

NEAR EAST UNIVERSITY



Faculty of Engineering

Department of Computer Engineering

**GRADUATION PROJECT
COM – 400**

STOCK CONTROL MANAGEMENT

Student : Selman AKÇAALAN

**Supervisor : ASSOC. PROF. DR. RAHİB
ABİYEV**

Nicosia - 2004

ACKNOWLEDGEMENT

Firstly I would like to thank my dear parents who helped me until this moment.

Secondly I would like to thank all my instructor and all my friends.

Thirdly I want to thank all my friends who helped for this project.

Expecially I want to thank my supervisor who is Assoc. Prof. Dr. Rahib Abiyev for his infinite helpness while I was prepearing this project and his kinds.

ABSTRACT

As the information age has effected every aspect of our life, the need for computerizing many information systems has raised.

Once of the important branches that are effected by information revolution is the computer programming languages.

This project is concerned about using computer program in stock control management system. It is written using Visual Basic 6.0 programming language and used Microsoft Access Database language for databases. Visual Basic is one of the best and easy programming languages.

This project is accomplish stock control and customer management program, that covers all services needed in most firm, such as stock records, customer records, debt of customer records, debt of firm records, and many other stock management services.

Before coming to this point, this project has gone through some important steps:

- First one that I had to have some knowlegde about how to works stock control and customer management programs.
- Second step was to design and to put in order informations about the program.
- The later steps were steps of the implementation of the designed information on computer by using Visual Basic Language.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	1
ABSTRACT.....	2
TABLE OF CONTENTS.....	3
HISTORY OF VISUAL BASIC.....	4
INTRODUCTION	11
INTRODUCTION TO STOCK CONTROL	13
DATABASE STRUCTURE.....	14
IMPLEMENTATION OF PROGRAM.....	17
1. Password Form	17
2. Main Form	18
3. Stock Data Entry	19
4. Amount Paid Or Not Form	21
5. Modify Item Type, Item Name, Price Form	21
6. Sales Data Entry.....	22
7. Unpaid Sales Amount Form	23
8. Unpaid Purchase Form.....	25
9. Purchase View Form.....	26
10. Sales View Form.....	27
11. Customer Form	28
12. Customer Detail Modify Form	29
13. New User Form.....	30
14. Backup Database Form	31
15. Restore Database Form	32
16. Available Stock Form	33
17. Critic Stock Level	33
18. Reports	34
CONCLUSION.....	36
REFERENCES.....	37
APPENDIX	38

HISTORY OF VISUAL BASIC

Microsoft and the IBM PC

In 1975, Microsoft launched its first product: a BASIC compiler for the MITS Altair, an early kit microcomputer.

When IBM launched its Personal Computer (PC), the software supplied included small ROM- and disk-based versions of BASIC. IBM's PC-DOS (written by Microsoft) included an expanded, disk-based version of BASIC called BASICA (advanced BASIC). Microsoft's MS-DOS for PC compatibles included a similar program called GWBASIC. The difference between BASICA and GWBASIC was that BASICA required the built-in ROM BASIC to be present.

Both BASICA and GWBASIC were interpreters that translate and execute one instruction at a time. Interpreters are easier to implement and require no memory for object code, but the code runs much slower than compiled programs.

QuickBASIC was a BASIC compiler launched around 1983 for commercial programmers who wanted to write larger programs in BASIC on PC's. Programs compiled with QuickBASIC ran four to ten times faster than under BASICA or GWBASIC. Microsoft claimed that, on an 8-MHZ IBM PC-AT, the QuickBASIC compiler could translate code at 150,000 lines per minutes (fast compared to many compilers for other languages). Furthermore, QuickBASIC was upwards compatible from the BASIC interpreters. QuickBASIC went through several upgrades, ending with version 4.5 released in 1988.

In 1987, IBM launched the PS/2 personal computers. Newer IBM and compatible PCs stopped including ROM BASIC with the hardware. Other factors, including the rapid development of applications software and increasingly sophisticated compiled languages, combined to make the original BASIC interpreters obsolete. Microsoft shipped a replacement, called QBASIC, with MS-DOS versions 5 (May 1991) and 6 (March 1993). QBASIC is a disk-based interpreter system that comes with MS-DOS and with Windows 95. QBASIC implements the same language as QuickBASIC, but does not include some of the advanced debugging commands. Internal memory management is also different.

A number of improvements distinguish QuickBASIC and QBASIC (together, QBs) from earlier BASIC interpreters. Source files are saved in ASCII format, whereas earlier BASIC systems stored compressed encoded source files. Both QBs include a full-screen, menu-driven editor. The newer languages allow a maximum program/data space of 160K, where the previous limit was 64K. New data types were added for increased computing power.

The Microsoft Windows Graphical Operating Environment

Graphical User Environments/Interfaces (GUIs) were demonstrated at the Xerox Corporation's Palo Alto Research Center (Xerox PARC) in 1975. Located in Silicon Valley, near one of the world's leading schools of computer science (Stanford University), and founded in 1970, Xerox PARC was responsible for many stellar innovations in computing and electronics. It is certain that neither Apple nor Microsoft had anything to do with the original conception of GUIs.

The Apple Computer company introduced two machines featuring GUIs in the 1980s. The first, named the Lisa (1983), was an evolutionary advance for Apple although not a commercial success. The second model was the Macintosh (1984), first in a product line that has continued to this date.

In 1985, four years after the introduction of the PC, Microsoft launched version 1 of its Windows interface. Early versions of Windows were add-ons that ran "on top of" the MS-DOS operating system. Versions 1 and 2 of Windows included a primitive user interface similar to the Windows Explorer. To run a program under these systems, one located the file and double-clicked it.

Windows 3.0, introduced in 1990, included the first predecessor of the "desktop" of today's Windows systems. An updated version, Windows 3.1, was launched in April 1992, and included some key technological advances, including the powerful TrueType font system licensed from Apple. This was the version that "caught fire" and began a revolution in PC-compatible software markets. Windows 95 was the first version that stood alone and did not require the DOS operating system to run. It was also the first version to run code in the 32-bit "native" mode of newer Intel processors such as the 486 and Pentium families. Windows 1, 2 and 3.x ran code in a slower 16-bit "compatibility" mode.

Visual Basic is Born

Alan Cooper is considered the father of Visual Basic. In 1987, the then Director of Applications Software for Coactive Computing Corporation wrote a program called Ruby that delivered visual programming to the average programmer/user.

The increasing popularity and sophistication of graphical user interfaces (GUIs) led Microsoft to introduce Visual Basic (not spelled with capitals) in 1991. Tom Button, Group Product Manager for Applications Programmability at Microsoft, headed the team that produced QuickBASIC and QBASIC. This same group developed Visual Basic by combining Ruby with QuickBASIC.

On June 15th 2001, a page on Microsoft's Web site entitled "Visual Basic 10th Birthday" included the following paragraph, entitled "Thunder": «Initially, Visual Basic 1.0 was intended to be a very tactical product. Microsoft had several initiatives in development leading up to Visual Basic 1.0, all of which were intended to develop into long-term, strategic, graphical, object-oriented programming tools. As is typical with version 1.0 products, however, the Visual Basic 1.0 product team was forced to cut features from its long list of ideas in order to actually deliver the product to market. As a result, the first Visual Basic offering included little more than the Embedded Basic technology that had originally shipped in Microsoft QuickBasic 4.0 (Microsoft's threaded p-code and incremental compiler) and a simple shell design tool originally licensed for but never used in Windows 3.0. Approximately 12 months after development on version 1.0 began, Microsoft released this "placeholder" development tool, code-named "Thunder."»

The Visual Basic (VB) system is a fourth generation programming system which produces much of the code itself as the programmer designs the interface for his or her application. Microsoft surveys in the late 1990's showed that roughly two-thirds of all business applications programming on PCs was being done in Visual Basic.

At one time Visual Basic could produce code for both DOS and Windows applications. Today, however, Microsoft considers DOS to be obsolete and promotes the Windows environment exclusively. QBASIC continued to ship on the Windows CD-ROM up to (at least) version 98SE and so, at the time of writing, may still be available.

When Visual Basic 1.0 was released, Bill Gates, Chairman and CEO of Microsoft, described it as 'awesome'. Steve Gibson in Infoworld said Visual Basic is a 'stunning new miracle' and would 'dramatically change the way people feel about and use [Microsoft] Windows.' Stewart Alsop was quoted in the New York Times as saying Visual Basic is 'the perfect programming environment for the 1990's'.

VB's success may be largely due to the simplification that it brought to Windows application programming. Prior to Visual Basic, Windows applications programming required mastery of huge subroutine libraries and hundred of lines of code to create even simple screen elements. VB eliminates the need to write code for GUI input/output, thus reducing by orders of magnitude the length of code and time to develop an application. Charles Petzold, author of many of the standard reference works on Windows programming in C, was quoted in the New York Times as saying "For those of us who make our living explaining the complexities of Windows programming to programmers, Visual Basic poses a real threat to our livelihood".

However, successful programming in this system requires an understanding of asynchronous event-driven multi-programming, networked, client-server and database architectures, and therefore it has been suggested that QBASIC and other third generation languages still better meet the design goals that Kurtz and Kemeny originally

set, i.e. to be easy to learn and rapidly useful for a wide range of simple programming problems.

The Evolution of Visual Basic

Visual Basic 1.0 for Windows was first released on May 20, 1991 at the Windows World convention in Atlanta Georgia. In September 1992, Microsoft announced Microsoft Visual Basic for MS-DOS in Standard and Professional editions. Like Visual Basic for Windows, this version combined the ease of graphical design with the power and versatility of traditional programming. Developers simply drew the user interface and attached code that responded to events. However, following the release of Windows 3.1 in March 1992 it became apparent that the DOS environment had come to the end of its useful life. The last version of MS-DOS, 6.22, was released in 1994.

VB version 2.0 for Windows (November 1992) was faster, more powerful and easier to use than version 1. VB 2 was also available in a freeware student release called the Primer edition. Visual Basic 3.0 (1993) added tools to access and control databases and Object Linking and Embedding (OLE) version 2. It came in Standard and Professional versions.

A superset of VB, called Visual Basic for Applications, was released as part of Microsoft Excel 5 and Microsoft Project 4 in 1993. It has since become the internal programming language of the Microsoft Office family of products, and is available for license by other software companies.

Visual Basic 4 was released in 1995 and supported the new Windows 95 family of 32-bit operating systems. The Professional Edition could also compile code to run on the older 16-bit Windows 3.x systems. Visual Basic Scripting Edition (VBScript) was also announced in 1995. VBScript is used to write embedded code for inclusion in web pages, although not all web browsers will run VBScript.

With the introduction of Visual Basic version 5 in early 1997, 16-bit systems were no longer supported. Between versions 4 and 5, significant changes were made in the user interface. Visual Basic 5 added, among other things, the ability to create true executables and to create your own custom controls. It also supported Microsoft's Active-X technology.

Visual Basic 5 was available in Standard (Learning), Professional and Enterprise Editions. A free edition, called Control Creation Edition, could be downloaded from www.microsoft.com, and was included with many textbooks. Visual Basic 5 was also included as part of a package known as Visual Studio 97.

Visual Basic 6 (VB6) was introduced in 1998 and was included as part of a package known as Visual Studio 6.0. VB6 added new capabilities in the areas of data access, Internet features, controls, component creation, language features and wizards. To quote Microsoft's web site, «Visual Basic 6.0 features provide graphical, integrated data access to any ODBC or OLE DB data source, and additional database-design tools for Oracle and Microsoft SQL Server™-based databases. New Web development features bring the easy-to-use, component-based programming model of Visual Basic to the creation of HTML- and Dynamic HTML (DHTML)-based applications.» Many organizations are still using this version today.

INTRODUCTION

Visual Basic is a Microsoft Windows programming Language. Visual Basic programs are created in an Integrated Development Environment (IDE) . The IDE allows the programmer to create , run and debug Visual Basic programs conveniently. IDEs allow a programmer to create working programs in a fraction of the time that it would normally take to code programs without using IDEs. The process of rapidly creating an application is typically referred to as Rapid Application Development(RAD). Visual Basic is the world's most widely used RAD language.

Visual Basic is derived from the BASIC programming language. Visual Basic is a distinctly different language providing powerful features such as graphical user interfaces, even handling, access to the Win32 API, object-oriented features, error handling, structured programming, and much more.

The Visual Basic IDE allows Windows programs to be created without the need for the programmer to be a Windows programming expert.

Microsoft provides several versions of Visual Basic, namely the Learning Edition , the Professional Edition and the Enterprise Edition. The Learning Edition provides fundamental programming capabilities than the Professional Edition and is the choice of many programmers to write Visual Basic applications. The Enterprise Edition is used for developing large-scale computing systems that meet the needs of substantial organizations.

Visual Basic is an interpreted language. However , the professional and Enterprise Edition allows Visual Basic code to be compiled to native code.

Visual Basic evolved from BASIC(Beginner's All purpose Symbolic Instruction Code). Basic was developed in the mid 1960's by Professors John Kemeny and Thomas Kurtz of Dartmouth College as a language for writing simple programs. BASIC's primary purpose was to help people learn how to program.

The widespread use of BASIC with various types of computers (sometimes called hardware platforms) led to many enhancements to the language. With the development of the Microsoft windows graphical user interface (GUI) in the late 1980s and the early 1990s, the natural evolution of BASIC was Visual Basic, which was created by Microsoft Corporation in 1991.

Until Visual Basic appeared, developing Microsoft Windows-based applications was a difficult and cumbersome process. Visual Basic greatly simplifies Windows application development. Since 1991 six versions have been released, with the latest-Visual Basic 6-appearing in september 1998.

After a brief explanation about the Visual Basic 6.0 and the developing layers, I hope that you will find the necessary information that you need all about the Visual Basic even if you are a text based programmer.

INTRODUCTION TO STOCK CONTROL

This project is about writing software of stock control and customer management in any firm by database (Microsoft Access) and Visual Basic 6.0 as an interface. The user with this program can learn system of circulation of stocks how it is managed. Also the user can control of customers how it is managed.

With using this project firm's staff can make registration of stocks and new customer , and circulation of stocks between registered customers. The staff of firm can learn that amount of stocks and also learn critic level of stocks. Moreover staff of firm can learn that debt of customer and also learn debt of firm.

This project allows the user to view the sales and purchase. And this project allows to take reports about sales, sales unpaid and available stock.

This project allows the user to back up the database and at the same time when database is damage, the user use back up file and thus program of database recoveries.

DATABASE STRUCTURE

1. Customer Database

With this database information of customer holds. There are customerid, customername, phone, mobil, fax, and address fields in this customer database. The database is shown below:

	Field Name	Data Type
?	customerid	Text
	customername	Text
	phone	Text
	mobil	Text
	fax	Text
	address	Text

2. Quantity of Item Database

With this database situation of available of stocks holds. There are itemtype, itemname, quantity, price, total fields in this database. The database fields is shown below:

	Field Name	Data Type
	itemtype	Text
	itemname	Text
	quantity	Number
	price	Currency
	total	Currency

3. Item Master Database

With this database information of stocks holds. There are itemno, itemtype, itemname, price fields in this database. The database fields is shown below:

	Field Name	Data Type
?	itemno	Text
	itemtype	Text
	itemname	Text
	price	Currency
	quantity	Text

4. Purchase Master Database

In this database information of purchase holds. There are invoiceno, itemtype, itemname, quantity, price, total, date, description fields in the database. The database fields are shown below:

	Field Name	Data Type
	invoiceno	Text
	itemtype	Text
	itemname	Text
	quantity	Number
	price	Currency
	total	Currency
	date	Date/Time
	description	Text

5. Sales Master Database

In this table information of sales holds. There are invoiceno, customername, customerid, date, itemtype, itemname, quantity, price, total in the table. The fields of table are shown below:

	Field Name	Data Type
	invoiceno	Text
	customername	Text
	customerid	Text
	date	Date/Time
	itemtype	Text
	itemname	Text
	quantity	Number
	price	Currency
	total	Currency

6. Pass User Database

In this table record of username and password holds. There are username and userpass fields in the table. The fields of table are shown below:

	Field Name	Data Type
	username	Text
	userpass	Text

7. Amount Unpaid Remind Database

In this table both debt of customer and debt of firm information holds. Type of transaction is 'SALES' for customer and type of transaction is 'PURCHASE' for firm.

In this table there are trans_type, date, amount_unpaid, invoiceno, and customername fields. The fields of table are shown below:

	Field Name	Data Type
	trans_type	Text
	date	Text
	amount_unpaid	Currency
	invoiceno	Text
	customername	Text

8. Item Type Database

In this table type of item holds. And this table related with item_master table. In this table there is itemtype field. The table is shown below:

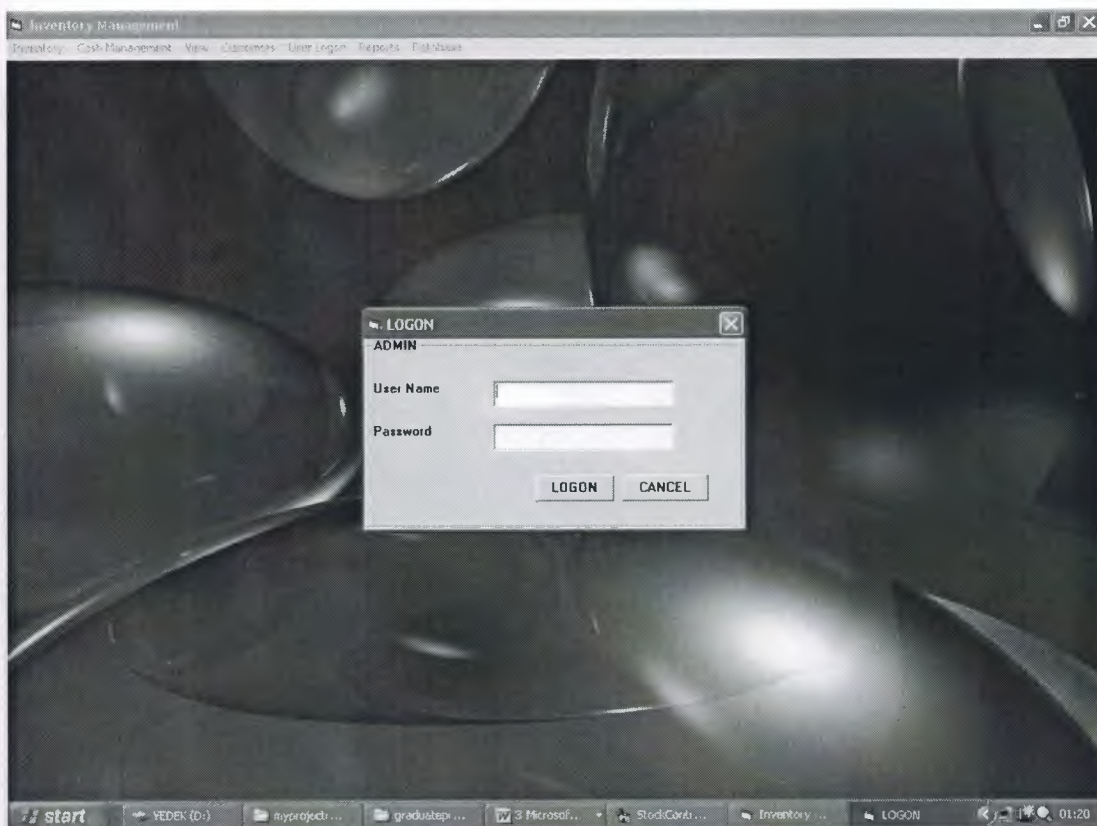
	Field Name	Data Type
	itemtype	Text

IMPLEMENTATION OF PROGRAM

I am going to explain the user interfaces part of my program to a user who does not know anything about the program.

