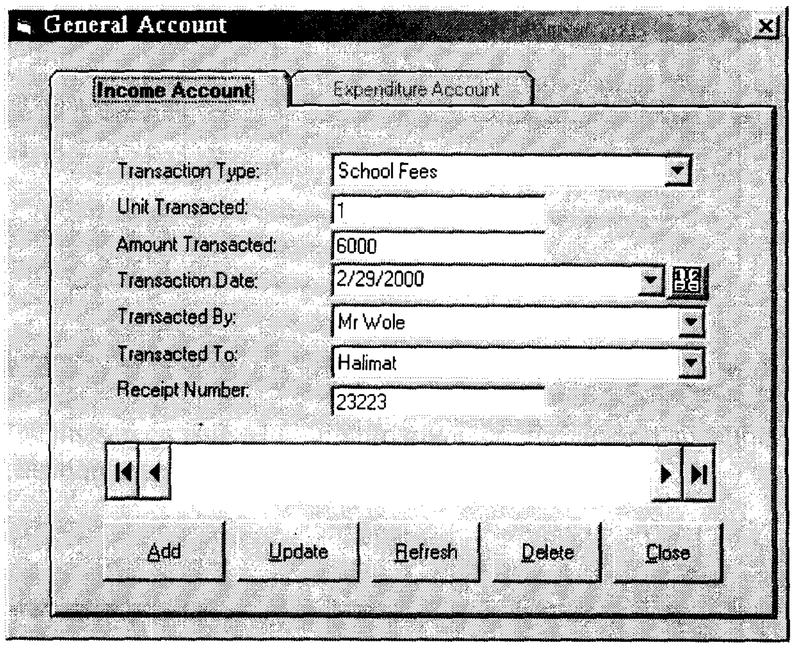
Requisition Quantity:

Balance:

Note: To Add to Stock Enter a Positive Quantity
To Take From Stock Enter a Negative Quantity

frmGAccount.Frm

Screen Purpose: This screen is used to input parameters for general accounts. That is income and expenditure accounts.



Transaction Type: School Fees

Unit Transacted: 1

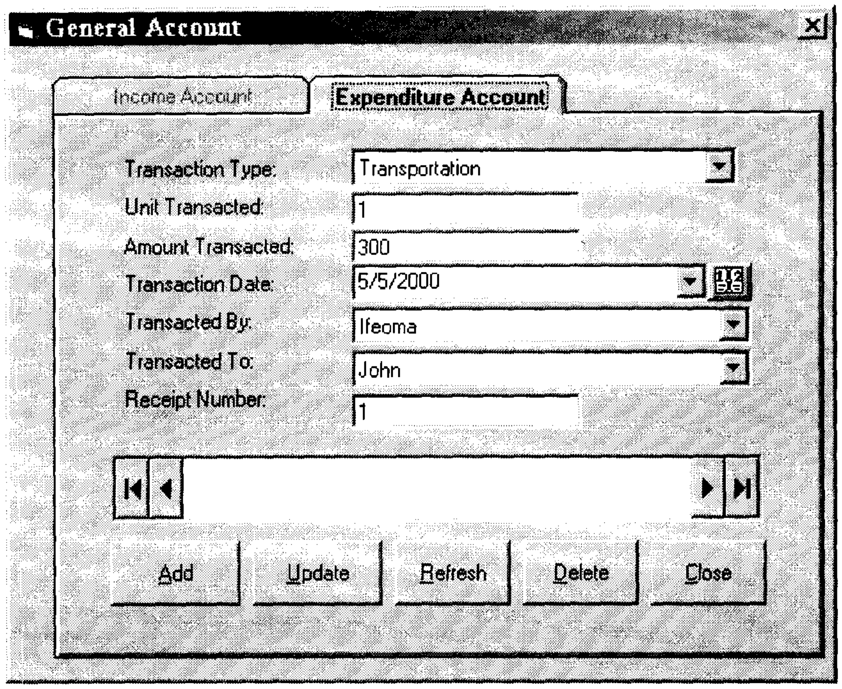
Amount Transacted: 6000

Transaction Date: 2/29/2000

Transacted By: Mr Wole

Transacted To: Halimat

Receipt Number: 23223

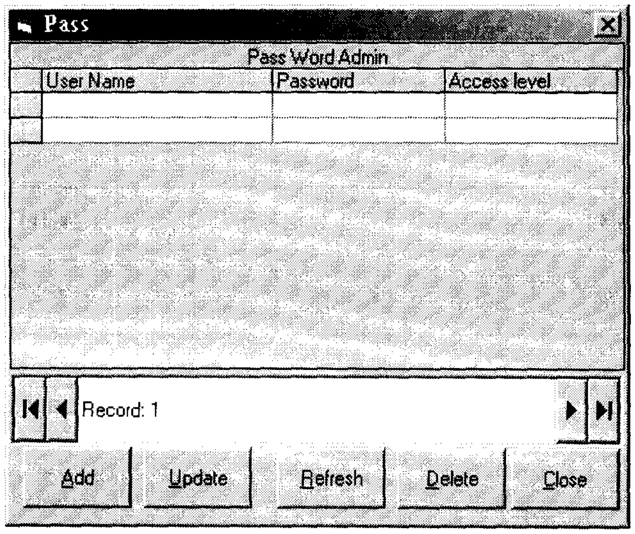

General Account

Income Account | Expenditure Account

Transaction Type:	Transportation
Unit Transacted:	1
Amount Transacted:	300
Transaction Date:	5/5/2000
Transacted By:	Ifeoma
Transacted To:	John
Receipt Number:	1

<u>frmPass.frm</u>

Screen Purpose: for entering new and editing existing user password and access levels.


Pass

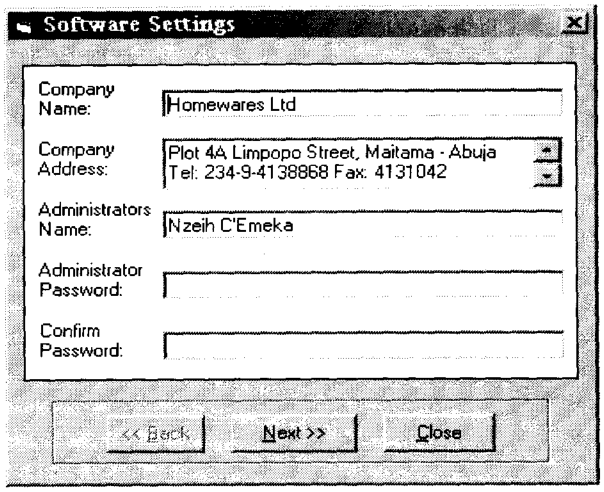
Pass Word Admin

User Name	Password	Access level

Record: 1

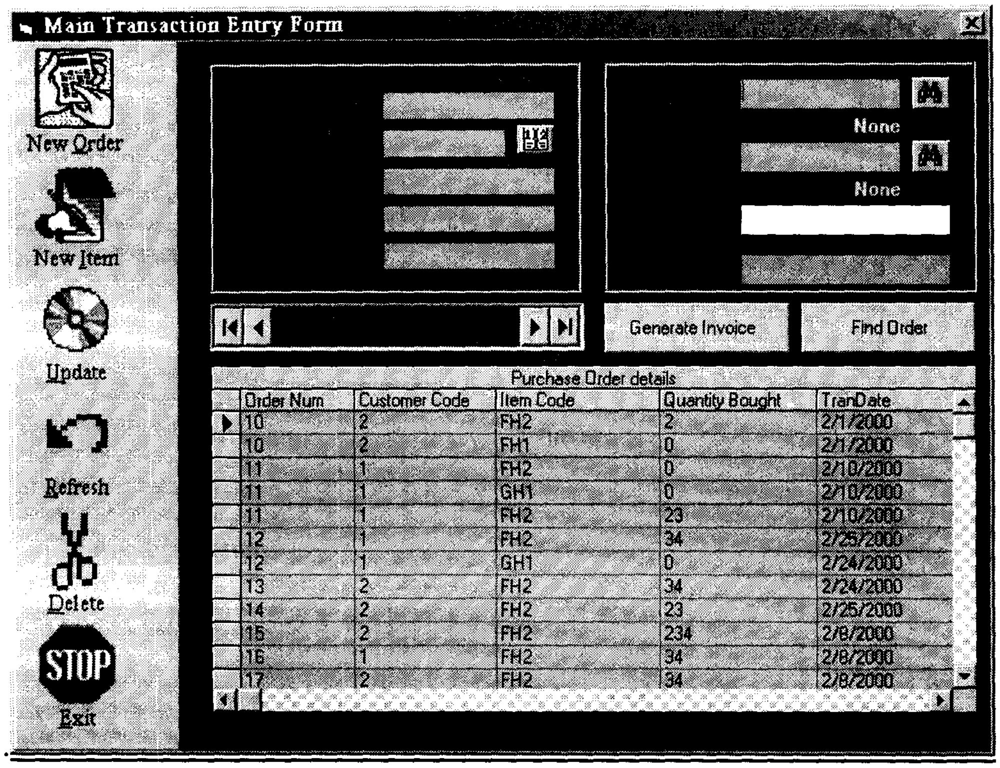
frmSetting.frm

Screen Purpose: this form is used to enter company particulars and name of administrators password.



Company Name:	Homewares Ltd
Company Address:	Plot 4A Limpopo Street, Maitama - Abuja Tel: 234-9-4138868 Fax: 4131042
Administrators Name:	Nzeih C'Emeka
Administrator Password:	
Confirm Password:	

frmTransaction.frm



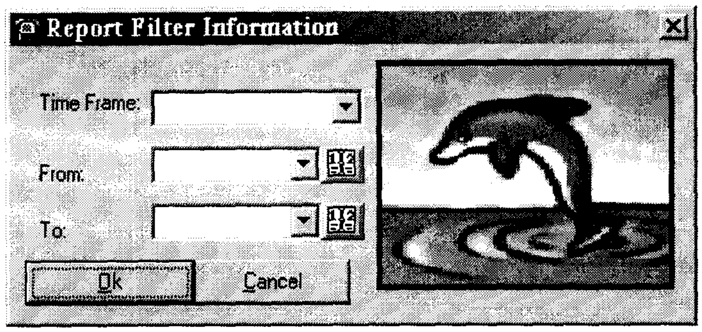
Purchase Order details

	Order Num	Customer Code	Item Code	Quantity Bought	TranDate
▶	10	2	FH2	2	2/1/2000
	10	2	FH1	0	2/1/2000
	11	1	FH2	0	2/10/2000
	11	1	GH1	0	2/10/2000
	11	1	FH2	23	2/10/2000
	12	1	FH2	34	2/25/2000
	12	1	GH1	0	2/24/2000
	13	2	FH2	34	2/24/2000
	14	2	FH2	23	2/25/2000
	15	2	FH2	234	2/8/2000
	16	1	FH2	34	2/8/2000
	17	2	FH2	34	2/8/2000

Screen Purpose: This form is use to place new orders, editing or delete and existing order, generate invoices and for general transaction processing.

FrmGetFilter.frm

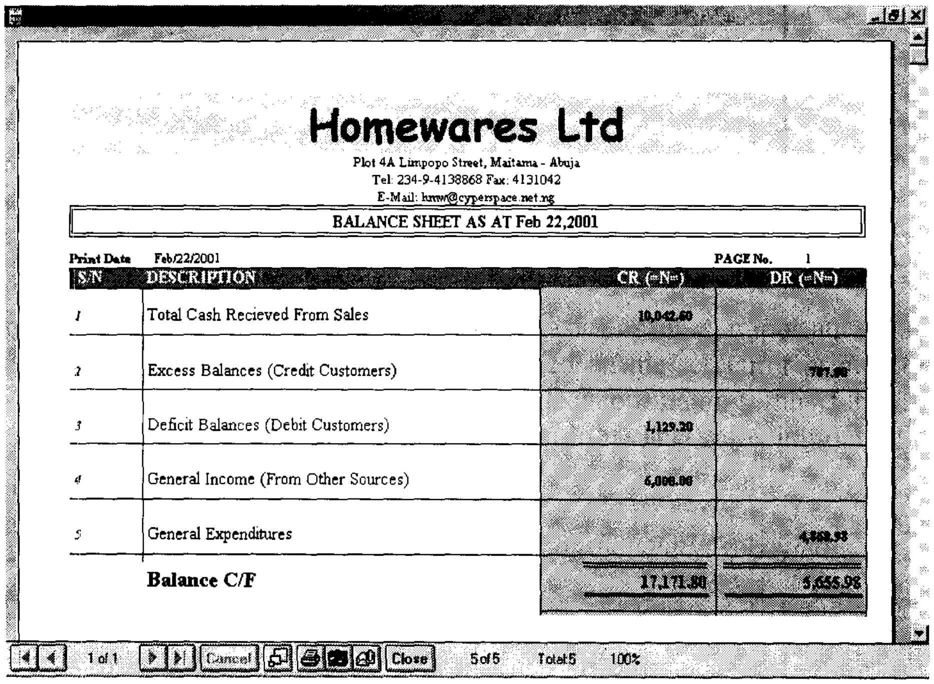
Screen Purpose: This is used to specify the period or time frame or a financial report would be generated.



3.12 OUTPUT DESIGN

Output Design (REPORTS as called in Visual Basic) is also important in any system development process. Reports can either be displayed on the screen or printed on paper. A good report is a basic instrument for management decision making, thus for a report to be alright it has to contain all bit of information required in it. The following output designs are used.