1. Password Form

Every user must have a password for using to program. Firstly user must enter the correct username and correct password. If user enter invalid username or password “username or password incorrect” message displayed on the screen . If user enter invalid username or password three times then the program automatically shut down. If username and password valid then main form is enabled. If the user clicks “Cancel” button then program is shut down.



2. Main Form

Main form is appearing during program running and includes sub menus as stock entry , sales data entry, customer information, database backup and restore etc. on its top. We reaches sub forms by using the sub menus on the main form. The user reaches Stock Entry Data, Sales Data Entry, Available Stok, Critic Stock and Modify Item by using Inventory sub menu. The user reaches Sales Payment Received and Purchase Payment Given sub form by using Cash Management sub menu. The user reaches Sales Entries and Purchase Entries by using View sub menu. The user reaches Add New Customer and Customer Modify sub form by using Customers sub menu. The user reaches New User sub form by using User Logon sub menu. The user reaches reports sub forms by using Reports sub menu. The user reaches Back up Database and Restore Database sub form by using Database sub menu.



3. Stock Data Entry

This form allows the user to input items. This form using user can join the new items to stock or to available items can make to append. If the user make to output from stock firstly clicks “Add New” button and select item type and item name from comboboxes. Then the user enter quantity for choose item. Then the user clicks “Save” button. If the user make to output for another item again clicks “Add New” button. If the user want to delete item then selects the item from datagrid and then clicks “Delete” button or if the user want to modify item then selects the item from datagrid and then clicks “Modify” and then the user modify item. If the user clicks “Cancel” button then output is cancel. If the user clicks the “Save Entry” then finished output.

Inventory Management

Stock Entry

Invoice No: A0015 Entry Date: 20.05.2004

Item Type: CDROM New Item Type

Item Name: SAMSUNG 52X New Item

Total Qty: 3 Price Per Unit: 15 Total Amount: 45

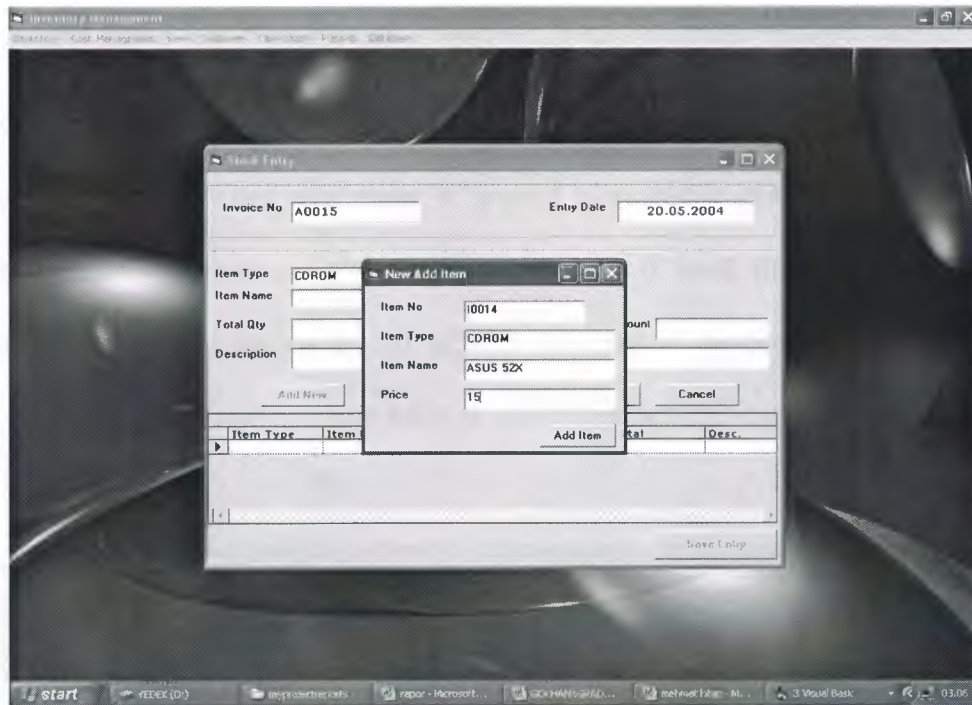
Description:

Add New Save Delete Modify Cancel

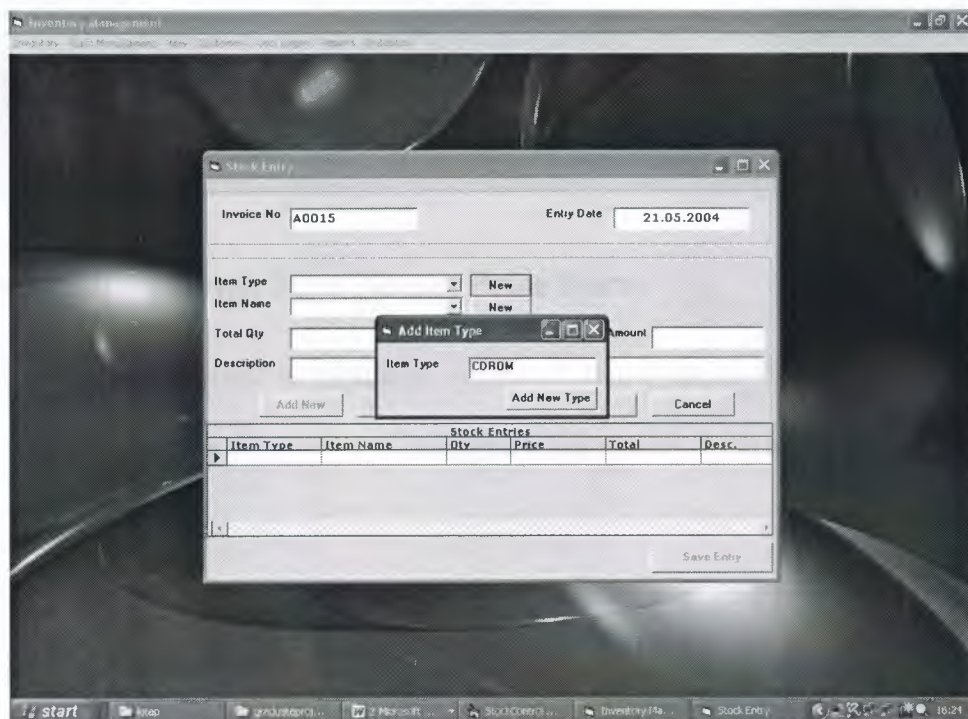
Item Type	Item Name	Qty	Price	Total	Desc.
CDROM	SAMSUNG 52X	3	15	45	

Save Entry

If the user can define new item firstly clicks “Add New” button then selects item type from combobox then clicks “New Item” button. And the New Add Item form shows and the user enter the item name and price of item then clicks “Add Item” button. Then this form unload.



If the user want to enter new item but this type of item is not available in the stock then the user clicks “New Item Type” button. Then the Add Item Type form shows. Then the user enter item type and clicks “Add New Type” button. Then this form is unload. And then the user clicks “New Item” button and enter information of item.



4. Amount Paid Or Not Form

This form uses both of sales and purchase. To this form reaches after user append the new item to stock or when user make to append to available item or when sales item. This form allows the to record either sales or purchase if paid. Either sales or purchase if does not paid then records the all debt to database. If payment will not complete during either sales or purchase then balance of debt records the database.

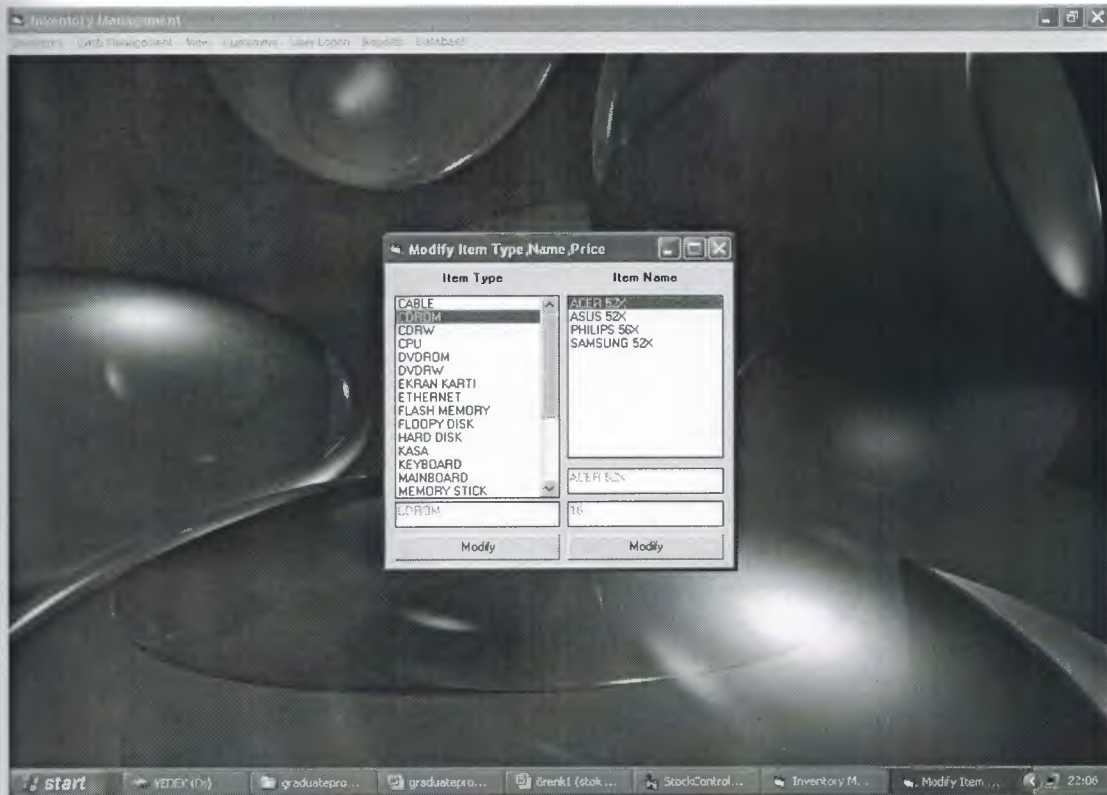
The screenshot shows a Windows XP desktop environment with a taskbar at the bottom. The taskbar includes the Start button and several open applications: Windows Explorer, a folder named 'SCHOOLING', 'graduateproject', 'StockControl - Mr...', and 'Inventory Manage...'. The system clock in the bottom right corner shows the date '21.05.2004' and time '16:29'. A dialog box titled 'Amount Paid Or Not?' is centered on the screen. The dialog box has a title bar with standard Windows window controls. Inside, it displays 'Invoice No A0015' and 'Transaction Purchase'. Below this, a question mark icon is followed by the text 'Are the Amount of this Transaction Paid or not ?, if paid then how much amount paid...'. To the right of this text is a date field containing '21.05.2004'. Underneath, there is a section titled 'How much Paid ?' with a radio button selected for 'Amount Paid' and a text input field containing the number '4'. Below this, there is a radio button for 'Amount Not Paid' and a text input field containing '48'. At the bottom right of the dialog box, there is a checkbox labeled 'Transaction Finished' which is currently unchecked. An 'OK' button is located at the bottom center of the dialog box.

5. Modify Item Type, Item Name, Price Form

This form allows the user to modifies types of items, names of items, and prices of items. For example if a price of item change we can modify by using this form.

If the user want to modify name of item type then selects item type from list of item type. Then the user clicks “Modify” button and enable item type textbox and the user modify name of item type.

If the user want to modify name of item name or price then before selects item type from list of item type after selects item name then clicks “Modify” button and enables item name textbox and price textbox and the user modify name of item or price.



6. Sales Data Entry

This form allows the user to output the item. If the user want to make output from stock firstly clicks “Add New” button. Then the user selects customer name from combobox of customer name. Then user selects items from combobox of item type and item name. According to items available quantity of item and price of item changes. Then the user enter the order quantity of item. Then user clicks “Save” button. The user again same process for other item to add. If the user want to delete item firstly selects into item from datagrid and clicks “Delete” button. If the user want to modify any item firstly

selects into item from datagrid then clicks “Modify” button and user can modify the item. If the user clicks “Cancel” button , process of output is cancel. The user want to complete process of output then clicks “Complete to Sale” button.

The screenshot shows the 'Sales Entry' window with the following data:

Invoice No	Sales Date	Customer Name	Customer No	Item Type	Item Name	Available Qty	Order Qty	Price Per Unit	Total
S000000010	21.05.2004	Abdurrahman Alsayyaf	C000000003	CDRW	SONY 24X12X12	1	1	35	35

The table at the bottom of the window shows the current sales entry:

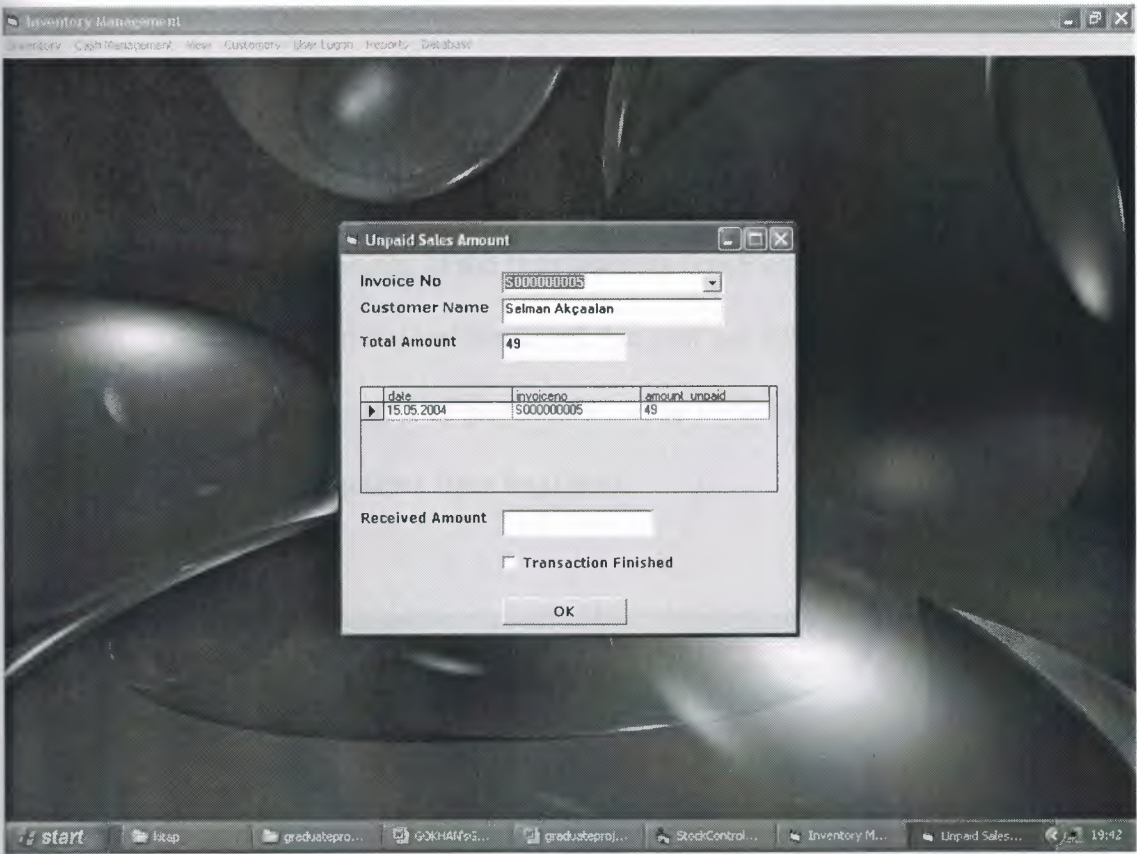
Item Type	Item Name	Qty	Price	Total	Desc
CDRW	SONY 24X12X12	1	35	35	

7. Unpaid Sales Amount Form

This sub menu allows the user to see the customer’s debt and accept paid. If the customer will make complete payment the “Transaction Finished” signs. Thus debt of customer completed. If the customer will not make complete payment then amount of pay subtraction from total debt.

The user selects invoice no from combobox. According to selecting invoice no , customer name and amount unpaid of customer changes. If the customer want to finish

all of debt then the user sign “Transaction Finished” and field of Received Amount not fill. If the customer will not make complete payment then amount of payment is write to field of Received Amount and amount of payment substractions from total debt.



8. Unpaid Purchase Form

This sub menu allows the user to see the firm's debt and accept paid. If the firm will make complete payment the "Transaction Finished" signs. Thus debt of firm completed. If the firm will not make complete payment then amount of pay subtraction from total debt.

The user selects invoice no from combobox. According to selecting invoice no , amount unpaid of firm changes. If the firm want to finish all of debt then the user sign "Transaction Finished" and field of Received Amount not fill. If the firm will not make complete payment then amount of payment is write to field of Received Amount and amount of payment substractions from total debt.

The screenshot shows a Windows-style application window titled "Inventory Management" with a menu bar containing "Inventory", "Cash Management", "View", "Customers", "User Login", "Help", and "Database". A modal dialog box titled "Unpaid Purchase Amount" is open in the center. It contains the following elements:

- Invoice No:** A dropdown menu showing "A0003".
- Total Amount:** A text box containing "26".
- Table:** A table with two columns, "Date" and "Unpaid". It contains one row with the date "17.05.2004" and the value "26".
- Paid Amount:** An empty text box.
- Transaction Finished:** A checkbox that is currently unchecked.
- OK:** A button at the bottom of the dialog.

The taskbar at the bottom of the screen shows the "start" button and several open applications: "gradustapro...", "VEDEK (C:)", "gradustapro...", "örank1 (stok...", "StockControl...", "Inventory M...", "Unpaid Purch...", and a clock showing "20:20".

9. Purchase View Form

This sub menu allows the user to view the purchase for stock. For this user selects to invoice number from combobox and user see detail of purchase.



10. Sales View Form

This form allows the user to view the sales from stock. For this user uses the comboboxes and user want to see which sales , user selects from comboboxes. At the same time user can get the report for each sales.