BalSheet.rpt: *displays or prints the balance sheet as at the month specified.*



Homewares Ltd

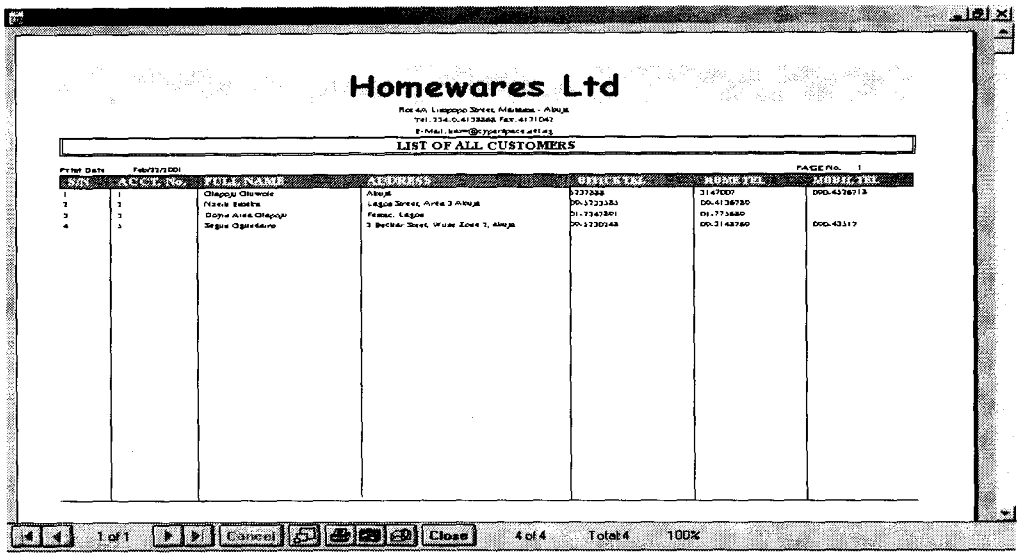
Plot 4A Limpopo Street, Maitama - Abuja
Tel: 234-9-4138868 Fax: 4131042
E-Mail: hmw@cyperspace.net.ng

BALANCE SHEET AS AT Feb 22,2001

Print Date Feb/22/2001 PAGE No. 1

S/N	DESCRIPTION	CR (=N=)	DR (=N=)
1	Total Cash Recieved From Sales	10,042.60	
2	Excess Balances (Credit Customers)		787.00
3	Deficit Balances (Debit Customers)	1,129.20	
4	General Income (From Other Sources)	6,000.00	
5	General Expenditures		4,868.98
	Balance C/F	17,171.80	5,655.98

Customer.rpt: *diplays the list of all registered customers and their other particulars.*



Homewares Ltd

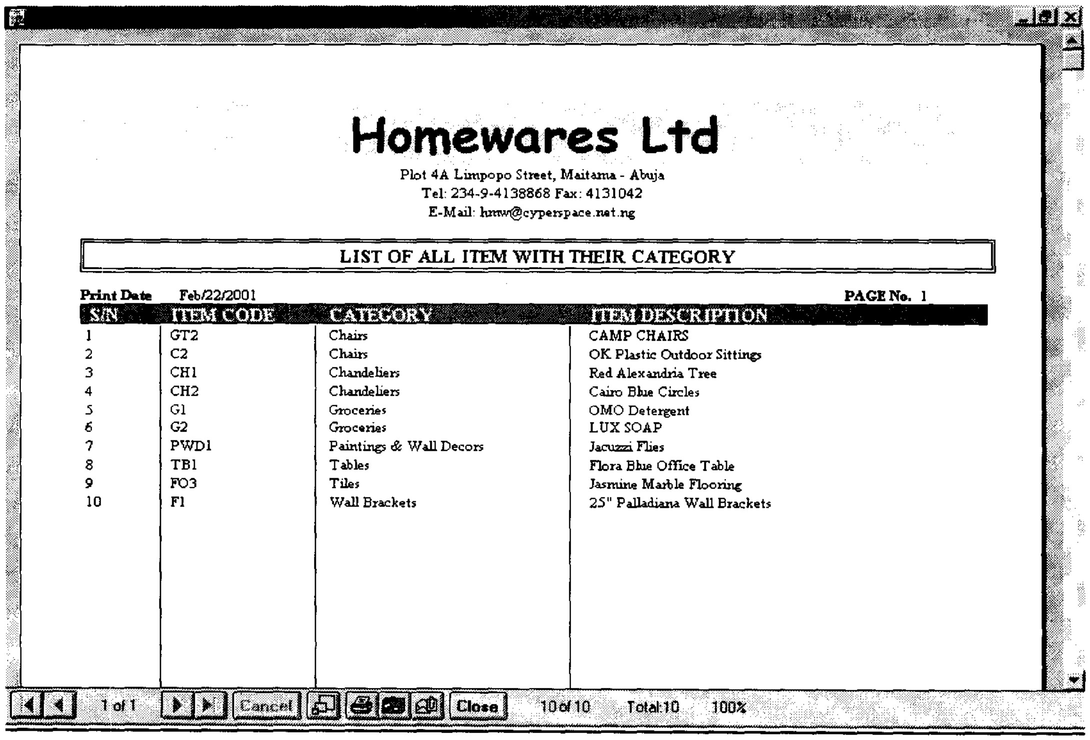
Plot 4A Limpopo Street, Maitama - Abuja
Tel: 234-9-4138868 Fax: 4131042
E-Mail: hmw@cyperspace.net.ng

LIST OF ALL CUSTOMERS

Print Date Feb/22/2001 PAGE No. 1

S/N	ACCT. No.	FULL NAME	ADDRESS	OFFICE TEL	HOME TEL	MOBIL TEL
1	1	Olapoju Oluwole	Abuja	5237338	3147007	090-4376713
2	2	Nzeik Ebeire	Lagos Street, Area 3 Abuja	09-5233383	09-4136720	
3	3	Doyin Aina Olapoju	Femac, Lagos	01-7347801	01-773680	
4	5	Segun Ogundairo	3 Bechar Street, Wuse Zone 7, Abuja	09-5230248	09-3143760	090-43317

Itemlist.rpt: ***displays the list of all item in stcok with their category.***



Homewares Ltd

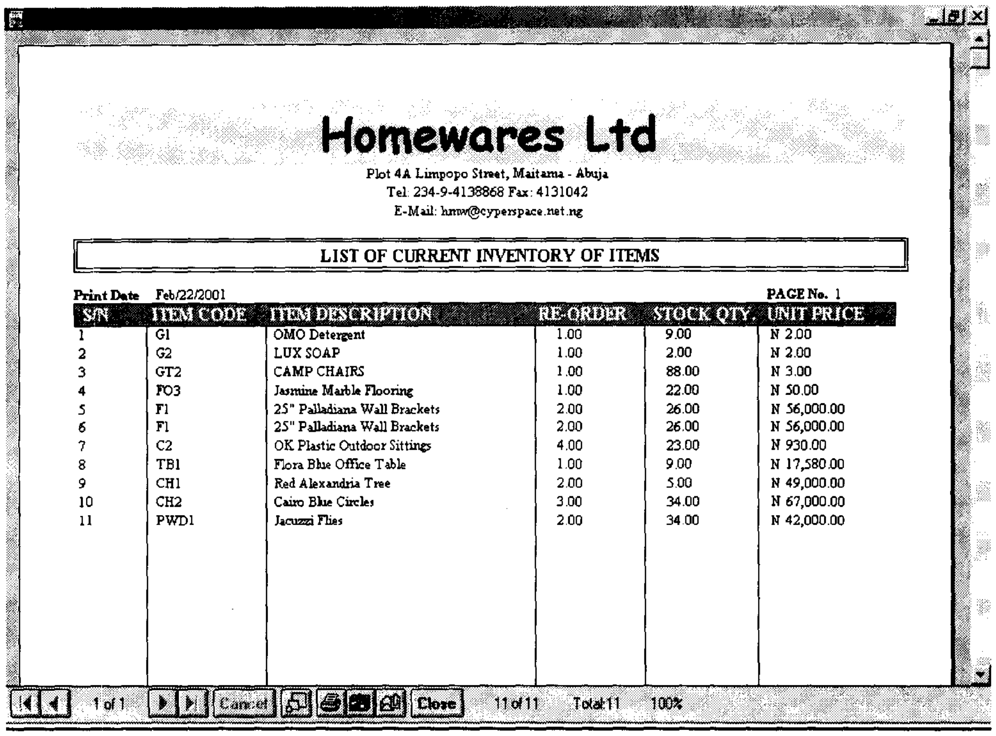
Plot 4A Limpopo Street, Maitama - Abuja
Tel: 234-9-4138868 Fax: 4131042
E-Mail: hmw@cyperspace.net.ng

LIST OF ALL ITEM WITH THEIR CATEGORY

Print Date Feb/22/2001 **PAGE No.** 1

S/N	ITEM CODE	CATEGORY	ITEM DESCRIPTION
1	GT2	Chairs	CAMP CHAIRS
2	C2	Chairs	OK Plastic Outdoor Sittings
3	CH1	Chandeliers	Red Alexandria Tree
4	CH2	Chandeliers	Cairo Blue Circles
5	G1	Groceries	OMO Detergent
6	G2	Groceries	LUX SOAP
7	PWD1	Paintings & Wall Decors	Jacuzzi Flies
8	TB1	Tables	Flora Blue Office Table
9	FO3	Tiles	Jasmine Marble Flooring
10	F1	Wall Brackets	25" Palladiana Wall Brackets

ItemInvent.rpt: *displays Items in Inventory, the quantity in stock and the unit selling price of each item.*



Homewares Ltd

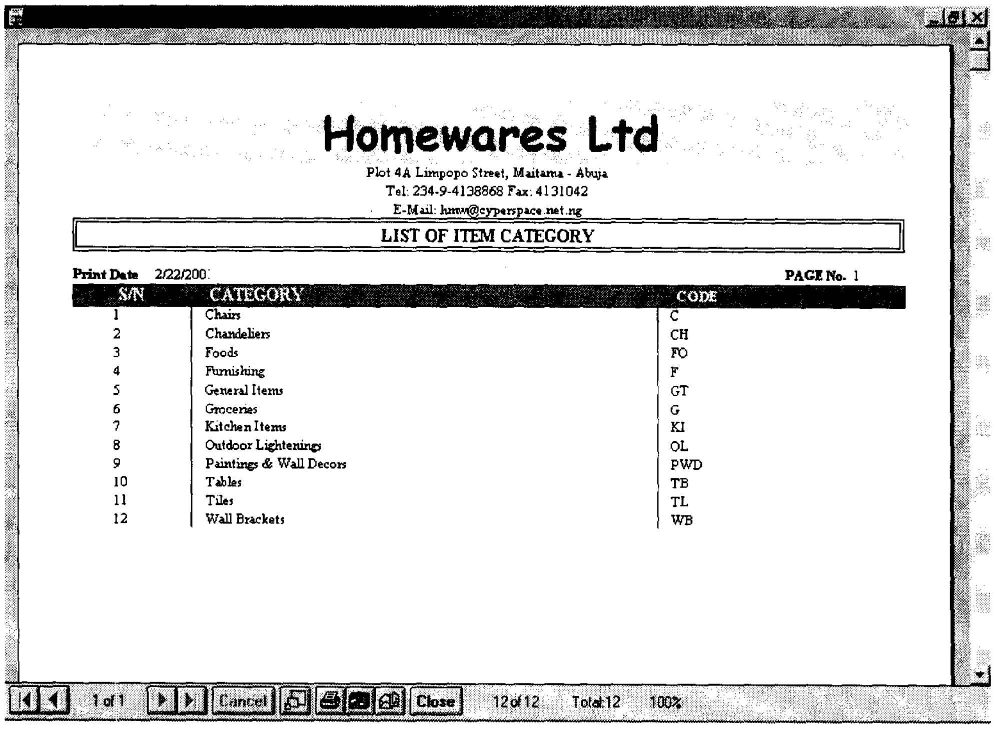
Plot 4A Limpopo Street, Maitama - Abuja
Tel: 234-9-4138868 Fax: 4131042
E-Mail: hmw@cyperspace.net.ng

LIST OF CURRENT INVENTORY OF ITEMS

Print Date Feb/22/2001 PAGE No. 1

S/N	ITEM CODE	ITEM DESCRIPTION	RE-ORDER	STOCK QTY.	UNIT PRICE
1	G1	OMO Detergent	1.00	9.00	N 2.00
2	G2	LUX SOAP	1.00	2.00	N 2.00
3	GT2	CAMP CHAIRS	1.00	88.00	N 3.00
4	FO3	Jasmine Marble Flooring	1.00	22.00	N 50.00
5	F1	25" Palladiana Wall Brackets	2.00	26.00	N 56,000.00
6	F1	25" Palladiana Wall Brackets	2.00	26.00	N 56,000.00
7	C2	OK Plastic Outdoor Sittings	4.00	23.00	N 930.00
8	TB1	Flora Blue Office Table	1.00	9.00	N 17,580.00
9	CH1	Red Alexandria Tree	2.00	5.00	N 49,000.00
10	CH2	Cairo Blue Circles	3.00	34.00	N 67,000.00
11	PWD1	Jacuzzi Flies	2.00	34.00	N 42,000.00

Category.rpt: *displays item categories and the appropriate codes assigned to them.*



Homewares Ltd

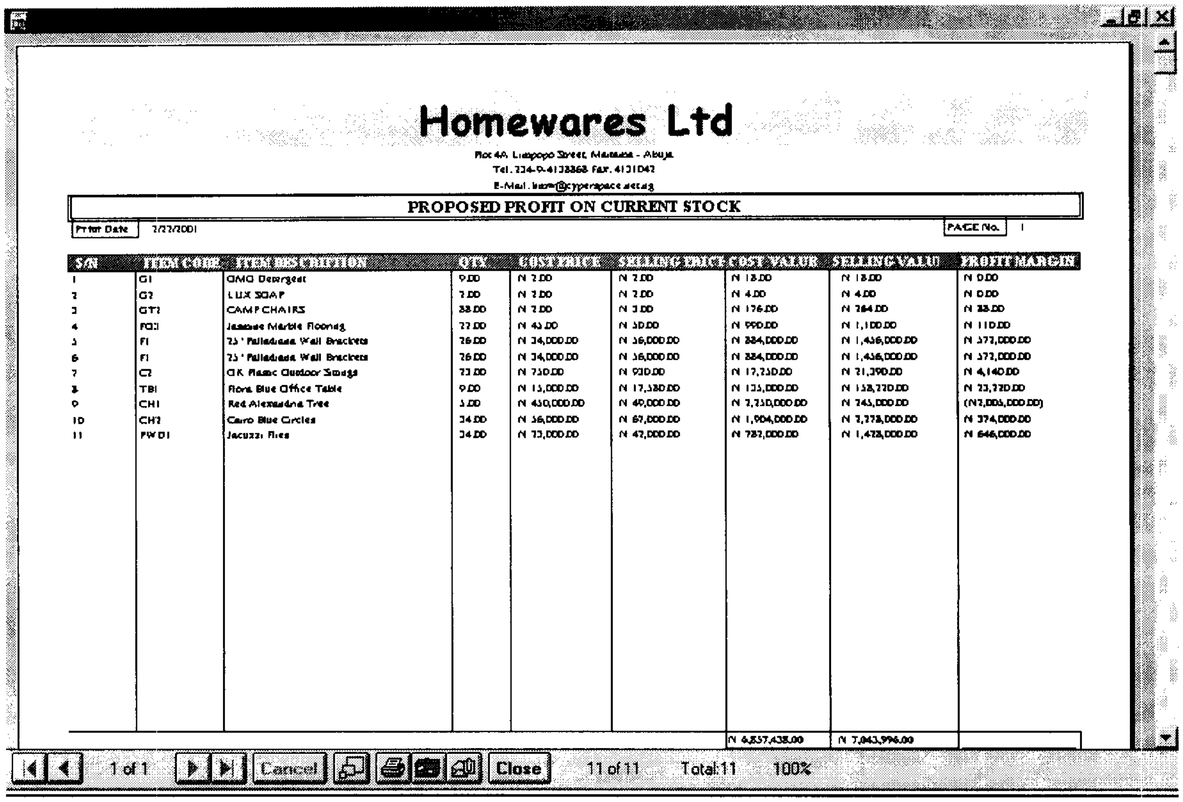
Plot 4A Limpopo Street, Maitama - Abuja
Tel: 234-9-4138868 Fax: 4131042
E-Mail: hmw@cyperspace.net.ng

LIST OF ITEM CATEGORY

Print Date 2/22/2001 PAGE No. 1

S/N	CATEGORY	CODE
1	Chairs	C
2	Chandeliers	CH
3	Foods	FO
4	Furnishing	F
5	General Items	GT
6	Groceries	G
7	Kitchen Items	KI
8	Outdoor Lightenings	OL
9	Paintings & Wall Decors	PWD
10	Tables	TB
11	Tiles	TL
12	Wall Brackets	WB

Profit1.rpt: *shows the profit the organization stands to make from selling outstanding items in stock.*



Homewares Ltd

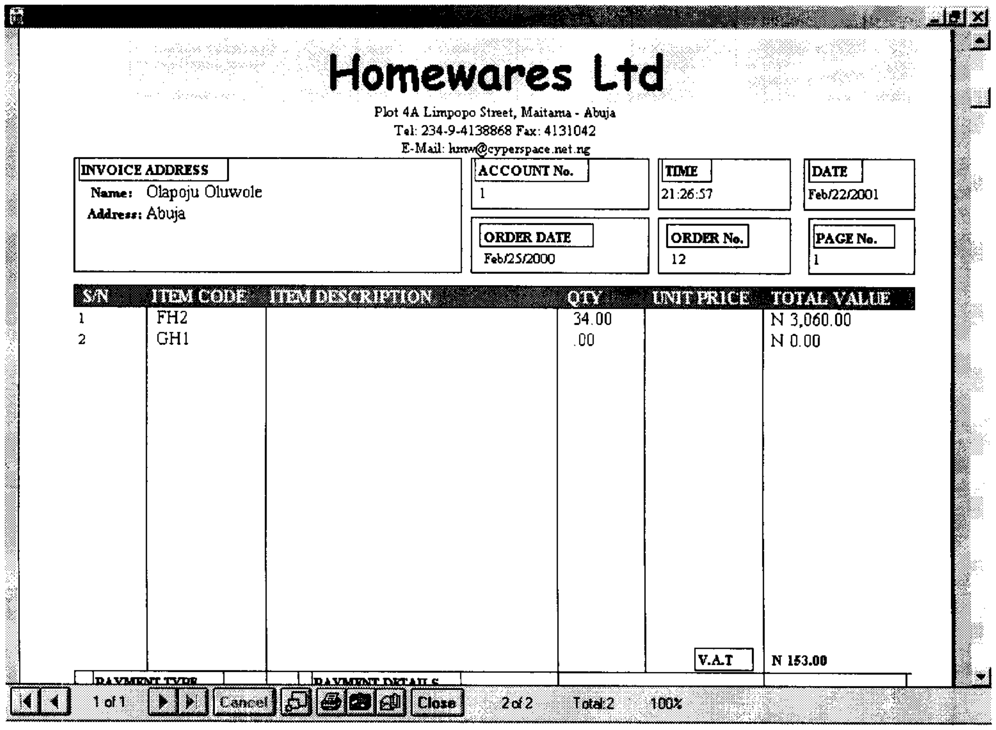
Plot 4A Limpopo Street, Maitama - Abuja
Tel. 234-9-4138868 Fax. 4131042
E-Mail. hmw@cyperspace.net.ng

PROPOSED PROFIT ON CURRENT STOCK

Print Date 2/22/2001 PAGE No. 1

S/N	ITEM CODE	ITEM DESCRIPTION	QTY	COST PRICE	SELLING PRICE	COST VALUE	SELLING VALUE	PROFIT MARGIN
1	G1	OMO Detergent	9.00	N 2.00	N 2.00	N 18.00	N 18.00	N 0.00
2	G2	LUX SOAP	2.00	N 2.00	N 2.00	N 4.00	N 4.00	N 0.00
3	GT2	CAMP CHAIRS	88.00	N 2.00	N 3.00	N 176.00	N 264.00	N 88.00
4	FO2	Jasmine Marble Flooring	22.00	N 45.00	N 50.00	N 990.00	N 1,100.00	N 110.00
5	F1	25 ' Palladiana Wall Brackets	26.00	N 34,000.00	N 56,000.00	N 884,000.00	N 1,456,000.00	N 572,000.00
6	F1	25 ' Palladiana Wall Brackets	26.00	N 34,000.00	N 56,000.00	N 884,000.00	N 1,456,000.00	N 572,000.00
7	C2	GK Plastic Outdoor Swings	23.00	N 750.00	N 930.00	N 17,250.00	N 21,390.00	N 4,140.00
8	TB1	Flora Blue Office Table	9.00	N 15,000.00	N 17,580.00	N 135,000.00	N 158,220.00	N 23,220.00
9	CH1	Red Alexandria Tree	5.00	N 450,000.00	N 49,000.00	N 2,250,000.00	N 245,000.00	(N2,005,000.00)
10	CH2	Cairo Blue Circles	34.00	N 56,000.00	N 67,000.00	N 1,904,000.00	N 2,278,000.00	N 374,000.00
11	FWD1	Jacuzzi Flies	34.00	N 23,000.00	N 42,000.00	N 782,000.00	N 1,428,000.00	N 646,000.00
						N 6,557,438.00	N 7,043,996.00	

Invoice.rpt: ***generates an invoice for a processed transaction/order.***



Homewares Ltd

Plot 4A Limpopo Street, Maitama - Abuja
Tel: 234-9-4138868 Fax: 4131042
E-Mail: hmw@cyperspace.net.ng

INVOICE ADDRESS **Name:** Olapoju Oluwole **Address:** Abuja	**ACCOUNT No.** 1	**TIME** 21:26:57	**DATE** Feb/22/2001
	ORDER DATE Feb/25/2000	**ORDER No.** 12	**PAGE No.** 1

S/N	ITEM CODE	ITEM DESCRIPTION	QTY	UNIT PRICE	TOTAL VALUE
1	FH2		34.00		N 3,060.00
2	GH1		.00		N 0.00
				V.A.T	N 153.00

PAYMENT TYPE PAYMENT DETAILS

CHAPTER FOUR

System Implementation

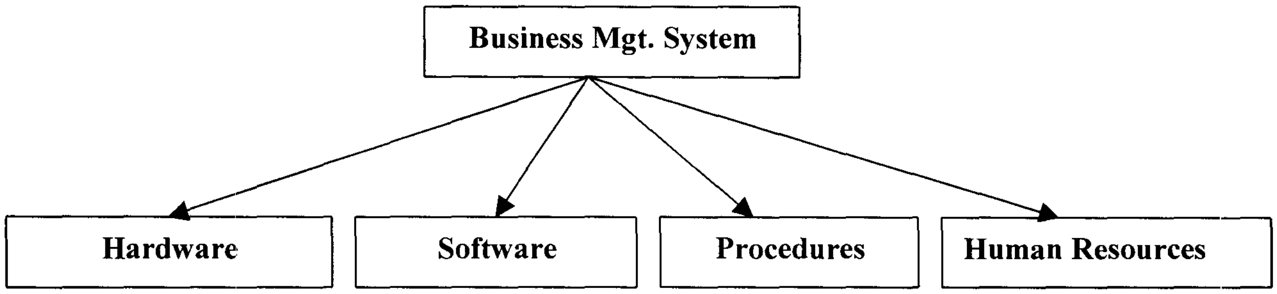
4.1 INTRODUCTION

Implementation is the process of applying the developed system for the purpose it is meant for. System implementation involves the development of quality assurance procedures, including data security, back-up, recovery and system control system implementation objective is to complete the orderly and unobtrusive installation of the new system. During the system implementation, the new system is installed and users have the opportunity to operate the new system in "parallel" with the existing system.

The system implementation comprises the following task:

- Application system installation
- Documentation to provide user manuals
- Users Training on the new system
- Parallel system testing
- Data conversion/migration
- Acceptance of Testing
- System setup

4.2 REQUIREMENTS FOR IMPLEMENTING A BUSINESS MANAGEMENT SOFTWARE.



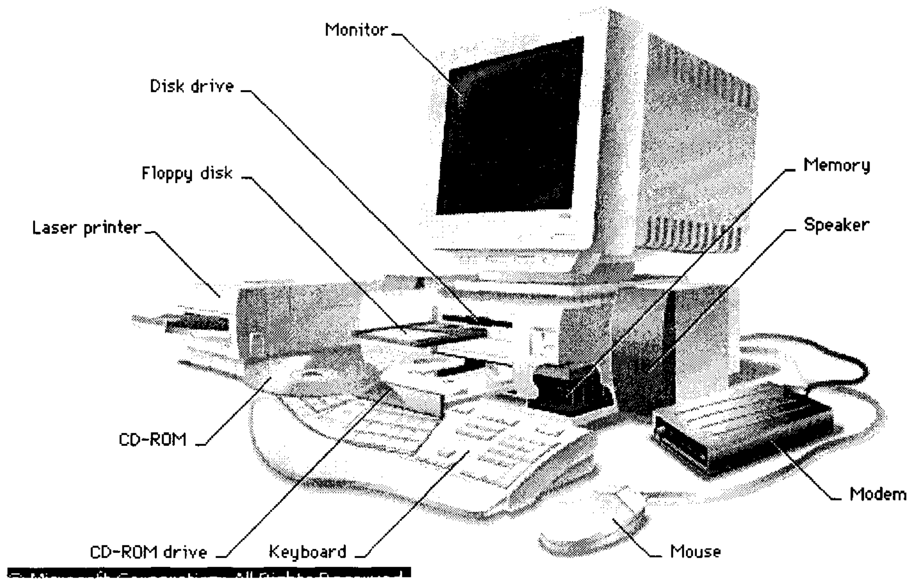
4.2.1 Hardware Requirement

The Hardware requirement for implementing the Computer Based Business Management System is Complete PC (Personal Computer) and its associated accessories.

Personal Computer (PC) are machine capable of repetitively and quickly performing calculations and instructions. Designed to be used by a single person, a PC is smaller, less expensive, and easier to use than other classes of computers, such as supercomputers, mainframe computers, and workstations.

PCs have revolutionized entertainment, science, the media, art, medicine, education, and business because they provide computational abilities at a low cost to people with no extensive programming experience. PCs enable artists to envision and manipulate images. Musicians use them for learning, creating, and recording music. Businesses track finances and forecast company performance using PCs. Foreign correspondents can compose news stories on portable PCs, called laptops, and electronically submit these stories from remote locations. Many people work at home and communicate with fellow workers via their PCs in a practice known as telecommuting. PCs are also able to interface with worldwide communication networks, such as the Internet, and the graphics-based information database known as the World Wide Web to find information on any subject.

PCs consist of electronic circuitry called a microprocessor, such as the central processing unit (CPU), that directs logical and arithmetical functions and runs computer programs. A PC also has electronic memory to temporarily store programs and data and mass storage devices—such as hard, floppy, and compact disc (CD-ROM) drives—to permanently store programs and data. Information and commands are entered by the user via a keyboard or a pointing device called a mouse. Information from the PC is displayed on a video monitor or on a liquid crystal display (LCD) video screen, or it can be printed on laser, dot-matrix, or inkjet printers



With the continuous manufacturing of Microprocessors of higher computing strength by major processor manufacturers such as Intel Inc AMD (advance Micro Devices) and Cyrix Technologies Corp., the power of PCs have grown sporadically in recent times. PCs have gradually grown from stand alone and single user computers to system that can support a network for multi user access. PC has gradually encroached into areas that hitherto used to be the exclusive domain of Minis and mainframe computers. With the entry of newer processor models like the Intel Pentium III series, AMK K62-3D and others the market the computing strength of PCs would continue to be on the increase.

Hardware Specification

- Intel Pentium III 700
- 64 MB Synchronous DRAM.
- 8 MB AGP Set.
- 10.2 Gb Hard Disk Drive.
- 52x CD-ROM.
- 15" Super VGA.
- Minitower ATX Casing.
- Microsoft PS/2 Mouse
- 1.4MB 3.5" FDD.
- Windows PS/2 Keyboard.
- IOMEGA 250MB Zip Disk.
- Full Multimedia.

Other Accessories.

- HP Deskjet 1120 Printer.
- APC 650 Smart UPS.

4.2.2 Software Requirement

The software required by the system is a combination of both required operating (system software) and the application software is a business management software.

Operating System is the basic software that controls a computer. The operating system has three major functions: It coordinates and manipulates computer hardware, such as computer memory, printers, disks, keyboard, mouse, and monitor; it organizes files on a variety of storage media, such as floppy disk, hard drive, compact disc, and tape; and it manages hardware errors and the loss of data.

Operating systems control different computer processes, such as running a spreadsheet program or accessing information from the computer's memory. One important process is the interpretation of commands that allow the user to communicate with the computer. Some command interpreters are text oriented, requiring commands to be typed in. Other command interpreters are graphically oriented and let the user communicate by pointing and clicking on an *icon,* an on-screen picture that represents a specific command. Beginners generally find graphically oriented interpreters easier to use, but many experienced computer users prefer text-oriented command interpreters because they are more powerful.

Operating systems are either single-tasking or multitasking. The more primitive single-tasking operating systems can run only one process at a time. For instance, when the computer is printing a document, it cannot start another process or respond to new commands until the printing is completed.

The operating that would be used for the system would be the Microsoft Windows Operating System; any of the following versions of Windows would suffice; Microsoft Windows 98, Windows Me (Millennium Edition) and Microsoft Windows 2000 Professional.

4.2.3 **Human Resources Requirement**

The existing staff of the company would be trained on the mode of operation of the system. This is necessary because it would be easier for the personnel that are already conversant with the operations of the manual system to understand the new system. Therefore it is not necessary for new persons to be recruited.

4.2.4 **PROCEDURES**

Procedures are step-by-step method(s) of using a system to be able to achieve result. A procedure in the context of this project is a physical component because they are provided in a physical form such as manual and instruction booklets. The major types of procedures that are required are:

- User instructions
- Instruction for preparation of input
- Operating instructions for the computer center personnel.

4.3 **SYSTEM TESTING**

After the installation of the new system, the system must undergo a test, once all the programs have been written and the training of the personnel to use the system is completed. The system testing is to ensure that all the sub-programs have been efficiently and correctly written. The system testing entails the execution of the program with test data so as to enable the system developer and the management to know the operational efficiency of the system.

The system testing will also enable the designer to correct errors and delete programs or modules that are not efficient or relevant by a process called debugging, using test data input into the programs so as to produce the desired output reports. Test data of all possible type/kind are used in other that all likely behaviors of the system to the input is ascertained before actual system implementation.

During this task, the Programmers or the System designer(s) assists the project staff in conducting the testing of the developed system so as to ensure that the system meets all the users needs and requirements. System testing entails the testing and certification of the system developed. This phase ensures that all required features, functions and capabilities are present in the system developed, and that all other requirements are met. Any necessary revisions are made during the system testing.

It is note-worthy that test data should be of 'real-live' nature.

4.4 SYSTEM SET-UP

After the successful System Testing, and the system output or requirements are mutually accepted by the Users of the system, the System Analyst and System Programmer(s), the next thing in the system implementation line is the setting up of the installed system, that is, putting in place or entering the basic information necessary for the system smooth take off.