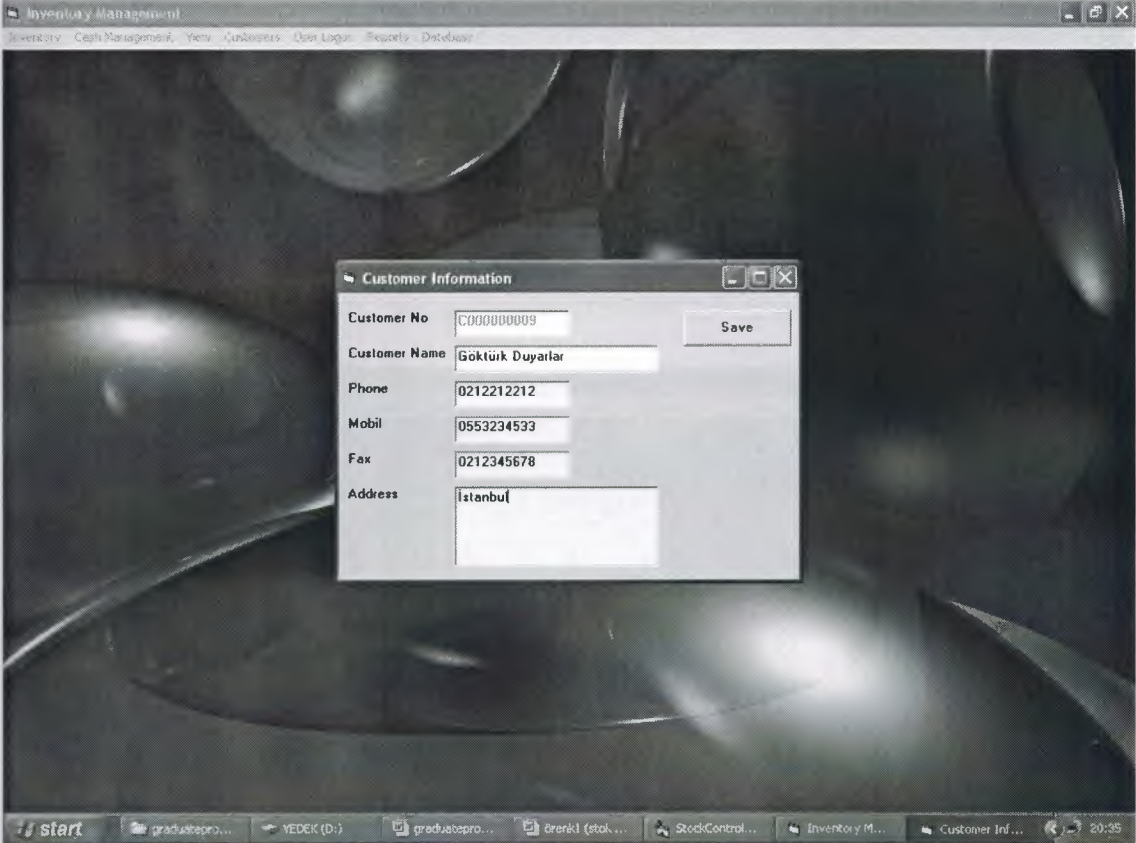
The screenshot shows a Windows application window titled 'Inventory Management'. Inside, a dialog box titled 'View Sales Entry...' is open. The dialog has three input fields: 'Customer Name' with a dropdown menu showing 'Selman Akpaalan', 'Invoice No' with a dropdown menu showing 'S000000005', and 'Sales Date' with a text box showing '15.05.2004'. Below these fields is a table with the following data:

Invoice No	Customer Id	Sales Date	Item Type	Item Name	Qty	Price
S000000005	C000000001	15.05.2004	CDRW	SONY 24X12X12	1	21
S000000005	C000000001	15.05.2004	DVDROM	SONY 16X	2	18
S000000005	C000000001	15.05.2004	CDROM	SAMSUNG 52X	2	13

At the bottom right of the dialog is a 'Report' button. The Windows taskbar at the bottom shows several open applications, including 'gradustepro...', 'VEDEK (D:)', and 'Inventory M...'. The system clock shows 20:30.

11. Customer Form

This sub menu allows the user to the form which the details of the new customer is entered. On this form the customer number is given automatically. The name can also be entered, apart from these the phone, mobil, fax and address of the customer is saved.



The screenshot shows a Windows application window titled 'Inventory Management'. Inside, there is a 'Customer Information' dialog box. The dialog box has the following fields and values:

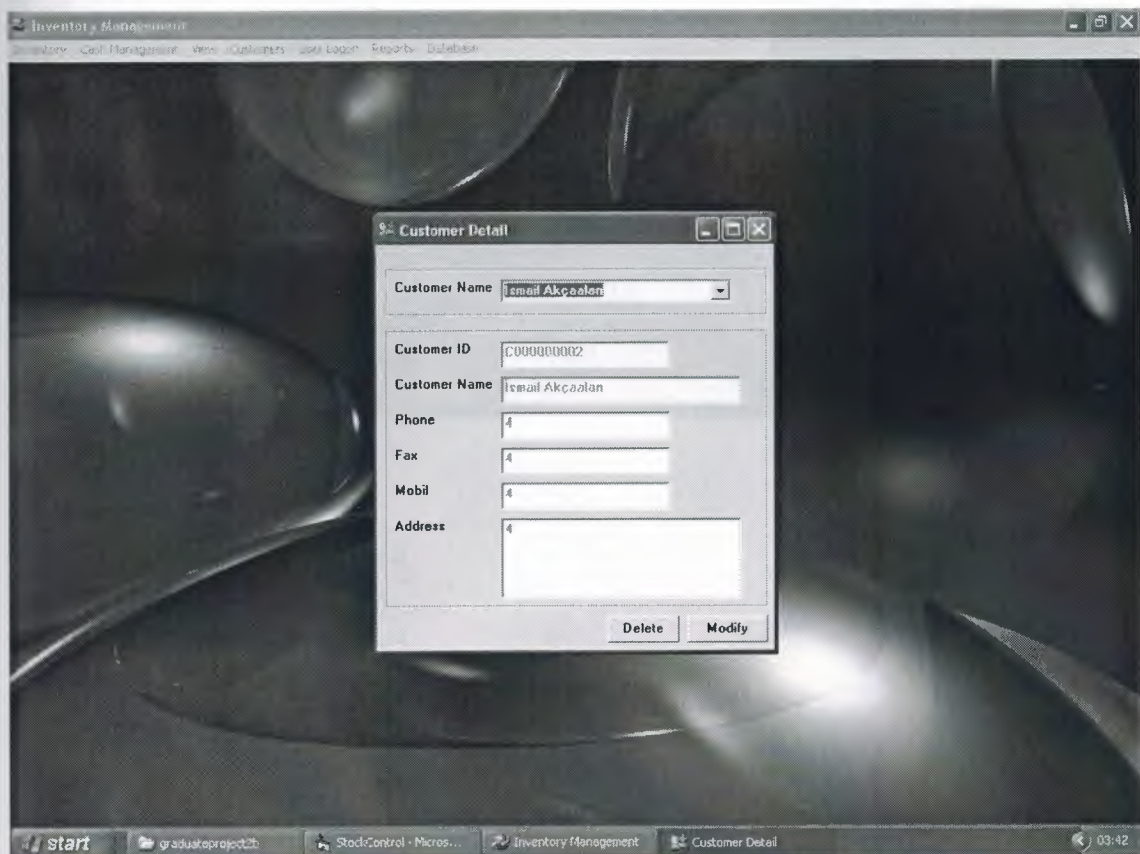
Field	Value
Customer No	000000009
Customer Name	Gökürk Duyarlar
Phone	0212212212
Mobil	0553234533
Fax	0212345678
Address	İstanbul

A 'Save' button is located to the right of the 'Customer No' field. The taskbar at the bottom shows several open applications, including 'gradueapro...', 'YEDEK (D:)', 'örnek1 (etok...', 'StockControl...', 'Inventory M...', and 'Customer Inf ...'. The system clock shows 20:35.

12. Customer Detail Modify Form

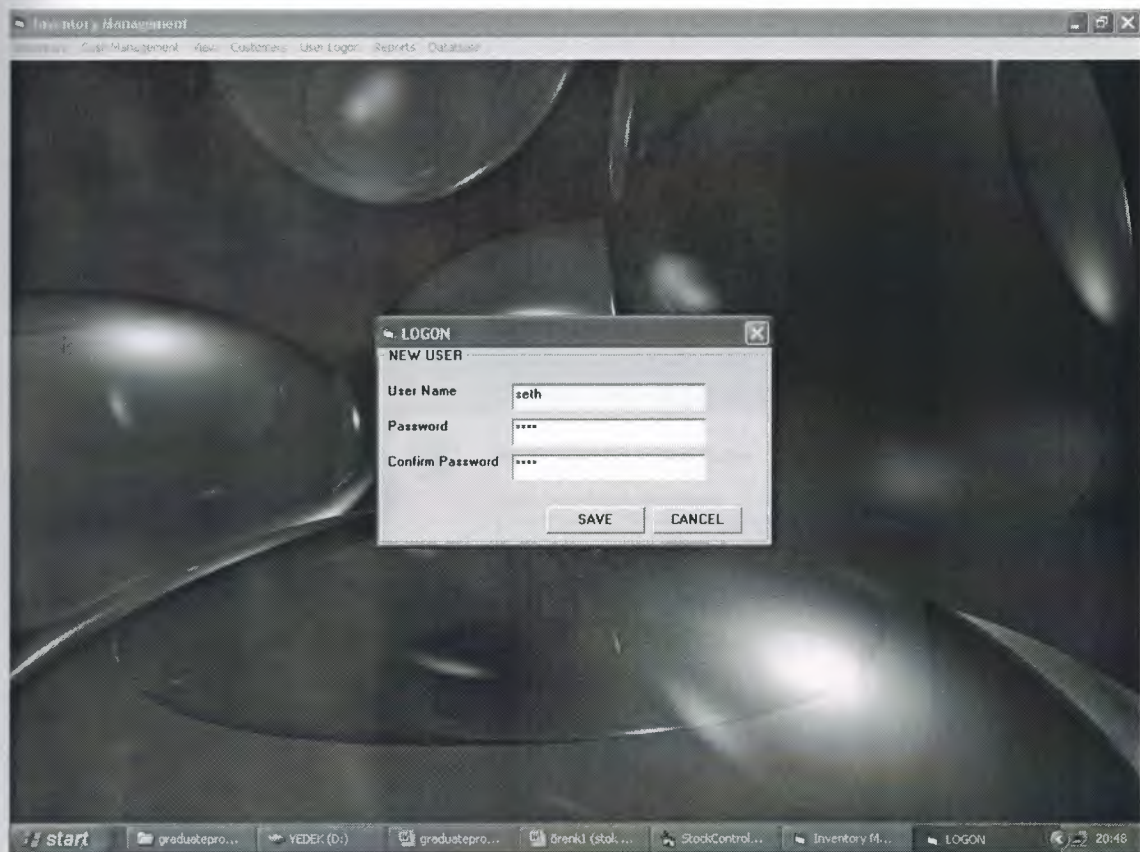
This sub menu allows the user to modify which details of the customers. Namely, the user can make to modify the information of customers. At the same time the user to delete customer. If customer is indebted then user can not delete to customer. If customer is indebted to “Delete” button unable on the form.

If the user want to modify any customer details then firstly selects customer name from combobox. Then user clicks “Modify” button and all of textbox is enable then user can modify to customer details. Then user clicks “Save” button. If the user want to delete a customer , user selects customer name from combobox. If customer is indebted , “Delete” button is unable or not. Then user clicks “Delete” button and deletes customer.



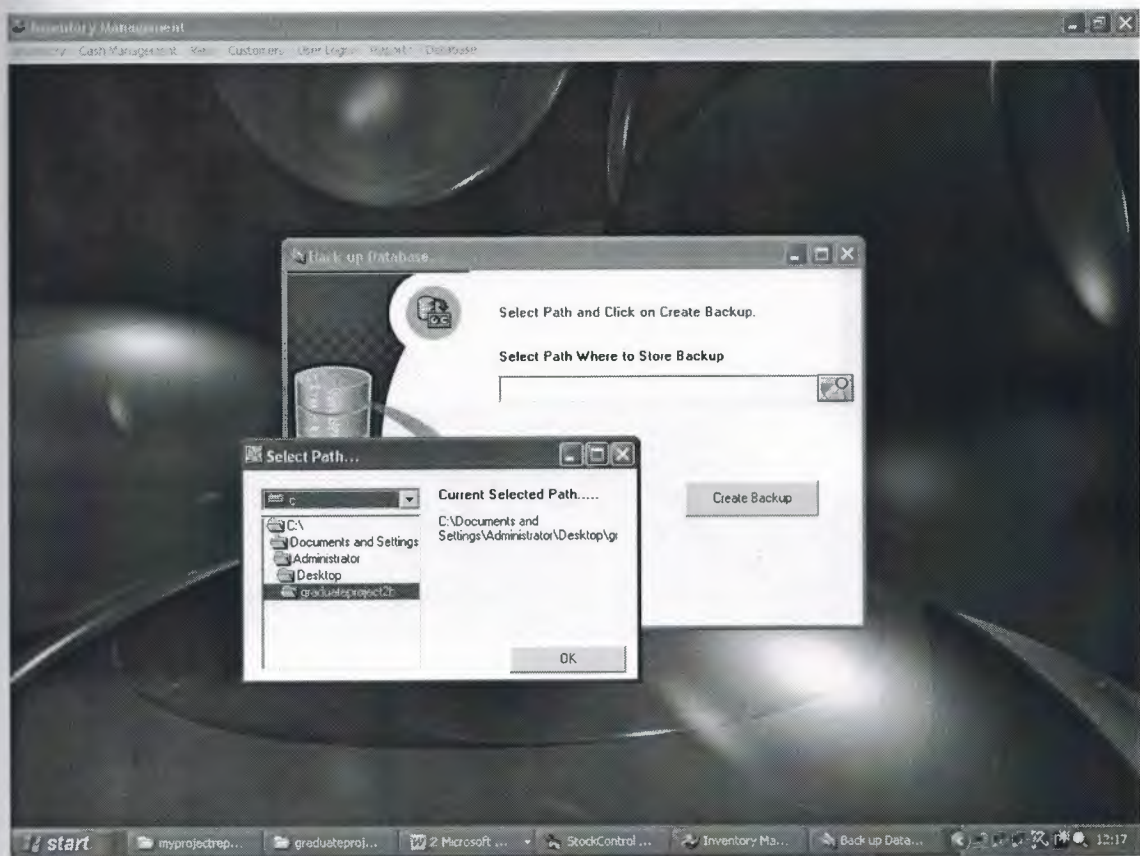
13. New User Form

This form allows the user to define new user. For this user enter username and password and confirm password. Both password and confirm password is true then the program is restart. If they are not true then appears error message “Wrong Confirm Password”.



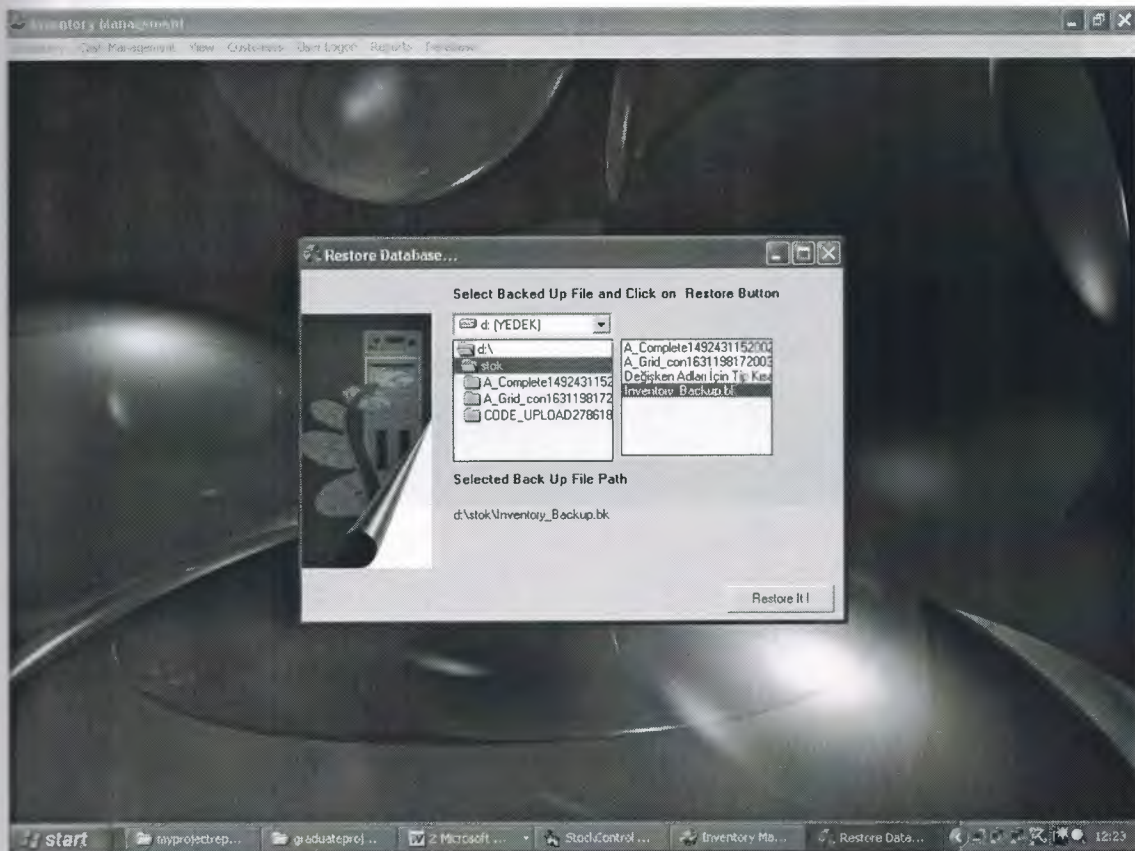
14. Backup Database Form

This form allows the user to backup the database of program. The user must backup to database everyday. For this user firstly clicks backup button and appears Select Path form. Then user select path for backup. Then user clicks “OK” button and Selects Path form unloads. Then selecting path appers textbox of Select Path Where to Store Backup. Then user clicks “Create Backup” buttons and process of backup finishes.



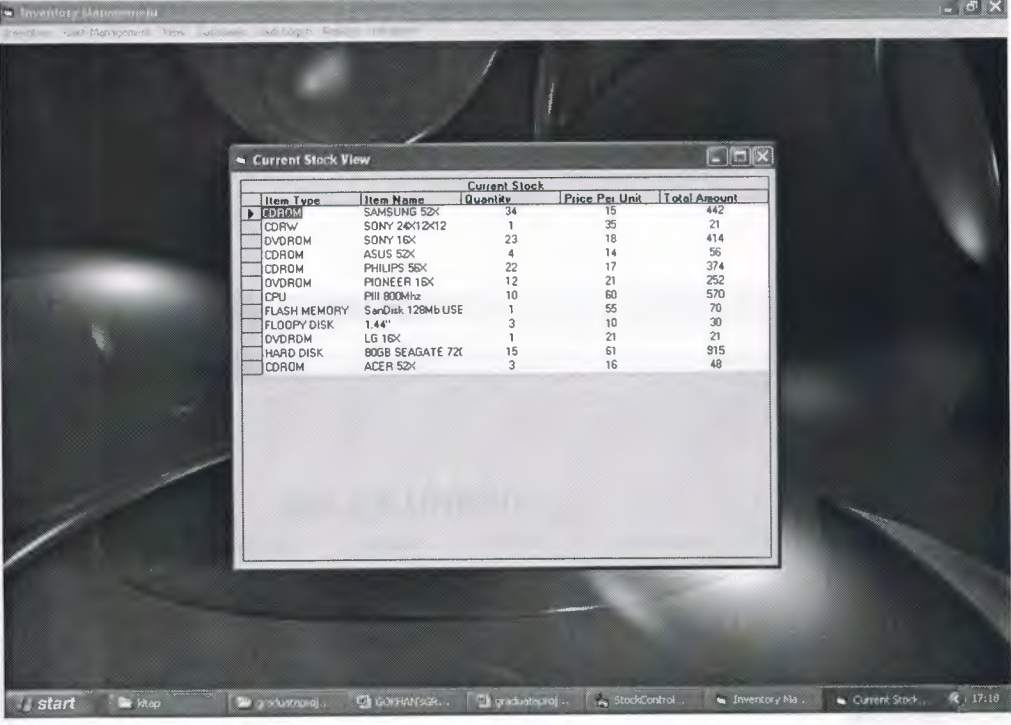
15. Restore Database Form

This form allows the user to restore the database. If database of program is damage the the user selects where back up file ("Inventory_Backup.bk"). Then user clicks "Restore It!" button. And process of restore database finishes.