Setup information in this new system include:

- **Authorization Setup**: Here it is required to identify the users of the new system and Password assigned to them. It is also note-worthy that not every user can have unlimited access to the entire system, thus, access levels would also be determined for each user of the system. Authorization need not be done every time the system is ran, but only when new users are to be given access to the system or if modification is necessary – changing password or access level. The assignment of access to users can only be done be the System Administrator.

- **User Information:** The new system is an 'open-system', and can be used by any other organization or company that uses or want to use the same method in this system for its pension administration. Thus it is pertinent that the user information is supplied once, upon the first running of the system.

4.5 CHANGE-OVER & DATA MIGRATION

The change over from old to new system may take place when the system has been proved to the satisfaction of the new System Analyst and the other implementation activities have been completed.

The method and approach used for the change over is the parallel running system. The parallel system testing means processing current data by both the old and new system concurrently, to cross check the result and compares them. The main advantage is that the old system is kept alive and operational until the new system has been proved for at least one system circle. Using 'real-live' data in the real operational environment of the equipments, people and data, the results of the new system will be compared with old system to ensure the efficiency, capability and durability before acceptance by the user.

The change over task is designed to ensure that the software developed replicate the functionality of the system to be replaced.

Once the change over ends, the user staff complete their training and the parallel system testing are successful, the conversion of records of the old system to the file format of the new system, which involve data entry/capture of several forms of data using the software (*Data Migration*), is necessary.

Chapter Five

5.1 Conclusion

For Computers to be effectively put to use; there must be software that would be used to complement functions of the hardware. The Business Management Software have been designed with the principles of effective Software development in mind. This is aimed at making the Software user friendly, hence easy to use. Full Windows feels and look features and controls have been properly used. Hence the software can be used with minimal supervision.

5.2 Recommendation

This software is recommended for use by any organization involved in the nature of business as specified by that carried out by Homeware Lightening Systems Ltd. The Software can be adapted to suit the business by changing the Company Name, Address and Administrators particulars. Also the source code for the software can be modified if need be to incorporate other business function to suit the need of any organization.

References

Andersen R.G., *Data Processing and Management Information Systems: Vol I & II* M & E Handbooks, London 1998.

Amudsen Mike. *Teach Yourself Visual Basic 5,* 2nd Ed. Sams Publising, Inc, USA.

Loren D. Edahl *Platinum Edition, Using Visual Basic 5.0;* Loren D. Eidahl; Macmillan Computer Publishing, 1997.

Mcmanus P. Jeffrey *How to Program in Visual Basic 5,* Macmillian Publishing, CA, USA, 1997.

Peter Norton and William Stanek *Peter Norton's Guide to Java Programming;* Sams Publishing, IN, USA, 1996

Microsoft Developers Network 6.0 (MSDN Library Visual Studio 6.0), Microsoft Inc, Redmond WA, USA

Microsoft Encarta Encyclopaedia, Micosoft Inc., Redmond WA, USA 1999.

Code for Module 1 (Hwares.bas)

```
Public dbs As Database
Public TheOrderNo As String
Public nItem As ListItem
Public rstTemp As Recordset
Public State, UserName, RepTitle As String
Public Time, CoyName, CoyAdd As String
Public VAT, TransSum, NewTransSum As Double
Public TheCurrentStock, SellingPrice As Double
Public mode, PrintMode, AccessLevel, TheFinancialFlag As
Integer
```

Codes for FrmAccess.frm

```
Public mode As Integer

Private Sub Command1_Click()
If Command1.Caption = "&Start" Then
FrmAccess.Height = 6540
Command1.Caption = "&End"
 Exit Sub
End If
If Command1.Caption = "&End" Then End
End Sub


Private Sub DBCombo1_Click(Area As Integer)
Text1.SetFocus
End Sub

Private Sub DBCombo1_KeyPress(KeyAscii As Integer)
Call Text1_KeyPress(13)
End Sub

Private Sub Form_Click()
RESP = MsgBox("Do you want to exit y/n?", vbYesNo +
vbCritical)
If RESP = vbYes Then End
End Sub

Private Sub Form_Load()
On Error GoTo handler
mode = 1
FrmSettings.Data1.Refresh
If FrmSettings.Data1.Recordset.EOF = False Then
FrmSettings.Data1.Recordset.MoveFirst
CoyName = FrmSettings.Data1.Recordset![CoyName]
CoyAdd = FrmSettings.Data1.Recordset![CoyAdd]
VAT = FrmSettings.Data1.Recordset![VAT]
Else
FrmSettings.Show 1
FrmSettings.Data1.Refresh
If FrmSettings.Data1.Recordset.EOF = False Then
   FrmSettings.Data1.Recordset.MoveFirst
   CoyName = FrmSettings.Data1.Recordset![CoyName]
   CoyAdd = FrmSettings.Data1.Recordset![CoyAdd]
   VAT = FrmSettings.Data1.Recordset![VAT]
Else
   MsgBox "No Company Settings found in Database...Some
operations may be abnormal"
End If
End If
Label1.Caption = CoyAdd
Me.Caption = "Authorisation Code"
Data1.Refresh
Exit Sub
handler:
MsgBox Err.Description
End Sub


Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Data1.Refresh
With Data1.Recordset
Do While .EOF = False
If Trim(Text1.Text) = ![Password] And
Trim(DBCombo1.Text) = ![UserName] Then
AccessLevel = ![AccessLevel]
UserName = DBCombo1.Text
FrmMainMenu.Show
Unload FrmAccess
Exit Sub
End If
.MoveNext
Loop
MsgBox "Invalid user"
End With
End If
End Sub

Private Sub Timer1_Timer()
If Label1.Left <= FrmAccess.Left - FrmAccess.Width + 900
Then
Label1.Left = FrmAccess.Width
End If
Label1.Left = Label1.Left - 10
End Sub
Private Sub Timer2_Timer()
If Frame1.Height >= 1695 Then
Command1.Enabled = True
GoTo ending
End If
Frame1.Height = Frame1.Height + 10
ending:
End Sub
```

Code for FrmCAtegory.frm

```
Private Sub cmdAdd_Click()
On Error GoTo handler
 datPrimaryRS.Recordset.AddNew
 Exit Sub
handler:
 MsgBox Err.Description
```

```
End Sub

Private Sub cmdDelete_Click()
On Error GoTo handler
 With datPrimaryRS.Recordset
 RESP = MsgBox("The Current Record would be
Deleted...Continue (Y/N)", vbYesNo + vbInformation)
 If RESP = vbYes Then
   .Delete
   .MoveNext
   If .EOF Then .MoveLast
  End If
  End With
 Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdRefresh_Click()
On Error GoTo handler
 'This is only needed for multi user apps
 datPrimaryRS.Refresh
   Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
On Error GoTo handler
 datPrimaryRS.UpdateRecord
 datPrimaryRS.Recordset.Bookmark =
datPrimaryRS.Recordset.LastModified
 Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdClose_Click()
 Unload Me
End Sub

Private Sub datPrimaryRS_Error(DataErr As Integer,
Response As Integer)
 'This is where you would put error handling code
 'If you want to ignore errors, comment out the next line
 'If you want to trap them, add code here to handle them
 MsgBox "Data error event hit err:" & Error$(DataErr)
 Response = 0  'Throw away the error
End Sub

Private Sub datPrimaryRS_Reposition()
 On Error Resume Next
 'This will synch the grid with the Master recordset
 datSecondaryRS.RecordSource = "select
[code],[Name],[Address] from [customer] where [code]='" &
datPrimaryRS.Recordset![Code] & "'" & " Order by [code]"
 datSecondaryRS.Refresh
 'This will display the current record position for dynasets and
snapshots
 datPrimaryRS.Caption = "Record: " &
(datPrimaryRS.Recordset.AbsolutePosition + 1)
End Sub

Private Sub datPrimaryRS_Validate(Action As Integer,
Save As Integer)
 'This is where you put validation code
 'This event gets called when the following actions occur
 Select Case Action
  Case vbDataActionMoveFirst
  Case vbDataActionMovePrevious
  Case vbDataActionMoveNext
  Case vbDataActionMoveLast
  Case vbDataActionAddNew
  Case vbDataActionUpdate
  Case vbDataActionDelete
  Case vbDataActionFind
  Case vbDataActionBookmark
  Case vbDataActionClose
   ' Screen.MousePointer = vbDefault
 End Select
 'Screen.MousePointer = vbHourglass
End Sub

Private Sub Txtfields_KeyPress(Index As Integer,
KeyAscii As Integer)
If KeyAscii = 13 Then If Index = 0 Then Txtfields(1) =
Left(Txtfields(0), 1)
End Sub
```

Codes for frmCustDetails

```
Private Sub cmdAdd_Click()
On Error GoTo handler
 datPrimaryRS.Refresh
 If datPrimaryRS.Recordset.EOF = False Then
datPrimaryRS.Recordset.MoveLast
 datPrimaryRS.Refresh
 datPrimaryRS.Recordset.AddNew
 txtFields(0).Text = datPrimaryRS.Recordset.RecordCount +
1
 txtFields(1).SetFocus
 Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdDelete_Click()
On Error GoTo handler
 With datPrimaryRS.Recordset
 RESP = MsgBox("The Current Record would be
Deleted...Continue (Y/N)", vbYesNo + vbInformation)
 If RESP = vbYes Then
   .Delete
   .MoveNext
   If .EOF Then .MoveLast
  End If
  End With
 Exit Sub
```

```
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdRefresh_Click()
On Error GoTo handler
 'This is only needed for multi user apps
 datPrimaryRS.Refresh
   Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
On Error GoTo handler
 datPrimaryRS.UpdateRecord
 datPrimaryRS.Recordset.Bookmark =
datPrimaryRS.Recordset.LastModified
 Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdClose_Click()
 Unload Me
End Sub

Private Sub datPrimaryRS_Error(DataErr As Integer,
Response As Integer)
  'This is where you would put error handling code
  'If you want to ignore errors, comment out the next line
  'If you want to trap them, add code here to handle them
  MsgBox "Data error event hit err:" & Error$(DataErr)
  Response = 0  'Throw away the error
End Sub

Private Sub datPrimaryRS_Reposition()
  'Screen.MousePointer = vbDefault
  On Error Resume Next
  'This will synch the grid with the Master recordset
  'datSecondaryRS.RecordSource = "select
[code],[Name],[Address] from [customer] where [code]='" &
datPrimaryRS.Recordset![code] & "'" & " Order by [code]"
  'datSecondaryRS.Refresh
  'This will display the current record position for dynasets and
snapshots
  datPrimaryRS.Caption = "Record: " &
(datPrimaryRS.Recordset.AbsolutePosition + 1)
End Sub

Private Sub datPrimaryRS_Validate(Action As Integer,
Save As Integer)
  'This is where you put validation code
  'This event gets called when the following actions occur
  Select Case Action
    Case vbDataActionMoveFirst
    Case vbDataActionMovePrevious
    Case vbDataActionMoveNext
    Case vbDataActionMoveLast
    Case vbDataActionAddNew
    Case vbDataActionUpdate
    Case vbDataActionDelete
    Case vbDataActionFind
    Case vbDataActionBookmark
    Case vbDataActionClose
      'Screen.MousePointer = vbDefault
  End Select
  'Screen.MousePointer = vbHourglass
End Sub

Private Sub Form_Load()
  'Create the grid's recordset
'  datPrimaryRS.Refresh
End Sub

Private Sub Form_Unload(Cancel As Integer)
  'Screen.MousePointer = vbDefault
End Sub
```

Codes for frmGAccount.frm

```
Private Sub CmbMonth_Click()
If Format(CmbMonth, "mm") = "Jan" Or Format(CmbMonth,
"mm") = "Mar" Or Format(CmbMonth, "mm") = "May" Or
Format(CmbMonth, "mm") = "Jul" Or Format(CmbMonth,
"mm") = "Aug" Or Format(CmbMonth, "mm") = "Oct" Or
Format(CmbMonth, "mm") = "Dec" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = True
End If
If Format(CmbMonth, "mm") = "Sep" Or Format(CmbMonth,
"mm") = "Apr" Or Format(CmbMonth, "mm") = "Jun" Or
Format(CmbMonth, "mm") = "Nov" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = False
End If
If Format(CmbMonth, "mm") = "Feb" Then
If Val(cmbYear.Text) Mod 4 = 0 Or Val(cmbYear.Text) Mod
100 = 0 Then
Day(28).Visible = True
Day(29).Visible = False
Day(30).Visible = False
Else
Day(28).Visible = False
Day(29).Visible = False
Day(30).Visible = False
End If
End If
End Sub


Private Sub cmbYear_Click()
If Format(CmbMonth, "mm") = "Feb" Then
If Val(cmbYear.Text) Mod 4 = 0 Or Val(cmbYear.Text) Mod
100 = 0 Then
Day(28).Visible = True
Day(29).Visible = False
Day(30).Visible = False
```

```
Else
Day(28).Visible = False
Day(29).Visible = False
Day(30).Visible = False
End If
End If
End Sub

Private Sub cmdAdd_Click()
On Error GoTo handler
 Data1.Recordset.AddNew
Exit Sub
handler:
  MsgBox (Error(Err.Number))
End Sub

Private Sub cmdClose_Click()
Unload Me
End Sub

Private Sub cmdDelete_Click()
On Error GoTo handler
 With Data1.Recordset
 RESP = MsgBox("The Current Record would be Deleted
(y/n)?..", vbYesNo + vbCritical)
 If RESP = vbYes Then
    .Delete
   .MoveNext
  If .EOF Then .MoveLast
  End If

 End With
 Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub cmdRefresh_Click()
On Error GoTo handler
 'This is only needed for multi user apps
 Data1.Refresh
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
On Error GoTo handler
 Data1.UpdateRecord
 Data1.Recordset.Bookmark = Data1.Recordset.LastModified
Exit Sub
handler:
MsgBox Err.Description

End Sub

Private Sub Combo3_Click()
If Format(Combo4, "mm") = "Feb" Then
If Val(Combo3.Text) Mod 4 = 0 Or Val(Combo3.Text) Mod
100 = 0 Then
Day(28 + 31).Visible = True
Day(29 + 31).Visible = False
Day(30 + 31).Visible = False
Else
Day(28 + 31).Visible = False
Day(29 + 31).Visible = False
Day(30 + 31).Visible = False
End If
End If
End Sub

Private Sub Combo4_Click()
If Format(Combo4, "mm") = "Jan" Or Format(Combo4,
"mm") = "Mar" Or Format(Combo4, "mm") = "May" Or
Format(Combo4, "mm") = "Jul" Or Format(Combo4, "mm") =
"Aug" Or Format(Combo4, "mm") = "Oct" Or
Format(Combo4, "mm") = "Dec" Then
Day(28 + 31).Visible = True
Day(29 + 31).Visible = True
Day(30 + 31).Visible = True
End If
If Format(Combo4, "mm") = "Sep" Or Format(Combo4,
"mm") = "Apr" Or Format(Combo4, "mm") = "Jun" Or
Format(Combo4, "mm") = "Nov" Then
Day(28 + 31).Visible = True
Day(29 + 31).Visible = True
Day(30 + 31).Visible = False
End If
If Format(Combo4, "mm") = "Feb" Then
If Val(Combo3.Text) Mod 4 = 0 Or Val(Combo3.Text) Mod
100 = 0 Then
Day(28 + 31).Visible = True
Day(29 + 31).Visible = False
Day(30 + 31).Visible = False
Else
Day(28 + 31).Visible = False
Day(29 + 31).Visible = False
Day(30 + 31).Visible = False
End If
End If
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Command2_Click()
On Error GoTo handler
 Data5.UpdateRecord
 Data5.Recordset.Bookmark = Data5.Recordset.LastModified
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub Command3_Click()
If FraCal.Visible = True Then
  FraCal.Visible = False
  Exit Sub
End If
```

```
FraCal.Visible = True
End Sub

Private Sub Command4_Click()
On Error GoTo handler
 'This is only needed for multi user apps
 Data5.Refresh
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub Command5_Click()
On Error GoTo handler
 With Data5.Recordset
 RESP = MsgBox("The Current Record would be Deleted
(y/n)?..", vbYesNo + vbCritical)
 If RESP = vbYes Then
    .Delete
   .MoveNext
   If .EOF Then .MoveLast
   End If

 End With
 Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub Command6_Click()
On Error GoTo handler
 Data5.Recordset.AddNew
Exit Sub
handler:
  MsgBox (Error(Err.Number))
End Sub

Private Sub Command7_Click()
If Frame1.Visible = True Then
  Frame1.Visible = False
  Exit Sub
End If
Frame1.Visible = True
End Sub

Private Sub Day_Click(Index As Integer)

If Index < 31 Then
TheDate = CmbMonth & "/" & Day(Index).Caption & "/" &
cmbYear
Combo1 = TheDate
FraCal.Visible = False
End If
If Index >= 31 Then
TheDate = Combo4 & "/" & Day(Index - 31).Caption & "/" &
Combo3
Combo2 = TheDate
Frame1.Visible = False
End If
End Sub

Private Sub Form_Load()

' For The Calender
For i = 1900 To 2100
cmbYear.AddItem (i)
Combo3.AddItem (i)
Next i
CmbMonth = Format(Now, "mmm")
Combo4 = Format(Now, "mmm")
cmbYear = Format(Now, "yyyy")
Combo3 = Format(Now, "yyyy")
Day(Format(Now, "dd") - 1).Value = True

If Format(CmbMonth, "mm") = "Jan" Or Format(CmbMonth,
"mm") = "Mar" Or Format(CmbMonth, "mm") = "May" Or
Format(CmbMonth, "mm") = "Jul" Or Format(CmbMonth,
"mm") = "Aug" Or Format(CmbMonth, "mm") = "Oct" Or
Format(CmbMonth, "mm") = "Dec" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = True
End If
If Format(CmbMonth, "mm") = "Sep" Or Format(CmbMonth,
"mm") = "Apr" Or Format(CmbMonth, "mm") = "Jun" Or
Format(CmbMonth, "mm") = "Nov" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = False
End If
If Format(CmbMonth, "mm") = "Feb" Then
If Val(Combo3.Text) Mod 4 = 0 Or Val(Combo3.Text) Mod
100 = 0 Then
Day(28).Visible = True
Day(29).Visible = False
Day(30).Visible = False
Else
Day(28).Visible = False
Day(29).Visible = False
Day(30).Visible = False
End If
End If


If Format(Combo4, "mm") = "Jan" Or Format(Combo4,
"mm") = "Mar" Or Format(Combo4, "mm") = "May" Or
Format(Combo4, "mm") = "Jul" Or Format(Combo4, "mm") =
"Aug" Or Format(Combo4, "mm") = "Oct" Or
Format(Combo4, "mm") = "Dec" Then
Day(28 + 31).Visible = True
Day(29 + 31).Visible = True
Day(30 + 31).Visible = True
End If
If Format(Combo4, "mm") = "Sep" Or Format(Combo4,
"mm") = "Apr" Or Format(Combo4, "mm") = "Jun" Or
Format(Combo4, "mm") = "Nov" Then
Day(28 + 31).Visible = True
Day(29 + 31).Visible = True
Day(30 + 31).Visible = False
```

```
End If
If Format(Combo4, "mm") = "Feb" Then
If Val(Combo3.Text) Mod 4 = 0 Or Val(Combo3.Text) Mod
100 = 0 Then
Day(28 + 31).Visible = True
Day(29 + 31).Visible = False
Day(30 + 31).Visible = False
Else
Day(28 + 31).Visible = False
Day(29 + 31).Visible = False
Day(30 + 31).Visible = False
End If
End If

' End The Calender
End Sub
```

__Codes for FrmFinanceReport.frm__

```
Private Sub Command1_Click()
If TheFinancialFlag = 1 Then DailyFinance
If TheFinancialFlag = 2 Then WeeklyFinance
frmFinanceReport.WindowState = vbMinimized
Unload Me
frmView.Show
End Sub

Private Sub Command2_Click()
Unload frmFinanceReport
End Sub

Private Sub Command3_Click()
QuaterlyFinance
frmFinanceReport.WindowState = vbMinimized
Unload Me
frmView.Show
End Sub

Private Sub Command4_Click()
Unload frmFinanceReport
End Sub

Private Sub Command5_Click()
Unload frmFinanceReport
End Sub

Private Sub Command6_Click()
yearlyFinance
frmFinanceReport.WindowState = vbMinimized
Unload Me
frmView.Show
End Sub

Private Sub Command7_Click()
Unload frmFinanceReport
End Sub

Private Sub Command8_Click()
MonthlyFinance
frmFinanceReport.WindowState = vbMinimized
Unload Me
frmView.Show
End Sub

Public Sub DailyFinance()
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
If Combo1.Text <> "" And Combo2.Text <> "" And
Combo3.Text <> "" Then
TheDate = Trim(Combo2.Text) & "/" & Trim(Combo1.Text)
& "/" & Trim(Combo3.Text)
dbs.Execute ("DELETE * FROM TransTrash")
dbs.Execute ("INSERT INTO TransTrash SELECT * FROM
[Transaction] " & _
    "WHERE TranDate = #" & CDate(TheDate) & "#")
RepTitle = "LIST OF TRANSACTION FOR " & TheDate
frmView.CR1.ReportFileName = "c:\Hwares\Trans.rpt"
Else
MsgBox "Invalid Date Specification"
Exit Sub
End If

End Sub

Public Sub yearlyFinance()
On Error GoTo handler
TheDate = Trim(Combo2.Text) & "/" & Trim(Combo1.Text)
& "/" & Trim(Combo3.Text)
If Combo7.Text <> "" Then
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
myquery1 = "Select * From [Transaction] where
datepart('M',[Transaction].TranDate)>1 AND
Datepart('yyyy',[Transaction].TranDate)=" + Combo7.Text
dbs.Execute ("DELETE * FROM TransTrash")
dbs.Execute ("INSERT INTO TransTrash " & myquery1)
RepTitle = "LIST OF TRANSACTION FOR YEAR  " &
Combo7
frmView.CR1.ReportFileName = "c:\Hwares\Trans.rpt"
Else
MsgBox "Invalid Date Specification"
Exit Sub
End If
Exit Sub
handler:
MsgBox Err.Description

End Sub

Public Sub WeeklyFinance()
On Error GoTo handler
TheDate = Trim(Combo2.Text) & "/" & Trim(Combo1.Text)
& "/" & Trim(Combo3.Text)
If Combo1.Text <> "" And Combo2.Text <> "" And
Combo3.Text <> "" Then
    Sqlstr = "Select * From [Transaction] where
datepart('M',[Transaction].TranDate)=" + "'" + Combo2.Text
+ "'"
```

```
        Sqlstr = Sqlstr + " AND
Datepart('yyyy',[Transaction].TranDate)=" + Combo3.Text
        Sqlstr = Sqlstr + " AND
Datepart('d',[Transaction].TranDate)>=" + Combo1.Text
        myquery1 = Sqlstr + " OR
Datepart('d',[Transaction].TranDate)<=" +
Str(Val(Combo1.Text + 7))

Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
If Combo1.Text <> "" And Combo2.Text <> "" And
Combo3.Text <> "" Then
TheDate = Trim(Combo2.Text) & "/" & Trim(Combo1.Text)
& "/" & Trim(Combo3.Text)
dbs.Execute ("DELETE * FROM TransTrash")
dbs.Execute ("INSERT INTO TransTrash " & myquery1)
RepTitle = "LIST OF TRANSACTION FOR WEEK
ENDING  " & TheDate
frmView.CR1.ReportFileName = "c:\Hwares\Trans.rpt"
Else
MsgBox "Invalid Date Specification"
Exit Sub
End If
End If

Exit Sub
handler:
MsgBox Err.Description
Exit Sub

If datForTheReport.Recordset.BOF = False Then
datForTheReport.Recordset.MoveFirst
Do While datForTheReport.Recordset.EOF = False
datForTheReport.Recordset.Delete
datForTheReport.Recordset.MoveNext
Loop
TheDate = Trim(Combo2.Text) & "/" & Trim(Combo1.Text)
& "/" & Trim(Combo3.Text)
If Combo1.Text <> "" And Combo2.Text <> "" And
Combo3.Text <> "" Then
' Query for Other transactions

        Sqlstr = "Select * From [OtherTrans] where
datepart('M',[OtherTrans].TransDate)=" + "" + Combo2.Text
+ ""
        Sqlstr = Sqlstr + " AND
Datepart('yyyy',[OtherTrans].TransDate)=" + Combo3.Text
        Sqlstr = Sqlstr + " AND
Datepart('d',[OtherTrans].TransDate)>=" + Combo1.Text
        myquery1 = Sqlstr + " OR
Datepart('d',[OtherTrans].TransDate)<=" +
Str(Val(Combo1.Text + 7))


' Query for Expenditures

        Sqlstr = "Select * From [Expenditure] where
datepart('M',[Expenditure].TransDate)=" + "" + Combo2.Text
+ ""
        Sqlstr = Sqlstr + " AND
Datepart('yyyy',[Expenditure].TransDate)=" + Combo3.Text
        Sqlstr = Sqlstr + " AND
Datepart('d',[Expenditure].TransDate)>=" + Combo1.Text
        myquery3 = Sqlstr + " AND
Datepart('d',[Expenditure].TransDate)<=" +
Str(Val(Combo1.Text + 7))


' Query for Banks

        Sqlstr = "Select * From [Bank] where
datepart('M',[Bank].TransactionDate)=" + "" + Combo2.Text
+ ""
        Sqlstr = Sqlstr + " AND
Datepart('yyyy',[Bank].TransactionDate)=" + Combo3.Text
        Sqlstr = Sqlstr + " AND
Datepart('d',[Bank].TransactionDate)>=" + Combo1.Text
        myquery4 = Sqlstr + " AND
Datepart('d',[Bank].TransactionDate)<=" +
Str(Val(Combo1.Text + 7))

'For Other Transactions
Set mydata = OpenDatabase("c:\BizMan2000\BizBank.mdb")
If temp <> "E" Then
Set mytab = mydata.OpenRecordset(myquery1)
With mytab
Do While .EOF = False
datForTheReport.Recordset.AddNew
If ![TransType] <> "" Then
datForTheReport.Recordset![TransType] = ![TransType]
If ![TranSpec] <> "" Then
datForTheReport.Recordset![TransDesc] = ![TranSpec]
If ![TransAmount] <> "" Then
datForTheReport.Recordset![TransAmount] =
![TransAmount]
If ![TransTotal] <> "" Then
datForTheReport.Recordset![TransTotal] = ![TransTotal]
If ![TransBy] <> "" Then
datForTheReport.Recordset![TransBy] = ![TransBy]
If ![TransTo] <> "" Then
datForTheReport.Recordset![TransTo] = ![TransTo]
If ![Receipt_no] <> "" Then
datForTheReport.Recordset![Receipt_no] = ![Receipt_no]
datForTheReport.Recordset![TransDate] = TheDate
datForTheReport.Recordset.Update
.MoveNext
Loop
End With
End If

If temp = "E" Then
Set mytab = mydata.OpenRecordset(myquery3)
With mytab
Do While .EOF = False
datForTheReport.Recordset.AddNew
If ![TransType] <> "" Then
datForTheReport.Recordset![TransType] = ![TransType]
'If ![TranSpec] <> "" Then
datForTheReport.Recordset![TransDesc] = ![TranSpec]
If ![TransAmount] <> "" Then
datForTheReport.Recordset![TransTotal] = ![TransAmount]
```

```
f ![TransUnit] <> "" Then
datForTheReport.Recordset![TransAmount] = ![TransUnit]
f ![TransBy] <> "" Then
datForTheReport.Recordset![TransBy] = ![TransBy]
f ![TransTo] <> "" Then
datForTheReport.Recordset![TransTo] = ![TransTo]
f ![Receipt_no] <> "" Then
datForTheReport.Recordset![Receipt_no] = ![Receipt_no]
datForTheReport.Recordset![TransDate] = TheDate
datForTheReport.Recordset.Update
MoveNext
Loop
End With
End If


End If
End Sub

Public Sub MonthlyFinance()
On Error GoTo handler
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
If Combo6.Text <> "" And Combo8.Text <> "" Then
dbs.Execute ("DELETE * FROM TransTrash")
Sqlstr = "Select * From [Transaction] where
datepart('M',[transaction].TranDate)=" +
Str(Combo6.ListIndex + 1) + "AND
Datepart('yyyy',[Transaction].TranDate)=" + Combo8.Text
        myquery1 = Sqlstr + " AND
Datepart('yyyy',[transaction].TranDate)=" + Combo8.Text
dbs.Execute ("INSERT INTO TransTrash " + myquery1)
RepTitle = "LIST OF TRANSACTION FOR " &
UCase(Format$(Combo6, "mm")) & "," & Combo8
frmView.CR1.ReportFileName = "c:\Hwares\Trans.rpt"
Else
MsgBox "Invalid Date Specification"
End If
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub Form_Load()
If mode = 1 Or mode = 2 Or mode = 10 Or mode = 20 Then
Frame1.Visible = True
Frame2.Visible = False
Frame3.Visible = False
Frame4.Visible = False
For i = 1 To 31
Combo1.AddItem i
Next i

Combo2.AddItem "January"
Combo2.AddItem "February"
Combo2.AddItem "March"
Combo2.AddItem "April"
Combo2.AddItem "May"
Combo2.AddItem "June"
Combo2.AddItem "July"
Combo2.AddItem "August"
Combo2.AddItem "September"
Combo2.AddItem "October"
Combo2.AddItem "November"
Combo2.AddItem "December"
For i = 0 To 10
Combo3.AddItem Format$(Now, "yyyy") - i
Next i
End If

If mode = 3 Or mode = 30 Then
Frame1.Visible = False
Frame2.Visible = False
Frame3.Visible = False
Frame4.Visible = True
Combo6.AddItem "January"
Combo6.AddItem "February"
Combo6.AddItem "March"
Combo6.AddItem "April"
Combo6.AddItem "May"
Combo6.AddItem "June"
Combo6.AddItem "July"
Combo6.AddItem "August"
Combo6.AddItem "September"
Combo6.AddItem "October"
Combo6.AddItem "November"
Combo6.AddItem "December"
For i = 0 To 10
Combo8.AddItem Format$(Now, "yyyy") - i
Next i
End If
If mode = 4 Or mode = 40 Then
Frame1.Visible = False
Frame3.Visible = False
Frame4.Visible = False
Frame2.Visible = True
Combo5.AddItem "First"
Combo5.AddItem "Second"
Combo5.AddItem "Third"
Combo5.AddItem "Fourth"
For i = 0 To 10
Combo4.AddItem Format$(Now, "yyyy") - i
Next i
End If
If mode = 5 Or mode = 50 Then
Frame1.Visible = False
Frame2.Visible = False
Frame4.Visible = False
Frame3.Visible = True
For i = 0 To 10
Combo7.AddItem Format$(Now, "yyyy") - i
Next i
End If
End Sub

Public Sub QuaterlyFinance()
On Error GoTo handler
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
If Combo5.Text <> "" And Combo4.Text <> "" Then
    Select Case Combo5.Text
        Case "First"
        ' Query for Other transactions
        Sqlstr = "Select * From [Transaction] where
datepart('M',[Transaction].TranDate)<=3"
        myquery1 = Sqlstr + " AND
Datepart('yyyy',[Transaction].TranDate)=" + Combo4.Text
        Case "Second"
        ' Query for Other transactions
        Sqlstr = "Select * From [Transaction] where
datepart('M',[Transaction].TranDate)>3 and
datepart('M',[Transaction].TranDate)<=6"
        myquery1 = Sqlstr + " AND
Datepart('yyyy',[Transaction.TranDate)=" + Combo4.Text
        Case "Third"
        ' Query for Other transactions
        Sqlstr = "Select * From [Transaction] where
datepart('M',[Transaction].TranDate)>6 and
datepart('M',[Transaction].TranDate)<=9"
        myquery1 = Sqlstr + " AND
Datepart('yyyy',[Transaction].TranDate)=" + Combo4.Text
        Case "Fourth"
        ' Query for Other transactions
        Sqlstr = "Select * From [Transaction] where
datepart('M',[Transaction].TranDate)>9"
        myquery1 = Sqlstr + " AND
Datepart('yyyy',[Transaction].TranDate)=" + Combo4.Text
      End Select
dbs.Execute ("DELETE * FROM TransTrash")
dbs.Execute ("INSERT INTO TransTrash " + myquery1)
RepTitle = "LIST OF TRANSACTION FOR " &
UCase(Combo5) & " QUARTER " & "," & Combo4
frmView.CR1.ReportFileName = "c:\Hwares\Trans.rpt"
Else
MsgBox "Invalid Date Specification"
End If
Exit Sub
handler:
MsgBox Err.Description
End Sub
```