16. Available Stock Form

This form allows the user to see situation of stock.



Item Type	Item Name	Quantity	Price Per Unit	Total Amount
CDROM	SAMSUNG 52X	34	15	442
CDRW	SONY 24X12X12	1	35	21
DVDROM	SONY 16X	23	18	414
CDROM	ASUS 52X	4	14	56
CDROM	PHILIPS 56X	22	17	374
DVDROM	PIONEER 16X	12	21	252
CPU	PIII 800Mhz	10	60	570
FLASH MEMORY	SanDisk 128Mb USE	1	55	70
FLOPPY DISK	1.44"	3	10	30
DVDROM	LG 16X	1	21	21
HARD DISK	80GB SEAGATE 72K	15	61	915
CDROM	ACER 52X	3	16	48

17. Critic Stock Level

This form allows the user to see situation of critic stock.

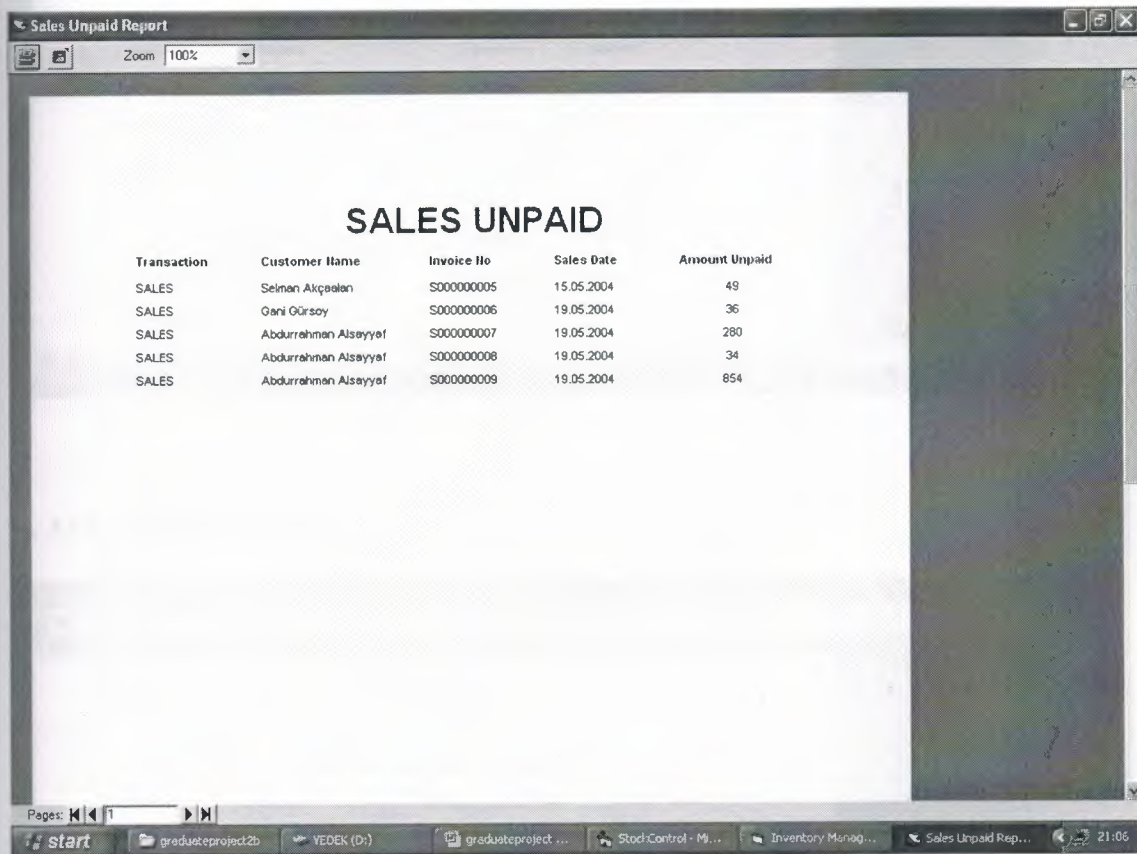


Itemtype	Itemname	quantity
CDRW	SONY 24X12X12	1
CDROM	ASUS 52X	4
FLASH MEMORY	SanDisk 128Mb USB2.0	1
FLOPPY DISK	1.44"	3
DVDROM	LG 16X	1
CDROM	ACER 52X	3

18. Reports

The user can take the reports about sales unpaid, sales and available stock. The user can reach to this sub menu from Main Form by using Reports sub menu.

a. Sales Unpaid Report



Transaction	Customer Name	Invoice No	Sales Date	Amount Unpaid
SALES	Selman Akçeslen	S000000005	15.05.2004	49
SALES	Gani Gürsoy	S000000006	19.05.2004	36
SALES	Abdurrahman Alsayyaf	S000000007	19.05.2004	280
SALES	Abdurrahman Alsayyaf	S000000008	19.05.2004	34
SALES	Abdurrahman Alsayyaf	S000000009	19.05.2004	854

b. Sales Report

Sales Report

Zoom 100%

SALES REPORT

Invoice No	S000000001	Item Name	SAMSUNG 52X
Customer ID	C000000003	Item Type	CDROM
Customer Name	Abdurrahman	Sales Price	13
Sales Date	11.05.2004	Quantity	3
		Total	39

Invoice No	S000000002	Item Name	SONY 16X
Customer ID	C000000003	Item Type	DVDROM
Customer Name	Abdurrahman	Sales Price	18
Sales Date	11.05.2004	Quantity	2
		Total	36

Invoice No	S000000003	Item Name	SONY 16X
Customer ID	C000000005	Item Type	DVDROM
Customer Name	Fatih Yavuz	Sales Price	18

Pages: 1

start graduateproject2b YEDEI (D:) graduateproject ... StockControl - M... Inventory Manag... Sales Report 21:07

c. Available Stock Report

Available Stock Report

Zoom 100%

AVAILABLE STOCK

Item Type	Item Name	Quantity	Price	Total Amount
CDROM	SAMSUNG 52X	34	15	442
CDRW	SONY 24X12X12	1	35	21
DVDROM	SONY 16X	23	18	414
CDROM	ASUS 52X	4	14	56
CDROM	PHILIPS 56X	22	17	374
DVDROM	PIONEER 16X	12	21	252
CPU	PII 800Mhz	10	60	570
FLASH MEMORY	SanDisk 128Mb USB2.0	1	55	70
FLOPPY DISK	1 44"	3	10	30
DVDROM	LG 16X	1	21	21
HARD DISK	80GB SEAGATE 7200 RPM	15	61	915
CDROM	ACER 52X	3	16	48

Pages: 1

start graduateproject2b YEDEI (D:) graduateproject ... StockControl - M... Inventory Manag... Available Stock R... 21:07

CONCLUSION

Nowadays, windows oriented programs became more popular and flexible. Visual Basic 6.0 is one of the best well-known programming language based on window's environment. That's why I prefer this project. Now I can understand why these programming languages are very popular. Even I do not have experience with Visual Basic, this project did not become difficult to me. Visual Basic 6.0 has lots of help than other programming languages.

In my project, I have used important components of Visual Basic 6.0. Therefore I learned these components very well. Now I can use these components of Visual Basic 6.0 in an efficient manner. Also I have learned how to use new data access logic, which is ActiveX Data Objects (ADO). Additionally, I have used a database in my project. So I have gained many practices, experiences and knowledge of database. As known, database is very important topic for software programmers.

Finally, most important thing is for me that I have learned how to prepare an individual software project by using Visual Basic 6.0 to real life problems. After I have started my projects, I saw that you could face with unexpected real life problems. These real life problems are very different from the courses problem. This project became a good exercise to me for the real life and I used the things in my project that I learned from courses as theoretically.

REFERENCES

- Memik Yanık, **Visual Basic 6.0**, Beta Yayınevi
- İhsan Karagülle, Zeydin Pala, **Visual Basic 6.0 Pro**, Türkmen Kitabevi
- İhsan Karagülle, Zeydin Pala, **Access 2002**, Türkmen Kitabevi

A P P E N D I X

1. Codes of Password Form

```
Dim rs_user As New Recordset
```

```
Dim rs_pass As New Recordset
```

```
Private Sub Command1_Click()
```

```
Static i As Integer
```

```
If fSQLÇalıştır(gnBağlantı, rs_pass, "select * from pass_user where username=" +  
Text1.Text + " and userpass=" + Text2.Text + "") Then
```

```
If rs_pass.RecordCount > 0 Then
```

```
frmMainform.Enabled = True
```

```
frmUserLogon.Visible = False
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
Else
```

```
MsgBox "USER NAME OR PASSWORD WRONG", vbInformation, "ERROR"
```

```
i = i + 1
```

```
If i = 3 Then
```

```
MsgBox "Program will be shut down", vbInformation, "Shut Down"
```

```
End
```

```
End If
```

```
End If
```

```
End If
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
Frame2.Visible = True
```

```
End Sub
```

```
Private Sub Command3_Click()
```

```
End
```

```
End Sub
```



```

Private Sub Command4_Click()
If Text3 <> "" And Text4 <> "" Then
If Text4 = Text5 Then
If fSQLÇalıştır(gnBağlantı, rs_user, "select * from pass_user") Then
With rs_user
.AddNew
.Fields("username") = Text3.Text
.Fields("userpass") = Text5.Text
.Update
End With
Else
MsgBox "Wrong Confirm Password", vbCritical, "Wrong Confirm Password"
End If
End If
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
End If
Frame2.Visible = False
End Sub

```

```

Private Sub Form_Load()
frmMainform.Show
frmMainform.Enabled = False
Frame2.Visible = False
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
End Sub

```

2. Codes of Main Form

```
Dim Itemset As Recordset
```

```
Private Sub d_Click()
```

```
frmAddItem.Show
```

```
End Sub
```

```
Private Sub avastock_Click()
```

```
frmCurStock.Show
```

```
End Sub
```

```
Private Sub backup_Click()
```

```
frmBackup.Show
```

```
End Sub
```

```
Private Sub criticstock_Click()
```

```
frmCriticStock.Show
```

```
End Sub
```

```
Private Sub custinf_Click()
```

```
frmCustomer.Show
```

```
End Sub
```

```
Private Sub detail_Click()
```

```
frmCustomerControl.Show
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
VeriTabanıAç
```

```
frmUserLogon.Show
```

```
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
```

```
Unload frmUserLogon
```

```
If fVeriTabanıKapat(gnBağlantı, grKayıt) Then
```

End If

End Sub

Private Sub newuser_Click()

frmUserLogon.Show

frmUserLogon.Frame2.Visible = True

End Sub

Private Sub purcent_Click()

frmViewPurchase.Show

End Sub

Private Sub purchasepay_Click()

On Error Resume Next

frmUnpaidPurchase.Show

Exit Sub

End Sub

Private Sub restore_Click()

frmRestore.Show

End Sub

Private Sub salesdata_Click()

frmSalesEntry.Show

End Sub

Private Sub salesent_Click()

On Error Resume Next

frmViewSales.Show

Exit Sub

End Sub

Private Sub salespay_Click()

On Error Resume Next

frmUnpaidSales.Show

Exit Sub

End Sub

Private Sub salesreport_Click()

On Error Resume Next

DataEnvironment1.rsCommand1.Close

DataEnvironment1.rsCommand1.Open "select * from sales_master"

DataReport1.Refresh

DataReport1.Show

End Sub

Private Sub stockdata_Click()

frmStockEntry.Show

End Sub

Private Sub stockreport_Click()

On Error Resume Next

DataEnvironment1.rsCommand3.Close

DataEnvironment1.rsCommand3.Open "select * from quantity_of_item"

DataReport3.Refresh

DataReport3.Show

End Sub

Private Sub unpaidsalesreport_Click()

On Error Resume Next

DataEnvironment1.rsCommand2.Close

DataEnvironment1.rsCommand2.Open "select * from amount_unpaid_remind where
trans_type='SALES'"

DataReport2.Refresh

DataReport2.Show

End Sub

Private Sub upitem_Click()

frmUpdateItem.Show

End Sub

3. Codes of Stock Data Entry Form

New Add Item Form Sourcecode;

```
Dim additem As Recordset
Dim CheckData As Recordset
Dim rs As Recordset
Private Sub cmdAddItem_Click()
If Len(Text2.Text) > 0 Then
If fSQLÇalıştır(gnBağlantı, CheckData, "select * from item_master where itemname='"
& Text3.Text & "'") Then
If CheckData.EOF <> True Then
MsgBox "Item Already Exist ...", vbCritical, "Item Exist ..."
Text3.SetFocus
Exit Sub
End If
With additem
.AddNew
.Fields("itemno") = Text1.Text
.Fields("itemtype") = Text2.Text
.Fields("itemname") = Text3.Text
.Fields("price") = Text4.Text
On Error GoTo A1:
.Update
End With
frmStockEntry.Refresh_combobox (2)
frmStockEntry.Combo2.Text = Text3.Text
Unload Me
End If
Exit Sub
A1:
MsgBox "Wrong Item Type ...", vbCritical, "Wrong Item..."
additem.CancelUpdate
Text2.Text = Clear
Text2.SetFocus
```

End If

End Sub

Private Sub Form_Load()

Text2.Text = frmStockEntry.Combol.Text

If fSQLÇalıştır(gnBağlantı, additem, "select * from item_master") Then

Text3.Text = ""

Text4.Text = ""

If additem.EOF <> True Then

additem.MoveLast

LAST_ID= Mid(additem.Fields("itemno").Value, 2, Len(additem.Fields("itemno")))

Dim ID As Integer

ID = Val(LAST_ID)

ID = ID + 1

LAST_ID = ID

If Len(LAST_ID) = 1 Then

LAST_ID = "I000" & LAST_ID

ElseIf Len(LAST_ID) = 2 Then

LAST_ID = "I00" & LAST_ID

ElseIf Len(LAST_ID) = 3 Then

LAST_ID = "I0" & LAST_ID

ElseIf Len(LAST_ID) = 4 Then

LAST_ID = "I" & LAST_ID

End If

Text1.Text = LAST_ID

Else

Text1.Text = "I0001"

End If

End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

additem.Close

End Sub

Add Item Type Form Sourcecode

```
Dim addtype As Recordset
Private Sub Command1_Click()
If Len(Text1.Text) <> 0 Then
addtype.AddNew
addtype.Fields("itemtype") = Text1.Text
On Error GoTo A1:
addtype.Update
frmStockEntry.Combol.Text = Text1.Text
Unload Me
Else
MsgBox "Enter Item type ...", vbInformation, "You can not save Zero length Item name
..."
End If
Exit Sub
A1:
MsgBox "Duplicate Item name Found ..." & vbCrLf & "Enter Another name of Close
this form ...", vbCritical, "Duplicate Entry Found ..."
addtype.CancelUpdate
End Sub
Private Sub Form_Load()
If fSQLÇalıştır(gnBağlantı, addtype, "select * from item_type") Then
Text1.Text = ""
End If
End Sub
Private Sub Form_Unload(Cancel As Integer)
addtype.Close
End Sub
```

Stock Data Entry Form Sourcecode

```
Dim StockSet As Recordset
Dim Itemset As Recordset
Dim InvoiceSet As Recordset
```

```

Dim RecordCountSet As Recordset
Dim CurStockSet As Recordset
Dim AvaStockSet As Recordset
Dim rs_del As Recordset
Public TotalTransactionAmount As Double
Dim LAST_ID As String
Dim rs_cur_stock As Recordset

Private Sub cmdEntrySave_Click()
Dim t As Integer
t = MsgBox("Are you sure you want to save purchase bill", vbQuestion Or vbYesNo,
"Want to save Purchase bill")
If t = 7 Then
Exit Sub
End If
If fSQLÇalıştır(gnBağlantı, AvaStockSet, "select * from available_pur_stock") Then
CurStockSet.Requery
RecordCountSet.Requery
If RecordCountSet.Fields(0) > 0 Then
TotalTransactionAmount = TotalAmount("PURCHASE")
While CurStockSet.EOF <> True
grKayıt.AddNew
grKayıt.Fields("invoiceno") = Text1.Text
grKayıt.Fields("date") = Text2.Text
grKayıt.Fields("itemtype") = CurStockSet.Fields("itemtype")
grKayıt.Fields("itemname") = CurStockSet.Fields("itemname")
grKayıt.Fields("quantity") = CurStockSet.Fields("quantity")
grKayıt.Fields("price") = CurStockSet.Fields("price")
grKayıt.Fields("total") = CurStockSet.Fields("total")
If Len(CurStockSet.Fields("description")) > 0 Then
grKayıt.Fields("description") = CurStockSet.Fields("description")
End If
On Error GoTo B1
grKayıt.Update

```

```

GoTo A1:
B1:
MsgBox "Duplicate Entry Found...", vbCritical, "Duplicate Entry"
grKayıt.CancelUpdate
Exit Sub
A1:
rs_cur_stock.Close
If fSQLÇalıştır(gnBağlantı, rs_cur_stock, "select itemtype,itemname from
quantity_of_item where itemtype='" & CurStockSet.Fields("itemtype") & "' and
itemname='" & CurStockSet.Fields("itemname") & "'"") Then
If rs_cur_stock.EOF = False And rs_cur_stock.BOF = False Then
' BU ITEM VARMI YOKMU KONTROL EDİLİYOR
Else
With rs_cur_stock
.AddNew
.Fields("itemtype") = CurStockSet.Fields("itemtype")
.Fields("itemname") = CurStockSet.Fields("itemname")
.Update
End With
End If
End If
rs_cur_stock.Close
If fSQLÇalıştır(gnBağlantı, rs_cur_stock, "select * from quantity_of_item where
itemtype='" & CurStockSet.Fields("itemtype") & "' and itemname='" &
CurStockSet.Fields("itemname") & "'"") Then
rs_cur_stock.Fields("quantity") = Val(rs_cur_stock.Fields("quantity")) +
Val(CurStockSet.Fields("quantity"))
rs_cur_stock.Fields("price") = CurStockSet.Fields("price")
rs_cur_stock.Fields("total") = Val(rs_cur_stock.Fields("price")) *
Val(rs_cur_stock.Fields("quantity"))
rs_cur_stock.Update
CurStockSet.MoveNext
End If
Wend

```



```

CurStockSet.MoveFirst
While CurStockSet.EOF <> True
CurStockSet.Delete
CurStockSet.MoveNext
Wend

frmPaidNotPaid.Label3.Caption = Text1.Text
frmPaidNotPaid.Label4.Caption = "Purchase"
frmPaidNotPaid.Label8.Caption = TotalTransactionAmount
frmPaidNotPaid.Label10.Visible = False
frmPaidNotPaid.Label11.Visible = False
Unload Me
frmPaidNotPaid.Show vbModal
End If
AvaStockSet.Close
End If
End Sub

Private Sub Combo1_Click()
Refresh_combobox (2)
End Sub

Private Sub Combo2_Click()
StockSet.Close
If fSQLÇalıştır(gnBağlantı, StockSet, "select * from item_master where itemtype='" &
Combo1.Text & "' and itemname='" & Combo2.Text & "'") Then

If StockSet.EOF <> True Then
Text4.Text = StockSet.Fields("price")
StockSet.MoveNext
End If
End If
End Sub

Private Sub Command1_Click(Index As Integer)
If Index = 0 Then

```

```

Call ButtonDurumları(False)
enable_disable (True)
CurStockSet.AddNew
Clear_Box
cmdEntrySave.Enabled = False
ElseIf Index = 1 Then
With CurStockSet
.Fields("itemtype") = Combo1.Text
.Fields("itemname") = Combo2.Text
.Fields("quantity") = Text3.Text
.Fields("price") = Text4.Text
.Fields("total") = Text5.Text
.Fields("description") = Text6.Text
.Update
.UpdateBatch
End With
Set DataGrid1.DataSource = Nothing
CurStockSet.Requery
Set DataGrid1.DataSource = CurStockSet
enable_disable (False)
ButtonDurumları (True)
cmdEntrySave.Enabled = True
ElseIf Index = 2 Then
RecordCountSet.Requery
If RecordCountSet.Fields(0) > 0 Then
CurStockSet.Delete
CurStockSet.MoveNext
If CurStockSet.EOF <> True Then
Call FillText
Else
RecordCountSet.Requery
If RecordCountSet.Fields(0) > 0 Then
CurStockSet.MoveLast
Call FillText

```

```

Else
Clear_Box
MsgBox "All Item Deleted...", vbInformation, "Item Deleted"
End If
End If
Else
Clear_Box
MsgBox "All Item Deleted...", vbInformation, "Item Deleted"
End If
ElseIf Index = 3 Then
RecordCountSet.Requery
If RecordCountSet.Fields(0) > 0 Then
enable_disable (True)
ButtonDurumları (False)
cmdEntrySave.Enabled = False
End If
ElseIf Index = 4 Then
RecordCountSet.Requery
With CurStockSet
.CancelBatch
.CancelUpdate
.Requery
End With
If RecordCountSet.Fields(0) > 0 Then
CurStockSet.MoveFirst
Else
Clear_Box
End If
Call ButtonDurumları(True)
enable_disable (False)
cmdEntrySave.Enabled = True
End If :End Sub
Private Sub Command6_Click()
Refresh_combobox (1)

```



```
frmNewItemType.Show vbModal
```

```
End Sub
```

```
Private Sub Command7_Click()
```

```
Refresh_combobox (2)
```

```
If Len(Combo1.Text) <> 0 Then
```

```
frmAddItem.Show vbModal
```

```
Else
```

```
MsgBox "Select Item Type to add new item...", vbInformation, "Select Item Type ..."
```

```
Combo1.SetFocus
```

```
End If
```

```
End Sub
```

```
Private Sub DataGrid1_Click()
```

```
Call FillText
```

```
End Sub
```

```
Private Sub DataGrid1_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
If KeyCode = 27 Then
```

```
If Command1(0).Enabled = False Then
```

```
Dim x As Integer
```

```
x = MsgBox("Are you sure you want to Cancel updates.", vbQuestion Or vbYesNo,  
"Want to cancel ?")
```

```
If x = 6 Then
```

```
Call Command1_Click(4)
```

```
End If
```

```
Exit Sub
```

```
End If
```

```
Dim y As Integer
```

```
y = MsgBox("Are you Sure you want to Cancel Purchase Bill?", vbQuestion Or  
vbYesNo, "Want to Cancel Purchase Invoice ?")
```

```
If y = 6 Then
```

```
Unload Me
```

```
End If
```

ElseIf KeyCode = 116 Then

Call Command1_Click(3)

End If

End Sub

Private Sub Form_Load()

Dim DelCurStockSet As Recordset

If fSQLÇalıştır(gnBağlantı, DelCurStockSet, "select * from cur_stock") Then

While DelCurStockSet.EOF <> True

DelCurStockSet.Delete

DelCurStockSet.MoveNext

Wend

End If

Text2 = Date

Text3.Text = ""

Text4.Text = ""

Text5.Text = ""

Text6.Text = ""

If fSQLÇalıştır(gnBağlantı, Itemset, "select * from item_type") Then

If fSQLÇalıştır(gnBağlantı, StockSet, "select * from item_master") Then

If fSQLÇalıştır(gnBağlantı, grKayıt, "select * from purchase_master") Then

If fSQLÇalıştır(gnBağlantı, CurStockSet, "select * from cur_stock") Then

If fSQLÇalıştır(gnBağlantı, RecordCountSet, "select count(*) from cur_stock") Then

If fSQLÇalıştır(gnBağlantı, rs_cur_stock, "select * from quantity_of_item") Then

Set DataGrid1.DataSource = CurStockSet

Refresh_combobox (1)

Refresh_combobox (2)

Call enable_disable(False)

ButtonDurumları (True)

Dim LAST_ID As String

If grKayıt.EOF <> True Then

grKayıt.MoveLast

LAST_ID=Mid(grKayıt("invoiceno").Value,2,Len(grKayıt.Fields("invoiceno")))

Dim ID As Integer

```

ID = Val(LAST_ID)
ID = ID + 1
LAST_ID = ID
If Len(LAST_ID) = 1 Then
LAST_ID = "A000" & LAST_ID
ElseIf Len(LAST_ID) = 2 Then
LAST_ID = "A00" & LAST_ID
ElseIf Len(LAST_ID) = 3 Then
LAST_ID = "A0" & LAST_ID
ElseIf Len(LAST_ID) = 4 Then
LAST_ID = "A" & LAST_ID
End If
Text1.Text = LAST_ID
Else
Text1.Text = "A0001"
End If
End If
End If
End If
End If
End If
End If
End Sub

```

```

Public Function enable_disable(t As Boolean)
If t = True Then
Combo1.Enabled = True
Combo2.Enabled = True
Command6.Enabled = True
Command7.Enabled = True
Text3.Enabled = True
Text4.Enabled = True
Text6.Enabled = True
Text3.SetFocus

```


Else

Combo1.Enabled = False

Combo2.Enabled = False

Command6.Enabled = False

Command7.Enabled = False

Text3.Enabled = False

Text4.Enabled = False

Text6.Enabled = False

End If

End Function

Public Sub ButtonDurumları(t As Boolean)

If t = True Then

DataGrid1.Enabled = True

Command1(0).Enabled = True

Command1(1).Enabled = False

Dim CheckDataSet As Recordset

If fSQLÇalıştır(gnBağlantı, CheckDataSet, "select count(*) from cur_stock") Then

If CheckDataSet.Fields(0) > 0 Then

Command1(2).Enabled = True

Command1(3).Enabled = True

Else

Command1(2).Enabled = False

Command1(3).Enabled = False

End If

Command1(4).Enabled = False

CheckDataSet.Close

End If

ElseIf t = False Then

DataGrid1.Enabled = False

Command1(0).Enabled = False

Command1(1).Enabled = True

Command1(2).Enabled = False

Command1(3).Enabled = False

```
Command1(4).Enabled = True
```

```
End If
```

```
End Sub
```

```
Public Sub Clear_Box()
```

```
Refresh_combobox (1)
```

```
Refresh_combobox (2)
```

```
Text3.Text = Clear
```

```
Text4.Text = Clear
```

```
Text5.Text = Clear
```

```
Text6.Text = Clear
```

```
End Sub
```

```
Public Sub FillText()
```

```
Combo1.Text = CurStockSet.Fields("itemtype")
```

```
Combo2.Text = CurStockSet.Fields("itemname")
```

```
Text3.Text = CurStockSet.Fields("quantity")
```

```
Text4.Text = CurStockSet.Fields("price")
```

```
Text5.Text = CurStockSet.Fields("total")
```

```
Text6.Text = CurStockSet.Fields("description")
```

```
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
```

```
If cmdEntrySave.Enabled = False Then
```

```
Call Command1_Click(4)
```

```
End If
```

```
End Sub
```

```
Private Sub Text3_Change()
```

```
If Val(Text3.Text) <> 0 And Val(Text4.Text) <> 0 Then
```

```
Text5.Text = Val(Text3.Text) * Val(Text4.Text)
```

```
End If :End Sub
```

```
Private Sub Text4_Change()
```

```
If Val(Text3.Text) <> 0 And Val(Text4.Text) <> 0 Then
```

```
Text5.Text = Val(Text3.Text) * Val(Text4.Text)
```

```
End If
```

```
End Sub
```

```
Public Sub Refresh_combobox(Index As Integer)
```

```
If Index = 1 Then
```

```
Combo1.Clear
```

```
Itemset.Requery
```

```
While Itemset.EOF <> True
```

```
Combo1.additem Itemset.Fields("itemtype").Value
```

```
Itemset.MoveNext
```

```
Wend
```

```
ElseIf Index = 2 Then
```

```
Combo2.Clear
```

```
StockSet.Close
```

```
If fSQLÇalıştır(gnBağlantı, StockSet, "select * from item_master where itemtype =" &  
Combo1.Text & """) Then
```

```
While StockSet.EOF <> True
```

```
Combo2.additem StockSet.Fields("itemname")
```

```
StockSet.MoveNext
```

```
Wend
```

```
End If
```

```
End If
```

```
End Sub
```



4. Codes of Amount Paid Or Not Form

```
Private Sub Command1_Click()  
If Option1.Value = True Then  
If Len(Text1.Text) = 0 Then  
MsgBox "Enter Amount of Rs Paid ...", vbInformation, "Paid Amount not Found ..."  
Exit Sub  
End If  
If Val(Text1.Text) = 0 Then  
MsgBox "Paid Amount can not be Zero" & vbCrLf & "Select Amount Not paid Option  
, If Amount is not paid ...", vbCritical, "Zero Value not Allowed ..."  
Exit Sub  
End If  
Dim rs As Recordset  
If Check1.Value = 1 Then  
Dim x As Integer  
x = MsgBox("Are you sure this Transaction is Complete", vbQuestion Or vbYesNo,  
"Trasaction Completed ?")  
If x <> vbYes Then  
Check1.Value = 0  
Exit Sub  
End If  
If (Val(Label8.Caption) - Val(Text1.Text)) > 0 Then  
If Label4.Caption = "Sales" Then  
If fSQLÇalıştır(gnBağlantı, rs, "select * from expense") Then  
rs.AddNew  
rs.Fields("date").Value = Label9.Caption  
rs.Fields("expense_type").Value = "Discounted Amount(Sales)"  
rs.Fields("total_amount").Value = Val(Label8.Caption) - Val(Text1.Text)  
rs.Fields("invoiceno").Value = Label3.Caption  
rs.Update  
rs.Close  
End If  
ElseIf Label4.Caption = "Purchase" Then
```



```

If fSQLÇalıştır(gnBağlantı, rs, "select * from income") Then
rs.AddNew
rs.Fields("date").Value = Label9.Caption
rs.Fields("income_type").Value = "Discounted Amount(Purchase)"
rs.Fields("total_amount").Value = Val(Label8.Caption) - Val(Text1.Text)
rs.Fields("invoiceno").Value = Label3.Caption
rs.Update
rs.Close
End If
End If
If fSQLÇalıştır(gnBağlantı, rs, "select * from amount_unpaid_remind") Then
rs.AddNew
If Label4.Caption = "Sales" Then
rs.Fields("trans_type").Value = "SALES"
ElseIf Label4.Caption = "Purchase" Then
rs.Fields("trans_type").Value = "PURCHASE"
End If
rs.Fields("date") = Label9.Caption
rs.Fields("amount_unpaid") = Val(Label8.Caption) - Val(Text1.Text)
rs.Fields("invoiceno") = Label3.Caption
rs.Fields("customername") = Label11.Caption
rs.Update
End If
Unload Me
Exit Sub
End If
If fSQLÇalıştır(gnBağlantı, rs, "select * from amount_unpaid_remind") Then
rs.AddNew
If Label4.Caption = "Sales" Then
rs.Fields("trans_type").Value = "SALES"
ElseIf Label4.Caption = "Purchase" Then
rs.Fields("trans_type").Value = "PURCHASE"
End If
rs.Fields("date") = Label9.Caption

```

```

rs.Fields("amount_unpaid") = Val(Label8.Caption) - Val(Text1.Text)
rs.Fields("invoiceno") = Label3.Caption
rs.Fields("customername") = Label11.Caption
rs.Update
End If
End If
Unload Me
ElseIf Option2.Value = True Then
If fSQLÇalıştır(gnBağlantı, rs, "select * from amount_unpaid_remind") Then
rs.AddNew
If Label4.Caption = "Sales" Then
rs.Fields("trans_type").Value = "SALES"
ElseIf Label4.Caption = "Purchase" Then
rs.Fields("trans_type").Value = "PURCHASE"
End If
rs.Fields("date") = Label9.Caption
rs.Fields("amount_unpaid") = Val(Label8.Caption) - Val(Text1.Text)
rs.Fields("invoiceno") = Label3.Caption
rs.Fields("customername") = Label11.Caption
rs.Update
End If
Unload Me
End If
End Sub

```

```

Private Sub Form_Load()
Label6.Visible = False
Text1.Visible = False
Label7.Visible = False
Label8.Visible = False
Check1.Visible = False
Label9 = Date
End Sub

Private Sub Option1_Click()

```

```
Label6.Visible = True
Text1.Visible = True
Label7.Visible = True
Label8.Visible = True
Check1.Visible = True
End Sub
```

```
Private Sub Option2_Click()
```

```
Label6.Visible = False
```

```
Text1.Visible = False
```

```
Label7.Visible = False
```

```
Label8.Visible = False
```

```
Check1.Visible = False
```

```
End Sub
```

```
Private Sub Text1_Change()
```

```
If Val(Text1.Text) > Val(Label8.Caption) Then
```

```
MsgBox "You can not Enter Greater than the required value ...", vbCritical, "Enter  
Proper Value ..."
```

```
Text1.Text = Clear
```

```
End If
```

```
If Val(Text1.Text) = Val(Label8.Caption) Then
```

```
Check1.Value = 1
```

```
Check1.Enabled = False
```

```
End If
```

```
If Val(Text1.Text) <> Val(Label8.Caption) Then
```

```
Check1.Value = 0
```

```
Check1.Enabled = True
```

```
End If
```

```
End Sub
```

5. Codes of Available Stock Form

```
Dim current_stock As Recordset
Private Sub Form_Load()
If fSQLÇalıştır(gnBağlantı, current_stock, "select * from quantity_of_item") Then
If current_stock.EOF <> True Then
Set DataGrid1.DataSource = current_stock
End If
End If
End Sub
```

6. Codes of Critic Stock Level Form

```
Dim criticstock As Recordset
Private Sub Form_Load()
If fSQLÇalıştır(gnBağlantı, criticstock, "select itemtype,itemname,quantity from
quantity_of_item where quantity<10") Then
If criticstock.EOF <> True Then
Set DataGrid1.DataSource = criticstock
End If
End If
End Sub
```

7. Codes of Modify Item Type, Item Name, Price Form

```
Dim rsitem As New Recordset
Dim rsitem1 As New Recordset
Private Sub Command1_Click(Index As Integer)
If Index = 0 Then
If Len(Text1.Text) > 0 Then
If Command1(0).Caption = "Modify" Then
List1.Enabled = False
List2.Enabled = False
```



```

Command1(1).Enabled = False
Command1(0).Caption = "Save"
Text1.Enabled = True
Else
Dim modify_rs As Recordset
If fSQLÇalıştır(gnBağlantı, modify_rs, "select * from item_type where itemtype='" &
List1.List(List1.ListIndex) & "'") Then
modify_rs.Fields("itemtype") = Text1.Text
modify_rs.Update
End If
Text1.Enabled = True
List1.Enabled = True
List2.Enabled = True
Command1(0).Caption = "Modify"
Command1(1).Enabled = True
rsitem.Close
If fSQLÇalıştır(gnBağlantı, rsitem, "select * from item_type") Then
List1.Clear
List2.Clear
While rsitem.EOF <> True
List1.additem rsitem.Fields("itemtype")
rsitem.MoveNext
Wend
End If
MsgBox "Item Type Updated Successfully ...", vbInformation, "Item type Updated ..."
End If
End If
ElseIf Index = 1 Then
If Len(Text2.Text) > 0 Then
If Command1(1).Caption = "Modify" Then
List1.Enabled = False
List2.Enabled = False
Command1(0).Enabled = False
Command1(1).Caption = "Save"

```

```

Text2.Enabled = True
Text3.Enabled = True
Else
Dim update_itemname As Recordset
If fSQLÇalıştır(gnBağlantı, update_itemname, "select itemname,price from item_master
where itemname='" & List2.List(List2.ListIndex) & "'") Then
update_itemname.Fields("itemname") = Text2.Text
update_itemname.Fields("price") = Text3.Text
update_itemname.Update
gnBağlantı.Execute "update quantity_of_item set itemname='" & Text2.Text &
"',price='" & Text3.Text & "' where itemname='" & List2.List(List2.ListIndex) & "'"
End If
Text2.Enabled = False
Text3.Enabled = False
Command1(1).Caption = "Modify"
Command1(0).Enabled = True
List1.Enabled = True
List2.Enabled = True
Text2.Text = Clear
Text3.Text = Clear
List2.Clear
List1.ListIndex = -1
MsgBox "Item name Updated Successfully", vbInformation, "Item name updated ..."
End If
End If
End If
End Sub

```

```

Private Sub Form_Load()
Text1.Enabled = False
Text2.Enabled = False
Text3.Enabled = False
If fSQLÇalıştır(gnBağlantı, rsitem, "select * from item_type") Then
List1.Clear

```

```

List2.Clear
While rsitem.EOF <> True
List1.additem rsitem.Fields("itemtype")
rsitem.MoveNext
Wend
If fSQLÇalıştır(gnBağlantı, rsitem1, "select itemname,price from item_master where
itemtype='" & List1.List(0) & "'") Then
List2.Clear
While rsitem1.EOF <> True
List2.additem rsitem1.Fields("itemname")
List2.ItemData(List1.NewIndex) = rsitem1.Fields("itemno")
rsitem1.MoveNext
Wend
End If
End If
End Sub

Private Sub List1_Click()
Text1.Text = List1.List(List1.ListIndex)
rsitem.Close
If fSQLÇalıştır(gnBağlantı, rsitem, "select itemname from item_master where
itemtype='" & Text1.Text & "'") Then
List2.Clear
While rsitem.EOF <> True
List2.additem rsitem.Fields("itemname")
rsitem.MoveNext
Wend
End If
Text2 = Clear
Text3 = Clear
End Sub

Private Sub List2_Click()
Text2.Text = List2.List(List2.ListIndex)

```

```

rsitem1.Close
If fSQLÇalıştır(gnBağlantı, rsitem1, "select * from item_master where itemname=" &
Text2.Text & "") Then
Text3.Text = rsitem1.Fields("price")
End If
End Sub