Codes for GenInvoice.frm

```
Private Sub Command1_Click()
Data1.Recordset.FindFirst ("OrderNo=" & "" & Text4 & "")
If Data1.Recordset.NoMatch = True Then
MsgBox "Transaction Details has not been saved"
Command7.SetFocus
Exit Sub
End If
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
dbs.Execute ("DELETE * FROM TransTrash")
dbs.Execute ("INSERT INTO TransTrash SELECT * FROM
Transaction WHERE OrderNo=" & "" & Text4 & "")
CR1.ReportFileName = "c:\Hwares\Invoice.rpt"
CR1.Formulas(0) = "Time=" & "" & Format(Now,
"hh:mm:ss") & ""
CR1.Formulas(1) = "CoyName=" & "" & CoyName & ""
'CR1.Formulas(2) = "CoyAdd=" & "" & CoyAdd & ""
CR1.Destination = crptToPrinter
CR1.PrintReport
End Sub

Private Sub Command2_Click()
FrmViewOrder.Show 1
End Sub


Private Sub Command3_Click()
On Error GoTo handler
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
dbs.Execute ("DELETE * FROM PayDetails WHERE
OrderNo = " & "" & Text4 & "")
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub Command4_Click()
Data1.Recordset.FindFirst ("OrderNo=" & "" & Text4 & "")
If Data1.Recordset.NoMatch = True Then
MsgBox "Transaction Details has not been saved"
Command7.SetFocus
Exit Sub
End If

Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
dbs.Execute ("DELETE * FROM TransTrash")
dbs.Execute ("INSERT INTO TransTrash SELECT * FROM
Transaction WHERE OrderNo=" & "" & Text4 & "")
CR1.ReportFileName = "c:\Hwares\Invoice.rpt"
CR1.Formulas(0) = "Time=" & "" & Format(Now,
"hh:mm:ss") & ""
CR1.Formulas(1) = "CoyName=" & "" & CoyName & ""
'CR1.Formulas(2) = "CoyAdd=" & "" & CoyAdd & ""
CR1.Destination = crptToWindow
CR1.WindowState = crptMaximized
CR1.PrintReport
End Sub

Private Sub Command5_Click()
Me.Width = 4875
Unload Me
End Sub

Private Sub Command7_Click()
On Error GoTo handler
If Option3.Value = True Then
If Txt(0) = "" And Txt(1) = "" And Txt(2) = "" Then
MsgBox "Enter Cheque Information...Saving Aborted"
Txt(0).SetFocus
Exit Sub
End If
End If
Data1.Recordset.AddNew
Data1.Recordset![OrderNo] = Text4
Data1.Recordset![CustCode] = txtFields(1)
Data1.Recordset![Paid] = txtFields(10)
Data1.Recordset![VAT] = txtFields(11)
Data1.Recordset![Bal] = txtFields(9)
Data1.Recordset![BankName] = Txt(0)
```

```
ta1.Recordset![ChequeNo] = Txt(1)
ta1.Recordset![AccountNo] = Txt(2)
ta1.Recordset![Date] = Text1
Option1.Value = True Then
ta1.Recordset![PayType] = "Cash"
e
ta1.Recordset![PayType] = "Cheque"
d If
ta1.Recordset.Update
mmand1.SetFocus
t Sub
ndler:
gBox Err.Description
d Sub

vate Sub Form_Load()
xt1 = Format(Now, "mmm-dd-yyyy")
d Sub

ivate Sub mnu1_Click()
sgBox "For a Credit Customer (i.e -ve Balance) Enter -ve
fset to Reduce Credit  ***** For a Debit Customer (i.e +ve
lance) Enter +ve Offset to Reduce Debit"
nd Sub

rivate Sub Option1_Click()
rame1.Enabled = False
nd Sub

rivate Sub Option3_Click()
rame1.Enabled = True
xt(0).SetFocus
nd Sub

rivate Sub Text2_KeyPress(KeyAscii As Integer)
KeyAscii = 13 Then Call Text2_LostFocus
nd Sub

ivate Sub Text2_LostFocus()
Error GoTo handler
ta3.Recordset.FindFirst ("ORDERNO=" & "'" & Text2 &
)
Data3.Recordset.NoMatch = False Then
ta3.Recordset.Move (0)
ta3.Recordset.Edit
xt3.SetFocus
e
gBox "Such Order No. not existing"
xt2 = ""
d If
t Sub
dler:
gBox Err.Description
d Sub

vate Sub Text3_KeyPress(KeyAscii As Integer)
Error GoTo handler
KeyAscii = 13 Then
```

```
TheStart:
TheNewBal = Val(Text3)
MsgBox TheNewBal


If TheNeBal <> "" Then
Data3.Refresh
'If Data3.Recordset.EOF <> False Then
Data3.Recordset.MoveFirst
If Data3.Recordset![bal] < Val(Text3) Then
TheNewBal = Val(Text3) - Data3.Recordset![bal]
Data3.Recordset.Edit
Data3.Recordset![Paid] = Data3.Recordset![Paid] +
Data3.Recordset![bal]
Data3.Recordset![bal] = 0 'Data3.Recordset![bal] - Val(Text3)
Data3.Recordset.Update
DBGrid1.Refresh
End If
MsgBox TheNewBal
Text3 = TheNewBal
If Val(Text3) <> 0 Then GoTo TheStart
End If
End If

Exit Sub
handler:
MsgBox Err.Description
Exit Sub


'Do While Data3.Recordset.EOF = False
'If Data3.Recordset![bal] < Val(Text3) Then
'TheNewBal = Data3.Recordset![bal] - Val(Text3)
'Data3.Recordset.Edit
'Data3.Recordset![Paid] = Data3.Recordset![Paid] +
Val(Text3)
'Data3.Recordset![bal] = Data3.Recordset![bal] - Val(Text3)
'Data3.Recordset.Update
'Data3.Refresh

 '  If Data3.Recordset.EditMode = 1 Then

 '     DBGrid1.Refresh
 ' End If
'End If
End Sub

Private Sub Text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then Call Text4_LostFocus
End Sub

Private Sub Text4_LostFocus()
Data2.RecordSource = "SELECT * FROM Transaction
WHERE OrderNo=" & "'" & Text4 & "'"
Data2.Refresh
If Data2.Recordset.EOF = True Then
MsgBox "Such Order No. not existing"
Text4 = ""
Else
```

```
Data2.Recordset.FindFirst ("ORDERNO=" & "'" & Text4 &
"'")
If Data2.Recordset.NoMatch = False Then
Data2.Recordset.Move (0)
txtFields(1) = Data2.Recordset![CusCode]
Data2.Refresh
TransSum = 0
Do While Data2.Recordset.EOF = False
If Data2.Recordset![TotalAmount] <> "" Then
TransSum = TransSum +
Val(Data2.Recordset![TotalAmount])
End If
Data2.Recordset.MoveNext
Loop
txtFields(10) = ""
txtFields(10).SetFocus
txtFields(11) = Str((TransSum * VAT) / 100)
txtFields(9) = Str(TransSum + Val(txtFields(11)))
NewTransSum = Val(txtFields(9))
If TransSum = 0 Then
Command1.Enabled = False
Command4.Enabled = False
Command7.Enabled = False
Data3.RecordSource = "SELECT * FROM PayDetails
WHERE CusCode='**********'"
Data3.Refresh
Exit Sub
Else
Command1.Enabled = True
Command4.Enabled = True
Command7.Enabled = True
End If
Else
txtFields(1).Text = " "
MsgBox "No Customer Entry for the Order Number"
End If
End If
```
End Sub

Private Sub TxtFields_Change(Index As Integer)
```
If Index = 1 Then
Data3.RecordSource = "SELECT * FROM PayDetails
WHERE CusCode=" & "'" & txtFields(1) & "'" & " AND bal
<> VAL(0) ORDER BY OrderNo ASC;"
Data3.Refresh
DBGrid1.Refresh
For i = 0 To DBGrid1.ApproxCount - 1
DBGrid1.Row = i
Total = Total + Val(DBGrid1.Columns(2).Text)
Next i
Label8.Caption = Total

If Data3.Recordset.EOF = False Then well = MsgBox("The
Customer is a Debit or Credit Customer...Reconcile Account
(Y/N?", vbYesNo + vbInformation)
If well = vbYes Then
   Me.Width = 8370
Else
   Me.Width = 4875
End If
End If

If Index = 10 Then
txtFields(9) = NewTransSum - Val(txtFields(10))
If txtFields(9) < 0 Then
RESP = MsgBox("Credit Customer Y/N?", vbYesNo +
vbInformation)
If RESP = vbNo Then txtFields(10) = ""
End If
End If
```
End Sub

Private Sub Txtfields_KeyPress(Index As Integer, KeyAscii As Integer)
```
If KeyAscii = 13 And Index = 10 Then Command7.SetFocus
```
End Sub

Private Sub txtFields_LostFocus(Index As Integer)
```
If Index = 7 Then Command10.SetFocus
```
End Sub

Codes for Transaction.frm

Private Sub CmbMonth_Click()
```
If Format(CmbMonth, "mm") = "Jan" Or Format(CmbMonth,
"mm") = "Mar" Or Format(CmbMonth, "mm") = "May" Or
Format(CmbMonth, "mm") = "Jul" Or Format(CmbMonth,
"mm") = "Aug" Or Format(CmbMonth, "mm") = "Oct" Or
Format(CmbMonth, "mm") = "Dec" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = True
End If
If Format(CmbMonth, "mm") = "Sep" Or Format(CmbMonth,
"mm") = "Apr" Or Format(CmbMonth, "mm") = "Jun" Or
Format(CmbMonth, "mm") = "Nov" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = False
End If
If Format(CmbMonth, "mm") = "Feb" Then
If Val(cmbYear.Text) Mod 4 = 0 Or Val(cmbYear.Text) Mod
100 = 0 Then
Day(28).Visible = True
Day(29).Visible = False
Day(30).Visible = False
Else
Day(28).Visible = False
Day(29).Visible = False
Day(30).Visible = False
End If
End If
```
End Sub

Private Sub cmbYear_Change()
```
If Format(CmbMonth, "mm") = "Feb" Then
If Val(cmbYear.Text) Mod 4 = 0 Or Val(cmbYear.Text) Mod
100 = 0 Then
Day(28).Visible = True
Day(29).Visible = False
Day(30).Visible = False
Else
Day(28).Visible = False
Day(29).Visible = False
Day(30).Visible = False
End If
End If
End Sub

Private Sub cmbYear_Click()
If Format(CmbMonth, "mm") = "Feb" Then
If Val(cmbYear.Text) Mod 4 = 0 Or Val(cmbYear.Text) Mod
100 = 0 Then
Day(28).Visible = True
Day(29).Visible = False
Day(30).Visible = False
Else
Day(28).Visible = False
Day(29).Visible = False
Day(30).Visible = False
End If
End If
End Sub

Private Sub Command1_Click()
On Error GoTo handler
mode = 111
wol = InputBox("Enter Order Number", "Quick Find Order")
Set dbs = OpenDatabase("c:\hwares\homewares.mdb")
Set rstTemp = dbs.OpenRecordset("SELECT * FROM
Transaction WHERE OrderNo =" & "'" & wol & "'")
dbs.Recordsets.Refresh
If rstTemp.EOF = False Then
'rstTemp.MoveLast
datPrimaryRS.RecordSource = "SELECT * FROM
Transaction WHERE OrderNo =" & "'" & wol & "'"
datPrimaryRS.Refresh
Else
MsgBox "No Match", vbInformation
End If
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub Command2_Click()
FrmViewItem.Show 1
Call txtFields_LostFocus(2)
End Sub

Private Sub Command3_Click()
FrmViewCust.Show 1
End Sub

Private Sub Command4_Click()
If FraCal.Visible = True Then
  FraCal.Visible = False
  Exit Sub
End If
FraCal.Visible = True
End Sub

Private Sub Command5_Click()
FrmGenInvoice.Text4 = TxtFields(0)
FrmGenInvoice.Show 1
End Sub

Private Sub datPrimaryRS_Error(DataErr As Integer,
Response As Integer)
 'This is where you would put error handling code
 'If you want to ignore errors, comment out the next line
 'If you want to trap them, add code here to handle them
 MsgBox "Data error event hit err:" & Error$(DataErr)
 Response = 0  'Throw away the error
End Sub

Private Sub datPrimaryRS_Reposition()
 'Screen.MousePointer = vbDefault
 On Error Resume Next
 'This will synch the grid with the Master recordset

 'datSecondaryRS.RecordSource = "select
[Icode],[UnitInStock],[Quantity],[TranDate] from
[Transaction] where [OrderNo]='" &
datPrimaryRS.Recordset![OrderNo] & "'" & " Order by
[Icode]"
 'datSecondaryRS.Refresh

 'This will display the current record position for dynasets and
snapshots
 datPrimaryRS.Caption = "Record: " &
(datPrimaryRS.Recordset.AbsolutePosition + 1)
End Sub

Private Sub datPrimaryRS_Validate(Action As Integer,
Save As Integer)
 'This is where you put validation code
 'This event gets called when the following actions occur
 Select Case Action
  Case vbDataActionMoveFirst
  Case vbDataActionMovePrevious
  Case vbDataActionMoveNext
  Case vbDataActionMoveLast
  Case vbDataActionAddNew
  Case vbDataActionUpdate
  Case vbDataActionDelete
  Case vbDataActionFind
  Case vbDataActionBookmark
  Case vbDataActionClose
   ' Screen.MousePointer = vbDefault
 End Select
 'Screen.MousePointer = vbHourglass
End Sub

Private Sub Day_Click(Index As Integer)
TheDate = CmbMonth & "/" & Day(Index).Caption & "/" &
cmbYear
TxtFields(6) = TheDate
FraCal.Visible = False
```

```
End Sub


Private Sub Form_Load()
mode = 1
Label2.Caption = CoyName
datPrimaryRS.RecordSource = "SELECT * FROM
[Transaction] WHERE Icode='*****&&&*******'"
datPrimaryRS.Refresh

' For The Calender
For i = 1900 To 2100
cmbYear.AddItem (i)
Next i
CmbMonth = Format(Now, "mmm")
cmbYear = Format(Now, "yyyy")
Day(Format(Now, "dd") - 1).Value = True
If Format(CmbMonth, "mm") = "Jan" Or Format(CmbMonth,
"mm") = "Mar" Or Format(CmbMonth, "mm") = "May" Or
Format(CmbMonth, "mm") = "Jul" Or Format(CmbMonth,
"mm") = "Aug" Or Format(CmbMonth, "mm") = "Oct" Or
Format(CmbMonth, "mm") = "Dec" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = True
End If
If Format(CmbMonth, "mm") = "Sep" Or Format(CmbMonth,
"mm") = "Apr" Or Format(CmbMonth, "mm") = "Jun" Or
Format(CmbMonth, "mm") = "Nov" Then
Day(28).Visible = True
Day(29).Visible = True
Day(30).Visible = False
End If
If Format(CmbMonth, "mm") = "Feb" Then
If Val(cmbYear.Text) Mod 4 = 0 Or Val(cmbYear.Text) Mod
100 = 0 Then
Day(28).Visible = True
Day(29).Visible = False
Day(30).Visible = False
Else
Day(28).Visible = False
Day(29).Visible = False
Day(30).Visible = False
End If
End If

' End The Calender

End Sub

Private Sub Form_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)
mode = 0
End Sub

Private Sub grdDataGrid_DblClick()
On Error GoTo handler
grdDataGrid.Col = 0
mode = 111
wol = grdDataGrid.Text
Set dbs = OpenDatabase("c:\hwares\homewares.mdb")
Set rstTemp = dbs.OpenRecordset("SELECT * FROM
Transaction WHERE OrderNo =" & "'" & wol & "'")
dbs.Recordsets.Refresh
If rstTemp.EOF = False Then
'rstTemp.MoveLast
datPrimaryRS.RecordSource = "SELECT * FROM
Transaction WHERE OrderNo =" & "'" & wol & "'"
datPrimaryRS.Refresh
Else
MsgBox "No Match", vbInformation
End If
Exit Sub
handler:
MsgBox Err.Description

End Sub

Private Sub TB1_ButtonClick(ByVal Button As
MSComctlLib.Button)
On Error GoTo handler
mode = 1
Select Case Button.Tag
Case "1"
Set dbs = OpenDatabase("c:\hwares\homewares.mdb")
Set rstTemp = dbs.OpenRecordset("SELECT DISTINCT
OrderNo FROM transaction ORDER BY OrderNo ASC;")
dbs.Recordsets.Refresh
If rstTemp.EOF = False Then rstTemp.MoveLast
TheOrderNo = rstTemp.RecordCount + 1
datPrimaryRS.RecordSource = "SELECT * FROM
[Transaction] WHERE OrderNo=" & "'" & TheOrderNo & "'"
datPrimaryRS.Refresh
Case "2"
  TransNew
  mode = 1
Case "3"
  TransUpdate
  mode = 2
Case "4"
 datPrimaryRS.Refresh
Case "5"
 RESP = MsgBox("The Current Transaction would be
Deleted ..continue (Y/N)", vbYesNo + vbInformation)
 If RESP = vbYes Then
 Set dbs = OpenDatabase("c:\hwares\homewares.mdb")
 Set rstTemp = dbs.OpenRecordset("SELECT * FROM
ItemInvent WHERE Icode =" & "'" & TxtFields(2) & "'")
 dbs.Recordsets.Refresh
 Do While rstTemp.EOF = False
   rstTemp.Edit
   rstTemp![unitinstock] = rstTemp![unitinstock] +
TxtFields(5)
   rstTemp.Update
 rstTemp.MoveNext
 Loop

 dbs.Recordsets.Refresh
 With datPrimaryRS.Recordset
  .Delete
```

```
.MoveNext
If .EOF Then .MoveLast
End With
nd If
ase "6"
Unload Me
nd Select
xit Sub
andler:
MsgBox Err.Description

End Sub

Private Sub Timer1_Timer()
If Label2.Left <= frmTransaction.Left - frmTransaction.Width
- 900 Then
abel2.Left = frmTransaction.Width
End If
abel2.Left = Label2.Left - 10
End Sub

Private Sub TxtFields_Change(Index As Integer)

If mode <> 111 Then
If TxtFields(5) <> "" Then
If Val(TxtFields(5)) < 0 Then
MsgBox "Cannot be Negative"
TxtFields(5) = ""
Exit Sub
End If
End If

If Index = 5 Then
TxtFields(3) = Val(TheCurrentStock) - Val(TxtFields(5))
If Val(TxtFields(3)) < Val(TxtFields(4)) Then
MsgBox "Below Re-Order Level"
TxtFields(5) = ""
TxtFields(3) = TheCurrentStock
Exit Sub
End If
nd If

f Index = 5 Then
f TxtFields(Index) <> "" Then
xtFields(8) = SellingPrice * Val(TxtFields(5))
nd If
nd If
nd If

End Sub

Private Sub Txtfields_KeyPress(Index As Integer,
KeyAscii As Integer)
KeyAscii = 13 And Index = 2 Then Call
tFields_LostFocus(2)
KeyAscii = 13 And Index = 1 Then Call
tFields_LostFocus(1)
KeyAscii = 13 And Index = 5 Then
 If mode = 1 Then
   mode = 2
     TransUpdate
     Exit Sub
  End If
  If mode = 2 Then
     mode = 1
     TransNew
  End If
End If
End Sub

Private Sub txtFields_LostFocus(Index As Integer)
On Error GoTo handler
Set dbs = OpenDatabase("c:\hwares\homewares.mdb")

If Index = 1 And TxtFields(1) <> "" Then
Set rstTemp = dbs.OpenRecordset("SELECT * FROM
Customer WHERE CusCode =" & "'" & TxtFields(Index) &
"'")
dbs.Recordsets.Refresh
rstTemp.FindFirst ("CusCode =" & "'" & TxtFields(Index) &
"'")
If rstTemp.NoMatch = True Then
RESP = MsgBox("Such Customer Code Record do not
exist...Register Now (Y/N)?", vbYesNo + vbInformation)
If RESP = vbYes Then frmCustDetails.Show 1
Else
Label3.Caption = rstTemp![Name]
TxtFields(2).SetFocus
End If
End If

If Index = 2 And TxtFields(2) <> "" Then
Set rstTemp = dbs.OpenRecordset("SELECT * FROM
ItemInvent WHERE Icode =" & "'" & TxtFields(Index) & "'")
dbs.Recordsets.Refresh
rstTemp.FindFirst ("Icode =" & "'" & TxtFields(Index) & "'")
If rstTemp.NoMatch = False Then
rstTemp.Move (0)
Label4.Caption = rstTemp![Desc]
SellingPrice = rstTemp![unitSellprice]
TxtFields(3) = rstTemp![unitinstock]
TheCurrentStock = rstTemp![unitinstock]
TxtFields(4) = rstTemp![reorder]
TxtFields(5).SetFocus
Else
MsgBox ("No Item Details not Registered")
End If
End If
Exit Sub
handler:
MsgBox Err.Description
'If Index = 5 Then Call txtFields_KeyPress(5, 13)
End Sub

Public Sub TransUpdate()
On Error GoTo handler
Set dbs = OpenDatabase("c:\hwares\homewares.mdb")
 Set rstTemp = dbs.OpenRecordset("SELECT * FROM
ItemInvent WHERE Icode =" & "'" & TxtFields(2) & "'")
 dbs.Recordsets.Refresh
 rstTemp.FindFirst ("Icode =" & "'" & TxtFields(2) & "'")
 If rstTemp.NoMatch = False Then
  rstTemp.Move (0)
  rstTemp.Edit
  rstTemp![unitinstock] = TxtFields(3)
  rstTemp.Update
 End If
 datPrimaryRS.UpdateRecord
 datPrimaryRS.Recordset.Bookmark =
datPrimaryRS.Recordset.LastModified
 Set rstTemp = dbs.OpenRecordset("SELECT * FROM
ItemInvent WHERE Icode ='************$$$***'")
 dbs.Recordsets.Refresh
 datSecondaryRS.RecordSource = "select * from
[Transaction] where [OrderNo]='" &
datPrimaryRS.Recordset![OrderNo] & "'" & " Order by
[Icode]"
 datSecondaryRS.Refresh
Exit Sub
handler:
MsgBox Err.Description
End Sub

Public Sub TransNew()
On Error GoTo handler
 If TxtFields(0) <> "" Then TheOrderNo = TxtFields(0)
 If TxtFields(1) <> "" Then TheCust = TxtFields(1)
 If TxtFields(6) <> "" Then TheDate = TxtFields(6)
 TxtFields(0) = ""
 TxtFields(1) = ""
 TxtFields(6) = ""
 datPrimaryRS.Refresh
 datPrimaryRS.Recordset.AddNew
 If TxtFields(6) = "" Then
 TxtFields(6) = Format(Date, "mmm/dd/yyyy")
 Else
 TxtFields(6) = TheDate
 End If
 TxtFields(0) = TheOrderNo
 TxtFields(7) = UserName
 TxtFields(1) = TheCust
 TxtFields(3) = ""
 TxtFields(4) = ""
 TxtFields(8) = ""
 TxtFields(1).SetFocus
Exit Sub
handler:
MsgBox Err.Description
End Sub
```

Codes for FrmInvent.frm

```
Private Sub cmdAdd_Click()
On Error GoTo handler
 datPrimaryRS.Recordset.AddNew
 Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdDelete_Click()
On Error GoTo handler
 With datPrimaryRS.Recordset
 RESP = MsgBox("The Current Record would be
Deleted...Continue (Y/N)", vbYesNo + vbInformation)
 If RESP = vbYes Then
  .Delete
  .MoveNext
  If .EOF Then .MoveLast
  End If
  End With
 Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdRefresh_Click()
On Error GoTo handler
 'This is only needed for multi user apps
 datPrimaryRS.Refresh
   Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
On Error GoTo handler
GenCode
 datPrimaryRS.UpdateRecord
 datPrimaryRS.Recordset.Bookmark =
datPrimaryRS.Recordset.LastModified
 Exit Sub
handler:
 MsgBox Err.Description
End Sub

Private Sub cmdClose_Click()
 Unload Me
End Sub

Private Sub Command1_Click()
FrmViewItem.Show 1
End Sub

Private Sub Command2_Click()
On Error GoTo handler
wol = InputBox("Enter Quantity", "Add to current existing
stock")
If wol <> "" Then
datPrimaryRS.Recordset.Edit
txtFields(3) = Val(txtFields(3)) + Val(wol)
End If
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub Command3_Click()
```

```
On Error GoTo handler
wol = InputBox("Enter Item Description", "Quick Find Item")
datPrimaryRS.Recordset.FindFirst ("Desc=" & "'" & wol &
"'")
If datPrimaryRS.Recordset.NoMatch = False Then
datPrimaryRS.Recordset.Move (0)
Else
MsgBox "No Match", vbInformation
End If
Exit Sub
handler:
MsgBox Err.Description
```
End Sub

Private Sub datPrimaryRS_Error(DataErr As Integer, Response As Integer)
```
 'This is where you would put error handling code
 'If you want to ignore errors, comment out the next line
 'If you want to trap them, add code here to handle them
 MsgBox "Data error event hit err:" & Error$(DataErr)
 Response = 0  'Throw away the error
```
End Sub

Private Sub datPrimaryRS_Reposition()
```
 'Screen.MousePointer = vbDefault
 On Error Resume Next
 'This will synch the grid with the Master recordset
 datSecondaryRS.RecordSource = "select
[code],[Name],[Address] from [customer] where [code]='" &
datPrimaryRS.Recordset![Code] & "'" & " Order by [code]"
 datSecondaryRS.Refresh
 'This will display the current record position for dynasets and
snapshots
 datPrimaryRS.Caption = "Record: " &
(datPrimaryRS.Recordset.AbsolutePosition + 1)
```
End Sub

Private Sub datPrimaryRS_Validate(Action As Integer, Save As Integer)
```
 'This is where you put validation code
 'This event gets called when the following actions occur
 Select Case Action
   Case vbDataActionMoveFirst
   Case vbDataActionMovePrevious
   Case vbDataActionMoveNext
   Case vbDataActionMoveLast
   Case vbDataActionAddNew
   Case vbDataActionUpdate
   Case vbDataActionDelete
   Case vbDataActionFind
   Case vbDataActionBookmark
   Case vbDataActionClose
     'Screen.MousePointer = vbDefault
 End Select
 'Screen.MousePointer = vbHourglass
```
End Sub

Private Sub DBCombo1_LostFocus()
```
On Error GoTo handler
Data1.Recordset.FindFirst ("Name=" & "'" &
DBCombo1.Text & "'")
If Data1.Recordset.NoMatch = True Then
RESP = MsgBox("The Category *** " & DBCombo1 & "'" &
" Does not Exist Do you want it Registered (Y/N)?", vbYesNo
+ vbInformation)
If RESP = vbYes Then
Data1.Recordset.AddNew
Data1.Recordset![Name] = DBCombo1
Data1.Recordset.Update
Data1.Refresh
Else
DBCombo1 = ""
DBCombo1.SetFocus
End If
End If
Exit Sub
handler:
MsgBox Err.Description
End Sub
Private Sub TxtFields_Change(Index As Integer)
If Index = 5 Then
If txtFields(5) <> "" Then
If Val(txtFields(5)) < 0 Then
MsgBox "Negative Re-Order Level"
txtFields(5) = ""
End If
End If
End If
```
End Sub