```

8. Codes of Sales Data Enrty Form

```

Dim Itemset As Recordset
Dim CustomerSet As Recordset
Dim CurSalesSet As Recordset
Dim RsRecordCount As Recordset
Dim RsItemSet As Recordset
Dim RsItemType As Recordset
Dim Rs_Pur_Item As Recordset
Dim rs_itemtype As Recordset
Dim Invoice_Number As String
Dim DATAGRID1_CLICKED As Boolean
Private Sub cmdCompleteSale_Click()
Dim rs_update_item_master As Recordset
Dim rs_aps As Recordset
CurSalesSet.Requery
RsRecordCount.Requery
If RsRecordCount.Fields(0) > 0 Then
TotalTransactionAmount = TotalAmount("SALES")
While CurSalesSet.EOF <> True
If fSQLÇalıştır(gnBağlantı, rs_update_item_master, "select * from item_master
itemtype where itemtype=" & CurSalesSet.Fields("itemtype") & " and itemname=" &
CurSalesSet.Fields("itemname") & "") Then
If fSQLÇalıştır(gnBağlantı, rs_aps, "select * from quantity_of_item where itemtype=" &
&CurSalesSet.Fields("itemtype")&"anditemname="& CurSalesSet.Fields("itemname")
& " ") Then

```



```

grKayıt.AddNew
grKayıt.Fields("invoiceno") = Text1.Text
grKayıt.Fields("customername") = Combo3.Text
grKayıt.Fields("customerid") = Text2.Text
grKayıt.Fields("date") = Text4.Text
grKayıt.Fields("itemtype") = CurSalesSet.Fields("itemtype")
grKayıt.Fields("itemname") = CurSalesSet.Fields("itemname")
grKayıt.Fields("quantity") = CurSalesSet.Fields("quantity")
grKayıt.Fields("price") = CurSalesSet.Fields("price")
grKayıt.Fields("total") = CurSalesSet.Fields("total")
grKayıt.Fields("pur_invoiceno") = CurSalesSet.Fields("invoiceno")
rs_update_item_master.Fields("quantity") =
Val(rs_update_item_master.Fields("quantity")) - Val(CurSalesSet.Fields("quantity"))
rs_update_item_master.Update
rs_update_item_master.Close
End If
rs_aps.Fields("quantity")=Val(rs_aps.Fields("quantity"))-
Val(CurSalesSet.Fields("quantity"))
rs_aps.Fields("total") = Val(rs_aps.Fields("quantity")) * Val(rs_aps.Fields("price"))
rs_aps.Update
If rs_aps.Fields("quantity") = 0 Then
rs_aps.Delete
End If
rs_aps.Close
CurSalesSet.MoveNext
End If
Wend
grKayıt.Update
CurSalesSet.MoveFirst
While CurSalesSet.EOF <> True
CurSalesSet.Delete
CurSalesSet.MoveNext
Wend
frmPaidNotPaid.Label3.Caption = Text1.Text

```

frmPaidNotPaid.Label4.Caption = "Sales"

frmPaidNotPaid.Label8.Caption = TotalTransactionAmount

frmPaidNotPaid.Label11.Caption = Combo3.Text

Unload Me

frmPaidNotPaid.Show vbModal

End If

End Sub

Private Sub Combo1_Click()

Refresh_combobox (2)

End Sub

Private Sub Combo2_Click()

Rs_Pur_Item.Close

If fSQLÇalıştır(gnBağlantı, Rs_Pur_Item, "select * from quantity_of_item where itemtype='" & Combo1.Text & "' and itemname='" & Combo2.Text & "'") Then

If Rs_Pur_Item.EOF <> True Then

Text5.Text = Rs_Pur_Item.Fields("quantity")

Text7.Text = Rs_Pur_Item.Fields("price")

Rs_Pur_Item.MoveNext

End If

End If

End Sub

Private Sub Combo3_Click()

Dim cus_id_set As Recordset

If fSQLÇalıştır(gnBağlantı, cus_id_set, "select * from customer where customername='" & Combo3.Text & "'") Then

Text2.Text = cus_id_set.Fields("customerid")

End If

End Sub

Private Sub Command2_Click(Index As Integer)

If Index = 0 Then

```

cmdCompleteSale.Enabled = False
Button_Durumları (False)
EnableDisable (True)
CurSalesSet.AddNew
ClearBox
Text7.Enabled = False
ElseIf Index = 1 Then
If Text6.Text = "" Then
MsgBox "Please Enter The Quantity of Order ", vbInformation, "Please Enter"
Text6.SetFocus
Exit Sub
Else
With CurSalesSet
.Fields("itemtype") = Combo1.Text
.Fields("itemname") = Combo2.Text
.Fields("quantity") = Text6.Text
.Fields("price") = Text7.Text
.Fields("total") = Text8.Text
.Fields("invoiceno") = Invoice_Number
.Update
.UpdateBatch
End With
Set DataGrid1.DataSource = Nothing
CurSalesSet.Requery
Set DataGrid1.DataSource = CurSalesSet
EnableDisable (False)
Button_Durumları (True)
cmdCompleteSale.Enabled = True
End If
ElseIf Index = 2 Then
RsRecordCount.Requery
If RsRecordCount.Fields(0) > 0 Then
CurSalesSet.Delete
CurSalesSet.MoveNext

```

```

If CurSalesSet.EOF <> True Then
    Call FillText
Else
    RsRecordCount.Requery
    If RsRecordCount.Fields(0) > 0 Then
        CurSalesSet.MoveLast
        Call FillText
    Else
        ClearBox
        MsgBox "All Item Deleted...", vbInformation, "Item Deleted"
    End If
End If
Else
    ClearBox
    MsgBox "All Item Deleted...", vbInformation, "Item Deleted"
End If
ElseIf Index = 3 Then
    RsRecordCount.Requery
    If RsRecordCount.Fields(0) > 0 Then
        EnableDisable (True)
        Button_Durumlari (False)
        cmdCompleteSale.Enabled = False
    End If
ElseIf Index = 4 Then
    RsRecordCount.Requery
    With CurSalesSet
        .CancelBatch
        .CancelUpdate
        .Requery
    End With
    If RsRecordCount.Fields(0) > 0 Then
        CurSalesSet.MoveFirst
    Else
        ClearBox

```



```

End If
Call Button_Durumları(True)
EnableDisable (False)
cmdCompleteSale.Enabled = True
End If
End Sub

```

```

Private Sub DataGrid1_Click()
If KeyCode = 27 Then
If Command2(0).Enabled = False Then
Dim x As Integer
x = MsgBox("Are you sure you want to Cancel updates ...", vbQuestion Or vbYesNo,
"Want to cancel ?")
If x = 6 Then
Call Command2_Click(4)
End If
Exit Sub
End If
Dim y As Integer
y = MsgBox("Are you Sure you want to Cancel Purchase Bill?", vbQuestion Or
vbYesNo, "Want to Cancel Purchase Invoice ?")
If y = 6 Then
Unload Me
End If
ElseIf KeyCode = 116 Then
Call Command2_Click(3)
End If
End Sub

```

```

Private Sub Form_Load()
Text4 = Date
Text5.Text = ""
Text6.Text = ""

```

```

Text7.Text = ""
Text8.Text = ""
Dim DelCurSalesSet As Recordset
If fSQLÇalıştır(gnBağlantı, DelCurSalesSet, "select * from cur_sales") Then
While DelCurSalesSet.EOF <> True
DelCurSalesSet.Delete
DelCurSalesSet.MoveNext
Wend
End If
If fSQLÇalıştır(gnBağlantı, grKayıt, "select * from sales_master") Then
If fSQLÇalıştır(gnBağlantı, CustomerSet, "select * from customer") Then
If fSQLÇalıştır(gnBağlantı, CurSalesSet, "select * from cur_sales") Then
If fSQLÇalıştır(gnBağlantı, RsRecordCount, "select count(*) from cur_sales") Then
If fSQLÇalıştır(gnBağlantı, RsItemSet, "select * from item_master") Then
If fSQLÇalıştır(gnBağlantı, RsItemType, "select count(*) from item_type") Then
If fSQLÇalıştır(gnBağlantı, Rs_Pur_Item, "select * from quantity_of_item where
itemtype='" & Combo1.Text & "' and itemname='" & Combo2.Text & "'") Then
If fSQLÇalıştır(gnBağlantı, rs_itemtype, "select * from quantity_of_item") Then
Set DataGrid1.DataSource = CurSalesSet
ComboBoxGüncelle Combo3, gnBağlantı, CustomerSet, "select * from customer",
"customername", False
Refresh_combobox (1)
Refresh_combobox (2)
EnableDisable (False)
Button_Durumları (True)
DataGrid1.Enabled = False
If grKayıt.EOF <> True Then
grKayıt.MoveLast
Dim LAST_ID As String
LAST_ID=Mid(grKayıt.Fields("invoiceno"),2, Len(grKayıt.Fields("invoiceno").Value))
LAST_ID = Val(LAST_ID) + 1
If Len(LAST_ID) = 1 Then
LAST_ID = "S00000000" & LAST_ID
ElseIf Len(LAST_ID) = 2 Then

```

```

LAST_ID = "S0000000" & LAST_ID
ElseIf Len(LAST_ID) = 3 Then
LAST_ID = "S000000" & LAST_ID
ElseIf Len(LAST_ID) = 4 Then
LAST_ID = "S00000" & LAST_ID
ElseIf Len(LAST_ID) = 5 Then
LAST_ID = "S0000" & LAST_ID
ElseIf Len(LAST_ID) = 6 Then
LAST_ID = "S000" & LAST_ID
ElseIf Len(LAST_ID) = 7 Then
LAST_ID = "S00" & LAST_ID
ElseIf Len(LAST_ID) = 8 Then
LAST_ID = "S0" & LAST_ID
ElseIf Len(LAST_ID) = 9 Then
LAST_ID = "S" & LAST_ID
End If
Text1.Text = LAST_ID
Else
Text1.Text = "S0000000001"
End If
Text1.Enabled = False
End If
End If
End If
End If
End If
End If
End If
End If
End If
End Sub

Public Sub Button_Durumları(t As Boolean)
If t = True Then
DataGrid1.Enabled = True

```

```

Command2(0).Enabled = True
Command2(1).Enabled = False
Dim CheckData As Recordset
If fSQLÇalıştır(gnBağlantı, CheckData, "select count(*) from cur_sales") Then
If CheckData.Fields(0) > 0 Then
Command2(2).Enabled = True
Command2(3).Enabled = True
Else
Command2(2).Enabled = False
Command2(3).Enabled = False
End If
Command2(4).Enabled = False
CheckData.Close
End If
ElseIf t = False Then
DataGrid1.Enabled = False
Command2(0).Enabled = False
Command2(1).Enabled = True
Command2(2).Enabled = False
Command2(3).Enabled = False
Command2(4).Enabled = True
End If : End Sub

```

```

Public Sub EnableDisable(t As Boolean)

```

```

If t = True Then
Combo1.Enabled = True
Combo2.Enabled = True
ElseIf t = False Then
Combo1.Enabled = False
Combo2.Enabled = False
End If : End Sub

```

```

Public Sub ClearBox()

```

```

Refresh_combobox (1)

```



```
Refresh_combobox (2)
```

```
Text5.Text = Clear
```

```
Text6.Text = Clear
```

```
Text7.Text = Clear
```

```
Text8.Text = Clear
```

```
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
```

```
If cmdCompleteSale.Enabled = False Then
```

```
Call Command2_Click(4)
```

```
End If : End Sub
```

```
Private Sub Text6_Change()
```

```
If Val(Text6.Text) > Val(Text5.Text) Then
```

```
MsgBox "You can not Enter Greater than Available Quantity", vbInformation, "You  
can not enter"
```

```
Text6 = Clear
```

```
Else
```

```
Text8.Text = Val(Text6.Text) * Val(Text7.Text)
```

```
End If
```

```
End Sub
```

```
Public Sub FillText()
```

```
If Rs_Pur_Item.EOF <> True Then
```

```
Combo1.Text = Rs_Pur_Item.Fields("itemtype")
```

```
Combo2.Text = Rs_Pur_Item.Fields("itemname")
```

```
Text5.Text = Rs_Pur_Item.Fields("quantity")
```

```
Text7.Text = Rs_Pur_Item.Fields("price")
```

```
End If : End Sub
```

```
Public Sub Refresh_combobox(Index As Integer)
```

```
If Index = 1 Then
```

```
Combo1.Clear
```

```

rs_itemtype.Close
If fSQLÇalıştır(gnBağlantı, rs_itemtype, "select itemtype from quantity_of_item group
by itemtype") Then
While rs_itemtype.EOF <> True
Combo1.additem rs_itemtype.Fields("itemtype").Value
rs_itemtype.MoveNext
Wend
End If
ElseIf Index = 2 Then
Combo2.Clear
Rs_Pur_Item.Close
If fSQLÇalıştır(gnBağlantı, Rs_Pur_Item, "select * from quantity_of_item where
itemtype='" & Combo1.Text & "'") Then
While Rs_Pur_Item.EOF <> True
Combo2.additem Rs_Pur_Item.Fields("itemname")
Rs_Pur_Item.MoveNext
Wend
End If
End If
End Sub

```

9. Codes of Unpaid Sales Amount Form

```

Dim cus_name As Recordset
Dim rs_unpaid As Recordset
Dim rs1 As Recordset

Private Sub Combo1_Click()
'If Len(Combo1.Text) > 0 Then
Refresh_Grid
'End If
End Sub

Private Sub Command1_Click()
If Check1.Value = 1 Then

```

```

Dim x As Integer
x = MsgBox("Are you sure this Transaction is Complete", vbQuestion Or vbYesNo,
"Trasaction Completed ?")
If x <> 6 Then
Check1.Value = 0
Exit Sub
End If
If Check1.Enabled = True Then
Dim Add_Expense As Recordset
If fSQLÇalıştır(gnBağlantı, Add_Expense, "select * from expense") Then
Add_Expense.AddNew
Dim ld As Recordset
If fSQLÇalıştır(gnBağlantı, ld, "select * from amount_unpaid_remind where
trans_type='SALES' and invoiceno='" & Combo1.Text & "'") Then
ld.MoveLast
Add_Expense.Fields("date") = ld.Fields("date")
ld.Close
Add_Expense.Fields("expense_type") = "Discounted Amount(Sales)"
Add_Expense.Fields("total_amount") = Val(Text1.Text) - Val(Text2.Text)
Add_Expense.Update
Add_Expense.Close
Dim Del_Unpaid As Recordset
If fSQLÇalıştır(gnBağlantı, Del_Unpaid, "select * from amount_unpaid_remind where
trans_type='SALES' and invoiceno='" & Combo1.Text & "'") Then
While Del_Unpaid.EOF <> True
Del_Unpaid.Delete
Del_Unpaid.MoveNext
Wend
End If
End If
End If
End If
Unload Me
Exit Sub

```

```

ElseIf Check1.Value = 0 Then
Dim Up_Unpaid As Recordset
If fSQLÇalıştır(gnBağlantı, Up_Unpaid, "select * from amount_unpaid_remind where
trans_type='SALES' and invoiceno='" & Combol.Text & "'") Then
Dim total As Double
total = Val(Text2.Text)
If total < Up_Unpaid.Fields("amount_unpaid") Then
Up_Unpaid.Fields("amount_unpaid") = Up_Unpaid.Fields("amount_unpaid") - total
Up_Unpaid.Update
Unload Me
Exit Sub
End If
If total = Up_Unpaid.Fields("amount_unpaid") Then
Up_Unpaid.Delete
Unload Me
Exit Sub
End If
If total > Up_Unpaid.Fields("amount_unpaid") Then
Dim ta As Double
ta = total
Dim t As Boolean
t = True
While t = True
If ta >= Up_Unpaid.Fields("amount_unpaid") Then
ta = ta - Up_Unpaid.Fields("amount_unpaid")
Up_Unpaid.Delete
Up_Unpaid.MoveNext
Else
Up_Unpaid.Fields("amount_unpaid") = Up_Unpaid.Fields("amount_unpaid") - ta
Up_Unpaid.Update
t = False
End If
Wend
Unload Me

```


Exit Sub

End If

End If

End If

End Sub

Private Sub DataGrid1_Click()

Text1.Text = rs_unpaid.Fields("amount_unpaid")

End Sub

Private Sub Form_Load()

Text1.Text = ""

Text2.Text = ""

Text3.Text = ""

If fSQLÇalıştır(gnBağlantı, cus_name, "select distinct invoiceno from amount_unpaid_remind where trans_type='SALES'") Then

If cus_name.RecordCount = 0 Then

MsgBox "No Unpaid Customer Found", vbInformation, "No Unpaid Customer Found"

Unload Me

Exit Sub

End If

Combo1.Clear

While cus_name.EOF <> True

Combo1.AddItem cus_name.Fields("invoiceno")

cus_name.MoveNext

Wend

End If

DataGrid1.Enabled = True

End Sub

Public Sub Refresh_Grid()

If fSQLÇalıştır(gnBağlantı, rs_unpaid, "select date,invoiceno,amount_unpaid from grid_sales_unpaid where invoiceno= '" & Combo1.Text & "'") Then

Set DataGrid1.DataSource = rs_unpaid

End If

If fSQLÇalıştır(gnBağlantı, rs1, "select sumofamount_unpaid,customername from QRY_UNPAID_REPORT where invoiceno=" & Combo1.Text & "" and trans_type='SALES'") Then

Text1.Text = rs1.Fields("sumofamount_unpaid")

Text3.Text = rs1.Fields("customername")

End If

End Sub

10. Codes of Unpadi Purchase Form

Dim InvoiceSet As Recordset

Private Sub Combo1_Click()

If Len(Combo1.Text) > 0 Then

Refresh_Grid

End If

End Sub

Private Sub Command1_Click()

If Check1.Value = 1 Then

Dim x As Integer

x = MsgBox("Are you sure this Transaction is Complete", vbQuestion Or vbYesNo, "Trasaction Completed ?")

If x <> 6 Then

Check1.Value = 0

Exit Sub

End If

If Check1.Enabled = True Then

Dim Add_Expense As Recordset

If fSQLÇalıştır(gnBağlantı, Add_Expense, "select * from income") Then

Add_Expense.AddNew

Dim ld As Recordset

If fSQLÇalıştır(gnBağlantı, ld, "select * from amount_unpaid_remind where trans_type='PURCHASE' and invoiceno=" & Combo1.Text & """) Then

ld.MoveLast

```

Add_Expense.Fields("date") = Id.Fields("date")
Id.Close
Add_Expense.Fields("income_type") = "Discounted Amount(Purchase)"
Add_Expense.Fields("total_amount") = Val(Text1.Text) - Val(Text2.Text)
Add_Expense.Update
Add_Expense.Close
Dim Del_Unpaid As Recordset
If fSQLÇalıştır(gnBağlantı, Del_Unpaid, "select * from amount_unpaid_remind where
trans_type='PURCHASE' and invoiceno='" & Combo1.Text & "'"") Then
While Del_Unpaid.EOF <> True
Del_Unpaid.Delete
Del_Unpaid.MoveNext
Wend
End If
End If
End If
End If
Unload Me
Exit Sub
ElseIf Check1.Value = 0 Then
Dim Up_Unpaid As Recordset
If fSQLÇalıştır(gnBağlantı, Up_Unpaid, "select * from amount_unpaid_remind where
trans_type='PURCHASE' and invoiceno='" & Combo1.Text & "'"") Then
Dim total As Double
total = Val(Text2.Text)
If total < Up_Unpaid.Fields("amount_unpaid") Then
Up_Unpaid.Fields("amount_unpaid") = Up_Unpaid.Fields("amount_unpaid") - total
Up_Unpaid.Update
Unload Me
Exit Sub
End If
If total = Up_Unpaid.Fields("amount_unpaid") Then
Up_Unpaid.Delete
Unload Me

```

```

Exit Sub
End If
If total > Up_Unpaid.Fields("amount_unpaid") Then
Dim ta As Double
ta = total
Dim t As Boolean
t = True
While t = True
If ta >= Up_Unpaid.Fields("amount_unpaid") Then
ta = ta - Up_Unpaid.Fields("amount_unpaid")
Up_Unpaid.Delete
Up_Unpaid.MoveNext
Else
Up_Unpaid.Fields("amount_unpaid") = Up_Unpaid.Fields("amount_unpaid") - ta
Up_Unpaid.Update
t = False
End If
Wend
Unload Me
Exit Sub
End If
End If
End If
End Sub

```

```

Private Sub Form_Load()
Text1.Text = ""
Text2.Text = ""
If fSQLÇalıştır(gnBağlantı, InvoiceSet, "select distinct invoiceno from
amount_unpaid_remind where trans_type='PURCHASE'") Then
If InvoiceSet.RecordCount = 0 Then
MsgBox "No Unpaid Invoice Found", vbInformation, "No Unpaid Invoice Found"
Unload Me
Exit Sub

```



```

End If
Combo1.Clear
While InvoiceSet.EOF <> True
Combo1.additem InvoiceSet.Fields("invoiceno")
InvoiceSet.MoveNext
Wend
End If
DataGrid1.Enabled = True
End Sub

Public Sub Refresh_Grid()
Dim rs_unpaid As Recordset
If fSQLÇalıştır(gnBağlantı, rs_unpaid, "select date,amount_unpaid from
grid_purchase_unpaid where invoiceno=" & Combo1.Text & "") Then
Set DataGrid1.DataSource = rs_unpaid
End If
Dim rs1 As Recordset
If fSQLÇalıştır(gnBağlantı, rs1, "select sumofamount_unpaid from
QRY_UNPAID_REPORT where invoiceno=" & Combo1.Text & "" and
trans_type='PURCHASE'") Then
Text1.Text = rs1.Fields("sumofamount_unpaid")
End If
End Sub

```

11. Codes of Purchase View Form

```

Dim rs_pur As Recordset
Dim rs_data As Recordset
Private Sub Combo1_Click()
get_data
Text1.Text = rs_data.Fields("date")
End Sub

Private Sub Form_Load()

```

```

If fSQLÇalıştır(gnBağlantı, rs_pur, "select distinct invoiceno from purchase_master")
Then
If rs_pur.RecordCount = 0 Then
MsgBox "No Record Found", vbInformation, "No Record Found"
Unload Me
Exit Sub
End If
Combo1.Clear
While rs_pur.EOF <> True
Combo1.additem rs_pur.Fields("invoiceno")
rs_pur.MoveNext
Wend
End If
DataGrid1.Enabled = True
End Sub

```

```

Public Sub get_data()
If fSQLÇalıştır(gnBağlantı, rs_data, "select * from purchase_master where invoiceno
=" & Combo1.Text & " orderby invoiceno, itemtype, itemname, quantity, price, total,
description") Then
Set DataGrid1.DataSource = rs_data
End If
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
rs_data.Close
rs_pur.Close
End Sub

```

12. Codes of Sales View Form

```

Dim cust_names As New Recordset
Dim rs_data As New Recordset
Dim rsSalesRpt As Recordset

```

```

Private Sub Combo1_Click(Index As Integer)
If Index = 0 Then
Dim invoice As New ADODB.Recordset
If fSQLÇalıştır(gnBağlantı, invoice, "select distinct invoiceno from Sales_master where
customername="" & Combo1(0).Text & """) Then
Combo1(1).Clear
While invoice.EOF <> True
Combo1(1).additem invoice.Fields("invoiceno").Value
invoice.MoveNext
Wend
End If
ElseIf Index = 1 Then
GETDATA
Text1.Text = rs_data.Fields("date").Value
End If
End Sub

```

```

Private Sub Command1_Click()
On Error Resume Next
DataEnvironment1.rsCommand1.Close
DataEnvironment1.rsCommand1.Open "select * from sales_master where invoiceno=""
& Combo1(1).Text & ""
DataReport1.Refresh
DataReport1.Show
End Sub

```

```

Private Sub Form_Load()
If fSQLÇalıştır(gnBağlantı, cust_names, "select distinct customername from
Sales_master") Then
If fSQLÇalıştır(gnBağlantı, rsSalesRpt, "select * from sales_master") Then
If cust_names.RecordCount = 0 Then
MsgBox "No Record Found ...", vbInformation, "No Record Found ..."
Unload Me
Exit Sub

```

```

End If
Combo1(0).Clear
While cust_names.EOF <> True
Combo1(0).additem cust_names.Fields("customername")
cust_names.MoveNext
Wend
End If
End If
End Sub

```

```

Public Sub GETDATA()
If fSQLÇalıştır(gnBağlantı,rs_data, "select * from Sales_master where
customername=" & Combo1(0).Text & "and invoiceno=" & Combo1(1).Text & "")
Then
Set DataGrid1.DataSource = rs_data
End If
Text1.Text = rs_data.Fields("date")
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
rs_data.Close
End Sub

```

13. Codes of Customer Form

```

Private Sub Command1_Click()
With grKayıt
.AddNew
.Fields("customerid") = Text1.Text
.Fields("customername") = Text2.Text
.Fields("phone") = Text3.Text
.Fields("mobil") = Text4.Text
.Fields("fax") = Text5.Text
.Fields("address") = Text6.Text

```



```

.Update
End With
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text2.SetFocus
Unload Me
End Sub

```

```

Private Sub Form_Load()
If fSQLÇalıştır(grBağlantı, grKayıt, "select * from customer") Then
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
If grKayıt.EOF <> True Then
grKayıt.MoveLast
Dim LAST_ID As String
LAST_ID=Mid(grKayıt.Fields("customerid"),2,Len(grKayıt.Fields("customerid").
Value))
LAST_ID = Val(LAST_ID) + 1
If Len(LAST_ID) = 1 Then
LAST_ID = "C00000000" & LAST_ID
ElseIf Len(LAST_ID) = 2 Then
LAST_ID = "C0000000" & LAST_ID
ElseIf Len(LAST_ID) = 3 Then
LAST_ID = "C000000" & LAST_ID
ElseIf Len(LAST_ID) = 4 Then
LAST_ID = "C00000" & LAST_ID
ElseIf Len(LAST_ID) = 5 Then
LAST_ID = "C0000" & LAST_ID

```

```

ElseIf Len(LAST_ID) = 6 Then
LAST_ID = "C000" & LAST_ID
ElseIf Len(LAST_ID) = 7 Then
LAST_ID = "C00" & LAST_ID
ElseIf Len(LAST_ID) = 8 Then
LAST_ID = "C0" & LAST_ID
ElseIf Len(LAST_ID) = 9 Then
LAST_ID = "C" & LAST_ID
End If
Text1.Text = LAST_ID
Else
Text1.Text = "C0000000001"
End If
Text1.Enabled = False
End If
End Sub

```

14. Codes of Customer Detail Modify Form

```

Dim cust_rs As Recordset
Dim count_rs As Recordset
Private Sub Combo1_Click()
If Len(Combo1.Text) > 0 Then
Dim rs As Recordset
If fSQLÇalıştır(gnBağlantı, rs, "select * from customer where customername=" &
Combo1.Text & "'") Then
Text1.Text = rs.Fields("customerid")
Text2.Text = rs.Fields("customername")
Text3.Text = rs.Fields("phone")
Text4.Text = rs.Fields("fax")
Text5.Text = rs.Fields("mobil")
Text6.Text = rs.Fields("address")
End If
rs.Close

```

End If

End Sub

Private Sub Command1_Click()

Combo1.Enabled = False

Dim rs_cus As Recordset

If Command1.Caption = "Modify" Then

Text1.Enabled = False

Text2.Enabled = True

Text3.Enabled = True

Text4.Enabled = True

Text5.Enabled = True

Text6.Enabled = True

Command1.Caption = "Save"

Else

Text1.Enabled = False

Text2.Enabled = False

Text3.Enabled = False

Text4.Enabled = False

Text5.Enabled = False

Text6.Enabled = False

If fSQLÇalıştır(gnBağlantı, rs_cus, "select * from customer where customerid=" &
Text1.Text & """) Then

With rs_cus

.Fields("customerid") = Text1.Text

.Fields("customername") = Text2.Text

.Fields("phone") = Text3.Text

.Fields("fax") = Text4.Text

.Fields("mobil") = Text5.Text

.Fields("address") = Text6.Text

.Update

End With

On Error GoTo A1

Combo1.Enabled = True

```

gnBağlantı.Execute " update amount_unpaid_remind set customername='" &
Text2.Text & "' where customername='" & Combo1.Text & "' and trans_type='SALES'"
End If
rs_cus.Close
If fSQLÇalıştır(gnBağlantı, rs_cus, "select * from customer") Then
Combo1.Clear
rs_cus.Requery
While rs_cus.EOF <> True
Combo1.additem rs_cus.Fields("customername")
rs_cus.MoveNext
Wend
Combo1.Text = Text2.Text
Command1.Caption = "Modify"
End If
Exit Sub
A1:
rs_cus.CancelUpdate
MsgBox "Duplicate Customer Name", vbInformation
End If
End Sub

```

```

Private Sub Form_Activate()

```

```

Text1.Text = ""

```

```

Text2.Text = ""

```

```

Text3.Text = ""

```

```

Text4.Text = ""

```

```

Text5.Text = ""

```

```

Text6.Text = ""

```

```

Text1.Enabled = False

```

```

Text2.Enabled = False

```

```

Text3.Enabled = False

```

```

Text4.Enabled = False

```

```

Text5.Enabled = False

```

```

Text6.Enabled = False

```


End Sub

Private Sub Form_Load()

If fSQLÇalıştır(gnBağlantı, cust_rs, "select * from customer") Then

If fSQLÇalıştır(gnBağlantı, count_rs, "select count(*) from customer") Then

If count_rs.Fields(0) = 0 Then

MsgBox "No Customer Record Found", vbInformation, "No Customer Entry.."

Unload Me

Exit Sub

End If

While cust_rs.EOF <> True

Combo1.additem cust_rs.Fields("customername")

cust_rs.MoveNext

Wend

End If

End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

cust_rs.Close

count_rs.Close

End Sub

15. Codes of New User Form

Dim rs_user As New Recordset

Dim rs_pass As New Recordset

Private Sub Command1_Click()

Static i As Integer

If fSQLÇalıştır(gnBağlantı, rs_pass, "select * from pass_user where username='" + Text1.Text + "' and userpass='" + Text2.Text + "'") Then

If rs_pass.RecordCount > 0 Then

frmMainform.Enabled = True

```
frmUserLogon.Visible = False
```

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
Else
```

```
MsgBox "USER NAME OR PASSWORD WRONG", vbInformation, "ERROR"
```

```
i = i + 1
```

```
If i = 3 Then
```

```
MsgBox "Program will be shut down", vbInformation, "Shut Down"
```

```
End
```

```
End If
```

```
End If
```

```
End If
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
Frame2.Visible = True
```

```
End Sub
```

```
Private Sub Command3_Click()
```

```
End
```

```
End Sub
```

```
Private Sub Command4_Click()
```

```
If Text3 <> "" And Text4 <> "" Then
```

```
If Text4 = Text5 Then
```

```
If fSQLÇalıştır(gnBağlantı, rs_user, "select * from pass_user") Then
```

```
With rs_user
```

```
.AddNew
```

```
.Fields("username") = Text3.Text
```

```
.Fields("userpass") = Text5.Text
```

```
.Update
```

```
End With
```

```
Else
```

```
MsgBox "Wrong Confirm Password", vbCritical, "Wrong Confirm Password"
```

```

End If
End If
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
End If
Frame2.Visible = False
End Sub

```

```

Private Sub Form_Load()
    frmMainform.Show
    frmMainform.Enabled = False

```

```

Frame2.Visible = False
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
End Sub

```

16. Codes of Backup Database Form

```

Private WithEvents huffman As clsHuffman
Private Sub cmdBackup_Click()
    If Len(Text1.Text) = 0 Then
        MsgBox "Select Backup File Path and then Click Create Backup Button...",
            vbInformation, "Select Backup Path ..."
    End If
    Exit Sub
End If
Label3.Visible = True
Dim dest As String
dest = Text1.Text & "\Inventory_Backup.bk"
Dim OldTimer As Single

```

```

ProgressBar1.Visible = True
OldTimer = Timer
Call huffman.EncodeFile(App.Path & "\\StockData.mdb", dest)
ProgressBar1.Value = 0
Unload Me
Exit Sub
End Sub

```

```

Private Sub Command2_Click()
frmSelectPath.Show vbModal
End Sub

```

```

Private Sub Form_Load()
Set huffman = New clsHuffman
Label3.Visible = False
ProgressBar1.Visible = False

```

```

End Sub
Private Sub Huffman_Progress(Procent As Integer)
Label3.Caption = "Compressing Database"
ProgressBar1.Value = Procent
If ProgressBar1.Value = 100 Then
Label3.Caption = "Saving Compressed File ..."
End If
DoEvents
End Sub

```

Select Path Form Sourcecode

```

Private Sub Dir1_Change()
Label1(1).Caption = Dir1.Path
End Sub

```

```

Private Sub Drive1_Change()

```



```

On Error GoTo A1:
Dir1.Path = Drive1.Drive
Exit Sub
A1:
MsgBox "Drive Can not be Accessed ...", vbInformation, "Drive not Accessed ..."
Drive1.Drive = "c:"
End Sub

Private Sub Form_Load()
Drive1.Drive = "c:"
Label1(1).Caption = Dir1.Path
End Sub

Private Sub Command1_Click()
frmBackup.Text1.Text = Label1(1).Caption
Unload Me
End Sub

```

17. Codes of Restore Database Form

```

Private WithEvents huffman As clsHuffman
Private Sub cmdRestoreIt_Click()
If Len(Label2(1).Caption) = 0 Then
MsgBox "Select Backup File Path and then Click Restore Button...", vbInformation,
"Select Path ..."
Exit Sub
End If
Dim OldTimer As Single
ProgressBar1.Visible = True
Label3.Visible = True
OldTimer = Time
Call huffman.DecodeFile(Label2(1).Caption, App.Path & "\StockData.mdb")
ProgressBar1.Value = 0
Unload Me
Exit Sub

```

```

End Sub

Private Sub Dir1_Change()
Label2(1).Caption = ""
On Error GoTo A1:
File1.Path = Dir1.Path
Exit Sub
A1:
MsgBox "Folder Can not be Accessed ...", vbInformation, "Drive not Accessed ..."
Drive1.Drive = "c:"
End Sub

```

```

Private Sub Drive1_Change()
Label2(1).Caption = ""
On Error GoTo A1:
Dir1.Path = Drive1.Drive
Exit Sub
A1:
MsgBox "Drive Can not be Accessed ...", vbInformation, "Drive not Accessed ..."
Drive1.Drive = "c:"
End Sub

```

```

Private Sub File1_Click()
Label2(1).Caption = File1.Path & "\" & File1.Filename
End Sub

```

```

Private Sub Form_Load()
Label2(1).Caption = ""
Label3.Visible = False
ProgressBar1.Visible = False
Set huffman = New clsHuffman
Drive1.Drive = "c:"
End Sub

```

```

Private Sub Huffman_Progress(Procent As Integer)

```

```

Label3.Caption = "Uncompressing Database"
ProgressBar1.Value = Procent
If ProgressBar1.Value = 100 Then
Label3.Caption = "Restoring Uncompressed File ..."
End If
DoEvents
End Sub

```

MODULES

1. Araçlar Modülü

```

Option Explicit

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal
hwnd As Long, ByVal wParam As Long, ByVal lParam As Any) As
Long

Public gfSormadanÇık As Boolean
Public gfNesneClickleriÇalışabilirMi As Boolean
Public gfYöneticiHesabıAktifMi As Boolean
Public tYardımDosyası As String
Public gvCevap As VbMsgBoxResult

Public Function fBaşkaÇalışanVarMı() As Boolean
If App.PrevInstance Then
fBaşkaÇalışanVarMı = True
Else
fBaşkaÇalışanVarMı = False
End If
End Function

Public Function tDosyaBul(ptDosya As String) As String
tDosyaBul = Dir(ptDosya)
End Function

```

```

Public Function fDosyaKopyala(ptKaynak As String, ptHedef As String) As Boolean
On Error GoTo Hata
FileCopy ptKaynak, ptHedef
fDosyaKopyala = True
Exit Function
Hata:
fDosyaKopyala = False
End Function

```

```

Public Sub ListBoxGüncelle(ByRef plstListBox As ListBox, ByRef prKayıt As
Recordset, ptSQL As String, ptVeriAlanı1 As String, Optional ptVeriAlanı2 As String
= "", Optional ptAnahtarAlanı As String = "", Optional plSeçilecekElemanNo As Long
= -1)
Dim lTMP As Long
Dim tVeri As String
If fSQLÇalıştır(gnBağlantı, prKayıt, ptSQL) Then
If prKayıt.RecordCount > 0 Then
plstListBox.Clear
For lTMP = 1 To prKayıt.RecordCount
tVeri = prKayıt.Collect(ptVeriAlanı1)
If ptVeriAlanı2 <> "" Then
tVeri = tVeri & " " & prKayıt.Collect(ptVeriAlanı2)
End If
plstListBox.AddItem tVeri, lTMP - 1
If ptAnahtarAlanı <> "" Then
plstListBox.ItemData(lTMP - 1) = prKayıt.Collect(ptAnahtarAlanı)
End If
prKayıt.MoveNext
Next
If plSeçilecekElemanNo >= 0 Then
plstListBox.ListIndex = plSeçilecekElemanNo
Else
plstListBox.ListIndex = 0
End If

```


End If
End If
End Sub

```
Public Sub ComboBoxElemanSeç(ByRef pcboComboBox As ComboBox, Optional  
ptSeçilecekEleman As String = "")  
Dim lSayaç As Long  
If pcboComboBox.ListCount > 0 Then  
If ptSeçilecekEleman = "" Then  
pcboComboBox.ListIndex = 0  
Else  
For lSayaç = 0 To pcboComboBox.ListCount - 1  
pcboComboBox.ListIndex = lSayaç  
If pcboComboBox.Text = ptSeçilecekEleman Then  
Exit Sub  
End If  
Next  
pcboComboBox.ListIndex = 0  
End If  
End If  
End Sub
```