Public Sub GenCode()
```
Set dbs = OpenDatabase("c:\hwares\homewares.mdb")
Set rstTemp = dbs.OpenRecordset("SELECT * FROM
ItemInvent WHERE Category =" & "'" & DBCombo1 & "'")
dbs.Recordsets.Refresh
If rstTemp.EOF = False Then rstTemp.MoveLast
Data1.Recordset.FindFirst ("Name =" & "'" & DBCombo1 &
"'")
'MsgBox Data1.Recordset.NoMatch
'Data1.Recordset.Move (0)
If Data1.Recordset.NoMatch = False Then
Data1.Recordset.Move (0)
TheCategoryCode = Data1.Recordset![Code]
TheCode = TheCategoryCode & rstTemp.RecordCount + 1
txtFields(0).Text = TheCode
Else
MsgBox "The Category not registered"
End If
```
End Sub

<u>Codes for FrmMainMenu.frm</u>

Private Sub Command1_Click()
```
FrmInvent.SSTab1.Tab = 1
FrmInvent.SSTab1.TabVisible(1) = True
FrmInvent.SSTab1.TabVisible(0) = False
FrmInvent.Show 1
```

```
On Error GoTo handler
wol = InputBox("Enter Item Description", "Quick Find Item")
datPrimaryRS.Recordset.FindFirst ("Desc=" & "'" & wol &
"'")
If datPrimaryRS.Recordset.NoMatch = False Then
datPrimaryRS.Recordset.Move (0)
Else
MsgBox "No Match", vbInformation
End If
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub datPrimaryRS_Error(DataErr As Integer,
Response As Integer)
 'This is where you would put error handling code
 'If you want to ignore errors, comment out the next line
 'If you want to trap them, add code here to handle them
 MsgBox "Data error event hit err:" & Error$(DataErr)
 Response = 0  'Throw away the error
End Sub

Private Sub datPrimaryRS_Reposition()
 'Screen.MousePointer = vbDefault
 On Error Resume Next
 'This will synch the grid with the Master recordset
 datSecondaryRS.RecordSource = "select
[code],[Name],[Address] from [customer] where [code]='" &
datPrimaryRS.Recordset![Code] & "'" & " Order by [code]"
 datSecondaryRS.Refresh
 'This will display the current record position for dynasets and
snapshots
 datPrimaryRS.Caption = "Record: " &
(datPrimaryRS.Recordset.AbsolutePosition + 1)
End Sub

Private Sub datPrimaryRS_Validate(Action As Integer,
Save As Integer)
 'This is where you put validation code
 'This event gets called when the following actions occur
 Select Case Action
   Case vbDataActionMoveFirst
   Case vbDataActionMovePrevious
   Case vbDataActionMoveNext
   Case vbDataActionMoveLast
   Case vbDataActionAddNew
   Case vbDataActionUpdate
   Case vbDataActionDelete
   Case vbDataActionFind
   Case vbDataActionBookmark
   Case vbDataActionClose
    'Screen.MousePointer = vbDefault
 End Select
 'Screen.MousePointer = vbHourglass
End Sub

Private Sub DBCombo1_LostFocus()
On Error GoTo handler
Data1.Recordset.FindFirst ("Name=" & "'" &
DBCombo1.Text & "'")
If Data1.Recordset.NoMatch = True Then
RESP = MsgBox("The Category *** " & DBCombo1 & "'" &
" Does not Exist Do you want it Registered (Y/N)?", vbYesNo
+ vbInformation)
If RESP = vbYes Then
Data1.Recordset.AddNew
Data1.Recordset![Name] = DBCombo1
Data1.Recordset.Update
Data1.Refresh
Else
DBCombo1 = ""
DBCombo1.SetFocus
End If
End If
Exit Sub
handler:
MsgBox Err.Description
End Sub
Private Sub TxtFields_Change(Index As Integer)
If Index = 5 Then
If txtFields(5) <> "" Then
If Val(txtFields(5)) < 0 Then
MsgBox "Negative Re-Order Level"
txtFields(5) = ""
End If
End If
End If
End Sub

Public Sub GenCode()
Set dbs = OpenDatabase("c:\hwares\homewares.mdb")
Set rstTemp = dbs.OpenRecordset("SELECT * FROM
ItemInvent WHERE Category =" & "'" & DBCombo1 & "'")
dbs.Recordsets.Refresh
If rstTemp.EOF = False Then rstTemp.MoveLast
Data1.Recordset.FindFirst ("Name =" & "'" & DBCombo1 &
"'")
'MsgBox Data1.Recordset.NoMatch
'Data1.Recordset.Move (0)
If Data1.Recordset.NoMatch = False Then
Data1.Recordset.Move (0)
TheCategoryCode = Data1.Recordset![Code]
TheCode = TheCategoryCode & rstTemp.RecordCount + 1
txtFields(0).Text = TheCode
Else
MsgBox "The Category not registered"
End If
End Sub
```

Codes for FrmMainMenu.frm

```
Private Sub Command1_Click()
FrmInvent.SSTab1.Tab = 1
FrmInvent.SSTab1.TabVisible(1) = True
FrmInvent.SSTab1.TabVisible(0) = False
FrmInvent.Show 1
```

```
End Sub

Private Sub Command10_Click()
Frame5.Visible = True
mode = 3
End Sub

Private Sub Command12_Click()
FrmPayrollInfor.Show 1
End Sub

Private Sub Command13_Click()
Frame6.Visible = False
End Sub

Private Sub Command15_Click()
FrmGAccount.SSTab1.Tab = 1
FrmGAccount.Show 1
End Sub

Private Sub Command16_Click()
FrmGAccount.SSTab1.Tab = 0
FrmGAccount.Show 1
End Sub

Private Sub Command18_Click()
Index = 6
Call Label5_Click(6)
End Sub

Private Sub Command2_Click()
FrmInvent.SSTab1.Tab = 0
FrmInvent.SSTab1.TabVisible(0) = True
FrmInvent.SSTab1.TabVisible(1) = False
FrmInvent.Show 1
End Sub

Private Sub Command21_Click()
mode = 1
FrmModifyInvent.Data1.RecordSource = "SELECT * FROM
[Invent]WHERE StockType = 'Non-Fixed'"
FrmModifyInvent.Data1.Refresh
FrmModifyInvent.Show 1
End Sub

Private Sub Command22_Click()
frmItemInvent.Show 1
End Sub

Private Sub Command24_Click()
frmCategory.Show 1
End Sub

Private Sub Command25_Click()
frmCustDetails.Show 1
End Sub

Private Sub Command28_Click()
Index = 6
Call Label5_Click(6)
```

```
End Sub

Private Sub Command3_Click()
Frame5.Visible = True
mode = 2
End Sub

Private Sub Command4_Click()
FrmStaffRepDialog.Show 1
End Sub

Private Sub Command5_Click()
FrmStudRepDialog.Show 1
End Sub

Private Sub Command6_Click()
Frame5.Visible = True
mode = 1
End Sub

Private Sub Command7_Click()
frmTransaction.Show 1
End Sub

Private Sub Command8_Click()
RESP = InputBox("Enter Form Number ?", "Form
Validation")
wole = RESP
frmStudForm.datPrimaryRS.Refresh
frmStudForm.datPrimaryRS.Recordset.FindFirst ("FormNo="
& "'" & RESP & "'")
If frmStudForm.datPrimaryRS.Recordset.NoMatch = True
Then
well = MsgBox("Form Number does not Exist", vbOKCancel
+ vbInformation)
Exit Sub
Else
FrmRegister.datPrimaryRS.RecordSource = "SELECT *
FROM [StudRec] WHERE FormNo = " & "'" & RESP & "'"
FrmRegister.datPrimaryRS.Refresh
FrmRegister.datPrimaryRS.Recordset.AddNew
FrmRegister.txtForm.Text = RESP
FrmRegister.datPrimaryRSF.Recordset.AddNew
FrmRegister.datPrimaryRSM.Recordset.AddNew
FrmRegister.datPrimaryRSO.Recordset.AddNew
FrmRegister.cmdAdd.Enabled = False
FrmRegister.Show 1
End If
End Sub

Private Sub Command9_Click()
cboselectmonth.Clear
cboselectmonth.Text = Format(Now, " dd-mmm-yyyy")
FrmPayrollInfor.Show 1
Data4.Refresh
With Data4.Recordset
cbobanks.Clear
cbobanks.AddItem ("All Staff")
Do While .EOF = False
cbobanks.AddItem ![staffnum]
```

```
veNext
p
 With
electMonth.Visible = True
a5.Refresh
While Data5.Recordset.EOF = False
15.Recordset.Delete
15.Recordset.MoveNext
p
Sub
 Sub

ate Sub Form_Load()
ccessLevel <> 1 Then
1.Visible = False
To.Visible = False
f
tusBar.Panels(2).Text = UserName
aption = CoyName
9.Caption = CoyName
Sub

ate Sub Label5_Click(Index As Integer)
ex = 6 Then
P = MsgBox("Exiting " & CoyName & "...Continue
", vbYesNo + vbInformation)
SP = vbYes Then
d

t Sub
f
f
Sub

ate Sub Label5_MouseMove(Index As Integer, Button
teger, Shift As Integer, X As Single, Y As Single)
 ex <= 2 Then
 6.Visible = True
 7.Visible = False
 4.Visible = False
 5.Visible = False

 x = 3 Or Index = 4 Then
 .Visible = False
 .Visible = False
 .Visible = False
 .Visible = True

 = 5 Then
 Visible = False
 Visible = True
 Visible = False
 Visible = False

 = 6 Then
 Visible = False
 Visible = False
 Visible = True
 Visible = False
```

```
End If


If Index = 0 Then
Frame1.Visible = True
Frame2.Visible = False
'Frame3.Visible = False
Frame4.Visible = False
Frame7.Visible = False
Frame8.Visible = False
Frame9.Visible = False
End If

If Index = 1 Then
Frame4.Visible = True
'Frame3.Visible = False
Frame2.Visible = False
Frame1.Visible = False
Frame7.Visible = False
Frame8.Visible = False
Frame9.Visible = False
End If

If Index = 2 Then
'Frame3.Visible = False
Frame2.Visible = False
Frame4.Visible = False
Frame1.Visible = False
Frame7.Visible = False
Frame8.Visible = True
Frame9.Visible = False
End If

If Index = 3 Then
Frame2.Visible = True
Frame1.Visible = False
'Frame3.Visible = False
Frame4.Visible = False
Frame7.Visible = False
Frame8.Visible = False
Frame9.Visible = False
End If

If Index = 4 Then
'Frame3.Visible = False
Frame2.Visible = False
Frame1.Visible = False
Frame4.Visible = False
Frame7.Visible = True
Frame8.Visible = False
Frame9.Visible = False
End If

If Index = 5 Then
'Frame3.Visible = False
Frame2.Visible = False
Frame1.Visible = False
Frame7.Visible = False
Frame4.Visible = False
Frame8.Visible = True
```

```
Frame9.Visible = False
End If

If Index = 6 Then
'Frame3.Visible = False
Frame2.Visible = False
Frame4.Visible = False
Frame1.Visible = False
Frame7.Visible = False
Frame8.Visible = False
Frame9.Visible = True
End If

Label5(Index).ForeColor = vbYellow
For i = 0 To 6
If i <> Index Then Label5(i).ForeColor = vbRed
Next i
End Sub

Private Sub mnu3_Click()
temp = ""
mode = 1
TheFinancialFlag = 1
frmFinanceReport.Show
End Sub

Private Sub mnu4_Click()
temp = ""
mode = 2
TheFinancialFlag = 2
frmFinanceReport.Show
End Sub

Private Sub mnu5_Click()
temp = ""
mode = 3
TheFinancialFlag = 3
frmFinanceReport.Show

End Sub

Private Sub mnu6_Click()
temp = ""
mode = 4
TheFinancialFlag = 4
frmFinanceReport.Show

End Sub

rivate Sub mnu7_Click()
emp = ""
node = 5
heFinancialFlag = 5
mFinanceReport.Show
nd Sub

rivate Sub mnuAB_Click()
nAbout.Show 1
nd Sub

Private Sub mnuC_Click()
On Error GoTo handler
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
myquery1 = "Select * From PayDetails WHERE bal < 0 "
dbs.Execute ("DELETE * FROM PayDTrash")
dbs.Execute ("INSERT INTO PayDTrash " & myquery1)
RepTitle = "LIST OF ALL CREDIT CUSTOMERS"
frmView.CR1.Formulas(4) = "Desc =" & "'Total Credit'"
frmView.CR1.ReportFileName = "c:\Hwares\cusstatus.rpt"
frmView.Show 1
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub mnuCL_Click()
frmView.CR1.ReportFileName = "c:\Hwares\customer.rpt"
RepTitle = "LIST OF ALL CUSTOMERS"
frmView.Show 1
End Sub

Private Sub mnuD_Click()
On Error GoTo handler
Set dbs = OpenDatabase("c:\Hwares\Homewares.mdb")
myquery1 = "Select * From PayDetails WHERE bal > 0 "
dbs.Execute ("DELETE * FROM PayDTrash")
dbs.Execute ("INSERT INTO PayDTrash " & myquery1)
RepTitle = "LIST OF ALL DEBIT CUSTOMERS"
frmView.CR1.Formulas(4) = "Desc =" & "'Total Debit'"
frmView.CR1.ReportFileName = "c:\Hwares\cusstatus.rpt"
frmView.Show 1
Exit Sub
handler:
MsgBox Err.Description
End Sub

Private Sub mnui2_Click()
frmView.CR1.ReportFileName = "c:\Hwares\itemlist.rpt"
RepTitle = "LIST OF ALL ITEM WITH THEIR
CATEGORY"
frmView.Show 1
End Sub

Private Sub mnuI3_Click()

frmView.CR1.ReportFileName = "c:\Hwares\category.rpt"
RepTitle = "LIST OF ITEM CATEGORY"
frmView.Show 1
End Sub

Private Sub mnuI4_Click()
frmView.CR1.ReportFileName = "c:\Hwares\iteminvent.rpt"
RepTitle = "LIST OF CURRENT INVENTORY OF ITEMS"
frmView.Show 1
End Sub

Private Sub mnuPAD_Click()
frmPass.Show 1
End Sub
```

```
ivate Sub mnupf_Click()
pTitle = "PROPOSED PROFIT ON CURRENT STOCK"
nView.CR1.ReportFileName = "c:\Hwares\profit1.rpt"
nView.Show 1
nd Sub

rivate Sub mnuPI_Click()
mGenInvoice.Show
nd Sub

rivate Sub mnuSS_Click()
mSettings.Show 1
nd Sub

rivate Sub Timer2_Timer()
Label9.Left <= FrmMainMenu.Left - FrmMainMenu.Width
000 Then
abel9.Left = FrmMainMenu.Width
nd If
abel9.Left = Label9.Left - 10
nd Sub
```