```
Public Sub ComboBoxGüncelle(ByRef pcboComboBox As ComboBox, pnBağlantı As  
Connection, ByRef prKayıt As Recordset, ptSQL As String, ptAlan As String,  
pfNesneClickleriÇalışabilirMi As Boolean)  
Dim lKayıtAdedi As Long, lSayaç As Long  
gfNesneClickleriÇalışabilirMi = pfNesneClickleriÇalışabilirMi  
pcboComboBox.Clear  
If fSQLÇalıştır(pnBağlantı, prKayıt, ptSQL) Then  
lKayıtAdedi = prKayıt.RecordCount  
If lKayıtAdedi > 0 Then  
For lSayaç = 1 To lKayıtAdedi  
pcboComboBox.AddItem prKayıt.Collect(ptAlan)  
prKayıt.MoveNext
```

Next

ComboBoxElemanSeç pcboComboBox

End If

End If

gfNesneClickleriÇalışabilirMi = True

End Sub

Public Sub ComboTemizle(ByRef pcboComboBox As ComboBox)

pcboComboBox.Text = ""

If pcboComboBox.ListCount > 0 Then

pcboComboBox.ListIndex = -1

End If

End Sub

Public Function fDahaÖncedenKayıtlıMı(pnBağlantı As Connection, ByRef prKayıt As

Recordset, ptSQL As String) As Boolean

If fSQLÇalıştır(gnBağlantı, prKayıt, ptSQL) Then

If prKayıt.RecordCount = 0 Then

fDahaÖncedenKayıtlıMı = False

Else

fDahaÖncedenKayıtlıMı = True

End If

End If

End Function

Public Sub ListBoxaYeniElemanEkle(ByRef plstListBox As ListBox, ptEleman As
String, plAnahtar As Long, pfSeçilsinMi As Boolean)

Dim lTMP As Long

lTMP = plstListBox.ListCount

plstListBox.AddItem ptEleman, lTMP

plstListBox.ItemData(lTMP) = plAnahtar

If pfSeçilsinMi Then

plstListBox.ListIndex = lTMP

End If

End Sub

```

Public Function fListBoxElemanSil(ByRef plstListBox As ListBox, plEleman As Long)
    As Boolean
    fListBoxElemanSil = False
    If plstListBox.ListIndex >= 0 Then
        plstListBox.RemoveItem (plEleman)
        fListBoxElemanSil = True
    If plstListBox.ListCount > 0 Then
        If plEleman <= (plstListBox.ListCount - 1) Then
            plstListBox.ListIndex = plEleman
        Else
            plstListBox.ListIndex = plEleman - 1
        End If
    End If
    End If
End Function

```

```

Public Sub ListBoxaYatayKaydırmaÇubuğuEkle(ByRef plstListe As ListBox)
    Dim fTMP As Boolean
    fTMP = SendMessage(plstListe.hwnd, &H415, 2 * plstListe.Width /
        Screen.TwipsPerPixelX, 0)
End Sub

```

2. Veritabanı Modülü

```

Option Explicit
Public gtDosyaAdı As String
Public gtYedekDosyaAdı As String
Public gtSql As String
Public gnBağlantı As Connection
Public grKayıt As Recordset
Public grYedekKayıt As Recordset

```

```

Public Function fVeriTabanıAç(ptDosyaAdı As String, ByRef pnBağlantı As
Connection, ByRef prKayıt As Recordset, ptSQL As String) As Boolean
On Error GoTo Hata
Set pnBağlantı = New Connection
pnBağlantı.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & ptDosyaAdı &
";Persist Security Info=False"
If fSQLÇalıştır(pnBağlantı, prKayıt, ptSQL) = True Then
fVeriTabanıAç = True
Else
fVeriTabanıAç = False
End If
Exit Function
Hata:
fVeriTabanıAç = False
End Function

```

```

Public Function fVeriTabanıKapat(ByRef pnBağlantı As Connection, ByRef prKayıt
As Recordset) As Boolean
On Error GoTo Hata
prKayıt.Close
pnBağlantı.Close
Set pnBağlantı = Nothing
Set prKayıt = Nothing
fVeriTabanıKapat = True
fVeriTabanıYedekle gtDosyaAdı, gtYedekDosyaAdı
Exit Function
Hata:
fVeriTabanıKapat = False
End Function

```

```

Public Function fVeriTabanıYedekle(ptDosyaAdı As String, ptYedekDosyaAdı As
String) As Boolean
If fDosyaKopyala(ptDosyaAdı, ptYedekDosyaAdı) Then
fVeriTabanıYedekle = True

```


Else

fVeriTabanıYedekle = False

End If

End Function

Public Function fSQLÇalıştır(pnBağlantı As Connection, ByRef prKayıt As Recordset,
ptSQL As String) As Boolean

On Error GoTo Hata

Set prKayıt = New Recordset

prKayıt.CursorLocation = adUseClient

prKayıt.Open ptSQL, pnBağlantı, adOpenKeyset, adLockOptimistic

lKayıtElemanSayısı prKayıt

fSQLÇalıştır = True

Exit Function

Hata:

fSQLÇalıştır = False

End Function

Public Sub lKayıtElemanSayısı(ByRef prKayıt As Recordset)

If Not prKayıt.EOF Then

prKayıt.MoveLast

prKayıt.MoveFirst

End If

End Sub

3. Module 1

Dim LAST_ID As Integer

Public Sub VeriTabanıAç()

gtDosyaAdı = "stockdata.mdb"

If fVeriTabanıAç(gtDosyaAdı, gnBağlantı, grKayıt, "select * from customer") Then

' veri tabanı açıldı

Else

MsgBox "veritabanı dosyası bulunamadı", vbCritical

End If

End Sub

Public Function TotalAmount(TransactionType As String) As Double

Dim TotalSet As New Recordset

Dim t As Double

t = 0

If TransactionType = "SALES" Then

If fSQLÇalıştır(gnBağlantı, TotalSet, "select * from cur_sales") Then

While TotalSet.EOF <> True

t = t + Val(TotalSet.Fields(4).Value)

TotalSet.MoveNext

Wend

TotalAmount = t

End If

ElseIf TransactionType = "PURCHASE" Then

If fSQLÇalıştır(gnBağlantı, TotalSet, "select * from cur_stock") Then

While TotalSet.EOF <> True

t = t + Val(TotalSet.Fields(4).Value)

TotalSet.MoveNext

Wend

TotalAmount = t

End If

End If

End Function

Public Function SALES_INVOICE_NUMBER() As String

Dim ino As Recordset

Dim INO_COUNT As Recordset

If fSQLÇalıştır(gnBağlantı, ino, "select * from sorted_invoiceno") Then

If fSQLÇalıştır(gnBağlantı, INO_COUNT, "select count(*) from sorted_invoiceno")

Then

If INO_COUNT.Fields(0).Value = 0 Then

SALES_INVOICE_NUMBER = "COMP0001"

```

Else
ino.MoveLast
While ino.BOF <> True
If Mid(ino.Fields(0).Value, 1, 4) <> "WITH" Then
Dim st As String
st = ino.Fields(0).Value
Dim N As Integer
N = Mid(st, 5, Len(ino.Fields(0).Value))
N = N + 1
Dim NO As String
NO = N
ino.Close
If Len(NO) = 1 Then
SALES_INVOICE_NUMBER = "COMP000" & NO
ElseIf Len(NO) = 2 Then
SALES_INVOICE_NUMBER = "COMP00" & NO
ElseIf Len(NO) = 3 Then
SALES_INVOICE_NUMBER = "COMP0" & NO
ElseIf Len(NO) = 4 Then
SALES_INVOICE_NUMBER = "COMP0" & NO
End If
Exit Function
End If
ino.MovePrevious
If ino.BOF = True Then
SALES_INVOICE_NUMBER = "COMP0001"
Exit Function
End If
Wend
End If
End If
End If
End Function

```

5. clsHuffman Class Module

Option Explicit

'Progress Values for the encoding routine

Private Const PROGRESS_CALCFREQUENCY = 7

Private Const PROGRESS_CALCCRC = 5

Private Const PROGRESS_ENCODING = 88

Private Const PROGRESS_DECODING = 89

Private Const PROGRESS_CHECKCRC = 11

'Events

Event Progress(Procent As Integer)

Private Type HUFFMANTREE

ParentNode As Integer

RightNode As Integer

LeftNode As Integer

Value As Integer

Weight As Long

End Type

Private Type ByteArray

Count As Byte

Data() As Byte

End Type

Private Declare Sub CopyMem Lib "kernel32" Alias "RtlMoveMemory" (Destination
As Any, Source As Any, ByVal Length As Long)

Public Sub EncodeFile(SourceFile As String, DestFile As String)

On Error GoTo errh

Dim ByteArray() As Byte

Dim Filenr As Integer

'Make sure the source file exists

If (Not FileExist(SourceFile)) Then


```

Err.Raise vbObjectError, "clsHuffman.EncodeFile()", "Source file does not exist"
End If
'Read the data from the sourcefile
FileNr = FreeFile
Open SourceFile For Binary As #FileNr
ReDim ByteArray(0 To LOF(FileNr) - 1)
Get #FileNr, , ByteArray()
Close #FileNr
'Compress the data
Call EncodeByte(ByteArray(), UBound(ByteArray) + 1)
If (FileExist(DestFile)) Then Kill DestFile
'Save the destination string
Open DestFile For Binary As #FileNr
Put #FileNr, , ByteArray()
Close #FileNr
Call MsgBox("Your database is now Backed up and saved." & vbCrLf & "Remember
To Back your database everyday", vbInformation)
Exit Sub
errh:
If Err.Number = 71 Then
Call MsgBox("There is no discette in drive A:" & vbCrLf & "Please insert a disk to
backup your data" & vbCrLf & Err.Description, vbExclamation)
Else
MsgBox Err.Number & vbCrLf & Err.Description
End If : End Sub
Public Sub DecodeFile(SourceFile As String, DestFile As String)
Dim ByteArray() As Byte
Dim FileNr As Integer
'Make sure the source file exists
If (Not FileExist(SourceFile)) Then
Err.Raise vbObjectError, "clsHuffman.DecodeFile()", "Source file does not exist"
End If
'Read the data from the sourcefile
FileNr = FreeFile

```

```

Open SourceFile For Binary As #Filenr
ReDim ByteArray(0 To LOF(Filenr) - 1)
Get #Filenr, , ByteArray()
Close #Filenr
'Uncompress the data
Call DecodeByte(ByteArray(), UBound(ByteArray) + 1)
'If (FileExist(DestFile)) Then Kill DestFile
Open DestFile For Binary As #Filenr
Put #Filenr, , ByteArray()
Close #Filenr
Dim f As New FileSystemObject
f.CopyFile DestFile, App.Path & "\StockData.mdb", True
' f.DeleteFile DestFile
End Sub

```

```

Public Sub EncodeByte(ByteArray() As Byte, ByteLen As Long)
Dim i As Long
Dim j As Long
Dim Char As Byte
Dim BitPos As Byte
Dim lNode1 As Long
Dim lNode2 As Long
Dim lNodes As Long
Dim lLength As Long
Dim Count As Integer
Dim lWeight1 As Long
Dim lWeight2 As Long
Dim Result() As Byte
Dim ByteValue As Byte
Dim ResultLen As Long
Dim Bytes As ByteArray
Dim NodesCount As Integer
Dim NewProgress As Integer
Dim CurrProgress As Integer

```

```

Dim BitValue(0 To 7) As Byte
Dim CharCount(0 To 255) As Long
Dim Nodes(0 To 511) As HUFFMANTREE
Dim CharValue(0 To 255) As ByteArray
If (ByteLen = 0) Then
ReDim Preserve ByteArray(0 To ByteLen + 3)
If (ByteLen > 0) Then
Call CopyMem(ByteArray(4), ByteArray(0), ByteLen)
End If
ByteArray(0) = 72 "H"
ByteArray(1) = 69 "E"
ByteArray(2) = 48 "0"
ByteArray(3) = 13 'vbCr
Exit Sub
End If
ReDim Result(0 To 522)
"HE3" & vbCr identification string
Result(0) = 72
Result(1) = 69
Result(2) = 51
Result(3) = 13
ResultLen = 4
'Count the frequency of each ASCII code
For i = 0 To (ByteLen - 1)
CharCount(ByteArray(i)) = CharCount(ByteArray(i)) + 1
If (i Mod 1000 = 0) Then
NewProgress = i / ByteLen * PROGRESS_CALCFREQUENCY
If (NewProgress <> CurrProgress) Then
CurrProgress = NewProgress
RaiseEvent Progress(CurrProgress)
End If
End If
Next
'Create a leaf for each character

```

```

For i = 0 To 255
If (CharCount(i) > 0) Then
With Nodes(NodesCount)
.Weight = CharCount(i)
.Value = i
.LeftNode = -1
.RightNode = -1
.ParentNode = -1
End With
NodesCount = NodesCount + 1
End If
Next
'Create the Huffman Tree
For INodes = NodesCount To 2 Step -1
'Get the two leafs with the smallest weights
INode1 = -1: INode2 = -1
For i = 0 To (NodesCount - 1)
If (Nodes(i).ParentNode = -1) Then
If (INode1 = -1) Then
IWeight1 = Nodes(i).Weight
INode1 = i
ElseIf (INode2 = -1) Then
IWeight2 = Nodes(i).Weight
INode2 = i
ElseIf (Nodes(i).Weight < IWeight1) Then
If (Nodes(i).Weight < IWeight2) Then
If (IWeight1 < IWeight2) Then
IWeight2 = Nodes(i).Weight
INode2 = i
Else
IWeight1 = Nodes(i).Weight
INode1 = i
End If
Else

```



```

lWeight1 = Nodes(i).Weight
lNode1 = i
End If
ElseIf (Nodes(i).Weight < lWeight2) Then
lWeight2 = Nodes(i).Weight
lNode2 = i
End If
End If
Next

'Create a new leaf
With Nodes(NodesCount)
.Weight = lWeight1 + lWeight2
.LeftNode = lNode1
.RightNode = lNode2
.ParentNode = -1
.Value = -1
End With

'Set the parentnodes of the two leafs
Nodes(lNode1).ParentNode = NodesCount
Nodes(lNode2).ParentNode = NodesCount

'Increase the node counter
NodesCount = NodesCount + 1
Next

ReDim Bytes.Data(0 To 255)
Call CreateBitSequences(Nodes(), NodesCount - 1, Bytes, CharValue)
For i = 0 To 255
If (CharCount(i) > 0) Then
lLength = lLength + CharValue(i).Count * CharCount(i)
End If
Next

lLength = IIf(lLength Mod 8 = 0, lLength \ 8, lLength \ 8 + 1)
If ((lLength = 0) Or (lLength > ByteLen)) Then
ReDim Preserve ByteArray(0 To ByteLen + 3)
Call CopyMem(ByteArray(4), ByteArray(0), ByteLen)

```

```

ByteArray(0) = 72
ByteArray(1) = 69
ByteArray(2) = 48
ByteArray(3) = 13
Exit Sub
End If
Char = 0
For i = 0 To (ByteLen - 1)
Char = Char Xor ByteArray(i)
If (i Mod 10000 = 0) Then
NewProgress      =      i      /      ByteLen      *      PROGRESS_CALC_CRC      +
PROGRESS_CALC_FREQUENCY
If (NewProgress <> CurrProgress) Then
CurrProgress = NewProgress
RaiseEvent Progress(CurrProgress)
End If
End If
Next
Result(ResultLen) = Char
ResultLen = ResultLen + 1
Call CopyMem(Result(ResultLen), ByteLen, 4)
ResultLen = ResultLen + 4
For i = 0 To 7
BitValue(i) = 2 ^ i
Next
Count = 0
For i = 0 To 255
If (CharValue(i).Count > 0) Then
Count = Count + 1
End If
Next
Call CopyMem(Result(ResultLen), Count, 2)
ResultLen = ResultLen + 2
Count = 0

```

```

For i = 0 To 255
If (CharValue(i).Count > 0) Then
Result(ResultLen) = i
ResultLen = ResultLen + 1
Result(ResultLen) = CharValue(i).Count
ResultLen = ResultLen + 1
Count = Count + 16 + CharValue(i).Count
End If
Next
ReDim Preserve Result(0 To ResultLen + Count \ 8)
BitPos = 0
ByteValue = 0
For i = 0 To 255
With CharValue(i)
If (.Count > 0) Then
For j = 0 To (.Count - 1)
If (.Data(j)) Then ByteValue = ByteValue + BitValue(BitPos)
BitPos = BitPos + 1
If (BitPos = 8) Then
Result(ResultLen) = ByteValue
ResultLen = ResultLen + 1
ByteValue = 0
BitPos = 0
End If
Next
End If
End With
Next
If (BitPos > 0) Then
Result(ResultLen) = ByteValue
ResultLen = ResultLen + 1
End If
ReDim Preserve Result(0 To ResultLen - 1 + lLength)
Char = 0

```

```

BitPos = 0
For i = 0 To (ByteLen - 1)
    With CharValue(ByteArray(i))
        For j = 0 To (.Count - 1)
            If (.Data(j) = 1) Then Char = Char + BitValue(BitPos)
            BitPos = BitPos + 1
            If (BitPos = 8) Then
                Result(ResultLen) = Char
                ResultLen = ResultLen + 1
                BitPos = 0
                Char = 0
            End If
        Next
    End With
    If (i Mod 10000 = 0) Then
        NewProgress = i / ByteLen * PROGRESS_ENCODING + PROGRESS_CALCCRC +
        PROGRESS_CALCFREQUENCY
        If (NewProgress <> CurrProgress) Then
            CurrProgress = NewProgress
            RaiseEvent Progress(CurrProgress)
        End If
    End If
Next
If (BitPos > 0) Then
    Result(ResultLen) = Char
    ResultLen = ResultLen + 1
End If
ReDim ByteArray(0 To ResultLen - 1)
Call CopyMem(ByteArray(0), Result(0), ResultLen)
If (CurrProgress <> 100) Then
    RaiseEvent Progress(100)
End If
End Sub

```



```

Public Function DecodeString(Text As String) As String
Dim ByteArray() As Byte
ByteArray() = StrConv(Text, vbFromUnicode)
Call DecodeByte(ByteArray, Len(Text))
DecodeString = StrConv(ByteArray(), vbUnicode)
End Function

```

```

Public Function EncodeString(Text As String) As String
Dim ByteArray() As Byte
ByteArray() = StrConv(Text, vbFromUnicode)
Call EncodeByte(ByteArray, Len(Text))
EncodeString = StrConv(ByteArray(), vbUnicode)
End Function

```

```

Private Sub CreateBitSequences(Nodes() As HUFFMANTREE, ByVal NodeIndex As
Integer, Bytes As ByteArray, CharValue() As ByteArray)
Dim NewBytes As ByteArray
If (Nodes(NodeIndex).Value > -1) Then
CharValue(Nodes(NodeIndex).Value) = Bytes
Exit Sub
End If
If (Nodes(NodeIndex).LeftNode > -1) Then
NewBytes = Bytes
NewBytes.Data(NewBytes.Count) = 0
NewBytes.Count = NewBytes.Count + 1
Call CreateBitSequences(Nodes(), Nodes(NodeIndex).LeftNode, NewBytes,
CharValue)
End If
If (Nodes(NodeIndex).RightNode > -1) Then
NewBytes = Bytes
NewBytes.Data(NewBytes.Count) = 1
NewBytes.Count = NewBytes.Count + 1
Call CreateBitSequences(Nodes(), Nodes(NodeIndex).RightNode, NewBytes, CharValue)
End If

```

```
End Sub
Private Function FileExist(Filename As String) As Boolean
On Error GoTo FileDoesNotExist
Call FileLen(Filename)
FileExist = True
Exit Function
FileDoesNotExist:
FileExist = False
End Function
```